Database Systems

Lecture 2: Introduction

Dr. Momtazi
momtazi@aut.ac.ir

based on the slides of the course book

Outline

- **Schemas and Instances**
- Data Definition Language and Data Manipulation Language
- Database Design
- Database Engine
- Database Architecture
- Data Mining and Information Retrieval
- Data Models
- History of Database Systems

Schemas and Instances

- Similar to types and variables in programming languages

- **Schema**
  - **Physical schema**– the overall physical structure of the database
  - **Logical Schema** – the overall logical structure of the database
    - Example: The database consists of information about a set of customers and accounts in a bank and the relationship between them
    - Analogous to type information of a variable in a program

- **Instance** – the actual content of the database at a particular point in time
  - Analogous to the value of a variable

Instances and Schemas

- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
  - Applications depend on the logical schema
  - In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others.

Outline

- Schemas and Instances
- **Data Definition Language and Data Manipulation Language**
- Database Design
- Database Engine
- Database Architecture
- Data Mining and Information Retrieval
- Data Models
- History of Database Systems

Data Definition Language (DDL)

- Specification notation for defining the database schema
- DDL compiler generates a set of table templates stored in a ***data dictionary***
- Data dictionary contains metadata (i.e., data about data)
- A special type of table that can only be accessed and updated by the database system itself (not a regular user)

Data Definition Language (DDL)

- The data values stored in the database must satisfy certain consistency constraints

- Database systems implement these constraints that can be tested with minimal overhead
  - Domain Constraints
  - Referential Integrity
  - Assertions
  - Authorization

Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
  - DML also known as query language

- Different types of access:
  - Retrieval of information stored in the database
  - Insertion of new information into the database
  - Deletion of information from the database
  - Modification of information stored in the database

Data Manipulation Language (DML)

- DML types:
  - Procedural DMLs require a user to specify what data are needed and how to get those data
  - Declarative DMLs (nonprocedural DMLs) require a user to specify what data are needed without specifying how to get those data

- Two classes of languages
  - **Pure** – used for proving properties about computational power and for optimization
    - Relational Algebra
    - Tuple relational calculus
    - Domain relational calculus
  - **Commercial** – used in commercial systems
    - SQL is the most widely used commercial language

SQL

- The most widely used commercial language
- To be able to compute complex functions SQL is usually embedded in some higher-level language
- Application programs generally access databases through one of
  - Language extensions to allow embedded SQL
  - Application program interface (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database

DDL and DML Examples

- DDL

  create table *instructor* (

  ID **char**(5),
  name **varchar**(20)**,**
  dept_name **varchar**(20),
  salary **numeric**(8,2))

- DML

  select *instructor.name*

  from *instructor*

  where *instrctor.dept_name* = ’History’;

  select *instructor. ID* , *department.dept_name*

  from *instructor*, *department*

  where *instructor.dept_name* = *department.dept_name*

  and *department.budget* > 95000;

Outline

- Schemas and Instances
- Data Definition Language and Data Manipulation Language
- **Database Design**
- Database Engine
- Database Architecture
- Data Mining and Information Retrieval
- Data Models
- History of Database Systems

Database Design

The process of designing the general structure of the database:

- Logical Design – Deciding on the database schema. Database design requires that we find a “good” collection of relation schemas.
  - Business decision – What attributes should we record in the database?
  - Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?
- Physical Design – Deciding on the physical layout of the database

Database Design

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

Database Design

- Is there any problem with this relation?

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Design Approaches

- Need to come up with a methodology to ensure that each of the relations in the database is “good”

- Two ways of doing so:
  - Entity Relationship Model (Chapter 6)
    - Models an enterprise as a collection of *entities* and *relationships*
    - Represented diagrammatically by an *entity-relationship diagram*
  - Normalization Theory (Chapter 7)
    - Formalize what designs are bad, and test for them

Design Approaches

- Specification of functional requirements
  - Users describe the kinds of operations (or transactions) that will be performed on the data
    - Modifying or updating data
    - Searching for and retrieving specific data
    - Deleting data
  - The designer can review the schema to ensure it meets functional requirements

