目　录

第一部分　基本知识点与习题解析

第二部分 实 验 指 导

附 录

第一部分

基本知识点与习题解析

《数据库系统概论》(第5版)(以下简称《概论》)是计算机类专业、信息管理与信息系统专业和其他相关专业的大学本科教材,常常是学生学习数据库课程的第一本书。本书是《概论》教材的配套辅导书,因此,应该在学习《概论》教材的同时阅读本书。

第一部分讲解《概论》第1章~第11章的基本知识点,对各章的习题进行解析,并增加了补充习题及其解答。

本书对所涉及的知识点进行了大致的分类:需要了解的、需要牢固掌握的和需要举一反三的。此外,还给出了每一章的难点。

对部分习题的解析是为了帮助读者更好地理解习题所涉及的基本概念和解题的方法步骤。

第1章 绪 论

由于读者刚刚步入数据库技术的新领域，刚刚开始学习《数据库系统概论》（以下简称《概论》）这本书，为了给读者一个“什么是数据库”的大致概念，在《概论》书第1章中概要介绍了什么是数据库、数据库的优点、数据库的若干最基本的概念，使读者明白为什么要学习数据库技术、为什么要使用数据库系统，以及《概论》书中讲解的主要内容。

1.1 基本知识点

本章阐述了数据库的基本概念，介绍了数据管理技术的进展、数据库技术产生和发展的背景，数据库系统的组成。

学习本章的重点在于将注意力放在基本概念和基本知识的把握方面，从而为以后各个章节的学习打下扎实的基础。

读者在刚学习本章时有些概念可能比较抽象，不太容易理解，可以通过具体实例尽可能地把握其核心思想，在以后各个章节的学习中，更深入地理解这些概念并真正掌握它们。

本章的内容较多，为了使读者在学习的过程中具有更好的针对性，对所涉及的知识点进行如下的分类：

① 需要了解的：了解数据管理技术的产生和发展过程、数据库系统的优点、层次数据模型及网状数据模型的基本概念、数据库系统的组成、数据库管理员（DBA）的职责等。

这部分内容有的是知识性的。例如，数据管理技术的产生和发展的历史过程。读者了解数据库技术发展的脉络将有助于了解数据库系统的优点，了解数据库系统和文件系统的区别。

这部分内容有的是技术性和概念性的。例如，层次数据模型及网状数据模型的基本概念。由于当前最常用的是关系数据库系统，《概论》的重点也就放在关系数据库系统技术的讨论上面。我们把层次和网状数据库的内容加以精简和压缩后放在第1章介绍。

这两类系统虽然有它们的缺点，但是执行效率高是它们的显著优点。

② 需要牢固掌握的：掌握概念模型的基本概念、关系数据模型的相关概念、数据库系统三级模式和两级映像的体系结构、数据库系统的逻辑独立性和物理独立性等。

③ 难点:本章的难点是读者在短时间内学习了数据库领域大量的基本概念,有些概念对于刚刚学习数据库的读者来说会感到比较抽象,不容易理解。但不要紧,随着学习的逐渐推进,在后续章节中,这些抽象的概念会变得清晰和具体起来。

此外,数据模型及数据库系统三级模式和两级映像的体系结构也是本章的难点。

下面给出《概论》中一些习题的参考答案,同时对一些问题进行了解析,以便帮助读者理解有关的概念和技术。

1.2 习题解答和解析

1. 试述数据、数据库、数据库管理系统、数据库系统的概念。

答:

有关的概念请阅读《概论》第 4~6 页,下面对概念做一些解析。

(1) 关于数据的解析

在现代计算机系统中数据的概念是广义的。早期的计算机系统主要用于科学计算,处理的数据是整数、实数、浮点数等传统数学中的数据等。现在计算机能存储和处理的对象越来越广泛,表示这些对象的数据也越来越复杂。

数据与其语义是不可分的,这点很重要。例如,500 这个数字可以表示某物品的价格是 500 元,还可以表示一袋奶粉重 500 克,等等。

数据库技术是管理数据的技术。数据是数据库管理的基本对象。因此应该首先要知道什么是数据,知道数据有多种形式。

(2) 关于数据库概念的解析

① 简单地讲,数据库的数据具有永久储存、有组织和可共享三个基本特点。

② 数据模型是数据库的核心概念。每个数据库中的数据都是按照某一种数据模型来组织、描述和存储的。

(3) 关于数据库管理系统(DBMS)的概念解析

DBMS 是一个大型复杂的软件系统,是计算机中的基础软件。目前,专门研制 DBMS 的厂商及其研制的 DBMS 产品很多。著名的有美国 IBM 公司的 DB2 关系数据库管理系统、IMS 层次数据库管理系统;美国 Oracle 公司的 Oracle 关系数据库管理系统;美国微软公司的 SQL Server 关系数据库管理系统等。

主要的国产 DBMS 有北京人大金仓信息技术股份有限公司的金仓数据库管理系统 KingbaseES;武汉达梦公司的达梦数据库管理系统;神通公司的 Oscar 数据库管理系统等。

(4) 关于数据库系统概念的解析

数据库系统是一个人-机系统,数据库是数据库系统的一个组成部分。数据库系统和数据库是两个概念,但在日常工作中人们常常把数据库系统简称为数据库。希望读者能够从人

们讲话或文章的上下文中区分“数据库系统”和“数据库”,不要引起混淆。

2. 使用数据库系统有什么好处?

解析:使用数据库系统的好处很多,既便于数据的集中管理,控制数据冗余,提高数据的利用率和一致性,又有利于应用程序的开发和维护。例如:

① 使用数据库系统可以大大提高应用开发的效率。因为在数据库系统中应用程序不必考虑数据的定义、存储和数据存取的具体路径,这些工作都由 DBMS 来完成。打一个通俗的比喻,使用了 DBMS 就如有了一个好参谋、好助手,许多具体的技术工作都由这个助手来完成。开发人员就可以专注于用户需求的理解、应用逻辑的设计,而不必为管理数据的许多复杂细节操心。

② 当数据的逻辑结构需要改变时,由于数据库系统提供了数据与程序之间的独立性,而数据逻辑结构的改变是 DBA 的责任,开发人员不必修改应用程序,或者只需要修改很少的应用程序,从而既简化了应用程序的编制,又大大减少了应用程序的维护和修改。

③ 使用数据库系统可以减轻 DBA 维护系统的负担。因为 DBMS 在数据库建立、运行和维护时对数据库进行统一的管理和控制,包括数据的完整性和安全性控制,多用户并发控制,故障恢复等都由 DBMS 执行。

读者可以在自己今后的工作中结合具体应用,认真加以体会和总结。

为什么有这些好处,可以结合第 5 题数据库系统的特点来回答。

3. 试述文件系统与数据库系统的区别和联系。

解析:文件系统与数据库系统的区别:

① 文件系统的数据是面向某一应用的,文件的共享性差、冗余度大,独立性差,文件的记录虽然是有结构的,但是整体无结构。所谓“数据面向某个应用”是指数据结构是针对某个应用设计的,只被这个应用程序或应用系统使用,数据是某个应用的“私有资源”。

② 数据库系统中的数据不再仅仅面向某一个应用,而是面向整个组织或企业。数据的共享性高、冗余度小,具有高度的物理独立性和一定的逻辑独立性,数据库中的数据用数据模型组织和描述,由数据库管理系统提供数据安全性、完整性、并发控制和恢复能力。

读者可以参考《概论》表 1.1 中的有关内容。

文件系统与数据库系统的联系:

① 文件系统与数据库系统都是计算机系统中管理数据的软件。

② 文件系统是操作系统的重要组成部分,而 DBMS 是独立于操作系统的软件。我们不能独立购买一个文件系统,但一般需要独立购买 DBMS 软件产品。DBMS 是在操作系统的基础上实现的,数据库中数据的组织和存储是通过操作系统中文件系统来实现的。因此,DBMS 的实现与操作系统中的文件系统是紧密相关的。数据库实现的基础是文件,对数据库的任何操作最终要转化为对文件的操作。所以在 DBMS 实现中数据库物理组织的基本问题是如何利用或如何选择操作系统提供的基本的文件组织方法。这里就不具体展开了。读者

可以参考《概论》第 12 章数据库管理系统中的有关内容。第 12 章是 DBMS 实现技术的概述,有兴趣的读者还可以进一步学习有关课程。

4. 举出适合用文件系统而不是数据库系统的例子;再举出适合用数据库系统的应用例子。

解析:读者可以根据自己所使用的或了解到的实际应用来回答。例如:

① 目前,许多手机上的小型应用都把数据存放在手机操作系统的文件中,如照片、短信和微信等数据。一般来说,功能比较简单、比较固定的应用系统适合用文件系统。

② 目前,几乎所有企业或部门的信息系统都以数据库系统为基础,都使用数据库系统。例如,一个工厂的信息系统会包括多个子系统,如库存管理系统、物资采购系统、作业调度系统、设备管理系统和人事管理系统等;再如学校的学生管理系统、人事管理系统、图书管理系统等都适合用数据库系统。因此,数据库系统已经成为信息系统的基础和核心。

5. 试述数据库系统的特点。

解析:数据库系统主要有以下 4 方面的特点。

① 数据结构化。数据库系统实现整体数据的结构化,这是数据库系统与文件系统的本质区别。

注意这里“整体”两个字。在数据库系统中,数据不再仅仅针对某一个应用,而是面向全组织,可以支持许多应用;不仅数据内部是结构化的,而且数据之间是具有联系的,整体是结构化的。

② 数据的共享性高,冗余度低,易扩充。数据库的数据可以被多个用户、多个应用,用多种不同的程序设计语言共享使用,而且容易增加新的应用,这就使得数据库系统易于扩充,称之为“弹性大”。数据共享可以大大减少数据冗余,节约存储空间,同时还能够避免数据之间的不相容性与不一致性。

所谓“弹性大”是指应用系统容易扩充也容易收缩,即应用增加或减少时不必修改整个数据库的结构,或者只要做很少的修改。我们可以取整体数据的各种子集用于不同的应用系统,当应用需求改变或增加时,只要重新选取不同的子集或加上一部分数据便可以满足新的需求。

③ 数据独立性高。数据独立性包括数据的物理独立性和数据的逻辑独立性。

所谓“独立性”即相互不依赖。数据独立性是指数据和程序相互不依赖。即数据的逻辑结构或物理结构改变了,程序不会跟着改变。数据与程序的独立把数据的定义从程序中分离出去,加上数据的存取又由 DBMS 负责,简化了应用程序的编制,大大减少了应用程序的维护和修改。

④ 数据由 DBMS 统一管理和控制。数据库的共享是并发的共享,即多个用户可以同时存取数据,库中的数据甚至可以同时存取数据库中同一个数据。为此,DBMS 必须提供统一的数据控制功能,包括:

- 数据的安全性保护：保护数据以防止不合法的使用造成的数据泄密和破坏；
- 数据的完整性检查：将数据控制在有效的范围内或保证数据之间满足一定的关系，一定的约束条件；
- 并发控制：对多用户的并发操作加以控制和协调，保证并发操作的正确性；
- 数据库恢复：当计算机系统发生硬件故障、软件故障，或者由于操作员的失误以及故意的破坏影响数据库中数据的正确性，甚至造成数据库部分或全部数据的丢失时，能将数据库从错误状态恢复到某一已知的正确状态，亦称为完整状态或一致状态。

数据库系统的出现，使信息系统从以加工数据的程序为中心转向围绕共享数据库为中心的新阶段。

6. DBMS 的主要功能有哪些？

答：

① 数据库定义功能；

② 数据组织、存储和管理功能；

③ 数据操纵功能；

④ 数据库的事务管理和运行管理；

⑤ 数据库的建立和维护功能；

⑥ 其他功能，如不同数据库之间的互访和互操作功能等。

7. 什么是概念模型？试述概念模型的作用。

答：

概念模型是现实世界到机器世界的一个中间层次。概念模型用于信息世界的建模，是数据库设计人员进行数据库设计的有力工具，也是数据库设计人员和用户之间进行交流的语言。

参考《概论》第 15~16 页。

8. 定义并解释概念模型中术语：实体、实体型、实体集、实体之间的联系。

答：

实体：客观存在并可以相互区分的事物叫实体。

实体型：用实体名及其属性名集合来抽象和刻画同类实体称为实体型。

实体集：同型实体的集合称为实体集。

实体之间的联系：包括实体(型)内部的联系和实体(型)之间的联系。实体内部的联系通常是指组成实体的各属性之间的联系；实体之间的联系通常是指不同实体集之间的联系。实体之间的联系有一对一、一对多和多对多等多种类型。

9. 试述数据模型的概念、数据模型的作用和数据模型的三个要素。

答：

数据模型是数据库系统中最重要的概念之一。数据模型是数据库中用来对现实世界进行抽象的工具，是数据库中用于提供信息表示和操作手段的形式构架。

数据模型是数据库系统的基础。任何一个 DBMS 都以某一个数据模型为基础，或者说支持某一个数据模型。

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。

① 数据结构：描述数据库的组成对象和对象之间的联系，是对系统的静态特性的描述。

② 数据操作：是指对数据库中各种对象（型）的实例（值）允许进行的操作的集合，包括操作及有关的操作规则，是对系统动态特性的描述。

③ 数据的约束条件：是完整性规则的集合，完整性规则是给定的数据模型中数据及其联系所具有的制约和依存规则，用以限定符合数据模型的数据库状态以及状态的变化，以保证数据的正确、有效和相容。

解析：数据库系统中模型有不同的层次。根据模型应用的不同目的，可以将模型分成两类或者说两个层次：一是概念模型，是按用户的观点来对数据和信息建模，用于信息世界的建模，强调语义表达能力，概念简单清晰；二是数据模型，是按计算机系统的观点对数据建模，用于机器世界，人们可以用它定义和操纵数据库中的数据。

10. 试述层次模型的概念，举出三个层次模型的实例。

答：

满足下面两个条件的基本层次联系的集合为层次模型。

① 有且只有一个结点没有双亲结点，这个结点称为根结点；

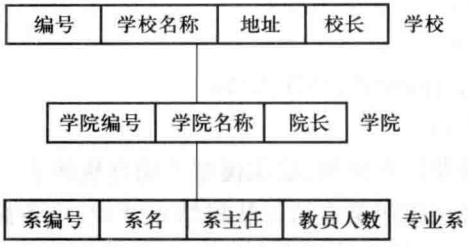
② 根以外的其他结点有且只有一个双亲结点。

层次模型的实例：

① 行政机构层次数据库模型：

② 行政区域层次数据库模型（记录的字段从略）：

③ 学校层次数据库模型：



11. 试述网状模型的概念，举出三个网状模型的实例。

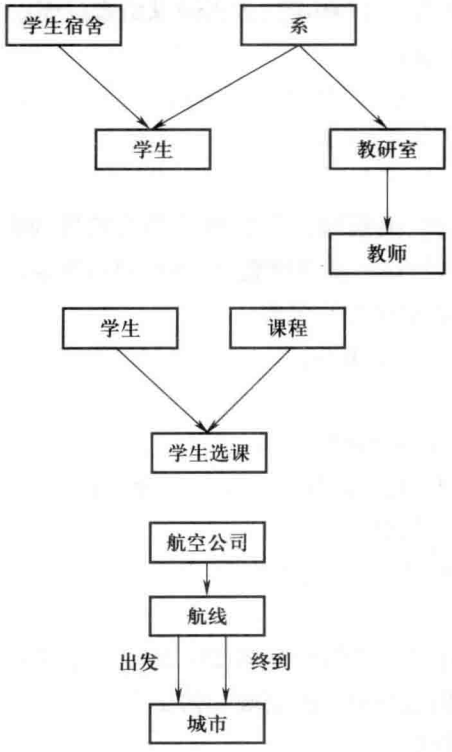
答：

满足下面两个条件的基本层次联系的集合为网状模型。

① 允许一个以上的结点无双亲；

② 一个结点可以有多于一个的双亲。

网状模型的实例：



12. 试述层次数据库、网状数据库的优缺点。

答：

层次数据库的优点主要有：

① 层次模型的数据结构比较简单清晰；

② 层次数据库的查询效率高；

③ 层次数据模型提供了良好的完整性支持。

层次数据库的缺点主要是：

现实世界中很多联系是非层次性的，层次模型不能自然地表示这类联系；

层次数据库中的查询必须按照层次结构从根结点开始，沿着路径进行。因此，用户必须清楚所用数据库的层次结构，对用户的要求自然比较高了。

网状数据库的优点主要有：

① 能够更为直接地描述现实世界，如一个结点可以有多个双亲。

② 具有良好的性能，存取效率较高。

网状数据库的缺点主要是：

① 结构比较复杂，而且随着应用环境的扩大，数据库的结构就变得越来越复杂，不利于最终用户掌握。

② 网状数据库的数据定义语言(DDL)、数据操纵语言(DML)比较复杂，要求用户掌握数据库结构和存取路径，不容易使用。

13. 试述关系模型的概念、定义并解释以下术语：

关系，属性，域，元组，码，分量，关系模式

答：

关系模型由关系数据结构、关系操作集合和关系完整性约束三部分组成。在用户观点下，关系模型中数据的逻辑结构是一张二维表，它由行和列组成。

① 关系：一个关系对应通常说的一张表。

② 属性：表中的一列即为一个属性；

③ 域：属性的取值范围；

④ 元组：表中的一行即为一个元组；

⑤ 码：表中的某个属性组，它可以唯一确定一个元组；

⑥ 分量：元组中的一个属性值；

⑦ 关系模式：对关系的描述，一般表示为

关系名(属性 1，属性 2，…，属性 n)。

解析：这些概念的定义可以参考《概论》第 25~26 页中有关这些概念的初步解释和定义。《概论》第 2 章、第 3 章会给出这些概念的更确切的定义。

14. 试述关系数据库的特点。

答：

关系数据库是建立在关系数据模型上的，具有下列优点：

① 关系模型与非关系模型不同，它具有严格的数学基础。

② 关系模型的概念单一。所以其数据结构简单、清晰，用户易懂易用。

③ 关系模型的存取路径对用户透明，从而具有更高的数据独立性、更好的安全保密性，也简化了程序员的工作和数据库开发建立的工作。

当然，关系数据模型也有缺点，其中最主要的是，由于存取路径对用户透明，查询效率往往不如非关系数据模型。因此为了提高性能，必须对用户的查询请求进行优化，这就增加了开发关系数据库管理系统软件的难度。

15. 试述数据库系统三级模式结构，并说明这种结构的优点是什么？

答：

数据库系统的三级模式结构由外模式、模式和内模式组成（参考《概论》图 1.16）。

外模式，亦称子模式或用户模式，是数据库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。

模式，亦称逻辑模式，是数据库中全体数据的逻辑结构和特性的描述，是所有用户的公共数据视图。模式描述的是数据的全局逻辑结构。外模式通常是模式的子集。

内模式，亦称存储模式，是数据在数据库系统内部的表示，即对数据的物理结构和存储方式的描述。

为了能够在内部实现这三个抽象层次的联系和转换，数据库系统在这三级模式之间提供了两级映像：外模式/模式映像和模式/内模式映像。正是这两级映像保证了数据库系统中的数据能够具有较高的逻辑独立性和物理独立性。

16. 定义并解释术语：

模式，外模式，内模式，数据定义语言，数据操纵语言

答：

模式、外模式、内模式的解释参见第 15 题。

数据定义语言：用来定义数据库模式、外模式和内模式的语言。

数据操纵语言：用来对数据库中的数据进行查询、插入、删除和修改的语句。

17. 什么叫数据与程序的物理独立性？什么叫数据与程序的逻辑独立性？为什么数据库系统具有数据与程序的独立性？

答：

数据与程序的物理独立性：当数据库的存储结构改变了，由数据库管理员对模式/内模式映像作相应改变，可以使模式保持不变，从而应用程序也不必改变，这就是数据与程序的物理独立性，简称数据的物理独立性。

数据与程序的逻辑独立性：当数据的逻辑结构即模式改变时，由数据库管理员对各个外模式/模式的映像作相应改变，可以使外模式保持不变，从而应用程序不必修改，这就是数据与程序的逻辑独立性，简称数据的逻辑独立性。

DBMS 在三级模式之间提供的两级映像保证了数据库系统中的数据能够具有较高的逻辑独立性和物理独立性。

18. 试述数据库系统的组成。

答：

数据库系统一般由数据库、数据库管理系统(及其开发工具)、应用系统、数据库管理员和用户构成。详细内容参考《概论》第 31~34 页。

19. 试述数据库管理员、系统分析员、数据库设计人员、应用程序员的职责。

答：

数据库管理员全面负责管理和控制数据库系统。具体职责包括：

① 决定数据库的信息内容和结构；

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 监控数据库的使用和运行；

⑤ 数据库系统的改进和重组重构。

系统分析员负责应用系统的需求分析和规范说明，他要和用户及 DBA 相结合，确定系统的硬件软件配置，并参与数据库系统的概要设计。

数据库设计人员负责数据库中数据的确定和数据库各级模式的设计。数据库设计人员必须参加用户需求调查和系统分析，然后进行数据库设计。

应用程序员负责设计和编写应用系统的程序模块，并进行调试和安装。

1.3 补充习题

1. 选择题

(1) 数据库系统的核心和基础是(　　)。

A. 物理模型　　B. 概念模型　　C. 数据模型　　D. 逻辑模型

(2) 实现将现实世界抽象为信息世界的是(　　)。

A. 物理模型　　B. 概念模型　　C. 关系模型　　D. 逻辑模型

(3) 数据管理技术经历了若干阶段，其中人工管理阶段和文件系统阶段相比文件系统的一个显著优势是(　　)。

A. 数据可以长期保存　　B. 数据共享性很强

C. 数据独立性很好　　D. 数据整体结构化

(4) 能够保证数据库系统中的数据具有较高的逻辑独立性的是(　　)。

A. 外模式/模式映像　　B. 模式

C. 模式/内模式映像　　D. 外模式

(5) IBM 公司的 IMS 数据库管理系统采用的数据模型是(　　)。

A. 层次模型　　B. 网状模型　　C. 关系模型　　D. 面向对象模型

(6) DBMS 是一类系统软件,它是建立在下列哪种系统之上的?(　　)

A. 应用系统　　B. 编译系统　　C. 操作系统　　D. 硬件系统

(7) 关于网状数据库,以下说法正确的是(　　)。

A. 只有一个结点可以无双亲　　B. 一个结点可以有多于一个的双亲

C. 两个结点之间只能有一种联系　　D. 每个结点有且只有一个双亲

(8) 下列说法中,正确的是(　　)。

A. 数据库的概念模型与具体的 DBMS 有关

B. 三级模式中描述全体数据的逻辑结构和特征的是外模式

C. 数据库管理员负责设计和编写应用系统的程序模块

D. 从逻辑模型到物理模型的转换一般是由 DBMS 完成的

(9) 长期存储在计算机内,有组织的、可共享的大量数据的集合是(　　)。

A. 数据(Data)　　B. 数据库(DataBase)

C. 数据库管理系统(DBMS)　　D. 数据库系统(DBS)

(10) 在数据管理技术发展过程中,需要应用程序管理数据的是(　　)。

A. 人工管理阶段　　B. 人工管理阶段和文件系统阶段

C. 文件系统阶段和数据库系统阶段　　D. 数据库系统阶段

2. 判断题

(1) 在文件系统管理阶段,由文件系统提供数据存取方法,所以数据已经达到很强的独立性。(　　)

(2) 通常情况下,外模式是模式的子集。(　　)

(3) 数据库管理系统是指在计算机系统中引入数据库后的系统,一般由 DB、DBS、应用系统和 DBA 组成。(　　)

