第一章

什么是数据管理

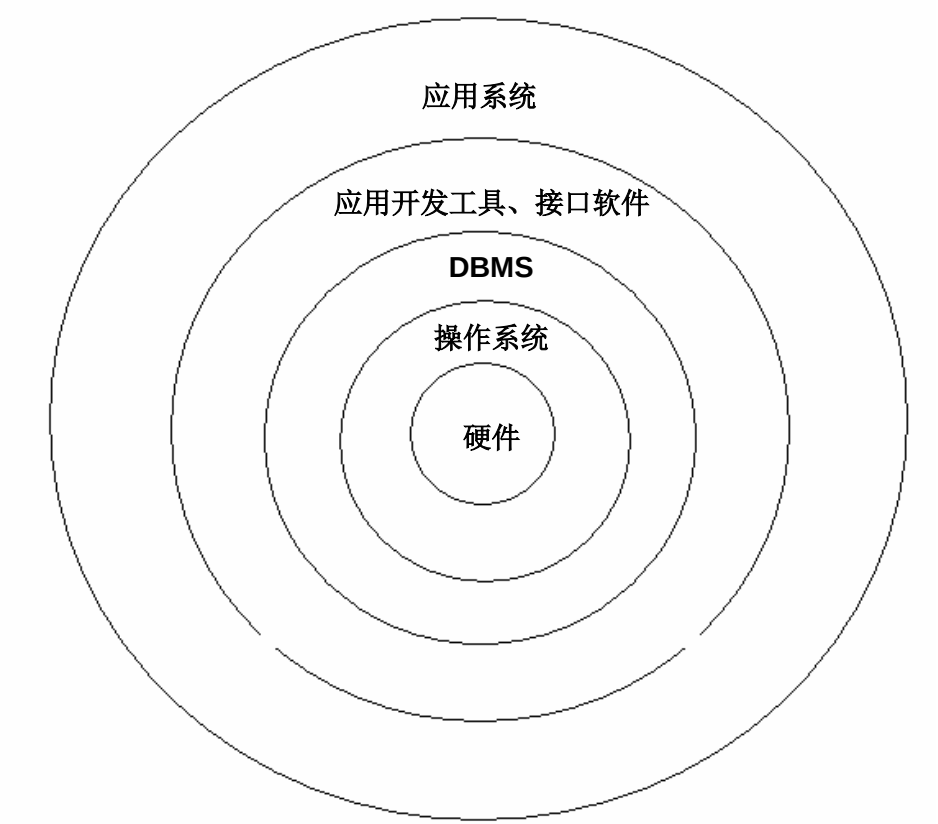
对数据进行分类、组织、编码、存储、检索和维护，是数据处理的中心问题

数据管理技术的发展过程

- 人工管理阶段（40年代中--50年代中）缺点
- 文件系统阶段（50年代末--60年代中）缺点
- 数据库系统阶段（60年代末--现在）优点

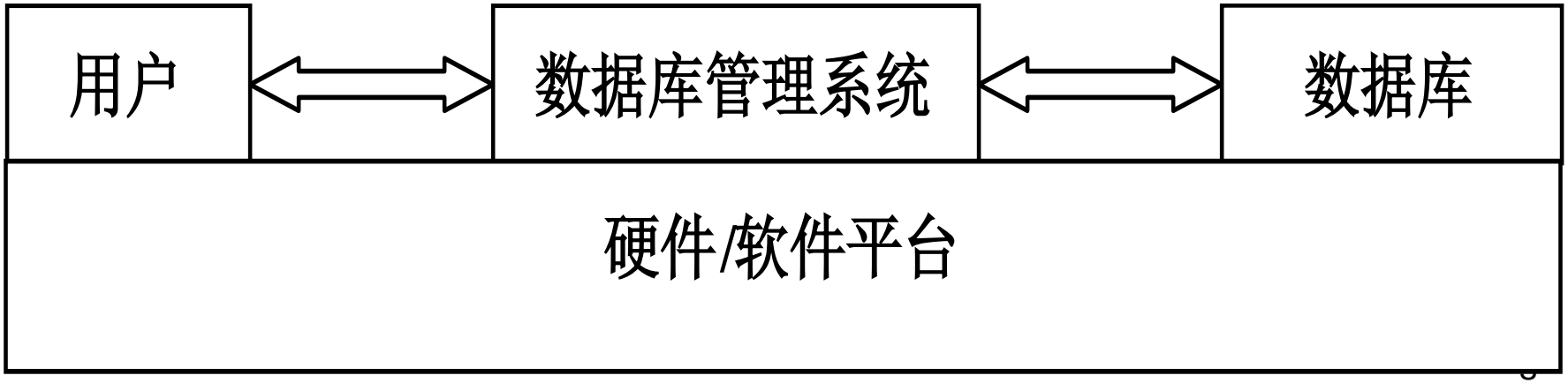
第一章

数据库管理系统（Database Management System，简称DBMS）是位于用户与操作系统之间的一层数据管理软件。



数据库系统的构成

- 数据库
- 数据库管理系统
- 应用系统
- 数据库管理员（DBA）



数据独立性

- **物理独立性**

  指用户的应用程序与存储在磁盘上的数据库中数据是相互独立的。当**数据的物理存储改变了，应用程序不用改变。**

- **逻辑独立性**

  用户的应用程序与数据库的逻辑结构是相互独立的。**数据的逻辑结构改变了，用户程序可以不变。**

数据模型

数据模型分成两个不同的层次

（1）概念模型

也称信息模型，它是按用户的观点来对数据和信息建模。

（2）数据模型

主要包括网状模型、层次模型、关系模型等，它是按计算机系统的观点对数据建模。

DBMS的主要功能:

数据定义功能

提供数据定义语言(DDL: Data Definition Language)

通过DDL可以定义数据库中的数据对象

数据操纵功能:

提供数据操纵语言(DML: Data Manipulation Language)

操纵数据实现对数据库的基本操作（查询、插入、删除和修改）

DBMS的主要功能:

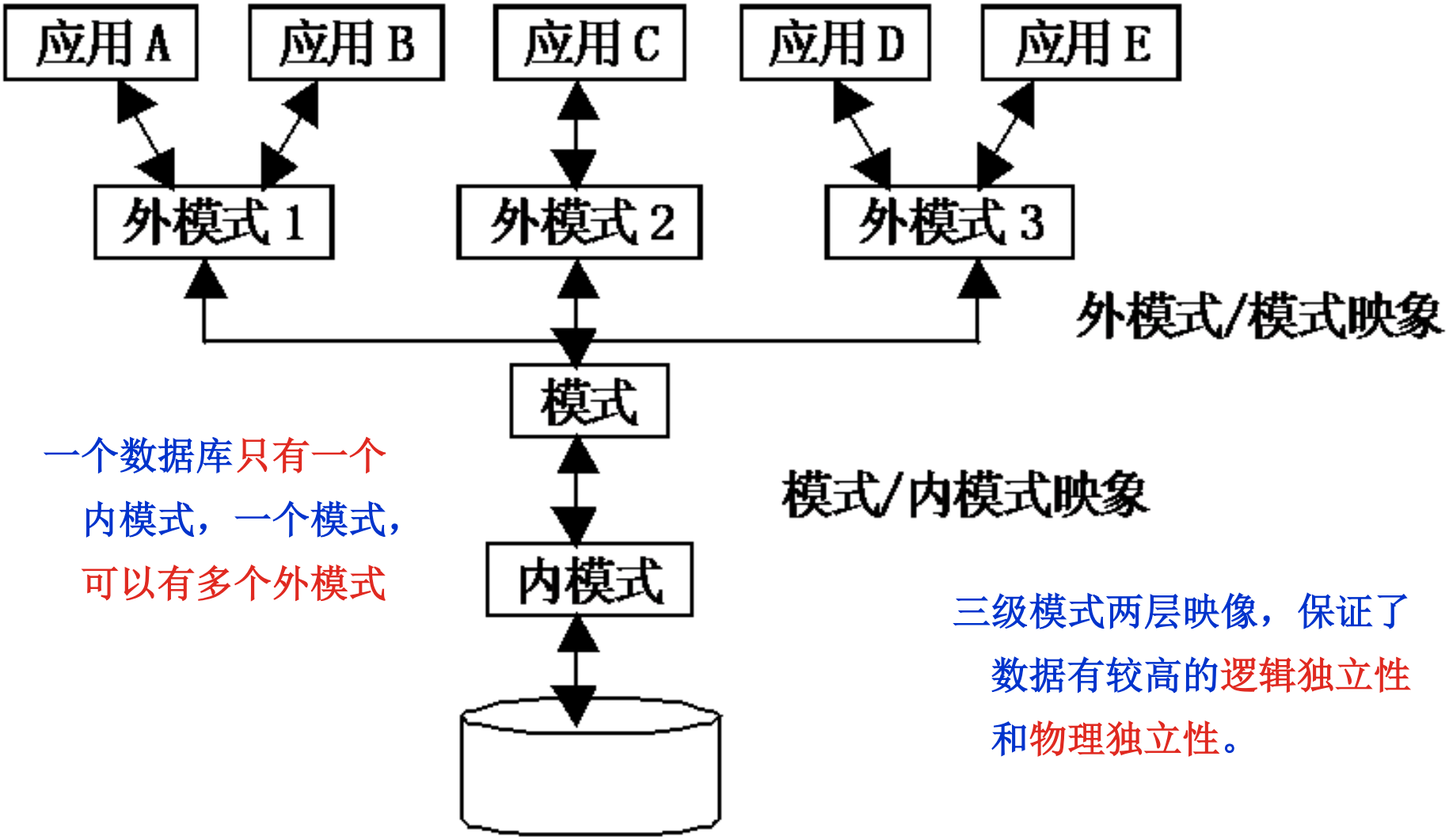
数据库的运行管理

这是DBMS运行时的核心部分，它包括并发控制，安全性检查，完整性约束条件的检查和执行，发生故障后的恢复等。

数据库的建立和维护功能

它包括数据库初始数据的输入及转换，数据库的转储与恢复，数据库的重组功能和性能的监视与分析功能等。

数据库系统的三级模式结构



一个数据库只有一个内模式，一个模式，可以有多个外模式

三级模式两层映像，保证了数据有较高的逻辑独立性和物理独立性。

数据库系统的三级模式结构

数据库系统依靠模式分级、各级之间有映像机制支持了数据独立性

一个数据库只有一个内模式，一个模式，可以有多个外模式。

三级模式两层映像，保证了数据有较高的逻辑独立性和物理独立性。

数据库系统的三级模式结构

- 数据库的三级模式是指外模式（子模式）、模式、内模式（物理模式）。
- 外模式是对各个用户或程序所涉及到的数据的逻辑结构和数据特征的描述。
- （逻辑）模式是对数据库中数据的整体逻辑结构和特征的描述。
- 内模式是数据的内部表示或低层描述。

基本概念

数据库中存储的是 数据以及数据之间的联系 。

DDL：数据定义语言：定义模式、定义表等等

DQL ： 数据查询语言： 查询

DML： 数据操纵语言： 插入、修改、删除

DCL： 数据控制语言： 授权、收回授权等

属性的取值范围称为该属性的域。

一个候选码可由**一个或多个**能唯一标识关系模式中任何元组的属性组成。

数据模型的**三要素**是数据结构、数据操作和完整性约束条件。

关系模型的组成

- 数据结构：二维表格，用“键”导航数据
- 数据操作：集合操作，存取路径隐蔽
- 数据完整性约束：保证数据和语义的正确、有效
  - 实体完整性：　由**primary key** 定义
  - 参照完整性：　由**foreign key** 定义
  - 用户定义完整性:

[not null, unique, check(…)]

