INFS1200/7900
Introduction to Information Systems

Data Warehousing & OLAP

Hassan Khosravi

Learning Objectives

Description	Tag
Compare and contrast OLAP and OLTP processing (e.g., focus, clients, amount of data, abstraction levels, concurrency, and accuracy).	Data warehousing
Given a multidimensional cube, write regular SQL queries that perform roll-up, drill-down, slicing, dicing, and pivoting operations on the cube.	
Use the SQL:1999 standards for aggregation (e.g., GROUP BY CUBE) to efficiently generate the results for multiple views.	
Explain the differences between a star schema design and a snowflake design for a data warehouse, including potential tradeoffs in performance.	

Motivation

Data Warehousing

On-Line Analytical Processing

ROLLUP and CUBE Operators

Star Schema vs Snowflake Schema

What We Have Focused on So Far

- OLTP (On-Line Transaction Processing)
  - class of information systems that facilitate and manage transaction-oriented applications, typically for data entry and retrieval transaction processing.
  - the system responds immediately to user requests.
  - high throughput and insert- or update-intensive database management. These applications are used concurrently by hundreds of users.
- The key goals of OLTP applications are availability, speed, concurrency and recoverability.

source: Wikipedia

On-Line Transaction Processing

❖ OLTP Systems are used to "run" a business.

- Examples: Answering queries from a Web interface, sales at cash registers, selling airline tickets, transactions at an ATM.

	OLTP
Typical User	Basically Everyone (Many Concurrent Users)
Type of Data	Current, Operational, Frequent Updates
Type of Query	Short, Often Predictable
# of Queries	Many concurrent queries
Access	Many reads, writes and updates
DB Design	Application oriented not subject oriented
Schema	E-R model, RDBMS
Normal Form	Often 3NF or BCNF
Typical Size	MB to GB
Protection	Concurrency Control, Crash Recovery
Function	Day to day operations

Can We Do More?

- Increasingly, organizations are analyzing current and historical data to identify useful patterns and support business strategies.
  - "Decision Support", "Business Intelligence"

- The emphasis is on complex, interactive, exploratory analysis of very large datasets created by integrating data from across all parts of an enterprise.

A Producer Wants to Know …

Who are our lowest/highest margin customers ?

want to save money

What is the most effective distribution channel?

Who are my customers, and what products are they buying?

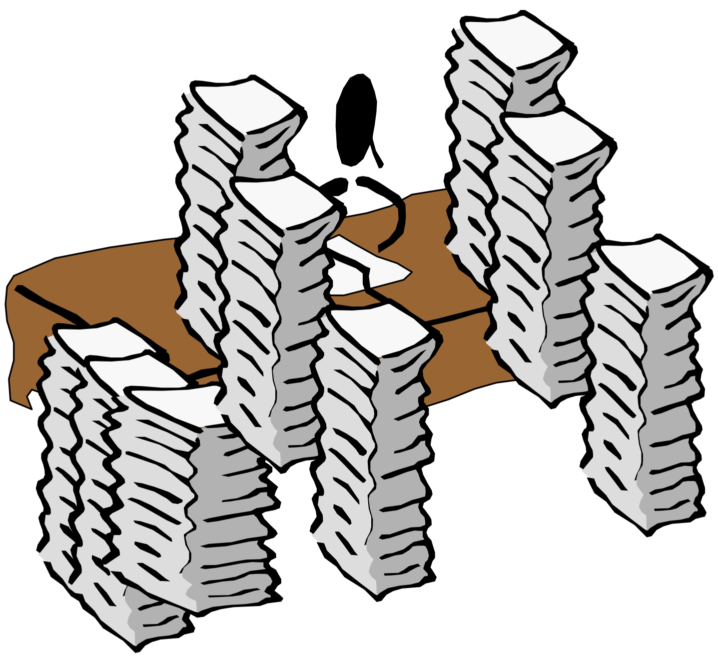
What product promo- -tions have the biggest impact on revenue?

Which customers are most likely to go to the competition ?

What impact will new products/services have on revenue and margins?

assuming have a change

Data, Data, Everywhere, yet ...



- I can’t find the data I need
  - Data is scattered over the network
  - Many versions, many sources, subtle differences, incompatible formats, missing values
- I can’t get the data I need
  - Need an expert to get the data from various sources
- I can’t understand the data I found
  - Poorly documented
- I can’t use the data I found
  - Results are unexpected
  - Not sure what I’m looking for
  - Data needs to be transformed

Motivation

Data Warehousing

On-Line Analytical Processing

ROLLUP and CUBE Operators

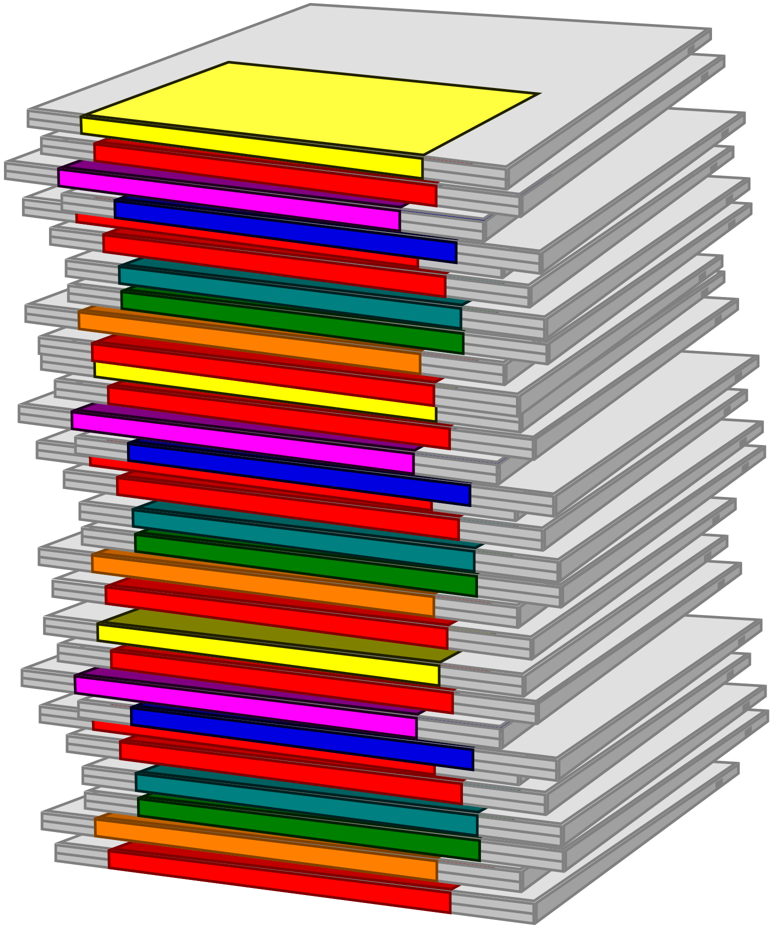
Star Schema vs Snowflake Schema

What is 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

Recognized by many as the father of the data warehouse



Data Warehouse—Subject-Oriented

- **Subject-Oriented:** Data that gives information about a particular subject area such as customer, product, and sales instead of about a company's ongoing operations.
  1. Focusing on the modeling and analysis of data for decision makers, not on daily operations or transaction processing
  2. Provides a simple and concise view around particular subject issues by excluding data that are not useful in the decision support process

Application-Oriented

Membership levels

ID	Type	Fee
A	Gold	$100
B	Basic	$50

Visit Level

ID	Type	Fee
YP	Pool	$15
NP	No pool	$10

Members

ID	Name	Level	StartDate
111	Joe	A	01/01/2008
222	Sue	B	01/01/2008
333	Pat	A	01/01/2008

Non-member Visits

ID	VID	VisitDate
1	YP	01/01/2008
2	YP	01/01/2008
3	NP	01/01/2008

Subject-Oriented

Revenue

R-ID	Date	By	Amount
		….	
7235	01/01/2008	Non-Member	$15
7236	01/01/2008	Member	$100
7237	01/01/2008	Member	$50
7238	01/01/2008	Member	$100
7239	01/01/2008	Non-Member	$10
7240	01/01/2008	Non-Member	$15
		…	

Data Warehouse—Integrated

- Constructed by integrating multiple, heterogeneous data sources.
  - relational databases, XML, flat files, on-line transaction records
- Data cleaning and data integration techniques are applied.
  - Ensure consistency in naming conventions, encoding structures, attribute measures, etc. among different data sources.
  - e.g., Hotel price depends on: currency, various room taxes, whether breakfast or Internet is included, etc.

DW—Time Variant

- Time-Variant: All data in the data warehouse are associated with a particular time period.
- The time horizon for a DW is significantly longer than for operational systems.
  - Operational DB: all data is current, and subject to change
  - DW: contains lots of historical data that may never change, but may have utility to the business when determining trends, outliers, profitability; effect of business decisions or changes to policy; pre-compute aggregations; record monthly balances or inventory; etc.
  - DW data is tagged with date and time, explicitly or implicitly

DW—Non-volatile

- **Non-volatile:** Data is stable in a data warehouse. More data is added, but data is not removed. This enables management to gain a consistent picture of the business.
- Real-time updates of operational data typically does not occur in the DW environment, but can be done in bulk later (e.g., overnight, weekly, monthly).
  - DW does not require transaction processing, recovery, and concurrency control mechanisms.
  - DW focuses on two major operations:
    - loading of data and accessing of data

Operational DBMS vs. Data Warehouse

- Operational DBMS
  - Day-to-day operations: purchasing, inventory, banking, payroll, manufacturing, registration, accounting, etc.
  - Used to run a business

- Data Warehouse
  - Data analysis and decision making
  - Integrated data spanning long time periods, often augmented with summary information
  - helps to "optimize" the business

Why a Separate Data Warehouse?