Database Design for a University Organization

- General description
  - The university is organized into departments. Each department is identified by a unique name (dept_name), is located in a particular building, and has a budget.
  - Each department has a list of courses it offers. Each course has associated with a course id, title, dept_name, and credits, and may also have associated prerequisites.
  - Instructors are identified by their unique ID . Each instructor has name, associated department (dept_name), and salary.
  - Students are identified by their unique ID . Each student has a name, an associated major department (dept_name), and tot_cred (total credit hours the student earned thus far)

Database Design for a University Organization

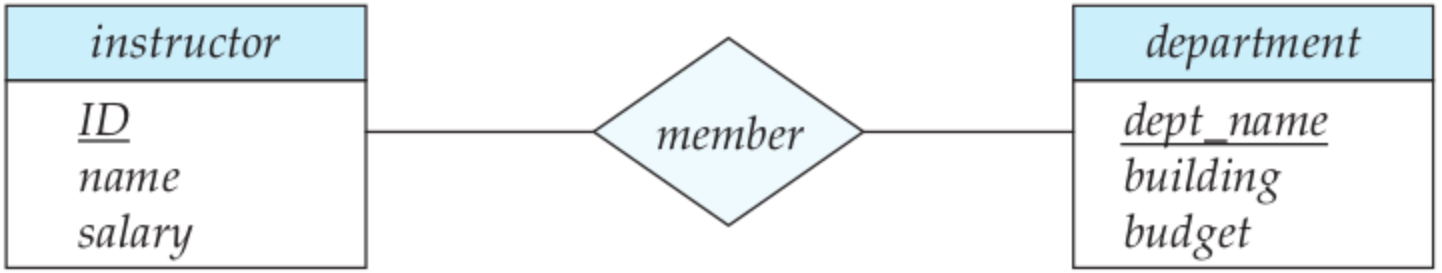
- General description (cont.)
  - The university maintains a list of classrooms, specifying the name of the building, room number, and room capacity.
  - The university maintains a list of all classes (sections) taught. Each section is identified by a course_id, sec_id, year, and semester, and has associated with it a semester, year, building, room_number, and time_slot_id (the time slot when the class meets).
  - The department has a list of teaching assignments specifying, for each instructor, the sections the instructor is teaching.
  - The university has a list of all student course registrations, specifying, for each student, the courses and the associated sections that the student has taken (registered for).

The Entity-Relationship Model

- Uses a collection of
  - Entities (basic objects)
  - Relationships among these objects
- Entities are described in a database by a set of attributes
- Example:
  - Entity: instructor
  - Attributes: ID , name, and salary
    - The extra attribute ID is used to identify an instructor uniquely (since it may be possible to have two instructors with the same name and the same salary).
  - Relationship: member (associates an instructor with his/her department)

The Entity-Relationship Model

- Entity-relationship (E-R) diagram expresses the overall logical structure (schema) of a database
  - Unified Modeling Language (UML)



Normalization

- Goal:
  - Designing schemas that are in an appropriate normal form.

- Generating a set of relation schemas that allows us
  - Storing information without unnecessary redundancy
  - Retrieving information easily

Database Design

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

(a) The *instructor* table

dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

Database Design

- Is there any problem with this relation?

ID	*name*	*salary*	*dept_name*	*building*	*budget*
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califieri	62000	History	Painter	50000
83821	Brandt	92000	Comp. Sci	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

Normalization

- Problems of a bad design
  - Repetition of information
  - Inability to represent certain information

Outline

- Schemas and Instances
- Data Definition Language and Data Manipulation Language
- Database Design
- **Database Engine**
- Database Architecture
- Data Mining and Information Retrieval
- Data Models
- History of Database Systems

Database Engine

- Storage manager
- Query processing
- Transaction manager

Storage Management

- **Storage manager** is a program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.

