GUJARAT TECHNOLOGICAL UNIVERSITY

# (GTU)

## INNOVATION COUNCIL (GIC)

## Patent Search & Analysis Report (PSAR)


**Date of Submission :** 24/09/2016

Dear **Rathod Harshrajsinh Vijaysinh,**

| | | |
|---|---|---|
| **Studied Patent Number for generation of PSAR** | : | **16BE7_130020107086_3** |

## PART 1: PATENT SEARCH DATABASE USED

| | | |
|---|---|---|
| **1. Patent Search Database used** | : | Google Patents |
| **Web link of database** | : | https://patents.google.com/ |
| **2. Keywords Used for Search** | : | Database,SQL,Website,SaaS |
| **3. Search String Used** | : | Create database using SQL for SaaS website |
| **4. Number of Results/Hits getting** | : | 2438 |

## PART 2: BASIC DATA OF PATENTED INVENTION /BIBLIOGRAPHIC DATA

| | | |
|---|---|---|
| **5. Category/ Field of Invention** | : | Computer/IT Engineering |
| **6. Invention is Related to/Class of Invention** | : | Related to database management |
| **6 (a) : IPC class of the studied patent** | : | G06F 17/30545 , G06F 17/30 20060101 G06F 017/30 |
| **7. Title of Invention** | : | Database system and method of optimizing cross database query |
| **8. Patent No.** | : | US20110106789A1 |
| **9. Application Number** | : | US12916412 |
| **9 (a) : Web link of the studied patent** | : | https://patents.google.com/patent/US20110106789A1/en?q=database&q=sql&q=website&q=saas&page=2 |
| **10. Date of Filing/Application (DD/MM/YYYY)** | : | 30/10/2009 |
| **11. Priority Date (DD/MM/YYYY)** | : | 30/10/2009 |
| **12. Publication/Journal Number** | : | 20110106789 A1 |
| **13. Publication Date (DD/MM/YYYY)** | : | 05/05/2011 |
| **14. First Filled Country : Albania** | : | United States |

### 15. Also Published as

| Sr.No | Country Where Filled | Application No./Patent No. |
|---|---|---|
| 1 | China | 200910209075.X |

### 16. Inventor/s Details.

| Sr.No | Name of Inventor | Address/City/Country of Inventor |
|---|---|---|
| 1 | Gao Bo | Beijing, CN |
| 2 | Guo Chang Jie | Beijing, CN |
| 3 | Jiang Zhong Bo | Beijing, CN |
| 4 | Sun Wei | Beijing, CN |
| 5 | Tang Kai | Beijing, CN |
| 6 | Wang Feng Juan | Beijing, CN |

### 17. Applicant/Assignee Details.

| Sr.No | Name of Applicant/Assignee | Address/City/Country of Applicant |
|---|---|---|
| 1 | International Business Mac | Armonk<br>NY |

**18. Applicant for Patent is** : Company

## PART 3: TECHNICAL PART OF PATENTED INVENTION

### 19. Limitation of Prior Technology / Art

A database system comprising: a plurality of databases, at least two of the plurality of databases are stored with one or more data collections composed of tables with the same structure, a federated view of the data collection is created, by at least one computer processor, on each of the at least two databases; and a request routing layer for routing, in response to a query request crossing the data collections, the query request crossing the data collections to one of the at least two databases according to a predetermined routing rule, so as to query by using the federated view of the database.

### 20. Specific Problem Solved / Objective of Invention

A computer-implemented method of optimizing cross-database query in a database system comprising a plurality of databases, at least two of the plurality of databases are stored with one or more data collections composed of tables with the same structure, the method comprising the steps of: a federated view creating step for creating a federated view of the data collections on each of the at least two databases; and a routing step for routing, in response to a query request crossing the data collections, the query request crossing the data collections to one of the at least two databases according to a predetermined routing rule so as to query by using the federated view of the database.

### 21. Brief about Invention

The figures form a part of the specification and are used to describe the embodiments of the invention and explain the principle of the invention together with the literal statement.

FIG. 1 shows an exemplary SaaS database system using the scaling out mechanism;

FIG. 2 shows several example cases requiring cross-tenant access;

FIG. 3 shows examples of tables stored in a database;

FIG. 4 is a schematic view showing a SaaS database system according to the first embodiment of the present invention;

FIG. 5 shows a flowchart of process for building a federated view on each database in the SaaS database system according to the embodiment of the present invention;

FIG. 6 is a schematic view showing a SaaS database system according to the second embodiment of the present invention;

FIG. 7 is a weighted directed graph illustrating an example of tracking result obtained by the request tracker 600;

FIG. 8 is a flowchart showing a method of optimizing cross-database query according to the first embodiment of the present invention;

FIG. 9 is a flowchart showing a method of optimizing cross-database query according to the second embodiment of the present invention;

FIG. 10 is a flowchart showing an embodiment of moving step performed by tenant data mover according to the present invention; and

FIG. 11 is a flowchart showing an embodiment of determining processing in FIG. 10.

## 22. Key learning Points

Since the data required for satisfying a cross-tenant query request of one tenant generally include the data in the tenant data of the tenant itself, the data traffic between databases can be reduced by routing using Routing Rule 1. When the tenant data involved in a cross-tenant query request are located in one database, data transmission between databases can even be avoided. For example, data transmission is avoided by routing a request T3 (T3, T4) to the database B. Herein, the request T3 (T3, T4) schematically represents a query request from the tenant T3 that involves the tenants T3 and T4.

