EXHIBIT C

GOOGLE, INC.'S PROPOSED CONSTRUCTIONS OF DISPUTED CLAIM TERMS

No.	Term	Claim(s)	Google, Inc.'s Proposed Construction & Evidentiary Support
1.	non-relational, distributed database system	claims 1, 8, 13	a database, stored across multiple computers on a network, wherein data objects exist independently of their attribute values, and wherein data is not extracted using relational algebra **Intrinsic Support:** '593 patent at Abstract; col. 1:10-37; col. 1:65-2:18; col. 2:66-3:59; col. 4:23-7:38; col. 10:25-51; col. 11:24-44; col. 12:6-33; Fig. 1, Fig. 2, Fig. 8, and Fig. 8a. '593 Prosecution History, June 7, 1996 Amendment at 7-14, 16-19; Dec. 11, 1996 Amendment at 8-9; May 14, 1997 Amendment at 3. Chaturvedi et al., *Scheduling the Allocation of Data Fragments in a Distributed Database Environment: A Machine Learning Approach*, IEEE TRANS ON ENG'G MGMT., vol. 41, no. 2, 1994, pp. 194-206. (GN 524 – 537) Houtsma et al., *Parallel Hierarchical Evaluation for Transitive Closure Queries*, IEEE Apr. 1991. (GN 4932 – 4941) U.S. Patent 4,811,199. (GN 7147 – 7162) **Extrinsic Support:** "Relational database: A database in which the data are organized and accessed according to relations." *Dictionary of Computing*. Research Triangle Park, NC: International Business Machines Corporation, 1991, p. 475. (GN 292968) "Relational database: A database is which data are organized into one or more relations that may be manipulated using a relational algebra." Christopher Booth, ed. *The New IEEE Standard Dictionary of Electrical and Electronics Terms* (5th Ed.: Inst. of Electrical & Electronics Engineers, Inc.), 1991, p. 1106.

No.	Term	Claim(s)	Google, Inc.'s Proposed Construction & Evidentiary Support
			(GN 300219) "Relational database: A database is which data are organized into one or more relations that may be manipulated using a relational algebra." *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries* (Inst. of Electrical & Electronics Engineers, Inc.), 1991, p. 170. (GN 300234) "Relation: (1) In a relational database, a set of entity occurrences that have the same attributes … (3) In a relational database, a table that identifies entities and their attributes. Synonymous with flat file …." *Dictionary of Computing*. Research Triangle Park, NC: International Business Machines Corporation, 1991, p. 475. (GN 292968) "Relation: In a relational data model or relational database, a set of tuples, each of which has the same attributes." Christopher Booth, ed. *The New IEEE Standard Dictionary of Electrical and Electronics Terms* (5th Ed.: Inst. of Electrical & Electronics Engineers, Inc.), 1991, p. 1106. (GN 300219) "Relation: In a relational data model or relational database, a set of tuples, each of which has the same attributes." *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries* (Inst. of Electrical & Electronics Engineers, Inc.), 1991, p. 170. (GN 300234) "Relational database management system: A database organization scheme that treats files as tables of data in which the rows represent fixed-length records and columns represent fields. Multiple keys can be used for retrieving the data stored within the database. A database in which some data items in one type of record refer to records of a different type. Relational databases give the user the flexibility to link (join, or create a relationship between) information stored in many disk files. It allows users to interchange and cross-reference information between two different types of records, such as comparing the

No.	Term	Claim(s)	Google, Inc.'s Proposed Construction & Evidentiary Support
			information in a group of invoices to the information in an inventory. Most people do not need relational databases. To merely keep track of a mailing list doesn't require a relational database, nor is such a database needed to keep a simple inventory of something. However, those who want to print out a mailing list of people who ordered products from their inventory will need a relational database. Relational databases are more powerful, more complex, more difficult to use, and more expensive than other database systems." *Webster's New World Dictionary of Computer Terms* (4TH ed.) (Prentice Hall: New York, NY), 1992, p. 353. "Relational database management system (RDBMS): A database management system based on the relational model. This claim is often made (particularly for personal computer packages) principally on the grounds that the data is treated as a series of two-dimensional tables, known as relations. Stricter criteria would require also that algebraic operations, such as JOIN or PROJECT, could be used to manipulate the data and to create new tables based on various combinations of the original tables." Gunton, Tony. *A Dictionary of Information Technology and Computer Science*, 257, 2nd ed. Oxford, UK: NCC Blackwell Ltd. 1993, p. 257. (GN 292982) Kenneth Baclawski and J. Elliott Smith, *A Unified Approach to High-Performance, Vector-Based Information Retrieval*, March 21, 1994. (JAR 294–313)
2.	a plurality of home nodes and a plurality of query nodes connected by a network a plurality of home nodes; and a plurality of query nodes; said plurality of	claim 1 claims 8, 13	a plurality of home nodes and query nodes connected by a network arranged with no central server and wherein, for any given query, any node may be defined as a home node or a query node **Intrinsic Support:** '593 patent at Abstract; col. 1:65-2:18; col. 2:66-3:59; col. 4:8-7:38; col. 10:25-51; col. 11:24-44; col. 12:6-33; Fig. 1, Fig. 2, Fig. 8, Fig. 8a.

