UNIT-1

Introduction to Data Warehousing

Syllabus: UNIT –I:

Introduction to Data Warehousing: Introduction to Data Ware House, Differences between operational data base systems and data Ware House, Data Ware House characteristics, Data Ware House Architecture and its components, Extraction-Transformation-Loading, Data warehouse Modelling, Schema Design, star and snow-Flake Schema, Fact Constellation, Fact Table, Fully Addictive, Semi-Addictive, Non-Addictive Measures; Fact-Less-Facts, Dimension Table characteristics; OLAP cube, OLAP Operations, OLAP Server Architecture-ROLAP, MOLAP and HOLAP.

Introduction to Data Ware House

Data: Known, raw facts that can be recorded and have implicit meaning.

Database (DB): Collection of interrelated data.

Data Warehouse (DW): Repository (database) of data collected from multiple data sources

Data warehousing: The process of constructing and using data warehouses

Data Mining (DM): Extracting knowledge from huge amounts of data

Data Warehouse (DWH)

- DWH is a repository for long-term storage of data from multiple sources, stored under a unified schema, usually resides at a single site, organized so as to facilitate management decision making.
- A DWH is a **subject-oriented, integrated, time-variant, and nonvolatile** collection of data in support of management's decision making process.
- A data warehouse can be viewed as an organization repository of data, setup to support strategic decision making.
- In many organizations, the data warehouse can be the foundation for business intelligence, providing a single, integrated source of data for whole organization
- Data warehousing provides architectures and tools for business executives to systematically organize, understand, and use their data to make strategic decisions.
- A DWH refers to a database that is maintained separately from an organization's operational databases.
- DW support information processing by providing a solid platform of consolidated historical data for analysis.

How are organizations using the information from data warehouses?

Many organizations use this information to support business decision-making activities, including:

- increasing customer focus, which includes the analysis of customer buying patterns (such as buying preference, buying time, budget cycles, and appetites for spending)
- repositioning products and managing product portfolios by comparing the performance of sales by quarter, by year, and by geographic regions in order to fine- tune production strategies
- analyzing operations and looking for sources of profit
- Managing the customer relationships, making environmental corrections, and managing the cost of corporate assets.

Differences between Operational DB Systems (OLTP) and DWHs(OLAP)

Users and system orientation:

- An OLTP system is *customer-oriented* and is used for transaction and query processing by clerks, clients, and information technology professional.
- 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.
- An OLAP system manages large amounts of historical data.

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 *snowflake* model and a subject- oriented database design.

Access patterns:

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

View:

- An OLTP system focuses mainly on the current data within an enterprise or department, without referring to historical 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.

Other features that distinguish between OLTP and OLAP systems include database size, frequency of operation and performance metrics which are summarized as follows:

Feature	**OLTP**	**OLAP**
Characteristic	Operational processing	Informational processing
Orientation	Transaction	Analysis
User	Clerk,DBA, database professional	Knowledge worker
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	Historic, 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	GB to high-order GB	>=TB
Priority	High performance, high availability	High flexibility, end-user autonomy
Metric	Transation throughput	Query

Differences Between Database and Data warehouse

Data base	Data warehouse
1.Collection of interrelated data	Collection of data from multiple data sources
2.Manage current data	Manage historic data
3.Used for transaction and Processing	Used for data analysis and decision support
4.Used by clerks, clients and Professionals	Used by managers, executives and analysts
5.Modeled by ER data model	Modelled by data cube
6. It is customer-oriented.	It is market-oriented.
7. DB size is 100 MB to GB	DB size is 100 GB to TB

Similarities between DB and DW

- Both DB and DW is used for storing data i.e., both are repositories of information, storing huge amounts of persistent data.
- Both support multiuser access.
- Both require queries for accessing the data.
- Both servers can be present on the company premise or on the cloud.

Why Have a Separate Data Warehouse?

- Separation promotes the *high performance of both systems*. Processing OLAP queries in operational databases would substantially degrade the performance of operational tasks.
- In OLTP, Concurrency control and recovery mechanisms, such as locking and logging, are required to ensure the consistency and robustness of transactions. An OLAP query often needs read-only access of data records for summarization and aggregation. Concurrency control and recovery mechanisms, if applied for such OLAP operations, reduce the throughput of an OLTP system.
- Decision support requires historical data, whereas operational databases do not typically maintain historical data. In this context, the data in operational databases, though abundant, is usually far from complete for decision making. Decision support requires consolidation (such as aggregation and summarization) of data from heterogeneous sources, resulting in high- quality, clean, and integrated data.

Data Ware House Characteristics

Data Ware house has four important features or characteristics:

1. Subject-oriented
2. Integrated
3. Time-variant
4. Non-volatile

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 on-line transaction records. Data cleaning and data integration techniques are applied to ensure consistency in naming conventions, attribute measures and so on.

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*.

