Advanced DB

CHAPTER 18
DATA ANALYSIS & MINING

Chapter 18: Data Analysis and Mining

- Decision Support Systems
- Data Analysis and OLAP
- Data Warehousing
- Data Mining

Online Transaction Processing

- Transaction processing activities are needed to capture and process data involved in transactions
- Online transaction processing systems
  - Real-time systems that capture and process transactions immediately
  - Can help provide superior service to customers and other trading partners
- An integral part of the every day business of an enterprise
  - Order processing system of an electronics distributor
  - Banking system of a bank
  - Order control and billing system of a hospital
  - Switches and billing system of a telephone company

Decision Support Systems

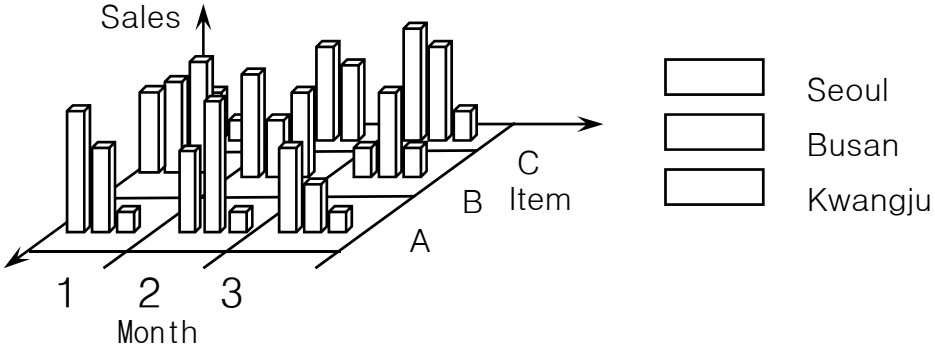
- Decision-support systems are used to make business decisions
  - What items to stock?
  - What insurance premium to change?
  - To whom to send advertisements?
- Examples of data used for making decisions
  - Retail sales transaction details
  - Customer profiles (income, age, gender, etc.)
- Identifying patterns in customer behavior and Using the patterns to make business decisions
  - A Sudden spurt in purchases of flannel shirts
  - Most of small sports cars are bought by young woman with more than $50,000 annual salary
- EIS (Executive Information System)
  - A reporting subsystem for decision makers of an enterprise
  - Summary, trend, comparative, integrated information

Multidimensional View

- Data that can be modeled as dimension attributes and measure attributes

```
sales(item-name, color, size, number)
```

- *Measure* attributes
  - example: *number*
  - Measure some value
  - Can be aggregated upon



- *Dimension* attributes
  - example: *item-name, color, size*
  - Define the dimensions on which measure attributes, and summaries of measure attributes

Online Analytical Processing (OLAP)

- Interactive analysis of summary information
- Statistical analysis often requires grouping on multiple attributes
- Although complex statistical analysis is best left to statistics packages, database should support simple, commonly used, forms of data analysis
- Several SQL extensions have been developed to support OLAP tools

Cross Tabulation of *sales* by *item-name* and *color*

size: **all**

item-name \ *color*	dark	pastel	white	Total
skirt	8	35	10	53
dress	20	10	5	35
shirt	14	7	28	49
pant	20	2	5	27
Total	62	54	48	164

- An example of a **cross-tabulation** (**cross-tab**)
  - also referred to as a **pivot-table**.
  - Values for one of the dimension attributes form the row headers
  - Values for another dimension attribute form the column headers
  - Other dimension attributes are listed on top
  - Values in individual cells are (aggregates of) the values of the dimension attributes that specify the cell.

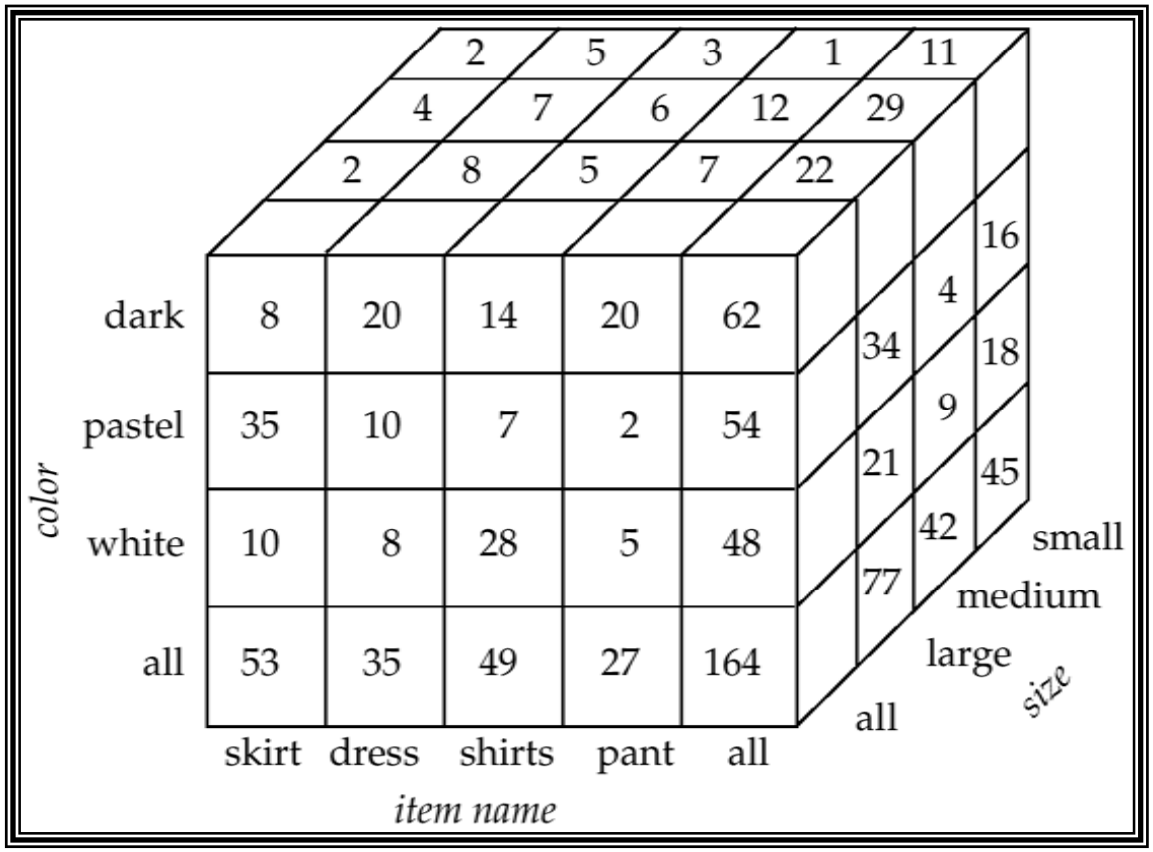
Relational Representation of Cross-tabs

- Cross-tabs can be represented as relations
  - We use the value **all** is used to represent aggregates
  - The SQL:1999 standard actually uses null values in place of **all** despite confusion with regular null values

item-name	*color*	*number*
skirt	dark	8
skirt	pastel	35
skirt	white	10
skirt	**all**	53
dress	dark	20
dress	pastel	10
dress	white	5
dress	**all**	35
shirt	dark	14
shirt	pastel	7
shirt	white	28
shirt	**all**	49
pant	dark	20
pant	pastel	2
pant	white	5
pant	**all**	27
all	dark	62
all	pastel	54
all	white	48
all	**all**	164

Data Cube

- A **data cube** is a multidimensional generalization of a cross-tab
  - Can have *n* dimensions; we show 3 below
  - Cross-tabs can be used as views on a data cube

