Sustainable Cities and Society 35 (2017) 232–240

Contents lists available at ScienceDirect

Sustainable Cities and Society

journal homepage: www.elsevier.com/locate/scs

Efficient data access and performance improvement model for virtual data warehouse

Fakhri Alam Khan[a,*], Awais Ahmad[a], Muhammad Imran[b], Mafawez Alharbi[c], Mujeeb-ur-rehman[a], Bilal Jan[b]

[a] *Centre of Excellence in IT, Imsciences Peshawar, Pakistan*
[b] *Sarhad University of Science and Technology, Peshawar, Pakistan*
[c] *Computer Science Department, College of Science, Majmaah University, Saudi Arabia*

ARTICLE INFO

Keywords:
Business intelligence
OLTP
Performance optimization
Query optimization
Virtual data warehouse

ABSTRACT

This paper presents a model for improving query performance in Virtual Data Warehouse (VDW) by simulating VDW environment on a cellular phone billing and customer care system which involves processing millions of Call Detail Records (CDRs) generated by thousands of counters across the country. Processing aggregations on millions of CDRs requires expensive systems, especially when analysing customers' traffic trends and encompasses several performance optimization techniques used for improvement of query performance in VDW. In this regard, VDW offers several advantages such as real-time analytic reports, reduced maintenance, low cost solution and flexible data integration, but performance is still one of its critical shortcomings. This paper enhances performance of VDW by using techniques like partitioned materialized views, index performance optimization, query rewrite in materialized views, analytic functions, sub-queries and enabling parallel execution etc. The study uses Oracle 10 g as a backend database; Oracle management console and SQL query analyser are used for monitoring performance concerns during validation of VDW model; standard PL/SQL developer is used for extracting and loading test data; and finally, Hyperion Development suite is used for testing time comparisons of datasets both in normal OLTP and simulated VDW environments.

1. Introduction

The concept of data warehouse dates back since 1980s when IBM researchers Barry Devlin and Paul Murphy introduced the need of a decision support system (DSS). Afterwards in early 1990s, Virtual Data Warehouse (VDW) concept became a potential research topic when processing power and storage capacities were comparatively expensive. The concept of VDW refers to developing a data warehouse on the existing operational and transactional OLTP legacy system by utilizing resources of the normal OLTP environment, either specifically sharing both processing and storage or practically by having direct connections to transactional systems.

The concept of VDW takes many distinct advantages, but still there is a major disadvantage called the Operational Performance that snags its implementation. Certainly, using data warehouse on the existing transactional system will put extra load on transactional system, thus decreasing the overall operational performance. Furthermore, unlike normal data warehouses, amassing huge volumes of archived data in the transactional system is obviously a significant challenge. The need for efficient DSS became more demanding in early 1990s due to high storage costs and time required for generating analytical reports. Even today, there are high costs involved in installing a full-fledge data warehouse – specifically the installation of expensive equipment, purchase of costly applications, cost of renewals and maintenance of BI applications and need of integration among different software. Ultimately, all these factors drastically increase the average cost of producing the management reports (Bara, 2010). Since VDWs are built on the existing transactional infrastructure, they eliminate the need of additional costs involved for purchasing new equipment and related applications.

The strategies to improve performance factor in VDW implementation encompasses several performance optimization techniques including table partitioning (Bara, 2010), SQL query optimization (Geng & Zhang, 2009), compression techniques (DeWitt, Madden, & Stonebraker, 2006), query rewrite in materialized views (Ashadevi & Balasubramanian, 2009), analytic functions (Lungu, Bâra, & Diaconita, 2006b), selection of data warehouse model (Ashadevi & Balasubramanian, 2009), sub-queries (Geng & Zhang,

* Corresponding author.
E-mail address: fakhri.alam@imsciences.edu.pk (F.A. Khan).

http://dx.doi.org/10.1016/j.scs.2017.08.003
Received 14 March 2017; Received in revised form 17 July 2017; Accepted 4 August 2017
Available online 18 August 2017
2210-6707/ © 2017 Elsevier Ltd. All rights reserved.

2009), parallel execution, indexing, SQL Performance Analyzer 11-g (Yagoub et al., 2008), SQL tuning advisor 11-g (Chan, 2005a) and rules involved in performance tuning of distributed database systems (Stonebraker & Cattell, 2011).

In this paper, we propose a conceptual model called Efficient Data Access (EDA); Performance Improvement Model for Virtual Data Warehouse that allows model-based strategies for efficient data retrieval while implementing VDW in an organization. We have developed two different variants for the proposed model and have drawn a comparative analysis based on the output produced by the original proposed model and its variants. Techniques used for experimentation are partitioned materialized views, index performance optimization techniques, query rewrite in materialized views, use of analytic functions, use of sub-queries and enabling parallel execution. The proposed model is validated using CRM and ERP datasets of a cellular company.

The proposed EDA framework consist of three layers. The first layer caters for the conceptual virtual data warehouse model entails with nine different processes. The second layer explains the variants of the proposed model. In the third layer of paper describes the performance evaluation and benchmarks using CRM and ERP datasets of a cellular company.

2. Literature review

The key to data warehouse is the design of the data, therefore, the user needs to define the necessary data, the determination of the sources of evidence and organizing data in a dimensional model that represents the business needs (MOÇKA & Daniel, 2013). Excessive load on the transactional system could be observed while implementing VDW on the existing operational system which can result in diminishing the overall system performance. The improvement of query performance in VDW is a big challenge e.g., low implementation cost versus its benefits. The contemporary literature outlines several performance optimization techniques for improving query performance in VDW including table partitioning and compression techniques etc.

Table partitioning is a logical division of table into distinct parts. Collectively all the partitions are identified as a single table at a time for better manageability and improved performance. The key objective of table partitioning is to limit the disk activity and amount of data to handle. It is reported that access time is drastically reduced when same queries are tested on partitioned tables. It means that partitioning is a key technique to enhance performance in virtual systems. Retaining historical data in the operational systems is possible by applying different partitioning methods (e.g., range partitioning, list partitioning, hash partitioning, key partitioning and sub partitioning) which logically divide operational tables and hence improve query performance without affecting operational system performance. Aggregate functions commonly used with GROUP BY and HAVING clauses slow down the SELECT operation; therefore, avoiding the use of multiple group functions is considered as a decent performance improvement strategy. In virtual OLAP environment, this strategy can also be useful to create materialized views for OLAP queries to fetch input data from operational system (Bara, 2010).

Materialized views retain data on their creation which is contrary to the simple views which are just snapshot of the online tables. Due to this fact, changes made in the original tables are not reflected in the materialized views. In fact, materialized views fetch instructed data only on their creation. Therefore, loading data from online tables to materialized views decreases load on the database which is ultimately another step towards improving performance measures in virtual OLAP systems (Ashadevi & Balasubramanian, 2009; Thakur & Gosain, 2011).

