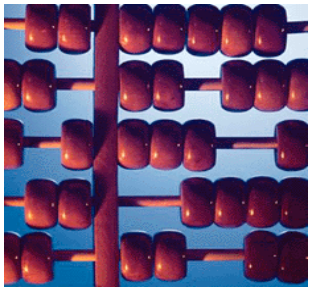
Database Heterogeneity

Lecture 13



Outline

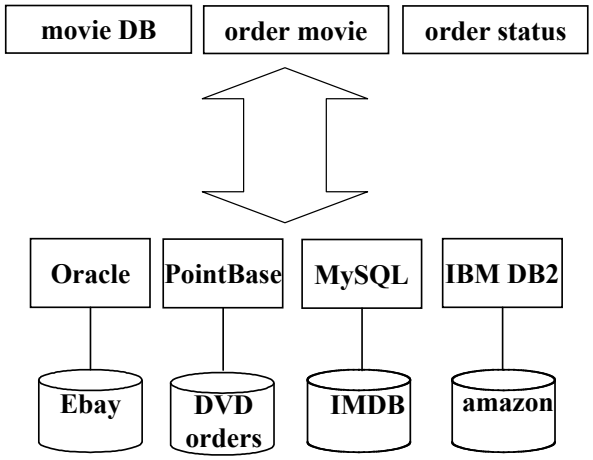
- Database Integration
- Wrappers
- Mediators
- Integration Conflicts
- Data Warehousing

Motivation

- If we all use the same database, things are “quite simple”
- However, we often use
  - Heterogeneous data sources
  - Heterogeneous DBMS
  - Different data formats/data types etc.
- Key word: company merger

1. Database Integration

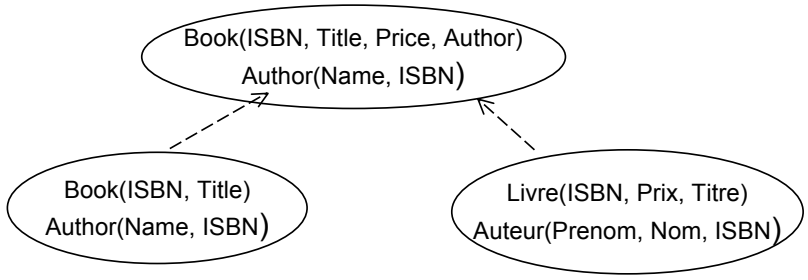
- Goal: providing a uniform access to multiple heterogeneous information sources
- More than data exchange (e.g., ASCII, EDI, XML)
- Old problem, difficult, well-known (partial) solutions



Data Integration

- We did not directly work on it in the project; however, used different service interfaces which requires data interchange
- Typically requires (some) manual interaction

Old-School Approach (1)
Manual, Global Integration



- Manually merge multiple databases into a new global database
- Time consuming and error prone
- Local autonomy lost
- Static solution
- Does not scale with number of databases