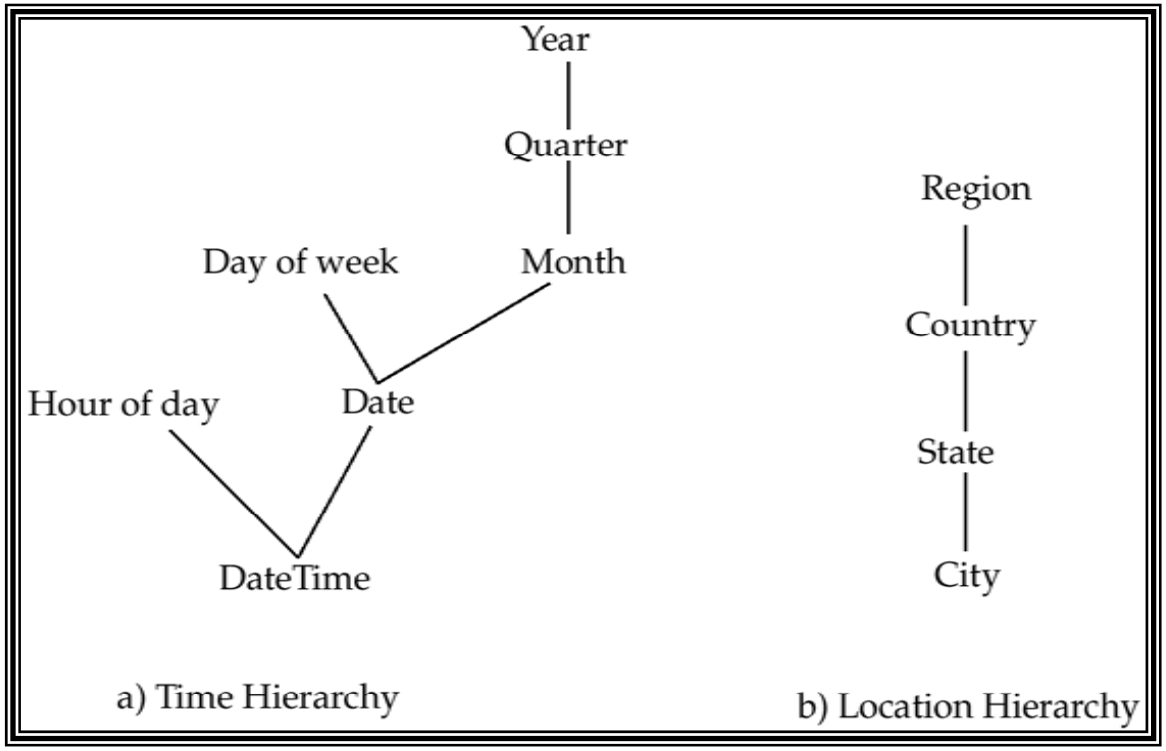


Online Analytical Processing Operations

- **Pivoting**: changing the dimensions used in a cross-tab is called
- **Slicing:** creating a cross-tab for fixed values only
  - Sometimes called **dicing**, particularly when values for multiple dimensions are fixed.
- **Rollup:** moving from finer-granularity data to a coarser granularity
- **Drill down:** The opposite operation - that of moving from coarser-granularity data to finer-granularity data

Hierarchies on Dimensions

- **Hierarchy** on dimension attributes: lets dimensions to be viewed at different levels of detail
  - E.g. the dimension DateTime can be used to aggregate by hour of day, date, day of week, month, quarter or year



Cross Tabulation With Hierarchy

- Cross-tabs can be easily extended to deal with hierarchies
  - Can drill down or roll up on a hierarchy

category	*item-name*					
		dark	pastel	white	total	
womenswear	skirt	8	8	10	53	
	dress	20	20	5	35	
	subtotal	28	28	15		88
menswear	pants	14	14	28	49	
	shirt	20	20	5	27	
	subtotal	34	34	33		76
total		62	62	48		164

OLAP Implementation

- MOLAP (multidimensional OLAP)
  - Multidimensional arrays in memory to store data cubes (earliest OLAP systems)
- ROLAP (relational OLAP)
  - Data stored in relational database
- HOLAP (hybrid OLAP)
  - Store some summaries in memory
  - Store the base data and other summaries in rdb
- Many OLAP systems are implemented as client-server systems

OLAP Implementation (Cont.)

- Precompute *all* possible aggregates in order to provide online response
  - 2^n combinations of **group by**
- Precompute some aggregates, and compute others on demand from one of the precomputed aggregates
  - Can compute aggregate on (*item-name, color*) from an aggregate on (*item-name, color, size*)
- Several optimizations available for computing multiple aggregates
  - Can compute aggregate on (*item-name, color*) from an aggregate on (*item-name, color, size*)
  - Can compute aggregates on (*item-name, color, size*), (*item-name, color*), and (*item-name*) using a single sorting of the base data

Extended Aggregation in SQL:1999

- The **cube** operation computes union of **group by**'s on every subset of the specified attributes
- E.g. consider the query

  select *item-name, color, size,* **sum**(*number*)
  from *sales*
  group by cube(*item-name, color, size*)

  computes the union of eight different groupings of the *sales* relation:

  { (*item-name, color, size*), (*item-name, color*), (*item-name, size*),
  (*color, size*), (*item-name*), (*color*), (*size*), ( ) }

  where ( ) denotes an empty **group by** list.
- For each grouping, the result contains the null value for attributes not present in the grouping.

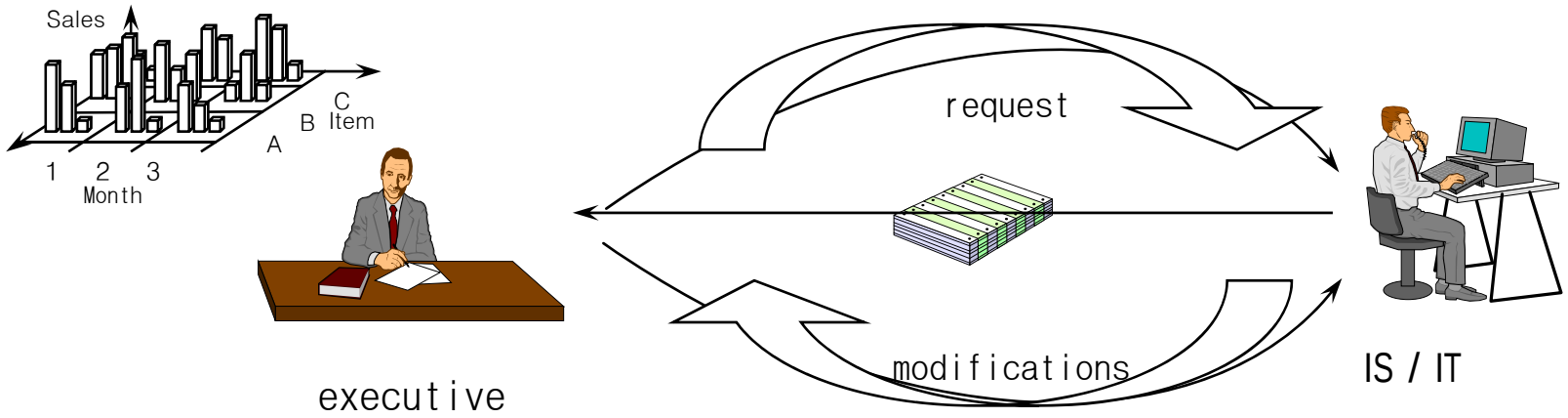
Extended Aggregation (Cont.)

- Relational representation of cross-tab of figure 18.2 can be computed by

select *item-name*, *color*, **sum**(*number*)
from *sales*
group by cube(*item-name, color*)