Views can also be used as an optimizer by eliminating unnecessary fields by restricting WHERE criteria to be precisely specific, especially this is true for complex views. For instance, an example could be: CREATE OR REPLACE VIEW as SELECT SUM (employee salary) from employee table WHERE employee department = ‘FINANCE’.

Similarly, the proper use of indexing can also enhance query performance, therefore, it is recommended to index all those database fields which are likely to be used in the WHERE clause. Geng and Zhang (2009) recommend removing index on tables while inserting data as it increases efficiency of the query. It is advised to always use BITMAP index on frequently repeated values. For example, in case of a gender field, use bitmap index for its possible values ‘male’ or ‘female’.

Indexing performance techniques include avoiding full table scan and disk activity by reducing I/O operations. It provides a quicker way to access data from the tables. Several indexing techniques are used in different types of databases e.g., B-Tree Index is used in Oracle databases. Other types are bitmap indexes and cluster indexes which are employed based on associated use; for example, indexes applied on tables with few rows will slow down query access time. Lungu, Velicanu, Bara, Diaconita, & Botha, 2008) describe SORT-MERGE-JOIN as the most efficient technique to increase database performance and efficiency, particularly when it is applied to the properly indexed portioned tables (Geng & Zhang, 2009).

Performance improvement not only depends on hardware but is equally dependent on the software artifacts. Vertical database partitioning and compression supported databases bring immense performance advantage by using shared-nothing hardware architecture (DeWitt et al., 2006). In distributed computing, vertical database partitioning refers to as a logical division of database into column-oriented database architecture. Such architecture bears database performance advantage when used with shared-nothing hardware architecture because query only accesses the specified columns; hence decreasing the disk reads. Similarly, the compression supported databases also help increase the overall system performance by utilizing more CPU and storage (i.e., CPU and storage cost per byte decreases), and by minimizing burden on the disk bandwidth (i.e., expensive memory that loads data from storage is used less frequently) (DeWitt et al., 2006; Stonebraker & Cattell, 2011).

Analytic functions offer an enhanced functionality over the existing aggregate functions in the data warehouse environment, especially they are more useful for improving performance in virtual data warehouse environment. For example, extracting aggregations from online views in virtual data warehouse environment can be time consuming but analytic functions perform more efficiently. Analytic functions are useful in case integration between source data and virtual data warehouse is not possible (Lungu et al., 2006b). The most challenging task for DBAs is to predict the impact of changes on database performance before implementing the required changes into the production system. Also, the software testers and application developers can use Oracle SQL Performance Analyzer to identify performance issues within their domains and take corrective measures to tackle performance degradation accordingly (Yagoub et al., 2008).

Query rewrite can be used in materialized views with SELECT, INSERT or CREATE TABLE clause as a performance optimization technique. Query rewrite can be explicitly enabled or disabled in materialized view using the command query rewrite = enabled/disabled. It means a query rewrite can automatically occur without involvement of a DBA or application developer if it is enabled both in session and materialized view. Implementing materialized views in data warehouse environment brings several performance advantages such as improving response time and maintaining data integrity. This is done by using specific query integrity modes TRUSTED, ENFORCED and STATE_TOLERATED (Ashadevi & Balasubramanian, 2009).

A special case of the snowflake schema is the Star Schema which consists of one or more fact tables referencing to multiple dimension tables (Levene & Loizou, 2003). The star schema is considered as the simplest style of data mart schema and is more effective for handling simpler queries. A distinct difference between data warehouse and OLTP systems is the use of dimensional modeling technique called ‘Star Schema’ which is represented through a FACT table (containing facts about anything e.g., subscriber transactions) and a DIMENSION table.

Both tables are joined by simple primary-foreign key relationship in a star shaped model. In oracle 11 g, the optimizer quickly recognizes star queries and creates an optimization plan. The star schema provides distinct advantages of fast and efficient star queries which are widely supported by BI applications for end-user presentation (Lane, 2003).

Unlike OLTP environment where short queries are used, parallel execution offers performance improvement in data warehouse environment with loads of CPU and memory intensive queries. Parallel implementation can be done in large partitioned tables, large indexes, bulk DML (i.e., batch transformation) and large materialized views which bring performance advantage by fully utilizing CPU bandwidth and memory. However, parallel execution is not recommended for short query transactions (i.e., in OLTP environment) or on a system with lesser amount of hardware resources (Chan, 2005a; Stonebraker & Cattell, 2011).

The characteristics of transactional databases, operational data stores, data warehouses and virtual data warehouses are examined in (Chan, 2005b) by describing enterprise architecture to provide an integrated and complimentary view of various data warehousing constructs. For this, it is necessary to formulate data warehousing strategy based on organizational needs duly classified by the types of business processes defined by the requirements of supporting functional areas and the levels of decision structures. Aji et al. (2013) present a scalable spatial data warehousing system called Hadoop-GIS for executing large scale declarative spatial queries on Hadoop by integrated into Hive. For efficient query processing, it provides support for different types of spatial queries on MapReduce by using global partition indexing and customizable on demand local spatial indexing.

3. Proposed conceptual model

In the recent past, telecom sector has witnessed tremendous growth and enhanced level of innovation and creativity in terms of new communication services and products i.e., from traditional fixed landline services to wireless mobile phone networks, and from traditional broadband services to wireless broadband services. With advancements in the cutting-edge technology, the traditional TV services have witnessed a paradigm shift towards IPTV technology and advance IP based security surveillance services. However, innovation in the technology has brought about numerous assorted challenges to comply with new competitive business and consumer requirements. Nowadays, managing consumer expectations has shifted from traditional services to ultramodern state-of-the-art services. To stay competitive, the need for active and dynamic processing of data has become a key challenge for organizations for effective decision making. On the contrary, the process of taking effective decisions is becoming more and more complex due to the involvement of diverse technologies and growing customer demands.

