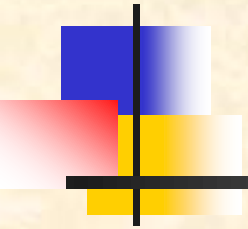


数据库系统原理

Database System Principle

邵蓥侠

Email：shaoyx@bupt.edu.cn

北京邮电大学计算机学院

计算机应用技术中心

北京邮电大学

前言

CONTENTS

1/ 课程目标

- 掌握数据库系统基本原理、基本概念、基本理论和基本方法；
- 熟悉了解一种具体的数据库管理系统的使用方法，能够熟练运用SQL语言来操作数据库。

2/ 教材

- 《Database System Concepts》（第七版）
  Abraham Silberschatz，Henry F.Korth，
  S.Sudarshan 著

- 《数据库系统概念 （第7版）》
  杨冬青，李红燕，唐世渭 等译
  机械工业出版社

3/ 课程内容

Chapter 1: Introduction

Chapter 2: Relation Model

Chapter 3: SQL

Chapter 4: Intermediate SQL

Chapter 5: Advanced SQL

Chapter 6: Database Design Using the E-R Model

Chapter 7: Relational Database Design

Chapter 8: Complex Data Types

3/ 课程内容

Chapter 9: Application Development

Chapter 13: Data Storage Structures

Chapter 14: Indexing

Chapter 15: Query Processing

Chapter 16: Query Optimization

Chapter 17: Transactions

Chapter 18: Concurrency Control

Chapter 19: Recovery System

4/ 学习方法

理论学习：基本概念、原理和方法

实践操作：运用SQL语言操作一种具体的数据库系统

5/ 成绩

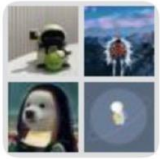
- 平时作业 10%
- 期中练习 10%
- 实验 20%
- 期末成绩 60%

6/ 助教信息

- **助教1 （317班）**
  - 顾希之 guxizhi@bupt.edu.cn
- **助教2 （318班）**
  - 蒲赠霖 pzl_bupt@bupt.cn

6/ 课程线上平台

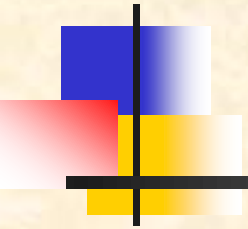
- 爱课堂
  - 课程信息
  - 课件发布
  - 作业发布与提交
- 腾讯会议
  - 会议号：571 7229 1626
- 微信群昵称
  - 班级-姓名



2021 数据库系统原理

该二维码 7 天内 (9月18日前) 有效，重新进入将更新

课程微信群



Chapter 1

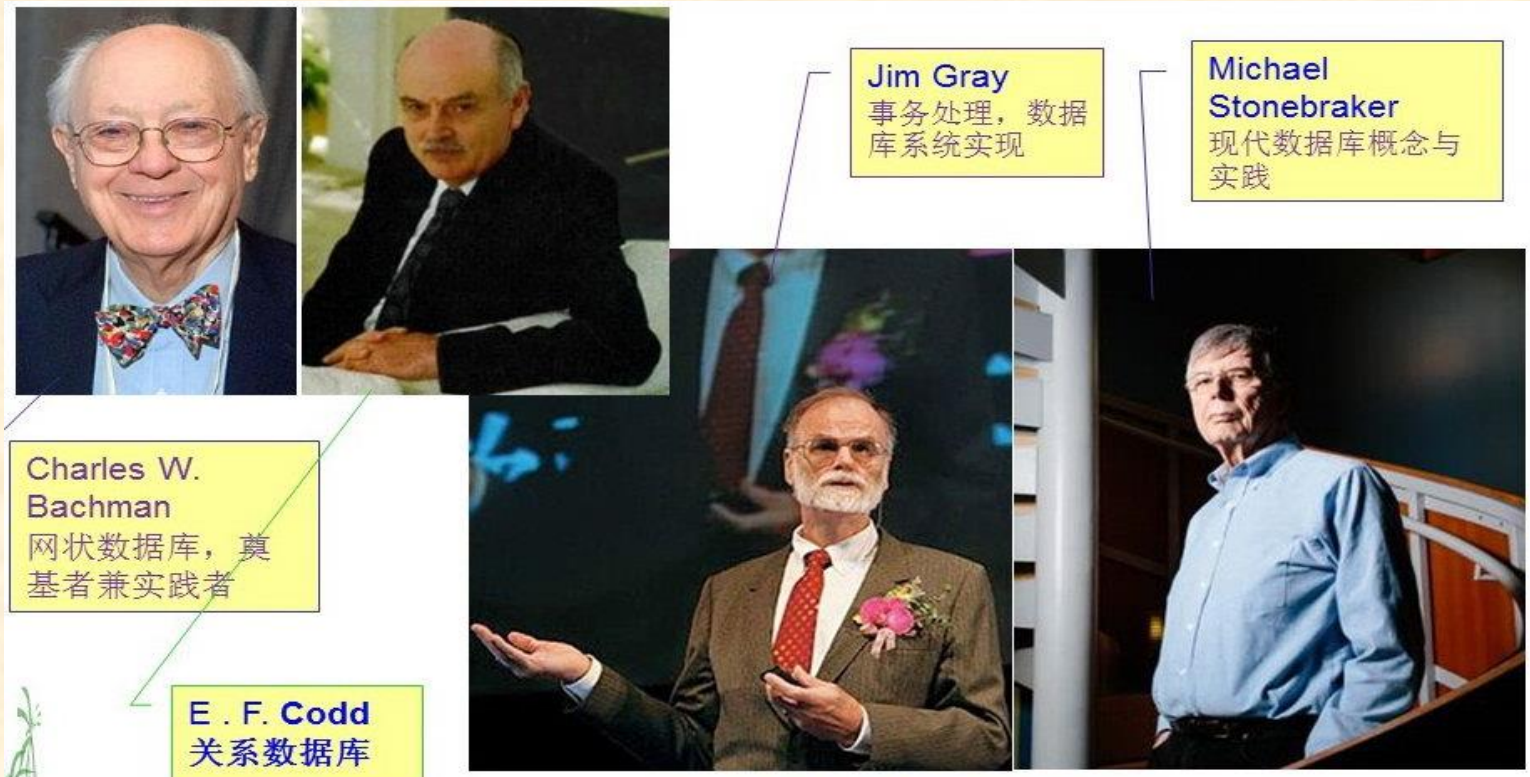
Introduction

数据库的地位

- 数据库技术产生于六十年代末，是数据管理的最新技术，是计算机科学的重要分支。
- 随着信息管理水平的不断提高， 应用范围的日益扩大，信息已成为企业的重要财富和资源。
- 作为管理信息的数据库技术也得到了很大的发展，其应用领域也越来越广泛。
- 数据库技术是信息系统的核心和基础，它的出现极大地促进了计算机应用向各行各业的渗透。

Appendix A
Turing Prize Winners in Database Areas

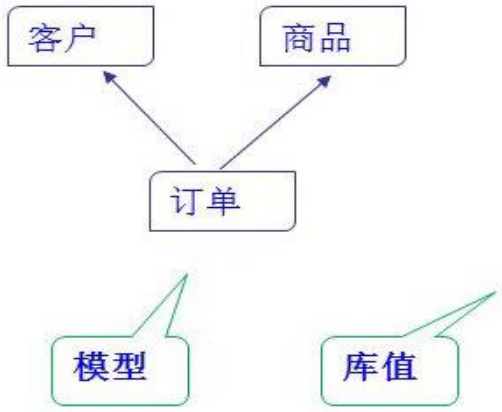
- 图灵奖
  - 计算机界最高奖项，从1966年开始颁奖
  - 每年1位获奖人，涵盖计算机学科各个领域



数据库界的四位图灵奖 得主

Appendix A
Turing Prize Winners in Database Areas(cont.)

- 1973年图灵奖获得者 Charles W.Bachman
  - 网状数据库之父
    - 生于1924年, 主持设计与开发了最早的网状数据库管理系统IDS （ Integrated Data Store），在不惑之年（1964年）正式推出IDS
    - 积极推动与促进了数据库技术标准的制定——DBTG报告与DBTG系统结构
    - DBTG: 提出三级模式，数据库语言分成DDL和DML



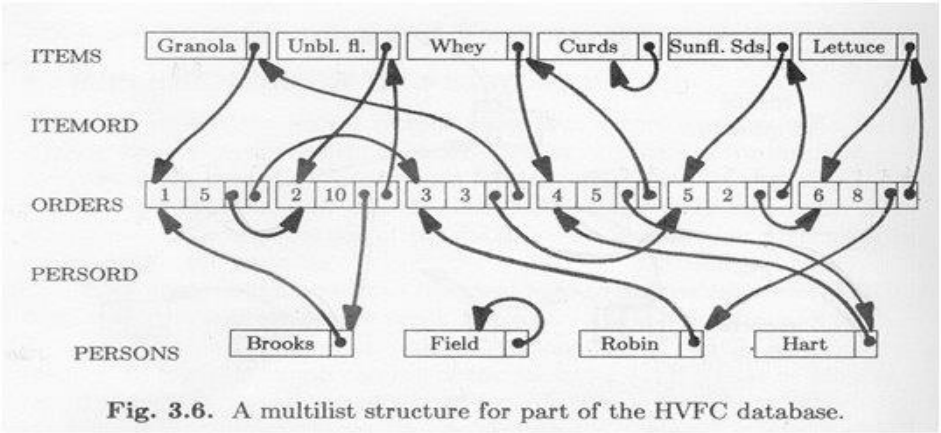
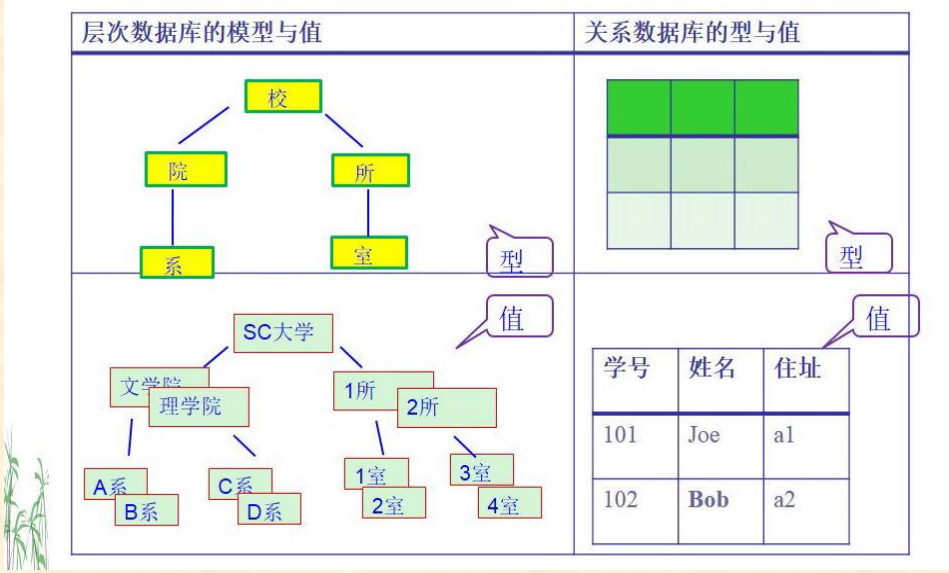


Fig. 3.6. A multilist structure for part of the HVFC database.

