

SCS5623 Data Mining and Data Warehousing

UNIT IV

DATA WAREHOUSING

Introduction

A data warehouse is constructed by integrating data from multiple heterogeneous sources. It supports analytical reporting, structured and/or ad hoc queries and decision making. This tutorial adopts a step-by-step approach to explain all the necessary concepts of data warehousing. The term "Data Warehouse" was first coined by Bill Inmon in 1990.

According to Inmon, a data warehouse is a subject oriented, integrated, time-variant, and non-volatile collection of data. This data helps analysts to take informed decisions in an organization. An operational database undergoes frequent changes on a daily basis on account of the transactions that take place. Suppose a business executive wants to analyze previous feedback on any data such as a product, a supplier, or any consumer data, then the executive will have no data available to analyze because the previous data has been updated due to transactions. A data warehouse provides us generalized and consolidated data in multidimensional view. Along with generalized and consolidated view of data, a data warehouse also provides us Online Analytical Processing (OLAP) tools. These tools help us in interactive and effective analysis of data in a multidimensional space. This analysis results in data generalization and data mining. Data mining functions such as association, clustering, classification, prediction can be integrated with OLAP operations to enhance the interactive mining of knowledge at multiple level of abstraction. That's why data warehouse has now become an important platform for data analysis and online analytical processing.

Understanding a Data Warehouse

- A data warehouse is a database, which is kept separate from the organization's operational database.
- There is no frequent updating done in a data warehouse.
- It possesses consolidated historical data, which helps the organization to analyze its business.
- A data warehouse helps executives to organize, understand, and use their data to take strategic decisions.
- Data warehouse systems help in the integration of diversity of application systems.
- A data warehouse system helps in consolidated historical data analysis.

Why a Data Warehouse is Separated from Operational Databases

A data warehouse 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.
- An operational database maintains current data. On the other hand, a data warehouse maintains historical data.

Data Warehouse Features

The key features of a data warehouse are discussed below:

- **Subject Oriented** - A data warehouse is subject oriented because it provides information around a subject rather than the organization's ongoing operations. These subjects can be product, customers, suppliers, sales, revenue, etc. A data warehouse does not focus on the ongoing operations, rather it focuses on modelling and analysis of data for decision making.
- **Integrated** - A data warehouse is constructed by integrating data from heterogeneous sources such as relational databases, flat files, etc. This integration enhances the effective analysis of data.
- **Time Variant** - The data collected in a data warehouse is identified with a particular time period. The data in a data warehouse provides information from the historical point of view.
- **Non-volatile** - Non-volatile means the previous data is not erased when new data is added to it. A data warehouse is kept separate from the operational database and therefore frequent changes in operational database is not reflected in the data warehouse.

Note: A data warehouse does not require transaction processing, recovery, and concurrency controls, because it is physically stored and separate from the operational database.

Data Warehouse Applications

As discussed before, a data warehouse helps business executives to organize, analyze, and use their data for decision making. A data warehouse serves as a sole part of a plan-execute-assess "closed-loop" feedback system for the enterprise management. Data warehouses are widely used in the following fields:

- Financial services
- Banking services
- Consumer goods
- Retail sectors
- Controlled manufacturing

Types of Data Warehouse

Information processing, analytical processing, and data mining are the three types of data warehouse applications that are discussed below:

- **Information Processing** - A data warehouse allows to process the data stored in it. The data can be processed by means of querying, basic statistical analysis, reporting using crosstabs, tables, charts, or graphs.
- **Analytical Processing** - A data warehouse supports analytical processing of the information stored in it. The data can be analyzed by means of basic OLAP operations, including slice-and-dice, drill down, drill up, and pivoting.
- **Data Mining** - Data mining supports knowledge discovery by finding hidden patterns and associations, constructing analytical models, performing classification and prediction. These mining results can be presented using the visualization tools.

No.	Data Warehouse (OLAP)	Operational Database(OLTP)
1	It involves historical processing of information.	It involves day-to-day processing.
2	OLAP systems are used by knowledge workers such as executives, managers, and analysts.	OLTP systems are used by clerks, DBAs, or database professionals.
3	It is used to analyze the business.	It is used to run the business.
4	It focuses on Information out.	It focuses on Data in.
5	It is based on Star Schema, Snowflake Schema, and Fact Constellation Schema.	It is based on Entity Relationship Model.
6	It focuses on Information out.	It is application oriented.
7	It contains historical data.	It contains current data.
8	It provides summarized and consolidated data.	It provides primitive and highly detailed data.
9	It provides summarized and multidimensional view of data.	It provides detailed and flat relational view of data.
10	The number of users is in hundreds.	The number of users is in thousands.
11	The number of records accessed is in millions.	The number of records accessed is in tens.
12	The database size is from 100GB to 100 TB.	The database size is from 100 MB to 100 GB.
13	These are highly flexible.	It provides high performance.

Integrating Heterogeneous Databases

To integrate heterogeneous databases, we have two approaches:

- Query-driven Approach
- Update-driven Approach

Query-Driven Approach

This is the traditional approach to integrate heterogeneous databases. This approach was used to build wrappers and integrators on top of multiple heterogeneous databases. These integrators are also known as mediators.

Process of Query-Driven Approach

- When a query is issued to a client side, a metadata dictionary translates the query into an appropriate form for individual heterogeneous sites involved.
- Now these queries are mapped and sent to the local query processor.
- The results from heterogeneous sites are integrated into a global answer set.

Disadvantages

- Query-driven approach needs complex integration and filtering processes.
- This approach is very inefficient.
- It is very expensive for frequent queries.
- This approach is also very expensive for queries that require aggregations.

Update-Driven Approach

This is an alternative to the traditional approach. Today's data warehouse systems follow update-driven approach rather than the traditional approach discussed earlier. In update-driven approach, the information from multiple heterogeneous sources are integrated in advance and are stored in a warehouse. This information is available for direct querying and analysis.

Advantages

This approach has the following advantages:

- This approach provide high performance.
- The data is copied, processed, integrated, annotated, summarized and restructured in semantic data store in advance.
- Query processing does not require an interface to process data at local sources.

Functions of Data Warehouse Tools and Utilities

The following are the functions of data warehouse tools and utilities:

- **Data Extraction** - Involves gathering data from multiple heterogeneous sources.
- **Data Cleaning** - Involves finding and correcting the errors in data.
- **Data Transformation** - Involves converting the data from legacy format to warehouse format.
- **Data Loading** - Involves sorting, summarizing, consolidating, checking integrity, and building indices and partitions.
- **Refreshing** - Involves updating from data sources to warehouse.

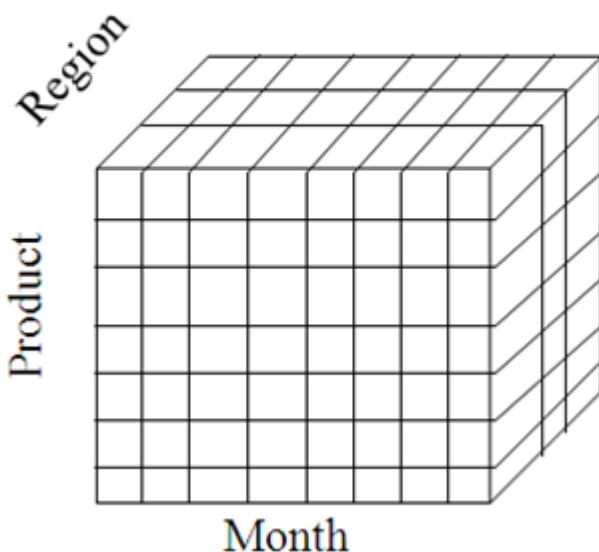
Note: Data cleaning and data transformation are important steps in improving the quality of data and data mining results.

