0000000000

第 1 章　数据库系统概述

习题参考答案

税务局使用数据库存储纳税人（个人或公司）　　信息、 纳税人缴纳税款信息等。　典型的数据处理包括纳税、退税处理、统计各类纳税人纳税情况等。

银行使用数据库存储客户基本信息、　　客户存贷款信息等。　典型的数据处理包括处理客户存取款等。

超市使用数据库存储商品的基本信息、会员客户基本信息、客户每次购物的详细清单。

典型的数据处理包括收银台记录客户每次购物的清单并计算应交货款。

1.2　DBMS　是数据库管理系统的简称，是一种重要的程序设计系统。它由一个相互关联的数据集合和一组访问这些数据的程序组成。

数据库是持久储存在计算机中、　有组织的、 可共享的大量数据的集合。　数据库中的数据按一定的数据模型组织、　描述和存储，可以被各种用户共享，　具有较小的冗余度、　较高的数据独立性，并且易于扩展。

数据库系统由数据库、　DBMS （及其开发工具） 、应用系统和数据库管理员组成。

数据模型是一种形式机制，用于数据建模，描述数据、数据之间的联系、数据的语义、数据上的操作和数据的完整性约束条件。

数据库模式是数据库中使用数据模型对数据建模所产生设计结果。对于关系数据库而言，数据库模式由一组关系模式构成。

数据字典是　DBMS　维护的一系列内部表，用来存放元数据。所谓元数据是关于数据的数据。

1.3 DBMS　提供如下功能：

（1）　数据定义：提供数据定义语言　　DDL ，用于定义数据库中的数据对象和它们的结构。

（2）　数据操纵：提供数据操纵语言　　DML ，用于操纵数据，实现对数据库的基本操作（查询、插入、删除和修改）　。

（3）　事务管理和运行管理：　统一管理数据、　控制对数据的并发访问，　保证数据的安全性、完整性，确保故障时数据库中数据不被破坏，并且能够恢复到一致状态。

（4）　数据存储和查询处理：确定数据的物理组织和存取方式，提供数据的持久存储和有

效访问；确定查询处理方法，优化查询处理过程。

（5） 数据库的建立和维护：提供实用程序，完成数据库数据批量装载、数据库转储、介质故障恢复、数据库的重组和性能监测等。

（6） 其他功能：包括 DBMS 与其它软件通信、异构数据库之间数据转换和互操作等。

1.4 使用数据库进行信息管理具有如下优点：

（1） 数据整体结构化：在数据库中，数据的组织面向整个机构、面向所有可能的应用，而不是某个具体部门或某个特定的应用。数据结构不仅描述现实世界的对象，而且描述对象之间的联系。

（2） 数据可以充分共享：数据库中的数据的面向整个机构组织使得它能够更好地被多个用户、多个应用程序共享。

（3） 数据独立性：数据独立性是指数据与应用程序相互独立，包括数据的物理独立性和数据的逻辑独立性。数据的结构用数据模型定义，无需程序定义和解释。

（4） 数据由 DBMS 同一管理和控制，使得系统能够为数据管理提供更多的支持。这些支持包括：提供事务支持、增强安全性、保证完整性、平衡相互冲突的请求和面对故障的弹性。

（5） 标准化：使用数据库进行信息管理有利于制定部门标准、行业标准、工业标准、国家标准和国际标准，促进数据库管理系统和数据库开发工具的研制、开发，推动数据管理应用的健康发展。

1.5 数据模型的三个基本要素是：

数据结构：描述数据库的对象和对象之间的联系，是对数据的静态描述。

数据操作：数据库中各种对象允许的操作和操作规则，使对系统的动态描述。

完整性约束： 一组完整性规则， 用以限定符合数据模型的数据库状态和状态的变化， 保证数据的正确、有效和相容。

对于关系数据库而言， 关系模型只有一种数据结构——关系。 现实世界中的对象和对象之间的联系都用关系表示。关系是元组的集合。从用户角度来看，关系是一张二维表。

在关系模型中， 定义数据操作的方法有两种： 关系代数和关系演算。 关系代数显式地定义了一些关系运算， 而关系演算的基础是一阶谓词逻辑， 它用逻辑公式表示查询结果必须满足的条件。

关系模型的完整性约束包括实体完整性、 参照完整性和用户定义的完整性。 其中实体完整性和参照完整性是通用完整性约束，由关系模型明确定义。

1.6 数据库系统的三级模式是指外模式、模式和内模式。外模式是特定数据库用户的数据视图，是与某一具体应用相关的数据局部逻辑结构的描述。 模式是数据库中全体数据的总体逻辑结构描述， 是所有用户的公共数据视图。 内模式是数据物理结构和存储方式的描述， 定义数据在数据库内部的表示方式。

数据库系统的三级模式提供了三个层次的数据抽象。 这样做的一个优点是可以隐蔽数据存储细节， 从而隐蔽系统内部的复杂性， 简化系统的用户界面。 另一个优点是可以带来数据的独立性。

1.7 所谓数据独立性是指数据独立于应用程序， 分数据的逻辑独立性和数据的物理独立性两种。

数据的逻辑独立性是指应用程序与数据库的逻辑结构之间的相互独立性。 当数据的逻辑

结构改变时，通过修改外模式 -模式映像，保持外模式不变，从而使得建立在外模式上的应用程序也可以不变。

数据的物理独立性是指应用程序与存储在磁盘上的数据库中数据之间的相互独立性。当数据的物理存储结构改变时，通过修改模式 -内模式映像，保持模式不变。由于外模式是定义在模式上的， 模式不变， 则外模式不需要改变， 从而使得建立在外模式上的应用程序也可以不变。

数据的逻辑独立性是指数据的逻辑结构改变不影响应用程序， 而数据的物理独立性是指数据的物理组织（存储结构）改变不影响应用程序。

1.8 DBA 的主要职责包括：

（1） 决定数据库中的信息内容和数据的逻辑结构。

（2） 决定数据库的存储结构和存取策略。

（3） 定义数据的安全性要求和完整性约束条件。

（4） 数据库系统的日常维护：周期性转储数据库、故障恢复、监督系统运行、优化系统性能、设置必要的审计。

（5） 重组和重构数据库。

第 2 章　实体-联系模型

部分习题参考答案

2.1 解释术语：

实体是客观存在并且可以相互区分的任何事物。

实体集是具有相同属性的实体的集合。

联系是多个实体之间的相互关联。

联系集是相同类型联系的集合。形式地说，设 $E_1, E_2, \cdots, E_n$ 是 n（$n \geq 2$）个实体集，它们不必互不相同。 联系集 R 是$\{(e_1, e_2, \cdots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \cdots, e_n \in E_n\}$的一个子集， 其中 $(e_1, e_2, \cdots, e_n) \in R$ 是一个联系，并称 e_i（$1 \leq i \leq n$）是该联系的参与者， n 是联系的度（元）。

简单属性是不能划分成更小的部分的属性。

复合属性是可以划分成更小部分的属性（即可以分成一些其他属性）。

单值属性是一个特定的实体在该属性上只能取单个值的属性。

多值属性是特定的实体在该属性上可以取多个值的属性。

基本属性是其值不能通过其他属性的值推导出来的属性。

派生属性又称计算属性，是其值可以从其他相关属性或实体计算得到的属性。

码是主码或候选码的简称。

主码是指数据库的设计者选中的，用来区分同一实体集中不同实体的候选码。

候选码：其真子集都不是超码的极小超码称为候选码。

超码：其值可以惟一确定实体集中每个实体的属性集称为该实体集的超码。

一对一联系：如果 E_1 中的每个实体最多与 E_2 中的一个实体相关联，并且 E_2 中的每个实体也最多与 E_1 中的一个实体相关联，则称 E_1 和 E_2 之间联系为一对一联系。

一对多联系：如果 E_1 中的每个实体都可以与 E_2 中任意多个实体相关联，而 E_2 中的每个实体最多与 E_1 中一个实体相关联，则称这种联系为 E_1 到 E_2 的一对多联系。

多对一联系：如果 E_1 中的每个实体最多与 E_2 中的一个实体相关联，而 E_2 中的每个实体都可以与 E_1 中任意多个实体相关联，则称这种联系为 E_1 到 E_2 的多对一联系。

多对多联系：如果 E_1 中的每个实体都可以与 E_2 中任意多个实体相关联，并且 E_2 中的每个实体也可以与 E_1 中任意多个实体相关联，则称 E_1 和 E_2 之间联系为多对多联系。

2.2 商品应当包含如下属性：

商品条码：标识商品。

商品名称：用户识别。

商品类别：用于商品分类。

生产商：

生产时间：

进价：

销售价：

存货数量：

2.3 所有部门形成一个实体集， 所有经理形成一个实体集。 假定每个部门最多只有一个经理，而每个人只能在一个部门出任经理， 那么部门与经理之间的联系“管理”是一对一联系。如果允许部门经理空缺， 但一个人是经理的话， 必须在一个部门任职， 那么经理对联系 “管理”的参与是全部参与，而部门是部分参与。

所有学生形成一个实体集，　所有院系形成一个实体集。　每个院系由多个学生，　而每个学生只能在一个院系。　因此，学生与院系之间的联系是多对一联系。　通常，一个学生总在一个院系中，而每个院系都有学生。因此，学生和院系对该联系都是全部参与。

商品是一个实体集，　订单是一个实体集。　每个订单可以包括多种商品，　而一种商品可以被多个订单订购。这样，商品与订单之间的联系“订购”是多对多联系。通常，每个订单至少包含一种商品，而每种商品都会被某个订单订购（否则就不再销售这种商品）　。这样，商品和订单对该联系的参与都是全部参与。

2.4 按以下要求各举一个实际例子：　（1）三个实体集两两之间都存在多对多联系（在你的例子中，三个实体集之间还存在有意义的联系吗？）　，（2）三个实体集之间存在多对多联系　（在你的例子中，其中两个实体集之间还存在有意义的联系吗？）　。

(1) 实体集教师、课程和学生两两之间的多对多联系

教师和课程之间的联系“讲授”　是多对多的：　一个教师教多门课程，一门课程由多位教师讲授

课程和学生之间的联系“选修”　是多对多的：　一门课程可以被多个学生选修，一个学生可以选多门课程。

学生和教师之间的联系“师生”　也是多对多的：一个学生可以有多位教师，　一个教师可以有多个学生。

教师、课程和学生三者之间也存在有意义的联系，　表明特定的学生选修了特定教师讲授的特定课程。

(2) 供应商、零件和项目之间的多对多联系“供应”

一个供应商向多个项目提供多种零件；一种零件由多个供应商提供，并用于多个项目；一个项目使用多个供应商提供的多种零件。

这三个实体集中两个实体集之间的有意义联系实际上“供应”的投影。

如果一个实体集的任何属性集都不足以形成该实体集的码，称该实体集为弱实体集。

职工家属（没有系统内专属的编号）（职工）

2.5 弱实体集的主码可以通过它与强实体集的联系推断。　如果将强实体集的主码属性添加到弱实体集，　那么这些属性将通过实体集和联系两种方式提供，　从而导致冗余。此外，实体集应当只包含描述该实体的属性，　强实体集的主码属性并不是描述弱实体集的，　因此添加它们使得模型不清晰。

2.6 如果一部分实体集通过　E-R 图的一条路径相连接，　则这些实体集是相关的，　或许是间接相关的。一个非连通的图意味一部分实体集与另一部分实体集是不相关的。　如果我们将　E-R 图划分成连通分支，则事实上我们就有了一些分离的数据库，每个对应一个连通分支。

如上所述，　一对实体集之间的路径指明这两个实体集之间的一种联系　（可能是间接的）　。如果图中存在环，则环中每对实体集至少可以通过两种不同的方式相关联。如果　E-R图是无环的，则每对实体集之间至多存在一条路径，因此每对实体集之间至多存在一种联系。

2.7 假定每辆汽车只属于一位客户。

涉及的实体集有：客户、汽车和事故。

需要建立如下联系：

拥有：客户与汽车之间的多对一联系

发生：客户、汽车和事故之间的多对多联系。

损坏估计最好作为联系“发生”　的属性，　因为损坏估计不仅与事故有关，而且与特定客户的特定汽车有关。

E-R图如图 2.1所示。

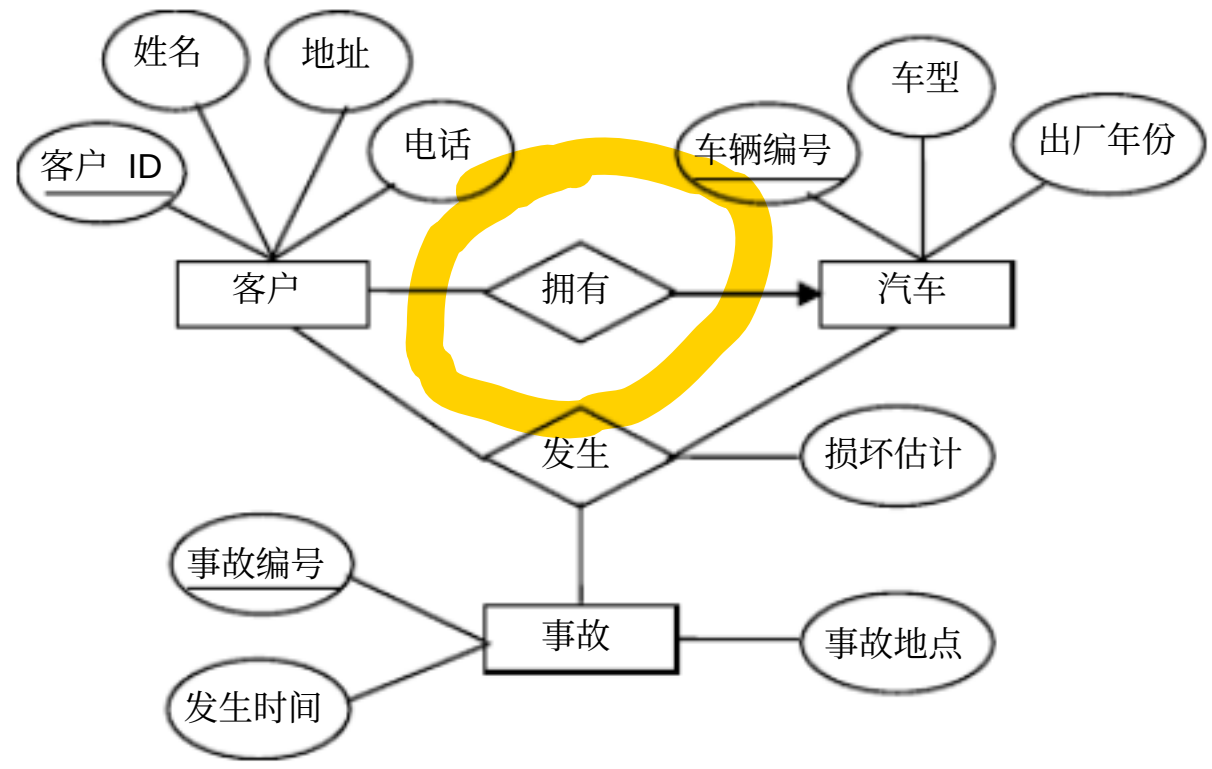


图 2.1 习题 2.7 的 E-R 图

2.8 假定一个客户可以有多个账户，但一个账户只属于一个客户。

涉及的实体集有：账户、支行、客户和贷款。题中已经清楚描述。

建立如下联系：

账户 -支行：账户与支行之间的多对一联系，其中账户全部参与。

贷款 -支行：贷款与支行之间的多对一联系，其中贷款全部参与。

借贷：客户与贷款之间的多对一联系，其中贷款全部参与。

账户与客户之间有两种联系：

存取款：客户与账户之间的多对多联系，包括属性存取金额和存取日期。

属于：客户与账户之间的一对多联系。

E-R 图如图 2.2 所示。

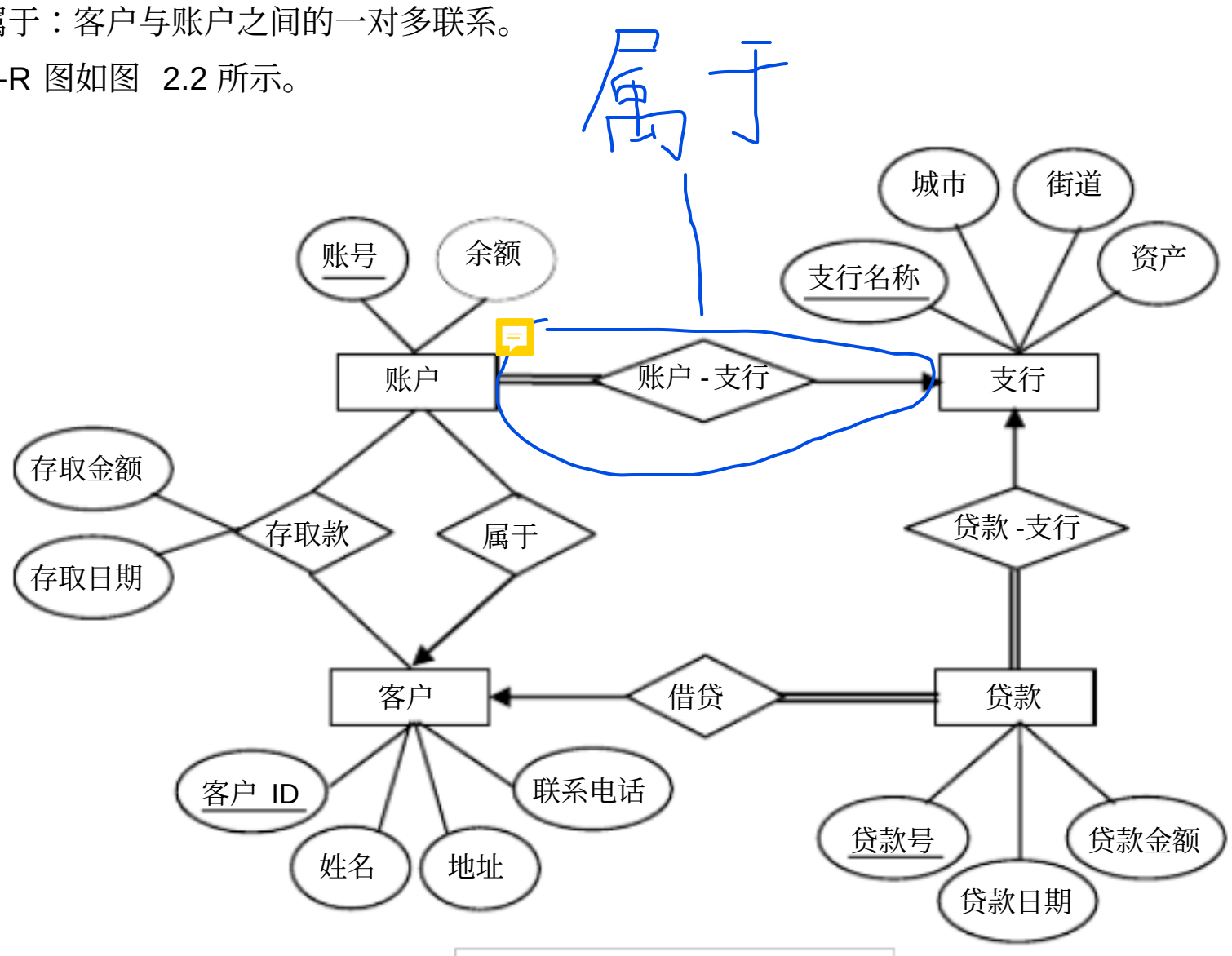


图 2.2 习题 2.8 的 E-R 图

2.9 方法一：使用弱实体

建立弱实体集“ 贷款偿还 ” ，包括属性：偿还编号（顺序号） 、偿还日期、偿还金额；

建立建立“ 贷款偿还 ”与其标识实体集“ 贷款 ”之间的标识性联系“ 还贷 ”

