Database Partitioning Using Frequent Item Set for Getting Optimal Solution

Shraddha Bhor[1], Jyotsna Gavali[2], Rekha Malviya[3]

[1,2,3] Student, Computer Department, Dr. D. Y. Patil Institute of technology, Pimpri, Pune

Abstract: *We present an idea integrating frequent pattern clustering for finding an finest database partition. First, the Apriori algorithm is used to find out weighted frequent patterns related to a operation profile. Depending on the weighted frequent patterns, we build up two techniques designed for division of database: the candidate method and the optimal method. The finest technique contain a branch-and-bound algorithm and costs in each step of combining attributes can be taken into account until an optimal solution is obtained. Furthermore, we refined the optimal method for expediting the execution by minimizing the exploration space. Finally, results show that the given method performs the highest among all previous methods.*

Key Words: *Data partitioning, Frequent itemsets, Apriori Algorithm, Cosine Similarity, Branch and Bound technique.*

1. INTRODUCTION

In a relational database scheme, efficiency is dominant while a operation is processed. For accessing databases, a huge quantity of data can be take out in a operation, but a few of them can be inadequate in handing. consequently, a number of approaches for example indexing and data separation have been proposed for accessing databases more effectively and reducing disk input/output (I/O).

In common, database design can be separated into rational and physical database design. For logical database design, a database administrator choose the text of the record at an conceptual level or generate the abstract scheme. For physical database design, the record administrator also decides how the data must be characterize in database. Once database design is completed, the database schema would work for a long time until the schema must be modified. Therefore, determine an optimal database design is critical to the performance of database. Application that involve accessing data in structure-variant manner are not suitable for using a database.

Here , we presents an idea for detecting an finest partition. First, the modified Apriori algorithm and cosine similarity are merged to find out weighted frequent patterns. The modified Apriori algorithm is familiar with finding out the combination of pattern frequently accessed in transactions. Cosine similarity is used to calculate the similarity of pattern, which are them fused into a weighted frequent pattern. after that a branch and bound algorithm is utilized to create an finest database division according to a cost structure.

2. RELATED WORK

This paper gives AutoPart,an algorithm that mechanically partition database tables to minimize in order access to prior knowledge of a representative workload. The scheme is indexed using a fraction of the space required for indexing the original schema. This experiment can be AutoPart in the context of the Sloan Digital Sky Survey database, and a real-world astronomical database, running on SQL Server 2000 is used for partitioning.[1]

This paper highlighted on issues (a) the importance of taking an integrated approach to automated physical design and (b) the scalability of the techniques. Techniques that enable a scalable solution to the integrated physical design problem of indexes, materialized views, vertical and horizontal partitioning for both performance and manageability. The paper focus on horizontal partitioning with on single-node partitioning[2]

The paper gives an approach integrating frequent pattern clustering and branch – and bound algorithms for finding an optimal database partition. Apriori algorithm and cosine similarity are used to determine weighted frequent patterns according to a transaction profile.[3]

3. PROPOSED SYSTEM

1. List out popular Queries on Database for grocery chain application

Popular SQL queries need to be extracted from the application code and they need to be analyzed for performance bottleneck. List of database columns and their tables need to be extracted.

2. Create Transaction Profile

Each query is assigned a transaction id and its columns are marked with 1/0. Transaction profile helps us identify common columns occurring in all the queries.

Transaction profile.

	a1	a2	a3	a4	a5	a6	Freq.
Query1	0	1	1	1	0	0	15
Query2	1	1	0	0	1	1	10
Query3	0	0	0	1	1	1	25
Query4	0	1	1	0	0	0	20

3. Apply Apriori algorithm

The Apriori algorithm is designed for removal frequent object sets in a transactional database. Here, we apply this algorithm to learn the grouping of features commonly accessed by transactions. certain columns are referred in maximum of queries, we extract such columns and find their weights.

