Next Generation Data Warehouse Design with OLTP and OLAP Systems Sharing same Database

S. Santhosh Baboo, PhD.
Reader, P.G. and Research
Dept. of Computer Science
D.G.Vaishnav College Chennai
600106

,

P Renjith Kumar
Research scholar,
Computer Science,
Manonmaniam Sundaranar
University, Tirunelveli, 627012
SAP Business Intelligence
Consultant, SAP Labs India

ABSTRACT
Online Transaction Processing (OLTP) systems have been using the traditional relational Database management system for many years. Online Analytical Processing (OLAP) system for analytical reporting is optimized to aggregate many records which involve more read operations. Hence to enhance the performance in OLAP systems various modeling options like star schema, extended star schema was implemented. But with the latest innovations in hardware technologies like availability of multi core dual processors, availability of 64 bit processors, decreasing cost of main memory and software technology innovations like In-Memory computing, Columnar storage and other with various other new features now there is less concern towards the performance degradation and performance optimization in a data warehouse system. If there is an option to use same database for the OLTP and OLAP systems then the business intelligence system can get the real time data for the analytical query reporting. This document presents the overview of the next generation data warehouse architecture which shares the database along with OLTP and discuss how it is going to use the in memory capability to apply real time analytics.

Keywords
In-Memory analytics, columnar data storage, Next generation data warehouse, Business intelligence, Real time data analysis, Row store

1. INTRODUCTION
With the use of RDBMS every business has enormous amount of data in their database systems, to utilize the power of data traditional data warehouse system were designed which involves the extraction of huge volume of historical data from various On Line Transaction (OLTP) systems across various complex system landscapes. Generally the traditional Data warehouse systems extract's data from various OLTP source system using Extraction Transformation and Loading (ETL) tools at specified time interval like per day delta load or per hour load basis. But the business trends are frequently changing, customer buying habits are constantly changing there is a need for almost real time data which could help to better analysis and to assist in making better decisions which helps organization to be best running business. The concept of extracting the data from an OLTP system to OLAP system in various intervals seems to provide data with delay of day or even less. But business is changing rapidly to adapt to the changing market there in need for real time analysis for our analytical system, in today's market there is a need to analyse the business as it happens. Even an hour delay of the data for the OLAP system will have impact on the overall decision making there by creating a setback in the constantly changing business trends. Now it creates a question

The question is
"Will it be possible to design a Business intelligence system where the OLTP and OLAP will be in one system together which enable us to take the real time business data and use in-memory technology for accessing real time business data directly from the OLTP source table with interactive reporting in OLAP layer which helps organizations in taking time critical business decision to adopt to the changing business trends and use the power of business data to its fullest potential"

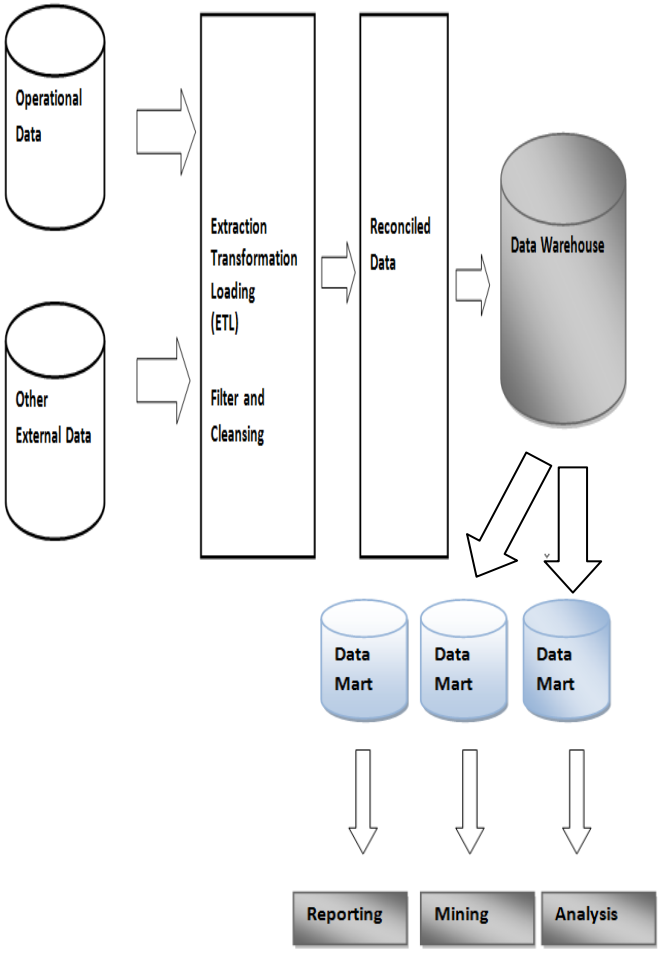
If it is possible to achieve this type of data warehouse system, it will be a step towards next generation data warehouse design and this will enable business and users to react to business events more quickly through real –time analysis and reporting of operational data and helps management to support the deployment of innovative new business decisions in real time.

