Introduction to Data Warehouse and Implementation

Fangyu Lin
Mei Yang

Outline

- 1. Introduction and Definition
- 2. Multi-dimensional Data Warehouse Model
- 3. Data Mining Query Language (DMQL)
- 4. Introduce to OLAP and its Architecture
- 5. Data Warehouse Architecture and Design
- 6. Data Warehouse Implementation (Integration)

Introduction

- Definition of Data Warehouse:
- 1. Subject Oriented:

  Provide a simple and concise view of major subjects by focusing on modeling and analyzing of data for the decision maker.

- 2. Integrated:

  Construct multiple data sources by using data cleaning and data integration technique, and data is converted.

- 3. Time Variant:

  Every DW may or may not contains explicit or inexplicit time and provides information from a historical perspective.

- 4. Non-volatile:

  Data in are physical separate stored in DW which does not have operational update, and only allows operation of initial loading data and access of data

DW vs. Heterogeneous & Operational DBMS

1. Heterogeneous DBMS	a. build wrappers or mediators on top of DM to do the integration. b. Query Driven, complex info filtering
2. Operational DBMS	a. OLTP (on-line transaction processing) b. Operations in purchasing, inventory, banking, manufacturing, payroll registration, accounting…
3. Data Warehouse	a. Update-driven, high performance b. OLAP (on-line analytical processing) c. Data analysis and decision making

OLAP v.s. OLTP

DBMS | Data Warehouse

	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 separate data warehouse?

Outline

- 1. Introduction and Definition
- 2. Multi-dimensional Data Warehouse Model
- 3. Data Mining Query Language (DMQL)
- 4. Introduce to OLAP and its Architecture
- 5. Data Warehouse Architecture and Design
- 6. Data Warehouse Implementation (Integration)

Modeling of Data Warehouse

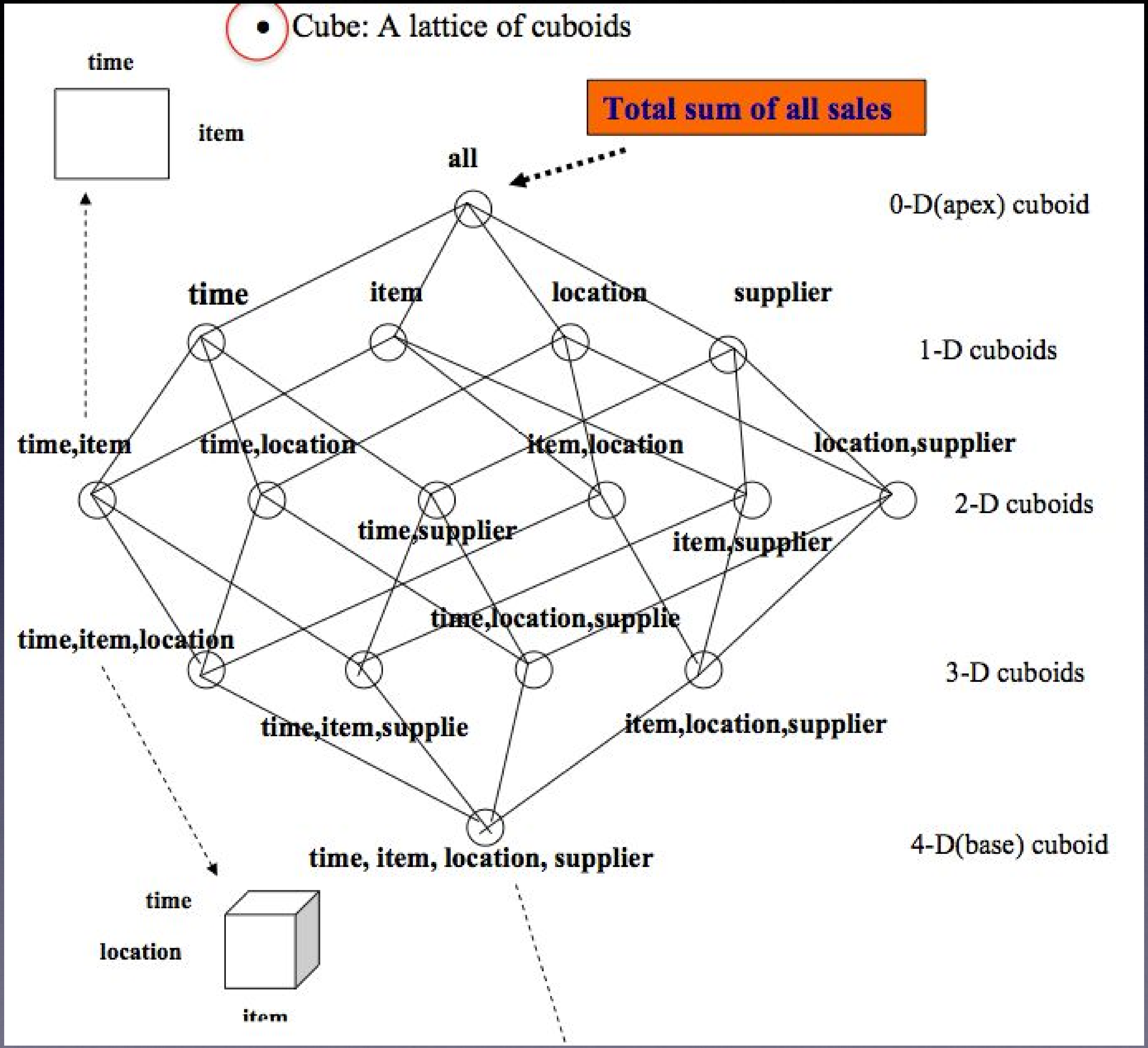
- 1. View data in the form of cube

  n-D cube = n entitis
  The n-D base cube is called a base cuboid
  The top 0-D cuboid is called apex cuboid

- Example cube: sales

  Dimensional Tables: time, item, location, supplier
  Fact Table: contains Keys and Measures

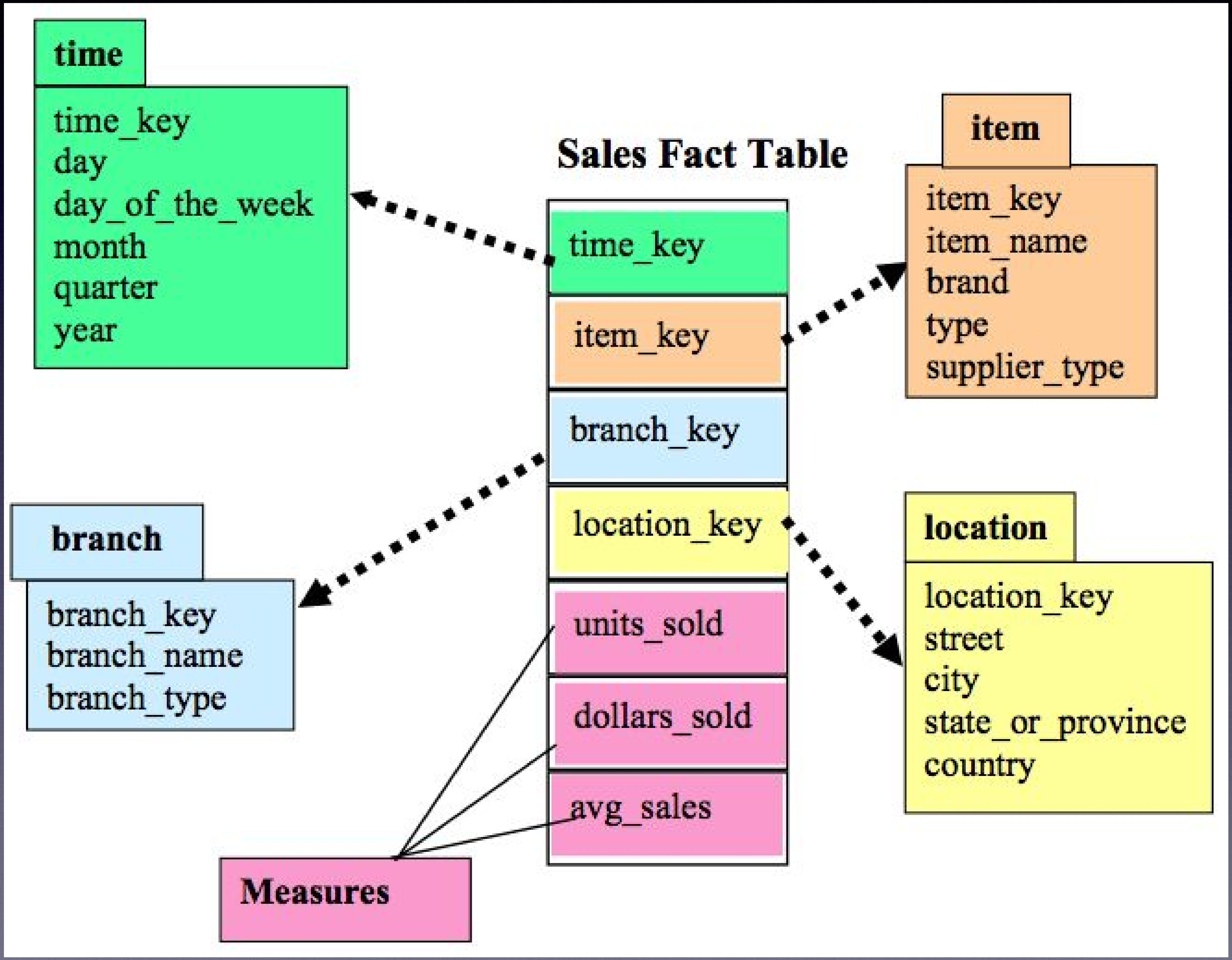
Example of cuboids (table sales):