Data Warehouse - Motivation

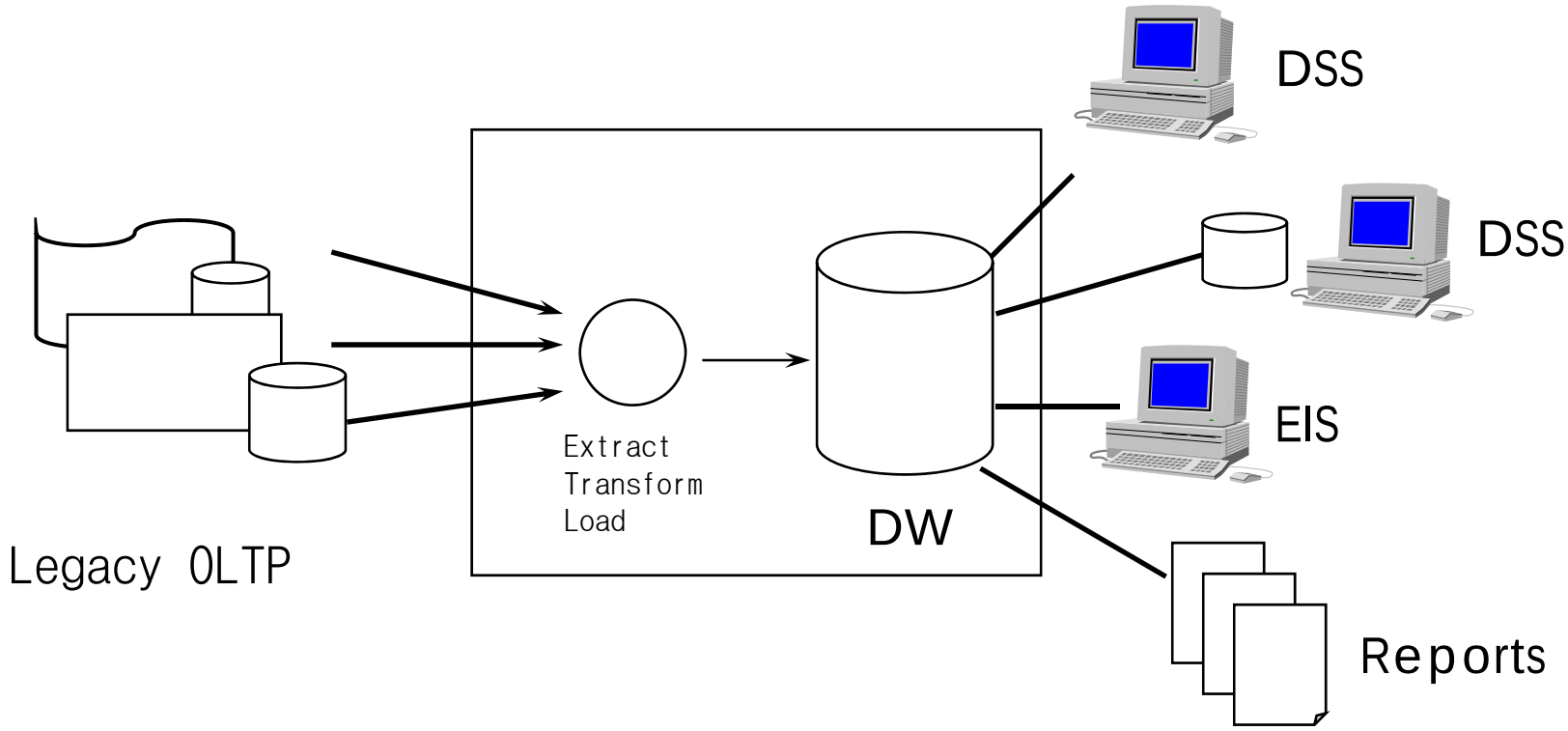
- Decision Support Problems in conventional systems (W. H. Inmon)
  - Credibility of data: no time basis, algorithmic difference, no common source
  - Productivity: multiple sources in multiple environment, insufficient information, repeated work
  - Data into information: data from multiple applications, consistency, past history
- Conventional DSS Cycle



Solution 1: Rebuild System

- Pros
  - A new & consistent integrated data model
  - Opportunity adopt new HW, SW, & methodology
- Cons
  - Large long-term project
  - Change in management/organization/operations during project
  - Transition risks
  - Still operations (OLTP) oriented
    - DSS is second priority at best

Solution 2: DW-based Restructuring



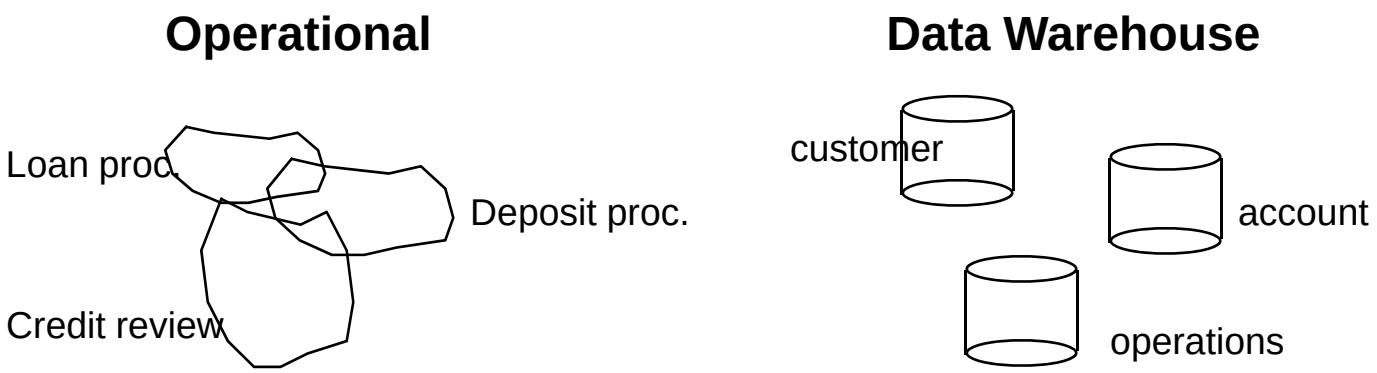
Data Processing (Transaction) ⇦⇨ **Information Processing**

Data Warehouse – Features

- A repository (or archive) of information gathers from multiple sources, stored under a unified schema, at a single site
  - Once gathered, the data are stored for a long time, permitting access to historical data
  - Provide the user a single consolidated interface to data, making decision-support queries easier to write
  - Online transaction-processing systems are not affected by the decision-support workload
- Key Features
  - Subject oriented
  - Integrated
  - Time-variant, historical
  - Nonvolatile

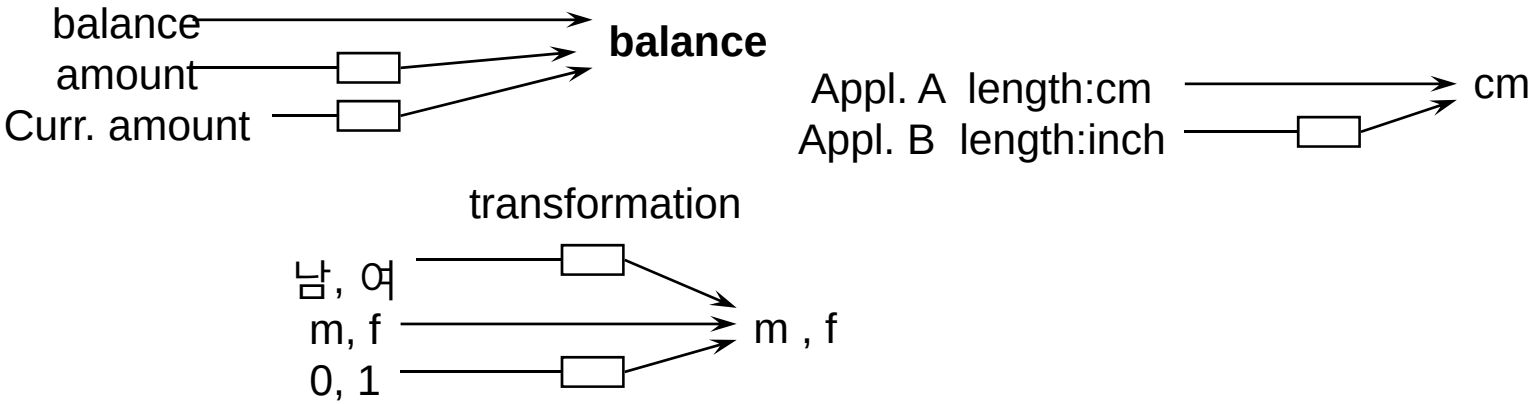
Data Warehouse – Features

- Subject oriented
  - Data in a OLTP system are designed/managed/used with a process/function-oriented view
  - Data in a DW are designed/managed/used with a subject-oriented view
    - Focus of a DSS is in the subjects of an organization (not the processes)
    - Not all data from OLTP need to be moved to the DW



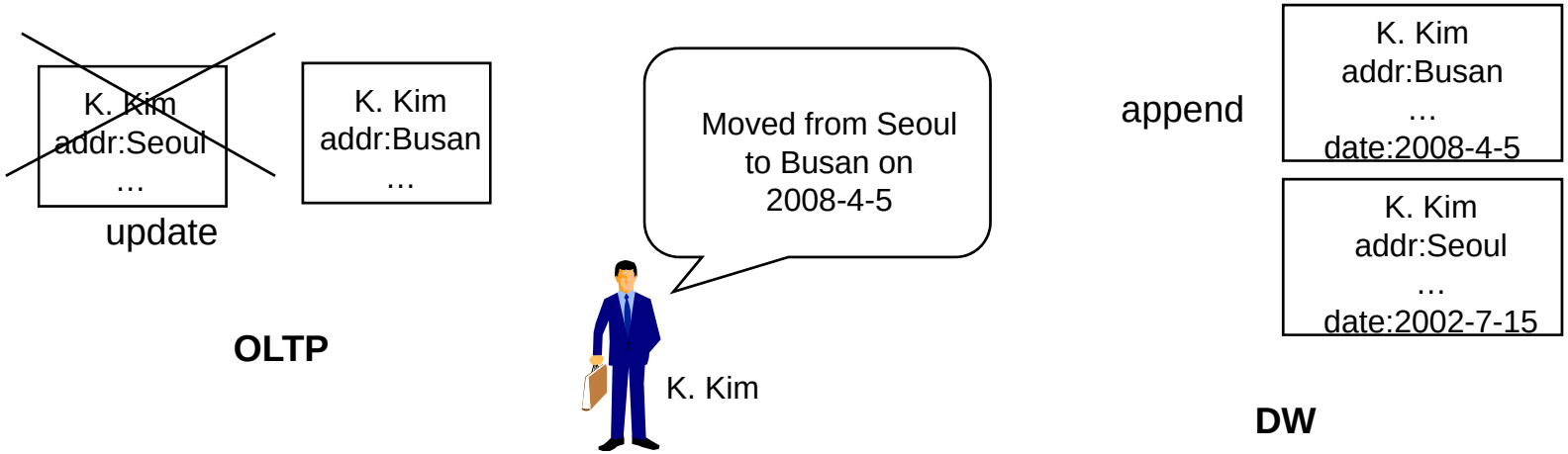
Data Warehouse – Features

- Integrated
  - Names: table names and column names
  - Units, Values and codes
  - Aggregation and consolidation
  - A consistent source of information



