UNIT – III

Data Warehouse and OLAP Technology

Objective:

To introduce the concepts of Data warehousing and Data mining.

Syllabus:

3.1 Data warehouse: Basic concepts,
3.2 OLAP vs. OLTP;
3.3 Data warehousing: A multitiered architecture; Datawarehouse modelling:
3.4 Data cube: A multidimensional data model, star, snowflake and fact constellation schemas for multidimensional data models,
3.5 Role of concept hierarchies,
3.6 Typical OLAP operations

Learning Outcomes:

At the end of the unit, students will be able to:

CO3 : Illustrate the major concepts and operations of multi dimensional data models.

Learning Material

3.1 Data warehouse:

- Data warehousing provides architectures and tools for business executives to systematically organize, understand, and use their data to make strategic decisions.
- Loosely speaking, a data warehouse refers to a data repository that is maintained separately from an organization's operational databases.

- **Definition**: According to William H. Inmon, "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, vendor, product, and sales. A data warehouses typically provide a simple and concise view around 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 on-line transaction records. Data cleaning and data integration techniques are applied to ensure consistency of the data.
- **Time-variant:** Data are stored to provide information from a historical perspective (e.g., the past 5-10 years). Every key structure in the data warehouse contains, either implicitly or explicitly, an element of **time**.
- **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.

3.1.1. The traditional database approach to heterogeneous database integration

1) The traditional database approach to heterogeneous database integration is to build **wrappers** and **integrators** (or **mediators**) on top of multiple, heterogeneous databases.
2) When a query is posed to a client site, a metadata dictionary is used to translate the query into queries appropriate for the individual heterogeneous sites involved.
3) These queries are then mapped and sent to local query processors. The results returned from the different sites are integrated into a global answer set.

4) This **query-driven approach** requires complex information filtering and integration processes, and competes with local sites for processing resources.
5) It is inefficient and potentially expensive for frequent queries, especially queries requiring aggregations.

3.1.2 update driven approach:

1) Rather than using a query-driven approach, data warehousing employs an **updatedriven** approach in which information from multiple, heterogeneous sources is integrated in advance and stored in a warehouse for direct querying and analysis.
2) Unlike online transaction processing databases, data warehouses do not contain the most current information.
3) However, a data warehouse brings high performance to the integrated heterogeneous database systembecause data are copied, preprocessed, integrated, annotated, summarized, and restructured into one semantic data store.
4) Furthermore, query processing in data warehouses does not interfere with the processing at local sources.
5) Moreover, data warehouses can store and integrate historic information and support complex multidimensional queries. As a result, data warehousing has become popular in industry.

A model data warehouse of All Electronics is as follows:

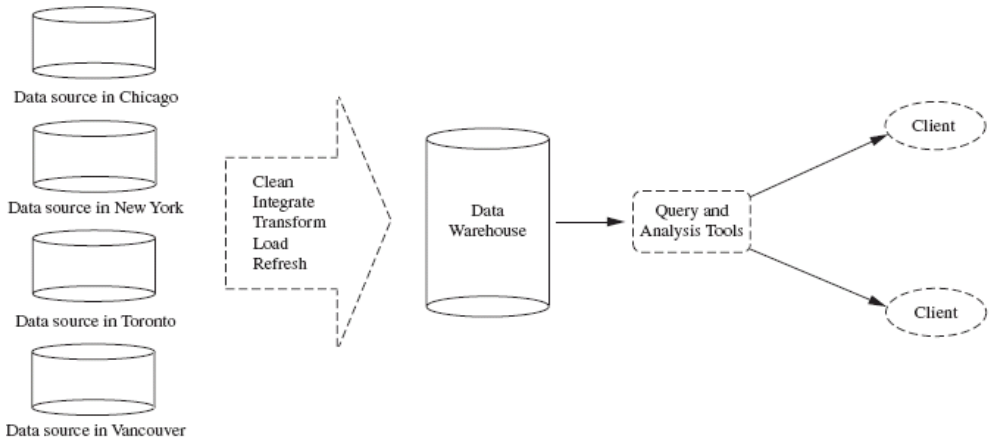


Fig : Data Warehousing and OLAP

3.2 OLAP vs. OLTP:

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.

3.2.1 Differences between OLTP and OLAP

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

2) **Data contents**:

a. An OLTP system manages current data that, typically, are too detailed to be easily used for decision making. (OLTP) systems cover most of the day-to-day operations of an organization, such as purchasing, inventory, manufacturing, banking, payroll, registration, and accounting.

b. 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.

3) Database design:

a. An OLTP system usually adopts an entity-relationship (ER) data model and an application-oriented database design.

b. An OLAP system typically adopts either a *star* or a *snowflake* model and a subject-oriented database design.

4) View:

a. An OLTP system focuses mainly on the current data within an enterprise or department, without referring to historic data or data in different organizations.

b. 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 fromdifferent organizations, integrating information from many data stores. Because of their huge volume, OLAP data are stored on multiple storage media.

5) Access patterns:

a. The access patterns of an OLTP system consist mainly of short, atomic transactions. Such a system requires concurrency control and recovery mechanisms.

b. 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.