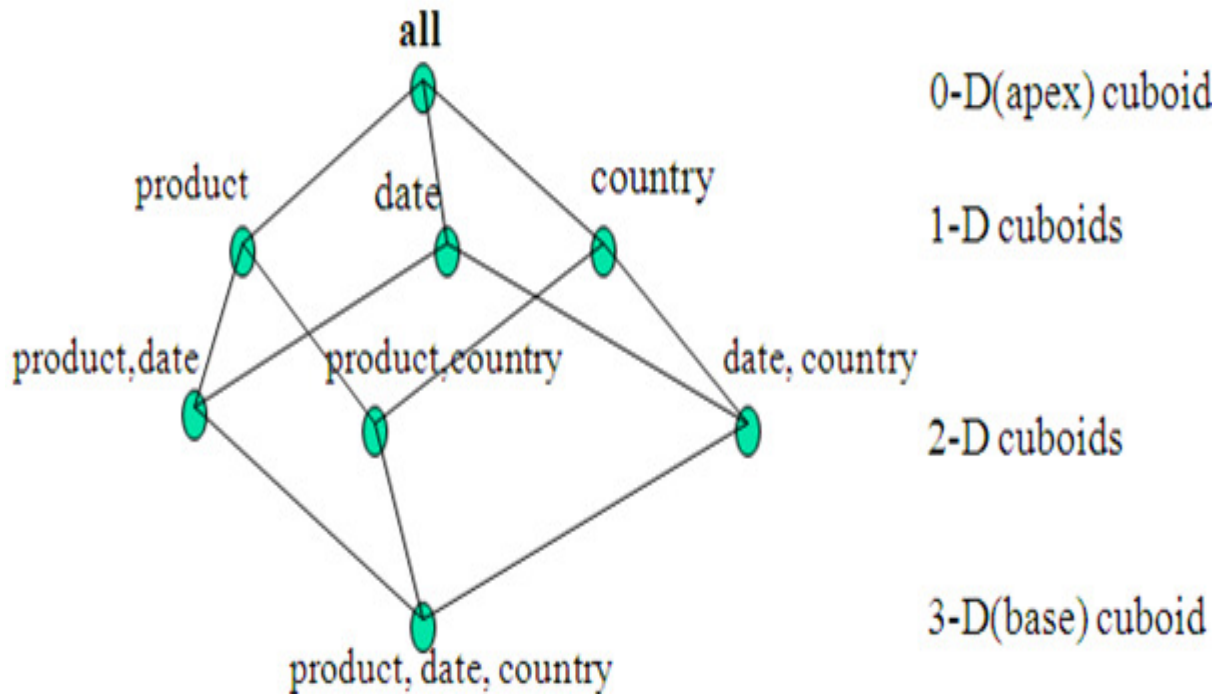
Multidimensional data model

Data warehouses and OLAP tools are based on a multidimensional data model. This model views data in the form of a *data cube*. In this section, you will learn how data cubes model n -dimensional data. You will also learn about concept hierarchies and how they can be used in basic OLAP operations to allow interactive mining at multiple levels of abstraction.

Sales volume as a function of product, month, and region

Dimensions: Product, Location, Time





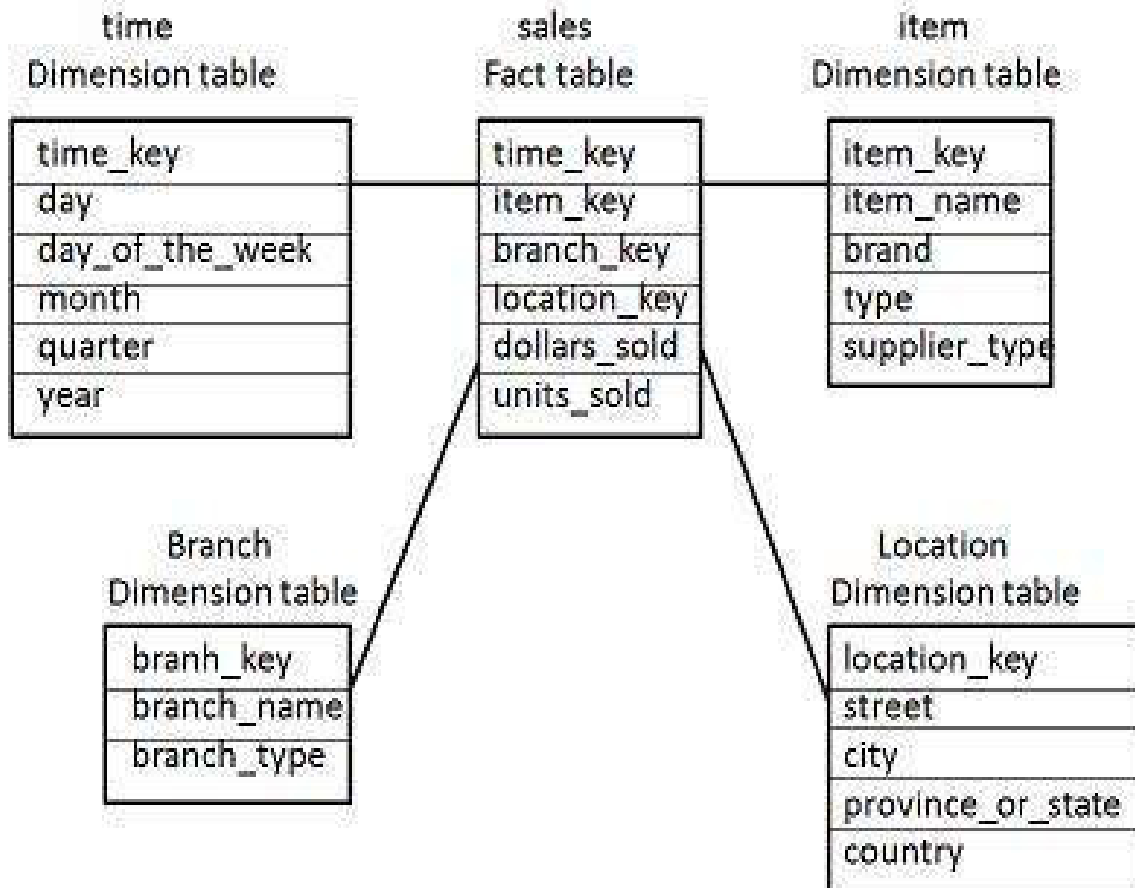
Cuboid Structure

Schema

Schema is a logical description of the entire database. It includes the name and description of records of all record types including all associated data-items and aggregates. Much like a database, a data warehouse also requires to maintain a schema. A database uses relational model, while a data warehouse uses Star, Snowflake, and Fact Constellation schema. In this chapter, we will discuss the schemas used in a data warehouse.

