作者：希赛教育软考学院　来源：希赛网　2014年05月20日

考点分析

第7章 数据库系统

数据库系统是软件设计师的一个考试重点，上午考试中大约有6分的试题是数据库技术方面的试题，下午有1道试题是数据库技术方面的试题，占15分。在这些试题中，主要考查数据库模型、数据的基本操作、数据库设计，以及事务处理。

7.1 考点分析

本节把历次考试中数据库系统方面的试题进行汇总，得出本章的考点，如表7-1和表7-2所示。

表7-1 数据库系统上午试题知识点分布

考试时间	分数	考查知识点
10.11	5	实体联系（1）、候选键（1）、主键（2）、模式分解（1）
11.05	5	SQL 语言（3）、元组演算（2）
11.11	5	集合运算（1）、关系代数（1）、SQL 语言（1）、候选键（1）、模式分解（1）
12.05	5	关系代数（1）、完整性（1）、SQL 语言（2）、事务回滚（1）
12.11	5	元组演算（1）、外键（1）、SQL 语言（1）、并发操作（2）
13.05	6	关系代数（1）、完整性（1）、SQL 语言（2）、函数依赖（2）
13.11	6	完整性（1）、全码（1）、范式（1）、关系代数（2）、封锁（1）
14.05	6	三级模式结构（2）、实体联系（2）、关系运算（2）

根据表7-1,我们可以得出数据库技术基础的考点主要有：

（1）数据库模型：主要考查模型的分类、三级模式。

（2）数据操作：包括关系代数、元组演算和SQL语言。

（3）数据库设计：包括候选键、主键、外键、函数依赖、范式、模式分解、E-R图等。

（4）事务处理：包括事务的概念和并发处理方面的问题。

对这些知识点进行归类，然后按照重要程度进行排列，如表7-2所示。其中的星号（*）代表知识点的重要程度，星号越多，表示越重要。

另外，在下午考试中，每次都有1道数据库系统的试题，占15分。下午试题主要涉及的内容如表7-3所示。

表7-2 数据库系统各知识点重要程度

知识点	10.11	11.05	11.11	12.05	12.11	13.05	13.11	14.05	合计	比例	重要程度
数据操作		5	3	3	2	3	2	2	20	46.51%	★★★★★
数据库设计	5		2		1	2	2	2	14	32.56%	★★★
事务处理				2	2	1	2		7	16.28%	★★
数据库模型								2	2	4.65%	★

表7-3 数据库系统下午试题知识点分布

考试时间	分数	考查知识点
10.11	15	E-R 图，关系模式，主码、外码、SQL 语句
11.05	15	主键，外键，SQL 语句，完整性
11.11	15	关系模式，主键，E-R 图
12.05	15	E-R 图，关系模式，主键，外键
12.11	15	E-R 图，关系模式，索引，SQL 语言
13.05	15	E-R 图，关系模式，主键
13.11	15	E-R 图，关系模式，主键
14.05	1	E-R 图，关系模式，主键

从表7-3中我们可以看出：下午考试的知识点与上午考试的知识点是相同的，主要集中在数据库设计和SQL语言上。在本章的后续内容中，我们将主要介绍这些知识点。

上一节　本书简介　下一节

数据库模型

7.2 数据库模型

数据库系统的设计目标是允许用户逻辑地处理数据，而不必涉及这些数据在计算机中是怎样存放的，在数据组织和用户应用之间提供某种程度的独立性。

上一节　本书简介　下一节

数据库系统的三级模式

7.2.1 数据库系统的三级模式

数据库系统的三级模式为概念模式、外模式和内模式。

1.概念模式

概念模式（模式、逻辑模式）用以描述整个数据库中数据库的逻辑结构，描述现实世界中的实体及其性质与联系，定义记录、数据项、数据的完整性约束条件及记录之间的联系是数据项值的框架。概念模式通常还包含有访问控制、保密定义、完整性检查等方面的内容，以及概念/物理之间的映射。

概念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。一个数据库只有一个概念模式。

2.外模式

外模式（子模式、用户模式）用以描述用户看到或使用的那部分数据的逻辑结构，用户根据外模式用数据操作语句或应用程序去操作数据库中的数据。外模式主要描述组成用户视图各个记录的组成、相互关系、数据项的特征、数据的安全性和完整性约束条件。

外模式是数据库用户（包括程序员和最终用户）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。一个数据库可以有多个外模式。一个应用程序只能使用一个外模式。

3.内模式

内模式是整个数据库的最低层表示，不同于物理层，它假设外存是一个无限的线性地址空间。内模式定义的是存储记录的类型、存储域的表示、存储记录的物理顺序，指引元、索引和存储路径等数据的存储组织。

内模式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。一个数据库只有一个内模式。

4.三级模式的关系

（1）模式是数据库的中心与关键。

（2）内模式依赖于模式，独立于外模式和存储设备。

（3）外模式面向具体的应用，独立于内模式和存储设备。

（4）应用程序依赖于外模式，独立于模式和内模式。

5.两级独立性

数据库系统两级独立性是指物理独立性和逻辑独立性。三级模式之间通过两级映射（外模式/模式映射，模式/内模式映射）进行相互转换，使得数据库的三级形成一个统一的整体。

（1）物理独立性。物理独立性是指用户的应用程序与存储在磁盘上的数据库中的数据是相互独立的。当数据的物理存储改变时，应用程序不需要改变。物理独立性存在于概念模式和内模式之间的映射转换，说明物理组织发生变化时应用程序的独立程度。

（2）逻辑独立性。逻辑独立性是指用户的应用程序与数据库中的逻辑结构是相互独立的。当数据的逻辑结构改变时，应用程序不需要改变。逻辑独立性存在于外模式和概念模式之间的映射转换，说明概念模式发生变化时应用程序的独立程度。

值得注意的是逻辑独立性比物理独立性更难实现。

上一节　本书简介　下一节

第 7 章：数据库系统　作者：希赛教育软考学院　来源：希赛网　2014年05月20日

数据模型的分类

7.2.2 数据模型的分类

数据模型主要有两大类，分别是概念数据模型（实体联系模型）和基本数据模型（结构数据模型）。

1.概念数据模型

概念数据模型是按照用户的观点来对数据和信息建模，主要用于数据库设计。概念模型主要用实体联系方法表示，所以也称E-R模型。

基本数据模型是按照计算机系统的观点对数据和信息建模，主要用于DBMS的实现。基本数据模型是数据库系统的核心和基础。基本数据模型通常由数据结构、数据操作和完整性约束三部分组

成。其中数据结构是对系统静态特性的描述，数据操作是对系统动态特性的描述，完整性约束是一组完整性规则的集合。

2.基本数据模型

常用的基本数据模型有层次模型、网状模型、关系模型和面向对象模型。

层次模型用树型结构表示实体类型及实体间联系。层次模型的优点是记录之间的联系通过指针来实现，查询效率较高。层次模型的缺点是只能表示1:n联系，虽然有多种辅助手段实现m:n联系，但较复杂，用户不易掌握。由于层次顺序的严格和复杂，使得数据的查询和更新操作很复杂，应用程序的编写也比较复杂。

网状模型用有向图表示实体类型及实体间联系。网状模型的优点是记录之间的联系通过指针实现，m:n联系也容易实现，查询效率高。其缺点是编写应用程序比较复杂，程序员必须熟悉数据库的逻辑结构。

关系模型用表格结构表达实体集，用外键表示实体间联系，其优点有：

（1）建立在严格的数学概念基础上。

（2）概念单一（关系），结构简单、清晰，用户易懂易用。

（3）存取路径对用户透明，从而数据独立性、安全性好，简化数据库开发工作。

关系模型的缺点主要是由于存取路径透明，查询效率往往不如非关系数据模型高。

第 7 章：数据库系统　作者：希赛教育软考学院　来源：希赛网　2014年05月20日

关系模型

7.2.3 关系模型

先学习几个基本概念。

（1）域：一组具有相同数据类型的值的集合。

（2）笛卡儿积：给定一组域$D_1,D_2,\ldots,D_n$,其中可以有相同的域。

$D_1,D_2,\ldots,D_n$的笛卡儿积为：

$D_1\times D_2\times\ldots\times D_n=\{(d_1,d_2,\ldots,d_n)|d_j\in D_j,j=1,2,\ldots,n\}$

其中每一个元素$(D_1,D_2,\ldots,D_n)$叫做一个n元组（简称为元组）。元组中的每一个值d_j叫做一个分量。

（3）关系：D1×D2×…×Dn的子集叫做在域$D_1,D_2,\ldots,D_n$上的关系，表示为：

$R(D_1,D_2,\ldots,D_n)$

这里R表示关系的名字，n是关系的目或度。

关系中的每个元素是关系中的元组，通常用t表示。关系是笛卡儿积的子集，所以关系也是一个二维表，表的每行对应一个元组，表的每列对应一个域。由于域可以相同，为了加以区分，必须为每列起一个名字，称为属性。

派生属性是指可以指由其他属性经过运算得到的属性，因而派生属性会产生冗余，通常不存

储。例如，如果某关系中有年龄、出生日期这两个属性，则年龄就是派生属性，因为可用出生年月计算出年龄的值。

多值属性（组合属性）是指可同时由多个值表示的属性。例如，包含关于员工信息的关系可能包含关于员工兴趣的数据。一个员工可能有多个兴趣，例如运动、电影、投资、烹调等，并且由于这些值的任何一个或所有这些值可能同时是某一个员工的兴趣。

关系的度是指关系中属性的个数，关系的势是指关系中元组的个数。关系中的不同属性可以取在相同的域中取值，属性的个数与域的个数并不相同。

关系可以有三种类型：基本关系（通常又称为基本表或基表）、查询表和视图表。基本表是实际存在的表，它是实际存储数据的逻辑表示。查询表是查询结果对应的表。视图表是由基本表或其他视图表导出的表，是虚表，不对应实际存储的数据。