Comparison between OLTP and OLAP systems:

Feature	*OLTP*	*OLAP*
Characteristic	operational processing	informational processing
Orientation	transaction	analysis
User	clerk, DBA, database professional	knowledge worker (e.g., manager, executive, analyst)
Function	day-to-day operations	long-term informational requirements, decision support
DB design	ER based, application-oriented	star/snowflake, subject-oriented
Data	current; guaranteed up-to-date	historical; accuracy maintained over time
Summarization	primitive, highly detailed	summarized, consolidated
View	detailed, flat relational	summarized, multidimensional
Unit of work	short, simple transaction	complex query
Access	read/write	mostly read
Focus	data in	information out
Operations	index/hash on primary key	lots of scans
Number of records accessed	tens	millions
Number of users	thousands	hundreds
DB size	100 MB to GB	100 GB to TB
Priority	high performance, high availability	high flexibility, end-user autonomy
Metric	transaction throughput	query throughput, response time

3.2.2 Why Have a Separate Data Warehouse?

A data warehouses is kept **separate** from operational databases due to the following reasons:

- An operational database is constructed for **well-known tasks** and workloads such as searching particular records, indexing, etc. In contrast, data warehouse queries are often **complex** and they present a general form of data.

- Operational databases support concurrent processing of multiple transactions. Concurrency control and recovery mechanisms are required for operational databases to ensure robustness and consistency of the database.
- An operational database query allows to **read and modify** operations, while an OLAP query needs only **read only** access of stored data.
- A data warehouse maintains historical data whereas operational databases do not typically maintain historical data.
- Decision support requires consolidation (such as aggregation and summarization) of data from heterogeneous sources, resulting in high-quality, clean, and integrated data. In contrast, operational databases contain only detailed raw data, such as transactions, which need to be consolidated before analysis.
- Since the two systems provide quite different functionalities and require different kinds of data it is presently necessary to maintain separate databases.

3.3 DataWarehousing: A Multitiered Architecture:

1. The bottom tier is a **warehouse database server** that is almost always a relational database system.

a. 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).

b. Back-end 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.

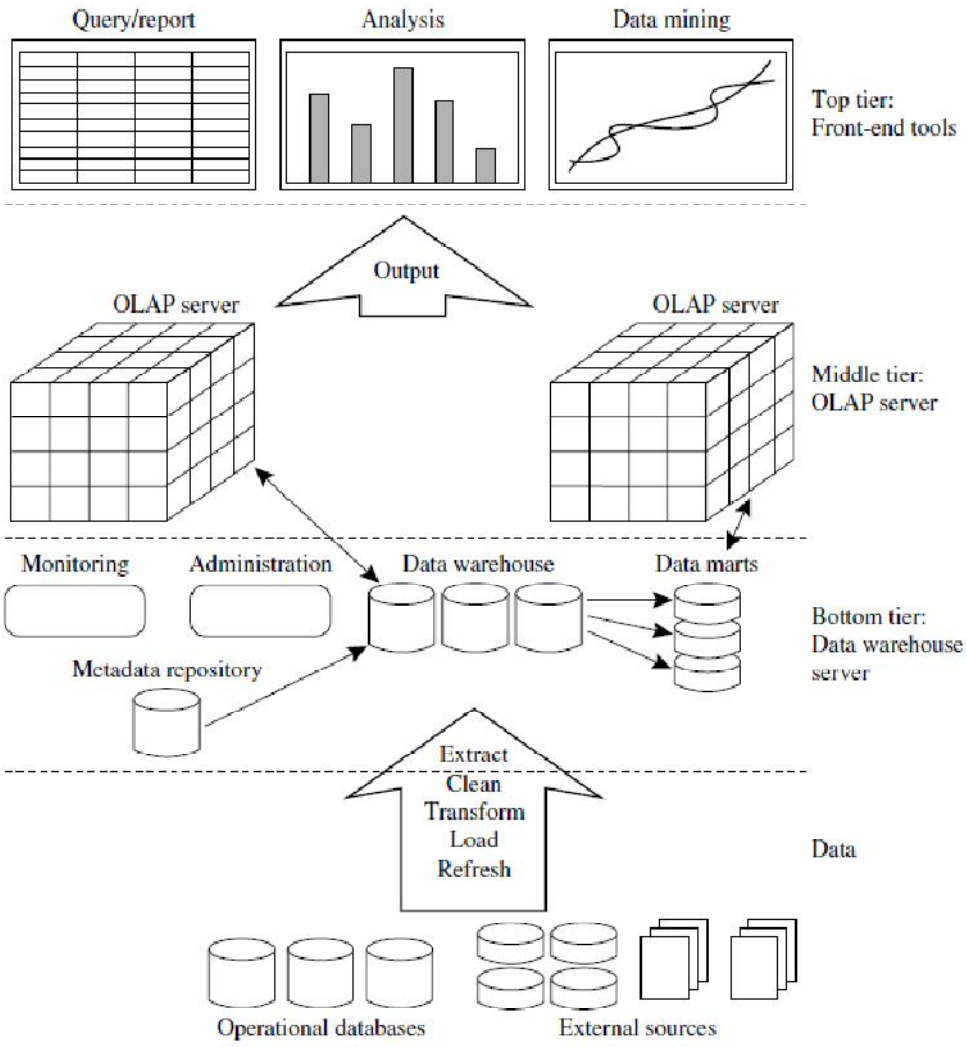


Figure 4.1 A three-tier data warehousing architecture.

c. The data are extracted using application program interfaces known as gateways.

d. A gateway is supported by the underlying DBMS and allows client programs to generate SQL code to be executed at a server.

e. Examples of gateways include ODBC (Open Database Connection) and OLEDB (Object Linking and Embedding Database) by Microsoft and JDBC (Java Database Connection).

f. This bottom tier also contains a metadata repository, which stores information about t he data warehouse and its contents.

