EXHIBIT A

US007209911B2

(12) **United States Patent**
Boothby et al.

(10) **Patent No.: US 7,209,911 B2**
(45) **Date of Patent: Apr. 24, 2007**

(54) **SYNCHRONIZATION OF DATABASES USING FILTERS**

(75) Inventors: **David J. Boothby**, Nashua, NH (US); **David W. Morgan**, Derry, NH (US)

(73) Assignee: **Intellisync Corporation**, San Jose, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 471 days.

(21) Appl. No.: **09/776,452**

(22) Filed: **Feb. 2, 2001**

(65) **Prior Publication Data**

US 2001/0005849 A1 Jun. 28, 2001

Related U.S. Application Data

(63) Continuation of application No. 09/036,400, filed on Mar. 5, 1998, now Pat. No. 6,212,529, which is a continuation of application No. 08/748,645, filed on Nov. 13, 1996, now Pat. No. 6,141,664.

(51) **Int. Cl.**
G06F 17/30 (2006.01)

(52) **U.S. Cl.** ..... **707/2**; 707/4; 707/100; 707/101; 707/102; 707/103 R; 707/104.1

(58) **Field of Classification Search** ..... 707/8, 707/104.1, 10, 200, 201, 203, 2, 4, 100, 102, 707/103 R
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,807,182	A	2/1989	Queen	395/144
4,819,156	A	4/1989	DeLorme et al.	364/200
4,827,423	A	5/1989	Beasley et al.	364/200
4,866,611	A	9/1989	Cree et al.	395/600
4,875,159	A	10/1989	Cary et al.	395/619
4,956,809	A	9/1990	George et al.	395/601
4,980,844	A	12/1990	Demjanenko et al.	364/550
5,065,360	A	11/1991	Kelly	395/800
5,136,707	A	8/1992	Block et al.	395/600
5,142,619	A	8/1992	Webster, III	395/161
5,155,850	A	10/1992	Janis et al.	707/202
5,170,480	A	12/1992	Mohan et al.	395/600
5,187,787	A	2/1993	Skeen et al.	395/600
5,210,868	A	5/1993	Shimada et al.	395/615
5,228,116	A	7/1993	Harris et al.	395/54
5,237,678	A	8/1993	Kuechler et al.	395/600
5,251,151	A	10/1993	Demjanenko et al.	364/550
5,251,291	A	10/1993	Malcolm	395/161
5,261,045	A	11/1993	Scully et al.	395/161
5,261,094	A	11/1993	Everson et al.	395/617
5,272,628	A	12/1993	Koss	364/419.19
5,278,978	A	1/1994	Demers et al.	395/600
5,278,982	A	1/1994	Daniels et al.	707/202
5,283,887	A	2/1994	Zachery	359/500
5,293,627	A	3/1994	Kato et al.	395/550
5,301,313	A	4/1994	Terada et al.	395/600
5,315,709	A	5/1994	Alston, Jr. et al.	395/606
5,327,555	A	7/1994	Anderson	395/617
5,333,252	A	7/1994	Brewer, III et al.	395/767
5,333,265	A	7/1994	Orimo et al.	395/200
5,333,316	A	7/1994	Champagne et al.	395/600

(Continued)

OTHER PUBLICATIONS

U.S. Application No. 08/964,751, filed Nov. 5, 1997.
Alfieri, "The Best of WordPerfect Version 5.0," Hayden Books, pp. 153–165, 429–435 (1988).

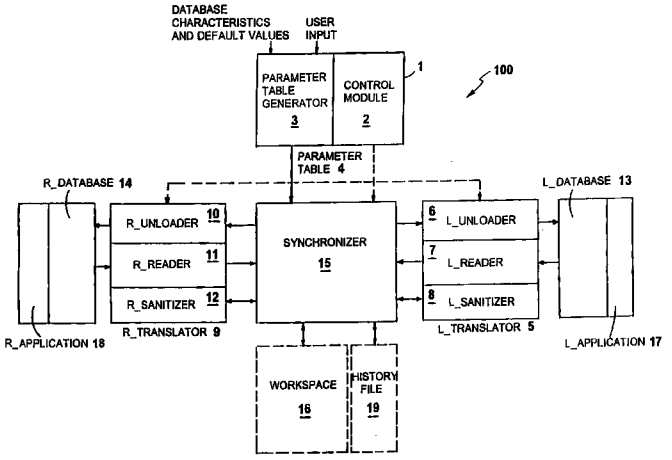
(Continued)

Primary Examiner—Frantz Coby
(74) *Attorney, Agent, or Firm*—Fish & Richardson P.C.

(57) **ABSTRACT**

A computer program is provided for synchronizing at least a first and a second database. A plurality of records of the first database fitting a selected criterion are identified. At least one of the identified records of the first database is then synchronized with a record of the second database. On a computer display, a record selection criteria input region may be displayed for a user to input the selected criterion.

14 Claims, 16 Drawing Sheets



U.S. PATENT DOCUMENTS

5,339,392	A	8/1994	Risberg et al.	395/161
5,339,434	A	8/1994	Rusis	395/700
5,355,476	A	10/1994	Fukumura	395/600
5,375,234	A	12/1994	Davidson et al.	395/600
5,392,390	A	2/1995	Crozier	395/335
5,396,612	A	3/1995	Huh et al.	395/575
5,434,994	A	7/1995	Shaheen et al.	395/617
5,444,851	A	8/1995	Woest	395/200.1
5,463,735	A	10/1995	Pascucci et al.	395/200.1
5,475,833	A	12/1995	Dauerer et al.	395/617
5,511,188	A	4/1996	Pascucci et al.	395/600
5,519,606	A	5/1996	Frid-Nielsen et al.	395/228
5,560,005	A	9/1996	Hoover et al.	395/600
5,568,402	A	10/1996	Gray et al.	364/514 C
5,583,793	A	12/1996	Gray et al.	364/514 C
5,600,834	A	2/1997	Howard	395/617
5,613,113	A	3/1997	Goldring	707/202
5,615,364	A	3/1997	Marks	395/617
5,619,689	A	4/1997	Kelly	395/617
5,630,081	A	5/1997	Rybicki et al.	395/948
5,666,530	A	9/1997	Clark et al.	395/617
5,666,553	A	9/1997	Crozier	395/803
5,682,524	A	10/1997	Freund et al.	395/605
5,684,984	A	11/1997	Jones et al.	395/610
5,684,990	A	11/1997	Boothby	395/619
5,689,706	A	11/1997	Rao et al.	
5,701,423	A	12/1997	Crozier	395/335
5,708,812	A	1/1998	Van Dyke et al.	395/712
5,708,840	A	1/1998	Kikinis et al.	395/800
5,710,922	A	1/1998	Alley et al.	395/617
5,727,202	A	3/1998	Kucala	395/610
5,729,735	A	3/1998	Meyering	395/610
5,745,712	A	4/1998	Turpin et al.	395/333
5,758,083	A	5/1998	Singh et al.	707/10
5,758,150	A	5/1998	Bell et al.	395/610
5,758,337	A	5/1998	Hammond	
5,758,355	A	5/1998	Buchanan	707/201
5,778,388	A	7/1998	Kawamura et al.	707/203
5,781,908	A	7/1998	Williams et al.	
5,790,789	A	8/1998	Suarez	395/200.32
5,813,013	A	9/1998	Shakib et al.	
5,845,293	A	12/1998	Veghte et al.	707/202
5,857,201	A	1/1999	Wright, Jr. et al.	
5,870,759	A	2/1999	Bauer et al.	707/201
5,870,765	A	2/1999	Bauer et al.	707/203
5,878,415	A	3/1999	Olds	707/9
5,884,323	A	3/1999	Hawkins et al.	707/201
5,884,324	A	3/1999	Cheng et al.	707/201
5,884,325	A	3/1999	Bauer et al.	707/201
5,897,640	A	4/1999	Veghte et al.	707/202
5,926,816	A	7/1999	Bauer et al.	
5,926,824	A	7/1999	Hashimoto et al.	707/520
5,943,676	A	8/1999	Boothby	707/201
5,978,813	A	11/1999	Foltz et al.	707/201
6,044,381	A	3/2000	Boothby et al.	707/201
6,141,664	A *	10/2000	Boothby	707/201
6,212,529	B1 *	4/2001	Boothby et al.	707/201

OTHER PUBLICATIONS

"Automatically Synchronized Objects," Research Disclosure #29261, p. 614 (Aug. 1988).

Cobb et al., "Paradox 3.5 Handbook 3rd Edition," Bantam, pp. 803–816 (1991).

"FRx Extends Reporting Power of Platinum Series: (IBM Desktop Software's Line of Accounting Software)," Doug Dayton, PC Week, v. 8, n. 5, p. 29(2) (Feb. 4, 1991).

IntelliLink Brochure (1990).

"Logical Connectivity: Applications, Requirements, Architecture, and Research Agenda," Stuart Madnick & Y. Richard Wang, MIT, Systems Sciences, 1991 Hawaii Int'l, vol. 1, IEEE (Jun. 1991).

"Open Network Computing—Technical Overview," Sun Technical Report, Microsystems, Inc., pp. 1–32 (1987).

Organizer Link II Operation Manual, Sharp Electronics Corporation, no date.

Bishop et al., "The Big Picture (Accessing information on remote data management system)", UNIX Review, v. 7, n. 8, p. 38(7), Aug. 1989.

User Manual for Connectivity Pack for the HP 95LX, Hewlett Packard Company (1991).

User Manual for PC–Link for the B.O.S.S. and the PC–Link for the B.O.S.S., Traveling Software, Inc. (1989).

Zahn et al., Network Computing Architecture, pp. 1–11; 19–31; 87–115; 117–133; 187–199; 201–209 (1990).

U.S. Application No. 08/749,926, filed Nov. 13, 1996.

Chapura, Inc., 3 Compare, http://www.chapura.com/3compare.html (1997), pp. 1–2.

Chapura, Inc., PilotMirror Features Page, http://www.chapura.com/features.html (1997), pp. 1–4.

Wiederhold et al., Consistency Control of Replicated Data in Federated Databases, IEEE, pp. 130–132 (Nov. 1990).

Bowen et al., Achieving Throughput and Functionality in a Common Architecture: The DataCycle Experiment, IEE, p. 178 (Dec. 1991).

Informix Guide to SQL Tutorial Version 7.1, Dec. 1994.

Lomet, D., Using timestamping to optimize two phase commit; Parallel and Distributed Information Systems, 1993, Proceeding of the Second International Conference, Jan. 20–22, 1993: pp. 48–55.

Oracle 7 Distributed Database Technology and Symmetric Replication, Oracle White Paper, Apr. 1995.

Oracle 7 Server Distributed Systems, vol. II: Replicated Data, Release 7.3, Feb. 1996.

Oracle 7™ Server SQL Manual Release 7.3, Feb. 1996.

Quaglia, F. et al., Grain Sensitive Event Scheduling in Time Warp Parallel Discrete Event Simulation, Fourteenth Workshop on Parallel Distributed Simulation, PADS 2000, May 28–31, 2000: pp. 173–180.

Salzberg, B., Timestamping After Commit; Procs. Of the Third Int. Conf. On Parallel and Distributed Information Systems, Sep., 28–30, 1994: pp. 160–167.

Zhang et al., Impact of Workload and System Parameters on Next Generation Cluster Scheduling Mechanisms, IEEE Trans. On Parallel and Distributed Systems, vol. 12, No. 9, Sep. 2001: pp. 967–985.

* cited by examiner

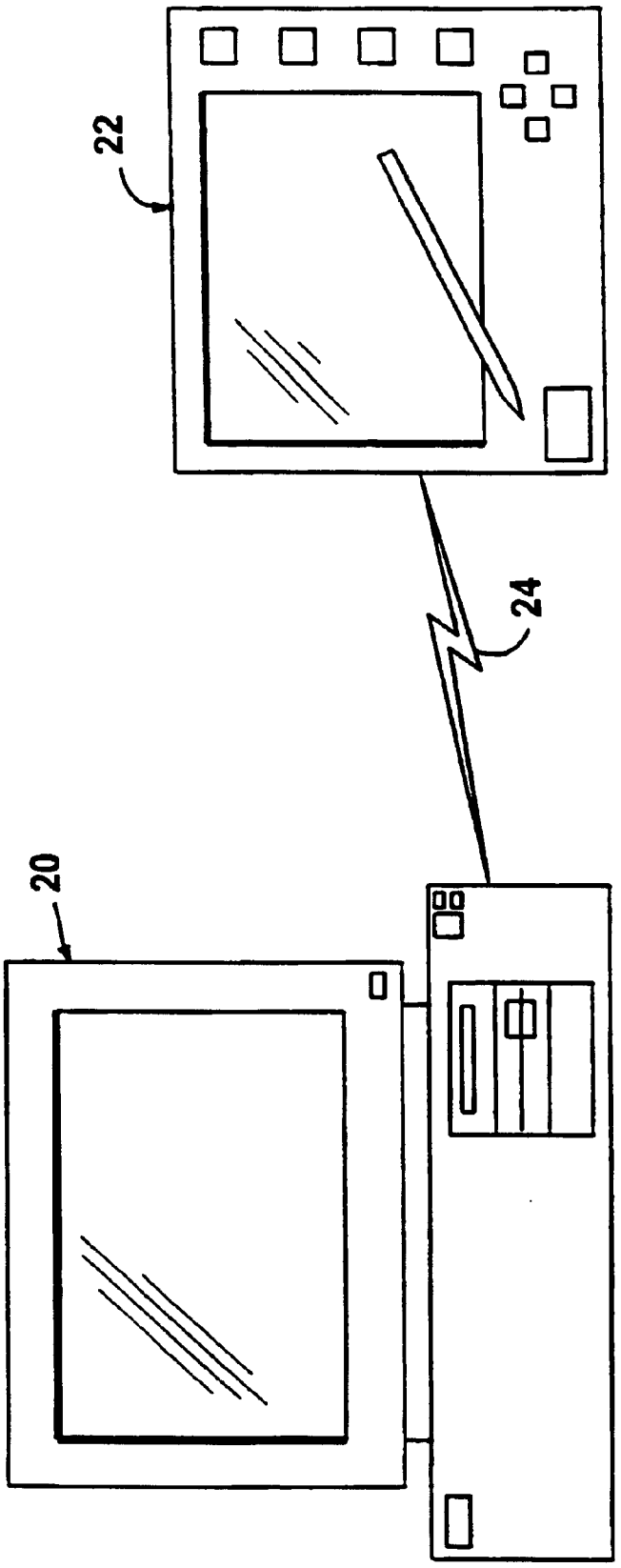


FIG. 1

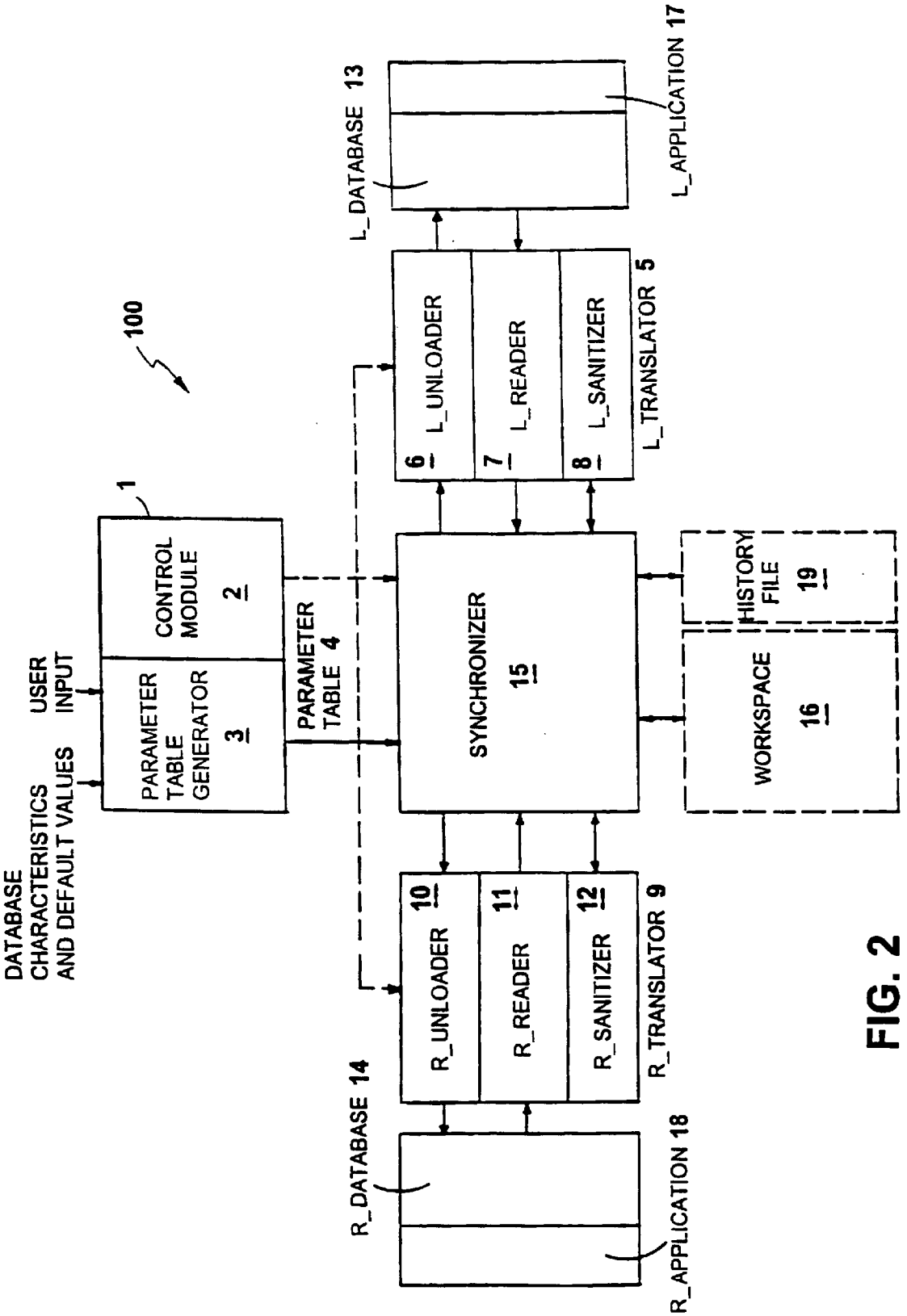

100
1
CONTROL MODULE 2
PARAMETER TABLE GENERATOR 3
USER INPUT
DATABASE CHARACTERISTICS AND DEFAULT VALUES
PARAMETER TABLE 4
L_DATABASE 13
L_APPLICATION 17
L_UNLOADER 6
L_READER 7
L_SANITIZER 8
L_TRANSLATOR 5
SYNCHRONIZER 15
HISTORY FILE 19
WORKSPACE 16
R_UNLOADER 10
R_READER 11
R_SANITIZER 12
R_TRANSLATOR 9
R_DATABASE 14
R_APPLICATION 18

FIG. 2

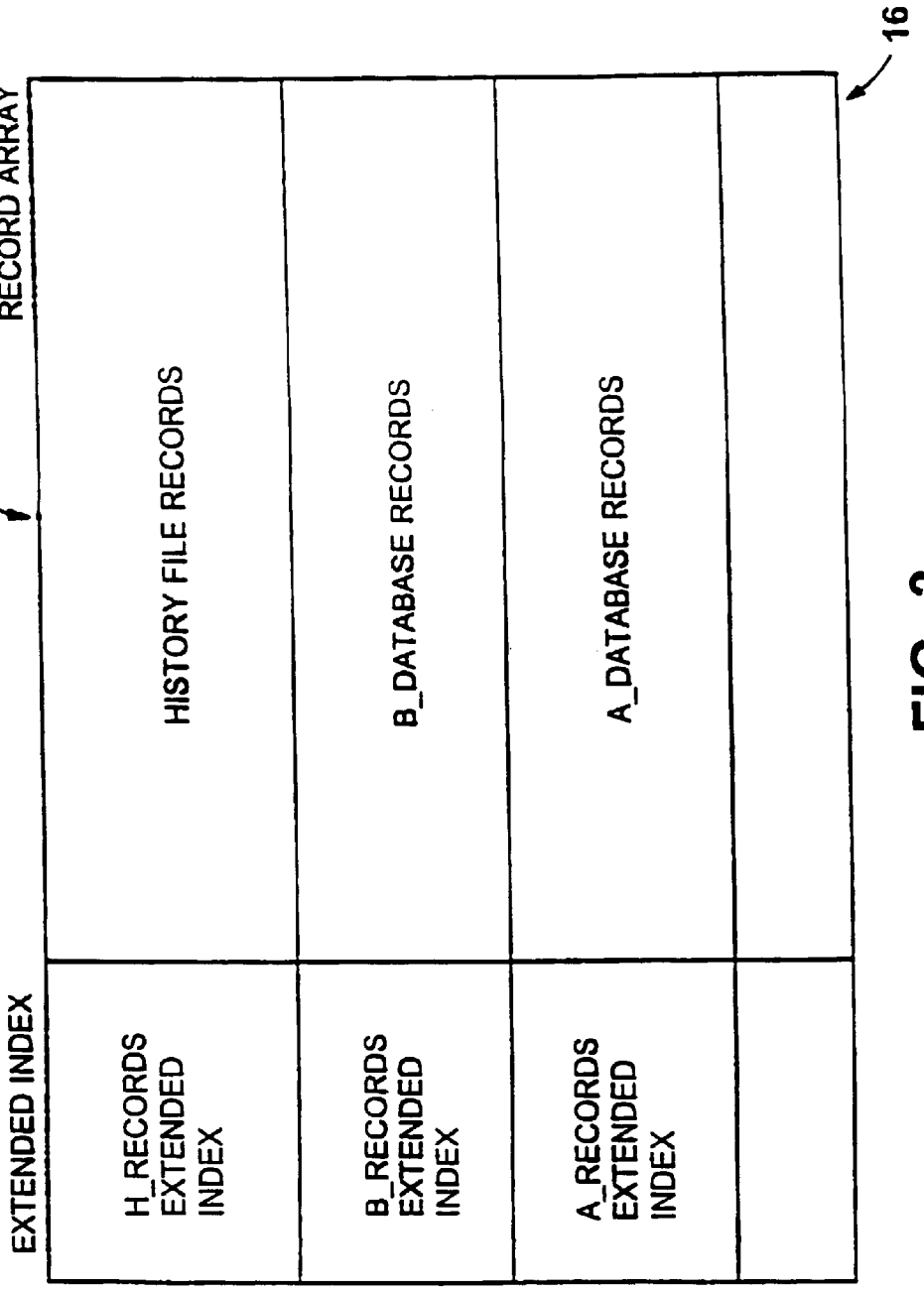

16
21
RECORD ARRAY
EXTENDED INDEX
HISTORY FILE RECORDS
B_DATABASE RECORDS
A_DATABASE RECORDS
H_RECORDS EXTENDED INDEX
B_RECORDS EXTENDED INDEX
A_RECORDS EXTENDED INDEX

FIG. 3

Pseudo Code for Translation Engine Control Module

100. INSTRUCT parameter table generator to create parameter table and initialize filter
101. INSTRUCT Synchronizer to initialize itself
102. INSTRUCT Synchronizer to LOAD the History_File into its WORKSPACE
103. INSTRUCT R_Translator to LOAD R_records from R_Database
104. INSTRUCT L_Translator to SANITIZE R_records that were just LOADED
105. INSTRUCT L_Translator to LOAD L_Records from L_Database and SEND to Synchronizer
106. INSTRUCT R_Translator to SANITIZE L_Records that were just LOADED.
107. INSTRUCT Synchronizer to do CAAR (Conflict Analysis And Resolution) on all the records in WORKSPACE.
108. INFORM user exactly what steps Synchronizer proposes to take (i.e. Adding, Changing, and Deleting records). WAIT for User.
109. IF User inputs NO, then ABORT.
110. INSTRUCT R_Translator to UNLOAD all applicable records to R_Database.
111. INSTRUCT L_Translator to UNLOAD all applicable records to L_Database.
112. INSTRUCT Synchronizer to CREATE a new History File.

FIG. 4

Pseudocode for Generating Parameter Table

```
{Get Input from the user}
150.    ASK user to select whether to use a filter expression
151.    IF the user selected to use a filter THEN
152.        IF a new filter to be used THEN
153.            Obtain from the user filter name
154.            Obtain filter expression
155.            STORE the current date and time in the FILTER_CHANGED_TIMESTAMP
                parameter
156.            Assign a unique filter ID to the filter
157.        ELSE Obtain from the user filter name
158.            retrieve the filter expression and unique filter ID
159.            IF user selects to edit the filter THEN display the filter and obtain edits
160.        SET FILTER_ID parameter to unique filter ID code of the selected filter
161.        SET USE_FILTER flag
162.        PARSE the filter expression into a filter token array
163.    END IF
164.    CREATE parameter table
```

FIG. 5

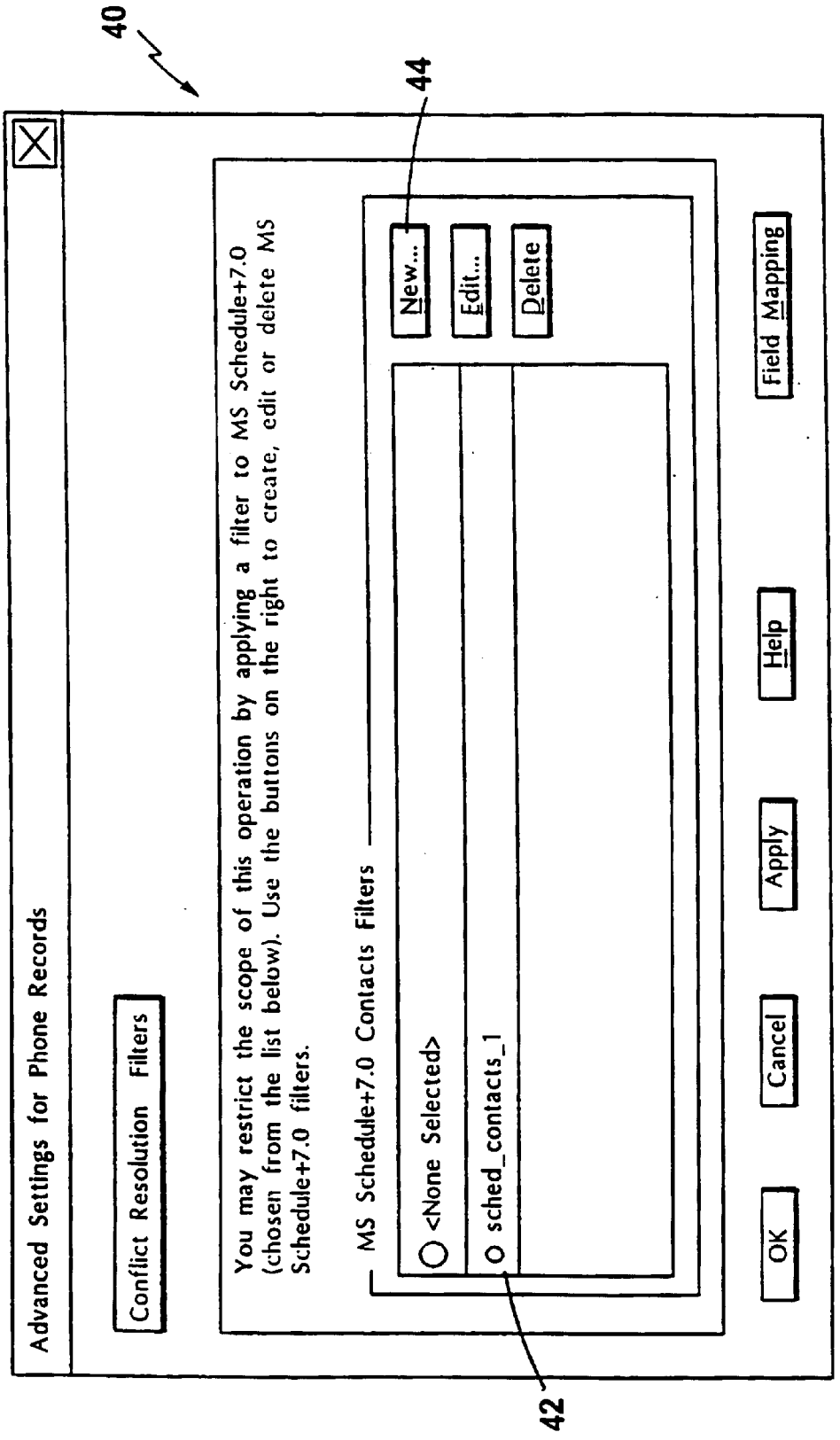

40
44
42
Advanced Settings for Phone Records
Conflict Resolution Filters
You may restrict the scope of this operation by applying a filter to MS Schedule+7.0 (chosen from the list below). Use the buttons on the right to create, edit or delete MS Schedule+7.0 filters.
MS Schedule+7.0 Contacts Filters
<None Selected>
sched_contacts_1
New...
Edit...
Delete
Field Mapping
OK
Cancel
Apply
Help

FIG. 6

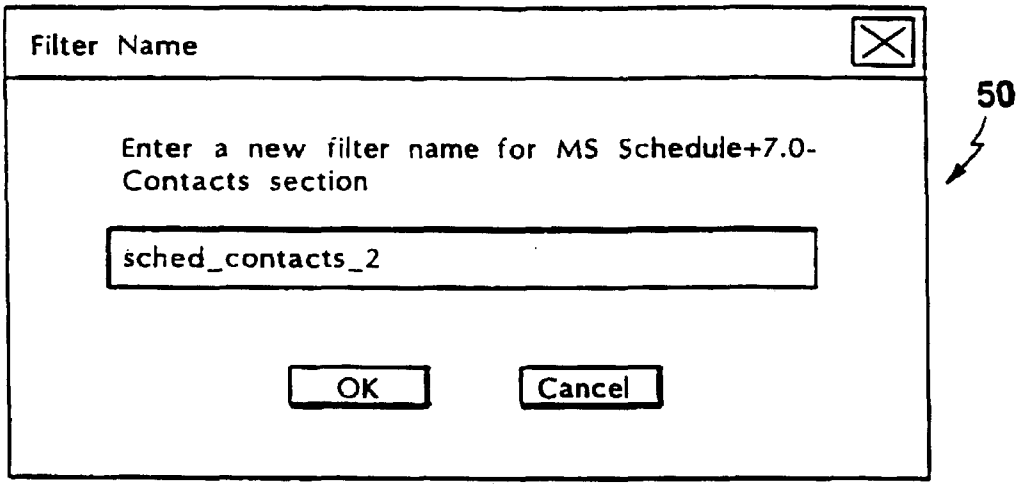

Filter Name
Enter a new filter name for MS Schedule+7.0-Contacts section
sched_contacts_2
OK
Cancel
50

FIG. 7

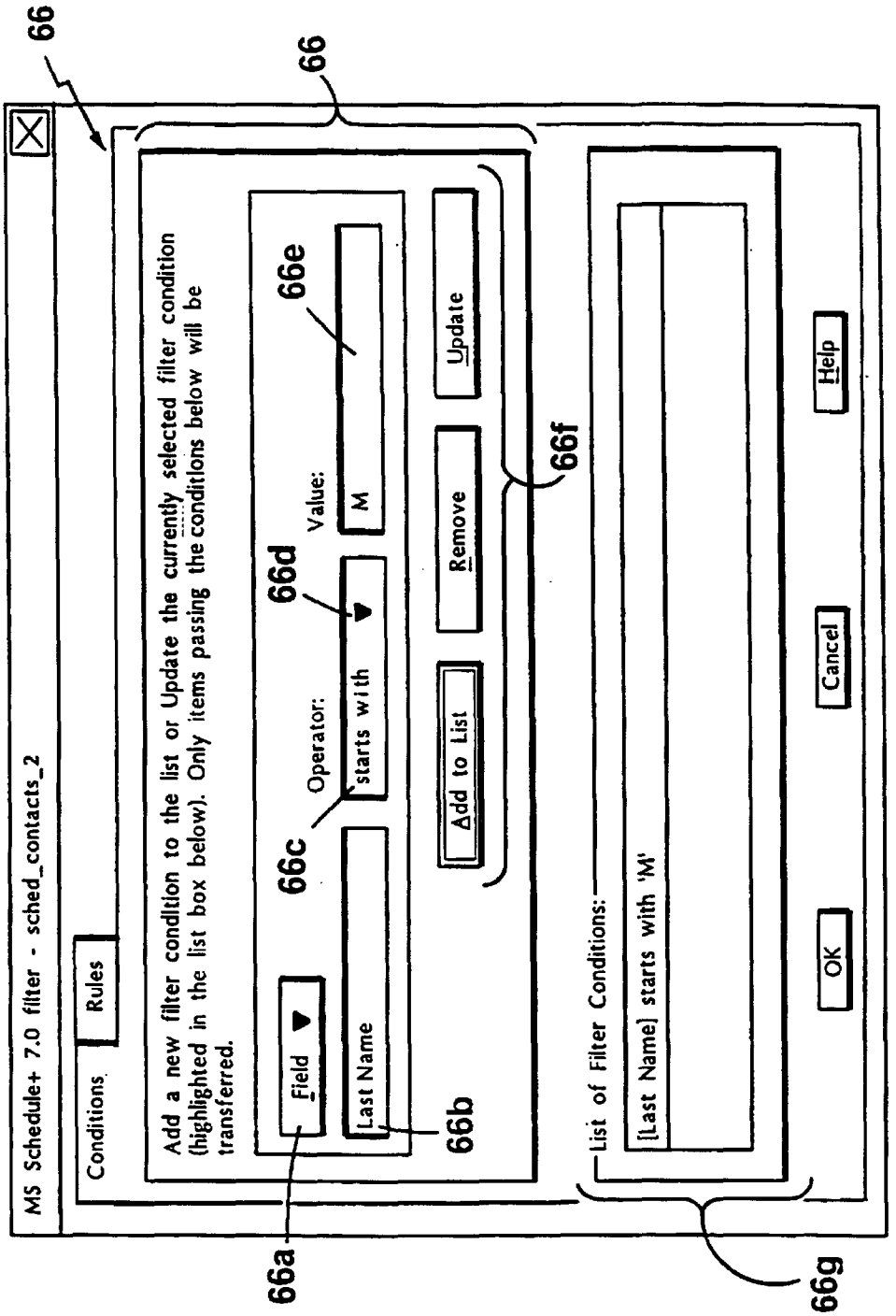

66
66
MS Schedule+ 7.0 filter - sched_contacts_2
Conditions
Rules
Add a new filter condition to the list or Update the currently selected filter condition (highlighted in the list box below). Only items passing the conditions below will be transferred.
Field
Last Name
66a
66b
66c
Operator:
starts with
66d
Value:
M
66e
Add to List
Remove
Update
66f
List of Filter Conditions:
[Last Name] starts with 'M'
OK
Cancel
Help
66g

FIG. 8

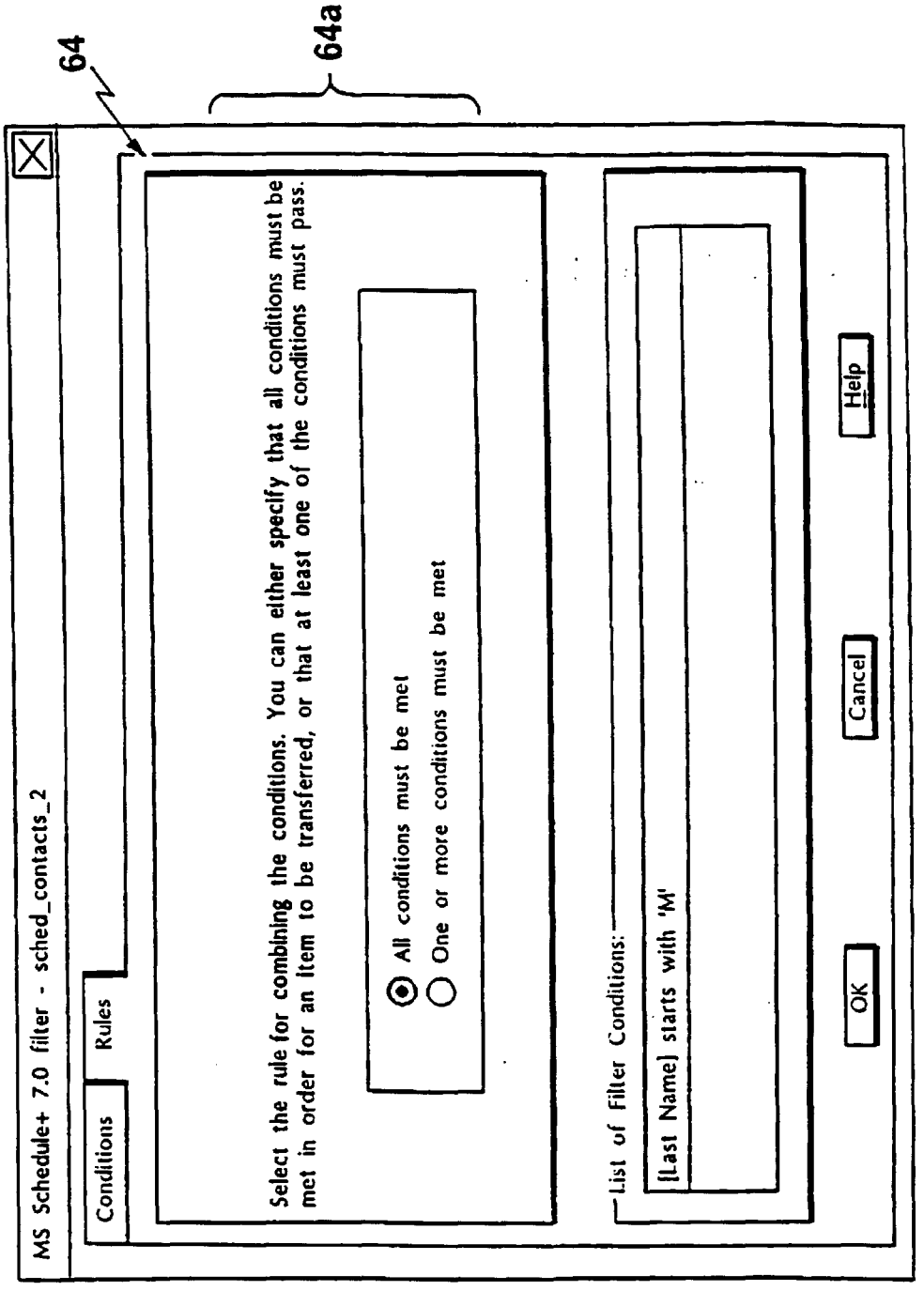

64
64a
MS Schedule+ 7.0 filter - sched_contacts_2
Conditions
Rules
Select the rule for combining the conditions. You can either specify that all conditions must be met in order for an item to be transferred, or that at least one of the conditions must pass.
All conditions must be met
One or more conditions must be met
List of Filter Conditions:
[Last Name] starts with 'M'
OK
Cancel
Help

FIG. 9

Filter language specification

Expression = Condition1 [AND Condition2] ... [OR Condition3] ...

Condition = ARG1 OP ARG2

OP = OP_SET_1 | OP_SET_2 | OP_SET_3 | OP_SET_4 | OP_SET_5 | OP_SET_6

OP_SET_1 = EQ | LE | GE | NE | LT | GT

OP_SET_2 = OP_SET_1 TODAY - | OP_SET_1 TODAY +

OP_SET_3 = OP_SET_1 NOW - | OP_SET_1 NOW +

OP_SET_4 = STARTS_WITH | CONTAINS | DOES_NOT_CONTAIN | IS_EMPTY | IS_NOT_EMPTY

*OP_SET_5 = + | - | * | / | %*

OP_SET_6 = IS

For Dates - ARG1 OP ARG2:

[Date Fieldname]	*OP_SET_1*	*'YYYYMMDD' | [Date Fieldname2] | TODAY*
[Date Fieldname]	*OP_SET_2*	*integer*

For Times - ARG1 OP ARG2

[Time Fieldname]	*OP_SET_1*	*'HHMM' | [Time Fieldname2] | NOW*
[Time Fieldname]	*OP_SET_3*	*integer*

For TextStrings - ARG1 OP ARG2

[String Fieldname]	*OP_SET_1*	*'textstring' | [String Fieldname2]*
[String Fieldname]	*OP_SET_4*	*'textstring'*

For Booleans - ARG1 OP ARG2

[Boolean Fieldname]	*OP_SET_6*	*TRUE*
[Boolean Fieldname]	*OP_SET_6*	*FALSE*

For Numbers - ARG1 OP ARG2

[Number Fieldname]	*OP_SET_1*	*integer | float*
[Number Fieldname]	*OP_SET_5*	*integer | float*

FIG. 10

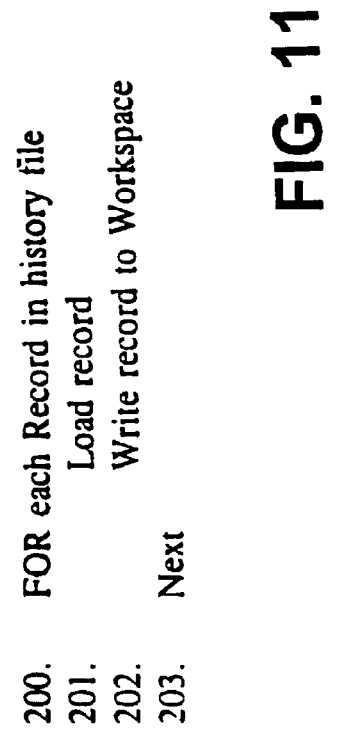

200. FOR each Record in history file
201. Load record
202. Write record to Workspace
203. Next

FIG. 11

```
300.  IF Use_Filter = TRUE and R_Application_Is_Filtering = FALSE THEN
301.        FOR each Record in the remote database
302.              Load record
303.              Filter the loaded record
304.              IF record passes the filter THEN mark as PASSED_FILTER
305.              ELSE mark as FAILED_FILTER
306.              Send record to synchronizer
307.              In Synchronizer: Write record to Workspace
308.        Next
309.  ELSE IF Use_Filter = TRUE and R_Application_Is_Filtering = TRUE THEN
310.        Send the filter expression to R_Application
311.        Load filtered records
312.        IF the record passes current filter THEN Mark as PASSED_FILTER ELSE Mark as
            FAILED_FILTER
313.        Send records to synchronizer
314.        In Synchronizer: Write records to Workspace
315.  END IF
```

FIG. 12

```
350.  Form all records in the workspace into CIGs
351.  For each CIG
352.        Compare the records in CIG
353.        Determine synchronization outcome
354.        IF a synchronization outcome is a conflict THEN
355.        IF one of the database records in the CIG does not pass the current filter, THEN skip CIG and
            mark results as DO NOT UPDATE any of the records
356.              ELSE resolve conflict by reference to a user-selected rule or input from the user
357.        END IF
358.        IF the most up to date record fails the filter, THEN mark all records as having failed the
            current filter
359.        IF the filter expressions contains an unmapped field and one of the database records in the CIG
            are marked as having failed the filter, THEN mark all records as having failed the filter
360.        IF a fanned out recurring record is partially outside of the current filter, THEN mark the
            record to be fanned when being unloaded and delete previous fanned nonrecurring records
361.  Next
```

FIG. 13

```
400.  FOR each remote database record
401.      IF Use_Filter = TRUE and the filter is a static filter THEN
402.          IF record is marked as FAILED_FILTER THEN
403.              Delete record on the remote database
404.          Else IF the record is marked as PASSED_FILTER THEN add, delete, or modify
              record according to results of synchronization obtained during CAAR analysis
405.      ELSE IF Use_Filter = TRUE and the filter is a dynamic filter THEN
406.          IF record fails the current filter THEN
407.              Delete record on the remote database
408.          Else IF the record passes the current filter THEN add, delete, or modify record
              according to results of synchronization obtained during CAAR analysis
409.      END IF
410.  Next
```

FIG. 14

```
450.  FOR each local database record
451.      IF Use_Filter = TRUE and the filter is a static filter THEN
452.          IF record is marked as FAILED_FILTER THEN
453.              IF CAAR outcome is to modify the record then modify the record on the
                  local database
454.          Else IF the record is marked as PASSED_FILTER THEN add, delete, or modify
              record according to results of synchronization obtained during CAAR analysis
455.      ELSE IF Use_Filter = TRUE and the filter is a dynamic filter THEN
456.          IF record fails the current filter but marked as PASSED_FILTER THEN
457.              IF CAAR outcome is to modify the record then modify the record on the
                  local database
458.          Else IF the record passes the current filter THEN add, delete, or modify record
              according to results of synchronization obtained during CAAR analysis
459.      END IF
460.  Next
```

FIG. 15

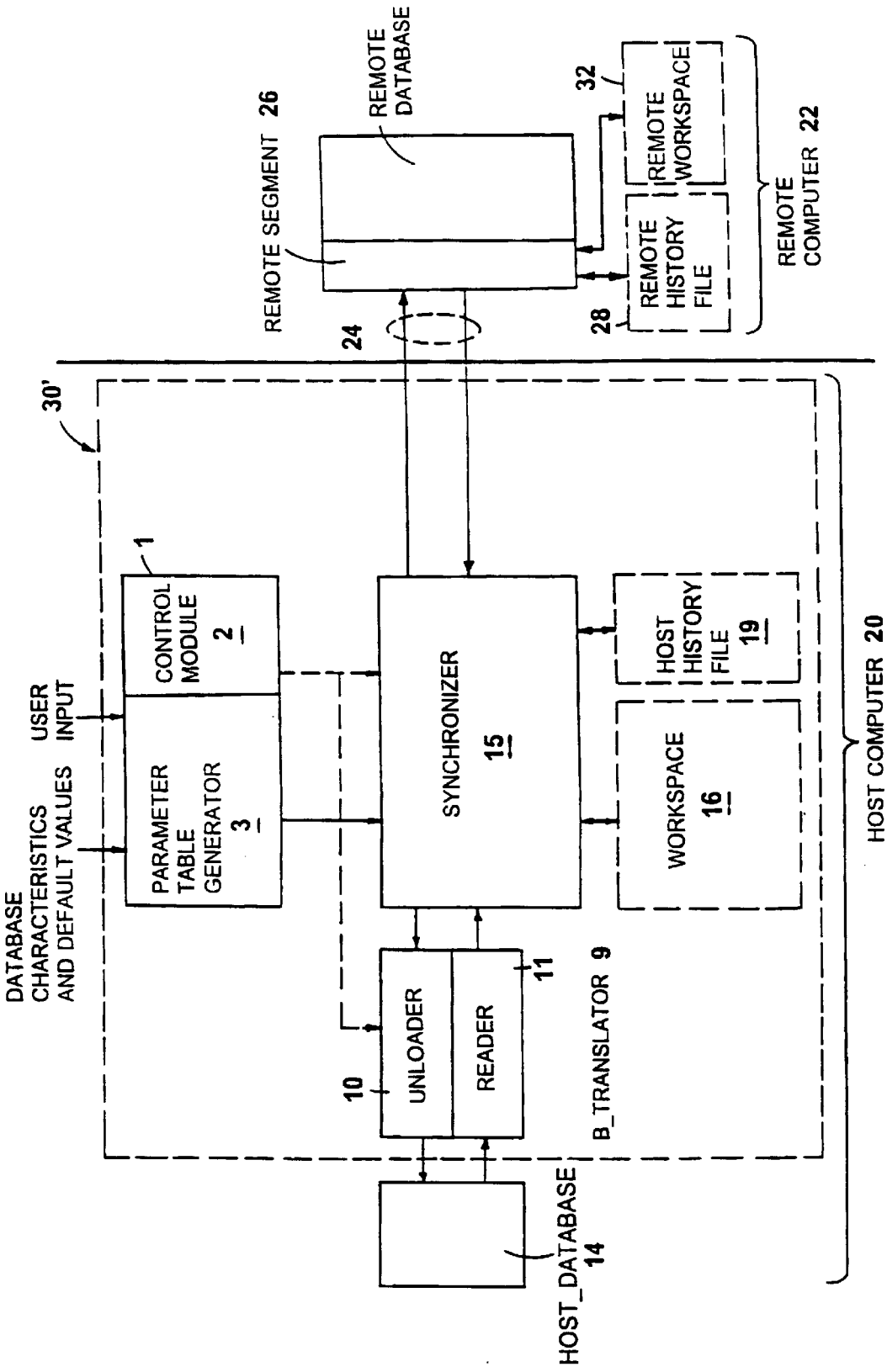

REMOTE SEGMENT 26
REMOTE DATABASE
32
REMOTE WORKSPACE
REMOTE HISTORY FILE
REMOTE COMPUTER 22
24
28
30'
1
CONTROL MODULE 2
PARAMETER TABLE GENERATOR 3
USER INPUT
DATABASE CHARACTERISTICS AND DEFAULT VALUES
SYNCHRONIZER 15
HOST HISTORY FILE 19
WORKSPACE 16
HOST COMPUTER 20
10
UNLOADER
READER
11
B_TRANSLATOR 9
HOST_DATABASE 14

FIG. 16

SYNCHRONIZATION OF DATABASES USING FILTERS

CROSS REFERENCE TO RELATED APPLICATIONS

This application is a continuation of U.S. application Ser. No. 09/036,400, filed Mar. 5, 1998 PCT/6,212,529 which is a continuation in part of "Synchronization of Databases with Date Range," Ser. No. 08/748,645, filed Nov. 13, 1996 now issued U.S. Pat No. 6,141,664.

REFERENCE TO MICROFICHE APPENDIX

An appendix forms part of this application. The appendix, which includes a source code listing relating to an embodiment of the invention, includes 1024 frames on 11 microfiche.

BACKGROUND

This invention relates to synchronizing databases.

Databases are collections of data entries which are organized, stored, and manipulated in a manner specified by applications known as database managers (hereinafter also referred to as "Applications"; hereinafter, the term "database" also refers to a database manager combined with a database proper). The manner in which database entries are organized in a database is known as the data structure of the database. There are generally two types of database managers. First are general purpose database managers in which the user determines (usually at the outset, but subject to future revisions) what the data structure is. These Applications often have their own programming language and provide great flexibility to the user. Second are special purpose database managers that are specifically designed to create and manage a database having a preset data structure. Examples of these special purpose database managers are various scheduling, diary, and contact manager applications for desktop and handheld computers. Database managers organize the information in a database into records, with each record made up of fields. Fields and records of a database may have many different characteristics depending on the database manager's purpose and utility.

Databases can be said to be incompatible with one another when the data structure of one is not the same as the data structure of another, even though some of the content of the records is substantially the same. For example, one database may store names and addresses in the following fields: FIRST_NAME, LAST_NAME, and ADDRESS. Another database may, however, store the same information with the following structure: NAME, STREET_NO., STREET_NAME, CITY_STATE, and ZIP. Although the content of the records is intended to contain the same kind of information, the organization of that information is completely different.

Often users of incompatible databases want to be able to synchronize them with one another. For example, in the context of scheduling and contact manager Applications, a person might use one application on a desktop computer at work while another on his handheld computer or his laptop computer while away from work. It is desirable for many of these users to be able to synchronize the entries on one with entries on another. U.S. patents of the assignee hereof, Puma Technology, Inc. of San Jose, Calif. (U.S. Pat. No. 5,392,390, hereinafter, "the '390 patent", incorporated by reference herein; and U.S. Pat. No. 5,684,990, filed on Jan. 11, 1995, incorporated by reference herein) show two methods for synchronizing incompatible databases and solving some of the problems arising from incompatibility of databases.

SUMMARY

In one general aspect, the invention features a computer program for synchronizing at least a first and a second database. A plurality of records of the first database fitting a selected criterion are identified. At least one of the identified records of the first database is then synchronized with a record of the second database.

In another general aspect, the invention features a computer program for synchronizing at least a first and a second database. On a computer display, a record selection criteria input region is displayed for a user to input a record selection criteria. Then, the first database is synchronized with the second database using the record selection criteria.

Preferred embodiments may include one or more of the following features.

Records representative of the records of the first and second databases during a prior synchronization are stored in a history file. In that case, when synchronizing the identified records of the first database with the records of the second database, the history file is used.

Records of the second database may also be identified based on a selected criterion. In that case, the identified records of the first database are synchronized with the identified records the second database.

Records of the first and second databases may include text, number, boolean, binary, date, and time fields. The criteria for identifying the records in turn may include text, number, boolean, binary, date and time criteria with which the fields of the records and databases are compared.

The first database can be located on a first computer and the second database located on a second computer. At the first computer, it is then determined whether a record of the first database has been changed or added since a previous synchronization, using a first history file located on the first computer including records representative of records of the first database at the completion of the previous synchronization. If the record of the first database has not been changed or added since the previous synchronization, information which the second computer uses to identify the record of the first database to be unchanged is sent from the first computer to the second computer. Additionally, the identification of the records of the first database based on the selected criterion may be performed at either the first or second computer.

Based on data reflecting whether the records of the first database have been added or changed since a previous synchronization, it may be determined whether the records of the first database have been changed or added since a previous synchronization. If one of the records of the first database has not been changed or added since the previous synchronization, a synchronization with records of the second database using a record representative of the one record at the time of a previous synchronization is performed. The representative record is stored in a history file containing records reflecting the contents of records of the databases at the time of a previous synchronization.

A second plurality of the records of the first database failing to fit the selected criterion may be deleted.

The selected criterion may have a current value during a current synchronization being different from a previous value during a previous synchronization. In that case, a plurality of records of the second database may be updated, based on results of the synchronization, where the plurality of records of the second database fit the previous value of the selected criterion but fail to fit the current value of the selected criterion.

A third database may be synchronized with the second database by identifying a plurality of records of a third database fitting a second selected criterion and synchronizing at least one of the identified records of the third database with a record of the second database. The record of the second database can include a code identifying the record as having originated from the third database.

A record selection criteria may be transmitted to a database manager which manages the first database and the database manager may select records of the first database fitting the record selection criteria. The database manager may then transmit the selected records to the synchronization program. The records of the first database fitting the record selection criteria may also be selected at a synchronization program.

The selected criterion may be, for example, a filter or filter expression which a record must match or fit in order for that record to pass the filter expression.

Embodiments of the invention may include one or more of the following advantages.

Users of various embodiments of the invention can use those embodiments to achieve a variety of ends. For example, handheld computers typically have limited storage capacity. Using the filtering capability of some embodiments, a user can limit the records stored on the handheld computer to only those records which fit a selected filter.

A user can also use a filter in some embodiments to increase the speed of synchronization. For example, it may be the case that the data transfer link between the two computers has a low data transfer rate. Therefore, the user by using a filter reduces the number of records to be transferred from one computer to the other.

A user can also use the filtering mechanism to synchronize some of the records of his or her database with the related records of another user's database, without affecting the unrelated records. For example, consider the situation where two database users work on a shared project or take a joint business trip on behalf of their enterprise. These two database users may desire to synchronize their databases but only with respect to those records relating to that project or trip. By using a filter, they may limit the synchronization between the two databases to records which relate to the project or the trip, without affecting other records in their respective databases.

The invention may be implemented in hardware or software, or a combination of both. Preferably, the technique is implemented in computer programs executing on programmable computers that each include a processor, a storage medium readable by the processor (including volatile and non-volatile memory and/or storage elements), at least one input device, and at least one output device. Program code is applied to data entered using the input device to perform the functions described above and to generate output information. The output information is applied to one or more output devices.

Each program is preferably implemented in a high level procedural or object oriented programming language to communicate with a computer system. However, the programs can be implemented in assembly or machine language, if desired. In any case, the language may be a compiled or interpreted language.

Each such computer program is preferably stored on a storage medium or device (e.g., ROM or magnetic diskette) that is readable by a general or special purpose programmable computer for configuring and operating the computer when the storage medium or device is read by the computer to perform the procedures described in this document. The system may also be considered to be implemented as a computer-readable storage medium, configured with a computer program, where the storage medium so configured causes a computer to operate in a specific and predefined manner.

Other features and advantages of the invention will become apparent from the following description of preferred embodiments, including the drawings, and from the claims.

BRIEF DESCRIPTION OF THE DRAWING

FIG. **1** shows two computers connected via a data transfer link.

FIG. **2** is a schematic drawing of the various modules constituting an embodiment of a synchronization program.

FIG. **3** is a representation of a workspace data array used by the synchronization program of FIG. **2**.

FIG. **4** is pseudocode for a Translation Engine Control Module of the synchronization program of FIG. **2**.

FIG. **5** is pseudo code for the steps taken by Parameter Table Generator module of the synchronization program of FIG. **2**.

FIG. **6** shows a filter selection graphical user interface (GUI) window.

FIG. **7** shows a filter name input graphical user interface (GUI) window.

FIGS. **8** and **9** show a filter criteria input graphical user interface (GUI) window.

FIG. **10** shows a table detailing semantics of a filter language used in the synchronization program of FIG. **2**.

FIG. **11** is pseudocode for loading a history file.

FIG. **12** is pseudocode for the steps taken by a translator to load records of a remote database.

FIG. **13** is pseudocode for the steps taken by the synchronizer module of the synchronization program of FIG. **2** for performing Conflict Analysis and Resolution when synchronizing using a filter expression.

FIG. **14** is pseudocode for the steps taken by a translator to unload records to a remote database.

FIG. **15** is pseudocode for the steps taken by a translator to unload records to a local database.

FIG. **16** is a schematic drawing of the various modules constituting an embodiment of a distributed synchronization program.

DESCRIPTION

We will describe embodiments of the invention in detail below, but briefly, referring to FIGS. **1** and **2**, a synchronization program **100** runs on a local computer **20** (e.g. a desktop or server computer) which is typically connected to a remote computer **22** (e.g. a handheld or notebook computer) via a data transfer link **24** enabling the computers to transfer data between them. Data transfer link **24** may be a serial infrared link, serial cable, modem and telephone line

combination, or other such data transfer links. Each of the local and remote computers stores a corresponding local or remote database, which may, for example, be a scheduling database (such as those sold under the tradenames Microsoft Schedule+ and Lotus Organizer).

Synchronization program **100** synchronizes the records of the local and remote databases typically using a history file that contains records reflecting the records of the two databases at the end of a previous synchronization. The synchronization program uses the history file to determine, for example, which records have been changed, added or deleted since the previous synchronization and which records of the two databases correspond to one another.

Synchronization program **100** allows the user to input a filter expression. Generally, a filter or filter expression may considered to be a set of conditions or criteria which a record must match or fit in order for that record to pass the filter expression. A record that does not fit those criteria therefore fails the filter.

Synchronization program **100** uses the filter expression to identify and mark which records of the local and remote databases pass or fail the filter expression. The two databases are then synchronized. After synchronization, synchronization program **100** uses the results of synchronizing the two databases to add, modify, or delete the records of the two databases. At this point, the user can, for example, select to have those records falling outside the filtering criteria to be deleted, not to be affected at all by the synchronization program, or be treated in other manner, as will be described below. In other embodiments, synchronization program **100** uses the filter expression to identify and mark the records of only one of the databases.

We will now describe in detail the structure of synchronization program **100** and the method it uses to synchronize the local and remote databases using filter expressions. FIG. **2** shows the relationship between the various modules of an embodiment of synchronization program **100**. Translation Engine **1** comprises Control Module **2** and Parameter Table Generator **3**. Control Module **2** is responsible for controlling the synchronizing process by instructing various modules to perform specific tasks on the records of the two databases being synchronized. (FIG. **4** shows the steps taken by this module.)

Parameter Table Generator **3** is responsible for creating a Parameter_Table **4** which is used by all other modules for synchronizing the databases. Generally, Parameter_Table **4** stores various information which may be used by the modules of the synchronization program. The information stored in Parameter_Table **4** includes user preferences, the names and locations of the databases, and the names and locations of various files stored on disk including the name and location of the history file from the previous synchronization. Parameter Table Generator **3** also provides the user with various graphical user interface windows for inputting filter expressions to be used during synchronization. Parameter Table Generator **3** converts such user input filter expressions or previously stored filter expressions into data structures which may then be used by various modules of synchronization program **100** during synchronization. The steps taken by Parameter Table Generator **3** in relation to filter expressions will be described in detail below.

Synchronizer **15** has the primary responsibility for carrying out the core synchronizing functions. It is a table-driven code which is capable of synchronizing various types of databases whose characteristics are provided in Parameter_Table **4**. Synchronizer **15** creates and uses workspace **16** (also shown in FIG. **3**), which is a temporary data array used during the synchronization process.

Synchronization program **100** has two translator modules **5** and **9** which are generally responsible for data communication between synchronization program **100** and databases **13** and **14**. Translator (L_translator) **5** is assigned to the local database (L_database) **13** and translator **9** (R_translator) to the remote database (R_database) **14**. Each of the database translators **5** and **9** comprises three modules: reader modules **6** and **10** (L_reader and R_reader) which load (or read) records from databases **13** and **14**; unloader modules **8** and **12** (L_unloader and R_unloader) which analyze and unload records from workspace **16** into databases **13** and **14**; and sanitizing modules **7** and **11** (L_sanitizer and R_sanitizer) which analyze the records of the opposing database when they are loaded into the workspace and modify them according to rules of data value of the modules's own database. Briefly stated, rules of data value are generally rules that define the permitted content of the fields of the records of a database. An example of such a rule would be that no more than 100 characters may be present in a field, or that content of a field designating a priority for a "to do" item should be limited to 1, 2, or 3. Sanitizing a record is to change the content of the fields of a record of one database to conform to the rules of data value of another database. Rules of data value and sanitization are described in detail in the following commonly owned U.S. patent applications, incorporated in their entirety by reference, "Synchronization of Recurring Records in Incompatible Databases", Ser. No. 08/752,490, filed on Nov. 13, 1996 (hereinafter, "application '490"); "Synchronization of Databases with Record Sanitizing and Intelligent Comparison," Ser. No. 08/749,926, filed Nov. 13, 1996 (hereinafter, "application '926"); "Synchronization of Databases with Date Range," Ser. No. 08/748,645, filed Nov. 13, 1996 (hereinafter, "application '645").

In the described embodiment, the modules of L_translator **5** are designed specifically for interacting with local database **13** and local application **17**. The design of the modules of L_translator **5** is specifically based on the record and field structures and the rules of data value imposed on them by the local application, the Application Program Interface (API) requirements and limitations of local application **17** and other characteristics of the local database and application. The same is true of the modules of R_translator **9**. These translators are typically not able to interact with other databases or Applications and are only aware of the characteristics of the database and application for which they are designed. Therefore, when the user chooses two applications for synchronization, Translation Engine **1** chooses the two translators which are able to interact with those applications. In an alternate embodiment, the translators can be designed as table-driven codes, where a general translator is able to interact with a variety of applications and databases based on supplied parameters.

Having described the structure of synchronization program **100** in reference to its various modules, we will now describe the operation of synchronization program **100**. During synchronizing the two database, Control Module **2** instructs the various modules in synchronization program **100** to perform specific tasks. We will describe the operation of synchronization program **100** by describing the steps taken by Control Module **2** (as set out in the pseudo code in FIG. **4**) and describing in detail the actions by the various modules as they are instructed by Control Module **2**.

Referring to FIG. **4**, in the first step of synchronizing the two databases, Control Module **2** instructs the Parameter

Table Generator **3** to create parameter table **4** (Step **100**). In this step, as part of creating parameter table **4**, Parameter Table Generator **3** obtains from the user a filter expression, if any, to be used during synchronization or alternatively accesses a previously stored filter expression, if any.

FIG. **5** is pseudo code for the steps taken by Parameter Table Generator **3** in relation to filter expressions. In step **150**, Parameter Table Generator **3** determines from the user the whether a filter should be used during the current synchronization. FIG. **6** shows a filter selection window **40** which the user uses to select whether to use a filter for the current synchronization. If the user selects to use a filter (step **151**), Parameter Table Generator **3** next determines the filter to be used (step **152**). This filter may be a previously stored filter expression (e.g. filter **42** in FIG. **6**) or a filter expression which the user inputs (e.g. by selecting "New" button **44** in FIG. **6**).

For a new filter, the user uses a filter name input window **50** to input the filter's name (step **153**), shown in FIG. **7**. The user then uses a filter criteria input window **60** (step **154**), shown in FIGS. **8** and **9**, to input the filter expression for the new filter. We will describe window **60** in further detail below. Parameter Table Generator **3** next stores the current date and time in the FILTER_CHANGED_TIMESTAMP parameter (step **155**). This parameter is used to determine whether a filter has changed since a previous synchronization. Parameter Table Generator **3** then assigns a unique filter identifier code to the filter expression, which is then used to identify the filter (step **156**).

If the user selects to use a previously stored filter expression, Parameter Table Generator **3** obtains the name of the filter from the user (step **157**) and then retrieves the filter expression and the unique filter identifier code of that filter (step **158**). If the user selects to edit the filter, Parameter Table Generator **3** displays the filter expression in window **60** and allows the user to edit the filter expression (step **159**). In this step, Parameter Table Generator **3** also stores the current date and time in the FILTER_CHANGED_TIMESTAMP parameter, as in step **155**.

In step **160**, Parameter Table Generator **3** sets the FILTER_ID parameter to the unique filter identifier of the filter that was selected. The modules of synchronization program **100** use FILTER_ID parameter to determine the correct filter to use. Parameter Table Generator **3** next sets USE_FILTER flag (step **161**). This flag indicates to various module of synchronization program **100** that a filter is to be used during synchronization and causes the appropriate modules to take the necessary steps, as will be described in further detail below.

During synchronization, the filter may be applied to the records of the database by either the translator for that database or by the database manager application of that database, if the database manager application is capable of applying a filter. In step **162**, Parameter Table Generator **3** parses the filter expression into a filter token array which the translators use when filtering records of the databases and history file. The filter token array and the manner in which it is used will be described in detail below. Parameter table generator **3** will next create Parameter_Table **4**, as described in detail in the '490, '926 and '645 applications.

Having described the steps taken by Parameter Table Generator **3** with respect to a filter to be applied during synchronization, we will now describe in detail graphical user interface (GUI) windows displayed for entering the filter expressions, the semantics of filter language used in the described embodiment, the manner in which inputted filter criteria are stored, and the method used by translators to determine whether a record passes the filter. However, it should be noted that other filtering languages and methods may also be used to filter records during synchronization.

Generally, synchronization program **100** uses two types of filters: static and dynamic filters. Static filters have a fixed and unchanging filter expression. For example, the filter "appointments in 1997" is a static filter. It will always the cover the appointments in 1997. Dynamic filters have a changing filter threshold value, which may depend on a changing parameter. For example, the filter "appointments from today until a year from today" is a dynamic filter because the threshold value of the filter changes as the value of "today" changes. (It should be noted that the above examples are also examples of date range filters. Date range filters filter records based on whether the dates of the records fall within a range of dates specified by the filter.)

In the case of dynamic filters, Parameter Table Generator **3** uses the dynamic filter to create two filter expressions to be used during synchronization. The first filter expression, which we will refer to as the current filter, is based on the value of the filter for the current synchronization. For example, in the case of date range filters based on the value of the current synchronization's date, the value of the current filter will be based on the current synchronization's date. (Date range filters and a method of synchronizing databases using them are described in detail in the '490, '926 and '645 applications.) Since a dynamic filter is a changing filter, records which were previously within the filter may not be within the current filter. However, those records and any changes in those records should be used during the current synchronization since the user likely treated those records as being validly within the filter and might have modified them. Therefore, Parameter Table Generator **3** creates a second filter expression, which we will refer to as the loading filter, which combines the value the current filter with the value of the filter as it was during a previous synchronization. For example, in the case of a dynamic date range, the loading filter would be a concatenation of the current filter and the filter based on the date of the previous synchronization. The manner in which these two filters are used will be described below.

In synchronization program **100**, a filter expression applied to both local and remote databases. However, the filter expression is typically inputted and stored based on the list of fields of one of the databases—in the described embodiments, the local database. A field map which maps the fields of the two databases onto one another is used by synchronization program **100** to apply a filter expression based on the field list of one of the databases to the fields in the records of the other database, as will be described in detail below.

The table in FIG. **10** shows the specification of the semantics of the filter language. We define a filter expression (or filter criteria) as a group of one or more filter conditions (or filter criterion). In the filter language described here, when a record is evaluated against a filter condition, the result may be either a boolean value (i.e. TRUE or FALSE) or a numeric value (e.g. 1, 2, 3). However, the final result of evaluating a record against the filter expression is a boolean value which indicates whether the record has passed or failed the filter.

A filter condition may be considered to be a sentence having three parts: <evaluated operand> <filter operator> <filter threshold operand>, where the evaluated operand is the operand to be evaluated against the filter, filter threshold

operand is the threshold value of the filter condition, and the operator is the filtering function to be performed between the evaluated operand and filter threshold operand. For example, the following is a filter condition: <date field> <is greater than or equal to> <TODAY>.

An operand may have one of two values: a value inputted by a user or a value taken from a record to be evaluated. To indicate that the value of an operand is to be taken from a field in a record, the name of that field is used as the operand. In the described filter language, a field name is enclosed with brackets—e.g. [Start Date]. During evaluation of a record against the filter, the value stored in that field of the evaluated record will be used as the operand.

In the case where the operand has a user inputted value, the operand contains that value—e.g. 'Smith', 456–7896, or 'TODAY'. In the described filter language, the user-inputted operands may have one of the following types of values, which is coextensive with the possible field values of the local and remote databases: DATE, TIME, TEXTSTRING, BOOLEAN or NUMBER.

We will now describe in detail an example of the range of values the various types of user-inputted operands in the described filter language may have. However, it should be noted that other embodiments may have other ranges and limitations. In the described embodiment, DATE operands are formatted as 'YYYYMMDD' (in some embodiments, single quotes must be included)—example '19980101' is New Years Day of 1998. DATE operands may also have the value TODAY, which indicates the date for today obtained from the operating system of the computer. TIME operands are formatted as 'HHMM' on a 24-hour clock basis (in some embodiments, single quotes must be included)—for example, '0600' represents 6:00 AM and '1300' represents 1:00 PM. TIME operands may also have the value NOW, which indicates the current time obtained from the operating system of the local computer.

TEXTSTRING operands can contain any text value (in some embodiments, single quotes must be included to indicate that the value is a text string)—examples are 'Puma Technology, Inc.', '15', and '#$%'. BOOLEAN operands must be of value TRUE or FALSE (no use of single quotes). NUMBER operands may be any integer or floating point number (in some embodiments, single quotes are not used for NUMBER operands).

In the described filter language, the available filter operators are organized into seven different operator sets. The first operator set (also referred to as "OP_SET_**1**") includes the following filter operators: EQ (equal), LE (less than or equal), GE (greater than or equal), NE (not equal), LT (less than) and GT (greater than). This set of operators may be used for all of the various types of operands, provided that both operands involved in the filter condition are of the same type.

The second operator set (also referred to as "OP_SET_**2**") is to be used only when evaluated operand is of the type DATE. OP_SET_**2** provides for using dynamic filters for operands of type DATE (e.g. dynamic date range filters). OP_SET_**2** is made up of all combinations of the filter operators in OP_SET_**1** and the variables TODAY− and TODAY+. An OP_SET_**2** filter operator is followed by a filter threshold operand whose value is a selected number of days. The variable TODAY+ in an OP_SET_**2** filter operator then indicates the number of days in the filter threshold operand is to be added to the date of the current synchronization prior using the OP_SET_**1** filter operator to evaluate filter. For example, consider the filter condition: <appointment date><LT TODAY+><3>. In this filter expression, 3 days are added to the date of the current synchronization to obtain the date of the third day after today and then the filter operator LT is used to determine whether the appointment date is less than the date of the third day after today.

The third operator set (also referred to as "OP_SET_**3**") is similar the second operator set and is used only when both operands are of the type TIME. OP_SET_**3** provides for using dynamic filters for operands of type TIME. OP_SET_**3** includes all combinations of filter operators in OP_SET_**1** with the variables NOW− and NOW+. The variable NOW represents the time of the current synchronization, typically obtained from the operating system. An OP_SET_**3** filter operator is followed by a filter threshold operand whose value is a selected number of seconds. The variables NOW+ and NOW− are used in the same manner as the variables TODAY+ and TODAY−.

The fourth operator set (also referred to as "OP_SET_**4**") may be used when both operands are of the type TEXTSTRING. OP_SET_**4** includes the following operators: STARTS_WITH, CONTAINS, DOES_NOT_CONTAIN, IS_EMPTY and IS_NOT_EMPTY.

The fifth operator set (also referred to as "OP_SET_**5**") may be used when both operands are of the type NUMBER. OP_SET **5** includes the following operators: + (addition), − (subtraction), * (multiplication), / (division) and % (modulus).

The sixth operator set (also referred to as "OP_SET_**6**") may be used only when both operands are of the type BOOLEAN. OP_SET_**6** includes only the operator IS.

The seventh operator set (termed OP_SET_**7**) may be used to combine two filter conditions. OP_AND_OR includes the relational operators AND and OR. Filter conditions may be combined using the seventh operator set to form filter expressions. In the described embodiment, the operand from the seventh operator set are used so as to achieve one of two results: a record must either meet all of the filter conditions in a filter expression or only one of the filter conditions. In other embodiments, more complex filter expressions may be permitted. In such embodiments, the order of evaluation may follow a predetermined order of evaluation (e.g. the order of evaluation in the 'C' programming language) which may in turn be modified by parentheses.

Following are several exemplary filter expressions based on the above described filter language:

[Full Name] CONTAINS 'Smith' AND [Private Flag] IS FALSE

([Priority]/2) GT 0

[Alarm Date] EQ '**19980101**' AND ([Regarding CONTAINS 'meeting' OR [Regarding] CONTAINS 'training') AND [Location] CONTAINS 'Boston'

Referring back to FIGS. **8** and **9**, the user uses the filter expression input window **60** to enter the filter expression to be used during synchronization. Filter expression input window **60** includes two so-called tabs (e.g. used in operating systems sold under the tradename Windows by Microsoft Corporation of Redmond, Wash.). Conditions tab **62** (shown in FIG. **8**) is used to input the filter conditions. Rules tab **64** (shown in FIG. **9**) is used to select how the filter conditions should be combined to create the filter expression.

Referring to FIG. **8**, conditions tab **62** includes a filter condition input region **66**. The user can select a field name from a list of field names of the database on which the filter

is based by using pull-down menu **66***a*. Alternatively, the user may type a valid name in field name region **66***b*. The user may also type in a valid filter threshold operand in threshold operand region **66***e*. In filter operator region **66***c*, the user may type a valid filter operator or select one from a filter operator pull-down menu (not shown; only button **66***d* for clicking on to pull-down the menu is shown). It should be noted that the list of available options in pull-down menu **66***d* is limited by the type of operand entered in field name region **66***b* or filter threshold operand region **66***e*.

The user may add a filter condition to the list displayed in filter conditions display region **66***g* or remove a filter condition from that list, by selecting the appropriate button in region **66***f*. The user may also update (i.e. change or edit) a filter condition, by selecting the appropriate "button" in region **66***f*.

Referring to FIG. **9**, in rules tab **64**, more particularly in region **64***a*, the user may select whether a record must either meet all of the filter conditions inputted by the user or only one of the filter conditions in order to pass the filter.

The filter expression input by the user in the filter expression input window **60** is then stored as a filter expression based on the above described filter language.

We will now describe the filtering methodology used in synchronization program **100** to evaluate a record against a filter. Briefly, the filtering methodology used in synchronization program **100** generally has two steps. The first step is parsing the stored filter expression and forming a filter token array which is designed to facilitate evaluating the records against the filter. In synchronization program **100**, Parameter Table Generator **3** performs this step (FIG. **5**, step **162**). The second step is evaluating each record of the database or the history file to be filtered against the filter expression to determine whether the record passes the filter. In synchronization program **100**, the translators performs this step in the case of the local and remote databases. Synchronizer **15** performs this step for the history file. We will now describe each of these steps in detail.

Parameter Table Generator **3** parses the stored filter expression and then forms the filter expression into a filter token array to be used during evaluation of the records against the filter expression. A token in the filter token array is a data structure which represents either an operand (i.e an evaluated operand or a filter threshold operand) or a filter operator. A token contains two pieces of information—the type of the token and the value of the token. The type of the token may be one of the following: TEXTSTRING, DATE, TIME, BOOLEAN, NUMBER, OP_SET_**1**, OP_SET_**2**, OP_SET_**3**, OP_SET_**4**, OP_SET_**5**, OP_**6** or OP_SET_**7**. The value of the token will be the actual content of the token and is stored as a string of characters which is obtained from the filter expression. For example, a token which represents a date filter threshold operand "Feb. 7, 1988" will have a DATE type and a "Feb. 7, 1988" value. A filter operand GE (greater than or equal to) will have a OP_SET_**1** type and a "GE" value. An evaluated operand having a field name "[contact address]" will have a TEXTSTRING type and a "[contact address]" value.

As Parameter Table Generator **3** parses the filter expression, it turns the operands and operators of the filter conditions into tokens. Each filter condition yields three tokens. Parameter Table Generator **3** stores the tokens in the filter token array in a specific order. In the case of each filter condition, the evaluated operand token is stored first, followed by the filter threshold operand token and then the filter operator token. In the case of filter operators from OP_SET_**7** (i.e., the filter operators AND and OR), the two filter conditions connected by the operator are stored first, followed by the filter operator. Ordering the tokens in this manner facilitates evaluating the records against the filter, as will be described below.

As an example, parameter generator **3** parses the following filter expression:

[Alarm Date] EQ '19980101' AND ([Regarding] CONTAINS 'meeting' OR [Regarding] CONTAINS 'training') AND [Location] CONTAINS 'Boston'

into the following token array:

value	type
AND	OP_SET_7
CONTAINS	OP_SET_4
'Boston'	TEXTSTRING
[Location]	TEXTSTRING
AND	OP_SET_7
OR	OP_SET_7
CONTAINS	OP_SET_4
'training'	TEXTSTRING
[Regarding]	TEXTSTRING
CONTAINS	OP_SET_4
'meeting'	TEXTSTRING
[Regarding]	TEXTSTRING
EQ	OP_SET_1
'19980101'	DATE
[Alarm Date]	DATE

As stated above, the second step in filtering the records of a database is evaluating each record of that database against the filter expression. The translator of the database to be filtered performs this step. Both of these modules use the same method of evaluation, which we will now describe.

To evaluate a record against a filter expression, a translator uses a stack, which we will refer to as the evaluation stack. To evaluate the record, starting with the last item in the filter token array, the translator proceeds through the filter token array and pushes copies of the operand tokens onto the evaluation stack. When an operand token is a field name, the translator instead of pushing the field name, pushes the data stored in that field of the record onto the stack. When the translator encounters an operator token in the token array, the translator pops the last two items in the evaluation stack and evaluates the two items based on the filter operator. The translator push the result of the evaluation, which may be either of the type BOOLEAN or NUMBER, onto the evaluation stack. After the translator evaluates the entire filter expression, a single boolean value is left in the stack which indicates whether the record has passed the filter.

