International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 11 Issue 2 –NOVEMBER 2014.

Data Warehousing Security Encapsulation with Bitmap Indexing Mechanisms

[1] Uma Pavan Kumar Kethavarapu and [2]Dr.B.Lakshma Reddy

[1] *Research Scholar, Pondicherry Engineering College, Associate Professor, AIMS Institutions, Bangalore*

[2]*Director-Computer Science, Garden City College of Science and Management, Bangalore*

Abstract— Data warehousing is an environment which will allows the various categories of the source data in the formats of ERP, XML, Relational model, Flat Files (.DOCS, Excel sheets, .PPT...)which is in the area of Online Transaction Processing(OLTP),later the data will be integrated into the uniform format and stored into a data warehousing, Data Marts. Finally the data will be processed to produce the results in the form of Reports which is known as Online Analytical Processing (OLAP).The ultimate goal of data warehousing is to produce strategic decisions by making analysis in the huge amount of the data. The research article deals with the preprocessing considerations in the handling of Bitmap indexing creation, which will help the processing of the data warehousing data in effective and efficient manner. The data warehousing data is huge in general to preserve data in secured manner we have to follow up some security aspects without compromising the faster processing of the data. We are proposing the Bitmap encryption methodologies in the processing of the data warehousing data with secured and faster rates. The default nature of bitmap indexing is encryption and faster processing of the data. The article covers various formats of the data indexing with Bitmaps and integration of slowly changing dimensions (SCD) types to capture only changed dimensions without getting the entire data. Aggregation of data is also allows the users to get the summation of data rather than processing the entire set of the data.

Keywords: Bitmap Indexing, SCD, Security, Aggregation, Data warehousing

I. INTRODUCTION

The data warehousing environment mainly provides Online Transaction processing and Online Analytical Processing along with this ETL activity is playing a vital role. Usually data warehousing environments are meant for handling bulk data such as tera bytes and pico bytes, and at least 5 to 10 years of data are managed. In the data processing the fastest querying is mandatory for various levels of users so as to handle the enterprise data. The best mechanism of faster data processing is Indexing mechanism, which is a way of handling the data in range wise by creating the indexing on the data. Out of existing Bitmap indexing is proven as best method of faster data processing. The data is observed and kind of encoding is chosen for establishing a bitmap and best bitmap categories are simple, encoded, enhanced encoded, compressed and tunable methods. The existing mechanisms of bitmap can be integrated with data mining techniques, Usage of Ice berg queries, implementation of soft set computing; will give more efficiency for the query processing. Slowly changing Dimensions (SCD) is a way of loading data into dimension tables with irregular, random and variable schedule.

II. BITMAP INDEXING IMPORTANCE

QL> create bitmap index normal_empno_bmx on test_normal (empno);
Index created.
Elapsed: 00:00:29.06
Statistics
29 recursive calls
0 db block gets
5 consistent gets
0 physical reads
0 redo size
515 bytes sent via SQL*Net to client
499 bytes received via SQL*Net from client
2 SQL*Net round trips to/from client
0 sorts (memory)
0 sorts (disk)
1 row processed

Consistent Reads	Physical Reads	EMPNO
5	0	1000
5	2	2398
5	2	8545
5	2	98008
5	2	85342
5	2	128444
5	2	858

SQL> create bitmap index random_empno_bmx on test_random(empno);

Index created.

SQL> select * from test_random where empno=&empno;
Enter value for empno: 1000
old 1: select * from test_random where empno=&empno
new 1: select * from test_random where empno=1000

Elapsed: 00:00:00.01

Statistics
---
0 recursive calls
0 db block gets
5 consistent gets
0 physical reads
0 redo size
515 bytes sent via SQL*Net to client
499 bytes received via SQL*Net from client
2 SQL*Net roundtrips to/from client
0 sorts (memory)
0 sorts (disk)
rows processed

Consistent Reads	Physical Reads	EMPNO
331	0	1-2300
285	0	8-1980
346	19	1850-4250
427	31	28888-31850
371	27	82900-85478
2157	149	984888-1000000

SQL> select * from test_random where empno between &range1 and &range2;
Enter value for range1: 1
Enter value for range2: 2300
old 1: select * from test_random where empno between &range1 and &range2
new 1: select * from test_random where empno between 1 and 2300

2300 rows selected.