4. Detect Weighted Frequent Patterns

In the first process, we combine the Apriori algorithm and cosine similarity to find out weighted frequent patterns. In the predictable Apriori algorithm, the prototype is used to confirm whether the model is common. though, here we make use of the frequency of a query and cosine similarity as the weighted support of a pattern to validate if the pattern is weighted frequent. First, the original formula of cosine similarity used between two objects is modified for use among n objects as follows:

$$Sim(x_1, x_2, \ldots, x_n) = \frac{x_1 \cdot x_2 \cdots x_n}{\|x_1\| \; \|x_2\| \cdots \|x_n\|}$$

weighted support of a pattern is then defined as follows:

$$WS(pattern) = \left(\frac{pattern's\ freqency}{total\ frequency}\right) \times (pattern's\ cosine\ similarity)$$

5. Vertical Partitioning

Vertical partitioning involves dividing a table into multiple fragments in which each fragment contains the same number of rows but fewer columns compared with the original table. For vertical partitioning, many methods are used for splitting an original table because many combinations of attributes can be applied for vertical partitioning. For example, we can split a three-attribute table {a, b, c} into five combinations of attributes i.e { (a),(ab,c),(a,bc),(a,b,c),(ac,b)}

6. Compare Results of with partitioning and without partitioning

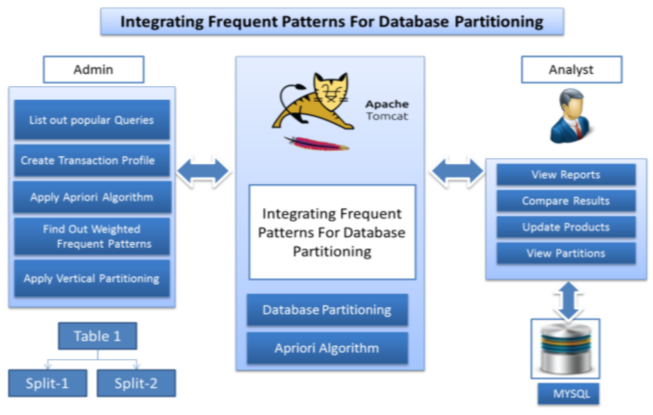

Integrating Frequent Patterns For Database Partitioning
Admin
List out popular Queries
Create Transaction Profile
Apply Apriori Algorithm
Find Out Weighted Frequent Patterns
Apply Vertical Partitioning
Table 1
Split-1
Split-2
Apache Tomcat
Integrating Frequent Patterns For Database Partitioning
Database Partitioning
Apriori Algorithm
Analyst
View Reports
Compare Results
Update Products
View Partitions
MYSQL

Fig-1: Proposed System Architecture

To estimate the performance of whole unit, We have to install resource monitoring and load balancing tools on the test bed and evaluate the need of available resources such as Storage Pricing, CPU pricing, Request Pricing and Storage Management Price. The solution should be able to geographically distributed and accommodate a high number of customers and brokers.

4. ALGORITHM USED

1. Modified Apriori Algorithm

- Let us consider the database D.
- Let us assume minimum support as 2.
- For 1-Frequent item set L1
- Firstly, scan all transactions to get frequent 1-itemset L1 which contains the items and their support count and the transactions ids that contain these items, and then eliminate the candidates that are infrequent or their support are less than the minimum support.

Itemset	Support	Transaction ID set
Apple	6	{1,4,5,7,8,9}
Milk	7	{1,2,3,4,6,8,9}
Bread	6	{3,5,6,7,8,9}
Crisps	2	{2,4}
Beer	2	{1,8}

2-Frequent itemset c2

- The next step is to generate candidate 2-itemset from L1. To get support count for every itemset, split each itemset in 2-itemset into two elements then use l1 table to determine the transactions where you can find the itemset in, rather than searching for them in all transactions.
- For instance, let's acquire the initial item (Apple, Milk), in the original Apriori we scan all 9 transactions to find the item (Apple, Milk); but in our proposed modified algorithm we will split the item (Apple, Milk) into Apple and Milk and get the minimum support between them using L1, here Apple has the smallest minimum support .After that we search for itemset (Apple, Milk) only in the transactions 1, 4, 8 and 9.

