CS 412 Intro. to Data Mining

Chapter 3. Data Warehousing and On-line Analytical Processing

Hanghang Tong, Computer Science, Univ. Illinois at Urbana-Champaign, 2023

Chapter 3: Data Warehousing and On-line Analytical Processing

- Data Warehouse: Basic Concepts
- Data Warehouse Modeling
- OLAP Operations
- Data Cube Computation: Concepts and Methods
- Summary

What is a Data Warehouse?

- Defined in many different ways, but not rigorously
  - Support decision
  - Maintained Separately
  - Information processing
- "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
- Data warehousing:
  - The process of constructing and using data warehouses

Data Warehouse—Subject-Oriented

- Help make decisions
  - A **simple** and **concise** view (modeling and analysis)
  - Not details (transaction processing)
  - Organizing around major subjects, such as customer, product, sales
  - Excluding data that are not useful in the decision support process

Data Warehouse—Integrated

- Integrating multiple, heterogeneous sources
  - Ex. relational databases, flat files, on-line transaction records
- Consistency
  - Data cleaning and data integration techniques are applied.
  - Ex. Hotel price: differences on currency, tax, breakfast covered, and parking
  - When data is moved to the warehouse, it is converted

Data Warehouse—Time Variant

Data Warehouse	Operational Database
Long time horizon (e.g., past 5-10 years)	current value data
Contains an element of time, explicitly or implicitly	data may or may not contain "time element"

Data Warehouse—Nonvolatile

- Independence – A physically separate store
- Static – No data management (updates, transaction processing, recovery, and concurrency control mechanisms)
- Requires only two operations in data accessing:
  - *initial loading of data* and *access of data*

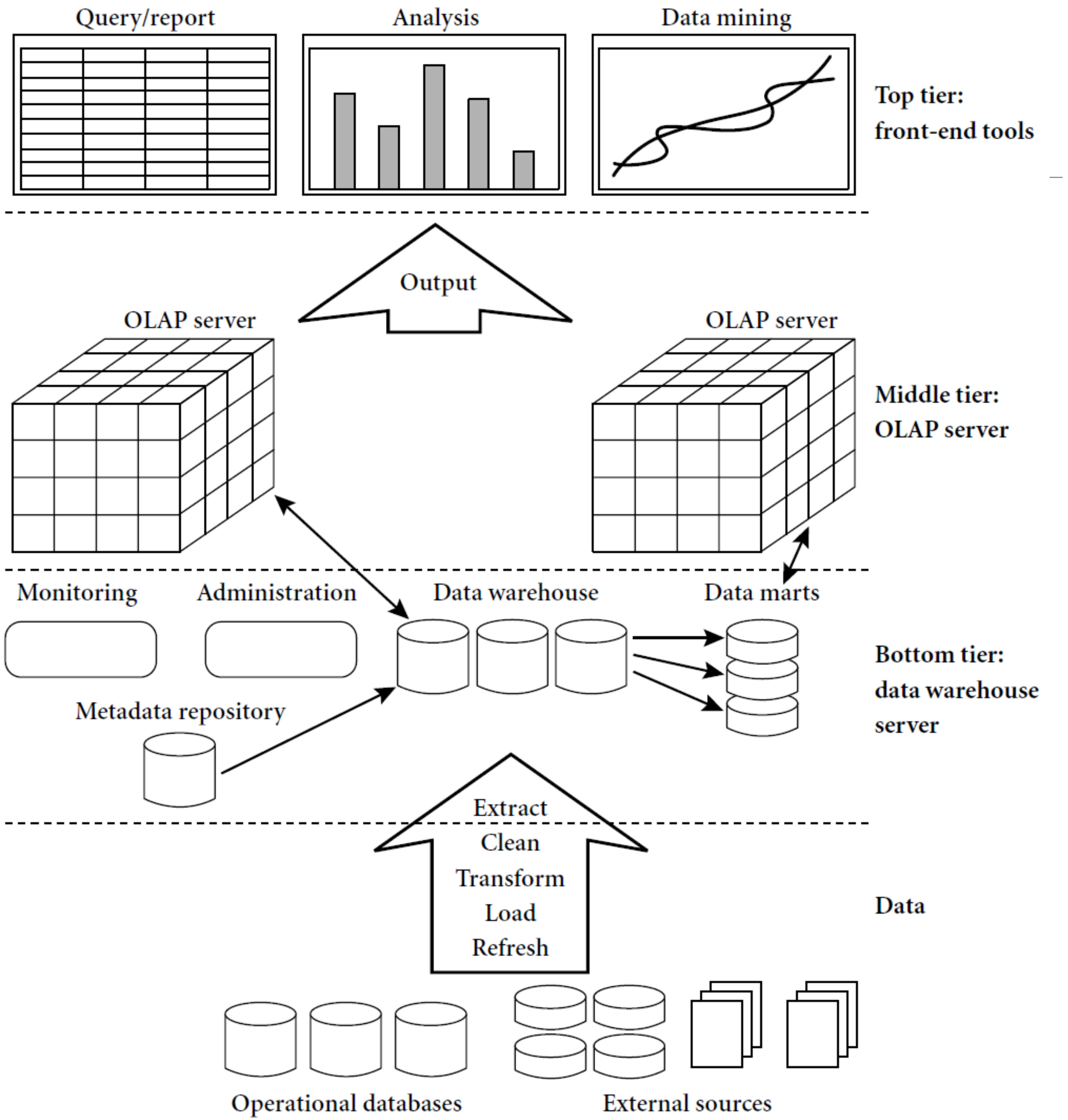
OLTP vs. OLAP

- OLTP: Online transactional processing
  - DBMS operations
  - Query and transactional processing
- OLAP: Online analytical processing
  - Data warehouse operations
  - Drilling, slicing, dicing, etc.

	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

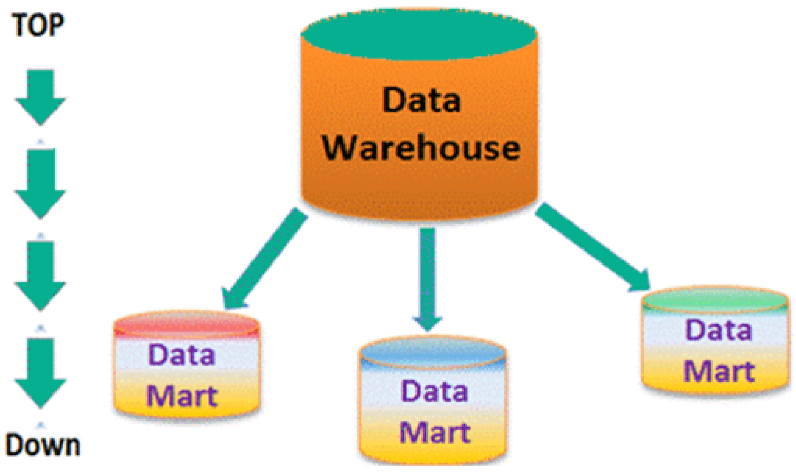
Data Warehouse: A Multi-Tiered Architecture

- Top Tier: Front-End Tools
- Middle Tier: OLAP Server
- Bottom Tier: Data Warehouse Server
- Data



Three Data Warehouse Models

- **Enterprise warehouse -** Specially designed for the entire organization
- **Data Mart**
  - Specific set of subjects, selected groups of users
  - Independent vs. dependent (directly from warehouse) data mart
- **Virtual warehouse**
  - A set of views over operational databases
  - Only some of the possible summary views may be materialized



https://www.guru99.com/data-warehouse-vs-data-mart.html

Extraction, Transformation, and Loading (ETL)

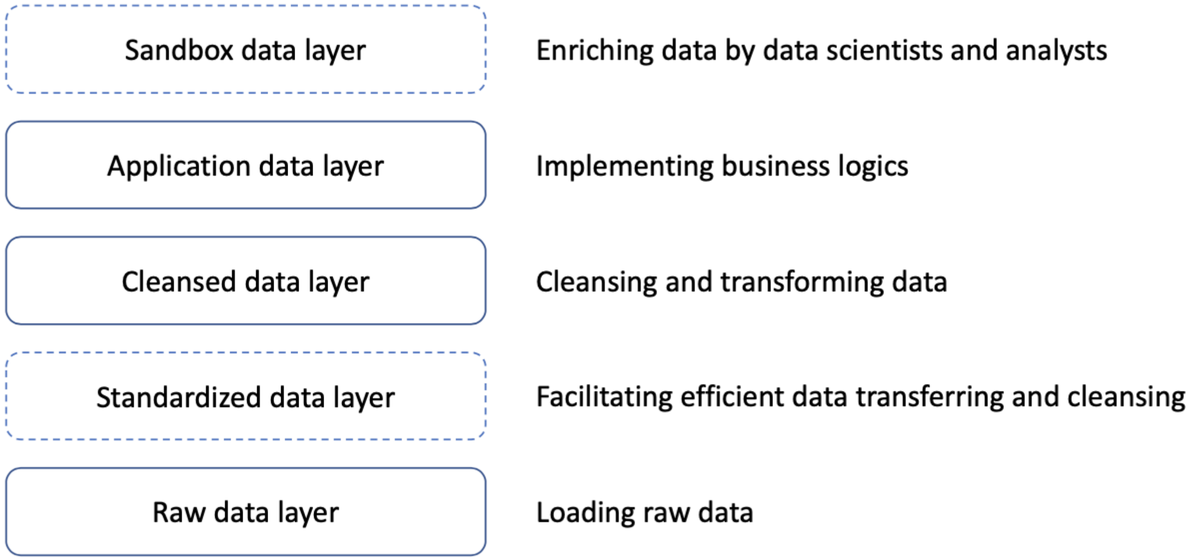
- **Data extraction**
  - get data from multiple, heterogeneous, and external sources
- **Data cleaning**
  - detect errors in the data and rectify them when possible
- **Data transformation**
  - convert data from legacy or host format to warehouse format
- **Load**
  - sort, summarize, consolidate,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

Metadata Repository

- **Meta data** is data about data. It stores:
  - Description of structure (schema, etc.)
  - Operational meta-data
    - data lineage (history of migrated data and transformation path), currency of data (active, archived, or purged), monitoring information (warehouse usage statistics, error reports, audit trails)
  - The algorithms used for summarization
  - The mapping from operational environment to the data warehouse
  - Data related to system performance
    - warehouse schema, view and derived data definitions
  - Business data
    - business terms and definitions, ownership of data, charging policies

From Data Warehouse to Data Lake

- Data lake: a single repository of all enterprise data in the natural format
  - Relational data, semi-structured data (e.g., XML, JSON), unstructured data (e.g., emails, Pdf files) and binary data (e.g., images, videos, audio)
- Data lake vs. data warehouse
  - Data Warehouse: top-down, structured and centralized
  - Data Lake: bottom-up, quick prototyping and democratic
- Data storage in data lake
  - Mandatory layers:
    - raw data, cleansed data, and application data
  - Optional layers
    - standardized data layer & sandbox data layer



Chapter 3: Data Warehousing and On-line Analytical Processing

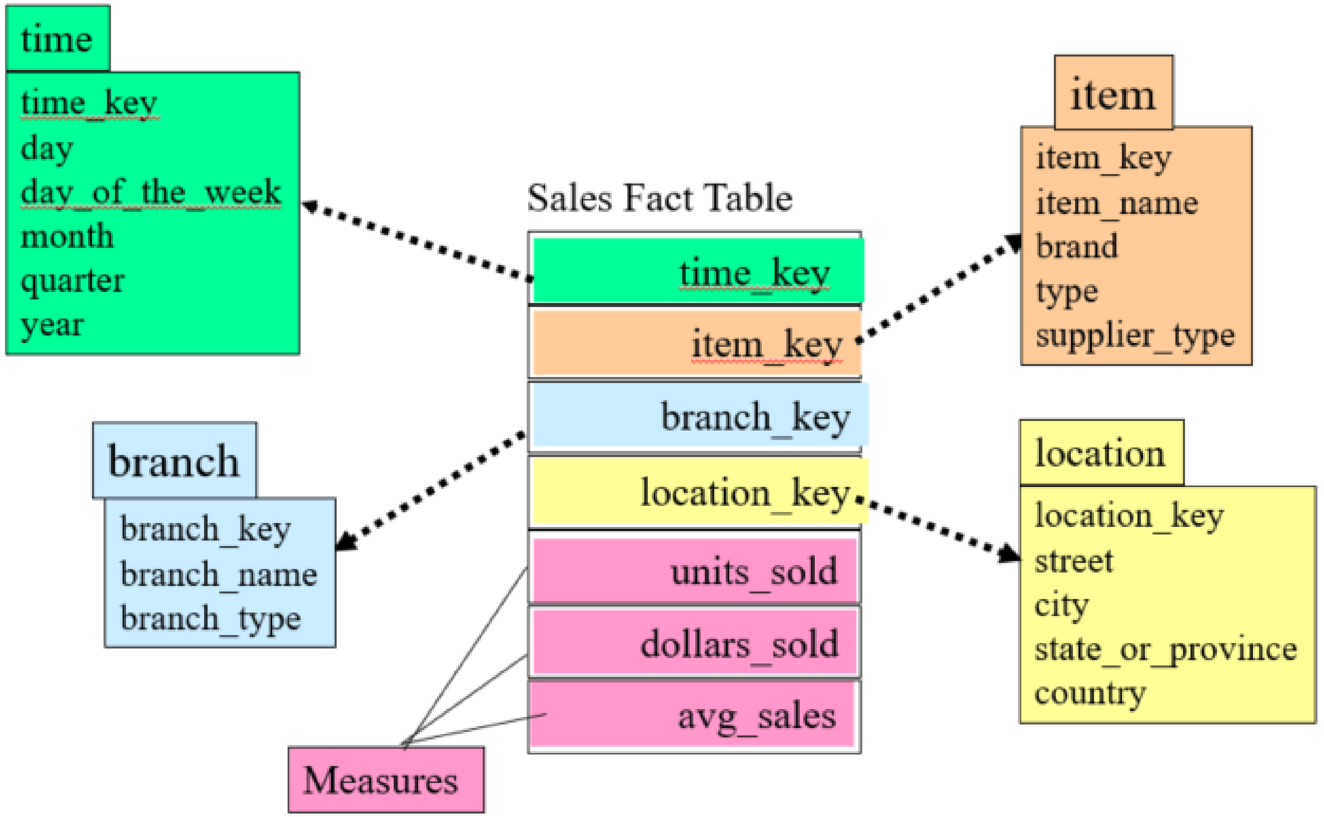
- Data Warehouse: Basic Concepts
- Data Warehouse Modeling
- OLAP Operations
- Data Cube Computation: Concepts and Methods
- Summary

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
- Main function is to provide summarizations of the data
  - E.g., summarize the units or dollars sold at a particular store over a particular time period
- Can compute summarizations online (as they are requested)
  - Can be very slow
- Better to pre-calculate some summarizations

Design of Data Warehouses

- **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
- Different schema exist
  - Star
  - Snowflake
  - Fact constellation

