

Database System Concepts

Chapter 1: Introduction
8/18/2022



山东大学
SHANDONG UNIVERSITY

Resources

- Slides are modified based on the resources on the textbook's official website
 - <http://www.db-book.com/>
- Online SQL interpreter
 - <https://sql.js.org/examples/GUI/index.html>
 - <https://www.db-book.com/university-lab-dir/sqljs.html>
- 网易云课堂
 - <https://mooc.study.163.com/course/1000031000#/info>

Chapter	Formats	Last Updated
Part 1: Overview		
1. Introduction	pptx , pdf	Jan 7, 2019
Part 1: Relational Languages		
2. Introduction to the Relational Model	pptx , pdf	Sep 8, 2019
3. Introduction to SQL	pptx , pdf	Jan 20, 2019
4. Intermediate SQL	pptx , pdf	Feb 2, 2019
5. Advanced SQL	pptx , pdf	Feb 2, 2019
Part 2: Database Design		
6. Database Design Using The E-R Model	pptx , pdf	Feb 13, 2020
7. Relational Database Design	pptx , pdf	Jan 24, 2019
Part 3: Application Design and Development		
8. Complex Data Types	pptx , pdf	Jun 24, 2019
9. Application Development	pptx , pdf	Jan 24, 2019

Online SQL interpreter

Enter some SQL1 DROP TABLE IF EXISTS employees;
2 CREATE TABLE employees (id integer, name text,
3 designation text, manager integer,
4 hired_on date, salary integer,
5 commission float, dept integer);
6
7 INSERT INTO employees VALUES (1, 'JOHNSON', 'ADMIN', 6, '1990-12-17', 18000, NULL, 4);
8 INSERT INTO employees VALUES (2, 'HARDING', 'MANAGER', 9, '1998-02-07', 52000, 300, 3);
9 INSERT INTO employees VALUES (3, 'TAPT', 'SALES I', 2, '1996-01-02', 25000, 500, 3);
10 INSERT INTO employees VALUES (4, 'HOOVER', 'SALES I', 2, '1990-04-02', 27000, NULL, 3);
11 INSERT INTO employees VALUES (5, 'LINCOLN', 'TECH', 6, '1994-06-23', 22500, 1400, 4);
12 INSERT INTO employees VALUES (6, 'GARFIELD', 'MANAGER', 9, '1993-05-01', 54000, NULL, 4);
13 INSERT INTO employees VALUES (7, 'FOLK', 'TECH', 6, '1997-09-22', 25000, NULL, 4);
14 INSERT INTO employees VALUES (8, 'GRANT', 'ENGINEER', 10, '1997-03-30', 32000, NULL, 2);
15 INSERT INTO employees VALUES (9, 'JACKSON', 'CEO', NULL, '1990-01-01', 75000, NULL, 4);
16 INSERT INTO employees VALUES (10, 'YILLMORE', 'MANAGER', 9, '1994-08-09', 56000, NULL, 2);
17 INSERT INTO employees VALUES (11, 'ADAMS', 'ENGINEER', 10, '1996-03-15', 34000, NULL, 2);
18 INSERT INTO employees VALUES (12, 'WASHINGTON', 'ADMIN', 6, '1998-04-16', 18000, NULL, 4);
19 INSERT INTO employees VALUES (13, 'MONROE', 'ENGINEER', 10, '2000-12-03', 30000, NULL, 2);
20 INSERT INTO employees VALUES (14, 'ROOSEVELT', 'CPA', 9, '1995-10-12', 35000, NULL, 1);
21
22 SELECT designation, COUNT(*) AS nbr, (AVG(salary)) AS avg_salary FROM employees GROUP BY designation ORDER BY a
23 SELECT name, hired_on FROM employees ORDER BY hired_on;

Execute Save the db Load an SQLite database file: 选择文件 未选择任何文件

Results will be displayed here

课件

- > 第一章--引论
- > 第二章--关系数据模型
- > 第三章SQL语言(1)-表定义
- > 第四章SQL语言(2)-数据查询和操作
- > 第五章SQL语言(3)-视图与索引
- > 第六章SQL语言(4)-数据完整性、安全性和事务
- > 实验环境搭建
- > 第七章SQL语言(5)-嵌入式SQL和ODBC
- > 第九章函数依赖和关系模式分解

Outline

- Database-System Applications
- Purpose of Databases Systems
- View of Data
- Database Languages
- Database Design
- Database Engine
- Database Architecture
- Database Users and Administrators
- History of Database Systems

Outline

- Database-System Applications
- Purpose of Databases Systems
- View of Data
- Database Languages
- Database Design
- Database Engine
- Database Architecture
- Database Users and Administrators
- History of Database Systems

Outline

- Database-System Applications
- Purpose of Databases Systems
- View of Data
- Database Languages
- Database Design
- Database Engine
- Database Architecture
- Database Users and Administrators
- History of Database Systems

Database Applications Examples

- Enterprise Information
 - Sales: customers, products, purchases
 - Accounting: payments, receipts, assets
 - Human Resources: Information about employees, salaries, payroll taxes.
- Manufacturing: management of production, inventory, orders, supply chain.
- Banking and finance
 - customer information, accounts, loans, and banking transactions.
 - Credit card transactions
 - Finance: sales and purchases of financial instruments (e.g., stocks and bonds; storing real-time market data
- Universities: registration, grades

Database Applications Examples (cont.)

- Airlines: reservations, schedules
- Telecommunication: records of calls, texts, and data usage, generating monthly bills, maintaining balances on prepaid calling cards
- Web-based services
 - Online retailers: order tracking, customized recommendations
 - Online advertisements
- Document databases
- Navigation systems: For maintaining the locations of various places of interest along with the exact routes of roads, train systems, buses, etc.

