Query Dispatching Tool Supporting Fast Access to Data Warehouse

Anmar Aljanabi[1], Alaa Alhamami[2], and Basim Alhadidi[3]
[1]Computer Science Department, University of Technology, Iraq
[2]Computer Science and Informatics College, Amman Arab University, Jordan
[3]Computer Science Department, Al-Balqa' Applied University, Jordan

Abstract: *Data warehousing hastens the process of retrieving information needed for decision making. The spider web diversity between both end-users and data marts increases traffic, load, and delay in accessing the requested information. In this research, we have developed a query dispatching tool facilitating the access to the information within data marts, eventually data warehouse in fast, and an organized fashionable way. The dispatching tool takes the query, analyzes it, and decides which data mart as a destination that query should be forwarded to. This research is based on Ralph Kimball's methodology. The results show that the dispatching tool reduces the check time spent in the data dictionary within a logical side of the data warehouse deciding the intended data mart and hence, minimizing execution time.*

Keywords: *Data warehouse, metadata, query dispatching tool, execution time.*

Received October 12, 2010; accepted May 24, 2011; published online March 1, 2012

1. Introduction

Global corporations today compete with each other in satisfying customers through introducing better services, and performance in the most cost effective way. Millions of transactions had been captured through the years in enterprises producing enormous amounts of raw data spread among different distant departments [1, 4]. Earlier, operational database systems also, known as On-Line Transactional Processing (OLTP) systems were used for decision making resulting non accurate decisions. This drawback urged database experts to construct database in a way to support analysis and decision making and that's where the Data Warehouse (DW) came from [2, 3].

Several different definitions have been given of data warehouse, according to Barry Devlin, IBM Consultant "a data warehouse is simply a single, complete and consistent store of data obtained from a variety of sources and made available to end users in a way they can understand and use it in a business context" [2, 6]. Data warehouse is segregated from transactional databases and contains consistent cleansed data. Modeling is an important step in a data warehouse design process including logical data modeling, physical data modeling, and metadata management satisfying rapid information retrieval and ad hoc formulations [9]. Metadata is an enabling technology that supports the user interface to warehouse such as R/OLAPXL (ROLAP front-end tool) which makes use of metadata to display data warehouse tables and fields. Access tools that utilize metadata are a powerful evolution of a warehousing process. In this research, we present a front-end Query Dispatching Tool (QDT) that supports fast access to the data warehouse exploiting metadata. This tool redirects queries to the intended destination in a definitive way, and the most important thing is that end-users need only a cursory knowledge about the warehouse architecture itself. An implementation along with the Graphical User Interface (GUI) is presented. In the next section, we briefly look at the data warehouse model. Section 3 examines the tool interface and implementation. The QDT implementation and results are given in section 4 and conclusions in section 5.

2. The Warehouse Model

Data warehouse is a read-intensive database, modeled according to enterprise requirements within different environments. It can be seen as building blocks involving a number of essential components glued together consisting of source data, data staging, data storage, and information delivery [8]. The metadata is used to combine and manage these blocks. Different data warehouse architectures had been conducted. Among the most popular is Kimball's methodology also, known as Bottom-Up approach as shown in Figure 1 [5, 9]. The model shows a back-end technological component, which focuses on data at sources and stages the required data prior to extraction, transformation, then loading to already structured data warehouse tables. Furthermore, a front-end component interested with user's access to data resides within warehouse, in other words, front-end supports

warehouse browsing, query managements and monitoring activity. The metadata plays a major role in data warehouse technology. It contains information such as extraction frequencies, extraction methods, indexes, and algorithms used in integration and conversion of data into a warehouse repository. Metadata helps end-users to use their own terminologies to find the information. In other words, metadata can be seen as a navigational map to the data warehouse [7, 8].