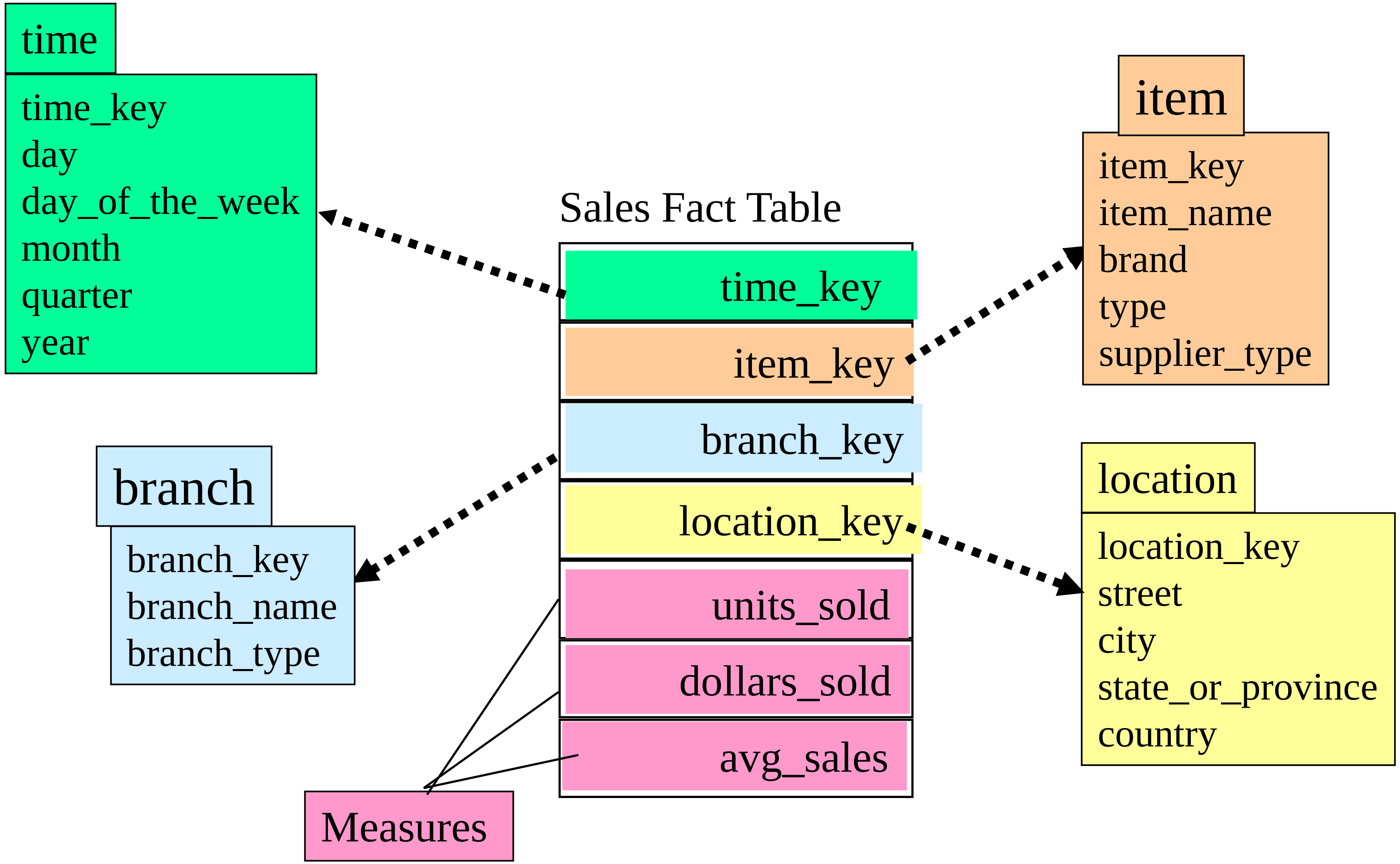


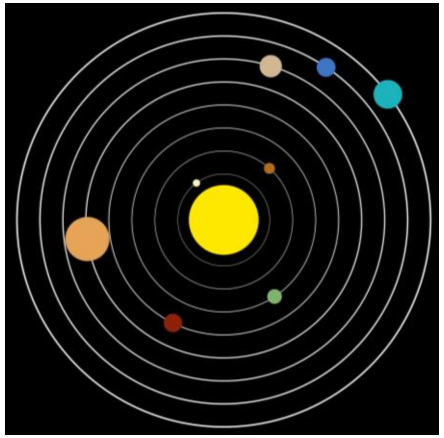
Conceptual Modeling of Data Warehouses

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

Star Schema: An Example

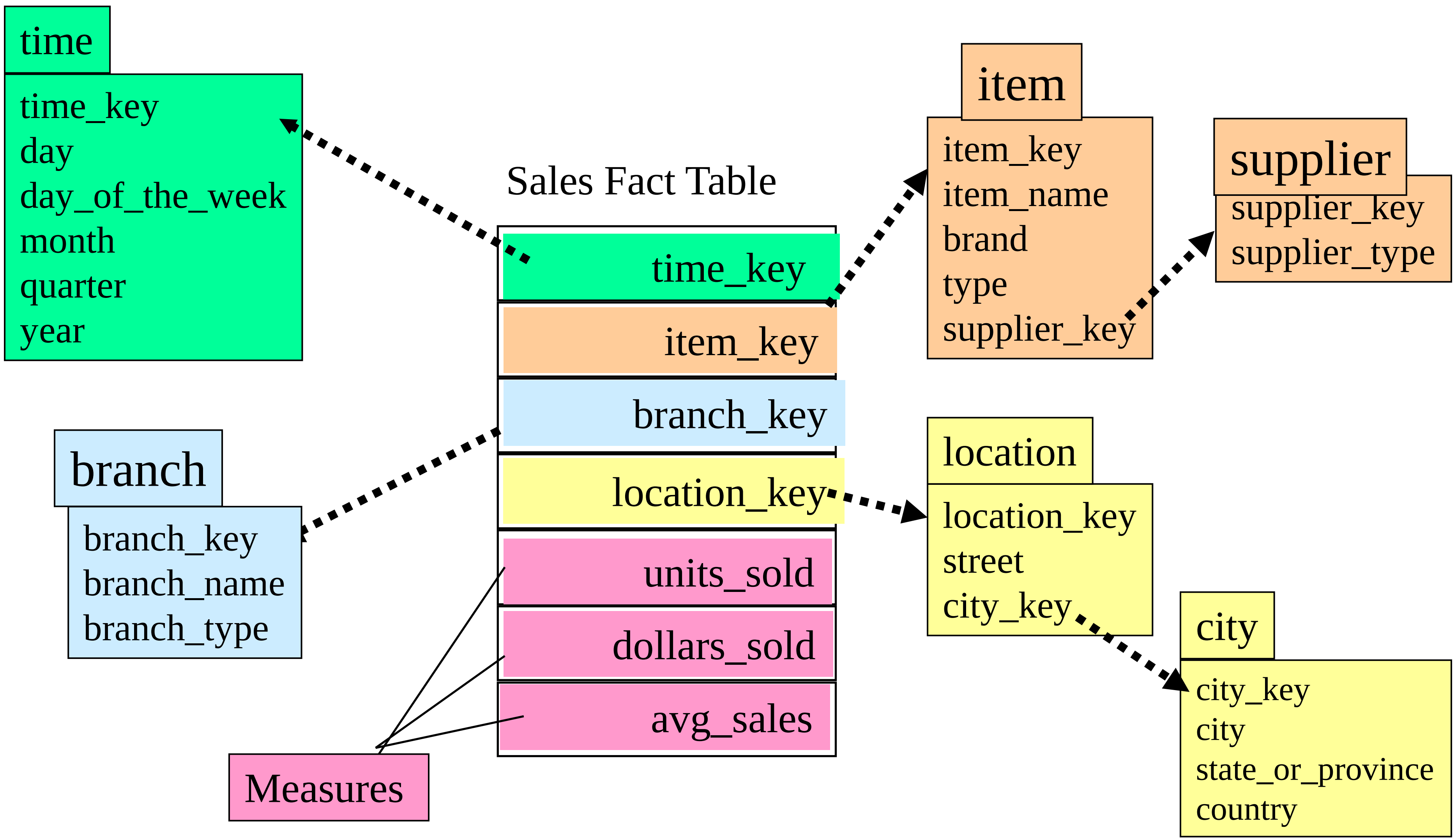
A fact table in the middle connected to a set of dimension tables





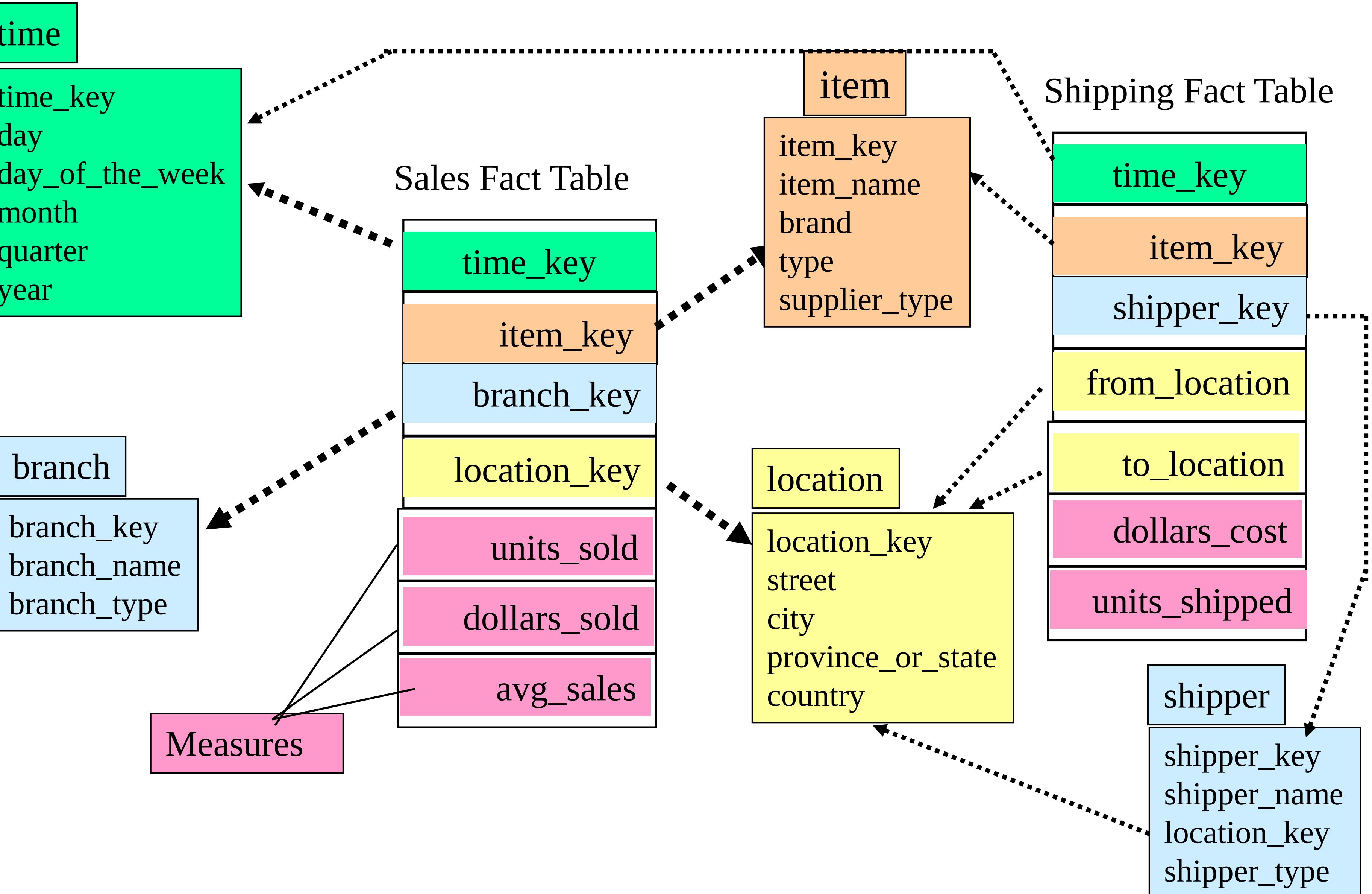
Snowflake Schema: An Example

A refinement of star schema where some dimensional hierarchy is normalized into a set of smaller dimension tables, forming a shape similar to snowflake

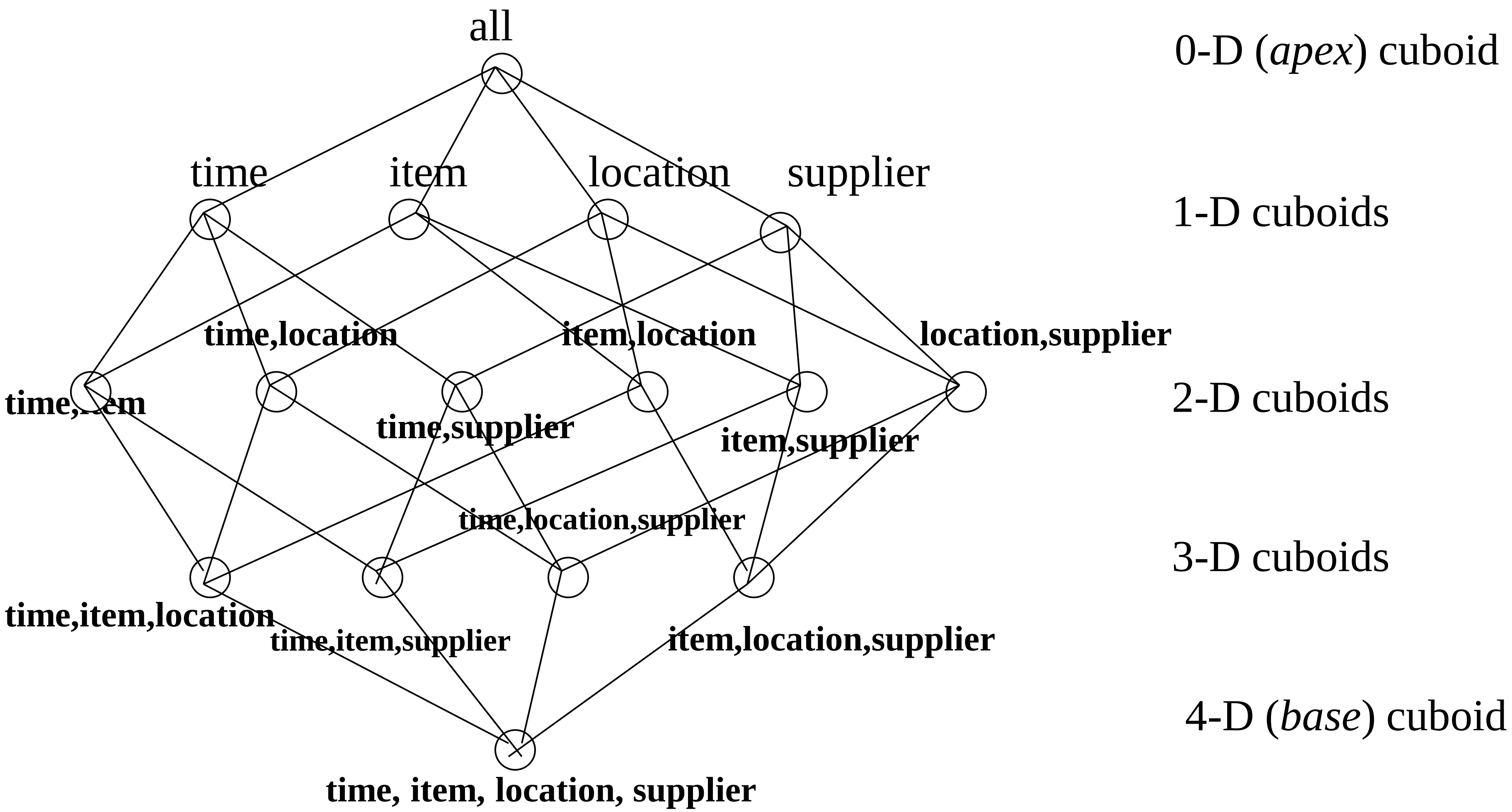


Fact Constellation: An Example

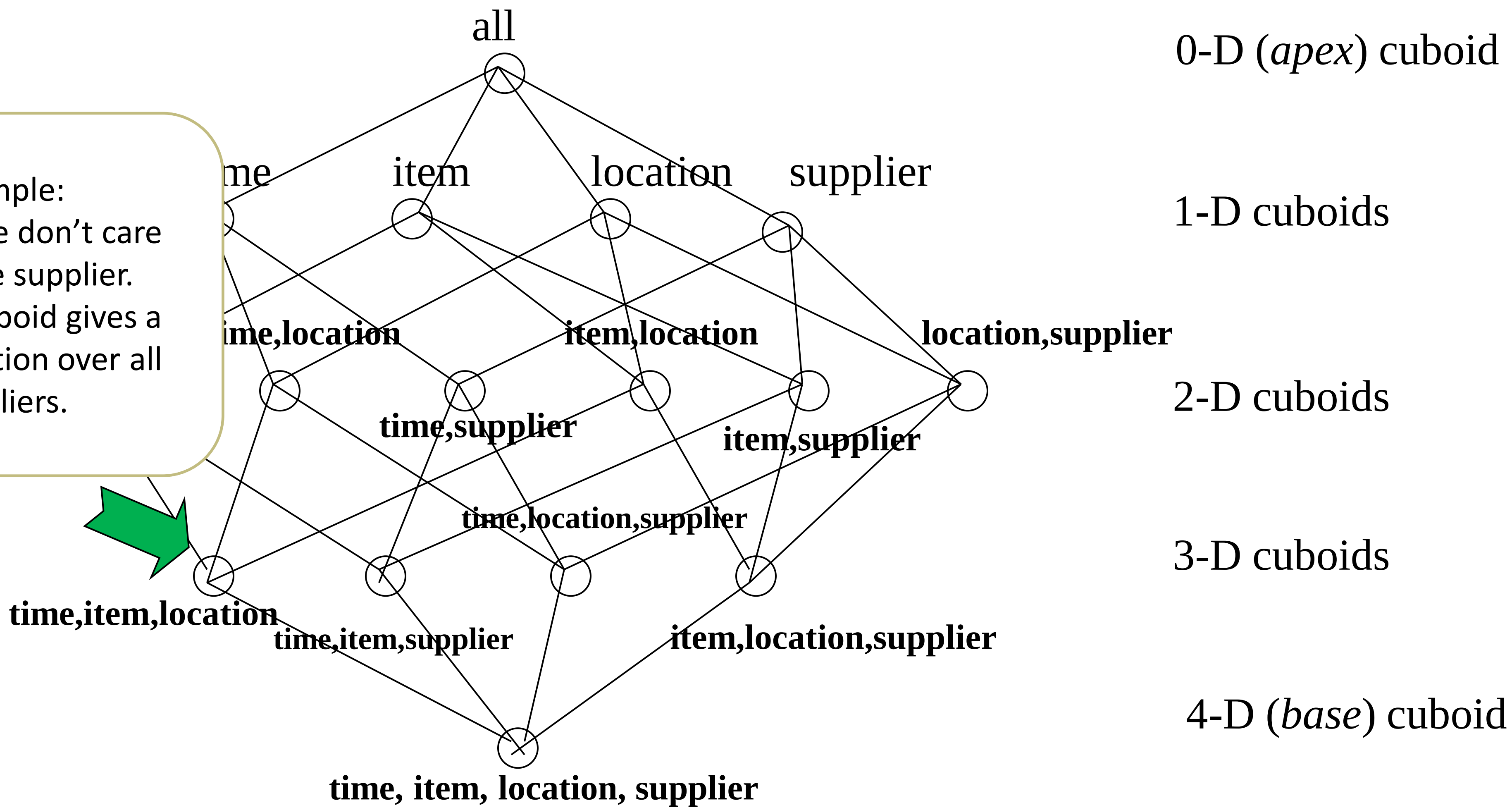
Multiple fact tables share dimension tables, viewed as a collection of stars, therefore called galaxy schema or fact constellation