Outline

- Database-System Applications
- **Purpose of Databases Systems**
- View of Data
- Database Languages
- Database Design
- Database Engine
- Database Architecture
- Database Users and Administrators
- History of Database Systems

DB Table Example

- 学生分数表设计

学号	姓名	专业	校区	DB平时	DB期末	DB总评成绩	OS平时	OS期末	OS总评成绩
3023001093	刘宇昕	计基地16	青岛	90	80	85			80
3011112340	刘安	计基地16	青岛	80	90	85			90
3020621034	崔晨	计基地16	青岛	85	85	85			85
3021131123	张玉杰	计基地16	青岛	70	90	80			88
3022112102	郭文栋	计基地16	青岛	95	85	90			90
3022112002	高德琛	计算机14	济南	80	90	85			85
3022112003	计丕迪	计算机14	济南	90	90	90			88
3021131113	焦江涛	计算机14	济南	95	85	90			90

这样的设计好么？

DB Table Example

这样的设计好么？为什么？

- 另一种设计

Course

CID	CName	Credit
1	DB	4
2	OS	5
3	English	4
4	Math	4

Location

Specialty	Location
计基地16	青岛
计算机14	济南

Students

SID	Sname	SSex	SAge	Specialty
3023001093	崔晨	M	21	计基地16
3011112340	张玉杰	F	20	计基地16
3020621034	郭文栋	M	18	计基地16
3020831035	高德琛	M	19	计算机14
3021131123	计丕迪	F	22	计算机14

Enrolled

SID	CID	Grade1	Grade2	Grade3
3023001093	1			
3011112340	2			
3020621034	1			
3020831035	1			
3021131123	2			

DB Table Example

- 学生分数表设计

结构不稳定

学号	姓名	专业	校区	DB平时	DB期末	DB总评成绩	OS平时	OS期末	OS总评成绩
3023001093	刘宇昕	计基地16	青岛	90	80	85			80
3011112340	刘安	计基地16	青岛	80	90	85			90
3020621034	崔晨	计基地16	青岛	85	85	85			85
3021131123	张玉杰	计基地16	青岛	70	90	80			88
3022112102	郭文栋	计基地16	青岛	95	85	90			90
3022112002	高德琛	计算机14	济南	80	90	85			85
3022112003	计丕迪	计算机14	济南	90	90	90			88
3021131113	焦江涛	计算机14	济南	95	85	90			90

信息冗余

University Database Example

- Suppose a university organization, keep information about
 - *instructor*, *students*, *departments*, and *course* offering
- To manipulate the information, a number of application program examples
 - add new students, instructors, and courses
 - register students for courses, and generate class rosters (班级名单)
 - assign grades to students, compute grade point averages (GPA) and generate transcripts (成绩单)
- In the early days, database applications were built directly on top of file systems

Drawbacks of using file systems to store data

- Data redundancy(冗余) and inconsistency(不一致)
 - Multiple file formats, duplication of information in different files

Example

Student, double major: Math, Music

- records in the Math Dept.
- records in the Music Dept.

Duplication info in two departments

Data *inconsistency* if changes make in one place

Drawbacks of using file systems to store data

- Difficulty in accessing data
 - Need to write a new program to carry out each new task

Example

Query 1:

to generate the list of **all** the students

Query 2:

to generate the list of students in Math Dept.

Do not allow needed data to be retrieved in a ***convenient*** and ***efficient*** manner

Drawbacks of using file systems to store data

- Data isolation (隔离性)
 - Multiple files and formats

Example

Data are scattered in *various files*, and files may be in *different formats*

Difficult to write new applications to retrieve the appropriate data

Drawbacks of using file systems to store data

- Integrity(完整性) problems
 - Integrity constraints (e.g., account balance > 0) become “buried” in program code rather than being stated explicitly
 - Hard to add new constraints or change existing ones

Example

Suppose the university maintains an account for each dept., and records the balance amount in each account > 0

Write code in various programs to enforce these constraints
New constraints added, difficult to change the program to enforce them

Drawbacks of using file systems to store data

- Atomicity(原子性) of updates
 - Failures may leave database in an inconsistent state with partial updates carried out

Example

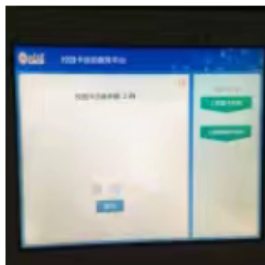
Consider a program to transfer \$500 from the account balance of Dept. A to Dept. B. If a system failure occurs during the program execution.

Transfer of funds from one account to another should either complete or not happen at all



gantian 🍒 🍇

求教有没有人遇到充值校园卡的问题呀 🙏 🙏
🙏 中午吃饭的时候充卡，充了100（现在想想可能最后一步没有结束我就拿走卡了？），我去吃饭的时候发现余额还是个位数，然后我就立马回充值机去看，显示的还是10X.XX。以为是同步问题之类的，蹭了个饭卡就忘记了。刚刚晚上吃饭，发现充值机上也显示的是X.XX了。回lab登录校园卡系统，发现密码忘记了已经--||。打了服务电话没人接，求教有没有人有类似的经历