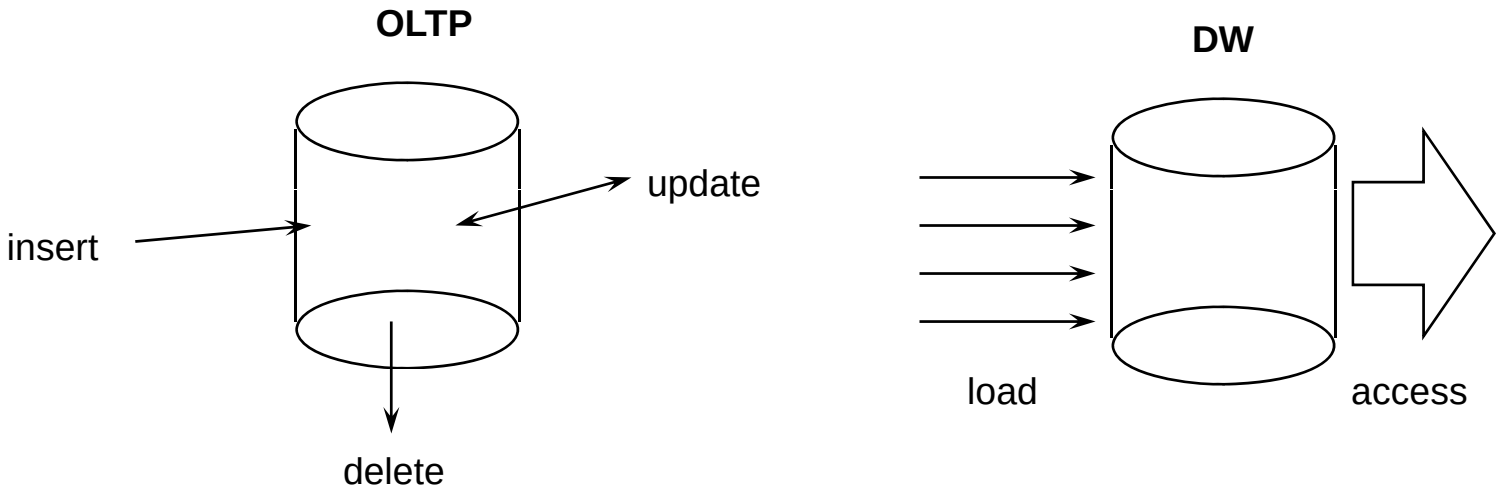
Data Warehouse – Features

- Time Variant
  - OLTP: represents the current state of the domain world
  - DW: need to log the history of a subject
    - Monthly call charges for K. Kim for last year
    - Weekly sales for each brand (or item)
  - Time element must be present in the key



Data Warehouse – Features

- Nonvolatile
  - Little (or no) updates
  - Most operations are new inserts (load)
  - Normalization is not important

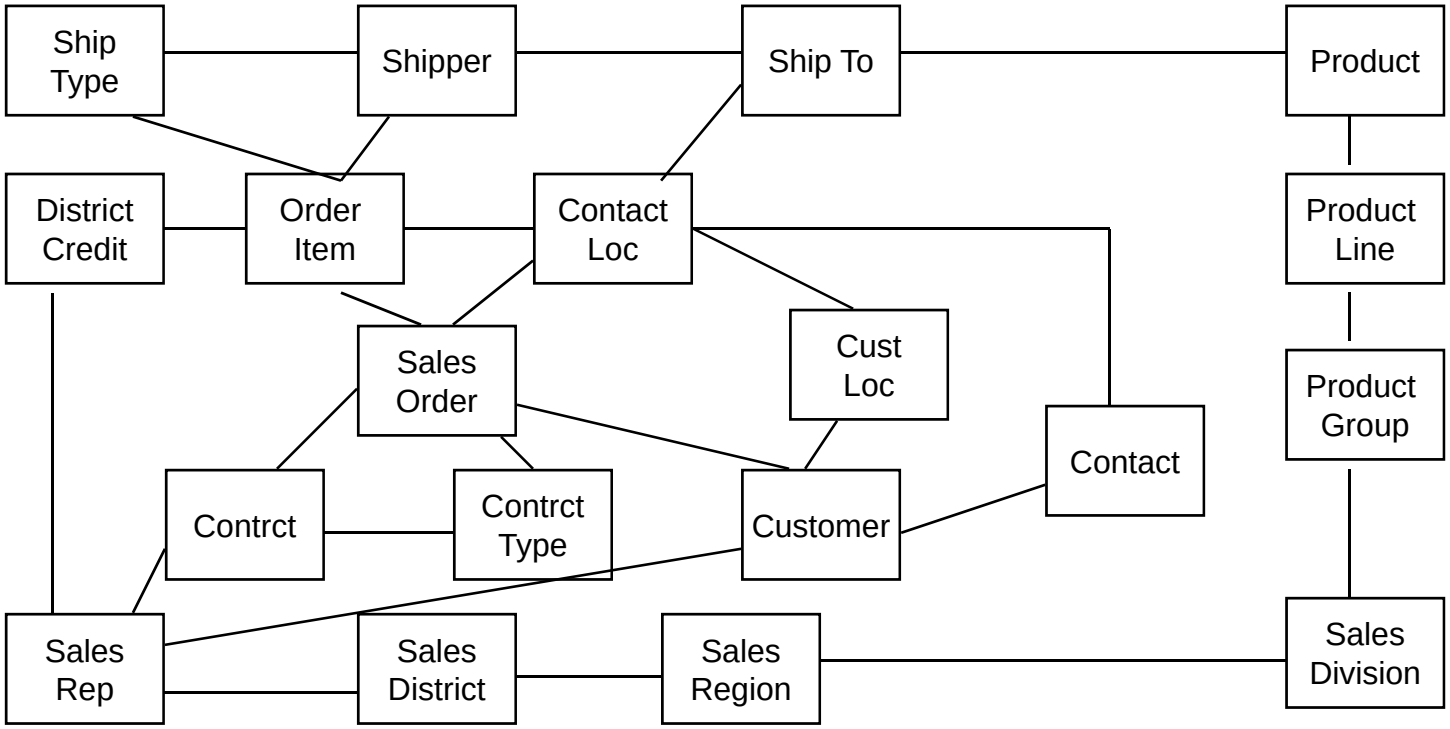


Data Modeling

- DB Design for OLTP
  - Normalization: minimize update anomalies by eliminating redundancy
  - Transactions usually access (relatively) small number of records
  - Node – table / Link – join path
  - Ad hoc join paths – not easy to predict access patterns

- DB Design for Data warehouse
  - Historical and summary oriented
  - Analytical "views"
  - Non-volatile
  - Dimensional model
    - Fact tables and dimensional tables

Data Modeling – OLTP



Q: The top selling brands on holidays in Q1?

