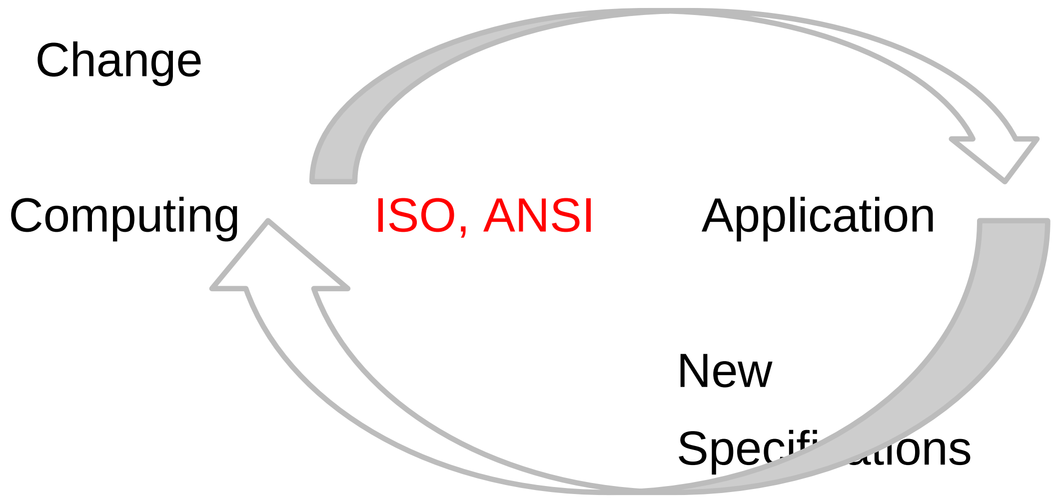
DataWare House
Data Mining
(Tools and techniqus in e-gov)

e-governance (2024)

S. Bhalla
Dept. of CSE,
IITD

Computing as enterprise
(25 Years before 1993)



- **Prior to 1993**:
- Database System
- Distributed Systems on ETHERNET (web ?, internet ?)
  – Banking, Stock exchange, Airlines, Railways,…

Using Data Analysis and Mining

The following techniques have been evolved by business world [Also available for e-governance]

Decision Support Systems

- Data Analysis and OLAP
- Data Warehousing
- Data Mining

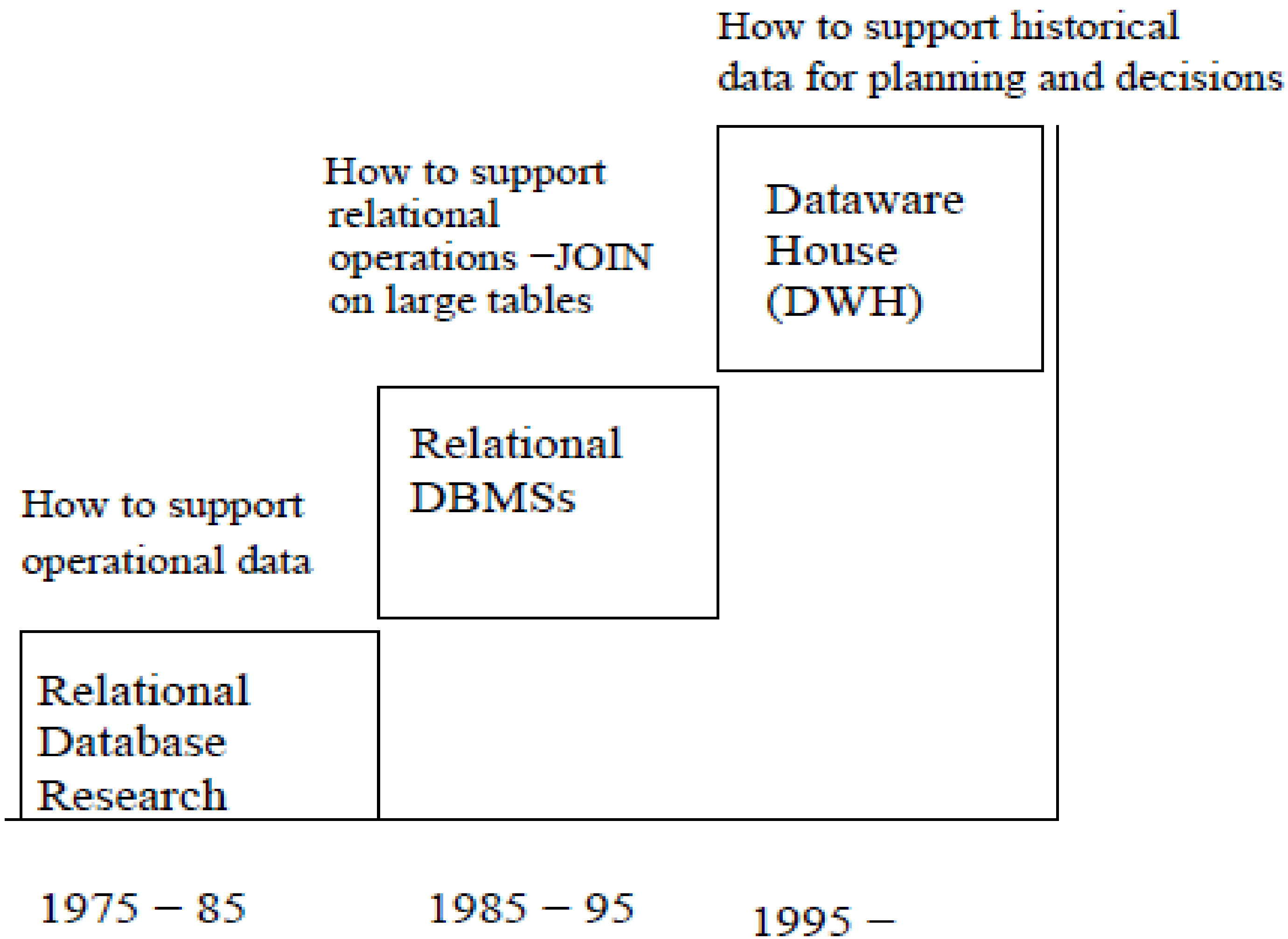
Decision Support Systems (DSSs)

- DSSs are used to make business decisions. Decisions are based on data collected by on-line transaction-processing systems.
- Examples of 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.)