3 January 2017 5:39 PM 🧑🧑 删除





3 January 5:43 PM

我觉得可以再冲10块试试，强行再次同步这张卡



gantian 🍒 🍇

3 January 5:44 PM

回复 试过！充了50！然后就只是加了50😂😂😂



gantian 🍒 🍇

3 January 5:44 PM

回复 我都怀疑我中午充钱是不是我做梦来着😂😂😂



3 January 5:45 PM

回复 gantian 🍒 🍇 : 有没有充值记录查询的功能。。。



gantian 🍒 🍇

3 January 5:47 PM

回复 有！我密码忘记了进不去了😂😂😂 估计要跑几十公里的另一个校区去处理😂😂😂 哈哈哈哈哈



3 January 5:47 PM

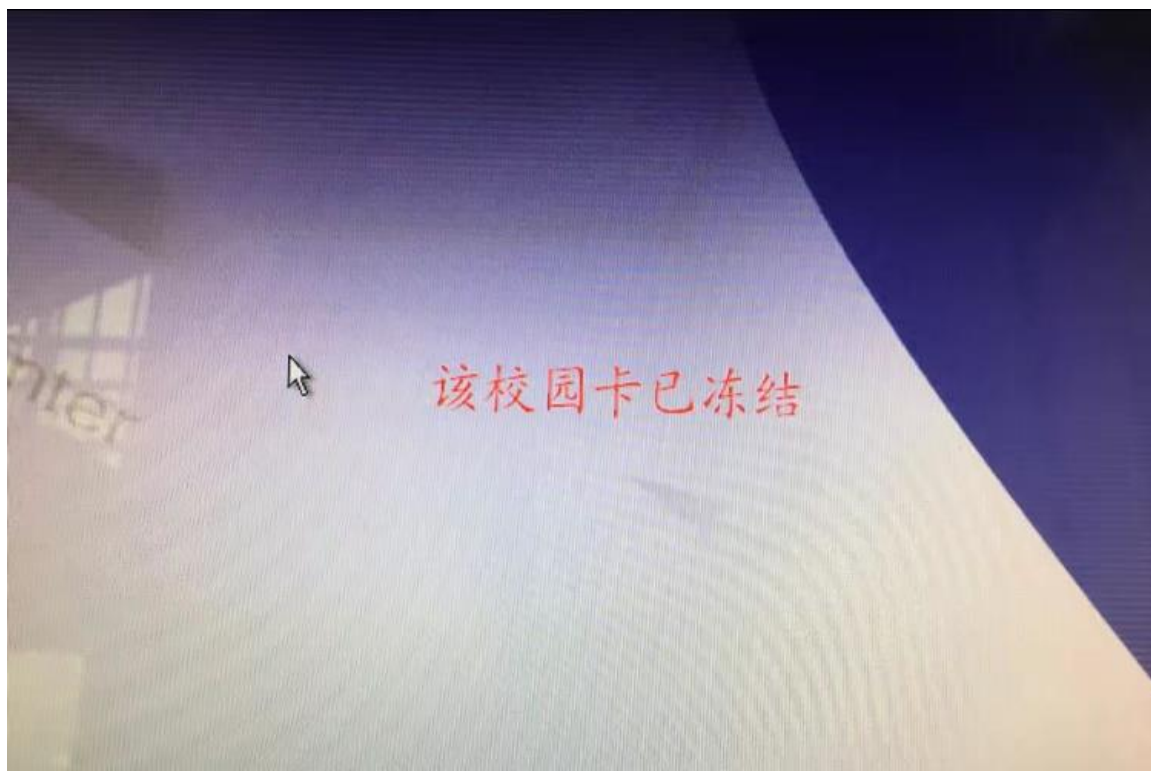
回复 gantian 🍒 🍇 : 醉了醉了



gantian 🍒 🍇

3 January 5:48 PM

当年数据库学的不是操作没完成，要回滚的么😡 钱呢😡😡😡 以后别碰到让我来教数据库的课😡😡😡🙏🙏🙏



终于打通电话了，告诉我，我是充了100，但是因为充值机网络传输太慢了，系统没有更新上！然后！我的卡还被莫名其妙冻结了！今天只能去千佛山校区和中心校区下午三点前解锁！软件园要等到周五🎁🎁🎁



赞



评论



2



41

Drawbacks of using file systems to store data (Cont.)

- Concurrent(并发) access by multiple users
 - Concurrent access needed for performance
 - Uncontrolled concurrent accesses can lead to inconsistencies

Example

Two people reading a balance (say 100) and updating it by withdrawing() money (say 50 each) at the same time

Drawbacks of using file systems to store data (Cont.)

- Security problems
 - Hard to provide user access to some, but not all, data

Example

Students can only view their own information

In file-processing system, application programs are added to the system in an ad hoc (特设) manner, enforcing such security constraints is difficult!

Drawbacks of using file systems to store data

- Data redundancy and inconsistency
- Difficulty in accessing data
- Data isolation
- Integrity problems
- Atomicity of updates
- Concurrent access by multiple users
- Security problems

Database systems offer solutions to all the above problems

Outline

- Database-System Applications
- Purpose of Databases Systems
- **View of Data**
- Database Languages
- Database Design
- Database Engine
- Database Architecture
- Database Users and Administrators
- History of Database Systems