(4) 在数据模型的组成要素中,数据结构是刻画一个数据模型性质最重要的方面,人们通常按照数据结构的类型来命名数据模型。(　　)

(5) 数据库系统的三级模式是对数据进行抽象的 3 个级别,把数据的具体组织留给 DBMS 管理。(　　)

(6) 层次模型是比网状模型更具普遍性的结构,网状模型是层次模型的一个特例。(　　)

3. 填空题

(1) 数据库系统的逻辑模型按照计算机的观点对数据建模,主要包括_____、_____、_____、面向对象模型、对象关系模型和半结构化数据模型等。

(2) 最经常使用的概念模型是_____。

(3) 数据独立性是数据库领域的重要概念,包括数据的_____独立性和数据的_____独立性。

(4) 数据库系统的三级模式结构是指数据库系统是由_____、_____和_____三级构成。

(5) 两个实体型之间的联系可以分为三种:一对一联系、_____和_____。

(6) 数据库管理系统提供的数据控制方面的功能包括数据的_____保护、数据的_____检查、_____和数据库恢复。

(7) 数据库的三级模式结构中,描述局部数据的逻辑结构和特征的是_____。

(8) 层次模型和网状模型中的单位是基本层次联系,这是指两个_____以及它们之间的_____(包括一对一)的联系。

(9) 数据模型的组成要素中描述系统的静态特性和动态特性的分别是_____和_____。

4. 问答题

*(1) 试述数据管理的文件系统阶段和数据库系统阶段"数据独立性"有何不同。

(2) 叙述使用文件系统管理数据的缺点。

1.4 补充习题答案

1. 选择题

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
C	B	A	A	A	C	B	D	B	A

2. 判断题

(1)	(2)	(3)	(4)	(5)	(6)
×	√	×	√	√	×

3. 填空题

(1) 层次模型 网状模型 关系模型

(2) E-R 模型

(3) 物理 逻辑

(4) 外模式 模式 内模式

(5) 一对多联系 多对多联系

(6) 安全性 完整性 并发控制

(7) 外模式

(8) 记录(型) 一对多

(9) 数据结构 数据操作

4. 问答题

*(1) 试述数据管理的文件系统阶段和数据库系统阶段“数据独立性”有何不同。

答:

文件系统中数据被组织成相互独立的数据文件,程序按照文件名访问数据,“数据独立性”是一种“设备独立性”。

数据库系统中的“数据独立性”包括“物理独立性”和“逻辑独立性”,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;逻辑独立性是指用户的应用程序与数据库的逻辑结构是相互独立的。

(2) 叙述使用文件系统管理数据的缺点。

答:

① 数据共享性差,冗余度大;

② 数据独立性差。

第 2 章　关系数据库

本章系统地讲解了关系数据库的重要概念，并着重对关系模型进行了阐述。

关系模型包括关系数据结构、关系操作集合以及关系完整性约束三个组成部分。本章分别对这三个部分的内容进行了详细的分析与论述。

2.1　基本知识点

关系模型和关系数据库是《概论》一书的重点，在全书中占有较大的篇幅。因此，掌握本章的关键内容是学习后续各章的基础。

① 需要了解的：了解关系数据库理论产生和发展的过程；了解关系数据库产品的发展及沿革；了解关系演算的概念，这部分内容不包括在本科教学大纲内。

② 需要牢固掌握的：掌握关系模型的三个组成部分及各部分所包括的主要内容；牢固掌握关系数据结构及其形式化定义；关系的三类完整性约束的概念。

③ 需要举一反三的：关系代数，关系代数中的各种运算，包括并、交、差、选择、投影、连接、除及广义笛卡儿积等。

④ 难点：本章的难点在于关系代数。由于关系代数较为抽象，因此在学习的过程中一定要结合具体的实例进行练习。

2.2　习题解答和解析

1. 试述关系模型的三个组成部分。

答：关系模型由关系数据结构、关系操作集合和关系完整性约束三部分组成。

2. 试述关系数据语言的特点和分类。

答：关系数据语言可以分为三类：关系代数语言、关系演算语言以及具有关系代数和关系演算双重特点的语言。

3. 定义并理解下列术语，说明它们之间的联系与区别：

(1) 域,笛卡儿积,关系,元组,属性

答:

域:域是一组具有相同数据类型的值的集合。

笛卡儿积:给定一组域 $D_1, D_2, \cdots, D_n$,允许其中某些域是有相同的。这组域的笛卡儿积为

$$D_1 \times D_2 \times \cdots \times D_n = \{(d_1, d_2, \cdots, d_n) \mid d_i \in D_i, i = 1, 2, \cdots, n\}$$

关系:在域 $D_1, D_2, \cdots, D_n$ 上笛卡儿积 $D_1 \times D_2 \times \cdots \times D_n$ 的子集称为关系,表示为

$$R(D_1, D_2, \cdots, D_n)。$$

注意,这里是用较为形式化的方法来定义关系。在第 1 章中则是用通俗的语言来说明什么是关系,是一种不严格的定义。

元组:关系中的每个元素是关系中的元组。

属性: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,必须对每列起一个名字,称为属性(attribute)。

(2) 候选码,主码,外码

答:

候选码:若关系中的某一属性组的值能唯一地标识一个元组,而其子集不能,则称该属性组为候选码(candidate key)。

主码:若一个关系有多个候选码,则选定其中一个为主码(primary key)。

外码:设 F 是基本关系 R 的一个或一组属性,但不是关系 R 的码,如果 F 与基本关系 S 的主码 K_s 相对应,则称 F 是基本关系 R 的外部码(foreign key),简称外码。

(3) 关系模式,关系,关系数据库

关系模式:关系的描述称为关系模式(relation schema)。它可以形式化地表示为

$$R(U, D, \mathrm{DOM}, F)$$

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:见(1),关系是关系模式在某一时刻的状态或内容。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。

关系数据库:关系数据库也有型和值之分。关系数据库的型称为关系数据库模式,是对关系数据库的描述,它包括若干域的定义以及在这些域上定义的若干关系模式。关系数据库的值是这些关系模式在某一时刻对应的关系的集合,通常就称为关系数据库。

4. 举例说明关系模式和关系的区别。

答:

关系模式是型;关系是值,是关系模式的实例。例如:

Student(Sno,Sname,Sage)是关系模式,下面的表是关系,即某一时刻关系模式的值。

Sno	Sname	Sage
S_1	张俊丽	18
S_2	李红钰	19
S_3	王敏英	19

5. 试述关系模型的完整性规则。在参照完整性中，什么情况下外码属性的值可以为空值？

答：

关系模型中可以有三类完整性约束：实体完整性、参照完整性和用户定义的完整性。关系模型的完整性规则是对关系的某种约束条件。

① 实体完整性规则：若属性 A 是基本关系 R 的主属性，则属性 A 不能取空值。

② 参照完整性规则：若属性（或属性组）F 是基本关系 R 的外码，它与基本关系 S 的主码 K_s 相对应（基本关系 R 和 S 不一定是不同的关系），则对于 R 中每个元组在 F 上的值必须为下面二者之一：

- 或者取空值（F 的每个属性值均为空值）；
- 或者等于 S 中某个元组的主码值。

③ 用户定义的完整性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。

在参照完整性中，如果外码属性不是其所在关系的主属性，外码属性的值可以取空值。

例如，在下面的“学生”表中，“专业号”是一个外码，它不是学生表的主属性，可以为空。其语义是，该学生的专业尚未确定。

学生（学号，姓名，性别，专业号，年龄）

专业（专业号，专业名）

而在下面的“选修”表中的“课程号”虽然也是一个外码属性，但它又是“选修”表的主属性，选修表必须满足实体完整性，所以其主属性“课程号”不能为空。

课程（课程号，课程名，学分）

选修（学号，课程号，成绩）

6. 设有一个 SPJ 数据库，包括 S、P、J、SPJ 4 个关系模式：

S(SNO,SNAME,STATUS,CITY)；

P(PNO,PNAME,COLOR,WEIGHT)；

J(JNO,JNAME,CITY)；

SPJ(SNO,PNO,JNO,QTY)；

供应商表 S 由供应商代码（SNO）、供应商姓名（SNAME）、供应商状态（STATUS）、供应商所在城市（CITY）组成。

零件表 P 由零件代码（PNO）、零件名（PNAME）、颜色（COLOR）、重量（WEIGHT）组成。

工程项目表 J 由工程项目代码(JNO)、工程项目名(JNAME)、工程项目所在城市(CITY)组成。

供应情况表 SPJ 由供应商代码(SNO)、零件代码(PNO)、工程项目代码(JNO)、供应数量(QTY)组成,表示某供应商供应某种零件给某工程项目的数量为 QTY。

今有若干数据如下:

S 表

SNO	SNAME	STATUS	CITY
S1	精益	20	天津
S2	盛锡	10	北京
S3	东方红	30	北京
S4	丰泰盛	20	天津
S5	为民	30	上海

P 表

PNO	PNAME	COLOR	WEIGHT
P1	螺母	红	12
P2	螺栓	绿	17
P3	螺丝刀	蓝	14
P4	螺丝刀	红	14
P5	凸轮	蓝	40
P6	齿轮	红	30

J 表

JNO	JNAME	CITY
J1	三建	北京
J2	一汽	长春
J3	弹簧厂	天津
J4	造船厂	天津
J5	机车厂	唐山
J6	无线电厂	常州
J7	半导体厂	南京

SPJ 表

SNO	PNO	JNO	QTY
S1	P1	J1	200
S1	P1	J3	100
S1	P1	J4	700
S1	P2	J2	100
S2	P3	J1	400
S2	P3	J2	200
S2	P3	J4	500
S2	P3	J5	400
S2	P5	J1	400
S2	P5	J2	100
S3	P1	J1	200
S3	P3	J1	200
S4	P5	J1	100
S4	P6	J3	300
S4	P6	J4	200
S5	P2	J4	100
S5	P3	J1	200
S5	P6	J2	200
S5	P6	J4	50

试用关系代数、ALPHA 语言、QBE 语言完成下列操作：

(1) 求供应工程 J1 零件的供应商号 SNO。

答：

关系代数：$\prod_{SNO}(\sigma_{JNO='J1'}(SPJ))$

ALPHA 语言：GET W(SPJ.SNO):SPJ.JNO='J1'

QBE 语言：

SPJ	SNO	PNO	JNO	QTY
	P.S1		J1	

(2) 求供应工程 J1 零件 P1 的供应商号 SNO。

答：

关系代数：$\prod_{SNO}(\sigma_{JNO='J1'\wedge PNO='P1'}(SPJ))$

ALPHA 语言：GET W(SPJ.SNO):SPJ.JNO='J1'∧SPJ.PNO='P1'

QBE 语言：

SPJ	SNO	PNO	JNO	QTY
	P.S1	P1	J1	

(3) 求供应工程 J1 红色零件的供应商号 SNO。

答：

关系代数：$\prod_{SNO}(\prod_{SNO,PNO}(\sigma_{JNO='J1'}(SPJ))\bowtie\prod_{PNO}(\sigma_{COLOR='红'}(P)))$

ALPHA 语言：

```
RANGE P PX
    GET W(SPJ.SNO):SPJ.JNO='J1'∧∃PX(PX.COLOR='红'∧PX.PNO=SPJ.PNO)
```

QBE 语言：

SPJ	SNO	PNO	JNO	QTY
	P.S1	P1	J1	

P	PNO	PNAME	COLOR	WEIGHT
	P1		红	

(4) 求没有使用天津供应商生产的红色零件的工程号 JNO。

答：

关系代数：$\prod_{JNO}(J)-\prod_{JNO}(\prod_{SNO}(\sigma_{CITY='天津'}(S))\bowtie\prod_{SNO,PNO,JNO}(SPJ)$

$$\bowtie \prod_{PNO}(\sigma_{COLOR='红'}(P)))$$

解析：

减法运算中，被减的部分是使用了天津供应商生产的红色零件的所有工程号，$\prod_{JNO}(J)$是全部工程的工程号，两者相减就是没有使用天津供应商生产的红色零件的工程号，包括没有使用任何零件的工程号。

ALPHA 语言：

```
RANGE SPJ SPJX
      P PX
      S SX
GET W(J.JNO):¬∃SPJX(SPJX.JNO=J.JNO ∧
                ∃SX(SX.SNO=SPJX.SNO ∧SX.CITY='天津'∧
                ∃PX(PX.PNO=SPJX.PNO ∧PX.COLOR='红'))
```

解析：

① S、P、SPJ 表上各设了一个元组变量。

② 解题思路是，因为要找的是满足给定条件的工程号 JNO，因此，对工程表 J 中的每一个 JNO 进行判断：

a. 看 SPJ 中是否存在这样的元组，其 JNO = J.JNO，并且所用的零件是红色的，该零件的供应商是天津的。

b. 如果 SPJ 中不存在这样的元组，则该工程号 JNO 满足条件，放入结果集合中。

c. 如果 SPJ 中存在这样的元组，则该工程号 JNO 不满足条件，不放入结果集中。再对工程表 J 中的下一个 JNO 进行同样的判断。

d. 直到所有 JNO 都检查完。

e. 结果集中是所有没有使用天津供应商生产的红色零件的工程号，包括没有使用任何零件的工程号。

QBE 语言：

当不考虑没有使用任何零件的工程时：

S	SNO	SNAME	STATUS	CITY
	S1			天津

P	PNO	PNAME	COLOR	WEIGHT
	P1		红	

SPJ	SNO	PNO	JNO	QTY
¬	S1	P1	P.J1	

解析：

本题是从 SPJ 表中输出满足条件的 JNO，没有使用任何零件的工程项目的工程号是不会出现在 SPJ 中的。所以本题的结果不包括没有使用任何零件的工程项目号。

考虑没有使用任何零件的工程：

J	JNO	JNAME	CITY
¬	P.J1		

S	SNO	SNAME	STATUS	CITY
	S1			天津

P	PNO	PNAME	COLOR	WEIGHT
	P1		红	

SPJ	SNO	PNO	JNO	QTY
	S1	P1	J1	

解析：

本题是从 J 表中输出满足条件的 JNO，未使用任何零件的工程项目的工程号也满足条件。所以本题的结果包括未使用任何零件的工程号。

(5) 求至少用了 S1 供应商所供应的全部零件的工程号 JNO。

答：

关系代数：$\prod_{JNO,PNO}(SPJ) \div \prod_{PNO}(\sigma_{SNO='S1'}(SPJ))$

解析：

第一部分是所有工程与该工程所用的零件，第二部分是 S1 所供应的全部零件号，对于 SPJ 表中的某一个 JNO，如果该工程使用的所有零件的集合包含了 S1 所供应的全部零件号，则该 JNO 符合本题条件，在除法运算的结果集中。

可以看到，使用关系代数的除法运算概念清晰，语言表达也很简洁。

ALPHA 语言：(类似于《概论》[例 2.27])

```
RANGE SPJ SPJX
      SPJ SPJY
      P PX
```

GET W(J.JNO):∀PX(∃SPJX(SPJX.PNO=PX.PNO ∧SPJX.SNO='S1')
=>∃SPJY(SPJY.JNO=J.JNO ∧SPJY.PNO=PX.PNO))

解析:

① SPJ 表上设了两个元组变量 SPJX、SPJY;P 表上设了一个元组变量 PX。

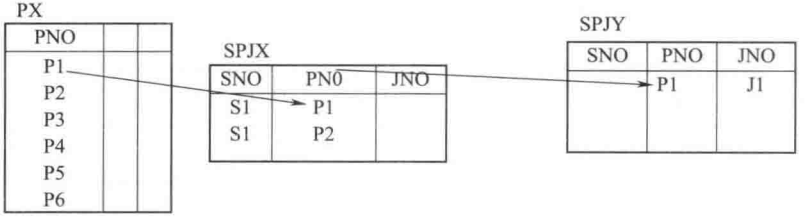
② 解题思路是,由于要找的是满足给定条件的工程号 JNO,因此,对工程表 J 中的每一个 JNO(例如 J1),进行以下一组操作:

a. 对零件 *PX* 中的所有零件,依次对每一个零件进行以下检查:

b. 例如零件 P1,检查 SPJX,看 S1 是否供应了该零件,如果供应了,则再看这一个 JNO(例如 J1)是否使用了该零件。

c. 如果对于 S1 所供应的每种零件,这一个 JNO(例如 J1)都使用了,则该 JNO 为(例如 J1)满足要求的工程项目。

③ 为了帮助理解,读者可以画出所涉及的三个表,给出一些数据,按照上面的解析步骤逐步分析,就能掌握解题方法。从而达到举一反三的要求。



QBE:(不要求)。

7. 试述等值连接与自然连接的区别和联系。

答:

它们是连接运算中两种最为重要,也最为常用的连接。

自然连接是一种特殊的等值连接,它要求两个关系中进行比较的分量,即连接属性必须是相同的属性组,并且要在结果中去掉其中一个的重复属性。

8. 代数的基本运算有哪些? 如何用这些基本运算来表示其他运算?

答:

在 8 种关系代数运算中,并、差、笛卡儿积、投影和选择 5 种运算为基本运算;其他三种运算,即交、连接和除,均可以用这 5 种基本运算来表达。

交运算:$R \cap S = R-(R-S)$

连接运算:$R \underset{A\theta B}{\bowtie} S = \sigma_{A\theta B}(R \times S)$

除运算:$R(X,Y) \div S(Y,Z) = \prod_X(R) - \prod_X(\prod_X(R) \times \prod_Y(S) - R)$

X、Y、Z 为属性组，R 中的 Y 和 S 中的 Y 可以有不同的属性名，但必须出自相同的域集。

2.3 补充习题

1. 选择题

(1) 关于关系模型，下列叙述不正确的是(　　)。

A. 一个关系至少要有一个候选码　　B. 列的次序可以任意交换

C. 行的次序可以任意交换　　D. 一个列的值可以来自不同的域

(2) 下列说法正确的是(　　)。

A. 候选码都可以唯一地标识一个元组　　B. 候选码中只能包含一个属性

C. 主属性可以取空值　　D. 关系的外码不可以取空值

(3) 关系操作中，操作的对象和结果都是(　　)。

A. 记录　　B. 集合

C. 元组　　D. 列

(4) 假设存在一张职工表，包含“性别”属性，要求这个属性的值只能取“男”或“女”，这属于(　　)。

A. 实体完整性　　B. 参照完整性

C. 用户定义的完整性　　D. 关系不变性

(5) 有两个关系 $R(A,B,C)$ 和 $S(B,C,D)$，将 R 和 S 进行自然连接，得到的结果包含几个列(　　)。

A. 6　　B. 4　　C. 5　　D. 2

2. 判断题

(1) 关系模型的一个特点是，实体以及实体之间的联系都可以使用相同的结构类型来表示。(　　)

(2) 关系模型中，非主属性不可能出现在任何候选码中。(　　)

(3) 在左外连接中，保留的是左边关系中所有的元组。(　　)

(4) 关系模式是对关系的描述，关系是关系模式在某一时刻的状态或内容。(　　)

3. 填空题

(1) 在关系模型中，关系操作包括查询、_____、_____和_____等。

(2) 关系模型的三类完整性约束是指_____、_____和_____。

(3) 关系模型包括 8 种查询操作，其中_____、_____、并、_____和笛卡儿积是 5 种基本操作，其他操作可以用基本操作定义和导出。

(4) 职工(职工号，姓名，年龄，部门号)和部门(部门号，部门名称)存在引用关系，其中_____是参照关系，_____是外码。

4. 问答题

(1) 解释候选码和主码以及它们之间的关系。

(2) 说明什么是关系完备性？关系演算在语言表达能力上是完备的吗？

5. 综合题

(1) 假设有一个数据库包含以下关系模式：

Teacher(Tno,Tname,Tage,Tsex)　/*主码下面加了下划线*/

Department(Dno,Dname,Tno)

Work(Tno,Dno,Year,Salary)

教师表 Teacher 由教师代码(Tno)、教师名字(Tname)、教师年龄(Tage)、教师性别(Tsex)组成。

系表 Department 由系代码(Dno)、系名(Dname)、系主任代码(Tno)组成。

工作表由教师代码(Tno)、系代码(Dno)、入职年份(Year)、工资(Salary)组成。

使用关系代数表示每个查询：

① 列出工资超过 5 000 的教师的不同年龄；

② 查找不在计算机系工作的教师代码；

③ 系主任 T1 管辖范围内的所有教师姓名；

④ 假设对关系 r，$\rho_x(r)$ 表示得到别名为 x 的一个相同的关系，系里的每个教师都有工资，列出比 D1 系的所有教师工资都高的教师代码。

(2) 考虑第(1)题描述的数据库，每个关系包含的元组如下：

Teacher

Tno	Tname	Tage	Tsex
T1	张丽	42	女
T2	李波	45	男
T3	王艳	33	女
T4	赵明	29	男

Department

Dno	Dname	Tno
D1	计算机系	T1
D2	数学系	T2
D3	电子系	NULL

Work

Tno	Dno	Year	Salary
T1	D1	1995	6 000
T2	D2	1992	6 500
T3	D1	2005	4 500

假设符号⟕和⟖分别表示左外连接和右外连接，使用关系代数完成以下查询并给出结果：

① 列出所有教师的姓名以及所在的系名；

② 列出所有系的名称以及包含的教师姓名。

(3) 有两个关系 $S(A,B,C,D)$ 和 $T(C,D,E,F)$，分别包括 N_1、N_2 个元组，$N_2>N_1>0$，对下列每个关系代数表达式，计算在使表达式有意义的情况下，可以得到的最大、最小元组的数目以及列的数目。

① $S\cup T$　② $S\cap T$　③ $S-T$　④ $S\times T$

⑤ $\sigma_{A=10}(S)$　⑥ $\prod_{A,B}(S)$　⑦ $S\bowtie T$　⑧ $\prod_{C,D}(S)\times T$

2.4　补充习题答案

1. 选择题

(1)	(2)	(3)	(4)	(5)
D	A	B	C	B

2. 判断题

(1)	(2)	(3)	(4)
√	√	√	√

3. 填空题

(1) 插入　删除　修改

(2) 实体完整性　参照完整性　用户定义的完整性

(3) 选择　投影　差

(4) 职工　部门号

4. 问答题

(1) 解释候选码和主码的关系。

答：

若关系中的某一属性组的值能唯一标识一个元组，该属性组称为候选码；如果一个关系有多个候选码，其中一个被选为主码。候选码可以有多个，主码是其中的一个。

(2) 说明什么是关系完备性？关系演算在语言表达能力上是完备的吗？

答：

关系完备性是指一个查询语言能够表示关系代数可以表示的所有查询。关系演算具有

完备的表达能力。

5. 综合题

(1)

① $\prod_{Tage}(Teacher \bowtie \sigma_{Salary>5000}(Work))$

② $\prod_{Tno}(Teacher)-\prod_{Tno}(Work \bowtie \prod_{Dno}(\sigma_{Dname='计算机系'}(Department)))$

③ $\prod_{Tname}(Teacher \bowtie \prod_{Tno}(Work \bowtie \sigma_{Tno='T1'}(Department)))$

④ $\prod_{Tno}(Work)-(\prod_{Work.Tno}(Work \bowtie_{Work.Salary \leqslant Work2.Salary \wedge Work2.Dno='D1'} \rho_{Work2}(Work)))$

(2)