Decision-Support Systems in e-gov

- **Data analysis** use specialized tools and SQL extensions
  - Example tasks
    - For each product category and each region, what were the total sales in the last quarter and how do they compare with the same quarter last year
    - As above, for each product category and each customer category
- **Statistical analysis** packages (e.g., : S++) interfaced with databases
  - Statistical analysis is a large field
- **Data mining** seeks to discover knowledge automatically in the form of statistical rules and patterns from large databases.
- A **data warehouse** archives information gathered from multiple sources, and stores it under a unified schema, at a single site.
  - Important for large businesses that generate data from multiple divisions, possibly at multiple sites

Development of Data warehouse



Difference between Database

- Database → Operational Data

- Data warehouse (DWH) → Cummulative data (over space , time and data for other factors)

Data ←→ Set Theoretic View

- Set X - { a, b, c } → a, b, c.
- Fruits - {Banana, Apple} ; Color - {Red, Green}

Banana	Green
Apple	Red
Apple	Green

- Table → is a proper subset of set 1 X Set 2 (RELATION)
- Set Theoretc tools ; Union, Intersect , Minus, X , rename, = , ...
- SQL → Structured Query Language (Relational Databases)

Example Database

[I.A] Suppliers-Parts-Projects (Jobs) Database

Suppliers (S)

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

Parts (P)

P#	PNAME	COLOR	WEIGHT	CITY
P1	Nut	Red	12.0	London
P2	Bolt	Green	17.0	Paris
P3	Screw	Blue	17.0	Rome
P4	Screw	Red	14.0	London
P5	Cam	Blue	12.0	Paris
P6	Cog	Red	19.0	London

Projects (J)

J#	JNAME	CITY
J1	Sorter	Paris
J2	Display	Rome
J3	OCR	Athens
J4	Console	Athens
J5	RAID	London
J6	EDS	Oslo
J7	Tape	London

(SPJ)

S#	P#	J#	QTY
S1	P1	J1	200
S1	P1	J4	700
S2	P3	J1	400
S2	P3	J2	200
S2	P3	J3	200
S2	P3	J4	500
S2	P3	J5	600
S2	P3	J6	400
S2	P3	J7	800
S2	P5	J2	100
S3	P3	J1	200
S3	P4	J2	500
S4	P6	J3	300
S4	P6	J7	300
S5	P2	J2	200
S5	P2	J4	100
S5	P5	J5	500
S5	P5	J7	100
S5	P6	J2	200
S5	P1	J4	100
S5	P3	J4	200
S5	P4	J4	800
S5	P5	J4	400
S5	P6	J4	500

Set Theretic Tools → SQL in RDBMSs

With reference to relational Database Management System (RDBMS) : (a)

Select S#

From S {choose table S, act on where clause}

Where S# = "S3" ; {S3, Blake, 30. Paris}

Find Parts Supplied by S3

Select P#

From S, SPJ {choose tables S and SPJ}

Where S# = S3 and S.S# = SPJ.S# ; { P3, P4 }

Set Theretic Tools → SQL in RDBMSs

Relational Database Management System (RDBMS) :

And

SQL

→ Can answer all questions

Difference between Database

- Database → Operational Data

- Data warehouse (DWH) → Cummulative data (over space , time and data for other factors)

Data warehouse (DWH) as a Database

[I.A] Suppliers-Parts-Projects (Jobs) Database

locid	city	state	country
1	Ames	Iowa	USA
2	Chennai	TN	India
5	Temple	Arizona	USA

Locations

pid	pname	category	price
11	Lee Leans	Apparel	25
12	Zord	Toys	18
13	Biro Pen	Stationery	2

Products

pid	timeid	locid	sales
11	1	1	25
11	2	1	8
11	3	1	15
12	1	1	30
12	2	1	20
12	3	1	50
13	1	1	8
13	2	1	10
13	3	1	10
11	1	2	35
11	2	2	22
11	3	2	10
12	1	2	26
12	2	2	45
12	3	2	20
13	1	2	20
13	2	2	40
13	3	2	5

Sales

View → aggregate values over entities

[I] Dimension Heirarchies [31]

- Locations(*locid*: integer, *city*: string, *country*: string)
- Products(*pid*: integer, *pname*: string, *catagory*: string, *price*: real)
- Times(*timeid*: integer, *date*: string, *week*: integer, *month*: integer, *quarter*: integer, *year*: integer, *holiday_flag*: boolean)

```
                          Times                 Locations
  Products               year
                          |
                          |
                       quarter                country
  catagory             /     \                  |
     |              week     month            state
   pname               \     /                  |
     |                  date                   city
    pid                  |                      |
                       timeid                 locid
```