Appendix A
Turing Prize Winners in Database Areas(cont.)

- 1981年图灵奖获得者Edgar Frank Codd
  - 关系数据库之父
    - 生于1923年，IBM研究人员
    - “A Relational Model of Data for Large Shared Data Bank ” in *Communication of the **ACM*** in 1970, and relational database theory appeared



Appendix A
Turing Prize Winners in Database Areas (cont.)

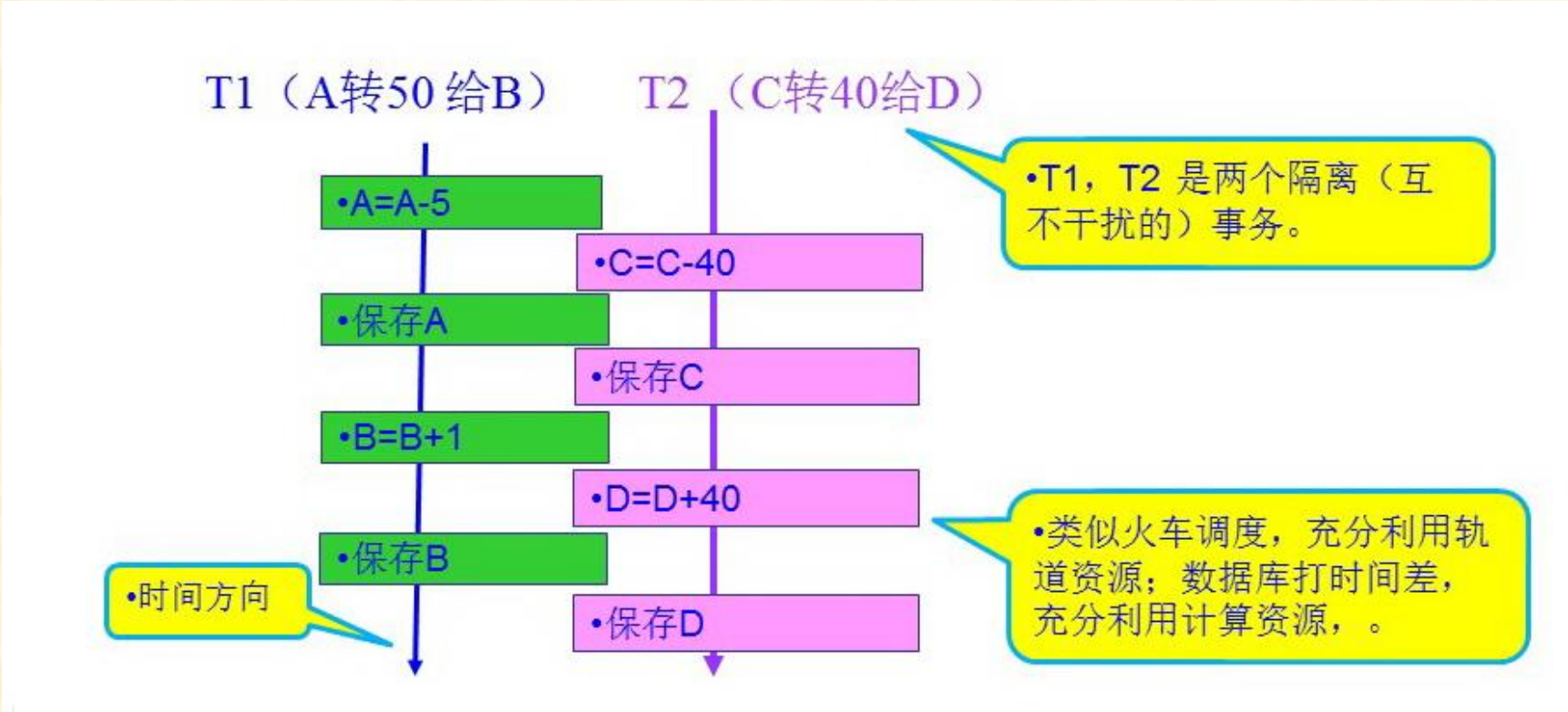
- 发展完善了关系数据库理论，提出了关系代数和关系演算，研究了关系规范化，定义了关系的基本操作，奠定了SQL语言的基础
- IBM 开展 System R项目。推出商用化的DB2和SQL等关系数据库产品

Appendix A
Turing Prize Winners in Database Areas (cont.)

- 1998年图灵奖获得者James Gray
  - 数据库技术和“事务处理”专家
    - 生于1944年，先后供职于Bell, IBM, DEC, Microsoft
    - IBM期间，参与和主持过IMS, System R, SDL/DS, DB2等项目的开发；
    - 在微软主持开发了MS SQL Server 7.0
    - 1998年ACM授予IBM System R软件系统奖6位获奖人之一

Appendix A
Turing Prize Winners in Database Areas (cont.)

- 提出并完善了数据库“事务处理”技术，事务的ACID特性，事务并发控制和死锁处理，事务两阶段提交协议，事务故障恢复机制



Appendix A
Turing Prize Winners in Database Areas (cont.)

- 2014年图灵奖获得者Michael Stonebraker
  - MIT教授，美国工程院院士
  - **“发明了许多几乎应用在所有现代数据库系统中的概念，并且创立多家公司，成功地商业化了他关于数据库技术的开创性工作，对现代数据库系统底层的概念和实践做出基础性贡献”**
    - 1992年提出对象关系数据库模型
    - 众多数据库公司的创始人之一，其中包括**Ingres**、Illustra、Cohera、StreamBase Systems和Vertica等
    - 参与的项目包括：Aurora，C-Store，H-Store，Morpheus，以及SciDB系统

Appendix A
Turing Prize Winners in Database Areas (cont.)

- SQL Server/Sysbase的奠基人
  - 1987年左右，Sybase联合微软共同开发SQL Server。原始代码的来源与Ingres有些渊源。
  - 后来1994年，两家公司合作终止。此时，两家公司都拥有一套完全相同的SQLServer代码
  - 可以认为，Stonebraker教授是目前主流数据库的奠基人

Database System Applications

- Databases and DBS touch all aspects of our lives
  - Enterprise Information
  - Banking and Finance: all transactions
  - Universities: registration, courses, grades
  - Airlines: reservations, schedules
  - Sales: customers, products, purchases
  - Telecommunications, e.g. TD-LTE networks
  - Manufacturing: production, inventory, orders, supply chain
  - Human resources: employee records, salaries, tax deductions

数字化校园信息系统整体架构

数据库

§ 1.0 DB, DBMS, DBS, DBAS

- Definitions *in the textbook*
  - Database (DB)

  — a collection of *interrelated data*

  — *stored in systems as files*
  - Database management system (DBMS)

  — DB, or a collection of *interrelated data*

  — a set of *programs* to access the data in DB
- Database system (DBS)
  - having the same definition as DBMS in the textbook
  - the term DBS and DBMS are used interchangeably in the textbook

§ 1.0 DB, DBMS, DBS(cont.)

- DBMS provides an environment that is both *convenient* and *efficient* for store and retrieve information
  - definition of structures for storage of information
  - data manipulation mechanisms
  - data safety mechanisms

- Definitions *in some other textbooks*
  - Fig. 1.0.1
  - Database(DB)

— a collection of interrelated data , *stored in systems as files*

DBMS


ORACLE
DATABASE
Microsoft
SQL Server
PostgreSQL
TERADATA
MariaDB
MySQL
IBM
DB2
SYBASE
H2
Access
Apache Derby
HIVE
SQLite
HyperSQL

§ 1.0 DB, DBMS, DBS(cont.)

- Definitions *in some other textbooks (cont.)*
  - Database management system (DBMS)

  — a system/mechanism to manage data in DB

  — *or:* set of *programs* to access the data in DB
  - Database system(DBS)

  — DB + DBMS + Users/Administers
  - Database application system

  — DB + DBMS + *Application programs* + Users/Administers

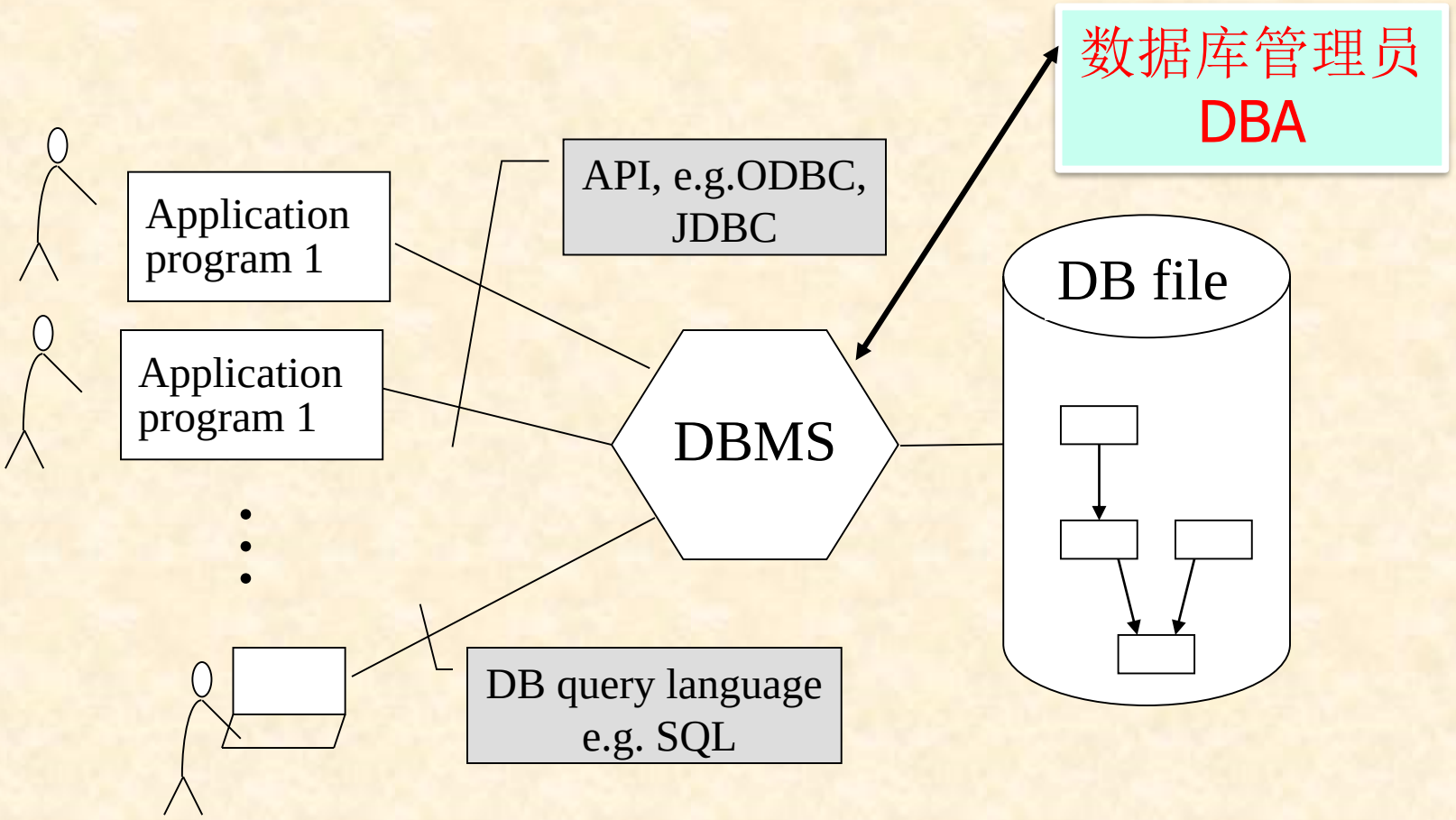


Fig.1.0.1 DBS and DBAS

§ 1.2 Purpose of Database Systems (cont.)

