第一部分　数据库系统概论要点

第 1 章　数据库系统概述

随着计算机软硬件的发展，数据管理技术不断地完善，数据管理技术经历了 3 个阶段：人工管理阶段、文件系统阶段和数据库系统阶段。其中重点是数据库系统阶段。

数据库系统的特点有：数据结构化；数据共享性好；数据独立性好；数据存取粒度小；数据库管理系统对数据进行统一的管理和控制；为用户提供了友好的接口。应该通过理解数据库系统的这些主要特点，对数据库系统的全貌有个初步的了解，能够正确地理解和数据库有关的基本术语。下面列出一些主要的术语及其含义。

（1）**数据**：凡是计算机中用来描述事物的记录，可统称为数据。它包括数字、文字、图形、图像、声音等。

（2）**数据模型**：数据模型是一种对客观事物抽象化的表现形式，它要真实地反映现实世界，易于理解，便于实现。

（3）**数据库**：数据库是按照一定的数据模型组织的、长期储存在计算机内、可为多个用户共享的数据的聚集。也可以说，数据库是由数据库管理系统统一管理和控制的数据的聚集。

（4）**数据库管理系统**：专门用于建立和管理数据库的一套软件，介于应用程序和操作系统之间。

（5）**数据库系统**：包括和数据库有关的数据库、DBMS、应用程序以及数据库管理员和用户等。

关系数据库系统是当前数据库系统的主流，也是本书的重点。在主教材第 1 章，要理解关系数据库中的一些基本概念：关系、关系模型、关系数据库系统等。所谓关系，就是一张表。表的各列以属性开始，属性是列的入口；数据以“关系”的形式，也就是以二维表的形式来表示，其数据模型就是所谓的关系模型；关系数据库系统就是以关系模型为基础的数据库系统。关系模型有严格的数学基础，而且简单清晰，便于理解和使用。

数据库管理系统的组成分为 3 个部分。

（1）**查询处理程序**：它不仅负责查询，也负责发出更新数据或模式的请求。

（2）**存储管理程序**：它的功能是从数据库中获得想要查询的数据，并按照更新请求更新相应的信息。

（3）**事务管理程序**：它的作用是保证多个事务并发执行。为此，它要与查询处理程序和存储管理程序互相配合。

第 2 章　数据库建模

主教材第 2 章要求掌握数据库建模的两种基本方法：对象定义语言(ODL)和实体-联系模型(E-R 图)。会用这两种方法建立简单的数据库模型。

ODL(对象定义语言)是用面向对象的术语来说明数据库结构的一种推荐的标准语言，其主要用途是书写面向对象数据库的设计，进而将其直接转换成面向对象数据库管理系统的说明。

在进行 ODL 类的设计时，有 3 种特性需要描述：属性、联系、方法。

在 ODL 中，形式最简单的类的说明包括：关键字 interface(接口)、类的名字、用花括号括起来的类的特性表(特性包括属性、联系和方法)。

要说明一个类，最简单的形式是：

interface＜名字＞{

　　＜特性表＞

}；

两个类之间的联系可分为 3 种不同的类型：多对多、多对一或一对多以及一对一。

E-R 图方法建模也有 3 个主要的部分：实体集、属性、联系。在 E-R 图中，不仅实体集可以有属性，实体集之间的联系也可以有属性。在 E-R 图中，可以用箭头的有无来区别 3 种不同的联系。假如一个联系是从实体集 A 到实体集 B 的多对一的联系，就画一个指向 B 的箭头；假如是 A 到 B 的一对一联系，就画两个箭头，分别指向 A 和 B。

主教材第 2 章中一个很重要的部分就是了解数据库设计的基本原则，并且在数据库设计中能够实践这些原则。这些原则是：真实性、避免冗余、简单性、合理选择元素类型。

类和子类的层次关系反映了现实世界的层次结构。在 ODL 中，定义子类的一般方法是：在类 A 说明的类名 A 之后加上冒号和另一个类 B 的名字，就可以定义类 A 是类 B 的子类。ODL 中的子类和类都与 E-R 模型中的实体集类似。所以在 E-R 图中，假定类 A 是类 B 的子类，类 A 对应于 E-R 图中的实体集 A，类 B 对应于 E-R 图中的实体集 B，为了表示出 A 和 B 之间的关系，用一种称作“属于”(“isa”)的特殊联系将实体集 A 和 B 相连。任何只和子类 A 有关的属性和联系都连到实体集 A 的方框上，而与类 A 和 B 都有关的属性和联系则连到实体集 B 的方框上。还应该了解子类的继承性。