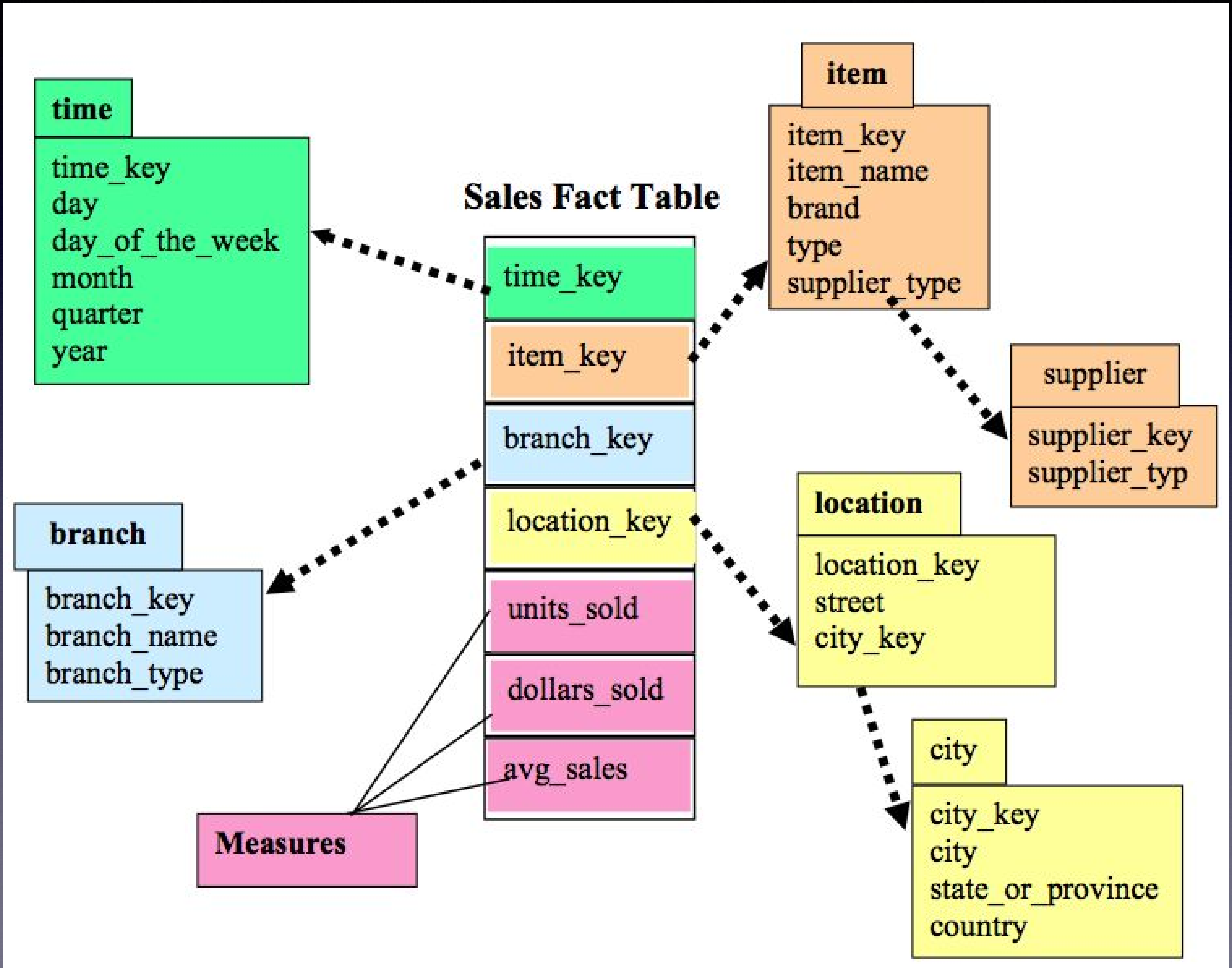
Conceptual Modeling of Data Warehouse

1. Star Schema	A fact table in the middle connected to a set of dimension tables
2. Snowflake schema	A refinement of the star schema where some dimensional hierarchy is normalized into a set of smaller dimensional tables.
3. Fact constellations (Galaxy Schema)	Multiple fact tables share dimensional tables, which can be viewed as collection of stars.