Data Cube: A Lattice of Cuboids

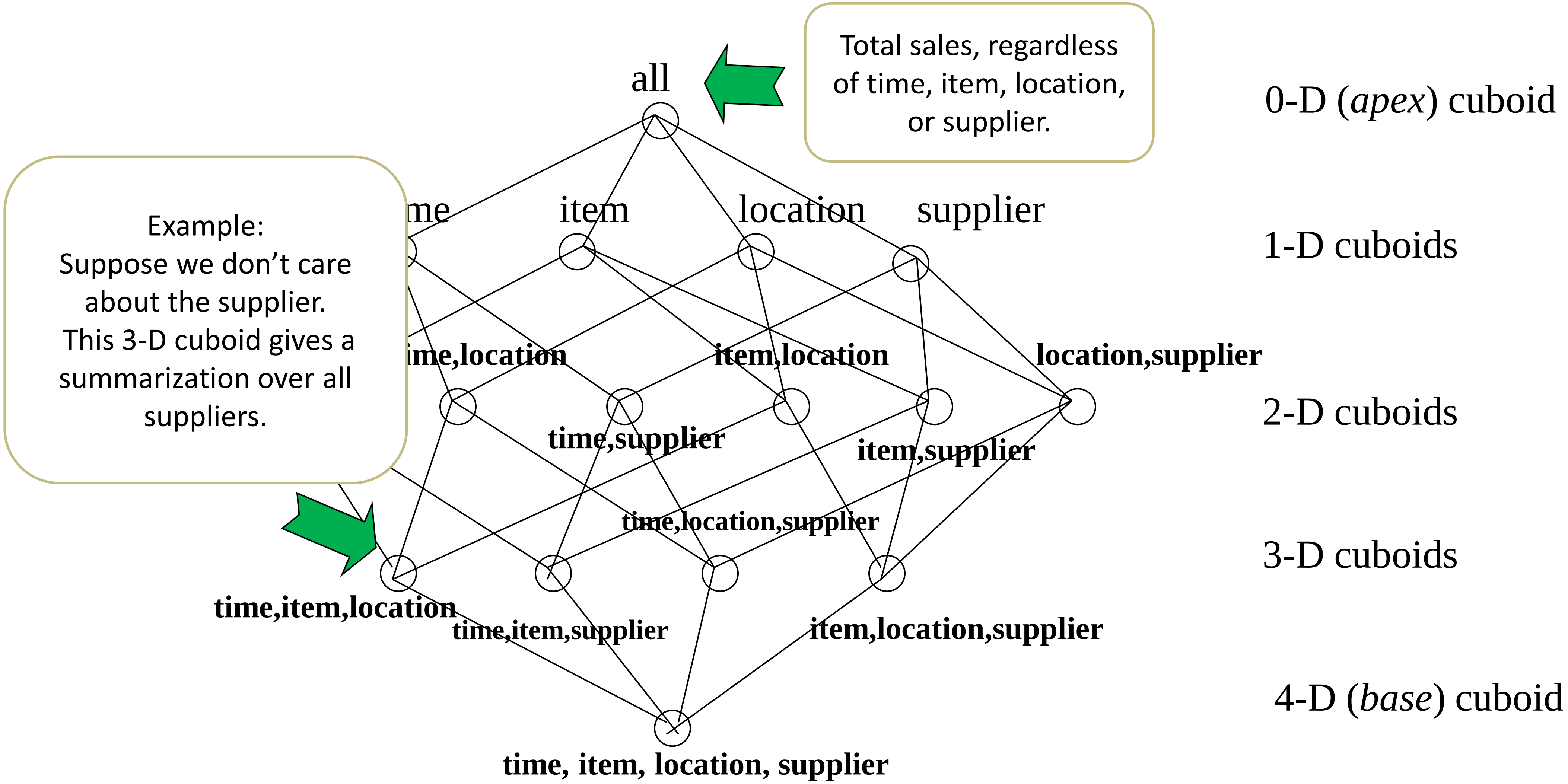


Data Cube: A Lattice of Cuboids



Example:
Suppose we don't care about the supplier. This 3-D cuboid gives a summarization over all suppliers.

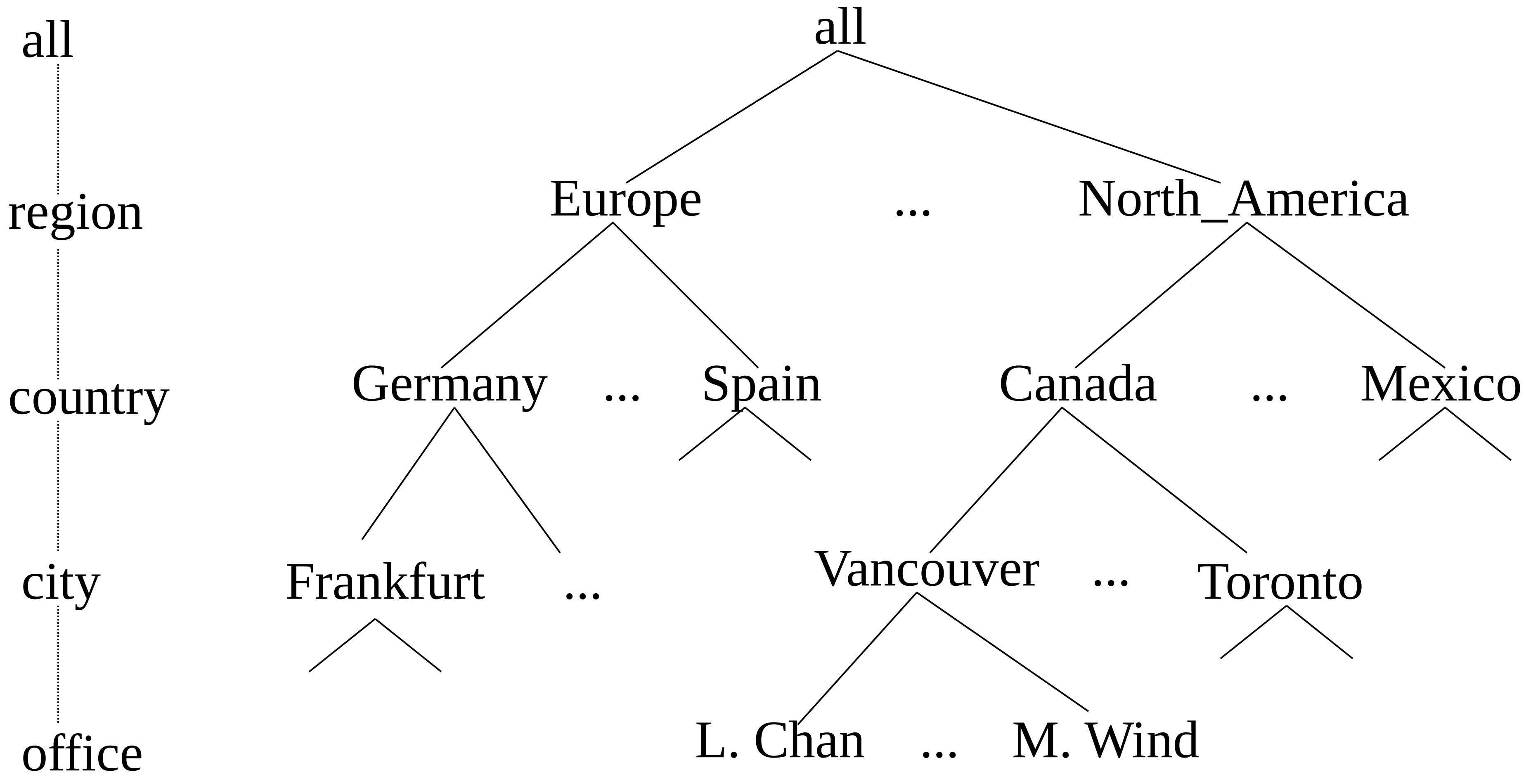
Data Cube: A Lattice of Cuboids



Calculating Number of Cuboids

- Consider dimensions as binary numbers
- Example: 4 dimensions
  - Each is either in the cuboid, or not in the cuboid
  - (, , ,) ← choice of 0 or 1 for each element of vector
  - Sum up for each position: $2^3 + 2^2 + 2^1 + 2^0 + 1$ (0-d cuboid) $= 2^4$
- In general, 2^d cuboids (d = number of dimensions)

A Concept Hierarchy for a Dimension (location)

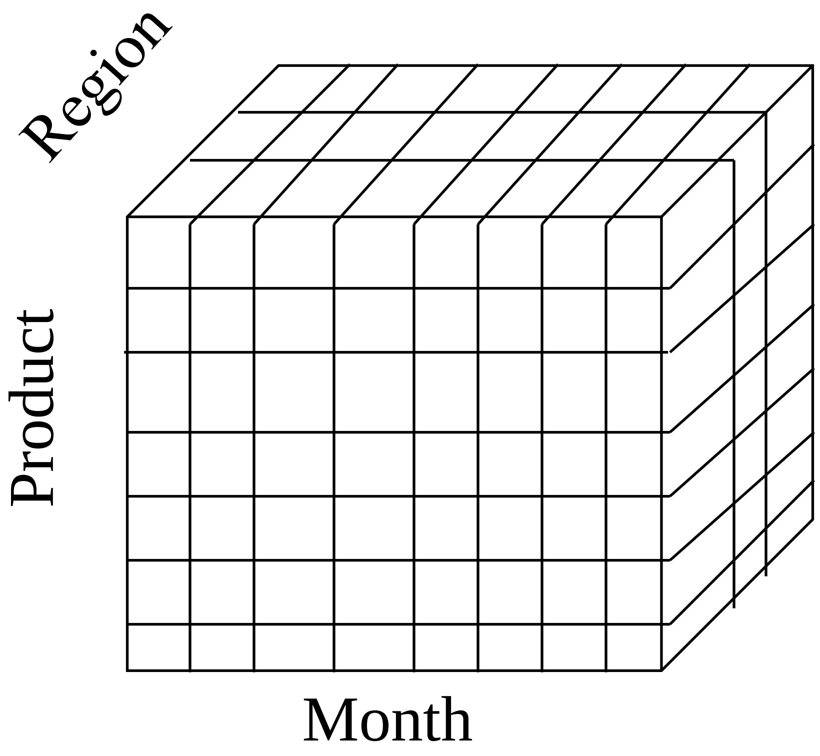


Chapter 3: Data Warehousing and On-line Analytical Processing

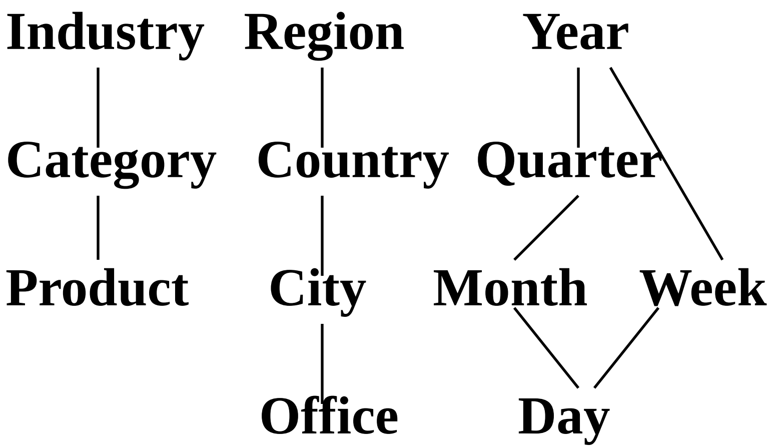
- Data Warehouse: Basic Concepts
- Data Warehouse Modeling
- OLAP Operations
- Data Cube Computation: Concepts and Methods
- Summary

Multidimensional Data

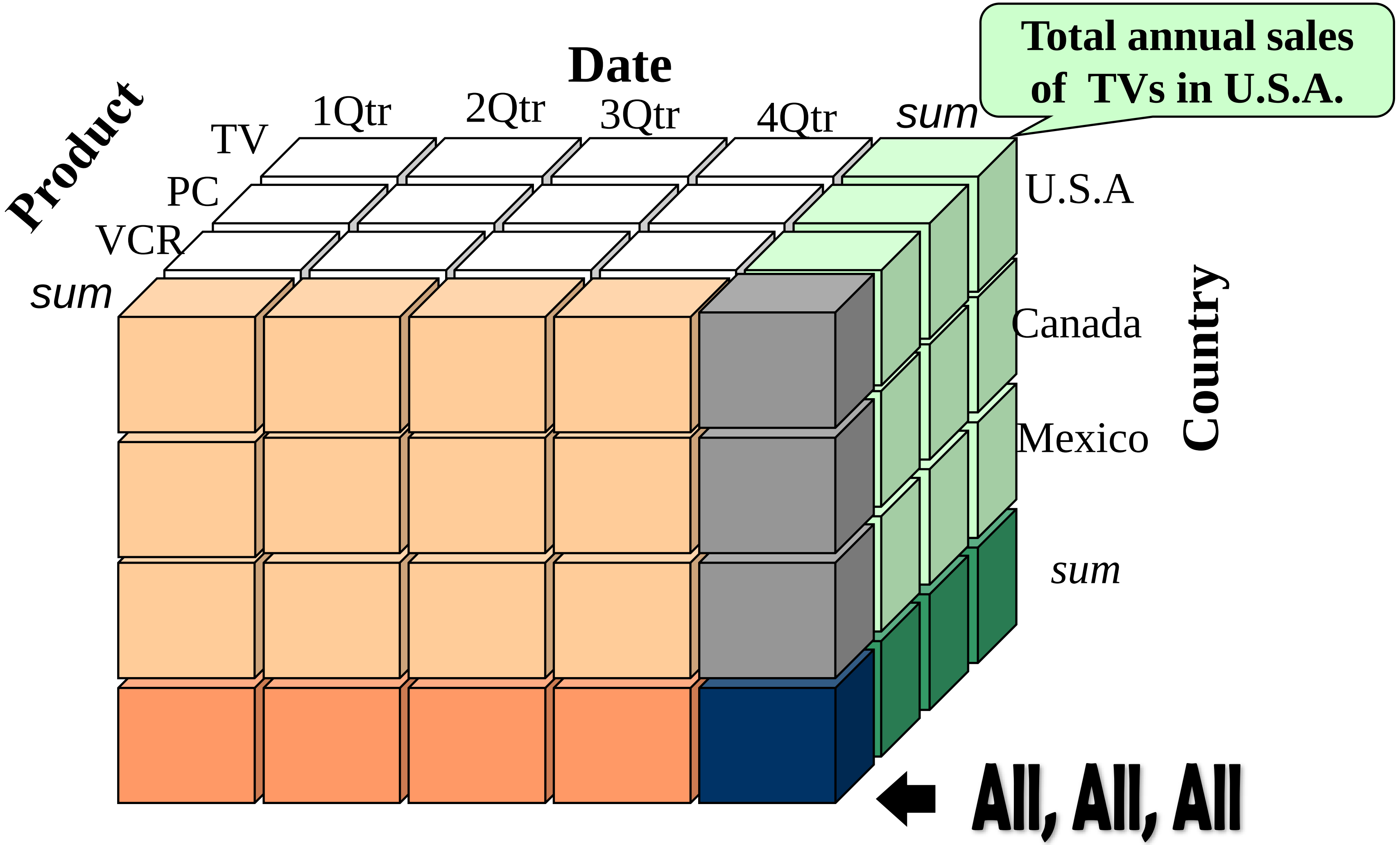
- Sales volume as a function of product, month, and region