约束建模是数据库建模的重要组成部分。要深入理解键码和引用完整性这两个基本概念。键码是在类的范围内惟一标识一个对象(在 ODL 中)，或者在实体集的范围内惟一标识一个实体(在 E-R 图中)的属性或属性集。一个实体的某个属性(集)值只能引用另一实体确实存在的键码属性(集)值，称为引用完整性。

第 3 章　关系模型和关系运算

关系模型是关系数据库的基础，因此理解关系模型的基本概念就成为学习关系数据库的基础。下面列出关系模型中的一些概念。

① **属性**：属性就是关系标题栏中各列的名字。

② **模式**：关系的名称和关系的属性集称为关系的模式。

③ **元组**：除了关系的标题栏以外，其他各行统称元组。

④ **域**：与关系的每个属性相关的特定的基本类型称为域。

⑤ **关系实例**：给定关系中元组的集合称为该关系的实例。

在用对象定义语言或实体-联系模型对数据库建模之后，面临的问题就是如何将其转换成所需要的关系模型，这是要求掌握的一个重点内容。

(1) 从 ODL 设计到关系设计

ODL 类转换为关系分为属性和联系两方面的转换。对于原子类型的属性，类的每个属性对应于关系的一个属性。对于非原子属性，若为结构，则把结构中的每个元素作为关系的一个属性；若为集合，则按元素的个数把相关的一个元组扩展为多个元组；若为数组，则按元素的个数既可扩展为多个元组，也可扩展为多个属性。

ODL 中联系的转换：若为单值联系，则把相关类中构成键码的属性(集)作为关系的附加属性(集)。若为多值联系，如为集合类型，那么，首先把相关类的键码属性(集)作为关系的附加属性(集)；其次为相关对象集合的每个元素建立一个关系元组。对于联系与反向联系，常用的方法是将其独立出来作为一个新的关系。

ODL 子类转换成关系模式：每个子类都对应于一个关系；这个关系用相应子类的所有特性(包括从超类继承下来的全部特性)来表示。

(2) 从 E-R 图到关系设计

实体集与联系转换为关系：实体集可直接转换为关系，实体集的每个属性都对应于关系中的一个属性。E-R 图中的联系转换为关系时，其属性由两部分组成——与该联系有关的每个实体集的键码属性(集)；该联系本身的属性。

多向联系转换为关系也非常类似。不管一个联系 R 涉及几个实体集，在将 R 转换为关系的时候，只要让 R 的属性包括与其相关的所有实体集的键码属性(集)和它本身的属性即可。

对于已经建成的数据库，人们最关心的是查询其中的数据，在学习具体的查询语言之前，先学 3 种抽象的查询语言，以便为以后的学习打下坚实的基础。其中关系代数部分用到的运算如下：

(1) 关系的 3 种集合运算

对于集合 R 和 S，3 种集合运算定义如下：

R∪S　R 和 S 的并，它是 R 中的元素和 S 中的元素共同组成的集合。

R∩S　R 和 S 的交，它是既出现在 R 中又出现在 S 中的元素组成的集合。

R−S　R 和 S 的差，它是只在 R 中出现，不在 S 中出现的元素组成的集合。

要想对两个关系 R 和 S 进行上述运算，R 和 S 必须满足如下条件：R 和 S 的模式具有相同的属性集；在对 R 和 S 进行集合运算之前，要对 R 和 S 的属性列进行排序，保证两个关系的属性顺序相同。

(2) 投影

投影运算符是 π，该运算作用于关系 R 将产生一个新关系 S，S 只具有 R 的某几个属性列。投影运算的一般表达式如下：

$$\mathbf{S=\pi_{A_1,A_2,\cdots,A_n}(R)}$$

(3) 选择

选择运算符是 σ，该运算符作用于关系 R 也将产生一个新关系 S，S 的元组集合是 R 的一个满足某条件 C 的子集。选择运算的一般表达式为：

$$\mathbf{S=\sigma_C(R)}$$

(4) 笛卡儿积

两个关系 R 和 S 的笛卡儿积记作 $\mathbf{R\times S}$，它是一个新关系，它的关系模式是 R 和 S 的模式的并集，假如 R 和 S 有同名的属性，例如 A，则至少要为其中一个属性重新命名，通常用 R. A 和 S. A 来区分来自 R 的属性 A 和来自 S 的属性 A。R×S 的元组是 R 的一个元组和 S 的一个元组串联而成的长元组。R×S 是把 R 和 S 的元组以所有可能的方式组合起来，因此，R×S 拥有的元组数量应该是 R 的元组数与 S 的元组数的乘积。