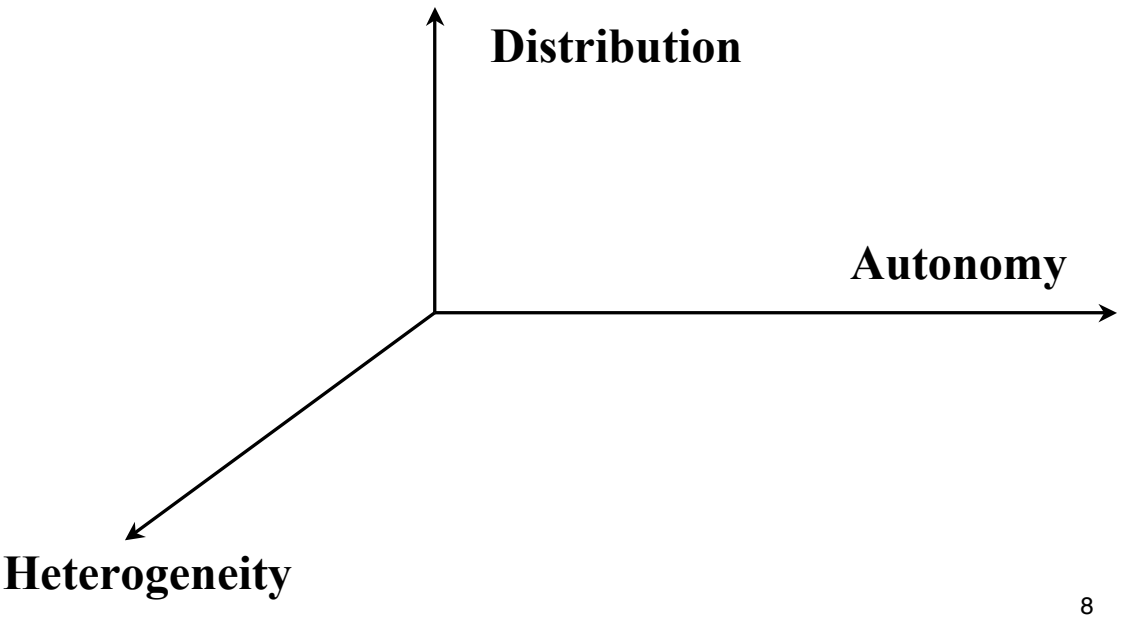
Old-School Approach (2) Multi-database Language Approach

- No attempt at integrating *schemas*
- Language (e.g., SQL) used to integrate information sources at run-time
- Simple example:

```
Use S1, S2
Select Titre
From S1.Book, S2.Livre
Where S1.Book.ISBN = S2.Livre.ISBN
```

- Not transparent (you need to know *all* databases!)
- Heavy burden on (expert) users
- Global queries subject to local changes

Problem Dimensions



How to Deal with Distribution?

- Problems
  - data access over the network
  - inconsistent replicated data
- Solutions
  - solved by using Web access (over HTTP)
  - Web Services, Java RMI, …
  - publishing using JSP
  - JDBC to access remote databases
  - etc.

How to Deal with Autonomy?

- Problems
  - changing structure of Web page
  - different coverage of Web sites
  - availability of services
- Solutions
  - manually adapt to changes
  - replication, materialization (availability)
  - contacts, agreements, … standards

How to Deal with Heterogeneity?

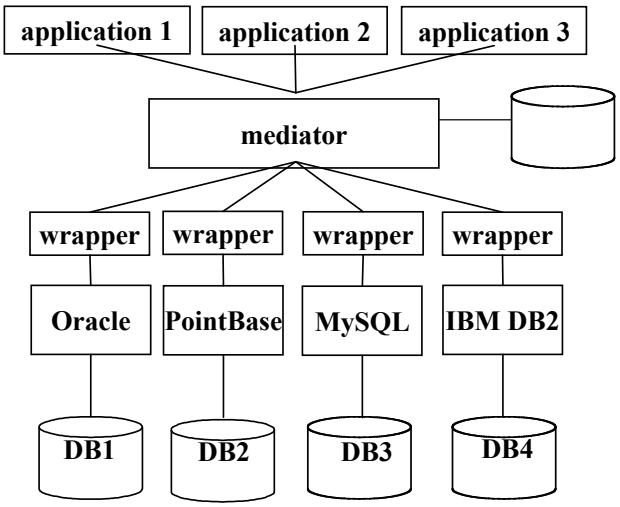
- Problems
  - Data models
  - Schemas
  - Data
- Solutions
  - Mappings, schema integration
  - Standards

Solution Variants

- General issues
  - Bottom-up vs. top-down engineering
  - Virtual vs. materialized integration
  - Read-only vs. read-write access
  - Transparency: language, schema, location

