# WHAT IS DATA WAREHOUSE?

A decision support database that is maintained separately from the organization's operational database and Supports information processing by providing a solid platform of consolidated, historical data for analysis.

(or)

A repository of multiple heterogeneous data sources organized under a unified schema at a single site in order to facilitate management decision making.

- "A data warehouse is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management's decision-making process."
- The process of constructing and using data warehouses Data warehousing

# KEY FEATURES OF A DATA WAREHOUSE.

**Subject-oriented:** 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. Such as customer,supplier, product, and sales etc..

**Integrated:** A data warehouse is usually constructed by integrating multiple heterogeneous sources, such as relational databases, flat files, and on-line transaction records.

**Time-variant**: Data are stored to provide information from a historical perspective (e.g., the past 5–10 years).

**Nonvolatile**: A data warehouse is always a physically separate store of data transformed from the application data found in the operational environment. It usually requires only two operations in data accessing: initial loading of data and access of data.

# DIFFERENCES BETWEEN OPERATIONAL DATABASE SYSTEMS AND DATA WAREHOUSES

| *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 |

# A MULTIDIMENSIONAL DATA MODEL

**Why Separate Data Warehouse?**

- High performance for both systems
  - DBMS— Tuned for OLTP: access methods, indexing, concurrency control, recovery
  - Warehouse—Tuned for OLAP: complex OLAP queries, multidimensional view, consolidation
- Different functions and different data:
  - **MISSING DATA**: Decision support requires historical data which operational DBs do not typically maintain
  - **DATA CONSOLIDATION**: DS requires consolidation (aggregation, summarization) of data from heterogeneous sources
  - **DATA QUALITY:** different sources typically use inconsistent data representations, codes and formats which have to be reconciled
- Note: There are more and more systems which perform OLAP analysis directly on relational databases

## FROM TABLES AND SPREADSHEETS TO DATA CUBES (WHY TO USE DATA CUBES INSTEAD OF TABLES AND SPREAD SHEETS)

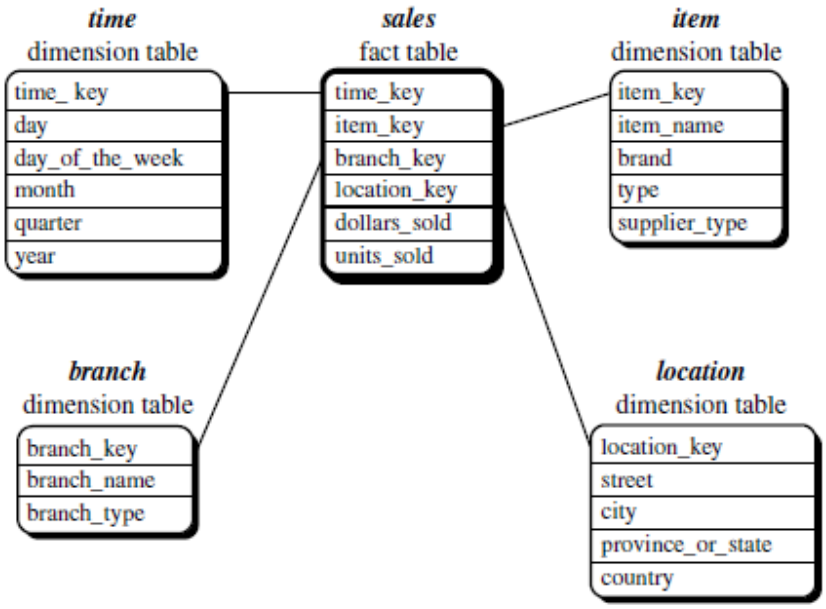
- A data warehouse is based on a multidimensional data model which views data in the form of a data cube
- A data cube, such as sales, allows data to be modeled and viewed in multiple dimensions
  - Dimension tables, such as item (item_name, brand, type), or time(day, week, month, quarter, year)
  - Fact table contains measures (such as dollars_sold) and keys to each of the related dimension tables

In data warehousing literature, an n-D base cube is called a base cuboid. The top most 0-D cuboid, which holds the highest-level of summarization, is called the apex cuboid. The lattice of cuboids forms a data cube.