方法二：使用多值属性

将“ 贷款偿还 ”作为贷款的多值复合属性，它包括属性：偿还编号（顺序号） 、偿还日期、偿还金额

方法三：使用强实体集

建立强实体集“ 贷款偿还 ” ，包括属性：贷款编号、偿还编号（顺序号） 、偿还日期、偿还金额；

建立建立“ 贷款偿还 ”与 “ 贷款 ”之间的联系“ 还贷 ”

方法一最好，方法三最差，理由与职工 -家属的例子类似。

2.10 假定：

每位职工在同一时间段只从事一项工作。

每位职工不能同时在多个部门工作，也不能是多个部门的经理。

每位职工不能同时参加多个项目。

每位职工的办公室唯一。

一个项目只由一个部门承担。

一个办公室职能属于一个部门。

该问题涉及的实体集有：部门、职工、项目、办公室和职工的工作经历，其中工作经历存在依赖于职工，是弱实体集，其余是强实体集。

电话只有一个属性“ 电话号码 ” ，不把它视为实体集。一个办公室有多部电话，但假定每位职工只有一部电话。

需要建立如下联系：

管理：职工与部门之间一对一联系

工作：职工与部门之间多对一联系

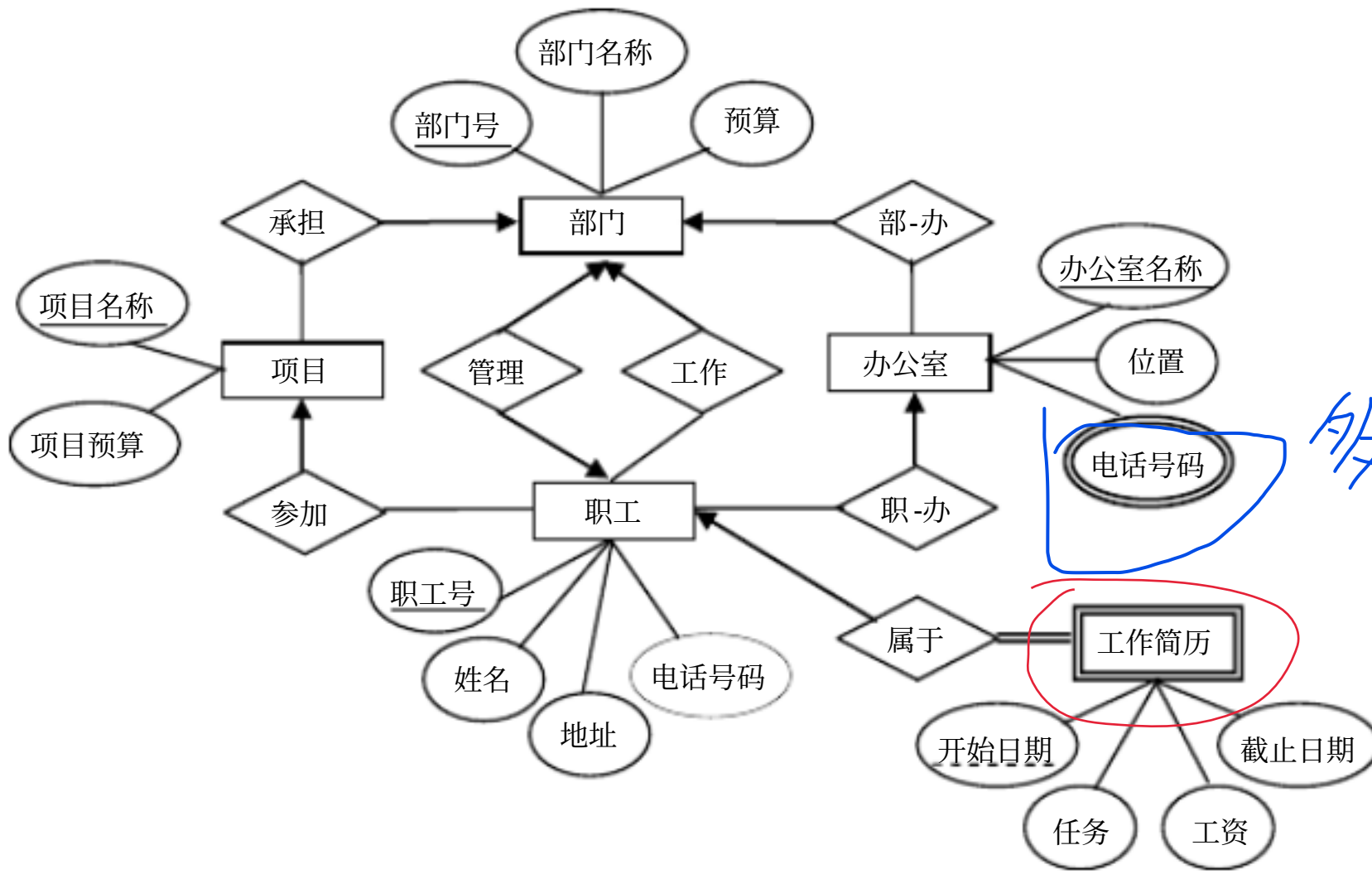


图 2.3 习题 2.10 的 E-R 图

承担：部门与项目之间一对多联系

部-办：部门与办公室之间一对多联系

参加：职工与项目之间多对一联系

职-办：职工与办公室之间多对一联系

属于：弱实体集工作简历与标识实体集职工之间的多对一联系

E-R 图如图 2.3 所示。

第 3 章　关系模型

习题参考答案

3.1 解释术语：

域是具有相同类型的值的集合。

笛卡尔积：给定　n 个域 $D_1, D_2, \cdots, D_n$（它们不必互不相同）上的　笛卡尔积 记作 $D_1 \times D_2 \times \cdots \times D_n$，定义为 $\{(d_1, d_2, \cdots, d_n) \mid d_1 \in D_1 \wedge d_2 \in D_2 \wedge \cdots \wedge d_n \in D_n\}$。

关系：域 $D_1, D_2, \cdots, D_n$ 上的关系 r 是笛卡尔积 $D_1 \times D_2 \times \cdots \times D_n$ 的任意子集。表记录

元组：笛卡尔积或关系的每个元素　$(d_1, d_2, \cdots, d_n)$称为一个 n-元组（简称元组）

属性：关系用一个二维表表示。列通常是命名的，称为属性。

关系的码：关系　R 的属性集　K 是它的码，如果　K 是 R 的超码，并且　K 的任何真子集都不是 R 的超码（即　K 是极小超码）。或 X 是关系　R 的超码，如果　t_1 和 t_2 是 R 的任意实例中的元组，并且　$t_1[X] = t_2[X]$，则 $t_1 = t_2$。

候选码：所有的码都称候选码。**可以唯一标识一个元组，且去掉任何一个属性之后就不能标识。候选码不唯一。**

主码：由多个码中选出的作为惟一识别关系元组的码

外码：如果　FK 是关系　R 的属性集，并且不是　R 的码，但是　FK 与关系　R 的主码　K 对应，则称　FK 是关系　R 的外码。

关系模式：　关系模式用关系模式名、　关系模式的诸属性和属性对应的域，　以及属性间的数据依赖集定义。通常简单地用关系模式名和属性列表表示　　$R(A_1, A_2, \cdots, A_n)$。

关系数据库模式：由若干域的定义和一组定义在这些域上的关系模式组成。

3.2　实体完整性：关系　R 的所有元组在主码上的值必须惟一，并且在主码的任何属性上都不能取空值。

参照完整性：如果属性集　FK 是关系　R 的外码，它参照关系　S 的主码　Ks，则 R 的任何元组在　FK 上的值或者等于　S 的某个元组在主码　Ks 上的值，或者为空值。

3.3 除了语义约束之外，关系数据库对关系的主要限制是：

(1) 在关系数据库中，我们只考虑有限关系（笛卡尔积的有限子集）　，因为无限关系既不能显式存储，也不能有效地显示。

(2) 关系的每个属性都必须是原子的，　即每个属性只能取原子值。　在关系数据库中，　原子值是数据访问的最小单位。属性的原子性要求是规范化关系的基本要求。

3.4 事实上，关系　R 的外码参照被参照关系　S（目标关系）反映　R 的某些元组每个都与　S 的某个特定元组之间存在联系。有些实际问题允许　　R 的某些元组与　S 的任何元组都没有联系，在这种情况下，允许　R 的这些元组在外码上取空值。例如，在关系　　Employees (Eno, Ename, Salary, Dno) 中，Dno 是外码。有些公司允许某些职工　（如公司总裁）　不属于任何特定的部门，这些职工的元组在　Dno 上可以取空值。

假定所有的关系模式都是　E-R 图转换得到的。

(1) 如果关系　R 是联系集转换的，　则 R 代表联系集，　其外码的值代表参与联系的特定实体集的一个特定实体。此时，　R 外码都不能取空值。

(2) 设关系　R 的外码　F_R 参照被参照关系　S。如果关系　R 是实体集　E_1 转换的，　则 E_1 必然通过某个联系　R 与　S 对应的实体集相关联。　当这种联系不要求是完全的时，　R 的某些元组可以不参照　S 的任何元组，此时外部码　F_R 的属性值可以为空；反之不能为空。

3.5 自然连接和等值连接的相同之处是二者都是根据属性值相等进行连接。 二者的不同之处是：自然连接在相同属性上进行相等比较， 并投影去掉重复属性； 等值连接并不要求一定在相同属性上进行相等比较，也不删除重复属性。

3.6 由强实体集得到的关系模式：

Employees (Eno, Ename, Salary)

Departments (Dno, Dptname)

Suppliers (Sno, Sname, Saddress)

Items (Ino, Iname, Stocks)

Orders (Ono, Data)

Customers (Cno, Cname, Caddress, Balance)

其中主码用下横线标示（下同） 。注意， Departments 的 Dptname 也是码，但我们选择 Dno 为主码。由弱实体集 Dependents 得到如下关系模式：

Dependents (Eno, Dname, ReltoEmp, Birthday)

将联系转换成关系模式时， 不再考虑弱实体集的存在依赖联系。 其余 6 个联系产生如下关系模式：

Manages (Dno, Eno)

Works_in (Eno, Dno)

Carries (Dno, Ino)

Supplies (Sno, Ino, Price)

Includes (Ono, Ino, Quantity)

Placed_by (Cno, Ono)

其中 Works_in 和 Manages 有相同的属性，但它们的实际意义不同。

下一步，我们合并具有相同码的关系模式。首先，考虑 Manages。它与 Employees 和 Departments 都包含相同的码。 Maneges 与 Employees 合并更容易回答“某职工的经理是谁”这类问题，而与 Departments 合并更容易回答“某部门的经理是谁”这类问题。考虑到后一类问题更经常出现，我们决定将 Manages 合并到 Departments，并将 Manages 中属性 Eno 改名为 Mrgno （表示经理的职工号） ，得到如下关系模式：

Departments (Dno, Dptname, Mrgno)

还有三对关系具有相同的码，它们是 Employees 和 Works_in， Items 和 Carries， Orders 和 Placed_by。它们都可以直接合并。 最后，我们得到图 3.11 所示 E-R 模型的一组关系模式：

Employees (Eno, Ename, Salary, Dno)

Department (Dno, Dptname, Mrgno)

Suppliers (Sno, Sname, Saddress)

Items (Ino, Iname, Stocks, Dno)

Orders (Ono, Data, Cno)

Customers (Cno, Cname, Caddress, Balance)

Dependents (Eno, Dname, ReltoEmp, Birthday)

Supplies (Sno, Ino, Price)

Includes (Ono, Ino, Quantity)

3.7 习题 3.6 的关系模式的模式图如图 3.1 所示。

1、由实体集得到关系模式
2、由联系集得到关系模式
3、合并具有相同码的关系模式

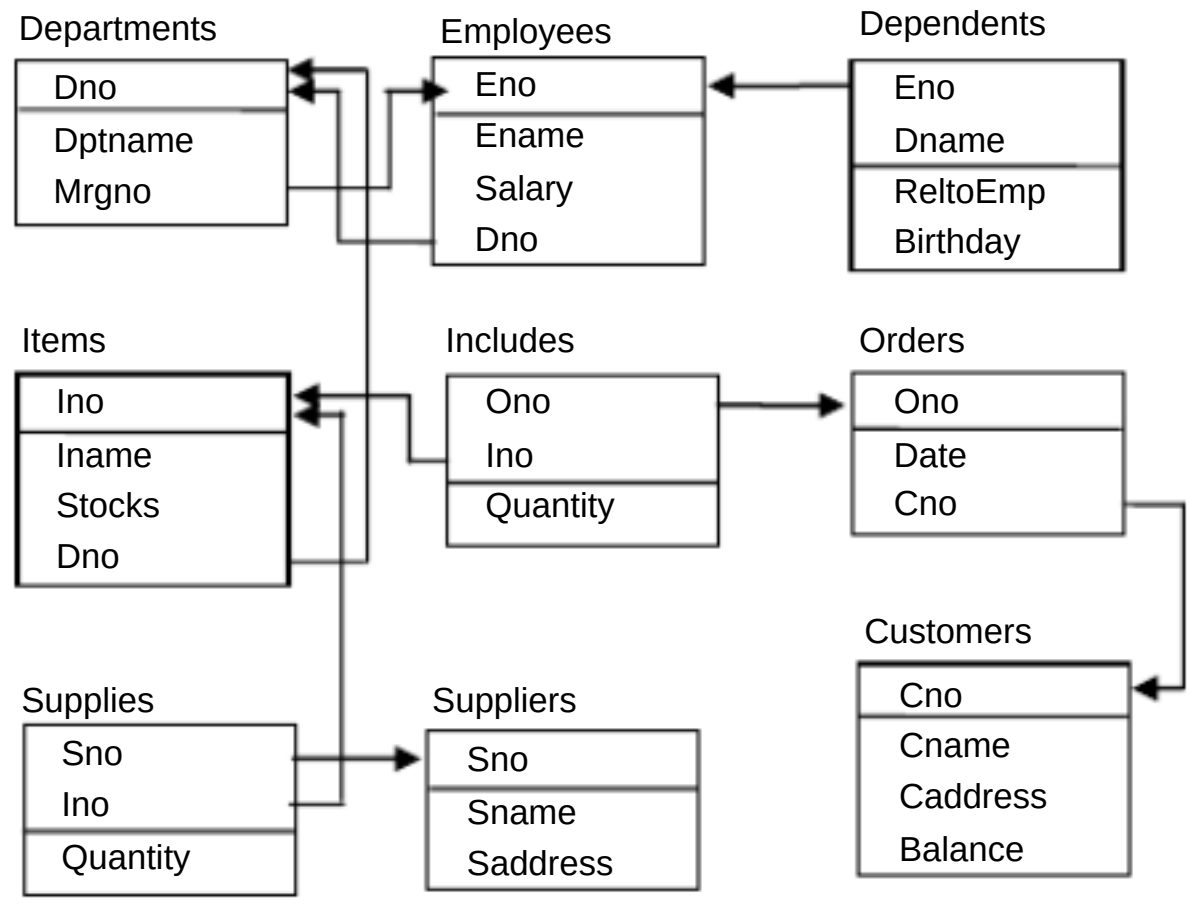


图 **3.1** 某公司数据库系统的数据库模式图

3.8 得到的一组关系模式如下：

客户（客户 ID，姓名，地址，电话）

汽车（车辆编号，车型，出厂年份，车主 ID）

事故（事故编号，发生时间，事故地点）

发生（驾照号，车辆编号，事故编号，损坏估计）

其中，客户与汽车之间的联系已经合并到关系模式“汽车”中，并将“客户 ID”改为“车主 ID”。

3.9 由实体集得到如下关系模式：

账户（账号，余额）

支行（支行名称，城市，街道，资产）

客户（客户 ID，姓名，地址，联系电话）

贷款（贷款号，贷款日期，贷款金额）

由联系得到如下关系模式：

账户 -支行（账号，支行名称）

贷款 -支行（贷款号，支行名称）

借贷（客户 ID，贷款号）

存取款（客户 ID，账号，存取金额，存取日期）

属于（账号，客户 ID）

合并具有相同码的关系模式：

账户、账户 -支行、属于具有相同码，合并成一个关系模式账户，并用开户行替换支行名称，开户人替换客户标识。合并后的关系模式如下：

账户（账号，余额，开户行，开户人）

贷款和贷款 -支行具有相同码，合并成一个关系模式贷款，并用贷款支行替换支行名称。合并后的关系模式如下：

贷款（贷款号，贷款日期，贷款金额，贷款支行）

最后得到的一组关系模式如下：

账户（账号，余额，开户行，开户人）
支行（支行名称，城市，街道，资产）
客户（客户标识，姓名，地址，联系电话）
贷款（贷款号，贷款日期，贷款金额，贷款支行）
借贷（客户　ID，贷款号）
存取款（客户　ID，账号，存取金额，存取日期）

