Introduction to Data Mining

Part 2: Data Cube and OLAP

Spring 2015

Ming Li

Department of Computer Science and Technology
Nanjing University
What is data warehouse? (I)

**Data warehouse** is a subject-oriented, integrated, time-variant, and nonvolatile collection of data in support of management’s decision making process

W.H. Inmon’s definition (1996)

数据仓库是支持管理决策过程的面向主题的、集成的、长期的、稳定的数据集合

**Data warehousing**: the process of constructing and using data warehouse
What is data warehouse? (II)

- **Why subject-oriented?**
  A data warehouse is organized around major subjects

- **Why integrated?**
  A data warehouse is usually constructed by integrating multiple heterogeneous data sources

- **Why time-variant?**
  Data are stored to provide information from a historical perspective (5-10 years in data warehouse vs. 30-90 days in operational databases)

- **Why nonvolatile?**
  Update of data does not occur frequently in a data warehouse (without transaction processing, recovery and concurrency control mechanisms)
Data warehouse vs. Operational DBMS

• Operational DBMS
  – On-line transaction processing (OLTP)
    Day-today operations such as purchasing, banking, payroll, etc.

• Data warehouse
  – On-line analytical processing (OLAP)
    Data analysis and decision making
# OLTP vs. OLAP

<table>
<thead>
<tr>
<th></th>
<th>OLTP</th>
<th>OLAP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>users</strong></td>
<td>clerks, IT professionals</td>
<td>managers, executives</td>
</tr>
<tr>
<td><strong>function</strong></td>
<td>day to day operations</td>
<td>decision support</td>
</tr>
<tr>
<td><strong>DB design</strong></td>
<td>application-oriented</td>
<td>subject-oriented</td>
</tr>
<tr>
<td><strong>data</strong></td>
<td>current, up-to-date detailed, flat relational isolated</td>
<td>historical, summarized, multidimensional integrated, consolidated</td>
</tr>
<tr>
<td><strong>access</strong></td>
<td>read/write</td>
<td>mostly read</td>
</tr>
<tr>
<td><strong>unit of work</strong></td>
<td>short, simple transaction</td>
<td>complex query</td>
</tr>
<tr>
<td><strong>number of</strong></td>
<td>tens</td>
<td>millions</td>
</tr>
<tr>
<td><strong>accessed records</strong></td>
<td>100MB-GB</td>
<td>100GB-TB</td>
</tr>
<tr>
<td><strong>number of users</strong></td>
<td>thousands</td>
<td>hundreds</td>
</tr>
<tr>
<td><strong>DB size</strong></td>
<td>transaction throughput</td>
<td>query throughput, response time</td>
</tr>
</tbody>
</table>
Why separate data warehouse?

• **High performance of both systems**
  - Data warehouse in tuned for OLAP
  - Operational DBMS is tuned for OLTP
    - Processing OLAP queries in operational databases would substantially degrade the performance of operational tasks
    - Concurrency control and recovery mechanisms, if applied for OLAP operations, may jeopardize the execution of concurrent transactions and thus substantially reduce the throughput of an OLTP system

• **Different structures, contents, and uses of data**
  - Data warehouse contains *historical, consolidated data*
  - Operational DBMS contains *detailed raw data*
Multi-tiered data warehouse architecture

Operational database

External sources

Metadata repository

Monitoring

Administration

OLAP Server

Data Warehouse

Data Marts

OLAP Server

Query/report

Analysis

Data mining

Front-end tools

Introduction to Data Mining: Part 2
What is multidimensional data model?

**Multidimensional data model** is a data model that organizes data from the view of multiple dimensions.

**Dimensions** are the perspectives or entities with respect to which an organization wants to keep records.

We sell products in various markets, and we measure our performance over time.

We sell **Products** in various **Markets**, and we measure our performance over **Time**.