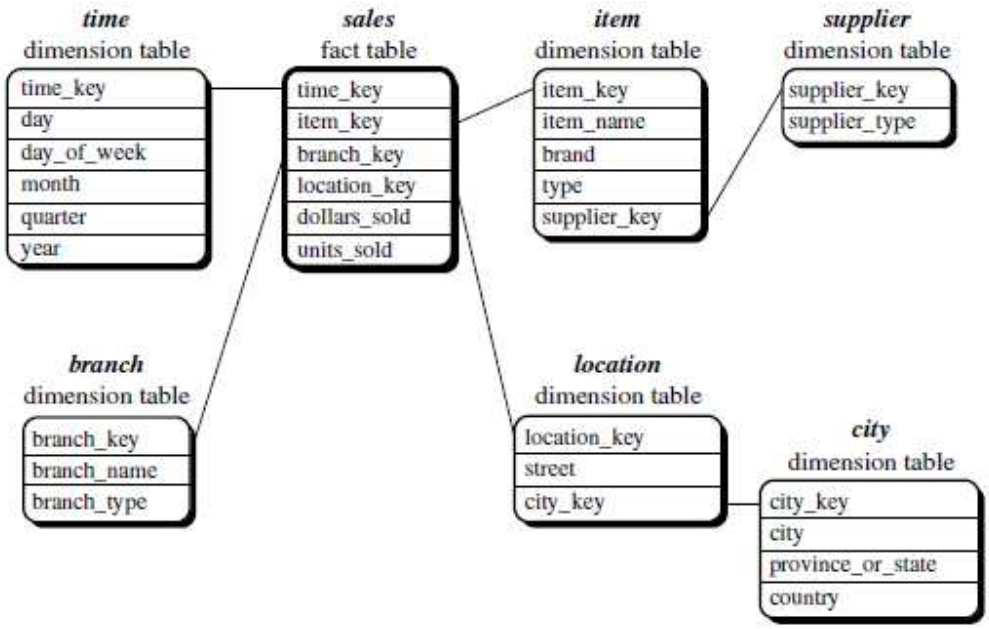
# STARS, SNOWFLAKES, AND FACT CONSTELLATIONS SCHEMAS FOR MULTIDIMENSIONAL DATABASES

- The entity-relationship data model is commonly used in the design of relational databases.
- In the same way multi dimensional model is used for designing data warehouse. Such a model can exist in the form of a star schema, a snowflake schema, or a fact constellation schema

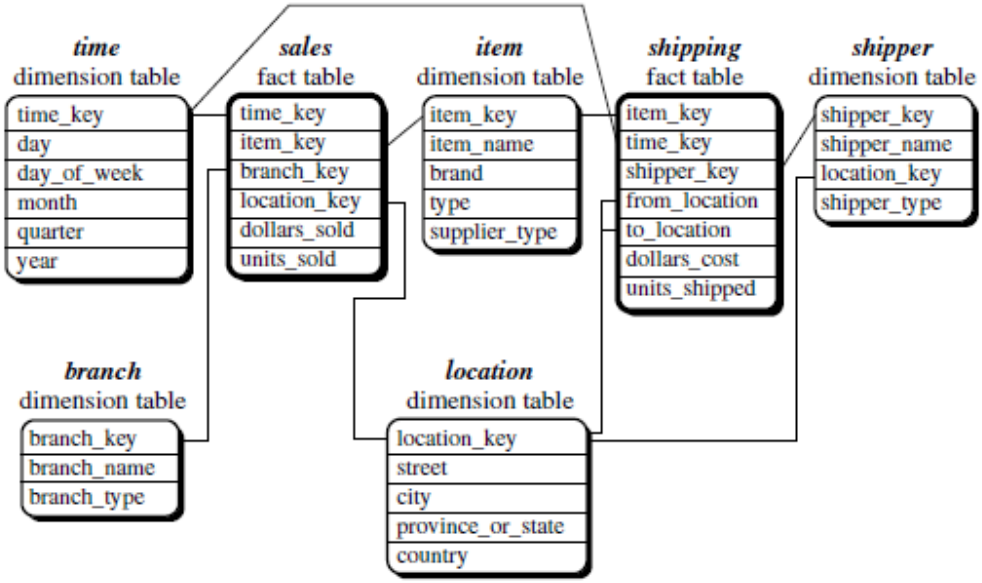
**Star schema:** The most common modeling paradigm is the star schema, in which the data warehouse contains.
(1) A large central table (fact table) containing the bulk of the data and
(2) A set of smaller attendant tables (dimension tables), one for each dimension.




**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.



**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.



## EXAMPLES FOR DEFINING STAR, SNOWFLAKE, AND FACT CONSTELLATION SCHEMAS

## BASIC SYNTAX

- Cube Definition (Fact Table)
  define cube <cube_name> [<dimension_list>]: <measure_list>

- Dimension Definition (Dimension Table)
  define dimension <dimension_name> as (<attribute_or_subdimension_list>)

- Special Case (Shared Dimension Tables)
  - define dimension <dimension_name> as <dimension_name_first_time> in cube <cube_name_first_time>

## DEFINING STAR SCHEMA IN DMQL

define cube sales_star [time, item, branch, location]:
dollars_sold = sum(sales_in_dollars), avg_sales = avg(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**)**

## DEFINING SNOWFLAKE SCHEMA IN DMQL

**define cube** sales_snowflake [time, item, branch, location]:
dollars_sold = sum(sales_in_dollars), avg_sales = avg(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, province_or_state, country))

