Module 2: DW & OLAP
Chapter 4

CSC 533 Data Mining

Dr. Peng Kang

Dept. of Computer Science, UIS

Logistics

- Quiz on Ch2 and 3 due Feb. 19, 11:59 PM
- Ex2-1 released and due Feb. 25, 11:59 PM

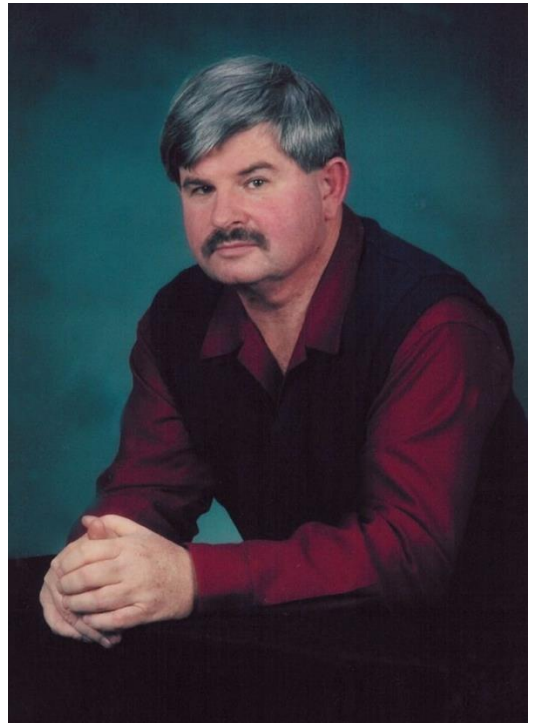
4.1 Data Warehouse: Basic Concepts

Recall: What is a Data Warehouse?

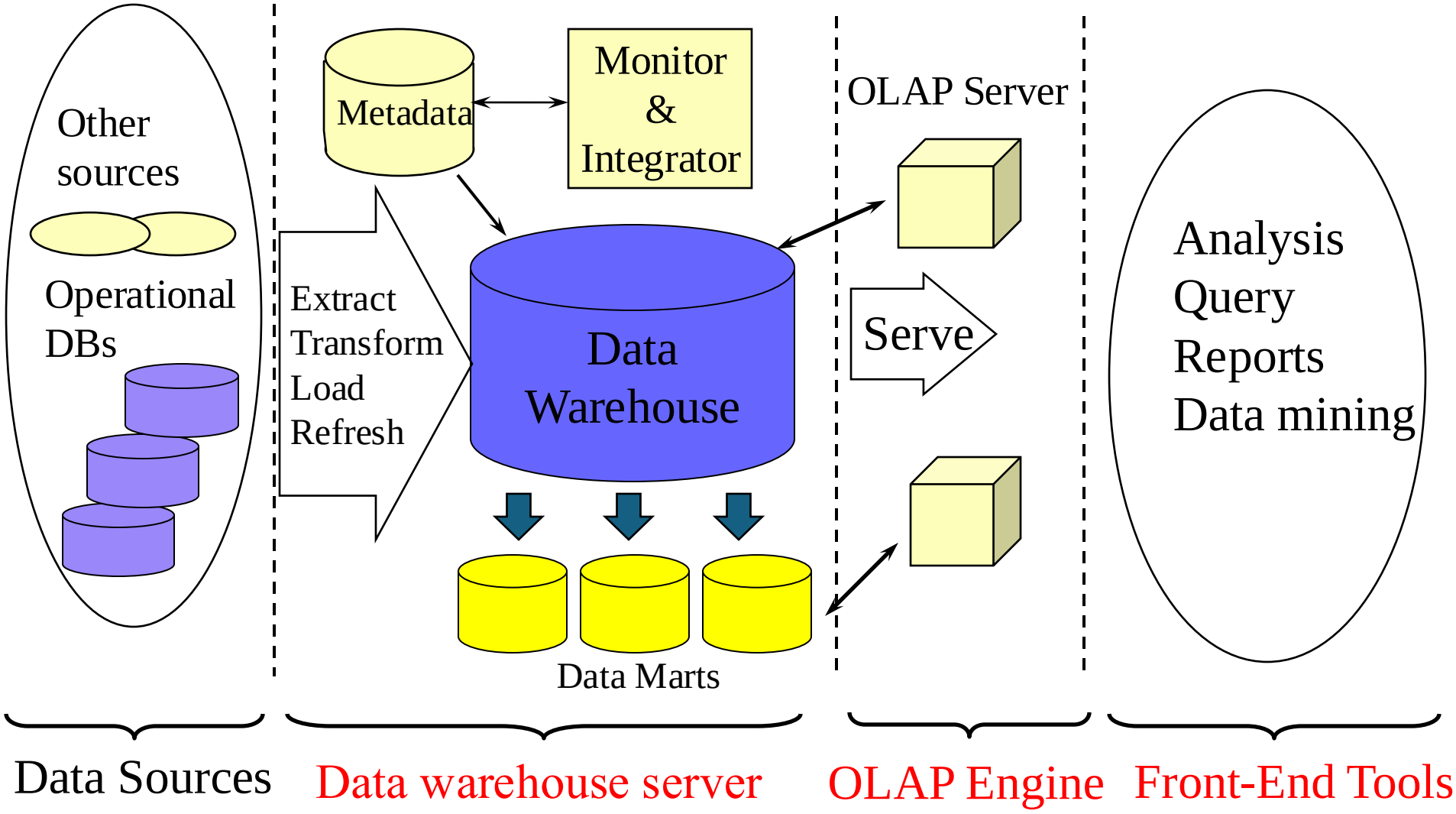
- Defined in many different ways, but not rigorously
  - A decision support database that is maintained separately from the organization's operational database
  - Support information processing by providing a solid platform of consolidated, historical data for analysis.
- Loosely speaking, data warehouse is a specialized database
  - Traditional operational database: used for day-to-day operations
  - Data warehouse: designed for analysis and reporting.
- Data warehousing:
  - The process of constructing and using data warehouses

Recall: What is a Data Warehouse?