第二章 关系数据库

常用的关系操作：

- 查询

  选择、投影、连接、除、并、交、差

- 数据更新

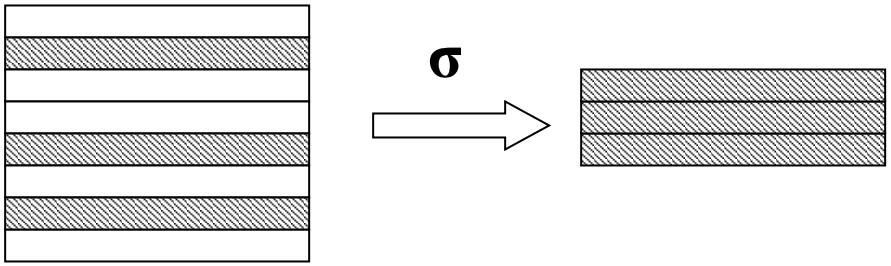
  插入、删除、修改

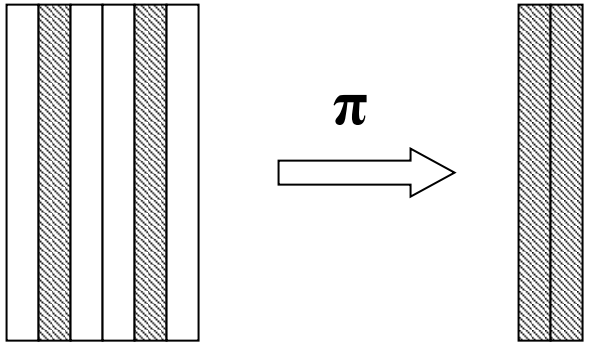
关系操作的特点：

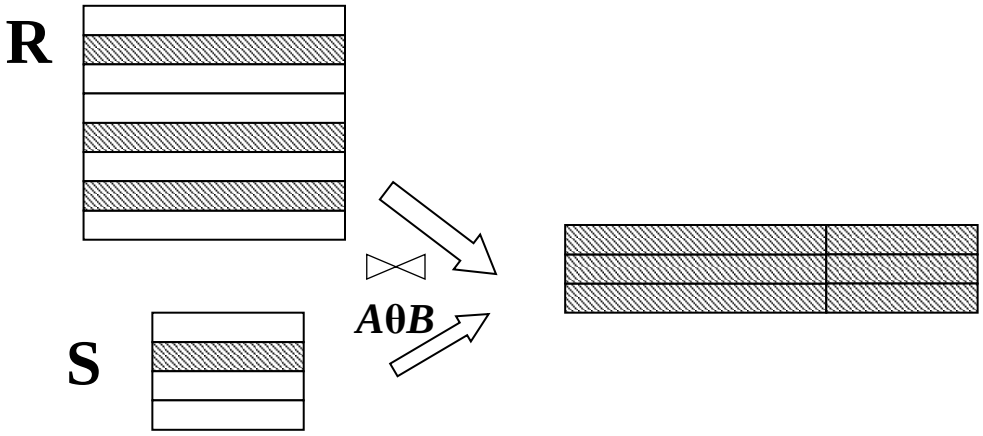
操作的对象和结果都是集合。

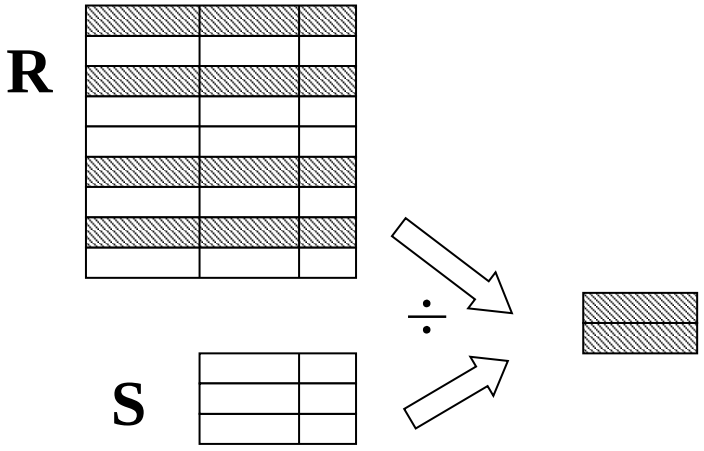
关系代数运算符：

- 选择 σ
- 投影 π
- 连接 ⋈
- 除 ÷









三类常用连接运算

等值连接

$$R \underset{A=B}{\bowtie} S = \{ \widehat{t_r t_s} \mid t_r \in R \wedge t_s \in S \wedge t_r[A] = t_s[B] \}$$

自然连接（Natural join）

*R*和*S*具有相同的属性组*B*

$$R \bowtie S = \{ \widehat{t_r t_s} \mid t_r \in R \wedge t_s \in S \wedge t_r[B] = t_s[B] \}$$

一般连接：从行的角度进行运算

自然连接与等值连接的区别：

自然连接要求两个关系中进行比较的属性或属性组必须同名和相同值域，而等值连接只要求比较属性有相同的值域；自然连接的结果中，同名的属性只保留一个；自然连接还需要取消重复列，所以是同时从行和列的角度进行运算。

连接之后，元组的数目可能是多少？

[例5]

R

A	B	C
a_1	b_1	5
a_1	b_2	6
a_2	b_3	8
a_2	b_4	12

S

B	E
b_1	3
b_2	7
b_3	10
b_3	2
b_5	2

$R \underset{C<E}{\bowtie} S$

A	R.B	C	S.B	E
a_1	b_1	5	b_2	7
a_1	b_1	5	b_3	10
a_1	b_2	6	b_2	7
a_1	b_2	6	b_3	10
a_2	b_3	8	b_3	10

外连接

- 左外连接
  - 列出左边关系中所有的元组
- 右外连接
  - 列出右边关系中所有的元组

除（Division）

给定关系R (X,Y) 和S (Y,Z)，其中X,Y,Z为属性组。R中的Y与S中的Y可以有不同的属性名，但必须出自相同的域集。R与S的除运算得到一个新的关系$P(X)$，P是R中满足下列条件的元组在X属性列上的投影：元组在X上分量值x的象集Y_x包含S在Y上投影的集合。

$$R \div S = \{t_r[X] \mid t_r \in R \wedge \pi_Y(S) \subseteq Y_x\}$$

其中Y_x：x在R中的象集，$x = t_r[X]$。

除（续）

R

A	B	C
a_1	b_1	c_2
a_2	b_3	c_7
a_3	b_4	c_6
a_1	b_2	c_3
a_4	b_6	c_6
a_2	b_2	c_3
a_1	b_2	c_1

S

B	C	D
b_1	c_2	d_1
b_2	c_1	d_1
b_2	c_3	d_2

$R \div S$
A
a_1

关系代数运算小结

关系代数运算：

◆ 并、差、交、笛卡尔积、投影、选择、连接、除

五种基本运算：

并、差、笛卡尔积、投影、选择

∪，　—，　×，　π，　σ

◆交、连接、除可以用5种基本运算来表达。引进它们并不增加语言的能力，但可以简化表达。

交、并、差 运算

R和S

具有相同的目*n*（即两个关系都有*n*个属性）
相应的属性取自同一个域

R∩S

其结果仍为*n*目关系，由既属于*R*又属于*S*的元组组成

$$R \cap S = \{ t \mid t \in R \wedge t \in S \}$$

R∪S

结果仍为*n*目关系，由属于*R*或属于*S*的元组组成

$$R \cup S = \{ t \mid t \in R \vee t \in S \}$$

R-S

其结果仍为*n*目关系，由属于*R*而不属于*S*的所有元组组成

$$R-S = \{ t \mid t \in R \wedge t \notin S \}$$

设有学生-课程关系数据库，它由三个关系组成，它们的模式是：学生S（学号SNO，姓名SN，所在系DEPT，年龄AGE）、课程C（课程号CNO，课程名CN，先修课号CPNO）、SC（学号SNO，课程号CNO，成绩SCORE）。

请用关系代数写出下列查询：

（1）检索学生的所有情况。

$\sigma(S)$ 或 $\sigma(S \bowtie SC \bowtie C)$

（2）检索学生年龄大于等于 20 岁的学生姓名。

$\Pi_{SN}(\sigma_{AGE>=20}(S))$

（3）检索先修课号为 C2 的课程号。

$\Pi_{CNO}(\sigma_{CPNO='C2'}(C))$

（4）检索课程号 C1 的成绩为 A 的所有学生姓名。

$\Pi_{sn}(S \bowtie (\sigma_{CNO='C1' \wedge G='A'}(SC)))$

第三章

[例2] 建立一个“学生选课”表SC，它由学号Sno、课程号Cno，修课成绩Grade组成，其中(Sno, Cno)为主码。

```
CREATE TABLE SC(
        Sno CHAR(5) ,
        Cno CHAR(3) ,
        Grade   int,
        Primary key (Sno, Cno));
```

```
CREATE TABLE 职工
     (
      职工号 CHAR(5) NOT NULL  PRIMARY KEY,
      姓名 VARCHAR(30) NOT NULL
        );

 CREATE TABLE 供应商
     (
      供应商号 CHAR(5) NOT NULL PRIMARY KEY,
      供应商名 VARCHAR(30) NOT NULL
        );
```

[例]建立订购单

```
CREATE TABLE 订购单
 (订购单号 CHAR(5) PRIMARY KEY,
  职工号 CHAR(5) NOT NULL FOREIGN KEY REFERENCES 职工,
  供应商号 CHAR(5) NULL FOREIGN KEY REFERENCES 供应商,
  订购日期 DATETIME DEFAULT getdate(),
  订购数量 INT CHECK(订购数量 >=10 AND 订购数量<=50));
```

```
CREATE TABLE 订购单
     (订购单号 CHAR(5),
      职工号 CHAR(5) NOT NULL ,
      供应商号 CHAR(5) NULL ,
      订购日期 DATE NULL DEFAULT SYSDATE,
      订购数量 INT CHECK(订购数量 >=10 AND 订购数量<=50),
      constraint pk_dd_ddno PRIMARY KEY(订购单号),
      FOREIGN KEY (职工号) REFERENCES 职工(职工号),
      FOREIGN KEY (供应商号)  REFERENCES 供应商(供应商号)
     );

INSERT INTO 订购单
(订购单号,职工号,供应商号,订购数量)
VALUES('0001','90001','1001',20);
```

```
--外键在DM和KINGBASEES中可以简写为:

CREATE TABLE 订购单
   (订购单号 CHAR(5) PRIMARY KEY,
    职工号 CHAR(5) NOT NULL REFERENCES 职工(职工号),
    供应商号 CHAR(5) NULL REFERENCES 供应商(供应商号) ,
    订购日期 DATE NULL DEFAULT SYSDATE,
    订购数量 INT CHECK(订购数量 >=10 AND 订购数量<=50)
);
```

```
--外键在DM和KINGBASEES中,或简写为:
CREATE TABLE 订购单
      (订购单号 CHAR(5) PRIMARY KEY,
       职工号 CHAR(5) NOT NULL REFERENCES 职工,
       供应商号 CHAR(5) NULL REFERENCES 供应商,
       订购日期 DATE NULL DEFAULT SYSDATE,
       订购数量 INT CHECK(订购数量 >=10 AND 订购数量<=50)
       );
```

综合查询

[例] 求各个课程号及相应的选课人数。

```
SELECT Cno,count(sno)
FROM   SC
GROUP BY cno;
```

结果：

Cno	COUNT(Sno)
1	22
2	34
3	44
4	33
5	48

思考： 如果去掉GROUP BY Cno，结果是什么？（出错）

插入单个元组

[例1] 插入一个新学生记录（学号：200215128；姓名：陈冬；性别：男；所在系：IS；年龄：18岁）插入到Student表中。

```
INSERT INTO Student
VALUES ('200215128','陈冬','男','IS',18);
```

[例3] 插入一条选课记录('200215128','1')。

```
INSERT  INTO SC(Sno,Cno)
VALUES ('200215128','1');
```

新插入的记录在Grade列上取空值

修改元组的值

[例6] 将信息系所有学生的年龄增加**1**岁。

```
UPDATE Student
SET Sage= Sage+1
WHERE Sdept='IS';
```

例题

[例7] 修改SC的Grade的值

```
UPDATE SC   SET Grade =80
WHERE Sno='2018100123' AND Cno='321100'
```

[例8] 删除Student表

```
DROP TABLE   Student  CASCADE;
```

删除语句

[例10] 删除计算机科学系所有学生的选课记录。

```
DELETE
FROM SC
WHERE  'CS'=
   (SELETE Sdept
    FROM Student
    WHERE Student.Sno=SC.Sno)；
```

