Database Architecture with Row and Column stores

Mrs G.Prisilla Jayanthi

ABSTRACT:

Data warehouses perform better with Column Stores as they involve analytical applications like reporting from the information present in them. However, there are reports other than the Informational reports that business organizations need. *Real-time* reports, called the Operational Reports are needed.
Operational reporting differs from informational reporting in that its scope is on day-to-day operations and thus requires data on the detail of individual transactions. It is often not desirable to maintain data on such detailed level in the data warehouse and the update frequency required for operational reports. Therefore, a Hybrid Row-Column database architecture is proposed. An OLTP database architecture that serves the conventional OLTP load out of a row-store database and serves operational reporting queries out of a column-store database which holds the subset of the data in the row store required for operational reports. The column-store is updated within the transaction of the row database; hence OLTP changes are directly reflected in operational reports.
The project done with SAP AG have been studied and the results of are described here. The described solution for Operational Reporting was implemented in SAP Business ByDesign and SAP Business Warehouse.
Row and Column stores along with their applications have been briefed. Then Operational Reporting is described as being different from Informational Reporting and the need for Operational reporting is identified. As the Hybrid Row-Column Database Architecture for Operational reporting solution was implemented in SAP Business ByDesign and SAP Business Warehouse, SAP Business ByDesign, TREX and MaxDB have been discussed briefly.

Informational And Operational Reporting:

Informational Reports are summary reports usually containing past information. These reports help the top or middle management in making long-term decisions.

Operational Reports are detailed reports and are real-time. They help front-line operations personnel in making very short term detailed decisions.

Informational Reports are used strategically whereas operational reports are used tactically by the corporation. Middle management uses informational reports in order to make long-term decisions. In an informational report, the details are almost irrelevant, but the summarizations are everything. Examples of Informational reporting include monthly sales trends, annual revenue, regional sales by product line for the quarter, industry production figures for the year, number of employees by quarter and weekly shipping costs by carrier. The decisions that these kinds of reports support are longer term and tend to be strategic [1]. However, informational reports are not the only reports needed by organizations for decision making. Reports on day-to-day operations are also needed for small-term decision making. These reports are called operational reports and the activity is Operational Reporting.

Operational Reporting is about details. It is typically used by the front-line operations personnel. Very short-term, detailed decisions are made from operational reports. Operational Reporting is designed to support the detailed day-to-day activities of the corporation at the transaction level. In operational reporting, detail is much more important than summary. In fact, in operational reporting, summary information is often irrelevant. Examples of operational reporting include daily account audits and adjustments, daily production records, flight-by-flight traveler logs and transaction logs. The kinds of decisions made from these reports are very detailed, immediate decisions are made under the line-manager level .

Why Operational Reporting?

Operational Reporting is also called “Enterprise Reporting” or “Transactional Reporting” . Operational Reporting is made up of detailed information, organized to meet the needs of each area of business or function (eg: report turnover of a period type of a client or product portfolio composition of orders by geographical area or branch, retail cost structure of a particular department, the average cost resources per cost center).

The operational reporting provides a thorough and detailed management process for each center of responsibility (focusing for example, on revenues of individual products / divisions or costs related to each cost center).

It is a tool for those responsible for operational functions of the company and commercial production, always useful to have financial data under control (revenues, costs and margins of its areas) both indicators efficiency of processes .

One example of operational reporting is planning activities, for example when trying to forecast sales for products offered. Such planning is important to estimate cash-flow and to drive manufacturing planning in order to cater for the expected demand. This plan will then be tracked against actual sales and will frequently be updated [4]. Often there are requirements to look at the sales data on intra-day basis to support decisions related to production planning or financial planning.

Data Warehousing or OLTP Store?

The fact that operational reporting is real-time and very different from informational reporting makes it difficult to choose from row-stores and column-stores. Data warehousing appears to be an obvious option since reporting is involved hence column stores can be considered. However, the real-time nature of the queries makes (OnLine Transaction Processing)OLTP database also a viable option making us consider row-stores also.

