TIES443

Lecture 3

Data Warehousing

Mykola Pechenizkiy

Course webpage: *http://www.cs.jyu.fi/~mpechen/TIES443*

November 3, 2006

Department of Mathematical Information Technology
University of Jyväskylä

Topics for today

- What is a data warehouse?
- Data warehouse architectures
  - Conceptual DW Modelling
  - Physical DW Modelling
- A multi-dimensional data model
  - Data Cubes
- OLAP
  - 12 Codd's rules for OLAP
  - Main OLAP operations
    - New buzzwords
- Data warehouse implementation and maintenance

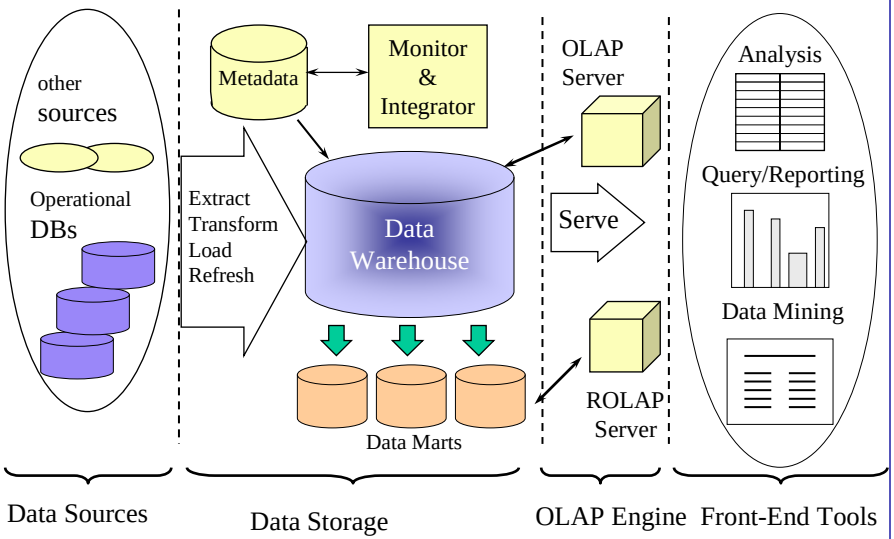
Data Warehouse

A decision support DB that is maintained separately from the organization's operational databases.

Why Separate Data Warehouse?

- High performance for both systems
  - DBMS— tuned for OLTP
    - access methods, indexing, concurrency control, recovery
  - Warehouse—tuned for OLAP
    - complex OLAP queries, multidimensional view, consolidation.
- Different functions and different data
  - Missing data: Decision support requires historical data which operational DBs do not typically maintain
  - Data consolidation: DS requires consolidation (aggregation, summarization) of data from heterogeneous sources
  - Data quality: different sources typically use inconsistent data representations, codes and formats which have to be reconciled