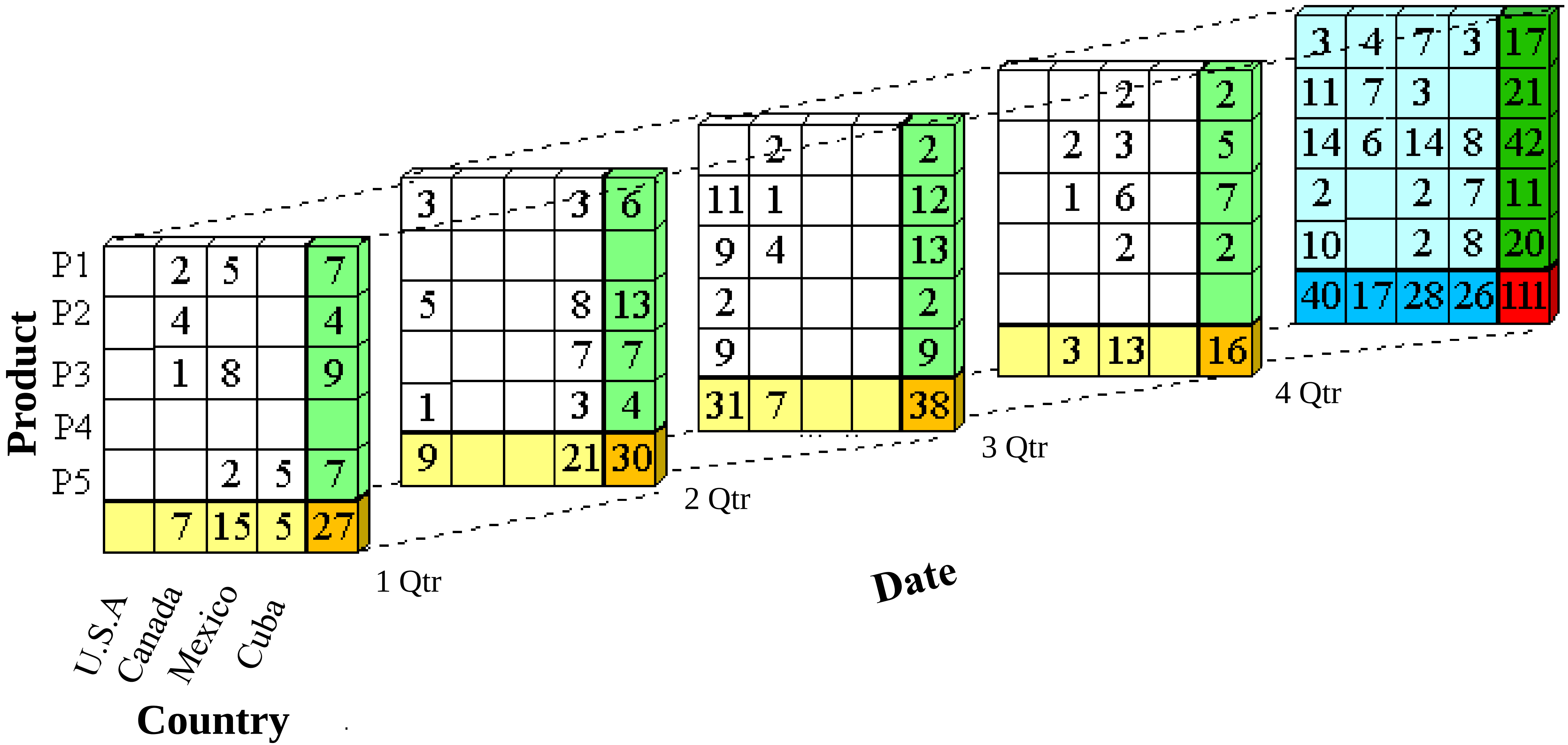


Dimensions: *Product*, *Location*, *Time*
Hierarchical summarization paths

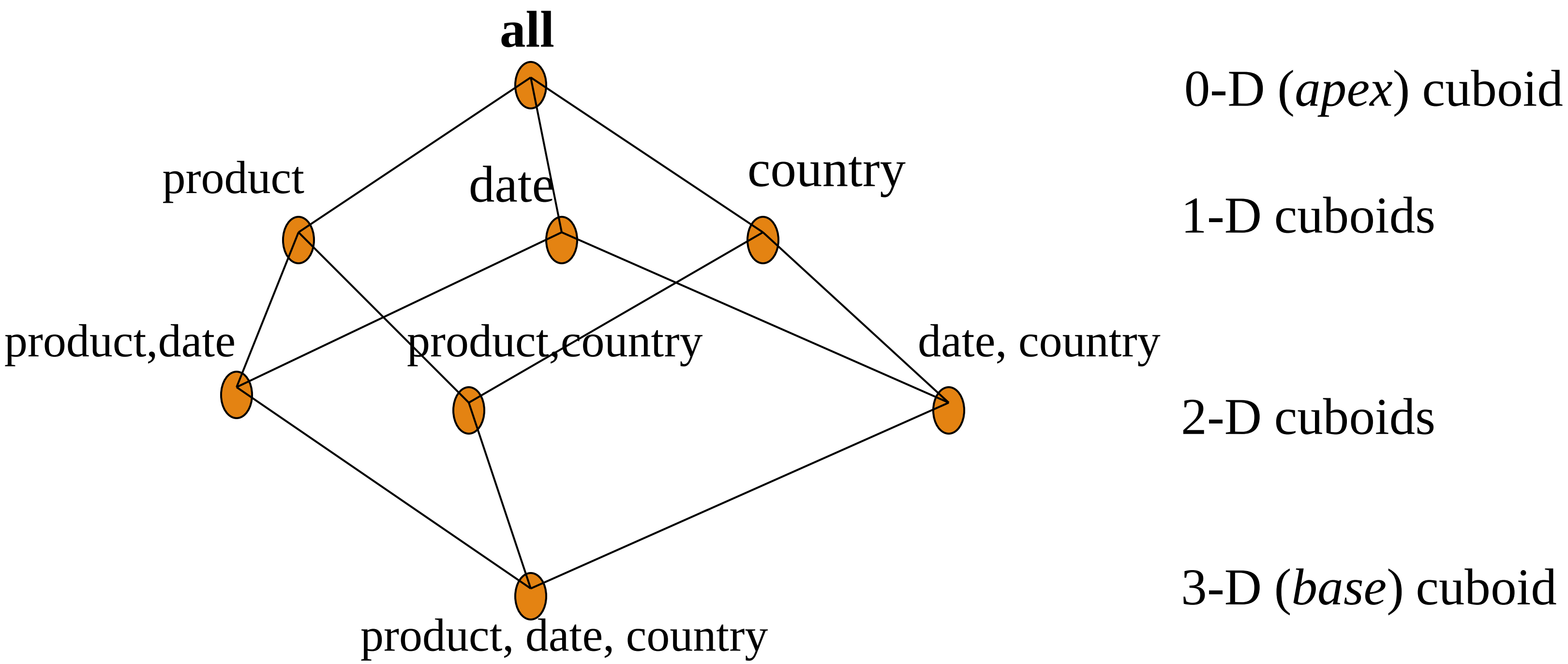


A Sample Data Cube




Product
P1
P2
P3
P4
P5
U.S.A
Canada
Mexico
Cuba
Country
Date
1 Qtr
2 Qtr
3 Qtr
4 Qtr

Cuboids Corresponding to the Cube



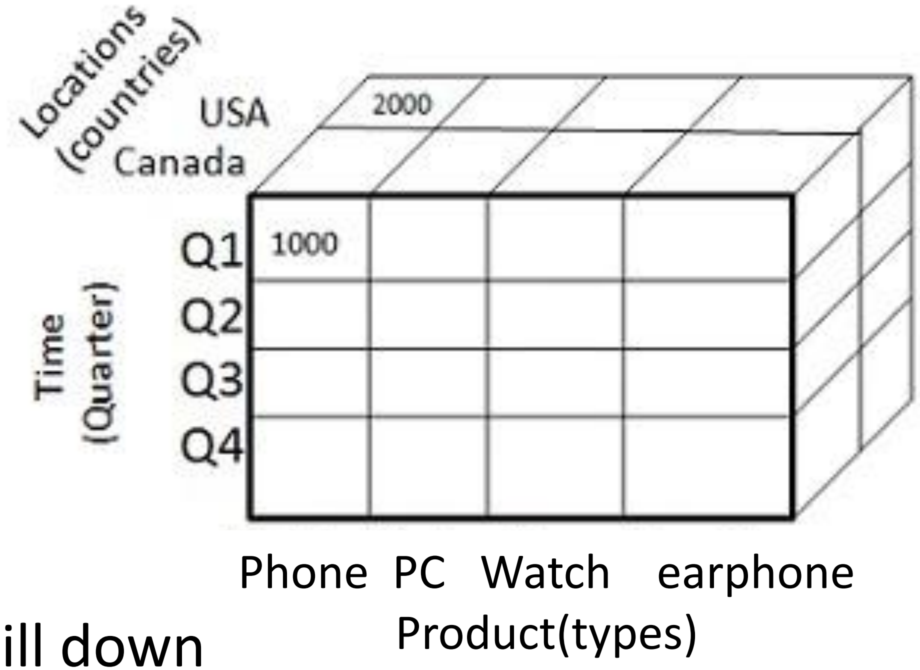
How can we play with the Cube?

Roll up & Drill down

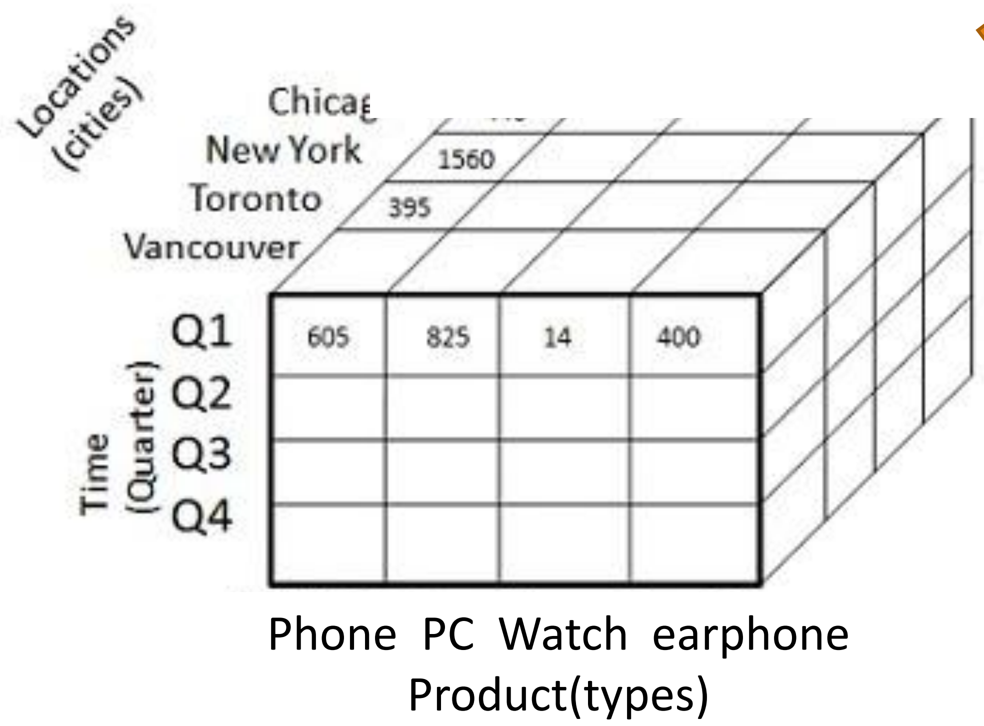
Roll up (drill-up): summarize data

by climbing up hierarchy or by dimension reduction

Roll up



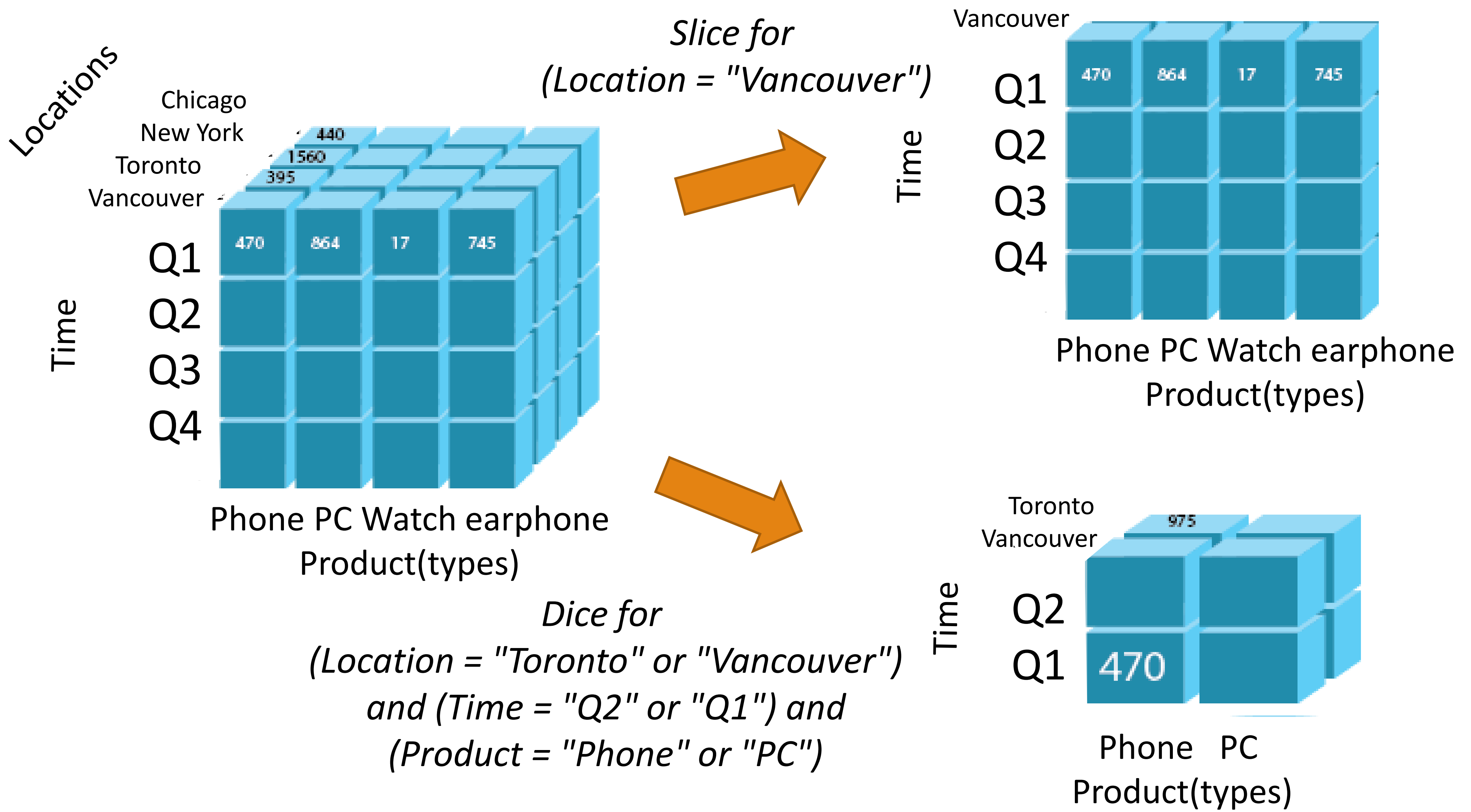
Drill down



Drill down (roll down): reverse of roll-up

from higher level summary to lower level summary or detailed data, or introducing new dimensions

Dice and Slice



Other Typical OLAP Operations

- Pivot (rotate):
  - *reorient the cube, visualization, 3D to series of 2D planes*
- Drill across:
  - *involving (across) more than one fact table*
- Drill through:
  - *through the bottom level of the cube to its back-end relational tables (using SQL)*

Data Cube Measures: Three Categories

- Distributive: if the result derived by applying the function to *n* aggregate values is the same as that derived by applying the function on all the data without partitioning
  - E.g., count(), sum(), min(), max()
- Algebraic: if it can be computed by an algebraic function with *M* arguments (where *M* is a bounded integer), each of which is obtained by applying a distributive aggregate function
  - avg(x) = sum(x) / count(x)
  - How about standard_deviation()

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2 = \frac{1}{n-1}[\sum_{i=1}^{n}x_i^2 - \frac{1}{n}(\sum_{i=1}^{n}x_i)^2]$$

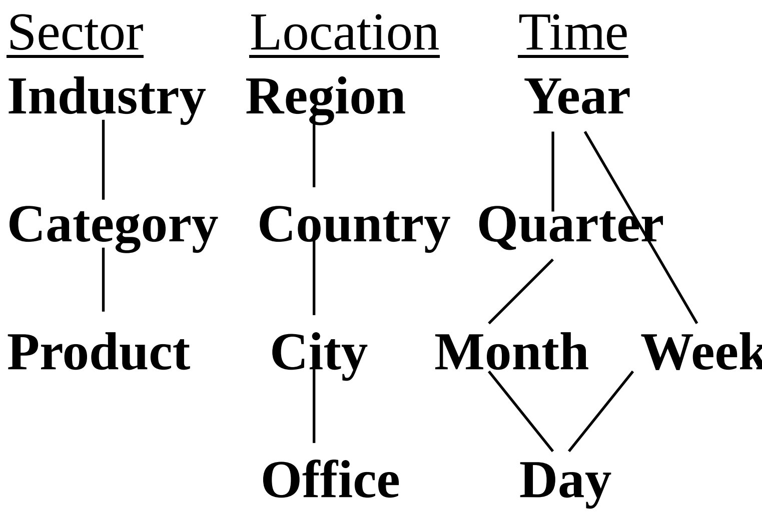
- Holistic: if there is no constant bound on the storage size needed to describe a subaggregate.
  - E.g., median(), mode(), rank()

Efficient Data Cube Computation

- If I have ***n*** dimensions, each with L_i levels, how many cuboids are needed to preprocess all?
- Calculating **all** cuboids is costly in computation and time.
- How to decide which cuboid be pre-calculated (Materialization)?
  - Based on size of data, sharing, access frequency, etc.
  - Example: I know my users always search by Quarter, so that cuboid should be pre-calculated.
  - Example: If I pre-calculate days, I can use days as input to Months (30 or 31 days), or weeks (7 days), etc.