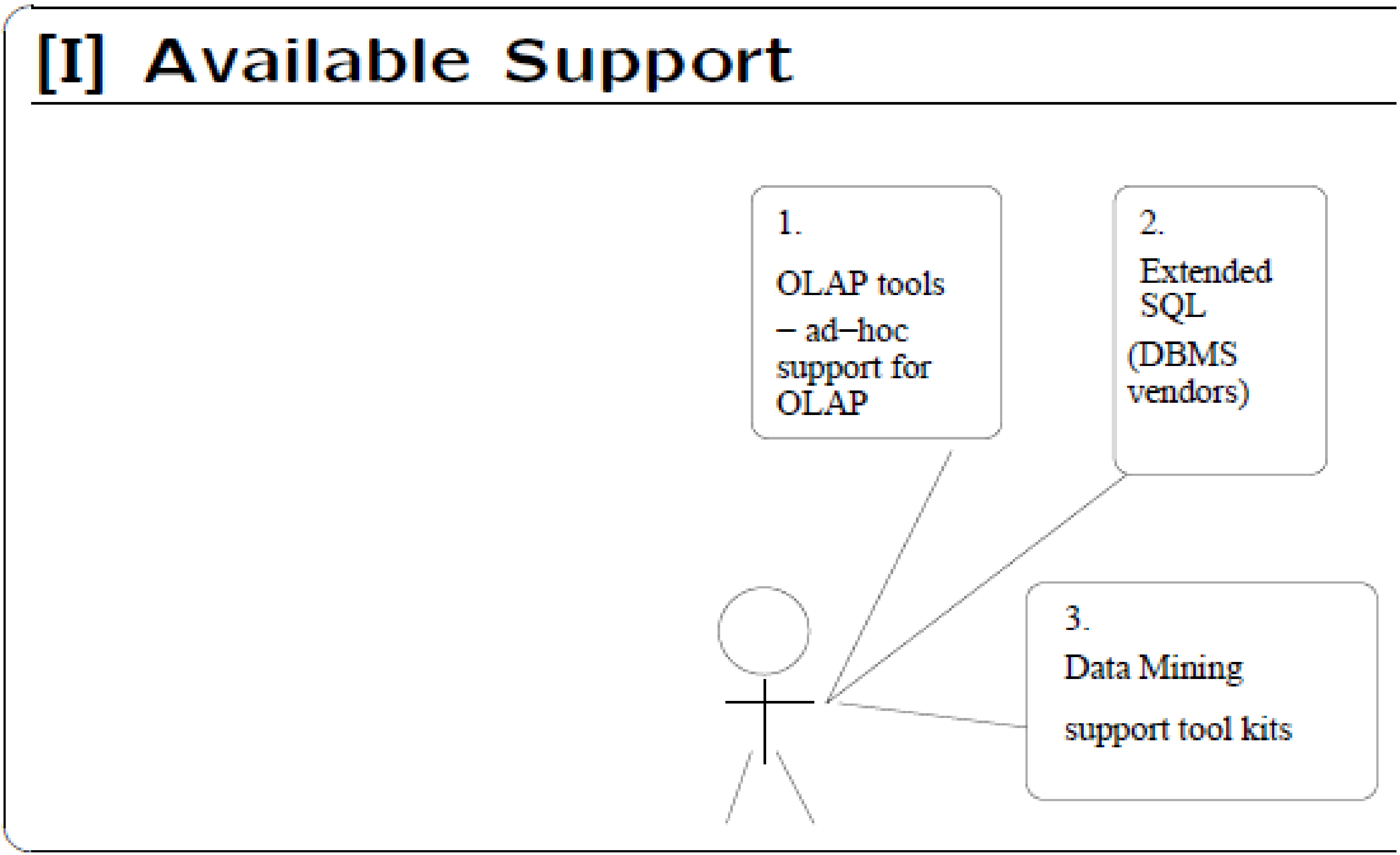
Using aggregates

[I] Dimension Heirarchies [32]

- 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
- Crosstabs can be easily extended to deal with hierarchies

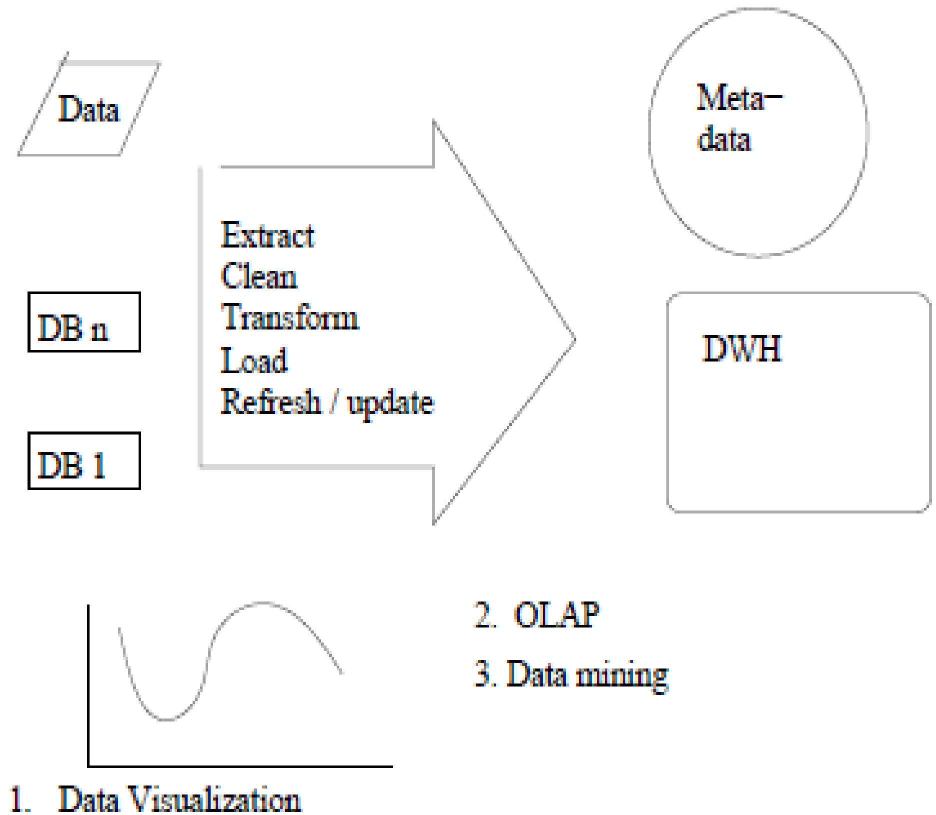
- Can drill down or roll up on a hierarchy

Visualization Tools



DWH Architecture

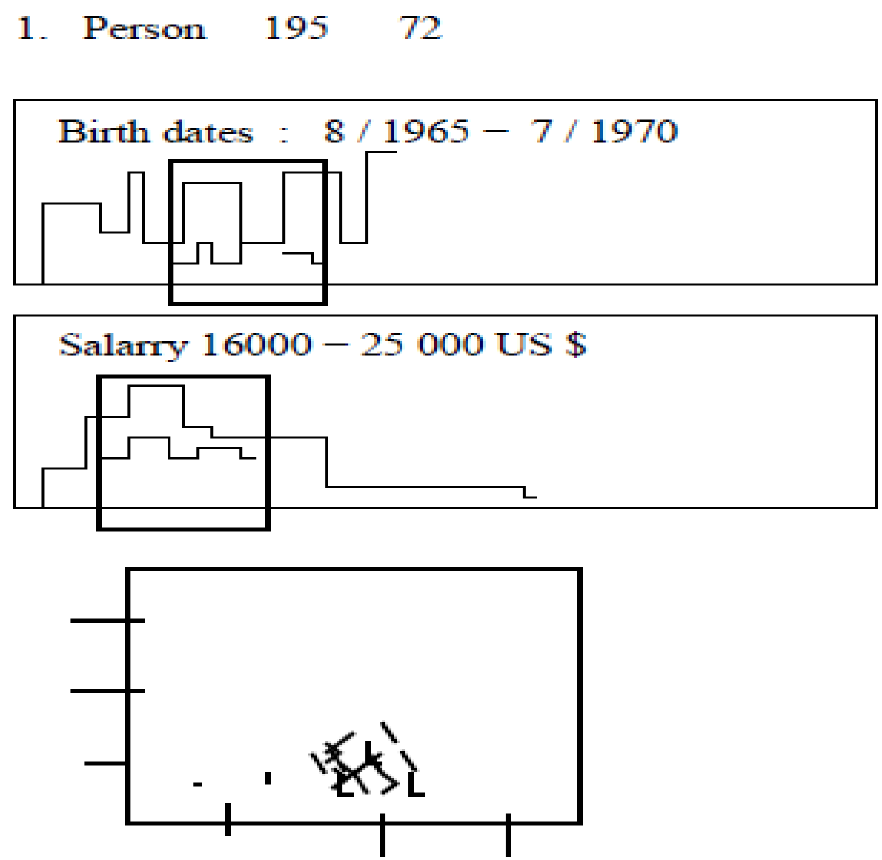
[I] DWH Architecture



[I] Visualization

- Dynamic query :

range 1. - 195 persons , range 2. - 72 persons



Data Analysis and OLAP

- **Online Analytical Processing (OLAP)**
  - Interactive analysis of data, allowing data to be summarized and viewed in different ways (with no delay)

- Data that can be modeled as dimension attributes and measure attributes are called **multidimensional data**.
  - **Measure attributes**
    - measure some value
    - can be aggregated upon
  - **Dimension attributes**
    - define the dimensions on which measure attributes (or aggregates thereof) are viewed
    - e.g. the attributes *item_name, color,* and *size* of the *sales* relation

Materialized Views in e-Gov
Cross Tabulation of *sales* by *item-name* and *color*

size: **all**

color

item-name	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

- The table is a **cross-tabulation** (**cross-tab**), or 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.

