Database Principles and Design
(数据库原理与设计)

Xiaolin Zhao

Chapter 1:
The Worlds of Database Systems

Outline

- Why do we need a DBMS (Database Management System)?
- What can a DBMS do for an application?
- Why study database systems?
- Data Models: Overview of a Relational Model
- Levels of Abstraction in a DBMS
- Sample Queries in DBMS
- Transaction Management Overview
- Structure of a DBMS

Why DBMS?

- Suppose that you want to build an university database. It must store the following information:
  - Entities: Students, Professors, Classes, Classrooms
  - Relationships: Who teaches what? Who teaches where? Who teaches whom?

What can DBMS do for applications?

- Store huge amount of data (e.g., TB+) over a long period of time
- Allow apps to query and update data
  - Query: what is Mary's grade in the "Operating System" course?
  - Update: enroll Mary in the "Database" course
- Protect from unauthorized access.
  - Students cannot change their course grades.
- Protect from system crashes
  - When some system components fail (hard drive, network, etc.), database can be restored to a good state.

More on what can DBMS do for applications?

- Protect from incorrect inputs
  - Mary has registered for 100 courses
- Support concurrent access from multiple users
  - 1000 students using the registration system at the same time
- Allow administrators to easily change data schema
  - At a later time, add TA info to courses.
- Efficient database operations
  - Search for students with 5 highest GPAs

Alternative to Using a DBMS

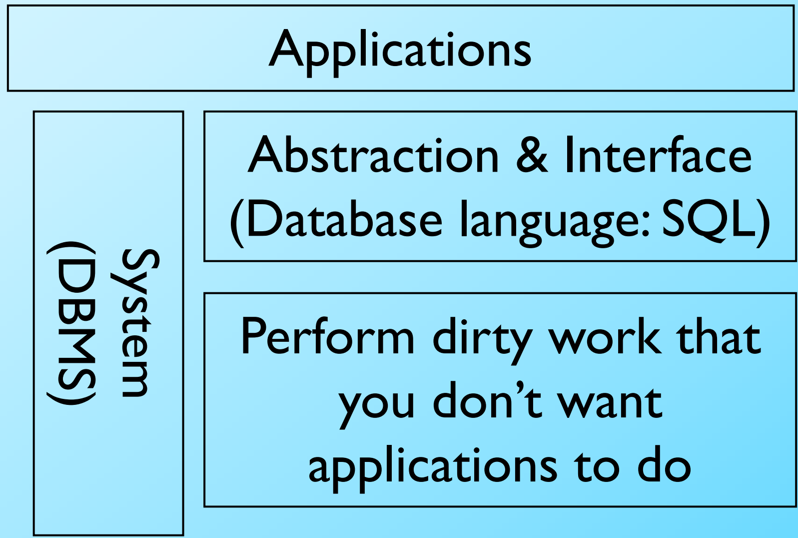
- Store data as files in operating systems.
- Applications have to deal with the following issues:
  - 32-bit addressing (4GB) is insufficient to address 100GB+ data file
  - Write special code to support different queries
  - Write special code to protect data from concurrent access
  - Write special code to protect against system crashes
  - Optimize applications for efficient access and query
  - May often rewrite applications
- Easier to buy a DBMS to handle these issues

Database Management System (DBMS)

- DBMS is software to store and manage data, so applications don’t have to worry about them.
- What can a DBMS do for applications?
  - Can you think of them?

What can a DBMS do for applications?

- Define data: Data Definition Language (DDL)
- Access and operate on data: Data Manipulation Language (DML)
  - Query language
- Storage management
- Transaction Management
  - Concurrency control
  - Crash recovery
- Provide good security, efficiency, and scalability



Why Study Database Systems?

- They are everywhere.
  - Online stores, real stores
  - Banks, credit card companies
  - Passport control
  - Police (criminal records)
  - Airlines and hotels (reservations)
- DBMS vendors & products
  - Oracle, Microsoft (Access and SQL server), IBM (DB2), Sybase, …

Data Models

- A data model is a collection of concepts for describing data.
  - Entity-relation (ER) model
  - Relational model (main focus of this course)
- A schema is a description of data.
- The relational model is the most widely used data model.
  - A relation is basically a table with rows and columns of records.
  - Every relation has a schema, which describes the columns, or fields.

