The Optimization of Query Processing in Seabase Cloud Databases based on CCEVP Model

Abdulkadir ÖZDEMİR
Faculty of Social Sciences, Ataturk University
Erzurum, Turkey

Hasan Asil
Faculty of Computer, Azarshahr Branch,
Islamic Azad University, Azarshahr, Iran

***Abstract*—A cloud database is a database usually installed on cloud computing software platforms. There are several methods for query processing in cloud databases. This study tried to optimize query processing in the SeaBase cloud database and reduce query processing time. This method used adaptability for optimization. This method was designed for cloud-based databases. The algorithm is composed of three components: 1) multi cloud query separator; 2) query similarity detector based on the execution plan; and 3) replacement policy. This method is implemented as a system for a fully object-oriented simulation. The system is added to the SeaBase as an agent. The evaluation result show that this method reduced response time by 1.9 percent.**

***Keywords*—SeaBase; optimization; query processing; database; adaption**

I. Introduction

Database management systems are software packages that can be used to create and maintain one or more databases. However, with the rise of cloud computing, database management systems have become a new kind of service with unique advantages. In these services, DBMS is a part of a larger service which is likely to be more effective in terms of results and assigned tasks [1].

A cloud database is a database usually installed on cloud computing software platforms. Using a virtual machine, users can independently launch databases on cloud, or they can purchase an account to access database services maintained by cloud database providers [2].

SeaBase is an implementation based on cloud computing. Based on CCEVP model, it can convert different data types into one. In fact, SeaBase is a relational cloud database which can merge a pair of databases together [3]. Like SQL server, DB2, Sybase, MySQL, and other similar databases, SeaBase was designed in order to integrate data taken from several heterogeneous databases and provide users with them in a unified way [4]. Fig. 1 indicates the structure of SeaBase based on CCEVP model.

The CCEVP model uses three layers: 1) physical; 2) virtual; and 3) effective. The physical layer is a set of multisource physical tables from similar or dissimilar databases. The virtual layer is a set of relationship schemas determined by the SeaBase users. The effective layer allows users to create a unified vision to the SeaBase.

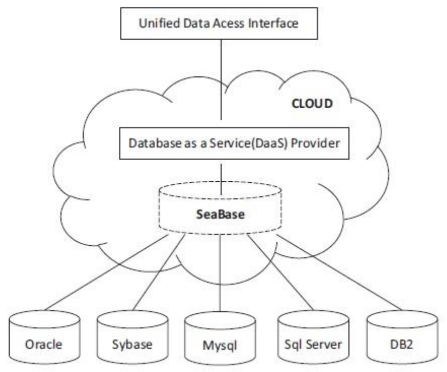

Unified Data Acess Interface
CLOUD
Database as a Service(DaaS) Provider
SeaBase
Oracle
Sybase
Mysql
Sql Server
DB2

Fig. 1. CCEVP Model (Cloud Computing-based Effective-Virtual-Physical) [1].

II. Optimizing Query Processing in Cloud Database Ease of Use

There are several methods for query processing in cloud databases. Many of these methods have offered new technologies to optimize query processing in the database [3]. Some of these methods use replication for query processing and accelerate the process by data sampling. Some methods use traditional methods for query processing in the database. Some methods attempted to optimize the execution plans, which are known as Silinger methods [4], [5].

Most of these methods use a special procedure (Selinger-Style) for optimization, and generally after query processing and query execution, the plan optimized for query processing is eliminated. But today, in addition to these methods, other methods are also being offered to optimize query processing in the database [6], [7]. But the question is that whether the produced optimized plan (or the frequent queries sent to the SeaBase) can be used for executing subsequent queries. As we know, in the application-related databases, usually the queries sent to the database have high adaptability power because in these systems, the queries sent to the database have the same structure and are adapted as soon as possible. This method can be used to optimize the SeaBase.

III. The Proposed Method

The aim of this study was to optimize query processing in the database model. Matching techniques were used to optimize processing. This method was proposed in order to use the optimal schemas generated for the execution of next queries or the ones frequently sent to the database. An agent was added to this model for implementation [8].

Combining the technologies of optimizing queries and agents in this algorithm, a multi-agent system was proposed. Using information collection technology based on processing users' queries, this system tries to provide users with an adaptable environment based on the query types and how they are made [9].

Given the fact that requests are sent to the system by applications or users, the majority of queries have the same structure. They are repeated over time. Therefore, this paper was intended to propose a method by which the database could identify the queries of the same type over time. Using a specific schema of execution, it was also intended to identify the highly frequent queries sent to the database and answer them. Put another way, the database was to be matched so that the queries would be processed with the prepared execution schemas at a lower cost. In fact, it was meant to decrease the number of steps required for processing highly frequent queries sent to the database.