C2 is shown as:

Itemset	Support	Minimum	Transaction ID set
{Apple, Milk}	4	Apple	{1,4,8,9}
{Apple, Bread}	4	Apple	{5,8,9}
{Apple, Crisps}	1	Crisps	{4}
{Apple, Beer}	2	Apple	{1,8}
{Milk, Bread}	5	Bread	{3,6,8,9}
{Milk, Crisps}	2	Crisps	{2,4}
{Milk, Beer}	2	Beer	{1,8}
{Bread, Beer}	1	Bread	{8}

L2 is shown as:

Itemset	Support	Minimum	Transaction ID set
{Apple, Milk}	4	Apple	{1,4,8,9}
{Apple, Bread}	4	Apple	{5,8,9}
{Apple, Beer}	2	Apple	{1,8}
{Milk, Bread}	5	Bread	{3,6,8,9}
{Milk, Crisps}	2	Crisps	{2,4}
{Milk, Beer}	2	Beer	{1,8}

3-Frequent Itemset C3& L3 are as follows:

Itemset	Support	Minimum	Transaction ID set
{Apple, Milk, Bread}	2	{Apple, Milk}	{8,9}
{Apple, Milk, Beer}	2	{Milk, Beer}	{1,8}

1. Weight Calculation for patterns

First, the original formula of cosine similarity used between two objects is modified for use among n objects as follows:

$$Sim(x_1, x_2, \ldots, x_n) = \frac{x_1 \cdot x_2 \cdots x_n}{\|x_1\| \; \|x_2\| \cdots \|x_n\|}$$

The weighted support of a pattern is then defined as follows

$$WS(pattern) = \left(\frac{pattern's\ freqency}{total\ frequency}\right) \times (pattern's\ cosine\ similarity)$$

Mathematical Model:

(a) Let S be the system.

S={I}

Identify I as input, Set of Queries

I={i1,i2,i3...in}

I1,i2,i3…in -> 'n' number of food items

S={I}

(b) Identify P as process

P={F,D}

Where,

F -> Frequent itemsets from database using modified apriori.

D -> Partitions of Database using frequent itemsets.

S={I,P}

(c) Identify O as Output.

O={p,q}

Where,

P->Time complexity to execute querry on partitioned database.

Q->Time complexity to execute querry on non-partitioned database.

S={I,P,O}

(d) Identify A as case of success.

A={J,K}

J ->Querries are valid

K -> Time complexities of partition is greater than non-partition database.

S={I,P,O,A}

(e) Identify F as case of success.

F={L,M}

L ->Querries are valid

M -> Time complexities of partition is not greater than non-partition database.

S={I,P,O,A,F}

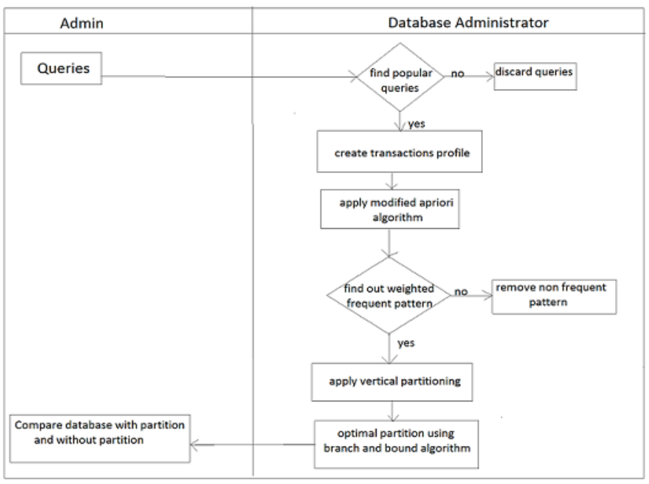

Admin
Database Administrator
Queries
find popular queries
no
discard queries
yes
create transactions profile
apply modified apriori algorithm
find out weighted frequent pattern
no
remove non frequent pattern
yes
apply vertical partitioning
optimal partition using branch and bound algorithm
Compare database with partition and without partition

Fig - 2: Project Flow

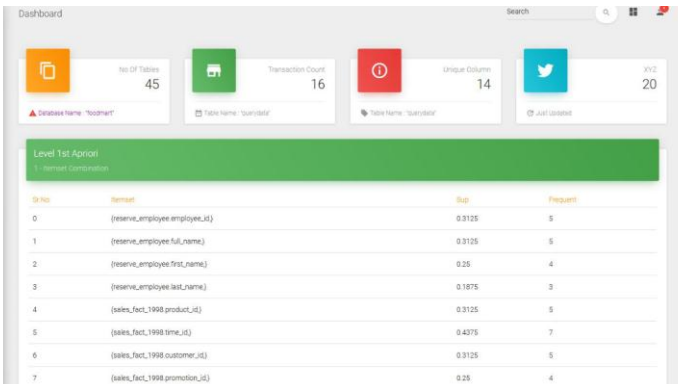
5. IMPLEMENTATION DETAILS

The proposed system is supported to android studio 1.0 which provides tools for android development and debugging. An open source SDK used for hybrid mobile

APP development like, android and iOS. Software programmer bulid application for mobile devices using Javascript and HTML6. Node. jsused for developing a diverse

variety of server tools and applications which are open source, cross-platform, Javascript runtime environment. The systems GUI was designed using java JSP. Core Technologies used were Java, JSP. The overall development was done in the Eclipse Juno and for DB we used MY SQL GUI browser.

