Database System

- **Textbook: Database System Concepts (6th Edition)**
- **Instructor : SUN Jianling(孙建伶)** (Professor)

  Office:Yi-Fu Gong Shang Building 234
  (逸夫工商管理楼 234)
  Yu Quan Campus (玉泉校区)
  Email: sunjl@zju.edu.cn
  Tel: 87952700, 13705818519
  wechat: sobeksun
- **Course TA : 毛启衡（ Master Student)**

  Email:maoqiheng@gmail.com
  Tel: 18868111930
  wechat: HN0821521

Database System

- **Course Grading Policy （课程成绩评定细则）：**

Exercise （作业）	10%
Quiz （测试＋讨论）	10%
Lab & Project （实验和大程）	30%
Exam （考试）	50%

(Open two-page notes, handwriting, with student ID & name)

Database System

- **Course URL:**
  - https://courses.zju.edu.cn/course/26958/content#/ （学在浙大）
  - Wechat: 2021 春夏数据库系统

Database System in CS

AI 、 CAD&CG 、 Multimedia 、 e-commerence 、 Numerical Analysis 、 Software Engine 、 Embedded System
C 、 C++ 、 Java 、 Data Structure 、 Algorithm
Complier 、 **Database System** 、 Network
Operating System
Computer Organization 、 Computer Architecture 、 Assemble
Circuit 、 Digital circuit

Chapter 1: Introduction

Database System Concepts, 6th Ed.

Outline

- Database Systems
- Database Applications
- Purpose of Database Systems
- View of Data
- Data Models
- Database Languages
- Database Design
- Database Engine
- Database Users and Administrators
- History of Database Systems

Database Systems

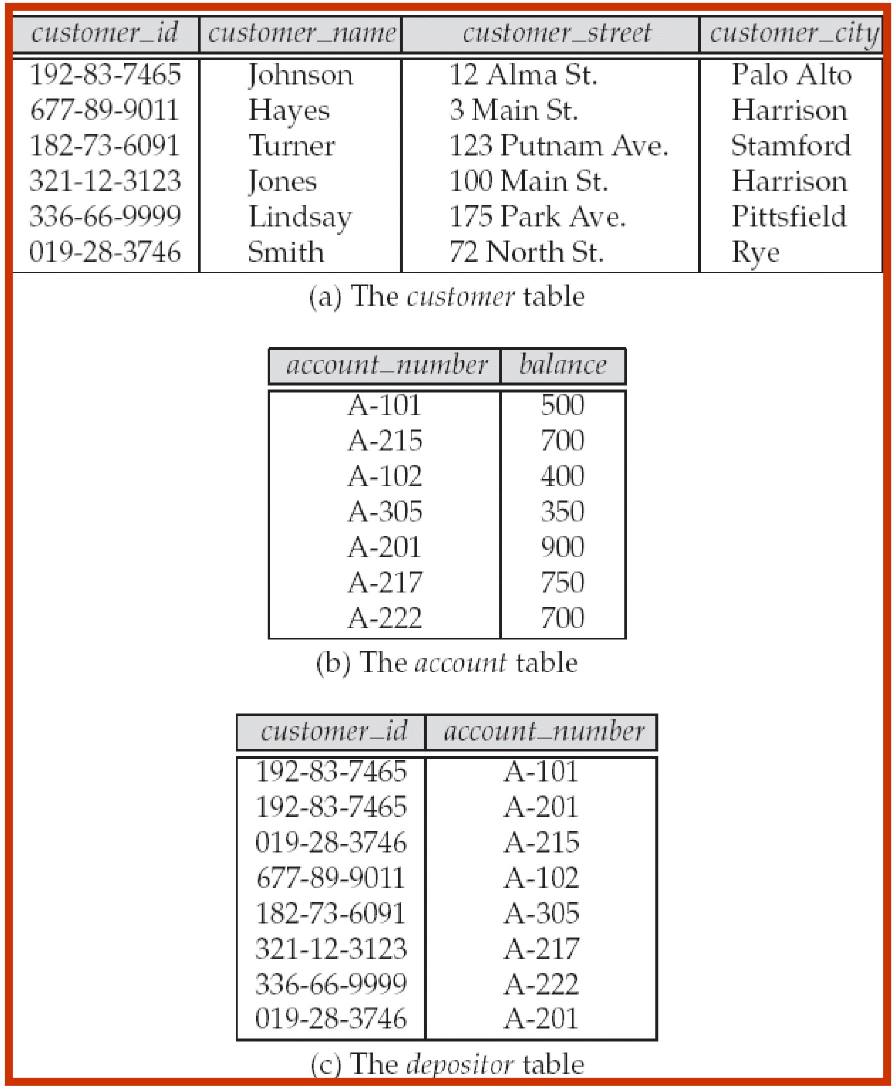
- **Database** is a collection of interrelated data about a enterprise, which is managed by a **DBMS(**Database Management System**).**
- The primary goal of a DBMS is to provide a way to store and retrieve database information that is both convenient and efficient.
- Management of data involves both defining structures for storage of information and providing mechanisms for the manipulation of information.
- The **database system** must ensure the safety of the information stored, despite system crashes or attempts at unauthorized access.
- If data are to be shared among several users, the system must provide concurrency control mechanisms to avoid possible anomalous results.

Database Applications

- **Database Applications（数据库应用，数据库应用系统）：**
  - **Banking:** accounts, loans, transactions
  - **Universities**: course, registration, grades
  - **Airlines**: reservations, schedules
  - **Sales**: customers, products, purchases
  - **Online retailers**: order tracking, customized recommendations
  - **Manufacturing**: production, inventory, orders, supply chain
  - **Human resources**: employee records, salaries, tax deductions
- Databases can be very large. →**big data**
- Databases touch all aspects of our lives

Database Example-Banking

- Application program example - **Banking**
  - Add customers
  - Open accounts
  - Save/Withdraw money
  - Lend/ Repay loans



customer_id	*customer_name*	*customer_street*	*customer_city*
192-83-7465	Johnson	12 Alma St.	Palo Alto
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

account_number	*balance*
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

(b) The *account* table

customer_id	*account_number*
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

(c) The *depositor* table

Database Example- University

- Application program example - **University**
  - Add new students, instructors, and courses
  - Register students for courses, and generate class rosters
  - Assign grades to students
  - compute grade point averages (GPA) and generate transcripts