The proposed tool requires from end-users nothing but a cursory knowledge about the data warehouse inner structure. SQL queries are auto generated leaving no room for errors concerning the requested information. Query dispatching tool uses a copy of metadata to ease the access to the investigated information. It also, shows a single image of the data warehouse. The most important is that the tool offers accuracy, and minimizes time for reaching information.

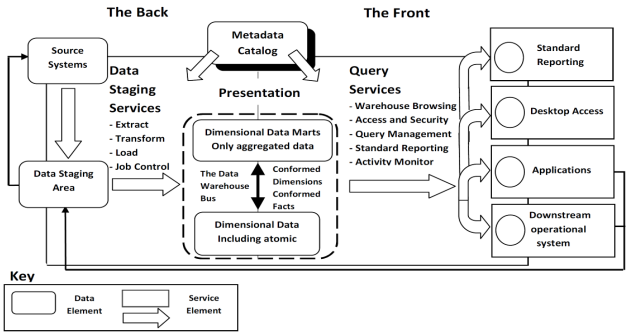

Figure 1. High level technical architecture for data warehouse (adapted from Kimball et al. [5]).

3. Interface Design and Implementation

3.1. Introduction

This section introduces the design and implementation of the QDT user interface. This tool is designed to work with very simple type of queries. Bus modelling is adopted for the warehouse design. Data Marts (DMs) are integrated using shared dimension, which is the time dimension. The data dictionary used within this tool contains three major tables describing DMs used and their tables along with their contained fields. Furthermore, metadata contains a description of the method that QDT uses to direct queries after investigation and knowing what data needs to be extracted.

3.2. Interface Design

The QDT uses the concepts of tree and tabs that make the interface user friendly. It is classified into two tabs and three modules. The layout is shown in Figure 2. A brief look at the main parts of the interface helps to set up the discussion of the data warehouse access method.

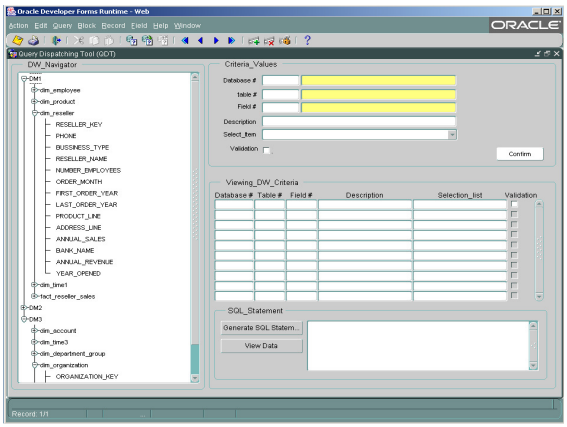

Figure 2. Query dispatching tool graphical user interface.

3.2.1. Data Selection

This tab consists of three modules as shown below:

- *DW_Navigator's Module:* This is an essential module in a QDT in which users navigate through the database system. It excels in visualizing the entire data warehouse system. Users only have to surf through the tree and determine data mart(s), table(s), and field(s).
- *Criteria_Values Module:* After the criteria are defined through the DW_Navigator. Information on the specified criteria will be shown referring to the data mart (database) number, name of the selected table, field's name, and description. Still one thing that is the value of the field of criteria needs to be specified representing the values that QDT will use to retrieve the information based on it. This is done throughout the select_item list retrieving data automatically from distant data marts based on database links, which are configured internally with proper authentication for each of the remote DMs. Validation of these criteria and their chosen values is done through confirm button to commit these values to a database table known as criteria.
- *Viewing_DW_Criteria Module:* In this module, all the criteria selected earlier are exhibited here to show users what their criterion are, which contributes to constructing the final SQL SELECT statement. This region also, contains SQL_Statement sub-module that helps in viewing the SQL command before execution, and it contains three items. The items are:

  1. Generate SQL statement button.
  2. A non-editable text area (script area) displaying data warehouse query, which is generated internally and automatically all the way through the proper functions and procedures.
  3. View data button that executes the script and displays results in the system outputs tab.