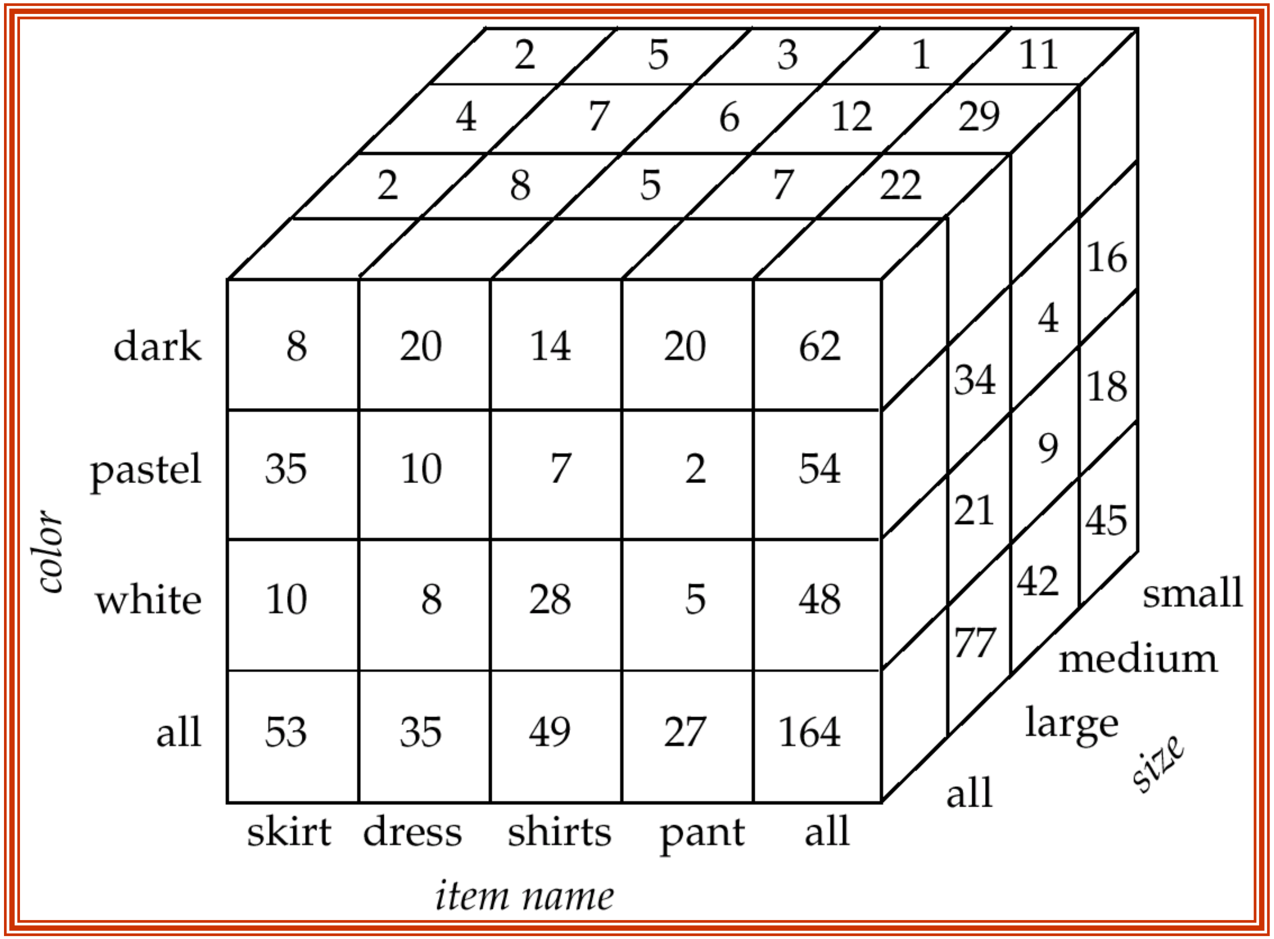
View → Relational Representation

- 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

Storage view → 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

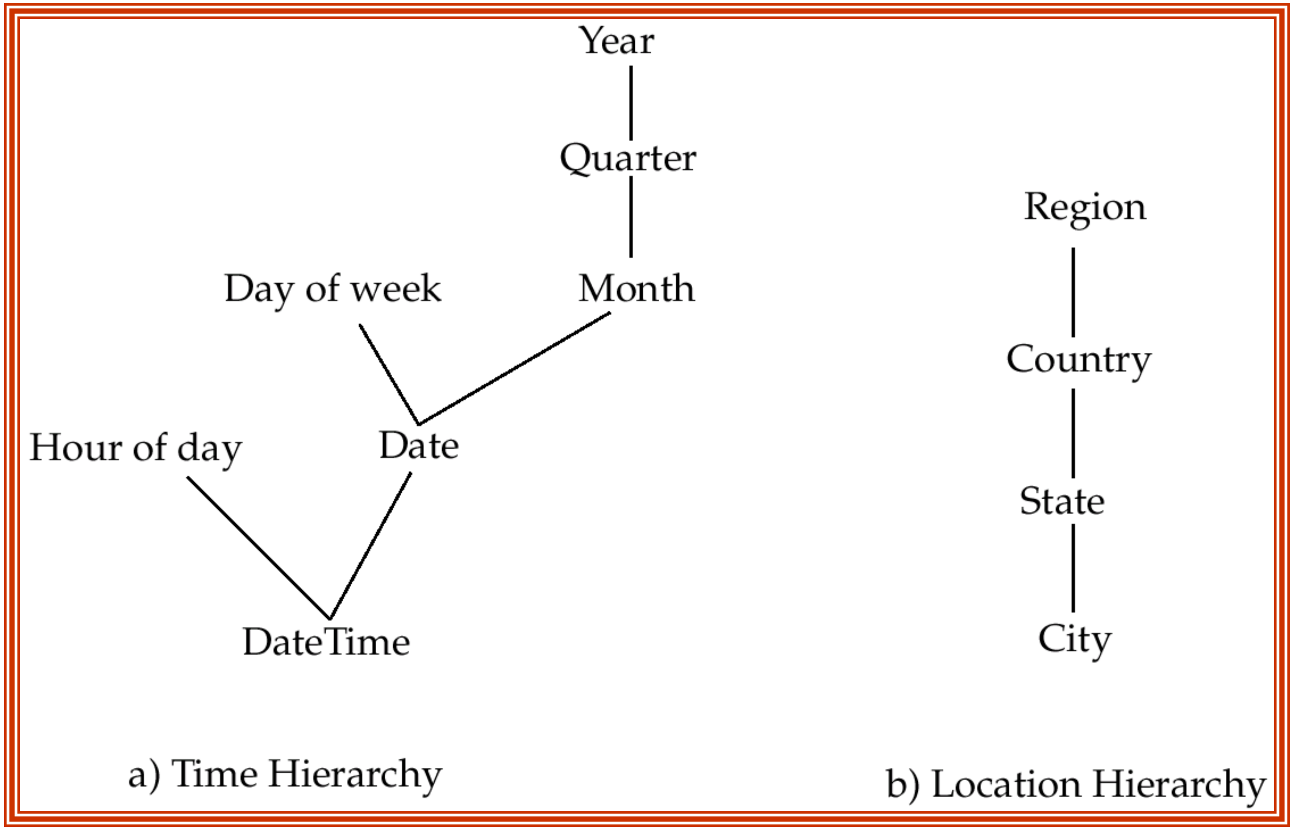


e-gov → Online Analytical Processing

- **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



View → 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

- The earliest OLAP systems used multidimensional arrays in memory to store data cubes, and are referred to as **multidimensional OLAP (MOLAP)** systems.
- OLAP implementations using only relational database features are called **relational OLAP (ROLAP)** systems. SQL query language support exists.
- Hybrid systems, which store some summaries in memory and store the base data and other summaries in a relational database, are called **hybrid OLAP (HOLAP)** systems.

OLAP Implementation (Cont.)

- Early OLAP systems precomputed *all* possible aggregates in order to provide online response
  - Space and time requirements for doing so can be very high
    - 2^n combinations of different type of grouping (**group by)**
  - It suffices to 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*)
      - For all but a few "non-decomposable" aggregates such as *median*
      - is cheaper than computing it from scratch
- 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*)

  This 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 that we saw earlier, but with *null* in place of **all**, can be computed by

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

- The function **grouping()** can be applied on an attribute
  - Returns 1 if the value is a null value representing all, and returns 0 in all other cases.

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

- Can use the function **decode()** in the **select** clause to replace such nulls by a value such as **all**
  - E.g. replace *item-name* in first query by

    decode(**grouping**(item-*name*), 1, 'all', *item-name*)

Extended Aggregation (Cont.)

- The **rollup** construct generates union on every prefix of specified list of attributes
- E.g.

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

  Generates union of four groupings:

  { (*item-name, color, size*), (*item-name, color*), (*item-name*), ( ) }
- Rollup can be used to generate aggregates at multiple levels of a hierarchy.
- E.g., suppose table *itemcategory*(*item-name, category*) gives the category of each item. Then

  select *category, item-name*, **sum**(*number*)
  from *sales, itemcategory*
  where *sales.item-name = itemcategory.item-name*
  group by rollup(*category, item-name*)

  would give a hierarchical summary by *item-name* and by *category.*

Extended Aggregation (Cont.)

- Multiple rollups and cubes can be used in a single group by clause
  - Each generates set of group by lists, cross product of sets gives overall set of group by lists
- E.g.,

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