We will now provide an example of evaluating a record against a filter. Prior to the evaluating step, Parameter Table Generator **3** would have parsed the following filter expression:

[Last Name] STARTS_WITH 'A' AND [City] EQ 'Boston' into the following token array:

value	type
AND	OP_SET_7
EQ	OP_SET_1
'Boston'	TEXTSTRING
[City]	TEXTSTRING
STARTS_WITH	OP_SET_4
'A'	TEXTSTRING
[Last Name]	TEXTSTRING

During filter evaluation, the first two tokens are pushed onto the evaluation stack:

token array	evaluation stack
AND	[Last Name]
EQ	'A'
'Boston'	
[City]	
STARTS_WITH	

Next, the operator STARTS_WITH is encountered. Therefore, two operands are popped from the evaluation stack and evaluated using the operator. The result or this evaluation (e.g. TRUE) is pushed back onto the evaluation stack:

token array	evaluation stack
AND	TRUE
EQ	
'Boston'	
[City]	

Two more operands from token array are pushed onto the evaluation stack:

token array	evaluation stack
AND	TRUE
EQ	[City]
	'Boston'

The operator EQ is next encountered. Therefore, two operands are popped from the evaluation stack and evaluated using the operator. The result of this evaluation (e.g. TRUE) is then pushed onto the evaluation stack.

token array	evaluation stack
AND	TRUE
	TRUE

Finally, the remaining operator AND is encountered in the filter token array. Two operands are popped from the evaluation stack and evaluated using the operator. The result of this evaluation (i.e. TRUE) is also pushed onto the evaluation stack. Because there are no more tokens in the filter token array, the remaining token on the evaluation stack is the final result of the filter evaluation. This final token indicates that the record being evaluated passes the filter.

As stated briefly above, synchronization program **100** applies a filter expression based the field list of the remote database to the local database. To do so, synchronization program **100** uses a field map to determine which field of the local database corresponds to a field of the remote database used in the filter expression. Field mapping is described in U.S. Pat. No. 5,392,390, incorporated by reference. Briefly, to synchronize records of two databases, it is essential to determine which field or fields of one database should be synchronized with which field or fields of the other database. To accomplish this, a field map is used which correlates the fields of the two databases to one another. It should be noted that not all fields of a database are mapped onto the other database and therefore such unmapped fields are not synchronized with the other database.

In the described embodiment, L_translator **5** uses a filter expression which is based on the field list of the remote database to filter the records of the local database. For every token in the filter token array that contains a field name, L_translator **5** uses the remote database to the local database field map to determine the corresponding field in the local database record being evaluated and pushes the content of that field onto the evaluation stack. It may be the case that the field in the filter expression is an unmapped field and therefore does not have counterpart in the local database record being evaluated. If that is the case, L_translator **5** marks the token and its corresponding operator and operand tokens with a SKIP_EVAL flag. During the evaluation phase if an operator token is marked with a SKIP_EVAL flag, L_translator **5** determines the result of that operation to be TRUE. (If the operation to be skipped is to return a NUMBER type value, L_translator **5** determines the result to be '0' or zero. L_translator **5** then marks the resulting token, and that token's corresponding operator token and other operand token, with a SKIP_EVAL flag.) L_translator **5** applies the filter to the record in this manner until a final result remains in the evaluation stack. In alternative embodiments, the filter expression may be based on the field list of the local database or the user may select the database on whose field list the filter expression is to be based.

As stated above, instead of the translator for the database to be filtered, the database manager application for that database may filter the records of that database. The translator for the database parses the filter expression into a set of instructions formatted for the Application Programmer Interface (API) of the database manager application. In that case, the database application manager transmits to synchronization program **100** only those records that pass the filter.

To determine whether the translator or the application will filter the records of a database, the modules of synchronization program **100** use two types of flags. First, as described above, Parameter Table Generator **3** sets a USE_FILTER flag if a filter is being used (FIG. **5**, step **161**). Second, each of translators **5** and **9** in turn set an appropriate flag, i.e. R_Application_Is_Filtering or L_Application_Is_Filtering, to indicate that the database manager application will apply the filter. If both the USE_FILTER and the R_Application_Is_Filtering (or L_Application_Is_Filtering) flags are set, the flags indicate the remote (or local) database manager application will apply the filter to its database.

Referring back to FIG. **4**, after Parameter Table Generator **3** creates the parameter table, Control Module **2** of the Translation Engine **1** instructs synchronizer **15** to initialize itself (step **101**). Synchronizer **15** in response creates the workspace data array **16**. Control Module **2** of the Translation Engine **1** then instructs synchronizer **15** to load history file **19** into workspace **16** (step **102**). History file **19** is a file that was saved at the end of last synchronization and contains records reflecting the records of the two databases at the end of the previous synchronization. Synchronizer **15** uses history file **19** during current synchronization to analyze the records of the local and remote database to determine the changes, additions, and deletions in each of two databases since the previous synchronization. Synchronizer **15**, as result of this analysis, then can determine what additions, deletions, or updates need be made to synchronize the records of the two databases.

In various situations, synchronizer **15** does not load history file **19**. For example, if no history file from a previous synchronization exists or if the user chooses to

synchronize not using the history file, synchronizer **15** will not load history file **19**. Additionally, the user may wish to use a filter expression that is different from the filter expression used in the previous synchronization (including using no filter expression during the current synchronization). In that case, if one of the database manager applications filtered its database, then synchronizer **15** does not load the stored history file. This is because when a database manager application filters the records of a database, the history file contains only those records which pass the previous filter and not necessarily the records necessary for performing a synchronization using the current filter. Obviously, in the case where a history file is not loaded, synchronizer **15** synchronizes the two databases without using a history file.

FIG. **11** is pseudocode for the steps taken by synchronizer **15** to load history file **19**. For each Record in history file **19** (step **200**), synchronizer **15** first loads the record (step **201**) and then writes the loaded record into workspace **16** (step **202**). Synchronizer **15** repeats these steps until all of the records of the history file are loaded into the workspace.

Referring back to FIG. **4**, after the history file is loaded into the workspace, Control Module **2** instructs R_translator **13** to load the remote database records (step **103**). FIG. **12** is pseudocode for the steps taken by R_translator **13** to load the remote database records. If the USE_FILTER flag is set but R_Application_Is_Filtering flag is not set (step **300**) then R_reader module **11** will apply the filter to the records of the remote database. For each record of the remote database (step **301**), R_reader module **11** of the R_translator first loads the record (step **302**). R_reader module **11** applies the filter expression identified by the parameter Filter_Id to the loaded record (step **303**). If the record passes the filter then R_reader module **11** marks the record as having passed the filter (step **304**). If the record does not pass the filter then R_reader module **11** marks the record as having failed the filter (step **305**). R_reader module **11** then sends the record to synchronizer **15** (step **306**) and synchronizer **15** writes the loaded record into the workspace (step **307**). Steps **302**–**307** are performed until all records of the remote database are loaded.

If the Use_Filter flag and R_Application_Is_Filtering flags are both set (step **309**), then the remote database application will filter the loaded records. R_reader module **11** converts the filter expression into the format required by remote database application's API and sends the converted filter expression to the remote database application (step **310**). R_reader module **11** then loads the filtered records (step **311**). If a record that was loaded passes the current filter expression, then R_reader module **11** marks that as having passed the filter; otherwise, R_reader module marks the record as having failed the filter (step **312**). Since only those records that have passed the filter are loaded, R_reader module **11** does not mark any of the loaded records as having failed the filter. R_reader module **11** then sends the loaded records to synchronizer **15** (step **313**) and synchronizer **15** writes the loaded records into the workspace (step **314**).

Following loading the remote database records, Control Module **2** instructs L_sanitizer module **8** of L_translator **5** to sanitize the remote database records in the workspace (step **104**).

Control Module **2** of the Translation Engine **1** then instructs the L_translator **5** to load the records from the local database (step **105**). L_translator **5** and synchronizer **15** load records of the local database in the same manner as described for R_translator **9** in reference to FIG. **12**, except for two differences. First, as described above, L_translator **5** filters the records of the local databases using the remote database to local database field map. It should be noted that the field maps are contained within the parameter table and the contents of the parameter table are transmitted to L_translator as read-only data. Second, as synchronizer **15** receives each local database record from the L_reader module **7** of the L_translator **5**, synchronizer **15** maps that record using the local database to remote database map before writing the record into the next available spot in the workspace. This is due to the fact that, in the described embodiment, records in the workspace are stored according to the remote database data structure.

Referring back to FIG. **4**, after all records are loaded into the workspace, Control Module **2** instructs synchronizer **15** to perform a Conflict Analysis and Resolution ("CAAR") procedure on the records in the workspace, which procedure is described in detail in the '490, '926 and '645 applications (step **107**). Briefly, referring to FIG. **13**, synchronizer **15** processes the records in the workspace, including comparing them to one another, in order to form them into groups of related records called corresponding item groups (CIGs). Synchronizer **15** forms the CIGs as it loads the records into the workspace and completes the process as the first step in CAAR (step **350**). Each CIG may include at most one record from each of the databases and the history file. Each record in a CIG may be a recurring or a nonrecurring records. Where a database does not support recurring records, in the case of a recurring event, a CIG would contain related nonrecurring records from that database which together represent that recurring event. Based on this group of nonrecurring records, synchronizer **15** may create a model recurring record for use during the synchronization and include that model recurring record in the CIG. Hereinafter, when referring to a "record" in a CIG, we also intend to refer to such a group of related nonrecurring records in the CIG.

For each CIG (step **351**), synchronizer **15** then compares the records in the CIG to one another, determines their differences, and decides what synchronization action should be taken (step **352**). In essence, synchronizer **15** determines which record in the CIG contains the most current data. Synchronizer **15** then determine what synchronization action should be taken to conform the other records in the CIG to the record with the most current data (i.e. how the other records in the CIG should be changed). Synchronization actions with respect to a record include updating, deleting, adding, or not modifying that record.

We will now provide some examples of the results obtained in the CAAR analysis. If after comparing the records in a CIG, synchronizer **15** determines that the record from the local database is unchanged and the one from remote database is changed, synchronizer **15** determines that the local database record should be changed to conform to the remote database record. Or, if both records are changed (an example of what we refer to as a "conflict" since there is no clear choice of synchronization action), synchronizer **15** may use a user-selected rule to decide what synchronization should be taken. The rule may require, for example, not modifying either of the records, changing the remote database record to conform to the local database record, or asking the user to resolve conflicts.

In the described embodiment, when a filter expression is used during the synchronization, synchronizer **15** alters the synchronization outcome in at least three cases. First, if the synchronization outcome is that a conflict exists (step **354**), synchronizer **15** determines whether one of the records fails the current filter. If one of the records in the CIG fails the filter, then synchronizer **15** marks all records in the CIG so

that they are not updated (step **355**). In other embodiments, synchronizer may use the user-selected rule for resolving conflicts to resolve the conflict. If one of the records in the CIG does not fail the filter, then synchronizer **15** uses the user-selected rule to resolve the conflict (step **356**).

Second, as stated above, synchronizer **15** changes the content of all records in a CIG to that of the record with the most up to date data. Therefore, if the record that contains the most up to date data fails the current filter, then the other records in the CIG when updated will also fail the filter. To resolve this issue, in the described embodiment, instead of updating the records in the CIG, all records in that CIG are marked as having failed filter (step **358**). Therefore, the records are not updated when unloaded to the databases at the end of synchronization, as will be described below.

Third, the filter expression may contain a filter condition based on an unmapped remote database field. As described above, when applying such a filter to the local database records, the local translator evaluates the filter condition containing the unmapped field as having a TRUE value. In that case, some filtered local database records may be marked as having passed the filter while their corresponding remote database records may be marked as having failed the filter, even though the mapped fields of both records may contain the same data. In the described embodiment, if a such a filter expression is used and one of the records in a CIG is marked as having failed the current filter, then synchronizer **15** marks all of the CIG records as failing the filter (step **359**).

As stated above, in some cases, one of the databases may support recurring records while the other database may not. Therefore, a recurring record is fanned into a number of non-recurring records before being unloaded to the database that does not support recurring records. In such a situation, during CAAR, synchronizer **15** examines each recurring record to determine whether there are some fanned nonrecurring records which pass the value of the dynamic filter expression during the previous synchronization but fail the current value of the filter. If so, then the dynamic filter has changed in such a way that part of the set of fanned records fall outside of the current filter. In the described embodiment, in such a situation, synchronizer **15** determines that the recurring record should be fanned again to generate new fanned nonrecurring records and previously fanned nonrecurring records should be deleted. To accomplish this, synchronizer **15** flags the recurring record and the appropriate translator fans the recurring record into the appropriate database and deletes the previous instances (step **360**).

When synchronizer **15** finishes performing CAAR on the records, synchronizer **15** would have determined what synchronization action should be taken with respect all records to be synchronized. The records may then be unloaded into their respective databases. The translators will perform the specific synchronization actions to be taken with respect to the records of the databases. However, prior to doing so, the user is asked to confirm proceeding with unloading (FIG. **4**, steps **108**–**109**). Up to this point, neither the databases nor the history file have been modified. The user may obtain through the Translation Engine's User Interface various information regarding what synchronization actions will be taken upon unloading.

If the user chooses to proceed with synchronization and to unload, the records are then unloaded. The unloader modules **6**,**10** of the translators **5**,**9** perform the unloading for the databases. During unloading, translators may use the filter expression to limit the data that is unloaded to the databases. For example, the translators may unload only those records which fall within the filter expression and delete any record which falls outside of the filter expression. During unloading, synchronizer **15** also creates the history file and unloads the records into the history file. We will now describe the unloading of the records into the databases and the history file in detail.

Control Module **2** of Translation Engine **1** first instructs R_translator **9** to unload remote database records from workspace into the remote database (FIG. **4**, step **110**). FIG. **14** is pseudocode for the steps taken by R_translator **9** to unload the records. For each remote database record in the workspace (step **400**), R_translator **9** first determines whether the Use_Filter is set and the filter is a static filter (step **401**). If that is the case, R_translator **9** determines whether the record passes or fails the filter. (It should be noted that, as described above, some records are marked as failing the filter during CAAR. In that case, during unloading, those records are considered to fail the filter.)

If the record fails the filter, R_translator **9** deletes the record from the remote database. If the record is passes the filter, then R_translator **9** adds, deletes, or modifies the record according to results of synchronization obtained during CAAR analysis (step **404**). If the remote database does not support recurring records or in other circumstances described in detail in the '490, '926 and '645 applications, R_translator **9** in step **404** may fan a recurring record by creating an appropriate number of nonrecurring records corresponding to the recurring record. If, as described above, synchronizer **15** during CAAR marks a recurring record for re-fanning (FIG. **13**, step **360**), R_translator **9** in this step will re-fan the record. When fanning, the number of nonrecurring records would be limited by any date based filters (i.e any date range) or other filters, so that nonrecurring records falling outside the filter are not created. Additionally, if a user has selected to limit the number of fanned nonrecurring records for each recurring record, R_translator would create only a limited number of instances, as described in more detail in the '490, '926 and '645 applications.

If the Use_Filter is not set or the filter is not a static filter (step **401**), R_translator **9** determines whether the Use-Filter flag is set and the filter is a dynamic filter (step **405**). If the Use-Filter flag is set and the filter is a dynamic filter, then R_translator **9** determines whether the record to be unloaded passes or fails the current filter. If the record does not pass the current filter (step **406**) then the record is deleted on the remote database (step **407**). If the record passes the current filter, then R_translator **9**, in the same manner as step **404**, adds, deletes, or modifies the record according to results of synchronization obtained during CAAR analysis (step **408**).

By deleting records which fail the filter expression in steps **403** and **407**, R_translator **9** uses the filter to limit the size of the remote database. If the remote database is located on a handheld computer, R_translator manages the memory of the handheld device by limiting the size of the database stored on the handheld computer.

Following unloading of the remote database records, Control Module **2** instructs the L_translator to unload the local database records from the workspace (FIG. **4**, step **111**). FIG. **15** is pseudocode for the steps taken by L_translator **5** to unload the local database records in the workspace. The steps **450**–**452**, **454**–**455**, **458**–**460** are respectively the same as steps **400**–**402**, **404**–**405**, and **408**–**410** in FIG. **14**, described in detail above. Unlike R_translator **9**, L_translator **5** does not delete records falling outside of the filter. Therefore, in step **453**, if the filter

is a static filter and the record does not fit the filter, then L_translator **5** modifies (i.e updates) the record if that is the synchronization result obtained during the CAAR analysis. However, synchronizer **15** does not add or delete a record, if that is synchronization result obtained during the CAAR analysis. In step **456**, similarly, if the record is within the loading filter, then L_translator **5** modifies the record if that is the synchronization result obtained during the CAAR analysis (step **457**). In this manner, L_translator **5** propagates to the records of the local database changes to those remote database records which do not fit the current filter expression. It should be noted that such remote database records are deleted from the remote database (step **407**, FIG. **14**) since those remote database records fail the current filter. Also, during unloading, the unloader module of the L_translator uses the remote database to local database map to map the records in the workspace back into the format of local database records.

Additionally, it should be noted that where the local database manager application filters the local database, the local database manager application only provides those records which pass the loading filter. Therefore, the effect of step **457** is to update those records which passed the previous value of the filter expression but fail the current value of the filter expression.

It should be noted that in other embodiments, translators for the local and remote databases may use the filter expression during unloading in different manner than in the above described embodiments. For example, the remote translator may be configured in a similar manner as the local translator described above. Or, either the remote or local translator may only update records within the filter and leave completely unaffected records outside the filter. A translator may also add new records from one database to another, if those records fall outside of the current filter but are within the loading filter.

Control Module **2** next instructs synchronizer **15** to create a new history file (step **112**). The process of creating a history file is described in detail in the '490, '926 and '645 applications. Briefly, for each CIG, synchronizer **15** during the CAAR process determines which one of the records in the CIG should be saved as the history file record. Based on these results, synchronizer **15** creates a history file. Synchronizer **15** also stores with each history file record the PASSED_FILTER/FAILED_FILTER flag based on whether the record passes or fails the current filter. Synchronizer also stores the value which determined the value of the dynamic filter for the current synchronization (e.g. the date of the current synchronization in the case of a dynamic date range filter).

At this point Synchronization is complete.

Other embodiments are within the following claims.

For example, if one of the databases has the capability to provide database generated information or data which can be used to determine, for example, whether a record has been changed, added, or deleted since a previous synchronization, the synchronization program uses that information to determine whether a record has been changed, added, or deleted. Of course, that database generated information is less than the whole record of the database. For example, that information may be a date and time stamp, or a flag, set when the record was last modified or when the record was added, whichever is later. We will briefly describe an embodiment of such a synchronization program, which is described in detail in the commonly assigned copending U.S. patent application, incorporated by reference in its entirety, entitled "Synchronization of Databases," filed on Nov. 5, 1997, Ser. No. 08/964,751 (hereinafter, the "'751 application").

There are generally two types of such databases: "medium synchronization" and "fast synchronization" databases. A "fast synchronization" database is a database which provides information regarding changes, deletions, and additions to its records from one synchronization to the next. A fast synchronization database also assigns to each record of the database a unique identification code (i.e. a unique ID) which uniquely identifies that record. Unique IDs are required to accurately identify records over a period of time. A fast synchronization database also provides a mechanism for keeping track of which records are added, changed, or deleted from synchronization to synchronization, including a list of deleted records.

A "medium synchronization" database typically has more limited capabilities than a fast synchronization database for keeping track of addition, deletions, or changes. In short, a medium synchronization database does not keep track of deletions. Such a database however still has the capability to provide information regarding what records were added or modified since a previous synchronization. A medium synchronization database also provides unique IDs.

If the information provided by a database indicates that a record has not been changed or added since a previous synchronization, the synchronization program need not load that record and can use the history file to reconstruct the relevant contents of that record for synchronizing the two databases. The history file contains a copy of the relevant content of that record as the record was at the time of (e.g. at the end of) the previous synchronization. Using the history file to reconstruct the record instead of loading the record can result in significant saving of time—where for example the data transfer link between the two computers is slow—since typically a majority of records in databases are unchanged records. The synchronization program thereby increases the efficiency of performing synchronization between two databases.

The synchronization program does not synchronize a record of the fast or medium synchronization database that fails the filter expression with the records of the other database. Therefore, the synchronization program does not reconstruct those unchanged records which fail the filter expression. However, the synchronization reconstructs unchanged records of the fast or medium synchronization database which failed the filter during the last synchronization but pass the current filter. In that case, since the records previously failed the filter, the records would not be in the other database. After reconstructing these unchanged records, the synchronization program treats these records as if these records were newly added to the fast or medium synchronization database and therefore adds these record to the other database.

As is apparent, when synchronizing a fast or medium synchronization database, the synchronization program may use the history file to reconstruct unchanged records, whether those unchanged records fail or pass the current filter. Therefore, the synchronization program at the end of each synchronization ensures that the records of the history file are synchronized with the records of the fast or medium synchronization database. In essence, the synchronization program ensures that the history file contains records which represent all of the records of the fast or medium synchronization at the end of the current synchronization, whether the records pass or fail the current filter. Since, as described above, some records of the fast or medium synchronization database are not present in the other database, the history file contains some records that are present only in the fast or medium synchronization database but not in the other database.

Where the database manager application of a fast or medium synchronization database filters the records of the database, the synchronization program does not receive those records of the database which fail the filter. Therefore, if the filter changes such that some of the unchanged records which were previously outside of the filter are now within the filter, the synchronization program can not rely on the history file. In that case, the synchronization program loads all records of the database and proceeds to synchronize without using the history file.

In some embodiments, both computers on which the two databases run are capable of running programs other than a database, as in the case of, for example, general purpose computers such as desktop and notebook computers, or handheld computers having sufficient memory and processing power. In such a case, the synchronization program may be distributed between the two computers so as to, for example, increase the efficiency of using of a slow data transfer link between the two machines. We will briefly describe an embodiment of such a distributed synchronization program, which is described in detail in the commonly assigned copending U.S. patent application, incorporated herein in its entirety by reference, entitled "Distributed Synchronization of Databases", filed on Sep. 11, 1997, Ser. No. 08/927,922 (hereinafter, the "'922 application").

Briefly, referring to FIG. **16**, at remote computer **22**, remote segment **26** of the synchronization program loads records of remote database **13**. Remote segment **26** then determines which records of the remote database have been changed/added, deleted or left unchanged since a previous synchronization. If the remote database assigns unique identification codes (i.e. unique ID) to its records, remote segment **26** can further differentiate between records than have been added and those than have been changed since the previous synchronization. Remote segment **26** uses a remote history file **30** which stores data representing or reflecting the records of the database at the completion of the previous synchronization. This data may be a copy of remote database **13**. It may also be hash numbers for each of the records of the remote database. If the remote database assigns unique IDs, the remote history file may contain those unique IDs together with the hash numbers of the records corresponding to the stored unique IDs.

Remote segment **26** sends those records of the remote database that have been changed or added to the host segment or the host computer. However, the remote segment does not send the unchanged or deleted records to the host computer. Instead, the remote segment sends a flag indicating the status of the record (e.g. unchanged or changed) and some data or information that uniquely identifies the record to the host segment. This data or information may be a hash number of all or selected fields in the record at the completion of the last synchronization. It may also be the unique ID assigned to the record by the remote database, if the database assigns one to its records.

Host segment **28** uses the received information or data that uniquely identifies the unchanged record to access a record in host history file **19** that corresponds to the received information or data. This record contains a copy of the data of the remote database record that the remote segment found to have been unchanged. Host segment **19** then uses this record to synchronize the databases in the same manner as described above. After synchronization, the remote and host history files are updated. Also, the records are unloaded to the remote and local database based on the filter expression, in the same manner as described above. Since the unchanged records which typically constitute most of the records of a database are not transferred to the host computer, a data transfer link, specially a slow data transfer link, is used with increased efficiency.

In such a distributed synchronization program, the remote and host segments would apply the filter expression to the records of the databases. In the case of the host segment, the process would be similar to that for the above described embodiments. In the case of the remote database, along with information identifying the record or the record's field values, remote segment **26** sends the host segment information indicating that the record passed or failed the filter. The synchronization process then proceeds as for the previously described embodiments. During unloading, the host segment sends the remote segment information with respect whether the records passed or failed the filter expression, along with the result of CAAR. Remote segment **26** then uses this information and the filter expressions, in the same manner as the above described translators when unloading records to the remote database.

In some embodiments, two or more databases on one computer may be synchronized with one database on the other computer. For example, it may be that the remote database application supports only one name and address database while the local database application supports two name and address databases. To synchronize the multiple local databases with the single remote database, one of the local databases is designated as the main local database for synchronization with the remote database. During synchronization, as synchronization program **100** adds records from the local databases to the remote database, synchronization program tags the added records with codes (i.e. origin tags) identifying the source of the database record, e.g. the first local database, the second local database, etc. Synchronization program **100** uses these tags during future synchronizations to ensure that the tagged records are synchronized with the correct database, i.e. the database from which they originated. Additionally, during synchronization, synchronization adds new records from the remote database only to the local database which was designated as the main database. This method of synchronization is described in detail the '490, '926 and '645 applications.

To enable filtering the records during synchronization, synchronization program **100** uses the origin tags of the records to ensure that the correct filter is applied to the correct records. For example, consider the case where two local databases are synchronized with a single remote database. Each of the local databases may have a unique filter. Or one database may have a filter and the other may not. When synchronization program **100** is synchronizing the local databases with the remote database, synchronization program **100** uses the origin tags of the remote database records to determine which filter should be applied to each record. If the origin tag of a remote database record indicates that it originated from the first database, then the filter expression for that database, if any, is used. Similarly, if the origin tag of a remote database record indicates that it originated from the second database, then the filter expression for that database, if any, is used. Additionally, if a remote database record is new (i.e. newly added since the previous synchronization), the filter expression for the local database that was designated as the main local database is used. It should be noted that while this method of synchronization was described for synchronizing a single remote database with multiple local database, same method may be used for synchronizing multiple remote databases with a single local database, or multiple remote databases with multiple local databases.

It should be noted that the synchronization process in the above embodiments was described primarily in reference to using a filter during synchronization. If a user chooses not to use a filter, the synchronization proceeds generally in the manner described the '751, '922, '490, '926 and '645 applications.

What is claimed is:

1. A computer implemented method of synchronizing at least a first and a second personal information management database of the type having a plurality of records, wherein the records of the first and second personal information management databases include fields, the method comprising:

using a filter to select a plurality of records of the first database, the filter comprising one or more user definable conditions or criteria that fields of records of the first database must match or fit to be selected, and

selecting the plurality of records of the first database includes evaluating fields of the first database with the user-definable conditions or criteria, and

synchronizing the selected records of the first database with records of the second database, the synchronizing comprising adding, modifying, or deleting records,

whereby synchronization is performed for a subset of the records of the databases.

2. A computer implemented method of synchronizing at least a first and a second personal information management database of the type having a plurality of records, wherein the records of the first and second personal information management databases include fields, the method comprising:

using a filter to select a plurality of records of the first database, the filter comprising one or more user selectable conditions or criteria that fields of records of the first database must match or fit to be selected, and

selecting the plurality of records of the first database includes evaluating fields of the first database with the user selectable conditions or criteria, and

synchronizing the selected records of the first database with records of the second database, the synchronizing comprising adding, modifying, or deleting records,

whereby synchronization is performed for a subset of the records of the databases.

3. The method of claim **1** or **2** further comprising:

deleting a second plurality of the records of the first database failing to fit the conditions or criteria.

4. The method of claim **3** further comprising:

updating a plurality of records of the second database failing to fit the current value of the conditions or criteria.

5. A computer program, resident on a computer readable medium, for synchronizing at least a first and a second personal information management database of the type having a plurality of records, wherein the records of the first and second personal information management databases include fields, comprising instructions for:

using a filter to select a plurality of records of the first database, the filter comprising one or more user definable conditions or criteria that fields of records of the first database must match or fit to be selected, and

selecting the plurality of records of the first database includes evaluating fields of the first database with the user-definable conditions or criteria, and

synchronizing the selected records of the first database with records of the second database, the synchronizing comprising adding, modifying, or deleting records,

whereby synchronization is performed for a subset of the records of the databases.

6. A computer program, resident on a computer readable medium, for synchronizing at least a first and a second personal information management database of the type having a plurality of records, wherein the records of the first and second personal information management databases include fields, comprising instructions for:

using a filter to select a plurality of records of the first database, the filter comprising one or more user selectable conditions or criteria that fields of records of the first database must match or fit to be selected, and

selecting the plurality of records of the first database includes evaluating fields of the first database with the user selectable conditions or criteria, and

synchronizing the selected records of the first database with records of the second database, the synchronizing comprising adding, modifying, or deleting records,

whereby synchronization is performed for a subset of the records of the databases.

7. The computer program of claim **5** or **6** further comprising instructions for:

deleting a second plurality of the records of the first database failing to fit the conditions or criteria.

8. The computer program of claim **7** further comprising instructions for:

updating a plurality of records of the second database failing to fit the current value of the conditions or criteria.

9. The method of claim **1** or **2** further comprising

displaying a record selection criteria input region on a computer display for a user to input a record selection criteria that specifies the conditions or criteria.

10. The computer program of claim **5** or **6** further comprising instructions for displaying a record selection criteria input region on a computer display for a user to input a record selection criteria that specifies the conditions or criteria.

11. The method of claim **1** or **2**, wherein the first and second databases are located on different computers and at least one of the computers is a handheld computer.

12. The computer program of claim **5** or **6** wherein the first and second databases are located on different computers and at least one of the computers is a handheld computer.

13. The method of claim **11**, wherein the handheld computer has less storage capacity than the other computer, wherein fewer records are stored in the database on the handheld computer than in the database on the other computer, and wherein the filter is used to limit the number of records added to the database on the handheld computer during a synchronization.

14. The computer program of claim **12**, wherein the handheld computer has less storage capacity than the other computer, wherein fewer records are stored in the database on the handheld computer than in the database on the other computer, and wherein the filter is used to limit the number of records added to the database on the handheld computer during a synchronization.

* * * * *

EXHIBIT B

US006212529B1

(12) **United States Patent**
Boothby et al.

(10) **Patent No.: US 6,212,529 B1**
(45) **Date of Patent: *Apr. 3, 2001**

(54) **SYNCHRONIZATION OF DATABASES USING FILTERS**

(75) Inventors: **David J. Boothby**, Nashua; **David W. Morgan**, Derry; **John R. Marien**, Nashua, all of NH (US)

(73) Assignee: **Puma Technology, Inc.**, San Jose, CA (US)

(*) Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/036,400**

(22) Filed: **Mar. 5, 1998**

Related U.S. Application Data

(63) Continuation-in-part of application No. 08/748,645, filed on Nov. 13, 1996.

(51) **Int. Cl.**[7] **G06F 17/30**; G06F 12/00

(52) **U.S. Cl.** **707/201**; 707/10

(58) **Field of Search** 707/10, 201, 202, 707/204, 203

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,432,057	2/1984	Daniell et al.	395/608
4,807,182	2/1989	Queen	395/144
4,819,156	4/1989	DeLorme et al.	364/200
4,827,423	5/1989	Beasley et al.	364/468
4,866,611	9/1989	Cree et al.	395/600
4,875,159	10/1989	Cary et al.	395/619
4,956,809	9/1990	George et al.	395/601
4,980,844	12/1990	Demjanenko et al.	364/550
5,065,360	11/1991	Kelly	395/800
5,136,707	8/1992	Block et al.	395/600
5,142,619	8/1992	Webster, III	395/161
5,155,850 *	10/1992	Janis et al.	707/202
5,170,480	12/1992	Mohan et al.	395/600
5,187,787	2/1993	Skeen et al.	395/600
5,210,868	5/1993	Shimada et al.	395/615
5,228,116	7/1993	Harris et al.	395/54
5,237,678	8/1993	Kuechler et al.	395/600
5,251,151	10/1993	Demjanenko et al.	364/550
5,251,291	10/1993	Malcolm	395/161
5,261,045	11/1993	Scully et al.	395/161
5,261,094	11/1993	Everson et al.	395/617
5,272,628	12/1993	Koss	364/419.19
5,278,978	1/1994	Demers et al.	395/600

(List continued on next page.)

OTHER PUBLICATIONS

U.S. application No. 08/927,922, filed Sep. 11, 1997.

U.S. application No. 08/964,51, filed Nov. 5, 1997.

Alfieri, "The Best of WordPerfect Version 5.0," Hayden Books, pp. 153–165, 429–435 (1988).

"Automatically Synchronized Objects," Research Disclosure #29261, p. 614 (Aug. 1988).

(List continued on next page.)

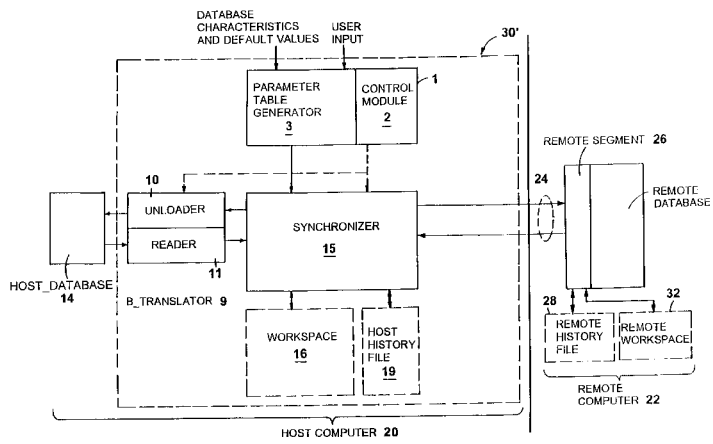
Primary Examiner—Thomas G. Black
Assistant Examiner—Frantz Coby
(74) *Attorney, Agent, or Firm*—Fish & Richardson P.C.

(57) **ABSTRACT**

A computer program is provided for synchronizing at least a first and a second database. A plurality of records of the first database fitting a selected criterion are identified. At least one of the identified records of the first database is then synchronized with a record of the second database. On a computer display, a record selection criteria displayed for a user to input the selected criterion.

52 Claims, 16 Drawing Sheets

Microfiche Appendix Included
(11 Microfiche, 1024 Pages)



U.S. PATENT DOCUMENTS

5,278,982	*	1/1994	Daniels et al.	707/202
5,283,887		2/1994	Zachery	359/500
5,293,627		3/1994	Kato et al.	395/550
5,301,313		4/1994	Terada et al.	395/600
5,315,709		5/1994	Alston, Jr. et al.	395/606
5,327,555		7/1994	Anderson	395/617
5,333,252		7/1994	Brewer, III et al.	395/767
5,333,265		7/1994	Orimo et al.	395/200
5,333,316		7/1994	Champagne et al.	395/600
5,339,392		8/1994	Risberg et al.	395/161
5,339,434		8/1994	Rusis	395/700
5,355,476		10/1994	Fukumura	395/600
5,375,234		12/1994	Davidson et al.	395/600
5,392,390		2/1995	Crozier	395/335
5,396,612		3/1995	Huh et al.	395/575
5,434,994		7/1995	Shaheen et al.	395/617
5,444,851		8/1995	Woest	395/200.1
5,463,735		10/1995	Pascucci et al.	395/200.1
5,475,833		12/1995	Dauerer et al.	395/617
5,511,188		4/1996	Pascucci et al.	395/600
5,519,606		5/1996	Frid-Nielsen et al.	395/228
5,560,005		9/1996	Hoover et al.	395/600
5,568,402		10/1996	Gray et al.	364/514 C
5,583,793		12/1996	Gray et al.	364/514 C
5,600,834		2/1997	Howard	395/617
5,613,113	*	3/1997	Goldring	707/202
5,615,364		3/1997	Marks	395/618
5,619,689		4/1997	Kelly	395/617
5,630,081		5/1997	Rybicki et al.	395/948
5,666,530		9/1997	Clark et al.	395/617
5,666,553		9/1997	Crozier	395/803
5,682,524		10/1997	Freund et al.	395/605
5,684,984		11/1997	Jones et al.	395/610
5,684,990		11/1997	Boothby	395/619
5,701,423		12/1997	Crozier	395/335
5,708,812		1/1998	Van Dyke et al.	395/712
5,708,840		1/1998	Kikinis et al.	395/800
5,710,922		1/1998	Alley et al.	395/617
5,727,202		3/1998	Kucala	395/610
5,729,735		3/1998	Meyering	395/610
5,745,712		4/1998	Turpin et al.	395/333
5,758,083	*	5/1998	Singh et al.	707/10
5,758,150		5/1998	Bell et al.	395/610
5,758,355		5/1998	Buchanan	707/201
5,778,388		7/1998	Kawamura et al.	707/203
5,790,789		8/1998	Suarez	395/200.32
5,845,293		12/1998	Veghte et al.	707/202
5,870,759	*	2/1999	Bauer et al.	707/201
5,870,765	*	2/1999	Bauer et al.	707/203
5,878,415	*	3/1999	Olds	707/9
5,884,323		3/1999	Hawkins et al.	707/201
5,884,324		3/1999	Cheng et al.	707/201
5,884,325	*	3/1999	Bauer et al.	707/201
5,897,640		4/1999	Veghte et al.	707/202
5,926,824		7/1999	Hashimoto et al.	707/520
5,978,813		11/1999	Foltz et al.	707/201

OTHER PUBLICATIONS

Cobb et al., "Paradox 3.5 Handbook 3rd Edition," Bantam, pp. 803–816 (1991).

"FRx Extends Reporting Power of Platinum Series: (IBM Desktop Software's Line of Accounting Software)," Doug Dayton, PC Week, v. 8, n. 5, p. 29(2) (Feb. 4, 1991).

IntelliLink Brochure (1990).

"Logical Connectivity: Applications, Requirements, Architecture, and Research Agenda," Stuart Madnick & Y. Richard Wang, MIT, Systems Sciences, 1991 Hawaii Int'l, vol. 1, IEEE (Jun. 1991).

"Open Network Computing—Technical Overview," Sun Technical Report, Microsystems, Inc., pp. 1–32 (1987).

Organizer Link II Operation Manual, Sharp Electronics Corporation.

Bishop et al., "The Big Picture (Accessing Information on Remote Data Management System," UNIX Review, v. 7, n. 8, p. 38(7) (Aug. 1989).

User Manual for Connectivity Pack for the HP 95LX, Hewlett Packard Company (1991).

User Manual for PC–Link for the B.O.S.S. and the PC–Link for the B.O.S.S., Traveling Software, Inc. (1989).

Zahn et al., *Network Computing Architecture*, pp. 1–11; 19–31; 87=115; 117–133; 187–199;201–209 (1990).

U.S. application No. 08/749926, filed Nov. 13, 1996.

U.S. application No. 08/752,490, filed Nov. 13, 1996.

U.S. application No. 08/748,645, filed Nov. 13, 1996.

Chapura, Inc., *3 Compare*, http://www.chapura.com/3compare.html (1997).

Chapura, Inc., *PilotMirror Features Page*, http://www.chapura.com/features.html (1997).

Wiederhold et al., Consistency Control of Replicated Data in Federated Databases, IEEE, pp. 130–132, Nov. 1990.*

Bowen et al., Achieving Throughput and Functionality in a Common Architecture: The DataCycle Experiment, IEEE, p. 178, Dec. 1991.*

* cited by examiner

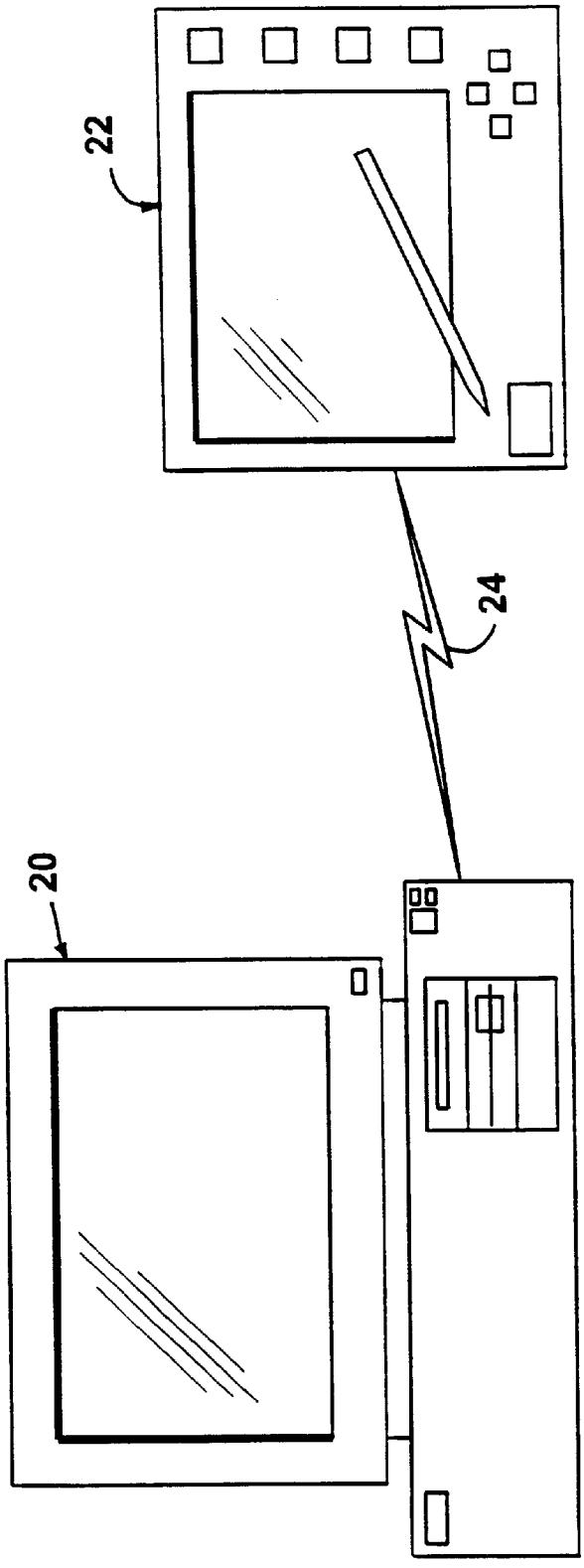


FIG. 1

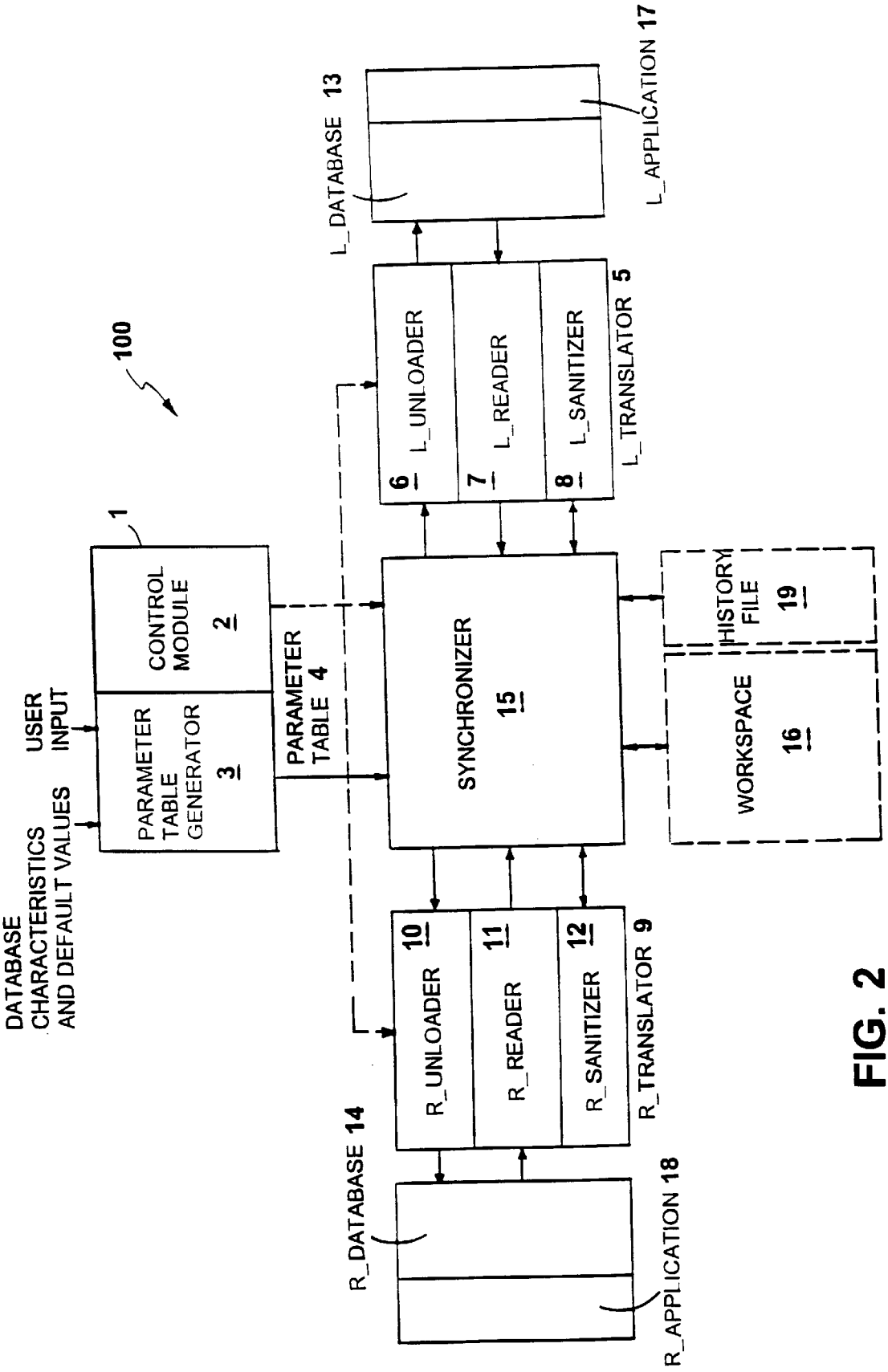

100
1
CONTROL MODULE 2
PARAMETER TABLE GENERATOR 3
USER INPUT
DATABASE CHARACTERISTICS AND DEFAULT VALUES
PARAMETER TABLE 4
L_TRANSLATOR 5
6 L_UNLOADER
7 L_READER
8 L_SANITIZER
L_DATABASE 13
L_APPLICATION 17
SYNCHRONIZER 15
HISTORY FILE 19
WORKSPACE 16
R_TRANSLATOR 9
10 R_UNLOADER
11 R_READER
12 R_SANITIZER
R_DATABASE 14
R_APPLICATION 18

FIG. 2

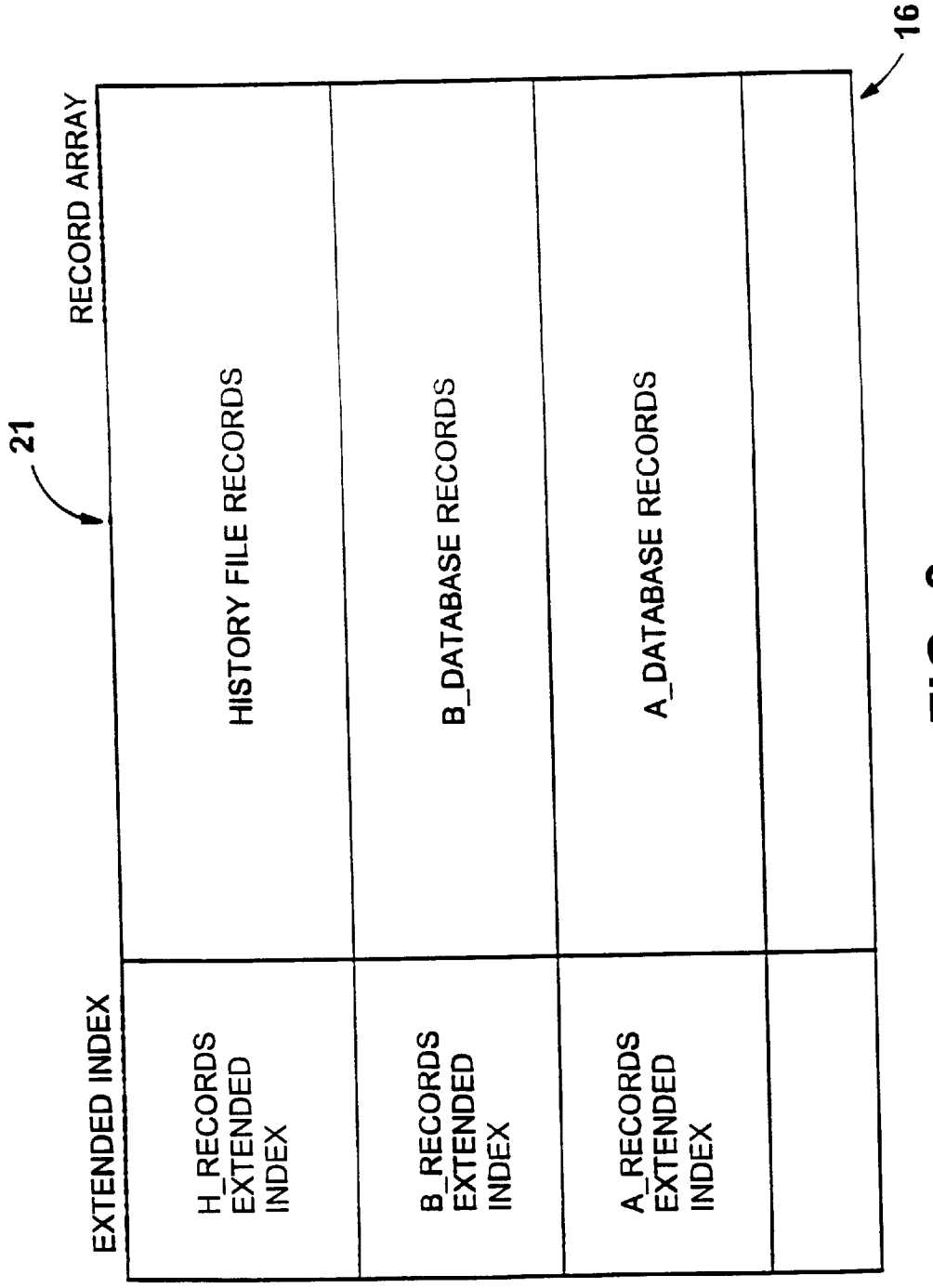

16
21
RECORD ARRAY
EXTENDED INDEX
HISTORY FILE RECORDS
B_DATABASE RECORDS
A_DATABASE RECORDS
H_RECORDS EXTENDED INDEX
B_RECORDS EXTENDED INDEX
A_RECORDS EXTENDED INDEX

FIG. 3

Pseudo Code for Translation Engine Control Module

100. INSTRUCT parameter table generator to create parameter table and initialize filter
101. INSTRUCT Synchronizer to initialize itself
102. INSTRUCT Synchronizer to LOAD the History_File into its WORKSPACE
103. INSTRUCT R_Translator to LOAD R_records from R_Database
104. INSTRUCT L_Translator to SANITIZE R_records that were just LOADED
105. INSTRUCT L_Translator to LOAD L_Records from L_Database and SEND to Synchronizer
106. INSTRUCT R_Translator to SANITIZE L_Records that were just LOADED.
107. INSTRUCT Synchronizer to do CAAR (Conflict Analysis And Resolution) on all the records in WORKSPACE.
108. INFORM user exactly what steps Synchronizer proposes to take (i.e. Adding, Changing, and Deleting records). WAIT for User.
109. IF User inputs NO, then ABORT.
110. INSTRUCT R_Translator to UNLOAD all applicable records to R_Database.
111. INSTRUCT L_Translator to UNLOAD all applicable records to L_Database.
112. INSTRUCT Synchronizer to CREATE a new History File.

FIG. 4

Pseudocode for Generating Parameter Table

```
{Get Input from the user}
150.    ASK user to select whether to use a filter expression
151.    IF the user selected to use a filter THEN
152.            IF a new filter to be used THEN
153.                    Obtain from the user filter name
154.                    Obtain filter expression
155.                    STORE the current date and time in the FILTER_CHANGED_TIMESTAMP
                        parameter
156.                    Assign a unique filter ID to the filter
157.            ELSE Obtain from the user filter name
158.                    retrieve the filter expression and unique filter ID
159.                    IF user selects to edit the filter THEN display the filter and obtain edits
160.            SET FILTER_ID parameter to unique filter ID code of the selected filter
161.            SET USE_FILTER flag
162.            PARSE the filter expression into a filter token array
163.    END IF
164.    CREATE parameter table
```

FIG. 5

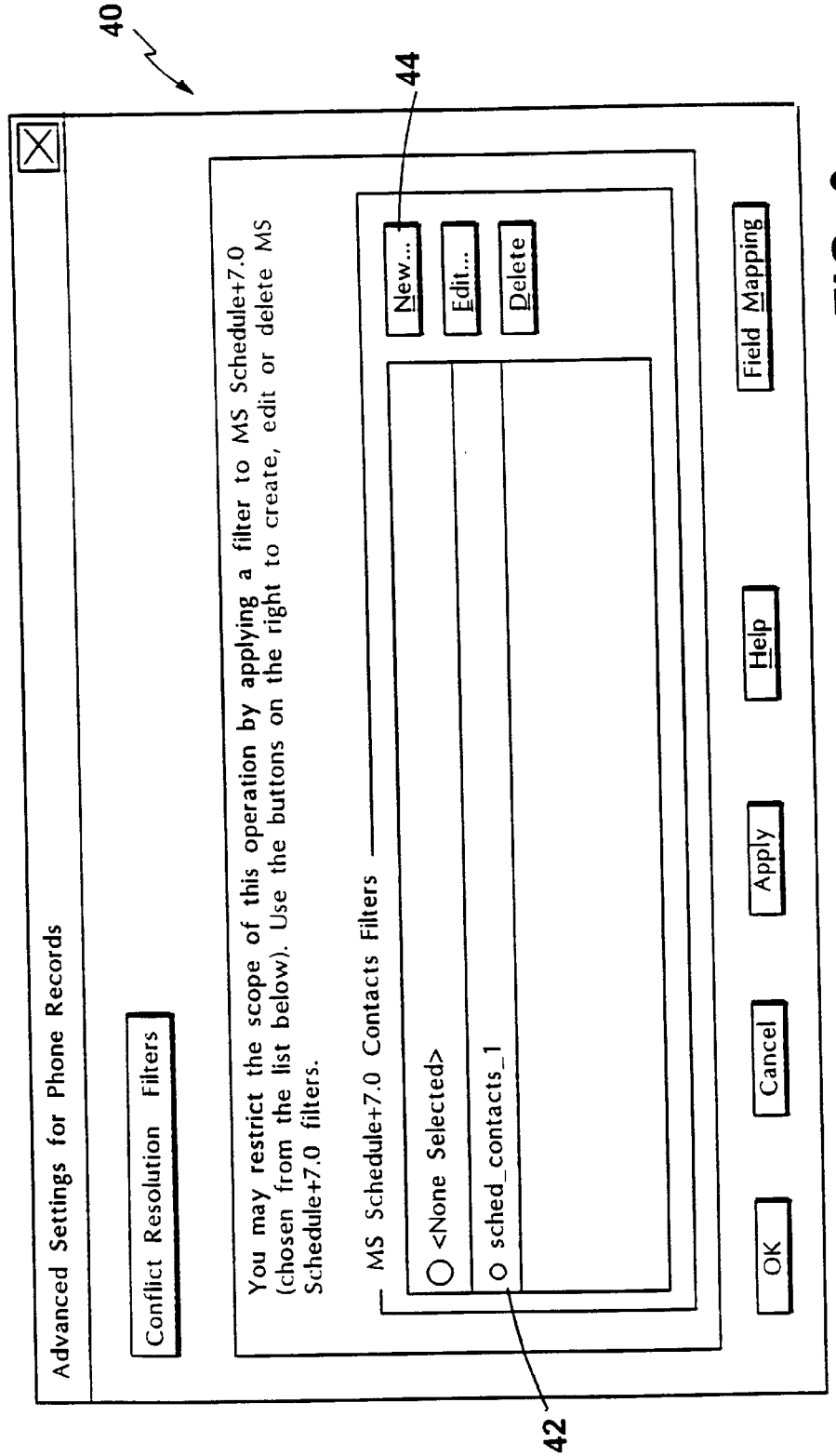

40
44
42
Advanced Settings for Phone Records
Conflict Resolution
Filters
You may restrict the scope of this operation by applying a filter to MS Schedule+7.0 (chosen from the list below). Use the buttons on the right to create, edit or delete MS Schedule+7.0 filters.
MS Schedule+7.0 Contacts Filters
<None Selected>
sched_contacts_1
New...
Edit...
Delete
Field Mapping
Help
Apply
Cancel
OK

FIG. 6

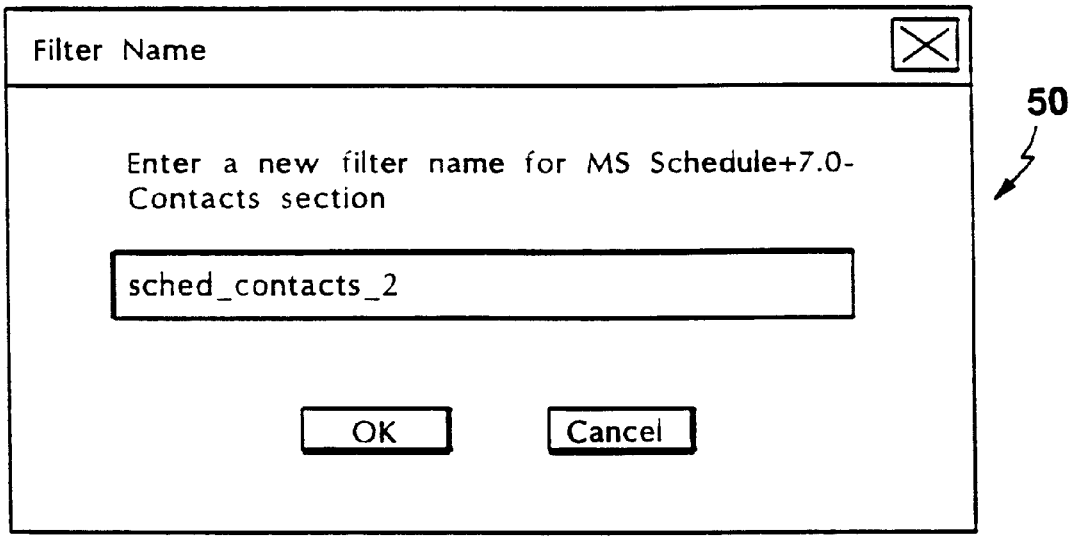

Filter Name
Enter a new filter name for MS Schedule+7.0-Contacts section
sched_contacts_2
OK
Cancel
50

FIG. 7

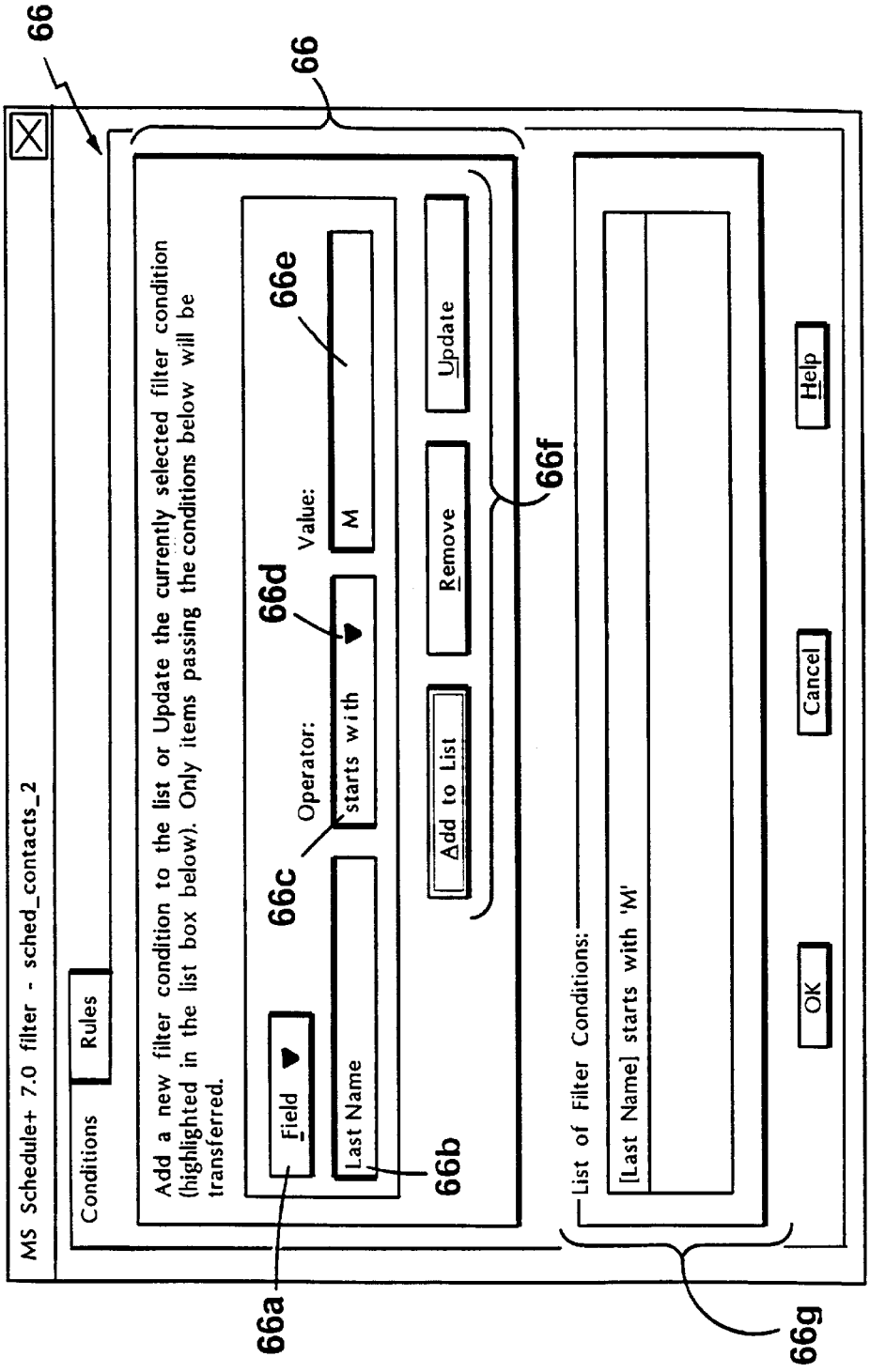

66
66
MS Schedule+ 7.0 filter - sched_contacts_2
Conditions
Rules
Add a new filter condition to the list or Update the currently selected filter condition (highlighted in the list box below). Only items passing the conditions below will be transferred.
Field
Last Name
Operator:
starts with
Value:
M
Add to List
Remove
Update
66a
66b
66c
66d
66e
66f
List of Filter Conditions:
[Last Name] starts with 'M'
OK
Cancel
Help
66g

FIG. 8

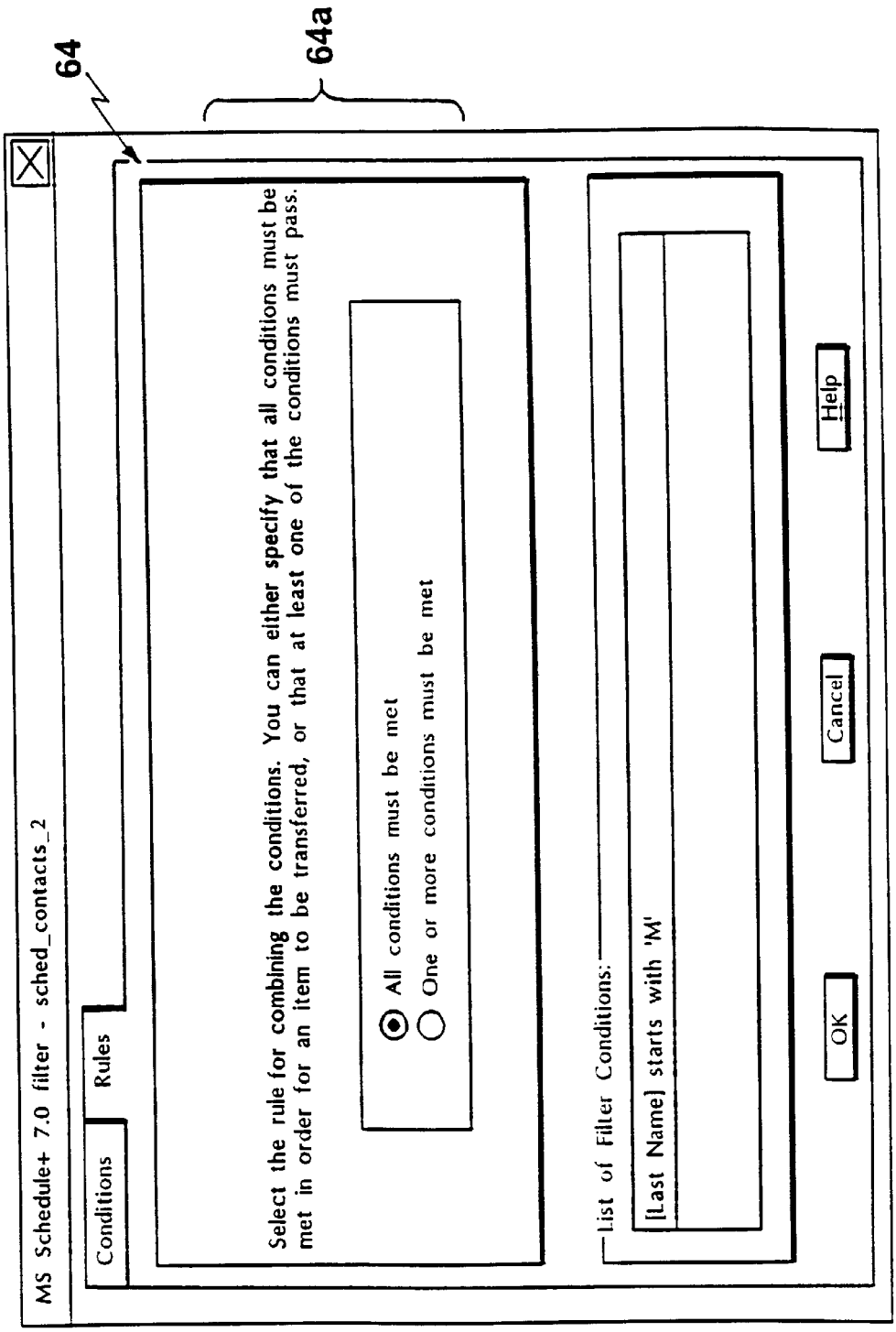

64
64a
MS Schedule+ 7.0 filter - sched_contacts_2
Conditions
Rules
Select the rule for combining the conditions. You can either specify that all conditions must be met in order for an item to be transferred, or that at least one of the conditions must pass.
All conditions must be met
One or more conditions must be met
List of Filter Conditions:
[Last Name] starts with 'M'
OK
Cancel
Help

FIG. 9

Filter language specification

Expression = Condition1 [AND Condition2] ... [OR Condition3] ...

Condition = ARG1 OP ARG2

OP = OP_SET_1 | OP_SET_2 | OP_SET_3 | OP_SET_4 | OP_SET_5 | OP_SET_6

OP_SET_1 = EQ | LE | GE | NE | LT | GT

OP_SET_2 = OP_SET_1 TODAY - | OP_SET_1 TODAY +

OP_SET_3 = OP_SET_1 NOW - | OP_SET_1 NOW +

OP_SET_4 = STARTS_WITH | CONTAINS | DOES_NOT_CONTAIN | IS_EMPTY | IS_NOT_EMPTY

*OP_SET_5 = + | - | * | / | %*

OP_SET_6 = IS

For Dates - ARG1 OP ARG2:		
[Date Fieldname]	*OP_SET_1*	*'YYYYMMDD' | [Date Fieldname2] | TODAY*
[Date Fieldname]	*OP_SET_2*	*integer*
For Times - ARG1 OP ARG2		
[Time Fieldname]	*OP_SET_1*	*'HHMM' | [Time Fieldname2] | NOW*
[Time Fieldname]	*OP_SET_3*	*integer*
For TextStrings - ARG1 OP ARG2		
[String Fieldname]	*OP_SET_1*	*'textstring' | [String Fieldname2]*
[String Fieldname]	*OP_SET_4*	*'textstring'*
For Booleans - ARG1 OP ARG2		
[Boolean Fieldname]	*OP_SET_6*	*TRUE*
[Boolean Fieldname]	*OP_SET_6*	*FALSE*
For Numbers - ARG1 OP ARG2		
[Number Fieldname]	*OP_SET_1*	*integer | float*
[Number Fieldname]	*OP_SET_5*	*integer | float*

FIG. 10

200. FOR each Record in history file
201. Load record
202. Write record to Workspace
203. Next

FIG. 11

```
300.  IF Use_Filter = TRUE and R_Application_Is_Filtering = FALSE THEN
301.        FOR each Record in the remote database
302.              Load record
303.              Filter the loaded record
304.              IF record passes the filter THEN mark as PASSED_FILTER
305.              ELSE mark as FAILED_FILTER
306.              Send record to synchronizer
307.              In Synchronizer: Write record to Workspace
308.        Next
309.  ELSE IF Use_Filter = TRUE and R_Application_Is_Filtering = TRUE THEN
310.        Send the filter expression to R_Application
311.        Load filtered records
312.        IF the record passes current filter THEN Mark as PASSED_FILTER ELSE Mark as
            FAILED_FILTER
313.        Send records to synchronizer
314.        In Synchronizer: Write records to Workspace
315.  END IF
```

FIG. 12

```
350.  Form all records in the workspace into CIGs
351.  For each CIG
352.        Compare the records in CIG
353.        Determine synchronization outcome
354.        IF a synchronization outcome is a conflict THEN
355.        IF one of the database records in the CIG does not pass the current filter, THEN skip CIG and
            mark results as DO NOT UPDATE any of the records
356.              ELSE resolve conflict by reference to a user-selected rule or input from the user
357.        END IF
358.        IF the most up to date record fails the filter, THEN mark all records as having failed the
            current filter
359.        IF the filter expressions contains an unmapped field and one of the database records in the CIG
            are marked as having failed the filter, THEN mark all records as having failed the filter
360.        IF a fanned out recurring record is partially outside of the current filter, THEN mark the
            record to be fanned when being unloaded and delete previous fanned nonrecurring records
361.  Next
```

FIG. 13

```
400.  FOR each remote database record
401.      IF Use_Filter = TRUE and the filter is a static filter THEN
402.          IF record is marked as FAILED_FILTER THEN
403.              Delete record on the remote database
404.          Else IF the record is marked as PASSED_FILTER THEN add, delete, or modify
              record according to results of synchronization obtained during CAAR analysis
405.      ELSE IF Use_Filter = TRUE and the filter is a dynamic filter THEN
406.          IF record fails the current filter THEN
407.              Delete record on the remote database
408.          Else IF the record passes the current filter THEN add, delete, or modify record
              according to results of synchronization obtained during CAAR analysis
409.      END IF
410.  Next
```

FIG. 14

```
450.  FOR each local database record
451.      IF Use_Filter = TRUE and the filter is a static filter THEN
452.          IF record is marked as FAILED_FILTER THEN
453.              IF CAAR outcome is to modify the record then modify the record on the
                  local database
454.          Else IF the record is marked as PASSED_FILTER THEN add, delete, or modify
              record according to results of synchronization obtained during CAAR analysis
455.      ELSE IF Use_Filter = TRUE and the filter is a dynamic filter THEN
456.          IF record fails the current filter but marked as PASSED_FILTER THEN
457.              IF CAAR outcome is to modify the record then modify the record on the
                  local database
458.          Else IF the record passes the current filter THEN add, delete, or modify record
              according to results of synchronization obtained during CAAR analysis
459.      END IF
460.  Next
```

FIG. 15

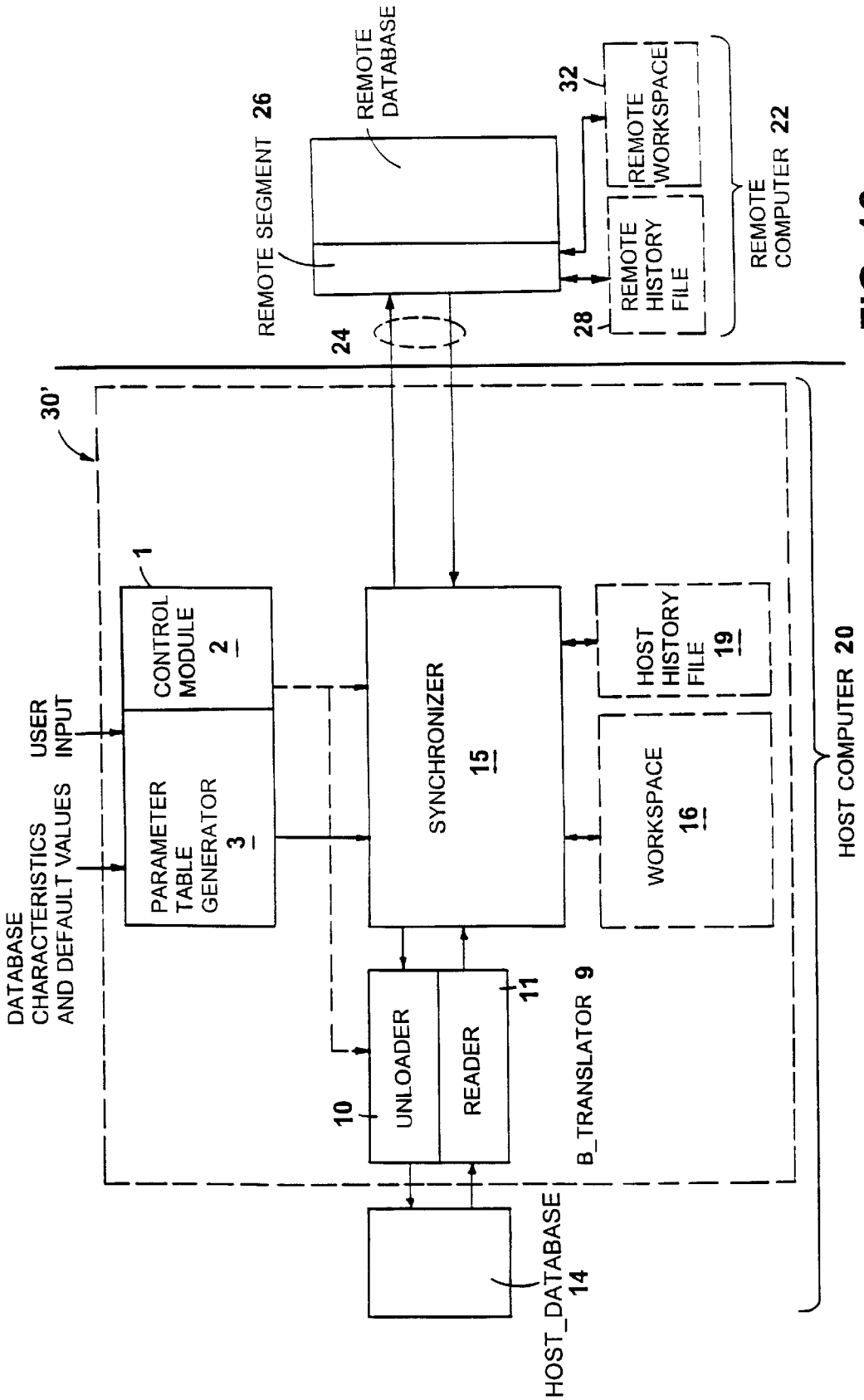

REMOTE SEGMENT 26
REMOTE DATABASE
32
REMOTE WORKSPACE
REMOTE HISTORY FILE
REMOTE COMPUTER 22
24
28
30'
1
CONTROL MODULE 2
PARAMETER TABLE GENERATOR 3
USER INPUT
DATABASE CHARACTERISTICS AND DEFAULT VALUES
SYNCHRONIZER 15
HOST HISTORY FILE 19
WORKSPACE 16
HOST COMPUTER 20
10
UNLOADER
READER
11
B_TRANSLATOR 9
HOST_DATABASE 14

FIG. 16

SYNCHRONIZATION OF DATABASES USING FILTERS

CROSS REFERENCE TO RELATED APPLICATIONS

This application is a continuation in part of "Synchronization of Databases with Date Range," Serial No. 08/748,645, filed Nov. 13, 1996.

REFERENCE TO MICROFICHE APPENDIX

An appendix (appearing now in paper format to be replaced later in microfiche format) forms part of this application. The appendix, which includes a source code listing relating to an embodiment of the invention, includes 1024 frames on 11 microfiche.

BACKGROUND

This invention relates to synchronizing databases.

Databases are collections of data entries which are organized, stored, and manipulated in a manner specified by applications known as database managers (hereinafter also referred to as "Applications"; hereinafter, the term "database" also refers to a database manager combined with a database proper). The manner in which database entries are organized in a database is known as the data structure of the database. There are generally two types of database managers. First are general purpose database managers in which the user determines (usually at the outset, but subject to future revisions) what the data structure is. These Applications often have their own programming language and provide great flexibility to the user. Second are special purpose database managers that are specifically designed to create and manage a database having a preset data structure. Examples of these special purpose database managers are various scheduling, diary, and contact manager applications for desktop and handheld computers. Database managers organize the information in a database into records, with each record made up of fields. Fields and records of a database may have many different characteristics depending on the database manager's purpose and utility.

Databases can be said to be incompatible with one another when the data structure of one is not the same as the data structure of another, even though some of the content of the records is substantially the same. For example, one database may store names and addresses in the following fields: FIRST_NAME, LAST_NAME, and ADDRESS. Another database may, however, store the same information with the following structure: NAME, STREET_NO., STREET_NAME, CITY_STATE, and ZIP. Although the content of the records is intended to contain the same kind of information, the organization of that information is completely different.

Often users of incompatible databases want to be able to synchronize them with one another. For example, in the context of scheduling and contact manager Applications, a person might use one application on a desktop computer at work while another on his handheld computer or his laptop computer while away from work. It is desirable for many of these users to be able to synchronize the entries on one with

entries on another. U.S. patents of the assignee hereof, Puma Technology, Inc. of San Jose, Calif. (U.S. Pat. No. 5,392,390, hereinafter, "the '390 patent", incorporated by reference herein; and U.S. Pat. No. 5,684,990, filed on Jan. 11, 1995, incorporated by reference herein) show two methods for synchronizing incompatible databases and solving some of the problems arising from incompatibility of databases.

SUMMARY

In one general aspect, the invention features a computer program for synchronizing at least a first and a second database. A plurality of records of the first database fitting a selected criterion are identified. At least one of the identified records of the first database is then synchronized with a record of the second database.

In another general aspect, the invention features a computer program for synchronizing at least a first and a second database. On a computer display, a record selection criteria input region is displayed for a user to input a record selection criteria. Then, the first database is synchronized with the second database using the record selection criteria.

Preferred embodiments may include one or more of the following features.

Records representative of the records of the first and second databases during a prior synchronization are stored in a history file. In that case, when synchronizing the identified records of the first database with the records of the second database, the history file is used.

Records of the second database may also be identified based on a selected criterion. In that case, the identified records of the first database are synchronized with the identified records the second database.

Records of the first and second databases may include text, number, boolean, binary, date, and time fields. The criteria for identifying the records in turn may include text, number, boolean, binary, date and time criteria with which the fields of the records and databases are compared.