Apriori algorithm and cosine similarity is used to determine weighted frequent patterns to ascertain the combination of attributes. Figure shows the item set combination of level first apriori algorithm.



No Of Tables 45
Database Name : 'foodmart'

Transaction Count 16
Table Name : 'querydata'

Unique Column 14
Table Name : 'querydata'

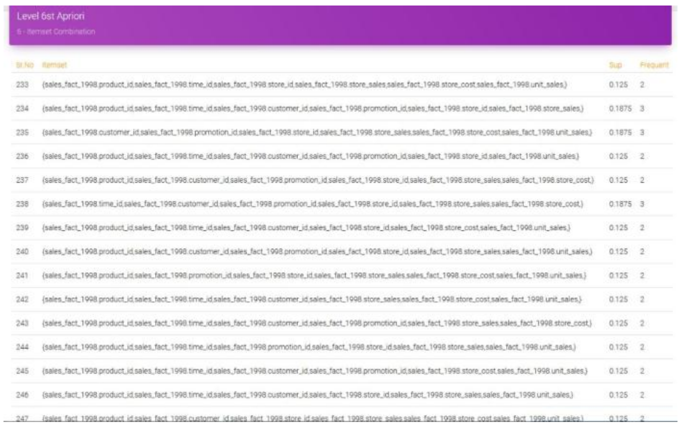
XYZ 20
Just Updated

Level 1st Apriori
1 - Itemset Combination

Sr.No	Itemset	Sup	Frequent
0	{reserve_employee.employee_id}	0.3125	5
1	{reserve_employee.full_name}	0.3125	5
2	{reserve_employee.first_name}	0.25	4
3	{reserve_employee.last_name}	0.1875	3
4	{sales_fact_1998.product_id}	0.3125	5
5	{sales_fact_1998.time_id}	0.4375	7
6	{sales_fact_1998.customer_id}	0.3125	5
7	{sales_fact_1998.promotion_id}	0.25	4

Fig -3: Level 1st Apriori

Item set combination of level 6 apriori is shown below.



