METHOD AND SYSTEM FOR QUERYING AN ON DEMAND DATABASE SERVICE

Inventors: Chirag Rajan, San Francisco, CA (US); Jesse Collins, San Francisco, CA (US); Seung-Yeoul Yang, San Francisco, CA (US); Hoon Kim, Redwood City, CA (US)

Assignee: SALESFORCE.COM, INC., San Francisco, CA (US)

Filed: Jun. 15, 2012

Publication Classification

Int. Cl. G06F 17/30 (2006.01)

U.S. Cl. 707/714; 707/718; 707/E17.017

ABSTRACT

Methods and systems are provided for querying a multi-tenant database. One exemplary method of generating an improved query plan to a database from an input query submitted to an on-demand database service over a network involves determining guidance information appropriate to the input query based at least in part on an identity of a source of the input query, determining a database hint for processing the input query based at least in part on the guidance information, and providing the database hint to the database to form the improved query plan, thereby enabling the database to return an improved result responsive to the input query based at least in part upon the database hint.
QUERYING PROCESS

DETERMINE DATABASE
UTILIZATION STATISTICS

RECEIVE INPUT QUERY

IDENTIFY GUIDANCE INFORMATION
ASSOCIATED WITH INPUT QUERY

DETERMINE HINTS BASED ON
INPUT QUERY AND GUIDANCE INFORMATION

PROVIDE HINTS AND INPUT QUERY TO DATABASE
FOR DETERMINING QUERY PLAN BASED ON HINTS

RECEIVE QUERY RESULTS FROM
DATABASE

EXIT

FIG. 3
METHOD AND SYSTEM FOR QUERYING AN ON DEMAND DATABASE SERVICE

CROSS-REFERENCE TO RELATED APPLICATION(S)

This application claims the benefit of U.S. provisional patent application Ser. No. 61/499,304, filed Jun. 21, 2011, the entire content of which is incorporated by reference herein.

TECHNICAL FIELD

Embodiments of the subject matter described herein relate generally to computer systems and networks configured to support applications executing on behalf of users accessing them as services. More particularly, embodiments of the subject matter relate to methods and systems for efficiently querying a database service being provided in an on demand environment.

BACKGROUND

Modern software development is evolving away from the client-server model toward network-based processing systems that provide access to data and services via the Internet or other networks. In contrast to traditional systems that host networked applications on dedicated server hardware, a "cloud" computing model allows applications to be provided over the network supplied by an infrastructure provider. The infrastructure provider typically abstracts the underlying hardware and other resources used to deliver a customer-developed application so that the customer no longer needs to operate and support dedicated server hardware. The cloud computing model can often provide substantial cost savings to the customer over the life of the application because the customer no longer needs to provide dedicated network infrastructure, electrical and temperature controls, physical security and other logistics in support of dedicated server hardware.

Cloud-based architectures have been developed to improve collaboration, integration, and community-based cooperation between customer tenants without sacrificing data security. During operation, there are numerous situations in which data and/or information needs to be retrieved (e.g., for presentation to users) from a database being provided to customers in such a cloud-based environment. Conventional databases include management software that determines what the database manufacturer considers to be an "optimal query plan" for executing the query and retrieving the desired set of data and/or information. However, the manufacturer’s database management software is not always fully adapted for use in the cloud environment, so the "optimal query plan" generated by such conventional systems does not reflect the nature of users querying the database, and therefore, may actually produce "optimal" query plans that are not in fact optimal when used in the cloud environment. What is really needed is a remedy to this and other shortcomings of the traditional database manager.

BRIEF DESCRIPTION OF THE DRAWINGS

A more complete understanding of the subject matter may be derived by referring to the detailed description and claims when considered in conjunction with the following figures, wherein like reference numbers refer to similar elements throughout the figures.

Fig. 1 is a block diagram of an exemplary system in which select embodiments may be realized;

Fig. 2 is a block diagram of an exemplary querying system suitable for use with the system of Fig. 1 in accordance with one or more embodiments; and

Fig. 3 is a flow diagram of an exemplary querying process that may be performed by the system of Fig. 1 and/or the querying system of Fig. 2 in accordance with one or more exemplary embodiments.

DETAILED DESCRIPTION

Select embodiments may employ the techniques described herein implemented as one or a combination of methods, systems or processor executed code to form an improved query plan based at least in part upon a query to a database received from a submitter in an on demand environment. In one example embodiment, guidance information may be used to provide a "hint" to an on-demand database service's database management system in accordance with the improved query plan. Guidance information may be determined at least in part on an identity of source of the input query, for example. As used herein, guidance information may take one or a combination of forms, such as without limitation, a metric, a value, integer, numeral, text, or the like, that is indicative of or otherwise influenced by one or more entries, qualities or characteristics of a particular submitter in the on demand database service. As will be described below with reference to specific embodiments, hints provided to a database management system may indicate to the database management system, and/or query optimizer, how at least a portion of an improved query plan can be can be generated or otherwise constructed. Accordingly, exploiting such guidance information can thereby enable querying an on demand database service more efficiently.

