SYST 542
Decision Support Systems Engineering

Instructor: Kathryn Blackmond Laskey

Fall Semester, 2006

Unit 6: DSS Elements:
The Data Subsystem

Outline

- Functions of data subsystem
- Data modeling
  - Relational model
  - Object oriented model
  - Data and meta-data
- Emerging concepts in data management
  - Distributed databases
  - Data warehousing
  - Active databases and intelligent databases

The Way it Used to Be

- Each program managed its own data
  - data formats
  - access methods
  - updates and integrity checks
- Logical view of data tied to physical architecture of computer
- Data often duplicated, difficult to share among applications

Database Management

- Logical view of data separate from physical representation
- Standardized, common methods for access, updates, administration
- Data repositories shared among multiple users for multiple applications
- DBMS enforces consistency, security, coordination of interactions

Database Management Systems

- A database is a structure that can house information about multiple types of entities, as well as relationships among the entities.
- A database management system (DBMS) is a software product through which users interact with a database.
- A data model refers to how the DBMS views the data. A data model has two components:
  - Structure (schema)
  - Operations (updates, queries)
- A view refers to what a particular user sees. Different users may have different views.

Functions of DBMS

1. Data storage, retrieval, update
2. User-accessible catalog
3. Transaction support (including logging and rolling back)
4. Concurrency control services
5. Recovery services
6. Authorization services
7. Support for data communication
8. Integrity services
9. Services to promote data independence
10. Utility services
11. Replication support

Data Model

- A data model is a representation of the structure of a database
  - Data types
  - Relationships that can hold
  - Constraints that should hold on data (integrity rules)
  - Basic operations (retrieval, storage, query)
- Examples:
  - Relational data model (current standard)
    - » Invented by Codd
    - » First abstract data model to be defined
    - » Theoretical basis in predicate logic
  - Network and hierarchical data models (legacy models)
    - » Defined after the fact by abstraction from implementations
  - Object model (emerging)
  - Entity-relation model (conceptual-level model used in DB design)

Schema, Instance, State

- **Schema:**
  - Description of database
    - » Kinds of entities DB can hold
    - » Attributes of the entities
    - » Relationships that can hold among entities
  - Specified during design; changes infrequently
- **Instance:**
  - Particular objects or records
  - Changes every time records are added or deleted
- **DB state:**
  - Snapshot of DB at a given moment of time
  - Current set of instances of all schema constructs

- *The **schema** describes the categories of things that can exist in the database.*
- *The **instances** are the things in each category that exist in the actual database at a given time.*
- *The **state** completely describes the database at a given time*

STUDENT

Surname	StudentNumber	Major

COURSE

CourseName	CourseNumber	Department

Part of a schema for a university database

STUDENT

Surname	StudentNumber	Major
Amani	932478063	SYST
Brooks	473206793	CS
Craig	824529754	ECE
…	…	…

Some instances of the student category

Key Concepts

- Type, class - represents a set of real or abstract related things having common characteristics
  - STUDENT, COURSE
- Instance - represents individual member of set (also called entity instance or object instance)
  - [Amani, 932478063, SYST] is an instance of STUDENT
- Attribute - relevant properties, qualities or characteristics of things
  - "Surname" is an attribute of STUDENT
- Domain - a named and defined set of values from which one or more attributes can take their values
  - Domain of "Surname" is alphabetic strings with first letter capitalized
- Relationships - represent associations between entities
  - Part/whole (connection relationship)
  - Type/subtype (categorization relationship)
- Names - labels for things (entity instances)
- Key - name that uniquely identifies an instance
  - Student number uniquely identifies an instance of STUDENT

