

数据库复习

乔铭宇

2020/12/24

如何画E-R图

1 按照问题的描述一步一步找出每一句话中涉及的
实体

2 根据给出的实际语义，画出实体之间的联系

对实体之间联系的语义描述有时不是直截了当的——从对现实世界的整体描述中进行分析，导出实体之间的某种联系

3 例如：

✓ 问题——零件和仓库的联系是多对多联系？一对多联系？

✓ 描述——零件按所属的不同产品分别放在仓库中

一个产品由多种零件组成的 → 一个仓库中放多种零件

反过来一种零件是放在一个仓库还是多个仓库中呢？

一种零件可以用在多种产品上，这些零件按所属的不同产品分别放在仓库中 → 一种零件可以放在多个仓库中

✓ 结论——所以零件和仓库之间是多对多的联系

8)基本关系的性质

- ① 列是同质的（Homogeneous）：来自同一个域
- ② 不同的列可出自同一个域
 - 其中的每一列称为一个属性
 - 不同的属性要给予不同的属性名
- ③ 列的顺序无所谓，列的次序可以任意交换
- ④ 任意两个元组的候选码不能相同
- ⑤ 行的顺序无所谓，行的次序可以任意交换
- ⑥ 分量必须取原子值，即每个分量必须是不可再分的数据项。

已知关系R如下表所示，关系R的主属性为__(1)___，候选关键字分别为__(2)___。

R
A B C D
a b c d
a c d e
b d e f
a d c g
b c d g
c b e g

- (1) A. ABC B. ABD C. ACD D. ABCD
(2) A. ABC B. AB、AD C. AC、AD和CD D. AB、AD、BD和CD

答案: (1)D (2)D

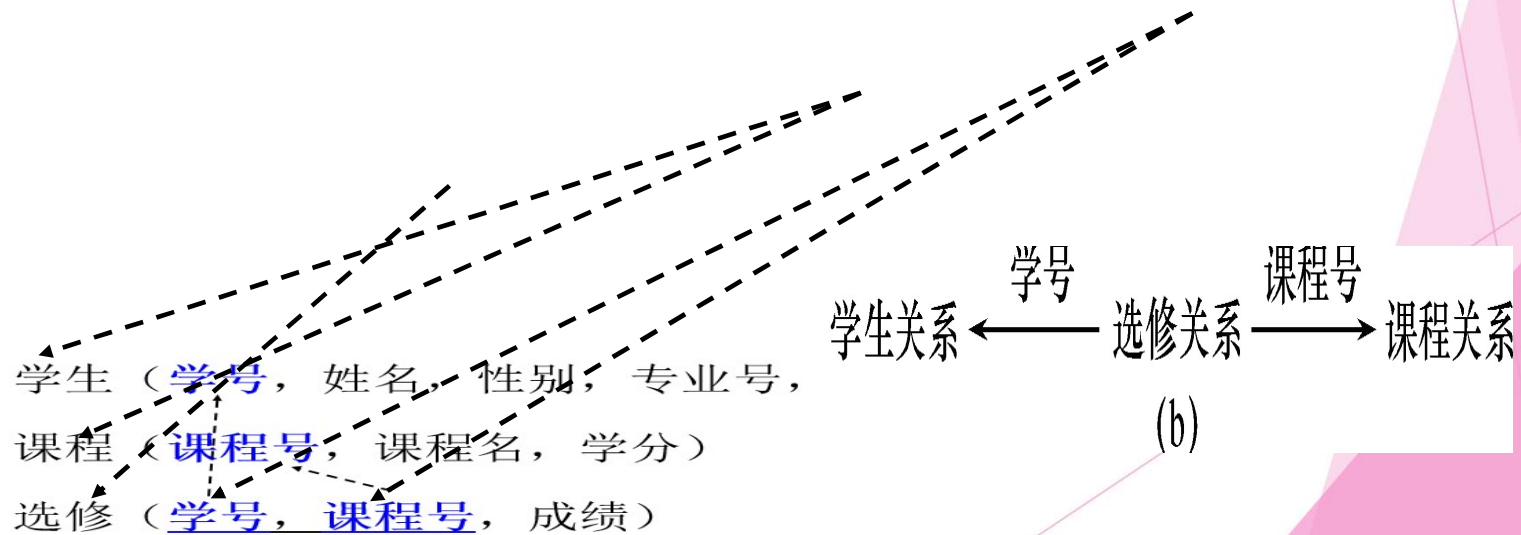
外码例

[例2]：

选修关系的“学号”与学生关系的主码“学号”相对应

选修关系的“课程号”与课程关系的主码“课程号”相对应

- ▶ “学号”和“课程号”是选修关系的外码
- ▶ 学生关系和课程关系均为被参照关系
- ▶ 选修关系为参照关系

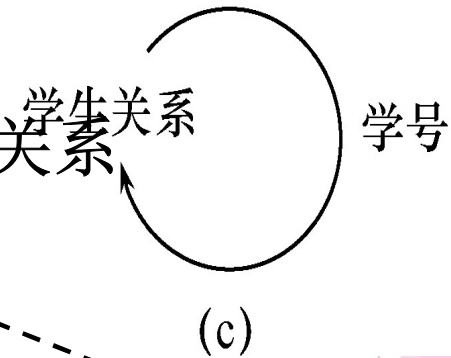


外码例

[例3]：“班长”与本身的主码“学号”相对应

▶ “班长”是外码

▶ 学生关系既是参照关系也是被参照关系



学 号	姓 名	性 别	专 业 号	年 龄	班 长
8 0 1	张 三	女	0 1	1 9	8 0 2
8 0 2	李 四	男	0 1	2 0	
8 0 3	王 五	男	0 1	2 0	8 0 2
8 0 4	赵 六	女	0 2	2 0	8 0 5
8 0 5	钱 七	男	0 2	1 9	

外码的性质

- * 关系 R 和 S 不一定是不同的关系
- * 目标关系 S 的主码 K_s 和参照关系的外码 F 必须定义在同一个（或一组）域上。
- * 外码并不一定要与相应的主码同名。
当外码与相应的主码属于不同关系时，往往取相同的名字，以便于识别。

学生关系

学号	姓名	所在系
9901	张三	计算机
9902	李四	计算机
9903	王五	计算机
9904	赵六	计算机

主码

课程关系

课程号	课程名	学时数
C01	高等数学	100
C02	数据结构	70
C03	操作系统	80
C04	数据库	60

主码

选修关系

学号	课程号	成绩
9901	C01	90
9901	C02	80
9902	C04	90
9904	C04	85
9904	C03	70

主属性+外码

主属性+外码

选修关系（学号，课程号，成绩）

“学号”和“课程号”可能的取值约束：

- (1) 选修关系中的主属性，不能取空值（“学号”和“课程号”）
- (2) 只能取相应被参照关系中已经存在的主码值

那比较好的是加外键呢？还是在应用中维护？

不必须，一般情况下都是不加外键的，外键靠应用中维护
因为表加了外键，在开发、测试和更新数据时非常麻烦。

那么在什么情况下适合加外键？什么情况不适合加外键？

回答

开发期间可以不加外键，上生产后可以加上外键（在数据库级保证数据完整性），前提是把数据的关系都搞对，当然，如果你的应用能保证数据的完整性，就可以不加外键。

[例5] 设关系 R 、 S 分别为下图的(a)和(b), $R \div S$ 的结果为图(c)

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

(a)

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

(b)

$R \div S$
A
a_1

(c)