- “A 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

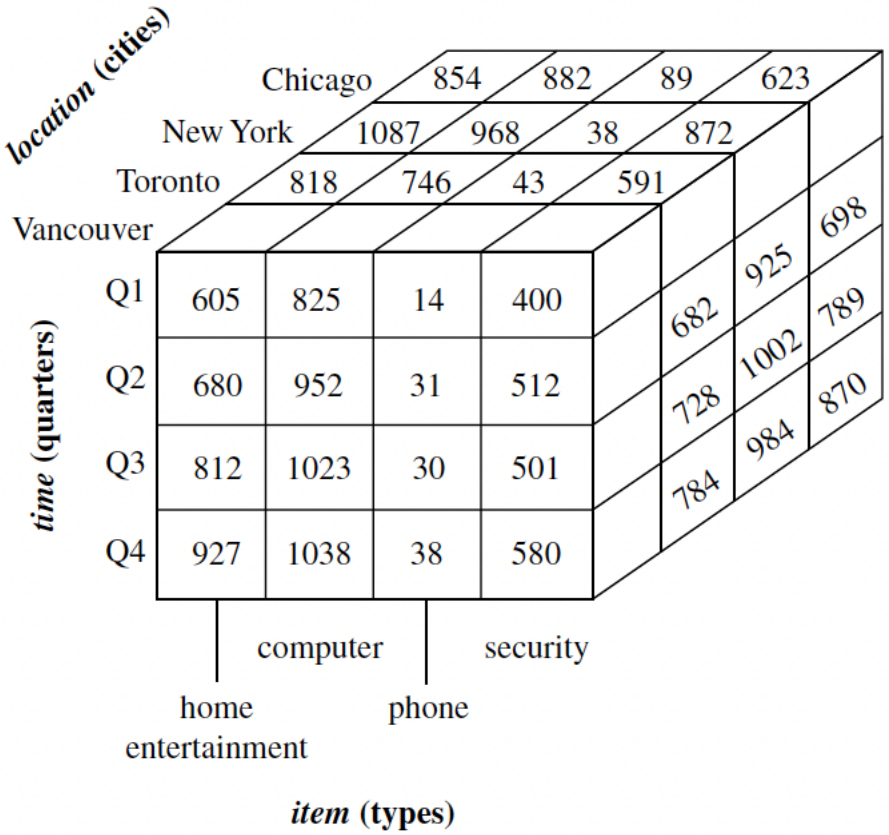


Recall: Data Warehouse: A Multitiered Architecture



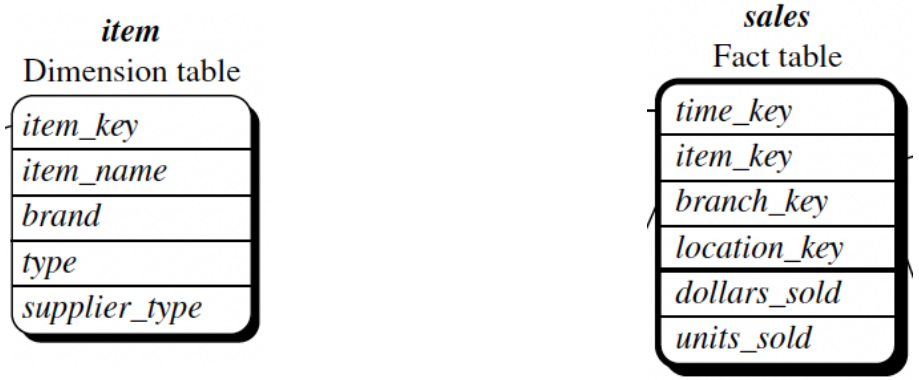
4.2 Data Warehouse Modeling: Data Cube

Recall: Data Cube is n-dimensional

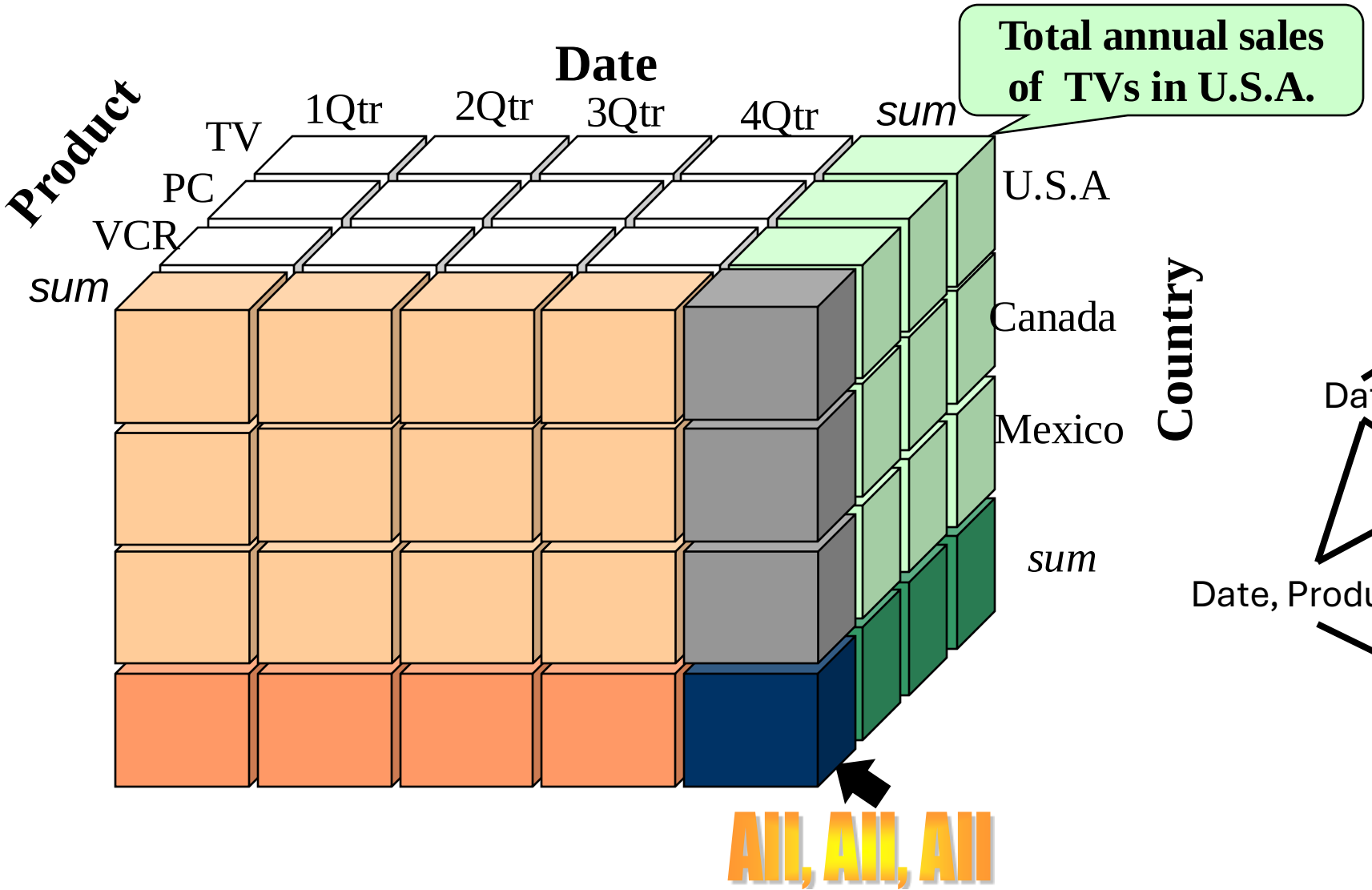


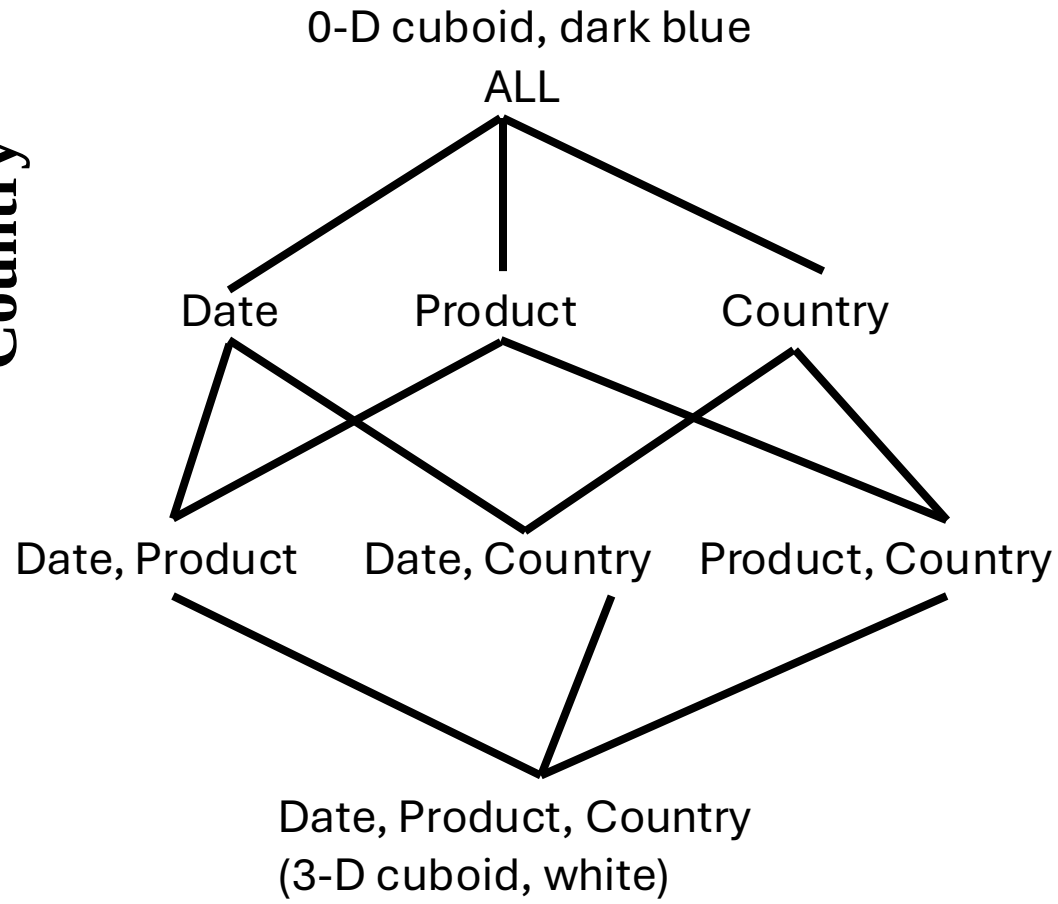
Recall: Data Cube: A Multidimensional Data Model

- 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



Recall: Cuboid and Data Cube

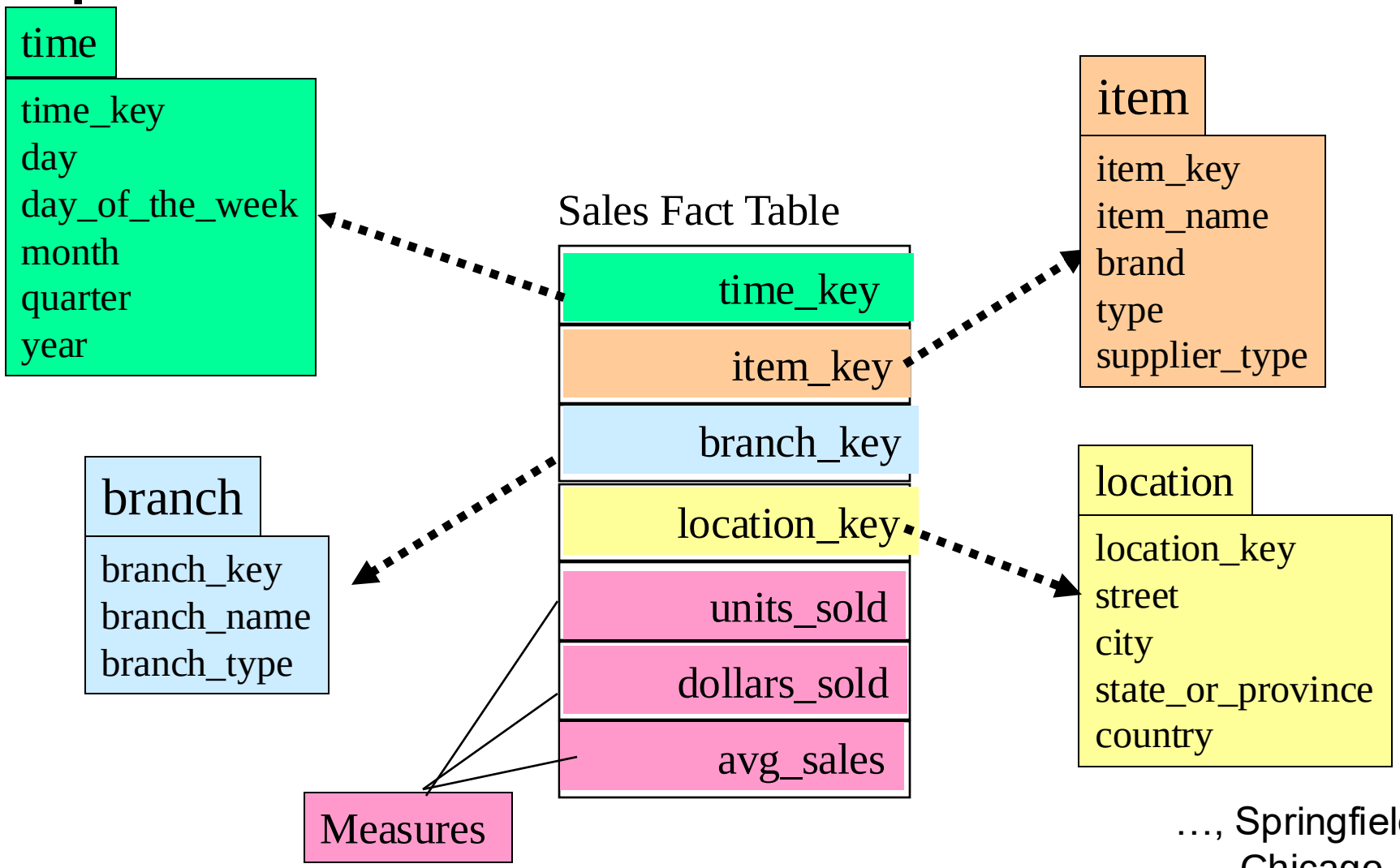




Recall: Stars, Snowflakes, and Fact Constellations: Schemas for Multidimensional Data Models

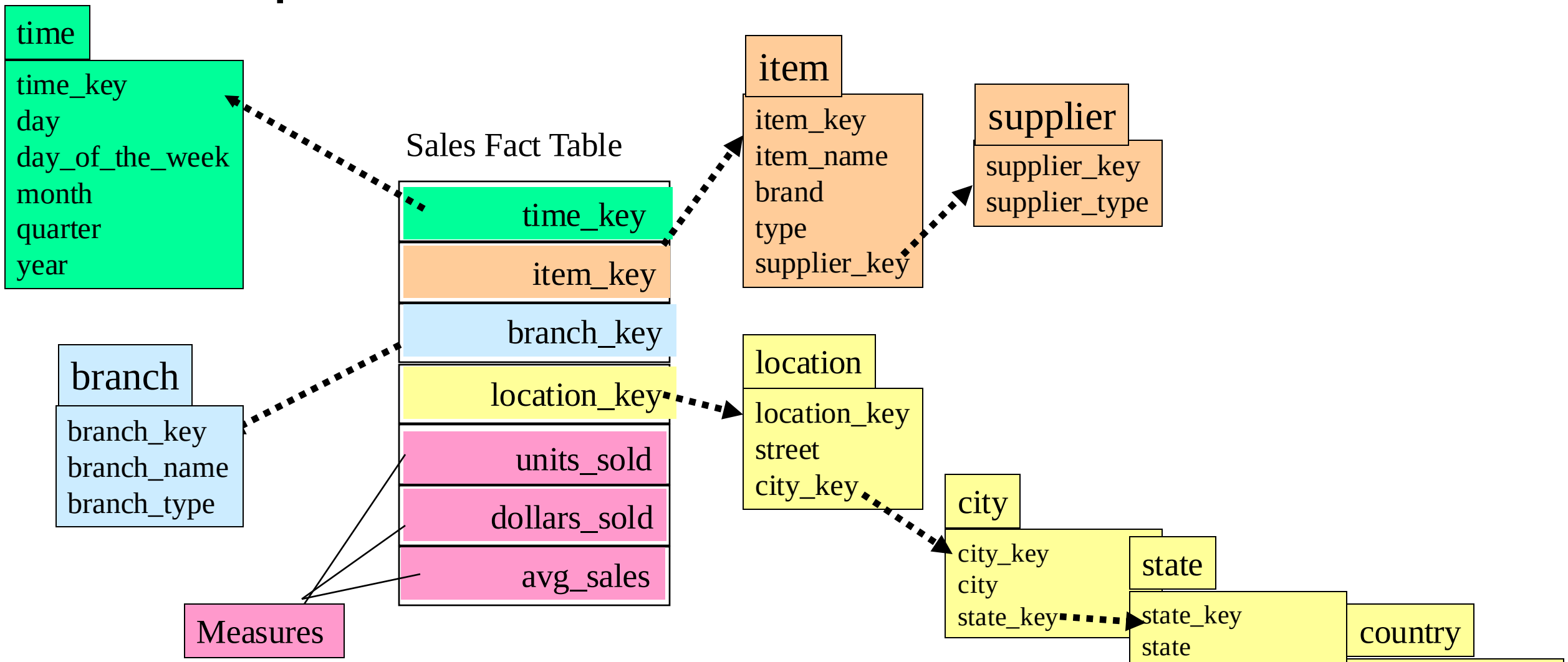
- Modeling data warehouses: dimensions & measures
  - Star schema
  - Snowflake schema
  - Fact constellations (galaxy schema)