关于空值

NULL是一个常量，仅在数值和字符串类型的列中有意义，代表的是没有意义或者是不确定的值。

不能写“=NULL”

而是“ is NULL”

计算COUNT, AVEAGE等聚集函数时，自动忽略空值；

排序时空值大于任一具体的值。

视图的特点

虚表，是从一个或几个基本表（或视图）导出的表；

只存放视图的定义，不存放视图对应的数据；

基表中的数据发生变化，从视图中查询出的数据也随之改变。

单个用户使用的数据视图的描述称为外模式。

视图的优点

（1）视图能够简化用户的操作；

（2）视图使用户能以多种角度看待同一数据；

（3）视图对重构数据库提供了一定程度的逻辑独立性；

（4）视图能够对机密数据提供安全保护。

建立视图

[例2] 建立信息系学生的视图，并要求透过该视图进行的更新操作只涉及信息系学生。

```
CREATE VIEW IS_Student
    AS
    SELECT Sno,Sname,Sage
    FROM  Student
    WHERE  Sdept= 'IS'
    WITH CHECK OPTION;
```

WITH CHECK OPTION

将来对视图进行增删改操作时，**DBMS**会自动加上子查询中的条件表达式。

练习：

5．设有如下四个基本表 S，C，SC，T，结构如图 3-20 所示。

S

S# 学号	SN 学生姓名	AGE 年龄	DEPT 所在系
S1	丁一	20	计算机
S2	王二	19	计算机
S3	张三	19	外语
…			

C

C# 课程号	CN 课程名称
C1	数据库
C2	操作系统
C3	微机原理
…	

SC

S# 学号	C# 课程号	GR 成绩
S1	C1	80
S1	C2	89
S2	C3	59
…		

T

T# 教师号	TN 教师姓名	SAL 工资	COMM 职务津贴	C# 所讲课程
T1	王力	800		C1
T2	张兰	1200	300	C2
T3	李伟	700	150	C1
…				

图 3-20　某教学数据库实例

（1）用SQL的DDL语言创建表

答：

```
create table S
(
S# char(10) primary key not null,
SN varchar(30) not null,
AGE tinyint null,
DEPT nchar(30)
);
create table C (
 C# char(6) primary key,
CN varchar(20) not null );
```

```
create table T
(
T# char(5) primary key not null,
TN varchar(30) not null,
PROF varchar(10),
SAL BIGINT,
COMM BIGINT,
C# char(6)
);
```

```
create table SC (
 S# char(10) not null,
C# char(6)  not null,
GR tinyint    CHECK (GR >=0  AND GR <=100),
 constraint pk_s_c PRIMARY KEY (c#,s#),
 constraint fk_s FOREIGN KEY (S#) REFERENCES S (S#),
constraint fk_c FOREIGN KEY (C#) REFERENCES C (C#)
);
```

（2）创建计算机系学生的视图，该视图的属性列由学号、姓名、课程号和任课教师号组成。

```
Create  view  ComputerDEPT(S#,SN,C#,T#)
as
select S.S#,SN,C.C#,T# from S,C,SC,T
where DEPT in ('计算机')  and S.S#=SC.S#  and C.C#=SC.C#
and T.C#=C.C# ;
```

（3）检索计算机系年龄在20岁以上的学生学号。

答：

```
select   S#   from   S
where DEPT in ('计算机')  and AGE>20;
```

（4）检索姓王的教师所讲课程的课程号及课程名称。

答：

```
select C.C#,CN from C,T
where T# like '王%'  and C.C#=T.C# ;
```

（5）检索张三同学所学课程的成绩，列出SN，C#，GR。

答：
```
select SN,C.C#,GR from S,C,SC
where SN='张三' and S.S#=SC.S#  and C.C#=SC.C# ;
```

（6）检索选修总收入超过1000元的教师所讲课程的学生姓名、课程号和成绩。

答：
```
select SN,SC.C#,GR from S,SC,T
where (SAL+COMM)>1000   and S.S#=SC.S#   and SC.C#=T.C#;
```

（7）检索没有选修C1课程且选修课程数为两门的学生的姓名和平均成绩，并按平均成绩降序排列

答：
```
select SN,avg(GR) from S,SC
where (S.S#=SC.S#)  and('C1' not in (select C# from SC
where S.S#=SC.S#))
group by SN
having count(SC.S#)=2 order by avg(gr)  desc ;
```

（8）检索选修和张三同学所课程中任意一门相同的学生姓名，课程名。

答：
```
select SN,CN from S,C,SC
where S.S#=SC.S# and C.C#=SC.C# and SC.C# in (select
SC.C# from SC where S#='S3');
```

（9）S1同学选修了C3,将此信息插入SC表中。

答： insert into SC(S#,C#) values('S1','C3');

（10）删除S表中没有选修任何课程的学生记录。

答：
```
delete from S
 where S# not in(select S# from SC) ;
```

第四章 数据库安全性

- 通过 SQL 的 GRANT 语句和 REVOKE 语句实现
- 用户权限组成
  - 数据对象
  - 操作类型
- 定义用户存取权限：定义用户可以在哪些数据库对象上进行哪些类型的操作
- 定义存取权限称为授权

第四章 数据库安全性

- 数据库系统的安全机制
  - 用户标识和鉴别
  - 存取控制
    - 自主存取控制（DAC）
      - GRANT语句
      - REVOKE语句
    - 强制存取控制（MAC）
    - 基于角色的访问控制**定义存取权限称为授权**
  - 其它安全控制方法
    - 视图机制
    - 审计
    - 数据加密
    - 统计数据库安全性
- 数据库安全机制的设计目标
  - 试图破坏安全的人所花费的代价 >> 得到的利益

第四章 数据库安全性

- 强制存取控制提供了更高级别的数据库安全性
  - 强制存取控制（MAC）是对数据本身进行密级标记
  - 无论数据如何复制，标记与数据是一个不可分的整体
  - 只有符合密级标记要求的用户才可以操纵数据，从而提供了更高级别的安全性。

授权： 例题

［例1］把查询**Student**表权限授给用户U1

```
GRANT   SELECT
ON   TABLE   Student
TO   U1;
```

回收授权： REVOKE（续）

[例9] 收回用户**CC**对表**SC**的查询权限

```
REVOKE SELECT
ON TABLE SC
FROM CC;
```

第五章 数据库完整性

数据库的完整性与数据库的安全性

数据的完整性和安全性是两个不同的概念,但是有一定的联系。

完整性是为了防止数据库中存在不符合语义的数据,防止错误信息的输入和输出,即所谓垃圾进垃圾出(Garbage In Garbage Out)所造成的无效操作和错误结果。

安全性是保护数据库防止恶意的破坏和非法的存取。 也就是说,安全性措施的防范对象是非法用户和非法操作,完整性措施的防范对象是不合语义的数据。

第五章 数据库完整性

实体完整性检查和违约处理

1. 检查主码值是否唯一，如果不唯一则拒绝插入或修改；
2. 检查主码的各个属性是否为空，只要有一个为空就拒绝插入或修改。

参照完整性违约处理

1. 拒绝执行（默认策略）；
2. 级联操作；
3. 设置为空值（SET-NULL）：对于参照完整性，除了应该定义外码，还应定义外码列是否允许空值。

第五章 数据库完整性

DBMS的完整性控制机制的功能

定义功能：即提供定义完整性约束条件的机制；

检查功能：即检查用户发出的操作请求是否违背了完整性约束条件；

违约反应：如果发现用户的操作请求使数据违背了完整性约束条件，则采取一定的动作来保证数据的完整性。

第五章 数据库完整性

实体完整性

参照完整性

用户定义的完整性

例：判别以下约束属于哪种完整性约束？

学生的年龄必须是14-29的整数；

学生的性别只能是男或女；

学生的学号一定是唯一的；

学生所在的系必须是学校开设的系；

第五章 数据库完整性

实体完整性

参照完整性

用户定义的完整性

例：判别以下约束属于哪种完整性约束？

学生的年龄必须是14-29的整数；

学生的性别只能是男或女；

学生的学号一定是唯一的；

学生所在的系必须是学校开设的系；

1. 不允许取空值

[例5.5]在定义SC表时，说明Sno、Cno属性不取空

```
CREATE TABLE SC
    ( Sno CHAR(8)  NOT NULL,
      Cno CHAR(4)  NOT NULL,
      Grade SMALLINT,
      PRIMARY KEY (Sno, Cno) );
```

--因为是两个属性，必须用表级约束定义

2. 列值惟一

[例5.6]建立部门表Dept，要求部门名称Dname惟一，部门编号为主码。

```
CREATE TABLE Dept
    ( Dno NUMERIC(2) NOT NULL,
      Dname CHAR(20) UNIQUE,
      Location CHAR(10),
      PRIMARY KEY (Dno)
      );
```

3. 用CHECK短语指定列值应该满足的条件

[例5.8]在定义SC表时，说明Grade应该在0和100之间。

```
CREATE TABLE SC
    ( Sno CHAR(5) NOT NULL,
      Cno CHAR(2) NOT NULL,
      Grade SMALLINT CHECK (Grade>=0
                            AND Grade<=100),
      PRIMARY KEY (Sno, Cno)
      ……
    );
```

第六章 关系数据理论

规范化理论 用来改造关系模式，通过分解关系模式来消除其中不合适的数据依赖，以解决插入异常、删除异常、更新异常和数据冗余问题。

第六章 关系数据理论

函数依赖： 设R(U)是一个属性集U上的关系模式，X和Y是U的子集。若对于R(U)的任意一个可能的关系r，r中不可能存在两个元组在X上的属性值相等，而在Y上的属性值不等，则称“X函数确定Y”或“Y函数依赖于X”，记作X→Y。

完全函数依赖与部分函数依赖

定义6.2 在**R(U)**中，如果**X→Y**，并且对于**X**的任何一个真子集**X'**，都有**X'↛Y**，则称**Y**对**X**完全函数依赖，记作

$X \xrightarrow{F} Y$。

若**X→Y**，但**Y**不完全函数依赖于**X**，则称**Y**对**X**部分函数依赖，记作$X \xrightarrow{P} Y$。

平凡函数依赖与非平凡函数依赖

在关系模式R(U)中，对于U的子集X和Y，

如果X→Y，但Y ⊈ X，则称X→Y是非平凡的函数依赖

若X→Y，但Y ⊆ X， 则称X→Y是平凡的函数依赖

- 例：在关系SC(Sno, Cno, Grade)中，

  非平凡函数依赖： (Sno, Cno) → Grade

  平凡函数依赖： (Sno, Cno) → Sno

  (Sno, Cno) → Cno

范式

- 范式是符合某一种级别的关系模式的集合
- 关系数据库中的关系必须满足一定的要求。满足不同程度要求的为不同范式
- 范式的种类：

第一范式(1NF)
第二范式(2NF)
第三范式(3NF)
BC范式(BCNF)
第四范式(4NF)
第五范式(5NF)

范式

- 各种范式之间存在联系：

$$1NF \supset 2NF \supset 3NF \supset BCNF \supset 4NF \supset 5NF$$

- 某一关系模式**R**为第**n**范式，可简记为**R**∈**n**NF。
- 一个低一级范式的关系模式，通过模式分解可以转换为若干个高一级范式的关系模式的集合，这种过程就叫规范化

6.2.4 2NF

- **1NF的定义**

  如果一个关系模式R的所有属性都是不可分的基本数据项，则R∈1NF

- **2NF的定义**

  定义6.6 若R∈1NF，且每一个非主属性完全函数依赖于码，则R∈2NF。

如果一个关系中存在着传递依赖，则它最高只是第一范式。

3NF

- **3NF的定义**

定义6.7 关系模式$R<U,F>$ 中若不存在这样的码X、属性组Y及非主属性$Z(Z\subsetneq Y)$, 使得$X\rightarrow Y$，$Y\rightarrow Z$成立， $Y\not\rightarrow X$，则称$R<U,F>\in 3NF$。

