

第1章：绪论

邹兆年

哈尔滨工业大学
计算机科学与技术学院
海量数据计算研究中心
电子邮件: znzou@hit.edu.cn

2022年春

教学内容¹

- 1 Data Management
- 2 Database Systems
- 3 Data Models
- 4 Data Schemas
- 5 Database Languages
- 6 Transaction Processing
- 7 System Catalogs
- 8 Architecture of a DBMS

¹课件更新于2022年2月21日

Data Management

数据(Data)

数据(data): 能够被记录且具有实际意义的已知事实

Example (数据)

- “Everest”: 世界最高峰的英文名
- 8,848: 世界最高峰的高度(单位: 米)
- 29,029: 世界最高峰的高度(单位: 英尺)
- “Asia”: 世界最高峰所在的大洲



- : 世界最高峰的照片

数据管理(Data Management)

数据管理(data management): 在计算机中对数据进行存储、检索、更新及共享

- 几乎所有应用程序都要进行数据管理

Example (数据管理)

- 学生入学时录入学籍信息
- 学生查找课程信息及选课
- 教师录入学生成绩
- 教学秘书统计学生成绩排名
- ...

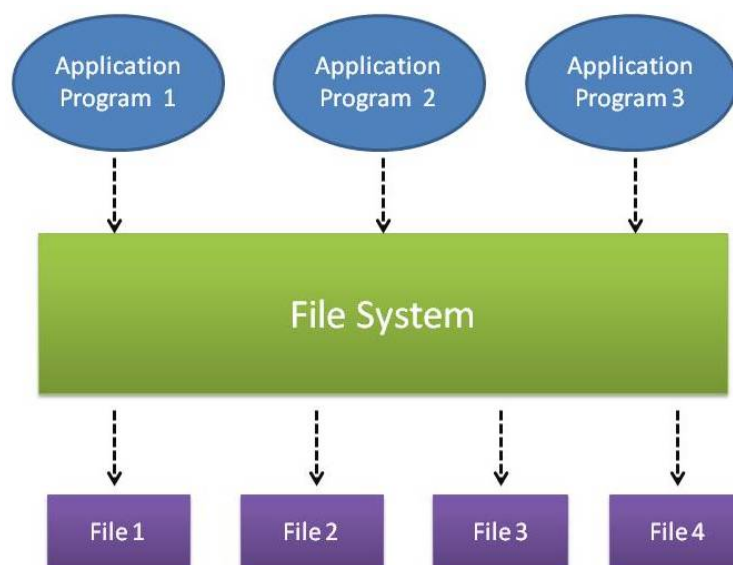
调查

- ① 你管理过数据吗?
- ② 你用过什么方法管理数据?
- ③ 你认为什么是好的数据管理方法?

基于文件系统的数据管理方法

特点:

- ① 数据存储于文件中
- ② 数据由应用程序经过文件系统进行管理



基于文件系统的数据管理方法

实现方法:

- 编写应用程序(文本文件、二进制文件)
- 使用shell (文本文件)

Example (基于文件系统的数据管理)

student.txt

CS-001	Elsa	F	19	CS	Turing
CS-002	Ed	M	19	CS	Turing
MA-001	Abby	F	18	Math	Gauss
MA-002	Cindy	F	19	Math	Gauss
PH-001	Nick	M	20	Physics	Newton

grade.txt

CS-001	1002	95
CS-001	3006	90
CS-002	3006	80
MA-001	1002	
PH-001	1002	92
PH-001	2003	85
PH-001	3006	88

这些查询怎么做?

- 找出计算机系(CS)的所有学生
- 找出选修了1002号课程的学生

基于文件系统的数据管理方法的缺点

- ① 每当文件格式发生变化, 就要修改应用程序

Example (基于文件系统的数据管理方法的缺点)

student.txt

CS-001	Elsa	F	CS	Turing
CS-002	Ed	M	CS	Turing
MA-001	Abby	F	Math	Gauss
MA-002	Cindy	F	Math	Gauss
PH-001	Nick	M	Physics	Newton

grade.txt

CS-001	1002	95
CS-001	3006	90
CS-002	3006	80
MA-001	1002	
PH-001	1002	92
PH-001	2003	85
PH-001	3006	88

基于文件系统的数据管理方法的缺点

- ① 每当文件格式发生变化，就要修改应用程序
- ② 文件中存在冗余数据

Example (基于文件系统的数据管理方法的缺点)

student.txt

CS-001	Elsa	F	19	CS	Turing
CS-002	Ed	M	19	CS	Turing
MA-001	Abby	F	18	Math	Gauss
MA-002	Cindy	F	19	Math	Gauss
PH-001	Nick	M	20	Physics	Newton

grade.txt

CS-001	1002	95
CS-001	3006	90
CS-002	3006	80
MA-001	1002	
PH-001	1002	92
PH-001	2003	85
PH-001	3006	88

基于文件系统的数据管理方法的缺点

- ① 每当文件格式发生变化，就要修改应用程序
- ② 文件中存在冗余数据
- ③ 文件修改可能造成数据不一致

Example (基于文件系统的数据管理方法的缺点)

student.txt

CS-001	Elsa	F	19	CS	Valiant
CS-002	Ed	M	19	CS	Turing
MA-001	Abby	F	18	Math	Gauss
MA-002	Cindy	F	19	Math	Gauss
PH-001	Nick	M	20	Physics	Newton

grade.txt

CS-001	1002	95
CS-001	3006	90
CS-002	3006	80
MA-001	1002	
PH-001	1002	92
PH-001	2003	85
PH-001	3006	88

基于文件系统的数据管理方法的缺点

- ① 每当文件格式发生变化，就要修改应用程序
- ② 文件中存在冗余数据
- ③ 文件修改可能造成数据不一致
- ④ 文件修改可能破坏数据正确性

Example (基于文件系统的数据管理方法的缺点)