**Data cube is the core of multidimensional data model**.
Cube representation: An example (I)

<table>
<thead>
<tr>
<th>time</th>
<th>Location=&quot;Vancouver&quot;</th>
<th></th>
<th>Location=&quot;Montreal&quot;</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Item</td>
<td></td>
<td>Item</td>
<td></td>
</tr>
<tr>
<td></td>
<td>home ent.</td>
<td>comp.</td>
<td>phone</td>
<td>sec.</td>
</tr>
<tr>
<td>Q1</td>
<td>605K</td>
<td>825K</td>
<td>14K</td>
<td>400K</td>
</tr>
<tr>
<td>Q2</td>
<td>680K</td>
<td>952K</td>
<td>31K</td>
<td>512K</td>
</tr>
<tr>
<td>Q3</td>
<td>812K</td>
<td>1023K</td>
<td>30K</td>
<td>501K</td>
</tr>
<tr>
<td>Q4</td>
<td>927K</td>
<td>1038K</td>
<td>38K</td>
<td>580K</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>time</th>
<th>Location=&quot;New York&quot;</th>
<th></th>
<th>Location=&quot;Chicago&quot;</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Item</td>
<td></td>
<td>Item</td>
<td></td>
</tr>
<tr>
<td></td>
<td>home ent.</td>
<td>comp.</td>
<td>phone</td>
<td>sec.</td>
</tr>
<tr>
<td>Q1</td>
<td>1087K</td>
<td>968K</td>
<td>38K</td>
<td>872K</td>
</tr>
<tr>
<td>Q2</td>
<td>1130K</td>
<td>1024K</td>
<td>41K</td>
<td>925K</td>
</tr>
<tr>
<td>Q3</td>
<td>1034K</td>
<td>1047K</td>
<td>45K</td>
<td>1002K</td>
</tr>
<tr>
<td>Q4</td>
<td>1142K</td>
<td>1091K</td>
<td>54K</td>
<td>984K</td>
</tr>
</tbody>
</table>
Cube representation: An example (II)
Cube representation: An example (III)

$n$-D data can be represented as a series of $(n-1)$-D “cubes”
What is data cube?

**Data Cube**: The lattice of cuboids

- 0-D (apex) cuboid
- 1-D cuboids
- 2-D cuboids
- 3-D cuboids
- 4-D (base) cuboid

Data cube is not a paradigm of actual physical storage
Multidimensional database schemas

- The multidimensional data model for data warehouse can exist in the form of:
  - Star schema
  - Snowflake schema
  - Fact constellation schema

- In those schemas:
  - Each dimension is represented by a dimension table that describes the dimension
  - Numerical measures are called facts. A fact table contains the facts as well as keys to each of the related dimension tables
Star schema

- A fact table in the middle connected to a set of dimension tables

**Time**
- time_key
- day
- day_of_the_week
- month
- quarter
- year

**Branch**
- branch_key
- branch_name
- branch_type

**Location**
- location_key
- street
- city
- province_or_state
- country

**Item**
- item_key
- item_name
- brand
- type
- supplier_type
- supplier_name

**Sales Fact**
- time_key
- item_key
- branch_key
- location_key
- dollars_sold
- units_sold

**Measures**
A compromise between star schema and snowflake schema is to adopt a mixed schema where only the very large dimension tables are normalized.
Fact constellation schema

also called *galaxy* schema

Multiple fact tables share some dimension tables

**Sales Fact**
- `time_key`
- `item_key`
- `branch_key`
- `location_key`
- `dollars_sold`
- `units_sold`

**Time**
- `time_key`
- `day`
- `day_of_the_week`
- `month`
- `quarter`
- `year`

**Branch**
- `branch_key`
- `branch_name`
- `branch_type`

**Location**
- `location_key`
- `province_or_state`
- `country`

**Measures**

**Shipping Fact**
- `time_key`
- `item_key`
- `shipper_key`
- `from_location`
- `to_location`
- `dollars_cost`
- `units_shipped`