Example of **Star Schema**

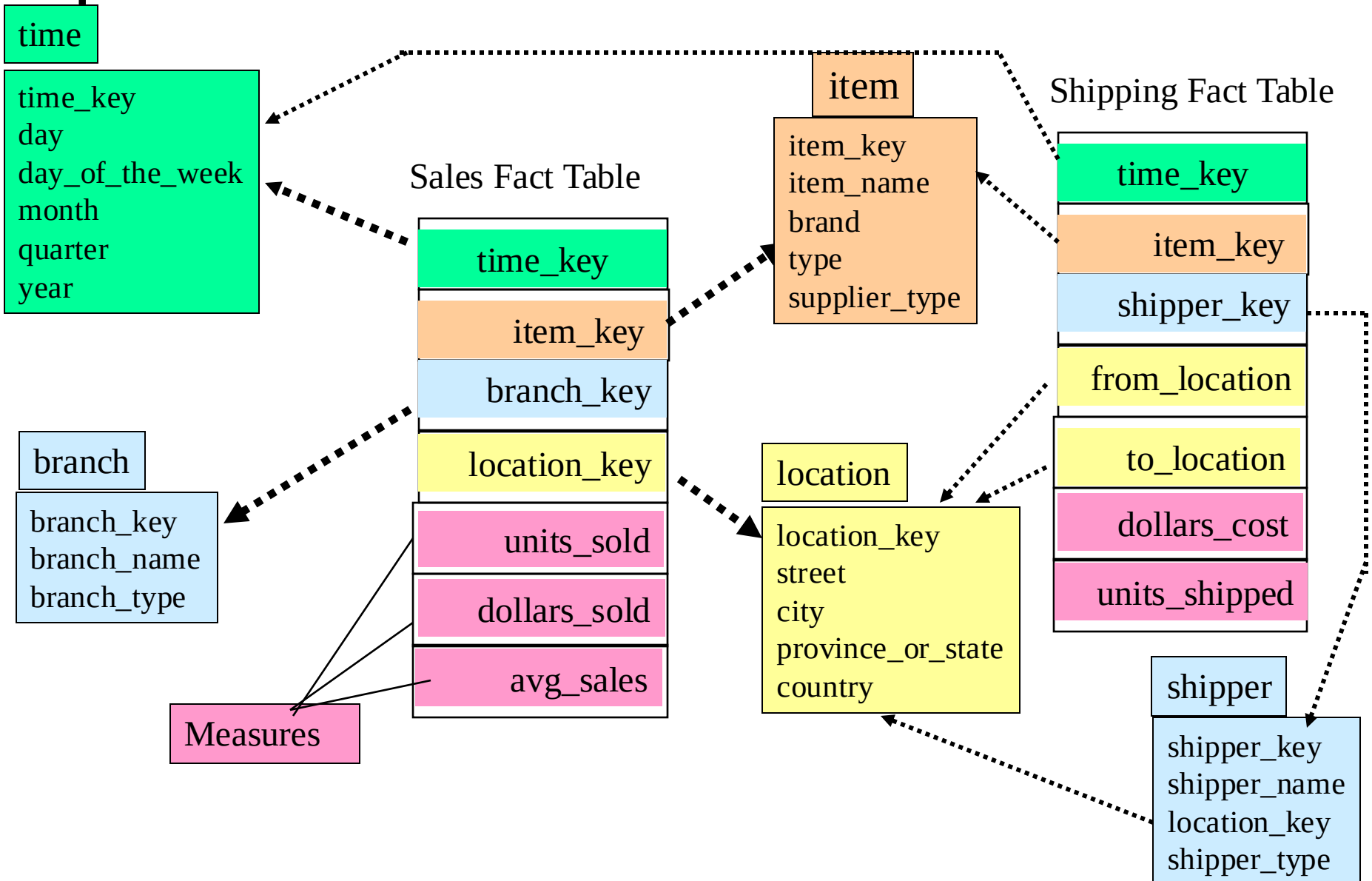


…, Springfield, IL, USA
...., Chicago, IL, USA
Redundancy

Example of **Snowflake Schema**

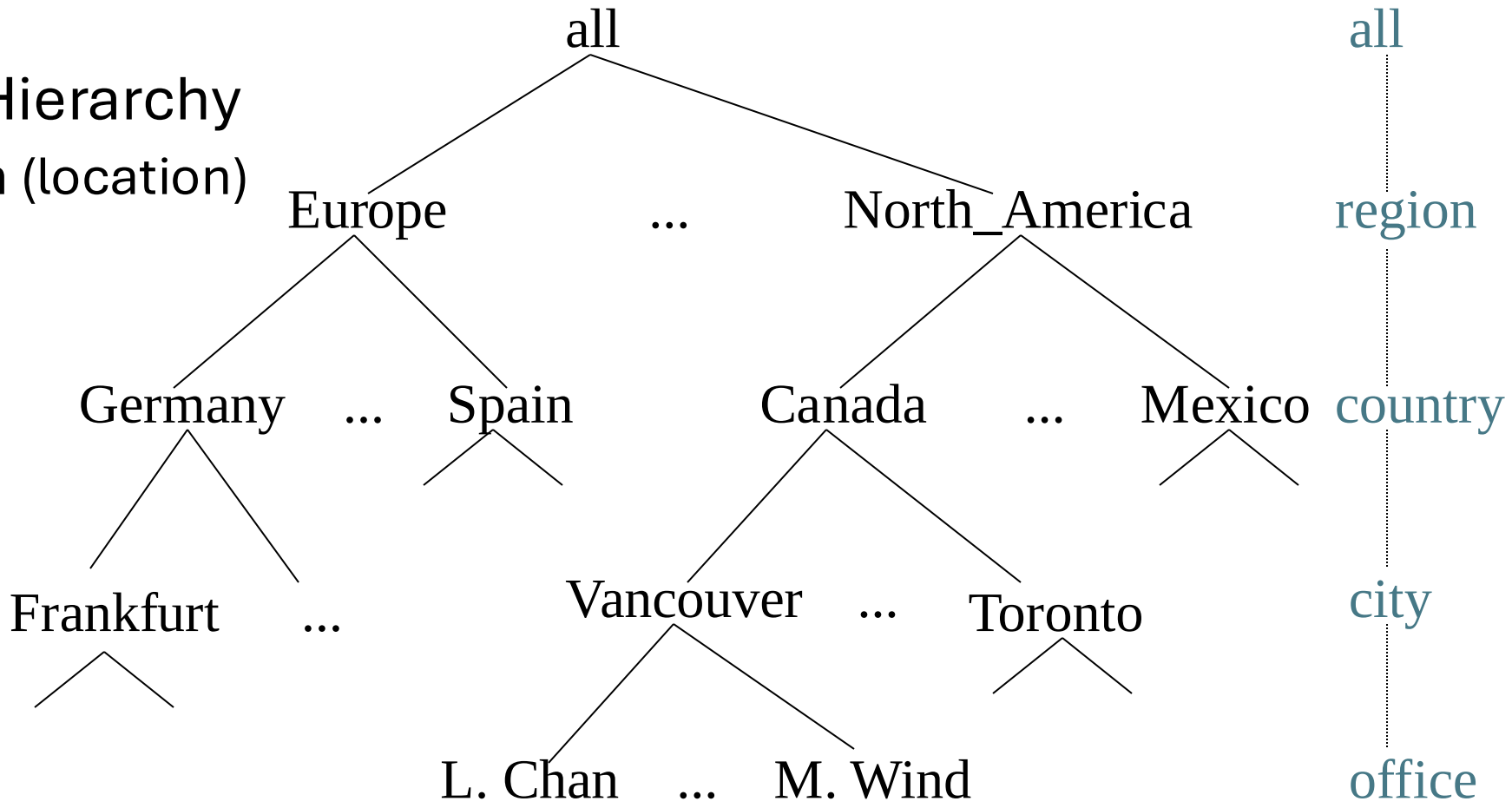


Example of **Fact Constellation**

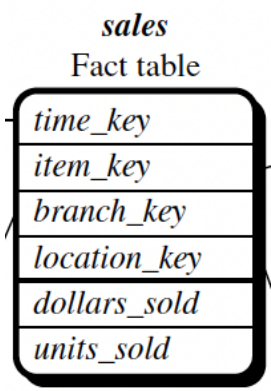


Recall: Dimensions: The Role of Concept Hierarchies

- A Concept Hierarchy
  - Dimension (location)

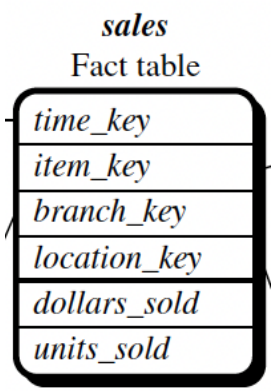


Recall: 4.2.4 Measures: Their Categorization and Computation



- 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()
    - Sum(A1) + Sum(A2) = Sum ([A1, A2])
- 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()
    - Avg() = Sum()/Count()

4.2.4 Measures: Their Categorization and Computation

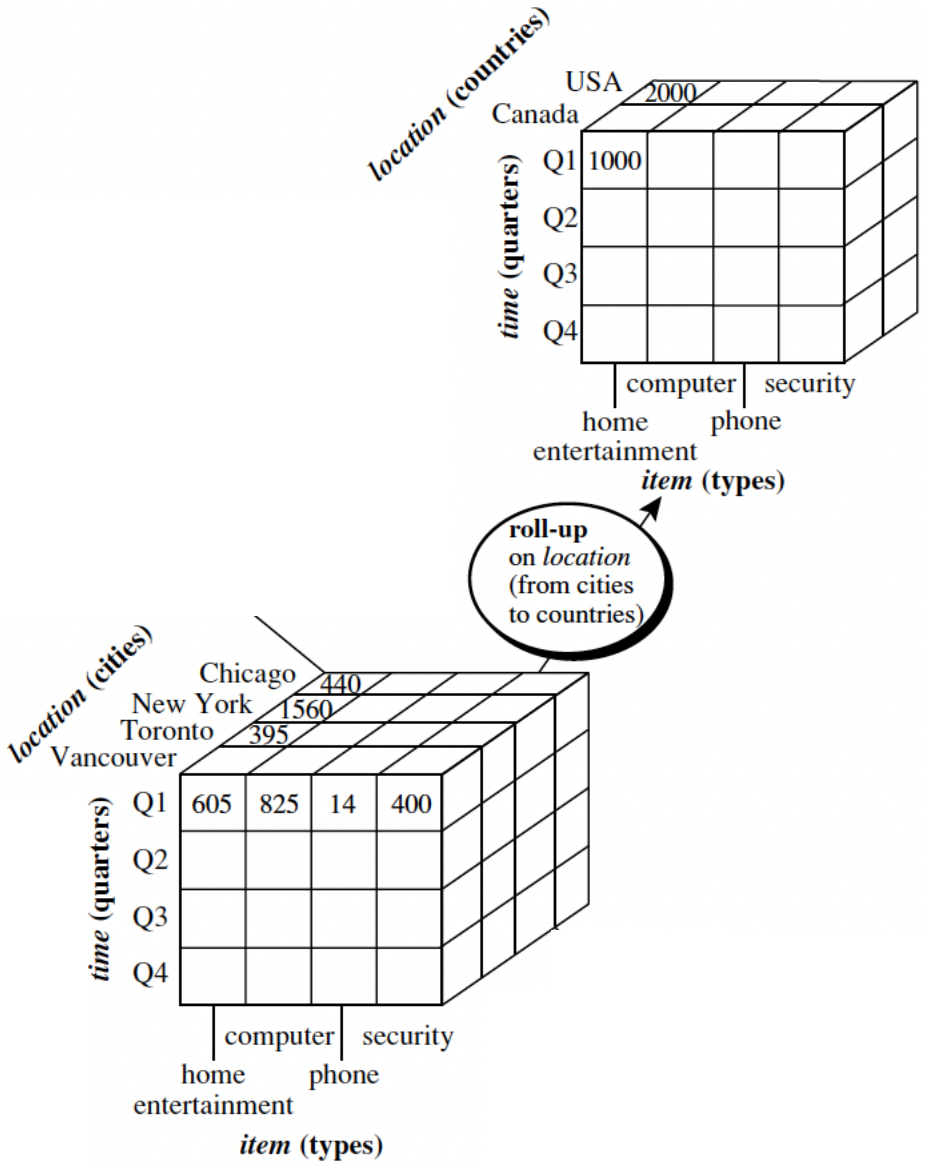


- Holistic: the result requires access to the full dataset and cannot be computed from partial aggregates
    - E.g., median(), mode(), rank()
    - Median(Median(A1), median(A2)) != Median([A1, A2])

4.2.5 Typical OLAP Operations

- Roll up (drill-up): summarize data
  - by climbing up hierarchy or by dimension reduction