Row stores store the content by row. They are suitable for OLTP. An OLTP database is designed to record hence they are *write-optimized.* A row-oriented implementation of a DBMS stores every attribute of a given row in sequence, with the last entry of one row followed by the first entry of the next.

Column stores store the content by column. They better suit the (OnLine Analytical Processing)OLAP applications and the data warehouse. A data warehouse is a database that is designed for facilitating querying and analysis. Often designed as OLAP systems, these databases contain read-only data that can be queried and analyzed far more efficiently as compared to regular OLTP application databases. Hence they are *read-optimized.*

Row-stores can be emulated as column-stores using mechanisms like Vertical Partitioning, Index-only Plans and Materialized Views but it is seen that none of them perform as well as column-stores do on data warehouses. Thus, column stores are more efficient than row-stores when it comes to data retrieval, analysis and informational reporting.

The fact that operational reporting is real-time and very different from informational reporting makes it difficult to choose from row-stores and column-stores. Data warehousing appears to be an obvious option since reporting is involved hence column stores can be considered. However, the real-time nature of the queries makes OLTP database also a viable option making us consider row-stores also.

Row stores store the content by row. They are suitable for OLTP. An OLTP database is designed to record hence they are *write-optimized.* A row-oriented implementation of a DBMS stores every attribute of a given row in sequence, with the last entry of one row followed by the first entry of the next.

Column stores store the content by column. They better suit the OLAP applications and the data warehouse. A data warehouse is a database that is designed for facilitating querying and analysis. Often designed as OLAP systems, these databases contain read-only data that can be queried and analyzed far more efficiently as compared to regular OLTP application databases. Hence they are *read-optimized.*

Row-stores can be emulated as column-stores using mechanisms like Vertical Partitioning, Index-only Plans and Materialized Views but it is seen that none of them perform as well as column-stores do on data warehouses. Thus, column stores are more efficient than row-stores when it comes to data retrieval, analysis and informational reporting.

Using an OLTP store for Operational Reporting:
Though operational reporting is real-time, the operational reporting queries are relatively long-running in comparison to pure OLTP workloads resulting in resource contention as the locks of long-running queries block the short-running ones. Also, it is well known that the OLTP data model is not optimized for reporting.

Using a DW for Operational Reporting:
What happens to the data warehouse environment if you try to force all reports out of the data warehouse? Some consequences are:

- The data warehouse is forced to be designed to the lowest level of granularity of the corporation. This may or may not be an acceptable design constraint.
- Update will need to be done in the data warehouse. While this can usually be done in the DBMS that manages the data warehouse, this in many ways runs contrary to the way data warehouses operate.
- The timing of adjustments and transactions being entered and reflected inside the data warehouse becomes an issue, usually to the detriment of other operations occurring inside the data warehouse.

In short, the constraints placed on the data warehouses are not positive when operational reporting is done on the data warehouses [1].

Real-time DW Architectures:

The general architecture of data warehouses is well known. The characteristics defined by Inmon, i.e. that data in a warehouse must be subject-oriented, integrated, time-variant, and non-volatile, led to an architecture separating operational and analytical data. Data in OLTP systems is organized according to the relational model, i.e. data is highly normalized in order to ensure consistency and to run day-to-day according to the dimensional model, using for example, the star or the snow-flake schema. The reason for this is mainly the wish to achieve the best query performance for both OLTP and OLAP.

The data warehouse contains an ETL processor which extracts data from various OLTP sources into a staging area, where data transformations for cleansing and integration are applied. Once this process has been completed, the ETL processor stores the data according to a dimensional data storage paradigm, so that an OLAP engine can run queries against this dimensional data store.

With the proliferation of (Business Intelligence)BI technologies, this general architecture has been extended with concepts such as data marts or Operational Data Stores (ODS). Data marts aim at decentralizing warehouses in order to optimize performance around certain subject areas. The downside is that in data mart architectures, the data warehouse cannot provide a consolidated view on all data relevant for strategic decision making in an enterprise, which was the original intent of data warehouses. ODSs store OLTP data, often using an integrated schema, i.e. the ETL steps of data mapping and cleansing are applied before moving data into an ODS. The result is increased timeliness of the data on which reporting can be done, as well as the possibility to work on line-item level incase the ODS is modeled that way. It remains, however, an expensive operation to refresh the ODS. One possibility to optimize timeliness of operational data being available in an OLAP environment would be to shorten the intervals between ETL runs to a minimum. The main disadvantage of such *Microbatch* approaches is the resource consumption of the frequent ETL runs: The ETL process should only run in a defined batch window, because the query performance of the warehouse is dramatically affected during ETL processing time.