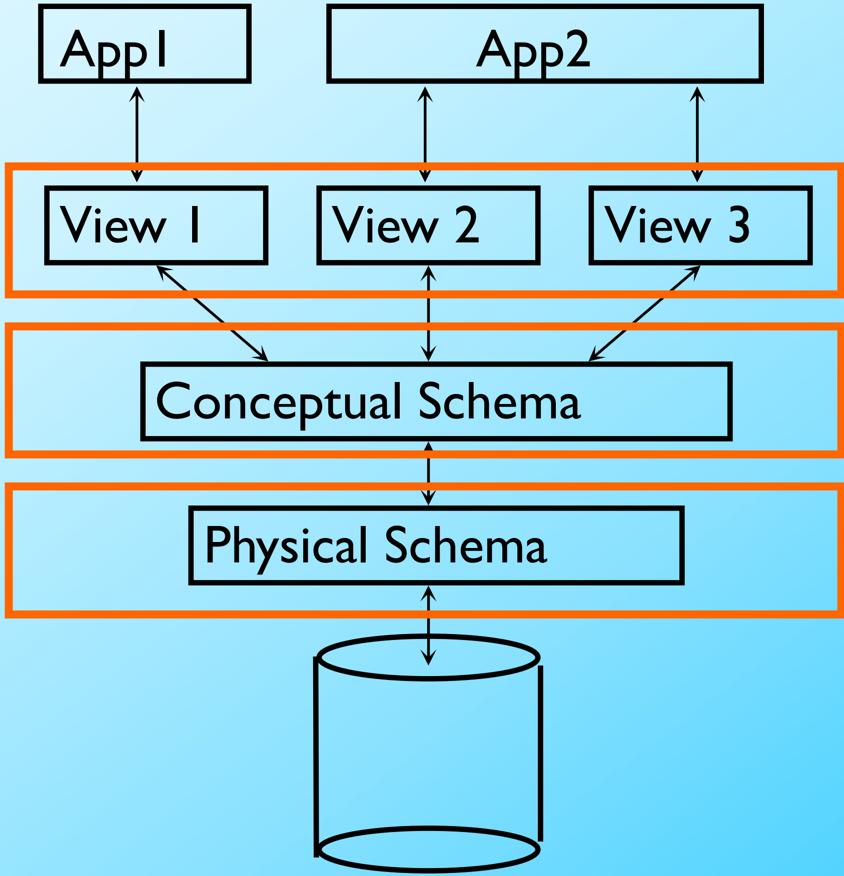
Relational Model

- The entire table shows an instance of the Students relation.
- The Students schema is the column heads
  - Students(Sid: String, Name: String, Login: String, age: Integer,…)

sid	name	email	age	gpa
53666	Jones	Jones@cs	18	3.4
53688	Smith	Smith@ee	18	3.2
53650	Joe	Joe@cs	19	2.5

Levels of Abstractions in DBMS

- Many views, one conceptual schema and one physical schema.
  - Conceptual schema defines logical structure
    - Relation tables
  - Physical schema describes the file and indexing used
    - Sorted file with B+ tree index
  - Views describe how applications (users) see the data
    - Relation tables but not store explicitly