4.2.5 Roll up



all

region

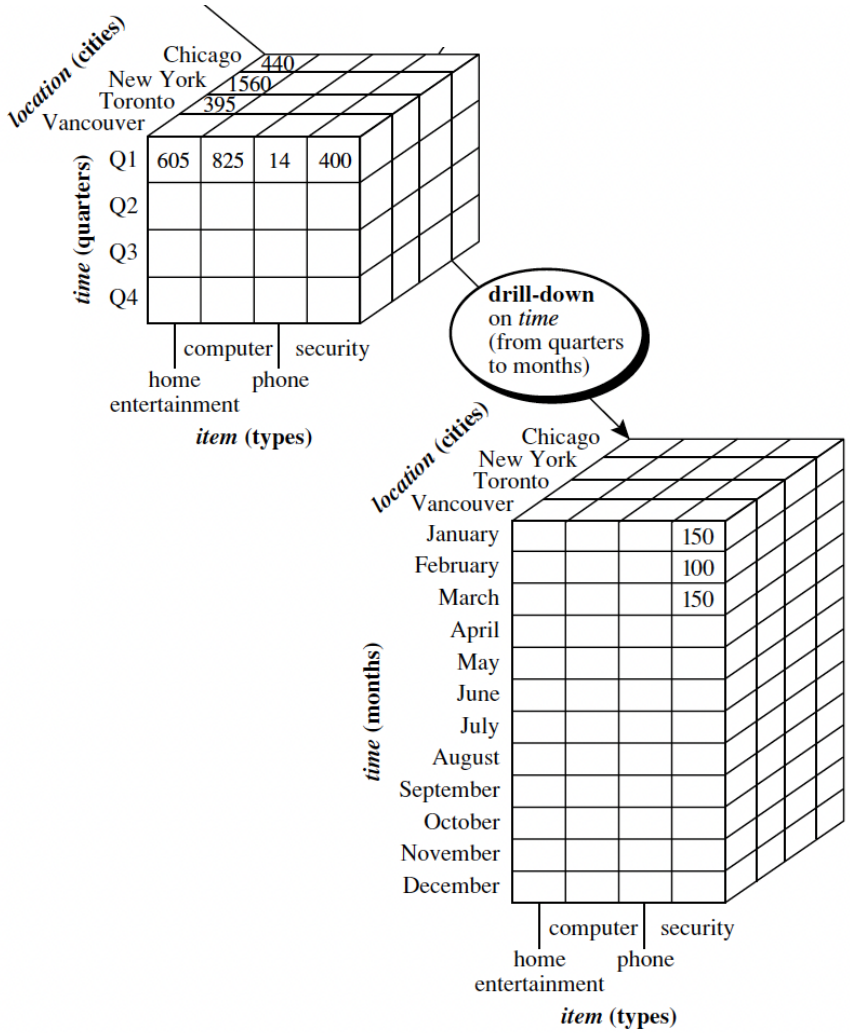
country

city

4.2.5 Typical OLAP Operations

- Drill down (roll down): reverse of roll-up
  - from higher level summary to lower level summary or detailed data, or introducing new dimensions

4.2.5 Drill down



Year

Quarter

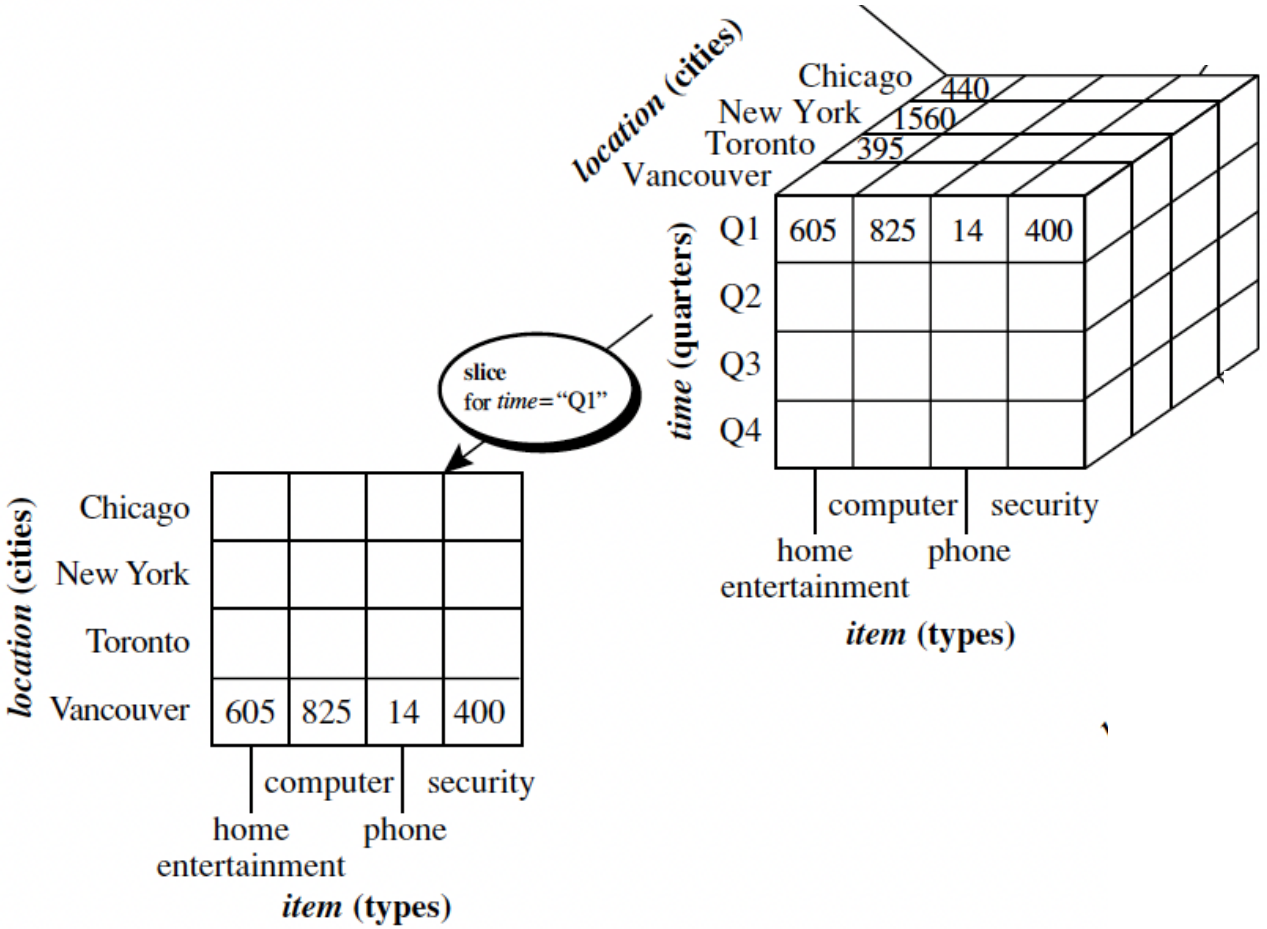
Month

Day

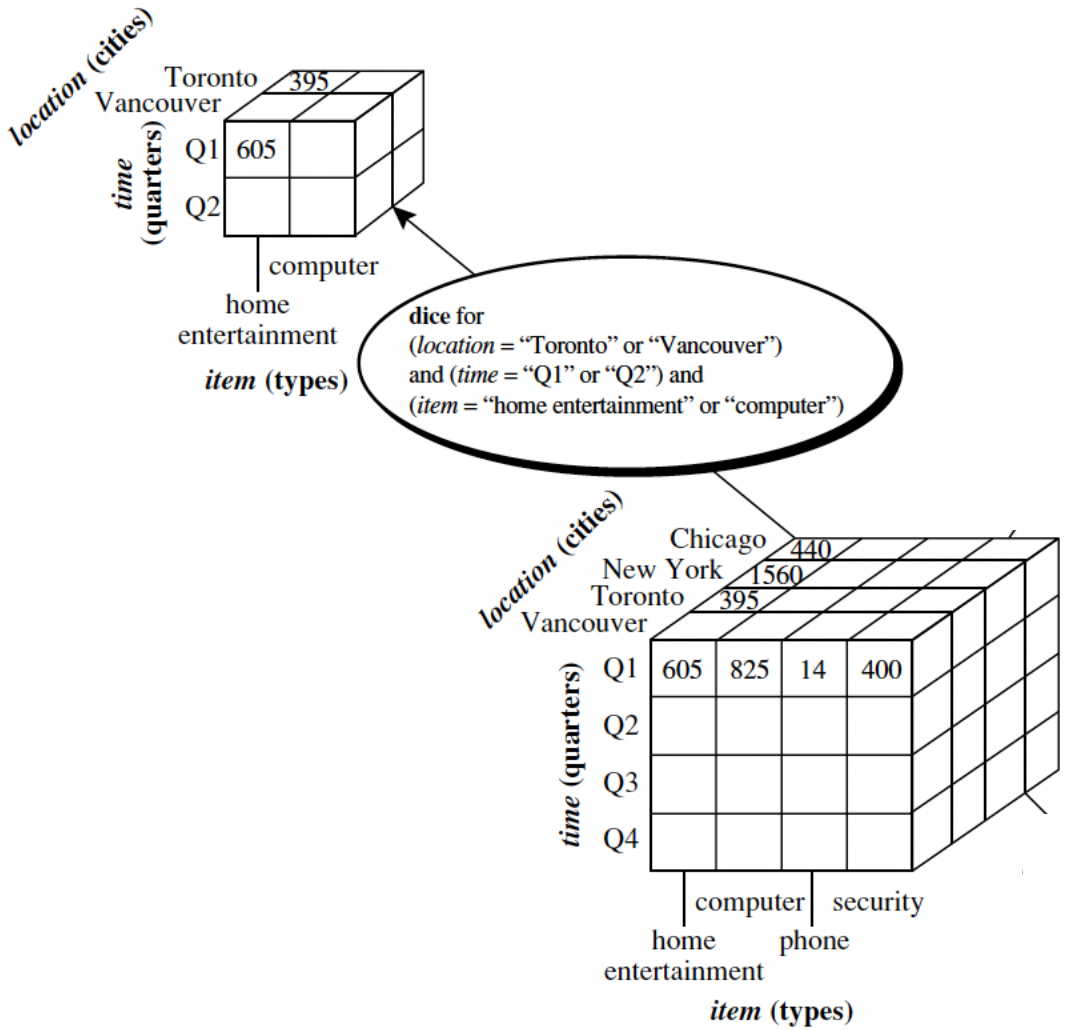
4.2.5 Typical OLAP Operations

- Slice and dice: project and select

4.2.5 Slice


location (cities)
Chicago
New York
Toronto
Vancouver
440
1560
395
time (quarters)
Q1
Q2
Q3
Q4
605
825
14
400
slice
for time="Q1"
location (cities)
Chicago
New York
Toronto
Vancouver
605
825
14
400
computer
security
home
entertainment
phone
item (types)
computer
security
home
entertainment
phone
item (types)