student.txt

CS-001	Elsa	F	19	CS	Turing
CS-002	Ed	M	19	CS	Turing
MA-001	Abby	F	18	Math	Gauss
MA-002	Cindy	F	19	Math	Gauss
PH-001	Nick	M	20	Physics	Newton

grade.txt

CS-001	1002	95
CS-001	3006	90
CS-002	3006	80
CS-003	1002	
PH-001	1002	92
PH-001	2003	85
PH-001	3006	88

基于文件系统的数据管理方法的缺点

- ① 每当文件格式发生变化，就要修改应用程序
- ② 文件中存在冗余数据
- ③ 文件修改可能造成数据不一致
- ④ 文件修改可能破坏数据正确性
- ⑤ 没有索引，只能扫描文件，数据访问效率低

Example (基于文件系统的数据管理方法的缺点)

student.txt

CS-001	Elsa	F	19	CS	Turing
CS-002	Ed	M	19	CS	Turing
MA-001	Abby	F	18	Math	Gauss
MA-002	Cindy	F	19	Math	Gauss
PH-001	Nick	M	20	Physics	Newton

grade.txt

CS-001	1002	95
CS-001	3006	90
CS-002	3006	80
MA-001	1002	
PH-001	1002	92
PH-001	2003	85
PH-001	3006	88

基于文件系统的数据管理方法的缺点

- ① 每当文件格式发生变化，就要修改应用程序
- ② 文件中存在冗余数据
- ③ 文件修改可能造成数据不一致
- ④ 文件修改可能破坏数据正确性
- ⑤ 没有索引，只能扫描文件，数据访问效率低
- ⑥ 只能对整个文件进行访问控制，数据安全性差

Example (基于文件系统的数据管理方法的缺点)

student.txt

CS-001	Elsa	F	19	CS	Turing
CS-002	Ed	M	19	CS	Turing
MA-001	Abby	F	18	Math	Gauss
MA-002	Cindy	F	19	Math	Gauss
PH-001	Nick	M	20	Physics	Newton

grade.txt

CS-001	1002	95
CS-001	3006	90
CS-002	3006	80
MA-001	1002	
PH-001	1002	92
PH-001	2003	85
PH-001	3006	88

基于文件系统的数据管理方法的缺点

- ⑦ 没有并发控制，多个应用程序同时读写文件可能产生冲突

Example (基于文件系统的数据管理方法的缺点)

student.txt

CS-001	Elsa	F	19	CS	Turing
CS-002	Ed	M	19	CS	Turing
MA-001	Abby	F	18	Math	Gauss
MA-002	Cindy	F	19	Math	Gauss
PH-001	Nick	M	20	Physics	Newton

grade.txt

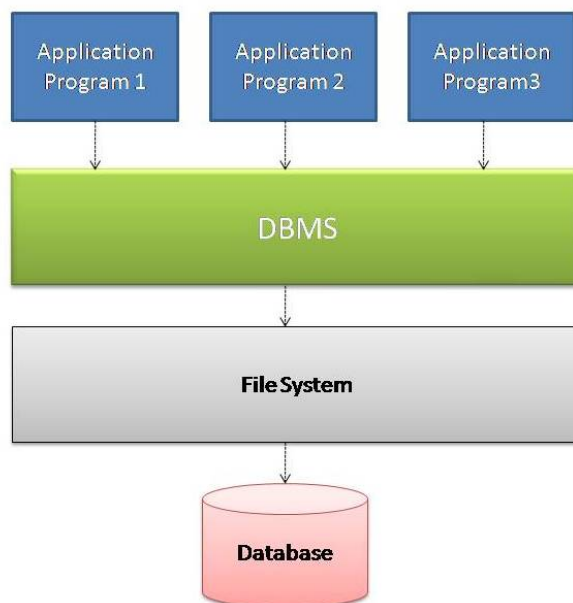
CS-001	1002	95
CS-001	3006	90
CS-002	3006	80
MA-001	1002	
PH-001	1002	92
PH-001	2003	85
PH-001	3006	88

步骤	教师A	教师B
1	将分数读入变量X ($X = 95$)	
2		将分数读入变量Y ($Y = 95$)
3	将分数更新为 $X + 1 = 96$	
4	关闭文件	
5		将分数更新为 $Y + 1 = 96$
6		关闭文件

基于数据库管理系统的数据管理方法

特点:

- 数据存储于 **数据库(database)** 中
- 数据由应用程序经过 **数据库管理系统(database management systems, DBMS)** 进行管理



基于FS的方法 vs. 基于DBMS的方法

- ① FS: 每当文件格式发生变化, 就要修改应用程序
DBMS: 只要数据库模式不发生重大变化, 应用程序基本无需修改(第1章)
- ② FS: 文件中存在冗余数据
DBMS: 如果数据库模式设计得规范, 则数据库中的数据冗余较少(第5章)
- ③ FS: 文件修改可能造成数据不一致
DBMS: 如果数据库模式设计得规范, 则数据库修改基本不会造成数据不一致(第5章)

基于FS的方法 vs. 基于DBMS的方法

- ④ FS: 文件修改可能破坏数据正确性
DBMS: DBMS会对数据更新进行完整性检查, 防止数据更新破坏数据库的正确性(第2章)
- ⑤ FS: 没有索引, 只能扫描文件, 数据访问效率低
DBMS: DBMS提供索引结构, 可以提高数据访问效率(第6章)
- ⑥ FS: 只能对整个文件进行访问控制, 数据安全性差
DBMS: DBMS可以规定一个用户可以对数据库的哪个部分进行哪些操作(第3章)
- ⑦ FS: 没有并发控制, 多个应用程序同时读写文件可能产生冲突
DBMS: DBMS提供事务并发控制机制(第11章)