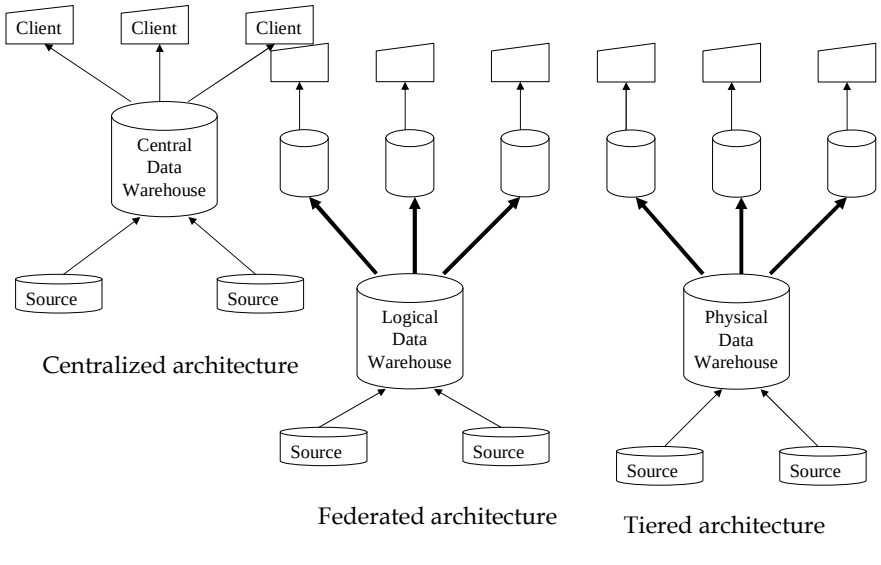
Three-Tier Architecture



Three-Tier Architecture

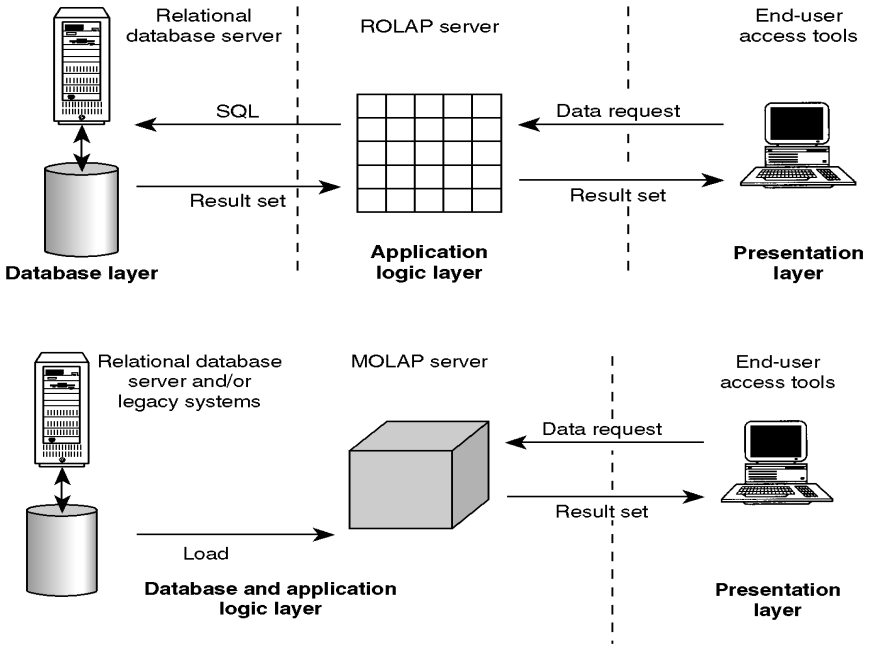
- Warehouse database server
  - Almost always a relational DBMS, rarely flat files
  - Schema design
  - Specialized scan, indexing and join techniques
  - Handling of aggregate views (querying and materialization)
  - Supporting query language extensions beyond SQL
  - Complex query processing and optimization
  - Data partitioning and parallelism
- OLAP servers
  - Relational OLAP (ROLAP): extended relational DBMS that maps operations on multidimensional data to standard relational operators
  - Multidimensional OLAP (MOLAP): special-purpose server that directly implements multidimensional data and operations
  - Hybrid OLAP (HOLAP): user flexibility, e.g., low level: relational, high-level: array
  - Specialized SQL servers: specialized support for SQL queries over star/snowflake schemas
- Clients
  - Query and reporting tools
  - Analysis tools
  - Data mining tools

Warehouse Physical Architectures



Three Data Warehouse Models

- *Enterprise warehouse*: collects all information about subjects (`customers, products, sales, assets, personnel`) that span the entire organization
  - Requires extensive business modeling (may take years to design and build)
- *Data Marts*: Departmental subsets that focus on selected subjects
  - Marketing data mart: customer, product, sales
  - Faster roll out, but complex integration in the long run
- *Virtual warehouse*: views over operational DBs
  - Materialize selective summary views for efficient query processing
  - Easy to build but require excess capability on operat. db servers



Data Warehouse

- A data warehouse is a
  - subject-oriented,
  - integrated,
  - time-varying,
  - non-volatile

  collection of data that is used primarily in organizational decision making

Data Warehouse—Subject-Oriented

- Organized around major subjects, such as customer, product, sales
- Focusing on the modeling and analysis of data for decision makers, not on daily operations or transaction processing
- Provide a simple and concise view around particular subject issues by excluding data that are not useful in the decision support process

Data Warehouse – Integrated

- Constructed by integrating multiple, heterogeneous data sources
  - relational databases, 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: currency, tax, breakfast covered, etc.
  - When data is moved to the warehouse, it is converted

Data Warehouse – Time Variant

- The time horizon for the data warehouse is significantly longer than that of operational systems
  - Operational database: current value data
  - Data warehouse data: provide information from a historical perspective (e.g., past 5-10 years)
- Every key structure in the data warehouse
  - Contains an element of time, explicitly or implicitly
  - But the key of operational data may or may not contain "time element"

Data Warehouse—Non-Volatile

- A physically separate store of data transformed from the operational environment
- Operational update of data does not occur in the data warehouse environment
  - Does not require transaction processing, recovery, and concurrency control mechanisms
  - Requires only two operations in data accessing:

    initial loading of data and *access of data*

Data Warehouse vs. Heterogeneous DBMS

- Traditional heterogeneous DB integration
  - Build wrappers/mediators on top of heterogeneous databases
  - Query driven approach
    - When a query is posed to a client site, a meta-dictionary is used to translate the query into queries appropriate for individual heterogeneous sites involved, and the results are integrated into a global answer set
    - Complex information filtering, compete for resources
- Data warehouse
  - update-driven, high performance
  - Information from heterogeneous sources is integrated in advance and stored in warehouses for direct query and analysis

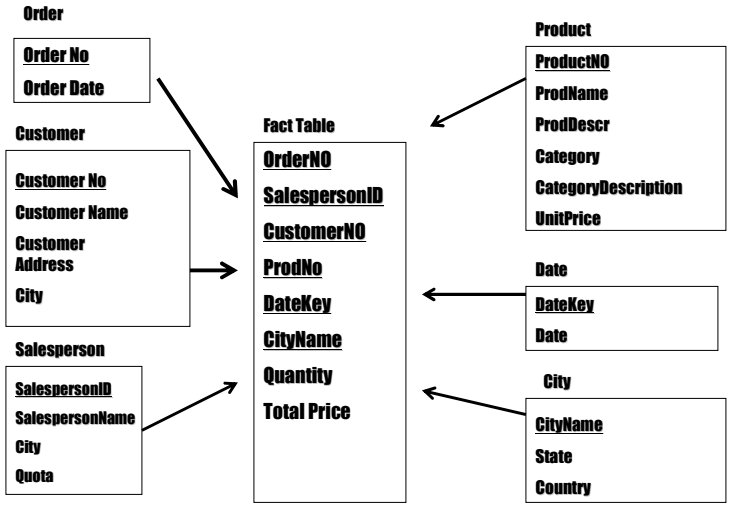
Data Warehouse vs. Operational DBMS

- OLTP (On-Line Transaction Processing)
  - Major task of traditional relational DBMS
  - Day-to-day operations: purchasing, inventory, banking, manufacturing, payroll, registration, accounting, etc.
- OLAP (On-Line Analytical Processing)
  - Major task of data warehouse system
  - Data analysis and decision making
- Distinct features (OLTP vs. OLAP):
  - User and system orientation: customer vs. market
  - Data contents: current, detailed vs. historical, consolidated
  - Database design: ER + application vs. star + subject
  - View: current, local vs. evolutionary, integrated
  - Access patterns: update vs. read-only but complex queries