The first database can be located on a first computer and the second database located on a second computer. At the first computer, it is then determined whether a record of the first database has been changed or added since a previous synchronization, using a first history file located on the first computer including records representative of records of the first database at the completion of the previous synchronization. If the record of the first database has not been changed or added since the previous synchronization, information which the second computer uses to identify the record of the first database to be unchanged is sent from the first computer to the second computer. Additionally, the identification of the records of the first database based on the selected criterion may be performed at either the first or second computer.

Based on data reflecting whether the records of the first database have been added or changed since a previous synchronization, it may be determined whether the records of the first database have been changed or added since a previous synchronization. If one of the records of the first database has not been changed or added since the previous synchronization, a synchronization with records of the second database using a record representative of the one record at the time of a previous synchronization is performed. The representative record is stored in a history file containing records reflecting the contents of records of the databases at the time of a previous synchronization.

A second plurality of the records of the first database failing to fit the selected criterion may be deleted.

The selected criterion may have a current value during a current synchronization being different from a previous value during a previous synchronization. In that case, a plurality of records of the second database may be updated, based on results of the synchronization, where the plurality of records of the second database fit the previous value of the selected criterion but fail to fit the current value of the selected criterion.

A third database may be synchronized with the second database by identifying a plurality of records of a third database fitting a second selected criterion and synchronizing at least one of the identified records of the third database with a record of the second database. The record of the second database can include a code identifying the record as having originated from the third database.

A record selection criteria may be transmitted to a database manager which manages the first database and the database manager may select records of the first database fitting the record selection criteria. The database manager may then transmit the selected records to the synchronization program. The records of the first database fitting the record selection criteria may also be selected at a synchronization program.

The selected criterion may be, for example, a filter or filter expression which a record must match or fit in order for that record to pass the filter expression.

Embodiments of the invention may include one or more of the following advantages.

Users of various embodiments of the invention can use those embodiments to achieve a variety of ends. For example, handheld computers typically have limited storage capacity. Using the filtering capability of some embodiments, a user can limit the records stored on the handheld computer to only those records which fit a selected filter.

A user can also use a filter in some embodiments to increase the speed of synchronization. For example, it may be the case that the data transfer link between the two computers has a low data transfer rate. Therefore, the user by using a filter reduces the number of records to be transferred from one computer to the other.

A user can also use the filtering mechanism to synchronize some of the records of his or her database with the related records of another user's database, without affecting the unrelated records. For example, consider the situation where two database users work on a shared project or take a joint business trip on behalf of their enterprise. These two database users may desire to synchronize their databases but only with respect to those records relating to that project or trip. By using a filter, they may limit the synchronization between the two databases to records which relate to the project or the trip, without affecting other records in their respective databases.

The invention may be implemented in hardware or software, or a combination of both. Preferably, the technique is implemented in computer programs executing on programmable computers that each include a processor, a storage medium readable by the processor (including volatile and non-volatile memory and/or storage elements), at least one input device, and at least one output device. Program code is applied to data entered using the input device to perform the functions described above and to generate output information. The output information is applied to one or more output devices.

Each program is preferably implemented in a high level procedural or object oriented programming language to communicate with a computer system. However, the programs can be implemented in assembly or machine language, if desired. In any case, the language may be a compiled or interpreted language.

Each such computer program is preferably stored on a storage medium or device (e.g., ROM or magnetic diskette) that is readable by a general or special purpose programmable computer for configuring and operating the computer when the storage medium or device is read by the computer to perform the procedures described in this document. The system may also be considered to be implemented as a computer-readable storage medium, configured with a computer program, where the storage medium so configured causes a computer to operate in a specific and predefined manner.

Other features and advantages of the invention will become apparent from the following description of preferred embodiments, including the drawings, and from the claims.

BRIEF DESCRIPTION OF THE DRAWING

FIG. **1** shows two computers connected via a data transfer link.

FIG. **2** is a schematic drawing of the various modules constituting an embodiment of a synchronization program.

FIG. **3** is a representation of a workspace data array used by the synchronization program of FIG. **2**.

FIG. **4** is pseudocode for a Translation Engine Control Module of the synchronization program of FIG. **2**.

FIG. **5** is pseudo code for the steps taken by Parameter Table Generator module of the synchronization program of FIG. **2**.

FIG. **6** shows a filter selection graphical user interface (GUI) window.

FIG. **7** shows a filter name input graphical user interface (GUI) window.

FIGS. **8** and **9** show a filter criteria input graphical user interface (GUI) window.

FIG. **10** shows a table detailing semantics of a filter language used in the synchronization program of FIG. **2**.

FIG. **11** is pseudocode for loading a history file.

FIG. **12** is pseudocode for the steps taken by a translator to load records of a remote database.

FIG. **13** is pseudocode for the steps taken by the synchronizer module of the synchronization program of FIG. **2** for performing Conflict Analysis and Resolution when synchronizing using a filter expression.

FIG. **14** is pseudocode for the steps taken by a translator to unload records to a remote database.

FIG. **15** is pseudocode for the steps taken by a translator to unload records to a local database.

FIG. **16** is a schematic drawing of the various modules constituting an embodiment of a distributed synchronization program.

DESCRIPTION

We will describe embodiments of the invention in detail below, but briefly, referring to FIGS. **1** and **2**, a synchronization program **100** runs on a local computer **20** (e.g. a desktop or server computer) which is typically connected to a remote computer **22** (e.g. a handheld or notebook computer) via a data transfer link **24** enabling the computers to transfer data between them. Data transfer link **24** may be a serial infrared link, serial cable, modem and telephone line

combination, or other such data transfer links. Each of the local and remote computers stores a corresponding local or remote database, which may, for example, be a scheduling database (such as those sold under the tradenames Microsoft Schedule+and Lotus Organizer).

Synchronization program **100** synchronizes the records of the local and remote databases typically using a history file that contains records reflecting the records of the two databases at the end of a previous synchronization. The synchronization program uses the history file to determine, for example, which records have been changed, added or deleted since the previous synchronization and which records of the two databases correspond to one another.

Synchronization program **100** allows the user to input a filter expression. Generally, a filter or filter expression may considered to be a set of conditions or criteria which a record must match or fit in order for that record to pass the filter expression. A record that does not fit those criteria therefore fails the filter.

Synchronization program **100** uses the filter expression to identify and mark which records of the local and remote databases pass or fail the filter expression. The two databases are then synchronized. After synchronization, synchronization program **100** uses the results of synchronizing the two databases to add, modify, or delete the records of the two databases. At this point, the user can, for example, select to have those records falling outside the filtering criteria to be deleted, not to be affected at all by the synchronization program, or be treated in other manner, as will be described below. In other embodiments, synchronization program **100** uses the filter expression to identify and mark the records of only one of the databases.

We will now describe in detail the structure of synchronization program **100** and the method it uses to synchronize the local and remote databases using filter expressions. FIG. **2** shows the relationship between the various modules of an embodiment of synchronization program **100**. Translation Engine **1** comprises Control Module **2** and Parameter Table Generator **3**. Control Module **2** is responsible for controlling the synchronizing process by instructing various modules to perform specific tasks on the records of the two databases being synchronized. (FIG. **4** shows the steps taken by this module.)

Parameter Table Generator **3** is responsible for creating a Parameter_Table **4** which is used by all other modules for synchronizing the databases. Generally, Parameter_Table **4** stores various information which may be used by the modules of the synchronization program. The information stored in Parameter_Table **4** includes user preferences, the names and locations of the databases, and the names and locations of various files stored on disk including the name and location of the history file from the previous synchronization. Parameter Table Generator **3** also provides the user with various graphical user interface windows for inputting filter expressions to be used during synchronization. Parameter Table Generator **3** converts such user input filter expressions or previously stored filter expressions into data structures which may then be used by various modules of synchronization program **100** during synchronization. The steps taken by Parameter Table Generator **3** in relation to filter expressions will be described in detail below.

Synchronizer **15** has the primary responsibility for carrying out the core synchronizing functions. It is a table-driven code which is capable of synchronizing various types of databases whose characteristics are provided in Parameter_Table **4**. Synchronizer **15** creates and uses workspace **16** (also shown in FIG. **3**), which is a temporary data array used during the synchronization process.

Synchronization program **100** has two translator modules **5** and **9** which are generally responsible for data communication between synchronization program **100** and databases **13** and **14**. Translator (L_translator) **5** is assigned to the local database (L_database) **13** and translator **9** (R_translator) to the remote database (R_database) **14**. Each of the database translators **5** and **9** comprises three modules: reader modules **6** and **10** (L_reader and R_reader) which load (or read) records from databases **13** and **14**; unloader modules **8** and **12** (L_unloader and R_unloader) which analyze and unload records from workspace **16** into databases **13** and **14**; and sanitizing modules **7** and **11** (L_sanitizer and R_sanitizer) which analyze the records of the opposing database when they are loaded into the workspace and modify them according to rules of data value of the modules's own database. Briefly stated, rules of data value are generally rules that define the permitted content of the fields of the records of a database. An example of such a rule would be that no more than 100 characters may be present in a field, or that content of a field designating a priority for a "to do" item should be limited to 1, 2, or 3. Sanitizing a record is to change the content of the fields of a record of one database to conform to the rules of data value of another database. Rules of data value and sanitization are described in detail in the following commonly owned U.S. patent applications, incorporated in their entirety by reference, "Synchronization of Recurring Records in Incompatible Databases", Ser. No. 08/752,490, filed on Nov. 13, 1996 (hereinafter, "application '490"); "Synchronization of Databases with Record Sanitizing and Intelligent Comparison," Ser. No. 08/749,926, filed Nov. 13, 1996 (hereinafter, "application '926"); "Synchronization of Databases with Date Range," Ser. No. 08/748,645, filed Nov. 13, 1996 (hereinafter, "application '645").

In the described embodiment, the modules of L_translator **5** are designed specifically for interacting with local database **13** and local application **17**. The design of the modules of L_translator **5** is specifically based on the record and field structures and the rules of data value imposed on them by the local application, the Application Program Interface (API) requirements and limitations of local application **17** and other characteristics of the local database and application. The same is true of the modules of R translator **9**. These translators are typically not able to interact with other databases or Applications and are only aware of the characteristics of the database and application for which they are designed. Therefore, when the user chooses two applications for synchronization, Translation Engine **1** chooses the two translators which are able to interact with those applications. In an alternate embodiment, the translators can be designed as table-driven codes, where a general translator is able to interact with a variety of applications and databases based on supplied parameters.

Having described the structure of synchronization program **100** in reference to its various modules, we will now describe the operation of synchronization program **100**. During synchronizing the two database, Control Module **2** instructs the various modules in synchronization program **100** to perform specific tasks. We will describe the operation of synchronization program **100** by describing the steps taken by Control Module **2** (as set out in the pseudo code in FIG. **4**) and describing in detail the actions by the various modules as they are instructed by Control Module **2**.

Referring to FIG. **4**, in the first step of synchronizing the two databases, Control Module **2** instructs the Parameter

Table Generator **3** to create parameter table **4** (Step **100**). In this step, as part of creating parameter table **4**, Parameter Table Generator **3** obtains from the user a filter expression, if any, to be used during synchronization or alternatively accesses a previously stored filter expression, if any.

FIG. **5** is pseudo code for the steps taken by Parameter Table Generator **3** in relation to filter expressions. In step **150**, Parameter Table Generator **3** determines from the user whether a filter should be used during the current synchronization. FIG. **6** shows a filter selection window **40** which the user uses to select whether to use a filter for the current synchronization. If the user selects to use a filter (step **151**), Parameter Table Generator **3** next determines the filter to be used (step **152**). This filter may be a previously stored filter expression (e.g. filter **42** in FIG. **6**) or a filter expression which the user inputs (e.g. by selecting "New" button **44** in FIG. **6**).

For a new filter, the user uses a filter name input window **50** to input the filter's name (step **153**), shown in FIG. **7**. The user then uses a filter criteria input window **60** (step **154**), shown in FIGS. **8** and **9**, to input the filter expression for the new filter. We will describe window **60** in further detail below. Parameter Table Generator **3** next stores the current date and time in the FILTER_CHANGED_TIMESTAMP parameter (step **155**). This parameter is used to determine whether a filter has changed since a previous synchronization. Parameter Table Generator **3** then assigns a unique filter identifier code to the filter expression, which is then used to identify the filter (step **156**).

If the user selects to use a previously stored filter expression, Parameter Table Generator **3** obtains the name of the filter from the user (step **157**) and then retrieves the filter expression and the unique filter identifier code of that filter (step **158**). If the user selects to edit the filter, Parameter Table Generator **3** displays the filter expression in window **60** and allows the user to edit the filter expression (step **159**). In this step, Parameter Table Generator **3** also stores the current date and time in the FILTER_CHANGED_TIMESTAMP parameter, as in step **155**.

In step **160**, Parameter Table Generator **3** sets the FILTER_ID parameter to the unique filter identifier of the filter that was selected. The modules of synchronization program **100** use FILTER_ID parameter to determine the correct filter to use. Parameter Table Generator **3** next sets USE_FILTER flag (step **161**). This flag indicates to various module of synchronization program **100** that a filter is to be used during synchronization and causes the appropriate modules to take the necessary steps, as will be described in further detail below.

During synchronization, the filter may be applied to the records of the database by either the translator for that database or by the database manager application of that database, if the database manager application is capable of applying a filter. In step **162**, Parameter Table Generator **3** parses the filter expression into a filter token array which the translators use when filtering records of the databases and history file. The filter token array and the manner in which it is used will be described in detail below. Parameter table generator **3** will next create Parameter_Table **4**, as described in detail in the '490, '926 and '645 applications.

Having described the steps taken by Parameter Table Generator **3** with respect to a filter to be applied during synchronization, we will now describe in detail graphical user interface (GUI) windows displayed for entering the filter expressions, the semantics of filter language used in the described embodiment, the manner in which inputted filter criteria are stored, and the method used by translators to determine whether a record passes the filter. However, it should be noted that other filtering languages and methods may also be used to filter records during synchronization.

Generally, synchronization program **100** uses two types of filters: static and dynamic filters. Static filters have a fixed and unchanging filter expression. For example, the filter "appointments in 1997" is a static filter. It will always the cover the appointments in 1997. Dynamic filters have a changing filter threshold value, which may depend on a changing parameter. For example, the filter "appointments from today until a year from today" is a dynamic filter because the threshold value of the filter changes as the value of "today" changes. (It should be noted that the above examples are also examples of date range filters. Date range filters filter records based on whether the dates of the records fall within a range of dates specified by the filter.)

In the case of dynamic filters, Parameter Table Generator **3** uses the dynamic filter to create two filter expressions to be used during synchronization. The first filter expression, which we will refer to as the current filter, is based on the value of the filter for the current synchronization. For example, in the case of date range filters based on the value of the current synchronization's date, the value of the current filter will be based on the current synchronization's date. (Date range filters and a method of synchronizing databases using them are described in detail in the '490, '926 and '645 applications.) Since a dynamic filter is a changing filter, records which were previously within the filter may not be within the current filter. However, those records and any changes in those records should be used during the current synchronization since the user likely treated those records as being validly within the filter and might have modified them. Therefore, Parameter Table Generator **3** creates a second filter expression, which we will refer to as the loading filter, which combines the value the current filter with the value of the filter as it was during a previous synchronization. For example, in the case of a dynamic date range, the loading filter would be a concatenation of the current filter and the filter based on the date of the previous synchronization. The manner in which these two filters are used will be described below.

In synchronization program **100**, a filter expression applied to both local and remote databases. However, the filter expression is typically inputted and stored based on the list of fields of one of the databases—in the described embodiments, the local database. A field map which maps the fields of the two databases onto one another is used by synchronization program **100** to apply a filter expression based on the field list of one of the databases to the fields in the records of the other database, as will be described in detail below.

The table in FIG. **10** shows the specification of the semantics of the filter language. We define a filter expression (or filter criteria) as a group of one or more filter conditions (or filter criterion). In the filter language described here, when a record is evaluated against a filter condition, the result may be either a boolean value (i.e. TRUE or FALSE) or a numeric value (e.g. 1, 2, 3). However, the final result of evaluating a record against the filter expression is a boolean value which indicates whether the record has passed or failed the filter.

A filter condition may be considered to be a sentence having three parts: <evaluated operand> <filter operator> <filter threshold operand>, where the evaluated operand is the operand to be evaluated against the filter, filter threshold

operand is the threshold value of the filter condition, and the operator is the filtering function to be performed between the evaluated operand and filter threshold operand. For example, the following is a filter condition: <date field> <is greater than or equal to> <TODAY>.

An operand may have one of two values: a value inputted by a user or a value taken from a record to be evaluated. To indicate that the value of an operand is to be taken from a field in a record, the name of that field is used as the operand. In the described filter language, a field name is enclosed with brackets—e.g. [Start Date]. During evaluation of a record against the filter, the value stored in that field of the evaluated record will be used as the operand.

In the case where the operand has a user inputted value, the operand contains that value—e.g. 'Smith', 456–7896, or 'TODAY'. In the described filter language, the user-inputted operands may have one of the following types of values, which is coextensive with the possible field values of the local and remote databases: DATE, TIME, TEXTSTRING, BOOLEAN or NUMBER.

We will now describe in detail an example of the range of values the various types of user-inputted operands in the described filter language may have. However, it should be noted that other embodiments may have other ranges and limitations. In the described embodiment, DATE operands are formatted as 'YYYYMMDD' (in some embodiments, single quotes must be included)—example '19980101' is New Years Day of 1998. DATE operands may also have the value TODAY, which indicates the date for today obtained from the operating system of the computer. TIME operands are formatted as 'HHMM' on a 24-hour clock basis (in some embodiments, single quotes must be included)—for example, '0600' represents 6:00 AM and '1300' represents 1:00 PM. TIME operands may also have the value NOW, which indicates the current time obtained from the operating system of the local computer.

TEXTSTRING operands can contain any text value (in some embodiments, single quotes must be included to indicate that the value is a text string)—examples are 'Puma Technology, Inc.', '15', and '#$%'. BOOLEAN operands must be of value TRUE or FALSE (no use of single quotes). NUMBER operands may be any integer or floating point number (in some embodiments, single quotes are not used for NUMBER operands).

In the described filter language, the available filter operators are organized into seven different operator sets. The first operator set (also referred to as "OP_SET_1") includes the following filter operators: EQ (equal), LE (less than or equal), GE (greater than or equal), NE (not equal), LT (less than) and GT (greater than). This set of operators may be used for all of the various types of operands, provided that both operands involved in the filter condition are of the same type.

The second operator set (also referred to as "OP_SET_2") is to be used only when evaluated operand is of the type DATE. OP_SET_2 provides for using dynamic filters for operands of type DATE (e.g. dynamic date range filters). OP_SET_2 is made up of all combinations of the filter operators in OP_SET_1 and the variables TODAY– and TODAY+. An OP_SET_2 filter operator is followed by a filter threshold operand whose value is a selected number of days. The variable TODAY+ in an OP_SET_2 filter operator then indicates the number of days in the filter threshold operand is to be added to the date of the current synchronization prior using the OP_SET_1 filter operator to evaluate filter. For example, consider the filter condition: <appointment date> <LT TODAY+> <3>. In this filter expression, 3 days are added to the date of the current synchronization to obtain the date of the third day after today and then the filter operator LT is used to determine whether the appointment date is less than the date of the third day after today.

The third operator set (also referred to as "OP_SET_3") is similar the second operator set and is used only when both operands are of the type TIME. OP_SET_3 provides for using dynamic filters for operands of type TIME. OP_SET_3 includes all combinations of filter operators in OP_SET_1 with the variables NOW– and NOW+. The variable NOW represents the time of the current synchronization, typically obtained from the operating system. An OP_SET_3 filter operator is followed by a filter threshold operand whose value is a selected number of seconds. The variables NOW+ and NOW– are used in the same manner as the variables TODAY+ and TODAY–.

The fourth operator set (also referred to as "OP_SET_4") may be used when both operands are of the type TEXTSTRING. OP_SET_4 includes the following operators: STARTS_WITH, CONTAINS, DOES_NOT_CONTAIN, IS_EMPTY and IS_NOT_EMPTY.

The fifth operator set (also referred to as "OP SET_5") may be used when both operands are of the type NUMBER. OP_SET_5 includes the following operators: + (addition), – (subtraction) * (multiplication), / (division) and % (modulus).

The sixth operator set (also referred to as "OP_SET_6") may be used only when both operands are of the type BOOLEAN. OP_SET_6 includes only the operator IS.

The seventh operator set (termed OP_SET_7) may be used to combine two filter conditions. OP_AND_OR includes the relational operators AND and OR. Filter conditions may be combined using the seventh operator set to form filter expressions. In the described embodiment, the operand from the seventh operator set are used so as to achieve one of two results: a record must either meet all of the filter conditions in a filter expression or only one of the filter conditions. In other embodiments, more complex filter expressions may be permitted. In such embodiments, the order of evaluation may follow a predetermined order of evaluation (e.g. the order of evaluation in the 'C' programming language) which may in turn be modified by parentheses.

Following are several exemplary filter expressions based on the above described filter language:

[Full Name] CONTAINS 'Smith' AND [Private Flag] IS FALSE

([Priority] / 2) GT 0

[Alarm Date] EQ '19980101' AND ([Regarding CONTAINS 'meeting' OR [Regarding] CONTAINS 'training') AND [Location] CONTAINS 'Boston'

Referring back to FIGS. **8** and **9**, the user uses the filter expression input window **60** to enter the filter expression to be used during synchronization. Filter expression input window **60** includes two so-called tabs (e.g. used in operating systems sold under the tradename Windows by Microsoft Corporation of Redmond, Wash.). Conditions tab **62** (shown in FIG. **8**) is used to input the filter conditions. Rules tab **64** (shown in FIG. **9**) is used to select how the filter conditions should be combined to create the filter expression.

Referring to FIG. **8**, conditions tab **62** includes a filter condition input region **66**. The user can select a field name from a list of field names of the database on which the filter

is based by using pull-down menu **66***a*. Alternatively, the user may type a valid name in field name region **66***b*. The user may also type in a valid filter threshold operand in threshold operand region **66***e*. In filter operator region **66***c*, the user may type a valid filter operator or select one from a filter operator pull-down menu (not shown; only button **66***d* for clicking on to pull-down the menu is shown). It should be noted that the list of available options in pull-down menu **66***d* is limited by the type of operand entered in field name region **66***b* or filter threshold operand region **66***e*.

The user may add a filter condition to the list displayed in filter conditions display region **66***g* or remove a filter condition from that list, by selecting the appropriate button in region **66***f*. The user may also update (i.e. change or edit) a filter condition, by selecting the appropriate "button" in region **66***f*.

Referring to FIG. **9**, in rules tab **64**, more particularly in region **64***a*, the user may select whether a record must either meet all of the filter conditions inputted by the user or only one of the filter conditions in order to pass the filter.

The filter expression input by the user in the filter expression input window **60** is then stored as a filter expression based on the above described filter language.

We will now describe the filtering methodology used in synchronization program **100** to evaluate a record against a filter. Briefly, the filtering methodology used in synchronization program **100** generally has two steps. The first step is parsing the stored filter expression and forming a filter token array which is designed to facilitate evaluating the records against the filter. In synchronization program **100**, Parameter Table Generator **3** performs this step (FIG. **5**, step **162**). The second step is evaluating each record of the database or the history file to be filtered against the filter expression to determine whether the record passes the filter. In synchronization program **100**, the translators performs this step in the case of the local and remote databases. Synchronizer **15** performs this step for the history file. We will now describe each of these steps in detail.

Parameter Table Generator **3** parses the stored filter expression and then forms the filter expression into a filter token array to be used during evaluation of the records against the filter expression. A token in the filter token array is a data structure which represents either an operand (i.e an evaluated operand or a filter threshold operand) or a filter operator. A token contains two pieces of information—the type of the token and the value of the token. The type of the token may be one of the following: TEXTSTRING, DATE, TIME, BOOLEAN, NUMBER, OP_SET_**1**, OP_SET_**2**, OP_SET_**3**, OP_SET_**4**, OP_SET_**5**, OP_SET_**6** or OP_SET_**7**. The value of the token will be the actual content of the token and is stored as a string of characters which is obtained from the filter expression. For example, a token which represents a date filter threshold operand "2/7/1988" will have a DATE type and a "2/7/1988" value. A filter operand GE (greater than or equal to) will have a OP_SET **1** type and a "GE" value. An evaluated operand having a field name "[contact address]" will have a TEXTSTRING type and a "[contact address]" value.

As Parameter Table Generator **3** parses the filter expression, it turns the operands and operators of the filter conditions into tokens. Each filter condition yields three tokens. Parameter Table Generator **3** stores the tokens in the filter token array in a specific order. In the case of each filter condition, the evaluated operand token is stored first, followed by the filter threshold operand token and then the filter operator token. In the case of filter operators from OP_SET_**7** (i.e., the filter operators AND and OR), the two filter conditions connected by the operator are stored first, followed by the filter operator. Ordering the tokens in this manner facilitates evaluating the records against the filter, as will be described below.

As an example, parameter generator **3** parses the following filter expression:

[Alarm Date] EQ '19980101' AND ([Regarding] CONTAINS 'meeting' OR [Regarding] CONTAINS 'training') AND [Location] CONTAINS 'Boston'

into the following token array:

value	type
AND	OP_SET_7
CONTAINS	OP_SET_4
'Boston'	TEXTSTRING
[Location]	TEXTSTRING
AND	OP_SET_7
OR	OP_SET_7
CONTAINS	OP_SET_4
'training'	TEXTSTRING
[Regarding]	TEXTSTRING
CONTAINS	OP_SET_4
'meeting'	TEXTSTRING
[Regarding]	TEXTSTRING
EQ	OP_SET_1
'19980101'	DATE
[Alarm Date]	DATE

As stated above, the second step in filtering the records of a database is evaluating each record of that database against the filter expression. The translator of the database to be filtered performs this step. Both of these modules use the same method of evaluation, which we will now describe.

To evaluate a record against a filter expression, a translator uses a stack, which we will refer to as the evaluation stack. To evaluate the record, starting with the last item in the filter token array, the translator proceeds through the filter token array and pushes copies of the operand tokens onto the evaluation stack. When an operand token is a field name, the translator instead of pushing the field name, pushes the data stored in that field of the record onto the stack. When the translator encounters an operator token in the token array, the translator pops the last two items in the evaluation stack and evaluates the two items based on the filter operator. The translator push the result of the evaluation, which may be either of the type BOOLEAN or NUMBER, onto the evaluation stack. After the translator evaluates the entire filter expression, a single boolean value is left in the stack which indicates whether the record has passed the filter.

We will now provide an example of evaluating a record against a filter. Prior to the evaluating step, Parameter Table Generator **3** would have parsed the following filter expression:

[Last Name] STARTS_WITH 'A' AND [City] EQ 'Boston' into the following token array:

value	type
AND	OP_SET_7
EQ	OP_SET_1
'Boston'	TEXTSTRING
[City]	TEXTSTRING
STARTS_WITH	OP_SET_4
'A'	TEXTSTRING

-continued

value	type
[Last Name]	TEXTSTRING

During filter evaluation, the first two tokens are pushed onto the evaluation stack:

token array	evaluation stack
AND EQ 'Boston' [City] STARTS_WITH	[Last Name] 'A'

Next, the operator STARTS WITH is encountered. Therefore, two operands are popped from the evaluation stack and evaluated using the operator. The result of this evaluation (e.g. TRUE) is pushed back onto the evaluation stack:

token array	evaluation stack
AND EQ 'Boston' [City]	TRUE

Two more operands from token array are pushed onto the evaluation stack:

token array	evaluation stack
AND EQ	TRUE [City] 'Boston'

The operator EQ is next encountered. Therefore, two operands are popped from the evaluation stack and evaluated using the operator. The result of this evaluation (e.g. TRUE) is then pushed onto the evaluation stack.

token array	evaluation stack
AND	TRUE TRUE

Finally, the remaining operator AND is encountered in the filter token array. Two operands are popped from the evaluation stack and evaluated using the operator. The result of this evaluation (i.e. TRUE) is also pushed onto the evaluation stack. Because there are no more tokens in the filter token array, the remaining token on the evaluation stack is the final result of the filter evaluation. This final token indicates that the record being evaluated passes the filter.

As stated briefly above, synchronization program **100** applies a filter expression based the field list of the remote database to the local database. To do so, synchronization program **100** uses a field map to determine which field of the local database corresponds to a field of the remote database used in the filter expression. Field mapping is described in U.S. Pat. No. 5,392,390, incorporated by reference. Briefly, to synchronize records of two databases, it is essential to determine which field or fields of one database should be synchronized with which field or fields of the other database. To accomplish this, a field map is used which correlates the fields of the two databases to one another. It should be noted that not all fields of a database are mapped onto the other database and therefore such unmapped fields are not synchronized with the other database.

In the described embodiment, L_translator **5** uses a filter expression which is based on the field list of the remote database to filter the records of the local database. For every token in the filter token array that contains a field name, L_translator **5** uses the remote database to the local database field map to determine the corresponding field in the local database record being evaluated and pushes the content of that field onto the evaluation stack. It may be the case that the field in the filter expression is an unmapped field and therefore does not have counterpart in the local database record being evaluated. If that is the case, L_translator **5** marks the token and its corresponding operator and operand tokens with a SKIP_EVAL flag. During the evaluation phase if an operator token is marked with a SKIP_EVAL flag, L_translator **5** determines the result of that operation to be TRUE. (If the operation to be skipped is to return a NUMBER type value, L_translator **5** determines the result to be '0' or zero. L_translator **5** then marks the resulting token, and that token's corresponding operator token and other operand token, with a SKIP_EVAL flag.) L_translator **5** applies the filter to the record in this manner until a final result remains in the evaluation stack. In alternative embodiments, the filter expression may be based on the field list of the local database or the user may select the database on whose field list the filter expression is to be based.

As stated above, instead of the translator for the database to be filtered, the database manager application for that database may filter the records of that database. The translator for the database parses the filter expression into a set of instructions formatted for the Application Programmer Interface (API) of the database manager application. In that case, the database application manager transmits to synchronization program **100** only those records that pass the filter.

To determine whether the translator or the application will filter the records of a database, the modules of synchronization program **100** use two types of flags. First, as described above, Parameter Table Generator **3** sets a USE_FILTER flag if a filter is being used (FIG. **5**, step **161**). Second, each of translators **5** and **9** in turn set an appropriate flag, i.e. R_Application_Is_Filtering or L_Application_Is_Filtering, to indicate that the database manager application will apply the filter. If both the USE_FILTER and the R_Application_Is_Filtering (or L_Application_Is_Filtering) flags are set, the flags indicate the remote (or local) database manager application will apply the filter to its database.

Referring back to FIG. **4**, after Parameter Table Generator **3** creates the parameter table, Control Module **2** of the Translation Engine **1** instructs synchronizer **15** to initialize itself (step **101**). Synchronizer **15** in response creates the workspace data array **16**. Control Module **2** of the Translation Engine **1** then instructs synchronizer **15** to load history file **19** into workspace **16** (step **102**). History file **19** is a file that was saved at the end of last synchronization and contains records reflecting the records of the two databases at the end of the previous synchronization. Synchronizer **15**

uses history file **19** during current synchronization to analyze the records of the local and remote database to determine the changes, additions, and deletions in each of two databases since the previous synchronization. Synchronizer **15**, as result of this analysis, then can determine what additions, deletions, or updates need be made to synchronize the records of the two databases.

In various situations, synchronizer **15** does not load history file **19**. For example, if no history file from a previous synchronization exists or if the user chooses to synchronize not using the history file, synchronizer **15** will not load history file **19**. Additionally, the user may wish to use a filter expression that is different from the filter expression used in the previous synchronization (including using no filter expression during the current synchronization). In that case, if one of the database manager applications filtered its database, then synchronizer **15** does not load the stored history file. This is because when a database manager application filters the records of a database, the history file contains only those records which pass the previous filter and not necessarily the records necessary for performing a synchronization using the current filter. Obviously, in the case where a history file is not loaded, synchronizer **15** synchronizes the two databases without using a history file.

FIG. **11** is pseudocode for the steps taken by synchronizer **15** to load history file **19**. For each Record in history file **19** (step **200**), synchronizer **15** first loads the record (step **201**) and then writes the loaded record into workspace **16** (step **202**). Synchronizer **15** repeats these steps until all of the records of the history file are loaded into the workspace.

Referring back to FIG. **4**, after the history file is loaded into the workspace, Control Module **2** instructs R_translator **13** to load the remote database records (step **103**). FIG. **12** is pseudocode for the steps taken by R_translator **13** to load the remote database records. If the USE_FILTER flag is set but R_Application_Is_Filtering flag is not set (step **300**) then R_reader module **11** will apply the filter to the records of the remote database. For each record of the remote database (step **301**), R_reader module **11** of the R_translator first loads the record (step **302**). R_reader module **11** applies the filter expression identified by the parameter Filter_Id to the loaded record (step **303**). If the record passes the filter then R_reader module **11** marks the record as having passed the filter (step **304**). If the record does not pass the filter then R_reader module **11** marks the record as having failed the filter (step **305**). R_reader module **11** then sends the record to synchronizer **15** (step **306**) and synchronizer **15** writes the loaded record into the workspace (step **307**). Steps **302–307** are performed until all records of the remote database are loaded.

If the Use_Filter flag and R_Application_Is_Filtering flags are both set (step **309**), then the remote database application will filter the loaded records. R_reader module **11** converts the filter expression into the format required by remote database application's API and sends the converted filter expression to the remote database application (step **310**). R_reader module **11** then loads the filtered records (step **311**). If a record that was loaded passes the current filter expression, then R_reader module **11** marks that as having passed the filter; otherwise, R_reader module marks the record as having failed the filter (step **312**). Since only those records that have passed the filter are loaded, R_reader module **11** does not mark any of the loaded records as having failed the filter. R_reader module **11** then sends the loaded records to synchronizer **15** (step **313**) and synchronizer **15** writes the loaded records into the workspace (step **314**).

Following loading the remote database records, Control Module **2** instructs L_sanitizer module **8** of L_translator **5** to sanitize the remote database records in the workspace (step **104**).

Control Module **2** of the Translation Engine **1** then instructs the L_translator **5** to load the records from the local database (step **105**). L_translator **5** and synchronizer **15** load records of the local database in the same manner as described for R_translator **9** in reference to FIG. **12**, except for two differences. First, as described above, L_translator **5** filters the records of the local databases using the remote database to local database field map. It should be noted that the field maps are contained within the parameter table and the contents of the parameter table are transmitted to L_translator as read-only data. Second, as synchronizer **15** receives each local database record from the L_reader module **7** of the L_translator **5**, synchronizer **15** maps that record using the local database to remote database map before writing the record into the next available spot in the workspace. This is due to the fact that, in the described embodiment, records in the workspace are stored according to the remote database data structure.

Referring back to FIG. **4**, after all records are loaded into the workspace, Control Module **2** instructs synchronizer **15** to perform a Conflict Analysis and Resolution ("CAAR") procedure on the records in the workspace, which procedure is described in detail in the '490, '926 and '645 applications (step **107**). Briefly, referring to FIG. **13**, synchronizer **15** processes the records in the workspace, including comparing them to one another, in order to form them into groups of related records called corresponding item groups (CIGs). Synchronizer **15** forms the CIGs as it loads the records into the workspace and completes the process as the first step in CAAR (step **350**). Each CIG may include at most one record from each of the databases and the history file. Each record in a CIG may be a recurring or a nonrecurring records. Where a database does not support recurring records, in the case of a recurring event, a CIG would contain related nonrecurring records from that database which together represent that recurring event. Based on this group of nonrecurring records, synchronizer **15** may create a model recurring record for use during the synchronization and include that model recurring record in the CIG. Hereinafter, when referring to a "record" in a CIG, we also intend to refer to such a group of related nonrecurring records in the CIG.

For each CIG (step **351**), synchronizer **15** then compares the records in the CIG to one another, determines their differences, and decides what synchronization action should be taken (step **352**). In essence, synchronizer **15** determines which record in the CIG contains the most current data. Synchronizer **15** then determine what synchronization action should be taken to conform the other records in the CIG to the record with the most current data (i.e. how the other records in the CIG should be changed). Synchronization actions with respect to a record include updating, deleting, adding, or not modifying that record.

We will now provide some examples of the results obtained in the CAAR analysis. If after comparing the records in a CIG, synchronizer **15** determines that the record from the local database is unchanged and the one from remote database is changed, synchronizer **15** determines that the local database record should be changed to conform to the remote database record. Or, if both records are changed (an example of what we refer to as a "conflict" since there is no clear choice of synchronization action), synchronizer **15** may use a user-selected rule to decide what synchronization should be taken. The rule may require, for example,

not modifying either of the records, changing the remote database record to conform to the local database record, or asking the user to resolve conflicts.

In the described embodiment, when a filter expression is used during the synchronization, synchronizer **15** alters the synchronization outcome in at least three cases. First, if the synchronization outcome is that a conflict exists (step **354**), synchronizer **15** determines whether one of the records fails the current filter. If one of the records in the CIG fails the filter, then synchronizer **15** marks all records in the CIG so that they are not updated (step **355**). In other embodiments, synchronizer may use the user-selected rule for resolving conflicts to resolve the conflict. If one of the records in the CIG does not fail the filter, then synchronizer **15** uses the user-selected rule to resolve the conflict (step **356**).

Second, as stated above, synchronizer **15** changes the content of all records in a CIG to that of the record with the most up to date data. Therefore, if the record that contains the most up to date data fails the current filter, then the other records in the CIG when updated will also fail the filter. To resolve this issue, in the described embodiment, instead of updating the records in the CIG, all records in that CIG are marked as having failed filter (step **358**). Therefore, the records are not updated when unloaded to the databases at the end of synchronization, as will be described below.

Third, the filter expression may contain a filter condition based on an unmapped remote database field. As described above, when applying such a filter to the local database records, the local translator evaluates the filter condition containing the unmapped field as having a TRUE value. In that case, some filtered local database records may be marked as having passed the filter while their corresponding remote database records may be marked as having failed the filter, even though the mapped fields of both records may contain the same data. In the described embodiment, if a such a filter expression is used and one of the records in a CIG is marked as having failed the current filter, then synchronizer **15** marks all of the CIG records as failing the filter (step **359**).

As stated above, in some cases, one of the databases may support recurring records while the other database may not. Therefore, a recurring record is fanned into a number of non-recurring records before being unloaded to the database that does not support recurring records. In such a situation, during CAAR, synchronizer **15** examines each recurring record to determine whether there are some fanned nonrecurring records which pass the value of the dynamic filter expression during the previous synchronization but fail the current value of the filter. If so, then the dynamic filter has changed in such a way that part of the set of fanned records fall outside of the current filter. In the described embodiment, in such a situation, synchronizer **15** determines that the recurring record should be fanned again to generate new fanned nonrecurring records and previously fanned nonrecurring records should be deleted. To accomplish this, synchronizer **15** flags the recurring record and the appropriate translator fans the recurring record into the appropriate database and deletes the previous instances (step **360**).

When synchronizer **15** finishes performing CAAR on the records, synchronizer **15** would have determined what synchronization action should be taken with respect all records to be synchronized. The records may then be unloaded into their respective databases. The translators will perform the specific synchronization actions to be taken with respect to the records of the databases. However, prior to doing so, the user is asked to confirm proceeding with unloading (FIG. **4**, steps **108–109**). Up to this point, neither the databases nor the history file have been modified. The user may obtain through the Translation Engine's User Interface various information regarding what synchronization actions will be taken upon unloading.

If the user chooses to proceed with synchronization and to unload, the records are then unloaded. The unloader modules **6,10** of the translators **5,9** perform the unloading for the databases. During unloading, translators may use the filter expression to limit the data that is unloaded to the databases. For example, the translators may unload only those records which fall within the filter expression and delete any record which falls outside of the filter expression. During unloading, synchronizer **15** also creates the history file and unloads the records into the history file. We will now describe the unloading of the records into the databases and the history file in detail.

Control Module **2** of Translation Engine **1** first instructs R_translator **9** to unload remote database records from workspace into the remote database (FIG. **4**, step **110**). FIG. **14** is pseudocode for the steps taken by R_translator **9** to unload the records. For each remote database record in the workspace (step **400**), R_translator **9** first determines whether the Use_Filter is set and the filter is a static filter (step **401**). If that is the case, R_translator **9** determines whether the record passes or fails the filter. (It should be noted that, as described above, some records are marked as failing the filter during CAAR. In that case, during unloading, those records are considered to fail the filter.)

If the record fails the filter, R_translator **9** deletes the record from the remote database. If the record is passes the filter, then R_translator **9** adds, deletes, or modifies the record according to results of synchronization obtained during CAAR analysis (step **404**). If the remote database does not support recurring records or in other circumstances described in detail in the '490, '926 and '645 applications, R_translator **9** in step **404** may fan a recurring record by creating an appropriate number of nonrecurring records corresponding to the recurring record. If, as described above, synchronizer **15** during CAAR marks a recurring record for re-fanning (FIG. **13**, step **360**), R_translator **9** in this step will re-fan the record. When fanning, the number of nonrecurring records would be limited by any date based filters (i.e. any date range) or other filters, so that nonrecurring records falling outside the filter are not created. Additionally, if a user has selected to limit the number of fanned nonrecurring records for each recurring record, R_translator would create only a limited number of instances, as described in more detail in the '490, '926 and '645 applications.

If the Use_Filter is not set or the filter is not a static filter (step **401**), R_translator **9** determines whether the Use-Filter flag is set and the filter is a dynamic filter (step **405**). If the Use-Filter flag is set and the filter is a dynamic filter, then R_translator **9** determines whether the record to be unloaded passes or fails the current filter. If the record does not pass the current filter (step **406**) then the record is deleted on the remote database (step **407**). If the record passes the current filter, then R_translator **9**, in the same manner as step **404**, adds, deletes, or modifies the record according to results of synchronization obtained during CAAR analysis (step **408**).

By deleting records which fail the filter expression in steps **403** and **407**, R_translator **9** uses the filter to limit the size of the remote database. If the remote database is located on a handheld computer, R_translator manages the memory of the handheld device by limiting the size of the database stored on the handheld computer.

Following unloading of the remote database records, Control Module **2** instructs the L_translator to unload the local database records from the workspace (FIG. **4**, step **111**). FIG. **15** is pseudocode for the steps taken by L_translator **5** to unload the local database records in the workspace. The steps **450–452**, **454–455**, **458–460** are respectively the same as steps **400–402**, **404–405**, and **408–410** in FIG. **14**, described in detail above. Unlike R_translator **9**, L_translator **5** does not delete records falling outside of the filter. Therefore, in step **453**, if the filter is a static filter and the record does not fit the filter, then L_translator **5** modifies (i.e updates) the record if that is the synchronization result obtained during the CAAR analysis. However, synchronizer **15** does not add or delete a record, if that is synchronization result obtained during the CAAR analysis. In step **456**, similarly, if the record is within the loading filter, then L translator **5** modifies the record if that is the synchronization result obtained during the CAAR analysis (step **457**). In this manner, L_translator **5** propagates to the records of the local database changes to those remote database records which do not fit the current filter expression. It should be noted that such remote database records are deleted from the remote database (step **407**, FIG. **14**) since those remote database records fail the current filter. Also, during unloading, the unloader module of the L_translator uses the remote database to local database map to map the records in the workspace back into the format of local database records.

Additionally, it should be noted that where the local database manager application filters the local database, the local database manager application only provides those records which pass the loading filter. Therefore, the effect of step **457** is to update those records which passed the previous value of the filter expression but fail the current value of the filter expression.

It should be noted that in other embodiments, translators for the local and remote databases may use the filter expression during unloading in different manner than in the above described embodiments. For example, the remote translator may be configured in a similar manner as the local translator described above. Or, either the remote or local translator may only update records within the filter and leave completely unaffected records outside the filter. A translator may also add new records from one database to another, if those records fall outside of the current filter but are within the loading filter.

Control Module **2** next instructs synchronizer **15** to create a new history file (step **112**). The process of creating a history file is described in detail in the '490, '926 and '645 applications. Briefly, for each CIG, synchronizer **15** during the CAAR process determines which one of the records in the CIG should be saved as the history file record. Based on these results, synchronizer **15** creates a history file. Synchronizer **15** also stores with each history file record the PASSED_FILTER/FAILED_FILTER flag based on whether the record passes or fails the current filter. Synchronizer also stores the value which determined the value of the dynamic filter for the current synchronization (e.g. the date of the current synchronization in the case of a dynamic date range filter).

At this point Synchronization is complete.

Other embodiments are within the following claims.

For example, if one of the databases has the capability to provide database generated information or data which can be used to determine, for example, whether a record has been changed, added, or deleted since a previous synchronization, the synchronization program uses that information to determine whether a record has been changed, added, or deleted. Of course, that database generated information is less than the whole record of the database. For example, that information may be a date and time stamp, or a flag, set when the record was last modified or when the record was added, whichever is later. We will briefly describe an embodiment of such a synchronization program, which is described in detail in the commonly assigned copending U.S. patent application, incorporated by reference in its entirety, entitled "Synchronization of Databases," filed on Nov. 5, 1997, Ser. No. 08/964,751 (hereinafter, the "'751 application").

There are generally two types of such databases: "medium synchronization" and "fast synchronization" databases. A "fast synchronization" database is a database which provides information regarding changes, deletions, and additions to its records from one synchronization to the next. A fast synchronization database also assigns to each record of the database a unique identification code (i.e. a unique ID) which uniquely identifies that record. Unique IDs are required to accurately identify records over a period of time. A fast synchronization database also provides a mechanism for keeping track of which records are added, changed, or deleted from synchronization to synchronization, including a list of deleted records.

A "medium synchronization" database typically has more limited capabilities than a fast synchronization database for keeping track of addition, deletions, or changes. In short, a medium synchronization database does not keep track of deletions. Such a database however still has the capability to provide information regarding what records were added or modified since a previous synchronization. A medium synchronization database also provides unique IDs.

If the information provided by a database indicates that a record has not been changed or added since a previous synchronization, the synchronization program need not load that record and can use the history file to reconstruct the relevant contents of that record for synchronizing the two databases. The history file contains a copy of the relevant content of that record as the record was at the time of (e.g. at the end of) the previous synchronization. Using the history file to reconstruct the record instead of loading the record can result in significant saving of time—where for example the data transfer link between the two computers is slow—since typically a majority of records in databases are unchanged records. The synchronization program thereby increases the efficiency of performing synchronization between two databases.

The synchronization program does not synchronize a record of the fast or medium synchronization database that fails the filter expression with the records of the other database. Therefore, the synchronization program does not reconstruct those unchanged records which fail the filter expression. However, the synchronization reconstructs unchanged records of the fast or medium synchronization database which failed the filter during the last synchronization but pass the current filter. In that case, since the records previously failed the filter, the records would not be in the other database. After reconstructing these unchanged records, the synchronization program treats these records as if these records were newly added to the fast or medium synchronization database and therefore adds these record to the other database.

As is apparent, when synchronizing a fast or medium synchronization database, the synchronization program may use the history file to reconstruct unchanged records, whether those unchanged records fail or pass the current filter. Therefore, the synchronization program at the end of

each synchronization ensures that the records of the history file are synchronized with the records of the fast or medium synchronization database. In essence, the synchronization program ensures that the history file contains records which represent all of the records of the fast or medium synchronization at the end of the current synchronization, whether the records pass or fail the current filter. Since, as described above, some records of the fast or medium synchronization database are not present in the other database, the history file contains some records that are present only in the fast or medium synchronization database but not in the other database.

Where the database manager application of a fast or medium synchronization database filters the records of the database, the synchronization program does not receive those records of the database which fail the filter. Therefore, if the filter changes such that some of the unchanged records which were previously outside of the filter are now within the filter, the synchronization program can not rely on the history file. In that case, the synchronization program loads all records of the database and proceeds to synchronize without using the history file.

In some embodiments, both computers on which the two databases run are capable of running programs other than a database, as in the case of, for example, general purpose computers such as desktop and notebook computers, or handheld computers having sufficient memory and processing power. In such a case, the synchronization program may be distributed between the two computers so as to, for example, increase the efficiency of using of a slow data transfer link between the two machines. We will briefly describe an embodiment of such a distributed synchronization program, which is described in detail in the commonly assigned copending U.S. patent application, incorporated herein in its entirety by reference, entitled "Distributed Synchronization of Databases", filed on Sep. 11, 1997, Ser. No. 08/927,922 (hereinafter, the "'922 application").

Briefly, referring to FIG. **16**, at remote computer **22**, remote segment **26** of the synchronization program loads records of remote database **13**. Remote segment **26** then determines which records of the remote database have been changed/added, deleted or left unchanged since a previous synchronization. If the remote database assigns unique identification codes (i.e. unique ID) to its records, remote segment **26** can further differentiate between records than have been added and those than have been changed since the previous synchronization. Remote segment **26** uses a remote history file **30** which stores data representing or reflecting the records of the database at the completion of the previous synchronization. This data may be a copy of remote database **13**. It may also be hash numbers for each of the records of the remote database. If the remote database assigns unique IDs, the remote history file may contain those unique IDs together with the hash numbers of the records corresponding to the stored unique IDs.

Remote segment **26** sends those records of the remote database that have been changed or added to the host segment or the host computer. However, the remote segment does not send the unchanged or deleted records to the host computer. Instead, the remote segment sends a flag indicating the status of the record (e.g. unchanged or changed) and some data or information that uniquely identifies the record to the host segment. This data or information may be a hash number of all or selected fields in the record at the completion of the last synchronization. It may also be the unique ID assigned to the record by the remote database, if the database assigns one to its records.

Host segment **28** uses the received information or data that uniquely identifies the unchanged record to access a record in host history file **19** that corresponds to the received information or data. This record contains a copy of the data of the remote database record that the remote segment found to have been unchanged. Host segment **19** then uses this record to synchronize the databases in the same manner as described above. After synchronization, the remote and host history files are updated. Also, the records are unloaded to the remote and local database based on the filter expression, in the same manner as described above. Since the unchanged records which typically constitute most of the records of a database are not transferred to the host computer, a data transfer link, specially a slow data transfer link, is used with increased efficiency.

In such a distributed synchronization program, the remote and host segments would apply the filter expression to the records of the databases. In the case of the host segment, the process would be similar to that for the above described embodiments. In the case of the remote database, along with information identifying the record or the record's field values, remote segment **26** sends the host segment information indicating that the record passed or failed the filter. The synchronization process then proceeds as for the previously described embodiments. During unloading, the host segment sends the remote segment information with respect whether the records passed or failed the filter expression, along with the result of CAAR. Remote segment **26** then uses this information and the filter expressions, in the same manner as the above described translators when unloading records to the remote database.

In some embodiments, two or more databases on one computer may be synchronized with one database on the other computer. For example, it may be that the remote database application supports only one name and address database while the local database application supports two name and address databases. To synchronize the multiple local databases with the single remote database, one of the local databases is designated as the main local database for synchronization with the remote database. During synchronization, as synchronization program **100** adds records from the local databases to the remote database, synchronization program tags the added records with codes (i.e. origin tags) identifying the source of the database record, e.g. the first local database, the second local database, etc. Synchronization program **100** uses these tags during future synchronizations to ensure that the tagged records are synchronized with the correct database, i.e. the database from which they originated. Additionally, during synchronization, synchronization adds new records from the remote database only to the local database which was designated as the main database. This method of synchronization is described in detail the '490, '926 and '645 applications.

To enable filtering the records during synchronization, synchronization program **100** uses the origin tags of the records to ensure that the correct filter is applied to the correct records. For example, consider the case where two local databases are synchronized with a single remote database. Each of the local databases may have a unique filter. Or one database may have a filter and the other may not. When synchronization program **100** is synchronizing the local databases with the remote database, synchronization program **100** uses the origin tags of the remote database records to determine which filter should be applied to each record. If the origin tag of a remote database record indicates that it originated from the first database, then the filter

expression for that database, if any, is used. Similarly, if the origin tag of a remote database record indicates that it originated from the second database, then the filter expression for that database, if any, is used. Additionally, if a remote database record is new (i.e. newly added since the previous synchronization), the filter expression for the local database that was designated as the main local database is used. It should be noted that while this method of synchronization was described for synchronizing a single remote database with multiple local database, same method may be used for synchronizing multiple remote databases with a single local database, or multiple remote databases with multiple local databases.

It should be noted that the synchronization process in the above embodiments was described primarily in reference to using a filter during synchronization. If a user chooses not to use a filter, the synchronization proceeds generally in the manner described the '751, '922, '490, '926 and '645 applications.

What is claimed is:

1. A computer implemented method of synchronizing at least a first and a second personal information management database wherein the records of at least the first database include a text field, the method comprising:

using a filter to select a plurality of records of the first database, the filter comprising one or more user definable conditions or criteria that a record must match or fit to be selected, wherein the conditions or criteria include a user definable text criterion, and selecting the plurality records of the first database includes comparing the text field with the text criterion; and

synchronizing the selected records of the first database with records of the second database,

the synchronizing comprising adding, modifying, or deleting records,

whereby synchronization is performed for a subset of the records of the databases.

2. The method of claim **1** wherein records representative of the records of one of the first and second databases during a prior synchronization are stored in a history file, and wherein synchronizing the at least one of the selected records of the first database with a record of the second database further includes using the history file.

3. The method of claim **1** further comprising selecting records of the first database based on a selected criterion, wherein synchronizing the at least one of the selected records of the first database with a record of the second database further includes synchronizing the at least one of the selected records of the first database with at least one of the selected records the second database.

4. The method of claim **1** wherein records of the first database include a number field and the conditions criteria include a number criterion, and selecting the plurality records of the first database includes comparing the number field with the number criterion.

5. The method of claim **1** wherein records of the first database include a date field and the conditions or criteria include a date criterion, and selecting the plurality records of the first database includes comparing the date field with the date criterion.

6. The method of claim **1** wherein records of the first database include a boolean field and the conditions or criteria include a boolean criterion, and selecting the plurality records of the first database includes comparing the boolean field with the boolean criterion.

7. The method of claim **1** wherein records of the first database include a time field and the conditions or criteria include a time criterion, and selecting the plurality records of the first database includes comparing the time field with the time criterion.

8. The method of claim **1** wherein a selected plurality of the fields of the records of the first database are mapped onto a selected plurality of corresponding fields of the records of the second database and using a filter to select a plurality of records of the first database includes determining whether contents of a field of the records of the first database fit the one or more conditions or criteria wherein the field of the records of the first database is not mapped onto a corresponding field of the records of second database.

9. The method of claim **1** wherein the first database is located on a first computer and the second database located on a second computer, the method further comprising:

determining, at the first computer, whether a record of the first database has been changed or added since a previous synchronization, using a first history file located on the first computer comprising records representative of records of the first database at the completion of the previous synchronization;

if the record of the first database has not been changed or added since the previous synchronization, sending from the first computer to the second computer information which the second computer uses to select the record of the first database to be unchanged.

10. The method of claim **9** wherein selecting the plurality of records of the first database is performed at the first computer.

11. The method of claim **9** wherein selecting the plurality of records of the first database is performed at the second computer.

12. The method of claim **1** further comprising:

determining whether the records of the first database have been changed or added since the previous synchronization, based on data reflecting whether the records of the first database have been added or changed since a previous synchronization;

if one of the records of the first database has not been changed or added since the previous synchronization, performing a synchronization with records of the second database using a record representative of the one record at the time of a previous synchronization, the representative record being stored in a history file containing records reflecting the contents of records of the first database at the time of a previous synchronization.

13. The method of claim **12** wherein the history file contains at least one record representative of at least one record of the first database failing to fit the one or more conditions or criteria at the time of the previous synchronization and failing to be synchronized with the records of the second database at the time of the previous synchronization.

14. The method of claim **1** further comprising:

deleting a second plurality of the records of the first database failing to fit the conditions or criteria.

15. The method of claim **14** fixer comprising:

updating a plurality of records of the second database failing to fit the current value of the conditions or criteria.

16. The method of claim **1** wherein the conditions or criteria has a current value during a current synchronization being different from a previous value during a previous synchronization, further comprising:

updating a plurality of records of the second database, based on results of the synchronization, wherein the

plurality of records of the second database fit the previous value of the conditions or criteria but fail to fit the current value of the conditions or criteria.

17. The method of claim **1** further comprising synchronizing a third database with one of the first and second databases.

18. The method of claim **17** wherein synchronizing the third database with one of the first and second databases includes:

selecting a plurality of records of a third database fitting second conditions or criteria; and

synchronizing at least one of the identified records of the third database with a second record of the one of the first and second databases.

19. The method of claim **18** wherein the first-mentioned conditions or criteria and the second conditions or criteria are the same.

20. The method of claim **18** wherein the second record of the one of the first and second databases includes a code identifying the second record as having originated from the third database.

21. The method of claim **1** further comprising displaying a record selection criteria input region on a computer display for a user to input a record selection criteria that specifies the conditions or criteria.

22. The method of claim **21** further comprising:

transmitting the record selection criteria to a database manager, wherein the database manager manages the first database, and using the record selection criteria to select records of the first database fitting the conditions or criteria.

23. The method of claim **1**, wherein the first and second databases are different copies of the personal information management database of the same individual.

24. The method of claim **23**, wherein the personal information management database includes a schedule database.

25. The method of claim **1**, wherein the first and second databases are located on different computers and at least one of the computers is a handheld computer.

26. The method of claim **25**, wherein the handheld computer has less storage capacity than the other computer, wherein fewer records are stored in the database on the handheld computer than in the database on the other computer, and wherein the filter is used to limit the number of records added to the database on the handheld computer during a synchronization.

27. A computer program, resident on a computer readable medium, for synchronizing at least a first and a second personal information management database wherein the records of at least the first database include a text field, comprising instructions for:

using a filter to select a plurality of records of the first database,

the filter comprising one or more user definable conditions or criteria that a record must match or fit to be selected, wherein the conditions or criteria include a user definable text criterion, and selecting the plurality records of the first database includes comparing the text field with the text criterion; and

synchronizing the selected records of the first database with records of the second database,

the synchronizing comprising adding, modifying, or deleting records,

whereby synchronization is performed for a subset of the records of the databases.

28. The computer program of claim **27** wherein records representative of the records of the first and second databases during a prior synchronization are stored in a history file, and wherein synchronizing the at least one of the selected records of the first database with a record of the second database further includes using the history file.

29. The computer program of claim **27** further comprising instructions for selecting records of the first database based on a selected criterion, wherein synchronizing the at least one of the selected records of the first database with a record of the second database further includes synchronizing the at least one of the selected records of the first database with at least one of the selected records the second database.

30. The computer program of claim **27** wherein records of the first database include a number field and the conditions or criteria include a number criterion, and selecting the plurality records of the first database includes comparing the number field with the number criterion.

31. The computer program of claim **27** wherein records of the first database include a date field and the conditions or criteria include a date criterion, and selecting the plurality records of the first database includes comparing the date field with the date criterion.

32. The computer program of claim **27** wherein records of the first database include a boolean field and the conditions or criteria a include a boolean criterion, and selecting the plurality records of the first database includes comparing the boolean field with the boolean criterion.

33. The computer program of claim **27** wherein records of the first database include a time field and the conditions or criteria include a time criterion, and selecting the plurality records of the first database includes comparing the time field with the time criterion.

34. The computer program of claim **27** wherein a selected plurality of the fields of the records of the first database are mapped onto a selected plurality of corresponding fields of the records of the second database and using a filter to select a plurality of records of the first database includes determining whether contents of a field of the records of the first database fit one or more conditions or criteria, wherein the field of the records of the first database is not mapped onto a corresponding field of the records of second database.

35. The computer program of claim **27** wherein the first database is located on a first computer and the second database located on a second computer, the computer program further comprising instructions for:

determining, at the first computer, whether a record of the first database has been changed or added since a previous synchronization, using a first history file located on the first computer comprising records representative of records of the first database at the completion of the previous synchronization;

if the record of the first database has not been changed or added since the previous synchronization, sending from the first computer to the second computer information which the second computer uses to select the record of the first database to be unchanged.

36. The computer program of claim **35** wherein selecting the plurality of records of the first database is performed at the first computer.

37. The computer program of claim **35** wherein selecting the plurality of records of the first database is performed at the second computer.

38. The computer program of claim **27** further comprising instructions for:

determining whether the records of the first database have been changed or added since the previous synchronization, based on data reflecting whether the records of the first database have been added or changed since a previous synchronization;

if one of the records of the first database has not been changed or added since the previous synchronization, performing a synchronization with records of the second database using a record representative of the one record at the time of a previous synchronization, the representative record being stored in a history file containing records reflecting the contents of records of the databases at the time of a previous synchronization.

39. The computer program of claim **38** wherein the history file contains at least one record representative of at least one record of the first database failing to fit the one or more conditions or criteria at the time of the previous synchronization and failing to be synchronized with the records of the second database at the time of the previous synchronization.

40. The computer program of claim **27** further comprising instructions for:

deleting a second plurality of the records of the first database failing to fit the conditions or criteria.

41. The computer program of claim **40** further comprising instructions for:

updating a plurality of records of the second database failing to fit the current value of the conditions or criteria.

42. The computer program of claim **27** wherein the conditions or criteria has a current value during a current synchronization being different from a previous value during a previous synchronization, further comprising instructions for:

updating a plurality of records of the second database, based on results of the synchronization, wherein the plurality of records of the second database fit the previous value of the conditions or criteria but fail to fit the current value of the conditions or criteria.

43. The computer program of claim **27** further comprising instructions for synchronizing a third database with one of the first and second databases.

44. The computer program of claim **43** wherein synchronizing the third database with one of the first and second databases includes instructions for:

selecting a plurality of records of a third database fitting second conditions or criteria; and

synchronizing at least one of the identified records of the third database with a second record of the one of the first and second databases.

45. The computer program of claim **44** wherein the first-mentioned conditions or criteria and the second conditions or criteria are the same.

46. The computer program of claim **44** wherein the second record of the second database includes a code identifying the second record of the second database as having originated from the third database.

47. The computer program of claim **27** further comprising instructions for displaying a record selection criteria input region on a computer display for a user to input a record selection criteria that specifies the conditions or criteria.

48. The computer program of claim **47** further comprising instructions for:

transmitting the record selection criteria to a database manager, wherein the database manager manages the first database, and using the record selection criteria to select records of the first database fitting the conditions or criteria.

49. The computer program of claim **27**, wherein the first and second databases are different copies of the personal information management database of the same individual.

50. The computer program of claim **49**, wherein the personal information management database includes a schedule database.

51. The computer program of claim **27**, wherein the first and second databases are located on different computers and at least one of the computers is a handheld computer.—

52. The method of claim **51**, wherein the handheld computer has less storage capacity than the other computer, wherein fewer records are stored in the database on the handheld computer than in the database on the other computer, and wherein the filter is used to limit the number of records added to the database on the handleld computer during a synchronization.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 6,212,529 B1
DATED : April 3, 2001
INVENTOR(S) : David J. Boothby, David W. Morgan and John R. Marien

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Title page,
Item [75], Inventors, "**John R. Marien**, Nashua" should read -- **John R. Marien**, Hollis --.
Item [57], **ABSTRACT**,
Line 6, after "criteria", insert -- input region may be --.
Item [56], **References Cited**, U.S. PATENT DOCUMENTS, "U.S. application no. 08/749926" should be -- U.S. application no. 08/749,926 --.

Column 2,
Line 33, after "records", insert -- of --.

Column 8,
Line 8, after "always", delete "the".

Column 10,
Line 8, after "similar", insert -- to --.

Column 13,
Line 19, "STARTS WITH" should be -- STARTS_WITH --.

Column 18,
Line 30, after "record", delete "is".

Column 20,
Line 61, "record" should be -- records --.

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 6,212,529 B1
DATED : April 3, 2001
INVENTOR(S) : David J. Boothby, David W. Morgan and John R. Marien

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 23,
Line 18, after "described", insert -- in --.
Line 30, after "plurality", insert -- of --.
Line 50, after "records", insert -- of --.
Line 52, after "conditions", insert -- or --.

Column 24,
Line 58, "fixer" should be -- further --.

Column 26,
Line 11, after "records", insert -- of --.

Signed and Sealed this

Tenth Day of December, 2002

JAMES E. ROGAN
Director of the United States Patent and Trademark Office

US006212529C1

(12) EX PARTE REEXAMINATION CERTIFICATE (5217th)

United States Patent
Boothby et al.

(10) **Number: US 6,212,529 C1**
(45) **Certificate Issued: *Oct. 11, 2005**

(54) **SYNCHRONIZATION OF DATABASES USING FILTERS**

(75) Inventors: **David J. Boothby**, Nashua, NH (US); **David W. Morgan**, Derry, NH (US); **John R. Marien**, Hollis, NH (US)

(73) Assignee: **Pumatech, Inc.**, San Jose, CA (US)

(*) Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

Reexamination Request:
No. 90/006,569, Mar. 10, 2003

Reexamination Certificate for:

Patent No.:	**6,212,529**
Issued:	**Apr. 3, 2001**
Appl. No.:	**09/036,400**
Filed:	**Mar. 5, 1998**

Certificate of Correction issued Dec. 10, 2002.

Related U.S. Application Data

(63) Continuation-in-part of application No. 08/748,645, filed on Nov. 13, 1996, now Pat. No. 6,141,664.

(51) **Int. Cl.**[7] **G06F 12/00**; G06F 17/30
(52) **U.S. Cl.** **707/201**; 709/216
(58) **Field of Search** 707/6, 8, 10, 104.1, 707/201; 345/762

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,162,610	A		7/1979	Levine
4,807,154	A		2/1989	Scully et al.
4,807,155	A		2/1989	Cree et al.
4,817,018	A		3/1989	Cree et al.
4,819,191	A		4/1989	Scully et al.
4,831,552	A		5/1989	Scully et al.
4,939,689	A		7/1990	Davis et al.
5,124,912	A		6/1992	Hotaling et al.
5,134,564	A		7/1992	Dunn et al.
5,197,000	A		3/1993	Vincent
5,201,010	A		4/1993	Deaton et al.
5,220,540	A		6/1993	Nishida et al.
5,276,876	A		1/1994	Coleman et al.
5,323,314	A		6/1994	Baber et al.
5,327,555	A		7/1994	Anderson 707/201
5,392,390	A	*	2/1995	Crozier 345/762
5,412,801	A		5/1995	De Remer et al.
5,421,012	A		5/1995	Khoyi et al.
5,455,945	A		10/1995	VanderDrift
5,530,853	A		6/1996	Schell et al.
5,530,939	A		6/1996	Mansfield, Jr et al.
5,557,518	A		9/1996	Rosen
5,581,753	A		12/1996	Terry et al.
5,581,754	A		12/1996	Terry et al.
5,596,574	A		1/1997	Perlman et al.

(Continued)

OTHER PUBLICATIONS

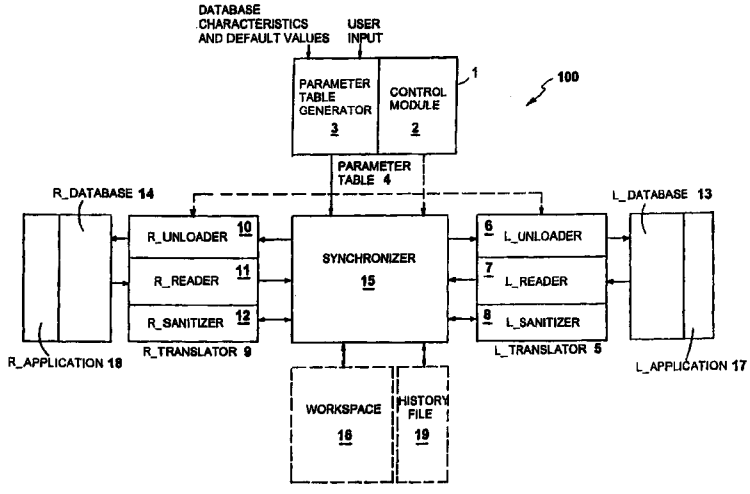
Extended Systems' Final Invalidity Contentions (Oct. 10, 2003).
Defendant and Cross–Complainant Extended Systems, Inc.'s Identification of Prior Art Publications Pursuant to Patent L.R. 3–3(a) (Oct. 17, 2003).
Defendant and Cross–Complainant Extended Systems, Inc.'s Amended Identification of Prior Art Publications Pursuant to Patent L.R. 3–3(a) (Oct. 31, 2003).
Expert Report of John P. J. Kelly, Ph.D. (Oct. 24, 2003).

(Continued)

Primary Examiner—Safet Metjahic

(57) **ABSTRACT**

A computer program is provided for synchronizing at least a first and a second database. A plurality of records of the first database fitting a selected criterion are identified. At least one of the identified records of the first database is then synchronized with a record of the second database. On a computer display, a record selection criteria input region may be displayed for a user to input the selected criterion.



U.S. PATENT DOCUMENTS

5,608,865	A	*	3/1997	Midgely et al.	395/180
5,615,109	A		3/1997	Eder	
5,623,540	A		4/1997	Morrison et al.	
5,649,182	A		7/1997	Reitz	
5,659,741	A	*	8/1997	Eberhardt	707/104.1
5,666,530	A		9/1997	Clark et al.	707/201
5,671,407	A		9/1997	Demers et al.	
5,689,706	A		11/1997	Rao et al.	
5,704,029	A		12/1997	Wright, Jr.	
5,706,452	A		1/1998	Ivanov	
5,706,509	A		1/1998	Man Hak Tso	
5,727,202	A		3/1998	Kucala	707/10
5,729,735	A	*	3/1998	Meyering	707/10
5,737,539	A		4/1998	Edelson et al.	
5,758,337	A	*	5/1998	Hammond	707/6
5,781,908	A		7/1998	Williams et al.	
5,809,494	A		9/1998	Nguyen	
5,813,009	A		9/1998	Johnson et al.	
5,813,013	A		9/1998	Shakib et al.	
5,819,272	A		10/1998	Benson	
5,819,274	A		10/1998	Jackson, Jr.	
5,832,218	A		11/1998	Gibbs et al.	
5,832,489	A		11/1998	Kucala	
5,838,923	A		11/1998	Lee et al.	
5,857,201	A		1/1999	Wright, Jr. et al.	707/104.1
5,875,242	A		2/1999	Glaser et al.	
5,877,760	A		3/1999	Onda et al.	
5,892,909	A		4/1999	Grasso et al.	
5,924,094	A		7/1999	Sutter	
5,926,816	A	*	7/1999	Bauer et al.	707/8
5,928,329	A		7/1999	Clark et al.	
5,943,676	A		8/1999	Boothby	
5,956,508	A		9/1999	Johnson et al.	
5,966,714	A	*	10/1999	Huang et al.	707/201
5,974,238	A		10/1999	Chase, Jr.	
5,995,980	A		11/1999	Olson et al.	
5,999,947	A		12/1999	Zollinger et al.	
6,018,303	A		1/2000	Sadeh	
6,044,381	A		3/2000	Boothby	
6,081,806	A		6/2000	Chang et al.	
6,098,078	A		8/2000	Gehani et al.	
6,125,369	A		9/2000	Wu et al.	
6,141,664	A		10/2000	Boothby	
6,212,221	B1		4/2001	Wakayama et al.	
6,233,452	B1		5/2001	Nishino	
6,272,074	B1		8/2001	Winner	
6,324,542	B1		11/2001	Wright, Jr. et al.	
6,330,568	B1		12/2001	Boothby	
6,405,218	B1		6/2002	Boothby	

OTHER PUBLICATIONS

IntelliLink for Windows User's Guide, Version 3.0, IntelliLink Corporation (1993).

Database Subsetting Tool: Introduction to DST and DST Designer's Guide, Syware, Inc. (1993).

Sarin, "Robust Application Design in Highly Available Distributed Databases," Proc. 5th Symp. Reliability in Distributed Software and Database Systems, pp. 87–94 (Jan. 13–15, 1986, Los Angeles).

Distributed Management of Replicated Data: Final Report, Computer Corporation of America (Oct. 9, 1984).

Sarin et al., "Overview of SHARD: A System for Highly Available Replicated Data", Computer Corporation of America (Apr. 8, 1988).

SRI Int'l, Network Reconstitution Protocol, RADC–TR–87–38, Final Technical Report (Jun. 1987).

Danberg, "A Database Subsetting Tool" (patent application) (Apr. 12, 1993).

Lamb et al., "The Objectstore Database System," Communications of the ACM, vol. 34, No. 10, pp. 50–63 (Oct. 1991).

TT Interchange, Time Technology, AVG Sales & Marketing Ltd. (1995).

Goldberg et al., "Using Collaborative Filtering to Weave an Information Tapestry," Communications of the ACM, vol. 35, No. 12, pp. 61–70 (Dec. 1992).

Now Up–to–Date Version 2.0 User's Guide, Now Software, Inc. (1992).

An Introduction to DataPropagator Relational Version 1, IBM Corporation (1993).

Data Propagator Relational Guide Release 1, IBM Corporation (May 1994).

DataPropagator Relational Guide Release 2, IBM Corporation (Dec. 1994).

DataPropagator NonRelational MVS/ESA Version 2 Utilities Guide, IBM Corporation (Jul. 1994).

DPROPR Planning and Design Guide, IBM Corporation (Nov. 1996).

DataPropagator Relational Capture and Apply/400 Version 3, IBM Corporation (Jun. 1996).

DataPropagator Relational Capture and Apply for OS/400 Version 3, IBM Corporation (Nov. 1996).

Newton Connection Utilities User's Manual for the Macintosh Operating System, Apple Computer, Inc. (1996).

Newton Connection Utilities User's Manual for Windows, Apple Computer, Inc.

Newton Connection Utilities User's Manual for Macintosh, Apple Computer, Inc.

Newton Backup Utility User's Guide for the Windows Operating System, Apple Computer, Inc. (1995).

Newton Backup Utility User's Guide for the Macintosh Operating System, Apple Computer, Inc. (1995).

Newton Utilities User Manual, Apple Computer, Inc. (1995).

FileMaker Pro Server Administrator's Guide, Claris Corporation (1994).

Connectivity Pack User's Guide for the HP 200LX and the HP 100LX, Hewlett Packard.

Lotus cc:Mail Release 2, Lotus Development Corporation (1991–1993).

User's Guide Lotus Organizer Release 1.0, Lotus Development Corporation (1992).

FileMaker Pro User's Guide, Claris Corporation (1990, 1992).

Poesio et al., "Metric Constraints for Maintaining Appointments: Dates and Repeated Activities".

Slater, "Newton's Legacy; 3COM and Microsoft Battle for Market Share; Apple Newton, 3Com Palm III, Microsoft Palm–size PC peronal digital assistants; Product Information", Information Access Company (1998).

Negrino, "ACT 2.5.1, ACT for Newton 1.0", UMI, Inc. (1996).

Zilber, "Toy story; personal digital assistants; Product Information", Information Access Company (1996).

Wingfield, "Desktop to Newton connectivity", UMI, Inc. (1996).

"Now Software Announces Updated Synchronization Software for Newton 2.0 Devices; Now Synchronize Simultaneously Updates MessagePad, Now Up–to–Date & Contact", Business Wire, Inc. (1995).

"Claris Ships FileMaker Pro 3.0 for Macintosh and Windows", Business Wire, Inc. (1995).
Alsorp, "Distributed Thinking; Realizing the gravity of its PDA problems, Apple has drawn me back to Newton", InfoWorld Media Group (1995).
Rubin, "Now Software stays in sync; Now Synchronize file synchronization software for Macs and Newton PDAs; Software Review; EvaluationBrief Article", Information Access Company (1995).
"Now Calendar/Scheduler/Contact Mgr for Mac Update", Post–Newsweek Business Information Inc. (1995).
Staten, "csInStep middleware lets Newton talk to PIMs; Concierge Software LC's csInStep; Brief Article; Product Announcement; Brief Article", Information Access Company (1995).
Baum, "Designing Moble applications; A new approach needed for on–the–road systems", InfoWorld Media Group (1994).
Parkinson, "Remote users get in sync with office files; News Analysis", Information Access Company (1994).
Informix Guide to SQL Tutorial Version 7.1, Dec. 1994.
Oracle7 Distributed Database Technology and Symmetric Replication, Oracle White Paper, Apr. 1995.
Oracle7 Server Distributed Systems, vol. II: Replicated Data, Release 7.3, Feb. 1996.
Oracle7™ Server SQL Manual Release 7.3, Feb. 1996.
Extended Systems' Preliminary Invalidity Contentions.
Extended Systems' First Supplemental Preliminary Invalidity Contentions.
Extended Systems, Inc.'s Preliminary Claim Contructions and Preliminary Identification of Extrinsic Evidence.
Patent Local Rule 4–2 Preliminary Claim Constructions and Extrinsic Evidence.
Joint Claim Construction and Prehearing Statement.
Extended Systems' Second Supplemental Preliminary Invalidity Contentions [Re: Reexamination Requests for the '390, '664, and '529 Patents].
Pumatech, Inc.'s Opening Claim Construction Brief; Declaration of Marc David Peters in Support of Pumatech, Inc.'s Opening Claim Construction Brief.
Extended Systems, Inc.'s Responsive Claim Construction Brief; Declaration of Jordan Trent Jones in Support of Extended Systems, Inc.'s Responsive Claim Construction Brief.
Supplemental Decalaration of Marc David Peters in Support of Pumatech, Inc.'s Reply Claim Construction Brief.
Pumatech's Revised [Proposed] Claim Construction Order.
Pumatech, Inc.'s Reply Claim Construction Brief.
Statement of Recent Decision.
Pumatech's [Proposed] Claim Construction Order.
Synchrologic's Preliminary Invalidity Contentions.
Adly, "HARP: A Hierarchical Asynchronous Replication Protocol for Massively Replicated Systems," Computer Laboratory, Cambridge University, United Kingdom (undated).
Adly et al., "A Hierarchical Asynchronous Replication Protocol for Large Scale Systems," Computer Laboratory, Cambridge University, United Kingdom, Computer Science Department, Alexandria University, Egypt (undated).
Alexander, "Designed, sold, delivered, serviced," Computerworld Client/Server Journal, p. 43 (Oct. 1, 1995).
"All I need is a miracle; computer–aided educational packages; Small Wonders," Coastal Associates Publishing L.P. (Mar. 1992).
Alonso et al., "Database System Issues in Nomadic Computing," Matsushita Information Technology Laboratory, New Jersey (undated).
Badrinath et al., "Impact of Mobility on Distributed Computations," Operating Systems Review (Apr. 1, 1993).
Barbara et al., "Sleeper and Workaholics: Caching Strategies in Mobile Environments (Extended Version)" (Aug. 29, 1994).
Brandel, "New offerings fuel revival of PIM," Computerworld, p. 39 (Sep. 12, 1994).
Brodersen, "InfoPad—An Experiment in System Level Design and Integration," (Mar. 1, 1997).
Demers et al., "The Bayou Architecture: Support for Data Sharing Among Mobile Users," Computer Science Laboratory, Xerox Palo Alto Research Center, California (undated).
DeVoe et al., "SOFTWARE: Day–Timer Organizer 2.0 based on format of paper–based PIM," InfoWorld, vol. 17 (Aug. 21, 1995).
Froese, "File System Support for Weakly Connected Operation," pp. 229–238 (undated).
Greenberg et al., "Real Time Groupware as a Distributed System: Concurrency Control and its Effect on the Interface," Procs. of the ACM CSCW Conf. On Computer Supported Cooperative Work, Oct. 22–26, North Carolina, ACM Press (Jan. 1, 1994).
Guy, "Ficus: A Very Large Scale Reliable Distributed File System," Technical Report CSD–910018, Computer Science Dept. UCLA (Technical Report) (Jun. 3, 1991).
Guy et al., "Implementation of the Ficus Replicated File System," appeared in Procs. Of the Summer USENIX Conf., Anaheim, CA, pp. 63–71 (Jun. 1, 1990).
Haber, "Renegade PIMS," Computerworld, p. 109 (Dec. 12, 1994).
Hammer et al., "An Approach to Resolving Semantic Heterogeneity in a Federation of Autonomous, Heterogeneous Database Systems," Computer Science Department, University of Southern California (undated).
Hammer et al., "Object Discovery and Unification in Federated Database Systems," University of Southern California (undated).
HP and IntelliLink connect HP 95LX with HP NewWave; IntelliLink for the HP NewWave; product announcement, HP Professional (Aug. 1991).
"HP announces expanded memory version of palmtop PC, introduces 1–Megabyte HP 95LX and 1–Megabyte memory cards," Business Wire, Inc. (Mar. 4, 1992).
Huston et al., "Disconnected Operation of AFS," CITI Technical Report 93–3, Center for Information Technology Integration, University of Michigan (Jun. 18, 1993).
IBM Dictionary of Computing, Tenth Edition, 1993, pp. 268, 269, 31_.
IBM Dictionary of Computing, Tenth Edition, 1993, pp. 165, 268, 349, 370, 417.
IEEE Standard Dictionary of Electrical and Electronics Terms, Fourth Edition, 1988, p. 372, 368, 509, 563.
Imielinski, "Mobile Computing—DataMan Project Perspective," Rutgers University (undated).
"IntelliLink 2.2: the software connection from desktop to palmtop; Software Review; IntelliLink 2.2; Evaluation," PC Magazine (Apr. 28, 1992).
"IntelliLink transfers palmtop, PC data; communications software from IntelliLink Inc; brief article; Product Announcement," PC Week (Nov. 18, 1991).