2. 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).

3. 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).

3.4 DataWarehouse Modeling: Data Cube and OLAP

Data warehouses and OLAP tools are based on a multidimensional data model. This model views data in the form of a data cube.

3.4.1 Data Cube: A Multidimensional Data Model

a. *"What is a data cube?"* A **data cube** allows data to be modeled and viewed in multiple dimensions. It is defined by dimensions and facts.

b. **Dimensions** are the perspectives or entities with respect to which an organization wants to keep records. For example, *AllElectronics* may create a *sales* data warehouse in order to keep records of the store's sales with respect to the **dimensions** *time*, *item*, *branch*, and *location*.

c. These dimensions allow the store to keep track of things like monthly sales of items and the branches and locations at which the items were sold.

d. Each dimension may have a table associated with it, called a **dimension table**, which further describes the dimension. For example, a dimension table for *item* may contain the attributes *item name, brand*, and *type*.

e. Dimension tables can be specified by users or experts, or automatically generated and adjusted based on data distributions.

f. A multidimensional data model is typically organized around **a central theme**, such as *sales*. This theme is represented by a **fact table**.

g. **Facts** are numeric measures. Think of them as the quantities by which we want to analyze relationships between dimensions.

h. Examples of **facts** for a sales data warehouse include *dollars sold* (sales amount in dollars), *units sold* (number of units sold), and *amount budgeted*.

i. The **fact table** contains the names of the *facts*, or measures, as well as keys to each of the related dimension tables.

j. Although we usually think of cubes as 3-D geometric structures, in data warehousing the data cube is ***n*-dimensional**.

From Tables and Spreadsheets to Data Cubes

- ✓ In particular, we will look at the ***All Electronics*** sales data for items sold per quarter in the city of Vancouver.
- ✓ These data are shown in Table (a). In this 2-D representation, the sales for Vancouver are shown with respect to the *time* dimension (organized in quarters) and the *item* dimension (organized according to the types of items sold). The fact or measure displayed is *dollars sold* (in thousands).

Table(a) A 2-D view of sales data for *AllElectronics* according to the dimensions *time* and *item*, where the sales are from branches located in the city of Vancouver. The measure displayed is *dollars_sold* (in thousands).

location = "Vancouver"				
	item (type)			
time (quarter)	home entertainment	computer	phone	security
Q1	605	825	14	400
Q2	680	952	31	512
Q3	812	1023	30	501
Q4	927	1038	38	580

✓ Suppose that we would now like to view our sales data with a third dimension for instance, suppose we would like to view the data according to time ,item, as well as location for the cities Vancouver, New Yark, Chicago, Toronto. These 3-D data shown Table(b). The Table(b) are represented as a series of 2-D tables. Conceptually ,we may also represent the same data in the form of 3-D data cube, as in figure(a)

Table(b) A 3-D view of sales data for *AllElectronics*, according to the dimensions *time*, *item*, and *location*. The measure displayed is *dollars_sold* (in thousands).

	location = "Chicago"				location = "New York"				location = "Toronto"				location = "Vancouver"			
	item				item				item				item			
time	home ent.	comp.	phone	sec.	home ent.	comp.	phone	sec.	home ent.	comp.	phone	sec.	home ent.	comp.	phone	sec.
Q1	854	882	89	623	1087	968	38	872	818	746	43	591	605	825	14	400
Q2	943	890	64	698	1130	1024	41	925	894	769	52	682	680	952	31	512
Q3	1032	924	59	789	1034	1048	45	1002	940	795	58	728	812	1023	30	501
Q4	1129	992	63	870	1142	1091	54	984	978	864	59	784	927	1038	38	580