Conceptual Modeling of Data Warehouses

- ER design techniques not appropriate - design should reflect multidimensional view
  - Star Schema
    - A fact table in the middle connected to a set of dimension tables
  - Snowflake Schema
    - A refinement of star schema where some dimensional hierarchy is normalized into a set of smaller dimension tables, forming a shape similar to snowflake
  - Fact Constellation Schema
    - Multiple fact tables share dimension tables, viewed as a collection of stars, therefore called galaxy schema or fact constellation

Example of a Star Schema



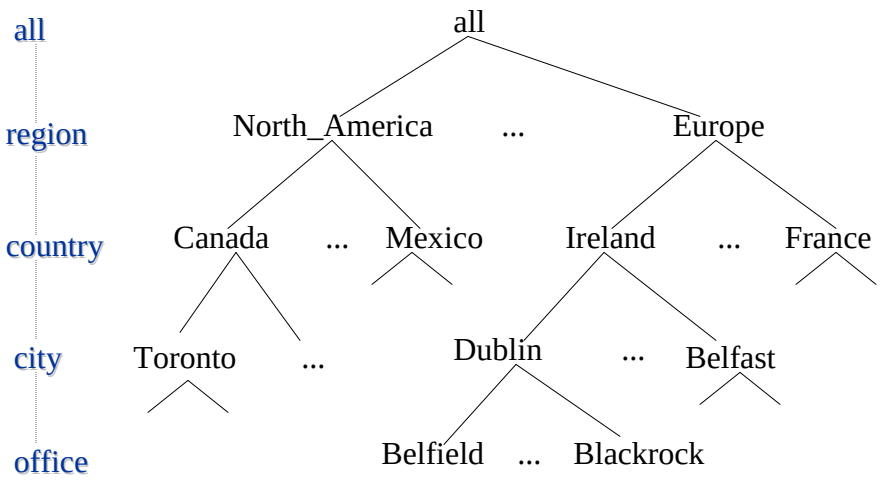
Star Schema

- A single fact table and a single table for each dimension
- Every fact points to one tuple in each of the dimensions and has additional attributes
- Does not capture hierarchies directly
- Generated keys are used for performance and maintenance reasons
- Fact constellation: Multiple Fact tables that share many dimension tables
  - Example: Projected expense and the actual expense may share dimensional tables

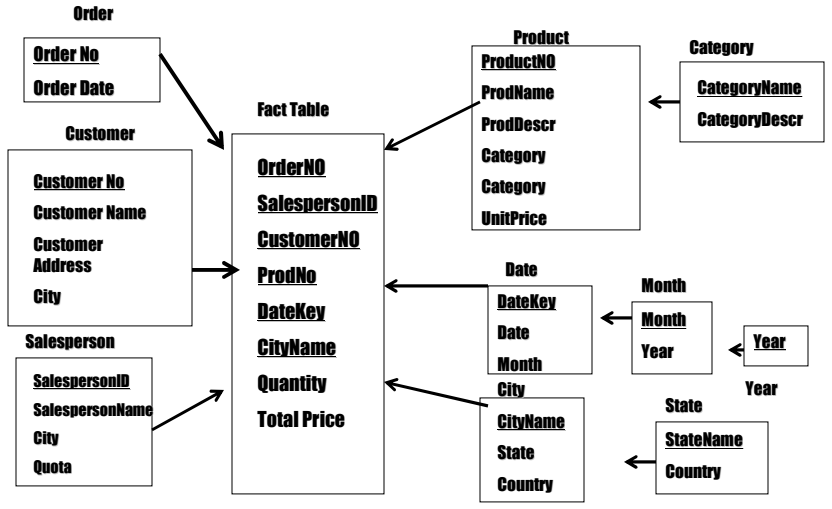
Some Terms

- Relation, which relates the dimensions to the measure of interest, is called the fact table (e.g. sale)
- Information about dimensions can be represented as a collection of relations – called the dimension tables (product, customer, store)
- Each dimension can have a set of associated attributes
- For each dimension, the set of associated attributes can be structured as a hierarchy

A Concept Hierarchy: Dimension (location)

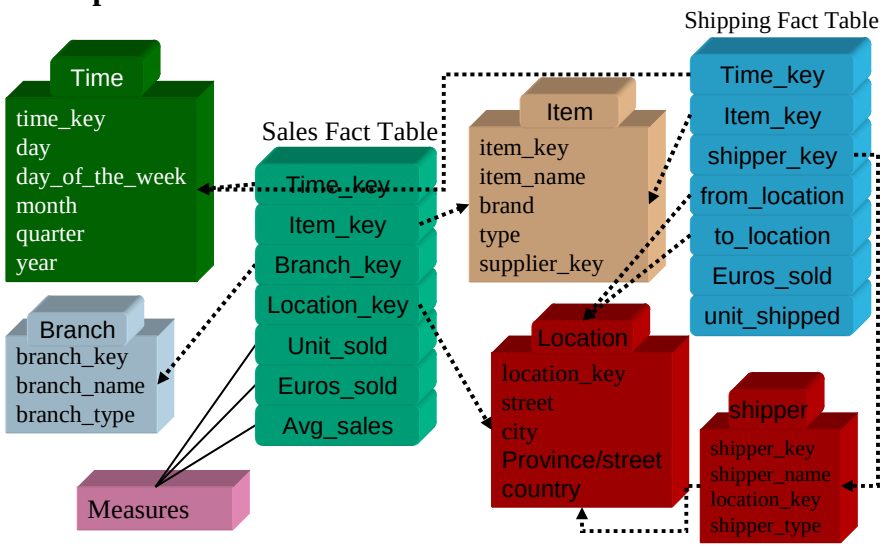


Example of a Snowflake Schema



Example of Fact Constellation

Multiple fact tables share dimension tables



Multidimensional Data Model

Fact relation

sale	Product	Client	Amt
	p1	c1	12
	p2	c1	11
	p1	c3	50
	p2	c2	8

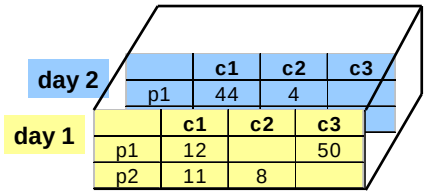
Two-dimensional cube

	c1	c2	c3
p1	12		50
p2	11	8	

Fact relation

sale	Product	Client	Date	Amt
	p1	c1	1	12
	p2	c1	1	11
	p1	c3	1	50
	p2	c2	1	8
	p1	c1	2	44
	p1	c2	2	4