No.	Term	Claim(s)	Google, Inc.'s Proposed Construction & Evidentiary Support
	home nodes and said plurality of query nodes connected by a network		'593 Prosecution History, April 16, 1996 Office Action at 2-3; June 7, 1996 Amendment at 13, 16; May 14, 1997 Amendment at 2-3. Chaturvedi et al., *Scheduling the Allocation of Data Fragments in a Distributed Database Environment: A Machine Learning Approach*, IEEE TRANS ON ENG'G MGMT., vol. 41, no. 2, 1994, pp. 194-206. (GN 524 – 537) Houtsma et al., *Parallel Hierarchical Evaluation for Transitive Closure Queries*, IEEE Apr. 1991. (GN 4932 – 4941) U.S. Patent 4,811,199. (GN 7147 – 7162) **Extrinsic Support:** Kenneth Baclawski and J. Elliott Smith, *A Unified Approach to High-Performance, Vector-Based Information Retrieval*, March 21, 1994. (JAR 294–313)
3.	randomly selecting	claim 1	selecting by chance, independently of preceding selections, where each item in the set has equal probability of being chosen **Intrinsic Support:** '593 patent at col. 3:17-25; col. 10:25-51; col. 11:24-44; col. 12:6-33; Fig. 1, Fig. 2. '593 Prosecution History, June 7, 1996 Amendment at p. 13, 16; September 11, 1996 Office Action at 2; March 13, 1997 Office Action at 2; May 14, 1997 Response to Office Action at 2. Chaturvedi et al., *Scheduling the Allocation of Data Fragments in a Distributed Database Environment: A Machine Learning Approach*, IEEE TRANS ON ENG'G MGMT., vol. 41, no. 2, 1994, pp. 194-206. (GN 524 – 537)

No.	Term	Claim(s)	Google, Inc.'s Proposed Construction & Evidentiary Support
			Houtsma et al., *Parallel Hierarchical Evaluation for Transitive Closure Queries*, IEEE Apr. 1991. (GN 4932 – 4941) **<u>Extrinsic Support</u>:** "Random: 1. occurring or done without definite aim, reason, or pattern: random examples. 2. Statistics. of or characterizing a process of selection in which each item of a set has an equal probability of being chosen. . . . 5. at random, without regard to rules, schedules, etc.; haphazardly." *Random House Webster's College Dictionary*, New York, NY: Random House Inc. 1991, p. 1116. (GN 300206) "Random: . . . b. Statistics. Governed by or involving equal chances for each of the actual or hypothetical members of a population; also, produced or obtained by a random process (and therefore completely unpredictable in detail) . . . . the movement of something in successive steps … each step being governed by chance independently of preceding steps." J.A. Simpson & E.S.C. Wiener, eds. *Oxford English Dictionary*, 2d ed., vol. 13, 1989 (Clarendon Press: Oxford, UK), p. 168. (GN 300405) "Random: (3) (modeling and simulation). Pertaining to a process or variable whose outcome or value depends on chance or on a process that simulates chance, often with the implication that all possible outcomes or values have an equal probability of occurrence; for example, the outcome of flipping a coin or executing a computer-programmed random number generator." Christopher Booth, ed. *The New IEEE Standard Dictionary of Electrical and Electronics Terms* (5th Ed.: Inst. of Electrical & Electronics Engineers, Inc.), 1991, p. 1064. (GN 300218) "Random: Pertaining to a process or variable whose outcome or value depends on chance or on a process that simulates chance, often with the implication that all possible outcomes or values have an equal probability of occurrence; for

No.	Term	Claim(s)	Google, Inc.'s Proposed Construction & Evidentiary Support
			example, the outcome of flipping a coin or executing a computer-programmed random number generator." *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries* (Inst. of Electrical & Electronics Engineers, Inc.), 1991, p. 167. (GN 300233) Kenneth Baclawski and J. Elliott Smith, *A Unified Approach to High-Performance, Vector-Based Information Retrieval*, March 21, 1994. (JAR 294–313)
4.	query fragment	claims 1, 8, 13	a part of a query consisting of a limited number of attributes and attribute values joined by relationships, specified in the same formal, artificial language and ontology which describes the attribute values of objects of the database **Intrinsic Support:** '593 patent at Abstract; col. 1:10-31; col. 2:3-18; col. 3:25-4:7; col. 4:22-36; col. 4:60-5:29; col. 10:25-51; col. 11:24-44; col. 12:6-33; Fig. 2, Fig. 8, Fig. 8a. '593 Prosecution History, June 7, 1996 Amendment at 9-16; Dec. 11, 1996 Amendment at 8-9; May 14, 1997 Response to Office Action at 2-3. Chaturvedi et al., *Scheduling the Allocation of Data Fragments in a Distributed Database Environment: A Machine Learning Approach*, IEEE TRANS ON ENG'G MGMT., vol. 41, no. 2, 1994, pp. 194-206. (GN 524 – 537) Houtsma et al., *Parallel Hierarchical Evaluation for Transitive Closure Queries*, IEEE Apr. 1991. (GN 4932 – 4941) U.S. Patent 4,811,199. (GN 7147 – 7162) **Extrinsic Support:** "The '593 query fragments are defined at '593 column 1, lines 27-31, and consist of a part of the query with a limited number of attribute values joined by