2. ANALYTICAL AND TRANSACTION DATA
An analytical application typically processes large chunks of historical data. An analytical application might highlight segments of data, make comparisons between different parts of the data, produce visual representations of the data, or predict trends based on past data, but makes very limited, if any, changes to the database or data set. Analytical applications generally use OLAP (Online Analytical Processing), which is optimized to aggregate many records, and, when dealing with structured data, they run faster on databases that present the data in columns, quickly querying large volumes of data from the pertinent columns only.
A transactional application is used to record or update data. Transactional applications typically query a small amount of data at a time alters the data. For example, when someone buys something, the record of the purchase will be added into the company's database. Transactional applications generally use OLTP (Online Transactional Processing), which is optimized to retrieve a small number of records at a time, and includes measures to prevent update conflicts. Databases that present data in rows are by nature suited transactional applications, which deal with adding, editing, or viewing single records or rows at a time. Both systems, OLTP and OLAP, are based on the relational theory but using different technical approaches [1]

For OLTP systems, tuples are arranged in rows which are stored in blocks. The blocks reside on disk and are cached in the main memory within in the database server. Sophisticated indexing allows fast access to single tuples; however access get increasingly slower as the number of requested tuples increases. For OLAP systems, in contrast, data is often organized in star schemas, where a popular optimization is to compress attributes (columns) with the help of dictionaries. after the conversion of attributes into integers, processing becomes faster. [2]

3. CURRENT DATA WAREHOUSE SYSTEM ARHITECTURE

Fig 1: Data warehouse architecture



The traditional data warehouse system gets data from multiple source system; this is called as OLTP system. The data that is sent from source system is raw in format, hence when multiple systems are connected to a single data warehouse system there should be a predefined extraction and transformation process. There is need to define the filer and cleaning criteria's. The data from source system will be detailed in nature, for an analytical reporting purpose this data have to be aggregated in the data warehouse system and then transform the data from OLTP system into a summarized data in the data warehouse system. OLAP cubes stored the multidimensional view of summarised data which can be used for analytical reporting purpose. When an OLAP cube contain huge volume of historical data, the query that reads that data from the OLAP cube will have performance issues, hence to optimize the analytical reporting query performance we can create many data marts with predefined selections based on the OLAP cube. Data mart is again a physical cube but contains only a subset of data that is required for reporting. Hence when a query is executed it will first check if there is any associated data mart for the selections if it is available the data is fetched from there, else the base OLAP cube is read further.

4. CHALLENGES IN CURRENT DATAWAREHOUSE

The big problem that lies in front of the data warehouse field is it normally have huge amount of data and it is still getting even bigger. Every business creates massive data and the success of each business depends on how those data is used effectively. There is semantic data from various sources which could also be analysed. Social networking sites like Facebook and twitter contains huge amount of data that provides potential information about the changing customer trends and behaviour. The retail industry has huge volume of transaction data per day. When there are millions of transactions happening per day in a retail industry OLTP database it becomes extremely difficult to bring data to data warehouse system each day and do an analytical reporting based on the data. But the current business trends suggest even the delay of data per day can create a huge impact to the business. Business need to do analytical reporting as the business happens in real time, it will help us to adapt to the changing business market in real time. There is a need for real time business intelligence solution that provides analytical reporting on real time data. Handling of the real time data is big challenge that a future business intelligence systems will face. There is a need for more dedicated hardware resource to store and fetch data at real time. Much business needs instant access to any information that can help to take business decisions. Instant access means in the moment access, business may not take the right business decision if the data is not available at that moment. Hence gone are the days where data warehouse if used to store historical data, now there is a need for a data warehouse that provide instant data and help companies to access that instant data and take dynamic decision based on the real time instant data. All these factors make a huge necessity to design a new database.