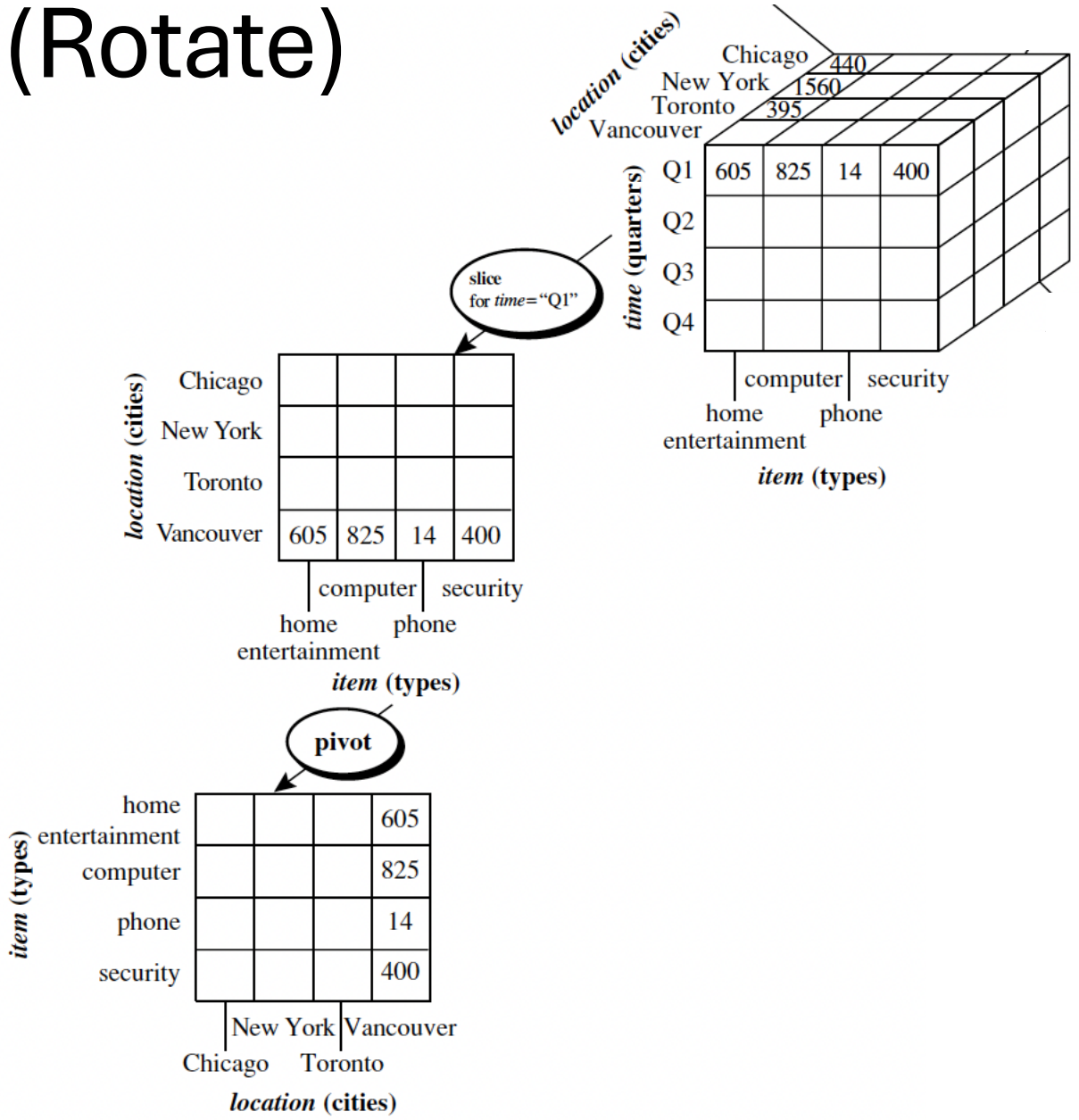
4.2.5 Dice


location (cities)
Toronto
Vancouver
395
time (quarters)
Q1
605
Q2
computer
home entertainment
item (types)
dice for
(location = "Toronto" or "Vancouver")
and (time = "Q1" or "Q2") and
(item = "home entertainment" or "computer")
location (cities)
Chicago
440
New York
1560
Toronto
395
Vancouver
time (quarters)
Q1
605
825
14
400
Q2
Q3
Q4
computer
security
home entertainment
phone
item (types)

4.2.5 Typical OLAP Operations

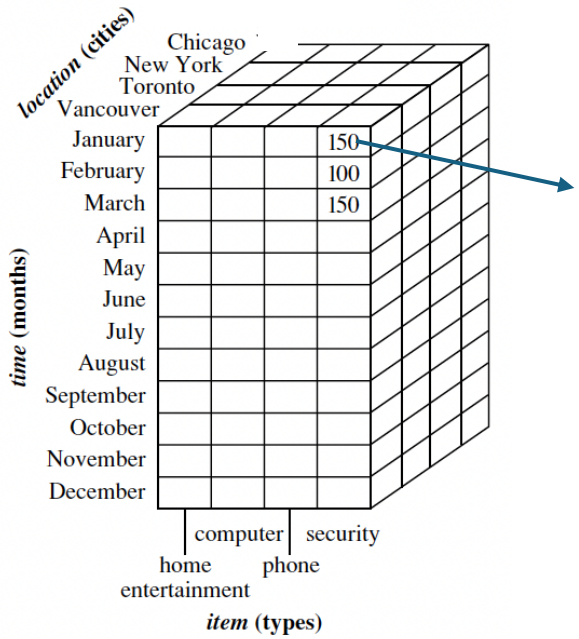
- Pivot (rotate): reorient the cube, visualization, 3D to series of 2D planes

4.2.5 Pivot (Rotate)



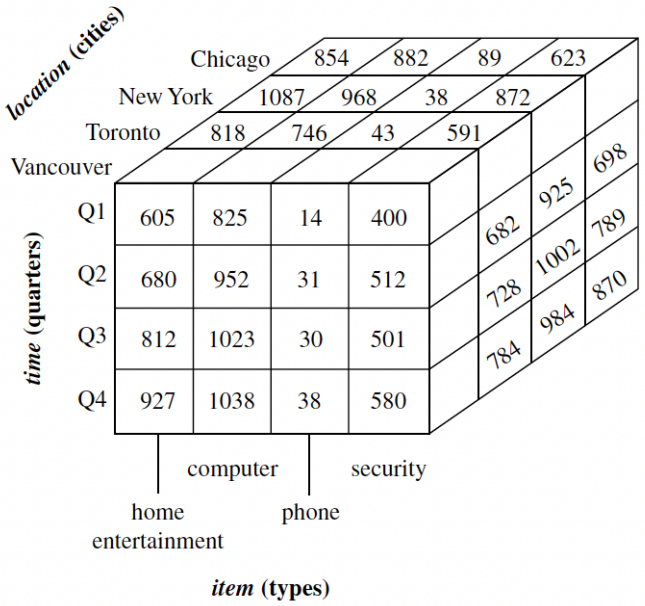
4.2.5 Typical OLAP Operations

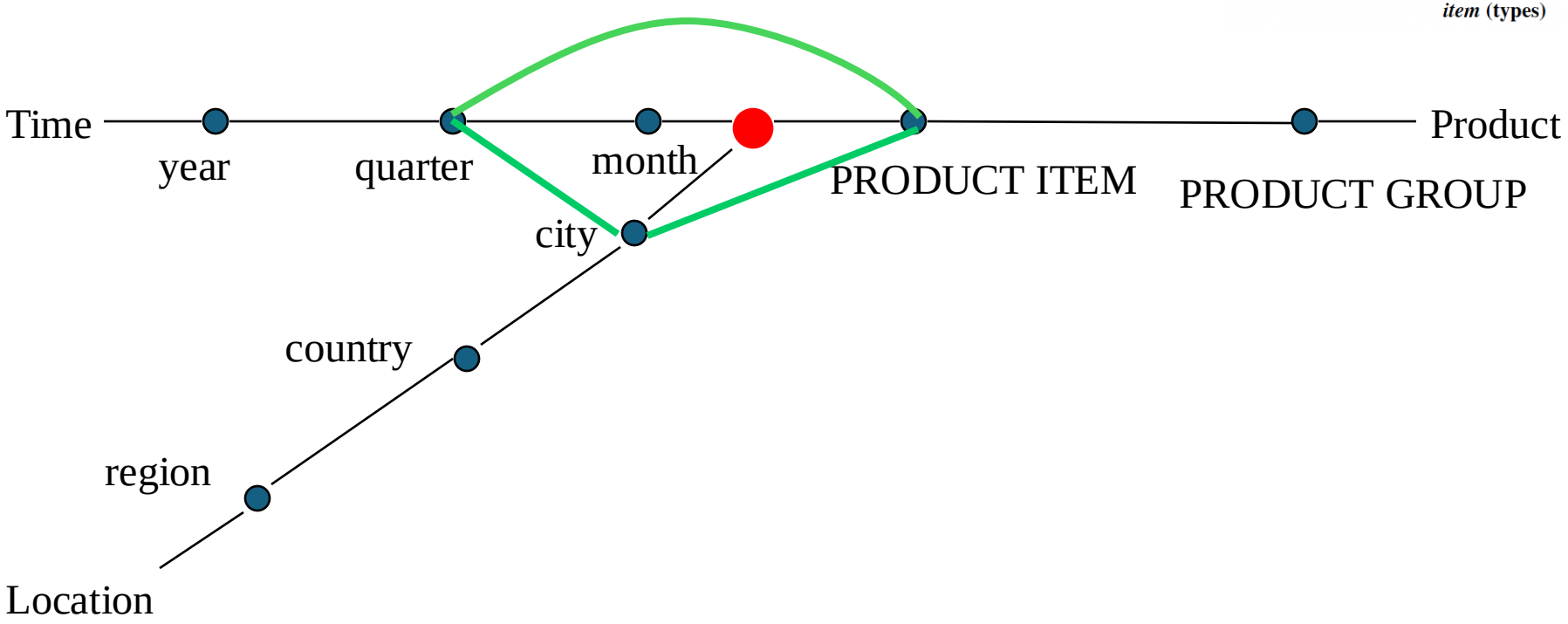
- Other operations
  - drill through: through the bottom level of the cube to its back-end relational tables (using SQL)



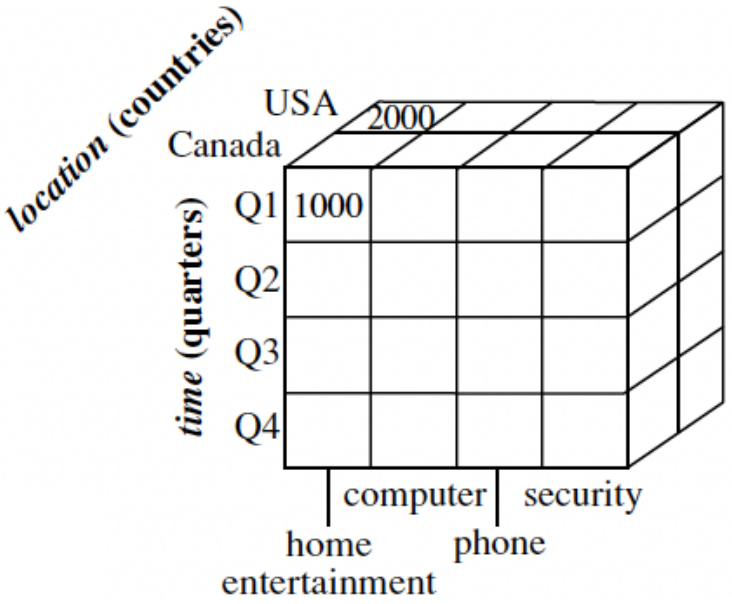
Transaction ID	Date	Product	Units Sold	Sales Amount
1001	2024-01-05	Product A	5	$1,000
1002	2024-01-12	Product A	10	$2,000
...	...	...	...	...