基本关系具有以下六条性质。

（1）列是同质的，即每一列中的分量是同一类型的数据，来自同一个域。

（2）不同的列可出自同一个域，称其中的每一列为一个属性，不同的属性要给予不同的属性名。

（3）列的顺序无所谓，即列的次序可以任意交换。

（4）任意两个元组不能完全相同。但在大多数实际关系数据库产品中，如Oracle等，如果用户没有定义有关的约束条件，它们都允许关系表中存在两个完全相同的元组。

（5）行的顺序无所谓，即行的次序可以任意交换。

（6）分量必须取原子值，即每一个分量都必须是不可分的数据项。

关系的描述称为关系模式。一个关系模式应当是一个五元组。它可以形式化地表示为：R（U,D,DOM,F）。其中R为关系名，U为组成该关系的属性名集合，D为属性组U中属性所来自的域，DOM为属性向域的映像集合，F为属性间数据的依赖关系集合。关系模式通常可以简记为：R（A_1,A_2,…，A_n）。其中R为关系名，A_1,A_2,…，A_n为属性名。

关系实际上就是关系模式在某一时刻的状态或内容。也就是说，关系模式是型，关系是它的值。关系模式是静态的、稳定的，而关系是动态的、随时间不断变化的，因为关系操作在不断地更新着数据库中的数据。但在实际当中，常常把关系模式和关系统称为关系，读者可以从上下文中加以区别。

在关系模型中，实体及实体间的联系都是用关系来表示的。在一个给定的现实世界领域中，相应于所有实体及实体之间的联系的关系集合构成一个关系数据库。

关系数据库也有型和值之分。关系数据库的型也称为关系数据库模式，是对关系数据库的描述，是关系模式的集合。关系数据库的值也称为关系数据库，是关系的集合。关系数据库模式与关系数据库通常统称为关系数据库。

关系代数

7.3 关系代数

本知识点主要考查集合运算和关系运算。关系代数是以集合代数为基础发展起来的，它是以关系为运算对象的一组高级运算集合。关系代数可以分为基本的集合运算和专有的关系运算两大类。

第 7 章：数据库系统　作者：希赛教育软考学院　来源：希赛网　2014年05月20日

集合运算

7.3.1 集合运算

传统的集合运算是二目运算，包括并、交、差、广义笛卡儿积四种运算。

1.并

设关系R和S具有相同的关系模式，R和S的并是由属于R或属于S的元组组成的集合，记为R∪S.形式定义如下：

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

式中t是元组变量（下同）。显然，R∪S=S∪R.

2.差

关系R和S具有相同的关系模式，R和S的差是由属于R但不属于S的元组组成的集合，记为R–S.形式定义如下：

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

3.交

关系R和S具有相同的关系模式，R和S的交是由既属于R又属于S的元组组成的集合，记为R∩S.形式定义如下：

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

显然，R∩S =R–（R–S），或者R∩S = S–（S–R）。

4.笛卡儿积

设关系R和S元数分别为r和s.R和S的笛卡儿积是一个r+s元的元组集合，每个元组的前r个分量来自R的一个元组，后 s个分量来自S的一个元组，记为R×S.形成定义如下：

$$R \times S \equiv \{t \mid t = <t_r, t_s> \wedge t_r \in R \wedge t_s \in S\}$$

若R有m个元组，S有n个元组，则R×S有m×n个元组。

5.集合运算实例

例如：设关系R和S如表7-4所示。则R∪S与R∩S如表7-5所示，R–S和S–R如表7-6所示，R×S如表7-7所示。

表7-4 关系R和S

R 关系				S 关系		
A1	A2	A3		A1	A2	A3
a	b	c		a	b	a
b	a	d		b	a	d
c	d	d		c	d	d
d	f	g		d	s	c

表7-5 R∪S与R∩S

R∪S				R∩S		
A1	A2	A3		A1	A2	A3
a	b	c		b	a	d
b	a	d		c	d	d
c	d	d				
d	f	g				
a	b	a				
d	s	c				

表7-6 R–S和S–R

R–S				S–R		
A1	A2	A3		A1	A2	A3
a	b	c		a	b	a
d	f	g		d	s	c

表7-7 R×S

A1	A2	A3	A1	A2	A3
a	b	c	a	b	a
b	a	d	a	b	a
c	d	d	a	b	a
d	f	g	a	b	a
a	b	c	b	a	d
b	a	d	b	a	d
c	d	d	b	a	d
d	f	g	b	a	d
a	b	c	c	d	d
b	a	d	c	d	d
c	d	d	c	d	d
d	f	g	c	d	d
a	b	c	d	s	c
b	a	d	d	s	c
c	d	d	d	s	c
d	f	g	d	s	c

上一节　本书简介　下一节

作者：希赛教育软考学院　来源：希赛网　2014年05月20日

关系运算

7.3.2 关系运算

在7.3.1节的集合运算基础上，关系数据库还有一些专门的运算，主要有投影、选择、连接、除法和外连接。它们是关系代数最基本的操作，也是一个完备的操作集。在关系代数中，由这5种基本代数操作经过有限次复合的式子称为关系代数运算表达式。表达式的运算结果仍是一个关系。我们可以用关系代数表达式表示各种数据查询和更新处理操作。

1.投影

投影操作从关系R中选择出若干属性列组成新的关系，该操作对关系进行垂直分割，消去某些列，并重新安排列的顺序，再删去重复元组。记为：

$$R \bigcup S \equiv \{t \mid t \in R \vee t \in S\}$$

其中A为R的属性列。

2.选择

选择操作在关系R中选择满足给定条件的所有元组，记为：

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

其中F表示选择条件，是一个逻辑表达式（逻辑运算符+算术表达式 ）。选择运算是从行的角度进行的运算。

3.θ连接

θ连接从两个关系的笛卡儿积中选取属性间满足一定条件的元组，记为：

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

其中A和B分别为R和S上度数相等且可比的属性组。θ为"="的连接，称为等值连接，记为：

$$R \times S \equiv \{t \mid t = <t_r, t_s> \wedge t_r \in R \wedge t_s \in S\}$$

如果两个关系进行比较的分量必须是相同的属性组，并且在结果中把重复的属性列去掉，则称为自然连接，记为：

$$\pi_A(R) \equiv \{t[A] \mid t \in R\}$$

4.除法

设两个关系R和S的元数分别为r和s（设r>s>0），那么R÷S是一个（r–s）元的元组的集合。R÷S是满足下列条件的最大关系：其中每个元组t与S中每个元组u组成新元组<t,u>必在关系R中。其具体计算公式如下：

$$\sigma_F(R) \equiv \{t \mid t \in R \wedge F(t) = \text{true}\}$$

5.外连接

两个关系R和S进行自然连接时，选择两个关系R和S公共属性相等的元组，去掉重复的属性列构成新关系。这样，关系R中的某些元组有可能在关系S中不存在公共属性值相等的元组，造成关系R中这些元组的值在运算时舍弃了；同样关系S中的某些元组也可能舍弃。为此，扩充了关系运算左外连接、右外连接和完全外连接。

（1）左外连接：R和S进行自然连接时，只把R中舍弃的元组放到新关系中。

（2）右外连接：R和S进行自然连接时，只把S中舍弃的元组放到新关系中。

（3）完全外连接：R和S进行自然连接时，只把R和S中舍弃的元组都放到新关系中。

6.关系运算实例

设两个关系模式R和S如表7-8所示，则的结果如表7-9所示，的结果如表7-10所示，的结果如表7-11所示，R与S的左外连接如表7-12所示，R与S的右外连接如表7-13所示，R与S的完全外连接如表7-14所示。

表7-8 关系R和S

R 关系				S 关系		
A1	A2	A3		A1	A2	A4
a	b	c		a	z	a
b	a	d		b	a	h
c	d	d		c	d	d
d	f	g		d	s	c

表 7-9 对关系 R 求投影操作

A1	A2
a	b
b	a
c	d
d	f

表 7-10 对关系 R 求选择操作

A1	A2	A3
b	a	d

表 7-11 对关系 R 和 S 的自然连接

A1	A2	A3	A4
b	a	d	h
c	d	d	d

表 7-12 R 与 S 的左外连接

A1	A2	A3	A4
a	b	c	null
b	a	d	h
c	d	d	d
d	f	g	null

表 7-13 R 与 S 的右外连接

A1	A2	A3	A4
a	z	null	a
b	a	d	h
c	d	d	d
d	s	null	c

表 7-14 R 与 S 的完全外连接

A1	A2	A3	A4
a	b	c	null
b	a	d	h
c	d	d	d
d	f	g	null
a	z	null	a
d	s	null	c

第 7 章：数据库系统 作者：希赛教育软考学院 来源：希赛网 2014年05月20日

元组演算

7.3.3 元组演算

在元组演算中，元组演算表达式简称为元组表达式，其一般形式为{t|P（t）},其中，t是元组变量，表示一个元数固定的元组；P是公式，在数理逻辑中也称为谓词，也就是计算机语言中的条件表达式。{t|P（t）}表示满足公式P的所有元组t的集合。

在元组表达式中，公式由原子公式组成。原子公式有下列三种形式。

（1）R（s），其中R是关系名，s是元组变量。其含义是"s是关系R的一个元组".

（2）s[i]θu[j],其中s和u是元组变量，θ是算术比较运算符，s[i]和u[j]分别是s的第i个分量和u的第j个分量。原子公式s[i] θu[j]表示"元组s的第i个分量与元组u的第j个分量之间满足θ运算".