Jacobs et al., "A Generalized Query–by–Example Data Manipulation Language Based on Database Logic," IEEE Transactions on Software Engineering, vol. SE–9, No. 1 (Jan. 1983).
Jenkins, "Users struggle with E–mail Woes," Computerworld, p. 97 (Oct. 24, 1994).
Johnson et al., "Hierarchical Matrix Timestamps for Scalable Update Propogation," submitted to the 10th Int. Workshop on Distributed Algorithms (Jun. 25, 1996).
Joshi et al., "A Survey of Mobile Computing Technologies and Applications," (Oct. 29, 1995).
Kistler et al., "Disconnected Operation in the Coda File System," School of Computer Science, Carnegie Melon University, Pennsylvania (undated).
Krill, "NETWORKING: Tech Update," InfoWorld, vol. 18 (Feb. 12, 1996).
Kumar et al., "Log–Based Directory Resolution in the Coda File System," School of Computer Science, Carnegie Melon University, Pennsylvania (undated).
Larson et al., "A Theory of Attribute Equivalence in Databases with Application to Schema Integration," IEEE Transactions on Software Engineering, vol. 15, No. 4, Apr. 1989.
Lomet, D., Using timestamping to optimize two phase commit; Parallel and Distributed Information Systems, 1993, Proc. Of the 2nd Int. Conf., Jan. 20–22, 1993, pp. 48–55.
Mannino et al., "Matching Techniques in Global Schema Design," IEEE 1984.
Marshall, "Product Reviews: Windows contact managers," InfoWorld, vol. 18 (Mar. 25, 1996).
McGoveran, "Distributed not yet delivered," Computerworld, p. 112 (Jun. 6, 1994).
Meckler Corporation, "Palmtop–to–desktop linkage software," Database Searcher (Jun. 1992).
Microsoft Press Computer Dictionary, Second Edition, 1994, p. 164.
Microsoft Press Computer Dictionary, Second Edition, 1994, pp. 105, 217, 227, 228.
Microsoft Press Computer Dictionary, Third Edition, 1997, pp. 194, 228, 234, 449.
Milliken, "Resource Coordination Objects: A State Distribution Mechanism," (DRAFT) (Dec. 10, 1993).
Nash, "Replication falls short," Computer world, p. 65 (Nov. 21, 1994).
Noble et al., "A Research Status Report for Adaptation for Mobile Data Access," School of Computer Science, Carnegie Melon University (undated).
"PackRat PIM gets older and wiser with Release 4.0; PIM update sports enhanced interface, greater ease of use," InfoWorld (Dec. 23, 1991).
"Palmtop PCs: power by the ounce; Hardware Review; overview of six evaluations of palm–top computers; includes related articles on Editor's Choices, suitability–to–task ratings, impression by individual users; evaluation," PC Magazine (Jul. 1991).
"Pen–based PCs ready for prime time; includes related article on comparison of operating systems, list of vendors of pen–based products," PC–Computing (Nov. 1991).
Perera, "Synchronization Schizophrenia," Computerworld Client/Server Journal, p. 50 (Oct. 1, 1995).
Petersen et al., "Bayou: Replicated Database Services for World–wide Applications," Computer Science Laboratory, Xerox Palo Alto Research Center, California (undated).
"Product comparison: Atari Portfolio, Casio Executive BOSS, HP 95LX, Poqet PC, Psion series 3, Sharp Wizard," InfoWorld (Dec. 16, 1991).
"Product Comparison: Personal information managers," InfoWorld, vol. 17 (Aug. 7, 1995).
Qu et al., Technical Report entitled "Mobile File Filtering," TR–CS–97–02–Australian National University (Feb. 1, 1997).
Radosevich, "Replication mania," Computerworld Client/Server Journal, p. 53 (Oct. 1, 1995).
Ratner et al., "The Ward Model: A Replication Architecture for Mobile Environments," Department of Computer Science, University of California (undated).
Reiher et al., "Peer–to–Peer Reconciliation Based Replication for Mobile Computers," UCLA (undated).
Reiher et al., "Resolving File Conflicts in the Ficus File System," Department of Computer Science, University of California (undated).
Ricciuti, "Object database server," InfoWorld, vol. 18 (Jan. 29, 1996).
"Riding the NewWave from PC to Palmtop: IntelliLink lets NewWave users transfer files," InfoWorld (Jun. 3, 1991).
Saltor et al., "Suitability of data models as canonical models for federated databases," Universitat Politecnica de Catalunya, Spain (undated).
Salzberg, B., Timestamping After Commit, Procs. Of the Third Int. Conf. On Parallel Distributed Information Systems, Sep. 28–30, 1994, pp. 160–167.
Satyanarayanan, "Coda: A Highly Available File System for a Distributed Workstation Environment," School of Computer Science, Carnegie Mellon University (undated).
Satyanarayanan, "Fundamental Challenges in Mobile Computing," School of Computer Science, Carnegie Mellon University (undated).
Satyanarayanan, "Mobile Information Access," IEEE Personal Communications, vol. 3, No. 1 (Feb. 1996).
Sherman, "Information Technology: 'What Software Should I Use to Organize My Life'," (undated).
Sheth et al., "A Tool for Integrating Conceptual Schemas and User Views," IEEE 1988.
Schilit et al., "The ParcTab Mobile Computing System," Xerox Palo Alto Research Center, California (undated).
SPI Database Software Technologies Record Displays: Record 2, Serial No. TDB0291.0094 and Record 4, Serial No. iets0901.0073 (undated).
Staten, "PowerMerge 2.0 ships; syncs moved filed," MacWEEK, vol. 8, p. 38(1) (Jan. 3, 1994).
Tait, Doctoral Thesis entitled "A File System for Mobile Computing," (Jan. 1, 1993).
Tolly, "Enhanced Notes 4.0 gets thumbs–up," Computerworld, p. 54 (Dec. 18, 1995).
Webster's Ninth New Collegiate Dictionary, 1986, pp. 114, 436, 440, 462, 573, 597, 620, 717, 906, 963, 979, 989, 1000, 1053, 1130, 1142, 1152, 1162, 1166.
Wiederhold, Gio, Database Design, Second Edition, McGraw–Hill Book Company, 1983, p. 2.
Zaino, "Tapping the Top Values in PDAs—Personal digital assistants that sell for as little as $300 can put a PC in the palm of your hand. Get the scoop on 8 contenders," HomePC, p. 97 (Oct. 1, 1996).
Zisman et al., "Towards Inoperability in Heterogeneous Database Systems," Imperial College Research Report No. DOC 95/11 (Dec. 1, 1995).

* cited by examiner

EX PARTE REEXAMINATION CERTIFICATE ISSUED UNDER 35 U.S.C. 307

THE PATENT IS HEREBY AMENDED AS INDICATED BELOW.

Matter enclosed in heavy brackets [ ] appeared in the patent, but has been deleted and is no longer a part of the patent; matter printed in italics indicates additions made to the patent.

AS A RESULT OF REEXAMINATION, IT HAS BEEN DETERMINED THAT:

The patentability of claims **1, 2, 12, 14, 15, 21, 25–28, 38, 40, 41, 47, 51** and **52** is confirmed.

Claims **4–11, 13, 16–20, 22–24, 30–37, 39, 42–46** and **48–50** are cancelled.

Claims **3** and **29** are determined to be patentable as amended.

3. The method of claim **1** further comprising selecting records of the **[**first**]** *second* database based on a selected criterion, wherein synchronizing the at least one of the selected records of the first database with a record of the second database further includes synchronizing the at least one of the selected records of the first database with at least one of the selected records *of* the second database.

29. The computer program of claim **27** further comprising instructions for selecting records of the **[**first**]** *second* database based on a selected criterion, wherein synchronizing the at least one of the selected records of the first database with a record of the second database further includes synchronizing the at least one of the selected records of the first database with at least one of the selected records *of* the second database.

* * * * *

US006212529C1

(12) EX PARTE REEXAMINATION CERTIFICATE (5217th)

United States Patent

Boothby et al.

(10) Number: US 6,212,529 C1

(45) Certificate Issued: *Oct. 11, 2005

(54) SYNCHRONIZATION OF DATABASES USING FILTERS

(75) Inventors: **David J. Boothby**, Nashua, NH (US); **David W. Morgan**, Derry, NH (US); **John R. Marien**, Hollis, NH (US)

(73) Assignee: **Pumatech, Inc.**, San Jose, CA (US)

(*) Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

Reexamination Request:
No. 90/006,569, Mar. 10, 2003

Reexamination Certificate for:

Patent No.:	**6,212,529**
Issued:	**Apr. 3, 2001**
Appl. No.:	**09/036,400**
Filed:	**Mar. 5, 1998**

Certificate of Correction issued Dec. 10, 2002.

Related U.S. Application Data

(63) Continuation-in-part of application No. 08/748,645, filed on Nov. 13, 1996, now Pat. No. 6,141,664.

(51) **Int. Cl.**[7] **G06F 12/00**; G06F 17/30

(52) **U.S. Cl.** **707/201**; 709/216

(58) **Field of Search** 707/6, 8, 10, 104.1, 707/201; 345/762

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,162,610	A		7/1979	Levine
4,807,154	A		2/1989	Scully et al.
4,807,155	A		2/1989	Cree et al.
4,817,018	A		3/1989	Cree et al.
4,819,191	A		4/1989	Scully et al.
4,831,552	A		5/1989	Scully et al.
4,939,689	A		7/1990	Davis et al.
5,124,912	A		6/1992	Hotaling et al.
5,134,564	A		7/1992	Dunn et al.
5,197,000	A		3/1993	Vincent
5,201,010	A		4/1993	Deaton et al.
5,220,540	A		6/1993	Nishida et al.
5,276,876	A		1/1994	Coleman et al.
5,323,314	A		6/1994	Baber et al.
5,327,555	A		7/1994	Anderson 707/201
5,392,390	A	*	2/1995	Crozier 345/762
5,412,801	A		5/1995	De Remer et al.
5,421,012	A		5/1995	Khoyi et al.
5,455,945	A		10/1995	VanderDrift
5,530,853	A		6/1996	Schell et al.
5,530,939	A		6/1996	Mansfield, Jr et al.
5,557,518	A		9/1996	Rosen
5,581,753	A		12/1996	Terry et al.
5,581,754	A		12/1996	Terry et al.
5,596,574	A		1/1997	Perlman et al.

(Continued)

OTHER PUBLICATIONS

Extended Systems' Final Invalidity Contentions (Oct. 10, 2003).

Defendant and Cross–Complainant Extended Systems, Inc.'s Identification of Prior Art Publications Pursuant to Patent L.R. 3–3(a) (Oct. 17, 2003).

Defendant and Cross–Complainant Extended Systems, Inc.'s Amended Identification of Prior Art Publications Pursuant to Patent L.R. 3–3(a) (Oct. 31, 2003).

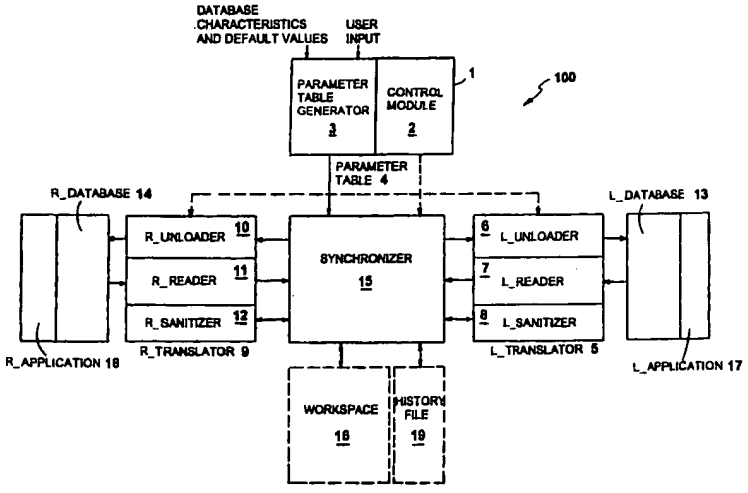
Expert Report of John P. J. Kelly, Ph.D. (Oct. 24, 2003).

(Continued)

Primary Examiner—Safet Metjahic

(57) **ABSTRACT**

A computer program is provided for synchronizing at least a first and a second database. A plurality of records of the first database fitting a selected criterion are identified. At least one of the identified records of the first database is then synchronized with a record of the second database. On a computer display, a record selection criteria input region may be displayed for a user to input the selected criterion.



U.S. PATENT DOCUMENTS

5,608,865	A	*	3/1997	Midgely et al.	395/180
5,615,109	A		3/1997	Eder	
5,623,540	A		4/1997	Morrison et al.	
5,649,182	A		7/1997	Reitz	
5,659,741	A	*	8/1997	Eberhardt	707/104.1
5,666,530	A		9/1997	Clark et al.	707/201
5,671,407	A		9/1997	Demers et al.	
5,689,706	A		11/1997	Rao et al.	
5,704,029	A		12/1997	Wright, Jr.	
5,706,452	A		1/1998	Ivanov	
5,706,509	A		1/1998	Man Hak Tso	
5,727,202	A		3/1998	Kucala	707/10
5,729,735	A	*	3/1998	Meyering	707/10
5,737,539	A		4/1998	Edelson et al.	
5,758,337	A	*	5/1998	Hammond	707/6
5,781,908	A		7/1998	Williams et al.	
5,809,494	A		9/1998	Nguyen	
5,813,009	A		9/1998	Johnson et al.	
5,813,013	A		9/1998	Shakib et al.	
5,819,272	A		10/1998	Benson	
5,819,274	A		10/1998	Jackson, Jr.	
5,832,218	A		11/1998	Gibbs et al.	
5,832,489	A		11/1998	Kucala	
5,838,923	A		11/1998	Lee et al.	
5,857,201	A		1/1999	Wright, Jr. et al.	707/104.1
5,875,242	A		2/1999	Glaser et al.	
5,877,760	A		3/1999	Onda et al.	
5,892,909	A		4/1999	Grasso et al.	
5,924,094	A		7/1999	Sutter	
5,926,816	A	*	7/1999	Bauer et al.	707/8
5,928,329	A		7/1999	Clark et al.	
5,943,676	A		8/1999	Boothby	
5,956,508	A		9/1999	Johnson et al.	
5,966,714	A	*	10/1999	Huang et al.	707/201
5,974,238	A		10/1999	Chase, Jr.	
5,995,980	A		11/1999	Olson et al.	
5,999,947	A		12/1999	Zollinger et al.	
6,018,303	A		1/2000	Sadeh	
6,044,381	A		3/2000	Boothby	
6,081,806	A		6/2000	Chang et al.	
6,098,078	A		8/2000	Gehani et al.	
6,125,369	A		9/2000	Wu et al.	
6,141,664	A		10/2000	Boothby	
6,212,221	B1		4/2001	Wakayama et al.	
6,233,452	B1		5/2001	Nishino	
6,272,074	B1		8/2001	Winner	
6,324,542	B1		11/2001	Wright, Jr. et al.	
6,330,568	B1		12/2001	Boothby	
6,405,218	B1		6/2002	Boothby	

OTHER PUBLICATIONS

IntelliLink for Windows User's Guide, Version 3.0, IntelliLink Corporation (1993).

Database Subsetting Tool: Introduction to DST and DST Designer's Guide, Syware, Inc. (1993).

Sarin, "Robust Application Design in Highly Available Distributed Databases," Proc. 5th Symp. Reliability in Distributed Software and Database Systems, pp. 87–94 (Jan. 13–15, 1986, Los Angeles).

Distributed Management of Replicated Data: Final Report, Computer Corporation of America (Oct. 9, 1984).

Sarin et al., "Overview of SHARD: A System for Highly Available Replicated Data", Computer Corporation of America (Apr. 8, 1988).

SRI Int'l, Network Reconstitution Protocol, RADC–TR–87–38, Final Technical Report (Jun. 1987).

Danberg, "A Database Subsetting Tool" (patent application) (Apr. 12, 1993).

Lamb et al., "The Objectstore Database System," Communications of the ACM, vol. 34, No. 10, pp. 50–63 (Oct. 1991).

TT Interchange, Time Technology, AVG Sales & Marketing Ltd. (1995).

Goldberg et al., "Using Collaborative Filtering to Weave an Information Tapestry," Communications of the ACM, vol. 35, No. 12, pp. 61–70 (Dec. 1992).

Now Up–to–Date Version 2.0 User's Guide, Now Software, Inc. (1992).

An Introduction to DataPropagator Relational Version 1, IBM Corporation (1993).

Data Propagator Relational Guide Release 1, IBM Corporation (May 1994).

DataPropagator Relational Guide Release 2, IBM Corporation (Dec. 1994).

DataPropagator NonRelational MVS/ESA Version 2 Utilities Guide, IBM Corporation (Jul. 1994).

DPROPR Planning and Design Guide, IBM Corporation (Nov. 1996).

DataPropagator Relational Capture and Apply/400 Version 3, IBM Corporation (Jun. 1996).

DataPropagator Relational Capture and Apply for OS/400 Version 3, IBM Corporation (Nov. 1996).

Newton Connection Utilities User's Manual for the Macintosh Operating System, Apple Computer, Inc. (1996).

Newton Connection Utilities User's Manual for Windows, Apple Computer, Inc.

Newton Connection Utilities User's Manual for Macintosh, Apple Computer, Inc.

Newton Backup Utility User's Guide for the Windows Operating System, Apple Computer, Inc. (1995).

Newton Backup Utility User's Guide for the Macintosh Operating System, Apple Computer, Inc. (1995).

Newton Utilities User Manual, Apple Computer, Inc. (1995).

FileMaker Pro Server Administrator's Guide, Claris Corporation (1994).

Connectivity Pack User's Guide for the HP 200LX and the HP 100LX, Hewlett Packard.

Lotus cc:Mail Release 2, Lotus Development Corporation (1991–1993).

User's Guide Lotus Organizer Release 1.0, Lotus Development Corporation (1992).

FileMaker Pro User's Guide, Claris Corporation (1990, 1992).

Poesio et al., "Metric Constraints for Maintaining Appointments: Dates and Repeated Activities".

Slater, "Newton's Legacy; 3COM and Microsoft Battle for Market Share; Apple Newton, 3Com Palm III, Microsoft Palm–size PC peronal digital assistants; Product Information", Information Access Company (1998).

Negrino, "ACT 2.5.1, ACT for Newton 1.0", UMI, Inc. (1996).

Zilber, "Toy story; personal digital assistants; Product Information", Information Access Company (1996).

Wingfield, "Desktop to Newton connectivity", UMI, Inc. (1996).

"Now Software Announces Updated Synchronization Software for Newton 2.0 Devices; Now Synchronize Simultaneously Updates MessagePad, Now Up–to–Date & Contact", Business Wire, Inc. (1995).

"Claris Ships FileMaker Pro 3.0 for Macintosh and Windows", Business Wire, Inc. (1995).
Alsorp, "Distributed Thinking; Realizing the gravity of its PDA problems, Apple has drawn me back to Newton", InfoWorld Media Group (1995).
Rubin, "Now Software stays in sync; Now Synchronize file synchronization software for Macs and Newton PDAs; Software Review; EvaluationBrief Article", Information Access Company (1995).
"Now Calendar/Scheduler/Contact Mgr for Mac Update", Post–Newsweek Business Information Inc. (1995).
Staten, "csInStep middleware lets Newton talk to PIMs; Concierge Software LC's csInStep; Brief Article; Product Announcement; Brief Article", Information Access Company (1995).
Baum, "Designing Moble applications; A new approach needed for on–the–road systems", InfoWorld Media Group (1994).
Parkinson, "Remote users get in sync with office files; News Analysis", Information Access Company (1994).
Informix Guide to SQL Tutorial Version 7.1, Dec. 1994.
Oracle7 Distributed Database Technology and Symmetric Replication, Oracle White Paper, Apr. 1995.
Oracle7 Server Distributed Systems, vol. II: Replicated Data, Release 7.3, Feb. 1996.
Oracle7™ Server SQL Manual Release 7.3, Feb. 1996.
Extended Systems' Preliminary Invalidity Contentions.
Extended Systems' First Supplemental Preliminary Invalidity Contentions.
Extended Systems, Inc.'s Preliminary Claim Contructions and Preliminary Identification of Extrinsic Evidence.
Patent Local Rule 4–2 Preliminary Claim Constructions and Extrinsic Evidence.
Joint Claim Construction and Prehearing Statement.
Extended Systems' Second Supplemental Preliminary Invalidity Contentions [Re: Reexamination Requests for the '390, '664, and '529 Patents].
Pumatech, Inc.'s Opening Claim Construction Brief; Declaration of Marc David Peters in Support of Pumatech, Inc.'s Opening Claim Construction Brief.
Extended Systems, Inc.'s Responsive Claim Construction Brief; Declaration of Jordan Trent Jones in Support of Extended Systems, Inc.'s Responsive Claim Construction Brief.
Supplemental Decalaration of Marc David Peters in Support of Pumatech, Inc.'s Reply Claim Construction Brief.
Pumatech's Revised [Proposed] Claim Construction Order.
Pumatech, Inc.'s Reply Claim Construction Brief.
Statement of Recent Decision.
Pumatech's [Proposed] Claim Construction Order.
Synchrologic's Preliminary Invalidity Contentions.
Adly, "HARP: A Hierarchical Asynchronous Replication Protocol for Massively Replicated Systems," Computer Laboratory, Cambridge University, United Kingdom (undated).
Adly et al., "A Hierarchical Asynchronous Replication Protocol for Large Scale Systems," Computer Laboratory, Cambridge University, United Kingdom, Computer Science Department, Alexandria University, Egypt (undated).
Alexander, "Designed, sold, delivered, serviced," Computerworld Client/Server Journal, p. 43 (Oct. 1, 1995).
"All I need is a miracle; computer–aided educational packages; Small Wonders," Coastal Associates Publishing L.P. (Mar. 1992).
Alonso et al., "Database System Issues in Nomadic Computing," Matsushita Information Technology Laboratory, New Jersey (undated).
Badrinath et al., "Impact of Mobility on Distributed Computations," Operating Systems Review (Apr. 1, 1993).
Barbara et al., "Sleeper and Workaholics: Caching Strategies in Mobile Environments (Extended Version)" (Aug. 29, 1994).
Brandel, "New offerings fuel revival of PIM," Computerworld, p. 39 (Sep. 12, 1994).
Brodersen, "InfoPad—An Experiment in System Level Design and Integration," (Mar. 1, 1997).
Demers et al., "The Bayou Architecture: Support for Data Sharing Among Mobile Users," Computer Science Laboratory, Xerox Palo Alto Research Center, California (undated).
DeVoe et al., "SOFTWARE: Day–Timer Organizer 2.0 based on format of paper–based PIM," InfoWorld, vol. 17 (Aug. 21, 1995).
Froese, "File System Support for Weakly Connected Operation," pp. 229–238 (undated).
Greenberg et al., "Real Time Groupware as a Distributed System: Concurrency Control and its Effect on the Interface," Procs. of the ACM CSCW Conf. On Computer Supported Cooperative Work, Oct. 22–26, North Carolina, ACM Press (Jan. 1, 1994).
Guy, "Ficus: A Very Large Scale Reliable Distributed File System," Technical Report CSD–910018, Computer Science Dept. UCLA (Technical Report) (Jun. 3, 1991).
Guy et al., "Implementation of the Ficus Replicated File System," appeared in Procs. Of the Summer USENIX Conf., Anaheim, CA, pp. 63–71 (Jun. 1, 1990).
Haber, "Renegade PIMS," Computerworld, p. 109 (Dec. 12, 1994).
Hammer et al., "An Approach to Resolving Semantic Heterogeneity in a Federation of Autonomous, Heterogeneous Database Systems," Computer Science Department, University of Southern California (undated).
Hammer et al., "Object Discovery and Unification in Federated Database Systems," University of Southern California (undated).
HP and IntelliLink connect HP 95LX with HP NewWave; IntelliLink for the HP NewWave; product announcement, HP Professional (Aug. 1991).
"HP announces expanded memory version of palmtop PC, introduces 1–Megabyte HP 95LX and 1–Megabyte memory cards," Business Wire, Inc. (Mar. 4, 1992).
Huston et al., "Disconnected Operation of AFS," CITI Technical Report 93–3, Center for Information Technology Integration, University of Michigan (Jun. 18, 1993).
IBM Dictionary of Computing, Tenth Edition, 1993, pp. 268, 269, 31_.
IBM Dictionary of Computing, Tenth Edition, 1993, pp. 165, 268, 349, 370, 417.
IEEE Standard Dictionary of Electrical and Electronics Terms, Fourth Edition, 1988, p. 372, 368, 509, 563.
Imielinski, "Mobile Computing—DataMan Project Perspective," Rutgers University (undated).
"IntelliLink 2.2: the software connection from desktop to palmtop; Software Review; IntelliLink 2.2; Evaluation," PC Magazine (Apr. 28, 1992).
"IntelliLink transfers palmtop, PC data; communications software from IntelliLink Inc; brief article; Product Announcement," PC Week (Nov. 18, 1991).

Jacobs et al., "A Generalized Query–by–Example Data Manipulation Language Based on Database Logic," IEEE Transactions on Software Engineering, vol. SE–9, No. 1 (Jan. 1983).

Jenkins, "Users struggle with E–mail Woes," Computerworld, p. 97 (Oct. 24, 1994).

Johnson et al., "Hierarchical Matrix Timestamps for Scalable Update Propogation," submitted to the 10th Int. Workshop on Distributed Algorithms (Jun. 25, 1996).

Joshi et al., "A Survey of Mobile Computing Technologies and Applications," (Oct. 29, 1995).

Kistler et al., "Disconnected Operation in the Coda File System," School of Computer Science, Carnegie Melon University, Pennsylvania (undated).

Krill, "NETWORKING: Tech Update," InfoWorld, vol. 18 (Feb. 12, 1996).

Kumar et al., "Log–Based Directory Resolution in the Coda File System," School of Computer Science, Carnegie Melon University, Pennsylvania (undated).

Larson et al., "A Theory of Attribute Equivalence in Databases with Application to Schema Integration," IEEE Transactions on Software Engineering, vol. 15, No. 4, Apr. 1989.

Lomet, D., Using timestamping to optimize two phase commit; Parallel and Distributed Information Systems, 1993, Proc. Of the 2nd Int. Conf., Jan. 20–22, 1993, pp. 48–55.

Mannino et al., "Matching Techniques in Global Schema Design," IEEE 1984.

Marshall, "Product Reviews: Windows contact managers," InfoWorld, vol. 18 (Mar. 25, 1996).

McGoveran, "Distributed not yet delivered," Computerworld, p. 112 (Jun. 6, 1994).

Meckler Corporation, "Palmtop–to–desktop linkage software," Database Searcher (Jun. 1992).

Microsoft Press Computer Dictionary, Second Edition, 1994, p. 164.

Microsoft Press Computer Dictionary, Second Edition, 1994, pp. 105, 217, 227, 228.

Microsoft Press Computer Dictionary, Third Edition, 1997, pp. 194, 228, 234, 449.

Milliken, "Resource Coordination Objects: A State Distribution Mechanism," (DRAFT) (Dec. 10, 1993).

Nash, "Replication falls short," Computer world, p. 65 (Nov. 21, 1994).

Noble et al., "A Research Status Report for Adaptation for Mobile Data Access," School of Computer Science, Carnegie Melon University (undated).

"PackRat PIM gets older and wiser with Release 4.0; PIM update sports enhanced interface, greater ease of use," InfoWorld (Dec. 23, 1991).

"Palmtop PCs: power by the ounce; Hardware Review; overview of six evaluations of palm–top computers; includes related articles on Editor's Choices, suitability–to–task ratings, impression by individual users; evaluation," PC Magazine (Jul. 1991).

"Pen–based PCs ready for prime time; includes related article on comparison of operating systems, list of vendors of pen–based products," PC–Computing (Nov. 1991).

Perera, "Synchronization Schizophrenia," Computerworld Client/Server Journal, p. 50 (Oct. 1, 1995).

Petersen et al., "Bayou: Replicated Database Services for World–wide Applications," Computer Science Laboratory, Xerox Palo Alto Research Center, California (undated).

"Product comparison: Atari Portfolio, Casio Executive BOSS, HP 95LX, Poqet PC, Psion series 3, Sharp Wizard," InfoWorld (Dec. 16, 1991).

"Product Comparison: Personal information managers," InfoWorld, vol. 17 (Aug. 7, 1995).

Qu et al., Technical Report entitled "Mobile File Filtering," TR–CS–97–02–Australian National University (Feb. 1, 1997).

Radosevich, "Replication mania," Computerworld Client/Server Journal, p. 53 (Oct. 1, 1995).

Ratner et al., "The Ward Model: A Replication Architecture for Mobile Environments," Department of Computer Science, University of California (undated).

Reiher et al., "Peer–to–Peer Reconciliation Based Replication for Mobile Computers," UCLA (undated).

Reiher et al., "Resolving File Conflicts in the Ficus File System," Department of Computer Science, University of California (undated).

Ricciuti, "Object database server," InfoWorld, vol. 18 (Jan. 29, 1996).

"Riding the NewWave from PC to Palmtop: IntelliLink lets NewWave users transfer files," InfoWorld (Jun. 3, 1991).

Saltor et al., "Suitability of data models as canonical models for federated databases," Universitat Politecnica de Catalunya, Spain (undated).

Salzberg, B., Timestamping After Commit, Procs. Of the Third Int. Conf. On Parallel Distributed Information Systems, Sep. 28–30, 1994, pp. 160–167.

Satyanarayanan, "Coda: A Highly Available File System for a Distributed Workstation Environment," School of Computer Science, Carnegie Mellon University (undated).

Satyanarayanan, "Fundamental Challenges in Mobile Computing," School of Computer Science, Carnegie Mellon University (undated).

Satyanarayanan, "Mobile Information Access," IEEE Personal Communications, vol. 3, No. 1 (Feb. 1996).

Sherman, "Information Technology: 'What Software Should I Use to Organize My Life'," (undated).

Sheth et al., "A Tool for Integrating Conceptual Schemas and User Views," IEEE 1988.

Schilit et al., "The ParcTab Mobile Computing System," Xerox Palo Alto Research Center, California (undated).

SPI Database Software Technologies Record Displays: Record 2, Serial No. TDB0291.0094 and Record 4, Serial No. iets0901.0073 (undated).

Staten, "PowerMerge 2.0 ships; syncs moved filed," MacWEEK, vol. 8, p. 38(1) (Jan. 3, 1994).

Tait, Doctoral Thesis entitled "A File System for Mobile Computing," (Jan. 1, 1993).

Tolly, "Enhanced Notes 4.0 gets thumbs–up," Computerworld, p. 54 (Dec. 18, 1995).

Webster's Ninth New Collegiate Dictionary, 1986, pp. 114, 436, 440, 462, 573, 597, 620, 717, 906, 963, 979, 989, 1000, 1053, 1130, 1142, 1152, 1162, 1166.

Wiederhold, Gio, Database Design, Second Edition, McGraw–Hill Book Company, 1983, p. 2.

Zaino, "Tapping the Top Values in PDAs—Personal digital assistants that sell for as little as $300 can put a PC in the palm of your hand. Get the scoop on 8 contenders," HomePC, p. 97 (Oct. 1, 1996).

Zisman et al., "Towards Inoperability in Heterogeneous Database Systems," Imperial College Research Report No. DOC 95/11 (Dec. 1, 1995).

* cited by examiner

EX PARTE REEXAMINATION CERTIFICATE ISSUED UNDER 35 U.S.C. 307

THE PATENT IS HEREBY AMENDED AS INDICATED BELOW.

Matter enclosed in heavy brackets [ ] appeared in the patent, but has been deleted and is no longer a part of the patent; matter printed in italics indicates additions made to the patent.

AS A RESULT OF REEXAMINATION, IT HAS BEEN DETERMINED THAT:

The patentability of claims **1**, **2**, **12**, **14**, **15**, **21**, **25–28**, **38**, **40**, **41**, **47**, **51** and **52** is confirmed.

Claims **4–11**, **13**, **16–20**, **22–24**, **30–37**, **39**, **42–46** and **48–50** are cancelled.

Claims **3** and **29** are determined to be patentable as amended.

3. The method of claim **1** further comprising selecting records of the [first] *second* database based on a selected criterion, wherein synchronizing the at least one of the selected records of the first database with a record of the second database further includes synchronizing the at least one of the selected records of the first database with at least one of the selected records *of* the second database.

29. The computer program of claim **27** further comprising instructions for selecting records of the [first] *second* database based on a selected criterion, wherein synchronizing the at least one of the selected records of the first database with a record of the second database further includes synchronizing the at least one of the selected records of the first database with at least one of the selected records *of* the second database.

* * * * *

EXHIBIT C

US006141664A

United States Patent [19]

Boothby

[11] Patent Number: 6,141,664

[45] Date of Patent: *Oct. 31, 2000

[54] **SYNCHRONIZATION OF DATABASES WITH DATE RANGE**

[75] Inventor: **David J. Boothby**, Nashua, N.H.

[73] Assignee: **Puma Technology, Inc.**, San Jose, Calif.

[*] Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

[21] Appl. No.: **08/748,645**

[22] Filed: **Nov. 13, 1996**

[51] **Int. Cl.**[7] **G06F 17/30**

[52] **U.S. Cl.** **707/201**; 707/203

[58] **Field of Search** 707/10, 8, 104, 707/200, 201, 203

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,432,057	2/1984	Daniell et al.	364/300
4,807,182	2/1989	Queen	364/900
4,819,156	4/1989	DeLorme et al.	364/200
4,827,423	5/1989	Beasley et al.	364/468.02
4,866,611	9/1989	Cree et al.	364/300
4,875,159	10/1989	Cary et al.	364/200
4,956,809	9/1990	George et al.	364/900
4,980,844	12/1990	Demjanenko et al.	702/56
5,065,360	11/1991	Kelly	395/800
5,136,707	8/1992	Block et al.	395/600
5,142,619	8/1992	Webster, III	395/157
5,170,480	12/1992	Mohan et al.	707/201
5,187,787	2/1993	Skeen et al.	395/600
5,210,868	5/1993	Shimada et al.	395/600
5,228,116	7/1993	Harris et al.	706/50
5,237,678	8/1993	Keuchler et al.	395/600
5,251,151	10/1993	Demjanenko et al.	702/56
5,251,291	10/1993	Malcolm	395/146
5,261,045	11/1993	Scully et al.	395/161

(List continued on next page.)

OTHER PUBLICATIONS

"FRx extends reporting power of Platinum Series: (IBM Desktop Software's line of accounting software)", Doug Dayton, PC Week, v. 8, n. 5, p. 29(2), Feb. 4, 1991.

"The Big Picture (Accessing information on remote data management system)", UNIX Review, v. 7, n. 8, p. 38(7), Aug. 1989.

"Logical Connectivity: Applications, Requirements, Architecture, and Research Agenda," Stuart Madnick & Y. Richard Wang, MIT, System Sciences, 1991, Hawaii Int'l, vol. 1, IEEE, Jun. 1991.

"Automatically Synchronized Objects", Research Disclosure #29261, p. 614 (Aug. 1988).

Cobb et al., "Paradox 3.5 Handbook 3rd Edition", Bantam (1991), pp. 803–816.

Alfieri, "The Best Book of: WordPerfect Version 5.0", Hayden Books (1988), pp. 153–165 and 429–435.

IntelliLink Brochure (1990).

(List continued on next page.)

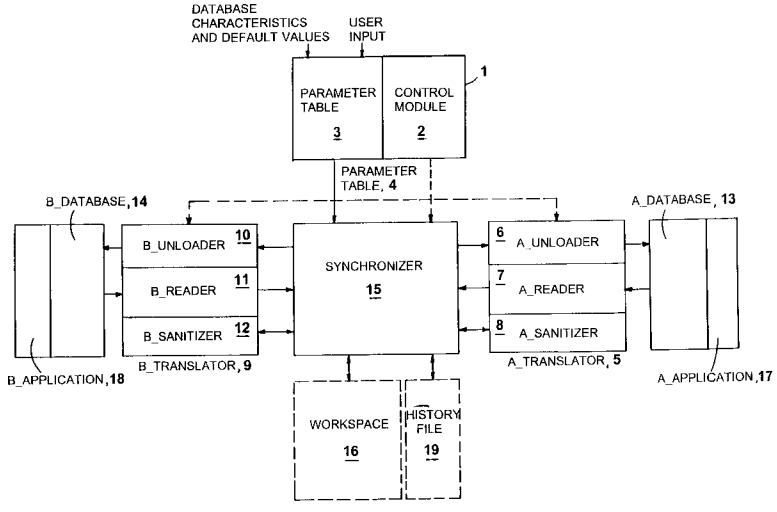
Primary Examiner—Thomas G. Black
Assistant Examiner—John C. Loomis
Attorney, Agent, or Firm—Fish & Richardson P.C.

[57] **ABSTRACT**

A method of synchronizing at least a first and a second database each containing dated records such as events, wherein the records of the first and second databases are synchronized across a narrow date range narrower than the date range of the records of at least one of the databases. A prior synchronization can be performed across a prior date range. The date range of the prior synchronization is stored, along with the history file containing information representative of the content of the databases following the prior synchronization. When a current synchronization is performed, it is performed across a date range that combines the prior date range with the current date range.

80 Claims, 41 Drawing Sheets

Microfiche Appendix Included
(8 Microfiche, 691 Pages)



U.S. PATENT DOCUMENTS

5,261,094	11/1993	Everson et al.	395/600
5,272,628	12/1993	Koss	364/419
5,278,978	1/1994	Demers et al.	707/101
5,283,887	2/1994	Zachery	395/500
5,293,627	3/1994	Kato et al.	395/558
5,301,313	4/1994	Terada et al.	395/600
5,315,709	5/1994	Alston, Jr. et al.	395/600
5,327,555	7/1994	Anderson	395/600
5,333,252	7/1994	Brewer, III et al.	395/148
5,333,265	7/1994	Orimo et al.	395/200.31
5,333,316	7/1994	Champagne et al.	707/8
5,339,392	8/1994	Risberg et al.	395/161
5,339,434	8/1994	Rusis	395/700
5,355,476	10/1994	Fukumura	707/1
5,375,234	12/1994	Davidson et al.	707/202
5,392,390	2/1995	Crozier	395/161
5,396,612	3/1995	Huh et al.	395/575
5,434,994	7/1995	Shaheen et al.	395/500
5,444,851	8/1995	Woest	395/200.52
5,463,735	10/1995	Pascucci et al.	395/200.52
5,475,833	12/1995	Dauerer et al.	395/600
5,511,188	4/1996	Pascucci et al.	707/203
5,519,606	5/1996	Frid-Nielsen et al.	364/401
5,568,402	10/1996	Gray et al.	395/200.54
5,583,793	12/1996	Gray et al.	395/200.53
5,600,834	2/1997	Howard	707/201
5,666,530	9/1997	Claric et al.	707/201
5,666,553	9/1997	Crozier	707/540
5,682,524	10/1997	Freund et al.	711/5
5,684,984	11/1997	Jones et al.	707/10
5,684,990	11/1997	Boothby	707/203
5,701,423	12/1997	Crozier	345/335
5,710,922	1/1998	Alley et al.	707/201
5,727,202	3/1998	Kucala	707/10
5,729,735	3/1998	Meyering	707/10
5,758,150	5/1998	Bell et al.	707/210
5,758,355	5/1998	Buchanan	707/201
5,778,388	7/1998	Kawamura et al.	707/203

OTHER PUBLICATIONS

User Manual for PC–Link for the B.O.S.S. and the PC–Link for the B.O.S.S., Traveling Software, Inc. (1989).

User Manual for Connectivity Pack for the HP 95LX, Hewlett Packard Company (1991).

Organizer Link II Operation Manual, Sharp Electronics Corporation, no date.

"Open Network Computing—Technical Overview," Sun Technical Report, Microsystems, Inc., pp. 1–32 (1987).

Zahn et al., Network Computing Architecture, pp. 1–11; 19–31; 87–115; 117–133; 187–199; 201–209 (1990).

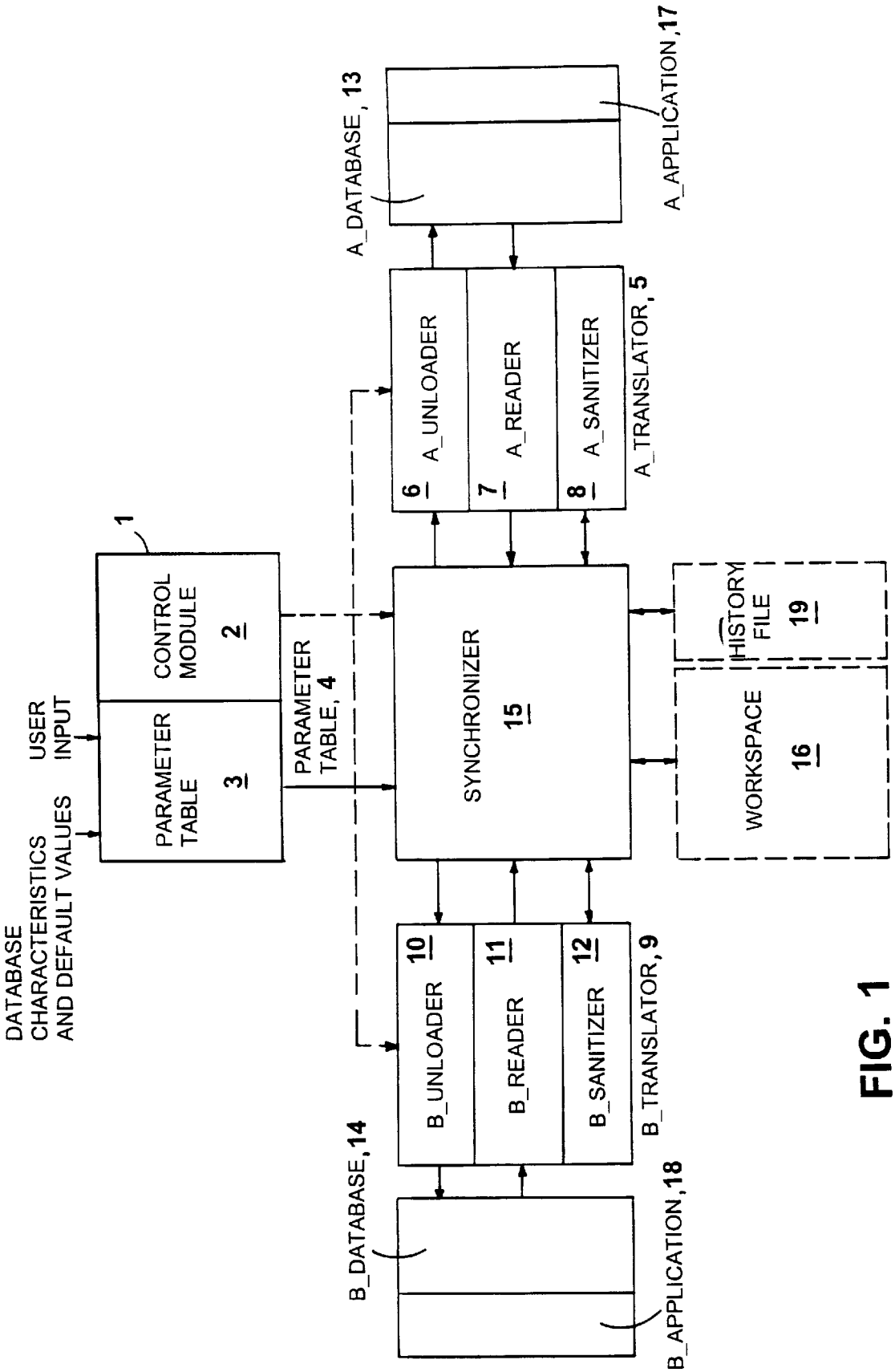

A_APPLICATION, 17
A_DATABASE, 13
A_TRANSLATOR, 5
6 A_UNLOADER
7 A_READER
8 A_SANITIZER
1
CONTROL MODULE 2
PARAMETER TABLE 3
USER INPUT
DATABASE CHARACTERISTICS AND DEFAULT VALUES
PARAMETER TABLE, 4
SYNCHRONIZER 15
HISTORY FILE 19
WORKSPACE 16
B_TRANSLATOR, 9
10 B_UNLOADER
11 B_READER
12 B_SANITIZER
B_DATABASE, 14
B_APPLICATION, 18

FIG. 1

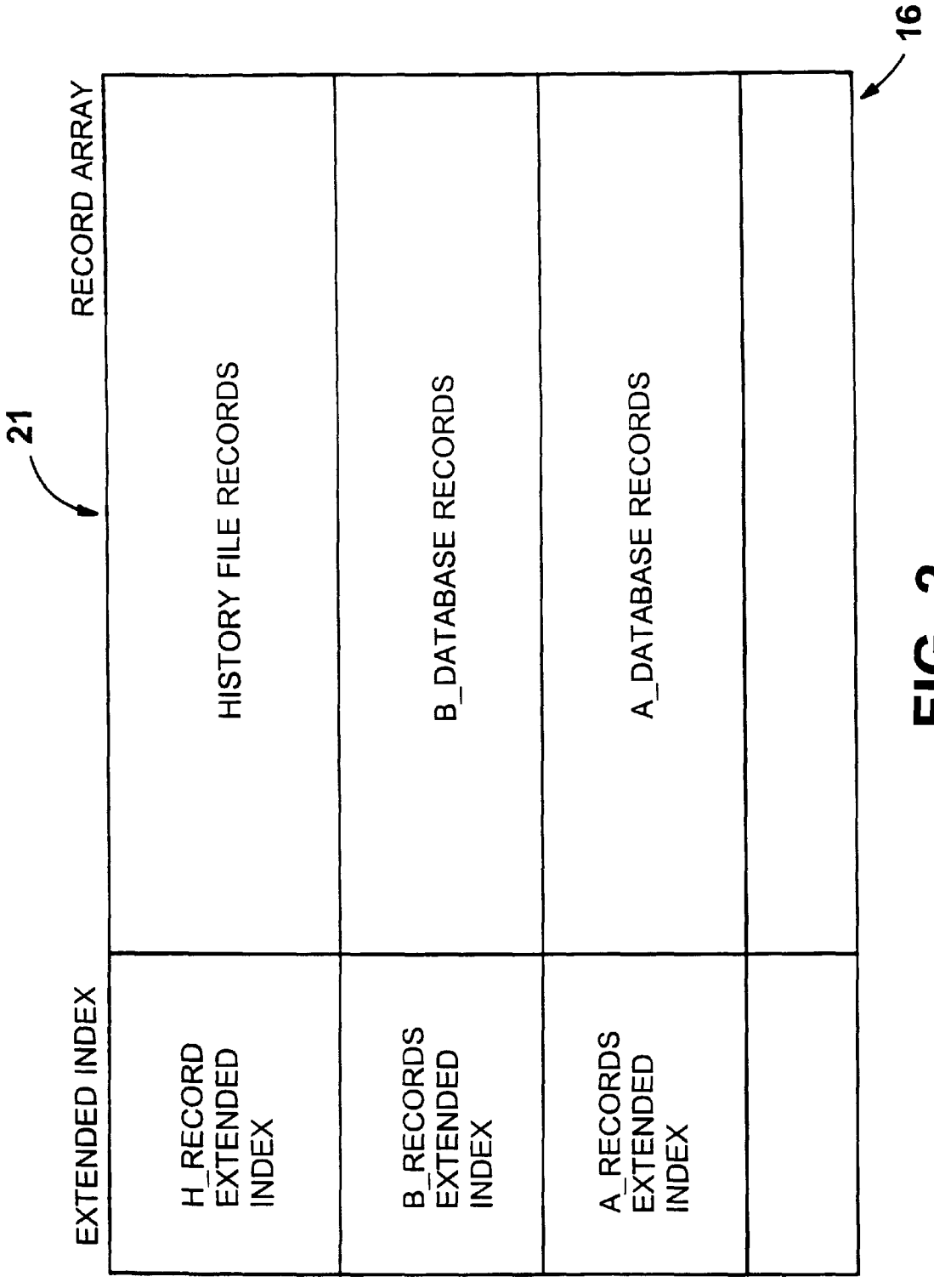

16
21
RECORD ARRAY
EXTENDED INDEX
HISTORY FILE RECORDS
B_DATABASE RECORDS
A_DATABASE RECORDS
H_RECORD EXTENDED INDEX
B_RECORDS EXTENDED INDEX
A_RECORDS EXTENDED INDEX

FIG. 2

Pseudo Code for Translation Engine Control Module

100. CREATE Parameter_Table from User Input A & B database characteristics and default values
101. INSTRUCT Synchronizer to initialize itself
102. INSTRUCT Synchronizer to LOAD the History_File into its WORKSPACE
103. INSTRUCT B_Translator to LOAD all of B_records from B_Database and SEND to Synchronizer (Synchronizer STORES these records in WORKSPACE)
104. INSTRUCT A_Translator to SANITIZE B_records that were just LOADED (A_Translator USES Synchronizer services to read and write records in the WORKSPACE; Synchronizer maps these records using the B→A_Map before sending them to A_Translator and maps them back using A→B_Map before rewriting them into the WORKSPACE)
105. INSTRUCT A_Translator to LOAD all of A_records from A_Database and SEND to Synchronizer (Synchronizes STORES these records in WORKSPACE by first mapping them using the A→B_Map and then storing in their new form)
106. INSTRUCT B_Translator to SANITIZE A_records that were just LOADED (B_Translator uses Synchronizer services to read and write records in the WORKSPACE)
107. INSTRUCT Synchronizer to do CAAR (Conflict Analysis And Resolution) on all the records in WORKSPACE.
108. INFORM user exactly what steps Synchronizer proposes to take (i.e. Adding, Changing, and Deleting records). WAIT for User
109. IF user inputs NO, THEN ABORT
110. INSTRUCT B_Translator to UNLOAD all applicable records to B_Database.
111. INSTRUCT A_Translator to UNLOAD all applicable records to the A_Database.
112. INSTRUCT Synchronizer to CREATE a new History File.

FIG. 3

Pseudocode for Generating Parameter Table

```
{Get Input from the user}
150.    ASK user whether to synchronize based on a previously stored set of preferences
        (Previous_Preferences) or based on a set of new preferences (New_Preferences)
151.    IF New_Preferences THEN
152.            ASK user whether Incremental_Synchronization or Synchronization_from_Scratch
153.            ASK user following information and STORE in Parameter_Table
        a.      A_Application and B_Application Names
        b.      ADB and BDB Names
        c.      ADB and BDB Locations
        d.      Which sections to Synchronize
        e.      Conflict Resolution Option: IGNORE, ADD, DB WINS, BDB WINS, or NOTIFY
        f.      Other user preferences
154.            ASK user whether wants default mapping for the selected sections of the two databases or wants
                to modify default mapping
155.            LOAD A_Database→B_Database (2)
156.            IF Default_Mapping THEN
157.                    STORE A→B_Map AND B→A_Map in Parameter_Table
158.            END IF
159.            IF Modified_Mapping THEN
160.                    DISPLAY A→B_Map and B→A_Map
161.                    ASK user to modify Maps as desired
162.                    STORE the new A→B_Map and B→A_Map in the Parameter_Table
163.            END IF
165.            END IF
```

FIG. 4A

166. IF Previous_Preferences THEN
167. ASK user whether Incremental_Synchronization or Synchronization_from_Scratch
168. STORE in Parameter_Table
169. LOAD Previous Preferences regarding which databases, mapping, and so on
170. STORE in the Parameter_Table
171. END IF
{User now specifies Date Range}
172. ASK user to choose Date Range Option
a. Previously chosen Automatic_Date_ Range calculated from today
b. Input New Automatic_Date_Range
c. Input static Date Range for this Synchronization
d. All dates
173. CALCULATE Start_Current_Date_Range and End_Current_Date_Range based on values from step 171
174. STORE in Parameter_Table
175. LOAD parameters setting out characteristics of A_Database and B_Database from Parameters database, including
a. Field_List_A and Field_List_B
b. A_Translator and B_Translator Module Identifiers
c. ADB_Section_Names and BDB_Section_Name
176. STORE in Parameters Table

FIG. 4B

200. RECEIVE following from Parameter Table
1) Name of A_App
2) Name of B_App
3) Name and Location of A_DB
4) Name and Location of B_DB
5) Section name of A_Application to be synchronized
6) Section name of B_Application to be synchronized
7) Incremental_Synchronization or Synchronization_From_Scratch Flags
201. SEARCH for H_File matching Parameters 1-6
202. If Found H-File and Incremental_Synchronization THEN DO nothing
203. IF Found H-File and Synchronization_from_Scratch, THEN DELETE H_File
204. IF NOT found H-File, THEN SET Synchronization_from_Scratch AND ASSIGN file name for history file.
205. LOAD from Parameter_Table Start_Current_Date_Range and End_Current_Date_Range
206. LOAD from Parameter_Table Field_Lists for A-DB and B-DB and field and mapping information
207. If Incremental_Synchronization THEN COMPARE Field_Lists and Maps from Parameter_Table with History_Field_Lists and Maps
208. IF exact match THEN DO nothing
209. IF not exact match THEN DELETE H_file AND SET Synchronization_from_Scratch
210. CREATE WORKSPACE using Field_List_B
211. If Incremental_Synchronization THEN Copy H_file into WORKSPACE
212. FOR each H-Record update
{analyze & update source of extended index}
213. Do Nothing to NEXT_IN_FIG
214.

FIG. 5A

```
215.            FIND H-Record with matching KeyFields
216.            IF FOUND THEN Update NEXT_IN_SKG of H-Record
217.            IF Appointment type and Non-Recurring record THEN
218.                    IF (Start_Date after End_Previous_Date_Range) OR (End_Date before
                        Start_Previous_Date_Range) THEN SET Bystander Flag END IF
219.                    IF (Start_Date after End_Current_Date_Range) OR (End_Date before Start_Current
                        _Date_Range) THEN SET Outside_Current_Range END IF
{Recurring records}
220.                    ELSE
221.                    Fan_Out_Recurrence_Pattern for H-Record
222.                    SET Bystander Flag and Outside_Current_Range Flags for H-Record
223.                    For all Fanned out Instances
224.                            IF (Start_Date Before End_Previous_Date_Range) OR (End_Date after
                                Start_Previous_Date_Range) THEN UN-SET Bystander Flag of Recurring H-
                                Record END IF
225.                            IF (Start_Date before End_Current_Date_Range) OR (End_Date after
                                Start_Current _Date_Range) THEN UN-SET Outside_Current_Range END IF
226.                    END LOOP
227.            END IF
228.    END LOOP
```

FIG. 5B

```
235.  LOAD Rep_Basic, Start_Date, Stop_Date, Frequency
236.  CALCULATE Useful Start_Date and Useful_Stop_Date based on Start_Date, Stop_Date, Max_Fan_Out
      and Usefulness_Range_Future & Past
237.  REPEAT
238.        CALCULATE Next_Date based on Useful_Start_Date, Current_Date, Rep_Basic, Frequency,
            Max_Fan_Out
239.        IF Next_Date After Useful_Stop_Date, THEN EXIT
240.        STORE Next_Date
241.              Fan_Out_Date_Array
242.        Current_Date = Next_Date
243.  END LOOP
```

FIG. 6

Pseudocode for Key_Field_Match

```
250.    RECEIVE Key_Field_Hash and WORKSPACE_ID
251.    For all records in WORKSPACE
252.        IF Match_Hash_Value equals Hash Values of Record THEN LOAD the two records
253.            COMPARE the key fields two records
254.            IF Exact Match THEN SET Match_Found
255.            EXIT LOOP
256.        END IF
257.    END LOOP
258.    If Match_Found THEN SEND Success Flag and WORKSPACE ID of Matching record
```

FIG. 7

Pseudo Code for Loading Records of B_database into WORKSPACE

B_Translator:

```
300.    FOR ALL Records in B_DB
301.            READ Record from B_DB
302.            IF (record outside of combination of Current_Date_Range and Previous_Date_Range), THEN
                GOTO END LOOP
303.            IF NOT right origin tag for this synchronization THEN GOTO END LOOP
304.            SEND Record to Synchronizer 325-236
305.    END LOOP
```

Synchronizer:

```
325.    RECEIVE B_Record
326.    STORE in WORKSPACE in next available space
```

FIG. 8

Pseudo Code for Generic A_Sanitization of B_DB Records in Workspace

A_Translator:

```
350.  REPEAT
351.        FOR EVERY Field in an A_Record
352.              REQUEST Field from Synchronizer
353.              IF Last_Field, THEN EXIT LOOP
354.              SANITIZE Field, according to A_Sanitization rules
355.        END LOOP
356.        IF Last_Field, THEN EXIT LOOP
357.        SANITIZE Record according to A_Sanitization rule
358.        FOR EVERY Field in an A_Record
359.              SEND Field value to Sanitizer
360.        END FOR
361.  UNTIL EXIT
```

SYNCHRONIZER:

```
375.  In Response to Request for Field by A_Sanitizer
376.  REPEAT UNTIL LAST RECORD
377.        READ B_Record
378.        MAP Record according to B_A Map
379.        REPEAT UNTIL A_Translator Request a field from a new Record
380.              SEND REQUESTED B_field to A_Translator
381.              WAIT FOR RETURN of B_Field from A_Translator
382.              STORE field Value in Mapping_Cache
383.        END LOOP
384.        MAP record in Cache according to A→B Map
385.        STORE record in WORKSPACE
386.  END LOOP
387.  SEND Last_Field flag in response to REQUEST
```

FIG. 9

```
Specific Example of Sanitization
400.    IF StartDate and EndDate are both blank
401.            Make Alarm Date blank and make Alarm Flag = FALSE
402.      ELSE IF EndDate is blank THEN SET EndDate = StartDate
403.            ELSE IF StartDate is blank OR is greater than EndDate THEN      SET StartDate =
                EndDate END IF

404.            IF AlarmFlag is TRUE and AlarmDate is blank THEN SET  AlarmDate = StartDate
405.                    ELSE IF AlarmDate is greater than EndDate THEN SET AlarmDate = EndDate
406.    END IF
```

FIG. 10

Pseudo_code for Orientation Analysis (Index Value analysis)

```
450.  FOR EVERY Record of database in WORKSPACE
451.        CALCULATE Key_Field_Hash from Section Subtype value for the record & all Mapped Key
           Fields
452.        CALCULATE Non_key_Fields_Hash from all Mapped Non_key Fields which are not marked as
           No_Reconcile
453.        CALCULATE Exclusion_List_Hash, if Recurring_Master, from Exclusion_List
454.        CALCULATE Non_Date_Hash from all non-date mapped non-key fields which are not
           No_Reconcile fields
455.        If B_Record THEN CALCULATE B_ID_Hash
456.        IF A_Record THEN CALCULATE A_DB ID Hash
457.        CALCULATE Start_Date_Time values (for Appointments and TO DO Lists)
458.        CALCULATE End_Date_Time
459.        IF Recurring Item and No instances in Current Date Range THEN SET Out_Of_Range
460.        IF (Start_Date After End_Current_Date_Range OR End_DateBefore Start_Current_Date_Range,
           THEN SET Out_Of_Range_Flag ELSE SET IN_Range Flag
461.        END IF
462.        IF Matching Unique ID in H_records THEN ADD to CIG
463.        IF Matching Unique_ID in H_records, THEN SET WARNING FLAG
464.        IF an H or current database record with same key field values (using Key_Field_Match function,
           Fig. 7), THEN ADD Current Record to SKG of the H or A_record
465.  END LOOP
```

FIG. 11

Pseudocode for Conflict Analysis And Resolution (CAAR)

500. Analyze ID_Bearing FIGS.
501. Analyze and expand ID_bearing CIGs
502. Finding Matches between Recurring Items and Non-Unique ID bearing Instances
503. Analyze SKGs
504. SET CIG Types

FIG. 12

Pseudocode for Analyzing ID_bearing FIGs

```
550.  FOR EVERY Recurring Master of ID_Bearing FIGS in H_file
551.      FOR EVERY FIG H_Record in Recurring Master FIG
552.          REMOVE Record from SKG it belongs to
553.          IF Record is a singleton CIG, THEN ADD to New_Exclusion_List
554.          IF Record is a doubleton CIG, THEN
555.              IF the two Records in CIG are Identical, THEN remove other RECORD from
                  its SKG
556.              END IF
557.              ELSE IF the two records are NOT Identical, THEN ADD FIG record to
                  New_Exclusion_List and change records into singleton CIGs
558.          END IF
559.      END LOOP
560.      CREATE Synthetic Master record entry in WORKSPACE
561.      COPY value from one of the CIG mates into Synthetic Master
562.      COPY Rep Basic (i.e. recurrence pattern) from the Recurring Master into Synthetic Master
563.      COPY Exclusion List from the database Recurring Master into Synthetic Master and MERGE
          with New_Exclusion_List
564.      COMPUTE all Hash values for Synthetic Master
565.      CREATE new FIG between Synthetic Master the CIGmates of the H-FIG records
566.      CREATE CIG among the three Recurring Masters

{Fan Out Creep}

567.      Fan out Recurring Master with Previoius_Date_Range
568.      Fan out Recurring Master with Current_Date_Range
569.      IF two date arrays are NOT identical, THEN MARK CIG with Fan_Out_Creep flag
570.      MARK all Records in H_File Recurring Master FIG and Synthetic Master FIG as
          Dependent_FIG
571.  END LOOP
```

FIG. 13

Pseudo Code for EXPANDING ID_BASED CIGs

```
600.  For each H_record,
601.        IF single record CIG, THEN GO TO END LOOP
602.        IF triple record CIG, THEN REMOVE CIG records from their SKGs
603.        IF Dependent_FIG, THEN GO TO END LOOP
604.        IF record needed to make triple has to be from a DB with unique ID, THEN GO TO END
            LOOP
605.        For all members of SKG to which H_record belongs
606.              IF Non_Key_Field_Hash of H_record and SKG_record Match, THEN
607.                    IF Exact Match of all fields with H item THEN Strong_Match is found  END
                      IF
608.              ELSE
609.                    IF H_Record is a Recurring Master, THEN Find Fanned Instance (Table
                      Recurring Master/Instance Match) which is Strong_Match
610.              END IF
611.        END LOOP
612.        IF Strong_Match is found AND IF the SKG_Record is Weak_Match member of a CIG, THEN
613.              REMOVE SKG Record from Weak_Match CIG AND Seek Alternate Weak_Match for
                  the CIG
614.              ADD SKG record to Current doubleton CIG AND Record for the Weak_Match_CIG
615.              REMOVE all records in CIG from SKG
616.        END IF
617.        IF Strong Match is NOT found, THEN FIND Weak_Match
618.        IF Weak Match is found, THEN create Weak_CIG
619.              ELSE REMOVE all records in CIG from SKG
620.        END IF
621.  END LOOP
```

FIG. 14

Pseudo Code for Finding Weak Matches for a Record

```
622.  FOR EVERY Record in SKG
623.        IF (SKG record is from same database as records for which match is sought OR
624.              SKG record already is a Weak_Match record in a CIG OR
625.              SKG record is a Dependent_FIG OR
626.              SKG record is Non_Recurring AND records for which is sought are not, OR
627.              SKG record is Recurring AND records for which is sought are not)
628.        THEN
629.              GO TO END LOOP
630.        ELSE
631.              If recurring item OR Key_Date_Field match Exactly, THEN Weak_Match is found
632.        END IF
633.  END LOOP
```

FIG. 15

Pseudo Code for Finding Matches between Recurring items and Non_Unique ID Bearing Instances

```
650.  IF Instances' database does not have unique ID OR synchronizing from scratch THEN CONTINUE
651.        ELSE EXIT
652.  END IF
653.  FOR any Recurring_Master not in Instances database,
654.        Fan out Recurring_Master for Previous_Date_Range into Previous_Date_Array
655.        MARK all entry as Previous_Date_Range_Instance
656.        Fan out Current_Recurring_Master for Current Data Range into Current_Dates_Array
657.        MARK all entries as Current_Date_Range_Instance
658.        MARK records in Exclusion_List as EXCLUDED_Dates
659.        MERGE Exclusion_List, Previous_Date_Array and Current_Date_Array into
            Merged_Date_Array
660.        CREATE Slave_Date_Array
661.        FOR EVERY item in SKG of Recurring_Master
662.              IF Recurring item OR NOT Instances database record, THEN GO TO END LOOP
663.              IF Start_Date of SKG record Matches an Entry in Merged_Date_Array THEN STORE
                  in Slave_Array WORKSPACE record number of SKG record AND
                  Merged_Date_Array in Slave Array
664.        END LOOP
665.        FOR EVERY Unique Non_Date Hash of Slave_Array records
666.              FIND Slave_Array records with matching Non_Date Hash
667.              COUNT number of matches
668.        END LOOP
669.        FIND the largest number of match counts
670.        IF largest is less than 30% of number of unexcluded instances of Master Recurring, THEN
            EXIT
```

FIG. 16A

```
671.        IF Match equals one, THEN IF NOT exact match, THEN EXIT
672.        CREATE Homogenous_Instance_Group from the records which have the same Non_Date_Hash
            value as the largest match
673.        CREATE new record Synthetic_Master in WORKSPACE
674.        COPY Basic Repeat Pattern of Recurring_Master into Synthetic Master
675.        COPY Other values from 1st item of Homogeneous Instance Group into Synthetic Master
676.        CREATE Synthetic_Master Exclusion_List based on differences between Merged_Date_Array
            and Homogeneous_Instance_Group
677.        COMPUTE Hash values for Synthetic_Master
678.        ADD Synthetic_Master to CIG of Recurring_Master
679.        CREATE Synthetic_Master FIG from all Homogeneous_Instances_Group item
680.        FOR EVERY Homogeneous_Instances_Group_item,
681.              IF Weak_match in another CIG, THEN REMOVE from CIG AND FIND New WEAK
                  MATCH for that CIG
682.              REMOVE from its SKG
683.              MARK as Dependent_FIG
684.        END LOOP
685.        IF dates in Previous_Date_Array which are not in Current_Date_Array OR Vice_versa THEN
            MARK CIG Fan_Out_Creep Flag (for unload time)
686.  END LOOP
```

FIG. 16B

Pseudocode for Completing SKG Analysis

```
700.  IF A_database AND B_database are unique ID bearing DBs, THEN REMOVE ALL remaining H_items
      from SKGs
702.  END IF
703.  FOR ALL SKGs in WORKSPACE
704.        IF SKG is singleton, THEN GO TO END LOOP
705.        FOR ALL items in Current_SKG
706.              IF item is Weak_Match AND part of ID_based pair, THEN REMOVE from SKG
707.        END LOOP
708.        FOR ALL records in Current_SKG begining with H_Records
709.              Call Set CIG_Max_Size in Figure 18
710.              FIND Strong Match or Master/Instance Match between Non_ID bearing database
                  record and H_Records
711.              IF FOUND, THEN ADD to CIG
712.                    ELSE IF FIND Strong_Match in SKG between BA and B database records
                      THEN Attach records together as CIG END IF
713.              END IF
714.              IF CIG_Size = CIG_MAX_Size, THEN REMOVE ALL CIG members from SKG
715.        END LOOP
716.        IF CIG_Max_Size = 3, THEN
717.              FOR EVERY two record CIG in SKG,
718.                    FIND Weak_Match (Same Key_Date_Field and Same Recurrence Level)
719.                    IF Weak_Match item from opposing DB, THEN ADD to CIG
720.                    REMOVE records in CIG from SKG
721.              END LOOP
722.        END IF
723.        FOR EVERY SKG item
724.              FIND Weak_Match (Same Key_Date_Field and Same Recurrence Level)
725.              IF FOUND, THEN ADD to CIG and REMOVE from SKG
726.        END LOOP
727.  END LOOP
```

FIG. 17

Pseudocode for setting Maximum CIG Size for Every CIG analyzed in Fig. 17.

750. CIG_Max_Size = the number of non-unique ID bearing applications +1
751. If the CIG_Max_size = 1 and CIG is not a H_Record THEN CIG_MAX_Size = 2

FIG. 18

Pseudo Code for setting CIG types

```
800.  FOR EVERY CIG
801.      IF CIG Size is 1, THEN
802.          DETERMINE origin of the CIG record
803.          IF H_Record, THEN CIG_Type = 010
804.          IF B_Record, THEN CIG_Type = 001
805.          IF A_Record, THEN CIG_Type = 100
806.      END IF
807.      IF CIG Size is 2, THEN
808.          COMPARE the two CIG records
809.          IF two members are the same, THEN
810.              DETERMINE the origin of the CIG records
811.              IF B_Record and H_Record, THEN CIG_Type = 011
812.              IF A_Record and H_Record, THEN CIG_type = 110
813.              IF B_Record and A_Record, THEN CIG_type = 101
814.          END IF
815.          IF two records are different, THEN
816.              DETERMINE the origin of the CIG records
817.              IF B_Record and H_Record, THEN CIG_Type = 012
818.              IF A_Record and H_Record, THEN CIG_type = 210
819.              IF B_Record and A_Record, THEN CIG_type = 102
820.          END IF
```

FIG. 19A

```
821.            END IF
822.            IF CIG_Size = 3, THEN
823.                    COMPARE records
824.                    DETERMINE origins of records
825.                    IF ALL records are the same, THEN CIG_Type = 111
826.                    IF A_Record different from the other two and B_Record= H_Record,    THEN
                        CIG_Type = 211
827.                    IF B_Record different from the other two and A_Record= H_Record,    THEN
                        CIG_Type = 112
828.                    IF H_Record different from the other two and B_Record= A_Record,    THEN
                        CIG_Type = 212
829.                    IF ALL records are different, THEN CIG_Type = 213
830.            END IF
831.    END LOOP
```

FIG. 19B

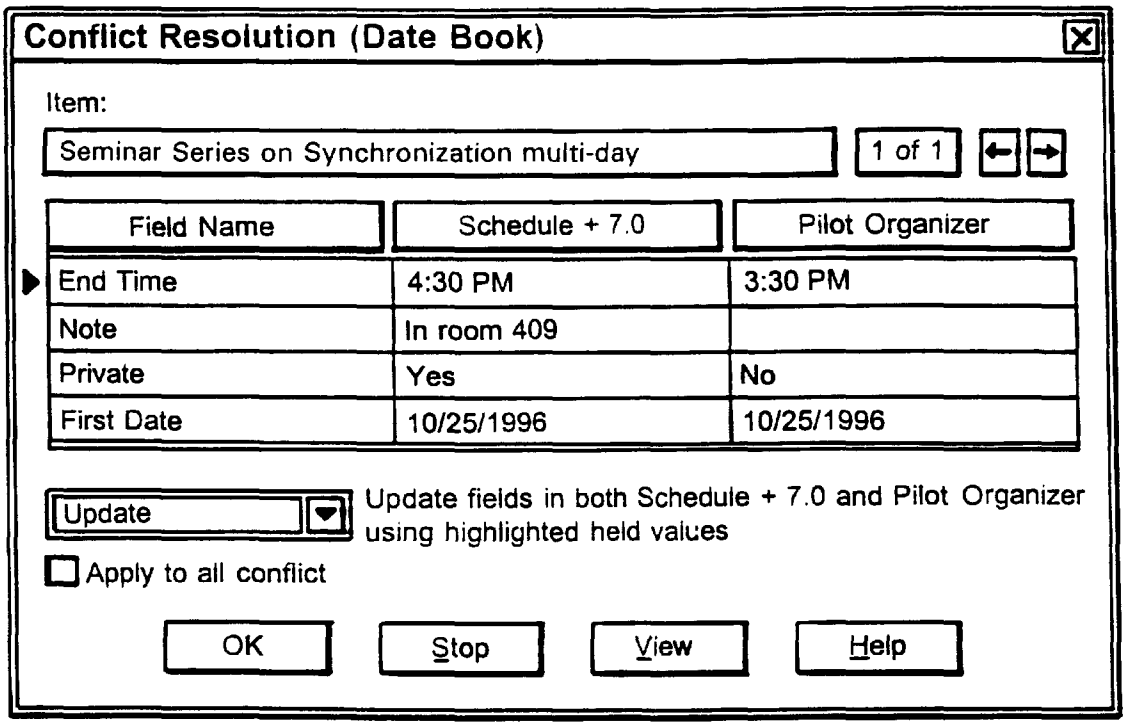

Conflict Resolution (Date Book)
Item:
Seminar Series on Synchronization multi-day
1 of 1
Field Name
Schedule + 7.0
Pilot Organizer
End Time
4:30 PM
3:30 PM
Note
In room 409
Private
Yes
No
First Date
10/25/1996
10/25/1996
Update
Update fields in both Schedule + 7.0 and Pilot Organizer using highlighted held values
Apply to all conflict
OK
Stop
View
Help

FIG. 20

Pseudocode for Merging Exclusion Lists

```
850.    FOR ALL Recurring Masters,
851.        IF CIG_Type is 102 and conflict is unresolved THEN GO TO END LOOP
{Changing CIG TYPE}
852.        COMPARE Exclusion_Lists of Current_CIG A and B records to determine Exclusion instances
            which appear in only one of the two records (i.e. One_Side_Only_Exclusion)
853.        IF None THEN do nothing
854.            ELSE IF One_side_only_Exclusion in A_Record but not in B THEN USE Table in
                FIG. 22 to Convert CIG_Type
855.            ELSE IF One_Side_Only_Exclusion in B record but not in A THEN USE Table in
                FIG. 23 to Convert CIG_Type
856.            ELSE IF One_Side_Only_Exclusion in both records, THEN USE Table in FIG. 24 to
                convert CIG_Type
857.        END IF
858.    END LOOP
```

FIG. 21

Old CIG + choice	new CIG	new Conflict Resolution Choice	Other Instructions & Comments
101	102	ADB Wins	
111	211		
112	132		Replace H_Record with a copy of the B_Record, plus the ADB Exclusion List
211	211		
212	213	ADB Wins	
132	132		Copy ADB ExclusionList into P-Item
102-Ig	102	Ignore	
102-SW	102	ADB Wins	
102-TW	132		Create H_Record by copying the B_Record, plus the ADB Exclusion List
213-Ig	213	ADB Wins, Excl Only	The Excl Only flag is set so that only the Exclusion List will be updated. Other BDB Fields will remain unchanged.
213-SW	213	ADB Wins	
213-TW	132		Replace P-Item with a copy of the B_Record, plus the ADB Exclusion List

(Ig for Ignore, SW for ADB Wins, or TW for BDB Wins).