No.	Term	Claim(s)	Google, Inc.'s Proposed Construction & Evidentiary Support
			relationships." U.S. Patent No. 6,505,191 Prosecution History, July 3, 2002 Response to office action at 5. (GN 299941 – 300172) Kenneth Baclawski and J. Elliott Smith, *A Unified Approach to High-Performance, Vector-Based Information Retrieval*, March 21, 1994. (JAR 294–313) Kenneth Baclawski and J. Elliott Smith, *KEYNET: Fast Indexing for Semantically Rich Information Retrieval*, December 7, 1993 (JAR 828–845) "Prototype Specifications . . . The hash algorithm is taken from Knuth, volume 3, section 6.4." (JAR 219469)
5.	hashing / hashes	claims 1, 8, 13	performing a mathematical function on a key value to generate the address of the location of data associated with the key value **Intrinsic Support:** '593 patent at Abstract; col. 1:33-62; col. 2:3-12; col. 3:25-4:7; col. 4:60-5:29; col. 6:39-7:38; col. 8:61-9:2; col. 10:25-51; col. 11:24-44; col. 12:6-33; Fig. 8, Fig. 8a. '593 Prosecution History, June 7, 1996 Amendment at 12, 14-16; May 14, 1997 Amendment at 3-4. Chaturvedi et al., *Scheduling the Allocation of Data Fragments in a Distributed Database Environment: A Machine Learning Approach*, IEEE TRANS ON ENG'G MGMT., vol. 41, no. 2, 1994, pp. 194-206. (GN 524 – 537) Houtsma et al., *Parallel Hierarchical Evaluation for Transitive Closure Queries*, IEEE Apr. 1991. (GN 4932 – 4941) U.S. Patent 4,811,199. (GN 7147 – 7162)

No.	Term	Claim(s)	Google, Inc.'s Proposed Construction & Evidentiary Support
			Extrinsic Support: "hashing: (1) a key-to-address transformation in which the keys determine the location of the data. (2) The process of applying a formula to a record key to yield a number that represents a disk address." *Webster's New World Dictionary of Computer Terms* (4TH ed.) (Prentice Hall: New York, NY), 1992, p. 187. (GN 300244) "hashing: A technique for arranging a set of items, in which a hash function is applied to the key of each item to determine its hash value. The hash value identifies each item's primary position in a hash table, and if this position is already occupied, the item is inserted wither in an overflow table or in another available positioning the table." Christopher Booth, ed. *The New IEEE Standard Dictionary of Electrical and Electronics Terms* (5th Ed.: Inst. of Electrical & Electronics Engineers, Inc.), 1991, p. 586. (GN 300215) "hashing: A technique for arranging a set of items, in which a hash function is applied to the key of each item to determine its hash value. The hash value identifies each item's primary position in a hash table, and if this position is already occupied, the item is inserted wither in an overflow table or in another available positioning the table." *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries* (Inst. of Electrical & Electronics Engineers, Inc.), 1991, p. 99. (GN 300228) "hashing algorithm: An algorithm used to derive an address within a specified range from a key value. A hashing algorithm is used with a random file to determine the address of the block in which a given record should be stored." Gunton, Tony, *A Dictionary of Information Technology and Computer Science* (2nd ed.) (Oxford, UK: NCC Blackwell Ltd.), 1993, p. 136. (GN 292980) "Section 6.3 treats digital searching, and Section 6.4 discusses an important

No.	Term	Claim(s)	Google, Inc.'s Proposed Construction & Evidentiary Support
			class of methods called hashing techniques, based on arithmetic transformations of the actual keys." Donald Knuth, The *Art of Computer Programming, Volume 3, Sorting and Searching*, Addison-Wesley, 1973, p. 390. "6.4: Hashing. So far we have considered search methods based on comparing the given argument K to the keys in the table, or using its digits to govern a branching process. A third possibility is to avoid all this rummaging around by doing some arithmetical calculation on K, computing a function f(K) which is the location of K and the associated data on the table. . . . These considerations lead to a popular class of search methods commonly known as hashing or scatter storage techniques. The verb "to hash" means to chop something up or to make a mess out of it; the idea in hashing is to chop off some aspects of the key and to use this partial information as the basis for searching. We compute a hash function h(K) and use this value as the address where the search begins." Donald Knuth, The *Art of Computer Programming, Volume 3, Sorting and Searching*, Addison-Wesley, 1973, p. 506-07. "Prototype Specifications . . . The hash algorithm is taken from Knuth, volume 3, section 6.4." (JAR 219469) "hash: An associative access technique that maps the key via a hash function uniformly among a partitioned set of hash buckets. A key search consists of hashing the key to a bucket address and then examining the small hash bucket for records with the desired key." Jim Gray, Andreas Reuter, *Transaction Processing: Concepts and Techniques*, Morgan Kaufman, 1993, p. 120; *see also generall*y pp. 831-851. "Associative access on a single relation can be supported by two types of access paths that use fundamentally different approaches to solving the problem of translating attribute values into tuple addresses. These approaches, which yield fundamentally different functionality, are generally referred to as hashing