(5) 自然连接

两个关系 R 和 S 的自然连接，记作 $\mathbf{R\bowtie S}$，得到一个新关系：它的关系模式是 R 和 S 模式的并集。$R\bowtie S$ 所拥有的元组是这样产生的：假设 $A_1,A_2,\cdots,A_n$ 是 R 和 S 模式中的公共属性，那么如果 R 的元组 r 和 S 的元组 s 在这些属性上取值都相同，r 和 s 组合而成的元组就归入 $R\bowtie S$ 中。

(6) θ 连接

两个关系 R 和 S 基于条件 C 的 θ 连接用

$$\mathbf{R\bowtie_c S}$$

表示，它是这样得到的：先作 R 和 S 的笛卡儿积，然后从 R×S 的元组中选择满足条件 C 的元组集合。显然，R 与 S 的 θ 连接其关系模式应该与 R×S 相同，即为 R 和 S 模式的并集。

(7) 改名

如下运算

$$\mathbf{\rho_{S(A_1,A_2,\cdots,A_n)}(R)}$$

用来把关系 R 改名为关系 S，同时把关系 S 的属性从左至右依次命名为 $A_1,A_2,\cdots,A_n$。假如只想改变关系名，不想改变关系模式中的属性名，那么用如下形式：

$$\mathbf{\rho_S(R)}$$

即可达到目的。经过改名运算所得到的关系 S 具有和关系 R 完全相同的元组。

主教材第 3 章要求熟练掌握关系代数语言的使用，并掌握关系演算语言和关系逻辑语言的使用。

第 4 章　数据库语言 SQL

用结构化查询语言 SQL 对数据库进行查询，是本书的重要内容，也是主教材第 4 章的重点。下面是应该熟练掌握的内容。

(1) 简单查询

SELECT *

FROM 关系名

WHERE 选择条件

(2) 别名

SELECT 属性的原名 AS 别名

(3) 模糊匹配

s LIKE p 和 s NOT LIKE p

其中，s 是普通字符串，p 是模式，表示包含‘%’和‘_’两种任选的具有特殊含义的字符的字符串。p 中的普通字符只和完全相同的字符匹配，而‘%’能和由任意字符组成的任意长度的字符序列匹配，‘_’能和任意一个字符匹配。SQL 的习惯用法是在字符串中用两个连续的单引号来表示一个真正的、单独的单引号，而在%和_之前加上转义符“\”表示这里的%和_是一个普通的字符，而不是为了匹配之用，连续两个\就表示一个真正的反斜线\。

(4) 排序

ORDER BY ＜属性表＞

默认的顺序为升序。也可以在属性的后面加上关键字 DESC 来实现降序输出。SQL 还提供了另一个关键字 ASC 来明确地指定升序输出，不过这个关键字其实没有必要存在，因为默认的顺序就是升序。

(5) 分组

GROUP BY　分组属性列表

如果只希望查询满足一定条件的分组情况，可以使用关键字 HAVING 来选择具有给定条件的分组。

(6) 聚合运算符

① **SUM**　求某列中所有值的和；

② **AVG**　求某列中所有值的平均值；

③ **MIN**　求某列中的最小值；

④ **MAX**　求某列中的最大值；

⑤ **COUNT**　求某列中值的个数。

对关系中某一列的值可以用关键字 SUM、AVG(平均值)、MIN、MAX 或 COUNT 加以汇总，这 5 种处理统称为聚合。可在聚合之前用关键字 GROUP BY 对元组进行分组，从而得到各分组的汇总数据。还可在分组的基础上用关键字 HAVING 给出选择条件，从而取出具有给定条件的分组。

(7) 连接与笛卡儿积

在 FROM 子句中列出涉及到的每个关系，这样 SELECT 和 WHERE 子句就可以引用 FROM 子句中列出的任何关系的属性了。

(8) 嵌套查询

把 SELECT-FROM-WHERE 查询作为子查询用于另一个查询的 WHERE 子句中，称为嵌套查询。可把运算符 **EXISTS**、**IN**、**ALL** 和 **ANY** 用于子查询的结果关系来表示布尔值条件；若对表达式求反，可在适当位置加上 NOT。如果能确信子查询的结果为单值(仅一个元组、一个分量)，就可以在外层查询的 WHERE 子句中直接用比较运算符。