Ontology

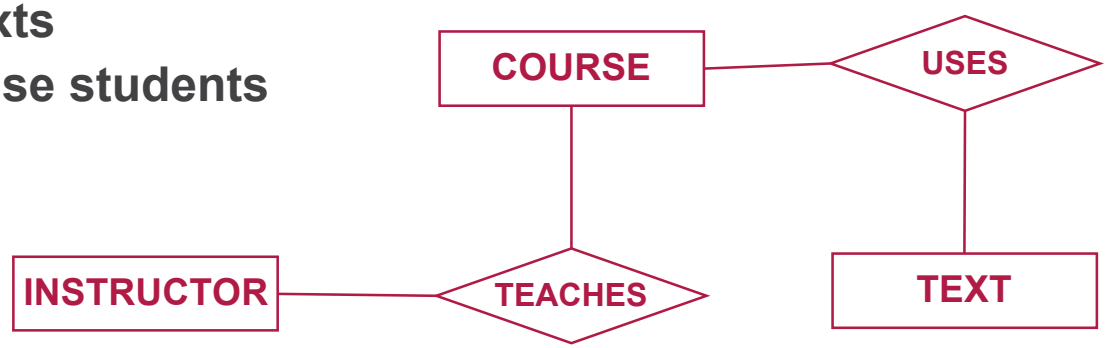
- Categories of things that can exist in a domain
  - Organized hierarchically into types / subtypes
  - Things of a given type have:
    - » Similar structure (attributes)
    - » Similar behavior (processes)
    - » Similar associations
  - Subtypes can inherit structure, behavior, association from supertype
- Ontology includes:
  - Physical or abstract entities
  - Things or processes
  - Entities and relationships
- Ontology specifies
  - Types of things that can be represented
  - Allowable values for attributes
  - Allowable relationships
- Specifying an ontology
  - Formal - types defined by logical rules
  - Informal - types specified via prototypical instances

Ontology versus DB Schema

- DB schema and ontology both represent:
  - Types of entities that can exist
  - Relationships that can hold among entities
- Purpose and usage is different
  - DB schema represents structure of data in a particular data store and is typically not designed for interoperability
  - Ontology is intended to represent knowledge about a domain in a structured, shareable way

Entity-Relation Model

- **Entity represents things in the domain**
  - Entity ***type*** represents collection of similar entity ***instances***
  - Each entity type has characteristic ***attributes***
    - » Attributes of STUDENT: Surname, StudentNumber, GPA, Major, etc.
    - » Attributes can be:
      - Simple/composite
      - Single-valued/multi-valued
      - Stored/derived
  - ***Key*** is an attribute that uniquely identifies an entity
- **Entities can be related to other entities**
  - Instructors teach courses
  - Courses use texts
  - Professors advise students



See http://www.modelingstyle.info for tips on creating better software diagrams

Relational Model

Invented by Edgar Codd

- Seminal paper published in 1970
- Came into widespread use in 1980s

Current standard for data modeling

- Many commercial products
- Most highly developed theory
- Many legacy systems are relational DBs or have relational DB wrappers

Advantages

- Logical and physical characteristics separated
- Well understood by community of modelers
- Powerful operators available
- Sound, well-developed, clearly defined theoretical basis
- Many COTS systems provide full or partial implementations

Definitions

- **A relation is a 2-dimensional table in which**
  - **entries in table are single-valued (may be null)**
  - **each column has a distinct name (called attribute name)**
  - **all values in column are values of same attribute**
  - **order of columns is immaterial**
  - **each row is distinct***
  - **order of rows is immaterial**
- ***(Unnormalized relation satisfies all but *)***
- **A relational database is an organized collection of relations**
- **Keys**
  - **Primary key is attribute or collection of attributes that uniquely defines each row**
  - **Foreign key is attribute in a relation that matches primary key in another relation**

Primary key in STUDENT relation

Foreign key - primary key in DEGREE relation

STUDENT

Name	StudentNumber	Major
Amani	932478063	SYST
Brooks	473206793	CS
Craig	824529754	ECE
Dopko	378623972	SYST
...	...	...

Operations on Databases: The Relational Algebra

SELECT - create a new table containing rows of original table satisfying certain criteria

PROJECT - create a new table by deleting certain columns from original table

PRODUCT - form new table by concatenating each row in first table with each row in second

UNION - form new table containing all rows of both input tables (tables must match on all attributes)

DIFFERENCE - form new table containing all rows in first table but not in second (tables must match on all attributes)

JOIN* - form new table by combining records from two tables that match on a common attribute

INTERSECTION* - form new table containing all entries appearing in both input tables (tables must match on all attributes)

* Operations are redundant

SELECT

Name	StudentNumber	Year	Major	GPA
Amani	932478063	Sophomore	SYST	3.72
Brooks	473206793	Freshman	CS	3.19
Craig	824529754	Junior	ECE	2.86
Dopko	378623972	Junior	SYST	3.25
...	...	...	...	...

Name	StudentNumber	Year	Major	GPA
Amani	932478063	Sophomore	SYST	3.72
Dopko	378623972	Junior	SYST	3.25
...	...	...	...	...