While implementation specific differences exist, some embodiments may employ multi-tenancy techniques when providing the above described benefits, however multi-tenancy is not required by all embodiments. Generally speaking, multi-tenancy refers to a technique where a hardware and software platform simultaneously supports multiple user groups (also referred to as "organizations" or "tenants") from a common computational resource, such as a data storage element. An example type of multi-tenant data storage is a relational database (referred to as a "multi-tenant database"), however embodiments may be realized using object oriented and other types of databases as well. For example, a "tenant" or an "organization" can be used to refer to a group of one or more users that shares access to common subset of the data within the multi-tenant database. In this regard, each tenant includes one or more users associated with, assigned to, or otherwise belonging to that respective tenant. Tenants may represent customers, customer departments, business or legal organizations, and/or any other entities that maintain data for particular sets of users within a common multi-tenant system.

Although multiple tenants may share access to the server 102 and the database 130, the particular data and services provided from the server 102 to each tenant can be securely isolated from those provided to other tenants. The multi-tenant architecture therefore allows different sets of users to share functionality without necessarily sharing any of the data 132 belonging to or otherwise associated with other tenants. Multi-tenant design choices can enable one or more advantages over conventional server virtualization systems. First, the multi-tenant platform operator can often make
improvements to the platform based upon collective information from the entire tenant community. Additionally, because all users in the multi-tenant environment execute applications within a common processing space, it is relatively easy to grant or deny access to specific sets of data for any user within the multi-tenant platform, thereby improving collaboration and integration between applications and the data managed by the various applications. The multi-tenant architecture therefore allows convenient and cost effective sharing of similar application features between multiple sets of users.

[0012] Turning now to FIG. 1, an exemplary system 100 includes a server 102 that dynamically creates and supports virtual applications 128 based upon data 132 from a common database 130 such that it is shared. In some embodiments, database 130 may be shared among different tenants, and in such case, it may be referred to as a multi-tenant database; however, multi-tenancy is not a requirement of database 130. Data and services generated by the virtual applications 128 are provided via a network 145 to any number of client devices 140, as desired. Each virtual application 128 is suitably generated at run-time using a common application platform 110 that securely provides access to the data 132 in the database 130 for each of the various submitters subscribing to the system 100. In accordance with one non-limiting example, the system 100 can implement one or more specific functions, such as for example a customer relationship management (CRM) system, an Enterprise resource planning (ERP) system, a Partner Relationship Management (PRM) system and the like.

[0013] The database 130 is any sort of repository or other data storage system capable of storing and managing the data 132 associated with any number of submissions. The database 130 may be implemented using any type of conventional database server hardware. In various embodiments, the database 130 stores processing hardware 104 with the server 102. In other embodiments, the database 130 is implemented using separate physical and/or virtual database server hardware that communicates with the server 102 to perform the various functions described herein. In an exemplary embodiment, the database 130 includes a database management system or other equivalent software capable of determining a query plan for retrieving and providing a particular subset of the data 132 to an instance of virtual application 128 in response to a query initiated or otherwise provided by a user of a client device 140, as described in greater detail below.

[0014] In practice, the data 132 may be organized and formatted in any manner to support the application platform 110. In various embodiments, the data 132 is suitably organized into a relatively small number of large data tables to maintain a semi-amorphous “heap”-type format. The data 132 can then be organized as needed for a particular virtual application 128. In various embodiments, conventional data relationships are established using any number of pivot tables 134 that establish indexing, uniqueness, relationships between entities, and/or other aspects of conventional database organization as desired. Further data manipulation and report formatting is generally performed at run-time using a variety of metadata constructs. Metadata within a universal data directory (UDD) 136, for example, can be used to describe any number of forms, reports, workflows, user access privileges, business logic and other constructs that are common Tenant-specific formatting, functions and other constructs may be maintained as tenant-specific metadata 138 for each tenant, as desired. Rather than forcing the data 132 into an inflexible global structure that is common to all tenants and applications, the database 130 is organized to be relatively amorphous, with the pivot tables 134 and the metadata 138 providing additional structure on an as-needed basis. To that end, the application platform 110 suitably uses the pivot tables 134 and/or the metadata 138 to generate “virtual” components of the virtual applications 128 to logically obtain, process, and present the relatively amorphous data 132 from the database 130.