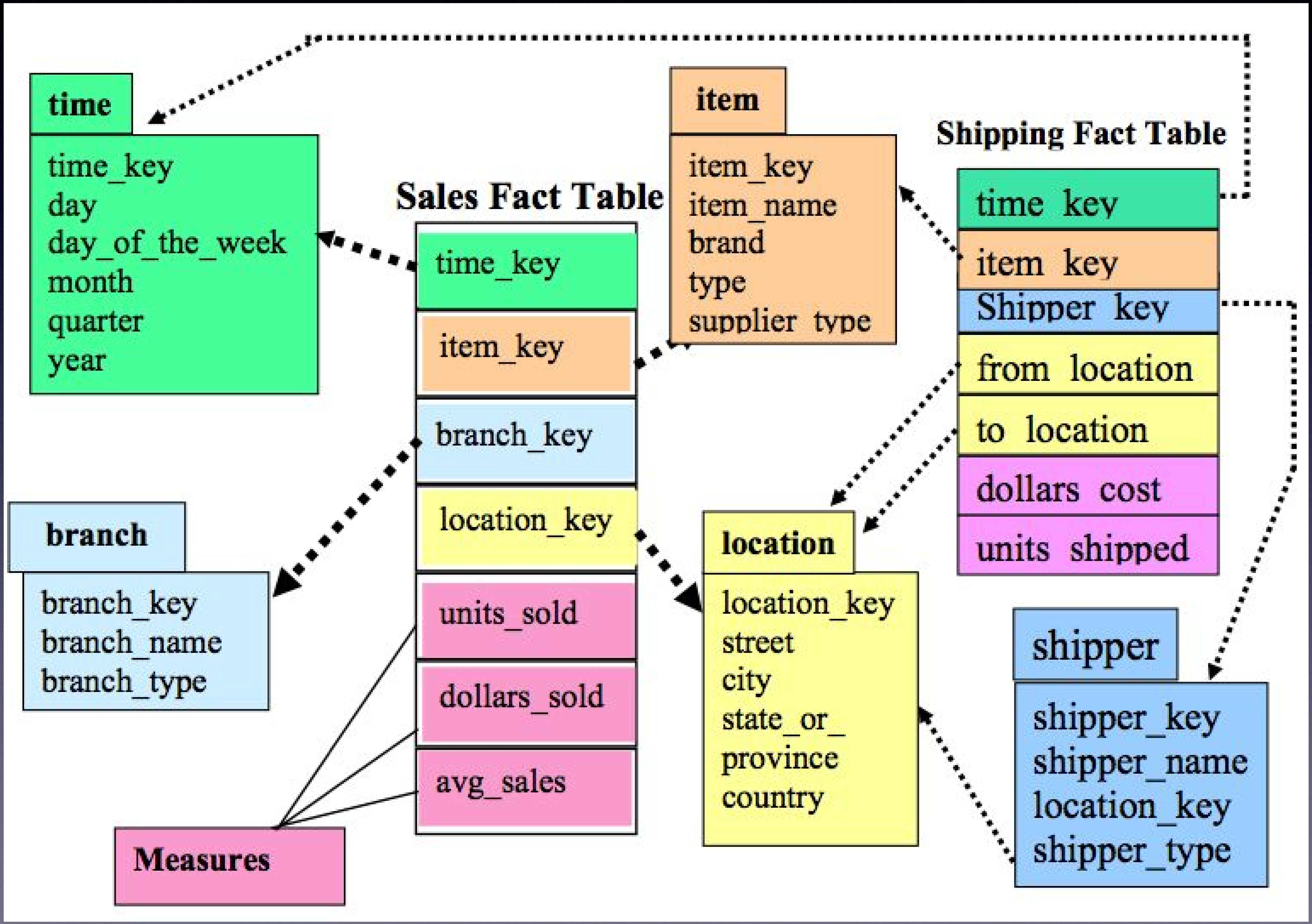
Star Schema


time
time_key
day
day_of_the_week
month
quarter
year
Sales Fact Table
time_key
item_key
branch_key
location_key
units_sold
dollars_sold
avg_sales
item
item_key
item_name
brand
type
supplier_type
branch
branch_key
branch_name
branch_type
location
location_key
street
city
state_or_province
country
Measures

Snowflake Schema



Fact constellations



Outline

- 1. Introduction and Definition
- 2. Multi-dimensional Data Warehouse Model
- 3. Data Mining Query Language (DMQL)
- 4. Introduce to OLAP and its Architecture
- 5. Data Warehouse Architecture and Design
- 6. Data Warehouse Implementation (Integration)

Data Mining Query Language

- 1. Definition:

Need a data definition language to define the table in the conceptual model

- 2. Syntax:

define cube: <cube_name> [<dimensional_list>]: <measure_list>

fact table

define dimension: <dimension_name> **as** (<attributes_or_list_of_subdimension>)

dimensional table

define dimension <dimension_name> **as** <dimension_name_first_time>
in cube <cube_name_first_time>

share dimensional table

Example of define cube sales

define cube sales [time, item, branch, location]:

dollars_sold = sum(sales_in_dollars),

avg_sales = avg(sales_in_dollars),
units_sold = count(*)

star

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

define dimension item **as** (item_key, item_name, brand, type,
supplier(supplier_key, supplier_type))
define dimension location **as** (location_key, street,
city(city_key, province_or_state, country))

example of fact constellation

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 in DMQL

1. Distributive	The Result derived by applying the function to n aggregate values is the same as that derived by applying the function on all data without partitioning. **Example**: count(), sum(), min(), max()
2. Algebraic	Use distributive aggregate functions it is computed by an algebraic function with M arguments, each of which is obtained by applying a distributive aggregate function. **Example**: avg(), min_N(), standard_deviation()
3. Holistic	If there is no constant bound on the storage size needed to describe a sub-aggregate. **Example**: median(), mode(), rank()

Measures Example

- Sales Table:

time (time_key, day, day_of_week, month, quarter, year)
item (item_key, item_name, brand, type, supplier(supplier_key, supplier_type))
branch (branch_key, branch_name, branch_type)
location (location_key, street, city, province_or_state, country)

sales (time_key, item_key, branch_key, location_key, number_of_unit_sold, price)

To compute dollar_sold & unit_sold:

select s.time_key, s.item_key, s.branch_key, s.location_key,
    sum(s.number_of_units_sold*s.price),
    sum(s.number_of_units_sold)
from time t, item i, branch b, location l, sales s
where s.time_key = t.time_key and s.item_key = i.item_key
    and s.branch_key = b.branch_key and s.location_key = l.location_key
group by s.time_key, s.item_key, s.branch_key, s.location_key

Questions?

- What’s the relation between “data cube” and “group by” ?

- what’s query for the 0-D cuboid or apex?

Outline

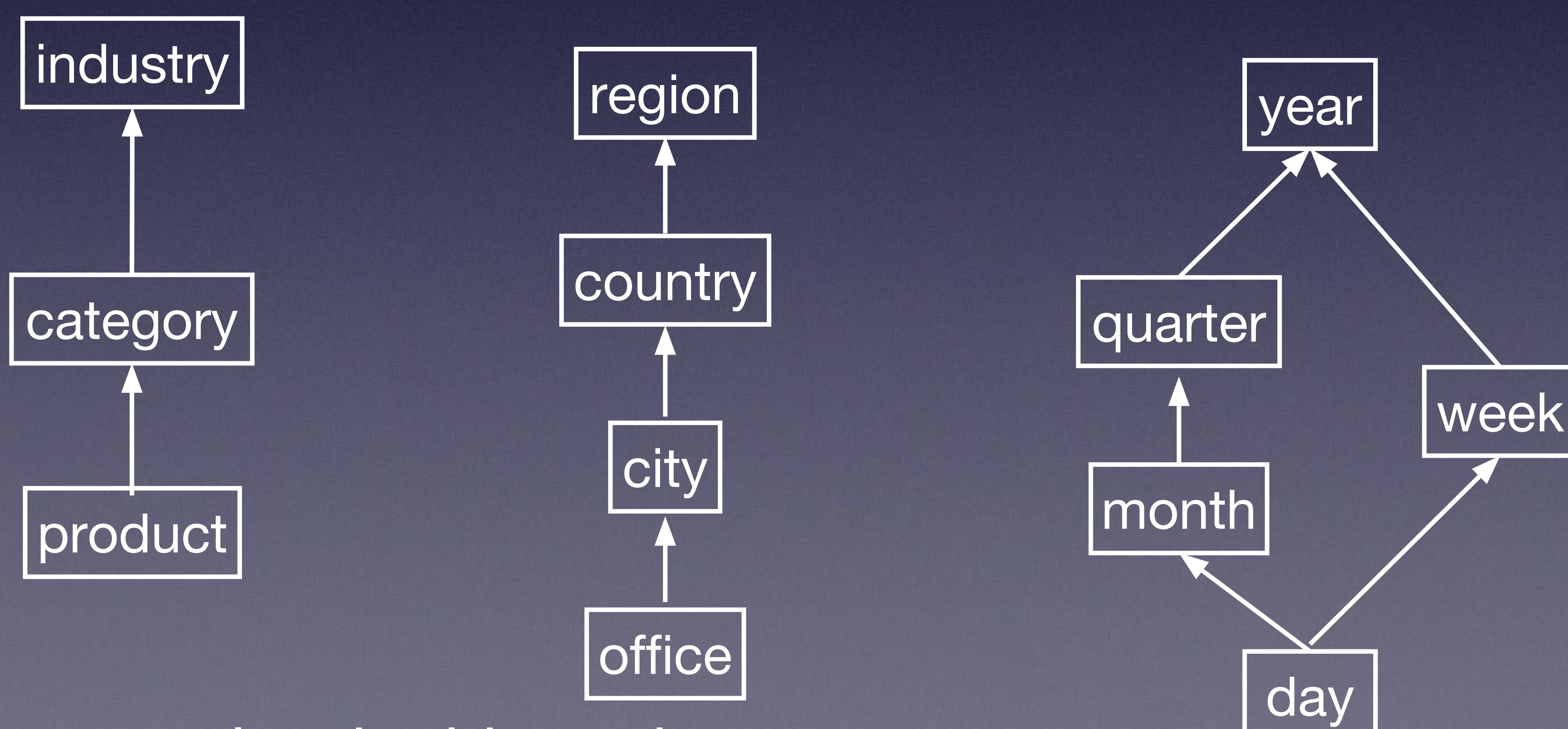
- 1. Introduction and Definition
- 2. Multi-dimensional Data Warehouse Model
- 3. Data Mining Query Language (DMQL)
- 4. Introduce to OLAP and Operations
- 5. Data Warehouse Architecture and Design
- 6. Data Warehouse Implementation (Integration)