关系代数小结

► 五种基本代数运算：

- 并、差、笛卡尔积、选择、投影

$$R \cup S, R - S, R \times S, \sigma_F(R), \pi_A(R)$$

象集 是 对一个关系而言
除 是 两个关系的运算

► 组合运算：

- 连接(一般连接、等值连接、自然连接、外连接)、除

$$R \bowtie S, R \div S$$

- 象集： 设 $R(X, Y)$ 则 $Y_{xI} = \pi_Y(\sigma_{X=xI}(R))$

- 除： 给定关系 $R(X, Y)$ 和 $S(Y, Z)$, 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 上投影的集合记作

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

4.以同一种语法结构提供多种使用方式

SQL是**独立的语言**，SQL又是**嵌入式语言**

5.语言简洁，易学易用

表 3.1 SQL 语言的动词

SQL 功 能	动 词
数 据 查 询	SELECT
数 据 定 义	CREATE, DROP, ALTER
数 据 操 纵	INSERT, UPDATE DELETE
数 据 控 制	GRANT, REVOKE

三级模式结构：外模式、模式和内模式

一、模式 (Schema)

定义：也称逻辑模式，是数据库中全体数据的逻辑结构和特征的描述，是所有用户的公共数据视图。

理解：

- ① 一个数据库只有一个模式；
- ② 是数据库数据在逻辑级上的视图；
- ③ 数据库模式以某一种数据模型为基础；
- ④ 定义模式时不仅要定义数据的逻辑结构（如数据记录由哪些数据项构成，数据项的名字、类型、取值范围等），而且要定义与数据有关的安全性、完整性要求，定义这些数据之间的联系。

二、外模式 (External Schema)

定义：也称子模式 (Subschema) 或用户模式，是数据库用户（包括应用程序员和最终用户）能够看见和使用的局部数据的逻辑结构和特征的描述，是数据库用户的数据视图，是与某一应用有关的数据的逻辑表示。

理解：

- ① 一个数据库可以有多个外模式；
- ② 外模式就是用户视图；
- ③ 外模式是保证数据安全性的一个有力措施。

三、内模式 (Internal Schema)

定义：也称存储模式 (Storage Schema)，它是数据物理结构和存储方式的描述，是数据在数据库内部的表示方式（例如，记录的存储方式是顺序存储、按照B树结构存储还是按hash方法存储；索引按照什么方式组织；数据是否压缩存储，是否加密；数据的存储记录结构有何规定）。

理解：

- ① 一个数据库只有一个内模式；
- ② 一个表可能由多个文件组成，如：数据文件、索引文件。

它是数据库管理系统(DBMS)对数据库中数据进行有效组织和管理的方法

其目的有：

- ① 为了减少数据冗余，实现数据共享；
- ② 为了提高存取效率，改善性能。

► 基本表

- 本身独立存在的表
- SQL中一个关系就对应一个基本表
- 一个(或多个)基本表对应一个存储文件
- 一个表可以带若干索引

► 存储文件

- 存储文件的逻辑结构组成了关系数据库的内模式
- 物理结构是任意的，对用户透明

► 视图

- 从一个或几个基本表导出的表
- 数据库中只存放视图的定义而不存放视图对应的数据
- 视图是一个虚表
- 用户可以在视图上再定义视图

3.3 数据定义

SQL的数据定义功能: 模式定义、表定义、视图和索引的定义

表 3.2 SQL 的数据定义语句

操 作 对 象	操 作 方 式		
	创 建	删 除	修 改
模式	CREATE SCHEMA	DROP SCHEMA	
表	CREATE TABLE	DROP TABLE	ALTER TABLE
视 图	CREATE VIEW	DROP VIEW	
索 引	CREATE INDEX	DROP INDEX	

注：视图和索引无修改语句！

数据类型

数据类型	含义
CHAR(n)	长度为n的定长字符串
VARCHAR(n)	最大长度为n的变长字符串
INT	长整数（也可以写作INTEGER）
SMALLINT	短整数
NUMERIC(p, d)	定点数，由p位数字（不包括符号、小数点）组成，小数后面有d位数字
REAL	取决于机器精度的浮点数
Double Precision	取决于机器精度的双精度浮点数
FLOAT(n)	浮点数，精度至少为n位数字
DATE	日期，包含年、月、日，格式为YYYY-MM-DD
TIME	时间，包含一日的时、分、秒，格式为HH:MM:SS

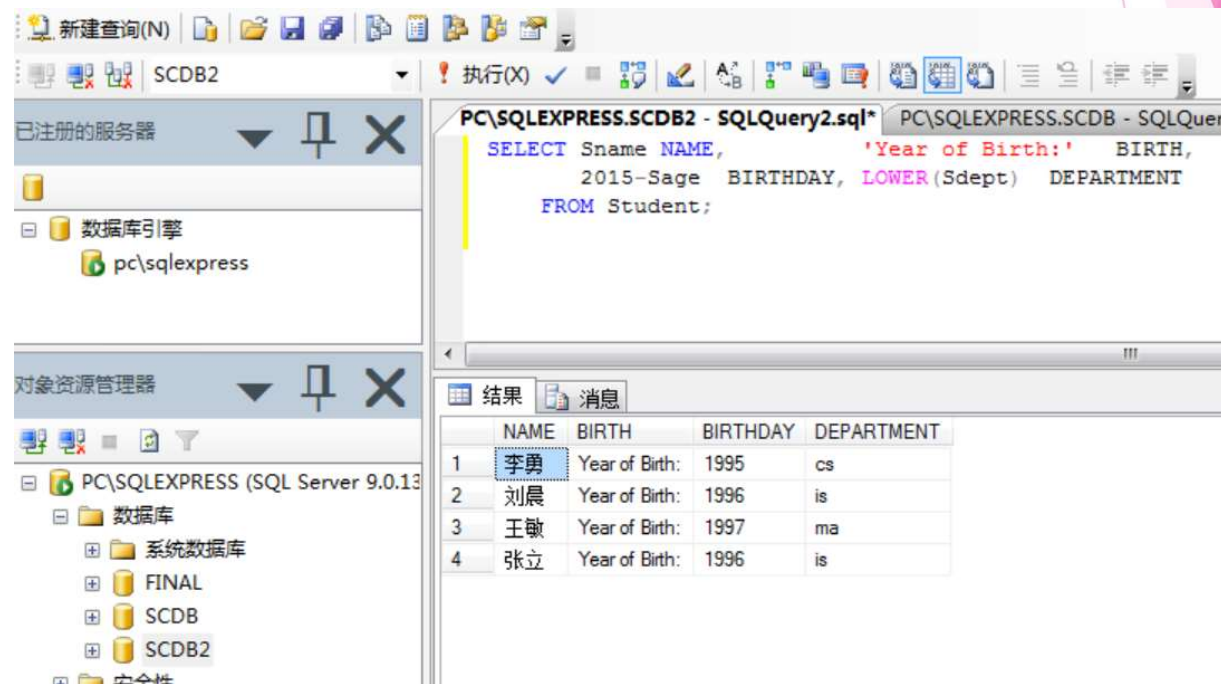
SQL语言简易格式

- ▶ **SELECT** <列名> : 投影
- ▶ **FROM** <表名> : 连接
- ▶ **WHERE** <条件> : 选择
- ▶ **GROUP BY** <列名> : 分组
- HAVING** <条件> : 满足条件的分组
- ▶ **ORDER BY** <列名> : 排序

使用列别名改变查询结果的列标题

```
SELECT Sname NAME, 'Year of Birth:' BIRTH,  
2020-Sage BIRTHDAY, LOWER(Sdept) DEPARTMENT  
FROM Student;
```

输出结果:



The screenshot shows the Microsoft SQL Server Enterprise Manager interface. The left pane displays the '对象资源管理器' (Object Explorer) for the 'PC\SQLEXPRESS (SQL Server 9.0.13)' instance, showing the '数据库' (Databases) folder expanded. The right pane shows the 'SQLQuery2.sql' file with the following query:

```
SELECT Sname NAME, 'Year of Birth:' BIRTH,  
2020-Sage BIRTHDAY, LOWER(Sdept) DEPARTMENT  
FROM Student;
```

Below the query editor, the '结果' (Results) tab is active, displaying the output of the query as a table with 4 rows and 4 columns: NAME, BIRTH, BIRTHDAY, and DEPARTMENT.