To enable less resource-intensive ETL processing, architectures have been proposed that move the data transformation outside of the ETL process. Instead, the transformations are done in the warehouse after extraction and loading. Such processing is called ELT, respectively. Also, *push architectures* for ETL have been proposed in order to replace bulk processing with the handling of deltas on a business or database transaction level. Kimball further suggests separating historical data from recent data in a warehouse. The recent data is constantly copied into the so-called *real-time partition* using the push approach described above. In doing so, the data warehouse can still be optimized for queries on historical data, while recent events in an enterprise are also recorded in the warehouse.

TABLE 1: REAL-TIME DATAWAREHOUSE ARCHITECTURE

Architecture	***Working***	***Usage***	***Limitation(s)***
ODS	Stores copy of the OLTP data using an integrated schema	Reporting can be done with data of increased timeliness	High granularity but no up-to-date data Refreshing ODS with updates is a very expensive operation
ODS Microbatch	Configure ETL process to run in very short intervals	ETL process should run in a defined batch window	Resource consumption of frequent ETL runs dramatically affect the query performance
Push Architecture (ELT)	Data transformations are moved outside the ETL process. Transformations are done in the datawarehouse after extraction and loading	Handles deltas on a business or database transaction level	Resource intensive High granularity but no up-to-date data
Virtual ODS (Pull-oriented architecture)	Queries are redirected against OLTP system	*Inmon* argues that Virtual ODS architectures are of limited use when the data in the source systems is not integrated.	Affects performance of OLTP system

The notion of virtual ODS, as opposed to the traditional, physical ODS describes a pull-oriented OLAP architecture which gathers the requested information at query run-time. The ODS is virtual in the sense that it translates data warehousing queries into downstream queries to OLTP or third-party systems without persisting any data. Inmon argues that virtual ODS architectures are of limited use when the data in the source systems is not integrated. This is due to the fact that virtual ODS systems do not provide ETL transformations at run-time, which would be necessary to provide for data integration. The reason is that ETL transformations are costly and there is, thus, a trade-off between the extent of functionality offered in a virtual ODS and the response times experienced by end-users [1]. The 'ETL'comes from **'Extract, transform, and load'**- the ETL tools were created to improve and facilitate data warehousing.

SAP:

SAP stands for Systeme, Andwendungen, Produkte in der Datenverarbeitung (in German) . SAP, the company was founded in Germany in 1972 by five ex-IBM engineers.

SAP is the leading Enterprise Information and Management package worldwide. Use of this package makes it possible to track and manage, in real-time, sales, production, finance accounting and human resource in an enterprise.

Traditional computer information systems used by many businesses today have been developed to accomplish some specific tasks and provide reports and analysis of events that have already taken place. Examples are accounting general ledger systems. Occasionally some systems operate in "real-time" mode, that is have up to date information in them and can be used to actually control events. A typical company has many separate systems to mange different processes like production, sales and accounting. Each of these systems has its own database and seldom passes information to other systems in a timely manner.

SAP takes a different approach. There is only one information system in an enterprise, SAP. All applications access common data. Real events in the business initiate transactions. Accounting is done automatically by events in sales and production. Sales can see when products can be delivered. Production schedules are driven by sales. The whole system is designed to be real-time and not historical.

SAP structure embodies what are considered the "best business practices" .

SAP BI Accelerator:

SAP's Business Intelligence Accelerator is a product that's all RAM based, and generally geared for multi-hundred-gigabytes data marts. The basic design is a compression-heavy column-based architecture, evolved from SAP's text-indexing technology TREX.