Data Ware House Architecture and its Components

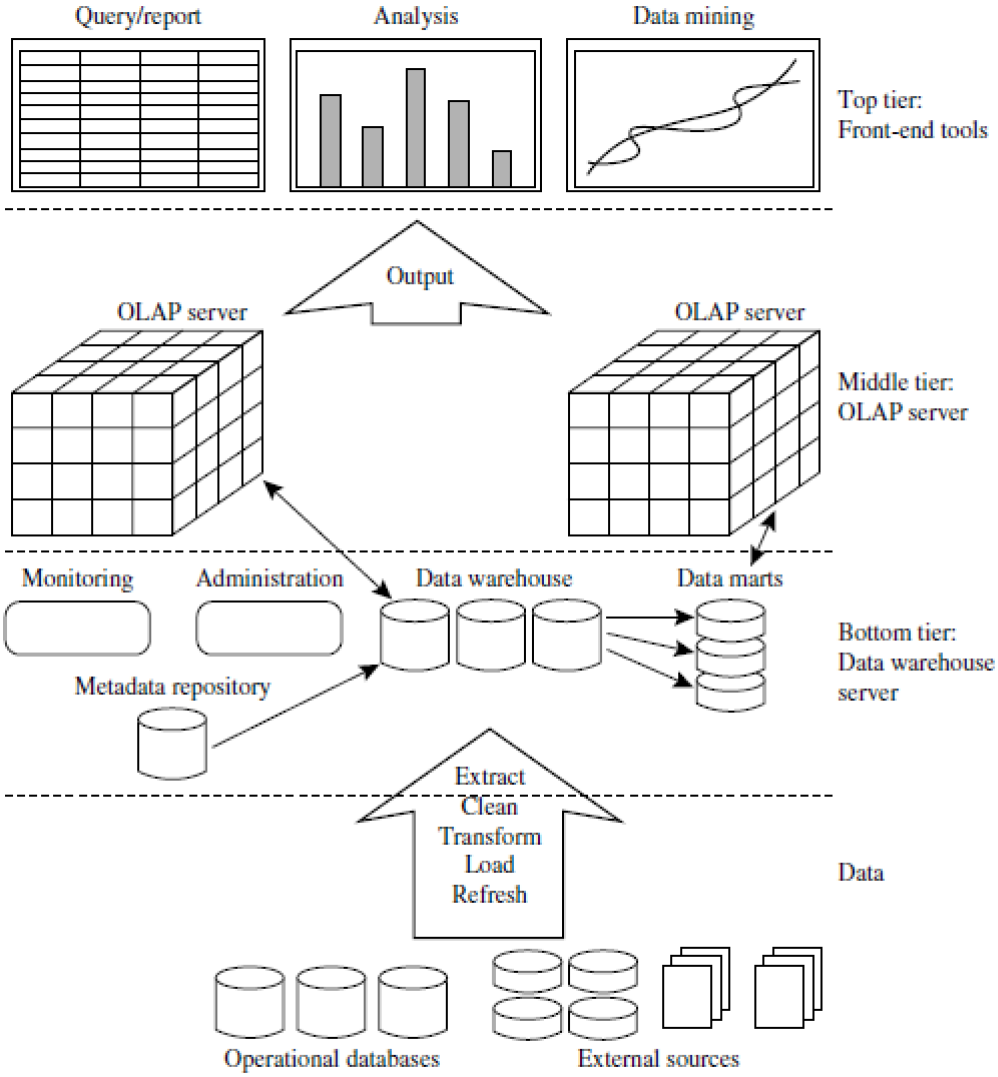
Data warehouses often adopt **three-tier architecture**. The bottom tier is a *warehouse database server*, which is typically a relational database system. The middle tier is an *OLAP server*, and the top tier is a *client* that contains query and reporting tools.

1. **The bottom tier : warehouse database server**
   - 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.
   - These tools and utilities perform data extraction, cleaning, and 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.
2. **The middle: OLAP server**

   The middle tier is an **OLAP server** that is typically implemented using either (1) a **relational OLAP (ROLAP)** model (i.e., an extended relational DBMS that maps operations on multidimensional data to standard relational operations); or (2) a **multidimensional OLAP (MOLAP)** model (i.e.,a special-purpose server that directly implements multidimensional data and operations.

3. **The top tier: front-end client layer**

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).



A three-tier data warehousing architecture.

Data Warehouse Models

From the architecture point of view, there are three data warehouse models:

1. Enterprise Warehouse,
2. Data Mart
3. Virtual Warehouse

Enterprise warehouse:

- An enterprise warehouse collects all of the information about subjects spanning the entire organization.
- It provides corporate-wide data integration, usually from one or more operational systems or external information providers, and is cross-functional in scope.

- It typically contains detailed data as well as summarized data, and can range in size from a few gigabytes to hundreds of gigabytes, terabytes, or beyond.
- An enterprise data warehouse may be implemented on traditional mainframes, computer super servers, or parallel architecture platforms.
- It requires extensive business modelling and may take years to design and build.

Data mart:

- A data mart contains a subset of corporate-wide data that is of value to a specific group of users. The scope is confined to specific selected subjects.
- Data marts are usually implemented on low-cost departmental servers that are Unix/Linux or Windows based.
- The implementation cycle of a data mart is more likely to be measured in weeks rather than months or years.
- Depending on the source of data, data marts can be categorized as independent or dependent. **Independent** data marts are sourced from data captured from one or more operational systems or external information providers, or from data generated locally within a particular department or geographic area. **Dependent** data marts are sourced directly from enterprise data warehouses.

Virtual warehouse:

- A virtual warehouse is a set of views over operational databases.
- For efficient query processing, only some of the possible summary views may be materialized.
- A virtual warehouse is easy to build but requires excess capacity on operational database servers.

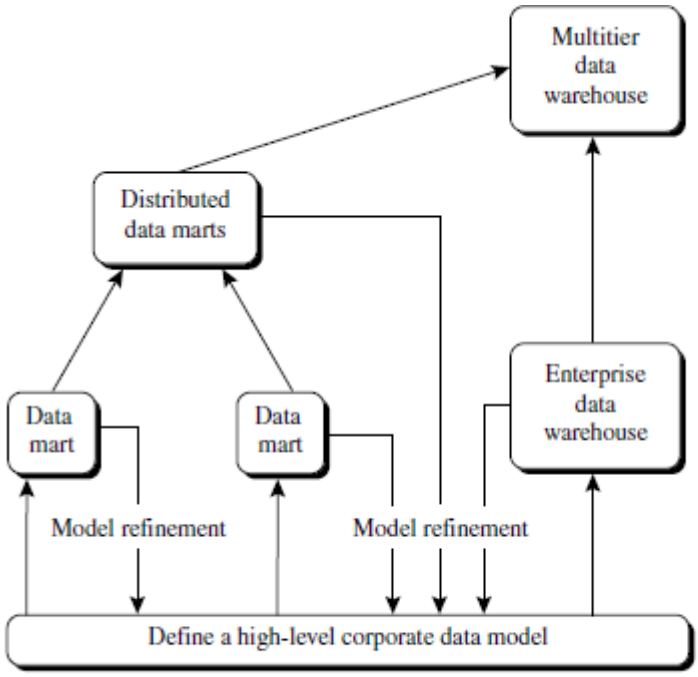
Distinction between a data warehouse and a data mart

(What is the difference between a data warehouse and a data mart?)

Data Warehouse	Data Mart
1.It collects information about subjects that span the entire organization	It is a department subset of the data warehouse that is of value to a specific group of users.
2.Its scope is enterprise-wide(corporate-wide)	Its scope is department-wide i.e., It focuses on selected subjects
3.For data warehouses, the fact constellation schema is commonly used	For data mart, the star or snowflake schema are commonly used
4. Data warehouse is vast (more than 100 GB) in size.	Data mart is smaller than(less than 100 GB) data warehouse
5.usually implemented on mainframes, computer super servers or parallel architecture platforms	Usually implemented on low-cost departmental servers that are UNIX/LINIX or windows based.

6.may take years to design and build	take weeks rather than months or years
7.typically contains detailed as well as summarized data	typically contains summarized data.
8.provides corporate-wide data integration	It is confined to specific selected subjects.

Approach for data warehouse development:



A recommended approach for data warehouse development.

