Scientific Data Analysis: Apriori and Data Warehouses

Jarek Szlichta
http://data.science.uoit.ca/

Acknowledgments: Jiawei Han, Micheline Kamber and Jian Pei, Data Mining - Concepts and Techniques

Apriori Technique

- Frequent Itemset Mining Methods
  - Apriori
- Which Patterns Are Interesting?
- Summary

What Is Frequent Pattern Analysis?

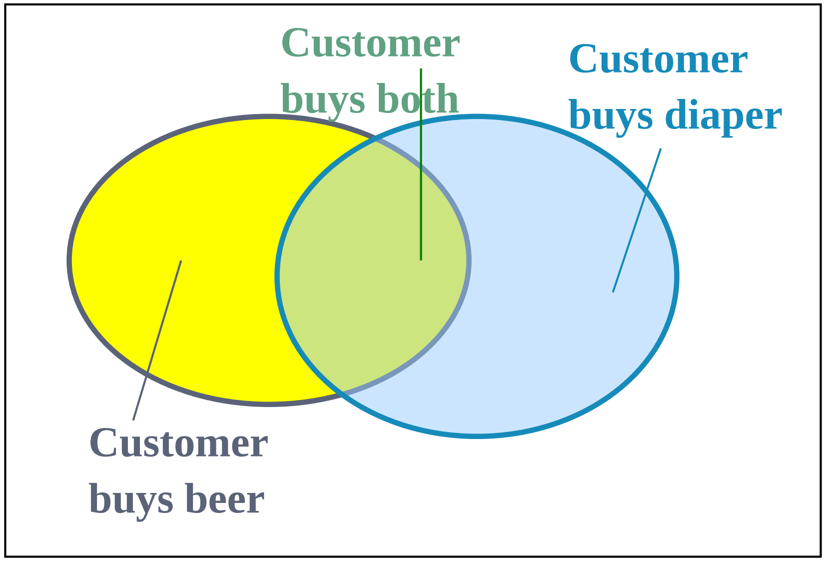
- Frequent pattern: a pattern (a set of items) that occurs frequently in a data set
- Motivation: Finding inherent regularities in data
  - What products were often purchased together?— Beer and diapers?!
  - What kinds of DNA are sensitive to this new drug?
  - Can we automatically classify web documents?
- Applications
  - Basket data analysis, cross-marketing, catalog design, sale campaign analysis, Web log (click stream) analysis, and DNA sequence analysis.

Why Is Frequent Pattern Mining Important?

- Foundation for many essential data mining tasks
  - Association and correlation analysis
  - Structural (e.g., sub-graph) patterns
  - Time-series, and stream data
  - Cluster analysis: frequent pattern-based clustering
  - Data warehousing

Basic Concepts: Frequent Patterns

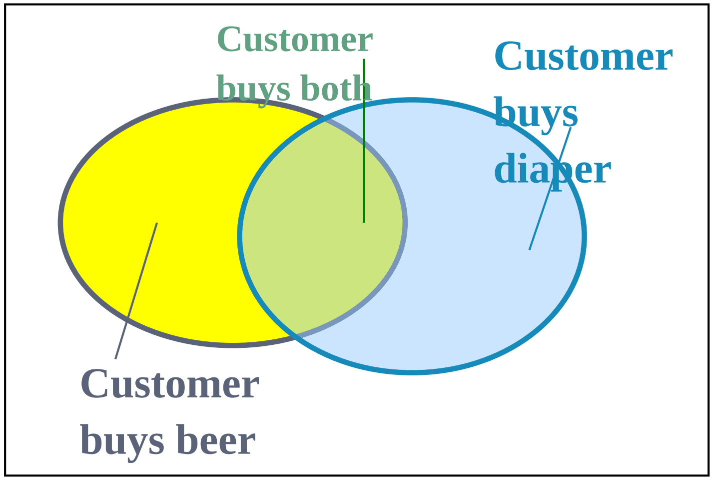
Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



- itemset: A set of one or more items
- k-itemset $X = \{x_1, \ldots, x_k\}$
  - e.g., 2-itemset {Beer, Diaper}
- *(absolute) support*, or, *support count* of X: Frequency or occurrence of an itemset X
  - e.g., absolute support for {Beer, Diaper} equals 3
- *(relative) support*, *s*, is the fraction of transactions that contains X (i.e., the probability that a transaction contains X)
  - e.g., relative support for {Beer, Diaper} equals 3/5
- An itemset X is *frequent* if X's support is no less than a *minsup* threshold
  - e.g., assume minsup = 2 (40%). Then, for instance, {Beer, Diaper} (support 3), {Milk, Eggs, Nuts} (support 2) are frequent itemset patterns.

Basic Concepts: Association Rules

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk



- Find all the rules $X \rightarrow Y$ with minimum support and confidence
  - support, *s*, probability that a transaction contains $X \cup Y$
  - confidence, *c*, conditional probability that a transaction having X also contains *Y*

Let minsup = 50%, minconf = 50%

Freq. Pat.: Beer:3, Nuts:3, Diaper:4, Eggs:3, {Beer, Diaper}:3

- Association rules: (many more!)
  - *Beer* → *Diaper* (3/5 = 60%, 3/3 = 100%)
  - *Diaper* → *Beer* (3/5= 60%, 3/4 = 75%)

Computational Complexity

- How many itemsets are potentially to be generated in the worst case?
  - The number of frequent itemsets to be generated is sensitive to the minsup threshold
  - When minsup is low, there exist potentially an exponential number of frequent itemsets
  - The worst case: M^N where M: # distinct items, and N: max length of transactions

The Downward Closure Property

- The downward closure property of frequent patterns
  - Any subset of a frequent itemset must be frequent
  - If **{milk, eggs, nuts}** is frequent, so is **{milk, eggs}**
  - i.e., every transaction having {milk, eggs, nuts} also contains {milk, eggs}

Apriori: A Candidate Generation & Test Approach

- Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested!
- Method:
  - Initially, scan DB once to get frequent 1-itemset
  - Generate length (k+1) candidate itemsets from length k and length 1 frequent itemsets
  - Test the candidates against DB
  - Terminate when no frequent or candidate set can be generated

The Apriori Algorithm—An Example

$Sup_{min} = 2$

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

1st scan → C_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

→ L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2nd scan → C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

→ L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3

Itemset
{A, B, C}
{A, C, E}
{A, B, E}
{B, C, E}

3rd scan → L_3

Itemset	sup
{B, C, E}	2

The Apriori Algorithm (Pseudo-Code)

C_k: Candidate itemset of size k

L_k : frequent itemset of size k

L_1 = {frequent items};

for (k = 1; L_k !=$\varnothing$; k++) **do begin**

C_{k+1} = candidates generated from L_k *and* L_1;

for each transaction t in database do

increment the count of all candidates in C_{k+1} that are contained in t

L_{k+1} = candidates in C_{k+1} with min_support

end

return $\cup_k L_k$;

Implementation of Apriori

- How to generate candidates?
  - Step 1: joining L_k and L_1
  - Pruning
- Example of Candidate-generation
  - L_3={*abc, abd, acd, ace, bcd*}
  - Joining: $L_3 * L_1$
    - *e.g., abce etc.*
  - Pruning:
    - Assume d is not in L_1 but e is
    - *abcd* is not a candiate because *d* is not in L_1
  - C_4 = {*abce, ...*}

Frequent Pairs in SQL

- Baskets(TID, item)

```
SELECT b1.item, b2.item
FROM Baskets b1, Baskets b2
WHERE b1.TID = b2.TID
  AND b1.item < b2.item
GROUP BY b1.item, b2.item
HAVING COUNT(*) >= s;
```

Look for two Basket tuples with the same TID and different items. First item must precede second, so we don't count the same pair twice.

Throw away pairs of items that do not appear at least *s* times.

Create a group for each pair of items that appears in at least one basket.

A-Priori Trick – (1)

- Straightforward implementation involves a join of a huge Baskets relation with itself.
- The *a-priori algorithm* speeds the query by recognizing that a pair of items $\{i, j\}$ cannot have support s unless both $\{i\}$ and $\{j\}$ do.

A-Priori Trick – (2)

- Use a materialized view (table) to hold only information about frequent items.

```
INSERT INTO Baskets1(TID, item)
SELECT * FROM Baskets
WHERE item IN (
SELECT item FROM Baskets
GROUP BY item
HAVING COUNT(*) >= s
);
```

Items that appear in at least *s* baskets.