Need for new Database architecture

[6] With the new innovations in the hardware and enterprise applications it is high time to check for a new Database that suits both OLTP and OLAP scenarios. It should use all the available technology advances and new data management redesign needed to handle changes in

- Hardware trends (CPU/cache/memory)
- Changed workloads
- Data characteristics
- Data amount
- Reporting trends and requirements

When saying about the changes in hardware, main memory is becoming cheaper and growing larger along with it can be seen that now there are several advance in enterprise software for processing data which include column-oriented data organization and various other compression methods The new data management model should try to utilize the complete potential of the changed hardware features and it can use the latest software's which are capable of utilizing the advanced hardware to optimal level. When we say about data warehousing applications then we need to handle very

complex enterprise applications like inventory management in a powerful way, hence our focus should also be in handling the business or enterprise applications effectively.
In any general computing system as we are aware that the CPU mostly waits for RAM [7]

- In General the traditional system stores the data in the disk drives and for computing or analysis accessing the data from Disk will have more impact on performance.
- When compare to Disk the access from Random Access Memory (RAM) is very less. The performance in this scenario will be improved as there is no need to fetch data from disk drives.
- When there is an option available to store the frequently required data in a local cache, it will be first place to be hit when there is a search for data and then it will be better than accessing from Random Access Memory (RAM).
- Making all data available for reporting or analysis in RAM will be the future architecture with the changing customer trends and need for high performance applications being the core drivers.
- Main Memory DB is going to be common in future which will help application to improve the performance in complex analytical applications and help to do real time analytics.

5. ROW AND COLUMN BASED DATA STORAGE.

Row-store:
Rows are stored consecutively in the memory, It is very much useful for update intensive applications like On Line transaction Processing (OLTP), It is Optimal for row-wise access (e.g. Select *, Select Single *)
Row store is recommended if: [5]

- The table has a small number of rows, such as configuration tables. Usually the configuration tables are required more in numbers in case of ERP systems. There is need for configuration tables for storing many configuration based data such as currencies, quantities and other information.
- Row store is useful in cases where the application needs to process only a single record at a time (many selects or updates of single records).
- In Enterprise applications there is always a need to fetch the complete set of record like complete details of material master which means the application should extract the many fields from the row. Hence row store is used if the application typically needs to access the complete record.
- Compression will be helpful if there is more unique values in the columns, a compression on this scenario will be helpful. But when the columns contain mainly distinct values so the compression rate would be low. If the compression rate is low then the option is to go for row based storage.
- Online Transaction Processing system mainly requires updates hence there will be less need for aggregation. Also when fetching the records from a database table usually few set of transaction records with the limited time frame is required in can of OLTP systems and there is no need for fast and quick searching. Hence If Aggregations and fast searching are not required then row store can be preferred.

Column-store:
Columns are stored consecutively in the memory, It is very much useful in the read insensitive applications like On Line Analytical Processing (OLAP) , Hence this is Optimal for attribute focused access (e.g. SUM, GROUP BY) [3]
Column store is recommended if: [5]

- In analytical applications there will be necessity to fetch or project any single column based on any descriptive value. In these scenarios there will be more read on a single columns. So when the calculations are executed on a single column or a few columns only column store is highly recommended. This is the reason OLAP system prefer column storage.
- If any application has many tables and if there is a requirement that the table is searched based on the values of only a few columns then best the option recommended would be to go for column based store.
- There may be scenarios in data modeling where the table will have more columns than rows. In this case of table having more columns is a preferred model to go for column store. Maximum performance gain can be achieved in this scenario.
- In any application when there are large number of rows it is recommended to go for row based storage, but if the application requires any operation on columns like aggregating the values in the column or there is need to scan the complete column, these scenarios make ideal to go for column store.
- When there are many rows and columns in any application. Check the distinct values in the columns that a table has. If The majority of columns contain only a few distinct values (compared to the number of rows), resulting in higher compression rates then the best option would be to go for column based storage..

The figure given gives a detailed look on how a table is stored in a row store and column store in memory and it is analyzed further.