No.	Term	Claim(s)	Google, Inc.'s Proposed Construction & Evidentiary Support
			(key transformation) and key comparison." Jim Gray, Andreas Reuter, *Transaction Processing: Concepts and Techniques*, Morgan Kaufman, 1993, p. 833. "Hashing is based on the idea of using the primary key value as a parameter to a function which returns the storage location of the tuple as a result. This is the very principle of hashing, and it has a number of interesting properties. If the transformation function can be kept simple—that is, if it does not need large data structures with search paths and routing data—then the access cost to retrieve a tuple via its key is minimal: take the key value, do some arithmetic, and find the tuple at the address delivered by the function. Ideal hash algorithms allow for one-access retrieval, which is to say that only the page holding the tuple needs to be read when accessing via the primary key. The functional principle of simple hashing is to relate to key value of a tuple and the page number in which it is stored, through a predefined function; because of this, simple hashing is not just an access method, but also a file organization technique. (This was explained in Chapter 14.) As such, it needs page address spaces with special properties, as with become obvious during the detailed description. Hash-based access paths support only queries of the type (key-attribute = const). They do not efficiently support range predicates such as key-attribute between A and B. The attribute used for hashing typically is a primary key of the relation, but can also be used for nonunique attributes." Jim Gray, Andreas Reuter, *Transaction Processing: Concepts and Techniques*, Morgan Kaufman, 1993, p. 833. "Key comparison comprises all methods for maintaining a dynamic search structure on the set of values in the key attribute. These values (or compressed versions of them) can be organized into tables, lists, trees, and so on, depending on the amount of data, the type of search operations to be supported, and the storage size that is available for maintaining the search structure. If such a technique is used for a primary access path, the entire tuple can be stored in

No.	Term	Claim(s)	Google, Inc.'s Proposed Construction & Evidentiary Support
			search structure (see Section 14.4.5, on key-sequenced files, in Chapter 14); otherwise, it will contain pointers to the tuples (e.g., TIDs). Sorting a file along some attribute and keeping it sorted under updates is a very simple example of a search structure based on key comparison—in this case, it is a sequential sorted list of tuples. If records are clustered within blocks according to the key attribute values, searching and scanning in sorted order can be performed efficiently." Jim Gray, Andreas Reuter, *Transaction Processing: Concepts and Techniques*, Morgan Kaufman, 1993, p. 833. "Algorithms to implement associative access come in two flavors. The first one, hashing, supports access based on value equality only; it uses a transformation function that turns the attribute value into a page address where the tuple can be found." Jim Gray, Andreas Reuter, *Transaction Processing: Concepts and Techniques*, Morgan Kaufman, 1993, p. 835. "As in hash-based memory organizations, there has to be a function to transform the attribute value into a file address, where the tuple will (most likely) be found." Jim Gray, Andreas Reuter, *Transaction Processing: Concepts and Techniques*, Morgan Kaufman, 1993, p. 835. "[A] good hash function H has to map primary key values, which are very unevenly distributed over a large value range, into a tuple address space that is much smaller (proportional to the number of existing tuples), such that the resulting addresses are evenly distributed over the address range." Jim Gray, Andreas Reuter, *Transaction Processing: Concepts and Techniques*, Morgan Kaufman, 1993, p. 839. "[i]f the hash function h(p) is properly chosen, then the hash values will be approximately uniformly distributed over their range." (JAR 108483) Kenneth Baclawski and J. Elliott Smith, *A Unified Approach to High-*

No.	Term	Claim(s)	Google, Inc.'s Proposed Construction & Evidentiary Support
			Performance, Vector-Based Information Retrieval, March 21, 1994 (JAR 294–313) Kenneth Baclawski and J. Elliott Smith, *KEYNET: Fast Indexing for Semantically Rich Information Retrieval*, December 7, 1993 (JAR 828–845) JAR0146229-146266 "The examination node 102 then encodes each feature fragment of the object by using a predefined hashing function. Data in the system was previously stored locally on the various index nodes using this hashing function to generate an index to the data in the local database. Thus, the use of the same hashing function to generate an index for data storage and to generate hashed feature fragments for an information object assures that (1) data is distributed uniformly over the index nodes of the routing search engine during the storing of data and (2) the feature fragments are scattered uniformly over the index nodes during the processing of an object." U.S. Patent No. 6,192,364, col. 6:46-58. (GN 299812 – 831) "The home node 107 then encodes each feature of the query by using a predefined hashing function. Data in the system was previously stored locally on the various query nodes 109 using this hashing function to generate an index to the data in the local database. Thus, the use of the same hashing function to generate an index for data storage and to generate hashed probes for a data query assures that (1) data is distributed uniformly over the query nodes 109 of the search engine during the storing of data and (2) the probes are scattered uniformly over the query nodes 109 during the processing of a query." U.S. Patent No. 6,424,973, col. 7:58-8:2. (GN299832 – 854) "The home node encodes each fragment of the query by using a predefined hashing function. The same hashing function preferably is also used in