Star Schema

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

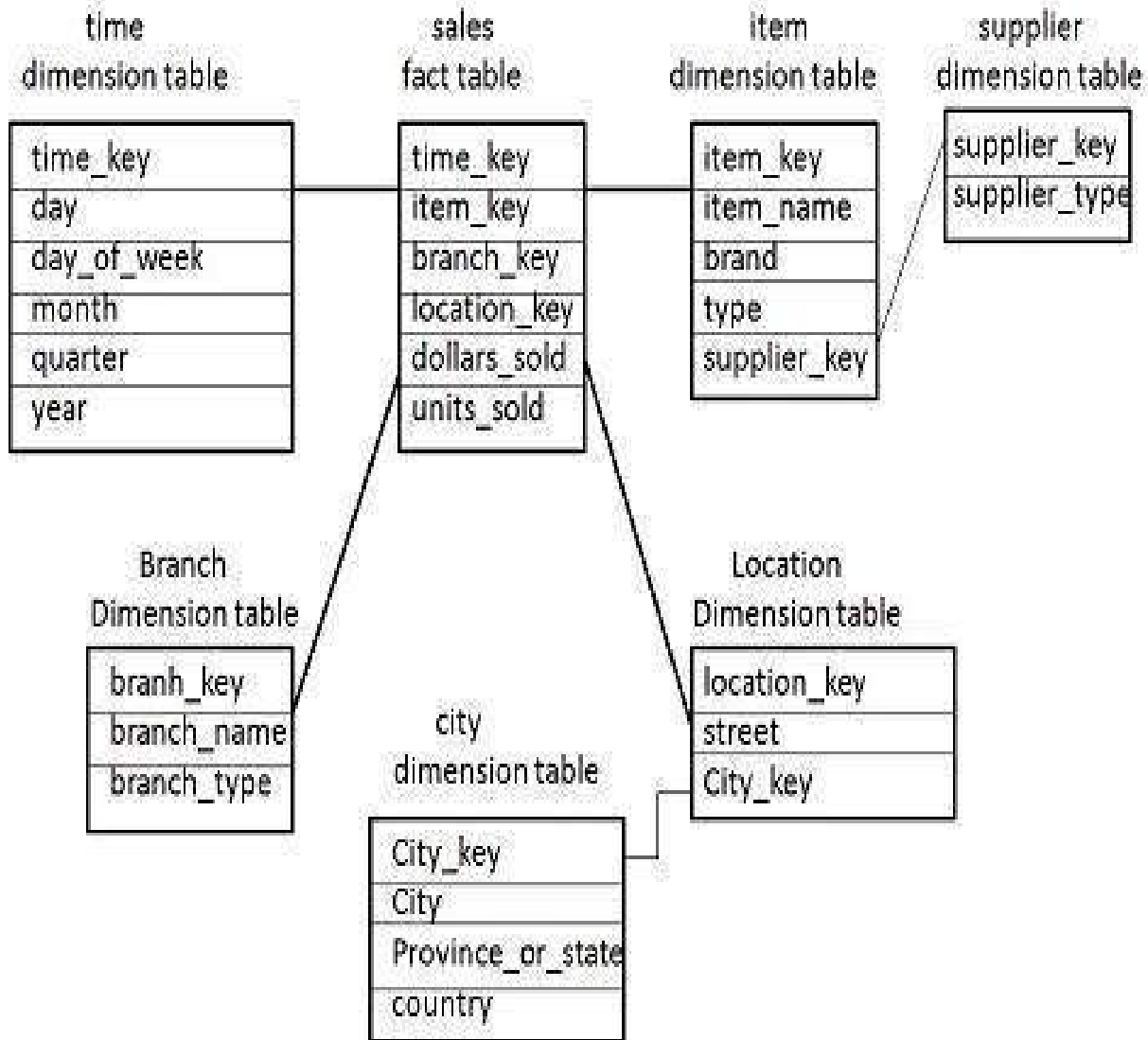


- There is a fact table at the center. It contains the keys to each of four dimensions.
- The fact table also contains the attributes, namely dollars sold and units sold.

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

Snowflake Schema

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

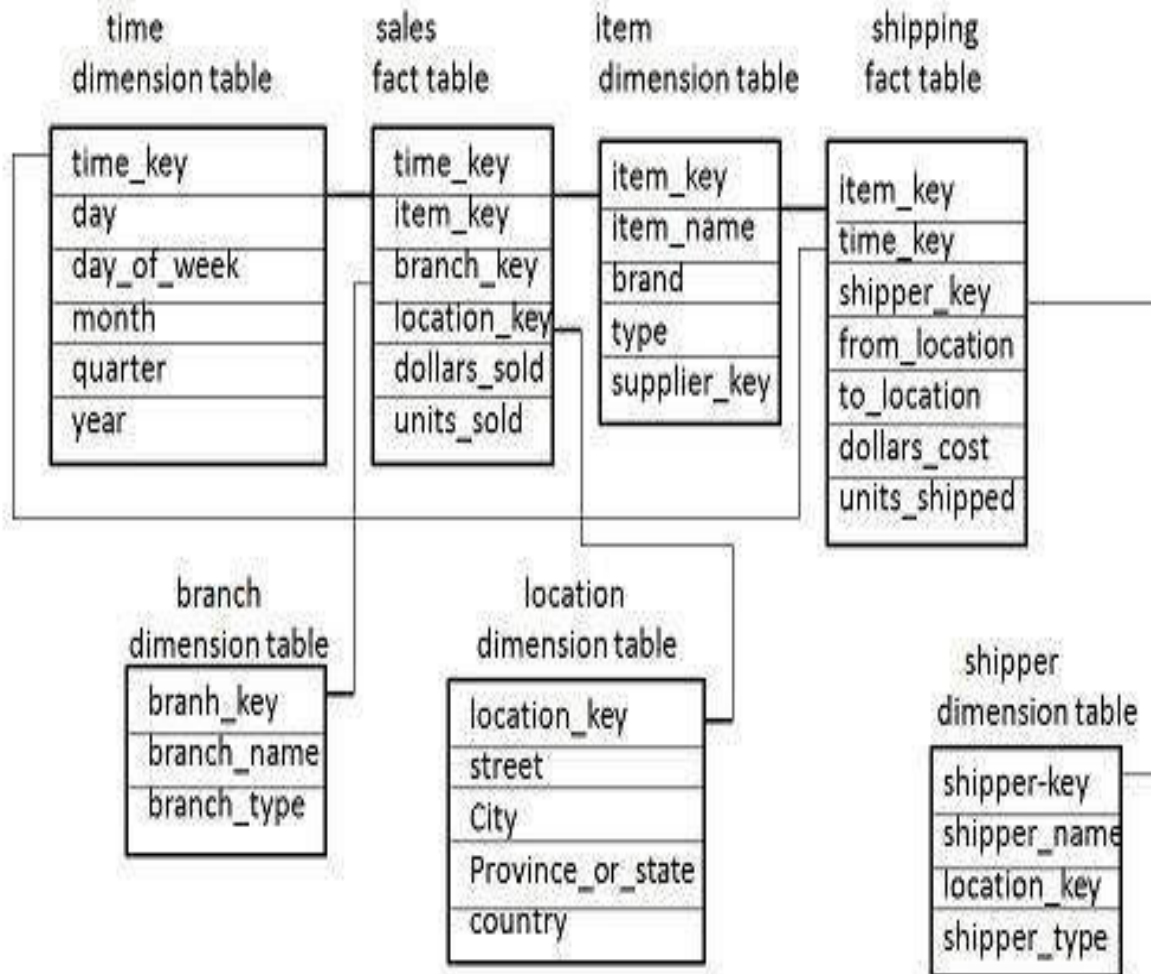


- Now the item dimension table contains the attributes item_key, item_name, type, brand, and supplier-key.
- The supplier key is linked to the supplier dimension table. The supplier dimension table contains the attributes supplier_key and supplier_type.

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

Fact Constellation Schema

- A fact constellation has multiple fact tables. It is also known as galaxy schema.
- The following diagram shows two fact tables, namely sales and shipping.



- The sales fact table is same as that in the star schema.
- The shipping fact table has the five dimensions, namely item_key, time_key, shipper_key, from_location, to_location.
- The shipping fact table also contains two measures, namely dollars sold and units sold.
- It is also possible to share dimension tables between fact tables. For example, time, item, and location dimension tables are shared between the sales and shipping fact table.

Schema Definition

Multidimensional schema is defined using Data Mining Query Language (DMQL). The two primitives, cube definition and dimension definition, can be used for defining the data warehouses and data marts.

