Introduction to DataWarehousing

What is Data Warehouse?

A data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management's decision making process.

Subject – Oriented

- A data warehouse is organized around major subjects such as customer, supplier, product, and sales.
- Rather than concentrating on the day-to-day operations and transaction processing of an organization, a data warehouse focuses on the modeling and analysis of data for decision makers.
- Hence, data warehouses typically provide a simple and concise view of particular subject issues by excluding data that are not useful in the decision support process.

Integrated

- A data warehouse is usually constructed by integrating multiple heterogeneous sources, such as relational databases, flat files, and online transaction records.
- Data cleaning and data integration techniques are applied to ensure consistency in naming conventions, encoding structures, attribute measures, and so on.

Time-variant

- Data are stored to provide information from an historic perspective (e.g., the past 5–10 years). Every key structure in the data warehouse contains, either implicitly or explicitly, a time element.

Nonvolatile

- A data warehouse is always a physically separate store of data transformed from the application data found in the operational environment. Due to this separation, a data warehouse does not require transaction processing, recovery, and concurrency control mechanisms.
- It usually requires only two operations in data accessing: *initial loading of data* and *access of data*.

Differences between Operational Database Systems and Data Warehouses

- The major task of online operational database systems is to perform online transaction and query processing. These systems are called **online transaction processing (OLTP)** systems. They cover most of the day-to-day operations of an organization such as purchasing, inventory, manufacturing, banking, payroll, registration, and accounting.
- Data warehouse systems, on the other hand, serve users or knowledge workers in the role of data analysis and decision making. Such systems can organize and present data in various formats in order to accommodate the diverse needs of different users. These systems are known as **online analytical processing (OLAP)** systems.

Differences between OLTP and OLAP

The major distinguishing features of OLTP and OLAP are summarized as follows:

Users and system orientation:

- An OLTP system is ***customer-oriented*** and is used for transaction and query processing by clerks, clients, and information technology professionals.
- An OLAP system is ***market-oriented*** and is used for data analysis by knowledge workers, including managers, executives, and analysts.

Data contents:

- An OLTP system manages current data that, typically, are too detailed to be easily used for decision making.
- An OLAP system manages large amounts of historic data, provides facilities for summarization and aggregation, and stores and manages information at different levels of granularity. These features make the data easier to use for informed decision making.

Database design:

- An OLTP system usually adopts an entity-relationship (ER) data model and an application-oriented database design.
- An OLAP system typically adopts either a *star* or a *snowflake* model and a subject-oriented database design.

View:

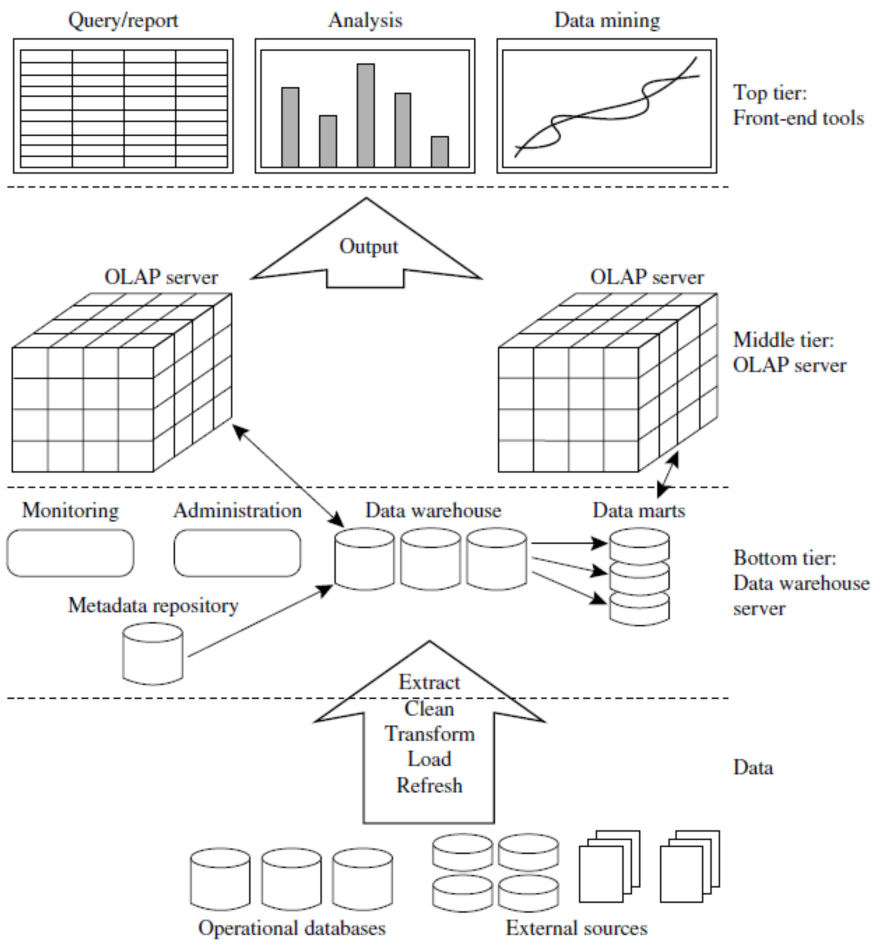
- An OLTP system focuses mainly on the current data within an enterprise or department, without referring to historic data or data in different organizations.
- An OLAP system often spans multiple versions of a database schema, due to the evolutionary process of an organization.
- OLAP systems also deal with information that originates from different organizations, integrating information from many data stores. Because of their huge volume, OLAP data are stored on multiple storage media.