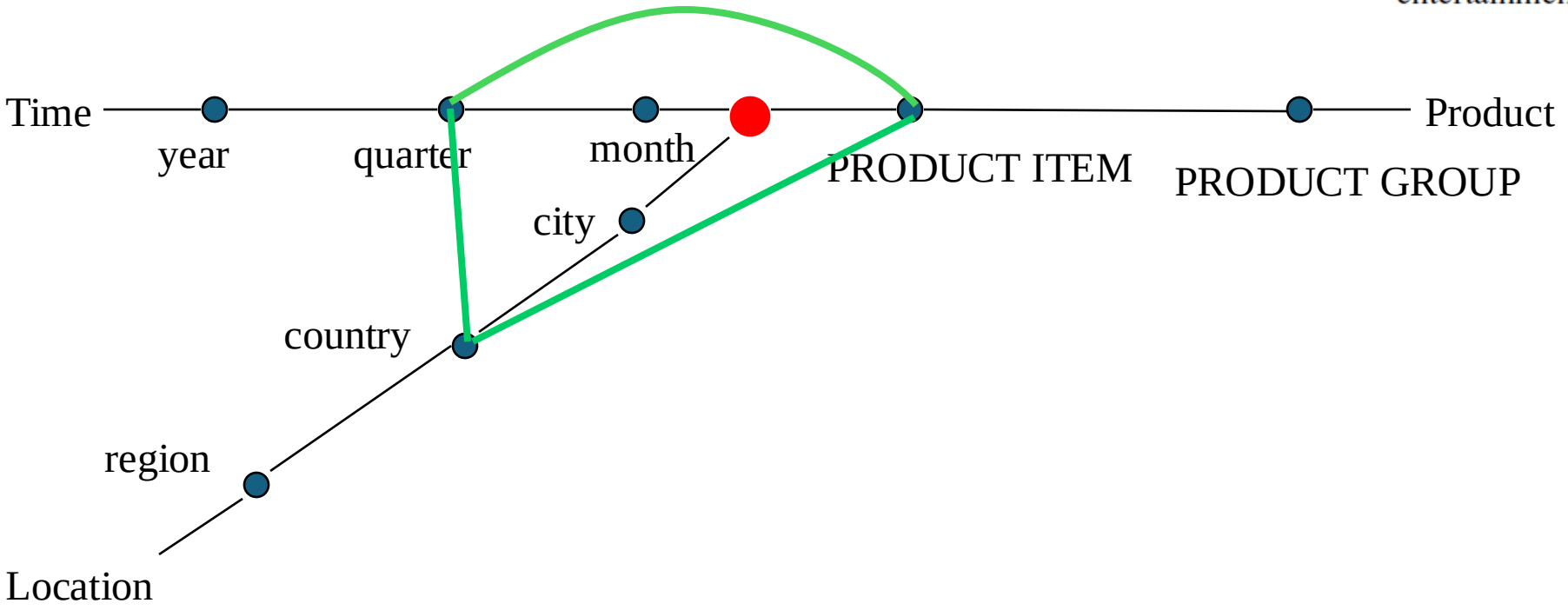
4.2.6 A Star-Net Query Model





4.2.6 A Star-Net Query Model





4.4 Data Warehouse Implementation

4.4.1 The “Compute Cube” Operator

- Cube definition and computation in DMQL

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

  compute cube sales_cube

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

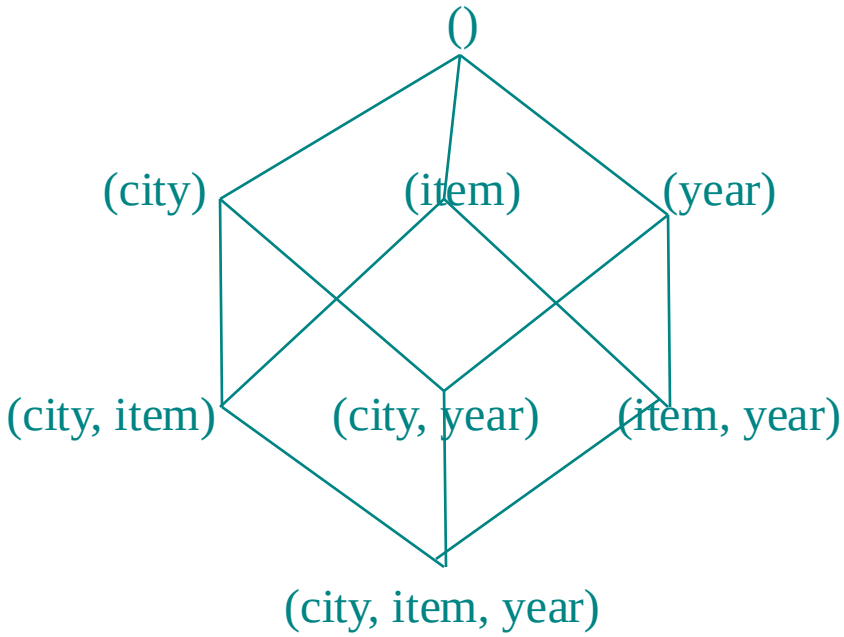
  SELECT item, city, year, SUM (amount)

  FROM SALES

  CUBE BY item, city, year

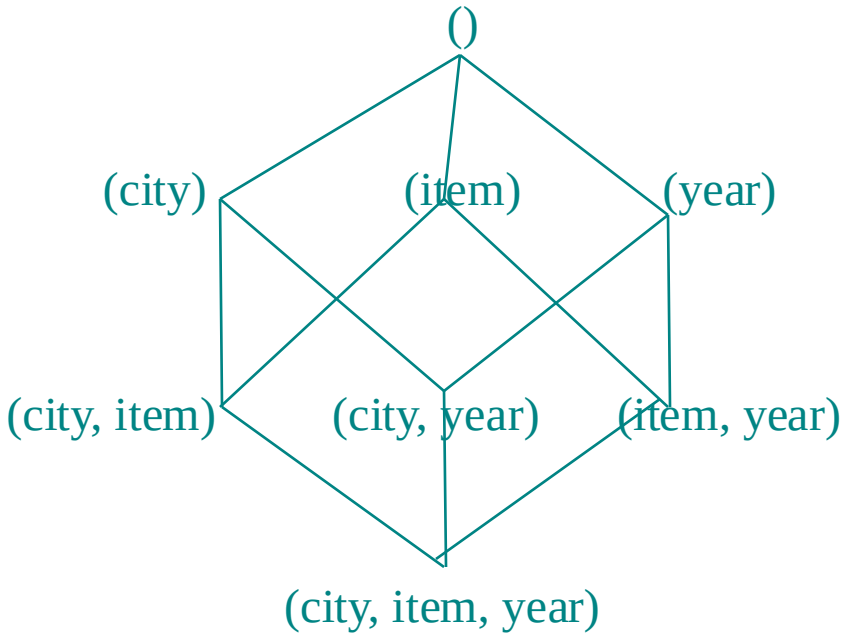
- Need compute the following Group-Bys

  (item, city, year),
  (item,city),(item, year), (city, year),
  (item), (city), (year)
  ()



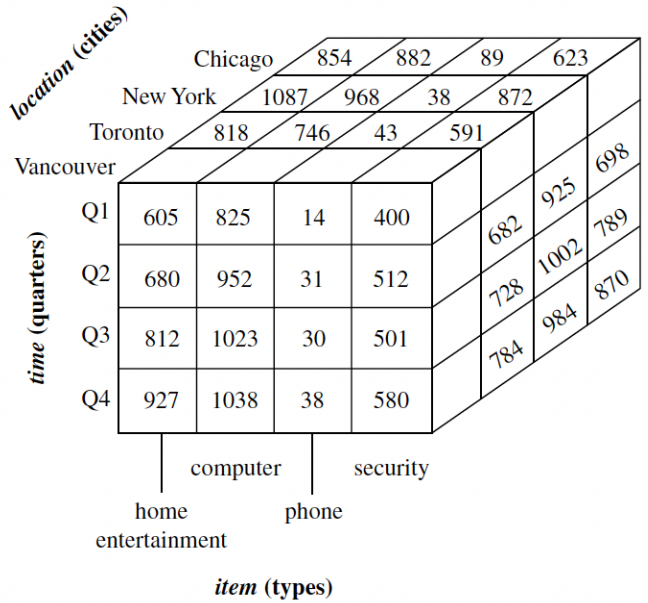
4.4.1 The “Compute Cube” Operator

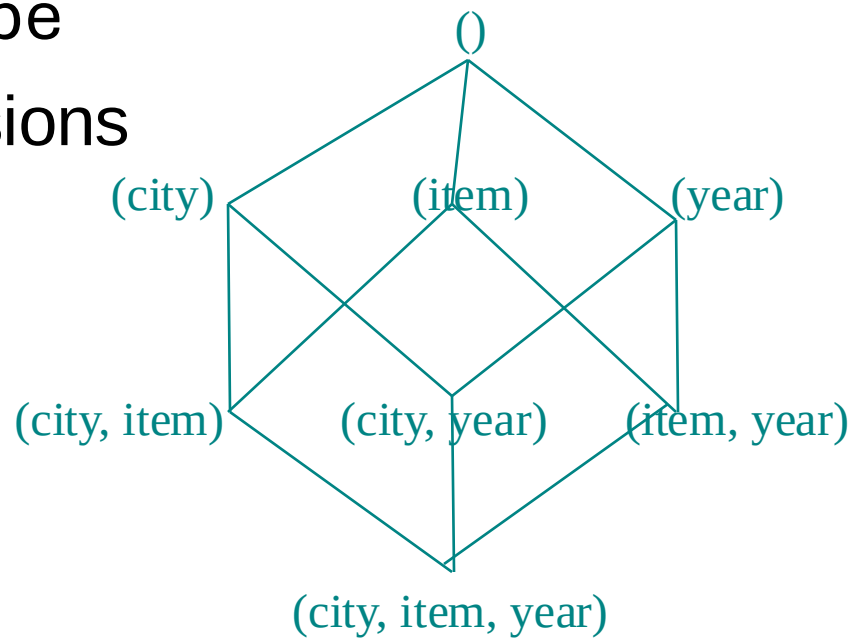
- Is it a good idea to pre-compute all the cuboids in the lattice?



4.4.1 Efficient Data Cube Computation

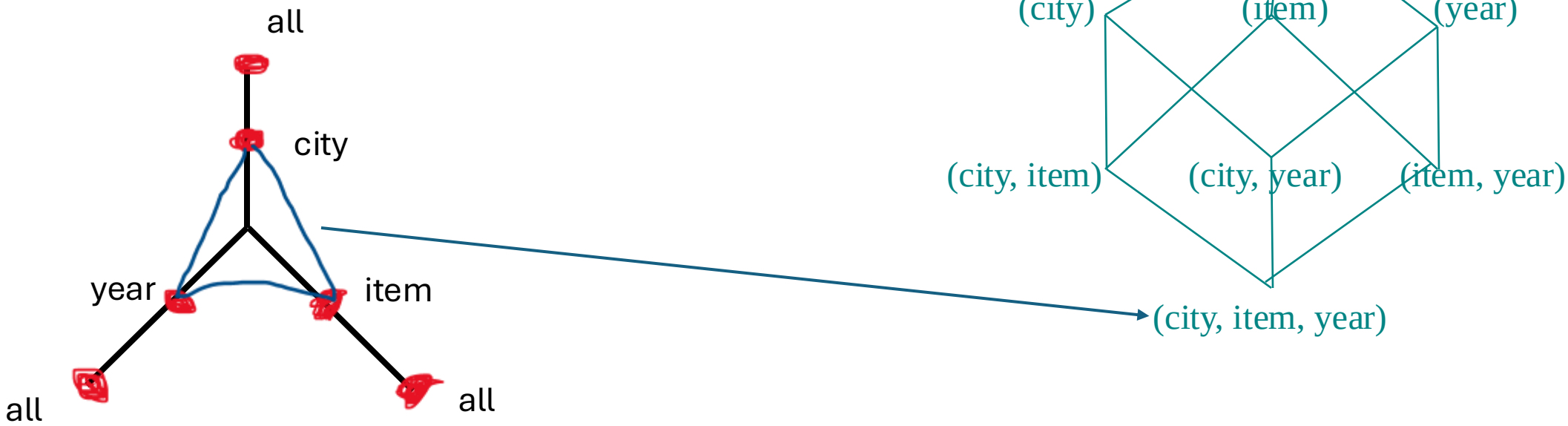
- Curse of dimensionality: need lots of storage space for cuboid precomputation!
- How many cuboids in an n-dimensional cube
  - Without concept hierarchies for all dimensions