- High performance for both systems
  – DBMS— tuned for OLTP: access methods, indexing, concurrency control, recovery
  – Warehouse—needs to be tuned for complex queries, multidimensional views, consolidation

- Different types of queries
  – Extensive use of statistical functions which are poorly supported in DBMS
  – Running queries that involve conditions over time or aggregations over a time period, which are poorly supported in a DBMS

sql of data warehouse is normally more complex, so process much slower

Motivation

Data Warehousing

On-Line Analytical Processing

ROLLUP and CUBE Operators

Star Schema vs Snowflake Schema

On-Line Analytical Processing

- Technology used to perform complex analysis of the data in a data warehouse.
  - OLAP is a category of software technology that enables analysts, managers, and executives to gain insight into data through fast, consistent, interactive access to a wide variety of possible views of information.
  - The data has been transformed from raw data to reflect the dimensionality of the enterprise as understood by the user.
- OLAP queries are, typically:
  - Full of grouping and aggregation
  - Few, but complex queries -- may run for hours

transaction analytical

OLTP vs. OLAP

	OLTP	OLAP
Typical User	Basically Everyone (Many Concurrent Users) general users	Managers, Decision Support Staff (Few)
Type of Data	Current, Operational, Frequent Updates	Historical, Mostly read-only
Type of Query	Short, Often Predictable	Long, Complex
# query	Many concurrent queries	Few queries
Access	Many reads, writes and updates	Mostly reads
DB design	Application oriented	Subject oriented
Schema	E-R model, RDBMS	Star or snowflake schema
Normal Form	Often 3NF	Unnormalized
Typical Size	MB to GB	GB to TB
Protection	Concurrency Control, Crash Recovery	Not really needed (it's backup of data that use to analyse)
Function	Day to day operation	Decision support

Actionable Business Intelligence

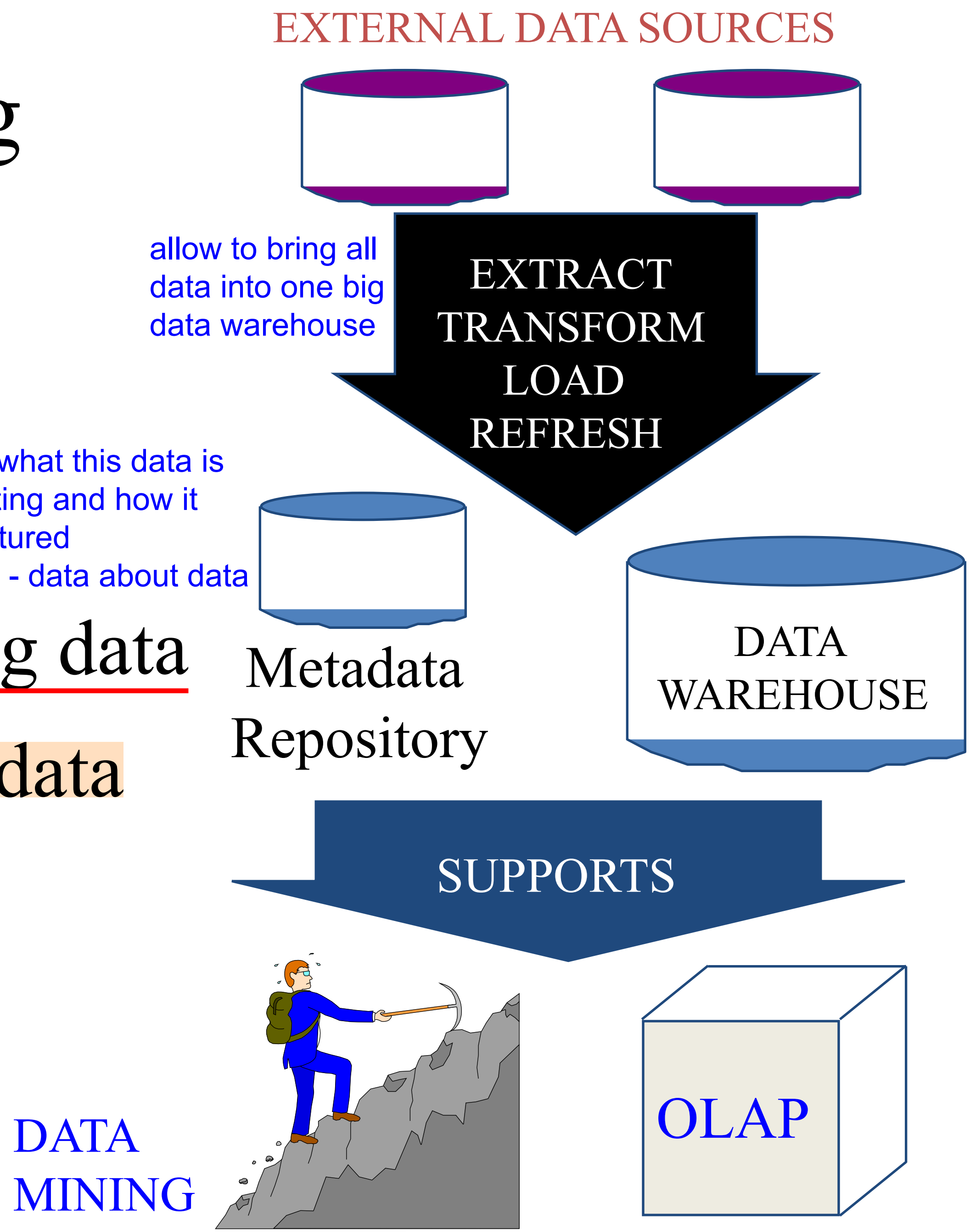
- In essence, the goal of business intelligence is to make strategic business decisions that improve sales, profits, response times, customer satisfaction, customer relationships, etc.

Data Warehousing

- The process of constructing and using data warehouses is called data warehousing.

allow to bring all data into one big data warehouse

describe what this data is representing and how it been captured
Metadata - data about data



OLAP Queries

- OLAP queries are full of groupings and aggregations.
- The natural way to think about such queries is in terms of a multidimensional model, which is an extension of the table model in regular relational databases.
- This model focuses on:
  - a set of numerical measures: quantities that are important for business analysis, like sales, etc.
  - a set of dimensions: entities on which the measures depend on, like location, date, etc.

Multidimensional Data Model

- The main relation, which relates dimensions to a measure via foreign keys, is called the fact table.
  - The fact table has FKs to the dimension tables.
  - These mappings are essential.

- Each dimension can have additional attributes and an associated dimension table. Attributes can be numeric, categorical, temporal, counts, sums

- Fact tables are much larger than dimensional tables.

- There can be multiple fact tables.

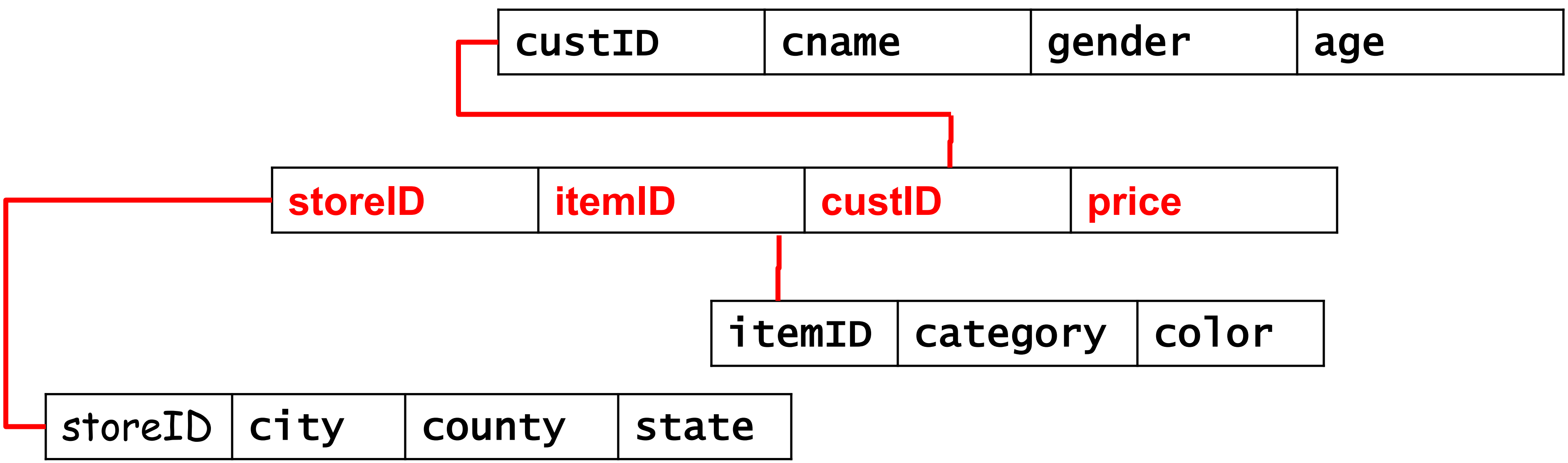
Design Issues

- The schema that is very common in OLAP applications, is called a star schema:
  - one table for the fact, and
  - one table per dimension
- The fact table is in BCNF.
- The dimension tables are not normalized. They are small; updates/inserts/deletes are relatively less frequent. So, redundancy is less important than good query performance.