Access patterns:

- The access patterns of an OLTP system consist mainly of short, atomic transactions. Such a system requires concurrency control and recovery mechanisms.
- The accesses to OLAP systems are mostly read-only operations (because most data warehouses store historic rather than up-to-date information), although many could be complex queries.

DataWarehousing: A Multitiered Architecture

Data warehouses often adopt a three-tier architecture.



I A three-tier data warehousing architecture.

Bottom Tier

- The bottom tier is a **warehouse database server** that is almost always a relational database system.
- Back-end tools and utilities are used to feed data into the bottom tier from operational databases or other external sources (e.g., customer profile information provided by external consultants). These tools and utilities perform data extraction, cleaning, and transformation (e.g., to merge similar data from different sources into a unified format), as well as load and refresh functions to update the data warehouse.
- The data are extracted using application program interfaces known as **gateways**. A gateway is supported by the underlying DBMS and allows client programs to generate SQL code to be executed at a server. Examples of gateways include ODBC (Open Database Connection) and OLEDB (Object Linking and Embedding Database) by Microsoft and JDBC (Java Database Connection).
- This tier also contains a metadata repository, which stores information about the data warehouse and its contents.

Middle Tier

- The middle tier is an **OLAP server** that is typically implemented using either a **relational OLAP(ROLAP)** model (i.e., an extended relational DBMS that maps operations on multidimensional data to standard relational operations); or a **multidimensional OLAP (MOLAP)** model (i.e., a special-purpose server that directly implements multidimensional data and operations).

Top Tier

- The top tier is a **front-end client layer**, which contains query and reporting tools, analysis tools, and/or data mining tools (e.g., trend analysis, prediction, and so on).

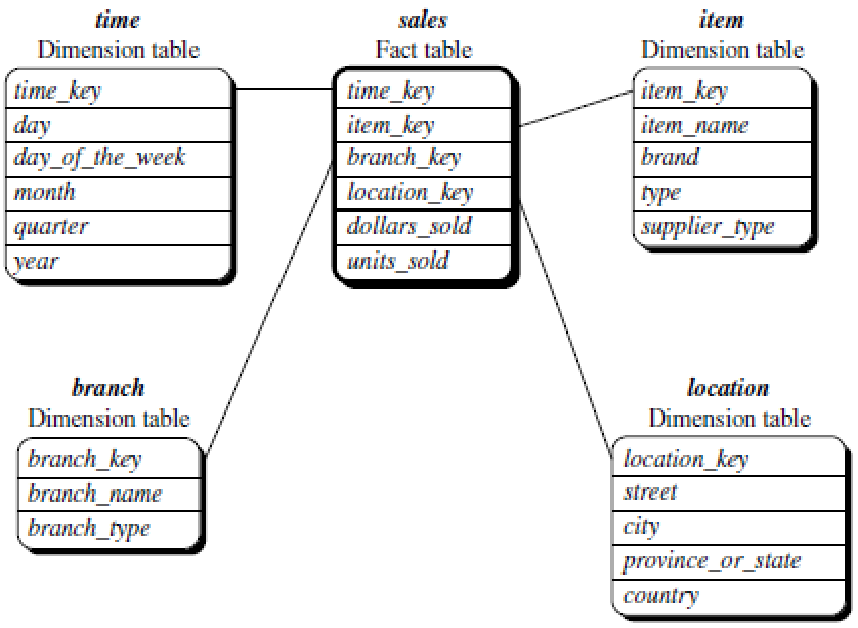
Stars, Snowflakes, and Fact Constellations
(Schemas for Multidimensional Data Models)

The entity-relationship data model is commonly used in the design of relational databases, where a database schema consists of a set of entities and the relationships between them. Such a data model is appropriate for online transaction processing.

