The ISC Int'l Journal of Information Security

ISeCure

November 2020, Volume 12, Number 3 (pp. 81–85)

http://www.isecure-journal.org

Selected Paper at the ICCMIT'20 in Athens, Greece

Indexed View Technique to Speed-up Data Warehouse Query Processing**

Talib M. J. Al Taleb [1,*], Sami Hasan [1], and Yaqoob Yousif Mahdi [1]

[1] *College of Information Engineering, Al-Nahrain University, Baghdad, Iraq.*

ARTICLE INFO.

Keywords:
Data Warehouse (Star Schema), Indexed View, Query Processing

Type: Research Article

doi: 10.22042/isecure.2021.271059.615

ABSTRACT

The data warehouse size and the query complexity may cause unacceptable delay in decision support queries. A basic condition for the success of a data warehouse is the capability to supply decision makers with both precise information and best response time. For this purpose, the concept of indexed views is used. Indexed views help to speed-up query processing and reduce the response time for tracing queries, especially for queries about past histories.

1 Introduction

The goal of a data warehouse is to establish a data repository collects data from multiple data sources and is supposed to provide storage, functionality and responsiveness to queries beyond the capabilities of databases [1, 2]. The data warehouse has both a very detailed level of data and summarize data and it has ability to deal with current and historical data and creating analytical reports and optimizes query performance. At this paper, the data warehouse is designed for the healthcare centres [3, 4]. Star is the simplest type of schema, as shown in Figure 1 which contains fact table (Fact_Table_OPHC) sits in the center and associated with other dimension tables like a star [5, 6]. The significant volume of the data warehouses involves an increase of the response time that leads to appear the necessity for using optimization techniques to reduce. One of the most important techniques used is the indexed view which efficiently allows users to query huge amounts of data much more quickly than they could access from the base table. However, the more use indexed views lead to the more storage space is needed. Therefore, effective selection of indexed views should properly satisfy the factors of response time and storage space.

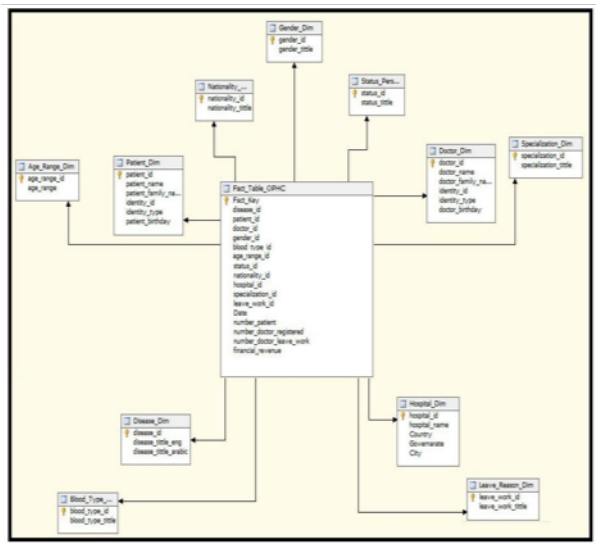

Figure 1. Star Schema.

* Corresponding author.

**The ICCMIT'20 program committee effort is highly acknowledged for reviewing this paper.

Email addresses: 1talib_altalib@yahoo.com, sami.hasan@coie.nahrainuniv.edu.iq, 3yakob.alz_2006@yahoo.com

ISSN: 2008-2045

2 Adding an Indexed View

Create view

- Creating a unique clustered indext.
- Creating additional indexes (non-clustered index).
  At first, this article is applicable to data warehouse designed by using SQL server. It is not applicable if the data warehouse was designed by using another software like Oracle, DB2 and Informix. For many years, Microsoft SQL server has supported the ability to create virtual tables known as views.

Views are a powerful tool in SQL server and helpful in many ways which are [7, 8]:

- Help to provide a layer of additional security that restricts users to a certain subset of data in one or more base tables.
- Help to provide a mechanism that allows developers to customize how users can logically view the data stored in base tables.
- Help to write queries faster and more efficiently, but they don't improve the underlying query performance. Thus, indexed views are designed to address this shortcoming.