Why this formula?

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



Indexing OLAP Data

- **Indexing**
  - Main purpose of indexing is to make the calculation faster/efficient
- **Common Warehouse Index: Bitmap Index**
  - **Benefits in Warehousing:**
    - Reduced response time for large classes of ad hoc queries.
    - Reduced storage requirements compared to other indexing techniques.
    - Dramatic performance gains even on hardware with a relatively small number of CPUs or a small amount of memory.

https://docs.oracle.com/database/121/DWHSG/schemas.htm#DWHSG9041

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. (WHY?)
- A recent bit compression technique, Word-Aligned Hybrid (WAH), makes it work for high cardinality domain as well [Wu, et al. TODS'06]

Base table

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

Index on Region

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

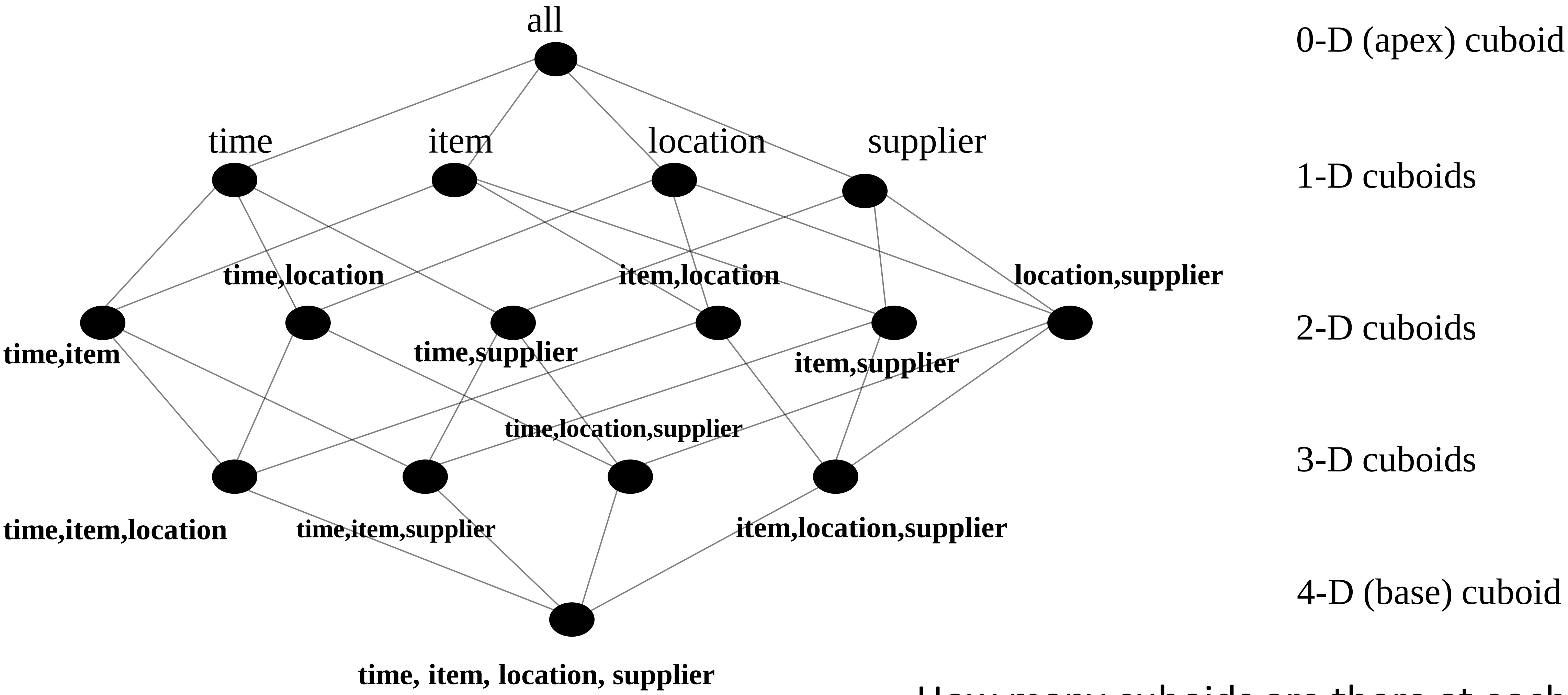
Index on Type

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

Chapter 3: Data Warehousing and On-line Analytical Processing

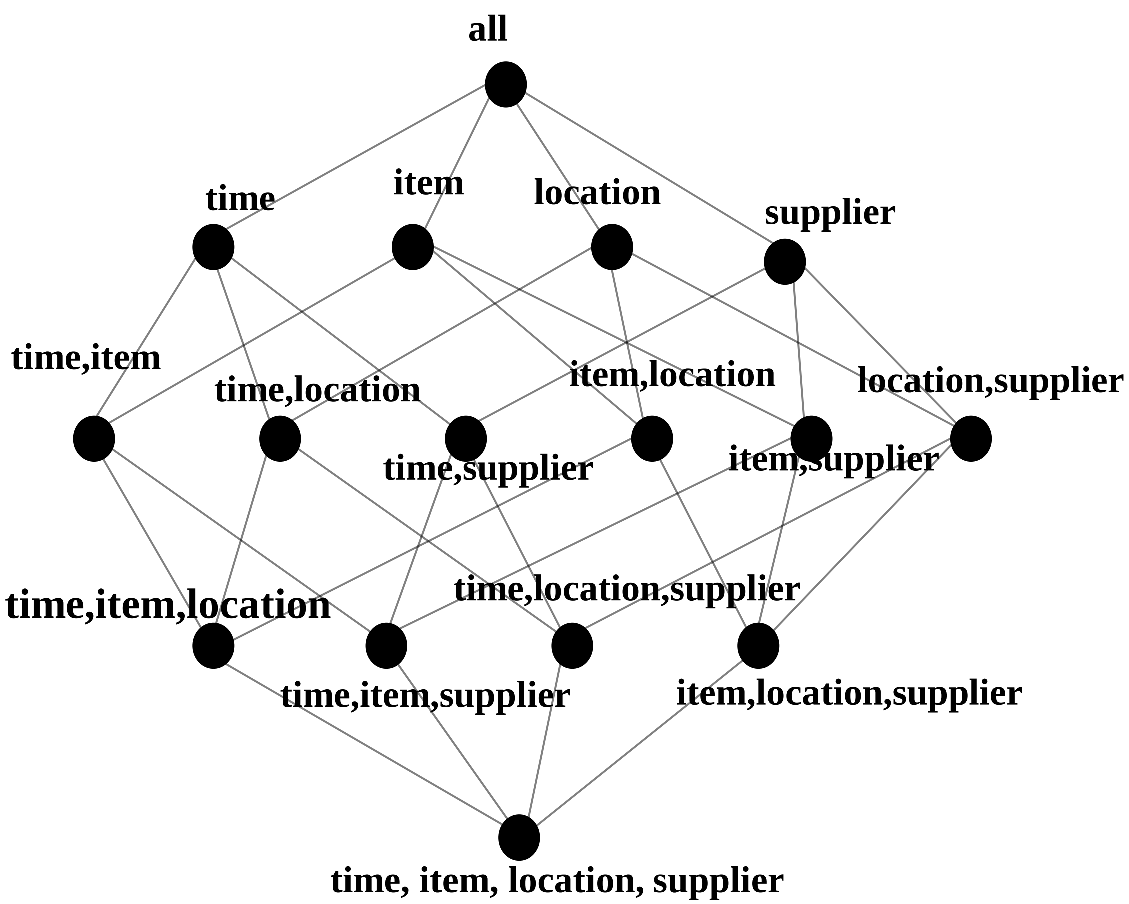
- Data Warehouse: Basic Concepts
- Data Warehouse Modeling
- OLAP Operations
- Data Cube Computation: Concepts and Methods
- Summary

Data Cube: A Lattice of Cuboids



How many cuboids are there at each level?

Data Cube: A Lattice of Cuboids



- Base vs. aggregate cells
- Ancestor vs. descendant cells
- Parent vs. child cells

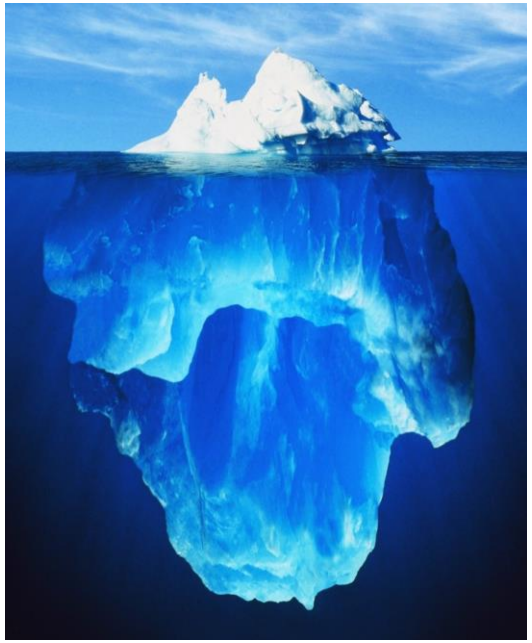
0-D (agg)	(*,*,*,*)
1-D (agg)	(*, milk, *, *)
2-D (agg)	(*, milk, Urbana, *)
2-D (agg)	(*, milk, Chicago, *)
3-D (agg)	(9/15, milk, Urbana, *)
4-D (base)	(9/15, milk, Urbana, Dairy_land)

Cube Materialization: Full Cube vs. Iceberg Cube

- Full cube vs. iceberg cube

  compute cube *sales_iceberg* **as**
  SELECT *month, city, customer_group*, **COUNT(*)**
  FROM *salesInfo*
  CUBE BY *month, city, customer_group*
  HAVING count(*) >= min support

  iceberg condition



- Compute *only* the cells whose measure satisfies the iceberg condition
  - Ex.: Show only those cells whose count is at least 100
- Only a small portion of cells may be "above the water'' in a sparse cube

Why Iceberg Cube?

- No need to save nor show those cells whose value is below the threshold (iceberg condition)
- Efficient methods may even avoid computing the un-needed, intermediate cells
- Avoid explosive growth

Example

- Example: A cube with 100 dimensions
  - Suppose it contains only 2 base cells: $\{(a_1, a_2, a_3, \ldots., a_{100}), (a_1, a_2, b_3, \ldots, b_{100})\}$
  - How many aggregate cells if "having count >= 1"?
    - Answer: $(2^{101} - 2) - 4$ (Why?!)

Example

- Example: A cube with 100 dimensions
  - Suppose it contains only 2 base cells: $\{(a_1, a_2, a_3, \ldots., a_{100}), (a_1, a_2, b_3, \ldots, b_{100})\}$
  - What about the iceberg cells, (i,e., with condition: "having count >= 2")?
    - Answer: 4 (Why?!)

Is Iceberg Cube Good Enough? Closed Cube & Cube Shell

- Let cube P have only 2 base cells: $\{(a_1, a_2, a_3 \ldots, a_{100}){:}10, (a_1, a_2, b_3, \ldots, b_{100}){:}10\}$
  - How many cells will the iceberg cube contain if "having count(*) ≥ 10"?
    - Answer: $2^{101} - 4$ (still too big!)
- **Closed cube:**
  - A cell c is ***closed*** if there exists no cell *d*, such that *d* is a descendant of *c*, and *d* has the same measure value as *c*
    - Ex. The same cube P has only 3 closed cells:
      - $\{(a_1, a_2, *, \ldots, *){:}\ 20, (a_1, a_2, a_3 \ldots, a_{100}){:}\ 10, (a_1, a_2, b_3, \ldots, b_{100}){:}\ 10\}$
  - A ***closed cube*** is a cube consisting of only closed cells
- **Cube Shell:** The cuboids involving only a small # of dimensions, e.g., 2
  - Idea: Only compute cube shells, other dimension combinations can be computed on the fly

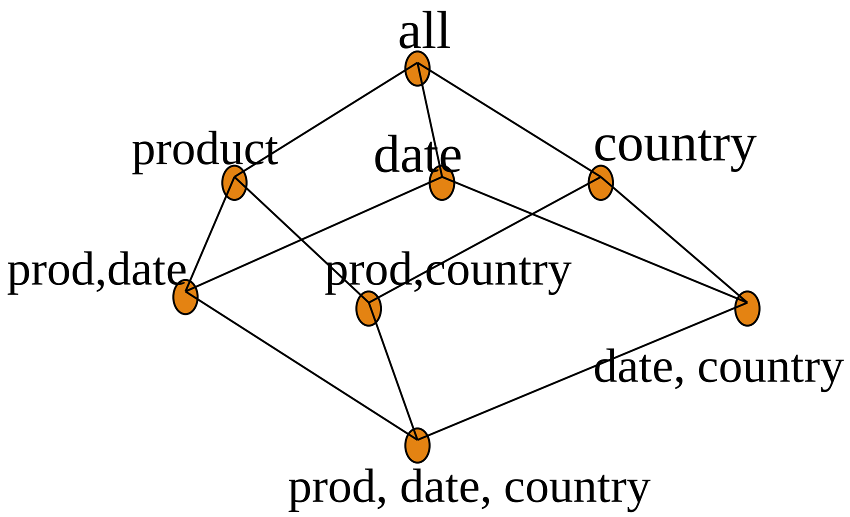
Roadmap for Efficient Computation

- General computation heuristics [1]
- Computing full/iceberg cubes: 3 methodologies
  - Bottom-Up:
    - Multi-Way array aggregation [2]
  - Top-down:
    - BUC [3]
- High-dimensional OLAP:
  - A Shell-Fragment Approach [4]
- Computing alternative kinds of cubes:
  - Partial cube, closed cube, approximate cube, ……

1. (Agarwal et al.'96)
2. (Zhao, Deshpande & Naughton, SIGMOD'97)
3. (Beyer & Ramarkrishnan, SIGMOD'99)
4. (Li, et al. VLDB'04)

Efficient Data Cube Computation: General Heuristics

- Sorting, hashing, and grouping operations are applied
  - **Share-sorts**
  - **Share-partitions**

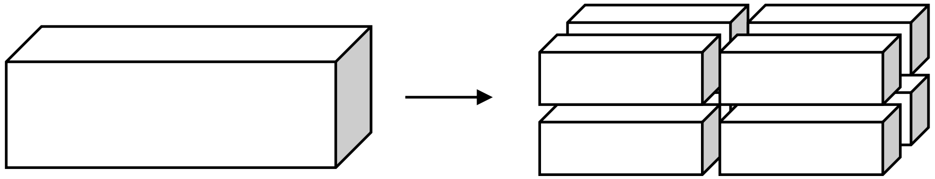


- **Reuse**
  - **Smallest-child:** computing a cuboid from the smallest, previously computed cuboid
  - **Cache-results:** caching results of a cuboid from which other cuboids are computed to reduce disk I/Os
  - **Amortize-scans:** computing as many as possible cuboids at the same time to amortize disk reads