Syntax for Cube Definition

```
define cube < cube_name > [ < dimension-list > ]: < measure_list >
```

Syntax for Dimension Definition

```
define dimension < dimension_name > as ( < attribute_or_dimension_list > )
```

Star Schema Definition

The star schema that we have discussed can be defined using Data Mining Query Language (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)
```

Snowflake Schema Definition

Snowflake schema can be defined using 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))
```

Fact Constellation Schema Definition

Fact constellation schema can be defined using 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)
define cube shipping [time, item, shipper, from location, to location]:
dollars cost = sum(cost in dollars), units shipped = count(*)
define dimension time as time in cube sales
define dimension item as item in cube sales
define dimension shipper as (shipper key, shipper name, location as location in cube sales, shipper type)
define dimension from location as location in cube sales
define dimension to location as location in cube sales
```

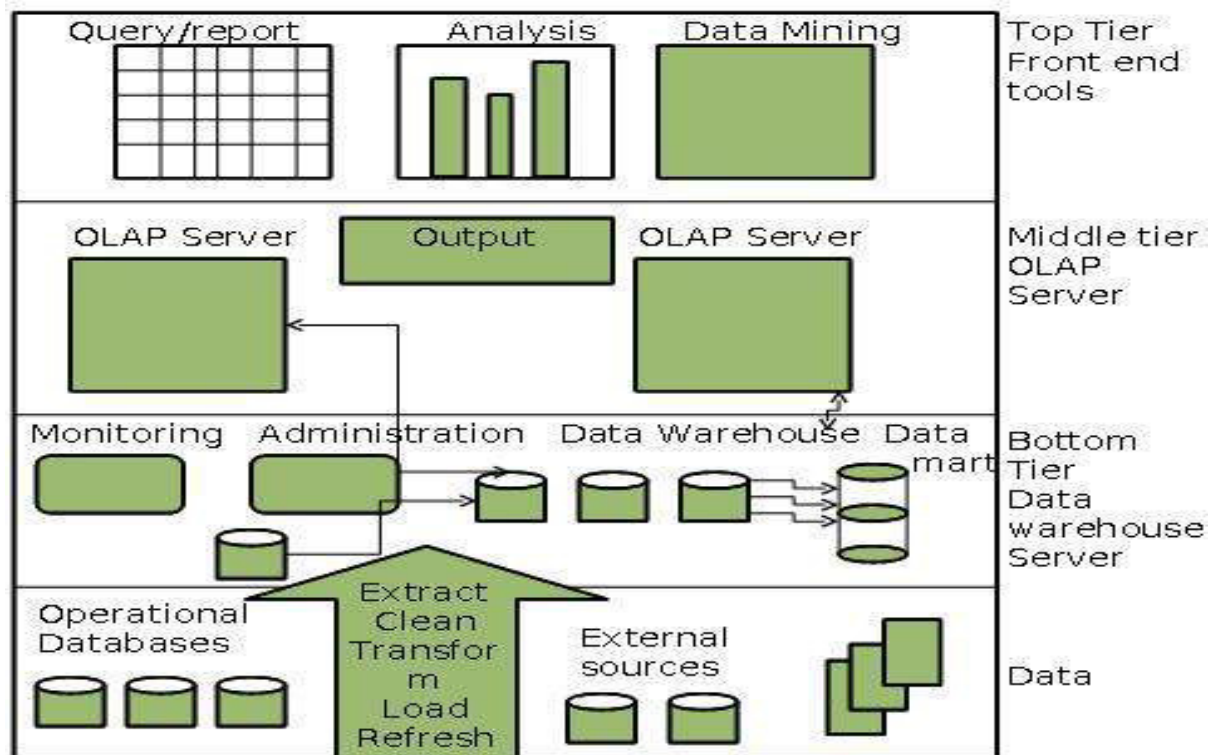
Data Warehouse Architecture

Three-Tier Data Warehouse Architecture

Generally a data warehouse adopts a three-tier architecture. Following are the three tiers of the data warehouse architecture.

- **Bottom Tier** - The bottom tier of the architecture is the data warehouse database server. It is the relational database system. We use the back end tools and utilities to feed data into the bottom tier. These back end tools and utilities perform the Extract, Clean, Load, and refresh functions.
- **Middle Tier** - In the middle tier, we have the OLAP Server that can be implemented in either of the following ways.
 - By Relational OLAP (ROLAP), which is an extended relational database management system. The ROLAP maps the operations on multidimensional data to standard relational operations.
 - By Multidimensional OLAP (MOLAP) model, which directly implements the multidimensional data and operations.
- **Top-Tier** - This tier is the front-end client layer. This layer holds the query tools and reporting tools, analysis tools and data mining tools.

The following diagram depicts the three-tier architecture of data warehouse:



Data Warehouse Models

From the perspective of data warehouse architecture, we have the following data warehouse models:

- Virtual Warehouse
- Data mart
- Enterprise Warehouse

Virtual Warehouse

The view over an operational data warehouse is known as a virtual warehouse. It is easy to build a virtual warehouse. Building a virtual warehouse requires excess capacity on operational database servers.

Data Mart

Data mart contains a subset of organization-wide data. This subset of data is valuable to specific groups of an organization.

In other words, we can claim that data marts contain data specific to a particular group. For example, the marketing data mart may contain data related to items, customers, and sales. Data marts are confined to subjects.

Points to remember about data marts:

- Window-based or Unix/Linux-based servers are used to implement data marts. They are implemented on low-cost servers.
- The implementation data mart cycles is measured in short periods of time, i.e., in weeks rather than months or years.
- The life cycle of a data mart may be complex in long run, if its planning and design are not organization-wide.
- Data marts are small in size.
- Data marts are customized by department.
- The source of a data mart is departmentally structured data warehouse.
- Data mart are flexible.

Enterprise Warehouse

- An enterprise warehouse collects all the information and the subjects spanning an entire organization
- It provides us enterprise-wide data integration.
- The data is integrated from operational systems and external information providers.