A Generic System Architecture

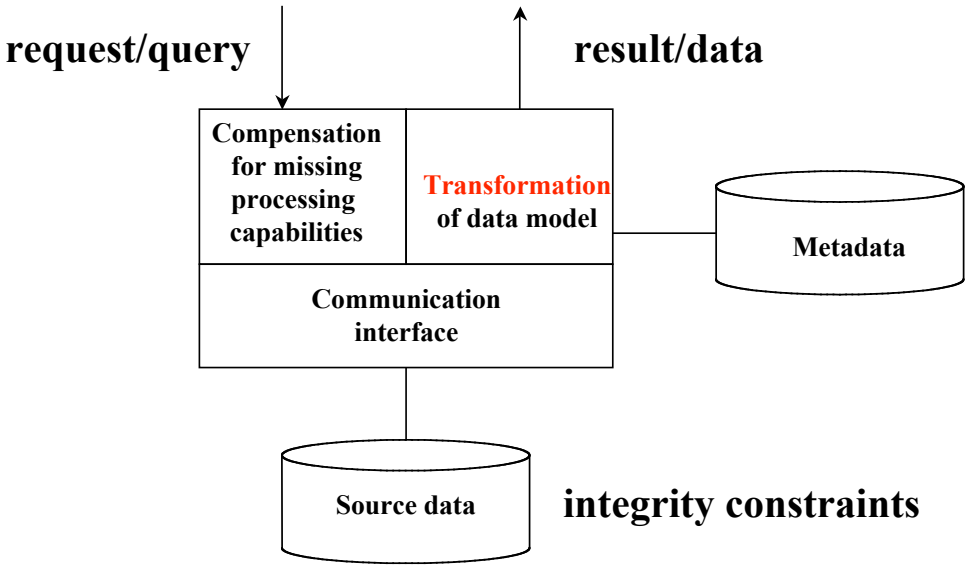
- One solution: the Wrapper-Mediator architecture



mediators integrate the data from the DBs

wrappers convert to a common representation

2. Wrappers



Wrapper Tasks

- Translate among different data models
- Data Model consists of
  - Data types
  - Integrity constraints
  - Operations (e.g. query language)
- Overcome other "syntactic" heterogeneity

A Closer Look at Data Models

- Data model used by sources
  - relational? HTML? XML? RDF? Custom? Text?
- Data model used by integrated DB
  - canonical data model (e.g. relational, XML)
- Query models
  - Structured queries (SQL), retrieval queries (in information retrieval), data mining (statistics)

Example: Wrapping Relational Data into XML/HTML

- Data types
  - trivial
- Integrity Constraints (e.g. primary keys)
  - requires XML Schema
- Operations
  - none in HTML

Example: Wrapping XML/HTML into Relational

- Data Types
  - which difficulties?
- Integrity Constraints
  - none in HTML
- Operations
  - requires generally XQuery
  - form fields can be considered as hard-coded queries

3. The Mediator

- Integrate data with same "real-world meaning", but different *representations*
  - Semantics are important
  - integration mapping ⇒ schema integration
  - can be implemented, e.g., as database views
- Decompose queries against the integrated schema to queries against source DBs
  - only for *virtual* integration

An Example: LAV

- Local As View approach
- Each local database is defined as a view on the integrated schema

A simple Example:

Source A: R1(prof, course, university)
Source B: R2(title, prof, course)

Definition of the global, integrated schema:

Global(prof, course, title, university)

Source A defined as:

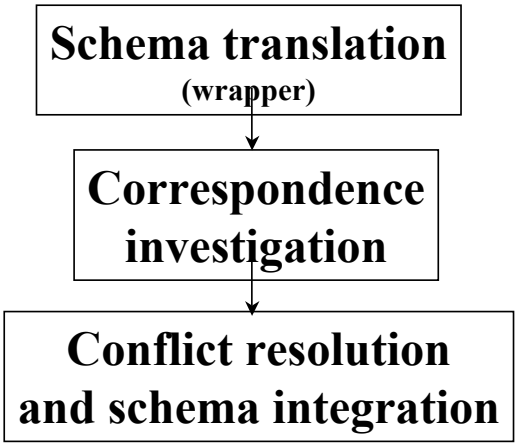
```
Create view R1 as
SELECT prof, course, university FROM Global
```

Source B defined as:

```
Create view R2 sa
SELECT  title, prof, course FROM Global
```

Schema Integration

- Standard Methodology



Identifying Schema Correspondences (1)

Sources of information

- source schema
- source database
- source application
- database administrator, developer, user