[0015] The server 102 is implemented using one or more actual and/or virtual computing systems that collectively provide the dynamic application platform 110 for generating the virtual applications 128. For example, the server 102 may be implemented using a cluster of actual and/or virtual servers operating in conjunction with each other, typically in association with conventional network communications, cluster management, load balancing and other features as appropriate. The server 102 operates with any sort of conventional processing hardware 104, such as a processor 105, memory 106, input/output features 107 and the like. The input/output features 107 generally represent the interface(s) to networks (e.g., to the network 145, or any other local area, wide area or other network), mass storage, display devices, data entry devices and/or the like. The processor 105 may be implemented using any suitable processing system, such as one or more processors, controllers, microprocessors, microcontrollers, processing cores and/or other computing resources spread across any number of distributed or integrated systems, including any number of “cloud-based” or other virtual systems. The memory 106 represents any non-transitory short or long term storage or other computer-readable media capable of storing programming instructions for execution on the processor 105, including any sort of random access memory (RAM), read only memory (ROM), flash memory, magnetic or optical mass storage, and/or the like. The computer-executable programming instructions, when read and executed by the server 102 and/or processor 105, cause the server 102 and/or processor 105 to establish, generate, or otherwise facilitate the application platform 110 and/or virtual applications 128 and perform additional tasks, operations, functions, and processes herein. It should be noted that the memory 106 represents one suitable implementation of such computer-readable media, and alternatively or additionally, the server 102 could receive and cooperate with computer-readable media (not separately shown) that is realized as a portable or mobile component or platform, e.g., a portable hard drive, a USB flash drive, an optical disc, or the like.

[0016] The application platform 110 is any sort of software application or other data processing engine that generates the virtual applications 128 that provide data and/or services to the client devices 140. In a typical embodiment, the application platform 110 gains access to processing resources, communications interfaces and other features of the processing hardware 104 using any sort of conventional or proprietary operating system 108. The virtual applications 128 are typically generated at run-time in response to input received from the client devices 140. For the illustrated embodiment, the application platform 110 includes a bulk data processing engine 112, a query generator 114, a search engine 116 that provides text indexing and other search functionality, and a runtime application generator 120. Each of these features may be implemented as a separate process or other module,
and many equivalent embodiments could include different and/or additional features, components or other modules as desired.

[0017] The runtime application generator 120 dynamically builds and executes the virtual applications 128 in response to specific requests received from the client devices 140. The virtual applications 128 are typically constructed in accordance with the tenant-specific metadata 138, which describes the particular tables, reports, interfaces and/or other features of the particular application 128. In various embodiments, each virtual application 128 generates dynamic web content that can be served to a browser or other client program 142 associated with its client device 140, as appropriate.

[0018] The runtime application generator 120 suitably interacts with the query generator 114 to efficiently obtain data 132 from the database 130 as needed in response to input queries initiated or otherwise provided by users of the client devices 140. In a typical embodiment, the query generator 114 considers the identity of the user requesting a particular function (as well as possibly the user’s associated tenant in some implementations), and then builds and executes queries to the database 130 using system-wide metadata 136, user specific metadata 138, pivot tables 134, and/or any other available resources. The query generator 114 in this example therefore maintains security of the common database 130 by ensuring that queries are consistent with access privileges granted to the user that initiated the request.

[0019] As will be described in greater detail below with reference to specific embodiments of FIGS. 2-3, in an exemplary embodiment employing multi-tenancy, the query generator 114 maintains guidance information comprising database utilization statistics for each of the various tenants supported by system 100 (e.g., tenant-specific database utilization statistics), and in response to a query initiated by a user associated with a particular tenant, the query generator 114 utilizes the database utilization statistics associated with that particular tenant to determine one or more database hints that are utilized by the database 130 to determine an improved query plan for executing the query.

[0020] Still referring to FIG. 1, the data processing engine 112 performs bulk processing operations on the data 132 such as uploads or downloads, updates, online transaction processing, and/or the like. In many embodiments, less urgent bulk processing of the data 132 can be scheduled to occur as processing resources become available, thereby giving priority to more urgent data processing by the query generator 114, the search engine 116, the virtual applications 128, etc.