A data warehouse, however, requires a concise, subject-oriented schema that facilitates online data analysis. The most popular data model for a data warehouse is a **multidimensional model**, which can exist in the form of a **star schema**, a **snowflake schema**, or a **fact constellation schema**.

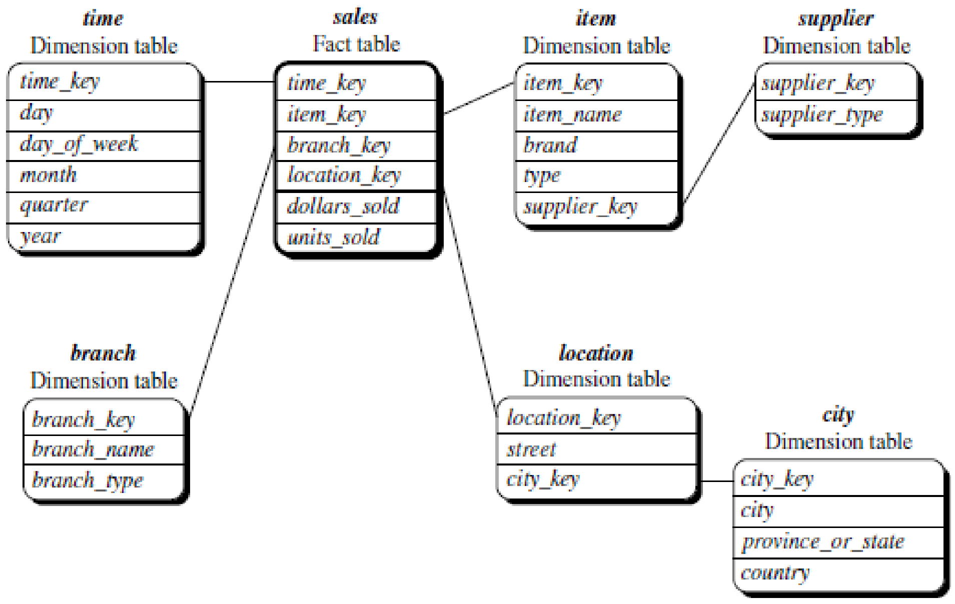
Star schema:

- The most common modeling paradigm is the star schema, in which the data warehouse contains (1) a large central table (**fact table**) containing the bulk of the data, with no redundancy, and (2) a set of smaller attendant tables (**dimension tables**), one for each dimension. The schema graph resembles a starburst, with the dimension tables displayed in a radial pattern around the central fact table.
- For example, A star schema for *AllElectronics* sales is shown in figure given below. Sales are considered along four dimensions: *time, item, branch*, and *location*. The schema contains a central fact table for *sales* that contains keys to each of the four dimensions, along with two measures: *dollars sold* and *units sold*. To minimize the size of the fact table, dimension identifiers (e.g., *time key* and *item key*) are system-generated identifiers.
- Notice that in the star schema, each dimension is represented by only one table, and each table contains 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 introduce some redundancy. For example, "Urbana" and "Chicago" are both cities in the state of Illinois, USA. Entries for such cities in the *location* dimension table will create redundancy among the attributes *province or state* and *country*; that is, ..., Urbana, IL, USA/ and ...., Chicago, IL, USA/.Moreover, the attributes within a dimension table may form either a hierarchy (total order) or a lattice (partial order).



Snowflake schema:

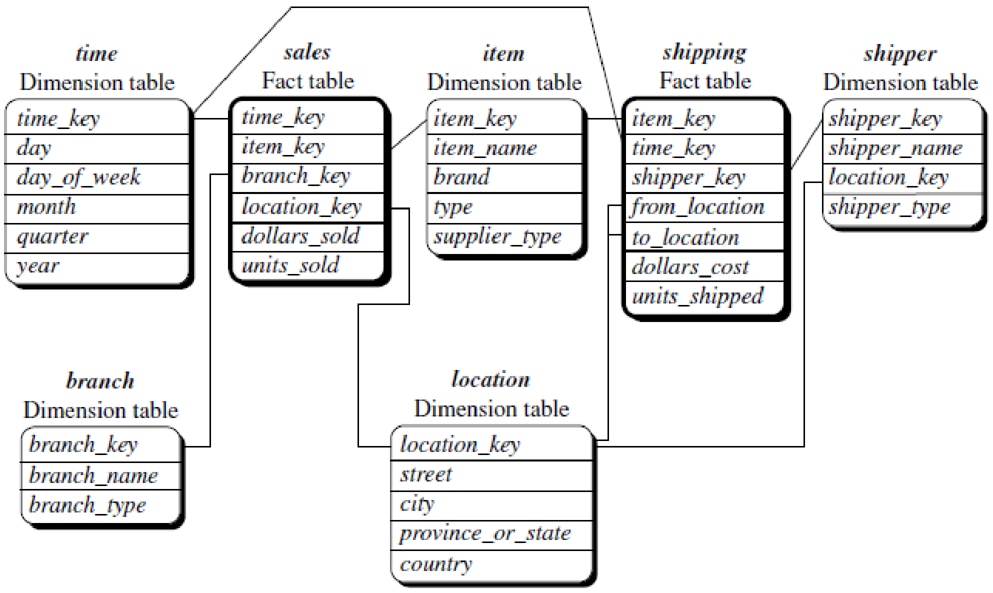
- The snowflake schema is a variant of the star schema model, where some dimension tables are *normalized*, thereby further splitting the data into additional tables. The resulting schema graph forms a shape similar to a snowflake.
- The major difference between the snowflake and star schema models is that the dimension tables of the snowflake model may be kept in normalized form to reduce redundancies. Such a table is easy to maintain and saves storage space. However, this space savings is negligible in comparison to the typical magnitude of the fact table. Furthermore, the snowflake structure can reduce the effectiveness of browsing, since more joins will be needed to execute a query. Consequently, the system performance may be adversely impacted. Hence, although the snowflake schema reduces redundancy, it is not as popular as the star schema in data warehouse design.



- The single dimension table for *item* in the star schema is normalized in the snowflake schema, resulting in new *item* and *supplier* tables. For example, the *item* dimension table now contains the attributes *item key, item name, brand, type,* and *supplier key,* where *supplier key* is linked to the *supplier* dimension table, containing *supplier key* and *supplier type* information. Similarly, the single dimension table for *location* in the star schema can be normalized into two new tables: *location* and *city*. The *city key* in the new *location* table links to the *city* dimension. Notice that, when desirable, further normalization can be performed on *province or state* and *country* in the snowflake schema shown in Figure given above.

Fact constellation:

- Sophisticated applications may require multiple fact tables to *share* dimension tables. This kind of schema can be viewed as a collection of stars, and hence is called a **galaxy schema** or a **fact constellation**.