No.	Term	Claim(s)	Google, Inc.'s Proposed Construction & Evidentiary Support
			generating indexes to storage locations for storing data locally in local databases on the various query nodes. The use of the same hashing function to generate an index for data storage and to generate hashed probes for a query assures that data is distributed uniformly over the query nodes of the search engine during the storing of data, and that the probes are scattered uniformly over the query nodes during the processing of a query." U.S. Patent No. 6,463,433, col. 11:18-29. (GN 299855 – 877) "Thus, the use of the same hashing function to generate an index for data storage and to generate hashed probes for an object assures that data is distributed uniformly over the index nodes 106 of the data warehouse during the storing of data." U.S. Patent No. 6,470,333 Col. 7:36-41. "The home node 105 encodes each fragment of the query by using a predefined hashing function. Data in the distributed computer database system was previously stored locally on the various index nodes 112 using this hashing function to generate an index to the data in the local database. In particular, if a fragment includes a link then it is hashed and stored as a link fragment, while if a fragment does not include a link then it is hashed and stored as an index fragment. Thus, the use of the same hashing function to generate an index for data storage and to generate hashed probes for a query assures that 1. data is distributed uniformly over the index nodes of the search engine during the storing of data and 2. the probes are scattered uniformly over the index nodes during the processing of a query." U.S. Patent No. 6,505,191, col. 8:61-9:8. (GN 299917 – 940) "The examination node 102 then encodes each feature fragment of the object by using a predefined hashing function. Data in the system was previously stored locally on the various index nodes using this hashing function to generate an index to the data in the local database. Thus, the use of the same hashing function to generate an index for data storage and to generate hashed feature

No.	Term	Claim(s)	Google, Inc.'s Proposed Construction & Evidentiary Support
			fragments for an information object assures that (1) data is distributed uniformly over the index nodes of the routing search engine during the storing of data and (2) the feature fragments are scattered uniformly over the index nodes during the processing of an object." U.S. Patent No. 6,535,881, col. 6:59-7:3. (GN 299898 – 916) U.S. Patent No. 6,505,191 Prosecution History, July 3, 2002 Response to Office Action at 5. (GN 299941 – 300172) *See generally* Gerald Salton, *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*, Addison-Wesley, pp. 159-226 (1989). (GN 005396 – 5463) See generally Gerald Salton, Michael McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, 1983, pp. 329-48. See generally William B. Frakes, Ricardo Baeza-Yates, ed., *Information Retrieval / Data Structures & Algorithms*, Prentice Hall (1992) (GN 4430 – 60)
6.	a first portion and a second portion	claims 1, 8, 13	a first part separate and distinct from a second part **Intrinsic Support:** '593 patent at col. 2:3-2:12; col. 3:37-50; col. 4:60-5:29; col. 6:39-7:55; col. 10:25-51; col. 10:63-64; col. 11:24-44; col. 11:52-54, col. 12:6-33. **Extrinsic Support:** "portion: a part of a whole, either separated from or integrated with it; segment." *Random House Webster's College Dictionary*, Random House, New York, 1991, p. 1052. (GN 300204) "The logical, plain meaning of 'first and second part' is that the item described must have two components: a first and a second. . . . The figure drawings . . .

No.	Term	Claim(s)	Google, Inc.'s Proposed Construction & Evidentiary Support
			affirm this common sense and undisputed interpretation." *Anchor Wall Systems, Inc. v. Concrete Products of New London, Inc.*, 2003 WL 1589532 at *3, No. Civ. 01-465 ADM/AJB (D. Minnesota, March 26, 2003). (GN 299783 – 789) "Menu options from a region are placed into specific location in *another* region. . . . There is no clearer way to interpret 'a first region' and 'a second region' than the language of the claim itself. The clear and unambiguous meaning is they are two different regions. No further construction is necessary." *Merit Indust. v. JVL Corp.*, 2007 WL 2463377 at *6-8, Civ. No. 03-1618 (E.D.Pa. Aug. 27, 2007). (GN 299790 – 811) Kenneth Baclawski and J. Elliott Smith, *A Unified Approach to High-Performance, Vector-Based Information Retrieval*, March 21, 1994 (JAR 294 – 313)
7.	transmitting, by said selected home node, each said hashed query fragment of said plurality of query fragments to a respective one of said plurality of query nodes indicated by said first portion of each said hashed query fragment transmits each said hashed query fragment to a respective one of said plurality of query nodes indicated by said first potion of said hashed query	claim 1 claim 8	the selected home node sends each hashed query fragment to exactly one node on the network, that node being identified by said first portion of the hashed query fragment **Intrinsic Support:** '593 patent at Abstract; col. 2:3-2:12; col. 2:66-3:56; col. 4:22-36; col. 4:56-5:29; col. 7:24-65; col. 10:25-51; col. 11:24-44; col. 12:6-33; Fig. 1, Fig. 2, Fig. 8, Fig. 8a. '593 Prosecution History, June 7, 1996 Amendment at 12; March 13, 1997 Office Action at 2; May 14, 1997 Amendment at 2-3. Chaturvedi et al., *Scheduling the Allocation of Data Fragments in a Distributed Database Environment: A Machine Learning Approach*, IEEE TRANS ON ENG'G MGMT., vol. 41, no. 2, 1994, pp. 194-206. (GN 524 – 537)