  generates the groupings

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

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

Ranking

- Ranking is done in conjunction with an order by specification.
- Given a relation student-marks(student-id, marks) find the rank of each student.

  select *student-id*, **rank**( ) **over** (**order by** *marks* **desc) as** *s-rank*
  from *student-marks*

- An extra **order by** clause is needed to get them in sorted order

  select *student-id*, **rank** ( ) **over** (**order by** *marks* **desc**) **as** *s-rank*
  from *student-marks*
  order by *s-rank*

- Ranking may leave gaps: e.g. if 2 students have the same top mark, both have rank 1, and the next rank is 3
  - **dense_rank** does not leave gaps, so next dense rank would be 2

Ranking (Cont.)

- Ranking can be done within partition of the data.
- "Find the rank of students within each section."

  select *student-id, section*,
  rank ( ) **over** (**partition by** *section* **order by** *marks* **desc**)
  as *sec-rank*
  from *student-marks, student-section*
  where *student-marks.student-id = student-section.student-id*
  order by *section, sec-rank*
- Multiple **rank** clauses can occur in a single **select** clause
- Ranking is done *after* applying **group by** clause/aggregation

Ranking (Cont.)

- Other ranking functions:
  - **percent_rank** (within partition, if partitioning is done)
  - **cume_dist** (cumulative distribution)
    - fraction of tuples with preceding values
  - **row_number** (non-deterministic in presence of duplicates)
- SQL:1999 permits the user to specify **nulls first** or **nulls last**

select *student-id*,
    rank ( ) **over** (**order by** *marks* **desc nulls last**) **as** *s-rank*
from *student-marks*

Ranking (Cont.)

- For a given constant *n*, the ranking the function *ntile*(*n*) takes the tuples in each partition in the specified order, and divides them into *n* buckets with equal numbers of tuples.
- E.g.:

select *threetile*, **sum**(*salary*)
from (
        select *salary*, **ntile**(3) **over** (**order by** *salary*) **as** *threetile*
        from *employee*) **as** *s*
group by *threetile*

Windowing

- Used to smooth out random variations.
- E.g.: moving average: “Given sales values for each date, calculate for each date the average of the sales on that day, the previous day, and the next day”
- **Window specification** in SQL:
  - Given relation *sales(date, value)*

    select *date*, ***sum***(*value*) **over**
    (**order by** *date* **between rows** 1 **preceding and** 1 **following**)
    from *sales*
- Examples of other window specifications:
  - **between rows unbounded preceding and current**
  - **rows unbounded preceding**
  - **range between** 10 **preceding and current row**
    - All rows with values between current row value –10 to current value
  - **range interval** 10 **day preceding**
    - Not including current row

Windowing (Cont.)