Cellular companies ordinarily process gigantic volumes of data. For instance, on daily basis, tens of millions of end-user data are generated in the form of Usage Detailed Records (UDRs) pertaining to voice calls, text messages, multimedia content and video-streaming services. This gigantic volume of data puts an immense load on OLTP system of the cellular companies. The high degree of data processing requirements requires customized intelligent systems comprising deployment of expensive full-fledged commercial data warehouse to meet ever increasing intelligent decision making demands. However, this results in boosting up the cost of maintaining and processing information which ultimately affect organization's ability to retain cash inflows and the overall revenues. Our proposed conceptual virtual data warehouse model entails nine different processes as described below.

a.* New Project Planning*: In this process, the data pertaining to inline and ongoing projects is fed into the Enterprise Resource Planning (ERP) system including project plans, schedules, budgets, financing methods, costing methods and other associated resource allocation procedures.

b. Order/Customer Support: This process involves handling data relating to customers acquiring mobile phone services and broadband services. This also includes after sales services and support. All the data is fed into Customer Relationship Management (CRM) system.

c. Bill Generation: Monthly billing to subscribers of the acquired services is principally based on the CDRs collected from the switches, types of offers/packages and government tax policies. These tariffs are normally charged on Minutes of Usage (MOU) basis.

d. Bank Payments: In this process, the MOUs are compared with outstanding arrears viz-a-viz payments received at banks followed by generating invoices accordingly.

e. Source OLTP Systems: OLTP systems including ERP, CRM and Billing act as a source to data warehouse for extraction of miscellaneous information. Customarily, traditional systems retain six months' history data in OLTP systems; whereas, a data warehouse may contain all the historical data of an organization since its inception.

f. MIS Reporting: Management Information System (MIS) supports short term organizational reporting needs and generally involves structured reporting. Thus, an MIS is restricted to analysing data retained by the OLTP system and the data accessed by the defined user groups.

g. Data Warehouse for Business Intelligence (BI) Reporting: Another aspect to establish data warehouse is to meet the organizational long term reporting needs and process unstructured information to produce ad hoc reports (also called BI reporting). As an extension to the data warehouse, mini data marts or regional data marts can be engendered.

h. Extraction Transformation Loading (ETL): At the core of a data warehouse, a data manipulation process known as ETL (Extraction, Transform and Load) decides performance factor of a BI system. Several BI tools are available to perform ETL process that may entail additional procurement, licensing and upgrading costs.

i. Application Services: Finally, the application services act as presentation layer for formatting different reports based on end-user requirements as well as to manage different sessions transported to the database.

Our elementary conceptual virtual data warehouse is shown in Fig. 1 which is much like the notions of the general-purpose data warehouses deployed by the organizations.

3.1. Conceptual model varients

Our proposed virtual data warehouse model differs from the basic data warehouse in a way that it omits ETL process as well as separates the data warehouse and application server setups (Fig. 2). The proposed model allows end users to directly query the OLTP environment to search the required information. The proposed solution is cost-effective as it does not involve acquiring expensive hardware equipment or purchasing new BI and ETL tools, and maintenance services. It may increase BI query response time and put an extra load on OLTP system, thus slowing down overall system performance in case same query is executed by many users. Our model further constitutes two variants. The first variant employs the proposed model as an extension to the existing systems and the second variant involves setting up the proposed solution on a separate machine.

The first variant (Fig. 3) overcomes certain shortcomings such as increase in the query response time and the overall system performance. As shown in Fig. 3, the first variant requires creation of an additional warehouse database as an extension of the existing OLTP environment. This additional warehouse database contains virtual tables (or views) to save query results/aggregations so that these results are readily available when multiple users require them. The proposed approach of saving results/aggregations in virtual tables decreases load on the OLTP by making available the output of repeated queries from the virtual record set. Though the proposed solution bears merit of sharing CPU and memory of existing system, but there is a caveat in the

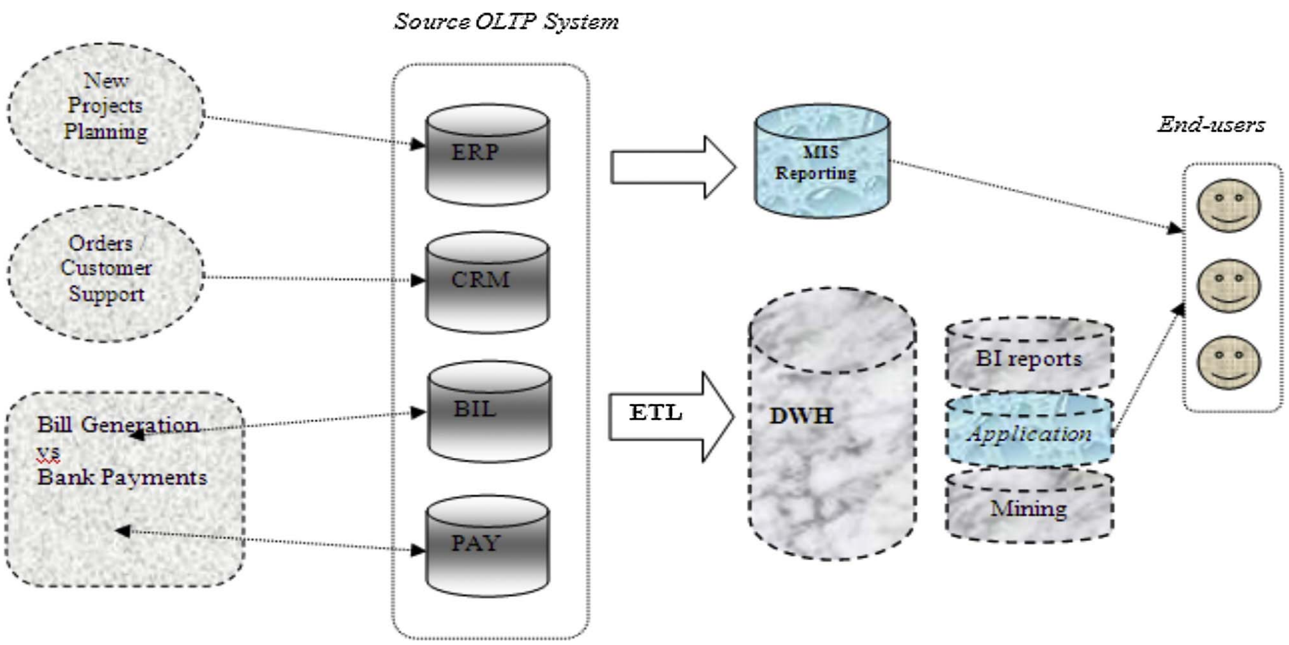


Fig. 1. Showing Elementary Data Warehouse Model.

proposed approach – the aggregated data is subject to becoming outdated quickly as it does not reflect the new changes made in the source tables.

To overcome the shortcoming of shared CPU and memory (first variant of the proposed model), we further enhance our model by deploying a separate machine for information processing (Fig. 4). The resultant solution is a second variant of our proposed model where virtual tables are created on a separate machine for storage and processing of information. This approach minimizes load on the existing OLTP system and at the same time helps balance the end-user query load.