PROJECT

Name	StudentNumber	Year	Major	GPA
Amani	932478063	Sophomore	SYST	3.72
Brooks	473206793	Freshman	CS	3.19
Craig	824529754	Junior	ECE	2.86
Dopko	378623972	Junior	SYST	3.25
...	...	...	...	...

Name	StudentNumber	Major
Amani	932478063	SYST
Brooks	473206793	CS
Craig	824529754	ECE
Dopko	378623972	SYST
...	...	...

JOIN

Name	StudentNumber	Major
Amani	932478063	SYST
Brooks	473206793	CS
Craig	824529754	ECE
Dopko	378623972	SYST
...	...	...

StudentNumber	Course	Grade
932478063	SYST202	A-
932478063	CHEM251	B+
473206793	CS211	B
473206793	CHEM251	C
473206793	MATH213	B+
...	...	...

Name	StudentNumber	Major	Course	Grade
Amani	932478063	SYST	SYST202	A-
Amani	932478063	SYST	CHEM251	B+
Brooks	473206793	CS	CS211	B
Brooks	473206793	CS	CHEM251	C
Brooks	473206793	CS	MATH213	B+
...	...	...	...	...

Classification of DBMS Systems
(Date)

Tabular - data viewed as tables

Minimally relational - tabular and supports SELECT, PROJECT and JOIN

Relationally complete - tabular and supports all operations of relational algebra

Fully relational - relationally complete and supports integrity rules

- entity integrity: primary key cannot be null
- referential integrity: if foreign key in A matches primary key in B then all values must match or be null in A

Query Languages

- User accesses and manipulates data using a query language
- SQL
  - Developed at IBM Research Center and incorporated into SQL/DS and DB2
  - Standard SQL became an ANSI standard
  - No commercial DBMS fully obeys standard SQL
    - » ORACLE is based on a dialect of SQL
    - » ACCESS provides a dialect of SQL
- SQL is considered a standard query language for relational DBs but has features not consistent with relational model strictly interpreted
  - NULL values allowed in SQL but not in strict relational model
- A query language is relationally complete if it implements the full relational algebra
  - Practical query languages should be relationally complete
  - Many commercial languages support additional features not expressible in relational algebra

Some SQL Operations

Create relations

Update relations

Define data types

Retrieve records

Sort and group records

Join

Union

Apply aggregate operators (e.g., sum, average)

Typical DBMS Capability

Operations

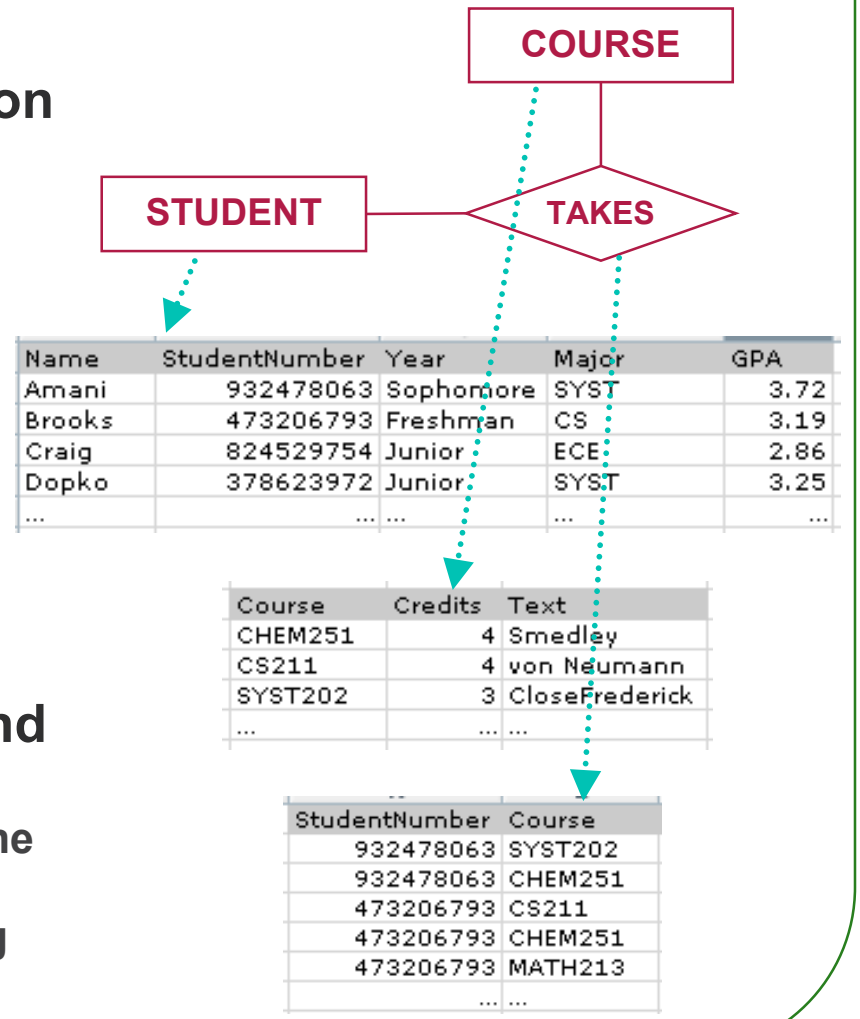
- easy DB creation and update
- report generation (incl. mailmerge)
- views
- query (subset of SQL and/or QBE)
- custom forms (for DB update)
- applications generator
- programming language