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

Purpose of Database Systems

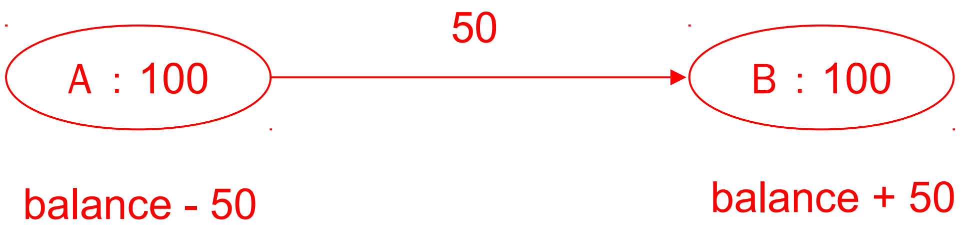
- In the early days, database applications were built directly on top of file systems， which leads to:
  - **Data redundancy**（**数据冗余**） and inconsistency（**不一致**）
    - Multiple file formats, duplication of information in different files
  - **Data isolation**（**数据孤立，数据孤岛**）－ multiple files and formats
  - Difficulty in **accessing data** （**存取数据困难**）
    - Need to write a new program to carry out each new task

Purpose of Database Systems

- **Integrity** problems（**完整性问题**）
  - Integrity constraints become "buried" in program code rather than being stated explicitly(**显式的**)
  - Example: "account balance >=1"
  - Hard to add new constraints or change existing ones

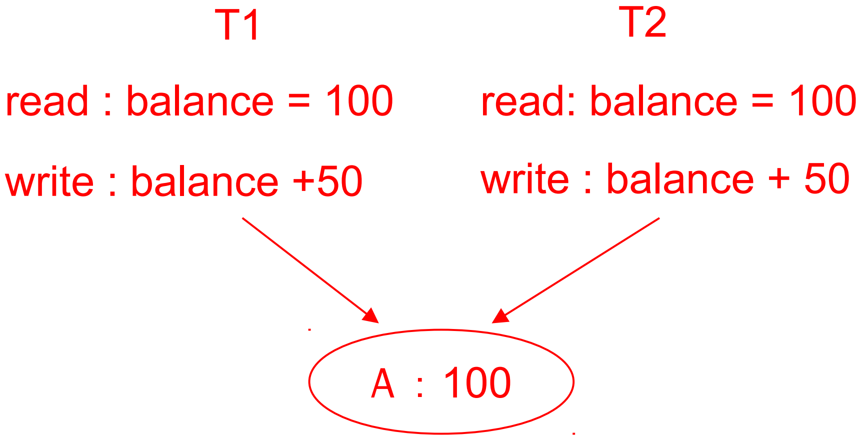
Purpose of Database Systems

- **Atomicity** problems（**原子性问题**）
  - Failures may leave database in an inconsistent state with partial updates carried out
  - Example: Transfer of funds from one account to another should either complete or not happen at all



Purpose of Database Systems

- **Concurrent access** anomalies（**并发访问异常**）
  - Concurrent access needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
  - Example: Two people reading a balance (say 100) and updating it by saving money (say 50 each) at the same time



T1

read : balance = 100

write : balance +50

T2

read: balance = 100

write : balance + 50

A : 100

Purpose of Database Systems

- **Security** problems（**安全性问题**）
  - Hard to provide user access to some, but not all, data
  - Authentication(认证）
  - Priviledge（权限）
  - Audit（审计）

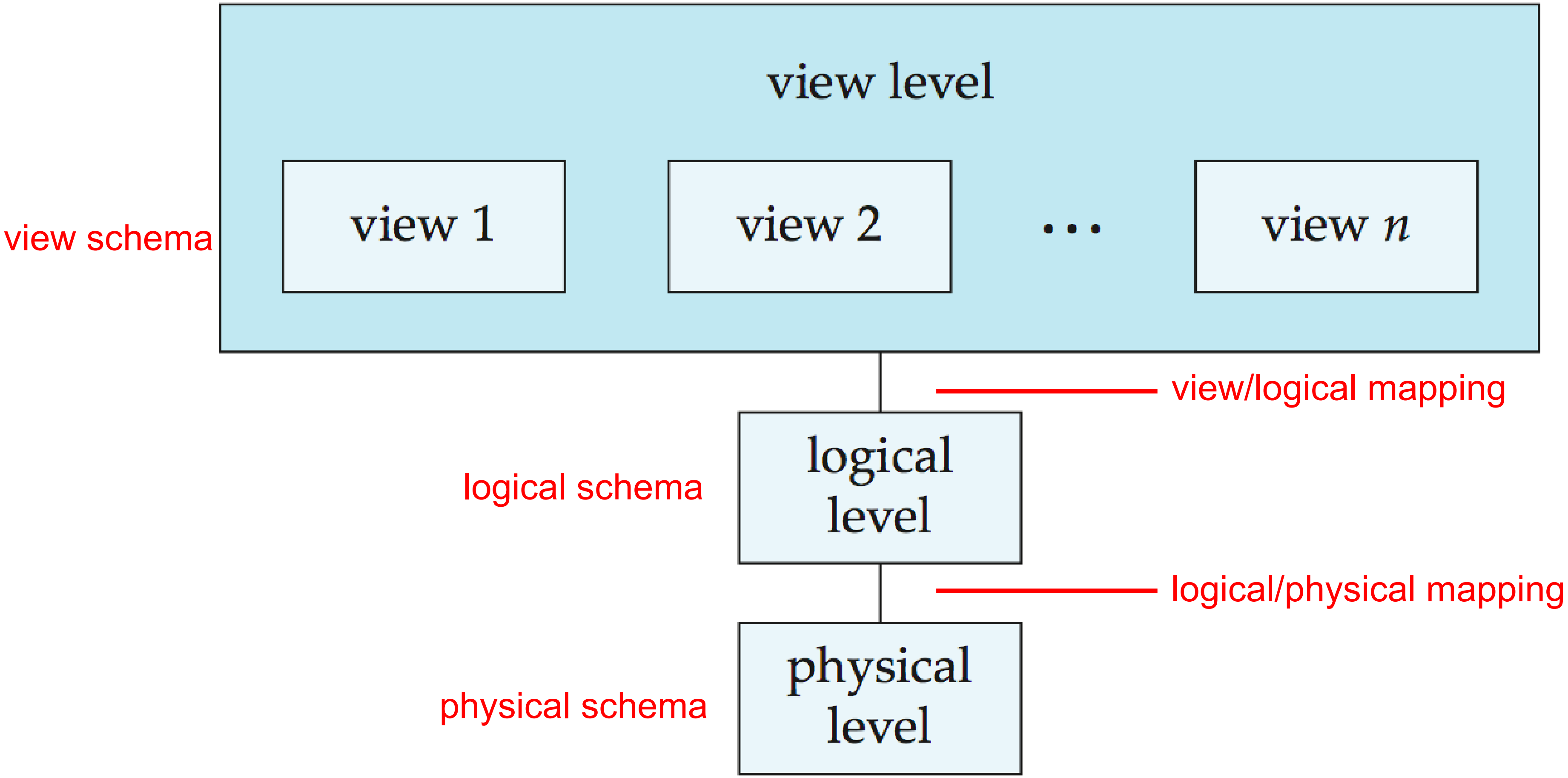
Database systems offer solutions to all the above problems

Characteristics of Databases

- Characteristics of Databases
  - data persistence(**数据持久性**)
  - convenience in accessing data(**数据访问便利性**)
  - data integrity （**数据完整性**）
  - concurrency control for multiple user(**多用户并发控制**)
  - failure recovery（**故障恢复**）
  - security control（**安全控制**）

View of Data

Three-level abstraction of databases



Schema and Instance

- Similar to types and variables in programming languages
- **Schema（模式）**– the logical structure of the database
  - Example: The database consists of information about a set of customers and accounts and the relationship between them
  - Analogous to type information of a variable in a program
  - **Physical schema（物理模式）**: database design at the physical level
  - **Logical schema（逻辑模式）:** database design at the logical level
- **Instance（实例）**– the actual content of the database at a particular point in time
  - Analogous to the value of a variable

Data Independence

- **Physical Data Independence（物理数据独立性）**－ 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.

