THE UNIVERSITY OF WESTERN AUSTRALIA

Data Warehousing

Lecture 3 Data Cube Technologies

CITS3401
CITS5504

Zeyi Wen

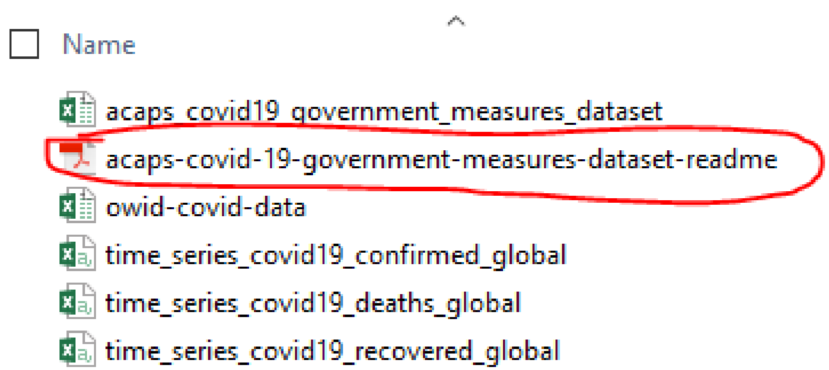
Computer Science and Software Engineering

School of Maths, Physics and Computing

Acknowledgement: The lecture slides are based on online sources.

Project 1

- A new file which is helpful for CITS5504 students.
  - acaps-covid-19-government-measures-dataset-readme.pdf
- 5% software environment evaluation ends next week.
  - Failing in this part is an indicator that you will fail to pass this unit.
- Completing the tasks in lab sheets are crucial!
- Start early!!



Lecture Outline

- Review of Previous lectures
- Data Cube
  - Cuboids
  - Types of Cells
  - Types of Cubes
- Answering Queries with Data Cube
- Storage of Data Cube
  - MOLAP and ROLAP
  - Cube Materialisation
  - Indexing Data to Support OLAP

Lecture 1 & 2

- Understand What is a Data Warehouse
- Differences between OLTP and OLAP
- Star, Snowflake and Galaxy Schema
- Concept Hiearchies
- Roll-up, Drill-down, Slice & Dice, Pivot
- 3-tier Architecture of Data Warehouses

Sample Questions

- What is a data cube? Explain the OLAP operations roll up and drill down in relation to a data cube.

- Explain the concept of a data warehouse and the main steps required for constructing a data warehouse.

- Explain the meaning of star schema and snowflake schema in relation to a data warehouse.

What is Data Warehouse? (Lecture 1)

- **A Data Warehouse is a**
  - subject-oriented,
  - integrated,
  - time-variant, and
  - nonvolatile

  collection of data in support of management’s decision-making process.

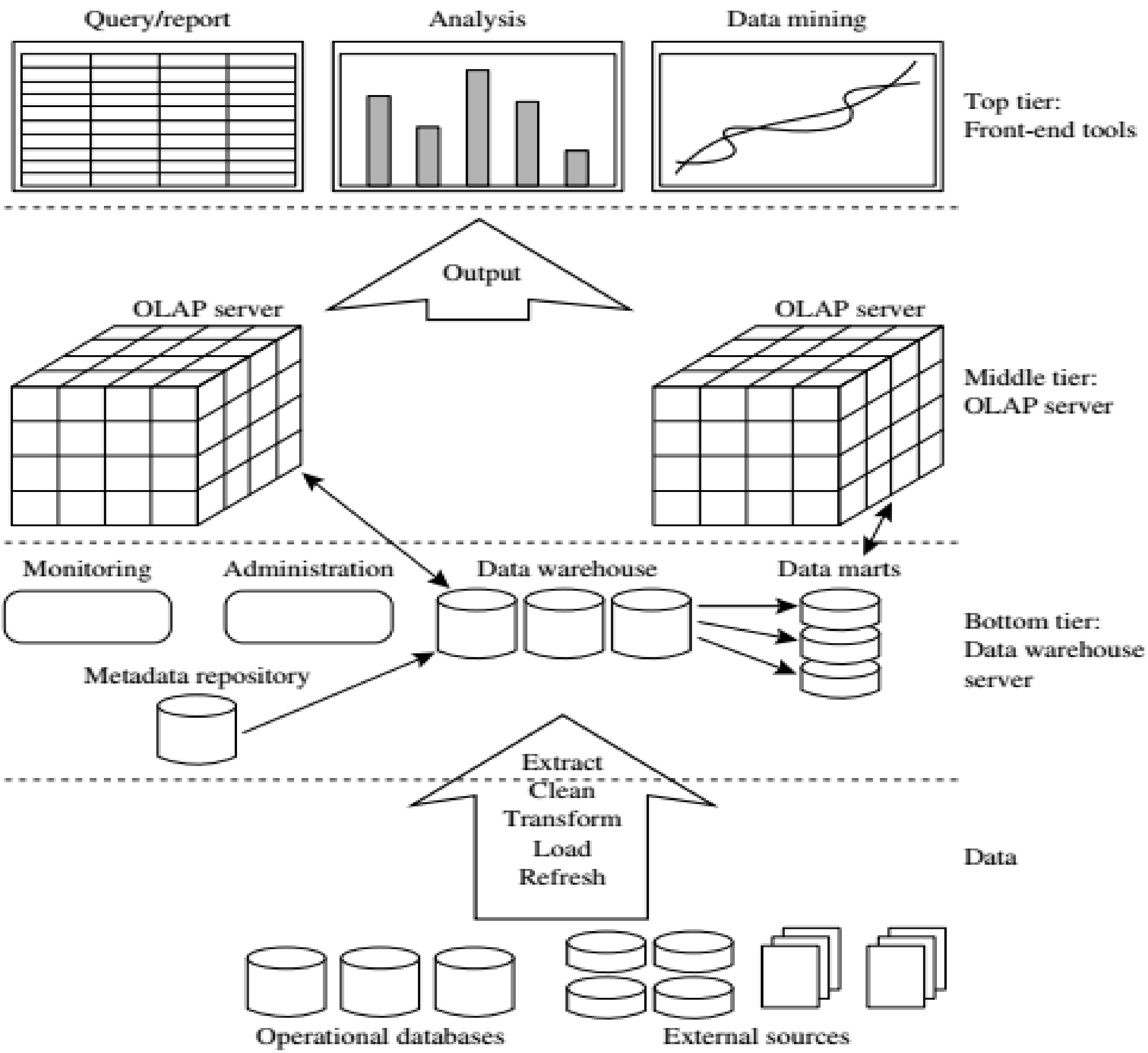
Data Warehouse (OLAP) vs. Operational DBMS (OLTP) (Lecture 1)

THE UNIVERSITY OF WESTERN AUSTRALIA

- **On-Line Transaction Processing (OLTP)**
  - Many short transactions (queries + updates)
  - Examples:
    - Update account balance
    - Enroll in course
    - Add book to shopping cart
  - Queries touch small amounts of data (e.g. a few records)
  - Updates are frequent
  - Concurrency is biggest performance concern

- **On-Line Analytical Processing (OLAP)**
  - Long transactions, complex queries
  - Examples:
    - Report total sales for each department in each month
    - Identify top-selling books
    - Count classes with < 10 students
  - Queries touch large amounts of data
  - Updates are infrequent
  - Individual queries can require lots of resources

A three-tier data warehousing architecture (Lecture 1)



Multi-dimensional View of Data (3-D) (Lecture 2)

THE UNIVERSITY OF WESTERN AUSTRALIA

Table 4.3 3-D View of Sales Data for *AllElectronics* According to *time*, *item*, and *location*

	***location* = "Chicago"**				***location* = "New York"**				***location* = "Toronto"**				***location* = "Vancouver"**			
	item				***item***				***item***				***item***			
time	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*
Q1	854	882	89	623	1087	968	38	872	818	746	43	591	605	825	14	400
Q2	943	890	64	698	1130	1024	41	925	894	769	52	682	680	952	31	512
Q3	1032	924	59	789	1034	1048	45	1002	940	795	58	728	812	1023	30	501
Q4	1129	992	63	870	1142	1091	54	984	978	864	59	784	927	1038	38	580

Note: The measure displayed is *dollars_sold* (in thousands).

Table 4.2 2-D View of Sales Data for *AllElectronics* According to *time* and *item*

	***location* = "Vancouver"**			
	item *(type)*			
time *(quarter)*	*home entertainment*	*computer*	*phone*	*security*
Q1	605	825	14	400
Q2	680	952	31	512
Q3	812	1023	30	501
Q4	927	1038	38	580

Note: The sales are from branches located in the city of Vancouver. The measure displayed is *dollars_sold* (in thousands).

Multi-dimensional View of Data (Data Cube) (Lecture 2)