Table: Employee

	Employee ID	Name	Country
Row 1	E1001	Kumar	US
Row 2	E1002	Mani Vannan	India
Row 3	E1003	Renjith	India
Row 4	E1004	Ram	India

Fig 2: Storage in Row and Column oriented Database

Organized by Row in memory

E1001	Kumar	US	E10002	Mani Vannan	India	…

Organized by Column in memory

E1001	E20002	E10003	E10004	Kumar	Mani	…

With the above storage it is clear that when there is a read based on a single column columnar Database take the advantage as the columns are stored together.

6. COLUMN STORE FOR OLTP SYSTEM

Modern CPUs with multi-core architecture provide an enormous amount of computing power. Blades with 8 CPU's and 16 cores per CPU will populate next-generation blade servers. That gives about 128 computing units with up to approximately 500 GB of main memory [2]. To optimize the use of these computing devices there is a need to understand memory hierarchies, cache sizes, and how to enable parallel processing within one program [4]

Below are the advantages of column store DB for OLTP system for application development

Compression [5]

When there are many unique values in ant table then the compression will be effective. In general columnar data storage allows highly efficient compression. Especially if the column is sorted, there will be ranges of the same values in contiguous memory, so compression on these types of scenarios will be highly effective. Hence compression various methods such as run-length coding or cluster coding can be used.

Studies suggest that in column stores a compression factor of 10 can be achieved compared to traditional row-oriented database systems. This can increase the speed because of below reason.

- Loading of data in CPU cache is one of important factor. Compressed data can be loaded into CPU cache more quickly. As the limiting factor is the data transport between memory and CPU cache, the performance gain will exceed the additional computing time needed for decompression.

- With dictionary coding, the columns are stored as sequences of bit-coded integers. Checks for equality can be executed on the integers (for example, during scans or join operations). This is much faster than comparing string values.

- When an analytical application is there need for scan and aggregation is more. Hence compression can speed up operations such as scans and aggregations if the operator is aware of the compression. Given a good compression rate, computing the sum of the values in a column will be much faster if the column is run length coded and many additions of the same value can be replaced by a single multiplication.

Less cache is needed

Reading only the necessary columns exhibits a more cache-friendly I/O pattern, for both on-disk and in-memory column store databases. In the case of an on-disk column store, few disk seeks are needed to locate the page of a block that has the first field of a particular column. From there, data can be read in large blocks. In the case of an in-memory column store, sequentially laid out columns typically ensure a good L2 cache hit ratio, since modern DRAM controllers use pre-fetching mechanisms which exploit spatial locality [8] also there is no need for maintaining aggregates

Minimized index maintenance *[5]*

To increase the performance in read operation in the OLTP system the normal practice is to create secondary index on the field that is read. But the creation and maintenance of index is expensive in terms of database. Hence when there is huge volume of records in any database tables in application's like retail and inventory management scenario's there is need for creation of secondary index on the base tables.

In these cases columnar storage eliminates the need for additional index structures in many cases. Storing data in columns works like having a built-in index for each column: the column scanning speed of the in-memory column store and the compression mechanisms, especially dictionary compression, already allow read operations with very high performance. In many cases, additional index structures will not be required. Eliminating additional indexes reduces complexity and eliminates effort for defining and maintaining metadata

Current Challenges in updates with Column store

In any transaction processing system there will be many updates each day like sales order, purchase order, production order etc. The database table is inserted for each transaction. The requirement for enterprise in transaction processing system is more focused towards effective update and there are less reporting requirements. Hence update insensitive applications generally prefer row store database. This will be big challenge when having a common database approach for OLTP and OLAP system.

Generally the updates in a column oriented database required more disk I/O than row oriented database. But if this update can be handled effectively then the implementation of same database for OLTP and OLAP becomes feasible.

6.1 A Study on C-store

C-Store is a column oriented It is an analytical columnar database system[9] and it, handles updates by splitting their architecture in a "read-store" that manages the bulk of all data and a "write-store" that manages updates that have been made recently. Consequently, all queries access both the base table information from the read-store, as well as all corresponding differences from the write-store and merge this on-the-fly. Also, in order to keep the write-store small (it resides typically in RAM), changes in it are periodically propagated into the read-store [9] [10]

Some other prototype column-oriented in-memory uses different approach to handle this update by using differential buffers.