Level 6st Apriori
6 - Itemset Combination

Sr.No	Itemset	Sup	Frequent
233	{sales_fact_1998.product_id,sales_fact_1998.time_id,sales_fact_1998.store_id,sales_fact_1998.store_sales,sales_fact_1998.store_cost,sales_fact_1998.unit_sales}	0.125	2
234	{sales_fact_1998.product_id,sales_fact_1998.time_id,sales_fact_1998.customer_id,sales_fact_1998.promotion_id,sales_fact_1998.store_id,sales_fact_1998.store_sales}	0.1875	3
235	{sales_fact_1998.customer_id,sales_fact_1998.promotion_id,sales_fact_1998.store_id,sales_fact_1998.store_sales,sales_fact_1998.store_cost,sales_fact_1998.unit_sales}	0.1875	3
236	{sales_fact_1998.product_id,sales_fact_1998.time_id,sales_fact_1998.customer_id,sales_fact_1998.promotion_id,sales_fact_1998.store_id,sales_fact_1998.unit_sales}	0.125	2
237	{sales_fact_1998.product_id,sales_fact_1998.customer_id,sales_fact_1998.promotion_id,sales_fact_1998.store_id,sales_fact_1998.store_sales,sales_fact_1998.store_cost}	0.125	2
238	{sales_fact_1998.time_id,sales_fact_1998.customer_id,sales_fact_1998.promotion_id,sales_fact_1998.store_id,sales_fact_1998.store_sales,sales_fact_1998.store_cost}	0.1875	3
239	{sales_fact_1998.product_id,sales_fact_1998.time_id,sales_fact_1998.customer_id,sales_fact_1998.store_id,sales_fact_1998.store_cost,sales_fact_1998.unit_sales}	0.125	2
240	{sales_fact_1998.product_id,sales_fact_1998.customer_id,sales_fact_1998.promotion_id,sales_fact_1998.store_id,sales_fact_1998.store_sales,sales_fact_1998.unit_sales}	0.125	2
241	{sales_fact_1998.product_id,sales_fact_1998.promotion_id,sales_fact_1998.store_id,sales_fact_1998.store_sales,sales_fact_1998.store_cost,sales_fact_1998.unit_sales}	0.125	2
242	{sales_fact_1998.product_id,sales_fact_1998.time_id,sales_fact_1998.customer_id,sales_fact_1998.store_sales,sales_fact_1998.store_cost,sales_fact_1998.unit_sales}	0.125	2
243	{sales_fact_1998.product_id,sales_fact_1998.time_id,sales_fact_1998.customer_id,sales_fact_1998.promotion_id,sales_fact_1998.store_sales,sales_fact_1998.store_cost}	0.125	2
244	{sales_fact_1998.product_id,sales_fact_1998.time_id,sales_fact_1998.promotion_id,sales_fact_1998.store_id,sales_fact_1998.store_sales,sales_fact_1998.unit_sales}	0.125	2
245	{sales_fact_1998.product_id,sales_fact_1998.time_id,sales_fact_1998.customer_id,sales_fact_1998.promotion_id,sales_fact_1998.store_cost,sales_fact_1998.unit_sales}	0.125	2
246	{sales_fact_1998.product_id,sales_fact_1998.time_id,sales_fact_1998.customer_id,sales_fact_1998.store_id,sales_fact_1998.store_sales,sales_fact_1998.unit_sales}	0.125	2
247	{sales_fact_1998.product_id,sales_fact_1998.customer_id,sales_fact_1998.store_id,sales_fact_1998.store_sales,sales_fact_1998.store_cost,sales_fact_1998.unit_sales}	0.125	2

Fig - 4: Level 6th Apriori

Figure shows the comparison between the simple apriori and modified apriori. Time taken by apriori algorithm for making the combination of the itemset is reduced in modified apriori algorithm.

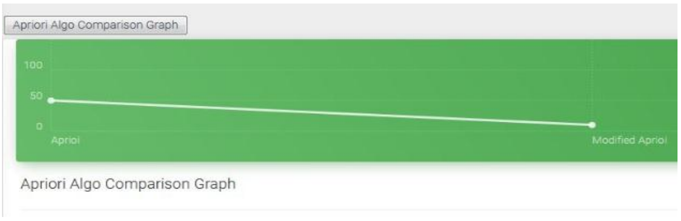


Figure 5: Apriori Algo Comparison Graph

After making all the combination of itemset the time required for searching particular combination without partitioning is greater than that of with partitioning as shown in below figure.

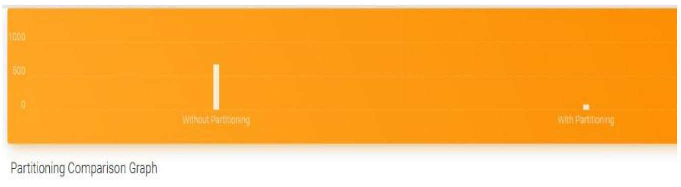


Fig - 6: Partitioning Comparison Graph

6. CONCLUSIONS

We propose an approach integrating frequent pattern clustering for finding an optimal database partition. The presented system of database partitioning will be more efficient to access for bulk database. It will work out the time complication of partitioned and without partition database.

FUTURE SCOPE:

With the of this proposed model we will be able to get data in less execution time, this research work has not taken the following aspects into account:

- Generation of only user interested association rule.
- Setting up of different minimum support values of various items present in the database.

REFERENCES

[1] Stratos Papadomanolakis, Anastassia Ailamaki, "Auto Part: Automating Schema Design for Large Scientific Databases Using Data Partitioning",16th International Conference on Scientific and Statistical Database Management (SSDBM).Santorini Island, Greece. June 21-23, 2004.

[2] Sanjay Agrawal,Vivek Narasayya,"Integrating Vertical and Horizontal Partitioning into Automated Physical Database Design",SIGMOD 2004, June 13–18, 2004, Paris, France.

[3] Yin- Fu Huang, Chen JuLai "Integrating frequent pattern clustering and branch- and bound approaches for data partitioning", Information Sciences328(2016)288–301