Frequent Pairs in SQL (Apriori)

- Baskets1 (TID, item)

```
SELECT b1.item, b2.item
FROM Baskets1 b1, Baskets1 b2
WHERE b1.TID = b2.TID
  AND b1.item < b2.item
GROUP BY b1.item, b2.item
HAVING COUNT(*) >= s;
```

A-Priori Algorithm

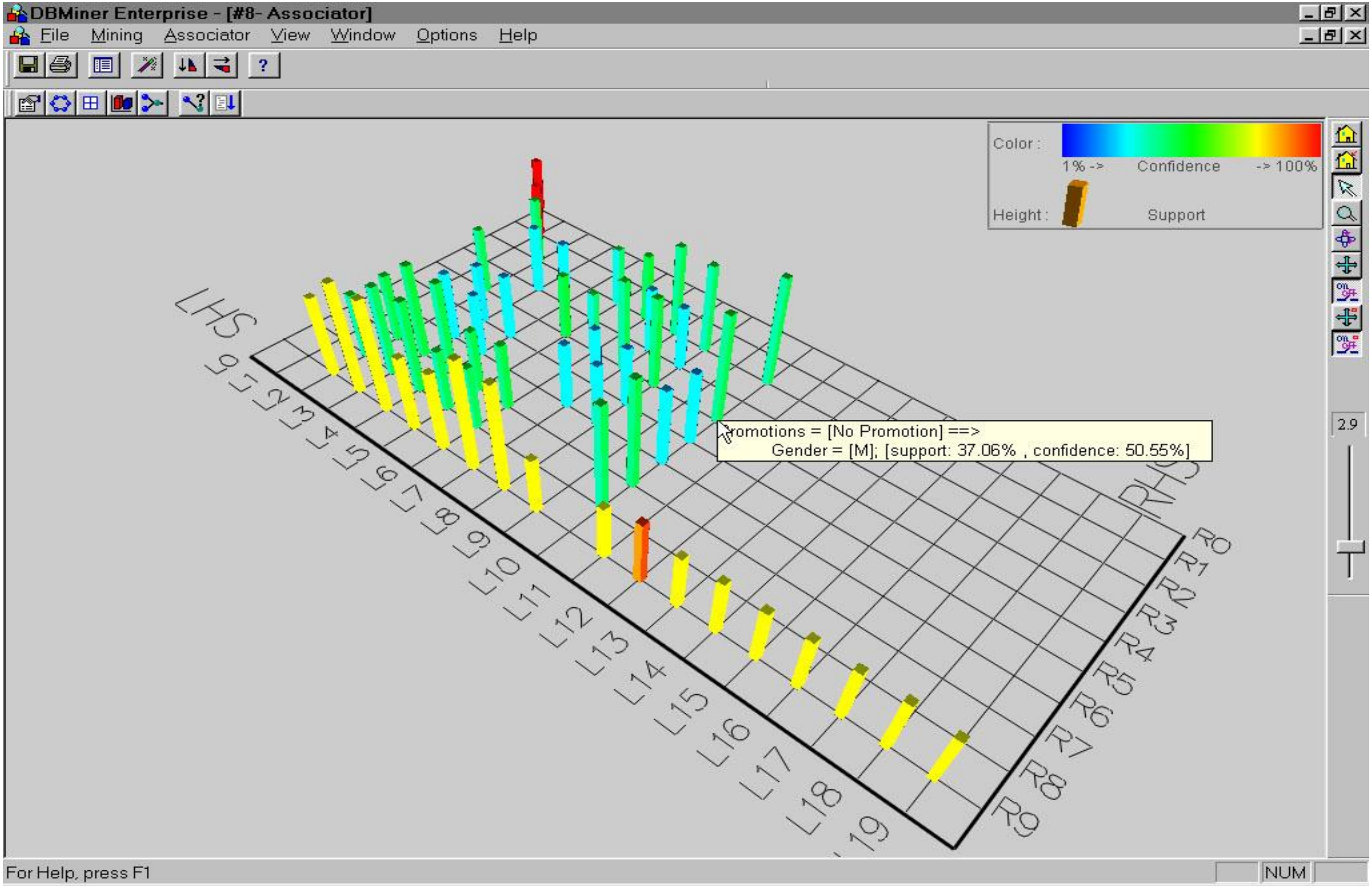
1. Materialize the view Baskets1.
2. Run the obvious query, but on Baskets1 instead of Baskets.

- Computing Baskets1 is cheap, since it does not involve a join.
- Baskets1 *probably* has many fewer tuples than Baskets.
  - Running time shrinks with the *square* of the number of tuples involved in the join.

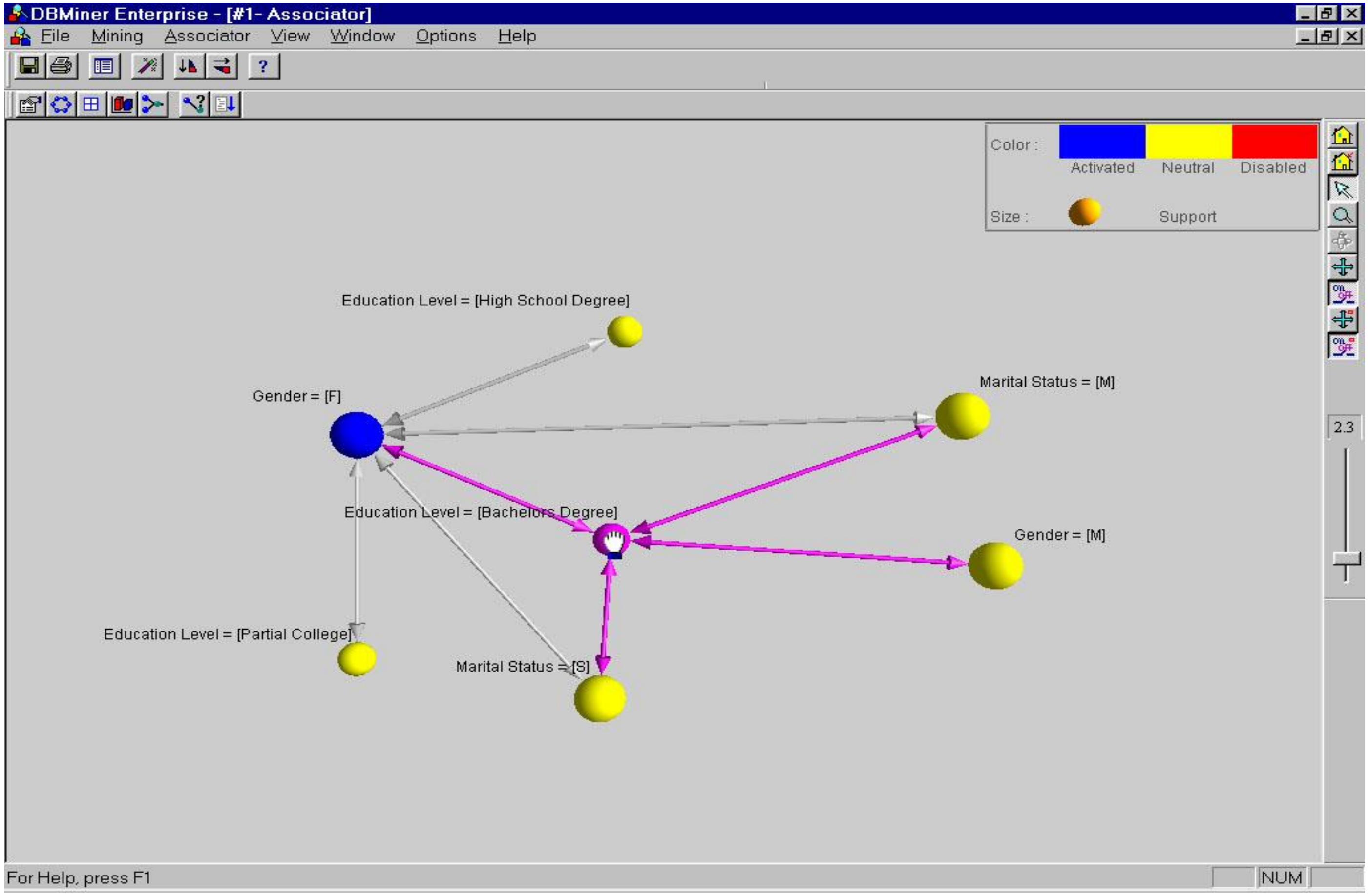
Sampling for Frequent Patterns

- Select a *sample* of original database, mine frequent patterns within sample using Apriori
- Scan *entire* database once to verify frequent itemsets found in sample (to make sure they are actually frequent over entire dataset)