	NAME	BIRTH	BIRTHDAY	DEPARTMENT
1	李勇	Year of Birth:	1995	cs
2	刘晨	Year of Birth:	1996	is
3	王敏	Year of Birth:	1997	ma
4	张立	Year of Birth:	1996	is

2. 查询满足条件的元组

- | | |
|---|--|
| <ul style="list-style-type: none">• 查询条件 | <ul style="list-style-type: none">• 谓词 |
| <ul style="list-style-type: none">• 比较 | <ul style="list-style-type: none">• =,>,<,>=,<=,!<,!>,!=;<>,!<,>;NOT+上述比较运算符 |
| <ul style="list-style-type: none">• 确定范围 | <ul style="list-style-type: none">• BETWEEN AND, NOT BETWEEN AND |
| <ul style="list-style-type: none">• 确定集合 | <ul style="list-style-type: none">• IN, NOT IN |
| <ul style="list-style-type: none">• 字符匹配 | <ul style="list-style-type: none">• LIKE, NOT LIKE |
| <ul style="list-style-type: none">• 空值 | <ul style="list-style-type: none">• IS NULL, IS NOT NULL |
| <ul style="list-style-type: none">• 多重条件
(逻辑运算) | <ul style="list-style-type: none">• AND, OR, NOT |

b. 匹配串为含通配符的字符串

%（百分号）代表任意长度字符串

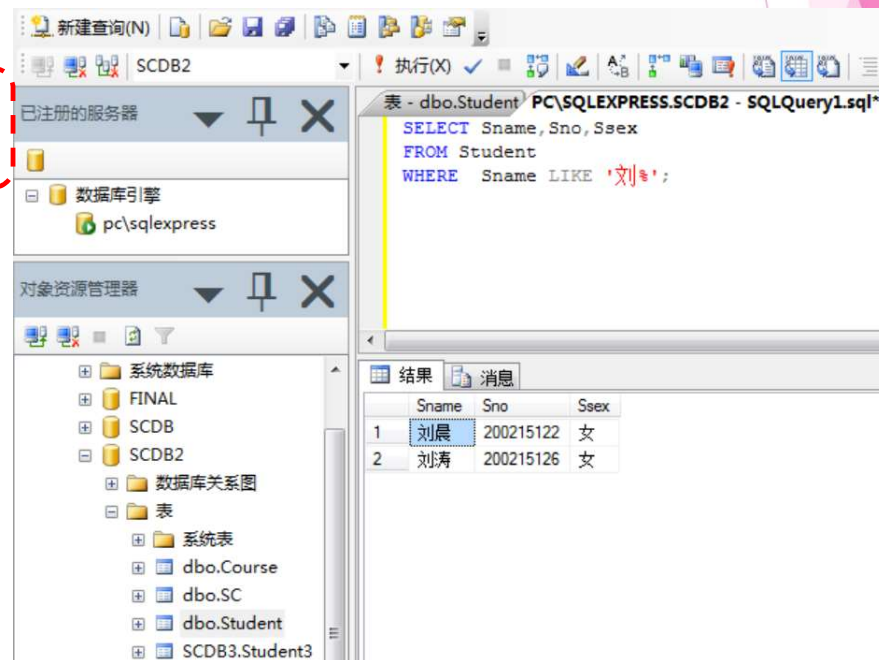
_（下横线）代表任意单个字符

[例15] 查询所有姓刘学生的姓名、学号和性别。

```
SELECT Sname, Sno, Ssex
```

```
FROM Student
```

```
WHERE Sname LIKE '刘%';
```

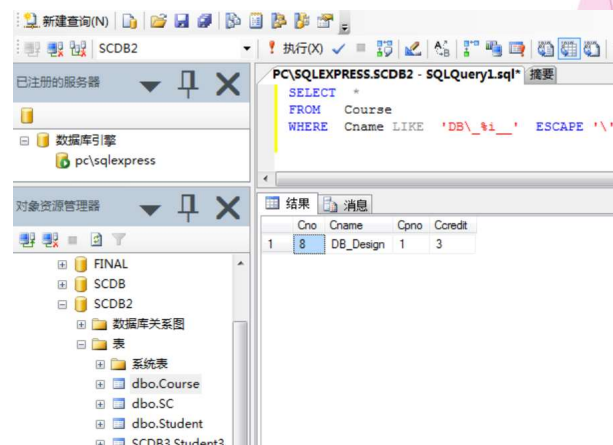


c. 将通配符转义为普通字符

► **ESCAPE '\'** 表示 “ \ ” 为**换码字符**, 这样匹配串中紧跟在\后面的字符 “_” 不再具有通配符的含义。

[例20] 查询以“DB_”开头, 且**倒数第3个字符为 i**的课程的具体情况。

```
SELECT *  
FROM Course  
WHERE Cname LIKE 'DB\__%i__' ESCAPE '\';
```



(5) 涉及空值的查询

* 谓词: **IS NULL** 或 **IS NOT NULL**

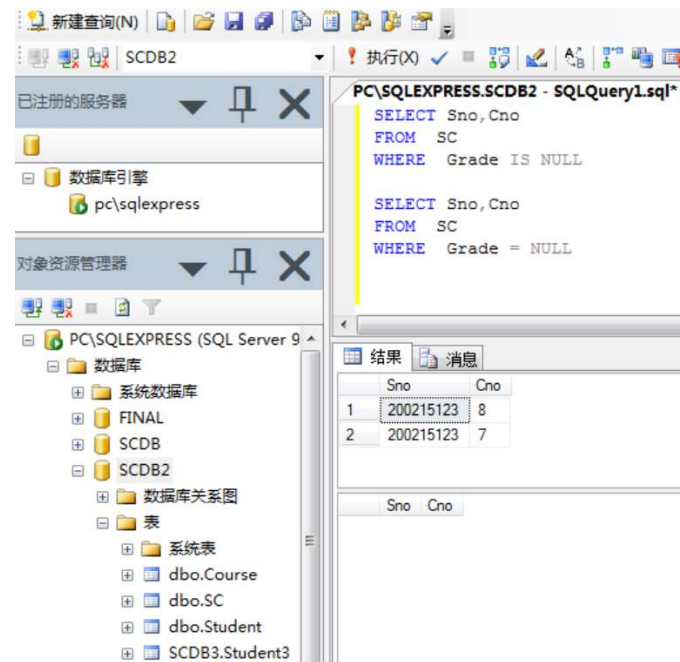
* “IS” 不能用 “=” 代替

[例21] 某些学生选修课程后没有参加考试,所以有选课记录,但没有考试成绩。查询缺少成绩的学生的学号和相应的课程号。

```
SELECT Sno, Cno
```

```
FROM SC
```

```
WHERE Grade IS NULL
```



[例25] 查询全体学生情况,查询结果按所在系的系号升序排列,同一系中的学生按年龄降序排列。

```
SELECT *  
FROM Student  
ORDER BY Sdept, Sage DESC;
```

The screenshot displays the SQL Server Enterprise Manager interface. The left pane shows the '对象资源管理器' (Object Explorer) with the '数据库' (Databases) folder expanded, showing 'SCDB2'. The right pane shows the 'SQLQuery1.sql' file with the following query:

```
/*查询全体学生情况,查询结果按所在系的系号升序排列,同一系中的学生按年龄降序排列。*/  
SELECT *  
FROM Student  
ORDER BY Sdept, Sage DESC;
```

Below the query editor, the '结果' (Results) pane shows the output of the query as a table with 8 rows and 6 columns: Sno, Sname, Ssex, Sage, Sdept. The data is sorted by Sdept (CS, IS, MA, MT, PT) and then by Sage (descending).