SAP TREX:

When TREX updates an index, it rewrites the majority of the index files. If the indexes are large this process can take a long time and generate a high system load.

TREX allows you activate a *Delta Index* in order to speed up the update. The delta index is a separate index that TREX creates in addition to the main index. The main index and its delta index only differ TREX-internally. Outside TREX they form a unit.

If the delta index is activated changes flow into the delta index. Because, the delta index is smaller than the main index, fewer documents are affected by the update. The delta index can therefore be updated more quickly.

The delta index only speeds up the update if it is kept small. If it becomes too large, it no longer improves performance. When it reaches a certain size you have to integrate it in the main index. You can integrate the delta index manually or configure TREX so that TREX regularly integrates it automatically. TREX creates a new delta index automatically when the integration of the previous delta index is complete.
The integration process involves TREX rewriting all main index files. The duration of the integration process depends on the size of the main index.
The index server cannot index new documents during the integration of the delta index. This has the following effects:

- If indexing takes place with a queue server, the queue server retains the documents until the integration process has been completed. Then the queue server transmits the documents to the index server.
- If indexing takes place without a queue server, the application can continue to send indexing requests to the index server. However, the index server only processes them after the completion of the integration process. This means that it takes longer for indexing requests to be processed and for the application to receive the relevant response [7].

SAP MaxDB

SAP MaxDB is the database management system developed and supported by SAP AG. It has its focus on the requirements of SAP customers and SAP applications and can be used as a less expensive alternative to databases from other vendors for your own or third-party applications as well. It is a competitive database management system for medium to large server configurations and also a convincing offering for a desktop or laptop database management system, as SAP MaxDB is very easy to install and operate.

The key benefits of SAP MaxDB are its many built-in self-administering features. SAP MaxDB is available for the most prominent operating system/hardware platforms Microsoft Windows, Linux, and Unix .

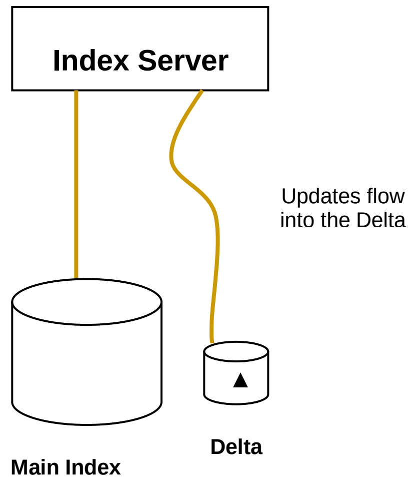

Figure 1: TREX Index Server

SAP Business ByDesign

Business ByDesign is a SAP application software suite. The software is a software-as-a-service (SaaS) product and is remarkable in its overall product breadth.

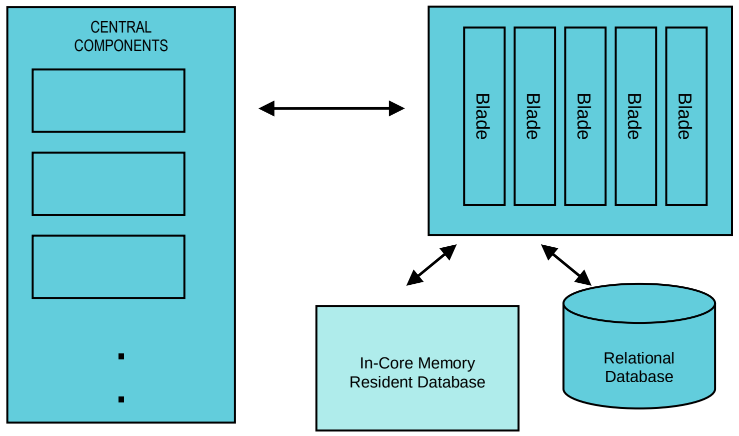

Figure 2: SAP Business ByDesign Technical Architecture

Originally, SAP could support approximately 20 concurrent users per blade. Today, the company believes it can support 150-200 concurrent users/blade.

SAP has created a number of “central components”. These are data and processing constants that do not change because of a single customer. Think of these components as containing functionality and tables to support: tax calculations, printing services, knowledge management and more.