Visualization of Association Rules: Plane Graph

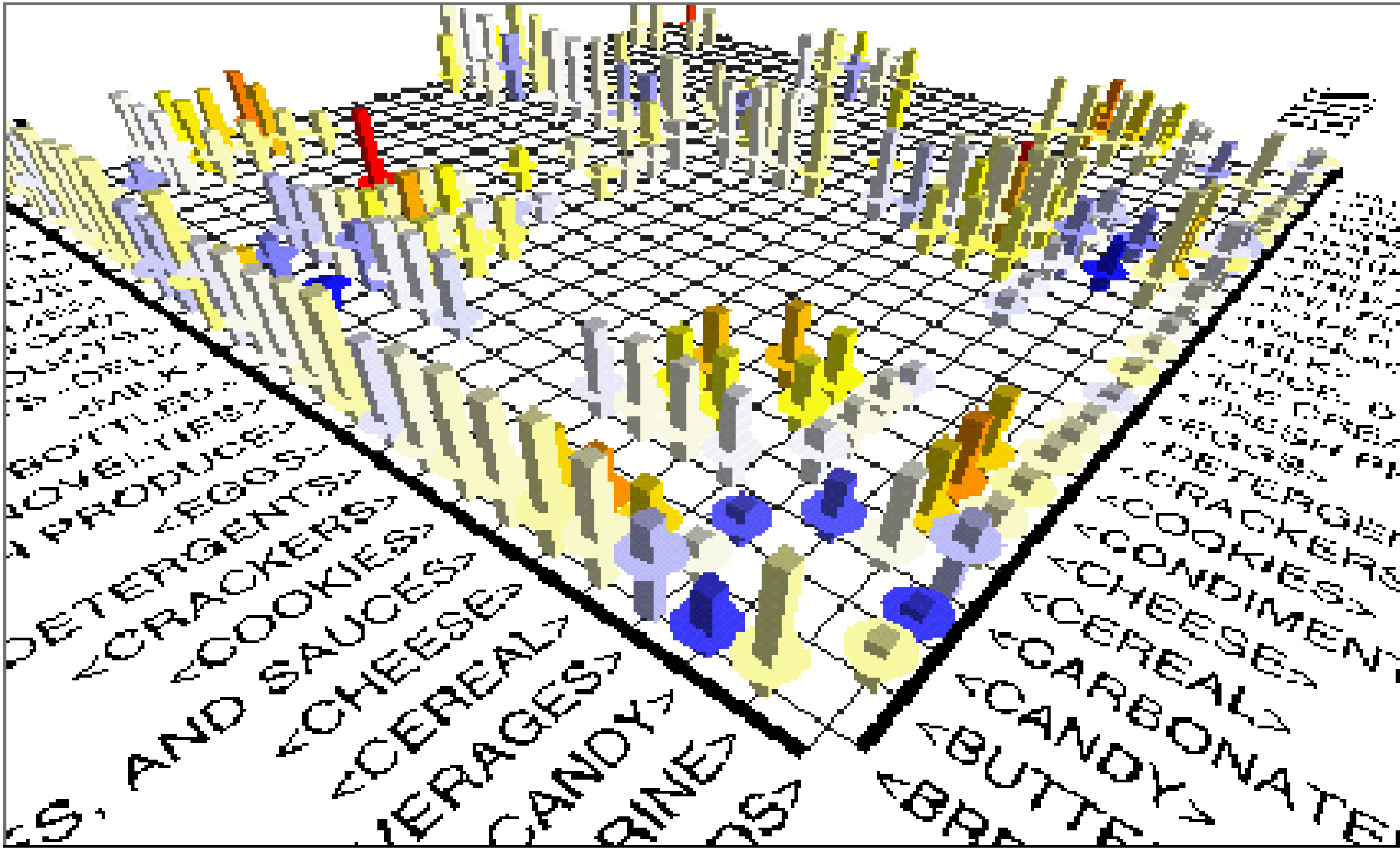


Visualization of Association Rules: Rule Graph



- Items are visualized as balls
- Arrows indicate rule implication
- Size represents support

Visualization of Association Rules (SGI/MineSet 3.0)



Interestingness Measure: Correlations (Lift)

- *play basketball* $\Rightarrow$ *eat cereal* [40%, 66.7%] is misleading
  - The overall % of students eating cereal is 75%
- *play basketball* $\Rightarrow$ *not eat cereal* [20%, 33.3%] is more accurate, although with lower support and confidence
- Measure of dependent/correlated events: lift

$$lift = \frac{P(A \cap B)}{P(A)P(B)}$$

	Basketball	Not basketball	Sum (row)
Cereal	2000	1750	3750
Not cereal	1000	250	1250
Sum(col.)	3000	2000	5000

$$lift(Basketball, Cereal) = \frac{2000/5000}{3000/5000*3750/5000} = 0.89$$

$$lift(Basketball, \neg Cereal) = \frac{1000/5000}{3000/5000*1250/5000} = 1.33$$

Summary

- Concepts: association rules, support-confident framework
- Scalable frequent pattern mining methods
  - Apriori (Candidate generation & test)
  - Sampling
- Which patterns are interesting?
  - Pattern evaluation methods

Data Warehousing and On-line Analytical Processing

- Data Warehouse: Basic Concepts
- Data Warehouse Modeling: Data Cube and OLAP
- Data Warehouse Design and Usage
- Summary

What is a Data Warehouse?

- Defined in many different ways.
  - A decision support database that is maintained separately from the organization's operational database
  - Support information processing by providing a solid platform of consolidated, historical data for analysis

Data Warehouse—Subject-Oriented

- Organized around major subjects, such as customer, product, sales
- Focusing on the modeling and analysis of data for decision makers, not on daily operations or transaction processing
- Provide a simple and concise view around particular subject issues by excluding data that are not useful in the decision support process

Data Warehouse—Integrated

- Constructed by integrating multiple, heterogeneous data sources
  - relational databases, flat files, on-line transaction records
- Data cleaning and data integration techniques are applied.
  - Ensure consistency in naming conventions, encoding structures, attribute measures, etc. among different data sources
    - E.g., Hotel price: currency, tax, breakfast covered, etc.
  - When data is moved to the warehouse, it is converted.

Data Warehouse—Time Variant

- The time horizon for the data warehouse is significantly longer than that of operational systems
  - Operational database: current value data
  - Data warehouse data: provide information from a historical perspective (e.g., past 5-10 years)

Data Warehouse—Nonvolatile

- A physically separate store of data transformed from the operational environment
- Operational update of data does not occur in the data warehouse environment
  - Does not require transaction processing, recovery, and concurrency control mechanisms
  - Requires only two operations in data accessing:
    - *initial loading of data* and *access of data*

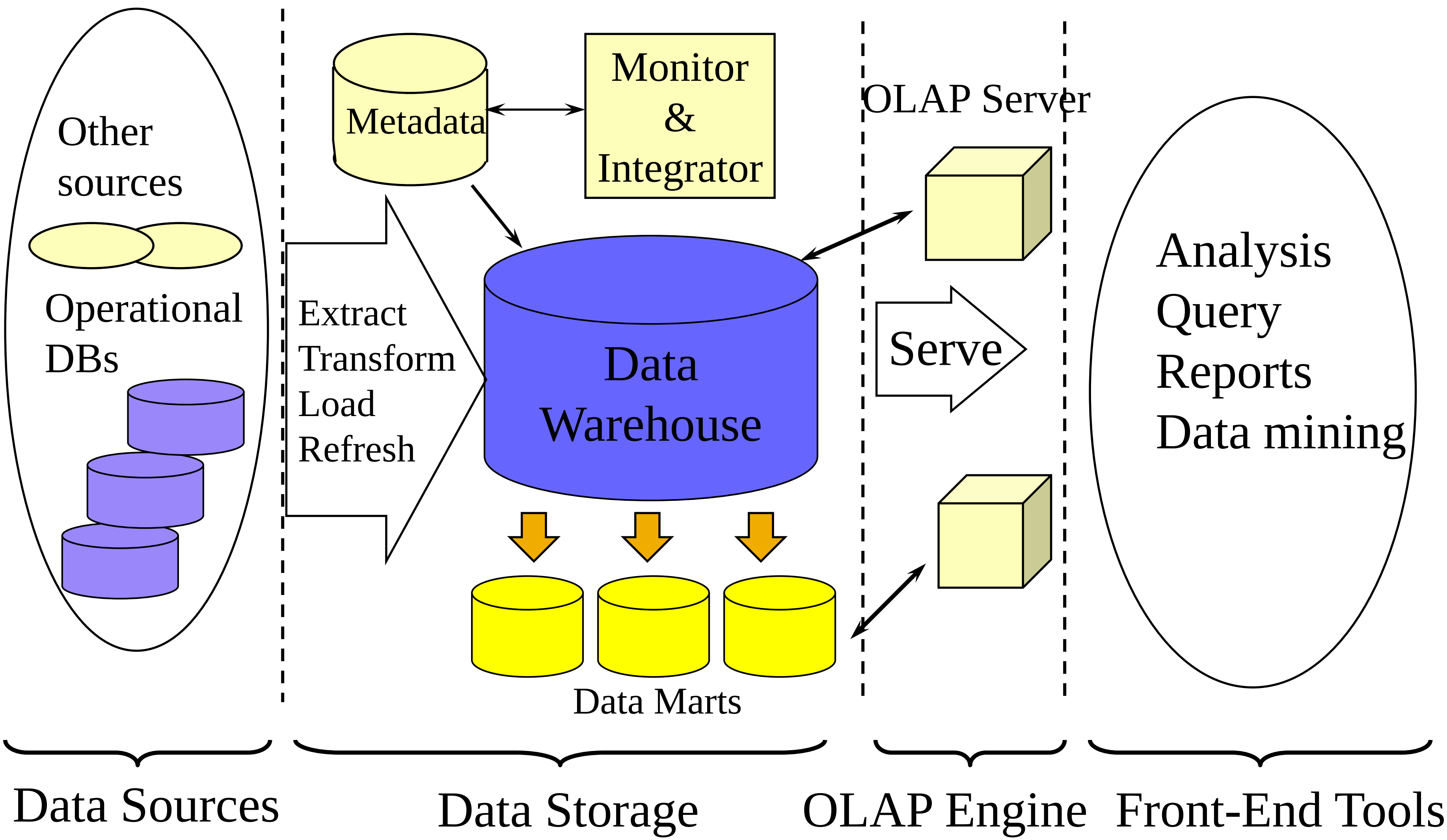
OLTP vs. OLAP

	OLTP	OLAP
users	clerk, IT professional	knowledge worker
function	day to day operations	decision support
DB design	application-oriented	subject-oriented
data	current, up-to-date detailed, flat relational isolated	historical, summarized, multidimensional integrated, consolidated
usage	repetitive	ad-hoc
access	read/write index/hash on prim. key	lots of scans
unit of work	short, simple transaction	complex query
# records accessed	tens	millions
#users	thousands	hundreds
DB size	100MB-GB	100GB-TB
metric	transaction throughput	query throughput, response

Why a Separate Data Warehouse?

