UNIVERSITY OF KASHMIR

www.kashmirUniversity.net

Department Of Computer Sciences

Course No:MCA-305-DCE

Data Warehousing

Compiled by: Miss. Qurat ul Ain

2.1. Basics of Dimensional Modeling:

Dimensional modeling is one of the methods of data modeling, that help us store the data in such a way that it is relatively easy to retrieve the data from the database. All the modeling techniques give us different ways to store the data. Different ways of storing data gives us different advantages. For example, ER Modeling gives us the advantage of storing data in such a way that there is less redundancy. Dimensional modeling, on the other hand, give us the advantage of storing data in such a fashion that it is easier to retrieve the information from the data once the data is stored in database and this is the reason why dimensional modeling is used mostly in data warehouses built for reporting. Dimensional model is the underlying data model used by many of the commercial OLAP products available today in the market. In the dimensional model, we store all data in two types of tables viz: Fact table a.k.a (measure table) and Dimension table. The Fact table contains the main facts or measures and links to many dimension tables through foreign keys.

Now that we know in dimensional modeling everything is divided in 2 distinct categories - dimension or facts(measures). Anything we try to model, must fit in one of these two categories. So let's us say, we want to store information of how many burgers and fries are getting sold per day from a single KFC outlet, we will have to first classify this data in dimension and facts. And then we will have two different categories of tables i.e. Dimension table and Fact table(measure table). Let's us try to work on a practical scenario and see how to identify dimensions and facts to model the scenario.

Step by Step Approach to Dimensional Modeling:

Consider the business scenario for a KFC fast food chain. The business objective is to create a data model that can store and report number of burgers and fries sold from a specific KFC outlet per day. To model this scenario the whole modeling approach is divided in 5 steps as depicted below.

Step 1: Identify the dimensions

Dimensions are the object or context. That is - dimensions are the 'things' about which something is being spoken. In the above statement, we are speaking about 3 different things - we are speaking about some "food", some specific KFC "store" and some specific "day". So we have 3 dimensions - "food" (e.g. burgers and fries), "store" and "day". Burgers and fries are 2 different members of "food" dimension. We will have to create separate tables for separate dimensions.

Step 2: Identify the measures

Measures are the quantifiable subjects and these are often numeric in nature. In the above statement, the number of burgers/fries sold is a measure. Measures are not stored in the dimension tables. A separate table is created for storing measures. This table is called Fact Table.

Step 3: Identify the attributes or properties of dimensions

Now that we have decided we need 3 tables to store the information of 3 dimensions, next we need to know what are the properties or attributes of each dimension that we need to store in our table. This is important since knowing the properties let us decide what columns are required to be created in each dimension table.

As you might have guessed, each dimension might have number of different properties, but for a given context, not all of them are relevant for us. As an example, let's take the dimension "food". We can think of so many different attributes of food - e.g. names of the food, price of the food, total calories in the food, color of the food and so on. But as I said, we need to check which of these

attributes are relevant to us - that is - which of these attributes are required for reporting on this data. As for the given statement above, we just need to know only one attribute of the "food" dimension - i.e. name of the food (burger or fries). So the structure of our food dimension will be rather easy. **Like below: Food**

KEY	NAME
1	Burger
2	**Fries**

Similarly, the structure of our store and day dimensions will be like this: **Store**

KEY	NAME
1	Store 1
2	Store 2
…	…

Day

KEY	DAY
1	01 Jan 2012
2	02 Jan 2012
3	03 Jan 2012
…	…

As I said, this is really a super simplified structure as we are only interested about basic attribute. But in a complex scenario, we might need to add tens or hundreds on columns to each dimension table if those attributes are required for reporting.
Also note, each dimension table above has a key column. Key is a not null column and it's a unique column which helps us identify each record of the table.

Step 4: Identify the granularity of the measures

Granularity refers to the lowest (or most granular) level of information stored in any table. if a table contains sales data for each and every day, then it has a daily granularity and if a table contains total sales data for each month, then it has monthly granularity. Let us take this example, if we say, a specific KFC store sells 200 burgers on a specific day and 5000 burgers on a specific month, then in the first case the granularity of our information is daily whereas in the second case the granularity of our information is monthly. It is important to identify the granularity of the information required. In our case, we need the information on a daily basis. But if our requirement was "To store how many burgers and fries are getting sold from a specific KFC outlet per month", required granularity would have changed to monthly from daily. **Why is this important**? This is important because this information help us decide what columns are required to be stored in our fact table.

For example, since in our case the granularity is food getting sold per store per day, we will need to add key columns from food/store and Day dimensions to the Fact table like figure1 as shown below:

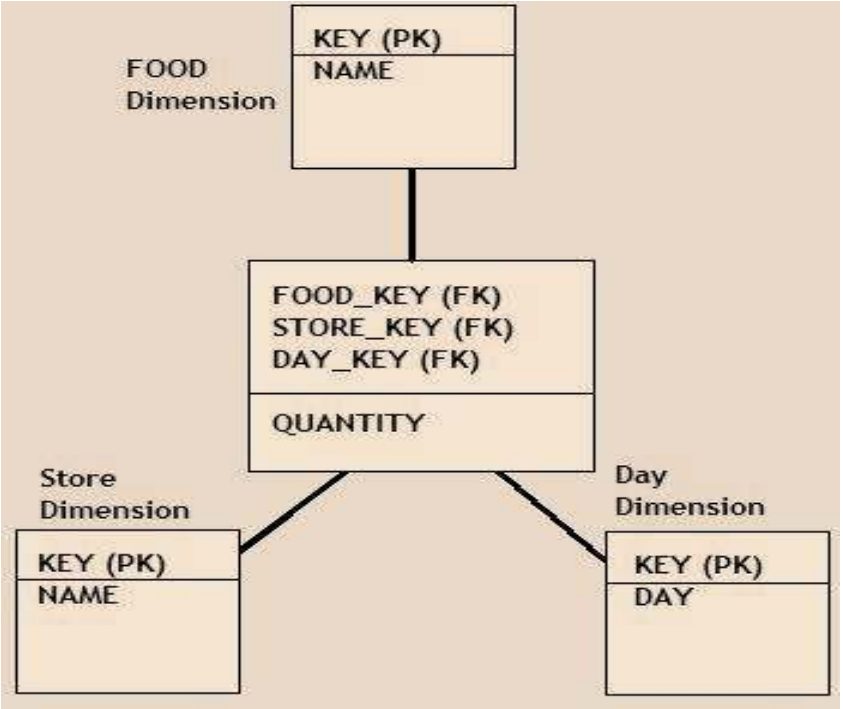

Figure 1

Step 5: History Preservation (Optional)

If you have followed all the above steps till now, we have then designed 3 dimension tables and 1 fact table. The fact table stores the "number" of food sold in "Quantity" column against a given store, food and day columns. These store/food/day columns are basically foreign key columns of the primary keys in respective dimension tables. This kind of schema is called "Star Schema" because of the star like formation.

The above schema is certainly capable of storing all the information that we intended to store in our dimensional modeling. However, there is a subtle problem. The problem is, we are not sure what would happen if any attribute of any dimension get changed in the future. Let's say KFC decided to change the name of the food "burger" to "jumbo burger" for some promotional reason. If they do that, they would update the burger record in the dimension table and update the name to "jumbo burger". So far so good. But the problem is we will lose the old information once they change the

name. This means, after they change the name if you look at the data in the model, you will not know that until now the product was called "burger" and not "jumbo burger". *This is a problem if one of the objectives of your modeling is to store history*. Fortunately, this can be solved by designing the dimension tables as "slowly changing dimension". Identify which dimensions are slowly changing (or fast changing or unchanging) is the last and final step of modeling.

2.2. Use of Case Tools

Many case tools are available for data modeling. You can use these tools for creating the logical schema and the physical schema for specific target database management systems (DBMSs). You can use a case tool to define the tables, the attributes, and the relationships. You can assign the primary keys and indicate the foreign keys. You can form the entity-relationship diagrams. All of this is done very easily using graphical user interfaces and powerful drag-and-drop facilities. After creating an initial model, you may add fields, delete fields, change field characteristics, create new relationships, and make any number of revisions with utmost ease. Another very useful function found in the case tools is the ability to forward-engineer the model and generate the schema for the target database system you need to work with. Forward-engineering is easily done with these case tools

2.3. Schema:

Schema is a logical description of the entire database. It includes the name and description of records of all record types including all associated data-items and aggregates. Much like a database, a data warehouse also requires to maintain a schema. A database uses relational model, while a data warehouse uses Star, Snowflake, and Fact Constellation schema. The description of each of the schema are given below.

2.3.1. The STAR Schema

The star schema architecture is the simplest data warehouse schema. It is called a star schema because the diagram resembles a star, with points radiating from a center. The center of the star consists of fact table and the points of the star are the dimension tables. Usually the fact tables in a star schema are in third normal form(3NF) whereas dimensional tables are de-normalized. Despite the fact that the star schema is the simplest architecture, it is most commonly used nowadays and is recommended by Oracle.

Features of Star Schema

- Each dimension in a star schema is represented with only one-dimension table.
- This dimension table contains the set of attributes.
- The following diagram shows the sales data of a company with respect to the four dimensions, namely time, item, branch, and location.

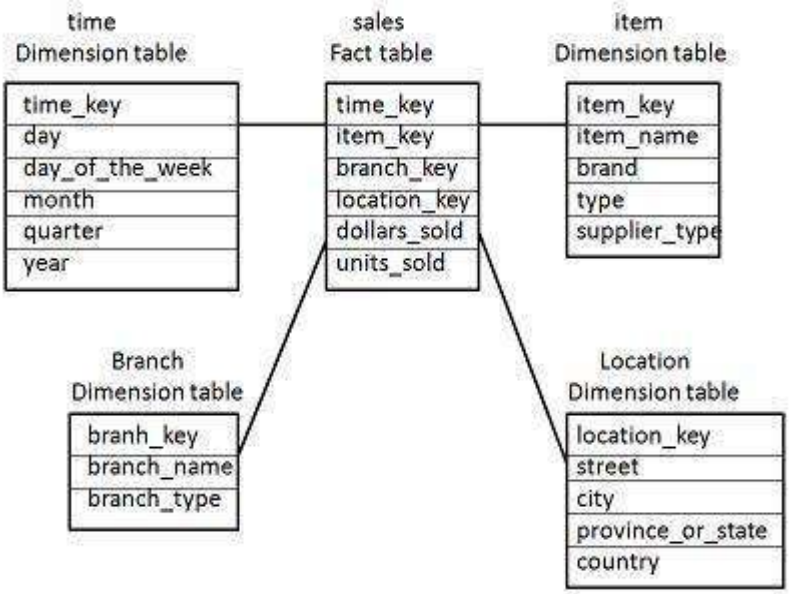

Figure 2

- There is a fact table at the center. It contains the keys to each of four dimensions.
- The fact table also contains the attributes, namely dollars sold and units sold

Note: Each dimension has only one dimension table and each table holds a set of attributes. For example, the location dimension table contains the attribute set {location_key, street, city, province_or_state,country}. This constraint may cause data redundancy. For example, "Vancouver" and "Victoria" both the cities are in the Canadian province of British Columbia. The entries for such cities may cause data redundancy along the attributes province_or_state and country.