Data Models

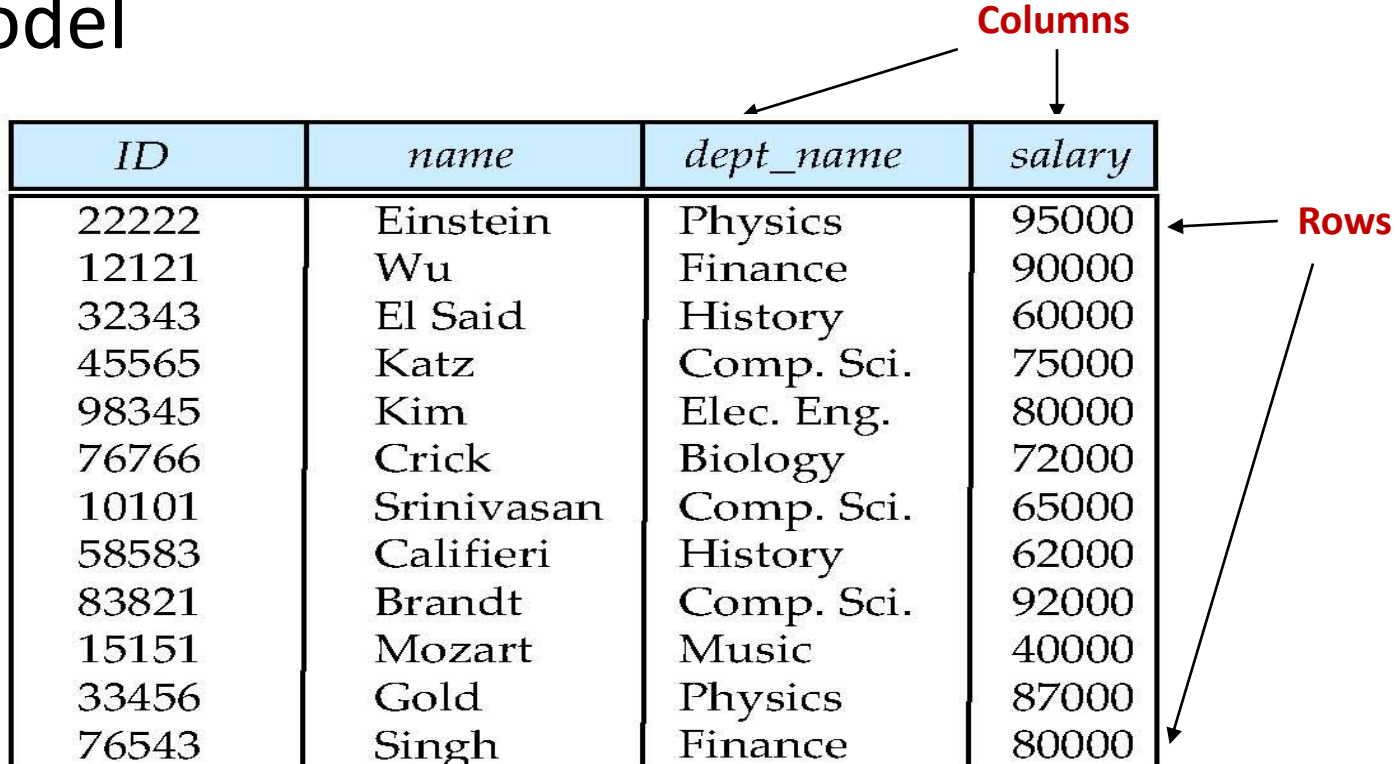
- A collection of tools for describing
 - Data
 - Data relationships
 - Data semantics (语义)
 - Data constraints

Data Models

- Relational model
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semistructured data model (XML)
- Other older models:
 - Network model
 - Hierarchical model

Relational Model

- All the data is stored in various tables.
- Example of tabular data in the relational model



The diagram shows a table with four columns and 13 rows. An arrow labeled 'Columns' points to the top row, and an arrow labeled 'Rows' points to the right side of the table.

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
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

A Sample Relational Database

<i>ID</i>	<i>name</i>	<i>dept_name</i>	<i>salary</i>
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

instructor

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

department

Levels of Abstraction

- **Physical level:** describes how a record (e.g., instructor) is stored.
- **Logical level:** describes data stored in database, and the relationships among the data.

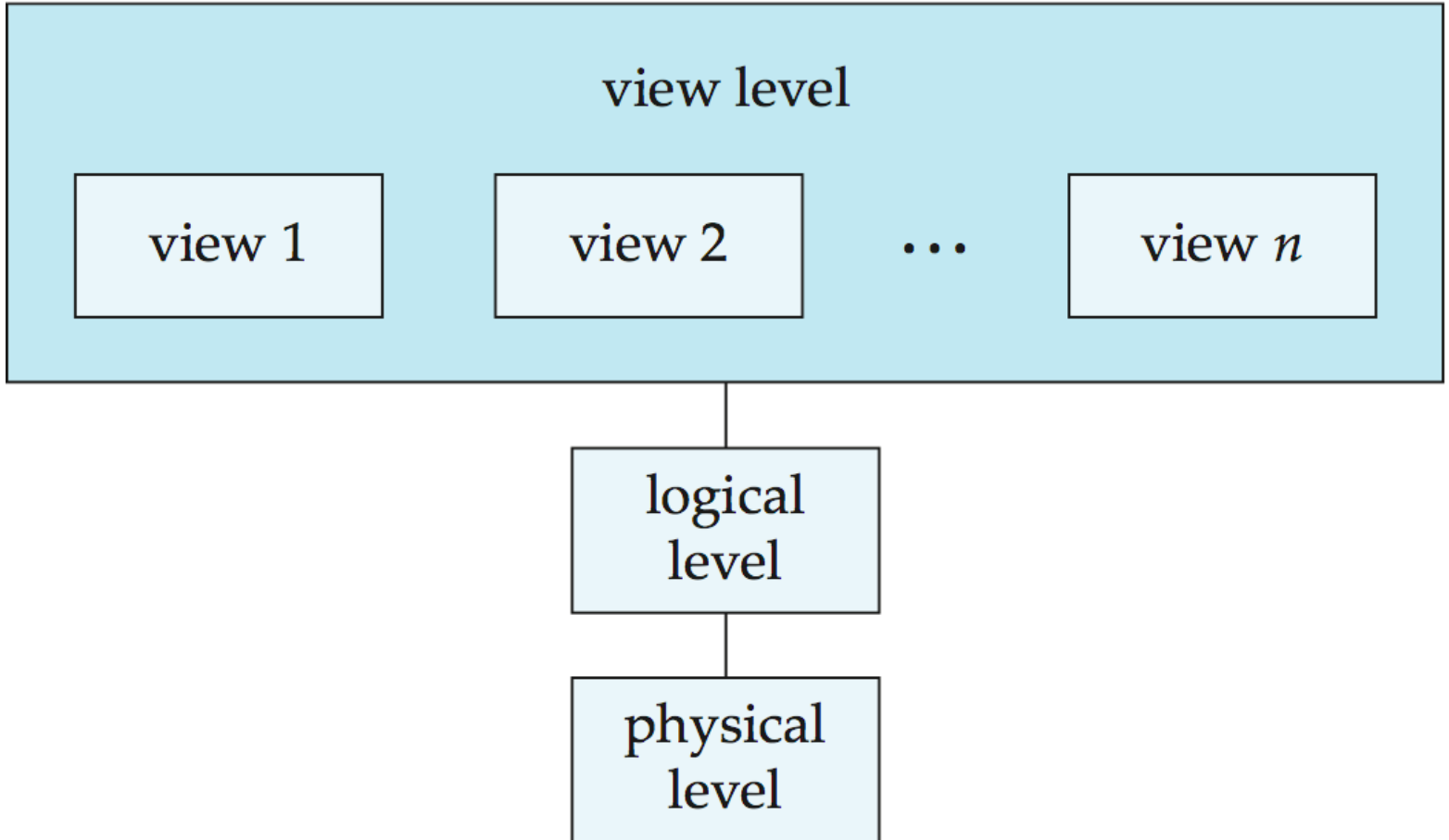
```
type instructor = record
```

```
    ID : string;  
    name : string;  
    dept_name : string;  
    salary : integer;  
end;
```