It is worth mentioning, criteria as well can be reconfigured from this module.

3.2.2. System Outputs

This second tab does no more than showing end-user the retrieved information based on the SQL query. It views the information, whether it is drill down or drill across at the same tab.

3.2.3. Query Dispatching Tool Services

This query dispatching tool also, provides some of the built-in functions which come with oracle forms with the ability to copy, paste, enter/execute/cancel query, insert/remove/lock records, help, and menus for other standard activities.

3.3. Interface Implementation

The interface of the QDT excels in simplicity. Once again, it uses tree and tabs concepts making the layout user friendly and easy to handle. This tool is implemented using oracle forms developer, which is a powerful front-end application from oracle developer suite 10g, and is tested using oracle database as a back-end. The proposed tool requires no more than Internet Explorer 5 (IE5) or higher version from client's side. Another application known as Oracle Containers for J2EE (OC4J) is also, needed, and is used to run oracle forms and examine them using Java Virtual Machine (JVM), which already exists in all systems allowing forms to run in any operating system, thus removing the need for oracle 10g application server, which is used for this very same purpose.

3.4. Example

Briefly look at an example of a very simple query concerning drill down process using QDT interface. *Example 1:* Warehouse attributes: All the fields in the fact_finance data marts.
Warehouse conditions: fiscal year='2004' & Department_Group_Name='Executive General and Administration' & Organization_Name= 'Canadian Division'
Query dispatching tool SQL statement:

```
Select To_ Char('Product_Key'),
To_Char(Time_Key),
To_Char('Reseller_Key'),
To_Char('Employee_Key'),
To_Char('Customer_Key'),
To_Char('Currency_Key'),
To_Char('Sales_Territory_Key'),
To_Char('Sales_Order_Number'),
To_Char('Sales_Order_Line_Number'),
To_Char('Sales_Line_Order_Number'),
To_Char('Order_Quantity'),
To_Char('Unit_Price'),
To_Char('Extended_Ammount'),
To_Char('Product_Standard_Cost'),
To_Char('Total_Product_Cost'),
To_Char('Tax_Ammount'), To_Char('Freight'),
To_Char('Carier_Track_Number'),
To_Char('Customer_Po_Number'),
To_Char(Organization_Key),
To_Char(Department_Group_Key),
To_Char(Account_Key), To_Char(Ammount),
From Fact_Finance@Rdm3
Where Time_Key In (Select Distinct Time_Key From Dim_Time3
@Rdm3 Where Fiscal_Year= '2004' )
And Department_Group_Key In (Select Distinct
Department_Group_Key From Dim_Department_Group @Rdm3
Where Department_Group_Name = 'Executive General And
Administration' )
And Organization_Key In (Select Distinct Organization_Key From
Dim_Organization @Rdm3
 Where Organization_Name='Canadian Division');
```

Figure 3 shows the information required from the user. The attributes of the fact table which is the target for analysts are generated automatically based on the data mart selection. Warehouse conditions are formed according to the criteria selection. The criteria are stored within a table named criteria. The metadata contains information about the data marts, tables, descriptions, attributes, db_links, mechanisms of how to get to the information remotely, and any other necessary information regarding the data warehouse. After end-users select their criteria they will have to validate and confirm it through confirm button and then press on generate SQL statement button for the auto generation process leaving no room for errors in query formation. Last but not least view data button displays the requested data in system output tab as shown in Figure 4.

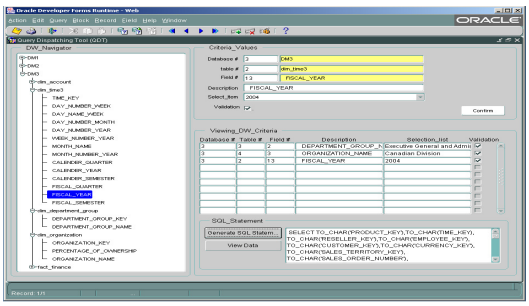

Figure 3. Screen of input for example 1.