[0021] In operation, developers may use the application platform 110 to create data-driven virtual applications 128 for the customers that they support. Such virtual applications 128 may make use of interface features such as tenant-specific, for example, screens 124, universal screens 122 or the like. Any number of tenant-specific and/or universal objects 126 may also be available for integration into virtual applications 128. The data 132 associated with each virtual application 128 is provided to the database 130, as appropriate, and stored until it is requested or is otherwise needed, along with the metadata 138 that describes the particular features (e.g., reports, tables, functions, etc.) of that particular virtual application 128. For example, a virtual application 128 may include a number of objects 126 accessible to users of a particular tenant, for example, wherein for each object 126, information pertaining to its object type along with values for various fields associated with that respective object type are maintained as metadata 138 in the database 130. In this regard, the object type can define the structure (e.g., the formatting, functions and other constructs) of each respective object 126 and the various fields associated therewith.

[0022] Still referring to FIG. 1, the data and services provided by the server 102 can be retrieved using any sort of personal computer, mobile telephone, tablet or other network-enabled client device 140 on the network 145. In an exemplary embodiment, the client device 140 includes a display device, such as a monitor, screen, or another conventional electronic display capable of graphically presenting data and/or information retrieved from the database 130, as will be described in greater detail below. Typically, the user operates a conventional browser or other client program 142 executed by the client device 140 to contact the server 102 via the network 145 using a networking protocol, such as the hypertext transport protocol (HTTP) or the like. The user typically authenticates his or her identity to the server 102 to obtain a session identifier (“SessionID”) that identifies the user in subsequent communications with the server 102. When the identified user requests access to a virtual application 128, the runtime application generator 120 suitably creates the application at run time based upon the metadata 138, as appropriate. As noted above, the virtual application 128 may contain Java, ActiveX, or other content that can be presented using conventional client software running on the client device 140; other embodiments may simply provide dynamic web or other content that can be presented and viewed by the user, as desired. As described in greater detail below, the query generator 114 suitably obtains the requested subsets of data 132 from the database 130 as needed to populate the tables, reports or other features of the particular virtual application 128.

[0023] FIG. 2 illustrates an exemplary querying system 200 suitable for use in a computing system, such as the system 100 for example. One embodiment of the illustrated querying system 200 includes a server 202, which can be one example embodiment of server 102 of FIG. 1, and a database 204, which can be one specific embodiment of database 130 of FIG. 1. The server 202 supports a virtual application 206, which can be one example embodiment of virtual application 128 of FIG. 1, and a query generator 208, which can be one embodiment of query generator 114, that are cooperatively configured to retrieve data and/or information from database 204 and present or otherwise provide the retrieved data to a user of the virtual application 206, in one of a variety of techniques, examples of which will be described in greater detail below. While the elements in the querying system 200 of FIG. 2 correspond to elements described above in the context of system 100 of FIG. 1, and may share common features and/or functionality, FIG. 2 does not represent the only embodiment of FIG. 1 contemplated herein.

[0024] As illustrated in FIG. 2, in an exemplary embodiment, the database 204 maintains data, which can be one specific embodiment of data 132 of FIG. 1, in at least one of a plurality of different tables 210. Any one table 210 may comprise a set of data that includes a number of rows corresponding to the number of entries in that respective table 210, and a number of columns across the rows that correspond to the different pieces of data and/or information maintained in that respective table 210. The database 204 includes a database management system 212 that provides an interface between the data maintained by the database 204 (e.g., in tables 210 in one example embodiment) and the server 202.
and/or other external elements. The database management system 212 embodies the software module of the database 204 that may be configured to determine and/or otherwise maintain statistics for the database 204, such as, for example, the number of rows and/or columns of each particular table 210, the number of different distinct values for each column or each particular table 210, and schema information and/or other relational information for the one or more tables 210. The statistics maintained by the database management system 212 need not be tenant-specific and need not reflect multi-tenancy of the various tables 210.

In an exemplary embodiment, the database management system 212 receives queries from the query generator 208. As illustrated, the database management system 212 generates or otherwise supports a query engine 214 that determines an improved query plan for performing the query. The database management system 212 then executes or otherwise performs the query in accordance with the query plan determined by the query engine 214 to retrieve the desired subset of data maintained in the tables 210 and/or database 204, and the database management system 212 provides the retrieved data to the query generator 208 and/or virtual application 206 as the result of the query.