Running Example

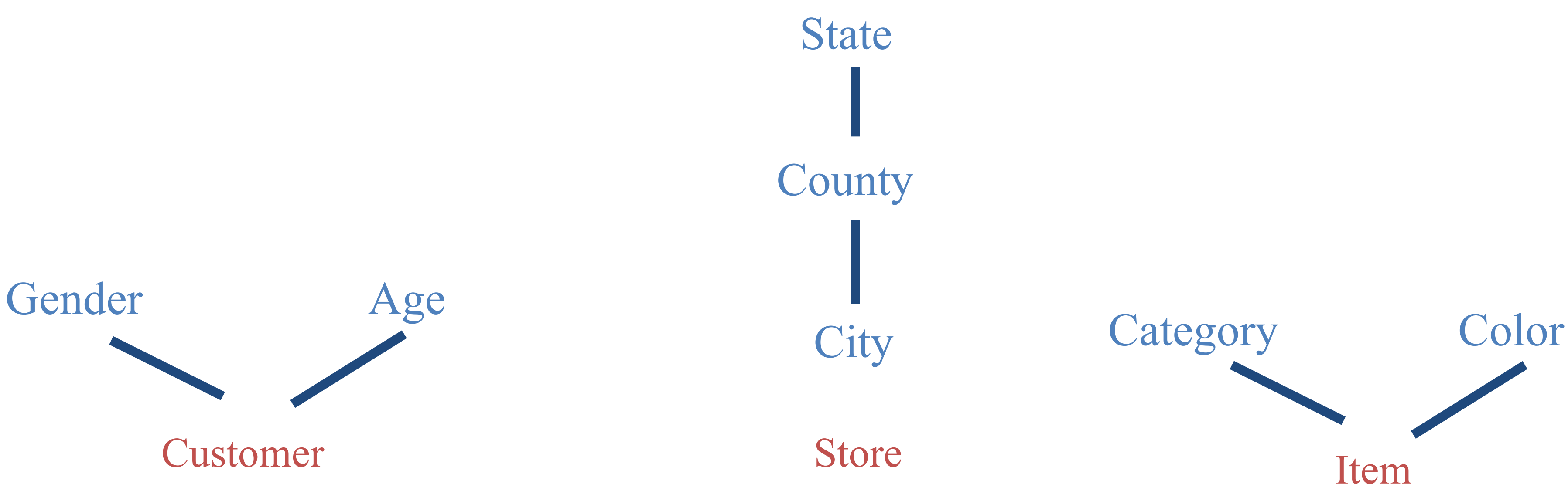
- Star Schema – fact table references dimension tables
  - Join → Filter → Group → Aggregate

```
Sales(storeID, itemID, custID, price)
Store(storeID, city, county, state)
Item(itemID, category, color)
Customer(custID, cname, gender, age)
```



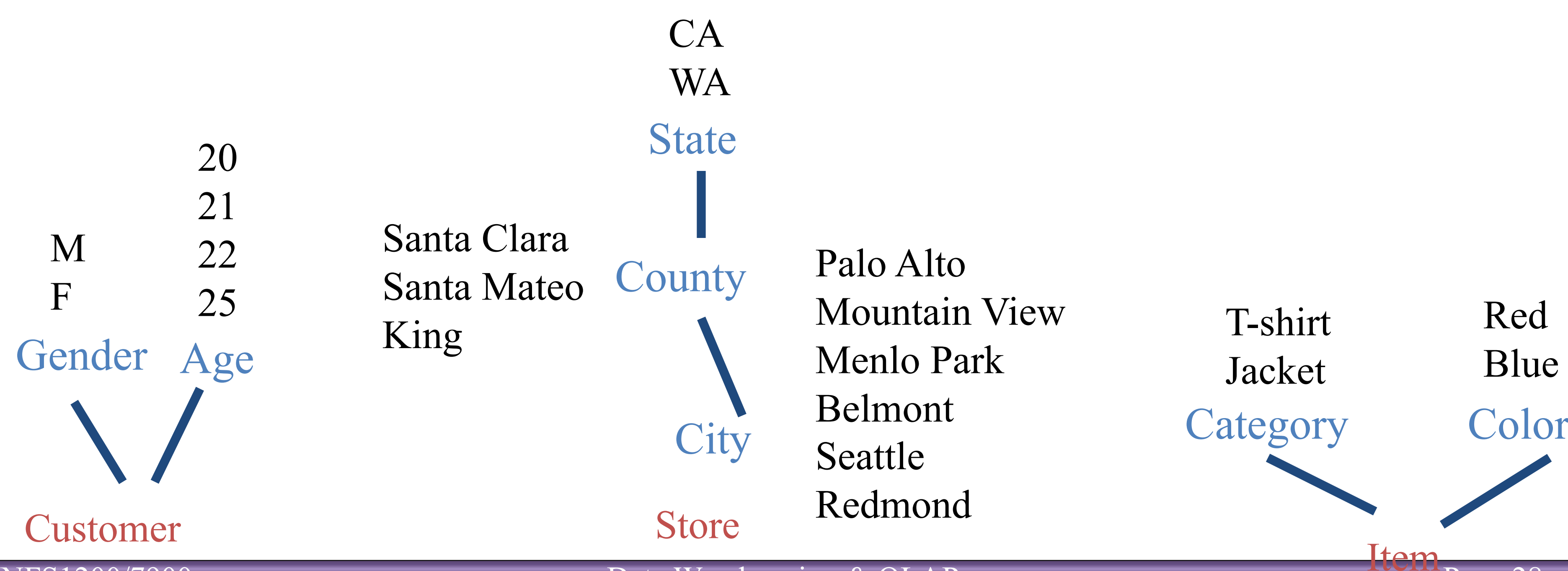
Dimension Hierarchies

- For each dimension, the set of values can be organized in a hierarchy:



Running Example (cont.)

```
Sales(storeID, itemID, custID, price)
Store(storeID, city, county, state)
Item(itemID, category, color)
Customer(custID, cname, gender, age)
```



Full Star Join

- An example of how to find the full star join (or complete star join) among 4 tables (i.e., fact table + all 3 of its dimensions) in a Star Schema:
  - Join on the foreign keys

```
SELECT *
FROM    Sales F, Store S, Item I, Customer C
WHERE F.storeID = S.storeID and
           F.itemID = I.itemID and
           F.custID = C.custID;
```

- If we join fewer than all dimensions, then we have a star join.
- In general, OLAP queries can be answered by computing some or all of the star join, then by filtering, and then by aggregating.

Full Star Join Summarized

- Find total sales by store, item, and customer.

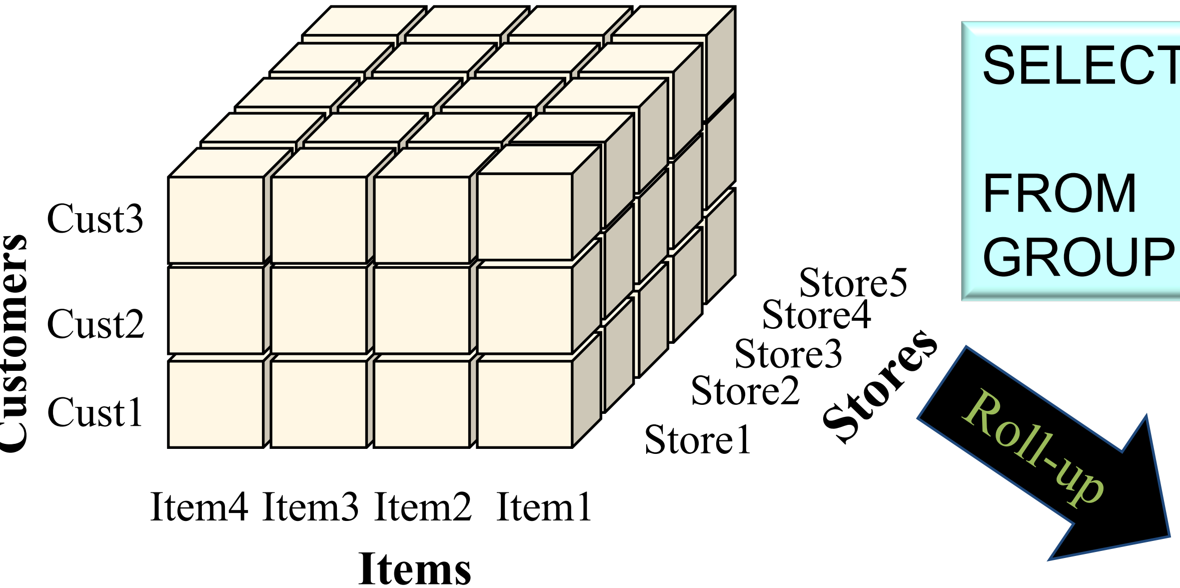
OLAP Queries – Roll-up

- Roll-up allows you to summarize data by:
  - changing the level of granularity of a particular dimension
  - dimension reduction

provide high level data (summary)

Roll-up Example 1 (Hierarchy)

- Use Roll-up on total sales by store, item, and customer to find total sales by item and customer for each county.