3.10　电话号码是多值属性，需要创建一个关系模式。我们称该关系模式为电话，它的码为电话号码。该关系模式定义如下：

电话（电话号码，办公室名称）

由强实体集得到如下关系模式：

部门（部门号，部门名称，预算）
项目（项目名称，项目预算）
办公室（办公室名称，位置）
职工（职工号，姓名，地址，电话号码）

由弱实体集“工作简历”得到如下关系模式：

工作简历（职工号，开始时间，任务，工资，截止时间）

由联系集得到如下关系模式：

承担（项目名称，部门号）
部-办（办公室名称，部门号）
管理（职工号，部门号）
参加（职工号，项目名称）
工作（职工号，部门号）
职-办（职工号，办公室名称）

合并具有相同码的关系模式：

项目与承担具有相同码，合并为项目，并将部门号改为承担部门，得到：

项目（项目名称，项目预算，承担部门）

办公室与部-办具有相同码，合并为办公室，并将部门号改为所属部门，得到：

办公室（办公室名称，位置，所属部门）

管理有两个码，可以与职工合并（有利于回答“某职工的经理是谁”这类问题），也可以与部门合并（有利于回答“某部门的经理是谁”这类问题）。考虑“某部门的经理是谁”这类问题更常出现，决定于部门合并，并将职工号改为经理，得到：

部门（部门号，部门名称，预算，经理）

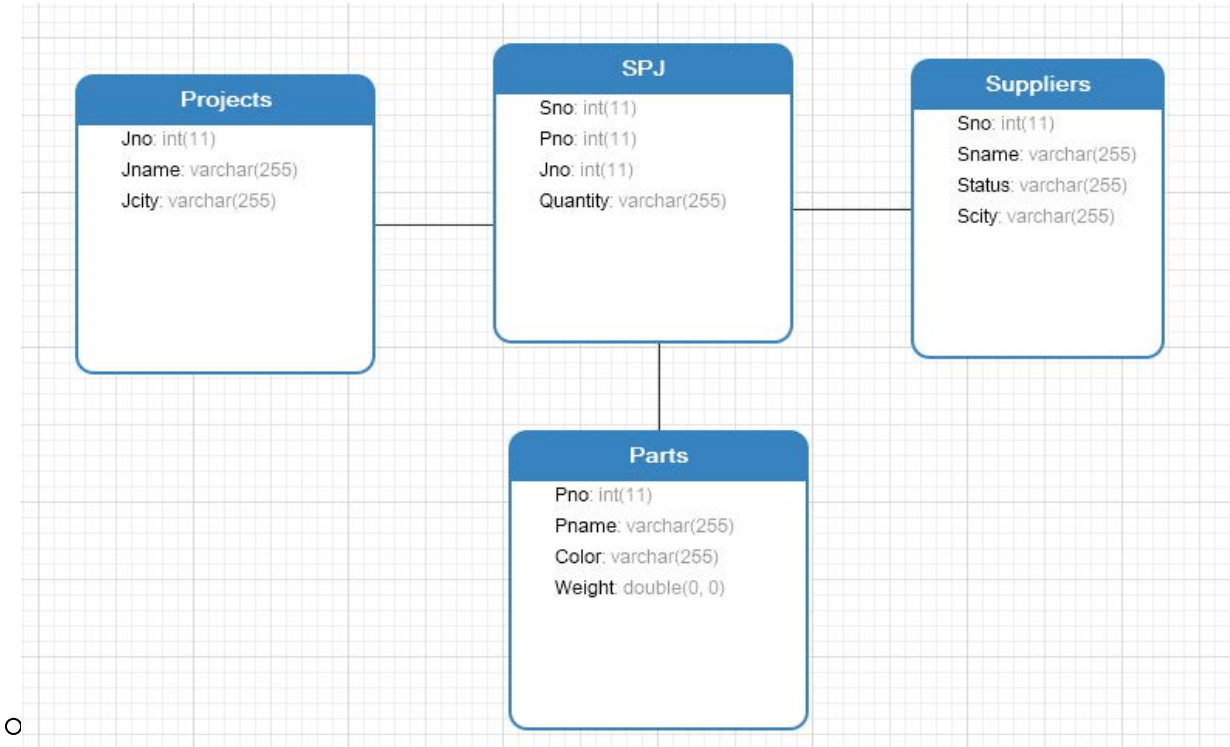
职工、参加、工作和职-办都具有相同码，合并为职工，得到：

职工（职工号，姓名，地址，电话号码，项目名称，部门号，办公室名称）

最后，我们得到如下关系模式：

部门（部门号，部门名称，预算，经理）
项目（项目名称，项目预算，承担部门）
办公室（办公室名称，位置，所属部门）
职工（职工号，姓名，地址，电话号码，项目名称，部门号，办公室名称）
工作简历（职工号，开始时间，任务，工资，截止时间）
电话（电话号码，办公室名称）

3.11



(1) 求上海的所有供应商的信息。

$\sigma_{Scity=?上海?}(Suppliers)$

(2) 求位于郑州的所有工程的信息。

$\sigma_{Jcity=?上海?}(Projects)$

(3) 求数量在 100~150 之间的供应。

$\sigma_{Quantity \geq 100 \wedge Quantity \leq 150}(SPJ)$

(4) 求为工程 J1 提供零件的供应商号。

$\pi_{Sno}(\sigma_{Jno=?J1?}(SPJ))$

(5) 求供应工程 J1 红色零件的供应商号。

$\pi_{Sno}(\sigma_{Jno=,J1? \wedge Color=,红色?}(SPJ \bowtie Parts))$

自然连接，对两个关系的共同属性做等值连接，再投影去掉重复属性

(6) 求至少提供一种红色零件的供应商名称。

$\pi_{Sname}(\sigma_{Color=,红色?}(Suppliers \bowtie SPJ \bowtie Parts))$

(7) 求不提供零件 P2 的供应商名称

$\pi_{Sname}(Suppliers) - \pi_{Sname}(\sigma_{Pno=,P2?}(Suppliers \bowtie SPJ))$

(8) 求没有使用天津供应商生产的红色零件的工程号。

使用了天津供应商生产的红色零件的工程号

$\pi_{Jno}(\sigma_{Scity=,天津?}(Suppliers) \bowtie SPJ \bowtie \sigma_{Color=,红色?}(Parts))$

该题的解

$\pi_{Jno}(Projects) - \pi_{Jno}(\sigma_{Scity=,天津?}(Suppliers) \bowtie SPJ \bowtie \sigma_{Color=,红色?}(Parts))$

(9) 求使用了本地供应商提供的零件的工程号和工程名称。

$\pi_{Jno,Jname}(\sigma_{Scity=Jcity}(Projects \bowtie SPJ \bowtie Suppliers))$

(10) 求未使用本地供应商提供的零件的工程号和工程名称。

$$\pi_{Jno,Jname}(Projects)$$

$$\pi_{Jno,Jname}(\sigma_{Scity=Jcity}(Projects \bowtie SPJ \bowtie Suppliers))$$

(11) 求至少用了供应商 S1 所供应的全部零件的工程号。

$$\pi_{Jno,Pno}(SPJ) \div \pi_{Pno}(\sigma_{Sno=,S1?}(SPJ))$$

RS÷S的意义就是：“在R和S的联系RS中，找出与S中所有的元组有关系的R元组”。

(12) 求提供所有零件的供应商名称。

$$\pi_{Sname}(\pi_{Sno,Pno}(SPJ) \div \pi_{Pno}(Parts)) \bowtie Suppliers)$$

~~3.12 对于供应商 - 工程 -零件数据库，用元组 /域关系演算表示~~习题 3.9 中的查询

(1) 求上海的所有供应商的信息。

$\{t \mid Suppliers(t) \wedge t[Scity] = ?上海?\}$

(2) 求位于郑州的所有工程的信息。

$\{t \mid Projects(t) \wedge t[Jcity] = ?郑州?\}$

(3) 求数量在 100~150 之间的供应。

$\{t \mid SPJ(t) \wedge t[Quantity] \geq 100 \wedge t[Quantity] \leq 150\}$

(4) 求为工程 J1 提供零件的供应商号。

$\{t^{(1)} \mid (\exists u)(SPJ(u) \wedge u[Jno] = ?J1? \wedge t[Sno] = u[Sno])\}$

(5) 求供应工程 J1 红色零件的供应商号。

$\{t^{(1)} \mid (\exists u)(\exists v)(SPJ(u) \wedge Parts(v) \wedge u[Pno] = v[Pno] \wedge$
$u[Jno] = ?J1? \wedge v[Color] = ?红色? \wedge t[Sno] = u[Sno])\}$

(6) 求至少提供一种红色零件的供应商名称。

$\{t^{(1)} \mid (\exists u)(\exists v)(\exists w)(Suppliers(u) \wedge SPJ(v) \wedge Parts(w) \wedge u[Sno] = v[Sno] \wedge$
$v[Pno] = w[Pno] \wedge v[Color] = ?红色? \wedge t[Sname] = u[Sname])\}$

(7) 求不提供零件 P2 的供应商名称

$\{t^{(1)} \mid (\exists u)(Suppliers(u) \wedge t[Sname] = u[Sname] \wedge$
$\neg(\exists v)(SPJ(v) \wedge u[Sno] = v[Sno] \wedge v[Pno] = ?P2?))\}$

(8) 求没有使用天津供应商生产的红色零件的工程号。

$\{t^{(1)} \mid (\exists u)(Projects(u) \wedge t[Jno] = u[Jno] \wedge$
$\neg((\exists v1)(\exists v2)(\exists v3)(Suppliers(v1) \wedge SPJ(v2) \wedge Parts(v3) \wedge$
$u[Jno] = v2[Jno] \wedge v1[Sno] = v2[SnoO] \wedge v2[Pno] = v3[Pno] \wedge$
$v1[Scity] =, 天津? \wedge v3[Color] =, 红色?))\}$

(9) 求使用了本地供应商提供的零件的工程号和工程名称。

$\{t^{(2)} \mid (\exists u)(\exists v)(\exists w)(Projects(u) \wedge SPJ(v) \wedge Suppliers(w) \wedge$
$u[Jno] = v[Jno] \wedge v[Sno] = w[Sno] \wedge u \quad Jcity] = w[Scity] \wedge$
$t[Jno] = u[Jno] \wedge t[Jname] = u[Jname])\}$

(10) 求未使用本地供应商提供的零件的工程号和工程名称。

$\{t^{(2)} \mid (\exists s)(Projects(s)$

$$\neg((\exists u)(\exists v)(\exists w)\ (Projects(u) \wedge SPJ(v) \wedge Suppliers(w) \wedge$$
$$u[Jno]=v[Jno] \wedge v[Sno]=w[Sno] \wedge u\ \ Jcity]=w[Scity] \wedge$$
$$t[Jno]=u[Jno] \wedge t[Jname]=u[Jname])$$

(11) 求至少用了供应商 S1 所供应的全部零件的工程号。

$$\{t^{(1)} \mid (\exists u)\ (Projects(u) \wedge t[Jno]=u[Jno] \wedge$$
$$\neg(\exists v)(SPJ(v) \wedge v[Sno]=,S1? \wedge$$
$$\neg(\exists w)(SPJ(w) \wedge u[Jno]=w[Jno] \wedge v[Sno]=w[Sno]))\}$$

(12) 求提供所有零件的供应商名称。

$$\{t^{(1)} \mid (\exists u)\ (Suppliers(u) \wedge t[Sname]=u[Sname] \wedge$$
$$\neg(\exists v)(Parts(v)$$
$$\neg(\exists w)(SPJ(w) \wedge u[Sno]=w[Sno] \wedge v[Pno]=w[Pno]))\}$$

~~3.13 用域关系演算完成例~~ 3.12 和例 3.13 中的查询

例 3.12

(1) 列出系编号为 MA （数学系）的所有学生的详细信息。

$$\{(x_1, x_2, x_3, x_4, x_5, ?MA?) \mid Students(x_1, x_2, x_3, x_4, x_5, ?MA?)\}$$

(2) 列出所有课程的课程号、课程名和学分。

$$\{(x_1, x_2, x_3) \mid (\exists y)(Courses(x_1, x_2, y, x_3))\}$$

(3) 列出年龄不超过 45 岁的所有副教授的姓名、性别和年龄。

$$\{(x_1, x_2, x_3) \mid (\exists y_1, y_2)\ (Teachers(y_1, x_1, x_2, x_3, ?副教授?, y_2))\}$$

(4) 列出选修了课程号为 CS201 的课程的所有学生的学号。

$$\{(x) \mid (\exists y)\ (SC(x, ?CS201?, y))\}$$

例 3.13

(1) 列出选修了课程号为 CS201 的课程的所有学生的学号和姓名。

$$\{(x_1, x_2) \mid (\exists x_3, x_4, x_5, x_6)(\exists y)(Students(x_1, x_2, x_3, x_4, x_5, x_6) \wedge SC(x_1, ?CS201?, y))\}$$

(2) 列出每个学生选修的每门课程的成绩，要求列出的学号、姓名、课程名和成绩。

$$\{(x_1, x_2, x_3, x_4) \mid (\exists (y_1, y_2, y_3, y_4)(\exists z)(\exists w_1, w_2, w_3, w_4)(Students(x_1, x_2, y_1, y_2, y_3, y_4) \wedge$$
$$SC(x_1, w_1, z) \wedge Courses(w_1, w_2, w_3, w_4)\}$$

(3) 求评估得分高于 90 分的教师所在院系名称、教师姓名、课程名和评估得分。

$$\{(x_1, x_2, x_3, x_4) \mid (\exists (y_1, y_2)(\exists z_1, z_2, z_3, z_4, z_5)(\exists w_1, w_2, w_3)\ (\exists v)(Departments\ (y_1, x_1, y_2) \wedge$$
$$Teachers(z_1, x_2, z_2, z_3, z_4, z_5) \wedge Courses(w_1, x_3, w_2, w_3) \wedge$$
$$Teaches(x_1, w_1, v))\}$$

14

(1) 求提供了零件的供应商的个数。

$$g_{count\text{-}distinct(Sno)}(SPJ)$$

(2) 求所有零件的平均重量。

$$g_{avg(Weight)}(Parts)$$

(3) 求供应商 S1 提供的每种零件的总数量。

$$P_{no}\gamma_{sum(Quantity)}(\sigma_{Sno=?S1?}(SPJ))$$

(4) 求供应商 S1 供应工程 J1 的每种零件的总重量。

$$P_{no}\gamma_{sum(Weight)}(\pi_{Pno,\ Weight}(\sigma_{Sno=?S1?\ \wedge\ Jno=?J1?}(Parts \bowtie SPJ)))$$

3.15 右外连接

$$R \bowtie\!\sqsubset S = (R \bowtie S) \cup \{(\underbrace{null, \cdots, null}_{\text{限制在 R 的不属于 S 的属性}})\} \times (S - \pi_S(R \bowtie S))$$

全外连接

$$R \bowtie\!\sqsubset S = (R \bowtie S) \cup (R - \pi_R(R \bowtie S)) \times \{(\underbrace{null, \cdots, null}_{\text{限制在 S 的不属于 R 的属性上}})\} \cup \{(\underbrace{null, \cdots, null}_{\text{限制在 R 的不属于 S 的属性上}})\} \times (S - \pi_S(R \bowtie S))$$

3.16

(1) 将 Cno、Cname、Caddress 和 Balance 分别为 C0199、李华、郑州市大学北路 46 号、6000 的客户信息插入 Customers。

$$Customers \leftarrow Customers \cup \{(C0199, 李华, 郑州市大学北路\ 46\ 号, 6000)\}$$

(2) 从 Dependents（家属）中删除删除 1979 年前出生的子女（ ReltoEmp= ‘子女’）。

$$Dependents \leftarrow Dependents - \sigma_{year(Birthday)<1979\ \wedge\ ReltoEmp=?子女?}(Dependents)$$

(3) 将销售部门（ Dname= ‘销售’）的职工工资（ Salary）提高 4%。

$$Employees \leftarrow \pi_{Eno, Ename, Salary*1.04, Dno}(\sigma_{Dname=?销售?}(Employees \bowtie Departments)) \cup (Employees - \pi_{Eno, Ename, Salary, Dno}(\sigma_{Dname=?销售?}(Employees \bowtie Departments)))$$

第 4 章　关系数据库标准语言　SQL

习题参考答案

4.1　SQL 的基本特点是：

(1) 集多种数据库语言于一体：　SQL 语言集数据定义、数据操纵和数据控制（　DCL ）功能于一体，语言简洁、风格统一，使用　SQL 就可以独立完成数据管理的核心操作。

(2) 高度非过程化：使用　SQL 语言时，用户只需要说明做什么，而不必指出怎么做。

(3) 面向集合的操作方式：　SQL 语言采用集合操作方式，其运算对象、运算结果均是元组的集合。

(4) 一种语法两种使用方式：　SQL 既可以作为一种自含式语言独立使用，也可以作为一种嵌入式语言与通用程序设计语言配合使用。在两种使用方式下，　SQL 语言的语法结构基本一致。

(5) 功能强大，语言简洁：　SQL 是一种完整地数据库语言，其功能涵盖数据定义、数据操纵、数据控制等数据管理的主要需求。但　SQL 语言相对比较简洁，其核心动词只有　9 个。另外，　SQL　语言的语法简单，与英语口语的风格类似，易学易用。

4.2　SQL 的基本功能包括：

(1) SQL 的数据定义语言（　DDL ）提供了模式定义、修改和删除，基本表定义、修改和删除、域定义修改和删除。

(2) SQL 的数据操纵语言　（DML ）提供了数据查询子语言。　SQL 的数据查询子语言是关系完备的，并且具有关系代数和关系演算的双重特征。

(3) SQL　DML　不仅包括数据查询，而且包括数据更新（数据插入、删除和修改）语句，允许用户更新数据库。

(4) SQL DDL　还允许用户定义视图，　并且　SQL DML　允许用户对视图进行查询和受限的更新操作。

(5) SQL DDL　允许用户定义各种完整性约束条件，　并在数据库访问时自动检查，　确保数据库操作不会破坏完整性约束条件。

(6) SQL DDL　还包括授权定义，用来定义用户对数据库对象（基本表、视图等）的访问权限，防止非法访问，确保数据库的安全性。

(7) SQL 还支持事务，提供了定义事务开始和结束的语句。

4.3　SQL 的数据定义语言　DDL 包括

模式定义、修改和删除；

基本表定义、修改和删除；

域定义、修改和删除；

视图的定义、修改和删除；

断言的定义、修改和删除；

授权的定义与回收。

4.4

```
CREATE TABLE    Suppliers
   (Sno          CHAR   (8) PRIMARY KEY    ,
    Sname        CHAR   (8) NOT NULL   ,
    Status       INT ,
    Scity        CHAR (10));
CREATE TABLE    Parts
```

```
    (Pno        CHAR  (8) PRIMARY KEY   ,
     Pname      CHAR (16) NOT NULL  ,
     Color      CHAR (4),
     Weight     NUMERIC  (7, 2));
CREATE TABLE   Projects
    (Jno        CHAR  (8) PRIMARY KEY   ,
     Jname      CHAR (20) NOT NULL  ,
     Jcity      CHAR (10));
CREATE TABLE   SPJ
        (Sno        CHAR  (8),
         Pno        CHAR (8),
         Jno        CHAR (8),
         Quantity   INT ,
         PRIMARY KEY    (Sno,Pno,Jno),
         FOREIGN KEY    Sno REFERENCES   Suppliers(Sno),
         FOREIGN KEY    Pno REFERENCES   Parts(Pno),
         FOREIGN KEY    Jno REFERENCES   Projects(Jno) );
```