(9) SQL 的关系模型

SQL 实际上把关系看作是元组的包(Bag)(即允许有完全相同的元组)，而不是元组的集合(Set)(不允许有完全相同的元组)。如果要求结果的关系为集合，则需另加关键字 **DISTINCT** 以删除重复的元组。

(10) 集合查询

SQL 使用关键字 **UNION**、**INTERSECT** 和 **EXCEPT** 分别代表并、交和差。这类关键字用在两个查询之间(将每个查询结果作为一个运算对象)，查询必须用圆括号括起来。

当两个查询生成具有相同属性集的关系时，可用关键字 UNION、INTERSECT 和 EXCEPT 进行关系代数中的集合运算：并、交和差，从而实现对多个关系的查询。用上述 3 种运算符将自动删除重复元组，若想保留重复元组，则需在运算符后加上关键字 ALL。

(11) 数据更新

SQL 中的数据更新可利用关键字 **INSERT**(插入新元组)、**DELETE**(删除已有元组)或 **UPDATE**(修改已有元组中的部分分量)来实现。更新语句既可处理单一元组，也可处理一批元组(因格式或给定条件而异)。

(12) 数据定义

对数据库模式中的各个元素进行说明称为数据定义。SQL 的数据定义包括定义和撤销模式(Schema)、表(Table，又称基本表，也就是关系)、视图(View)、索引(Index)等内容。其中，最基本的语句是 **CREATE TABLE**，用于定义关系(SQL 中称为表)模式，指定其属性和类型，规定各种约束。此外，还可用 **ALTER** 语句更改模式，比如，在关系模式中增加或删除属性；或者用 **DROP** 语句完全撤销关系或视图。

(13) 视图

由已有关系定义的、并不实际存在而只是逻辑上的关系称为视图。视图的定义以子查询为基础，这正体现了视图是由已有的关系(或视图)构造而成。对视图可以像对关系一样进行查询，只是具体的执行过程有所不同。

第 5 章　查询优化与并发控制

对于给定的查询选择代价最小的操作序列，使查询过程既省时间，又省空间，具有较高的效率，这就是所谓的查询优化。对于关系数据库系统，用户只需提出“做什么”，而由

系统解决“怎么做”的问题。具体来说，是数据库管理系统中的查询处理程序自动实现查询优化。查询优化是数据库系统实现范畴的一个重要问题。在学会用 SQL 语言对数据库进行查询的基础上，了解数据库系统如何对查询进行优化，有助于在认识上实现从感性到理性的飞跃。

查询优化既有逻辑方法的优化（如关系代数表达式的优化），也有物理方法的优化（如存取路径和存取方法的优化）。由于磁盘的读写比 CPU 的处理在速度上慢得多，因此查询优化的关键就是减少对磁盘的访问。具体策略主要包括：**一元选择**（只针对一个关系、只涉及一个属性的选择）**首先做**；**投影、选择同时做**；**乘积、选择合并做**（把笛卡儿积与随后以选择形式出现的连接条件一起做连接运算）；**索引、排序预先做**。深入理解查询优化的策略；掌握用**关系代数等价变换规则**对关系代数查询表达式进行优化的方法；学会按查询优化的步骤对查询进行优化。

安排事务执行的次序称为调度；利用分时的方法同时处理多个事务，则称为事务的并发调度，也就是并发操作的调度。数据库并发操作可能带来数据不一致的问题，主要有 3 种：**丢失修改、读“脏”数据和不可重复读**。并发控制是数据库系统实现范畴的又一个重要问题。封锁是实现并发控制的主要技术。封锁分**排它锁（X 锁）和共享锁（S 锁）**两种类型。对 X 锁或 S 锁何时申请（加锁）、何时释放（解锁）均有约定的规则，称之为封锁协议。共有**三级封锁协议**可分别达到系统一致性的不同级别，依次解决不丢失修改、不读“脏”数据、可重复读等问题。把事务的执行过程分成申请封锁（加锁）阶段和释放封锁（解锁）阶段，这种规则称为**两段锁协议**。两段锁协议是保证并发调度可串行性的封锁协议。也就是说，凡遵守两段锁协议的任何调度，都是可串行化的调度。