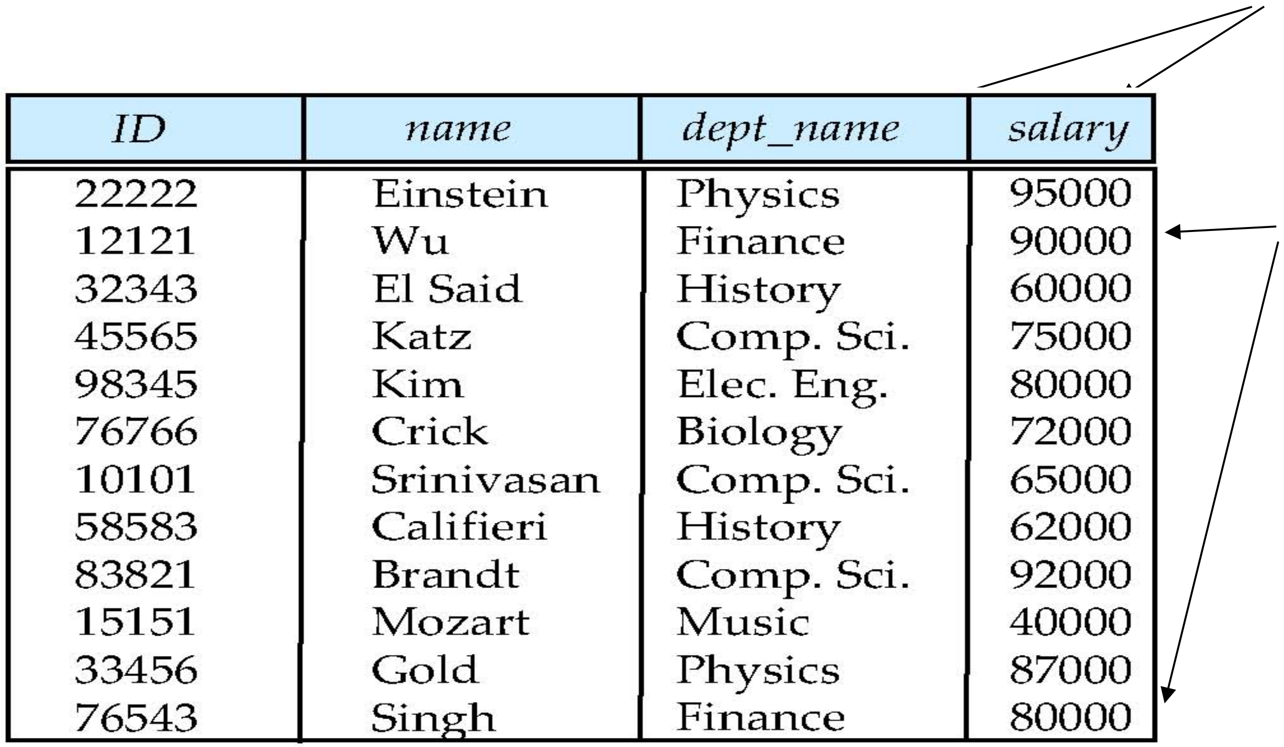
- **Logical Data Independence(逻辑数据独立性)** – the ability to modify the logical schema without changing the user view schema

Data Models（数据模型）

- A collection of tools for describing
  - Data (**数据**)
  - Data relationships(**联系**)
  - Data semantics(**语义**)
  - Data constraints(**约束**)
- **Relational model**(**关系模型**)
- Entity-Relationship(**实体-联系**) data model
- Object-based data models
  - Object-oriented （**面向对象数据模型**）
  - Object-relational(**对象-关系模型模型**)
- Semistructured data model (XML)(**半结构化数据模型**)
- Other older models:
  - Network model （**网状模型**）
  - Hierarchical model(**层次模型**)

Relational Model

- Relational model (**Chapter 2， chapter 6**)
- Example of tabular data in the relational model

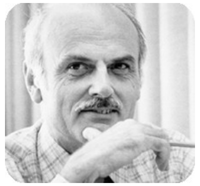


Columns / **Attributes**
(列 /属性)

Rows / **Tuples**
（行 /元组）

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



Ted Codd
Turing Award 1981

Database Languages

- Data Definition Language (DDL)
- Data Manipulation Language (DML)
- SQL Query Language
- Application Program Interface（ API ）

Data Definition Language (DDL) （数据定义语言）

- 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（数据字典）**
- Data dictionary contains **metadata (元数据， i.e., data about data)**
  - Database schema
  - Integrity constraints **（完整性约束）**
    - Primary key (ID uniquely identifies instructors) **（主健）**
    - Referential integrity (**references** constraint in SQL) **（参照完整性）**
      - e.g. dept_name value in any instructor tuple must appear in department relation
  - Authorization **（权限）**

Data Manipulation Language (DML)
（数据操作语言）

- Language for accessing and manipulating the data organized by the appropriate data model
  - DML also known as query language
- Two classes of languages
  - **Procedural（过程式）**－ user specifies what data is required and how to get those data
  - **Declarative (nonprocedural， 陈述式，非过程式)** – user specifies what data is required without specifying how to get those data
- SQL is the most widely used query language

SQL Query Language

- **SQL**: widely used non-procedural language
  - Example 1: Find the name of the instructor with ID 22222

```
select    name
from      instructor
where     instructor.ID = ‘22222’
```

  - Example 2: Find the ID and building of instructors in the Physics dept.

```
select instructor.ID, department.building
from instructor, department
where instructor.dept_name = department.dept_name and
          department.dept_name = ‘Physics’
```