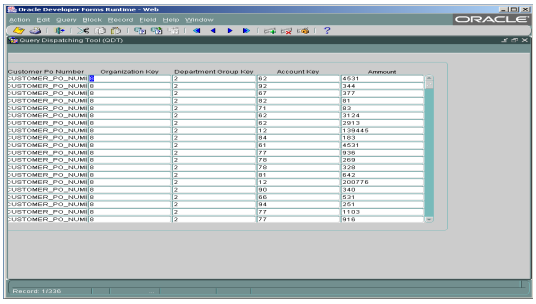

Figure 4. Screen of output for example 1.

4. QDT Implementation

4.1. Introduction

This section presents implementation of QDT. Using oracle forms developer suite 10g environment and PL/SQL language as a front-end and oracle 10g database as back-end. Performance measurement is

also, presented concerning execution time on both conventional and developed QDT data warehouse systems. A detailed technical comparison between performance measurements as well will be made. In section 4.2 performance criteria will be investigated, parameters' discipline in 4.3, and finally experimental scenarios are presented in 4.4 considering both drill down and drill across processes.

4.2. Performance Criteria

To have a good comparison between the old system and QDT one, it is important to select a fixed criterion of comparison. In this research, fetch and check times are selected deducing execution time for a query as the base of comparison and are evaluated by using the following equation:

$$ET_{(t)} = CT_{(t)} + FT_{(t)} \quad (1)$$

where ET_t, CT_t are total values for execution and check times respectively, and finally FT_t is the sum of all fetch time values. The following is a presentation of comparison criterion for this study:

- Query dispatching tool consists of a replica for data dictionary, specifically look up tables about database tables. Checking every single record in metadata will increase the check time in order to find the intended table in the SQL command, whether it exists or not. In other words, check time within a data dictionary is evaluated by the following equation:

$$CT_{(t)}=CT_{(i)}+CT_{(i+1)}+.....+CT_{(n-1)}+CT_{(n)} \quad (2)$$

  where CT_t is the total check times, CT_i is a single check time, and ($i=1,2,...,n$) where n is the iteration check time.
- After checking is done and table objects are confirmed fetching its data phase begins, the following equation evaluates the total fetch time:

$$FT_{(t)}=FT_{(i)}+FT_{(i+1)}+.....+FT_{(n-1)}+FT_{(n)} \quad (3)$$

where *FTt,* and *FTi* mean the total fetch time values and a single fetch time in a consecutive manner, and ($i=1,2,...,n$) where n is the number of the table being fetched.

4.3. Parameters Discipline

Many parameters in the test environment interact, controlling them would help us make minimum effects on test setup:

- *Session Load:* Only SQL statement is executed in the database, within only one session. Connecting to the database guarantees full capabilities of the database are concentrated on executing the SQL command.
- *Connectivity:* It is important to ensure that connectivity between dispatcher database where QDT resides and the three DMs from the other side is flawless and authenticated properly throughout Db_links.
- *Time Estimation:* This research is interested in reducing execution time for the SQL statement via the check times in the data dictionary. Oracle issues an explain plan for a detailed execution every time read/write operation is needed and an optimizer chooses the best plan to execute the query before the real execution. It gives many details like IO disks, nested loops, joins, union, view, sort, connect by, etc., but there is no information regarding how much elapsed time has been taken to find the table containing the investigated information within the data dictionary.