FIG. 22

Old CIG + choice	new CIG	new Conflict Resolution Choice	Other Instructions & Comments
101	102	BDB Wins	
111	112		
112	112		
211	132		Replace P-Item with a copy of the A_Record, plus the BDB Exclusion List
212	213	BDB Wins	
132	132		Copy BDB ExclusionList into P-Item
102-Ig	102	Ignore	
102-SW	132		Create P-Item by copying A_Record, plus the BDB Exclusion List
102-TW	102	BDB Wins	
213-Ig	213	BDB Wins, Excl Only	The Excl Only flag is set so that only the Exclusion List will be updated. Other ADB Fields will remain unchanged.
213-SW	132		Replace P-Item with a copy of the A_Record, plus the BDB Exclusion List
213-TW	213	BDB Wins	

(Ig for Ignore, SW for ADB Wins, or TW for BDB Wins)

FIG. 23

Old CIG + choice	new CIG	new Conflict Resolution Choice	Other Instructions & Comments
101	132		Create P-Item by copying B_Record, plus the Merged Exclusion List
111	132		Copy Merged Exclusion List into P-Item.
112	132		Replace P-Item with a copy of the B_Record, plus the Merged Exclusion List
211	132		Replace P-Item with a copy of the A_Record, plus the Merged Exclusion List
212	132		Replace P-Item with a copy of the B_Record, plus the Merged Exclusion List
132	132		Copy Merged ExclusionList into P-Item
102-Ig	102	Ignore	
102-SW	132		Create P-Item by copying A_Record, plus the Merged Exclusion List
102-TW	132		Create P-Item by copying B_Record, plus the Merged Exclusion List
213-Ig	132	Excl Only	Copy Merged ExclusionList into P-Item. The Excl Only flag is set so that only the Exclusion List will be updated. Other ADB and BDB Fields will remain unchanged.
213-SW	132		Replace P-Item with a copy of the A_Record, plus the Merged Exclusion List
213-TW	132		Replace P-Item with a copy of the B_Record, plus the Merged Exclusion List

(Ig for Ignore, SW for ADB Wins, or TW for BDB Wins)

FIG. 24

Pseudo Code for Unloading Records from WORKSPACE to a database for non_rebuild_all database

```
899.  FOR all Recurring Masters which require Fanning and Outcome is UPDATE or DELETE, call
      Synchronizer Function Fanning for Unloading, Fig.27
900.  COUNT RECORDS to be Unloaded by examining all CIGs
901.   FOR EVERY RECORD to be Unloaded
{DETERMINE OUTCOME}
902.        IF MARKED GARBAGE, THEN SKIP
903.        IF BYSTANDER AND NOT History File Unload, THEN SKIP
904.        IF WRONG_SUBTYPE AND NOT Rebuild_All Translator, THEN SKIP
905.        IF Recurring_Master THEN IF Fanned for the database THEN UNLOAD Instances when
            unloading END IF
906.              ELSE UNLOAD Recurring Master when unloading
907.        END IF
908.        LOOK UP Outcome_Sync (i.e., Unload Instructions) in Fig. 26 Table based on CIG_TYPE]
909.        IF Date Range Limited Database and Date_Range_Option = LENIENT, THEN
910.              IF RECORD is Out of Current_Date_Range AND Outcome is not DELETE, THEN
                  SKIP Record
911.        ELSE IF Date Range Limited Database and Date_Range_Option = STERN, THEN
912.              IF RECORD is Out of Current_Date_Range, THEN Outcome=DELETE
913.        END IF
914.        IF Outcome = DELETE, THEN
915.              Get Info Required for this database to DELETE RECORD
916.              (may include unique ID, Record ID, or the original values of one or more key fields, to
                  look up record so that it can be deleted)
917.              DELETE Record
918.              SEND Synchronizer SUCCESS/FAILURE FLAG
919.        END IF
```

FIG. 25A

```
920.        IF Outcome = ADD, THEN
921.              GET Current values of all Fields, from Synchronizer
                  (Synchronizer maps for A database based on B→A, in response to each request)
922.              CREATE new RECORD in DB
923.              IF Unique_ID DB, THEN GET Unique_ID
924.              SEND to Synchronizer (Success FLAG with any Unique_ID) OR (Failure Flag)
925.              Synchronizer: Store Unique_ID in WORKSPACE
926.        END IF
927.        IF Outcome is UPDATE THEN GET Current values to be unloaded and original values loaded
            from database from Synchronizer
928.              COMPARE and DETERMINE which Field to be updated
929.              UPDATE fields in the record to be updated
930.              SEND to Synchronizer (Success flag AND Unique_ID) OR (Failure Flag)
931.              Synchronizer:  STORE Unique_ID in WORKSPACE
932.        END IF
933.  END LOOP
```

FIG. 25B

```
// Original  Current
//  Item      Item      Outcome
// -------   -------   -------
{

//--- TIFCIG_001 - 1  (0) // item is present in BDB only

    B,          B,          oLEAVE_ALONE,   // unloading to BDB
    B,          B,          oADD,           // unloading to ADB
  B,      B,     oSAVE,      // unloading to History File

//--- CIG_100 - 1  (1) // item is present in ADB only

  A_      A_      oADD,        // unloading to BDB
  A_      A_      oLEAVE_ALONE, // unloading to ADB
  A_      A_      oSAVE,       // unloading to History File

//--- CIG_101 - 1  (2) // item is identical in ADB and BDB

  B_      B_      oLEAVE_ALONE, // unloading to BDB
  A_      A_      oLEAVE_ALONE, // unloading to ADB
  A_      B_      oSAVE,       // unloading to History File

//--- CIG_102 - 1  (3) // NEW ADB ITEM < > NEW BDB ITEM
                   // (the BDB WINS outcome is shown here)

  B_      B_      oLEAVE_ALONE, // unloading to BDB
  A_      B_      oUPDATE,     // unloading to ADB
  A_      B_      oSAVE,       // unloading to History File

//--- CIG_111 - 1  (4) // item is unchanged across the board

  B_      B_      oLEAVE_ALONE, // unloading to BDB
  A_      A_      oLEAVE_ALONE, // unloading to ADB
  H_      H_      oSAVE,       // unloading to History File

//--- CIG_112 - 1  (5) // item CHANGED in BDB since last sync

  B_      B_      oLEAVE_ALONE, // unloading to BDB
  A_      B_      oUPDATE,     // unloading to ADB
  H_      B_      oSAVE,       // unloading to History File

//--- CIG_110 - 1  (6) // item DELETED from BDB since last sync

  H_      H_      oLEAVE_DELETED, // unloading to BDB
  A_      A_      oDELETE,     // unloading to ADB
  H_      H_      oDISCARD,    // unloading to History File

//--- CIG_211 - 1  (7) // item CHANGED in ADB since last sync

  B_      A_      oUPDATE,     // unloading to BDB
```

FIG. 26A

```
A_      A_      oLEAVE_ALONE,  // unloading to ADB
H_      A_      oSAVE,         // unloading to History File

//--- CIG_212 - 1  (8) // item CHANGED IDENTICALLY in Src & BDB

B_      B_      oLEAVE_ALONE,  // unloading to BDB
A_      A_      oLEAVE_ALONE,  // unloading to ADB
H_      A_      oSAVE,         // unloading to History File

//--- CIG_213 - 1  (9) // item CHANGED DIFFERENTLY in Src & BDB
                   // (the BDB WINS outcome is shown here)

B_      B_      oLEAVE_ALONE,  // unloading to BDB
A_      B_      oUPDATE,       // unloading to ADB
H_      B_      oSAVE,         // unloading to History File

//--- CIG_210 - 1  (10) // CHANGED in ADB, DELETED from BDB

A_      A_      oADD,          // unloading to BDB
A_      A_      oLEAVE_ALONE,  // unloading to ADB
H_      A_      oSAVE,         // unloading to History File

//--- CIG_011 - 1  (11) // item DELETED from ADB since last sync

B_      B_      oDELETE,       // unloading to BDB
H_      H_      oLEAVE_DELETED, // unloading to ADB
H_      H_      oDISCARD,      // unloading to History File

//--- CIG_012 - 1  (12) // DELETED from ADB, CHANGED in BDB

B_      B_      oLEAVE_ALONE,  // unloading to BDB
B_      B_      oADD,          // unloading to ADB
H_      B_      oSAVE,         // unloading to History File

//--- CIG_010 - 1  (13) // item DELETED from both ADB & BDB

H_      H_      oLEAVE_DELETED, // unloading to BDB
H_      H_      oLEAVE_DELETED, // unloading to ADB
H_      H_      oDISCARD,      // unloading to History File

//--- CIG_132 - 1  (14) // 102 conflict resolved interactively
                   // to a "compromise" value stored in P-item
                   // outcome is always UPDATE BOTH

B_      H_      oUPDATE,       // unloading to BDB
A_      H_      oUPDATE,       // unloading to ADB
A_      H_      oSAVE,         // unloading to History File

//--- CIG_13F - 1  (15) // 132 UPDATE-BOTH
                   // which has been Fanned To BDB

B_      B_      oDELETE,       // unloading to BDB
A_      H_      oUPDATE,       // unloading to ADB
A_      H_      oSAVE          // unloading to History File
```

FIG. 26B

```
  // Note that we delete the recurring master on the BDB Side;
  // fanned instances take its place.

};
```

The table entries above for CIG_102 and CIG_213 are only relevant when the Conflict Resolution Option is set to BDB WINS. If the Conflict Resolution Option is set to IGNORE or ADB WINS then those table entries are adjusted accordingly. For IGNORE we use the following table entries:

```
// Original  Current
// Item      Item     Outcome
// -------   -------  -------
//--- _CIG_TYPE_102   // NEW ADB ITEM < > NEW BDB ITEM

    B_        B_       oLEAVE_ALONE,  // unloading to BDB
    A_        A_       oLEAVE_ALONE,  // unloading to ADB
    B_        B_       oDISCARD,      // unloading to History File

//--- _CIG_TYPE_213   // item CHANGED DIFFERENTLY in Src & BDB

    B_        B_       oLEAVE_ALONE,  // unloading to BDB
    A_        A_       oLEAVE_ALONE,  // unloading to ADB
    H_        H_       oSAVE,         // unloading to History File
```

And for ADB WINS we use the following table entries:

```
// Original  Current
// Item      Item     Outcome
// -------   -------  -------

//--- _CIG_TYPE_102   // NEW ADB ITEM < > NEW BDB ITEM

    B_        A_       oUPDATE,       // unloading to BDB
    A_        A_       oLEAVE_ALONE,  // unloading to ADB
    B_        A_       oSAVE,         // unloading to History File

//--- _CIG_TYPE_213   // item CHANGED DIFFERENTLY in Src & BDB

    B_        A_       oUPDATE,       // unloading to BDB
    A_        A_       oLEAVE_ALONE,  // unloading to ADB
    H_        A_       oSAVE,         // unloading to History File
```

When the NOY option is in effect, CIG-specific conflict outcomes are recorded in the CIG members' flag bits. When this is the case the following lookup table is used:

```
static unsigned char TableAfterILCR [_SYNC_OUTCOME_COUNT]
                                    [AFTER_ILCR_CIG_TYPE_COUNT]
                                    [SYNC_UNLOAD_PHASE_COUNT]
                                    [3] =
// Original  Current
// Item      Item     Outcome
// -------   -------  -------
{
```

FIG. 26C

```
//-------------------- Entries for _OUTCOME_SYNC_BDB_WINS

  //--- _CIG_TYPE_102  // NEW ADB ITEM < > NEW BDB ITEM

     B_      B_      oLEAVE_ALONE,  // unloading to BDB
     A_      B_      oUPDATE,       // unloading to ADB
     A_      B_      oSAVE,         // unloading to History File

  //--- _CIG_TYPE_213  // item CHANGED DIFFERENTLY in Src & BDB

     B_      B_      oLEAVE_ALONE,  // unloading to BDB
     A_      B_      oUPDATE,       // unloading to ADB
     H_      B_      oSAVE,         // unloading to History File

//-------------------- Entries for _OUTCOME_SYNC_ADB_WINS

  //--- _CIG_TYPE_102  // NEW ADB ITEM < > NEW BDB ITEM

     B_      A_      oUPDATE,       // unloading to BDB
     A_      A_      oLEAVE_ALONE,  // unloading to ADB
     B_      A_      oSAVE,         // unloading to History File

  //--- _CIG_TYPE_213  // item CHANGED DIFFERENTLY in Src & BDB

     B_      A_      oUPDATE,       // unloading to BDB
     A_      A_      oLEAVE_ALONE,  // unloading to ADB
     H_      A_      oSAVE,         // unloading to History File

//-------------------- Entries for IGNORE (LEAVE UNRESOLVED)

  //--- _CIG_TYPE_102  // NEW ADB ITEM < > NEW BDB ITEM

     B_      B_      oLEAVE_ALONE,  // unloading to BDB
     A_      A_      oLEAVE_ALONE,  // unloading to ADB
     B_      B_      oDISCARD,      // unloading to History File

  //--- _CIG_TYPE_213  // item CHANGED DIFFERENTLY in Src & BDB

     B_      B_      oLEAVE_ALONE,  // unloading to BDB
     A_      A_      oLEAVE_ALONE,  // unloading to ADB
     H_      H_      oSAVE          // unloading to History File

}; //---- TableAfterILCR
```

FIG. 26D

```
FANNING Recurring_Items for Unloading (for A DB)

    Fan Pattern for paper Date Range (Fig. XX)

950.  IF Outcome is UPDATE, THEN
951.      IF (CIG A_Record was a Recurring Master but now to be fanned and CIG B_Record is a
          Recurring Master) THEN IF CIG_Type = 132 THEN CIG_Type = 13F
952.          GOTO Fanning For ADD
953.          ELSE
954.              SET A_Record CIG_Type to 100
955.              SET B_Record CIG_Type to 001
956.              SET H_Record CIG_Type to 010
957.              MARK A_Record with DELETE_ME Flag
958.              GOTO Fanning for Add
959.          END IF
960.      END IF
961.      IF (CIG A_Records were fanned previously and Fanned now) AND (CIG B_record recurring),
          THEN
962.          FOR ALL A items in Synthetic Master FIG
963.              STORE Start_Date in Date_Array_Temporary
964.          END LOOP
965.          Fan_Out_Recurring_Pattern of B Master
966.          COMPARE Date_Array_Temp with Fan_Out_Date_Array
967.          MARK Dates which NOT IN Fan_Out_Date Array with DELETE_Me Flag
968.          IF Date NOT IN Date_Array_Temp, THEN
969.              CREATE WORK SPACE Record by Copy Recurring_Master but Omit Rep
                  Basic, Rep Excl, Unique ID Field
970.              SET Start_Date, End_Date, Alarm_Date to values for Current Instance
971.              Compute Hash
972.              MARK Fanned_For_A
973.          END IF
```

FIG. 27A

```
974.                IF Date in Date_Array_Temp AND Fan_Out_Date_Array THEN
975.                        Compare Non_Date Hash to Synthetic Master Non_Date_Hash
976.                        IF Same, THEN MARK Leave_Alone
977.                        ELSE MARK UPDATE END IF
978.                END IF
979.        END IF
980.        IF (A_Record Recurring previously and to be Fanned now) AND (CIG B_Record is Instances)
            THEN
981.                MARK CIG items as Garbage
982.                MARK FIG items of CIG H_record as Garbage
983.                MAKE FIG items of CIG B_record singletons
984.        END IF
985.  ELSE [Fanning For Add]
986.        Fan out Recurrence Pattern
987.        For each Date in Fan_Out_Date_Array
988.                COPY Master item into new WORKSPACE Record except Omit Rep_Basic,
                    Rep_Exclusion, and Unique ID
989.                Use Date for Start Date and End Date
990.                Set Alarm Date, if necessary
991.                Compute Hash Values
992.                Attach to Recurring_Master FIG
993.                Set Fanned_for_A Flag
994.        END LOOP
995.  END IF
```

FIG. 27B

Pseudocode for Unloading History FILE

```
1000.  ERASE previous History File and CREATE new one
1001.  FOR EVERY CIG in WORKSPACE
1002.      Look up in Fig. 26 Table based on CIG_Type AND DETERMINE whether should be unloaded
           into the History File
1003.      IF NO THEN GOTO END LOOP
1004.      IF Exclusion_List_Only Flag is set when merging of Exclusion_List THEN REPLACE History
           RECORD Exclusion_List with new Merged Exclusion_List
1005.      Clear all Flag bits except for Recurring_Record flag
1006.      SET origin flag to History_Record
1007.      Clear FIG, SKG and CIG words
1008.      STORE Applicable Unique IDs
1009.      IF Recurring item, THEN STORE ALL ID_Bearing FIG records AND SET their FIG in
           History_File to keep them together
1010.      STORE Record in History File
1011.      IF current record is a recurring master for an ID-bearing FIG THEN STORE FIG Records(i.e.
           all Fanned Instances) in the History File, with the FIG linkage words set in the History File to
           hold the FIG together.
1012.  END LOOP
1013   STORE Field Lists, Application Names, Database Names, Current Date Range,
```

FIG. 28

	How Item is stored in Other Database	How stored in Unloader's Database Before Fanning For Update	How stored in Unloader's Database After Fanning For Update
1	Master	Master	Instances
2	Master	Instances	Instances
3	Instances	Master	Instances

FIG. 29

1050. Verify History File
1051. If verified, Then Proceed as Fast Synch
1052. If not, Then Proceed as Synchronization from Scratch load all record in databasse

1053. If Fast Synch
1054. LOAD records into the Workspace. Map if necessary
1055. Sanitize Records not marked as Deletion
1056. Orientation analysis (Fig. 11).
1057. For each H_Record, analyze the CIG that the H_Record belongs to.
1058. IF the H_Record's CIG contains no Record from the Fast Synchronization database, THEN CLONE the H-Item, label it a Fast Synchronization Record, and add it to the H_Record's CIG.
1059. If the H_Record's CIG contains a Fast Synchronization record that is marked as a a Deletion, it is now removed from the CIG.
1060. If the H_Record's CIG contains a non-Delete Fast Synchronization Record, then do nothing.
1061. END LOOP

FIG. 30

1150. Verify History File
1151. If verified, Then Proceed as Fast Synch
1152. If not, Then Proceed as Synchronization from Scratch

1153. IF synchronization from scratch
1154. IF record outside of current_date_range THEN MARK record as out-of-range

1155. If Fast Synch
1156. Load History File into Workspace
1157. MARK History File records outside of previous_date_range as Bystander
1158. Load All Fast Synchronization Records into the Workspace; mapped if necessary.
1159. SANITIZE Records which are not DELETEs
1160. Orientation analysis (Fig. 11).
1161. If Added Fast Synchronization record is out of current date range THEN MARK Out-Of_Range
1162. If Changed or deleted Fast Synchronization record in a CIG with Bystander H_Record, MARK the Bystander record as Garbage

FIG. 31A

```
1163.   For each H_Record, analyze the CIG that the H_Record belongs to.
1164.       If the H_Record's CIG contains no Record from the Fast Synchronization database,
            then make a clone of the H-Item, label it a Fast Synchronization Record, and adding it
            to the H_Record's CIG.
1165.       If H_Record is not a Bystander, THEN Make a clone of H_Record, mark as Fast
            Synchronization record, and Add to CIG
1166.       IF H_Record is Bystander THEN
1167.           IF outside of Current date range THEN Do Nothing
1168.           ELSE {Within Current Date Range}
1169.               Mark H_Record as Garbage, Clone H_Record and Mark it as from
                    Fast Synchronization database
1170.           END IF
1171.       END IF
1172.       If the H_Record's CIG contains a Fast Synchronization record that is marked as a
            deletion, it is now removed from the CIG.
1173.       If the H_Record's CIG contains a non-deletion Fast Synchronization Record, then do
            nothing.
1174.       Any Fast Synchronization records which are not joined to any H_Record's CIG
            represent additions and no transformation is required.
1175.   END LOOP
```

FIG. 31B

SYNCHRONIZATION OF DATABASES WITH DATE RANGE

An appendix forms part of this application. The appendix, which includes a source code listing relating to an embodiment of the invention, includes 691 frames on 8 microfiche.

BACKGROUND OF THE INVENTION

This invention relates to synchronizing incompatible databases.

Databases are collections of data entries which are organized, stored, and manipulated in a manner specified by applications known as database managers (hereinafter also referred to as "Applications"). The manner in which database entries are organized in a database is known as the data structure. There are generally two types of database managers. First are general purpose database managers in which the user determines (usually at the outset, but subject to future revisions) what the data structure is. These Applications often have their own programming language and provide great flexibility to the user. Second are special purpose database managers that are specifically designed to create and manage a database having a preset data structure. Examples of these special purpose database managers are various scheduling, diary, and contact manager Applications for desktop and handheld computers. Database managers organize the information in a database into records, with each record made up of fields. Fields and records of a database may have many different characteristics depending on the database manager's purpose and utility.

Databases can be said to be incompatible with one another when the data structure of one is not the same as the data structure of another, even though some of the content of the records is substantially the same. For example, one database may store names and addresses in the following fields: FIRST_NAME, LAST_NAME, and ADDRESS. Another database may, however, store the same information with the following structure: NAME, STREET_NO., STREET_NAME, CITY_STATE, and ZIP. Although the content of the records is intended to contain the same kind of information, the organization of that information is completely different.

It is often the case that users of incompatible databases want to be able to synchronize the databases. For example, in the context of scheduling and contact manager Applications, a person might use one Application on the desktop computer at work while another on his handheld computer or his laptop computer at home. It is desirable for many of these users to be able to synchronize the entries on one with entries on another. However, the incompatibility of the two databases creates many problems that need to be solved for successful synchronization. The U.S. patent and copending patent application of the assignee hereof, Puma Technology, Inc. of San Jose, Calif. (U.S. Pat. No. 5,392,390; U.S. application, Ser. No. 08/371,194, filed on Jan. 11, 1995, incorporated by reference herein) show two methods for synchronizing incompatible databases and solving some of the problems arising from incompatibility of databases. However, other problems remain.

Some database Application run on computer systems with very limited storage capacity, such as handheld computers. It is often desirable to synchronize the databases on these devices with databases on larger computers such as desktop computers which have much higher storage capacity. However, a straight synchronization between the Applications on the two devices may result in storage capacity of the smaller devices being mostly consumed with the records from the larger device, rendering the smaller device inoperable.

SUMMARY OF THE INVENTION

The invention solves the difficulty of synchronizing databases in which events are maintained across different date ranges. A date range is set for which synchronization will take place. Records falling outside of the date range are not synchronized. The date range of the prior synchronization is stored, and a current synchronization is performed across the combination of the current and prior date ranges. The problems of synchronization software attempting to fill a smaller capacity device with events across a wide date range that can only practically be stored on a larger capacity device are avoided.

The invention features a computer implemented method of synchronizing at least a first and a second database each containing dated records such as events, wherein the records of the first database extend across a narrow date range narrower than the date range of the records of the second database. A prior synchronization is performed across a prior date range set using the date of the prior synchronization and the narrow date range. The date range of the prior synchronization is stored, along with the history file containing information representative of the content of the databases following the prior synchronization. When a current synchronization is performed, it is performed across a date range that combines the prior date range with a current date range set using the date of the current synchronization and the narrow date range.

The invention may be implemented in hardware or software, or a combination of both. Preferably, the technique is implemented in computer programs executing on programmable computers that each include a processor, a storage medium readable by the processor (including volatile and non-volatile memory and/or storage elements), at least one input device, and at least one output device. Program code is applied to data entered using the input device to perform the functions described above and to generate output information. The output information is applied to one or more output devices.

Each program is preferably implemented in a high level procedural or object oriented programming language to communicate with a computer system. However, the programs can be implemented in assembly or machine language, if desired. In any case, the language may be a compiled or interpreted language.

Each such computer program is preferably stored on a storage medium or device (e.g., ROM or magnetic diskette) that is readable by a general or special purpose programmable computer for configuring and operating the computer when the storage medium or device is read by the computer to perform the procedures described in this document. The system may also be considered to be implemented as a computer-readable storage medium, configured with a computer program, where the storage medium so configured causes a computer to operate in a specific and predefined manner.

Other features and advantages of the invention will become apparent from the following description of preferred embodiments, including the drawings, and from the claims.

BRIEF DESCRIPTION OF THE DRAWING

FIG. **1** is a schematic drawing of the various modules constituting the preferred embodiment.

FIG. **2** is a representation of the Workspace data array.

FIG. **3** is the pseudocode for the Translation Engine Control Module.

FIG. **4** is the pseudocode for generating the parameter Table.

FIG. **5** is the pseudocode for fanning a recurring record.

FIG. **6** is the pseudocode for the Synchronizer loading the History File.

FIG. **7** is the pseudocode for matching key fields (Key_Field_Match).

FIG. **8** is the pseudocode for loading records of B_Database into Workspace.

FIG. **9** is the pseudocode for A_Sanitization of B_Database records in Workspace.

FIG. **10** is the Pseudocode for a specific example of a rule of data value used for sanitization.

FIG. **11** is the pseudocode for orientation analysis.

FIG. **12** is the pseudocode for Conflict Analysis And Resolution (CAAR).

FIG. **13** is the pseudocode for analyzing unique ID bearing Fanned Instance Groups (FIGs).

FIG. **14** is the pseudocode for expanding CIGs created from unique ID bearing records.

FIG. **15** is the pseudocode for finding weak matches for a record.

FIG. **16** is the pseudocode for finding matches between recurring items and non_unique ID bearing instances.

FIG. **17** is the pseudocode for completing Same Key Group (SKG) analysis.

FIG. **18** is the pseudocode for setting the Maximum_CIG_Size for every CIG analyzed in FIG. **17**.

FIG. **19** is the pseudocode for setting CIG_Types.

FIG. **20** is the User Interface for conflict resolution when the Notify option is selected.

FIG. **21** is the pseudocode for merging exclusion lists.

FIG. **22** is a look up table used by the function in FIG. **21**.

FIG. **23** is a look up table used by the function in FIG. **21**.

FIG. **24** is a look up table used by the function in FIG. **21**.

FIG. **25** is a pseudocode for unloading records from Workspace to a non-rebuild-all database.

FIG. **26** is the look up table for determining unloading outcome results.

FIG. **27** is the pseudocode for fanning recurring records of A-Database for unloading.

FIG. **28** is the pseudocode for unloading the History File.

FIG. **29** is a table showing cases for fanning Recurring Masters into own database.

FIG. **30** is the pseudocode for loading records by a fast synchronization Translator.

FIG. **31** is the pseudocoe for loadin records by a fast synchronization Translator.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

FIG. **1** shows the relationship between the various modules of the preferred embodiment. Translation Engine **1** comprises Control Module **2** and Parameters Table Generator **3**. Control Module **2** is responsible for controlling the synchronizing process by instructing various modules to perform specific tasks on the records of the two databases being synchronized. The steps taken by this module are demonstrated in FIG. **3**. The Parameters Table Generator **3** is responsible for creating a Parameter_Table **4** which is used by all other modules for synchronizing the databases. Details of the Parameter_Table are described in more detail below. The Synchronizer **15** has primary responsibility for carrying out the core synchronizing functions. It is a table-driven code which is capable of synchronizing various types of databases whose characteristics are provided in the Parameter_Table **4**. The Synchronizer creates and uses the Workspace **16**, which is a temporary data array used during the synchronization process.

A Translator **5** (A_Translator) is assigned to the A_database **13** and another Translator **9** (B_Translator) to the B_database **14**. Each of the database Translators **5** and **9** comprises three modules: Reader modules **6** and **10** (A_Reader and B_Reader), which read the data from the databases **13** and **14**; Unloader modules **8** and **12** (A_Unloader and B_Unloader), which analyze and unload records from the Workspace into the databases **13** and **14**; and Sanitizing modules **7** and **11** (A_Sanitizer and B_Sanitizer), which analyze the records of the other database loaded into the Workspace and modify them according to rules of data value of its own database. In the preferred embodiment, the modules of the A_Translator **5** are designed specifically for interacting with the A_database **13** and the A_Application **17**. Their design is specifically based on the record and field structures and the rules of data value imposed on them by the A_Application, the Application Program Interface (API) requirements and limitations of the A_Application and other characteristics of A_Database and A_Application. The same is true of the modules of B_Translator **9**. These Translators are not able to interact with any other databases or Applications. They are only aware of the characteristics of the database and the Application for which they have been designed. Therefore, in the preferred embodiment, when the user chooses two Applications for synchronization, the Translation Engine chooses the two Translators which are able to interact with those Applications. In an alternate embodiment, the translator can be designed as a table-driven code, where a general Translator is able to interact with a variety of Applications and databases based on the parameters supplied by the Translation Engine **1**.

Referring to FIGS. **1**, **2** and **3**, the synchronization process is as follows. The Parameter_Table **4** is generated by the Parameter Table Generator **3**. The Synchronizer **15** then creates the Workspace **16** data array and loads the History File **19** into the Workspace **16**. The B_Reader module **11** of the B_Translator reads the B_database records and sends them to the Synchronizer for writing into the Workspace. Following the loading of B_Database records, the A_Sanitizer module **8** of the A_Translator **5** sanitizes the B_Records in the Workspace. The A_Reader module **7** of the A_Translator **5** then reads the A_Database records and sends them to the Synchronizer **16** for writing into the Workspace. The B_Sanitizer module **12** of the B_Translator **9** then sanitizes the A_Records in the Workspace. The Synchronizer then performs the Conflict Analysis and Resolution (CAAR) on the records in Workspace. At the end of this analysis the user is asked whether he/she would like to proceed with updating the A_ and B_databases. If so, the B_Unloader module of the B_Translator unloads

the appropriate records into the B_database. The A_Unloader module **6** then performs the same task for the A_Database. Finally, the Synchronizer creates a new History File **19**.

FIG. **3** is the pseudocode for the preferred embodiment of the Control Module **2** of the Translation Engine **1**. Control Module **2** first instructs the Parameter Table Generator **3** of the Translation Engine **1** to create the Parameter_Table (Step **100**). FIG. **4** is the pseudocode for the preferred embodiment of the Parameter Table Generator module **3**. The user is first asked to choose whether to use a previously chosen and stored set of preferences or to enter a new set of preferences (Step **150**). Steps **151–165** show the steps in which the user inputs his/her new preferences. In step **152**, the user chooses whether to perform a synchronization from scratch or an incremental synchronization. In a synchronization from scratch, synchronization is performed as if this was the first time the two databases were being synchronized. In an incremental synchronization, the History File from the previous file is used to assist with synchronization. The user will likely choose incremental synchronization if there has been a prior synchronization, but the user may choose to synchronize from scratch where the user would like to start with a clean slate (perhaps due to significant change in the nature of the data in the databases). The user then selects the two Applications and related databases (A_Database and B_Database) to be synchronized (step **153**). The user then chooses (step **154**) whether the Synchronizer should use the default field mapping for those two databases during synchronization or the user will modify the field mapping. Field mapping is generally described in U.S. Pat. No. 5,392,390 (incorporated by reference). In accordance with the user's preferences, the Parameter Table Generator then stores the appropriate A_Database to B_Database fields map (A→B_Map) and B_Database to A_Database fields map (B→A_Map) in the Parameter_Table (Steps **155–158** and **159–163**, accordingly).

If in step **150** the user selected to use previously chosen and stored set of preferences (steps **166–171**), those preferences are loaded and stored in the Parameter_Table (steps **169–170**).

In case of date bearing records such as appointments and ToDo lists, the user enters the date range for which the user wants the records to be synchronized (step **172**). The preferred embodiment allows the user to use relative date ranges (Automatic_Date_Range) (substeps **171**(*a*) and (*b*)). For example, the user can select the date range to be 30 days into the past from today's date and 60 days into the future from today's date. The Parameter Table Generator **3** then calculates and stores in the Parameter_Table the Start_Current_Date_Range and End_Current_Date_Range values, the two variables indicating the starting point and the ending point of the date range for the current synchronization session (step **173–174**).

In steps **174** and **175**, various parameters identifying the characteristics of the A_Database and Application and B_Database and Application are loaded from a database (not shown) holding such data for different Applications. These are in turn stored in the Parameter Table. One of the sets of parameters loaded and stored in the Parameter_Table is the Field_List for the two databases. The Field_List_A and Field_List_B contain the following information about each field in the data structure of the two databases:

1. Field name.
2. Field Type.
3. Field Limitations.
4. No_Reconcile Flag.
6. Key_Field Flag.
7. Mapped_Field Flag.

Field name is the name given to the field which the Translator for this Application uses. This name may also be the name used by the Application. Field Type identifies to the Synchronizer **15** the nature of the data in a field, e.g., Data, Time, Boolean, Text, Number, or Binary. The Field Name does not supply this information to the Synchronizer. Field Limitations identifies the various limitations the database manager imposes on the contents of a field. These limitations include: maximum length of text fields, whether the text field must be in upper-case, range of permissible values (for example, in ToDo records priority field, the range of permissible values may be limited from 1 to 4), and whether a single line or multiple line field.

No_Reconcile flag indicates whether a field is a No_Reconcile field, meaning that it will not be used to match records nor will it be synchronized although it will be mapped and possibly used in synchronization. Almost all fields will not be designated as No Reconcile. However, sometimes it is necessary to do so. Key_Field flag indicates that a field should be considered as a key field by the Synchronizer **15**.

Key fields are used by the Synchronizer in various stages of synchronization as will be discussed in detail below. The decision of identifying certain fields as key is based on examining the various Applications to be synchronized, their data structure, and the purpose for which the database is used. Such examination reveals which fields would best function as key fields for synchronization. For example, for an address book database, the lastname, firstname, and company name field may be chosen as key fields. For Appointments, the date field and the description field may be chosen as key fields.

Mapped_Field flag indicates whether a field is mapped at all. The Synchronizer uses this flag to determine whether it should use the A→B_Map or B→A_Map to map this field. Unlike a No_Reconcile field, an unmapped field will not be carried along through the synchronization.

Another set of parameters in the Parameter_Table identify the Translator Modules **13**, **14** for the two Applications which the user has selected. Because each Application is assigned its own Translator, it is necessary to identify to the Command Module and the Synchronizer which Translators should be used.

In step **102** of FIG. **1**, the Translation Engine instructs the Synchronizer to load the History File. History File is the file which was saved at the end of last synchronization. It contains the history of the previous synchronization which is necessary for use with the current synchronization in case of Incremental Synchronization. Records from the A_Database and B_Database are analyzed against the records of the history file to determine the changes, additions, and deletions in each of two databases since last synchronization and whether additions, deletions, or updates need to be done to the records of the databases. Referring to FIG. **5**, in steps **200–201**, the Synchronizer finds the appropriate History file to be loaded. If Synchronization_from_Scratch flag is set, the History File is deleted (step **203**). If no History File is found, the synchronization will proceed as if it was a synchronization from scratch (step **204**). If the Field Lists stored in the History File are not the same as the current Field Lists in the Parameter_Table, or the mapping information is not the same, the synchronization will proceed as synchronization from scratch because the differences indicate that the History File records will not properly match the database records (steps **206–209**).

In step **210**, the Synchronizer uses the Field_List for database B to create the Workspace **16**. It is a large record array which the Synchronizer uses during synchronization. Referring to FIG. **2**, Workspace **16** consist of two sections. First, the Synchronizer uses the Field List for the B_Database to make a record array **21** which has all the characteristics of the B_Database record structure. In addition, in each record in the Workspace, certain internal fields are added. One field is _subtype containing Origin Tags. Two other fields, called Rep_Basic and Rep_Excl, are included for all Appointment and ToDo Sections. The Rep_Basic field gives a full description of the recurrence pattern of a recurring record. It includes the following parameters:

1. Basic_Repeat_Type
2. Frequency
3. StopDate
4. other parameters
5. Rep_Excl

Basic_Repeat_Type contains the variable which indicates whether the recurring record is a daily, weekly, monthly (same date each month), monthly by position (e.g., 3rd Friday of each month), yearly (e.g., July 4th each year), yearly by Position (e.g., 3rd Friday of September each year), quarterly, etc. This variable is set to No_Repeat for non-recurring records.

Frequency indicates whether the pattern is, for example, for every week, every other week, etc. StartDate and StopDate show the first date and last date in the pattern. Some other parameters in the Rep_Basic include, for example, a list of days to be included for the pattern (e.g. I plan to hold a weekly staff meeting every Thursday starting Nov. 15, 1997.)

Rep_Excl is the exclusion list. It is a list of dates which at some point belonged to the recurring record, but have since been deleted or modified and no longer are an event represented by the recurring record.

Since some databases do not provide for recurring types of records, the synchronization process sometimes must create single records for each of the instances of a recurring record for those databases. For example, for a recurring lunch every Thursday, the synchronization must produce a single record for each Thursday in such a database. This is accomplished by the process of fanning which uses Rep_Basic. Each of those instances is called a fanned instance. FIG. **6** sets out the preferred embodiment of the process of fanning a record.

Fanning of recurring records also takes into account another set of considerations regarding date range limitations and usefulness of instances to the user.

First, fanning is limited to the applicable date range. Second, the number of fanned instances is limited. When synchronizing Databases A and B, the preferred embodiment permits different sets of limits on fanned instances to be established for each Database. This, for example, assists with managing storage capacity of a memory-constrained handheld device when being synchronized with a database on a desktop PC.

If the current Date Range is large enough to accommodate more than the maximum number of instances which might be generated, those instances will be chosen which are likely to be most useful to the user. In the preferred embodiment, it is assumed that future instances are always more useful than past instances, that near future instances are more useful than distant future instances, and that recent past instances are more useful than distant past instances. Therefore, based on these assumptions, a fanning date range is calculated (FIG. **6**, step **236**).

Referring to FIG. **2**, in the second step of creating the Workspace, the Synchronizer establishes an Extended Index Array **20** which has an index entry associated with each entry in the record array. Each index contains the following variables:

1. Next_In_CIG:
2. Next_In_SKG:
3. Next_In_FIG
4. Key_Field_Hash
5. A_Unique_ID_Hash
6. B_Unique_ID_Hash
7. Non_Key_Field_Hash
8. Non_Date_Hash
9. Exclusion_List_Hash
10. Start_Date&Time
11. End_Date&Time
12. Various bit flags

Next_In_CIG is a linkage word, pointing to next member of the same Corresponding Item Group (CIG). A CIG is a group of records, one from each database and the History File, if applicable, which represent the same entry in each of the databases and the History File. There may be one, two or three records in a CIG. Next_In_SKG is a linkage word, pointing to next member of the Same Key Fields Group (SKG). An SKG is a group of records having the same key fields. Next_In_FIG is a linkage word, pointing to the next member of the Fanned Instances Group (FIG). A FIG is the group of fanned instances which correspond to a single recurring record.

Key_Field_Hash is hash of all Key_Fields. A_unique_ID_Hash is hash of unique ID, if any, assigned by A_Database. B_unique_ID_Hash is hash of unique ID, if any, assigned by B_Database. Non_Key_Field_Hash is hash of all Non-Key Match Field, a Match Field being any mapped field which is not flagged as No_Reconcile. Non_Date_Hash is hash of all Non-Date Non-Key Match Fields. Exclusion_List_Hash is hash of recurring record's exclusion list.

Start_Date&Time and End_Date&Time are used for Appointment and ToDo type record only, indicating the start and end date and time of the record. They are used to speed up comparing functions throughout the synchronization. Hash values are also used to speed up the process of comparison. The preferred embodiment uses integer hashes. Hash value computation takes into account certain rules of data value for fields, as will be described in more detail below.

In the preferred embodiment, the record array **21** is stored on magnetic disk of a computer whereas the Extended Index **20** is held resident in memory. The Extended Indexes have record pointer fields which point to each of the records on the disk file.

The Control Module **2** now instructs the synchronizer to load the History File into the Workspace (FIG. **3**, step **102**). Referring to FIG. **6**, the synchronizer loads the records beginning in first available spot in the Workspace (step **211**). The Synchronizer then performs an analysis on each of the records and resets some of the values in the records (steps **212–228**). The records are also checked against the current date range and those falling outside of it are marked appropriately for Fast synchronization function, which will be described below. In case of recurring records, if any of the instances is within the current date range, then the recurring

record itself will be considered within the current date range (steps **217–227**).

The synchronizer then builds SKGs by finding for each history record one record which has matching key fields and by placing that record in the SKG of the history record (step **215–216**). Referring to FIG. **7**, steps **250–258** describe the Key_Field_Match function used for matching records for SKG.

When comparing two records or two fields, in the preferred embodiment, the COMPARE function is used. The COMPARE function is intelligent comparison logic, which takes into account some of the differences between the rules of data value imposed by the A_Application and the B_Application on their respective databases. Some examples are as follows. The COMPARE function is insensitive to upper and lower case letters if case insensitive field attribute is present. Because some Applications require entries to be in all capital letter, the COMPARE function ignores the differences between upper and lowercase letters. The COMPARE function takes into account any text length limitations. For example, when comparing "App" in the A_Database and "Apple" in the B_Database, the COMPARE function takes into account that this field is limited to only 3 characters in the A_Database. It also takes into account limits on numerical value. For example, priority fields in the A_Application may be limited to only values up to 3, whereas in the B_Application there may not be any limitation. The COMPARE function would treat all values in B_records above 3 as 3.

The COMPARE function may ignore various codes such as end of line characters. It may strip punctuation from some fields such as telephone numbers and trailing white space from text fields (i.e "Hello " is treated as "Hello"). It also considers field mapping. For example, if the only line that is mapped by the A→B Map is the first line of a field, then only that line is compared. When comparing appointment fields, because different databases handle alarm date and time differently when Alarmflag is false, the COMPARE function treats them as equal even though the values in them are not the same. It skips Alarm Date and Time, if the Alarm Flag is False. It also ignores exclusion lists when comparing recurring records.

In an alternate embodiment, the COMPARE function may take into account more complicated rules for data value of the two Applications, such as the rules for data value imposed by Microsoft Schedule+, described above. Such a COMPARE function may be implemented as a table driven code, the table containing the rules imposed by the A_Application and the B_Application. Because the COMPARE function has a specific comparison logic and takes into account a number of rules, the hashing logic must also follow the same rules. It should be noted that the COMPARE function is used throughout the preferred embodiment for field comparisons.

Now that the History File is loaded into the Workspace, the Control Module **2** instructs the B_Translator **13** to load the B_Database records (FIG. **3**, step **103**). Referring to FIG. **8**, steps **300–308**, the B_Reader module **11** of the B_Translator **13** loads each B_record which has the right Origin Tag, which will be explained in more detail below.

The record must also be within the loading date range, which is a concatenation of the previous and current date ranges. The B_Translator sends these records to the Synchronizer which in turn stores them in the Workspace. When synchronizing with a date range limitation, all records which fall within either the previous or the current date ranges are loaded. The current date range is used during unloading to limit the unloading of the records to only those records which fall within the database's current date range. In an alternate embodiment of the invention, each database or Application can have its own date range for each synchronization.

Most Applications or databases permit record-specific and field-specific updates to a Database. But some Applications or databases do not. Instead the Translator for these Application must re-create the whole database from scratch when unloading at the end of synchronization. These databases are identified as Rebuild_All databases. To accommodate this requirement all records from such a database must be loaded into the Workspace, so that they can later be used to rebuild the whole database. These databases records, which would otherwise have been filtered out by the date range or the wrong origin tag filters, are instead marked with special flag bits as Out_Of_Range or Wrong_Section_Subtype. These records will be ignored during the synchronization process but will be written back unmodified into the database from which they came by the responsible Unloader module **6**, **10**.

Control Module **2** next instructs the A_Translator **5** to sanitize the B-records. Referring to FIG. **9**, steps **350–361**, the A_Sanitizer module **8** of the A_Translator **5** is designed to take a record having the form of an A_Record and make it conform to the specific rules of data value imposed by the A_Application on records of the A_Database. A_Sanitizer is not aware which database's field and records it is making to conform to its own Application's format. It is only aware of the A_Application's field and record structure or data structure. Therefore, when it requests a field from the sanitizer using the A_Database field name, it is asking for fields having the A_Database data structure. The Synchronizer, in steps **375–387**, therefore maps each record according to the B→A_Map. In turn, when the Synchronizer receives the fields from the A_SANITIZER, it waits until it assembles a whole record (by keeping the values in a cache) and then maps the record back into the B format using the A→B_Map.

How a record or a field is sanitized in step **354** and **357** depends on the rules of data value imposed by the A_Application. For example, all of the logic of intelligent comparison in the COMPARE function described above can be implemented by sanitization. However, sanitization is best suited for more complex or unique types of database rules for data value. For example, consider the Schedule+ rules regarding alarm bearing Tasks records described above. FIG. **10** shows a sanitization method for making records of incompatible databases conform to the requirements of Schedule+. Without sanitization, when a Tasks record of a Schedule+ database is compared to its corresponding record in another database, the Tasks record may be updated in fields which should be blank according to the Schedule+ rules of data value. Such an update may possibly affect the proper operation of Schedule+ after synchronization.

Referring to FIG. **11**, following sanitization of all B_Records into the Workspace, the Synchronizer sets the values for the Extended Index of each record based on the record's values (steps **451–459**). Also if the records in the B_Database bear a unique ID, and matches for those unique IDs are found in the H_Records in the Workspace, the two records are joined in a CIG because they represent the same record in both History File and B_Database (step **462**). The record is also joined to an SKG it may belong to (step **464**). The loading of B_Records is now complete.

The Control Module **2** of the Translation Engine **3** now instructs the A_Translator **5** to load the records from the

A_Database (step **105**). The loading process for the A_Records is the same as the loading process for the B_Database, except for some differences arising from the fact that records in the Workspace are stored according to the B_Database data structure. Therefore, as the synchronizer **15** receives each A_record from the A_Reader module **7** of the A_Translator **5**, the Synchronizer maps that record using the A→B_Map before writing the record into the next available spot in the Workspace. Since the A_records are mapped into the B_Record format, when the B_Sanitizer is instructed by the Control Module **2** to begin sanitizing those records and starts asking for them from the synchronizer, they already have the B_Database format. Therefore, the synchronizer **15** does not need to map them before sending them to the B_Sanitizer module **12** of the B_Translator **19**. For the same reason, there is no need for them to be mapped once they are sent back by the B_Sanitizer after having been sanitized. Once all the records are loaded, the records will undergo the same orientation analysis that the B_Records underwent (FIG. **11**).

At this point, all records are loaded into the Workspace. SKGs are complete since every record at the time of loading is connected to the appropriate SKG. CIGs now contain all records that could be matched based on unique IDs. At this point, the records in the Workspace will be analyzed according to Conflict Analysis and Resolution ("CAAR") which is set out in FIG. **12** and in more detail in FIGS. **13–18** and corresponding detailed description.

First, in step **500**, ID bearing fanned instances in the History File records are matched to the fanned instances in the ID bearing database from which they came. The records from the database which have remained unchanged are formed into a new FIG. A new Synthetic Master is created based on those records and joined to them. The records which have been changed or deleted since last synchronization are set free as single records. They also result in a new exclusion list being created based on an old exclusion list and these new single records.

Second, in step **501**, matches are sought for the ID based CIGs which are the only CIGs so far created in order to increase the membership of those CIGs. Preferably an exact all fields match is sought between current members of a CIG and a new one. Failing that, a weaker match is sought.

Third, in step **502**, master/instances match is sought between recurring records and non-unique ID bearing instances by trying to find the largest group of instances which match certain values in the Recurring Master.

Fourth, in step **503**, the items remaining in the SKGs are matched up based on either exact all field match or master/instance match, or a weaker match.

Fifth, in step **501**, the appropriate CIG_Types are set for all the CIGs. CIG_Types will determine what the outcome of unloading the records will be.

Referring to FIG. **13**, first step in CAAR is analyzing unique ID bearing Fanned Instance Groups. This analysis attempts to optimize using unique IDs assigned by databases in analyzing fanned instances of recurring records.

The analysis is performed for all Recurring Masters (i.e. all recurring records) which have ID-bearing fanned instances (or FIG records) in the H_File (step **550**). All FIG records in the History File associated with a Recurring Master are analyzed (steps **551–559**). They are all removed from the SKG. If a FIG record is a singleton CIG, it means that it was deleted from the database since the previous synchronization. Therefore, it is added to the New_Exclusion_List (step **553**). If a FIG record is a doubleton and is an exact match, it means that the record was not modified since the previous synchronization. In this case, the record from the database is also removed from SKG (step **555**). If a FIG record is a doubleton but is not an exact match for its counterpart in the database, it means that the record was changed in the database. The History File record is treated as a deletion and therefore added to the New_Exclusion_List. The modified record in the database, which does not match the recurring record any longer, is treated as a free standing record un-associated with the Recurring Master (step **557**).

Upon analysis of all FIG records, a new record, the Synthetic Master, is created and joined in a CIG with the Recurring Master (step **231–236**). The Synthetic Master has the same characteristics as the Recurring Master, except that it has a new exclusion list which is a merger of the New_Exclusion_List and the Exclusion_List of the Recurring Master (step **563**). Also a new FIG is created between the Synthetic Master and the CIG-mates of all FIG records from the History File (step **565**).

In steps **567–569**, the Synchronizer checks to see if there are some instances of the Recurring Master which fall within the previous synchronization's date range but fall outside of the current synchronization's date range. If so, the Fan_Out_Creep flag is set, indicating that the date range has moved in such a way as to require the record to be fanned for the database before unloading the record. The Fan_Out_Creep flag is an increase in the value in the Non_Key_Field Hash of the Recurring Master. In this way, the Recurring Master during the unloading of the records will appear as having been updated since the last synchronization and therefore will be fanned for the current date range.

In step **570**, all the FIG records analyzed or created in this analysis are marked as Dependent_FIGs. This results in these records being ignored in future analysis except when the recurring records to which they are attached are being analyzed.

At the end of the above analysis, all the records having a unique ID assigned by their databases have been matched based on their unique ID. From this point onward, the records which do not have unique IDs must be matched to other records based on their field values. In the preferred embodiment, there are two categories of field value matches: strong matches and weak matches. A strong match between two records that have matching key fields is when non-key fields of the two records match or it is a Recurring Master and a fanned instance match (FIG. **14**, steps **606–610**). Referring to FIG. **15**, a weak match between two records that have matching key fields is when the following are true: each of the two records are from different origins, because two records from the same source should not be in a CIG (e.g., A_Database and History File); each is not a weak match for another record because there is no reason to prefer one weak match over another; each is not a Dependent_FIG since these records do not have an independant existence from their recurring masters; both records are either recurring or nonrecurring since a recurring and a nonrecurring should not be matched except if one is an instance of the other in which case it is a strong match; and, in case of non-recurring, they have matching Key_Date_Field which is the same as the Start_Date in the preferred embodiment because items on the same date are more likely to be modified versions of one another.

Referring to FIG. **14**, these two types of matching are used to match records to existing CIGs for History File records which have been created based on matching unique IDs. Only doubleton CIGs are looked at, because singleton CIGs are handled in step **504** of FIG. **12** and tripleton CIGs are

complete (steps **601–604**). If a strong match is found, then if the record was a weak match in another CIG, it is removed from that CIG, and new weak match is found for that CIG (**612–614**). While weak matches are left in SKGs in case they will find a strong match, strong matches are removed from their SKGs (step **614**). If a strong match is not found, then a weak match is sought (steps **617–620**). All records in the CIG are removed from SKG if no weak match is found, because this means that there is no possibility of even a weak match for this record (step **619**).

The next step in CAAR is finding non-unique ID bearing instances for recurring items (FIG. **12**, step **503**). Referring to FIG. **16**, this analysis takes place only if the database from which instances matching a recurring record are sought does not provide unique ID or if we are synchronizing from scratch (steps **650–653**). The goal of this analysis is to find matching instances for each Recurring Master from a different source than the Recurring Master. This analysis counts the number of records in SKG of the Recurring Master which have matching Non_Date_Hash value (steps **665–669**). The group of matching SKG records having the same non_Date_Hash value and having the highest number of members (if the number of members exceeds 30% of unexcluded instances) is then formed into a Homogeneous_Instances_Group (steps **670–672**). A Synthetic Master is created using the Rep_Basic of the Recurring Master and using the values from the homogeneous instances group. An Exclusion list is created based on the items belonging to the recurrence pattern but missing from the Homogeneous_Instances_Group. The Synthetic Master is added to the CIG of the Recurring Master (steps **673–678**). A new FIG for the Synthetic Master is then created using the Homogeneous_Instances_Group (step **679**). These records are removed from any CIGs to which they belonged as weak matches and new weak matches are sought for those CIGs (steps **680–684**). Since the records in Homogeneous_Instances_Group have now been matched to a recurring record, they are marked as Dependent_FIGs (step **683**). The Recurring Master's CIG is then marked with Fan_Out_Creep flag, if necessary (step **685**).

The next step in CAAR is completing analysis of records in SKGs (FIG. **12**, step **504**). Referring to FIG. **17**, this analysis attempts to increase the population of CIGs up to a maximum by finding key field based matches with records from a source different from those of the CIG records. This analysis is performed by analyzing all the records in the SKGs except for the singleton SKGs (steps **703** and **712**). The first thing is to remove any members that have already been marked as WEAK matches attached to ID-based doubleton CIGs. Those are left in the SKG up to this point to allow for the possibility that a STRONG match would be found instead. But that is not possible any longer (steps **713–715**). Once the weak matches have been removed, all remaining SKG members belong to singleton CIGs. Any non-singleton CIGs which are formed from here on will be purely key field based.

Throughout the remaining SKG Analysis we are careful not to seek H_Record-A_Record or H_Record-B_Record matches for unique ID-bearing Source, since that would violate the exclusively ID-based matching scheme that applies in such cases. Note however that an A_Record-B_Record match is acceptable even if both A_Database and B_Database are unique ID-bearing databases.

Given that Key Field should not be performed where ID based matches are available (or otherwise there may be matches between records with differing IDs), there are limits to how big CIGs can get at this point. If both A and B_Databases are unique ID-bearing, any remaining H_Record must remain in Singleton CIGS, because they are prohibited from forming key fields based matches with items from either databases. Such H_Records are simply removed from the SKG when they are encountered. If just one of the two databases being synchronized is unique ID-bearing then the maximum population that any CIG can now attain is **2** (FIG. **18**, steps **750–751**). If neither database is unique ID bearing then the CIG_Max_Size is three. For every CIG which is analyzed in FIG. **17**, the CIG_Max_Size is set according to this logic. When a CIG reaches its maximum possible population all of its members are removed from the appropriate SKG.

First, strong matches for the H-records are searched for, before trying to find A-B matches. If both Databases are non-unique ID-bearing then two strong matches for each H_Record, an H-A and an H-B match, are sought (steps **715–720**). If finding a strong match results in reaching the CIG_Max_Size, all members of the CIG are removed from the SKG (step **721**).

When maximum CIG population is 3, weak matches are sought for strong matching CIG doubleton in order to build triplet CIGs. The first weakly matching SKG member is added to the CIG (steps **722–728**). Whether or not a weak match is found for any of the doubleton CIGs, its members are removed from the SKG (step **726**). As there are no strong matches left in the SKG, weak matches are found for any remaining SKG members and joined to them in CIGs (steps **722–725**).

At this stage, all CIGs are built. They must now be examined to determine what needs to be done to these records so that the databases are synchronized, i.e. whether the records in the CIGs need to be added, deleted or changed in the two databases. First step is determining the CIG_TYPE which represents the relation between the records. The following CIG types are defined, all using a 3-digit number that represents values found for A_DATABASE, History File, and B_Database, respectively:

1. **001**—record is "new" in the B_DATABASE
2. **010**—record is present in History, but absent in both A_Database and B_Databases
3. **100**—record is "new" in the A_Database
4. **101**—record is "new" in both A_Database and B_DATABASE; same in both
5. **102**—record is "new" in both A_Database and B_DATABASE; different in each (conflict)
6. **110**—record deleted from B_DATABASE
7. **011**—record deleted from A_Database
8. **012**—record deleted from A_Database and changed on B_DATABASE (DEL vs CHANGE conflict)
9. **210**—record changed on A_Database and deleted from B_DATABASE(DEL vs CHANGE conflict)
10. **111**—record unchanged since previous synchronization
11. **112**—record changed on B_DATABASE only since previous synchronization
12. **211**—record changed on A_Database only since previous synchronization
13. **212**—record changed identically on both since previous synchronization
14. **213**—record changed differently on each since previous synchronization (conflict)
15. **132**—a conflict (**102** or **213**) was resolved by forming a compromise value; Update both

16. **13**F—created when a **132** Update both CIG is Fanned into the B_DATABASE

FIG. **19** shows the method used for setting all except the last two CIG_Types which are set in other operations.

Four of the CIG types assigned above involve conflicts: **102**, **213**, **012**, and **210**. Conflicts are those instances where a specific conflict resolution rule chosen by the user or set by default, or the user's case by case decision, must be used to determine how the records from the databases should be synchronized. CIG types **012** and **210** are cases where a previously synchronized record is changed on one side and deleted on the other. In the preferred embodiment, such conflicts are resolved according to the rule that CHANGE overrules the DELETE. So the net result for CIG type **012** is to add a new record to the A_Database to match the record in the B_DATABASE. The reverse is true for CIG type **210**, where a new record is added to the B_Database. In an alternate embodiment, the user may be allowed to register an automatic preference for how to resolve such conflicts or decide on a case-by-case basis a conflict resolution option.

The other two conflict types—**102** and **213**—are resolved in the preferred embodiment according to the Conflict Resolution Option established by the user. First, the user may choose to ignore the conflict. This option leaves all **102** and **213** conflicts unresolved. Every time synchronization is repeated the conflict will be detected again and ignored again, as long as this option remains in effect and as long as the conflicting records are not changed by other means.

The user may choose to add a new record to each of the two databases. This option resolves **102** and **213** conflicts by adding the new A_Record to the B_Database, and adding the new B_Record to the A_Database. This option is implemented by breaking a **102** CIG into two separate CIGs (types **100** and **001**) and a **213** CIG into three separate CIGs (types **100**, **010**, and **001**). Subsequent processing of those descendant CIGs causes new records to be added across and stored in the History File.

The user may elect that A_Database records should always trump or win over B_database records. This option is implemented by changing the CIG type to **211**—the processing during unloading the records changes the record value in the B_Database to match the current record value in the A_Database.

The user may elect that B_Database records should always trump or win over B_database records. This option is implemented by changing the CIG type to **112**—the processing during unloading the records changes the record value in the A_Database to match the current record value in the B_Database.

The user may choose to be notified in case of any conflict. The user is notified via a dialog box **30**, shown in FIG. **20**, whenever a CIG type conflict of **102** or **213** arises. The dialog box shows the record that is involved in the conflict **31**. It also shows the A_Database **32** and B_Database **33** values for all conflicting fields, in a tabular display, with Field Names appearing in the left column **34**. A dropdown list (not shown) in the lower left hand corner of the dialog **37**, offers a total of three choices—add, ignore, and update. The use may choose to add new records or ignore the conflict. The user may also choose that the A_Record or B_Record should be used to update the other record. The user may also decide to create a compromise record by choosing values of different fields and then choosing update option. In this case, the CIG type is changed to **132**, which results in an updating both databases with the new record compromise record.

When the user has chosen to be notified in case of conflict, if the user chooses to ignore conflict or that either the record of the A_Database or the B_DATABASE should win, the CIG type is left as a conflict CIG type (**102** or **213**) and a separate Conflict Resolution Choice is stored in the FLAGS word associated with each CIG member.

The final step in setting CIG_Types is the process for dealing with difficulties which arise from exclusion lists. For example, in a triple Recurring Master CIG, suppose the History File Recurring Master does not have any excluded instances. The A_Record has the following exclusion list:

12/1/96, 12/8/96 The B_Record has the following exclusion list:

1/1/97, 1/8/97, 1/15/97, 1/22/97, 1/29/97

If comparison of the Recurring Masters includes comparing exclusion list Field Values, this set of changes would cause the Synchronizer to report a CIG type **213** conflict.

If the Conflict Resolution Option is set to A_Database record wins, then the outcome prescribed by the Synchronizer would be for the A_Database to keep its exclusion list as is and for the B_Database to make its exclusion list match that of the A_Database.

The result would be to have a lot of duplicate entries in both Databases. The A_Database would have five duplicate entries in January 97—that is the five unmodified Recurring Master instances, plus the five modified instances added across from B_Database to A_Database. The B_Database would have five duplicate entries in January 97, since synchronization has wiped out the five exclusions that were previously recorded in the B_Database exclusion list.

Two steps are implemented for dealing with this problem. First, the COMPARE function does not take into account exclusion list differences when comparing recurring records. Second, referring to FIG. **21**, any new exclusions added on to one recurring record will be added to the other record. The merging of exclusion lists is done regardless of any updates or conflicts, even unresolved conflicts, between the A_Database and B_Database copies of a Recurring Master. One exception is for CIG type **102** conflict which is left unresolved where Exclusion lists are not merged, because the user has chosen to leave those records as they are.

In most cases where it is necessary to merge exclusion lists, the CIG types and/or the Conflict Resolution Choice to arrange for all necessary updates to be performed during the unloading phases of synchronization.

First, A_Database and B_Database records' exclusion lists are compared. In case of databases which do not permit recurring items, the exclusion list of the Synthetic Master is compared to the recurring record of the other database (step **852**). If there is no difference, then nothing is done (step **853**). If there are differences, then it is determined which exclusions appear only in one record. This comparison always yields one of the following scenarios: (1) all one-side-only Exclusions are on the A_Database (so Exclusions should be added to the B_Database); (2) all one-side-only Exclusions are on the B_Database (so Exclusions should be added to the A_Database); and (3) there are one-side-only Exclusions on both sides (so Exclusions should be added to both databases).

In each of these cases a separate table is used to look up instructions, for how to handle each specific situation (FIGS. **22–24**). The tables cover all possible combinations of previous CIG types and outcome codes with all possible exclusion list changes (new and different exclusions added on A_Database, or on B_Database, or on both sides). FIG. **22** table is used in case of scenario **1**. FIG. **23** table is used in case of scenario **2**. FIG. **24** table is used in case of scenario **3** (FIG. **21** steps **854–856**).

The analysis of records is now complete, and the records can be unloaded into their respective databases, including any additions, updates, or deletions. However, prior to doing so, the user is asked to confirm proceeding with unloading (FIG. **3**, step **108–109**). Up to this point, neither of the databases nor the History File have been modified. The user may obtain through the Translation Engine's User Interface various information regarding what will transpire upon unloading.

If the user chooses to proceed with synchronization and to unload, the records are then unloaded in order into the B_Database, the A_Database and the History File. The Unloader modules **6,10** of the Translators **5,9** perform the unloading for the databases. The Synchronizer creates the History File and unloads the records into it. The Control Module **2** of the Translation Engine **1** first instructs the B_Translator to unload the records from Workspace into the B_Database. Referring to FIG. **25**, for each CIG to be unloaded (determined in steps **902–907**), based on the CIG_TYPE and which database it is unloading into (i.e., A or B), the unloader looks up in the table in FIG. **26** the outcome that must be achieved by unloading—that is, whether to update, delete, add, or skip (Leave_Alone) (step **908**). In steps **909–913**, the unloader enforces date range restriction for a database subject to date range. The user may select, or a selection may be made by default, whether to enforce the date range sternly or leniently. In case of stern enforcement, all records outside of the current date range would be deleted. This is useful for computers with small storage capacity. In case of lenient enforcement, the records are left untouched.

Based on the result obtained from looking up the unloading outcome in the table, the unloader then either adds a new record (steps **920–926**), deletes an existing record (steps **914–919**), or updates an existing record (steps **927–933**). It should be noted that because we only update those fields which need to be updated (step **928**), the fields which were sanitized but need not be updated are not unloaded. Therefore, the values in those fields remain in unsanitized form in the database.

Referring to step **914**, in some Applications when a Recurring Master must be added or updated, the record may have to be fanned out despite the ability of the Application to support recurring records. For example, the Schedule+ Translator is generally able to put almost any Recurring Master Item into Schedule+ without fanning, but there are some exceptions. The Schedule+ Translator uses one Schedule section to handle all appointments and events. For appointments, almost any recurrence pattern is allowed, but for events the only allowable true repeat type is YEARLY. DAILY recurring events can be dealt with by being translated into Schedule+ multi-day events which are not recurring but extend over several days by setting the EndDate some time after the Start Date. But for the DAILY case there are restrictions. In particular exclusions in the midst of a multi-day Schedule+ event cannot be created. So the Translator decides that if section type is ToDos or the item is a non-Event Appointment, then the record need not be fanned out. But if item is a YEARLY or DAILY with no exclusions then it can be stored as a Schedule+ yearly or daily event. Otherwise, it must be fanned.

Referring to FIG. **27**, steps **950–984** set out the preferred embodiment of fanning recurring records that must be updated. All cases fall within three scenarios, shown in FIG. **29**.

In the first scenario a record which is a Recurring Master, and its counterpart in the other database is a Recurring Master, must be fanned now for its own database (steps **951–959**). If the CIG_TYPE of the record is **132** (i.e. update both records), then it is changed to **13**F which is a special value specifically for this situation (step **951**). For other CIG_Types, the CIG is broken into three singleton and given CIG_Types signifying their singleton status. In both of these cases, the function Fanning_For_Add (steps **986–996**, described below) is called.

In the second scenario, the record was fanned previously and is going to be fanned now also. First, the dates of the instances are recorded in a temporary date array (steps **961–963**). This array is compared to an array of the fanned instances of the recurrence pattern of the CIG Recurring Master from the other database (steps **965–966**). The dates which are not in the array of fanned instance are marked for deletion (step **967**). The dates which are not in the temporary date array should be added to the unloading databases and therefore new FIG records are created for those dates (steps **968–973**). The dates which appear in both arrays are compared to the Synthetic Master and marked accordingly for UPDATE or Leave_Alone (steps **974–978**).

In the third scenario, the record which was previously fanned should now be fanned also. The opposing database's record in this scenario is also fanned instances. This is perhaps the most peculiar of the three cases. For example, a database may be able to handle multi-day (i.e. daily recurring) records but not any exclusion dates for such items. Such database may be synchronized with another database which fans all records in the following manner. A record representing a 7-day vacation in the Planner section of the database is fanned out to form 7 individual vacation days in the other database. One instance is deleted in the other database. Upon synchronizing the two databases, b/c the first databases does not does not provide for exclusion lists, the record must now be fanned.

In this scenario, Master Records in a CIG are marked as Garbage. Any FIG members attached to the H_Record, if any, are also marked as Garbage. All Instances found in the opposing database's FIG are truned to singleton CIGs with CIG type **100** or **001** so that they will be added to the unloader's database when unloading is done. In this way the instances from one database is copied to the database providing for recurring records.

Steps **985–995** describe the Fanning_For_Add Function which is used when outcome is to update or when the function is called by the Translator fanning for update. For each instance generated by fanning out the recurring record, a clone of the Recurring Master is created but excluding Rep_Basic and Rep_Excl field values and the unique ID field. All adjustable Date Fields (e.g. Start Date, End Date, and Alarm Date) are set and hash values for the new record is computed. The new record is then marked as Fanned_For_A or Fanned_For_B, as the case may be. This is then attached to the Recurring Master Item as a FIG member.

Following unloading of the B_RECORDS, the Control Module **2** instructs the A_Translator to unload the A_Records from the Workspace (FIG. **3**, step **111**). This unloading is done in the same way as it was done by the B_Translator. In case of Rebuild_All Translators which have to reconstruct the database, all records which were loaded from the database but were not used in synchronization are appended and unloaded as the Translator builds a new database for its Application.

The Control Module **3** next instructs the Synchronizer to create a new History File (step **112**). Referring to FIG. **28**, for every CIG in the Workspace, it is first determined which record should be unloaded to History File (steps

1001–1003). In the next step, Excl_Only flag is checked, which is set by the Merge_Exclusion_List logic (FIGS. 21–24). If that flag is set, a new record for unloading is created which has all fields taken from the History File record, except that the newly merged exclusion list is inserted into that record (step 1004). Before storing the record in the History File, all Flag Bits in the Extended Index are cleared except the bit that indicating whether or not this is a recurring item (step 1005). The item is marked as a History File record to indicate its source. The CIG, FIG, and SKG are reset. All the HASH values and Start&EndDate&Time will be stored. All applicable unique ID are also stored (Steps 1006–1009). The current record is then stored in the new History File (step 1010). If the current record is a Recurring Master for an ID-bearing FIG, we now store the whole FIG (i.e. all Fanned Instances) in the History File, with the FIG linkage words set in the History File to hold the FIG records together (step 1011). Fanned instances which do not bear unique IDs are not stored in the History File since they can be re-generated by merely fanning out the Recurring Master.

Once all records are unloaded, various information necessary for identifying this History File and for the next synchronization are written into the History File (step 1013).

At this point Synchronization is complete.

Applications, such as scheduling Applications, often have more than one database. Each of these databases are known as sections. Each of these sections contain different data and must be synchronized with their corresponding sections in other Applications. However, there is not necessarily a one to one relationship between sections of various Applications. For example, Application A may comprise of the following sections: Appointments, Holidays, Business Addresses, Personal Addresses, and ToDo. Application B however may comprise of the following sections: Appointments, Addresses, ToDo-Tasks, and ToDo-Calls. Although the general character of the sections are the same, there is not a one to one relation between the sections of these two Applications: Appointments and Holidays in A contain the same type of data as Appointments in B; Business Addresses and Personal Addresses in A contain the same type of data as Addresses in B; and ToDo in A contains the same type of data as ToDo-Tasks and ToDo-Calls in B. Therefore, when synchronizing the sections of these two Applications, it is necessary to synchronize at least two sections of one Application with one section of another Application.

The preferred embodiment performs this type of synchronization by providing for a number of section categories: Appointment, ToDo, Note, Address, and General Database. All sections of a particular Application are studied and categorized according to this categorization. Therefore, in the above example of Application A, Appointments and Holidays are categorized Appointment type sections (or database), Business Address and Personal Address as Address type sections, and ToDo as a ToDo type section.

For creating the map for mapping sections onto each other, an exact section match is always sought between sections of the two Applications. If not, one of the sections which were categorized as a section type is chosen to be the Main_Section among them. Other sections of the same type are referred to as subsections. All databases of the same type from the other Application will be mapped onto the Main_Section.

To properly synchronize from one time to the next, it is necessary to keep track of the source of records in the Main_Section. In the preferred embodiment, if a record in the Main_Section of the A_Application does not come from the Main_Section of the B_Application, one of fields in the record, preferably a text field, is tagged with a unique code identifying the subsection which is the source of the record. This is the record's Origin Tag. All records in the Workspace and the History File include a hidden internal field called _subType which contains the unique subsection code. Main_Section's field value in the preferred embodiment is zero so that it will not be tagged. When a record is loaded from a database into the Synchronization Workspace, the tag is stripped from the TagBearer field and put in the _subType field. If there is no tag, then the _subType is set to be the subType of the present section. If the TagBearer field is mapped then when reading records into the Workspace the tag, if any, is stripped from the TagBearer field value place it in _subtype.

Conversely when unloading records from the Workspace to a Database, the TagBearer field is tagged by a tag being added if the record is not from the Main_Section.

A Fast Synchronization database is a database which provides a method of keeping track of changes, deletions, and additions to its records from one synchronization to the next. These databases speed up the synchronization process because only those records which have been modified need to be loaded from the database. Since the majority of records loaded by regular Translators are unchanged records, far fewer records are loaded from the database into the Synchronizer.

Certain features are required for a database to be a Fast Synchronization database. The database records must have unique IDs and must have a mechanism for keeping track of which records are added, changed, or deleted from synchronization to synchronization, including a list of deleted records. Unique IDs are required to accurately identify records over a period of time.

There are at least two ways to keep track of additions, changes, and deletions in a database.

First, some databases maintain one Dirty bit per record which is a boolean flag that is set when a record is created or modified and is cleared when a function for clearing Dirty bits is called. Some databases offer a Clear DirtyBit function that clears the bit of an individual record. Other databases offer a ClearDirtyBits function that clears the Dirty bits of all records in a database. The record-specific ClearDirtyBit function allows the preferred embodiment to use the database itself to keep track of additions and changes.

The global ClearDirtyBits function forces the preferred embodiment to clear all Dirty bits at the conclusion of every Synchronization. Then as database edits are made by the user in between synchronizations, the affected records are marked as Dirty. When Synchronization is performed again, only the Dirty records are loaded.

Second, some databases maintain a Date&Time stamp of when the record was added or last time the record was modified. A Translator for such a database finds all records which were added or modified since the previous synchronization by searching for Date&Time stamps more recent than the Date&Time of the Last Synchronization.

A Fast Synchronization database must also keep track of deletions. This is done by maintaining a list of deleted records which can be read by a Translator.

A Translator sending Fast Synchronization database records to the Synchronizer provides only records which have been changed, deleted, and added since the previous synchronization. Therefore, unlike a regular database Translator, a Fast Synchronization Translator does not provide the Synchronizer with unchanged records. Moreover, unlike a regular Translator it provides deleted records, which the regular Translators does not.

In order for such databases to be synchronized without resorting to treating them as regular databases, the Synchronizer transforms Fast Synchronization records from the Translator into the equivalent regular database records. These transformed records are then used by the Synchronizer in the synchronization. There are two transformations which are necessary. First, the Synchronizer needs to transform deleted records received from the Fast Synchronization Translator into a regular database deletions. Second, synchronization needs to transform lack of output by the Fast Synchronization Translator into unchanged records.

The invention performs these transformations by using the History File. During the first synchronization, all records in the Fast Synchronization database are loaded into the history file. As changes, additions, and deletions are made to the Fast Synchronization database, during each of the subsequent synchronizations the same change, additions, and deletions are made to the History File. Therefore, the History File at the end of each subsequent synchronization is an exact copy of the Fast Synchronization database.

When a Fast Synchronization Translator supplies no input for a unique ID H_Record, the Synchronizer finds the corresponding H_Record in the Workspace and copies it into the Workspace as a record supplied as if it were loaded by the Fast Synchronization translator itself.

Referring to FIG. **30**, steps **1050–1051**, the Synchronizer first verifies that there is an appropriate History File. Because the Fast Synchronizing process relies heavily on the History File, it is important to ensure that the same history file as the last Synchronization is used. Moreover, the History File is the background against which the transformation of the Translator outputs into regular Translator outputs takes place. The History File keeps a date and time stamp of the last synchronization. Each of the Fast Synchronization database (if able to) and the Fast Synchronization Translator also stores the same date and time stamp. The time and date stamp is used because it is unlikely that another History File will have exactly the same time and date entry, for the same two databases. It also identifies when last the Fast Synchronizer database and the History File contained the same records.

At the start of an incremental synchronization, the Synchronizer and the Fast Synchronization Translator compare date and time stamps. If time and date stamp synchronization parameters have changed since the previous synchronization, then the synchronization proceeds from scratch (step **1052**). In a synchronization from scratch all records in the Fast Synchronization database are loaded into the History File.

In the preferred embodiment, all records supplied as Fast Synchronization inputs have a special hidden field called _Delta, which carries a single-letter value—'D' for Delete or 'A' for Add and 'C' for Change. Records are loaded by the Fast Synchronization Translator into the Workspace (step **1054**). If necessary the records are mapped when loaded. Records which are marked as changes or additions are sanitized by the Translator for the other database, but deleted records are not because their field values are going to be deleted (step **1055**). Orientation analysis (FIG. **11**) is performed on the records so that all deletions and changes to Fast Synchronization database records are joined with their History File counterparts in unique ID bearing CIGs (step **1107**).

All History File records and their CIGs are now examined. If there is no corresponding record from the Fast Synchronization database, it means that the record was unchanged. A clone of the record is made, labelled as being from Fast Synchronization database, and joined to the H_Record's CIG. At this point the deleted Fast Synchronization database records marked as deletions are removed from CIGs (step **1109**). The Fast Synchronization records marked as changed are joined in doubleton CIGS. Those marked as additions are singletons. At this point, the synchronization can proceed as if record of a unique ID bearing regular database were just loaded into the Workspace.

Whenever we are loading from a Fast Synchronization database, all records are loaded so that at the end of synchronization the history file will be the same as the Fast Synchronization Database. Therefore, referring to FIG. **31**, in order to perform date range limited synchronization, the invention marks the records which fall outside the current and the previous date ranges. For a record marked as an addition, or during synchronizing from scratch, if the record falls outside the current date range, it is marked as Out_Of_Range (steps **1101** and **1153–1154**). This record will be written into the History File but not into the other database or take part in the synchronization. When the Fast Synchronization database records are loaded from the History File, if they fall outside of the previous date range, they are marked as Bystander (steps **1152–1157**). If a Bystander record forms a CIG with a Fast Synchronization record marked as a deletion or a change, the Bystander is marked with a Garbage flag because its field values serve no useful purpose any more: the record marked as DELETION should be deleted and the record marked as CHANGED should replace the Bystander H_Record (step **1162**).

H_Records for which there are no inputs are transformed in the same manner as before (steps **1164–1165**). If a Bystander record falls within the current date range, it is equivalent to a regular database record coming into the current date range. Therefore, the H_Record is cloned and marked as a Fast Synchronizer record while the Bystander record is marked as Garbage (steps **1166–1171**). Therefore, just like a new record of a regular database, it has no H_Record counterpart.

If the user selects to abort a synchronization or selects the option to ignore a conflict or conflicts in general, some of the records loaded from the Fast Synchronization database will not be accepted and recorded in the History File. Therefore, the Translator should provide that record again at the next synchronization. However, because Fast Synchronization Translators supply only records which have been changed, deleted, or added since the previous synchronization, the records which were not accepted will not be supplied. Therefore, in the invention, Fast Synchronization Translator waits for an acknowledgement from the Synchronizer that the record has been accepted.

In case no such acknowledgement is received for a record, the Translator needs to be able to provide that record again to the Synchronizer. If the database allows resetting individual Dirty bits, the Translator merely does not set that bit. If not, the Translator keeps a separate file in which it keeps a record of which Fast Synchronization records were not accepted. The file may contain the unique IDs of those records. The Translator then uses that file to provide the synchronizer with those records during the next synchronization.

Other embodiments are within the following claims.

I claim:

1. A computer implemented method of synchronizing at least a first and a second database each containing dated records such as events, wherein the records of the first database extend across a narrow date range narrower than the date range of the records of the second database, the method comprising:

performing a prior synchronization across a prior date range set using the date of the prior synchronization and the narrow date range;

storing the prior date range and a history file containing information representative of the content of the databases following the prior synchronization;

performing a current synchronization across a date range that combines the prior date range with a current date range set using the date of the current synchronization and the narrow date range.

2. The method of claim **1** wherein the date of each record being synchronized is compared to a start and stop date of a date range to determine whether the record is in range.

3. A computer implemented method of synchronizing at least a first and a second database, each one containing date bearing records, wherein the date comprises a user-specified date, the method comprising:

identifying date bearing records of the first and second database that are within a narrow date range narrower than a date range of the records of one of the first and the second databases; and

performing a current synchronization across the narrow date range by synchronizing the identified date bearing records.

4. The method of claim **3** wherein the step of performing a current synchronization further comprises adding, modifying, or deleting records of the first database within the narrow date range.

5. The method of claim **3** wherein the step of performing a current synchronization further includes deleting records of the first database that are outside of the narrow date range.

6. The method of claim **3** wherein the narrow date range has a start date and a stop date and the date of a record being synchronized is compared to the start date and the stop date of the narrow date range to determine whether the record is within the narrow date range.

7. The method of claim **6** wherein the date of a record being synchronized includes a record start date and a record stop date, the method further comprising:

performing a comparison of the record start date to the stop date of the narrow date range and of the record stop date to the start date of the narrow date range;

determining based on the comparison whether the record is within the narrow date range.

8. The method of claim **3** wherein the first database contains a recurring record, the method further comprising fanning the recurring record within the narrow date range.

9. The method of claim **3** wherein the first database contains a recurring record, further comprising:

fanning the recurring record within a useful portion of the narrow date range, the useful portion being determined by application of a preference based on a current date of the current synchronization.

10. The method of claim **9** wherein the preference includes a preference for future dates compared to the current date over past dates compared to the current date.

11. The method of claim **9** wherein the preference includes a preference for dates closer to the current date over dates further from the current date.

12. The method of claim **3** wherein a prior synchronization was performed across a prior narrow date range, such prior narrow date range being different from a current narrow date range of the current synchronization, wherein records representatives of the records of the first and second databases during the prior synchronization are stored in a history file, and wherein performing the current synchronization further comprises performing the synchronization using the history file and the prior narrow date range.

13. The method of claim **12** wherein the narrow date range is a concatenation of the current narrow date range and the prior narrow date range stored in a history file during the prior synchronization.

14. The method of claim **12** wherein the step of performing a current synchronization further comprises adding, modifying, or deleting records of the first database that are within the current narrow date range.

15. The method of claim **12** wherein the step of performing a current synchronization further includes deleting records of the first database that were present during a previous synchronization and that, following the current synchronization, are outside of the current narrow date range.

16. The method of claim **12** wherein the step of performing a current synchronization further includes updating or deleting records of the first and second databases that are outside of the current narrow date range and within the narrow date range, based on the current synchronization.

17. The method of claim **12** wherein the step of performing a current synchronization further comprises, based on a selection by a user, performing one of:

(a) deleting the records of the first database that were present during a previous synchronization and that, following the current synchronization, are outside of the current narrow date range, and

(b) updating or deleting records of the first and second databases that are outside of the current narrow date range and within the narrow date range, based on the current synchronization.

18. The method of claim **12** wherein one of the narrow date range, the prior narrow date range, and the current narrow date range includes a start date and a stop date and the date of a record being synchronized is compared to the start date and the stop date to determine whether the record is within a corresponding one of the narrow date range, the prior narrow date range, and the current narrow date range.