- High performance for both systems
  - DBMS— tuned for OLTP (Operational Transaction Processing): access methods, indexing, concurrency control, recovery
  - Warehouse—tuned for OLAP (Online Analytical Processing): 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: Decision support 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

Data Warehouse: A Multi-Tiered Architecture



Three Data Warehouse Models

- Enterprise warehouse
  - collects all of the information about subjects spanning the entire organization
- Data Mart
  - a subset of corporate-wide data that is of value to a specific groups of users. Its scope is confined to specific, selected groups, such as marketing data mart
- Virtual warehouse
  - A set of views over operational databases
  - Only some of the possible summary views may be materialized

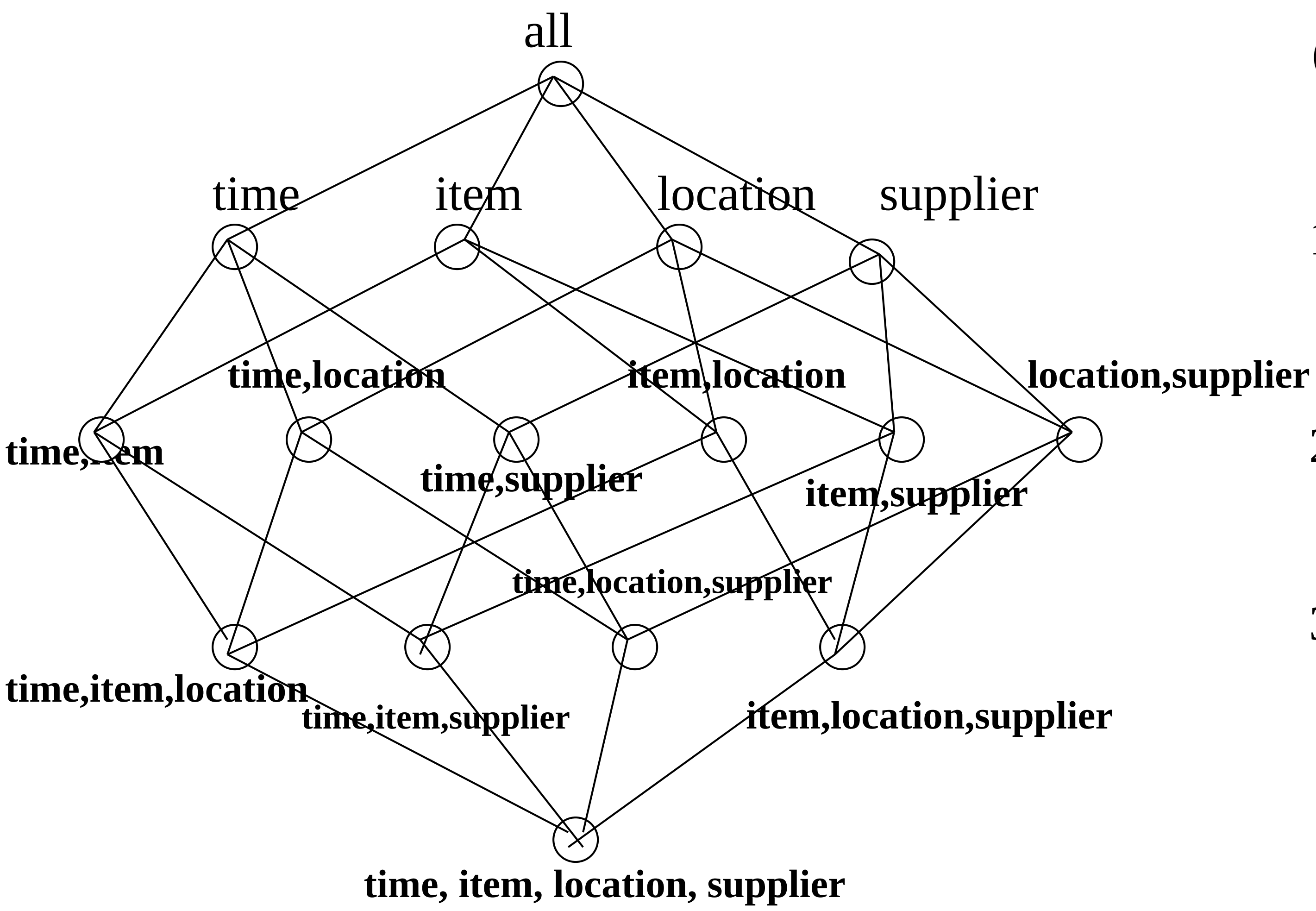
Extraction, Transformation, and Loading (ETL)

- **Data extraction**
  - get data from multiple, heterogeneous, and external sources
- **Data cleaning**
  - detect errors in the data and rectify them when possible
- **Data transformation**
  - convert data from legacy or host format to warehouse format
- **Load**
  - sort, summarize, consolidate, compute views, check integrity, and build indicies and partitions
- **Refresh**
  - propagate the updates from the data sources to the warehouse

From Tables and Spreadsheets to Data Cubes

- A **data warehouse** is based on a multidimensional data model which views data in the form of a data cube
- A data cube 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**, such as sales 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.