OLAP Operations in a Multidimensional Data

- 1. Dimension Hierarchical Concept:
    a. Total order hierarchy
    b. Partial order hierarchy
- 2. Operations:
    a. roll up (drill up)
    b. drill down (droll down)
    c. slice and dice (project & select)
    d. pivot (rotate)
    e. drill cross & drill through

Example of Dimensional Hierarchy

- Product dimension: Product<Category<industry
- Location dimension: Office<city<Country<Region
- Time dimension: Day<{month<quarter;week}<year

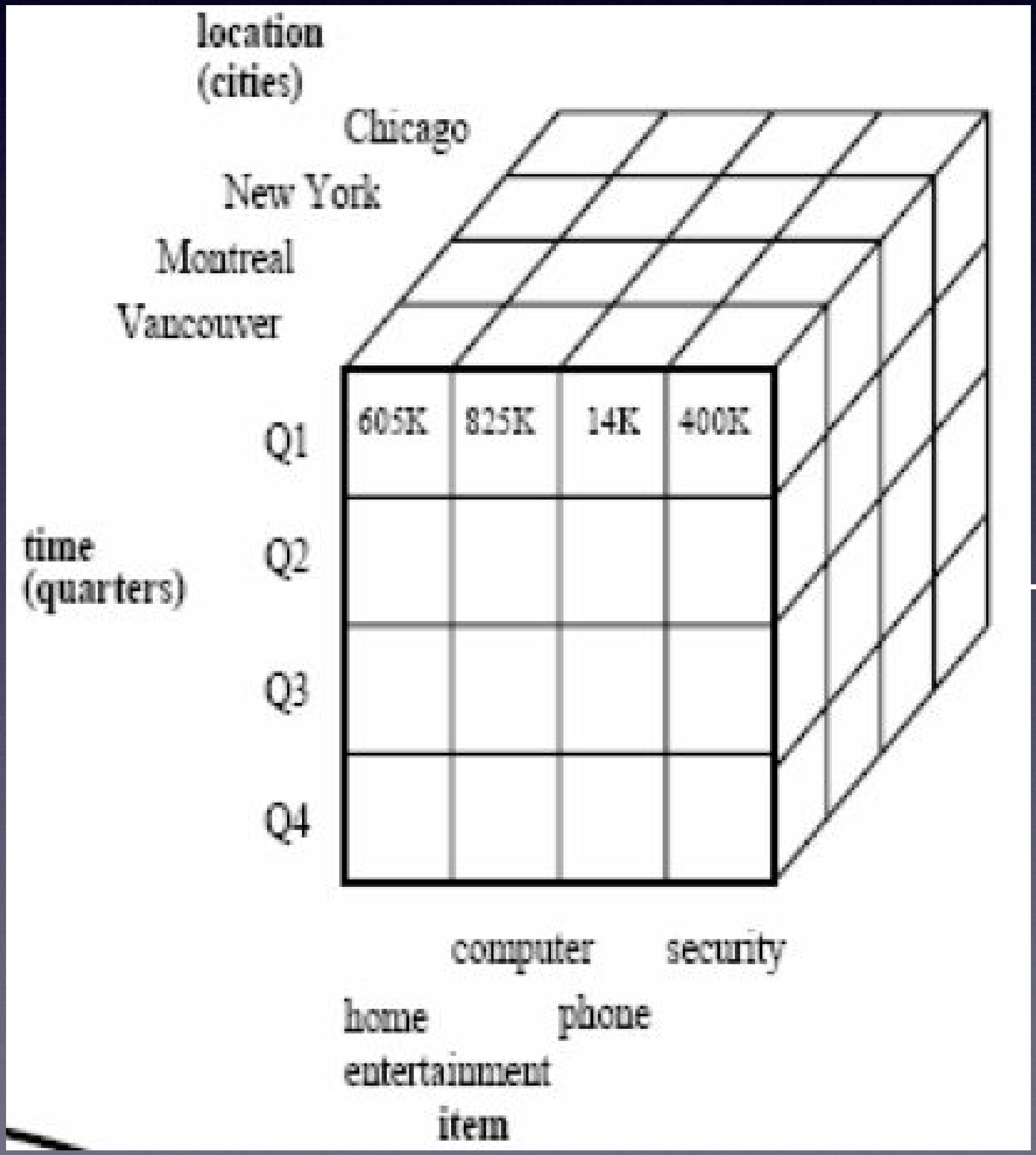


total order hierarchy

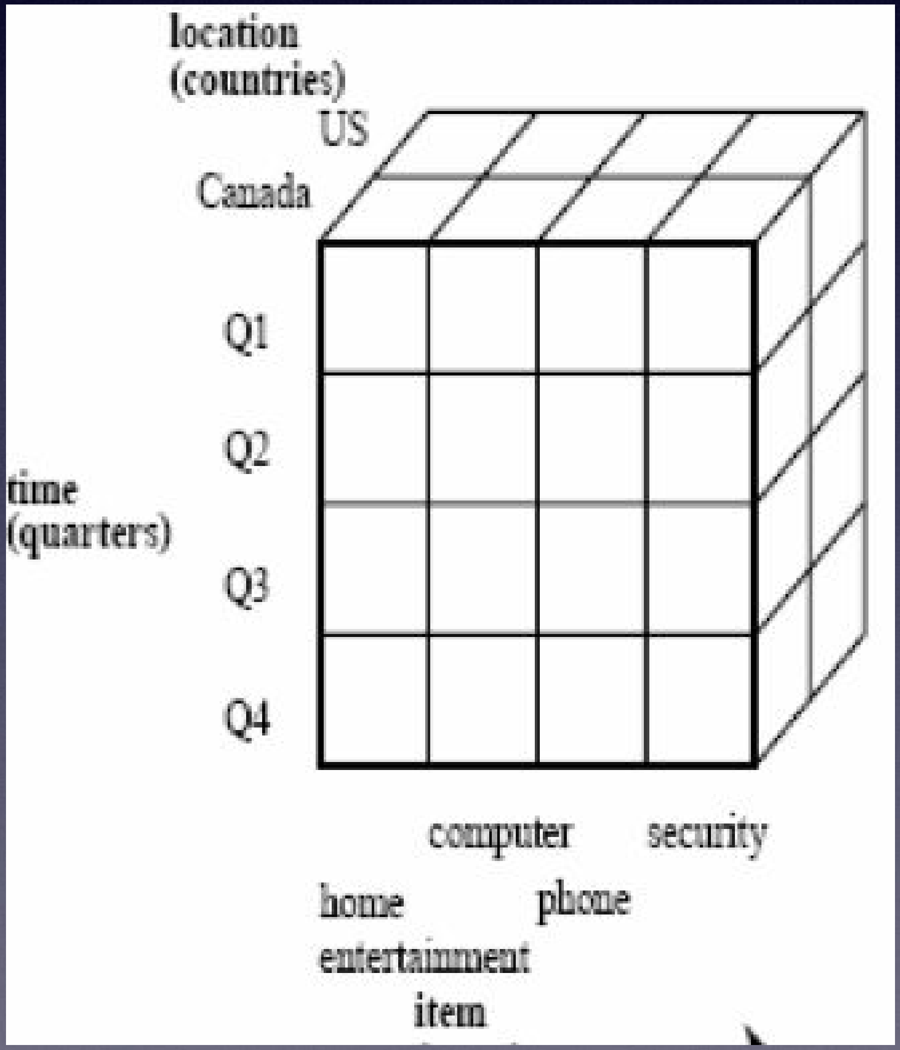
partial order hierarchy

Example of Operations in Cube

- 1. Roll up (drill up) —summarize data

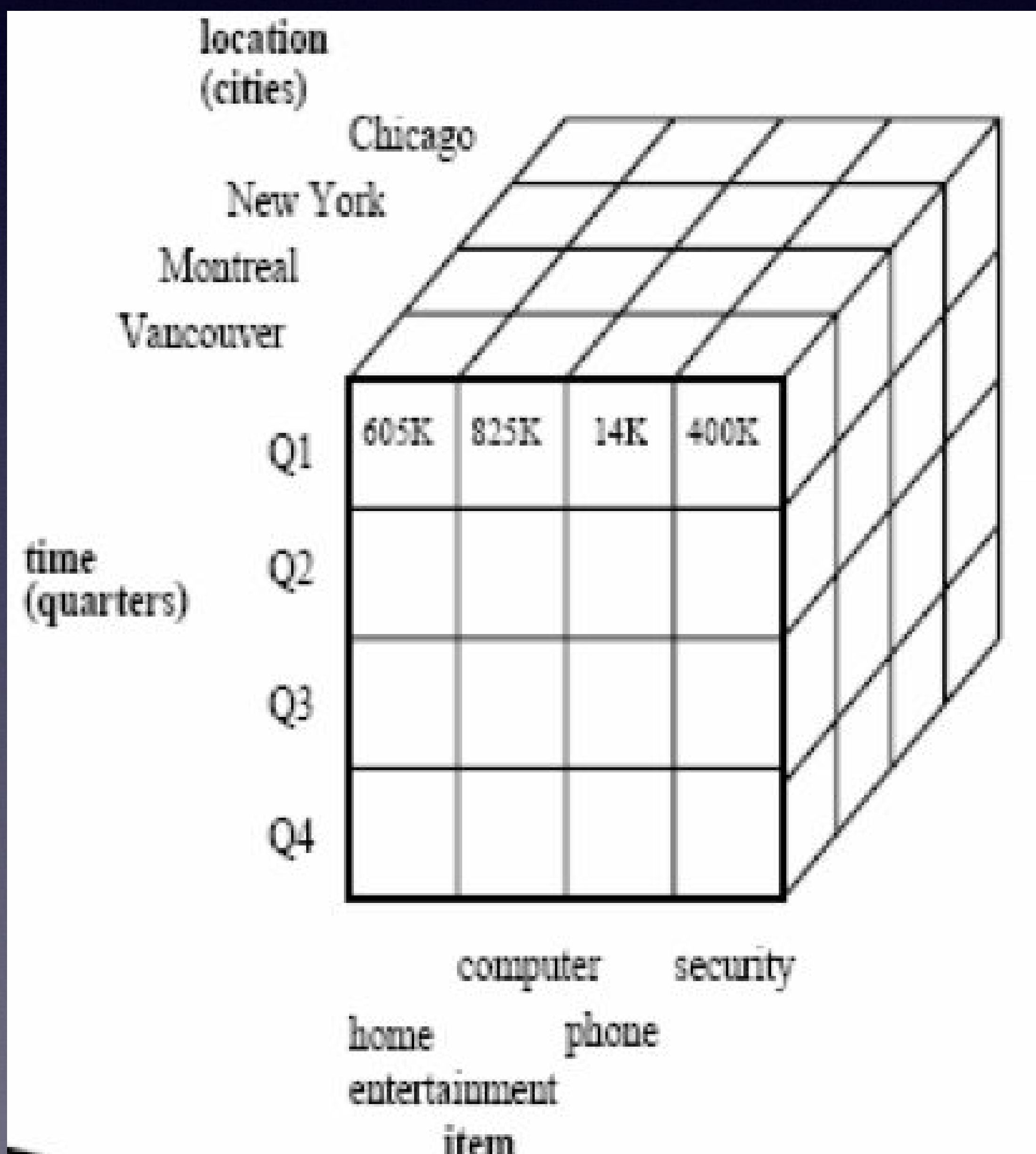


location