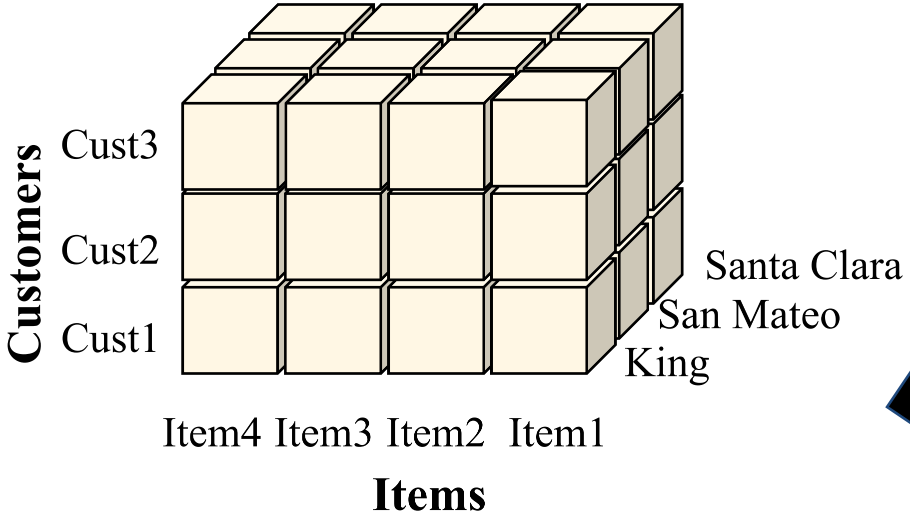
on stores level

```
SELECT storeID, itemID, custID,
           SUM(price)
FROM    Sales F
GROUP BY storeID, itemID, custID;
```

```
SELECT county, itemID, custID, SUM(price)
FROM Sales F, Store S
WHERE F.storeID = S.storeID
GROUP BY county, itemID, custID
```

Roll-up Example 2 (Hierarchy)

- Use Roll-up on total sales by item, customer, and county to find total sales by item, gender and county.

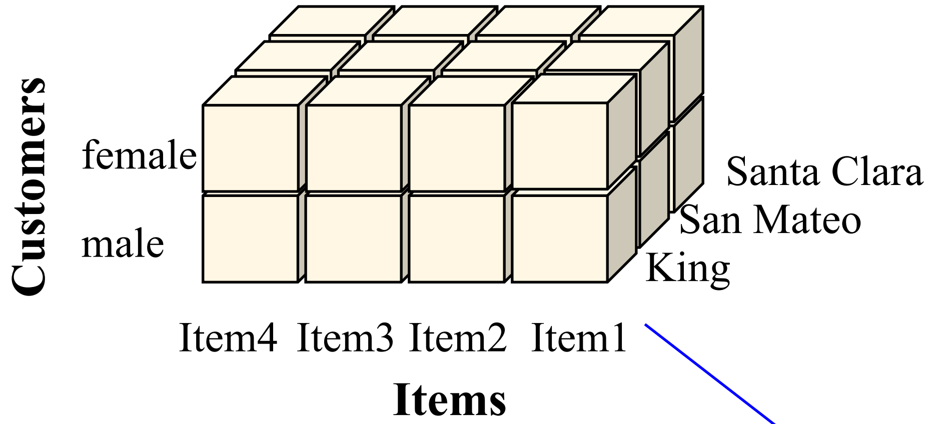



```
SELECT county, itemID, custID,
           SUM(price)
FROM    Sales F, Store S
WHERE F.storeID = S.storeID
GROUP BY county, itemID, custID;
```


```
SELECT itemID, gender, county, SUM(price)
FROM Sales F, Store S, Customer C
WHERE F.storeID = S.storeID AND C.custID = F.custID
GROUP BY itemID, gender, county
```

Roll-up Example 3 (Dimension)

- Use Roll-up on total sales by item, gender and county to find total sales by item for each county.




```
SELECT county, itemID, gender,
          SUM(price)
FROM    Sales F, Store S, Customer C
WHERE F.storeID = S.storeID AND
          F.custID = C.custID
GROUP BY county, itemID, gender;
```

Customers do not matter --> merge the table into 2 dimension

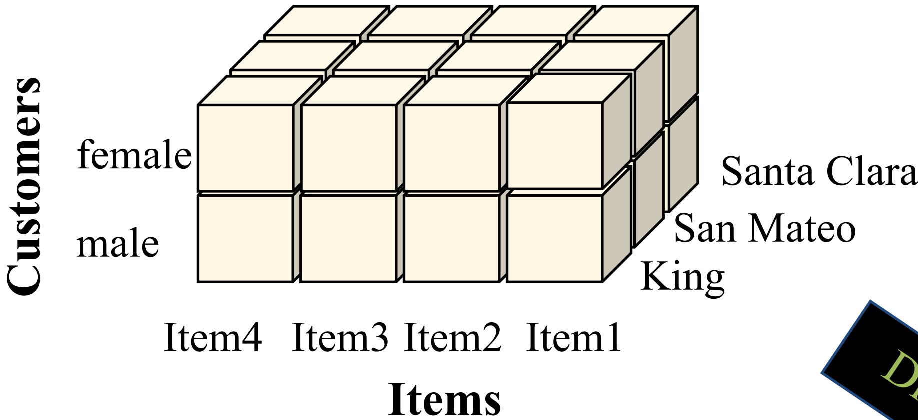
```
SELECT county, itemID, SUM(price)
FROM Store S, Sales F
WHERE F.storeID = S.storeID
GROUP BY county, itemID
```

OLAP Queries – Drill-down

- Drill-down: reverse of roll-up
  - From higher level summary to lower level summary (i.e., we want more detailed data) higher level --> lower level
  - Introducing new dimensions

Drill-down Example 1 (Hierarchy)

- Use Drill-down on total sales by item and gender for each county to find total sales by item and gender for each city.



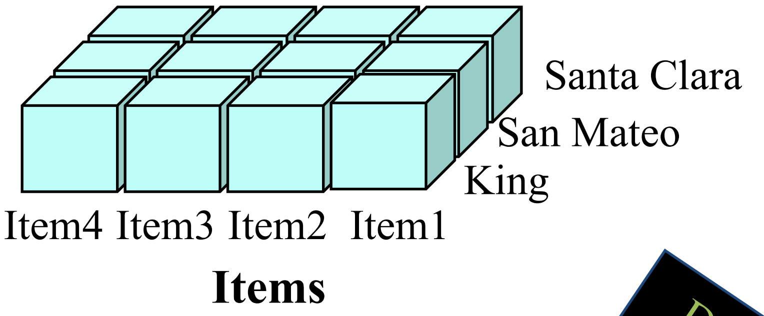

```
SELECT county, itemID, gender,
          SUM(price)
FROM    Sales F, Store S, Customer C
WHERE F.storeID = S.storeID AND
          F.custID = C.custID
GROUP BY county, itemID, gender;
```

Drill-down

```
SELECT city, itemID, gender, SUM(price)
FROM Sales F, Store S, Customer C
WHERE F.storeID = S.storeID AND F.custID = C.custID
GROUP BY city, itemID, gender
```

Drill-down Example 2 (Dimension)

- Use Drill-down on total sales by item and county to find total sales by item and gender for each county.



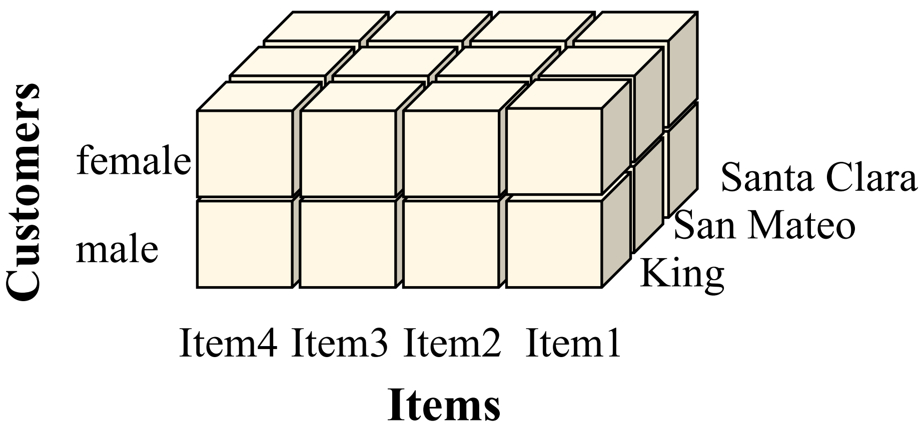

```
SELECT county, itemID, SUM(price)
FROM    Sales F, Store S
WHERE F.storeID = S.storeID
GROUP BY county, itemID;
```

Bring back the customer table

```
SELECT country, itemID, gender, SUM(price)
FROM Sales F, Store S, Customer C
WHERE F.storeID = S.storeID AND F.custID = C.custID
GROUP BY country, itemID, gender
```

OLAP Queries – Slicing

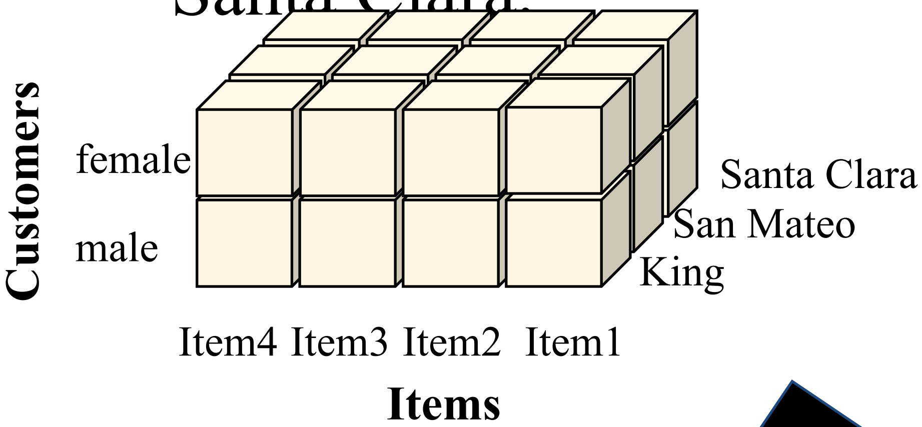
- The slice operation produces a slice of the cube by picking a specific value for one of the dimensions.
- To start our example, let's specify:
  - Total sales by item and gender for each county




```
SELECT county, itemID, gender,
           SUM(price)
FROM    Sales F, Store S, Customer C
WHERE F.storeID = S.storeID AND
           F.custID = C.custID
GROUP BY county, itemID, gender;
```

Slicing Example 1

- Use Slicing on total sales by item and gender for each county to find total sales by item and gender for Santa Clara.