基于FS的方法 vs. 基于DBMS的方法

	基于FS的方法	基于DBMS的方法
数据冗余度	高	低
数据一致性	No	Yes
数据正确性	No	Yes
索引	No	Yes
访问控制	No	Yes
并发控制	No	Yes
故障恢复	No	Yes

数据管理的功能

- 数据定义(data definition): 定义数据的结构、类型及约束
- 数据存储(data storage): 存储和存取数据
- 数据操纵(data manipulation): 查询数据、更新数据(插入数据、修改数据、删除数据)
- 数据共享(data sharing): 事务管理(transaction management)、并发控制(concurrency control)、故障恢复(failure recovery)
- 数据控制(data control): 保证数据完整性(data integrity)、数据安全性(data security)
- 数据维护(data maintenance): 数据录入、数据转换、数据备份、数据恢复、性能监控

数据定义(Data Definition)

定义数据的结构、类型及约束

Example (数据定义)

- 定义Student表

```
CREATE TABLE Student (                                -- 定义表名
    Sno CHAR(6) PRIMARY KEY,                            -- 定义Sno属性
    Sname VARCHAR(10) NOT NULL,                        -- 定义Sname属性
    Ssex CHAR CHECK (Ssex IN ('M', 'F')),              -- 定义Ssex属性
    Sage INT CHECK (Sage > 0),                          -- 定义Sage属性
    Sdept VARCHAR(20)                                   -- 定义Sdept属性
);
```

- 查看Student表的模式 [▶ 演示](#)

- ▶ PostgreSQL和openGauss: \d Student;
- ▶ MySQL: describe Student;

数据存储(Data Storage)

存储和存取数据

Example (数据存储)

- 查看Student表存储于哪个文件 ▶ 演示

```
SELECT pg_relation_filepath('Student');
```

- 查看Student表中的数据 ▶ 演示

```
SELECT * FROM Student;
```

Example (索引)

- 查看College数据库中的索引 ▶ 演示

▶ PostgreSQL和openGauss: \di

▶ MySQL: SHOW INDEX

- 创建索引 ▶ 演示

```
CREATE INDEX student_idx_sname ON Student(Sname);
```

数据操纵(Data Manipulation)

查询数据、更新数据(插入数据、修改数据、删除数据)

Example (查询数据)

- 查找计算机(CS)系的全体学生 ▶ 演示

```
SELECT * FROM Student WHERE Sdept = 'CS';
```

Example (更新数据)

- 插入学生元组 ▶ 演示

```
INSERT INTO Student VALUES  
( 'CS-003', 'Jill', 'F', 19, 'CS' );
```

- 修改学生元组 ▶ 演示

```
UPDATE Student SET Sage = Sage + 1  
WHERE Sno = 'CS=003';
```

- 删除学生元组 ▶ 演示

```
DELETE FROM Student WHERE Sno = 'CS-003';
```

数据共享(Data Sharing) I

事务管理(transaction management)、并发控制(concurrency control)、故障恢复(failure recovery)

Example (事务)

- 事务：给CS-001号学生的1002号课成绩加1分 ▶ 演示

BEGIN; -- 事务启动

```
SELECT Grade FROM SC
```

WHERE Sno = 'CS-001' AND Cno = '1002'; -- 查询分数

```
UPDATE SC Set Grade = Grade + 1
```

WHERE Sno = 'CS-001' AND Cno = '1002'; -- 修改分数

END; -- 事务结束

- 事务具有原子性(atomic): 事务的所有命令要么全部执行, 要么全部不执行
- 在事务执行过程中终止事务会发生什么现象? ▶ 演示

数据共享(Data Sharing) II

Example (并发控制)

- 2个事务分别给CS-001号学生的1002号课成绩加1分 ▶ 演示

会话1: BEGIN: -- 事务1启动

会话2: BEGIN: -- 事务2启动

会话1: SELECT Grade FROM SC

WHERE Sno = 'CS-001' AND Cno = '1002'; -- 事务1查分数

会话2: SELECT Grade FROM SC

WHERE Sno = 'CS-001' AND Cno = '1002'; -- 事务2查分数

会话1: UPDATE SC Set Grade = Grade + 1

WHERE Sno = 'CS-001' AND Cno = '1002'; -- 事务1改分数

会话2: UPDATE SC Set Grade = Grade + 1

WHERE Sno = 'CS-001' AND Cno = '1002'; -- 事务2等待

会话1: END; -- 事务1结束, 事务2完成分数修改

会话2: END; -- 事务2结束

Database Systems

数据库(Database)

数据库(database, 简称DB): 有组织的、共享的、持久存储的数据集合

A database (DB) is a set of related data that is **organized**, **shared**, and **persistent**

Example (数据库)

- College是一个(关系)数据库
- 列出数据库 [▶ 演示](#)
 - ▶ PostgreSQL和openGauss: \1
 - ▶ MySQL: SHOW DATABASES;

数据库管理系统(DBMS)

数据库管理系统(database management system, 简称DBMS): 一种通用系统软件, 它简化了在不同用户及应用程序之间组织、存储、操纵、控制及维护数据库

A database management system (DBMS) is a general-purpose system software that facilitates the **organization**, **storage**, **manipulation**, **control**, and **maintenance** of databases among various users and application programs.