4.5

3.8 题对应的表

```
        CREATE TABLE   Clients
          (Driver_id        CHAR (10) PRIMARY KEY   ,
          Cname             CHAR (8) NOT NULL  ,
          Address           CHAR (20));
        CREATE TABLE   Cars
          (Car_no           CHAR (10) PRIMARY KEY   ,
           Model            CHAR (20),
           Year             DATE );
        CREATE TABLE   Accidents
          (Report_no        CHAR (8) PRIMARY KEY   ,
           Accident_Date    DateTime ,
           Location         CHAR (40));
        CREATE TABLE   Participated
          (Driver_id        CHAR (10),
           Car_no           CHAR (10),
           Report_no        CHAR (8)
            Damage_amount   NUMERIC  (8, 2),
            PRIMARY KEY     (Driver_id, Car_no, Report_no),
            FOREIGN KEY     Driver_id  REFERENCES   Clients (Driver_id),
            FOREIGN KEY     Car_no REFERENCES   Cars(Car_no),
            FOREIGN KEY     Report_no  REFERENCES   Accidents (Report_no));
```

3.9 题对应的表

```
        CREATE TABLE   Accounts
          (Account_no       CHAR (10) PRIMARY KEY   ,
```

```
    Balance          NUMERIC  (11, 2),
    Branch_name      CHAR (20),
    Client_id        CHAR (18),
    FOREIGN KEY   Branch_name REFERENCES   Branchs(Branch_name),
    FOREIGN KEY   Client_id  REFERENCES   Clients(Client_id));
CREATE TABLE   Branches
  (Branch_name      CHAR (20)  PRIMARY KEY   ,
    City            CHAR (10),
    Street          CHAR (20)
    Asset           NUMERIC  (11, 2));
CREATE TABLE   Clients
  (Client_id        CHAR (18)  PRIMARY KEY   ,
    Cname           CHAR (8)  NOT NULL   ,
    Address         CHAR (40),
    Phone           CHAR (11) );
CREATE TABLE   Loans
  (Loan_id          CHAR (5)  PRIMARY KEY   ,
    Loan_Date       DATE ,
    Amount          NUMERIC  (11, 2),
    Branch_name     CHAR (20),
    FOREIGN KEY   Branch_name REFERENCES   Branchs(Branch_name));
CREATE TABLE   Borrow
  (Client_id        CHAR (18),
    Loan_id         CHAR (5),
    PRIMARY KEY   (Client_id , Loan_id),
    FOREIGN KEY   Client_id  REFERENCES   Clients(Client_id),
    FOREIGN KEY   Loan_id  REFERENCES   Loans(Loan_id));
CREATE TABLE   Deposit
  (Client_id        CHAR (18),
    Account_no      CHAR (10),
    Amount          NUMERIC  (11, 2),
    Deposit_Date    DATE ,
    PRIMARY KEY   (Client_id, Account_no),
    FOREIGN KEY   Client_id  REFERENCES   Clients(Client_id),
    FOREIGN KEY   Account_id  REFERENCES   Accounts(Account_id));
```

3.10 题对应的表

部门（部门号，部门名称，预算，经理）

项目（项目名称，项目预算，承担部门）

办公室（办公室名称，位置，所属部门）

职工（职工号，姓名，地址，电话号码，项目名称，部门号，办公室名称）

工作简历（职工号，开始时间，任务，工资，截止时间）

电话（电话号码，办公室名称）

```
CREATE TABLE   Departments
```

```
    (Dno            CHAR (5) PRIMARY KEY    ,
     Dptname        CHAR (20),
     Budget         NUMERIC  (11, 2),
     Mrgno          CHAR (8),
     FOREIGN KEY    Eno REFERENCES  Employees(Eno));
CREATE TABLE  Projects
    (Pname          CHAR (20) PRIMARY KEY    ,
     Pbudget        NUMERIC  (11, 2),
     Dno            CHAR (5),
     FOREIGN KEY    Dno REFERENCES  Departments(Dno));
CREATE TABLE  Offices
    (Oname          CHAR (20) PRIMARY KEY    ,
     Location       CHAR (20),
     Dno            CHAR (5),
     FOREIGN KEY    Dno REFERENCES  Departments(Dno));
CREATE TABLE  Employees
    (Eno            CHAR (8) PRIMARY KEY    ,
     Ename          CHAR (8),
     Address        CHAR (40),
     Phone          CHAR (11),
     Pname          CHAR (20),
     Dno            CHAR (5),
     Oname          CHAR (20),
     FOREIGN KEY    Pname REFERENCES  Projects(Pname),
     FOREIGN KEY    Dno REFERENCES  Departments(Dno),
     FOREIGN KEY    Oname REFERENCES  Offices(Oname));
CREATE TABLE  Resume
    (Eno            CHAR (8),
     Start_time     DATE ,
     Task           CHAR (20),
     Salary         NUMERIC  (8, 2),
     End_time       DATE ,
     PRIMARY KEY    (Eno,Start_time),
     FOREIGN KEY    Eno REFERENCES  Employees(Eno));
CREATE TABLE  Phones
    (Phone-no       CHAR (11) PRIMARY KEY    ,
     Oname          CHAR (20)
     FOREIGN KEY    Oname REFERENCES  Offices(Oname));
```

4.6

(1) 求上海的所有供应商的信息

```
SELECT  *
FROM  Suppliers
WHERE  Scity=, 上海 ?;
```

(2) 求位于郑州的所有工程的信息

```
SELECT *
FROM Projects
WHERE Jcity=, 郑州 ?;
```

(3) 求数量在 100~150 之间的供应

```
SELECT *
FROM SPJ
WHERE Quantity BETWEEN 100 AND 150;
```

(4) 求为工程 J1 提供零件的供应商号

```
SELECT Sno
FROM SPJ
WHERE Jno=,J1?;
```

(5) 求供应工程 J1 红色零件的供应商号

```
SELECT Sno
FROM SPJ, Parts
WHERE SPJ.Pno=Parts.PnoAND Jno=,J1?AND Color=, 红色 ?;
```

(6) 求至少提供一种红色零件的供应商名称

```
SELECT Sname
FROM SPJ, Parts, Suppliers
WHERE SPJ.Pno=Parts.PnoAND SPJ.Sno=Suppliers.Sno AND Color=, 红色 ?;
```

(7) 求不提供零件 P2 的供应商名称

找出不存在这种行为的 (提供了P2) 供应商

```
SELECT Sname
FROM Suppliers
WHERE NOT EXISTS
    (SELECT *
     FROM SPJ
     WHERE SPJ.Sno= Suppliers.Sno AND Pno=,P2?);
```

(8) 求没有使用天津供应商生产的红色零件的工程号

找出不存在符合条件的工程号

条件：没有使用天津供应商生产的红色零件

```
SELECT Jno
FROM Projects
WHERE NOT EXISTS
   (SELECT *
    FROM SPJ, Parts, Suppliers
    WHERE SPJ.Sno= Suppliers.Sno AND
          Parts.Pno=SPJ.Pno AND
          Color=, 红色 ?AND Scity=, 天津 ?);
```

(9) 求使用了本地供应商提供的零件的工程号和工程名称

```
SELECT Jno, Jname
FROM Projects,Suppliers,SPJ
WHERE SPJ.Sno= Suppliers.Sno AND
      Projects.Jno=SPJ.Jno AND
      Projects.Jcity= Suppliers.Scity;
```

或

```
SELECT Jno,Jname
```

```
FROM   Projects
WHERE EXISTS
    (SELECT   *
      FROM   SPJ, Suppliers
      WHERE    SPJ.Sno= Suppliers.Sno AND
                  Projects.Jno=SPJ.Jno AND
                  Projects.Jcity= Suppliers.Scity);
```

查询存在这样条件（使用了本地供应商提供的零件）的工程

(10) 求未使用本地供应商提供的零件的工程号和工程名称

```
SELECT   Jno,Jname
FROM   Projects
WHERE NOT EXISTS
    (SELECT   *
      FROM   SPJ, Suppliers
      WHERE    SPJ.Sno= Suppliers.Sno AND
                  Projects.Jno=SPJ.Jno AND
                  Projects.Jcity= Suppliers.Scity);
```

查询 不 存在这样条件（使用了本地供应商提供的零件）的工程

查询不存在这样条件（使用了供应商s1提供的全部零件）的工程

(11) 求至少用了供应商 S1 所供应的全部零件的工程号

```
SELECT   Jno
FROM   Projects
WHERE NOT EXISTS
    (SELECT *
      FROM   SPJ SPJ1
      WHERE   SPJ1.Sno = ,S1?AND
          NOT EXISTS
              (SELECT   *
                FROM   SPJ SPJ2
                WHERE    SPJ2.Jno= Projects.Jno AND
                            SPJ2.Pno= SPJ1.Pno));
```

例题同类型

(12) 求提供所有零件的供应商名称

```
SELECT   Sname
FROM   Suppliers
WHERE NOT EXISTS
    (SELECT *
      FROM   SPJ SPJ1
      WHERE NOT EXISTS
          (SELECT   *
            FROM   SPJ SPJ2
            WHERE    SPJ2.Sno= Suppliers.Sno AND
                        SPJ2.Pno= SPJ1.Pno));
```

例题同类型

4.7

(1) 求提供了零件的供应商的个数

```
SELECT distinct   Sno
FROM   SPJ;
```

(2) 求所有零件的平均重量

```
SELECT avg (weight)
FROM Parts;
```

(3) 求供应商 S1 供应工程 J1 的每种零件的总重量

```
SELECT sum (weight)
FROM Parts,SPJ
WHERE SPJ.Pno=Parts.PnoAND Sno=,S1? AND Jno=,J1?
GROUP BY Pno;
```

(4) 求供应商 S1 提供的每种零件的总数量

```
SELECT Pno, sum(weight)
FROM SPJ
WHERE Sno=,S1?
GROUP BY Pno;
```

4.8

(1) 将 Cno、Cname、Caddress 和 Balance 分别为 C0199、李华、 郑州市大学北路 46 号、6000 的客户信息插入 Customers

```
INSERT INTO Customers
VALUES (,C 0199?,?李华 ?,?郑州市大学北路 46 号?,6000)
```

(2) 从 Dependents（家属）中删除删除 1979 年前出生的子女（ ReltoEmp=, 子女 ?）

```
DELETE FROM Dependents
WHERE ReltoEmp=, 子女 ? AND year(Birthday)<1979;
```

(3) 将销售部门（ Dname=, 销售 ?）的职工工资（ Salary）提高 4%

```
UPDATE Employees
SET Salary=Salary*1.04
WHERE Dno IN
    (SELECT Dno
     FROM Departments
     WHERE Dname=, 销售 ?);
```

4.9

(1) 找出 2006 年其车辆出过事故的总人数

```
SELECT Count(Distinct Driver_id)
FROM Participated, Accidents
WHERE Participated.Report_no= Accidents.Report_no
    AND Date=2006;
```

(2) 找出与王明的车有关的事故数量

```
SELECT Count(Report_no)
FROM Participated, Clients
WHERE Participated.Driver_id= Clients. Driver_id AND Cname=, 王明 ?;
```

(3) 删除李莉的 Mazada 车

```
DELETE FROM Cars
WHERE Model =,Mazada? AND
    Car_no IN
```

(**SELECT** Car_no

FROM Owns, Clients

WHERE Owns.Driver_id=Clients. Driver_id **AND** Cname=, 李莉 ?);

(4) 对于一个新事故，需要将相应信息插入表 Accidents 和 Participated 中。

4.10 所谓基本表是其关系元组存储在数据库中的表。视图是一种用查询定义的命名的导出表，其关系不存储在数据库中，而是在查询时执行定义视图的查询，由基本表导出。

基本表与视图之间的主要区别是前者对应的关系存储在数据库中，而后者对应的关系不在数据库中存储（物化视图除外）。从使用角度而言，对于查询，二者没有区别；而对于更新，只有可更新视图才可以更新。

二者之间的联系体现在：所有视图都是直接或间接由基本表定义的。

4.11 使用视图的优点有：

(1) 使用视图可以使一些查询表达更加简洁。

(2) 视图提供了一定程度的逻辑独立性。

(3) 视图与授权配合使用，可以在某种程度上对数据库起到保护作用。

(4) 视图使得用户能够以不同角度看待相同的数据。

4.12 并非所有的视图都是可以更新的，因为对视图的更新要转化为对相应基本表的更新，而对于某些视图，有些更新不能唯一地转换成对基本表的更新。

所有更新都能唯一转换成对基本表的更新的视图是可更新的，否则是不可更新的。目前，二者之间的明确边界尚不清楚。我们知道“行列子集”视图是可更新的，而使用聚集函数定义的视图不是可更新的。

4.13 在需要使用数据库管理数据时，使用 SQL 语言建立数据库，并对数据进行操作比使用其他通用程序设计语言更便捷，常常也更有效。然而，由于（1）SQL 能够表达常见的查询，但是不能表达所有查询。（2）一些非数据库操作，如打印报表、将查询结果送到图形用户界面中，都不能用 SQL 语句实现。一个应用程序通常包括多个组件，查询、更新只是一个组件，而许多其他组件都需要用通用编程语言实现。这时，我们需要使用嵌入式 SQL，而不是单独使用 SQL 或某种通用程序设计语言。

嵌入式sql：例如jdbc，sql嵌入其他编程语言中使用。

第 5 章　完整性和安全性

习题参考答案

5.1 数据库的完整性是指数据库中的数据的正确性、一致性和相容性。

所谓计算机系统安全性是指为计算机系统建立和采取各种安全保护措施，以保护计算机系统中的硬件、软件及数据，防止因偶然或恶意的原因使系统遭到破坏，数据遭到更改或泄露。所谓数据库的安全性是指保护数据库，防止因用户非法使用数据库造成数据泄露、更改或破坏。

数据的完整性和安全性是一个问题的两个方面，都是为了保护数据库中的数据。前者旨在保护数据库中的数据，防止合法用户对数据库进行修改时破坏数据的一致性；而后者旨在保护数据库，防止未经授权的访问和恶意破坏和修改。

5.2 为了维护数据库的完整性，完整性控制应当作为 DBMS 核心机制，必须提供：

(1) 说明和定义完整性约束条件的方法：DBMS 的 DDL 允许用户根据实际问题的语义说明和定义各种完整性约束条件。

(2) 完整性检查机制：DBMS 在数据更新可能破坏完整性时自动进行完整性检查。检查可以在更新操作执行时立即执行，也可以在事务提交时进行。

(3) 违约处理：当数据更新违反完整性约束时，DBMS 应当采取相应的措施，确保数据的完整性。

5.3 当违反参照完整性时，除了简单的拒绝之外，还可以：

(1) 级联：进行更新，并且对更新导致违反参照完整性的参照关系元组进行相应更新。

通常，这种情况发生在被参照关系的主码改变时。例如，部门调整时将部分或所有部门重新编号，可以级联地更新其他包含部门号的关系元组。

(2) 置空值：进行更新，并且对更新导致违反参照完整性的参照关系元组的外码置空值。这种处理方法仅当外码允许取空值时才能使用。

通常，这发生在被参照关系的元组被删除，并且允许参照关系的元组不参照时。例如，通常允许某个职工不在任何具体部门，这种职工的部门号（外码）可以取空值。当一个部门被撤销时，可以删除该部门在 Departments 中的对应元组，并且同时将原来在该部门的职工的 EMPS 元组置空值。

(3) 置缺省值：进行更新，并且对更新导致违反参照完整性的参照关系元组的外码置缺省值；其中缺省值必须是被参照关系某元组主码上的值。

通常，这发生在被参照关系的元组被删除，并且参照关系的元组在外码上有合理缺省值时。例如，许多单位都有人才交流中心这样的部门，对职工进行再培训和重新安排。当一个部门被撤销时，该部门原有职工可以到人才交流中心。这时，可以将这些职工的部门号设置成缺省值——人才交流中心的部门号。

5.4

```
CREATE TABLE  Suppliers
  (Sno        CHAR  (8) PRIMARY KEY  ,
   Sname      CHAR  (8) NOT NULL  ,
   Status     INT,
   Scity      CHAR(10));
```

```
CREATE TABLE    Parts
   (Pno          CHAR   (8) PRIMARY KEY    ,
    Pname        CHAR(16) NOT NULL,
    Color        CHAR(4),
    Weight       NUMERIC   (7, 2));
CREATE TABLE    Projects
   (Jno          CHAR   (8) PRIMARY KEY    ,
    Jname        CHAR(20) NOT NULL,
    Jcity        CHAR(10));
```

假定只能删除不向任何工程提供任何零件的供应商，只能删除没有任何供应商提供的零件，但是允许工程下马不依赖于是否有供应商提供零件。

```
CREATE TABLE  SPJ
   (Sno          CHAR   (8),
    Pno          CHAR(8),
    Jno          CHAR(8),
    Quantity     INT,
   PRIMARY KEY   (Sno,Pno,Jno),
   FOREIGN KEY   Sno REFERENCES   Suppliers(Sno)
    ON UPDATE CASCADE   );
   FOREIGN KEY    Pno REFERENCES    Parts(Pno)
    ON UPDATE CASCADE    );
   FOREIGN KEY    Jno REFERENCES    Projects(Jno)
    ON UPDATE CASCADE ON DELETE CASCADE        );
```

外键定义在中间表里

级联更新

保证参照完整性
CASCADE 级联
SET NULL 置空
SET DEFAULT 置缺省值
NO ACTION 拒绝

5.5 如果被删除的 Employee 元组不是经理，则没有任何 Employee 参照它。此时，仅仅删除该元组。然而，如果被删除的 Employee 元组是经理，则导致该经理的直接和间接下属的 Employee 元组被级联地删除。即初始的删除将触发该经理的直接下属的 Employee 元组被删除。如果这些被删除的元组有经理的话，将触发第二层下属的 Employee 元组被删除。如此下去，直到初始被删除的经理的所有的直接和间接下属的 Employee 元组都被删除。