① $\prod_{Tname,\ Dname}(Teacher \overline{\bowtie}(Work \bowtie Department))$

Tname	Dname
张丽	计算机系
李波	数学系
王艳	计算机系
赵明	NULL

② $\prod_{Tname,\ Dname}((Teacher \bowtie Work) \overline{\bowtie} Department)$

Tname	Dname
张丽	计算机系
李波	数学系
王艳	计算机系
NULL	电子系

(3)

表达式	最大元组数目	最小元组数目	列的数目
$S \cup T$	N_1+N_2	N_2	4
$S \cap T$	N_1	0	4
$S-T$	N_1	0	4

续表

表达式	最大元组数目	最小元组数目	列的数目
S×T	$N_1 * N_2$	$N_1 * N_2$	8
$\sigma_{A=10}(S)$	N_1	0	4
$\prod_{A,B}(S)$	N_1	1	2
S⋈T	$N_1 * N_2$	0	6
$\prod_{C,D}(S)\times T$	$N_1 * N_2$	N_2	6

第 3 章 关系数据库标准语言 SQL

《概论》第 3 章详细介绍关系数据库语言 SQL。SQL 语言是关系数据库的标准语言。它内容十分丰富，是学习关系数据库概念和技术的重要组成部分。

3.1 基本知识点

关系模型和关系数据库是《概论》的重点，第 3 章又是重点中的重点，是全书篇幅最长的一章，因为关系数据库系统的主要功能是通过 SQL 语言来实现的。

① 需要了解的：SQL 语言的发展过程，从而进一步了解关系数据库技术和关系数据库管理系统（RDBMS）产品的发展过程。

② 需要牢固掌握的：掌握 SQL 语言的特点和优点，体会面向过程的语言和 SQL 语言的区别。体会关系数据库系统为数据库应用系统的开发提供良好环境，减轻了用户负担，提高用户生产率的原因。

③ 需要举一反三的：熟练正确地使用 SQL 语言完成对数据库的查询、插入、删除、更新操作，特别是各种各样的查询，掌握 SQL 语言强大的查询功能。

在完成具体的 SQL 语句时，希望读者能有意识地和关系代数、关系演算等语言进行比较，了解它们各自的特点。

④ 难点：本章的难点在于用 SQL 语言正确完成复杂查询。因此在学习的过程中一定要多练习，要在某一个 RDBMS 产品上进行实际运行，检查自己的答案，检测查询的结果是否正确。只有通过大量练习才能真正达到举一反三的熟练程度。

3.2 习题解答和解析

1. 试述 SQL 的特点。

答：

① 综合统一。SQL 语言集数据定义语言（DDL）、数据操纵语言（DML）和数据控制语言

(DCL)的功能于一体。

② 高度非过程化。用 SQL 语言进行数据操作,只要提出“做什么”,而无须指明“怎么做”,因此无须了解存取路径。存取路径的选择以及 SQL 语句的操作过程由系统自动完成。

③ 面向集合的操作方式。SQL 语言采用集合操作方式,不仅操作对象、查找结果可以是元组的集合,而且一次插入、删除、更新操作的对象也可以是元组的集合。

④ 以同一种语法结构提供两种使用方式。SQL 语言既是自含式语言,又是嵌入式语言。作为自含式语言,它能够独立地用于联机交互的使用方式,也能够嵌入到高级语言程序中,供程序员设计程序时使用。

⑤ 语言简洁,易学易用。

解析:

详细内容可参考《概论》第 3.1.2 小节。注意不要仅仅背这些特点,关键是要通过具体的练习,通过使用 SQL 语句来理解这些特点。

2. 说明在 DROP TABLE 时,RESTRICT 和 CASCADE 的区别。

答:

RESTRICT 表示表的删除是有限制条件的。要删除的基本表不能被其他表的约束所引用,不能有视图,不能有触发器,不能有存储过程或函数等。如果存在这些依赖该表的对象,则表不能被删除。

CASCADE 表示表的删除没有限制条件,在删除基本表的同时,相关的依赖对象(如视图)都将被删除。

3. 有两个关系 $S(A,B,C,D)$ 和 $T(C,D,E,F)$,写出与下列查询等价的 SQL 表达式:

(1) $\sigma_{A=10}(S)$;(2) $\prod_{A,B}(S)$;(3) $S \bowtie T$;(4) $S \underset{S.C=T.C}{\bowtie} T$;(5) $S \underset{A<E}{\bowtie} T$;(6) $\prod_{C,D}(S) \times T$。

答:

(1) SELECT * FROM S WHERE A=10

(2) SELECT DISTINCT A,B FROM S

(3) SELECT A,B,S.C,S.D,E,F FROM S,T WHERE S.C=T.C AND S.D=T.D

(4) SELECT A,B,S.C,S.D,T.C,T.D,E,F FROM S,T WHERE S.C=T.C

(5) SELECT A,B,S.C,S.D,T.C,T.D,E,F FROM S,T WHERE A < E

(6) SELECT S1.C,S1.D,T.C,T.D,E,F FROM T,(select DISTINCT C,D FROM S) AS S1

4. 用 SQL 语句建立第 2 章习题 6 中的 4 个表;针对建立的 4 个表用 SQL 语言完成第 2 章习题 6 中的查询。

答:

建 S 表:

```
S(SNO,SNAME,STATUS,CITY);
CREATE TABLE S
```

```
(SNO CHAR(3),
 SNAME CHAR(10),
 STATUS CHAR(2),
 CITY CHAR(10) );
```

建 P 表：

```
P(PNO,PNAME,COLOR,WEIGHT);
CREATE TABLE P
(PNO CHAR(3),
 PNAME CHAR(10),
 COLOR CHAR(4),
 WEIGHT INT);
```

建 J 表：

```
J(JNO,JNAME,CITY);
CREATE TABLE J
(JNO CHAR(3),
 JNAME CHAR(10),
 CITY CHAR(10) );
```

建 SPJ 表：

```
SPJ(SNO,PNO,JNO,QTY);
CREATE TABLE SPJ
(SNO CHAR(3),
 PNO CHAR(3),
 JNO CHAR(3),
 QTY INT);
```

解析：

读者完成建表后首先插入若干数据，如第 2 章第 6 题。

① 求供应工程 J1 零件的供应商号码 SNO。

```
SELECT SNO
FROM SPJ
WHERE JNO='J1';
```

② 求供应工程 J1 零件 P1 的供应商号码 SNO。

```
SELECT SNO
FROM SPJ
WHERE JNO='J1' AND PNO='P1';
```

③ 求供应工程 J1 零件为红色的供应商号码 SNO。

```
SELECT SNO                    /*这是嵌套查询*/
```

```
FROM SPJ
WHERE JNO='J1'
  AND PNO IN                    /*找出红色零件的零件号码 PNO */
      (SELECT PNO
       FROM P                   /*从 P 表中找*/
       WHERE COLOR='红');
```

或

```
SELECT SNO
FROM SPJ,P                      /*这是两表连接*/
WHERE JNO='J1'                  /*这是复合条件连接查询*/
   AND SPJ.PNO=P.PNO AND COLOR='红';
```

④ 求没有使用天津供应商生产的红色零件的工程号 JNO。

解析：

读者可以对比第 2 章习题 5 中用 ALPHA 语言来完成该查询的解答：

```
GET W (J.JNO): ¬∃SPJX( SPJX .JNO=J.JNO ∧
                  ∃SX ( SX.SNO=SPJX .SNO ∧SX .CITY='天津' ∧
                  ∃PX ( PX .PNO=SPJX .PNO ∧PX .COLOR='红' ))
```

如果理解了有关该题的解析说明，那么本题的解答可以看成是把关系演算用 SQL 来表示的过程。

```
SELECT JNO                                  /*这种解法是使用多重嵌套查询*/
FROM J                                      /*注意:从 J 表入手,以包含那些*/
WHERE NOT EXISTS                            /*尚未使用任何零件的工程号*/
    (SELECT *
     FROM SPJ
     WHERE SPJ.JNO=J.JNO
       AND SNO IN
           (SELECT SNO                      /*天津供应商的 SNO*/
            FROM S
            WHERE CITY='天津')
       AND PNO IN                           /*红色零件的 PNO*/
           (SELECT PNO
            FROM P
            WHERE COLOR='红'));
```

或

```
SELECT JNO
FROM J
```

```
WHERE NOT EXISTS
    (SELECT *
     FROM SPJ,S,P          /*这里的子查询是一个多表连接*/
     WHERE SPJ.JNO=J.JNO AND SPJ.SNO=S.SNO
         AND SPJ.PNO=P.PNO AND S.CITY='天津'
         AND P.COLOR='红');
```

⑤ 求至少用了供应商 S1 所供应的全部零件的工程号 JNO(类似于《概论》书 3.4.3 小节[例 3.63])。

解析:

本查询的解析可以参考第 2 章第 5 题,用 ALPHA 语言的逻辑蕴涵来表达。

上述查询可以抽象为:要求这样的工程 x,使 $(\forall y)\ p \rightarrow q$ 为真。即:

对于所有的零件 y,满足逻辑蕴涵 $p \rightarrow q$:

P 表示谓词:"供应商 S1 供应了零件 y"

q 表示谓词:"工程 x 选用了零件 y"

即:只要"供应商 S1 供应了零件 y"为真,则"工程 x 选用了零件 y"为真。

逻辑蕴涵可以转换为等价形式:

$$(\forall y)p \rightarrow q \equiv \neg(\exists y\ (\neg(p \rightarrow q)) \equiv \neg(\exists y\ (\neg(\neg p \vee q) \equiv \neg\exists y(p \wedge \neg q)$$

它所表达的语义为:不存在这样的零件 y,供应商 S1 供应了 y,而工程 x 没有选用 y。用 SQL 语言表示如下:

```
SELECT DISTINCT JNO
FROM SPJ SPJZ
WHERE NOT EXISTS          /*这是一个相关子查询 */
    (SELECT *             /*父查询和子查询均引用了 SPJ 表*/
     FROM SPJ SPJX        /*用别名 SPJZ、SPJX 将父查询*/
     WHERE SNO='S1'       /*与子查询中的 SPJ 表区分开*/
     AND NOT EXISTS
        (SELECT *         /*用别名 SPJY 与父查询*/
         FROM SPJ SPJY    /*中的 SPJ 表区分开*/
         WHERE SPJY.PNO=SPJX.PNO
             AND SPJY.JNO=SPJZ.JNO));
```

可以把 SQL 语言和关系代数、ALPHA 语言、QBE 语言进行比较,体会各种语言的优点。

5. 针对习题 3 中的 4 个表,试用 SQL 语言完成以下各项操作:

答:

① 找出所有供应商的姓名和所在城市。

```
SELECT SNAME,CITY
```

```
FROM S;
```

② 找出所有零件的名称、颜色、重量。

```
SELECT PNAME,COLOR,WEIGHT
FROM P;
```

③ 找出使用供应商 S1 所供应零件的工程号码。

```
SELECT JNO
FROM SPJ
WHERE SNO='S1';
```

④ 找出工程项目 J2 使用的各种零件的名称及其数量。

```
SELECT P.PNAME,SPJ.QTY
FROM P,SPJ
WHERE P.PNO=SPJ.PNO
    AND SPJ.JNO='J2';
```

⑤ 找出上海厂商供应的所有零件号码。

```
SELECT DISTINCT  PNO
FROM SPJ
WHERE SNO IN
    (SELECT SNO
     FROM S
     WHERE CITY='上海');
```

⑥ 找出使用上海产的零件的工程名称。

```
SELECT JNAME
FROM J,SPJ,S
WHERE J.JNO=SPJ.JNO
    AND SPJ.SNO=S.SNO
    AND S.CITY='上海';
```

或

```
SELECT JNAME
FROM J
WHERE JNO IN
    (SELECT JNO
     FROM SPJ,S
     WHERE SPJ.SNO=S.SNO
         AND S.CITY='上海');
```

⑦ 找出没有使用天津产的零件的工程号码。

```
SELECT JNO
```

```
FROM J
WHERE NOT EXISTS
    (SELECT  *
     FROM SPJ
     WHERE SPJ.JNO=J.JNO
         AND SNO IN
             (SELECT SNO
              FROM S
              WHERE CITY='天津'));
```

或

```
SELECT JNO
FROM J
WHERE NOT EXISTS
    (SELECT  *
     FROM SPJ,S
     WHERE SPJ.JNO=J.JNO
         AND SPJ.SNO=S.SNO
         AND S.CITY='天津');
```

⑧ 把全部红色零件的颜色改成蓝色。

```
UPDATE P
SET COLOR='蓝'
WHERE COLOR='红';
```

⑨ 由 S5 供给 J4 的零件 P6 改为由 S3 供应,请作必要的修改。

```
UPDATE SPJ
SET SNO='S3'
WHERE SNO='S5'
    AND JNO='J4'
    AND PNO='P6';
```

⑩ 从供应商关系中删除 S2 的记录,并从供应情况关系中删除相应的记录。

```
DELETE
FROM SPJ
WHERE SNO='S2';

DELETE
FROM S
WHERE SNO='S2';
```

解析:

注意删除顺序,应该先从 SPJ 表中删除供应商 S2 所供应零件的记录,然后从 S 表中删除 S2。

⑪ 请将 (S2,J6,P4,200) 插入供应情况关系。

```
INSERT INTO SPJ(SNO,JNO,PNO,QTY)      /* INTO 子句中指明列名 */
VALUES(S2,J6,P4,200);                 /* 插入的属性值与指明列要对应 */
```

或

```
INSERT INTO SPJ          /* INTO 子句中没有指明列名 */
VALUES(S2,P4,J6,200); /* 插入的记录在每个属性列上均有值 */
                      /* 并且属性列要和表定义中的次序一致 */
```

6. 什么是基本表? 什么是视图? 两者的区别和联系是什么?

答:

基本表是本身独立存在的表,在 SQL 中一个关系就对应一个基本表。

视图是从一个或几个基本表导出的表。视图本身不独立存储在数据库中,是一个虚表。即数据库中只存放视图的定义而不存放视图对应的数据,这些数据仍存放在导出视图的基本表中。视图在概念上与基本表等同,用户可以如同基本表那样使用视图,可以在视图上再定义视图。

7. 试述视图的优点。

答:

① 视图能够简化用户的操作。

② 视图使用户能以多种角度看待同一数据。

③ 视图对重构数据库提供了一定程度的逻辑独立性。

④ 视图能够对机密数据提供安全保护。

详细解释参见《概论》第 3.7.4 小节。

8. 哪类视图是可以更新的,哪类视图是不可更新的? 各举一例说明。

答:

基本表的行列子集视图一般是可更新的。如《概论》第 3.7.1 小节中的[例 3.84]。

若视图的属性来自聚集函数、表达式,则该视图肯定是不可以更新的。如《概论》第 3.7.1 小节[例 3.89]中的 S_G 视图。

9. 请为三建工程项目建立一个供应情况的视图,包括供应商代码(SNO)、零件代码(PNO)、供应数量(QTY)。针对该视图完成下列查询:

① 找出三建工程项目使用的各种零件代码及其数量。

② 找出供应商 S1 供应三建工程的情况。

答:

创建视图:

```
CREATE VIEW V_SPJ AS
```

```
SELECT SNO,PNO,QTY
FROM SPJ
WHERE JNO=
    (SELECT JNO
     FROM J
     WHERE JNAME='三建');
```

对该视图查询：

```
① SELECT PNO,QTY
  FROM V_SPJ;
② SELECT PNO,QTY                    /* S1 供应三建工程的零件代码和对应的数量 */
  FROM V_SPJ
  WHERE SNO='S1';
```

3.3 补 充 习 题

1. 选择题

(1) 关于 SQL 语言，下列说法正确的是(　　)。

A. 数据控制功能不是 SQL 语言的功能之一

B. SQL 采用的是面向记录的操作方式，以记录为单位进行操作

C. SQL 是非过程化的语言，用户无须指定存取路径

D. SQL 作为嵌入式语言语法与独立的语言有较大差别

(2) 对表中数据进行删除的操作是(　　)。

A. DELETE　　B. DROP　　C. ALTER　　D. UPDATE

(3) 数据库中建立索引的目的是为了(　　)。

A. 加快建表速度　　B. 加快存取速度　　C. 提高安全性　　D. 节省存储空间

(4) 视图是数据库系统三级模式中的(　　)。

A. 外模式　　B. 模式　　C. 内模式　　D. 模式映像

(5) 下列说法不正确的是(　　)。

A. 基本表和视图一样，都是关系

B. 可以使用 SQL 对基本表和视图进行操作

C. 可以从基本表或视图上定义视图

D. 基本表和视图中都存储数据

2. 判断题

(1) 视图不仅可以从单个基本表导出，还可以从多个基本表导出。　　(　　)

(2) 不是所有的视图都可以进行更新，但视图都可以进行插入。　　(　　)

(3) SELECT 子句中的目标列可以是表中的属性列,也可以是表达式。 ( )

(4) 在 SQL 语句中表达某个属性 X 为空,可以使用 WHERE X=NULL。 ( )

(5) SQL 语句中逻辑运算符 AND 和 OR 的优先级是一样的。 ( )

(6) 使用 ANY 或 ALL 谓词时必须与比较运算符同时使用。 ( )

3. 填空题

(1) SQL 语言具有_____、_____、_____和数据控制的功能。

(2) SQL 语句中用来消除重复的关键词是_____。

(3) 若一个视图是从单个基本表导出的,并且只是去掉了基本表的某些行和某些列,但保留了主码,这类视图称为_____。

(4) SQL 语言的数据定义功能包括_____、表定义、视图定义和_____等。

4. 问答题

(1) 解释相关子查询和不相关子查询。

(2) 写出 ANY 和 ALL 谓词与聚集函数或 IN 谓词可能存在的等价转换关系。

5. 综合题

关系 R 包含 A、B、C 三个属性,包含的数据如下:

R

A	B	C
10	NULL	20
20	30	NULL

写出对查询语句 SELECT * FROM R WHERE X;当 X 为下列条件时的查询结果:

① A IS NULL

② A>8 AND B<20

③ A>10 OR B<20

④ C+10>25

⑤ EXISTS(SELECT B FROM R WHERE A=10)

⑥ C IN (SELECT B FROM R)

3.4 补充习题答案

1. 选择题

(1)	(2)	(3)	(4)	(5)
C	A	B	A	D

2. 判断题

(1)	(2)	(3)	(4)	(5)	(6)
√	×	√	×	×	√

3. 填空题

(1) 数据定义　数据查询　数据操纵

(2) DISTINCT

(3) 行列子集视图

(4) 模式定义　索引定义

4. 问答题

(1) 解释相关子查询和不相关子查询。

答：

在嵌套查询中，如果子查询的查询条件不依赖于父查询，称为不相关子查询；如果子查询的查询条件依赖于父查询，称为相关子查询。

(2) 写出 ANY 和 ALL 谓词与聚集函数或 IN 谓词可能存在的等价转换关系。

答：

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

5. 综合题

① 空的结果集

② 空的结果集

③

A	*B*	*C*
20	30	NULL

④

A	*B*	*C*
10	NULL	20

⑤

A	*B*	*C*
10	NULL	20
20	30	NULL

解析：

SELECT * FROM R WHERE EXISTS(SELECT B FROM R WHERE A=10)

这里的条件中(SELECT B FROM R WHERE A=10)返回一条记录，只不过这条记录只有一个属性 B，且其值是 NULL，所以 EXISTS 是返回真值。因此最后的答案是返回 R 的全部记录。

⑥ 空的结果集

解析：

SELECT * FROM R WHERE C IN (SELECT B FROM R),

(SELECT B FROM R)执行的结果是(NULL,30)，由于该集合中存在 NULL 值，无论 C 取何值，条件表达式 C IN (NULL,30) 总为 NULL 值，因此该 WHERE 条件为假，SELECT * FROM R WHERE C IN (NULL,30) 执行结果应该是空的结果集。

第 4 章　数据库安全性

《概论》第 4 章详细介绍数据库安全性问题和实现技术。信息安全、计算机系统安全以及数据库系统安全是信息安全的重要内容。随着计算机特别是计算机网络的发展，数据的共享日益加强，数据的安全保密越来越重要。

4.1　基本知识点

数据库的安全性问题和计算机系统的安全性是紧密联系的，计算机系统的安全性问题可分技术安全类、管理安全类和政策法律类三大类安全性问题。我们讨论数据库的安全性，讨论数据库技术安全类问题，即从技术上如何保证数据库系统的安全性。

① 需要了解的：什么是计算机系统安全性问题，什么是数据库的安全性问题，威胁数据库安全性的因素有哪些。

② 需要牢固掌握的：TCSEC 和 CC 标准的主要内容。C2 级 DBMS、B1 级 DBMS 的主要特征。DBMS 提供的安全措施，包括用户身份鉴别、自主存取控制和强制存取控制技术、视图技术和审计技术、数据加密存储和加密传输等。

③ 需要举一反三的：使用 SQL 语言中的 GRANT 语句和 REVOKE 语句来实现自主存取控制。

④ 难点：强制存取控制机制中确定主体能否存取客体的存取规则，要理解并掌握存取规则为什么要这样规定，特别是规则(2)关于主体写客体的规则。

4.2　习题解答和解析

1. 什么是数据库的安全性？

答：数据库的安全性是指保护数据库以防止不合法的使用所造成的数据泄露、更改或破坏。

2. 举例说明对数据库安全性产生威胁的因素。

解析：

请读者列出自己生活和工作中遇到、听到的实际案例。

3. 试述信息安全标准的发展历史，试述 CC 评估保证级划分的基本内容。

答：

信息安全标准的发展历史详见《概论》书图 4.1。

简单地讲，TCSEC 是 1985 年美国国防部颁布的《DoD 可信计算机系统评估准则》。CC 通用准则 V2.1 版于 1999 年被 ISO 采用为国际标准，2001 年被我国采用为国家标准。目前 CC 已经基本取代了 TCSEC，成为评估信息产品安全性的主要标准。

CC 评估保证级划分的基本内容如下表所示。

评估保证级	定义
EAL1	功能测试
EAL2	结构测试
EAL3	系统的测试和检查
EAL4	系统的设计、测试和复查
EAL5	半形式化设计和测试
EAL6	半形式化验证的设计和测试
EAL7	形式化验证的设计和测试

详见《概论》表 4.2。

4. 试述实现数据库安全性控制的常用方法和技术。

答：

实现数据库安全性控制的常用方法和技术有以下几种。

① 用户身份鉴别：系统提供多种方式让用户标识自己的名字或身份。用户要使用数据库系统时，由系统进行核对，通过鉴定后才可以使用数据库。

② 多层存取控制：系统提供用户权限定义和合法权限检查功能，用户只有获得某种权限才能访问数据库中的某些数据。

③ 视图机制：为不同的用户定义不同的视图，通过视图机制把要保密的数据对无权存取的用户隐藏起来，从而自动对数据提供一定程度的安全保护。

④ 审计：建立审计日志，把用户对数据库的所有操作自动记录下来放入审计日志中，审计员可以利用审计信息重现导致数据库现有状况的一系列事件，找出非法存取数据的人、时间和内容等。

⑤ 数据加密：对存储和传输的数据进行加密处理，从而使不知道解密算法的人无法获知

数据的内容。

5. 什么是数据库中的自主存取控制方法和强制存取控制方法?