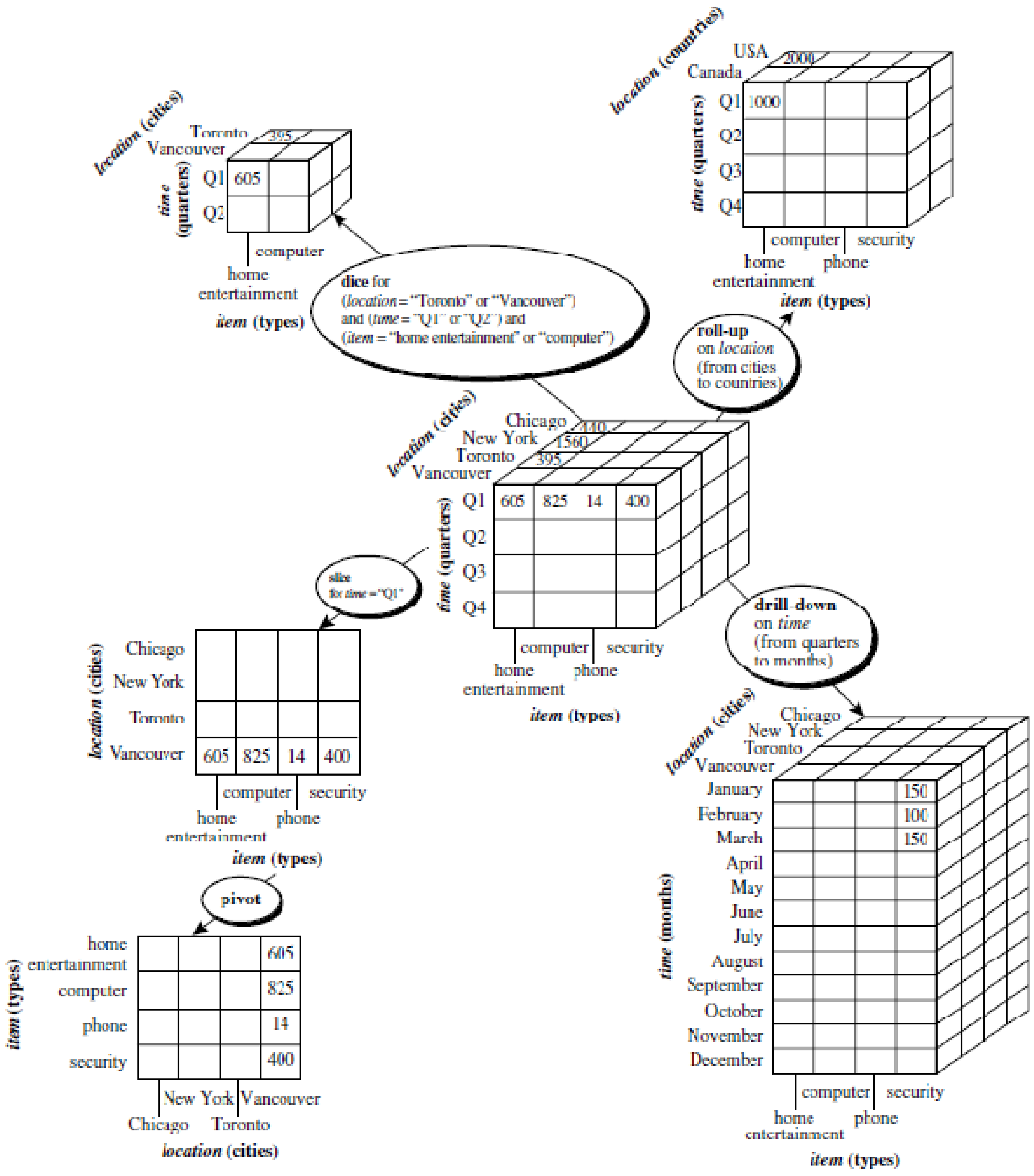


- This schema specifies two fact tables, *sales* and *shipping*. The *sales* table definition is identical to that of the star schema. The *shipping* table has five dimensions, or keys—*item key, time key, shipper key, from location,* and *to location*—and two measures—*dollars cost* and *units shipped*.
- A fact constellation schema allows dimension tables to be shared between fact tables. For example, the dimensions tables for *time, item,* and *location* are shared between the *sales* and *shipping* fact tables.

Typical OLAP Operations

- Let's look at some typical OLAP operations for multidimensional data.
- Each of the following operations described is illustrated in the Figure given below.
- At the center of the figure is a data cube for *AllElectronics* sales. The cube contains the dimensions *location, time,* and *item,* where *location* is aggregated with respect to city values, *time* is aggregated with respect to quarters, and *item* is aggregated with respect to item types.

- To aid in our explanation, we refer to this cube as the central cube. The measure displayed is *dollars sold* (in thousands). (For improved readability, only some of the cubes' cell values are shown.) The data examined are for the cities Chicago, New York, Toronto, and Vancouver.



<u>Roll-up:</u>

- The roll-up operation (also called the *drill-up* operation by some vendors) performs aggregation on a data cube, either by *climbing up a concept hierarchy* for a dimension or by *dimension reduction*.
- Figure given above shows the result of a roll-up operation performed on the central cube by climbing up the concept hierarchy for *location*. This hierarchy was defined as the total order "*street* < *city* < *province_or_state* < *country*."
- The roll-up operation shown aggregates the data by ascending the *location* hierarchy from the level of *city* to the level of *country*. In other words, rather than grouping the data by city, the resulting cube groups the data by country.
- When roll-up is performed by dimension reduction, one or more dimensions are removed from the given cube. For example, consider a sales data cube containing only the *location* and *time* dimensions. Roll-up may be performed by removing, say, the *time* dimension, resulting in an aggregation of the total sales by location, rather than by location and by time.

<u>Drill-down:</u>

- Drill-down is the reverse of roll-up.
- It navigates from less detailed data to more detailed data. Drill-down can be realized by either *stepping down a concept hierarchy* for a dimension or *introducing additional dimensions*.
- Figure given above shows the result of a drill-down operation performed on the central cube by stepping down a concept hierarchy for *time* defined as "*day* < *month* < *quarter* < *year*."
- Drill-down occurs by descending the *time* hierarchy from the level of *quarter* to the more detailed level of *month*. The resulting data cube details the total sales per month rather than summarizing them by quarter.
- Because a drill-down adds more detail to the given data, it can also be performed by adding new dimensions to a cube. For example, a drill-down on the central cube of Figure given above can occur by introducing an additional dimension, such as *customer group*.

<u>Slice and dice:</u>

- The *slice* operation performs a selection on one dimension of the given cube, resulting in a subcube.
- Figure given above shows a slice operation where the sales data are selected from the central cube for the dimension *time* using the criterion *time* = "Q1."
- The *dice* operation defines a subcube by performing a selection on two or more dimensions. Figure given above shows a dice operation on the central cube based on the following selection criteria that involve three dimensions: (*location* = "Toronto" or "Vancouver") and (*time* = "Q1" or "Q2") and (item = "home entertainment" or "computer").