- This information can vary from a few gigabytes to hundreds of gigabytes, terabytes or beyond.

Data Warehouse Implementation

Data warehouses contain huge volumes of data. OLAP servers demand that decision support queries be answered in the order of seconds. Therefore, it is crucial for data warehouse systems to support highly efficient cube computation techniques, access methods, and query processing techniques. In this section, we present an overview of methods for the efficient implementation of data warehouse systems.

Efficient Computation of Data Cubes

At the core of multidimensional data analysis is the efficient computation of aggregations across many sets of dimensions. In SQL terms, these aggregations are referred to as group-by's. Each group-by can be represented by a *cuboid*, where the set of group-by's forms a lattice of cuboids defining a data cube. In this section, we explore issues relating to the efficient computation of data cubes.

The compute cube Operator and the Curse of Dimensionality

One approach to cube computation extends SQL so as to include a compute cube operator. The compute cube operator computes aggregates over all subsets of the dimensions specified in the operation. This can require excessive storage space, especially for large numbers of dimensions. We start with an intuitive look at what is involved in the efficient computation of data cubes.

Example 3.11 A data cube is a lattice of cuboids. Suppose that you would like to create a data cube for

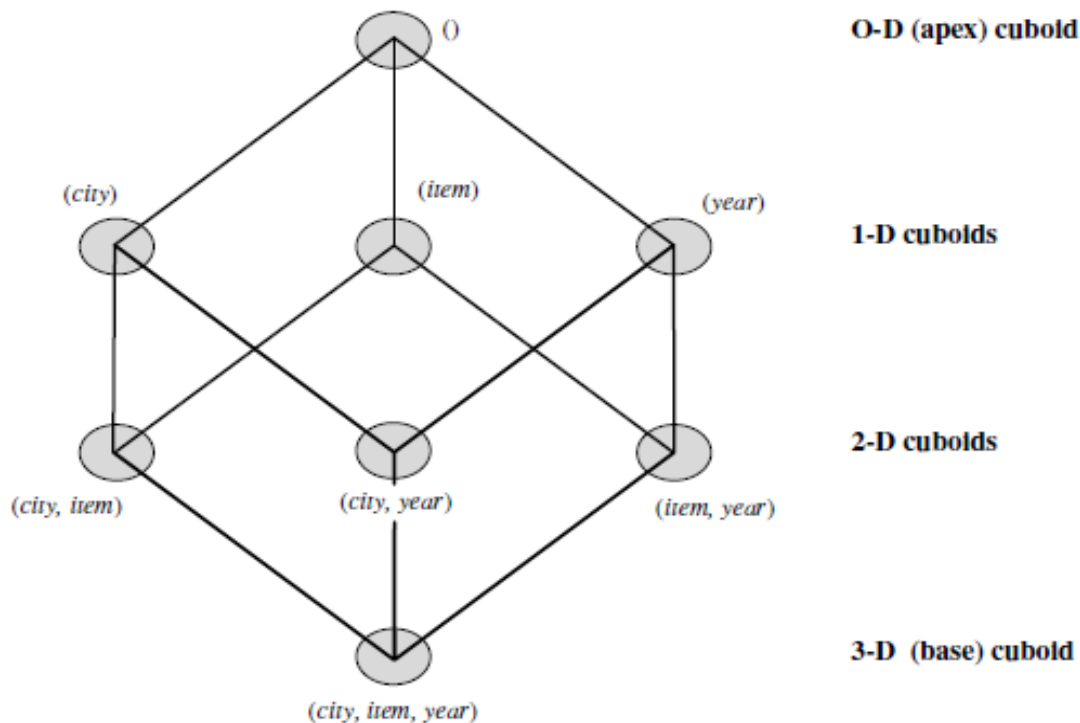
AllElectronics sales that contains the following: *city*, *item*, *year*, and *sales in dollars*. You would like to be able to analyze the data, with queries such as the following:

“Compute the sum of sales, grouping by city and item.”

“Compute the sum of sales, grouping by city.”

What is the total number of cuboids, or group-by's, that can be computed for this data cube? Taking the three attributes, *city*, *item*, and *year*, as the dimensions for the data cube, and *sales_in_dollars* as the measure, the total number of cuboids, or group-by's, that can be computed for this data cube is $2^3 = 8$. The possible group-by's are the following: $\{(city, item, year), (city, item), (city, year), (item, year), (city), (item), (year), ()\}$, where $()$ means that the group-by is empty (i.e., the dimensions are not grouped). These group-by's form a lattice of cuboids for the data cube, as shown in Figure 3.14. The **base cuboid** contains all three dimensions, *city*, *item*, and *year*. It can return the total sales for any combination of the three dimensions. The **apex cuboid**, or 0-D cuboid, refers to the case where the group-by is empty. It contains the total sum of all sales. The base cuboid is the least generalized (most specific) of the cuboids. The apex cuboid is the most generalized (least specific) of the cuboids, and is often denoted as *all*. If we start at the apex cuboid and explore downward in the lattice, this is equivalent to drilling down within the data cube. If we start at the base cuboid and explore upward, this is akin to rolling up.

An SQL query containing no group-by, such as “compute the sum of total sales,” is a *zero-dimensional operation*. An SQL query containing one group-by, such as “compute the sum of sales, group by city,” is a *one-dimensional operation*. A cube operator on n dimensions is equivalent to a collection of **group by** statements, one for each subset



Lattice of cuboids, making up a 3-D data cube. Each cuboid represents a different group-by. The base cuboid contains the three dimensions *city*, *item*, and *year*.

of the n dimensions. Therefore, the cube operator is the n -dimensional generalization of the **group by** operator.

Based on the syntax of DMQL introduced in Section 3.2.3, the data cube in Example 3.11 could be defined as