答:

自主存取控制方法:定义各个用户对不同数据对象的存取权限。当用户对数据库访问时首先检查用户的存取权限。防止不合法用户对数据库的存取。

强制存取控制方法:每一个数据对象被(强制地)标以一定的密级,每一个用户也被(强制地)授予某一个级别的许可证。系统规定只有具有某一许可证级别的用户才能存取某一个密级的数据对象。

自主存取控制中自主的含义是:用户可以将自己拥有的存取权限"自主"地授予别人。即用户具有一定的"自主"权。

6. 对下列两个关系模式:

学生(学号,姓名,年龄,性别,家庭住址,班级号)

班级(班级号,班级名,班主任,班长)

使用 GRANT 完成下列授权功能:

① 授予用户 U1 拥有对两个表的所有权限,并可给其他用户授权;

② 授予用户 U2 对学生表具有查看权限,对家庭住址具有更新权限;

③ 将对班级表查看权限授予所有用户。

④ 将对学生表的查询、更新权限授予角色 R1。

⑤ 将角色 R1 授予用户 U1,并且 U1 可继续授权给其他角色。

答:

① GRANT ALL PRIVILEGES ON TABLE 学生,班级 TO U1 WITH GRANT OPTION;

② GRANT SELECT,UPDATE(家庭住址) ON TABLE 学生 TO U2;

③ GRANT SELECT ON TABLE 班级 TO PUBLIC;

④ GRANT SELECT,UPDATE ON TABLE 学生 TO R1;

⑤ GRANT R1 TO U1 WITH ADMIN OPTION;

7. 今有两个关系模式:

职工(职工号,姓名,年龄,职务,工资,部门号)

部门(部门号,名称,经理名,地址,电话号)

请用 SQL 的 GRANT 和 REVOKE 语句(加上视图机制)完成以下授权定义或存取控制功能:

① 用户王明对两个表有 SELECT 权限。

GRANT SELECT ON TABLE 职工,部门 TO 王明;

② 用户李勇对两个表有 INSERT 和 DELETE 权限。

GRANT INSERT,DELETE ON TABLE 职工,部门 TO 李勇;

③ *每个职工只对自己的记录有 SELECT 权限;

```
GRANT SELECT ON TABLE 职工 WHEN USER( )= NAME TO ALL;
```

这里假定系统的 GRANT 语句支持 WHEN 子句和 USER( )的使用。用户将自己的名字作为 ID。注意,不同的系统这些扩展语句可能是不同的。读者应该了解所使用的 DBMS 产品的扩展语句。

④ 用户刘星对职工表有 SELECT 权限,对工资字段具有更新权限。

```
GRANT SELECT,UPDATE(工资) ON TABLE 职工 TO 刘星;
```

⑤ 用户张新具有修改这两个表的结构的权限。

```
GRANT ALTER TABLE ON TABLE 职工,部门 TO 张新;
```

⑥ 用户周平具有对两个表所有权限(读,插,改,删数据),并具有给其他用户授权的权限。

```
GRANT ALL PRIVILEGES ON TABLE 职工,部门 TO 周平 WITH GRANT OPTION;
```

⑦ 用户杨兰具有从每个部门职工中 SELECT 最高工资、最低工资、平均工资的权限,他不能查看每个人的工资。

首先建立一个视图。然后对这个视图定义杨兰的存取权限。

```
CREATE VIEW 部门工资 AS
SELECT 部门.名称,MAX(工资),MIN(工资),AVG(工资)
FROM 职工,部门
WHERE 职工.部门号=部门.部门号
GROUP BY 职工.部门号;
GRANT SELECT ON TABLE 部门工资 TO 杨兰;
```

8. 把习题 7 中①~⑦的每一种情况,撤销各用户所授予的权限。

答:

① REVOKE SELECT ON TABLE 职工,部门 FROM 王明;

② REVOKE INSERT,DELETE ON TABLE 职工,部门 FROM 李勇;

③ REOVKE SELECT ON TABLE 职工 WHEN USER( )= NAME FROM ALL;

这里假定用户将自己的名字作为 ID,且系统的 REOVKE 语句支持 WHEN 子句,系统也支持 USER( )的使用。

④ REVOKE SELECT,UPDATE ON TABLE 职工 FROM 刘星;

⑤ REVOKE ALTER TABLE ON TABLE 职工,部门 FROM 张新;

⑥ REVOKE ALL PRIVILEGES ON TABLE 职工,部门 FROM 周平;

⑦ REVOKE SELECT ON TABLE 部门工资 FROM 杨兰;
DROP VIEW 部门工资;

9. 理解并解释 MAC 机制中主体、客体、敏感度标记的含义。

答：

主体是系统中的活动实体，既包括 DBMS 所管理的实际用户，也包括代表用户的各进程。

客体是系统中的被动实体，是受主体操纵的，包括文件、基本表、索引、视图等。

对于主体和客体，DBMS 为它们每个实例（值）指派一个敏感度标记。敏感度标记被分成若干级别，如绝密、机密、可信、公开等。主体的敏感度标记称为许可证级别，客体的敏感度标记称为密级。

10. 举例说明强制存取控制机制是如何确定主体能否存取客体。

答：

假设要对关系变量 S 进行强制存取控制，为简化起见，假设要控制存取的数据单元是元组，则每个元组标以密级，如下表所示（4＝绝密，3＝机密，2＝秘密）。

SNO	SNAME	STATUS	CITY	CLASS
*S*1	Smith	20	London	2
*S*2	Jones	10	Paris	3
*S*3	Clark	20	London	4

假设系统的存取规则是：

① 仅当主体的许可证级别大于或等于客体的密级时才能读取相应的客体；

② 仅当主体的许可证级别小于或等于客体的密级时才能写相应的客体。

假定用户 U1 和 U2 的许可证级别分别为 3 和 2，则根据规则用户 U1 能读元组 S1 和 S2，可修改元组 S2；用户 U2 只能读元组 S1，修改元组 S1。

11. 什么是数据库的审计功能，为什么要提供审计功能？

答：

审计功能是指 DBMS 的审计模块在用户对数据库执行操作的同时，把所有操作自动记录到系统的审计日志中。

因为任何系统的安全保护措施都不是完美无缺的，蓄意盗窃破坏数据的人总可能存在。利用数据库的审计功能，审计员可以根据审计日志中记录的信息，分析和重现导致数据库现有状况的一系列事件，找出非法存取数据的人、时间和内容等。

4.3　补 充 习 题

1. 选择题

（1）强制存取控制策略是 TCSEC/TDI 哪一级安全级别的特色（　　）。

A. C1　　B. C2　　C. B1　　D. B2

(2) SQL 的 GRANT 和 REVOKE 语句可以用来实现(　　)。

A. 自主存取控制　　B. 强制存取控制

C. 数据库角色创建　　D. 数据库审计

(3) 在强制存取控制机制中,当主体的许可证级别等于客体的密级时,主体可以对客体进行如下操作(　　)。

A. 读取　　B. 写入　　C. 不可操作　　D. 读取、写入

2. 填空题

(1) 数据库安全技术包括用户身份鉴别、_____、_____、_____和数据加密等。

(2) 在数据加密技术中,原始数据通过某种加密算法变换为不可直接识别的格式,称为_____。

(3) 数据库角色实际上是一组与数据库操作相关的各种_____。

(4) 在对用户授予列 INSERT 权限时,一定要包含对_____的 INSERT 权限,否则用户的插入会因为空值被拒绝。除了授权的列,其他列的值或者取_____,或者为_____。

4.4 补充习题答案

1. 选择题

(1)	(2)	(3)
C	A	D

2. 填空题

(1) 自主存取控制和强制存取控制　视图　审计

(2) 密文

(3) 权限

(4) 主码　空值　默认值

第 5 章 数据库完整性

《概论》第 5 章详细介绍数据库的完整性。数据库的完整性是指数据库中数据的正确性。由于数据库中的数据之间是相互联系的,因此数据库的完整性还包含数据的相容性。

数据库的完整性包括三个方面,完整性约束定义机制、完整性检查机制和违背完整性约束条件时应采取的预防措施。

5.1 基本知识点

① 需要了解的:什么是数据库的完整性约束条件,完整性约束条件的分类,数据库的完整性概念与数据库的安全性概念的区别和联系。

② 需要牢固掌握的:DBMS 完整性控制机制的三个方面,即完整性约束条件的定义、完整性约束条件的检查和违约处理;使用触发器实现数据库完整性的方法。

③ 需要举一反三的:用 SQL 语言定义关系模式的完整性约束条件,包括定义每个模式的主码;定义参照完整性;定义与应用有关的完整性。

④ 难点:RDBMS 如何实现参照完整性的策略,即当操作违反实体完整性、参照完整性和用户定义的完整性约束条件时,RDBMS 应该如何进行处理,以确保数据的正确与有效。其中比较复杂的是参照完整性的实现机制。

5.2 习题解答和解析

1. 什么是数据库的完整性?

答:数据库的完整性是指数据的正确性和相容性。

2. 数据库的完整性概念与数据库的安全性概念有什么区别和联系?

答:数据的完整性和安全性是两个不同的概念,但是有一定的联系。前者是为了防止数据库中存在不符合语义的数据,防止错误信息的输入和输出,即所谓垃圾进垃圾出(Garbage In Garbage Out)所造成的无效操作和错误结果;后者是保护数据库防止恶意的破坏和非法的

存取。也就是说,安全性措施的防范对象是非法用户和非法操作,完整性措施的防范对象是不合语义的数据。

3. 什么是数据库的完整性约束条件?

答:完整性约束条件是指数据库中的数据应该满足的语义约束条件。

4. DBMS 的完整性控制机制应具有哪三个方面的功能?

答:

① 定义功能,即提供定义完整性约束条件的机制。

② 检查功能,即检查用户发出的操作请求是否违背了完整性约束条件。

③ 违约处理功能:如果发现用户的操作请求使数据违背了完整性约束条件,则采取一定的动作来保证数据的完整性。

5. RDBMS 在实现参照完整性时需要考虑哪些方面?

答:

RDBMS 在实现参照完整性时需要考虑可能破坏参照完整性的各种情况,以及用户违约后的处理策略。

《概论》表 5.1 清楚地总结了可能破坏参照完整性的 4 种情况以及可以采取的不同的违约处理策略,详细讨论可以参见《概论》第 5.2.2 小节。

被参照表(例如 Student)	参照表(例如 SC)	违约处理
可能破坏参照完整性 ←	插入元组	拒绝
可能破坏参照完整性 ←	修改外码值	拒绝
删除元组 →	可能破坏参照完整性	拒绝/级联删除/设置为空值
修改主码值 →	可能破坏参照完整性	拒绝/级联修改/设置为空值

6. 假设有下面两个关系模式:

职工(职工号,姓名,年龄,职务,工资,部门号),其中职工号为主码;

部门(部门号,名称,经理名,电话),其中部门号为主码;

用 SQL 语言定义这两个关系模式,要求在模式中完成以下完整性约束条件的定义:(1) 定义每个模式的主码;(2) 定义参照完整性;(3) 定义职工年龄不得超过 60 岁。

答:

```
CREATE TABLE DEPT
    (Deptno NUMBER(2)PRIMARY KEY,
     Deptname VARCHAR(10),
     Manager VARCHAR(10),
```

```
    PhoneNumber Char(12)
    );

CREATE TABLE EMP
    (Empno NUMBER(4)PRIMARY KEY,
     Ename VARCHAR(10),
     Age NUMBER(2),
     Job VARCHAR(9),
     Sal NUMBER(7,2),
     Deptno NUMBER(2),
        CONSTRAINT C1 CHECK(Age<=60),
        CONSTRAINT FK_DEPTNO FOREIGN KEY(Deptno)REFERENCES DEPT(Deptno));
```

7. 在关系系统中，当操作违反实体完整性、参照完整性和用户定义的完整性约束条件时，一般是如何分别进行处理的？

答：对于违反实体完整性和用户定义的完整性的操作，一般都采用拒绝执行的方式进行处理；而对于违反参照完整性的操作，并不都是简单地拒绝执行，有时要根据应用语义执行一些附加的操作，以保证数据库的正确性。具体的处理可以参见上面第 5 题或《概论》第 5.2.2 小节。

8. 某单位想举行一个小型的联谊会，关系 Male 记录注册的男宾信息，关系 Female 记录注册的女宾信息。建立一个断言，将来宾的人数限制在 50 人以内（提示，先创建了关系 Female 和关系 Male）。

答：

```
CREATE TABLE Male                                    /*创建关系 Male*/
    (SerialNumber SmallInt PRIMARY KEY,              /*注册的序列号*/
     Name Char(8),
     Age SmallInt,
     Occupation Char(20)
    );
CREATE TABLE Female                                  /*创建关系 Female*/
    (SerialNumber SmallInt PRIMARY KEY,              /*注册的序列号*/,
     Name Char(8),
     Age SmallInt,
     Occupation Char(20)
    );
    CREATE ASSERTION Party                           /*创建断言 PARTY*/
    CHECK((SELECT COUNT(*) FROM Male)+(SELECT COUNT(*)FROM Female)
```

<=50);

注意,KingbaseES 目前还不支持断言。

5.3 补充习题

1. 选择题

(1) 定义关系的主码意味着主码属性(　　)。

A. 必须唯一　　B. 不能为空

C. 唯一且部分主码属性不为空　　D. 唯一且所有主码属性不为空

(2) 关于语句 CREATE TABLE R(no int,sum int CHECK(sum > 0)) 和 CREATE TABLE R(no int,sum int,CHECK(sum > 0)),以下说法不正确的是(　　)。

A. 两条语句都是合法的

B. 前者定义了属性上的约束条件,后者定义了元组上的约束条件

C. 两条语句的约束效果不一样

D. 当 sum 属性改变时检查,上述两种 CHECK 约束都要被检查

(3) 下列说法正确的是(　　)。

A. 使用 ALTER TABLE ADD CONSTRAINT 可以增加基于元组的约束

B. 如果属性 A 上定义了 UNIQUE 约束,则 A 不可以为空

C. 如果属性 A 上定义了外码约束,则 A 不可以为空

D. 不能使用 ALTER TABLE ADD CONSTRAINT 增加主码约束

2. 填空题

(1) 在 CREATE TABLE 时,用户定义的完整性可以通过_____、_____、_____等子句实现。

(2) 关系 R 的属性 A 参照引用关系 T 的属性 A,T 的某条元组对应的 A 属性值在 R 中出现,当要删除 T 的这条元组时,系统可以采用的策略包括_____、_____、_____。

(3) 定义数据库完整性一般是由 SQL 的_____语句实现的。

3. 综合题

(1) 考虑下面的关系模式:

研究人员(人员编号,姓名,年龄,职称)

项目(项目编号,名称,负责人编号,类别)

参与(项目编号,人员编号,工作时间)/*一个研究人员可以参加多个项目,一个项目有多个研究人员参加,工作时间给出某研究人员参加某项目的月数*/

写出下面的完整性约束:

① 定义三个关系中的主码、外码、参照完整性;

② 每个研究人员的年龄不能超过 35 岁；

③ 每个研究人员的职称只能是“讲师”、“副教授”或“教授”；

④ 一个研究人员参加各种项目的总工作时间不能超过 12 个月；

⑤ 每个项目至少有 5 位研究人员；

⑥ 每个研究人员参加的项目数不能超过 3 个。

(2) 考虑题(1)中的关系模式，使用 ALTER TABLE ADD CONSTRAINT 声明如下完整性约束。

① 负责人编号参照研究人员的“人员编号”属性，当对“研究人员”更新时，若违反约束则拒绝操作；

② 同①，但当违反约束时将负责人编号置为 NULL；

③ 同①，但当违反约束时将项目中的相应元组删除或修改；

④ 工作时间在 1 到 12 之间；

⑤ 项目名称不能为空。

(3) 使用 CHECK 短语写出题(1)中项目关系的参照完整性约束。

(4) 考虑下面的关系模式：

```
Teacher(Tno,Tname,Tage,Tsex)
Department(Dno,Dname,Tno)      /* 其中 Tno 为系主任的职工号 */
Work(Tno,Dno,Year,Salary)      /* 某系某职工在某一年的工资 */
```

将下列要求写成触发器：

① 在插入新教师时，也将此教师信息插入到 Work 关系中，不确定的属性赋以 NULL 值；

② 在更新教师年龄时，如果新年龄比旧年龄低，则用旧年龄代替。

5.4　补充习题答案

1. 选择题

(1)	(2)	(3)
D	C	A

2. 填空题

(1) NOT NULL　UNIQUE　CHECK

(2) 拒绝执行　级联删除　设为空值

(3) DDL

3. 综合题

(1) CREATE TABLE 研究人员

```
    (人员编号 int PRIMARY KEY,
     姓名 Char(8),
     年龄 SmallInt CHECK(年龄 <=35),
     职称 Char(8) CHECK(职称 in('讲师','副教授','教授'))
    );
CREATE TABLE 项目
    (项目编号 int PRIMARY KEY,
     名称 Char(20),
     负责人编号 int,
     类别 Char(8),
     FOREIGN KEY(负责人编号) REFERENCES 研究人员(人员编号)
    );
CREATE TABLE 参与
    (项目编号 int,
     人员编号 int,
     工作时间 SmallInt,
     PRIMARY KEY(项目编号,人员编号),
     FOREIGN KEY(项目编号) REFERENCES 项目(项目编号),
     FOREIGN KEY(人员编号) REFERENCES 研究人员(人员编号),
    );
    CREATE ASSERTION 工作时间限制     /*创建断言工作时间限制*/
    CHECK(12 >=ALL(SELECT SUM(工作时间) FROM 参与 GROUP BY 人员编号));

    CREATE ASSERTION 项目参加人数     /*创建断言项目参加人数限制*/
    CHECK(5 <=ALL(SELECT COUNT(人员编号) FROM 参与 GROUP BY 项目编号));

    CREATE ASSERTION 研究员参加项目   /*创建断言研究员参加项目数限制*/
    CHECK(3 >=ALL(SELECT COUNT(项目编号) FROM 参与 GROUP BY 人员编号));
```

(2)

① ALTER TABLE 项目 ADD CONSTRAINT C1 FOREIGN KEY(负责人编号) REFERENCES 研究人员(人员编号) ON UPDATE NO ACTION ; /*违反约束时 KingbaseES 的默认值是 NO ACTION,所以 NO ACTION 也可以缺省*/

② ALTER TABLE 项目 ADD CONSTRAINT C2 FOREIGN KEY(负责人编号) REFERENCES 研究人员(人员编号) ON DELETE SET NULL ON UPDATE SET NULL;

③ ALTER TABLE 项目 ADD CONSTRAINT c3 FOREIGN KEY(负责人编号) REFERENCES 研究人员(人员编号) ON DELETE CASCADE ON UPDATE CASCADE;

④ ALTER TABLE 参与 ADD CONSTRAINT c4 CHECK(工作时间 >=1 AND 工作时间 <=12);

⑤ ALTER TABLE 项目 ADD CONSTRAINT c5 CHECK(名称 IS NOT NULL);

(3) CHECK(负责人编号 IN(SELECT 人员编号 FROM 研究人员));

(4)

① CREATE TRIGGER NewTeacher
AFTER INSERT ON Teacher
FOR EACH ROW AS
BEGIN　/* 插入新教师后将此教师信息插入到 Work 中,不确定的属性赋以 NULL */
　　INSERT INTO work VALUES(new.tno,NULL,NULL,NULL);
END;

② CREATE TRIGGER UpdateAge
AFTER UPDATE of tage on Teacher
FOR EACH ROW AS
BEGIN
　　IF(new.tage < old.tage)　　/* 更新教师年龄时,如果新年龄比旧年龄小 */
　　UPDATE Teacher SET tage=old.tage WHERE tno=new.tno;　/* 则用旧年龄代替 */
END;

注意:不同的 RDBMS 实现的触发器语法各不相同,互不兼容。请读者在上机实验时注意阅读所用系统的使用说明。

第 6 章 关系数据理论

本章讲解关系数据理论。这是关系数据库的重点内容。学习本章的目的有两个:一个是理论方面的,本章用更加形式化的关系数据理论来描述和研究关系模型;另一个是实践方面的,关系数据理论是进行数据库设计的有力工具。因此,人们也把关系数据理论中的规范化理论称为数据库设计理论。本书把它放在数据库设计与应用开发部分介绍,以强调它对数据库设计的指导作用。

6.1 基本知识点

本章讲解关系数据理论,内容的理论性较强。本章内容分为基本要求部分(《概论》第 6.1~6.3 节)和高级部分(《概论》第 6.4 节)。前者是本科学生应该掌握的内容,后者是研究生应该学习掌握的内容。

① 需要了解的:什么是一个“不好”的数据库模式,什么是模式的插入异常和删除异常,规范化理论的重要意义。

② 需要牢固掌握的:关系的形式化定义,数据依赖的基本概念(函数依赖、平凡函数依赖、非平凡的函数依赖、部分函数依赖、完全函数依赖、传递函数依赖的概念,码、候选码、外码的概念和定义,多值依赖的概念),范式的概念,从 1NF 到 4NF 的定义,规范化的含义和作用。

③ 需要举一反三的:4 个范式的理解与应用,各个级别范式中存在的问题(插入异常、删除异常、数据冗余)和解决方法。能够根据应用语义完整地写出关系模式的数据依赖集合,并能根据数据依赖分析某一个关系模式属于第几范式。

④ 难点:各个级别范式之间的联系及其证明。

6.2 习题解答和解析

1. 理解并给出下列术语的定义:

函数依赖、部分函数依赖、完全函数依赖、传递依赖、候选码、主码、外码、全码(all-key)、

1NF、2NF、3NF、BCNF、多值依赖、4NF。

*解析:

解答本题不能仅仅把《概论》上的定义写下来,关键是要能真正理解和运用这些概念。

答:

函数依赖:设 $R(U)$ 是一个关系模式,U 是 R 的属性集合,X 和 Y 是 U 的子集。对于 $R(U)$ 的任意一个可能的关系 r,如果 r 中不存在两个元组,它们在 X 上的属性值相同,而在 Y 上的属性值不同,则称"X 函数确定 Y"或"Y 函数依赖于 X",记作 $X\rightarrow Y$。

*解析:

① 函数依赖是最基本的一种数据依赖,也是最重要的一种数据依赖。

② 函数依赖是属性之间的一种联系,体现在属性值是否相等。由上面的定义可以知道,如果 $X\rightarrow Y$,则 r 中任意两个元组,若它们在 X 上的属性值相同,那么在 Y 上的属性值一定也相同。

③ 我们要从属性间实际存在的语义来确定它们之间的函数依赖,即函数依赖反映了(描述了)现实世界的一种语义。

④ 函数依赖不是指关系模式 R 在某个时刻的关系(值)满足的约束条件,而是指 R 在任何时刻的一切关系均要满足的约束条件。

答:

完全函数依赖、部分函数依赖:在 $R(U)$ 中,如果 $X\rightarrow Y$,并且对于 X 的任何一个真子集 X',都有 $X'\not\rightarrow Y$,则称 Y 对 X 完全函数依赖,记作:

$$X \xrightarrow{F} Y$$

若 $X\rightarrow Y$,但 Y 不完全函数依赖于 X,则称 Y 对 X 部分函数依赖,记作:

$$X \xrightarrow{P} Y$$

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

候选码、主码:设 K 为 $R<U,F>$ 中的属性或属性组合,若 $K\xrightarrow{F}U$,则 K 为 R 的候选码。若候选码多于一个,则选定其中的一个为主码。

*解析:

① 这里用函数依赖来严格定义码的概念,在第 2 章 2.1.1 小节中只是描述性地定义码。若关系中的某一属性组的值能唯一地标识一个元组,则称该属性组为候选码。

② 因为码有了严格定义,读者在学习了《概论》6.3 节数据依赖的公理系统后,就可以从 $R<U,F>$ 的函数依赖集 F 出发,用算法来求候选码。

答:

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

全码:整个属性组是码,称为全码(all-key)。

答：

1NF：如果一个关系模式 R 的所有属性都是不可分的基本数据项，则 $R \in 1NF$。

＊解析：

第一范式是对关系模式的最起码的要求。不满足第一范式的数据库模式不能称为关系数据库。

答：

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

3NF：关系模式 $R<U,F>$ 中若不存在这样的码 X，属性组 Y 及非主属性 $Z(Z \nsubseteq Y)$ 使得 $X \rightarrow Y$，$(Y \nrightarrow X)$，$Y \rightarrow Z$ 成立，则称 $R<U,F> \in 3NF$。

BCNF：关系模式 $R<U,F> \in 1NF$。若 $X \rightarrow Y$ 且 $Y \nsubseteq X$ 时 X 必含有码，则 $R<U,F> \in BCNF$。

＊解析：

读者要真正理解这些范式的内涵。各种范式之间的联系：$5NF \subset 4NF \subset BCNF \subset 3NF \subset 2NF \subset 1NF$（《概论》图 6.2）。能够理解为什么有这种包含关系。

答：

多值依赖：设 $R(U)$ 是属性集 U 上的一个关系模式，X、Y、Z 是 U 的子集，并且 $Z=U-X-Y$。关系模式 $R(U)$ 中多值依赖 $X \rightarrow\rightarrow Y$ 成立，当且仅当对 $R(U)$ 的任一关系 r，给定的一对 (x,z) 值，有一组 Y 的值，这组值仅仅决定于 x 值而与 z 值无关。

4NF：关系模式 $R<U,F> \in 1NF$，如果对于 R 的每个非平凡多值依赖 $X \rightarrow\rightarrow Y(Y \nsubseteq X)$，$X$ 都含有码，则称 $R<U,F> \in 4NF$。

＊解析：

对于多值依赖的定义有多种。《概论》书定义 6.9 的后面又给出了一种等价的定义。习题中的第 4 题是另一种等价的定义。读者可以对比不同的定义来理解多值依赖，选择自己容易理解的一种定义来掌握多值依赖概念。

2. 建立一个关于系、学生、班级、学会等诸信息的关系数据库。

描述学生的属性有：学号、姓名、出生年月、系名、班号、宿舍区。

描述班级的属性有：班号、专业名、系名、人数、入校年份。

描述系的属性有：系名、系号、系办公室地点、人数。

描述学会的属性有：学会名、成立年份、地点、人数。

有关语义如下：一个系有若干专业，每个专业每年只招一个班，每个班有若干学生。一个系的学生住在同一宿舍区。每个学生可参加若干学会，每个学会有若干学生。学生参加某学会有一个入会年份。

请给出关系模式，写出每个关系模式的极小函数依赖集，指出是否存在传递函数依赖，对于函数依赖左部是多属性的情况，讨论函数依赖是完全函数依赖，还是部分函数依赖。

指出各关系的候选码、外部码，并说明有没有全码存在。

答：

关系模式：学生 S(SNO,SN,SB,DN,CNO,SA)

班级 C(CNO,CS,DN,CNUM,CDATE)

系 D(DNO,DN,DA,DNUM)

学会 P(PN,DATE1,PA,PNUM)

学生-学会 SP(SNO,PN,DATE2)

其中，SNO 学号，SN 姓名，SB 出生年月，SA 宿舍区；

CNO 班号，CS 专业名，CNUM 班级人数，CDATE 入校年份；

DNO 系号，DN 系名，DA 系办公室地点，DNUM 系人数；

PN 学会名，DATE1 成立年月，PA 地点，PNUM 学会会员人数；

DATE2 入会年份。

依据上面给出的语义，写出每个关系模式的极小函数依赖集如下。

S:SNO→SN,SNO→SB,SNO→CNO,CNO→DN,DN→SA

/*一个系的学生住在同一宿舍区*/

C:CNO→CS,CNO→CNUM,CNO→CDATE,CS→DN,(CS,CDATE)→CNO

/*每个专业每年只招一个班*/

D:DNO→DN,DN→DNO,DNO→DA,DNO→DNUM

/*按照实际情况，系名和系号是一一对应的*/

P:PN→DATE1,PN→PA,PN→PNUM

SP:(SNO,PN)→DATE2 /*学生参加某学会有一个入会年份*/

S 中存在的传递函数依赖：

因为 SNO→CNO,CNO→DN,所以存在传递函数依赖 SNO→DN,

因为 CNO→DN,DN→SA,所以存在传递函数依赖 CNO→SA,

因为 SNO→CNO,CNO→DN,DN→SA,所以存在传递函数依赖 SNO→SA。

C 中存在的传递函数依赖：

因为 CNO→CS,CS→DN ,所以存在传递函数依赖 CNO→DN。

函数依赖左部是多属性的情况：

(SNO,PN)→DATE2 和(CS,CDATE)→CNO 函数依赖左部具有 2 个属性，它们都是完全函数依赖，没有部分函数依赖的情况。

关系	候选码	外部码	全码
S	SNO	CNO,DN	无
C	CNO 和(CS,CDATE)	DN	无
D	DNO 和 DN	无	无
P	PN	无	无
SP	(SNO,PN)	SNO,PN	无

关系模式 C 和 D 都有 2 个候选码。

3. 试由 Armostrong 公理系统推导出下面三条推理规则：

① 合并规则：若 $X\rightarrow Z$，$X\rightarrow Y$，则有 $X\rightarrow YZ$；

② 伪传递规则：由 $X\rightarrow Y$，$WY\rightarrow Z$ 有 $XW\rightarrow Z$；

③ 分解规则：$X\rightarrow Y$，$Z\subseteq Y$，有 $X\rightarrow Z$。

证：

① 已知 $X\rightarrow Z$，由增广律知 $XY\rightarrow YZ$，又因为 $X\rightarrow Y$，可得 $XX\rightarrow XY\rightarrow YZ$，最后根据传递律得 $X\rightarrow YZ$。

② 已知 $X\rightarrow Y$，据增广律得 $XW\rightarrow WY$，因为 $WY\rightarrow Z$，所以 $XW\rightarrow WY\rightarrow Z$，通过传递律可知 $XW\rightarrow Z$。

③ 已知 $Z\subseteq Y$，根据自反律知 $Y\rightarrow Z$，又因为 $X\rightarrow Y$，所以由传递律可得 $X\rightarrow Z$。

4. 关于多值依赖的另一种定义是：

给定一个关系模式 $R(X,Y,Z)$，其中 X,Y,Z 可以是属性或属性组合。

设 $x\in X$，$y\in Y$，$z\in Z$，xz 在 R 中的像集为

$$Y_{xz}=\{r.Y \mid r.X=x \wedge r.Z=z \wedge r\in R\}$$

定义　$R(X,Y,Z)$ 当且仅当 $Y_{xz}=Y_{xz'}$ 对于每一组 (x,z,z') 都成立，则 Y 对 X 多值依赖，记作 $X\rightarrow\rightarrow Y$。这里，允许 Z 为空集，在 Z 为空集时，称为平凡的多值依赖。

请证明这里的定义和《概论》第 6.2.7 小节中定义 6.9 是等价的。

答：

先把定义 6.9 写在下面：

设 $R(U)$ 是属性集 U 上的一个关系模式。X,Y,Z 是 U 的子集，并且 $Z=U-X-Y$。关系模式 $R(U)$ 中多值依赖 $X\rightarrow\rightarrow Y$ 成立，当且仅当对 $R(U)$ 的任一关系 r，给定的一对 (x,z) 值，有一组 Y 的值，这组值仅仅决定于 x 值而与 z 值无关。

证：

设 $Y_{xz}=Y_{xz'}$ 对于每一组 (x,z,z') 都成立，现要证其能推出定义 6.9 的条件：

设 s、t 是关系 r 中的两个元组，$s[X]=t[X]$，由新定义的条件知对于每一个 z 值，都对应相同的一组 y 值。这样一来，对相同的 x 值，交换 y 值后所得的元组仍然属于关系 r，即定义 6.9 的条件成立。

如果定义 6.9 的条件成立，则对相同的 x 值，交换 y 值后所得的元组仍然属于关系 r，由于任意性及其对称性，可知每个 z 值对应相同的一组 y 值，所以 $Y_{xz}=Y_{xz'}$ 对于每一组 (x,z,z') 都成立。

综上可知，新定义和定义 6.9 的条件是等价的，所以新定义和定义 6.9 是等价的。

5. 试举出三个多值依赖的实例。

答：

① 关系模式 $MSC(M,S,C)$ 中，M 表示专业，S 表示学生，C 表示该专业的必修课。假设每个专业有多个学生，有一组必修课。设同专业内所有学生选修的必修课相同，实例关系如下。按照语义对于 M 的每一个值 M_i，S 有一个完整的集合与之对应而不问 C 取何值，所以 $M\rightarrow\rightarrow S$。由于 C 与 S 的完全对称性，必然有 $M\rightarrow\rightarrow C$ 成立。

M	S	C
$M1$	$S1$	$C1$
$M1$	$S1$	$C2$
$M1$	$S2$	$C1$
$M1$	$S2$	$C2$
……	……	……

② 关系模式 $ISA(I,S,A)$ 中，I 表示学生兴趣小组，S 表示学生，A 表示某兴趣小组的活动项目。假设每个兴趣小组有多个学生，有若干活动项目。每个学生必须参加所在兴趣小组的所有活动项目，每个活动项目要求该兴趣小组的所有学生参加。

按照语义有 $I\rightarrow\rightarrow S$，$I\rightarrow\rightarrow A$ 成立。

③ 关系模式 $RDP(R,D,P)$ 中，R 表示医院的病房，D 表示责任医务人员，P 表示病人。假设每个病房住有多个病人，有多个责任医务人员负责医治和护理该病房的所有病人。按照语义有 $R\rightarrow\rightarrow D$，$R\rightarrow\rightarrow P$ 成立。

6. 考虑关系模式 $R(A,B,C,D,E)$，回答下面各个问题：

① 若 A 是 R 的候选码，具有函数依赖 $BC\rightarrow DE$，那么在什么条件下 R 是 $BCNF$；

答：属性 BC 包含码。

② 如果存在依赖：$A\rightarrow B$，$BC\rightarrow D$，$DE\rightarrow A$，列出 R 的所有码；

答：ACE，DEC，BCE。

③ 如果存在依赖：$A\rightarrow B$，$BC\rightarrow D$，$DE\rightarrow A$，R 属于 $3NF$ 还是 $BCNF$？

答：因为 A、B、C、D、E 都是主属性，所以 R 是 $3NF$。

因为所有函数依赖的决定因素 A、BC、DE 都不含码，R 不是 $BCNF$。

7. 下面的结论哪些是正确的？哪些是错误的？对于错误的结论请给出理由或给出一个反例说明之。

① 任何一个二目关系都是属于 3NF 的。 (√)

② 任何一个二目关系都是属于 BCNF 的。 (√)

③ 任何一个二目关系都是属于 4NF 的。　(√)

解析：

$R(X,Y)$ 如果 $X\rightarrow\rightarrow Y$，即 X、Y 之间存在平凡的多值依赖，R 属于 4NF。

④ 当且仅当函数依赖 $A\rightarrow B$ 在 R 上成立，关系 $R(A,B,C)$ 等于其投影 $R1(A,B)$ 和 $R2(A,C)$ 的连接。　(×)

解析：

当 $A\rightarrow B$ 在 R 上成立，关系 $R(A,B,C)$ 等于其投影 $R1(A,B)$ 和 $R2(A,C)$ 的连接，反之则不然。

正确的应该是：

当且仅当多值依赖 $A\rightarrow\rightarrow B$ 在 R 上成立，关系 $R(A,B,C)$ 等于其投影 $R1(A,B)$ 和 $R2(A,C)$ 的连接（参见《概论》定理 6.6）。

⑤ 若 $R.A\rightarrow R.B$，$R.B\rightarrow R.C$，则 $R.A\rightarrow R.C$　(√)

⑥ 若 $R.A\rightarrow R.B$，$R.A\rightarrow R.C$，则 $R.A\rightarrow R.(B,C)$　(√)

⑦ 若 $R.B\rightarrow R.A$，$R.C\rightarrow R.A$，则 $R.(B,C)\rightarrow R.A$　(√)

⑧ 若 $R.(B,C)\rightarrow R.A$，则 $R.B\rightarrow R.A$，$R.C\rightarrow R.A$　(×)

反例：关系模式 $SC(SNO,CNO,G)$

$$(SNO,CNO)\rightarrow G,\text{但是 } SNO\nrightarrow G,CNO\nrightarrow G$$

8. 证明：

① 如果 R 是 BCNF 关系模式，则 R 是 3NF 关系模式，反之则不然。

② 如果 R 是 3NF 关系模式，则 R 一定是 2NF 关系模式。

证：

① 如果 R 是 BCNF 关系模式，则 R 是 3NF 关系模式，反之则不然。

a. 如果 R 是 BCNF 关系模式，则 R 是 3NF 关系模式。

证明：用反证法。

设关系 $R\in$BCNF，但 $R\notin$3NF。

则关系 R 中存在候选码 X，属性组 Y 和非主属性 $Z(Z\not\subseteq Y)$，满足

$$X\rightarrow Y,Y\nrightarrow X,Y\rightarrow Z$$

由于 $Y\nrightarrow X$，因此 Y 不包含候选码。

即 $Y\rightarrow Z$ 函数依赖的决定因素 Y 不包含候选码与 $R\in$BCNF 相矛盾。

所以如果 $R\in$BCNF，则 $R\in$3NF。

b. R 是 3NF 关系模式，但 R 不一定是 BCNF 关系模式。

证明：对于关系模式 $STJ(S,T,J)$，S 表示学生，T 表示教师，J 表示课程。每一教师只教一门课。每门课有若干教师，某一学生选定某门课，就对应一个固定的教师。由语义可得到如下的函数依赖：$(S,J)\rightarrow T$；$(S,T)\rightarrow J$；$T\rightarrow J$。

这里(S,J)、(S,T)都是候选码。

STJ是3NF,因为没有任何非主属性对码传递依赖或部分依赖。但STJ不是BCNF关系,因为T是决定因素,而T不包含码。

② 如果R是3NF关系模式,则R一定是2NF关系模式。

证明:用反证法。

设关系$R \in 3NF$,但$R \notin 2NF$。

则必然存在一个非主属性Z,不完全函数依赖于码。

因此,存在候选码X的真子集Y,$Y \subset X$,$Y \rightarrow Z$。

而由于Y是X的真子集,因此$Y \not\rightarrow X$;同时由于Y是主属性,Z不是主属性,因此$Z \not\subseteq Y$。

综上,存在候选码X,属性组Y,非主属性$Z(Z \not\subseteq Y)$,有$X \rightarrow Y$,$Y \not\rightarrow X$,$Y \rightarrow Z$与$R \in 3NF$相矛盾。

所以如果$R \in 3NF$,则$R \in 2NF$。

6.3 补充习题

1. 考虑关系模式$R(A,B,C,D)$,写出满足下列函数依赖时R的码,并给出R属于哪种范式(1NF、2NF、3NF或BCNF)。

① $B \rightarrow D, AB \rightarrow C$;

② $A \rightarrow B, A \rightarrow C, D \rightarrow A$;

③ $BCD \rightarrow A, A \rightarrow C$;

④ $B \rightarrow C, B \rightarrow D, CD \rightarrow A$;

⑤ $ABD \rightarrow C$。

2. 考虑属性集$ABCDEF$和函数依赖集$\{AB \rightarrow C, B \rightarrow D, BC \rightarrow E, AC \rightarrow D, E \rightarrow F, CD \rightarrow A\}$,对下面每个属性集,回答下面两个问题:a. 写出在属性集上的函数依赖集,说明是否是最小覆盖;b. 指出属于哪种范式(1NF、2NF、3NF或BCNF)。

① ABC

② $ABCD$

③ $BCDE$

④ $CDEF$

⑤ CDF

3. 对于属性集$ABCDEF$和函数依赖集$\{A \rightarrow BC, CD \rightarrow E, B \rightarrow D, BE \rightarrow F, EF \rightarrow A\}$,说明下列的分解 a. 是否是无损连接分解;b. 是否保持函数依赖。

① $\{ABCD, EFA\}$

② $\{ABC, BD, BEF\}$

4. 关系 R 有属性 $ABCD$，包括如下记录：

A	B	C	D
2	4	3	3
2	3	5	3
1	4	3	2
3	1	2	2

指出下列函数依赖或多值依赖在 R 上是否成立：

① $A \rightarrow B$

② $A \rightarrow BC$

③ $A \rightarrow\!\!\rightarrow BC$

④ $B \rightarrow C$

⑤ $BC \rightarrow\!\!\rightarrow A$

⑥ $C \rightarrow\!\!\rightarrow D$

5. 关系模式 R（员工编号，日期，零件数，部门名称，部门经理），表示某个工厂里每个员工的日生产零件数以及员工所在的部门和经理信息。

假设：每个员工每天只有一个日生产零件数，每个员工只在一个部门工作，每个部门只有一个经理，那么：

① 写出模式 R 的基本函数依赖和码；

② R 是否是 2NF，如果不是，把 R 分解成 2NF；

③ 进一步将 R 分解成 3NF。

6. 对于关系模式 R（会议，主持人，时间，会议室，会员，职务），假设一个会议有唯一的一个主持人；在一个时间地点只能召开一个会议；在给定时间一个主持人只能在一个会议室；在给定时间一个会员只能在一个会议室；一个会员在一个会议中只能有一个职务。按照语义可以得到 R 的函数依赖 $F=\{$会议→主持人，(时间，会议室)→会议，(时间，主持人)→会议室，(时间，会员)→会议室，(会议，会员)→职务$\}$

① 写出 R 的所有码；

② 说明给出的函数依赖集 F 是否极小函数依赖集；

③ 将 R 分解成具有无损连接和保持函数依赖的 3NF，判断是否有违反 BCNF 的关系。

7. 对于下列各个关系模式和依赖，回答是否是 4NF，若不是则将关系分解为满足 4NF 的关系：

① $R(A,B,C)$，存在如下依赖：$A \rightarrow\!\!\rightarrow B$ 和 $A \rightarrow C$；

② $R(A,B,C,D)$，存在如下依赖：$A \rightarrow\!\!\rightarrow C$ 和 $C \rightarrow\!\!\rightarrow BD$。

6.4 补充习题答案

1.

① R 的码为 AB；R 是 1NF。

② R 的码为 D；R 是 2NF。

③ R 的码为 BCD、ABD；R 是 3NF。

④ R 的码为 B；R 是 2NF。

⑤ R 的码为 ABD；R 是 BCNF。

2.

① ABC：函数依赖集为 $AB\rightarrow C$，是最小覆盖；是 BCNF。

解析：可以求得码 $=AB$，决定因素 AB 是码，所以是 BCNF。

② $ABCD$：函数依赖集为 $AB\rightarrow C$，$B\rightarrow D$，$AC\rightarrow D$，$CD\rightarrow A$，是最小覆盖；是 1NF。

解析：码 $=AB$，因为 $B\rightarrow D$，存在 非主属性 D 对码 AB 的部分函数依赖。

③ $BCDE$：函数依赖集为 $B\rightarrow D$，$BC\rightarrow E$，是最小覆盖；是 1NF。

解析：码 $=BC$，因为 $B\rightarrow D$，存在 非主属性 D 对码 BC 的部分函数依赖。

④ $CDEF$：函数依赖集为 $E\rightarrow F$，是最小覆盖；是 1NF。

解析：码 $=CDE$，因为 $E\rightarrow F$，存在 非主属性 F 对码 CDE 的部分函数依赖。

⑤ CDF：没有函数依赖，是最小覆盖；是 BCNF。

解析：码 $=CDF$，没有函数依赖，是 BCNF。

3.

① $U_1=ABCD$，$U_2=EFA$，是无损连接分解，没有保持函数依赖。

解析：$U=ABCDEF$，根据函数依赖集 $\{A\rightarrow BC, CD\rightarrow E, B\rightarrow D, BE\rightarrow F, EF\rightarrow A\}$，可以求得码 $=A$ 和 BE，所以有 $A\rightarrow EF$ 成立。

因为 $U_1\cap U_2=A$，$U_2-U_1=EF$，即 $U_1\cap U_2\rightarrow U_2-U_1$，所以分解 $U_1=ABCD$、$U_2=EFA$ 是无损连接分解。

分解 $U_1=ABCD$ 上面的函数依赖集为 $\{A\rightarrow BC, B\rightarrow D\}$，分解 $U_2=EFA$ 上面的函数依赖集为 $\{EF\rightarrow A\}$，丢失了 $CD\rightarrow E$、$BE\rightarrow F$，因此没有保持函数依赖。

② 解析：

对于分解 $U_1=ABC \quad F_1=\{A\rightarrow BC\}$

$U_2=BD \quad F_2=\{B\rightarrow D\}$

$U_3=BEF, F_3=\{BE\rightarrow F\}$

构造初始表：

A	B	C	D	E	F
$a1$	$a2$	$a3$	$b14$	$b15$	$b16$
$b21$	$a2$	$b23$	$a4$	$b25$	$b26$
$b31$	$a2$	$b33$	$b34$	$a5$	$a6$

第一遍扫描由 $B\rightarrow D$ 可将 $b14$、$b34$ 改为 $a4$：

A	B	C	D	E	F
$a1$	$a2$	$a3$	$a4$	$b15$	$b16$
$b21$	$a2$	$b23$	$a4$	$b25$	$b26$
$b31$	$a2$	$b33$	$a4$	$a5$	$a6$

第二遍扫描表无变化，没有出现一行 $a1$、$a2$、$a3$、$a4$、$a5$、$a6$，因此不是无损连接分解。分解也没有保持函数依赖，因为丢失了 $CD\rightarrow E$，$EF\rightarrow A$。

4.

① 不成立

② 不成立

③ 不确定

④ 不确定

⑤ 不成立

⑥ 不成立

解析：针对关系的某个值，可以判断某个函数依赖和多值依赖不成立，不能确定它们成立。

5. 答：

① 根据给出的语义，R（员工编号，日期，零件数，部门名称，部门经理）的函数依赖有：{（员工编号，日期）→零件数，员工编号→部门名称，部门名称→部门经理}。

码：（员工编号，日期）。

② R 不是 2NF，因为（员工编号，日期）是码，存在非主属性-部门名称对码的部分依赖（员工编号→部门名称）。

分解为 2NF：$R1$（员工编号，日期，零件数）

$R2$（员工编号，部门名称，部门经理）

③ 分解为 3NF：

$R1$（员工编号，日期，零件数）

$R2$（员工编号，部门名称）

$R3$（部门名称，部门经理）

6.

① $R(U,F)$，$U=$（会议，主持人，时间，会议室，会员，职务）

R 的码为（时间，会员），因为（时间，会员）$_F^+=U$。

② 函数依赖集 F 是极小函数依赖集。理由如下：

首先，所有函数依赖的右边都已经是单个属性；

其次，没有一个函数依赖在删除之后可以被剩下的函数依赖集逻辑蕴含；

最后，对于右部是属性组的函数依赖，（时间，会议室）→会议，（时间，主持人）→会议室，（时间，会员）→会议室，（会议，会员）→职务，考察它们的子集：时间→会议，会议室→会议，时间→会议室，主持人→会议室，会员→会议室，会议→职务，会员→职务，都无法由函数依赖集导出。所以，函数依赖集 F 是极小函数依赖集。

③ 根据合成法（《概论》算法 6.3）可以将 R 分解为：

U_1（会议，主持人），U_2（时间，会议室，会议），U_3（时间，主持人，会议室），

U_4（时间，会员，会议室），U_5（会议，会员，职务），

其中 U_4（时间，会员，会议室）包含码（时间，会员），因此不需要添加新的关系，此分解是具有无损连接和保持函数依赖的 3NF。

分解后的每个关系具有一个函数依赖，每个决定因素是码，因此也是 BCNF。

7.