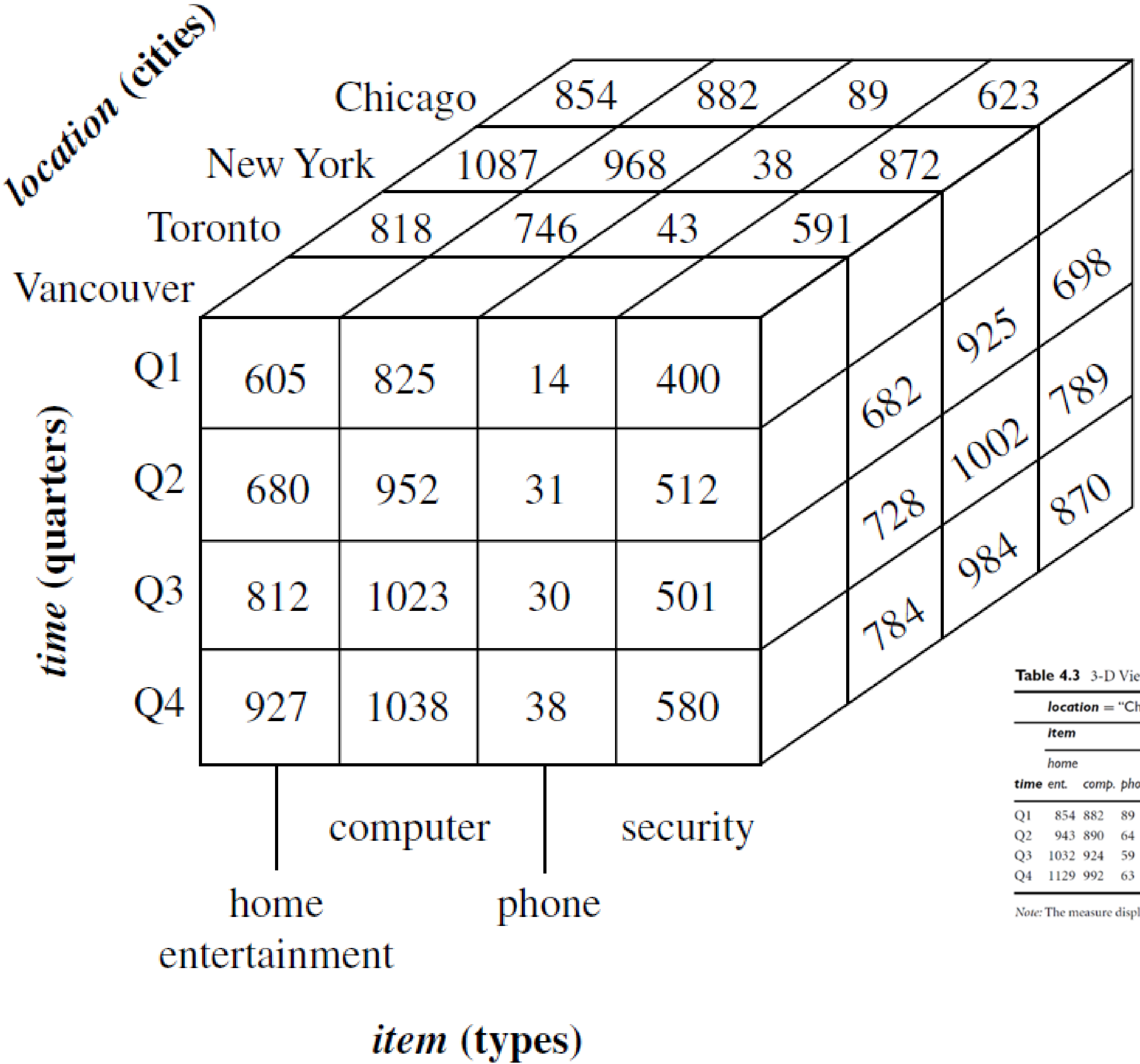


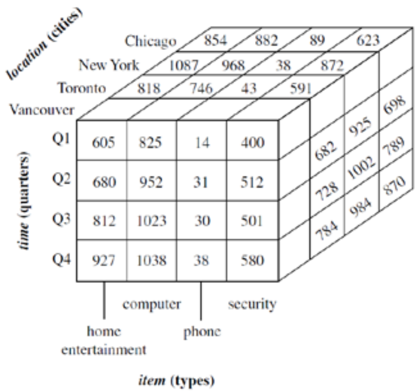
Table 4.3 3-D View of Sales Data for *AllElectronics* According to *time*, *item*, and *location*

	location = "Chicago"				***location*** = "New York"				***location*** = "Toronto"				***location*** = "Vancouver"			
	item				***item***				***item***				***item***			
time	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*	*home ent.*	*comp.*	*phone*	*sec.*
Q1	854	882	89	623	1087	968	38	872	818	746	43	591	605	825	14	400
Q2	943	890	64	698	1130	1024	41	925	894	769	52	682	680	952	31	512
Q3	1032	924	59	789	1034	1048	45	1002	940	795	58	728	812	1023	30	501
Q4	1129	992	63	870	1142	1091	54	984	978	864	59	784	927	1038	38	580

Note: The measure displayed is *dollars_sold* (in thousands).

From Tables to Data Cubes (Lecture 2)

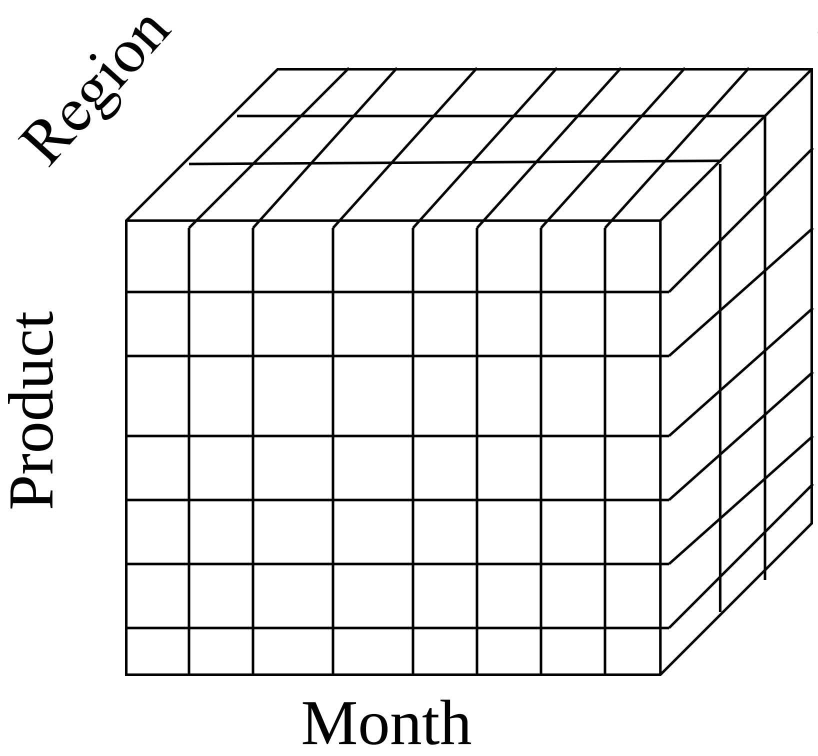
- A data warehouse is based on a multi-dimensional data model which views data in the form of a data cube
- A data cube, is organised around a central theme, such as sales, allows data to be modelled and viewed in multiple dimensions
  - Dimension tables, such as item (item_name, brand, type), or time (day, week, month, quarter, year) or location (branch, city, state, country)
  - Fact table contains measures of central theme (such as dollars_sold, units sold) and keys to each of the related dimension tables

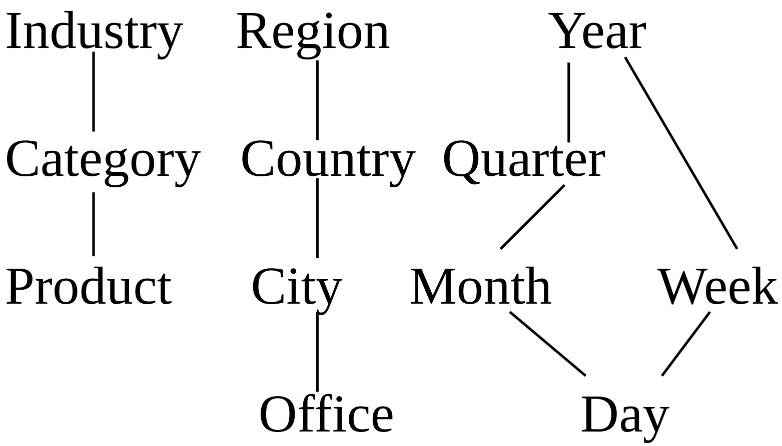


Concept Hierarchy in Dimensions (Lecture 2)

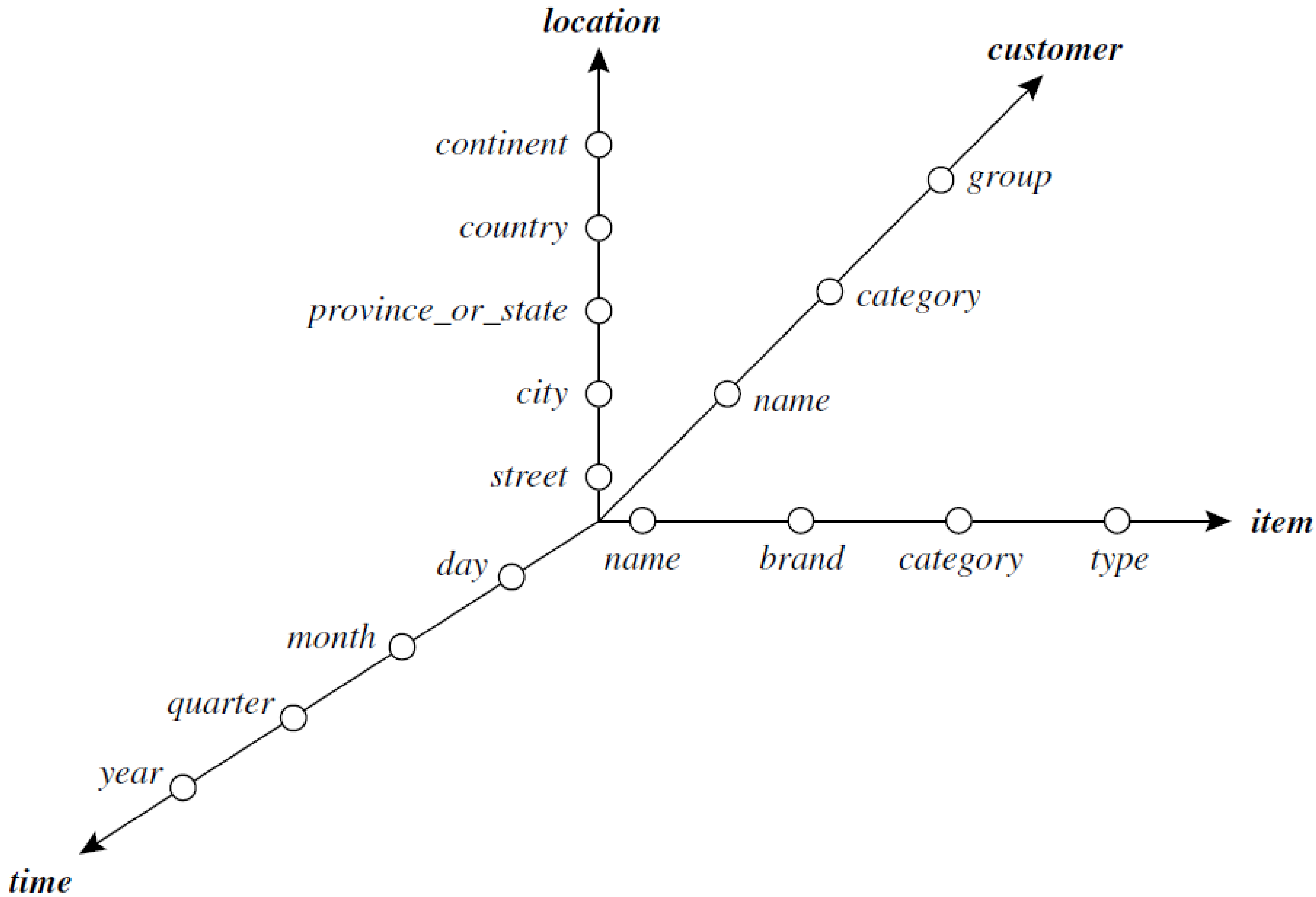
Sales volume as a function of product, month, and region

Dimensions: Product, Location, Time
Hierarchical summarisation paths





A Starnet Model of Business Queries (Lecture 2)



Data Warehouse Design Template (Lecture 2)