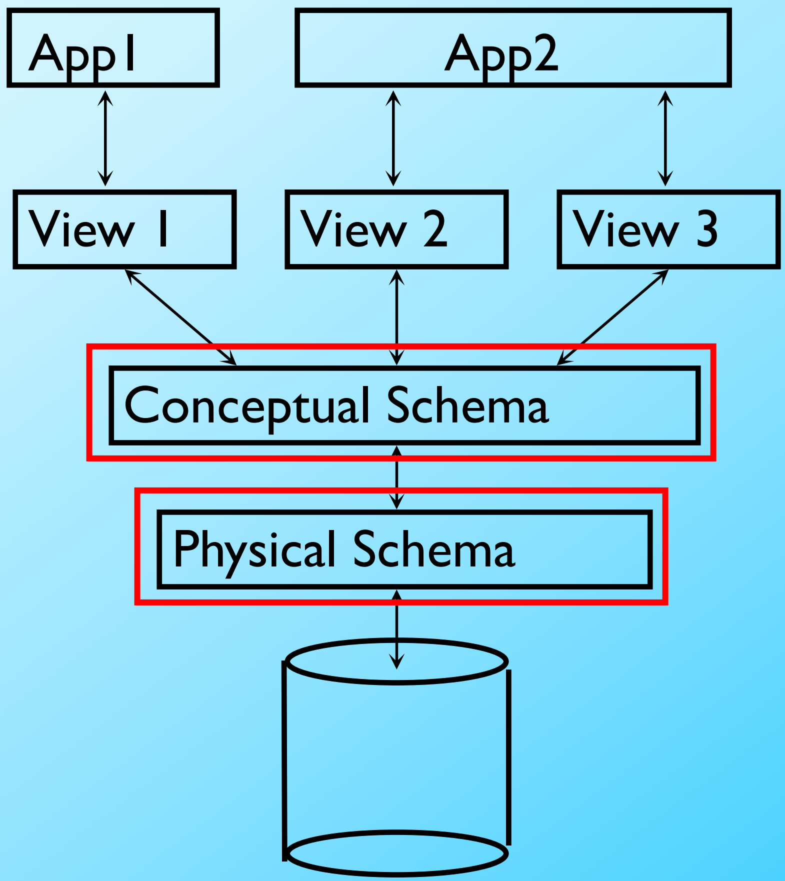


Example: University Database

- Conceptual schema:
  - Students (sid: string, name: string, login: string, age: integer, gpa:real)
  - Courses (cid: string, cname:string, credits:integer)
  - Enrolled (sid:string, cid:string, grade:string)
- Physical schema:
  - Relations stored as unordered files.
  - Index on first column of Students.
- View (External Schema):
  - Course_info(cid:string, enrollment:integer)
  - Why?

Data Independence

- Three levels of abstraction provides data independence.
  - Changes in one layer only affect one upper layer.
  - E.g., applications are not affected by changes in conceptual & physical schema.



Queries in DBMS

- Sample queries on university database:
  - What is the name of the student with student ID 123456?
- The key benefits of using a relational database are
  - Easy to specify queries using a query language: Structured Query Language (SQL)

```
SELECT S.name
FROM Students S
WHERE S.sid = 123456
```

  - Efficient query processor to get answer

Transaction Management

- A transaction is an execution of a user program in a DBMS.
- Transaction management deals with two things:
  – Concurrent execution of transactions
  – Incomplete transactions and system crashes

Concurrency Control

- Example: two travel agents (A, B) are trying to book one remaining airline seat (two transactions), only one transaction can succeed in booking.

  // num_seats is 1

  Transactions A and B: if num_seats > 0, book the seat & num_seat--;

  // overbook!

- How to solve this?

Concurrency Control (Solution)

// num_seats is 1

Transactions A and B: if num_seats > 0, book the seat & num_seat--;
// overbook!

- Solution: use locking protocol

Transaction A: get exclusive lock on num_seats

Transaction B: wait until A releases lock on num_seats

Transaction A: if num_seats > 0, book & num_seat--;
// book the seat, num_seat is set to 0

Transaction A: release exclusive lock on num_seats

Transaction B: num_seats = 0, no booking; // does not book the seat

Crash Recovery

- Example: a bank transaction transfers $100 from account A to account B

  A = A - $100

    <system crashes>   // good for the bank!

  B = B + $100

- How to solve this?

Crash Recovery (Solution)

A = A - $100

<system crashes> // good for the bank!

B = B + $100

- Solution: use logging, meaning that all write operations are recorded in a log on a stable storage.