4. Virtual data warehouse performance benchmarks

We carried out validation of our proposed virtual data warehouse models at the site of a cellular company which operates all over the country including all the standard service such as telephony services and broadband services. The company operates throughout the country. The offered services are based on a network of multiple systems that either operate independently or work in conjunction with other systems to effectively managing its mobile services. We have used subscriber's data warehouse (SDW) and interconnect traffic data warehouse (ITDW) to evaluate the efficiency of our proposed model. SDW includes details of all subscribers having different services including CDRs, billing details and payments records. The purpose of this warehouse is to analyze the customer trends. ITDW includes CDRs with other

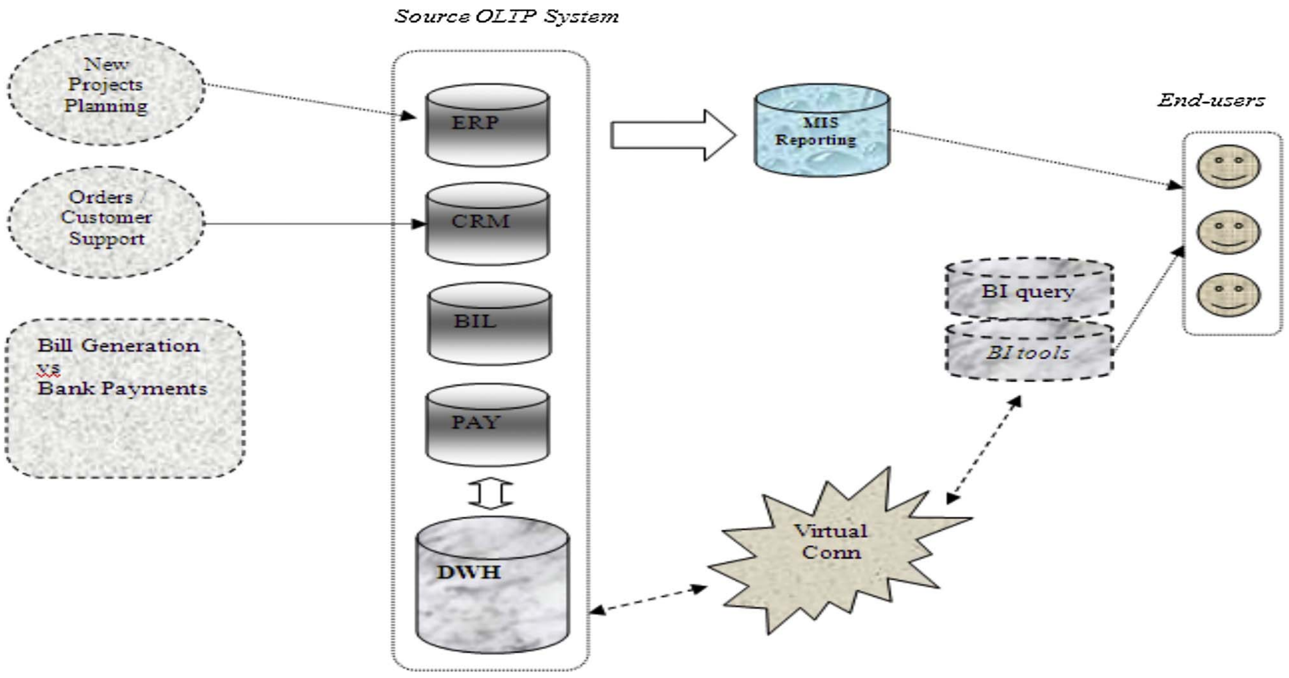


Fig. 2. Virtual Data Warehouse Model.

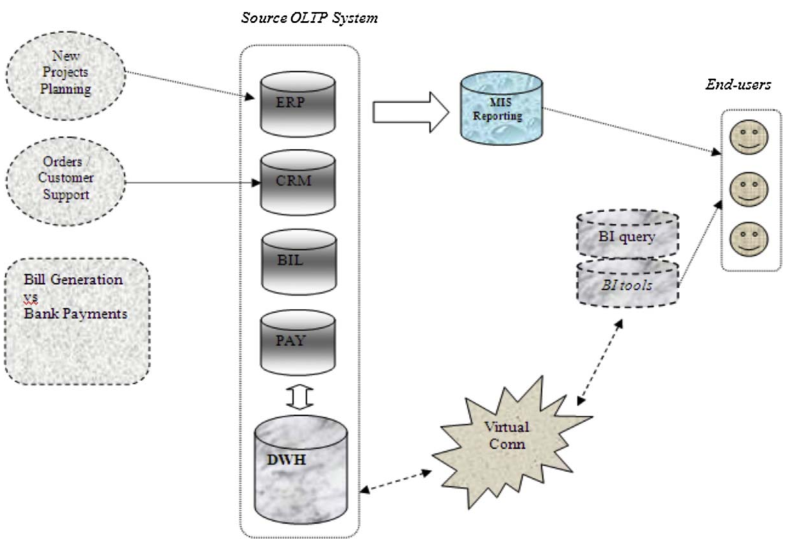


Fig. 3. Virtual Data Warehouse Model – Variant 1.

telecommunication operators and the purpose of this warehouse is to analyze traffic patterns for network traffic management and to settle payments with other operators. ITDW contains records of all calls generated from our system onto other network operators and calls originated from other network operators which were dropped to our system.

The proposed virtual models along with its two variants were tested against the datasets of SDW and ITDW. The following system attributes were used as a benchmark to measure performance of the proposed models.

The performance metrics are selected on basis of their importance in processing and implementation of a system. CPU Cost, Cache cost, response time are the three main performance indexes that define efficiency of a system. Details of these metrics are described below:

- CPU Cost: This criterion measures consumption of CPU during query processing on OLTP alone versus consumption of CPU during query processing on virtual OLAP system. The lesser the CPU usage on a standard BI query, more efficient the system is likely to be validated.
- Cache Cost: Quantifying the consumption of RAM during different virtual OLAP models helps in deciding which method is more effective in deployment of OLAP queries. Lesser memory usage means that the system is more optimized for OLAP queries.
- Response Time: Likewise, the measure of how much time the same query takes to be tested on different models can help determine which model is more efficient. The query is considered less efficient if its response time is more.