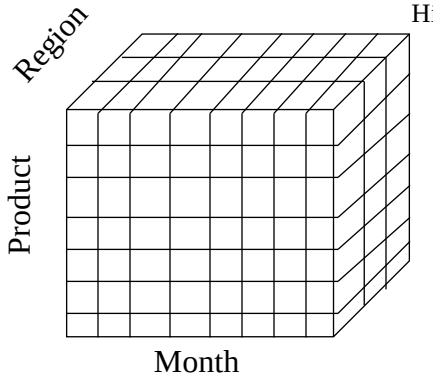
3-dimensional cube

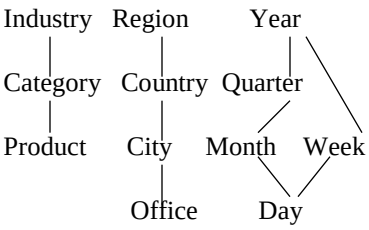


Multidimensional Data

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

Dimensions: Product, Location, Time
Hierarchical summarization paths

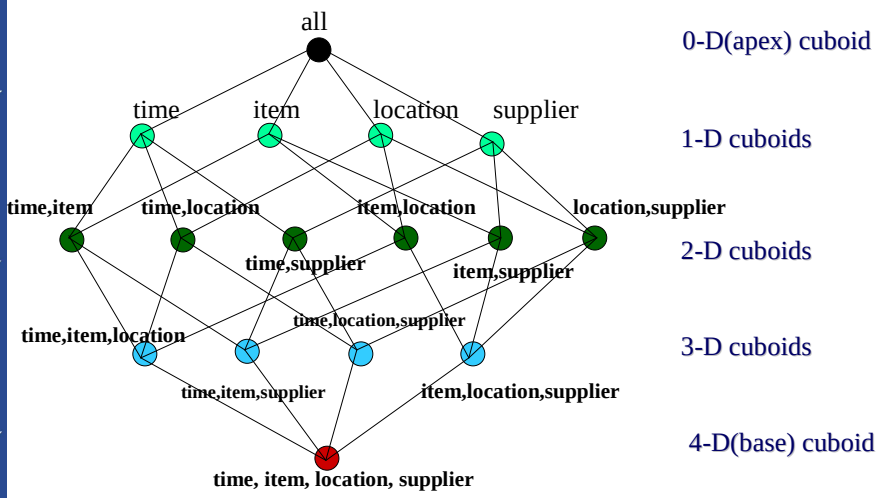




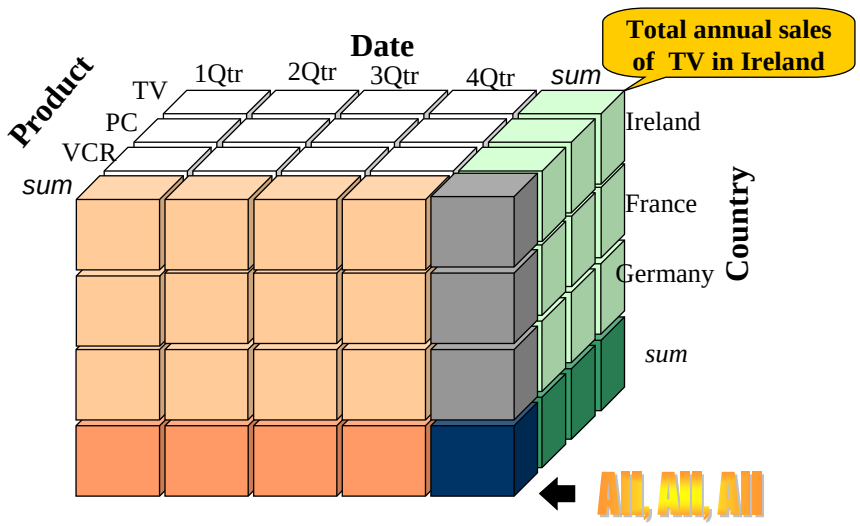
From Tables to Data Cubes

- A data warehouse is based on
  - multidimensional data model which views data in the form of a data cube
- A data cube allows data to be modeled and viewed in multiple dimensions (such as sales)
  - Dimension tables, such as item (item_name, brand, type), or time(day, week, month, quarter, year)
  - Fact table contains measures (such as dollars_sold) and keys to each of the related dimension tables
- Definitions
  - an n-Dimensional base cube is called a base cuboid
  - The top most 0-D cuboid, which holds the highest-level of summarization, is called the apex cuboid
  - The lattice of cuboids forms a data cube

Cube: A Lattice of Cuboids



A Sample Data Cube



Typical OLAP Operations

- Roll up (drill-up): summarize 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, visualization, 3D to series of 2D planes.*
- Other operations
  - *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)*
  - *rankings*
  - *time functions*: e.g. time avg.