```
define cube sales_cube [city, item, year]: sum(sales_in_dollars)
```

For a cube with n dimensions, there are a total of 2^n cuboids, including the base cuboid. A statement such as

```
compute cube sales_cube
```

would explicitly instruct the system to compute the sales aggregate cuboids for all of the eight subsets of the set $\{city, item, year\}$, including the empty subset. A cube computation operator was first proposed and studied by Gray et al. [GCB⁺97].

On-line analytical processing may need to access different cuboids for different queries. Therefore, it may seem like a good idea to compute all or at least some of the cuboids in a data cube in advance. Precomputation leads to fast response time and avoids some redundant computation. Most, if not all, OLAP products resort to some degree of precomputation of multidimensional aggregates.

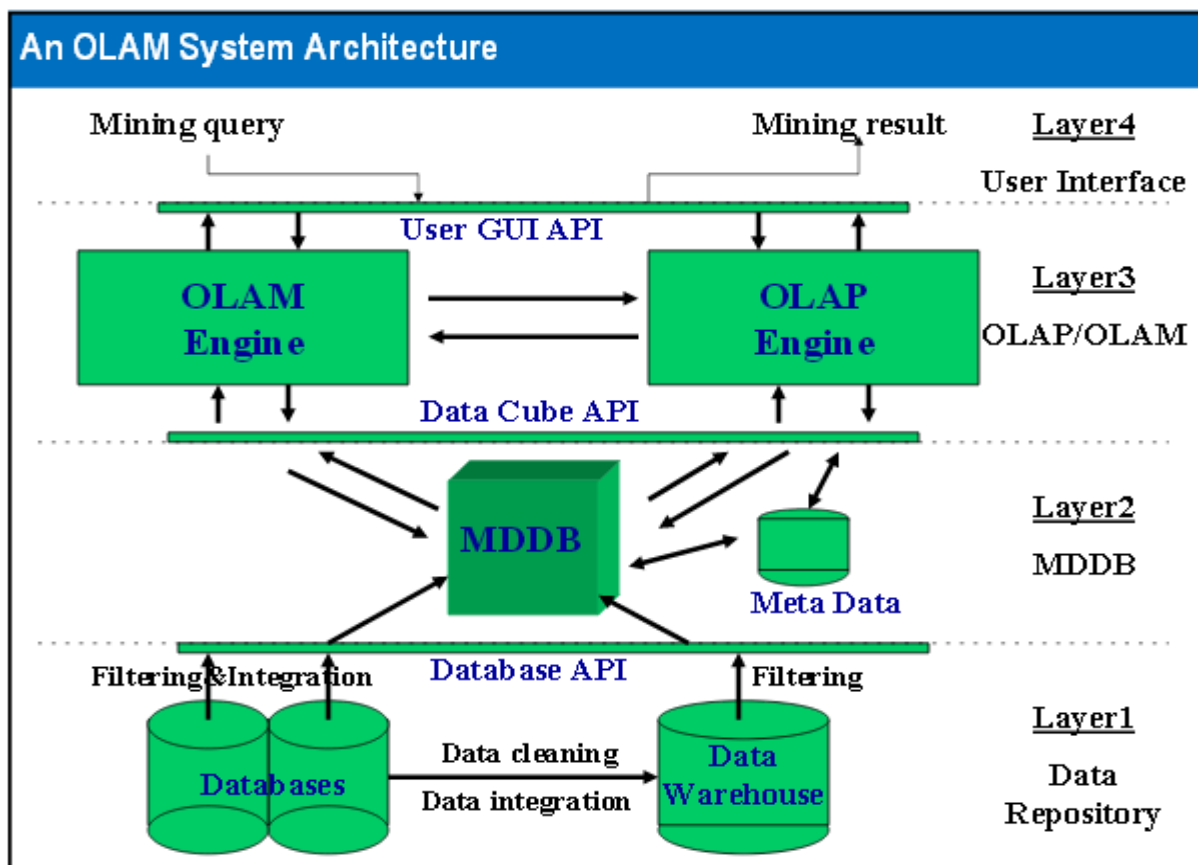
A major challenge related to this precomputation, however, is that the required storage space may explode if all of the cuboids in a data cube are precomputed, especially when the cube has many dimensions. The storage requirements are even more excessive when many of the dimensions have associated concept hierarchies, each with multiple levels. This problem is referred to as the **curse of dimensionality**. ⁵

From Data warehousing to Data Mining

From OLAP to On Line Analytical Mining (OLAM)

Why online analytical mining?

- » High quality of data in data warehouses
 - DW contains integrated, consistent, cleaned data
- » Available information processing structure surrounding data warehouses
 - ODBC, OLEDB, Web accessing, service facilities, reporting and OLAP tools
- » OLAP-based exploratory data analysis
 - Mining with drilling, dicing, pivoting, etc.
- » On-line selection of data mining functions
 - Integration and swapping of multiple mining functions, algorithms, and tasks



OLAP- Need - Categorization of OLAP Operations