	Sno	Sname	Ssex	Sage	Sdept
1	200215126	刘涛	女	21	CS
2	200215121	李勇	男	20	CS
3	200215122	刘晨	女	19	IS
4	200215125	张立	男	19	IS
5	200215123	王敏	女	18	MA
6	200215129	欧阳娜娜	女	24	MT
7	200215128	何炅	男	23	PT
8	200215127	谢娜	女	22	PT

查询学生总人数

```
SELECT COUNT(*)  
FROM Student;
```

查询选修了课程的学生人数

```
SELECT COUNT(DISTINCT Sno)  
FROM SC;
```

计算1号课程的学生平均成绩

```
SELECT AVG(Grade)  
FROM SC  
WHERE Cno= '1';
```

查询选修1号课程的学生最高分数

```
SELECT MAX(Grade)  
FROM SC  
WHERE Cno= '1';
```



- 如果**分组后**还要求按一定的**条件**对这些组进行筛选,最终只输出满足指定条件的组,则可以使用**HAVING**短语指定筛选条件。

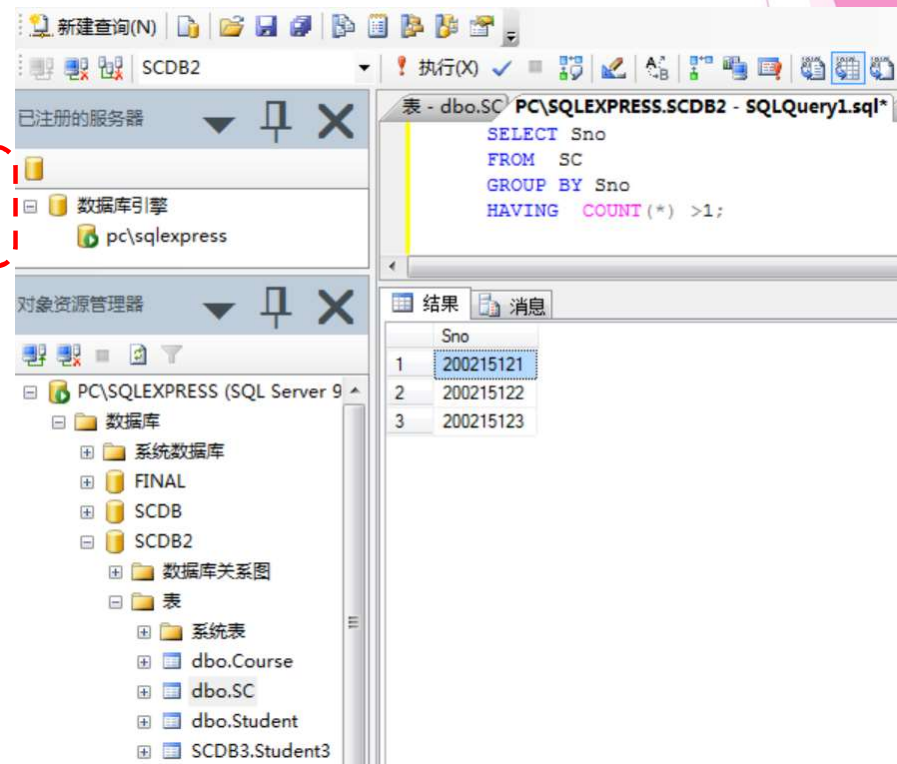
[例32] 查询选修了**1门以上课程**的学生学号。

SELECT Sno

FROM SC

GROUP BY Sno

HAVING COUNT(*) > 1;



- ▶ 如果**分组后**还要求按一定的**条件**对这些组进行筛选,最终只输出满足指定条件的组,则可以使用**HAVING**短语指定筛选条件。

✱ **HAVING**短语与**WHERE**子句的**区别**:

作用对象不同

- **WHERE**子句作用于**基表或视图**,从中选择满足条件的元组
- **HAVING**短语作用于**组**,从中选择满足条件的组

3.4.3 嵌套查询

▶ 嵌套查询概述

- ▶ 一个SELECT-FROM-WHERE语句称为一个查询块
- ▶ 将一个查询块嵌套在另一个查询块的WHERE子句或HAVING短语的条件中的查询称为嵌套查询

* 子查询的限制

- 不能使用ORDER BY子句（只能对最终结果排序）
- * 层层嵌套方式反映了 SQL语言的结构化
- * 有些嵌套查询可以用连接运算替代

► ANY(或SOME)、ALL与聚集函数、IN谓词的等价转换关系

	=	<> 或 !=	<	<=	>	>=
ANY	IN	--	<MAX	<=MAX	>MIN	>= MIN
ALL	--	NOT IN	<MIN	<= MIN	>MAX	>= MAX

四、带有EXISTS谓词的子查询

1. EXISTS谓词

- 存在量词 \exists
- 带有EXISTS谓词子查询不返回任何数据, 只产生逻辑真值“true”或逻辑假值“false”。
 - ▶ 若内层查询结果非空, 则外层的WHERE子句返回真值
 - ▶ 若内层查询结果为空, 则外层的WHERE子句返回假值
- 由EXISTS引出的子查询, 其目标列表表达式通常都用select *, 因为带EXISTS的子查询只返回真值或假值, 给出列名无实际意义。

2. NOT EXISTS谓词

- 若内层查询结果非空, 则外层的WHERE子句返回假值
- 若内层查询结果为空, 则外层的WHERE子句返回真值

▶ 不同形式的查询间的替换

- 一些带EXISTS或NOT EXISTS谓词的子查询不能被其他形式的子查询等价替换
- 所有带IN谓词、比较运算符、ANY和ALL谓词的子查询都能用带EXISTS谓词的子查询等价替换

[例5] 将学生200215121的年龄改为22岁

```
UPDATE Student  
SET Sage=22  
WHERE Sno=' 200215121 ';
```

[例6] 将所有学生的年龄增加1岁

```
UPDATE Student  
SET Sage= Sage+1;
```

*行列子集视图

▶ 若一个视图是从单个基本表导出的，并且只是去掉了基本表的某些行和某些列，但保留了主码，我们称这类视图为行列子集视图。

- ▶ 只要行列子集视图可以update
- ▶ WHERE子句中是不能用聚集函数作为条件表达式的

► 基于视图的视图

[例4] 建立计算机系选修了1号课程且成绩在90分以上的学生的视图

```
CREATE VIEW IS_S2  
AS  
SELECT Sno, Sname, Grade  
FROM CS_S1  
WHERE Grade>=90;
```