6.2 A Study on HYRISE

HYRISE is a prototypical column-oriented, in-memory database prototype used to empirically validate the various findings by Hasso Plattner Institute for IT Systems engineering [10] It has the below architecture.

The storage manager maintains the physically stored data in main memory and provides access methods for accessing data while organizing data along columns with applied dictionary compression.

Consequently, all modifications are handled by a dedicated differential buffer for each table to postpone

Re-compression cost to later point of time to distribute the re-compression cost over all data modifications stored in the buffer [10]

Merge process have the following phases

- Prepare Merge
- Attribute Merge
- Commit Merge

The prepare merge phase locks the differential buffer and main storage and creates a new empty differential buffer storage for all new inserts, updates, and deletes which occur during the merge process.

In the attribute merge phase the following steps are executed for each attribute: The first step is merging the dictionaries of the differential buffer and main storage.

Next, the value ids of main storage and write buffer are copied to a new main storage - thereby changes in the dictionary have to be applied to the new value ids; invalidated values of the original main storage are not copied and can be transferred to a history log. [10]

The commit merge phase starts by acquiring a write lock of the table. This ensures that all running queries are finished prior to the switch to the new main storage including the updated value IDs. [10]

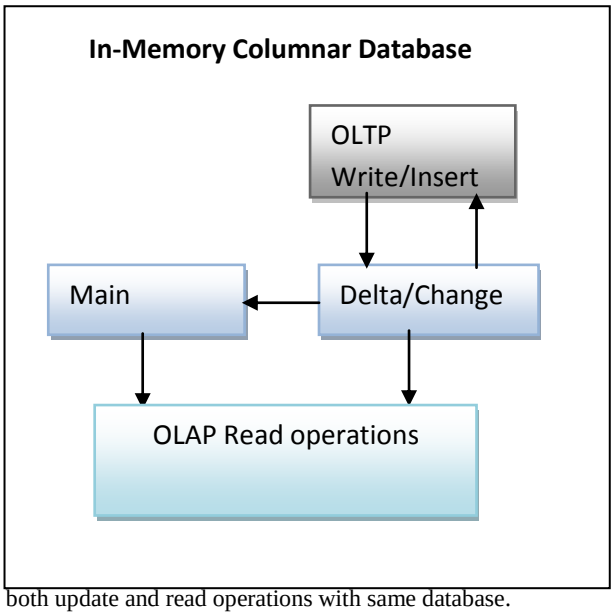
By this method the updates are handled effectively in the HYRISE prototype column-oriented in-memory database.

7. THE PROPOSED ARCHITECTURE

Having a common database or hybrid Database for the OLTP and OLAP systems is the goal of this paper. OLTP systems should be able to do transactions and OLAP system should be able to do effective analysis from the same database but these operations should not be interfered. When there is an In-Memory database it should be handled effectively. With In-Memory the IO time is reduced.

There is an In-Memory Columnar Database OLTP systems involves many writes to the Database layer and generally write in a columnar Database id expensive. The OLTP system uses transaction with full Database level. An transaction system always have changes called delta and it is complex to handle, hence the changes or delta is captured in a delta layer of the database to avoid the huge expense caused by the Insert operation in any columnar Database. Now the changes or deltas will be stored in the delta layer. But from analytical view both the data is needed. The delta is then transferred to the columnar main read part of the In Memory Database. The OLAP sessions will read the data from both the Delta and other layer of the In-Memory database.

Fig 3: Proposed model for common Database



both update and read operations with same database.

8. DISTINCTIVE FEATURES

This part of the paper tries to analyze the distinctive features that are available in the proposed system when compared the systems and methods available at present. This does not mean to discount any available methods or systems.

8.1 C-Store and proposed system

C-store is a column oriented analytical columnar database. It manages the updated by splitting the architecture by splitting the architecture into write-store and read-store. The updates are handled in the write-store. But when a query is executed both the read-store an write-store is accessed together. In the proposed system the OLTP updates are save in the delta/change layer and it is sent to the main layer. But when there is an analytical query the data is read from both the layer. The key part is the proposed system is In-Memory data base system.