5.6 断言如下：

```
CREATE   ASSERTION   sumConstraint  CHECK
  (SELECT   sum(amount)
     FROM   loan
     WHERE   loan.branch-name=  ”大学路支行   ”)
>=  (SELECT   sum(amount)
     FROM   account
     WHERE   loan.branch- name= ”大学路支行   ”))
```

更一般地，下面的断言保证每个支行的贷款总额不超过该支行的资产：

```
CREATE   ASSERTION   sum-constraint  CHECK
  (NOT EXISTS
     (SELECT   *
      FROM   Branch
      WHERE
```

```
        (SELECT   sum(amount)
          FROM   loan
          WHERE   loan.branch-name=branch.branch-name   )
    >=  (SELECT   sum(amount)
          FROM   account
          WHERE   loan.branch-name=branch.branch-name)))
```

5.7 假设 (Sno,Cname)是 IEGrades 的码

(1)

```
CREATE TRIGGER      InsertionTriggerONSC
AFTER INSERT ON     SC
REFERENCING NEW ROW AS       nrow
FOR EACH ROW
WHEN   nrow.Sno IN
          (SELECT   Sno
            FROM   Students
            WHERE   Dno= ,IE ?)
BEGIN ATOMIC
    INSERT INTO     IEGrades
    SELECT   S.Sno, Sname, Cname, Grade
     FROM   Students S, SC, Courses C
     WHERE   S.Sno=nrow.Sno AND   C.Cno=nrow.Cno  AND
               S.Sno = SC.Sno AND  C.Cno = SC.Cno;
    END
```

(2)

```
    CREATE TRIGGER      UpdateTriggerONSC
    AFTER UPDATE OF     Grade ON  SC
    REFERENCING NEW ROW AS       nrow
    FOR EACH ROW
    WHEN   nrow.Sno IN
              (SELECT   Sno
                FROM   Students
                WHERE   Dno= ,IE ?)
    BEGIN ATOMIC
        UPDATE   IEGrades
        SET  Grade=nrow.Grade
            WHERE   S.Sno=nrow.Sno AND  Cname=
                (SELECT   Cname
                  FROM   SC, Courses C
                  WHERE   C.Cno=nrow.Cno  AND  SC.Sno=nrow.Sno AND
                     C.Cno = SC.Cno;
    END
```

(3)

```
    CREATE TRIGGER      DeleteTriggerONSC
```

```
AFTER DELETE ON      SC
REFERENCING OLD ROW AS       orow
FOR EACH ROW
WHEN   orow.Sno IN
              (SELECT   Sno
                FROM   Students
                WHERE   Dno= ,IE ?)
BEGIN ATOMIC
     DELETE   FROM   IEGrades
          WHERE   S.Sno=orow.Sno AND  Cname=
              (SELECT   Cname
                FROM   SC, Courses C
                WHERE   C.Cno=orow.Cno  AND  SC.Sno=orow.Sno AND
                   C.Cno = SC.Cno
     END
```

5.8 保证银行数据的安全性的措施包括：

(1) 机房的安全性：防止入侵者强行闯入或非法潜入（物理层次）　　。

(2) 数据的远程备份：防止火灾等灾害性事件破坏数据（物理层次）　　。

(3) 安全的网络系统：专用网络和安全的网络软件（网络层次）　　。

(4) 安全的操作系统（操作系统层面）和安全的数据库管理系统。

(5) 严格授权管理：银行要制定严格的权限规定和授权审批制度（行政制度层面）　　，DBA 只能对允许的权限实施授权（数据库系统层面）　　。

(6) 审计跟踪：对数据库数据的所有修改都必须记录到审计日志中（数据库系统层面）　　，并且只有经过行政授权的高级管理人员才能查阅审计记录（行政制度层面）　　。

5.9 SQL 中涉及自主存取控制的语句包括：

(1) 创建用户：在创建用户的同时对创建数据库模式和创建表授权

CREATE USER < 用户名 > WITH DBA | RESOURSE;

(2) 授权与授权回收 ： 是自主存取控制的主要手段 ． 包括以下语句

授权语句

GRANT < 权限 >, …, <权限 > ON < 对象类型 > <对象名 > TO < 用户 >, …, <用户 >

[WITH GRANT OPTION];

授权回收语句

REVOKE < 权限 >, …, <权限 > ON < 对象类型 > <对象名 >, …, < 对象类型 > < 对象名 >

FROM < 用户 >, …,<用户 > [CASCADE | RESTRICT];

(3) 关于角色的语句。这些语句祥见下题。

5.10 当一组用户必须具有相同的存取权限时，　　使用角色定义存取权限，　并对用户授权是方便的，可以简化授权，并有利于授权管理。

关于角色的语句：联合使用实现灵活的授权。这些语句是

(1) 创建角色：

CREATE ROLE < 角色名 >;

(2) 给角色授权：

GRANT < 权限 >,⋯,<权限 > ON < 对象类型 > <对象名 > TO < 角色 >,⋯,<角色 >;

(3) 将角色授予其他角色或用户：

GRANT < 角色 >, ⋯,<角色 > TO < 角色 >|<用户 >, ⋯;<角色 >|<用户 > [WITH ADMIN OPTION];

(4) 回收角色权限：

REVOKE < 权限 >, ⋯,<权限 > ON < 对象类型 ><对象名 > FROM < 角色 >, ⋯;<角色 >;

5.11 在 MAC 机制中，主体是系统中的活动实体。主体可以是 DBMS 管理的实际用户、代表用户的各进程。客体是系统中的被动实体，是受主体操纵的，如文件、基本表、索引、视图等。

对于主体和客体，DBMS 为它们的每个实例（值）指派一个敏感度标记（Label）。敏感度标记分成若干级别，如绝密（Top Secret）、机密（Secret）、秘密（Confidential）、公开（Public）。

主体的敏感度标记称为许可证级别（Clearance Level）。客体的敏感度标记称为密级（Classification Level）

5.12

(1) **GRANT ALL PRIVILIGES ON** Employee, Department **TO** Wanglan **WITH GRANT OPTION** ;

(2) **GRANT SELECT ON** Department **TO PUBLIC** ;

(3) 我们需要在 Employee 上创建一个视图 EMPS：

```
CREATE VIEW EMPS AS
    SELECT Ename, Title, Dno
    FROM Employee;
```

然后，将 EMPS 上的查询权限授予所有用户：

```
GRANT SELECT ON EMPS TO PUBLIC ;
```

(4) **GRANT UPDATE** (Salary) **ON TABLE** Employee **TO** LiYong;

(5) 我们需要定义一个视图，包含每个部门的最低、最高和平均工资：

```
CREATE VIEW DeptSalary AS
SELECT Dname, MIN (Salary) MinSalary , MAX (Salary) MaxSalary ,
        AVG (Salary) averageSalary
FROM Employee E, Department D
WHERE E.Dno=D.Dno
GROUP BY Dname
```

然后，将 DeptSalary 上的查询权限授予 ShangHua：

```
GRANT SELECT ON DeptSalary TO ShangHua；
```

(6) 首先创建一个角色 Secretary：

```
CREATE ROLE Secretary；
```

然后，对角色 Secretary 授权：

```
GRANT ALL PRIVILIGES ON Department TO Secretary;
GRANT SELECT, DELETE ON Employee TO Secretary;
GRANT UPDATE (Eno, Ename, Birthday, Title, Dno)
    ON TABLE Employee TO Secretary;
```

(7) **GRANT** Secretary **TO** Lihua;

5.13

(1) **REVOKE ALL PRIVILIGES ON** Employee, Department
FROM Wanglan **CASCADE;**
REVOKE SELECT ON Department **FROM PUBLIC** ;
REVOKE SELECT ON EMPVIEW **FROM PUBLIC** ;
REVOKE UPDATE (Salary) **ON TABLE** Employee **FROM** LiYong;

(2) **REVOKE UPDATE** (Title)
ON TABLE Employee **FROM** Secretary;

第 6 章　关系数据库的设计理论

习题参考答案

平凡的函数依赖：x—>y且y∈x，即整体确定局部。
此处未列举平凡的函数依赖。

6.2 函数依赖

Driver-id → Name, Driver-id → Address, Driver-id → Phone-no

x—>y，x确定y，y依赖于x

Car-no → Model , Car-no → Year

Report-no → Date, Report-no → Location , Report-no → Damage

6.3 函数依赖

Cno → Balance, Cno → CreditLimit

Ono → Date, Ono → Address

Ono,Ino → QTY

Ino → Description （假设同一种货物可以由不同制造商制造，但具有相同描述）

Ino,Plant → QTYOH

6.4

(1) S→D 表示股票红利由股票唯一确定

I→B 表示每个投资人至多有一个经纪人

IS→Q 表示投资人和股票唯一确定他　/她对该股票的拥有量

B→O 表示每个经纪人都有唯一的办公室

(2) R(B, O, I, S, Q, D)的一个码为 IS

6.5 初始： Result ← BD

求闭包

Result ← BDEG （使用 D→EG）

Result ← BCDEG （使用 BE→C）

使用 CG→BD 不改变 Result

Result ← ABCDEG （使用 CE→AG）

Result 已包含所有属性，不可能再增大。因此　$(BD)^+$ = ABCDEG

6.6 初始： Result=AC

Result ← ABC（使用 A→B）

Result ← ABCDE （使用 BC→DE）

再无可用的函数依赖。因此　$(AC)^+$ = ABCDE。

F 不逻辑蕴涵函数依赖　ACF→DG，因为 DG 不在 AC 的闭包中。

6.7 F_1和 F_2等价。这是因为

A的每个函数依赖都逻辑蕴含于B。（相反也成立）
（A的每个函数依赖的决定因素关于B的闭包 包含此函数依赖的所有元素。）

显然 A→B 在 F_2^+中。而 AB 关于 F_2 的闭包为 ABC，因此 AB→C 在 F_2^+中。而 D 关于 F_2 的闭包为 ABCDE，因此 D→AC 和 D→E 都在 F_2^+中。这就证明了 $F_1 \subseteq F_2^+$。

反过来， A 关于 F_1 的闭包为 ABC，因此 A→BC 在 F_1 中。 D 关于 F_1 的闭包为 ABCDE，因此 D→AE 在 F_1 中。这表明 $F_2 \subseteq F_1^+$。

综上， $F_2^+ \equiv F_1^+$。

两函数依赖集等价 等价于 两者的闭包相等
但函数依赖集的闭包数量非常的

6.8 事实上， F 还有 3 个极小依赖集。对 F 右端极小化，得到：

求函数依赖集的极小覆盖

$A\rightarrow B, A\rightarrow C, B\rightarrow A, B\rightarrow C, C\rightarrow A, C\rightarrow B$

依次删除 $A\rightarrow C$ 和 $C\rightarrow A$ 得到 F 的一个极小覆盖 $F_{m2}=\{A\rightarrow B, B\rightarrow A, B\rightarrow C, C\rightarrow B\}$

依次删除 $B\rightarrow C$ 和 $C\rightarrow B$ 得到 F 的一个极小覆盖 $F_{m3}=\{A\rightarrow B, A\rightarrow C, B\rightarrow A, C\rightarrow A\}$

依次删除 $A\rightarrow B$ 和 $B\rightarrow A$ 得到 F 的一个极小覆盖 $F_{m4}=\{A\rightarrow C, B\rightarrow C, C\rightarrow A, C\rightarrow B\}$

6.9 F 不是极小函数依赖集。

显然， F 右部是极小的。

考虑 $ABD\rightarrow E$：$(AB)^+=ABGH$，不包含 E；$(AD)^+=AD$，不包含 E；$(BD)^+=BDK$，不包含 E；因此， $ABD\rightarrow E$ 的左部是极小的。

考虑 $AB\rightarrow G$：$A^+=A$，不包含 G；$B^+=BK$，不包含 G；因此， $AB\rightarrow G$ 的左部是极小的。

考虑 $CJ\rightarrow I$：$C^+=CIJ$，包含 I；因此 $CJ\rightarrow I$ 的左部不是极小的。可以删除 J，得到 $C\rightarrow I$。

现在，我们有 F 的等价函数依赖集 $F=\{ABD\rightarrow E, AB\rightarrow G, B\rightarrow K, C\rightarrow J, C\rightarrow I, G\rightarrow H\}$。容易验证它是极小的。

关系模式 R 的码是 ABCD。

6.10 初始化表在（ a）中。

$A\rightarrow C$ 将 b_{23} 和 b_{53} 改变为 b_{13}， $B\rightarrow C$ 将 b_{33} 改变为 b_{13}，结果表在（ b）中。

A	B	C	D	E
a_1	b_{12}	b_{13}	a_4	b_{15}
a_1	a_2	b_{23}	b_{24}	b_{25}
b_{31}	a_2	b_{33}	b_{34}	a_5
b_{41}	b_{42}	a_3	a_4	a_5
a_1	b_{52}	b_{53}	b_{54}	a_5

(a)

A	B	C	D	E
a_1	b_{12}	b_{13}	a_4	b_{15}
a_1	a_2	b_{13}	b_{24}	b_{25}
b_{31}	a_2	b_{13}	b_{34}	a_5
b_{41}	b_{42}	a_3	a_4	a_5
a_1	b_{52}	b_{13}	b_{54}	a_5

(b)

$C\rightarrow D$ 将 b_{24}、b_{34} 和 b_{54} 改变为 a_4， $DE\rightarrow C$ 将 b_{13} 改变为 a_3，结果表在（ c）中。

$CE\rightarrow A$ 将 b_{31} 和 b_{41} 改变为 a_1，结果表在（ d）中。此时，表的第三行为 $a_1a_2a_3a_4a_5$，因此 ρ 是无损连接分解。

A	B	C	D	E
a_1	b_{12}	a_3	a_4	b_{15}
a_1	a_2	a_3	a_4	b_{25}
b_{31}	a_2	a_3	a_4	a_5
b_{41}	b_{42}	a_3	a_4	a_5
a_1	b_{52}	a_3	a_4	a_5

(c)

A	B	C	D	E
a_1	b_{12}	a_3	a_4	b_{15}
a_1	a_2	a_3	a_4	b_{25}
a_1	a_2	a_3	a_4	a_5
a_1	b_{42}	a_3	a_4	a_5
a_1	b_{52}	a_3	a_4	a_5

(d)

6.11 因为 $R_1\cap R_2=S$， $R_1-R_2=A$，而 $S\rightarrow A\in F$，所以 $\rho=\{R_1(S, A), R_2(S, I, P)\}$ 是无损连接分解。

6.12 使用完全函数依赖概念， 2NF 的等价定义如下：

如果关系模式 $R\in 1NF$，并且每一个非主属性都完全函数依赖于 R 的码，则 $R\in 2NF$。

6.13 证明：我们用反证法证明。

设 R 满足 6.5.2 节的 3NF 定义。假设 R 不满足习题中的 3NF 定义，则存在非主属性 A，它传递地依赖于 R 的某个码， 设为 K。于是，存在 Y 使得 $Y\rightarrow K$ 在 R 中不成立，但是 A 既不属于 K 也不属于 Y，并且 $Y\rightarrow A$。由于 $Y\rightarrow K$ 在 R 中不成立， Y 不是 R 的超码。又因为 A 不

是主属性，因此 Y→A 违反 6.5.2 节的 3NF 定义，这与 R 满足 6.5.2 节的 3NF 定义相矛盾。这一矛盾表明“ R 不满足习题中的 3NF 定义”的假设不成立。

设 R 满足习题中的 3NF 定义。假设 R 不满足 6.5.2 节中的 3NF 定义，则存在属性 A，属性集 Y 使得 Y→A，但是 A 不是主属性，而 Y 不是 R 的超码。设 K 为 R 的超码，则 K→Y，但 Y→K 在 R 中不成立。于是非主属性 A 传递地依赖于 R 的码 K，与 R 满足习题中的 3NF 定义相矛盾。这一矛盾表明“ R 不满足 6.5.2 节中的 3NF 定义”的假设不成立。

综上，习题中的 3NF 定义与 6.5.2 节的 3NF 定义等价。

6.14 证明：

(1) 如果 R 的所有属性都是主属性， 则 R 中的任何函数依赖都不违反 3NF 条件，因此 R 是 3NF。

(2) （反证法）设 R 的码包含 R 的的所有属性（全码），但 R 不是 BCNF。于是， R 中存在 X→A（即 $X \rightarrow A \in F^+$），并且 $A \notin X$，但是 X 不是 R 的超码。令 $Y=R-\{A\}$。由于 $A \notin X$，于是 $X \subseteq Y$。由此， Y→X。注意到 X→A，于是 Y→A。又因为 $Y=R-\{A\}$，于是 Y→R。这与 R 的码包含所有属性矛盾。这一矛盾表明“ R 不是 BCNF”的假设不成立。

6.15 证明：

(1) 证明：如果关系模式 R 是 3NF，则 R 一定是 2NF。举例说明其逆不真。

设 R 是 3NF。假设 R 不是 2NF。于是， R 中存在 X→A，并且 $A \notin X$，但是 A 不是主属性，而 X 是 R 的某个码（设为 K）的真子集。因为 X 是 K 的真子集，因此 X 不可能是 R 的超码（否则与 K 是 R 的码矛盾）。X→A 违反 3NF 定义，于是 R 不是 3NF，与设 R 是 3NF 矛盾。这一矛盾表明“ R 不是 2NF”的假设不成立。

例如，考虑关系模式 SDL(Sno, Sdept, Sloc)，其中 Sn0、Sdept 和 Sloc 分别为学生的学号、所在系和住处。假设每个系的学生住在同一座楼，我们有函数依赖：

Sno→Sdept, Sdept→Sloc

SDL 的码为 Sno。容易验证 SDL 是 2NF，但不是 3NF。

(2) 如果关系模式 R 是 BCNF，则 R 一定是 3NF。举例说明其逆不真。

设 R 是 BCNF。假设 R 不是 3NF。于是， R 中存在 X→A，并且 $A \notin X$，但是 A 不是主属性，并且 X 不是 R 的超码。这显然与 R 是 BCNF 矛盾。这一矛盾表明“ R 不是 3NF”的假设不成立。

例如，考虑关系模式 STC(S, T, C)，其中 S、T 和 C 分别表示学生、教师和课程。这里，非平凡的函数依赖为 SC→T 和 ST→C。STC 有两个码 (S,C)和(S,T)。STC∈3NF，但 STC 不是 BCNF。

(3) 如果关系模式 R 是 4NF，则 R 一定是 BCNF。举例说明其逆不真。

设 R 是 4NF。假设 R 不是 BCNF。于是， R 中存在 X→A，并且 $A \notin X$，但是 X 不是 R 的超码。如果 XA=R，则 X 是超码。因此， XA 不包含所有属性。根据 A8，X→A 意味 X→→A。由于 $XA \neq R$，并且 $A \notin X$， X→→A 违反 4NF 条件，与 R 是 4NF 矛盾。

例如，考虑关系模式 CTB(C,T,B)，其中 C 表示课程， T 表示教师， B 表示参考书。一门课程可以由多位教师讲述，但他们必须使用通一组参考书。 CTB 中存在如下多值依赖：

C→→T, C→→B

CTB 为全码，因此是 BCNF（习题 16.14）。但 C→→T 违反 4NF 条件，因此 CTB 不是 4NF。

6.16 该关系模式的码是 HS（例 6.14），所以不是 BCNF。C→T 违反 BCNF 条件，使用它将 CTHRSG 分解为 {CT, CHRSG}。

CHRSG 的码仍然是 HS，不是 BCNF。CS→G 违反 BCNF 条件，使用它将 CHRSG 分解为{ CSG, CHRS}。

CHRS 的码仍然是 HS，不是 BCNF。HR→C 违反 BCNF 条件，使用它将 CHRS 分解为{HRC，HRS}。

现在，所有的模式都是 BCNF。我们得到的具有无损连接性的 BCNF 分解 {CT，CSG，HRC，HRS}

6.17 R (B，O，I，S，Q，D)的函数依赖为 S→D，I→B，IS→Q，B→O，码为 IS。

(1) S→D 违反 BCNF 条件，使用它将 R 分解为 {SD, BOISQ}。

BOISQ 的码为 IS。B→O 违反 BCNF 条件，使用它将 BOISQ 分解为 {BO, BISQ}。

BISQ 的码为 IS。I→B 违反 BCNF 条件，使用它将 BISQ 分解为 {IB，ISQ}。

最后得到 R 的一个具有无损连接性的 BCNF 分解{SD, BO，IB，ISQ}

(2) R 的函数依赖已经是正则的，R 的一个具有无损连接性和保持函数依赖的 3NF 分解为{ SD, IB，ISQ, BO}

6.18 容易明白，R 的码为 VD 和 SD。

(1) S→T 违反 BCNF 条件，使用它将 R 分解为 { ST, SVCPD}.