Functions

- catalog (sometimes minimal)
- shared update (automatic or programmer lock)
- backups and (limited) journal
- security (password/encryption; levels of access)
- usually minimal integrity checking

ER and Relational Model

- Each entity type in ER model corresponds to an "entity" relation in relational model
- Each relationship in ER model corresponds to "relationship" relation in relational model
  - Foreign keys reference participating entities
- Single-valued attributes in ER model correspond to attributes in relational model
  - E.g., student name
- Multi-valued attributes correspond to relation in relational model
  - E.g. a student can take more than one course
  - Foreign keys reference entity having multi-valued attribute



Name	StudentNumber	Year	Major	GPA
Amani	932478063	Sophomore	SYST	3.72
Brooks	473206793	Freshman	CS	3.19
Craig	824529754	Junior	ECE	2.86
Dopko	378623972	Junior	SYST	3.25
...	...	...	...	...

Course	Credits	Text
CHEM251	4	Smedley
CS211	4	von Neumann
SYST202	3	CloseFrederick
...	...	...

StudentNumber	Course
932478063	SYST202
932478063	CHEM251
473206793	CS211
473206793	CHEM251
473206793	MATH213
...	...

Limits of the Relational Model

Not all information fits naturally into the relational model

Examples:

- Spatial information
- Complex constraints
- Uncertainty about values of attributes (esp. dependencies)
- Subtypes that share some but not all structure and features of parent type
- Context-dependent structure

NULL Value

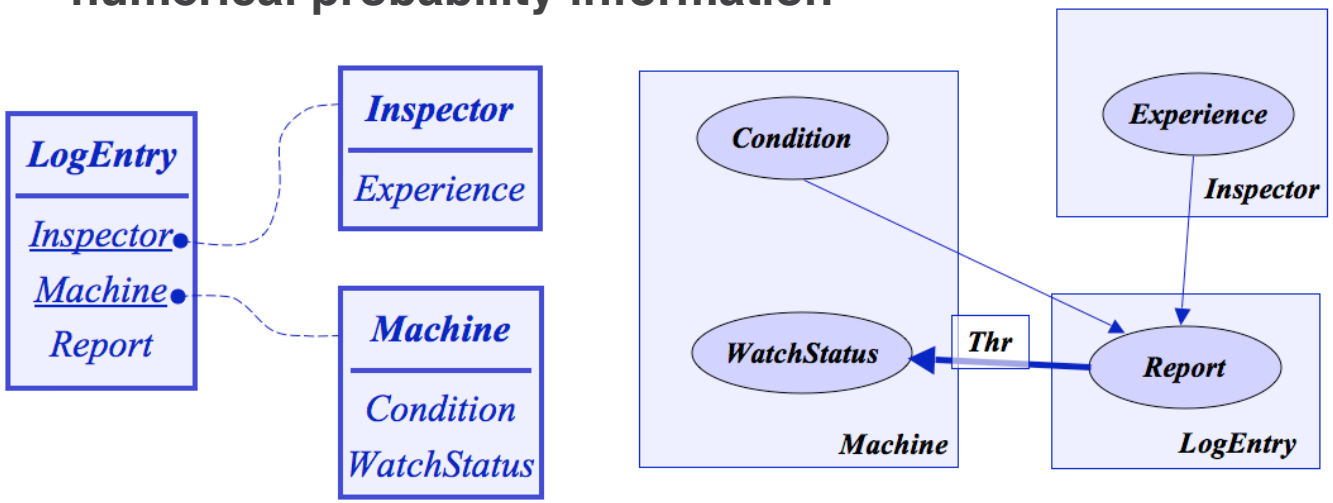
- Strict relational data model has no way to represent unknown
  - Every entry in every table must have a value in the domain of the attribute
  - Theory is based on two-valued logic (true/false)
- SQL and most commercial relational DBs allow special `NULL` entry
  - SQL operators do not treat `NULL` like an ordinary attribute value
  - Behavior of `NULL` conforms to 3-valued logic (true/false/unknown)
    - » Example: Query X=Y? when either X or Y is `NULL` yields `NULL`
- `NULL` is used to represent values that are unknown, not applicable, or not recorded

