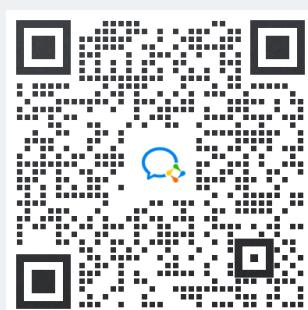


数据库 · 突击课

3.5小时突击

课程讲义

干货福利, 互助答疑



蜂考独家编辑, 版权所有



江苏博事达律师事务所

JIANGSU BOOMSTAR LAW OFFICE

中国 南京 奥体大街 68 号国际研发总部园 4A 栋 17 楼 邮编: 210019

17F 4ABuilding NO. 68 Aoti Street, Nanjing, China P.C: 210000

电话(Tel): (86)-25-82226685 传真(Fax): (86)-25-82226696

律师声明

江苏博事达律师事务所接受蜂考品牌的委托,发表以下律师声明:

“蜂考系列课程”(含视频、讲义、音频等)内容均为蜂考原创,蜂考品牌公司对此依法享有著作权,任何单位或个人不得侵犯。

蜂考品牌公司为维护创作作品的合法权益,已与江苏博事达律师事务所开展长期法律顾问合作,凡侵犯课程版权等知识产权的,蜂考品牌公司将授权江苏博事达律师事务所依据国家法律法规提起民事诉讼。对严重的侵权盗版分子将报送公安部门采取刑事手段予以严厉打击。

感谢大家对蜂考品牌的长期支持,愿与各位携手共同维护知识产权保护。遵守国家法律法规,从自身做起,抵制盗版!

特此声明!

江苏博事达律师事务所
二〇二一年七月十四日



目录

课时一 数据库系统绪论.....	1
1. 数据模型	1
2. 数据库系统的三级模式结构.....	2
3. 数据库的二级映像功能与数据独立性.....	3
课时一 练习题	6
课时二 关系代数	7
1. 传统集合运算	7
2. 专门的关系运算.....	9
课时二 练习题	17
课时三 数据定义与数据更新.....	20
1. 数据定义	20
2. 数据更新	24
课时三 练习题	29
课时四 数据查询	31
1. 单表查询	31
2. 连接查询	38
3. 嵌套查询	40
4. 集合查询	43
课时四 练习题	44
课时五 视图	45
1. 定义视图	45
2. 查询视图	48
3. 更新视图	51
课时五 练习题	53
课时六 数据库安全性.....	54
1. 自助存取控制方法.....	54
2. 授权与回收	54
3. 视图机制	59
课时六 练习题	61
课时七 数据库完整性.....	62
1. 实体完整性	62
2. 参照完整性	64
3. 用户定义的完整性.....	66
4. 触发器	71
课时七 练习题	74
课时八 关系数据理论.....	75
1. 函数依赖	75
2. 码	75
课时九 数据库设计	83
1. 数据库设计概述	83
2. 概念结构设计	86
3. 逻辑结构设计	91

配套课程 习题答案



课时九 练习题	95
课时十 并发控制	96
1. 事务	96
2. 封锁和封锁协议	98
3. 活锁与死锁	99
4. 并发调度的可串行性	101
5. 两段锁协议	103
6. 锁的粒度	103
课时十 练习题	107

配套课程 习题答案



课时一 数据库系统绪论

考点	重要程度	占分	题型
1. 数据模型	★★	1~2	选择
2. 数据库系统的三级模式结构	★★★★	1~2	选择、填空
3. 数据库的二级映像功能与数据独立性	★★★★	1~2	选择、填空

1. 数据模型

数据模型，是数据特征的抽象，它从抽象层次上描述了系统的静态特征、动态行为和约束条件，为数据库系统的信息表示与操作提供一个抽象的框架。数据模型所描述的内容有三部分，分别是数据结构、数据操作和数据约束。

数据模型分为两类，第一类是概念模型，第二类是逻辑模型和物理模型。下面将逐一进行说明。

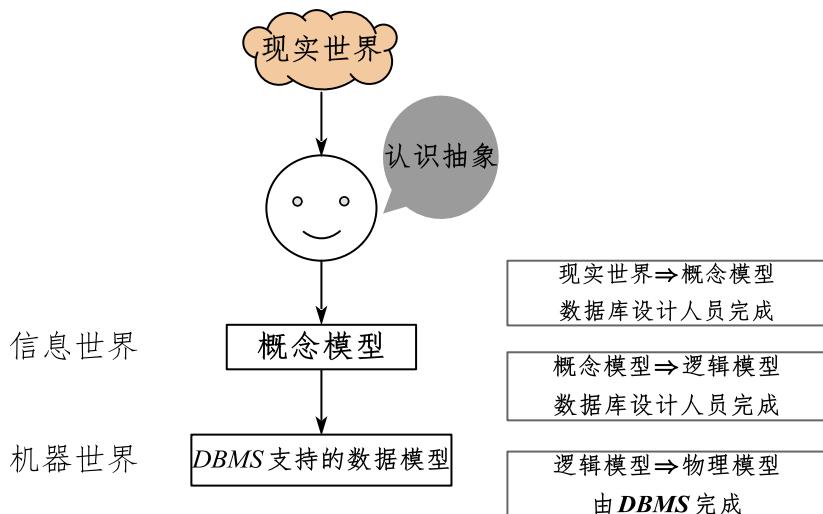
第一类，概念模型。概念模型用于信息世界的建模，是现实世界到信息世界的第一层抽象。首先把现实世界中的客观对象抽象为某一种信息结构，这种信息结构并不依赖于具体的计算机系统，不是某一个数据库管理系统（DBMS）支持的数据模型。

一种典型的表示方法：实体-联系方法，即E-R模型。

第二类，逻辑模型和物理模型。逻辑模型是将概念模型转化为具体的数据模型的过程，即按照概念结构设计阶段建立的基本E-R图，按选定的管理系统软件支持的数据模型（层次/网状/关系/面向对象），转换成相应的逻辑模型，这种转换要符合关系数据模型的原则。物理模型是对真实数据库的描述。

物理模型是在具体的物理介质上实现了对上述逻辑模型所述的内容。

示意图如下图所示。



配套课程 习题答案



现实世界中客观对象的抽象过程

题 1. 概念模型是现实世界的第一层抽象，这一类模型最著名的模型是（ ）

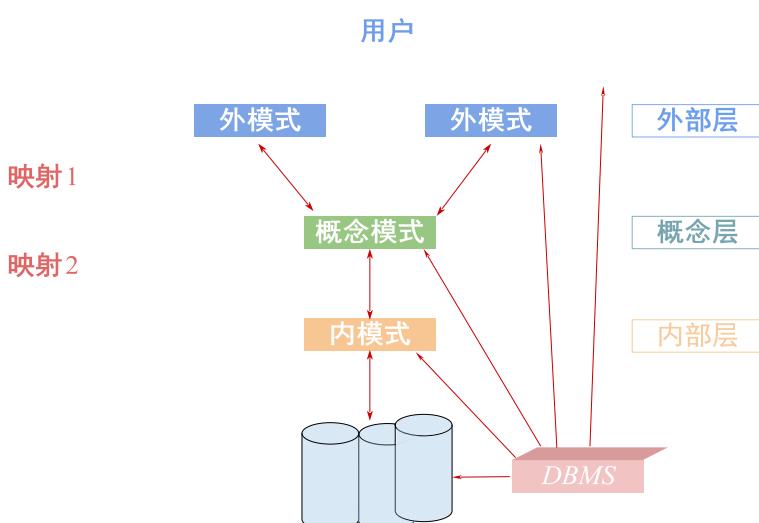
- A. 层次模型 B. 关系模型 C. 网状模型 D. 实体-联系模型

答案：D

解析：概念模型是对信息世界建模，所以概念模型能够方便的、准确的表示上述信息世界中的常用概念，最长是1976年提出的实体-联系模型，也称E-R模型。

2. 数据库系统的三级模式结构

为了有效地组织、管理数据库中的数据，提高数据库的逻辑独立性和物理独立性，人们为数据库设计了三级模式结构，即外模式（*External Schema*）、模式（*Schema*）和内模式（*Internal Schema*）。



模式：也称为逻辑模式（*Logic Schema*），它是由数据库设计者综合所有的数据需求，从全局的角度对数据库中全部数据的逻辑结构和特征的总体描述，是所有用户的公共数据视图即全局视图。

它是数据库模式结构的中间层，既不涉及数据的物理存储细节和硬件环境，也与具体的应用程序、所使用的应用开发工具及高级程序设计语言无关。

定义模式时不仅要定义数据的逻辑结构（如数据的型、数据类型、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。

外模式：也称为子模式（*Subschema*）或用户模式（*User Schema*），是程序员和最终用户能看见和使用的局部数据的逻辑结构和特征的描述，是与某一应用有关的数据的逻辑表示。



外模式通常是模式的子集。由于外模式是面向程序员和最终用户的，因此又称为用户数据视图。可以有多个外模式。另一方面，同一外模式也可以为某一用户的多个应用系统所使用，但一个应用程序只能使用一个外模式。

使用外模式可以带来几点好处：

①数据库设计者和用户不必关心整个数据库的数据，而只关心与自己的局部应用有关的一部分数据，并且不需了解它们的物理存储结构，这就使得程序设计和数据使用工作都得到了简化。

②用户只能操作与其有关的数据，而不能操作与其无关的数据，这有利于数据的安全保密。

③由于同一模式可以派生出多个外模式，故有利于数据独立性和数据共享。

内模式，又叫做存储模式（*Storage Schema*）或物理模式（*Physical Schema*），它是数据物理结构和存储方式的底层描述，包括记录的存储方式、索引组织方式、数据是否压缩和加密。

题 1. 对数据库物理存储方式的描述称为（ ）

- A. 外模式 B. 内模式 C. 概念模式 D. 逻辑模式

答案： B

解析： 内模式也称存储模式，一个数据库只有一个内模式。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。

题 2. 在数据库的三级结构中，描述数据库中的全体数据的全局逻辑结构和特征是（ ）。

- A. 外模式 B. 内模式 C. 存储模式 D. 模式

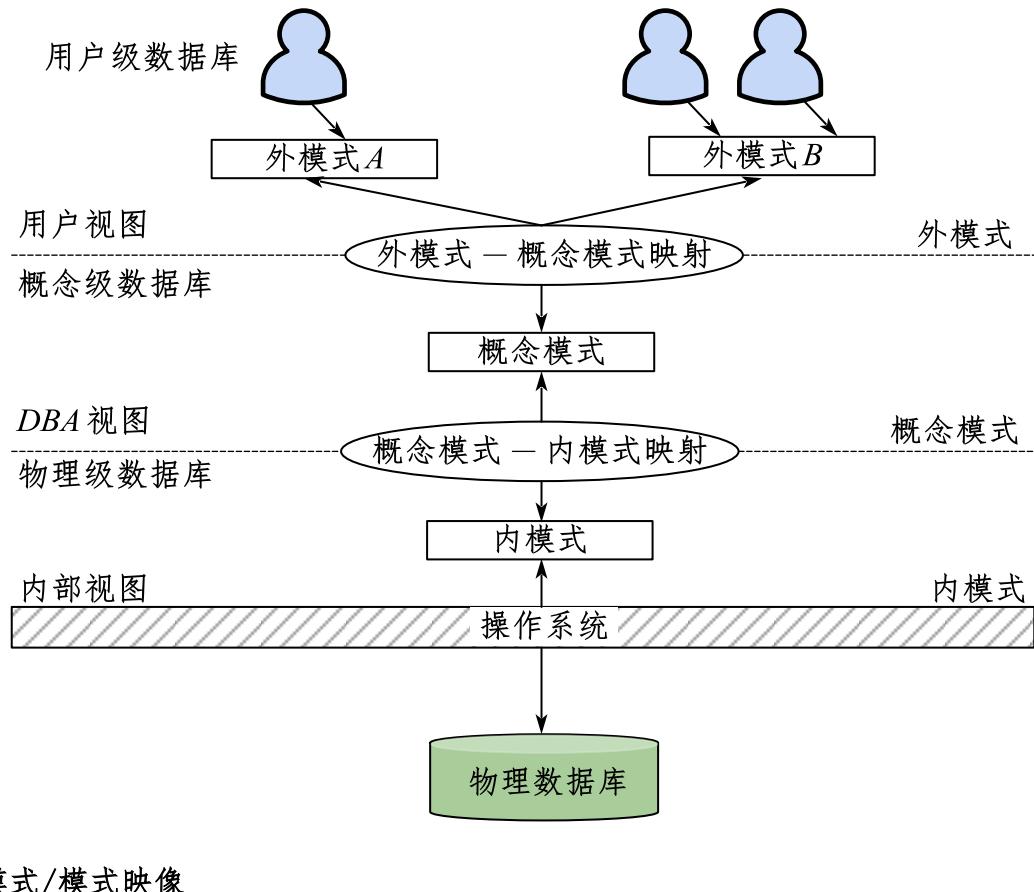
答案： D

解析： 模式（*Schema*）：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。

3. 数据库的二级映像功能与数据独立性



为了能够在内部数据库的三个抽象层次的联系和转换，数据库系统在这三级模式之间提供二级映像，即外模式/模式映像和模式/内模式映像。示意图如下。



外模式/模式映像

对每一个外模式，有一个外模式/模式映像，定义外模式与模式之间的对应关系，映像定义通常包含在各外模式的描述中，从而保证数据的逻辑独立性。

当模式改变时，数据库管理员对外模式/模式映像作相应改变，使外模式不变。应用程序是依据数据的外模式编写的，应用程序不必修改，保证了数据与程序的逻辑独立性，简称数据的逻辑独立性。

模式/内模式映像

定义了数据全局逻辑结构与存储结构之间的对应关系。数据库中模式/内模式映像是唯一的。该映像定义通常包含在模式描述中，从而保证数据的物理独立性。当数据库的存储结构改变了（例如选用了另一种存储结构），数据库管理员修改模式/内模式映像，使模式保持不变。

模式不变，则应用程序不变。保证了数据与程序的物理独立性，简称数据的物理独立性。



数据库的二级映像功能保证了应用程序的稳定性，除非应用需求本身发生变化，否则应用程序一般不需要修改。

以程序为中心发展为以数据为中心，具有了数据与程序之间的独立性，使得数据的定义和描述可以从应用程序中分离出去。

数据的存取由数据库管理系统管理，简化了应用程序的编制，大大减少了应用程序的维护和修改。

题 1. 在数据库三级模式间引入二级映像的主要作用是（ ）

- A. 提高数据与程序的独立性
- B. 提高数据与程序的安全性
- C. 保持数据与程序的一致性
- D. 提高数据与程序的可移植性

答案： A

解析：二级映像的主要作用是保证数据库系统中数据的独立性，即数据的物理组织改变与逻辑概念级改变相互独立，使得只要调整映射方式而不必改变用户模式。

题 2. 要保证数据库物理数据独立性，需要修改的是（ ）

- A. 模式
- B. 模式与内模式的映射
- C. 模式与外模式的映射
- D. 内模式

答案： B

解析：数据的物理独立性是指当数据的存储结构改变时，应用程序不必修改。要达到此目的，必须由数据库管理员对模式/内模式映像做相应的修改，而模式保持不变，从而使应用程序不必修改。

题 3. 数据独立性分为_____和_____。

答案：数据逻辑独立性、数据物理独立性

解析：1) 数据逻辑独立性：当模式改变时，由数据库管理员对各个外模式/模式映象作相应改变，能够使外模式维持不变。应用程序是依据数据的外模式编写的，从而应用程序没必要修改，保证了数据与程序的逻辑独立性，简称为数据逻辑独立性。

2) 数据物理独立性：当数据库的存储结构等内模式改变了，由数据库管理员对模式/内模式映象作相应改变，能够使模式维持不变，从而应用程序也没必要改变。保证了数据与程序的物理独立性，简称为数据物理独立性。



课时一 练习题

1. 外模式/模式映像可以保证数据和应用程序之间的_____。
2. 模式/内模式映像可以保证数据和应用程序之间的_____。
3. 在关系数据库中，表是三级模式结构中的（ ）
A. 外模式 B. 模式 C. 存储模式 D. 内模式
4. 要保证数据库的数据独立性，需要修改的是（ ）
*A. 三层模式之间的两种映射 B. 模式与内模式
C. 模式与外模式 D. 三层模式*
5. 在数据库技术中，*E-R*模型是一种（ ）
A. 概念数据模型 B. 结构数据模型 C. 物理数据模型 D. 逻辑数据模型
6. 数据的逻辑独立性是指（ ）
*A. 内模式改变，模式不变 B. 模式改变，内模式不变
C. 模式改变，外模式和应用程序不变 D. 内模式改变，外模式和应用程序不变*
7. 描述数据库全体数据的全局逻辑结构和特性的是（ ）
A. 模式 B. 内模式 C. 外模式 D. 模式和外模式
8. 三级模式存在两种映像，他们是（ ）
*A. 模式与子模式间，模式与内模式间
B. 子模式与内模式间，外模式与内模式
C. 子模式与外模式间，模式与内模式间
D. 模式与子模式间，模式与内模式间*



课时二 关系代数

考点	重要程度	占分	题型
1. 传统集合运算	★★★	1~2	选择
2. 专门的关系运算	★★★★	2~10	选择、简答

1. 传统集合运算

传统的集合运算是二目运算，包括并、差、交、笛卡尔积4种运算。

设关系 R 和关系 S 具有相同的目 n ，也就是两个关系中都有 n 个属性，且相应的属性取自同一个域， t 是元组变量， $t \in R$ 表示 t 是 R 的一个元组。

假定有 R 和 S 分别如下：

R			S		
A	B	C	A	B	C
a_1	b_1	c_1	a_1	b_2	c_2
a_1	b_2	c_2	a_1	b_3	c_2
a_2	b_2	c_1	a_2	b_2	c_1

(1) 并 (union)

关系 R 与关系 S 的并记作： $R \cup S = \{t | t \in R \vee t \notin S\}$ ，其结果仍为 n 目关系，由属于 R 而不属于 S 的元组组成。

例：

$R \cup S =$	A	B	C
	a_1	b_1	c_1
	a_1	b_2	c_2
	a_2	b_2	c_1
	a_1	b_3	c_2

(2) 差 (except)

关系 R 与关系 S 的差记作： $R - S = \{t | t \in R \wedge t \notin S\}$ ，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。

例：

$R - S =$	A	B	C
	a_1	b_1	c_1

配套课程 习题答案



(3) 交 (intersection)

关系 R 与关系 S 的交记作: $R \cap S = \{t | t \in R \wedge t \in S\}$, 其结果仍为 n 目关系, 由既属于 R 又属于 S 的元组组成, 关系的交还可以用差来表示: $R \cap S = R - (R - S)$ 。

例:

$R \cap S =$	A	B	C
	a_1	b_2	c_2
	a_2	b_2	c_1

(4) 笛卡尔积 (cartesianproduct)

笛卡尔积的元素是元组。两个分别为 n 目和 m 目的关系 R 和 S 的笛卡尔积是一个 $(n+m)$ 列的元组的集合, 若 R 有 K_1 个元组, S 有 K_2 个元组, 则关系 R 和关系 S 的笛卡尔积有 $K_1 \times K_2$ 个元组, 记作: $R \times S = \{t_r t_s | t_r \in R \wedge t_s \in S\}$ 。

例:

$R \times S =$	$R.A$	$R.B$	$R.C$	$S.A$	$S.B$	$S.C$
	a_1	b_1	c_1	a_1	b_2	c_2
	a_1	b_1	c_1	a_1	b_3	c_2
	a_1	b_1	c_1	a_2	b_2	c_1
	a_1	b_2	c_2	a_1	b_2	c_2
	a_1	b_2	c_2	a_1	b_3	c_2
	a_1	b_2	c_2	a_2	b_2	c_1
	a_2	b_2	c_1	a_1	b_2	c_2
	a_2	b_2	c_1	a_1	b_3	c_2
	a_2	b_2	c_1	a_2	b_2	c_1

题 1. 设关系 R , S , W 各有 10 个元组, 那么这 3 个关系的笛卡尔积的元组个数为 ()。

- A. 10 B. 30 C. 1000 D. 不确定 (与计算结果有关)

答案: C

解析: R 、 S 、 W 三个关系, 每个关系中有 10 个元组, 那么其笛卡尔积就是 $10 \times 10 \times 10 = 1000$ 。



题 2：假设有关系 R 和 S ，关系代数表达式 $R - (R - S)$ 表示的是（ ）。

A. $R \cap S$ B. $R \cup S$ C. $R - S$ D. $R \times S$

答案： A

解析： $R - S$ 表示属于 R 但不属于 S ， $R - (R - S)$ 表示属于 R 但不属于 $(R - S)$ ，即相当于 $R \cap S$ 。

2. 专门的关系运算

专门的关系运算包括选择、投影、连接、除运算等。

假定有学生-课程数据库，包括如下三张数据库表，即学生关系 $Student$ 、课程关系 $Course$ 和选修关系 SC 。

Student

学号 <i>Sno</i>	姓名 <i>Sname</i>	性别 <i>Ssex</i>	年龄 <i>Sage</i>	所在系 <i>Sdept</i>
201215121	李勇	男	20	CS
201215122	刘晨	女	19	CS
201215123	王敏	女	18	MA
201215125	张立	男	19	IS

Course

课程号 <i>Cno</i>	课程名 <i>Cname</i>	先行课 <i>Cpno</i>	学分 <i>Ccredit</i>
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 <i>Sno</i>	课程号 <i>Cno</i>	成绩 <i>Grade</i>
201215121	1	92
201215121	2	85
201215121	3	88
201215122	2	90
201215122	3	80

(1) 选择 (*selection*)

选择又称为限制 (*Restriction*)，选择运算符的含义，在关系 R 中选择满足给定条件的诸元组，记作：

$$\sigma_F(R) = \{t | t \in R \wedge F(t) = \text{‘真’}\}$$

其中：F：选择条件，是一个逻辑表达式，取值为“真”或“假”。

基本形式为： $X1\theta Y1$

θ 表示比较运算符，它可以是 $>$ ， \geq ， $<$ ， \leq ， $=$ 或 $<>$ ，选择运算是从关系 R 中选取使逻辑表达式 F 为真的元组，是从行的角度进行的运算。

题 3：查询学生表年龄为19岁的全体学生。

答案：即求 $\sigma_{Sage} = 19$ (*Student*)

<i>Sno</i>	<i>Sname</i>	<i>Ssex</i>	<i>Sage</i>	<i>Sdept</i>
201215125	张立	男	19	IS
201215122	刘晨	女	19	CS