- A recommended method for the development of data warehouse systems is to implement the warehouse in an incremental and evolutionary manner, as shown in above Figure.
- First, a high-level corporate data model is defined within a reasonably short period (such as one or two months) that provides a corporate-wide, consistent, integrated view of data among different subjects and potential usages. This high-level model, although it will need to be refined in the further development of enterprise data warehouses and departmental data marts, will greatly reduce future integration problems.
- Second, independent data marts can be implemented in parallel with the enterprise warehouse based on the same corporate data model set noted before.
- Third, distributed data marts can be constructed to integrate different data marts via hub servers.
- Finally, a **multitier data warehouse** is constructed where the enterprise warehouse is the sole custodian of all warehouse data, which is then distributed to the various dependent data marts.

Extraction, Transformation, and Loading

Data warehouse systems use back-end tools and utilities to populate and refresh their data. These tools and utilities include the following functions:

- **Data extraction**, which typically gathers data from multiple, heterogeneous, and external sources.
- **Data cleaning**, which detects errors in the data and rectifies them when possible
- **Data transformation**, which converts data from legacy or host format to warehouse format.
- **Load**, which sorts, summarizes, consolidates, computes views, checks integrity, and builds indices and partitions.
- **Refresh**, which propagates the updates from the data sources to the warehouse.

Metadata Repository

- **Metadata** are data about data.
- When used in a data warehouse, metadata are the data that define warehouse objects.

Metadata should be stored and managed persistently (i.e., on disk).

A metadata repository should contain the following:

1. 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.
2. *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).
3. 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.
4. *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).
5. *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.
6. *Business metadata*, which include business terms and definitions, data ownership information, and charging policies.

Role of Meta data in data warehouse:

(Importance of Meta data in DWH)

- A data warehouse contains different levels of summarization, of which metadata is one. Other types include current detailed data (which are almost always on disk), older detailed data (which are usually on tertiary storage), lightly summarized data, and highly summarized data (which may or may not be physically housed).

- Metadata play a very different role than other data warehouse data and are important for many reasons:
    - Meta data are used as a directory to help the decision support system analyst locate the contents of the data warehouse.
    - Meta data are used as a guide to the data mapping when data are transformed from the operational environment to the data warehouse environment.
    - Metadata also serve as a guide to the algorithms used for summarization between the current detailed data and the lightly summarized data, and between the lightly summarized data and the highly summarized data.

Multidimensional Data Model.

- Data warehouses and OLAP tools are based on a **multidimensional data model**.
- This model views data in the form of a *data cube*.
- The data cube is a metaphor for multi-dimensional data storage.
- A **data cube** allows data to be modeled and viewed in multiple dimensions.
- A data cube provides a multidimensional view of data and allows the pre computation and fast accessing of summarized data.
- It is defined by dimensions and facts.

Dimensions:

- **Dimensions** are the perspectives or entities with respect to which an organization wants to keep records.
    - Example: a *sales* data warehouse in order to keep records of the store's sales with respect to the dimensions *time, item, branch, and location.* These dimensions allow the store to keep track of things like monthly sales of items and the branches and locations.

Dimension table:

A table that further describes the dimension.

Example: a dimension table for *item* may contain the attributes *item-name, brand*, and *type*. Dimension tables can be specified by users or experts, or automatically generated and adjusted based on data distributions.

Facts:

Facts are numerical measures. They are Quantities by which we want to analyze relationships between dimensions.

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

Fact Table:

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

Data Cube Example:

In data warehousing, the data cube is n-dimensional.

Example: Consider the data of a shop for items sold per quarter in the city of Delhi. The data is shown in the table. In this 2D representation, the sales for Delhi are shown for the time dimension (organized in quarters) and the item dimension (classified according to the types of an item sold). The fact or measure displayed in rupees-sold (in thousands).

Location="Delhi"				
Time (quarter)	item (type)			
	Egg	Milk	Bread	Biscuit
Q1	260	508	15	60
Q2	390	256	20	90
Q3	436	396	50	40
Q4	528	483	35	50

Table: A 2-D view of sales data according to the dimensions time and item

Now, if we want to view the sales data with a third dimension, For example, suppose the data according to time and item, as well as the location is considered for the cities Chennai, Kolkata, Mumbai, and Delhi. These 3D data are shown in the table. The 3D data of the table are represented as a series of 2D tables.

	Location="Chennai"				Location="Kolkata"				Location="Mumbai"				Location="Delhi"			
	item				item				item				item			
Time	Egg	Milk	Bread	Biscuit	Egg	Milk	Bread	Biscuit	Egg	Milk	Bread	Biscuit	Egg	Milk	Bread	Biscuit
Q1	340	360	20	10	435	460	20	15	390	385	20	39	260	508	15	60
Q2	490	490	16	50	389	385	45	35	463	366	25	48	390	256	20	90
Q3	680	583	46	43	684	490	39	48	568	594	36	39	436	396	50	40
Q4	535	694	39	38	335	365	83	35	338	484	48	80	528	483	35	50

Table: A 3-D view of sales data according to the dimensions time, item and location

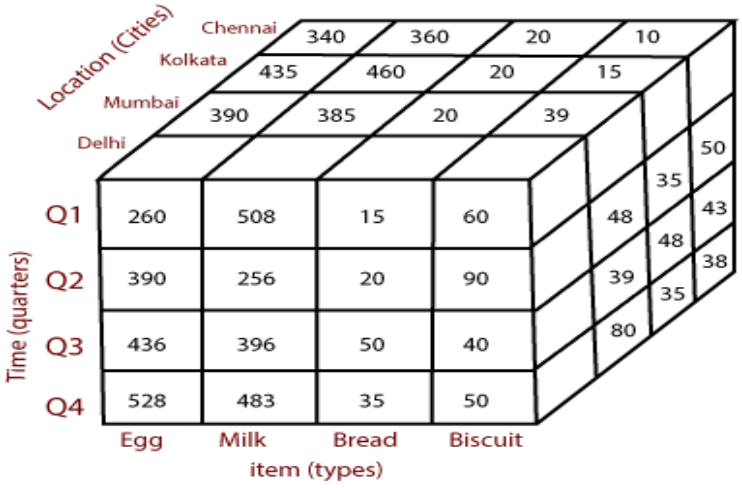


Fig: A 3-D data cube representation of the data in above table according to the dimensions, time, item and location

- In the data warehousing, a data cube is often referred to as a cuboid.
- 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.
- Example 1: Below figure shows a lattice of cuboids forming a data cube for the dimensions *time (year)*, *item*, and *location (city)*.