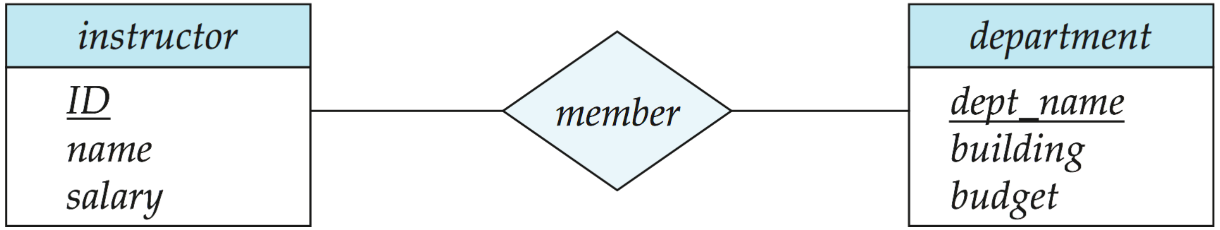
- **Chapters 3, 4 and 5**

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.
- Application programs generally access databases through one of
  - Language extensions to allow **embedded SQL**
  - **API（Application program interface）** (e.g., ODBC/JDBC) which allow SQL queries to be sent to a database

Database Design(数据库设计)

- Entity Relationship Model （实体 - 联系模型）(**Chapter 7**)
  - Models an enterprise as a collection of data *entities* and *relationships*
  - Represented diagrammatically by an *entity-relationship diagram.*



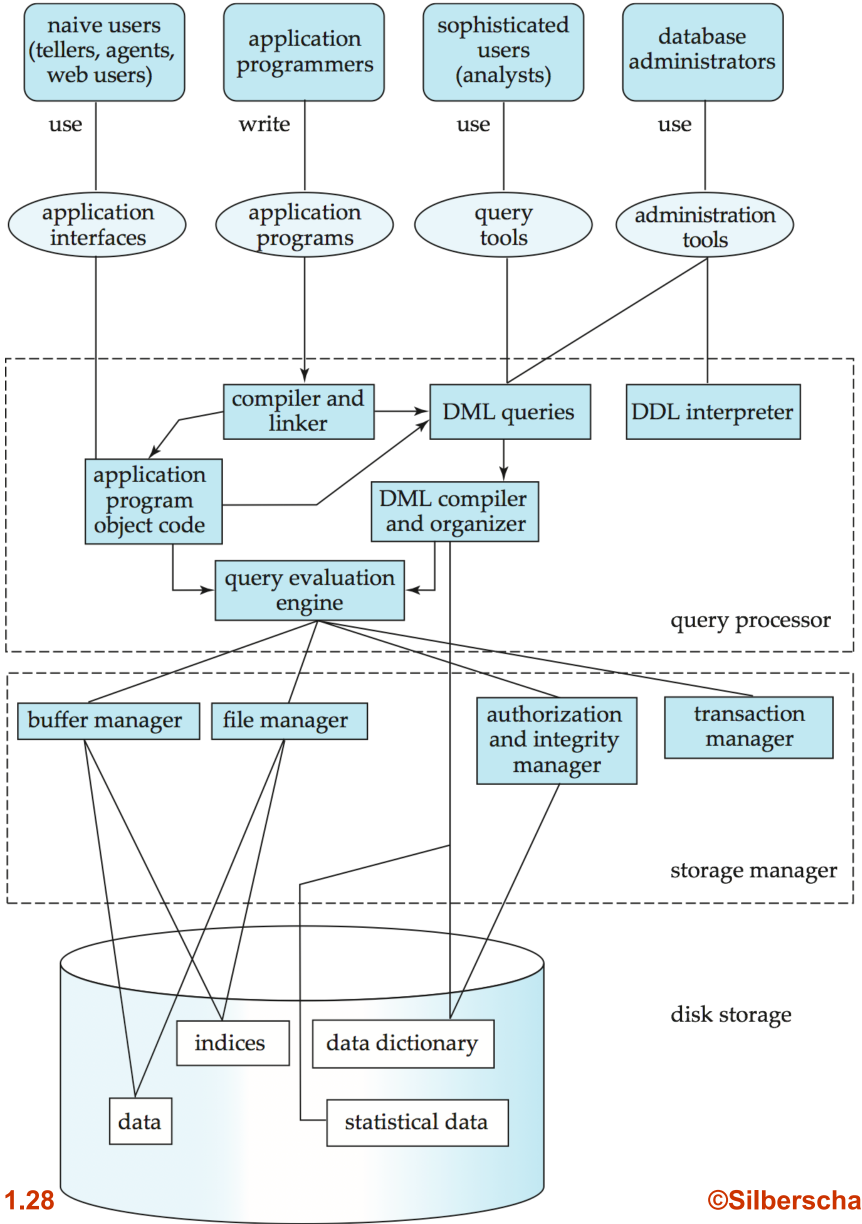
- Normalization Theory （规范化理论） (**Chapter 8**)
  - Formalize what designs are bad, and test for them

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

Is there any problem with this design?

Database Engine(数据库引擎)

- A database system**(database engine)** 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.



naive users (tellers, agents, web users)

application programmers

sophisticated users (analysts)

database administrators

use

write

use

use

application interfaces

application programs

query tools

administration tools

compiler and linker

DML queries

DDL interpreter

application program object code

DML compiler and organizer

query evaluation engine

query processor

buffer manager

file manager

authorization and integrity manager

transaction manager

storage manager

disk storage

indices

data dictionary

data

statistical data

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:
  - **File manager**
  - **Buffer manager**
  - **Authorization and integrity manager**
  - **Transaction manager**