**Shipper**
- `shipper_key`
- `shipper_name`
- `location_key`
- `shipper_type`
What is concept hierarchy?

A concept hierarchy defines a sequence of mappings from a set of low level concepts to higher level, more general concepts.
**Specification of hierarchies (I)**

**Schema hierarchy:**

A concept hierarchy that is a total or partial order among attributes in a databases schema

- **Total order:**
  - Continent
  - Country
  - City
  - Street
  - street < city < country < continent

- **Partial order:**
  - Year
  - Quarter
  - Month
  - Week
  - Day
  - day < \{month < quarter; week\} < year

A concept hierarchy may involve a single attribute or several attributes
Set-grouping hierarchy:
A concept hierarchy that is defined by discretizing or grouping values for a given dimension or attribute

There may be more than one concept hierarchy for a given dimension, based on different user viewpoints
OLAP operations on data cube (I)

- **roll-up**: performs aggregation on a data cube, either by climbing-up a concept hierarchy for a dimension or by dimension removal.

---

**Example**

- **Roll-up on location** (from cities to countries)
OLAP operations on data cube (II)

- **drill-down**: navigates a data cube from less detailed data to more detailed data, either by stepping-down a concept hierarchy for a dimension or by introducing additional dimensions.

**Example**:

- **Drill-down on time** (from quarters to months): The data cube transitions from quarterly to monthly aggregation.
OLAP operations on data cube (IV)

- **dice**: performs a selection on multiple dimensions of a data cube

  for (location = “Toronto” or “Vancouver”) and (time = “Q1” or “Q2”) and (item = “home entertainment” or “computer”)
OLAP operations on data cube (III)

- **slice**: performs a selection on one dimension of a data cube

  slice
  
  for time “Q1”
OLAP operations on data cube (V)

- **pivot**: rotates the data axes in view (also called rotate)

![Diagram showing pivot operation on a data cube]

**location (cities)**
- Chicago
- New York
- Toronto
- Vancouver

**item (types)**
- home
- entertainment
- computer
- phone
- security

<table>
<thead>
<tr>
<th></th>
<th>home</th>
<th>entertainment</th>
<th>computer</th>
<th>phone</th>
<th>security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chicago</td>
<td>605</td>
<td>825</td>
<td>14</td>
<td>14</td>
<td>400</td>
</tr>
<tr>
<td>New York</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Toronto</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Vancouver</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The pivot operation rotates the data axes, changing the view of the data cube.
A measure is a numerical function that can be evaluated at each point in the data space.

Distributive aggregate function

- Suppose the data is partitioned into $n$ sets.
- The computation of the function on each partition results in one aggregate value.
- If the result derived by applying the function to $n$ aggregate values is as same as that derived by applying the function on all the data without partitioning.

E.g., `count()`, `sum()`, `min()`, `max()`
Three categories of measures:

- **Distributive**: if it is obtained by applying a distributive aggregate function
  
  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()`, `max_N()`, `standard_deviation()`

- **Holistic**: if there is no constant bound on the storage size needed to describe a subaggregate
  
  e.g., `rank()`, `median()`, `mode()`
OLAP server architectures

- **ROLAP (Relational OLAP)**
  - Use relational or extended-relational DBMS to store and manage warehouse data
  - Good scalability

- **MOLAP (Multidimensional OLAP)**
  - Array-based multidimensional storage engine (sparse matrix techniques)
  - Fast indexing to pre-computed summarized data

- **HOLAP (Hybrid OLAP)**
  - Good scalability of ROLAP + fast computation of MOLAP

- **Specialized SQL servers**
  - Specialized support for SQL queries over star/snowflakes schemas
Why precomputation?