- In the early days, data management applications were built on top of file systems

- Application program examples
  - Add new students, instructors, and courses
  - Register students for courses, and generate class rosters（名册）
  - Assign grades to students, compute grade point averages (GPA) and generate transcripts（成绩单）

§ 1.2 Purpose of Database Systems (cont.)

- Data management based on file systems
  - E.g. **type** *instructor* = **record**

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

    end;

§ 1.2 Purpose of Database Systems (cont.)

- Data management based on file systems
  - FS access needs

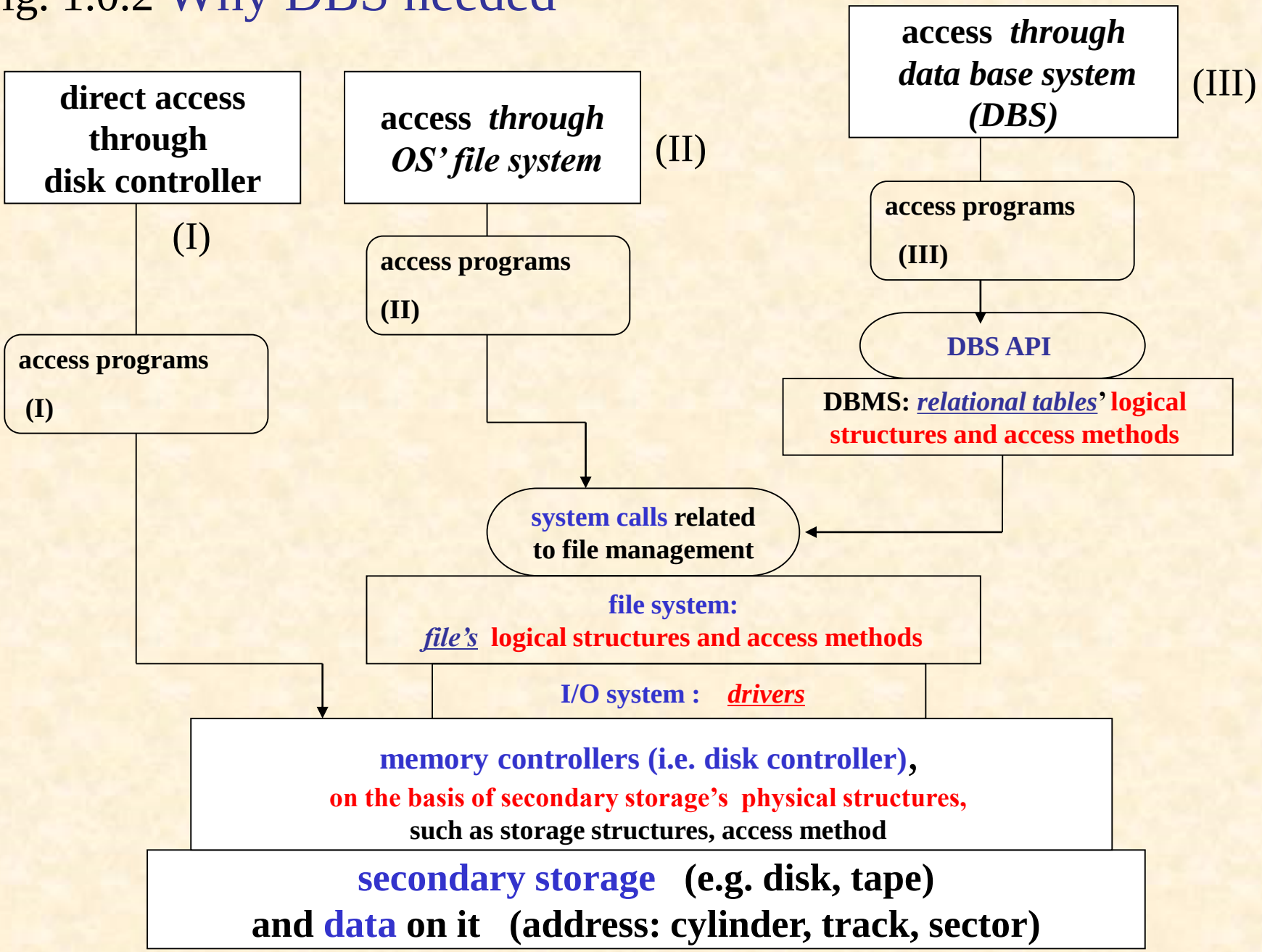
— *file name*, *item name*, length of record, length of item , position of item in the record, storage structure, access method, ….

  - Demerits（缺点）

— application programs are not independent of the file system (or operating system)

— dependency among application programs and data (程序与数据的非独立/依存性）

Fig. 1.0.2 Why DBS needed



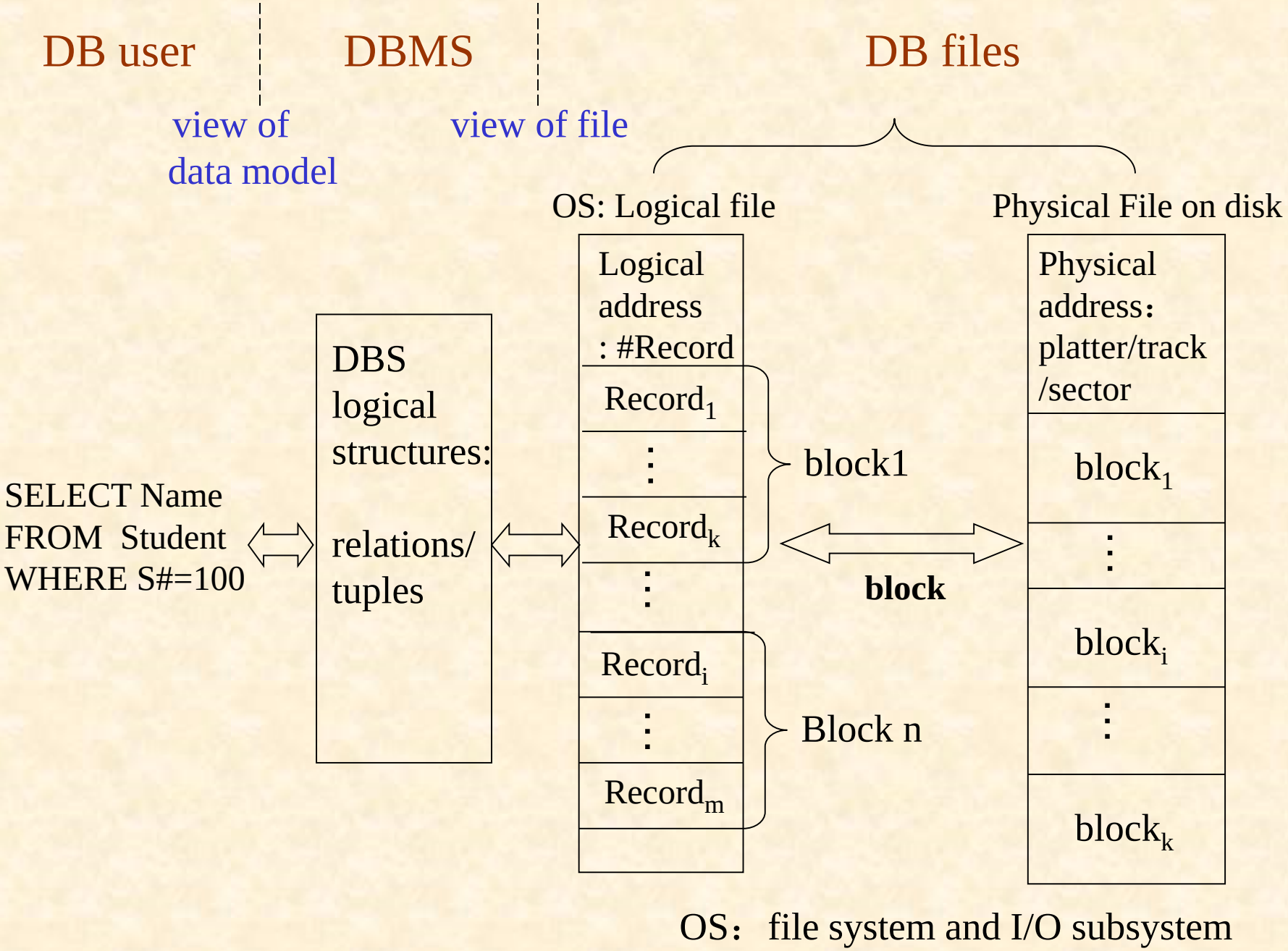
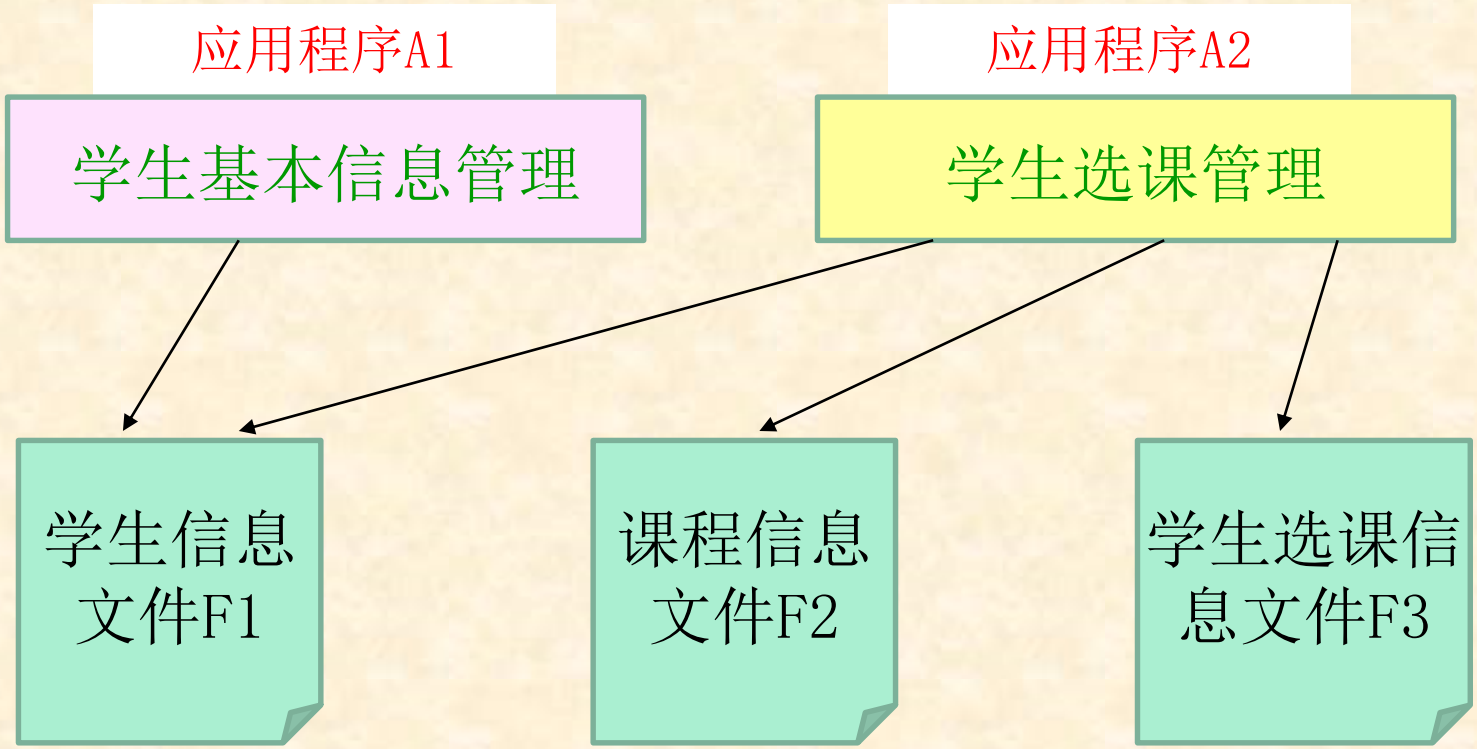