（3）s[i] θa或aθu[j],其中的a为常量。其含义类似于（2）。

在一个公式中，如果元组变量未用存在量词$^{\exists}$或全称量词$^{\forall}$等符号定义，那么称为自由元组变量，否则称为约束元组变量。

公式的递归定义如下：

（1）每个原子是一个公式。其中的元组变量是自由变量。

（2）如果P1和P2是公式，那么┐P1、P1∨P2、P1∧P2和P1→P2也是公式。

（3）如果P1是公式，那么（$^{\exists}$s）（P1）和（$^{\forall}$s）（P1）也都是公式。

（4）公式中各种运算符的优先级从高到低依次为：θ，$^{\exists}$和$^{\forall}$，┐，∧和∨，→。在公式外还可

以加括号，以改变上述优先顺序。

（5）公式只能由上述四种形式构成，除此之外构成的都不是公式。

在元组演算的公式中，有下列4个等价的转换规则：

P1∧P2等价于┐（┐P1∨┐P2）。

P1∨P2等价于┐（┐P1∧┐P2）。

（$\forall$s）（P1（$\exists$s））等价于┐（s）（┐P1（s））。

（$\exists$s）（P1（$\forall$s））等价于┐（s）（┐P1（s））。

P1→P2等价于┐P1∨P2.

关系代数表达式可以转换为元组表达式，例如：

R∪S可用{t|R（t）∨S（t）}表示。

R–S可用{t|R（t）∧┐S（t）}表示。

R×S可用{t|（$\exists$u）（$\exists$v）（R（u）∧S（v）∧t[1]=u[1]∧t[r]=u[r]∧…∧t[r]=u[r]∧t[r+1]=v[1]∧…∧t[r+s]=v[s]）}表示，此处设R是r元，S是s元。

设投影操作是π2,3（R），那么元组表达式可写成{t|（$\exists$u）（R（u）∧t[l]=u[2]∧t[2]=u[3]）}.

σF（R）可用{t|R（t）∧F'}表示，F'是F的等价表示形式。例如，σ2='d'（R）可写成{t|（R（t）∧t[2]='d'）。

例如：设学生S、课程C、学生选课SC的关系模式分别为：S（Sno,Sname,Sage,Saddr）、C（Cno,Cname,Pcno），以及SC（Sno,Cno,Grade）。我们求与关系代数表达式$\pi_{sno,sname,grade}(\sigma_{cname='data'}(S \triangleright\triangleleft SC \triangleright\triangleleft C))$等价的元组演算表达式。因为涉及三个关系模式S、SC、C,为了转换成等价的元组演算表达式，需要设置3个元组变量u、v、w,而且这3个元组变量只要用存在量词$\exists$限定就可以了。（$\exists$u）S（u）表示在S关系中存在一个元组，（$\exists$v）SC（v）表示在SC关系中存中一个元组，（$\exists$w）C（w）表示在C关系中存在一个元组。因为u[1]对应的是S.Sno,v[1]对应的是SC.Sno,v[2]对应的是SC.Cno,w[1]对应的是C.Cno,w[2]对应的是C.Cname,所以S.Sno=SC.Sno且SC.Cno=C.Cno且Cname='data'等价于u[1]=v[1]∧v[2]=w[1]∧w[2]='data'.因为本题的结果集为Sno、Sname和Grade,而u[1]对应的是S.Sno,u[2]对应的是S.Sname,v[3]对应的是SC. Grade,所以对属性列Sno、Sname和Grade的投影等价于t[1]=u[1]∧t[2]=u[2]∧t[3]=v[3].因此，$\pi_{sno,sname,grade}(\sigma_{cname='data'}(S \triangleright\triangleleft SC \triangleright\triangleleft C))$与等价的元组演算表达式为{$(\exists u)(\exists v)(\exists w)$S（u）∧SC（v）∧C（w）∧u [1]= v [1]∧v [2]= w [l]∧w [2]='data'∧t[1]= u [1]∧t [2]= u [2]∧t [3]= v [3]}.

上一节　本书简介　下一节

SQL语言

7.4 SQL语言

SQL-86是第一个SQL标准，后续的有SQL-89、SQL-92（SQL2）、SQL-99（SQL3）等。但作为考试而言，所考查的是一些基本的语法知识。

基本表操作

7.4.1 基本表操作

在基本表操作方面，主要考查定义基本表语句和查询基本表语句。

1.定义基本表

SQL语言使用动词CREATE定义基本表，其具体语法格式如下：

```
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]][,
<表级完整性约束条件>]）；
```

在这个语句中，主要考查一些约束条件。有关这方面的知识，请参考7.4.3节。

2.基本表查询

SQL语言查询语句的基本格式如下：

```
SELECT [ALL|DISTINCT] <目标列表达式>[,<目标列表达式>]…
FROM <表或视图名>[,<表或视图名>]…
[WHERE <条件表达式>]
[GROUP BY <列名1> [HAVING <条件表达式>]]
[ORDER BY <列名2> [ASC|DESC]];
```

在这个语句中，主要考查各种条件，包括GROUP BY、HAVING等，以及考查IN和EXISTS的区别。另外，还要掌握有关集函数的应用。

（1）集函数

常用的集函数主要有：

COUNT（[DISTINCT|ALL]*）：统计元组个数。

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（必须是数值型）。

AVG（[DISTINCT|ALL]<列名>）：计算一列值的平均值（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。

（2）GROUP BY

GROUP BY指定用来放置输出行的组。如果SELECT子句"目标列表达式"中包含聚合函数，则GROUP BY将计算每组的汇总值。指定GROUP BY时，选择列表中任意非聚合表达式内的所有列都应包含在GROUP BY列表中，或者GROUP BY表达式必须与选择列表表达式完全匹配。

如果没有GROUP BY子句，则 SQL 列函数应用程序返回一行数据。当使用GROUP BY 时，会对每个组运用函数，所以所返回的行数与分组数相同。

当使用GROUP BY子句时，SQL 将选出的行按照是否符合表达式的值或者是否符合某一列或多列的值进行分组。接下来，SQL 处理每一组，从而为每组生成一行结果。在GROUP BY子句中，可以指定一列或多列或者表达式来对行进行分组。在SELECT语句中指定的项具有由行组成的每一组的属性，而不具备表中或视图中单个行的属性。

例如：EMPLOYEE 表有几组行，每一组中的这些行描述了某个特定部门的成员。为了查出每个部门中人员的平均薪水，可以写这样的 SQL 语句：

SELECT WORKDEPT, DECIMAL（AVG（SALARY），5,0）

FROM EMPLOYEE

GROUP BY WORKDEPT;

生成的结果被分成了几行，每行表示一个部门。

如果在GROUP BY子句中指定的列为空值，则会生成只有一行的结果，行中数据为空值。也可以将行按照多列或按照表达式进行分组。例如：您可能使用CORPDATA.EMPLOYEE表编写一条查找每个部门男性员工和女性员工平均薪水的SELECT语句。为了实现这一点，可以这样做：

SELECT WORKDEPT, SEX, DECIMAL（AVG（SALARY），5,0） AS AVG_WAGES

FROM CORPDATA.EMPLOYEE

GROUP BY WORKDEPT, SEX;

（3）HAVING

HAVING子句指定组或聚合应满足的搜索条件。当 HAVING 与 GROUP BY ALL 一起使用时，HAVING子句优于ALL.

HAVING 子句对 GROUP BY 子句设置条件的方式与 WHERE 子句和 SELECT 语句交互的方式类似。WHERE 子句搜索条件在进行分组操作之前应用；而HAVING搜索条件在进行分组操作之后应用。HAVING语法与WHERE语法类似，但HAVING可以包含聚合函数。HAVING 子句可以引用选择列表中出现的任意项。

下面的查询得到本年度截止到目前的销售额超过40000的出版商：

SELECT pub_id, total = SUM（ytd_sales）

FROM titles

GROUP BY pub_id

HAVING SUM（ytd_sales） > 40000;

HAVING 子句用来从分组的结果中筛选行。对于可以在分组操作之前或之后应用的搜索条件，在 WHERE 子句中指定它们更有效。这样可以减少必须分组的行数。应当在 HAVING 子句中指定的搜索条件只是那些必须在执行分组操作之后应用的搜索条件。

（4）IN和EXISTS的区别

从查询形式上来看，IN和EXISTS的主要区别在于，IN一般用于单个属性的条件，而EXISTS用于整个元组的条件。例如：

SELECT *

FROM table_A

WHERE attr IN

（SELECT attr FROM table_B）；

SELECT *

FROM table_A

WHERE EXISTS

（SELECT ATTR FROM table_B）；

从查询性能上来看，如果子查询得出的结果集记录较少，主查询中的表较大且又有索引时应该用IN.反之，如果外层的主查询记录较少，子查询中的表大，又有索引时使用EXISTS.

其实我们区分IN和EXISTS,主要是造成了驱动顺序的改变（这是性能变化的关键），如果是EXISTS,那么以外层表为驱动表，先被访问。如果是IN,那么先执行子查询。

（5）LIKE与通配符

在WHERE子句的条件表达式中，可以使用算术比较运算符号，对于字符型数据，可以使用LIKE关键词以及通配符。例如，要查找姓李的学生的学号和姓名的SQL语句如下：

SELECT Sno, Sname

FROM Student

WHERE Sname LIKE '李%';

这里的"%"就是一个通配符，代表0个或多个字符。例如，要查找某个属性中包含"希赛"的元组，则可以写成"... LIKE '%希赛%'".

还有一个通配符是"_"（下画线），这个通配符只代表1个字符。例如，要查找某个属性中第2个字和第3个字是"希赛",倒数第2个字和第3个字包含"教育"的元组，则可以写成"... LIKE '_希赛%教育_'".