Typical OLAP Operations: Drill & Roll

Drill Down

Total Sales
Total Sales per city
Total Sales per city per store
Total Sales per city per store per month

Drill Up

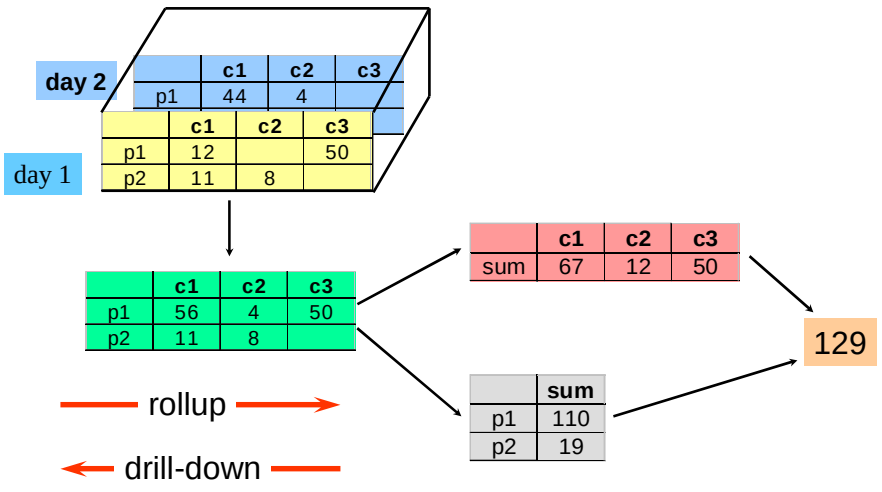
Drill Down

Total Sales
Total Sales per city
Total Sales per city by category

Drill Up

Drill Across

OLAP Queries: Rollup & Drill-Down



Typical OLAP Operations: Pivoting

- *Pivoting can be combined with aggregation*

sale	prodId	clientid	date	amt
	p1	c1	1	12
	p2	c1	1	11
	p1	c3	1	50
	p2	c2	1	8
	p1	c1	2	44
	p1	c2	2	4

day 2

	c1	c2	c3
p1	44	4	

day 1

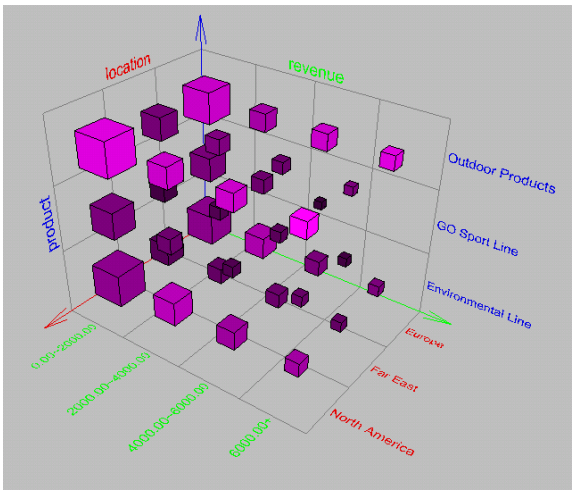
	c1	c2	c3
p1	12		50
p2	11	8	

	c1	c2	c3	Sum
1	23	8	50	81
2	44	4		48
Sum	67	12	50	129

	c1	c2	c3	Sum
p1	56	4	50	110
p2	11	8		19
Sum	67	12	50	129

The result of pivoting is called a cross-tabulation

Browsing a Data Cube



- Visualization
- OLAP capabilities
- Interactive manipulation

12 Codd's Rules for OLAP

1. Multi-Dimensional Concept View
   - The user should be able to see the data as being multidimensional insofar as it should be easy to 'pivot' or 'slice and dice'. (See later.)
2. Transparency
   - The OLAP functionality should be provided behind the user's existing software without adversely affecting the functionality of the 'host', i.e. OLAP server should shield the user for the complexity of the data and application
3. Accessibility
   - OLAP should allow the user to access diverse data stores (relational, nonrelational and legacy systems) but see the data within a common 'schema' provided by the OLAP tool, i.e. Users shouldn't have to know the location, type or layout of the data to access it.
   - OLAP server should automate the mapping of the logical schema to the physical data
4. Consistent Reporting Performance
   - There should not be significant degradation in performance with large numbers of dimensions or large quantities of data.

12 Codd's Rules for OLAP

5. Client-Server Architecture
   - Since much of the data is on mainframes, and the users work on PCs, the OLAP tool must be able to bring the two together
   - Different clients can be used
   - Data sources must be transparently supported by the OLAP server
6. Generic Dimensionality
   - Data dimensions must all be treated equally. Functions available for one dimension must be available for others.
7. Dynamic Sparse Matrix Handling
   - The OLAP tool should be able to work out for itself the most efficient way to store sparse matrix data.
8. Multi User Support
   - access, integrity, security

12 Codd's Rules for OLAP

9. Unrestricted Cross-Dimensional Operations
   - e.g., individual office overheads are allocated according to total corporate overheads divided in proportion to individual office sales.
   - Non-additive formulas cause the problems
     - Contribution = Revenue - Total Costs
     - Margin Percentage = Margin / Revenue
10. Intuitive Data Manipulation
   - Navigation should be done by operations on individual cells rather than menus.
   - Dimensions defined should allow automatic reorientation, drill-down, zoom-out, etc
   - Interface must be intuitive
11. Flexible Reporting
   - Row and column headings must be capable of more than one dimension each, and of displaying subsets of any dimension.
12. Unlimited Dimensions and Aggregation Levels
   - 15 - 20 dimensions are required in modelling, and within each there may be many hierarchical levels, i.e. unlimited aggregation
   - Rare in reporting to go beyond 12 dimensions, 6-7 is usual

DW Back-End Tools and Utilities

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

DW Information Flows

- INFLOW - Processes associated with the extraction, cleansing, and loading of the data from the source systems into the data warehouse.
- UPFLOW - Processes associated with adding value to the data in the warehouse through summarizing, packaging, and distribution of the data.
- DOWNFLOW - Processes associated with archiving and backing-up/recovery of data in the warehouse.
- OUTFLOW - Processes associated with making the data available to the end-users.
- METAFLOW - Processes associated with the management of the metadata.