- **Chapters 10 and 11**

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.
  - Statistical data

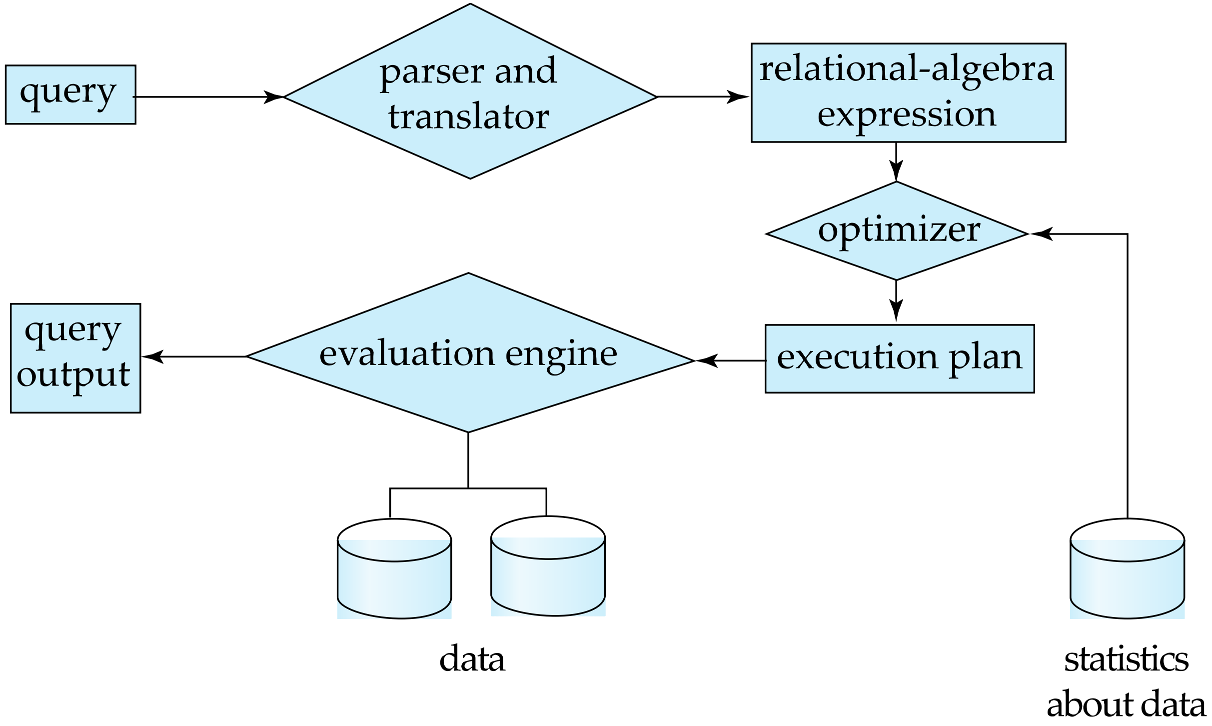
Query Processor

- 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.

- **Chapters 12 and 13**

Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation

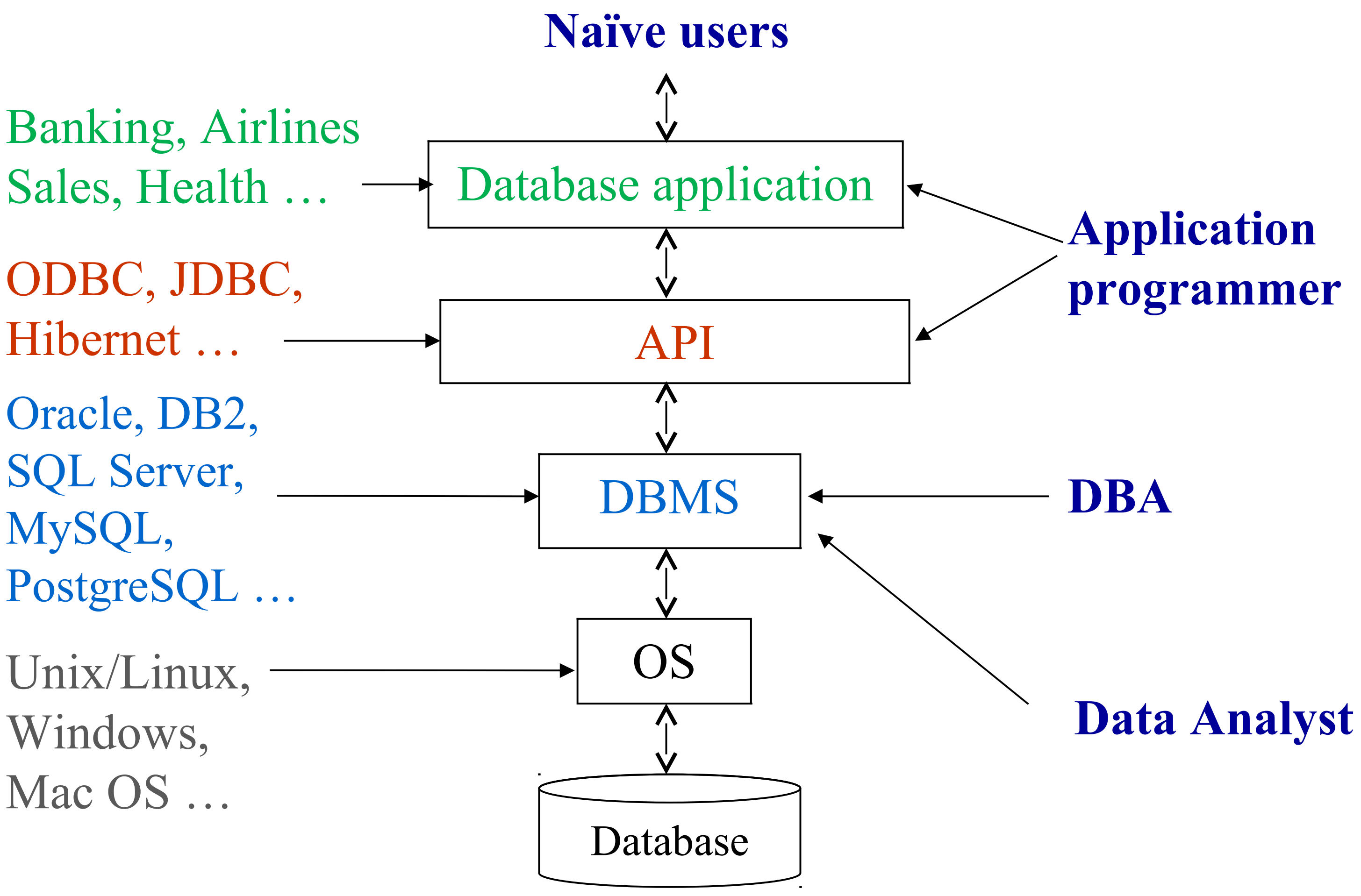


Transaction Management

- A **transaction** is a collection of operations that performs a single logical function in a database application.
- **Recover Manager** 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.