- **OLAP engines demand that decision support queries be answered in the order of seconds**
- **Precomputation of a data cube leads to fast response time and avoids some redundant computation**

- **How many cuboids are there in an n-D data cube (lattice)?**
  - If there is no hierarchy
    \[ T = 2^n \]
  - If each dimension is associated with \( L_i \) levels of hierarchy
    \[ T = \prod_{i=1}^{n} (L_i + 1) \]

For 10 dimensions each with 4 Levels, \( T = 5^{10} \approx 9.8 \times 10^6 \)
What’s the best way of precomputation?

- **no materialization** precompute only the base cuboid
  - *slow response*
- **full materialization** precompute all the cuboids
  - *huge storage*
- **Partial materialization** precompute some of the cuboids
  - *trade-off*

Selection of the cuboids to be materialized based on access frequency, sharing, updating, etc.

A popular approach: materialize the set of cuboids on which other popularly referenced cuboids are based
Efficient cube computation

• Since ROLAP and MOLAP employ different data structures (*tuples and relational tables* vs. *multidimensional array*), they utilize different cube computation techniques

ROLAP-based cube computation
  based on some *optimization techniques*
  relatively slow
  *(instead of cubing a table directly, it is even faster to convert the table to an array, cube the array, then convert the result back to a table)*

MOLAP-based cube computation
  based on *multiway array aggregation*
  relatively fast
ROLAP-based cube computation

- **Smallest-child**: computing a cuboid from the smallest child cuboid of previously computed ones
- **Cache-results**: caching results of a cuboid from which other cuboids are computed
- **Sharedsorts**: sharing sorting costs across multiple cuboids
- ....

Popular optimization techniques:
- sorting, hashing, and grouping operations are applied to the dimension attributes in order to record and cluster related tuples
- grouping is performed on some subaggregates as a “partial grouping step”
- aggregates may be computed from previously computed aggregates, rather than from the base fact table
Multiway array aggregation (I)

- Partition arrays into **chunks** (chunk is a small subcube which fits in memory)
- Compressed sparse array addressing: (chunk_id + offset)
- Compute aggregates in “multiway” by visiting cube cells in the order which minimizes the number of times that each cell must be revisited, thereby reducing memory access and storage cost

Suppose the size of the array for each dimension, A, B, and C, is 40, 400, and 4000

What is the best traversing order to do multiway aggregation?
Multiway array aggregation (II)

The minimal memory required for holding all relevant 2-D planes in chunk memory:

For BC plane:

\[ 100 \times 1,000 = 100,000 \]
**Multiway array aggregation (II)**

The minimal memory required for holding all relevant 2-D planes in chunk memory:

For BC plane:
\[ 100 \times 1,000 = 100,000 \]

For AC plane:
\[ 40 \times 1,000 = 40,000 \]
Multiway array aggregation (II)

The minimal memory required for holding all relevant 2-D planes in chunk memory:

For BC plane:
\[ 100 \times 1,000 = 100,000 \]

For AC plane:
\[ 40 \times 1,000 = 40,000 \]

For AB plane:
\[ 40 \times 400 = 16,000 \]

Total: 156,000
Multiway array aggregation (III)

The minimal memory required for holding all relevant 2-D planes in chunk memory:

For AB plane: \[ 10 \times 100 = 1,000 \]
The minimal memory required for holding all relevant 2-D planes in chunk memory:

For AB plane:
\[10 \times 100 = 1,000\]

For AC plane:
\[10 \times 4,000 = 40,000\]
Multiway array aggregation (III)

The minimal memory required for holding all relevant 2-D planes in chunk memory:

For AB plane:
\[ 10 \times 100 = 1,000 \]

For AC plane:
\[ 10 \times 4,000 = 40,000 \]

For BC plane:
\[ 400 \times 4,000 = 1,600,000 \]

Total: 1,641,000

The dimensions should be sorted according to their sizes \textit{in ascending order}.
Indexing OLAP data: Bitmap Index

- Index on a particular column
- Each value in the column has a bit vector
- The length of the bit vector is the number of the possible values of this column
- The $i$-th bit is set to 1 if the $i$-th possible value appears