Next, SAP has moved customer-specific data to a relational database. This database is stored on a more traditional disk storage device. Alternatively, SAP is also ramping up its use of in-core, memory resident technology .

ARCHITECTURE OF THE HYBRID SYSTEM:

In this architecture for operational reporting no replication into a data warehouse occurs, but data is accessed directly in the transactional system when queried.

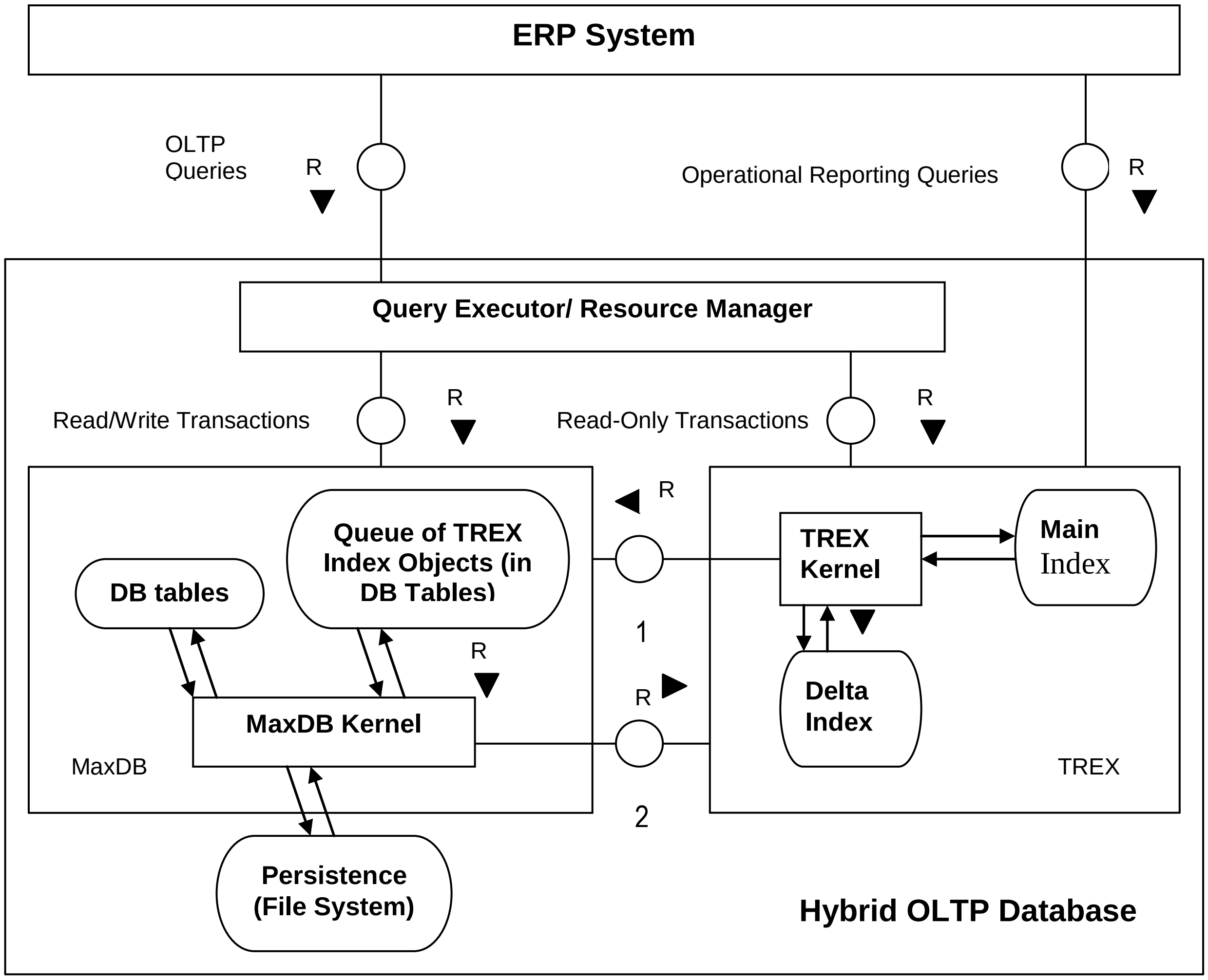

Figure 3: Hybrid Database Architecture

This architecture consists of both a row-oriented database for the entity-oriented OLTP operations (i.e “full-row” queries) and a column-oriented database to handle the operational reporting queries. SAP MaxDB is discussed as the row-oriented database, since it is the database underlying SAP Business ByDesign. It fully supports ACID. As the column-oriented database SAP’s Text Retrieval and Information Extraction engine (TREX), which is the engine underlying SAP BIA, is used that answers operational reporting in Business ByDesign. TREX has originally been developed as a text search engine for indexing and fast retrieval of unstructured data. The figure shows how TREX is integrated with MaxDB and how read and write access is distributed between TREX and MaxDB.

Requests that change data or insert new data are handled by MaxDB, which ensures the ACID properties. The MaxDB kernel stores the changes in the database tables and manages so-called “queue tables” for TREX. Those queue tables containing information about data that is relevant for operational reporting and that has been updated or inserted in MaxDB. TREX is notified about the changes and can update its own data with the help of the queue tables. This happens within the same database transaction using on-update and on-insert triggers, which are created inside MaxDB when the system is configured. The triggers fire stored procedures which forward the queries to TREX. Accordingly, TREX and MaxDB share a consistent view of data.

Queries for operational reporting and some read-only transactions go directly to TREX. TREX manages its data in so-called *main indexes*. The main index holds a subset (i.e. the OLTP data that must be available for operational reporting) of the database tables in MaxDB, except that the schema is flattened and the data is stored column-wise. Since the main index is highly optimized for read access, TREX holds a delta index to allow fast retrieval while concurrently updating its data set. All updates and inserts taken from the queue tables are collected in the delta index. When responding to a query, data in the delta index as well as the main index is accessed and the results are merged together in order