Uncertainty in Relational Data

- There are many situations in which entries in a DB are missing or may contain errors
  - Non-respondents in surveys
  - Missing entries due to malfunctioning recording equipment
  - Data known to contain errors
- Statistical models can be used to:
  - Represent correlations among attributes values in one or more relations
  - Estimate probability distribution for missing values
  - Evaluate likelihood that a recorded data value is an error
- Special statistical tools are needed for relational data

Probabilistic Relational Models

- Represents uncertainty in relational data
- Elements of PRM
  - *Relational schema* - specifies types of objects & relationships that can exist
  - *Skeleton* - assigns unique ID (primary key) & blank template for each individual entity instance
  - *Data* - Fills entries in blank template
  - *PRM structure* - represents probabilistic dependencies and numerical probability information



Object Model

- **Abstract data types**
  - Data types describe set of objects with same characteristics
  - Individual entities are modeled as instances of a data type
  - Manipulation of instances is done through operations associated with data type (methods, messages)
- **Inheritance**
  - Instances inherit properties of class
  - Classes arranged in hierarchy and inherit properties of parent class
  - Multiple inheritance (need to manage clashes)
  - Inheriting behavior: code sharing
  - Inheriting representation: structure sharing
- **Object identity**
  - Each object instance has unique identity
  - Objects can contain or refer to other objects
- **Encapsulation**
  - Objects have public and private attributes
  - Private attributes are *encapsulated* -- not visible from outside

Example

- Class `Salesperson` represents abstract properties of salesperson instances
- Attributes of `Salesperson` instances (examples):
  - `Name`
  - `LengthOfService`
  - `Salary`
  - `Accounts`
  - `Orders`
  - `Quota`
- Methods (examples)
  - `AddNewAccount`
  - `ChangeQuota`
  - `GiveRaise`
- Class `Salesperson` inherits structure (attributes) and behavior (methods) from class `Employee`

Advantages of Abstract Data Typing

- **Better conceptualization and modeling of the real world**
  - enhance representation and understandability
  - categorize objects based on common structure & behavior
- **Enhances robustness of system**
  - type checking to avoid run-time errors
  - integrity checks on data and operations
- **Enhances performance (potentially!)**
  - allows compile-time optimization
  - permits better clustering strategies for persistent objects
- **Captures semantics of type**
  - localizes operations and representation of attributes
- **Separates implementation from specification**
  - change implementationn without changing public interface
- **Extensibility due to reusability**

Comparing OO with Relational Model

Advantages of OO:

- Expressiveness
- Modularity
- Ability to model generalization / specialization hierarchies

Disadvantages of OO:

- Formal theory is less well developed than relational theory
- Standards less well-developed
- Fewer 3rd party applications
- *All these disadvantages are diminishing*

Differences:

- OO emphasizes *navigational* access; RM emphasizes *declarative* access
- OO and relational technology each appear to be more efficient for certain kinds of problem

Integrating Objects and Relations

- Many vendors offer limited object functionality & claim to support an "extended" object/relational model
- Many object-oriented ideas incorporated into SQL:1999
- Date (2001) argues that object functionality properly implemented is fully consistent with relational model