Cube: A Lattice of Cuboids



0-D (*apex*) cuboid

1-D cuboids

2-D cuboids

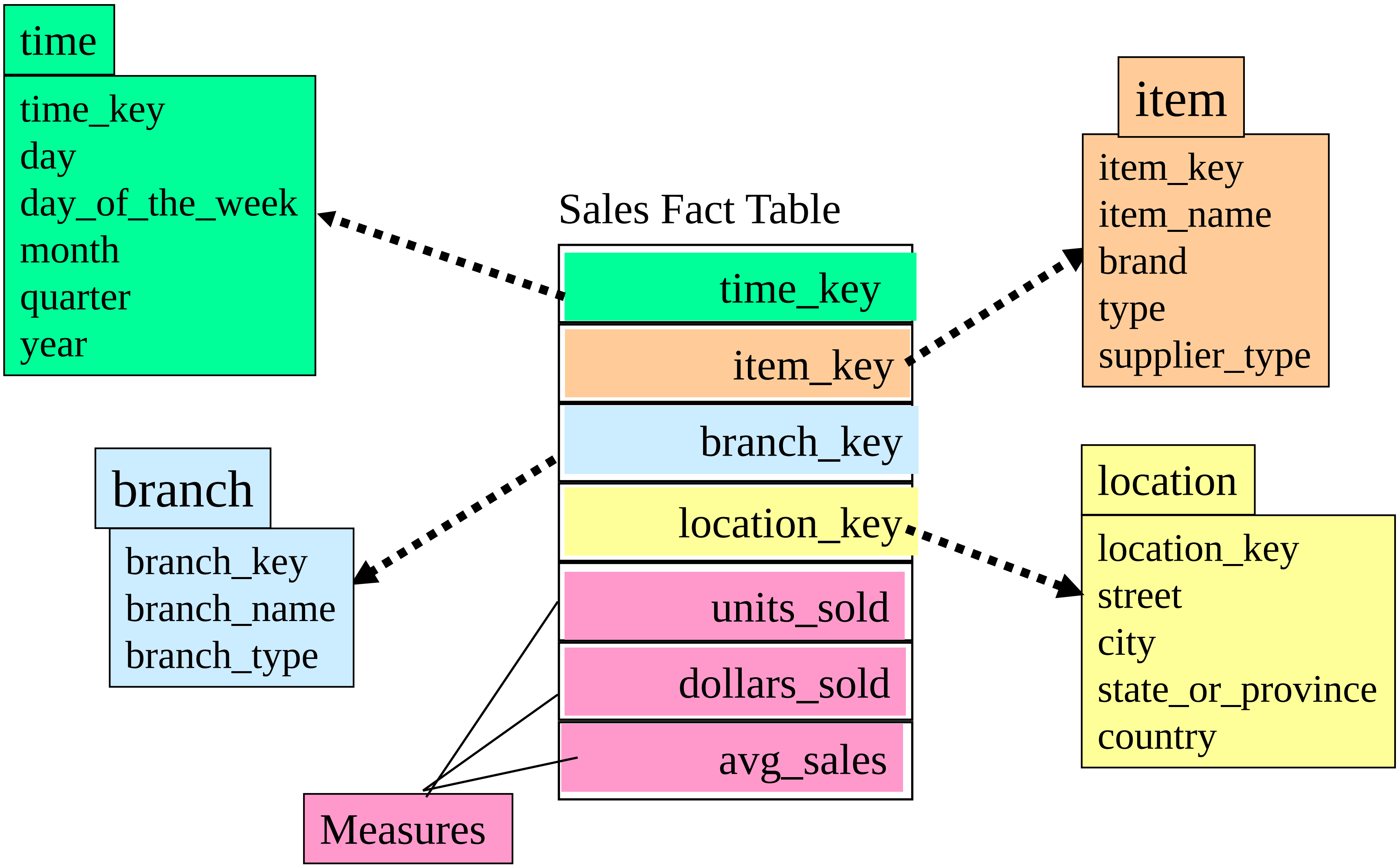
3-D cuboids

4-D (*base*) cuboid

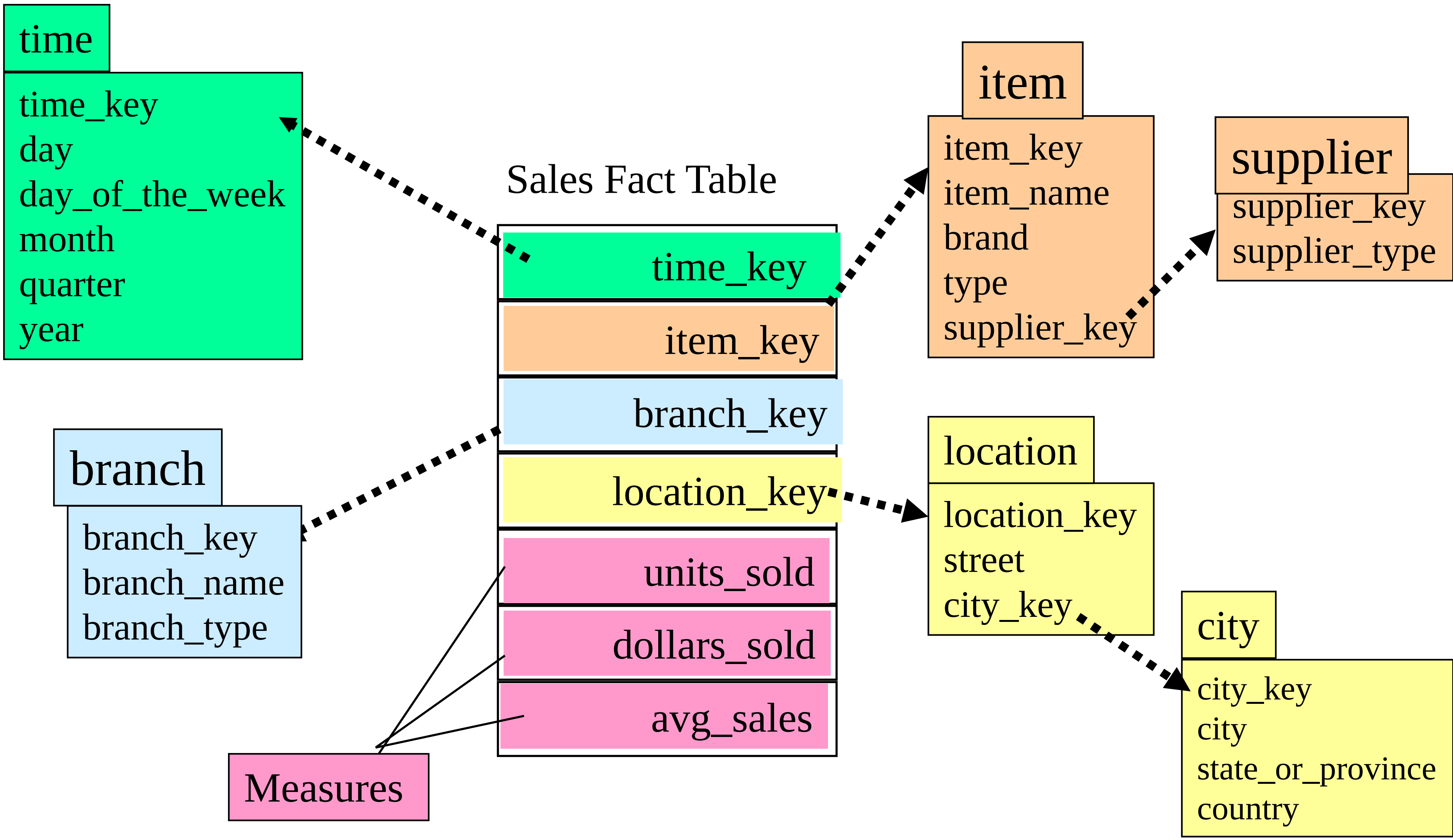
Conceptual Modeling of Data Warehouses

- Modeling data warehouses: dimensions & measures
  - Star schema: A fact table in the middle connected to a set of dimension tables
  - Snowflake schema: A refinement of star schema where some dimensional hierarchy is normalized into a set of smaller dimension tables, forming a shape similar to snowflake
  - Fact constellations: Multiple fact tables share dimension tables, viewed as a collection of stars, therefore called galaxy schema or fact constellation