SVCPD 的码为 VD 和 SD。V→SC 违反 BCNF 条件，使用它将 SVCPD 分解为 { VSC, VPD}。最后得到 R 的一个具有无损连接性的 BCNF 分解 {ST, VSC, VPD}

(2) R 的函数依赖已经是正则的，R 的一个具有无损连接性和保持函数依赖的 3NF 分解为{ ST, VSC, SDPV}

(3) 解释 R 为什么不存在具有无损连接性和保持函数依赖的 BCNF 分解 .

S→T， V→SC 和 SD→PV

6.19 例子很多，但需要注意的是：多值依赖的成立依赖于属性集，举例时需要说明属性集。

6.20 (1) 用定义和公理两种方法证明，其余用公理证明。

(1) 多值依赖的合并规则：如果 X→→Y，X→→Z 成立，则 X→→YZ 成立。

证明：方法一：使用公理证明：

对 X→→Y 用 X 增广得到 X→→XY。对 X→→Z 用 Y 增广得到 XY→→YZ。使用自反律，得到 XY→→(U－XY－YZ)。而 U－XY－YZ=U－X－Y－Z，于是 XY→→(U－X－Y－Z)。由 X→→XY 和 XY→→(U－X－Y－Z)，使用多值依赖的传递律得到 X→→((U－X－Y－Z) －XY)。而 (U－X－Y－Z)－XY =U－X－Y－Z。因此，我们有 X→→(U－X－Y－Z)。再次利用自反律，得到 X→→(U－X－(U－X－Y－Z))。由于 U－X－(U－X－Y－Z)= YZ，于是我们有 X→→YZ。

方法二：使用定义证明：设 U 为属性全集。

设 r 是属性集 U 上的任意关系，满足 X→→Y 和 X→→Z。设 t_1和 t_2是 r 的任意元组，满足 $t_1[X]= t_2[X]$。为证明 X→→YZ 成立，我们只要证明存在 $t_3,t_4 \in r$，使得 (1) $t_3[X]=t_4[X]= t_1[X]= t_2[X]$，(2) $t_3[YZ]= t_1[YZ]$，并且 $t_3[U－XYZ]= t_2[U－XYZ]$，(3) $t_4[YZ]= t_2[YZ]$，并且 $t_4[U－XYZ]= t_1[U－XYZ]$。

由于 $t_1,t_2 \in r$，X→→Y，存在 $u_1,u_2 \in r$，使得 (1) $u_1[X]= u_2[X]= t_1[X]= t_2[X]$，(2) $u_1[Y]= t_1[Y]$，并且 $u_1[U－XY]= t_2[U－XY]$，(3) $u_2[Y]= t_2[Y]$，并且 $u_2[U－XY]= t_1[U－XY]$。

$t_1,u_1 \in r$，X→→Z，存在 $u_3,u_4 \in r$，使得 (1) $u_3[X]= u_4[X]= t_1[X]= u_1[X]$，(2) $u_3[Z]= t_1[Z]$，并且 $u_3[U－XZ]= u_1[U－XZ]$，(3) $u_4[Z]= u_1[Z]$，并且 $u_4[U－XZ]= t_1[U－XZ]$。

$t_2,u_2 \in r$，X→→Z，存在 $u_5,u_6 \in r$，使得 (1) $u_5[X]= u_6[X]= t_2[X]= u_2[X]$，(2) $u_5[Z]= t_2[Z]$，并

且 $u_5[U-XZ]=u_2[U-XZ]$，(3) $u_6[Z]=u_2[Z]$，并且 $u_6[U-XZ]=t_2[U-XZ]$。

这些元组如下所示：

	X	Y	Z	U–X–Y
t_1	$a_1\cdots a_i$	$a_{i+1}\cdots a_j$	$a_{j+1}\cdots a_k$	$a_{k+1}\cdots a_n$
t_2	$a_1\cdots a_i$	$b_{i+1}\cdots b_j$	$b_{i+1}\cdots b_k$	$b_{k+1}\cdots b_n$
u_1	$a_1\cdots a_i$	$a_{i+1}\cdots a_j$	$b_{j+1}\cdots b_k$	$b_{j+1}\cdots b_n$
u_2	$a_1\cdots a_i$	$b_{i+1}\cdots b_j$	$a_{i+1}\cdots a_k$	$a_{j+1}\cdots a_n$
u_3	$a_1\cdots a_i$	$a_{i+1}\cdots a_j$	$a_{j+1}\cdots a_k$	$b_{j+1}\cdots b_n$
u_4	$a_1\cdots a_i$	$a_{i+1}\cdots a_j$	$b_{j+1}\cdots b_k$	$a_{j+1}\cdots a_n$
u_5	$a_1\cdots a_i$	$b_{i+1}\cdots b_j$	$b_{j+1}\cdots b_k$	$a_{j+1}\cdots a_n$
u_6	$a_1\cdots a_i$	$b_{i+1}\cdots b_j$	$a_{i+1}\cdots a_k$	$b_{j+1}\cdots b_n$

令 $t_3=u_3$，$t_4=u_5$，于是 $t_3,t_4\in r$，并且 (1) $t_3[X]=t_4[X]=t_1[X]=t_2[X]$，(2) $t_3[YZ]=t_1[YZ]$，并且 $t_3[U-XYZ]=t_2[U-XYZ]$，(3) $t_4[YZ]=t_2[YZ]$，并且 $t_4[U-XYZ]=t_1[U-XYZ]$。

(2) 多值依赖的伪传递规则：如果 $X\rightarrow\rightarrow Y$，$WY\rightarrow\rightarrow Z$ 成立，则 $WX\rightarrow\rightarrow(Z-WY)$ 成立。

证明：用 W 增广 $X\rightarrow\rightarrow Y$，得到 $WX\rightarrow\rightarrow WY$。注意到 $WY\rightarrow\rightarrow Z$ 成立，使用多值依赖的传递率，我们有 $WX\rightarrow\rightarrow(Z-WY)$。

(3) 混合伪传递规则：如果 $X\rightarrow\rightarrow Y$，$XY\rightarrow Z$ 成立，则 $X\rightarrow(Z-Y)$ 成立。

证明：：用 X 增广 $X\rightarrow\rightarrow Y$，得到 $X\rightarrow\rightarrow XY$。由 $XY\rightarrow Z$ 和 A7 得到 $XY\rightarrow\rightarrow Z$。由 $X\rightarrow\rightarrow XY$ 和 $XY\rightarrow\rightarrow Z$，利用多值依赖的传递率得到 $X\rightarrow\rightarrow(Z-XY)$。由于 $X\rightarrow\rightarrow(Z-XY)$，$(Z-XY)\subseteq(Z-XY)$，XY 与 $(Z-XY)$不相交，并且 $XY\rightarrow Z$，A8 表明 $X\rightarrow(Z-Y)$ 成立。

(4) 多值依赖的分解规则：如果 $X\rightarrow\rightarrow Y$，$X\rightarrow\rightarrow Z$ 成立，则 $X\rightarrow\rightarrow(Y\cap Z)$，$X\rightarrow\rightarrow(Y-Z)$，$X\rightarrow\rightarrow(Z-Y)$ 成立。

证明：由 $(Y-Z)\subseteq Y$ 得到 $Y\rightarrow(Y-Z)$。再利用 A7，得到 $Y\rightarrow\rightarrow(Y-Z)$。由 $X\rightarrow\rightarrow Y$ 和 $Y\rightarrow\rightarrow(Y-Z)$，利用多值依赖的传递率得到 $X\rightarrow\rightarrow((Y-Z)-Z)$。此即 $X\rightarrow\rightarrow(Y-Z)$。

类似地，可以证明 $X\rightarrow\rightarrow(Z-Y)$。

由 $X\rightarrow\rightarrow Y$ 和 $Y\rightarrow\rightarrow(Y-Z)$（上面已证），利用多值依赖的传递率，我们有 $X\rightarrow\rightarrow(Y-(Y-Z))$。而 $Y-(Y-Z)=Y\cap Z$，因而有 $X\rightarrow\rightarrow(Y\cap Z)$。

6.21 只考虑函数依赖，$\rho=\{CHR，CT，CSG\}$ 不是无损连接分解。这可以用算法 6.3 检测。初始表如（ a）所示。

C	T	H	R	S	G
a_1	b_{12}	a_3	a_4	b_{15}	b_{16}
a_1	a_2	b_{23}	b_{24}	b_{25}	b_{26}
a_1	b_{32}	b_{33}	b_{34}	a_5	a_6

(a)

C	T	H	R	S	G
a_1	a_2	a_3	a_4	b_{15}	b_{16}
a_1	a_2	b_{23}	b_{24}	b_{25}	b_{26}
a_1	a_2	b_{33}	b_{34}	a_5	a_6

(b)

$C\rightarrow T$ 将 b_{12} 和 b_{32} 改为 a_2，结果如（ b）所示。任何函数依赖都不能再改变该表，算法结束。由于不存在一行为 $a_1a_2a_3a_4a_5a_6$，因此 $\rho=\{CHR, CT, CSG\}$ 不是无损连接分解。

6.22 通过模式分解产生关系模式（仅考虑函数依赖）的三个目标是：无损连接分解、BCNF 和保持函数依赖的分解。

分解的无损连接性是对分解的基本要求，是必须保证的。不具有无损连接性的分解是有害的。这是因为关系模式 R 的分解 $\rho=\{R_1, R_2,\cdots, R_k\}$ 将原关系模式下的关系 r 分解成 k 个关系 $r_1, r_2,\cdots, r_k$。除了自然连接之外，没有其他一般性方法从这些关系得到原来的关系。因此，对 r 和 $r_1, r_2,\cdots, r_k$ 进行相同的查询可能得到不同的结果。这样，我们就不能认为 R 和 $\{R_1, R_2,\cdots, R_k\}$ 反映相同的现实世界。

分解的目的是减少冗余，从而消除存储异常。仅考虑函数依赖，不达到 BCNF 不能完全避免冗余，因此分解后的关系模式，如果可能的话，应该达到 BCNF。

函数依赖反映了数据之间的一种约束。如果分解保持函数依赖，这些约束可以比较有效地在分解后的一个关系中进行验证，否则有些约束需要将多个关系连接才能验证。因此，分解要尽可能保持函数依赖。

在某些情况下，BCNF 和保持函数依赖这两个目标不可兼得，因为有些关系模式不存在保持函数依赖的 BCNF 分解。在这种情况下，我们必须在 BCNF 和保持函数依赖之间进行取舍。

6.23 在关系数据库设计时，我们可能选择非 BCNF 设计的原因是有些关系模式不存在保持函数依赖的 BCNF 分解。当函数依赖是重要的时，为了使的函数依赖的验证更加有效，我们需要选择保持函数依赖的分解，而放弃 BCNF.

6.24 在关系数据库设计时，没有理由设计一个属于 2NF，但不属于更高范式的关系模式。因为关系模式达到 2NF，但不属于更高的范式将存在明显的数据冗余，进而导致存储异常。另一方面，对于任意 2NF，我们总可以通过模式分解将它转换成更高的关系模式，这些关系模式与原来的关系模式反映相同的现实世界（分解具有无损连接性），并且能够消除或减少数据冗余。

第 7 章　数据库设计

部分习题参考答案

7.1　数据库设计人员应具备哪些方面的知识？

1、计科和程序设计
2、数据库知识
3、软件工程
4、应用领域

7.2　简述数据库设计的步骤。

1、需求分析
2、概念设计（er图）
3、逻辑结构设计
4、物理设计
5、数据库的实施
6、运行与维护

7.3　什么是数据库设计？

1、数据项
2、数据结构
3、数据流
4、数据存储
5、处理过程

7.4　简述需求分析的步骤。

7.5　简述数据字典的内容及其作用。

7.6　概念数据库设计使用那些策略？

7.7　简述概念结构设计的基本方法和步骤。

自顶向下，概念结构设计独立于具体的DBMS

7.8　什么是数据库的逻辑设计？简述其步骤。

逻辑结构设计依赖于逻辑数据模型和DBMS，任务是将概念结构设计的ER图转换为关系模式

7.9　在合并局部　E-R 图中，如何消除各种冲突？

三种冲突：属性（需求协商）、命名、结构

7.10　E-R 图向关系模型转换的原则是什么？

7.11　简述物理数据库设计的任务、目标和步骤。

为一个给定数据库的逻辑结构选取一个最合适应用环境的物理结构和存取方法的过程。
依赖于具体的计算机系统。

7.12　在数据库的物理设计中如何选择索引方法？

7.13　试述聚簇设计的原则。

把某个或某些属性上具有相同值的元组集中存放在同一个物理块或相邻物理块或同一柱面内，称为聚簇

7.14　为什么要对数据库进行重组和重构？

使用时间增加，磁盘内碎块增多，重组来提高性能。常用方法：卸载再重新加载。
在原理设计的基础上适当扩充和修改称为重构，为了适应业务变化。

7.15　DBA　的作用是什么？

7.16　考察自己学校的学生成绩管理方法，编写出建立成绩管理的可行性分析报告、需求分析说明书、系统的概念结构设计和逻辑结构设计。

7.17　一个图书借阅管理数据库有以下需求，请设计该数据库的　　E－R 模式和关系模式。

(1)　希望能查询书库中现有书籍的种类、数量和存放位置。这里假设各种书籍均由书号惟一标识。

(2)　希望能查询书籍的借还情况信息，包括借书人单位、借书人单位电话、姓名、借书证号、借书日期、还书日期。这里规定，每个人最多可以借　　6 本书，借书证是读者的惟一标识符。

(3)　希望通过查询出版社的名称、电话、邮编、通讯地址等信息能向出版社订购有关书籍。这里假设一个出版社可以出版多种图书，同一书名的书仅由一个出版社出版，出版社名称具有惟一性。

第 8 章　查询处理与优化

部分习题参考答案

查询优化正关系数据库中的必要性和可能性

8.1　对于相同的查询，不同的查询执行计划的时间开销可能相差几个数量级。这意味好的执行计划可能是可行的，而差的执行计划在实践上是不可行。因此，即使查询只执行一次，查询优化也是必须的。

在非关系系统中，用户使用过程化的语言表达查询要求，执行何种操作，以及操作的序列都是由用户来决定的。系统为用户提供选择存取路径的手段，而用户必须了解存取路径，为自己的查询选择合适的存取路径。如果用户做了不当的选择，系统很难加以改进。关系数据库语言（如 SQL）是非过程化语言；为了表达查询，用户只需要说明做什么（查询什么、在什么关系上查询和查询结果满足的条件），而不必说明怎么做。这就为查询优化提供了更大的空间，使得系统可以分析查询语义，选择最佳的查询处理方案。

8.2　系统进行查询优化的优点是：系统进行查询优化减轻了用户选择存取路径的负担，使得用户可以将注意力放在如何正确地表达查询请求上，而不必考虑如何最有效地表达查询。此外，系统进行查询优化还可以比用户做得更好，因为

(1) 优化器可以从数据字典中获取许多统计信息，如每个关系的当前元组数目、每个属性上的不同值个数、有哪些索引等。这些信息对于选择高效的执行策略是重要的，但是用户难以获得这些信息。

(2) 当数据库的统计信息改变时，可能需要改变执行策略。优化器可以自动对查询重新进行优化，以选择相适应的执行策略。在非关系系统中必须重写程序，而重写程序在实际应用中往往是不太可能的。

(3) 优化器可以更全面地考虑，考察数百种不同的执行计划，从中确定最佳的执行计划。而用户一般只能考虑有限的几种可能性（很难超过三、四种）。

(4) 优化器中包括了很多复杂的优化技术，这些优化技术往往只有最好的程序员才能掌握。系统的自动优化相当于使得所有人都拥有这些优化技术。

8.3　代数优化的启发式规则包括：选择尽可能先做、投影机可能先做和尽量避免计算笛卡尔积等。

选择尽可能先做是最重要的启发式规则。这是因为尽管通常一个关系很大，但是满足某种选择条件的元组的数量却相对较小，并且选择条件越强，满足条件的元组越少。尽早进行选择能够大幅度减少下一步参与运算的元组数目，使得其后的运算可以更加有效的进行。

8.4　由于索引是非聚簇索引，如果对于连接属性的相同值，内层关系存在多个元组，这些元组可能散布在多个物理块，则对于外层关系的每个元组，我们可能需要访问内层关系的多个块。因此，索引嵌套循环的效率不高。

