Introduction to Databases CS348

University of Waterloo

Winter 2007

Course Outline

Why do we use Databases?

- Functionality provided by a DBMS.
- Models of databases: Network, Relational, OO.

How do we use a DBMS?

- Relational model, logic-based query languages.
- SQL
- Embedding of query languages and applications.
- Relational algebra and query execution.
- Transactions and Recovery.

How do we design databases?

- Data Modeling: The Entity-Relationship (ER) model.
- Data anomalies and Normal forms.
- Basic physical design.

What is a Database?

Definition (Database)

A *large* and *persistent* collection of (more-or-less similar) pieces of information organized in a way that facilitates safe and efficient *retrieval* and *modification*.

- a file cabinet
- a library system
- a personnel management system
- ...

Definition (Database Management System (DBMS))

A program (or set of procedures) that does the above.

Application of Databases

Original

- inventory control
- payroll
- banking and financial systems
- reservation systems

More recent

- computer aided design (CAD)
- software development (CASE)
- telecommunication systems
- e-commerce

Common Circumstances

- There is lots of data (mass storage)
- Data is formatted (lots of *similar records*)
- . . . and important
  - ∗ persistence and reliability
  - ∗ efficient and concurrent access

Note

the data maintained by the system are much much more important and valuable then the system itself.

Issues:

- many files with different structure
- shared files or replicated data
- need to exchange data (translation programs)

Common Circumstances

- There is lots of data (mass storage)
- Data is formatted (lots of *similar records*)
- ...and important
  - persistence and reliability
  - efficient and concurrent access

Note

the data maintained by the system are much much more important and valuable then the system itself.

Issues:

- many files with different structure
- shared files or replicated data
- need to exchange data (translation programs)

Database Management Systems

Idea

Abstracts common functions and creates a uniform well defined interface for applications accessing data.

1. Uniform Data Model
   all data stored in a well defined way
2. Access control
   only authorized people get to see/modify it
3. Concurrency control
   manage concurrent access the database
4. Database recovery
   nothing gets accidentally lost
5. Database maintenance

Data Independence

Idea

Applications do not access data directly but, rather through an abstract data model *provided by the DBMS.*

Indirect access supports:

- advanced data structures
- data restructuring
- distribution and load balancing
- ...

all without changes to applications.

Three Level Schema Architecture

Definition (Schema)

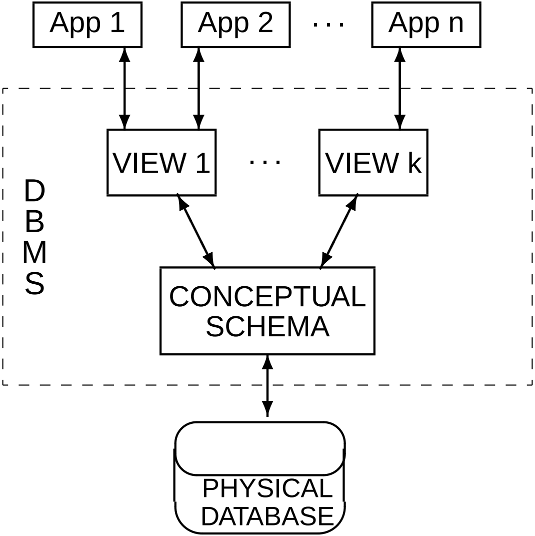
A **schema** is a description of the data interface to the database (i.e., how the data is organized).

1. External schema (view): what the application programs and user see. May differ for different users of the same database.
2. Conceptual schema: description of the logical structure of *all* data in the database.
3. Physical schema: description of physical aspects (selection of files, devices, storage algorithms, etc.)

Definition (Instance)

A **database instance** is a database (real data) that conforms to a given schema.

Three-level Schema Architecture (cont.)



Transactions

Idea

Every application may think they are the sole application accessing the data. The DBMS guarantees correct execution.