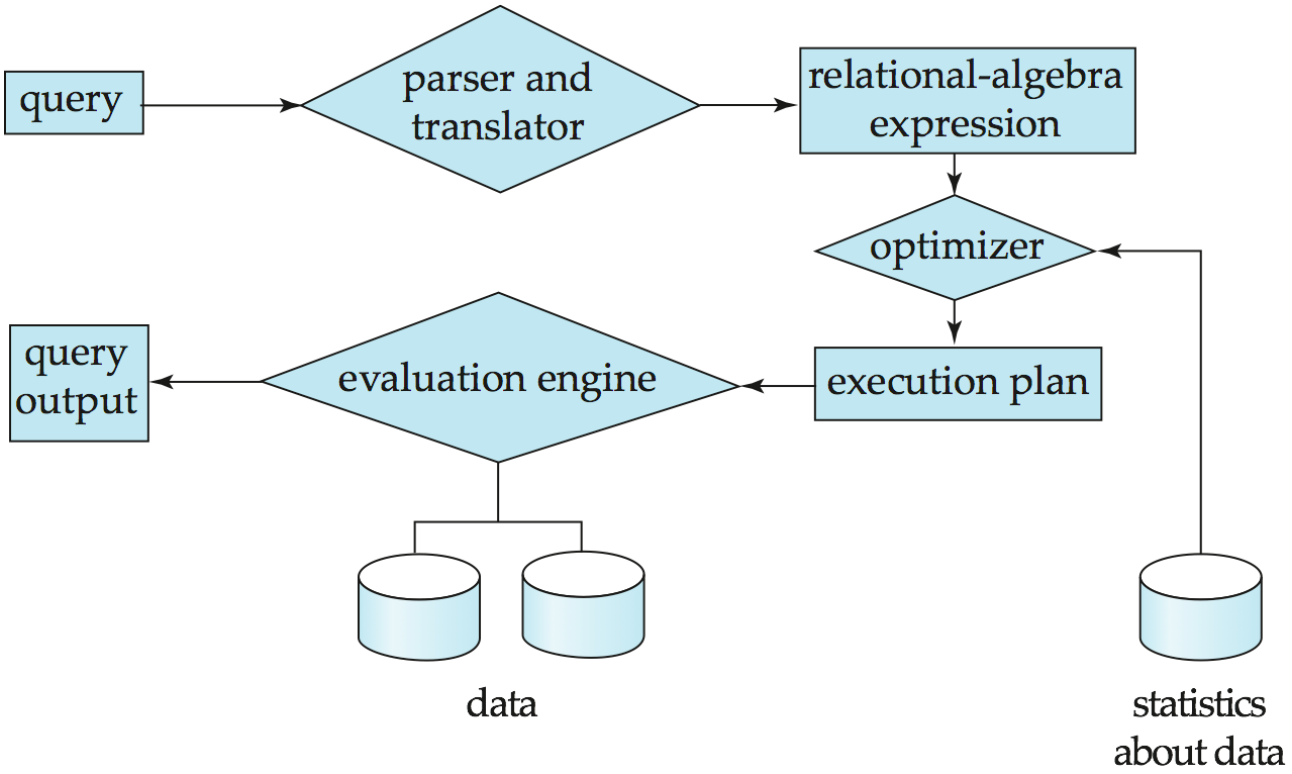
- The storage manager is responsible to the following tasks:
  - Interaction with the OS file manager
  - Efficient storing, retrieving and updating of data

Storage Management

- The storage manager components include:
  - Authorization and integrity manager
  - Transaction manager
  - File manager
  - Buffer manager
- Data structures:
  - Data files
  - Data dictionary
  - Indices

Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



Query Processing

- Cost difference between a good and a bad way of evaluating a query can be enormous

- Need to estimate the cost of operations
  - Depends critically on statistical information about relations which the database must maintain
  - Need to estimate statistics for intermediate results to compute cost of complex expressions