* 带表达式的视图

[例5] 定义一个反映学生出生年份的视图。

```
CREATE VIEW BT_S(Sno, Sname, Sbirth)  
AS  
SELECT Sno, Sname, 2016-Sage  
FROM Student;
```

► 更新视图的限制

一些视图是不可更新的，因为对这些视图的更新不能唯一地有意义地转换成对相应基本表的更新

例：视图S_G为不可更新视图。

```
UPDATE S_G
SET     Gavg=90
WHERE  Sno= '200215121';
```

这个对视图的更新无法转换成对基本表SC的更新

```
CREATE VIEW S_G (Sno, Gavg)
AS
SELECT Sno, AVG(Grade)
FROM SC
GROUP BY Sno;
```

✱ 视图作用

- 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
- 5. 适当的利用视图可以更清晰的表达查询

可能存在的问题

▶ 一个“**不好**”的关系模式可能存在的问题：

- ▶ 数据冗余（Data Redundancy）
- ▶ 插入异常（Insertion Anomalies）
- ▶ 删除异常（Deletion Anomalies）
- ▶ 更新异常（Update Anomalies）

* 结论：

- XSCJ关系模式不是一个好的模式。
- “好”的模式：不会发生插入异常、删除异常、更新异常，数据冗余应尽可能少。

* 原因：关系模式中存在不合适的数据依赖。

解决方法

- ▶ 解决方法：进行规范化处理，消除不合适的数据依赖-----模式分解

XSCJ(Sno , SName , Sdept , Director , Cno, Cname , Cscore)

- ▶ 把这个单一模式分成4个关系模式：

- ▶ Student (Sno, Sname, Sdept)
- ▶ Course (Cno, Cname)
- ▶ SC (Sno, Cno, Grade)
- ▶ DEPT (Sdept, Director)

三、什么是函数依赖

▶ 函数依赖的定义：

- ▶ 函数依赖是指属性之间的这种约束关系：一个或一组属性的值可以决定其他属性的值。
- ▶ 一般地，若 X 、 Y 是关系模式中两个不同的属性（组），如果 Y 函数依赖于 X ，或说 X 函数决定 Y ，则其依赖关系可表示为： $X \rightarrow Y$ 。

▶ 举例

- ▶ 如描述一个学生的关系，有学号、姓名、性别、年龄等属性，一个学号唯一对应一个学生，因此当“学号”的值确定之后，学生的姓名以及性别、年龄等属性的值也就唯一确定了
- ▶ 我们称之为学号函数决定姓名，或者说姓名函数依赖于学号
- ▶ 表示为：学号 \rightarrow 姓名，学号 \rightarrow 性别，学号 \rightarrow 年龄

4、传递函数依赖

定义6.3:

在 $R(U)$ 中, 如果 $X \rightarrow Y$, $(Y \subseteq X), Y \rightarrow X, Y \rightarrow Z$, 则
称 Z 对 X 传递函数依赖。

记为: $X \xrightarrow{\text{传递}} Z$

注: 如果 $Y \rightarrow X$, 即 $X \leftrightarrow Y$, 则 Z 直接依赖于 X 。

例: 在关系模式XSCJ中,

XSCJ(Sno , SName , Sdept , Director , Cno , Cname , Cscore)

存在: $Sno \rightarrow Sdept, Sdept \rightarrow Director$

所以Director传递函数依赖于Sno

5、候选码

定义6.4

设 K 为 $R\langle U, F \rangle$ 中的属性或属性组合。若 $K \xrightarrow{F} U$ ，则 K 称为 R 的候选码（Candidate Key）。

例如

设 $R(A, B, C, D, E)$ ，存在这样的函数依赖集合：

$F = \{AB \rightarrow CDE, E \rightarrow ABCD\}$ ，

该关系模式的候选码为：

AB, E 。

码(复习)

▶ 候选码 (Candidate key)

若关系中的某一属性组的值能唯一地标识一个元组, 则称该属性组为候选码

▶ 全码 (All-key)

关系模式的所有属性组是这个关系模式的候选码, 称为全码

▶ 主码 (Primary key)

若一个关系有多个候选码, 则选定其中一个为主码

▶ 主属性

候选码的每一个属性称为主属性

3NF的定义

► 定义1

- 若关系模式 $R \in 2NF$ ，而且它的任何一个非主属性都不传递依赖于 R 的任何候选码，则称 $R \in 3NF$ 。

► 定义2

- 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 ($Z \subseteq Y$)，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。

BCNF的定义

- ▶ 定义：关系模式 $R\langle U, F \rangle \in 1NF$ ，若 $X \rightarrow Y$ 且 $Y \not\subseteq X$ 时 X 必含有码，则 $R\langle U, F \rangle \in BCNF$ 。
- ▶ 等价于：每一个决定因素都包含码。

* 若 $R \in BCNF$

- 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

$$\begin{array}{ccc} * R \in BCNF & \xrightleftharpoons[\text{不必要}]{\text{充分}} & R \in 3NF \end{array}$$

- * $STC(\underline{Sno}, \underline{Tno}, \underline{Cno})$ 被分解为
 $ST(\underline{Sno}, \underline{Tno})$
 $TC(\underline{Tno}, \underline{Cno})$