to provide a consistent view of the entire data set compared with the data base tables of MaxDB. The delta index is not compressed to allow for fast writes, while it must be able to provide a reasonable read performance. In the current implementation delta indexes are organized as a B-tree in memory. It is important that delta indexes do not grow larger than a certain size limit, since the read time from the delta index should not exceed the read times from the main index of a given table (main and delta index are read in parallel). Upon reaching a certain size or pre-defined intervals the delta index is merged with the main index. To do so, a copy of the index is created in memory along with a new, empty delta index for that copied index. The queries coming in during the merge will be run against this structure. In particular, the new delta index that receives all inserts and updates during the merge. After the merge of the original delta index and the main index, the new delta index becomes the delta index for the merged index. The copy of the original main index in memory is now discarded. This procedure for merging a delta index with a main index ensures that neither read nor write accesses to delta index with a main index ensures that neither read nor write accesses to TREX are blocked during merge time.

In the prototypical implementation, MaxDB was extended to serve as the primary persistence for the TREX indexes (i.e. tables), as opposed to using the standard file system persistence of TREX. The reason is that in case of a disaster both MaxDb and TREX can be recovered to a valid state using MaxDB's recovery manager. Therefore, the queue tables in MaxDB also serve as a recovery log for the TREX delta indexes. A possible direction for future research would be to conceptually move the delta index in the column store completely into the row store.

VIRTUAL CUBE:

A virtual cube allows for seamless consumption of operational reports from a data warehouse environment. From a data warehouse perspective the consumption is seamless because the same interface is provided as for a standard cube.

Figure provides a high level view of the target architecture. The analytical engine accesses data through the virtual cube. The virtual cube provides the same interface for analysis as standard cubes in a data warehouse. This includes navigation along various levels of hierarchy (i.e. drill-down and roll-up) as well as slicing and dicing along different dimensions. In the prototypical implementation, a virtual cube that is plugged into the OLAP engine of SAP BI was created. In consequence, all reporting front-ends supported by SAP BI can be used to launch queries against the OLTP data.