In order to overcome this drawback, estimation for both fetch time and check time values has been developed based on some solid facts regarding the nature of data warehouse and RDBMS, they are:

- Since analysts' target has always been the fact table meaning that records' number within fact tables is very high, making the data block size large-which is interesting to analyst who view/fetch data (no inserts/updates), leaving us to reach a conclusion that the nature of the transaction is not a big deal in this case. In other words, leave the data within one data block or few as possible and the heavy load for the transaction on the database itself is acceptable again no updating but only viewing.
- Once the query is executed, execution plan is lifted to buffer cache to speed up the process minimizing fetch time for the next query runs, considering that buffer cache may choose a different actual execution plan when the statement is executed and there are several possible reasons for this to happen including: bind variables, optimizer session setting, and System Global Area (SGA) cache so, that times are estimated considering the minimal level for fetch time and check time, to calculate execution time.

4.4. Experimental Scenarios

This section will go through a variety of experiments showing advantage of the performance of QDT over the traditional one. To have a full view of the nature of the evaluation, the end-user will have to construct the SQL SELECT statement and monitors: check time values within a data dictionary, total fetch times for data which resides within data blocks in the physical side of the DB architecture, make a comparison involving the total fetch times between the old technique and the proposed one, and finally, calculate execution time that is an accumulation of both fetch and check times. Three experimental scenarios had

been taken into consideration DM1, DM2, DM3. Moreover, drill across scenario between these data marts has been conducted.

4.4.1. Drill Down

4.4.1.1. First Scenario

In this scenario, end-user constructs SQL statement for requesting data. The data resides within DM1. A comparison between execution time values of dimensions and the subject area for data mart one is shown in Figure 5.

Figure 5. Logarithmic pivot chart for execution times using conventional and proposed QDT.

A comparison between check times during different execution stages illustrated in Figure 6.

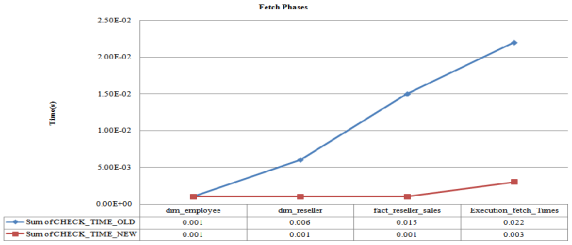

Figure 6. A comparison between old and proposed QDT check times.

Execution time of the same SQL statement for both old and new systems shows a slight difference because requested information resides within DM1 depending on the position of the tables within the data dictionary.

4.4.1.2. Second Scenario

In this scenario, investigated information resides within DM2. Comparison between conventional and QDT in terms of check, fetch, and execution times within the data dictionary is shown in Figures 7 and 8.

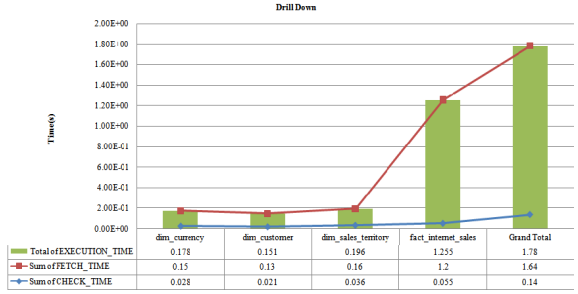

Figure 7. Pivot chart of fetch, check, and execution times for traditional drill down process.

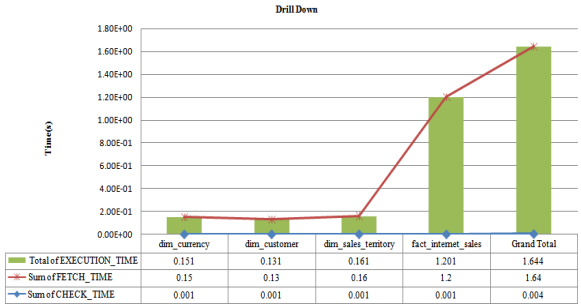

Figure 8. Pivot chart of fetch, check, and execution times for QDT drill down Process.

Comparison between old and QDT in terms of fetch time within metadata is shown in Figure 9.