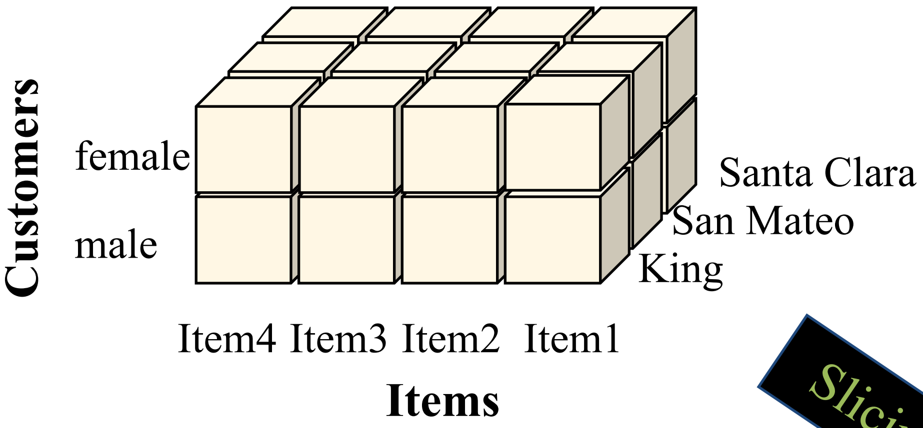

```
SELECT county, itemID, gender,
            SUM(price)
FROM     Sales F, Store S, Customer C
WHERE F.storeID = S.storeID AND
            F.custID = C.custID
GROUP BY county, itemID, gender;
```


```
SELECT itemID, gender, SUM(price)
FROM Sales F, Store S, Customer C
WHERE F.storeID = S.storeID
        AND F.custID = C.custID
        AND S.county = 'Santa Clara'
GROUP BY itemID, gender
```

look at one slice of it

Slicing Example 2

- Use Slicing on total sales by item and gender for each county to find total sales by gender and county for Jacket

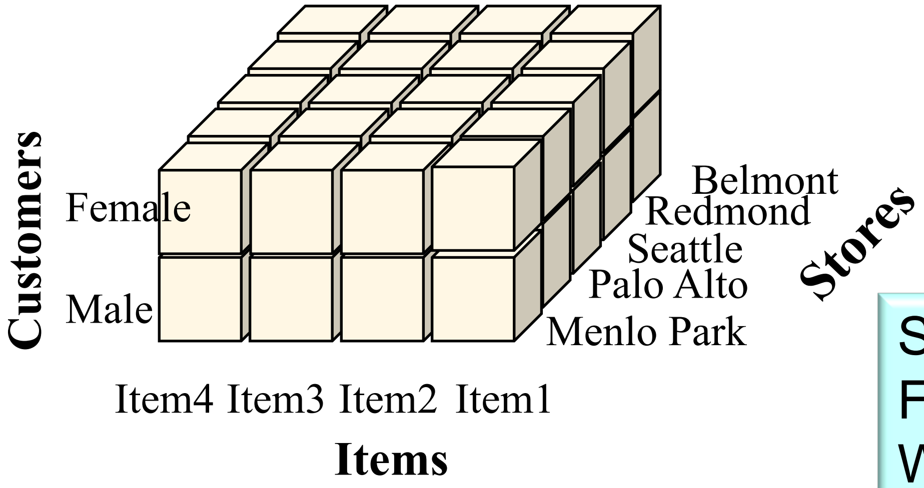



```
SELECT county, itemID, gender,
            SUM(price)
FROM    Sales F, Store S, Customer C
WHERE F.storeID = S.storeID AND
            F.custID = C.custID
GROUP BY county, itemID, gender;
```


```
SELECT county, gender, SUM(price)
FROM Sales F, Store S, Customer C, Item I
WHERE F.storeID = S.storeID AND F.custID = C.custID
        AND I.itemID = F.itemID
        AND I.category = 'Jacket'
GROUP BY county, gender
```

OLAP Queries – Dicing

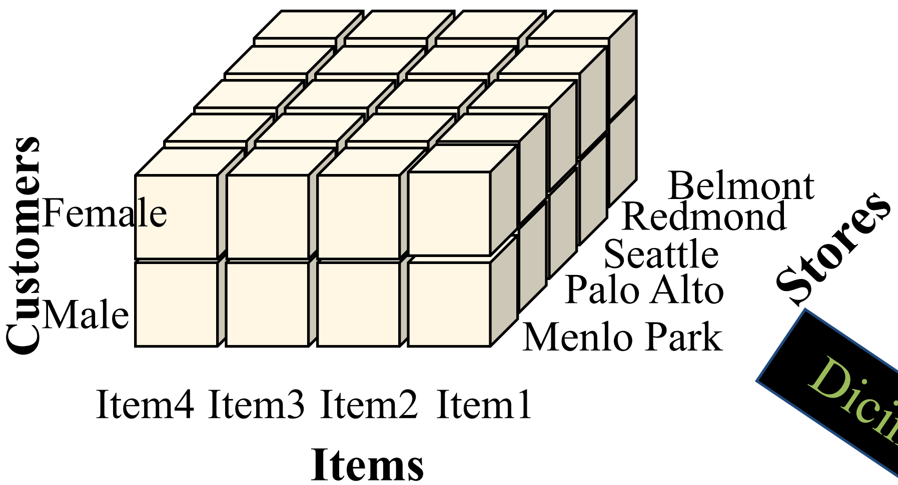
- The dice operation produces a sub-cube by picking specific values for multiple dimensions.
- To start our example, let's specify:
  - Total sales by gender, item, and city




```
SELECT city, itemID, gender, SUM(price)
FROM    Sales F, Store S, Customer C
WHERE F.storeID = S.storeID AND
            F.custID = C.custID
GROUP BY city, itemID, gender;
```

Dicing Example 1

- Use Dicing on total sales by gender, item, and city to find total sales by gender, category, and city for red items in the state of California (CA).




```
SELECT city, itemID, gender, SUM(price)
FROM    Sales F, Store S, Customer C
WHERE F.storeID = S.storeID AND
            F.custID = C.custID
GROUP BY city, itemID, gender;
```


```
SELECT category, city, gender, SUM(price)
FROM Sales F, Store S, Customer C, Item I
WHERE F.storeID = S.storeID AND F.custID = C.custID
          AND F.itemID = I.itemID
          AND color = 'red' AND state = 'CA"
GROUP BY category, city, gender
```

Clicker Question

- Consider a fact table Sales(saleID, itemID, color, size, qty, unitPrice), and the following three queries:
- Q1: SELECT itemID, color, size, Sum(qty*unitPrice) FROM Sales GROUP BY itemID, color, size
- Q2: SELECT itemID, size, Sum(qty*unitPrice) FROM Sales GROUP BY itemID, size
- Q3: SELECT itemID, size, Sum(qty*unitPrice) FROM Sales WHERE size < 10 GROUP BY itemID, size
- Which of the following statements is correct?

  A: Going from Q2 to Q3 is an example of roll-up. silcing

  B: Going from Q2 to Q1 is an example of drill-down.

  C: Going from Q3 to Q2 is an example of roll-up. unsilcing

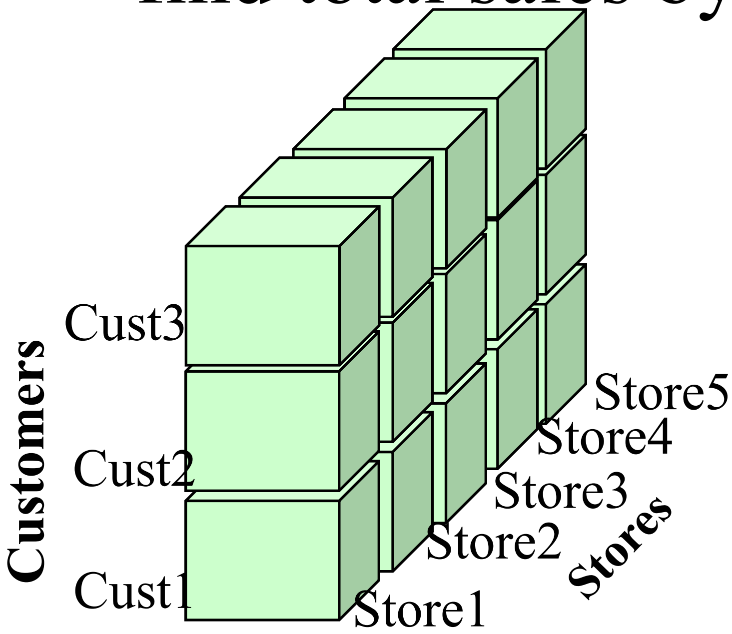
  D: Going from Q1 to Q2 is an example of drill-down. roll-up

OLAP Queries – Pivoting

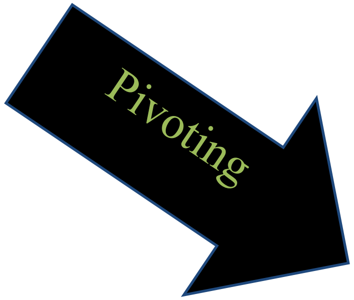
- Pivoting is a visualization operation that allows an analyst to rotate the cube in space in order to provide an alternative presentation of the data.