8.5　如果关系 r 和 s 在公共属性上无序，也没有索引。如果内存足够大，就 I/O 开销而言，计算 $r \bowtie s$ 的最有效方法是：将较小的关系整个读入内存，逐块读入较大的关系，使用较大的关系作为外层关系，执行嵌套循环连接。此时，I/O 操作数为 br + bs，而内存需求为 min(br, bs) + 2 页，其中 br 和 bs 分别为关系 r 和 s 所占用的块数。

8.6　假设关系 r 和 s 的公共属性 JoinAttrs 是 r 的主码、s 的外码，并且 r 和 s 在连接属性上是

有序的（假设为递增序）。用排序-归并方法计算 $r \bowtie s$ 的伪代码如下：

指针 p_r 和 p_s 分别指向 r 和 s 的第一个元组；

while ($p_r \neq$null **and** $p_s \neq$null) **do** {

$t_r \leftarrow$ p_r 指向的元组；

$t_s \leftarrow$ p_s 指向的元组；

while ($p_s \neq$null **and** t_s[JoinAttrs]< t_r[JoinAttrs]) **do**

让 p_s 指向 s 的下一个元组；

while ($p_s \neq$null **and** t_s[JoinAttrs]= t_r[JoinAttrs]) **do** {

把 $t_r.t_s$ 添加到结果中；

让 p_s 指向 s 的下一个元组；

}

if ($p_r \neq$nulll) **then**

让 p_r 指向 r 的下一个元组；

}

8.7 可以采用排序或散列方法来消除重复元组。

排序时等值元组相互邻近，删除其他副本，只留一个元组副本即可。如果使用外部归并排序，则可以直接在归并时删除重复元组。

散列时，重复元组会分到同一桶中。可以在元组加入桶中时检查元组是否已在桶中，无须加入已在桶中的元组。

8.8

(1) 使用散列方法实现集合运算 $r \cup s$ 的算法

初始：结果关系为空；

使用相同的散列函数 H 对两个关系 r 和 s 进行划分，产生 $r_0, r_1, \ldots, r_n$ 和 $s_0, s_1, \ldots, s_n$;

for (i=1; i<=n; i++) **do** {

对 r_i 建立内存散列索引；

for (s_i 中的每个元组 t_s) **do**

if (t_s 不在 r_i 的散列索引中) **then**

将 t_s 加入到 r_i 的散列索引中；

将 r_i 的内存散列索引中的元组加入结果关系中；

}

设关系 r 和 s 分别占 br 和 bs 块。使用散列函数 H 对两个关系 r 和 s 进行划分，需要读入和写出 br+bs 块。读入每个 r_i 建立它的内存散列索引，并将 s_i 的不在 r_i 的散列索引中的元组插入需要读入 br+bs 块。整个算法需要的 I/O 量为 3(br+bs)块。

如果内存足够大，可以将关系 r 和 s 读入到内存，I/O 减少到 br+bs 块。

(2) 使用散列方法实现集合运算 $r \cap s$ 的算法

初始：结果关系为空；

使用相同的散列函数 H 对两个关系 r 和 s 进行划分，产生 $r_0, r_1, \ldots, r_n$ 和 $s_0, s_1, \ldots, s_n$;

for (i=1; i<=n; i++) **do** {

对 r_i 建立内存散列索引；

for (s_i 中的每个元组 t_s) **do**

if (t_s 在 r_i 的散列索引中) **then**

将 t_s 加入到结果关系中；

}

使用散列方法实现集合运算 r − s 的算法

初始：结果关系为空；

使用相同的散列函数 H 对两个关系 r 和 s 进行划分，产生 $r_0, r_1, ..., r_n$ 和 $s_0, s_1, ..., s_n$;

for (i=1; i<=n; i++) **do** {

对 r_i 建立内存散列索引；

for (s_i 中的每个元组 t_s) **do**

if (t_s 在 r_i 的散列索引中) **then**

将 t_s 从 r_i 的散列索引中删除；

将 r_i 的内存散列索引中的元组加入结果关系中；

}

8.9

(1) 图 8.2 的语法树见（ a），分解合取选择后的语法树在（ b）中。

选择条件 Pno=,P001?只涉及关系 SP，因此可以将对应的选择下移，结果见（ c）。

将选择与其下的笛卡尔积转换成自然连接，结果见（ d）。

引入新的投影，去掉其后运算不需要的属性：关系 Suppliers 的属性 Sno和Sname是其后需要的，其余属性不需要。使用条件 Pno=,P001? 选择后，关系 SP的属性 Sno是需要的，其余属性不需要。引入投影后的结果在（ e）中。

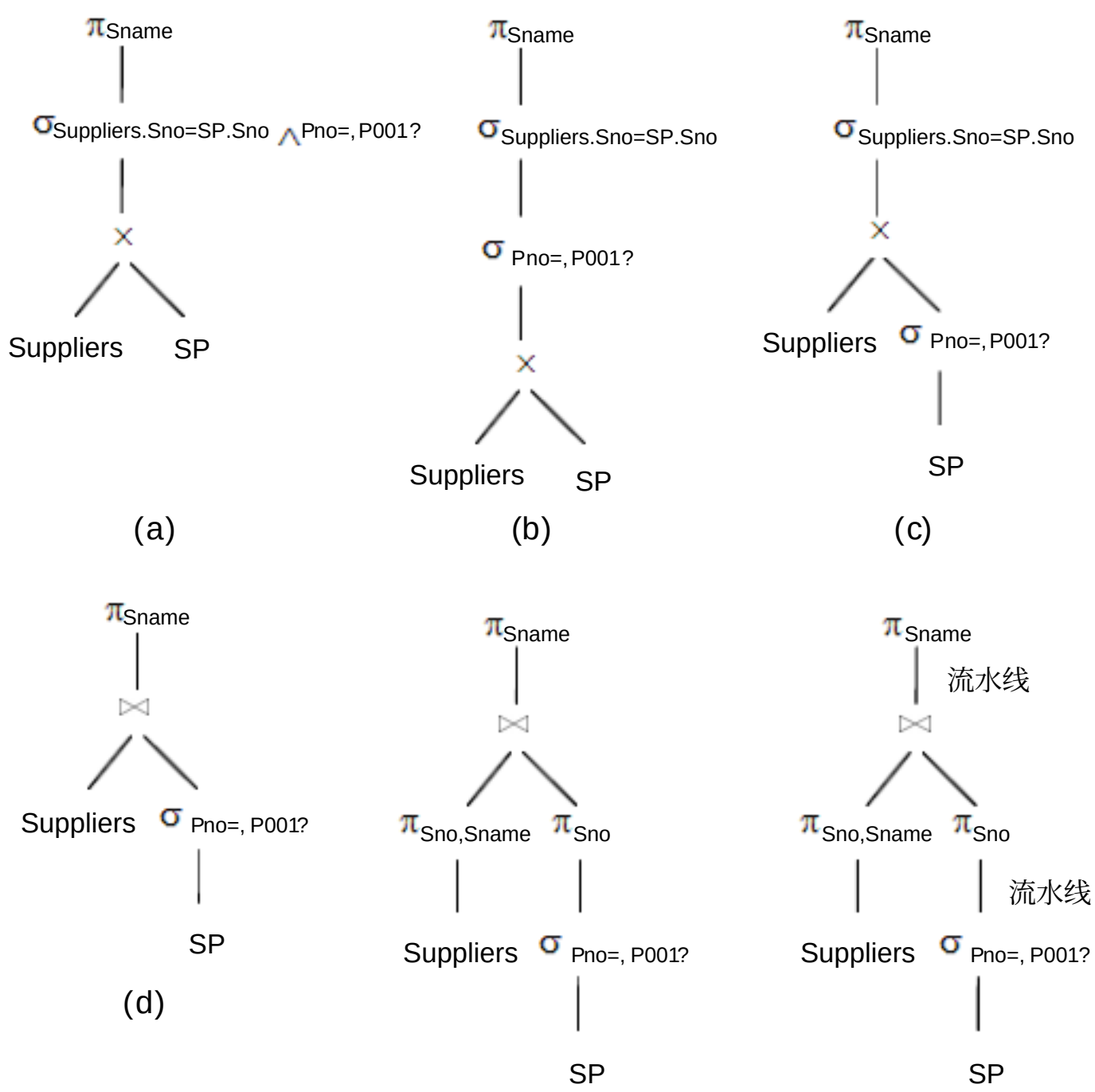


图 8.1 习题 8.9 的初始语法树及其优化

不假定自然连接可以有效计算。自然连接上方投影与该自然连接形成一个流水线，SP 上方两个相继的一目运算形成一个流水线，得到代数优化结果在（f）中。

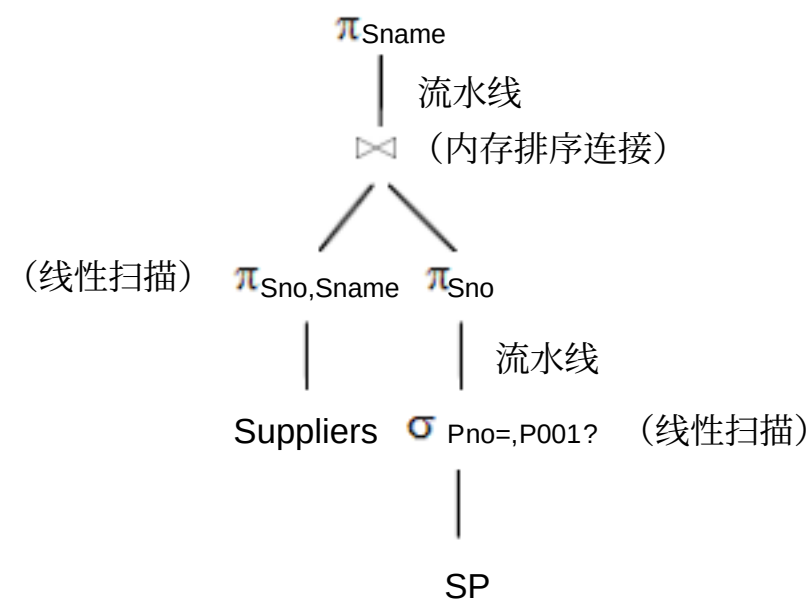
(2)

假设数据库中有 100 个 Suppliers 记录，10000 个 SP 记录，其中涉及产品 P001 的发货记录为 50 个。不对 Suppliers 和 SP 做任何有序或存在索引的假定。

将 Suppliers 投影到 Sno 和 Sname 上可以用线性扫描实现。由于结果元组只有 100 个，每个只有两个属性，结果可以放在内存。

从 SP 选择满足条件 Pno=,P001?的可以用线性扫描实现。由于只有 50 个记录，投影到 Sno 上之后可以放在内存。

自然连接可以用简单的内存排序连接（排序-归并连接的一种简单情况）实现。结果查询计划在（g）中。



(g)

图 8.1（续）习题 8.9 的初始语法树及其优化

8.10

查询的初始语法树在（a）中，其中关系 Accounts、Branches 和 Customers 分别简写为 A、B 和 C（下同）。

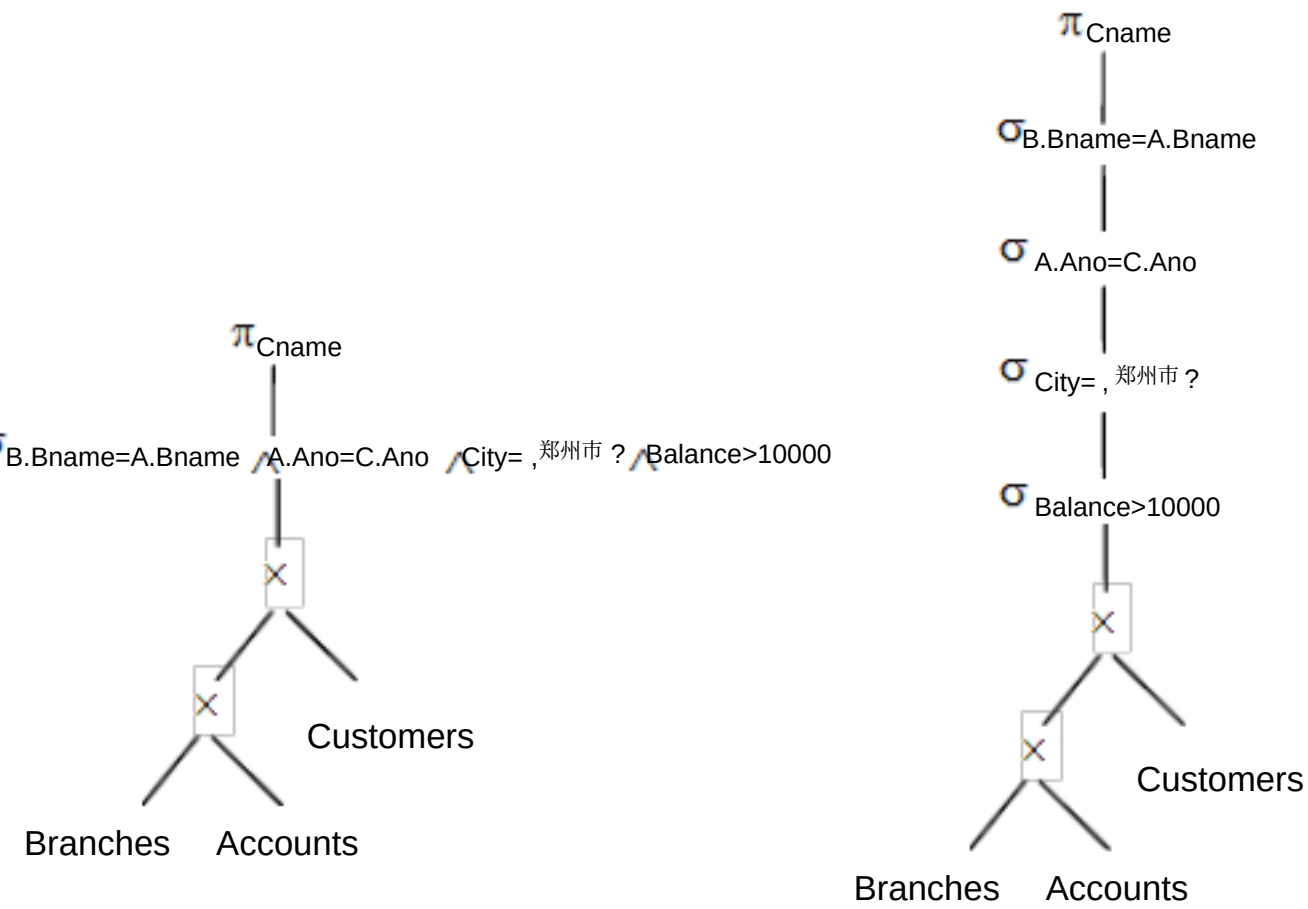


图 8.2 习题 8.10 的初始语法树及其优化

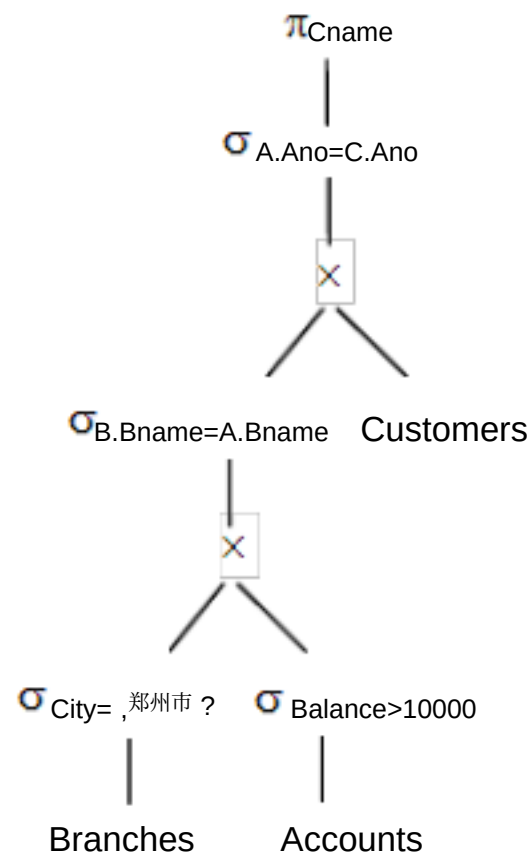
分解合取选择后的语法树在（　b）中。

选择条件　City= ,郑州市 ?只涉及关系　Branches，Balance>10000 只涉及关系　Accounts，可以分别下移到对应关系的上方；选择条件　B.Bname=A.Bname　只涉及关系　Branches　和 Accounts，可以下移到对应关系的笛卡尔积的上方。结果在（　c）中。

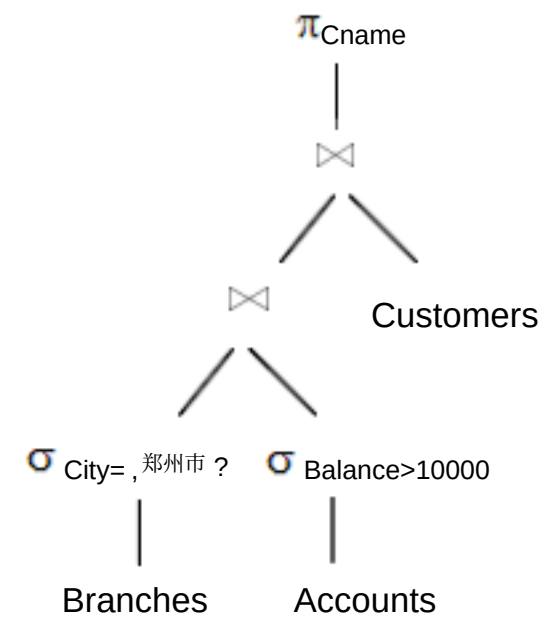
选择条件　A.Ano=C.Ano　和 B.Bname=A.Bname 可以分别与其下的笛卡尔积结合成自然连接，结果在（　d）中。