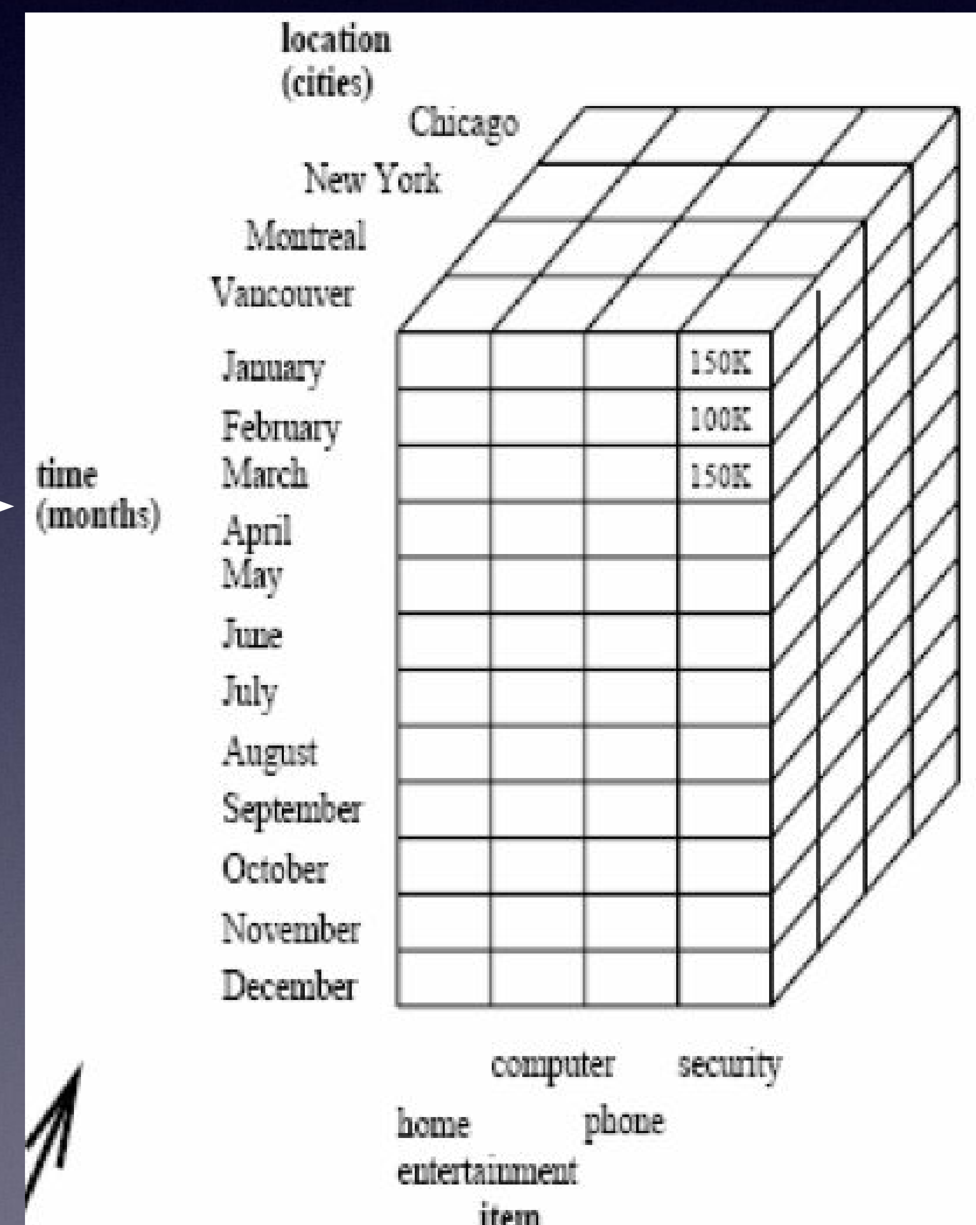


Example of Operations in Cube

- 2. Drill down (droll down) —reverse of roll up

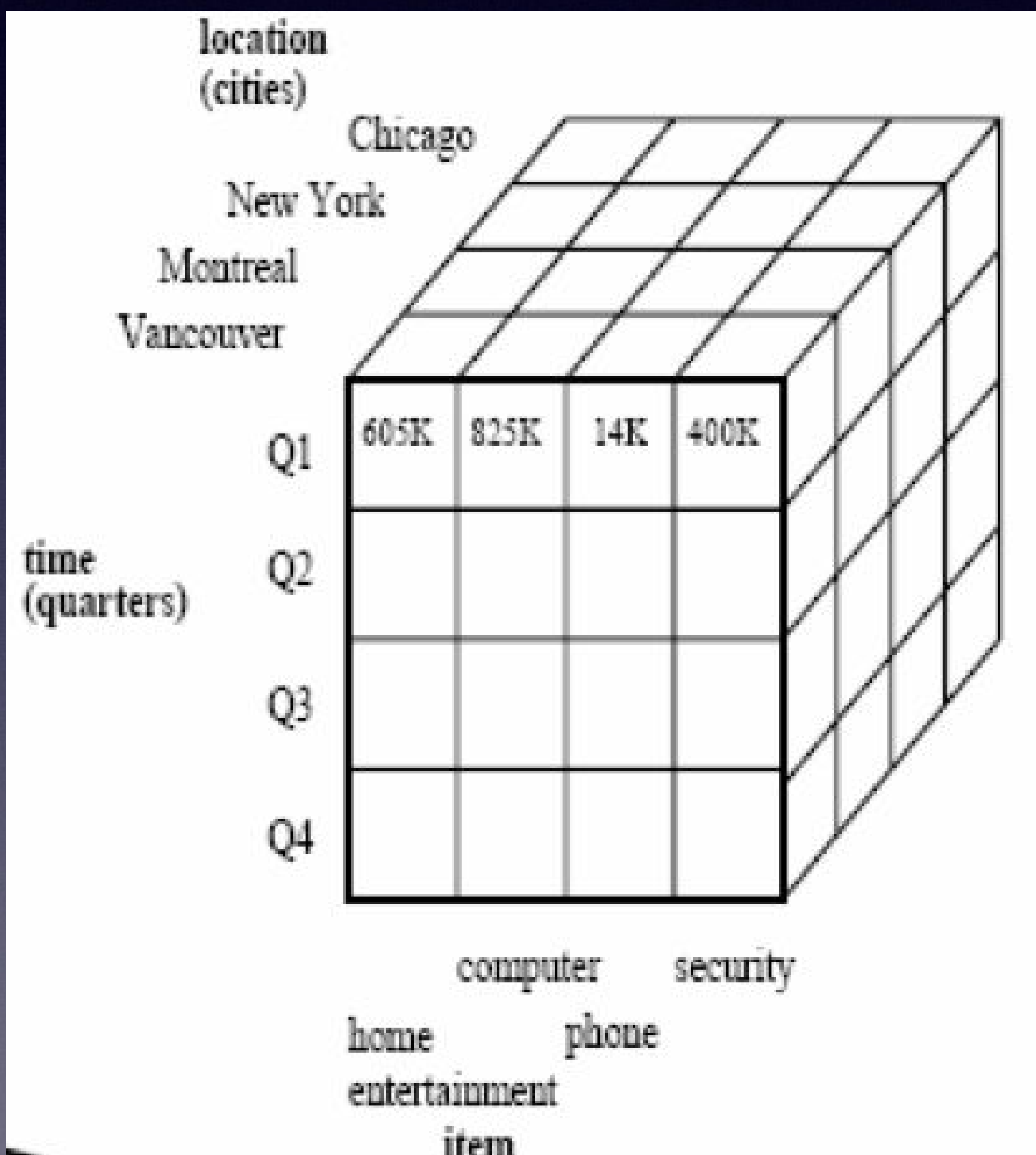


time

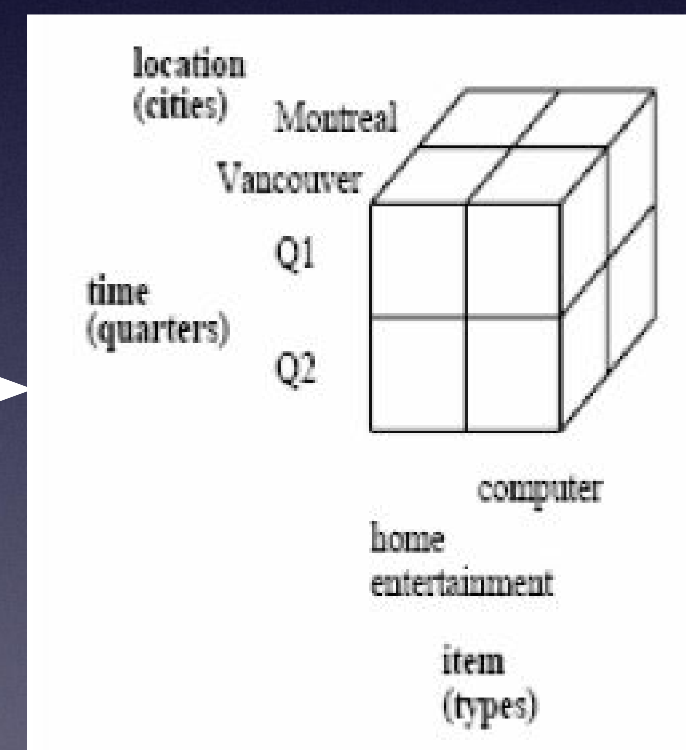


Example of Operations in Cube

- 3. Dice (project and select)

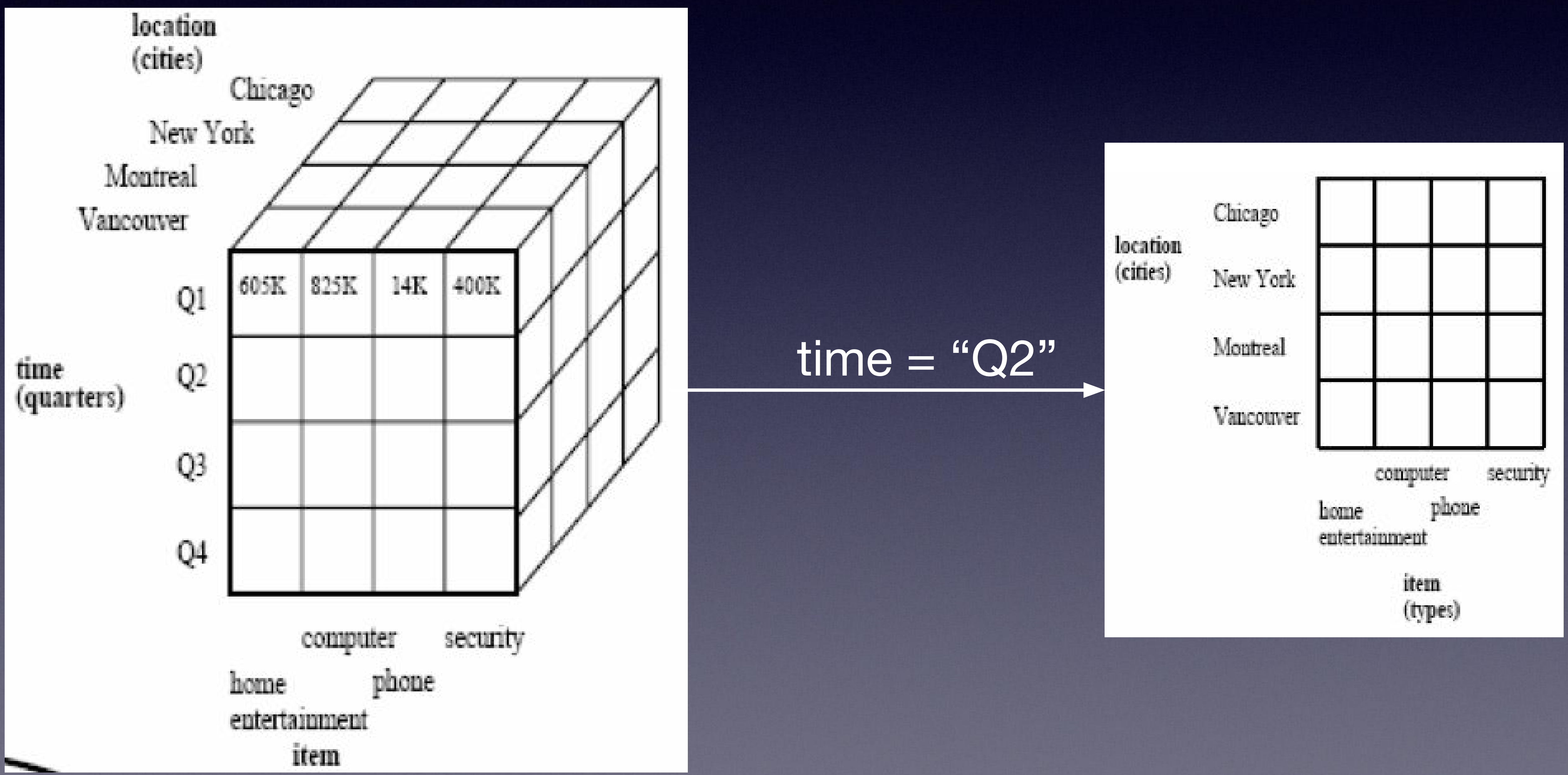


location
time
item



Example of Operations in Cube

- 4. Slice (Select)