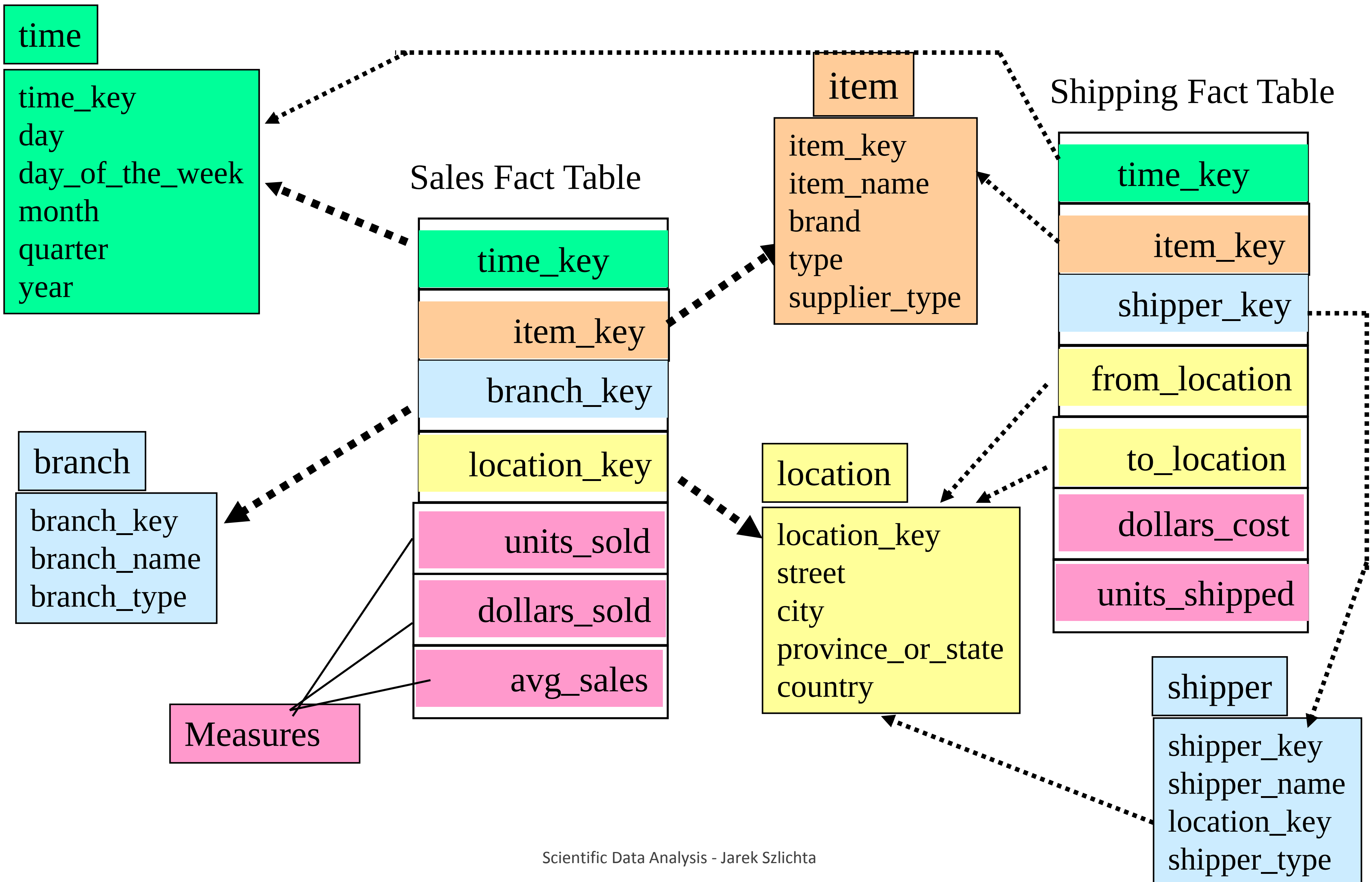
Example of Star Schema



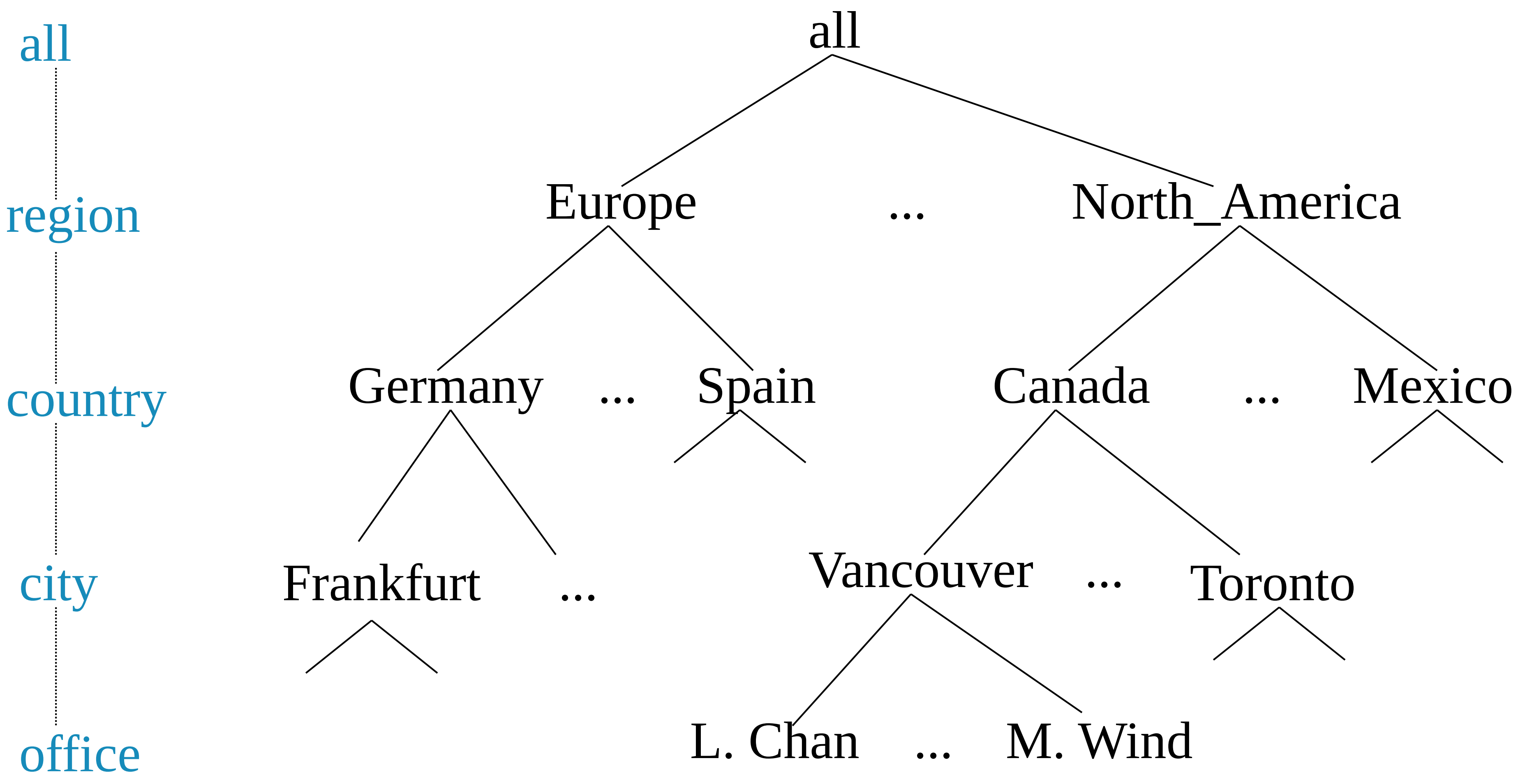
Example of Snowflake Schema



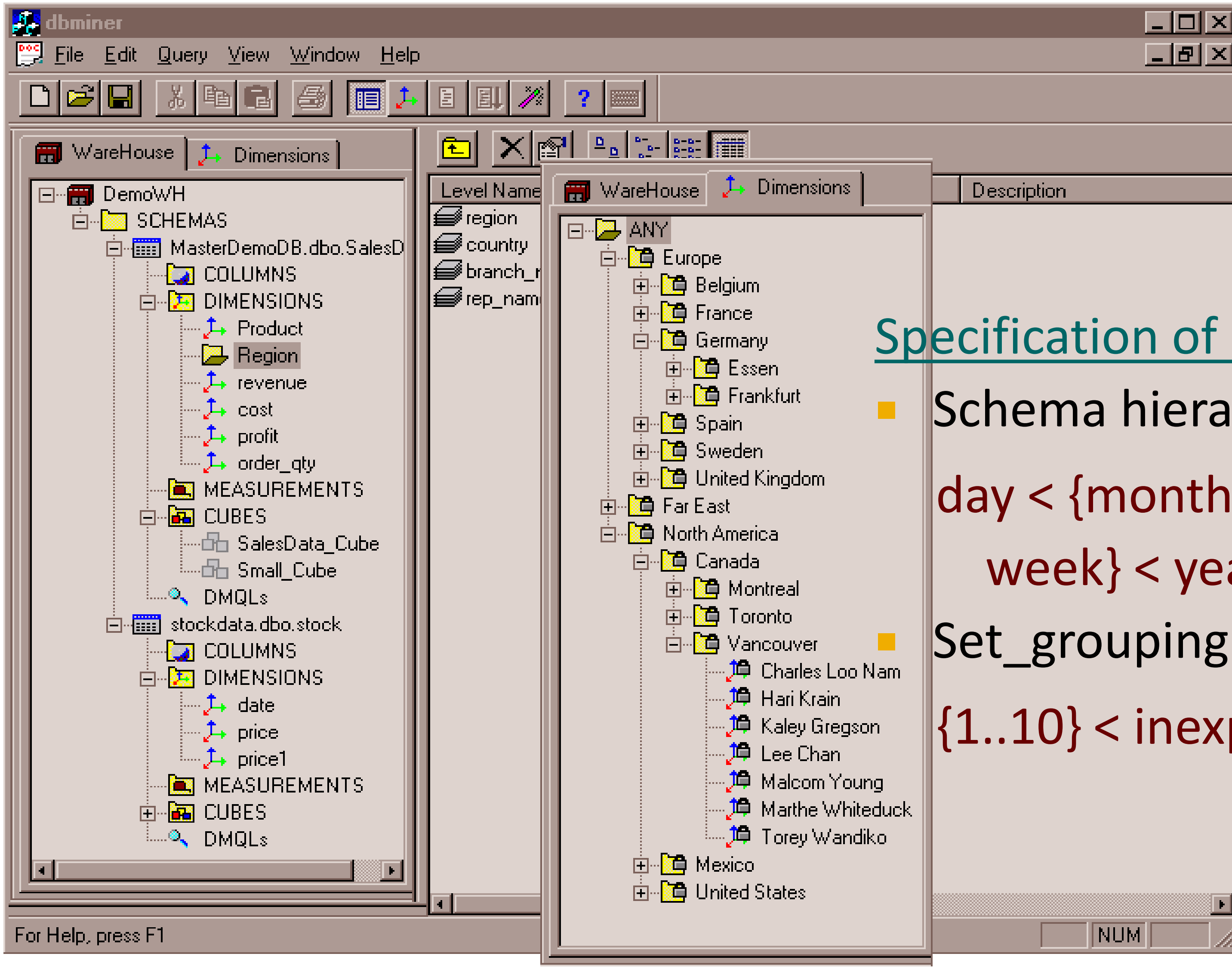
Example of Fact Constellation



A Concept Hierarchy: Dimension (location)