Pivoting Example 1

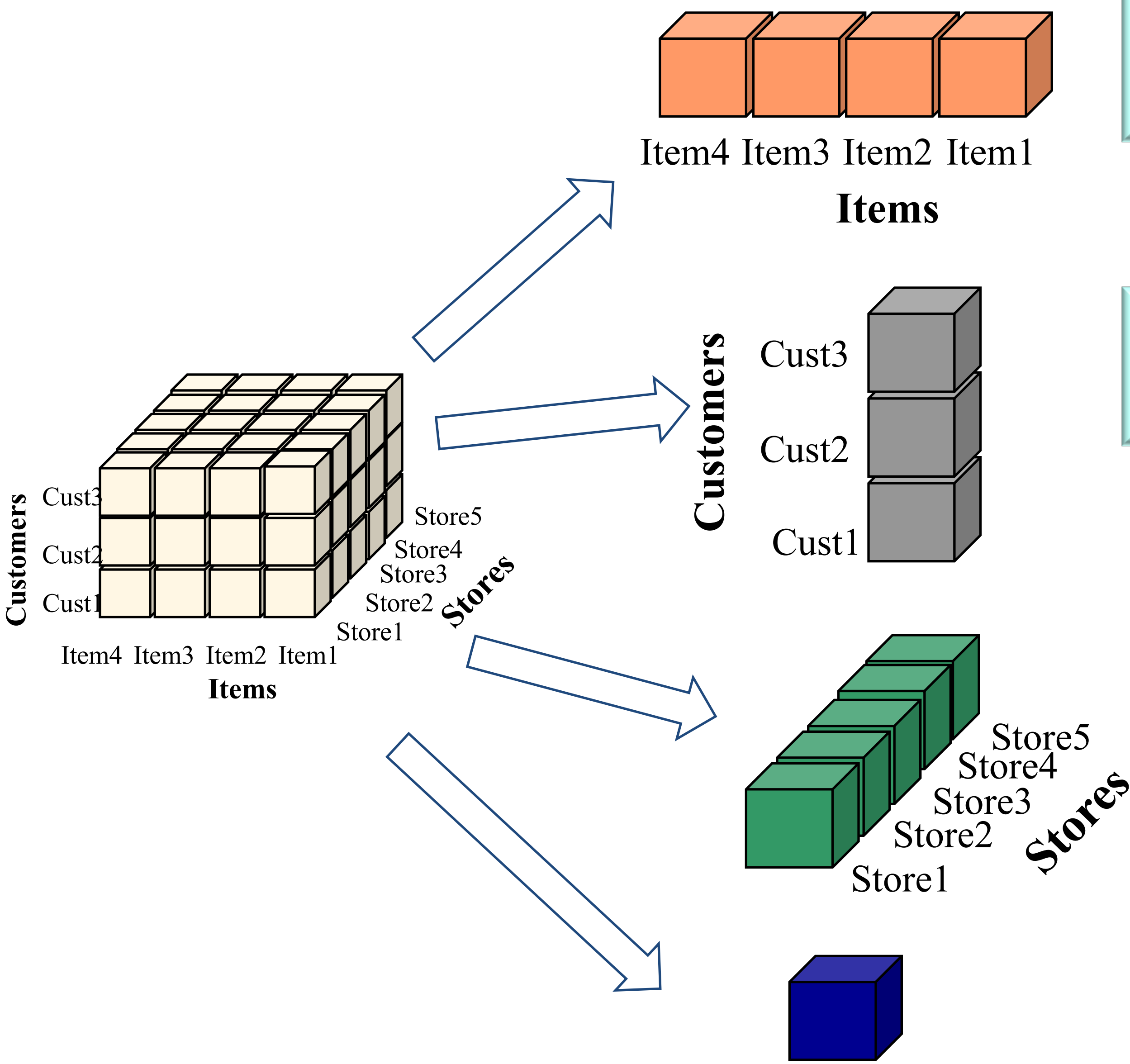
- From total sales by store and customer pivot to find total sales by item and store.




```
SELECT storeID, custID, sum(price)
FROM  Sales
GROUP BY storeID, custID;
```



Aggregating over Multiple Fact Tables




```
SELECT itemID, sum(price)
FROM    Sales
GROUP BY itemID;
```

```
SELECT custID, sum(price)
FROM    Sales
GROUP BY custID;
```

```
SELECT storeID, sum(price)
FROM    Sales
GROUP BY storeID;
```

```
SELECT  sum(price)
FROM    Sales
```

Motivation

Data Warehousing

On-Line Analytical Processing

ROLLUP and CUBE Operators

Star Schema vs Snowflake Schema

Combining Fact Data and Dimensions

- How can we run queries that contain both fact data and dimensions?

state	county	city	sum(price)
CA	San Mateo	Belmont	225
CA	San Mateo	Menlo Park	625
CA	Santa Clara	Mountain View	805
CA	Santa Clara	Palo Alto	325
WA	King	Redmond	795
WA	King	Seattle	575
CA	San Mateo	*NULL*	850
CA	Santa Clara	*NULL*	1130
WA	King	*NULL*	1370
CA	*NULL*	*NULL*	1980
WA	*NULL*	*NULL*	1370
NULL	*NULL*	*NULL*	3350

State

County

City

Combining Fact Data and Dimensions

```
SELECT state, county, city, sum(price)
FROM     Sales F, Store S
WHERE F.storeID = S.storeID
GROUP BY state, county, city
UNION
SELECT state, county, Null, sum(price)
FROM     Sales F, Store S
WHERE F.storeID = S.storeID
GROUP BY state, county
UNION
SELECT state, Null, Null, sum(price)
FROM     Sales F, Store S
WHERE F.storeID = S.storeID
GROUP BY state
UNION
SELECT Null, Null, Null, sum(price)
FROM     Sales F, Store S
WHERE F.storeID = S.storeID
```

state	county	city	sum(price)
CA	San Mateo	Belmont	225
CA	San Mateo	Menlo Park	625
CA	Santa Clara	Mountain View	805
CA	Santa Clara	Palo Alto	325
WA	King	Redmond	795
WA	King	Seattle	575
CA	San Mateo	*NULL*	850
CA	Santa Clara	*NULL*	1130
WA	King	*NULL*	1370
CA	*NULL*	*NULL*	1980
WA	*NULL*	*NULL*	1370
NULL	*NULL*	*NULL*	3350

WITH ROLLUP in one dimension

```
Select   dimension-attrs, aggregates
From     tables
Where    conditions
Group By dimension-attrs With Rollup
```
(order is matter)

- Can be used in dimensions that are organized in a hierarchy:

```
SELECT state, county, city, sum(price)
FROM     Sales F, Store S
WHERE F.storeID = S.storeID
GROUP BY state, county, city WITH ROLLUP
```

(supported by MySQL)