Elapsed: 00:00:03.04

Execution Plan
---
0 SELECT STATEMENT Optimizer=CHOOSE (Cost=613 Card=2299 Bytes=78166)
1 0 TABLE ACCESS (FULL) OF 'TEST_RANDOM' (Cost=613 Card=2299 Bytes=78166)

Statistics
---
0 recursive calls
0 db block gets
6415 consistent gets
4910 physical reads
0 redo size
111416 bytes sent via SQL*Net to client
2182 bytes received via SQL*Net from client
155 SQL*Net roundtrips to/from client
0 sorts (memory)
0 sorts (disk)
2300 rows processed

The optimizer opted for a full table scan rather than using the index because of the clustering factor:

Consistent Reads	Physical Reads	EMPNO
2463	1200	1-2300
2114	31	8-1980
2572	1135	1850-4250
3173	1620	28888-31850
2762	1358	82900-85478
7254	3329	984888-1000000

For the last range (984888-1000000) only, the optimizer opted for a full table scan for the bitmap index, whereas for all ranges, it opted for a full table scan for the B-tree index. This disparity is due to the clustering factor: The optimizer does not consider the value of the clustering factor when generating execution plans using a bitmap index, whereas for a B-tree index, it does. In this scenario, the bitmap index performs more efficiently than the B-tree index.

III. ICE-BERG REQUIREMENT IN INDEXING

Iceberg query is a special class of aggregation query, which computes aggregate values above a given threshold. Because of the small result set, iceberg queries can potentially be answered quickly even over a very large data set. The existing query optimization techniques for processing ice berg queries are tuplescanbased approach, which requires at least one table scan to read data. Ferro designed iceberg queries with a two-level bitmap index which is suffering from the massive empty-bitwise AND results problem. Proposes an index-pruning-based approach to compute iceberg queries using bitmap indices.

Parameters a)Iceberg threshold b)Number Of Distinct Groups c)Number Of Distinct Values d)Attribute Length e)Number Of Aggregate Attributes f)Number Of Attributes

Observations: The Algorithm is not sensitive to the number of distinct values, Number of attributes, Length of Attributes, Performance is better when the query is more iceberg-like, Number of Aggregation attribute is relatively small. Bitmap indices have an advantage of leveraging the anti-monotone property of iceberg queries to enable aggressive index pruning strategies. Iceberg queries anti-monotone property: If the count of a group is below T, the count of any super group of it must be below T.To overcome the challenge of empty bitwise-AND vector alignment algorithm with dynamic pruning can be used.

Slowly Changing Dimensions (SCD) Usage

Slowly Changing Dimension (Kimball, 2008) is the name of a data management process that loads data into dimension tables which contains data. To adopt SCD, the data has to change slowly on an irregular, random and variable schedule. There are 6 current types of SCD methodologies, namely Type 0, Type 1, Type 2, Type 3, Type 4, and Type 6. The most commonly practiced SCD types are 1, 2 and 3. Below we present the descriptions of the different Types of SCD. Type 0 - Type 0 SCD does not update the changes in the data. Original values of the record remain in the dimension that was initially created. Type 1 - Type 1 SCD overwrites old records with new records. Type 1 SCD is easy to maintain. However, no historical observations are kept in the data warehouse. Type 2 – Type 2 SCD updates the record by inserting new observation while preserving the historical observations. Unlimited historical observations can be preserving with this type. Type 3 – Type 3 SCD updates the record by creating new dimensions to the table structure. It preserves limited history - only the previous record could be preserve. Type 4 – Type 4 SCD updates the record in the current data table and preserves all or some historical observations in an archive table. Type 6 – Type 6 SCD is a hybrid of the methodologies of Type 1, 2 and 3. It incorporates Type 3 SCD by creating additional dimensions to the table structure while preserving historical observation. It incorporated Type 2 SCD by creating new dimensions to include the different version number. Type 1 SCD is incorporated by updating the latest record in the observation.