In this algorithm, an agent was added to SeaBase so that a matched cloud database would be generated. Decreasing the number of steps required for processing the highly frequently queries sent to the cloud database, this algorithm tried to increase the optimization of query processing in cloud databases.

This research uses the previous heterogeneous distributed database query processing method and develops it for SeaBase [4], [8].

The algorithm uses a method for optimizing query processing in the heterogeneous distributed databases. This method was designed for cloud-based databases. The algorithm is composed of three components:

- Multi Cloud query Separator
- Query similarity detector based on the execution plan
- Replacement policy

The main objective of this approach is to identify the most frequent instructions sent to the cloud database and store their execution plans in the system so that in case of a request to the database, the same execution plan is used for query execution. The separator part separates instructions and the instructions whose execution plan has not high cost. The query similarity detector is used to identify similar instructions. The replacement policy detects the most frequent instructions sent to the SeaBase and stores their execution plans. The following algorithm shows the structure of this method (Fig. 2).

```
Query processing optimization in SeaBase
    1. Begin
    2. Examine query by separator(can Separate
       distributed query)
    3. Produce query execution plan if query is one of
       exceptions
    4. If it's not an exception check it's execution plan
       availability in system by similarity recognizer
       4.1 If execution plan exists select it
       4.2 otherwise,
       4.3 send it in order to producing execution plan
    5. executing plan for replying to query
    6. Check whether it's the time for substituting or not?
       6.1 If so, do substitution
    7. END
```

Fig. 2. Suggested algorithm.

A. Separator of distributed instructions

Distributed instructions are instructions which include several sub-queries and receive information from several DBMSs.

The purpose of this function is to identify functions that based on assessment, need more time for execution or receive information from several databases. Different queries are sent to the SeaBase, and the database needs cost to respond to queries depending on their type and structure. As mentioned, three layers are used to execute the instructions sent to the database: physical, virtual and effective. Based on the layer, the separator detects instructions that require the use of several databases. The separator aims to identify the instructions that use multiple databases and need a link.

B. The query similarity detector based on execution plan

Any query for execution in the cloud database requires the same steps used in a non-cloud database. Any query for execution must have a specific plan. In applications, requests are usually sent to the database with a specific format and different parameters. The purpose of this section is to identify queries with similar plans.

To make adaptive the query processing in the database, we need a part in the proposed system that can compare the sent queries and identify similar queries. For example, consider the following two queries.

```
SELECT *
FROM tblKala INNER JOIN
tblHavaleKala ON tblKala.KalaiD = tblHavaleKala.KalaID INNER JOIN
tblHavale ON tblHavaleKala.HavaleID = tblHavale.HavaleIDwhere kalaid=20

SELECT*
FROM tblKala INNER JOIN tblHavaleKala ON tblKala.KalaiD = tblHavaleKala.KalaID INNER JOIN
tblHavale ON tblHavaleKala.HavaleID = tblHavale.HavaleIDwhere kalaid=31
```

As can be seen, these two instructions request information on products 20 and 31 from the database. But the two instructions are the same and can be executed with the same plan. This part of the system should be able to detect such instructions.

C. *Replacement policy*

In the algorithm, a set of frequent execution plans must be kept in the agent and in case of request, the same query request should be used. The replacement policy is used to create and update the set. An important part of this research is to determine the replacement action is done how, when and with what policy.

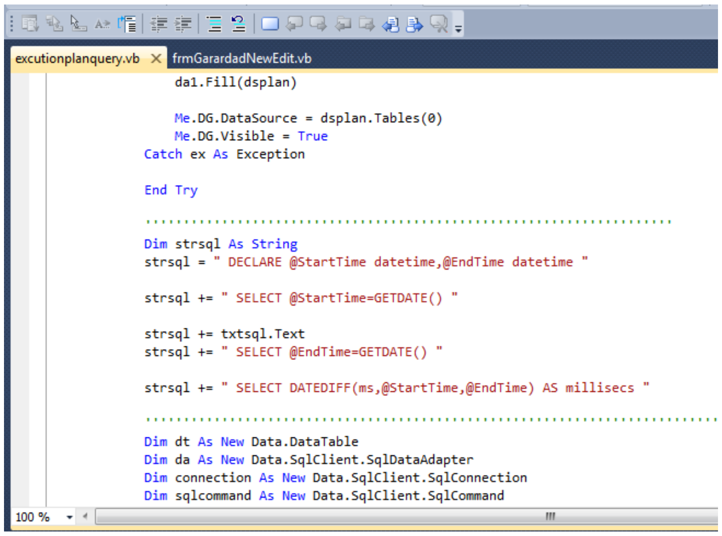
As know adding an agent to SeaBase, which always adapts queries, constrains cost to system. For adapting system the method examines sent queries to database for a while and the execution plan of similar frequent queries is substituted in database. (It's considered that only queries receipted by separator will be sent to this part). The time between two adaptation said that this time is calculated by value of adapted queries and dynamically. It means that whatever score goes up, adaptable queries will stay in the system for longer time and if adapting enjoys low score they will stay in the system for shorter time. It was found that on the long run, the increasing score of adapted queries have increase this time. The method does adapting on queries sent to database in busy hours. The method saves queries in busy hours and in quiet hours it will does adapting on these queries when it's time to adapt.