4.1. *Evaluation of the proposed model on subscriber data warehouse*

Subscriber data warehouse was selected for validation of virtual data warehouse. The extraction of subscriber data involves the following operational mappings as mentioned against each table of star schema components. All the dimension tables (subscriber, services, service type, region, exchange, and staff) have one-to-many relationship with the fact table (named as custinvoice) in the star-like schema

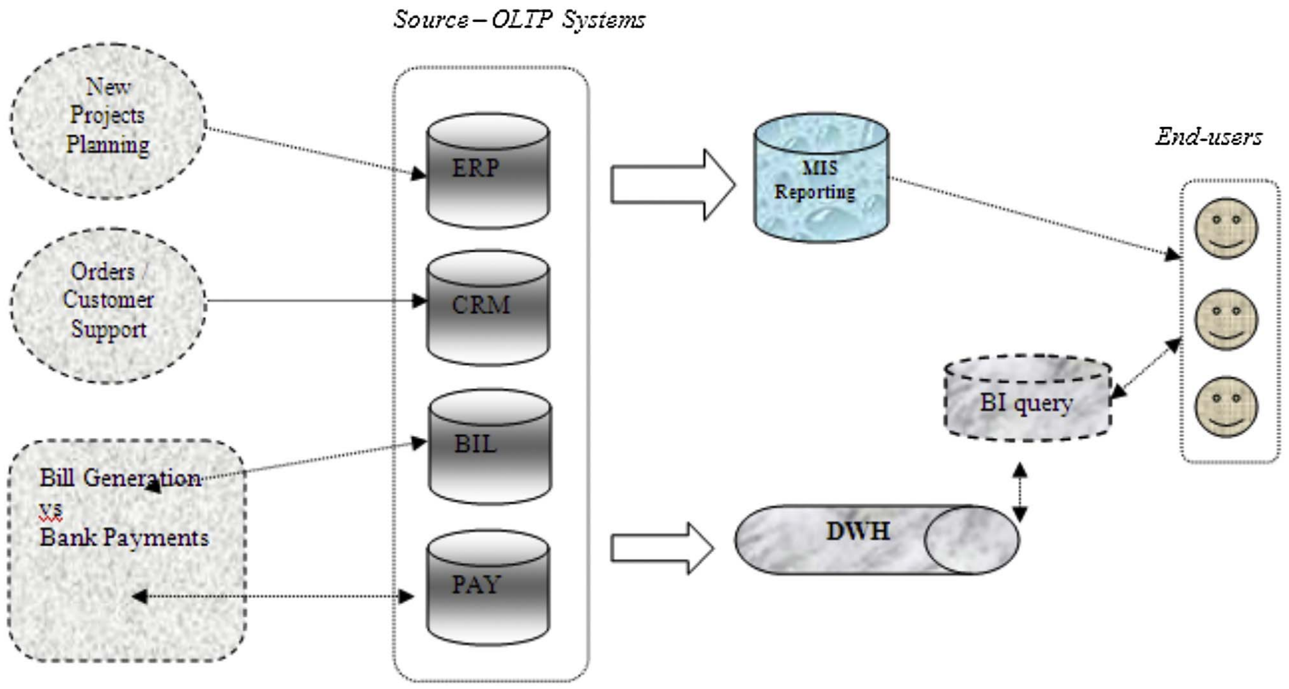


Fig. 4. Virtual Data Warehouse Model – Variant 2.

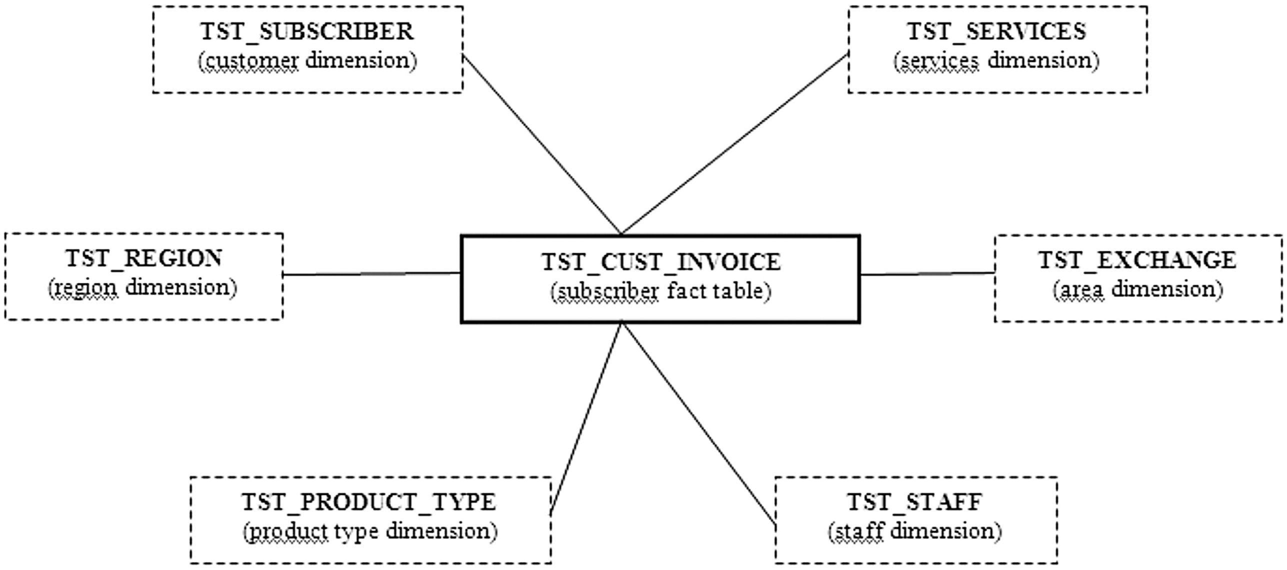
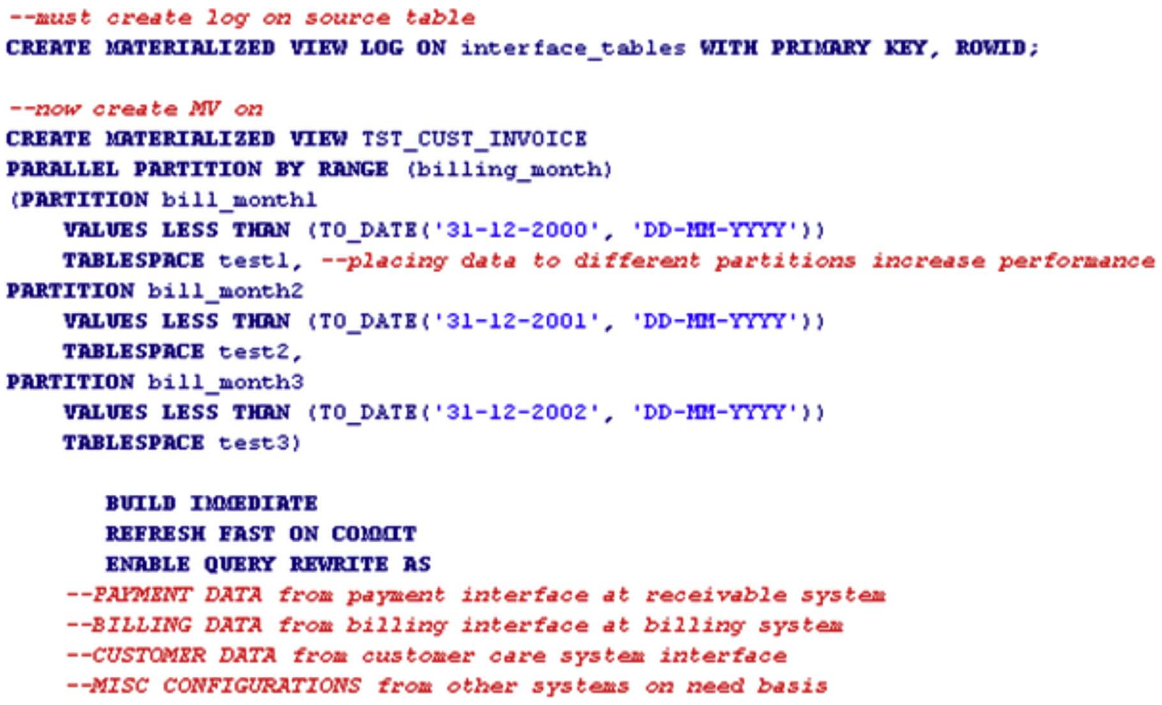