Some Commercial OODB Products

- GemStone
- JADE
- Objectivity/DB
- ObjectDB
- Versant (merged with POET)

http://www.service-architecture.com/products/object-oriented_databases.html

Distributed Databases

- **Shift from Mainframes to PCs**
  - 1960's - Computer means mainframe
  - 1970's - Minicomputers (esp. DEC VAX running UNIX)
  - 1980's - Personal computers
  - 1990's - Computer on every desktop
  - 2000+ - Everyone on LAN and Internet
- **Networking evolution**
  - Standalone PCs --> connection to LANs
  - File servers enabled data sharing
  - Distributed databases allowed multiple users at different sites to access common DB
  - Current trend: heterogeneous distributed databases
    - » Different vendors
    - » Different schemas & ontologies
    - » Different data models (e.g., relational and object-oriented)

File Servers and Database Servers

- File servers connect workstations on LAN
  - Run network OS
  - Provide archival storage for each workstation
  - Provide repositories for shared files
  - Interface with outside world - run email and web server software
  - Run printing services
  - Manage security
- Database server
  - Client sends high-level query
  - Database server processes query and returns only results
  - Reduced network traffic in contrast to standard block-IO required for client-side query processing

Database Server Responsibilities

- Functions typically performed on server
  - Integrity checks
  - Security
  - Recovery
  - Concurrency control
  - Method/query execution and optimization
  - Clustering and indexing of persistent object collections
- Libraries and tools typically stored on server
  - Graphical user interface building tools
  - Application programming interface libraries
    - » Allows user application to interface directly with DBMS to store and perform queries without going through DBMS HCI

Distributed Databases

- **Client sees one logical database that can consist of many physical databases distributed across server nodes on a network**
- **Strategies**
  - ***Remote request***: **local client opens session with remote DB, submits request, and receives results**
  - ***Remote updates***: **local client can update remote DB objects**
  - ***Remote unit-of-work***: **local client can submit transaction involving both reads and updates**
  - ***Distributed request***: **users can execute distributed transactions involving reads and writes from multiple distributed databases**

Databases and the Web

- Worldwide web provides access to files on Internet
  - Addressed by URL (uniform resource locator)
- Simple web site
  - URL points to file that is accessed by browser
  - HTML tags tell browser how to display document (formatting, images, etc.)
  - Plugins give added functionality (sounds, video clips)
- Web services
  - True interactivity requires more than just accessing a file and displaying contents
  - URL provides entry point to services provided by web site
  - User provides information that describes desired services
  - Information sent as parameters to program executing on server
  - Common gateway interface (CGI) protocol allows dynamic creation of html document to display to user
  - Program often includes database calls

XML

- **Markup Language:** language that combines text with additional information about the text
- **HTML:** hypertext markup language
  - Adequate for representing instructions for displaying documents and linking to other documents
  - Inadequate for representing abstract data types (no semantics)
  - Fixed tags with built-in meaning (e.g., <TITLE>, </TITLE>)
- **XML:** extensible markup language
  - Markup language for creating new markup languages
    - » Allows user to define new collection of tags with user-defined meaning and rules for use
    - » Can structure any type of document
  - XML can represent
    - » Elements (also called tags) - primary building blocks of document
    - » Attributes - further describe elements
    - » Entities and references - allow references to external files
  - Rapidly becoming standard file and interchange format
  - Standardized and maintained by Worldwide Web Consortium (W3C)

Query Languages for XML Data

- XML has grown rapidly in popularity
  - Growing numbers of documents on Web are represented in XML
  - XML is becoming common for exchanging information among applications and databases
- XML can represent the information content of:
  - structured and semi-structured documents
  - relational databases
  - object repositories
- An XML aware query language can query data expressed directly in XML or translated into XML
- XML Query Languages
  - XML-QL published as W3C submission in 1998
  - Grew into XQuery
  - W3C released XQuery 1.0 candidate recommendation in June 2006