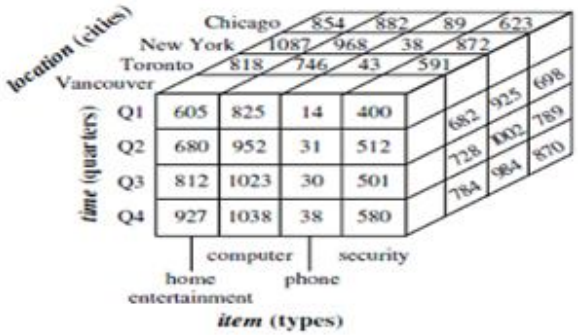
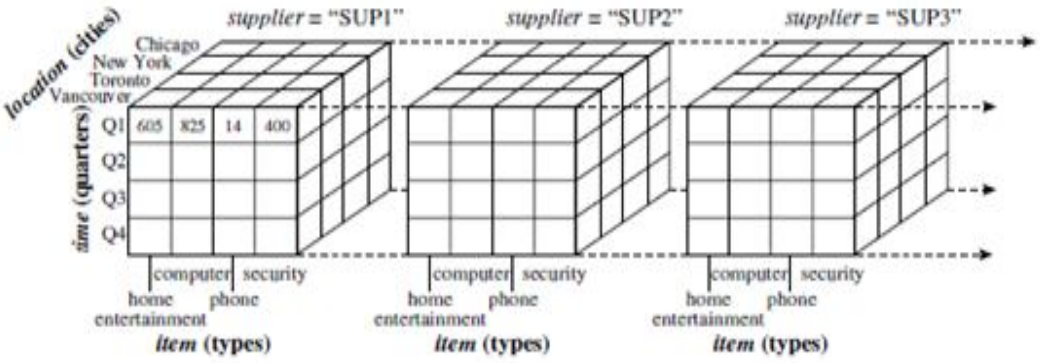


Figure (a) A 3-D data cube representation of the data in Table **(b)** according to the dimensions *time*, *item*, and *location*. The measure displayed is *dollars_sold* (in thousands).

- ✓ Suppose that we would now like to view our sales data with an additional fourth dimension, such as *supplier*. Viewing things in 4-D becomes tricky. However, we can think of a 4-D cube as being a series of 3-D cubes, as shown in figure(b) .

- ✓ If we continue in this way, we may display any *n*-D data as a series of (*n*-1)-D "cubes."



Figure(b) A 4-D data cube representation of sales data, according to the dimensions *time*, *item*, *location*, and *supplier*. The measure displayed is *dollars_sold* (in thousands). For improved readability, only some of the cube values are shown.

- Given a set of dimensions, we can generate a cuboid for each of the possible subsets of the given dimensions. The result would form a lattice of cuboids, each showing the data at a different level of summarization, or group by. The lattice of cuboids is then referred to as a data cube. Figure (c) shows a lattice of cuboids forming a data cube for the dimensions time, item, location, and supplier.
- For example, the 4-D cuboid in Figure (b) is the base cuboid for the given time, item, location, and supplier dimensions. Figure (a) is a 3-D (non base) cuboid for time, item, and location, summarized for all suppliers
- The cuboid that holds the lowest level of summarization is called the "**base cuboid"**, The 0-D cuboid, which holds the highest level of summarization, is called the "**apex cuboid".**

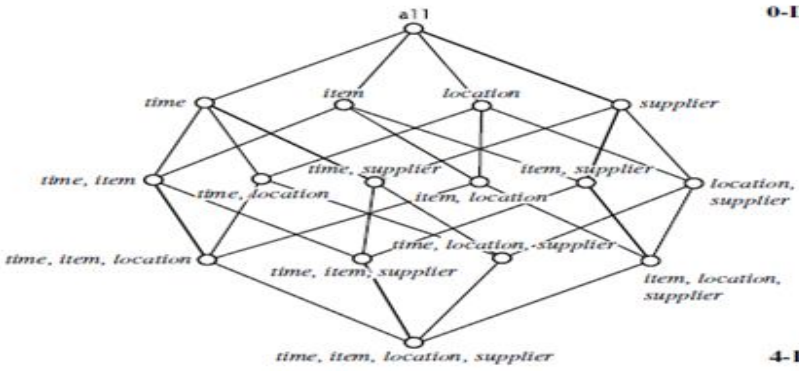


Figure (c) Lattice of cuboids, making up a 4-D data cube for the dimensions *time*, *item*, *location*, and *supplier*. Each cuboid represents a different degree of summarization.

3.4.2 Schemas for Multidimensional Databases- Stars, Snowflakes, and Fact Constellations:

The most popular data model for a data warehouse is a multidimensional model. Such a model can exist in the form of

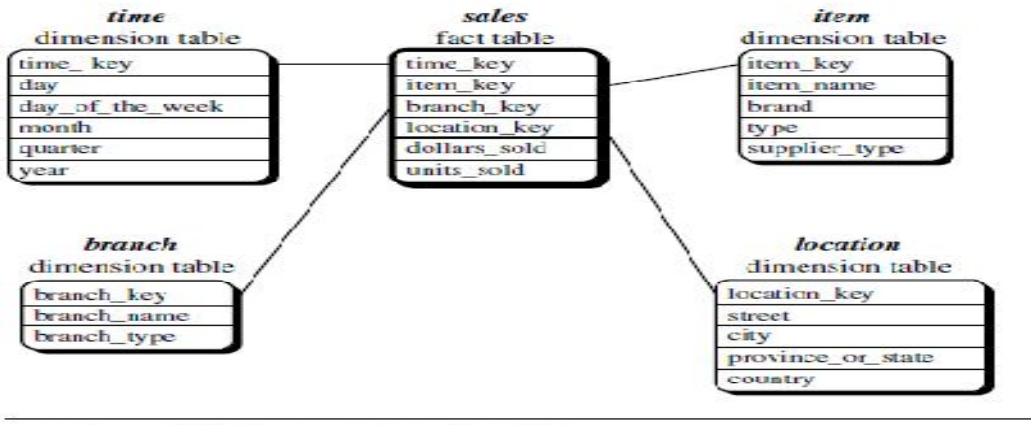
1) a star schema, 2) a snowflake schema, or 3) a fact constellation schema.