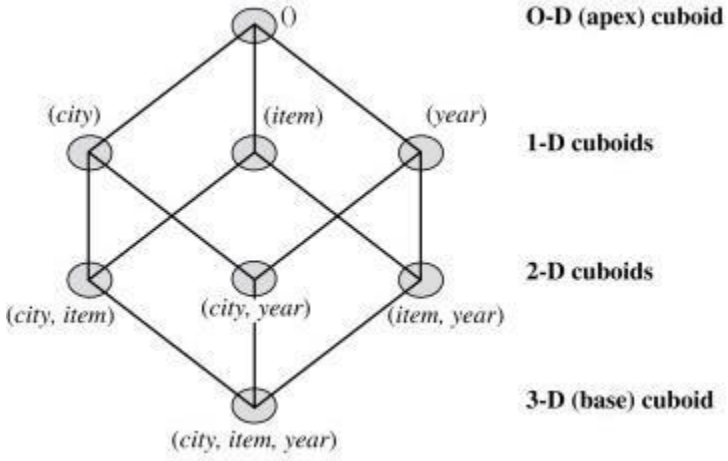


Figure 3.2 Lattice of cuboids, making up a 3-D data cube for the dimensions *time*, *item* and *location*. Each cuboid represents a different degree of summarization.

Example 2: Below figure shows a lattice of cuboids forming a data cube for the dimensions *time*, *item*, *location*, and *supplier*.

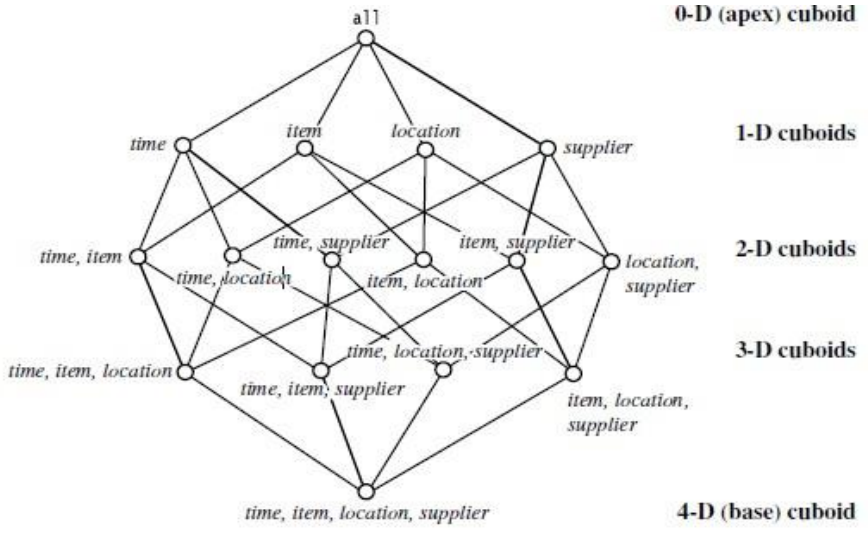


Figure 3.3 Lattice of cuboids, making up a 4-D data cube for the dimensions *time*, *item*, *location*, and *supplier*. Each cuboid represents a different degree of summarization.

The cuboid that holds the lowest level of summarization is called the **base cuboid**. Example: 4-D cuboid in fig. 3.3 and 3-D cuboid in fig.3.2

The cuboid, which holds the highest level of summarization, is called the **apex cuboid**. Example: 0-D cuboid

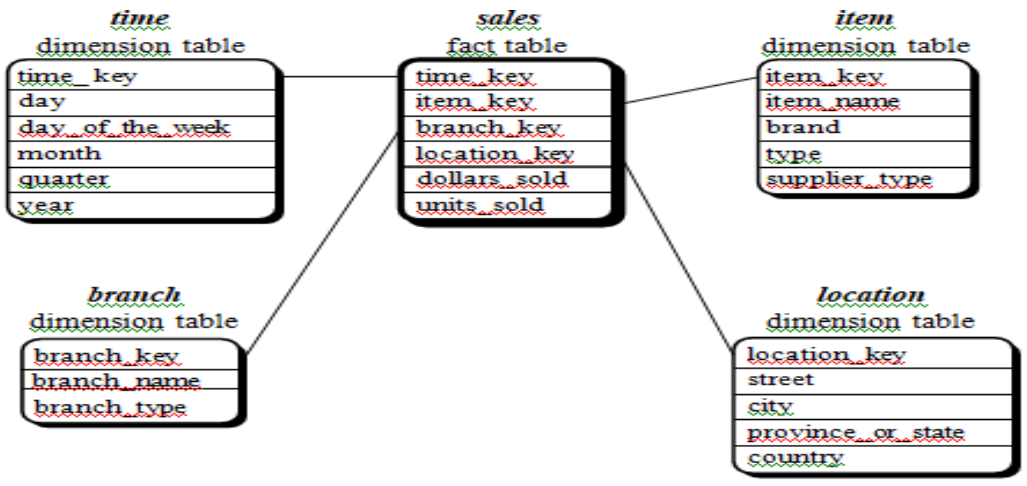
Schemas for Multidimensional Data Model

- The most popular data model for a data warehouse is a **multidimensional model**. i.e., a multi-dimensional data model used for the design of corporate data warehouse and departmental data marts.
- This model views the data in the form of data cube. i.e., the core of multi-dimensional model is the data cube.
- This model can exist in the form of a **star schema**, a **snowflake schema**, or a **fact constellation schema**. i.e., this model adopts a **star schema**, a **snowflake schema**, or a **fact constellation schema.**

Star Schema:

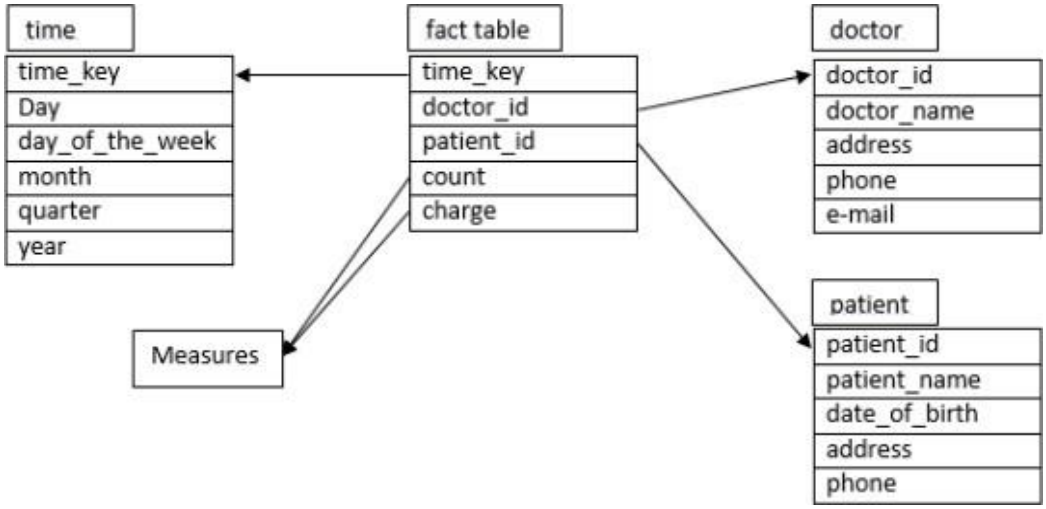
- The star schema is more popular and efficient.
- The most common modeling paradigm is the star schema, in which the data warehouse contains:
- a large central table (**fact table**) containing the bulk of the data, with no redundancy, and
- a set of smaller attendant tables (**dimension tables**), one for each dimension.
- In the star schema, each dimension is represented by only one table, and each table contains a set of attributes.
- The schema graph resembles a starburst, with the dimension tables displayed in a radial pattern around the central fact table.

Example 1: Suppose that a sales data warehouse consists of the four dimensions time, *item branch* ,and location , and the two measures, dollars- sold and units-sold, where dollars-sold is the sales amount in dollars and units-sold is number of units sold. Draw a ***star schema*** diagram for this data warehouse.



Star schema of a data warehouse for sales.

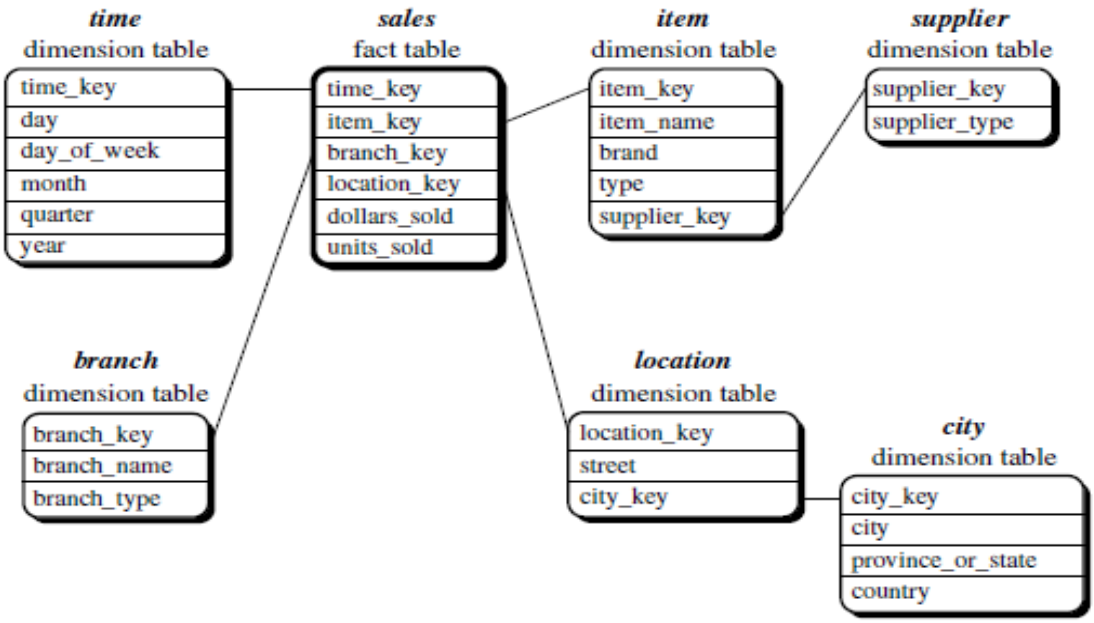
Example 2: Suppose that a data warehouse consists of the three dimension *time, doctor,* and *patient, and* the two measures *count* and charge, where *charge* is the fee that a doctor charges a patient for a visit. Draw a **Star schema** diagram for the above data warehouse



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.
- 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.

Example:

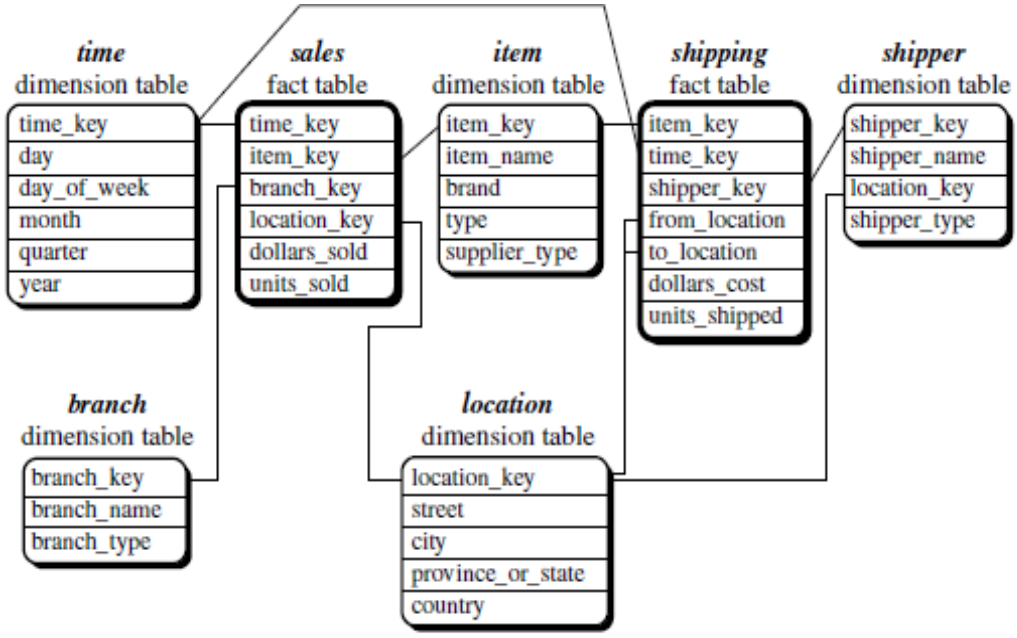


Snowflake schema of a data warehouse for sales.

Fact Constellation Schema or (Galaxy schema):

- Fact constellation schema consists of multiple fact tables to *share* dimension tables.
- A fact constellation schema allows dimension tables to be shared between fact tables
- This schema can be viewed as a collection of stars, and hence is called a galaxy schema or a fact constellation.