<u>Pivot (rotate):</u>

- *Pivot* (also called *rotate*) is a visualization operation that rotates the data axes in view to provide an alternative data presentation.
- Figure given above shows a pivot operation where the *item* and *location* axes in a 2-D slice are rotated.

- Other examples include rotating the axes in a 3-D cube, or transforming a 3-D cube into a series of 2-D planes.

Types of OLAP servers

- OLAP servers present business users with multidimensional data from data warehouses or data marts, without concerns regarding how or where the data are stored.
- However, the physical architecture and implementation of OLAP servers must consider data storage issues. Implementations of a warehouse server for OLAP processing include the following:

Relational OLAP (ROLAP) servers:

- These are the intermediate servers that stand in between a relational back-end server and client front-end tools.
- They use a *relational* or *extended-relational DBMS* to store and manage warehouse data, and OLAP middleware to support missing pieces.
- ROLAP servers include optimization for each DBMS back end, implementation of aggregation navigation logic, and additional tools and services.
- ROLAP technology tends to have greater scalability than MOLAP technology.
- The DSS server of Micro strategy, for example, adopts the ROLAP approach.

Multidimensional OLAP (MOLAP) servers:

- These servers support multidimensional data views through *array-based multidimensional storage engines*.
- They map multidimensional views directly to data cube array structures. The advantage of using a data cube is that it allows fast indexing to pre computed summarized data.
- Notice that with multidimensional data stores, the storage utilization may be low if the data set is sparse. In such cases, sparse matrix compression techniques should be explored.
- Many MOLAP servers adopt a two-level storage representation to handle dense and sparse data sets: Denser sub cubes are identified and stored as array structures, whereas sparse sub cubes employ compression technology for efficient storage utilization.

Hybrid OLAP (HOLAP) servers:

- The hybrid OLAP approach combines ROLAP and MOLAP technology, benefiting from the greater scalability of ROLAP and the faster computation of MOLAP. For example, a HOLAP server may allow large volumes of detailed data to be stored in a relational database, while aggregations are kept in a separate MOLAP store.
- The Microsoft SQL Server 2000 supports a hybrid OLAP server.

Metadata of Data Warehouse

- **Metadata** are data about data. When used in a data warehouse, metadata are the data that define warehouse objects.
- Metadata are created for the data names and definitions of the given warehouse.
- Additional metadata are created and captured for timestamping any extracted data, the source of the extracted data, and missing fields that have been added by data cleaning or integration processes.
- A metadata repository should contain the following:

- A description of the *data warehouse structure*, which includes the warehouse schema, view, dimensions, hierarchies, and derived data definitions, as well as data mart locations and contents.
- *Operational metadata*, which include data lineage (history of migrated data and the sequence of transformations applied to it), currency of data (active, archived, or purged), and monitoring information (warehouse usage statistics, error reports, and audit trails).
- The *algorithms used for summarization*, which include measure and dimension definition algorithms, data on granularity, partitions, subject areas, aggregation, summarization, and predefined queries and reports.
- *Mapping from the operational environment to the data warehouse*, which includes source databases and their contents, gateway descriptions, data partitions, data extraction, cleaning, transformation rules and defaults, data refresh and purging rules, and security (user authorization and access control).
- *Data related to system performance*, which include indices and profiles that improve data access and retrieval performance, in addition to rules for the timing and scheduling of refresh, update, and replication cycles.
- *Business metadata*, which include business terms and definitions, data ownership information, and charging policies.

Enhancement to Group by clause: Group by using Rollup and Cube

The **GROUP BY clause** is a SQL command that is used to **group** rows that have the same values. The **GROUP BY clause** is used in the SELECT statement . Optionally it is used in conjunction with aggregate functions to produce summary reports from the database.