Fig. 1.0.3 Database access

Drawbacks of using file systems to store data

- Data redundancy and inconsistency
  - Multiple file formats, duplication of information in different files
- Difficulty in accessing data
  - Need to write a new program to carry out each new task
- Data isolation（孤立）
  - Multiple files and formats
- Integrity 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

文件管理示例



（学号、姓名、性别、出生日期、联系电话、所在系、专业、班号 ）

（学号、姓名、所在系、专业、课程号、课程名、修课类型、修课时间、考试成绩 ）

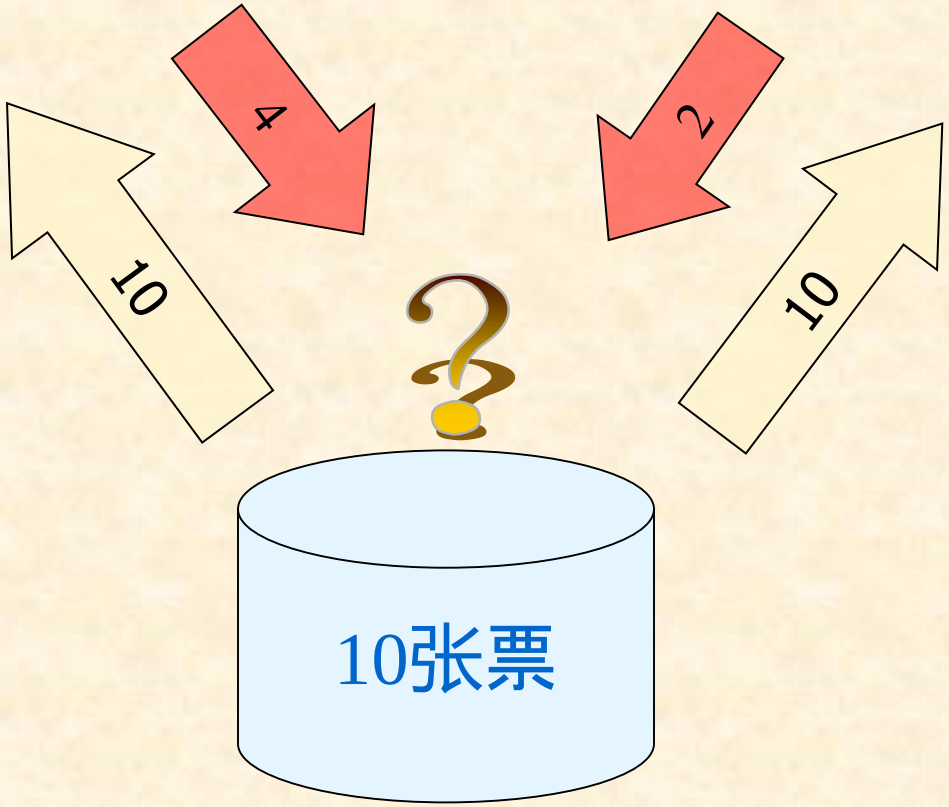
Drawbacks of using file systems to store data (Cont.)

- Atomicity of updates
  - 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
- Concurrent 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
- Security problems
  - Hard to provide user access to some, but not all, data

并发操作示例

订6张票：10-6=4

订8张票：10-8=2



§ 1.2 Purpose of Database Systems (cont.)

- **Database systems offer solutions to all the above problems**

§ 1.3 View of Data

- Basic concepts in DBS theory
  - view of data (数据视图),
    logical/physical/view level data abstract
  - data model, data schema (模式), schema instance
  - logical /physical isolation

1.3.1 Data abstraction

- Data View
  - data *features* from *some* viewpoints
  - e.g. about *student*, his *ID*, *name*, *dept_name*, *tot_cred*, ….
- To simplify and "divide and conquer" the design of complex data application systems
  - describes/design application data in three levels, and obtains three types of schema

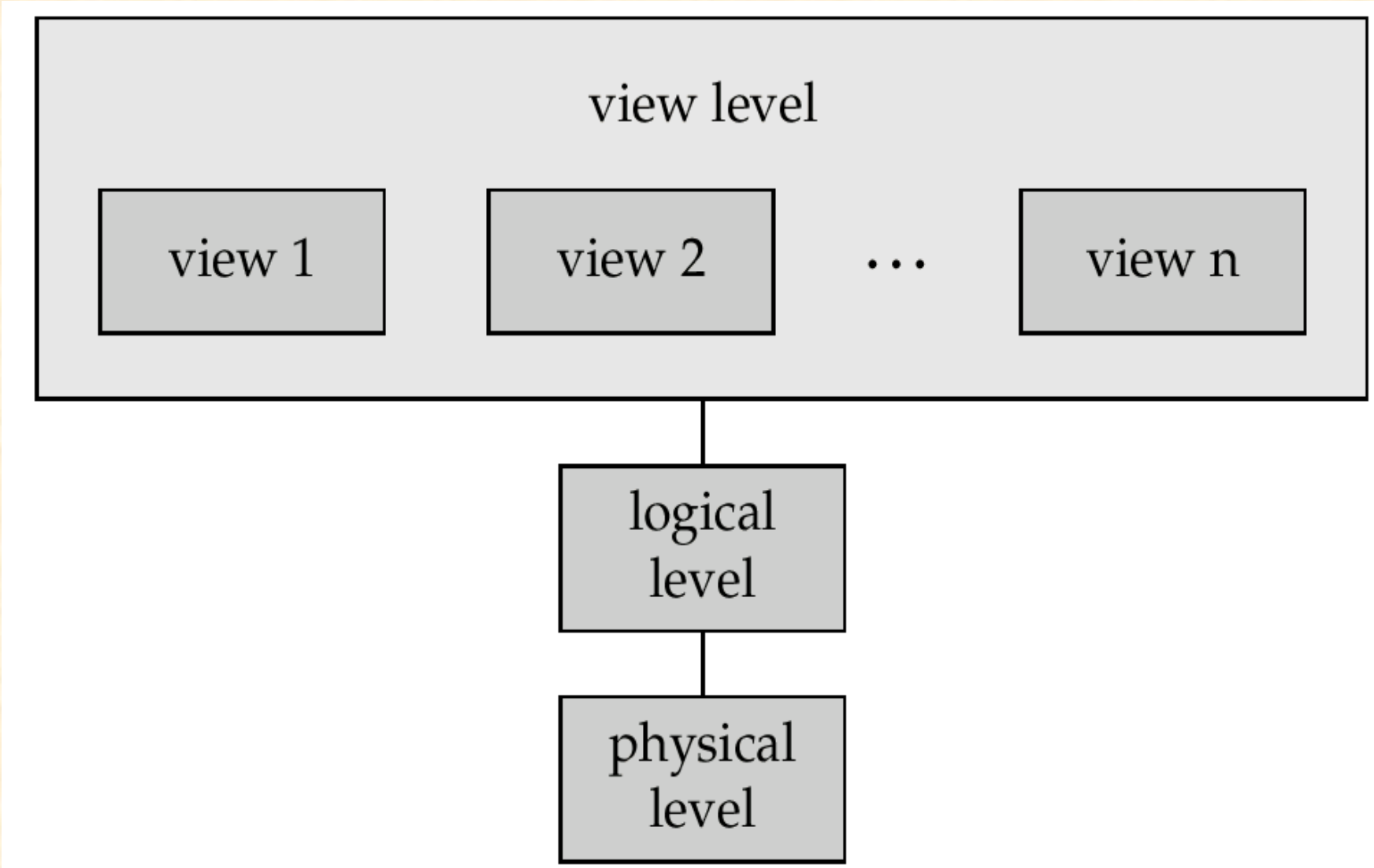


Fig.1.1 Levels of data abstraction

Levels of Abstraction

- **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;

- **Physical level:** describes how a record (e.g., instructor) is stored.
- **View level:** application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.

1.3.1 Data abstraction (cont.)

- View level
  - describe data form
    - how the data items in DB are used by ***different*** users
  - several views for one datum
  - description results
    - external schema（外模式）={view}, set of views
  - description procedure/Logical *DB design*
    - view abstraction
  - merits: application programs are programmed according to views.

Views can also hide information (e.g., salary) for security purposes.