Example:

```
withdraw(AC,1000)           withdraw(AC,500)
  Bal := getbal(AC)
                              Bal := getbal(AC)
  if (Bal>1000)               if (Bal>500)
    <give-money>                <give-money>
    setbal(AC,Bal-1000)
                                setbal(AC,Bal-500)
```

Recovery

Idea

Whenever the DBMS acknowledges a data item was stored in the database, then it won't disappear due to crash/power failure/. . .

- Synchronous writes to permanent storage?
  - too inefficient
  - buffer cache (not completely safe)
  - what needs to be written and in what order
- What happens after, e.g., power failure?
  - logs and recovery
  - restarting transactions

Users of DBMS

Three types of users:

1. end user/application developer
   - $\Rightarrow$ queries/updates data in a database or
   - $\Rightarrow$ develops applications for doing that
2. database administrator (DBA)
   - $\Rightarrow$ designs/maintains database schema
   - $\Rightarrow$ decides on physical layout of the DB
   - $\Rightarrow$ monitors and tunes DBMS performance
   - $\Rightarrow$ sets access policies
3. database implementor (e.g., at IBM)
   - $\Rightarrow$ implements a (parts of) a DBMS

A Brief History of DBMS: 1960s and 1970s

- hierarchical data model
  - IBM's Information Management System (IMS)
  - data organized as a tree
  - edges (pointers) represent relationships
  - access by following (next) pointers
  - queries are difficult to write
  - low level of data independence
  - COBOL
- Conference On DAta Systems Languages (CODASYL) model, a.k.a. the network model
- Codd's relational data model proposal (1970) and ensuing debate
- Chen's E-R model (1976)
- Initial development of transaction concepts
- IBM's System R and UC Berkeley's Ingres systems demonstrate feasibility of relational DBMS (late 1970s)

A Brief History of DBMS: 1980s

- development and deployment of commercial relational technology
  - DB2 (IBM)
  - Informix
  - Oracle
  - Sybase
- SQL standardization efforts through ANSI and ISO
- object-oriented database management systems
  - persistent objects
  - object id's, methods, inheritance
  - navigational interface reminiscent of the network model

A Brief History of DBMS: 1990s-present

- continued expansion of SQL standard and DBMS feature sets
- new application domains, e.g., On-Line Analytic Processing (OLAP), data warehousing
- object-relational capabilities
  - user-defined types within relational framework
- World Wide Web
  - information retrieval (keywords, ranked results)
  - eXtensible Markup Language (XML), XPath, XQuery
  - semantic web (RDF, OWL, . . .)

Relational Databases

Idea

all information is organized in (flat) relations. This provides an abstract interface to the stored data.

- elements (data items): without preassigned meaning
- A **relation** is a set of records of data items
  - information is captured **solely** by membership in relations
  - the database *doesn't understand* the data, it has to be interpreted by the user
- A **database** is a (finite) set of (finite) relations.

A Relation

Department

DeptNo	DeptName	MgrNo	AdmrDept
A00	Planning	000020	A00
E01	Support Services	000050	A00
E11	Operations	000090	E01
E21	Software Support	000100	E01

Schema and Instance

- The schema of a relational database includes the schemas of its relations.
- The schema of a relation includes the relation's name, the names of its attributes, and their associated domains.
- Schemas may also include additional integrity constraints, which constrain possible the database instances.
- An instance of a relation is a set of tuples conforming to the schema.
- A database instance is a set of relation instances conforming to the schema.

Constraints

- a *constraint* is a rule that restricts the tuples that may appear in a database instance
- common examples: primary key constraints, foreign key constraints
  - a primary key constraint for a relation R specifies a set of attributes of R whose values can be used to uniquely identify any tuple in R, i.e., no two tuples in R can have the same values for the key attribute(s)
  - a foreign key constraint specifies that values found in foreign key columns in a *referencing* relation R_1 must appear as primary keys in a *referenced* relation R_2

Relational Languages

Idea

- *Relations describe what items are related.*
- *Queries are questions about such relationships.*

- Features:
  - independent (logic based) semantics
  - closure: results of queries are relations
  - different ways to implement queries
  - query optimization
- data definition and modification have similar features