Outline

- 1. Introduction and Definition
- 2. Multi-dimensional Data Warehouse Model
- 3. Data Mining Query Language (DMQL)
- 4. Introduce to OLAP and its Architecture
- 5. Data Warehouse Architecture and Design
- 6. Data Warehouse Implementation (Integration)

View of Data Warehouse

Based on business requirements:

1. Top-down View	Allows Selection of the relevant information necessary for the data warehouse.
2. Data Source View	It exposes the information being captured, stored, and managed by operational systems. Example: ER model.
3. Data Warehouse View	Fact tables and dimensional tables.
4. Business Query View	It sees the perspective of data in the warehouse from the view of end-user.

Data Warehouse Design

- 1. Top-down: Overall Design and Planing
- 2. Bottom-up: WaterFall or Spiral
- 3. Design Process:
    1. choose business process model
    2. choose the atomic level of data of the business process
    3. choose the dimensions for each fact table record
    4. choose the measure that will populate each fact table

Data Warehouse Models

- 1. Enterprise Warehouse:

  Collect All of the information about subjects spanning the entire organization

- 2. Data Mart:

  a subset of corporate-wide data that is of value to a specific groups of users. Independent vs. dependent

- 3. Virtual Warehouse:

  A set of views over operational database (materialized)

OLAP Server Architectures

1. Relational OLAP	Use Relational DBMS as Backend, store manage warehouse data OLAP middle ware support, greater scalability.
2. Multidimensional OLAP	Array-based storage (sparse matrix techniques) Pre-computed data and fast indexing
3. Hybrid OLAP	Flexibility: relational or array Support SQL queries: Star or Snowflake schema
4. Data Storage Methods	a. Base Cuboid data: base fact table b. Aggregate data: base fact table, or Separate summary fact tables

OLAP DW Usages and Advantages

Three kinds of Usage:

1. Information Processing
2. Analytical Processing
3. Data Mining

Four Advantages:

1. High quality of data in data warehouses
2. Available information processing structure surrounding data warehouses
3. OLAP-based exploratory data analysis
4. On-line selection of data mining functions

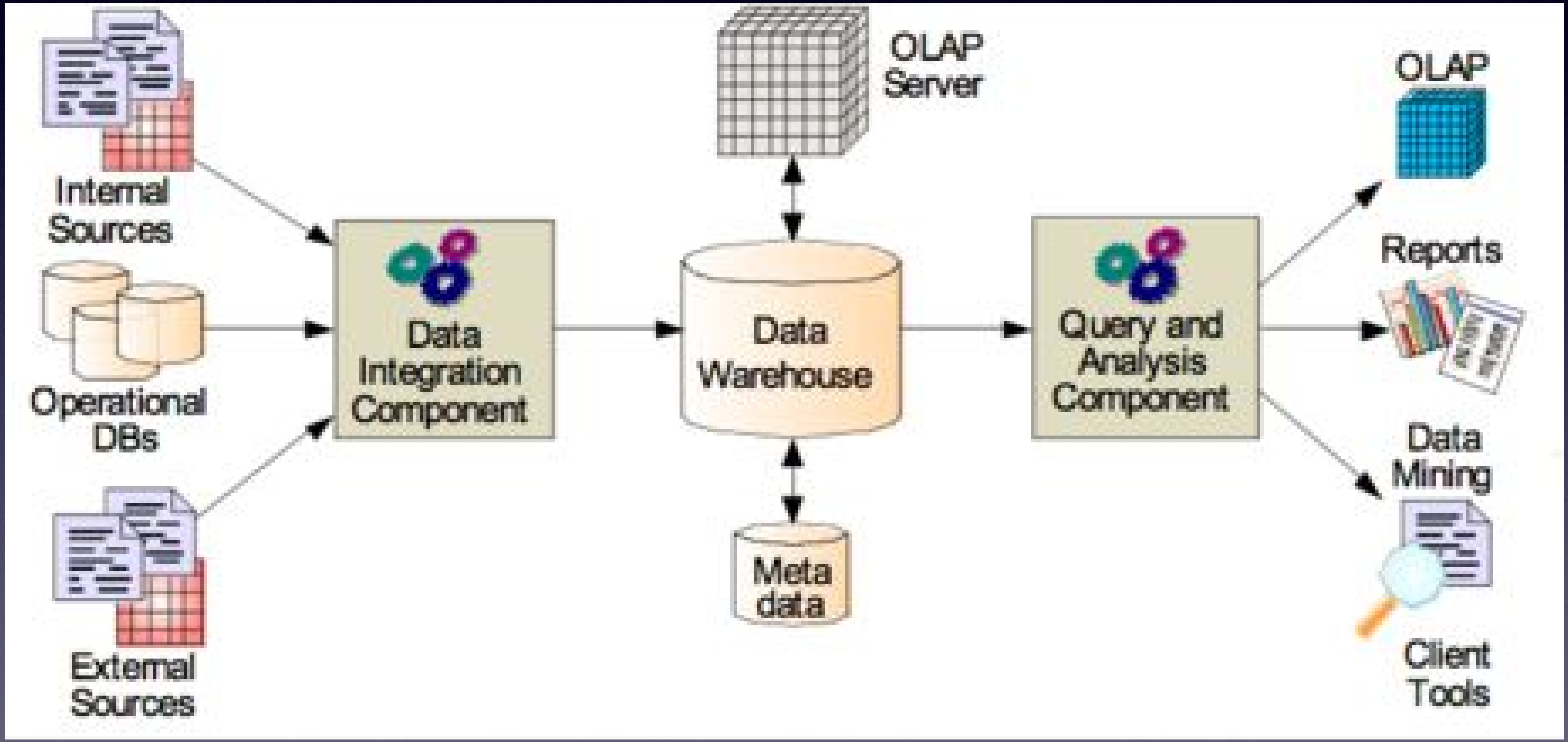
Outline

- 1. Introduction and Definition
- 2. Multi-dimensional Data Warehouse Model
- 3. Data Mining Query Language (DMQL)
- 4. Introduce to OLAP and its Architecture
- 5. Data Warehouse Architecture and Design
- 6. Data Warehouse Implementation (Integration)

Data Warehouse Implementation

- 1. **Monitoring**: sending data from the source
- 2. **Integrating**: loading, cleansing, schema matching…
- 3. **Processing**: cube computation, query processing, indexing

The Importance of integration



Usage-Based Schema Matching

Introduction:

- Problems in Data Integration:
  - Find correspondences between attributes of two schemas.
  - Proposed tchniques:
    - Schema-based techniques:
      - rely on metadata:
        - same attributes may have different meaning
    - Instance-based techniques:
      - rely on characteristics of data instances:
        - same user has different names in different tables
      - tacked schema matching with opaque attribute names
      - not complete schema

Usage-Based Schema Matching

Introduction:

- New Technique for Schema Matching:
  - Usage-based schema matching
    - Good matching quality
    - Identifies co-occurrence patterns:
      - attributes, relationship types
    - Genetic algorithm:
      - highest-score mappings
    - Opaque attributes and different layouts.