① R 的码是 AB，因为 $A\rightarrow\rightarrow B$，而 A 不是码，所以 R 不是 4NF；分解为 AB、AC，满足 4NF。

② R 的码是 $ABCD$，因此 R 不是 4NF；将 R 分解为 AC、CBD，满足 4NF。

第 7 章 数据库设计

《概论》第 7 章讲解数据库设计的方法和步骤。

《概论》把数据库设计作为一项工程来讲解和讨论。因为大型数据库的设计和开发是一项庞大的工程,是涉及多学科的综合性技术。数据库设计的重要性在于:数据库设计技术是信息系统开发和建设中的核心技术。

《概论》在讲解数据库设计时力求把数据库设计和应用系统设计相结合,把结构(数据)设计和行为(处理)设计密切结合起来。

许多教材把数据库设计简单地描述为:如何把一组数据储存在数据库中,为这些数据设计一个合适的逻辑结构。即如何设计关系模式,以及各个关系模式中的属性。这仅仅是数据库逻辑设计的内容。

在数据库设计的各个阶段,人们都研究和开发了各种数据库设计工具。关系数据理论是进行数据库逻辑设计的有力工具。

学习本章要把软件工程的思想、方法具体运用到数据库设计中。必须理论联系实际,能够完成一个实际部门的数据库应用系统设计全过程。

7.1 基本知识点

本章讲解数据库设计方法和技术,内容的实践性较强。

① 需要了解的:数据库设计的特点,数据库物理设计的内容和评价,数据库的实施和维护。

② 需要牢固掌握的:数据库设计的基本步骤,数据库设计过程中数据字典的内容,数据库设计各个阶段的具体设计内容、设计描述、设计方法等。

③ 需要举一反三的:E-R 图的设计,E-R 图向关系模型的转换。

④ 难点:技术上的难点是 E-R 图的设计,数据模型的优化。真正的难点是理论与实际的结合。读者一般缺乏实际经验,缺乏对实际问题解决的能力,特别是缺乏应用领域的知识。而数据库设计需要设计人员对应用环境、专业业务有具体深入的了解,这样才能设计出符合具体领域要求的数据库及其应用系统。希望读者在完成本章习题的基础上要认真完成大作

业。体会这些要点,从而真正掌握本章讲解的知识、方法和技术。

7.2 习题解答和解析

1. 试述数据库设计过程。

这里只概要列出数据库设计过程的6个阶段:

① 需求分析;

② 概念结构设计;

③ 逻辑结构设计;

④ 数据库物理设计;

⑤ 数据库实施;

⑥ 数据库运行和维护。

这是一个完整的实际数据库及其应用系统的设计过程。不仅包括设计数据库本身,还包括数据库的实施、数据库的运行和维护。

设计一个完善的数据库应用系统往往是上述6个阶段的不断反复。

解析:希望读者能够认真阅读《概论》第7.1节的内容,了解并掌握数据库设计过程。

2. 试述数据库设计过程中形成的数据库模式。

答:

数据库设计的不同阶段形成数据库的各级模式,即:

① 在概念结构设计阶段形成独立于机器特点、独立于各个DBMS产品的概念模式,在本篇中就是E-R图。

② 在逻辑结构设计阶段将E-R图转换成具体的数据库产品支持的数据模型,如关系模型,形成数据库逻辑模式;然后在基本表的基础上再建立必要的视图(view),形成数据的外模式。

③ 在物理结构设计阶段,根据DBMS特点和处理的需要进行物理存储安排,建立索引,形成数据库内模式。

读者可以参考《概论》图7.4。图中概念模式是面向用户和设计人员的,属于概念模型的层次;逻辑模式、外模式、内模式是DBMS支持的模式,属于数据模型的层次,可以在DBMS中加以描述和存储。

3. 需求分析阶段的设计目标是什么?调查的内容是什么?

答:

需求分析阶段的设计目标是通过详细调查现实世界要处理的对象(组织、部门、企业等),充分了解原系统(手工系统或计算机系统)工作概况,明确用户的各种需求,然后在此基础上确定新系统的功能。

调查的内容是“数据”和“处理”,即获得用户对数据库的如下要求:

① 信息要求。指用户需要从数据库中获得信息的内容与性质。由信息要求可以导出数据要求,即在数据库中需要存储哪些数据。

② 处理要求。指用户要完成什么处理功能,对处理的响应时间有什么要求,处理方式是批处理还是联机处理。

③ 安全性与完整性要求。

详细内容可以参考《概论》第 7.2 节。

4. 数据字典的内容和作用是什么?

答:

数据字典的内容通常包括数据项、数据结构、数据流、数据存储和处理过程 5 个部分。其中数据项是数据的最小组成单位,若干个数据项可以组成一个数据结构。数据字典通过对数据项和数据结构的定义来描述数据流、数据存储的逻辑内容。

数据字典的作用:数据字典是关于数据库中数据的描述,在需求分析阶段建立,是下一步进行概念设计的基础,并在数据库设计过程中不断修改、充实和完善。

详细内容参考《概论》第 7.2.3 小节。注意,数据库设计阶段形成的数据字典与第 12 章 DBMS 中的数据字典不同,后者是 DBMS 关于数据库中数据的描述。当然两者是有联系的。

5. 什么是数据库的概念结构? 试述其特点和设计策略。

答:

概念结构是信息世界的结构,即概念模型,其主要特点是:

① 能真实、充分地反映现实世界,包括事物和事物之间的联系,能满足用户对数据的处理要求,是对现实世界的一个真实模型。

② 易于理解,从而可以用它和不熟悉计算机的用户交换意见,用户的积极参与是数据库设计成功的关键。

③ 易于更改,当应用环境和应用要求改变时,容易对概念模型修改和扩充。

④ 易于向关系、网状、层次等各种数据模型转换。

概念结构的设计策略通常有 4 种:

① 自顶向下。即首先定义全局概念结构的框架,然后逐步细化。

② 自底向上。即首先定义各局部应用的概念结构,然后将它们集成起来,得到全局概念结构。

③ 逐步扩张。首先定义最重要的核心概念结构,然后向外扩充,以滚雪球的方式逐步生成其他概念结构,直至总体概念结构。

④ 混合策略。即将自顶向下和自底向上相结合,用自顶向下策略设计一个全局概念结构的框架,以它为骨架集成由自底向上策略中设计的各局部概念结构。

6. 定义并解释概念模型中以下术语:

实体,实体型,实体集,属性,码,实体-联系图(E-R 图)

答：

实体：客观存在并可以相互区分的事物叫实体。

实体型：具有相同属性的实体具有相同的特征和性质，用实体名及其属性名集合来抽象和刻画同类实体称为实体型。

实体集：同型实体的集合称为实体集。

属性：实体所具有的某一特性，一个实体可由若干个属性来刻画。

码：唯一标识实体的属性集称为码。

实体-联系图（E-R 图）：描述实体型、属性和联系的一种方法，其中：

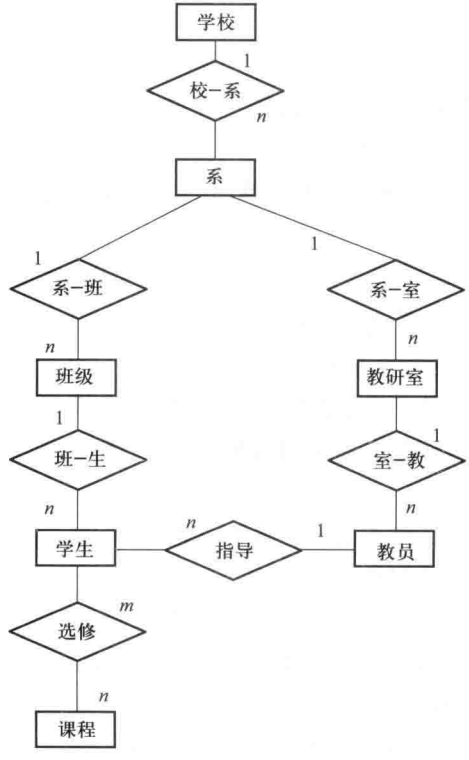
① 实体型用矩形表示，矩形框内写明实体名。

② 属性用椭圆形表示，并用无向边将其与相应的实体型连接起来。

③ 联系用菱形表示，菱形框内写明联系名，并用无向边分别与有关实体连接起来，同时在无向边旁标上联系的类型（1∶1，1∶n 或 m∶n）。

7. 学校中有若干系，每个系有若干班级和教研室，每个教研室有若干教员，其中有的教授和副教授每人各带若干研究生，每个班有若干学生，每个学生选修若干课程，每门课可由若干学生选修。请用 E-R 图画出此学校的概念模型。

答：



解析：

在画 E-R 图时，读者可以按照习题中对问题的描述一步一步画出每一句话中涉及的实体，再根据给出的实际语义画出实体之间的联系。例如，每个教研室有若干教员，每个班有若干学生，可以画出教研室和教员，班级和学生之间一对多的联系，从“有的教授和副教授每人各带若干研究生”，一个研究生一般指定一个导师，这是通常的规则，所以可以画出教员和学生之间一对多的联系。

E-R 图中各实体的属性假设为：（简便起见，未用图表示）

系：系编号，系名；

班级：班级编号，班级名；

教研室：教研室编号，教研室；

学生：学号，姓名，学历；

课程：课程编号，课程名；

教员：职工号，姓名，职称

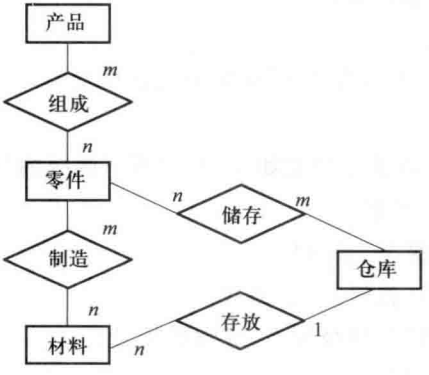
各联系的属性为：

选修：成绩；

其他联系无属性。

8. 某工厂生产若干产品，每种产品由不同的零件组成，有的零件可用在不同的产品上。这些零件由不同的原材料制成，不同零件所用的材料可以相同。这些零件按所属的不同产品分别放在仓库中，原材料按照类别放在若干仓库中。请用 E-R 图画出此工厂产品、零件、材料、仓库的概念模型。

答：



解析：

对实体之间联系的语义描述有时不是直截了当的，需要从对现实世界的整体描述中进行分析，导出实体之间的某种联系。就如本题中，“零件和仓库的联系”就要从以下描述中分

析:“零件按所属的不同产品分别放在仓库中”。因为一个产品由多种零件组成,所以一个仓库中放多种零件。反过来,一种零件是放在一个仓库还是多个仓库中呢?因为一种零件可以用在多种产品上,这些零件按所属的不同产品分别放在仓库中,于是可以知道一种零件可以放在多个仓库中。所以零件和仓库之间是多对多的联系。

“材料和仓库的联系”则根据“原材料按照类别放在若干仓库”这句话就可以得出:一个仓库中放多种材料,而一种材料只放在一个仓库中,所以仓库和材料之间是一对多的联系。

各实体的属性为:(简便起见,未用图表示)

产品:产品号,产品名

零件:零件号,零件名

原材料:原材料号,原材料名,类别

仓库:仓库号,仓库名

各联系的属性为:

产品组成:使用零件量

零件制造:使用原材料量

零件存储:存储量

材料存放:存放量

9. 什么是数据库的逻辑结构设计?试述其设计步骤。

答:

数据库的逻辑结构设计就是把概念结构设计阶段设计好的基本 E-R 图转换为与选用的 DBMS 产品所支持的数据模型相符合的逻辑结构。设计步骤为:

① 将概念结构转换为关系模型;

② 对数据模型进行优化。

10. 试把习题 7 和习题 8 中的 E-R 图转换为关系模型。

答:

习题 7 中的 E-R 图转换的关系模型如下,其中有下画线的属性是主码属性。

系(系编号,系名,学校名)

班级(班级编号,班级名,系编号)

教研室(教研室编号,教研室,系编号)

学生(学号,姓名,学历,班级编号,导师职工号)

课程(课程编号,课程名)

教员(职工号,姓名,职称,教研室编号)

选课(学号,课程编号,成绩)

习题 8 中的 E-R 图转换的关系模型如下,其中有下画线的属性是主码属性。

产品(产品号,产品名,仓库号)

零件(零件号,零件名)

原材料(原材料号,原材料名,类别,仓库号,存放量)

仓库(仓库号,仓库名)

产品组成(产品号,零件号,使用零件量)

零件组成(零件号,原材料号,使用原材料量)

零件储存(零件号,仓库号,存储量)

11. 试用规范化理论中有关范式的概念分析习题 7 中所设计的关系模型中各个关系模式的候选码,它们属于第几范式? 会产生什么更新异常?

答:

习题 7 中设计的各个关系模式的码都用下画线注明,这些关系模式都只有一个码,且都是唯一决定的因素,所以都属于 BCNF。不会产生更新异常现象。

12. 规范化理论对数据库设计有什么指导意义?

答:

规范化理论为数据库设计人员判断关系模式优劣提供了理论标准,可用以指导关系数据模型的优化,用来预测模式可能出现的问题,为设计人员提供了自动产生各种模式的算法工具,使数据库设计工作有了严格的理论基础。可参考《概论》第 7.4.2 小节有关数据模型的优化内容。

13. 试述数据库物理设计的内容和步骤。

答:

数据库在物理设备上的存储结构与存取方法称为数据库的物理结构,它依赖于给定的 DBMS。为一个给定的逻辑数据模型选取一个最适合应用要求的物理结构,就是数据库的物理设计的主要内容。

数据库的物理设计步骤通常分为两步:

① 确定数据库的物理结构,在关系数据库中主要指存取方法和存储结构;

② 对物理结构进行评价,评价的重点是时间和空间效率。

详细内容可以参考《概论》第 7.5 节。

14. 数据输入在实施阶段的重要性是什么? 如何保证输入数据的正确性?

答:

数据库是用来对数据进行存储、管理与应用的,因此在实施阶段必须将原有系统中的历史数据输入到数据库。数据量一般都很大,而且数据来源于部门中的各个不同的单位。数据的组织方式、结构和格式都与新设计的数据库系统有相当的差距,组织数据录入就要将各类源数据从各个局部应用中抽取出来,分类转换,最后综合成符合新设计的数据库结构的形式,输入数据库。因此这样的数据转换、组织入库的工作是相当费力费时的。特别是原系统是手

工数据处理系统时，各类数据分散在各种不同的原始表格、凭证、单据之中，数据输入工作量更大。

保证输入数据正确性的方法：为提高数据输入工作的效率和质量，应该针对具体的应用环境设计一个数据录入子系统，由计算机来完成数据入库的任务。在源数据入库之前要采用多种方法对它们进行检验，以防止不正确的数据入库。

15. 什么是数据库的再组织和重构造？为什么要进行数据库的再组织和重构造？

答：

数据库的再组织是指按原设计要求重新安排存储位置、回收垃圾、减少指针链等，以提高系统性能。

数据库的重构造则是指部分修改数据库的模式和内模式，即修改原设计的逻辑和物理结构。数据库的再组织是不修改数据库的模式和内模式的。

进行数据库的再组织和重构造的原因：

数据库运行一段时间后，由于记录不断增、删、改，会使数据库的物理存储情况变坏，降低了数据的存取效率，数据库性能下降，这时DBA就要对数据库进行重组织。DBMS一般都提供数据重组织用的实用程序。

数据库应用环境常常发生变化，如增加新的应用或新的实体，取消了某些应用，有的实体与实体间的联系也发生了变化等，使原有的数据库设计不能满足新的需求，需要调整数据库的模式和内模式。这就要进行数据库重构造。

7.3 补充习题

1. 选择题

(1) 数据库外模式是在下列哪个阶段设计(　　)。

A. 数据库概念结构设计　　B. 数据库逻辑结构设计

C. 数据库物理设计　　D. 数据库实施和维护

(2) 生成DBMS系统支持的数据模型是在下列哪个阶段完成(　　)。

A. 数据库概念结构设计　　B. 数据库逻辑结构设计

C. 数据库物理设计　　D. 数据库实施和维护

(3) 根据应用需求建立索引是在下列哪个阶段完成(　　)。

A. 数据库概念结构设计　　B. 数据库逻辑结构设计

C. 数据库物理设计　　D. 数据库实施和维护

(4) 员工性别的取值，有的为“男”、“女”，有的为“1”、“0”，这种情况属于(　　)。

A. 属性冲突　　B. 命名冲突

C. 结构冲突　　D. 数据冗余

2. 填空题

(1) 数据库设计方法包括_____、_____、_____、_____和统一建模语言(UML)方法等。

(2) 数据库设计的基本步骤包括需求分析、_____、_____、_____、数据库实施、数据库运行和维护等。

(3) 集成局部 E-R 图要分两个步骤,分别是_____和_____。

(4) 数据库常见的存取方法主要有_____、_____和 Hash 方法。

3. 问答题

(1) 在进行概念结构设计时,将事物作为属性的基本准则是什么。

(2) 将 E-R 图转换为关系模式时,可以如何处理实体型间的联系。

4. 综合题

(1) 某商场可以为顾客办理会员卡,每个顾客只能办理一张会员卡,顾客信息包括顾客姓名、地址、电话、身份证号,会员卡信息包括号码、等级、积分,给出该系统的 E-R 图。

(2) 按照下列说明修改题(1)中的要求,分别给出相应的 E-R 图:

① 顾客具有多个地址和多个电话号码,地址包括省、市、区、街道,电话号码包括区号、号码;

② 顾客具有多个地址,每个地址具有多个电话号码,地址包括省、市、区、街道,电话号码包括区号、号码。

(3) 某数据库记录乐队、成员和歌迷的信息,乐队包括名称、多个成员、一个队长,队长也是乐队的成员,成员包括名字、性别,歌迷包括名字、性别、喜欢的乐队、喜欢的成员。

① 画出基本的 E-R 图;

② 修改 E-R 图,使之能够表示成员在乐队的工作记录,包括进入乐队时间以及离开乐队时间。

(4) 考虑某个 IT 公司的数据库信息:

① 部门具有部门编号、部门名称、办公地点等属性;

② 部门员工具有员工编号、姓名、级别等属性,员工只在一个部门工作;

③ 每个部门有唯一一个部门员工作为部门经理;

④ 实习生具有实习编号、姓名、年龄等属性,只在一个部门实习;

⑤ 项目具有项目编号、项目名称、开始日期、结束日期等属性;

⑥ 每个项目由一名员工负责,由多名员工、实习生参与;

⑦ 一名员工只负责一个项目,可以参与多个项目,在每个项目具有工作时间比;

⑧ 每个实习生只参与一个项目。

画出 E-R 图,并将 E-R 图转换为关系模型(包括关系名、属性名、码和完整性约束条件)。

7.4　补充习题答案

1. 选择题

(1)	(2)	(3)	(4)
B	B	C	A

2. 填空题

(1) 新奥尔良方法　基于 E-R 模型的方法　3NF 的设计方法　面向对象的设计方法

(2) 概念结构设计　逻辑结构设计　物理结构设计

(3) 合并　修改和重构

(4) 索引　聚簇

3. 问答题

(1) 在进行概念结构设计时,将事物作为属性的基本准则是什么。

答:

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

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

(2) 将 E-R 图转换为关系模式时,可以如何处理实体型间的联系。

答:

① 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;

② 一个 1∶n 联系可以转换为一个独立的关系模式,也可以与 n 端对应的关系模式合并;

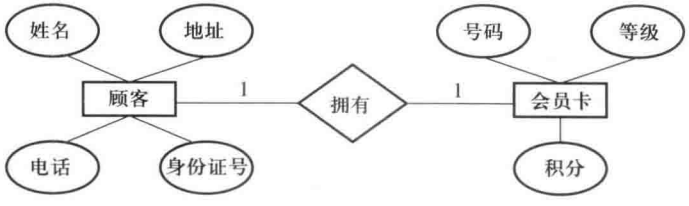
③ 一个 m∶n 联系可以转换为一个关系模式;

④ 3 个或 3 个以上实体间的一个多元联系可以转换为一个关系模式;

⑤ 具有相同码的关系模式可合并。

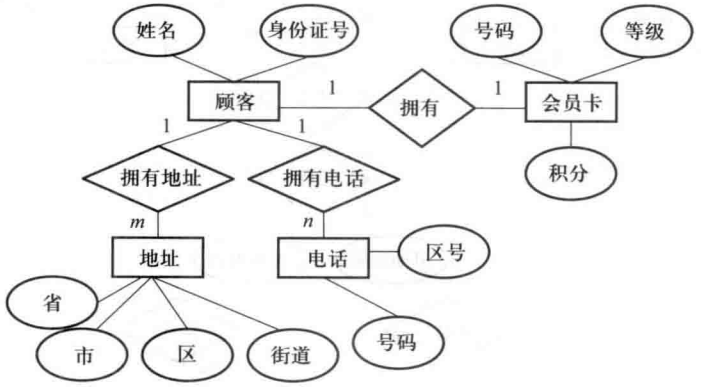
4. 综合题

(1)

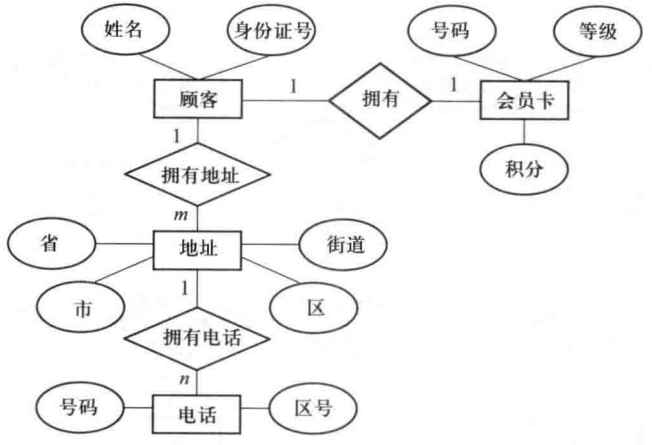


(2)

①

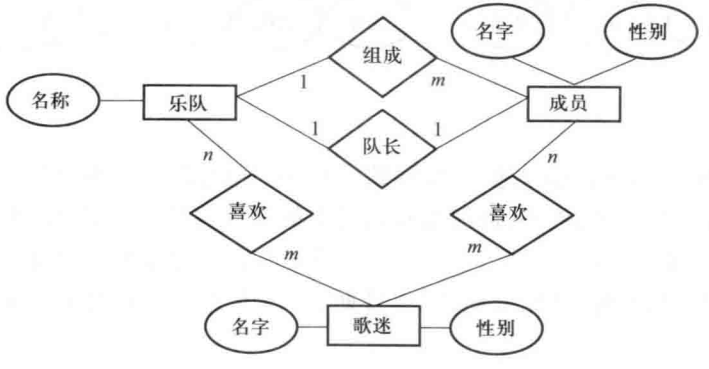

姓名
身份证号
号码
等级
顾客
1
拥有
1
会员卡
1
1
积分
拥有地址
拥有电话
m
n
地址
电话
区号
省
市
区
街道
号码

②

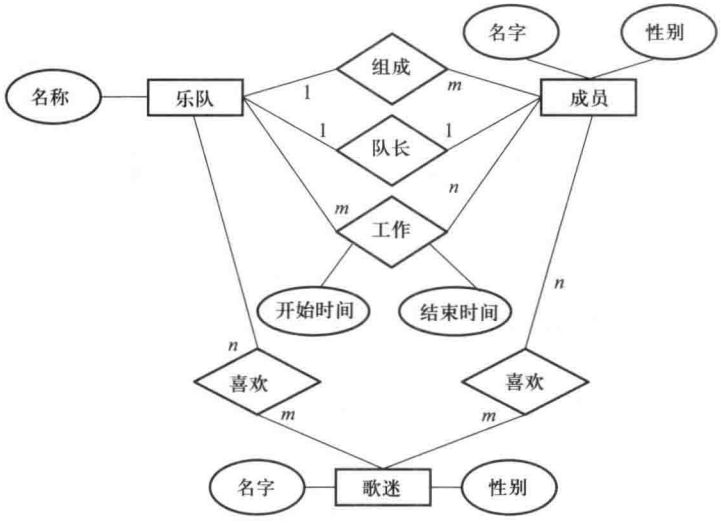

姓名
身份证号
号码
等级
顾客
1
拥有
1
会员卡
1
积分
拥有地址
m
省
地址
街道
市
1
区
拥有电话
n
号码
电话
区号

(3)