■若$R\in$3NF，则每一个非主属性既不部分依赖于码也不传递依赖于码。

关系模式R中的属性全是主属性，则R的最高范式必定是3NF

包含在任何一个候选码中的属性，称为主属性

3NF（续）

例：2NF关系模式S-L(Sno, Sdept, Sloc)中

– 函数依赖：

Sno→Sdept

Sdept ↛ Sno

Sdept→Sloc

可得：Sno $\xrightarrow{传递}$ Sloc，即S-L中存在非主属性对码的传递函数依赖，S-L∉*3NF*

BC范式（BCNF）

- 定义6.8　关系模式**R<U,F>∈1NF**，若**X→Y**且**Y ⊈ X**时**X**必含有码，则**R<U,F>∈BCNF**。

- 等价于：每一个决定属性因素都包含码

BCNF（续）

包含在任何一个候选码中的属性，称为主属性

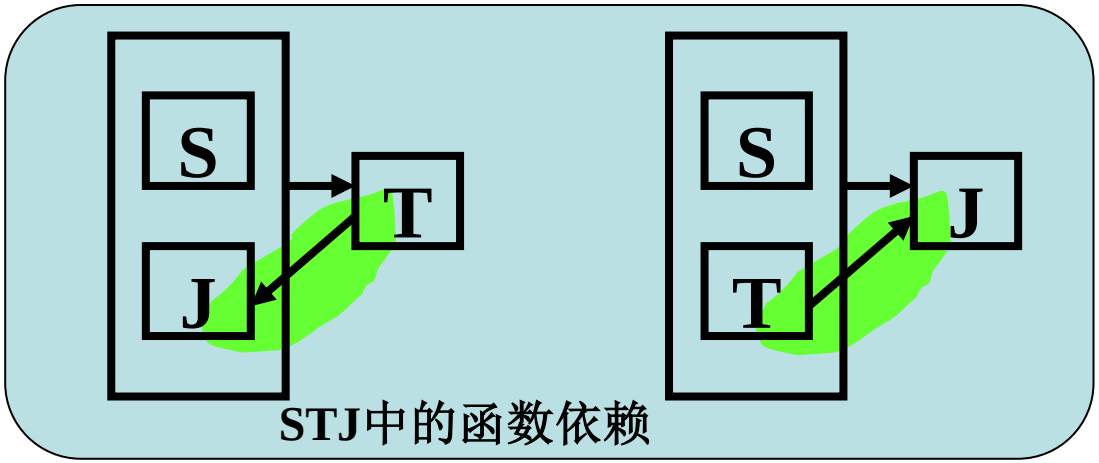
- 若R∈BCNF
  - 所有非主属性对每一个码都是完全函数依赖
  - 所有的主属性对每一个不包含它的码，也是完全函数依赖
  - 没有任何属性完全函数依赖于非码的任何一组属性
- R ∈BCNF ⇄（充分 / 不必要） R ∈3NF

BCNF

[例8] 在关系模式STJ(S,T,J)中，S表示学生，T表示教师，J表示课程。每个老师只教一门课。

函数依赖：(S,J)→T，(S,T)→J，T→J

(S,J)和(S,T)都是候选码，主属性J部分依赖于码(S, T)



这个模式体现不出每个老师只教一门课：同一教师不同学生，教不同的课，违反了约束，但这些元组依然能放进模式里

BCNF（续）

- **STJ∈3NF**
  - 没有任何非主属性对码传递依赖或部分依赖
- **STJ∉BCNF**
  - T是决定因素，T不包含码

3NF / BCNF

[例8] 在关系模式STJ(S,T,J)中，S表示学生，T表示教师，J表示课程。每个老师只教一门课。

– 函数依赖： (S,J)→T，(S,T)→J，T→J

解决方法：将STJ分解为二个关系模式：

ST(S,T)∈BCNF，

TJ(T,J)∈BCNF

STJ∈3NF

没有任何非主属性对码传递依赖或部分依赖

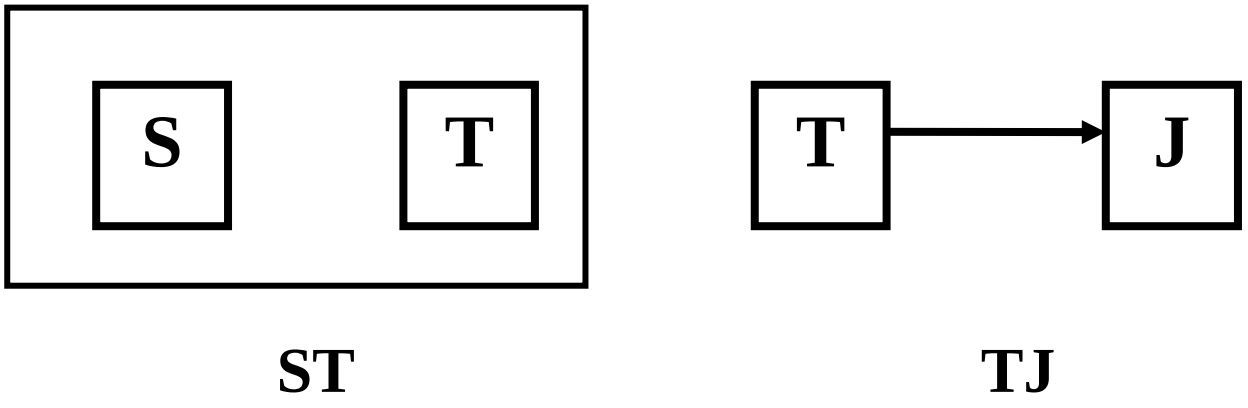
STJ∉BCNF

T是决定因素，T不包含码

BCNF（续）

解决方法：将STJ分解为二个关系模式：

ST(S,T)∈BCNF， TJ(T,J)∈BCNF



没有任何属性对码的部分函数依赖和传递函数依赖

3NF与BCNF的关系

- R ∈BCNF ⇄ R ∈3NF（充分；不必要）

- 如果R∈3NF，且R只有一个候选码

R ∈BCNF ⇄ R ∈3NF（充分；必要）

- 如果只考虑函数依赖，则属于**BCNF**的关系模式规范化程度已经是最高的了。
- 如果考虑多值依赖，则属于**4NF**的关系模式规范化程度已经是最高的了。
- 连接依赖不是由语义直接导出，而是在关系连接时才反映出来。如果消除了**4NF**关系模式中的连接依赖，则达到**5NF**。

规范化小结

- 关系模式规范化的基本步骤

消除决定属性集非码的非平凡函数依赖

1NF
↓ 消除非主属性对码的部分函数依赖
2NF
↓ 消除非主属性对码的传递函数依赖
3NF
↓ 消除主属性对码的部分和传递函数依赖
BCNF
↓ 消除非平凡且非函数依赖的多值依赖
4NF

范式	举例、问题、达到下一范式的方法（消除）
非规范	表中有表（消除之，达到每个分量都不可分）
1NF	**(Sno, Cno, Sdept, Mname, Grade)** **非主属性部分函数依赖于码,如Sdept部分函数依赖于(Sno, Cno)**
2NF	**(Sno, Sdept, Sloc)** **非主属性对码的传递函数依赖，如Sloc传递函数依赖于Sno**
3NF	**STJ(S,T,J), (S,J)→T，(S,T)→J，T→J** **主属性对不包含其的码的部分和传递函数依赖** **如J对不包含其的码的部分函数依赖(S,T)**
BCNF	**（BCNF的问题部分是选修内容，不做要求）**

习题

建立一个关于系、学生、班级、学会等诸信息的关系数据库。

学生：学号、姓名、出生年月、系名、班号、宿舍区。

班级：班号、专业名、系名、人数、入校年份。

系：系名、系号、系办公地点、人数。

学会：学会名、成立年份、办公地点、人数。

语义如下：一个系有若干专业，每个专业每年只招一个班，每个班有若干学生。一个系的学生住在同一宿舍区。每个学生可参加若干学会，每个学会有若干学生。学生参加某学会有一个入会年份。

请给出关系模式，写出每个关系模式的极小函数依赖集，指出是否存在传递函数依赖，对于函数依赖左部是多属性的情况讨论函数依赖是完全函数依赖，还是部分函数依赖。指出各关系模式的候选码、外部码，有没有全码存在?

习题

(1)关系模式如下：

学生：**S(Sno,Sname,Sbirth,Dept,Class,Rno)**

班级：**C(Class,Pname,Dept,Cnum,Cyear)**

系：**D(Dept,Dno,Office,Dnum)**

学会：**M(Mname,Myear,Maddr,Mnum)**

习题

学生：学号、姓名、出生年月、系名、班号、宿舍区。

学生：S(Sno,Sname,Sbirth,Dept,Class,Rno)

(2) 关系模式的最小函数依赖集如下：

学生S(Sno,Sname,Sbirth,Dept,Class,Rno)的函数依赖:

Sno→Sname，Sno→Sbirth，Sno→Class，Class→Dept，Dept→Rno

传递依赖如下：

由于Sno→Dept，而Dept↛Sno ，Dept→Rno（宿舍区）

所以Sno与Rno之间存在着传递函数依赖。

由于Class→Dept，Dept ↛ Class，Dept→Rno

所以Class与Rno之间存在着传递函数依赖。

由于Sno→Class，Class↛Sno，Class→Dept

所以Sno与Dept之间存在着传递函数依赖。

习题

班级：班号、专业名、系名、人数、入校年份

班级：C(Class,Pname,Dept,Cnum,Cyear)

班级C(Class，Pname，Dept，Cnum，Cyear)的函数依赖:

Class→Pname，Class→Cnum，Class→Cyear，Pname→Dept.

由于Class→Pname，Pname↛Class，Pname→Dept

所以Class与Dept之间存在着传递函数依赖。

设有关系模式：

TEACHER(教师编号，教师姓名，电话，所在部门，借阅图书编号，书名，借书日期，还书日期，备注)

（1）教师编号是候选码吗？

（2）该关系模式的主码是什么？

（3）该关系模式是否存在部分函数依赖？如果存在，请写出至少两个？

（4）该关系模式满足第几范式？

（5）将该关系模式分解为3NF。

（1）教师编号不是候选码。

（2）主码为：教师编号，借阅图书编号；

非主属性为：教师姓名、电话、所在部门、书名、 借书日期、还书日期、备注

（3）存在。

（教师编号，借阅图书编号）→教师姓名

（教师编号，借阅图书编号）→教师电话

（教师编号，借阅图书编号）→所在部门

（教师编号，借阅图书编号）→书名

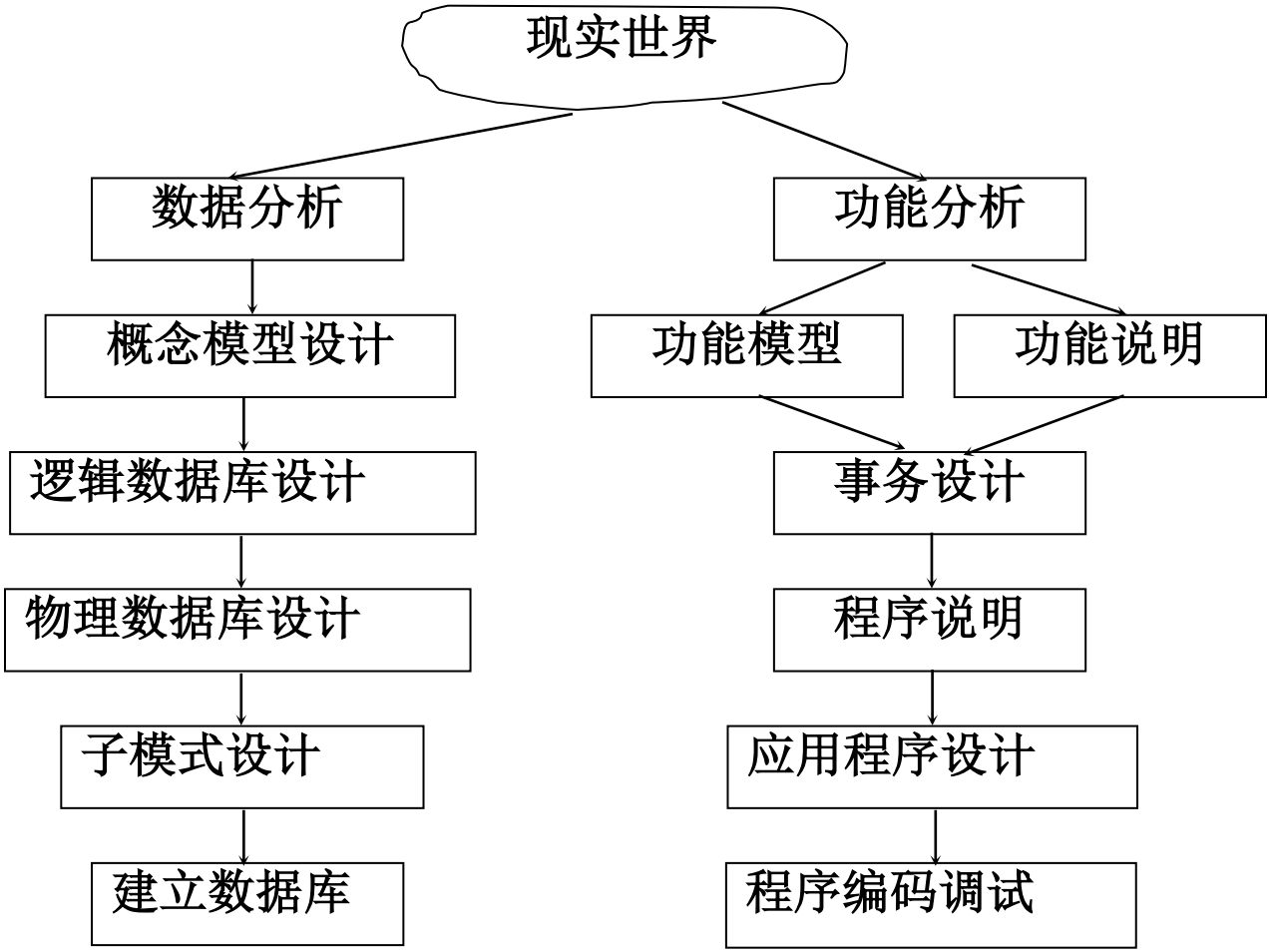
（4）因为存在非主属性对于码的部分函数依赖，所以，未达到2范式（2NF），只属于1范式(1NF)。