When creating a view, adding a stored query to the database that can be read in the same way as a table. The view acting as a virtual table that does not require any additional storage space, as the information is read directly from the tables. With SQL server, the functionality of SQL server view was expanded to provide system performance benefits. Indexed view can be a powerful tool that can satisfy a query. Then, under certain circumstances, this can significantly reduce the amount of work that SQL server needs to do to return the required data, and so improves query performance and solves the problem of decision support workloads. Applications that benefit from the implementation of indexed views include:

a. Decision support workloads
b. Data marts
c. warehouse
d. OLAP
e. mining workloads

On the contrary, OLTP system may not be able to take advantage of indexed views due to of the increased maintenance cost associated with updating both the view and underlying base tables. Frequent updates will kill the benefit. Another cost of indexed view is that the data is actually stored. By applying the clustered index, a copy of the data is created. So if several indexed views have on a single table, this will have several copies of the data, along with the statistics of the data and the indexes and all of the overhead that goes with it. The indexed view may not provide any significant performance gains if its size is similar to the size of the original table. However, when using indexed views, they can offer significant improvements to access the data. Caution is required when implementing an indexed view unless the base tables are relatively static.

3 B-Tree

When the first index is added to a view, it is built using a B-tree structure, see Figure 2 . In the B-tree structure, there are branches from the top that lead to leaf level that includes the data from the columns that are present in the index. A B-tree structure enables SQL server to find the row or rows associated with the key values quickly and efficiently. This can lead to large increases in the storage requirements for a database but improves the view's performance [9].
The balanced scorecard (BSC) was initiated in the United States, at the beginning of the 1990, by the authors KAPLAN and NORTON, in order to have the organizations with a management tool adapted to the situation of the current environment.
The BSC is a strategic performance management system that links performance to strategy using a multidimensional set of performance, financial and non-financial indicators. It aims at a better understanding of the causal links within organizations and the levers that can be used to improve corporate governance.
Its particularity lies in the fact that these indicators relate to four specific areas that are interrelated. These are the financial areas, customers, internal processes, learning and growth. These four areas must be taken into account in the management system to achieve the desired level of performance.

4 Clustered Index

There can be only one clustered index per table, because the data rows themselves can be sorted in only one order, more than one clustered index on one table is impossible [10, 11].

1. Usually depends on the primary key (PK).
2. Sort the records and store them physically on the disk according to the order.
3. When a table has a clustered index, the table is called a clustered table. If a table has no clustered index, its data rows are stored in an unordered structure called a heap.
4. A clustered index means you are telling the database to store close values actually close to

one another on the disk. This has the benefit of rapid scan/retrieval of records falling into some range of clustered index values.
5. Data retrieval is faster than non-clustered indexes.
6. Do not need extra space to store logical structure.
7. Clustered indexes don't need to store a pointer to the actual row because of the fact that the rows in the table are stored on disk in the same exact order as the clustered index.
8. Clustered index determines the storage order of rows in the table, and hence does not require additional disk space.

5 Non-clustered Index

According to [10, 11]:

1. There can be any number of non-clustered indexes in a table because they don't affect the order in which the rows are stored on disk like clustered indexes.
2. Usually made on any key.
3. Non-clustered index contains the non-clustered index key values and each key value entry has a pointer to the data row that contains the key value.
4. Do not affect the physical order. Create a logical order for data rows and use pointers to physical data files.
5. The pointer from an index row in a non-clustered index to a data row is called a row locator. The structure of the row locator depends on whether the data pages are stored in a heap or a clustered table. For a heap, a row locator is a pointer to the row. For a clustered table, the row locator is the clustered index key.
6. Use extra space to store logical structure.
7. Non-clustered index is stored separately from the table; therefore, additional storage space is required.

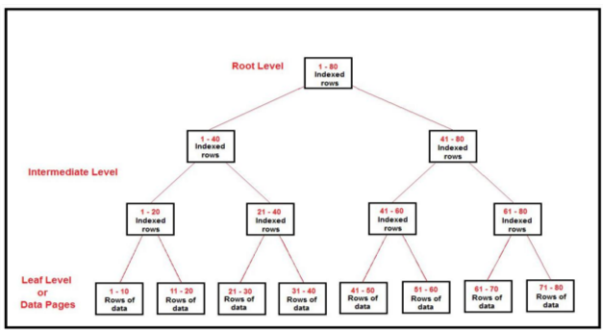

Figure 2. B-Tree

6 Limitations of Indexed View

1. Indexed views must be created using the SCHEMABINDING option. This option links the view to the schema elements that it uses, disallowing changes to the underlying tables that would break the view.
2. Indexed views may only select columns from base tables. They may not retrieve values from other views. The tables must be in the same database as the view and must have the same owner.
3. Indexed views may include inner joins but outer joins are not permitted.
4. Any functions used in the view's query must be deterministic. (Consistently providing the same result given the same input).