Example:



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

Similarities and Differences of the star schema and snowflake schema:

- They are similar in the sense that both have a fact table, as well as some dimensional tables.
- The major difference is that some dimension tables in the snowflake schema are normalized, thereby further splitting the data into additional tables.
- The advantage of the star schema is its simplicity, which will enable efficiency, but it requires more space.
- For the snowflake schema, it reduces some redundancy by sharing common tables: the tables are easy to maintain and save some space. However, it is less efficient and the saving of space is negligible in comparison with the typical magnitude of the fact table.
- Therefore, empirically, the star schema is better simply because efficiency typically has higher priority over space as long as the space requirement is not too huge.
-
-

- **Comparison between the snowflake schema and fact constellation:**
  - The snowflake schema and fact constellation are both variants of the star schema model, which consists of a fact table and a set of dimension tables
  - The snowflake schema contains some normalized dimension tables, whereas the fact constellation contains a set of fact tables that share some common dimension tables.
  - The resulting schema graph of snowflake schema forms a shape similar to a snowflake where as fact constellation schema can be viewed as a collection of stars, and hence is also called a galaxy schema.
  - The snowflake schema contains exactly one fact table where as fact constellation schema consists of multiple fact tables.

Measures and their Categorization

- A data cube **measure** is a numeric function that can be evaluated at each point in the data cube space.
- Measures can be organized into three categories distributive, algebraic, and holistic— based on the kind of aggregate functions used.

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. If the result derived by applying the function to the *n* aggregate values is the same as that derived by applying the function to the entire data set (without partitioning), the function can be computed in a distributed manner.
- For example, sum() can be computed for a data cube by first partitioning the cube into a set of sub-cubes, computing sum() for each sub-cube, and then summing up the counts obtained for each sub-cube. Hence, sum() is a distributive aggregate function. For the same reason, count(), min(), and max() are distributive aggregate functions.
- A measure is *distributive* if it is obtained by applying a distributive aggregate function. Distributive measures can be computed efficiently because of the way the computation can be partitioned.

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.
- A measure is *algebraic* if it is obtained by applying an algebraic aggregate function.

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().
- A measure is *holistic* if it is obtained by applying a holistic aggregate function.

Fully-Additive, Semi-Additive and Non-Additive measures

We can divide the Facts or measures in to these three types based on aggregate across dimensions:

- Fully-Additive
- Semi-Additive
- Non-Additive

Fully-Additive Facts:

- Additive Facts are Facts that can be summed up through all of the dimensions in the Fact table.
- They can be added across all the dimensions in the fact table.
- They are most flexible and useful facts
- For example, if there is a retail store and if we want to identify the total sales which have happened in the last six months, we can group the records of the last six months and get the summed up (aggregated) value.
- In this case, Sales or the Sales amount is an additive fact, because we can sum up (group) the fact records with other related dimensions (product, customer, supplier etc) present in the fact table.
- Example- Assume for a retail company, we have a fact table with the below attributes:

Date
Product
Store
Product sold

- This table is used to record the numbers of product sold for all the stores.
- The fact/column product sold can be summed up for each level of its dimensions (date, product, store)
- The product sold is called additive fact because it can be summed up though the entire dimension in the fact table.

Semi-Additive Facts:

- Semi-additive Facts are Facts that can be summed up for some of the dimensions in the Fact table, but not the others.

- One of the usual examples for this is are the current account or savings account balance amounts are common semi-additive facts. We can get/generate a balance amount from the overall transactions, but it doesn't make any sense to add (group) the balance amounts from different dates or months or across the time dimension.
- Example- Consider below balance fact table:

Date
Account
Balance

- for a particular day it makes sense to add all the account balance, but it does not make sense to add the balance through time dimension. Here balance is a semi additive fact.

Non-Additive Facts:

- Non-additive facts are facts that cannot be added for any of the dimensions present in the fact table.
- Eg: Facts which have percentages, Ratios calculated.
- Consider below fact table of a store: Assume that profit for the day 1 is 70% and for day 2 is 30%, so we can't say that total profit is 100%. In this case we cannot summed up for any dimensions in the fact table. This is the example of non-additive fact.

Date
Product
Store
Product sold

Fact-less-Facts:

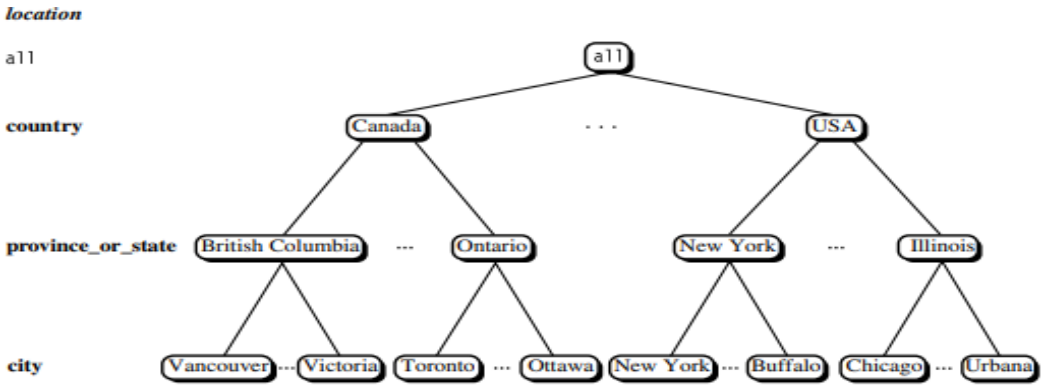
- In the real world, it is possible to have a fact table that contains no measures or facts. These tables are called "Fact less Fact tables".
- A fact table that contains no facts (or measures) is called fact-less-fact table.
- Example: A fact table which has only product key and date key is a fact less fact. There are no measures in this table. But still you can get the number products sold over a period of time.