THE UNIVERSITY OF WESTERN AUSTRALIA

Kimball's four steps

- Identify a business process to model
  - E.g. orders, invoices, shipments, sales …
- Determine the grain of the business process
  - E.g. individual transactions, individual daily snapshots
- Choose the dimensions that apply to fact table rows
  - Example dimensions are time, item, customer, supplier, transaction type and status
- Identify the measure that populates fact table rows
  - Typical measures are numeric additive quantities like dollars_sold and units_sold

Schema (Lecture 2)

- **Star Schema**
  - A fact table in the middle connected to a set of dimension tables
- **Snowflake Schema**
  - Some dimensional hierarchy is normalised into a set of smaller dimension tables, forming a shape similar to snowflake.
  - Reduces redundancy at the cost of efficiency.
- **Galaxy Schema (Fact Constellation)**
  - Multiple fact tables share dimension tables
  - Viewed as a collection of stars - Galaxy schema

Typical OLAP Operations (Lecture 2)

- **Roll up (drill up): summarise 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*
- **Slice and dice:**
  - *project and select*
- **Pivot (rotate):**
  - *reorient the cube, visualisation, 3D to series of 2D planes.*
- **Other operations (aside)**
  - *drill across: involving (across) multiple fact tables*
  - *drill through: through the bottom level of the cube to its back-end relational tables (using SQL)*

Lecture Outline

- Review of previous lectures

- Data Cube
  - Cuboids
  - Types of Cells
  - Types of Cubes

- Answering Queries with Data Cube

- Storage of data cube
  - MOLAP and ROLAP
  - Cube materialisation
  - Indexing data to support OLAP

What is a Data Cube

- A data cube, is organised around a central theme, such as sales, allows data to be modelled and viewed in multiple dimensions

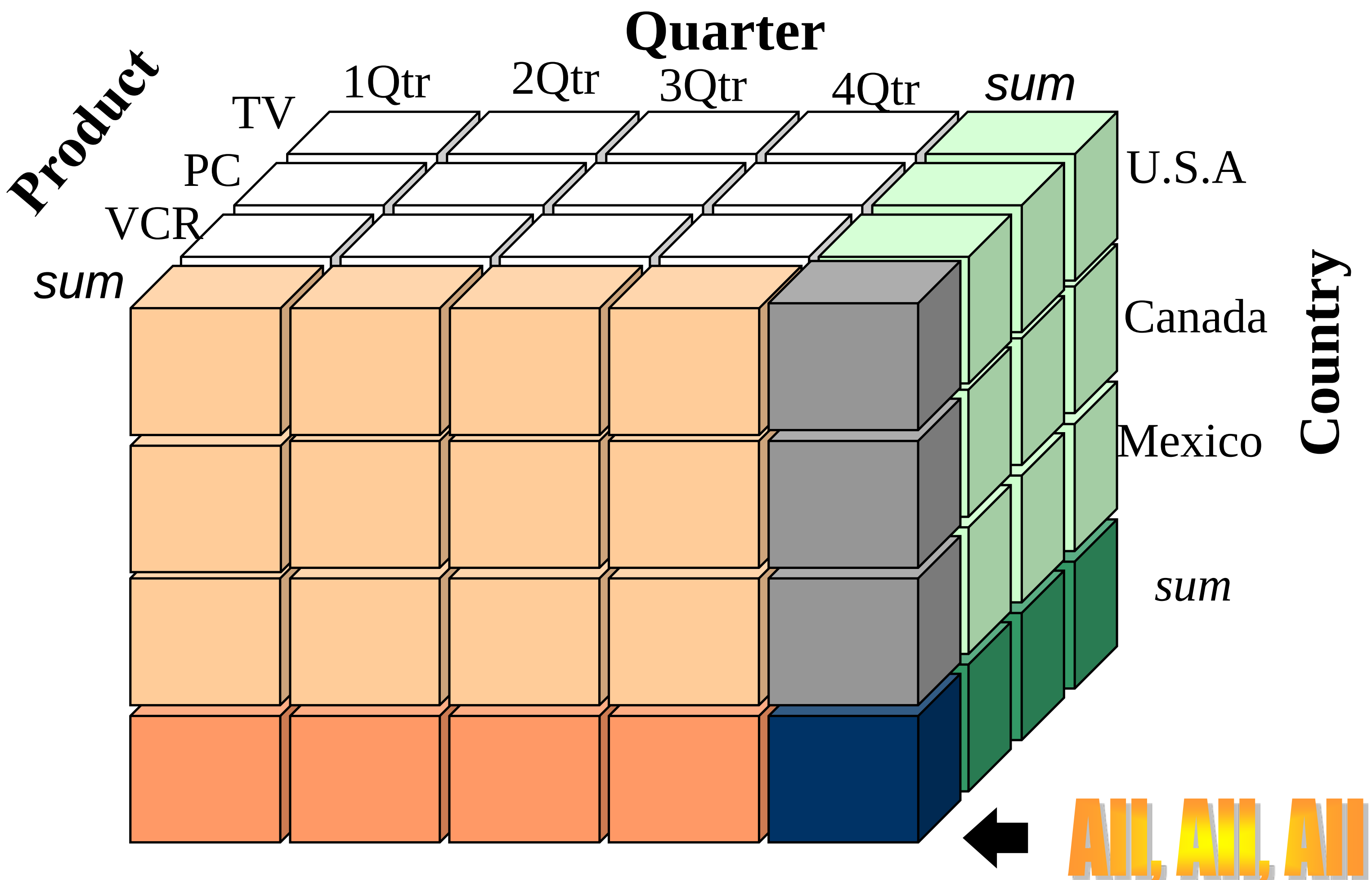
- Data cube is a metaphor for multi-dimentional data storage.

- The term hypercube is sometimes used, especially for data with more than three dimensions.

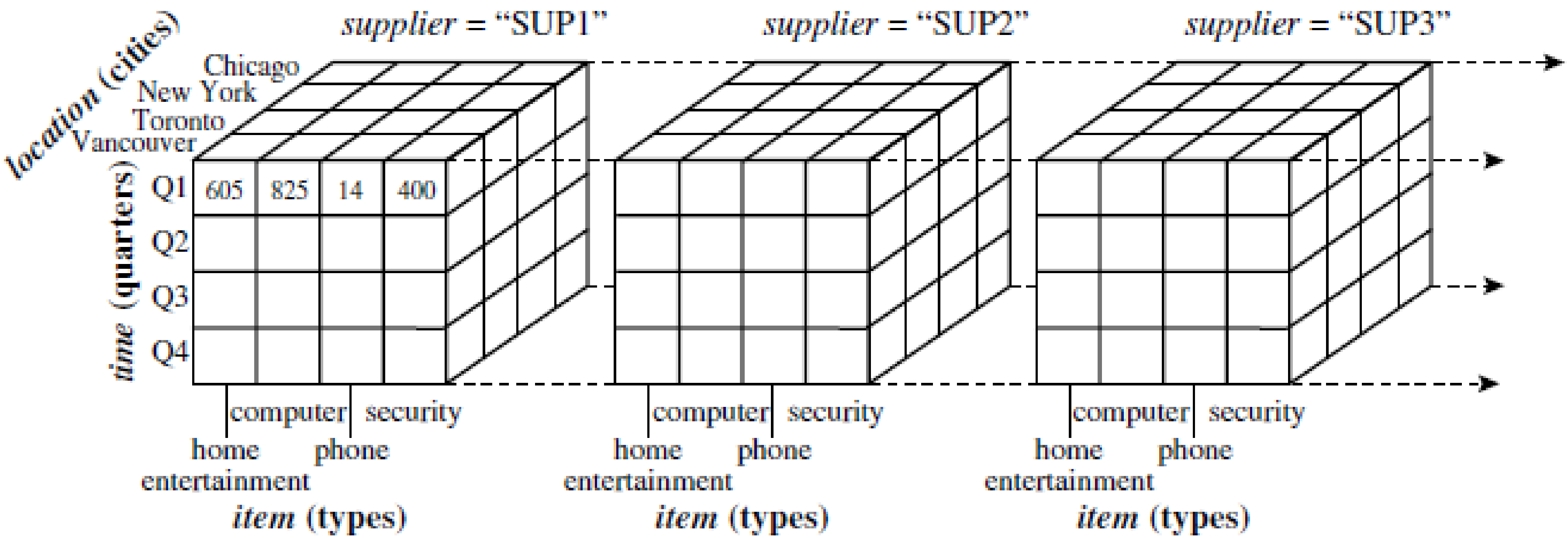
- A data cube is constructed from fact and dimension tables.

- A data cube is a lattice of cuboids.