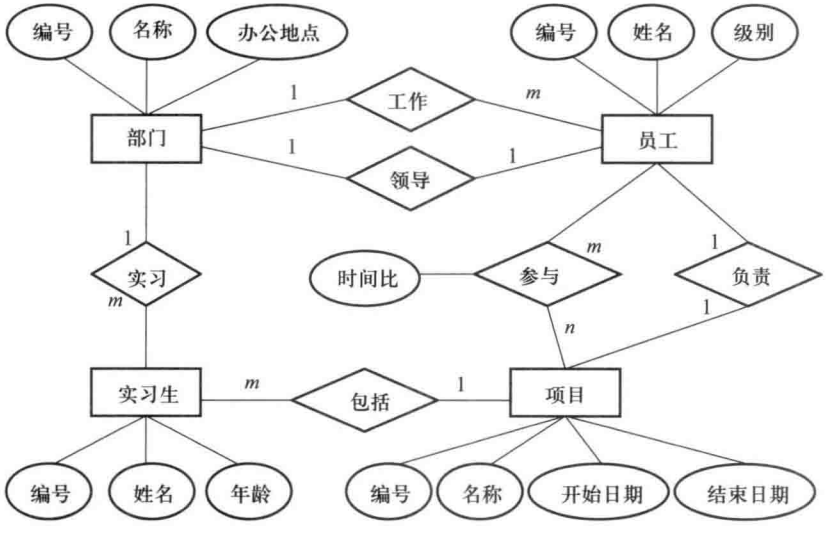
①


名字
性别
组成
名称
乐队
1
m
成员
1
队长
1
n
n
喜欢
喜欢
m
m
名字
歌迷
性别

②



(4)



关系模型为：

部门(编号,名称,办公地点,经理编号),部门的经理编号参照员工的编号；

员工(编号,姓名,级别,部门编号),员工的部门编号参照部门的编号；

实习生(编号,姓名,年龄,部门编号),实习生的部门编号参照部门的编号；

项目(编号,名称,开始日期,结束日期,负责人编号),项目的负责人编号参照员工的编号；

实习参与(实习生编号,项目编号),实习生编号、项目编号分别参照实习生的编号、项目的编号;

员工参与(员工编号,项目编号,时间比),员工编号、项目编号分别参照员工的编号、项目的编号,且一个员工的所有时间比相加不超过 100%。

第 8 章 数据库编程

在数据库应用系统的开发中常常使用编程方法对数据库进行操纵。本章讲解这些编程技术涉及的概念和使用的方法。主要包括嵌入式 SQL、游标的概念;SQL 的过程化扩展 PL/SQL、存储过程和自定义函数;使用 ODBC、OLEDB、JDBC 设计开发数据库应用程序的方法。

因为 OLEDB 编程、JDBC 编程与 ODBC 编程的思路基本相同,限于篇幅,《概论》主要讲解 ODBC,对于 OLEDB 和 JDBC 编程只做了简单介绍,读者可以通过上机实验了解这些技术。

8.1 基本知识点

本章讲解开发数据库应用系统的各种编程方法,内容的实践性较强。希望读者多上机实验。通过实践培养选择不同方法和技术开发数据库应用程序的能力。

① 需要了解的:了解 SQL 编程技术可以有效克服 SQL 实现复杂应用方面的不足,提高应用系统和 RDBMS 间的互操作性。

② 需要牢固掌握的:掌握嵌入 SQL 中游标的概念和使用方法;掌握PL/SQL 和存储过程的基本概念,基本结构,语句语法和用法。了解使用 ODBC 开发应用系统的体系结构,掌握 ODBC API 和 ODBC 的应用程序的工作流程。

③ 需要举一反三的:能够在实际安装的 RDBMS 上通过编程的方式开发应用程序,完成对数据库的各种操作。能够使用 ODBC 来进行数据库应用程序的设计,使设计的应用系统可移植性好,并且能同时访问不同的数据库,共享数据资源。

8.2 习题解答和解析

1. 使用嵌入式 SQL 对学生-课程数据库中的表,完成下面功能:

① 查询某一门课程的信息。要查询的课程由用户在程序运行过程中指定,放在主变量中。

② 查询选修某一门课程的选课信息,要查询的课程号由用户在程序运行过程中指定,放

在主变量中,然后根据用户的要求修改其中某些记录的成绩字段。

① 解析:

这里给出了完成查询的嵌入式 SQL 语句,完整的程序可参考《概论》书[例 8.1],再根据实际使用的 RDBMS 的用户手册完成。

```
/*定义主变量:HCno,:HCname,:HCpno,:HCcredit,:givencno*/
/*读入用户输入的课程号,放入主变量:givencno*/
EXEC SQL SELECT Cno,Cname,Cpno,Ccredit
         INTO:HCno,:HCname,:HCpno,:HCcredit   /*把 SQL 查询结果放入主变量中*/
         FROM Course
         WHERE Cno=:givencno;
```

② 解析:

这里给出了完成查询要使用的游标以及嵌入式 SQL 语句,完整的程序可参考《概论》书[例 8.1],再根据实际使用的 RDBMS 的用户手册完成。

```
/*定义主变量 :HSno,:HCno,:HGrade,:givencno,:NEWGrade*/
/*在程序中输入要查询的课程号,放入主变量:givencno*/
/*定义游标 SCX,对应的 SQL 语句*/
EXEC SQL DECLARE SCX CURSOR FOR
SELECT Sno,Cno,Gradet
  FROM SC
  WHERE Cno=:givencno;        /*要查询的课程号放在主变量 givencno 中*/
EXEC SQL OPEN SCX;            /*打开游标 SCX,指向查询结果的第一行*/
for(;;)                       /*用循环结构逐条处理结果集中的记录*/
{ EXEC SQL FETCH SCX INTO:HSno,:HCno,:HGrade;
        /*推进游标,将当前数据放入主变量*/
        /*显示查询结果,询问用户是否要更新该条记录的选修成绩*/
        /*如果选择更新操作*/
        /*用户输入新的成绩到:NEWgrade 中,然后对当前游标指向的选修课成绩进行更新*/
  EXEC SQL UPDATE SC                  /*嵌入式 SQL 更新语句*/
       SET grade=:NEWgrade
       WHERE CURRENT OF SCX;    /*对当前游标指向的选修课成绩进行更新*/
       /*全部处理完 SC 表后终止循环*/
};
EXEC SQL CLOSE SCX;                   /*关闭游标 SCX 不再和查询结果对应*/
/*提交更新*/
```

2. 对学生-课程数据库,编写存储过程,完成下面功能:

① 统计离散数学的成绩分布情况,即按照各分数段统计人数;

② 统计任意一门课的平均成绩；

③ 将学生选课成绩从百分制改为等级制(即 A、B、C、D、E)。

答：

①

```
CREATE PROCEDURE discrete_math_grade( )
AS
DECLARE CURSOR dist FOR          /*定义游标*/
    SELECT grade FROM SC WHERE cno=
            (SELECT Cno FROM Course WHERE Cname='离散数学');
    p_100 NUMBER :=0;
    p_90 NUMBER :=0;
    p_80 NUMBER :=0;
    p_70 NUMBER :=0;
    p_60 NUMBER :=0;
    p_othersNUMBER :=0;
    p_gradeNUMBER;
BEGIN
    OPEN dist;                   /*打开游标*/
    LOOP
        FETCH dist INTO p_grade;  /*使用游标*/
        EXIT WHEN(dist%NOTFOUND);
        IF(p_grade==100) THEN
            p_100:=p_100+1;
        ELSIF(p_grade >=90) THEN
            p_90:=p_90+1;
        ELSIF(p_grade >=80) THEN
            p_80:=p_80+1;
        ELSIF(p_grade >=70) THEN
            p_70:=p_70+1;
        ELSIF(p_grade >=60) THEN
            p_60:=p_60+1;
        ELSE
            p_others:=p_others+1;
        END IF;
    END LOOP;
    CLOSE dist;                  /*关闭游标*/
END;
```

②

解析:输入任意一门课的课程名,计算选修该课程学生的平均成绩。

```
CREATE PROCEDURE avegrade( incname CHAR(40) )
AS
BEGIN
  SELECT AVG( Grade) FROM SC WHERE Cno=
          (SELECT Cno FROM Course WHERE Cname=incname);
END;
```

③

解析:扫描 SC 表,把学生选课成绩 grade 的值由百分制更新为等级制。

```
CREATE PROCEDURE gradetype( )
AS
    DECLARE CURSOR gradecursor FOR SELECT grade FROM SC;
    scgrade NUMBER;
    score CHAR(1);
BEGIN
    OPEN gradecursor;                          /*打开游标*/
    LOOP
        FETCH gradecursor INTO scgrade;        /*使用游标*/
        EXIT WHEN( gradecursor %NOTFOUND);
        IF( scgrade >=90 AND scgrade <=100) THEN
            score :='A';
        ELSIF( scgrade  >=80) THEN
            score :='B';
        ELSIF( scgrade >=70) THEN
            score :='C';
        ELSIF( scgrade >=60) THEN
            score :='D';
        ELSE
            score :='E';
        END IF;
    END LOOP;
END;
```

3. 使用 ODBC 编写应用程序来对异构数据库进行各种数据操作。

配置两个不同的数据源,编写程序连接两个不同 RDBMS 的数据源,对异构数据库进行操作。例如,将 KingbaseES 数据库的某个表中的数据转移到 SQL Server 数据库的表中。

解析：

读者可以参考《概论》[例 8.12]、[例 8.13]，再根据实际使用的 RDBMS 用户手册上机编程实现。

8.3　补 充 习 题

1. 填空题

（1）嵌入式 SQL 语句中为了和主语言语句进行区分，在 SQL 语句前加前缀_____，以_____结束。

（2）主变量可以附加一个指示变量，指示变量可以表示输入主变量是否为_____。

（3）SQL 是面向集合的，主语言是面向记录的，可以使用_____解决这一问题。

（4）存储过程经过编译、优化之后存储在_____。

（5）应用程序中访问和管理数据库的方法有_____、_____、_____、_____和 OLEDB 等。

2. 综合题

（1）对于下面的数据库模式，使用嵌入式 SQL 完成各个查询：

Teacher(Tno, Tname, Tage, Tsex)

Department(Dno, Dname, Tno)

Work(Tno, Dno, Year, Salary)

① 根据用户输入的工资，给出所有工资比这个工资高的教师姓名、年龄、性别及工资；

② 根据用户输入的工资，将所有比这个工资低的教师工资都改为与之相同。

（2）根据下面的数据库模式，写出满足要求的 PL/SQL 存储过程：

Author(Ano, Name, Age, Sex)

Book(Bno, Title, Publisher)

Write(Ano, Bno, IsFirstAuthor)

① 给定作者姓名，删除作者信息并从 Write 中删除作者写作信息；

② 给定书名，如果此书的作者只有一人，则输出此作者名字，否则返回 NULL。

8.4　补充习题答案

1. 填空题

（1）EXEC SQL　分号；

（2）空值

（3）游标

(4) 数据库服务器中

(5) 嵌入式 SQL PL/SQL ODBC JDBC

2. 综合题

(1)

① 根据用户输入的工资,给出所有工资比这个工资高的教师姓名、年龄、性别及工资。

```
void showInfo( )
{
EXEC SQL BEGIN DECLARE SECTION;          /* 主变量说明开始 */
  char Hname[20];
  int Hage;
  char Hsex[2];
  int Hsalary;
EXEC SQL END DECLARE SECTION;            /* 主变量说明结束 */
long SQLCODE;
EXEC SQL INCLUDE sqlca;
int inputsalary;
printf("Please input the salary:");
scanf("%d",&inputsalary);

EXEC SQL DECLARE showCursor CURSOR FOR  /* 定义游标 showCursor */
SELECT tname,tage,tsex,salary FROM teacher,work WHERE teacher.tno=work.tno;

EXEC SQL OPEN showCursor;                /* 打开游标 showCursor */

for(;;)
{
  EXEC SQL FETCH showCursor INTO :Hname,:Hage,:Hsex,:Hsalary;
  if(sqlca.sqlcode != 0)
      break;
  if(inputsalary < Hsalary)
      printf("name:%s,age:%d,sex:%s,salary:%d\n",Hname,Hage,Hsex,Hsalary);
}
EXEC SQL CLOSE showCursor;               /* 关闭游标 showCursor */
}
```

② 根据用户输入的工资,将所有比这个工资低的教师工资都改为与之相同。

```
void updateSalary( )
```

```
{
EXEC SQL BEGIN DECLARE SECTION;
    int Hsalary;
    int inputsalary;
EXEC SQL END DECLARE SECTION;
long SQLCODE;
EXEC SQL INCLUDE sqlca;
printf("Please input the salary:");
scanf("%d",&inputsalary);
EXEC SQL DECLARE updateCursor CURSOR FOR        /*定义游标 updateCursor */
SELECT salary FROM work;
EXEC SQL OPEN updateCursor;                     /*打开游标 updateCursor */
for(;;)
{
  EXEC SQL FETCH updateCursor INTO :Hsalary;
  if(sqlca.sqlcode !=0)
      break;
  if(inputsalary > Hsalary)
      EXEC SQL UPDATE work SET salary= :inputsalary
  WHERE CURRENT OF updateCursor;
}
EXEC SQL CLOSE updateCursor;                    /*关闭游标 updateCursor */
EXEC SQL COMMIT WORK;
}
```

(2)

① 给定作者姓名,删除作者信息并从 Write 中删除作者写作信息。

```
CREATE PROCEDURE deleteAuthor(IN authorname CHAR(20))
AS
BEGIN
  DELETE FROM write WHERE ano in(SELECT ano FROM author WHERE name=authorname);
  DELETE FROM author WHERE name=authorname;
  COMMIT;
END;
```

② 给定书名,如果此书的作者只有一人,则输出此作者名字,否则返回 NULL。

```
CREATE PROCEDURE findAuthor(IN bookname CHAR(50),OUT authorname CHAR(20))
```

```
AS DECLARE
count INT;
BEGIN
    SET authorname=NULL;
    SELECT count(*) INTO count
    FROM write
    WHERE bno in
        (SELECT bno
         FROM book
         WHERE title=bookname);
IF 1==count THEN
    SELECT name INTO authorname
    FROM author
    WHERE ano in
        (SELECT ano
         FROM write
         WHERE bno in
           (SELECT bno
            FROM book
            WHERE title=bookname)
           );
END;
```

第 9 章　关系查询处理和查询优化

第 9 章讲解关系数据库管理系统查询处理和查询优化的基本概念、方法和技术。

查询优化是 RDBMS 的内部实现技术，对于一般用户是透明的，用户不必了解RDBMS 是如何对他给出的查询语句进行优化的。

由于 RDBMS 的优化技术并不都是做得很好，对于不同的 SQL 语句、不同的数据库状态其优化效果也不一样，有些优化效果好，有些优化得不够好。因此用户有必要了解查询优化的概念，以便能够写出“好”的查询，执行效率高的语句。特别对于 DBA 人员来说，“系统调优(performance tuning)”是他(们)的职责之一，更需要掌握查询优化的概念和 RDBMS 的内部优化技术。

9.1　基本知识点

本章首先讲解关系数据库查询处理的步骤和实现查询操作的算法，然后讲解关系数据库系统查询优化的基本概念和方法，包括启发式代数优化、基于规则的存取路径优化和基于代价的优化等方法。

查询处理是 RDBMS 的核心，是 RDBMS 语言处理中最重要、最复杂的部分。本章讲解关系数据库查询(query)处理的步骤和实现查询操作的算法。

为了提高关系数据库系统的执行效率，RDBMS 必须进行查询优化；由于关系查询语言，例如 SQL，具有较高的语义层次，使 RDBMS 可以进行查询优化。这就是 RDBMS 查询优化的必要性和可能性。

① 需要了解的：查询处理的基本步骤，包括查询分析、查询检查、查询优化和查询执行。

② 需要牢固掌握的：什么是关系系统的查询优化，查询优化的方法。

③ 需要举一反三的：能够画出一个查询的语法树以及优化后的语法树。

④ 难点：本章的难点在于优化算法，包括代数优化算法和物理优化算法。

9.2 习题解答和解析

1. 试述查询优化在关系数据库系统中的重要性和可能性。

解析：

重要性：关系系统的查询优化既是 RDBMS 实现的关键技术，又是关系系统的优点所在，它减轻了用户选择存取路径的负担，用户只要提出“干什么”，不必指出“怎么干”。

查询优化的优点不仅在于用户不必考虑如何最好地表达查询以获得较好的效率，而且在于系统可以比用户程序的“优化”做得更好。

可能性：

① 优化器可以从数据字典中获取许多统计信息，例如各个关系中的元组数，关系中每个属性值的分布情况，这些属性上是否有索引、是什么索引（B+树索引、Hash 索引、唯一索引，还是组合索引），等等。优化器可以根据这些信息选择有效的执行计划，而用户程序则难以获得这些信息。

② 如果数据库的物理统计信息改变了，系统可以自动对查询进行重新优化以选择相适应的执行计划。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 优化器可以考虑数十甚至数百种不同的执行计划，从中选出较优的一个，而程序员一般只能考虑有限的几种可能性。

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。

2. 假设关系 $R(A,B)$ 和 $S(B,C,D)$ 情况如下：R 有 20 000 个元组，S 有 1 200 个元组，一个块能装 40 个 R 的元组，能装 30 个 S 的元组，估算下列操作需要多少次磁盘块读。

① R 上没有索引，select * from R；

② R 中 A 为主码，其上有 3 层 B+树索引，select * from R where A = 10；

③ 嵌套循环连接 $R \bowtie S$；

④ 排序合并连接 $R \bowtie S$，区分 R 与 S 在 B 属性上有序和无序两种情况。

答：

① 需要对 R 进行全表扫描，块数 = 20 000/40 = 500。

② 对 R 进行索引扫描，块数 = 3+1 = 4；其中 3 块 B+树索引块，1 块数据块。

③ R 本身 20 000/40 = 500 个块，S 本身 1 200/30 = 40 个块，以 S 为外表，假设内存分配的块数为 k，嵌套循环连接需要的块数为：$40+\left\lceil \frac{40}{k-1} \right\rceil * 500$。

④ 如果 R 和 S 都在 B 属性上排好序，块数 500+40 = 540；如果都没有排序，则还要加上

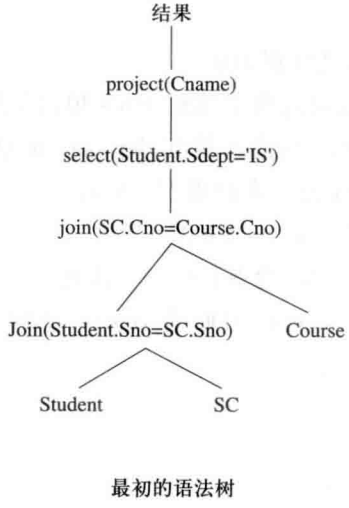
排序代价，结果为 $540+2*500*(\log_2 500+1)+2*40*(\log_2 40+1)$。

3. 对学生-课程数据库，查询信息系学生选修了的所有课程名称。

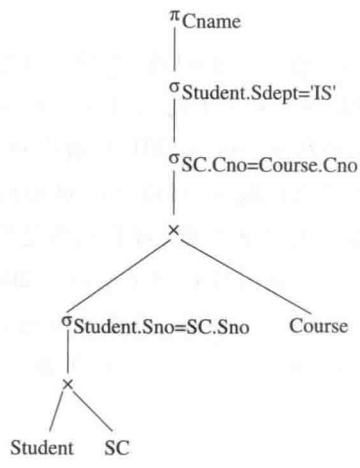
```
SELECT Cname
FROM Student,Course,SC
WHERE Student.Sno=SC.Sno AND SC.Cno=Course.Cno AND Student.Sdept='IS';
```

此查询要求信息系学生选修了所有课程名称。

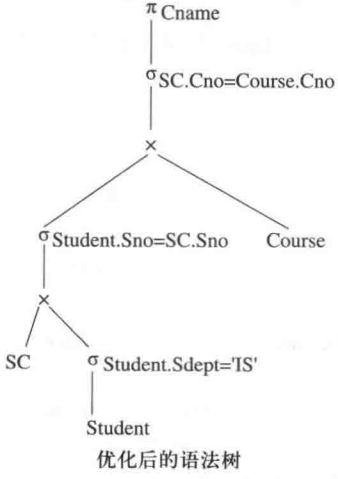
试画出用关系代数表示的语法树，并用关系代数表达式优化算法对原始的语法树进行优化处理，画出优化后的标准语法树。



最初的语法树



关系代数语法树



优化后的语法树

4. 对于下面的数据库模式

Teacher(Tno,Tname,Tage,Tsex);Department(Dno,Dname,Tno);Work(Tno,Dno,Year,Salary)

假设 Teacher 的 Tno 属性、Department 的 Dno 属性以及 Work 的 Year 属性上有 $B+$树索引，说明通常情况下列操作的处理方法。

① select * from teacher where tsex='女'

② select * from department where dno < 301

③ select * from work where year <> 2000

④ select * from work where year > 2000 and salary < 5000

⑤ select * from work where year < 2000 or salary < 5000

答：

① 对 teacher 进行全表扫描，查看元组是否满足性别为女。

② 如果满足 dno < 301 的元组数目较少，可以通过索引找到 dno=301 的索引项，然后顺着 $B+$树的顺序集得到 dno < 301 的索引项，通过这些指针找到 department 中的元组；如果 dno < 301 的元组数目较多，可以采用对 department 的全表扫描方式处理。

③ 对 work 进行全表扫描，查看元组是否满足 year <> 2000。

④ 通过 year 的索引找到满足 year > 2000 的元组，检查元组是否满足 salary < 5000。

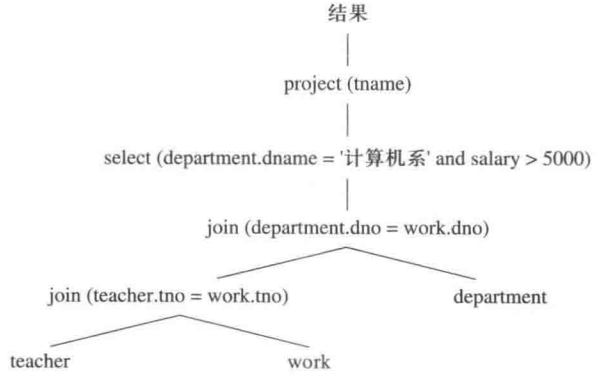
⑤ 对 work 进行全表扫描，查看元组是否满足 year < 2000 或 salary < 5000。