In one example embodiment, employing multi-tenancy referenced in the context of FIG. 3, the query generator 208 can maintain database utilization statistics, for example, by tenant in a multi-tenant implementation supported by the database 204. Guidance information is not, however, limited to database utilization statistics nor limited to multi-tenancy architectures, and may broadly include information about the environment, users or arrangement of data in the database that could reasonably assist in improving the query to the database. In an embodiment, the query generator 208 determines one or more database hints for the query engine 214 based at least in part upon the input query generated by the virtual application 206 and the tenant-specific database statistics for the tenant associated with the particular instance of the virtual application 206. A query statement (e.g., a structured query language (SQL) statement or the like) can be constructed to include or otherwise utilize the database hints. In this regard, a database hint should be understood as an instruction or other guidance provided to the query engine 214 that influences the improved query plan determined by the query engine 214. For example, in one embodiment and without limitation, a database hint may be the identification of a particular table of the plurality of tables 210 to query first, a particular order for joining two or more tables 210 of the database 204 (e.g., a join order), a particular method for joining two or more tables 210 of the database 204 (e.g., a join method or join operation), a particular access path for accessing and/or querying a particular table 210 (e.g., a primary key index, a secondary key index, or the like), or combinations of these and other techniques. In this regard, the tenant-specific database utilization statistics maintained by the query generator 208 for the querying tenant provide a better representation of the querying tenant's data distribution in the database 204 and/or the relationships between the querying tenant's data in the database 204, thereby allowing the query generator 208 to estimate or otherwise determine how to exploit the tenant-specific statistics (which the database management system 212 is not aware of) to achieve the optimal (or lowest cost) query plan, as described in greater detail below.

FIG. 3 depicts an exemplary embodiment of a querying process 300 suitable for implementation by one or more computing devices in a computing system to obtain data and/or information from a database in an on-demand database service. The various tasks performed in connection with the illustrated process 300 may be performed by software, hardware, firmware, or any combination thereof. For illustrative purposes, the following description may refer back to elements mentioned above in connection with FIGS. 1-2, however, it is not intended to limit the embodiments discussed with reference to FIGS. 1-2. In embodiments, portions of the querying process 300 may be performed by different elements of the system 100 and/or the querying system 200, such as, for example, the processor 105, the application platform 110, the data processing engine 112, the query generator 208, a virtual application 206, the database 204, the database management system 212, the query engine 214, and/or a client device 140. It should be appreciated that the querying process 300 may include any number of additional or alternative tasks, the tasks need not be performed in the illustrated order and/or the tasks may be performed concurrently, and/or the secure querying process 300 may be incorporated into embodiments, including procedure or process having additional functionality not limited to that described in detail herein. Moreover, one or more of the tasks shown and described in the context of FIG. 3 could be omitted from a practical embodiment of the querying process 300 without departing from the techniques described herein.

Referring to FIG. 3, in an exemplary embodiment, the querying process 300 begins by determining and maintaining database utilization statistics (task 302). Such database utilization statistics could be kept for each of the tenants supported by a multi-tenant database, for example. Alternatively, database utilization statistics may be kept for all users of a single tenant database running in a cloud environment, one or more instances of a database image executing under virtual machines in a networked cloud environment, or the like. In one example embodiment, the query generator 208 determines at least one utilization statistic for at least one of the tables 210 of the database 204 that includes an entry associated with (or belonging to) a respective tenant, for example. For example, in one embodiment, for each table 210 including entries associated with a first tenant, the query generator 208 may determine the number of rows (or entries) corresponding to the first tenant that exist within that respective table 210. It will be appreciated that such statistics are not limited to any particular utilization metric, and in practice, such statistics may additionally and/or alternatively include the number of different distinct values among the entries associated with that particular data in a particular table 210, such as without limitation, a frequency of search by submitters, a time of last access by a submitter, a speed of retrieval of the underlying database system, any relational information indicative of an association between the entries associated with that particular submitter of the query in one particular table 210 to one or more other tables 210 of the database 204, and/or other such schema information, or the like. In accordance with one embodiment, the query generator 208 determines statistics for each table 210 of the database 204 upon creation of the table 210 and updates the statistics whenever entries are added to and/or removed from a particular table 210. For example, each time a submitter utilizes the virtual application 206 to add and/or delete entries from the database 204, the virtual application 206 may notify the query generator 208 so that the query generator 208 may update the statistics for that tenant to reflect the current state of the database.
The statistics determined by the query generator 208 are stored or otherwise persistently maintained, in memory 106 of FIG. 1 for example, to facilitate determining database hints, as will be described in greater detail below.

Still referring to FIG. 3, the querying process 300 continues by receiving or otherwise obtaining an input query or another request for data and/or information from the database generated by a submitter of a virtual application (task 304). In an exemplary embodiment, the query generator 208 receives, from a virtual application 206, an input query indicative of a request to retrieve a particular subset of data 132, for example from the database 204 in response to an action by a submitter of a client device 140, for example. For example, the virtual application 206 may display one or more graphical user interface (GUI) elements in the web browser 142 that are adapted to allow a non-automatus submitter to view, modify, manipulate, or otherwise access data and/or information in the database 204 that is available to that particular submitter, such as the subset of the data 132 that the submitter is permitted to access based on the submitter’s access privileges and the like. In response to the submitter manipulating the GUI elements presented on the client device 140 to select or otherwise indicate the set of data that the submitter would like the virtual application 206 to present (e.g., in a report, table, chart, graph, or other visual format), the virtual application 206 provides, to the query generator 208, an input query indicative of or otherwise influenced by the GUI elements selected by the submitter to retrieve a desired subset of data 132 needed to present the selected table, chart, graph or other visual report provided by the virtual application 128, for example.