题 4：查询学生表年龄小于 20 岁的全体学生。

答案：即求 $\sigma_{Sage} < 20$ (*Student*)



Sno	Sname	Ssex	Sage	Sdept
201215122	刘晨	女	19	CS
201215123	王敏	女	18	MA
201215125	张立	男	19	IS

(2) 投影 (projection)

关系 R 上的投影是从 R 中选择出若干属性列组成新的关系，记作：

$$\pi_A(R) = t[A] | t \in R$$

其中 A 为 R 中的属性列。投影操作是从列的角度进行的运算。

例：查询学生的姓名和所在系。即求 $Student$ 关系上学生姓名和所在系两个属性上的投影，

即： $\pi_{Sname, Sdept}(Student)$ 。

Sname	Sdept
李勇	CS
刘晨	CS
王敏	MA
张立	IS

例：查询学生的姓名和所在系。即求 $Student$ 关系上学生姓名和所在系两个属性上的投影，

即 $\pi_{Sname, Sdept}(Student)$ 。

Sdept
CS
IS
MA

(3) 连接 (join)

连接也称为 θ 连接。它是从两个关系的笛卡尔积中选取属性间满足一定条件的元组。记作：

$$R \bowtie S = \left\{ \widehat{t_r t_s} | t_r \in R \wedge t_s \in S \wedge t_r[A] \theta t_s[S] \right\}$$



其中， A 和 B 分别为 R 和 S 上列数相等且可比的属性组， θ 是比较运算符。连接运算从 R 和 S 的笛卡尔积 $R \times S$ 中选取 R 关系在 A 属性组上的值与 S 关系在 B 属性组上的值满足比较关系 θ 的元组。

连接运算中有两种最为重要也最常用的连接，一种是等值连接 (*equijoin*)，另一种是自然连接 (*natural join*)。

举例：有关系 R 和关系 S 。

关系 R

A	B	C
2	4	6
3	5	7
4	6	8

关系 S

B	C	D
4	6	8
5	6	7
4	6	2
6	8	5



笛卡尔积：

R.A	R.B	R.C	S.B	S.C	S.D
2	4	6	4	6	8
2	4	6	5	6	7
2	4	6	4	6	2
2	4	6	6	8	5
3	5	7	4	6	8
3	5	7	5	6	7
3	5	7	4	6	2
3	5	7	6	8	5
4	6	8	4	6	8
4	6	8	5	6	7
4	6	8	4	6	2
4	6	8	6	8	5

等值连接(*equi join*)

θ 在 “=” 时的连接为等值连接。它是从关系 R 和 S 的广义笛卡尔积中选取 A 、 B 属性值相等的那些元组，即等值连接为：

$$R \bowtie_{A=B} S = \left\{ \widehat{t_r t_s} \mid t_r \in R \wedge t_s \in S \wedge t_r[A] = t_s[B] \right\}$$

自然连接(*natural join*)

自然连接是一种特殊的等值链接，它要求两个关系中进行比较的分量必须是相同的属性组，并且在结果中把属性重复的列去掉。即若 R 和 S 中具有相同的属性组 B ， U 为 R 和 S 的全体属性集合，则自然连接可记作：

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

一般的连接操作是从行的角度进行运算，但自然连接还需取消重复列，所以是同时从行和列的角度进行运算。

例：求 R 与 S 的自然连接。



关系 R 和关系 S 中的共同属性组为 B 和 C，找出笛卡尔积中 B 与 C 对应相等的元组，去掉其中重复的属性值。

A	B	C	D
2	4	6	8
2	4	6	2
4	6	8	5

左连接 (*left join*)

在自然连接的基础上加上左边表上不包含自然连接中所含元组(行)的元组。

例：求 R 与 S 的左连接。

对照关系 R 中，元组 3、7、5 不包含在自然连接中，在自然连接的基础上加上该元组，没有属性的列补空值。

A	B	C	D
2	4	6	8
2	4	6	2
4	6	8	5
3	5	7	<i>null</i>

右连接 (*right join*)

在自然连接的基础上加上右边表上不包含自然连接中所含元组(行)的元组。

例：求 R 与 S 的右连接。



A	B	C	D
2	4	6	8
2	4	6	2
4	6	8	5
null	5	8	7

外连接(*outer join*)

外连接 = 左连接 + 右连接。

例：求 R 与 S 的外连接。

A	B	C	D
2	4	6	8
2	4	6	2
4	6	8	5
3	5	7	null
null	5	8	7

(4) 除运算(*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，记作：

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

其中 Y_x 为 x 在 R 中的象集， $x = t_r[X]$ 。



题 5：查询至少选修1号课程和3号课程的学生号码。

答案：（1）首先建立一个临时关系 K ：

K
Cno
1
3

（2）求 $\Pi_{Sno} Cno (SC) \div k$

（3）结果为 {201215121}。

题 6：查询至少选修一门其直接先行课为 5 号课程的学生姓名。

参考答案： $\Pi_{sname}(\Pi_{sno}(\delta_{cpno} = '5')(course) \bowtie SC \bowtie \Pi_{sno, sname}(Student))$

参考解析：看到一个学生表里面的信息不全，没有公共属性；没有课程相关的，所以要先将学生表与课程表 $Course$ 还有连接在一起，还有 SC 表这三个表连接起来选修一门直接先行课为 5 号课 $\delta_{cpno} = 5$ ，查学生姓名但这个课程表并没有学生姓名，所以先对 SC 表做连接找到 5 号课对应的课程号，根据课程号能找到对应的学号（属性）。

条件：选修一门直接先行课为 5 号课 $\delta_{cpno} = 5 (course)$

投影：5 号课对应的课程号 Π_{sno}

学号对应的学生姓名 $\Pi_{sno, sname}(Student)$ 。



课时二 练习题

1. 关系数据库管理系统应能实现的专门关系运算包括（ ）。

A. 排序、索引、统计

B. 选择、投影、连接

C. 关联、更新、排序

D. 显示、打印、制表

2. 等值连接与自然连接的区别是什么？

3. 关系 R 和 S 如下图所示，试计算 $R \div S$ 。

R

A	B	C	D
a	b	c	d
a	b	e	f
a	b	h	k
b	d	e	f
b	d	d	I
c	k	c	d
c	k	e	f

S

C	D
c	d
e	f

4. 设有下列四个关系模式：

$S (SNO, SNAME, CITY)$

$P (PNO, PNAME, COLOR, WEIGHT)$

$J (JNO, JNAME, CITY)$

$SPJ (SNO, PNO, JNO, QTY)$

配套课程 习题答案



其中，供应商表 S 由供应商号 (SNO)、供应商姓名 ($SNAME$)、供应商所在城市 ($CITY$) 组成，记录各个供应商的情况；零件表 P 由零件号 (PNO)、零件名称 ($PNAME$)、零件颜色 ($COLOR$)、零件重量 ($WEIGHT$) 组成，记录各种零件的情况；工程项目表 J 由项目号 (JNO)、项目名 ($JNAME$)、项目所在城市 ($CITY$) 组成，记录各个工程项目的情况；供应情况表 SPJ 由供应商号 (SNO)、零件号 (PNO)、项目号 (JNO)、供应数量 (QTY) 组成，记录各供应商供应各种零件给各工程项目的情况。分别用关系代数完成下列查询：

- 1) 求供应工程项目号为 $J1$ 工程零件的供应商号 SNO
- 2) 求供应工程项目号为 $J1$ 工程零件号为 $P1$ 的供应商号 SNO
- 3) 求供应工程项目号为 $J1$ 工程红色零件的供应商号 SNO
- 4) 求至少使用天津供应商生产的红色零件的工程号 JNO
- 5) 求至少用了 $S1$ 供应商所供应的全部零件的工程号 JNO

5. 现有如下关系模式：

雇员 (员工姓名, 居住城市, 居住街道) 工作 (员工姓名, 公司名, 工资)

公司 (公司名, 公司所在城市) 主管 (员工姓名, 主管姓名)

用关系代数完成下列查询：

- 1) 找出所有在公司名为 “*firstbank*” 的公司工作的员工，显示员工姓名。
- 2) 显示为 “*firstbank*” 公司工作的员工姓名和居住城市。
- 3) 找出所有为 “*firstbank*” 公司工作且工资在1000 元以上的员工，显示员工姓名和工资。
- 4) 找出每个员工工资都在1000 元以上的公司，显示公司名。
- 5) 找出主管人员 *Smith* 领导的员工姓名及员工居住的城市。

6. 设有3个关系：

S ($S\#$, $SNAME$, AGE , SEX)

SC ($S\#$, $C\#$, $CNAME$)

C ($C\#$, $CNAME$, $TEACHER$)

试用关系代数表达式表示下列查询语句：



- 1) 检索 *LIU* 老师所授课程的课程号和课程名。
- 2) 检索年龄大于 23 岁的男学生的学号和姓名。
- 3) 检索学号为 *S3* 学生所学课程的课程名与任课教师名。
- 4) 检索至少选修 *LIU* 老师所授课程中一门课的女学生姓名。
- 5) 检索 *WANG* 同学不学的课程的课程名。
- 6) 检索全部学生都选修的课程的课程号与课程名。
- 7) 检索选修课程包含 *LIU* 老师所授全部课程的学生学号。



课时三 数据定义与数据更新

考点	重要程度	占分	题型
1. 数据定义	★★★★	2~10	选择题、简答题
2. 数据更新	★★★★★	5~10	选择题、简答题

1. 数据定义

关系数据库系统支持三级模式结构，其模式、外模式和内模式中的基本对象有模式、表、视图和索引等。因此SQL的数据定义功能包括模式定义、表定义、视图和索引的定义，如下图所示。

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	ALTER INDEX

1) 模式的定义

在SQL中，模式定义语句如下：

CREATE SCHEMA <模式名> AUTHORIZATION <用户名>;

如果没有指定<模式名>，那么<模式名>隐含为<用户名>。

要创建模式，调用该命令的用户必须拥有数据库管理员权限，或者获得了数据库管理员授予的CREATE SCHEMA的权限。

例1：为用户WANG定义一个学生-课程模式S-T

解：CREATE SCHEMA “S-T” AUTHORIZATION WANG;

注：定义模式实际上定义了一个命名空间，在这个空间中可以进一步定义该模式包含的数据对象，例如基本表、视图、索引等

目前，在CREATE SCHEMA中可以接收CREATE TABLE，CREATE VIEW和GRANT子句。也就是说用户可以在创建模式的同时在这个模式定义中进一步创建基本表、视图，定义授权。即CREATE SCHEMA <模式名> AUTHORIZATION <用户名> [<表定义子句>|<视图定义子句>|<授权定义子句>];

例2：为用户ZHANG创建一个模式TEST，并且在其中定义一个表TAB1。

配套课程 习题答案



```
CREATE SCHEMA TEST AUTHORIZATION ZHANG

CREATE TABLE TAB1(COL1 SMALLINT,
                  COL2 INT,
                  COL3 CHAR(20),
                  COL4 NUMERIC(10, 3),
                  COL5 DECIMAL(5, 2)
                );
```

题 1：下列 SQL 语句命令，属于 DDL 语言的是（ ）。

- A. SELECT B. CREATE C. UPDATE D. DELETE

答案： B

解析：DDL (data definition language) 是数据定义语言：DDL 比 DML 要多，主要的命令有 CREATE、ALTER、DROP 等，DDL 主要是用在定义或改变表 (TABLE) 的结构，数据类型，表之间的链接和约束等初始化工作上，他们大多在建立表时使用。

2) 模式的删除

在 SQL 中，删除模式语句如下：

```
DROP SCHEMA<模式名> <CASCADE|RESTRICT>;
```

其中 CASCADE 和 RESTRICT 两者必选其一。选择了 CASCADE(级联)，表示在删除模式的同时把该模式中所有的数据库对象删除；选择了 RESTRICT(限制)，表示如果该模式中已经定义了下属的数据库对象(如表、视图等)，则拒绝该删除语句的执行。只有当该模式中没有任何下属的对象时才能执行 DROP SCHEMA 语句。

例 3：删除模式 ZHANG，并且将模式下定义的表 TAB1 删除。

```
DROP SCHEMA ZHANG CASCADE;;
```

该语句删除了模式 ZHANG，同时，该模式中已经定义的表 TAB1 也被删除了。

3) 定义基本表

创建了一个模式就建立了一个数据库的命名空间，一个框架。在这个空间中首先要定义的是该模式包含的数据库基本表。

SQL 语言使用 CREATE TABLE 语句定义基本表，其基本格式如下：

```
CREATE TABLE<表名>(<列名><数据类型>[列级完整性约束条件],<列名><数据类型>[列级完整性约束条件]...)
```

配套课程 习题答案



[, <表级完整性约束条件>]) ;

建表的同时通常还可以定义与该表有关的完整性约束条件，这些完整性约束条件被存入系统的数据字典中，当用户操作表中的数据时由关系数据库管理系统自动检查该操作是否违背这些完整性约束条件。如果完整性约束条件设计该表的多个属性列，则必须定义在表级上，否则既可以定义在列级也可以定义在表级。

例 4：建立一个"学生表"Student

```
CREATE TABLE Student
  (Sno CHAR(9) PRIMARY KEY,          /*列级完整性约束条件，Sno 是主码*/
   Sname CHAR(20) UNIQUE,           /*Sname 取唯一值*/
   Ssex CHAR(2),
   Sage SMALLINT,
   Sdept CHAR(20)
);
```

例 5：建立一个"课程"表 Course

```
CREATE TABLE Course
  (Cno CHAR(4) PRIMARY KEY,          /*列级完整性约束条件，Cname 不能取空指*/
   Cname CHAR(40) NOT NULL,
   Cpno CHAR(4),
   Ccredit SMALLINT,
   FOREIGN KEY(Cpno) REFERENCES Course(Cno)
   /*表级完整性约束条件，Cpno 是外码，被参照表是 Course，被参照列是 Cno*/
);
```

题 2：在 Student 表的 Sname 列上建立一个唯一索引的 SQL 语句为：

```
CREATE _____ Stusname ON student(Sname);
```

答案：UNIQUE

解析：UNIQUE(唯一性)约束唯一标识数据库表中的每条记录。

题 3：设有一个工程供应数据库系统，包括如下四个关系模式：

```
S(Sno, Sname, Status, City) ;
P(Pno, Pname, Color, Weight) ;
J(Jno, Jname, City) ;
```

配套课程 习题答案



SPJ(Sno, Pno, Jno, Qty) ;

供应商表 S 由供应商号、供应商名、状态、城市组成；

零件表 P 由零件号、零件名、颜色、重量组成；

工程项目表 J 由项目号、项目名、城市组成；

供应情况表 SPJ 由供应商号、零件号、项目号、供应数量组成；

用 T-SQL 语句建立“供应商” S 表(主码必须定义)

答案：CREATE TABLE S

```
(Sno CHAR(6) PRIMARY KEY,
Sname CHAR(10),
Status INT,
City CHAR(20))
```

解析：语法：CREATE SCHEMA <模式名> AUTHORIZATION <用户名> [<表定义子句>|<视图定义子句>|<授权定义子句>]；其中 Sno 为主键，用 PRIMARY KEY。

4) 修改基本表

SQL 语言用 ALTER TABLE 语句修改基本表，其一般格式为：

```
ALTER TABLE<表名>
[ADD [COLUMN] <新列名><数据类型> [完整性约束]]
[ADD<表级完整性约束>]
[DROP [COLUMN] <列名> [CASCADE|RESTRICT]]
[DROP CONSTRAINT<完整性约束名> [RESTRICT|CASCADE]]
[ALTER COLUMN <列名><数据类型>];
```

其中<表名>是要修改的基本表，ADD 子句用于增加新列、新的列级完整性约束条件和新的表级完整性约束条件。DROP COLUMN 子句用于删除表中的列，如果指定了 CASCADE 短语，则自动删除引用了该列的其他对象，比如视图；如果指定了 RESTRICT 短语，则如果该列被其他对象引用，RDBMS 将拒绝删除该列。DROP CONSTRAINT 子句用于删除指定的完整性约束条件。ALTER COLUMN 子句用于修改原有的列定义，包括修改列名和数据类型。

例 6：向 Student 表增加“入学时间”列，其数据类型为日期型

```
ALTER TABLE Student ADD S_entrance DATE;
```

不论基本表中原来是否已有数据，新增加的列一律为空值。

配套课程 习题答案



例 7：将年龄的数据类型由字符型(假设原来的数据类型是字符型)改为整型

```
ALTER TABLE Student ALTER COLUMN Sage INT;
```

例 8：增加课程名称必须取唯一值的约束条件

```
ALTER TABLE Course ADD UNIQUE(Cname);
```

5) 删除基本表

当某个基本表不再需要时，可以使用 DROP TABLE 语句删除它。其一般格式为：

```
DROP TABLE <表名> [RESTRICT|CASCADE];
```

若选择 RESTRICT，则该表的删除是有限制条件的。欲删除的基本表不能被其它表的约束所引用(如 CHECK, FOREIGN KEY 等约束)，不能有视图，不能有触发器(trigger)，不能有存储过程或函数等。如果存在这些依赖该表的对象，则此表不能被删除。

若选择 CASCADE，则该表的删除没有限制条件。在删除基本表的同时，相关的依赖对象，例如视图，都将被一起删除。

例 9：删除 Student 表

```
Drop TABLE Student CASCADE;
```

基本表定义一旦被删除，不仅表中的数据和此表的定义将被删除，而且此表上建立的索引、触发器等对象一般也都将被删除。

2. 数据更新

数据更新操作有三种：向表中添加若干行数据、修改表中的数据和删除表中的若干行数据。

1) 插入数据

SQL 的数据插入语句 INSERT 通常有两种形式，一种是插入一个元组，另一种是插入子查询结果。后者可以一次插入多个元组。

插入元组：

插入元组的 INSERT 语句的格式为：

```
INSERT INTO<表名> [(<属性列 1>[,<属性列 2>]…)] VALUES(<常量 1>[,<常量 2>]…);
```

其功能是将元组插入指定表中。其中新元组的属性列1的值为常量1，属性列2的值为常量2，…。INTO 子句中没有出现的属性列，新元组在这些列上将取空值。但必须注意的是，在表定义时说明了 NOT NULL 的属性列不能取空值，否则会出错。

配套课程 习题答案



如果 INTO 子句中没有指明任何属性列名，则新插入的元组必须在每个属性列上均有值。

例 1：将一个新学生元组插入到 Student 表中

```
INSERT INTO Student(Sno, Sname, Ssex, Sdept, Sage) VALUES(‘201215128’ , ‘陈冬’ ,  
‘男’ , ‘IS’ , 18);
```

在 INTO 子句中指出了表名 Student，并指出了新增的元组在哪些属性上要赋值，属性的顺序可以与 CREATE TABLE 中的顺序不一样。VALUES 子句对新元组的各属性赋值，字符串常数要用单引号(英文字符)括起来，

如果在 INTO 子句中指出了表名，没有指出属性名。这表示新元组要在表的所有属性列上都指定值，属性列的次序与 CREATE TABLE 中的次序相同。

例 2：插入一条选课记录

```
INSERT INTO SC(Sno, Cno) VALUES(‘201215126’ , ‘1’);
```

关系数据库管理系统将在新插入记录的 Grade 列上自动地赋空值，

或者：

```
INSERT INTO SC VALUES(‘201215128’ , ‘1’ ,NULL);
```

因为没有指出 SC 的属性名，在 Grade 列上要明确给出空值，

插入子查询结果，

子查询不仅可以嵌套在 SELECT 语句中用以构造父查询的条件，也可以嵌套在 INSET 语句中以生成要插入的批量数据，

插入子查询结果的 INSET 语句格式为：

```
INSERT INTO<表名> (<属性列 1>[,<属性列 2>]) 子查询;
```

例 3：对每一个系，求学生的平均年龄，并把结果存入数据库。

首先在数据库中建立一个新表，其中一列存放系名，另一列存放相应的学生平均年龄

```
CREATE TABLE Dept_age (Sdept CHAR(15) , Avg_age SMALLINT);
```

然后对 Student 表按系分组求平均年龄，再把系名和平均年龄存入新表

```
INSERT  
INTO Dept_age(Sdept, Avg_age)  
SELECT Sdept, AVG(Sage)  
FROM Student  
GROUP BY Sdept;
```

配套课程 习题答案



题 1：用下面的 T-SQL 语句建立一个基本表：

CREATE TABLE Student (Sno CHAR(4) PRIMARY KEY, Sname CHAR(8) NOT NULL, Sex Char(2), Age INT) 可以插入到表中的元祖是（ ）。

- A. '5021', '刘洋', 男, 21 B. '5021', '刘洋', NULL, 21
 C. '5021', NULL, 男, 21 D. '5021', '刘洋', NULL, NULL

答案：D

解析：Sno 为主键，要求非空，Sname 为 NOT NULL 非空，不选择 C，同样 Char(2)、Age INT 无法插入男和 21，故选择 D。

题 2：基于题 1，用 SQL 语句将 (S2, P4, J6, 400) 插入供应情况关系。

答案：INSERT INTO SPJ VALUES ('S2', 'P4', 'J6', 400)

2) 修改数据

修改操作又称为更新操作，其语句的一般格式为：

UPDATE<表名>
 SET<列名>=<表达式> [, <列名>=<表达式>] ...
 [WHERE<条件>];

其功能是修改指定表中满足 WHERE 子句条件的元组。其中 SET 子句给出<表达式>的值用于取代相应的属性列值。如果省略 WHERE 子句，则表示要修改表中的所有元组。

修改某一个元祖的值：

例 4：将学生 201215121 的年龄改为 22 岁

UPDATE Student SET Sage=22 WHERE Sno= '201215121' ;

修改多个元祖的值：

例 5：将所有学生的年龄增加 1 岁

UPDATE Student SET Sage=Sage+1;

带子查询的修改语句：

子查询也可以嵌套在 UPDATE 语句中，用以构造修改的条件

例 6：将计算机科学系的全体学生成绩修改为零

UPDATE SC SET Grade=0 WHERE Sno IN
 (SELECT Sno FROM Student WHERE Sdpt='CS');

配套课程 习题答案



题 3：基于题 2 数据，用 SQL 语句将全部红色零件改为蓝色。

答案：UPDATE P SET Color='蓝' WHERE Color='红'；

3) 删除数据

删除语句的一般格式为：

```
DELETE
  FROM<表名>
  [WHERE<条件>];
```

DELETE 语句的功能是从指定表中删除满足 WHERE 子句条件的所有元组。如果省略 WHERE 子句则表示删除表中全部元组，但表的定义仍在字典中。也就是说，DELETE 语句删除的是表中的数据，而不是关于表的定义。