## DEFINING FACT CONSTELLATION IN DMQL

**define cube** sales [time, item, branch, location]:
dollars_sold = sum(sales_in_dollars), avg_sales = avg(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]:
dollar_cost = sum(cost_in_dollars), unit_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

# MEASURES OF DATA CUBE: THREE CATEGORIES

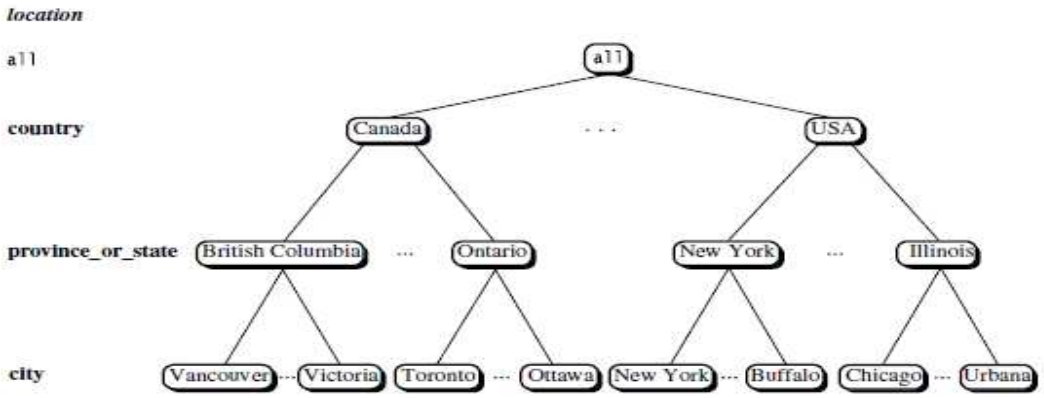
- **Distributive:** if the result derived by applying the function to *n* aggregate values is the same as that derived by applying the function on all the data without partitioning
  - E.g., count(), sum(), min(), max()
- **Algebraic:** if it can be computed by an algebraic function with *M* arguments (where *M* is a bounded integer), each of which is obtained by applying a distributive aggregate function
  - E.g., avg(), min_N(), standard_deviation()
- **Holistic:** if there is no constant bound on the storage size needed to describe sub aggregate.
  - E.g., median(), mode(), rank()

# CONCEPT HIERARCHIES

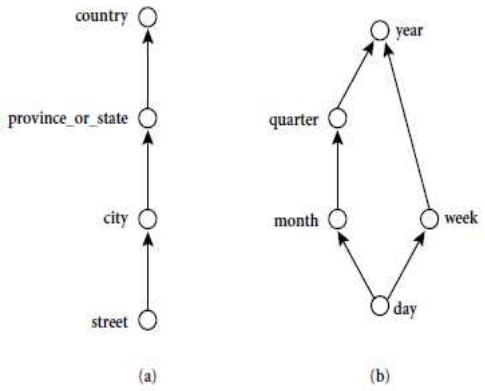
A concept hierarchy defines a sequence of mappings from a set of low-level concepts to higher-level, more general concepts.

**SCHEMA HIERARCHY**: A concept hierarchy that is a total or partial order among attributes in a database schema is called a schema hierarchy. Schema hierarchy may formally express existing relationship between attributes.

.

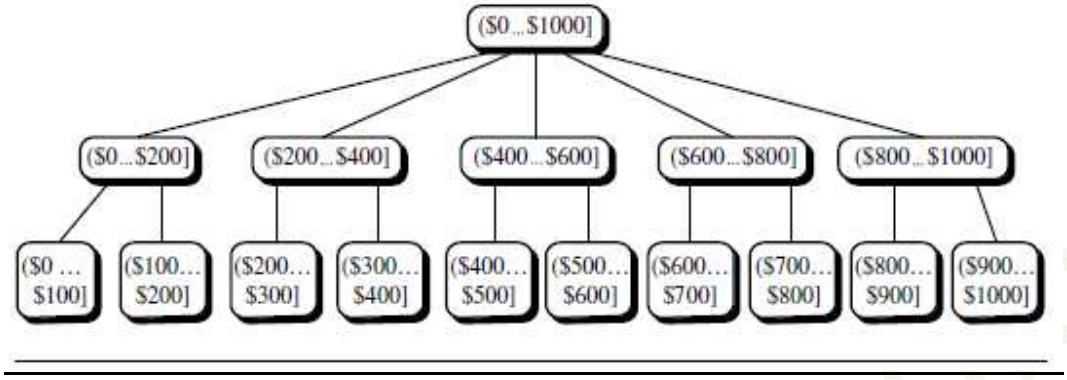



A concept hierarchy for the dimension *location*




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

**SET-GROUPING HIERARCHY:** Concept hierarchies may also be defined by discretizing or grouping values for a given dimension or attribute, resulting in a set-grouping hierarchy. A total or partial order can be defined among groups of values.