In an exemplary embodiment, in response to receiving the input query, the querying process 300 continues by identifying or otherwise determining the guidance information associated with the input query, such as utilization statistics of a particular submitter, or tenant, for example, and determining one or more database hints based on the input query and the guidance information, for the associated with the query submitter or tenant or the like (tasks 306, 308). In accordance with one embodiment, the virtual application 206 provides the query generator 208 with identifiers that indicate the appropriate guidance information, such as for example and without limitation an identity of a tenant and/or the submitter, associated with the virtual application 206 that generated the input query. The query generator 208 accesses or otherwise utilizes those identifiers to determine the appropriate guidance information (for example, the tenant associated with the submitter of the client device 140), and then obtains (for example, from memory 106) the guidance information corresponding to the query (for example, the database utilization statistics for the various tables of the database 204 that are associated with the query submitter).

After obtaining the statistics for the query, the query generator 208 determines one or more database hints based on the input query and the database utilization statistics for the query. In some embodiments, the database hints may be tenant-specific, in that the same input query from two different tenants may produce two different sets of database hints based on differences between the database utilization statistics for the two different tenants for example. In an exemplary embodiment, the database hints can include access paths (e.g., tables to be queried, indexes to be utilized, and the like), join methods or operations (e.g., the manner in which two or more tables should be combined for purposes of the query), join orders (e.g., the order in which two or more tables should be combined for purposes of the query), and/or the like. For example, based on the input query, the query generator 208 may initially identify or otherwise determine the possible combination of indexes and/or columns of the various tables 210 maintained by the database 204 to be queried. Based on the statistics associated with the query, the query generator 208 may then determine which of the tables 210 and/or indexes to be queried first and/or which of the tables 210 of the database 204 to be joined and/or combined most likely to obtain the results of the input query in accordance with some criterion or criteria, such as for example and without limitation, the lowest cost (e.g., least amount of time and/or computing resources required). To put it another way, based on the statistics for the query and the input query, the query generator 208 determines or otherwise identifies the most selective combinations of tables, indexes, access paths, join operations and/or join orders.

For example, the query generator 208 may determine that the input query requires data and/or information from a first column (Column 1) in a first table (Table 1) of the database 204 that has 100,000 entries (or rows) along with data and/or information from a second column (Column 2) in a second table (Table 2) of the database 204 that has 20,000 entries (or rows). Based on the number of rows in each of the identified tables that are associated with the query, the query generator 208 may determine or otherwise determine one of the columns as the primary index (or key) for executing the query. For example, if there are 5,000 entries applicable to the query in Table 1 and 10,000 entries applicable to the query in Table 2, the query generator 208 may determine that the database management system 212 should begin performing the query on Column 1 of Table 1 rather than Column 2 of Table 2, and thus, determine a database hint for the database management system 212 and/or engine 214 that indicates the query plan should begin with Column 1 of Table 1 (or that Table 1 should proceed Table 2 in a particular join operation). In other situations, based on the statistics, the query generator 208 may determine that Column 2 of Table 2 is more selective (e.g., a greater number of distinct values across the entries applicable to the query) than Column 1 of Table 1 and determine that the database management system 212 should begin performing the query on Column 2 of Table 2 rather than Column 1 of Table 1 (or that Table 2 should proceed Table 1 in a particular join operation).

Still referring to FIG. 3, after determining the database hint(s), the querying process 300 continues by providing the one or more database hints to the database management system 212 along with the input query for subsequent execution (task 310). In an exemplary embodiment, the query generator 208 constructs a query statement (e.g., in SQL or another suitable database querying language) based on the input query that also includes the database hint(s) that indicate to the database management system 212 and/or query engine 214 the access paths, join operations, join orders, tables and/or indexes of the database 204 that should be used by the database management system 212 when generating the query plan. In response to receiving the database hints, the query engine 214 generates or otherwise creates a plurality of possible query plans based on the input query and the database hints and identifies the possible query plan having the best fit to an established criterion (criteria), such as for example and without limitation, lowest estimated cost (e.g., based on required computing resources, amount of computing time
required, and the like) as the improved query plan. In this manner, the improved query plan is influenced by the database hints. In some embodiments, the improved query plan may be understood as being tenant-specific, in that the same input query from two different tenants may produce two different improved query plans based on the differences between the database hints statistics for the two tenants, for example. After the query engine 214 creates the improved query plan based on the input query and the database hints, the database management system 212 executes or otherwise performs the query using that improved query plan to obtain the results of the input query from the database 204.