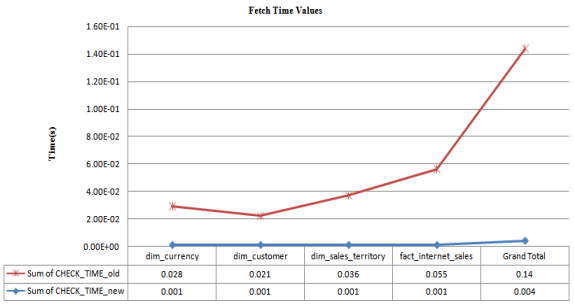

Figure 9. A comparison of check time values between traditional and QDT.

4.4.1.3. Third Scenario

End-user asks for data kept within data DM3. System responses to execution time for both conventional and the proposed QDT is given in a graph in Figure 10.

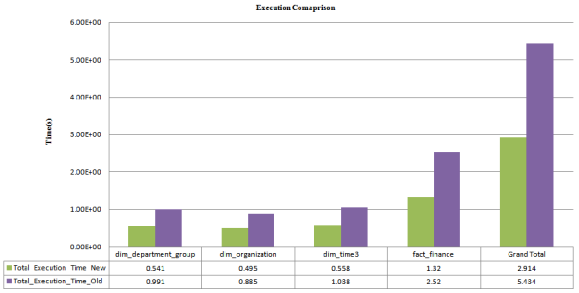

Figure 10. A comparison of execution time values between traditional method and QDT.

A comparison between the two systems showing a drastic change in check time within the data dictionary is exhibited in Figure 11.

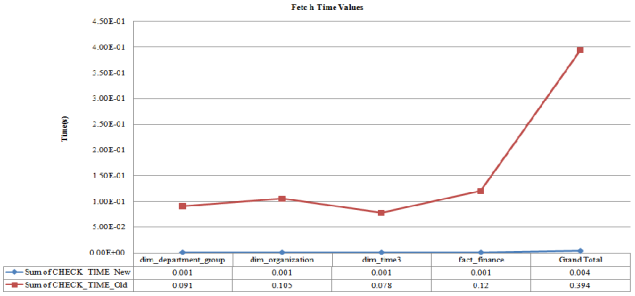

Figure 11. A comparison of check time values within data dictionary.

4.4.2. Drill Across

Fetching data from more than one database is a common thing, but in a data warehouse it is quite a bit of challenge. Figure 12 exhibits check, fetch, and execution time values of drilling across between first, second, and the third data marts.

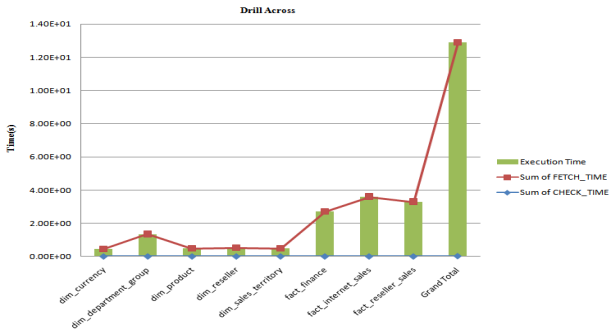

Figure 12. Shows check, fetch, execution times of a drill across process between DM1, DM2, and DM3 using QDT.

The following is a detailed statistical pivot table for the values of check, fetch, execution, and grand total for each one of DM1, DM2, and DM3 in table 1. generated by Excel 2007.

Table 1. A detailed statistical pivot table gives check, fetch, and execution time values for drill across process between DM1, DM2, and DM3.