- **View level:** application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.

View of Data

An architecture for a database system



Instances and Schemas

- **Schema (模式)**
 - the overall logical/physical structure of the database
 - analogous to variable declarations (with type information) in a program
- **Instance (实例)**
 - the actual content of the database at a particular point in time
 - analogous to the values of a variable at a point in time
- **Physical Data Independence**
 - applications do **NOT** depend on the physical schema
 - the ability to modify the physical schema without changing 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

- Database-System Applications
- Purpose of Databases Systems
- View of Data
- **Database Languages**
- Database Design
- Database Engine
- Database Architecture
- Database Users and Administrators
- History of Database Systems

Data Definition Language (DDL)

- Specification notation for defining the database schema

Example: **create table** *instructor* (
 ID **char**(5),
 name **varchar**(20),
 dept_name **varchar**(20),
 salary **numeric**(8,2))

- DDL compiler generates a set of table templates stored in a ***data dictionary***
- Data dictionary contains metadata (i.e., data about data)
 - Database schema
 - Integrity constraints
 - Primary key (ID uniquely identifies instructors)
 - Authorization
 - Who can access what

Data Manipulation Language (DML)

- Language for accessing and manipulating the data organized by the appropriate data model
 - DML also known as query language
- Two classes of languages
 - **Procedural DML** – require a user to specify what data are needed and how to get those data.
 - used for proving properties about computational power and for optimization
 - **Declarative DML** – require a user to specify what data are needed without specifying how to get those data
- The portion of a DML that involves information retrieval is called a **query** language

SQL

- SQL query language is nonprocedural. A query takes as input several tables (possibly only one) and always returns a single table.
- Example to find all instructors in Comp. Sci. dept

```
select name  
from instructor  
where dept_name = 'Comp. Sci.'
```
- 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

Database Access from Application Program

- Non-procedural query languages such as SQL are not as powerful as a universal Turing machine.
- SQL does not support actions such as input from users, output to displays, or communication over the network.
- Such computations and actions must be written in a **host language**, such as C/C++, Java or Python, with embedded SQL queries that access the data in the database.
- **Application programs**
 - are programs that are used to interact with the database in this fashion.

Outline

- Database-System Applications
- Purpose of Databases Systems
- View of Data
- Database Languages
- **Database Design**
- Database Engine
- Database Architecture
- Database Users and Administrators
- 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 (Cont.)

- Is there any problem with this relation?

<i>ID</i>	<i>name</i>	<i>salary</i>	<i>dept_name</i>	<i>building</i>	<i>budget</i>
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 7)
 - Models an enterprise as a collection of *entities* and *relationships*
 - Represented diagrammatically by an *entity-relationship diagram*:
 - Normalization Theory (Chapter 8)
 - Formalize what designs are bad, and test for them

Outline

- Database-System Applications
- Purpose of Databases Systems
- View of Data
- Database Languages
- Database Design
- **Database Engine**
- Database Architecture
- Database Users and Administrators
- History of Database Systems

Database Engine

- A database system is partitioned into modules that deal with each of the responsibilities of the overall system.
- The functional components of a database system can be divided into
 - The storage manager
 - The query processor component
 - The transaction management component

Storage Manager

- 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
- The storage manager components include:
 - Authorization and integrity manager
 - Transaction manager
 - File manager
 - Buffer manager

Storage Manager (Cont.)