[0034] It should be noted that by virtue of the techniques described, embodiments can enable providing an improved query plan that can be more efficient or otherwise have a lower cost than the "optimal" query plan that a conventional database management system would have otherwise chosen. For example, absent the techniques described herein, in a join of Table 1 and Table 2, a conventional database management system may otherwise choose a query plan that begins with Table 2 (e.g., by having Table 2 precede Table 1 in a join order) based on the total number of entries in Table 2 (20,000) relative to the total number of entries in Table 1 (100,000), when in fact, there are fewer entries in Table 1 (e.g., 5,000 as compared to 10,000) that are applicable to the query as discussed above. Thus, in contrast to such failings of conventional approaches, the improved query plans achieved by embodiments like those described herein may be more selective, more efficient, or otherwise achieve lower cost by enabling consideration for database utilization statistics which are otherwise unknown by the conventional approaches.

[0035] Referring again to FIG. 3, in an exemplary embodiment, the querying process 300 continues by providing or otherwise presenting the results of the query to the submitter responsible for generating the query (task 312). In this regard, after executing the input query, the database 204 (e.g., database management system 212) provides the query results to the instance of the virtual application 206 that generated the input query, wherein the virtual application 206 may format, organize, or otherwise modify the query results and display or otherwise present a graphical representation of at least a portion of the query results on the client device 140 within the web browser 142. For example, a virtual CRM application 206 may construct a table, chart, graph, report, or another graphical representation of the data and/or information retrieved from the database 204 in accordance with the GUI elements selected by a non-machine submitter of the client device 140.

[0036] The foregoing description is merely illustrative in nature and is not intended to limit the embodiments of the subject matter or the application and uses of such embodiments. Furthermore, there is no intention to be bound by any expressed or implied theory presented in the technical field, background, or the detailed description. As used herein, the word "exemplary" means "serving as an example, instance, or illustration." Any implementation described herein as exemplary is not necessarily to be construed as preferred or advantageous over other implementations, and the exemplary embodiments described herein are not intended to limit the scope or applicability of the subject matter in any way.

[0037] For the sake of brevity, conventional techniques related to computer programming, computer networking, database querying, and other functional aspects of the systems (and the individual operating components of the systems) may not be described in detail herein. In addition, those skilled in the art will appreciate that embodiments may be practiced in conjunction with any number of system and/or network architectures, data transmission protocols, and device configurations, and that the system described herein is merely one suitable example. Furthermore, certain terminology may be used herein for the purpose of reference only, and thus is not intended to be limiting. For example, the terms "first", "second" and other such numerical terms do not imply a sequence or order unless clearly indicated by the context.

[0038] Embodiments of the subject matter may be described herein in terms of functional and/or logical block components and with reference to symbolic representations of operations, processing tasks, and functions that may be performed by various computing components or devices. Such operations, tasks, and functions are sometimes referred to as being computer-executed, computerized, software-implemented, or computer-implemented. In this regard, it should be appreciated that the various block components shown in the figures may be realized by any number of hardware, software, and/or firmware components configured to perform the specified functions. For example, an embodiment of a system or a component may employ various integrated circuit components, e.g., memory elements, digital signal processing elements, logic elements, look-up tables, or the like, which may carry out a variety of functions under the control of one or more microprocessors or other control devices. In this regard, the subject matter described herein can be implemented in the context of any computer-implemented system and/or in connection with two or more separate and distinct computer-implemented systems that cooperate and communicate with one another. In exemplary embodiments, the subject matter described herein can be implemented in conjunction with a virtual application, such as customer relationship management (CRM) in an on demand environment.

[0039] While at least one exemplary embodiment has been presented in the foregoing detailed description, it should be appreciated that a vast number of variations exist. It should also be appreciated that the exemplary embodiment or embodiments described herein are not intended to limit the scope, applicability, or configuration of the claimed subject matter in any way. Rather, the foregoing detailed description will provide those skilled in the art with a convenient road map for implementing the described embodiment or embodiments. It should be understood that various changes can be made in the function and arrangement of elements without departing from the scope defined by the claims, which includes known equivalents and foreseeable equivalents at the time of filing this patent application.

