An Equijoin-Optimization Technique For Malaysian Hydrological Information System (MHIS) Data

Norazrin Kurmin, Harihodin Selamat, Mohd Taib Wahid, Mohd Shafry Mohd Rahim
Faculty of Computer Science and Information System
Universiti Teknologi Malaysia,
81310 Skudai, Johor.
E-mail : azrin@siswa.utm.my, harihodn@itp.utm.my, taib@fsksm.utm.my, shafry@fsksm.utm.my

ABSTRACT

Malaysian Hydrological Information System (MHIS) is one of the GIS applications that storing a large amount of data (such as rainfall, water level, evaporation and water quality). In such situation, it is important to ensure that the storage, retrieval and manipulation of these data are efficiently handled. However, the problem in MHIS is inefficient on retrieval data where it takes a long time to retrieve data. MHIS was developed based on cube system concept. The main objective introduced the cube system is to describe how the hydrology data will be store and retrieve in relational database. However, it is still lacking to handle a large amount of data. Thus, this paper proposed an Equijoin Optimization Technique to solve that problem. This technique was introduced based on modification of an equijoin method where an entity will be joined based on equijoin algorithm. In other words, the result relation is a new entity, T, contains tuples t made up of two tuples r and s, where r must be tuple in entity R and s must be tuple in entity S. Then, there are entity will be classified to other entity based on the appropriate attribute. Based on this technique, a SQL statement was created for data retrieval processing which is referring to the optimized entity. There are four main function in that process i) Select ii) Initialize iii) Operation and iv) Output. Also, in this research, testing and comparison analysis has been done between proposed technique, an Equijoin optimization and the previous technique MHIS Cube System. The result of the testing shows that the proposed technique can improve an execution time for query processing, then solve the issue has been discussed on this research.

KEY WORDS

Temporal data, hydrology data, database, join, retrieval, query optimization.

1. Introduction

An effective system for information system and analysis is becoming more important with the continuous and rapid growth of source and user information. With the rapid growing, “click” becoming more powerful, the world becoming a global village and we living in the information age. This phenomenon is motivated into finding out ways to achieve the faster response time for query processing in a database.

A database system is a collection of information, organized and presented to serve a specific purpose, where a database management system (DBMS) is the software that manages the database system. The information and data are stored in the database based on certain rules, which assure that they can be efficiently retrieved and modified. However, in such application, there is need to handle temporal data which time period attached to the data. For example, hydrology data constitutes of past, present and future state data. As the amounts become larger, this kind of database requires more complex of handling and processing data. Based on the literature review, the main issue that received big attention among the researchers is on data retrieval, which is low response time for data retrieval. Based on the literature review, query optimization is one of the techniques that can solve that problem [3, 4, 5, 6]. Its goal is to achieve efficient processing of user queries that retrieve data from an often very large database.

2. Temporal Data

For some application such as hydrological information system (HIS), time is an important element to be stored. It is a measurement that records history in a human life[1]. All the changes must be recorded, that is may be uses for analysis and forecasting. Data analysis can be more efficient with take into account the time element. Any information having a time component known as temporal data and can be represented in a general way in a temporal database. Temporal data stored in temporal database is different from the data stored in non temporal database in that a time period attached to the data.

2.1 Case Study : Malaysian Hydrological Information System (MHIS) Database

2.1.1 Hydrology Data

In general, Jabatan Pengairan dan Saliran (JPS) is responsible to handle hydrology data in Malaysia. JPS is responsible to manage, retrieve and analysis hydrology data includes rainfall data, water level data, water quality data, evaporation and water sediment data. A large database is needed to store these data properly so that analysis management and report representation can be efficient and precise regarding to user requirement[2].

2.1.2 Cube System Concept

Malaysian Hydrological Information System (MHIS) was developed based on cube system concept. The main objective introduced the cube system is to describes how the hydrology data will be store and retrieve in relational database. There are three main coordinate: Feature Identity (FID), Attribute Identity (DID) and Date and Time (TID). These coordinates are use to determine the data that stored in the database. FID refers to data value that had measured or value key for spatial data. DID refers to key that describes about FID that is key for spatial data that had taken. TID is a key for the time the data is taken. These features will be referred when accessed data from the database. Figure 1 describes the logical view of cube system.