（5）教师（教师编号，教师姓名，电话，所在部门）

图书（图书编号，图书名）

借阅（教师编号，图书编号，借书日期， 还书日期，备注）

第七章 数据库设计



结构和行为分离的设计

数据库设计的特点

- 数据库建设的基本规律
  - 三分技术，七分管理，十二分基础数据
- 结构(数据)设计和行为(处理)设计相结合
  - 将数据库结构设计和数据处理设计密切结合

数据库设计的过程(六个阶段)

1. 需求分析阶段

准确了解与分析用户需求（包括数据与处理）

最困难、最耗费时间的一步

2. 概念结构设计阶段（运用E-R图作为模型设计工具）

整个数据库设计的关键

通过对用户需求进行综合、归纳与抽象，形成一个独立于具体DBMS的概念模型

3. 逻辑结构设计阶段

将概念结构转换为某个DBMS所支持的数据模型

对其进行优化

数据库设计的过程(六个阶段)

4.数据库物理设计阶段

为逻辑数据模型选取一个最适合应用环境的物理结构（包括存储结构和存取方法，例如，创建什么索引等）

5.数据库实施阶段

运用DBMS提供的数据库语言（如SQL）及宿主语言，根据逻辑设计和物理设计的结果；建立数据库；编制与调试应用程序；组织数据入库；进行试运行

6.数据库运行和维护阶段

数据库应用系统经过试运行后即可投入正式运行

在数据库系统运行过程中必须不断地对其进行评价、调整与修改

E-R图转换为表的规则

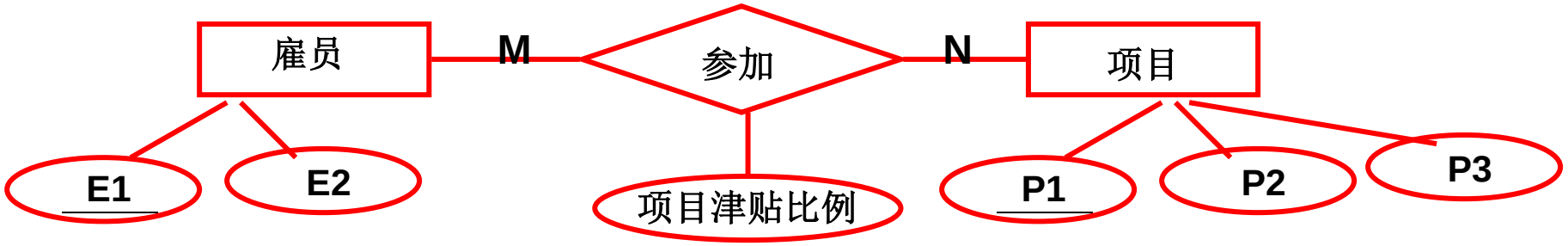
（1）实体的转换规则

每个实体转化为一张表，表中包含了除多值属性以外的所有属性，表的主键是实体的主标识，对于组合属性将其子属性转化为表中的属性。

E-R图转换为表的规则

（2）二元关系（无多值属性）

M:N 要转化为3张表，两个实体各1张表，每个实体的主标识转化为表的主键。关系的属性及两个实体的主标识形成关系表。关系表的主键为两个实体主标识的组合。



参加表（E1,P1,percent） （percent为项目津贴比例）

雇员表(E1,E2)

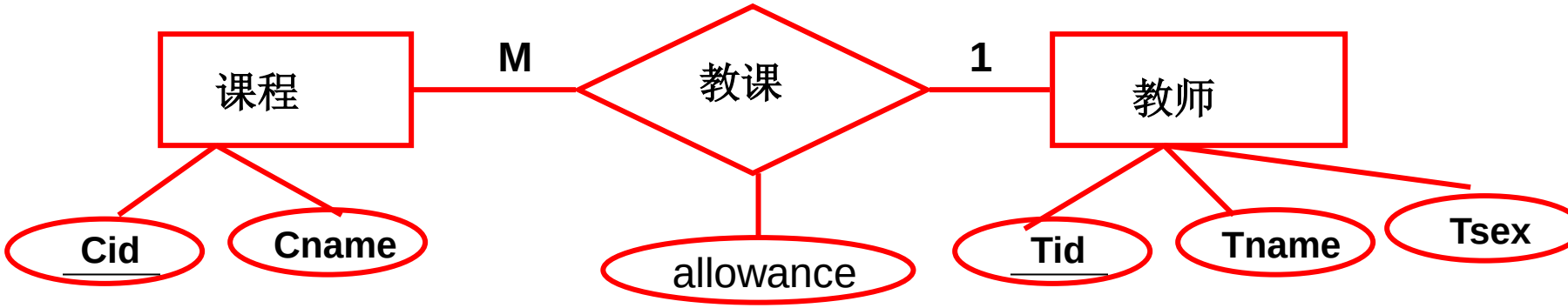
项目表(P1,P2,P3)

1:M　要转化为2张表，两个实体各1张表，将关系的属性及非多方的主标识加入到多方表，非多方表的主键是非多方实体的主标识，多方表的主键是多方实体的主标识，多方表的外键是非多方实体的主标识。

例子：实体转化的表如下

教师（**Tid，Tname，Tsex**）

课程（**Cid，Cname**）



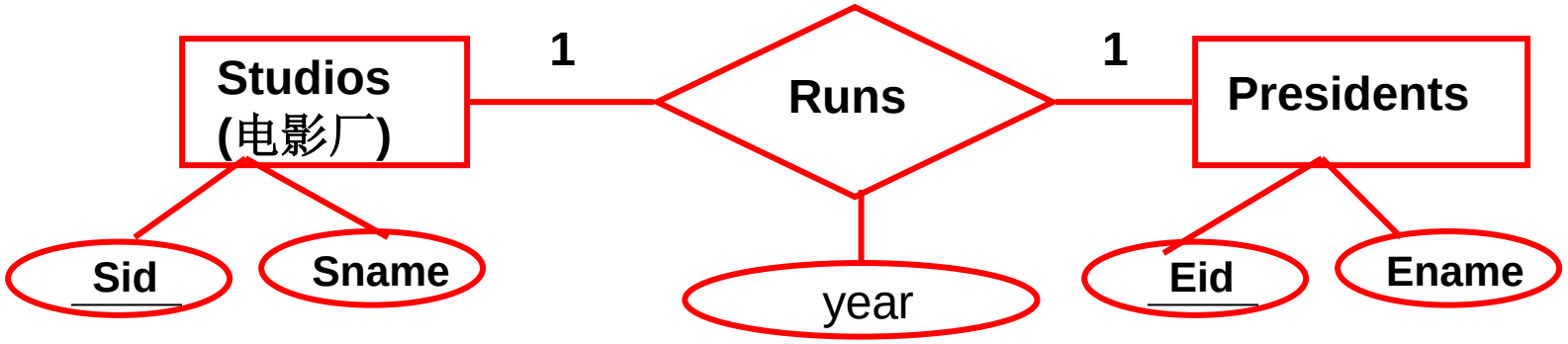
关系转化后形成的表

老师表（Tid，Tname，Tsex）

课程表（Cid，Cname，Tid，**allowance**）

外键

1:1　要转化为2张表。两个实体各1张表，两个实体各1张表，每个实体的主标识转化为表的主键。将关系的属性及任意一方（实体A）的主标识加入到另一个实体的表中，此表的主键不变，外键是实体A的主标识