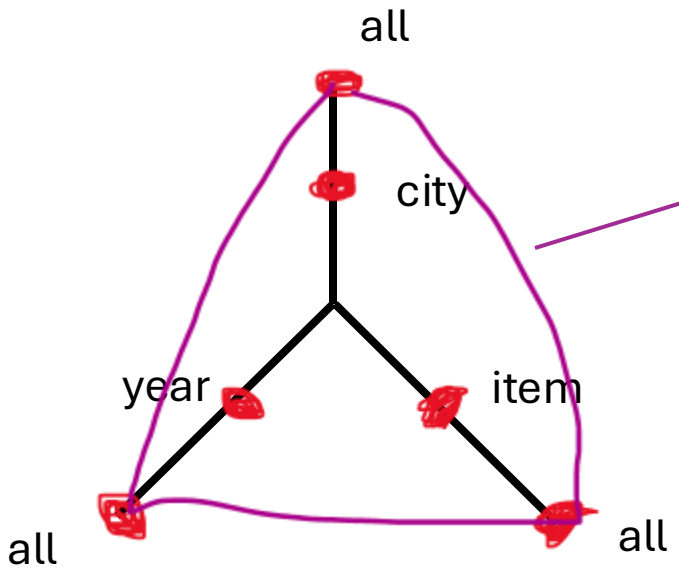
4.4.1 Efficient Data Cube Computation

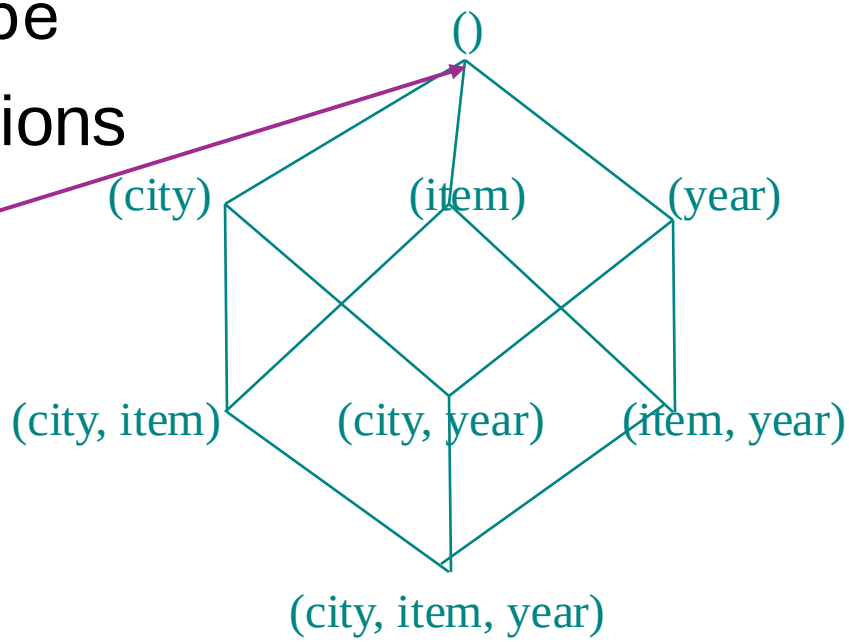
- Curse of dimensionality: need lots of storage space for cuboid precomputation!
- How many cuboids in an n-dimensional cube
  - Without concept hierarchies for all dimensions



4.4.1 Efficient Data Cube Computation

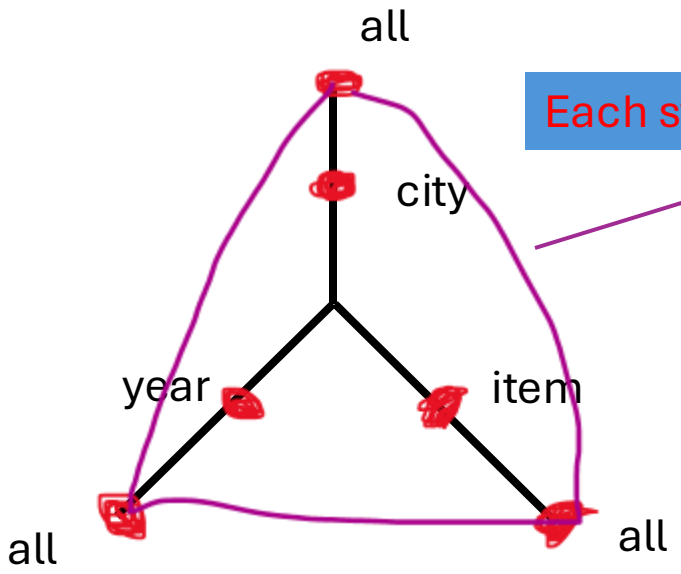
- Curse of dimensionality: need lots of storage space for cuboid precomputation!
- How many cuboids in an n-dimensional cube
  - Without concept hierarchies for all dimensions



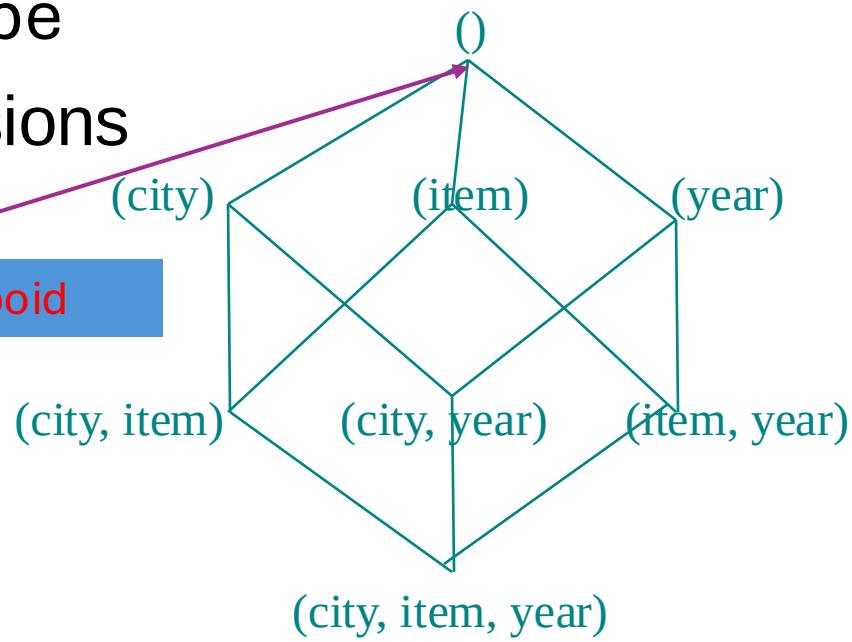


4.4.1 Efficient Data Cube Computation

- Curse of dimensionality: need lots of storage space for cuboid precomputation!
- How many cuboids in an n-dimensional cube
  - Without concept hierarchies for all dimensions

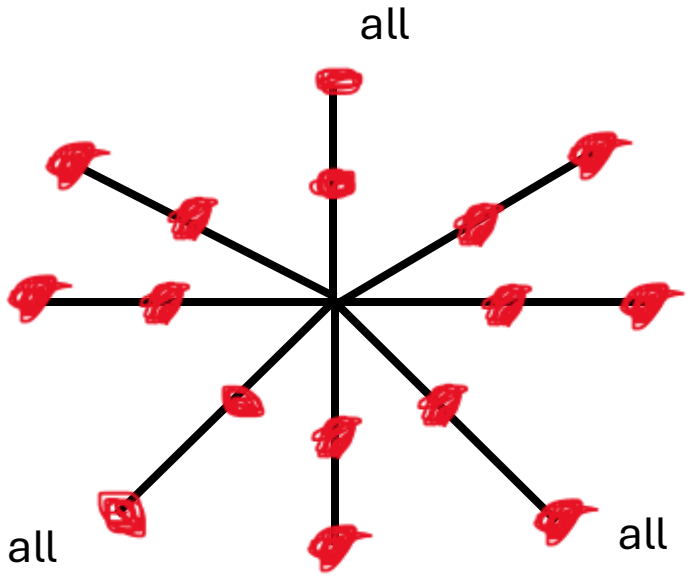


Each star is corresponding to a cuboid



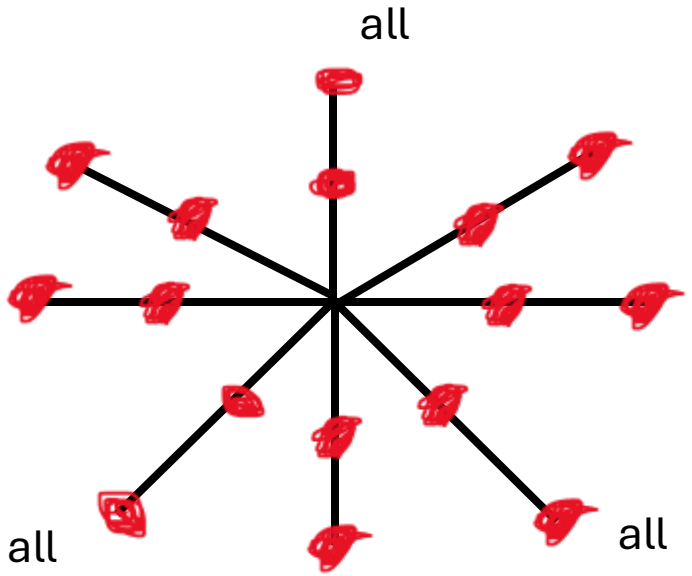
4.4.1 Efficient Data Cube Computation

- Curse of dimensionality: need lots of storage space for cuboid precomputation!
- How many cuboids in an n-dimensional cube
  - Without concept hierarchies for all dimensions



4.4.1 Efficient Data Cube Computation

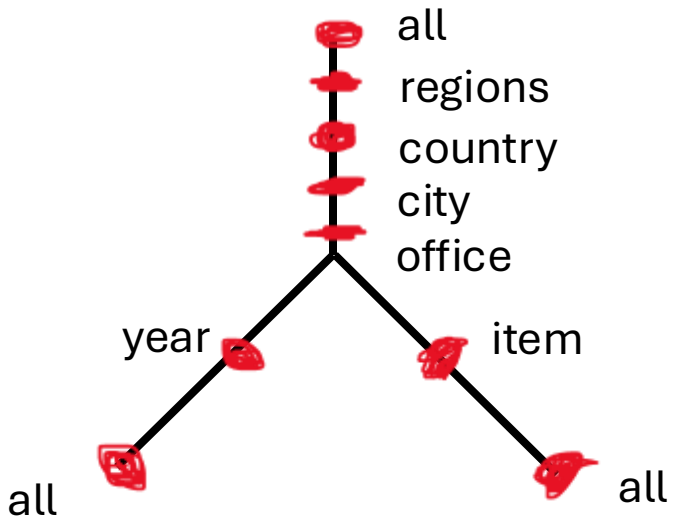
- Curse of dimensionality: need lots of storage space for cuboid precomputation!
- How many cuboids in an n-dimensional cube
  - Without concept hierarchies for all dimensions



$$2^n$$

4.4.1 Efficient Data Cube Computation

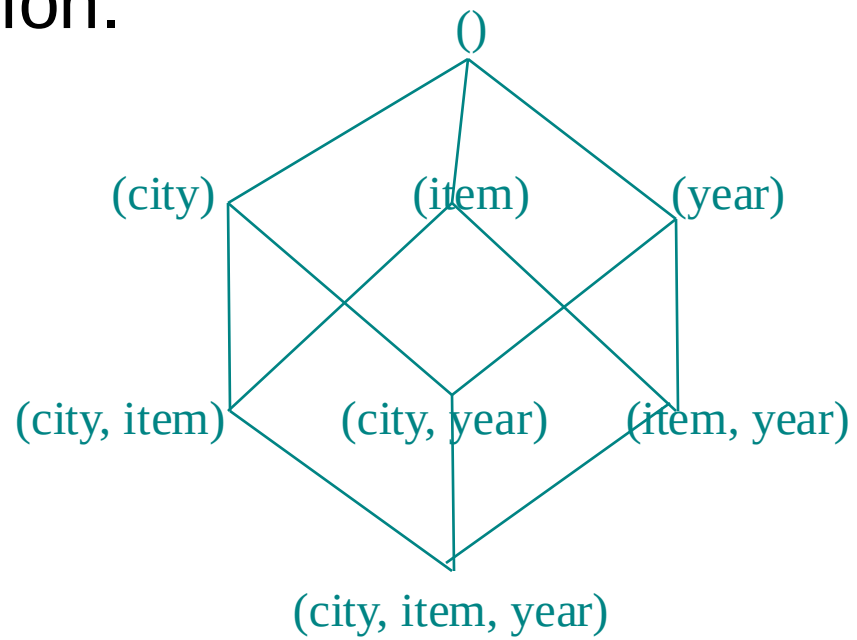
- Curse of dimensionality: need lots of storage space for cuboid precomputation!
- How many cuboids in an n-dimensional cube
  - With L_i levels for each dimension i?



$$T = \prod_{i=1}^{n} (L_i + 1)$$

4.4.1 Efficient Data Cube Computation

- Curse of dimensionality: need lots of storage space for cuboid precomputation!
- The size of each cuboid also depends on the cardinality (i.e., number of distinct values) of each dimension.



- 1000 cities x 1000 items → 1M records

for (city, item) cuboid!

4.4.1 Efficient Data Cube Computation

- As the number of dimensions, number of conceptual hierarchies, or cardinality increases, the storage space required explodes

- 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.