Multi-dimensional Modeling – Fact table

- Store all transactions for factual data about a business
- Have a multi-part key (composite key)
- Each element of the key is a FK to a single dimension table
- The remaining fields in fact table are known as Facts
- Facts are almost numeric – target of aggregation (count, sum, average, …)
- much larger than dimension tables

Multi-dimensional Modeling – Dimension table

- Dimensions are attribute of facts
- Store hierarchical relationships between different grouping or characteristics of those transactions
- Dimension hierarchy
  - Time : Year, Quarter, Week, Day
  - Product : Group, Subgroup, Department, Class, Item
  - Territory : Division, Country, Region, Distinct
  - Unit of drill-down & roll-up
- Have single key that is joined to a Fact table
- The remaining fields are called attributes
- Dimension attributes
  - Textual, Discrete
  - Source of constraints and grouping columns

Hierarchies on Dimensions

- The different levels of detail for an attribute can be organized into a hierarchy

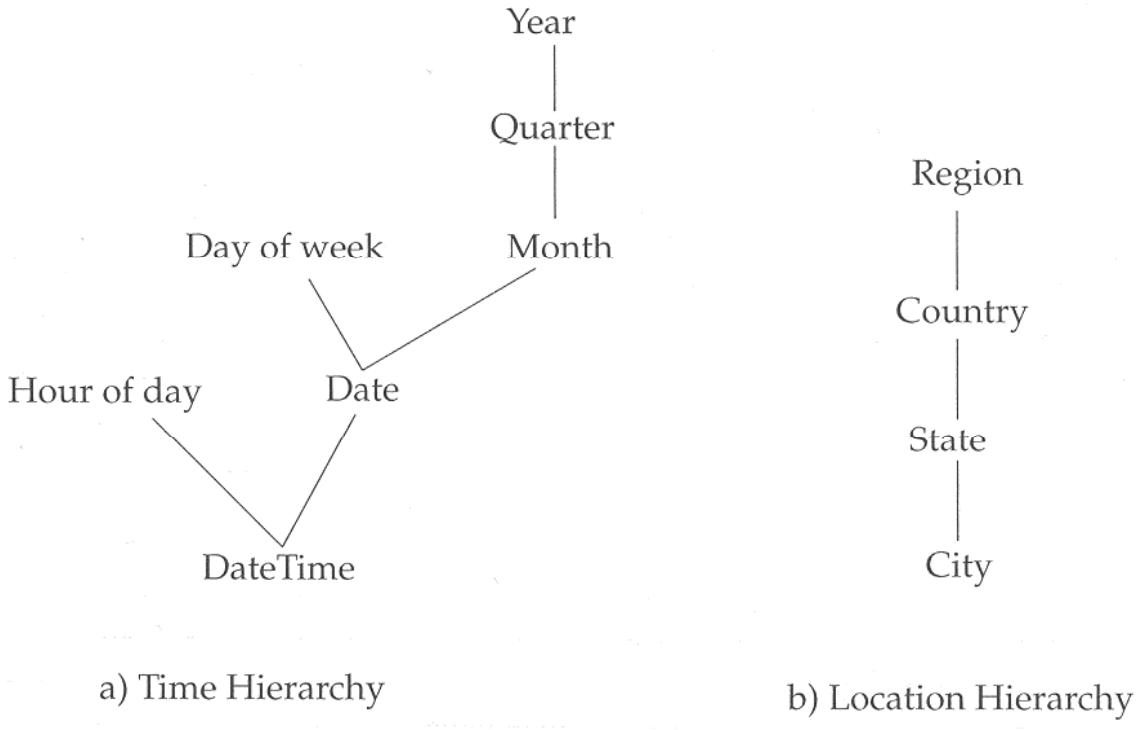
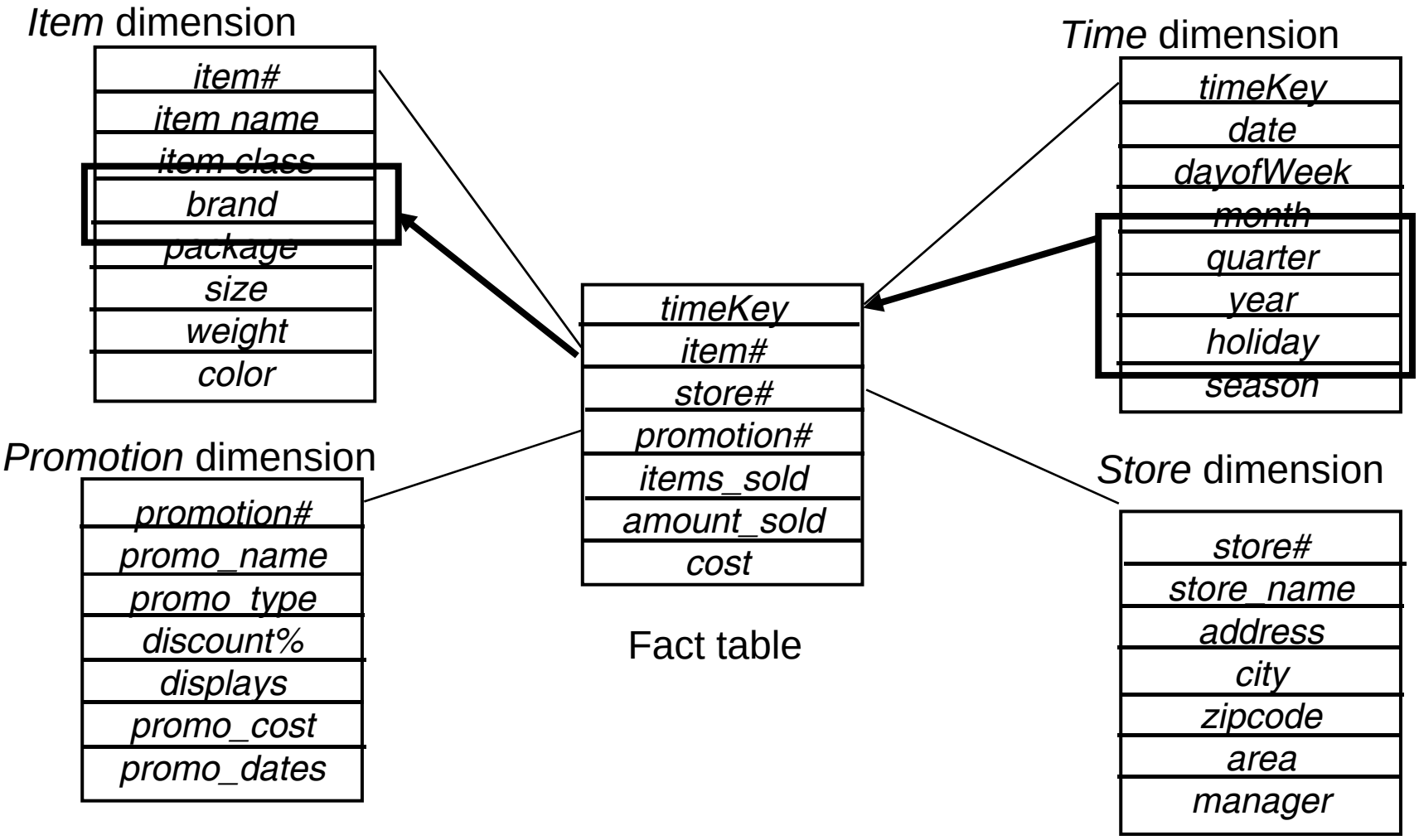


Figure 22.4 Hierarchies on dimensions.

Data Modeling Example

- Requirements
  - A convenience store chain
  - Sale results by date, item, and store
  - Sales by area, month, item or brand
  - Performance of promotions
- Fact table
  - Fact event: *sale transaction*
  - Atomic unit: *item, day, store, …*
  - Fact table attributes: *sales quantity, amount, cost*
- Dimension tables
  - *time*: key, date, month, quarter, year, day-of-week, holiday-flag, season
  - *item*: key, name, category, package-type, size, weight, color
  - *store*: key, name, street, city, zipcode, area, manager
  - *promotion*: key, name, type, discount rate, display-type, cost

Data Modeling – Multi-dim. *Star Schema*



Q: The top selling brands on holidays in Q1?