1.3.1 Data abstraction (cont.)

- E.g. University database in Fig.2.8 in page 45
  - *more than one* viewpoints, view integration, for a data object

    view1: *student*<id, name>

    view2: *student*<id, dept_name>

    view3: *student*<id, tot_cred>

1.3.1 Data abstraction (cont.)

Logical level

- implementation-oriented, describes how the data items are stored in DBS (e.g. as *relational tables*), and what relationships exist among those data :

type *instructor* = **record**

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

end;

- description results
  - logical schema (逻辑模式）， e.g. relational tables
- description procedure/*Logical DB design*
  - logical abstraction
- Merits: hiding physical implementation details

1.3.1 Data abstraction (cont.)

- Physical level
  - describes how the data (e.g., ***instructor*** record/table) are actually stored in files (or in secondary storage, e.g. disk)
  - description results
    - physical/internal schema (物理模式，内模式）
    - i.e. storage structure and access methods, such as *index*, *physical blocks*, *access methods* for secondary memory, etc.
  - description procedure/*Physical DB design*
    - physical abstraction

1.3.2 Instances and Schemas

- An ***instance*** of the database (数据库实例)
  - the collection of information stored in the database at a particular moment *(P12 in the textbook)*
  - the content of the database *at a particular point in time*
- **Schema**
  - *overall design of the database (P12 in the textbook)*
  - the structure of data organization in databases
  - physical schema
    - database design at the physical level
  - logical schema
    - database design at the logical level
- Similar to types and variables in programming languages

1.3.3 Data Model

- Data descriptions/abstractions in three levels must obey three types of specification（规格）, i.e. three types of data models

- Definition of data model (P8)
  - a collection of *conceptual tools* for describing
    - data
    - data relationships
    - data semantics
    - consistency constraints

  } integrity constraints

- 程序 = 数据结构 + 算法

1.3.3 Data Model (cont.)

- Another definition of data model
  - DBS中对数据组织与管理/使用方式的抽象描述，包括
    - 数据组织的语法定义，如数据项、数据项间的联系
    - 数据组织的语义定义，如完整性约束
    - 数据操作 *(note: only in some data models, e.g. relational data model)*

Data Models

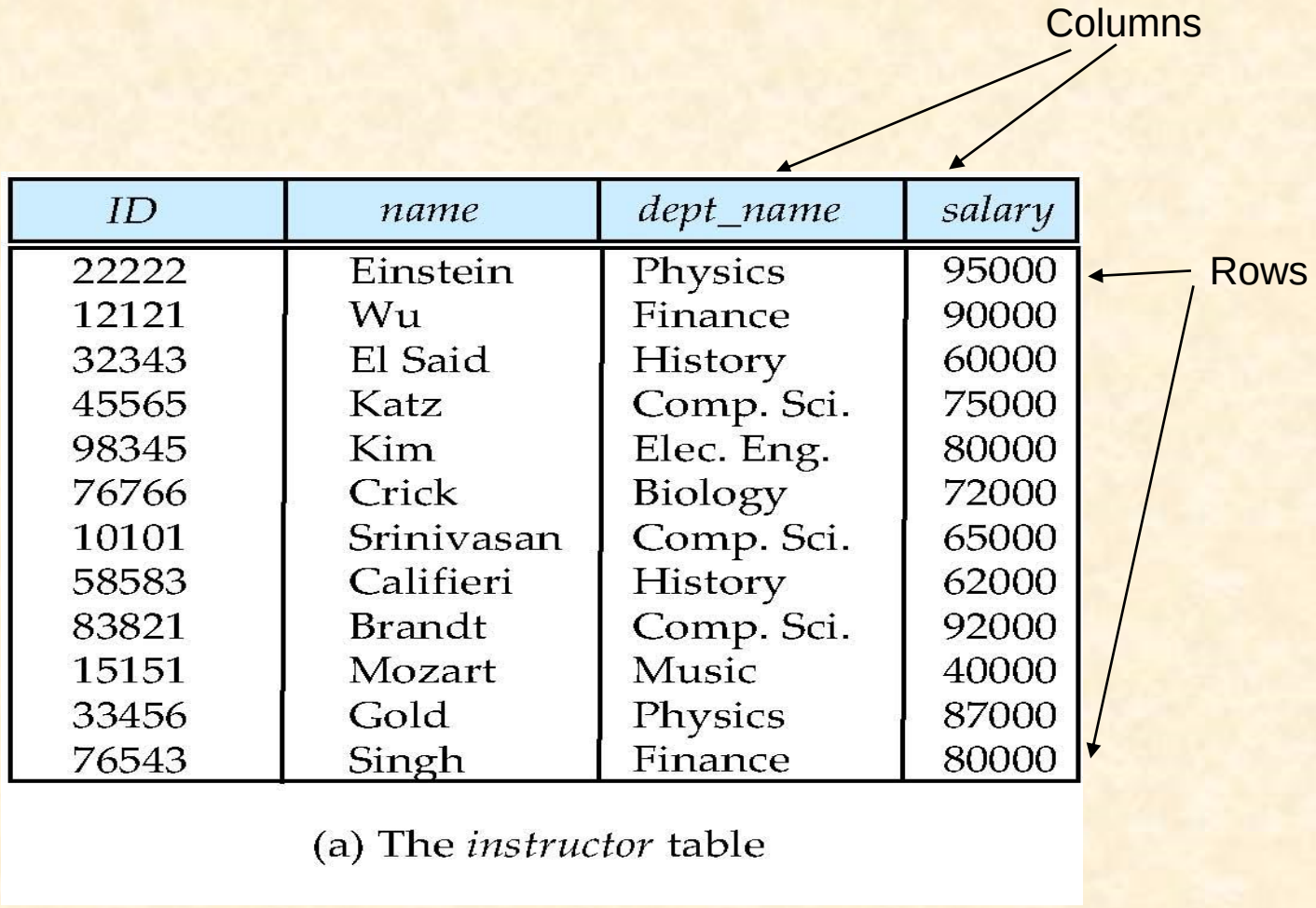
- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- Relational model
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semistructured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model

1.3.3 Data Model (cont.)

- Relational model (关系模型)
  - using a collection of *tables* to represent both data and the relationships among those data

Relational Model

- All the data is stored in various tables.
- Example of tabular(表格的) data in the relational model



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

A Sample Relational Database

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

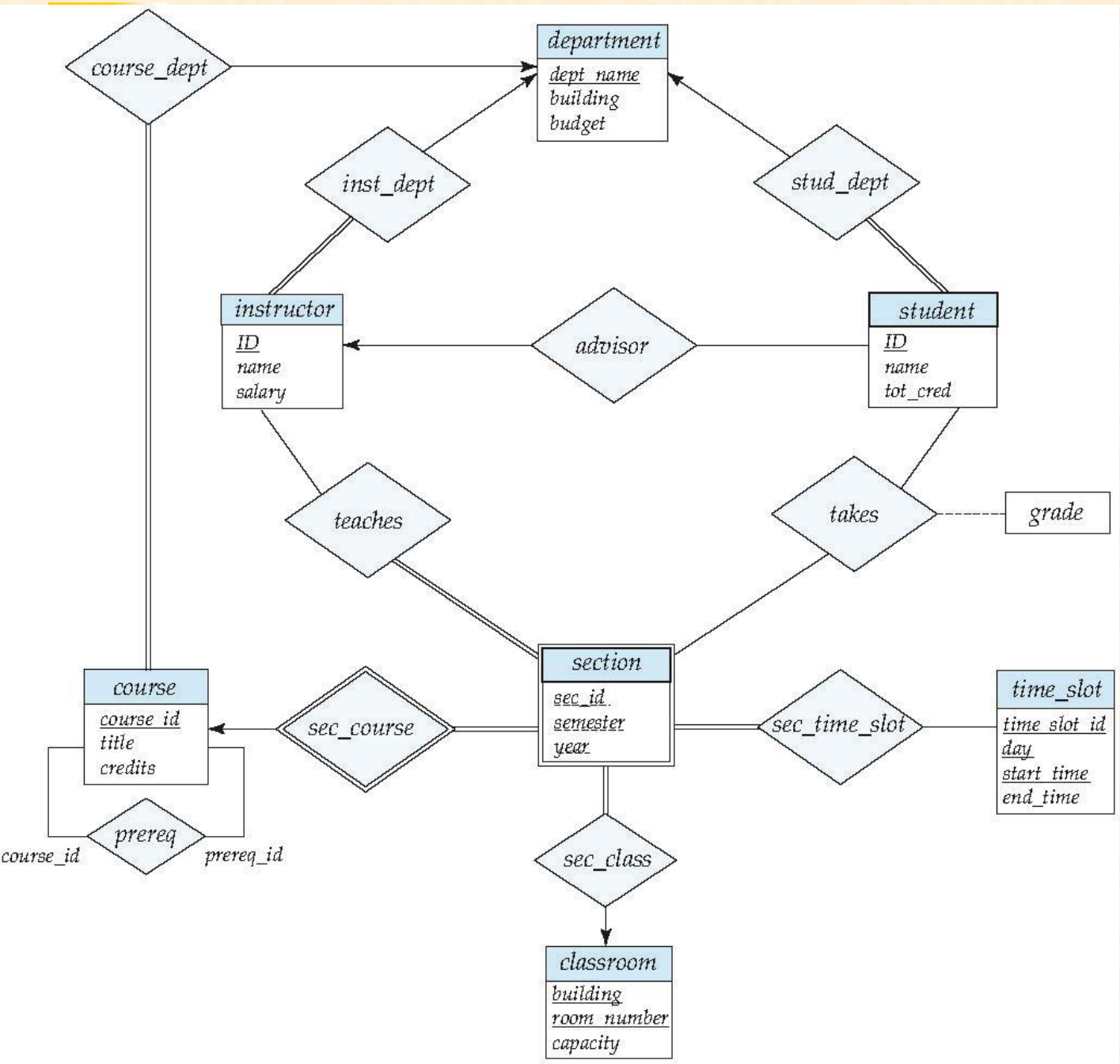
dept_name	*building*	*budget*
Comp. Sci.	Taylor	100000
Biology	Watson	90000
Elec. Eng.	Taylor	85000
Music	Packard	80000
Finance	Painter	120000
History	Painter	50000
Physics	Watson	70000

(b) The *department* table

1.3.3 Data Model (cont.)

- Entity-Relationship model (实体-联系模型)
  - using the ***entity*** and the ***relationship*** to model the *objec*t and the *association* among objects

E-R Diagram for a University Enterprise



1.3.3 Data Model (cont.)

- Network model and hierarchical model
  - only used for early DBS
- Other models
  - object-oriented model
  - semi-structured data models, e.g. XML (Chapter 30)

Model, Schema, Instance

- Data **model**, data **schema** (of data model), **instance** of schema

E.g.