4.4.2 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

customer dimension 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

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

CustomerID	Name
1	Alice
2	Bob

Customer table

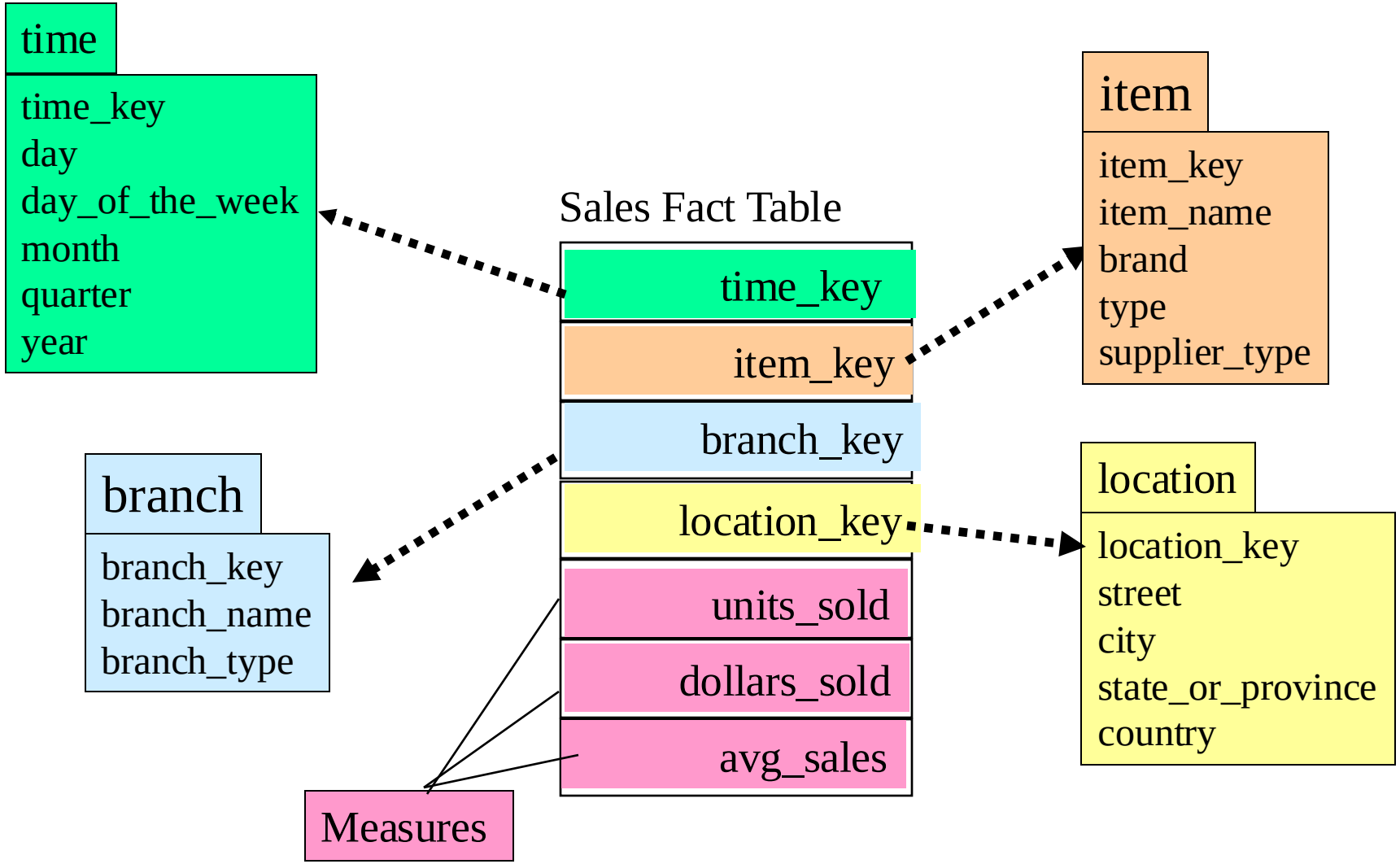
OrderID	CustomerID	Amount
100	1	50
101	2	100
102	1	70

Order table

(CustomerID, OrderID)
(1, 100)
(1, 102)
(2, 101)

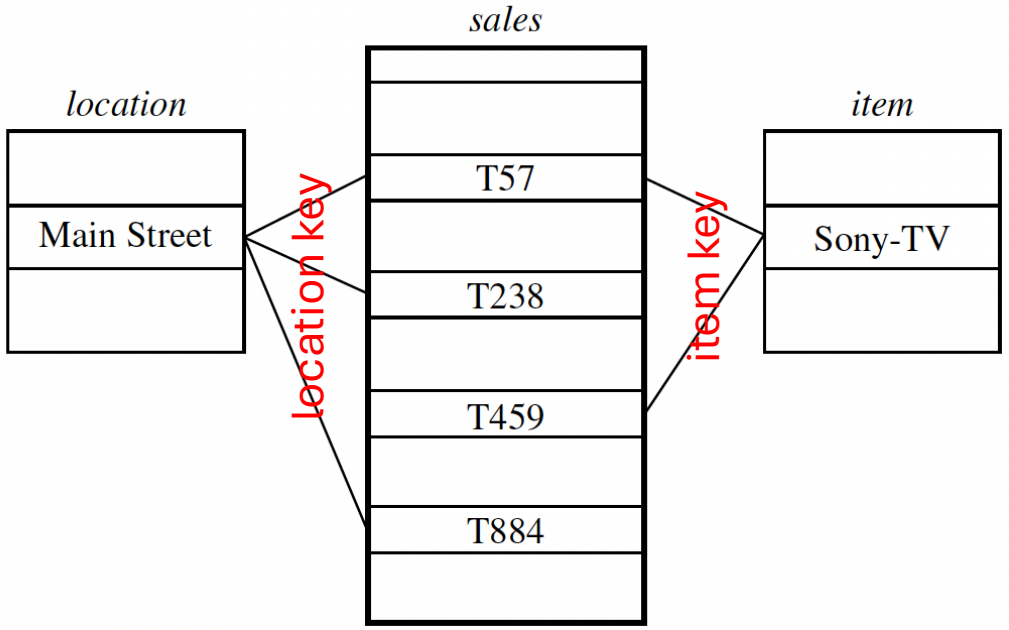
Join Index table

Recall: **Star Schema**



Indexing OLAP Data: **Join Indices**

- In data warehouses, join index relates the values of the dimensions of a star schema to rows in the fact table.
  - E.g. fact table: *Sales* and two dimensions *location* and *item*
    - 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



Indexing OLAP Data: **Join Indices**

Join index table for *location/sales*

location	*sales_key*
. . .	. . .
Main Street	T57
Main Street	T238
Main Street	T884
. . .	. . .

Join index table for *item/sales*

item	*sales_key*
. . .	. . .
Sony-TV	T57
Sony-TV	T459
. . .	. . .

Join index table linking *location* and *item* to *sales*

location	*item*	*sales_key*
. . .	. . .	. . .
Main Street	Sony-TV	T57
. . .	. . .	. . .

4.4.3 Efficient Processing OLAP Queries

- **1. Determine which operations** should be performed on the available cuboids
  - Transform drill, roll, etc. into corresponding SQL and/or OLAP operations, e.g., roll = ROLL UP

4.4.3 Efficient Processing OLAP Queries

- **2. Determine which materialized cuboid(s)** should be selected for OLAP op.
  - A sales cube [time, item, location]: sum(sales in dollars). The dimension hierarchies used are “day < month < quarter < year” for time; “item name < brand < type” for item; and “street < city < state < country” for location.

  - Let the query to be processed be on {brand, state} with the condition “year = 2024”, and there are 4 materialized cuboids available:

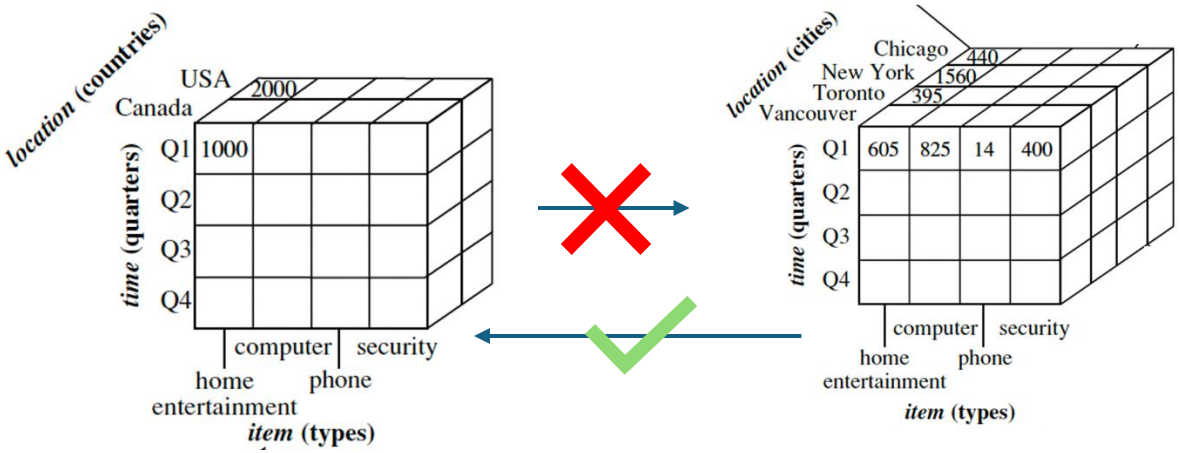
    1) {year, item_name, city}

    2) {year, brand, country}

    3) {year, brand, state}

    4) {item_name, state} where year = 2024

    Which should be selected to process the query?



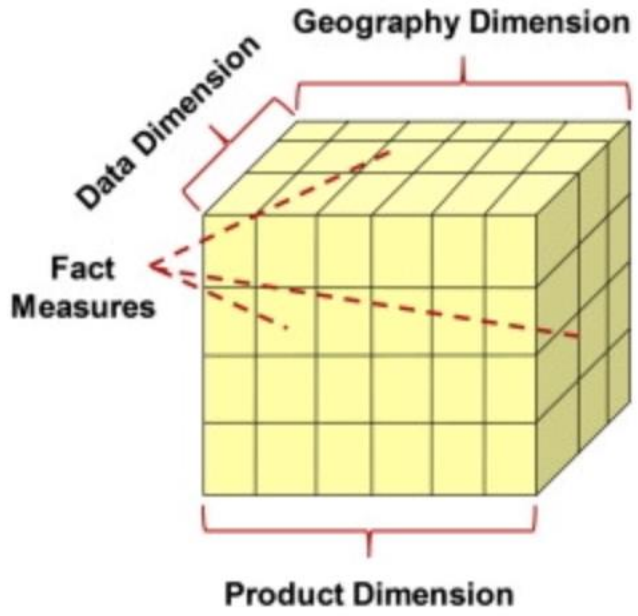
4.4.4 OLAP Server Architectures

- Relational OLAP (ROLAP)
  - Use relational or extended-relational DBMS to store and manage warehouse data and OLAP middle ware
  - Include optimization of DBMS backend, implementation of aggregation navigation logic, and additional tools and services
  - Greater scalability

RID	*item*	…	*day*	*month*	*quarter*	*year*	*dollars_sold*
1001	TV	…	15	10	Q4	2010	250.60
1002	TV	…	23	10	Q4	2010	175.00
…	…	…	…	…	…	…	…
5001	TV	…	all	10	Q4	2010	45,786.08
…	…	…	…	…	…	…	…

4.4.4 OLAP Server Architectures

- Multidimensional OLAP (MOLAP)
  - Use multi-dimensional array
  - Fast indexing to pre-computed summarized data




```
>>> np.random.rand(3,2,3)
array([[[0.97541511, 0.00678858, 0.65340981],
        [0.95958069, 0.93996982, 0.92168797]],

       [[0.1482778 , 0.04040431, 0.49215434],
        [0.54894307, 0.46190632, 0.4821958 ]],

       [[0.74778229, 0.45224821, 0.21350392],
        [0.02420554, 0.4659273 , 0.86234   ]]])
```

4.4.4 OLAP Server Architectures

- Hybrid OLAP (HOLAP) (e.g., Microsoft SQLServer)
  - Flexibility, e.g., low level: relational, high-level: array

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, slicing, dicing and pivoting
- Implementation: Efficient computation of data cubes
  - Partial vs. full vs. no materialization
  - Indexing OALP data: Bitmap index and join index
  - OLAP query processing
  - OLAP servers: ROLAP, MOLAP, HOLAP

Reference

- Jiawei Han and Micheline Kamber, Data Mining Concepts and Techniques, 3rd Edition, Morgan Kaufmann Publishers, ISBN-13: 978-9380931913, ISBN-10: 9780123814791