第 7 章：数据库系统　作者：希赛教育软考学院　来源：希赛网　2014年05月20日

视图操作

7.4.2 视图操作

视图不真正存在数据，只是把定义存于数据字典，在对视图进行查询时，才按视图的定义从基本表中将数据查出。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，但保留了码，则这个视图称为行列子集视图。

视图对应了数据库系统三级模式/两级映象中的外模式，重建视图即是修改外模式及外模式/模式映象，实现了数据的逻辑独立性。在DBMS中，视图的作用如下：

（1）简化用户的操作。

（2）使用户能从多种角度看待同一数据。

（3）对重构数据库提供了一定程度的逻辑独立性。

（4）能够对机密数据提供安全保护。

1.定义视图

建立视图的命令格式如下：

```
CREATE VIEW <视图名>[（<列名>[,<列名>]...）]
AS
```

子查询

[With Check Option]

其中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。

例如，建立信息系学生的视图：

```
CREATE VIEW IS_Student
AS
SELECT Sno,Sname,Sage
FROM Student
WHERE Sdept='IS'
With Check Option;
```

2.查询视图

因为视图没有真实数据，所以对视图的查询要转换为对相应表的查询，这个过程叫视图消解，视图消解过程由DBMS自动完成。

第 7 章：数据库系统　作者：希赛教育软考学院　来源：希赛网　2014年05月20日

完整性约束

7.4.3 完整性约束

数据库的完整性是指数据的正确性和相容性。数据库是否具备完整性关系到数据库系统能否真实地反映现实世界，因此维护数据库的完整性是非常重要的。

数据库的完整性可分为实体完整性、参照完整性和用户定义的完整性。

1.实体完整性

实体完整性是基于主码的，一个主码由一个或多个属性组成。实体完整性要求主码中的任一属性（列）不能为空，所谓空值是"不知道"或"无意义"的值。之所以要保证实体完整性主要是因为在关系中，每一个元组的区分是依据主码值的不同，若主码值取空值，则不能标明该元组的存在。

2.参照完整性

参照完整性是基于外码的，若基本关系R中含有与另一基本关系S的主码PK相对应的属性组FK（FK称为R的外码），则参照完整性要求，对R中的每个元组在FK上的值必须是S中某个元组的PK值，或者为空值。

参照完整性的合理性在于，R中的外码只能对S中主码的引用，不能是S中主码没有的值。如学生和选课表两关系，选课表中的学号是外码，它是学生表的主键，若选课表中出现了某个学生表中没有的学号，即某个学生还没有注册，却已有了选课记录，这显然是不合理的。

对于参照完整性，需要明确以下问题：

（1）外码能否接受空值问题根据实际应用决定。

（2）在被参照关系中删除元组的问题。

级联删除：将参照关系中所有外码值与被参照关系中要删除元组主码值相同的元组一起删除。如果参照关系同时又是另一个关系的被参照关系，则这种删除操作会继续级联下去。

受限删除（一般系统默认）：仅当参照关系中没有任何元组的外码值与被参照关系中要删除元组的主码值相同时，系统才可以执行删除操作，否则拒绝执行删除操作。

置空删除：删除被参照关系的元组时，并将参照关系中相应元组的外码值置为空值。

（3）在参照关系中插入元组的问题。

受限插入：仅当被参照关系中存在相应的元组时，其主码值与参照关系插入元组的外码值相同时，系统才执行插入操作，否则拒绝此操作。

递归插入：首先向被参照关系中插入相应的元组，其主码值等于参照关系插入元组的外码值，然后向参照关系插入元组。

3.用户定义的完整性

实体完整性和参照完整性适用于任何关系数据库系统。除此之外，不同的关系数据库系统根据其应用环境的不同，往往还需要一些特殊的约束条件。用户定义的完整性就是针对某一具体关系数据库的约束条件，它反映某一具体应用所涉及的数据必须满足的语义要求。

如果在一条语句执行完后立即检查，则称立即执行约束；如果在整个事务执行结束后再进行检查，则称延迟执行约束。

4.触发器

触发器（Trigger）是在关系数据库管理系统中应用得比较多的一种完整性保护措施。触发器的功能一般比完整性约束要强得多。一般而言，在完整性约束功能中，当系统检查出数据中有违反完整性约束条件时，则仅给出必要提示以通知用户，仅此而已。而触发器的功能则不仅仅起提示作用，它还会引起系统内自动进行某些操作以消除违反完整性约束条件所引起的负面影响。

在目前数据库中事件一般表示为数据的插入、修改、删除等操作。触发器除了有完整性保护功能外，还有安全性保护功能。

1.非空约束

非空是个规则，是对声明了非空约束的字段限制其取值域不能为空。它不允许在对一列的数据插入或更新时取NULL值。在默认情况下，是所有的列都允许填充NULL值的，只有在列声明了NOT NULL的列值或其他已经包含了NOT NULL约束的列值时，系统强制要求任何时候都不能填充NULL值。非空约束通常会包含在相关的其他约束上，如：主键约束，候选键约束等。因为索引是不对NULL值进行存储的，所以，如果希望使用索引进行表数据的查询，那么必须尽量地使某个要索引的列不包含NULL值。例如：下面语句对emp表的ename字段声明了非空约束。

```
create table emp（
ename varchar（32） not null,
emanager varchar（30）
）；
```

对于表emp,如果已经有了数据，而且ename列中已存在NULL值，则该操作将被禁止。只要将空值替换或将对应的行删除，该语句还是可以被允许的。当然对于不明确的列但又不能为空时，可以使用默认值（default）对列值做默认处理，这样也可以远离NULL值对列的影响。

2.CHECK约束

相对于非空约束，CHECK约束能更灵活地限制某字段取值的值域，客观地说，非空约束是

CHECK约束的一个特例而已。CHECK约束需要设计一个表达式，某条记录的某（些）字段的值，如果使这个表达式为假，则这条记录被禁止，反之，则被允许。如果定义一个CHECK约束设计一个某字段不为空的表达式，那么效果就跟非空约束一样了。CHECK约束设计的表达式具体应用到某条将要进行操作的记录的某（些）字段时，只有三个值--true、false、unknown,表达式值为false时将要进行的操作被禁止，其他的则允许。在CHECK约束的表达式中，有如下要求：

（1）该表达式在使用数据插入或更新操作的值时，必须是可以作出逻辑判断的表达式。

（2）该表达式不可以使用嵌套查询和序列器。

（3）该表达式不可以使用SQL函数。

（4）该表达式不可以使用自定义函数。

（5）该表达式不可以使用DBMS的伪字段（比如Oracle中的rownum、level等）。

下面语句说明如何在创建表的语句中声明CHECK约束。

```
CREATE TABLE Dept_tab（
Deptno  NUMBER（3），
Dname   VARCHAR2（15），
Loc     VARCHAR2（15），
CONSTRAINT Loc_check1 CHECK （loc IN （'北京', '上海', '广州' ））
）；
```

该语句为表Dept_tab中的Loc字段声明了名为Loc_check1的CHECK约束，这个约束要求Loc字段只能填写（'北京', '上海', '广州'）值中的一个，否则，操作将被禁止。

3.主键约束

主键是能唯一标识元组的最小属性集，并且是在数据库中被标记为PRIMARY KEY的属性集。如果某个表的某（些）字段声明为主键，那么这些字段就接受了要遵守如下两条规则。

（1）在这些作为主键的字段中任何字段任何行都不能为空。

（2）在这些作为主键的字段中任何两行之间的组合取值不能重复。

下面语句是在创建表的语句中声明了主键约束。

```
CREATE TABLE dept（
deptno NUMBER PRIMARY KEY,
dname VARCHAR2（30）
）；
```

该语句在表dept中声明了字段deptno为该表的主键，这个主键约束并未命名，它的名称可以由系统自动产生。再看下面的语句：

```
CREATE TABLE emp（
empno NUMBER,
ename VARCHAR2（30），
CONSTRAINT epk PRIMARY KEY （empno）
）；
```

该语句在表emp中声明了字段empno为主键，同时命名该主键约束为epk.

4.候选键约束

能唯一标识元组的最小属性集，但在数据库中未被标记为PRIMARY KEY,这样的属性集我们称为

候选键。这里的候选键实质就是唯一性约束。它的规则要求与主键的规则要求一样，只是未被声明为主键而已。标准SQL并没有直接定义候选键，候选键取值不能为空和不取重复值的约束可以通过UNIQUE NOT NULL来实现，也可以通过UNIQUE KEY来实现。具体要根据不同的DBMS而定。下列语句在创建表中声明了候选键约束。

CREATE TABLE emp（

empno NUMBER,

ename VARCHAR2（30），

CONSTRAINT euk UNIQUE KEY（empno）

）；

该语句为表emp的empno字段声明了一个叫euk的唯一性约束。

5.外键约束

外键是指在关系模式R中作为候选键的属性集，但在关系模式T中不是候选键。则对于T来说，这些属性集称为相对于R的外键。通常外键在数据库中会被标记为FOREIGN KEY字样。在这里，称R为父模式，T为子模式。通俗地说就是父表中的主键字段被子表中某个字段引用，那么子表的这个字段，则应声明为外键，而父表的那个字段可以声明为主键也可以声明为唯一键（候选键）。一旦子表的字段被声明为父表的外键，同样的一个字段在父表中我们称为引用的父键，在子表中称为引用的子键。子键的取值应遵守如下规则，否则将被禁止。

（1）每个子键的取值不能取父键中取值以外的值，但空除外。

（2）对于父键是组合字段的，子键也应是组合字段，父键要求能唯一地标识父表的每一条记录，而子键不要求能唯一标识子表的每一条记录。

（3）对于组合键，子键为空时，所有的键的成员字段都为空，不允许部分为空。

一个子表可以有多个子键，每个子键可以对应不同父表的父键。如果是组合键，那么，同一个字段可能作为不同的外键的成员。当然，子表和父表是相对的，子表如果有唯一键被其他表引用，那么这时的子表对于其他表关于某键，它又是父表。外键的引入，主要是为了表示表间的一对多问题。下面的语句说明了两个表间的外键引用情况。

CREATE TABLE dept（

deptno NUMBER PRIMARY KEY,

dname VARCHAR2（30）

）；

CREATE TABLE emp（

empno NUMBER,

ename VARCHAR2（30），

deptno NUMBER REFERENCES（dept），

CONSTRAINT efk FOREIGN KEY（deptno）REFERENCES（dept.deptno）

）；

这两条语句创建了dept和emp两个表，其中dept表中的deptno是它的主键，它被表emp引用，在emp中也有个同样名字的deptno字段，这个字段被声明为引用dept.deptno字段的外键。那么dept表是父表，deptno构成了引用中父表的父键，emp表是子表，deptno构成了引用中子表的子键。

因为外键是引用父表父键的子表的子键，所以父表的数据是引用的基础。一旦父表数据被修改或删除，引用也会造成相关的影响，表现在子表和子键上，这就需要做出相应的规则来约定父表父键发生改动后子表子键如何来进行维护活动。根据这种活动，我们将子表的外键分为3大类。

（1）禁止更新和删除父键，许多DBMS默认为这一类外键约束。一旦父表的父键被某个子表的子键引用了，那么父表中已有的数据是禁止修改和删除的。

（2）删除或更新父表数据的同时删除或更新子表子键中对应父键取值的行。

（3）删除或更新父表数据的同时将子表中对应父键的子键的取值设置为NULL.