Data Cleaning

- why?
  - Data warehouse contains data that is analyzed for business decisions
  - More data and multiple sources could mean more errors in the data and harder to trace such errors
  - Results in incorrect analysis
- finding and resolving inconsistency in the source data
- detecting data anomalies and rectifying them early has huge payoffs
- Important to identify tools that work together well
- Long Term Solution
  - Change business practices and data entry tools
  - Repository for meta-data

Data Cleaning Techniques

- Transformation Rules
  - Example: translate "gender" to "sex"
- Uses domain-specific knowledge to do scrubbing
- Parsing and fuzzy matching
  - Multiple data sources (can designate a preferred source)
- Discover facts that flag unusual patterns (auditing)
  - Some dealer has never received a single complaint

Load

- Issues:
  - huge volumes of data to be loaded
  - small time window (usually at night) when the warehouse can be taken off-line
  - When to build indexes and summary tables
  - allow system administrator to monitor status, cancel suspend, resume load, or change load rate
  - restart after failure with no loss of data integrity
- Techniques:
  - batch load utility: sort input records on clustering key and use sequential I/O; build indexes and derived tables
  - sequential loads still too long (~100 days for TB)
  - use parallelism and incremental techniques

Refresh

- when to refresh
  - on every update: too expensive, only necessary if OLAP queries need current data *(e.g., up-the-minute stock quotes)*
  - periodically (e.g., every 24 hours, every week) or after "significant" events
  - refresh policy set by administrator based on user needs and traffic
  - possibly different policies for different sources
- how to refresh
  - Full extract from base tables
    - read entire source table or database: expensive
  - Incremental techniques
    - detect & propagate changes on base tables: replication servers
    - logical correctness
    - transactional correctness: incremental load

Metadata Repository

- Administrative metadata
  - source databases and their contents
  - warehouse schema, view & derived data definitions
  - dimensions, hierarchies
  - pre-defined queries and reports
  - data mart locations and contents
  - data partitions
  - data extraction, cleansing, transformation rules, defaults
  - data refresh and purging rules
  - user profiles, user groups
  - security: user authorization, access control
- Business data
  - business terms and definitions
  - ownership of data
  - charging policies
- Operational metadata
  - data lineage: history of migrated data and sequence of transf-s applied
  - currency of data: active, archived, purged
  - monitoring information: warehouse usage statistics, error reports, audit trails.

Design of a DW: A Business Analysis Framework

- Four views regarding the design of a data warehouse
  - Top-down view: allows selection of the relevant information necessary for the data warehouse (mature)
  - Bottom-up: Starts with experiments and prototypes (rapid)
  - Data source view
    - exposes the information being captured, stored, and managed by operational systems
  - Data warehouse view
    - consists of fact tables and dimension tables
  - Business query view
    - sees the perspectives of data in the warehouse from the view of end-user

Data Warehouse Design Process

- From software engineering point of view
  - Waterfall: structured and systematic analysis at each step before proceeding to the next
  - Spiral: rapid generation of increasingly functional systems, short turn around time, quick turn around
- Typical data warehouse design process
  - Choose a business process to model, e.g., orders, invoices, etc.
  - Choose the grain (*atomic level of data*) of the business process
  - Choose the dimensions that will apply to each fact table record
  - Choose the measure that will populate each fact table record

DW Design: Issues to Consider

- What data is needed?
- Where does it come from?
- How to clean data?
- How to represent in warehouse (schema)?
- What to summarize?
- What to materialize?
- What to index?

DW Data Management: Issues to Consider

- Meta-data
- Data sourcing
- Data quality
- Data security
- Granularity
- History- how long and how much?
- Performance

Common DW Problems

- Underestimation of resources for data loading
- Hidden problems with source systems
- Required data not captured
- Increased end-user demands
- Data homogenization
- High demand for resources
- Data ownership
- High maintenance
- Long duration projects
- Complexity of integration

Research Issues

- Data cleaning
  - focus on data inconsistencies, not schema differences
  - data mining techniques
- Physical Design
  - design of summary tables, partitions, indexes
  - tradeoffs in use of different indexes
- Query processing
  - selecting appropriate summary tables
  - dynamic optimization with feedback
  - query optimization: cost estimation, use of transformations, search strategies
  - partitioning query processing between OLAP server and backend server.
- Warehouse Management
  - incremental refresh techniques
  - computing summary tables during load
  - failure recovery during load and refresh
  - process management: scheduling queries, load and refresh
  - use of workflow technology for process management

OLAP Mining: An Integration of DM and DW

- **Data mining systems, DBMS, Data warehouse systems coupling**
  - No coupling, loose-coupling, semi-tight-coupling, tight-coupling
- **On-line analytical mining data**
  - integration of mining and OLAP technologies
- **Interactive mining multi-level knowledge**
  - Necessity of mining knowledge and patterns at different levels of abstraction by drilling/rolling, pivoting, slicing/dicing, etc.
- **Integration of multiple mining functions**
  - Characterized classification, first clustering and then association

Summary

- What is a data warehouse
- Data warehouse architectures
  - Conceptual DW Modelling
  - Physical DW Modelling
- A multi-dimensional data model
  - Data Cubes
  - Main OLAP operations
- Data warehouse implementation and maintenance