Example (数据库管理系统)

Oracle、SQL Server、PostgreSQL、MySQL、openGauss、SQLite都是(关系)数据库管理系统

数据库用户(Database User)

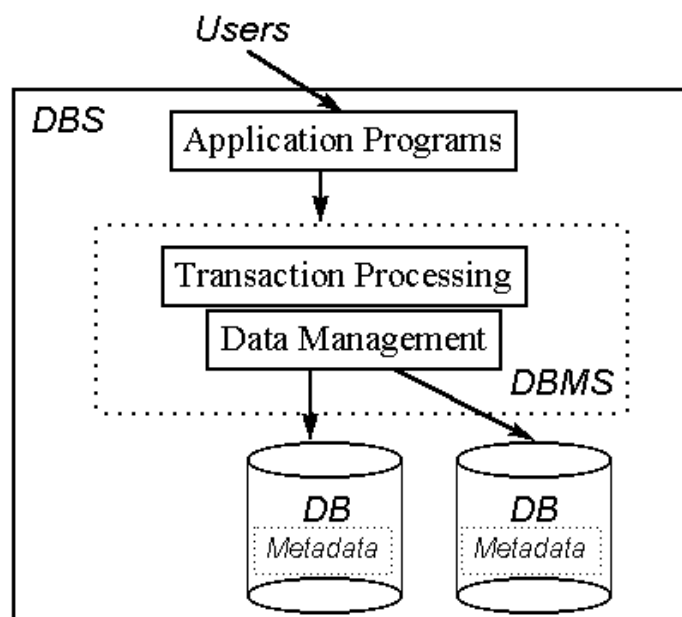
- 数据库管理员(database administrator, DBA): 授权数据库访问, 协调和监控数据库使用, 获取软硬件资源, 控制资源的使用, 监控数据库性能
- 数据库设计者(database designer): 负责定义数据库的内容、结构、约束、存储过程和事务
- 终端用户(end user): 查询数据库, 生成报表, 部分终端用户还可以修改数据库内容

Example (数据库用户)

- zhaonian是College数据库的用户
- 列出用户 
 - ▶ PostgreSQL和openGauss: \du
 - ▶ MySQL: SELECT user FROM mysql.user;

数据库系统(Database System)

数据库系统(database system, 简称DBS): 由数据库、数据库管理系统、应用程序和数据库用户在一起构成的系统



学习数据库系统的重要性

- ① 当你得到一个数据库，并需要对它进行管理时，你需要了解数据库系统的基本概念，掌握数据库语言，具备数据库系统的使用技能(第1-3章)
- ② 当你面对一个数据密集型应用设计与开发需求时，你需要掌握数据库的设计方法，了解如何评估设计方案的优劣，具备数据库系统应用开发能力(第4-6章)
- ③ 当你接手一个性能低下的数据密集型应用时，你需要了解数据库系统的工作原理，知道如何对系统进行优化和重新设计(第6-12章)
- ④ 当你参与一种新型数据库管理系统的研发时，你需要了解多种数据库管理系统的工作原理和设计方案，并具备一定的研究能力(第6-12章)

Data Models

数据抽象(Data Abstraction)

数据抽象(data abstraction)是将现实世界映射到计算机世界的过程

现实世界 → 信息世界 → 计算机世界

- 现实世界: 张三、李四、数学系、物理系、高等数学、大学物理...
- 信息世界: 实体、属性、联系、约束...
- 计算机世界: 记录、域、引用...

数据模型(Data Model)

数据模型(data model)是进行数据抽象的工具

数据模型具有三个要素

- ① 用于描述数据库结构的一系列概念
- ② 用于操纵数据结构的一系列操作
- ③ 数据库应当服从的一系列约束条件

Example (关系数据模型的三要素)

- ① 用于描述数据库结构的一系列概念: 关系、元组、属性等 [▶ 演示](#)
- ② 用于操纵数据结构的一系列操作: 选择、投影、连接、集合并、集合差、聚集等关系代数操作 [▶ 演示](#)
- ③ 数据库应当服从的一系列约束条件: 实体完整性约束、参照完整性约束、用户定义完整性约束 [▶ 演示](#)

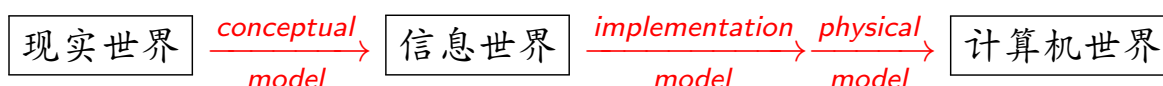
详细介绍见第2章

数据模型的分类

单独一种数据模型难以胜任数据抽象工作

按用途可将数据模型分为三类

- 概念数据模型(conceptual data model)
- 实现数据模型(implementation data model)
- 物理数据模型(physical data model)

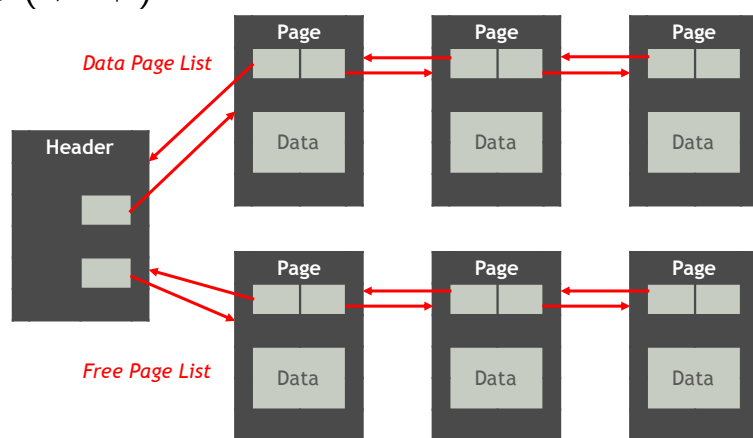


概念数据模型(Conceptual Data Model)