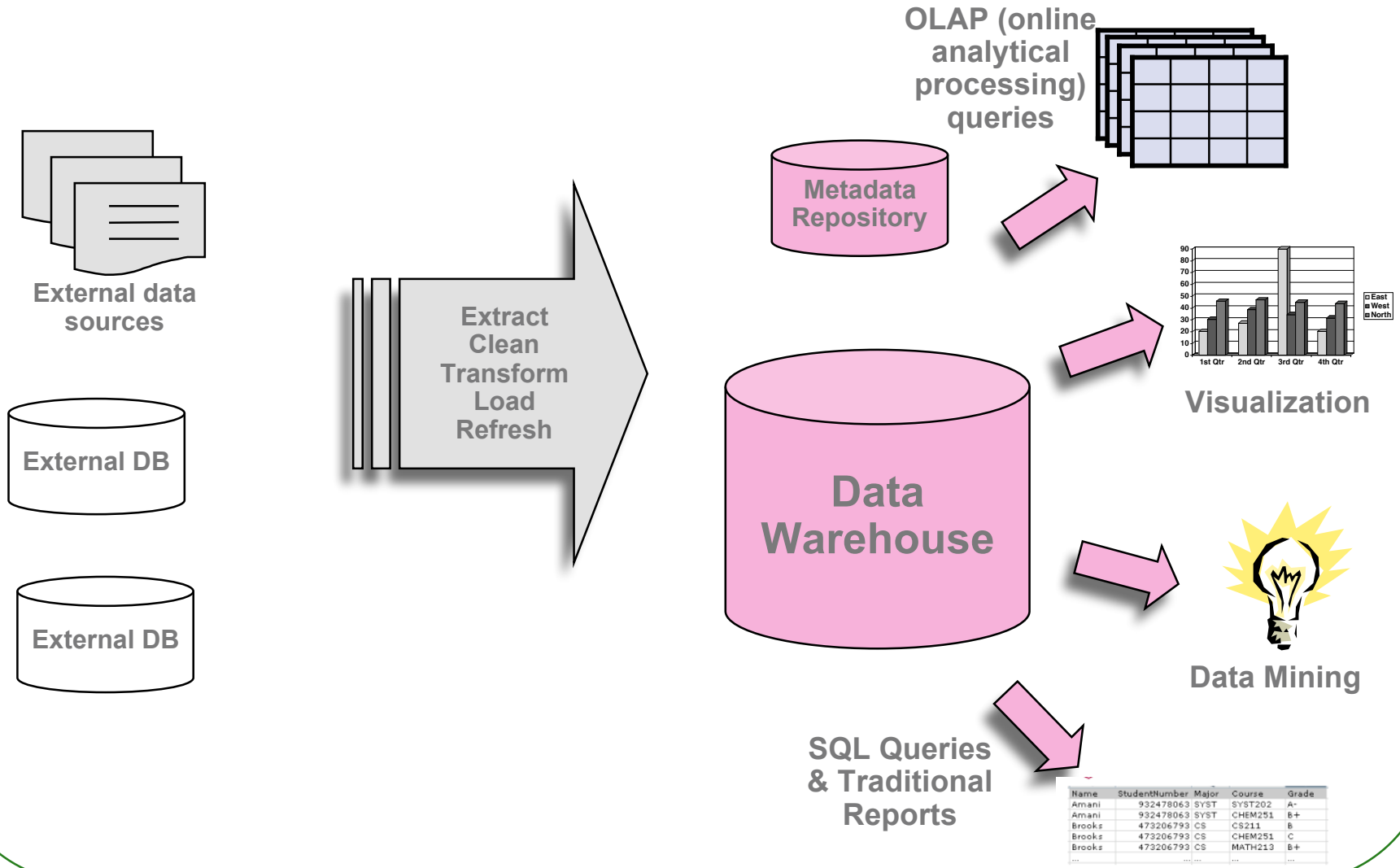
Metadata

- Metadata is data about data
- Describes what is in a database so that applications can determine whether and how to access the data
  - Attributes
  - Formats
  - Units
  - …
- Metadata is key to:
  - Data interchange
  - Effective and efficient use of distributed databases
  - Managing data warehouses
  - Web services
- Populating metadata repositories is a major challenge

Data Warehouse

- **Contains consolidated data from multiple sources**
  - Centralized schema
  - Usually cleansed
  - Must be refreshed when contributing data stores change
- **Advantages**
  - Solution to data integration
  - May be higher quality than individual data stores (can serve as cleansed corporate data store)
- **Difficulties**
  - Difficulty of getting everyone to agree on common schema
  - Tedious to create and maintain
  - Can have longer latency than individual data stores
  - Can go out of date
  - If schema is inflexible the data warehouse can become just another legacy system

Data Warehouse Architecture



Knowledge Discovery & Data Mining

- **Phases of knowledge discovery**
  - Data selection
  - Data cleansing
  - Enrichment
  - Transformation or encoding
  - Data Mining
  - Reporting and display
- **Data mining searches a database to discover patterns or rules**
  - Use to predict, describe, classify
  - Interface between databases, statistics, and artificial intelligence
- ***Should data mining be considered part of the model or data subsystem?***