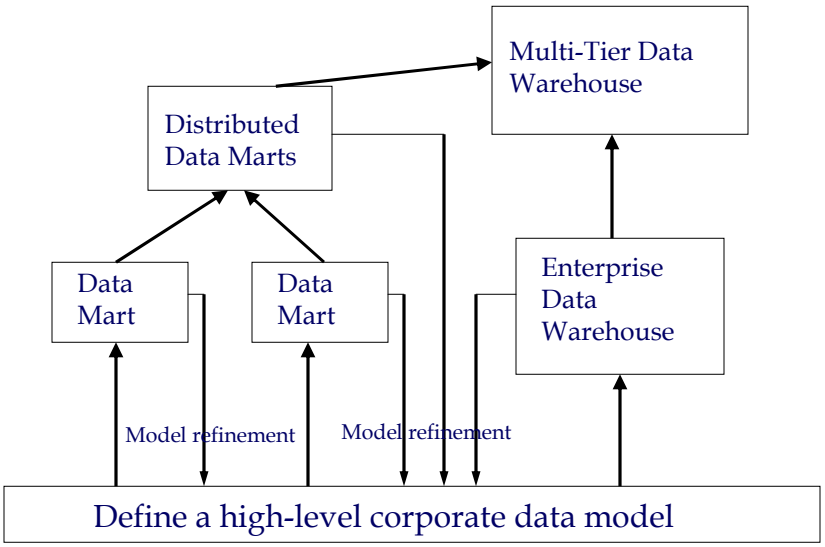
What else did you get from this lecture?

Additional Slides

Some Critics for Data Cubes

- Jargon for IT professionals
- Metaphor for end-users
  - Not useful beyond introduction
  - Users expect to see one
  - Rubic's cube
    - Easy to manipulate but difficult to solve
- Alternatives are better
  - Cognos' ways, Thomsen's multi-dimensional domain diagrams, Bulos's Adapt diagrams or simply well designed interactive reports

DW Development: A Recommended Approach



DMQL: Language Primitives

Nice presentation on Data Mining Query Languages can be found here:
http://www.cs.wisc.edu/EDAM/slides/Data%20Mining%20Query%20Languages.ppt

- Cube Definition (Fact Table)
  - define cube <cube_name> [<dimension_list>]: <measure_list>
- Dimension Definition (Dimension Table)
  - define dimension <dimension_name> as (<attribute_or_subdimension_list>)
- Special Case (Shared Dimension Tables)
  - First time as "cube definition"
  - define dimension <dimension_name> as <dimension_name_first_time> in cube <cube_name_first_time>

Defining a Star Schema in DMQL

```
define cube sales_star [time, item, branch, location]:
        dollars_sold = sum(sales_in_dollars),
        avg_sales = avg(sales_in_dollars),
        units_sold = count(*)
define dimension time as
        (time_key, day, day_of_week, month, quarter, year)
define dimension item as
        (item_key, item_name, brand, type, supplier_type)
define dimension branch as
        (branch_key, branch_name, branch_type)
define dimension location as
        (location_key, street, city, province_or_state, country)
```

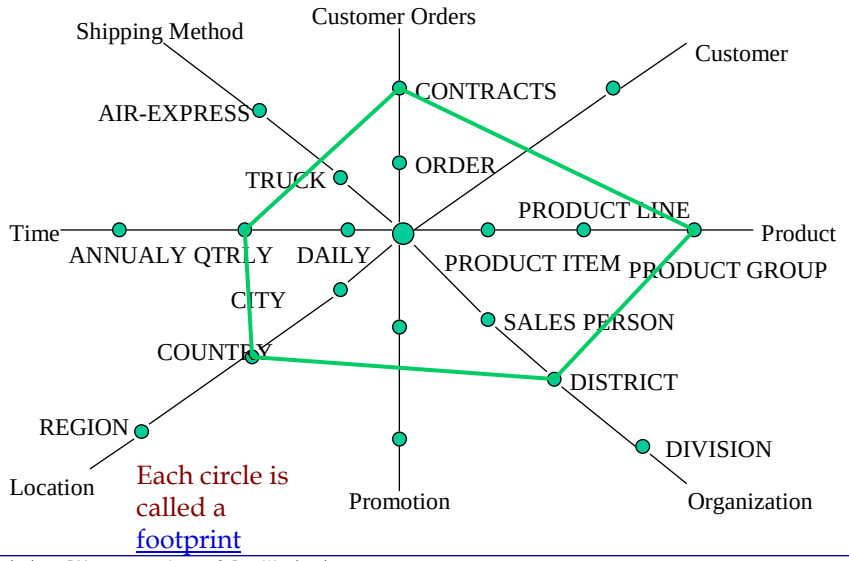

Defining a Snowflake Schema in DMQL

```
define cube sales_snowflake [time, item, branch, location]:
        dollars_sold = sum(sales_in_dollars),
        avg_sales = avg(sales_in_dollars),
        units_sold = count(*)
define dimension time as
        (time_key, day, day_of_week, month, quarter, year)
define dimension item as
        (item_key, item_name, brand, type, supplier(supplier_key, supplier_type))
define dimension branch as
        (branch_key, branch_name, branch_type)
define dimension location as
        (location_key, street, city(city_key, province_or_state, country))
```

Defining a Fact Constellation in DMQL

define cube sales [time, item, branch, location]:
dollars_sold = sum(sales_in_dollars), avg_sales = avg(sales_in_dollars),
units_sold = count(*)
define dimension time as (time_key, day, day_of_week, month, quarter, year)
define dimension item as (item_key, item_name, brand, type, supplier_type)
define dimension branch as (branch_key, branch_name, branch_type)
define dimension location as (location_key, street, city, province_or_state, country)
define cube shipping [time, item, shipper, from_location, to_location]:
dollar_cost = sum(cost_in_dollars), unit_shipped = count(*)
define dimension time as time in cube sales
define dimension item as item in cube sales
define dimension shipper as (shipper_key, shipper_name, location as location in cube sales, shipper_type)
define dimension from_location as location in cube sales
define dimension to_location as location in cube sales