2.3.2. Snowflake Schema

The snowflake schema is an extension of the star schema, where each point of the star explodes into more points. In a star schema, each dimension is represented by a single dimensional table, whereas in a snowflake schema, that dimensional table is normalized into multiple lookup tables, each representing a level in the dimensional hierarchy. For example, the Time Dimension that consists of 2 different hierarchies:

1. Year → Month → Day
2. Week → Day

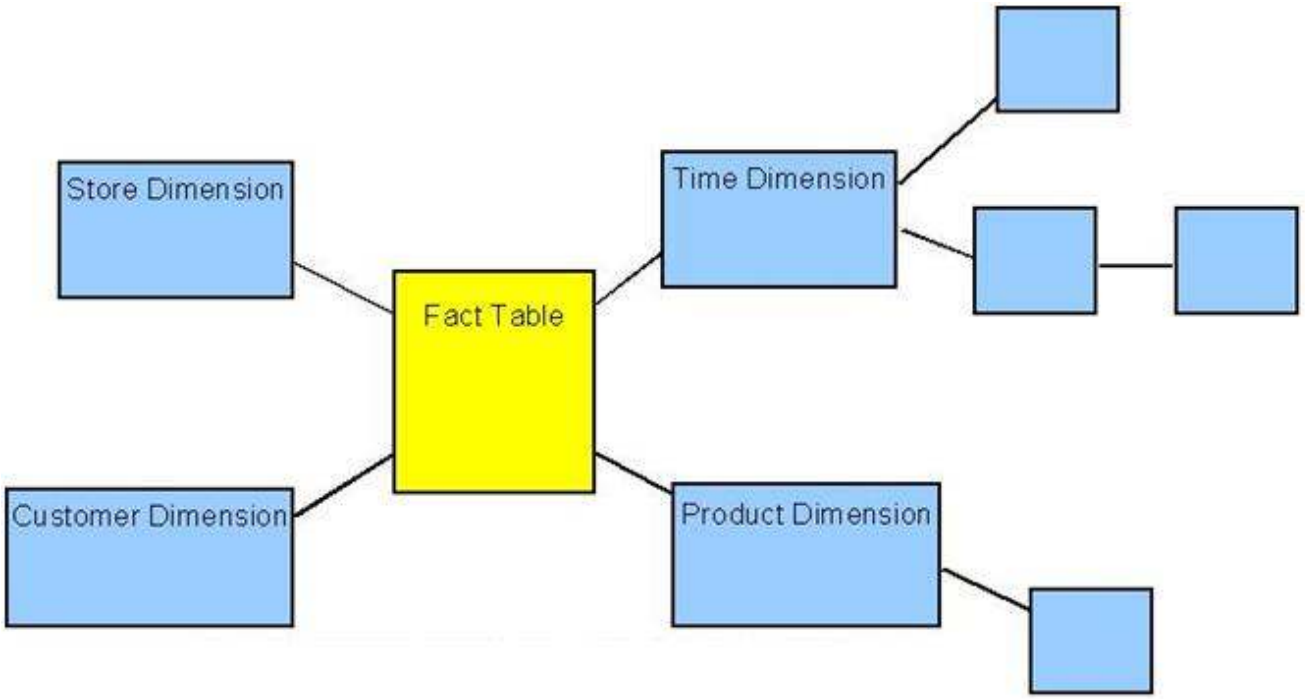

Figure 3

We will have 4 lookup tables in a snowflake schema: A lookup table for year, a lookup table for month, a lookup table for week, and a lookup table for day. Year is connected to Month, which is then connected to Day. Week is only connected to Day. A sample snowflake schema illustrating the above relationships in the Time Dimension is shown to the right. The main advantage of the snowflake schema is the improvement in query performance due to minimized disk storage requirements and joining smaller lookup tables. The main disadvantage of the snowflake schema is the additional maintenance efforts needed due to the increase number of lookup tables. The figure 3 above represents the snowflake schema in its simplest form.

Features of Snowflake Schema

- Some dimension tables in the Snowflake schema are normalized.
- Some dimension tables in the Snowflake schema are normalized.
- The normalization splits up the data into additional tables.
- Unlike Star schema, the dimensions table in a snowflake schema are normalized. For example, the item dimension table in star schema is normalized and split into two dimension tables, namely item and supplier table.

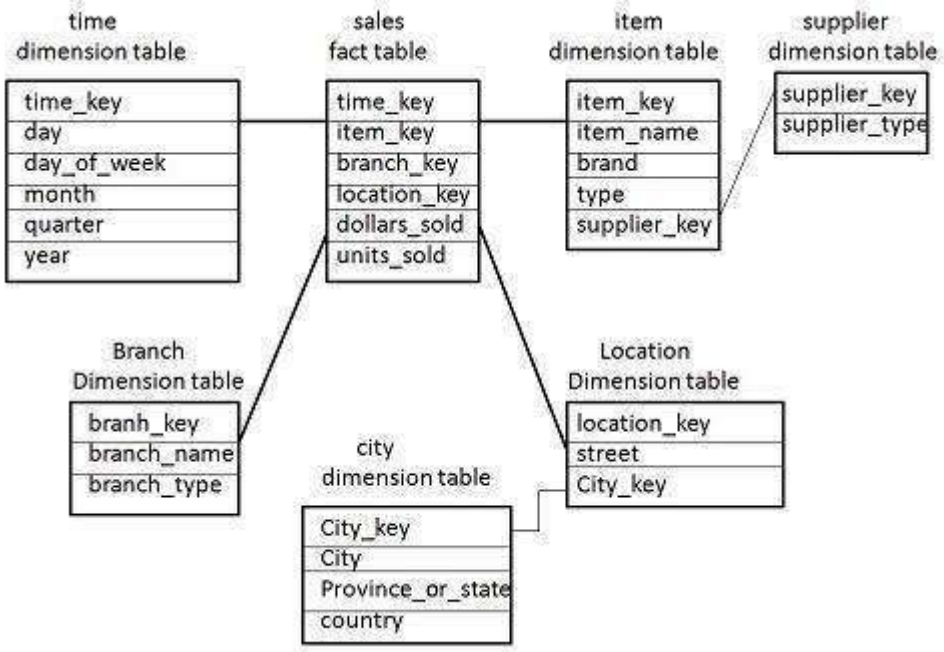

Figure 4

- Now the item dimension table contains the attributes item_key, item_name, type, brand, and supplier-key.
- The supplier key is linked to the supplier dimension table. The supplier dimension table contains the attributes supplier_key and supplier_type.

Note: Due to normalization in the Snowflake schema, the redundancy is reduced and therefore, it becomes easy to maintain and the save storage space.

2.3.2. Key differences in snowflake and star schema

The Star schema vs. Snowflake schema comparison brings forth four fundamental differences to the fore:

1. Data optimization:

Snowflake model uses normalized data, i.e. the data is organized inside the database in order to eliminate redundancy and thus helps to reduce the amount of data. The hierarchy of the business and its dimensions are preserved in the data model through referential integrity.

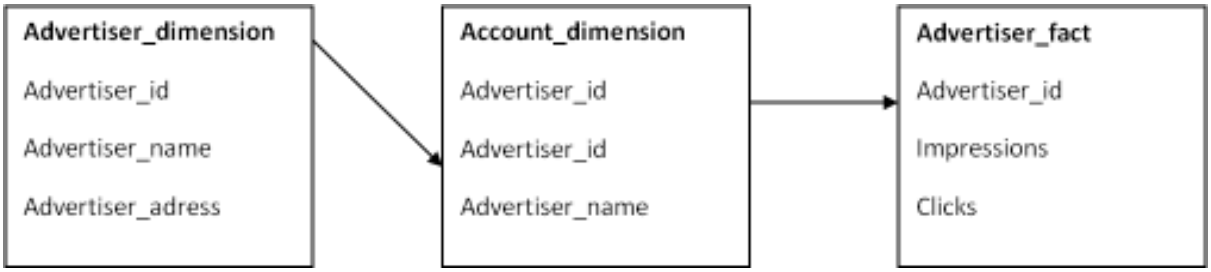

Figure 5: Snow flake model

Star model on the other hand uses de-normalized data. In the star model, dimensions directly refer to fact table and business hierarchy is not implemented via referential integrity between dimensions.

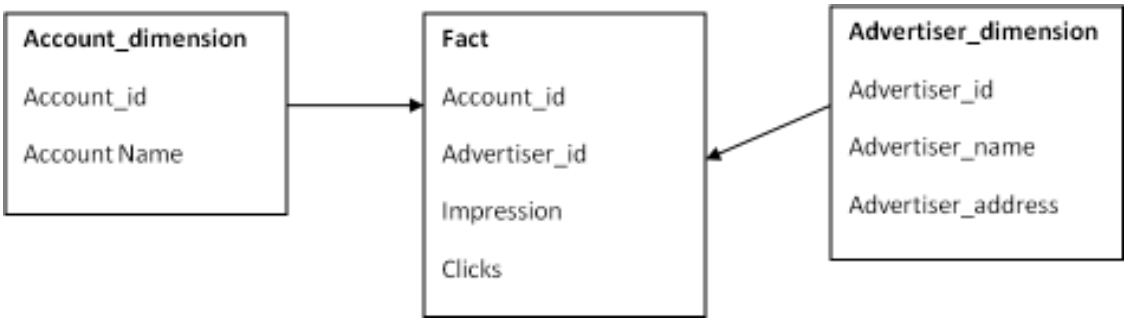

Figure 6: Snow flake model

2. Business model:
Primary key is a single unique key (data attribute) that is selected for a particular data. In the previous 'advertiser' example, the Advertiser_ID will be the primary key (business key) of a dimension table. The foreign key (referential attribute) is just a field in one table that matches a primary key of another dimension table. In our example, the Advertiser_ID could be a foreign key in Account_dimension.

In the **Snowflake model**, the business hierarchy of data model is represented in a primary key – Foreign key relationship between the various dimension tables.

In the **Star model** all required dimension-tables have only foreign keys in the fact tables.

3. Performance model:
The third differentiator in this Star schema vs Snowflake schema face off is the performance of these models.
The **Snowflake model** has higher number of joins between dimension table and then again the fact table and hence the performance is slower. For instance, if you want to know the Advertiser details, this model will ask for a lot of information such as the Advertiser Name, ID and address for which advertiser and account table needs to be joined with each other and then joined with fact table.

The **Star model** on the other hand has lesser joins between dimension tables and the facts table. In this model if you need information on the advertiser you will just have to join Advertiser dimension table with fact table.
4. ETL:
Snowflake model loads the data marts and hence the ETL job is more complex in design and cannot be parallelized as dependency model restricts it.

The **Star model** loads dimension table without dependency between dimensions and hence the ETL job is simpler and can achieve higher parallelism.

2.4. Process Flow in Data Warehouse

There are four major processes that contribute to a data warehouse
1. Extract and load the data.
2. Cleaning and transforming the data.
3. Backup and archive the data.

4. Managing queries and directing them to the appropriate data sources.

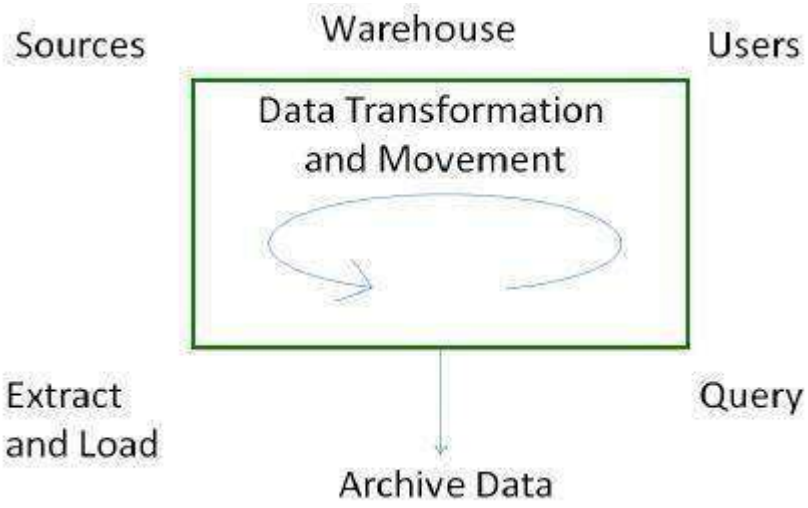

Figure 7

2.4.1 Extract and Load Process

Data extraction takes data from the source systems. Data load takes the extracted data and loads it into the data warehouse.

Note: Before loading the data into the data warehouse, the information extracted from the external sources must be reconstructed.

Controlling the Process

Controlling the process involves determining when to start data extraction and the consistency check on data. Controlling process ensures that the tools, the logic modules, and the programs are executed in correct sequence and at correct time.

When to Initiate Extract

Data needs to be in a consistent state when it is extracted, i.e., the data warehouse should represent a single, consistent version of the information to the user. For example, in a customer profiling data warehouse in telecommunication sector, it is illogical to merge the list of customers at 8 pm on Wednesday from a customer database with the customer subscription events up to 8 pm on Tuesday. This would mean that we are finding the customers for whom there are no associated subscriptions.

Loading the Data

After extracting the data, it is loaded into a temporary data store where it is cleaned up and made consistent.

Note: Consistency checks are executed only when all the data sources have been loaded into the temporary data store.

Clean and Transform Process
Once the data is extracted and loaded into the temporary data store, it is time to perform Cleaning and Transforming. Here is the list of steps involved in Cleaning and Transforming:

- Clean and transform the loaded data into a structure
- Partition the data
- Aggregation

Clean and Transform the Loaded Data into a Structure
Cleaning and transforming the loaded data helps speed up the queries. It can be done by making the data consistent:

- Within itself.
- With other data within the same data source.
- With the data in other source systems.
- With the existing data present in the warehouse.

Transforming involves converting the source data into a structure. Structuring the data increases the query performance and decreases the operational cost. The data contained in a data warehouse must be transformed to support performance requirements and control the ongoing operational costs.