19. The method of claim **18** wherein a record being synchronized includes a record start date and a record stop date, the method further comprising:

performing a comparison of the record start date to the stop date of one of the narrow date range, the prior narrow date range, and the current narrow date range, and of the record stop date to the start date of the corresponding one of the narrow date range, the prior narrow date range, and the current narrow date range; and

determining based on the comparison whether the record is within the corresponding one of the narrow date range, the prior narrow date range, and the current narrow date range.

20. The method of claim **12** wherein the current narrow date range comprises a relative narrow date range, the relative narrow date range being determined relative to a date of the current synchronization.

21. The method of claim **3** wherein the narrow date range comprises a relative narrow date range, the relative narrow date range being determined relative to a date of the current synchronization.

22. The method of claim **3** wherein the first database contains a first plurality of non-recurring records representing a plurality of recurring date bearing instances, the method further comprising:

generating a synthetic recurring record using the first plurality of non-recurring records;

performing a synchronization of the synthetic recurring record with a record of the second database;

if one of the record, the synthetic record, and a non-recurring record in the first plurality of non-recurring records is outside the narrow date range, fanning the synthetic record, or the record of the second database if it is a recurring record, into a second plurality of non-recurring records within the narrow date range.

23. The method of claim **3** wherein the narrow date range comprises a concatenation of a first date range for the first database and a second date range for the second database.

24. The method of claim **3** wherein records representatives of the records of the first and second databases during the prior synchronization are stored in a history file, and wherein performing the current synchronization includes performing the synchronization using the history file.

25. The method of claim **3** wherein the user-specified date includes a date determined based on a user-specified criterion.

26. The method of claim **25** wherein the date determined based on the user-specified criterion includes a date of an instance of a recurring record.

27. The method of claim **3** further comprising obtaining from a user information to determine the narrow date range.

28. A computer implemented method of synchronizing at least a first and a second database, each one containing date bearing records, wherein the date comprises a user-specified date, the method comprising:

identifying date bearing records of the first database that are within a narrow date range narrower than a date range of the records of the first database; and

synchronizing at least one of the identified date bearing records of the first database with one of the date bearing records of the second database.

29. The method of claim **28** wherein the step of performing a current synchronization further includes deleting records of the first database that are outside of the narrow date range.

30. The method of claim **28** wherein the first database contains a recurring record, the method further comprising fanning the recurring record within the narrow date range.

31. The method of claim **28** wherein the first database contains a recurring record, further comprising:

fanning the recurring record within a useful portion of the narrow date range, the useful portion being determined by application of a preference based on a current date of the current synchronization.

32. The method of claim **28** wherein records representatives of the records of the first and second databases during a prior synchronization are stored in a history file, and wherein performing the current synchronization includes performing the synchronization using the history file.

33. The method of claim **32** wherein the prior synchronization was performed across a prior narrow date range, such prior narrow date range being different from a current narrow date range of the current synchronization, and wherein performing the current synchronization further comprises performing the synchronization using the history file and the prior narrow date range.

34. The method of claim **33** wherein the narrow date range is a concatenation of the current narrow date range and the prior narrow date range stored in a history file during the prior synchronization.

35. The method of claim **33** wherein the step of performing a current synchronization further includes deleting records of the first database that were present during a previous synchronization and that, following the current synchronization, are outside of the current narrow date range.

36. The method of claim **33** wherein the step of performing a current synchronization further includes updating or deleting records of the first and second databases that are outside of the current narrow date range and within the narrow date range, based on the current synchronization.

37. The method of claim **28** wherein the current narrow date range comprises a relative narrow date range, the relative narrow date range being determined relative to a date of the current synchronization.

38. The method of claim **28** further comprising obtaining from a user information to determine the narrow date range.

39. The method of claim **28** wherein the user-specified date includes a date determined based on a user-specified criterion.

40. The method of claim **39** wherein the date determined based on the user-specified criterion includes a date of an instance of a recurring record.

41. A computer program, resident on a computer readable medium, for synchronizing at least a first and a second database each containing dated records such as events, wherein the records of the first database extend across a narrow date range narrower than the date range of the records of the second database, comprising instructions for:

performing a prior synchronization across a prior date range set using the date of the prior synchronization and the narrow date range;

storing the prior date range and a history file containing information representative of the content of the databases following the prior synchronization;

performing a current synchronization across a date range that combines the prior date range with a current date range set using the date of the current synchronization and the narrow date range.

42. The computer program of claim **41** wherein the date of each record being synchronized is compared to a start and stop date of a date range to determine whether the record is in range.

43. A computer program, resident on a computer readable medium, for synchronizing at least a first and a second database, each one containing date bearing records, wherein the date comprises a user-specified date, comprising instructions for:

identifying date bearing records of the first and second database that are within a narrow date range narrower than a date range of the records of one of the first and the second databases; and

performing a current synchronization across the narrow date range by synchronizing the identified date bearing records.

44. The computer program of claim **43** wherein the instruction for performing a current synchronization further comprises instructions for adding, modifying, or deleting records of the first database within the narrow date range.

45. The computer program of claim **43** wherein the instruction for performing a current synchronization further includes instructions for deleting records of the first database that are outside of the narrow date range.

46. The computer program of claim **43** wherein the narrow date range has a start date and a stop date and the date of a record being synchronized is compared to the start date and the stop date of the narrow date range to determine whether the record is within the narrow date range.

47. The computer program of claim **46** wherein the date of a record being synchronized includes a record start date and a record stop date, the computer program further comprises instructions for:

performing a comparison of the record start date to the stop date of the narrow date range and of the record stop date to the start date of the narrow date range;

determining based on the comparison whether the record is within the narrow date range.

48. The computer program of claim **43** wherein the first database contains a recurring record, the computer program further comprising instructions for fanning the recurring record within the narrow date range.

49. The computer program of claim **43** wherein the first database contains a recurring record, further comprising instructions for:

fanning the recurring record within a useful portion of the narrow date range, the useful portion being determined by application of a preference based on a current date of the current synchronization.

50. The computer program of claim **49** wherein the preference includes a preference for future dates compared to the current date over past dates compared to the current date.

51. The computer program of claim **49** wherein the preference includes a preference for dates closer to the current date over dates further from the current date.

52. The computer program of claim **43** wherein a prior synchronization was performed across a prior narrow date range, such prior narrow date range being different from a current narrow date range of the current synchronization, wherein records representatives of the records of the first and second databases during the prior synchronization are stored in a history file, and wherein performing the current synchronization further comprises instructions for performing the synchronization using the history file and the prior narrow date range.

53. The computer program of claim **52** wherein the narrow date range is a concatenation of the current narrow date range and the prior narrow date range stored in a history file during the prior synchronization.

54. The computer program of claim **52** wherein the instruction for performing a current synchronization further comprises instructions for adding, modifying, or deleting records of the first database that are within the current narrow date range.

55. The computer program of claim **52** wherein the instruction for performing a current synchronization further includes instructions for deleting records of the first database that were present during a previous synchronization and that, following the current synchronization, are outside of the current narrow date range.

56. The computer program of claim **52** wherein the instruction for performing a current synchronization further includes instructions for updating or deleting records of the first and second databases that are outside of the current narrow date range and within the narrow date range, based on the current synchronization.

57. The computer program of claim **52** wherein the instruction for performing a current synchronization further comprises instructions for, based on a selection by a user, performing one of:

(a) deleting the records of the first database that were present during a previous synchronization and that, following the current synchronization, are outside of the current narrow date range, and

(b) updating or deleting records of the first and second databases that are outside of the current narrow date range and within the narrow date range, based on the current synchronization.

58. The computer program of claim **52** wherein one of the narrow date range, the prior narrow date range, and the current narrow date range includes a start date and a stop date and the date of a record being synchronized is compared to the start date and the stop date to determine whether the record is within a corresponding one of the narrow date range, the prior narrow date range, and the current narrow date range.

59. The computer program of claim **58** wherein a record being synchronized includes a record start date and a record stop date, the computer program further comprising instructions for:

performing a comparison of the record start date to the stop date of one of the narrow date range, the prior narrow date range, and the current narrow date range, and of the record stop date to the start date of the corresponding one of the narrow date range, the prior narrow date range, and the current narrow date range;

determining based on the comparison whether the record is within the corresponding one of the narrow date range, the prior narrow date range, and the current narrow date range.

60. The computer program of claim **52** wherein the current narrow date range comprises a relative narrow date range, the relative narrow date range being determined relative to a date of the current synchronization.

61. The computer program of claim **43** wherein the narrow date range comprises a relative narrow date range, the relative narrow date range being determined relative to a date of the current synchronization.

62. The computer program of claim **43** wherein the first database contains a first plurality of non-recurring records representing a plurality of recurring date bearing instances, the computer program further comprising instructions for:

generating a synthetic recurring record using the first plurality of non-recurring records;

performing a synchronization of the synthetic recurring record with a record of the second database;

if one of the record, the synthetic record, and a non-recurring record in the first plurality of non-recurring records is outside the narrow date range, fanning the synthetic record, or the record of the second database if it is a recurring record, into a second plurality of non-recurring records within the narrow date range.

63. The computer program of claim **43** wherein the narrow date range comprises a concatenation of a first date range for the first database and a second date range for the second database.

64. The computer program of claim **43** wherein records representatives of the records of the first and second databases during the prior synchronization are stored in a history file, and wherein the performing the current synchronization includes instructions for performing the synchronization using the history file.

65. The computer program of claim **43** wherein the user-specified date includes a date determined based on a user-specified criterion.

66. The computer program of claim **65** wherein the date determined based on the user-specified criterion includes a date of an instance of a recurring record.

67. The computer program of claim **43** further comprising instructions for obtaining from a user information to determine the narrow date range.

68. A computer program, resident on a computer readable medium, for synchronizing at least a first and a second database, each one containing date bearing records, wherein the date comprises a user-specified date, comprising instructions for:

identifying date bearing records of the first database that are within a narrow date range narrower than a date range of the records of the first database; and

synchronizing at least one of the identified date bearing records of the first database with one of the date bearing records of the second database.

69. The computer program of claim **68** wherein the instruction for performing a current synchronization further includes instructions for deleting records of the first database that are outside of the narrow date range.

70. The computer program of claim **68** wherein the first database contains a recurring record, the computer program further comprising instructions for fanning the recurring record within the narrow date range.

71. The computer program of claim **68** wherein the first database contains a recurring record, further comprising instructions for:

fanning the recurring record within a useful portion of the narrow date range, the useful portion being determined by application of a preference based on a current date of the current synchronization.

72. The computer program of claim **68** wherein records representatives of the records of the first and second databases during the prior synchronization are stored in a history file, and wherein the performing the current synchronization includes instructions for performing the synchronization using the history file.

73. The computer program of claim **72** wherein the prior synchronization was performed across a prior narrow date range, such prior narrow date range being different from a current narrow date range of the current synchronization, and wherein performing the current synchronization further comprises instructions for performing the synchronization using the history file and the prior narrow date range.

74. The computer program of claim **73** wherein the narrow date range is a concatenation of the current narrow date range and the prior narrow date range stored in a history file during the prior synchronization.

75. The computer program of claim **73** wherein the instruction for performing a current synchronization further includes instructions for deleting records of the first database that were present during a previous synchronization and that, following the current synchronization, are outside of the current narrow date range.

76. The computer program of claim **73** wherein the instruction for performing a current synchronization further includes instructions for updating or deleting records of the first and second databases that are outside of the current narrow date range and within the narrow date range, based on the current synchronization.

77. The computer program of claim **68** wherein the narrow date range comprises a relative narrow date range, the relative narrow date range being determined relative to a date of the current synchronization.

78. The computer program of claim **68** further comprising instructions for obtaining from a user information to determine the narrow date range.

79. The computer program of claim **68** wherein the user-specified date includes a date determined based on a user-specified criterion.

80. The computer program of claim **77** wherein the date determined based on the user-specified criterion includes a date of an instance of a recurring record.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 6,141,664
DATED : October 31, 2000
INVENTOR(S) : David J. Boothby

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Title page,
Item [56], **References Cited**, U.S. PATENT DOCUMENTS, please add the following references:

--	5,155,850	10/1992	Janis et al.	395/600
	5,278,982	01/1982	Daniels et al.	395/600
	5,560,005	09/1996	Hoover et al.	395/600
	5,613,113	03/1997	Goldring	395/618
	5,615,364	03/1997	Marks	395/618
	5,619,689	04/1997	Kelly	395/617
	5,630,081	05/1997	Rybicki et al.	395/948
	5,708,812	01/1998	Van Dyke et al.	395/712
	5,708,840	01/1998	Kikinis et al.	395/800
	5,745,712	04/1998	Turpin et al.	395/333
	5,790,789	08/1998	Suarez	395/200.32
	5,870,759	02/1999	Bauer et al.	707/201
	5,870,765	02/1999	Bauer et al.	707/203
	5,884,323	03/1999	Hawkins et al.	707/201
	5,884,324	03/1999	Cheng et al.	707/201
	5,884,325	03/1999	Bauer et al.	707/201
	5,926,824	07/1999	Hashimoto et al.	707/520 --

Drawings,
FIG. 30, line 1059, "marked as a a" should be -- marked as a --.

Column 3,
Line 11, "FIG. 4" should be -- FIG. 4a-4b --.
Line 13, "FIG. 5" should be -- FIG. 5a-5b --.
Line 35, "FIG. 16" should be -- FIG. 16a-16b --.
Line 41, "FIG. 19" should be -- FIG. 19a-19b --.
Line 49, "FIG. 25" should be -- FIG. 25a-25b --.
Line 51, "FIG. 26" should be -- FIG. 26a-26d --.
Line 53, "FIG. 27" should be -- FIG. 27a-27b --.
Line 61, "pseudocoe" should be -- pseudocode --.
Line 61, "loadin" should be -- loading --.

Column 5,
Line 9, "FIG. 4" should be -- FIG. 4a-4b --.

Column 6,
Line 57, "FIG. 5" should be -- FIG. 5a-5b --.

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 6,141,664
DATED : October 31, 2000
INVENTOR(S) : David J. Boothby

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 9,
Line 35, "A→B Map" should be -- A→B_Map --.

Column 10,
Line 8, "Application" should be -- Applications --.

Column 13,
Line 13, "FIG. 16" should be -- FIG. 16a-16b --.

Column 15,
Line 3, "FIG. 19" should be -- FIG. 19a-19b --.
Line 60, "use" should be -- user --.
Line 66, after "updating", insert -- of --.

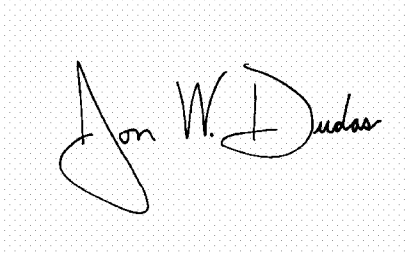
Column 17,
Line 18, "FIG. 25" should be -- FIG. 25a-25b --.
Line 21, "FIG. 26" should be -- FIG. 26a-26d --.
Line 62, "FIG. 27" should be -- FIG. 27a-27b --.

Column 18,
Line 34, delete the second occurrence of "does not".
Line 39, "truned" should be -- turned --.

Column 22,
Line 12, "FIG. 31" should be -- FIG. 31a-31b --.

Signed and Sealed this

Third Day of May, 2005



JON W. DUDAS
Director of the United States Patent and Trademark Office

US006141664C1

(12) EX PARTE REEXAMINATION CERTIFICATE (5235th)

United States Patent
Boothby

(10) Number: US 6,141,664 C1
(45) Certificate Issued: Nov. 22, 2005

(54) SYNCHRONIZATION OF DATABASES WITH DATE RANGE

(75) Inventor: **David J. Boothby**, Nashua, NH (US)

(73) Assignee: **Intellisync Corporation**, San Jose, CA (US)

Reexamination Request:
No. 90/006,583, Apr. 3, 2003

Reexamination Certificate for:

Patent No.:	**6,141,664**
Issued:	**Oct. 31, 2000**
Appl. No.:	**08/748,645**
Filed:	**Nov. 13, 1996**

Certificate of Correction issued May 5, 2005.

(51) **Int. Cl.**[7] **G06F 17/30**
(52) **U.S. Cl.** **707/201**; 707/203
(58) **Field of Search** 707/201, 203, 707/104.1, 200, 202, 8, 10

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,162,610 A		7/1979	Levine	
4,807,154 A		2/1989	Scully et al.	
4,807,155 A		2/1989	Cree et al.	
4,817,018 A		3/1989	Cree et al.	
4,819,191 A		4/1989	Scully et al.	
4,831,552 A		5/1989	Scully et al.	
4,866,611 A		9/1989	Cree et al.	364/300
4,939,689 A		7/1990	Davis et al.	
5,124,912 A		6/1992	Hotaling et al.	
5,134,564 A		7/1992	Dunn et al.	
5,155,850 A	*	10/1992	Janis et al.	707/202
5,197,000 A		3/1993	Vincent	
5,201,010 A	*	4/1993	Deaton et al.	382/139
5,204,958 A	*	4/1993	Cheng et al.	707/102
5,220,540 A		6/1993	Nishida et al.	
5,276,876 A		1/1994	Coleman et al.	
5,278,982 A	*	1/1994	Daniels et al.	707/202
5,323,314 A		6/1994	Baber et al.	
5,327,555 A		7/1994	Anderson	707/201
5,392,390 A		2/1995	Crozier	345/762
5,412,801 A		5/1995	De Remer et al.	
5,421,012 A		5/1995	Khoyi et al.	
5,455,945 A		10/1995	VanderDrift	
5,530,853 A		6/1996	Schell et al.	
5,530,861 A	*	6/1996	Diamant et al.	705/8
5,530,939 A		6/1996	Mansfield, Jr. et al.	

(Continued)

OTHER PUBLICATIONS

Terry et al. "Managing Update Conflicts in Bayou, a Weakly Connected Replicated Storage System." Procs. of the Fifteenth ACM Symposium on Operating Systems Principles, pp. 172–182, Dec. 1995. ACM Press.*

Adly, "HARP: A Hierarchical Asynchronous Replication Protocol for Massively Replicated Systems," Computer Laboratory, Cambridge University, United Kingdom (undated).

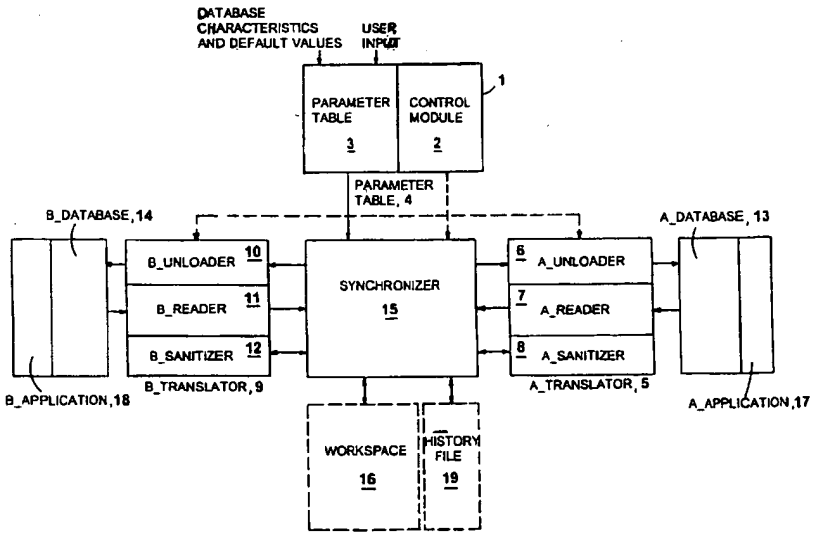
Adly et al., "A Hierarchical Asynchronous Replication Protocol for Large Scale Systems," Computer Laboratory, Cambridge University, United Kingdom, Computer Science Department, Alexandria University, Egypt (undated).

(Continued)

Primary Examiner—Safet Metjahic

(57) **ABSTRACT**

A method of synchronizing at least a first and a second database each containing dated records such as events, wherein the records of the first and second databases are synchronized across a narrow date range narrower than the date range of the records of at least one of the databases. A prior synchronization can be performed across a prior date range. The date range of the prior synchronization is stored, along with the history file containing information representative of the content of the databases following the prior synchronization. When a current synchronization is performed, it is performed across a date range that combines the prior date range with the current date range.



U.S. PATENT DOCUMENTS

5,557,518	A	*	9/1996	Rosen	705/69
5,560,005	A	*	9/1996	Hoover et al.	707/10
5,581,753	A		12/1996	Terry et al.	
5,581,754	A		12/1996	Terry et al.	
5,596,574	A		1/1997	Perlman et al.	
5,613,113	A	*	3/1997	Goldring	707/202
5,615,109	A	*	3/1997	Eder	705/8
5,615,364	A	*	3/1997	Marks	707/202
5,619,689	A	*	4/1997	Kelly	707/201
5,623,540	A	*	4/1997	Morrison et al.	379/112.01
5,630,081	A	*	5/1997	Rybicki et al.	715/839
5,649,182	A	*	7/1997	Reitz	707/7
5,649,195	A	*	7/1997	Scott et al.	707/201
5,659,741	A	*	8/1997	Eberhardt	707/104.1
5,666,530	A	*	9/1997	Clark et al.	707/201
5,671,407	A		9/1997	Demers et al.	
5,689,706	A		11/1997	Rao et al.	
5,704,029	A		12/1997	Wright, Jr.	
5,706,452	A		1/1998	Ivanov	
5,706,509	A		1/1998	Man-Hak Tso	
5,708,812	A	*	1/1998	Van Dyke et al.	717/171
5,708,840	A	*	1/1998	Kikinis et al.	708/105
5,727,202	A	*	3/1998	Kucala	707/10
5,737,539	A	*	4/1998	Edelson et al.	705/3
5,745,712	A	*	4/1998	Turpin et al.	715/763
RE35,793	E	*	5/1998	Halpern	702/62
5,758,083	A		5/1998	Singh et al.	
5,758,337	A		5/1998	Hammond	707/6
5,781,908	A		7/1998	Williams et al.	
5,790,789	A	*	8/1998	Suarez	709/202
5,790,974	A	*	8/1998	Tognazzini	455/456.5
5,809,494	A		9/1998	Nguyen	
5,813,009	A	*	9/1998	Johnson et al.	707/100
5,813,013	A		9/1998	Shakib et al.	707/102
5,819,272	A		10/1998	Benson	
5,819,274	A		10/1998	Jackson, Jr.	
5,832,218	A		11/1998	Gibbs et al.	
5,832,489	A		11/1998	Kucala	
5,838,923	A		11/1998	Lee et al.	
5,845,293	A		12/1998	Veghte et al.	
5,857,201	A		1/1999	Wright, Jr. et al.	707/104.1
5,870,759	A	*	2/1999	Bauer et al.	707/201
5,870,765	A	*	2/1999	Bauer et al.	707/203
5,875,242	A	*	2/1999	Glaser et al.	379/201.05
5,877,760	A	*	3/1999	Onda et al.	345/784
5,884,323	A	*	3/1999	Hawkins et al.	707/201
5,884,324	A	*	3/1999	Cheng et al.	707/201
5,884,325	A	*	3/1999	Bauer et al.	707/201
5,892,909	A		4/1999	Grasso et al.	
5,897,640	A		4/1999	Veghte et al.	
5,924,094	A	*	7/1999	Sutter	707/10
5,926,816	A		7/1999	Bauer et al.	707/8
5,926,824	A	*	7/1999	Hashimoto	715/520
5,928,329	A		7/1999	Clark et al.	
5,943,676	A		8/1999	Boothby	
5,956,508	A		9/1999	Johnson et al.	
5,974,238	A	*	10/1999	Chase, Jr.	709/248
5,978,813	A		11/1999	Foltz et al.	
5,995,980	A		11/1999	Olson et al.	
6,018,303	A		1/2000	Sadeh	
6,044,381	A		3/2000	Boothby	
6,098,078	A		8/2000	Gehani et al.	
6,253,188	B1	*	6/2001	Witek et al.	705/14
6,272,074	B1		8/2001	Winner	
6,324,542	B1		11/2001	Wright, Jr. et al.	
6,330,568	B1		12/2001	Boothby	
6,405,218	B1		6/2002	Boothby	

OTHER PUBLICATIONS

Alexander, "Designed, sold, delivered, serviced," Computerworld Client/Server Journal, p. 43 (Oct. 1, 1995).

"All I need is a miracle; computer-aided educational packages; Small Wonders," Coastal Associates Publishing L.P. (Mar. 1992).

Alonso et al., "Database System Issues in Nomadic Computing," Matsushita Information Technology Laboratory, New Jersey (undated).

Badrinath et al., "Impact of Mobility on Distributed Computations," Operating Systems Review (Apr. 1, 1993).

Barbara et al., "Sleeper and Workaholics: Caching Strategies in Mobile Environments (Extended Version)" (Aug. 29, 1994).

Bowen, "Achieving Throughput and Functionality in a Common Architecture: TheDatacycle Experiment," IEEE, p. 178 (Dec. 1991).

Brandel, "New offerings fuel revival of PIM," Computerworld, p. 39 (Sep. 12, 1994).

Demers et al., "The Bayou Architecture: Support for Data Sharing Among Mobile Users," Computer Science Laboratory, Xerox Palo Alto Research Center, California (undated).

DeVoe et al., "SOFTWARE: Day-Timer Organizer 2.0 based on format of paper-based PIM," InfoWorld, vol. 17 (Aug. 21, 1995).

Froese, "File System Support for Weakly Connected Operation," pp. 229–238 (undated).

Greenberg et al., "Real Time Groupware as a Distributed System: Concurrency Control and its Effect on the Interface," Procs. of the ACM CSCW Conf. On Computer Supported Cooperative Work, Oct. 22–26, North Carolina, ACM Press (Jan. 1, 1994).

Guy, "Ficus: A Very Large Scale Reliable Distributed File System," Technical Report CSD-910018, Computer Science Dept. UCLA (Technical Report) (Jun. 3, 1991).

Guy et al., "Implementation of the Ficus Replicated File System," appeared in Procs. Of the Summer USENIX Conf., Anaheim, CA, pp. 63–71 (Jun. 1, 1990).

Haber, "Renegade PIMS," Computerworld, p. 109 (Dec. 12, 1994).

Hammer et al., "An Approach to Resolving Semantic Heterogeneity in a Federation of Autonomous, Heterogeneous Database Systems," Computer Science Department, University of Southern California (undated).

Hammer et al., "Object Discovery and Unification in Federated Database Systems," University of Southern California (undated).

HP and IntelliLink connect HP 95LX with HP NewWave; IntelliLink for the HP NewWave; product announcement, HP Professional (Aug. 1991).

"HP announces expanded memory version of palmtop PC, introduces 1-Megabyte HP 95LX and 1-Megabyte memory cards," Business Wire, Inc. (Mar. 4, 1992).

Huston et al., "Disconnected Operation of AFS," CITI Technical Report 93-3, Center for Information Technology Integration, University of Michigan (Jun. 18, 1993).

IBM Dictionary of Computing, Tenth Edition, 1993, pp. 268, 269, 31_.

IBM Dictionary of Computing, Tenth Edition, 1993, pp. 165, 268, 349, 370, 417.

IEEE Standard Dictionary of Electrical and Electronics Terms, Fourth Edition, 1988, p. 372, 368, 509, 563.

Imielinski, "Mobile Computing—DataMan Project Perspective," Rutgers University (undated).

"IntelliLink 2.2: the software connection from desktop to palmtop; Software Review; IntelliLink 2.2; Evaluation," PC Magazine (Apr. 28, 1992).
"IntelliLink transfers palmtop, PC data; communications software from IntelliLink Inc; brief article; Product Announcement," PC Week (Nov. 18, 1991).
Jacobs et al., "A Generalized Query–by–Example Data Manipulation Language Based on Database Logic," IEEE Transactions on Software Engineering, vol. SE–9, No. 1 (Jan. 1983).
Jenkins, "Users struggle with E–mail Woes," Computerworld, p. 97 (Oct. 24, 1994).
Johnson et al., "Hierarchical Matrix Timestamps for Scalable Update Propogation," submitted to the 10th Int. Workshop on Distributed Algorithms (Jun. 25, 1996).
Joshi et al., "A Survey of Mobile Computing Technologies and Applications," (Oct. 29, 1995).
Kistler et al., "Disconnected Operation in the Coda File System," School of Computer Science, Carnegie Melon University, Pennsylvania (undated).
Krill, "NETWORKING: Tech Update," InfoWorld, vol. 18 (Feb. 12, 1996).
Kumar et al., "Log–Based Directory Resolution in the Coda File System," School of Computer Science, Carnegie Melon University, Pennsylvania (undated).
Larson et al., "A Theory of Attribute Equivalence in Databases with Application to Schema Integration," IEEE Transactions on Software Engineering, vol. 15, No. 4, Apr. 1989.
Lomet, D., Using timestamping to optimize two phase commit; Parallel and Distributed Information Systems, 1993, Proc. Of the 2nd Int. Conf., Jan. 20–22, 1993, pp. 48–55.
Mannino et al., "Matching Techniques in Global Schema Design," IEEE 1984.
Marshall, "Product Reviews: Windows contact managers," InfoWorld, vol. 18 (Mar. 25, 1996).
McGoveran, "Distributed not yet delivered," Computerworld, p. 112 (Jun. 6, 1994).
Meckler Corporation, "Palmtop–to–desktop linkage software," Database Searcher (Jun. 1992).
Microsoft Press Computer Dictionary, Second Edition, 1994, p. 164.
Microsoft Press Computer Dictionary, Second Edition, 1994, pp. 105, 217, 227, 228.
Microsoft Press Computer Dictionary, Third Edition, 1997, pp. 194, 228, 234, 449.
Milliken, "Resource Coordination Objects: A State Distribution Mechanism," (DRAFT) (Dec. 10, 1993).
Nash, "Replication falls short," Computer world, p. 65 (Nov. 21, 1994).
Noble et al., "A Research Status Report for Adaptation for Mobile Data Access," School of Computer Science, Carnegie Melon University (undated).
"PackRat PIM gets older and wiser with Release 4.0; PIM update sports enhanced interface, greater ease of use," InfoWorld (Dec. 23, 1991).
"Palmtop PCs: power by the ounce; Hardware Review; overview of six evaluations of palm–top computers; includes related articles on Editor's Choices, suitability–to–task ratings, impressions by individual users; evaluation," PC Magazine (Jul. 1991).
"Pen–based PCs ready for prime time; includes related article on comparison of operating systems, list of vendors of pen–based products," PC–Computing (Nov. 1991).
Perera, "Synchronization Schizophrenia," Computerworld Client/Server Journal, p. 50 (Oct. 1, 1995).
Petersen et al., "Bayou: Replicated Database Services for World–wide Applications," Computer Science Laboratory, Xerox Palo Alto Research Center, California (undated).
"Product comparison: Atari Portfolio, Casio Executive BOSS, HP 95LX, Poqet PC, Psion series 3, Sharp Wizard," InfoWorld (Dec. 16, 1991).
"Product Comparison: Personal information managers," InfoWorld, vol. 17 (Aug. 7, 1995).
Radosevich, "Replication mania," Computerworld Client/Server Journal, p. 53 (Oct. 1, 1995).
Ratner et al., "The Ward Model: A Replication Architecture for Mobile Environments," Department of Computer Science, University of California (undated).
Reiher et al., "Peer–to–Peer Reconciliation Based Replication for Mobile Computers," UCLA (undated).
Reiher et al., "Resolving File Conflicts in the Ficus File System," Department of Computer Science, University of California (undated).
Ricciuti, "Object database server," InfoWorld, vol. 18 (Jan. 29, 1996).
"Riding the NewWave from PC to Palmtop: IntelliLink lets NewWave users transfer files," InfoWorld (Jun. 3, 1991).
Saltor et al., "Suitability of data models as canonical models for federated databases," Universitat Politecnica de Catalunya, Spain (undated).
Salzberg, B., Timestamping After Commit, Procs. Of the Third Int. Conf. On Parallel Distributed Information Systems, Sep. 28–30, 1994, pp. 160–167.
Satyanarayanan, "Coda: A Highly Available File System for a Distributed Workstation Environment," School of Computer Science, Carnegie Mellon University (undated).
Satyanarayanan, "Fundamental Challenges in Mobile Computing," School of Computer Science, Carnegie Mellon University (undated).
Satyanarayanan, "Mobile Information Access," IEEE Personal Communications, vol. 3, No. 1 (Feb. 1996).
Sherman, "Information Technology: 'What Software Should I Use to Organize My Life'," (undated).
Sheth et al., "A Tool for Integrating Conceptual Schemas and User Views," IEEE 1988.
Schilit et al., "The ParcTab Mobile Computing System," Xerox Palo Alto Research Center, California (undated).
SPI Database Software Technologies Record Displays: Record 2, Serial No. TDB0291.0094 and Record 4, Serial No. iets0901.0073 (undated).
Staten, "PowerMerge 2.0 ships; syncs moved filed," MacWEEK, vol. 8, p. 38(1) (Jan. 3, 1994).
Tait, Doctoral Thesis entitled "A File System for Mobile Computing," (Jan. 1, 1993).
Tolly, "Enhanced Notes 4.0 gets thumbs–up," Computerworld, p. 54 (Dec. 18, 1995).
Webster's Ninth New Collegiate Dictionary, 1986, pp. 114, 436, 440, 462, 573, 597, 620, 717, 906, 963, 979, 989, 1000, 1053, 1130, 1142, 1152, 1162, 1166.
Wiederhold, Gio, Database Design, Second Edition, McGraw–Hill Book Company, 1983, p. 2.
Wiederhold et al., "Consistency Control of Replicated Data in Federated Databases," IEEE, pp. 130–132 (Jul. 12, 1990).
Zaino, "Tapping the Top Values in PDAs—Personal digital assistants that sell for as little as $300 can put a PC in the palm of your hand. Get the scoop on 8 contenders," HomePC, p. 97 (Oct. 1, 1996).

Zisman et al., "Towards Inoperability in Heterogeneous Database Systems," Imperial College Research Report No. DOC 95/11 (Dec. 1, 1995).
Extended Systems' Preliminary Invalidity Contentions.
Extended Systems' First Supplemental Preliminary Invalidity Contentions.
Extended Systems, Inc.'s Preliminary Claim Constructions and Preliminary Indentification of Extrinsic Evidence.
Patent Local Rule 4–2 Preliminary Claim Constructions and Extrinsic Evidence.
Joint Claim Construction and Prehearing Statement.
Extended Systems' Second Supplemental Preliminary Invalidity Contentions [Re: Reexamination Requests for the '390, '664, and '529 Patents].
Pumatech, Inc.'s Opening Claim Construction Brief; Declaration of Marc David Peters in Support of Pumatech, Inc.'s Opening Claim Construction Brief.
Extended Systems, Inc.'s Responsive Claim Construction Brief; Declaration of Jordan Trent Jones in Support of Extended Systems, Inc.'s Responsive Claim Construction Brief.
Supplemental Declaration of Marc David Peters in Support of Pumatech, Inc.'s Reply Claim Construction Brief.
Pumatech's Revised [Proposed] Claim Construction Order.
Pumatech, Inc.'s Reply Claim Construction Brief.
Statement of Recent Decision.
Pumatech's [Proposed] Claim Construction Order.
Synchrologic's Preliminary Invalidity Contentions.
Extended Systems' Final Invalidity Contentions (Oct. 10, 2003).
Defendant and Cross–Complainant Extended Systems, Inc.'s Identification of Prior Art Publications Pursuant to Patent L.R. 3–3(a) (Oct. 17, 2003).
Defendant and Cross–Complainant Extended Systems, Inc.'s Amended Identification of Prior Art Publications Pursuant to Patent L.R. 3–3(a) (Oct. 31, 2003).
Expert Report of John P. J. Kelly, Ph.D. (Oct. 24, 2003).
IntelliLink for Windows User's Guide, Version 3.0, IntelliLink Corporation (1993).
Database Subsetting Tool: Introduction to DST and DST Designer's Guide, Syware, Inc. (1993).
Sarin, "Robust Application Design in Highly Available Distributed Databases," Proc. 5th Symp. Reliability in Distributed Software and Database Systems, pp. 87–94 (Jan. 13–15, 1986, Los Angeles).
Distributed Management of Replicated Data: Final Report, Computer Corporation of America (Oct. 9, 1984).
Sarin et al., "Overview of SHARD: A System for Highly Available Replicated Data", Computer Corporation of America (Apr. 8, 1988).
SRI Int'l , Network Reconstitution Protocol, RADC–TR–87–38, Final Technical Report (Jun. 1987).
Danberg, "A Database Subsetting Tool" (patent application) (Apr. 12, 1993).
Lamb et al., "The Objectstore Database System," Communications of the ACM, vol. 34, No. 10, pp. 50–63 (Oct. 1991).
TT Interchange, Time Technology, AVG Sales & Marketing Ltd. (1995).
Goldberg et al., "Using Collaborative Filtering to Weave an Information Tapestry," Communications of the ACM, vol. 35, No. 12, pp. 61–70 (Dec. 1992).
Now Up–to–Date Version 2.0 User's Guide, Now Software, Inc. (1992).
An Introduction to DataPropagator Relational Version 1, IBM Corporation (1993).
Data Propagator Relational Guide Release 1, IBM Corporation (May 1994).
DataPropagator Relational Guide Release 2, IBM Corporation (Dec. 1994).
DataPropagator NonRelational MVS/ESA Version 2 Utilities Guide, IBM Corporation (Jul. 1994).
DPROPR Planning and Design Guide, IBM Corporation (Nov. 1996).
DataPropagator Relational Capture and Apply/400 Version 3, IBM Corporation (Jun. 1996).
DataPropagator Relational Capture and Apply for OS/400 Version 3, IBM Corporation (Nov. 1996).
Newton Connection Utilities User's Manual for the Macintosh Operating System, Apple Computer, Inc. (1996).
Newton Connection Utilities User's Manual for Windows, Apple Computer, Inc.
Newton Connection Utilities User's Manual for Macintosh, Apple Computer, Inc.
Newton Backup Utility User's Guide for the Windows Operating System, Apple Computer, Inc. (1995).
Newton Backup Utility User's Guide for the Macintosh Operating System, Apple Computer, Inc. (1995).
Newton Utilities User Manual, Apple Computer, Inc. (1995).
FileMaker Pro Server Administrator's Guide, Claris Corporation (1994).
Connectivity Pack User's Guide for the HP 200LX and the HP 100LX, Hewlett Packard.
Lotuc cc:Mail Release 2, Lotus Development Corporation (1991–1993).
User's Guide Lotus Organizer Release 1.0, Lotus Development Corporation (1992).
FileMaker Pro User's Guide, Claris Corporation (1990, 1992).
Poesio et al., "Metric Constraints for Maintaining Appointments: Dates and Repeated Activities".
Slater, "Newton's Legacy; 3COM and Microsoft Battle for Market Share; Apple Newton, 3Com Palm III, Microsoft Palm–size PC peronal digital assistants; Product Information", Information Access Company (1998).
Negrino, "ACT 2.5.1, ACT for Newton 1.0", UMI, Inc. (1996).
Zilber, "Toy story; personal digital assistants; Product Information", Information Access Company (1996).
Wingfield, "Desktop to Newton connectivity", UMI, Inc. (1996).
"Now Software Announces Updated Synchronization Software for Newton 2.0 Devices; Now Synchronize Simultaneously Updates MessagePad, Now Up–to–Date & Contact", Business Wire, Inc. (1995).
"Claris Ships FileMaker Pro 3.0 for Macintosh and Windows", Business Wire, Inc. (1995).
Alsop, "Distributed Thinking; Realizing the gravity of its PDA problems, Apple has drawn me back to Newton", InfoWorld Media Group (1995).
Rubin, "Now Software stays in sync; Now Synchronize file synchronization software for Macs and Newton PDAs; Software Review; EvaluationBrief Article", Information Access Company (1995).
"Now Calendar/Scheduler/Contact Mgr for Mac Update", Post–Newsweek Business Information Inc. (1995).

Staten, "csInStep middleware lets Newton talk to PIMs; Concierge Software LC's csInStep; Brief Article; Product Announcement; Brief Article", Information Access Company (1995).
Baum, "Designing Moble applications; A new approach needed for on–the–road systems", InfoWorld Media Group (1994).
Parkinson, "Remote users get in sync with office files; News Analysis", Information Access Company (1994).
Informix Guide to SQL Tutorial Version 7.1, Dec. 1994.
Oracle7 Distributed Database Technology and Symmetric Replication, Oracle White Paper, Apr. 1995.
Oracle7 Server Distributed Systems, vol. II: Replicated Data, Release 7.3, Feb. 1996.
Oracle7™ Server SQL Manual Release 7.3, Feb. 1996.

* cited by examiner

EX PARTE REEXAMINATION CERTIFICATE ISSUED UNDER 35 U.S.C. 307

THE PATENT IS HEREBY AMENDED AS INDICATED BELOW.

Matter enclosed in heavy brackets [ ] appeared in the patent, but has been deleted and is no longer a part of the patent; matter printed in italics indicates additions made to the patent.

ONLY THOSE PARAGRAPHS OF THE SPECIFICATION AFFECTED BY AMENDMENT ARE PRINTED HEREIN.

Column 18, lines 22–35:

In the third scenario, the record which was previously fanned should now be fanned also. The opposing database's record in this scenario is also fanned instances. This is perhaps the most peculiar of the three cases. For example, a database may be able to handle multi-day (i.e. daily recurring) records but not any exclusion dates for such items. Such database may be synchronized with another database which fans all records in the following manner. A

record representing a 7-day vacation in the Planner section of the database is fanned out to form 7 individual vacation days in the other database. One instance is deleted in the other database. Upon synchronizing the two databases, b/c the first database**[**s**]** does not provide for exclusion lists, the record must now be fanned.

THE DRAWING FIGURES HAVE BEEN CHANGED AS FOLLOWS:

The extraneous "a" at the end of line 1059 has been deleted.

AS A RESULT OF REEXAMINATION, IT HAS BEEN DETERMINED THAT:

The patentability of claims **3–6**, **8–10**, **12**, **14**, **15**, **20–22**, **24–33**, **35**, **37–40**, **43–46**, **48–50**, **52**, **54**, **55**, **60–62**, **64–73**, **75** and **77–79** is confirmed.

Claims **1**, **2**, **7**, **11**, **13**, **16–19**, **23**, **34**, **36**, **41**, **42**, **47**, **51**, **53**, **56–59**, **63**, **74** and **76** are cancelled.

Claim **80** is determined to be patentable as amended.

80. The computer program of claim **[77]** *79* wherein the date determined based on the user-specified criterion includes a date of an instance of a recurring record.

* * * * *

EXHIBIT D

US006445932B1

(12) **United States Patent**
Soini et al.

(10) **Patent No.: US 6,445,932 B1**
(45) **Date of Patent: Sep. 3, 2002**

(54) **MULTI-SERVICE MOBILE STATION**

(75) Inventors: **Veli-Matti Soini**; **Markku Rautiola**; **Jarmo J Mäkelä**; **Toni Sormunen**, all of Tampere; **Harri Halminen**, Pirkkaley; **Jari Toivanen**, Tampere, all of (FI)

(73) Assignee: **Nokia Mobile Phones Ltd.**, Espoo (FI)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **08/802,715**

(22) Filed: **Feb. 19, 1997**

(30) **Foreign Application Priority Data**

Feb. 23, 1996 (FI) 960859

(51) **Int. Cl.**[7] **H04M 11/00**

(52) **U.S. Cl.** **455/556**; 455/566; 455/567; 455/575

(58) **Field of Search** 455/556, 557, 455/418, 74, 66, 90, 566, 567, 575

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,128,981	A	*	7/1992	Tsukamoto et al.	455/567
5,189,632	A	*	2/1993	Paajanen et al.	455/556
5,241,284	A		8/1993	Nyqvist et al.	330/297
5,257,257	A		10/1993	Chen et al.	370/18
5,291,542	A		3/1994	Kivari et al.	379/58
5,337,346	A		8/1994	Uchikura	379/58
5,375,230	A		12/1994	Fujimori	395/575
5,378,935	A		1/1995	Korhonen et al.	327/114
5,416,435	A		5/1995	Jokinen et al.	327/113
5,422,656	A		6/1995	Allard et al.	345/173
5,471,655	A		11/1995	Kivari	455/127
5,491,718	A		2/1996	Gould et al.	375/205
5,517,552	A	*	5/1996	Yamashida	455/556
5,570,369	A		10/1996	Jokinen	370/95.3
5,581,244	A		12/1996	Jokimies et al.	340/825.44
5,584,054	A	*	12/1996	Tyneski et al.	455/90
5,615,384	A	*	3/1997	Allard et al.	345/121
5,797,089	A	*	8/1998	Nguyen	455/557

FOREIGN PATENT DOCUMENTS

EP	0704788 A2		4/1996	
WO	WO 93/23932	*	11/1993	H04B/7/00
WO	WO 93/23933	*	11/1993	H04B/7/00

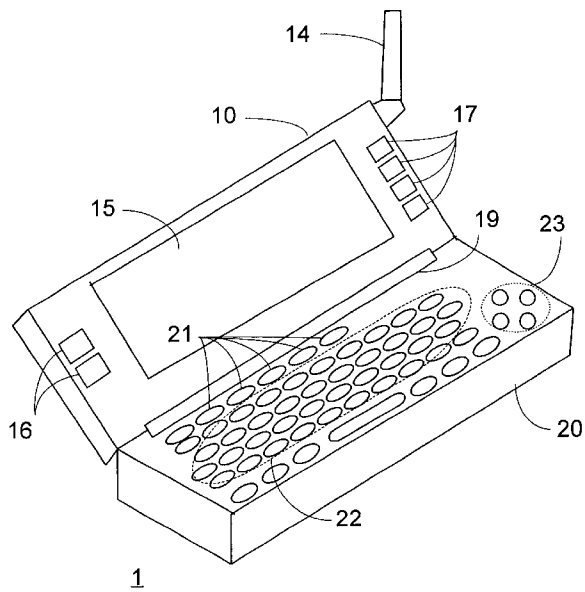
* cited by examiner

Primary Examiner—Lee Nguyen
(74) *Attorney, Agent, or Firm*—Perman & Green, LLP

(57) **ABSTRACT**

A multi-service mobile station comprises a module **(42)** for connecting the device by radio to a telecommunication network in order to utilize typical mobile station services, such as speech and data services. Additionally, the multi-service mobile station provides processing memory **(41, 40, 47)** for using various information processing services **(P1, P2)**, such as telefax service and electronic mail service. When using information processing services, the information processed by the user is automatically saved in the memory **(40, 47)** of the multi-service mobile station when a certain criterium is met. The criteria are, e.g., shifting from one service to another, going over to current saving mode, or, in a two-section multi-service terminal device, folding the device together. It is also characteristic of the multi-service mobile station disclosed, that the automatic data storing and current saving methods operate in a close-knit cooperation with operating system, resulting in the longest possible battery operating time for the multi-service mobile station.

18 Claims, 3 Drawing Sheets



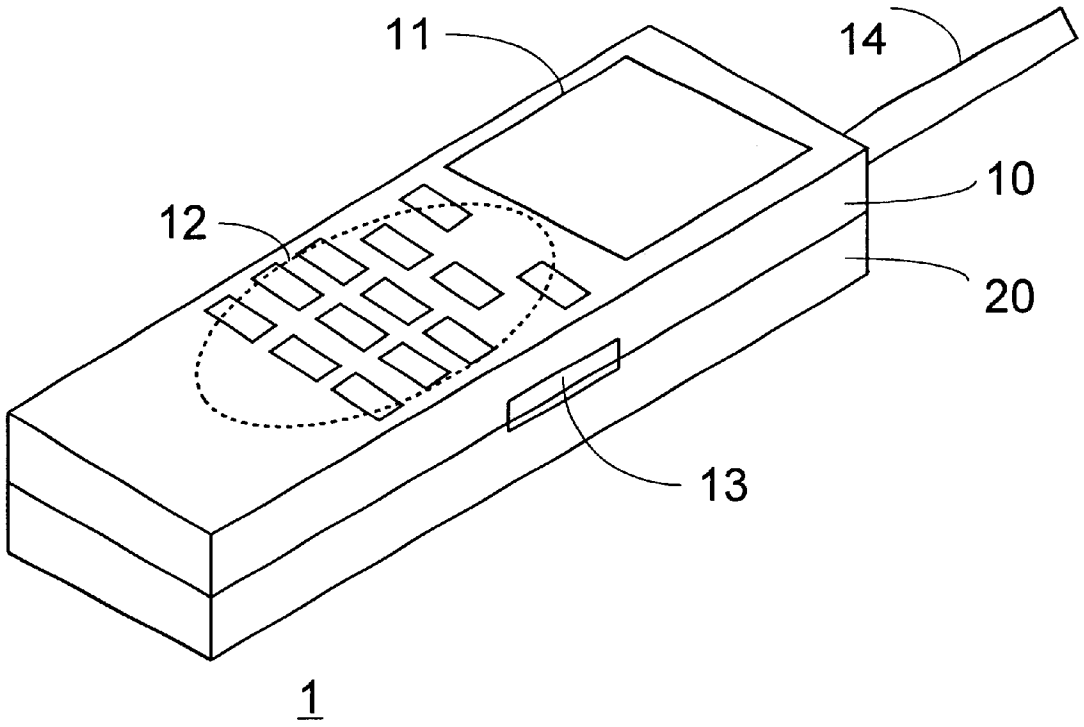


Figure 1

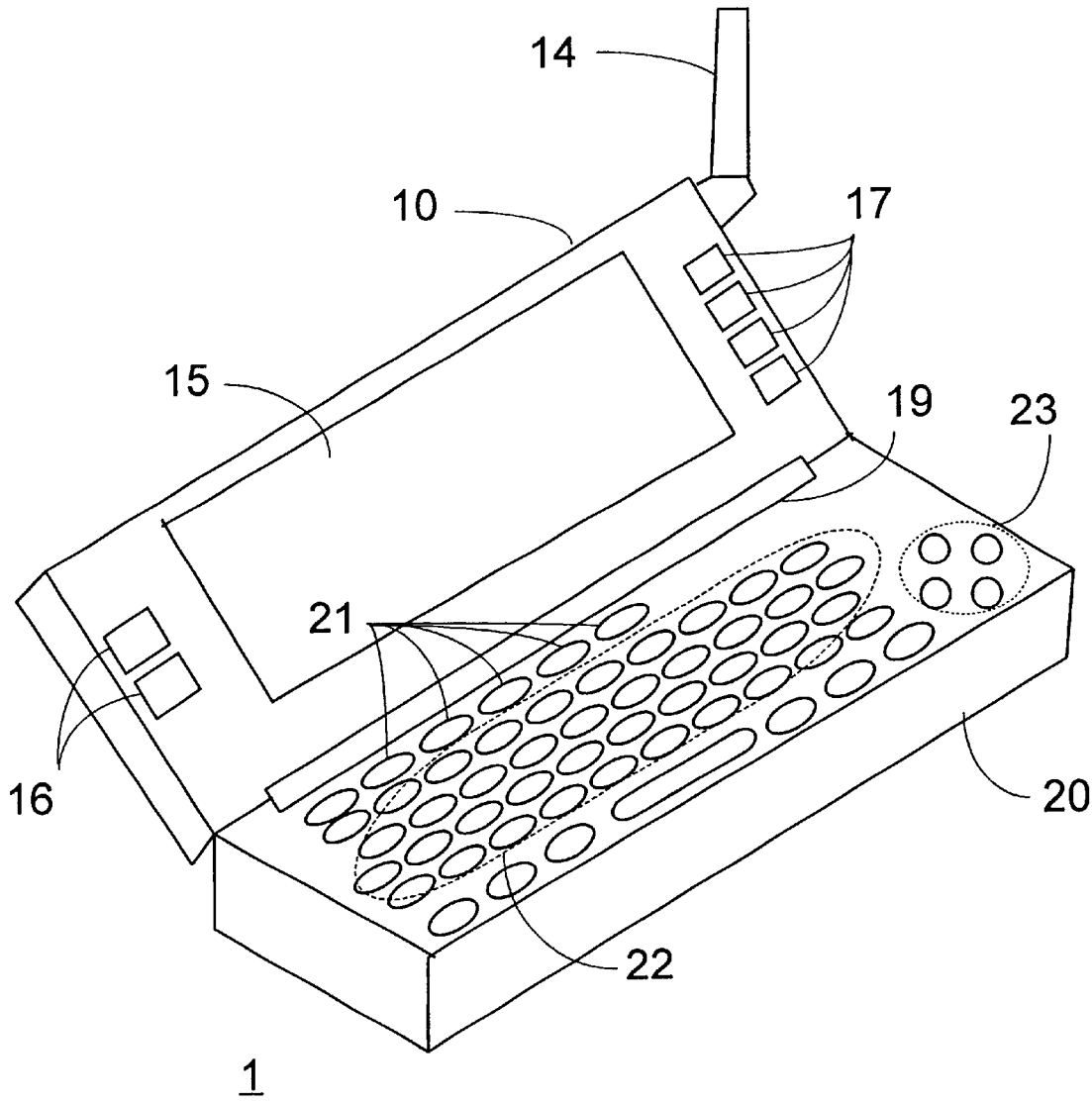


Figure 2

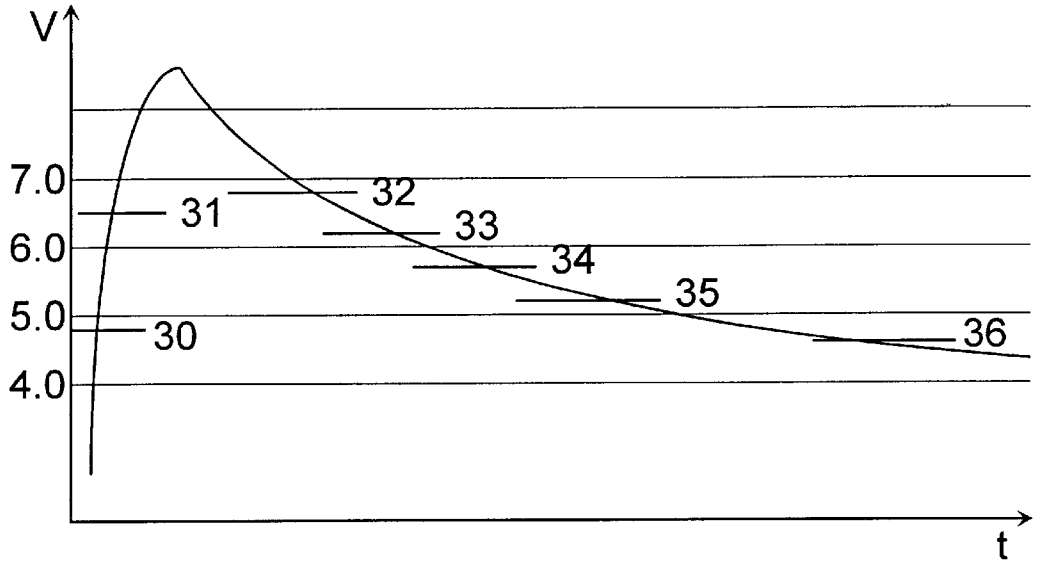


Figure 3

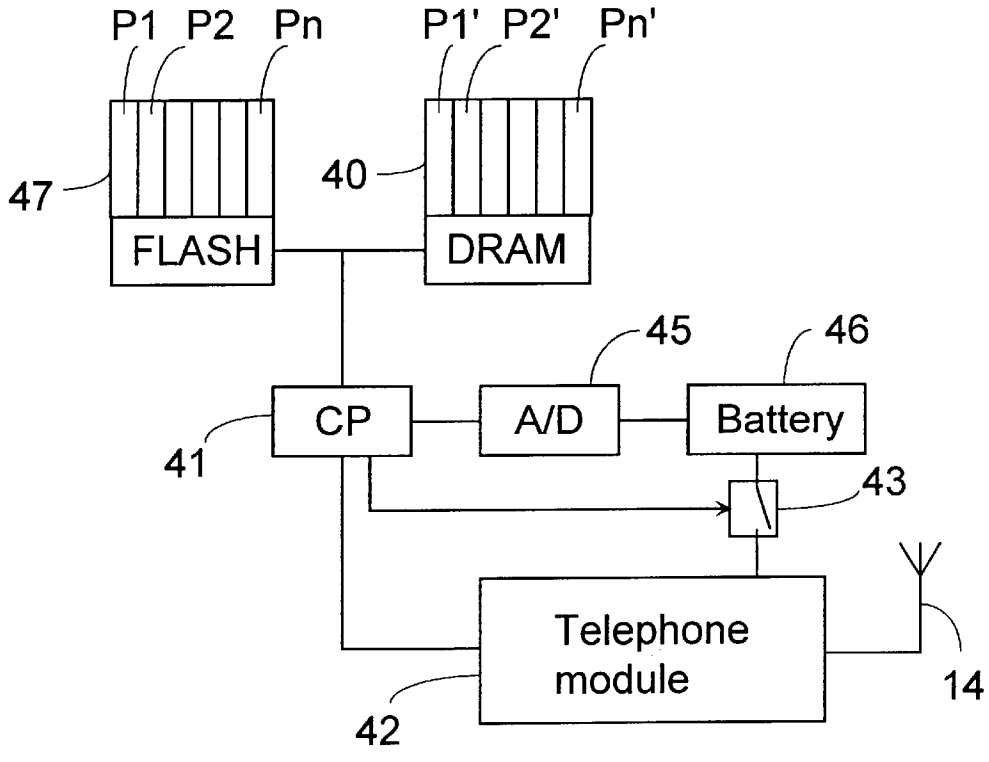


Figure 4

MULTI-SERVICE MOBILE STATION

FIELD OF THE INVENTION

The present invention is a multi-service mobile station, the data storing and power saving properties of which are optimized and which is connected to a mobile communication network by radio. In addition to the normal mobile station functions, such as speech and data services, said multi-service mobile station is suitable for a versatile data communication terminal. Said multi-service mobile station offers e.g. telefax (facsmile), calendar and notebook services and makes it possible to have a radio connection to other data communication networks, for example to Internet, for utilization of various services.

BACKGROUND OF THE INVENTION

In the modern information society people are more and more dependent on telecommunication networks and services offered by them. Utilizing telecommunication networks has been experienced to be so important that people no more want to be dependent on the bonds of traditional wired network. This is why a great number of people already have wireless mobile stations, e.g. GSM stations, which are mainly used for normal speech communication.

To a more increasing extent people have become dependent on even other telecommunication services available, such as telefax, electronic mail, Internet and other information transfer services. When one is used to these services in the traditional office environment, they are difficult to give up for example when leaving for a business trip.

In order to manage when on the road, as easily as possible, e.g. one's time and meetings, so called electronic notebooks have been developed. They are generally called PDA (Personal Digital Assistant), PCD (Personal Communication Device) and PIC (Personal Intelligent Communicator) devices. These devices are typically of the size of a largish pocket calculator and often equipped with a touch screen. The user can with a plastic tip pencil designed for this particular purpose, or even by touching the screen with fingers, write text and figures on the screen, from which the device interprets the information given. In addition to that, the user can control the functions of the device by touching menus which the device generates in the display. Typical services made possible by PDA devices are e.g. calendar and notebook services, reminding of agreed meetings by e.g. an acoustic signal, and a phone book from which information can be searched based upon a person's or company's name or other corresponding information. Additionally, PDA devices often also have a pocket calculator function.

When the features of a mobile station, a computer and a typical PDA device are integrated and packed in a very compact size, the result is a very versatile wireless data communication terminal. An example of a device comprising a touch screen known from PDA devices, DOS operating system known from computers, and a traditional mobile station, is presented in U.S. Pat. No. 5,422,656. With the device according to said patent publication it is possible to use various telecommunication services, e.g. to use it as a normal mobile telephone, to transmit and receive telefax messages or to use it as a pocket calculator.

The solutions presented in said US Patent yet have their disadvantages. Because the device is based on DOS operating system known from computers, the device must be at commissioning be initialized by loading the operating system in the processor of the device. After this, the first service, e.g. mobile telephone service, is loaded in a memory. When the mobile telephone service is initialized, it further initializes the necessary auxiliary components, and contacts a base station. All these initializing operations are time consuming and it takes quite a long time until the device is prepared for wireless communication after switch-on.

When shifting from one service to another, e.g. from said telefax service to said phone service, the user must first save the eventual information used by him in the previous service in order not to loose it when starting a new service. This is laborious and can easily be forgotten in a hurry. This will happen e.g. when the user is writing a telefax message and answers an incoming call too hurriedly. Alike when initializing the device for the first time, also all new services must be separately loaded in the memory of the device before they can be used. This downgrades the usability of services and does not offer a handy method for transferring information from one service to another. Neither does the operating system used make it possible to use several services simultaneously.

A very big disadvantage is also, that information which has not been stored, is lost when battery voltage drops too low, or if for some other reason power supply is cut off or disturbed. This happens e.g. if the battery gets loose or if there is poor contact in battery terminals.

Now a multi-service mobile station has been invented, with which at least part of the above mentioned problems can be avoided.

SUMMARY OF THE INVENTION

The multi-service mobile station according to the invention is always automatically ready for use when a sufficiently charged battery is connected. To enable this without reducing the operating time when battery powered, the multi-service mobile station is preferably provided with an advanced power saving automation. Moving from one service to another is executed either by selecting the required service using specific service keys or by touching the part of a touch screen which indicates the required service. When the user moves from one service to another, the information belonging to the previous service is stored automatically. This is the case e.g. when the user is using some service other than speech service at the moment when an incoming call is connected and preferably no information is lost even if the user moves directly to speech service.

Storing of information is carried out in such a way, that when the user selects a new service, the processor of the multi-service mobile station stores the user input and the status data connected with the previous service in the memory of the multi-service mobile station as a response to leaving a service. When the user for the next time returns to the same service, the processor of the multi-service mobile station retrieves, based upon said status data stored in the memory, the previous user situation of said service and presents it in the display of the multi-service mobile station. In this way the user can continue utilizing the service preferably directly from the situation, in which the service was when last used by the user.

Because the multi-service mobile station automatically stores the information processed by it, it is possible to set the multi-service mobile station in automatic answering mode, in which mode speech service or respectively telefax service is automatically activated upon an incoming message. Using other services during speech service is also possible.

The advanced automatic data storing and power saving methods take care of the storing of information also in case

of disturbances in power supply. If the state of battery charge goes down too much, or if e.g. the battery is disconnected from the multi-service mobile station, the information on all services and the processed information are stored automatically. All processed information is usable again when the power supply returns to normal.

The utilization of the services of the multi-service mobile station according to the invention is not limited only to services known from PDA devices and mobile stations, but it can also be equipped with an infrared transmitter-receiver, which enables utilizing the multi-service mobile station e.g. for transferring information between the multi-service mobile station and an external device, e.g. a personal computer, as a wireless pay terminal in shops, as a remote control device or as a key in security applications.

It is characteristic of the invention, that the multi-service mobile station comprises information storing means for automatically storing information related to the service in use, when a predetermined criterium is met.

There are several embodiments, different in their mechanical approach, for the multi-service mobile station according to the invention which utilizes automatic data storing methods and power saving methods. By enhancing the display and keyboard of a traditional mobile station it is possible to create a user interface large enough to provide the required services. This is the case in particular if the display of the multi-service mobile station is provided with a touch sensitive surface enabling using the display for receiving alphanumeric and graphic information. The above described solution is applicable if the volume of information to be processed by the multi-service mobile station is not very large, because the small size of the display and/or keyboard set their own limitations.

The multi-service mobile station according to the invention is described in detail in the following using one preferable embodiment, the mechanical solutions of which are different from traditional mobile stations. There are though no limitations to the utilization of the automatic data storing methods and power saving methods according to the invention in devices, the mechanical construction of which has a greater resemblance with traditional mobile stations. Said preferable embodiment enables the use of e.g. a larger display and a larger keyboard without an increase in the multi-service mobile station's main dimensions.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** presents one preferable embodiment of the multi-service mobile station according to the invention with its cover closed (mobile telephone position),

FIG. **2** presents one preferable embodiment of the multi-service mobile station according to the invention with its cover open (terminal mode position),

FIG. **3** presents limit values employed in automatic data storing and power saving automatics and

FIG. **4** presents the connections between the processing means and applications in the multi-service mobile station.

DETAILED DESCRIPTION

FIG. **1** presents one preferable embodiment of multi-service mobile station **1**, in which it has two folding sections which can be folded/unfolded. Multi-service mobile station **1** is in FIG. **1** presented in mobile telephone position, in which top section **10** and base section **20** have been folded together, supported by hinge **19** (FIG. **2**). The top side of cover section **10** comprises display **11** for displaying alphanumeric characters or graphics, and keyboard **12** for inputting alphanumeric characters in multi-service mobile station **1**. When multi-service mobile station **1** is in its mobile telephone position, cover section **10** and base section **20** are locked together with latch **13**. Antenna **14** is mounted in cover section **10** in the preferable embodiment according to the invention.

In FIG. **2** the same preferable embodiment of multi-service mobile station **1** is presented in its terminal (mode) position, with cover section **10** and base section **20** unfolded in a suitable angle, apart from each other, and supported by hinge **19**. In this position the inside of cover section **10** and the top side of base section **20** provide the user a user interface comprising display **15**, scroll keys **16** and function keys **17**, and base section **20** provides application keys **21**, QWERTY keys **22** and arrow keys **23**, both prior known from typewriters and computers. If needed, displays **11**, **15** can be equipped with a touch sensitive surface, making it possible to input information in multi-service mobile station **1** by touching display **11**, **15** with a specific pen or with fingers.

FIG. **3** presents the behavior of battery voltage as a function of time, while charging or discharging a battery. FIG. **3** especially presents the limit voltages employed for the automatic saving function and the power saving automation. These limit values are PowerOn **30**, RadioOK **31**, 1Warn(ing) **32**, 2Warn(ing) **33**, Save **34**, PowerOff **35** and BatteryProtection **36**.

FIG. **4** presents the internal structure of the multi-service mobile station **1** according to the invention. It comprises memory **40**, processor **41**, telephone module **42**, switch **43**, A/D-converter **45**, battery **46** and flash memory **47**. Additionally, the figure presents the segments of memory space in the flash memory allocated for each service P1 to Pn, in which segments the program codes of the application program for each service P1 to Pn are stored. The description uses references P1 to Pn also when referring to program codes executing services P1 to Pn. Additionally, FIG. **4** presents in memory **40** the segments of memory space, in which the information P1' to Pn' processed by the user is stored. The information P1' to Pn' is characteristic of services P1 to Pn.

In mobile telephone position, that is when the cover section **10** and base section **20** are folded together as shown in FIG. **1**, multi-service mobile station **1** operates in speech traffic like a traditional mobile telephone. Because now the parts of user interface used in terminal mode **15**, **16**, **17**, **21**, **22**, **23** are protected by cover section **10** and base section **20**, a user sees preferably only the simple user interface **11**, **12** which makes it much easier to use the device as a normal mobile telephone.

For example, an incoming call is normally answered using keyboard **12**. Differently from a conventional mobile telephone, in this preferable embodiment of multi-service mobile station **1**, an incoming call can be answered, in addition to keyboard **12**, also by opening multi-service mobile station **1** from mobile telephone position (FIG. **1**) to terminal position (FIG. **2**). If the incoming message is a call, speech service is activated automatically. If a telefax message or an electronic mail message is concerned, the corresponding service is activated automatically. The user can also, if wanted, disable the automatic answering at the opening of the cover section **10**, e.g. by selecting a suitable disable function, using user interface **11**, **12**, **15**, **16**, **17**.

If the user sets the multi-service mobile station **1** to automatic answering mode for telefax messages, short

messages, such as the SMS (Short Message Service) messages familiar from GSM mobile telephones and electronic mail services, said messages are stored in the memory means **40**, **47** of the multi-service mobile station **1**. Simultaneously the user can e.g. block incoming calls. Incoming messages are indicated, if so wished, by an acoustic signal and/or in display **11**, **15**. Multi-service mobile station **1** can transmit messages which have been saved in advance, such as telefax messages, automatically at a predetermined time, even if multi-service mobile station **1** is in the mobile telephone position (FIG. **1**). Receiving and transmitting messages and utilizing the data traffic services offered by the device is thus possible, even if multi-service mobile station **1** has been folded in the mobile telephone position.

If multi-service mobile station **1** is in the mobile telephone position and it is in speech mode, i.e. it is transmitting and receiving messages to/from the base station, the opening of multi-service mobile station **1** to mobile telephone position (FIG. **2**) does not disturb the ongoing transmission or reception of messages. If a normal call is in process, it is possible to go over to use multi-service mobile station **1** in "hands free" mode. When the user moves the multi-service mobile station **1** from his ear on a table and opens it to terminal position according to FIG. **2**, the built in microphone and loudspeaker (not shown in the figure) are activated. Because of the built in echo cancellation system, multi-service mobile station **1** can in speech service be used e.g. as a speaker phone, allowing several persons to participate in the discussion. The user can, if he wants, disable the automatic activation of said microphone and loudspeaker, e.g. by selecting an appropriate disable function from the menu of multi-service mobile station **1**, when he wants to keep the call secrecy. In this case said microphone and/or loudspeaker are activated individually using an appropriate key **16**, **17**, **21**. Simultaneously, while opening multi-service mobile station **1** to terminal position the microphone and loudspeaker, not shown in the figures, used in the mobile telephone position and located e.g. in the bottom surface of base section **20** of multi-service mobile station **1**, are switched off.

When the user closes multi-service mobile station **1** to the mobile telephone position (FIG. **1**) when a call is on, processor **41** of multi-service mobile station **1** stores the information, processed in the telecommunication services in memory **40** of multi-service mobile station **1**. Depending on the service P1 to Pn used, the processed information P1' to Pn' is stored in the segment of memory space allocated to it. Depending on the program codes P1 to Pn of the services P1 to Pn used, which program codes are, as mentioned above, located in the memory segments allocated for them in flash memory **47**, the information P1' to Pn' processed by the user, can be saved also in flash memory **47**. When closing the multi-service mobile station **1**, the microphone and loudspeaker,not shown in the figure, located for example in the base section **20** of multi-service mobile station **1**, are activated, and multi-service mobile station **1** operates again like a conventional mobile station. When the user wishes to end a call, this is done either in the mobile telephone position using user interface **11**, **12** or in the terminal position using user interface **15**, **16**, **17**, **21**, **22**, **23**.

In the multi-service mobile station **1** according to the invention, out of the means and functions of a portable computer, an electronic notebook and a mobile station, only those have been utilized which are necessary for the operation of multi-service mobile station **1**. By effective integration of the above mentioned means and functions, components can preferably be omitted, e.g. a floppy disk drive and a hard disk, enabling the implementation of the multi-service mobile station **1** according to the invention in a very compact size. The fewer components and the smaller size also facilitate clearly lower manufacturing cost. The highly integrated construction, in which the same processing devices **40**, **41**, **42**, **46**, **47** are utilized for providing the mobile station functions as well as telecommunication services, results in an entity which is easier to control. The easiness of control enables employing automatic data storing methods, which are handy for the user and improve the safekeeping of information. The highly integrated construction also enables combining the automatic saving methods with effective power saving methods, resulting in distinctly longer operating and stand-by times.

Multi-service mobile station **1** offers, in addition to speech and data services known from traditional mobile telephones, such as SMS (Short Message Service), also several additional services P1 to Pn. Such services are e.g. the possibility to transmit and receive telefax and electronic mail messages, the possibility to contact Internet, the possibility to use multi-service mobile station **1** as an electronic notebook, or even to utilize the special services of the telecommunication network, such as call forwarding or group call services. Using the familiar WWW (World Wide Web) pages known from Internet environment, it is also possible to utilize new services still under development, such as weather forecast and betting services.

These above mentioned services P1 to Pn are utilized by the user in the preferable embodiment according to the invention mainly with multi-service mobile station **1** folded in the terminal position shown in FIG. **2**. In order to have all services P1 to Pn offered by the multi-service mobile station **1** available immediately, without any delay, multi-service mobile station **1** remains preferably in stand by mode always when a sufficiently charged battery **46** is in place. This has been implemented so, that always when the voltage level of battery **46** exceeds a preset value, PowerOn **30** (FIG. **3**), the processor **41** (FIG. **4**) of multi-service mobile station **1** is activated and it starts the execution of the operating system in flash memory **40**. This method, prior known to people skilled in the art, in which a program is executed directly in the memory in which it is stored, is called execution in place (XIP, Execute in place). This method saves DRAM memory **40** for other purposes.

The application programs P1 to Pn offering the various services P1 to Pn, can be executed in the flash memory of multi-service mobile station **1**, alike the operating system (Execute In Place), but the processor may, based upon the information offered by application program P1 to Pn, load the required application programs P1 to Pn from flash memory **47** to memory **40**, in which the desired application programs also can be executed. This is a preferable feature for example when high read and/or write speed is required by an application program P1 to Pn which offers a certain service P1 to Pn. In memory **40** the application programs P1 to Pn are available for processor **41** without delay, until the battery voltage drops below a preset limit PowerOff **35** (FIG. **3**).

Flash memory **47** can be expanded according to a user's needs, and it facilitates updating of services P1 to Pn and adding new services even afterwards. Flash memory **47** can be expanded with an add-on module (not shown in the figure) which is connected to the PCMCIA bus.

When a user moves from a first service P1 to a second service P2, e.g. by pushing menu keys **21**, processor **41** of the multi-service mobile station **1** saves the information

from the display **15** of first service P1 and the user Input P1' in memory **40** of multi-service mobile station **1**. When the application program P1 so commands, which is preferable for securing the information, the user input is stored also in flash memory **47**. In order to secure user information, it is possible, according to commands given by service P1 to Pn, to store information from memory **40** to flash memory **47** also according to other criteria. The information can be stored e.g. at five minute intervals, or the storing process can be connected to the power saving automation. When connected to the power saving automation, the same criteria can be used e.g. for switching off the display.

In multi-service mobile station **1** according to the invention, memory **40** is preferably of DRAM (Dynamic Random Access Memory) type. SRAM (Static Random Access Memory) used widely in PDA equipment is expensive and consumes plenty of electricity. Using DRAM memory, the operating time of multi-service mobile station **1** has been increased. Another significant advantage is that the information processed by multi-service mobile station **1** is immediately readable from the DRAM memory. In this way said first service P1 and the information related to it and in its display **15** are ready for use in memory means **40**, **47** at once when the key assigned to said first service P1, e.g. menu key **21**, is pressed next time. This makes it possible, among other things, that when momentarily going over from typing a telefax message to e.g. the SMS service, and returning to the telefax service, typing the telefax message can be continued from the same point, in which the user was before going over to the SMS service. The status information of each service, such as which file it was that was open and at which point the user was editing said file, can for example be stored in the memory segment (P1' to Pn') allocated individually for each service. When each service is started, processor **41** of multi-service mobile station **1** reads the corresponding status data and initializes said status information of the service. Said first service P1 and second service P2 utilize preferably same memory media **40**, **47** making transferring information between different services easy and quick.

Said memory means **40**, **47** and services P1 to Pn are arranged in such a way that the desired information can be transmitted as a telefax message, as a SMS message known from GSM systems, or as electronic mail through Internet by pressing just one key **15**, **16**, **17**, **21**, **22**, **23**. The required telefax, telephone or electronic mail addresses are retrieved automatically by multi-service mobile station **1** from its memory means **40**, **47**, if they have been stored there in advance. The user simply e.g. writes the required message using QWERTY keyboard **22**, and after that just selects to whom and by which transfer method he sends the message. Respectively, the same message can easily be transmitted using several methods of transmission, e.g. as a SMS message to a mobile station and to a telefax in an office. If the address information is not found in the memory means, the user can input them individually for each transmission. The information to be transmitted is stored in memory means **40**, **47**, and when the user has selected the preferred transmission method, processor **41** of multi-service mobile station **1** executes the necessary processing operations, e.g. modifies text information to meet the requirements of telefax protocol. After this the required information is transferred by radio, utilizing the telephone module **42** of multi-service mobile station **1**, utilizing the data channel provided by the module. The function of telephone module **42** is alike that of a familiar mobile station, e.g. a GSM mobile telephone, known to a person skilled in the art.

In order to be able to control effectively the power saving and data storing automation of multi-service mobile station **1**, the processor **41** of multi-service mobile station **1** must know the charge status of battery **46**. The charge status of the battery is examined e.g. by monitoring the voltage over battery **46** during charging and discharging battery **46**. The voltage over battery **46** is measured employing analog/digital converter **45** connected to the poles of battery **46**. A/D converter **45** forwards to processor **41** the voltage over battery **46** in digital form with sufficient accuracy, typically some tens of millivolts.

The charge status of battery **46** is examined employing several limits **30** to **36** (FIG. **3**). If the voltage over well charged battery **46** drops below preset limit value **1**warning **32**, the processor **41** of the multi-service mobile station **1** notifies the user of the weakening of the charge status of the battery through display **11**, **15** or by an acoustic signal. If the voltage over battery **46** drops further below preset limit value **2**Warning **33**, multi-service mobile station **1** cuts off power supply to means **42**, which are connected with transmission and reception of messages by radio. This is carried out by employing switch **43** controlled by processor **41**. At the same time multi-service mobile station **1** outputs a second warning of the weakening battery charge status, and notifies that services requiring transferring messages by radio cannot be used until the battery has been recharged.

When the battery charge status further becomes weaker, or sufficient current supply is prevented for some other reason, so that voltage over the battery drops below preset limit value Store **34**, the processor **41** of multi-service mobile station **1** stores the information processed by all services in such memory means **47** of multi-service mobile station **1**, which are suitable for long time data storing and need no current. In this way no information is lost. When flash memory **47** is used, care must be taken, in order to secure the information storing, that the amount of unsaved data does not exceed the limit value characteristic of flash memory, typically approximately 1 kbyte, which is the largest amount of data when the fast storing method characteristic of flash memory is used. This requirement is easy to meet in application programs P1 to Pn, which are, due to their program being capable of measuring e.g. the amount of data, input by the user using QWERTY keyboard or the amount of data contained in an incoming telefax message.

If the voltage over battery **46** drops further below preset limit value PowerOff **35**, multi-service mobile station **1** cuts off power from even the section of multi-service mobile station **1** offering data traffic services and remains waiting for battery change or recharging. As a last precaution, battery **46** itself measures its own voltage and cuts off the voltage from its terminals when the battery cell voltage drops below preset limit value BatteryProtection **36**. The cutting off is executed by a built in switch (not shown in the figure) in battery **46**. When the voltage over the battery next time exceeds preset limit value PowerOn **30**, multi-service mobile station **1** is reactivated and services P1 to Pn are ready for use again.

In order to maximize the stand by and operation times, particular attention has been given to developing the power saving methods. In multi-service mobile station **1** such a power saving automation is employed, which is based upon a close-knit cooperation between hardware and operating system. In regard to its functions, multi-service mobile station **1** can be divided into two modules, telephone module **42** and service module. Telephone module **42** comprises means for connecting the multi-service mobile station **1** by radio to the telecommunication network. In addition to that,

display **11** and keyboard **12** belong to the telephone module. The service module comprises means for offering communication services, such as display **15**, and keyboards **16, 17, 21, 22, 23**. Both said modules employ partly or entirely same means of processing **40, 41, 46, 47**.