Sample Data Cube

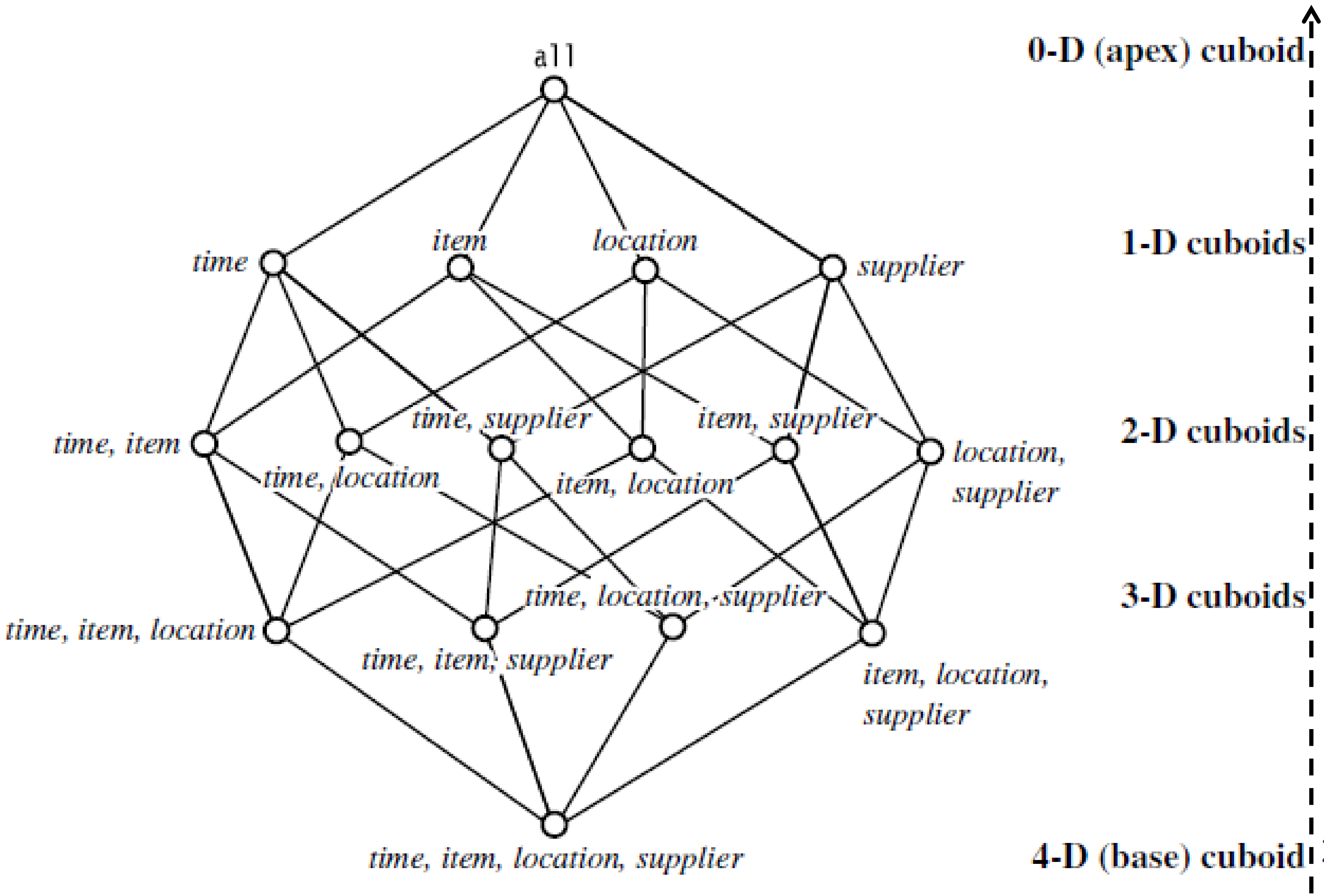


4-D Data Cube



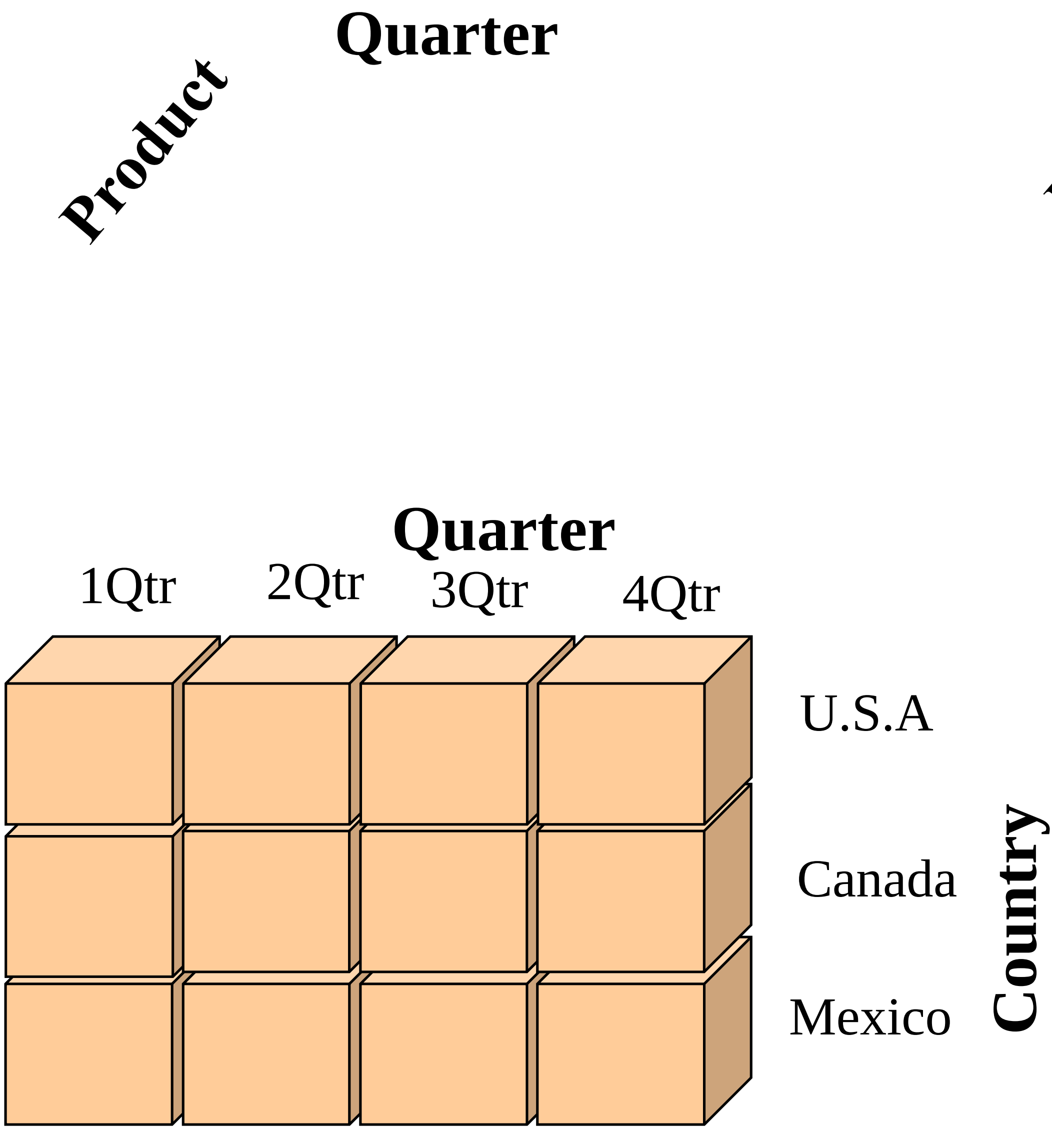
A 4-D data cube can be viewed as a set of 3-D data cubes.

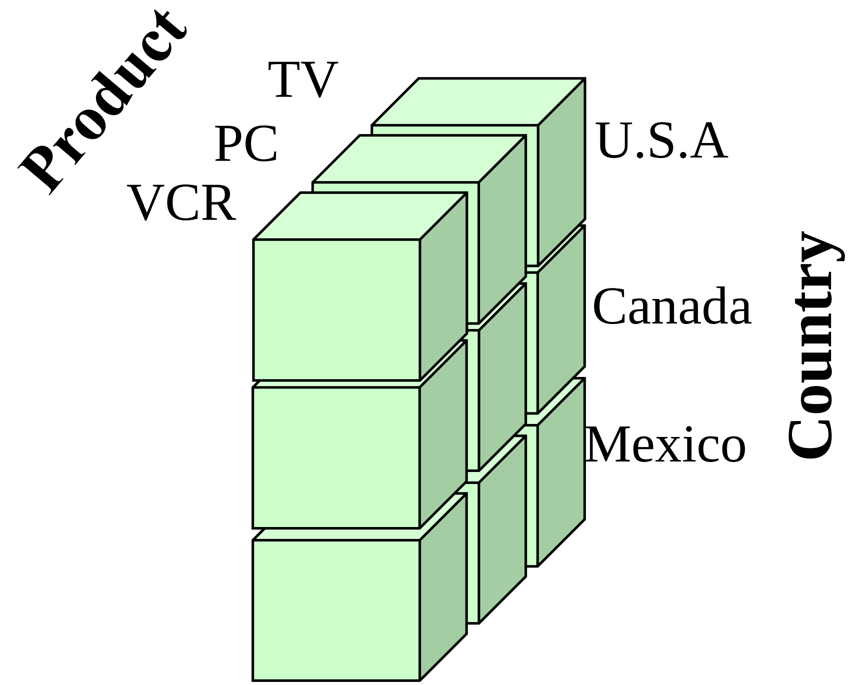
Cuboids



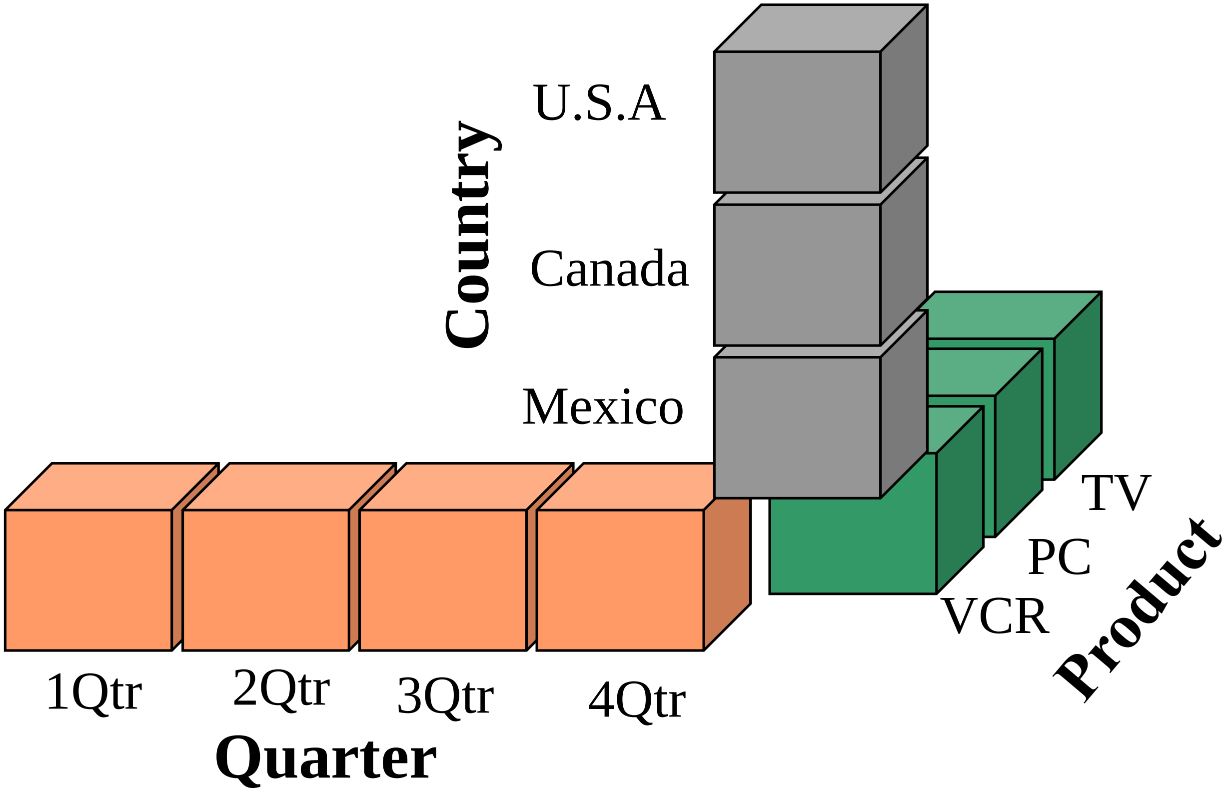
all
time
item
location
supplier
time, item
time, location
time, supplier
item, location
item, supplier
location, supplier
time, item, location
time, item, supplier
time, location, supplier
item, location, supplier
time, item, location, supplier
0-D (apex) cuboid
1-D cuboids
2-D cuboids
3-D cuboids
4-D (base) cuboid