删除某一个元组的值：

例 7：删除学号为 201215128 的学生记录

```
DELETE
  FROM Student
  WHERE Sno= '201215128' ;
```

删除多个元组的值：

例 8：删除所有的学生选课记录

```
DELETE FROM SC;
```

这条 DELETE 语句将使 SC 成为空表，它删除了 SC 的所有元组；

带子查询的删除语句：

子查询同样也可以嵌套在 DELETE 语句中，用以构造执行删除操作的条件；

例 9：删除计算机科学系所有学生的选课记录

```
DELETE
  FROM SC
  WHERE Sno IN
    (SELECT Sno FROM Student
    Where Sdept=' CS' );
```

题 4：给定 DEPT 表，若执行下面列出的操作，哪个操作不能成功执行？（ ）。

A. 从 DEPT 中删除部门号 = ' 03 ' 的行

配套课程 习题答案



B. 在 DEPT 中插入行 ('06', '计划部', '6号楼')

C. 在 DEPT 中部门号 = '02' 的部门号改为 '10'

D. 将 DEPT 中部门号 = '01' 的地址改为 5 号楼

EMP

雇员名	雇员号	部门号	工资
001	张山	02	2000
010	王宏达	01	1200
056	马林生	02	1000
101	赵敏	04	1500

DEPT

部门号	部门名	地址
01	业务部	1号楼
02	销售部	3号楼
03	服务部	2号楼
04	财务部	4号楼

答案： C

解析：因为在EMP 中存在部门号为 '02' 的记录，所以不能将 DEPT 中部门号 = '02' 的部门号改为 '10'。



课时三 练习题

1. 若要在基本表 S 中增加一列 CN(课程名), 可用()。

A. ADD TABLES (CN CHAR (8))

B. ADD TABLES ALTER (CN CHAR (8))

C. ALTER TABLES ADD (CN CHAR (8))

D. ALTER TABLES ADD (CN CHAR (8))

2. 在基本 SQL 语言中, 下列不能实现的是()。

A. 定义基本表

B. 修改基本表

C. 查询和修改视图

D. 并发控制

3. SQL 语句中, 添加表属性的命令是()。

A. DELETE

B. UPDATE

C. ALTER TABLE

D. REMOVE

4. 下题是一个创建表的代码, 请根据上下语句及提示完成填空, 其中学号(Sid)为主键, 定长为8位, 姓名为非空, 性别需要约束只能输入男或者女, 生日为日期类型

Create Table SoftStudent

(

Sid _____

Sname varchar(20) not null,

Ssex char(2) _____

Sbirthday _____

)

5. 删除数据库 company, 需要执行的 T-SQL 语句是_____。

6. 根据员工人事表 employee, 完成如下题目。

配套课程 习题答案



列名	数据类型	是否为空	主外键约束	列含义
emp_no	char(5)	Not null	primary key	员工编号
emp_name	char(10)	Not null		员工姓名

sex	char(1)	Not null		性别
dept	char(4)	Not null		所属部门
title	char(6)	Not null		职称
date_hired	datetime	Not null		到职日
birthday	datetime	null		生日
salary	int	Not null		薪水
addr	char(50)	null		住址
Mod_date	datetime	Default(getdate())		操作者

- (1) 写出创建 employee 表的 T-SQL 语句。
- (2) 向 employee 表中插入一条记录数据 (‘0081’，‘王华’，‘男’，‘技术’，‘工程师’)。
- (3) 删除表 employee 中所有姓“陈”的员工数据。



课时四 数据查询

考点	重要程度	占分	题型
1. 单表查询	★★★★★	5~10	选择、简答
2. 连接查询	★★★★★	5~10	选择、简答
3. 嵌套查询	★★★★	5~10	选择、简答
4. 集合查询	★★★★	5~10	选择、简答

1. 单表查询

1) 选择表中的若干列

查询指定列

单表查询是指仅涉及一个表的查询。

学生-课程数据库：

学生关系 Student、课程关系 Course 和选修关系 SC

Student

学号 Sno	姓名 Sname	性别 Ssex	年龄 Sage	所在系 Sdept
201215121	李勇	男	20	CS
201215122	刘晨	女	19	CS
201215123	王敏	女	18	MA
201215125	张立	男	19	IS

Course

课程号 Cno	课程名 Cname	先行课 Cpno	学分 Ccredit
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 Sno	课程号 Cno	成绩 Grade
201215121	1	92
201215121	2	85

配套课程 习题答案



201215121	3	88
201215122	2	90
201215122	3	80

例 1：查询全体学生的姓名、学号、所在系

```
SELECT Sname, Sno, Sdept FROM Student;
```

查询全部列

将表中的所有属性列都选出来有两种方法：一种方法就是在 SELECT 关键字后列出所有列名；

如果列的显示顺序与其在基表中的顺序相同，也可以简单地将<目标列表达式>指定为*。

例 2：查询全体学生的详细记录

```
SELECT *
```

```
FROM Student;
```

等价于：

```
SELECT Sno, Sname, Ssex, Sage, Sdept
```

```
FROM Student;
```

查询经过计算的值

SELECT 子句的<目标列表达式>不仅可以是表中的属性列，也可以是表达式。

例 3：查询全体学生的姓名、出生年份和所在院系，要求用小写字母表示系名

```
SELECT Sname NAME, 'Year of Birth:' BIRTH, 2014-Sage
```

```
BIRTHDAY, LOWER(Sdept) DEPARTMENT FROM Student;
```

2) 选择表中的若干元组

消除取值重复的行

两个本来并不完全相同的元组在投影到指定的某些列上后，可能会变成相同的行。可以用 DISTINCT 消除它们。

例 4：查询选修了课程的学生学号

```
SELECT Sno FROM SC;
```

该查询结果包含了许多重复的行，如果需要去掉重复行，需要加 DISTINCT。如果没有指定 DISTINCT 关键词，则默认为 ALL，即保留结果表中取值重复的行。

查询满足条件的元组

查询满足条件的元组可以通过 WHERE 子句实现，常用的查询条件如下：

配套课程 习题答案



常用的查询条件

查询条件	谓词
比较	=, >, <, >=, <=, !=, <>, !>, !<; NOT+上述比较运算符
确定范围	BETWEEN AND, NOT BETWEEN AND
确定集合	IN, NOT IN
字符匹配	LIKE, NOT LIKE
空值	IS NULL, IS NOT NULL
多重条件 (逻辑运算)	AND, OR, NOT

① 比较大小

用于进行比较的运算符一般包括=(等于), >(大于), <(小于), >=(大于等于), <=(小于等于), !=或<>(不等于), !>(不大于), !<(不小于)。

例 5: 查询考试成绩不及格的学生的学号

```
SELECT DISTINCT Sno FROM SC WHERE Grade<60;
```

② 确定范围

谓词 BETWEEN...AND... 和 NOT BETWEEN...AND... 可以用来查找属性值在(或不在)指定范围内的元组, 其中 BETWEEN 后是范围的下限(即低值), AND 后是范围的上限(即高值)。

例 6: 查询年龄不在 20 ~ 23 岁之间的学生姓名、系别和年龄

```
SELECT Sname, Sdept, Sage
  FROM Student
 WHERE Sage NOT BETWEEN 20 AND 23;
```

③ 确定集合

谓词 IN 可以用来查找属性值属于指定集合的元组;

例 7: 查询计算机科学系(CS)、数学系(MA)和信息系(IS)学生的姓名和性别

```
SELECT Sname, Ssex
  FROM Student
 WHERE Sdept IN( 'CS' , 'MA' , 'IS' );
```

与 IN 相对的谓词是 NOT IN, 用于查找属性值不属于指定集合的元组;

配套课程 习题答案



题 1. SELECT 语句查询中的谓词“!=ALL”与运算符_____等价。

答案：NOT IN

解析：与 IN 相对的谓词是 NOT IN，用于查找属性值不属于指定集合的元组。

④字符匹配

谓词 LIKE 可以用来进行字符串的匹配。其一般语法格式如下：[NOT] LIKE ‘<匹配串>’ [ESCAPE ‘<换码字符>’]；

其含义是查找指定的属性列值与<匹配串>相匹配的元组。<匹配串>可以是一个完整的字符串，也可以含有通配符%和_。其中：

%(百分号)代表任意长度(长度可以为 0)的字符串；

例如 a%b 表示以 a 开头，以 b 结尾的任意长度的字符串。如 acb、addgb、ab 等都满足该匹配串。

_ (下划线)代表单个字符；

例如 a_b 表示以 a 开头，以 b 结尾的长度为 3 的任意字符串。如 acb、afb 等都满足该匹配串。

例 8：查询名字中第二个字为"阳"的学生的姓名和学号

```
SELECT Sname
```

```
FROM Student
```

```
WHERE Sname LIKE '_阳%' ;
```

如果 LIKE 后面的匹配串中不含通配符，则可以用=(等于)运算符取代 LIKE 谓词，用!=或<>(不等于)运算符取代 NOT LIKE 谓词。

数据库字符集为 ASCII 时，一个汉字需要两个_；当字符集为 GBK 时，只需要一个_。如果用户要查询的字符串本身就含有通配符%或_，这时就要使用 ESCAPE ‘<换码字符>’ 短语对通配符进行转义了。

配套课程 习题答案



例 9：查询 DB_Design 课程的课程号和学分

```
SELECT Cno, Ccredit
  FROM Course
 WHERE Sname LIKE 'DB_Design' ESCAPE ' ';
```

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有，转义为普通的"_"字符。

⑤涉及空值的查询

例 10：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;
```

注意这里的"IS"不能用等号(=)代替

⑥多重条件查询

逻辑运算符 AND 和 OR 可用来连接多个查询条件。AND 的优先级高于 OR，但用户可以用括号改变优先级；

在 IN 谓词实际上是多个 OR 运算符的缩写，也可以用 OR 运算符写成如下等价形式：

```
SELECT Sname, Ssex
  FROM Student
 WHERE Sdept= 'CS' OR Sdept= 'MA' OR Sdept= 'IS' ;
```

ORDER BY 子句

用户可以用 ORDER BY 子句对查询结果按照一个或多个属性列的升序(ASC)或降序(DESC)排序，默认值为升序



例 11：查询全体学生情况，查询结果按所在系的系号升序排序，同一系中的学生按年龄降序排列

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

聚集函数

为了进一步方便用户，增强检索功能，SQL 提供了许多聚集函数，主要有：

COUNT(*) 统一元组个数

COUNT([DISTINCT|ALL] <列名>) 统计一列中值的个数

SUM([DISTINCT|ALL] <列名>) 计算一列值的总和(此列必须是数值型)

AVG([DISTINCT|ALL] <列名>) 计算一列值的平均值(此列必须是数值型)

MAX([DISTINCT|ALL] <列名>) 求一列值中的最大值

MIN([DISTINCT|ALL] <列名>) 求一列值中的最小值

如果指定 DISTINCT 短语，则表示在计算时要取消列中的重复值。如果不指定 DISTINCT 短语或指定 ALL 短语(ALL 为默认值)，则表示不取消重复值。

例 12：查询学生 201215012 选修课程的总学分数

```
SELECT SUM(Ccredit)
FROM SC, Course
WHERE Sno= '201215012' AND SC.Cno=Course.Cno;
```

当聚集函数遇到空值时，除 COUNT(*) 外，都跳过空值而只处理非空值。COUNT(*) 是对元组进行计数，某个元组的一个或部分列取空值不影响 COUNT 的统计结果。

WHERE 子句中是不能用聚集函数作为条件表达式的。聚集函数只能用于 SELECT 子句和 GROUP BY 中的 HAVING 子句。

GROUP BY 子句

配套课程 习题答案



GROUP BY 子句将查询结果按某一列或多列的值分组，值相等的为一组。对查询结果分组的目的是为了细化聚集函数的作用对象。如果未对查询结果分组，聚集函数将作用于整个查询结果。分组后聚集函数将作用于每一个组，即每一个组都有一个函数值。

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。

题 2. 下列聚合函数中不忽略控制 (null) 的是 () 。

- A. SUM(列名) B. MAX(列名) C. COUNT(*) D. AVG(列名)

答案： C

解析：在聚合函数中遇到空值时，除了 COUNT(*) 外，都跳过空值而去处理非空值。

例 13：查询选修了三门以上课程的学生学号

```
SELECT Sno  
FROM SC  
GROUP BY Sno  
HAVING COUNT(*)>3;
```

先用 GROUP BY 子句按 Sno 进行分组，再用聚集函数 COUNT 对每一组计数； HAVING 短语给出了选择组的条件，只有满足条件(即元组个数>3，表示此学生选修的课超过 3 门)的组才会被选出来。

WHERE 子句与 HAVING 短语的区别在于作用对象不同。WHERE 子句作用于基本表或视图，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。

题 3. 在分组检索中，要去掉不满足条件的分组和不满足条件的记录，应当_____。

- A. 使用 WHERE 子句
B. 使用 HAVING 子句

配套课程 习题答案



C. 先使用 HAVING 子句，再使用 WHERE 子句

D. 先使用 WHERE 子句，再使用 HAVING 子句

答案：D

解析：用 WHERE 子句去掉不满足条件的记录，再用 HAVING 子句进行分组。

题 4. 如下关系数据库：学生(学号，姓名，性别，专业、奖学金)，课程(课程号，名称，学分)，学习(学号，课程号，分数)。用 SQL 语言完成下列操作：

1) 检索没有任何一门课程成绩在80分以下的所有学生的信息，包括学号、姓名和专业；

2) 检索没有获得奖学金、同时至少有一门课程成绩在95分以上的学生信息，包括学号、姓名和专业。

答案：1) `SELECT 学号, 姓名, 专业 FROM 学生 WHERE 学号 NOT IN (SELECT 学号 FROM 学习 WHERE 分数 < 80)`

2) `SELECT 学号, 姓名, 专业 FROM 学生, 学习 WHERE 学生. 学号=学习. 学号 AND 学习. 课程号=课程. 课程号 AND 学生. 奖学金 <= 0 AND 学习. 分数 > 95。`

2. 连接查询

若一个查询同时涉及两个以上的表，则称之为连接查询。

1) 等值与非等值连接查询

连接查询的 WHERE 子句中用来连接两个表的条件称为连接条件或连接谓词，其一般格式为：[<表名 1>.] <列名 1><比较运算符> [<表名 2>.] <列名 2>；

此外连接谓词还可以使用下面形式：[<表名 1>.] <列名 1> BETWEEN [<表名 2>.] <列名 2> AND [<表名 3>.] <列名 3>。

当连接运算符为=时，称为等值连接。使用其他运算符称为非等值连接。

连接谓词中的列名称为连接字段。连接条件中的各连接字段类型必须是可比的，但名字不必相同。



例 1：查询每个学生及其选修课程的情况

```
SELECT Student.* , SC.*  
FROM Student , SC  
WHERE Student.Sno=SC.sno;
```

本例中，SELECT 子句与 WHERE 子句中的属性名前都加上了表名前缀，这是为了避免混淆。如果属性名在参加连接的各表中是唯一的，则可以省略表名前缀。

若在等值连接中把目标列中重复的属性列去掉则为自然连接。

自身连接

连接操作不仅可以在两个表之间进行，也可以是一个表与其自己进行连接，称为表的自身连接。

例 2：查询每一门课的间接先修课(即先修课的先修课)

```
SELECT FIRST.Cno , SECOND.Cpno  
FROM Course FIRST , Course SECOND  
WHERE FIRST.Cpno=SECOND.Cno;
```

在 Course 表中只有每门课的直接先修课信息，而没有先修课的先修课。要找到这个信息，必须先对一门课找到其先修课，再按此先修课的课程号，查找它的先修课程。这就要将 Course 表与其自身连接。

为此，要为 Course 表取两个别名，一个是 FIRST，另一个是 SECOND

完成该查询的 SQL 语句为：

```
SELECT FIRST.Cno , SECOND.Cpno  
FROM Course FIRST , Course SECOND  
WHERE FIRST.Cpno=SECOND.Cno;
```

配套课程 习题答案



结果表中没有 201215123 和 201215125 两个学生的信息，原因在于他们没有选课，在 SC 表中没有相应的元组，导致 Student 中这些元组在连接时被舍弃了。

有时想以 Student 表为主体列出每个学生的基本情况及其选课情况。若某个学生没有选课，仍把 Student 的悬浮元组保存在结果关系中，而在 SC 表的属性上填空值 NULL，这时就需要使用外连接。

```
SELECT Student. Sno, Sname, Ssex, Sdept, Cno, Grade  
FROM Student LEFT OUTER JOIN SC ON(Student. Sno=SC. Sno);
```

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。

多表连接

连接操作除了可以是两表连接、一个表与其自身连接外，还可以是两个以上的表进行连接，后者通常称为多表连接。

例 3：查询每个学生的学号、姓名、选修的课程名及成绩

```
SELECT Student. Sno, Sname, Cname, Grade  
FROM Student, SC, Course  
WHERE Student. Sno=SC. Sno AND SC. Cno=Course. Cno;
```

关系数据库管理系统在执行多表连接时，通常是先进行两个表的连接操作，再将其连接结果与第三个表进行连接。

3. 嵌套查询

在 SQL 语言中，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 (Nested Query)。

上层的查询块称为外层查询或父查询，下层查询块称为内层查询或子查询。

需要特别指出的是，子查询的 SELECT 语句中不能使用 ORDER BY 子句，ORDER BY 子句只能对最终查询结果排序。

配套课程 习题答案



1) 带有 IN 谓词的子查询

在嵌套查询中，子查询的结果往往是一个集合，所以谓词 IN 是嵌套查询中最经常使用的谓词。

例 1：查询与“刘晨”在同一个系学习的学生

```
SELECT Sno, Sname, Sdept
  FROM Student
 WHERE Sdept IN
       (SELECT Sdept FROM Student WHERE Sname='刘晨' );
```

2) 带有比较运算符的子查询

带有比较运算符的子查询是指父查询与子查询之间用比较运算符进行连接。当用户能确切知道内层查询返回的是单个值时，可以用 >、<、=、>=、<=、!= 或 <> 等比较运算符。

例 2：找出每个学生超过他自己选修课程平均成绩的课程号

```
SELECT Sno, Cno
  FROM SC x
 WHERE Grade >= (SELECT AVG(Grade) FROM SC y
                   WHERE y.Sno = x.Sno);
```

3) 带有 ANY(SOME) 或 ALL 谓词的子查询

子查询返回单值时可以用比较运算符，但返回多值时要用 ANY(有的系统用 SOME) 或 ALL 谓词修饰符。而使用 ANY 或 ALL 谓词时必须同时使用比较运算符。

例 3：查询非计算机科学系中比计算机科学系任意一个学生年龄小的学生姓名和年龄

```
SELECT Sname, Sage
  FROM Student
```

配套课程 习题答案



```

WHERE Sage < ANY
  (SELECT Sage FROM Student WHERE Sdept=' CS' )
  AND Sdept <> 'CS' ;

```

4) 带有 EXISTS 谓词的子查询

EXISTS 代表存在量词 \exists 。带有 EXISTS 谓词的子查询不返回任何数据，只产生逻辑真值 "true" 或逻辑假值 "false"；

可以利用 EXISTS 来判断 $x \in S$ 、 $S \subseteq R$ 、 $S = R$ 、 $S \cap R$ 非空等是否成立。

例 4：查询所有选修了 1 号课程的学生姓名

```

SELECT Sname
  FROM Student
 WHERE EXISTS
  (SELECT * FROM SC
 WHERE Sno=Student.Sno AND Cno= '1' );

```

题 1. 用 SQL 语句完成下列各题：现有一关系数据库，包含三个关系，具体信息如下：

描述员工信息的表 Employee(eno, ename, esex, edept, ejob)，各属性分别表示员工编号、姓名、性别、所在部门、职位；

描述工程信息的表 Project(pno, pname, ebudget)，各属性分别表示工程编号、工程名称、预算；

描述施工信息的表 Construct(eno, pno, cgs)，各属性分别表示员工编号、工程编号、工时(某员工在某工程中工作的小时数)。

用 SQL 进行表示：

1) 查询既参加了 5 号工程又参加了 4 号工程的员工编号；

2) 查询总工时在 500 小时以上的员工编号和总工时。

配套课程 习题答案



答案：1) Select eno from Construct Where pno='5' and eno in (select eno from Construct where pno='4')

2) Select eno,sum(cgs) from Construct Group by eno Having sum(cgs)>= 500

4. 集合查询

SELECT 语句的查找结果是元组的集合，所以多个 SELECT 语句的结果可进行集合操作。集合操作主要包括并操作 UNION、交操作 INTERSECT 和差操作 EXCEPT。

注意：参加集合操作的各查询结果的列数必须相同；对应项的数据类型也必须相同。

例 1：查询计算机科学系的学生及年龄不大于 19 岁的学生

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