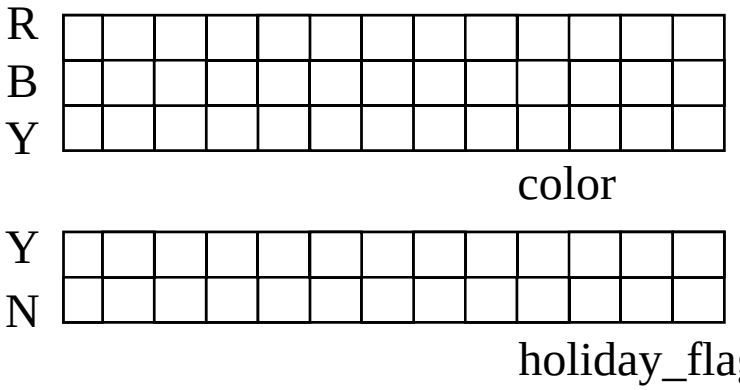
Example Queries

- For the past 3 months, compare holiday and workday sales for each brand
  - Time: restriction (*month*) & grouping (*holiday*)
  - Item: grouping (*brand*)
  - Measure: sum(*amount_sold*)
- Last quarter profitability per promotion type
  - Time: restriction (*quarter*)
  - Promotion : grouping (*promo_type*)
  - Measure : (sum(*amount_sold*)-sum(*cost*))/sum(*promo_cost*)

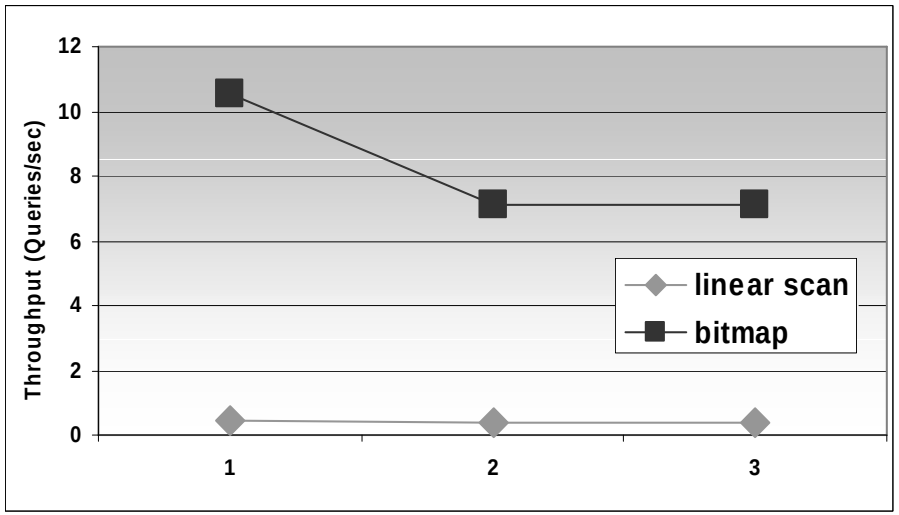
Data Warehouse Tuning

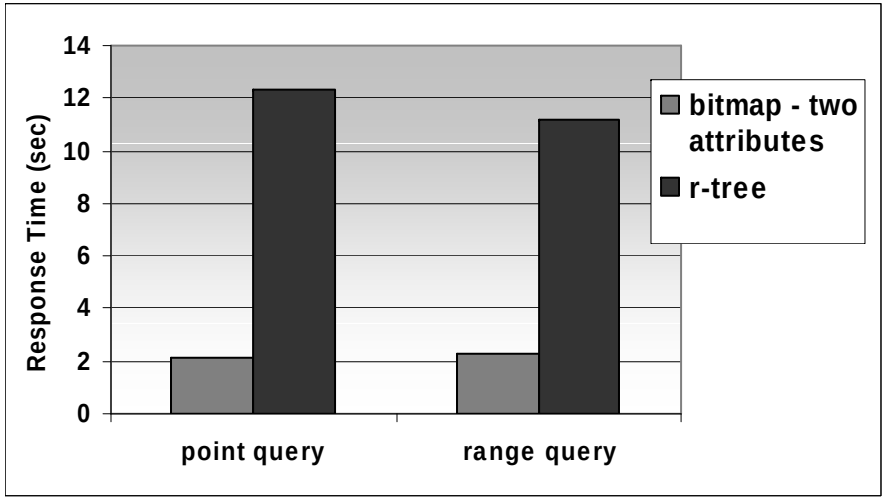
- Characteristics
  - Data Warehouses are bigger – terabytes~petabytes
  - They change less often
  - Queries use aggregation or complex WHERE clauses
- Implications
  - Scanning all data is very very slow
  - We can keep redundant data
  - Index is important
- Tuning techniques
  - Bitmap index
  - Materialized view
  - Approximate answer

Bitmaps

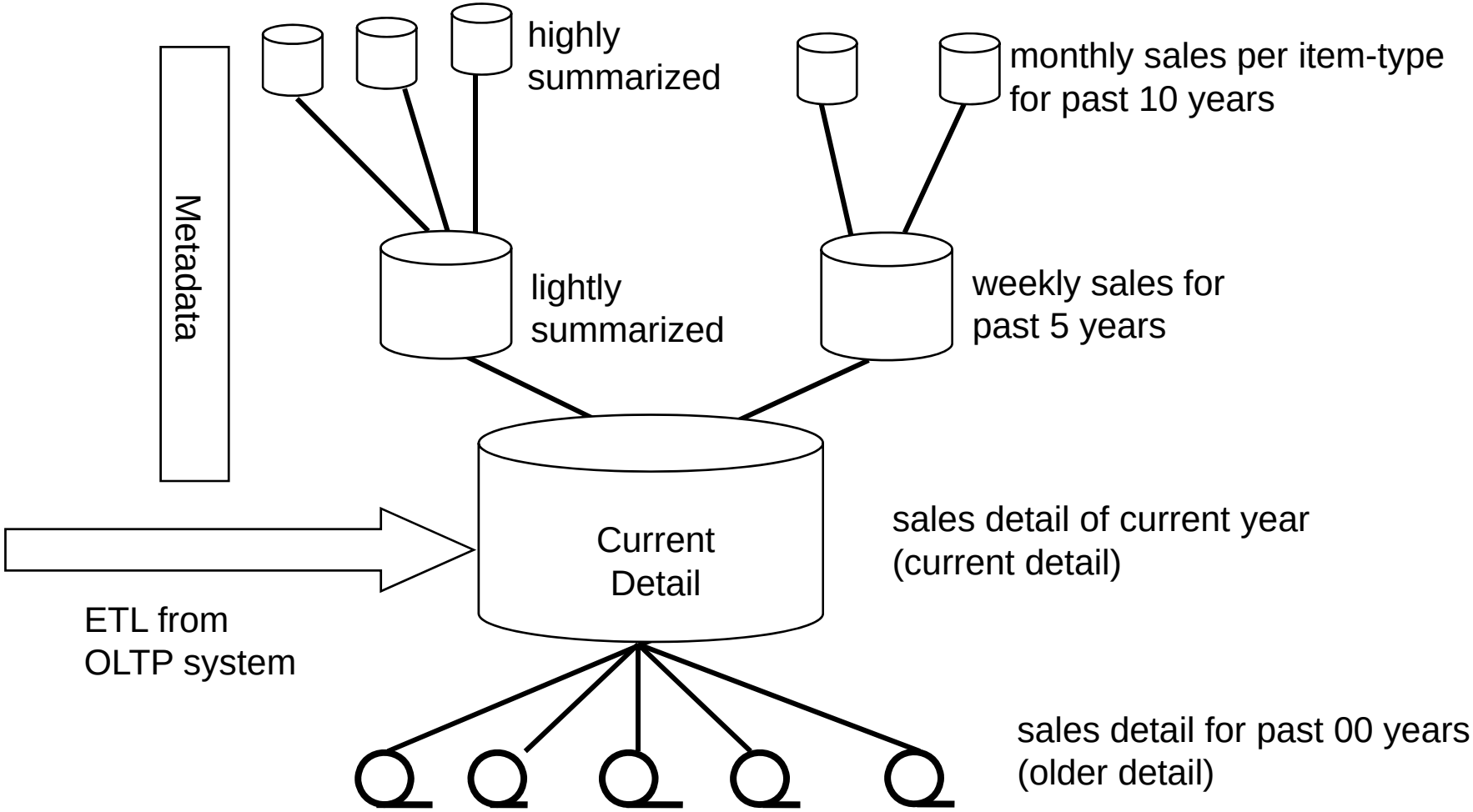


- Order of magnitude improvement compared to scan.
- Bitmaps are best suited for multiple conditions on several attributes, each having a low selectivity.