加入这3种外键约束声明的语句可以如下所示：

CREATE TABLE Emp_tab（

FOREIGN KEY（Deptno）REFERENCES Dept_tab

）；

该语句说明外键deptno默认为第一类。

CREATE TABLE Emp_tab（

FOREIGN KEY（Deptno）REFERENCES Dept_tab ON DELETE CASCADE

）；

该语句声明一个第二类的外键。

CREATE TABLE Emp_tab（

FOREIGN KEY（Deptno）REFERENCES Dept_tab ON DELETE SET NULL

）；

该语句声明一个第三类的外键。

第 7 章：数据库系统　作者：希赛教育软考学院　来源：希赛网　2014年05月20日

授权机制

7.4.4 授权机制

通常授权有两种方式，分别是直接方式和间接方式。直接方式是直接地授权给某个用户，间接方式是指先授权给某个角色（role），然后将该角色赋予一个或多个用户。

1.授予操作权限

授予操作权限的语法格式如下：

```
GRANT {object_privilege|ALL [PRIVILEGES]}[（column, ...）],...
on_object_clause TO grantee_clause
[WITH GRANT OPTION] [WITH HIERARCHY OPTION];
```

其中：

GRANT,TO:为授权谓词。

object_privilege:可以选择上节列表中表示模式对象权限的谓词，当然不同的DBMS可能略有差别。

ALL [PRIVILEGES]:指定该选项则指授予用户系统操作权限表中所有的权限。

（...）省略号：指明一条授权语句可以组合不同的权限一次性授予用户。

[（column, ...）]:可选语句只有当授权假定的操作对象是表或视图等时才可选用，表示具体授权操作这些表的哪些列，如果未指定该选项则默认为所有的列，对列操作权限授权时，仅能授予INSERT、UPDATE、REFERENCES这些权限。

on_object_clause:指明具体哪个模式对象，格式为[schema.]object.

grantee_clause:为授权子句。

[WITH GRANT OPTION]:该选项指明被授权的用户将有再授权给别的用户的能力。

[WITH HIERARCHY OPTION]:该选项指明被操作的模式对象的所有子对象也将被授权用户有权限操作。

例如：

```
GRANT SELECT ON emp TO user001;
```

该语句将表emp的SELECT权限授予了用户user001.

2.取消操作权限

取消操作权限的语法格式如下：

```
REVOKE {object_privilege|ALL [PRIVILEGES]}[（column, ...）],...
on_object_clause FROM grantee_clause
[CASCADE CONSTRAINTS] [FORCE];
```

其中：

REVOKE,FROM:为授权谓词。

object_privilege:可以选择上节列表中表示模式对象权限的谓词，当然，不同的DBMS可能略有差别。

ALL [PRIVILEGES]:指定该选项则指取消用户系统操作权限表中所有的权限。

（...）省略号：指明一条消权语句可以组合不同的权限一次性取消。

[（column, ...）]:可选语句只有当消权假定的操作对象是表或视图等时才可选用，表示具体消权操作这些表的哪些列，如果未指定该选项则默认为所有的列，对列操作权限取消时，只有取消INSERT、UPDATE、REFERENCES这些权限。

on_object_clause:指明具体哪个模式对象，格式为[schema.]object.

grantee_clause:为消权子句。

[CASCADE CONSTRAINTS]:只有在取消REFERENCES和ALL PRIVILEGE权限时，才可选用该选项，使用该选项后，将删除原来定义的在引用权限之上的所有相关引用约束。

[FORCE]:只有在取消EXECUTE权限时，才选用该选项，使用该选项后将执行权限取消的同时，如果其中依存了许多表及其他对象的数据，将被置为不可用或无效。

例如，

REVOKE SELECT ON emp FROM user001;

该语句将用户user001对表emp的SELECT权限取消。

函数依赖与范式

7.5 函数依赖与范式

本节的主要考点包括判断关系模式的主键、属于哪种范式、最小函数依赖集、模式分解等。

上一节　本书简介　下一节

第 7 章：数据库系统　　作者：希赛教育软考学院　来源：希赛网　2014年05月20日

函数依赖与范式

7.5.1 函数依赖与范式

在数据库设计过程中，往往遇到数据冗余、修改异常、插入异常和删除异常等问题，为了设计一个好的数据库，人们定义了一些好的关系模式标准，称它们为规范的关系模式（简称范式，NF）。目前共定义了多个范式，分别为1NF、2NF、3NF、BCNF、4NF和5NF.但实际应用中，一般只要达到3NF.

函数依赖是数据库的一种约束，决定了关系模式属于哪种范式。为了理解方便，下面我们先介绍函数依赖的有关概念。

1.函数依赖

设R（U）是属性U上的一个关系模式，X和Y是U的子集，r为R的任一关系，如果对于r中的任意两个元组u,v,只要有u[X]=v[X],就有u[Y]=v[Y],则称X函数决定Y,或称Y函数依赖于X,记为X→Y.

如果有X→U在关系模式R（U）上成立，并且不存在X的任一真子集X′使X′→U成立，那么称X是R的一个候选键（候选码）。也就是X值唯一决定关系中的元组。一个表可以有多个候选键，可以任意指定其中的一个为主键（主码），其他的候选键称作备用键。如果一个关系的所有属性组合成一个键，则称该关系为全码关系。属于任何一个候选键的属性称为主属性，不属于任何一个候选键的属性称为非主属性。

如果关系模式R中的属性或属性组非该关系的键（备用键或主键），而是其他关系的键，那么该属性或属性组对关系模式R而言是外键。

在R（U）中，如果X→Y,并且对于X的任何一个真子集X′，都有X′→Y不成立，则称Y对X完全函数依赖。若X→Y,但Y不完全函数依赖于X,则称Y对X部分函数依赖。

在R（U）中，如果X→Y（Y不是X的真子集），且Y→X不成立，Y→Z,则称Z对X传递函数依赖。

设U是关系模式R的属性集，F是R上成立的只涉及到U中属性的FD集，则有以下三条推理规则：

（1）自反性：若$Y \subseteq X \subseteq U$，则X→Y在R上成立。

（2）增广性：若X→Y在R上成立，且$Z \subseteq U$，则XZ→YZ在R上成立。

（3）传递性：若X→Y和Y→Z在R上成立，则X→Z在R上成立。

这里XZ,YZ等写法表示X∪Z,Y∪Z.上述三条推理规则是函数依赖的一个正确的和完备的推理系统。根据上述三条规则还可以推出其他三条常用的推理规则（Armstrong推理规则）：

（1）并规则：若X→Y和X→Z在R上成立，则X→YZ在R上成立。

（2）分解规则：若X→Y在R上成立，且$Z \subseteq Y$，则X→Z在R上成立。

（3）伪传递规则：若X→Y和WY→Z在R上成立，则WX→Z在R上成立。

在关系模式R（U,F）中被F逻辑蕴含的函数依赖全体叫做F的闭包，记做F+.

设F为属性集U上的一组函数依赖，X是U的子集，那么相对于F属性集X的闭包用X+表示，它是一个从F集使用推理规则推出的所有满足X→A的属性A的集合：

X+={属性A|X→A在F+中}

如果G+=F+,就说函数依赖集F覆盖G（F是G的覆盖，或G是F的覆盖），或F与G等价。

2.范式

有了上述的函数依赖概念之后，我们再介绍范式的概念。

（1）第一范式（1NF）：如果关系模式R的每个关系r的属性值都是不可分的原子值，那么称 R是1NF的模式，r是规范化的关系。关系数据库研究的关系都是规范化的关系。

（2）第二范式（2NF）：若关系模式R是1NF,且每个非主属性完全函数依赖于候选键，那么称R是2NF的模式。也就是说，只要有任何1个非主属性部分依赖于任何1个候选键，则就不是2NF.

（3）第三范式（3NF）：如果关系模式R是1NF,且每个非主属性都不传递依赖于R的候选键，则称R是3NF的模式。也就是说，只要有任何1个非主属性传递依赖于任何1个候选键，则就不是3NF.

（4）BC范式（BCNF）：若关系模式R是1NF,且每个属性都不传递依赖于R的候选键，那么称R是BCNF的模式。也就是说，只要有任何1个属性（注意：3NF中指的是非主属性，而这里指的是任何属性，也就是包括主属性）传递依赖于任何1个候选键，则就不是BCNF.