Sample Data Cube: 2-D cubiods





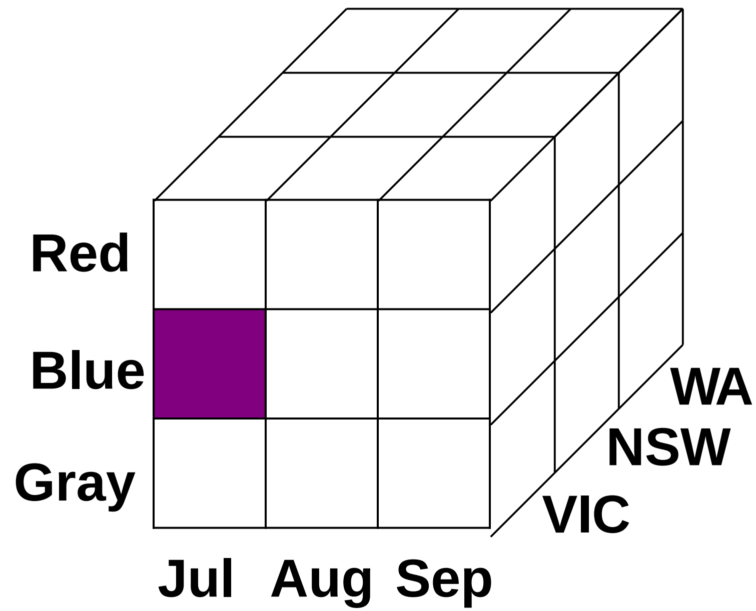
Sample Data Cube: 1-D cuboids



A Simple Data Cube

- Axes of the cube represent attributes of the data records
  - Generally discrete-valued/ categorical
  - e.g. color, month, state
  - Called dimensions
- Cells hold aggregated measurements
  - e.g. total $ sales, number of autos sold
  - Called facts
- Real data cubes have >> 3 dimensions

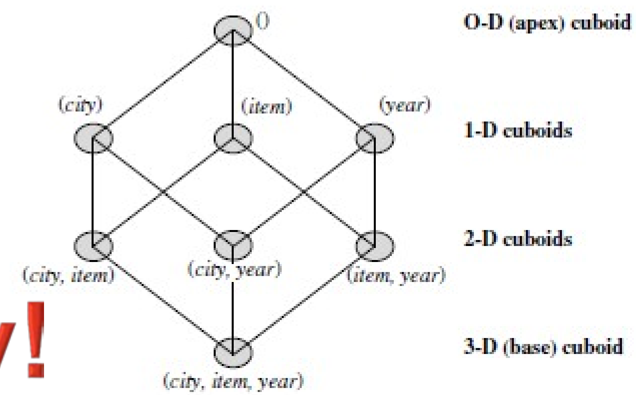
Auto Sales



What is the total number of cuboids?

- A data cube is a lattice of cuboids. Suppose that you want to create a data cube for AllElectronics sales that contains the following: *city*, *item*, *year*, and *sales* in dollars.
- Possible queries such as the following:
  - "Compute the sum of sales, grouping by city and item."
  - "Compute the sum of sales, grouping by city."
  - "Compute the sum of sales, grouping by item."

Curse of dimensionality!



Lecture Outline

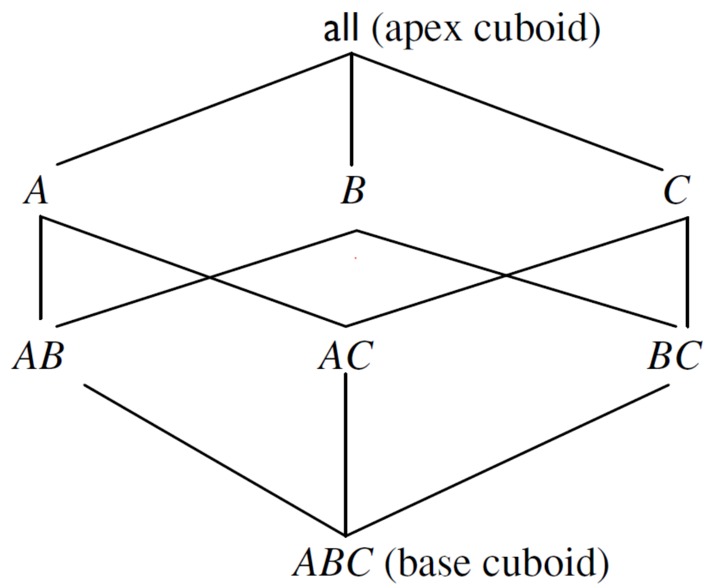
- Review of previous lectures
- Data Cube
  - Cuboids
  - Types of Cells
  - Types of Cubes
- Answering Queries with Data Cube
- Storage of data cube
  - MOLAP and ROLAP
  - Cube materialisation
  - Indexing data to support OLAP

Cells in a Data Cube

- Each cell of the cube holds a number that represents some measure of the business, such as sales, profits, expenses, budget and forecast.
- The measure is from the fact table.
- Measures are derived from the records in the fact table and dimensions are derived from the dimension tables.

Example

- Consider a data cube with the dimensions *month*, *city*, and *item type*, and the measure *sales*. What does the following notations mean?
  - (Jan, $*$, $*$, 2800) and ($*$, Perth, $*$, 1200)
  - (Jan, $*$, Phone, 150); and
  - (Jan, Perth, Phone, 45)



Ancestor and Descendant Cells

- An ancestor-descendant relationship may exist between cells.

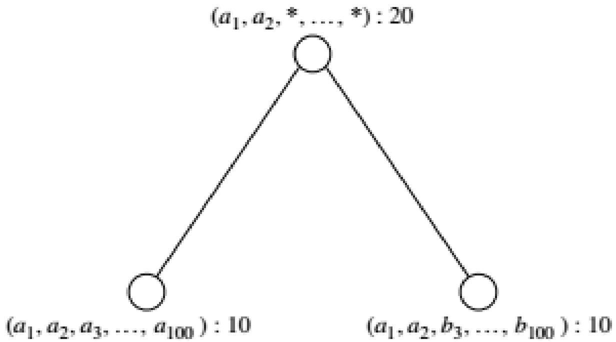
- In a n-dimensional data cube, given two cells
  - an $i - dimensional$ cell $a = (a_1, a_2, \ldots, a_i, measure_a)$
  - a $j - dimensional$ cell $b = (b_1, b_2, \ldots, b_j, measure_b)$
  - a is an ancestor of b, and b is a descendant of a if and only if
    - $i < j$ for $1 \le k \le n, a_k = b_k$ $wherever$ $a_k \ne *$.

- In particular, cell a is called a parent of cell b, and b is a child of a, if and only if $j = i + 1$.

- Example:
  - 1-D cell $a = (Jan, *, *, 2800)$ and 2-D cell $b = (Jan, *, Phone, 150)$ are ancestors of 3-D cell $c = (Jan, Perth, Phone, 45)$;
  - c is a descendant of both a and b;
  - b is a parent of c; and c is a child of b.

Closed cell and closed cube

- **Closed cell**
  - A cell, c, is a closed cell if there exists no cell, d, such that d is a specialisation (descendant) of cell c (i.e. where d is obtained by replacing $*$ in c with a non-$*$ value), and d has the same measure value as c.
  - No ancestor cell is created if its measure is equal to that of its descendent cell.
- **Closed Cube**
  - A closed cube is a data cube consisting of only closed cells.



Closed Cube in more detail

- Suppose that there are 2 base cells for a database of 100 dimensions, denoted as
  - $\{(a_1, a_2, a_3, \ldots, a_{100}, 10); (a_1, a_2, b_3, \ldots, b_{100}, 10)\}$, where each has a measure of 10.
  - The rest of the cells have a measure of 0.
- Is iceberg cube good enough in sparse cases like this? How many cuboids?
  - Apex cuboid + 1-D cuboids + 2-D cuboids + … + 99-D cuboids
    - $\binom{100}{0} + \binom{100}{1} + \binom{100}{2} + \cdots + \binom{100}{99} + 1 = 2^{100}$
  - Total number of distinct aggregate cells is $(2^{101} - 2) - 4$.
  - Ignore all the aggregate cells that can be obtained by replacing constants with $*$, only 3 cells really offer new information.
    - $\{(a_1, a_2, a_3, \ldots, a_{100}, 10), (a_1, a_2, b_3, \ldots, b_{100}, 10)\big), (a_1, a_2, *, \ldots, *, 20)\}$

Lecture Outline

- Review of previous lectures
- Data Cube
  - Cuboids
  - Types of Cells
  - Types of Cubes
- Answering Queries with Data Cube
- Storage of data cube
  - MOLAP and ROLAP
  - Cube materialisation
  - Indexing data to support OLAP

Types of cubes

- **Full cube**
  - The cube contains all cells and cuboids.
  - All possible combination of dimensions and values (*prohibitively expensive*).
    - 2^n - exponential to the number of dimensions (n)
    - more cuboids if we consider concept hierarchies for each dimension
    - the size of the cuboid depends on the cardinality of its dimensions
- **Closed cube**
  - No ancestor cell is created if its measure is equal to that of its descendent cell.
- **Iceberg cube (tip of iceberg)**
  - Only the cells in a cuboid whose measure value is above the minimum threshold.
  - Tradeoff between storage space and response time for OLAP.

```
compute cube sales_iceberg as
select month, city, customer_group, count(*)
from salesInfo
cube by month, city, customer_group
having count(*) >= min_sup
```

- **Shell cube**
  - Only cuboids with limited number of dimensions are precomputed.

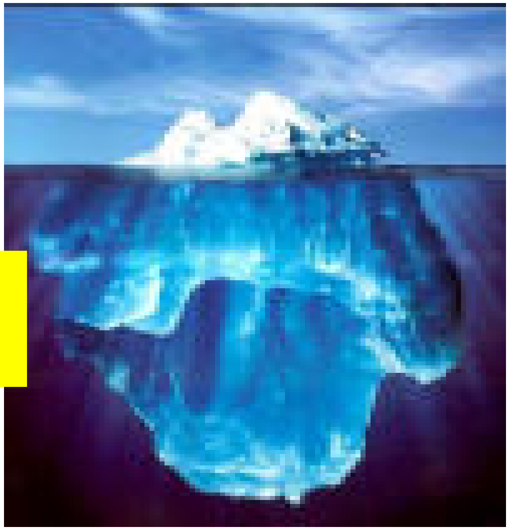
Materialisation: Full Cube vs. Iceberg Cube

THE UNIVERSITY OF WESTERN AUSTRALIA

- 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
- Only a small portion of cells may be "above the water'' in a sparse cube
- Ex.: Show only those cells whose count > 100

Full Data Cube with SubTotals

- Pre-computation of aggregates → fast answers to OLAP queries
- Ideally, pre-compute all 2^n types of subtotals
- Otherwise, perform aggregation as needed
- Coarser-grained totals can be computed from finer-grained totals
  - But not the other way around

Data Cube Terminologies – a review

THE UNIVERSITY OF WESTERN AUSTRALIA

A data cube is a lattice of cubiods.