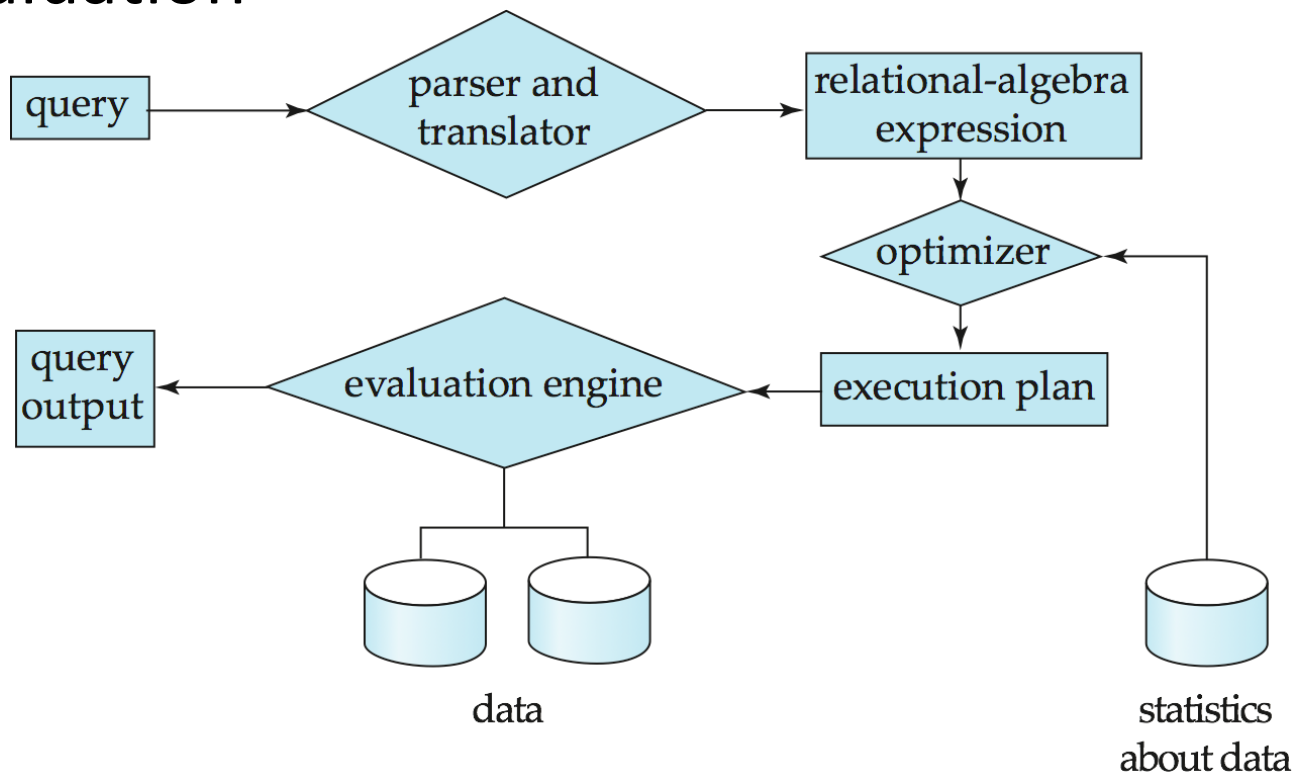
- The storage manager implements several data structures as part of the physical system implementation:
 - Data files
 - store the database itself
 - Data dictionary
 - stores metadata about the structure of the database, in particular the schema of the database.
 - Indices
 - can provide fast access to data items. A database index provides pointers to those data items that hold a particular value.

Query Processor

- The query processor components include:
 - DDL interpreter
 - interprets DDL statements and records the definitions in the data dictionary.
 - DML compiler
 - translates DML statements in a query language into an evaluation plan consisting of low-level instructions that the query evaluation engine understands.
 - The DML compiler performs query optimization; that is, it picks the lowest cost evaluation plan from among the various alternatives.
 - Query evaluation engine
 - executes low-level instructions generated by the DML compiler.

Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



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.
- **Concurrency-control manager** controls the interaction among the concurrent transactions, to ensure the consistency of the database.