Partition the Data
It will optimize the hardware performance and simplify the management of data warehouse. Here we partition each fact table into multiple separate partitions.

Aggregation
Aggregation is required to speed up common queries. Aggregation relies on the fact that most common queries will analyze a subset or an aggregation of the detailed data.

Backup and Archive the Data
In order to recover the data in the event of data loss, software failure, or hardware failure, it is necessary to keep regular backups. Archiving involves removing the old data from the system in a format that allow it to be quickly restored whenever required.

For example, in a retail sales analysis data warehouse, it may be required to keep data for 3 years with the latest 6 months data being kept online. In such as scenario, there is often a requirement to be able to do month-on-month comparisons for this year and last year. In this case, we require some data to be restored from the archive.

Query Management Process
This process performs the following functions:

- Manages the queries.
- Helps speed up the execution time of queries.
- Directs the queries to their most effective data sources.
- Ensures that all the system sources are used in the most effective way.
- Monitors actual query profiles.

The information generated in this process is used by the warehouse management process to determine which aggregations to generate. This process does not generally operate during the regular load of information into data warehouse.

Overview of ETL in Data Warehouses

ETL stands for Extract-Transform-Load. ETL covers a process of how the data are loaded from the source system to the data warehouse. Currently, the ETL encompasses a cleaning step as a separate step. The sequence is then Extract-Clean-Transform-Load. Let us briefly describe each step of the ETL process.

2.5. Data Extraction

Data Extraction is the operation of extracting data from a source system for further use in a data warehouse environment. This is the first step of the ETL process. After the data extraction, this data can be transformed and loaded into the data warehouse. The source systems for a data warehouse are typically transaction processing applications. For example, one of the source systems for a sales analysis data warehouse might be an order entry system that records all of the current order activities.

Designing and creating the data extraction process is often one of the most time-consuming tasks in the ETL process and, indeed, in the entire data warehousing process. The source systems might be very complex and poorly documented, and thus determining which data needs to be extracted can be difficult. The data has to be extracted normally not only once, but several times in a periodic manner to supply all changed data to the data warehouse and keep it up-to-date. Moreover, the source system typically cannot be modified, nor can its performance or availability be adjusted, to accommodate the needs of the data warehouse extraction process. These are important considerations for extraction and ETL in general. This topic, however, focuses on the technical considerations of having different kinds of sources and extraction methods. It assumes that the data warehouse team has already identified the data that will be extracted, and discusses common techniques used for extracting data from source databases.

Designing this process means making decisions about the following two main aspects:
1. Which extraction method do I choose?

This influences the source system, the transportation process, and the time needed for refreshing the warehouse.

2. How do I provide the extracted data for further processing?

This influences the transportation method, and the need for cleaning and transforming the data.

Data Extraction Methods in Data Warehouses

The data extraction method you should choose is highly dependent on the source system and also from the business needs in the target data warehouse environment. Very often, there is no possibility to add additional logic to the source systems to enhance an incremental extraction of data due to the performance or the increased workload of these systems. Sometimes even the customer is not allowed to add anything to an out-of-the-box application system. Based on the type of data to be extracted the data extraction methods are broadly classified as:

1. Logical Extraction Methods.
2. Physical Extraction Methods.

Logical Extraction Methods

There are two types of logical extraction:

- Full Extraction
- Incremental Extraction

Full Extraction

The data is extracted completely from the source system. Because this extraction reflects all the data currently available on the source system, there's no need to keep track of changes to the data source since the last successful extraction. The source data will be provided as-is and no additional logical information (for example, timestamps) is necessary on the source site. An example for a full extraction may be an export file of a distinct table or a remote SQL statement scanning the complete source table.

Incremental Extraction

At a specific point in time, only the data that has changed since a well-defined event back in history will be extracted. This event may be the last time of extraction or a more complex business event like the last booking day of a fiscal period. To identify this delta change there must be a possibility to identify all the changed information since this specific time event. This information can be either provided by the source data itself such as an application column, reflecting the last-changed timestamp or a change table where an appropriate additional mechanism keeps track of the changes besides the originating transactions. In most cases, using the latter method means adding extraction logic to the source system. Many data warehouses do not use any change-capture techniques as part of the extraction process. Instead, entire tables from the source systems are extracted to the data warehouse or staging area, and these tables are compared with a previous extract from the source system to identify the changed data. This approach may not have significant impact on the source systems, but it clearly can place a considerable burden on the data warehouse processes, particularly if the data volumes are large. Oracle's **Change Data Capture (CDC)** mechanism can extract and maintain such delta information.

Change Data Capture

An important consideration for extraction is incremental extraction, also called Change Data Capture. If a data warehouse extracts data from an operational system on a nightly basis, then the data warehouse requires only the data that has changed since the last extraction (that is, the data that has been modified in the past 24 hours). Change Data Capture is also the key-enabling technology for providing near real-time, or on-time, data warehousing. When it is possible to efficiently identify and extract only the most recently changed data, the extraction process (as well as all downstream operations in the ETL process) can be much more efficient, because it must extract a much smaller volume of data. Unfortunately, for many source systems, identifying the recently modified data may be difficult or intrusive to the operation of the system. Change Data Capture is typically the most challenging technical issue in data extraction.

Because change data capture is often desirable as part of the extraction process and it might not be possible to use the Change Data Capture mechanism, this section describes several techniques for implementing a self-developed change capture on Oracle Database source systems:

- Timestamps
- Partitioning
- Triggers

These techniques are based upon the characteristics of the source systems, or may require modifications to the source systems. Thus, each of these techniques must be carefully evaluated by the owners of the source system prior to implementation. Each of these techniques can work in conjunction with the data extraction technique discussed previously. For example, timestamps can be used whether the data is being unloaded to a file or accessed through a distributed query.

Timestamps

The tables in some operational systems have timestamp columns. The timestamp specifies the time and date that a given row was last modified. If the tables in an operational system have columns containing timestamps, then the latest data can easily be identified using the timestamp columns. For example, the following query might be useful for extracting today's data from an orders table:

```
SELECT * FROM orders

WHERE TRUNC(CAST(order_date AS date),'dd') =

  TO_DATE(SYSDATE,'dd-mon-yyyy');
```

If the timestamp information is not available in an operational source system, you will not always be able to modify the system to include timestamps. Such modification would require, first, modifying the operational system's tables to include a new timestamp column and then creating a trigger to update the timestamp column following every operation that modifies a given row.

Partitioning

Some source systems might use range partitioning, such that the source tables are partitioned along a date key, which allows for easy identification of new data. For example, if we are extracting from an orders table, and the orders table is partitioned by week, then it is easy to identify the current week's data.

Triggers

Triggers can be created in operational systems to keep track of recently updated records. They can then be used in conjunction with timestamp columns to identify the exact time and date when a given row was last modified. We do this by creating a trigger on each source table that requires change data capture. Following each DML statement that is executed on the source table, this trigger updates the timestamp column with the current time. Thus, the timestamp column provides the exact time and date when a given row was last modified. A similar internalized trigger-based technique is used for Oracle materialized view logs. These logs are used by materialized views to identify changed data, and these logs are accessible to end users. However, the format of the materialized view logs is not documented and might change over time.

If we want to use a trigger-based mechanism, we need to use synchronous change data capture. It is recommended that we use synchronous Change Data Capture for trigger based change capture, because CDC provides an externalized interface for accessing the change information and provides a framework for maintaining the distribution of this information to various clients.

Materialized view logs rely on triggers, but they provide an advantage in that the creation and maintenance of this change-data system is largely managed by the database.

However, Oracle recommends the usage of synchronous Change Data Capture for trigger-based change capture, since CDC provides an externalized interface for accessing the change information and provides a framework for maintaining the distribution of this information to various clients.

Trigger-based techniques might affect performance on the source systems, and this impact should be carefully considered prior to implementation on a production source system.

Physical Extraction Methods

Depending on the chosen logical extraction method and the capabilities and restrictions on the source side, the extracted data can be physically extracted by two mechanisms. The data can either be extracted online from the source system or from an offline structure. Such an offline structure might already exist or it might be generated by an extraction routine. There are the following methods of physical extraction:

- Online Extraction.
- Offline Extraction.

Online Extraction

The data is extracted directly from the source system itself. The extraction process can connect directly to the source system to access the source tables themselves or to an intermediate system that stores the data in a preconfigured manner (for example, snapshot logs or change tables). Note that the intermediate system is not necessarily physically different from the source system. With online extractions, we need to consider whether the distributed transactions are using original source objects or prepared source objects.

Offline Extraction

The data is not extracted directly from the source system but is staged explicitly outside the original source system. The data already has an existing structure (for example, redo logs, archive logs or transportable tablespaces) or was created by an extraction routine. We need to consider the following structures:

- Flat files
  Data in a defined, generic format. Additional information about the source object is necessary for further processing.
- Dump files
  Oracle-specific format. Information about the containing objects may or may not be included, depending on the chosen utility.
- Redo and archive logs
  Information is in a special, additional dump file.
- Transportable tablespaces
  A powerful way to extract and move large volumes of data between Oracle databases. Oracle recommends that you use transportable tablespaces whenever possible, because they can provide considerable advantages in performance and manageability over other extraction techniques.

2.6. Data Transformation

Data transformations are often the most complex and, in terms of processing time, the most costly part of the extraction, transformation, and loading (ETL) process. They can range from simple data conversions to extremely complex data scrubbing techniques. Many, if not all, data transformations can occur within all big types of databases like Oracle database and SQL databases, although transformations are often implemented outside of the database (for example, on flat files) as well.

This topic introduces techniques for implementing scalable and efficient data transformations within the Oracle Database. The examples in this topic are relatively simple. Real-world data transformations are often considerably more complex. However, the transformation techniques introduced in this topic meet the majority of real-world data transformation requirements, often with more scalability and less programming than alternative approaches.

This topic does not seek to illustrate all of the typical transformations that would be encountered in a data warehouse, but to demonstrate the types of fundamental technology that can be applied to implement these transformations and to provide guidance in how to choose the best techniques.

Transformation Flow

From an architectural perspective, we can transform our data in two ways:

- Multistage Data Transformation
- Pipelined Data Transformation

Multistage Data Transformation

The data transformation logic for most data warehouses consists of multiple steps. For example, in transforming new records to be inserted into a sales table, there may be separate logical transformation steps to validate each dimension key.

Figure 8 offers a graphical way of looking at the transformation logic.

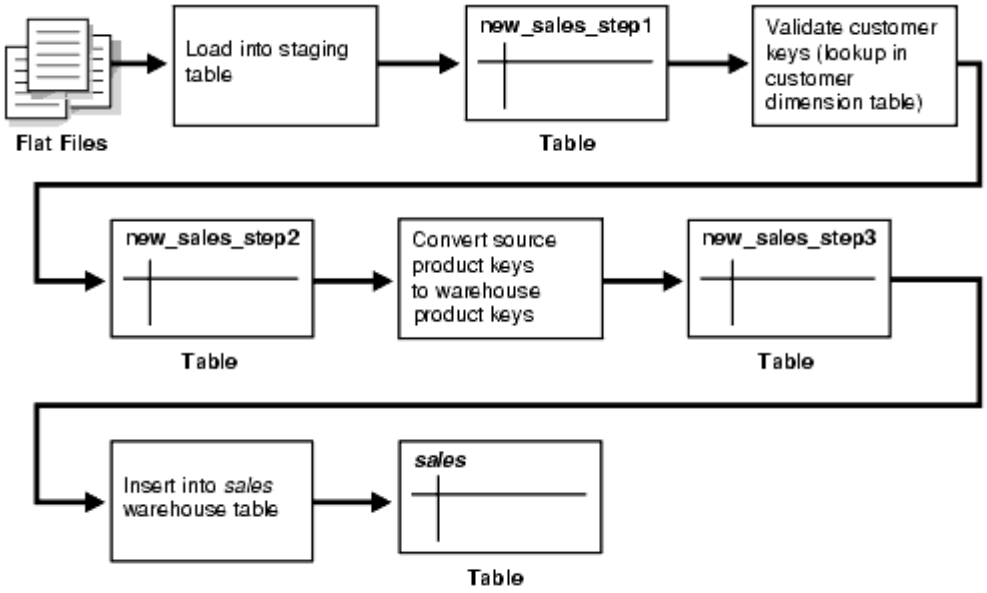

Figure 8: Multistage Data Transformation

When using Oracle Database as a transformation engine, a common strategy is to implement each transformation as a separate SQL operation and to create a separate, temporary staging table (such as the tables new_sales_step1 and new_sales_step2 in Figure 14-1) to store the incremental results for each step. This load-then-transform strategy also provides a natural checkpointing scheme to the entire transformation process, which enables to the process to be more easily monitored and restarted. However, a disadvantage to multistaging is that the space and time requirements increase. It may also be possible to combine many simple logical transformations into a single SQL statement or single PL/SQL procedure. Doing so may provide better performance than performing each step independently, but it may also introduce difficulties in modifying, adding, or dropping individual transformations, as well as recovering from failed transformations.