Product-key
Date-key

The Role of Concept Hierarchies in data warehouse

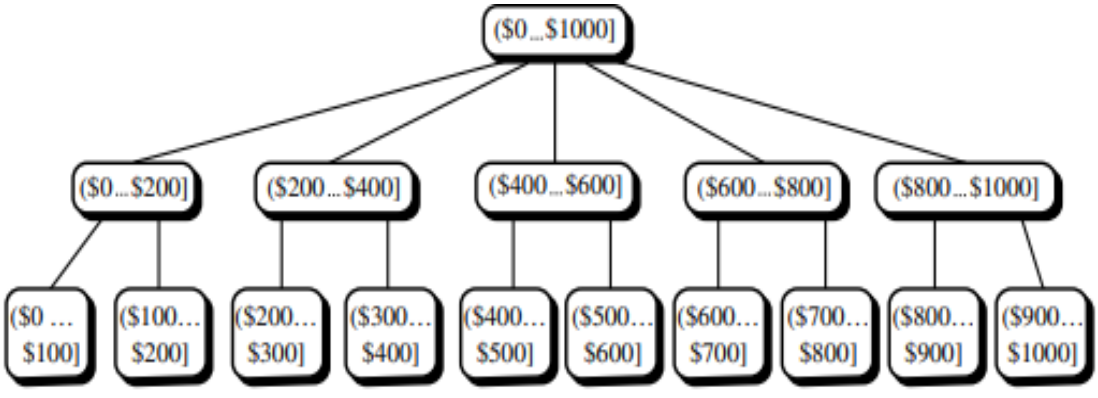
- A concept hierarchy organizes the concepts (attribute or dimensions) into varying levels of abstraction. i.e., Concept hierarchies are useful for mining at multiple levels of abstraction.
- A concept hierarchy defines a sequence of mappings from a set of low-level concepts to higher-level, more general concepts.

- The concept hierarchies can be used to transform the data into multiple levels of granularity. i.e., Concept hierarchies allow data to be handled at varying levels of abstraction.
- Concept hierarchies can be used to reduce the data by collecting and replacing low-level concepts with higher-level concepts.
- Concept hierarchies that are common to many applications may be predefined in data mining system such as Concept hierarchy for *time.*
- Concept hierarchies may be provided manually by system users, domain experts, or knowledge engineers, or may be automatically generated based on statistical analysis of the data distribution



A concept hierarchy for the dimension *location*. Due to space limitations, not all of the nodes of the hierarchy are shown (as indicated by the use of "ellipsis" between nodes).

Concept hierarchies may also be defined by discretizing or grouping values for given dimension or attribute, resulting in a **set-grouping hierarchy.**



A concept hierarchy for the attribute *price*.

Many concept hierarchies are implicit within the database schema. A user or expert can easily define a concept hierarchy by specifying a partial or total ordering of the attributes at the schema level.

For example, suppose that the dimension *location* is described by the attributes *number, street, city, province or state, zip*, and *country*. These attributes are related by a total order, forming a concept hierarchy such as **"*street* < *city* < *province or state* < *country*".** Alternatively, the attributes of a dimension may be organized in a partial order, forming a lattice. An example of a partial order for the *time* dimension based on the attributes *day, week, month, quarter*, and *year* is **"*day* < { *month* <*quarter; week* } < *year*".**

A concept hierarchy that is a total or partial order among attributes in a database schema is called a **schema hierarchy.**

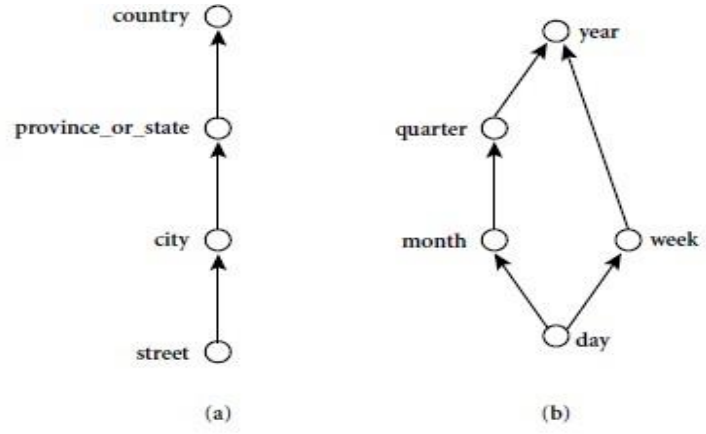


Figure 3.8 Hierarchical and lattice structures of attributes in warehouse dimensions: (a) a hierarchy for *location*; (b) a lattice for *time*.

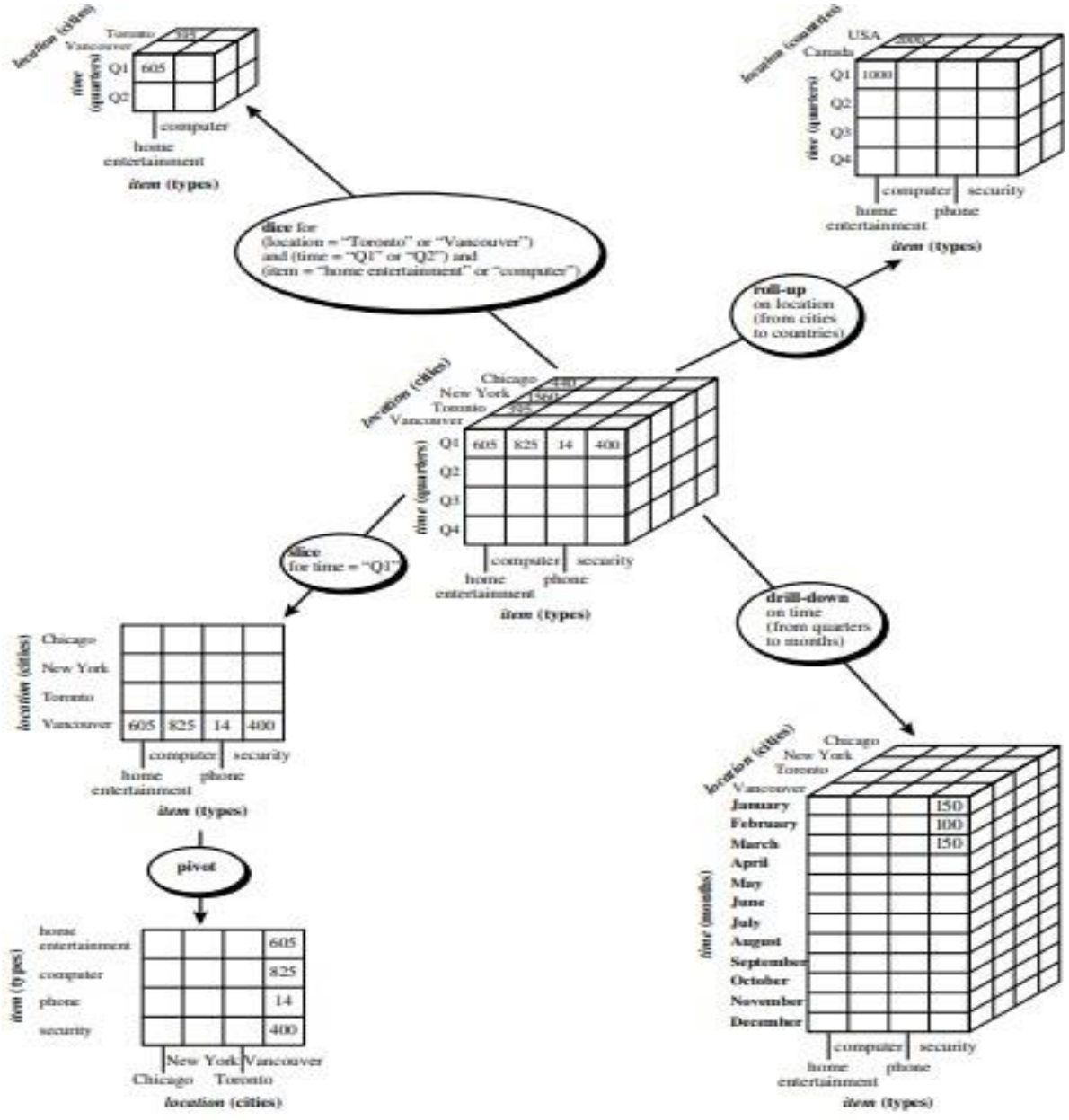
There may be more than one concept hierarchy for a given attribute or dimension, based on different user viewpoints.