Now about the ways of adapting, at first create a bank of sent queries then if sent query was similar to one of available queries in database we increase the weight of query and also if sent query was not available in the bank the method adds it to databank and continues adapting. After adapting the method saves queries with high scores. Ways of saving queries in database follows a distinct format and standard in order to constrain less cost when the queries are examined.

IV. Assessment and Practical Results

Several methods are currently used to measure the performance of the database system. One of the most common methods among the above methods is runtime in the system. Runtime is the time from the sending moment to the system response. This study tries to identify the most frequent queries sent to the database and keep their execution plans for executing subsequent queries. In fact, this method tries to make the query processing in the database adaptive.

For assessment, this method is implemented as a system for a fully object-oriented simulation. The system is added to the SeaBase as an agent. Then the results of execution using this method are compared with the SeaBase without this agent. Furthermore, we need the desired data based on relationship dependence. For this purpose, the SQL Toolbelt database and simulator is used to create data and determine the table dependence.The.NET and the SQL API functions are used to implement the algorithm and make comparisons. The following Fig. 3 shows some of the code in this system:



```
        da1.Fill(dsplan)

        Me.DG.DataSource = dsplan.Tables(0)
        Me.DG.Visible = True
    Catch ex As Exception

    End Try

    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

    Dim strsql As String
    strsql = " DECLARE @StartTime datetime,@EndTime datetime "

    strsql += " SELECT @StartTime=GETDATE() "

    strsql += txtsql.Text
    strsql += " SELECT @EndTime=GETDATE() "

    strsql += " SELECT DATEDIFF(ms,@StartTime,@EndTime) AS millisecs "

    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

    Dim dt As New Data.DataTable
    Dim da As New Data.SqlClient.SqlDataAdapter
    Dim connection As New Data.SqlClient.SqlConnection
    Dim sqlcommand As New Data.SqlClient.SqlCommand
```

Fig. 3. Part of the simulation.

After simulation of the system, the following results will be provided.

- The query runtime cost in a normal manner.
- This cost is equal to the time required for the SeaBase query processing and respond to the user. This cost is assessed without adding the agent to the system.
- The cost of the proposed algorithm execution.
- After adding the agent to the SeaBase, the adaptability cost and the query execution cost must be added up and evaluated. The algorithm execution cost is the adaptability cost.
- The execution cost of the adapted query as an execution plan.
- This cost is the execution cost of query with the help of agent. It is worth mentioning that with regard to the adaptability of some queries, the cost of some queries is normal and some less.

After obtaining the above results, the second and third costs are added up and compared with the first cost.

In the algorithm, the times required for executing the queries sent to the database are compared in adaptive and non-adaptive databases. Fig. 4 shows the time required to respond to the adaptive and non-adaptive queries per day. A cloud database with adaptive queries is called adaptive cloud base.

This diagram shows the total time required for executing adaptive queries in the database as well as the total time for executing adaptive queries in the non-adaptive mode. It should be noted that in this figure, the adaptability cost is not currently added to the above calculations because the system is not adaptive at any time and will do this action only at certain times of low traffic.

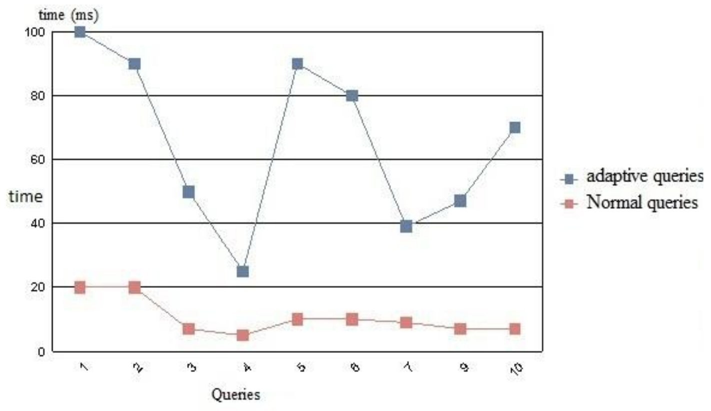


Fig. 4. Report of response time per day for adaptive queries.