Group by using ROLLUP

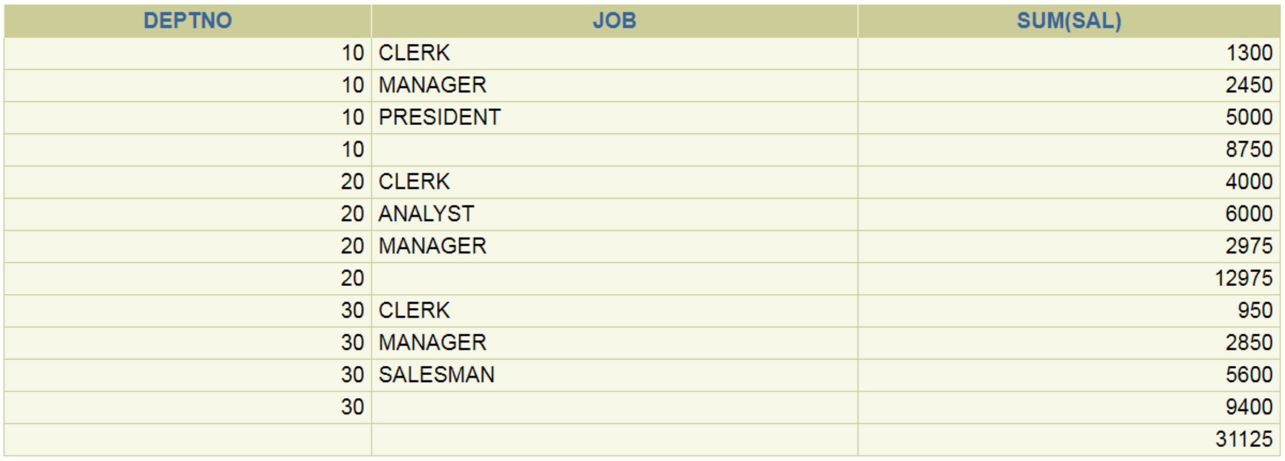
- The ROLLUP is an extension of the GROUP BY clause.
- The ROLLUP option allows you to include extra rows that represent the subtotals, which are commonly referred to as super-aggregate rows, along with the grand total row.
- By using the ROLLUP option, you can use a single query to generate multiple grouping sets.
- The following illustrates the basic syntax of the SQL ROLLUP:

```
SELECT
   c1, c2, aggregate_function(c3)
FROM
   table
GROUP BY ROLLUP (c1, c2);
```

- The ROLLUP assumes a hierarchy among the input columns. For example, if the input column is (c1,c2), the hierarchy c1 > c2. The ROLLUP generates all grouping sets that make sense considering this hierarchy. This is why we often use ROLLUP to generate the subtotals and the grand total for reporting purposes.

- In the syntax above, ROLLUP(c1,c2) generates three following grouping sets:
  (c1,c2)
  (c1)
  ( )

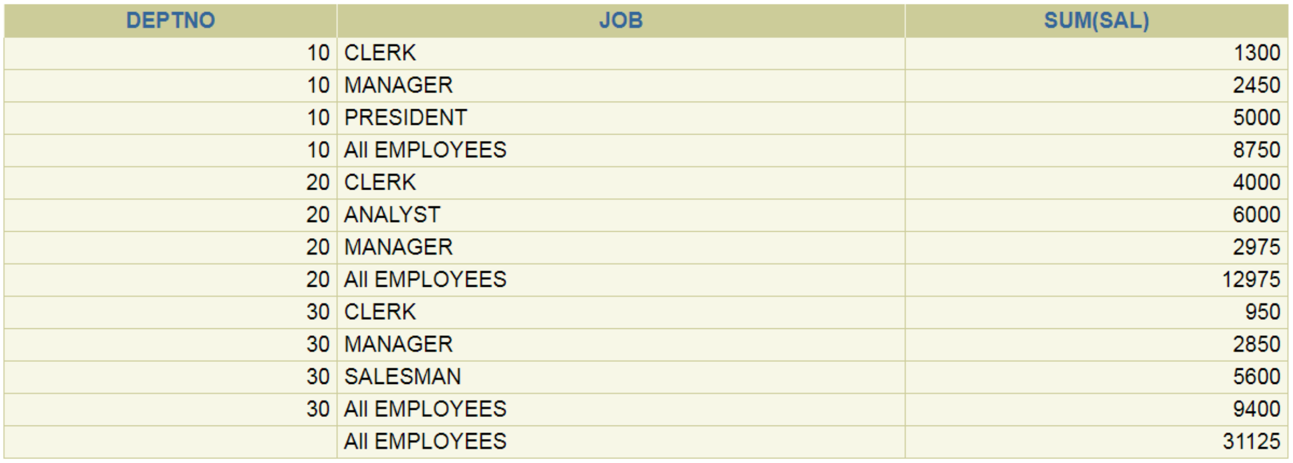
For example,
SELECT DEPTNO,JOB,SUM(SAL)
FROM EMP
GROUP BY ROLLUP (DEPTNO,JOB)



DEPTNO	JOB	SUM(SAL)
10	CLERK	1300
10	MANAGER	2450
10	PRESIDENT	5000
10		8750
20	CLERK	4000
20	ANALYST	6000
20	MANAGER	2975
20		12975
30	CLERK	950
30	MANAGER	2850
30	SALESMAN	5600
30		9400
		31125