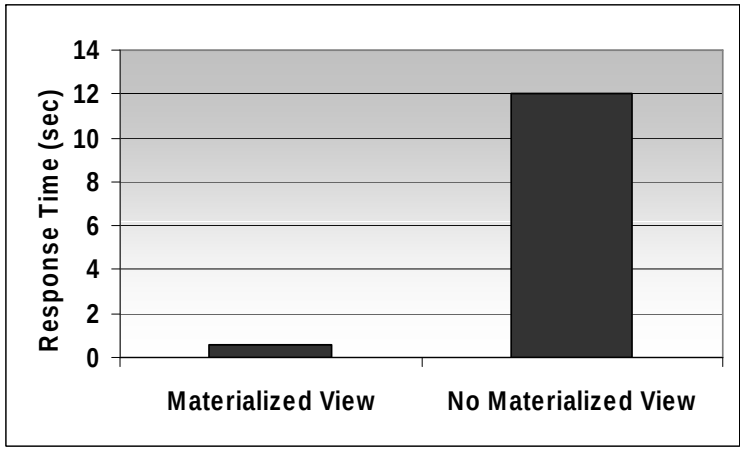


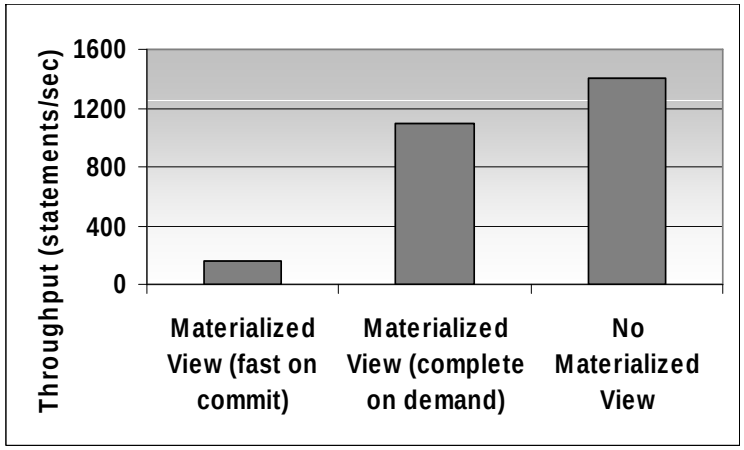


Materialized Views



Materialized Views



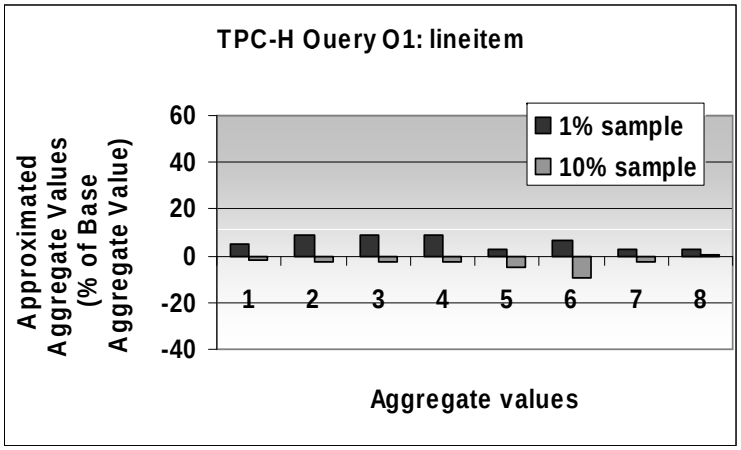


- Graph:
  - Oracle9i on Linux
  - Total sale by vendor is materialized
- Trade-off between query speed-up and view maintenance:
  - The impact of incremental maintenance on performance is significant.
  - Rebuild maintenance achieves a good throughput.
  - A static data warehouse offers a good trade-off.

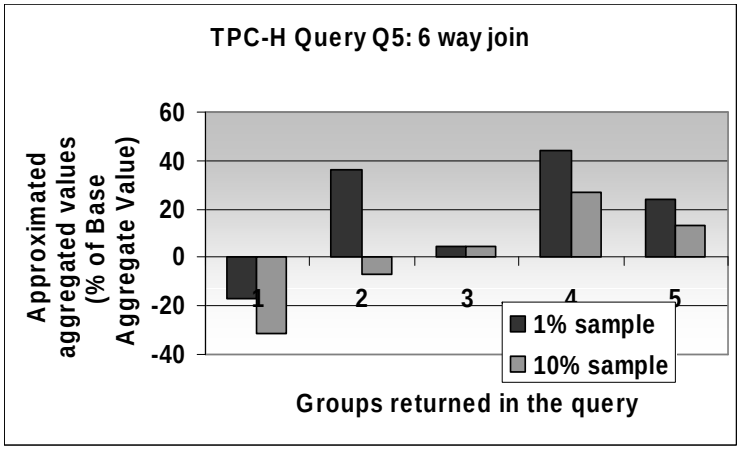
Approximations

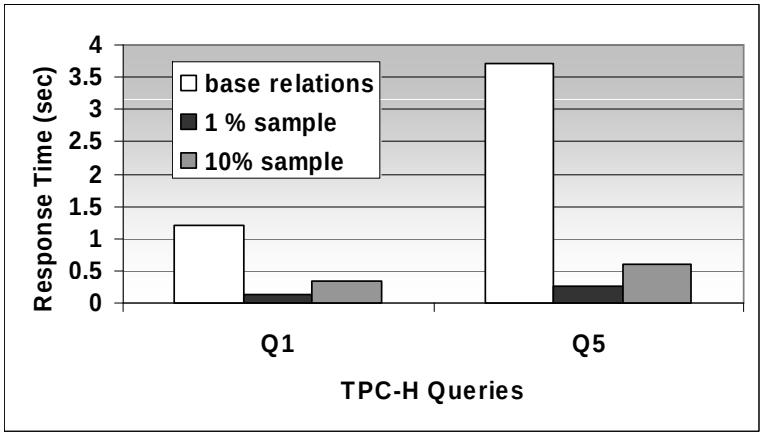
- Ideal application parameters: broad, dynamic, large data
- Rippling
  - When computing average of all salaries
  - If average value changes little with increasing size, then it is probably accurate for all data (within some error bound)
  - Can be extended to multiple joins

Approximations (cont.)



- Good approximation for query Q1 on lineitem
- The aggregated values obtained on a query with a 6-way join are significantly different from the actual values -- for some applications may still be good enough.





Other Issues

- When and how to gather data
  - Source-driven architecture
  - Destination-driven architecture
  - Data warehouses typically have slightly out-of-date data
- What schema to use
  - Usually an integration of multiple sources
  - Must decide on a unified (consolidated) view of the data
- Data cleansing
  - The process of correcting and preprocessing data
  - Merge-purge operation
  - Householding
- How to propagate updates
  - View-maintenance problem
- What data to summarize
  - Maintain only summary data obtained by aggregation on a relation
  - How do we decide on the unit granularity?

Data Mining

- Data mining is the process of semi-automatically analyzing large databases to find useful patterns
- Knowledge Discovery in (Large) Database (KDD)
  - "knowledge extraction", "data dredging", "data archaeology", "data/pattern analysis“, etc.
- **Prediction** based on past history
  - Predict if a credit card applicant poses a good credit risk, based on some attributes (income, job type, age, ..) and past history
  - Predict if a pattern of phone calling card usage is likely to be fraudulent

Data Mining (Cont.)

Some examples of prediction mechanisms:

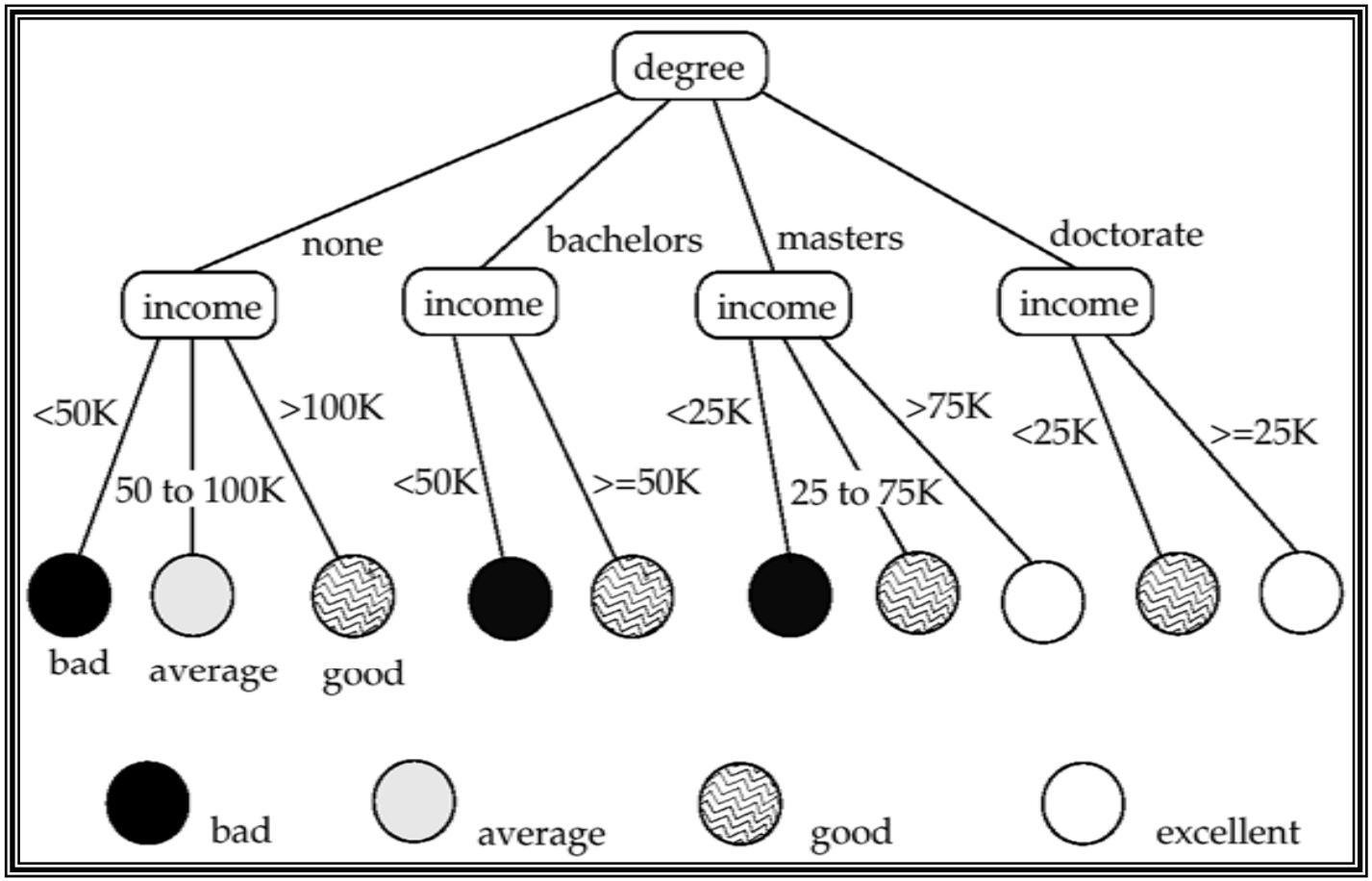
- **Classification**
  - Given a new item whose class is unknown, predict to which class it belongs
- **Regression** formulae
  - Given a set of mappings for an unknown function, predict the function result for a new parameter value
- **Associations**
  - Find books that are often bought by "similar" customers. If a new such customer buys one such book, suggest the others too.
  - Associations may be used as a first step in detecting **causation**
  - E.g. association between exposure to chemical X and cancer,
- **Clusters**
  - E.g. typhoid cases were clustered in an area surrounding a contaminated well
  - Detection of clusters remains important in detecting epidemics

Classification Rules

- Classification rules help assign new objects to classes.
  - E.g., given a new automobile insurance applicant, should he or she be classified as low risk, medium risk or high risk?
- Classification rules for above example could use a variety of data, such as educational level, salary, age, etc.
  - $\forall$ person P, P.degree = masters **and** P.income > 75,000
    $\Rightarrow$ P.credit = excellent
  - $\forall$ person P, P.degree = bachelors **and**
    (P.income $\geq$ 25,000 and P.income $\leq$ 75,000)
    $\Rightarrow$ P.credit = good
- Rules are not necessarily exact: there may be some misclassifications

Decision Tree

- Classification rules can be shown compactly as a decision tree.



Other Types of Classifiers

- Neural net classifiers are studied in artificial intelligence and are not covered here
- Bayesian classifiers use **Bayes theorem**, which says

$$p(c_j \mid d) = \frac{p(d \mid c_j)\, p(c_j)}{p(d)}$$

where

$p(c_j \mid d)$ = probability of instance d being in class c_j,

$p(d \mid c_j)$ = probability of generating instance d given class c_j,

$p(c_j)$ = probability of occurrence of class c_j

$p(d)$ = probability of instance d occurring

Naïve Bayesian Classifiers

- Bayesian classifiers require
  - computation of $p(d \mid c_j)$
  - precomputation of $p(c_j)$
  - $p(d)$ can be ignored since it is the same for all classes
- To simplify the task, assume attributes have independent distributions;

$$p(d \mid c_j) = p(d_1 \mid c_j) * p(d_2 \mid c_j) * \ldots * (p(d_n \mid c_j)$$

  - Each of the $p(d_i \mid c_j)$ can be estimated from a histogram on d_i values for each class c_j
  - the histogram is computed from the training instances

Association Rules

- Examples
  - Someone who buys bread is quite likely also to buy milk
  - A person who bought the book *Database System Concepts* is quite likely also to buy the book *Operating System Concepts*.
- **Association rules:**

  $bread \Rightarrow milk$

  $DB\text{-}Concepts, OS\text{-}Concepts \Rightarrow Networks$
  - Left hand side: antecedent, right hand side: consequent
  - An association rule must have an associated population; the population consists of a set of instances
    - E.g. each transaction (sale) at a shop is an instance, and the set of all transactions is the population

Association Rules (Cont.)

- Rules have an associated support, as well as an associated confidence.
- **Support**
  - P(X ∩ Y): how often does this pattern occur
  - what fraction of the population satisfies both the antecedent and the consequent of the rule
  - E.g. suppose only 0.001 percent of all purchases include milk and screwdrivers. The support for the rule is *milk* ⇒ *screwdrivers* is low.
- **Confidence**
  - P(Y | X): how prevalent is the pattern given X
  - how often the consequent is true when the antecedent is true.
  - E.g. the rule *bread* ⇒ *milk* has a confidence of 80 percent if 80 percent of the purchases that include bread also include milk.

Finding Association Rules

- To discover association rule,
  - $(i_1, i_2, \ldots, i_n) => i_0$
  - Find large itemsets
    - large itemset S: sets of items with sufficient support
  - Find rule $(S - s) \Rightarrow s$ for every subset $s \subset S$ where $(S - s) \Rightarrow s$ has sufficient confidence

+	B1 = {m, c, b}	B2 = {m, p, j}
−	B3 = {m, b}	B4 = {c, j}
−	B5 = {m, p, b}	+ B6 = {m, c, b, j}
	B7 = {c, b, j}	B8 = {b, c}

- An association rule: {m, b} -> c.
  - Confidence = 2/4 = 50%
  - Support = 2/8 = 25%
- A typical question is "find all association rules with support >= *s* and confidence >= *c*."

Finding Association Rules (cont.)

- Sample Database
  1. A supermarket sells 10,000 items
  2. The average basket has 10 items
  3. There are 10,000,000 baskets in the database
- Table structure
  - Baskets_long(BID, item1, item2, …, item10)
  - Baskets(BID, item)
    - 10 X 10,000,000 = 100,000,000 records
- The support threshold is 1% of the baskets
  - i.e., X=>Y is interesting only if X, Y are purchased together at least 1% of the time (100,000 baskets)

Frequent Pairs in SQL

```
SELECT b1.item, b2.item
FROM Baskets b1, Baskets b2
WHERE b1.BID = b2.BID
  AND b1.item < b2.item
GROUP BY b1.item, b2.item
HAVING COUNT(*) >= 100000;
```

Look for two Basket tuples with the same basket and different items. First item must precede second, so we don't count the same pair twice.

Create a group for each pair of items that appears in at least one basket.

Throw away pairs of items that do not appear at least *s* times.

A-Priori Algorithm

- The naïve way (query in previous page)
  - Join complexity: 100,000,000 X 100,000,000
  - Each basket contributes C(10, 2) = 45 pairs
  - Result of join has 45 X 10,000,000 records
  - Need to perform Group by and Having
- The *A-Priori algorithm*
  - Key idea: If item *i* does not appear in *s* baskets, then no pair including *i* can appear in *s* baskets
  - So, do not consider items that appear in less than *s* baskets

A-Priori Algorithm (cont.)

- The new query

```
INSERT INTO Baskets1(basket, item)
SELECT * FROM Baskets
WHERE item IN ( SELECT item FROM Baskets
                GROUP BY item
                HAVING COUNT(*) >= 100000 );
```

- Perform previous query with Baskets1
- Join computation
  - Computing Baskets1 is cheap since there is no join
    - Or could have built index from beginning
  - At most 1,000 items can be frequent
  - In practice Baskets1 is less than ¼ of Baskets
    - Join complexity is *N* square

Other Types of Mining

- **Text mining**: application of data mining to textual documents
  - cluster Web pages to find related pages
  - cluster pages a user has visited to organize their visit history
  - classify Web pages automatically into a Web directory
- **Data visualization** systems help users examine large volumes of data and detect patterns visually
  - Can visually encode large amounts of information on a single screen
  - Humans are very good a detecting visual patterns

END OF CHAPTER 18