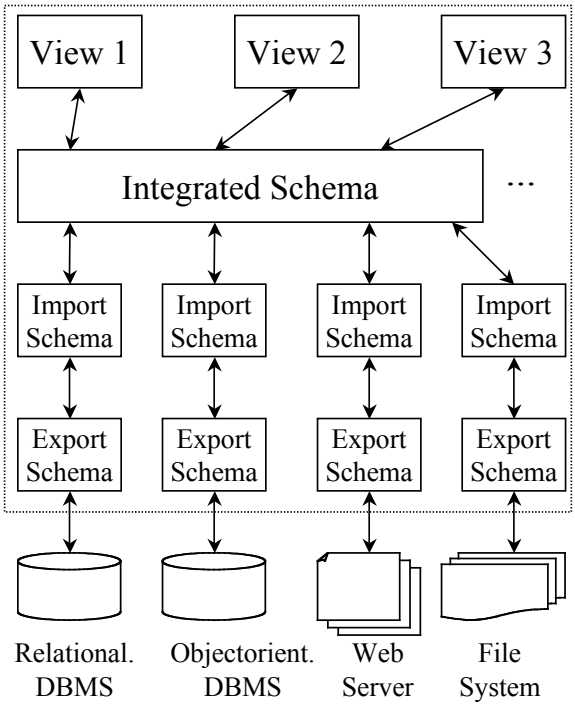
Identifying Schema Correspondences (2)

- Semantic correspondences
  - e.g. related names
  - One of the important current research topics
    - No obvious solutions yet
- Structural correspondences
  - reachability by paths
- Data analysis
  - distribution of values

A Closer Look at Schemas

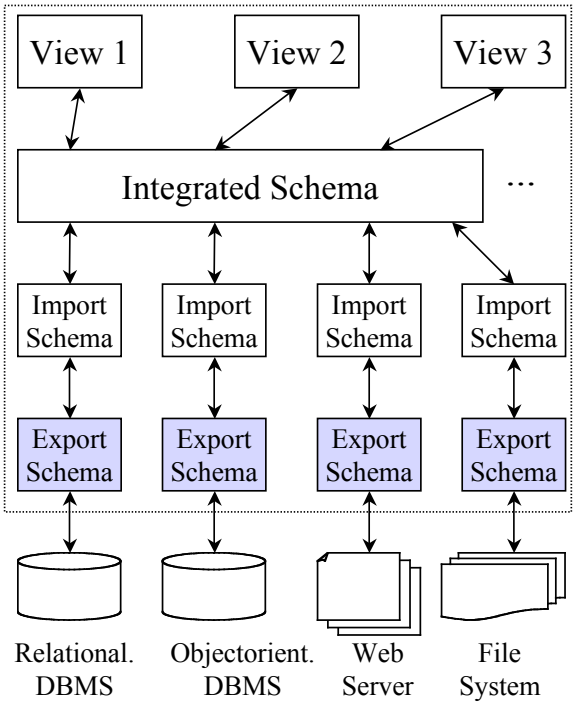
- Tight vs. loose integration
  - Is there a global schema?
- Support for semantic integration
  - collection, fusion, abstraction

A Widely Used Architecture: Federated DBMS



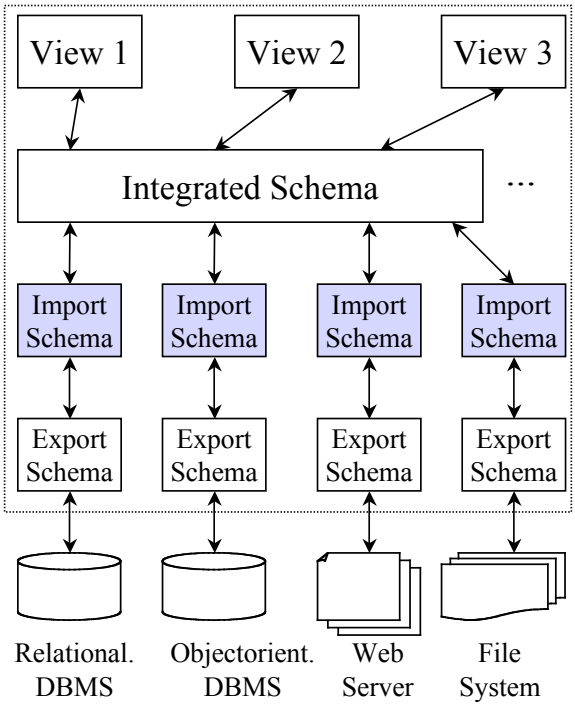
- accepted model for integrated database systems with integrated schema
- 5-level architecture
- data independence