Pipelined Data Transformation

The ETL process flow can be changed dramatically and the database becomes an integral part of the ETL solution. The new functionality renders some of the former necessary process steps obsolete while some others can be remodelled to enhance the data flow and the data transformation to become more scalable and non-interruptive. The task shifts from serial transform-then-load process (with most of the tasks done outside the database) or load-then-transform process, to an enhanced transform-while-loading. Oracle offers a wide variety of new capabilities to address all the issues and tasks relevant in an ETL scenario. It is important to understand that the database offers toolkit functionality rather than trying to address a one-size-fits-all solution. The underlying database has to enable the most appropriate ETL process flow for a specific customer need, and not dictate or constrain it from a technical perspective. Figure 9 illustrates the pipelined data transformation.

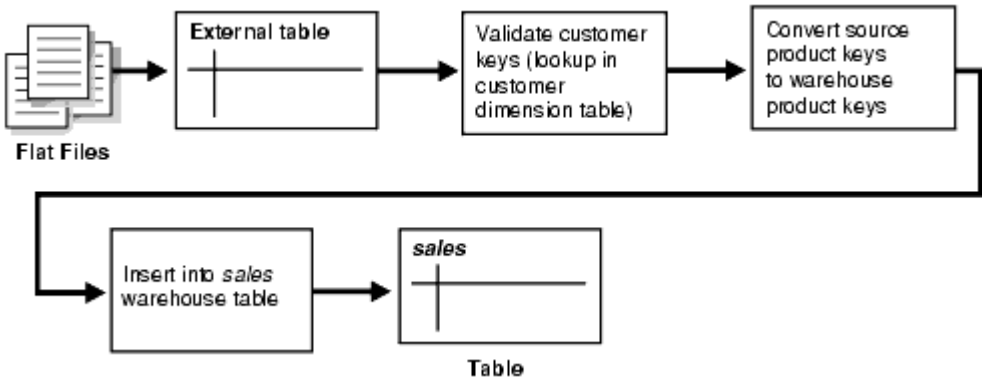

Figure 9: Pipelined Data Transformation

Transformation Mechanisms

Following are the choices for transforming data inside the database

I. Transforming Data Using SQL
II. Transforming Data Using PL/SQL
III. Transforming Data Using Table Functions

I. Transforming Data Using SQL

Once data is loaded into the database, data transformations can be executed using SQL operations. There are four basic techniques for implementing SQL data transformations:

- CREATE TABLE ... AS SELECT And INSERT /*+APPEND*/ AS SELECT
- Transforming Data Using UPDATE
- Transforming Data Using MERGE
- Transforming Data Using Multitable INSERT

CREATE TABLE ... AS SELECT And INSERT /*+APPEND*/ AS SELECT

The CREATE TABLE ... AS SELECT statement (CTAS) is a powerful tool for manipulating large sets of data. As shown in the following example, many data transformations can be expressed in standard SQL, and CTAS provides a mechanism for efficiently executing a SQL query and storing the results of that query in a new database table. The INSERT /*+APPEND*/ ... AS SELECT statement offers the same capabilities with existing database tables. Let us consider below table for understanding the transformation process.

```
CREATE TABLE sales_transactions_ext
```

```
(PROD_ID NUMBER, CUST_ID NUMBER,
 TIME_ID DATE, CHANNEL_ID NUMBER,
 PROMO_ID NUMBER, QUANTITY_SOLD NUMBER,
 AMOUNT_SOLD NUMBER(10,2), UNIT_COST NUMBER(10,2),
 UNIT_PRICE NUMBER(10,2))
ORGANIZATION external (TYPE oracle_loader
  DEFAULT DIRECTORY data_file_dir ACCESS PARAMETERS
  (RECORDS DELIMITED BY NEWLINE CHARACTERSET US7ASCII
    BADFILE log_file_dir:'sh_sales.bad_xt'
    LOGFILE log_file_dir:'sh_sales.log_xt'
    FIELDS TERMINATED BY "|" LDRTRIM
    ( PROD_ID, CUST_ID,
      TIME_ID         DATE(10) "YYYY-MM-DD",
      CHANNEL_ID, PROMO_ID, QUANTITY_SOLD, AMOUNT_SOLD,
      UNIT_COST, UNIT_PRICE))
  location ('sh_sales.dat')
)REJECT LIMIT UNLIMITED;
```

In a data warehouse environment, CTAS is typically run in parallel using NOLOGGING mode for best performance. A simple and common type of data transformation is data substitution. In a data substitution transformation, some or all of the values of a single column are modified. For example, our sales table has a ***channel_id*** column. This column indicates whether a given sales transaction was made by a company's own sales force (a direct sale) or by a distributor (an indirect sale).
You may receive data from multiple source systems for your data warehouse. Suppose that one of those source systems processes only direct sales, and thus the source system does not know indirect sales channels. When the data warehouse initially receives sales data from this system, all sales records have a NULL value for the sales.**channel_id** field. These NULL values must be set to the proper key value.

For example, you can do this efficiently using a SQL function as part of the insertion into the target sales table statement. The structure of source table **sales_activity_direct** is as follows:

```
DESC sales_activity_direct

Name           Null?    Type

------------   -----    ----------------
```

```
SALES_DATE              DATE

PRODUCT_ID              NUMBER

CUSTOMER_ID             NUMBER

PROMOTION_ID            NUMBER

AMOUNT                  NUMBER

QUANTITY                NUMBER
```

The following SQL statement inserts data from sales_activity_direct into the sales table of the sample schema, using a SQL function to truncate the sales date values to the midnight time and assigning a fixed channel ID of 3.

```
INSERT /*+ APPEND NOLOGGING PARALLEL */

INTO sales SELECT product_id, customer_id, TRUNC(sales_date), 3,

       promotion_id, quantity, amount

FROM sales_activity_direct;
```

Transforming Data Using UPDATE

Another technique for implementing a data substitution is to use an **UPDATE** statement to modify the sales.**channel_id** column. An **UPDATE** will provide the correct result. However, if the data substitution transformations require that a very large percentage of the rows (or all of the rows) be modified, then, it may be more efficient to use a **CTAS** statement than an **UPDATE**.

Transforming Data Using MERGE

Oracle Database's merge functionality extends SQL, by introducing the SQL keyword MERGE, in order to provide the ability to update or insert a row conditionally into a table or out of line single table views. Conditions are specified in the ON clause. This is, besides pure bulk loading, one of the most common operations in data warehouse synchronization.

Transforming Data Using Multitable INSERT

Many times, external data sources have to be segregated based on logical attributes for insertion into different target objects. It is also frequent in data warehouse environments to fan out the same source data into several target objects. Multitable inserts provide a new SQL statement for these kinds of transformations, where data can either end up in several or exactly one target, depending on the business transformation rules. This insertion can be done conditionally based on business rules or unconditionally.

It offers the benefits of the **INSERT ... SELECT** statement when multiple tables are involved as targets. In doing so, it avoids the drawbacks of the two obvious alternatives. You either had to deal with n independent **INSERT ... SELECT** statements, thus processing the same source data n times and increasing the transformation workload n times. Alternatively, you had to choose a procedural

approach with a per-row determination how to handle the insertion. This solution lacked direct access to high-speed access paths available in SQL. As with the existing INSERT ... SELECT statement, the new statement can be parallelized and used with the direct-load mechanism for faster performance.

II. Transforming Data Using PL/SQL

In a data warehouse environment, you can use procedural languages such as PL/SQL to implement complex transformations in the Oracle Database. Whereas CTAS operates on entire tables and emphasizes parallelism, PL/SQL provides a row-based approached and can accommodate very sophisticated transformation rules. For example, a PL/SQL procedure could open multiple cursors and read data from multiple source tables, combine this data using complex business rules, and finally insert the transformed data into one or more target table. It would be difficult or impossible to express the same sequence of operations using standard SQL statements.

Using a procedural language, a specific transformation (or number of transformation steps) within a complex ETL processing can be encapsulated, reading data from an intermediate staging area and generating a new table object as output. A previously generated transformation input table and a subsequent transformation will consume the table generated by this specific transformation. Alternatively, these encapsulated transformation steps within the complete ETL process can be integrated seamlessly, thus streaming sets of rows between each other without the necessity of intermediate staging. You can use table functions to implement such behavior.

III. Transforming Data Using Table Functions

Table functions provide the support for pipelined and parallel execution of transformations implemented in PL/SQL, C, or Java. Scenarios as mentioned earlier can be done without requiring the use of intermediate staging tables, which interrupt the data flow through various transformations steps.

What is a Table Function? A table function is defined as a function that can produce a set of rows as output. Additionally, table functions can take a set of rows as input. Prior to Oracle9i, PL/SQL functions:

- Could not take cursors as input.
- Could not be parallelized or pipelined.

Now, functions are not limited in these ways. Table functions extend database functionality by allowing:

- Multiple rows to be returned from a function.
- Results of SQL subqueries (that select multiple rows) to be passed directly to functions.
- Functions take cursors as input.
- Functions can be parallelized.
- Returning result sets incrementally for further processing as soon as they are created. This is called incremental pipelining

Table functions can be defined in PL/SQL using a native PL/SQL interface, or in Java or C using the Oracle Data Cartridge Interface (ODCI).

2.6. Data Loading

The loading phase is the last step of the ETL process. The information from data sources are loaded and stored in a form of tables. There are two types of tables in the database structure: fact

tables and dimensions tables. Once the fact and dimension tables are loaded, it is time to improve performance of the Business Intelligence data by creating aggregates. During the load step, it is necessary to ensure that the load is performed correctly and with as little resources as possible. The target of the Load process is often a database. In order to make the load process efficient, it is helpful to disable any constraints and indexes before the load and enable them back only after the load completes. The referential integrity needs to be maintained by ETL tool to ensure consistency.

Aggregations

In a large databases records are usually aggregated to improve performance. To design the most effective aggregates you should meet some basic requirements.

1. All aggregates should be stored in their own fact table(separated from base-level data).
2. All dimensions related should be shrunken versions of dimensions associated with base-level data.
3. Associate the base fact tables and aggregate fact tables in one family and force SQL to refer it.

The aggregations have beneficial influence on performance. It is a very popular technique for speeding up query time in business decisions. It is possible with using the aggregate navigator. The aggregate navigator allows to use information stored in aggregates automatically , understand the clients SQL and transform base-level SQL into an aggregate SQL.

Effective loading process

If you want to perform effective load process, there are a few things to have in mind:

- you should not forget about leaving all indexes before start loading tables. You may rebuild them without any problems after loading.
- you should manage the partitions. Partitions allow to divide one table in many smaller tables for administration purposes and to improve the query performance in large fact tables. The most recommended strategy is to partition tables by a data interval like year , month or quarter.
- you should make the following steps during loading the data: separate inserts from updates (separate old data with a new one) , make a bulk load , load in parallel(you may use partitions) delete updates and subsequently bulk load new versions of records , built outside aggregates.
- you should allow for incremental loading which keep the database synchronized with the source system.

Feeding a data warehouse

Loading dimension and fact tables do not cause the end of creating database. The data can be modified after loading without affecting the end user quire process. Such modifications are called the graceful modifications and include:

- Adding a fact or dimension to an existing fact table of the same grain
- Adding an attribute to an existing dimension
- Increase the granularity of existing fact and dimension table
- The data is also analyzed many times after the loading step. The most popular structure that allows fast analysis of data is OLAP(Online Analytical Processes)cube. It provide quickly multi-dimensional analytical queries in a short time.
- The loading process seems to be very time-consuming and quite difficult. However it is one of three the most important steps of creating databases and should be performed carefully and without hurry.

Generally loading dimension and fact tables are quite similarly but remember one thing: before loading fact tables first load dimension tables because facts make no sense without dimensions!

2.7. Data Quality and its Need

Data quality refers to the level of quality of data. There are many definitions of data quality but data are generally considered high quality if "they are fit for their intended uses in operations, decision making and planning. Alternatively, data is deemed of high quality if it correctly represents the real-world construct to which it refers. Furthermore, apart from these definitions, as data volume increases, the question of internal consistency within data becomes significant, regardless of fitness for use for any particular external purpose. People's views on data quality can often be in disagreement, even when discussing the same set of data used for the same purpose. Following are the few commonly used definitions of the data quality.