根据各范式的定义，成立如下关系：$1NF \supset 2NF \supset 3NF \supset BCNF$。

下面通过一个例子来说明范式的应用。

设有一图书管理数据库，其关系模式是R0（L#,B#,BNAME,BPRICE,BPUB），其属性分别表示个人借书证号、书号、书名、书价、图书出版社。按照通常图书管理方式的理解（假设不同的出版社所出的书名不能相同，不同的书的价格不能相同），则在该关系模式中，各属性都是原子属性，且有B#-> BNAME,B#->BPRICE,B#->BPUB,BNAME->BPRICE和BNAME->BPUB.所以根据以上信息，R0的候选键为（L#,B#），且存在部分依赖情况，所以R0不是2NF,当然就更加不是3NF了。

如果把R0分解成两个关系模式，则每个关系模式都必须包含L#或B#,所以，正确的分解为R1=（L#,B#）和R2=（B#,BNAME,BPRICE,BPUB）。这样分解以后，R1是全键模式，属于BCNF.而R2的候选键为B#,所以，R2是2NF.但因为有B#->BNAME,BNAME->BPRICE成立，即BPRICE传递依赖于候选键，所以R2不是3NF.

如果把R0分解成三个模式R3（L#,B#）、R4（B#,BNAME）、R5（BNAME,BPRICE,BPUB），则显然都是BCNF了。

3.最小依赖集

如果函数依赖集F满足下列条件，则称F为一个极小函数依赖集，也称为最小依赖集或最小覆盖。

（1）F中任一函数依赖的右部仅含有一个属性。

（2）F中不存在这样的函数依赖X→A,使得F与F-{X→A}等价。

（3）F中不存在这样的函数依赖X→A,X有真子集Z使得F-{X→A}∪{Z→A}与F等价。

一个常见的问题是给定一个关系模式的函数依赖集后，如何求候选键？通常的方法是这样的：

首先通过函数依赖集，求得它的最小函数依赖集A.然后，把A中的所有函数依赖的左边属性组成一个集合B.最后，再在A中找其左边和右边都在B中的函数依赖，把其右边的元素从B中删除。这样，B中剩下的元素就是候选键。要注意的是一个表的候选键不一定是唯一的。

例如：设关系模式R<U,F>,其中U={H,I,J,K,L},若F={H→IJ,J→K,IJK→L,L→H,L→K},则求F的最小函数依赖集的过程如下：

（1）在F中，因为有J→K和IJK→L,所以可以合并成IJ→L,即得到{ H→IJ,J→K,IJ→L,L→HK }.

（2）把第（1）步得到的函数依赖集化简（使每个函数依赖的右部只包含一个属性），进一步得到{ H→I,H→J,J→K,IJ→L,L→H,L→K}.

（3）在第（2）步得到的函数依赖集中，有{ L→H,H→J,J→K},由传递性规则，可以推导出L→K,因此，L→K是多余的，需要删除，最后得到最小函数依赖集为{ H→I,H→J,J→K,IJ→L,L→H}.

在最小函数依赖集中，所有函数依赖的左边属性的并集为{HIJL},下面分三种情况讨论：

（1）因为有L→H,H→I,H→J,所以可选候选键为{L}.

（2）因为有H→I,H→J,IJ→L,所以可选候选键为{H}.

（3）因为有IJ→L,J→K,L→H,所以可选候选键为{IJ}.

因此，关系模式R的候选键有3个，分别为{L}、{H}和{IJ},非主属性为{K}.因为有J→K成立，所以K是对候选键{IJ}的部分依赖，故关系模式R不是第2NF.

第 7 章：数据库系统　作者：希赛教育软考学院　来源：希赛网　2014年05月20日

模式分解

7.5.2 模式分解

将一个关系模式分解成范式的过程称为数据库的规范化。数据冗余是产生数据库操作异常的原因，而数据之间存在的相互关系（或叫数据依赖）是产生数据冗余的原因。因此，关系数据库的规范化理论就是要消除产生数据冗余的某些数据依赖，以设计一个好的数据库。

设有关系R,其中R_1、R_2、…、R_k是R的子集，设有关系模式R_1、R_2、…、R_k的集合ρ={R_1、R_2、…、R_k}并且有$R=R_1 \cup R_2 \cup \ldots \cup R_k$,那么，用ρ代替R的过程称为关系模式的分解。

设数据库模式ρ={R_1、R_2、…、R_k}是关系模式R的一个分解，F是R上的函数依赖集，ρ中每个模式Ri上的FD集是Fi.如果{R_1、R_2、…、R_k}与F是等价的（即相互逻辑蕴涵），那么称分解ρ保持FD.如果分解不能保持FD,那么ρ的实例上的值就可能有违反FD的现象。

关系模式R,分解成关系模式ρ={R_1、R_2、…、R_k},F是R上的一个函数依赖集。如果对R中满足F的每一个关系r都有$r = \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \ldots \bowtie \pi_{R_k}(r)$，则称分解后的关系模式集ρ是关系R的无损连接分解。

如果把关系模式分解为2个子关系模式，则可采用下列方法判定分解是否是无损连接分解：

设ρ={R_1,R_2}是R的一个分解，F是R上的FD集，那么分解ρ相对于F是无损连接分解的充分必要条件是（$R_1 \cap R_2$）→（R_1-R_2）或（$R_1 \cap R_2$）→（R_2-R_1）。

例如：设关系模式R（ABC）上成立的FD集为{A→B},ρ1={AB,BC}为R的一个分解，那么，因为

$R_1 \cap R_2=\{B\}, R_1-R_2=A, R_2-R_1=C$,因此分解ρ1是有损连接分解，但它保持函数依赖。

若R（ABC）上成立的FD集为{A→C,B→C},ρ2={AB,AC}为R的一个分解，那么因为R1∩R2={A},$R_1-R_2=\{B\}, R_2-R_1=\{C\}$,A→C成立，所以分解ρ2是无损连接分解，但它不保持函数依赖（丢失了B→C,因为属性B和C被分解到两个模式中去，不可能产生B→C的函数依赖关系）。

若R（ABC）上成立的FD集为{B→C},ρ3={AB,BC}为R的一个分解，那么，因为$R_1 \cap R2=\{B\}, R_1-R_2=\{B\}, R_2-R_1=\{C\}$,B→C成立，所以分解ρ3是无损连接分解，且它保持函数依赖。

如果把关系模式分解为3个以上的子模式，则上述方法就不起作用了。下面给出一个通用的判别方法。

设关系模式$R=A_1 \ldots A_n$,R上成立的FD集F,R的一个分解$\rho=\{R_1, \ldots, R_k\}$.无损连接分解的判断方法如下：

（1）构造一张k行n列的表格，每列对应一个属性A_j（$1 \le j \le n$），每行对应一个模式R_i（$1 \le i \le k$）。如果A_j在R_i中，那么在表格的第i行第j列处填上符号a_j,否则填上符号b_{ij}.

（2）把表格看成模式R的一个关系，反复检查F中每个FD在表格中是成立，若不成立，则修改表格中元素。修改方法如下：对于F中一个FD X->Y,如果表格中有两行在X分量上相等，在Y分量上不相等，那么把这两行在Y分量上改成相等。如果Y的分量中有一个是a_j,那么另一个也改成a_j;如果没有a_j,那么用其中的一个b_{ij}替换另一个（尽量把下标ij改成较小的数）。一直到表格不能修改为止。

（3）若修改的最后一张表格中有一行全是a,即$a_1, a_2, \ldots, a_n$,那么ρ相对于F是无损连接分解，否则是损失连接分解。

例如：设关系模式Student（Sno,Sname,Cno,Cname,Grade,Tname,Taddr）的属性分别表示学号、学生姓名、课程号、课程名、成绩、任课教师名和教师地址。其中一个学生可以选若干门课程，一个教师可以讲授若干门课程，一门课程可以由多个教师讲授，对于同一门课程，学生只能选定一个教师讲授的课程，教师不会重名。

根据以上描述，我们可以知道，关系模式Student的函数依赖集FD={Tname→Taddr,Sno→Sname,Cno→Cname,（Sno,Cno）→（Grade,Tname）}.下面，我们判断分解{（Sno,Sname），（Sno,Cno,Grade,Tname,Taddr），（Cno,Cname）}是否是无损连接分解。

根据上述判断方法，我们列出分解的初始表如表7-15所示。

表7-15 初始表

	Sno	Sname	Cno	Cname	Grade	Tname	Taddr
(Sno，Sname)	a1	a2	b13	b14	b15	b16	b17
(Sno，Cno，Grade，Tname，Taddr)	a1	b22	a3	b24	a5	a6	a7
(Cno，Cname)	b31	b32	a3	a4	b35	b36	b37

根据Tname→Taddr,因为在Tname列中没有两行相等，所以不需要修改；根据Sno→Sname,因为Sno列的第一行和第二行相等，可使b22=a2;根据Cno→Cname,因为Cno列的第二行和第三行相等，可使b24=a4.根据（Sno,Cno）→（Grade,Tname），因为在Sno和Cno列中没有相等的两行，所以也不要修改。修改后的结果如表7-16所示。

表7-16 最后结果

	Sno	Sname	Cno	Cname	Grade	Tname	Taddr
(Sno，Sname)	a1	a2	b13	b14	b15	b16	b17
(Sno，Cno，Grade，name，Taddr)	a1	a2	a3	a4	a5	a6	a7
(Cno，Cname)	B31	B32	a3	a4	b35	b36	b37

表7-16的第二行完全是a的形式，所以上述分解是无损连接分解。

第 7 章：数据库系统　作者：希赛教育软考学院　来源：希赛网　2014年05月20日

多值依赖与4NF

7.5.3 多值依赖与4NF

在实际应用中，一般需要把关系模式设计为BCNF就可以了，但是为了更好的解决关系模式中存在的一些问题，又引入了多值依赖和4NF的概念。

1.多值依赖

设U是关系模式R的属性集，X,Y是U的子集，Z=U-（X∪Y）。对R（U）的任一关系r,只要r中存在（x,y1,z1）和（x,y2,z2），就也存在（x,y2,z1）和（x,y1,z2），则称多值依赖X→→Y在关系模式R上成立。如果Z为空集，则称平凡的多值依赖。

根据上述定义，我们可以得出多值依赖具有下列性质：

（1）设关系模式R（U），X、Y和Z是U的子集且Z=U-（X∪Y）。若关系模式R满足多值依赖X→→Y,当且仅当对R上的任一关系r,给定一对（x,z）的值，有一组y的值，这组值仅仅决定于x值而与z的值无关。

（2）设关系模式R（U）和R上的关系r,X、Y和Z是U的子集且 Z=U-（X∪Y）。若r满足多值依赖X→→Y,则r满足多值依赖X→→Z.