Export Schema



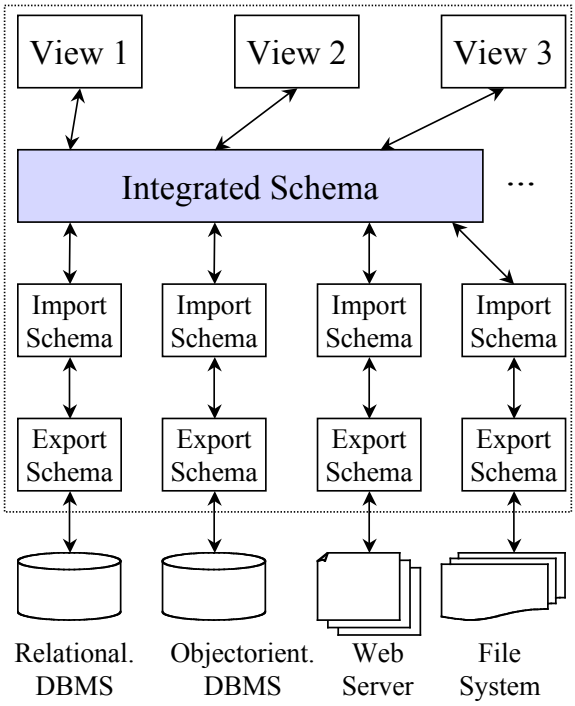
- provided by *data source*
- source DB can change w/o changing export schema

Import Schema



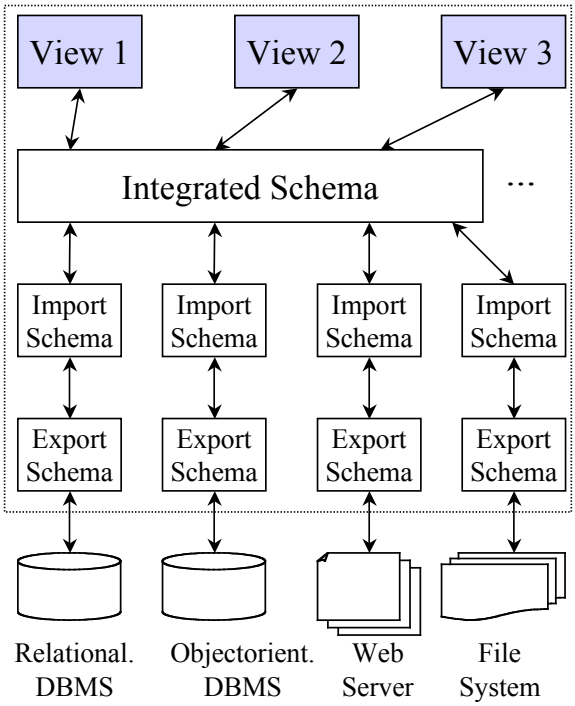
- provided by *wrapper*
- export schema can change w/o changing import schema

Integrated Schema



- provided by *mediator*
- import schemas can change w/o changing integrated schema

Application View



- provided by *application*
- integrated DB can change w/o changing application (code)

4. Handling Integration Conflicts

- What types of problems can one encounter integrating corresponding data?
- Different structural representation (e.g. attribute vs. table)
- Different naming schemes

Types of Conflicts

- Schema level
  - Naming conflicts
  - Structural conflicts
  - Classification conflicts
  - Constraint and behavioral conflicts
- Data level
  - Identification conflicts
  - Representational conflicts
  - Data errors

Conflict Resolution

- Depends on type of conflict
- Requires construction of mappings
- Mappings might be complex, e.g. not expressible as SQL views