- Degree of excellence exhibited by the data in relation to the portrayal of the actual scenario.
- The state of completeness, validity, consistency, timeliness and accuracy that makes data appropriate for a specific use.
- The totality of features and characteristics of data that bears on its ability to satisfy a given purpose; the sum of the degrees of excellence for factors related to data
- The processes and technologies involved in ensuring the conformance of data values to business requirements and acceptance criteria.
- Complete, standards based, consistent, accurate and time stamped.

If the ISO 9000:2015 definition of quality is applied, data quality can be defined as the degree to which a set of characteristics of data fulfils requirements. Examples of characteristics are: ***completeness***, ***validity***, ***accuracy***, ***consistency***, ***availability*** and ***timeliness***. Requirements are defined as the need or expectation that is stated, generally implied or obligatory.

There are a number of theoretical frameworks for understanding data quality. A systems-theoretical approach influenced by American pragmatism expands the definition of data quality to include information quality, and emphasizes the inclusiveness of the fundamental dimensions of accuracy and precision on the basis of the theory of science (Ivanov, 1972).

One framework, dubbed "***Zero Defect Data***" (Hansen, 1991) adapts the principles of statistical process control to data quality.

Another framework seeks to integrate the product perspective (conformance to specifications) and the service perspective (meeting consumers' expectations) (Kahn et al. 2002).

Another framework is based in semiotics to evaluate the quality of the form, meaning and use of the data (Price and Shanks, 2004).
In practice, data quality is a concern for professionals involved with a wide range of information systems, ranging from data warehousing and business intelligence to customer relationship management and supply chain management. One industry study estimated the total cost to the U.S. economy of data quality problems at over U.S. $600 billion per annum. Incorrect data – which includes invalid and outdated information – can originate from different data sources – through data entry, or data migration and conversion projects.

Data Quality Need
Data quality-related problems cost companies millions of dollars annually because of lost revenue opportunities, failure to meet regulatory compliance or failure to address customer issues in a timely manner. Poor data quality is often cited as a reason for failure of critical information-intensive projects. Data quality program is need for enabling organizations to achieve following:

- **Deliver** high-quality data for a range of enterprise initiatives including business intelligence, applications consolidation and retirement, and master data management
- **Reduce** time and cost to implement CRM, data warehouse/BI, data governance, and other strategic IT initiatives and maximize the return on investments
- **Construct** consolidated customer and household views, enabling more effective cross-selling, up-selling, and customer retention
- **Help** improve customer service and identify a company's most profitable customers
- **Provide** business intelligence on individuals and organizations for research, fraud detection, and planning
- **Reduce** the time required for data cleansing—saving on average 5 million hours, for an average company with 6.2 million records.

For data warehousing and CRM, data quality issues first arise during the initial application design stages when requirements for extracting and transforming data from operational systems are developed. However, data quality issues do not stop here; they remain an ongoing concern throughout application development, use and maintenance.

Data quality assurance

Data quality assurance is the process of profiling the data to discover inconsistencies and other anomalies in the data, as well as performing data cleansing activities (e.g. removing outliers, missing data interpolation) to improve the data quality. These activities can be undertaken as part of data warehousing or as part of the database administration of an existing piece of applications software.

Data quality control

Data quality control is the process of controlling the usage of data with known quality measurements for an application or a process. This process is usually done after a Data Quality Assurance (QA) process, which consists of discovery of data inconsistency and correction. Data QA processes provides following information to Data Quality Control (QC):

- Severity of inconsistency
- Incompleteness
- Accuracy
- Precision
- Missing / Unknown

The Data QC process uses the information from the QA process to decide to use the data for analysis or in an application or business process. For example, if a Data QC process finds that the data contains too many errors or inconsistencies, then it prevents that data from being used for its intended process which could cause disruption. For example, providing invalid measurements from several sensors to the automatic pilot feature on an aircraft could cause it to crash. Thus, establishing data QC process provides the protection of usage of data control and establishes safe information usage.

Optimum use of Data Quality

Data Quality (DQ) is a niche area required for the integrity of the data management by covering gaps of data issues. This is one of the key functions that aid data governance by monitoring data to find

exceptions undiscovered by current data management operations. Data Quality checks may be defined at attribute level to have full control on its remediation steps.

DQ checks and business rules may easily overlap if an organization is not attentive of its DQ scope. Business teams should understand the DQ scope thoroughly in order to avoid overlap. Data quality checks are redundant if business logic covers the same functionality and fulfils the same purpose as DQ. The DQ scope of an organization should be defined in DQ strategy and well implemented. Some data quality checks may be translated into business rules after repeated instances of exceptions in the past. Below are a few areas of data flows that may need perennial DQ checks:

- **Completeness and precision** DQ checks on all data may be performed at the point of entry for each mandatory attribute from each source system. Few attribute values are created way after the initial creation of the transaction; in such cases, administering these checks becomes tricky and should be done immediately after the defined event of that attribute's source and the transaction's other core attribute conditions are met.
- All data having attributes referring to Reference Data in the organization may be validated against the set of well-defined valid values of Reference Data to discover new or discrepant values through the ***validity*** DQ check. Results may be used to update Reference Data administered under Master Data Management (MDM).
- All data sourced from a third party to organization's internal teams may undergo ***accuracy*** (DQ) check against the third party data. These DQ check results are valuable when administered on data that made multiple hops after the point of entry of that data but before that data becomes authorized or stored for enterprise intelligence.
- All data columns that refer to Master Data may be validated for its ***consistency*** check. A DQ check administered on the data at the point of entry discovers new data for the MDM process, but a DQ check administered after the point of entry discovers the failure (not exceptions) of consistency.
- As data transforms, multiple timestamps and the positions of that timestamps are captured and may be compared against each other and its leeway to validate its value, decay, operational significance against a defined SLA (service level agreement). This ***timeliness*** DQ check can be utilized to decrease data value decay rate and optimize the policies of data movement timeline.
- In an organization complex logic is usually segregated into simpler logic across multiple processes. **Reasonableness** DQ checks on such complex logic yielding to a logical result within a specific range of values or static interrelationships (aggregated business rules) may be validated to discover complicated but crucial business processes and outliers of the data, its drift from BAU (business as usual) expectations, and may provide possible exceptions eventually resulting into data issues. This check may be a simple generic aggregation rule engulfed by large chunk of data or it can be a complicated logic on a group of attributes of a transaction pertaining to the core business of the organization. This DQ check requires high degree of business knowledge and acumen. Discovery of reasonableness issues may aid for policy and strategy changes by either business or data governance or both.
- ***Conformity*** checks and ***integrity checks*** need not covered in all business needs, it's strictly under the database architecture's discretion.

2.8. Data Quality Challenges

Deploying a data quality management program is not easy; there are significant challenges that must be overcome. Some of the most significant reasons companies do not pursue a formal data quality management initiative include:

- No business unit or department feels it is ***responsible*** for the problem.
- It requires ***cross-functional*** cooperation.

- It requires the organization to **recognize** that it has significant problems.
- It requires **discipline**.
- It requires an **investment** of financial and human resources.
- It is perceived to be extremely **manpower-intensive**.
- The **return on investment** is often difficult to quantify.

Responsibility

One of the more significant challenges is that no single business unit is responsible for all of the data in an enterprise, and the inclusion of a responsibility for data quality in a job description is very unusual. Furthermore, once the data is in the computer, business units often wash their hands of the problem and blame it on IT. IT cannot create the business rules nor should it be held responsible to make business decisions concerning the data. IT can only ensure that electronic rules, based on business rules, operate correctly. Effective data quality management requires organizations to adopt a data stewardship approach. Stewardship is different than ownership. A steward is a person who is expected to exercise responsible care over an asset that he or she does not own. The data is actually owned by the enterprise. The steward is responsible for caring for that asset. Data stewardship is important, but establishing a data stewardship program is very difficult! One immediate challenge to a stewardship program is to identify the group or person responsible for a set of data. Some responsibilities are fairly easy to allocate: the Facilities Management Department may be responsible for data about real estate and property; the Human Resources Department may be responsible for data about employees; and the Finance Department may be responsible for the organization's financial data. But who is responsible for payroll data (its financial data about employees), who is responsible for customer data, and who is responsible for product data? Another requirement is to get business people to focus on the data issues. The business leaders are concerned with their functional responsibilities. A Marketing Vice President understands customer segmentation and campaign management; data quality management is not his or her forte. The Marketing Vice President must recognize that unless his or her quality expectations are established for the data, it is unlikely that the condition of the data will be SUGI 29 Data Warehousing, Management and Quality 4 sufficient to support his or her needs. Similarly, the Manufacturing Vice President is focused on producing the products. He or she must recognize that data about the products (e.g., specifications, substitutable raw materials, inventory levels at warehouses and customer sites) impacts the department's ability to profitably produce the products.

Cross Functionality

The lack of clear responsibility for data leads to the second major challenge. Unlike most corporate functions, which are vertically aligned within an organizational unit, data quality management is horizontal in nature. That is, data quality management responsibilities cross organizational boundaries. Effective data quality management provides organizations with the ability to share data and that requires consistent definitions. If three business units (e.g., Manufacturing, Product Development, and Sales) have responsibility for products, they must work together to arrive at common business rules and definitions for data about products and the way they may be sold. This means that each of the autonomous units needs to give up some of its control over the product, and that is often difficult to attain.

Problem Recognition

At the beginning of a project, we've had clients tell us that they have no data quality problems. We've never had a client tell us that after completing that project. This is because during the course of the project, we helped the client understand its data better, and in that process, defects in the existing data come to light. One of the first steps to solving any problem is recognizing that the problem exists. Organizations are often in denial about their data quality problems, and it sometimes takes a major catastrophe to change that attitude. Absent such an event, organizations are not prone to spend money fixing something that they don't think is broken.

Discipline

The fourth major challenge is discipline. An effective data quality management program requires discipline. Responsibilities must be assigned and formal procedures must be created and followed – by everyone who handles data for the enterprise. Assigning responsibilities means that a field agent needs to understand the value of the data gathered about a product sale to a customer so that he or she captures it correctly, it means that IT understands the quality expectations so that it builds the systems to process the data correctly, and it means that the business analysts using the data understand its meaning so that they can make well-informed decisions. Further, it means that job descriptions for individuals in these positions reflect their data quality management responsibilities.

Investment

Philip B. Crosby wrote a book entitled "Quality is Free", claims that it isn't quality that is expensive; it's the cost of "un-quality". Examples of the cost of "un-quality" include the cost of sending duplicate promotional materials because customers are duplicated in the database, the opportunity cost of not sending materials to the right customers because the data used to segment customers is flawed, the opportunity cost of not shipping products to a customer because of inaccurate information about inventory levels, and the time spent finding and reconciling data needed to make effective decisions.

On-Going Effort

Organizations are constantly looking for ways to avoid increasing their staff size, or in many cases, to reduce the staff size. A data quality management program requires people, and it's very difficult to obtain authorization for an effort that will increase staff size. Implementing data quality management technology, however, automates manual data quality and integration projects and frees up resources for other projects. Because a data quality management software investment costs less than one employee (salary, benefits, rent, etc), Implementations actually support organizational goals of increased productivity or reduced staffing. Business representatives will be needed to perform the governance related activities, and this is unavoidable. The effort that is avoidable is the staff required to identify and correct data problems on an on-going basis. This is an area in which technology plays an important role. Data quality management technology can directly support each of the four phases of the program described in the next major section. The technology can help gain an understanding of the data (data profiling), take steps to correct problems (data quality), automate some discrepancy resolution resulting from merging data from multiple sources (data integration), and add value to the data (data augmentation). To be effective, the tools need to be customizable so that they impose rules established by the organization. In SUGI 29 Data Warehousing, Management

and Quality 5 addition, the tools need to learn from the routines they execute so that they can deploy rules based on learned data patterns.

Return On Investment

Data quality management efforts are difficult to fund because the cost of "un-quality" is not documented. The documentation of these costs requires recognition of the problem (as discussed earlier), and also requires managers to admit that they are wasting money or that they are not effectively utilizing resources at their disposals. Making these admissions, particularly in tough economic times, is risky. It is imperative that top management create an environment in which people are not unduly penalized for admitting to past problems. People within most organizations are aware of data quality problems and have taken steps to work around them. Solicit information from people working with the data and canvass customer and supplier complaints and try to discern the ones that may have been caused by erroneous data. While this information may not provide a return on investment prediction in financial terms, you are likely to find enough examples of problems to justify addressing at least one area. When you do address an area, be sure to document both the costs and the resultant savings, and use that information to justify data quality management initiatives in other areas.