7 Indexed View vs. Materialized View

Oracle supports database object called materialized view that contains the results of a query. They can be local copies of remote tables, or can be used to create summary tables for aggregating data. SQL server has indexed view that can provide similar functionalities as a materialized view. But there are some differences between them:

1. Referencing underlying database object(s) materialized view which can refer to any base table, view or another materialized view, SQL indexed view can only refer to base tables and the base table must be in the same database as the indexed view.
2. Query definition indexed view does not support query statement with SELECT * syntax to specify columns of the table. Column names must be explicitly stated. Also, the SELECT statement cannot contain (COUNT, AVG, MAX, MIN, STDEV, STDEVP, VAR, or VARP aggregate functions).
3. Index creation: in order to create an indexed view, a unique clustered must first be created. Additional non-clustered index can be created. If the index is dropped, the stored result set is removed and the view is processed as a normal view.

8 Design an Indexed View for the Data Warehouse System

At first, view is created to meet the user's requirements for some important query. Once the view has been created, a unique clustered index is created on it. The first index created on view must be a unique index and must be clustered and this unique clustered index must be created before any other indexes can be created on the view. Once a unique clustered index

is present for a view, further non-clustered indexes may be created. Non-clustered indexes on views can provide additional query performance and provide more options for the query optimizer to choose from during the compilation process. An identity single column is created on the fact table (Fact_Table_OPHC) and made it a primary key (PK) called column fact key. The alternative to a single column PK is to have multi-column PK. The PK is created from a combination of dimension key columns. Usually, it is not all dimension key columns, but only several of them. Only those whose combination enables us to uniquely identify each fact row. An identity single column PK is better than multi-column PK because the multi-column PK may not be unique. Even if the combination of those columns is unique now, it does not guarantee that in the future the primary key won't be duplicated. The single-column approach guarantees that the PK is unique, because it is an identity column and this is the first reason for creating fact key. The second reason for creating fact key is to improve insert performance because in case of multi-column PK, SQL server needs to identify first where to insert that row, based on the clustered column. That row does not automatically go to the end of the table; it goes in the middle of the table depending on the value of the clustered column. SQL server spends a lot of time identifying where to insert each row. The data type of fact key is a BIGINT. In data warehouse, the primary consideration is the query performance and secondary consideration is the load performance. Single column PK is better when it needs to balance between loading and query performance but multi-column PK is slower in loading and faster query than single column PK. A common way of indexing a fact table is by putting a clustered index on the fact key column, then putting non-clustered index on each of the dimension key column. This will allow each of the non-clustered index to be efficiently used, whilst ensuring that they are slim because in SQL server, a non-clustered index uses the clustered index key as the row locator. They are efficiently used because they are built on each of the dimension key. Each of them as the first index key, is not as a second or third index key. This ensures that they will be hit by the queries using that dimension key because they are the first column in the index key. If the updated row requires additional space, then over time and frequent updates can cause poor performance requiring maintenance to rebuild indexed views. The fact table (Fact_Table_OPHC) does not have a huge data, and it contains 103401 records only so that the results of test are in milliseconds therefore the load on system is increased by using SQLQueryStress tool in order to obtain results in minutes or in seconds. An indexed view is created for diseases to improve the performance of the main queries that decision makers were constantly asking, and its performance has tested by SQLQueryStress. This indexed view shows patients' number for each disease and both gender and various ages, in various regions and times. It consists of:

Table 1. Compares between regular view and indexed view of Disease_vw

Function	Regular view	Indexed view (Clustered Index (Fact_Key) only)	Indexed view (Clustered index (Fact_Key) with (Non-clustered Index (disease_id))
Elapsed Time	00:04:35.7427	00:03:45.2783	00:03:05.1483
Client Seconds /Iteration (AVG)	3.5339	2.6012	1.3110
Logical Reads/Iteration (AVG)	1283	222	222
CPU Seconds/Iteration (AVG)	0.0436	0.0344	0.0342
Actual Seconds/Iteration (AVG)	3.2672	2.4011	1.2163
Current space	Default	1.734 MB	2.609 MB

a. View (Fact_Key, disease_id, gender_id, hospital_id, age_range_id, Date, number_patient)
b. Clustered index (Fact_Table_OPHC (Fact_Key))
c. Non-clustered index (Fact_Table_OPHC (disease_id))