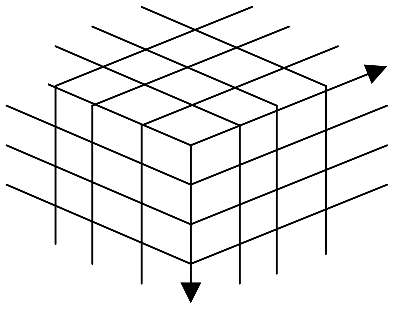

Figure 1: Cube System Logical View

This system was implemented using two different database system; Informix RDBMS and ArcInfo. Informix RDBMS is used for storing attribute data or non spatial data, then, ArcInfo is used for storing spatial data. This technique is lacking to handle a large amount of data.

3. Related Work (Optimization Technique)

There has been extensive work in query optimization and many algorithms have been developed using join method such as semijoin, two way semijoin, and composite semi join. The join operation is one of the fundamental relational database query operations. In 1983[7], Aper, Hevner and Yao proposed an optimization algorithm called Algorithm General (AHY algorithm), which mainly focused on minimize response time. It was developed using semijoin method. This algorithm provides a very efficient execution for simple queries. However, one important drawback of AHY algorithm is that static in nature. It means that once a query plan has been made, there is no way to change the plan. It strictly follows a given pattern and provides the response based on the plan.

Lubna Sachawani[8] proposed a JAL dynamic algorithm which is an enhancement of AHY algorithm. It makes query processing dynamic. The JAL dynamic algorithm modifies the AHY algorithm by making use of the profit concept. Also, like AHY algorithm, this algorithm was developed based on semi-join method. It is use this method as a relation reducer in database.

Other techniques to reduce query response time have also been proposed such as Jing Chen[5] proposed an optimization technique using histogram-based method. It was using an equijoin method to join two relations that has same attribute value, then implement it using histogram-based algorithm, Averaged R-ACM and T-ACM to improve the query processing execution.

4. An Equi-Join Optimization Technique

An EQ-Optimization technique was developed based on enhancement of equijoin method. In this technique, there is an entity will be join, then produce a new relation (Figure 2 (a)). In other words, assume a new entity, *T*, the result entity contains tuple t made up of two parts, *r* and *s* where *r* must be tuple in entity *R* and *s* must be tuple in entity *S*. In each tuple *t*, the value of the attributes is an identical value of *r* and *s*.

Then, an enhancement from an equijoin method is classified an entity into several entity based on appropriate attribute (Figure 2(b)). The main objective is to reduce a number of tuple for that entity, then can improve the response time where it's only refer to the related entity. In general, Figure 2 describes the design of the Equi-join Optimization technique.

Next, query processing phase was executed based on the optimization technique above. There are four main function includes i) *Select* ii) *Initialize* iii) *Operation* and iv) *Output.*