2.9. Data Quality Tools

Data quality tools generally fall into one of three categories: auditing, cleansing and migration. The focus of this article is on tools that clean and audit data, with a limited look at tools that extract and migrate data. Data auditing tools enhance the accuracy and correctness of the data at the source. These tools generally compare the data in the source database to a set of business rules. (Williams, 1997) When using a source external to the organization, business rules can be determined by using data mining techniques to uncover patterns in the data. Business rules that are internal to the organization should be entered in the early stages of evaluating data sources. Lexical analysis may be used to discover the business sense of words within the data. The data that does not adhere to the business rules could then be modified as necessary.

Data cleansing tools are used in the intermediate staging area. The tools in this category have been around for a number of years. A data cleansing tool cleans names, addresses and other data that can be compared to an independent source. These tools are responsible for parsing, standardizing, and verifying data against known lists such as U.S. Postal Codes. The data cleansing tools contain features which perform the following functions:

- Data parsing (elementizing)- breaks a record into atomic units that can be used in subsequent steps. Parsing includes placing elements of a record into the correct fields. In the following example "ST" is used in a variety of ways:
  Elizabeth St. Francis
  1130 1 st St.
  St. 101
  St. Paul, MN 50505 ·

- Data standardization- converts the data elements to forms that are standard throughout the data warehouse. For example, all incidences of avenue should be represented as ave., not Avenue, avenue, or av.
- Data correction and verification- matches data against know lists, such as U.S. Postal Codes, product lists, internal customer lists.
- Record matching- determines whether two records represent data on the same subject. For example, the following two records probably represent the same person: Sue Smith Suzanne Smith 19 Rt 9G AND 19 North Road Hyde Park, NY 12538 Hyde Park, NY 12538 (914)229-1111 (914)229-1111.
- Data transformation- ensures consistent mapping between source systems and data warehouse. For example, "1" for male, and "2" for female becomes "M" and "F".
- House holding – combining individual records that have the same address.
- Documenting – documenting the results of the data cleansing steps in the meta data

The third type of tool, **the data migration tool**, is used in extracting data from a source database, and migrating the data into an intermediate storage area. The migration tools also transfer data from the staging area into the data warehouse. The data migration tool is responsible for converting the data from one platform to another. A migration tool will map the data from the source to the data warehouse. There can be a great deal of overlap in these tools and many of the same features are found in tools of each category.

There are hundreds of tools that can be classified as data extraction, loading and cleansing tools. Only a small percentage of the tools perform the data cleansing and auditing functions, the majority perform extraction and loading functions.

Questions to be asked	**Features**	**Tools**
Auditing Tools		
Is your data complete and valid?	Data examination- determines quality of data, patterns within it, and number of different fields used	WizSoft- WizRule Vality- Integrity
Does your data comply to your business rules? (Do you have missing values, illegal values, inconsistent values, invalid relationships?)	Compare to business rules and assess data for consistency and completeness against rules	Prism Solutions, Inc.- Prism Quality Manager WizSoft - WizRule Vality- Integrity
Are you using sources that comply to your business rules?	Data reengineering- examining the data to determine what the business rules are	WizSoft – WizRule Vality- Integrity
Cleansing Tools		
Does your data need to be broken up	Data parsing (elementizing)- context and	Trillium Software- Parser

between source and data warehouse?	destination of each component of each field	i.d. Centric- DataRight
Does your data have abbreviations that should be changed to insure consistency?	Data standardizing- converting data elements to forms that are standard throughout the DW	Trillium Software- Parser i.d. Centric- DataRight
Is your data correct?	Data correction and verification- matches data against known lists (addresses, product lists, customer lists)	Trillium Software- Parser Trillium Software- GeoCoder i.d. Centric- ACE, Clear I.D. Library Group 1- NADIS
Is there redundancy in your data?	Record matching- determines whether two records represent data on the same object	Trillium Software- Matcher Innovative Systems- Match i.d. Centric- Match/Consolidation Group 1- Merge/Purge Plus
Are there multiple versions of company names in your database?	Record matching- based on user specified fields such as tax ID	Innovative Systems- Corp-Match
Is your data consistent prior to entering data warehouse?	Transform data- "1" for male, "2" for female becomes "M" & "F"- ensures consistent mapping between source systems and data warehouse	Vality- Integrity i.d. Centric- Match/Consolidation
Do you have information in free form fields that differs between databases?	Data reengineering- examining the data to determine what the business rules are	Vality- Integrity
Do you multiple individuals in the same household that need to be grouped together?	Householding- combining individual records that have same address	i.d. Centric- Match/Consolidation Trillium Software- Matcher
Does your data contain atypical words- such as industry specific words, ethnic or hyphenated names?	Data parsing combined with data verification- comparison to industry specific lists	i.d. Centric- ACE, Clear I.D.
Migration and Other Tools		
Do you have multiple formats to be	Access the data then map it to the dw schema	Enterprise/Integrator by

accessed- relational dbs, flat files, etc.?		Carleton.
Do you have free form text that needs to be indexed, classified, other?	Text mining- extracts meaning and relevance from large amounts of information	Semio- SemioMap
Have the rules established during the data cleansing steps been reflected in the metadata?	Documenting- documenting the results of the data cleansing steps in the metadata	Vality- Integrity

Conclusion: Data quality tools are available to enhance the quality of the data at several stages in the process of developing a data warehouse. Cleansing tools can be useful in automating many of the activities that are involved in cleansing the data- parsing, standardizing, correction, matching, transformation and house holding. Many of the tools specialize in auditing the data, detecting patterns in the data, and comparing the data to business rules.

2.10. Information Delivery

A data warehouse is never static; it evolves as the business expands. As the business evolves, its requirements keep changing and therefore a data warehouse must be designed to ride with these changes. Hence a data warehouse system needs to be flexible. Ideally there should be a delivery process to deliver a the information contained in a data warehouse. However data warehouse projects normally suffer from various issues that make it difficult to complete tasks and deliverables in the strict and ordered fashion demanded by the waterfall method. Most of the times, the requirements are not understood completely. The architectures, designs, and build components can be completed only after gathering and studying all the requirements.

Delivery Method

The delivery method is a variant of the joint application development approach adopted for the delivery of information contained in a data warehouse. There are four underlying methods for information delivery. we may be catering to the needs of any class of users. We may be constructing the information delivery system to satisfy the requirements of users with simple needs or those of

power users. Still the principal means of delivery are the same. are the same.

MATCHING INFORMATION TO THE CLASSES OF USERS

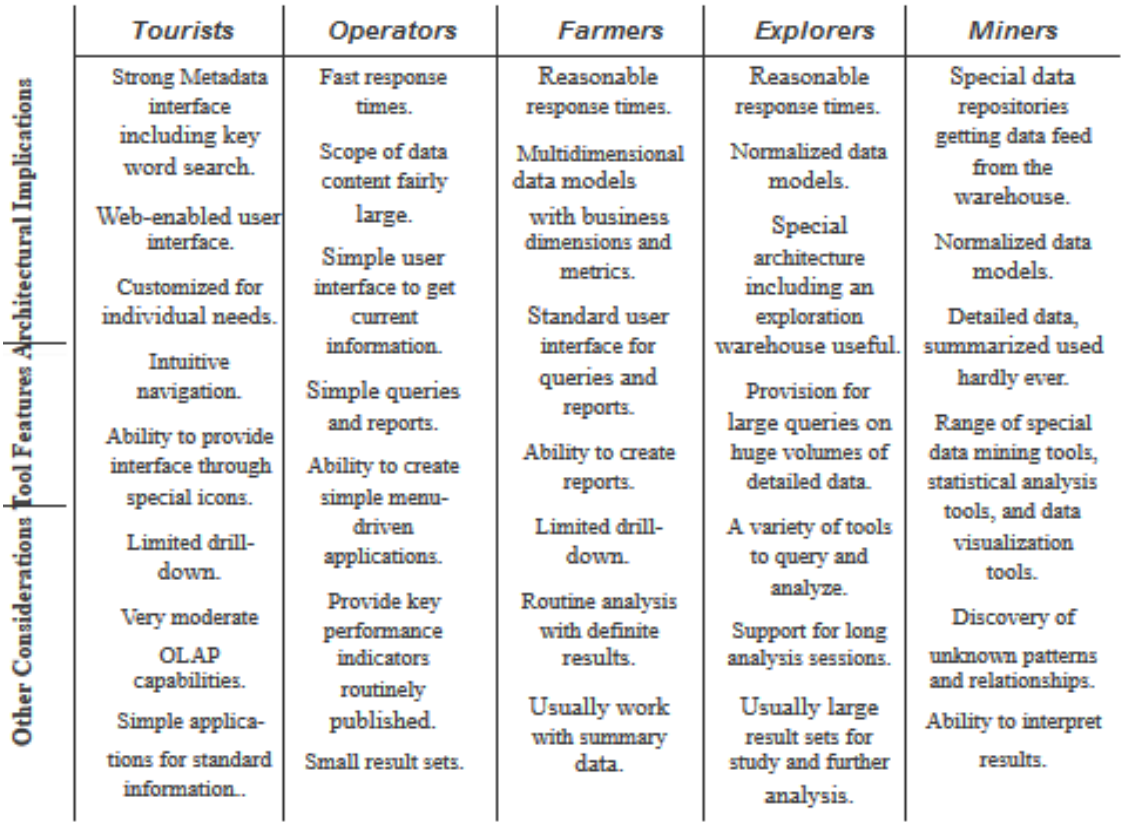

Figure 10 : How to provide information.

The first method is the delivery of information through reports. Of course, the formats and content could be sophisticated. Nevertheless, these are reports. The method of information delivery through reports is a carry-over from operational systems. You are familiar with hundreds of reports distributed from legacy operational systems. The next method is also a perpetuation of a technique from operational systems. In operational systems, the users are allowed to run queries in a very controlled setup. However, in a data warehouse, query processing is the most common method for information delivery. The types of queries run the gamut from simple to very complex. As you know, the main difference between queries in an operational system and in the data warehouse is the extra capabilities and openness in the warehouse environment.

The method of interactive analysis is something special in the data warehouse environment. Rarely are any users provided with such an interactive method in operational systems. Lastly, the data warehouse is the source for providing integrated data for down- stream decision support applications. The Executive Information System is one such application. But more specialized applications such as data mining make the data ware- house worthwhile. Figure 10. shows the comparison of information delivery methods between the data warehouse and operational systems.

Let us understand some basic features of the reporting and query environments and understand the details to be taken into account while designing these methods of information delivery.

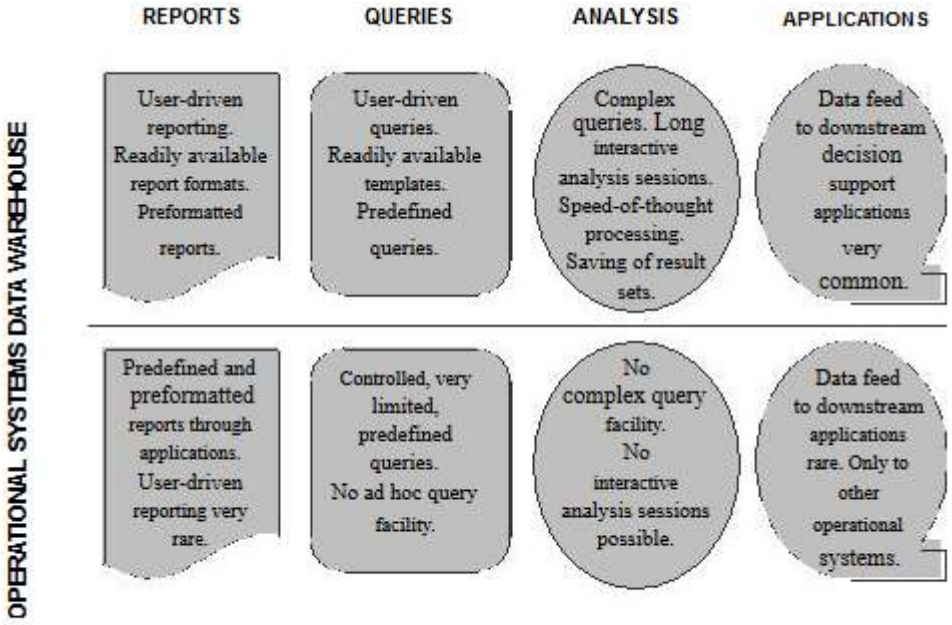

Figure 11 Information delivery: comparison between data warehouse and operational systems.