```
UNION
```

```
SELECT * FROM Student WHERE Sage<=19 ;
```



课时四 练习题

1. SQL 中, 下列涉及空值的操作, 不正确的是 () .

2. 已知学生关系（学号，姓名，年龄，班级），要检索班级为空值的学生姓名，其 SQL 查询语句中 WHERE 子句的表达式是 _____.

3. 设有学生选课关系 SC (学号, 课程号, 成绩), 使用 SQL 语句检索每门课程的最高分.

4. 已知一个教学管理数据库 JXGL，包含以下三张数据表：学生表、课程表以及学生成绩表。它们的结构如下：

学生表(学号, 姓名, 性别, 年龄, 专业); 课程表(课程号, 课程名); 学生成绩表(学号, 课程号, 成绩); 请按照要求写出相应的 SQL 语句:

- (1) 查询和苗舟同学同一个专业的学生信息(子查询), 按年龄降序;
 - (2) 查询平均成绩大于70的学生学号, 平均成绩.

5. 现有关系数据库如下：

学生(学号,姓名,性别,专业、奖学金);课程(课程号,名称,学分);学习(学号,课程号,分数);
用 SQL 实现:

- (1) 查询没有获得奖学金、同时至少有一门课程成绩在95分以上的学生成绩，包括学号、姓名和专业；

- (2) 查询没有任何一门课程成绩在80分以下的所有学生的信息,包括学号、姓名和专业.

6. 数据库中有如下关系:

学生表 student (stuId, stuName, stuAge, stuSex)

课程表 Course (courseId, courseId, courseId, courseId)

成绩表 Scores(stuId, courseId, score)

教师表 Teacher (teacherId, teacherName)

请完成以下查询：

查询所有学生的选课情况，输出学号，姓名，课程号，课程名。

配套课程 习题答案



课时五 视图

考点	重要程度	占分	题型
1. 定义视图	★★★★	5~10	选择、简答
2. 查询视图	★★★★	5~10	选择、简答
3. 更新视图	★★★	2~5	选择、简答

视图是从一个或几个基本表(或视图)导出的表。它与基本表不同，是一个虚表。数据库中只存放视图的定义，而不存放视图对应的数据，这些数据仍存放在原来的基本表中。所以一旦基本表中的数据发生变化，从视图中查询出的数据也就随之改变了。从这个意义上讲，视图就像一个窗口，透过它可以看到数据库中自己感兴趣的数据及其变化。

视图一经定义，就可以和基本表一样被查询、被删除。也可以在一个视图之上再定义新的视图，但对视图的更新(增、删、改)操作则有一定的限制。

1. 定义视图

1) 建立视图

SQL 语言用 CREATE VIEW 命令建立视图，其一般格式为：

CREATE VIEW<视图名>[(<列名>[<列名>]…)]

AS<子查询>

[WITH CHECK OPTION];

其中，子查询可以是任意的 SELECT 语句，是否可以含有 ORDER BY 子句和 DISTINCT 短语，则取决于具体系统的实现。

WITH CHECK OPTION 表示对视图进行 UPDATE、INSERT 和 DELETE 操作时要保证更新、插入或删除的行满足视图定义中的谓词条件(即子查询中的条件表达式)。

组成视图的属性列名或者全部省略或者全部指定，没有第三种选择。如果省略了视图的各个属性列名，则隐含该视图由子查询中 SELECT 子句目标列中的诸字段组成。但在下列三种情况下必须明确指定组成视图的所有列名：

- ①：某个目标列不是单纯的属性名，而是聚集函数或列表达式；
- ②：多表连接时选出了几个同名列作为视图的字段；
- ③：需要在视图中为某个列启用新的更合适的名字。

例 1：建立信息系学生的视图。

CREATE VIEW IS_Student

配套课程 习题答案



AS

```
SELECT Sno, Sname, Sage
FROM Student
WHERE Sdept='IS';
```

本例中省略了视图 IS_Student 的列名，隐含了由子查询中 SELECT 子句中的三个列名组成。

关系数据库管理系统执行 CREATE VIEW 语句的结果只是把视图的定义存入数据字典，并不执行其中的 SELECT 语句。只是在对视图查询时，才按视图的定义从基本表中将数据查出。

例 2：建立信息系学生的视图，并要求进行修改和插入操作时仍需保证该视图只有信息系的学生。

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

由于在定义 IS_Student 视图时加上了 WITH CHECK OPTION 子句，以后对该视图进行插入、修改和删除操作时，关系数据库管理系统会自动加上 Sdept='IS' 的条件。

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，则称这类视图为行列子集视图。IS_Student 视图就是一个行列子集视图。

视图不仅可以建立在单个基本表上，也可以建立在多个基本表上。

例 3：建立信息系选修了 1 号课程的学生的视图（包括学号、姓名、成绩）。

```
CREATE VIEW IS_S1(Sno, Sname, Grade)
AS
SELECT Student.Sno, Sname, Grade
FROM Student, SC
WHERE Sdept='IS' AND
Student.Sno=SC.Sno AND
SC.Cno = '1';
```

由于视图 IS_S1 的属性列中包含了 Student 表与 SC 表的同名列 Sno，所以必须在视图名后面明确说明视图的各个属性列名。



视图不仅可以建立在一个或多个基本表上，也可以建立在一个或多个已定义好的视图上，或建立在基本表与视图上。

例 4：建立信息系选修了 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 IS_S1
WHERE Grade>=90;
```

这里的视图 IS_S2 就是建立在视图 IS_S1 之上的。

定义基本表时，为了减少数据库中的冗余数据，表中只存放基本数据，由基本数据经过各种计算派生出的数据一般是不存储的。由于视图中的数据并不实际存储，所以定义视图时可以根据应用的需要设置一些派生属性列。这些派生属性由于在基本表中并不实际存在，也称它们为虚拟列。带虚拟列的视图也称为带表达式的视图。

题 1. 设某工厂数据库中有两个基本表：试建立一个有关女车间主任的职工号和姓名的视图，其结构如下：VIEW6(ENO, ENAME)，试写出创建视图 VIEW6 的 SQL 语句。

车间基本表：DEPT(DNO, DNAME, MGR_NO)，其属性分别表示车间编号、车间名和车间主任的职工号；

职工基本表：EMP(ENO, ENAME, AGE, SEX, SALARY, DNO)，其属性分别表示职工号、姓名、年龄、性别、工资和所在车间的编号；

答案：CREATE VIEW VIEW6

```
AS SELECT ENO, ENAME
FROM DEPT, EMP
WHERE MRG_ENO=ENO AND SEX= '女' ;
```

解析：创建视图语法，CREATE VIEW<视图名>[(<列名>[<列名>]…)] AS<子查询> [WITH CHECK OPTION]。

2) 删除视图

该语句的格式为：

```
DROP VIEW<视图名>[CASCADE] ;
```



视图删除后视图的定义将从数据字典中删除。如果该视图上还导出了其他视图，则使用 CASCADE 级联删除语句把该视图和由它导出的所有视图一起删除。

基本表删除后，由该基本表导出的所有视图均无法使用了，但是视图的定义没有从字典中清除。删除这些视图定义需要显式地使用 DROP VIEW 语句。

例 1：删除视图 BT_S 和视图 IS_S1。

```
DROP VIEW BT_S;
```

```
DROP VIEW IS_S1;
```

执行此语句时由于 IS_S1 视图上还导出了 IS_S2 视图，所以该语句被拒绝执行。如果确定要删除，则使用级联删除语句：DROP VIEW IS_S1 CASCADE；

题 2. 视图能够对机密数据提供安全保护（ ）。

- A. 正确 B. 错误

答案：A

解析：视图是关系数据库系统提供给用户的以多种角度观察数据库中数据的重要机制。视图是从一个或几个基本表(或视图)导出的表，它与基本表不同，是一个虚表。视图可以对机密数据提供安全保护，同时视图提供了一定程度的数据逻辑独立性。对视图的一切操作最终要转换为对基本表的操作。

题 3. 采用定义视图的机制在数据控制方法要解决的问题是（ ）。

- A. 数据安全性 B. 数据完整性 C. 数据库恢复 D. 数据库并发控制

答案：A

解析：采用定义视图的机制在数据控制方面要解决的问题是数据安全性。

2. 查询视图

视图定义后，用户就可以像对基本表一样对视图进行查询了。

例 1：在信息系学生的视图中找出年龄小于 20 岁的学生。

```
SELECT Sno, Sage
```

```
FROM IS_Student
```

配套课程 习题答案



```
WHERE Sage<20;
```

关系数据库管理系统执行对视图的查询时，首先进行有效性检查，检查查询中涉及的表、视图等是否存在。如果存在，则从数据字典中取出视图的定义，把定义中的子查询和用户的查询结合起来，转换成等价的对基本表的查询，然后再执行修正了的查询。这一转换过程称为视图消解 (View Resolution)。

本例转换后的查询语句为：

```
SELECT Sno, Sage
  FROM Student
 WHERE Sdept='IS'
   AND Sage<20;
```

例 2：查询选修了 1 号课程的信息系学生。

```
SELECT IS_Student.Sno, Sname
  FROM IS_Student, SC
 WHERE SC.Cno='1' AND
       IS_Student.Sno=SC.Sno;
```

本查询涉及视图 IS_Student (虚表) 和基本表 SC，通过这两个表的连接来完成用户请求。

在一般情况下，视图查询的转换是直截了当的。但有些情况下，这种转换不能直接进行，查询时就会出现问题。

例 3：在 S_G 视图中查询平均成绩在 90 分以上的学生学号和平均成绩，语句为：

```
SELECT *
  FROM S_G
 WHERE Gavg>=90;
```

例 2 中定义 S_G 视图的子查询为

```
SELECT Sno, AVG(Grade)
```

配套课程 习题答案



```
FROM SC
```

```
GROUP BY Sno;
```

将本例中的查询语句与定义 S_G 视图的子查询结合，形成下列查询语句：

```
SELECT Sno, AVG(Grade)
```

```
FROM SC
```

```
WHERE AVG(Grade) >= 90
```

```
GROUP BY Sno;
```

因为 WHERE 子句中是不能用聚集函数作为条件表达式的，因此执行此修正后的查询将会出现语法错误。正确转换的查询语句应该是：

```
SELECT Sno, AVG(Grade)
```

```
FROM SC
```

```
GROUP BY Sno
```

```
HAVING AVG(Grade) >= 90;
```

目前多数关系数据库系统对行列子集视图的查询均能进行正确转换。但对非行列子集视图的查询就不一定能做转换了，因此这类查询应该直接对基本表进行。

题 1. 在视图上不能完成的操作是（ ）。

- A. 更新视图
- B. 在视图上定义新的表
- C. 查询
- D. 在视图上定义新的视图

答案： B

解析：视图是从一个或几个基本表（或其他视图）导出的表，它与基本表不同，是一个虚表。

数据库中只存放视图的定义，而不存放视图对应的数据，这些数据仍存放在原来的基本表中。基本表中的数据发生变化，从视图中查询得出的数据也随之改变。视图一经定义，就可以和基本表一样被查询、删除，也可以在一个视图之上再定义新的视图，

配套课程 习题答案



但对视图的修改(插入、删除、更新)操作则有一定的限制。所以不能在视图上定义基本表。

题 2. 视图是一种常用的数据对象, 它是提供 () 和 () 数据的另一种途径, 可以简化数据库操作。

- A. 查看, 存放 B. 查看, 检索 C. 插入, 更新 D. 检索, 插入

答案: B

3. 更新视图

例 1: 将信息系学生视图 IS_Student 中学号为 “201215122” 的学生姓名改为 “刘辰”。

```
UPDATE IS_Student SET Sname='刘辰' WHERE Sno='201215122';
```

转换后的更新语句为:

```
UPDATE Student SET Sname='刘辰' WHERE Sno='201215122' AND Sdept='IS';
```

例 2: 向信息系学生视图 IS_Student 中插入一个新的学生记录, 其中学号为 “201215129”, 姓名为 “赵新”, 年龄为 20 岁。

```
INSERT INTO IS_Student VALUES('201215129','赵新',20);
```

转换后的插入语句为:

```
INSERT INTO Student(Sno,Sname,Sage,Sdept) VALUES('201215129','赵新',20,'IS');
```

例 3: 删 除 信 息 系 学 生 视 图 IS_Student 中 学 号 为 “201215129” 的 记 录。

```
DELETE FROM IS_Student WHERE Sno='201215129';
```

转换后的删除语句为:

```
DELETE FROM Student WHERE Sno='201215129' AND Sdept='IS';
```

在关系数据库中, 并不是所有的视图都是可更新的, 因为有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。

例如, 定义视图 S_G 是由学号和平均成绩两个属性列组成的, 其中平均成绩一项是由 Student 表中对元组分组后计算平均值得来的。如果想把视图 S_G 中学号为 “201215121” 的学生的平均成绩改成 90 分, SQL 语句如下:



```
UPDATE S_GSET Gavg=90 WHERE Sno='201215121';
```

但对这个视图的更新无法转换成对基本表 SC 的更新，因为系统无法修改各科成绩，以使平均成绩为 90，所以 S_G 视图是不可更新的。

一般地，行列子集视图是可更新的。目前，各个关系数据库管理系统一般都只允许对行列子集视图进行更新，而且各个系统对视图的更新还有更进一步的规定，这些规定也不尽相同。

注：不可更新的视图与不允许更新的视图是两个不同的概念。前者指理论上已证明其是不可更新的视图。后者指实际系统中不支持其更新，但它本身有可能是可更新的视图。

题 1. 修改视图使用的命令格式的关键字是_____。

答案：ALTER VIEW。

题 2. 在视图中插入一个元组，该元组会同时插入到基本表中。 ()

答案：✓

解析：视图是虚拟表，向虚拟表中插入数据实际是插入到基本表中。



课时五 练习题

1. 试述视图的作用.
2. 试述视图与基本表的区别和联系.
3. 视图是一个“虚表”，我们可以基于（ ）来构造视图.
A. 基本表或视图 B. 视图 C. 基本表 D. 数据字典
4. 对视图的一切操作最终要转换为对基本表的操作（ ）.
A. 正确 B. 错误
5. SQL 中的视图提高了数据库系统的（ ）.
A. 完整性 B. 并发控制 C. 隔离性 D. 安全性
6. 视图是一个虚表，它是从一个或几个基本表中导出的表。在数据库中，只存放视图的_____，不存放与视图对应的数据.
7. 设有一个顾客商品关系数据库，有三个基本表，表结构如下：

商品表：Article(商品号，商品名，单价，库存量)

客户表：Customer(顾客号，顾客名，性别，年龄，电话)

订单表：OrderItem(顾客号，商品号，数量，购买价，日期)

创建一个名为 S_VIEW 视图，检索库存量低于 Smin (临界库存) 的商品信息.

8. 设学生---社团数据库有三个基本表：

学生 (学号，姓名，年龄，性别)；

社团 (编号，名称，负责人，办公地点)；

参加 (学号，编号，参加日期)；

其中：学生表的主码为学号；社团表的主码为编号；外码为负责人，被参照表为学生表，对应属性为学号；参加表的学号和编号为主码；学号为外码，其被参照表为职工表，对应属性为学号；编号为外码，其被参照表为社团表，对应属性为编号。

试用 SQL 语句表达：建立视图：社团负责人 (社团编号，名称，负责人学号，负责人姓名，负责人性别) .



课时六 数据库安全性

考点	重要程度	占分	题型
1. 自主存取控制方法	★★	1~5	选择、填空
2. 授权与回收	★★★★	2~10	选择题、简答题
3. 视图机制	★★★	1~3	选择题、简答题

1. 自助存取控制方法

大型数据库管理系统都支持自主存取控制，SQL 标准也对自主存取控制提供支持。主要通过 SQL 的 GRANT 语句和 REVOKE 语句来实现。

用户权限由两要素组成：数据库对象和操作类型。

存取控制的对象不仅有对象本身（表中的数据和属性列上的数据），还有数据库模式（包括模式、基本表、视图和索引的创建等），具体存取系统中的存取权限如下图。

对象类型	对象	操作类型
数据库模式	模式	CREATE SCHEMA
	基本表	CREATE TABLE, ALTER TABLE
	视图	CREATE VIEW
	索引	CREATE INDEX
数据	基本表和视图	SELECT, INSERT, UPDATE, DELETE, REFERENCES, ALL PRIVILEGES
	属性列	SELECT, INSERT, UPDATE, REFERENCES, ALL PRIVILEGES

题 1：在 SQL 中授权语句中使用“ALL PRIVILEGES”，表示（ ）

- A、授权所有用户
 - B、所有的操作权限
 - C、对所有的数据集合
 - D、允许再授权

参考答案: B

参考解析：ALL PRIVILEGES 代表所有的操作权限。

2、授权与回收

SQL 中使用 `GRANT` 和 `REVOKE` 语句向用户授权或收回对数据的操作权限。`GRANT` 语句向用户授予权限, `REVOKE` 语句收回已授予用户的权限。

配套课程 习题答案



1) 授权—GRANT

GRANT 语句的一般格式：

GRANT <权限>[,<权限>]...

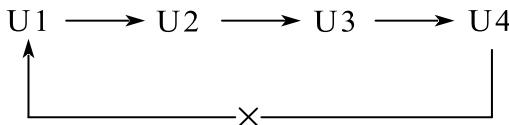
[ON <对象类型> <对象名>]

TO <用户>[,<用户>]...

[WITH GRANT OPTION];

语义：将对指定操作对象的指定操作权限授予指定的用户。那么发出 GRANT 可以是 DBA、数据库对象创建者（即属主 Owner）、拥有该权限的用户。接受权限的用户是一个或多个具体用户、也可以是 PUBLIC（全体用户）。

其中：WITH GRANT OPTION 子句：指定：可以再授予，没有指定：不能传播，但是不允许循环授权，如下图所示。



例 1：把查询 Student 表权限授给用户 U1

答案：GRANT SELECT

```

    GRANT SELECT
    ON TABLE Student
    TO U1;
  
```

例 2：把对 Student 表和 Course 表的全部权限授予用户 U2 和 U3

答案：GRANT ALL PRIVILEGES

```

    GRANT ALL PRIVILEGES
    ON TABLE Student, Course
    TO U2, U3;
  
```

例 3：把对表 SC 的查询权限授予所有用户

答案：GRANT SELECT

```

    GRANT SELECT
    ON TABLE SC
    TO PUBLIC;
  
```

例 4：把查询 Student 表和修改学生学号的权限授给用户 U4

答案：GRANT UPDATE(Sno), SELECT

```

    GRANT UPDATE(Sno), SELECT
    ON TABLE Student
    TO U4;
  
```

注：对属性列的授权时必须明确指出相应属性列名。

例 5：把对表 SC 的 INSERT 权限授予 U5 用户，并允许他再将此权限授予其他用户。

配套课程 习题答案



答案: GRANT INSERT

```
ON TABLE SC
TO U5
WITH GRANT OPTION;
```

执行例 5 后, U5 不仅拥有了对表 SC 的 INSERT 权限, 还可以传播此权限。

例 6: 对表 SC 的 INSERT 授予 U6 用户, 并允许他再将此权限授予其他用户。

答案: GRANT INSERT

```
ON TABLE SC
TO U6
WITH GRANT OPTION;
```

例 7: 对表 SC 的 INSERT 授予 U7 用户。

答案: GRANT INSERT

```
ON TABLE SC
TO U7;
```

但 U7 不能再传播此权限, 因为没有写这条语句 WITH GRANT OPTION。

经过执行如上操作, 例 1-例 7 后学生-课程数据库的用户权限定义如下:

授权用户名	被授权用户名	数据库对象名	允许的操作类型	能否转授权
DBA	U1	关系 Student	SELECT	不能
DBA	U2	关系 Student	ALL	不能
DBA	U2	关系 Course	ALL	不能
DBA	U3	关系 Student	ALL	不能
DBA	U3	关系 Course	ALL	不能
DBA	PUBLIC	关系 SC	SELECT	不能
DBA	U4	关系 Student	SELECT	不能
DBA	U4	关系 Student. Sno	UPDATE	不能
DBA	U5	关系 SC	INSERT	能
U5	U6	关系 SC	INSERT	能
U6	U7	关系 SC	INSERT	不能

题 2: 把对关系 SPJ 的属性 QTY 的修改权授予用户李勇的 T-SQL 语句是 ()

- A、GRANT QTY ON SPJ TO '李勇'
- B、GRANT UPDATE(QTY) ON SPJ TO '李勇'
- C、GRANT UPDATE(QTY) ON SPJ TO 李勇
- D、GRANT UPDATE ON SPJ(QTY) TO 李勇

答案: C

配套课程 习题答案



解析: GRANT XXX ON 表名 TO 用户

题 3: 今有两个关系模式:

教师 (教工号, 姓名, 职称, 职务, 工资)

系别 (系别号, 系名, 办公地址, 电话号)

请用 SQL 的 GRANT 语句完成以下授权定义或者存取控制功能:

(1) 用户李勇对教师表有 SELECT 权利, 对工资字段有更新权利。

答案: GRANT SELECT,UPDATE(工资) ON 教师 TO 李勇;

2) REVOKE (收回权限) 语句:

授予的权限可以由 DBA 或其他授权者用 REVOKE 语句收回。

REVOKE 语句的一般格式为:

REVOKE <权限>[,<权限>]...