No.	Term	Claim(s)	Google, Inc.'s Proposed Construction & Evidentiary Support
	fragment transmitting a query message containing every said hashed query fragment to a respective one of said plurality of query nodes indicated by said first portion of said hashed query fragment	claim 13	U.S. Patent 4,811,199. (GN 7147 – 7162) **Extrinsic Support:** Kenneth Baclawski and J. Elliott Smith, *A Unified Approach to High-Performance, Vector-Based Information Retrieval*, March 21, 1994. (JAR 294 –313)
8.	using, by said query node, said second portion of said respective hashed query fragment to access data according to a local hash table located on said query node each said query node uses said second portion of said hashed query fragment to access data according to a local hash table located on said query node said query node, upon receipt of said query message, using said second portion of said hashed query fragment to access data according to a local	claim 1 claim 8 claim 13	each query node receiving a hashed query fragment uses the second portion of the hashed query fragment as a key value to identify the address of data according to a local hash table stored on that query node **Intrinsic Support:** '593 patent at Abstract; col. 2:13-18; col. 2:66-3:16; col. 3:17-4:7; col. 4:23-8:7; col. 8:45-9:60; col. 10:25-51; col. 11:24-44; col. 12:6-33; Fig. 1, Fig. 2. Figs. 3, 3a, 3b, 4, 5, 6, 7, 7a, 8, and 8a. '593 Prosecution History, May 14, 1997 Amendment at 2-3. U.S. Patent 4,811,199. (GN 7147 – 7162) **Extrinsic Support:** Kenneth Baclawski and J. Elliott Smith, *A Unified Approach to High-Performance, Vector-Based Information Retrieval*, March 21, 1994. (JAR 294 –313)

No.	Term	Claim(s)	Google, Inc.'s Proposed Construction & Evidentiary Support
	hash table located on said query node		
9.	local hash table	claims 1, 8, 13	a table resident on and unique to a particular query node in which the unique location of the information in the table is determined by hashing a key value **Intrinsic Support:** '593 patent at col. 1:33-62; col. 3:26-4:7; col. 5:30-39; col. 6:1-27; col. 7:50-65; col. 10:25-51; col. 11:24-44; col. 12:6-33. '593 Prosecution History, May 14, 1997 Amendment at 5. Chaturvedi et al., *Scheduling the Allocation of Data Fragments in a Distributed Database Environment: A Machine Learning Approach*, IEEE TRANS ON ENG'G MGMT., vol. 41, no. 2, 1994, pp. 194-206. (GN 524 – 537) Houtsma et al., *Parallel Hierarchical Evaluation for Transitive Closure Queries*, IEEE Apr. 1991. (GN 4932 – 4941) U.S. Patent No. 4,811,199 (GN 7147 – 7162) **Extrinsic Support:** "hashing: 1) a key-to-address transformation in which the keys determine the location of the data. (2) The process of applying a formula to a record key to yield a number that represents a disk address." *Webster's New World Dictionary of Computer Terms*, 4th ed., Prentice Hall, 1992, p. 187. (GN 300244) "hashing: A technique for arranging a set of items, in which a hash function is applied to the key of each item to determine its hash value. The hash value identifies each item's primary position in a hash table, and if this position is already occupied, the item is inserted wither in an overflow table or in another

No.	Term	Claim(s)	Google, Inc.'s Proposed Construction & Evidentiary Support
			available positioning the table." Christopher Booth, ed. *The New IEEE Standard Dictionary of Electrical and Electronics Terms* (5th Ed.: Inst. of Electrical & Electronics Engineers, Inc.), 1991, p. 586. (GN 300215) "hashing: A technique for arranging a set of items, in which a hash function is applied to the key of each item to determine its hash value. The hash value identifies each item's primary position in a hash table, and if this position is already occupied, the item is inserted wither in an overflow table or in another available positioning the table." *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries* (Inst. of Electrical & Electronics Engineers, Inc.), 1991, p. 99. (GN 300228) "hashing algorithm: An algorithm used to derive an address within a specified range from a key value. A hashing algorithm is used with a random file to determine the address of the block in which a given record should be stored." Gunton, Tony, *A Dictionary of Information Technology and Computer Science* (2nd ed.) (Oxford, UK: NCC Blackwell Ltd.), 1993, p. 136. (GN 292980) "Section 6.3 treats digital searching, and Section 6.4 discusses an important class of methods called hashing techniques, based on arithmetic transformations of the actual keys." Donald Knuth, The *Art of Computer Programming, Volume 3, Sorting and Searching*, Addison-Wesley, 1973, p. 390. "6.4: Hashing. So far we have considered search methods based on comparing the given argument K to the keys in the table, or using its digits to govern a branching process. A third possibility is to avoid all this rummaging around by doing some arithmetical calculation on K, computing a function f(K) which is the location of K and the associated data on the table. . . . These considerations lead to a popular class of search methods commonly known as hashing or scatter storage techniques. The verb "to hash" means to chop something up or to make a mess out of it; the idea in hashing is to chop off some aspects of the