- **Chapters 14, 15 and 16**

Database Users



Naïve users

Banking, Airlines
Sales, Health …

Database application

Application programmer

ODBC, JDBC,
Hibernet …

API

Oracle, DB2,
SQL Server,
MySQL,
PostgreSQL …

DBMS

DBA

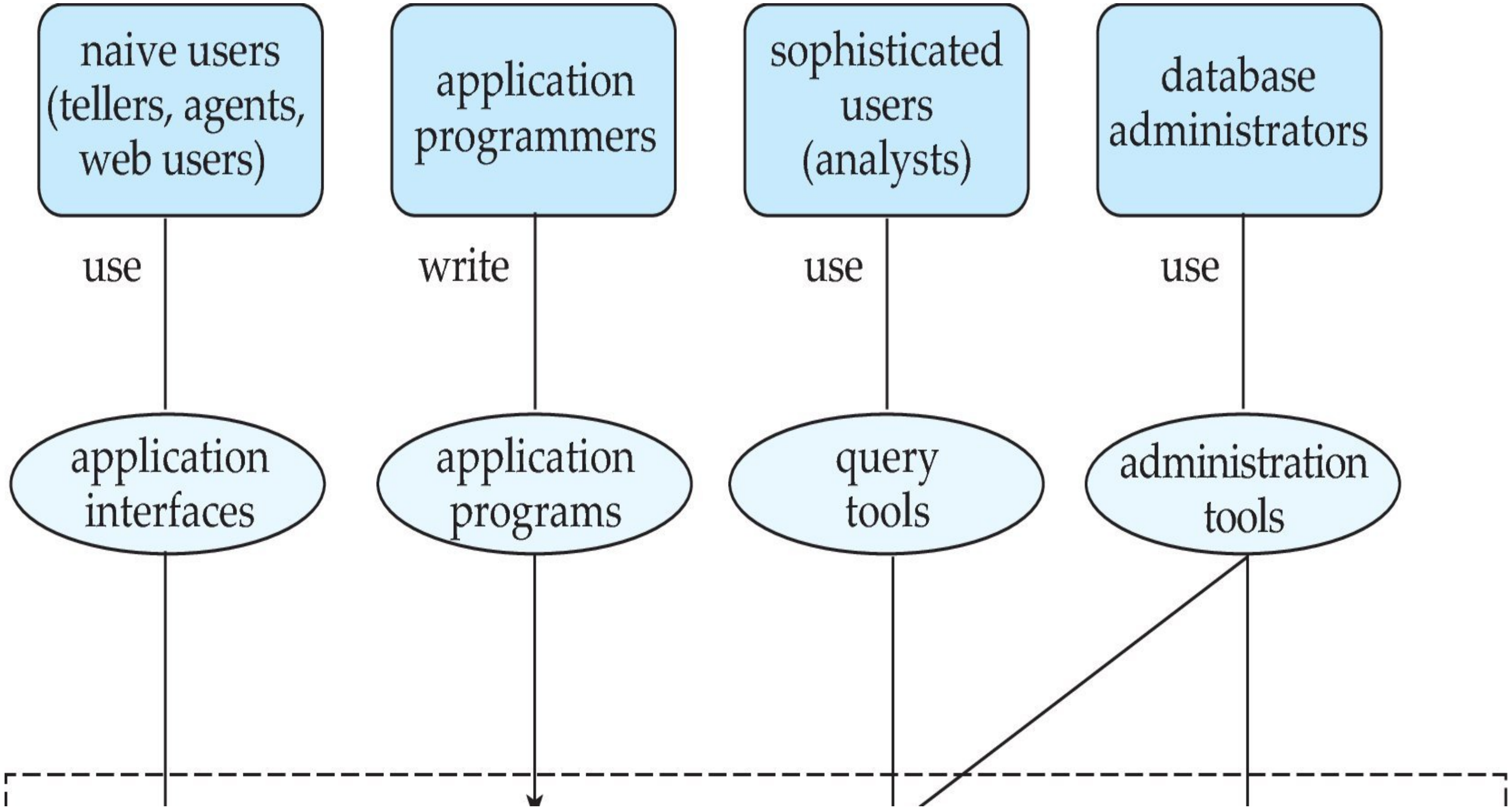
Unix/Linux,
Windows,
Mac OS …

OS

Data Analyst

Database

Database Users



Database

Database Users

Users are differentiated by the way they expect to interact with

the system

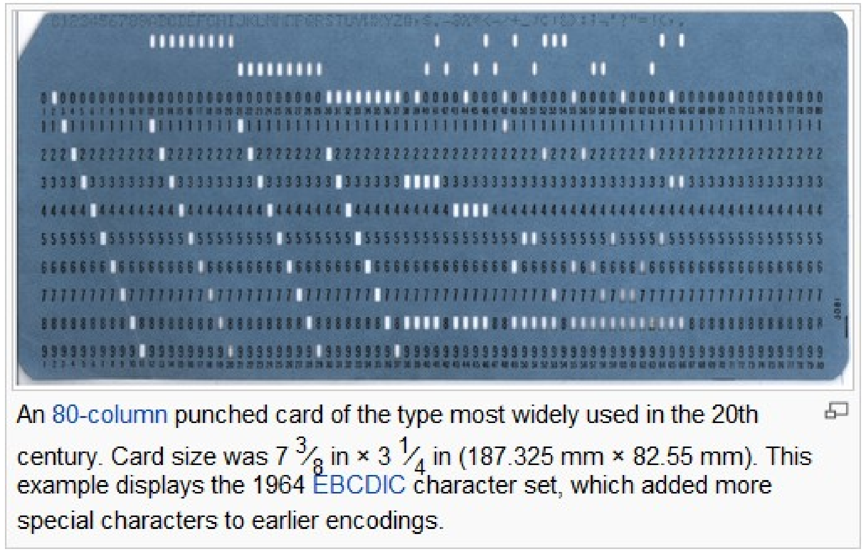
- **Application programmers** – interact with system through DML calls
- **Naïve users** – invoke one of the permanent application programs that have been written previously
  - Examples, people accessing database over the web, bank tellers, clerical staff
- **Database Administrator** - Coordinates all the activities of the database system; the database administrator has a good understanding of the enterprise’s information resources and needs.

Database Administrator （ DBA ）

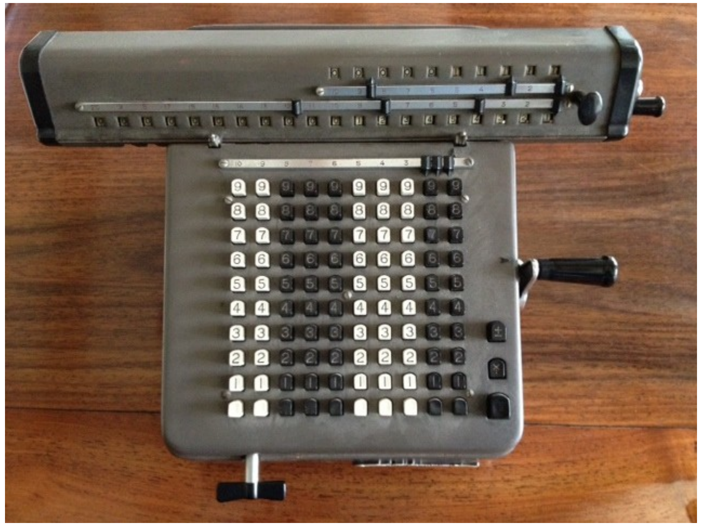
- Database administrator's duties include:
    - Schema definition
    - Storage structure and access method definition
    - Schema and physical organization modification
    - Granting user authority to access the database
    - Specifying integrity constraints
    - Acting as liaison with users
    - Monitoring performance and responding to changes in requirements

History of Database Systems

- 1950s and early 1960s:
  - Data processing using magnetic tapes for storage
    - Tapes provide only sequential access
  - Punched cards for input

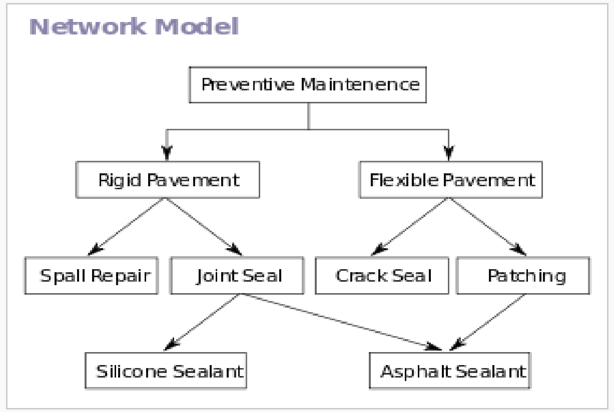


An 80-column punched card of the type most widely used in the 20th century. Card size was $7\frac{3}{8}$ in × $3\frac{1}{4}$ in (187.325 mm × 82.55 mm). This example displays the 1964 EBCDIC character set, which added more special characters to earlier encodings.