Queries

Query management ranks high in the provision of information delivery in a data ware- house. Because most of the information delivery is through queries, query management is very important. The entire query process must be managed with utmost care. First, consider the features of a managed query environment:

- Query initiation, formulation, and results presentation are provided on the client machine.
- Metadata guides the query process.
- Ability for the users to navigate easily through the data structures is absolutely essential.
- Information is pulled by the users, not pushed to them.
- Query environment must be flexible to accommodate different classes of users.

Let us look at the arena in which queries are being processed. Essentially, there are three sections in this arena. The first section deals with the users who need the query management facility. The next section is about the types of queries themselves. Finally, you have the data that resides in the data warehouse repository. This is the data that is used for the queries. Figure 11 shows the query processing arena with the three sections. Please note the features in each section. When you establish the managed query environment, take into account the features and make proper provisions for them.

Reports

Let us observe the significant features of the reporting environment in this subsection. Everyone is familiar with reports and how they are used. let us just discuss reporting services by relating these to the data warehouse. Consider the following brief list.

The information is pushed to the user, not pulled by the user as in the case of queries. Reports are published and the user subscribes to what he or she needs.

- Compared to queries, reports are inflexible and predefined.
- Most of the reports are preformatted and, therefore, rigid.
- The user has less control over the reports received than the queries he or she can formulate.

- A proper distribution system must be established.
- Report production normally happens on the server machine.

While constructing the reporting environment for our data warehouse, we need to use the following as guidelines:

Set of preformatted reports
Provide a library of preformatted reports with clear descriptions of the reports. Make it easy for users to browse through the library and select the reports they need.

Parameter-driven predefined reports
These give the users more flexibility than the preformatted ones. Users must have the capability to set their own parameters and ask for page breaks and subtotals.

Easy-to-use report development
When users need new reports in addition to preformatted or predefined reports, they must be able to develop their own reports easily with a simple report-writer facility.

Execution on the server
Run the reports on the server machine to free the client machines for other modes of information delivery.

Report scheduling
Users must be able to schedule their reports at a specified time or based on designated events.

Publishing and subscribing
Users must have options to publish the reports they have created and allow other users to subscribe and receive copies.

Delivery options
Provide various options to deliver reports including mass distribution, e-mail, the Web, automatic fax, and so on. Allow users to choose their own methods for receiving the reports.

Multiple data manipulation options
Allow the users to ask for calculated metrics, pivoting of results by interchanging the column and row variables, adding subtotals and final totals, changing the sort orders, and showing stoplight-style thresholds.

Multiple presentation options
Provide a rich variety of options including graphs, tables, columnar formats, cross-tabs, fonts, styles, sizes, and maps.

Administration of reporting environment
Ensure easy administration to schedule, monitor, and resolve problems.

Analysis

Who are the users seriously interested in analysis? Business strategists, market researchers, product planners, production analysts—in short, all the users we have classified as explorers. Because of its

rich historical data content, the data warehouse is very well suited for analysis. It provides these users with the means to search for trends, find correlations, and discern patterns.

In one sense, an analysis session is nothing but a session of a series of related queries. The user might start off with an initial query: What are the first quarter sales totals for this year by individual product lines? The user looks at the numbers and is curious about the sag in the sales of two of these product lines. The user then proceeds to drill down by individual products in those two product lines. The next query is for a breakdown by regions and then by districts. The analysis continues with comparison with the first quarterly sales of the two prior years. In analysis, there are no set predefined paths. Queries are formulated and executed at the speed of thought.

Analysis can become extremely complex, depending on what the explorer is after. The explorer may take several steps in a winding navigational path. Each step may call for large masses of data. The joins may involve several constraints. The explorer may want to view the results in many different formats and grasp the meaning of the results. Complex analysis falls in the domain of online analytical processing (OLAP).

Applications

A decision support application in relation to the data warehouse is any downstream system that gets its data feed from the data warehouse. In addition to letting the users access the data content of the warehouse directly, some companies create specialized applications for specific groups of users. Companies do this for various reasons. Some of the users may not be comfortable browsing through the data warehouse and looking for specific information. If the required data is extracted from the data warehouse at periodic intervals and specialized applications are built using the extracted data, these users have their needs satisfied.

A downstream decision support application may just start out to be nothing more than a set of preformatted and predefined reports. You add a simple menu for the users to select and run the reports and you have an application that may very well be useful to a number of your users. Executive Information Systems (EIS) are good candidates for downstream applications. EIS built with data from the warehouse proves to be superior to its counterparts of more than a decade ago when EIS were based on data from operational systems.

2.11. Information Delivery Tools

A large portion of the success of your data warehouse rests on the information delivery tools made available to the users. Selecting the right tools is of paramount importance. You have to make sure that the tools are most appropriate for your environment. If the tools are effective, usable, and enticing, your users will come to the data warehouse often. You have to select the information delivery tools with great care and thoroughness. Information delivery tools come in different formats to serve various purposes. The principal class of tools comprises query or data access tools. This class of tools enables the users to define, formulate, and execute queries and obtain results. Other types are the report writers or reporting tools for formatting, scheduling, and running reports. Other tools specialize in complex analysis. A few tools combine the different features so that your users may learn to use a single tool for queries and reports. More commonly, you will find more than one information delivery tool used in a single data warehouse environment.

Information delivery tools typically perform two functions: they translate the user requests of queries or reports into SQL statements and send these to the DBMS; they receive results from the data warehouse DBMS, format the result sets in suitable outputs, and present the results to the users. Usually, the requests to the DBMS retrieve and manipulate large volumes of data. Compared to the volumes of data retrieved, the result sets contain much lesser data.

The Desktop Environment

In the client–server computing architecture, information delivery tools run in the desktop environment. Users initiate the requests on the client machines. When you select the query tools for your information delivery component, you are choosing software to run on the client workstations. What are the basic categories of information delivery tools? Grouping the tools into basic categories broadens your understanding of what types of tools are available and what types you need for your users. Let us examine the array of information delivery tools you need to consider for selection. Please study Figure 12 carefully. This figure lists the major categories for the desktop environment and summarizes the use and purpose of each category. Note the purpose of each category. The usage and functions of each category of tools help you match the categories with the classes of users.

Methodology for Tool Selection

Because of the enormous importance of the information delivery tools in a data warehouse environment, you must have a well thought out, formalized methodology for selecting the appropriate tools. A set of tools from certain vendors may be the best for a given environment, but the same set of tools may be a total disaster in another data warehouse environment. There is no one-size-fits-all proposition in the tool selection. The tools for your environment are for your users and must be the most suitable for them. Therefore, before formalizing the methodology for selection, do reconsider the requirements of your
users.

Who are your users? At what organizational levels do they perform? What are the levels of their computing proficiency? How do they expect to interact with the data warehouse? What are their expectations? How many tourists are there? Are there any explorers at all? Ask all the pertinent questions and explore the answers.

Among the best practices in data warehouse design and development, a formal methodology ranks among
the top. A good methodology certainly includes your user representatives. Make your users part of the process. Otherwise your tool selection methodology is doomed to failure. Have the users actively involved in setting the criteria for the tools and also in the evaluation activity itself. Apart from considerations of user preferences, technical compatibility with other components of the data warehouse must also be
taken into account. Do not overlook technical aspects.

TOOL CATEGORY	PURPOSE AND USAGE
Managed Query	Query templates and predefined queries. Users supply input parameters. Users can receive results on GUI screens or as reports.
Ad Hoc Query	Users can define the information needs and compose their own queries. May use complex templates. Results on screen or reports.
Preformatted Reporting	Users input parameters in predefined report formats and submit report jobs to be run. Reports may be run as scheduled or on demand.
Enhanced Reporting	Users can create own reports using report writer features. Used for special reports not previously defined. Reports run on demand.
Complex Analysis	Users write own complex queries. Perform interactive analysis usually in long sessions. Store intermediate results. Save queries for future use.
DSS Applications	Pre-designed standard decision support applications. May be customized. Example: Executive Information System. Data from the warehouse.
Application Builder	Software to build simple downstream applications for decision support applications. Proprietary language component. Usually menu-driven.
Knowledge Discovery	Set of data mining techniques. Tools used to discover patterns and relationships not apparent or previously known.

Figure 12 Information delivery: the desktop environment.

The formal methodology you come up for the selection of tools for your environment must define the activities in each stage of the process.

Tool Selection Criteria

Lets us discuss a list of general criteria for selecting information delivery tools. This list is applicable to all the three areas. You may use the list as a guide and prepare your own checklist that is specific to your environment.

Ease of use

This is perhaps the most important way to make your users happy. Ease of use is specifically required for query creation, report building, and presentation flexibility.

Performance

Although system performance and response times are less critical in a data warehouse environment than in an OLTP system, still they rank high on the list of user requirements. Need for acceptable performance spans not only the information delivery system, but the entire environment.

Compatibility

The features of the information delivery tool must be exactly suited to the class of users it is intended for. For example, OLAP capability is not compatible with the class of users called tourists, nor are preformatted reports the precise requirement of the explorers.

Functionality

This is an extension of compatibility. The profile of every user class demands certain indispensable functions in the tool. For example, miners need a full range of functions from data capture through discovery of unknown patterns.

Integrated
More commonly, querying, analyzing, and producing reports may be tied together in one user session. The user may start with an initial query whose results lead to drilling down or other forms of analysis. At the end of the session, the user is likely to capture the final result sets in the form of reports. If this type of usage is common in your environment, your information delivery tool must be able to integrate different functions.

Tool administration
Centralized administration makes the task of the information delivery administrator easy. The tool must come with utility functions to configure and control the information delivery environment.

Web-enabled
The Internet has become our window into the world. Today's data warehouses have a big advantage over the ones built before Web technology became popular. It is important for the information delivery tool to be able to publish Web pages over the Internet and your company's intranet.

Data security
In most enterprises, safeguarding the warehouse data is as critical as providing security for data in operational systems. If your environment is a data sensitive one, the information delivery tools must have security features.

Data browsing capability
The user must be able to browse the metadata and review the data definitions and meanings. Also, the tool must present data sets as GUI objects on the screen for the user to choose by clicking on the icons.

Data selector ability
The tool must provide the user with the means for constructing queries without having to perform specific joins of tables using technical terminology and methods.

Database connectivity
The ability to connect to any of the leading database products is an essential feature needed in the information delivery tool.

Presentation feature
It is important for the tool to present the result sets in a variety of formats including texts, tabular formats, charts, graphs, maps, and so on.

Scalability
If your data warehouse is successful, you can be sure of a substantial increase in the number of users within a short time as well as a marked expansion in the complexity of the information requests. The

information delivery tool must be scalable to handle the larger volumes and extra complexity of the requests.

Vendor Dependability
As the data warehousing market matures you will notice many mergers and acquisitions. Also, some companies are likely to go out of business. The selected tool may be the best suited one for your environment, but if the vendor is not stable, you may want to rethink your selection.

2.12. Online Analytical Processing (OLAP)

As the name implies, OLAP has to do with the processing of data as it is manipulated for analysis. The data warehouse provides the best opportunity for analysis and OLAP is the vehicle for carrying out involved analysis. The data warehouse environment is also best for data access when analysis is carried out. In today's data warehousing environment, with such tremendous progress in analysis tools from various vendors, you cannot have a data warehouse without OLAP. OLAP is used to answer the complex queries posted on data warehouse. In order to solve the queries of nature 'who?' and 'what?' we can use the simple tools but to answer the advanced queries like 'what if?' and 'why?', we require special tool that can support online analytical processing (OLAP).

OLAP is defined as "The dynamic synthesis, analysis, and consolidation large volumes of multi-dimensional data."
OLAP is a term that describes a technology that uses a multi-dimensional view of aggregate data to provide quick access to strategic information for the purposes of advanced analysis. OLAP enables users to gain a deeper understanding and knowledge about various aspects of their corporate data through fast, consistent, interactive access to a wide variety of possible views of the data.

OLAP enables decision-making about future actions. Atypical OLAP calculation can be more complex than simply aggregating data, for example, 'What would be the effect on property sales in the different regions of Punjab if legal costs went up by 3.5% and Government taxes went down by 1.5% for properties over Rs 1OO,OOO?'. Analytical Queries per Minute (AQM) is used as a standard benchmark for comparison of performances of different OLAP tools. OLAP systems should as much possible hide users from the syntax of complex queries and provide consistent response times for all queries no matter how complex.