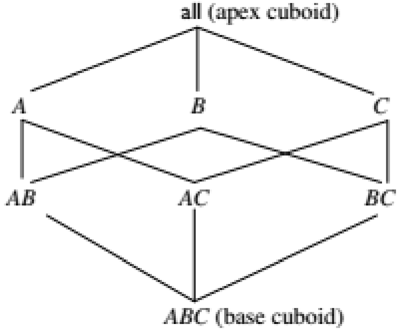
- Each cuboid represents a *group-by*.
- The *base cuboid* is the least generalised of all the cuboids.
- The *apex* cuboid is the most generalised of all the cuboids.

Operations

- Drill Down: move from the apex cuboid downward in the lattice.
- Roll Up: move from the base cuboid upward in the lattice.

Commonly used measures include:

- count(), sum(), min(), max()
- average()



Typical OLAP Operations

- **Roll up (drill-up): summarise 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*
- **Slice and dice:**
  - *project and select*
- **Pivot (rotate):**
  - *reorient the cube, visualisation, 3D to series of 2D planes.*
- **Other operations (aside)**
  - *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)*

Slice and Dice

- **Slice** is to pick a rectangular subset of a cube by choosing a single value for one of its dimensions, creating a new cube with one fewer dimension.

- **Dice:** The dice operation produces a subcube by allowing the analyst to pick specific values of multiple dimensions.

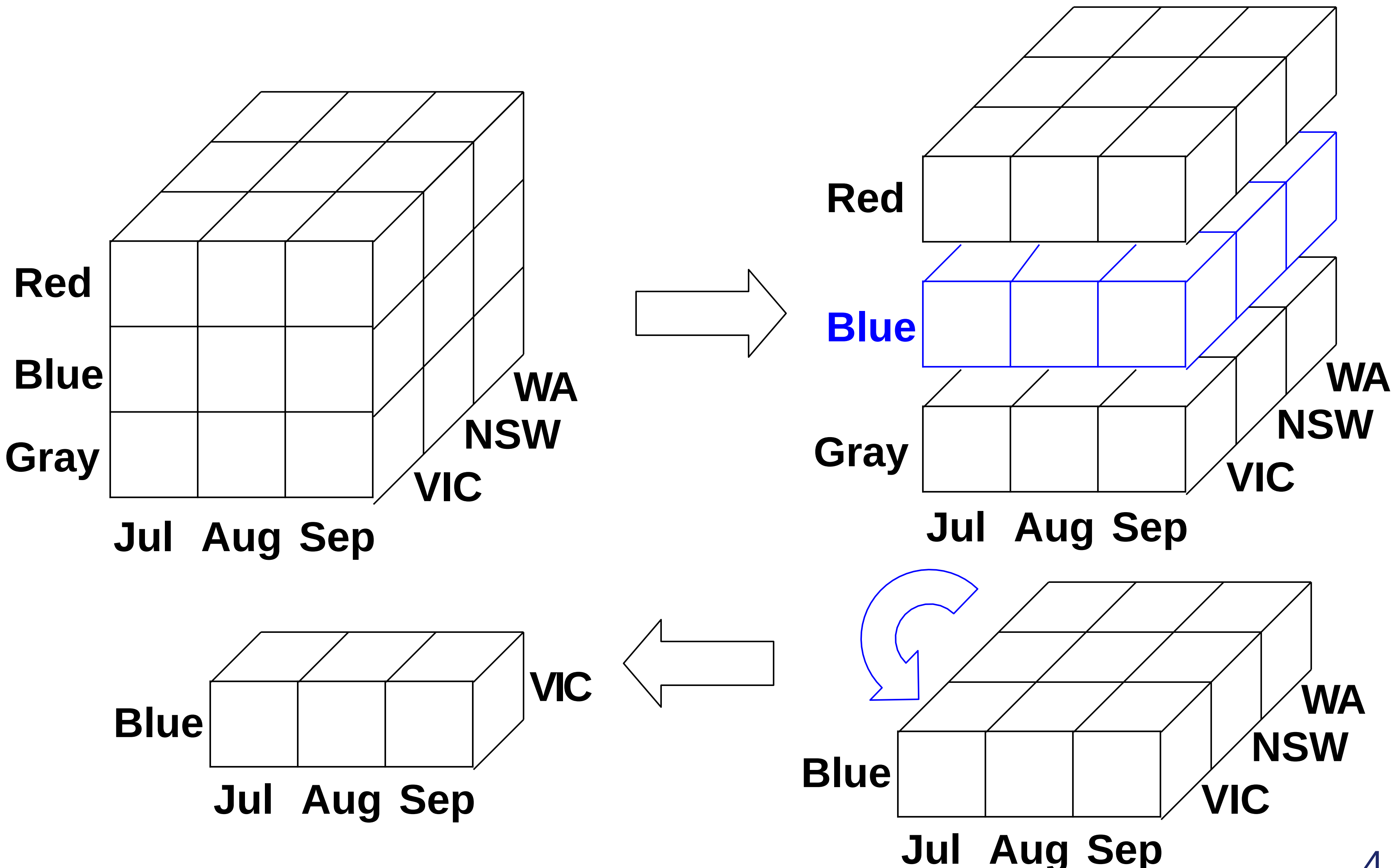
Data Cube Queries

- **Cross-tabulation**
  - "Cross-tab" for short
  - Report data grouped by 2 dimensions
  - Aggregate across other dimensions
  - Include subtotals
- **Operations on a cross-tab**
  - Roll up (further aggregation)
  - Drill down (less aggregation)

Autos Sold

	VIC	NSW	WA	Total
Jul	45	33	30	108
Aug	50	36	42	128
Sep	38	31	40	109
Total	133	100	112	345

Slicing and Dicing



Roll Up and Drill Down

Autos Sold

	VIC	NSW	WA	Total
Jul	45	33	30	108
Aug	50	36	42	128
Sep	38	31	40	109
Total	133	100	112	345

Roll up
by Month

Autos Sold

VIC	NSW	WA	Total
133	100	112	345

Drill down
by Color

Autos Sold

	VIC	NSW	WA	Total
Red	40	29	40	109
Blue	45	31	37	113
Gray	48	40	35	123
Total	133	100	112	345

Lecture Outline

- Review of previous lectures

- Data Cube
  - Cuboids
  - Types of Cells
  - Types of Cubes

- Answering Queries with Data Cube
  - SQL Server
  - Apache Hive

- Storage of data cube
  - MOLAP and ROLAP
  - Cube materialisation
  - Indexing data to support OLAP

Standard Operations to Answer Queries

Measurements

- Which fact(s) should be reported?

Filters

- What slice(s) of the cube should be used?

Grouping attributes

- How finely should the cube be diced?
- Each dimension is either:
  - (a) A grouping/categorical/discrete attribute
  - (b) Aggregated over ("Rolled up" into a single total)

n dimensions $\rightarrow 2^n$ sets of grouping attributes

Aggregation = projection to a lower-dimensional subspace

Efficient Processing of OLAP Queries

- Given four materialised cuboids, the query to be processed is on {brand, province or state}, with the selection constant "year = 2010."
  - cuboid 1: {year, item name, city}
  - cuboid 2: {year, brand, country}
  - cuboid 3: {year, brand, province or state}
  - cuboid 4: {item name, province or state}, where year = 2010

- Which one to choose?
  - Cuboids 1, 3, and 4 can be used to process the query because
    - they have the same set or a superset of the dimensions in the query,
    - the selection clause in the query can imply the selection in the cuboid, and
    - the abstraction levels for the item and location dimensions in these cuboids are at a finer level than brand and province or state, respectively.

Using Cube in Queries

- Queries with Data Cube in SQL Server

  SELECT month, state, SUM (amount)

  FROM SALES

  CUBE BY month, state

Creating Cross Tab with SQL

THE UNIVERSITY OF WESTERN AUSTRALIA

Grouping Attributes

Measurements

```
SELECT state, month, SUM(quantity)
FROM sales
GROUP BY state, month
WHERE color = 'Red'
```

Filters

	VIC	NSW	WA	Total
Jul	45	33	30	108
Aug	50	36	42	128
Sep	38	31	40	109
Total	133	100	112	345

Cross Tab Report

What about the totals

- SQL aggregation query with GROUP BY does not produce subtotals, totals
- Our cross-tab report is incomplete.

State	Month	SUM
VIC	Jul	45
VIC	Aug	50
VIC	Sep	38
NSW	Jul	33
NSW	Aug	36
NSW	Sep	31
WA	Jul	30
WA	Aug	42
WA	Sep	40

Autos Sold

	VIC	NSW	WA	Total
Jul	45	33	30	?
Aug	50	36	42	?
Sep	38	31	40	?
Total	?	?	?	?

One solution: a big UNION ALL

	VIC	NSW	WA	Total
Jul	45	33	30	?
Aug	50	36	42	?
Sep	38	31	40	?
Total	?	?	?	?

Original Query

```
SELECT state, month, SUM(quantity)
FROM sales
GROUP BY state, month
WHERE color = 'Red‘
UNION ALL
```

State Subtotals

```
SELECT state, "ALL", SUM(quantity)
FROM sales
GROUP BY state
WHERE color = 'Red'
UNION ALL
```

Month Subtotals

```
SELECT "ALL", month, SUM(quantity)
FROM sales
GROUP BY month
WHERE color = 'Red‘
UNION ALL
```

Overall Total

```
SELECT "ALL", "ALL", SUM(quantity)
FROM sales
WHERE color = 'Red'
```

“UNION ALL” on

> 2 attributes ??