方案1：presidents（Eid，Ename）

studios（Sid，Sname，Eid ，year ）

外键

外键

方案2： presidents（Eid，Ename ，Sid ，year）

studios（Sid，Sname）

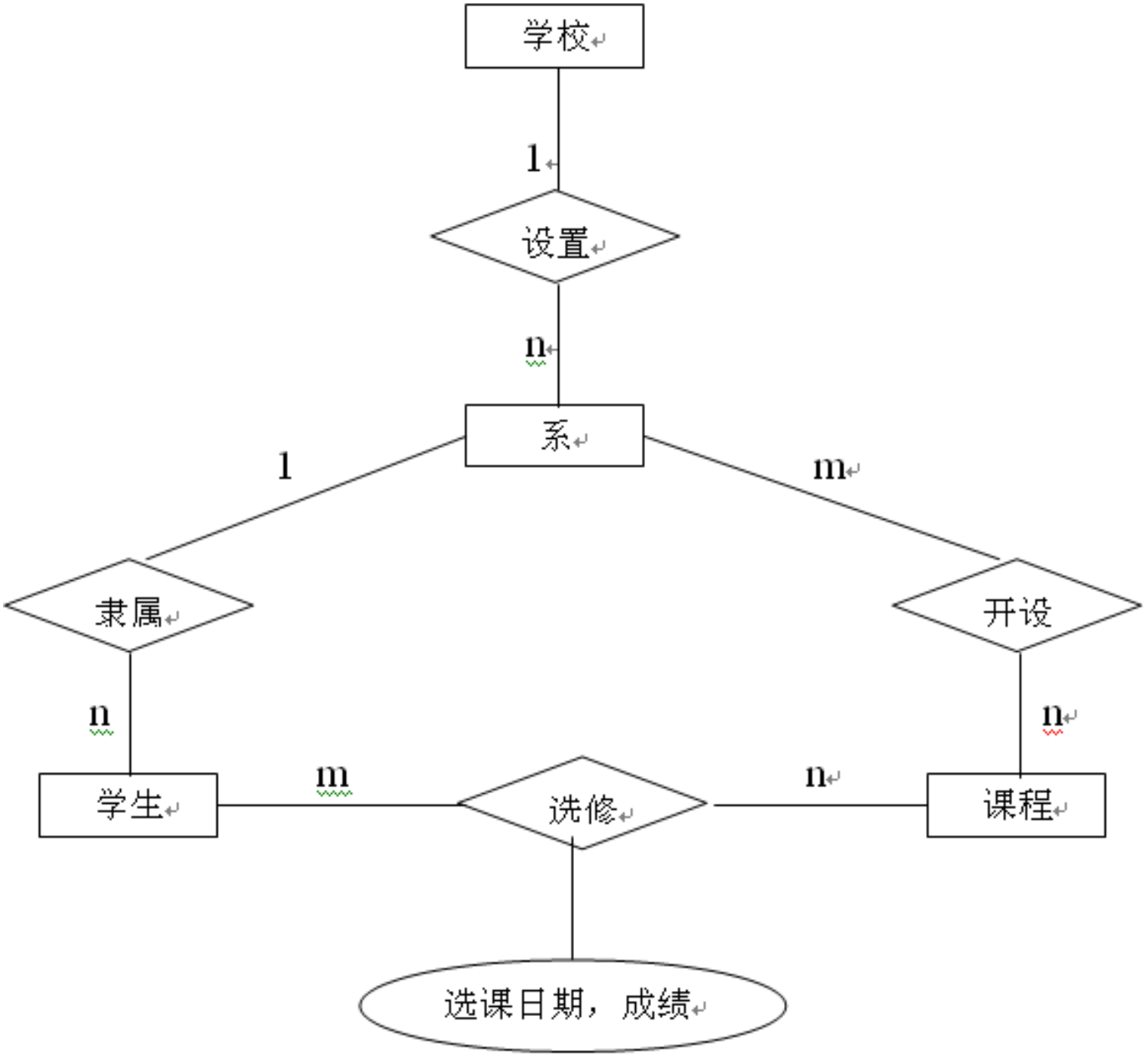
补充

某学校有若干系，每个系有若干学生、若干课程，每个学生选修若干课程，每门课有若干学生选修，某一门课可以为不同的系开设，现要建立该校学生选修课程的数据库。请设计：

（1）关于此学校数据库的E-R图；

（2）把E-R图转换为关系模型。

E-R图全称是什么？它用于建立数据库的概念模型。


学校
1
设置
n
系
1
m
隶属
开设
n
n
学生
m
选修
n
课程
选课日期，成绩

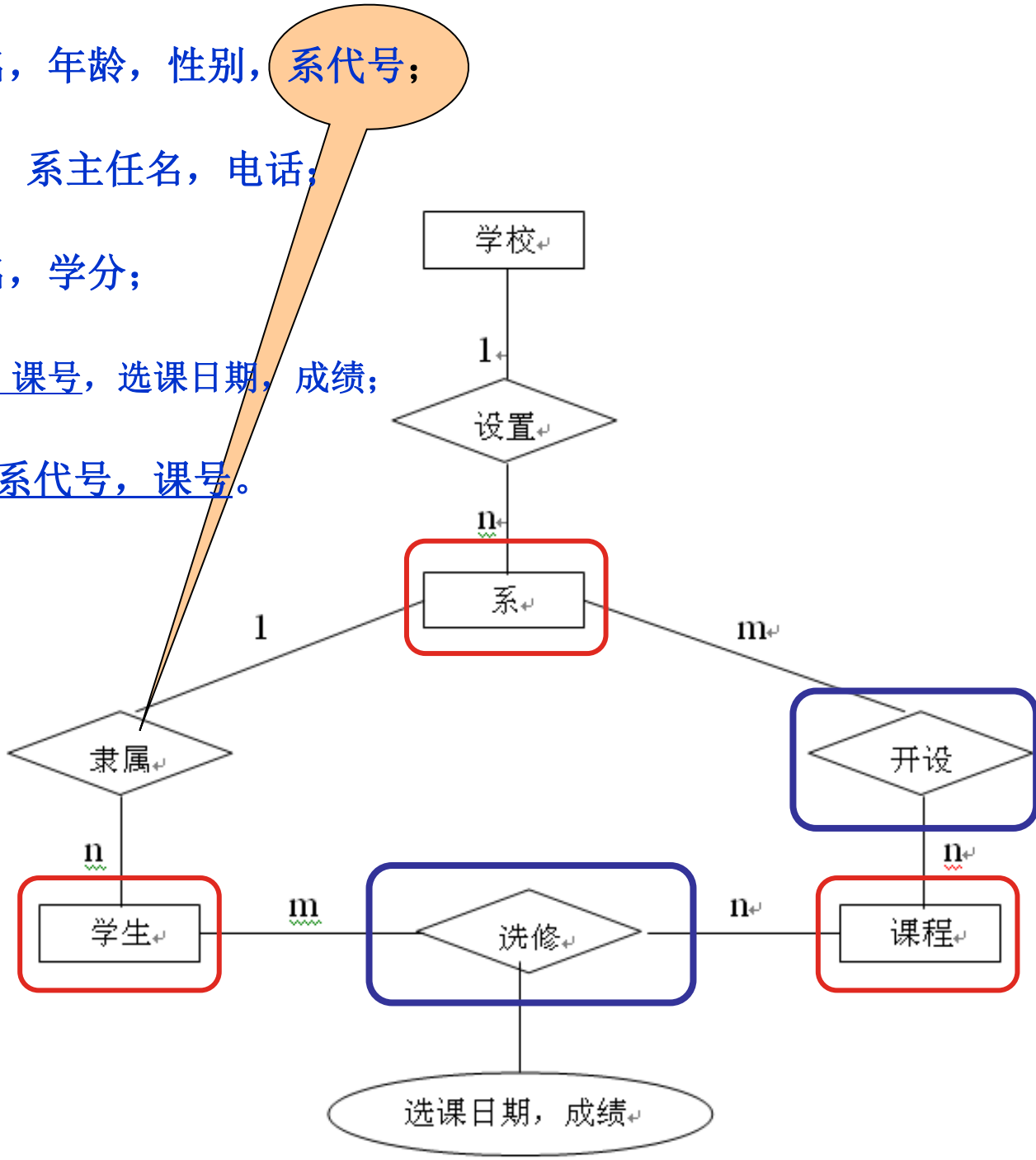
学生关系： 学号，姓名，年龄，性别，系代号；

系关系： 系代号，系名，系主任名，电话；

课程关系： 课号，课名，学分；

学生选课关系：学号，课号，选课日期，成绩；

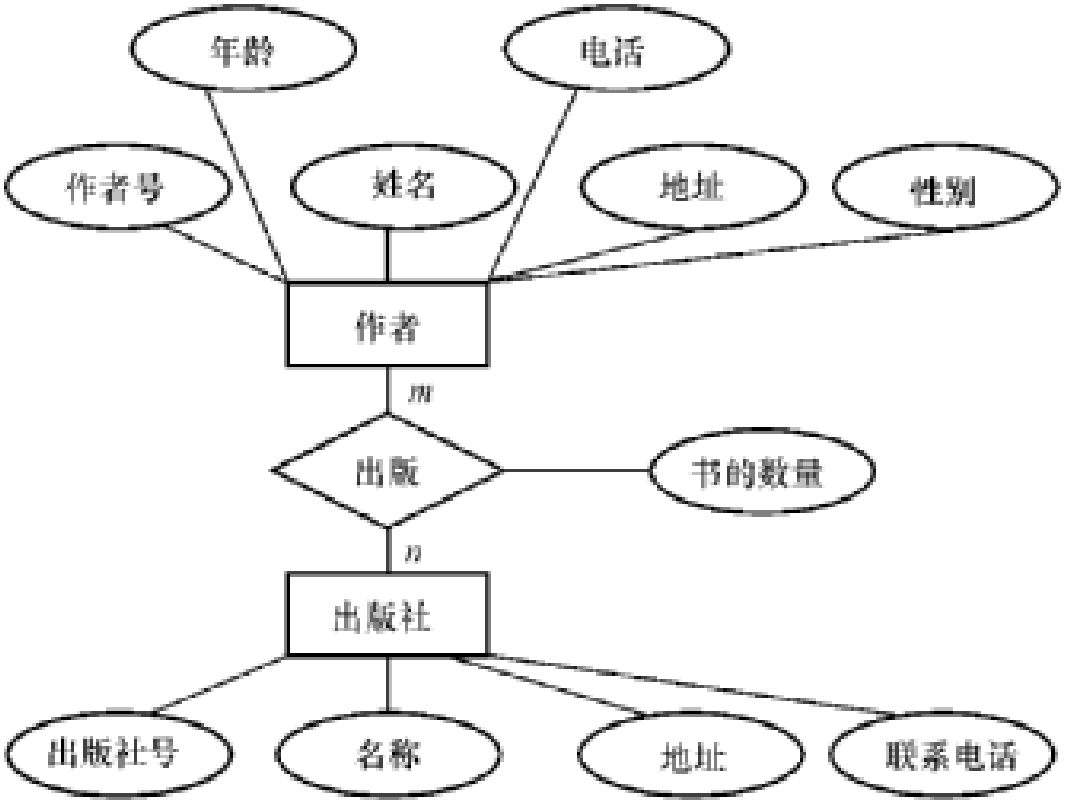
系开设课程关系： 系代号，课号。



习题

现有一局部应用，包括两个实体：“出版社”和“作者”，这两个实体是多对多的联系，请读者自己设计适当的属性，画出E-R 图，再将其转换为关系模型（包括关系名、属性名、码和完整性约束条件）。