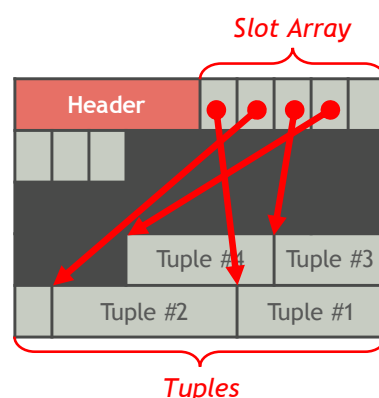
概念数据模型提供的概念最接近用户理解数据的方式，用于将现实世界映射到信息世界(第4章)

物理数据模型(Physical Data Model)

物理数据模型提供的概念用于描述数据库在计算机中的存储细节(第7章)



表文件的组织



文件页面的组织

实现数据模型(Implementation Data Model)

实现数据模型处在概念数据模型和物理数据模型之间，在实现DBMS时使用

- 层次数据模型
- 网络数据模型
- **关系数据模型**(本课程学习的实现数据模型, 第2章)
- 面向对象数据模型
- XML数据模型
- 文档数据模型
- 图数据模型

Data Schemas

数据库模式(Database Schema)

数据库模式(database schema): 对数据库的结构、类型、约束的描述

- 数据库模式是数据库的“类型声明”
- 数据库模式不经常变化

Example (关系模式)

- Student关系的模式

Sno	Sname	Ssex	Sage	Sdept
-----	-------	------	------	-------

- 查看Student关系的模式 [▶ 演示](#)
 - ▶ PostgreSQL和openGauss: \d Student;
 - ▶ MySQL: describe Student;

数据库实例(Database Instance)

数据库实例(database instance): 数据库在某一特定时间存储的数据

- 数据库实例是数据库的“值”
- 每当数据库被更新, 数据库实例就发生变化

Example (关系实例)

- Student关系的实例

Sno	Sname	Ssex	Sage	Sdept
PH-001	Nick	M	20	Physics
CS-001	Elsa	F	19	CS
CS-002	Ed	M	19	CS
MA-001	Abby	F	18	Math
MA-002	Cindy	F	19	Math

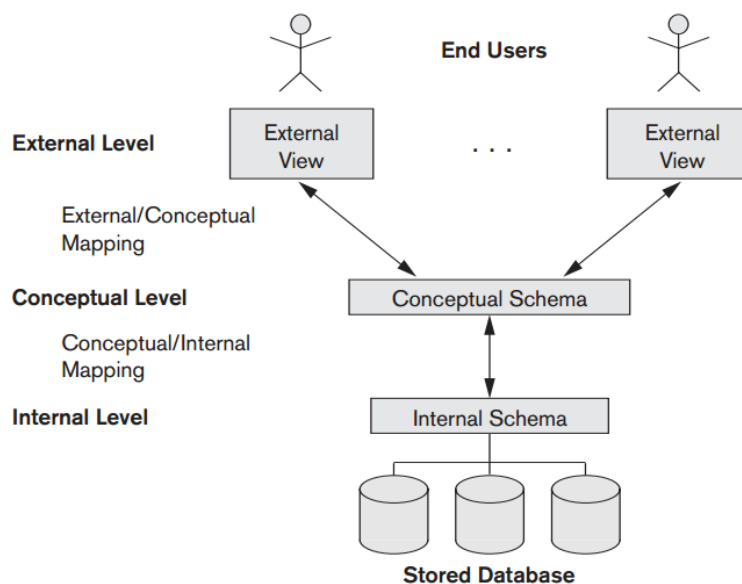
- 查看Student关系的实例 [▶ 演示](#)

```
SELECT * FROM Student;
```

数据库的三层模式结构(Three-Schema Architecture)

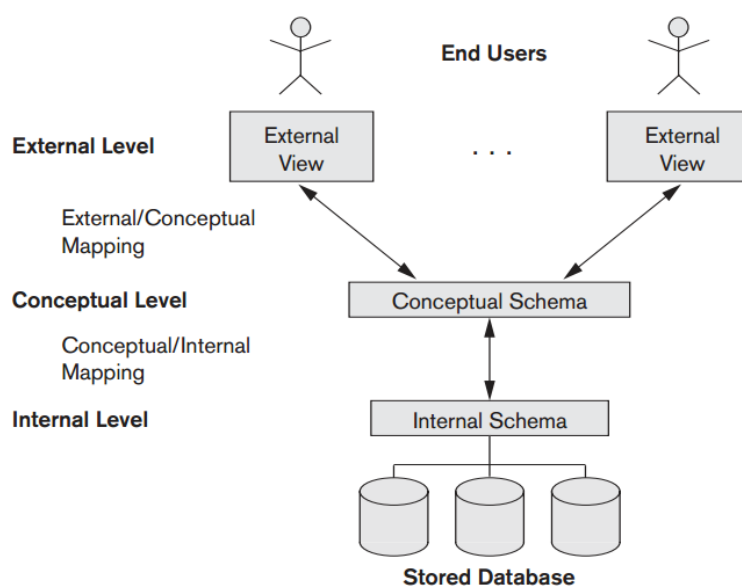
数据库不是仅用一种模式来刻画的，数据库模式通常分三个层次来定义，从低到高分别是

- ① 内模式(internal schema)/存储模式(storage schema)
- ② 概念模式(Conceptual Schema)
- ③ 外模式(external schema)/视图(view)



内模式(Internal Schema)/存储模式(Storage Schema)

- 描述数据库的物理存储结构和存取方法
- 数据库只有一个内模式
- 定义内模式时通常使用物理数据模型提供的概念



概念模式(Conceptual Schema)

- 为全体数据库用户描述整个数据库的结构和约束
- 数据库只有一个概念模式
- 定义概念模式时使用实现数据模型提供的概念

Example (概念模式)

College数据库的概念模式

- 关系Student(Sno, Sname, Ssex, Sage, Sdept)
- 关系Course(Cno)
- 关系SC(Sno, Cno, Grade)