View of Warehouses and Hierarchies



Specification of hierarchies

- Schema hierarchy

  day < {month < quarter; week} < year

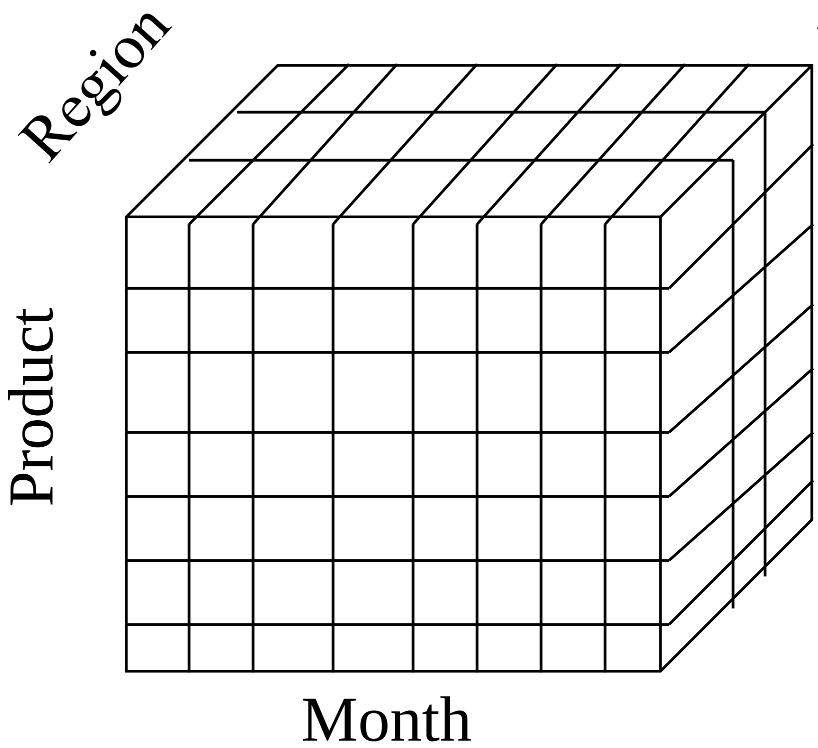
- Set_grouping hierarchy

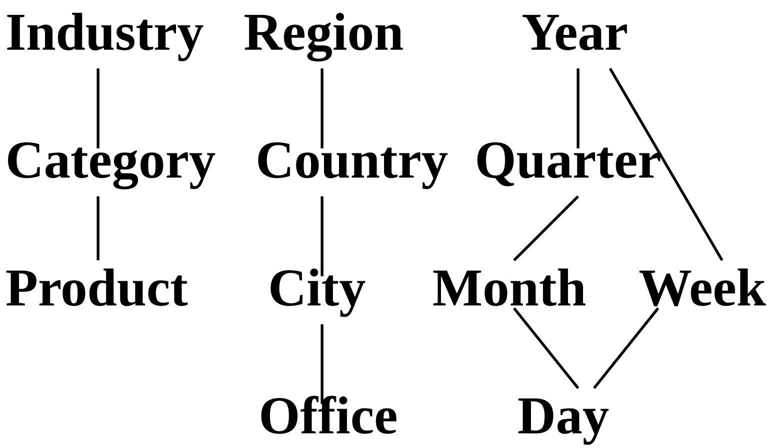
  {1..10} < inexpensive

Multidimensional Data

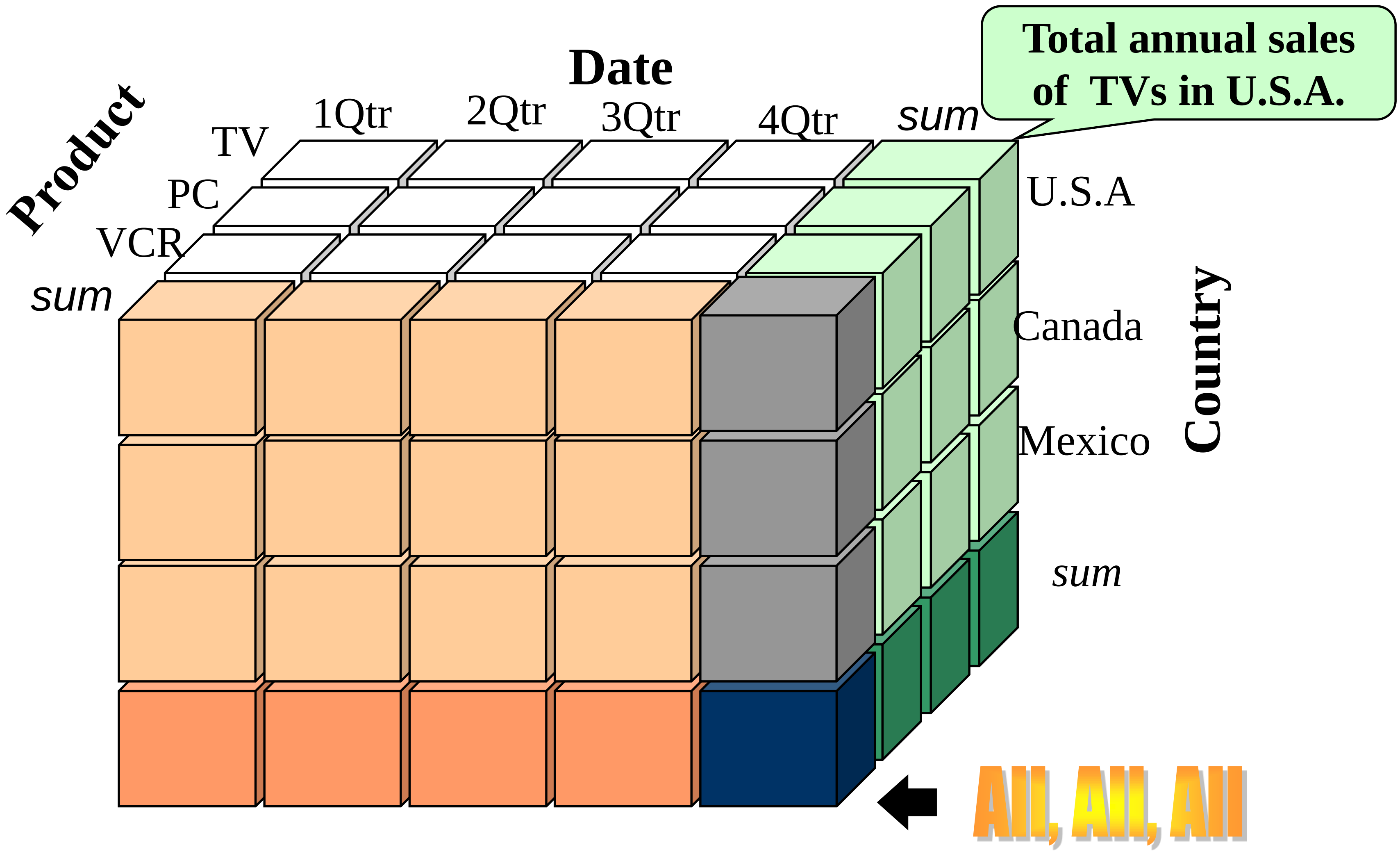
- Sales volume as a function of product, month, and region