E-R 图为：



习题18

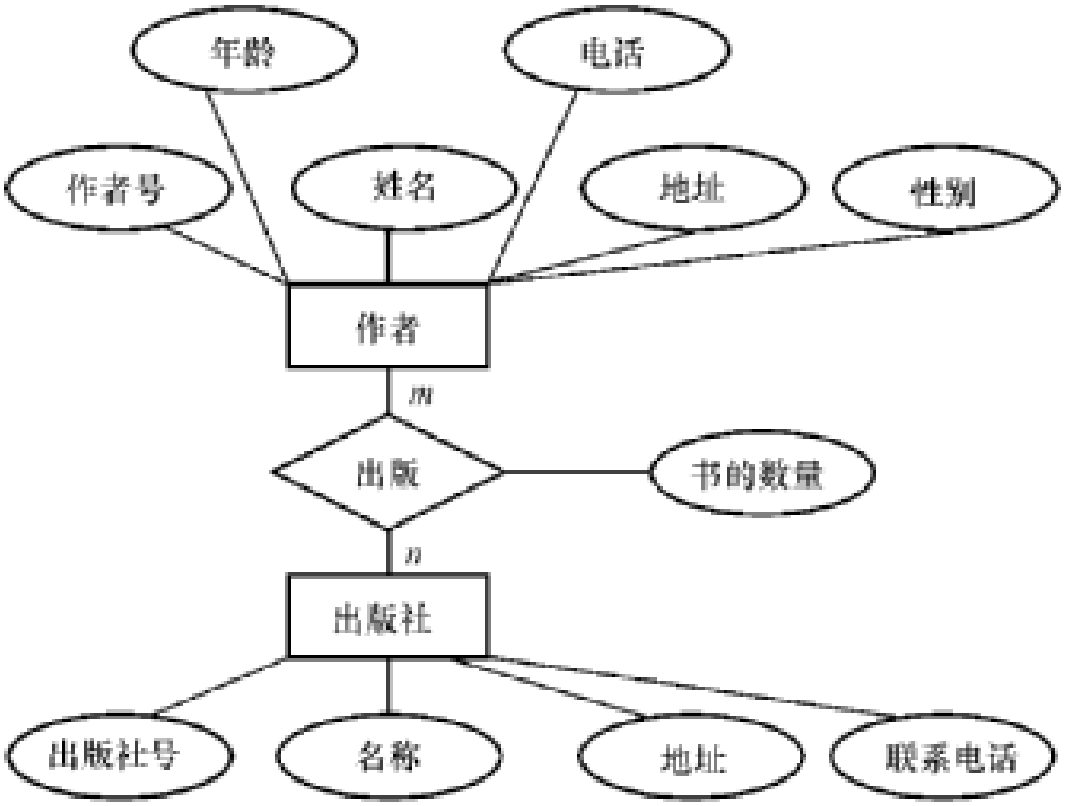
E-R 图为：

作者（作者号，姓名，年龄，性别，电话，地址）

出版社（出版社号，名称，地址，联系电话）

出版（作者号，出版社号，书的数量）

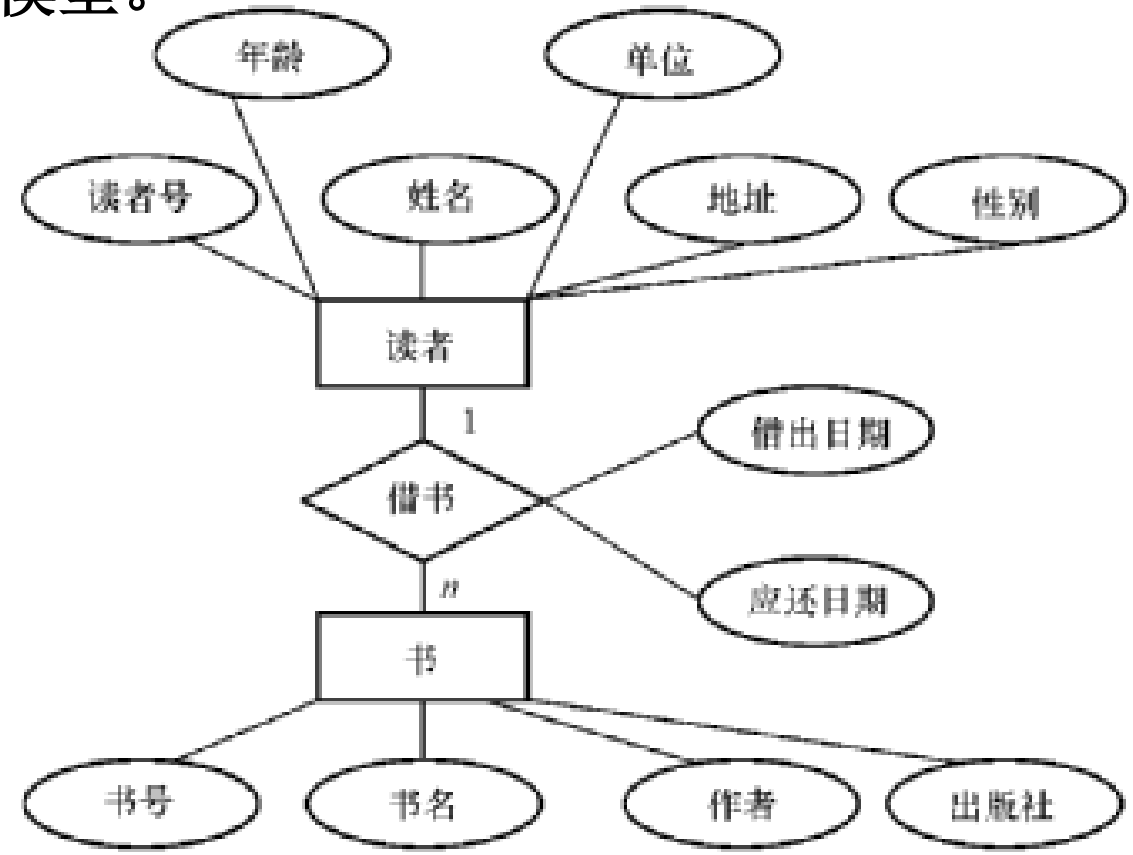
出版关系的主码作者号，出版社号分别参照作者关系的主码作者号和出版社关系的主码出版社号。



习题19

请设计一个图书馆数据库，此数据库中对每个借阅者保存读者记录，包括：读者号，姓名，地址，性别，年龄，单位。对每本书存有：书号，书名，作者，出版社。对每本被借出的书存有读者号、借出日期和应还日期。要求：给出**E-R**图，再将其转换为关系模型。

按每本书均给编号



习题19

读者（读者号，姓名，性别，年龄，单位，地址）

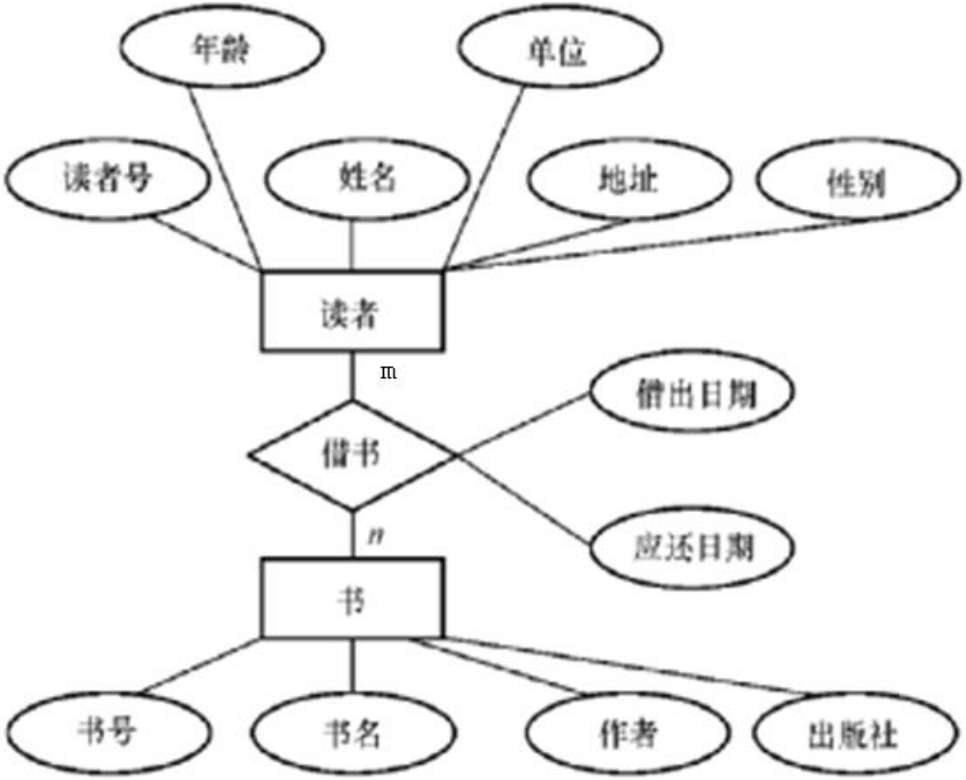
书（书号，书名，作者，出版社）

借书（书号，读者号，借出日期，应还日期）

按每本书均给编号

习题19

请设计一个图书馆数据库，此数据库中对每个借阅者保存读者记录，包括：读者号，姓名，地址，性别，年龄，单位。对每本书存有：书号，书名，作者，出版社。对每本被借出的书存有读者号、借出日期和应还日期。要求：给出**E-R**图，再将其转换为关系模型。



- 按每种书给一个编号，并记录每种书数量

习题19

读者（读者号，姓名，性别，年龄，单位，地址）

书（书号，书名，作者，出版社，数量）

借书（书号，读者号，借出日期，应还日期）

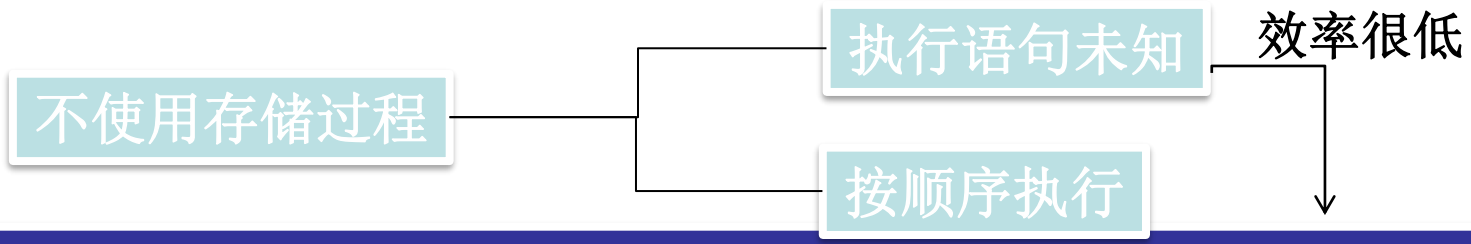
◆ 按每种书给一个编号，并记录每种书数量

第八章 数据库编程

SQL语言提供了两种不同的使用方式：

交互式 **&** 嵌入式

8.2.1 存储过程的概念、优点及分类



存储过程是一组为了完成特定功能的SQL语句集，是一种数据库的对象。

存储过程的优点：

模块化的程序设计，独立修改。

高效率的执行，一次编译。

减少网络流量。

可以作为安全机制使用。

创建存储过程的基本方法

```
CREATE [OR REPLACE] PROCEDURE 过程名 [([IN | OUT | IN OUT]参数名 数据类型, …)]
{AS |IS}
[LabelName
[DECLARE]
 [说明部分]
BEGIN
        语句序列
    [EXCEPTION 出错处理]
END [LabelName ];
```

注意：{IS | AS} 后面是一个完整的DMSQL块，可以定义变量、游标等。

说明：

① 过程名和参数名必须符合标识符命名规则。

② OR REPLACE是一个可选的关键字，建议用户使用此关键字，当数据库中已经存在此过程名，则该过程会被重新定义，并被替换。

③ 关键字IS和AS本身没有区别，选择其中一个即可。

④ IS后面是一个完整的DMSQL程序块，可以定义变量、游标等。

★创建存储过程时，可以定义零个或多个形式参数。

形式参数可以有三种模式----IN、OUT、IN OUT。

如没有为形式参数指定模式，那么缺省的模式是IN。

类型	描述
IN参数（默认模式）	输入参数，用来从调用环境中（实参）向存储过程传递值，在过程体内不能给IN参数赋值或修改；实参可以是常量或变量
OUT参数	输出参数，用来从存储过程中返回值给调用者（实参），在过程体内必须给OUT参数赋值；实参是变量
IN OUT参数	输入输出参数，既可以从调用者（实参）向存储过程中（形参）传递值，也可以从过程中返回可能改变的形参值给调用者（实参）；实参是变量

【例】 创建一个无参数的存储过程，输出当前系统的日期。

```
CREATE OR REPLACE PROCEDURE
out_date()
IS
BEGIN
PRINT  '当前系统日期为:'||SYSDATE;
END ;
```

```
CALL out_date();
```

【例】 创建一个名称为InsertRecord的存储过程，该存储过程的功能是向S数据表中插入一条记录，新记录的值由参数提供。

```
CREATE  OR  REPLACE  PROCEDURE  InsertRecord
(v_sno IN S.SNO%TYPE,
 v_sn IN  S.SN%TYPE ,
 v_sex IN S.SG%TYPE,
 v_SD IN S.SD%TYPE,
 v_sp IN S.SP%TYPE)
AS
BEGIN
INSERT  INTO  S  VALUES(v_sno,v_sn, v_sex, v_SD,
v_SP);
END;
```

```
CALL InsertRecord('S7','张三','男','2005-12-25','计算机');
```

【例】 定义具有参数默认值的存储过程。创建一个名称为InsertRecordDefa的存储过程，该存储过程的功能是向S数据表中插入一条记录，新记录的值由参数提供，如果未提供系别SP的值时，由参数的默认值代替。

```
CREATE OR REPLACE PROCEDURE InsertRecordDefa
(v_sno IN S.SNO%TYPE,
 v_sn IN  S.SN%TYPE ,
 v_sex IN S.SG%TYPE,
 v_SD IN S.SD%TYPE,
 v_sp IN S.SP%TYPE :='无')
AS
BEGIN
INSERT INTO S VALUES(v_sno,v_sn, v_sex, v_SD, v_SP);
END;


CALL InsertRecordDefa ('S8','张五','男','2002-12-25');
```

触发器（TRIGGER）

触发器是某个数据库操作发生时被自动调用的函数。

可以在INSERT、UPDATE或DELETE操作之前或之后调用触发器。

两种类型的触发器，一种是数据行级触发器，另外一种是语句级触发器。

对于数据行级的触发器，触发发触发器的语句每操作一个数据行，它就被执行一次。对于语句级的触发器，它只会被执行一次。

创建触发器

- **CREATE TRIGGER语法格式**

CREATE TRIGGER <触发器名>

{BEFORE | AFTER} <触发事件> ON <表名>

referencing OLD ROW AS "OLD" NEW ROW AS "NEW"

FOR EACH {ROW | STATEMENT}

[WHEN <触发条件>]<触发动作体>

- 触发器又叫做事件-条件-动作（**event-condition-action**）规则。
- 当特定的系统事件发生时，对规则的条件进行检查，如果条件成立则执行规则中的动作，否则不执行该动作。规则中的动作体可以很复杂，通常是一段**SQL**存储过程。

【例】创建一个语句级触发器，当执行删除S中信息操作后，输出提示信息“执行了删除操作…”。

```
CREATE OR REPLACE TRIGGER
delete_trigger1
AFTER DELETE ON S
BEGIN
PRINT '您执行了删除操作…';
END;
```