- relational data model : $R=\{ <a_1, a_2, \ldots, a_n> \}$
- relational data schema : customer=<c_name, c_id, street, city>
- instance of schema : <Wang_Li, 1001, Haidian, Beijing >
- *similar to* data types declaration, variables of data types, and values of variables

1.3.2 Data Independence

- 3种数据抽象、3种模式→DBS中2种数据独立性
  - 逻辑模式→内模式间的映射 与physical data independence
  - 外模式→逻辑模式间的映射 与logical data independence
  - refer to Fig.1.0.4

- **Logical data independence**
  - when DBS's logical schemas at logical level (DBMS-oriented) change, application-oriented external schema or application programs do not change
  - e.g. *student=<id, name, dept_name, tot_cred>* is extended into *student=<id, name,* ***age****, dept_name, tot_cred>*

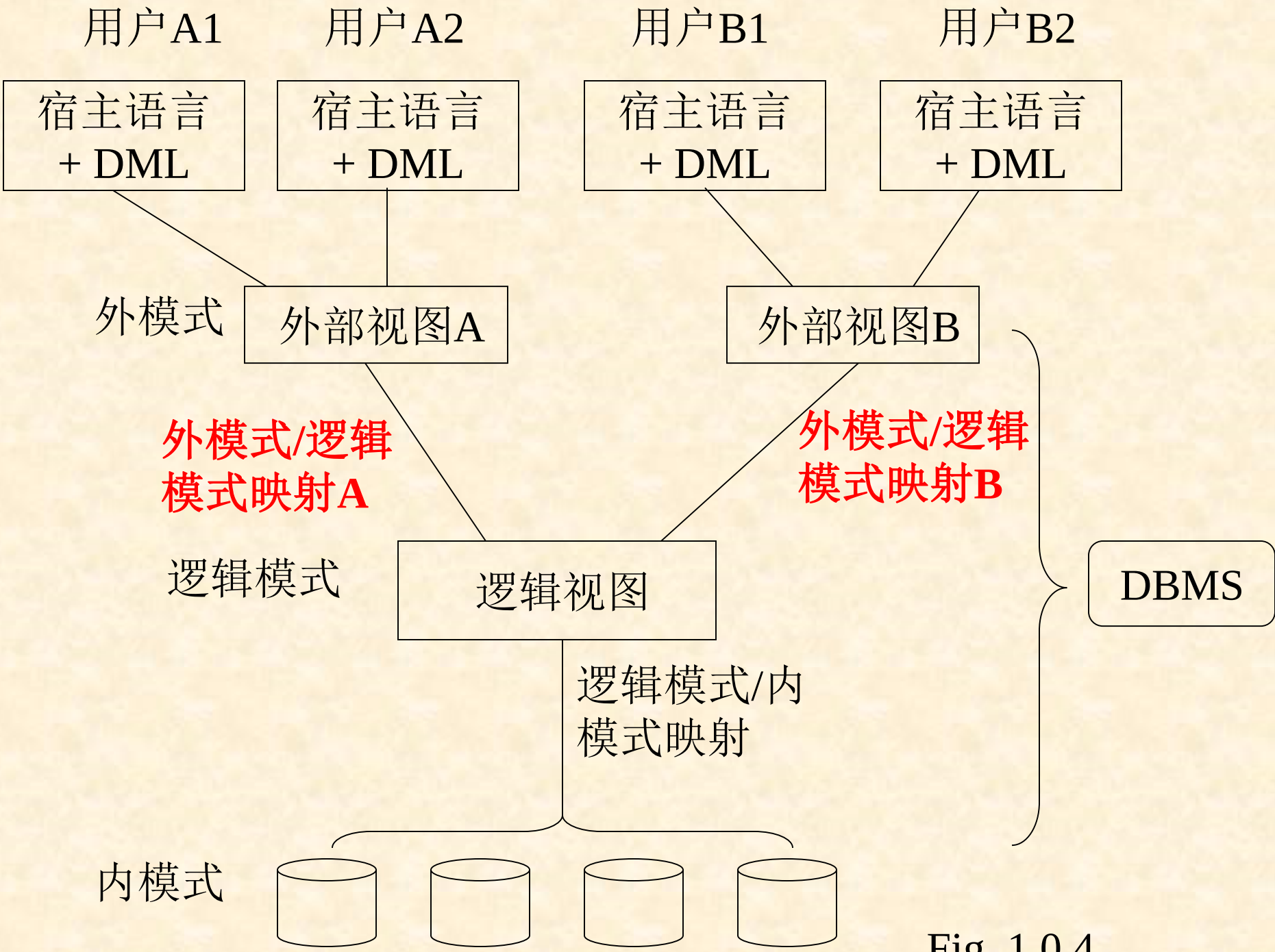

用户A1
用户A2
用户B1
用户B2
宿主语言
+ DML
宿主语言
+ DML
宿主语言
+ DML
宿主语言
+ DML
外模式
外部视图A
外部视图B
外模式/逻辑
模式映射A
外模式/逻辑
模式映射B
逻辑模式
逻辑视图
DBMS
逻辑模式/内
模式映射
内模式

Fig. 1.0.4

1.3.4 Data Independence (cont.)

- **Physical data independence**
  - application programs do not depend on DBS's physical schema, i.e. when physical schema (e.g. index, block-size) change, application programs remain unchanged
    - the ability to modify the physical schema without changing the external/logical schemas, on which application programs depend

§ 1.4 Database Language

- Database Languages as (*user-machine*) *interfaces*
  - data manipulation language, DML
  - data definition language, DDL

1.4.1 Data manipulation language

- DML is the language that enables users to access or manipulate the data as organized by the appropriate data model

- Types of DBS access
  - ***retrieve , insert, delete, modify***

1.4.1 Database Manipulation Language (cont.)

- **Query(查询）**
  - a statement requesting the retrieval of information
- DML also known as **query language**

- Two types of DML
  - procedural (过程性) – user specifies what data is required and how to get those data
  - nonprocedural (说明性，declarative) – user specifies what data is required without specifying how to get those data

Data Manipulation Language (DML)

- 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

- SQL is the most widely used ***non-procedural* query language**
  - others are QE, Datalog
- 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

SQL

- Example
  - to find all instructors in Comp. Sci. dept with salary > 80000

select *name*
from *instructor*
where *dept_name* = ‘*Comp. Sci.*'
\ and *salary* > *80000*

1.4.2 Database 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))

-

1.4.2 Database Definition Language

- 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

§ 1.5 Relational Database

- Refer to Chapter 2

1.5.4 DB Access for Application Programs

- Application programs generally access databases in two methods
  - application program interfaces (API, e.g. ODBC, JDBC, ADO, …), which allow SQL queries to be sent to a database
    - Fig. 1.0.5
  - embed DML call within ***host language*** (宿主语言, such as C and Java)

1.5.3 DB Access for Application Programs (cont.)

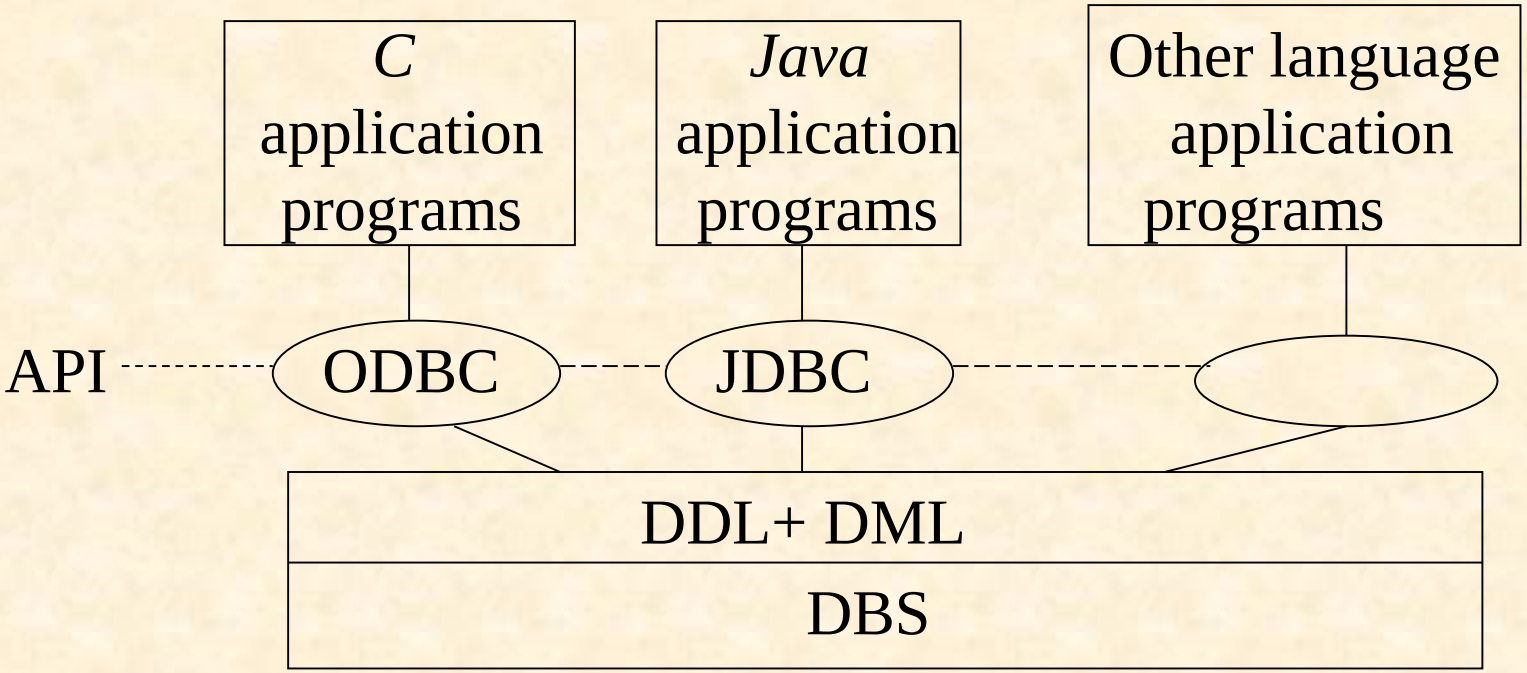
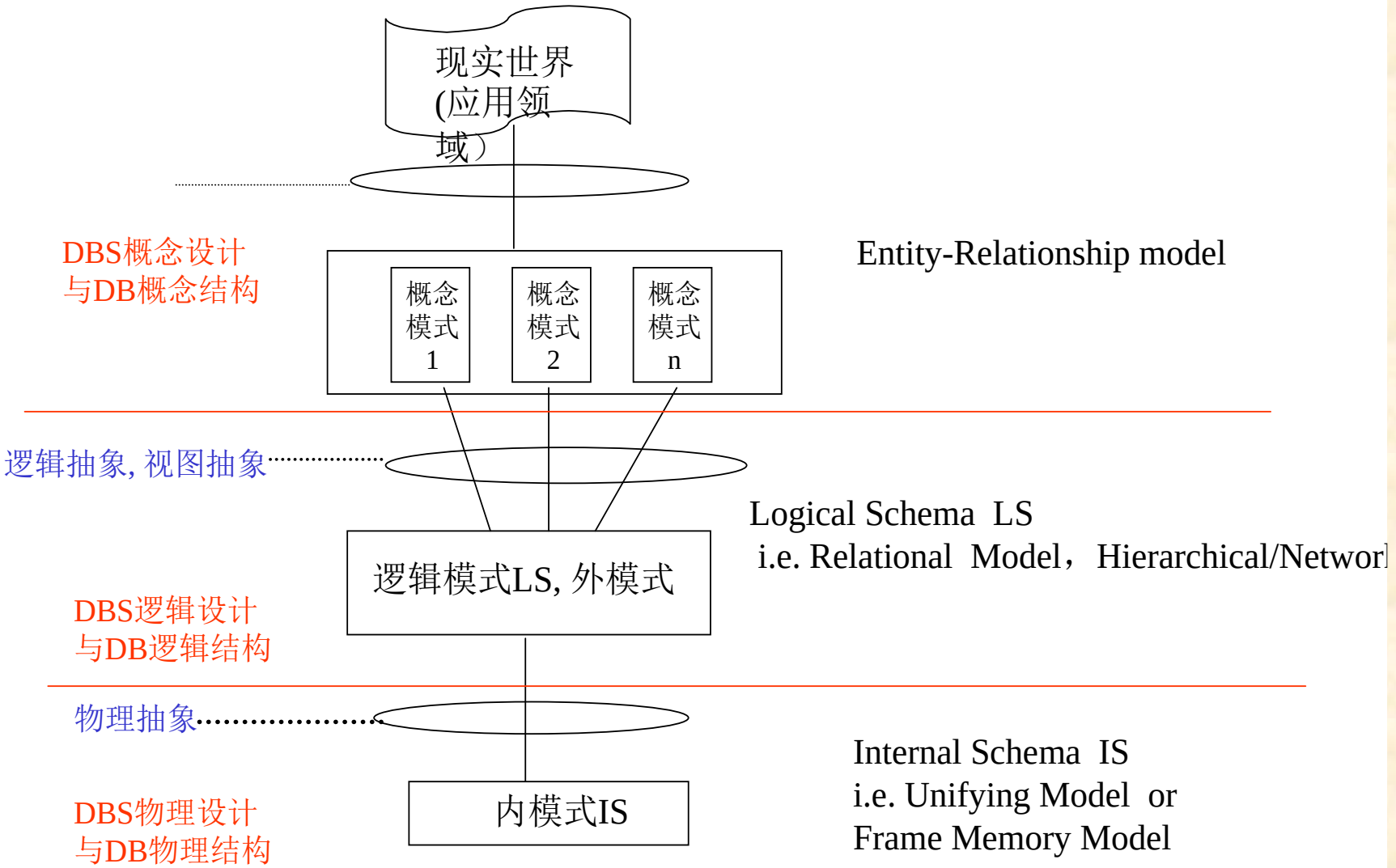