State

County

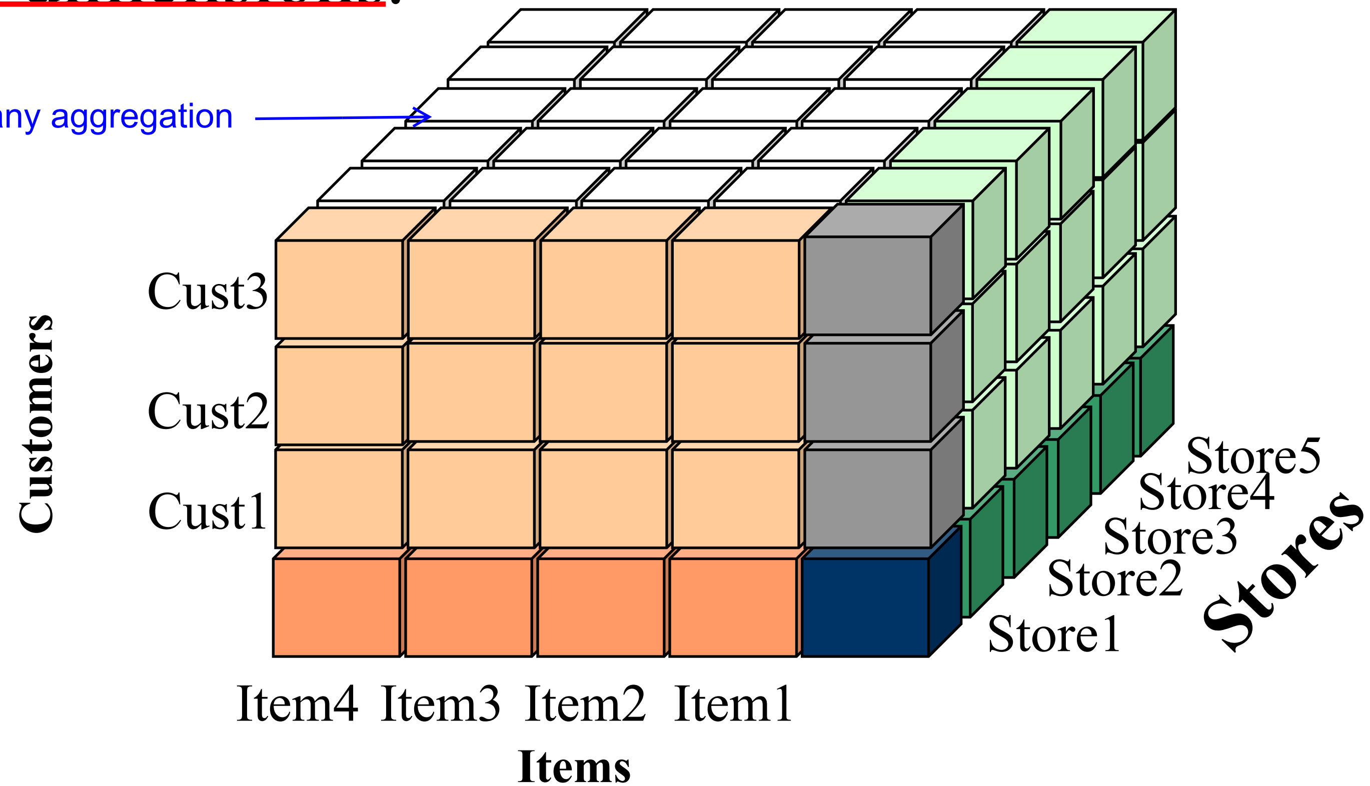
City

state	county	city	sum(price)
CA	San Mateo	Belmont	225
CA	San Mateo	Menlo Park	625
CA	Santa Clara	Mountain View	805
CA	Santa Clara	Palo Alto	325
WA	King	Redmond	795
WA	King	Seattle	575
CA	San Mateo	*NULL*	850
CA	Santa Clara	*NULL*	1130
WA	King	*NULL*	1370
CA	*NULL*	*NULL*	1980
WA	*NULL*	*NULL*	1370
NULL	*NULL*	*NULL*	3350

Data Cube with all the dimensions

- A *data cube* is a *k*-dimensional object containing both fact data and dimensions.

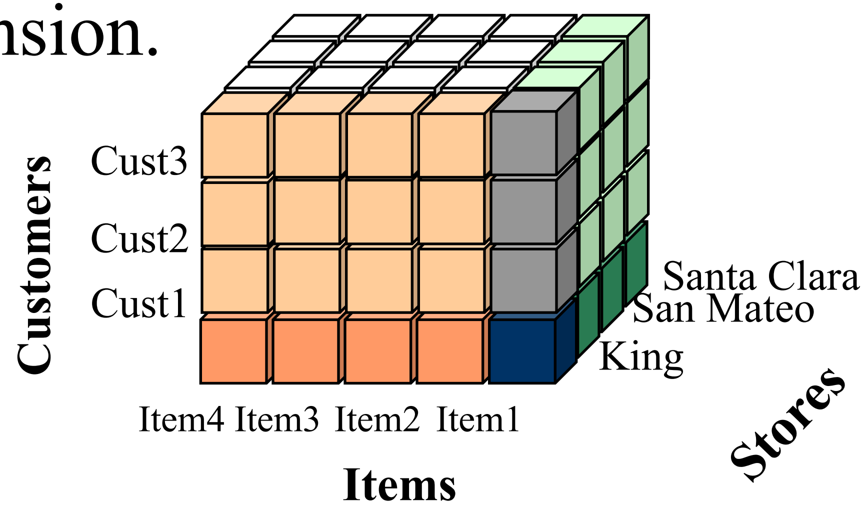
data without any aggregation



- A cube contains pre-calculated, aggregated, summary information to yield fast queries.

Data Cube (cont.)

- The small, individual blocks in the multidimensional cube are called cells, and each cell is uniquely identified by the members from each dimension.



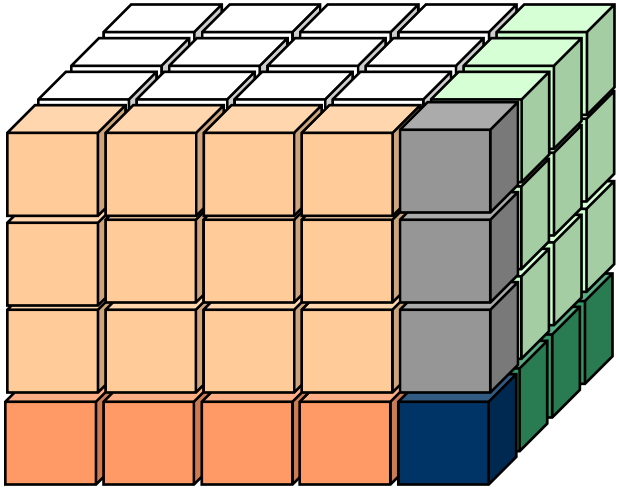
- The cells contain a measure group, which consists of one or more numeric measures. These are facts (or aggregated facts). An example of a measure is the dollar value in sales for a particular product

The CUBE Operator

- Roll-up, Drill-down, Slicing, Dicing, and Pivoting operations are expensive.
- SQL:1999 extended GROUP BY to support CUBE (and ROLLUP)
- GROUP BY CUBE provides efficient computation of multiple granularity aggregates by sharing work (e.g., passes over fact table, previously computed aggregates)

Clicker Question

- If we have 2 stores, 5 items, and 10 customers, how many potential "entries" are there in the data cube? (The cube diagram is just an arbitrary example.)



- A: 17
- B: 100
- C: 117
- D: 198

(2+1)*(5+1)*(10+1) = 198

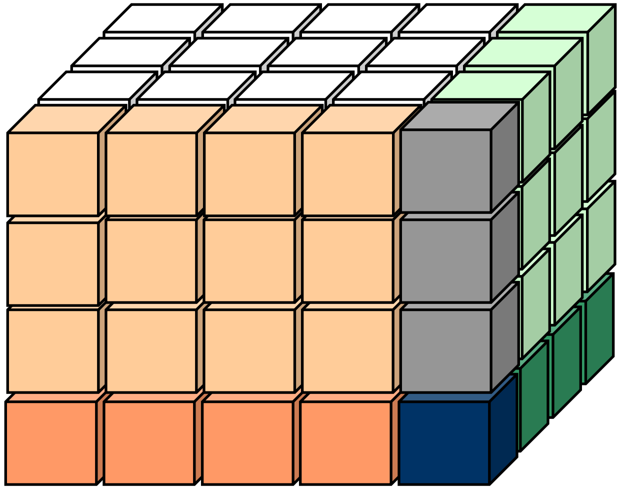
- E: none of the above

Clicker Question

- How many standard SQL queries are required for computing all of the cells of the cube?

2^3 = 8

- A: 2
- B: 4
- C: 6
- D: 8 7(aggregation) + 1(raw data)
- E: 10



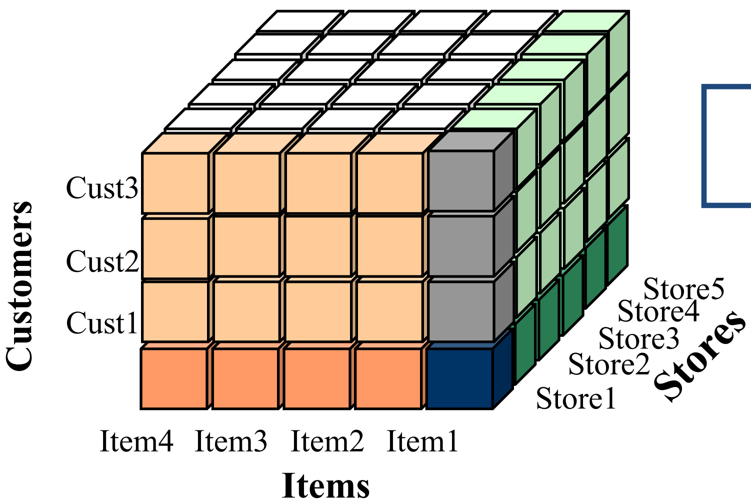
The CUBE Operator (cont.)

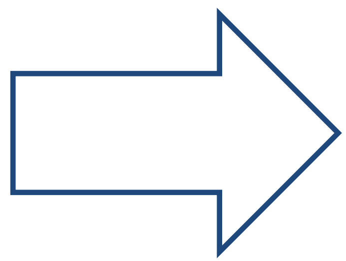
- Generalizing the previous example, if there are k dimensions, we have 2^k possible SQL GROUP BY queries that can be generated through pivoting on a subset of dimensions. A CUBE BY operator generated that.
  - It Is equivalent to rolling up Sales on all eight subsets of the set {storeID, itemID, custID }.
  - Each roll-up corresponds to an SQL query of the form:

```
SELECT SUM (price)
FROM     Sales S
GROUP BY grouping-list
```

Lots of research on optimizing the CUBE operator!