Demand For Online Analytical Processing
A data warehouse is meant for performing substantial analysis using the available data. The analysis leads to strategic decisions that are the major reasons for building data warehouses in the first place. For performing meaningful analysis, data must be cast in a way suitable for analysis of the values of key indicators over time along business dimensions. Data structures designed using the dimensional modeling technique support such analysis. In all the three approaches referred to above, the data marts rest on the dimensional model. Therefore, these data marts must be able to support dimensional analysis. In practice, these data marts seem to be adequate for basic analysis. However, in today's business conditions, we find that users need to go beyond such basic analysis. They must have the capability to perform far more complex analysis in less time. Let us examine how the traditional methods of analysis provided in a data warehouse are not sufficient and perceive what exactly is demanded by the users to stay competitive and to expand

Need for Multidimensional Analysis

Let us quickly review the business model of a large retail operation. If you just look at daily sales, you soon realize that the sales are interrelated to many business dimensions. The daily sales are meaningful only when they are related to the dates of the sales, the products, the distribution channels, the stores, the sales territories, the promotions, and a few more dimensions. Multidimensional views are inherently representative of any business model. Very few models are limited to three dimensions or less. For planning and making strategic decisions, managers and executives probe into business data through scenarios. For example, they compare actual sales against targets and against sales in prior periods. They examine the breakdown of sales by product, by store, by sales territory, by promotion, and so on. Decision makers are no longer satisfied with one-dimensional queries such as "How many units of Product A did we sell in the store in Edison, New Jersey?" Consider the following more useful query: How much revenue did the new Product X generate during the last three months, broken down by individual months, in the South Central territory, by individual stores, broken down by promotions, compared to estimates, and compared to the previous version of the product? The analysis does not stop with this single multidimensional query. The user continues to ask for further comparisons to similar products, comparisons among territories, and views of the results by rotating the presentation between columns and rows.

For effective analysis, your users must have easy methods of performing complex analysis along several business dimensions. They need an environment that presents a multidimensional view of data, providing the foundation for analytical processing through easy and flexible access to information. Decision makers must be able to analyze data along any number of dimensions, at any level of aggregation, with the capability of viewing results in a variety of ways. They must have the ability to drill down and roll up along the hierarchies of every dimension. Without a solid system for true multidimensional analysis, your data warehouse is incomplete.

In any analytical system, time is a critical dimension. Hardly any query is executed without having time as one of the dimensions along which analysis is performed. Further, time is a unique dimension because of its sequential nature—November always comes after October. Users monitor performance over time, as for example, performance this month compared to last month, or performance this month compared with performance the same month last year.

Another point about the uniqueness of the time dimension is the way in which the hierarchies of the dimension work. A user may look for sales in March and may also look for sales for the first four months of the year. In the second query for sales for the first four months, the implied hierarchy at the next higher level is an aggregation taking into account the sequential nature of time. No user looks for sales of the first four stores or the last three stores. There is no implied sequence in the store dimension. True analytical systems must recognize the sequential nature of time.

Fast Access and Powerful Calculations

Whether a user's request is for monthly sales of all products along all geographical regions or for year-to-date sales in a region for a single product, the query and analysis system must have consistent response times. Users must not be penalized for the complexity of their analysis. Both the size of the effort to formulate a query or the amount of time to receive the result sets must be consistent irrespective of the query types.

Let us take an example to understand how speed of the analysis process matters to users. Imagine a business analyst looking for reasons why profitability dipped sharply in the recent months in the entire enterprise. The analyst starts this analysis by querying for the overall sales for the last five months for the entire company, broken down by individual months. The analyst notices that although the sales do not show a drop, there is a sharp reduction in profitability for the last three months. The analysis proceeds further when the analyst wants to find out which countries show reductions. The analyst requests a breakdown of sales by major worldwide regions and notes that the European region is responsible for the reduction in profitability. Now the analyst senses that clues are becoming more pronounced and looks for a breakdown of the European sales by individual countries. The analyst finds that the profitability has increased for a few countries, decreased sharply for some other countries, and been stable for the rest. At this point, the analyst introduces another dimension into the analysis. Now the analyst wants the breakdown of profitability for the European countries by country, month, and product. This step brings the analyst closer to the reason for the decline in the profitability. The analyst observes that the countries in the European Union (EU) show very sharp declines in profitability for the last two months. Further queries reveal that manufacturing and other direct costs remain at the usual levels but the indirect costs have shot up. The analyst is now able to determine that the decline is due to the additional tax levies on some products in the EU. The analyst has also determined the exact effect of the levies so far. Strategic decisions follow on how to deal with the decline in profitability.

Note: Users certainly need the ability to perform multidimensional analysis with complex calculations, but we find that the traditional tools of report writers, query products, spreadsheets, and language interfaces are distressfully inadequate. What is the answer? Clearly, the tools being used in the OLTP and basic data warehouse environments do not match up to the task. We need different set of tools and products that are specifically meant for serious analysis. We need OLAP in the data warehouse. Let's look at the Basic virtues of OLAP which are as under:

- Enables analysts, executives, and managers to gain useful insights from the presentation of data. ☐
- Can reorganize metrics along several dimensions and allow data to be viewed from different perspectives.
- Supports multidimensional analysis. ☐
- Is able to drill down or roll up within each dimension. ☐
- Is capable of applying mathematical formulas and calculations to measures. ☐
- Provides fast response, facilitating speed-of-thought analysis. ☐
- Complements the use of other information delivery techniques such as data mining. ☐
- Improves the comprehension of result sets through visual presentations using graphs and charts. ☐
- Can be implemented on the Web. ☐
- Designed for highly interactive analysis

2.13. OLAP Features

There are the following key features of OLAP:

- Multi-dimensional views of data
- Support for complex calculations
- Time intelligence

Multi-dimensional views of data
A multi-dimensional view of data provides the basis for analytical processing through flexible access to corporate data. It enables users to analyze data across any dimension at any level of aggregation with equal functionality and ease.

Support for complex calculations
OLAP software must provide a range of powerful computational methods such as that required by sales forecasting such as moving averages and percentage growth.

Time intelligence
Time intelligence is used to judge the performance of almost any analytical application over time. For example, this month versus last month or this month versus the same month last

2.14. OLAP Models

There are two major models of OLAP referred as:

- ROLAP
- MOLAP.
- And another less used model called as DOLAP.

ROLAP stands for relational online analytical processing and MOLAP stands for multidimensional online analytical processing. In either case, the information interface is still OLAP. DOLAP stands for desktop online analytical processing. DOLAP is meant to provide portability to users of online analytical processing. In the DOLAP methodology, multidimensional datasets are created and transferred to the desktop machine, requiring only the DOLAP software to exist on that machine. DOLAP is a variation of ROLAP.

The MOLAP Model
As discussed, in the MOLAP model, data for analysis is stored in specialized multidimensional databases. Large multidimensional arrays form the storage structures. For example, to store sales number of 500 units for product ProductA, in month number 2001/01, in store StoreS1, under distributing channel Channel05, the sales number of 500 is stored in an array represented by the values (ProductA, 2001/01, StoreS1, Channel05). The array values indicate the location of the cells. These cells are intersections of the values of dimension attributes. If you note how the cells are formed, you will realize that not all cells have values of metrics. If a store is closed on Sundays, then the cells representing Sundays will all be nulls. Let us now consider the architecture for the MOLAP model. Please go over each part of Figure 13 carefully. Note the three layers in the multitier architecture. Pre calculated and prefabricated multidimensional data cubes are stored in multidimensional databases. The MOLAP engine in the application layer pushes a multidimensional view of the data from the MDDBs to the users. As mentioned earlier, multidimensional database management systems are proprietary software systems. These systems provide the capability to consolidate and fabricate summarized cubes during the process that loads data into the MDDBs from

the main data warehouse. The users who need summarized data enjoy fast response times from the pre consolidated data.

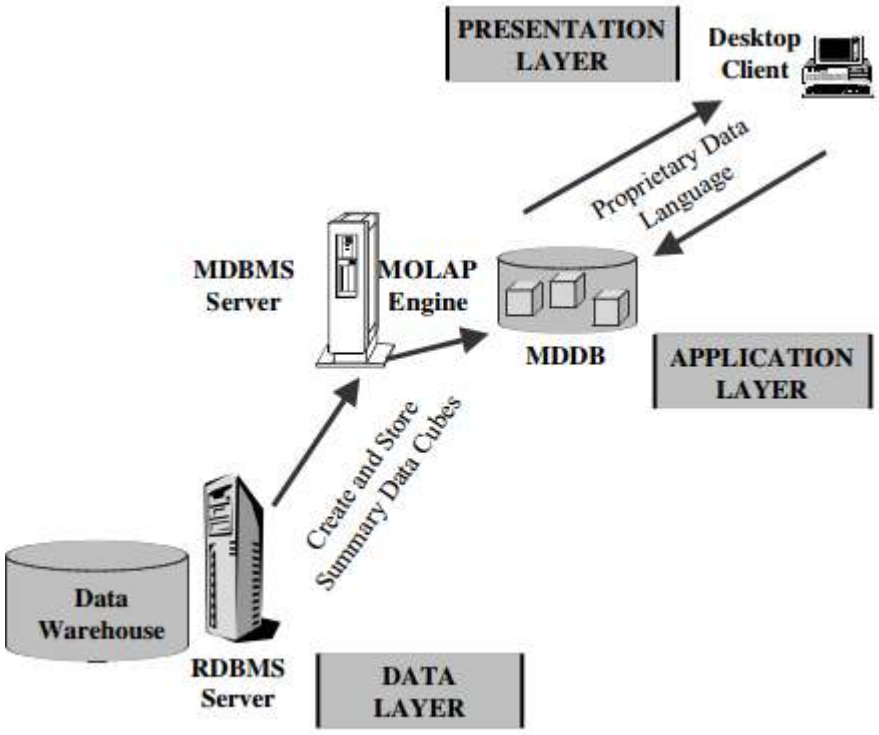

Figure 13 The MOLAP model.

The ROLAP Model

In the ROLAP model, data is stored as rows and columns in relational form. This model presents data to the users in the form of business dimensions. In order to hide the storage structure to the user and present data multidimensionality, a semantic layer of metadata is created. The metadata layer supports the mapping of dimensions to the relational tables. Additional metadata supports summarizations and aggregations. You may store the metadata in relational databases. Now see Figure 14. This figure shows the architecture of the ROLAP model. What you see is a three-tier architecture. The analytical server in the middle tier application layer creates multidimensional views on the fly. The multidimensional system at the presentation layer provides a multidimensional view of the data to the users. When the users issue complex queries based on this multidimensional view, the queries are transformed into complex SQL directed to the relational database.

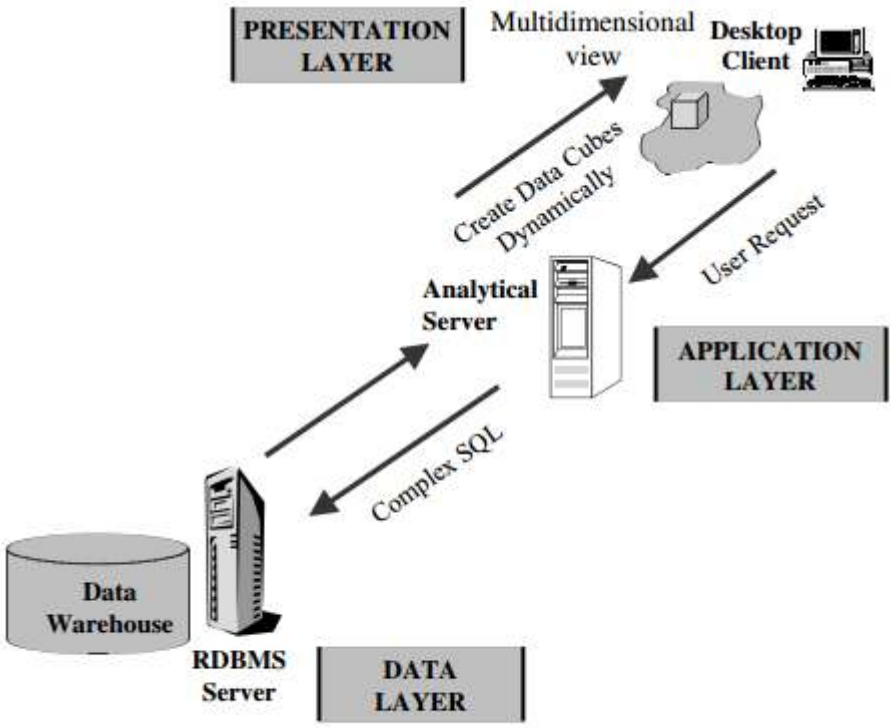

Figure 14 The ROLAP model.

Unlike the MOLAP model, static multidimensional structures are not created and stored. True ROLAP has three distinct characteristics: ☐

- Supports all the basic OLAP features and functions discussed earlier ☐
- Stores data in a relational form ☐
- Supports some form of aggregation.