Row Labels	Sum of Check_Time	Sum of Fetch_Time	Sum of Execution_Time
Dim_currency	0.003	0.45	0.453
Dim_department_group	0.003	1.35	1.353
Dim_product	0.003	0.48	0.483
Dim_reseller	0.003	0.51	0. 513
Dim_sales_territory	0.003	0.48	0.483
Fact_finance	0.003	2.7	2.703
Fact_internet_sales	0.003	3.6	3.603
Fact_reseller_sales	0.003	3.3	3.303
Grand Total	0.024	12.87	12.894

5. Conclusions

Decision support systems not only need a data warehouse as a central repository, but also, front-end techniques that facilitate accessing and retrieving information. Fast access to a data warehouse plays a major role for decision makers. Experimental result in this research have proven that a consumed fetch time for requested information has been minimized within the metadata, thus reducing execution time of end-user queries. The accuracy in reaching investigated information within the data dictionary has a superior performance to the conventional system, because data is accessed accurately and directly.

The QDT takes an end-user query, analyzes it, and redirects it to a proper data mart as a destination. The graphical user interface makes the architecture of the data warehouse understandable for end-users. This research can be seen as an avenue for some innovative futuristic ideas. Contrary to Data Staging Area (DSA) as a data warehouse phase where the background processes work, a phase named Query Dispatching Area (QDA) can be built supporting front-end applications. Furthermore, an agent can be developed to tune to affecting SQL statements such as finding a better explain plan effecting fetch time, and eventually execution time.

References

[1] Adhikari A., Ramachandrarao P., Prasad B., and Adhikari J., "Mining Multiple Large Data Sources," *The International Arab Journal of Information Technology*, vol. 7, no. 3, pp 241-249, 2010.

[2] Başaran B., "A Comparison of Data Warehouse Design Models," *MSc Thesis,* Computer Engineering, Atilim University, Turkey, 2005.

[3] Bontempo C. and Zagelow G., "The IBM Data Warehouse Architecture," *Communications of the ACM*, vol. 41, no. 9, pp 38-48, 1998.

[4] Hamad M., "Building Data Warehouse For Decision Support Systems," *PhD Dissertation,* Informatics Institute for Postgraduate Studies at Iraqi Commission for Computer and Informatics, Iraq, 2004.

[5] Kimball R., Ross W., and Thornthwaite W., *The Data Warehouse Toolkit: Expert Methods for Designing, Developing, and Deploying Data Warehouse*, Wiley, New York, 1998.

[6] Kumar P., "The Data Warehousing: Continuing the Evolution," *The Data Warehousing Institute World Conference*, USA, http://www.docstoc.com/docs/23482116/The-Data-Warehousing, 2009.

[7] Nagabhushana S., *Data Warehousing: OLAP and Data Mining*, New Age International Ltd. Publishers, India, 2006.

[8] Ponniah P., *Data Warehousing Fundamentals: A Comprehensive Guide for IT Professionals*, John Wiley & Sons, Inc., New York, 2001.

[9] Wu L., Miller L., and Nilakanta S., "Design of Data Warehouses Using Metadata," *Information and Software Technology*, vol. 43, no. 2, pp. 109-119, 2001.

Anmar Aljanabi is presently assistant lecturer at Computer Science Department, University of Technology, Iraq. He earned his BSc in computer science from Baghdad University, Iraq in 2003, and his MSc in computer science from Al-Balqa' Applied University, Jordan in 2010. His research interest is in data warehouse, and image processing.

Alaa Alhamami is presently professor of database security and dean of Computer Science and Informatics College, Amman Arab University, Jordan. He is a reviewer for several national and international journals and a keynote speaker for many conferences. He is supervising a lot of PhD, MSc, and Diploma theses. His research is focused on distributed databases, data warehouse, data mining, cryptography, steganography, and network security.

Basim Alhadidi is presently an associate professor at The Department of Computer Science, Al-Balqa' Applied University, Jordan. He earned his PhD in 2000, in Engineering Science (Computers, Systems and Networks). He received his MSc in 1996 in engineering science (Computer and Intellectual Systems and Networks). He published many research papers in many topics such as: computer networks, image processing, and artificial neural networks. He is a reviewer for several journals. He was appointed in many conferences as keynote speaker, reviewer, track chair and track co-chair.