[ON <对象类型> <对象名>]

FROM <用户>[,<用户>]...;

例 8: 把用户 U4 修改学生学号的权限收回

答案: REVOKE UPDATE(Sno) ON TABLE Student FROM U4;

例 9: 收回所有用户对表 SC 的查询权限

答案: REVOKE SELECT ON TABLE SC FROM PUBLIC;

例 10: 把用户 U5 对 SC 表的 INSERT 权限收回

答案: REVOKE INSERT ON TABLE SC FROM U5 CASCADE ;

将用户 U5 的 INSERT 权限收回的时候必须级联 (CASCADE) 收回, 系统只收回直接或间接从 U5 处获得的权限, 学生-课程数据库中的用户权限定义如下。

授权用户名	被授权用户名	数据库对象名	允许的操作类型	能否转授权
DBA	U1	关系 Student	SELECT	不能
DBA	U2	关系 Student	ALL	不能
DBA	U2	关系 Course	ALL	不能
DBA	U3	关系 Student	ALL	不能
DBA	U3	关系 Course	ALL	不能
DBA	U4	关系 Student	SELECT	不能

SQL 灵活的授权机制

DBA:

配套课程 习题答案



拥有所有对象的所有权限

不同的权限授予不同的用户

用户：拥有自己建立的对象的全部的操作权限

GRANT：授予其他用户

被授权的用户：

“继续授权”（WITH GRANT OPTION）许可：可以再授予给其它用户

所有授予出去的权力在必要时又都可用 REVOKE 语句收回。

题 4：今有两个关系模式：

教师（教工号，姓名，职称，职务，工资）

系别（系别号，系名，办公地址，电话号）

请用 SQL 的 GRANT 语句完成以下授权定义或者存取控制功能：

（2）撤销用户李勇的上述权利。

答案：REVOKE SELECT,UPDATE(工资) ON 教师 FROM 李勇；

创建数据库模式的权限

DBA 在创建用户时实现

```
CREATE USER <username>
  [WITH] [DBA | RESOURCE | CONNECT] ;
```

拥有的权限	可否执行的操作			
	CREATE USER	CREATE SCHEMA	CREATE TABLE	登录数据库执行 数据查询和操纵
DBA	可以	可以	可以	可以
RESOURCE	不可以	不可以	不可以	不可以
CONNECT	不可以	不可以	不可以	可以，但必须拥有相应权限

数据库角色：被命名的一组与数据库操作相关的权限

角色是权限的集合，可以为一组具有相同权限的用户创建一个角色

简化授权的过程

角色的创建：CREATE ROLE <角色名>

给角色授权：

配套课程 习题答案



```
GRANT <权限> [, <权限>] ... ON <对象类型>对象名 TO <角色> [, <角色>] ...
```

将一个角色授予其他的角色或用户：

```
GRANT <角色 1> [, <角色 2>] ... TO <角色 3> [, <用户 1>] ... [WITH ADMIN OPTION] ;
```

角色权限的收回：

```
REVOKE <权限> [, <权限>] ... ON <对象类型> <对象名> FROM <角色> [, <角色>] ...
```

题 5: SQL 中一组具有相同权限的用户称为（ ）

答案：角色

解析：角色是权限的集合，可以为一组具有相同权限的用户创建一个角色。

题 6: DBMS 通常提供授权功能来控制不同用户访问数据的权限，这主要是为了实现数据库的（ ）。

A、可靠性 B、一致性 C、完整性 D、安全性

答案：D

解析：DBMS 对数据安全控制主要是通过存取控制来实现的，也就是规定不同用户对不同数据对象所允许执行的操作，并控制各用户只能存取权限所允许的数据。

例 11: 请写出完成如下操作的步骤：

创建一个角色 R1，并使 R1 拥有 Student 表的 SELECT、UPDATE、INSERT 权限，并将这个角色授予王平，张明，赵玲。

答案：步骤如下：

1、首先创建一个角色 R1

```
CREATE ROLE R1;
```

2、然后使用 GRANT 语句，使角色 R1 拥有 Student 表的 SELECT、UPDATE、INSERT 权限

```
GRANT SELECT, UPDATE, INSERT ON TABLE Student TO R1;
```

3、将这个角色授予王平，张明，赵玲。使他们具有角色 R1 所包含的全部权限

```
GRANT R1 TO 王平, 张明, 赵玲;
```

例 12: 从角色 R1 收回对 Student 表的查询权限

```
REVOKE SELECT ON TABLE Student FROM R1;
```

3、视图机制



把要保密的数据对无权存取这些数据的用户隐藏起来，对数据提供一定程度的安全保护。

主要功能是提供数据独立性，无法完全满足要求

间接实现了支持存取谓词的用户权限定义

例 13：建立计算机系学生的视图，把对该视图的 SELECT 权限授于王平，把该视图上的所有操作权限授于张明

答案：先建立计算机系学生的视图 CS_Student：

```
CREATE VIEW CS_Student AS SELECT * FROM Student WHERE Sdept='CS';
```

在视图上进一步定义存取权限：

```
GRANT SELECT ON CS_Student TO 王平 ;
```

```
GRANT ALL PRIVILEGES ON CS_Student TO 张明;
```

题 7：采用定义视图的机制在数据控制方面要解决的问题是（ ）。

A、数据安全性

B、数据完整性

C、数据库恢复

D、数据库并发控制

答案：A

解析：采用定义视图的机制在数据控制方面要解决的问题是数据安全性。



课时六 练习题

配套课程 习题答案



课时七 数据库完整性

考点	重要程度	占分	题型
1. 实体完整性	★★★★★	2~5	选择、填空、简答
2. 参照完整性	★★★★★	2~5	选择、填空、简答
3. 用户定义的完整性	★★★	2~5	选择、填空、简答
4. 触发器	★★★	5~10	选择、简答

数据库的完整性 (integrity) 是指数据的正确性 (correctness) 和相容性 (compatibility)。

完整性控制机制：

提供定义完整性约束条件的机制

提供完整性检查的方法

进行违约处理

关系数据库管理系统使得完整性控制成为其核心支持的功能，从而能够为所有用户和应用提供一致的数据库完整性。

1. 实体完整性

1) 定义实体完整性

关系模型的实体完整性在 CREATE TABLE 中用 PRIMARY KEY 定义的。对单属性构成的码有两种说明方法，一种是定义为列级约束条件。对多个属性构成的码只有一种说明方法，即定义为表级约束条件。

例 1：将 Student 表中的 Sno 属性定义为码。

解：CREATE TABLE Student

```
(Sno CHAR (9) PRIMARY KEY, /* 在列级定义主码 */
  Sname CHAR (20) NOT NULL,
  Ssex CHAR (2),
  Sage SMALLINT,
  Sdept CHAR (20) );
```

CREATE TABLE Student

(Sno CHAR (9) ,

配套课程 习题答案



```

Sname CHAR (20) NOT NULL,
Ssex CHAR (2) ,
Sage SMALLINT,
Sdept CHAR (20) ,
PRIMARY KEY (Sno) ;    /*在表级定义主码 */

```

例 2：将 SC 表中的 Sno, Cno 属性组定义为码（多属性构成的码）。

解：CREATE TABLE SC,

```

(Sno CHAR (9) ,
Cno CHAR (4) ,
Grade SMALLINT,
PRIMARY KEY (Sno, Cno) ;    /*只能在表级定义主码*/

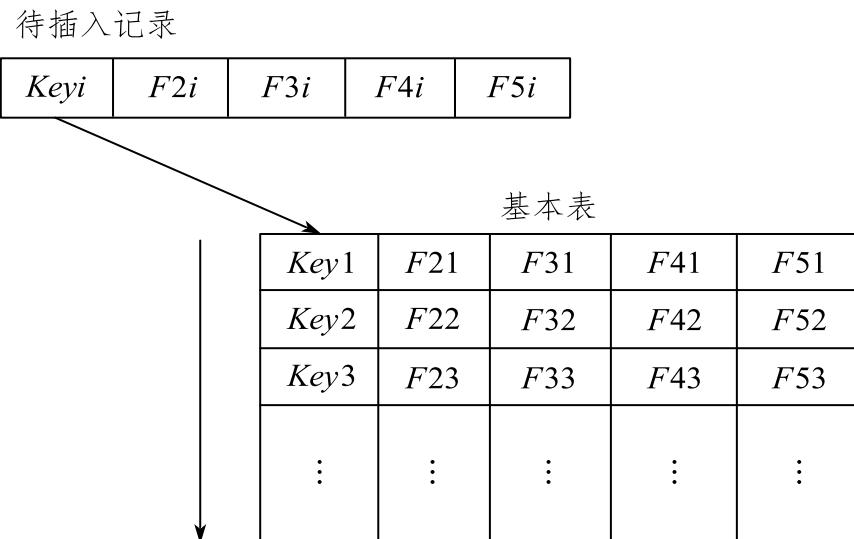
```

2) 实体完整性检查和违约处理

用 PRIMARY KEY 短语定义了关系的主码后，每当用户程序对基本表插入一条记录或对主码进行更新操作时，关系数据库自动进行检查：

- (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。

检查记录中主码值是否唯一的一种方法是进行全表扫描，依次判断表中每一条记录的主码值与将插入记录的主码值（或者修改的新主码值）是否相同。如下图所示。



全表扫描的缺点：十分耗时。

为了避免对基本表进行全表扫描，关系数据库管理系统一般都在主码上自动建立一个索引。

题 3：设属性 A 是关系 R 的主属性，则属性 A 不能取空值 (NULL)，这是（ ）。

- A. 实体完整性规则
- B. 参照完整性规则
- C. 用户定义完整性规则
- D. 域完整性规则

答案：A.

解析：实体完整性规则：这条规则要求关系中元组在组成主键的属性上不能有空值。如果出现空值，那么主键值就起不了唯一标识元组的作用。

2. 参照完整性

1) 定义参照完整性

关系模型的参照完整性在 CREATE TABLE 中用 FOREIGN KEY 短语定义哪些列为外码，用 REFERENCES 短语指明这些外码参照哪些表的主码。

例如，关系 SC 中一个元组表示一个学生选修的某门课程的成绩，(Sno, Cno) 是主码。Sno, Cno 分别参照 Student 表的主码和 Course 表的主码。

例 1：定义 SC 中的参照完整性。

解：CREATE TABLE SC

```
(Sno CHAR (9) ,
Cno CHAR (4) ,
Grade INT,
PRIMARY KEY (Sno, Cno), /*在表级定义实体完整性*/
FOREIGN KEY (Sno) REFERENCES Student (Sno) ,
/*表级定义参照完整性*/
FOREIGN KEY (Cno) REFERENCES Course (Cno)
);
```

2) 参照完整性检查和违约处理

一个参照完整性将两个表中的相应元组联系起来了。因此，对被参照表和参照表进行增删改操作时有可能破坏参照完整性，必须进行检查。



对表 SC 和 Student 有四种可能破坏参照完整性的情况，如下：

- (1) SC 表中增加一个元组，该元组的 Sno 属性的值在 Student 表中找不到一个元组，其 Sno 属性的值与之相等
- (2) 修改 SC 表中的一个元组，修改后该元组的 Sno 属性的值在 Student 表中找不到一个元组，其 Sno 属性的值与之相等
- (3) 从 Student 表中删除一个元组，造成 SC 表中某些元组的 Sno 属性的值在表 Student 中找不到一个元组，其 Sno 属性的值与之相等
- (4) 修改 Student 表中一个元组的 Sno 属性，造成 SC 表中某些元组的 Sno 属性的值在表 Student 中找不到一个元组，其 Sno 属性的值与之相等。

3) 参照完整性违约处理

- (1) 拒绝 (NO ACTION) 执行：不允许该操作执行。该策略一般设置为默认策略
- (2) 级联 (CASCADE) 操作：当删除或修改被参照表 (Student) 的一个元组造成了与参照表 (SC) 的不一致，则删除或修改参照表中所有造成不一致的元组。
- (3) 设置为空值 (SET-NULL)：当删除或修改被参照表的一个元组时造成了不一致，则将参照表中的所有造成不一致的元组的对应属性设置为空值。

例 2：显式说明参照完整性的违约处理示例。

解：CREATE TABLE SC

```
(Sno CHAR (9) NOT NULL,  
Cno CHAR (4) NOT NULL,  
Grade SMALLINT,  
PRIMARY KEY (Sno, Cno) ,  
/*在表级定义实体完整性，Sno, Cno 都不能取空值*/  
FOREIGN KEY (Sno) REFERENCES Student (Sno)  
/*在表级定义参照完整性*/  
ON DELETE CASCADE /*级联删除 SC 表中相应的元组*/  
ON UPDATE CASCADE, /*级联更新 SC 表中相应的元组*/  
FOREIGN KEY (Cno) REFERENCES Course (Cno)  
/*在表级定义参照完整性*/  
ON DELETE NO ACTION
```

配套课程 习题答案



```

/*当删除 course 表中的元组造成与 SC 表不一致时拒绝删除*/
ON UPDATE CASCADE
/*当更新 course 表中的 cno 时，级联更新 SC 表中相应的元组*/ ) ;

```

题 3：在关系数据库中，表与表之间的联系是通过（ ）实现。

- A. 实体完整性规则 B. 参照完整性规则
 C. 用户自定义完整性规则 D. 主键

答案：B.

解析：一个参照完整性将两个表中的相应元组联系起来了。因此，对被参照表和参照表进行增删改操作时有可能破坏参照完整性，必须进行检查。

题 4：试述关系模型的参照完整性规则？

解：参照完整性是关系模型的完整约束之一，属于数据完整性的一种，其余还有：实体完整性、用户自定义完整性。

参照完整性规则：若属性或属性组 F 是基本关系 R 的外键，它与基本关系 S 的主键 K_S 相对应（基本关系 R 和 S 不一定是不同的关系），则对于 R 中的每个元组在 F 上的值必须为：

- (1) 空值，F 的每个属性值均为空值。
 (2) S 中某个元组中的主键值（主码值）。

即参照的关系中的属性值必须能够在被参照关系找到或者取空值，否则不符合数据库的语义。在实际操作时如更新、删除、插入一个表中的数据，通过参照引用相互关联的另一个表中的数据，来检查对表的数据操作是否正确，不正确则拒绝操作。

3. 用户定义的完整性

用户定义的完整性是：针对某一具体应用的数据必须满足的语义要求。关系数据库管理系统提供了定义和检验用户定义完整性的机制，不必由应用程序承担。

1) 属性上约束条件的定义

在 CREATE TABLE 中定义属性的同时，可根据应用要求定义属性上的约束条件，即属性值限制。包括：

列值非空 (NOT NULL)；

配套课程 习题答案



列值唯一 (UNIQUE) ;

检查列值是否满足一个条件表达式 (CHECK 短语)。

(1) 不允许取空值

例 1: 在定义 SC 表时, 说明 Sno、Cno、Grade 属性不允许取空值。

解: CREATE TABLE SC (

Sno CHAR (9) NOT NULL,

Cno CHAR (4) NOT NULL,

Grade SMALLINT NOT NULL,

PRIMARY KEY (Sno, Cno), /*在表级定义实体完整性, 隐含了 Sno, Cno 不允许取空值, 在列级不允许取空值的定义可不写 */);

(2) 列值唯一

例 2: 建立部门表 DEPT, 要求部门名称 Dname 列取值唯一, 部门编号 Deptno 列为主码。

解: CREATE TABLE DEPT

(Deptno NUMERIC (2) ,

Dname CHAR (9) UNIQUE NOT NULL, /*要求 Dname 列值唯一, 并且不能取空值*/

Location CHAR (10) ,

PRIMARY KEY (Deptno)

) ;

(3) 用 CHECK 短语指定列值应该满足的条件

例 3: Student 表的 Ssex 只允许取“男”或“女”。

解: CREATE TABLE Student

(Sno CHAR (9) PRIMARY KEY,

Sname CHAR (8) NOT NULL,

Ssex CHAR (2) CHECK (Ssex IN ('男', '女')), /*性别属性 Ssex 只允许取'男'或'女' */

Sage SMALLINT,

Sdept CHAR (20)

) ;

配套课程 习题答案



例 4: SC 表的 Grade 的值应该在 0 和 100 之间。

解: CREATE TABLE SC

```
(Sno CHAR (9) ,
Cno CHAR (4) ,
Grade SMALLINT CHECK (Grade>=0 AND Grade<=100) , /*Grade 取值范围是 0
到 100*/
PRIMARY KEY (Sno, Cno) ,
FOREIGN KEY (Sno) REFERENCES Student (Sno) ,
FOREIGN KEY (Cno) REFERENCES Course (Cno)
);
```

2) 属性上约束条件的检查和违约处理

当往表中插入元组或修改属性的值时, 关系数据库管理系统将检查属性上的约束条件是否被满足, 如果不满足则操作被拒绝执行。

3) 元组上的约束条件

(1) 元组上约束条件的定义

在 CREATE TABLE 语句中可以用 CHECK 短语定义元组上的约束条件, 即元组级的限制。

同属性值限制相比, 元组级的限制可以设置不同属性之间的取值的相互约束条件。

例 5: 当学生的性别是男时, 其名字不能以 Ms. 打头。

解: CREATE TABLE Student

```
(Sno CHAR (9) ,
Sname CHAR (8) NOT NULL,
Ssex CHAR (2) ,
Sage SMALLINT,
Sdept CHAR (20) ,
PRIMARY KEY (Sno) ,
CHECK (Ssex='女' OR Sname NOT LIKE 'Ms.%')
); /*定义了元组中 Sname 和 Ssex 两个属性值之间的约束条件*/
```

配套课程 习题答案



(2) 元组上约束条件的检查和违约处理

当往表中插入元组或修改属性的值时，关系数据库管理系统将检查元组上的约束条件是否被满足，如果不满足则操作被拒绝执行。

完整性约束命名子句

完整性约束命名子句：

CONSTRAINT <完整性约束条件名> <完整性约束条件>

<完整性约束条件> 包括 NOT NULL、UNIQUE、PRIMARY KEY、FOREIGN KEY、CHECK 短语等。

例 6：建立学生登记表 Student，要求学号在90000~99999之间，姓名不能取空值，年龄小于30，性别只能是“男”或“女”。

解：CREATE TABLE Student

```
(Sno NUMERIC (6)
CONSTRAINT C1 CHECK (Sno BETWEEN 90000 AND 99999) ,
Sname CHAR (20)
CONSTRAINT C2 NOT NULL,
Sage NUMERIC (3)
CONSTRAINT C3 CHECK (Sage<30) ,
Ssex CHAR (2)
CONSTRAINT C4 CHECK (Ssex IN ('男', '女')) ,
CONSTRAINT StudentKey PRIMARY KEY (Sno)
) ;
```

例 7：建立教师表 TEACHER，要求每个教师的应发工资不低于3000元。应发工资是工资列

Sal 与扣除项 Deduct 之和。

解：CREATE TABLE TEACHER

```
(Eno NUMERIC (4) PRIMARY KEY      /*在列级定义主码*/
Ename CHAR (10) ,
Job CHAR (8) ,
Sal NUMERIC (7, 2) ,
Deduct NUMERIC (7, 2) ,
Deptno NUMERIC (2) ,
```

配套课程 习题答案



```

CONSTRAINT TEACHERFKey FOREIGN KEY (Deptno)
  REFERENCES DEPT (Deptno) ,
CONSTRAINT C1 CHECK (Sal + Deduct >= 3000)
) ;

```

修改表中的完整性限制

使用 ALTER TABLE 语句修改表中的完整性限制。

例 8：去掉例 6 中 Student 表中对性别的限制。

解：ALTER TABLE Student DROP CONSTRAINT C4。

例 9：修改例 6 中 Student 表中的约束条件，要求学号改为在 900000~999999 之间，年龄由小于 30 改为小于 40。

解：/* 可以先删除原来的约束条件，再增加新的约束条件。 */

```

ALTER TABLE Student
  DROP CONSTRAINT C1; ALTER TABLE Student
  ADD CONSTRAINT C1 CHECK (Sno BETWEEN 900000 AND 999999) ;
ALTER TABLE Student
  DROP CONSTRAINT C3; ALTER TABLE Student
  ADD CONSTRAINT C3 CHECK (Sage < 40) ;

```

题 10：关系的完整性包括：_____、_____、用户自定义完整性和域完整性。

答案：实体完整性、参照完整性。

题 11：有一个关系，学生（学号，姓名，系列），规定学号的值域的 8 个数字组成的字符串，这一规则属于（ ）。

- | | |
|---------------|-------------|
| A. 实体完整性约束 | B. 参照完整性约束 |
| C. 用户自定义完整性约束 | D. 关键字完整性约束 |

答案：C.

解析：用户定义完整性：由用户根据实际情况，对数据库中数据的内容所作的规定称为用户



定义的完整性规则。通过这些限制数据库中接受符合完整性约束条件的数据值，不接受违反约束条件的数据，从而保证数据库的数据合理可靠。

4. 触发器

触发器是用户定义在关系表上的一类由事件驱动的特殊过程，这些事件包括 INSERT、UPDATE、DELETE 事件。

1) 定义触发器

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 (如对一个表的增、删、改操作，事务的结束等) 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，可以涉及其他表和其他数据库对象，通常是一段 SQL 存储过程。

SQL 使用 CREATE TRIGGER 语句定义触发器，其格式如下：

```
CREATE TRIGGER <触发器名> /*每当触发事件发生时，该触发器被激活*/  
{BEFORE | AFTER} <触发事件> ON<表名> /*指明触发器激活的时间是在执行触发事件  
前或后*/
```

```
REFERENCING NEWOLD ROW AS<变量> /*REFERENCING 指出引用的变量*/  
FOR EACH{ROW | STATEMENT} /*定义触发器的类型，指明动作体执行的频率*/  
[WHEN <触发条件>] <触发动作体> /*仅当触发条件为真时才执行触发动作体*/
```

触发器名：可以包含模式名，也可以不包含；同一模式下，触发器名必须是唯一的，并且触发器名和表名必须在同一模式下。

表名：触发器只能定义在表上，不可在视图上。当基本表的数据发生变化时，将激活定义在该表上相应触发事件的触发器。

触发事件：可以是 INSERT、DELETE 或 UPDATE，也可以是它们的组合；同样也可以 UPDATE OF <列名, ..., >，也即进一步指明哪些列变化时需要激活触发器；**AFTER/BEFORE**是触发时机。

触发器类型：触发器按照所触发动作的间隔尺寸可以分为行级触发器 (FOR EACH ROW) 和语句级触发器 (FOR EACH STATEMENT)；语句级会执行一次，行级执行的次数以表的具体行数而定。



触发条件：触发器被激活时，只有当触发条件为真时触发动作体才执行，否则触发动作体不执行；如果省略 WHEN 触发条件，则触发动作体在触发器激活后立即执行。

触发动作体：触发动作体既可以是一个匿名 PL/SQL 过程块，也可以是对已创建存储过程的调用。如果是行级触发器，用户可以在过程体中使用 NEW 和 OLD 引用 UPDATE/INSERT 事件之后的新值和 UPDATE/DELETE 事件之前的旧值；如果是语句级触发器，则不能在触发动作体中使用 NEW 或 OLD 进行引用。如果触发动作体执行失败，激活触发器的事件（即对数据库的增、删、改操作）就会终止执行，触发器的目标表或触发器可能影响的其他对象不发生任何变化。

例 1：当对 SC 表的 Grade 属性进行修改时，若分数增加了 10%，则将此次操作记录到另一个表 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。

解：CREATE TRIGGER SC_T //触发器名字
 //在对 sc 表的 Grade 更新后再触发
 AFTER OF Grade ON SC
 REFERENCING
 OLDROW AS OldTuple
 NEWROW AS NewTuple
 FOR EACH ROW //行级触发器，也即每更新一次，下面规则就会执行一次
 WHEN (NewTuple.Grade >= 1.1*OldTuple.Grade) //触发条件为真才会执行
 INSERT INTO SC_U (Sno, Cno, OldGrade, NewGrade)
 VALUES (OldTuple.Sno, OldTuple.Cno, OldTuple.Grade, NewTuple.Grade)

在本例中 REFERENCING 指出引用的变量，如果触发事件是 UPDATE 操作并且有 FOR EACH ROW 子句，则可以引用的变量有 OLDROW 和 NEWROW，分别表示修改之前的元组和修改之后的元，若没有 FOR EACH ROW 子句，则可以引用的变量有 OLDTABLE 和 NEW TABLE，OLDTABLE 表示表中原来的内容，NEWTABLE 表示表中变化后的部分。

2) 激活触发器

触发器的执行是由触发器事件激活的，如果同一个表上有多个触发器，激活时会按照以下顺序执行：

- (1) 执行该表上的 BEFORE 触发器
- (2) 激活触发器的 SQL 语句
- (3) 执行该表上的 AFTER 触发器

配套课程 习题答案



对于同一个表上的多个 BEFORE (AFTER) 触发器，遵循“谁先创建谁先执行”的原则，即按照触发器创建的时间先后顺序执行

3) 删除触发器

语法：

```
DROP TRIGGER<触发器名> ON <表名>;
```

题 2：设有一个顾客商品关系数据库，有三个基本表，表结构如下：

商品表：Article (商品号, 商品名, 单价, 库存量)