With regard to a request T3 (T1, T2, T3), the request will be routed to the database B if Routing Rule 1 is adopted. However, T1 and T2 are located in the database A, data transmission can still be generated between the databases A and B if the query is performed using the database B. In this case, data traffic between databases can be further reduced by using Routing Rule 2. In particular, the sizes a, b and c of T1, T2 and T3 can be obtained from the underlying database by using database command or other mechanism, which, for example, may be the number of records in the tables involved in the current query, in T1, T2 and T3. Then, the data amount (a+b) involved in the database A and the data amount c involved in the database B are compared. If the former is relatively larger, the request T3 (T1, T2, T3) is routed to the database A; and if the latter is relatively larger, the request T3 (T1, T2, T3) is routed to the database B. The cross-tenant query request can be routed to the database that has most of data required for satisfying the request by using Routing Rule 2, thereby reducing data traffic between databases.

The forgoing only illustrates several possible routing rules for routing a cross-tenant query request, and does not intend to enumerate all of the routing rules. Those skilled in the art can understand that many known methods can be applied to SQL routing. The purpose of the above Routing Rules 1 and 2 is to reduce data transmission between databases as much as possible. However, in consideration of load balance, we can also adopt Routing Rule 3: routing a cross-tenant query request to a database with the lowest load based on the statuses of the underlying databases.

For the application of Routing Rule 3, the following case may be considered, for example. When the database (database A) where tenant data of the requesting tenant (such as tenant T1) are located is in a high load status or has a slow responding speed (e.g., lower than a threshold), Routing Rule 3 can be used. A cross-tenant request such as T1 (T1, T2) from the tenant T1 is routed to one of the databases B and C that has the lower load. Herein, the status of the underlying database refers to load, response speed and the like of each underlying database in the SaaS system. The load refers to CPU utilization, memory utilization or the like of a server where each database is located, which can be obtained by known methods such as calling a system function or the like. The response speed refers to the time required for returning a query result. The time can be obtained by timing in the request routing layer.

The request tracker 610 in FIG. 6 is located in the request routing layer for tracking cross-tenant query request behavior in runtime, in particular, for tracking frequency of queries by each tenant, involving other tenants. For example, Log4JDBC in the related art can be used to implement the function of the request tracker 610. Log4JDBC is a JDBC driver capable of recording information such as SQL log and SQL execution time and so on. For example, the request tracker 610 embedded in the request routing layer can track cross-tenant query behavior by analyzing SQL statement or returned ResultSet in runtime. The tracking result of the request tracker 610 is accumulated in the request routing layer, and can be used by the request routing layer to implement routing selection.

The tracking result of the request tracker 610 can be represented in a form of weighted directed graph. FIG. 7 shows an example of the tracking result obtained by the request tracker 610 in a form of weighted directed graph. In FIG. 7, a node denotes a tenant, and a directed side between nodes denotes a cross-tenant query request. For example, the directed side from T1 to T2 denotes that there exists a query request from the tenant T1 involving tenant data of the tenant T2. The weight value w.sub.ij on a directed side represents a frequency of cross-tenant query, where i denotes a serial number of a tenant that sends a query request, and j denotes a serial number of a tenant that is queried. For example, w43 refers to times of query request from the tenant T4 involving tenant data of the tenant T3. When wij is zero, it denotes that there is not a query request from tenant Ti involving tenant data of tenant Tj. The above weighted directed graph can be stored as a simple two-dimension array structure w[i][j]. Of course, other data structures known to those skilled in the art can be also adopted to store the weighted directed graph.

is a flowchart showing a method of optimizing cross-database query according to the second embodiment of the present invention. The processing in FIG. 9 includes a federated view building step 910, a routing query step 920, a tracking step 930 and a moving step 940. The operations in steps 910 and 920 are the same as those in steps 810 and 820 in FIG. 8, and the description thereof is not repeated herein. After the step 920, the process in FIG. 9 proceeds to the step 930 to track cross-tenant data query request behaviors, e.g. queries and frequencies thereof of each tenant involving other tenants, in runtime by using the request tracker 610. In addition, the request tracker 610 accumulates the tracking result, e.g., in a form of weighted array w[i][j], in the request routing layer. The process in FIG. 9 proceeds to the step 940 after the step 930. In the step 940, tenant data are moved among the plurality of databases based on the tracking result of the cross-tenant data query request behavior in off-line time by using the tenant data mover 620.

## 23. Summary of Invention

In order to solve the above problems, an example embodiment of the present invention is in a database system including a plurality of databases, provide a database system capable of optimizing, cross-database query by creating a federated view on each database, and a method of optimizing cross-database query.

The database system and the method of optimizing cross-database query of the present invention can reduce the complexity of programming logic, reduce the data traffic when cross-database query is performed, and balance the query request intensity among databases, and thereby

can increase query speed and efficiency. In addition, the database system and the method of optimizing cross-database query of the present invention can further use a function in a SQL statement of cross-database query.

**24. Number of Claims** : 22

**25. Patent Status** : Granted Patent & In-force Patent

**26. How much this invention is related with your IDP/UDP?**

< 70 %

**27. Do you have any idea to do anything around the said invention to improve it? (Give short note in not more than 500 words)**

After studying each patent thoroughly and by answering all the previous questions, I can say that A large-size SaaS application may have lots of tenants and data, and generally uses a scaling out mechanism as the business increases. The so-called scaling out is to divide the data of the application and to distribute data that should have been collectively stored onto different physical databases according to a certain rule..