As stated above, the operating voltage is always on in the service module of multi-service mobile station **1**, provided that the battery voltage exceeds preset limit value PowerOn **30**. Correspondingly, the telephone module can be used for transmission and reception of messages if the battery voltage exceeds preset limit value RadioOK **31**. Because the level of RadioOK **31** is higher than PowerOn **30**, the service module is always active if messages are transferred by radio. This makes it possible for processor **41** of the service module to control also the operation of the power saving automation of the telephone module. If the service module wants to transfer messages by radio, the processor **41** of the service module activates telephone module **42** in order to assist in executing the service. Preferably, the RF module (not shown in the figure), which is comprised in telephone module **42** and consumes plenty of energy, is activated individually only when it is required because of transmission or reception of a message. Correspondingly, power supply to the RF module is cut off immediately when possible with regard to offering the service.

If the user is not using multi-service mobile station **1** when the first preset, typically time related criterium is met, the processor **41** of multi-service mobile station **1** shifts multi-service mobile station **1** to the first stage of power saving, when the clock signals of the processing media **41** are stopped in order to save current. Multi-service mobile station **1** is reactivated if the user touches any of keys **12, 16, 17, 21, 22, 23**, folds it to the mobile telephone position shown in FIG. **1** or unfolds it to the terminal position shown in FIG. **2**. If the user is not using multi-service mobile station **1** even when the second preset criterium is met, the processor **41** of multi-service mobile station **1** saves the information contained in the services P1 to Pn being used from memory **40** to memory means suitable for long time saving, e.g. flash memory **47**, which needs no power for storage of information. After this, processor **41** of multi-service mobile station **1** sets the system in deep power saving mode. Because all information processed in services P1 to Pn is stored in memory means **47**, there is no more any need to store the information in the DRAM based memory means, and the power consuming refresh signal of the DRAM memory circuits can preferably be switched off. Because all information processed is now stored in flash memory **47**, there is no need to monitor the voltage over battery **46**, because even if the voltage over battery **46** would drop further or even if battery **46** would be entirely removed, the information processed has been saved. This reduces power consumption for its share.

Multi-service mobile station **1** according to the invention is equipped with an infrared transmitter/receiver unit, which makes it possible for the multi-service mobile station **1** to communicate with external equipment. This enables using the multi-service mobile station **1** e.g. instead of a credit card or other payment card in payment applications for example in shops and gas stations. The infrared transmitter/receiver unit enables using the multi-service mobile station **1** also as a remote control device, as a key or as a means of identification in security applications, or as an interface to a wireless WLAN network (Wireless Local Area Network). A very convenient feature is also the data transfer between multi-service mobile station **1** and a computer, and for example printing out received telefax messages using a printer connected to a network. These services may be integrated into multi-service mobile station **1**, or they can be added to multi-service mobile station **1** using separate add-on modules.

Because, additionally, the multi-service mobile station **1** according to the invention has the data traffic interfaces of the future and enhancements possibilities are well catered for, the present and the future services can preferably be utilized effectively using multi-service mobile station **1**.

We have here presented the execution and some embodiments of the invention utilizing examples. It is obvious to a person skilled in the art that the invention is not limited to the details in the above presented embodiments and that the invention can be executed even in another embodiment without deviating from the characteristics of the invention. The above presented embodiments should be regarded as enlightening but not limiting. Thus the possibilities of executing and using the invention are limited only by the enclosed claims. Accordingly, the various embodiments of the invention specified in the claims, equivalent embodiments included, are covered by the invention.

What is claimed is:

1. A multi-service mobile communication station which comprises:

means for connecting said mobile station by radio to a telecommunication network for using information processing services; and

a volatile memory for holding information;

a user interface for selecting said information processing service between at least a first service and a second service, and for receiving at least one command from a user and for allowing the user to edit information held in the volatile memory in the information processing service selected;

processing means for processing information related to the service selected, in accordance with the information processing service selected, and for processing information according to said at least one command and the user's editing,

automatic storing means comprising a non-volatile memory for automatically storing in the non-volatile memory both information related to the service selected and the information edited by the user in response to meeting a predetermined criterion.

2. A multi-service mobile station according to claim **1**, wherein said user interface comprises means for displaying information, and said automatic storing means stores status information related to said first and second service used in said automatic storage means and the information in said display, when a predetermined criterion is met.

3. A multi-service mobile station according to claim **2**, wherein said processing means, responsive to when one of said first and second services is started, restores the status information related to said started service and the information which was in the display from said storage means, to the display of the multi-service mobile station in the form in which said status information and the information in the display were when said started service was last exited.

4. A multi-service mobile station to claim **3**, wherein said central processing unit is arranged to start the last active service and to restore the status information related to said last active service and the latest information in display from said memory means to said display of the multi-service mobile station, when said first section and second section are unfolded apart from each other.

5. A multi-service mobile station according to claim **1**, wherein said criterion is shifting from said first service over to said second service.

6. A multi-service mobile station according to claim 1, wherein the multi-service mobile station comprises means for measuring the operating voltage over battery and that said criterium is the dropping of said operating voltage below a certain, predetermined limiting value.

7. A multi-service mobile station according to claim 1, wherein said criterium is the moment when the multi-service mobile station is shifting to current saving mode.

8. A multi-service mobile station according to claim 1, wherein said criterion is the time elapsed since the previous storing in the non-volatile memory of both information related to the service selected and information edited by a user.

9. A multi-service mobile station according to claim 1, wherein the multi-service mobile station comprises means for using said first service and said second service simultaneously.

10. A multi-service mobile station according to claim 1, wherein said multi-service mobile station has at least two constructional sections, comprising a first section and a second section, which are arranged to fold in relation to each other utilizing a joint element, which is fixed to both sections, allowing the multi-service mobile station to be folded into a first position and a second position, and that said criterium is folding said first section and second section from said first position to second position or vice versa.

11. A multi-service mobile station according to claim 10, wherein said multi-service mobile station comprises a first user interface and a second user interface, and that when the multi-service mobile station is folded in a position in which said first section and second section draw apart, said first user interface is switched off and said second user interface is activated.

12. A multi-service mobile station according to claims 11, wherein said first user interface comprises a microphone and a loudspeaker for using the multi-service mobile station like a mobile station, and said second user interface comprises a microphone and a loudspeaker for using the multi-service mobile station like a hands-free telephone.

13. A multi-service mobile station according to claim 10, wherein said multi-service mobile station comprises a first user interface and a second user interface, and that when the multi-service mobile station is folded in a position in which said first section and second section fold against each other, said second user interface is switched off, and said first user interface is activated.

14. A multi-service mobile station which comprises:

means for connecting said mobile station by radio to a telecommunication network for using typical mobile communication services, said communication services comprising at least one of speech and data services, and for using information processing services; and

a user interface for selecting an information processing service between at least a first service and a second service, and wherein the improvement comprises information storing means comprising:

automatic storage means for automatically storing information related to the one of said first and second services in use, and

user information storage means for storing information processed by a user, when a predetermined criterion is met and said criterion is met when the amount of information fed by a user to the multi-service mobile station exceeds a predetermined limiting value.

15. A multi-service mobile station comprising:

a processor for processing information in an information processing service;

a radio receiver for receiving information for use of the information processing service;

a volatile memory for receiving the information from the radio receiver;

a user interface for receiving user input;

wherein the processor is configured to process the information in the information processing service according to the user input; and

a non-volatile memory;

wherein the processor has been configured to automatically store, in response to a predetermined criterion, from the volatile memory into the non-volatile memory the information processed by the processor.

16. A multi-service mobile station according to claim 13, wherein the predetermined criterion is that the amount of the user input exceeds a predetermined limiting value.

17. A multi-service mobile communication station having means for connecting said mobile station by radio to a telecommunication network for using mobile communication services which include speech and information retrieval services, said mobile communication station comprising:

a user interface for selecting among said mobile communication services, entering commands relating to said communication services, and for allowing the user to edit information obtained from said selected information retrieval services;

processing means for processing said information and commands entered through said user interface and controlling said information related to the service selected;

a volatile memory for storing information received from said selected service, including information edited through said user interface;

a non-volatile memory for storing information as directed by said processor;

wherein said processor automatically transfers information from said volatile memory to said non-volatile memory in response to the meeting of at least one predetermined criterion.

18. A multi-service mobile communication station having means for connecting said mobile station by radio to a telecommunication network for using mobile communication services which include speech and information retrieval services, said mobile communication station, as described in claim 17, wherein said at least one criterion is selected from the following events: communication service changed, volatile memory storage limits exceeded, and/or power depleted below a predetermined limit.

* * * * *

EXHIBIT E

US005898740A

United States Patent [19]

Laakso et al.

[11] **Patent Number: 5,898,740**

[45] **Date of Patent: Apr. 27, 1999**

[54] **POWER CONTROL METHOD IN A CELLULAR COMMUNICATION SYSTEM, AND A RECEIVER**

[75] Inventors: **Timo Laakso**; **Hannu Hakkinen**, both of Espoo, Finland

[73] Assignee: **Nokia Telecommunications OY**, Espoo, Finland

[21] Appl. No.: **08/793,259**

[22] PCT Filed: **Aug. 23, 1995**

[86] PCT No.: **PCT/FI95/00450**

§ 371 Date: **Feb. 21, 1997**

§ 102(e) Date: **Feb. 21, 1997**

[87] PCT Pub. No.: **WO96/07246**

PCT Pub. Date: **Mar. 7, 1996**

[30] **Foreign Application Priority Data**

Aug. 24, 1994 [FI] Finland 943889

[51] **Int. Cl.**[6] **H04B 1/10**

[52] **U.S. Cl.** **375/346**; 375/349; 455/522; 455/69; 455/574; 370/318

[58] **Field of Search** 375/200, 206, 375/219, 221, 285, 343, 346, 349; 455/67.1, 69, 422, 127, 13.4, 38.3, 522, 574, 92, 226.1, 226.2; 370/320, 342, 318, 319

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,470,138	9/1984	Gutleber .	
5,218,619	6/1993	Dent .	
5,224,120	6/1993	Schilling .	
5,363,403	11/1994	Schilling .	
5,553,062	9/1996	Schilling et al.	370/479
5,574,982	11/1996	Almgren et al.	455/69

FOREIGN PATENT DOCUMENTS

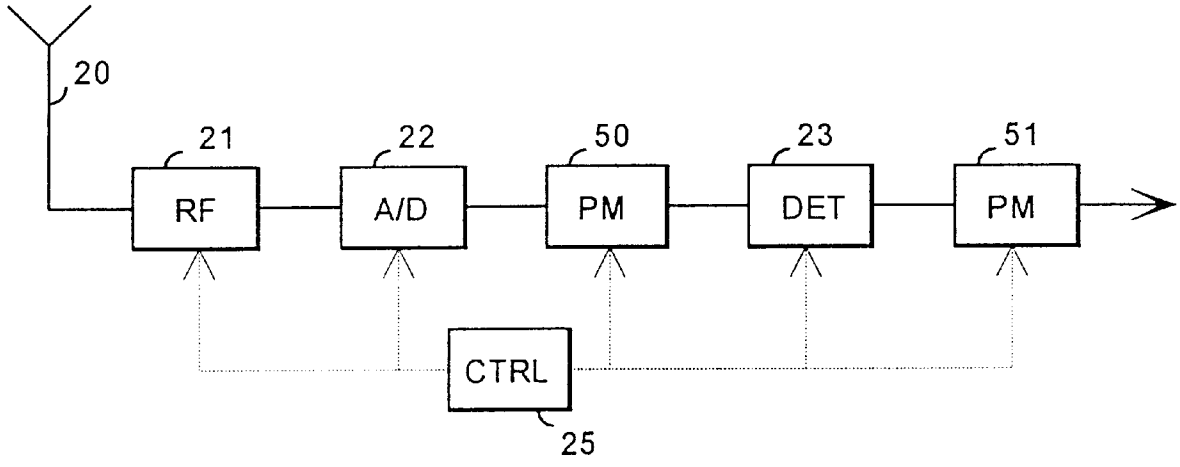
0 565 505	10/1993	European Pat. Off. .

Primary Examiner—Stephen Chin
Assistant Examiner—Amanda Le
Attorney, Agent, or Firm—Pillsbury Madison & Sutro LLP

[57] **ABSTRACT**

In a method for controlling transmission power in a cellular communication system which utilizes a technique for canceling multiple access interference, and in which a receiver controls the transmission power of a transmitter on the basis of a received signal, power control is carried out on the received signal after the received signal has been subjected to interference cancellation.

11 Claims, 3 Drawing Sheets



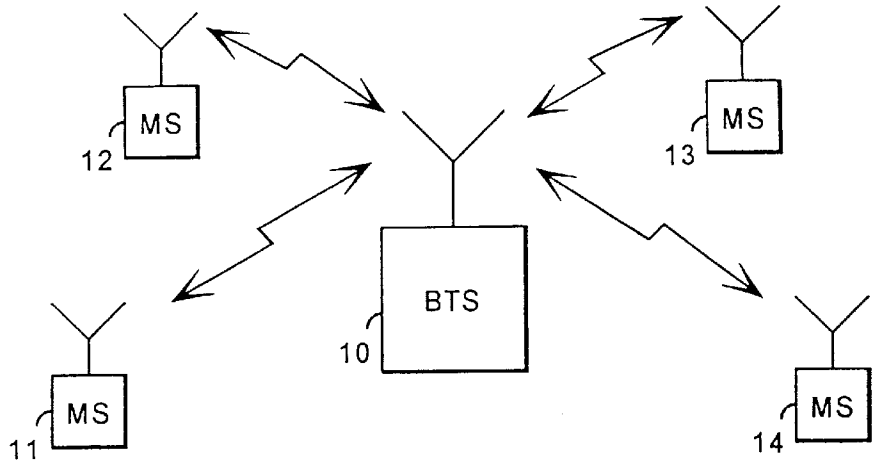

MS
12
MS
13
BTS
10
MS
11
MS
14

Fig. 1

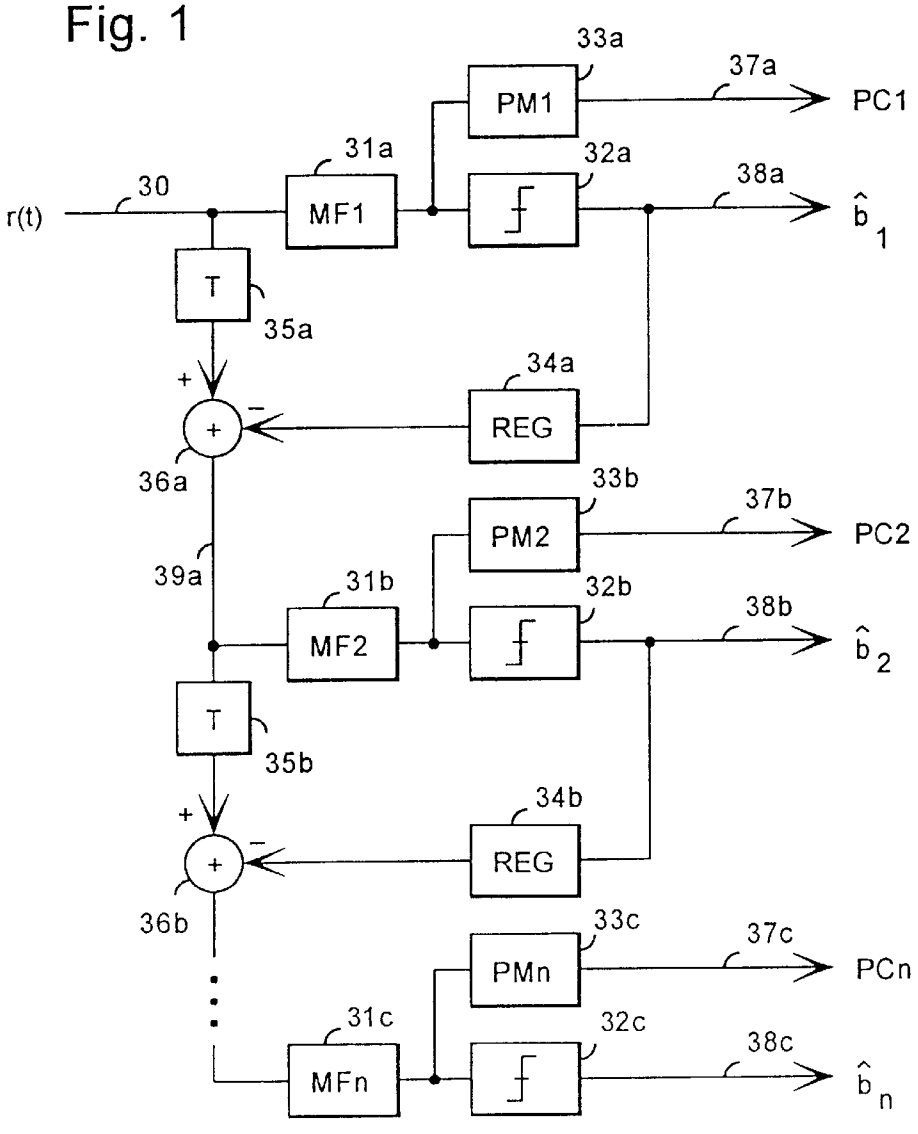

33a
37a
PM1
PC1
31a
32a
38a
r(t)
30
MF1
$\hat{b}_1$
T
35a
34a
+
−
REG
36a
33b
37b
PM2
PC2
39a
31b
32b
38b
MF2
$\hat{b}_2$
T
35b
34b
REG
36b
33c
37c
PMn
PCn
31c
32c
38c
MFn
$\hat{b}_n$

Fig. 3

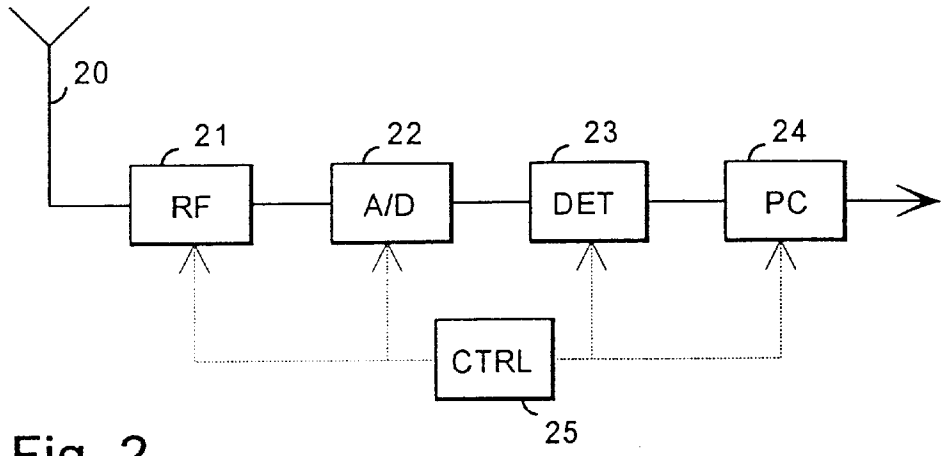

20
21
RF
22
A/D
23
DET
24
PC
CTRL
25

Fig. 2

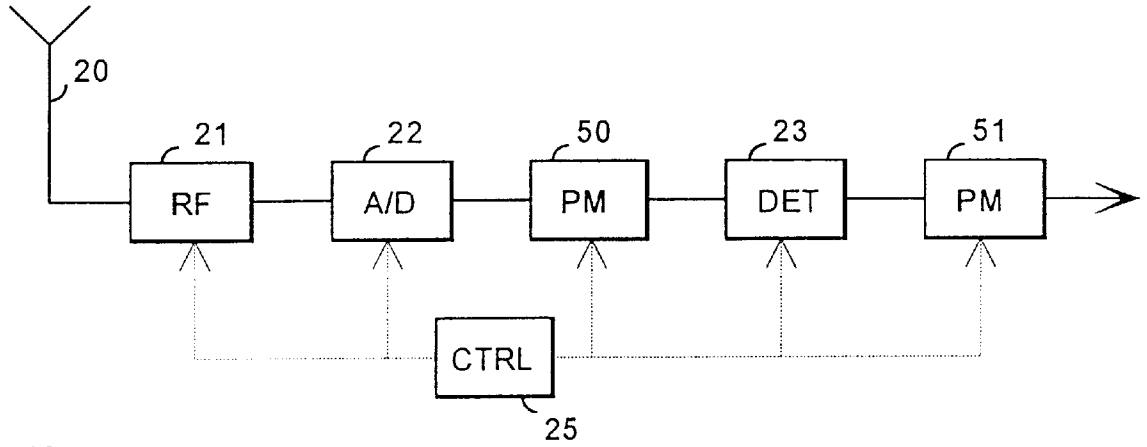

20
21
RF
22
A/D
50
PM
23
DET
51
PM
CTRL
25

Fig. 5

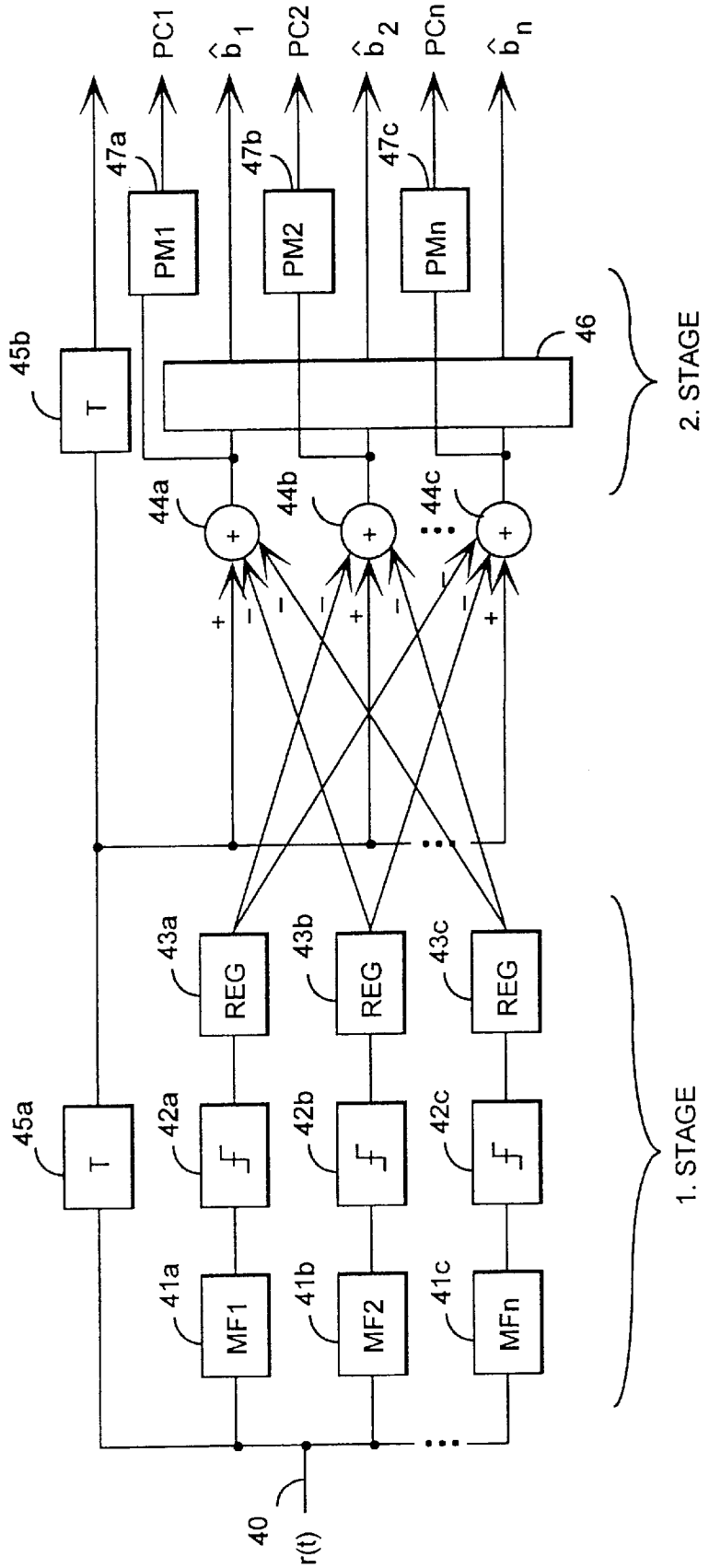

PC1
$\hat{b}_1$
PC2
$\hat{b}_2$
PCn
$\hat{b}_n$
47a
47b
47c
PM1
PM2
PMn
45b
T
46
2. STAGE
44a
44b
44c
+
−
43a
43b
43c
REG
45a
T
42a
42b
42c
41a
41b
41c
MF1
MF2
MFn
1. STAGE
40
r(t)

Fig. 4

POWER CONTROL METHOD IN A CELLULAR COMMUNICATION SYSTEM, AND A RECEIVER

This s application is the national phase of international application PCT/ F195 /00450 filed Aug. 23, 1995.

BACKGROUND OF THE INVENTION

The invention relates to a method for controlling transmission power in a cellular communication system in which some method for cancelling multiple access interference is utilized and in which a receiver controls the transmission power of a transmitter on the basis of a received signal.

When data transmission systems are designed and implemented, it is endeavored to maximize the number of simultaneous users on an available frequency band without compromising the quality of the transmission. An essential problem is simultaneous transmission and reception of the signals of several simultaneous users, so that the signals cause as little interference to each other as possible. Owing to this fact and the available transmission capacity, several different transmission protocols and multiple access methods have been developed, the most common of which in particular in mobile communication are the FDMA and the TDMA methods, and lately also the CDMA method. The present invention is suited for use in particular in CDMA cellular communication systems.

CDMA is a multiple access method, which is based on the spread spectrum technique, and which has been applied recently in cellular communication systems, in addition to the prior FDMA and TDMA methods. CDMA has several advantages over the prior methods, such as the simplicity of frequency planning and spectral efficiency, which results in a large capacity, i.e. the number of simultaneous users on a given frequency band.

In CDMA, the narrow-band data signal of the user is multiplied to a relatively wide band by means of a spreading code having a remarkably broader band than the data signal. Bandwidths used in known test systems are e.g. 1.25 MHz, 10 MHz and. 25 MHz. In connection with the multiplication, the data signal spreads onto the whole of the band used. All users transmit simultaneously by using the same frequency band. An individual spreading code is used on each connection between the base station and the mobile station, and the signals of the users may be distinguished from each other in the receivers on the basis of the spreading code of each user. An attempt is made for choosing the spreading codes so that they are mutually orthogonal, i.e. they do not correlate with each other.

Correlators or adapted filters in CDMA receivers implemented in a conventional way are synchronized with the desired signal, which is identified on the basis of the spreading code. The data signal is returned in the receiver onto the original band by multiplying it by the same spreading code as in the transmission phase. The signals which have been multiplied by some other spreading code neither correlate nor return to the narrow band in an ideal case. They thus appear as noise from the point of view of the desired signal. The aim is thus to detect the signal of the desired user among several interfering signals. In practice, the spreading codes are not completely non-correlated, and the signals of other users complicate the detection of the desired signal by distorting the received signal. This interference caused by the users for each other is termed as multiple access interference.

The mutual interference caused by simultaneous users for each other described above is the decisive factor for the capacity of the CDMA cellular communication system. The interference may be reduced e.g. by attempting to keep the transmission power levels of the mobile stations as low as possible by means of accurate power control. The power control may be based on some parameter measured or calculated from a received transmission, such as the received power, the signal-to-noise ratio or other quality parameter.

It is advantageous from the point of view of the capacity of the CDMA system if the base station receives the signal from all mobile stations with the same power. However, accurate and fast power control has been difficult to implement, and active reception methods based on interference cancellation thus have also been developed for reducing the interference. Such methods include, for example, interference cancellation methods (IC) and the multiuser detection (MUD). From the point of view of the present invention, the above mentioned reception methods are equal, and in the following they will be generally termed as interference cancellation methods.

In the solutions disclosed above, power control and interference cancellation are discussed as separate solutions. Conventional power control does not take into account the interference cancellation methods which are possibly used in the system, and which also improve the quality of the signal in the receiver, and conventional power control has thus resulted in an under-optimal result, in which case the available capacity has not been successfully utilized in the best possible way.

SUMMARY OF THE INVENTION

The object of the present invention is thus to implement a power control method which takes into account the effect of the interference cancellation on the received signal, and thus leads to a better result than heretofore provided from the point of view of the capacity of the system.

This is achieved with a method of the type set forth in the foregoing BACKGROUND section, which is characterized in that power control is carried out on the basis of the received signal processed with the interference cancellation method.

The invention further relates to a receiver in a cellular communication system, comprising means for reducing multiple access interference, and means for controlling the transmission power of the transmitter on the basis of the received signal. The receiver of the invention is characterized in that means for measuring the parameters required for power control from the received signal are connected after the interference cancellation means.

In the method of the invention, the measurement of the parameters that have an effect on power control is carried out from the signal from which interferences have been reduced with an appropriate interference cancellation method. Thus, a better result can be achieved in interference cancellation, which results in a larger capacity of the system.

The power control of the invention does not set any limits to the interference cancellation method used or the selection of the interference signals to be eliminated.

In a second embodiment of the invention, the measurement of the parameters is carried out both before and after the interference cancellation. Then the preceding measurement provides a rapid response, which does not depend on the delay of the interference cancellation. The latter measurement provides a final, accurate measurement result, which results in the desired quality of the user's signal.

BRIEF DESCRIPTION OF THE INVENTION

In the following, the invention will be described in greater detail with reference to the examples in the attached drawings, in which

FIG. **1** shows a cellular communication system in which the method of the invention may be applied,

FIG. **2** is a block diagram illustration of the structure of the receiver in accordance with a preferred embodiment of the invention,

FIG. **3** shows a more detailed illustration of a possible structure of the receiver of the invention,

FIG. **4** illustrates an example of a second possible structure of the receiver of the invention, and

FIG. **5** is a block diagram showing the principle of the structure of the second embodiment of the invention.

DETAILED DESCRIPTION

FIG. **1** illustrates a part of the cellular communication system in which the method of the invention may be applied. Each cell of a cellular radio network comprises at least one base station **10** communicating with the subscriber terminal equipments **11–14** within its area. All terminal equipments transmit on the same frequency to the base station **10**, which distinguishes the transmissions of different terminal equipments from each other on the basis of the spreading code used by each respective terminal equipment. As it has been described above, the signals of the terminal equipments interfere with each other. In the receiver, the power level perceived at the reception of each signal is measured. The results of this power measurement may be utilized for power control. On the basis of the power measurement, it is also possible to calculate other parameters to be utilized for power control and for other purposes, as well.

It is also possible that the base station of FIG. **1** has other frequency bands available to be used for communication with other terminal equipments located within its area. However, these terminal equipments on different frequencies do not interfere with each other, and within both frequencies, the operation of the cell may be assumed to be independent from the point of view of the invention.

Let us assume that in the cellular communication system of FIG. **1** some interference cancellation system is used for reducing multiple access interference. The method of the invention is suited to be used in connection with any known interference cancellation method or power control algorithm. The basic idea of the invention is to take into account the improvement caused by the interference cancellation method used in the system in the received signal prior to calculating and measuring the parameters that have an effect on power control. A power control command is calculated on the basis of the measured parameters, and the command is transmitted to the transmitter by means of known methods. As the measurement of the parameters is carried out after the interference cancellation, the power levels of different signals settle so that a desired quality of the signal is obtained for each user. The aim is generally to balance the final signal-to-interference ratio before the bit decision is made. The balancing leads to the same average bit-error rate for all users.

Filtering, such as averaging of successive parameters may also be connected with the measurement of the parameters, and the purpose of such filtering is to equalize the statistical variation of the estimates, and the prediction with the aid of which it is endeavored to follow and forecast the changes in the received signal.

The method of the invention may be applied, e.g., in connection with the successive interference cancellation. In the successive interference cancellation, the receiver processes the received transmission so that the signals are demodulated in a certain order, typically in order of magnitude, regenerated and reduced from the received transmission, whereafter the following signal is processed in the same way until all the signals have been processed. In the method of the invention, instead of estimating the power of each user from the total signal in which all interferences are included, the power control and the measurement of parameters connected therewith are carried out on the basis of the purged signal, from which the signals stronger than those of the current user have been reduced.

The invention is also suited to be used in connection with so-called multi-stage interference cancellation in which all the users to be received are processed in parallel, and the bit estimates are adjusted iteratively by repeating the reception procedure after the interference estimates have been reduced. Similarly, the estimates of the necessary power control parameters may be adjusted iteratively for obtaining as reliable power control as possible.

FIG. **2** illustrates a block diagram of the structure of the preferred embodiment of the invention. The receiver of the invention comprises an antenna **20**, by means of which the received signal is applied to radio frequency elements **21**. From the radio frequency elements, the signal is applied via an A/D converter **22** to means **23**, where the interference cancellation and detection of the received signal are carried out. The receiver further comprises means **24**, which carry out the measurement of power control parameters from the received signal, and which are connected after the interference cancellation and detection means **23**. The signal received from means **24** is further applied to other elements of the receiver. The receiver further comprises control means **25**, which control the operation of the elements mentioned above and calculate the actual power control commands on the basis of the parameters obtained from the measurement means.

FIG. **3** illustrates the structure of the receiver of the invention in closer detail as to the essential parts of the invention, in a case in which the successive interference cancellation is employed in the system. The blocks shown in FIG. **3** correspond to blocks **23** and **24** in FIG. **2**. As it has been described above, in the successive interference cancellation, the received transmission is processed in the receiver, so that the signals are demodulated in a certain order, typically in order of magnitude, regenerated and reduced from the received transmission, whereafter the following signal is processed in the same way, until all signals have been processed.

The receiver in accordance with FIG. **3** comprises a plurality of adapted filters or RAKE receivers **31***a*–**31***c*, which are each adapted to receive and demodulate the signal of one user, which signals may be distinguished from each other on the basis of the spreading code. The signals are typically demodulated in order of magnitude, whereby the interfering effect of the strongest signals may be eliminated prior to processing the weaker signals.

The received transmission **30** is applied to a first adapted filter **31***a*, where the desired signal is demodulated, and further to a first detector **32***a*, where the bit decision is made. Signal **38***a* obtained from detector **32***a*, thus comprising the estimate of the transmission of the first user, is further applied to other elements of the receiver. In accordance with the successive interference cancellation method, the signal obtained from the first detector **32***a* is also applied to a first regeneration means **34***a*, where the detected signal is regenerated again, i.e. re-multiplied by the spreading code. The obtained regenerated signal is further applied to a first

summing means **36***a,* where it is reduced from the received transmission **30**, which has been applied to summing means **36***a* via a first delay means **35***a*. The signal located at the output of the first adapted filter **31***a* is applied, in addition to the first detector **32***a,* to a first measurement means **33***a,* which carries out from the signal the measurement of the parameters required for power control. A typical measurement parameter is, e.g., the received power contained by the signal. The obtained measurement results **37***a* are applied to further processing.

A signal **39***a* obtained from the first summing means **36***a* thus comprises the received transmission, from which the effect of the signal demodulated by the first adapted filter **31***a,* i.e. typically the effect of the strongest signal has been eliminated. The signal **39***a* is applied to a second adapted filter **31***b* and a second detector **32***b,* the output of which provides the bit decision **38***b* of the signal of the second user. The signal **38***b* of the second user is correspondingly applied to a second regeneration means **34***b,* wregener is regenerated and applied to a second summing means **36***b,* where it is reduced from the transmission **39***a,* which has been applied to the summing means via a second delay means **35***b*. The signal at the output of the second adapted filter **31***b* is applied, in addition to the second detector **32***b,* to a second measurement means **33***b,* which carries out from the signal the measurement of the parameters required for power control. In the receiver of the invention, the measurement results correspond better to a real situation, since the interfering effect of the stronger signals has been eliminated from the signal from which the measurement is carried out. The obtained measurement results **37***b* are applied to further processing.

The corresponding operations are performed to all received signals until the last user, whose signal is demodulated by means of an adapted filter **31***c* and a detector means **32***c*. From the output signal of the adapted filter **31***c,* the measurement of the power control parameter of the last user is carried out in measurement means **33***c*. The signal **38***c* of the last user does not need to be regenerated as it does not have to be reduced from the received signal, since all users have already been detected.

FIG. **4** illustrates the structure of the receiver of the invention in closer detail as to the essential parts for the invention, when a parallel multi-stage interference cancellation is employed in the system. The blocks in FIG. **4** correspond to blocks **23** and **24** in FIG. **2**. As has been described above, in the multi-stage interference cancellation, all users received are processed in parallel, and bit estimates are adjusted iteratively by repeating the reception procedure after eliminating the interference estimates. The procedure may be repeated twice or several times, i.e. the receiver may comprise several successive stages. The estimates of the necessary power control parameters may also be adjusted iteratively, as well as the detected signal for accomplishing as reliable power control as possible.

FIG. **4** illustrates the first two stages of a multi-stage receiver, but there may be more stages. A received signal **40** is simultaneously applied to adapted filters **41***a*–**41***c,* which each demodulate the signal of one user. The number of the adapted filters is thus the same as the number of current users. The output signals of the adapted filters are applied to detector means **42***a*–**42***c,* where a bit decision is made for each signal. When desired, the bit decision may be made in other elements of the receiver, but it is not marked in the figure for the sake of simplicity. The detected signals are further applied to regeneration means **43***a*–**43***c,* where the original transmission of each user is regenerated from the detected signal estimates. The adapted filters **41***a*–**41***c,* the detector means **42***a*–**42***c* and the regeneration means **43***a*–**43***c* form the first stage of the receiver.

The regenerated signals are further applied as negative inputs to summing means **44***a*–**44***c,* the number of which is the same as the number of the users of the system. The original received transmission **40** is applied as a positive input to each summing means **44***a*–**44***c* via a first delay means **45***a*. In a first summing means **44***a,* the regenerated signals, i.e. the output signals of regeneration means **43***b*–**43***c* of the other users except that of the first user are reduced from the original signal. The output signal of the first summing means **44***a* thus comprises a transmission which comprises the signal of the first user, and from which the interference estimate of the other users has been reduced. Accordingly, the output signals of summing means **44***b* and **44***c* only comprise the transmission of the desired user, from which transmission the interference estimates of other users have been reduced. The obtained signals are further applied to a second stage **46** of the receiver, where signals are further processed and re-detected. There may be several successive stages.

The output signals of summing means **44***a*–**44***c* are also each applied to a separate measurement means **47***a*–**47***c,* where the parameters required for power control are measured from each signal, and those parameters are further applied to the unit which is responsible for power control. Since the interfering effect of other signals has been eliminated from the signal from which the measurement is carried out, the obtained result is more accurate than what has been possible to achieve with prior art methods. A corresponding measurement may also be carried out from the output signals of latter stages, which provides an even more accurate result.

The power control of the invention thus takes into account the effect of the interference cancellation on the signal-to-interference ratio, on the basis of which the quality of each signal is determined, and thus leads to a desired optimal power control result when interference cancellation algorithms are employed.

When the successive interference cancellation method is employed, the power control of the invention leads to a power distribution in which the signals to be removed first are the strongest, whereas in the multi-stage interference cancellation the power distribution is even. Special features may be added to the power control of the invention if desired, such as generation of different grades of service. For example, part of the users may be set a better signal-to-interference target level the power control aims at, which naturally results in capacity loss compared with a uniform system.

In the second embodiment of the invention the measurement of the power control parameters is carried out both before and after carrying out the interference cancellation. In that case, the first measurement rapidly provides a measurement result that does not depend on the delay inevitably connected with the interference cancellation. On the basis of the measurement after the interference cancellation, a final, more accurate, result is obtained on the basis of which a more accurate adjustment may be carried out that leads to a desired quality of the user's signal.

The structure of the receiver in accordance with the second embodiment of the invention is illustrated in the block diagram in FIG. **5**. As in the receiver in FIG. **2**, the receiver of the invention comprises an antenna **20**, by means of which the received signal is applied to radio frequency elements **21**. From the radio frequency elements the signal

is applied via an A/D converter **22** to a first measurement means **50**, where a preliminary measurement of the power control parameters is carried out. This corresponds to the measurement method employed in prior art methods. From the first measurement means **50** the signal is applied to means **23**, where the interference cancellation and detection of the received signal is carried out. The receiver further comprises means **51** which carry out the measurement of the power control parameters from the received signal which has been processed with interference cancellation methods, means **51** being connected after interference cancellation and detection means **23**. The signal obtained from means **51** is further applied to other elements of the receiver. The receiver further comprises control means **25** that control the operation of the above mentioned elements and calculate the actual power control commands on the basis of the parameters obtained from measurement means **50** and **51**.

Although the invention has been described above with reference to the examples in accordance with the attached drawings, it is obvious that the invention is not limited thereto, but it may be modified in a variety of ways within the scope of the inventive idea set forth in the attached claims.

We claim:

1. A method for controlling transmission power in a cellular communication system, comprising the steps of:

receiving a signal by a receiver as a result of transmission by a transmitter;

subjecting the thereby received signal to interference cancellation to obtain a received signal which has been subjected to interference cancellation;

measuring parameters required for power control of the transmitter from said received signal which has been subjected to interference cancellation; and

said receiver controlling transmission power of said transmitter by use of said parameters.

2. The method of claim **1**, wherein:

said parameters include power of said received signal.

3. The method of claim **1**, wherein:

said parameters include signal-to-interference ratio of said received signal.

4. The method of claim **1**, further including:

also measuring parameters required for power control of the transmitter from said received signal before subjecting said received signal to said interference cancellation;

and said receiver controlling said transmission power of said transmitter by use not only of the respective said parameters measured from said received signal which has been subjected to interference cancellation but also of the respective said parameters measured from said received signal before said received signal has been subjected to interference cancellation.

5. A method for controlling transmission power in a cellular communication system, comprising the steps of:

receiving each of a plurality of signals by a receiver as a result of respective transmissions by a plurality of transmitters;

subjecting each thereby received signal to interference cancellation to obtain a respective received signal which has been subjected to interference cancellation;

measuring parameters required for power control of respective ones of said transmitters from respective ones of said received signals which have been subjected to interference cancellation; and

said receiver controlling transmission power of respective ones of said transmitters by use of respective ones of said parameters;

said receiving comprising detecting each said received signal from a given frequency band, in decreasing magnitude of received signal power, such that prior to detection of each said received signal subsequent to one having a greatest received signal power, effects of interference of those of said received signals having stronger received signal power are eliminated from the respective said received signals.

6. A method for controlling transmission power in a cellular communication system, comprising the steps of:

receiving each of a plurality of signals by a receiver as a result of respective transmissions by a plurality of transmitters;

subjecting each thereby received signal to interference cancellation to obtain a respective received signal which has been subjected to interference cancellation;

measuring parameters required for power control of respective ones of said transmitters from respective ones of said received signals which have been subjected to interference cancellation; and

said receiver controlling transmission power of respective ones of said transmitters by use of respective ones of said parameters;

said receiving comprising detecting all of said received signals in parallel, from a given frequency band.

7. The method of claim **6**, comprising:

practicing said detecting in a plurality of stages; and

measuring said parameters in a plurality of stages.

8. A receiver in a cellular communication system, comprising:

means for reducing multiple access interference;

means for controlling transmission power of a transmitter on the basis of reception by said receiver of a signal as a result of transmission by said transmitter; and

first means for measuring parameters required for controlling transmission power of said transmitter by said receiver, said measuring means being connected in said receiver functionally after said interference cancellation means.

9. The receiver of claim **8**, further comprising:

second means for measuring parameters required for power control of the transmitter from said received signal before subjecting said received signal to said interference cancellation;

said means for controlling transmission power being arranged for controlling said transmission power by use not only of the respective said parameters measured by said first means for measuring, but also the respective of said parameters measured by said second means for measuring.

10. A receiver in a cellular communication system, comprising:

means for reducing multiple access interference;

means for controlling transmission power of each of a plurality of transmitters on the basis of reception by said receiver of respective signals as a result of respective transmissions by said transmitters, to thereby provide respective receiver signals;

means for measuring parameters required for controlling transmission power of respective ones of said transmitters by said receiver, said measuring means being connected in said receiver functionally after said interference cancellation means, and

detector means for detecting each said received signal from a given frequency band, in decreasing order of received signal power, such that prior to detection of each said received signal subsequent to one having a greatest received signal power, effects of interference of those at said received signals having stronger received signal power are eliminated from the respective said received signals.

11. A receiver in a cellular communication system, comprising:

means for reducing multiple access interference;

means for controlling transmission power of each of a plurality of transmitters on the basis of reception by said receiver of respective signals as a result of respective transmissions by said transmitters, to thereby provide respective receiver signals;

means for measuring parameters required for controlling transmission power of respective ones of said transmitters by said receiver, said measuring means being connected in said receiver functionally after said interference cancellation means, and

detector means for detecting all of said received signals in parallel, from a given frequency band.

* * * * *

EXHIBIT F

US007319874B2

(12) **United States Patent**
Rautiola et al.

(10) **Patent No.: US 7,319,874 B2**
(45) **Date of Patent: *Jan. 15, 2008**

(54) **DUAL MODE TERMINAL FOR ACCESSING A CELLULAR NETWORK DIRECTLY OR VIA A WIRELESS INTRANET**

(75) Inventors: **Markku Rautiola**, Tampere (FI); **Jussi Lemilainen**, Arlington, MA (US); **Markku Niemi**, Tampere (FI)

(73) Assignee: **Nokia Corporation**, Espoo (FI)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 115 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **10/941,915**

(22) Filed: **Sep. 16, 2004**

(65) **Prior Publication Data**

US 2005/0064896 A1 Mar. 24, 2005

Related U.S. Application Data

(63) Continuation of application No. 09/646,419, filed on Nov. 30, 2000, now Pat. No. 6,853,851.

(51) **Int. Cl.**
H04Q 7/20 (2006.01)

(52) **U.S. Cl.** **455/455**; 455/446; 455/422.1; 455/560; 455/432.1; 455/450; 455/550.1

(58) **Field of Classification Search** 455/455, 455/446, 422.1, 560, 432.1, 450, 127.4, 553.1, 455/552.1, 550.1
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,706,331	A	1/1998	Wang et al.	
5,790,953	A	8/1998	Wang et al.	
6,285,757	B1	9/2001	Carroll et al.	
6,295,461	B1	9/2001	Palmer et al.	
6,326,613	B1	12/2001	Heslin et al.	
6,405,049	B2	6/2002	Herrod et al.	
6,553,232	B1 *	4/2003	Shaffer et al.	455/440
6,711,401	B1 *	3/2004	Chow et al.	455/414.1
6,766,160	B1 *	7/2004	Lemilainen et al.	455/411
6,853,851	B1 *	2/2005	Rautiola et al.	455/553.1
6,904,025	B1 *	6/2005	Madour et al.	370/328
2005/0192048	A1 *	9/2005	Bridgelall	455/553.1

FOREIGN PATENT DOCUMENTS

AU	724249	11/1997
EP	0 735 789 A2	10/1996
EP	0 766 427 A2	4/1997
GB	2 292 047 A	2/1996
GB	2 316 581 A	2/1998

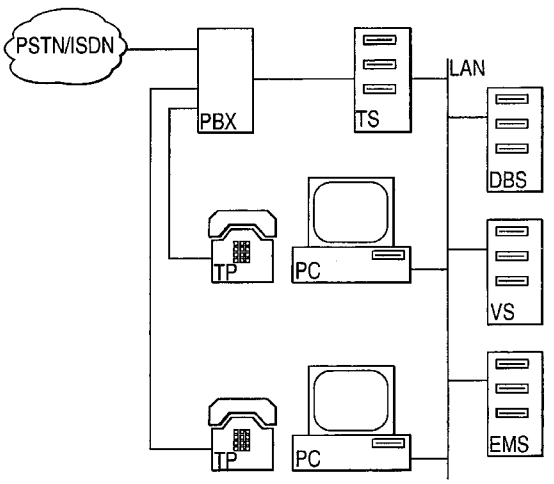
(Continued)

Primary Examiner—Rafael Perez-Gutierrez
Assistant Examiner—Khai Nguyen
(74) *Attorney, Agent, or Firm*—Foley and Lardner LLP

(57) **ABSTRACT**

A wireless intranet office (WIO) concept is disclosed, which integrates an IP based Intranet environment (**27**) and GSM network providing mobile telephones (**21**) with access to GSM through the GSM network or via the intranet. Access through the intranet to the GSM MSC (**26**) is provided by a WIO interworking unit (**24**) which may comprise several network entities (e.g. intranet mobile cluster (**241**), intranet location register (**242**), WIO gatekeeper (**243**), WIO gateway (**244**) and H.323 gateway). A dual mode terminal for such a system is also disclosed.

21 Claims, 15 Drawing Sheets

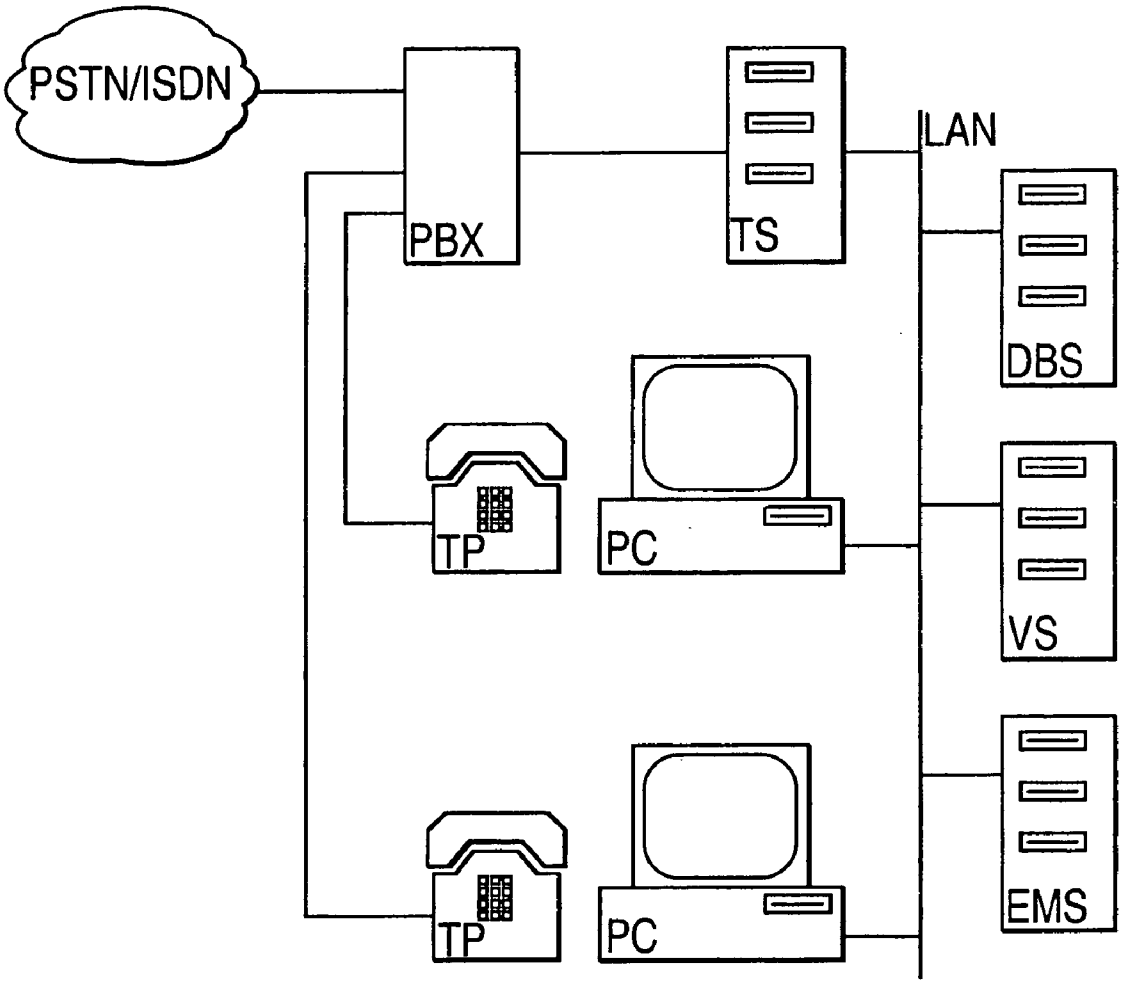


FOREIGN PATENT DOCUMENTS

GB	2 327 016 A	1/1999
WO	WO 95/33348	12/1995
WO	WO 97/34429	9/1997
WO	WO 97/36442	10/1997
WO	WO 98/09455	3/1998
WO	WO 98/10617	3/1998
WO	WO 98/15143	4/1998

* cited by examiner

FIG. 1



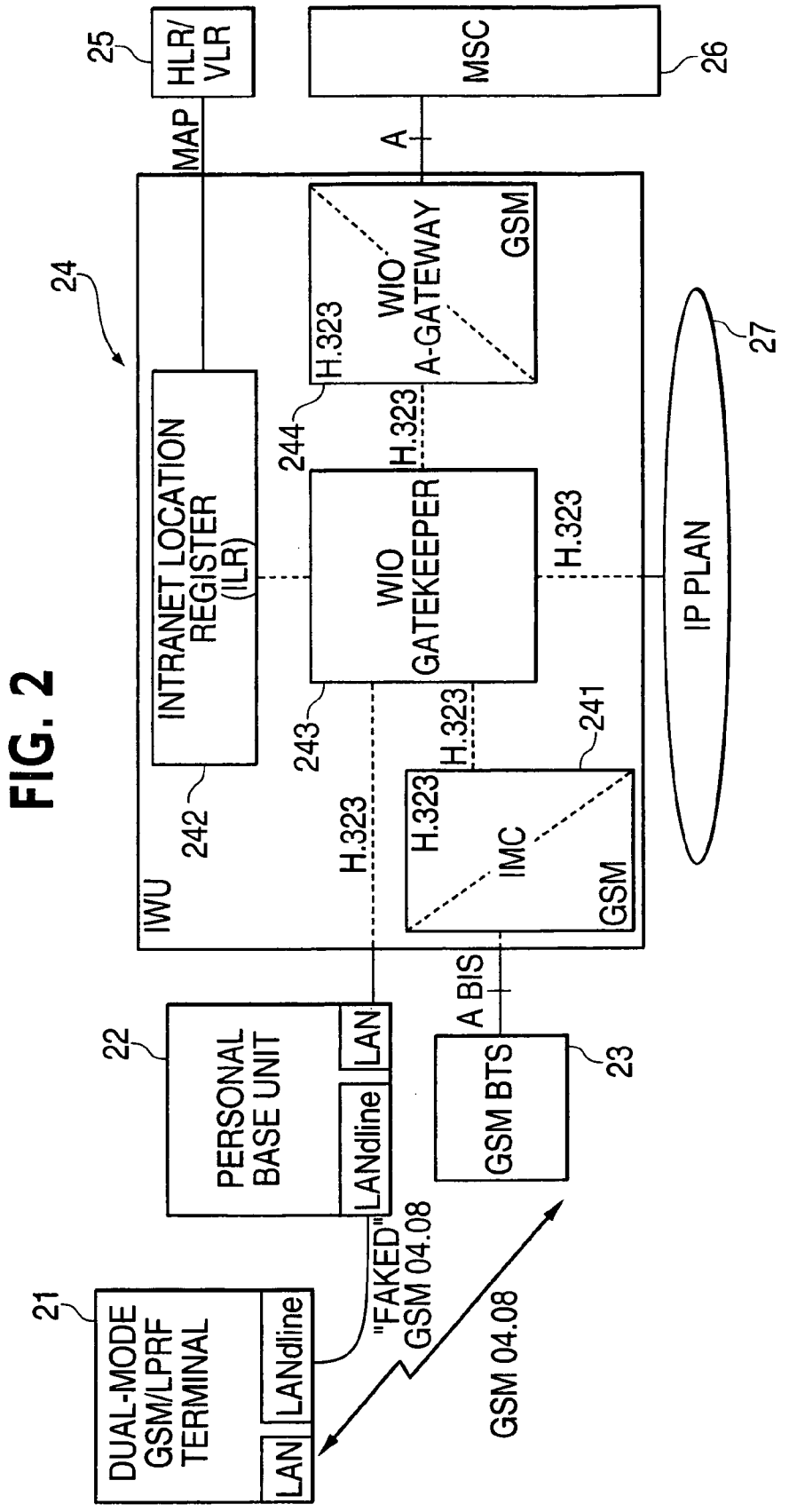
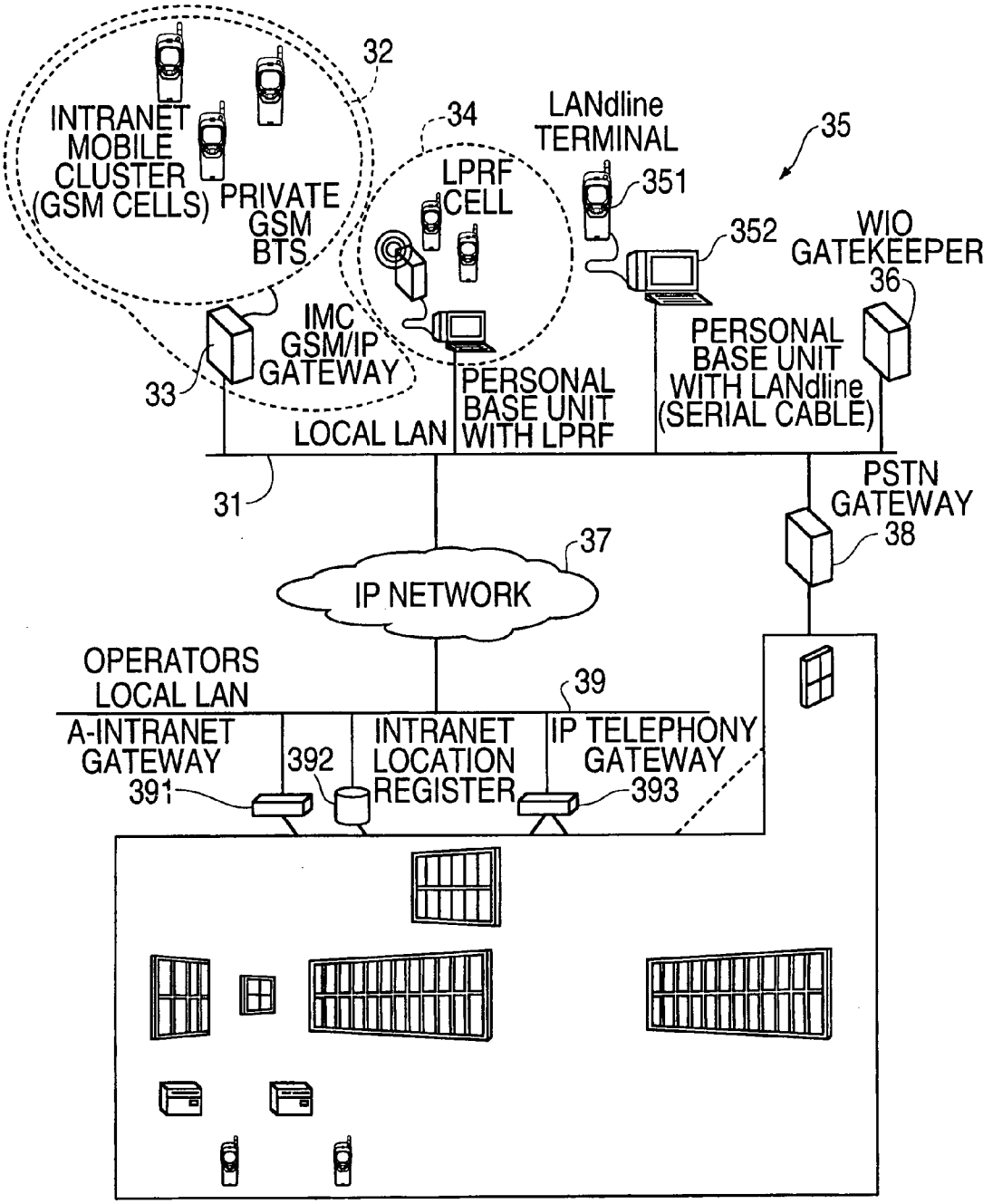

25
HLR/
VLR
MSC
26
MAP
A
24
WIO
A-GATEWAY
GSM
H.323
27
244
H.323
INTRANET LOCATION
REGISTER
(ILR)
WIO
GATEKEEPER
H.323
IP PLAN
243
H.323
241
H.323
IMC
GSM
242
IWU
H.323
A BIS
22
PERSONAL
BASE UNIT
LAN
LANdline
GSM BTS
23
"FAKED"
GSM 04.08
21
DUAL-MODE
GSM/LPRF
TERMINAL
LANdline
LAN
GSM 04.08

FIG. 2

FIG. 3


32
INTRANET
MOBILE
CLUSTER
(GSM CELLS)
PRIVATE
GSM
BTS
34
LPRF
CELL
LANdline
TERMINAL
351
35
352
WIO
GATEKEEPER
36
IMC
GSM/IP
GATEWAY
33
PERSONAL
BASE UNIT
WITH LPRF
PERSONAL
BASE UNIT
WITH LANdline
(SERIAL CABLE)
LOCAL LAN
31
PSTN
GATEWAY
38
37
IP NETWORK
OPERATORS
LOCAL LAN
39
A-INTRANET
GATEWAY
391
392
INTRANET
LOCATION
REGISTER
IP TELEPHONY
GATEWAY
393

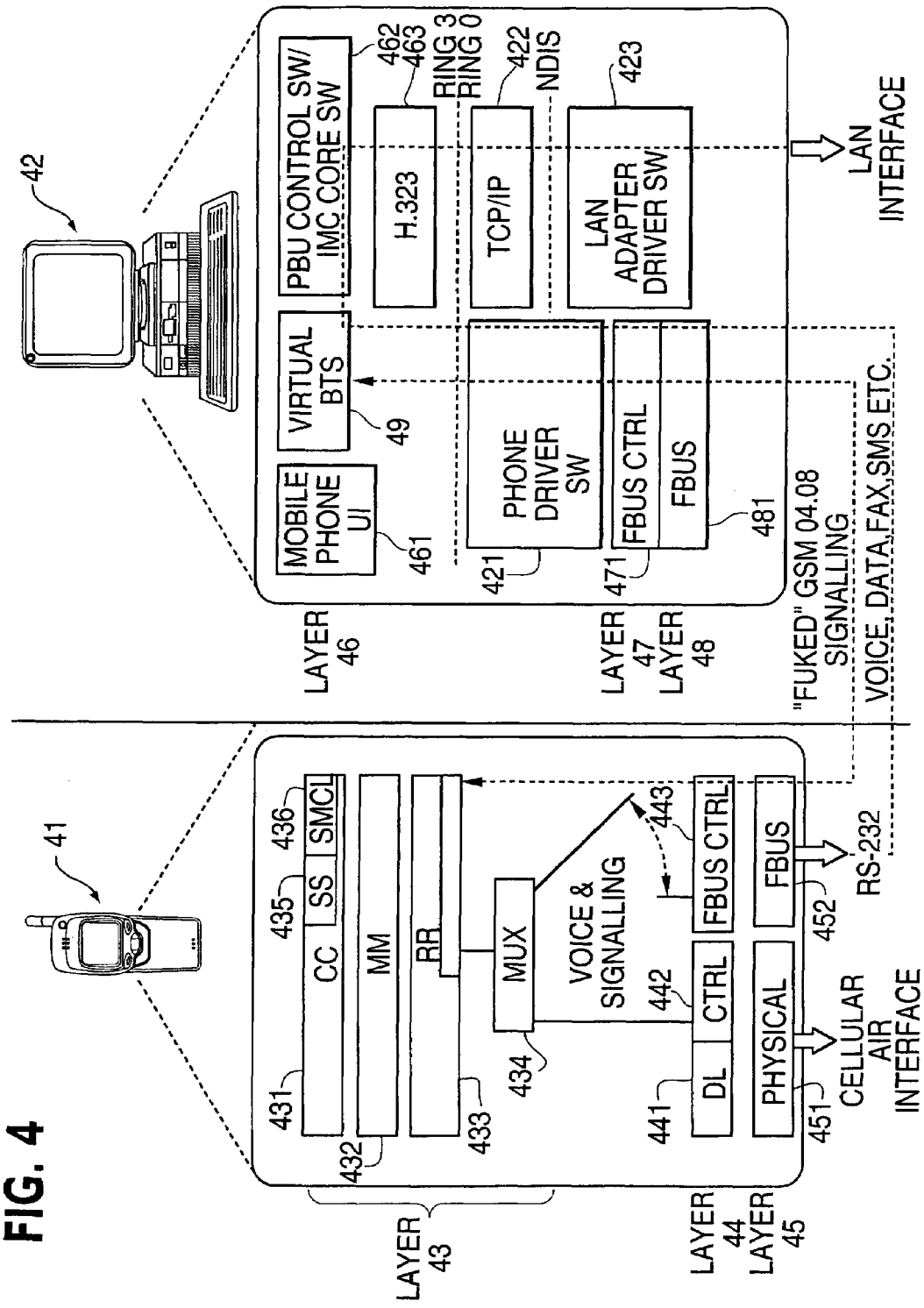

42
41
LAYER 46
LAYER 47
LAYER 48
LAYER 43
LAYER 44
LAYER 45
MOBILE PHONE UI
461
VIRTUAL BTS
49
PBU CONTROL SW/ IMC CORE SW
462
463
RING 3
RING 0
H.323
TCP/IP
422
NDIS
LAN ADAPTER DRIVER SW
423
LAN INTERFACE
PHONE DRIVER SW
421
FBUS CTRL
FBUS
471
481
"FUKED" GSM 04.08 SIGNALLING
VOICE, DATA,FAX,SMS ETC.
CC
SS
SMCL
435
436
MM
RR
431
432
433
MUX
434
VOICE & SIGNALLING
443
442
CTRL
FBUS CTRL
441
DL
PHYSICAL
FBUS
451
452
RS-232
CELLULAR AIR INTERFACE

FIG. 4

FIG. 5

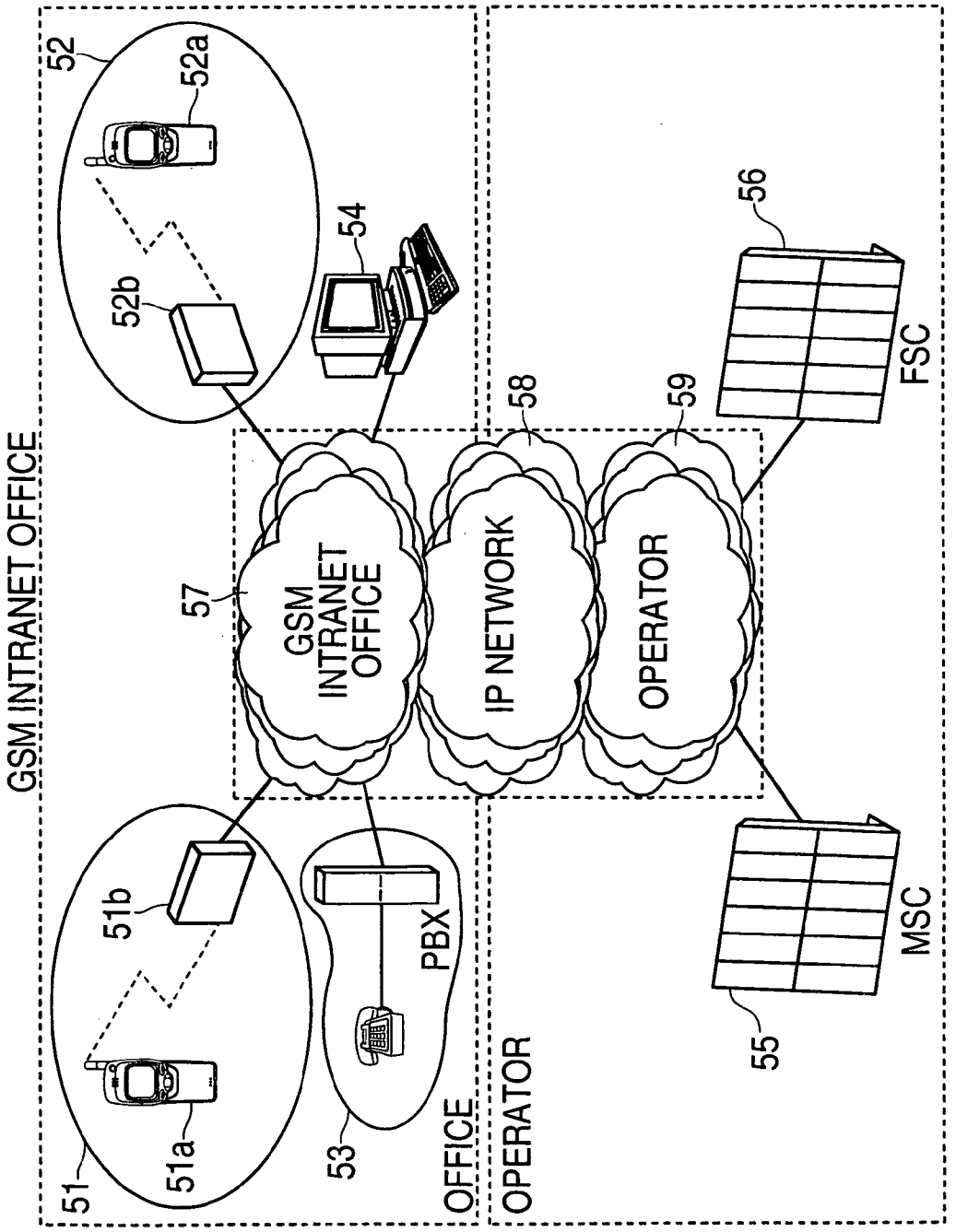

GSM INTRANET OFFICE
52
52a
52b
54
56
FSC
58
59
57
GSM INTRANET OFFICE
IP NETWORK
OPERATOR
51b
PBX
51
51a
53
OFFICE
OPERATOR
55
MSC