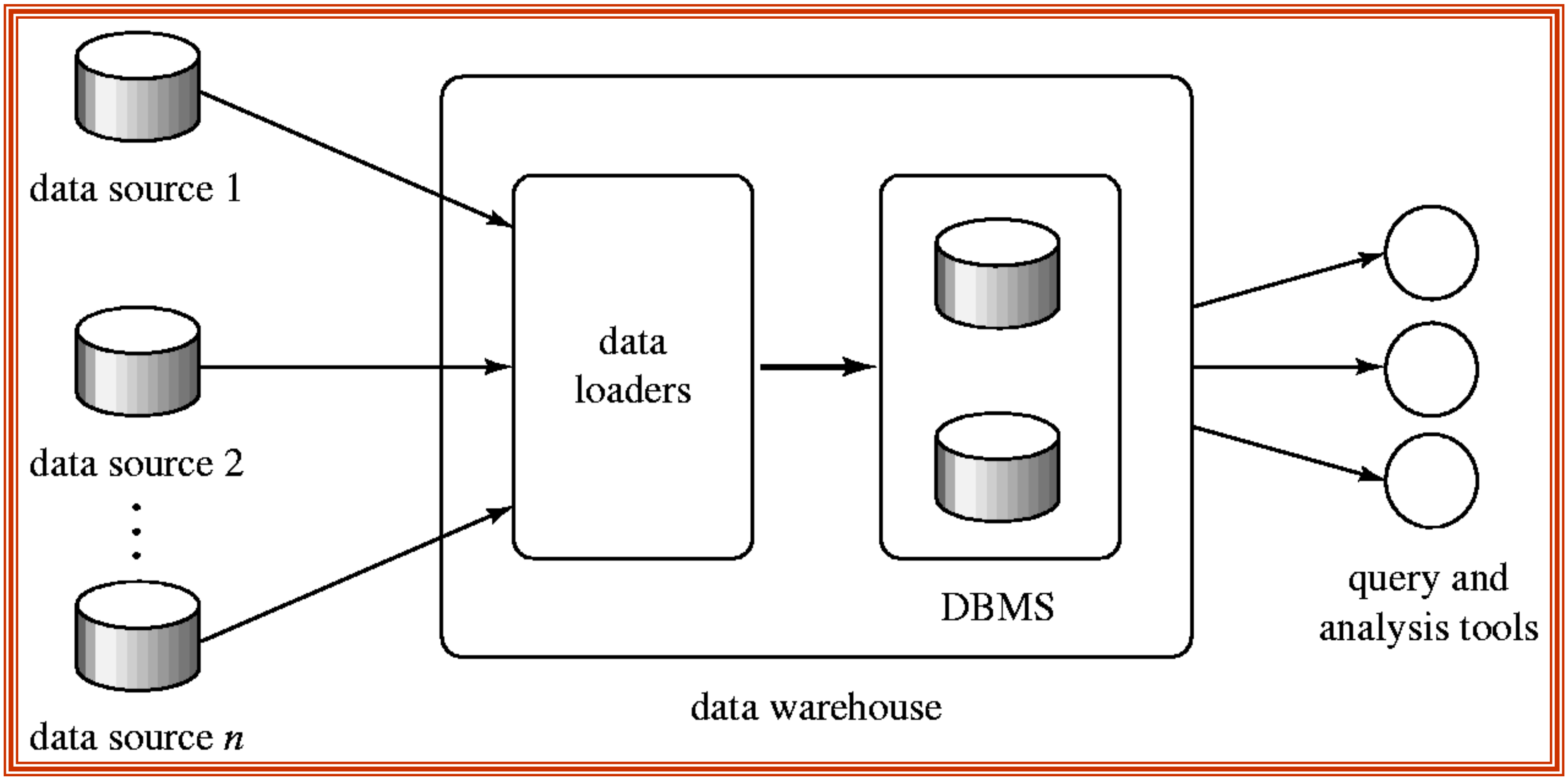
- Can do windowing within partitions
- E.g. Given a relation *transaction* (*account-number, date-time, value*), where value is positive for a deposit and negative for a withdrawal
  - "Find total balance of each account after each transaction on the account"

select *account-number, date-time*,
    sum (*value*) **over**
            (**partition by** *account-number*
            order by *date-time*
            rows unbounded preceding)
   as *balance*
from *transaction*
order by *account-number, date-time*

Data Warehousing

- Data sources often store only current data, not historical data
- Corporate decision making requires a unified view of all organizational data, including historical data
- A **data warehouse** is a repository (archive) of information gathered from multiple sources, stored under a unified schema, at a single site
  - Greatly simplifies querying, permits study of historical trends
  - Shifts decision support query load away from transaction processing systems

Data Warehousing



Design Issues

- *When and how to gather data*
  - Source driven architecture: data sources transmit new information to warehouse, either continuously or periodically (e.g. at night)
  - Destination driven architecture: warehouse periodically requests new information from data sources
  - Keeping warehouse exactly synchronized with data sources (e.g. using two-phase commit) is too expensive
    - Usually OK to have slightly out-of-date data at warehouse
    - Data/updates are periodically downloaded form online transaction processing (OLTP) systems.
- *What schema to use*
  - Schema integration