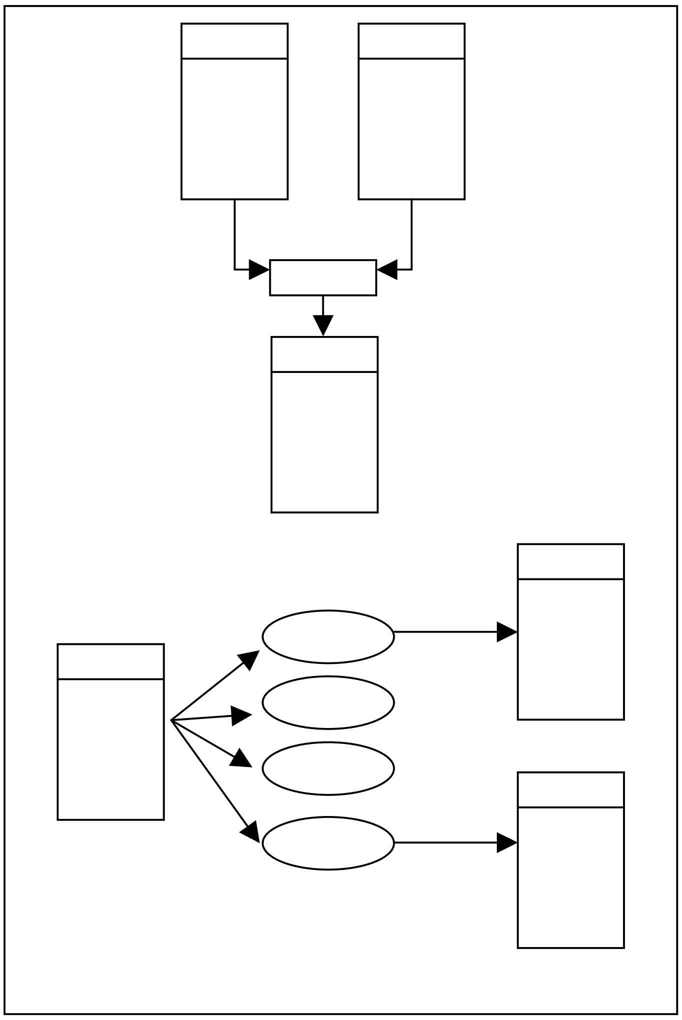

Figure 2 : An Equi-join Optimization Technique

4.1 Design of Equi-join Optimization Technique.

There are many optimization design was proposed based on a set of rules or heuristics where it intended to produce query execution with low cost such as execution time or memory usage[9, 10]. Hence, Equi-join Optimization was produced based on set rules as following:

i. Eliminate redundant.
ii. Combine entity into one.
iii. Classify entity into several entities based on appropriate attributes.

4.1.1 Eliminate Redundant Attribute

In this rule, redundant attribute will be eliminated for the joined attribute. It means, attribute that has same value for the joined attribute will be eliminated and place that value in a new entity. In simplest, this rule can represent as following:

$$\{ r' = r_1 \cap r_2 \mid r_1 \in R_1, r_2 \in R_2 , r' \in R'\}$$

From that equation, we can describe that R' is a new entity, the value r' is an intersection of r_1 and r_2 where r_1 and r_2, is a subset of R_1 and R_2 respectively. The main objective of this rule is to collect the same attribute into one entity. It also can reduce redundant information and reduce response time for retrieve data from database.

4.1.2 Combine Entity Into One

Based on this rule, there has entity will be joined and placed a result of join into a new entity. The value of attribute for a new entity is an attribute value that gets from rule (i) as describes above. For proposed technique, equality operator is used as a comparison operator. This rule can be defined as following:

$$R' = R_1 \bowtie_{r_{1(a)}=r_{2(b)}} R_2$$

Where R' is a new entity, result of join R_1 and R_2 that has same attribute value $r_{1(a)}=r_{2(b)}$. Also, main objective of this rule is to reduce response time, entity and relation between entities. Data retrieval processing can be efficiently, where it's only refer to one entity compared from previous, needs to refer two or more entity.

4.1.3 Classify Entity Into Several Entities Based On Appropriate Attribute.

Same as rule (i) and (ii), objective of this rule is to make the retrieval process efficiently handle. Entity has to classify into several entities based on appropriate attributes.

4.1.4 An Equi-join Optimization Algorithm

In general, all the rules have discussed above can be represented with the following equation:

$$R' \equiv (R_j \cup R_{j+1} \mid \sum_{i=1}^{n} \{r_{ji}=r_{j+1,i},\ r_{j,i+1}=r_{j+1,i+1},\ldots\ldots,r_{j,n}= r_{j+1,n}\})$$

Where R'= new entity (result entity)
R_j = entity that will be join; j = 1, 2, 3 ...n
n = number of tuple