FIG. 6

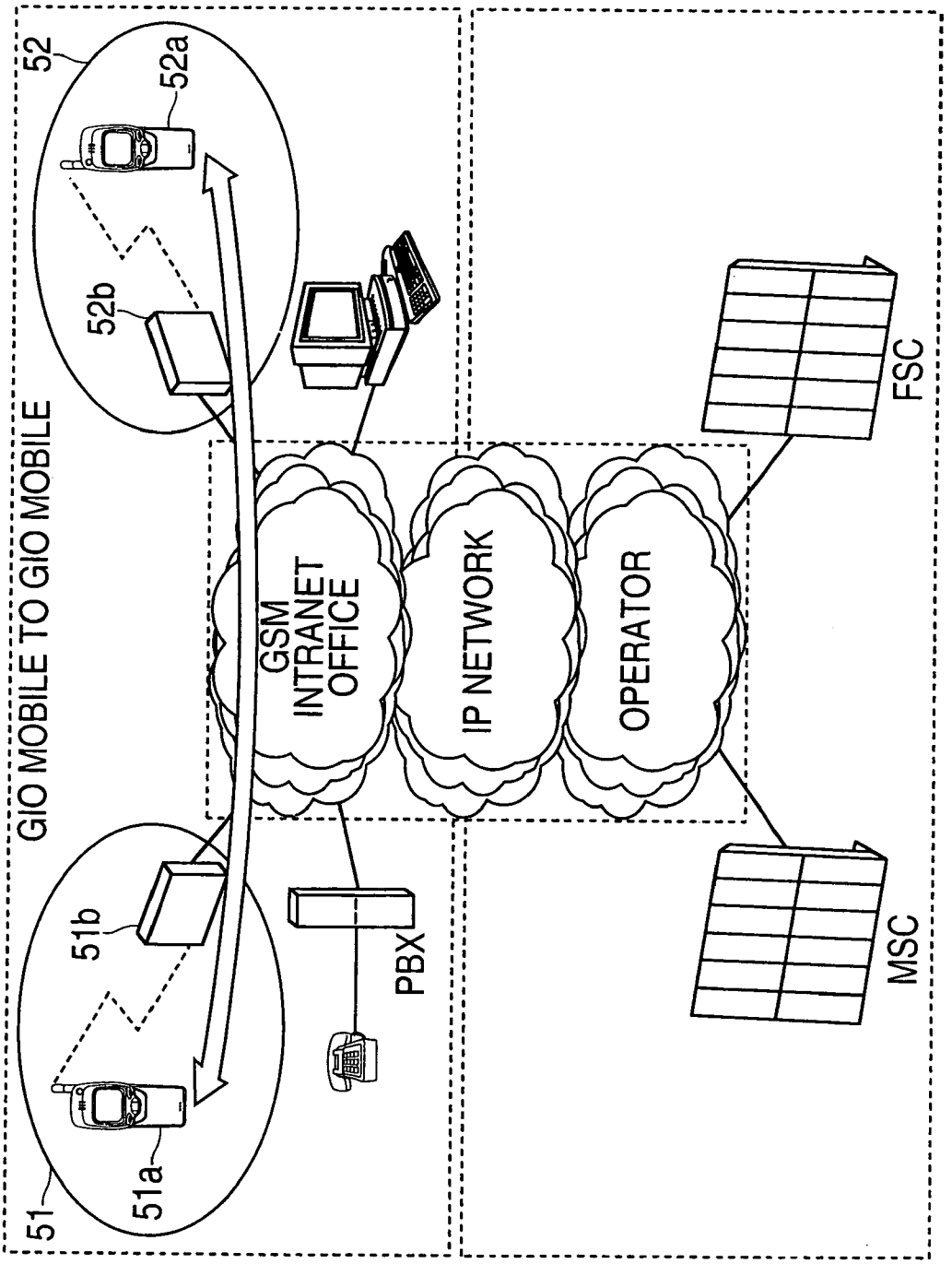

GIO MOBILE TO GIO MOBILE
52
52a
52b
51
51a
51b
GSM INTRANET OFFICE
IP NETWORK
OPERATOR
PBX
FSC
MSC

FIG. 7

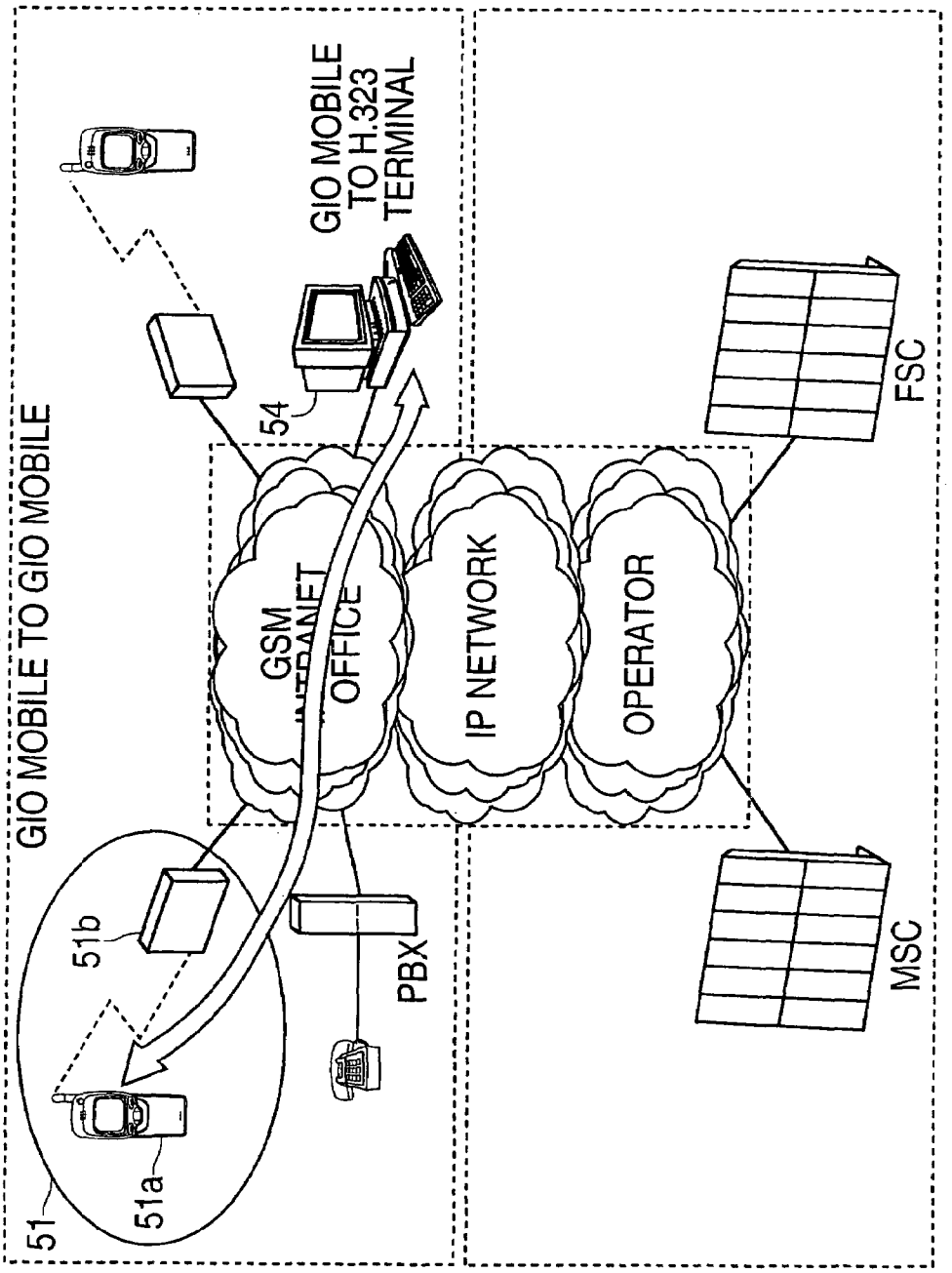

GIO MOBILE TO GIO MOBILE
GIO MOBILE TO H.323 TERMINAL
54
GSM INTRANET OFFICE
IP NETWORK
OPERATOR
FSC
MSC
PBX
51
51a
51b

FIG. 8

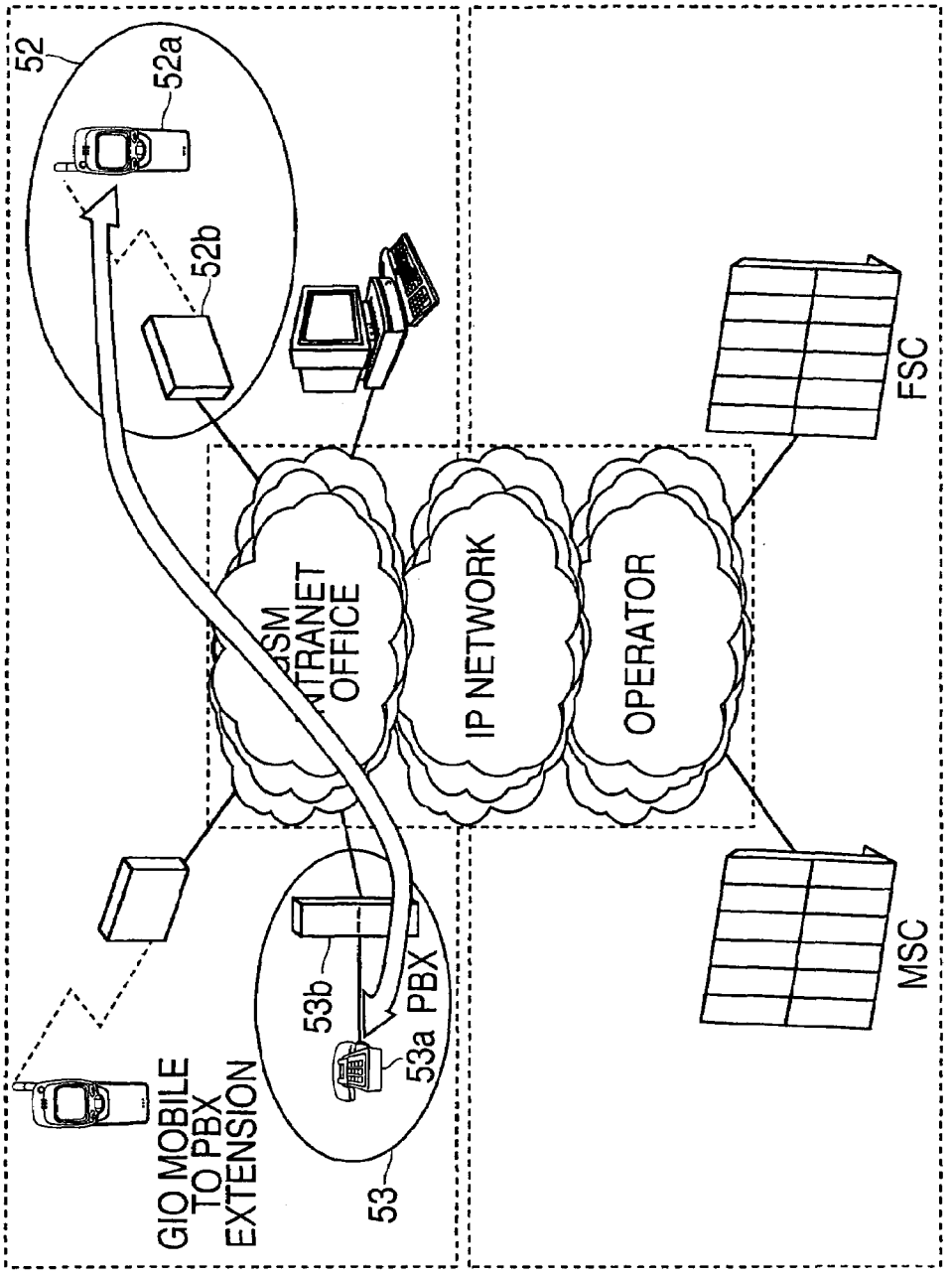

52
52a
52b
GSM INTRANET OFFICE
IP NETWORK
OPERATOR
FSC
MSC
53b
53a PBX
53
GIO MOBILE TO PBX EXTENSION

FIG. 9

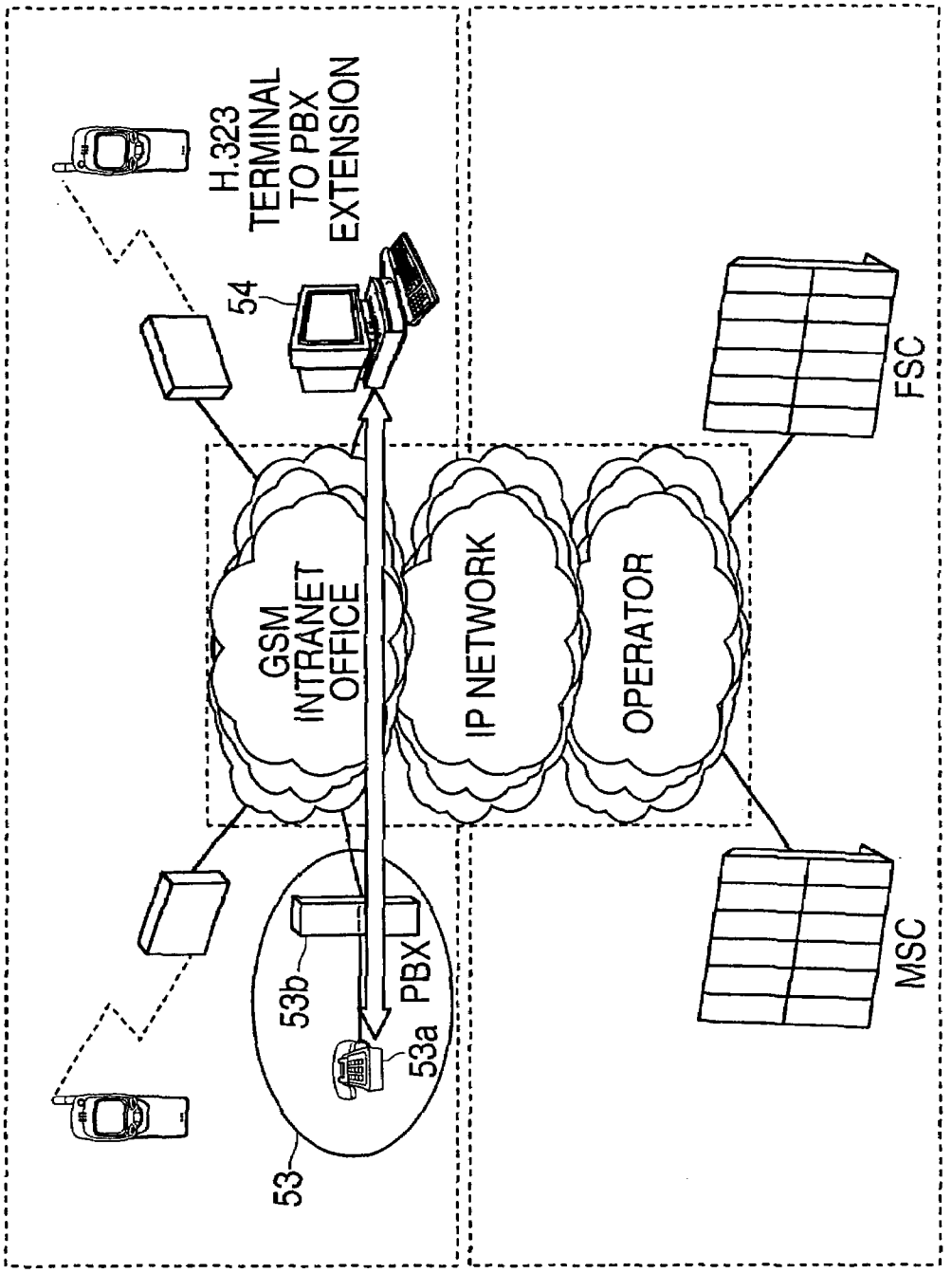

H.323
TERMINAL
TO PBX
EXTENSION
54
FSC
GSM
INTRANET
OFFICE
IP NETWORK
OPERATOR
MSC
53b
PBX
53a
53

FIG. 10

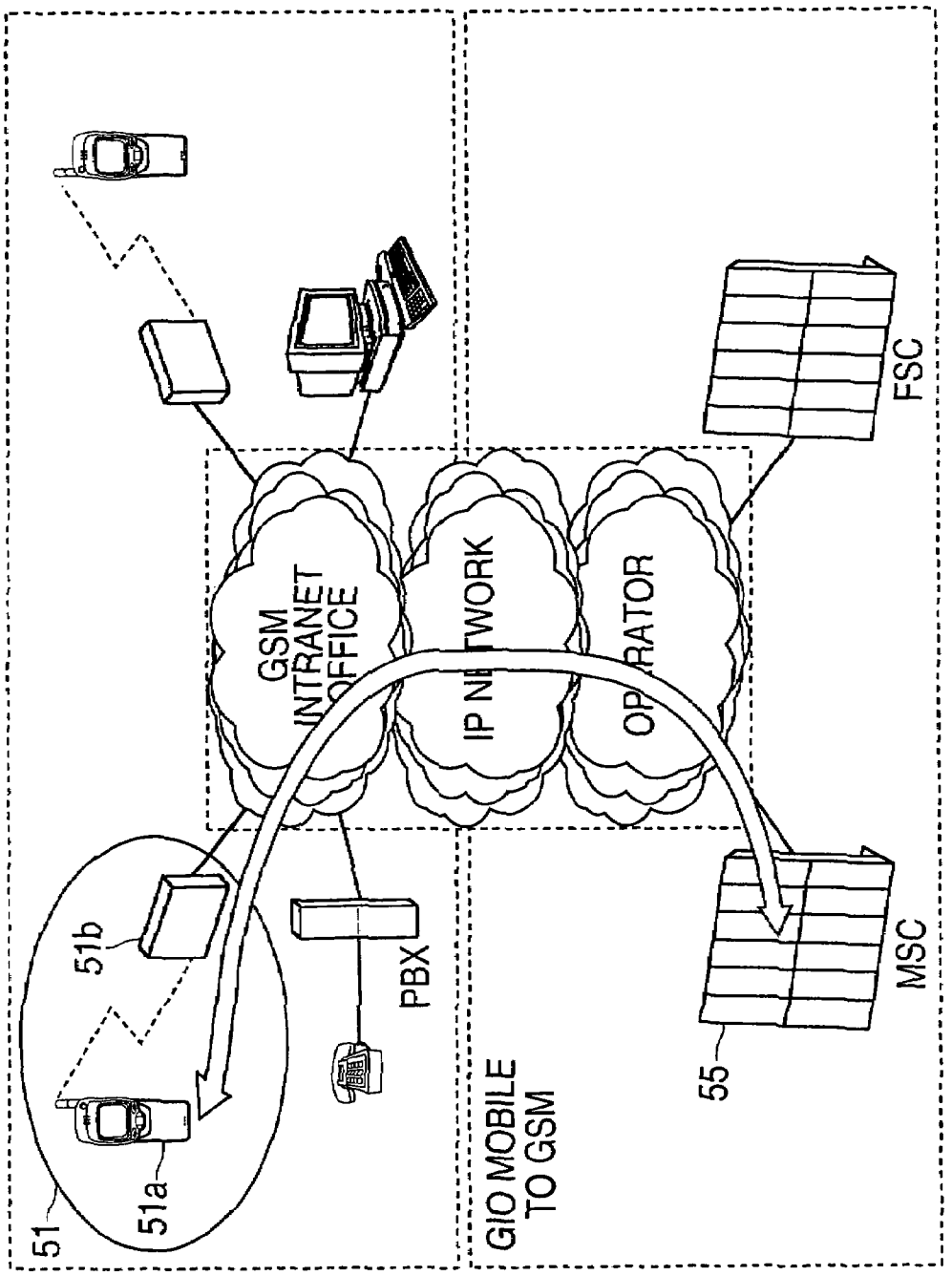

GSM
INTRANET
OFFICE
IP NETWORK
OPERATOR
FSC
MSC
55
PBX
51b
51a
51
GIO MOBILE
TO GSM

FIG. 11

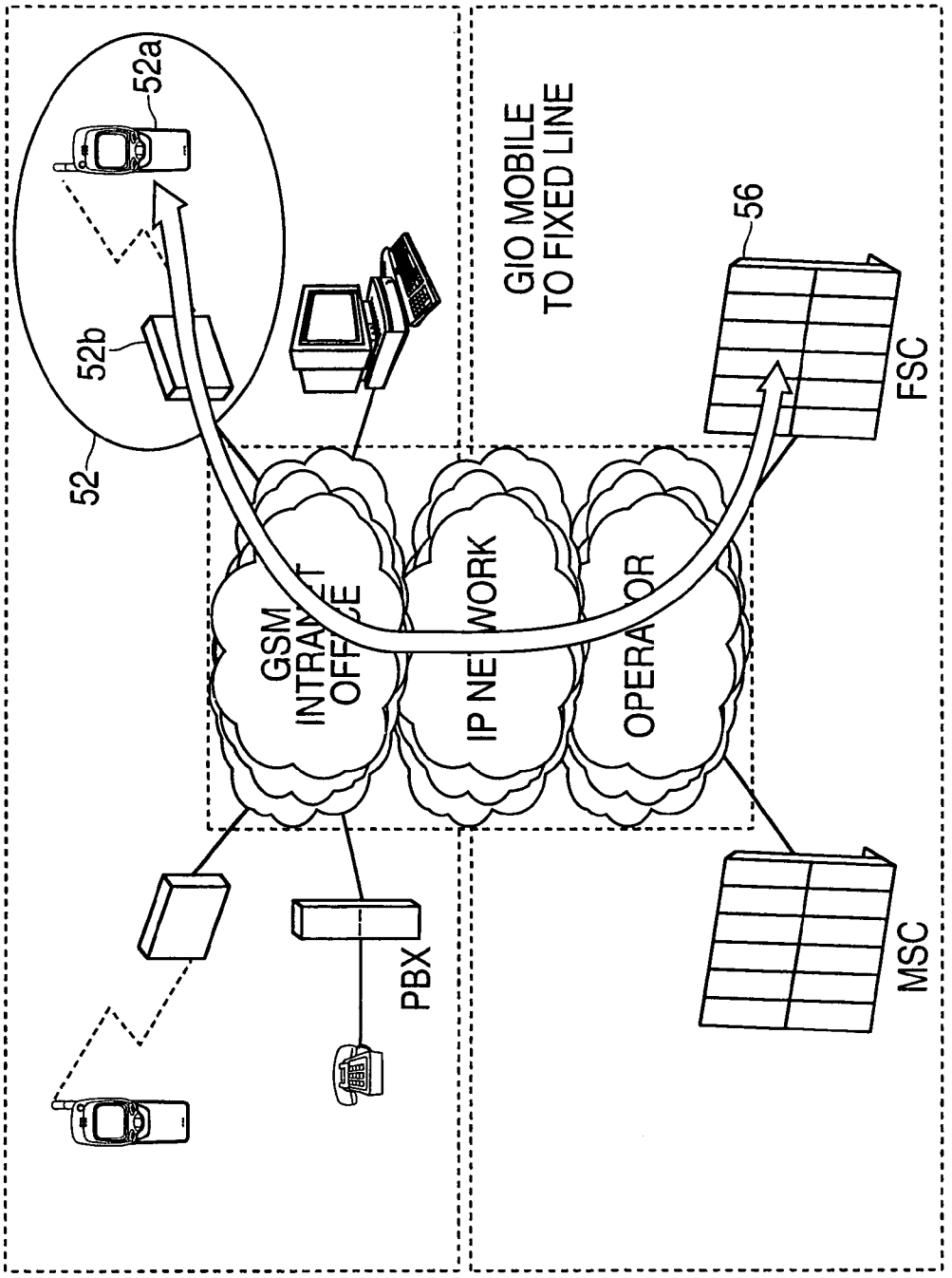

52a
52b
52
GIO MOBILE
TO FIXED LINE
56
FSC
GSM
INTRANET
OFFICE
IP NETWORK
OPERATOR
PBX
MSC

FIG. 12

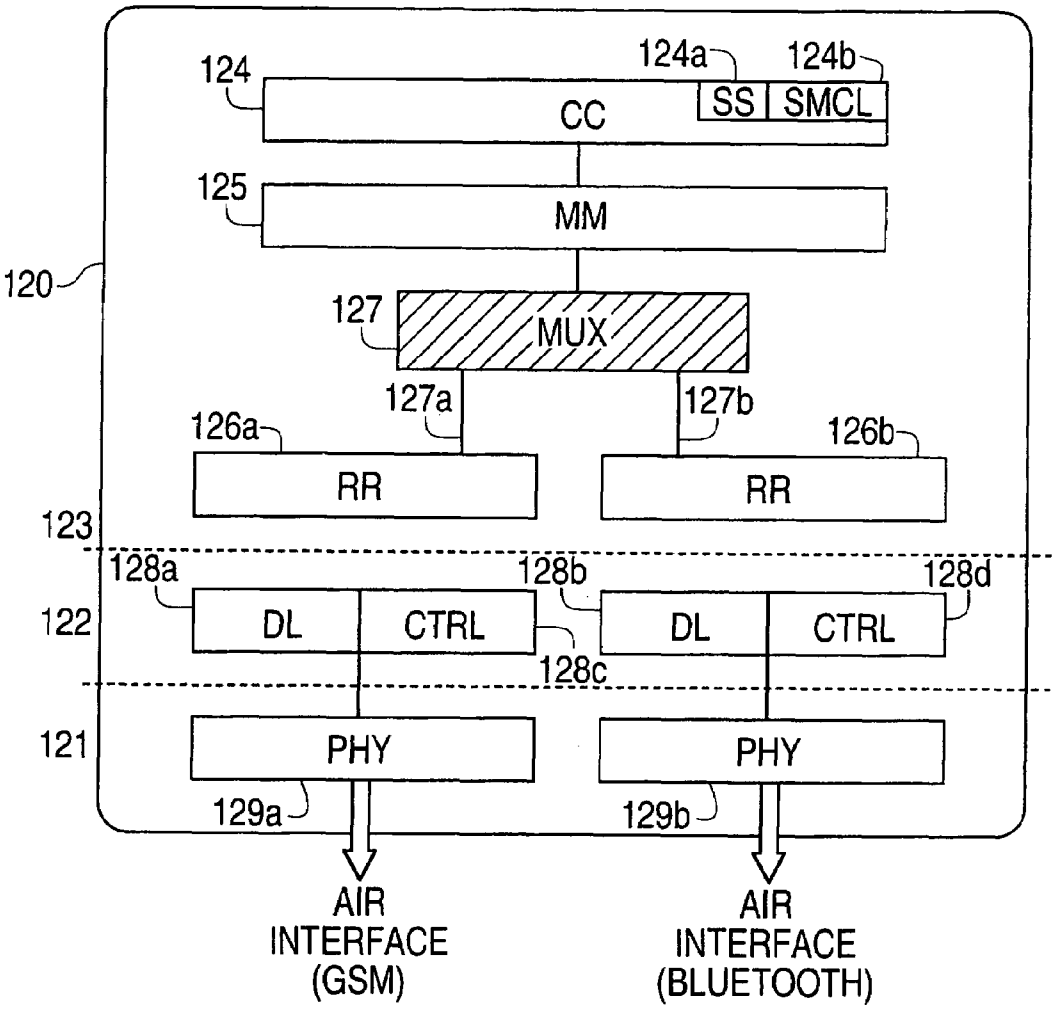

120
123
122
121
124
124a
124b
CC
SS
SMCL
125
MM
127
MUX
127a
127b
126a
126b
RR
RR
128a
128b
128d
DL
CTRL
DL
CTRL
128c
PHY
PHY
129a
129b
AIR INTERFACE (GSM)
AIR INTERFACE (BLUETOOTH)

FIG. 13

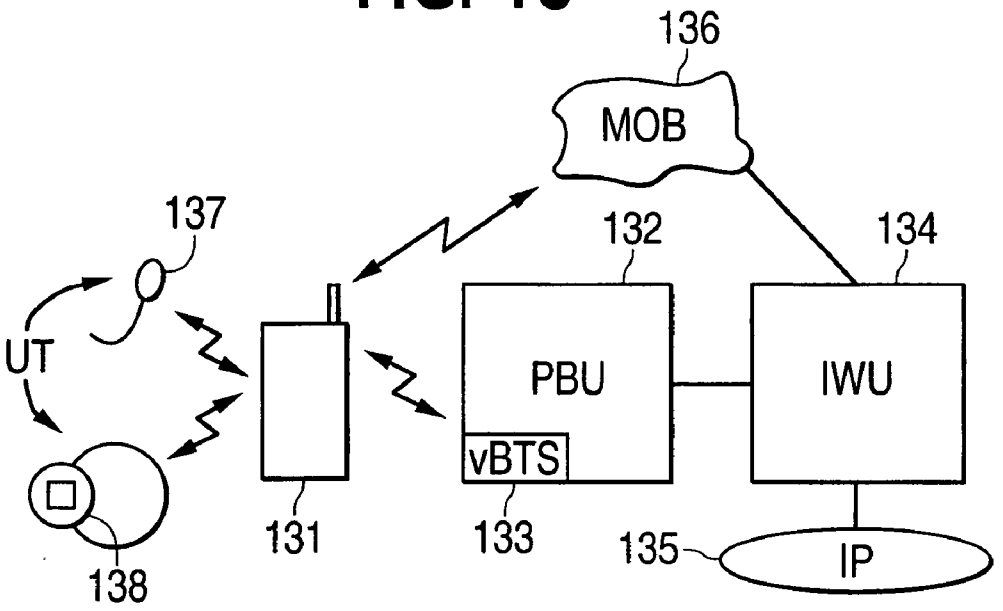

136
MOB
137
132
134
UT
PBU
IWU
vBTS
131
133
135
IP
138

FIG. 14

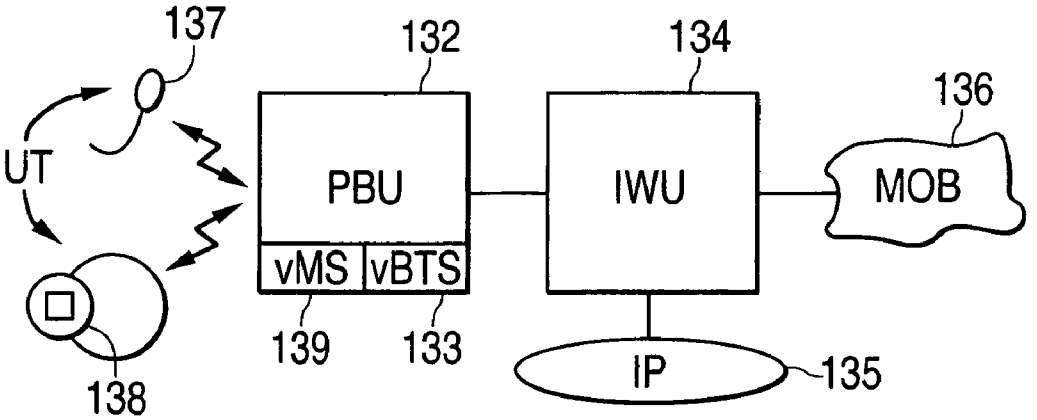

137
132
134
136
UT
PBU
IWU
MOB
vMS
vBTS
139
133
IP
135
138

FIG. 15

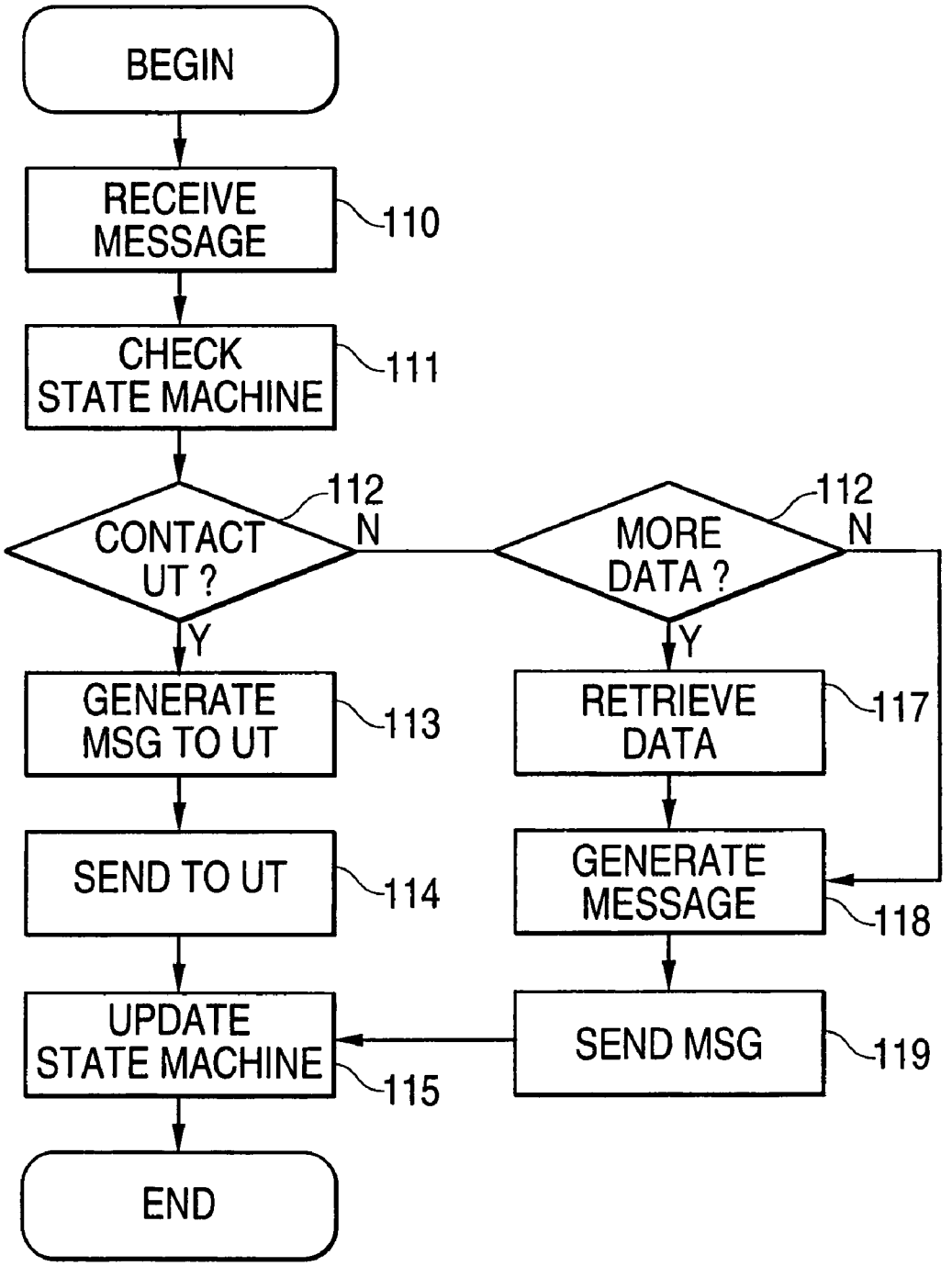
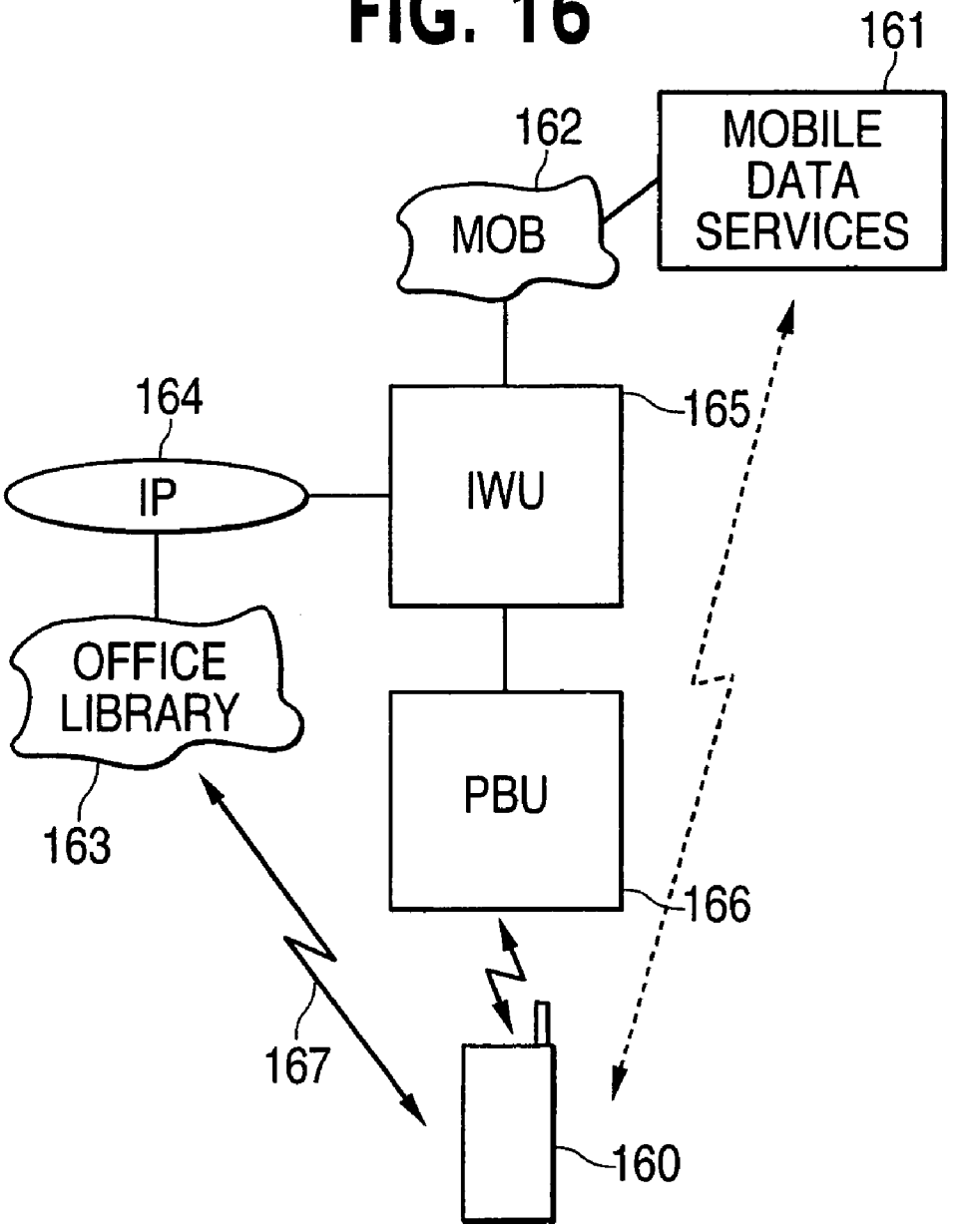

BEGIN
RECEIVE MESSAGE
110
CHECK STATE MACHINE
111
CONTACT UT ?
112
N
Y
MORE DATA ?
112
N
Y
GENERATE MSG TO UT
113
SEND TO UT
114
UPDATE STATE MACHINE
115
END
RETRIEVE DATA
117
GENERATE MESSAGE
118
SEND MSG
119

FIG. 16


161
MOBILE DATA SERVICES
162
MOB
164
IP
165
IWU
OFFICE LIBRARY
163
PBU
166
167
160

DUAL MODE TERMINAL FOR ACCESSING A CELLULAR NETWORK DIRECTLY OR VIA A WIRELESS INTRANET

CROSS REFERENCE TO RELATED APPLICATIONS

This application is a continuation of application Ser. No. 09/646,419, filed Nov. 30, 2000, now U.S. Pat. No. 6,853,851 which application is incorporated herein by reference in its entirety.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to a dual mode mobile station operable, for example in a public mobile communication network and a private network. The Invention also relates to the system in which such a dual mode terminal may operate, and further components of that system.

2. Description of the Prior Art

In modern office work it is necessary to provide the employees with versatile information transfer connections which can transfer speech, facsimile messages, electronic mail and other data—usually in digital form. Transfer of information is needed inside an office or corresponding working environment for communication between employees, for transfer of information between branch offices of an enterprise, which offices can be in other towns or even in other countries, and for communication between the office and "outside world". In this text and all of the following text "office" stands for an environment with several users, which users "belong together", and which office physically covers a reasonably limited area. There has been a trend in the telecommunication branch toward integrated systems in which various forms of telecommunication can be controlled as one entity.

A conventional realization of an above mentioned type of office communication system comprises a company telephone exchange for providing telephone services and telephones connected to it over twisted-pair connections and a separate local area network (LAN) in which applications for advanced telecommunication services have been implemented and which has the intelligence to run them. The local network is connected to the telephone exchange using a telecommunication server (Telephony Server) which supports the traditional subscriber server architecture in which subscribers are subscribers' computers connected to the local network. For example call-, data-, facsimile-, electronic mail- and speech mail services are connected within an office utilizing the telecommunication server. In an integrated system users can also e.g. control telephone services using their computer terminals connected to the local network. The whole integrated office communication system is connected to public telephone network through the telephone exchange.

FIG. **1** presents an example of a prior known office communication system in which users' telephones TP (TelePhone) have been connected by wire connections and a local area network (LAN) has been connected over a telecommunication server TS (Tele Server) into a telephone exchange PBX (Private Branch Exchange) which is connected to a public telephone network PSTN/ISDN (PSTN, Public Switched Telephone Network, ISDN, Integrated Services Digital Network). To the local area network (LAN) have been connected on one hand servers executing various services such as data base server DBS (Data Base Server), voice server VS (Voice Server) and electrical mail server EMS (Electrical Mail Server) and on the other hand the users' computers PC (Personal Computer). It can be regarded as a problem with this kind of realization that even if a user's telephone TP and computer PC usually are on the same table next to each other separate wire connections must be laid to the user's working room for them, on one hand from the telephone exchange PBX and on the other hand from the telecommunication server TS of the LAN. Building and maintenance of two overlapping telecommunication networks naturally causes cost.

The problem of overlapping telecommunication networks is increased by portable mobile stations utilizing radio connection coming rapidly more popular. Many persons working in an office need, because of their mobile work, a mobile station and often also a portable facsimile device and/or a combined portable computer/mobile station. In order to be able to use the devices based on radio connection also inside buildings, the constructions of which attenuate radio signals, it has been suggested that mobile radio networks should be supplemented with small base stations individual for offices or even for rooms, which base stations would be connected either directly or over wired telephone network to the central systems of mobile communication network. The network of small base stations would be already a third overlapping telecommunication network within the same office, and accordingly it is clear that in a preferable solution, which the present invention is aiming at, also the arrangement supporting radio communication stations should be realized using essentially the same means and telecommunication networks than the rest of the transfer of information in the office.

A challenge of its own to telecommunication systems is issued by the fact that work is done more and more in small office or domestic environment, which is described by the concept SOHO (Small Office, Home Office). Even here advanced office communication services are often needed and it is particularly preferable if such a flexible system is available which can be utilized even both in the office and at home. The present systems which require overlapping connections for the utilization of mobile communication services, conventional telephone services and fast data transfer services are very inflexible for working in a small- or home office. In addition to above, the following kinds of solutions connected with integrated telecommunication systems are known from prior art.

If an integrated office communication system is realized utilizing traditional technique, separate wired connections must be laid into a user's working room on one hand from telephone exchange PBX (FIG. **1**) and on the other hand from telecommunication server TS of local area network (LAN). Constructing and maintaining two overlapping networks naturally brings extra cost. In said solutions according to prior art a solution to this problem has not actually been striven for.

SUMMARY OF THE INVENTION

The present invention reduces the problems caused by overlapping networks. Additionally, the invention reduces problems caused by wireless information transfer inside an office and extra cost. The invention is an arrangement, in which said system, integrating information transfer, can also serve home office- and small office users. The invention is an arrangement of said kind, in which the carrier devices can be used as terminal devices (e.g. mobile stations) in the telecommunication system both in the office and outside it.

According to an aspect of the present invention, there is provided a dual mode mobile station comprising means for managing network information independently of the mode of operation of the mobile station; first linking means for linking to the interface of a mobile communication network so as to transfer control and mobility information between the mobile station and the mobile communication network; second linking means for providing a link to the interface of a further communication network so as to transfer control and mobility information between the mobile station and the further communication network; and means for coupling the managing means to the first linking means when the mobile station is in a first mode and to the second linking means when the mobile station is in the second mode.

This mobile station has common network layer information for both modes (i.e. when the mobile station is within and outside the wireless intranet office environment). Consequently, as there is no dual stack at this level, less code is required to implement the dual mode mobile station, hence making it simpler, faster and cheaper. It Is also easy to implement the second mode into existing mobile stations as this may be provided by virtue of a software enhancement to the conventional mobile station.

The network information is preferably at least mobile communication call control and mobility information. It may also further comprise mobile communication radio resources information. However, alternatively, the first linking means may comprise a radio resource manager for the mobile communication network, and the second linking means may comprise a radio resource manager for the further communication network. This may enable the mobile station to communicate with an interface on the further communication network by means of simple signalling. For example, the second linking means may comprise a radio resource of an unlicensed band such as a low power RF radio resource like Bluetooth.

In a preferred embodiment, the mobile station is further provided with a radio resource manager for a user terminal, and linking means for linking to the interface of the terminal device so as to transfer radio resource information between the mobile station and the user terminal. Furthermore, a mobile station may further comprise a browser, such as a WAP browser.

According to another aspect of the invention, there is provided a base station transceiver emulator for interfacing a mobile station of a mobile communication network and a further communication network, the base station transceiver emulator comprising means for determining the presence of a mobile station within its cell; transceiving means for receiving call transfer information from the mobile station when the mobile station is within the cell and for transmitting call transfer information to the mobile station as it prepares to leave the cell.

According to a further aspect of the invention, there is provided a mobile station emulator for interfacing a mobile station of a mobile communication network and a base transceiver station emulator of a further communication network, the mobile station emulator comprising means for receiving call transfer information from the mobile station and for forwarding it to the base transceiver station emulator, when the mobile station enters the cell of the base transceiver station emulator, means for maintaining the call transfer information while the mobile station is within the cell; and means for transmitting the call transfer information to the mobile station as it prepares to leave the cell.

Such an emulator enables simple signalling between the mobile station and base station transceiver emulator. Furthermore, it enables call forwarding. Moreover, it eliminates the need for a mobile station to be used once it has entered the wireless intranet office environment. For example, instead of using a mobile station when in the office environment, a user could use a lightweight terminal such as a wristwatch and headset instead, or indeed a PC with headset.

A device for coupling a mobile station of a mobile communication network to a further communication network may comprise a base transceiver emulator and/or a mobile station emulator. Preferably, the device is a personal base unit and comprises both of these emulators. Such a personal base unit may be implemented in a PC.

According to another aspect of the present invention, there is provided a system for transferring information between a mobile station and a further communication device, the system comprising the mobile station, a communication network to which the further communication device is coupled, and a base transceiver station emulator for interfacing the mobile station and the communication network, wherein the system transfers information over the communication network when the mobile station is within the cell of the base transceiver station emulator, and transfers information over a mobile communication network when the mobile station is outside the cell of the base transceiver station emulator.

A base transceiver station emulator and mobile station are also provided for such a system.

Such a system allows users to utilize communication networks, such as private intranets to carry cellular services (e.g. speech, data, SMS, facsimile etc.) when within a coverage area. In addition, the WIO concept provides a good platform for local multi-media extensions because it potentially offers higher bandwidth to the user. Access to the public cellular network (e.g. GSM) is offered by introducing a transparent location management method, which allows mobile stations connected to the communication network, such as the intranet, to be reached from the public cellular network in the normal way. Hence, the concept can be utilized to provide extra capacity in hot-spot areas, such as airports and malls.

The base transceiver station (BTS) emulator may be an actual base transceiver station or a virtual base transceiver station. In any event, it is an interface between the mobile station and the communication network over which the information (e.g., speech, data) is to be transmitted.

The BTS emulator may be the BTS of a mobile cluster. In this event, it is an actual base transceiver station. While a mobile station is within this BTS cell, the information to/from the mobile station is transmitted over the communication network, even if there is an overlap with the cell of another public GSM BTS.

Alternatively, the BTS emulator may form part of a personal base unit for a mobile station, in which case it is a virtual BTS. That is it looks like a BTS to the mobile communication network, but does not handover to another BTS.

In one embodiment, where the communication network is an IP network, the system takes care of the binding of GSM and IP numbers, so that only one number is required. Such E.164=IP# mapping may be performed in the IWU (e.g. by the gatekeeper or ILR, or alternatively in the personal base unit.

The communication system may be one of several kinds, such as a data communication network, internet, intranet, LAN, WAN, ATM packet network, Ethernet (™), or Token

Ring (™). Also, the further communication device may be one of several kinds, including a PBU, another mobile station, an MSC or an FSC.

The mobile station and PBU may be connected by RS232 cable. Alternatively, they may have an RF (preferably LPRF) or infrared connection. Examples include Bluetooth, Home RF, 802.11 WLAN etc. Also, they may be indirectly connected, for example via a connection device such as a mobile station cradle, deskstand or charger, or even a LAN of some kind.

According to another aspect of the present invention, there is provided a dual mode mobile station comprising control means for controlling transfer of information such that in a first mode transfer of information is between the mobile station and a mobile communication network, and in a second mode transfer of information is between the mobile station and a second communication network, and means for providing radio contact between the mobile station and the mobile communication network in both the first and second modes.

The first mode is, for example, when the mobile station is outside the office environment and the second, when it is within it,

In a preferred embodiment, the control means and means for providing radio contact are realized by virtue of a software enhancement to conventional mobile terminals. Hence, the terminals are much simpler than existing dual mode terminals, which, for example, require switches to change between the modes. Also, the terminal of the present invention remains connected to the mobile network while the actual data (data/speech etc.) is carried over another interface. Thus it provides the mobile network with what seems to be the same operation specified for the standard mobile communication network entities.

Now a system has been invented for transfer of information, e.g. speech or data, in which the trunk of information transfer is inside the office a local network (e.g. local area network, LAN), and between office units e.g. a traditional telephone network utilizing wired connections or a fast data packet network utilizing ATM (Asynchronous Transfer Mode) technique, for example.

According to one embodiment of the invention the mobile station may be connected to the terminal device by means of a connection device, having a functional connection to the terminal device, and having means for connecting functionally to the mobile station. In response to connecting a mobile station to the connection device, the system will be informed to direct calls to the mobile station via the data communication network. The connection device can be a desktop stand or desktop charger and may be a separate device or integrated into the terminal device.

A subscriber device means a terminal device connected to a telecommunication network, such as a telephone connected to a fixed telephone network, and a mobile station connected to a mobile communication network. A subscriber device also means servers and telephone exchanges connected to telecommunication networks, providing telecommunication services to the users of the telecommunication networks. In other words, a subscriber device means all the parts of a telecommunication network with which a telecommunication terminal device (e.g. a telephone) can communicate over a telecommunication network.

BRIEF DESCRIPTION OF THE DRAWINGS

Embodiments of the present invention will now be described, by way of example, with reference to the accompanying drawings, of which:

FIG. **1** presents traditional communication networks and terminal devices used in an office environment;

FIG. **2** illustrates a wireless intranet office architecture according to an embodiment of the present invention;

FIG. **3** illustrates a wireless intranet office architecture according to an embodiment of the present invention;

FIG. **4** illustrates the architecture of a mobile station and personal base unit of a wireless intranet office, according to an embodiment of the present invention;

FIG. **5** illustrates a general GSM intranet office concept;

FIGS. **6** to **11** show information flow from terminals of a GSM intranet office according to an embodiment of the present invention;

FIG. **12** illustrates the architecture of a mobile station according to a further embodiment of the present invention;

FIG. **13** illustrates a wireless intranet office system according to an embodiment of the invention in which the user is provided with a handsetless user terminal which communicates with his mobile station;

FIG. **14** illustrates a wireless intranet office system according to a further embodiment of the present invention, in which the user is provided with a handsetless user terminal which communicates directly with the personal base unit; and

FIG. **15** is a flow chart illustrating the functioning of a virtual terminal in the embodiment of FIG. **14**, and

FIG. **16** illustrates the handling of an electronic book service within a wireless intranet office.

FIG. **2** illustrates a wireless intranet office architecture according to an embodiment of the present invention.

As can be seen, the wireless intranet office integrates an IP based private intranet environment with a public cellular network, in this case the GSM network. This allows cellular users to utilize private intranets to carry the cellular services (i.e. speech data, SMS, facsimile, etc.) within the intranet coverage area. In addition, the wireless intranet office architecture provides a good platform for local multimedia extensions because it potentially offers higher bandwidth to the user. Access to public GSM network is offered by introducing a transparent location management method, which allows terminals connected to the intranet to be reached from the public GSM network in the normal manner. Thus, the wireless Intranet office arrangement can be utilized to provide extra capacity in hot spot areas, such as airports, malls etc., where this might be needed.

In this wireless intranet office arrangement, the intranet forms a new kind of access network to the GSM network. The communication between the GSM backbone network and the end user access node takes place via internet protocol based networks instead of the GSM air interface, as will be seen below,

FIG. **2** shows a mobile station **21** in a wireless intranet office environment. When outside this environment, the mobile station acts as a normal GSM phone connecting to a BTS of a public GSM network. However, when in the wireless intranet office environment, the mobile station may operate on one of two modes. In one mode, it connects to a personal base unit **22** (e.g. either with a inter-connection cable, a infrared connection, or with low power RF transmitter and receiver), and in another mode connects to a GSM base transceiver station (BTS) **23**. The mobile station **21** is connected to an IP local area network (LAN) and a home

location register (HLR) and visitor location register (VLR) **25** and a mobile station controller (MSC) **26** by virtue of an inter-working unit (IWU) **24**. This IWU comprises several network entities, including an intranet mobile cluster (IMC) GSM/IP Gateway **241**, an intranet location register (ILR) **242**, a WIO gatekeeper **43** and a WIO A-gateway **244**.

Information such as data and/or speech may be transferred from the mobile station to the IP local area network by two routes, each of which includes a BTS emulator. In a first mode, the mobile station **21** is connected to the local area network via a personal base unit **22** (PBU), which itself comprises a virtual BTS. This is further explained with reference to FIG. **4** below.

In a second mode, the mobile station **21** forms part of a mobile cluster (for example see reference **32** in FIG. **3**). In this case, the information is transmitted to the local area network via a private GSM BTS **23** dedicated to that cluster, and an IMC GSM/IP Gateway **241**. The BTS transmits the signal over the A bis interface, and the IMC Gateway **241** performs a protocol transform from GSM to H.323, so that the signal can be transmitted over the IP local area network. (As can be seen from this figure, the wireless intranet office architecture uses the H.323 protocol for the signalling and data connections inside the inter-working unit).

The basic access interfaces to the cellular network are the air interface, the A-interface, the MAP protocol, the ISUP/TUP interface and the DSS.1 interface. The A-interface is an interface to mobile switching center and the MAP interface is an interface to HLRNLR. ISUP/TUP interface connects switching centers, while the DSS.1 interface resides in between the BSX and switching center. The air interface connecting mobile terminals to the network can be any RF interface or infrared link. Candidate RF interfaces include e.g. Low Power RF (LPRF), 802.11, wireless LAN (WLAN) WATM and HIPERLAN. The air interface can also be replaced with a physical connection (e.g. RS-232 serial cable or Universal Serial Bus (USB). The GSM network sees this new access network as a BSS entity. New network entities are added to the access network to modify/de-modify cellular signalling. System design principle is to fulfill ITU-T's recommendation H.323 and enhances it with mobility extensions.

The WIO A-gateway **244** looks like a base station controller to the MSC **26**.

A general WIO network architecture is shown in FIG. **3**. A local area network **31** is provided with an intranet mobile cluster IMC **32**, an LPRF cell **34** and a landline connection **35**. The IMC comprises a plurality of mobile stations, a BTS (private GSM BTS) and a server in the form of an IMC GSM/IP gateway. The BTS interface between the BTS and IMC GSM/IP gateway is a GSM A-bis interface. The IMC GSM/IP gateway is responsible for signalling conversions between the GSM and H.323 protocols. The low power RF cell **34** comprises a personal base unit which has a virtual BTS and a low power transceiver, and associated mobile stations with corresponding low power RF transceivers. The PBU is directly connected to the WIO network. To provide the mobile stations with access to the GSM network, the PBU provides conversions between the GSM and H.323 protocols. These conversions can be seen as a bridge between cellular phone and H.323 features which support WIO location management and mobility features. The landline connection comprises a landline terminal **351** hardwired to a personal base unit **352**, which in turn is hardwired to the local area network.

Also connected to the local area network are a WIO gatekeeper **36**, which is responsible for the connection of mobile stations to within and outside the network. For example it might transfer a call from the server to an external system such as PSTN (via gateway **38**) or it could provide connection to the IP network **37**. The IP network, in turn, is connected to the operators local area network **39**. This local area network is provided with an A-intranet gateway **391**, an intranet Location Register **392** and IP telephony gateway **393**.

In this embodiment the main function of the Intranet Location Register **392** is to store mobility management information and call statistics of the subscribers configured into the Wireless Intranet Office system. Roaming of visitors is controlled by the mobile switching center. For visitors only temporary information will be stored into the Intranet Location Register.

The ILR has a MAP interface to cellular system network HLR **25**.

The IP Telephony Gateway **383** in this embodiment supports interworking between Internet telephony endpoints and mobile stations in the public cellular network. The interworking is based on the H.323 specifications.

The A-Intranet Gateway **391** in this embodiment makes protocol conversion between SCCP/MTP and IP protocols at the A-interface, and makes the cellular and Intranet location area associations. It has an O&M software entity which operates as an administrative server gateway for corresponding agents in intranet mobile clusters. The A-intranet Gateway operates as a firewall between public telecommunication network and private intranet solutions,

Further explanation of the network entities in FIGS. **2** and **3** are outlined below.

The Intranet Mobile Station is a generic terminal product portfolio consisting of full-featured cellular phone which supports services of GSM and GSM derivatives. It may have specific features such as extended office/home cell selection criterias, and support of office and home area priority. With a serial cable and with a piece of software to a PC, intranet mobile station—so called LANdline version—enables seamless landline communication to cellular system network and between other Internet telephony entities within IP network. It may be a GSM/LPRF dual-mode device enabling high value services within certain service areas.

The Personal Base Unit (PBU) may be a PC Card type of radio card for a desktop PC with a piece of software enabling wireless access to IP network. It provides LPRF cordless and wireless LAN—on 2.4 GHz band—dual-mode access exploiting an unlicensed radio spectrum. In cordless, "unlicensed" mode lower layers will be replaced with new ones, but signalling above them remains the cellular one. It also enables intelligent roaming of terminals between different radio frequency bands, i.e. between cellular and unlicensed bands.

The Intranet Mobile Cluster is simulating BSC in a local environment. It consists of minimum set of BTS functionality with reduced physical construction. Intranet mobile cluster is a BTS arid a BTS driver software package for Windows NT 5.0 including rate adaptation, an O&M agent software package and a GSM/IP telephony gateway entity. Intranet mobile cluster provides interworking with data services and facsimile as a direct access to IP network, and it may provide local call routing capability within its radio coverage.

The purpose of the GSM/IP Telephony Gateway is to reflect the characteristics of an Internet telephony endpoint to an Intranet Mobile Station, and the reverse, in a transparent fashion. The GSM/IP telephony gateway provides appropriate format translation of signalling and speech. i.e.,

audio format translations between GSM 06.10, 06.20, 06.60, J-STD-007 and G.711, G.723 and transformation of communications procedures. The gateway performs call setup and clearing on both the Internet telephony side and the Wireless intranet office side.

The MS-IP (WIO) gatekeeper 36,243 provides mobility and call management services, and certain radio resource management functions.

The MS-IP gatekeeper provides the following services:

Registration control—The MS-IP gatekeeper authenticates all the network entities, i.e., intranet mobile stations, intranet mobile clusters, A-intranet gateways, IP telephony gateways, intranet location registers, H.323 terminals, which have access to the system. In case of intranet mobile station, authentication and registration is based on automatic Gatekeeper discovery procedure. In other cases, it's based on manual gatekeeper registration procedure.

Connection ciphering—Part of the gatekeeper's authentication procedure is connection ciphering service. It provides key distribution, identification and encryption/decryption services to the gatekeeper and other entities in the system. Service has an option to select ciphering, hashing, key distribution and signature algorithms independently. Key distribution is based on public key cryptography and message ciphering is based on secret key cryptography.

Address translation—The MS-IP gatekeeper performs E.164 to transport address association and translation. This is done using directory service in the intranet location register which is updated during mobility management procedures, i.e., during TMSI reallocation, authentication, identification, IMSI detach, abort, and location updating.

Call control signalling—The MS-IP gatekeeper can be configured to route call control signalling to the cellular system network or to the local call management entity within the gatekeeper.

Call management—The MS-IP gatekeeper maintains also list of ongoing calls and collects call statistics. This information is stored into the intranet location register by the gatekeeper.

Cellular procedures—The MS-IP gatekeeper must be able to handle signalling and resource management procedures (BSSMAP resources) specified in GSM recommendation 08.08.

Status control—In order for the MS-IP gatekeeper to determine if the registered entity is turned off, or has otherwise entered a failure mode, the MS-IP gatekeeper uses status inquiry to poll the entity at a certain interval.

The MS-IP gatekeeper may, for example comprise software which uses a Windows NT platform together with some dedicated hardware in the IMC and gateways to fulfill the ITUT's H.323 gatekeeper specifications, extended with certain mobility management capabilities according to GSM 04.08.

FIG. **4** shows the architecture of a mobile station **41** and a personal base unit, personal computer **42**, according to an embodiment of the present invention.

The mobile station **41** and personal base unit **42** are represented showing layers **1** to **3** of the 7 layer OSI reference model, namely physical layer (layer **1**), data link layer (layer **2**) and network layer (layer **3**). (These are data communication protocols whose purpose is to provide a link between 2 communicating devices).

Network layer **43** of the mobile station **41** provides call control management **431** (including supplementary services **435** and short message services **436**). This layer also provides mobile management **432** and radio resource management **433**. Further, It comprises a MUX which "switches" to a second branch of layer **2** to demand services of the data link (phone bus FBUS) Ctrl **443**) and physical layer (FBUS **452**) when the mobile station **41** and the personal base unit **42** are "connected". In any event, the network layer demands the services of the data link layer **44** (data link **441** and control **442**) and the physical layer **45** of the first branch, to allow the mobile station **41** to perform and report its measurements about the surrounding GSM network (neighboring BTSs) and thus comply with GSM requirements.

Turning now to the personal base unit **42**, this PBU comprises a phone driver implementing the physical and data link layers **48** and **47** (FBUS **481** and FBUS ctrl **471**). The network layer **46** of the PBU comprises a PBU control/IMC core control **462** and an H.323 protocol entity **463** which provide protocol conversion between GSM and H.323. The conversions are needed for GSM layer **3** signalling messages while the speech is carried as GSM coded in the whole while this intranet office network. The PBU further comprises TCP/IP entity **422** and a local area network adapter driver for the **23** for interfacing with the local area network. The PBU control **462** comprises a virtual BTS **49** for communicating with the network layer **43** of a mobile station **421**.

This figure shows layers **1** and **2** of the second branch of the mobile station and the PBU as a phone bus (FBUS). This is because, in this embodiment an RS 232 serial connection is used. However, it is evident to a person skilled in the art that these layers would be implemented using different technologies if, for example, connection is via IR or RF.

The mobile phone also has a user interface **461**.

In the network, the mobile station interlaces the intranet mobile cluster and personal base unit entities. The interface to the personal base unit, as can be seen from FIG. **4**, uses a modified GSM layer **3** (04.08) signalling in this embodiment. (However, in an alternative embodiment, shown in FIG. **12**, the GSM radio resource is not delivered to the PBU from the mobile station. Instead, Bluetooth radio resourcing replaces it as a consequence of part of the virtual terminal being implemented inside the mobile station control software).

The mobile station **41** and PBU **42** operate as follows.

When the mobile station **41** is outside the wireless intranet office environment, it operates as a normal GSM phone. The MUX **434** does not couple the radio resource management entity **433** with the second branch **443**, **452**. Voice and signalling is transmitted via the data link layer **44** and physical layer **45** over the first (GSM) branch to the cellular air interface.

Also, if the mobile station **41** is within the wireless intranet office, but forms part of an intranet mobile cluster, this same path is taken to the cellular air interface and the information and signalling is transmitted to the GSM BTS of that cluster.

However, when the mobile station **41** is connected to a PBU **42** (for example by an RF232 serial cable or RF interface) information such as voice, data, fax, SMS etc., is transmitted over the local area network. In this case, the MUX **434** demands the service of the second (LAN) branch layers **1** and **2**, and layer **3** of the mobile station **41** is seen to communicate with the virtual BTS **49** of the PBU **42**. That is, the information (e.g. speech) and GSM layer **3** signalling messages are redirected to the second branch interface. As the mobile station **41** and the PBU **42** are linked, the field strength of the virtual BTS **49** will be greater than that of other BTSs in the GSM network. Consequently handover is made to the virtual BTS **49**. After this, the handover signalling relating to this virtual BTS is handled from the MUX

through the second branch. When handover has been made, the MUX handles all messages and forwards them to the new host cell through the RS 232 interface etc and "talks" to the other BTSs (as is conventional in GSM) over the first branch. General broadcast traffic is also seen by the mobile station **41**, for example from layers **1** and **2** to the MUX and from there through the mobile station/PBU interface to the virtual BTS **49**.

While in this mode, the speech and layer **3** signalling are routed to the personal base unit, and the radio resource management entity at layer **3** remains connected to the GSM layer **2** (**441**), that is branch **1**. As mentioned above, this is so that the mobile station can act as required by GSM (for example by measuring the RSSI for neighboring BTSs etc.).

The parameters in the virtual BTS **49** within the IMC core are set in such a manner that the terminal is forced to remain clamped to this virtual OSM cell. This avoids possible handovers to any other GSM cells the mobile station might hear.

The operation of the MUX can also be explained as follows. When the mobile station changes to "LANdline" mode (for example when the other interface is connected), the MUX communicates with the new BTS in a similar way to as it does to other BTSs to which it is not connected. In this phase, the mobile station notices that the field strength of the new BTS relating to this new interface is more powerful than the field strength of other BTSs, and hence makes the handover to this BTS. After the handover, signalling relating to the new BTS are handled by the MUX through the new interface, and the mobile station keeps on listening these and sends measurement reports to virtual BTS general broadcast traffic is also sent to the new mobile station, for example from the lower stage to the MUX and from there through the new interface to the virtual BTS.

FIG. **5** shows a general GSM intranet office concept, and FIGS. **6** to **11** show information flow between terminals—FIGS. **6** to **9** being within the office environment and FIGS. **10** and **11** extending to outside the environment.

FIG. **5** shows the GSM intranet office **57** comprising different terminal arrangements **51** to **54**. The intranet office interfaces with an internet protocol network **58**, which is partially situated within the office and partially at the operators location. The operator **59** controls transfer of information between the IP network **58** and network switching centers, such as mobile switching centers **55** and fixed line switching centers **56**.

Terminal arrangements **51** and **52** comprise a mobile station **51***a*, **52***a* and a BTS emulator **51***b*, **52***b*. These mobile stations can be within an intranet mobile cluster or can be coupled to a personal base unit comprising a virtual BTS.

FIG. **6** illustrates a call between mobile stations of the same office. In this case, the call might be sent by mobile station **51***a* to mobile station **52***a*. The information is transmitted from mobile station **51***a* to BTS emulator **51***b* and on to the LAN via the inter-working unit. The local area network then transfers the information to BTS emulator **52***b* which in turn forwards it to mobile station **52***a*.

FIG. **7** shows a call between a mobile station **51***a* and an H.323 terminal **54** within the same office. Information transferred from mobile station **51***a* will be forwarded to the LAN in the same manner as in FIG. **6** (i.e. via BTS emulator **51***b* and the WIO inter-working unit). The LAN then transfers the information to the terminal **54**.

FIG. **8** shows a call between a mobile station **52***a* and a fixed line extension **53***a* of a private branch exchange **53***b* of the same office. Again, information is transferred from mobile station **52***a* to a local area network via BTS emulator **52***b* and the office IWU. The information is then transferred over the local area network via a PSTN gateway to PBX **53***b*. This PBX then switches the information to the requisite extension **53***a*.

FIG. **9** shows a call between a H.323 terminal and a PBX extension of the same office. In this case, there is no GSM connection. Information is forwarded to the local area network from the terminal **54** where it is transferred to PBX **53***b* via the local area network on a PSTN gateway. The PBX **53***b* then switches the information to the requisite extension **53***a*.

FIG. **10** shows a call between a mobile station **51***a* of the WIO to the mobile network. In this case, information is transferred from mobile station **51***a* to the local area network via the BTS Emulator **51***b* and the inter-working unit. It is then transferred across the local area network and to a mobile switching center **55** via an A-gateway.

FIG. **11** shows a call between a mobile station **52***a* of the WIO and a fixed line network. In this case, information is transferred from mobile station **52***a* to the local area network via BTS emulator **52***b* and the inter-working unit. The information is then transferred over the LAN to a fixed line switching center **56** via a PSTN gateway.

In the information transfer system according to the invention, information transfer connections based upon ATM and GSM technologies may been utilized. Furthermore, it is fully possible to utilize instead of these techniques other kinds of information transfer connections. For example it is possible to arrange, instead of the ATM system, the information transfer connections between terminal devices **40** to **43**, teleservers **60**, **61** arid network server **90** entirely e.g. using systems based upon Ethernet and Token Ring or future wide band networks. Correspondingly it is possible to realize, instead of GSM-system, an information transfer system according to the invention even in connection with other mobile communication systems, such as e.g. TDMA (Time Division Multiple Access), CDMA, W-CDMA AMPS (Advanced Mobile Phone Service) and NMT (Nordic Mobile Telephone) systems.

Moreover, it can be transferred over WATM, 802.11 and mobile IP, which allows the network entities (PBU, IMC, etc.) being mobile. This enables, for example, forming a WIO cluster/IMC into a train or airplane.

FIG. **12** shows the architecture of the mobile station **120** according to another embodiment of the present invention. This mobile station is provided with both GSM and LPRF (Bluetooth) parts (processors, RF parts etc.), and communicates with the public mobile network using GSM, and the PBU of the WIO network using LPRF (Bluetooth). An example of communication using Bluetooth is described below with reference to a user terminal and PBU in FIG. **14**.

The mobile station **120** of this embodiment is represented showing layers **1** to **3**, namely physical layer (layer **1**) **121**, data link layer (layer **2**) **122** and network layer (layer **3**) **123**.

Network layer **123** of the mobile station **120** provides call control management **124** (including supplementary services **124***a* and short message services **124***b*) and mobile management **125**. That is, these layer **3** network management services are common to both GSM and Bluetooth modes of operation. This network layer further comprises a multiplexer, MUX **127**, which demands services of the layer **3** radio resource management **126** and also of the lower layers **121**, **122**. In this embodiment the MUX **127** connects to a second branch of layer **3**, to the Bluetooth radio resource management **126***b*, to demand services of the Bluetooth radio resource management **126***b*, data link (DL and CTRL **128***b*, **128***d*) and physical layer **129***b*, when the mobile station **120** is within the wireless intranet office environ-

ment. The call control and mobility management functions **124** and **125** of the network layer also demand the services of the GSM radio resource management part **126***a*, the data link layer (DL CTRL **127***a*, **128***a*) and the physical layer **129***a* of the first branch via the MUX **127**. This allows the mobile station **120** to perform and report Its measurements about the surrounding GSM network (neighboring BTSs) and thus comply with GSM requirements and also to communicate with a virtual BTS within the WIO if applicable.

When the mobile station **120** Is outside the wireless intranet office, the common network layer functions demand the services of the layer **3** GSM radio resource management **126***a* and services of the lower layers **128***a*, **128***c*, **129***a* of the first branch (GSM branch).

FIG. **13** illustrates a wireless intranet office arrangement according to another embodiment of the invention.

In this arrangement, a mobile station **131** connects to a PBU **132** which may, for example, be a personal computer. The PBU **132** comprises a BTS emulator in the form of a virtual BTS **133**. A radio connection is shown (e.g. infrared or LPRF) between the mobile station and PBU, but the connection may be a different type such as a wired connection. The mobile station **131** is connected to an IP LAN **135** and mobile communications network **136** by virtue of an IWU **134**. The IWU may comprise several entities such as a GSM/IP gateway, an intranet location register, a WIO gatekeeper and a WIO A gatekeeper, as mentioned above with reference to FIG. **2**. Rather than having to carry the mobile station around, the user is provided with a user terminal in the form of a wireless headset **137** and wristwatch user interface **138**. The wireless headset **137** connects to the mobile station **131** over an air interface using LPRF remote audio protocol (e.g. Bluetooth), and the wristwatch UI **138** is similarly connected over the air interface using LPRF remote user interface protocol (e.g. Bluetooth).

The mobile station **131** of this embodiment, like that of FIG. **12** has both GSM and LPRF (e.g. Bluetooth) parts. However, as explained above, in this embodiment Bluetooth is used for communication between the mobile station **131** and the user terminal **137**, **138**, as opposed to between the mobile station **131** and PBU **132**. Consequently, the mobile station's protocol stack will differ from that shown in FIG. **12**. More specifically, the Bluetooth physical layer **129***b* will couple to the air interface of the user terminal as opposed to that of the PBU. Moreover, layers **1** and **2** of the GSM protocol stack will be distinguished. That is, this first branch **127***a* is further divided by the provision of a MUX between layers **2** and **3** as shown in FIG. **4**, depending on whether an interface is required to a GSM BTS or to a virtual BTS within a WIO environment.

When the handset **131** is outside the wireless intranet office environment, the handset **131** operates as a normal GSM phone. That is, MUX **127** connects to the GSM radio resource management **126***a* and the GSM lower layers **121** and **122** to obtain connection to a public GSM BTS. The other layer **1** and **2** stack linking to the virtual BTS would be disconnected as described above with reference to FIG. **4**.

Optionally, the MUX **127** may also make connection to the Bluetooth radio resource management **126***b*, for example if the user selects an option to use user terminals **137**, **138** within the GSM environment.

When the handset enters the wireless intranet office environment, on the other hand, the MUX **127** may effect a connection so that the call control and mobility management functions may demand services of the Bluetooth radio resource function and layers **1** and **2**, **126***b*, **128***b, d*, **129***b*, either automatically or upon user selection. Such connection enables the provision of a communication channel between the mobile station **131** and the user terminal **137**, **138**. To effect a link between the mobile station and PBU **132**, the MUX **127** connects the GSM radio resource function **126***a* to the common layer **3** functions, namely call control **124** and mobility management **125**. The GSM radio resource function **126** will demand service of layers **1** and **2** of the stack for linking with the PBU when in this wireless intranet office environment. Further, the GSM network will require signalling updates. Hence, layers **1** and **2** linking to both the GSM, BTS air interface and PBU air interface are connected.

FIG. **14** shows an alternative embodiment of the invention, in which user terminals **137**, **138** communicate directly with a personal base unit, when in the wireless intranet office environment. The system is similar to that shown in FIG. **13**, but with one major difference. When the mobile station **131** is in the wireless intranet office environment, its functionality is transferred to the PBU **132**. That is, the PBU **132** then comprises a virtual mobile station **139**, as will be explained further below. As a consequence, the user terminal **137**, **138** can communicate directly with The PBU **132**, thereby eliminating the need for the mobile station to remain turned on.

When the mobile station MS changes over to the WIO mode, the mobile station **131** transfers the dynamic data relating to the state of the mobile station and the calls in progress to a virtual terminal vMS **139**, which is established in the PBU **132**.

This data is maintained in a state machine, which is located in the virtual terminal. In this context, the state machine means a functional entity that describes the allowed changes in the state relating to the functioning of the mobile station and the related messages according to the protocol. The functionality described by the state machine maintains the data on the possible changes in the state relating to said protocol layer, the instantaneous state, the data structures relating to the change in the state, etc. Thus, a state machine in connection with the GSM means the mobile station's functionality related to the mobile station's GSM Layer **3** protocol (NULL, current switched on, switched to a base station, etc.) In addition, said state machine in the higher level maintains a partial state machine for the mobile station's every connection, whereupon the state of the connection can be, for example, NULL, call initiated, call proceeding, active, etc.

The protocol stack of the virtual terminal vMS in PBU may comprise the GSM functionality described by a state machine **105**, which comprises at least a radio resource (RR), mobility management (MM) and call management (CM), i.e. functions related to protocol layer. It may also comprise an additional protocol **106** relating to communication between the PBU and the user terminal operating in the WIO mode (e.g. the Bluetooth functionality). This will be discussed later in more detail.

When the PBU **132** has the use of the data of the state machine, the PBU starts the virtual terminal vMS **139**, which emulates the functioning of the actual mobile station MS towards the mobile communication system. It receives signals from the mobile communication network and, on the basis of the status data it maintains, it carries out signalling towards the mobile communication system, either independently, or according to the information it requests from the user terminal UT in WIO made. It should be noted that since the state machine during WIO mode is maintained by the virtual terminal, the signalling to be implemented in different directions is independent, which means that changing of

the protocol in either direction does not interrupt the functioning of the virtual terminal.

The flow diagram presented FIG. **15** illustrates the functioning of a virtual terminal on the basis of a message arrived from a mobile communication network. In step **110**, the virtual terminal VMS **139** receives a message from the mobile communication network MOB. In step **111**, the virtual terminal vMS compares the contents of the message to the state machine it maintains and, on the basis of it, defines the message required for changing over to the next state. In step **112**, the virtual terminal defines whether a connection to the user terminal UT that operates in the WIO mode is required for generating the next message or whether the required data is available in the inter-working unit. If a connection to the user terminal UT is necessary, the virtual terminal generates the message relating to said function (step **113**) and sends it through the IP network to the user terminal UT (step **114**). At the same lime, it updates the state of the process in question to the signalling state maintained by it (step **115**). If no connection to the user terminal UT is required and the virtual terminal concludes that the necessary signalling can be managed by itself, the virtual terminal checks whether the subscriber information stored in the PBU is required for the reply or whether the reply message can directly be generated on the basis of the status data (step **116**). If additional information is required, the virtual terminal retrieves it from PBU's memory (step **117**) and, on the basis of it, generates a message to be transmitted to the mobile communication system (step **118**). If no additional information is required, the virtual terminal generates a message in accordance with the mobile communication system's protocol defined on the basis of the status data (step **118**). In step **119**, the message generated by the virtual terminal is transmitted to the mobile services switching center. At the same time, the virtual terminal updates the state of the process in question in the state machine it maintains (step **115**).

One way of managing a connection between the virtual terminal vMS **139** and the user terminal UT in WIO mode is to convert the GSM signalling into packets in accordance with the IP and to transfer the signalling to the user terminal UT in the GSM format. Anyhow, information transferred between the mobile communication network and the user terminal UT includes a lot of signalling relating to the use of a radio resource. Such traffic in the arrangement according to the invention is substantially unnecessary. Hence, in this embodiment, a connection is managed by simplifying the protocol during WIO operation. This kind of protocol can be established, for example, by selecting a group of AT commands, which are transported between the vMS and the MS in WIO mode. For the establishment of a connection, a simple, manufacturer-specific protocol can also be defined.

The implementation of said protocol could be illustrated by giving an example of the different functions, which are needed for communication between the vMS **139** and the UT in WIO mode. These include, for example, the functions 1.1.-1.7. listed In the first column of Table 1. The second column of Table I contains a functional description of messages.

TABLE I

Reference	Function	Messages
1.1	Making of Call	Request to Call MS->vMS Resetting of Request to Call vMS->MS

TABLE I-continued

Reference	Function	Messages
1.2	Reception of Call	Indication of Call vMS->MS Resetting of Indication of Call MS->vMS
1.3	Speech	Transport of Coded Speech Over UDP
1.4	Ringing Out	Request for Switching Off/Indication
1.5	SMS	SMS Transmission/Reception
1.6	FAX	Telecopy Transmission/Reception
1.7	Handover	Handover Message Transmission/Reception (State Machine)

When a subscriber wants to make a call (1.1), a user terminal UT makes a request for a call and receives the message of the setup of the call given by an vMS, before the transfer of the data relating to the call begins. When the subscriber receives a call (1.2), the user terminal UT receives the message of the incoming call from the vMS and informs the vMS of the reception of the call before the transfer of the data relating to the call begins. When either the subscriber or the other party wants to cut off the call (1.3), the user terminal UT gives or receives a request to cut off the call. On the basis of the protocol, both the user terminal UT and the VMS should be able to distinguish whether it is a question of the transfer of speech (1.4), a short message (1.5) or telecopied data (1.6). The message 1.7 contains the status data on the calls in progress, which are transported when the virtual terminal is taken into use or when the use of the virtual terminal is terminated as described above.

The above-mentioned command group is only one possible way of implementation. For example, making a call can be arranged so that the user terminal UT identifies, on the basis of the first speech packets, that a call is coming in, in which case not even a separate call phase is required. Correspondingly, the vMS can automatically adapted to cut off the call when the reception of the call packets from the user terminal UT stops. With a simple command group, it is possible to implement adequate functions by means of which the user terminal UT that operates in the WIO mode can utilize the mobile communication network's services, though part of the signalling is managed elsewhere.

Referring back to FIG. **14**, when a user enters the office carrying his traditional user terminal UT handset, the phone indicates that LPRF LAN access is available. When the subscriber so wishes, he/she can e.g. plug the handset into an intelligent charger such as described in PCT Publication Number WO98/15143, and thus enable "handsetless operation" using merely the wrist UI and wireless headset. In such an operation, the traditional terminal is inactive and the virtual terminal acts as a mobile station towards the mobile communication network. The traffic between the lightweight terminal and the virtual terminal is carried out through LPRF connection using the specific protocol layer as described earlier. While in the office, he/she can walk around the LPRF coverage area and use GSM services without the handset When leaving the office he/she can enter normal cellular operation by just taking his/her handset along and even continue the ongoing call. The invention thus facilitates a completely wearable communications device in office environment with the user identified as the same mobile subscriber as outside office with handset. The phone numbers, user setting, personalized features etc. will remain in both operating modes.

FIG. **16** illustrates the handling of an electronic book service within a wireless intranet office, according to a preferred embodiment of the invention. The system may utilize a dual mode terminal of the invention as is shown for example in FIG. **12**. Mobile data services are becoming increasingly prevalent from mobile communications operators. One such service may be electronic book (E-book) purchasing. In this embodiment, E-book purchasing **161** is available through the operator of a mobile communications network **162**. The user of mobile station **160** can access this service either via the public mobile network **162**, or via the WIO. In the latter case, connection to the mobile network **182** is via PBU **166** and IWU **165** as explained with reference to FIG. **2** above. Similarly, the book required may be downloaded via the public or private networks. In the event that the mobile station **160** is within the public mobile communications area **162** but outside the WIO environment, the book may be stored in the mobile station's memory (or if the mobile station is a portable computer with data card, then it may be stared an the computer's hard disk, for example). Ideally, this is a temporary measure, and the book can subsequently be transferred for storage within the WIO network when the mobile station enters the WIO environment. For example, the user could choose to store the E-book on his PC (PBU **166**), or alternatively in an office library **163** of the offices IP LAN **164**. Alternatively, if the mobile station is within the WIO environment, the user may request the E-book via the WIO network, and the book may automatically be downloaded to the requested WIO device (e.g. office library **163**, or users PC). An advantage of storing an E-book in the office library **163** is that it is accessible to other users of the office. Consequently, if the user's terminals (mobile station, PC etc.) have a suitable browser, the user can search through books, newspapers etc. for desired information. If the user's terminal is a PC, then a conventional IP browser may be used. Alternatively, if the user's terminal is a mobile station **160**, such as a mobile phone, then it is preferably provided with a WAP browser so that it may search the contents of the library **163**, over a tow power RF interface **167** such as Bluetooth.

The above is a description of the realization of the invention and its embodiments utilizing examples. It is self evident to a person skilled in the art that the invention is not limited to the details of the above presented embodiments and that the invention can be realized also in other embodiments without deviating from the characteristics of the invention. The presented embodiments should be regarded as illustrating but not limiting. Thus the possibilities to realize and use the invention are limited only by the enclosed claims. Thus different embodiments of the invention specified by the claims, also equivalent embodiments, are included in the scope of the invention.

For example, while the embodiments refer to intranet offices, it is not restricted to the intranet, but is also applicable to the internet.

The invention claimed is:

1. An apparatus, comprising:

a managing module adapted to manage information independently of the mode of operation of a mobile station;

a first linking module adapted to link to the interface of a mobile communication network, said first linking module comprising a radio resource manager for the mobile communication network;

a second linking module adapted to provide a link to the interface of a further communication network, said second linking module comprising a radio resource manager for the further communication network; and

a coupling arrangement adapted to couple the managing module to the first linking module when the mobile station is in a first mode and to the second linking module when the mobile station is in a second mode such that, when the mobile station is in the second mode, the mobile station remains connected to the mobile communication network, while actual data is carried over the further communication network.

2. The apparatus as claimed in claim **1**, wherein the managing module further manages radio resources information independently of the mode of operation of the mobile station.

3. The apparatus as claimed in claim **2**, wherein the radio resource management is that of the mobile communication network.

4. The apparatus as claimed in claim **1**, wherein the second linking module comprises a low power RF radio resource.

5. The apparatus as claimed in claim **4**, wherein the low power RF is Bluetooth.

6. The apparatus as claimed in claim **1**, wherein the call control and mobility management is that of the mobile communication network.

7. The apparatus as claimed in claim **1** , wherein the mobile communication network is GSM.

8. The apparatus as claimed in claim **1**, further comprising a radio resource manager for a user terminal, and a third linking module adapted to link to the interface of the terminal device so as to transfer radio resource information between the mobile station and the user terminal.

9. The apparatus as claimed in claim **1**, further comprising a browser.

10. The apparatus as claimed in claim **1**, further comprising:

a controller adapted to control transfer of information such that in a first mode transfer of information is between the mobile station and a mobile communication network, and in a second mode transfer of information is between the mobile station and a further communication network; and

a radio module adapted to provide radio contact between the mobile station and the mobile communication network in both the first and second modes.

11. A method comprising:

providing a first communication link to an interface between a mobile station and a mobile communication network, said first communication link including radio resource management of the mobile communication network;

providing a second communication link to the interface between the mobile station and a second communication network, said second communication link including radio resource management of the second communication network; and

maintaining said first communication link such that the mobile station remains connected to the mobile communication network, while actual data is carried over the second communication network.

12. A method as claimed in claim **11**, further comprising: managing radio resources information independently of the mode of operation of the mobile station.

13. A method as claimed in claim **11**, further comprising:

providing the second communication link as a low power RF radio resource.

14. A method as claimed in claim **13**, wherein the low power RF is Bluetooth.

15. A method as claimed in claim 11, further comprising:
providing call control and mobility management of the mobile station via the mobile communication network.

16. A communication system comprising a mobile communication network and a second communication network and a dual mode mobile station capable of communicating with both of said networks, wherein:
the mobile communication network comprises a base transceiver station emulator for providing a first communication link to the interface between the mobile station and the mobile communication network, said first communication link including radio resource management of the mobile communication network;
the second communication network comprises a second base transceiver station for providing a second communication link to the interface between the mobile station and the second communication network, said second communication link including radio resource management of the second communication network; and
the mobile communication network comprises controlling means for maintaining said first communication link such that the mobile station remains connected to the mobile communication network, while actual data is carried over the second communication network.

17. A communication system as claimed in claim 16, wherein the mobile communication network comprises managing means for managing radio resources information independently of the mode of operation of the mobile station.

18. A communication system as claimed in claim 16, wherein the second communication link is a low power RF radio resource.

19. A communication system as claimed in claim 18, wherein the low power RF is Bluetooth.

20. A communication system as claimed in claim 16, wherein call control and mobility management of the mobile station is managed via the mobile communication network.

21. A computer-readable medium including a computer program product stored thereon, the computer program product for controlling communication for a mobile station, comprising:
computer code for providing a first communication link to an interface between a mobile station and a mobile communication network, said first communication link including radio resource management of the mobile communication network;
computer code for providing a second communication link to the interface between the mobile station and a second communication network, said second communication link including radio resource management of the second communication network; and;
computer code for maintaining said first communication link such that the mobile station remains connected to the mobile communication network, while actual data is carried over the second communication network.

* * * * *