客户表：Customer (顾客号, 顾客名, 性别, 年龄, 电话)

订单表：OrderItem (顾客号, 商品号, 数量, 购买价, 日期)

创建一个删除顾客信息的触发器，当存在订购信息时不允许删除。

解：CREATE TRIGGER C_DELETE ON Customer FOR DELETE

```
AS IF (SELECT COUNT (*)
```

```
FROM OrderItem 0, deleted
```

```
WHERE 0. 顾客号=deleted. 顾客号) >0
```

```
ROLLBACK TRANSACTION
```

```
ELSE
```

```
DELETE Customer
```

```
FROM Customer, deleted
```

```
WHERE Customer. 顾客号 = deleted. 顾客号
```



课时七 练习题

1. 若属性 A 是基本关系 R 的主属性，则属性 A 不能取空值，这是（ ）规则。
A. 参照完整性 *B. 用户定义完整性*
C. 实体完整性 *D. 主码不能取空值*

2. 在 SQL 语言中 PRIMARY KEY 的作用是（ ）。
A. 定义主码 *B. 定义外部码*
C. 定义外部码的参照表 *D. 确定主码类型*

3. 实体完整性要求主属性值不能取空值，这一点可以通过（ ）来保证。
A. 定义外码 *B. 定义主码*
C. 用户定义的完整性 *D. 由关系系统自动*

4. 在创建谋数据库表时，给表指定了主索引。该主索引可以实现数据库完整性中的（ ）。
A. 参照完整性 *B. 域完整性*
C. 实体完整性 *D. 用户自定义完整性*

5. 通过 T-SQL 语句修改表约束，在表 employee 加入 CEHCK 约束：输入的员工编号必须为 E 开头的 5 位数编号，性别只能为 M/F。

6. 创建一个触发器，实现级联更新：当更新 employee 表中的 emp_no 列的值时，同时更新 sales 表中的 sale_id 列的值，并且只更新一次。

7. 举例说明关系参照完整性的含义。



课时八 关系数据理论

考点	重要程度	占分	常用题型
1. 函数依赖	★★★	2~5	选择、填空、简答
2. 码	★★★	2~5	选择、填空、简答
3. 函数范式（范式、2NF 3NF、BCNF）	★★★★	2~10	选择、填空、简答
4. 多值依赖	★★★	2~5	选择、填空、简答

1. 函数依赖

1) 定义：设 $R(U)$ 是属性集 U 上的关系模式， X, Y 是 U 的子集。若对于 $R(U)$ 的任意一个看人能的关系 r ， r 中不可能存在两个元组 X 上的属性值相等，而在 Y 上的属性值不相等，则称 X 函数确定 Y 或 Y 函数依赖于 X ，记作 $X \rightarrow Y$ 。

注：函数依赖不是指关系模式 R 中的某个或某些关系满足的约束条件，而是指 R 的一切关系均要满足的约束条件。

其中：

$X \rightarrow Y$ ，但 $Y \not\subseteq X$ ，则称 $X \rightarrow Y$ 是非平凡的函数依赖。

$X \rightarrow Y$ ，但 $Y \subseteq X$ ，则称 $X \rightarrow Y$ 是平凡的函数依赖。（对于任意的关系模式，平凡的函数依赖是必然成立的）

若 $X \rightarrow Y$ ，则 X 称为这个函数依赖的决定属性组，也称为决定因素。

若 $X \rightarrow Y$ ， $Y \rightarrow X$ ，则记作 $X \leftrightarrow Y$ 。

在 $R(U)$ 中，如果 $X \rightarrow Y$ ，并且对于 X 的任何一个真子集 X' ，都有 $X' \not\rightarrow Y$ ，则称 Y 对 X 完全函数依赖。若 $X \rightarrow Y$ ，但 Y 不完全依赖于 X ，则称 Y 对 X 部分函数依赖。

在 $R(U)$ 中，如果 $X \rightarrow Y (Y \not\subseteq X)$ ， $Y \not\rightarrow X$ ， $Y \rightarrow Z$ ， $Z \not\subseteq Y$ ，则称 Z 对 X 传递函数依赖。

题 1：在函数依赖，学号 → 姓名中，() 是决定因素。

答案：学号。

解析： X 函数确定 Y 或 Y 函数依赖于 X ，记作 $X \rightarrow Y$ ，题中 $学号 \rightarrow 姓名$ ，学号则是决定因素

2. 码

码是关系模式中的一个重要概念。

配套课程 习题答案



定义：设 K 为 $R(U, F)$ 中的属性的集合，若 U 对 K 完全函数依赖，则 K 为 R 的候选码。

若候选码多于一个，则选定其中的一个为主码。包含在任何一个候选码中的属性称为主属性；不包含在任何一个候选码中的属性称为非主属性或者非码属性。最简单的情况，单个属性是码。最极端的情况，整个属性组是码，称为全码。主码或者候选码都简称码。

定义：关系模式 R 中的属性或属性组 X 并非 R 的码，但 X 是另一个关系模式的码，则称 X 是 R 的外部码，也称外码。

题 1. 设有关系 R 和函数依赖 F ： $R(A, B, C, D, E), F = \{ABC \rightarrow DE, BC \rightarrow D, D \rightarrow E\}$ 试求关系

R 的候选码是什么？

答案：关系 R 的候选码是 (A, B, C) 。

解析： $L: \{A, B, C\}$

$R: \{E\}$

$LR: \{D\}$

故： (A, B, C) 为候选码。

题 2. 在数据库中，码（又称为关键字、主键），候选码是关系的一个或一组属性，它的值能唯一地标识一个_____。

答案：元组

解析：元组：表中的行；元组的集合构成关系；每个元组就是一个记录。

3. 函数范式

1) 范式

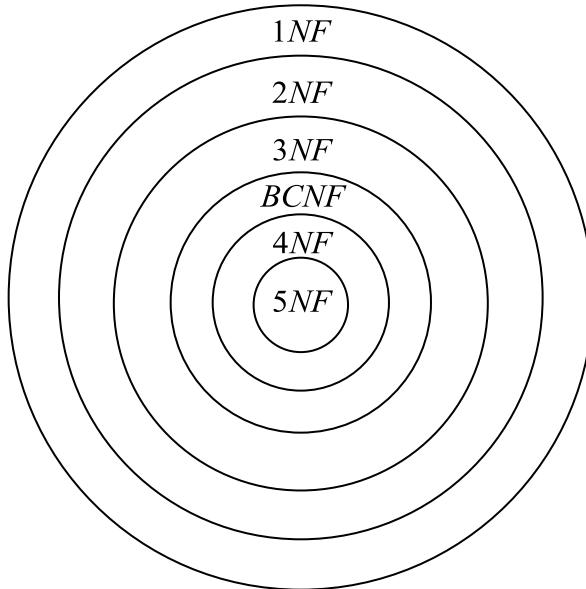
关系数据库中的关系是要满足一定要求的，满足不同程度要求的为不同范式。

所谓“第几范式”原本是表示关系的某一级别，所以常称某一种级别的关系模式 R 为第几范式。现在则把范式这个概念理解成符合某一种级别的关系模式的集合，则 R 为第几范式就可以写成 $R \in xNF$ 。

各种范式之间的关系：

对于各种范式之间的关系有 $5NF \subset 4NF \subset BCNF \subset 3NF \subset 2NF \subset 1NF$ ，如图所示





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高一级范式的关系模式的集合，这个过程就叫做规范化。

题 1. 设有关系 $R(S1, S2, X1, X2)$ ，其主码为 $(S1, S2)$ ，函数依赖关系集为

$\{S1 \rightarrow X1, (S1, S2) \rightarrow X2\}$ ，则此关系满足_____范式要求。

答案：第一

2) 2NF

定义：若 $R \subset 1NF$ ，且每一个非主属性完全函数依赖于任何一个候选码，则 $R \subset 2NF$ 。

题 2. 设有关系 $R(S1, S2, X1, X2)$ ，其主码为 $(S1, S2)$ ，函数依赖关系集为

$\{S1 \rightarrow X1, (S1, S2) \rightarrow X2\}$ 可分解为主码为 $(S1, S2)$ 的关系 $R1(S1, S2, X2)$ 和 _____ 两个

满足更高一级范式要求的范式。

答案： $R2(S1, X1)$

解析：根据第二范式的定义，每一个非主属性完全函数依赖于任何一个候选码，则 $R \subset 2NF$ ，则将其分为 $R1(S1, S2, X2)$ 和 $R2(S1, X1)$ 满足第二范式的要求。

3) 3NF

定义：设关系模式 $R < U, F > \in 1NF$ ，若 R 中不存在这样的码 X ，属性组 Y 及非主属性 $Z (Z \not\subseteq Y)$ ，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传递依赖于码，也不部分依赖于码。

题 3. 若要求分解保持函数依赖，那么模式分解一定能够达到（ ）

- A. 2NF B. 3NF C. BCNF D. 1NF

配套课程 习题答案



答案：B

解析：根据3NF定义，若要求分解保持函数依赖，则每一个非主属性既不传递依赖于码，也不部分依赖于码。

4) BCNF

定义：关系模式 $R \subset U, F \supseteq 1NF$ ，若 $X \rightarrow Y$ 且 $Y \not\subset X$ 时 X 必包含码，则 $R \subset U, F \supseteq BCNF$

题 4. 关系模式 $SJP \in 3NF$ ， S 是学生， J 是课程， P 是名次。每一个学生选修每门课程的成绩有一定的名次，每门课程中每一名次只有一个学生（假设没有并列名次的情况）。由语义得到下面的函数依赖： $(S, J) \rightarrow P, (J, P) \rightarrow S$

所以 (S, P) (J, P) 都可以作为候选码。这两个码由两个属性组成，而且它们是相交的。

这个关系模式中显然没有属性对码的传递依赖和部分依赖，所以 $SJP \in 3NF$ ，而且除 (S, P) 与 (J, P) 以外没有其他决定因素，所以 $SJP \in BCNF$ 。

题 5. 关系模式 $STJ(S, T, J)$ 中， S 表示学生， T 表示老师， J 表示课程。每一教师只教一门课程，每门课有若干老师，某一学生选定某门课，就对应一个固定的老师。有语义可得到如下函数依赖： $(S, J) \rightarrow T, (S, T) \rightarrow J, T \rightarrow J$ ， SJT 是第三范式，没有任何非主属性对码的部分函数依赖和传递函数依赖；但是不是 $BCNF$ ，因为 T 也是决定因素，但是 T 不包含码。

题 6. 关系模式由 $3NF$ 转换为 $BCNF$ 是消除了主属性对码的_____和_____。

答案：部分函数依赖、传递函数依赖

4. 多值依赖

题 1. 学校中某一门课程由多个教师讲授，他们使用相同的一套参考书，每个教师可以讲授多门课程，每种参考书可以提供多门课程使用，可以用一个非规范化的关系来表示教师 T 、课程 C 和参考书 B 之间的关系。如下图所示：

非规范化关系示例

课程 C	教师 T	参考书 B
物理	杜霄霖 王军	普通物理学 光学原理 物理习题集
数学	杜霄霖 张平	数学分析 微分方程 高等代数
计算数学	张平	数学分析

配套课程 习题答案



	周峰	
...

把这张表变成一个规范化的二维表：

课程 C	教师 T	参考书 B
物理	杜雷霖	普通物理学
物理	杜雷霖	光学原理
物理	杜雷霖	物理习题集
物理	王军	普通物理学
物理	王军	光学原理
物理	王军	物理习题集
数学	杜雷霖	数学分析
数学	杜雷霖	微分方程
数学	杜雷霖	高等代数
数学	张平	数学分析
数学	张平	微分方程
数学	张平	高等代数
...

关系模型 $\text{Teaching}(C, T, B)$ 的码是 (C, T, B) ，即全码。因而 $\text{Teaching} \in BCNF$ 。但是当某一个课程（如物理）增加一名讲师（如王磊）时，必须插入多个元组：（物理，王磊，普通物理学），（物理，王磊，光学原理），（物理，王磊，物理习题集）

因而对数据的增删改很不方便，数据冗余也十分明显。仔细考察这关系模式发现它具有一种称之为多值依赖的数据依赖。

定义：设 $R(U)$ 是属性集 U 上的一个关系模式。 X, Y, Z 是 U 的子集，并且 $Z = U - X - Y$ 。关系模式 $R(U)$ 中的多值依赖 $X \rightarrow Y$ 成立，当且仅当对 $R(U)$ 的任一关系 r ，给定一对 (x, z) 的值，有一组 y 的值，这组值仅仅取决于 x 的值而与 z 的值无关。

WSC 表

W	S	C
W1	S1	C1
W1	S1	C2
W1	S1	C3
W1	S2	C1
W1	S2	C2
W1	S2	C3
W2	S3	C4
W2	S3	C5

配套课程 习题答案



W2	S4	C4
W2	S4	C5

1) 判断多值依赖:

S 的值相同, 把两个 C 值不同的换行; 换行后新行仍在关系中, 则 $W \rightarrow S$ 。

2) 多值依赖具有以下性质:

多值依赖具有对称性。

多值依赖具有传递性

函数依赖可以看作多值依赖的特殊情况:

若 $X \twoheadrightarrow Y$, $X \twoheadrightarrow Z$, 则 $X \twoheadrightarrow YZ$

若 $X \twoheadrightarrow Y$, $X \twoheadrightarrow Z$, 则 $X \twoheadrightarrow Y \cap Z$

若 $X \twoheadrightarrow Y$, $X \twoheadrightarrow Z$, 则 $X \twoheadrightarrow Y - Z$, $X \twoheadrightarrow Z - Y$

题 1: 设有一个记录各个球队队员每场比赛进球数的关系模式 R (队员编号, 比赛场次, 进球数, 球队名, 队长名), 如果规定每个队员只能属于一个球队, 每个球队只有一个队长。

(1) 试写出关系模式 R 的基本 FD 和关键码。

(2) 说明 R 不是 $2NF$ 模式的理由, 并把 R 分解成 $2NF$ 模式集。

(3) 进而把 R 分解成 $3NF$ 模式集, 并说明理由。

解: (1) 根据每个队员只能属于一个球队, 可写出 FD : 队员编号 \rightarrow 球队名; 根据每个球队只有一个队长, 可写出 FD 球队名 \rightarrow 队长名;

“每个队员每场比赛只有一个进球数”, 这条规则也是成立的, 因此还可写 FD (队员编号, 比赛场次) \rightarrow 进球数。

从上述三个 FD 可知道, R 的码为 (队员编号, 比赛场次)。

(2) 从 (1) 可知, R 中存在下面两个 FD

(队员编号, 比赛场次) \rightarrow (球队名, 队长名)

队员编号 \rightarrow (球队名, 队长名)

显然, 其中第一个 FD 是一个局部依赖, 因此 R 不是 $2NF$ 模式。

对 R 应该进行分解, 由第二个 FD 的属性可构成一个模式, 即

$R1$ (队员编号, 球队名, 队长名);

另一个模式由 R 的属性集去掉第二个 FD 右边的属性组成, 即



$R2$ (队员编号, 比赛场次, 进球数)。

$R1$ 和 $R2$ 都是 $2NF$ 模式, 因此 $p = \{R1, R2\}$

(3) $r2$ (队员编号, 比赛场次, 进球数) 中, FD 是(队员编号, 比赛场次) \rightarrow 球数, 码为(队员编号, 比赛场次), 可见 $R2$ 已是 $3NF$ 模式。

$R1$ (队员编号, 球队名, 队长名) 中, FD 有两个: 队员编号 \rightarrow 球队名 球队名 \rightarrow 队长名
码为队员编号, 可见存在传递依赖, 因此 $R1$ 不是 $3NF$ 模式。对 $R1$ 应分解成两个模式: $R11$
(队员编号, 球队名), $R12$ (球队名, 队长名)。这两个模式都是 $3NF$ 模式。

因此, R 分解成 $3NF$ 模式集时, $p = \{R11, R12, R2\}$

配套课程 习题答案



课时八 练习题

1. 若关系模式 R 中只包含两个属性，则（ ）

- A. R 属于 $2NF$ ，但 R 不一定属于 $3NF$
- B. R 属于 $3NF$ ，但 R 不一定属于 $BCNF$
- C. R 属于 $BCNF$
- D. R 属于 $1NF$ ，但 R 不一定属于 $3NF$

2. 已知关系模式 $R(A, B, C, D, E)$ 及其上的函数相关性集合 $F = \{A \rightarrow D, B \rightarrow C, E \rightarrow A\}$ ，该关系模式的候选键是（ ）

- A. AB
- B. BE
- C. CD
- D. DE

3. 设学生关系 S (SNO , $SNAME$, $SSEX$, $SAGE$, $SDPART$) 的主键为 SNO ，学生选课关系 SC (SNO , CNO , $SCORE$) 的主键为 SNO 和 CNO ，则关系 R (SNO , CNO , $SSEX$, $SAGE$, $SDPART$, $SCORE$) 的主键为 SNO 和 CNO ，其满足（ ）。

- A. $1NF$
- B. $2NF$
- C. $3NF$
- D. $BCNF$

4. 设关系模式 $R(A, B, C, D)$ ， F 是 R 上的 FD 集， $F = \{AB \rightarrow C, D \rightarrow B\}$ ， R 的候选键为 _____。

5. 设关系模式 $R(A, B, C, D, E)$ ， F 是 R 上成立的 FD 集， $F = AB \rightarrow C$ ， $BC \rightarrow A$ ， $AC \rightarrow B$ ， $D \rightarrow E$ ， R 的候选键为（ ）。

- A. ABC, ACD, ACE
- B. ABD, BCD, ACD
- C. ABC, BCD, ACD
- D. ABD, ACD, ACE

6. 理解并给出下列术语的定义：

函数依赖、部分函数依赖、完全函数依赖、候选码、主码、外码、全码。

7. 设有一个记录各个球队队员每场比赛进球数的关系模式（队员编号，比赛场次，进球数，球队名，队长名），如果规定，每个队员只能属于一个球队，每个球队只有一个队长。

(1) 试写出关系模式 R 的基本函数依赖和主码。

(2) 说明 R 不是 $2NF$ 模式的理由，并把 R 分解成 $2NF$

(3) 进而将 R 分解成 $3NF$ ，并说明理由。

配套课程 习题答案



课时九 数据库设计

考点	重要程度	占分	题型
1. 数据库设计概述	★★★	2~4	选择、填空
2. 概念结构设计	★★★★★	5~10	简答
3. 逻辑结构设计	★★★★★	2~10	简答

1. 数据库设计概述

1) 数据库设计的特点

(1) 数据库建设的基本规律

设计特点之一：三分技术，七分管理，十二分基础数据。

管理：数据库建设项目管理、企业的业务管理

基础数据：数据的收集、整理、组织和更新。

(2) 结构设计和行为设计相结合

结构设计：即数据设计。

行为设计：即处理设计。

传统的软件工程：重视行为设计。

早期的数据库：重视结构设计。

2) 数据库设计的方法

(1) 手工试凑法

(2) 规范设计方法（手工设计方法）

基本思想：过程迭代和逐步求精

①新奥尔良（New Orleans）方法

②基于E-R模型的数据库设计方法

③3NF（第三范式）的设计方法

④面向对象的数据库设计方法

⑤统一建模语言（UML）方法

3) 数据库设计的基本步骤

数据库设计分为6个阶段：

配套课程 习题答案



需求分析

概念结构设计

逻辑结构设计

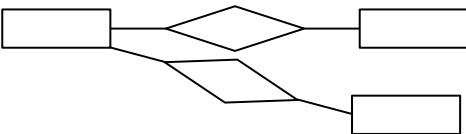
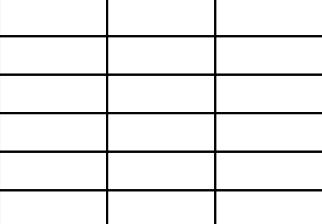
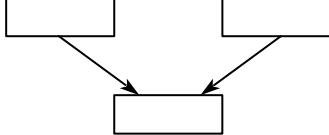
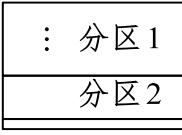
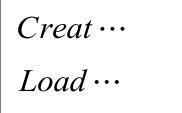
物理结构设计

数据库实施

数据库运行和维护

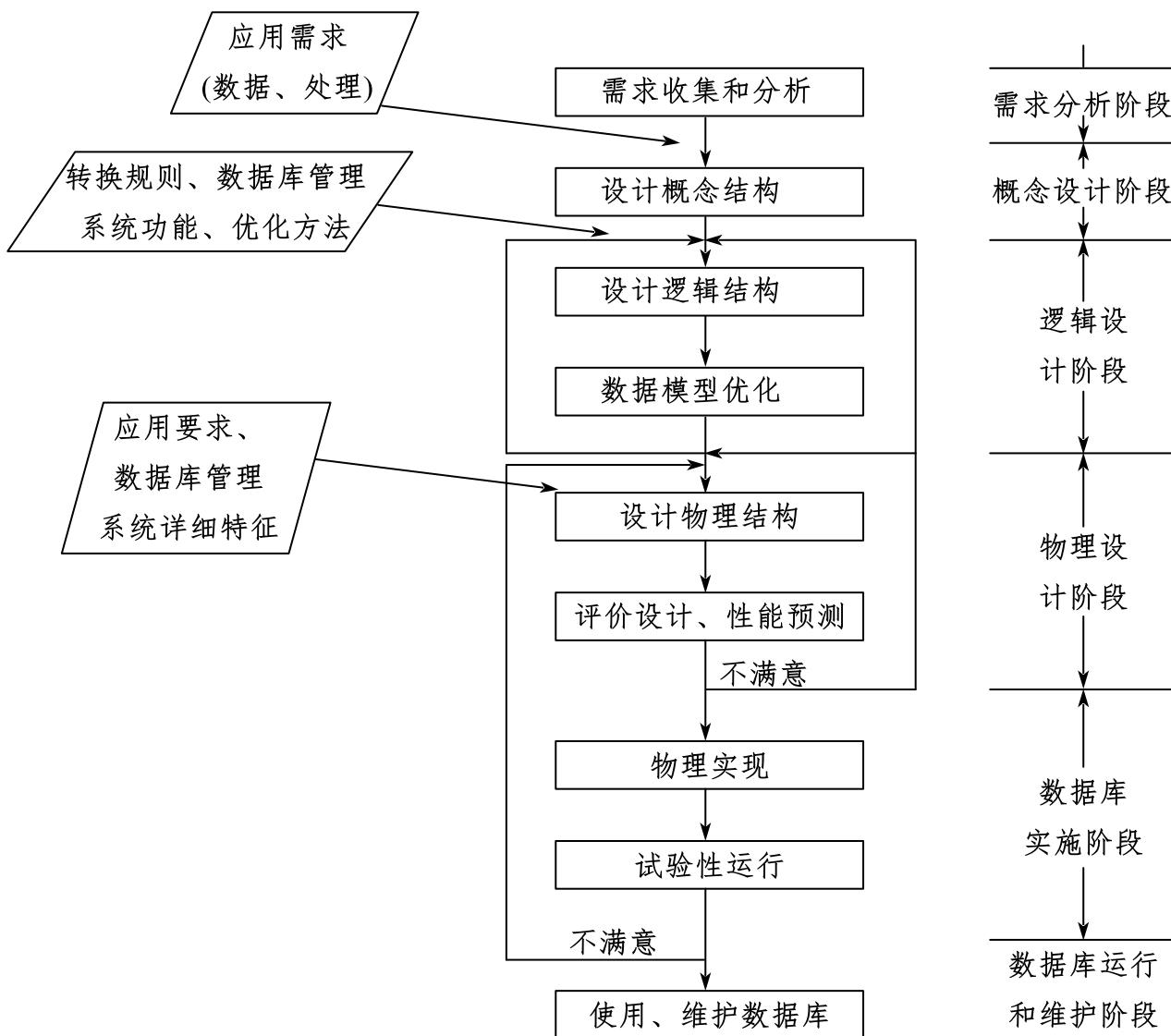
设计一个完善的数据库应用系统往往是上述6个阶段的不断反复。

这个设计步骤既是数据库设计的过程，也包括了数据库应用系统的设计过程，如下图所示。

设计阶段	设计描述	
需求分析	数字字典、全系统中数据项、数据结构、数据流、数据存储的描述	
概念结构设计	概念模型(<i>E-R</i> 图) 	数据字典
逻辑结构设计	某种数据模型 关系  非关系 	
物理结构设计	存储安排 存取方法选择 存取路径建立	
数据库实施	创建数据库模式 装入数据 数据库试运行	
数据库运行和维护	性能监测、转储/恢复、数据库重组和重构	