外模式(External Schema)/视图(View)

- 从不同类别用户的视角描述数据库的结构
- 数据库可以有多个外模式
- 定义外模式时也使用实现数据模型提供的概念

Example (外模式/视图)

- 学生选课数视图vw_qty(Sno, Qty)

- 创建视图 ▶ 演示

```
CREATE VIEW vw_qty(Sno, Qty) AS
  SELECT Sno, COUNT(*) FROM SC GROUP BY Sno;
```

- 查看视图定义 ▶ 演示

```
\sv vw_qty
```

- 在视图上做查询 ▶ 演示

```
SELECT * FROM vw_qty;
```

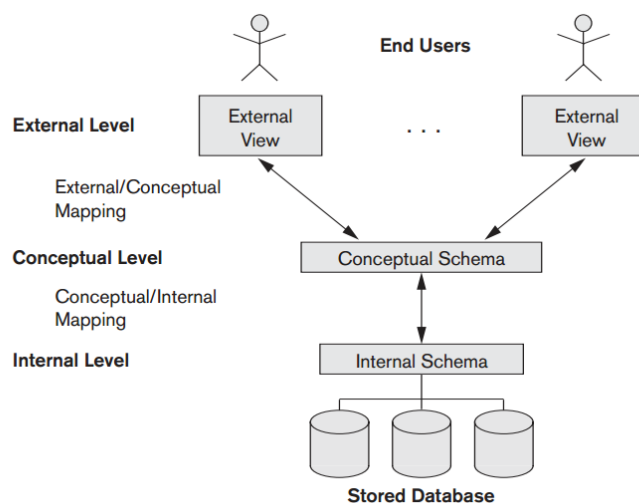
- 列出视图 ▶ 演示

```
\dv
```

模式映射(Schema Mapping)

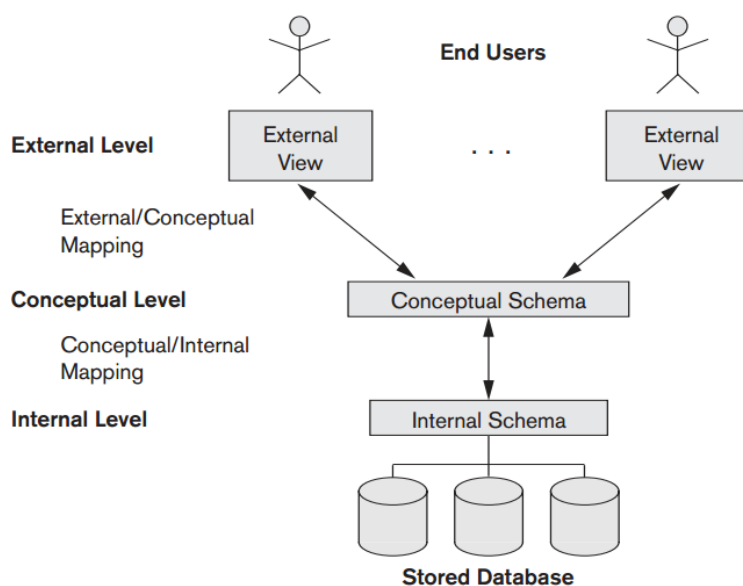
在三层模式结构中，不同层次模式间的映射用于完成应用程序与数据库之间的数据转换(data transformation)和请求转换(request transformation)

- **请求转换**: 应用程序是依据外模式开发的，应用程序在外模式上声明的数据请求通过模式映射转换为DBMS在内模式上的请求
- **数据转换**: 数据库的物理存储是按照内模式来组织的，DBMS检索到的数据通过模式映射转换为符合外模式的组织形式，返回给应用



模式映射的分类

- **外模式-概念模式映射(external/conceptual mapping)**: 从一个外模式到概念模式的映射
- **概念模式-内模式映射(conceptual/internal mapping)**: 从概念模式到内模式的映射



外模式-概念模式映射

Example (外模式-概念模式映射)

- 视图vw_qty到College数据库概念模式的映射

```
CREATE VIEW vw_qty(Sno, Qty) AS  
  SELECT Sno, COUNT(*) FROM SC GROUP BY Sno;
```

- 查看视图vw_qty到College数据库概念模式的映射

```
\sv vw_qty
```

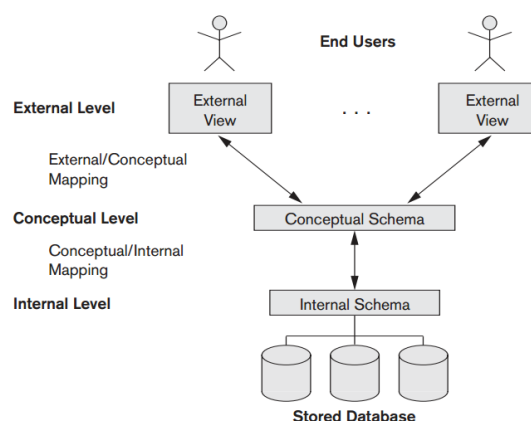
数据独立性(Data Independence)

逻辑数据独立性(logical data independence)

- 当概念模式发生改变时，只需修改外模式到概念模式的映射
- 外模式无需改变，依据外模式开发的应用程序也无需改变

物理数据独立性(physical data independence)

- 当内模式发生改变时，只需修改概念模式到内念模式的映射
- 概念模式和外模式均无需改变，依据外模式开发的应用程序也无需改变



Database Languages

数据库语言(Database Language)

数据库语言(database language): 用户/应用程序与DBMS交互时所使用的语言

- 关系数据库语言: SQL、Datalog
- 图数据库语言: Cypher、Gremlin
- XML数据库语言: XQuery
- ...