A Star-Net Query Model

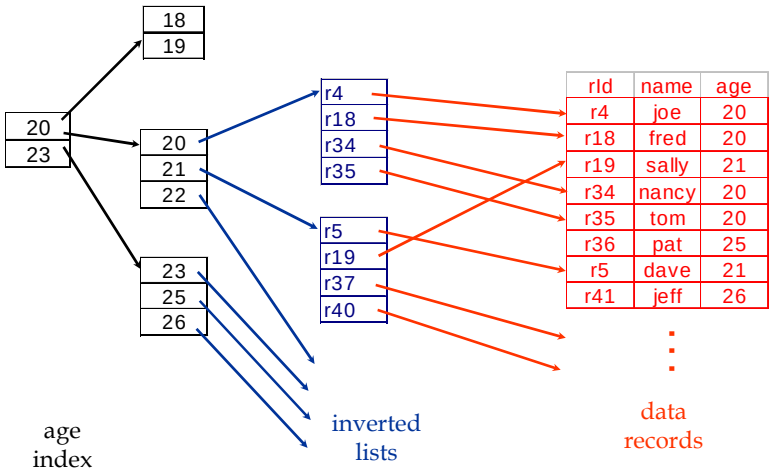


Index Structures

This and the following slides on Indexing for DW are adopted with minor modifications from: http://infolab.stanford.edu/~hector/cs245/Notes12.ppt

- Indexing principle:
  - mapping key values to records for associative direct access
- Most popular indexing techniques in relational database: B+-trees
- For multi-dimensional data, a large number of indexing techniques have been developed: R-trees
- Index structures applied in warehouses
  - inverted lists
  - bit map indexes
  - join indexes
  - text indexes

Inverted Lists



- Query:
  - Get people with age = 20 and name = "fred"

List for age = 20: r4, r18, r34, r35
List for name = "fred": r18, r52
Answer is intersection: r18

Bitmap Indexes

- Bitmap index: An indexing technique that has attracted attention in multi-dimensional database implementation

table

Customer	City	Car
c1	Detroit	Ford
c2	Chicago	Honda
c3	Detroit	Honda
c4	Poznan	Ford
c5	Paris	BMW
c6	Paris	Nissan

Bitmap Indexes

- The index consists of bitmaps:

Index on City:

ec1	Chicago	Detroit	Paris	Poznan
1	0	1	0	0
2	1	0	0	0
3	0	1	0	0
4	0	0	0	1
5	0	0	1	0
6	0	0	1	0

bitmaps

Index on Car:

ec1	BMW	Ford	Honda	Nissan
1	0	1	0	0
2	1	0	1	0
3	0	0	1	0
4	0	1	0	0
5	1	0	0	0
6	0	0	0	1

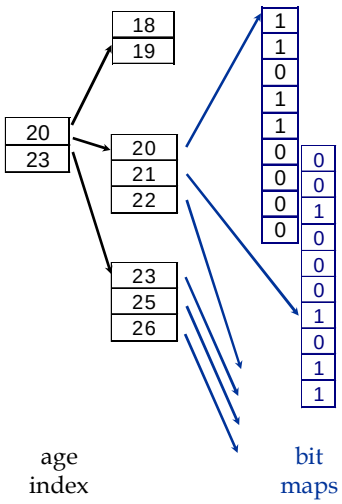
bitmaps

- Index on a particular column
- Index consists of a number of bit vectors - bitmaps
- Each value in the indexed column has a bit vector (bitmaps)
- The length of the bit vector is the number of records in the base table
- The i-th bit is set if the i-th row of the base table has the value for the indexed column

Bitmap Indexes

- Index on a particular column
- Index consists of a number of bit vectors - bitmaps
- Each value in the indexed column has a bit vector (bitmaps)
- The length of the bit vector is the number of records in the base table
- The *i*-th bit is set if the *i*-th row of the base table has the value for the indexed column

Bitmap Index



id	name	age
1	joe	20
2	fred	20
3	sally	21
4	nancy	20
5	tom	20
6	pat	25
7	dave	21
8	jeff	26

data records

Query:
Get people with age = 20 and name = "fred"

List for age = 20:
1101100000
List for name = "fred":
0100000001
Answer is intersection:
0100000000

Suited well for domains with small cardinality

Bitmap Index – Summary

- With efficient hardware support for bitmap operations (AND, OR, XOR, NOT), bitmap index offers better access methods for certain queries
  - e.g., selection on two attributes
- Some commercial products have implemented bitmap index
- Works poorly for high cardinality domains since the number of bitmaps increases
- Difficult to maintain - need reorganization when relation sizes change (new bitmaps)

Join

- "Combine" SALE, PRODUCT relations
- In SQL: SELECT * FROM SALE, PRODUCT

sale	prodId	storeId	date	amt
	p1	c1	1	12
	p2	c1	1	11
	p1	c3	1	50
	p2	c2	1	8
	p1	c1	2	44
	p1	c2	2	4

product	id	name	price
	p1	bolt	10
	p2	nut	5

joinTb	prodId	name	price	storeId	date	amt
	p1	bolt	10	c1	1	12
	p2	nut	5	c1	1	11
	p1	bolt	10	c3	1	50
	p2	nut	5	c2	1	8
	p1	bolt	10	c1	2	44
	p1	bolt	10	c2	2	4

Join Indexes

join index

product	id	name	price	jIndex
	p1	bolt	10	r1,r3,r5,r6
	p2	nut	5	r2,r4

sale	rId	prodId	storeId	date	amt
	r1	p1	c1	1	12
	r2	p2	c1	1	11
	r3	p1	c3	1	50
	r4	p2	c2	1	8
	r5	p1	c1	2	44
	r6	p1	c2	2	4

Join Indexes

- Traditional indexes map the value to a list of record ids. Join indexes map the tuples in the join result of two relations to the source tables.
- In data warehouse cases, join indexes relate the values of the dimensions of a star schema to rows in the fact table.
  - For a warehouse with a Sales fact table and dimension city, a join index on city maintains for each distinct city a list of RIDs of the tuples recording the sales in the city
- Join indexes can span multiple dimensions