A better solution

- "UNION ALL" solution gets cumbersome with more than 2 grouping attributes
- n grouping attributes → 2^n parts in the union
- OLAP extensions added to SQL 99 are more convenient
  - CUBE, ROLLUP

```
SELECT state, month, SUM(quantity)
FROM sales
GROUP BY CUBE(month, state)
WHERE color = 'Red'
```

Results of the Cube Query

THE UNIVERSITY OF WESTERN AUSTRALIA

	VIC	NSW	WA	Total
Jul	45	33	30	108
Aug	50	36	42	128
Sep	38	31	40	109
Total	133	100	112	345

Notice the use of NULL for totals

State	Month	SUM(quantity)
VIC	Jul	45
VIC	Aug	50
VIC	Sep	38
VIC	NULL	133
NSW	Jul	33
NSW	Aug	36
NSW	Sep	31
NSW	NULL	100
WA	Jul	30
WA	Aug	42
WA	Sep	40
WA	NULL	112
NULL	Jul	108
NULL	Aug	128
NULL	Sep	109
NULL	NULL	345

Subtotals at all levels

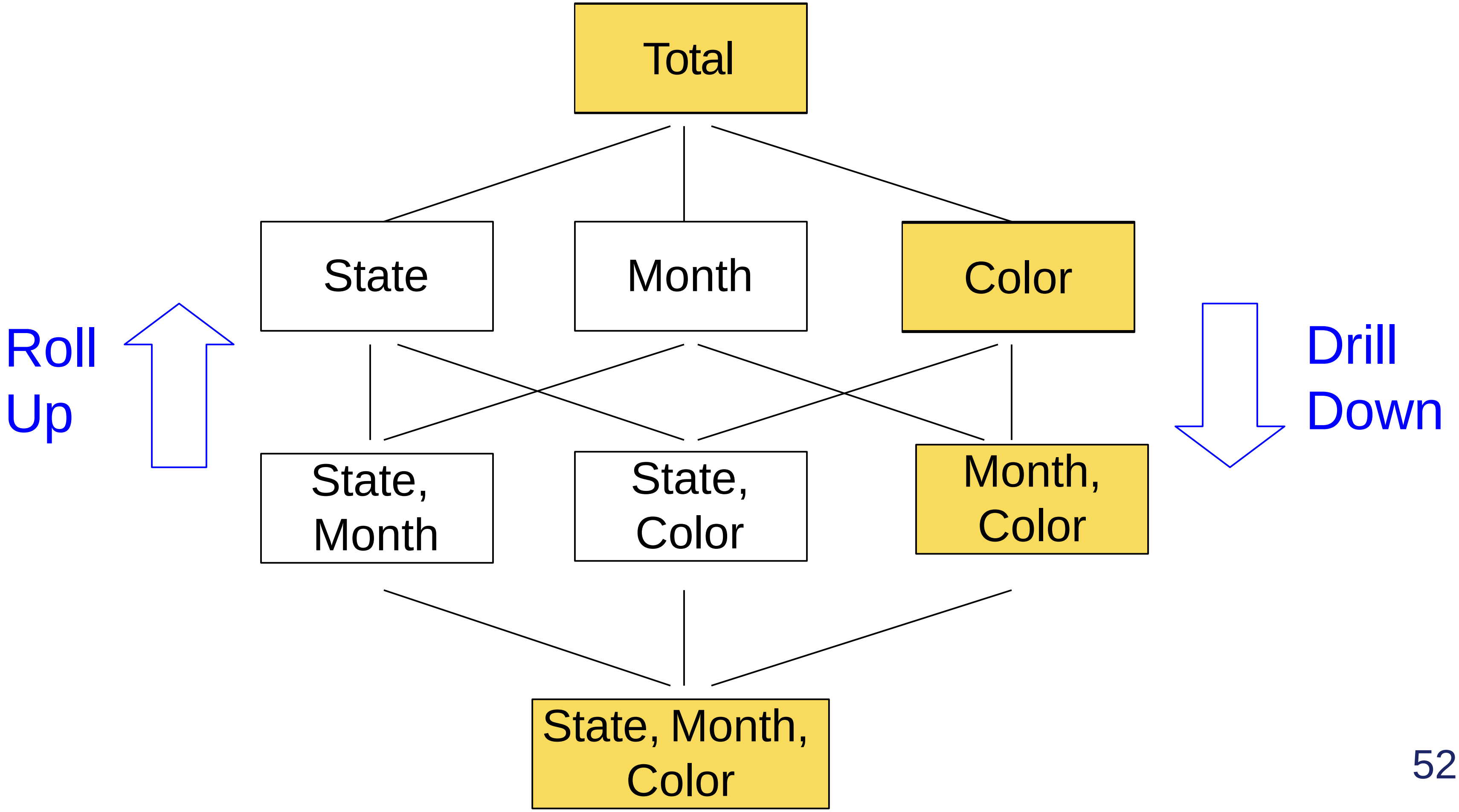
ROLLUP vs. CUBE

- CUBE computes entire lattice
- ROLLUP computes one path through lattice
  - Order of GROUP BY list matters
  - Groups by all prefixes of the GROUP BY list

- GROUP BY ROLLUP(A,B,C)
  - A,B,C
  - (A,B) subtotals
  - (A) subtotals
  - Total
- GROUP BY CUBE(A,B,C)
  - A,B,C
  - Subtotals for the following: (A,B), (A,C), (B,C), (A), (B), (C)
  - Total

Data Cube Lattice

```
SELECT color, month, state, SUM(quantity)
FROM sales
GROUP BY ROLLUP(color,month,state)
```



Lecture Outline

- Review of previous lectures

- Data Cube
  - ➢ Cuboids
  - ➢ Types of Cells
  - ➢ Types of Cubes

- Answering Queries with Data Cube
  - ➢ SQL Server
  - ➢ Apache Hive

- Storage of data cube
  - ➢ MOLAP and ROLAP
  - ➢ Cube materialisation
  - ➢ Indexing data to support OLAP

Data Cube in Apache Hive

THE UNIVERSITY OF WESTERN AUSTRALIA

- Hive is a data warehouse infrastructure tool to process structured data in Hadoop.

- It resides on top of Hadoop to summarise Big Data, and makes querying and analysing easy.

- Initially Hive was developed by Facebook

- later the Apache Software Foundation took it up and developed it further as an open source under the name Apache Hive.

https://hive.apache.org/

Features of Hive

THE UNIVERSITY OF WESTERN AUSTRALIA

- Hive is **not**
  - ✓ A relational database
  - ✓ A design for Online Transaction Processing (OLTP)
  - ✓ A language for real-time queries and row-level updates

- It stores schema in a database and processed data into Hadoop Distributed File Systems (HDFS).

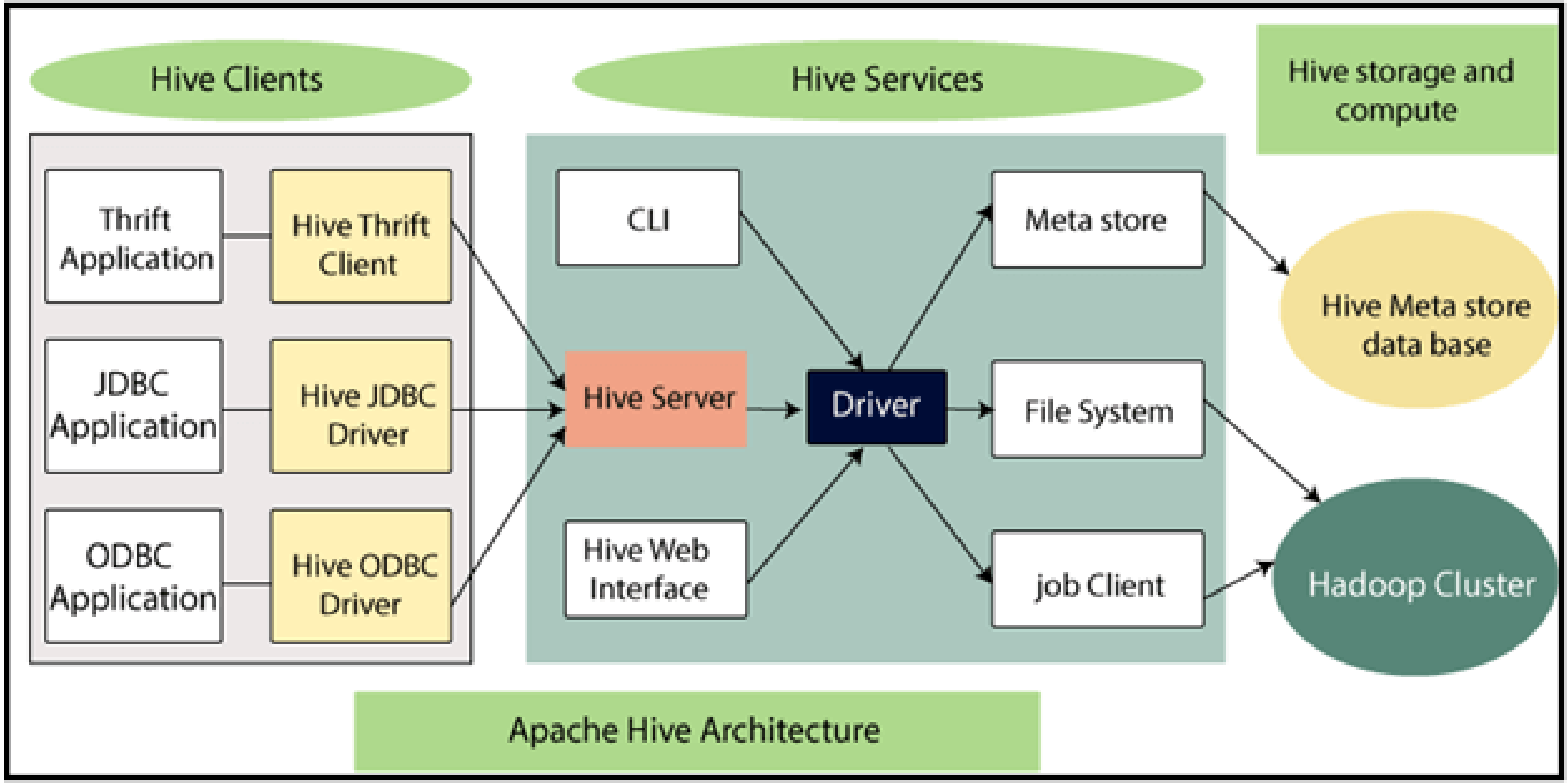
- It is designed for OLAP.

- It provides SQL-like language for querying called HiveQL.

- HiveQL is familiar, fast, scalable, and extensible.

THE UNIVERSITY OF WESTERN AUSTRALIA

Architecture of Hive



Data Model

A table `salesTable`:

Year	Quarter	Category	Subcategory	Sales
2019	2019-Q1	Stationary	Pen	231
2019	2019-Q1	Electronic	TV	198
2019	2019-Q1	Stationary	Pencil	438
2019	2019-Q1	Electronic	PC	981
2019	2019-Q2	Stationary	Pen	591
2019	2019-Q2	Electronic	TV	249
2019	2019-Q2	Stationary	Pencil	381
2019	2019-Q2	Electronic	PC	801
2020	2020-Q1	Stationary	Pen	53
2020	2020-Q1	Electronic	TV	982
2020	2020-Q1	Stationary	Pencil	364
2020	2020-Q1	Electronic	PC	65
2020	2020-Q2	Stationary	Pen	128
2020	2020-Q2	Electronic	TV	254
2020	2020-Q2	Stationary	Pencil	324
2020	2020-Q2	Electronic	PC	270

Goal: Statistics on sales per quarter or per category.