需求分析和概念设计独立于任何数据库管理系统，逻辑设计和物理设计与选用的数据库管理系统密切相关。



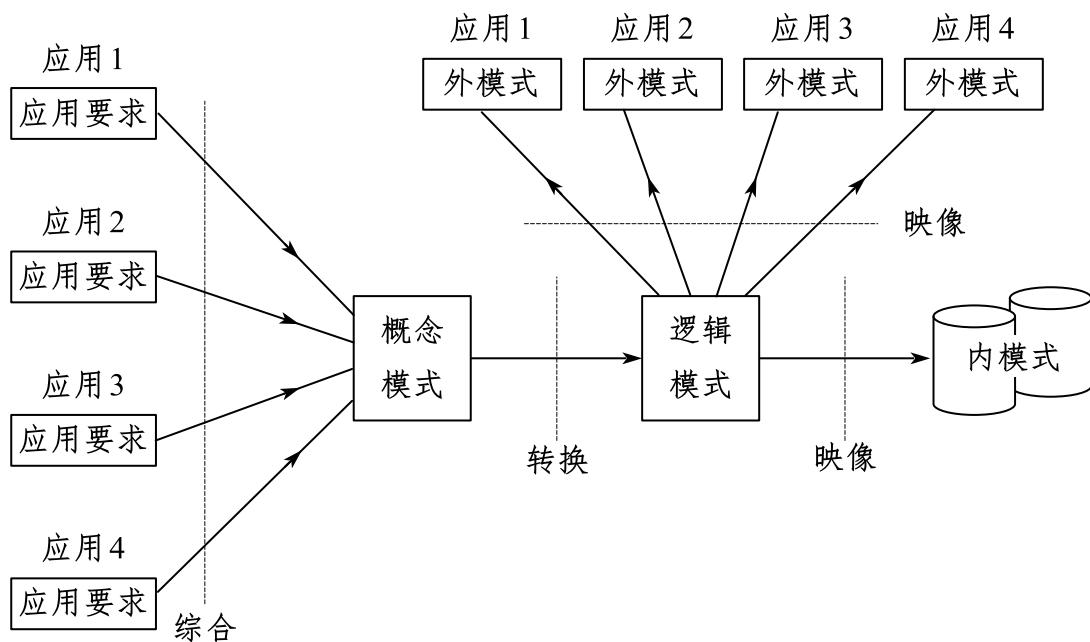


参加人员

- ①系统分析员和数据库设计人员
- ②数据库管理员和用户代表
- ③应用开发人员（程序员和操作员）

4) 数据库设计过程中的各级模式





需求分析阶段：

综合各个用户的应用需求

概念设计阶段：

形成独立于机器特点，独立于各个数据库管理系统产品的概念模型（E-R图）。

逻辑设计阶段：

①首先将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。

②然后根据用户处理的要求、安全性的考虑，在基础表的基础上再建立必要的视图view，形成外模式。

物理设计阶段：

根据数据库管理系统的特点和处理的需要，形成物理存储安排，建立索引，形成数据库的内模式。

2. 概念结构设计

1) 概念模型

将需求分析得到的用户需求抽象为信息结构（即概念模型）的过程就是概念结构设计，是整个数据库设计的关键。

描述概念模型的工具：E-R模型。



2) E-R 模型

(1) 实体之间的联系

两个实体型之间的联系

①一对联系 (1:1)

如果对于实体集 A 中的每一个实体，实体集 B 中至多有一个（也可以没有）实体与之联系，反之亦然，则称实体集 A 与实体集 B 具有一对一联系，记为 1:1。

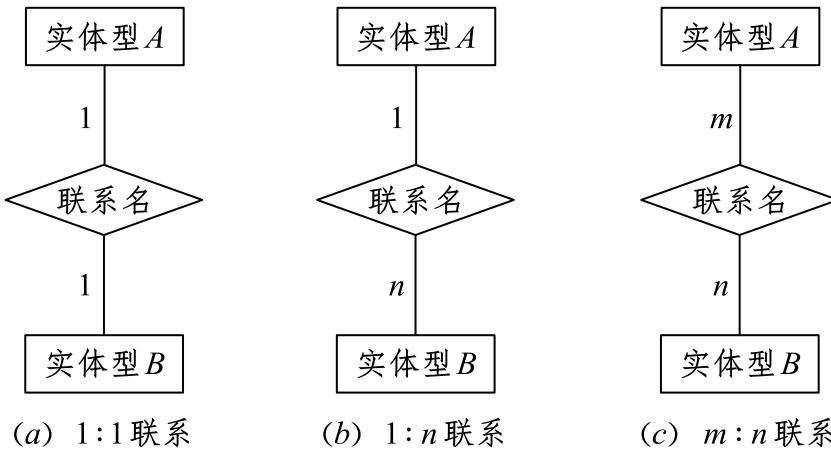
②一对多联系 (1:n)

如果对于实体集 A 中的每一个实体，实体集 B 中有 n 个实体 ($n \geq 0$) 与之联系，反之，对于实体集 B 中的每一个实体，实体集 A 中至多只有一个实体与之联系，则称实体集 A 与实体集 B 有一对多联系，记为 1:n。

③多对多联系 (m:n)

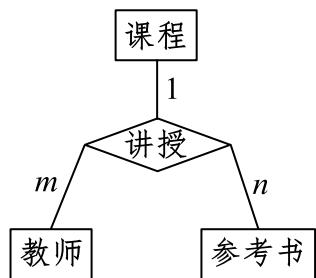
如果对于实体集 A 中的每一个实体，实体集 B 中有 n 个实体 ($n \geq 0$) 与之联系，反之，对于实体集 B 中的每一个实体，实体集 A 中也有 m 个实体 ($m \geq 0$) 与之联系，则称实体集 A 与实体集 B 具有多对多联系，记为 m:n。

两个以上的实体型之间的联系



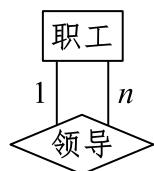
对于课程、教师与参考书3个实体型，如果一门课程可以有若干个教师讲授，使用若干本参考书，而每一个教师只讲授一门课程，每一本参考书只供一门课程使用，则课程与教师、参考书之间的联系是一对多的，如下图所示。





单个实体型内的联系

例如，职工实体型内部具有领导与被领导的联系，即某一职工（干部）“领导”若干名职工，而一个职工仅被另外一个职工直接领导，因此这是一对多的联系，如下图所示。



一般地，把参与联系的实体型的数目称为联系的度。

2个实体型之间的联系度为2，也称为二元联系；3个实体型之间的联系度为3，称为三元联系； N 个实体型之间的联系度为 N ，也称为 N 元联系。

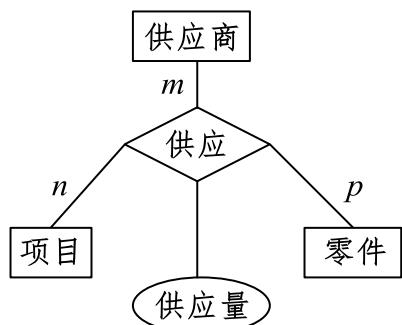
(2) $E - R$ 图

$E - R$ 图提供了表示实体型、属性和联系的方法：

实体型：用矩形表示，矩形框内写明实体名。

属性：用椭圆形表示，并用无向边将其与相应的实体型连接起来。

联系：用菱形表示，菱形框内写明联系名，并用无向边分别与有关实体型连接起来，同时在无向边旁标上联系的类型（1:1，1:n或m:n）。联系可以具有属性。



题 1. 某个工厂物资管理的概念模型。物资管理涉及的实体有：

①仓库：属性有仓库号、面积、电话号码；

②零件：属性有零件号、名称、规格、单价、描述；

③供应商：属性有供应商号、姓名、地址、电话号码、账号；

④项目：属性有项目号、预算、开工日期；

⑤职工：属性有职工号、姓名、年龄、职称；

这些实体之间的联系如下：

①一个仓库可以存放多种零件，一种零件可以存放在多个仓库中，因此仓库和零件具有多对多的联系。用库存量来表示某种零件在某个仓库中的数量。

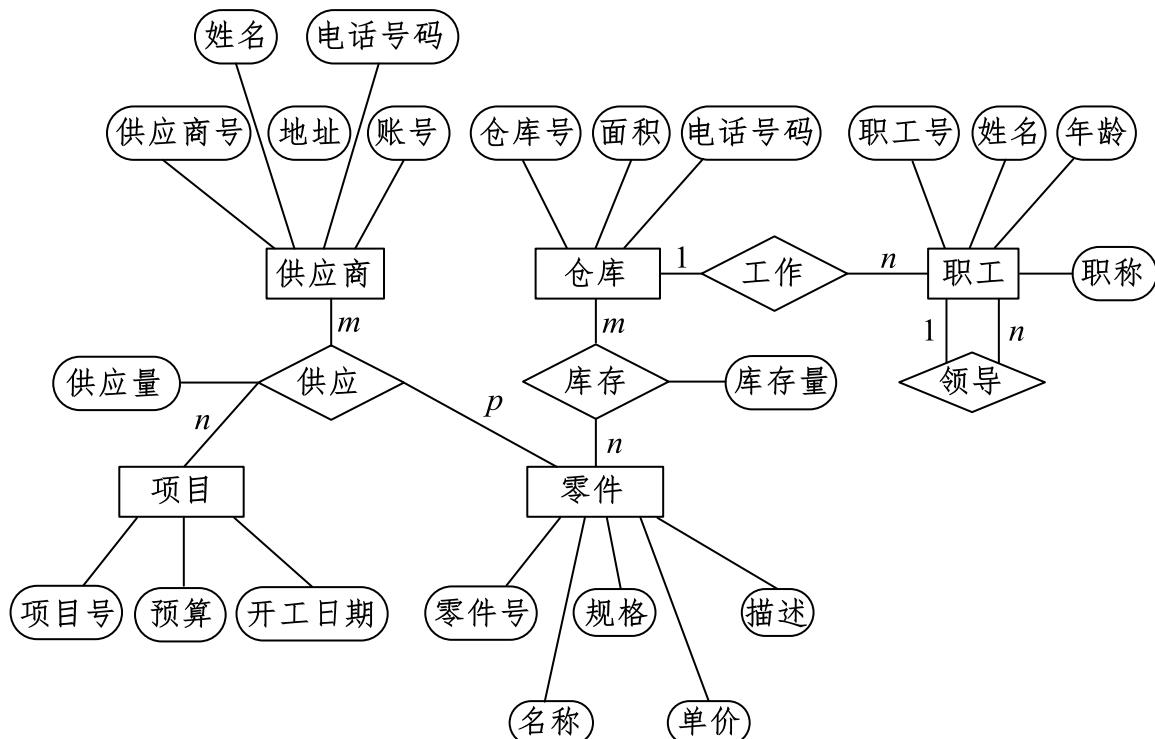
②一个仓库有多个职工当仓库保管员，一个职工只能在一个仓库工作，因此仓库和职工之间是一对多的联系。③职工之间具有领导与被领导关系。即仓库主任领导若干人。

④保管员，因此职工实体型中具有一对多的联系。

⑤供应商、项目和零件三者之间具有多对多的联系。即一个供应商可以供给若干项目多种零件，每个项目可以使用不同供应商供应的零件，每种零件可由不同供应商供给。

设计该模型系统的E-R图。

答案：其E-R图表示如下图：



题 2. 某医院病房计算机管理中需要如下信息：

科室：科名、科地址、科电话、医生姓名；

病房：病房号、床位号、所属科室名；

医生：姓名、职称、所属科室名、年龄、工作证号；

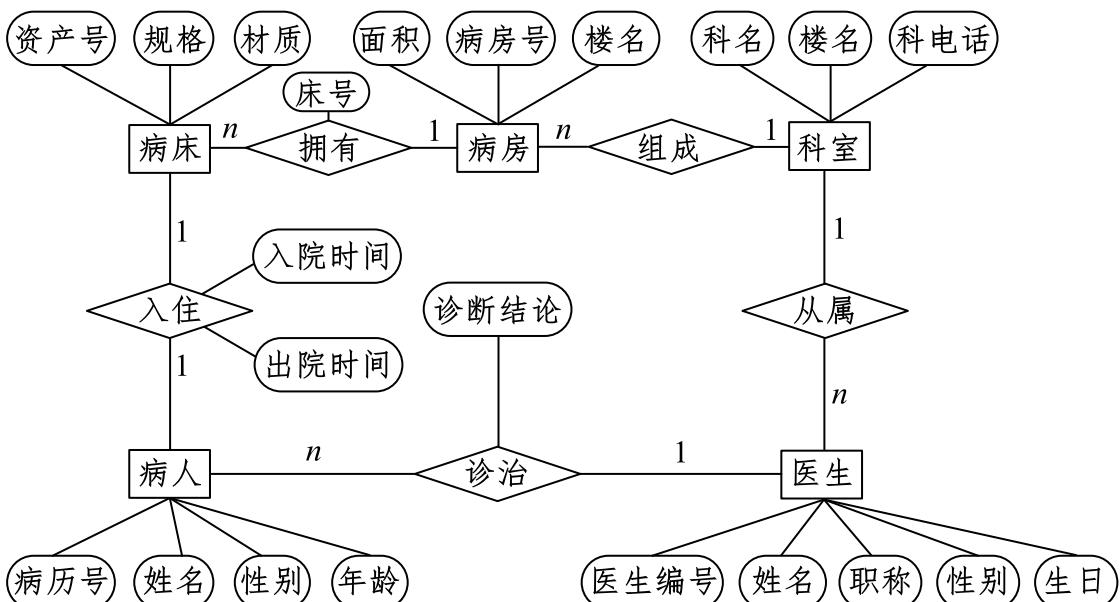
病人：病历号、姓名、性别、诊断、主管医生、病房号。

其中：一个科室由多个病房、多个医生、一个病房只能属于一个科室、一个医生只属于一个科室，但可负责多个病人的诊治、一个病人的主管医生只有一个。

试完成如下设计：

(1) 设计该计算机管理系统的 $E-R$ 图。

答案：其该计算机管理系统的 $E-R$ 图表示如下：



题 3. 分 $E-R$ 图之间的冲突主要有属性冲突、_____、结构冲突三种。

答案：命名冲突

(3) 概念结构设计

实体与属性的划分原则

①作为属性，不能再具有需要描述的性质。属性必须是不可分的数据项，不能包含其他属性。

②属性不能与其他实体具有联系，即 $E-R$ 图中所表示的联系是实体之间的联系。



3. 逻辑结构设计

任务：把概念结构设计阶段设计好的基本 $E-R$ 图转换为与选用数据库管理系统产品所支持的数据模型相符合的逻辑结构。

1) $E-R$ 图向关系模型的转换

(1) 转换内容

$E-R$ 图：由实体型、实体的属性和实体型之间的联系三个要素组成。

关系模型的逻辑结构：一组关系模式的集合

所以，将 $E-R$ 图转换为关系模型：将实体型、实体的属性和实体型之间的联系转化为关系模式。

(2) 转换原则

一个实体型转换为一个关系模式

关系的属性：实体的属性

关系的码：实体的码

实体型之间的联系有以下不同情况

① 1:1 联系

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。

a. 转换为一个独立的关系模式

关系的属性：与该联系相连的各实体的码以及联系本身的属性。

关系的候选码：与该联系相连的各实体的码。

b. 与某一端实体对应的关系模式合并

合并后关系的属性：加入对应关系的码和联系本身的属性

合并后关系的码：不变

② 1:n 联系

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

a. 转换为一个独立的关系模式

关系的属性：与该联系相连的各实体的码以及联系本身的属性

关系的码： n 端实体的码



b. 与 n 端对应的关系模式合并 (可以减少系统中的关系数, 一般更倾向于这种方法)

合并后关系的属性: 在 n 端关系中加入 1 端关系的码和联系本身的属性

合并后关系的码: 不变

③ $m:n$ 联系

转换为一个关系模式

关系的属性: 与该联系相连的各实体的码以及联系本身的属性

关系的码: 各实体码的组合

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。

关系的属性: 与该多元联系相连的各实体的码以及联系本身的属性

关系的码: 各实体码的组合

⑤ 具有相同码的关系模式可合并

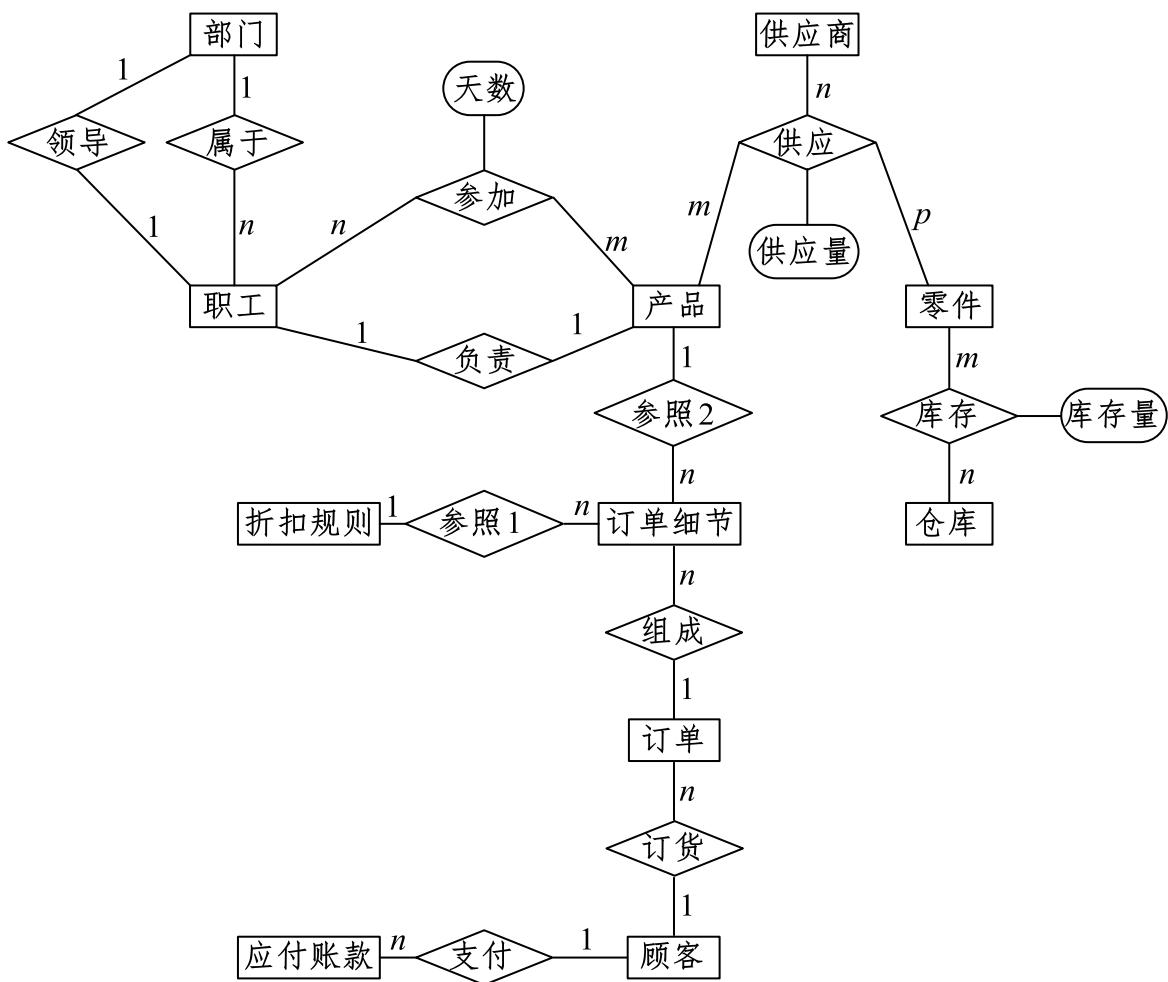
目的: 减少系统中的关系个数

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; 适当调整属性的次序)。

题 1. 将如下图的 $E-R$ 图进行转换。





答案：部门（部门号，部门名，经理的职工号，…）

职工（职工号、部门号，职工名，职务，…）

产品（产品号，产品名，产品组长的职工号，…）

供应商（供应商号，姓名，…）

零件（零件号，零件名，…）

职工工作（职工号，产品号，工作天数，…）

供应（产品号，供应商号，零件号，供应量）

题 2. 如题 1，请将 E-R 图转换为关系表。

答案：科室（科名，科地址，科电话）

医生（姓名，职称，年龄，工作证号，科名）



病人（病历号，姓名，性别，工作证号，病房号）

病房（病房号，床位号，科名）



课时九 练习题

1. 试述 $E-R$ 图转换关系模型的转换原则。
2. $E-R$ 模型用于数据库涉及的哪个阶段（ ）。
 - 需求分析
 - 概念结构设计
 - 逻辑结构设计
 - 物理结构设计
3. 当局部 $E-R$ 图合并成全局 $E-R$ 图可能会出现冲突，下列不属于合并冲突的是（ ）。
 - 属性冲突
 - 概念冲突
 - 语法冲突
 - 命名冲突
4. 从 $E-R$ 模型关系向关系模型转换时，一个 $m:n$ 联系转换为关系模式时，该关系模式的关键字为_____。
 - N 端实体的关键字
 - M 端实体的关键字
 - 重新选取其他属性
 - M 端实体的关键字和 N 端实体的关键字组合
5. 企业集团有若干工厂，每个工厂生产多种产品，且每一种产品可以在多个工厂生产，每个工厂按照固定的计划数量生产产品；每个工厂聘用多名职工，且每名职工只能在一个工厂工作，工厂聘用职工有聘期和工资。工厂的属性有工厂编号、厂名、地址，产品的属性有产品编号、产品名、规格，职工的属性有职工号、姓名。
 - 根据上述语义画出 $E-R$ 图；
 - 将该 $E-R$ 模型转换为关系模型。



课时十 并发控制

考点	重要程度	占分	常用题型
1. 事务	★★★★★	2~10	选择题、简答题
2. 封锁和封锁协议	★★★★	2~5	选择题、简答题
3. 活锁与死锁	★★★★	2~5	选择题、简答题
4. 并发调度的可串行性	★★	2~4	选择题
5. 两段锁协议	★★★	2~4	选择题
6. 封锁的粒度	★★★★	2~4	选择题、简答题

1. 事务

事务(Transaction)是用户定义的一个数据库操作序列，这些操作要么全做，要么全不做，是一个不可分割的工作单位。

1) 事务和程序是两个概念

在关系数据库中，一个事务可以是一条 SQL 语句，一组 SQL 语句或整个程序。

一个程序通常包含多个事务。

事务是恢复和并发控制的基本单位。

2) 定义事务

BEGIN TRANSACTION

SQL 语句 1

SQL 语句 2

.....