【例】创建一个语句级触发器，当执行删除S中信息操作后，输出提示信息“执行了删除操作…”。

```
CREATE OR REPLACE TRIGGER DELETE_TRIGGER1
AFTER DELETE  ON  S
FOR EACH STATEMENT
   BEGIN
   PRINT '执行了删除操作…';
  END;
```

触发器相关的特殊变量NEW和OLD

- 如果是行级触发器，用户都可以在过程体中使用NEW和OLD引用事件之后的新值和事件之前的旧值

【例】当对表**SC**的**Grade**进行修改时，若分数增加了**10%**则将此次操作记录到表**SC_U**（**Sno,Cno,Oldgrade,Newgrade**）中;其中:**Oldgrade**是修改前的分数，**Newgrade**是修改后的分数。

```
CREATE  OR  REPLACE  TRIGGER   SC_T    AFTER
UPDATE  OF  Grade  ON SC      FOR EACH ROW
WHEN (New.Grade >= 1.1*Old.Grade)
BEGIN
INSERT INTO SC_U(Sno,Cno,OldGrade,NewGrade)
VALUES(Old.Sno,Old.Cno,Old.Grade,New.Grade);
END;
```

游标（cursor）

游标是系统为用户开设的一个数据缓冲区，存放SQL语句的执行结果

为什么要使用游标

SQL语言是面向集合的，一条SQL语句原则上可以产生或处理多条记录

主语言是面向记录的，一组主变量一次只能存放一条记录

使用显式游标包括四个步骤

第十章 数据库恢复技术

- 事务(Transaction)定义

  一个数据库操作序列

  一个不可分割的工作单位

  恢复和并发控制的基本单位

- 事务和程序比较

  在关系数据库中，一个事务可以是一条或多条SQL语句,也可以包含一个或多个程序。

  一个程序通常包含多个事务

日志文件的格式和内容

- 什么是日志文件

  日志文件(log)是用来记录事务对数据库的更新操作的文件

- 日志文件的格式
  - 以记录为单位的日志文件
  - 以数据块为单位的日志文件

登记日志文件

必须先写日志文件，后写数据库

写日志文件操作：把表示这个修改的日志记录写到日志文件

写数据库操作：把对数据的修改写到数据库中

- 为什么要先写日志文件
  - 写数据库和写日志文件是两个不同的操作；
  - 在这两个操作之间可能发生故障；
  - 如果先写了数据库修改，而在日志文件中没有登记下这个修改，则以后就无法恢复这个修改了；
  - 如果先写日志，但没有修改数据库，按日志文件恢复时只不过是多执行一次不必要的UNDO操作，并不会影响数据库的正确性。

事务的特性(ACID特性)

事务的ACID特性：

- 原子性（Atomicity）
- 一致性（Consistency）
- 隔离性（Isolation）
- 持续性（Durability ）

习题

1. 试述事务的概念及事务的4 个特性。

原子性(Atomicity)：事务是数据库的逻辑工作单位，事务中包括的诸操作要么都做，要么都不做。

一致性(consistency)：事务执行的结果必须是使数据库从一个一致性状态变到另一个一致性状态。

隔离性(Isolation) ：一个事务的执行不能被其他事务干扰。即一个事务内部的操作及使用的数据对其他并发事务是隔离的，并发执行的各个事务之间不能互相干扰。

持续性(Durability) ：持续性也称永久性（Perfnanence ），指一个事务一旦提交，它对数据库中数据的改变就应该是永久性的。接下来的其他操作或故障不应该对其执行结果有任何影响。

例：银行DB有转帐事务:从帐号A转50元到帐号B

(1) 原子性：

由事务的原子性保持事务的一致性，但事务的执行有一定的时间，在某一个时刻会不一致，是正常的

(2) 一致性：

A-50，B+50　A+B的和不变

(3) 隔离性：

在A-50后，突然插入一个事务来计算A+B,那肯定会不对，这就要由DBMS的并发控制来控制。

(4) 持久性：

事务正确执行后，仍长期保存，不能丢失。

故障的种类

事务内部的故障：非预期的

系统故障：指造成系统停止运转的任何事件，使得系统要重新启动。整个系统的正常运行突然被破坏；所有正在运行的事务都非正常终止

介质故障：系统在运行过程中，由于某种硬件故障，使存储在外存上的数据部分损失或全部损失。

计算机病毒：一种可以繁殖和传播的人为的故障或破坏的计算机程序。

系统故障

称为软故障，是指造成系统停止运转的任何事件，使得系统要重新启动。

- 整个系统的正常运行突然被破坏
- 所有正在运行的事务都非正常终止
- 不破坏数据库
- 内存中数据库缓冲区的信息全部丢失

系统故障的常见原因：

- 特定类型的硬件错误（如CPU故障）
- 操作系统故障
- DBMS代码错误
- 系统断电

系统故障的恢复

- 发生系统故障时，事务未提交
  - 恢复策略：强行撤消（UNDO）所有未完成事务
- 发生系统故障时，事务已提交，但缓冲区中的信息尚未完全写回到磁盘上。
  - 恢复策略：重做（REDO）所有已提交的事务

介质故障的恢复

1.重装数据库：装入数据库发生介质故障前某个时刻的数据副本

2.重做已完成的事务：重做自此时始的所有成功事务，将这些事务已提交的结果重新记入数据库

数据转储方法

1. 静态转储与动态转储
2. 海量转储与增量转储

习题

3．数据库中为什么要有恢复子系统？它的功能是什么？

答：因为计算机系统中硬件的故障、软件的错误、操作员的失误以及恶意的破坏是不可避免的，这些故障轻则造成运行事务非正常中断，影响数据库中数据的正确性，重则破坏数据库，使数据库中全部或部分数据丢失，因此必须要有恢复子系统。

恢复子系统的功能是：把数据库从错误状态恢复到某一已知的正确状态（亦称为一致状态或完整状态），实现事务的持久性。

4.考虑下图所示的日志记录:

序号	日志
1	$T1$:开始
2	$T1$:写 A,$A=10$
3	$T2$:开始
4	$T2$:写 B,$B=9$
5	$T1$:写 C,$C=11$
6	$T1$:提交
7	$T2$:写 C,$C=13$
8	$T3$:开始
9	$T3$:写 A,$A=8$
10	$T2$:回滚
11	$T3$:写 B,$B=7$
12	$T4$:开始
13	$T3$:提交
14	$T4$:写 C,$C=12$

①如果系统故障发生在14之后,说明哪些事务需要重做,哪些事务需要回滚;

②如果系统故障发生在10之后,说明哪些事务需要重做,哪些事务需要回滚;

③如果系统故障发生在9之后,说明哪些事务需要重做,哪些事务需要回滚;

④如果系统故障发生在7之后,说明哪些事务需要重做,哪些事务需要回滚。

第十一章 并发控制

并发控制要解决的根本问题是保持数据库状态的一致性。

- 并发操作带来的数据不一致性

  丢失修改（**Lost Update**）

  不可重复读（**Non-repeatable Read**）

  读“脏”数据（**Dirty Read**）

- 记号 **R(x):读数据x**

  W(x):写数据x

什么是封锁

- 封锁就是事务T在对某个数据对象（例如表、记录等）操作之前，先向系统发出请求，对其加锁
- 加锁后事务T就对该数据对象有了一定的控制，在事务T释放它的锁之前，其它的事务不能更新此数据对象。

封锁类型

基本封锁类型

排它锁（**Exclusive Locks，简记为X锁**）

共享锁（**Share Locks，简记为S锁**）

常用意向锁

意向共享锁(Intent Share Lock，简称IS锁)

意向排它锁(Intent Exclusive Lock，简称IX锁)

共享意向排它锁(Share Intent Exclusive Lock，简称SIX锁)

死锁的诊断

超时法　　等待图法

- 在数据库中，产生死锁的原因是两个或多个事务都已封锁了一些数据对象，然后又都请求已被其他事务封锁的数据加锁，从而出现死等待。
- 防止死锁的发生其实就是要破坏产生死锁的条件。预防死锁通常有两种方法：

1）一次封锁法：要求每个事务必须一次将所有要使用的数据全部加锁，否则就不能继续执行；

2）顺序封锁法：预先对数据对象规定一个封锁顺序，所有事务都按这个顺序实行封锁。

可串行化调度

- **可串行化(Serializable)调度**

  多个事务的并发执行是正确的，当且仅当其结果与按某一次序串行地执行这些事务时的结果相同

- **可串行性(Serializability)**

  是并发事务正确调度的准则

  一个给定的并发调度，当且仅当它是可串行化的，才认为是正确调度

[例11.3] 设有调度Sc1= $r_1(A)$ $w_1(A)$ $r_2(A)$ $w_2(A)$ $r_1(B)$ $w_1(B)$ $r_2(B)$ $w_2(B)$

可以把$w_2(A)$ 与$r_1(B)$ $w_1(B)$交换,得到:

$r_1(A)$ $w_1(A)$ $r_2(A)$ $r_1(B)$ $w_1(B)$ $w_2(A)$ $r_2(B)$ $w_2(B)$

再把$r_2(A)$ 与$r_1(B)$ $w_1(B)$交换,得到:

Sc2 =$r_1(A)$ $w_1(A)$ $r_1(B)$ $w_1(B)$ $r_2(A)$ $w_2(A)$ $r_2(B)$ $w_2(B)$

Sc2等价于一个串行调度T_1， T_2。所以Sc2是冲突可串行化的调度。

两段锁协议（续）

- 事务遵守两段锁协议是可串行化调度的充分条件，而不是必要条件。
- 若并发事务都遵守两段锁协议，则对这些事务的任何并发调度策略都是可串行化的
- 若并发事务的一个调度是可串行化的，不一定所有事务都符合两段锁协议

可串行化调度（续）

[例]现在有两个事务，分别包含下列操作：

- 事务T1：读B；A=B+1；写回A
- 事务T2：读A；B=A+1；写回B

现给出对这两个事务不同的调度策略

串行化调度,正确的调度

T_1	T_2
Slock B	
Y=R(B)=2	
Unlock B	
Xlock A	
A=Y+1=3	
W(A)	
Unlock A	
	Slock A
	X=R(A)=3
	Unlock A
	Xlock B
	B=X+1=4
	W(B)
	Unlock B

- 假设A、B的初值均为2。
- 按T1→T2次序执行结果为A=3，B=4
- 串行调度策略,正确的调度

串行调度(a)

串行化调度,正确的调度

T_1	T_2
	Slock A
	X=R(A)=2
	Unlock A
	Xlock B
	B=X+1=3
	W(B)
	Unlock B
Slock B	
Y=R(B)=3	
Unlock B	
Xlock A	
A=Y+1=4	
W(A)	
Unlock A	

串行调度(b)

- 假设A、B的初值均为2。
- **T2→T1次序执行结果为B=3，A=4**
- **串行调度策略,正确的调度**

不可串行化调度，错误的调度

T_1	T_2
Slock B	
Y=R(B)=2	
	Slock A
	X=R(A)=2
Unlock B	
	Unlock A
Xlock A	
A=Y+1=3	
W(A)	
	Xlock B
	B=X+1=3
	W(B)
Unlock A	
	Unlock B

- 执行结果与(a)、(b)的结果都不同
- 是错误的调度

不可串行化的调度

可串行化调度，正确的调度

T_1	T_2
Slock B	
Y=R(B)=2	
Unlock B	
Xlock A	
	Slock A
A=Y+1=3	等待
W(A)	等待
Unlock A	等待
	X=R(A)=3
	Unlock A
	Xlock B
	B=X+1=4
	W(B)
	Unlock B

- 执行结果与串行调度(a)的执行结果相同
- 是正确的调度

可串行化的调度

两段锁协议

两段锁协议是指所有事务必须分两个阶段对数据项加锁和解锁。

1)在对任何数据进行读、写操作之前，首先要申请并获得对该数据的封锁；

2)在释放一个封锁之后，事务不再申请和获得任何其他封锁。

“两段”的含义是，事务分为两个阶段：第一阶段是获得封锁，也称为扩展阶段。在这阶段，事务可以申请获得任何数据项上的任何类型的锁，但是不能释放任何锁。第二阶段是释放封锁，也称为收缩阶段。在这阶段，事务释放已经获得的锁，但是不能再申请任何锁。