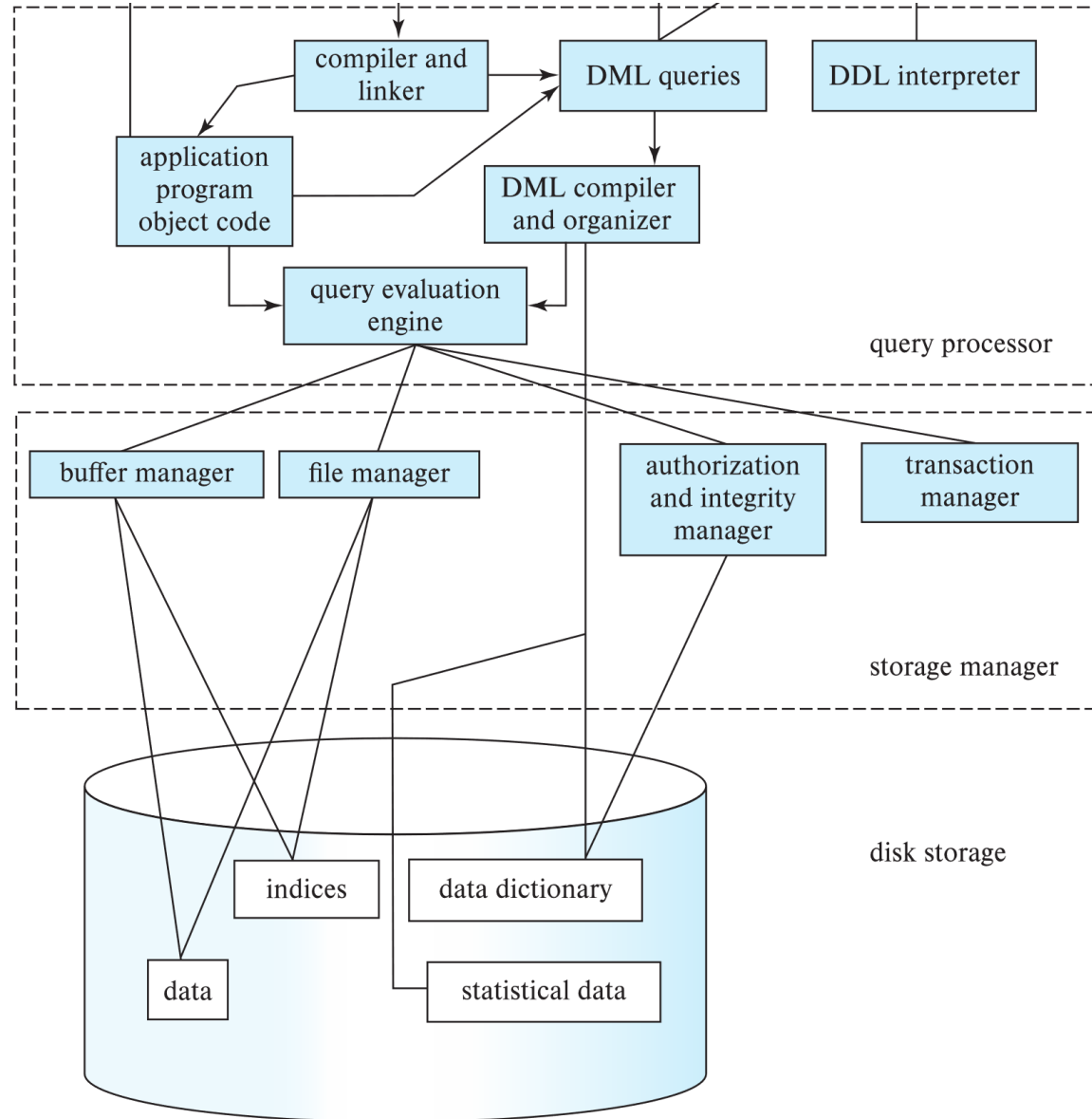
Outline

- Database-System Applications
- Purpose of Databases Systems
- View of Data
- Database Languages
- Database Design
- Database Engine
- **Database Architecture**
- Database Users and Administrators
- History of Database Systems

Database Architecture

- Centralized databases
 - One to a few cores, shared memory
- Client-server,
 - One server machine executes work on behalf of multiple client machines.
- Parallel databases
 - Many core shared memory
 - Shared disk
 - Shared nothing
- Distributed databases
 - Geographical distribution
 - Schema/data heterogeneity

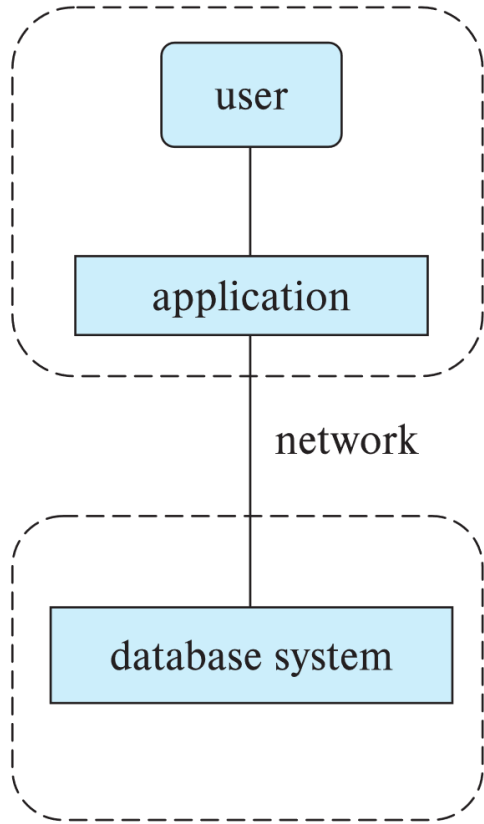
Database Architecture (Centralized/Shared-Memory)



Database Applications

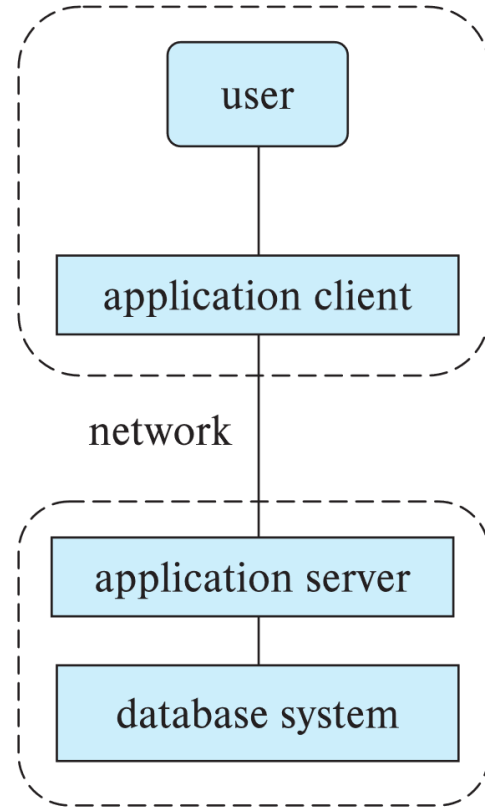
- Database applications are usually partitioned into two or three parts
 - Two-tier architecture
 - the application resides at the client machine, where it invokes database system functionality at the server machine
 - Three-tier architecture
 - the client machine acts as a front end and does not contain any direct database calls.
 - The client end communicates with an application server, usually through a forms interface.
 - The application server in turn communicates with a database system to access data.