COMMIT

3) COMMIT 和 ROLLBACK:

COMMIT: 事务正常结束，提交事务的所有操作（读+更新），事务中所有对数据库的更新写回到磁盘上的物理数据库中。

ROLLBACK: 事务异常终止，事务运行的过程中发生了故障，不能继续执行。系统将事务中对数据库的所有已完成的操作全部撤销，事务滚回到开始时的状态。

4) 事务的 ACID 特性：

ACID: 原子性 (Atomicity)、一致性 (Consistency)、隔离性 (Isolation)、持续性 (Durability)。



原子性 (Atomicity)：事务是数据库的逻辑工作单位，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。

持久性 (Durability)：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。

5) 恢复策略：事务故障的恢复、系统故障的恢复、介质故障的恢复、系统故障恢复。

重做 (REDO) 队列：在故障发生前已经提交的事务。这些事务既有 BEGIN TRANSACTION 记录，也有 COMMIT 记录

撤销 (UNDO) 队列：故障发生时尚未完成的事务。这些事务只有 BEGIN TRANSACTION 记录，无相应的 COMMIT 记录。

恢复中最经常使用的技术：数据库转储，登记日志文件。

题 1. (判断) 事务是数据库的最小逻辑单元 (一个原子单位)。()

答案：正确。

解析：事务是数据库操作的最小工作单元，是作为单个逻辑工作单元执行的一系列操作；这些操作作为一个整体一起向系统提交，要么都执行、要么都不执行；事务是一组不可再分割的操作集合（工作逻辑单元）。

题 2. 事务的四大特性_____、_____、_____ 和 _____。

答案：原子性 (Atomicity)、一致性 (Consistency))、隔离性 (Isolation)、持久性 (Durability)

题 3. 关于事务的故障与恢复，下列描述正确的是 () 。

- A. 事务日志用来记录事务执行的频度
- B. 采用增量备份，数据的恢复可以不使用事务日志文件
- C. 系统故障的恢复只需进行重做 (Redo) 操作
- D. 对日志文件设立检查点目的是为了提高故障恢复的效率

答案：D

解析：数据库系统出现故障的主要几大类：(1) 事务内部的故障。(2) 系统故障。(3) 介质故障。



建立冗余数据最常用的技术是数据转储和登录日志文件。事务日志主要是用来记录事务对数据库的更新操作的文件。转储可以分为海量转储和增量转储两种方式。海量转储是指每次转储全部数据库。增量转储则指每次只转储上一次转储后更新过的数据，从恢复的角度来看，使用海量转储得到的后备副本进行恢复一般说来会更方便些。但如果数据库很大，事务处理又十分频繁，则增量转储方式更实用更有效。不过它恢复是要联用事务日志文件一起来恢复。在恢复技术中，为了解决浪费大量时间在系统恢复中，于是就产生了具有检查点的恢复技术。这种技术在日志文件中增加一类新的记录——检查点记录，增加一个重新开始文件，并让恢复子系统在登录日志文件期间动态维护日志。

2. 封锁和封锁协议

1) 封锁

封锁就是事务 T 在对某个数据对象（表、记录等）操作之前，先向系统发出请求，对其进行加锁；

加锁后事务 T 对该数据对象就有了一定的控制，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；

DBMS 通常提供了多种类型的封锁，一个事务对某个数据对象加锁后究竟拥有怎样的控制是由封锁的类型决定的。

2) 类型

基本封锁类型有如下两种：

排他锁（ X 锁）：又称为写锁，若事务 T 对数据对象 A 加上 X 锁，则只允许 T 读取和修改 A ，其他任何事务都不能再对 A 加任何类型的锁，直到 T 释放 A 上的锁。

共享锁（ S 锁）：又称为读锁，若事务 T 对数据对象 A 加上 S 锁，则其他事务只能再对 A 加 S 锁，而不能加 X 锁，直到 T 释放 A 上的 S 锁。

3) 控制方式

排他锁与共享锁的控制方式可以用下图所示的相容矩阵来表示。

T_1	T_2	X	S	-
X		N	N	Y
S		N	Y	Y
-		Y	Y	Y

配套课程 习题答案



Y 代表相容的请求， N 代表不相容的请求。

最左边一列表示事务 T_1 已经获得的数据对象上的锁的类型，其中横线表示没有加锁。

最上面一行表示另一事务 T_2 对同一数据对象发出的封锁请求。

T_2 的封锁请求能否被满足用矩阵中的 Y 和 N 表示。其中 Y 表示事务 T_2 的封锁与 T_1 已持有的锁相容，封锁请求可以满足； N 表示 T_2 的封锁请求与 T_1 已持有的锁冲突，请求被拒绝。

4) 封锁协议

是指在运用 X 锁和 S 锁对数据对象加锁时需要遵照的一些规则。例如，何时申请、持续时间、和何时释放等。不同的封锁协议，为并发操作的正确调度提供了一定的保证，所能达到的系统一致性级别也是不同的。常用的封锁协议有：支持一致性维护的三级封锁协议、支持并行调度可串行化的两段锁协议/避免死锁的协议。

题 1. 事务 T 已在数据 R 上加了 S 锁，则其他事务在数据 R 上（ ）

- A. 可以加 S 锁不能加 X 锁 B. 不能加 S 锁可以加 X 锁
 C. 可以加 S 锁也可以加 X 锁 D. 不能加任何锁

答案：A

解析：在数据库中有两种基本的锁类型：排它锁（*Exclusive Locks*，即 X 锁）和共享锁

（*ShareLocks*，即 S 锁）。当数据对象被加上排它锁时，其他的事务不能对它读取和修改；加了共享锁的数据对象可以被其他事务读取，但不能修改。

题 2. 解决并发操作带来的数据不一致问题普通采用（ ）技术。

- A. 封锁 B. 存取控制 C. 恢复 D. 协商

答案：A

3. 活锁与死锁

一个问题的解决必然会导致另一个问题的出现。封锁技术可以有效地解决并发操作的一致性问题，但是会带来新的问题：活锁与死锁。

1) 活锁

两个或两个以上事务均处于等待状态，每个事务都在等待其中另一个事务封锁的数据，导致任何事务都不能向前推进的现象。

事务 T_1 封锁数据 R ，事务 T_2 又请求封锁 R ，因此事务 T_2 被迫等待。此时，事务 T_3 也请求



封锁 R ，因此事务 T_3 也被迫等待。当 T_1 释放 R 的封锁后，系统却首先批准了 T_3 的请求， T_2 只能继续等待。然后，又有别的事务到来，由于事务 T_2 的优先级可能较低，所以导致它长时间得不到服务，产生饥饿现象。这就是活锁。避免活锁可以废除特权，采用先来先服务算法。

2) 死锁

死锁是指两个或两个以上的进程在执行过程中，由于竞争资源或者由于彼此通信而造成的一种阻塞的现象，若无外力作用，它们都将无法推进下去；此时称系统处于死锁状态或系统产生了死锁，这些永远在互相等待的进程称为死锁进程。

(1) 死锁的预防

- ①破坏互斥条件
- ②破坏不可剥夺条件
- ③破坏持有并等待条件（对应一次封锁法）
- ④破坏循环等待条件（对应顺序封锁法）。

2) 死锁检测

- (1) 超时法：如果一个事务的等待时间超过了规定的时限，就认为发生了死锁。
- (2) 等待图法（资源分配图）：要求数据库保存两种信息，即锁的信息链表和事务等待链表，通过上面链表构造出一张图，图中若存在回路，就代表存在死锁，资源间发生相互等待。

死锁解除

一旦检测出死锁发生，就应该立即解除死锁。注意并不是系统中所有的进程都是死锁状态，使用死锁检测算法化简资源分配图后，还连着边的那些进程就是需要进行解除的死锁进程。

(3) 解除方法主要有：

资源剥夺法：挂起（暂时放到外存上）某些死锁进程，并抢占它的资源，将这些资源分配给其他的死锁进程。但是应防止被挂起的进程长时间得不到资源而饥饿。

撤销进程法（终止进程法）：强制撤销部分，甚至全部死锁进程，并剥夺这些进程的资源。这种方式的优点是实现简单，但所付出的代价可能性会很大。因为有些进程可能已经运行了很长时间，已经接近结束了，一旦被终止可谓功亏一篑，以后还得从头再来。

进程回退法：让一个或多个死锁进程回退到足以避免死锁的地步。这样就要求系统要记录进程的历史信息，设置还原点。



题 1. 若系统中存在一个等待事务集 $\{T_0, T_1, T_2, \dots, T_n\}$ ，其中 T_0 正等待被 T_1 锁住的数据项 A_1 ， T_1 正等待被 T_2 锁住的数据项 A_2 ， \dots ， T_{n-1} 正等待被 T_n 锁住的数据项 A_n ， T_n 正等待被 T_0 锁住的数据项 A_0 ，则系统处于()的工作状态。

- A. 并发处理 B. 封锁 C. 循环 D. 死锁

答案：D

解析：D 与操作系统中的进程调度类似，在数据库系统中，若系统中存在一个等待事务集 $\{T_0, T_1, T_2, \dots, T_n\}$ ，其中 T_0 正等待被 T_1 锁住的数据项 A_1 ， T_1 正等待被 T_2 锁住的数据项 A_2 ， \dots ， T_{n-1} 正等待被 T_n 锁住的数据项 A_n ， T_n 正等待被 T_0 锁住的数据项 A_0 ，则系统处于死锁的工作状态。

题 2. 简述几种破坏死锁条件的措施。

- 解：(1) 预防死锁。通过一些限制条件的设置来破坏死锁发生的四个必要条件中一个或多个，以预防死锁的发生。
- (2) 资源的动态分配的过程中用某些算法加以限制，防止系统进入不安全状态从而避免死锁的发生。态从而避免死锁的发生。
- (3) 检测死锁。采取一定的机制检测系统是否死锁，以配合死锁的解除。
- (4) 解除死锁。通过撤销一些进程回收资源把系统从死锁中解脱出来。

4. 并发调度的可串行性

1) 可串行化调度

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，称这种调度策略为可串行化(serializable)调度。可串行性是并发事务正确调度的准则，也即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。

例如，下面有两个事务，分别包含下列操作

事务 T_1 ：读 B 、 $A=B+1$ 、写回 A ；

事务 T_2 ：读 A 、 $B=A+1$ ，写回 B ；

假设 A B 初值均为 2，两个事务最多只有两种串行执行策略

$T_1 \rightarrow T_2 ; A = 3, B = 4$

$T_2 \rightarrow T_1 ; A = 4, B = 3$



因此事务 T_1 和 T_2 不管怎样交叉并行运行，只有两种正确的结果。其他结果均是错误的，相应调度也称之为不可串行化调度。

2) 冲突可串行化调度

(1) 冲突操作

是指不同事务对同一个数据的读写操作和写写操作。除此之外，其他操作均为不冲突操作

$R_i(x)$ 与 $W_j(x)$ ：事务 T_i 读 X ，事务 T_j 写 X

$W_i(x)$ 与 $W_j(x)$ ：事务 T_i 写 X ，事务 T_j 写 X

另外注意各种交换：

不同事务的冲突操作不可交换

同一事务内部的两个操作不可交换

不同事务，同一数据的读读操作可以交换

不同事务，不同数据，无论读写均可交换

可串行化调度的充分条件：冲突可串行化

一个调度 SC 在保证冲突操作的次序不变的情况下，通过交换两个事务不冲突操作的次序得到另一个调度 SC' ，如果 SC' 是串行的，则称调度 SC 为冲突可串行化的调度。若一个调度是冲突可串行化的，那么它一定是可串行化的调度。

注意：冲突可串行化调度是可串行化调度的充分条件，不是必要条件。也就是说有可能某个调度是可串行调度，但它却不是冲突可串行化调度

题 1. 设有如下调度 S ：

T_1	T_2	T_3
	read(B) write(B)	
		read(B)
	read(A)	
		write(B)
	write(A)	
read(B) write(B)		
		read(A) write(A)
read(A) write(A)		

(1) 判断 S 是否为冲突可串行化调度？

配套课程 习题答案



(2) 如果是冲突可创新化调度，则给出与 S 冲突的的串行调度。

解析：(1) S 为可串行化调度。

(2) 等价的串行调度为 T_2, T_3, T_1

5. 两段锁协议

两段锁协议 (2PL)：两段锁协议是三级封锁协议的特例，目前 DBMS 普遍采用该种协议实现并发调度的可串行性。

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。

在释放一个封锁之后，事务不再申请和获得任何其他封锁。

其中“两段”是指事务分为两个阶段

第一阶段：获得封锁，也称为扩展阶段

第二阶段：释放封锁，也称为收缩阶段

另外还需要注意：

事务遵守两段锁协议是可串行化调度的充分条件，而非必要条件。

若并发事物都遵循两段锁协议，则对其的任何并发点都策略都是可串行化的。

若对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。

最后注意区分两段锁协议和一次封锁法：

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，因此一次封锁法遵守两段锁协议。

但是两段锁协议并不要求事务必须一次将所有要使用的数据全部加锁，因此遵守两段锁协议的事务可能发生死锁。

题 1. 如果总是将事务为两个阶段，一个是加锁期，一个是解锁期，在加锁期不允许解锁，在解锁期不允许加锁，则将该规定称为（ ）。

答案：两段锁协议。

6. 锁的粒度

1) 概念

封锁粒度 (granularity)：是指封锁对象的大小。封锁对象可以是逻辑单元，也可以是物理单元。封锁粒度与系统并发度和并发控制的开销密切相关，一般来说，封锁粒度越大，数据库所能封锁的数据单元就越少，并发度越小，开销越小。



逻辑单元：元组、关系、整个数据库等

物理单元：页（数据页或索引页）、物理记录等

2) 选择封锁的原则

处理多个关系的大量元组的用户事务时以数据库为封锁单位

处理大量元组的用户事务时以关系为封锁单元

处理少量元组的用户事务时以元组为封锁单位

3) 多粒度封锁

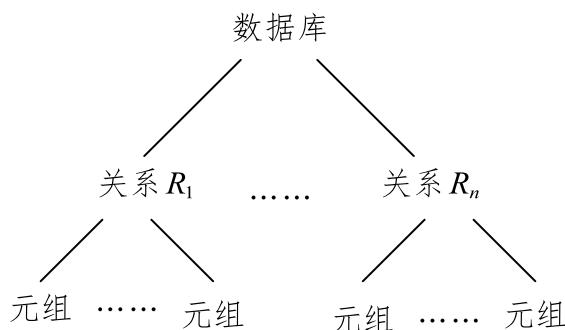
多粒度封锁：在一个系统中同时支持多种封锁粒度供不同的事务选择

(1) 多粒度树

多粒度树是以树形结构来表示多级封锁粒度的方法

根结点是整个数据库，表示最大的数据粒度

叶结点表示最小的数据粒度



(2) 多粒度封锁协议

多粒度封锁协议：允许多粒度树中的每个结点被独立地加锁，对一个结点加锁意味着这个结点的所有后裔结点也会被加上相同类型的锁。因此，在多粒度封锁中一个数据对象可能存在如下两种封锁方式

显式封锁：直接加到数据库对象上的封锁

隐式封锁：由于上级结点加锁而使该数据对象也被加锁

多粒度封锁方法中，显式封锁和隐式封锁的效果是一样的，因此系统检查封锁冲突时不仅要检查显式封锁还要检查隐式封锁

例如事务 T 要对关系 R_1 加 X 锁，系统必须搜索其上级结点数据库、关系 R_1 以及 R_1 的夏季结点，即 R_1 中的每一个元组，上下搜索。如果其中某一个数据对象已经加了不相容锁，则 T 必须等待。

4) 意向锁



一般地，对某个数据对象加锁，系统要检查该数据对象上有无显式封锁与之冲突：再检查其所有上级结点，看本事务的显式封锁是否与该数据对象上的隐式封锁（即由于上级结点已加的封锁造成的）冲突；还要检查其所有下级结点，看它们的显式封锁是否与本事务的隐式封锁（将加到下级结点的封锁）冲突。

可以看出，这样的检查方法效率很低，因此意向锁由此诞生。

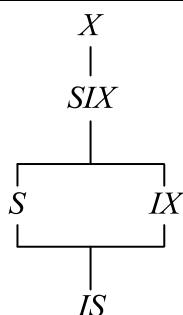
意向锁：如果对一个结点加意向锁，则说明该结点的下层结点正在被加锁；对任一结点加锁时，必须先对它的上层结点加意向锁。有如下三种常用的意向锁。

意向共享锁（IS锁）：如果对一个数据对象加IS锁，表示它的后裔结点拟（意向）加S锁。

意向排他锁（IX锁）：如果对一个数据对象加IX锁，表示它的后裔结点拟（意向）加X锁。

共享意向排他锁（SIX锁）：如果对一个数据对象加SIX锁，表示对它加S锁，再加IX锁，即。如下图为锁相容矩阵。

T_1	T_2	S	X	IS	IX	SIX
S		✓	✗	✓	✗	✗
X		✗	✗	✗	✗	✗
IS		✓	✗	✓	✓	✓
		✗	✗	✓	✓	✗
SIX		✗	✗	✓	✗	✗



锁强度的偏序关系

题 1. SIX 锁的中文全称是_____。

答案：共享意向排它锁。

题 2. 在并发控制的技术中，最常用的是封锁方法。对于共享锁（S）和排他锁（X）来说，下面列出的相容关系中，哪一个是不正确的。



A. $X/X:TRUE$ B. $S/S:TRUE$ C. $S/X:FALSE$ D. $X/S:FALSE$

答案： A

解析：为了避免发生并发操作引起的数据不一致性问题，则采用数据封锁技术实现并发控制。封锁是防止存取同一资源的用户之间相互干扰的机制，即当一个用户对数据库某个数据对象执行修改操作时，对该部分数据加锁，拒绝其他用户对该部分的并发访问要求，直至该事务执行完毕才释放数据对象。所以 $X/X:TRUE$ 是不对的。



课时十 练习题

1. 1 级封锁协议加上 T 要读取的数据 R 加 S 锁，这是（ ）

A. 3 级封锁协议 B. 4 级封锁协议

C. 2 级封锁协议 D. 1 级封锁协议

2. 事务的并发执行不会破坏 DB 的完整性，这个性质称为事务的（ ）。

A. 原子性 B. 隔离性 C. 持久性 D. 一致性

3. 如果事务 T_1 封锁了数据 R_1 , T_2 封锁了数据 R_2 ，然后 T_1 又请求封锁 R_2 ，因 T_2 已封锁了 R_2 ，于是 T_1 等待 T_2 释放 R_2 上的锁。接着 T_2 又申请封锁 R_1 ，因 T_1 已封锁了 R_1 ， T_2 也只能等待 T_1 释放 R_1 上的锁。这样就出现了 T_1 在等待 T_2 ，而 T_2 又在等待 T_1 的局面， T_1 和 T_2 两个事务永远不能结束，形成（ ）

A. 并发处理 B. 封锁 C. 循环 D. 死锁

4. 事务的持续性是指（ ）

A. 事务中包括的所有操作要么都做，要么都不做

B. 事务一旦提交，对数据库的改变是永久

C. 一个事务内部的操作及使用的数据对并发的其他事务是隔离的

D. 事务必须是使数据库从一个一致性状态变到另一个一致状态

5. 对并发操作若不加以控制，可能会带来数据的_____问题。

A. 不安全 B. 死锁 C. 死机 D. 不一致

6. 设有两个事务 T_1 , T_2 ，其并发操作如下表所示，

T_1	T_2
① 读 $A = 10$ ② ③ $A = A - 5$ 写回 ④	读 $A = 10$ $A = A - 8$ 写回



下面评价正确的是（ ）

A. 该操作不存在问题 B. 该操作丢失修改

C. 该操作不能重复读 D. 读操作读脏数据

6、一个事务中所有对数据库操作是一个不可分割的操作序列，这称为事务的（ ）

A. 原子性 B. 一致性 C. 隔离性 D. 持久性

7、简述什么是事务以及事务的四个基本特征。



恭喜你完成本课程学习！

领取练习题答案
&配套课程等资料
请关注公众号【蜂考】



一起学习，答疑解惑
请加入蜂考学习交流群