第 6 章　关系数据库设计理论

关系数据库的逻辑设计主要是设计关系模式，而深入理解函数依赖和键码的概念则是设计和分解关系模式的基础。

函数依赖：如果关系 R 中的两个元组在某个特定的属性集 A 上一致，则它们在某个其他特定属性 B 上也一致，就称 B 与 A 之间的关系为函数依赖。具体地说，就是 B 函数依赖于 A，或者 A 函数决定 B。

键码：关系 R 中能函数决定该关系所有属性的最小属性集称为关系 R 的键码。也就是说，键码的任何真子集都不能函数决定该关系的所有属性。

属性的封闭集：对于给定的函数依赖集 S，属性集 A 函数决定的属性的集合称为属性集 A 在依赖集 S 下的封闭集。可利用各种属性组合的封闭集确定关系的键码。要学会计算封闭集。

下面是几个范式的概念。

第二范式：若关系模式 R 属于第一范式，且每个非主属性都完全函数依赖于键码，则 R 属于第二范式。要使关系模式属于第二范式，就要消除非主属性对键码的部分依赖。

第三范式：若关系模式 R 属于第一范式，且每个非主属性都不传递依赖于键码，则 R 属于第三范式。因为部分依赖是传递依赖的特例，所以要使关系模式 R 属于第三范式，

既要消除非主属性对键码的部分依赖，也要消除非主属性对键码的传递依赖。

BC 范式：若关系模式 R 属于第一范式，且每个属性都不传递依赖于键码，则 R 属于 BC 范式。在函数依赖的范畴内，关系模式属于 BC 范式即已实现了模式的彻底分解，消除了数据冗余和更新异常。

模式设计是主教材第 6 章的重点。了解数据冗余和更新异常产生的根源；理解关系模式规范化的途径。准确理解第一范式、第二范式、第三范式和 BC 范式的含义、联系与区别；深入理解模式分解的原则；熟练掌握模式分解的方法，能正确而熟练地将一个关系模式分解成属于第三范式或 BC 范式的模式。

分解原则：无损连接和保持依赖是对关系模式进行分解的两个基本原则。采用规范的方法，使分解后的模式属于第二或第三范式，既能实现无损连接，又能实现保持依赖；使分解后的模式属于 BC 范式，只能保证无损连接，不能绝对保持依赖。

分解方法：模式分解以键码为中心，以函数依赖为基础，针对部分依赖、传递依赖和 BC 范式违例可用三种不同的规范方法，均能使分解后的模式实现无损连接。

方法 1　**部分依赖归子集；完全依赖随键码**。

方法 2　**基本依赖为基础，中间属性作桥梁**。

方法 3　**找违例自成一体，舍其右全集归一**。

了解多值依赖和第四范式的概念，掌握把关系模式分解成属于第四范式的模式的方法。

依赖与联系：函数依赖能描述属性之间多对一的联系（如多个学生在一个系）；“多”函数决定“一”或“一”函数依赖于“多”；属性间一对一联系对应于相互依赖；多对多联系则不存在函数依赖关系。而多值依赖能部分描述属性之间一对多的联系（如一个系有多个学生）：“一”多值决定“多”，或“多”多值依赖于“一”；属性间多对多联系可以多值依赖形式出现（如学生选课，一个学生选多门课）；一对一联系则不存在多值依赖关系。对于多对多（m：n）联系来说，当 n 减少到 1（如限制一个学生最多只能参加一个体育社团）时，多值依赖就演变成为函数依赖，因此，可把函数依赖看作是多值依赖的特例。

第 7 章　数据库设计

数据库的设计既要满足用户的需求，又与给定的应用环境密切相关，因此必须采用系统化、规范化的设计方法，按需求分析、概念设计、逻辑设计、物理设计 4 个阶段逐步深入展开。

需求分析也就是分析用户的要求，是数据库设计的基础。通过调查和分析了解用户的信息需求和处理需求，并以数据流图、数据字典等形式加以描述。

概念设计主要是把需求分析阶段得到的用户需求抽象化为概念模型。概念设计是整个数据库设计的关键。使用 E-R 模型作为概念设计的工具。

逻辑设计就是把概念设计阶段产生的概念模式转换为 DBMS 所支持的数据库逻辑模式。首先，把 E-R 图转换成关系模式：实体转换成关系模式时，属性与键码均一一对应；联系转换成关系模式时，构成连接关系，其键码如下：若联系为 1：1，则每个实体的键码均为其键码；若联系为 1：n，则 n 端实体的键码为其键码；若联系为 m：n，则各实体键