Fact Table

<table>
<thead>
<tr>
<th>Cust</th>
<th>Region</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>Asia</td>
<td>Retail</td>
</tr>
<tr>
<td>C2</td>
<td>Europe</td>
<td>Dealer</td>
</tr>
<tr>
<td>C3</td>
<td>Asia</td>
<td>Dealer</td>
</tr>
<tr>
<td>C4</td>
<td>America</td>
<td>Retail</td>
</tr>
<tr>
<td>C5</td>
<td>Europe</td>
<td>Dealer</td>
</tr>
</tbody>
</table>

Index on Region

<table>
<thead>
<tr>
<th>RecID</th>
<th>Asia</th>
<th>Europe</th>
<th>America</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Index on Type

<table>
<thead>
<tr>
<th>RecID</th>
<th>Retail</th>
<th>Dealer</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Advantages:**
- Comparison, join, and aggregation operations are reduced to bit-arithmetic operations
- Significant reduction in space and I/O since a string of characters can be represented by a single bit
Indexing OLAP data: Join Index

**register the joinable rows of two relations**

\[ \text{JI}(T\text{-id, S\text{-id}) where } T(T\text{-id, …}) \bowtie S(S\text{-id, …}) \]

- In star schema model of data warehouses, join index relates the values of the dimensions to rows in the fact table.

  e.g., a fact table sales and two dimensions location and item. A join index on location maintains for each distinct street (e.g., “main street”) a list of T-ids of the tuples [T57, T238, T459, T884] recording the sales in the street

- **Join indices can span multiple dimensions**
Metadata are data about data

In data warehouse, metadata are the data that define warehouse objects

- **Categories of data warehouse metadata:**
  - Information on potential data source
  - Information on data model
  - Information on the mapping between transactional data structure and warehouse data structure
  - Information on the usage of warehouse contents (so that the performance of warehouse can be tuned)
What is the difference between OLAP and data mining?

- **Target**
  - OLAP are targeted toward simplifying and supporting *interactive* data analysis
  - Data mining were targeted toward *automatic* data analysis

- **Function**
  - OLAP functions are essentially for *user-driven data summary and comparison*
  - Data mining covers a much broader spectrum of functions

- **Data**
  - OLAP is used to analyze data *in data warehouse*
  - Data mining is not limited to analyze data in data warehouse
Why integrating OLAP with data mining?

- **High quality of data in data warehouses**
  - DW contains integrated, consistent, cleaned data

- **Available information processing infrastructure surrounding data warehouses**
  - ODBC, OLEDB, Web accessing, service facilities, reporting and OLAP tools

- **OLAP-based exploratory data analysis**
  - Select portions of relevant data, analyze at different granularities, visualization tools

- **On-line selection of data mining functions**
  - Flexibility to select data mining functions and swap data mining tasks dynamically
An OLAM architecture
AOI for concept description

• **Attribute-Oriented Induction**
  
  [Y. Cai, N. Cercone, and J. Han, KDD Workshop at IJCAI’89]

• **Can be used for both characterization and Comparison**
  
  – **Characterization**: provide a concise and succinct summarization of the given collection of data
  
  – **Comparison (Discrimination)**: provide descriptions comparing two or more collections of data

• **The simplest kind of descriptive data mining; can also be regarded as extended OLAP**

  in its initial proposal, AOI is a relational database query-oriented, generalization-based, online data analysis technique; now data cube and offline precomputation can also be used
Attribute-oriented induction

General idea:

- Collect the task-relevant data
- Perform generalization by attribute removal or attribute generalization
- Apply aggregation by merging identical, generalized tuples and accumulating their respective counts

The Key: Data generalization

*a process which abstracts a large set of task-relevant data in a database from a relatively low conceptual level to higher conceptual levels*
Sketch of AOI

- **Data focusing**
  
  the specification of task-relevant data, resulting in the initial working relation

- **Data generalization**
  
  - **attribute removal**
    
    if there is a large set of distinct values for an attribute, but either (1) there is no generalization operator on the attribute, or (2) its higher level concepts are expressed in terms of other attributes
  
  - **attribute generalization**
    
    if there is a large set of distinct values for an attribute, and there exists a set of generalization operators on the attribute

- **Presentation**
How to control generalization?