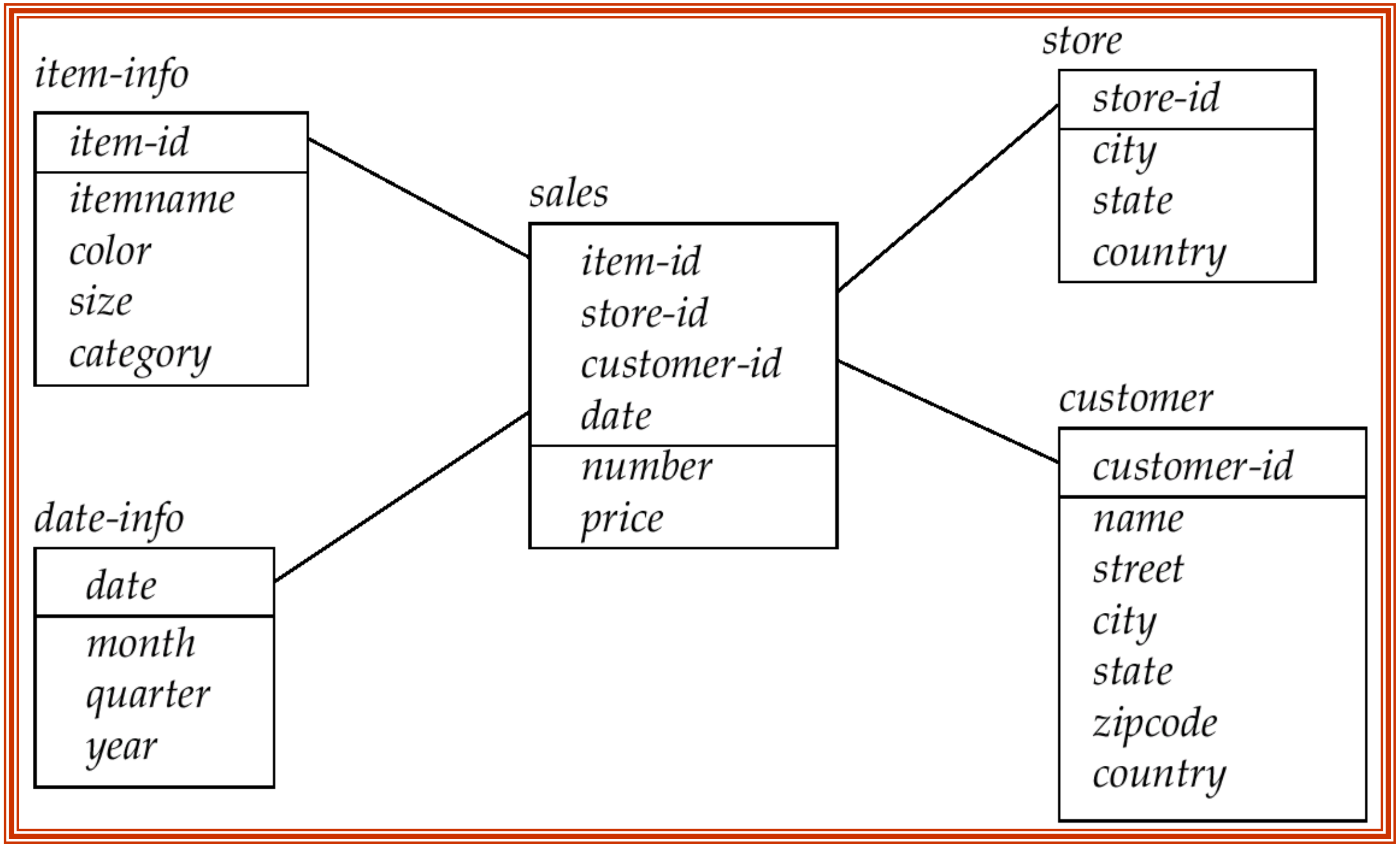
More Warehouse Design Issues

- *Data cleansing*
  - E.g. correct mistakes in addresses (misspellings, zip code errors)
  - Merge address lists from different sources and purge duplicates
- *How to propagate updates*
  - Warehouse schema may be a (materialized) view of schema from data sources
- *What data to summarize*
  - Raw data may be too large to store on-line
  - Aggregate values (totals/subtotals) often suffice
  - Queries on raw data can often be transformed by query optimizer to use aggregate values

Warehouse Schemas

- Dimension values are usually encoded using small integers and mapped to full values via dimension tables
- Resultant schema is called a **star schema**
  - More complicated schema structures
    - Snowflake schema: multiple levels of dimension tables
    - Constellation: multiple fact tables

Data Warehouse Schema



E-gov thru Data Mining

- A) Querying thru DBMS tool Box for DWH
- B) Data Mining
- C) Specialized Processing

Data Mining

- Data mining is the process of semi-automatically analyzing large databases to find useful patterns
- **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
- 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

Data Mining (Cont.)

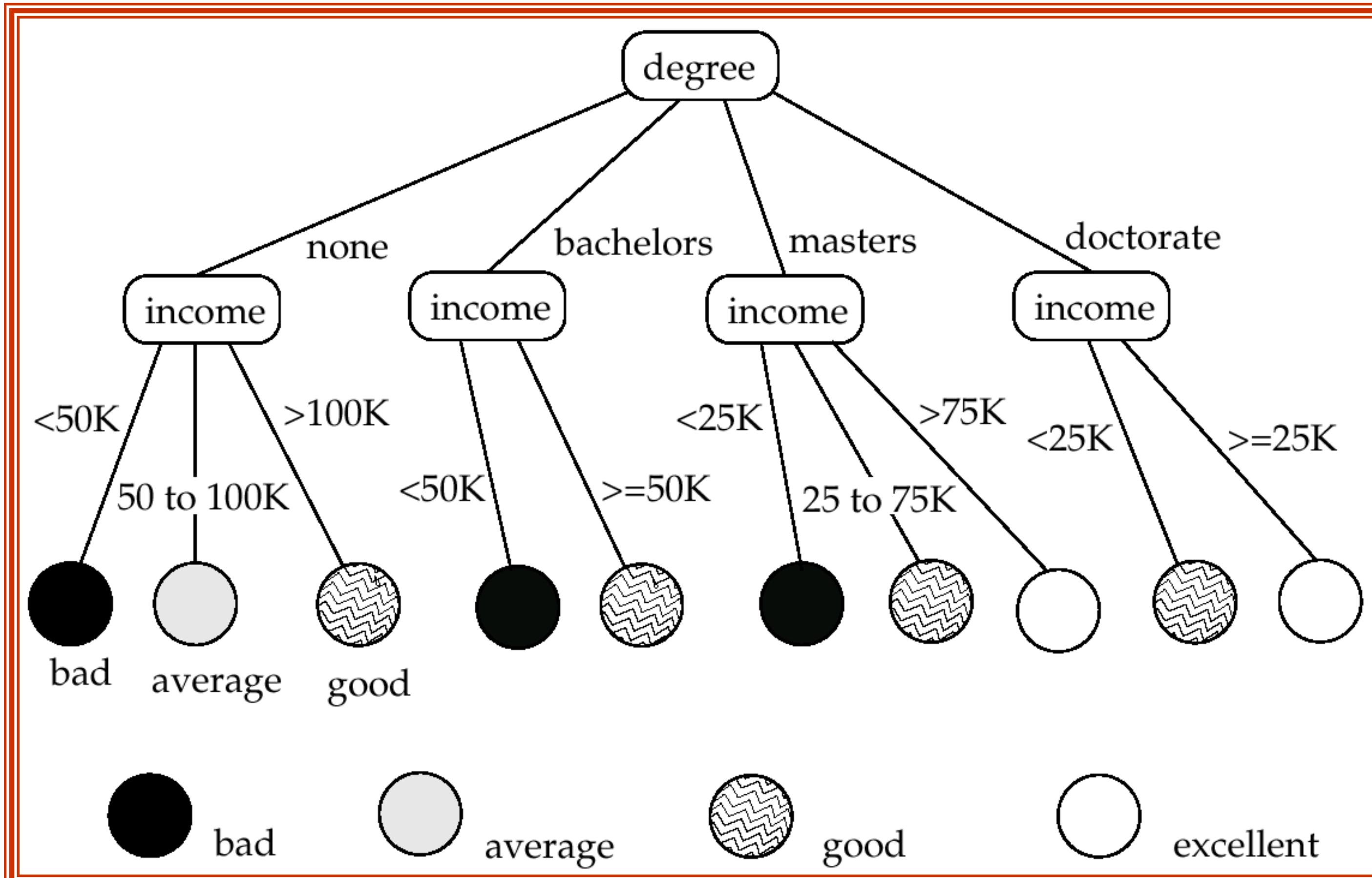
- **Descriptive Patterns**
  - **Associations**
    - Find books that are often bought by “similar” customers. If a new such customer buys one such book, suggest the others too.
  - Associations may be used as a first step in detecting **causation**
    - E.g. association between exposure to chemical 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
- Classification rules can be shown compactly as a decision tree.