数据库语言的分类

- 数据定义语言(data definition languages, DDL): DBA和数据库设计者用来声明数据库模式的语言
- 数据操纵语言(data manipulation languages, DML): 查询和更新数据库时所使用语言

数据库语言的分类

Example (数据定义语言/DDL)

- 定义Student表

```
CREATE TABLE Student (                                -- 定义表名
    Sno CHAR(6) PRIMARY KEY,                            -- 定义Sno属性
    Sname VARCHAR(10) NOT NULL,                        -- 定义Sname属性
    Ssex CHAR CHECK (Ssex IN ('M', 'F')),              -- 定义Ssex属性
    Sage INT CHECK (Sage > 0),                          -- 定义Sage属性
    Sdept VARCHAR(20)                                   -- 定义Sdept属性
);
```

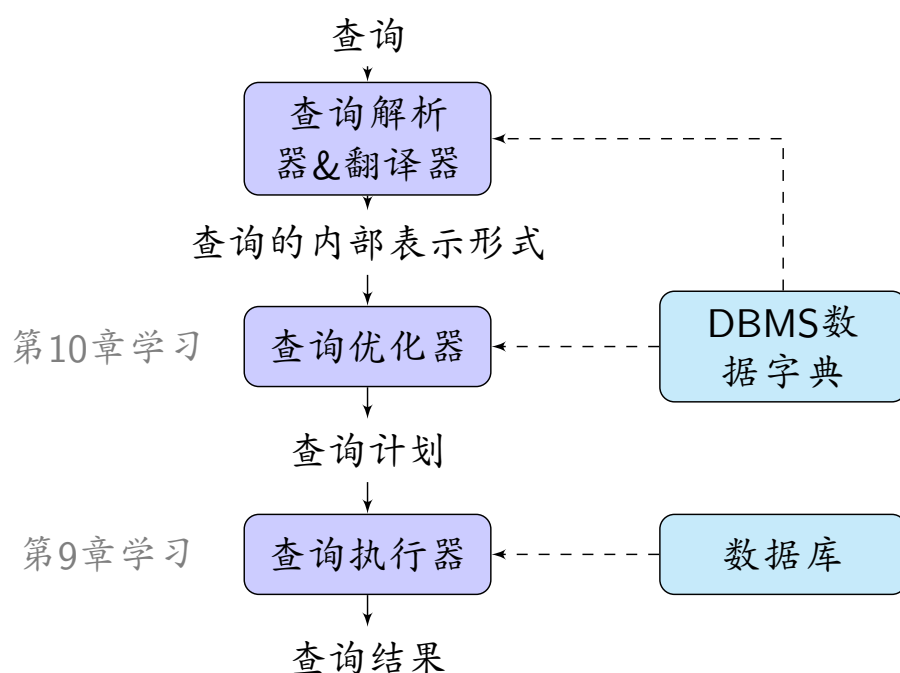
Example (数据操纵语言/DML)

- 查找计算机(CS)系的全体学生

```
SELECT * FROM Student WHERE Sdept = 'CS';
```

数据库查询(Database Queries)

- DML通常是**描述式的(descriptive)**，用它编写的数据库查询只描述查询意图，而不指明查询执行过程
- DBMS自动生成最优查询计划，然后在数据库上执行查询计划



Navigation icons: back, forward, search, etc.

数据库查询(Database Queries)

Example (查询计划)

查找计算机(CS)系的全体学生

- SQL查询语句

```
SELECT * FROM Student WHERE Sdept = 'CS';
```

- 查看查询计划 [▶ 演示](#)

```
EXPLAIN SELECT * FROM Student WHERE Sdept = 'CS';
```

Navigation icons: back, forward, search, etc.

Transaction Processing

事务(Transaction)

事务(transaction)是由数据库上的一系列操作完成的复杂任务，这些操作要么全执行，要么全不执行

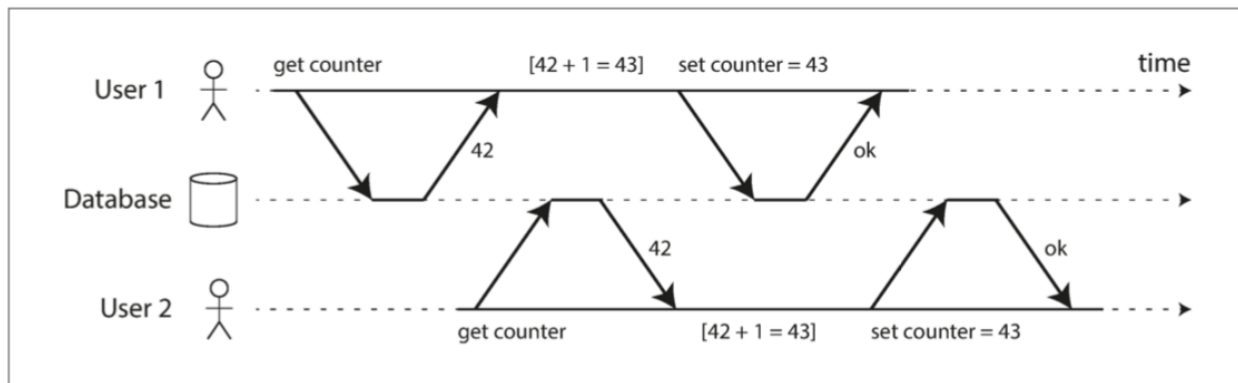
- 银行转帐
- 在线购物
- 会议室预定

事务的性质

- 原子性(atomicity)
- 一致性(consistency)
- 隔离性(isolation)
- 持久性(durability)

并发控制(Concurrency Control)

- 为了充分利用数据库系统，允许多个事务在数据库上并发执行
- 多个事务并发执行可能会破坏数据库的一致性



并发控制(concurrency control)确保多个事务并发执行不会破坏数据库的一致性(第11章学习)