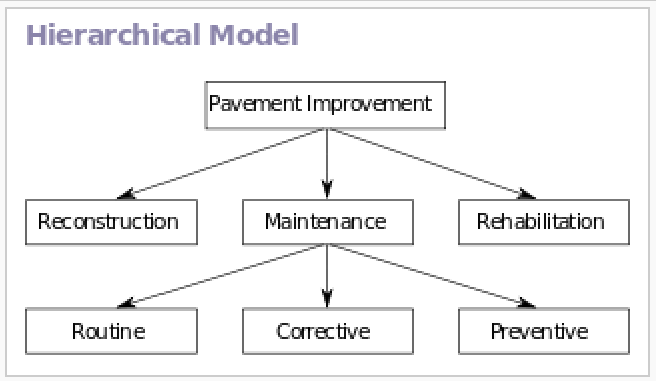


History of Database Systems

- 1960s :
  - Hard disks allow direct access to data
  - Network and hierarchical data models are widely used
  - IDS （ Integrated DataStore), 1961, GE
    - Charles W. Bachman
  - IBM IMS(Information Management System), 1968.



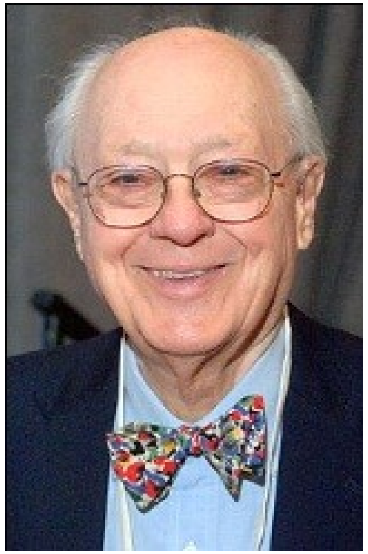
Example of a Network Model.



Example of a hierarchical model

Turing Award:Charles W. Bachman(1924-2017)

- IDS （ Integrated DataStore), 1961, GE(美国通用电气)
- “ father of databases"
- 1973 ACM Turing Award for his outstanding contribution to database technology

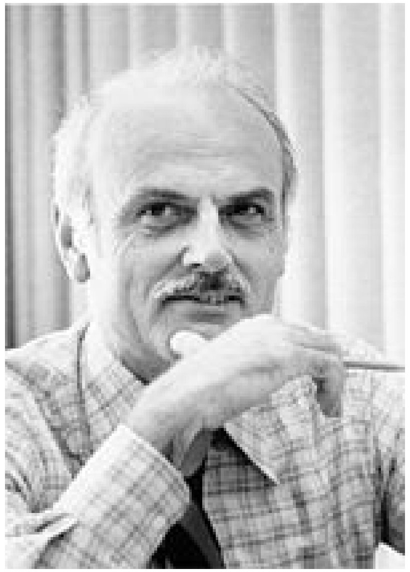


History of Database Systems

- 1970s:
  - Business Applications
    - OLTP (Online Transaction Processing)
  - **Edgar F. Codd** defines the relational data model
  - IBM Research begins System R prototype(1974, **Jim Gray** as a key player)
  - UC Berkeley begins Ingres prototype(1974, leaded by **Michael Stonebraker**)

Turing Award: Edgar F. Codd (1923-2003)

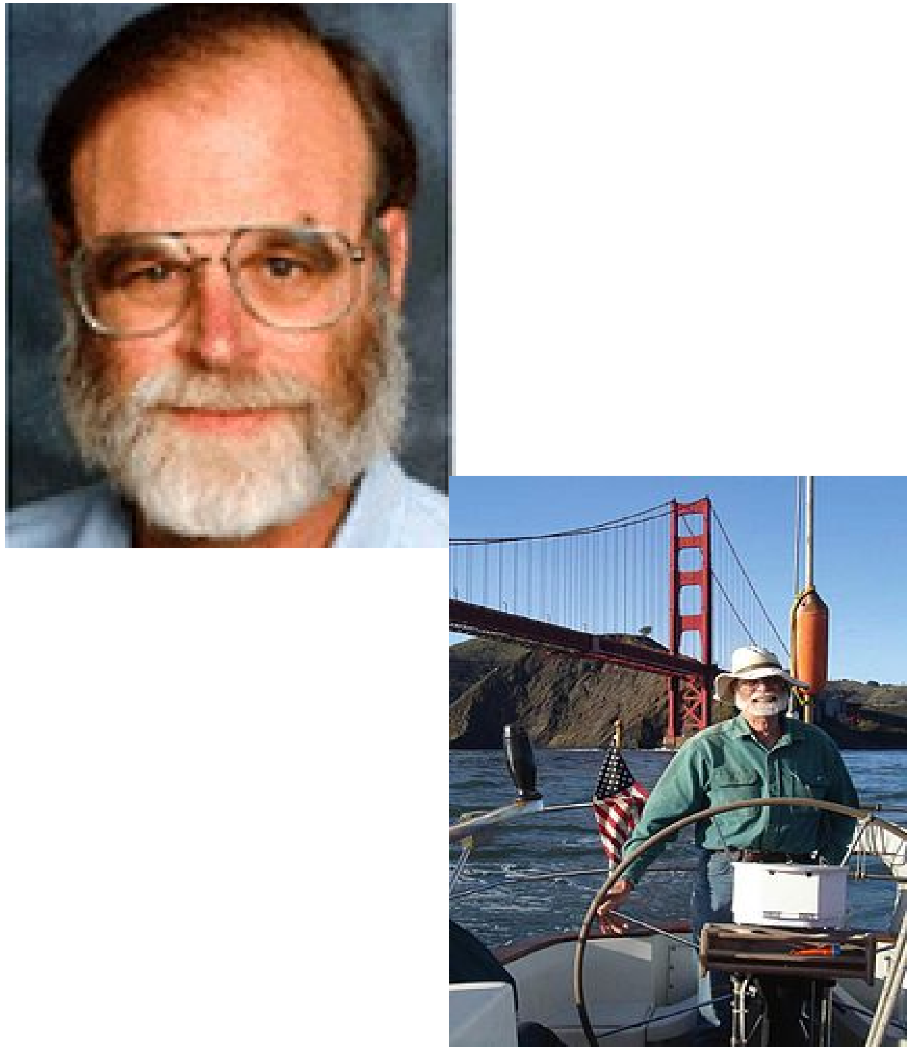
- A Relational Model of Data for Large Shared Data Banks, CACM 1970.
- 1981 ACM Turing Award
- In 2004, SIGMOD renamed its highest prize to the SIGMOD Edgar F. Codd Innovations Award.