No.	Term	Claim(s)	Google, Inc.'s Proposed Construction & Evidentiary Support
			key and to use this partial information as the basis for searching. We compute a hash function h(K) and use this value as the address where the search begins." Donald Knuth, The *Art of Computer Programming, Volume 3, Sorting and Searching*, Addison-Wesley, 1973, p. 506-07. "Prototype Specifications . . . The hash algorithm is taken from Knuth, volume 3, section 6.4." (JAR 219469) "hash: An associative access technique that maps the key via a hash function uniformly among a partitioned set of hash buckets. A key search consists of hashing the key to a bucket address and then examining the small hash bucket for records with the desired key." Jim Gray, Andreas Reuter, *Transaction Processing: Concepts and Techniques*, Morgan Kaufman, 1993, p. 120; *see also generall*y pp. 831-851. "Associative access on a single relation can be supported by two types of access paths that use fundamentally different approaches to solving the problem of translating attribute values into tuple addresses. These approaches, which yield fundamentally different functionality, are generally referred to as hashing (key transformation) and key comparison." Jim Gray, Andreas Reuter, *Transaction Processing: Concepts and Techniques*, Morgan Kaufman, 1993, p. 833. "Hashing is based on the idea of using the primary key value as a parameter to a function which returns the storage location of the tuple as a result. This is the very principle of hashing, and it has a number of interesting properties. If the transformation function can be kept simple—that is, if it does not need large data structures with search paths and routing data—then the access cost to retrieve a tuple via its key is minimal: take the key value, do some arithmetic, and find the tuple at the address delivered by the function. Ideal hash algorithms allow for one-access retrieval, which is to say that only the page

No.	Term	Claim(s)	Google, Inc.'s Proposed Construction & Evidentiary Support
			holding the tuple needs to be read when accessing via the primary key. The functional principle of simple hashing is to relate to key value of a tuple and the page number in which it is stored, through a predefined function; because of this, simple hashing is not just an access method, but also a file organization technique. (This was explained in Chapter 14.) As such, it needs page address spaces with special properties, as with become obvious during the detailed description. Hash-based access paths support only queries of the type (key-attribute = const). They do not efficiently support range predicates such as key-attribute between A and B. The attribute used for hashing typically is a primary key of the relation, but can also be used for nonunique attributes." Jim Gray, Andreas Reuter, *Transaction Processing: Concepts and Techniques*, Morgan Kaufman, 1993, p. 833. "Key comparison comprises all methods for maintaining a dynamic search structure on the set of values in the key attribute. These values (or compressed versions of them) can be organized into tables, lists, trees, and so on, depending on the amount of data, the type of search operations to be supported, and the storage size that is available for maintaining the search structure. If such a technique is used for a primary access path, the entire tuple can be stored in search structure (see Section 14.4.5, on key-sequenced files, in Chapter 14); otherwise, it will contain pointers to the tuples (e.g., TIDs). Sorting a file along some attribute and keeping it sorted under updates is a very simple example of a search structure based on key comparison—in this case, it is a sequential sorted list of tuples. If records are clustered within blocks according to the key attribute values, searching and scanning in sorted order can be performed efficiently." Jim Gray, Andreas Reuter, *Transaction Processing: Concepts and Techniques*, Morgan Kaufman, 1993, p. 833. "Algorithms to implement associative access come in two flavors. The first one, hashing, supports access based on value equality only; it uses a transformation function that turns the attribute value into a page address where

No.	Term	Claim(s)	Google, Inc.'s Proposed Construction & Evidentiary Support
			the tuple can be found." Jim Gray, Andreas Reuter, *Transaction Processing: Concepts and Techniques*, Morgan Kaufman, 1993, p. 835. "As in hash-based memory organizations, there has to be a function to transform the attribute value into a file address, where the tuple will (most likely) be found." Jim Gray, Andreas Reuter, *Transaction Processing: Concepts and Techniques*, Morgan Kaufman, 1993, p. 835. "[A] good hash function H has to map primary key values, which are very unevenly distributed over a large value range, into a tuple address space that is much smaller (proportional to the number of existing tuples), such that the resulting addresses are evenly distributed over the address range." Jim Gray, Andreas Reuter, *Transaction Processing: Concepts and Techniques*, Morgan Kaufman, 1993, p. 839. "[i]f the hash function h(p) is properly chosen, then the hash values will be approximately uniformly distributed over their range." (JAR 108483) Kenneth Baclawski and J. Elliott Smith, *A Unified Approach to High-Performance, Vector-Based Information Retrieval*, March 21, 1994 (JAR 294–313) Kenneth Baclawski and J. Elliott Smith, *KEYNET: Fast Indexing for Semantically Rich Information Retrieval*, December 7, 1993 (JAR 828–845) JAR0146229-146266 "The examination node 102 then encodes each feature fragment of the object by using a predefined hashing function. Data in the system was previously stored locally on the various index nodes using this hashing function to generate an index to the data in the local database. Thus, the use of the same hashing