OLAP Operations on Multidimensional Data Model

- In the multidimensional model, the data are organized into various dimensions, and each dimension includes multiple levels of abstraction described by concept hierarchies.
- This organization support users with the flexibility to view data from various perspectives
- A number of OLAP data cube operations exist to materialize different views, allowing interactive querying and analysis of the data at hand.
- OLAP provides a user-friendly environment for interactive data analysis.
- OLAP operations use background knowledge (such as concept hierarchy) regarding the domain of the data being studied in order to allow the presentation of data at *different levels of abstraction*.

Typical OLAP operations are:

1. Roll-up or Drill-up
2. Drill-down
3. Slice
4. Dice
5. Pivot or Rotate



Examples of typical OLAP operations on multidimensional data.

Roll up:

- It performs aggregation on the OLAP cube.
- It can be done by:
  - Climbing up in the concept hierarchy for a dimension or
  - Reducing the dimensions

- When roll-up operation performed by dimension reduction, one or more dimensions are removed from the given cube.For example, consider a sales data cube containing only the two dimensions *location* and *time*. 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.

Drill-down:

- It is the reverse of roll-up.
- The less detailed data is converted into highly detailed data.
- It can be done by:
- Moving down in the concept hierarchy for a dimension or
- adding a new dimension
- Since 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 can occur by introducing an additional dimension, such as *customer group*.

Slice:

- It selects a single dimension from the OLAP cube which results in a new sub-cube creation. i.e., slice is an operation that selects one specific dimension from a given data cubes and provides a new sub-cube.

Dice:

- It defines a sub cube from the OLAP cube by selecting two or more dimensions. i.e**.,** dice is an operation that selects two or more dimensions from a given data cube and provides a new subcube.

Pivot:

- It is also known as *rotation* operation.
- It rotates the current view to get a new view of the representation.
- It is a visualization operation that rotates the data axes in view in order to provide an alternative presentation of the data.

Other OLAP operations: Some OLAP systems offer additional drilling operations .For example :

drill across executes queries involving(ie aross) more than a fact table

drill–through operation uses relational SQL facilities to drill through the bottom level of data cube down to its back –end relational tables.

Other OLAP operations may include ranking the top N or bottom N items in lists as well as computing moving averages, growth rates, interests, internal return rates, depreciation, currency conversions and statistical functions.

OLAP Server Architectures: ROLAP versus MOLAP versus HOLAP

(Types of OLAP Servers)

- Logically, 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:

1. ROLAP(Relational OLAP
2. MOLAP(Multidimensional OLAP)
3. HOLAP(Hybrid OLAP)

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.

Advantages:

- ROLAP can handle the large amounts of data.
- ROLAP tools don't use pre-calculated data cubes.
- Data can be stored efficiently.
- ROLAP can leverage functionalities inherent in the relational database.

Disadvantages:

- Performance of ROLAP can be slow.
- In ROALP, difficult to maintain aggregate tables.
- Limited by SQL functionalities.

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.
- The storage utilization may be low with multidimensional data stores
- Many MOLAP server handle dense and sparse data sets by using two levels of data storage representation..

Advantages:

- MOLAP can perform complex calculations.
  - Excellent Performance. MOLAP is optimal for operation such as slice and dice.
  - MOLAP allows fastest indexing to the pre-computed summarized data.

Disadvantages:

- MOLAP can't handle large amount of data. i.e., limited in the amount of data it can handle.
- In MOLAP, Requires additional investment.
- Without re-aggregation, difficult to change dimension.

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.
- For summary-type information, HOLAP leverages cube technology for faster performance. When detail information is needed, HOLAP can "drill through" from the cube into the underlying relational data.
- The Microsoft SQL Server 2000 supports a hybrid OLAP server.

Advantages:

- HOLAP provides the functionalities of both MOLAP and ROLAP.
- HOLAP provides fast access at all levels of aggregation.

Disadvantages:

- HOLAP architecture is very complex to understand because it supports both MOLAP and ROLAP.

Difference between ROLAP, MOLAP and HOLAP

Basis	ROLAP	MOLAP	HOLAP
Full form	ROLAP stands for Relational Online Analytical Processing.	MOLAP stands for Multidimensional Online Analytical Processing.	HOLAP stands for Hybrid Online Analytical Processing.
Storage location for summary aggregation	Relational Database is used as storage location for summary aggregation.	Multidimensional Database is used as storage location for summary aggregation.	Multidimensional Database is used as storage location for summary aggregation.
Processing time	Processing time of ROLAP is very slow.	Processing time of MOLAP is fast.	Processing time of HOLAP is fast.
Storage space requirement	Large storage space requirement in ROLAP as compare to MOLAP and HOLAP.	Medium storage space requirement in MOLAP as compare to ROLAP and HOLAP.	Small storage space requirement in HOLAP as compare to MOLAP and ROLAP.

Basis	ROLAP	MOLAP	HOLAP
Storage location for detail data	Relational database is used as storage location for detail data.	Multidimensional database is used as storage location for detail data.	Relational database is used as storage location for detail data.
Latency	Low latency in ROLAP as compare to MOLAP and HOLAP.	High latency in MOLAP as compare to ROLAP and HOLAP.	Medium latency in HOLAP as compare to MOLAP and ROLAP.
Query response time	Slow query response time in ROLAP as compare to MOLAP and HOLAP.	Fast query response time in MOLAP as compare to ROLAP and HOLAP.	Medium query response time in HOLAP as compare to MOLAP and ROLAP