Online Analytical Processing Server (OLAP) is based on the multidimensional data model. It allows managers, and analysts to get an insight of the information through fast, consistent, and interactive access to information. This chapter cover the types of OLAP, operations on OLAP, difference between OLAP, and statistical databases and OLTP.

Types of OLAP Servers

We have four types of OLAP servers:

- Relational OLAP (ROLAP)
- Multidimensional OLAP (MOLAP)
- Hybrid OLAP (HOLAP)
- Specialized SQL Servers

Relational OLAP

ROLAP servers are placed between relational back-end server and client front-end tools. To store and manage warehouse data, ROLAP uses relational or extended-relational DBMS.

ROLAP includes the following:

- Implementation of aggregation navigation logic.
- Optimization for each DBMS back end.
- Additional tools and services.

Multidimensional OLAP

MOLAP uses array-based multidimensional storage engines for multidimensional views of data. With multidimensional data stores, the storage utilization may be low if the data set is sparse. Therefore, many MOLAP server use two levels of data storage representation to handle dense and sparse data sets.

Hybrid OLAP (HOLAP)

Hybrid OLAP is a combination of both ROLAP and MOLAP. It offers higher scalability of ROLAP and faster computation of MOLAP. HOLAP servers allows to store the large data volumes of detailed information. The aggregations are stored separately in MOLAP store.

Specialized SQL Servers

Specialized SQL servers provide advanced query language and query processing support for SQL queries over star and snowflake schemas in a read-only environment.

OLAP Operations

Since OLAP servers are based on multidimensional view of data, we will discuss OLAP operations in multidimensional data.

Here is the list of OLAP operations:

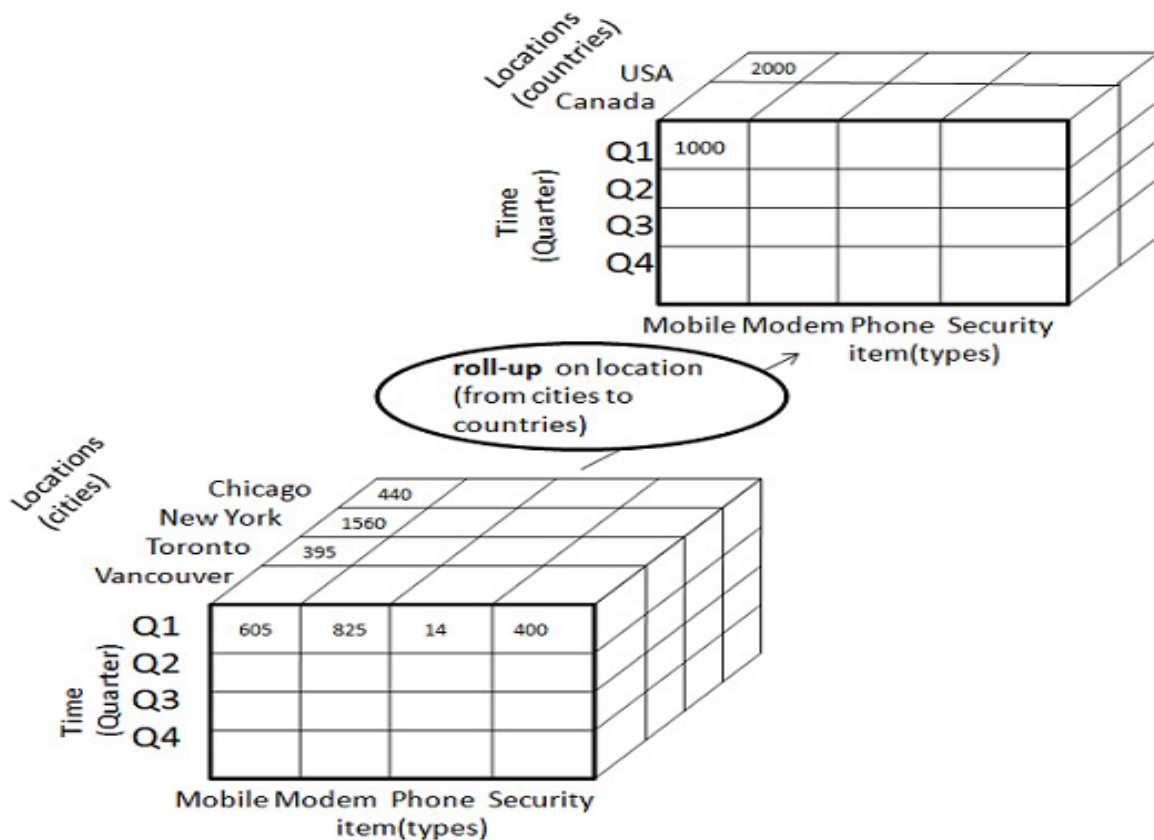
- Roll-up
- Drill-down
- Slice and dice
- Pivot (rotate)

Roll-up

Roll-up performs aggregation on a data cube in any of the following ways:

- By climbing up a concept hierarchy for a dimension
- By dimension reduction

The following diagram illustrates how roll-up works.



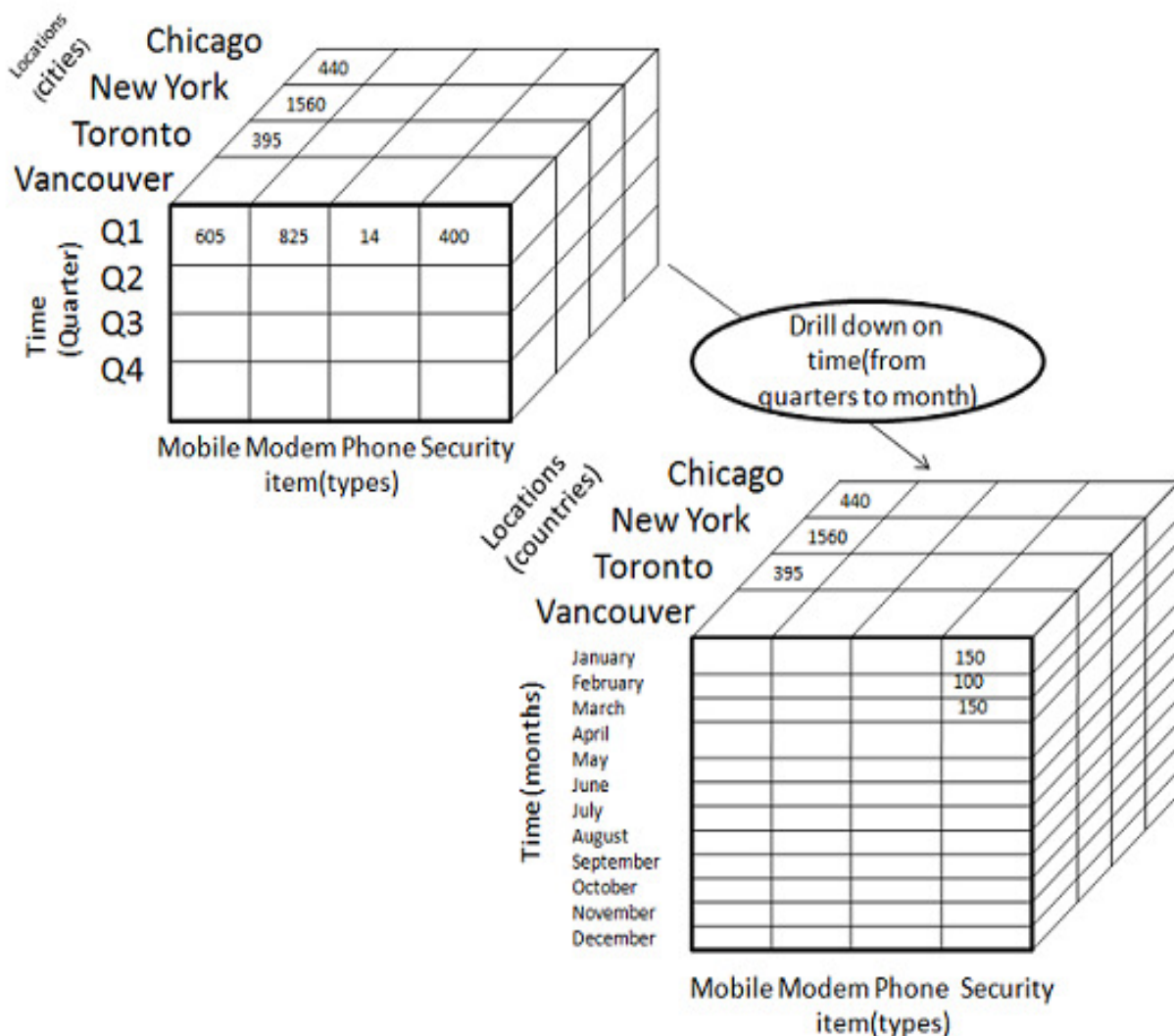
- Roll-up is performed by climbing up a concept hierarchy for the dimension location.