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(key and measures), with number of redundancy,
2. and a set of smaller attendant tables (dimension tables), one for each dimension. And each dimension table is joined to the fact table using primary key to foreign key join but dimension table are not joined to each other.

- ✓ It is also known as Star Join Schema.
- ✓ It is simplest style of data warehouse schema.
- ✓ It is called a Star Schema because the entity relationship diagram of schema resembles a star, with points radiating from central table.
- ✓ **Example 3.1 Star schema.** A star schema for *AllElectronics* sales is shown in Figure. 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.



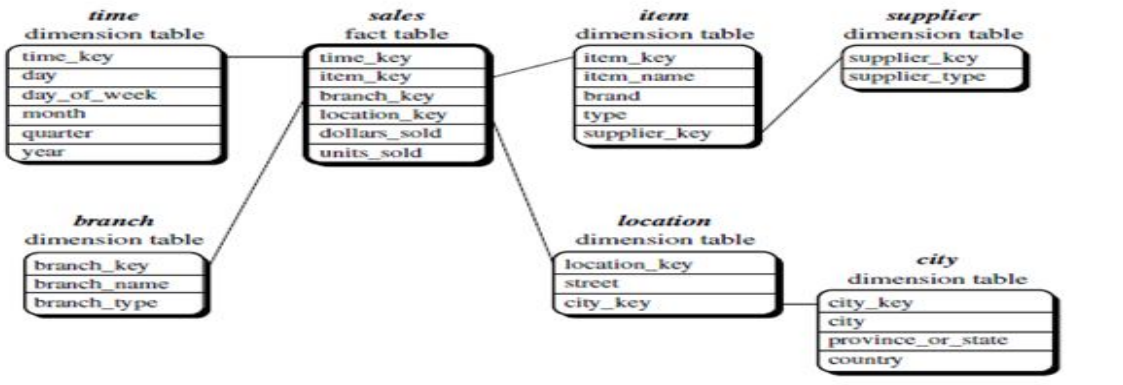
Star schema of a data warehouse for sales.

Advantages of Star Schema:

- Provide highly optimized performance for typical star queries.
- Provide a direct and intuitive mapping between the business entities being analyzed end uses and the schema design

Snowflake schema:

1. 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.
2. The resulting schema graph forms a shape similar to a snowflake..
3. The Snow flaking is only effecting the dimensional tables.

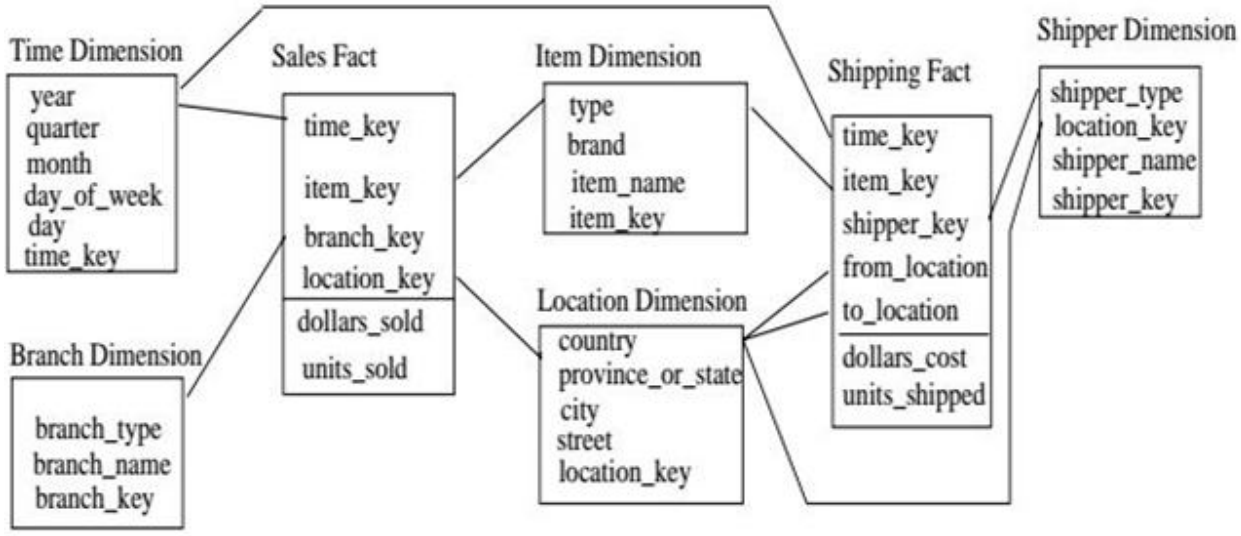


Snowflake schema of a data warehouse for sales.

The major difference between the Snowflake and Star schema models:

- The dimensional tables of the snowflake model may be kept in normalized form to reduce redundancies, which are easy to maintain and save storage a space.
- The Snowflakes structure can reduce the effectiveness of browsing since more joins will be needed to execute a query.
- Hence ,the Snow flake schema is not as popular as Star Schema in data warehouse design

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.