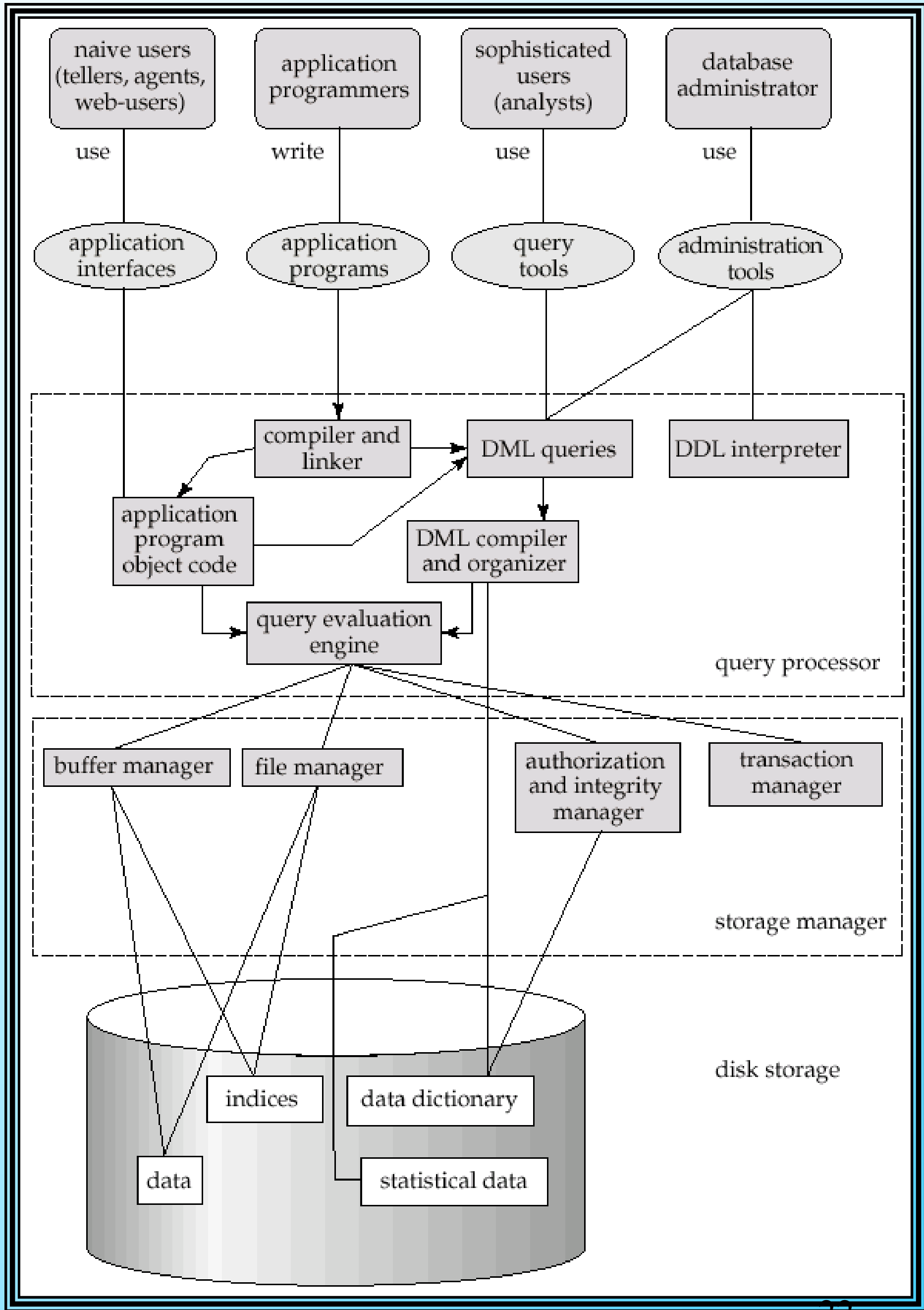
A = A - $100 // recorded A value (checkpoint) in a log

<system crashes>

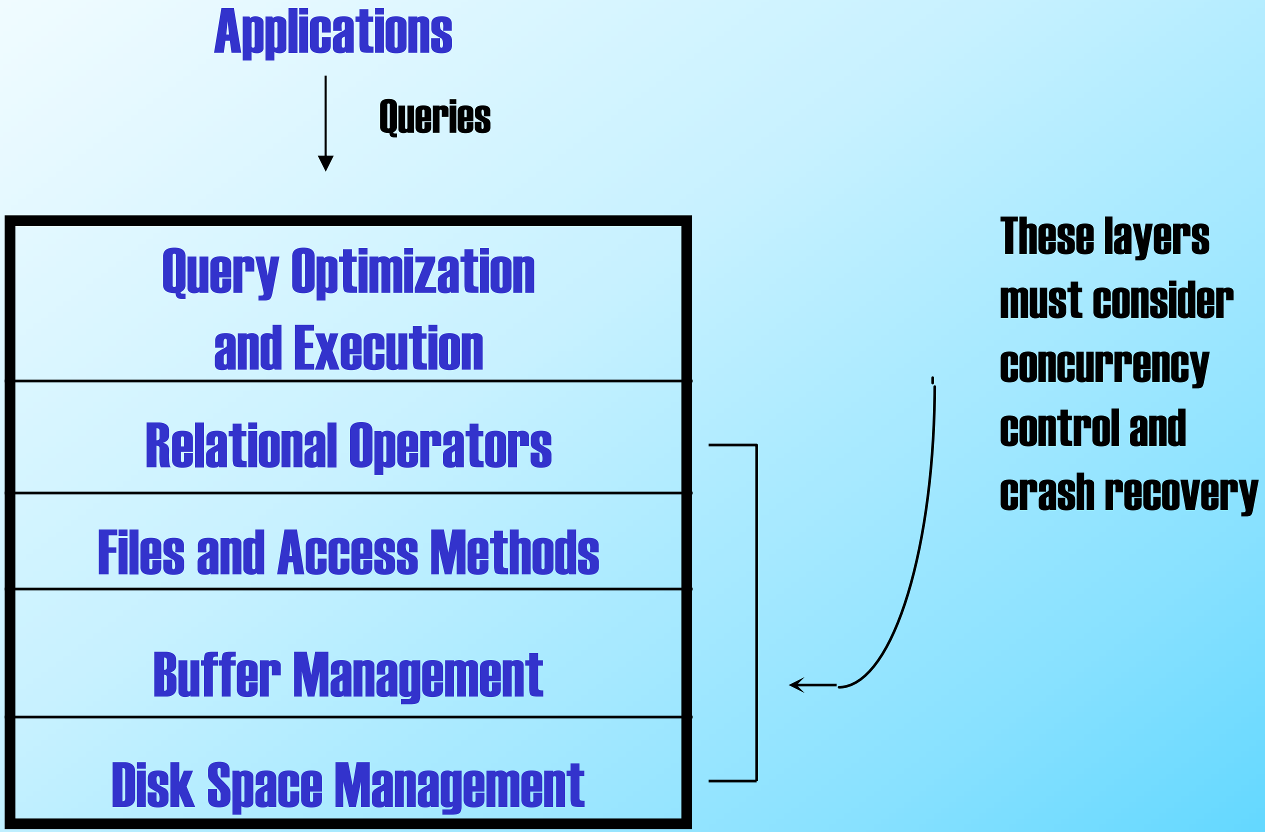
// start recovery: read the log from disk