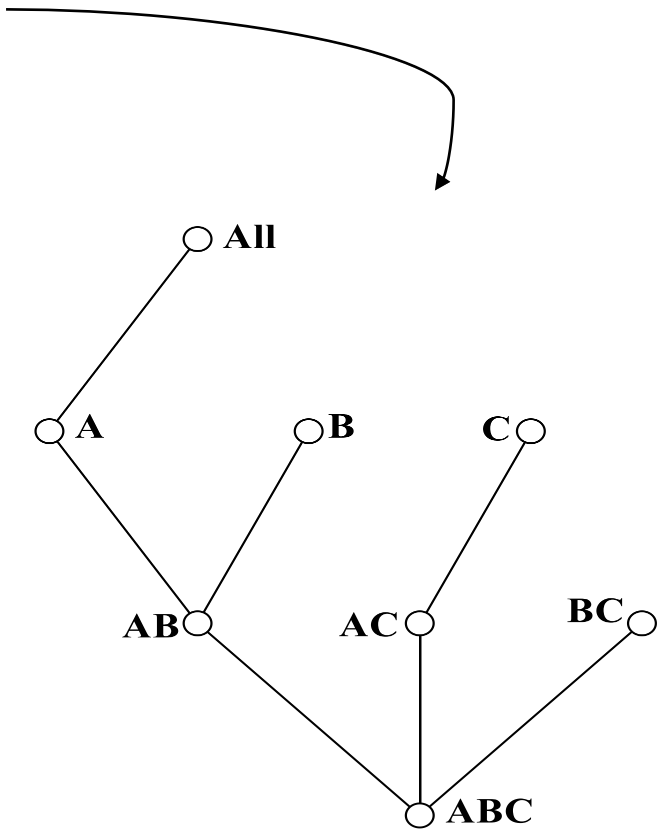
S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, S. Sarawagi. On the computation of multidimensional aggregates. VLDB'96

Multi-Way Array Aggregation (MOLAP)

- How can I efficiently calculate all group-by cell aggregations? Full cube computation
- *Fundamental Concept*: AB, AC, and BC can be computed from ABC.   A, B, and C can be computed from AB/AC/BC.
- *Common Practice with limited memory: Do not* load the entire dimension (in multi-way array form) into memory at once. Use Chunks:



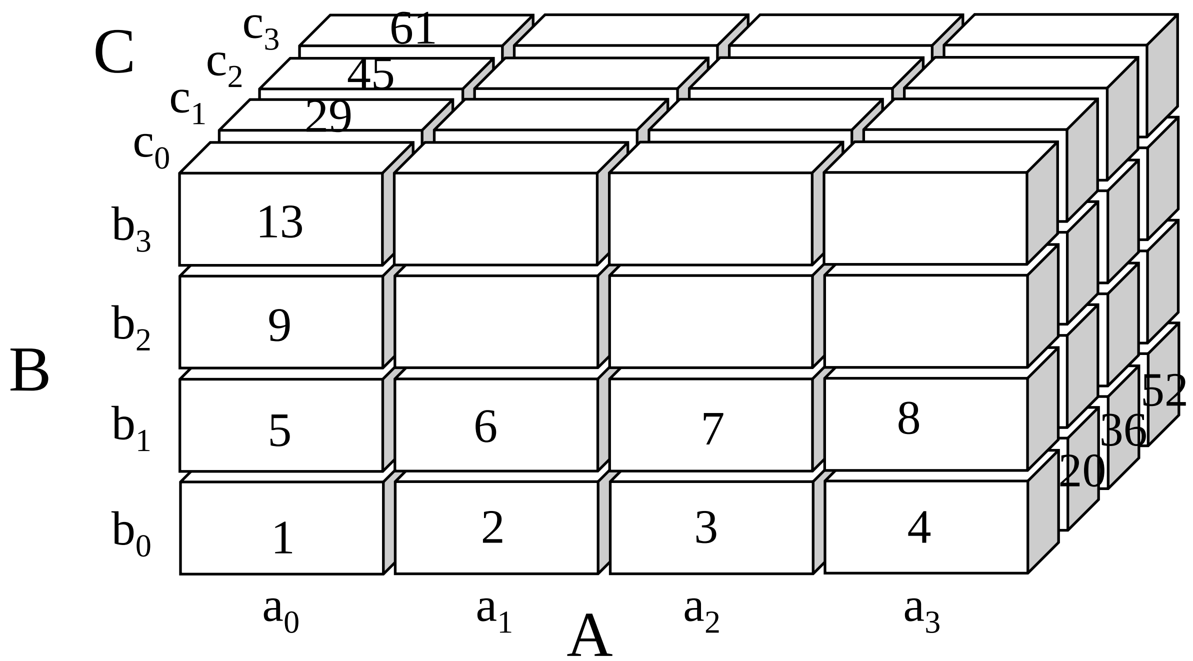
- http://pages.cs.wisc.edu/~nil/764/DADS/38_zhao97arraybased.pdf - Zhao et al. ‘97



Multi-Way Array Aggregation (MOLAP)

- Chunk is stored as(chunk_id, offset)
  - Tells which cells in the chunk have data
- *Goal:* Read chunk only once in memory
  - BC /AB only once
- Example: Student Record Data Warehouse
- *count*(A) > *count* (B) > *count(C*)
- What is best order to put the chunks in order to calculate the aggregation?

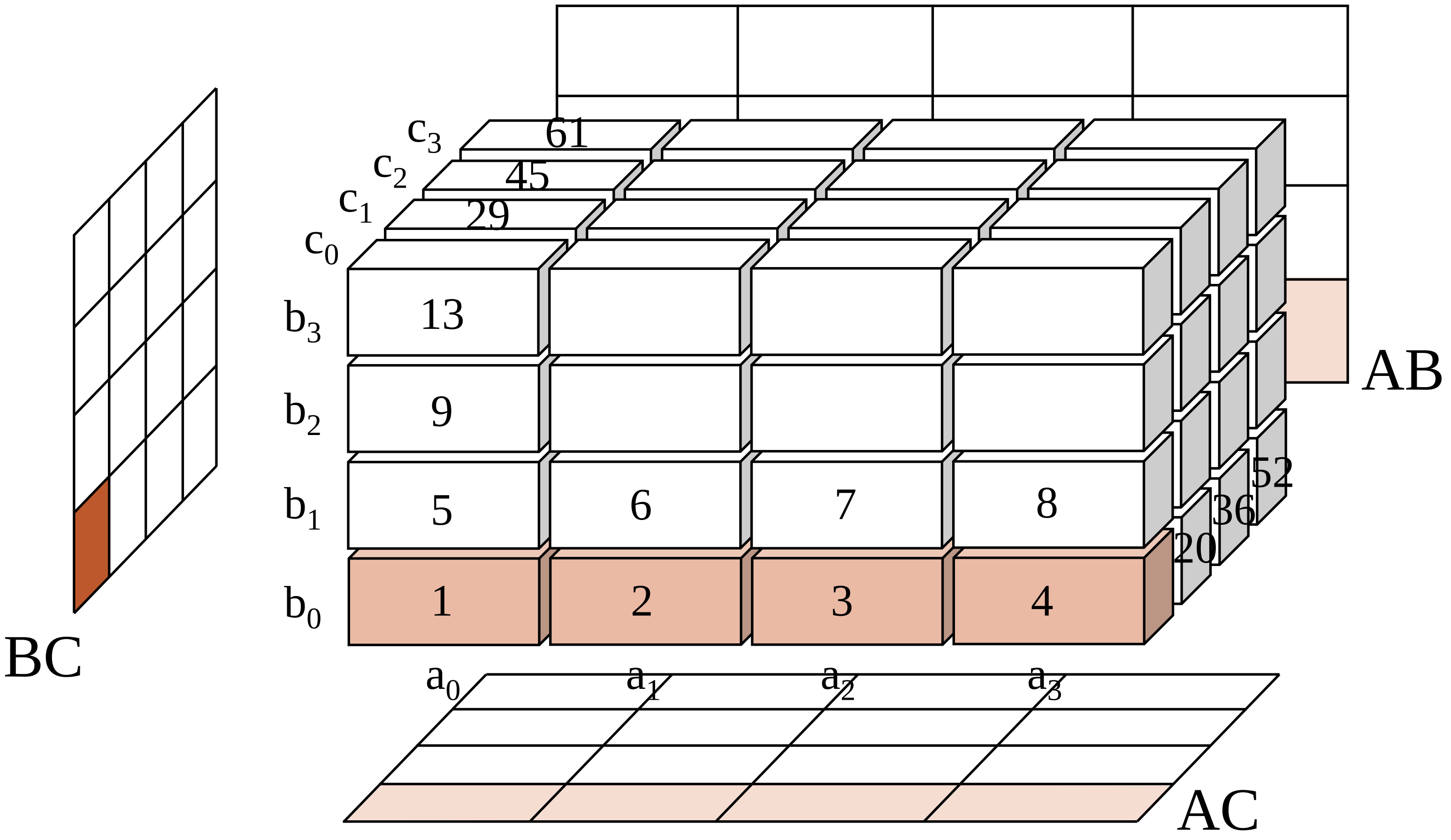
Example:
A: 4000, B: 400, C: 40
Chunk:
1000 x 100 x 10



Cube Computation: Multi-Way Array Aggregation (MOLAP)

- **Scan Order**: 1 – 2 – 3 – 4 – 5 – 6 – …
- ***Goal***: Fully compute chunk only once

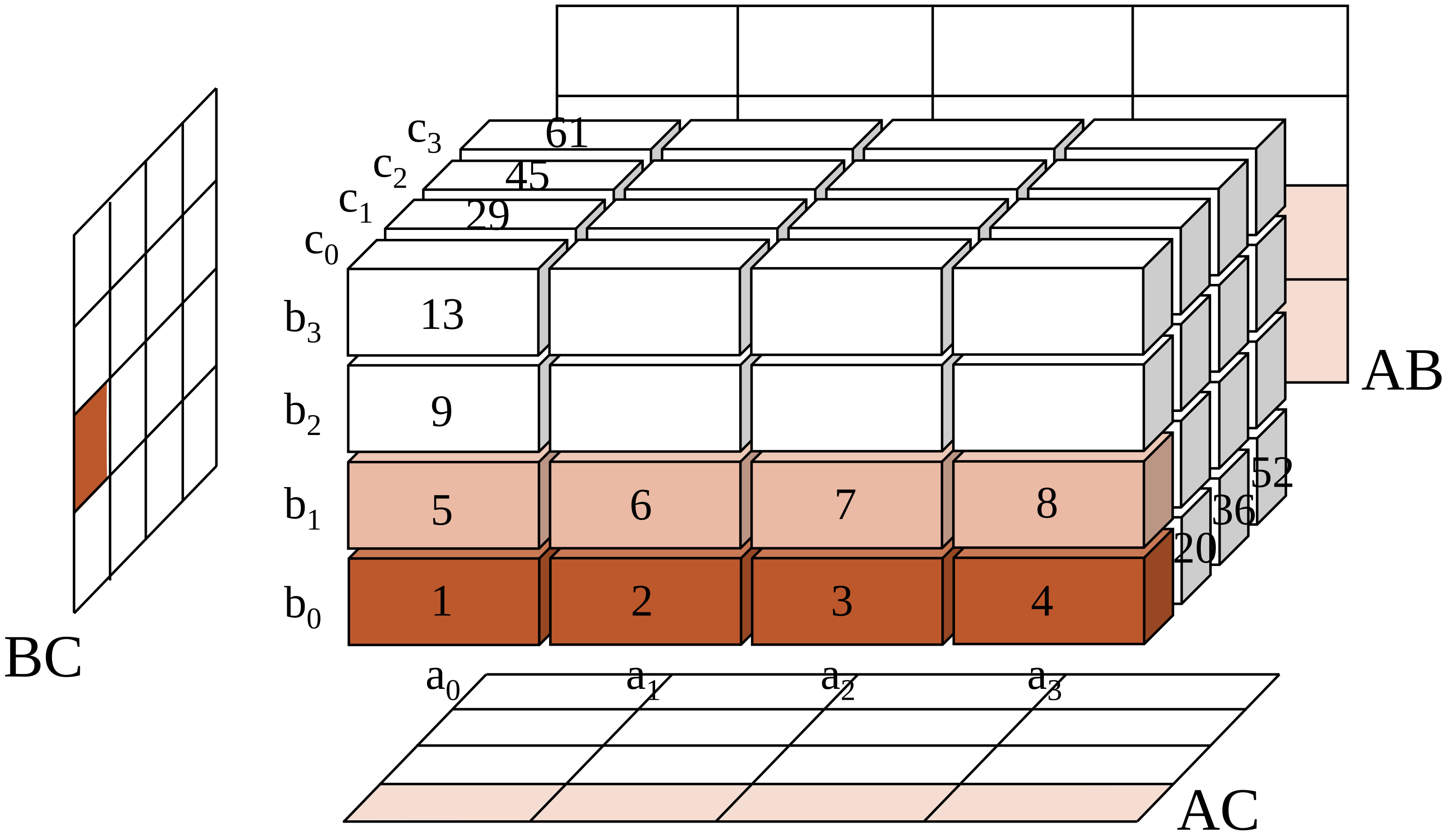
Example:
A: 4000, B: 400, C: 40
Chunk:
1000 x 100 x 10



- While we scan through 1..4
  - One **row** of AC plane is partially computed
  - One **chunk** of BC plane is fully computed (write to file)
  - One row in AB plane is partially computed
- now scan through 5…8

Cube Computation: Multi-Way Array Aggregation (MOLAP)

- **Scan Order**: 1 – 2 – 3 – 4 – 5 – 6 – …
- ***Goal***: Fully compute chunk only once

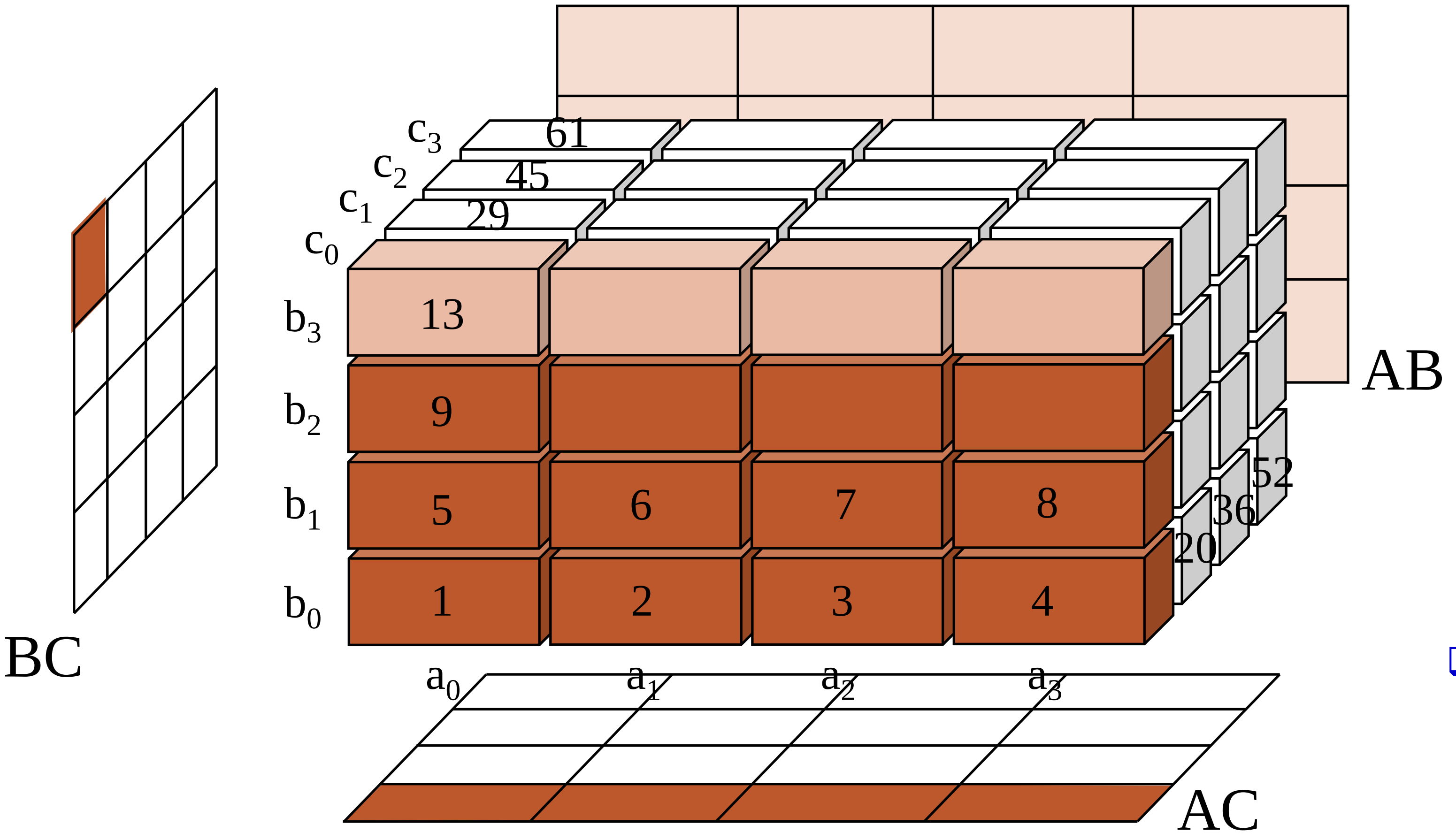


Example:
A: 4000, B: 400, C: 40
Chunk:
1000 x 100 x 10

- While we scan through 5…8
  - Same **row** of AC plane is updated
  - Another **chunk** of BC plane is fully computed (reuse the same place in memory)
  - another row in AB plane is partially computed
- Continue on 9…12
- Continue on 13…16