（3）设关系模式R（U），X、Y和Z是U的子集且Z=U-（X∪Y），当且仅当关系R无损地分解成关系模式R1=X∪Y和R2=X∪Z,则 r满足X→→Y.

多值依赖的推理公理有9个，分别列举如下：

自反性：若$Y \subseteq X$,则X→→Y.

增广性：若X→→Y,$W \subseteq Z$,则X∪Z→→Y∪W.

相加性：若X→→Y、X→→Z,则X→→Y∪Z.

投影性：若X→→Y、X→→Z,则X→→Y-Z、X→→Y∩Z .

传递性：若X→→Y、Y→→Z,则X→→Z-Y.

伪传递性：若X→→Y、Y∪W→→Z,则X∪W→→Z-（Y∪W）。

互补性（对称性）：若X→→Y、Z=U-（X∪Y），则X→→Z.

重复性：若X→Y,则X→→Y.

结合性： 若X→→Y,Z→W,其中$W \subseteq Y$和 Y∩Z=Φ，则X→W.

多值依赖与函数依赖比较，具有下面两个基本的区别：

（1）多值依赖的有效性与属性集的范围有关。若X→→Y在U上成立，则在U的子集W（要求X∪Y是W的子集）上一定成立；反之则不然。这是因为多值依赖的定义中不仅涉及属性组X和Y,而且

涉及U中其余属性Z.

（2）若函数依赖X→Y在R（U）上成立，则对于任何Y的子集Y',都有X→Y'成立。但多值依赖不行。

2.4NF

设R是一个关系模式，D是R上的多值依赖集合。如果D中成立非平凡多值依赖X→→Y时（$Y \subseteq X$），X必是R的候选键，那么称R是第四范式（4NF）的模式。若R（U）∈4NF,则R（U）∈BCNF.

例如：某高等学校为描述该校每个系有哪些教师和哪些学生，用关系模式DeptInfo（DeptName,Teacher,Sname）来表示系（DeptName）、教师（Teacher）和学生名（Sname）三者之间的关系。如表7-17所示。

表7-17 关系模式DeptInfo

DeptName	Teacher	Sname
通信工程系	张锋华	李小明
通信工程系	张锋华	王方方
通信工程系	张锋华	刘小宾
通信工程系	朱大红	李小明
通信工程系	朱大红	王方方
通信工程系	朱大红	刘小宾
…	…	…
计算机系	王山河	刘平江
计算机系	王山河	林程红
计算机系	张计林	刘平江
计算机系	张计林	林程红
…	…	…

关系模式DeptInfo的唯一候选键是{Deptname, Teacher},且没有属性对候选键的部分依赖和传递函数依赖，因此，关系模式DeptInfo是BCNF.

在关系模式DeptInfo中，对于属性（DeptName,Sname）上的一个值（通信工程系，李小明），就有Teacher上的一组值{张锋华，朱大红}与之对应，且这组值仅仅决定于属性DeptName上的值，而与Sname上的值无关，也就是说对于（DeptName,Sname）上的另一个值（通信工程系，王方方），它仍然对应于Teacher上的同一组值{张锋华，朱大红},尽管这时Sname的值已经从"李小明"变成了"王方方".因此Teacher多值依赖于DeptName,即DeptName→→Teacher.

但是关系模式DeptInfo（DeptName,Teacher,Sname）不属于 4NF.因为这个关系模式的唯一候选键是{DeptName,Teacher},而多值依赖DeptName→→Teacher不包含候选键。

我们可以把DeptInfo分解为DeptTeacher（DeptName,Teacher）和DeptStudent（DeptName,Sname），这两个关系模式都不存在非平凡的多值依赖，各有一个平凡的多值依赖DeptName→→Teacher和DeptName→→Sname.注意到DeptName是唯一候选键，所以关系模式DeptTeacher和DeptStudent都是4NF的。

函数依赖和多值依赖是两种最重要的数据依赖。如果只考虑函数依赖，则属于BCNF的关系模式规范化程度已经是最高的了。如果考虑多值依赖，则属于4NF的关系模式规范化程度是最高的。

7.6 E-R模型设计

本知识点主要考查E-R图的画法，各实体之间的关系，如何消除冲突，E-R图向关系模式的转换等。

上一节　本书简介　下一节

第 7 章：数据库系统　作者：希赛教育软考学院　来源：希赛网　2014年05月20日

E-R图的画法

7.6.1 E-R图的画法

E-R模型（实体联系模型）简称E-R图，它是描述概念世界，建立概念模型的实用工具。E-R图包括三个要素：

（1）实体（型）：用矩形框表示，框内标注实体名称。

（2）属性：用椭圆形表示，并用连线与实体连接起来。

（3）实体之间的联系：用菱形框表示，框内标注联系名称，并用连线将菱形框分别与有关实体相连，并在连线上注明联系类型。

例如：图7-1就是一个教学系统的E-R图（为了简单起见，省略了部分实体的属性和联系的属性）。

E-R图中的联系归结为三种类型：

（1）一对一联系（1:1）。设A、B为两个实体集。若A中的每个实体至多和B中的一个实体有联系，反过来，B中的每个实体至多和A中的一个实体有联系，称A对B或B对A是1:1联系。注意：1:1联系不一定都是一一对应的关系。可能存在着无对应。例如，在图7-1中，一个班只有一个班主任，一个班主任不能同时在其他班再兼任班主任，由于老师紧缺，某个班的班主任也可能暂缺。

（2）一对多联系（1:n）。如果A实体集中的每个实体可以和B中的几个实体有联系，而B中的每个实体至少和A中的一个实体有联系，那么A对B属于1:n联系。例如，在图7-1中，一个班级有多个学生，而一个学生只能编排在一个班级，班级与学生属于一对多的联系。

（3）多对多联系（m:n）。若实体集A中的每个实体可与和B中的多个实体有联系，反过来，B中的每个实体也可以与A中的多个实体有联系，称A对B或B对A是m:n联系。例如，在图7-1中，一个学生可以选修多门课程，一门课程由多个学生选修，学生和课程间存在多对多的联系。

必须强调指出：有时联系也有属性，这类属性不属于任一实体，只能属于联系。

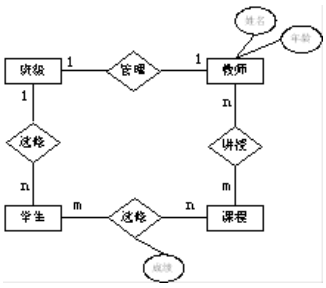

图7-1 某教学系统E-R图

　作者：希赛教育软考学院　来源：希赛网　2014年05月21日

E-R图的集成

7.6.2 E-R图的集成

在数据库的概念结构设计过程中，先设计各子系统的局部E-R图，设计过程可分为以下几个步骤：

确定局部视图的范围。

识别实体及其标识。

确定实体间的联系。

分配实体及联系的属性。

各子系统的局部E-R图设计好后，下一步就是要将所有的分E-R图综合成一个系统的总体E-R图，一般称为视图的集成。视图集成通常有两种方式：

（1）多个局部E-R图一次集成。这种方式比较复杂，做起来难度较大。

（2）逐步集成，用累加的方式一次集成两个局部E-R图。这种方式每次只集成两个局部E-R图，可以降低复杂度。

由于各子系统应用所面临的问题不同，且通常是由不同的设计人员进行局部视图设计，这就导致各个局部E-R图之间必定会存在许多不一致的问题，称之为冲突。因此合并E-R图时并不能简单地将各个局部E-R图画到一起，而是必须着力消除各个局部E-R图中的不一致，以形成一个能为全系统中所有用户共同理解和接受的统一的概念模型。

各局部E-R图之间的冲突主要有三类：

（1）属性冲突：包括属性域冲突和属性取值冲突。属性冲突理论上好解决，只要换成相同的属性就可以了，但实际上需要各部门协商，解决起来并不简单。

（2）命名冲突：包括同名异义和异名同义。处理命名冲突通常也像处理属性冲突一样，通过讨论和协商等行政手段加以解决。

（3）结构冲突：包括同一对象在不同应用中具有不同的抽象，以及同一实体在不同局部E-R图中所包含的属性个数和属性排列次序不完全相同。对于前者的解决办法是把属性变换为实体或实体变换为属性，使同一对象具有相同的抽象。对于后者的解决办法是使该实体的属性取各局部E-R图中属性的并集，再适当调整属性的次序。

另外，实体间的联系在不同的局部E-R图中可能为不同的类型，其解决方法是根据应用的语义对实体联系的类型进行综合或调整。

在初步的E-R图中，可能存在一些冗余的数据和实体间冗余的联系。冗余数据和冗余联系容易破坏数据库的完整性，给数据库维护增加困难，应当予以消除。消除冗余的主要方法为分析方法，即以数据字典和数据流图为依据，根据数据字典中关于数据项之间逻辑关系的说明来消除冗余。

E-R图向关系模式的转换

7.6.3 E-R图向关系模式的转换

E-R图向关系模式的转换规则如下：

（1）一个实体转换为一个关系模式，实体的属性就是关系的属性，实体的码（关键字）就是关系的码。

（2）一个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。如果转换为一个独立的模式，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键。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。