引进新的投影，去掉其后运算不再需要的属性，结果在（　e）中。

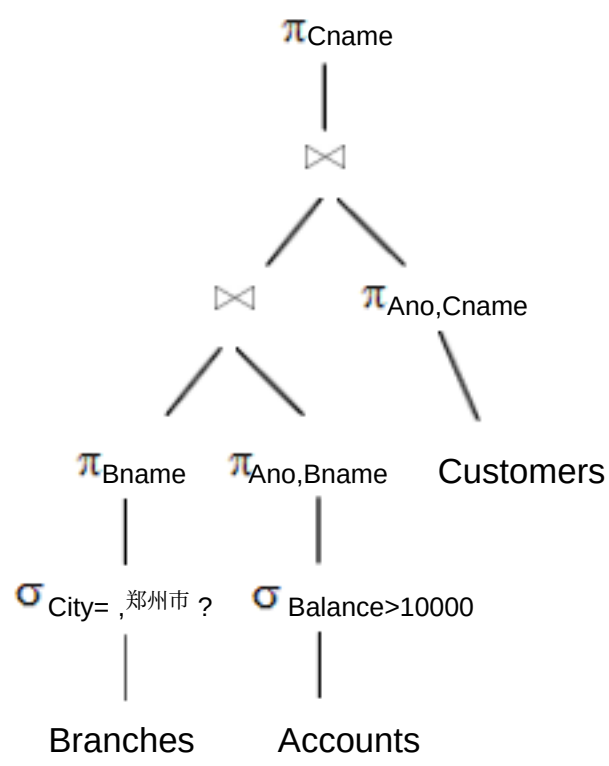
建立流水线：相继的一目运算形成流水线，上面自然连接上方的投影和　Customers 上方的投影形成流水线。不考虑诸关系的序和索引的话，不能再建立其他流水线。结果在（　f）中。



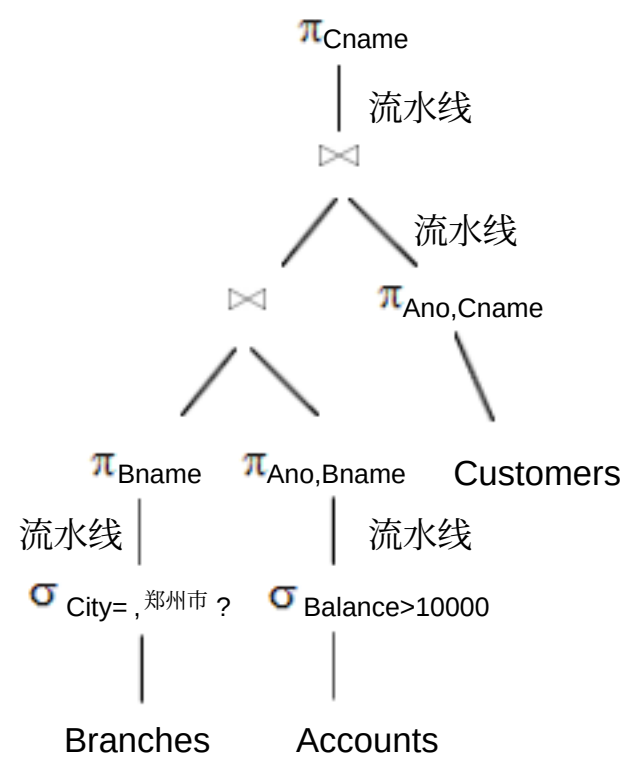
(c)



(d)



(e)



(f)

图 8.2（续）　习题 8.10 的初始语法树及其优化

8.11 例 8.6 假设：

关系数目分别为　n_C=400，n_S=20000，n_{SC}=100000；

所占的物理块数分别为 $b_C=8$，$b_S=500$，$b_{SC}=500$；

这些关系在部分属性上的不同值包括： $V(C, Cno)= n_C=400$， $V(S, Sno)= n_S=20000$， $V(S, Dno)=20$， $V(SC, Sno)= n_S=20000$， $V(SC, Cno)= n_C=400$；

Students 在 Dno 上有聚簇索引， SC 在 Sno 和 Cno 上分别有非聚簇索引， 并在 (Sno, Cno) 上有组合索引

此外，假设内存可以存放 20 个物理块（每块 2K 字节）。

由于 $V(S, Dno)=20$， Students 在 Dno 上的聚簇索引只占 1 块。

由 $b_s = \lceil n_s/f_s \rceil$ 得到 $500= \lceil 20000/f_s \rceil$，从而 $f_s=40$。

使用聚簇索引、等值选择计算 $\sigma_{S.Dno=?IE?}$(Students), 结果元组大约 1000 个, 大约在 1000/40 = 25 个物理块中，加上读索引 1 块 I/O，共需 26 块 I/O。结果元组直接投影到 Sno 上，结果 5 块放在内存。这样，计算 $\pi_{S.Sno}(\sigma_{S.Dno=?IE?}$(Students)) 需要 26 块 I/O。

使用块嵌套循环计算 $E_2= \pi_{SC.Sno,SC.Cno}(SC) \bowtie \pi_{S.Sno}(\sigma_{S.Dno=?IE?}$(Students))，$\pi_{SC.Sno,SC.Cno}(SC)$ 为内层关系。 $\pi_{S.Sno}(\sigma_{S.Dno=?IE?}$(Students)) 已经在内存，读入 SC 的元组时可以直接投影到 Sno 和 Cno 上，需要读入 SC 的所有元组，共 500 块。自然连接结果大约需要 72 块，需要作为临时关系保存。这样，计算 E_2 的 I/O 开销为 572 块。

使用块嵌套循环计算自然连接 $\pi_{C.Cno,\ C.Cname}$(Courses) $\bowtie E_2$， $\pi_{C.Cno,\ C.Cname}$(Courses) 为外层关系。由于 Courses 只有 8 块, 可以一次读入内存， 读入时直接投影得到 $\pi_{C.Cno,\ C.Cname}$ (Courses)。计算自然连接时只需要读入 E_2 的 72 块，总 I/O 开销为 80 块。

最终的计算结果 “信息工程学院学生选修的所有课程的课程号和课程名” 可以在产生元组时直接插入排序放删除重复，因此不需要附加的 I/O。

整个查询计划的 I/O 开销为： 26+572+80=678 （块）

第 9 章　事务与并发控制

部分习题参考答案

9.1　数据库中要有并发机制的主要有如下两条理由：

(1)　提高吞吐量和资源的利用率；

(2)　减少平均等待时间和平均响应时间。

并发控制技术能够保证事务的　ACID　性质，主要是保证事务的隔离性和数据库的一致性。

包括不可重复读和幻读

9.2　事务并发执行可能导致丢失修改、读“脏”数据和不可重复读等问题。读时不加锁、更新加排它锁

数据库系统解决这些问题的主要方法是采用封锁技术。一级锁协议可以保证不丢失修改，二级封锁协议可以避免读“脏”数据，而三级封锁协议可以保证可重复读。

读前加共享锁，读完释放　　读前加共享锁，事务完释放

9.3　封锁就是事务　T 在对某个数据对象操作之前，　先向系统发出加锁请求，　加锁后事务　T 就对该数据对象有了一定的控制权，在事务　T 释放它的锁之前，其它事务不能更新该数据对象。

共享锁和排它锁是两种基本锁类型。

共享锁又称读锁。　如果事务　T 获得了数据对象　Q 上的共享锁，　则 T 可以读但不能写　Q，并且在　T 释放　Q 上的　S 锁之前，其它事务只能获得　Q 上的　S 锁，而不能获得　Q 上的　X 锁。

排他锁又称写锁。　如果事务　T 获得了数据项　Q 上的排它锁，　则 T 既可以读又可以写　Q，但是在　T 释放　Q 上的　X 锁之前，其它事务既不能获得　Q 上的　S 锁，也不能获得　Q 上的　X 锁。

9.4　如何用封锁机制保证数据的一致性？

9.5　活锁又称饥饿，是某个事务因等待锁而处于无限期等待状态。

活锁是不公平的锁调度导致的。　可以采用先来先服务的策略来避免某个事务无限期期等待。即当多个事务请求封锁同一数据对象时，　封锁子系统按请求封锁的先后次序对事务排队。数据对象上的锁一旦释放，就将锁授予申请队列中的第一个事务。

9.6　死锁是两个或两个以上的事务之间的循环等待现象。死锁发生时，两个或多个事务都处于等待状态，每个事务都等待其它事务释放锁，以便可以继续执行。

预防死锁的基本方法是破坏死锁产生的条件，可以采用一次封锁、顺序封锁方法。

9.7　请给出检验死锁发生的一种方法，当发生死锁后如何解除死锁？

检测：超时法、等待图法

解除死锁：选择一个或多个处于死锁状态的事务，将其撤销并释放这些事务锁持有的所有的锁，打破循环等待条件，解除死锁。

9.8　什么样的并发调度是正确的调度？

并发调度正确性准则是可串行化。（两种形式：冲突可串行化、视图可串行化）

两段锁协议是可串行化调度的充分条件。

9.9　设 T_1、T_2、T_3 是如下的　3 个事务，设　A 的初值为　0：

T_1：A:=A+2

T_2：A:=A*2

T_3：A:=A**2　；（$A \leftarrow A^2$）

(1)　若这　3 个事务允许并发执行，则有多少种可能的正确结果，请一一列出；

(2) 请给出一个可串行化的调度，并给出执行结果；

(3) 请给出一个非串行化调度，并给出执行结果；

(4) 若这 3 个事务都遵守两段锁协议，请给出一个不产生死锁的可串行化调度；

(5) 若这 3 个事务都遵守两段锁协议，请给出一个产生死锁的调度。

不同事务内的指令操作同一个数据对象，其中至少有一个写操作，则成为冲突的。

可串行化是并发调度的正确性准则。

交换调度中的非冲突指令得到的调度与原调度等价，称冲突可串行化。

9.10 为什么要求并发事务调度的可串行化 ?什么是冲突可串行化和视图可串行化?

两个调度视图等价：满足三个复杂的条件

如果调度等价于一个串行调度，称调度是视图可串行化的

9.11 设 $r_i(Q)$表示事务 T_i执行 read(Q)，$w_i(Q)$ 表示事务 T_i执行 write(Q)。现有 3 个事务的一个调度 $<r_3(B)r_1(A)w_3(B)r_2(B)r_2(A)w_2(B)r_1(B)w_1(A)>$，请问该调度是冲突可串行化的调度吗？为什么？

9.12 什么是两段封锁协议，是举例说明，对并发事务的一个调度是可串行化的，而这些并发事务不一定遵守两段封锁协议。

两段锁协议要求所有事务：

1、读写任意数据之前，要申请并获得对该数据对象的相应封锁

2、在释放一个锁之后，事务不能再申请任何其他锁。

把事务分为两个阶段：

扩展阶段：可申请不能释放（获得锁）

收缩阶段：可释放不可再申请（释放锁）

如果所有事务都遵守两段锁协议，那么这些事务的任何并发都是可串行化的。

9.13 为什么要引进意向锁？意向锁的含义是什么？

引进意向锁是为了提高封锁子系统的效率

9.14 试述常用的意向锁： IS 锁、IX 锁、SIX 锁，给出它们的相容矩阵。

9.15 在加意向锁和释放意向锁时要注意什么问题，如何处理？结合具体的实例进行说明。

意向锁的含义是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。

IS锁：意向共享锁，若对一个数据对象加IS锁，表示他的后裔拟加S锁。//共享锁

IX锁：意向排它锁，若对一个数据对象加IX锁，表示他的后裔拟加X锁。//排它锁

SIX锁：共享意向排它锁，若对一个数据对象加SIX锁，表示对它加S锁，再加IX锁，即SIX=S+IX

SIX对整个关系（表）加S锁，对某些元组（记录）加X锁，可满足高并发读时写

第 10 章　数据库的恢复技术

习题参考答案

易失性：
主存、缓存
io速度最快，io开销最大

非易失性：
磁盘、磁带
可重复读写，io速度一般

稳定存储器：
理想存储器，信息永不丢失，理论上不能实现
只存数据

10.1　从 I/O 开销的角度解释易失性存储器、非易失性存储器和稳定存储器的区别。

10.2　为什么说稳定存储器是不可能实现，数据库系统如何处理该问题。

理论上可以使用raid实现。但外部环境影响（例如大火或洪水），所以不可能实现
数据库系统可以多备份数据防止丢失。

10.3　说明数据库系统日志的内容和用途。

日志可以恢复事务故障和系统故障。介质故障的恢复也需要日志。
包括：事务标识符、操作符、操作对象、旧值、新值

10.4　从实现的难易程度和开销代价的角度比较延迟修改和立即修改的基于日志的恢复机制。

10.5　为什么要引入检测点？提高系统故障恢复效率，不需要搜索全部日志

10.6　当系统从崩溃中恢复时，将构造一个 UNDO-LIST 和 REDO-LIST。解释为什么 UNDO-LIST 中的事务日志记录必须由后至前处理，而 REDO-LIST 中的事务日志要由前往后进行处理。

10.7　解释如何在块输出到磁盘前，与该块有关的某些日志还未输出到稳定存储器上，缓冲区管理器如何造成数据库状态不一致。

10.8　简述非易失性存储器数据丢失的数据库恢复方法。

10.9　从实现的难易程度和开销代价的角度比较影子分页恢复机制和基于日志的恢复机制

第 11 章 XML

习题参考答案

11.1

属性：元素可以具有属性，用来描述元素的有关信息。属性在元素的开始标签中用 attribute -name=attribute-value 说明。

实体：XML 文档中，可以将重复使用的文档内容定义为实体。 实体的格式为： <! ENTITY 实体名 “ 实体内容 ” >；引用实体时使用＆实体名。

元素的上下文：在某元素的开始标签和结束标签中出现的文本就称为该元素的上下文。

命名空间： 是用于解决标签名冲突 （同样的标签名在不同组织可能具有不同的含义） 的方法。在每个标签名前面使用一个惟一的串（后随冒号）作为前缀。由于 Web URL 的惟一性，通常可以使用 Web URL。

处理指令：是为使用一段特殊代码而设计的标记，它通常用来为处理 XML 文档的应用程序提供信息。 这些信息包括如何处理文档、 如何显示文档等。 处理指令由处理指令名称和数据组成，其格式为 <?target data?> 。

注释：在 XML 中注释 是以 <!- 开始，以 ->结束，这两个字符序列之间是注释的文本内容。注释可以在 XML 文档的任何地方出现。

11.2 XML 中的数据查询使用的语言有：

XPath：是由路径表达式组成的简单语言，是其余两种查询语言的基础。

XSLT：是一种转换语言，是 XSL 最重要的部分。 XSLT 可以把 XML 文档转换成另一个 XML 文档或 HTML 文档。

XQuery：是一个具有丰富功能的 XML 查询语言，用来从 XML 文档查找和提取元素和属性。

11.3 XML 的数据可以存储在关系数据库中，也可以存储在非关系数据库中。

存储在关系数据库中

11.4 XML 作为应用程序的接口主要使用两种模型，一种是文档对象模型，另一种是事件模型。

文档对象模型（ Document Object Model ， DOM ）是由 W3C 制定的一套标准接口规范。在应用程序中， 基于 DOM 的 XML 分析器将一个 XML 文档转换成一个对象模型的集合 （这个集合通常被称为 DOM 树）。 DOM 库提供了很多函数，用来遍历 DOM 树。

第二种应用程序接口是简单 XML API （Simple API for XML ， SAX ）。 SAX 是一种事件模型， 它用于提供一个语法分析器和应用程序之间的通用接口。 SAX 提供了一种对 XML 文档进行顺序访问的模式。

11.5

```
<!DOCTYPE db [
<!ELEMENT emp (ename, children*, skills*)     >
<!ELEMENT children (name, birthday)    >
<!ELEMENT birthday (day, month, year)    >
<!ELEMENT skills (type, exams+)    >
```

```
<!ELEMENT exams (year, city)   >
<!ELEMENT ename( #PCDATA )    >
<!ELEMENT name( #PCDATA )    >
<!ELEMENT day( #PCDATA )    >
<!ELEMENT month( #PCDATA )    >
<!ELEMENT year( #PCDATA )    >
<!ELEMENT type( #PCDATA )    >
<!ELEMENT city( #PCDATA )    >
] >
```

11.6

a. XPath: /db/emp/skills/type

b. XSLT:

```
<xsl:template match=        “ /db/emp    ”
<skills >
<xsl:apply-templates/ >
</skills >
</xsl:template >
<xsl:template match=        “ /skills      ”
<type>
<xsl:value- of select=      “ type    ” /
</type >
</xsl:template >
<xsl:temp late match=      “ . ” /
```

11.7 XML 数据的 DTD ：

```
<!DOCTYPE bank [
    <!ELEMENT bank ( ( account | customer | depositor)+)>
    <!ELEMENT account (account-number branch-name balance)>
    <!ELEMENT customer(customer-name customer-street customer-city)>
    <!ELEMENT depositor (customer-name account-number)>
    <!ELEMENT account-number (#PCDATA)>
    <!ELEMENT branch-name (#PCDATA)>
    <!ELEMENT balance(#PCDA  TA)>
    <!ELEMENT customer-name(#PCDATA)>
    <!ELEMENT customer-street(#PCDATA)>
    <!ELEMENT customer-city(#PCDA    TA)>
]>
```

11.8

```
for $b in distinct (/bank/account/branch-name)
return
<branch-total>
    <branch-name> $b/text() </branch-name>
```

```
        let $s := sum (/bank/account[branch-name=$b]/balance)
        return  <total-balance> $s </total-balance>
    </branch-total>
```

11.9

```
    <lojoin>
    for $b in /bank/account,
        $c in /bank/customer,
        $d in /bank/depositor
    where $a/account-number = $d/account-number
          and $c/customer-name = $d/customer-name
    return  <cust-acct> $c $a </cust-acct>
    |
    for $c in /bank/customer,
    where every $d in /bank/depositor  satisfies
    (not ($c/customer-name=$d/customer-name))
    return  <cust-acct> $c </cust-acct>
    </lojoin>
```

11.10

(1) 一个小数据示例：自行车

```
    <parts>
        <part>
    <name>  自行车   </name>
    <subpartinfo>
        <part>
            <name>  车轮   </name>
            <subpartinfo>
                <part>
                    <name>  轮圈   </name>
                </part>
                <qty> 1 </qty>
             </subpartinfo>
             <subpartinfo>
                  <part>
                       <name>  车条   </name>
                  </part>
                  <qty> 40 </qty>
             </subpartinfo>
             <subpartinfo>
                  <part>
                       <name>  车带   </name>
                  </part>
                  <qty> 1 </qty>
              </subpartinfo>
        </part>
        <qty> 2 </qty>
    </subpartinfo>
    <subpartinfo>
        <part>
            <name>  车闸   </name>
```

```
        </part>
        <qty> 2 </qty>
    </subpartinfo>
    <subpartinfo>
        <part>
            <name>  齿轮   </name>
        </part>
        <qty> 3 </qty>
    </subpartinfo>
    <subpartinfo>
        <part>
                <name>  车架   </name>
            </part>
            <qty> 1 </qty>
        </subpartinfo>
    </part>
</parts>
```

(2) 该 DTD 转化成如下关系模式：

part(partid,name)

subpartinfo(partid, subpartid, qty)

其中， subpartinfo 的partid 和subpartid都是外码， 参照 part的partid 。