Fig. 5. Star Schema for Subscriber DWH.

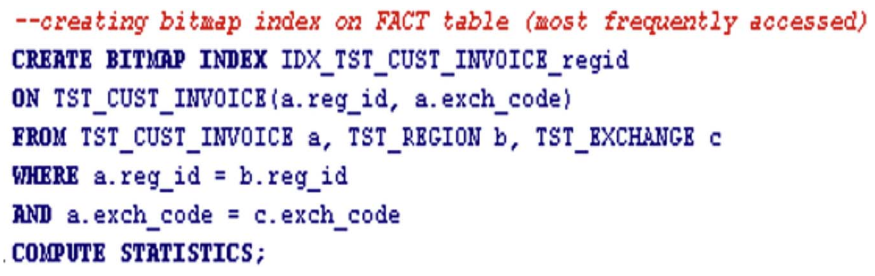


```
--must create log on source table
CREATE MATERIALIZED VIEW LOG ON interface_tables WITH PRIMARY KEY, ROWID;

--now create MV on
CREATE MATERIALIZED VIEW TST_CUST_INVOICE
PARALLEL PARTITION BY RANGE (billing_month)
(PARTITION bill_month1
    VALUES LESS THAN (TO_DATE('31-12-2000', 'DD-MM-YYYY'))
    TABLESPACE test1, --placing data to different partitions increase performance
PARTITION bill_month2
    VALUES LESS THAN (TO_DATE('31-12-2001', 'DD-MM-YYYY'))
    TABLESPACE test2,
PARTITION bill_month3
    VALUES LESS THAN (TO_DATE('31-12-2002', 'DD-MM-YYYY'))
    TABLESPACE test3)

       BUILD IMMEDIATE
       REFRESH FAST ON COMMIT
       ENABLE QUERY REWRITE AS
    --PAYMENT DATA from payment interface at receivable system
    --BILLING DATA from billing interface at billing system
    --CUSTOMER DATA from customer care system interface
    --MISC CONFIGURATIONS from other systems on need basis
```

Fig. 6. Creating partitioned materialized view.

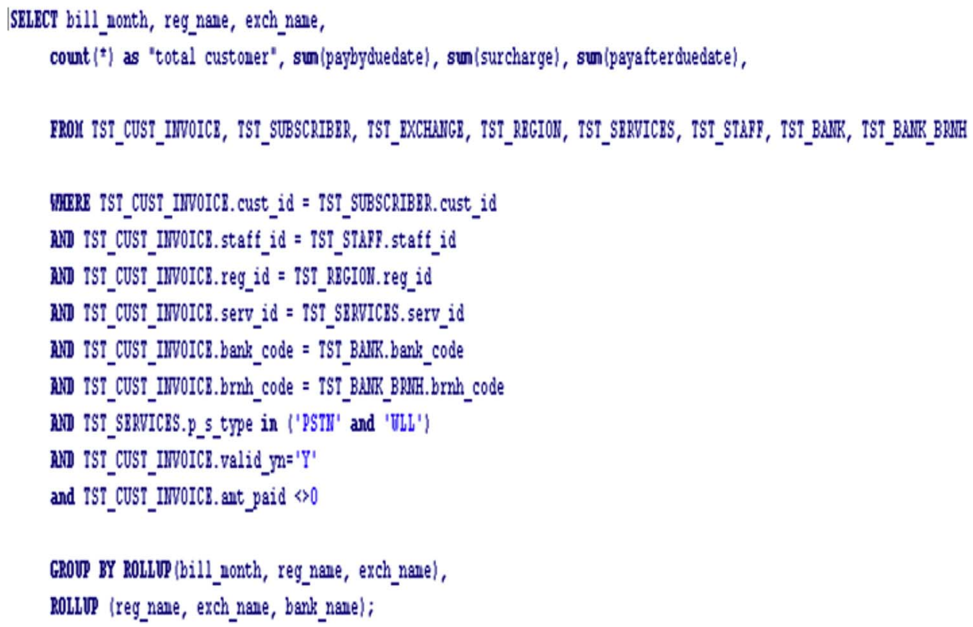


```
--creating bitmap index on FACT table (most frequently accessed)
CREATE BITMAP INDEX IDX_TST_CUST_INVOICE_regid
ON TST_CUST_INVOICE(a.reg_id, a.exch_code)
FROM TST_CUST_INVOICE a, TST_REGION b, TST_EXCHANGE c
WHERE a.reg_id = b.reg_id
AND a.exch_code = c.exch_code
COMPUTE STATISTICS;
```

Fig. 7. Creating bitmap index for fast retrieval.

as shown in Fig. 5. The description of each table used in the subscriber database is provided below.

- TST_CUST_INVOICE (FACT Table): This table contains information of subscriber from two systems — customer billing information (from billing system) and payment information (from receivable system). This table uniquely identifies each record by a composite key inv id (from billing system) and acct_id (from receivable system). This table includes both billing and payments information of all the subscribers.