Naming Conflicts

- Homonyms
  - same name used for different concepts
  - Resolution: introduce prefixes to distinguish the names
- Synonyms
  - different names for the same concepts
  - Resolution: introduce a mapping to a common name

Structural Conflicts

- Different, non-corresponding attributes
  - Resolution: create a relation with the union of the attributes
- Different datatypes
  - Resolution: build a mapping function
- Different data model constructs
  - e.g. attribute vs. relation
  - Resolution: requires higher order mappings

Classification Conflicts

- Relations can have different coverage (inclusion, non-empty intersection)
  - Resolution: build generalization hierarchies
- Additional problem
  - Identification of corresponding data instances
  - "real world" correspondence is application dependent

Data Correspondences

- Corresponding data instances
  - similar to naming conflicts at schema level
  - Resolution: mapping tables and functions
  - Similarity functions
- Corresponding data values, data conflicts
  - of corresponding data instances
  - Resolution: mapping tables and functions
  - Prefer data from more trusted data source

Constraint and Behavioral Conflicts

- Cardinality conflicts
  - different types of cardinality constraints on relationships
  - Resolution: use the more general constraint
- Behavioral conflicts for relation update
  - E.g. cascading delete vs. non-cascading
  - Resolution: add missing behavior at global level

More?

- Security
  - protecting data
- Data Quality
  - actively managing data quality
- Integration as Agreement Process
  - "emergent semantics"

5. Data Warehousing Motivation

- Economy is characterised by continuous changes in marketing and business opportunities
- Companies have accumulated data about their business
- Often, this data is heterogeneous, is obtained from different departments and/or customers/suppliers
- Businesses use data as decision support systems (based on information system)
- Information is often "hidden"

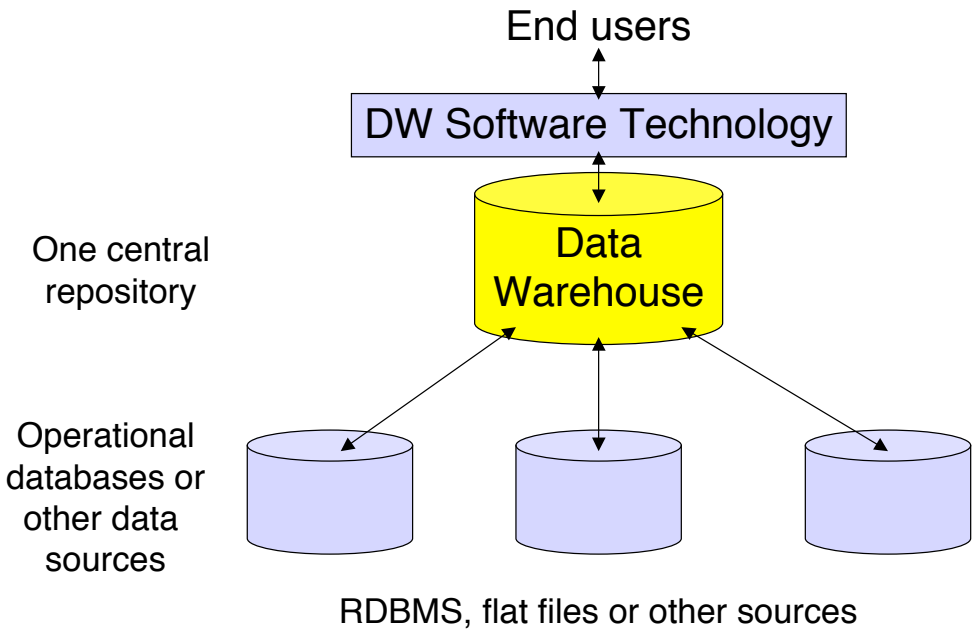
What is a data warehouse (DW)?