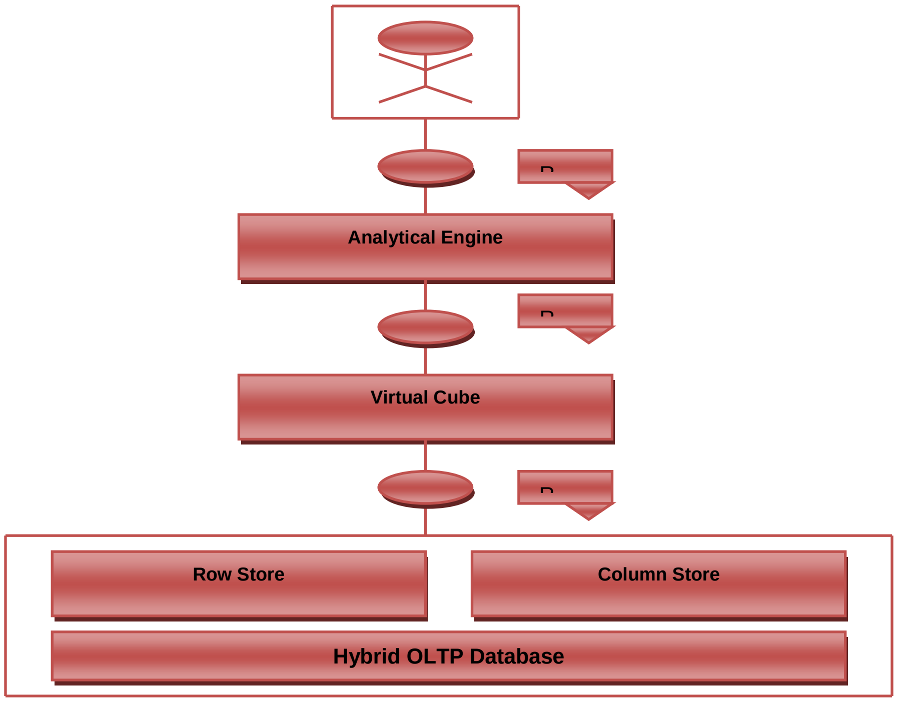

Figure 4: Virtual Cube

In comparison to traditional cubes in a warehouse, the virtual cube does not store any data. Instead, it is a collection of functions that are executed during the run-time of a report. The operational reporting queries from the reporting front-end are sent to the OLAP engine, which then executes OLAP queries against the virtual cube. The latter transforms the incoming queries into queries against TREX.
While the strategy for obtaining the best response times for operational reporting is to push as much as computation as possible down to TREX, predicate filtering on attributes is handled by the OLAP engine of SAP BI in the prototypical implementation. While it would technically be possible to push filtering down to the TREX layer, the result filtering capabilities of OLAP engine are used due to some of the particularities of TREX's query API that would have required to implement a more elaborate query rewriter between the OLAP engine and TREX.

Discussions:

There are many tools, techniques and software available helping in Operational Reporting by generating operational reports and making the real-time information flow across the different departments of an organization. SAP is itself a business suite which helps in this.

The aim of this study was not to identify or compare these tools or software but to identify which database architecture (row or column store) would better suit Operational Reporting. And the study concludes that row-stores or column-stores, if used individually, have their own disadvantages. Further, the study suggests that A Hybrid Row-Column Database Architecture would be needed for Operational Reporting.

References:

http://www.dbms2.com/2006/09/20/saps-bi-accelerator/
http://help.sap.com/saphelp_nw04s/helpdata/en/2a/c4a74046033813e10000000a155106/frameset.htm
http://www.sdn.sap.com/irj/sdn/maxdb?rid=/webcontent/uuid/400cac68-a958-2910-0b9e-a15a67fd4b06
http://blogs.zdnet.com/sommer/?p=735
http://www.information-management.com/issues/20000701/2349-1.html
http://www.rossmorrissey.com/uv
http://www.tagetik.com/resources/glossary/operational-reporting
http://www.vldb.org/conf/2008/workshops/WProc_BIRTE/p7-schaffner.pdf
http://www.onestopsap.com/introduction/1.asp