# OLAP OPERATIONS IN THE MULTIDIMENSIONAL DATA MODEL

**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*.

- Ex:roll-up operation aggregates data by ascending the location hierarchy from the level of city to the level of country

**Drill-down** : It 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*.

- Ex: drill-down for time
- day<month<quarter<year
- form the level of quarter to the more detailed level of month

**Slice:** a selection on one dimension of the cube resulting in subcube

Ex: sale data are selected for dimension time using time =Q1

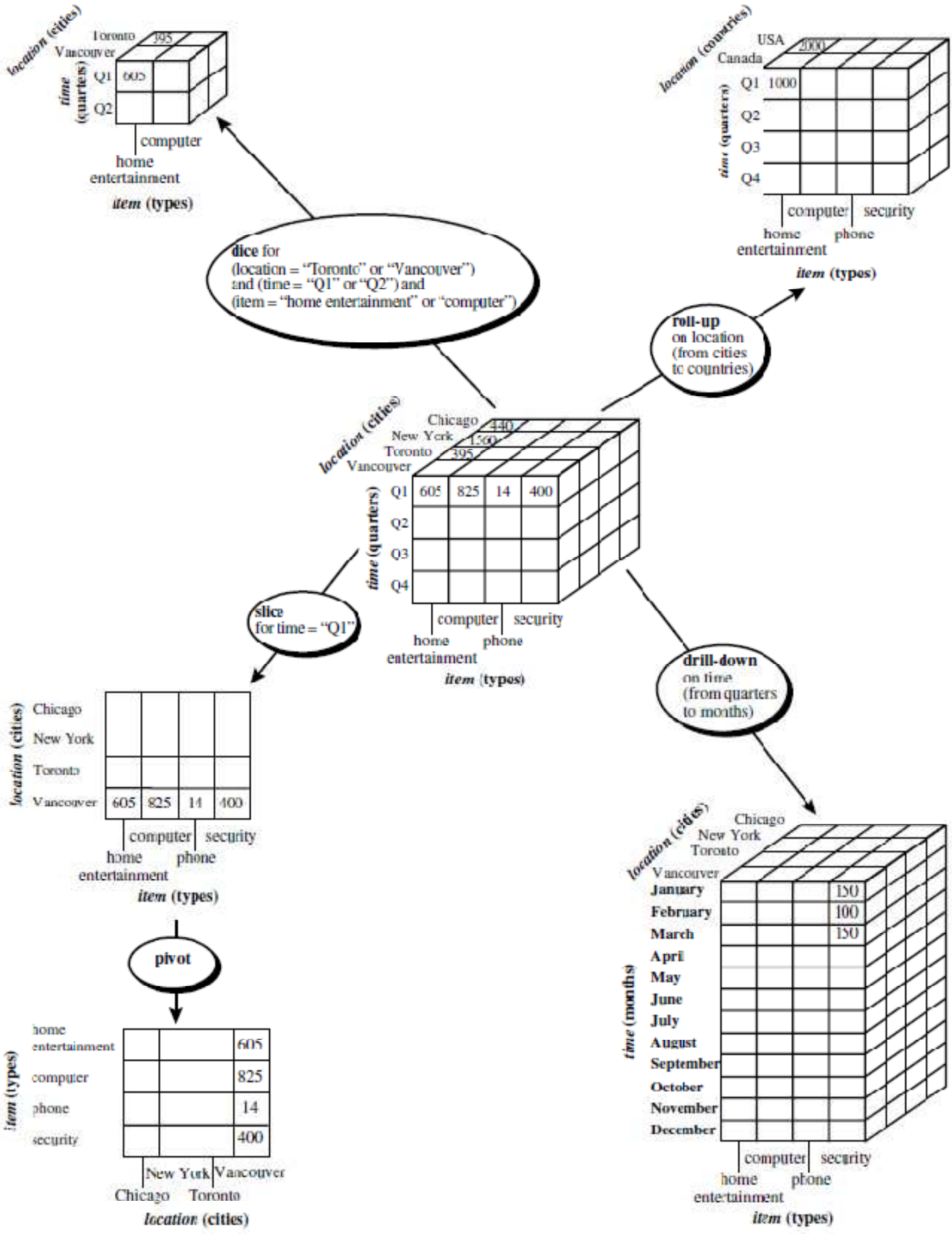
**dice:** defines a subcube by performing a selection on two or more dimensions