插入：学生未选课不能插入教师课程信息
删除：选修课程学生全部毕业，删除之后会删除相应课程信息
修改：修改某个课程号会涉及到多行学号。

BCNF的性质

► 如果 $R \in 3NF$ ，且 R 只有一个候选码，则：

$$R \in BCNF$$

$$R \in BCNF \xrightleftharpoons[\text{必要}]{\text{充分}} R \in 3NF$$

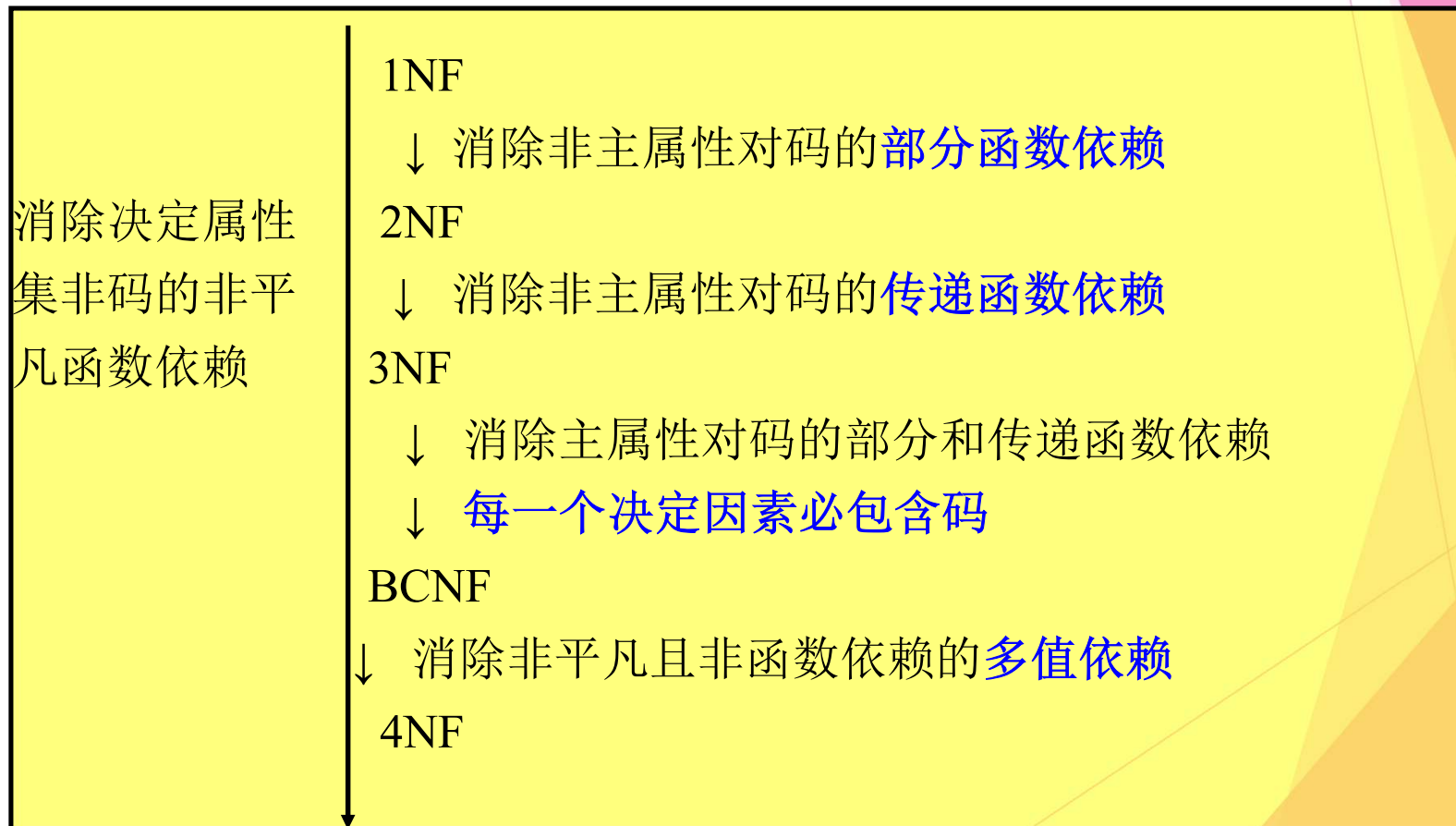
✱ 如果一个关系数据库中的所有关系模式都属于BCNF，那么在函数依赖范畴内，它已实现了模式的彻底分解，达到了最高的规范化程度。

规范化小结

- ▶ 关系数据库的规范化理论是数据库逻辑设计的工具。
- ▶ 规范化的基本思想是逐步消除数据依赖中不合适的部分，使模式中的各关系模式达到某种程度的“分离”，**让一个关系描述一个概念、一个实体或者实体间的一种联系**。若多于一个概念就把它“分离”出去。因此所谓规范化实质上是概念的**单一化**。

规范化小结

► 关系模式规范化的基本步骤



规范化小结

- ▶ 不能说规范化程度越高的关系模式就越好！

有时故意保留部分冗余可能更方便数据查询。尤其对于那些更新频度不高，查询频度极高的数据库系统更是如此。比如：

S-L (Sno, Sdept, Sloc)

(1) S-D (Sno, Sdept)

(2) D-L (Sdept, Sloc)

- ▶ 在设计数据库模式结构时，必须对现实世界的实际情况和用户应用需求作进一步分析，确定一个合适的、能够反映现实世界的模式

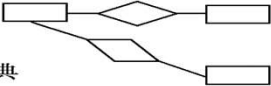
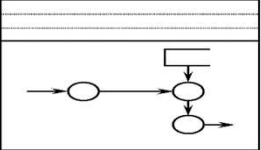

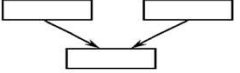
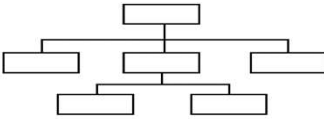

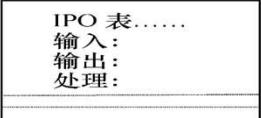
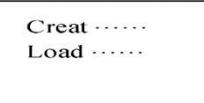
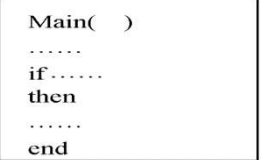
上面的规范化步骤可以在其中任何一步终止。

- ▶ 通常在实际工程应用中，每个关系模式达到3NF即可满足要求。

7.1.3 数据库设计的基本步骤

- ▶ 数据库设计分6个阶段
 - ▶ 需求分析
 - ▶ 概念结构设计
 - ▶ 逻辑结构设计
 - ▶ 物理结构设计
 - ▶ 数据库实施：数据库的建立、试运行、评价
 - ▶ 数据库运行和维护
- ▶ 需求分析和概念设计独立于数据库管理系统
- ▶ 逻辑设计和物理设计与选用的DBMS密切相关

数据库设计各个阶段的设计描述

设计阶段	设计描述	
	数 据	处 理
需求分析	数据字典、全系统中数据项、数据流、数据存储的描述	数据流图和判定表（判定树）、数据字典中处理过程的描述
概念结构设计	概念模型（E-R图）  数据字典	系统说明书包括： ① 新系统要求、方案和概图 ② 反映新系统信息流的数据流图 
逻辑结构设计	某种数据模型 关系  非关系 	系统结构图 （模块结构） 
物理设计	存储安排 方法选择 存取路径建立 	模块设计 IPO 表 
数据库实施阶段	编写模式 装入数据 数据库试运行 	程序编码、编译联结、测试 
数据库运行和维护	性能监测、转储 / 恢复 数据库重组和重构	新旧系统转换、运行、维护（修正性、适应性、改善性维护）

对基本E-R图的要求

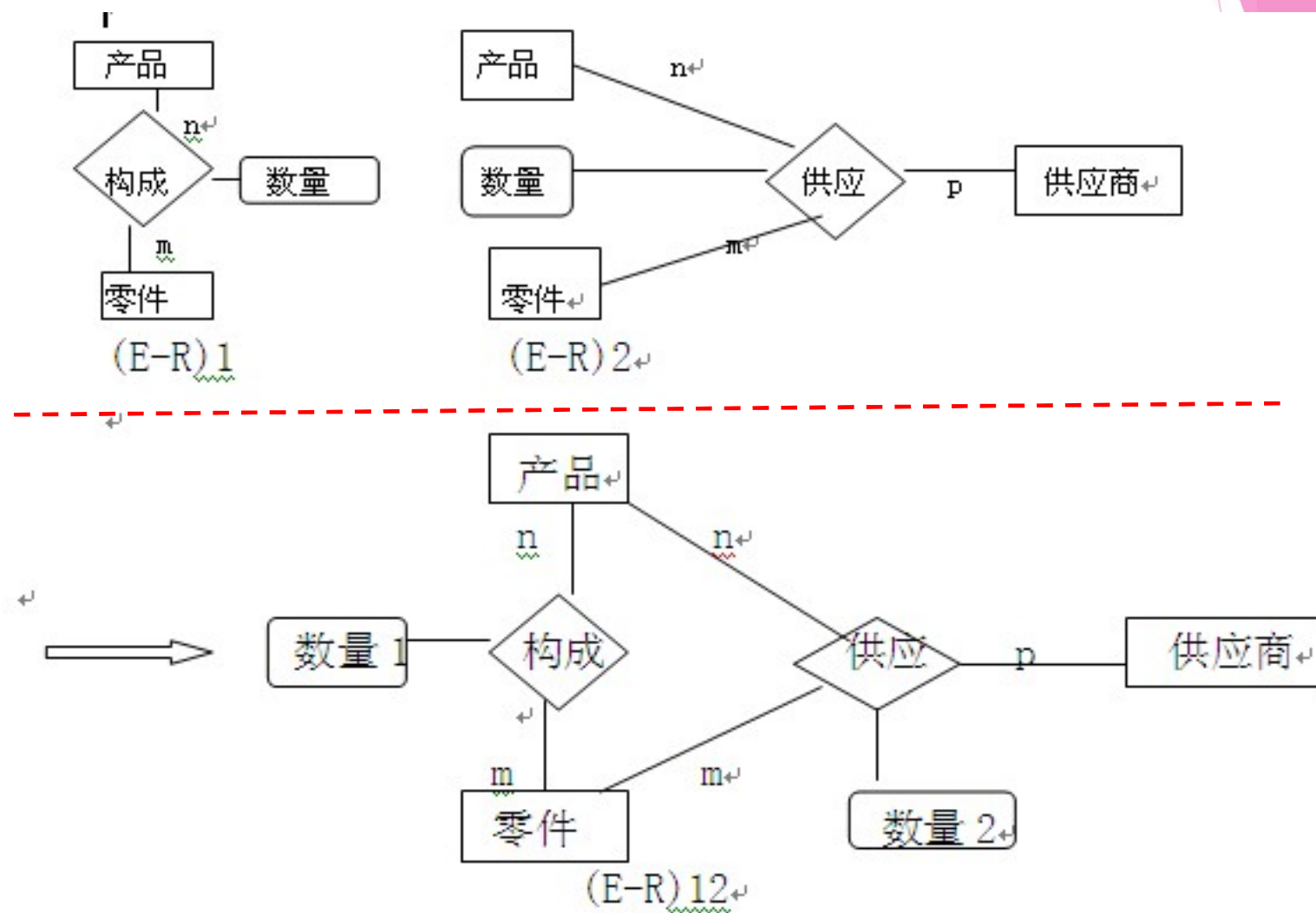
一个好的E-R模式，除了能够准确、全面的反映用户需求之外，还应该达到下列要求（系统越简单，越容易维护）：