Cube Computation: Multi-Way Array Aggregation (MOLAP)

- **Scan Order**: 1 – 2 – 3 – 4 – 5 – 6 – …
- ***Goal***: Fully compute chunk only once



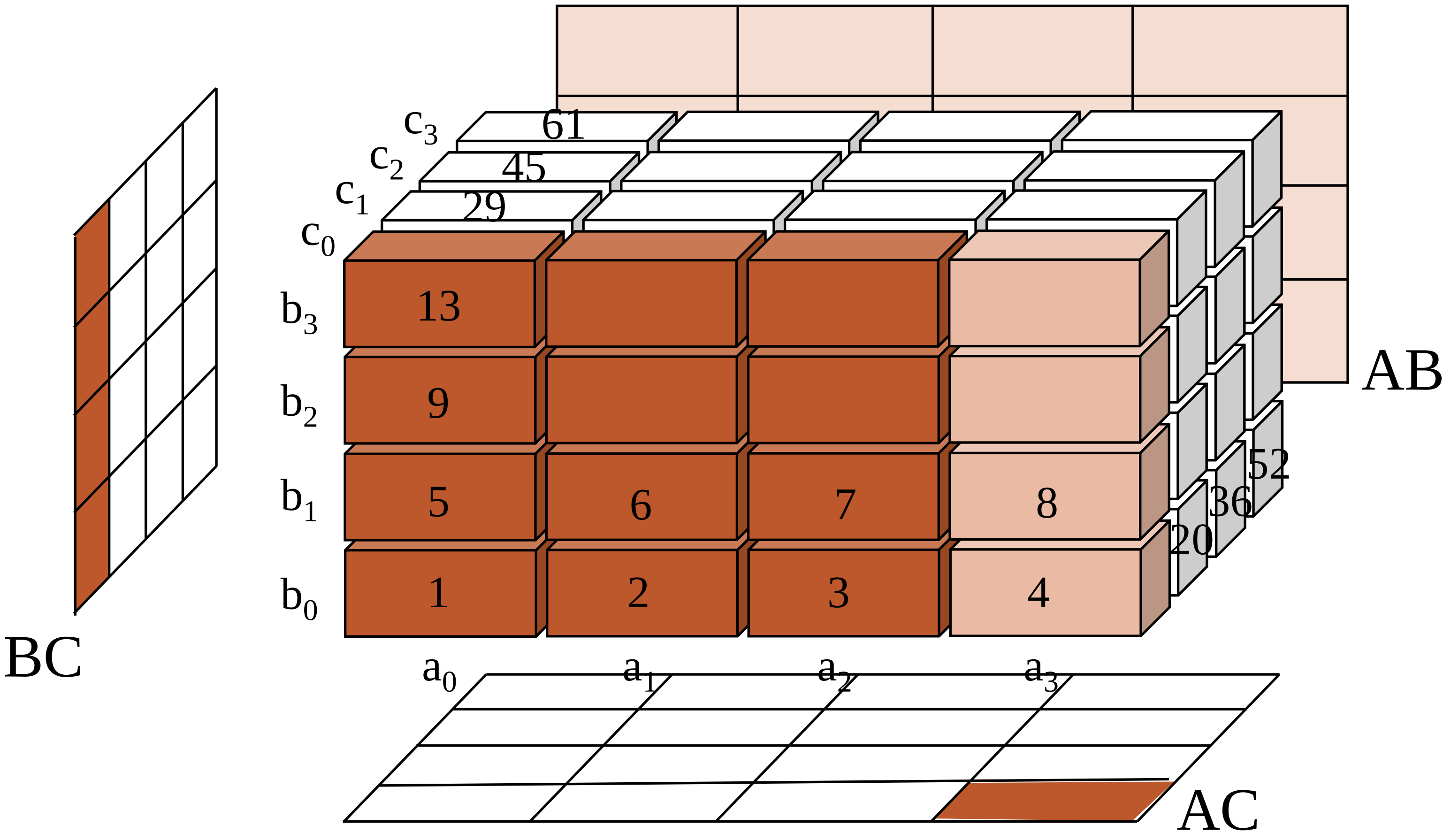
Example:
A: 4000, B: 400, C: 40
Chunk:
1000 x 100 x 10

- While we scan through 13…16
  - One **row** of AC plane is fully computed (write to file)
  - Another **chunk** of BC plane is fully computed (reuse the same place in memory)
  - Whole AB plane is partially computed
- Memory requirement:
  - 4000 x 10 (AC) + 100 x 10 (BC) + 4000 x 400 (AB) = **1,641,000 units**

Cube Computation: Multi-Way Array Aggregation (MOLAP)

- Dimension Order: 1 – 5 – 9 – 13 – 2 – 6 – …

Example:
A: 4000, B: 400, C: 40
Chunk:
1000 x 100 x 10

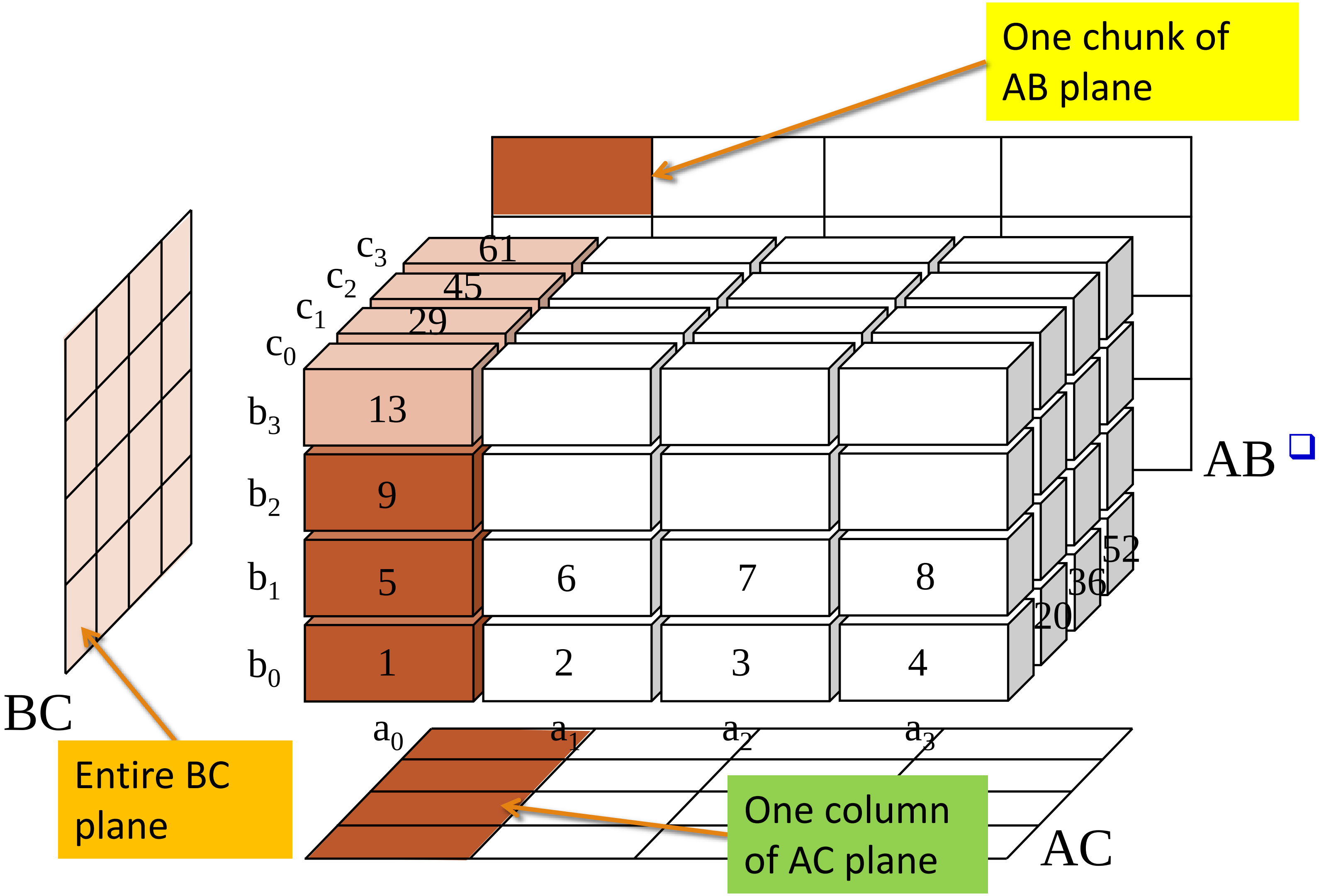


- One **column** of BC plane is fully computed (write to file)
- Another **chunk** of AC plane is fully computed (reuse the same place in memory)
- Whole AB plane is partially computed
- Memory:
  - 400 x 10 (BC) + 1000 x 10 (AC) + 4000 x 400 (AB)
  - **1,614,000 units**

Cube Computation: Multi-Way Array Aggregation (MOLAP)

- Dimension Order: 1 – 17 – … – 13 – 29 – 45 – 61 …

Example:
A: 4000, B: 400, C: 40
Chunk:
1000 x 100 x 10



- One **row** of AC plane
- One **chunk** of AB plane
- All chunks in BC plane
- Memory:
  - 1000 x 40 (AC) + 1000 x 100 (AB) + 400 x 40 (BC)
  - 156,000 units
  - The best order

Cube Computation: Multi-Way Array Aggregation (MOLAP)

- Main Goal of Multi-Way: Reducing memory and I/O
  - How?
    - Keep the smallest plane in main memory
    - Fetch and compute only one chunk at a time for the largest plane
  - The planes should be sorted and computed according to their size in ascending order
  - Suppose A>B>C>…

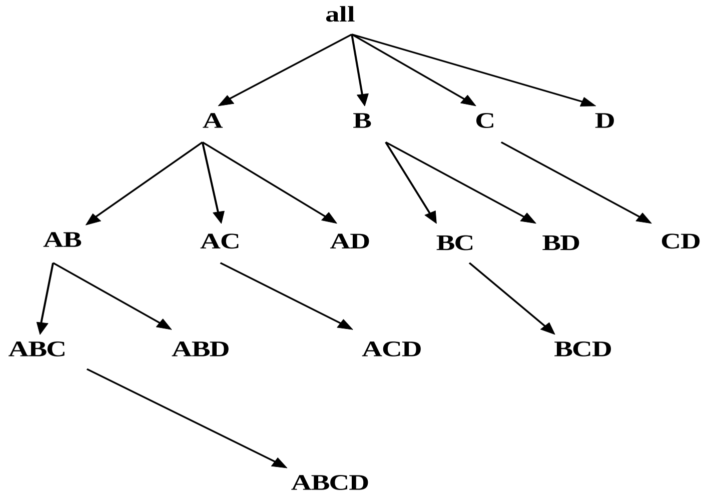
```
for a in A:
    for b in B:
        for c in C: …
```

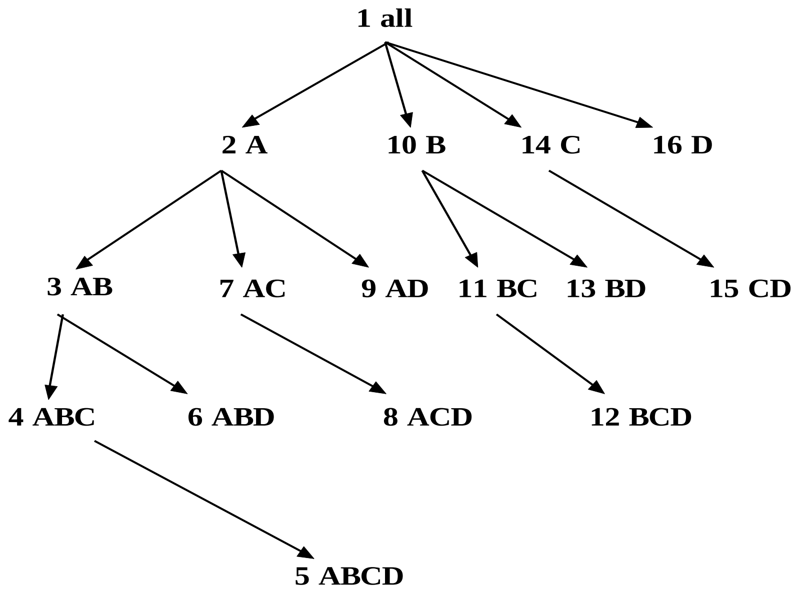
Cube Computation: Multi-Way Array Aggregation (MOLAP)

- Pros and Cons of Multi-Way
  - **Pro:** Efficient for computing the **full cube** for a small number of dimensions
  - **Con:** Can not calculate iceberg cube.
    - i.e: If there are a large number of dimensions, “top-down” computation and iceberg cube computation methods (e.g., BUC) should be used

Cube Computation: Computing in Reverse Order

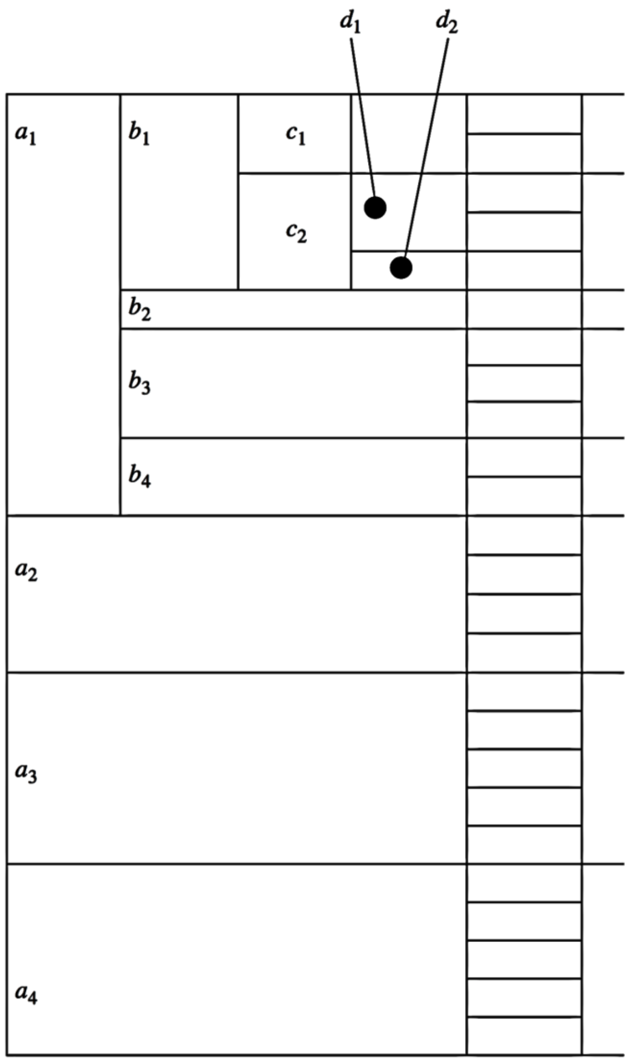
- Iceberg cube computation
- BUC (Beyer & Ramakrishnan, SIGMOD'99)
  - Bottom-Up (cube) Computation
  - "top-down" in our view since we put Apex cuboid on the top!
- Divides dimensions into partitions and facilitates iceberg pruning
  - Prune if not satisfy min_sup
  - If *minsup* = 1, compute full CUBE!
- No simultaneous aggregation



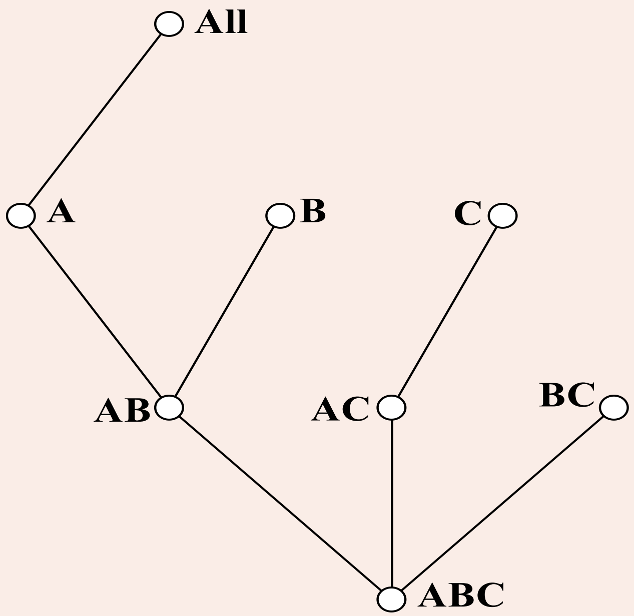
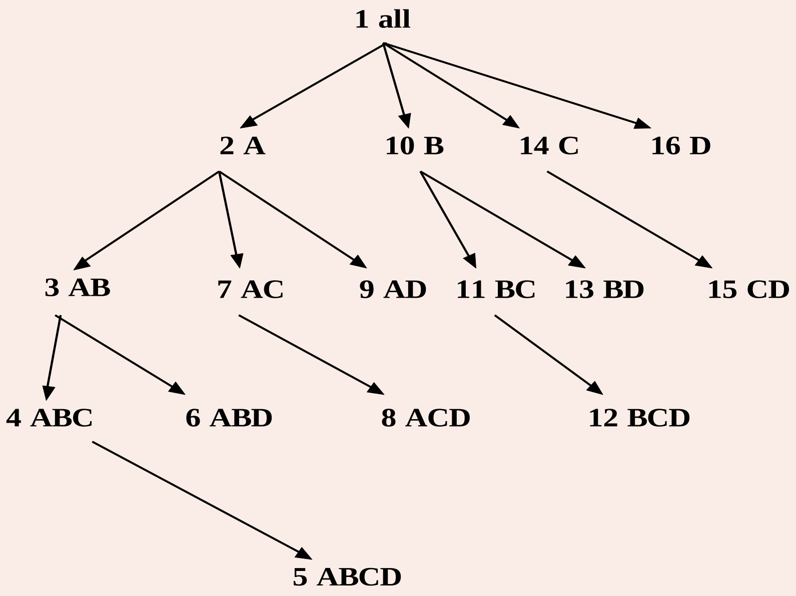