8.2 HYRISE and proposed system

HYRISE is a prototype columnar In-Memory database used by Hasso Plattner Institute for IT systems and engineering. In HYROSE the dictionary modifications are handled by differential buffers. It uses merge process to make the data available in the Main storage. Multiple actions like prepare and commit in the merge phase which makes the update to main storage possible. In the proposed system there is no concept of differential buffer and multiple phases in merge. The overall logic is to send the delta changes to main layer and the analytical query is designed to access both the main and delta layer.

Based on the above comparison it can be found that the concept of writing the OLTP updates to a different layer optimized for insert operations and moving the changes and updates to an main layer where the OLAP queries can be accessed and availability of both layer for OLAO access along with these keeping all these layers within the Memory is the unique features in the proposed design.

9. REAL TIME ANALYTICS

Today's Business requires analysis of data as the business runs. If a business can be able to identify the customer buying habit when the customer buy in real time then it can offer best deal to them instantly and make more sales in retail industry, but there is need for real time data for analytical purpose. This can be achieved when organizations have business intelligence applications that are capable to providing real time analytics. When the OLTP and OLAP share a common database the business intelligence system get the data in real time and it will be possible to build analytical queries based on the data.

10. FUTURE RESEARCH

When business stores the enterprise data in in-memory columnar data base the future research will be on the disaster recovery. The usage of big data, text analytics and unstructured data for the data warehousing needs to be deeply studied along with integration with cloud.

11. CONCLUSION

Today's business needs real time analytics to succeed in the customer market. The traditional OLAP analysis report which is based on the extraction of data in regular interval will not suit for enterprise that needs more real time analytics. Also performance is one of the major considerations for the business intelligence applications. Column oriented database is best suited for performance oriented business intelligence applications, but if that can be made real time enabled it will be still better. So there is a need for having a common database for OLTP and OLAP system which enables scope for real time analytics. Upon checking the option of having same common database for OLTP and OLAP it is clearly found that in-memory column store is best suited for ERP based Data management system which involves intensive updates also. It can utilize the modern hardware resources effectively and in-memory columnar store can support the OLAP system for analysis purpose effectively. When this OLTP and OLAP share same database such as in memory column store it will definitely enable better real time analytics based on real time data. When organization have real time analytics data it can help in effective decision making and create new business strategy's based on the result to make their business run better.

12. ACKNOWLEDGMENTS

Our sincere thanks to Prof.Dr.Hasso Plattner, Co-Founder of SAP AG, Germany for all his contributions through various articles and seminars about the possibility of having a shared database for OLTP and OLAP system. Our thanks to Hasso Plattner Institute for IT Systems Engineering articles that helped us a lot to explore more in this area.

13. REFERENCES

[1] W. H. Inmon. Building the Data Warehouse, 3rd Edition. John Wiley & Sons, Inc., New York, NY,USA, 2002

[2] Hasso Plattner, Hasso Plattner Institute for IT Systems Engineering, A Common Database Approach for OLTP and OLAP Using an In-Memory Column Database

[3] Jens Krueger, In-Memory Data Management for Enterprise Applications, Hasso Plattner Institute for Software Engineering

[4] P. A. Boncz, S. Manegold, and M. L. Kersten.Database Architecture Optimized for the New

[5] Bottleneck: Memory Access. In VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland,UK, pages 54-65. Morgan Kaufmann, 1999

[6] Whitepaper, SAP HANA® Databas e for Next-Generation Business Applications and Real-Time Analytics

[7] Jens Krueger, In-Memory Data Management for Enterprise Applications. Hasso Plattner Institute for Software Engineering

[8] J. Gray. Tape is Dead. Disk is Tape. Flash is Disk,RAM Locality is King. Storage Guru Gong Show,Redmon, WA, 2006.

[9] Jan Schaffner, Anja Bog, Jens Kr¨uger, and Alexander Zeier . A Hybrid Row-Column OLTP Database Architecture for Operational Reporting, Hasso Plattner Institute for IT Systems Engineering,

[10] M. Stonebraker et al. C-Store: A Column-oriented DBMS. In Proc. VLDB, 2005.

[11] Jens Krueger, Martin Grund, Johannes Wust, Alexander Zeier, Hasso Plattner, Merging Differential Updates in In-Memory Column Store, Hasso Plattner Institute for IT Systems Engineering