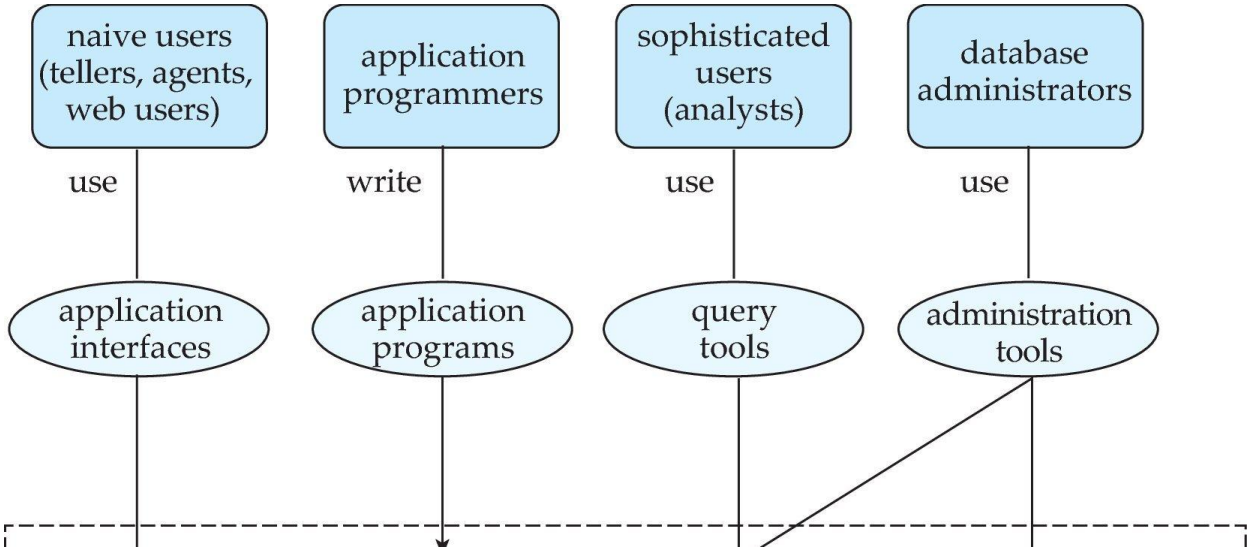
Transaction Management

- What if the system fails?
- What if more than one user is concurrently updating the same data?
- A **transaction** is a collection of operations that performs a single logical function in a database application
- **Transaction-management component** ensures that the database remains in a consistent (correct) state despite system failures (e.g., power failures and operating system crashes) and transaction failures.
  - Recovery manager detects system failures and restore the database to the state that existed prior to the occurrence of the failure.
  - Concurrency-control manager controls the interaction among the concurrent transactions, to ensure the consistency of the database.