Ex: a dice opp. Based on

location="toronto" or "vencover" and

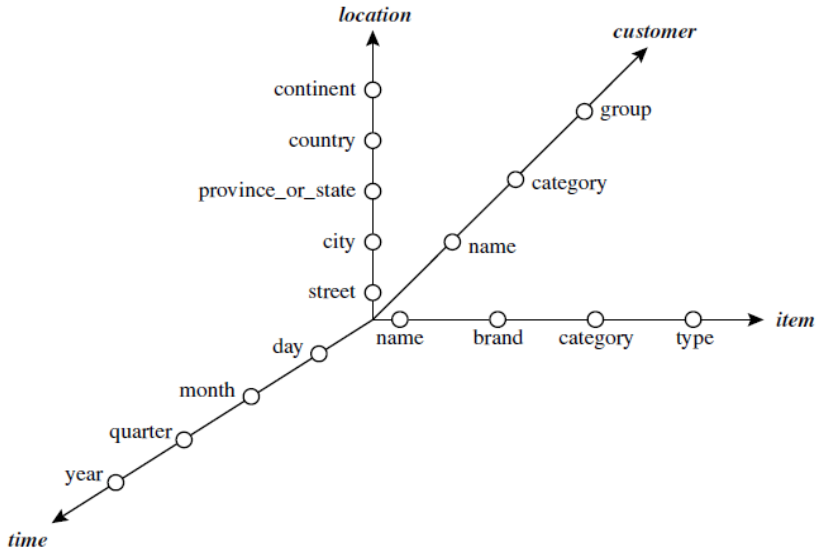
**time =Q1 or Q2 and**

**item = "home entertainment" or "computer"**


location (cities)
Toronto
Vancouver
395
time (quarters)
Q1
605
Q2
computer
home entertainment
item (types)
location (countries)
USA
2000
Canada
Q1 1000
Q2
Q3
Q4
time (quarters)
computer
security
home entertainment
phone
item (types)
dice for (location = "Toronto" or "Vancouver") and (time = "Q1" or "Q2") and (item = "home entertainment" or "computer")
roll-up on location (from cities to countries)
location (cities)
Chicago 440
New York 1560
Toronto 395
Vancouver
Q1 605 825 14 400
Q2
Q3
Q4
time (quarters)
computer
security
home entertainment
phone
item (types)
slice for time = "Q1"
drill-down on time (from quarters to months)
location (cities)
Chicago
New York
Toronto
Vancouver 605 825 14 400
computer
security
home entertainment
phone
item (types)
pivot
home entertainment 605
computer 825
phone 14
security 400
item (types)
New York
Vancouver
Chicago
Toronto
location (cities)
location (cities)
Chicago
New York
Toronto
Vancouver
January 150
February 100
March 150
April
May
June
July
August
September
October
November
December
time (months)
computer
security
home entertainment
phone
item (types)

# A STARNET QUERY MODEL FOR QUERYING MULTIDIMENSIONAL DATABASES

A starnet model consists of radial lines emanating from a central point, where each line represents a concept hierarchy for a dimension. Each abstraction level in the hierarchy is called a footprint. These represent the granularities available for use by OLAP operations such as drill-down and roll-up.



# DATA WAREHOUSE ARCHITECTURE

Design of Data Warehouse: A Business Analysis Framework

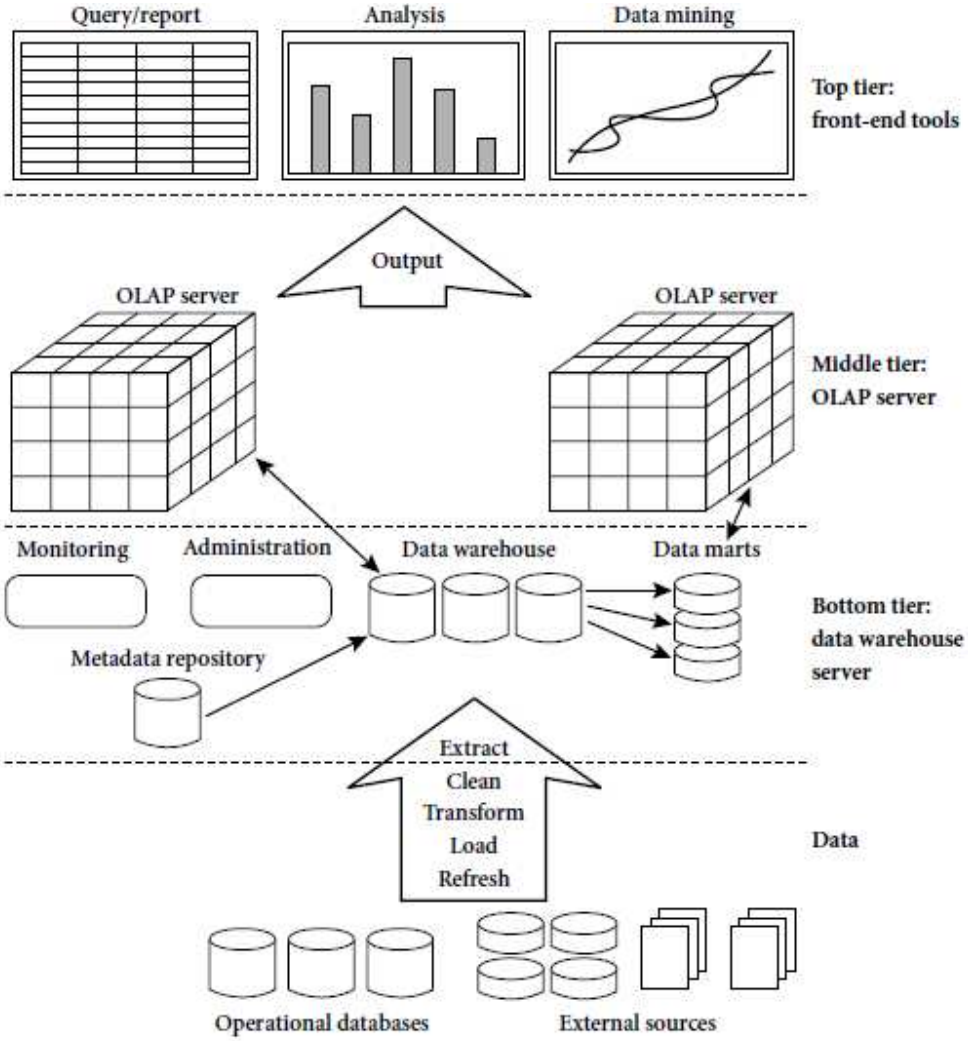
Four views regarding the design of a data warehouse:

- Top-down view :Allows selection of the relevant information necessary for the data warehouse
- Data source view :Exposes the information being captured, stored, and managed by operational systems
- Data warehouse view: Includes fact tables and dimension tables. It represents the information that is stored inside the data warehouse, including precalculated totals and counts, as well as information regarding the source, date, and time of origin
- Business query view : Sees the perspectives of data in the warehouse from the view of end-user

Data Warehouse Design Process

- Top-down, bottom-up approaches or a combination of both
  - Top-down: Starts with overall design and planning (mature)
  - Bottom-up: Starts with experiments and prototypes (rapid)
- From software engineering point of view
  - Waterfall: structured and systematic analysis at each step before proceeding to the next
  - Spiral: rapid generation of increasingly functional systems, short turn around time, quick turn around
- Typical data warehouse design process
  - Choose a business process to model, e.g., orders, invoices, etc.
  - Choose the *grain* (*atomic level of data*) of the business process
  - Choose the dimensions that will apply to each fact table record
  - Choose the measure that will populate each fact table record

# A THREE-TIER DATA WAREHOUSE ARCHITECTURE




**BOTTOM TIER:**

The bottom tier is a warehouse database server that is almost always a relational database system. The data are extracted using application program interfaces known as gateways. Examples of gateways include ODBC JDBC. This tier also contains a metadata repository, which stores information about the data warehouse and its contents.

**MIDDLE TIER:**

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

**TOP TIER:**

The top tier is a front-end client layer, which contains query and reporting tools, analysis tools, and/or data mining tools.

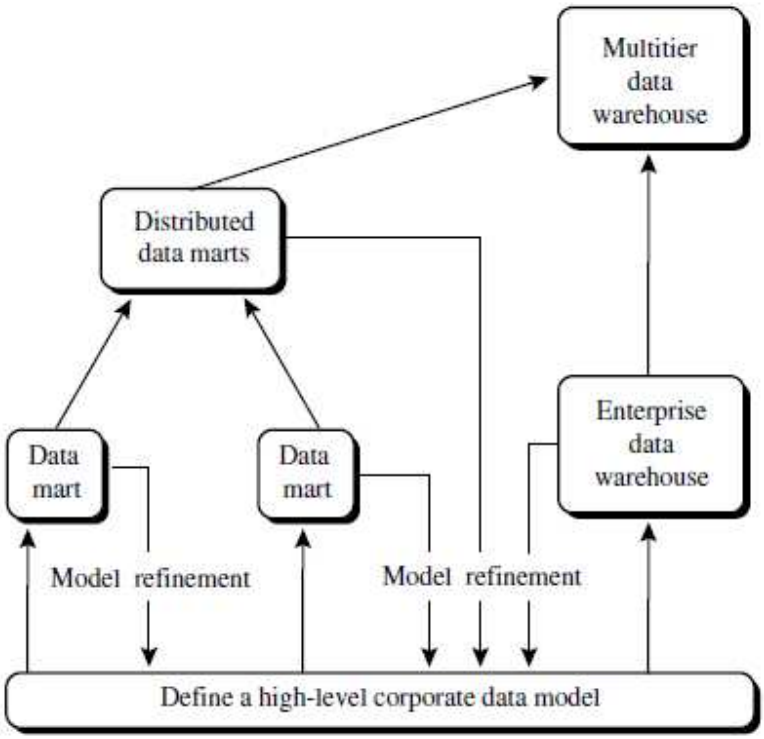
**ENTERPRISE WAREHOUSE**: collects all of the information about subjects spanning the entire organization. It provides corporate-wide data integration, usually from one or more operational systems or external information providers.