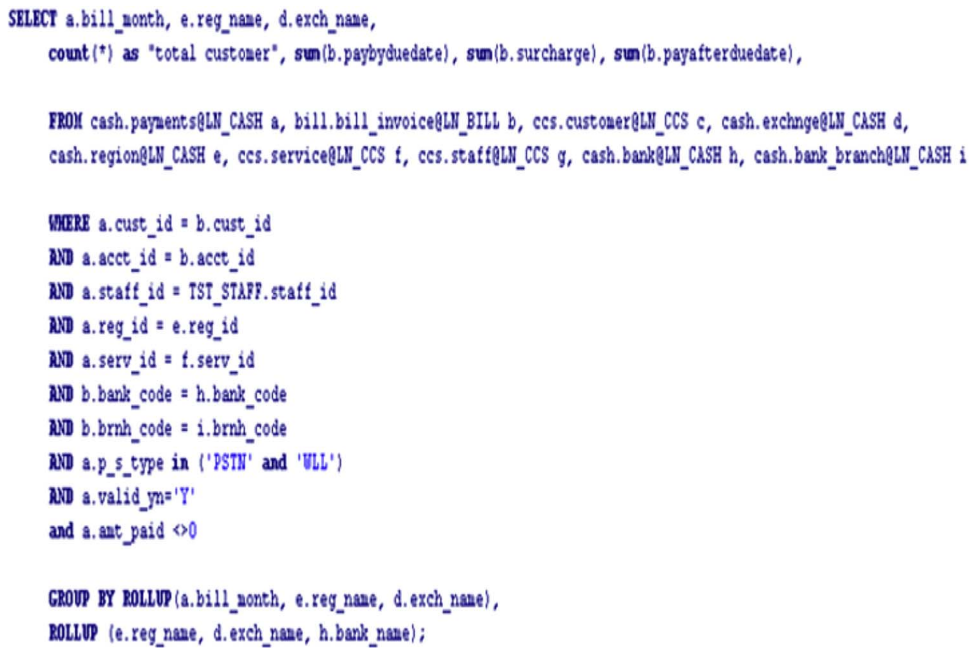
```
SELECT bill_month, reg_name, exch_name,
    count(*) as "total customer", sum(paybyduedate), sum(surcharge), sum(payafterduedate),

    FROM TST_CUST_INVOICE, TST_SUBSCRIBER, TST_EXCHANGE, TST_REGION, TST_SERVICES, TST_STAFF, TST_BANK, TST_BANK_BRNH

    WHERE TST_CUST_INVOICE.cust_id = TST_SUBSCRIBER.cust_id
    AND TST_CUST_INVOICE.staff_id = TST_STAFF.staff_id
    AND TST_CUST_INVOICE.reg_id = TST_REGION.reg_id
    AND TST_CUST_INVOICE.serv_id = TST_SERVICES.serv_id
    AND TST_CUST_INVOICE.bank_code = TST_BANK.bank_code
    AND TST_CUST_INVOICE.brnh_code = TST_BANK_BRNH.brnh_code
    AND TST_SERVICES.p_s_type in ('PSTN' and 'WLL')
    AND TST_CUST_INVOICE.valid_yn='Y'
    and TST_CUST_INVOICE.amt_paid <>0

    GROUP BY ROLLUP(bill_month, reg_name, exch_name),
    ROLLUP (reg_name, exch_name, bank_name);
```

Fig. 8. Test query on proposed model.



```
SELECT a.bill_month, e.reg_name, d.exch_name,
    count(*) as "total customer", sum(b.paybyduedate), sum(b.surcharge), sum(b.payafterduedate),

    FROM cash.payments@LN_CASH a, bill.bill_invoice@LN_BILL b, ccs.customer@LN_CCS c, cash.exchnge@LN_CASH d,
    cash.region@LN_CASH e, ccs.service@LN_CCS f, ccs.staff@LN_CCS g, cash.bank@LN_CASH h, cash.bank_branch@LN_CASH i

    WHERE a.cust_id = b.cust_id
    AND a.acct_id = b.acct_id
    AND a.staff_id = TST_STAFF.staff_id
    AND a.reg_id = e.reg_id
    AND a.serv_id = f.serv_id
    AND b.bank_code = h.bank_code
    AND b.brnh_code = i.brnh_code
    AND a.p_s_type in ('PSTN' and 'WLL')
    AND a.valid_yn='Y'
    and a.amt_paid <>0

    GROUP BY ROLLUP(a.bill_month, e.reg_name, d.exch_name),
    ROLLUP (e.reg_name, d.exch_name, h.bank_name);
```

Fig. 9. Test query on normal OLTP environment.

Table 1
Time to CPU usage comparison.

SR. No.	Activity description	Time (in min)	CPU Usage (%)
1	Direct virtual query on Source	15	61%
2	Virtual query as extension to OLTP system	1	57.90%

- TST_EXCHANGE (DIMENSION Table): This table shows total number of exchanges installed all over the country which is extracted from the centralized configurational database containing call details of the all the exchanges.

- TST_REGION (DIMENSION Table): This table holds configurations information of the respective regions pulled out from the configurational database.

- TST_SUBSCRIBER (DIMENSION Table): Subscribers have the key role in this schema, therefore, this data is fetched from customer service system.

- TST_SERVICES (DIMENSION Table): Different services offered to both individual and corporate subscribers are fetched from customer services system and is stored in this table.

- TST_PRODUCT_TYPE (DIMENSION Table): This table displays specific product/service related configurations and is selected from the configurational database.

- TST_STAFF (DIMENSION Table): This table is used to displays configurations picked from the configurational database.

4.2. DB creation and data loading considerations

Subscriber data selected to validate the virtual data warehouse system contains millions of subscribers. Generally, data retains in the OLTP system for six months. All CDRs are processed from each terminal/exchange into the mediator of billing system for provisioning of

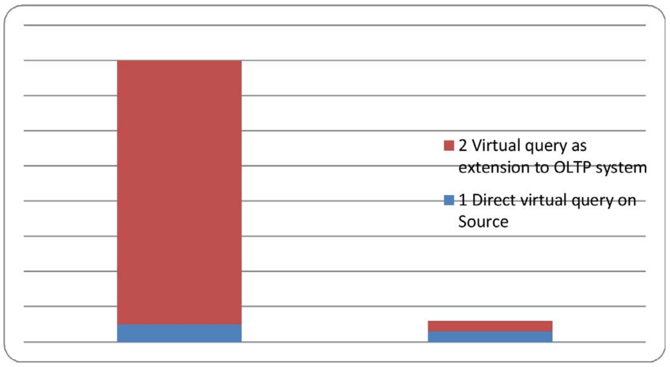


Fig. 10. Response time to CPU usage comparisons.

Table 2
Comparisons Key Database Performance Statistics.