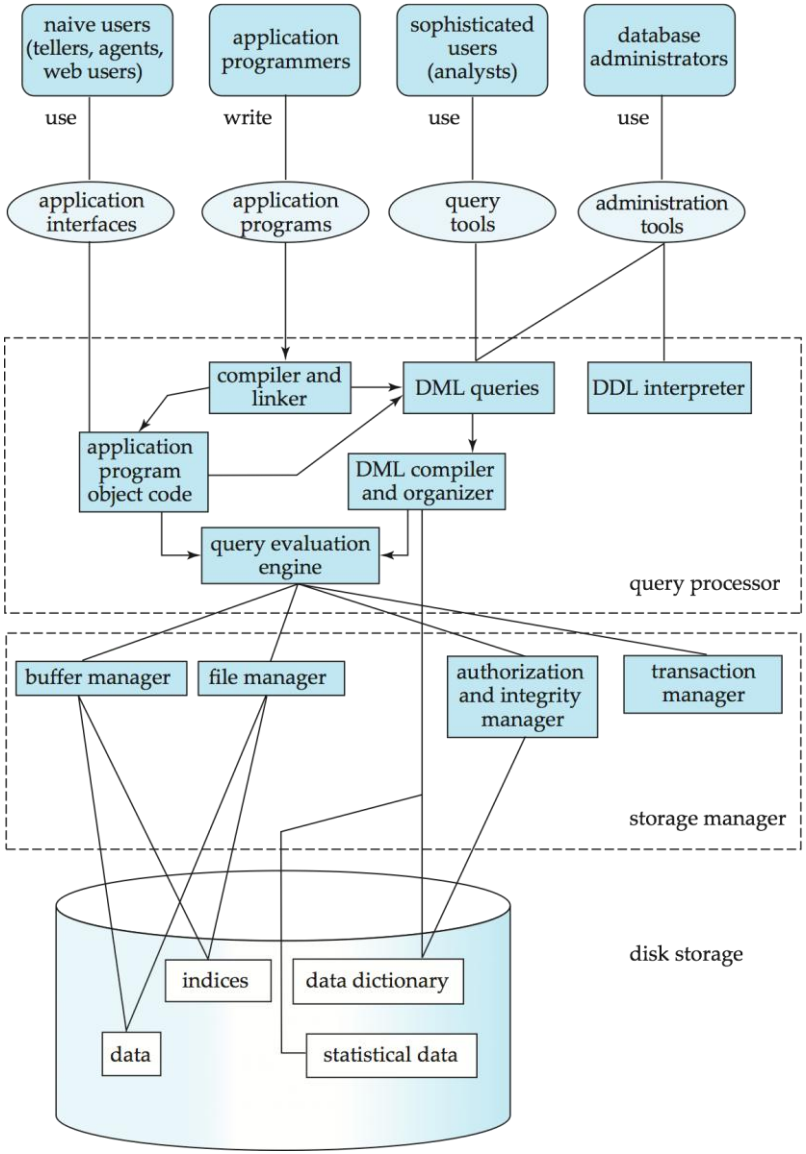
Outline

- Schemas and Instances
- Data Definition Language and Data Manipulation Language
- Database Design
- Database Engine
- **Database Architecture**
- Data Mining and Information Retrieval
- Data Models
- History of Database Systems

Database Users and Administrators



Database System Internals



Database Architecture

- The architecture of a database systems is greatly influenced by the underlying computer system on which the database is running:
  - Centralized
  - Client-server
  - Parallel (multi-processor)
  - Distributed

Outline

- Schemas and Instances
- Data Definition Language and Data Manipulation Language
- Database Design
- Database Engine
- Database Architecture
- **Data Mining and Information Retrieval**
- Data Models
- History of Database Systems

Data Mining and Information Retrieval

- **Data mining**: the process of semi-automatically analyzing large databases to find useful patterns.
- Also known as “knowledge discovery in databases”

- **Information retrieval**: querying of unstructured textual data
- Information retrieval systems have much in common with database systems—in particular, the storage and retrieval of data on secondary storage

Outline

- Schemas and Instances
- Data Definition Language and Data Manipulation Language
- Database Design
- Database Engine
- Database Architecture
- Data Mining and Information Retrieval
- **Data Models**
- History of Database Systems

Data Models

- Complex data types
  - JSON
  - XML
  - RDF
  - Object-Based Data Models

Object-Relational Data Models

- Relational model: flat, “atomic” values
- Object Relational Data Models
  - Extend the relational data model by including object orientation and constructs to deal with added data types.
  - Allow attributes of tuples to have complex types, including non-atomic values such as nested relations.
  - Preserve relational foundations, in particular the declarative access to data, while extending modeling power.
  - Provide upward compatibility with existing relational languages.

Outline

- Schemas and Instances
- Data Definition Language and Data Manipulation Language
- Database Design
- Database Engine
- Database Architecture
- Data Mining and Information Retrieval
- Data Models
- **History of Database Systems**

History of Database Systems

- 1950s and early 1960s:
  - Data processing using magnetic tapes for storage
    - Tapes provided only sequential access
  - Punched cards for input
- Late 1960s and 1970s:
  - Hard disks allowed direct access to data
  - Network and hierarchical data models in widespread use
  - Ted Codd defines the relational data model
  - High-performance (for the era) transaction processing
- 1980s:
  - Research relational prototypes evolve into commercial systems
    - SQL becomes industrial standard
  - Parallel and distributed database systems
  - Object-oriented database systems

History of Database Systems

- 1990s:
  - Large decision support and data-mining applications
  - Large multi-terabyte data warehouses
  - Emergence of Web commerce
- Early 2000s:
  - XML and XQuery standards
  - Automated database administration
- Later 2000s:
  - Giant data storage systems
    - Google BigTable, Yahoo PNuts, Amazon, ..

Questions?