Decision Tree



Construction of Decision Trees

- **Training set**: a data sample in which the classification is already known.
- **Greedy** top down generation of decision trees.
  - Each internal node of the tree partitions the data into groups based on a **partitioning attribute**, and a **partitioning condition** for the node
  - **Leaf** node:
    - all (or most) of the items at the node belong to the same class, or
    - all attributes have been considered, and no further partitioning is possible.

Regression

- Regression deals with the prediction of a value, rather than a class.
  - Given values for a set of variables, X_1, X_2, …, X_n, we wish to predict the value of a variable Y.
- One way is to infer coefficients a_0, a_1, a_1, …, a_n such that
  $$Y = a_0 + a_1 * X_1 + a_2 * X_2 + \ldots + a_n * X_n$$
- Finding such a linear polynomial is called **linear regression**.
  - In general, the process of finding a curve that fits the data is also called **curve fitting**.
- The fit may only be approximate
  - because of noise in the data, or
  - because the relationship is not exactly a polynomial
- Regression aims to find coefficients that give the best possible fit.

Association Rules

- Retail shops are often interested in associations between different items that people buy.
  - 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*.
- Associations information can be used in several ways.
  - E.g. when a customer buys a particular book, an online shop may suggest associated books.
- **Association rules**:

  bread $\Rightarrow$ *milk*   *DB-Concepts, OS-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** is a measure of 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* $\Rightarrow$ *screwdrivers* is low.
- **Confidence** is a measure of how often the consequent is true when the antecedent is true.
  - E.g. the rule *bread* $\Rightarrow$ *milk* has a confidence of 80 percent if 80 percent of the purchases that include bread also include milk.

Finding Association Rules

- We are generally only interested in association rules with reasonably high support (e.g. support of 2% or greater)
- Naïve algorithm
  1. Consider all possible sets of relevant items.
  2. For each set find its support (i.e. count how many transactions purchase all items in the set).
     - **Large itemsets**: sets with sufficiently high support
  3. Use large itemsets to generate association rules.
     1. From itemset A generate the rule $A - \{b\} \Rightarrow b$ for each $b \in A$.
        - Support of rule = support (A).
        - Confidence of rule = support (A) / support ($A - \{b\}$)

Finding Support

- Determine support of itemsets via a single pass on set of transactions
  - Large itemsets: sets with a high count at the end of the pass
- If memory not enough to hold all counts for all itemsets use multiple passes, considering only some itemsets in each pass.
- Optimization: Once an itemset is eliminated because its count (support) is too small none of its supersets needs to be considered.
- The **a priori** technique to find large itemsets:
  - Pass 1: count support of all sets with just 1 item. Eliminate those items with low support
  - Pass *i*: candidates: every set of *i* items such that all its *i-1* item subsets are large
    - Count support of all candidates
    - Stop if there are no candidates

Other Types of Associations

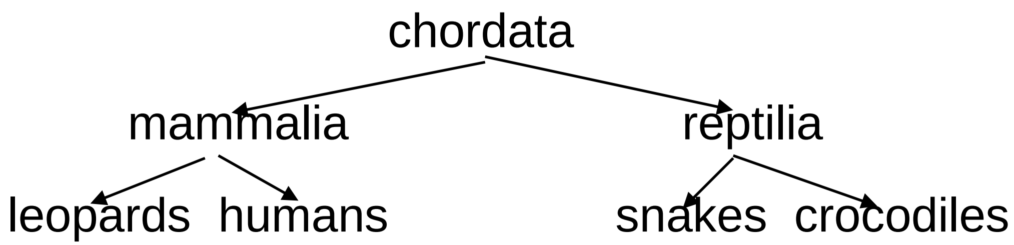
- Basic association rules have several limitations
- Deviations from the expected probability are more interesting
  - E.g. if many people purchase bread, and many people purchase cereal, quite a few would be expected to purchase both
  - We are interested in positive as well as negative correlations between sets of items
    - Positive correlation: co-occurrence is higher than predicted
    - Negative correlation: co-occurrence is lower than predicted
- Sequence associations / correlations
  - E.g. whenever bonds go up, stock prices go down in 2 days
- Deviations from temporal patterns
  - E.g. deviation from a steady growth
  - E.g. sales of winter wear go down in summer
    - Not surprising, part of a known pattern.
    - Look for deviation from value predicted using past patterns

Clustering

- Clustering: Intuitively, finding clusters of points in the given data such that similar points lie in the same cluster
- Can be formalized using distance metrics in several ways
  - Group points into *k* sets (for a given *k*) such that the average distance of points from the centroid of their assigned group is minimized
    - Centroid: point defined by taking average of coordinates in each dimension.
  - Another metric: minimize average distance between every pair of points in a cluster
- Has been studied extensively in statistics, but on small data sets
  - Data mining systems aim at clustering techniques that can handle very large data sets
  - E.g. the Birch clustering algorithm (more shortly)

Hierarchical Clustering

- Example from biological classification
  - (the word classification here does not mean a prediction mechanism)



- Other examples: Internet directory systems (e.g. Yahoo, lookup this topic)
- Agglomerative clustering algorithms
  - Build small clusters, then cluster small clusters into bigger clusters, and so on
- Divisive clustering algorithms
  - Start with all items in a single cluster, repeatedly refine (break) clusters into smaller ones

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

Summary

- 1. e-gov requires

→ Decision Support activity →

→ Data Ware House (DWH)

- 2. SQL modifications for DWH
- 3. Data Mining
- (a) Classification
- (b) Regression
- (c) Association
- (d) Cluster
- 4. Specialized Software systems

Reading Assignment 2

- READING: DWH → Book "Database Systems Concepts, by Silberschatz,
- Paper 3:

Big-Data Applications in the Government Sector

CACM, March 2014, pp. 78-85

E-Gov → In the same way business use big data to pursue profits, governments use it to promote public good

Based on reading (paper 3) Prepare a report on big Data Analytics in e-gov. Finallyalso prepare a "Summary and conclusions".