IV. Our Contribution

The above mentioned aspects such as Bitmap indexing, iceberg queries and slowly changing dimensions are independent, we observed that the integration of all the above independent aspects will give the better performance to observe the kind of the data, categories of the data, amount of the data, whether the base consists of slowly changing dimensions through which we can identify subsets of the data so as to get in the faster way rather than processing the entire set of the data. The following are the observations and implementation suggestions towards the better processing of the bitmap indexing in huge amounts of the data bases. There is a requirement of index advisory tool with all the above mentioned concepts which will give the pre-processing of the data items for better analysis of their kind (Flat files, xml, ERP…) and amounts of data to be processed is there any involvement of SCD etc. Analysis of generated data so as to apply the suitable Bitmap indexing.

The analysis involves identification of slowly changing dimensions, patterns of the data, uncertain data and size of the data to be processed in memory, out memory processing.

In case of slowly changing dimensions usage of TYPE-0, TYPE-I, TYPE-II and TYPE-III TYPE-IV and TYPE-VI methods need to integrate with existing bitmap indexing.

In case of Pattern based data usage of clustering and association mining methods in the generation of bitmaps.

A well-defined encoding representation of each distinct value an indexed attribute is important to optimize other types of queries performance.

Applicability of Data mining techniques such as classification, clustering and association could be used to group values attribute that are frequently queried together.

Processing of uncertain data through soft set computing in the generation of bitmap indexing.

The data which is fit into in memory best strategy is enhanced bitmap indexing.

The data which is not fit into in memory process the bitmap indexing with No Sql and Big Data like Hadoop methods.

Estimation of statistics of read/unread data with prediction models and processes them for new requirements.

V. Conclusion and Future Work

The overall aim of this work is generation of the decision algorithm which involves the pre-processing of data sets through the Bitmap indexing strategy. The main benefit of doing so is load balancing, identification frequent patterns of the data sets, kind of data types available in the data bases, slowly changing dimension scenarios handling and usage of aggregation in the form of ice-berg querying. Future scope involves implementation of ant monotone property of iceberg querying through bitmap processing, handling of uncertain data in the data sets which is a big issue in the processing of huge amounts of the data. The inherent benefit of slowly changing dimensions and Ice berg querying is security considerations. Without getting the entire source data only aggregated data will be processed. data only aggregated data will be processed.

VI. References

[1] Alex berson, “Data warehousing Concepts”.

[2] Elizabeth O Neil, "Bitmap Index Design Choices and Their Performance Implications", LBNL-62756, 2013.
[3] Md.Al Mamun,"Performance Improvement Techniques for Customized Data warehouse", IOSR-JCE, Mar-Apr.2013.
[4] Firdous Kausar, "Comparative Analysis of Bitmap Indexing Techniques in Data warehouse", IJETAE-Vol 4, Issue 6, June 2014.
[5] P.Niranjan,"A Fast Retrieval of Software Reusable Components Using Bitmap Index", IJCST vol 3, issue 4, Oct-Dec 2012.
[6] Shivam Dwivedi, "Evaluation of Bitmap Index Compression Using Data Pump in Oracle Data base", IOSR-JCE, Vol-16, Issue-3, Ver.III (May-Jun 2014).
[7] Hyoung Geun,"A study on the selection of Bitmap Join Index Using Data Mining Techniques", 2013 IEEE
[8] Bin He, "Efficient Iceberg Query Evaluation Using Compressed Bitmap Index", IEEE Transactions on Knowledge and data engineering, vol 24, No 9, Sept 2012.
[9] www.kimbal.com.
[10] http://www.oracle.com/technetwork/articles/sharma-indexes-093638.html
[11] Ralph Kimball (2008, September 22). Slowly Changing Dimensions, Part 2. Kimball Group. January 20 2013 from http://www.kimballgroup.com/2008/09/22/slowly-changing-dimensions-part-2/print/
[12] Margy Ross (2008, September 29). Practical Steps for Designing a Dimensional Model. Kimball Group. January 20, 2013 from http://www.kimballgroup.com/2008/09/29/practical-steps-for-designing-adimensional-model/print/
[13] Shauna Trainor (2013, February 13). Market In Modesty To Prevent Resentment. Covenant Group. February 13,2013 from http://www.covenantgroup.com/blog/market-in-modesty-to-prevent-resentment/