Each index (clustered or non-clustered) occupies from storage about 2.609 MB where row count = 36633 records related to diseases and the size increases with the increasing number of records. Table 1 compares between the performance of regular view and the indexed view of disease (Disease_vw) where default query and row count= 36633, number of threads=200, number of iterations=50. This table shows that performance has improved after adding a clustered index and performance has improved further when adding a non-clustered index, but the size of indexed view increases with each addition. Date is very important and is used in most queries, but it is not selected as non-clustered index in the indexed view of disease because the records are inserted regularly in the fact table (Fact_Table_OPHC). In case the date is selected as nonclustered index there will be a slight improvement in performance or it may decrease performance. Note the elapsed time can be affected by several factors such as load on the server, input/output load, network bit rates between server and client.

9 Conclusion

Indexed views may significantly be reducing the amount of work that SQL server to return the re-

quired data and improve query performance and solve the problem of decision support workloads.

References

[1] Ralph Kimball, Margy Ross, Warren Thornthwaite, Joy Mundy, and Bob Becker. *The data warehouse lifecycle toolkit*. John Wiley & Sons, 2008.

[2] Paulraj Ponniah. *Data warehousing fundamentals: a comprehensive guide for IT professionals*. John Wiley & Sons, 2004.

[3] Lekha Narra, Tony Sahama, and Peta Stapleton. Clinical data warehousing: A business analytics approach for managing health data. In *Proceedings of the 8th Australasian Workshop on Health Informatics and Knowledge Management [Conferences in Research and Practice in Information Technology (CRPIT), Volume 164]*, pages 101–104. Australian Computer Society, 2015.

[4] Osama El-Sayed Sheta and Ahmed Nour Eldeen. Building a health care data warehouse for cancer diseases. *arXiv preprint arXiv:1211.4371*, 2012.

[5] A Patel and J Patel. Data modeling techniques for data warehouse. *International Journal of Multidisciplinary Research*, 2(2):240–246, 2012.

[6] Ross Mistry and Stacia Misner. *Introducing Microsoft SQL Server 2014*. Microsoft Press, 2014.

[7] Andrew Brust and Leonard G Lobel. *Programming Microsoft SQL Server 2012*. Pearson Education, 2012.

[8] Goetz Graefe. Sorting and indexing with partitioned b-trees. In *CIDR*, volume 3, pages 5–8, 2003.

[9] Jason Strate and Ted Krueger. *Expert performance indexing for SQL server 2012*. Apress, 2012.

[10] Itzik Ben-Gan, Lubor Kollar, Dejan Sarka, and Steve Kass. *Inside Microsoft SQL Server 2008 T-SQL Querying: T-SQL Querying*. Microsoft Press, 2009.

[11] Gang Gou, Jeffrey Xu Yu, and Hongjun Lu. A/sup*/search: an efficient and flexible approach to materialized view selection. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 36(3):411–425, 2006.

Talib M. J. Al Taleb is a professor at College of Information Engineering, Al-Nahrain University, Baghdad, Iraq. His research interests are developing application of data warehousing and information systems.

Sami Hasan received the B.Sc. degree in control and systems engineering and the M.Sc. degree in control and instrumentation engineering from the University of Technology, Baghdad, Iraq. His Ph.D. degree in computer engineering has been granted from Newcastle University, United Kingdom, in 2013, for his research project; "Parallel Implementation of Multidimensional Filtering Systems". His research projects have been focused on developing architecture-based AI/ML/DL, DSP, parallel algorithms and parallel reconfigurable hardware. Accordingly, more than fifty works have been published as conference papers, journal papers, book chapter and books. Consequently, his research works have been granted international and local awards; the British NISA award, the Ministry of higher Education and scientific Research award and Al-Nahrain University award for the outstanding research works.

Yaqoob Yousif Mahdi received the B.Sc. degree in information and communication engineering from University of Baghdad, Baghdad, Iraq. He received the M.Sc. degree in information and communication engineering from Al-Nahrain University, Baghdad, Iraq. Currently, he is a Ph.D. candidate in networking engineering at UPC Universitat Politcnica de Catalunya, Barcelona, Spain. His research interests include big data, VANETs, AI & ML, federated learning, V2V, V2I, 6G & 5G, edge & fog computing, data warehouse, data bases, data analysis and OLAP.