码的组合为其键码。其次，对 E-R 图转换后形成的关系模式进行规范化和优化。

物理设计是为关系模式选择合适的存取方法和存储结构。

学完主教材第 7 章并结合相关的教学实验要求了解简单数据库设计的全过程，基本掌握概念设计和逻辑设计的方法，并了解物理设计的方法。

第 8 章　SQL 系统环境

在编写与数据库有关的应用程序时，通常把 SQL 语言嵌入到程序设计语言中，从而既能访问数据库，又能处理数据，这与前面介绍的 SQL 在使用上和格式上稍有差别。应学会用嵌入式 SQL 对数据库进行增、删、改、查等操作。

游标是主教材第 8 章新引入的概念，应学会利用游标进行查询。

在理解事务的概念之后，学会用嵌入式 SQL 处理简单事务。

SQL2 把表示完整性约束的各种技术作为数据库模式的一部分，从而在定义关系模式时就可以把有关的完整性约束明确地加以说明。

数据库中最重要的约束就是说明某个属性或属性集构成关系的键码。键码约束在 SQL 的建表语句中说明。可用两种方法说明：用关键字 **PRIMARY KEY**；用关键字 **UNIQUE**。

在 SQL 中可以把一个关系的属性或属性集说明为外键码。可用两种方法说明。

(1) 如果外键码为单一属性，则可以在属性名和类型之后加上如下说明：

REFERENCES ＜表＞（＜属性＞）

(2) 在建表语句的属性表之后加上一个或多个表明属性(集)为外键码的说明，格式如下：

FOREIGN KEY＜属性＞REFERENCES ＜表＞（＜属性＞）

检验(CHECK)约束是除了键码约束和外键码约束之外的第 3 种重要的约束。检验约束既可用于属性，也可用于元组。

数据库安全涉及到多方面的问题，我们侧重从用户对数据库的访问权限的角度进行讨论。应初步学会建立、授予和取消权限。

第 9 章　面向对象查询语言

主教材第 9 章首先在对象定义语言 ODL 的基础上介绍方法和范围的概念。类的范围类似于关系名，而类定义则类似于该关系属性的类型说明。

主教材第 9 章的重点是介绍对象查询语言 OQL，要求掌握路径表达式、基本查询表达式的使用，并初步掌握附加的表达式的使用。

OQL 的路径表达式：采用点表示法访问变量的分量。

如果 a 表示属于类 C 的对象，p 是该类的某个特性(即该类的属性、联系或者方法)，那么 a.p 就表示把 p 用于 a 的结果。也就是：

如果 p 是属性，那么 a.p 就是对象 a 的该属性值。

如果 p 是联系，那么 a.p 就是通过联系 p 与 a 相连的对象或者对象的聚集。

如果 p 是方法(或许带参数)，那么 a.p 就是把 p 用于 a 的结果。

利用量词表达式可以检查是否所有的集合成员或者至少有一个集合成员满足某个条件。

(1) 如果 S 中的每个 x 都满足 C(x)，则表达式

FOR ALL x IN S：C(x)

的结果就为真(TRUE)，否则为假(FALSE)。

(2) 如果 S 中至少有一个 x 使 C(x)为真(TRUE)，则表达式

EXISTS x IN S：C(x)

为真(TRUE)，否则为假(FALSE)。

能把 OQL 与面向对象的宿主语言混合编程使 OQL 在开发应用程序方面比 SQL 具有明显优势，要求能初步学会编写简单的程序。

从聚集中提取元素：在 OQL 中，可以用 **ELEMENT** 运算符从单个元素聚集中提取单个元素。还可以访问有多个成员的聚集中的元素，具体方法是：首先用 SELECT - FROM - WHERE 语句中的 **ORDER BY** 子句把集合或包转换成列表，然后用外层的宿主语言的循环程序按顺序访问该列表的每个元素。

SQL3 在 SQL 的基础上扩充了面向对象的概念。要求初步掌握行类型和抽象数据类型的使用。

第 10 章　数据库技术发展动态

主教材第 10 章较全面地介绍了数据库技术的发展动态，学完这一章后应该了解分布式数据库、并行数据库、多媒体数据库、主动数据库和数据仓库的概念及特点，并对各种数据库的结构有一定的了解。