However, these costs will be taken into account in the next assessments.

Fig. 5 shows the reduced time of executing adaptive queries. These queries in the SeaBase are queries which have become adaptive. Obviously, due to making high-traffic queries adaptive, this method reduces the server workload at times of high traffic.

The first row of Table 1 represents the total time for responding to adaptive queries and reduced time of response time for all adaptive queries sent to the database. It also shows the cost of making queries adaptive.

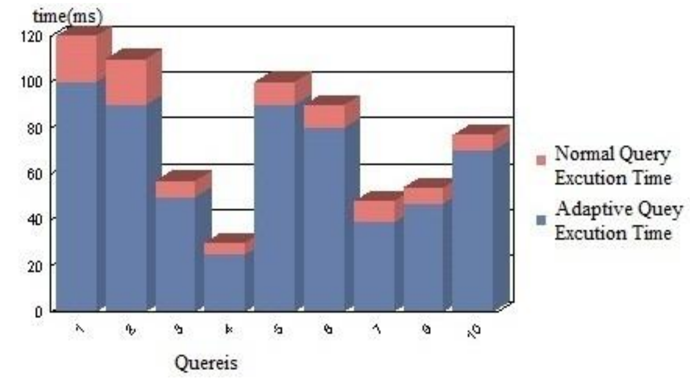


Fig. 5. Report of reduced cost of adapted query.

TABLE I. TOTAL SYSTEM EVOLUTION

Row	*Type of queries*	*Decrease response time*	*Total Execution time*
1	Distributed queries(Join)	15%	100%
2	All queries	1.9%	100%

In this system, when sending queries to the database, the query separator separates some queries and blocks their way to the database. The second row represents time and cost for all queries sent to the database plus adaptability cost.

As shown in Table 1, the system reduced response time by 1.9 percent.

V. CONCLUSION

The increase in data volume in many applications and the need for their calculations are the database challenges. Cloud computing and the use of SeaBase databases are a solution to integrate a variety of DBMSs and integrated access to tables in databases. This study tried to optimize query processing in the SeaBase cloud database and reduce query processing time. This method used adaptability for optimization. The purpose of this method is to make adaptive the execution plans of high-traffic queries sent to the SeaBase. For adaptability, this method uses three parts: separator, similarity detector and replacement policy. This method is added to the database as an agent. The results show that the system optimizes query processing in the database and reduces response time by one percent. Based on the replacement policy, this method also reduces workload. In the future, response time can further decrease by changing the replacement policy.

REFERENCE

[1] Shusheng Guo ; State Key Lab. of Comput. Syst. & Archit., China ; Zimu Yuan ; Li Zha ; Zhiwei Xu, “SeaBase: An Implementation of Cloud Database”, 10th International Conference on Semantics, Knowledge and Grids (SKG), Beijing, 2014

[2] Waleed Al Shehri, CLOUD DATABASE DATABASE AS A SERVICE, International Journal of Database Management Systems (IJDMS) Vol.5, No.2, April 2013

[3] Clayton Maciel Costa, Adaptive Query Processing in Cloud Database Systems,IEEE International Conference on Cloud and Green Computing,2013

[4] Ennaz zafarani , Mohammad_Reza Feizi_Derakhshi , Hasan Asil , Amir asil “Presenting a New Method for Optimizing Join Queries Processing in Heterogeneous Distributed Databases” , WKDD2010, Phuket , Thailand , 9-10 January, 2010.

[5] Mohammad_Reza Feizi_Derakhshi , Hasan Asil , Amir Asil,ennaz zafarani “Optimizing Query Processing in Practical Software Database by Adapting” WKDD2010, Phuket , Thailand , 9-10 January, 2010.

[6] Mohammad_Reza Feizi_Derakhshi, Hasan Asil, Amir Asil “Proposing a New Method for Query Processing Adaption in Data Base " WCSET 2009: World Congress on Science, Engineering and Technology Dubai, United Arab Emirates VOLUME 37, January 28-30, 2009 ISSN 2070-3740

[7] Amol Deshpande, Zachary Ives, and Vijayshankar Raman. Adaptive query processing. Foundations and Trends in Databases, 1(1), 2007.

[8] Mohammad_Reza Feizi_Derakhshi , Hasan Asil , Amir Asil,elnaz zafarani “Practical Software Query Optimizing by Adapting Why and How?” Australian Journal of Basic and Applied Sciences, jourdan , January,2010

[9] Agent Working Group, “Agent Technology Green Paper”. OMG Documentagent/00-09-01 Version 1.0, 2000.