Fact constellation schema of a data warehouse for sales and shipping.

In data warehousing, there is a distinction between a data warehouse and a data mart.

A data warehouse collects information about subjects that span the *entire organization*, such as *customers, items, sales, assets*, and *personnel*, and thus its scope is *enterprise-wide*.

For data warehouses, the fact constellation schema is commonly used, since it can model multiple, interrelated subjects.

A data mart, on the other hand, is a department subset of the data warehouse that focuses on selected subjects, and thus its scope is *department wide*. For data marts, the *star* or *snowflake* schema are commonly used,

Examples for defining star, snow flake, and fact constellation schemas

- Data warehouses and data marts can be defined using two language primitives, one for ***cube definition*** **and one for** ***dimension definition*****.**
- The *cube definition* statement has the following syntax:

define cube <cube name> [<dimension list>]: <measure list>

- The *dimension definition* statement has the following syntax:

define dimension <dimension name> as (<attribute or dimensions list>)

Star schema definition:

The star schema is defined in DMQL as follows:

define **cube sales star** [time, item, branch, location]:

dollars sold = sum(sales in dollars), units sold = count(*)

define dimension time as (time key, day, day of week, month, quarter, year) define dimension item as (item key, item name, brand, type, supplier type) define dimension branch as (branch key, branch name, branch type)

define dimension location as (location key, street, city, province or state, country)

The define cube statement defines a data cube called *sales star*, which corresponds to the central *sales* fact table with two measures, *dollars sold* and *units sold*.

The data cube has four dimensions, namely, *time, item, branch*, and *location*. A define dimension statement is used to define each of the dimensions.

Snowflake schema definition:

The snowflake schema is defined in DMQL as follows:

define cube sales **snowflake** [time, item, branch, location]:

dollars sold = sum(sales in dollars), units sold = count(*)

define dimension time as (time key, day, day of week, month, quarter, year)

define dimension item as (item key, item name, brand, type,

supplier(supplier key, supplier type))

define dimension branch as (branch key, branch name, branch type)

define dimension location as (location key, street,

city (city key, city, province or state, country))

This definition is similar to that of *sales star*), except that, here, the *item* and *location* dimension tables are normalized into two dimension tables, *item* and *supplier.*

Fact constellation schema definition:

The fact constellation schema is defined in DMQL as follows:

define cube sales [time, item, branch, location]:

dollars sold = sum(sales in dollars), units sold = count(*)

define dimension time as (time key, day, day of week, month, quarter, year)

define dimension item as (item key, item name, brand, type, supplier type)

define dimension branch as (branch key, branch name, branch type)

define dimension location as (location key, street, city, province or state, country).

Measures:

Measures can be organized into three categories based on the kind of aggregate functions used:

- Distributive,
- Algebraic,
- Holistic.

Distributive: An aggregate function is distributive if it can be computed in a distributed manner. Suppose the data are partitioned into *n* sets. We apply the function to each partition, resulting in *n* aggregate values.

For example, count() can be computed for a data cube by first partitioning the cube into a set of sub cubes, computing count() for each sub cube, and then summing up the counts obtained for each sub cube. Hence, count()is a distributive aggregate function.

sum(), min(), and max() are distributive aggregate functions.

Algebraic: An aggregate function is algebraic if it can be computed by an algebraic function with *m* arguments (where *m* is a bounded positive integer), each of which is obtained by applying a distributive aggregate function.

For example, avg() (average) can be computed by sum()/count(), where both sum() and count() are distributive aggregate functions.

Similarly, it can be shown that min N() and max N() (which find the N minimum and N maximum values, respectively, in a given set) and standard deviation() are algebraic aggregate functions

Holistic: An aggregate function is holistic if there is no constant bound on the storage size needed to describe

a sub aggregate. That is, there does not exist an algebraic function with *m* arguments (where *m* is a constant) that characterizes the computation.

Common examples of holistic functions include median(), mode(), and rank().

Category	**Examples**
Distributive	Sum(), Count(), Minimum(), Maximum()
Algebraic	Average(), StandardDeviation(), MaxN() (N largest values), MinN() (N smallest values), CenterOfMass()
Holistic	Median(), MostFrequent(), Rank()

3.5 Concept Hierarchies:

A concept hierarchy defines a sequence of mappings from a set of low level concepts to higher level, more general concepts.

Consider a concept hierarchy for the dimension location. City values for location include Vancouver, Montreal, New York, and Chicago. Each city, however, can be mapped to the province or state to which it belongs.

For example, Vancouver can be mapped to British Columbia, and Chicago to Illinois.

The provinces and states can in turn be mapped to the country to which they belong, such as Canada or the USA.