Dimensions: ***Product, Location, Time***
Hierarchical summarization paths

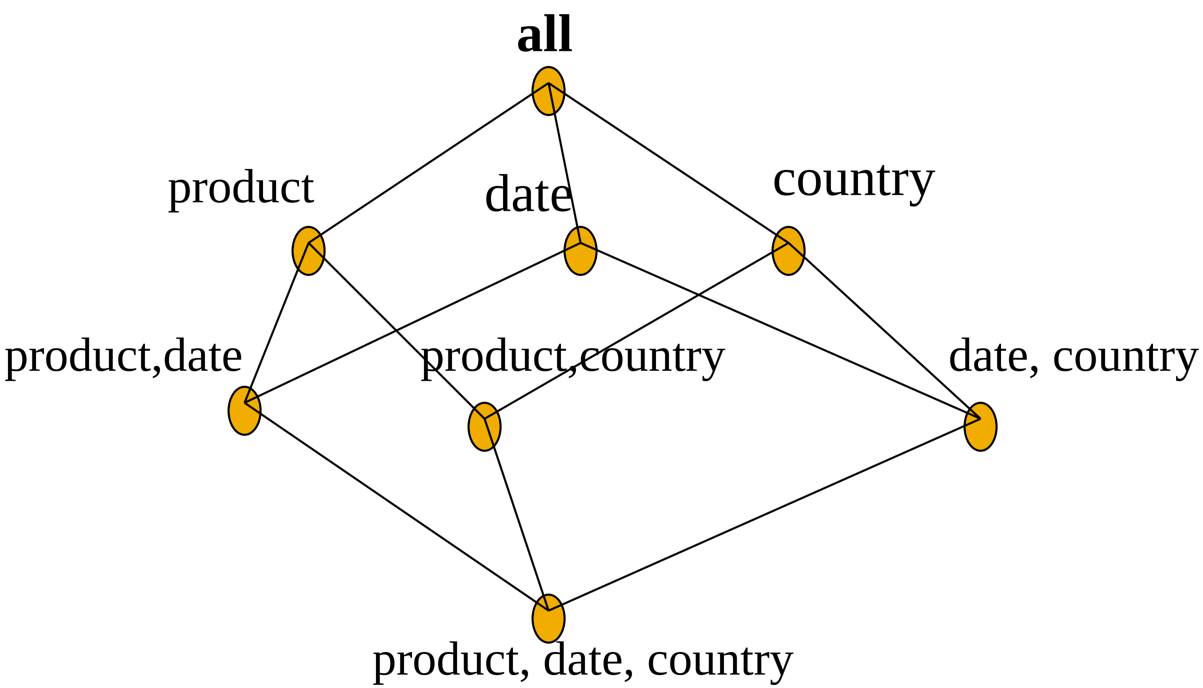




A Sample Data Cube



Cuboids Corresponding to the Cube



0-D (*apex*) cuboid

1-D cuboids

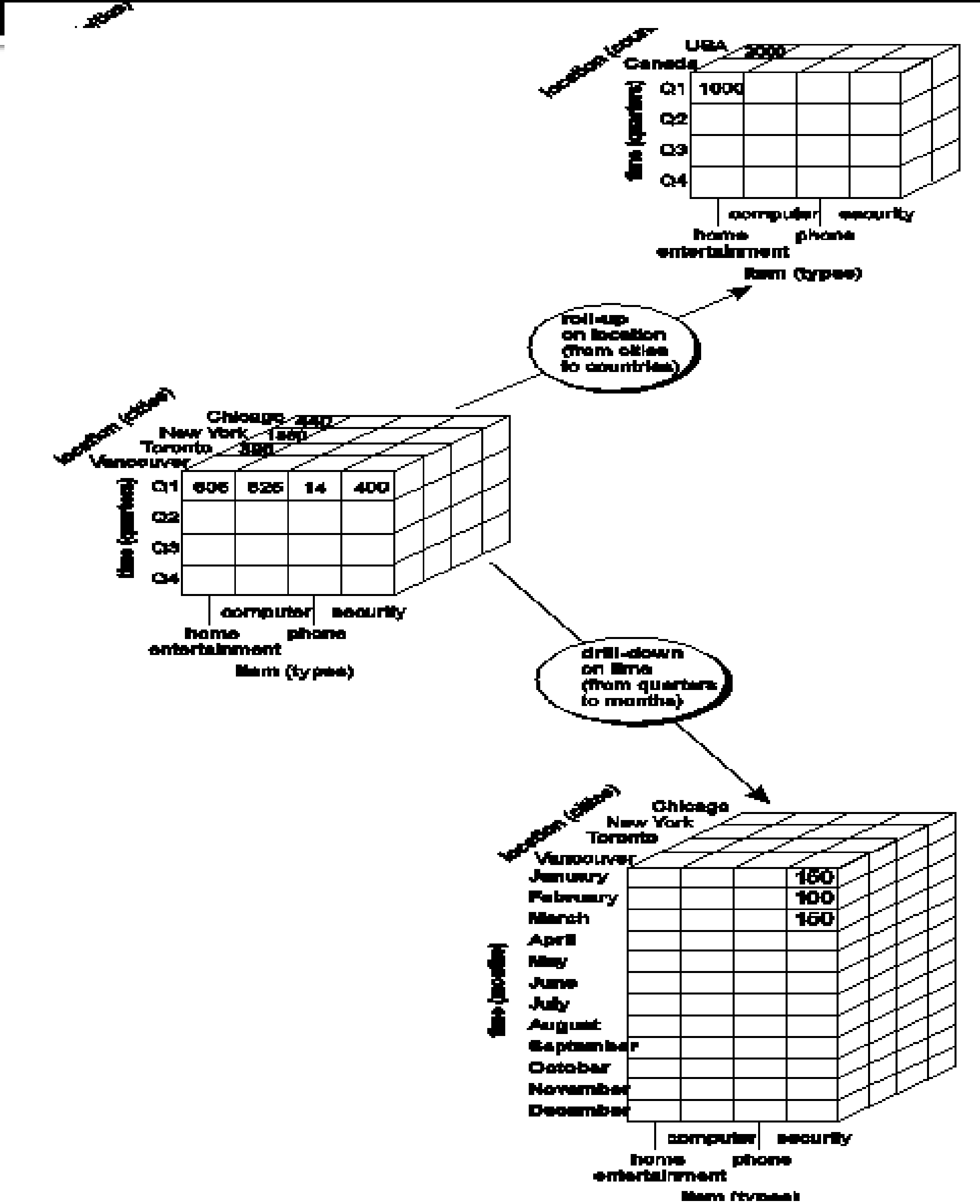
2-D cuboids

3-D (*base*) cuboid

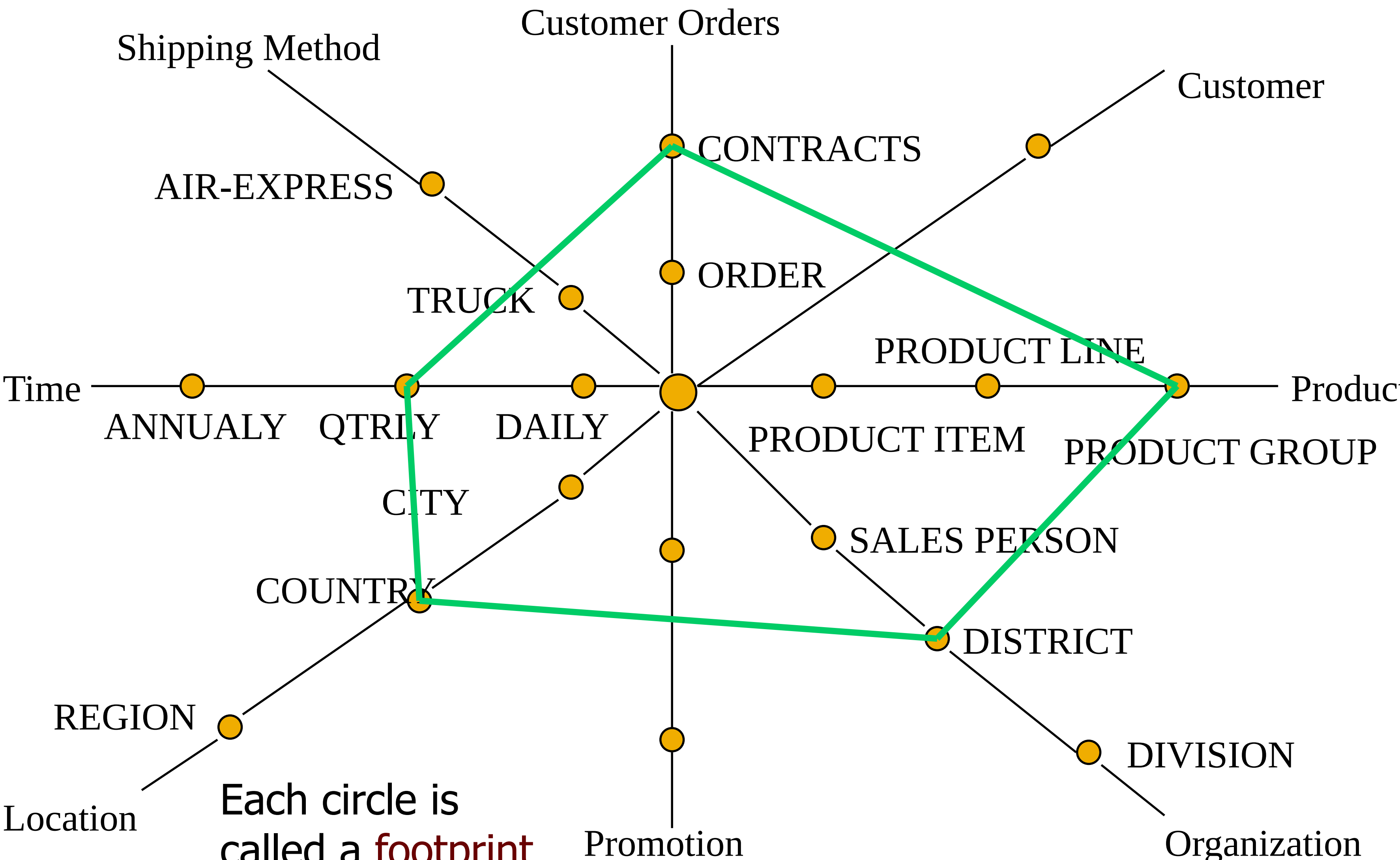
Typical OLAP Operations

- Roll up (drill-up): summarize data
  - *by climbing up hierarchy or by dimension reduction*
- Drill down (roll down): reverse of roll-up
  - *from higher level summary to lower level summary or detailed data, or introducing new dimensions*

Typical OLAP Operations

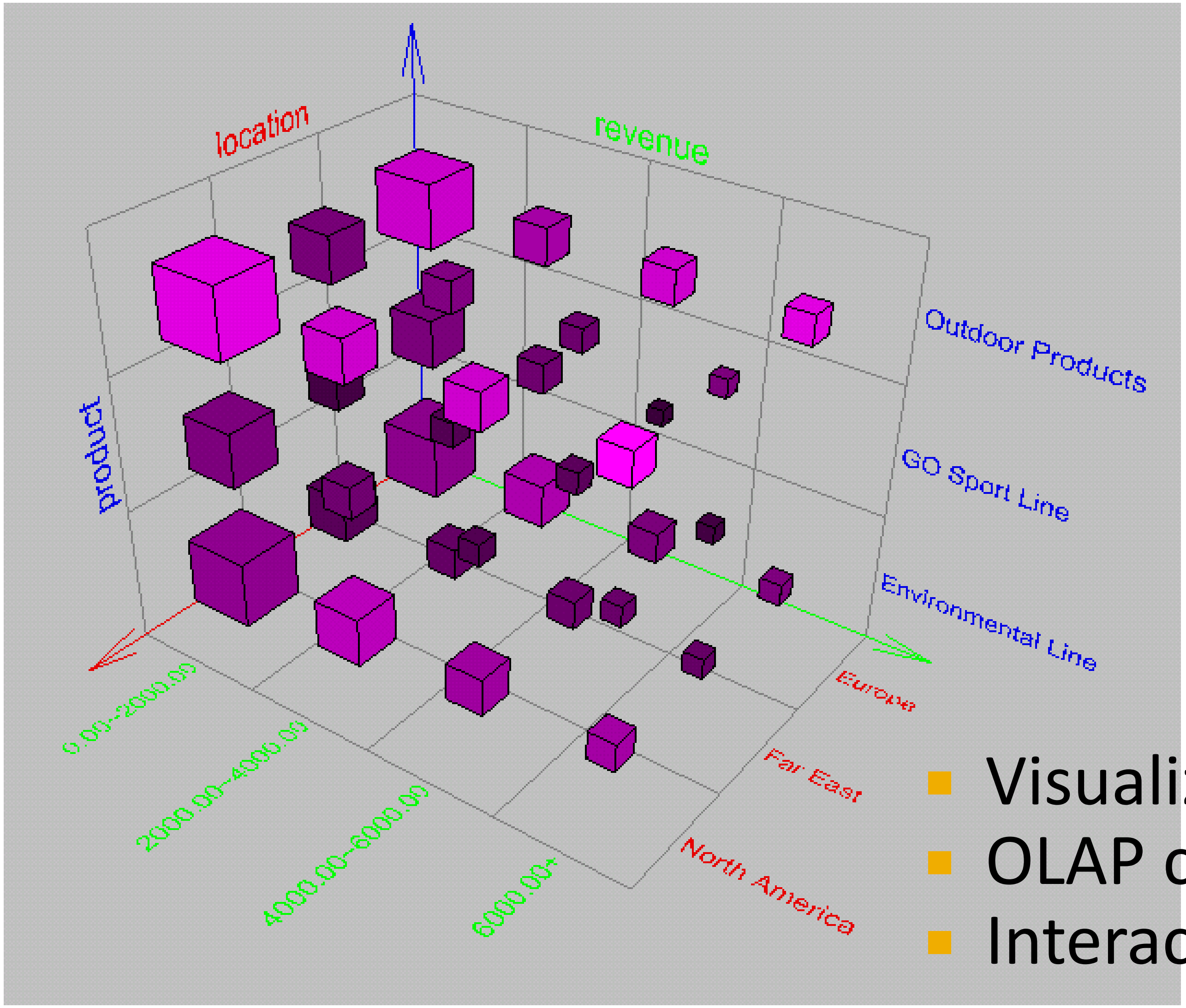


A Star-Net Query Model



Each circle is called a footprint

Browsing a Data Cube



- Visualization
- OLAP capabilities
- Interactive manipulation

Summary

- Data warehousing: A multi-dimensional model of a data warehouse
  - A data cube consists of *dimensions* & *measures*
  - Star schema, snowflake schema, fact constellations
  - OLAP operations: drilling, rolling
- Data Warehouse Architecture, Design, and Usage
  - Multi-tiered architecture
  - Business analysis design framework
  - Information processing, analytical processing, data mining

Reading List

- Recommended
  - Review Slides!
  - Book: Jiawei Han, Micheline Kamber and Jian Pei, Data Mining - Concepts and Techniques, Morgan Kaufmann, Third Edition, 2011 (or 2nd edition)
    - http://ccs1.hnue.edu.vn/hungtd/DM2012/DataMining_BOOK.pdf
    - Chapters: 3, 4, 5
- Optional
  - (Association Rules) R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. SIGMOD'93