Build a Data Cube in Hive

Create Cube:

```
CREATE TABLE salestable_cube as SELECT year,
quarter, category, subcategory,
SUM(sales) AS totalsales,
GROUPING__ID
FROM salesTable
GROUP BY year,quarter,category,subcategory
WITH CUBE
ORDER BY GROUPING__ID;
```

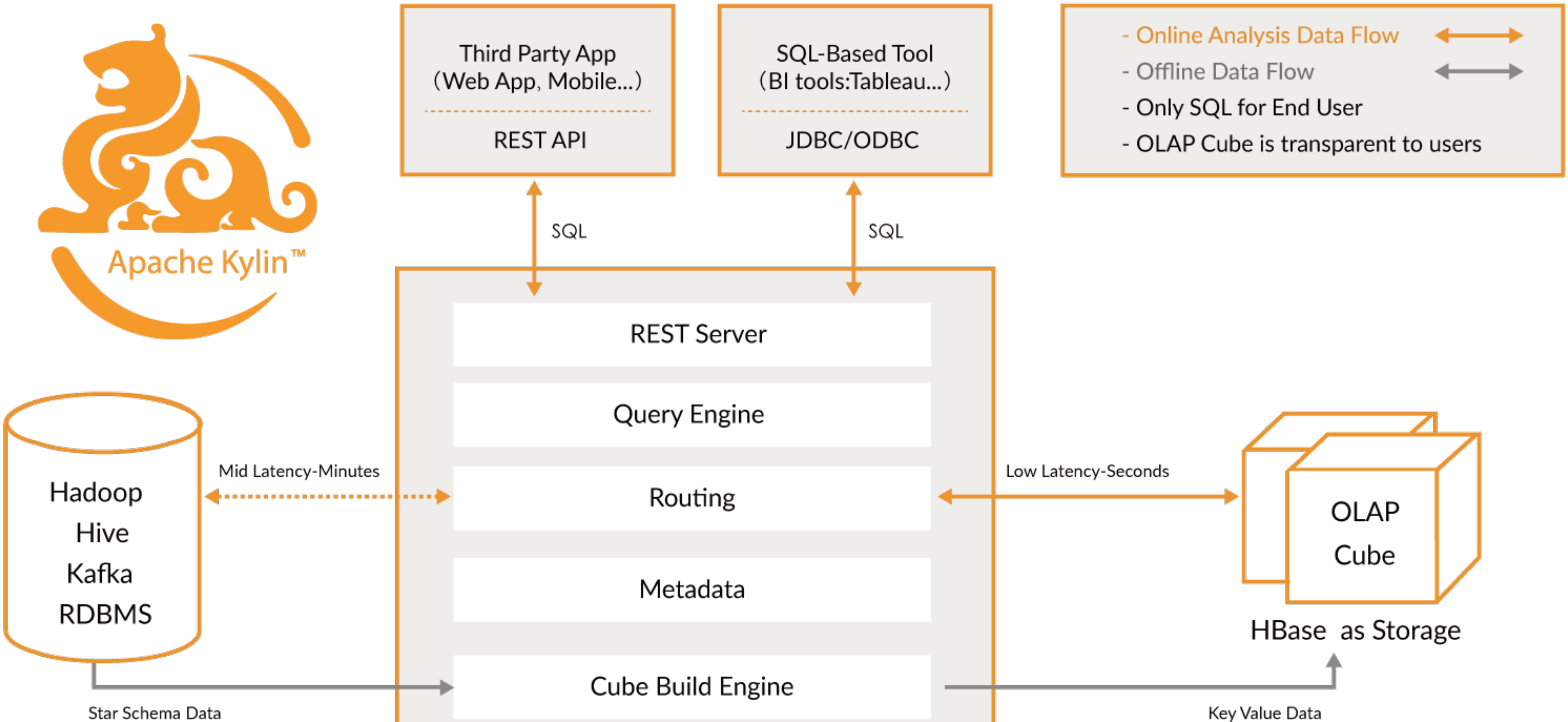
Grouping ID in Hive

- The grouping_id is used to select rows (i.e. cuboids) based on the dimensions of interest.

- Grouping ID is a bit vector of the dimensions in a cube and is stored as a base10 integer.

- It is a bitvector corresponding to whether the dimension is presented. For each dimension, a value of "1" is produced for a row in the result set if that column has been aggregated in that row, otherwise the value is "0".

Another Data Warehouse: Apache Kylin



Kylin is originally contributed from eBay Inc. in 2015.

Who Are Using Apache Kylin

THE UNIVERSITY OF WESTERN AUSTRALIA

http://kylin.apache.org/

Lecture Outline

THE UNIVERSITY OF WESTERN AUSTRALIA

- Review of Previous Lectures

- Data Cube
  - Cuboids
  - Types of Cells
  - Types of Cubes

- Answering Queries with Data Cube

- Storage of Data Cube
  - MOLAP and ROLAP
  - Cube Materialisation
  - Indexing Data to Support OLAP

MOLAP

- MOLAP = Multidimensional OLAP
- Store data cube as multidimensional array
- (Usually) pre-compute all aggregates
- Advantages:
  - Very efficient data access $\longrightarrow$ fast answers
- Disadvantages:
  - Doesn't scale to large numbers of dimensions
  - Requires special-purpose data store

Data Sparsity

THE UNIVERSITY OF WESTERN AUSTRALIA

- Imagine a data warehouse for Woolworths.
  - Suppose dimensions are: Customer, Product, Store, Day
- If there are 100,000 customers, 10,000 products, 1,000 stores, and 1,000 days…
  - …data cube has 1,000,000,000,000,000 (1000 Trillion) cells!
- Fortunately, most cells are empty.
  - A given store doesn't sell every product on every day.
  - A given customer has never visited most of the stores.
  - A given customer has never purchased most products.
- Multi-dimensional arrays are not an efficient way to store sparse data.

ROLAP

- ROLAP = Relational OLAP
- Store data cube in relational database
- Express queries in SQL

- Advantages:
  - Scales well to high dimensionality
  - Scales well to large data sets
  - Sparsity is not a problem
  - Uses well-known, mature technology
- Disadvantages:
  - Query performance is slower than MOLAP
  - Need to construct explicit indexes

Storage Modes in SQL Server

Microsoft SQL Server Analysis Services support

- both MOLAP and ROLAP
- Hybrid OLAP (HOLAP)

This link for more details of SQL Server OLAP support.

Lecture Outline

- Review of Previous Lectures

- Data Cube
  - Cuboids
  - Types of Cells
  - Types of Cubes

- Answering Queries with Data Cube

- Storage of Data Cube
  - MOLAP and ROLAP
  - Cube Materialisation
  - Indexing Data to Support OLAP

Data Cube and Cuboids

THE UNIVERSITY OF WESTERN AUSTRALIA

Data cube can be viewed as a lattice of cuboids

- The bottom-most cuboid is the base cuboid
- The top-most cuboid (apex) contains only one cell
- If no hierarchies in each dimension, the total # of cuboids for an n-dimensional data cube is 2^n.
- # of cuboids in an n-dimensional cube with L levels?

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

Materialisation of data cube

- Materialise every (cuboid) (full materialisation), none (no materialisation), or some (partial materialisation)
- Selection of which cuboids to materialise
  - Based on size, sharing, access frequency, etc.

Materialisation of Data Cube

- Materialise
  - <u>every</u> (cuboid) (full materialisation),
  - <u>none</u> (no materialisation), or
  - some (partial materialisation)

- Partial materialisation needs selection of which cuboids to materialise
  - ➢ Based on size, sharing, access frequency, etc.

Lecture Outline

THE UNIVERSITY OF WESTERN AUSTRALIA

- Review of Previous Lectures
- Data Cube
  - ➢ Cuboids
  - ➢ Types of Cells
  - ➢ Types of Cubes
- Answering Queries with Data Cube
- Storage of Data Cube
  - ➢ MOLAP and ROLAP
  - ➢ Cube Materialisation
  - ➢ Indexing Data to Support OLAP

Indexing OLAP Data: Bitmap Indexing

- **In the AllElectronics data warehouse,**
  - dimension item has four values (representing item types): “home entertainment (H),” “computer (C),” “phone (P),” and “security (S).”
  - Suppose that the cube is stored as a relation table with 100,000 rows. Because the domain of item consists of four values, the bitmap index table requires four bit vectors (or lists) for each record. We have a total 100,000 vectors.

Base table

RID	*item*	*city*
R1	H	V
R2	C	V
R3	P	V
R4	S	V
R5	H	T
R6	C	T
R7	P	T
R8	S	T

item bitmap index table

RID	H	C	P	S
R1	1	0	0	0
R2	0	1	0	0
R3	0	0	1	0
R4	0	0	0	1
R5	1	0	0	0
R6	0	1	0	0
R7	0	0	1	0
R8	0	0	0	1

city bitmap index table

RID	V	T
R1	1	0
R2	1	0
R3	1	0
R4	1	0
R5	0	1
R6	0	1
R7	0	1
R8	0	1

Note: H for “home entertainment,” C for “computer,” P for “phone,” S for “security,” V for “Vancouver,” T for “Toronto.”

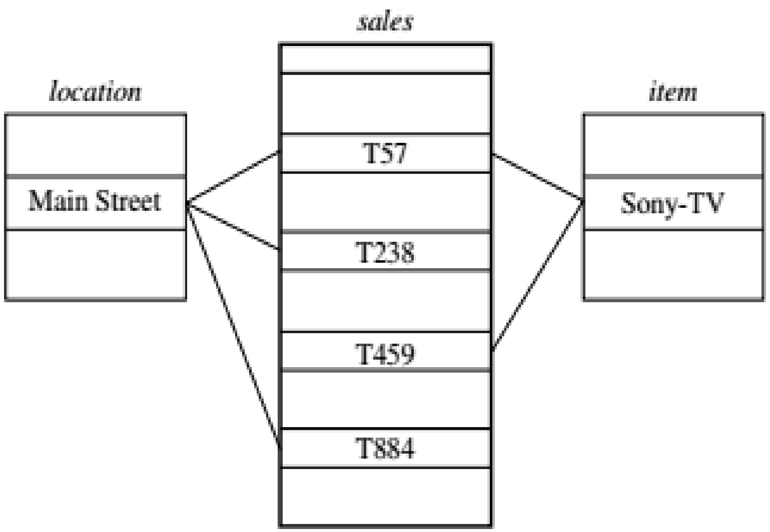
Another Example

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

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

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

Indexing on OLAP Data - Join indexing



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
. . .	. . .	. . .

References

- Readings
  - Han et al. Chapter 5
  - What is cross-tabulation?
  - How to implement one-to-one, one-to-many and many-to-many relationships of an ER model?