These mappings form a concept hierarchy for the dimension location, mapping a set of low level concepts (i.e., cities) to higher level, more general concepts (i.e., countries). The concept hierarchy described above is illustrated in the following Figure

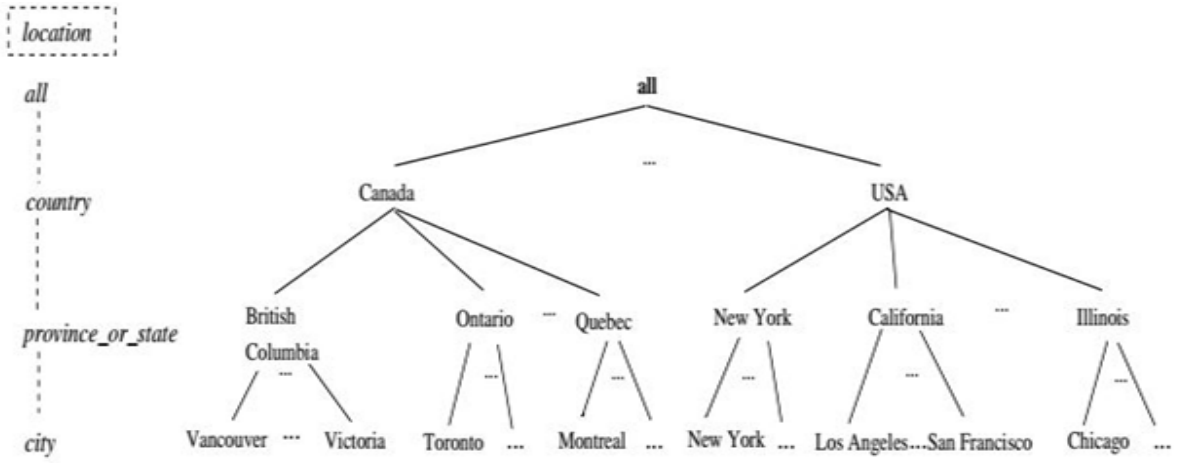
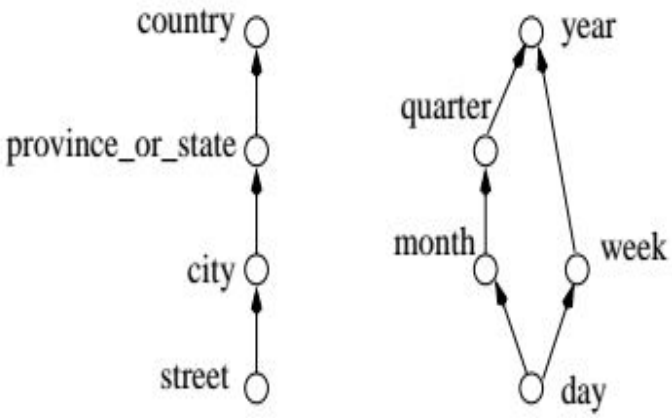


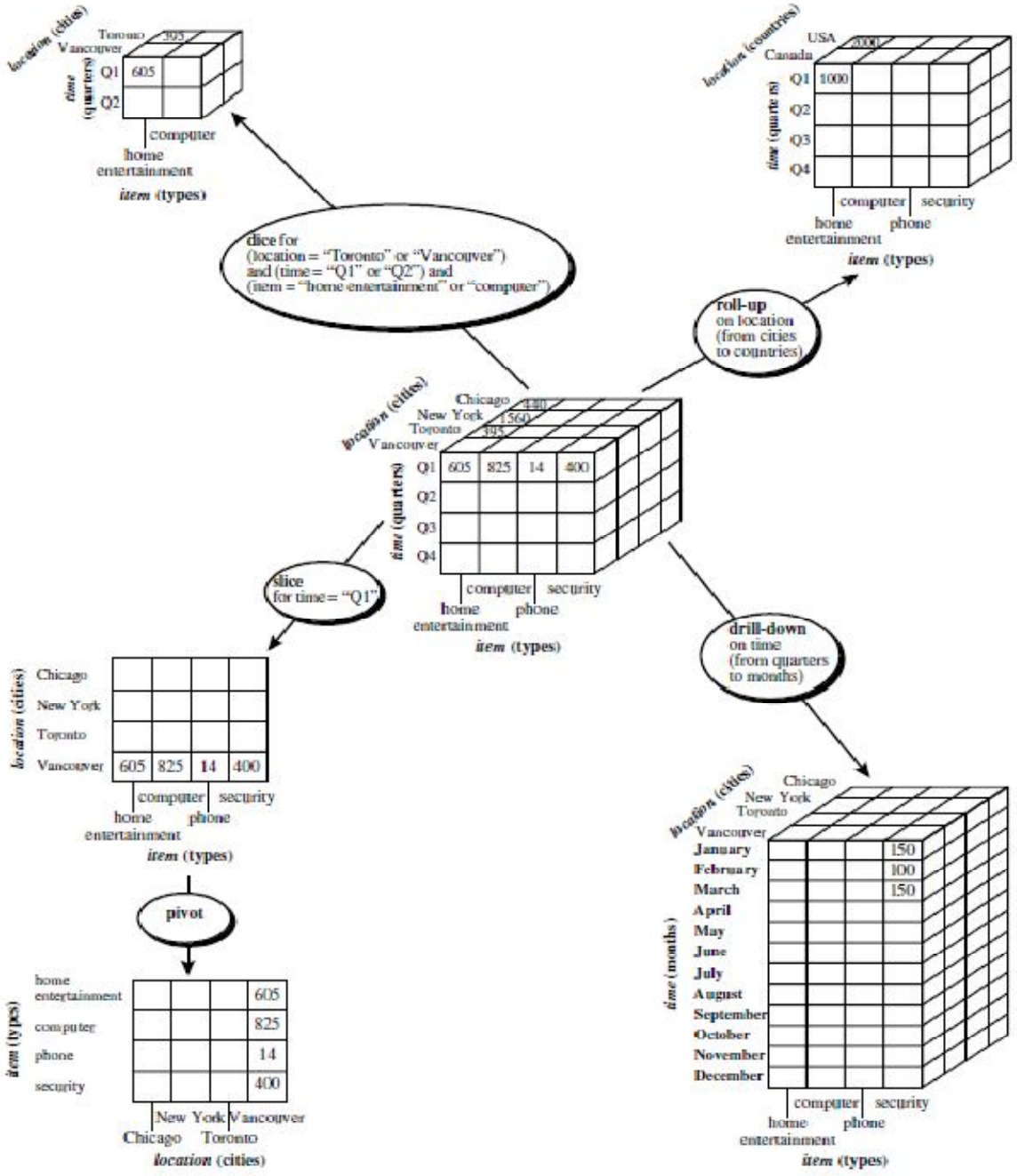
Figure A concept hierarchy for the dimension *location*.