History of Database Systems

- 1980s:
  - RDBMS implementation
  - Research relational prototypes evolve into commercial systems
    - Oracle(1983)
    - IBM DB2(1983)
    - Informix(1985)
    - Sybase(1987)
    - Postgres (PostgresSQL,1989)
  - Parallel database systems
  - Distributed database systems
  - Object-oriented database systems
  - Object-relational Database systems
  - Extended to Engineering Applications

Turing Award: Jim Gray(1944-2007)



- IMS 、 System R 、 SQL/DS 、 DB2
- 《Transaction Processing: Concepts and Techniques》
- 1998 ACM Turing Award for his seminal contribution to database and transaction processing research and technical leadership in system implementation.
- SIGMOD Jim Gray Doctoral Dissertation Award
- Disappearance on January 28, 2007 at sea in his sloop Tenacious, during a short solo sailing trip to the Farallon Islands near San Francisco to scatter his mother's ashes.

History of Database Systems

- 1990s:
  - **Business intelligence(BI)**
  - Large decision support and data-mining applications
  - Large multi-terabyte data warehouses
  - OLAP(Online Analytical Processing)
  - Emergence of Web commerce
    - The Web changes everything
    - New workloads – performance, concurrency, availability

History of Database Systems

- 2000s:
  - **Web Era**
    - Big data
  - XML and XQuery standards
  - Automated database administration
  - NoSQL

History of Database Systems

- 2000s: **NoSQL**
  - A **NoSQL(Not Only SQL)** database provides a mechanism for storage and retrieval of data that use looser consistency models than traditional relational databases in order to achieve horizontal scaling and higher availability.
  - **NoSQL** database systems are useful when working with a huge quantity of data (especially big data) when the data's nature does not require a relational model.
  - Some NoSQL DBMSs:
    - MongoDB, Cassandra, HBase

History of Database Systems

- 2010s:
  - NewSQL
  - Cloud database
  - Blockchain
  - Autonomous Database (AI powered Database)

History of Database Systems

- 2010s: **NewSQL**
  - **NewSQL** is a class of modern RDBMSs that seek to provide the same scalable performance of NoSQL systems for OLTP workloads while still maintaining the ACID guarantees of a traditional database system
  - NewSQL: An Alternative to NoSQL and Old SQL for New OLTP Apps
  - Some NewSQL DBMSs:
    - VoltDB, NuoDB, Clustrix, JustOneDB

Turing Award: Michael Stonebraker (1943~)

- 2014 ACM Turing Award for his fundamental contributions to the concepts and practices underlying modern database systems.
- Stonebraker invented many of the concepts that are used in almost all modern database systems.
- Stonebraker brought Relational Database Systems from concept to commercial success, set the research agenda for the multibillion-dollar database field for decades.
- Through practical application of his innovative database management technologies and numerous business start-ups, he has continually demonstrated the role of the *research university in driving economic development*.

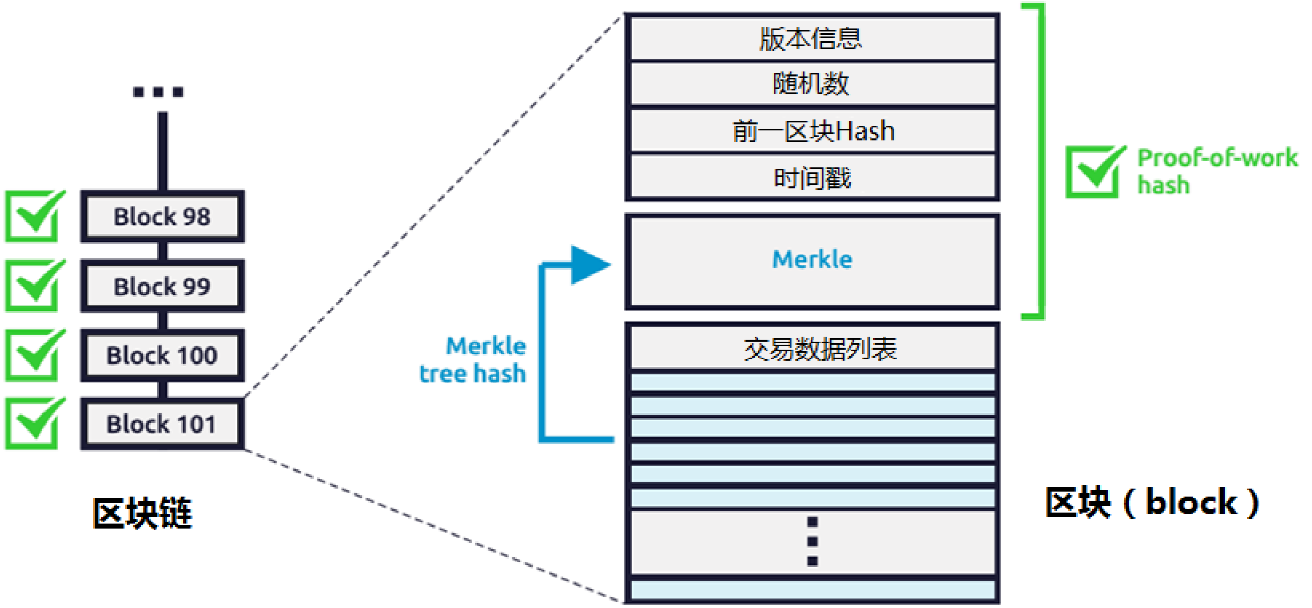
History of Database Systems

- 2010s: **Cloud Database**
  - A cloud database is a database that typically runs on a cloud computing platform, access to it is provided as a service.
  - **Characteristics**
    - Scalability
    - High availability
    - Resource transparency
    - Trustiness
    - Security and privacy

      - Processing Over Encrypted Data
  - **Vendors**
    - Amazon RDS/DynamoDB/SimpleDB
    - Microsoft Azure SQL Database
    - Google Cloud SQL/Bigtable
    - Aliyun PolarDB

History of Database Systems

- 2010s: **Blockchain**
  - A **blockchain** is a distributed database that maintains a continuously-growing ordered list of data blocks hardened against tampering and revision. -Wikipedia



End of Chapter 1