**DATA MART:** A subset of corporate-wide data that is of value to specific groups of users. Its scope is confined to specific, selected groups, such as marketing data mart. *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 set of views over operational databases
- Only some of the possible summary views may be materialized.

First, a high-level corporate data model is defined within a reasonably short period Second, independent data marts can be implemented in parallel with the enterprise warehouse based on the same corporate data model set as above. 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.




**DATA WAREHOUSE IMPLEMENTATION**

# Efficient Data Cube Computation

- Data cube can be viewed as a lattice of cuboids
  - The bottom-most cuboid is the base cuboid
  - The top-most cuboid (apex) contains only one cell
  - How many cuboids in an n-dimensional cube with L levels? $T = \prod_{i=1}^{n} (L_i + 1)$
- Materialization of data cube
  - Materialize every (cuboid) (full materialization), none (no materialization), or some (partial materialization)
  - Selection of which cuboids to materialize
    - Based on size, sharing, access frequency, etc.



# Cube Operation

- Cube definition and computation in DMQL

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

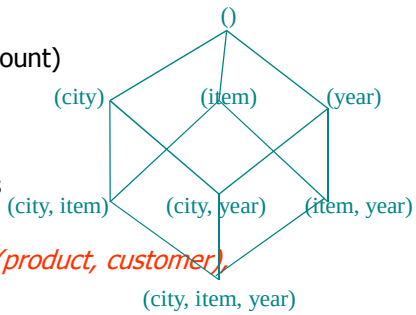
  compute cube sales
- Transform it into a SQL-like language (with a new operator cube by, introduced by Gray et al.'96)

  SELECT item, city, year, SUM (amount)

  FROM SALES

  CUBE BY item, city, year
- Need compute the following Group-Bys

  *(date, product, customer),*
  *(date,product),(date, customer), (product, customer),*
  *(date), (product), (customer)*
  *()*



# Indexing OLAP Data: Bitmap Index

- Index on a particular column
- Each value in the column has a bit vector: bit-op is fast
- The length of the bit vector: # of records in the base table
- The $i$-th bit is set if the $i$-th row of the base table has the value for the indexed column
- not suitable for high cardinality domains

**Base table**

| Cust | Region | Type |
|---|---|---|
| C1 | Asia | Retail |
| C2 | Europe | Dealer |
| C3 | Asia | Dealer |
| C4 | America | Retail |
| C5 | Europe | Dealer |

**Index on Region**

| RecID | Asia | Europe | America |
|---|---|---|---|
| 1 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 |
| 3 | 1 | 0 | 0 |
| 4 | 0 | 0 | 1 |
| 5 | 0 | 1 | 0 |

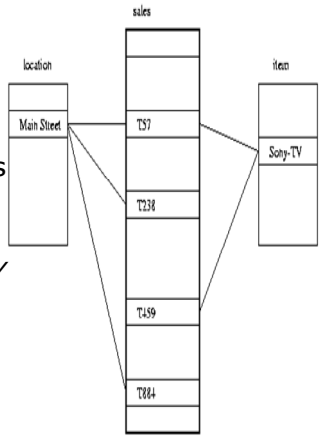
**Index on Type**

| RecID | Retail | Dealer |
|---|---|---|
| 1 | 1 | 0 |
| 2 | 0 | 1 |
| 3 | 0 | 1 |
| 4 | 1 | 0 |
| 5 | 0 | 1 |

January 23, 2012 Data Mining: Concepts and Techniques 51

# Indexing OLAP Data: Join Indices

- Join index: JI(R-id, S-id) where R (R-id, …) ▷◁ S (S-id, …)
- Traditional indices map the values to a list of record ids
  - It materializes relational join in JI file and speeds up relational join
- In data warehouses, join index relates the values of the dimensions of a start schema to rows in the fact table.
  - E.g. fact table: *Sales* and two dimensions *city* and *product*
    - A join index on *city* maintains for each distinct city a list of R-IDs of the tuples recording the Sales in the city
  - Join indices can span multiple dimensions




January 23, 2012 Data Mining: Concepts and Techniques 52

## Efficient Processing OLAP Queries

- Determine which operations should be performed on the available cuboids
  - Transform drill, roll, etc. into corresponding SQL and/or OLAP operations, e.g., dice = selection + projection
- Determine which materialized cuboid(s) should be selected for OLAP op.
  - Let the query to be processed be on {brand, province_or_state} with the condition "year = 2004", and there are 4 materialized cuboids available:

    1) {year, item_name, city}

    2) {year, brand, country}

    3) {year, brand, province_or_state}

    4) {item_name, province_or_state} where year = 2004

    Which should be selected to process the query?
- Explore indexing structures and compressed vs. dense array structs in MOLAP

January 23, 2012 Data Mining: Concepts and Techniques 53

**FURTHER DEVELOPMENT OF DATA CUBE TECHNOLOGY**

## Discovery-driven Exploration of Data Cubes

- Drawbacks of traditional data cubes:
  - Anomaly discovery is manual
  - Use of intuition & Hypothesis
  - High level aggregations mask low level details
  - Sheer volume of data to analyze

## Discovery driven cubes Contd…

- Guide the user in Data Analysis through Exception Indicators
  - pre-computed measures that indicate exceptions in Data
- All dimensions accounted during calculation

*"Exception – in a data cube cell is a significant deviation from anticipated value calculated through statistical measures"*

## Discovery driven cubes Contd…

- Methods to indicate Exceptions in cube cell
  - SelfExp – indicates degree of surprise for a cell value relative to others at the same level.
  - InExp – indicates degree of surprise somewhere beneath the cell
  - PathExp – indicates degree of surprise for each drill-down path from the cell.

*Degree of surprise – defined as deviation from the anticipated value of a date cell*

# Examples: Discovery-Driven Data Cubes

| item | all |
|---|---|
| region | all |

| Sum of sales | month | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Jan | Feb | Mar | Apr | May | Jun | Jul | Aug | Sep | Oct | Nov | Dec |
| Total | | 1% | -1% | 0% | 1% | 3% | -1 | -9% | -1% | 2% | -4% | 3% |

| Avg sales | month | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| item | Jan | Feb | Mar | Apr | May | Jun | Jul | Aug | Sep | Oct | Nov | Dec |
| Sony b/w printer | | 9% | -8% | 2% | -5% | 14% | -4% | 0% | 41% | -13% | -15% | -11% |
| Sony color printer | | 0% | 0% | 3% | 2% | 4% | -10% | -13% | 0% | 4% | -6% | 4% |
| HP b/w printer | | -2% | 1% | 2% | 3% | 8% | 0% | -12% | -9% | 3% | -3% | 6% |
| HP color printer | | 0% | 0% | -2% | 1% | 0% | -1% | -7% | -2% | 1% | -5% | 1% |
| IBM home computer | | 1% | -2% | -1% | -1% | 3% | 3% | -10% | 4% | 1% | -4% | -1% |
| IBM laptop computer | | 0% | 0% | -1% | 3% | 4% | 2% | -10% | -2% | 0% | -9% | 3% |
| Toshiba home computer | | -2% | -5% | 1% | 1% | -1% | 1% | 5% | -3% | -5% | -1% | -1% |
| Toshiba laptop computer | | 1% | 0% | 3% | 0% | -2% | -2% | -5% | 3% | 2% | -1% | 0% |
| Logitech mouse | | 3% | -2% | -1% | 0% | 4% | 6% | -11% | 2% | 1% | -4% | 0% |
| Ergo-way mouse | | 0% | 0% | 2% | 3% | 1% | -2% | -2% | -5% | 0% | -5% | 8% |

| item | IBM home computer | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Avg sales | month | | | | | | | | | | | |
| region | Jan | Feb | Mar | Apr | May | Jun | Jul | Aug | Sep | Oct | Nov | Dec |
| North | | -1% | -3% | -1% | 0% | 3% | 4% | -7% | 1% | 0% | -3% | -3% |
| South | | -1% | 1% | -9% | 6% | -1% | 39% | 9% | -34% | 4% | 1% | 7% |
| East | | -1% | -2% | 2% | -3% | 1% | 18% | -2% | 11% | -3% | -2% | -1% |
| West | | 4% | 0% | -1% | -3% | 5% | 1% | -18% | 8% | 5% | -8% | 1% |

## Complex Aggregation at Multiple Granularities: Multi-Feature Cubes

- Ex. Grouping by all subsets of {item, region, month}, find the maximum price in 1997 for each group, and the total sales among all maximum price tuples

```
select item, region, month, max(price), sum(R.sales)
from purchases
where year = 1997
cube by item, region, month: R
such that R.price = max(price)
```

## Data Warehouse Usage

- Three kinds of data warehouse applications
  - Information processing
    - supports querying, basic statistical analysis, and reporting using crosstabs, tables, charts and graphs
  - Analytical processing
    - multidimensional analysis of data warehouse data
    - supports basic OLAP operations, slice-dice, drilling, pivoting
  - Data mining
    - knowledge discovery from hidden patterns
    - supports associations, constructing analytical models, performing classification and prediction, and presenting the mining results using visualization tools

## From On-Line Analytical Processing (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

January 23, 2012 Data Mining: Concepts and Techniques 61

## ARCHITECTURE OF ON-LINE ANALYTICAL MINING