No.	Term	Claim(s)	Google, Inc.'s Proposed Construction & Evidentiary Support
			function to generate an index for data storage and to generate hashed feature fragments for an information object assures that (1) data is distributed uniformly over the index nodes of the routing search engine during the storing of data and (2) the feature fragments are scattered uniformly over the index nodes during the processing of an object." U.S. Patent No. 6,192,364, col. 6:46-58. (GN 299812 – 831) "The home node 107 then encodes each feature of the query by using a predefined hashing function. Data in the system was previously stored locally on the various query nodes 109 using this hashing function to generate an index to the data in the local database. Thus, the use of the same hashing function to generate an index for data storage and to generate hashed probes for a data query assures that (1) data is distributed uniformly over the query nodes 109 of the search engine during the storing of data and (2) the probes are scattered uniformly over the query nodes 109 during the processing of a query." U.S. Patent No. 6,424,973, col. 7:58-8:2. (GN299832 – 854) "The home node encodes each fragment of the query by using a predefined hashing function. The same hashing function preferably is also used in generating indexes to storage locations for storing data locally in local databases on the various query nodes. The use of the same hashing function to generate an index for data storage and to generate hashed probes for a query assures that data is distributed uniformly over the query nodes of the search engine during the storing of data, and that the probes are scattered uniformly over the query nodes during the processing of a query." U.S. Patent No. 6,463,433, col. 11:18-29. (GN 299855 – 877) "Thus, the use of the same hashing function to generate an index for data storage and to generate hashed probes for an object assures that data is distributed uniformly over the index nodes 106 of the data warehouse during the storing of data." U.S. Patent No. 6,470,333 Col. 7:36-41.

No.	Term	Claim(s)	Google, Inc.'s Proposed Construction & Evidentiary Support
			"The home node 105 encodes each fragment of the query by using a predefined hashing function. Data in the distributed computer database system was previously stored locally on the various index nodes 112 using this hashing function to generate an index to the data in the local database. In particular, if a fragment includes a link then it is hashed and stored as a link fragment, while if a fragment does not include a link then it is hashed and stored as an index fragment. Thus, the use of the same hashing function to generate an index for data storage and to generate hashed probes for a query assures that 1. data is distributed uniformly over the index nodes of the search engine during the storing of data and 2. the probes are scattered uniformly over the index nodes during the processing of a query." U.S. Patent No. 6,505,191, col. 8:61-9:8. (GN 299917 – 940) "The examination node 102 then encodes each feature fragment of the object by using a predefined hashing function. Data in the system was previously stored locally on the various index nodes using this hashing function to generate an index to the data in the local database. Thus, the use of the same hashing function to generate an index for data storage and to generate hashed feature fragments for an information object assures that (1) data is distributed uniformly over the index nodes of the routing search engine during the storing of data and (2) the feature fragments are scattered uniformly over the index nodes during the processing of an object." U.S. Patent No. 6,535,881, col. 6:59-7:3. (GN 299898 – 916) U.S. Patent No. 6,505,191 Prosecution History, July 3, 2002 Response to Office Action at 5. (GN 299941 – 300172) *See generally* Gerald Salton, *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*, Addison-Wesley, pp. 159-226 (1989). (GN 005396 – 5463)

No.	Term	Claim(s)	Google, Inc.'s Proposed Construction & Evidentiary Support
			See generally Gerald Salton, Michael McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, 1983, pp. 329-48. See generally William B. Frakes, Ricardo Baeza-Yates, ed., *Information Retrieval / Data Structures & Algorithms*, Prentice Hall (1992). (GN 4430 – 60)
10.	returning, by each said query node each said query node … returns said query node … returning	claim 1 claim 8 claim 13	each query node that accesses data returns an object identifier to the home node **Intrinsic Support:** '593 patent at Abstract; col. 1:11-13; col. 3:51-4:7; col. 4:22-36; col. 7:50-65; col. 10:25-51; col. 11:24-44; col. 12:6-33; Fig. 1, Fig. 2. '593 Prosecution History, Dec. 11, 1996 Amendment at p. 8-9; April 16, 1996 Office Action at 3; June 7 1996 Response to Office Action at 11, 14. **Extrinsic Support:** Kenneth Baclawski and J. Elliott Smith, *A Unified Approach to High-Performance, Vector-Based Information Retrieval*, March 21, 1994 (JAR 294–313)
11.	predetermined degree of relevance	claims 3, 9	a predefined degree of similarity; only results meeting or exceeding a predetermined level are returned to the user after the object identifier has been returned **Intrinsic Support:** '593 patent at col. 2:13-2:18; col. 3:51-4:21; col. 8:16-20; col. 8:45-53; col. 10:55-60; col. 11:45-48; Fig. 2, Fig. 8, Fig. 8a. '593 Prosecution History, June 7, 1996 Amendment at p. 16-17. Chaturvedi et al., *Scheduling the Allocation of Data Fragments in a Distributed*

No.	Term	Claim(s)	Google, Inc.'s Proposed Construction & Evidentiary Support
			Database Environment: A Machine Learning Approach, IEEE TRANS ON ENG'G MGMT., vol. 41, no. 2, 1994, pp. 194-206. (GN 524 – 537) **Extrinsic Support:**