- Initially the concept hierarchy was "street < city < province < country".
- On rolling up, the data is aggregated by ascending the location hierarchy from the level of city to the level of country.
- The data is grouped into cities rather than countries.
- When roll-up is performed, one or more dimensions from the data cube are removed.

Drill-down

Drill-down is the reverse operation of roll-up. It is performed by either of the following ways:

- By stepping down a concept hierarchy for a dimension
- By introducing a new dimension.

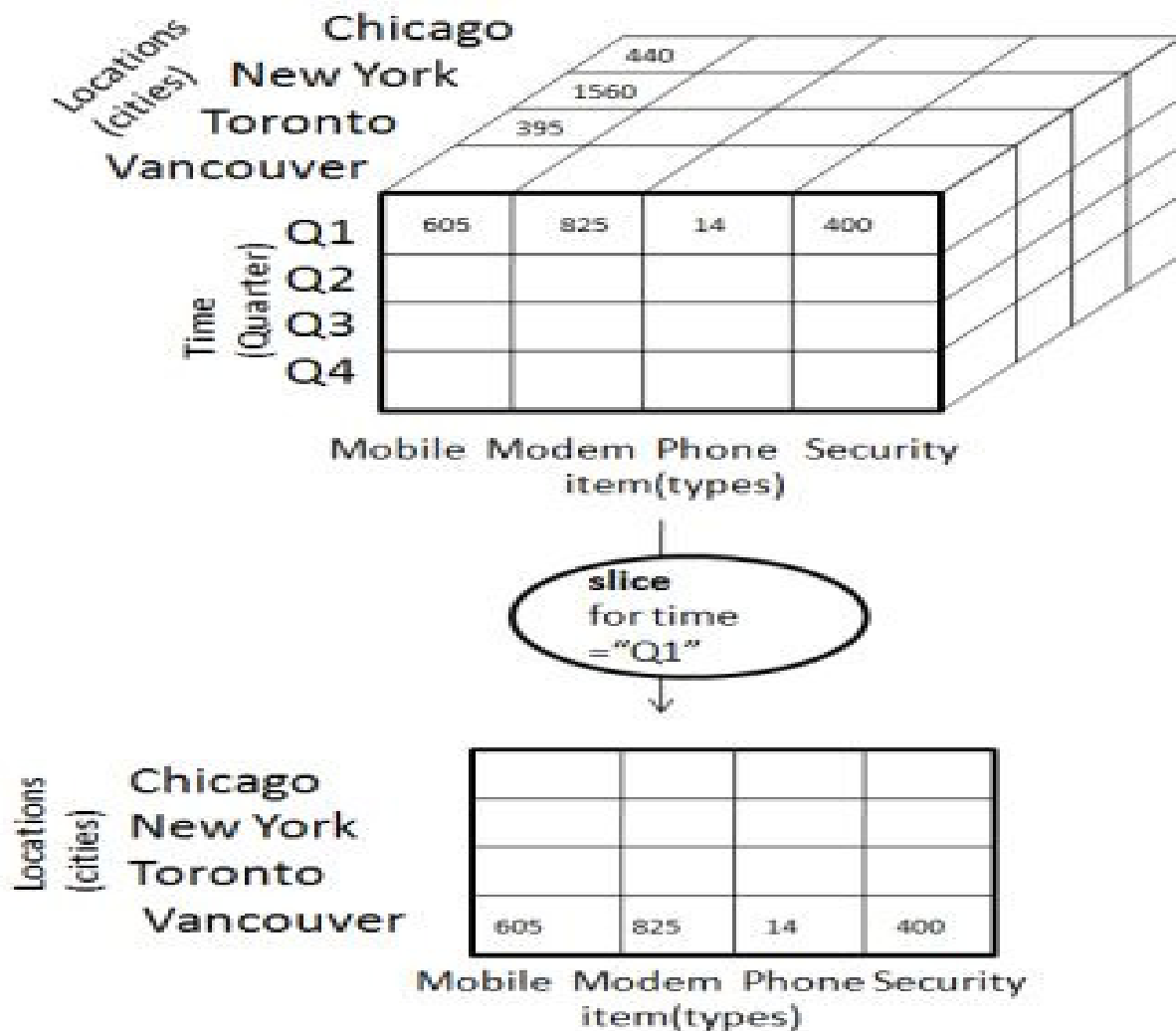
The following diagram illustrates how drill-down works:



- Drill-down is performed by stepping down a concept hierarchy for the dimension time.
- Initially the concept hierarchy was "day < month < quarter < year."
- On drilling down, the time dimension is descended from the level of quarter to the level of month.
- When drill-down is performed, one or more dimensions from the data cube are added.
- It navigates the data from less detailed data to highly detailed data.

Slice

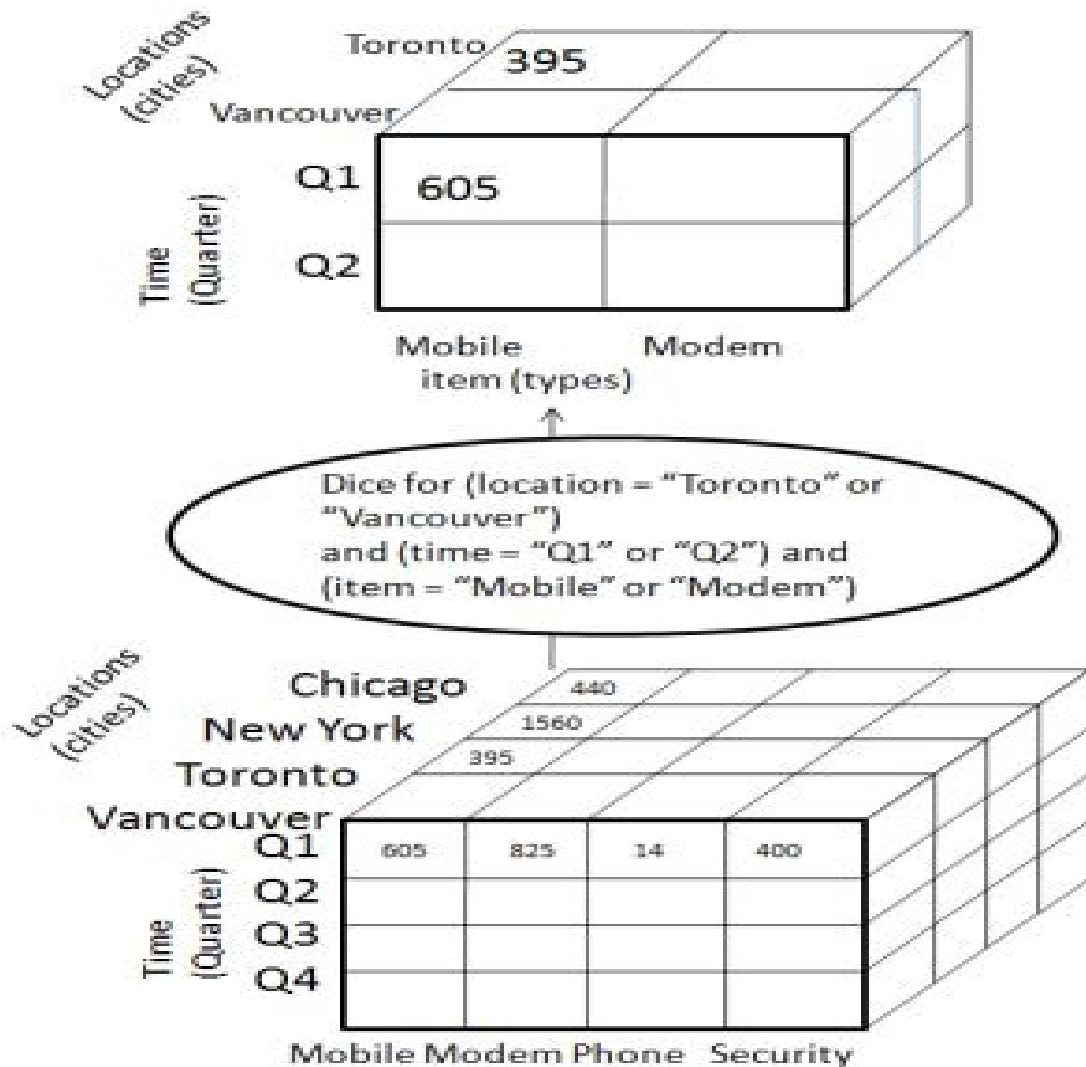
The slice operation selects one particular dimension from a given cube and provides a new sub-cube. Consider the following diagram that shows how slice works.



- Here Slice is performed for the dimension "time" using the criterion time = "Q1".
- It will form a new sub-cube by selecting one or more dimensions.

Dice

Dice selects two or more dimensions from a given cube and provides a new sub-cube. Consider the following diagram that shows the dice operation.

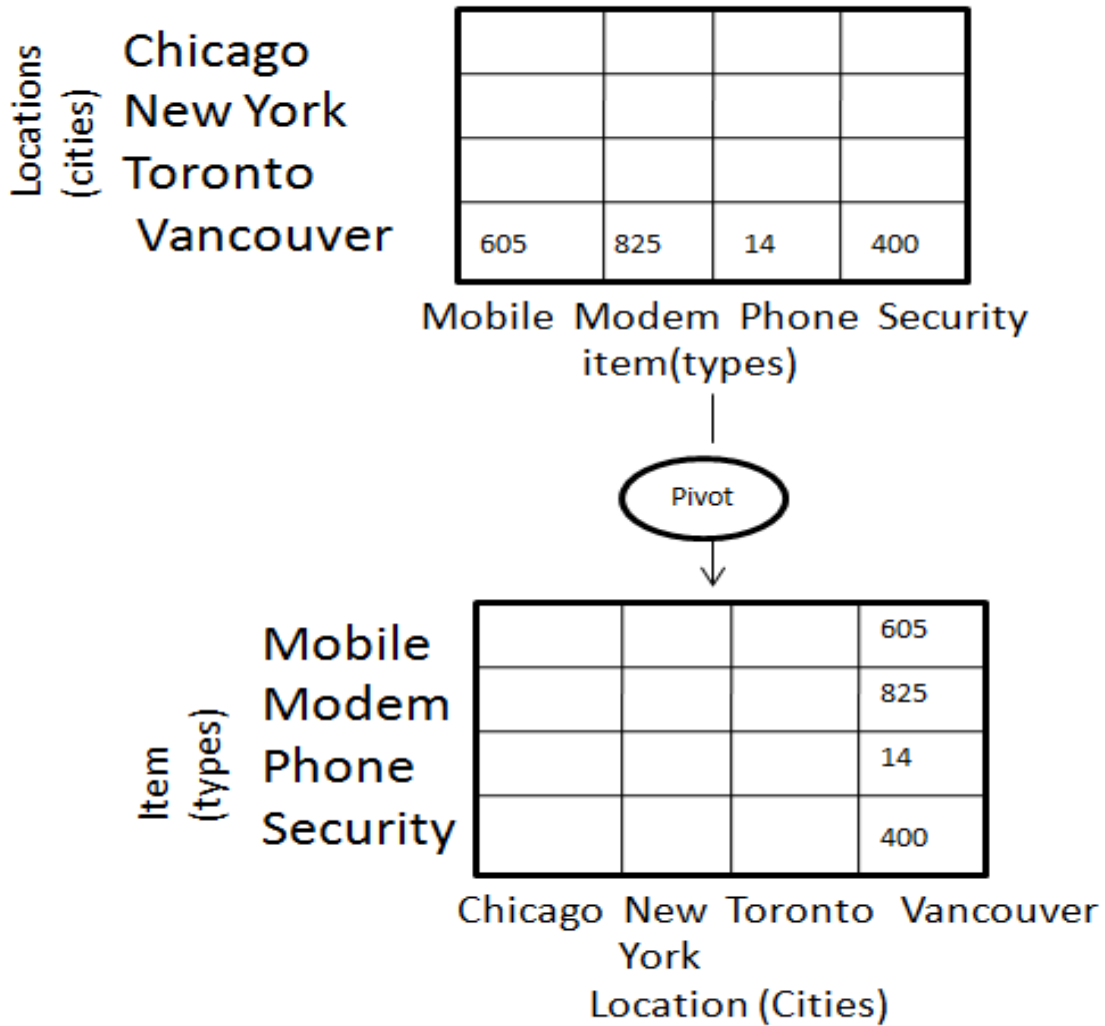


The dice operation on the cube based on the following selection criteria involves three dimensions.

- (location = "Toronto" or "Vancouver")
- (time = "Q1" or "Q2")
- (item = "Mobile" or "Modem")

Pivot

The pivot operation is also known as rotation. It rotates the data axes in view in order to provide an alternative presentation of data. Consider the following diagram that shows the pivot operation.



In this the item and location axes in 2-D slice are rotated.