Fig.1.0.5 DBS access interfaces

§ 1.6 Database Design

1.6.1 Design process

- 从保持data independence（数据无关性/独立性）角度出发，根据data models所定义的数据规范形式，在view、logical 、physical三个层次，采用三种data abstraction方法，通过DB **概念设计、DBS逻辑设计、 DBS物理设计三个阶段**，构造面向具体应用领域的DBS的 external schema、logical schema 、internal schema的集合, 从而得到conceptual DBS 、logical DBS 、physical DBS的设计结果
- As described in Fig.1.0.6
  - conceptual design
  - logical design
  - physical design



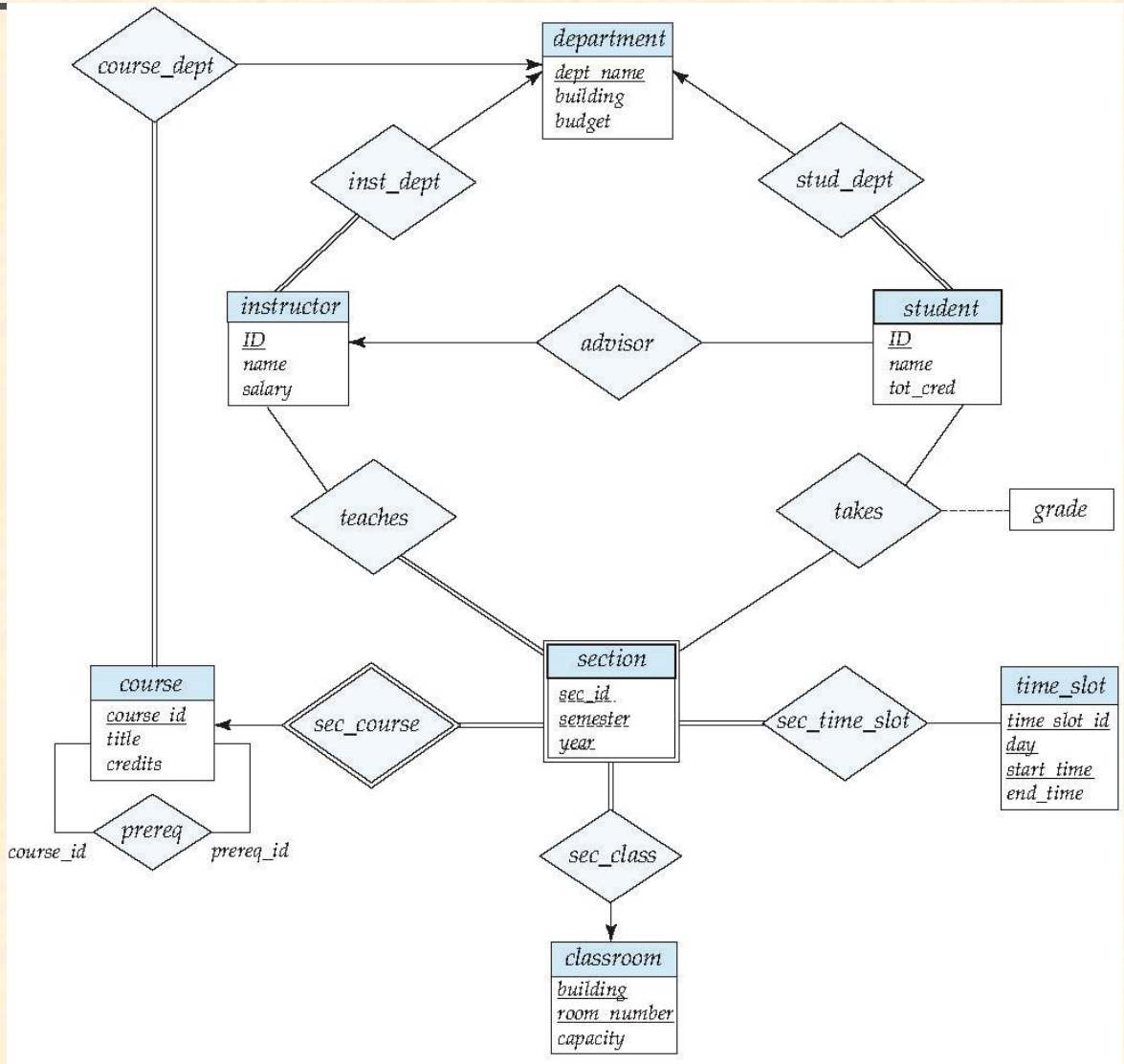
DBS设计与设计阶段： 数据模型—数据抽象—数据模式

Fig. 1.0.6 Data abstraction and Schema

1.6.4 Normalization in Database Design

- 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

1.6.4 Normalization in Database Design



Normalization in Database Design

- Is there any problem with this relation?

instructor + department, 信息冗余→

插入/删除/更新 异常、低效

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

1.9 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

DBS Architecture

- DBS = User/DBA + DBMS + DB
- DBMS
  - query processing
  - storage manager
- DB files
  - application data
  - data dictionary/directory
  - indices
  - statistic data

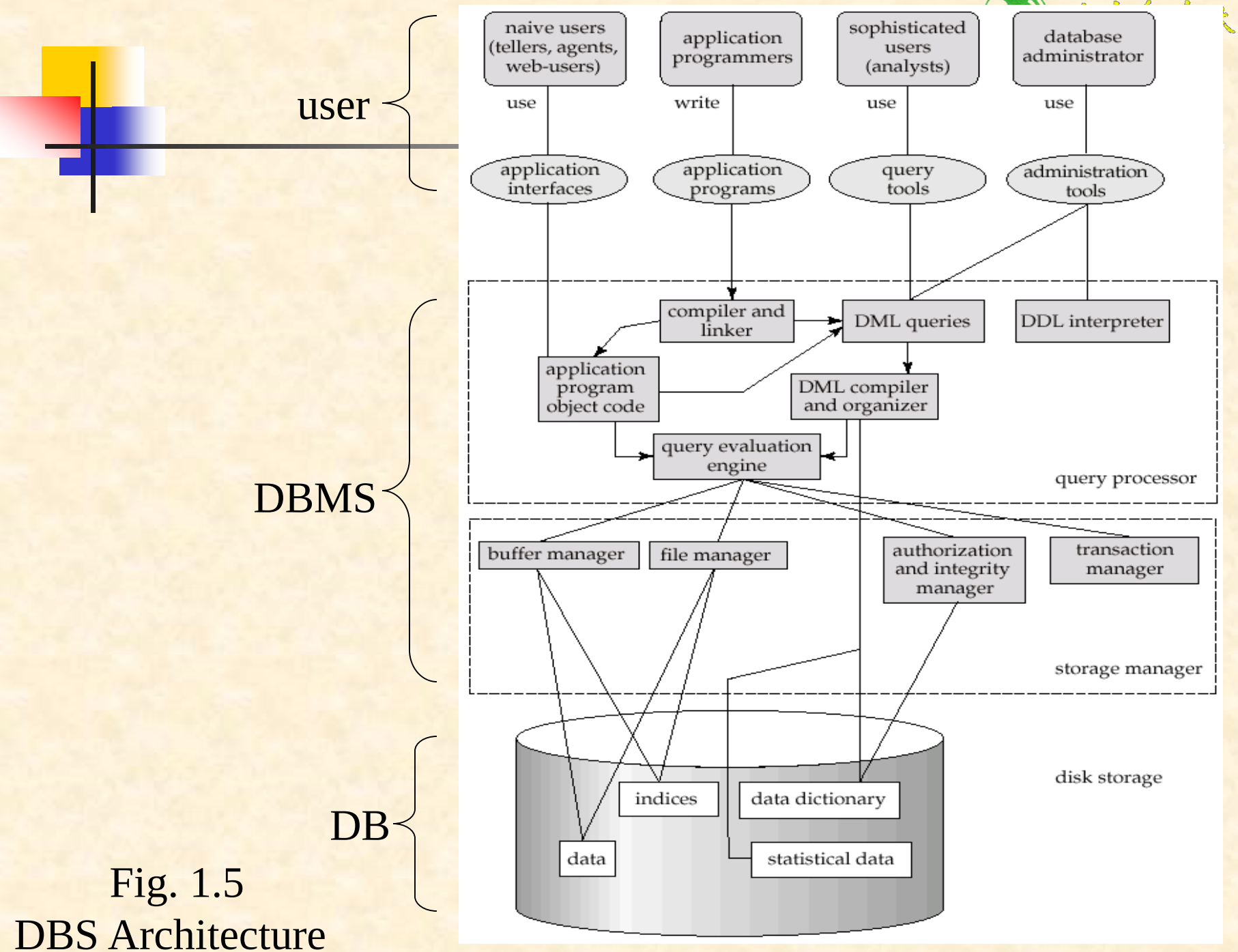


Fig. 1.5
DBS Architecture

Database Application System

- DBS
  - DB + DBMS + User/DBA
  - 实现数据管理功能
- DBAS (Database Application Systems)
  - DB + DBMS + *Application programs* + Users/DBA
  - 实现**数据处理**功能