Contributions

- Based on usage of attributes in query log:
  - Tow usage-based matchers:
    - SLUB : Structure-Level Usage-Based matcher
    - ELUB : Element-Level Usage-Based matcher
- Prototype implementation:
  - Employs a genetic algorithm to find highest score mapping
- An extensive experimental study:
  - Effective
  - Accurate

Main Point

- Goal:
  - Exploit similarities in query patterns to match attributes
- Feature extraction:
  - Uses query logs
  - Collects attributes' roles and interrelationships.
- Matching:
  - Examines potential mappings
  - Assigns a score for them
  - Terminates by reporting highest-score

Feature Extraction

- SLUB: Structure-Level Usage-Based matcher
  - Structure-level freatures:
    - An attribute A roles:
      - part of the answer (select clause)
      - filterint role (where or having)
      - grouping role (group by)
      - odering role (oder by)
    - Tow attributes in same query:
      - usage relationship
      - four possible roles results in 16 different possible relationships
      - 16 graphs with weights on endges (frequency of occurrence)

Usage relationship type
select-select
where-select
select-where
where-where
orderby-select
select-orderby
groupby-groupby
groupby-select
select-groupby
orderby-where
groupby-where
where-orderby
where-groupby
orderby-orderby
orderby-groupby
groupby-orderby

Feature Extraction

- Identification process
  - SPJGO:
    - relationship depends on two clauses
  - SPJGO-UEI:
    - relationships are identified separately
    - attributes in one subquery don't affect result of another
  - SPJGO-N:
    - relationships identifed separately for each block
      - outer query and inner subquery
    - more relationships identified between attributes in differernt blocks
      - inner subquery may be a filter of outer query (if it's in where or from clause of outer query)
      - they are considered to be related to all outer query attributes

```
Q1: select I_TITLE from Item, Author
    where I_A_ID=A_ID and A_LNAME='Gray'
Q2: select I_TITLE from Item
    where I_A_ID in(select A_ID from Author
                    where A_LNAME='Gray')
Q3: select I_TITLE
    from Item, (select A_ID from Author
                where A_LNAME='Gray')
    where I_A_ID=A_ID
```

```
Q4a: @list = select A_ID from Author
             where A_LNAME='Gray'
Q4b: select I_TITLE from Item
     where I_A_ID in @list
```

TABLE I. CONTRIBUTION OF THE QUERIES OF EXAMPLE 4.1 TO THE SELECT-WHERE RELATIONSHIP TYPE

select-where	A_ID	A_LNAME	I_A_ID	I_TITLE
A_ID	-	*Q2,Q3,Q4*	-	-
A_LNAME	-	-	-	-
I_A_ID	-	-	-	-
I_TITLE	*Q1,Q2,Q3*	*Q1,Q2,Q3*	*Q1,Q2,Q3,Q4*	-

Matching

- Genetic Algorithm:
  - Selection : generate population
    - S1 = {1,2,3,4,5,6,7}
    - S2 = {1,2,3,4,5,6,7}
    - Assume a population with 4: 011101, 101011,011100,111001
    - Fitness function: $f(x1,x2) = x_1^2 + x_2^2$

Individual No	Population(0)	x1 x2	Fitness	%	Selection time Random	Results
1	011101	3 5	34	0.24	1	011101
2	101011	5 3	34	0.24	1	111001
3	011100	3 4	25	0.17	0	101011
4	111001	7 1	50	0.35	2	111001
sum			143	1		

Matching

- Genetic Algorithm:
  - Genetic Operators:
    - Crossover:
      - Pair : random
      - Crossover position: random
      - Crossover part of gene

Individual No	Results	Pairs	Crossover position (Random)	Results
1	011101	1-2	2	011001
2	101011			111101
3	011100	3-4	4	101001
4	111001			111011
...				

Matching

- Genetic Algorithm:
  - Genetic Operators:
    - Mutation:
      - Mutation position: random
      - Change the bit with a probability

Individual No	Results	Mutation Position	Mutation Result	Population(1)
1	011001	4	011101	011101
2	111101	5	111111	111111
3	101001	2	111001	111001
4	111011	6	111010	111010
...				

- Compare Population(0) and Population(1)

Individual No	Population(0)	x1 x2	Fitness	%	Selection time Random	Results
1	011101	3 5	34	0.24	1	011101
2	101011	5 3	34	0.24	1	111001
3	011100	3 4	25	0.17	0	101011
4	111001	7 1	50	0.35	2	111001
sum			143	1		

⇩

Individual No	Population(1)	x1 x2	Fitness	%
1	011101	3 5	34	0.14
2	111111	7 7	98 ↑	0.42 ↑
3	111001	7 1	50	0.21
4	111010	7 2	53	0.23
sum			235 ↑	

Matching

- How to implement in Usage-Based Schema Matching
  - Compare 16 graphs of each table one by one to generate potentially mached attributes of two schemas
  - Fixed number of interation
  - Make crossover and mutation to find the highest mapping from possible ones.

- Fitness function is indentified to calculate the similarities of two schemas

Algorithm 1: Generate_New_Mapping (m)

M_i: set of matched S_i attributes, $i \in [1,2]$
L_i: set of S_2 attributes permitted to match x_i, $i \in [1,n']$
$R_{1,i}$: sum of edge weights from x_i to all $x_j \in M_1$ averaged over the 16 feature graphs of S_1, $i,j \in [1,n']$, $i \neq j$
$R_{2,i}$: sum of edge weights from y_i to all $y_j \in M_2$ averaged over the 16 feature graphs of S_2, $i,j \in [1,n']$, $i \neq j$

$M_1=\{\}$; $M_2=\{\}$;
for each iteration t
  if $|M_1|=n'$
    return m;
  Find an unmatched S_1 attribute x_{i_t} and an unmatched S_2 attribute y_{j_t} such that $y_{j_t} \in L_{i_t}$, $R_{1,i_t}>0$, $R_{2,j_t}>0$, $|R_{1,i_t}-R_{2,j_t}| \leq |R_{1,u}-R_{2,v}|$, $u \neq i_t$, $v \neq j_t$, $x_u \notin M_1$, $y_v \notin M_2$;
  if such pair (x_{i_t}, y_{j_t}) does not exist
    Let x_{i_t} be any random unmatched S_1 attribute, y_{j_t} be any random unmatched S_2 attribute, $y_{j_t} \in L_{i_t}$;
  Let $m(i_t)=j_t$;
  Add x_{i_t} to M_1;
  Add y_{j_t} to M_2;
  Remove y_{j_t} from L_u, $u \neq i_t$;

Algorithm 2: Make_Crossover (m_1, m_2)

c_i : the i^{th} child mapping to be generated, $i \in [1,2]$

Copy m_1 into c_1;
Randomly divide c_1 into two parts;
Keep the first part of c_1 unchanged;
For the second part, keep the matches for the constrained S_1 attributes unchanged;
Reorder the matching S_2 attributes for the unconstrained S_1 attributes in the second part of c_1 to follow the ordering of m_2;
Generate c_2 in the same way as c_1 after switching the roles of m_1 and m_2;
return $\{c_1, c_2\}$;

Algorithm 3: Make_Mutation (m)

c : the child mapping to be generated

Copy m into c;
Pick two random unconstrained S_1 attributes x_i and x_j;
Swap $c(x_i)$ and $c(x_j)$;
return c;

Conclusion

- A new schema matching: usage-based
- Find correspondences between attributes of two schemas with high accuracy
- For now focusing on relational schemas
- Futher more, tring to apply in an XML context and other schema

References

- J. Madhavan, P. Bernstein, A. Doan, and A. Halevy. Corpusbased schema matching. In ICDE, 2005.
- https://en.wikipedia.org/wiki/Genetic_algorithm
- http://blog.csdn.net/b2b160/article/details/4680853/
- http://cedric.cnam.fr/workshops/caise03/InvitedTalk.pdf
- J. Madhavan, P. Bernstein, and E. Rahm. Generic schema matching with Cupid. In VLDB, 2001.
- R. Warren and F. Tompa. Multi-colunm substring matching for database schema translation. In VLDB, 2006.