- Data repository with heterogeneous data sources
  - One approach to data integration
- Data is aggregated
- Contains several materialised views
  - (expressed e.g., as SQL views but stored)
- Views are updated at regular intervals
- Multi-dimension data model
- Designed for quick data retrieval by ad-hoc queries

Data Warehouse Features

- Data from multiple external data sources is
  - Extracted
  - Filtered
  - Merged
  - Enriched by historical information
  - Stored in a central repository
    - Original normal forms are often not used any more!
- Access to homogeneous, central repository with reduced access times
- Data warehouse (DW) is independent of original data sources

DW Architecture (Simplified)



Operational Database vs. DW

	Operational DB	DW
Purpose	For a particular application / purpose	Looks at several “views” at the same time
Requirements	Typically known	Rather vague
Main usage	Daily business	For management decision to maximise profit (analysis, forecasts, ad-doc reports)
Data access	Queries return relatively small results	Access to large data volumes
Tuning / data volume	Tuning for many queries to small data volumes	Infrequent access to large data volumes
Data freshness	Always up-to-date	Snapshots done at given intervals (mat. views)

Data warehousing

- Data warehousing is the process of creating a data warehouse

1. Data retrieval (extraction)
   - Includes data cleansing (data scrubbing): used to correct errors in data and potentially remove duplicate entries
   - ETL (next slide)
2. Data storage (includes indexing)
3. Data analysis (OLAP)

ETL (Extract-Transform-Load)

- Extract data from an external source
  - Heterogeneous resources and formats
- Transform data to fit to business needs
  - Certain data modifications are required
    - Selecting only certain attributes, joining tables, splitting/merging tables/columns
- Load data into a data warehouse
  - Different models to store data:
    - Keep historical values
    - Frequently update data, i.e. replace old values

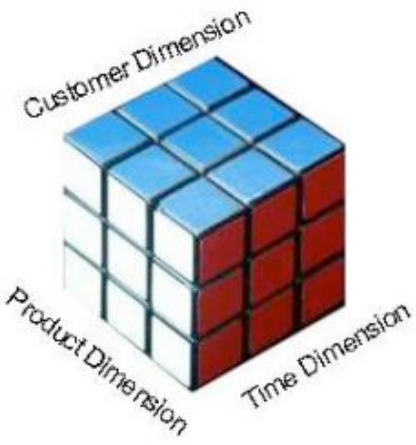
OLAP
OnLine Analytical Processing

- The content of a DW is analysed by OLAP
- To discover trends in data and patterns of behaviour
- Outcome is used for various marketing and business decisions
  - E.g., market analysis of demand and supply
- Usually, multi-dimensional queries are used
  - Requires multi-dimensional data structures such as data cubes, etc.

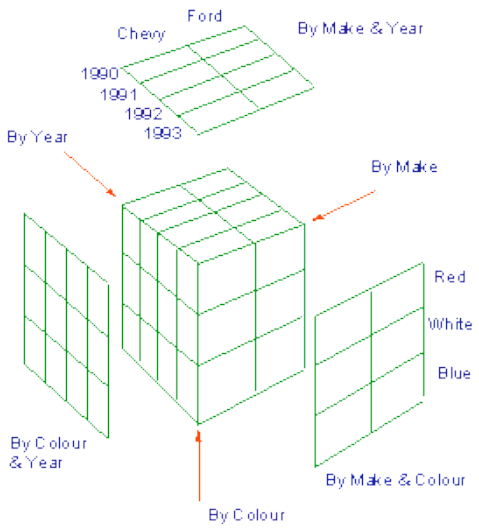
Data Cube and DW

- Multi-dimensional data structure that allows for fast retrieval of information
- Often, 3 dimensional cube
  - If more dimensions are used, it is called hypercube

Data Cube Examples



Source: http://hubpages.com/hub/DataCube



Source http://projects.cs.dal.ca/panda/datacube.html

Applications in Real World

- Supermarket chains
  - Accumulate data about customer behaviour
  - Use data mining to retrieve information
- Banks
- Insurance companies
- Stock markets
- Etc.

DW summary

- Data warehouse is a term that has been in use for several years
- Many companies use “data warehouses” but might not call them like that
- Several IT companies sell DW technology: often tightly coupled with relational databases
- *Main idea was to make you familiar with the basic concepts and terms*

Questions to Lecture