并发控制(Concurrency Control)

Example (并发控制)

- 2个事务分别给CS-001号学生的1002号课成绩加1分

会话1: BEGIN; -- 事务1启动

会话2: BEGIN; -- 事务2启动

会话1: SELECT Grade FROM SC

WHERE Sno = 'CS-001' AND Cno = '1002'; -- 事务1查分数

会话2: SELECT Grade FROM SC

WHERE Sno = 'CS-001' AND Cno = '1002'; -- 事务2查分数

会话1: UPDATE SC Set Grade = Grade + 1

WHERE Sno = 'CS-001' AND Cno = '1002'; -- 事务1改分数

会话2: UPDATE SC Set Grade = Grade + 1

WHERE Sno = 'CS-001' AND Cno = '1002'; -- 事务2等待

会话1: END; -- 事务1结束, 事务2完成分数修改

会话2: END; -- 事务2结束

故障恢复(Failure Recovery)

- 计算机软硬件系统随时可能发生故障
- 故障可能在事务执行过程中间发生，从而破坏数据库的一致性
 - ▶ 例：转账过程中系统发生故障
- 故障恢复(failure recovery)确保系统重启后数据库可以恢复到最近的一致性状态(第10章学习)

故障恢复(Failure Recovery)

Example (故障恢复)

- 事务：给CS-001号学生的1002号课成绩加1分

```
BEGIN; -- 事务启动
```

```
SELECT Grade FROM SC
```

```
WHERE Sno = 'CS-001' AND Cno = '1002'; -- 查询分数
```

```
UPDATE SC Set Grade = Grade + 1
```

```
WHERE Sno = 'CS-001' AND Cno = '1002'; -- 修改分数
```

```
END; -- 事务结束
```

- 在事务执行过程中关闭DBMS会发生什么现象？▶ 演示

System Catalogs

系统目录(System Catalog)

DBMS在其内部的系统目录(system catalog)中存储数据库的元数据(meta-data)

- 表、视图、索引、函数等对象的定义
- 用户、用户访问权限
- 数据库的统计信息，如表的大小、页面数、元组数、元组长度、属性值分布等

基本上所有DBMS都将系统目录存储为数据库

系统目录(System Catalog)

Example (系统目录)

- PostgreSQL中表的元数据存储在pg_tables表中
- 列出PostgreSQL中所有的表 ▶ 演示

```
SELECT tablename FROM pg_tables;
```

- PostgreSQL中视图的元数据存储在pg_views表中

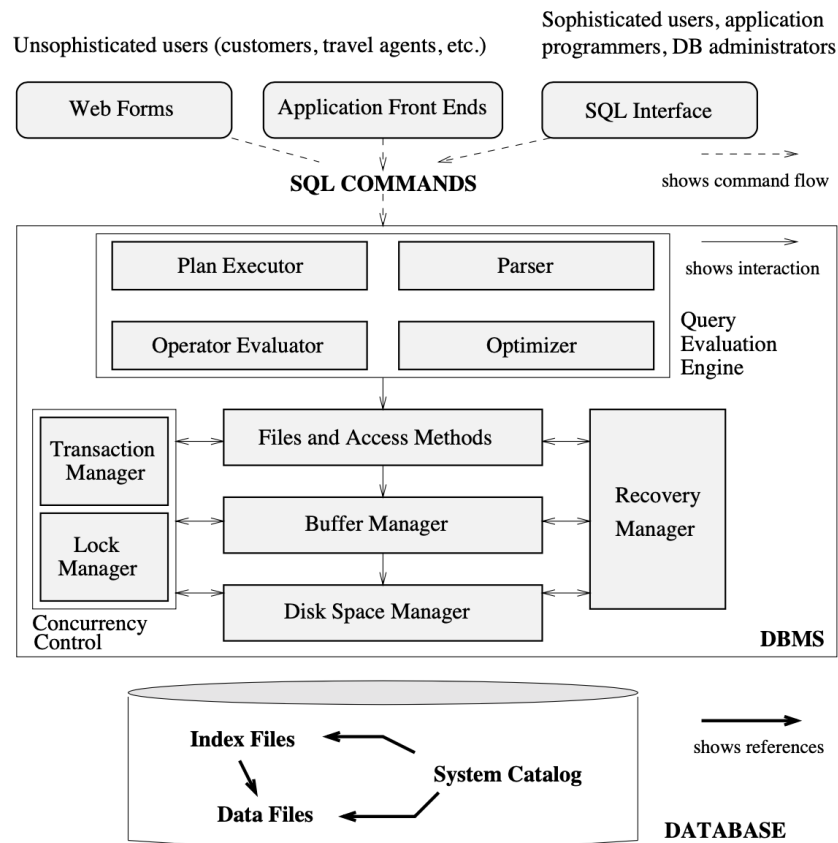
- 查找视图vw_qty的定义 ▶ 演示

```
SELECT definition FROM pg_views  
WHERE viewname = 'vw_qty';
```

等价于 \sv vw_qty

Architecture of a DBMS

DBMS的体系结构



DBMS的体系结构

- SQL语句解析器(编译原理课学习)
- 查询优化器(第10章学习)
- 查询执行器(第9章学习)
- 存取方法(第8章学习)
- 缓冲区管理器(第7章学习)
- 存储管理器(第7章学习)
- 事务管理器(第11章学习)
- 锁管理器(第11章学习)
- 故障恢复管理器(第12章学习)

总结

- ① Data Management
- ② Database Systems
- ③ Data Models
- ④ Data Schemas
- ⑤ Database Languages
- ⑥ Transaction Processing
- ⑦ System Catalogs
- ⑧ Architecture of a DBMS

致谢

- 感谢禹棋赢(1180910123)同学指出课件中的错误。