（3）一个1:n联系可以转换为一个独立的关系模式，也可以与任意n端对应的关系模式合并。如果转换为一个独立的模式，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1端关系模式的码和联系本身的属性。

（4）一个m:n联系转换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。

（5）三个以上实体间的一个多元联系可以转换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。

另外，还有4种情况是需要特别注意的：

（1）多值属性的处理。如果E-R图中某实体具有一个多值属性，则应该进行优化，把该属性提升为一个实体。或者在转化为关系模式时，将实体的码与多值属性单独构成一个关系模式。

（2）BLOB型属性的处理。典型的BLOB是一张图片或一个声音文件，由于它们的容量比较大，必须使用特殊的方式来处理。处理BLOB的主要思想就是让文件处理器（如数据库管理器）不去理会文件是什么，而是关心如何去处理它。因此，从优化的角度考虑，应采用的设计方案是将BLOB字段与关系的码独立为一个关系模式。

（3）派生属性的处理。因为派生属性可由其他属性计算得到，因此，在转化成关系模式时，通常不转换派生属性。

（4）在对象-关系数据模型中，本章的"关系模式"就对应"类",关系模式的属性就对应类的属性。

事务处理

7.7 事务处理

本知识点主要考查事务的性质、封锁机制等问题。

第 7 章：数据库系统 作者：希赛教育软考学院 来源：希赛网 2014年05月21日

事务的概念

7.7.1 事务的概念

据库管理系统运行的基本工作单位是事务，事务是用户定义的一个数据库操作序列，这些操作序列要么全做要么全不做，是一个不可分割的工作单位。事务具有以下特性（ACID特性）：

（1）原子性（Atomicity）：事务是数据库的逻辑工作单位，事务的所有操作在数据库中要么全做要么全都不做。

（2）一致性（Consistency）：事务的执行使数据库从一个一致性状态变到另一个一致性状态。

（3）隔离性（Isolation）：一个事务的执行不能被其他事务干扰。

（4）持续性（永久性）（Durability）：指一个事务一旦提交，它对数据库的改变必须是永久的，即便系统出现故障时也是如此。

事务通常以BEGIN TRANSACTION（事务开始）语句开始，以COMMIT或ROLLBACK语句结束。COMMIT称为"事务提交语句",表示事务执行成功地结束，把事务对数据库的修改写入磁盘（事务对数据库的操作首先是在缓冲区中进行的）。ROLLBACK称为"事务回退语句",表示事务执行不成功地结束，即把事务对数据库的修改进行恢复。

从终端用户来看，事务是一个原子，是不可分割的操作序列。事务中包括的所有操作要么都有做，要么都不做（就效果而言）。事务不应该丢失，或被分割地完成。

第 7 章：数据库系统 作者：希赛教育软考学院 来源：希赛网 2014年05月21日

封锁机制

7.7.2 封锁机制

在多用户共享系统中，许多事务可能同时对同一数据进行操作，称为"并发操作",此时数据库管理系统的并发控制子系统负责协调并发事务的执行，保证数据库的完整性不受破坏，同时避免用户得到不正确的数据。

封锁机制

1.并发操作的问题

数据库的并发操作带来的问题：丢失更新问题，不一致分析问题（读过时的数据），依赖于未提交更新的问题（读了"脏"数据）。这三个问题需要DBMS的并发控制子系统来解决。

入同一数据并修改，T2提交的结果破坏了T1

（2）读过时的数据（不可重复读）：事务T1读取某一数据，事务T2读取并修改了同一数据，T1为了对读取值进行校对再读此数据，得到了不同的结果。例如：T1读取B=100,T2读取B并把B改为200,T1再读B得200与第一次读取值不一致。

（3）读"脏"数据。事务T1修改某一数据，事务T2读取同一数据，而T1由于某种原因被撤销，则T2读到的数据就为"脏"数据，即不正确的数据。例如：T1把C由100改为200,T1读到C为200,而事务T1由于被撤销，其修改宣布无效，C恢复为原值100,而T2却读到了C为200,与数据库内容不一致。

2.封锁技术

处理并发控制的主要方法是采用封锁技术。有两种封锁：X封锁和S封

（1）排他型封锁（简称X封锁）：如果事务T对数据A（可以是数据项 … 数据集以至整个数据库）实现了X封锁，那么只允许事务T读取和修改数据A,其他事务要等 X封锁以后，才能对数据A实现任何类型的封锁。可见X封锁只允许一个事务独锁某个数据，具有排他性。

（2）共享型封锁（简称S封锁）：X封锁只允许一个事务独锁和使用数 求太严。需要适当从宽，例如可以允许并发读，但不允许修改，这就产生了S封锁概念。S封锁的含义是如果事务T对数据A实现了S封锁，那么允许事务T读取数据A,但不能修改数据A,在所有S封锁解除之前决不允许任何事务对数据A实现X封锁。

所谓封锁的粒度即是被封锁数据目标的大小，在关系数据库中封锁粒度有属性值、属性值集、元组、关系、某索引项（或整个索引）、整个关系数据库、物理页（块）等几种。

封锁粒度小则并发性高，但开销大；封锁粒度大则并发性低，但开销小，综合平衡照顾不同需求以合理选取适当的封锁的粒度是很重要的。

3.封锁协议

在多个事务并发执行的系统中，主要采取封锁协议来进行处理。

（1）一级封锁协议：事务T在修改数据R之前必须先对其加X锁，直到事务结束才释放。一级封锁协议可防止丢失修改，并保证事务T是可恢复的。但不能保证可重复读和不读"脏"数据。

（2）二级封锁协议：一级封锁协议加上事务T在读取数据R之前先对其加S锁，读完后即可释放S锁。二级封锁协议可防止丢失修改，还可防止读"脏"数据。但不能保证可重复读。

（3）三级封锁协议：一级封锁协议加上事务T在读取数据R之前先对其加S锁，直到事务结束才释放。三级封锁协议可防止丢失修改、防止读"脏"数据与数据重复读。

（4）两段锁协议：所有事务必须分两个阶段对数据项加锁和解锁。其中扩展阶段是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；收缩阶段是在释放一个封锁之后，事务不能再申请和获得任何其他封锁。若并发执行的所有事务均遵守两段封锁协议，则对这些事务的任何并发调度策略都是可串行化的。遵守两段封锁协议的事务可能发生死锁。

4.死锁

采用封锁的方法固然可以有效防止数据的不一致性，但封锁本身也会产生一些麻烦，最主要就是死锁问题。所谓死锁即是多个用户申请不同封锁，由于申请者均拥有一部分封锁权而又需等待另

外用户拥有的部分封锁而引起的永无休止的等待。一般讲，死锁是可以避免的，目前采用的办法有如下几种：

（1）预防法：此种方法即是采用一定的操作方式以保证避免死锁的出现，顺序申请法，一次申请法等即是此类方法。所谓顺序申请法即是对封锁对象按序编号。用户申请封锁时必须按编号顺序（从小到大或反之）申请，这样能避免死锁发生。所谓一次申请法即是用户在一个完整操作过程中必须一次性申请它所需要的所有封锁，并在操作结束后一次性归还所有封锁，这样世能避免死锁的发生。

（2）解除法：此种方法即是允许产生死锁，并在死锁产生后通过解锁程序以解除死锁。这种方法中需要有两个程序，一是死锁检测程序，用它以测定死锁是否发生，另一是解锁程序，一旦经测定系统已产生死锁则启动解锁程序以解除死锁。

第 8 章：系统开发和运行维护　作者：希赛教育软考学院　来源：希赛网　2014年05月21日

考点分析

第8章　系统开发和运行维护

系统开发和运行维护是软件设计师考试的一个重要考点，同时也是软件设计师考试的核心。上午考试中，有分数为10分左右的系统开发和运行维护方面的知识，下午考试中有1道试题（数据流图设计）属于系统开发和运行维护知识，占15分。

8.1　考点分析

本节把历次考试中系统开发和运行维护的试题进行汇总，得出本章的考点，如表8-1所示。

表8-1 系统开发和运行维护试题知识点分布

考试时间	分数	考查知识点
10.11	8	瀑布模型（1）、需求分析（1）、软件设计（1）、软件测试（1）、项目管理工具（1）、数据字典（1）、数据流图（1）、CMM（1）
11.05	8	软件可移植性（1）、系统转换（2）、数据流图（2）、软件测试（2）、瀑布模型（1）
11.11	11	开发模型（1）、测试计划（1）、风险（1）、代码评审（1）、甘特图（1）、软件评审（1）、软件测试（2）、操作手册（1）、开发进度（1）、软件维护（1）
12.05	10	项目管理工具（1）、耦合（2）、数据流图（1）、内聚（1）、项目风险（1）、软件质量（1）、技术评审（1）、软件测试（2）
12.11	11	开发模型（2）、CMM（1）、成本估算（1）、RUP（1）、可移植性（1）、可靠性（1）、用户手册（1）、软件维护（1）、路径覆盖（1）、确认测试（1）
13.05	13	数据流图（1）、原型法（1）、CVS（1）、开发过程（1）、风险分析（1）、程序注释（1）、软件质量（2）、CMM（1）、代码行（1）、判定覆盖（1）、等价类划分（1）、RUP（1）
13.11	7	甘特图（1）、PERT图（1）、开发工具（1）、内聚和耦合（1）、语句覆盖（1）、测试阶段（2）
14.05	11	开发方法（1）、甘特图（1）、风险分析（2）、极限编程（1）、CMM（1）、编程风格（1）、软件维护（1）、开发文档（1）、软件测试分类（2）

根据表8-1,我们可以得出系统开发和运行维护的考点主要有：

（1）系统开发模型：包括瀑布模型、RUP、原型法。