Based on that equation, the value of R' is a combination of value from R_j dan R_{j+1}. For each tuple R_j dan R_{j+1}, attribute value for each entity (r_j and r_{j+1}) will be compare using the equality operation. Whenever the condition is satisfied, the two tuples are concatenated and placed into a new entity R'. Figure 3 is an algorithm for the Equi-join Optimization technique.

```
1.  For each join entity, $R_1$ and $R_2$ , do
    1.1 For each tuple $R_1$, do
          a.  For each tuple $R_2$, do
                If attribute $R_1(a) = R_2(b)$ then
                    concatenate $R_1$ and $R_2$
          b.  Place in new relation, R'
2.  Create a variable number of entity
    and set initialize, For example : number_of_entity=0
3.  For each classify entity, $R_3$, do
    a.    Assign the first value of tuple,$t_1$, to
           an array variable,
          i.e. array[number_of_entity] = $t_1$
           i.    for each tuple,$t_n$ in $R_3$ ,do
                   if attribute $R_3(t_n)$ = array[number_of_entity]
                      then eliminate $t_n$
          ii.    Place tn in new relation $R_{t1}$
4.  number_of_entity = number_of_entity + 1
5.  Repeat step 3, 4 while number of tuple <> 0
```

Figure 3 : Table Optimization Algorithm (Equi-join Optimization)

4.2 Query Implementation Process.

In this technique, query is refer as a set of operation that processing and analyzing a set input data, then produce required output. Figure 4 depicts loops for query processing of this technique.

```
(*Data Set *)
I  ← Input
O ← Output
1.    $S_i$ ← Select (I, $q_i$)
      (*Initialize*)
2.    for each $S_i$ do
3.          $I_j$ ← Initialize ($S_i$)
      (*Process*)
4.    for each $I_j$ do
5.          $P_j$ ← Operation ($I_j$)
6.          $O_j$ ← Output ($P_j$)
```

Figure 4 : Query Processing Loop

In above figure, data set can be classified as an input (I) or output (O). Input is a data where output is a data was produced from the execution process. *Select* function (Step 1) will retrieve data using Structured Query Language (SQL) statement (q_i), then assigned to S_i. Next, several *Operation* function (Step 5) will apply to a data set and produced a required output (Step 6).

5. Conclusion

The proposed technique can be applied for any database application that has many relations with large amount of data. In this situation, user has to determine an appropriate attribute for a join and classification condition.

As a conclusion, we can say that join processing and entity classification has been done to make sure that response time for query processing can be improved and data retrieval in a database can be executed efficiently.

6. Acknowledgements

This work has been done under research grant projek IRPA RMK 8 Vot Number 74074 and Vot Number 74134, which are sponsored by Ministry of Science Technology and Innovation (MOSTI). We would also like to thank the Faculcy Science Computer and Information System (FSKSM) and Research Management Center (RMC) Universiti Teknologi Malaysia for the management of the research and facilities.

7. References

[1] Mohd Shafry Mohd Rahim." Spatial and Non Spatial Enhancement Database for Hydrological Information System (HIS)". Master Thesis. UTM Skudai, 2002.

[2] SMHB ITP (1999). "Establishment of Hydrological Water Resources Planning, Development and Management, Jabatan Pengairan dan Saliran Malaysia, Kuala Lumpur, Technical Report, Volume 1, 2, 3, 4.

[3] Shamkan B. N and Rafi Ahmed. " Temporal Extension to the Relational Model and SQL In: Tansel et al. *Temporal Databases : Theory, Design and Implementation, eds.* California: The Benjamin/Cumming Publishing, Inc. 92-109;1993.

[4] Haraty R and Fany R. Distributed Query Optimization using PERF Join. SAC'00 March 19-21, 2000. Como, Italy:ACM.2000:284-288.

[5] Jing Chen, B.Cs. "On Utilizing New Histogram-Based Methods for Query Optimization". Master Thesis, Carleton University, 2003.

[6] McMahan B.J. "Structural Heuristics for Query Optimization".Master Thesis. Rice University, Houston, Texas, 2004.

[7] Alan R.H, Wu O Q and Yao S B (1985), "Query Optimization on Local Area Networks". ACM Transaction on Office Information, 3910:35-62.

[8] Lubna Sachwani. "Dynamic Techniques in Distributed Query Optimzation". Master Thesis. University of Windsor, 2002.

[9] Christian M (2002). "A Survey of Database Query Optimization and Genetic Algorithms"

[10] Slivinskas G et al (2001). Adaptable Query Optimization and Evaluation in Temporal Middleware