- 信息/数据管理（management）
  - 数据的分类、收集、组织、编码、存储、检索和维护

- 信息/数据处理（processing）
  - 对数据进行分类、收集、组织、存储，进而从已有数据出发，抽取或推导出新的数据/信息

Database Application System (cont.)

- 信息/数据处理与信息/数据管理的区别
  - 数据管理侧重于对数据如何进行组织和存储，并根据用户需要提供对数据的访问支持
  - 数据处理除了具有数据管理功能外，还可以对通过数据管理得到的数据进行进一步深加工，从中获取新的数据、信息
- 例如，在一个商品零售系统中，可以利用数据库系统的数据管理功能存储各种商品基本信息（如商品数量、销售额等），并为各类用户提供对各类商品信息的查询功能。当从数据库系统中得到这些商品信息后，可以采用一些统计分析工具、数据挖掘（Data Mining）工具进一步进行分析处理，得到有关哪些商品属于热销商品、热销商品销售额等新信息。这种统计分析属于数据处理范畴

Database (Application System) Architecture

- Fig.1.6
- In DBAS, the application programs are responsible for information or data processing, e.g. data mining

- 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”

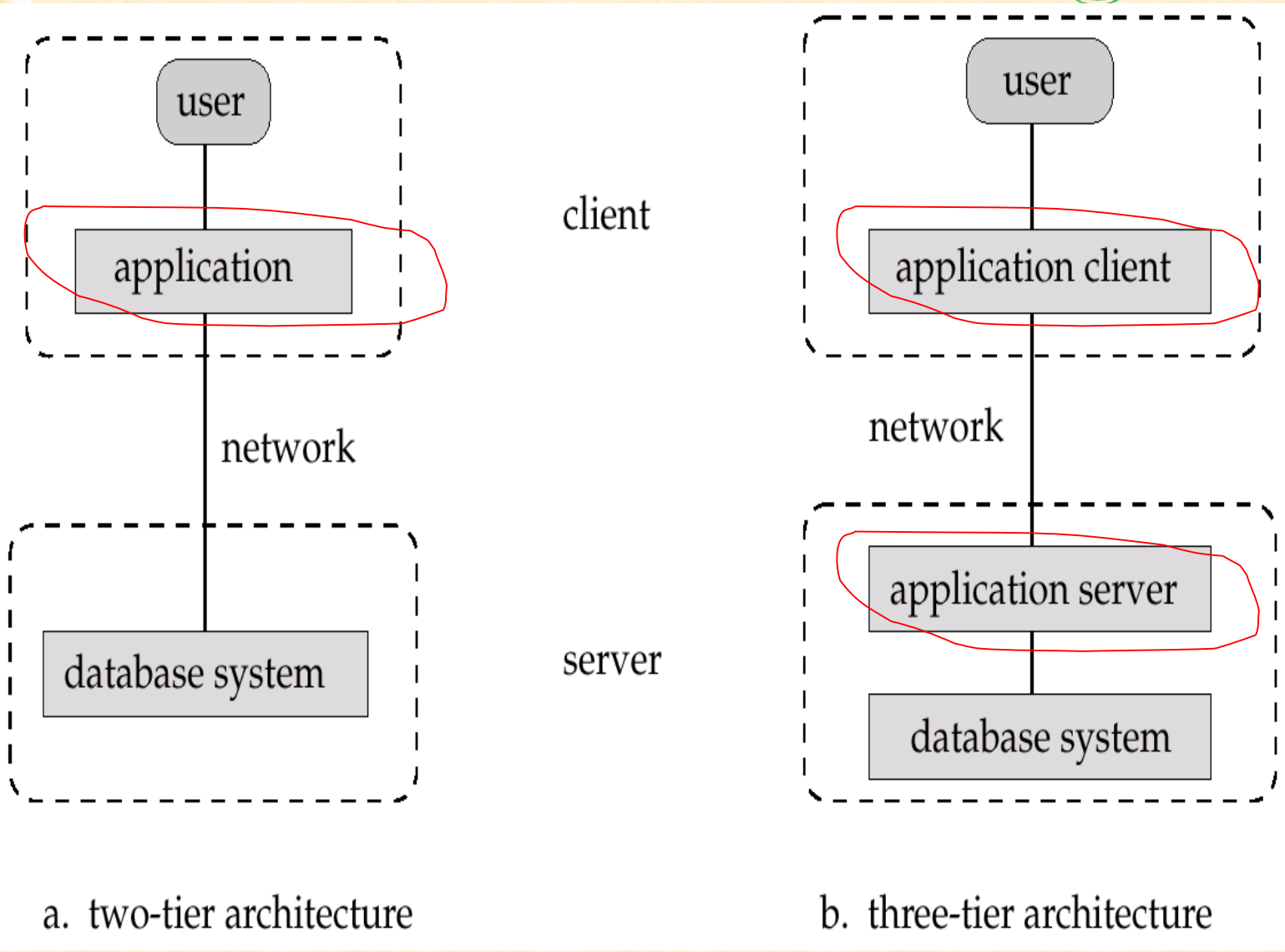


Fig.1.6 Two-tier and three-tier architectures

§ 1.7 Data Storage and Querying

- In DBS, DBMS provides data management mechanisms for users, that is, data storage and data querying
- DBMS consists of two modules
  - storage manager
  - query processor
- **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 for
  - translating DML statements into low-level file commands
  - efficient storing, retrieving and updating of data

1.7 Data Storage and Querying (cont.)

- **The modules in the storage manager** are
  - authorization and integrity manager, transaction manager, file manager, and buffer manager

- The **query processor includes**
  - DDL interpreter（DDL解释器）
  - DML compiler（DML编译器）
  - Query optimization（查询优化）
  - Query evaluation engine（查询执行引擎）


C, Pascal programs
词法/语法/语义分析
中间代码生成
(中间)代码优化
目标代码生成
目标程序代码
程序编译/编译器
process / thread
进程管理
并发控制&进程调度
死锁处理
程序执行/OS
SQL query
扫描和语法/语义分析
关系代数表达式&查询树
查询优化
(优化后)查询执行计划
查询代码生成
查询计划执行的代码
查询处理/DBMS
Chapter 15, 16
(§ 16)
(§ 15)
transaction
事务管理
并发控制&事务调度
死锁处理
恢复技术
(§ 17)
(§ 18)
(§ 19)
事务、存储处理/DBMS
Chapter 17,18, 19

§ 1.8 Transaction Management

- A ***transaction*** is a collection of *operations* that performs a single logical function in a database application
- E.g. Transaction T1: transfer $50 from *account_A* to *account_B*
  - T1 : read (*account_A*);
    account_A := account_A -50;
    write (*account_A*);
    read (*account_ B*);
    account_ B := *account_ B* + 50;
    write (*account_ B*);
  - integrity constraint
    total amount of *account_A* and *account_B* remains unchanged before and after the transaction execution

Transaction Management

- **Transaction properties**
  - **atomicity (原子性), consistency (一致性), isolation (独立性), durability(持久性)**
- ***Transaction-management*** component in DBMS
  - ensures that the database remains in a consistent/correct state despite system failures (e.g., power failures and operating system crashes) and transaction failures, by means of concurrency control and failure recovery.
  - controls the interaction among the concurrent transactions, to ensure the consistency of the database

1.11 Object-Relational Data Models/Databases

- Relational model: “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 (保持)relational foundations, in particular the declarative(声明) access to data, while extending modeling power.
  - Provide upward(向上的) compatibility with existing relational languages.

XML: Extensible Markup Language

- Defined by the WWW Consortium (W3C)
- Originally intended as a document markup language not a database language
- The ability to specify new tags(标记), and to create nested tag structures made XML a great way to exchange **data**, not just documents
- XML has become the basis for all new generation data interchange formats.
- A wide variety of tools is available for parsing, browsing and querying XML documents/data

§ 1.12 Database User and Administrators

1.12.1 Database users and user interfaces

- Users are differentiated by the way they expect to interact with the system, four categories of users, four categories of interfaces: Fig.1.5
  - naïve（无经验的） users – invoke（借助） one of the permanent application programs that have been written previously
    - e.g. people accessing database over the web, bank tellers, clerical staff（职员）
  - application programmers – interact with system through DML calls
  - sophisticated（老练的） users – form requests in a database query language or tools, and submit these queries to the query processor

§ 1.12 Database User and Administrators (cont.)

- specialized users – write specialized database applications that do not fit into the traditional data processing framework

1.12.2 Database Administrator (DBA)

- The responsibilities of DBA include, but *not limited to*
  - schema definition
  - schema and physical organization modification
  - granting of authorization for data access
  - routine maintenance（日常维护）

1.13 History of Database Systems

- 1950s and early 1960s:
  - Data processing using magnetic tapes（磁带）for storage
    - Tapes provided only sequential access
  - Punched cards（穿孔卡片）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.)

- **1969, IBM's IMS (Information Management system)**

  hierarchical data model，层次数据库系统

- **1971, 美国数据系统语言协会CODASYL(Conference on Data System Language) report network data model，网状模型**

- **1970, E.F.Codd defined the relational model，大型共享数据库数据的关系模型**

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.)

- Early 2000s:
  - XML and XQuery standards
  - Automated database administration
- Later 2000s: Big Data
  - Giant（巨型）data storage systems
    - Google BigTable, Yahoo PNuts, Amazon, ..

Conclusion

- Overview of this course
- The concepts of DB/DBMS/DBS/DBAS
- Three-level of database abstraction
- Data independence
- Data model
- … …