- As you can see in the result, the NULL value in the JOB column specifies the grand total super-aggregate line. In this example, the ROLLUP option causes the query to produce another row that shows the total salary in all departments.
- To make the output more readable, you can use the COALESCE() function to substitute the NULL value by the "ALL EMPLOYEES" as follows:

SELECT DEPTNO, COALESCE(JOB, 'All EMPLOYEES') AS JOB,
SUM(SAL)
FROM EMP
GROUP BY ROLLUP (DEPTNO,JOB)



DEPTNO	JOB	SUM(SAL)
10	CLERK	1300
10	MANAGER	2450
10	PRESIDENT	5000
10	All EMPLOYEES	8750
20	CLERK	4000
20	ANALYST	6000
20	MANAGER	2975
20	All EMPLOYEES	12975
30	CLERK	950
30	MANAGER	2850
30	SALESMAN	5600
30	All EMPLOYEES	9400
	All EMPLOYEES	31125

Group by using CUBE

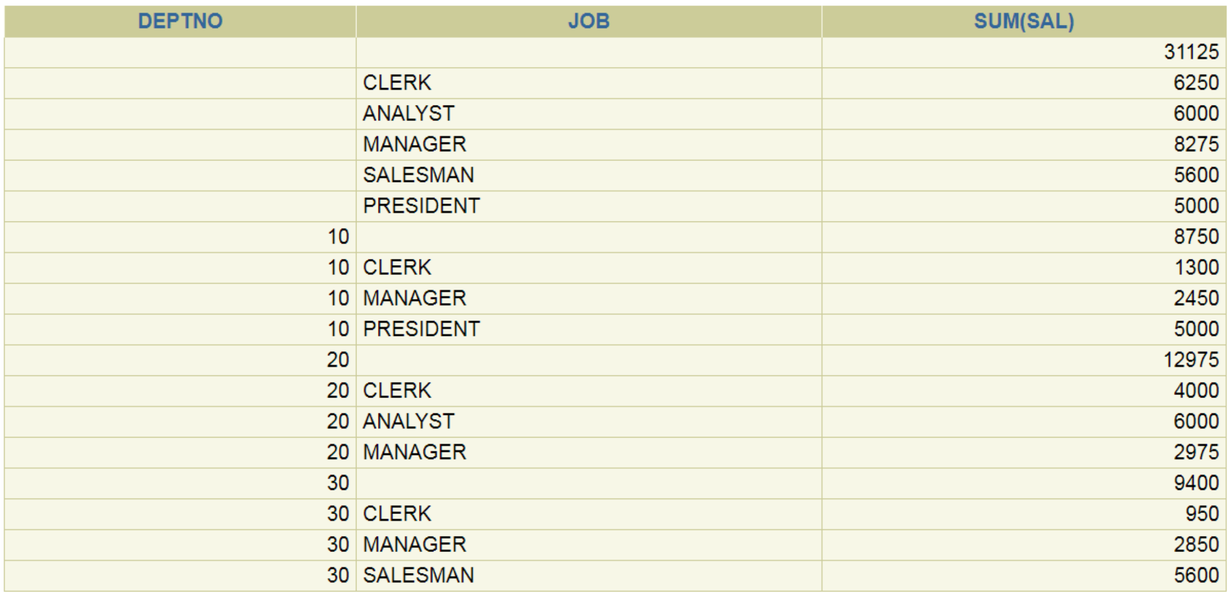
- Similar to the ROLLUP, CUBE is an extension of the GROUP BY clause.
- CUBE allows you to generate subtotals like the ROLLUP extension. In addition, the CUBE extension will generate subtotals for all combinations of grouping columns specified in the GROUP BY clause.
- The following illustrates the syntax of CUBE extension:

```
SELECT
   c1, c2, AGGREGATE_FUNCTION(c3)
FROM
   table_name
GROUP BY CUBE(c1 , c2)
```

- In this syntax, we have two columns specified in the CUBE. The statement creates two subtotal combinations. Generally, if you have n number of columns listed in the CUBE, the statement will create 2^n subtotal combinations.

For example,

```
SELECT   DEPTNO, JOB, SUM(SAL)
FROM EMP
GROUP BY CUBE (DEPTNO, JOB)
```



DEPTNO	JOB	SUM(SAL)
		31125
	CLERK	6250
	ANALYST	6000
	MANAGER	8275
	SALESMAN	5600
	PRESIDENT	5000
10		8750
10	CLERK	1300
10	MANAGER	2450
10	PRESIDENT	5000
20		12975
20	CLERK	4000
20	ANALYST	6000
20	MANAGER	2975
30		9400
30	CLERK	950
30	MANAGER	2850
30	SALESMAN	5600