//analyze, undo, & redo

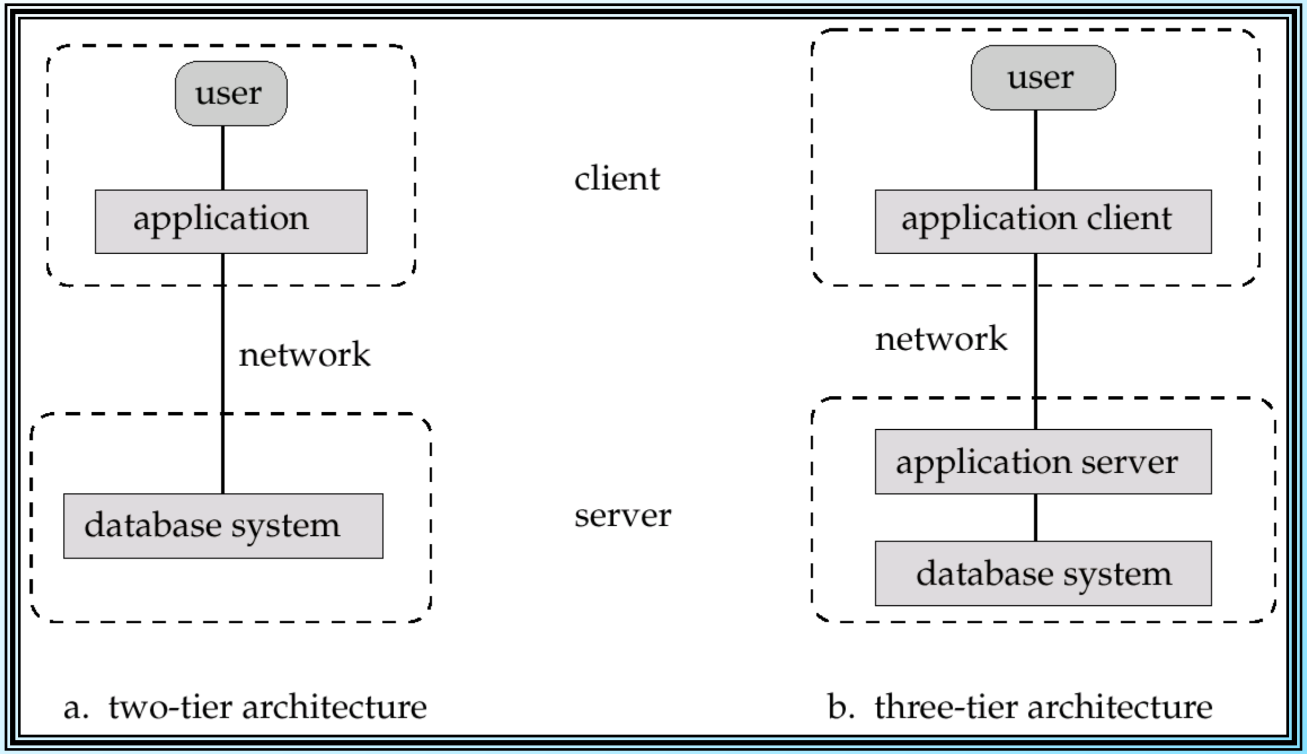
Overview of DBMS



Layered Architecture



Application Architectures



- **Two-tier architecture**: E.g. client programs using ODBC/JDBC to communicate with a database
- **Three-tier architecture**: E.g. web-based applications, and applications built using "middleware"

Homework

- Read Chapters 1
- Read Chapter 2 for next lecture