What is claimed is:

1. A method of generating an improved query plan to a database by: receiving an input query submitted to an on demand database service over a network; determining, based at least in part on an identity of a source of the input query, guidance information appropriate to the input query; determining at least one hint to the database to process the input query based at least in part on the guidance information; and providing the at least one hint to the database to form the improved query plan, thereby enabling the database to return an improved result responsive to the input query based at least in part upon the at least one hint.
2. The method of claim 1, further comprising:
receiving query results from the database, the query results comprising the improved result, wherein the database obtains the query results in accordance with the improved query plan; and
graphically presenting at least a portion of the query results on a display device.

3. The method of claim 2, further comprising receiving the input query from a virtual application being accessed by a client device over the network, the client device including the display device, wherein:
the client device is associated with a querying tenant;
determining the guidance information comprises determining the guidance information based at least in part on the identity of the querying tenant; and
graphically presenting the portion of the query results comprises displaying the portion of the query results within the virtual application.

4. The method of claim 3, the virtual application executing on a server coupled to the network, wherein:
the server receives the input query, obtains one or more database utilization statistics for the querying tenant, determines one or more tenant-specific database hints for the querying tenant based on the one or more database utilization statistics, and provides the one or more tenant-specific database hints to the database along with the input query; and
the database generates the improved query plan based on the one or more tenant-specific database hints and the input query and obtains the query results by executing the input query in accordance with the improved query plan.

5. The method of claim 1, wherein determining the guidance information comprises determining, for a first table of the database, a first metric influenced by entries associated with the source of the input query in the first table.

6. The method of claim 5, wherein determining the first metric comprises determining a number of the entries associated with the source in the first table or a number of distinct values across the entries associated with the source in the first table.

7. The method of claim 5, wherein determining the at least one hint comprises determining a first database hint for the first table based at least in part on the first metric.

8. The method of claim 7, wherein determining the first database hint comprises determining a join order for the first table; a join operation for the first table, an access path for the first table, or a key index for the first table.

9. The method of claim 5, wherein:
determining the guidance information further comprises determining, for a second table of the database, a second metric influenced by entries associated with the source of the input query in the second table; and
determining the at least one hint comprises determining a join order for the first table and the second table based at least in part on the first and second metrics.

10. A method of querying a database supporting a plurality of tenants in a multi-tenant system, the method comprising:
maintaining, by a server, database utilization statistics for a first tenant of the plurality of tenants;
receiving, by the server, an input query associated with the first tenant;
determining, by the server, one or more hints for querying the database based at least in part on the database utilization statistics for the first tenant;
providing, by the server, one or more hints and the input query to the database;
determining, by the database, an optimal query plan for the input query based at least in part on the one or more hints;
executing, by the database, the input query using the optimal query plan to obtain query results; and
providing, by the database, the query results to the server.

11. The method of claim 10, the input query being received from a virtual application executing on the server, the virtual application being accessed by a client device coupled to the server over a network, wherein the method further comprises displaying, by the server, at least a portion of the query results within the virtual application.

12. The method of claim 10, the database utilization statistics comprising a first metric for a first table of the database, the first metric being influenced by entries associated with the first tenant in the first table, wherein determining the one or more hints for querying the database comprises determining an access path for the first table based on the first metric.

13. The method of claim 10, the database utilization statistics comprising a first metric for a first table of the database and a second metric for a second table of the database, the first metric being influenced by entries associated with the first tenant in the first table and the second metric being influenced by entries associated with the first tenant in the second table, wherein determining the one or more hints for querying the database comprises determining a join order for the first and second tables based on the first and second metrics.

14. A computing system comprising a processor and a memory, wherein the memory comprises computer-executable instructions that, when executed by the processor, cause the computing system to:
receive an input query from a source supported by a database;
determine a database utilization statistic for the source in the database;
determine a database hint based on the database utilization statistic and the input query; and
provide the database hint and the input query to the database, wherein the database generates a query plan influenced by the database hint.

15. The computing system of claim 14, wherein:
the database executes the input query using the query plan to obtain query results and provides the query results to the computing system; and
the computer-executable instructions cause the computing system to:
generate a virtual application provided to a client device over a network; and
display at least a portion of the query results on a client device within the virtual application.

16. The computing system of claim 15, wherein the virtual application comprises a customer relationship management application.

17. The computing system of claim 14, the source comprising a querying tenant of a plurality of tenants supported by the database and the database utilization statistic comprising a tenant-specific database utilization metric for the querying tenant, wherein the computer-executable instructions cause the computing system to:
18. The computing system of claim 17, wherein the database hint is an access path for the first table.

19. The computing system of claim 17, wherein the computer-executable instructions cause the computing system to:

   determine, for a second table of the database, a second metric influenced by entries associated with the querying tenant in the second table; and

   determine the database hint based at least in part on the first metric and the second metric.

20. The computing system of claim 19, wherein the database hint is a join order for the first table and the second table.