Two-tier and three-tier architectures



(a) Two-tier architecture

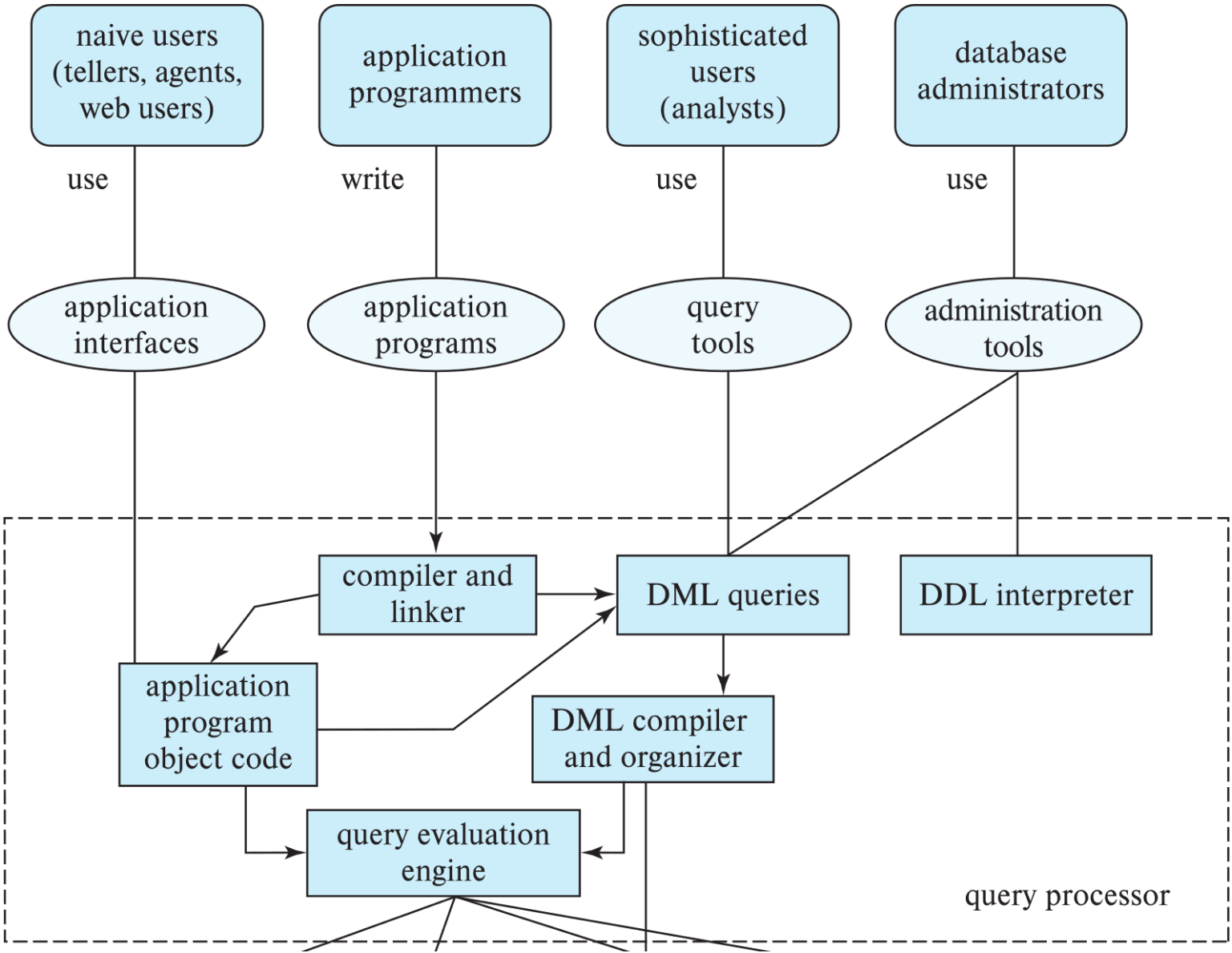
client



server

(b) Three-tier architecture

Database Users



Database Administrator

- A person who has central control over the system is called a **database administrator (DBA)**. Functions of a DBA include:
 - Schema definition
 - Storage structure and access-method definition
 - Schema and physical-organization modification
 - Granting of authorization for data access
 - Routine maintenance
 - Periodically backing up the database
 - Ensuring that enough free disk space is available for normal operations, and upgrading disk space as required
 - Monitoring jobs running on the database

Outline

- Database-System Applications
- Purpose of Databases Systems
- View of Data
- Database Languages
- Database Design
- Database Engine
- Database Architecture
- Database Users and Administrators
- **History of Database Systems**

History of Database Systems

- History of Database [[video](#)]
- 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
 - Would win the ACM Turing Award for this work
 - IBM Research begins System R prototype
 - UC Berkeley begins Ingres prototype
 - High-performance (for the era) transaction processing

History (cont.)

- 1980s:
 - Research relational prototypes evolve into commercial systems
 - SQL becomes industrial standard
 - Parallel and distributed database systems
 - Object-oriented database systems
- 1990s:
 - Large decision support and data-mining applications
 - Large multi-terabyte data warehouses
 - Emergence of Web commerce

History (cont.)

- 2000s
 - Big data storage systems
 - Google BigTable, Yahoo PNuts, Amazon,
 - “NoSQL” systems.
 - Big data analysis: beyond SQL
 - Map reduce and friends
- 2010s
 - SQL reloaded
 - SQL front end to Map Reduce systems
 - Massively parallel database systems
 - Multi-core main-memory databases

Thank you!

Q&A



山东大学
SHANDONG UNIVERSITY