The control of how high an attribute should be generalized is quite subjective. The control of this process is called attribute generalization control.

- **attribute generalization threshold control**
  - if the number of distinct values in an attribute is greater than the threshold, then further attribute removal or generalization should be performed
    - either set a generalization threshold for all the attributes, or set one threshold for each attribute
    - typically ranging from 2 to 8

- **generalized relation threshold control**
  - if the number of distinct tuples in a generalized relation is greater than the threshold, then further generalization should be performed
    - set a threshold for the size of the generalized relation
    - Typically ranging from 10 to 30
Example of AOI

- **Initial relation**

<table>
<thead>
<tr>
<th>name</th>
<th>gender</th>
<th>major</th>
<th>birth-place</th>
<th>birth-date</th>
<th>residence</th>
<th>phone#</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jim Woodman</td>
<td>M</td>
<td>CS</td>
<td>Vancouver, BC, Canada</td>
<td>8-12-93</td>
<td>3511 Main St., Richmond</td>
<td>687-4598</td>
<td>3.67</td>
</tr>
<tr>
<td>Scott Lachance</td>
<td>M</td>
<td>CS</td>
<td>Montreal, Que, Canada</td>
<td>28-7-94</td>
<td>345 1st Ave., Richmond</td>
<td>253-9106</td>
<td>3.70</td>
</tr>
<tr>
<td>Laura Lee</td>
<td>F</td>
<td>Physics</td>
<td>Seattle, WA, USA</td>
<td>25-8-87</td>
<td>125 Austin Ave., Burnaby</td>
<td>420-5232</td>
<td>3.83</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>removed</td>
<td>retianed</td>
<td>Sci, Eng, Business</td>
<td>country</td>
<td>age_range</td>
<td>City</td>
<td>removed</td>
<td>Excellent ...</td>
</tr>
</tbody>
</table>

- **Prime generalized relation**

<table>
<thead>
<tr>
<th>gender</th>
<th>major</th>
<th>birth_region</th>
<th>age_range</th>
<th>residence</th>
<th>gpa</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>M</td>
<td>Science</td>
<td>Canada</td>
<td>20-25</td>
<td>Richmond</td>
<td>Very-good</td>
<td>16</td>
</tr>
<tr>
<td>F</td>
<td>Science</td>
<td>USA</td>
<td>25-30</td>
<td>Burnaby</td>
<td>Excellent</td>
<td>22</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
**Presentation of characterization (I)**

- Generalized relation

**sales in 2014**

<table>
<thead>
<tr>
<th>location</th>
<th>item</th>
<th>sales (in million dollars)</th>
<th>Count (in thousands)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asia</td>
<td>TV</td>
<td>15</td>
<td>300</td>
</tr>
<tr>
<td>Europe</td>
<td>TV</td>
<td>12</td>
<td>250</td>
</tr>
<tr>
<td>North_America</td>
<td>TV</td>
<td>28</td>
<td>450</td>
</tr>
<tr>
<td>Asia</td>
<td>computer</td>
<td>120</td>
<td>1000</td>
</tr>
<tr>
<td>Europe</td>
<td>computer</td>
<td>150</td>
<td>1200</td>
</tr>
<tr>
<td>North_America</td>
<td>computer</td>
<td>200</td>
<td>1800</td>
</tr>
</tbody>
</table>
**Presentation of characterization (II)**