S/No	Key Parameters	Difference
1	CPU used when call started (%)	2 vs 2
2	CPU used by this session (%)	12 vs 4
3	session connect time (seconds)	54781 vs 68
4	logical read total (bytes)	11757445 vs 581
5	physical read total (bytes)	41125 vs 69452

payments and allied charges. First, we created two tablespaces: TST_STORE for core data and TST_INDEX for generating indexes. The sole reason for creating these separate tablespaces was to convert normal indexes into a bitmap indexing for getting optimum performance. The creation of multiple partitions that hold logical parts of a partitioned table boosts up the overall performance of a virtual system as its decreases the degree of scans and CPU usage. However, the bitmap indexing is only applied for selection operations. For DML operations like update, delete and insert, we used ordinary indexes. The partitioned indexes are created for enhanced performance to build virtual data cubes in our SDW system.

The **fact** table TST_CUST_INVOICE acts as a partitioned materialized view. It is set on real-time synchronization with other system interfaces based on “oncommit” method type refresh. Whenever any DML command is executed on an interface, then the same changes are automatically refreshed and reflected in tst cust_invoice materialized view. The code listed in Fig. 6 is used to create partitioned materialized view.

All other dimension tables (i.e. tst_staff, tst_region, tst_exchange) are treated as a *simple non-partitioned materialized view* set on real-time synchronization with other system interfaces based on “oncommit” method type refresh because these all tables hold configurations from different system interfaces. Creation of fast indexing (bitmap) is a key to performance in virtual warehouse system to minimize I/O overhead because storage space is much cheaper than CPU power. We opted for this tradeoff as per unit cost of storage media is comparatively cheaper than per unit CPU clock-tick (second). This approach is also in line with the DBA best practices to store bitmap indexes on a separate storage media. Therefore, in our proposed approach, we have opted to compromise storage in lieu for better performance. The code to generate bitmap indexes is shown in Fig. 7.

4.3. *Performing simulation on VDWH models*

The query illustrated in Fig. 8 when executed on the SDW test data warehouse takes nearly one minute of processing time to fetch monthly sum of payments made within a region and is grouped by month, region and exchange respectively. Whereas, the query shown in Fig. 9 when executed on a normal OLTP environment takes almost 15 min of processing time in addition to the increased load on the existing working OLTP system.

5. Results and discussion

Table 1 shows comparison of processing time and percentage CPU usage of direct virtual query on heterogeneous sources versus virtual query on our proposed model. Direct virtual query takes approximately 15 min and 61% of CPU resources whereas virtual query based on materialized view takes approximately one minute. The remarkable difference of 14 min in the processing time is due to the use of materialized view in the virtual query that refreshes “oncommit”; thus, decreasing load on the system by decreasing table scan on source tables due to the handy presence of data in materialized view. The same statistics are depicted in Fig. 10 in the form of a stacked-bar graph. Going into further detailed statistics, the comparison as shown in Table 2 indicates significant difference between querying directly on OLTP system versus query aggregates using materialized views as evident from the statistics obtained for direct query versus optimized query.

6. Conclusion and future work

In this paper, we have discussed a framework for efficient data retrieval in virtual data warehouse environment. For this purpose, we have developed variants for the proposed framework and have drawn a comparative analysis based on the output produced by the original proposed framework and its variants. Techniques used for experimentation in this research study include: partitioned materialized views, index performance optimization techniques, query rewrite in materialized views, use of analytic functions, use of sub-queries and enabling parallel execution etc. While performing validation of the proposed framework in a virtual environment on the SDW test data for billing and customer care system, it revealed that the use of virtual query processing suffers from certain drawbacks including extra load on the OLTP system and substantial increase in the overall query response time. Therefore, modeling direct queries on OLTP exhibits low performance due to the involvement of data processing and I/O overheads. Whereas, the hybrid architecture of virtual BI solution, as an extension to a working OLTP system, shows better performance by deploying partitioned materialized views and employing a combination of batch and real-time synchronization techniques as well as other performance management techniques. Thus, the goal to achieve further improvement of query performance in a virtual data warehousing environment built on normal transactional system creates space for further research opportunities. Likewise, implementing VDW environment in the distributed systems can be an additional prospective future work to enhance efficiency while minimizing maintenance and operational costs on such platforms. Similarly, comparative experimentation of working of different types of database products in distributed environment is also a potential alternative in implementation of virtual systems.

References

Aji, A., Wang, F., Vo, H., Lee, R., Liu, Q., Zhang, X., et al. (2013). Hadoop GIS: A high performance spatial data warehousing system over mapreduce. *Proceedings of the VLDB Endowment*, *6*(11), 1009–1020.

Ashadevi, B., & Balasubramanian, R. (2009). Optimized cost effective approach for selection of materialized views in data warehousing. *Journal of Computer Science & Technology*, 9.

Bara, A. (2010). Solutions for improving data extraction from virtual data warehouses. *Database Systems Journal*, *1*(1), 27–36.

Chan, I. (2005a). *Oracle database–performance tuning guide, 10 g release 2(10.2), vol 474*, Redwood City: Oracle Corporation, B14201–B14211.

Chan, J. O. (2005b). Optimizing data warehousing startegies. *Communications of the IIMA*, *5*(1).

DeWitt, D. J. D., Madden, S., & Stonebraker, M. (2006). *How to build a high-performance data warehouse*. Available: http://db.lcs.mit.edu/madden/high_perf.pdf [Accessed: June 2011].

Geng, Y., & Zhang, C. (2009). Study on the optimization method of select query sentence in standard SQL language. *Computer and Information Science*, *2*(1), p121.

Lane, P. (2003). *Data warehousing guide 10 g release 1 (10.1)*. B10736-01 [01 Agustus 2011. URL: http://download.oracle.com/./b14223.pdf].

Levene, M., & Loizou, G. (2003). Why is the snowflake schema a good data warehouse design? *Information Systems*, *28*(3), 225–240.

Lungu, I., Bâra, A., & Diaconita, V. (2006b). Building analitic reports for decision support systems-aggregate vs. *Analyic Functions, Economic Informatics*, *1-4*, 17–20.

Lungu, I., Velicanu, M., Bara, A., Diaconita, V., & Botha, I. (2008). Practices for designing and improving data extraction in a virtual data warehouses project. *Proceedings of ICCCC*, 369–375.

MOÇKA, B., & Daniel, L. E. K. A. (2013). Optimizing data warehouse. *Science Innovation New Technology*, 51.

Stonebraker, M., & Cattell, R. (2011). 10 rules for scalable performance in 'simple operation' datastores. *Communications of the ACM*, *54*(6), 72–80.

Thakur, G., & Gosain, A. (2011). A comprehensive analysis of materialized views in a data warehouse environment. *International Journal of Advanced Computer Science and Applications (IJACSA)*, *2*(5).

Yagoub, K., Belknap, P., Dageville, B., Dias, K., Joshi, S., & Yu, H. (2008). Oracle's SQL performance analyzer. *IEEE Data Engineering Bulletin*, *31*(1), 51–58.