Representing a Cube in a Two-Dimensional Table





storeID	itemID	custID	Sum
store1	item1	cust1	10
store1	item1	Null	70
store1	Null	cust1	145
store1	Null	Null	325
Null	item1	cust1	10
Null	item1	Null	135
Null	Null	cust1	670
Null	Null	Null	3350

....

- Add to the original cube: faces, edges, and corners … which are represented in the 2-D table using NULLs.

WITH CUBE Example Implemented WITH ROLLUP

(not available by MySQL)

- Implement the WITH CUBE operator using the WITH ROLLUP operator

```
SELECT storeID, itemID, custID, sum(price)
FROM Sales
GROUP BY storeID, itemID, custID with ROLLUP
UNION

SELECT storeID, itemID, custID, sum(price)
FROM Sales
GROUP BY itemID, custID, storeID with ROLLUP
UNION

SELECT storeID, itemID, custID, sum(price)
FROM Sales
GROUP BY custID, storeID, itemID with ROLLUP;
```

Clicker Question

- Consider a fact table Facts(D1, D2, D3, x), and the following three queries:

Q1: Select D1, D2, D3, Sum(x) From Facts Group By D1, D2, D3
Q2: Select D1, D2, D3, Sum(x) From Facts Group By D1, D2, D3 with cube
Q3: Select D1, D2, D3, Sum(x) From Facts Group By D1, D2, D3 with rollup

- Suppose attributes D1, D2, and D3 have n1, n2, and n3 different values respectively, and assume that each possible combination of values appears at least once in table Facts. Pick the one tuple (a,b,c,d,e,f) in the list below such that when n1=a, n2=b, and n3=c, then the result sizes of queries Q1, Q2, and Q3 are d, e, and f respectively.

d = a*b*c   e = (a+1)*(b+1)*(c+1)   f = d + a*b + a + 1

- A: (2, 2, 2, 8, 64, 15)
- B: (5, 4, 3, 60, 64, 80)
- C: (5, 10, 10, 500, 726, 556)
- D: (4, 7, 3, 84, 160, 84)

Hint: It may be helpful to first write formulas describing how d, e, and f depend on a, b, and c.

Motivation

Data Warehousing

On-Line Analytical Processing

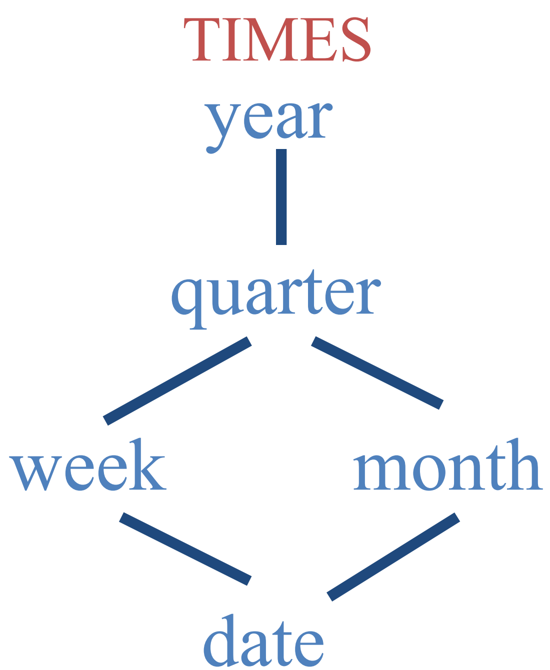
ROLLUP and CUBE Operators

Star Schema vs Snowflake Schema

"Date" or "Time" Dimension

- Date or Time is a special kind of dimension.

- It has some special and useful OLAP functions.
    - e.g., durations or time spans, fiscal years, calendar years, and holidays
    - Business intelligence reports often deal with time-related queries such as comparing the profits from this quarter to the previous quarter … or to the same quarter in the previous year.



Star Schema (Reminder)

- The schema that is very common in OLAP applications is called a star schema:
    - One table for the fact table
    - One table per dimension

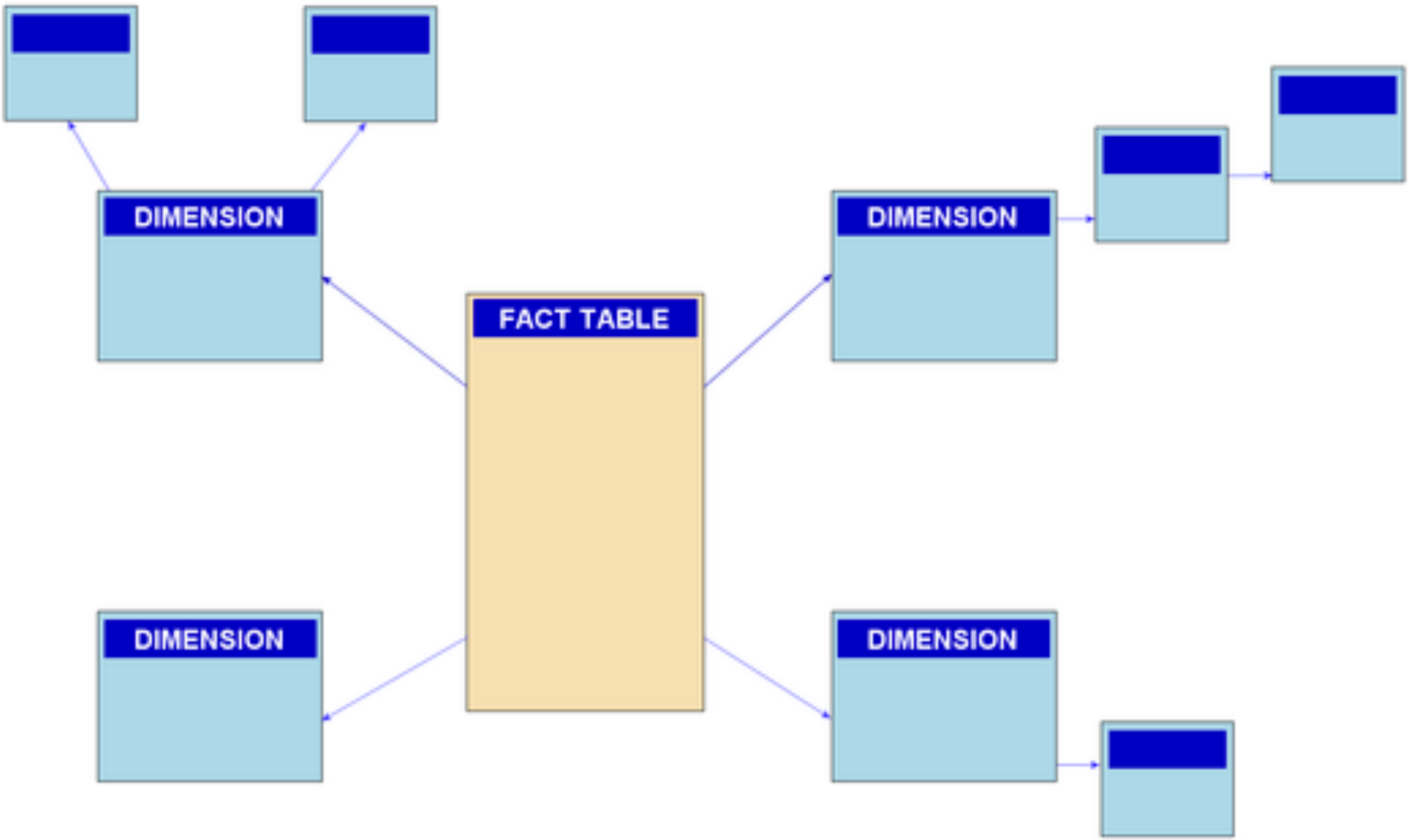
- The fact table is in BCNF.

- The dimension tables are not normalized.

Snowflake Schema

- The alternative organization is a snowflake schema:
  – each dimension is normalized into a set of tables
  – usually, one table per level of hierarchy, per dimension
- Example: TIMES table would be split into:
  – TIMES(timeid, date)
  – DWEEK(date, week)
  – DMONTH(date, month)
- Snowflake schema features:
  – Query formulation is inherently more complex (possibly many joins per dimension).
- The star schema is more popular, and is gaining interest.

Snowflake Schema example



Source: http://en.wikipedia.org/wiki/Snowflake_schema

Star vs. Snowflake

	Star	Snowflake
Ease of maintenance	Has redundant data and hence is less easy to maintain/change	No redundancy, schemas are easier to maintain and change.
Ease of Use	Lower complex query writing; easier to understand	More complex queries and hence less easy to understand
Query Performance	Fewer foreign keys and hence shorter query execution time (faster)	More foreign keys and hence longer query execution time (slower)
Joins	Fewer Joins	More Joins
Dimension table	A single dimension table for each dimension	May have more than one dimension table for each dimension
When to use	Star schema is the default choice	When dimension table is relatively big in size, or we expect a lot of updates
Normalization	Dimension Tables are not Normalized	Dimension Tables are Normalized

Learning Objectives Revisited

Description	Tag
Compare and contrast OLAP and OLTP processing (e.g., focus, clients, amount of data, abstraction levels, concurrency, and accuracy).	Data warehousing
Given a multidimensional cube, write regular SQL queries that perform roll-up, drill-down, slicing, dicing, and pivoting operations on the cube.	
Use the SQL:1999 standards for aggregation (e.g., GROUP BY CUBE) to efficiently generate the results for multiple views.	
Explain the differences between a star schema design and a snowflake design for a data warehouse, including potential tradeoffs in performance.	