BUC: Partitioning and Aggregating

- Cannot fit in main memory
  - Sort *distinct* values and partition to fit
  - Aggregation when sorting
  - Continue processing
- Iceberg cube
  - If count of (a1, b1, *, *, *) < min_support
  - No need to sort on C

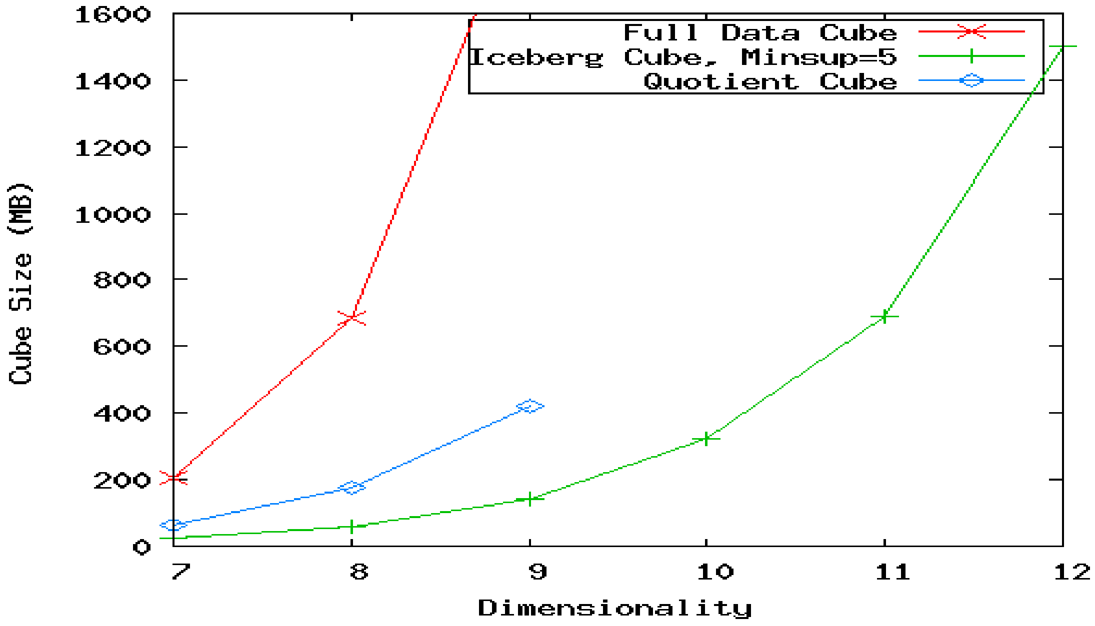


MultiWay VS BUC

	multiway	BUC
Input format	Multi-dimensional array	Relational database
Good for	Full cube	Iceberg cube
Key idea	Simultaneously Aggregation	Partition and sort
Calculation direction		

High-Dimensional OLAP?—The Curse of Dimensionality

- High-D OLAP Applications:
  - E.g. bio-data analysis, statistical surveys
- **None** of the previous cubing method can handle high dimensionality!
  - Iceberg cube and compressed cubes: only delay the inevitable explosion
  - Full materialization: still significant overhead in accessing results on disk
- **A shell-fragment approach**: X. Li, J. Han, and H. Gonzalez, High-Dimensional OLAP: A Minimal Cubing Approach, VLDB'04



A curse of dimensionality: A database of 600k tuples. Each dimension has cardinality of 100 and *zipf* of 2.

Fast High-D OLAP with Minimal Cubing

- Observation: OLAP occurs only on a small subset of dimensions at a time
- Semi-Online Computational Model
  - Partition the set of dimensions into **shell fragments**
  - Compute data cubes for each shell fragment while retaining **inverted indices**
  - Given the pre-computed **fragment cubes**, dynamically compute cube cells of the high-dimensional data cube *online*
- Major idea: Tradeoff between the amount of pre-computation and the speed of online computation
  - Reducing computing high-dimensional cube into precomputing a set of lower dimensional cubes
  - Online re-construction of original high-dimensional space
  - Lossless reduction

Computing a 5-D Cube with 2-Shell Fragments

- Example: Let the cube aggregation function be **count**

TID	**A**	**B**	**C**	**D**	**E**
1	a1	b1	c1	d1	e1
2	a1	b2	c1	d2	e1
3	a1	b2	c1	d1	e2
4	a2	b1	c1	d1	e2
5	a2	b1	c1	d1	e3

- Divide the 5-D table into 2 shell fragments:
  - (A, B, C) and (D, E)
- Build traditional invert index (**1-D**)

Attribute Value	TID List	List Size
a1	1 2 3	3
a2	4 5	2
b1	1 4 5	3
b2	2 3	2
c1	1 2 3 4 5	5
d1	1 3 4 5	4
d2	2	1
e1	1 2	2
e2	3 4	2
e3	5	1

Shell Fragment Cubes: Ideas

- Generalize the **1-D** inverted indices to **multi-dimensional** ones in the data cube sense
- Compute all cuboids for data cubes ABC and DE while retaining the inverted indices
  - Ex. shell fragment cube ABC contains 7 cuboids:
    - A, B, C; AB, AC, BC; ABC
- This completes the offline computation

Attribute Value	TID List	List Size
a1	1 2 3	3
a2	4 5	2
b1	1 4 5	3
b2	2 3	2
c1	1 2 3 4 5	5
d1	1 3 4 5	4
d2	2	1
e1	1 2	2
e2	3 4	2
e3	5	1

Shell-fragment AB

Cell	Intersection	TID List	List Size
a1 b1	1 2 3 ∩ 1 4 5	1	1
a1 b2	1 2 3 ∩ 2 3	2 3	2
a2 b1	4 5 ∩ 1 4 5	4 5	2
a2 b2	4 5 ∩ 2 3	ϕ	0

- ID_Measure Table
  - If measures other than count are present, store in *ID_measure* table separate from the shell fragments

tid	count	sum
1	5	70
2	3	10
3	8	20
4	5	40
5	2	30

Shell Fragment Cubes: Size and Design

- Given a database of T tuples, D dimensions, and F shell fragment size, the fragment cubes' space requirement is: $$O\left(T\left\lceil\frac{D}{F}\right\rceil(2^F-1)\right)$$
  - For F < 5, the growth is sub-linear
- Fragment groupings can be arbitrary to allow for maximum online performance
  - Known common combinations (e.g.,<city, state>) should be grouped together
- Shell fragment sizes can be adjusted for optimal balance between offline and online computation
- Shell fragments do not have to be disjoint

Attribute Value	TID List	List Size
a1	1 2 3	3
a2	4 5	2
b1	1 4 5	3
b2	2 3	2
c1	1 2 3 4 5	5
d1	1 3 4 5	4
d2	2	1
e1	1 2	2
e2	3 4	2
e3	5	1

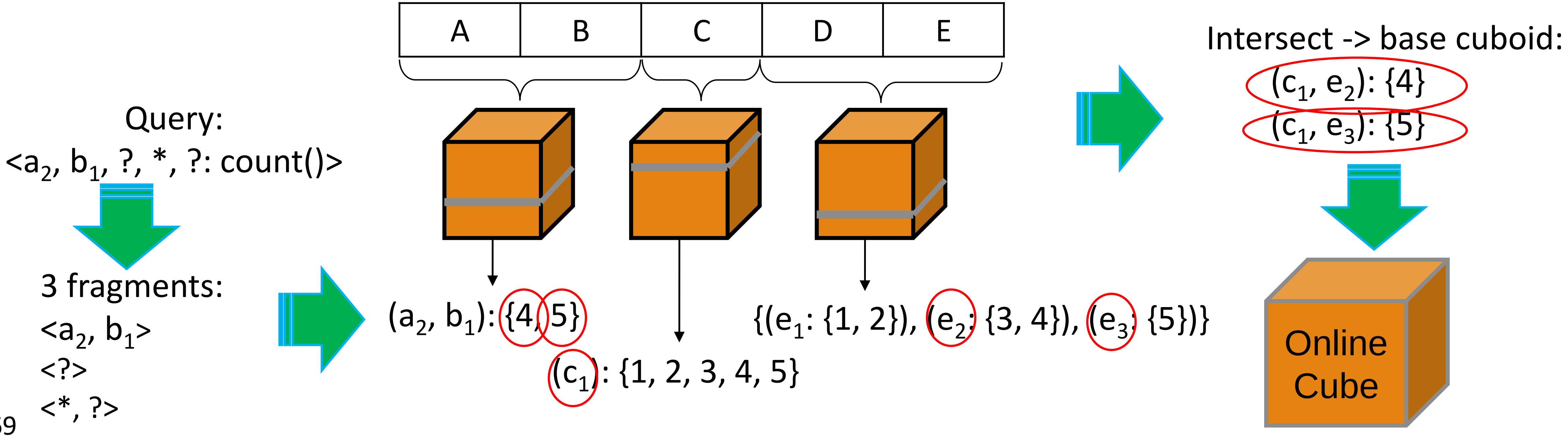
Cell	Intersection	TID List	List Size
a1 b1	1 2 3 ∩ 1 4 5	1	1
a1 b2	1 2 3 ∩ 2 3	2 3	2
a2 b1	4 5 ∩ 1 4 5	4 5	2
a2 b2	4 5 ∩ 2 3	ϕ	0

Online Query Computation with Shell-Fragments

- A query has the general form: $<a_1, a_2, \ldots, a_n: M>$
- Each a_i has 3 possible values
  - Instantiated value– this is what we want to look at
  - Inquire ? Function – want to analyze these dimensions
  - Aggregate * function – don't care about these dimensions
- Ex: Suppose we want to query student data for junior (year 3) students and want to compare scores for different genders and ages, but don't care about what high school they attended.
  - <3, ?, ?, *: count>

Online Query Computation with Shell-Fragments

- Method: Given the materialized fragment cubes, process a query as follows
  - Divide the query into fragments, same as the shell-fragment
  - Fetch the corresponding TID list for each fragment from the fragment cube
  - Intersect the TID lists from each fragment to construct **instantiated base table**
  - Compute the data cube using the base table with any cubing algorithm



Chapter 3: Data Warehousing and On-line Analytical Processing

- Data Warehouse: Basic Concepts
- Data Warehouse Modeling
- OLAP Operations
- Data Cube Computation: Concepts and Methods
- Summary

Summary

- Data warehousing: A multi-dimensional model of a data warehouse
- Data Warehouse Modeling
  - Data Cube: a multidimensional data model
  - Star schema, snowflake schema, fact constellations
- OLAP operations: drilling, rolling, slicing, dicing and pivoting
- Data Cube Computation:
  - Basic Concepts: cuboids; iceberg cube; closed cube and cube shell, OLAP servers
  - Computation Methods: MultiWay Array Aggregation, BUC, High-Dimensional OLAP with Shell-Fragments

References (I)

- S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, and S. Sarawagi. On the computation of multidimensional aggregates. VLDB'96
- D. Agrawal, A. E. Abbadi, A. Singh, and T. Yurek. Efficient view maintenance in data warehouses. SIGMOD'97
- R. Agrawal, A. Gupta, and S. Sarawagi. Modeling multidimensional databases. ICDE'97
- S. Chaudhuri and U. Dayal. An overview of data warehousing and OLAP technology. ACM SIGMOD Record, 26:65-74, 1997
- J. Gray, et al. Data cube: A relational aggregation operator generalizing group-by, cross-tab and sub-totals. Data Mining and Knowledge Discovery, 1:29-54, 1997.
- A. Gupta and I. S. Mumick. Materialized Views: Techniques, Implementations, and Applications. MIT Press, 1999
- J. Han. Towards on-line analytical mining in large databases. *SIGMOD Record*, 1998
- V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. SIGMOD'96

References (II)

- C. Imhoff, N. Galemmo, and J. G. Geiger. Mastering Data Warehouse Design: Relational and Dimensional Techniques. John Wiley, 2003
- W. H. Inmon. Building the Data Warehouse. John Wiley, 1996
- R. Kimball and M. Ross. The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling. 2ed. John Wiley, 2002
- P. O'Neil and D. Quass. Improved query performance with variant indexes. SIGMOD'97
- S. Sarawagi and M. Stonebraker. Efficient organization of large multidimensional arrays. ICDE'94
- P. Valduriez. Join indices. ACM Trans. Database Systems, 12:218-246, 1987.
- J. Widom. Research problems in data warehousing. CIKM’95.
- K. Wu, E. Otoo, and A. Shoshani, Optimal Bitmap Indices with Efficient Compression, ACM Trans. on Database Systems (TODS), 31(1), 2006, pp. 1-38.

References (III)

- S. Agarwal, R. Agrawal, P. M. Deshpande, A. Gupta, J. F. Naughton, R. Ramakrishnan, and S. Sarawagi. On the computation of multidimensional aggregates. VLDB'96
- K. Beyer and R. Ramakrishnan. Bottom-Up Computation of Sparse and Iceberg CUBEs.. SIGMOD'99
- J. Han, J. Pei, G. Dong, K. Wang. Efficient Computation of Iceberg Cubes With Complex Measures. SIGMOD'01
- L. V. S. Lakshmanan, J. Pei, and J. Han, Quotient Cube: How to Summarize the Semantics of a Data Cube, VLDB'02
- X. Li, J. Han, and H. Gonzalez, High-Dimensional OLAP: A Minimal Cubing Approach, VLDB'04
- X. Li, J. Han, Z. Yin, J.-G. Lee, Y. Sun, "Sampling Cube: A Framework for Statistical OLAP over Sampling Data", SIGMOD'08
- K. Ross and D. Srivastava. Fast computation of sparse datacubes. VLDB'97
- D. Xin, J. Han, X. Li, B. W. Wah, Star-Cubing: Computing Iceberg Cubes by Top-Down and Bottom-Up Integration, VLDB'03
- Y. Zhao, P. M. Deshpande, and J. F. Naughton. An array-based algorithm for simultaneous multidimensional aggregates. SIGMOD'97
- D. Burdick, P. Deshpande, T. S. Jayram, R. Ramakrishnan, and S. Vaithyanathan. OLAP over uncertain and imprecise data. VLDB'05

References (IV)

- R. Agrawal, A. Gupta, and S. Sarawagi. Modeling multidimensional databases. ICDE'97
- B.-C. Chen, L. Chen, Y. Lin, and R. Ramakrishnan. Prediction cubes. VLDB'05
- B.-C. Chen, R. Ramakrishnan, J.W. Shavlik, and P. Tamma. Bellwether analysis: Predicting global aggregates from local regions. VLDB'06
- Y. Chen, G. Dong, J. Han, B. W. Wah, and J. Wang, Multi-Dimensional Regression Analysis of Time-Series Data Streams, VLDB'02
- R. Fagin, R. V. Guha, R. Kumar, J. Novak, D. Sivakumar, and A. Tomkins. Multi-structural databases. PODS'05
- J. Han. Towards on-line analytical mining in large databases. SIGMOD Record, 27:97–107, 1998
- T. Imielinski, L. Khachiyan, and A. Abdulghani. Cubegrades: Generalizing association rules. Data Mining & Knowledge Discovery, 6:219–258, 2002.
- R. Ramakrishnan and B.-C. Chen. Exploratory mining in cube space. Data Mining and Knowledge Discovery, 15:29–54, 2007.
- K. A. Ross, D. Srivastava, and D. Chatziantoniou. Complex aggregation at multiple granularities. EDBT'98
- S. Sarawagi, R. Agrawal, and N. Megiddo. Discovery-driven exploration of OLAP data cubes. EDBT'98
- G. Sathe and S. Sarawagi. Intelligent Rollups in Multidimensional OLAP Data. *VLDB'01*