- ◆ 实体类型的个数应尽量少；
- ◆ 实体类型所含属性个数应尽可能少；
- ◆ 实体类型间的联系应无冗余。

7.3.4 视图的集成

- ▶ 各个局部视图即分E-R图建立好后，还需要对它们进行合并，集成为一个整体的数据概念结构即总E-R图。
- ▶ 集成视图一般采取逐步累积的方式，首先集成比较关键的两个局部E-R图，然后逐步将新的视图集成进来。一般按以下过程完成。
- ✱ 集成局部E-R图的步骤
 1. 合并
 2. 修改与重构
 3. 验证整体概念结构

合并两个分E-R图时的综合



(1) 1:1联系的转换规则:

可以转换为一个独立的关系模式，也可以与任意一端对应的关系模式合并。

- ▶ 如果转换为一个独立的关系模式，则与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性，每个实体的码均是该关系的候选码。

例：部门:领导 = 1:1 Deptment (DID, Dname, address)

系与系主任 Director (DirID, Tel)

(DID, DirID)

- ▶ 如果与某一端实体对应的关系模式合并，则需要在该关系模式的属性中加入另一个关系模式的码和联系本身的属性

Deptment (DID, Dname, address)

Director (DirID, Tel, DID)



(2) 1:n联系的转换规则

可以转换为一个独立的关系模式，也可以与n端对应的关系模式合并。

- 如果转换为一个独立的关系模式，则与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性，而关系的码为n端实体的码。
- 如果与N端实体对应的关系模式合并，则需要在该关系模式的属性中加入另一个关系模式的码和联系本身的属性。
- 例： 部门:职工=1:n (职工号, 职工名, 部门号)

(3) m:n联系的转换规则

- 转换为为一个独立的关系模式。与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性，而关系的码为各实体码的组合
- ▶ 例： 零件:仓库 = m:n

(4)三个或三个以上实体间的一个多元联系

- 转换为一个关系模式。与该多元联系相连的各实体的码以及联系本身的属性均转换为关系的属性，关系的码为各实体码的组合。
- 例：（产品号，供应商号，零件号）

(5)具有相同码的关系模式可以合并

- ▶ 目的：减少系统中的关系个数
- ▶ 合并方法：将其中一个关系模式的全部属性加入到另一个关系模式中，然后去掉其中的同义属性（可能同名也可能不同名），并适当调整属性的次序

7.4.2 数据模型的优化

5. 按照需求分析阶段得到的各种应用对数据处理的要求，对关系模式进行必要的分解，以提高数据操作的效率和存储空间的利用率

► 常用分解方法

- 水平分解:把(基本)关系的（常用）元组分为若干子集合，定义每个子集合为一个子关系，以提高系统的效率。
- 垂直分解:把关系模式R的（常用）属性分解为若干子集合，形成若干子关系模式。

二、聚簇存取方法的选择

► 聚簇

- 为了提高某个属性（或属性组）的查询速度，把这个或这些属性（称为聚簇码）上具有相同值的元组集中存放在连续的物理块称为聚簇。

* 聚簇的用途

1. 大大提高按聚簇码进行查询的效率
2. 节省存储空间

- 聚簇以后，聚簇码相同的元组集中在一起了，因而聚簇码值不必在每个元组中重复存储，只要在一组中存一次就行了

确定数据库的存储结构

► 确定数据库物理结构的内容

1. 确定数据的存放位置和存储结构

- 关系
- 索引
- 聚簇
- 日志
- 备份

2. 确定系统配置

数据库设计各阶段的成果

数据库的各产出是在设计过程中逐步形成的

- ▶ 需求分析阶段综合各个用户的应用需求（现实世界的需求），形成数据字典。
- ▶ 概念设计阶段形成独立于机器特点、独立于各个DBMS产品的概念模式（信息世界模型），用E-R图来描述。

数据库设计各阶段的成果

- ▶ 在**逻辑设计阶段**将E-R图转换成具体的数据库产品支持的数据模型如关系模型，形成数据库逻辑**模式**。然后根据用户处理的要求，安全性的考虑，在基本表的基础上再建立必要的视图（VIEW）形成数据的**外模式**。
- ▶ 在**物理设计阶段**根据DBMS特点和处理的需要，进行物理存储安排，设计索引，形成**数据库内模式**。

事务的基本概念

所谓事务是用户定义的一个数据库操作序列，这些操作要么都做，要么都不做，是一个不可分割的工作单位(并且需要正确执行)。

在关系数据库中，事务可以是一条SQL语句、一组SQL语句。

小 结

- ▶ 恢复中最经常使用的技术
 - ▶ 备份
 - ▶ 登记日志文件
- ▶ 恢复的基本原理
 - ▶ 利用存储在**后备副本**、**日志文件**和**数据库镜像**中的冗余数据来重建数据库
- ▶ 常用恢复技术
 - ▶ **事务故障**的恢复：UNDO
 - ▶ **系统故障**的恢复：REDO+UNDO
 - ▶ **介质故障**的恢复：REDO
 - ▶ **磁盘镜像**、**磁盘阵列**

结 论

- ✓ 在操作系统中广为采用的预防死锁的策略并不很适合数据库的特点；
- ✓ DBMS在解决死锁的问题上更普遍采用的是检测并解除死锁的方法。

数据独立性是指应用程序和数据结构之间相互独立, 互不影响。

数据独立性包括数据逻辑独立性和数据物理独立性。

数据模型三要素: 数据结构, 数据操作, 完整性约束

ER图转换关系模型规则: E是实体, R是联系。

关系模型中的关系模式数量等于ER图中的实体数量+n:m联系的数量。而一对一联系和一对多联系均可隐藏在实体对应的关系中。

由ER图中n:m联系所转换的关系模式中, 其关系属性由该联系属性和该联系所对应的实体的主码共同组成。

数据库管理系统 (DBMS) 的基本功能包括:

- 1) 数据库定义功能;
- 2) 数据操作功能;
- 3) 数据库的运行管理和控制功能;
- 4) 数据库的建立和维护功能;
- 5) 数据库存取。

在关系模型中，一个关系就是一个二维表，二维表名就是关系名。二维表的列称为属性，二维表的行称为元组。表中的第一行通常称为属性名，表中的每一个元组都是不可再分的。元组的次序是无关紧要的。

试述数据库的三级模式结构。这种结构有什么优点。



并发控制可以保证事务的一致性和隔离性。

[例7] 设关系 R 、 S 分别如下：

R:

A	B	C
a	b	c
b	b	f
c	a	d
a	a	d
a	c	f
b	d	a

S:

B	C	D
b	c	d
b	c	e
a	d	b
c	f	g

计算：

(1) $R \times S$ (2) $R \bowtie S$

(3) $R \Join S$ (4) $R \div S$ (5) R 与 S 的外连接



(1)

练习

设关系R、S分别如下：

R:			S:		
A	B	C	B	C	D
a	b	c	b	c	d
b	b	f	b	c	e
c	a	d	a	d	b
a	a	d	c	f	g
a	c	f			
b	d	a			

计算：

- (1) $R \times S$ (2) $R \bowtie_{R.A=S.B} S$
(3) $R \Join S$ (4) $R \div S$ (5) R与S的外连接

R.A	R.B	R.C	S.B	S.C	S.D
a	b	c	b	c	d
a	b	c	b	c	e
a	b	c	a	d	b
a	b	c	c	f	g
b	b	f	b	c	d
b	b	f	b	c	e
b	b	f	a	d	b
b	b	b	c	f	g
...

(2)

练习

设关系R、S分别如下：

R:

A	B	C
a	b	c
b	b	f
c	a	d
a	a	d
a	c	f
b	d	a

S:

B	C	D
b	c	d
b	c	e
a	d	b
c	f	g

计算：

(1) $R \times S$ (2) $R \bowtie_{R.A=S.B} S$

(3) $R \Join S$ (4) $R \div S$ (5) R与S的外连接

R.A	B	C	S.B	C	D
a	b	c	a	d	b
a	a	d	a	d	b
a	c	f	a	d	b
b	b	f	b	c	d
b	b	f	b	c	e
b	d	a	b	c	d
b	d	a	b	c	e
c	a	d	c	f	g

练习

(3)

设关系R、S分别如下：

R:

A	B	C
a	b	c
b	b	f
c	a	d
a	a	d
a	c	f
b	d	a

S:

B	C	D
b	c	d
b	c	e
a	d	b
c	f	g

计算：

(1) $R \times S$ (2) $R \bowtie_{R.A=S.B} S$

(3) $R \bowtie S$ (4) $R \div S$ (5) R与S的外连接

A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b
a	a	d	b
a	c	f	g

(4)

练习

设关系R、S分别如下：

R:

A	B	C
a	b	c
b	b	f
c	a	d
a	a	d
a	c	f
b	d	a

S:

B	C	D
b	c	d
b	c	e
a	d	b
c	f	g

计算：

(1) $R \times S$ (2) $R \bowtie_{R.A=S.B} S$

(3) $R \Join S$ (4) $R \div S$ (5) R与S的外连接

$R \div S$

A

a

练习

(5)

设关系R、S分别如下：

R:

A	B	C
a	b	c
b	b	f
c	a	d
a	a	d
a	c	f
b	d	a

S:

B	C	D
b	c	d
b	c	e
a	d	b
c	f	g

计算：

(1) $R \times S$ (2) $R \bowtie_{R.A=S.B} S$

(3) $R \Join S$ (4) $R \div S$ (5) R与S的外连接

A	B	C	D
a	b	c	d
a	b	c	e
b	b	f	NULL
c	a	d	b
a	a	d	b
a	c	f	g
b	d	a	NULL

Student 学生-课程数据库: SCDB

学号 Sno	姓名 Sname	性别 Ssex	年龄 Sage	所在系 Sdept
200215121	李勇	男	20	CS
200215122	刘晨	女	19	IS
200215123	王敏	女	18	MA
200215125	张立	男	19	IS

Course				SC		
课程号 Cno	课程名 Cname	先行课 Cpno	学分 Ccredit	学号 Sno	课程号 Cno	成绩 Grade
1	数据库	5	4			
2	数学		2	200215121	1	92
3	信息系统	1	4	200215121	2	85
4	操作系统	6	3	200215121	3	88
5	数据结构	7	4			
6	数据处理		2	200215122	2	90
7	PASCAL语言	6	4	200215122	3	80

[例14]

查询至少选修了2号和3号课程的学生姓名

- ✓ 包含关系
- ✓ 首先建立一个临时关系K:

Cno

2

3

从哪查
怎么查
查什么

[例15] 查询选修了全部课程的学生号码和姓名。

Student 学生-课程数据库: SCDB

从哪查
怎么查
查什么

学号 <u>Sno</u>	姓名 <u>Sname</u>	性别 <u>Ssex</u>	年龄 <u>Sage</u>	所在系 <u>Sdept</u>
200215121	李勇	男	20	CS
200215122	刘晨	女	19	IS
200215123	王敏	女	18	MA
200215125	张立	男	19	IS

Course

课程号 <u>Cno</u>	课程名 <u>Cname</u>	先行课 <u>Cpno</u>	学分 <u>Ccredit</u>
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	PASCAL 语言	6	4

SC

学号 <u>Sno</u>	课程号 <u>Cno</u>	成绩 <u>Grade</u>
200215121	1	92
200215121	2	85
200215121	3	88
200215122	2	90
200215122	3	80

[例15] 查询选修了全部课程的学生号码和姓名。

$$\pi_{Sno, Cno}(SC) \div \pi_{Cno}(Course) \bowtie \pi_{Sno, Sname}(Student)$$

Student 学生-课程数据库: SCDB				
学号 <u>Sno</u>	姓名 <u>Sname</u>	性别 <u>Ssex</u>	年龄 <u>Sage</u>	所在系 <u>Sdept</u>
200215121	李勇	男	20	CS
200215122	刘晨	女	19	IS
200215123	王敏	女	18	MA
200215125	张立	男	19	IS

Course				SC		
课程号 <u>Cno</u>	课程名 <u>Cname</u>	先行课 <u>Cpno</u>	学分 <u>Ccredit</u>	学号 <u>Sno</u>	课程号 <u>Cno</u>	成绩 <u>Grade</u>
1	数据库	5	4			
2	数学		2	200215121	1	92
3	信息系统	1	4	200215121	2	85
4	操作系统	6	3	200215121	3	88
5	数据结构	7	4			
6	数据处理		2	200215122	2	90
7	PASCAL语言	6	4	200215122	3	80

$\pi_{Cno}(Course)$ 表示全部课程的课号
 $\pi_{Sno, Cno}(SC)$ 表示所有学生的选课情况,
 $\pi_{Sno, Cno}(SC) \div \pi_{Cno}(Course)$ 表示选择了所有课程的学生学号。
这个结果和 $\pi_{Sno, Sname}(Student)$ 作连接就可以得到学生的学号和姓名

对于至少这样的包含关系查询通常用除法来完成。