Security

- Data must be protected from access by unauthorized persons
- Some approaches to security
  - Set limits on data that can be collected & stored
  - Build separate DB on isolated machine
  - Use encryption and password protection to protect against unauthorized access
  - Provide different levels of access for different users (e.g., role-based access control, or RBAC)
- Even the best security system can be breached
- Multi-layered protection combines prevention (e.g., access control, encryption) with mitigation (e.g., detection of violations, response to violations)
- There is increasing concern with *insider threat*
  - Trusted insiders have more privileges and can cause greater damage than intrusions by outsiders
  - Protection against violations by insiders is often weak or nonexistent

What Access to Allow?

- **Inference from multiple sources**
  - It is possible to piece together bits of information to infer information that is not represented explicitly in any of the individual sources
  - Enabling access to individually innocuous data can cause breach of security and/or violation of privacy
- **Security versus functionality**
  - Too strict security policies can inhibit workflow
  - Implementing a security policy that enables productive work while preventing unauthorized access is a difficult challenge

Security and DSS Design

- **Questions to ask:**
  - Who needs to have access to what information?
  - Who should *not* have access to what information?
  - What are the consequences of unauthorized access?
  - What laws, regulations and organizational policies govern access to data?
  - Are there ethical issues regarding access to data?
  - What would be the implications of public disclosure of security breaches and/or inadequate security protection?
- **DSS designers need to consult with security personnel to understand policies and their implications for DSS design**

Importance of Data Management for DSS

Inadequate
- data availability
- data management
- interfaces to existing data sources

is a common cause of DSS failure

Data Strategies for DSS

- Obtain data from existing database (or data warehouse)
- Database (or data warehouse) developed for DSS
- User inputs data at runtime

Advantages of Using Pre-existing Database
(if one exists)

1. Simplifies collection and maintenance of data used by the DSS
2. Limits the set of functions and users that the DSS needs to support
3. Simplifies the design of the DSS
4. Eliminates potential conflicting performance and security requirements (update versus retrieval)
5. Increases chances of data sharing among different DSS

Sprague and Carlson,1982

Database Development Options

- Use existing general-purpose DB or data warehouse
  - compatibility & interfaces
  - runtime speed
  - security
- Develop special-purpose DB
  - development & maintenance cost
  - redundancy; consistency
- Extraction (aggregation & subsetting)
  - Currency
  - Duplication
  - Conflict

Issues for DSS Data Management

- **Data representation** - efficiency of storage and appropriateness for modeling
- **Data conversion** - compatibility, consistency, currency
- **Data access** - protection, confidentiality
- **Data sharing** - coordination

Database Requirements for DSS

(Sprague and Carlson, 1982)

- **Memory aids** - save intermediate results; create links among data; reminding
- **Data reduction**
- **Varying levels of detail**
- **Varying amounts of data**
- **Multiple sources**
- **Catalog of sources**
- **Wide time frame** - past data and future projections
- **Public and private databases**
- **Varying degrees of accuracy**
- **Set operations**
- **Random access**
- **Support for relationships and views**
- **Performance** - response time relative to problem requirements and decision maker expectations
- **Interface to other DSS components**
- **End user interface**

In Summary...

References for Unit 6

Blaha, M.R. and Premeriani, W. *Object-Oriented Modeling and Design for Database Applications*, Prentice-Hall, 1999.

Date, C.J. *The Database Relational Model: A Retrospective Review and Analysis*. Addison Wesley, 2001.

Date, C.J. *An Introduction to Data Base Systems* Reading, MA: Addison-Wesley, Vol.I 4th Edition 1985, Vol. II, 1983.

Elmasri, R. and Navathe, S. *Fundamentals of Database Systems.* Addison-Wesley, 2000.

Harold, E.R. and Means, W.S. *XML in a Nutshell* (3rd edition). O'Reilly Media, 2004

Kemper, A. and Moerkotte, G. *Object-Oriented Database Management* Prentice-Hall, 1994.

McFarland, Gregory, Rudmik, A. and Lange, D. *Object-Oriented Database Management Systems Revisited: An Updated DACS State of the Art Report,* AFRL/IF, 1999. http://www.dacs.dtic.mil/techs/oodbms2/

Sprague, R.H. and Carlson, E.D. *Building Effective Decision Support Systems*. Englewood Cliffs, NJ: Prentice Hall, 1982.