- **Crosstab**

  sales in 2014

<table>
<thead>
<tr>
<th>location/item</th>
<th>TV</th>
<th>computer</th>
<th>both-items</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>sales</td>
<td>count</td>
<td>sales</td>
</tr>
<tr>
<td>Asia</td>
<td>15</td>
<td>300</td>
<td>120</td>
</tr>
<tr>
<td>Europe</td>
<td>12</td>
<td>250</td>
<td>150</td>
</tr>
<tr>
<td>North_America</td>
<td>28</td>
<td>450</td>
<td>200</td>
</tr>
<tr>
<td>All Regions</td>
<td>45</td>
<td>1000</td>
<td>470</td>
</tr>
</tbody>
</table>
Presentation of characterization (III)

- Bar chart

Sales in 2014
Presentation of characterization (IV)

- Pie chart

Sales in 2014

- Computer Sales
- TV Sales
- TV + Computer Sales
Presentation of characterization (V)

- 3-D cube

sales in 2014
Presentation of characterization (VI)

- Quantitative characteristic rule

  A logic rule that is associated with quantitative information is called a quantitative rule

\[
\forall X, \text{item}(X) = \text{“computer”} \Rightarrow \\
(\text{location}(X) = \text{“Asia”}) [t : 25.00\%] \lor (\text{location}(X) = \text{“Europe”}) [t : 30.00\%] \lor \\
(\text{location}(X) = \text{“North_America”}) [t : 45.00\%]
\]

The general form of a quantitative characteristic rule is:

\[
\forall X, \text{target\_class}(X) \Rightarrow \text{condition}_1(X)[t : w_1] \lor \cdots \lor \text{condition}_n(X)[t : w_n]
\]

Where \(t\text{-weight}\) describes the typicality of each disjunct in the rule

\[
t\_weight = \frac{\text{count}(q_a)}{\sum_{i=1}^{N} \text{count}(q_i)}
\]

Characteristic rule is necessary condition of the target class
Mining class comparisons

• Data generalization (including attribute removal and attribute generalization) should be performed synchronously among all the classes.

e.g., comparing sales in China on Nov. 9 with sales in USA in the year 2013 is usually meaningless.

• However, the user can over-write such a synchronous comparison with his/her own choices.

e.g., user may want to compare sales in Shanghai with sales in Malaysia.
Example of mining comparisons

Prime generalized relation for the target class: Graduate students

<table>
<thead>
<tr>
<th>birth_country</th>
<th>age_range</th>
<th>gpa</th>
<th>count %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canada</td>
<td>20-25</td>
<td>Good</td>
<td>5.53</td>
</tr>
<tr>
<td>Canada</td>
<td>25-30</td>
<td>Good</td>
<td>2.32</td>
</tr>
<tr>
<td>Canada</td>
<td>Over_30</td>
<td>Very_good</td>
<td>5.86</td>
</tr>
<tr>
<td>Other</td>
<td>Over_30</td>
<td>Excellent</td>
<td>4.68</td>
</tr>
</tbody>
</table>

Prime generalized relation for the target class: Undergraduate students

<table>
<thead>
<tr>
<th>birth_country</th>
<th>age_range</th>
<th>gpa</th>
<th>count %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Canada</td>
<td>15-20</td>
<td>Fair</td>
<td>5.53</td>
</tr>
<tr>
<td>Canada</td>
<td>15-20</td>
<td>Good</td>
<td>4.53</td>
</tr>
<tr>
<td>Other</td>
<td>Over_30</td>
<td>Excellent</td>
<td>0.68</td>
</tr>
</tbody>
</table>
Quantitative discriminant rule

<table>
<thead>
<tr>
<th>status</th>
<th>birth_country</th>
<th>age_range</th>
<th>gpa</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graduate</td>
<td>Canada</td>
<td>25-30</td>
<td>Good</td>
<td>90</td>
</tr>
<tr>
<td>Undergraduate</td>
<td>Canada</td>
<td>25-30</td>
<td>Good</td>
<td>210</td>
</tr>
</tbody>
</table>

\[
\forall X, \text{graduate\_student}(X) \iff \\
\text{birth\_country}(X) = "Canada" \land \text{age\_range}(X) = "25-30" \land \text{gpa}(X) = "good" [d : 30%]
\]