a) a hierarchy for *location* b) a lattice for *time*

Figure 2.8: Hierarchical and lattice structures of attributes in warehouse dimensions.

3.6 OLAP operations in the multidimensional data model:



Examples of typical OLAP operations on multidimensional data.

In the multidimensional model, data are organized into multiple dimensions and each dimension contains multiple levels of abstraction defined by concept hierarchies.

This organization provides users with the flexibility to view data from different perspectives.

A number of OLAP data cube operations exist to materialize these different views, allowing interactive querying and analysis of the data at hand. Hence, OLAP provides a user-friendly environment for interactive data analysis.

1. **Roll-up:** 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.

2. **Drill-down:** 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.

3. **Slice and dice:** The slice operation performs a selection on one dimension of the given cube, resulting in a sub cube.

4. **Pivot (rotate):** Pivot (also called \rotate") is a visualization operation which rotates the data axes in view in order to provide an alternative presentation of the data.

UNIT-III
Assignment-Cum-Tutorial Questions
SECTION-A

Objective Questions

1. _____ is a subject-oriented, integrated, time-variant, nonvolatile collection or data in support of management decisions. [ ]

   A) Data Mining B) Data Warehousing

   C) Document Mining D) Text Mining

2. Historical data is present in the Data warehouse. [T/F]

3. A data warehous is which of the following ? [ ]

   A) Can be updated by end users.

   B) Contains numerous naming conventions and formats.

   C) Organized around important subject areas.

   D) Contains only current data

4. ..... cuboid shows the highest level of summarization. [ ]

   A) apex cuboid. B) base cuboid C) 1-D cuboid D) 3-D cuboid

5. The lattice of cuboids is reffered as _____.

6. The type of relationship in star schema is ..... [ ]

   A) many to many B. one to one C. one to many D. many to one

7. A snow flake schema contains ___ type of tables [ ]

   A) Dimesion. B. Fact. C. Helper. D.All of the above.

8. Fact tables are _____ [ ]

   A) Completely demoralized. C. Partially demoralized.

   B) Completely Normalized. D. Partially normalized.

9. A concept hierarchy defines a sequence of mapping from a set of _____-level concepts to _____-level concepts.

10. Match the following in OLAP operations on multidimensional data

    a) Roll-Up i) Selection on One dimension of the cube

    b) Drill-Down ii) Rotate the data Access

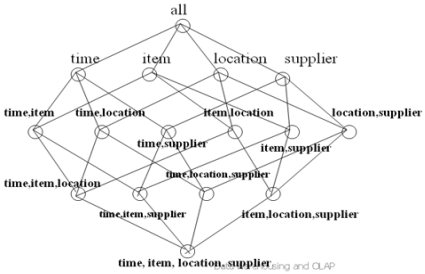
c) Slice and Dice iii) Step down Dimension

d) Pivot iv) Climbing Up dimension [ ]

A) i,iii,ii,iv B) iii,iv,ii,i C) iv,iii,i,ii D) iv,iii,ii,i

11. _____ is a subset of the data warehouse and is usually oriented to a specific business line or team.

12. The following figure shows _____



SECTION-B

SUBJECTIVE QUESTIONS

1. Define Data warehouse and Write about the need of a separate Data Warehouse.
2. Differentiate between the main functionalities of OLTP and OLAP.
3. Develop various multi dimensional data model schemas.
4. Elaborate OLAP operations in multidimensional data model.
5. Describe three tier data warehouse architecture with a neat diagram.
6. Draw a concept hierarchy for dimension location, by considering the location values as Village < mandal <district<state.
7. Suppose that a data warehouse consists of the three dimensions time, doctor, and patient, and the two measures count and charge, where charge is the fee that a doctor charges a patient for a visit.

- Draw star schema and snowflakes schema for the above data warehouse.

8. Draw a lattice of cuboids for the dimension containing five levels (including all), such as “student < major < status < university < all”?
9. Outline the implementation of data warehouse.