The general form of a quantitative discriminant rule is:

\[
\forall X, \text{target\_class}(X) \iff \text{condition}_1(X)[d : w_1] \lor \cdots \lor \text{condition}_n(X)[d : w_n]
\]

where \textit{d-weight} describes the discriminability of each disjunct in the rule

\[
d\_weight = \frac{\text{count}(q_a \in C_{\text{target}})}{\sum_{i=1}^{N} \text{count}(q_a \in C_i)}
\]

Discriminant rule is sufficient condition of the target class
Characteristic vs. discriminant rule

• Characteristic rule
  – necessary condition of the target class
  – $t_{weight_1} + \ldots + t_{weight_n} = 100\%$
  – Semantics of the rule without $t$-weight is fine

• Discriminant rule
  – Sufficient condition of the target class
  – $condport_1 * d_{weight_1} + \ldots + condport_n * d_{weight_n} = tclass_port$
    where $condport_i$ is the portion of the tuples covered by the $i$-th antecedents of the rule, and $tclass_port$ is the portion of the tuples belonging to the target class
  – Semantics of the rule without $d$-weight is not fine
    • Except that the antecedents of the rule do not cover any tuples of contrasting classes; however, in this case there is no contrasting class and thus, the task degenerates to characterization
Quantitative description rule

∀X, target_class(X) ⇔

\[ condition_1(X)[ t: w_1, d: w'_1 ] \lor \cdots \lor condition_n(X)[ t: w_n, d: w'_n ] \]

quantitative description rule can also be expressed as a crosstab example:

<table>
<thead>
<tr>
<th>location/item</th>
<th>TV</th>
<th>computer</th>
<th>both-items</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>count</td>
<td>t-weight</td>
<td>d-weight</td>
</tr>
<tr>
<td>Europe</td>
<td>80</td>
<td>240</td>
<td>320</td>
</tr>
<tr>
<td>North_America</td>
<td>120</td>
<td>560</td>
<td>680</td>
</tr>
<tr>
<td>both_regions</td>
<td>200</td>
<td>800</td>
<td>1000</td>
</tr>
</tbody>
</table>

∀X, Europe(X) ⇔

\[ (item(X) = "TV") \lor \cdots \lor (item(X) = "computer") \]
Quantitative description rule

∀X, target_class(X) ⇐

condition₁(X)[t : w₁, d : w'₁] ∨⋯∨ conditionₙ(X)[t : wₙ, d : w'ₙ]

quantitative description rule can also be expressed as a crosstab example:

<table>
<thead>
<tr>
<th>location/item</th>
<th>TV</th>
<th>computer</th>
<th>both-items</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>count</td>
<td>t-weight</td>
<td>d-weight</td>
</tr>
<tr>
<td>Europe</td>
<td>80</td>
<td>25%</td>
<td>40%</td>
</tr>
<tr>
<td>North_America</td>
<td>120</td>
<td>17.65%</td>
<td>60%</td>
</tr>
<tr>
<td>both_regions</td>
<td>200</td>
<td>20%</td>
<td>100%</td>
</tr>
</tbody>
</table>

∀X, Europe(X) ⇐

(item(X) = "TV")[t : 25%, d : 40%] ∨⋯∨ (item(X) = "computer")[t : 75%, d : 30%]
Incremental and parallel mining of concept description

Given the huge amounts of data in a database, it is highly preferred to update data mining results incrementally rather than mining from scratch on each database updates.

• **Incremental mining**
  - revision based on newly added data $\Delta DB$
  - generalize $\Delta DB$ to the same level of abstraction in the generalized relation $R$ to derive $\Delta R$
  - union $R \cup \Delta R$, i.e. merge counts and other statistical information to produce a new relation $R'$

such an idea can also be applied to parallel or distributed mining
Let's move to Part 3