5. 对于题 4 中的数据库模式，存在如下的查询：

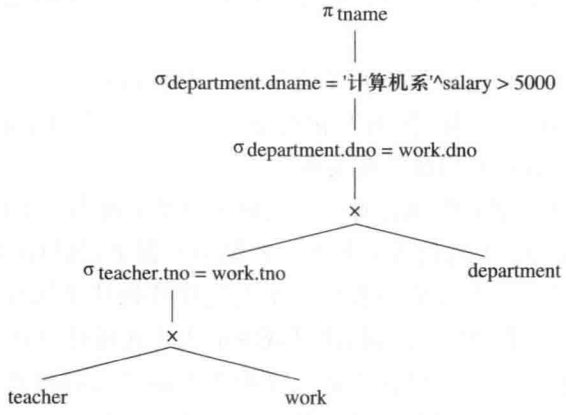
```
SELECT tname
FROM teacher,department,work
WHERE teacher.tno=work.tno AND department.dno=work.dno AND
department.dname='计算机系' AND salary > 5000
```

画出语法树以及用关系代数表示的语法树，并对关系代数语法树进行优化，画出优化后的语法树。

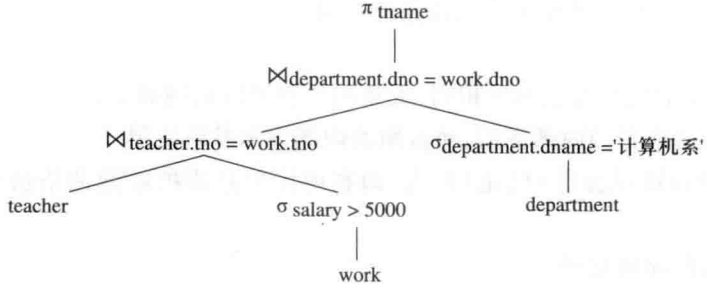
语法树：



关系代数语法树：



优化后的语法树：



6. 试述关系数据库管理系统查询优化的一般准则。

答：

下面的优化策略一般能提高查询效率：

① 选择运算应尽可能先做。

② 把投影运算和选择运算同时进行。

③ 把投影同其前或其后的双目运算结合起来执行。

④ 把某些选择同在它前面要执行的笛卡儿积结合起来成为一个连接运算。

⑤ 找出公共子表达式。

⑥ 选取合适的连接算法。

解析：

①~⑤是指代数优化策略。⑥涉及物理优化。

① 选择运算应尽可能先做。因为满足选择条件的元组一般是原来关系的子集，从而使计算的中间结果变小。

② 把投影运算和选择运算同时进行。如果在同一个关系上有若干投影运算和选择运算,则可以把投影运算和选择运算结合起来,即选出符合条件的元组后就对这些元组做投影。

③ 把投影同其前或其后的双目运算结合起来。双目运算有 JOIN 运算、笛卡儿积,与上面的理由类似,在进行 JOIN 运算、笛卡儿积时要选出关系的元组,没有必要为了投影操作(通常是去掉某些字段)而单独扫描一遍关系。

④ 把某些选择同在它前面要执行的笛卡儿积结合起来成为一个连接运算。连接特别是等连接运算要比在同样关系上的笛卡儿积产生的结果小得多,执行代价也小得多。

⑤ 找出公共子表达式。先计算一次公共子表达式并把结果保存起来共享,以避免重复计算公共子表达式。当查询的是视图时,定义视图的表达式就是公共子表达式的情况。

⑥ 选取合适的连接算法。连接操作是关系操作中最费时的操作,人们研究了许多连接优化算法。例如索引连接算法、排序合并算法、Hash 连接算法等。

选取合适的连接算法属于选择"存取路径",是物理优化的范畴。

7. 试述关系数据库管理系统查询优化的一般步骤。

答:

各个关系系统的优化方法不尽相同,大致的步骤可以归纳如下:

① 把查询转换成某种内部表示,通常用的内部表示是语法树。

② 把语法树转换成标准(优化)形式,即利用优化算法把原始的语法树转换成优化的形式。

③ 选择低层的存取路径。

④ 生成查询计划,选择所需代价最小的计划加以执行。

解析:

读者要把 SQL 查询处理和查询优化的概念结合起来学习,了解 SQL 查询处理工作的步骤。如《概论》图 9.1 所示,了解查询优化在查询处理中的核心地位。

9.3 补充习题

考虑关系模式 $R(A,B,C,D)$,假设 B 上具有 B+树索引,(C,D) 上具有聚簇的 B+树索引。每条关系记录占 100 个字节,索引的数据条目占 20 个字节,整个关系表占 10 000 个数据块。假设符合每个条件的数目比例为 10%,B+树为 3 层,每个数据块包括 40 个关系记录,计算执行下列语句的开销,选出较小的执行代价。

① select * from R where B > 1000

② select * from R where C = 10

③ select * from R where C = 20 and D > 100

④ select sum(B) from R where B > 1000

9.4　补充习题答案

答：

① select * from R where B > 1000

使用 *B*+树索引时代价为

$$2+10000*(20/100)*0.1+10000*40*0.1=40202$$

说明：(*B*+树前两层)+(*B*+树叶子节点的块数)+存取表中记录需要读取的块数；这里的计算比较保守，即需要存取 40 000 条记录，这些记录无序分布，所以要存取次数为记录数。

如果使用全表扫描需要 10 000 个数据块。

因此较小的执行代价是全表扫描的 10 000 个数据块。

② select * from R where C = 10

使用(C,D)上聚簇的 *B*+树索引：

$$2+10000*(20/100)*0.1+10000*0.1=1202$$

说明：(*B*+树前两层)+(*B*+树叶子节点的块数)+存取表中记录需要读取的块数；因为是聚簇的 *B*+树索引，所以 *C* = 10 的记录都聚集在一起，存取次数为总块数的 0.1。

③ select * from R where C = 20 and D > 100

因为每个条件都要乘 0.1，使用聚簇的 *B*+树索引的代价为

$$2+10000*(20/100)*0.1*0.1+10000*0.1*0.1=122$$

说明：(*B*+树前两层)+(*B*+树叶子节点的块数)+存取表中记录需要读取的块数。

④ select sum(B) from R where B > 1000

使用 *B*+树索引，因为可以对索引中属性 *B* 的值进行 sum 操作，不需要再访问表数据，代价为

$$2+10000*(20/100)*0.1=202$$

说明：(*B*+树前两层)+(*B*+树叶子节点的块数)。

第 10 章 数据库恢复技术

《概论》第 10 章和第 11 章讨论事务处理技术。事务处理技术主要包括数据库恢复技术和并发控制技术。数据库恢复机制和并发控制机制是数据库管理系统的重要组成部分。本章讨论数据库恢复的概念和常用技术。

10.1 基本知识点

① 需要了解的:什么是数据库的一致性状态;数据库运行中可能产生的故障类型,它们如何影响事务的正常执行,如何破坏数据库数据;数据转储的概念及分类;数据库的镜像功能。

② 需要牢固掌握的:事务的基本概念和事务的 ACID 性质,数据库恢复的实现技术,日志文件的内容及作用,登记日志文件所要遵循的原则,具有检查点的恢复技术。

③ 需要举一反三的:恢复的基本原理,针对不同故障的恢复策略和方法。

④ 难点:日志文件的使用,系统故障恢复策略。

事务管理模块是 DBMS 实现中的关键技术。事务恢复的基本原理是数据备份,它貌似简单,实际实现却很复杂。数据库的事务管理策略(不仅有数据库恢复策略,还有并发控制策略)与 DBMS 缓冲区管理策略、事务一致性级别密切相关,读者要在学习完全书后再来重新考虑这些问题,提升对这些技术的理解和掌握。

读者要掌握数据库故障恢复的策略和方法。对于刚刚学习数据库课程的读者来讲,可能并不能体会数据库故障恢复的复杂性和重要性。到了实际工作中,作为数据库管理员,则必须十分清楚每一个使用的 DBMS 产品提供的恢复技术与恢复方法,并且能够根据这些技术正确制定出实际系统的恢复策略,以保证数据库系统 7×24 小时正确运行,保证数据库系统在遇到故障时能及时恢复正常运行,提高抗灾难的能力。

10.2 习题解答和解析

1. 试述事务的概念及事务的4个特性。恢复技术能保证事务的哪些特性?

答:

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

事务具有4个特性:原子性(Atomicity)、一致性(Consistency)、隔离性(Isolation)和持续性(Durability)。这4个特性也简称为ACID特性。

原子性:事务是数据库的逻辑工作单位,事务中包括的诸操作要么都做,要么都不做。

一致性:事务执行的结果必须是使数据库从一个一致性状态变到另一个一致性状态。

隔离性:一个事务的执行不能被其他事务干扰。即一个事务内部的操作及使用的数据对其他并发事务是隔离的,并发执行的各个事务之间不能互相干扰。

持续性:持续性也称永久性(permanence),指一个事务一旦提交,它对数据库中数据的改变就应该是永久性的。接下来的其他操作或故障不应该对其执行结果有任何影响。

故障恢复可以保证事务的原子性与持续性。

2. 为什么事务非正常结束时会影响数据库数据的正确性,请举例说明之。

答:

事务执行的结果必须是使数据库从一个一致性状态变到另一个一致性状态。如果数据库系统运行中发生故障,有些事务尚未完成就被迫中断,这些未完成事务对数据库所做的修改有一部分已写入物理数据库,这时数据库就可能处于不正确的状态,或者说是不一致的状态。

例如某工厂的库存管理系统中,要把数量为Q的某种零件从仓库1移到仓库2存放,则可以定义一个事务T,T包括两个操作:$Q1=Q1-Q$,$Q2=Q2+Q$。如果T非正常终止时只做了第一个操作,则数据库就处于不一致性状态,库存量无缘无故少了Q。

3. 登记日志文件时为什么必须先写日志文件,后写数据库?

答:

把对数据的修改写到数据库中和把表示这个修改的日志记录写到日志文件中是两个不同的操作。有可能在这两个操作之间发生故障,即这两个写操作只完成了一个。

如果先写了数据库修改,而在运行记录中没有登记这个修改,则以后就无法恢复这个修改了。如果先写日志,但没有修改数据库,在恢复时只不过是多执行一次UNDO操作,并不会影响数据库的正确性。所以一定要先写日志文件,即首先把日志记录写到日志文件中,然后写数据库的修改。

4. 考虑下图所示的日志记录：

序号	日志
1	$T1$：开始
2	$T1$：写 A，$A=10$
3	$T2$：开始
4	$T2$：写 B，$B=9$
5	$T1$：写 C，$C=11$
6	$T1$：提交
7	$T2$：写 C，$C=13$
8	$T3$：开始
9	$T3$：写 A，$A=8$
10	$T2$：回滚
11	$T3$：写 B，$B=7$
12	$T4$：开始
13	$T3$：提交
14	$T4$：写 C，$C=12$

① 如果系统故障发生在 14 之后，说明哪些事务需要重做，哪些事务需要回滚；
② 如果系统故障发生在 10 之后，说明哪些事务需要重做，哪些事务需要回滚；
③ 如果系统故障发生在 9 之后，说明哪些事务需要重做，哪些事务需要回滚；
④ 如果系统故障发生在 7 之后，说明哪些事务需要重做，哪些事务需要回滚。
答：
① 重做：$T1$、$T3$；回滚：$T2$、$T4$。
② 重做：$T1$；回滚：$T2$、$T3$。
③ 重做：$T1$；回滚：$T2$、$T3$。
④ 重做：$T1$；回滚：$T2$。
5. 考虑题 4 所示的日志记录，假设开始时 A、B、C 的值都是 0：
① 如果系统故障发生在 14 之后，写出系统恢复后 A、B、C 的值。
② 如果系统故障发生在 12 之后，写出系统恢复后 A、B、C 的值。
③ 如果系统故障发生在 10 之后，写出系统恢复后 A、B、C 的值。
④ 如果系统故障发生在 9 之后，写出系统恢复后 A、B、C 的值。
⑤ 如果系统故障发生在 7 之后，写出系统恢复后 A、B、C 的值。
⑥ 如果系统故障发生在 5 之后，写出系统恢复后 A、B、C 的值。

答：

① $A=8, B=7, C=11$。

② $A=10, B=0, C=11$。

③ $A=10, B=0, C=11$。

④ $A=10, B=0, C=11$。

⑤ $A=10, B=0, C=11$。

⑥ $A=0, B=0, C=0$。

6. 针对不同的故障，试给出恢复的策略和方法。（即如何进行事务故障的恢复、系统故障的恢复，以及如何进行介质故障恢复。）

答：

事务故障的恢复步骤是：

① 反向扫描文件日志，查找该事务的更新操作。

② 对该事务的更新操作执行逆操作。即将日志记录中“更新前的值”写入数据库。

直至读到此事务的开始标记，该事务故障的恢复就完成了。

系统故障的恢复步骤是：

① 正向扫描日志文件，找出在故障发生前已经提交的事务队列（REDO 队列）和未完成的事务队列（UNDO 队列）。

② 对未完成的事务队列中的各个事务进行 UNDO 处理。

③ 对已经提交的事务队列中的各个事务进行 REDO 处理。

介质故障的恢复步骤是：

① 装入最新的数据库后备副本（离故障发生时刻最近的转储副本），使数据库恢复到最近一次转储时的一致性状态。

② 装入转储结束时刻的日志文件副本。

③ 启动系统恢复命令，由 DBMS 完成恢复功能，即重做已完成的事务。

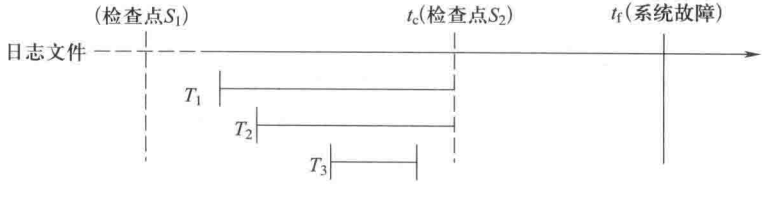
7. 什么是检查点记录，检查点记录包括哪些内容？

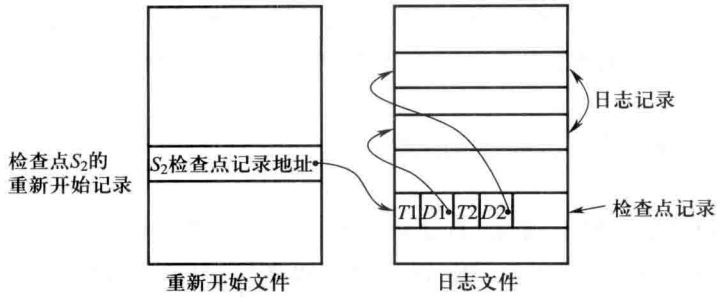
答：

检查点记录是一类新的日志纪录。它的内容包括：

① 建立检查点时刻所有正在执行的事务清单，如下图中的 $T1$、$T2$。

② 这些事务的最近一个日志记录的地址，如下图中的 $D1$、$D2$。





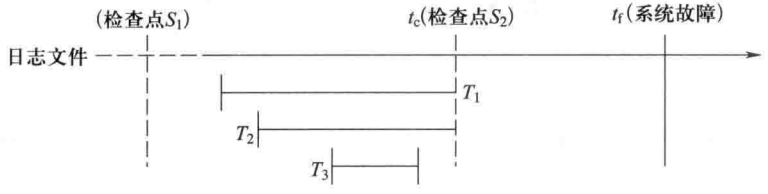
8. 具有检查点的恢复技术有什么优点？试举一个具体的例子加以说明。

答：

利用日志技术进行数据库恢复时，恢复子系统必须搜索整个日志，这将耗费大量的时间。此外，需要 REDO 处理的事务实际上已经将它们的更新操作结果写到数据库中了，恢复子系统又重新执行了这些操作，浪费了大量时间。

检查点技术就是为了解决这些问题。

例如：



在采用检查点技术之前，恢复时需要从头扫描日志文件，而利用检查点技术只需要从 t_c 开始扫描日志，这就缩短了扫描日志的时间。

事务 T_3 的更新操作实际上已经写到数据库中了，进行恢复时没有必要再 REDO 处理，采用检查点技术做到了这一点。

9. 试述使用检查点方法进行恢复的步骤。

答：

① 在重新开始文件（见第 7 题的图）中，找到最后一个检查点记录在日志文件中的地址，由该地址在日志文件中找到最后一个检查点记录。

② 由该检查点记录得到检查点建立时刻所有正在执行的事务清单 ACTIVE-LIST。

这里建立两个事务队列：

- UNDO-LIST：需要执行 undo 操作的事务集合；
- REDO-LIST：需要执行 redo 操作的事务集合。

把 ACTIVE-LIST 暂时放入 UNDO-LIST 队列，REDO 队列暂为空。

③ 从检查点开始正向扫描日志文件

- 如有新开始的事务 T_i，把 T_i 暂时放入 UNDO-LIST 队列；

• 如有提交的事务 T_j，把 T_j 从 UNDO-LIST 队列移到 REDO-LIST 队列，直到日志文件结束。

④ 对 UNDO-LIST 中的每个事务执行 UNDO 操作，对 REDO-LIST 中的每个事务执行 REDO 操作。

10. 什么是数据库镜像？它有什么用途？

答：

数据库镜像即根据 DBA 的要求，自动把整个数据库或者其中的部分关键数据复制到另一个磁盘上。每当主数据库更新时，DBMS 自动把更新后的数据复制过去，即 DBMS 自动保证镜像数据与主数据的一致性。

数据库镜像的用途：

① 用于数据库恢复。当出现介质故障时，镜像磁盘可继续使用，同时 DBMS 自动利用镜像磁盘数据进行数据库的恢复，不需要关闭系统和重装数据库副本。

② 提高数据库的可用性。在没有出现故障时，当一个用户对某个数据加排他锁进行修改时，其他用户可以读镜像数据库上的数据，而不必等待该用户释放锁。

10.3 补充习题

1. 问答题

(1) 在系统故障的恢复策略中，为什么 UNDO 处理反向扫描日志文件，REDO 处理正向扫描日志文件。

(2) 说明恢复系统是否可以保证事务的原子性和持续性。

2. 综合题

考虑下图所示的日志记录：

序号	日志	序号	日志
1	*T*1：开始	13	检查点
2	*T*1：写 *A*	14	*T*7：开始
3	*T*2：开始	15	*T*3：提交
4	*T*2：写 *B*	16	*T*4：回滚
5	*T*3：开始	17	*T*5：写 *B*
6	*T*1：提交	18	*T*8：开始
7	*T*2：回滚	19	*T*6：写 *A*
8	*T*3：写 *C*	20	*T*6：提交
9	*T*4：开始	21	*T*8：写 *A*
10	*T*4：写 *A*	22	*T*8：提交
11	*T*5：开始	23	*T*7：写 *C*
12	*T*6：开始		

（1）如果系统故障发生在 23 之后，说明系统如何进行恢复；

（2）如果系统故障发生在 19 之后，说明系统如何进行恢复。

10.4　补充习题答案

1. 问答题

（1）在系统故障的恢复策略中，为什么 UNDO 处理反向扫描日志文件，REDO 处理正向扫描日志文件。

答：

如果存在同一个数据的多个 UNDO 操作，需要将数据恢复到第一个失败事务之前，如果正向扫描处理日志文件，无法实现这一目标，因此应该反向扫描日志文件。对于同一个数据的多个 REDO 操作，需要将数据恢复到最后一个成功事务之后，因此应该正向扫描日志文件。

（2）说明恢复系统是否可以保证事务的原子性和持续性。

答：

原子性是指操作要么都做，要么都不做，在恢复策略中 UNDO 可以保证将未成功提交的事务所有操作都取消，REDO 可以保证将成功提交的事务所有操作都完成，因此需确保事务的原子性；持续性是指一旦事务提交，对数据库中数据的改变是永久性的，REDO 可以保证事务只要提交，改变一定被永久实现，因此要确保事务的持续性。

2. 综合题

（1）*T*3、*T*6、*T*8 重做，*T*4、*T*5、*T*7 撤销，*T*1、*T*2 不操作；

（2）*T*3 重做，*T*4、*T*5、*T*6、*T*7、*T*8 撤销，*T*1、*T*2 不操作。

第 11 章 并 发 控 制

事务处理技术主要包括数据库恢复技术和并发控制技术。本章讨论数据库并发控制的基本概念和实现技术。本章内容有一定的深度和难度。学习本章一定要做到概念清楚，为此要认真阅读《概论》中相应内容，特别是其中的例题，要自己动手先做一做例题，体会一下是否已经掌握了有关概念。

11.1 基本知识点

数据库是一个共享资源，当多个用户并发地存取数据库时就会产生多个事务同时存取同一个数据的情况。若对并发操作不加控制就可能会存取和存储不正确的数据，破坏数据库的一致性。所以 DBMS 必须提供并发控制机制。

并发控制机制的正确性和高效性是衡量一个 DBMS 性能的重要标志之一。

① 需要了解的：数据库并发控制技术的必要性，活锁与死锁的概念，

② 需要牢固掌握的：并发操作可能产生数据不一致性的情况，包括丢失修改、不可重复读、读“脏”数据等，要牢固掌握它们的确切含义；封锁的类型及不同封锁类型（如 X 锁、S 锁）的性质和定义，相关的相容控制矩阵；封锁协议的概念；封锁粒度的概念，多粒度封锁方法，多粒度封锁协议的相容控制矩阵。

③ 需要举一反三的：封锁协议与数据一致性的关系，并发调度的可串行性概念；两段锁协议与可串行性的关系，两段锁协议与死锁的关系。

④ 难点：两段锁协议与串行性的关系、与死锁的关系，具有意向锁的多粒度封锁方法的封锁过程。

11.2 习题解答和解析

1. 在数据库中为什么要并发控制？并发控制技术能保证事务的哪些特性？

答：

数据库是共享资源,通常有多个事务同时在运行。当多个事务并发地存取数据库时就会产生同时读取和/或修改同一数据的情况。若对并发操作不加控制就可能会存取和存储不正确的数据,破坏数据库的一致性。所以数据库管理系统必须提供并发控制机制。

并发控制可以保证事务的一致性和隔离性。

2. 并发操作可能会产生哪几类数据不一致?用什么方法能避免各种不一致的情况?

答:

并发操作带来的数据不一致性包括三类:

(1) 丢失修改

两个事务 T_1 和 T_2 读入同一数据并修改,T_2 提交的结果破坏了(覆盖了)T_1 提交的结果,导致 T_1 的修改被丢失。

(2) 不可重复读

不可重复读是指事务 T_1 读取某一数据后,事务 T_2 对其执行更新操作,使 T_1 无法再现前一次读取结果。不可重复读包括三种情况:

① 事务 T_1 读取某一数据后,事务 T_2 对其做了修改,当事务 T_1 再次读该数据时,得到与前一次不同的值。

② 事务 T_1 按一定条件从数据库中读取了某些数据记录后,事务 T_2 删除了其中部分记录,当 T_1 再次按相同条件读取数据时,发现某些记录消失了。

③ 事务 T_1 按一定条件从数据库中读取某些数据记录后,事务 T_2 插入了一些记录,当 T_1 再次按相同条件读取数据时,发现多了一些记录。

后两种不可重复读有时也称为幻影(phantom row)现象。

(3) 读"脏"数据

读"脏"数据是指事务 T_1 修改某一数据,并将其写回磁盘,事务 T_2 读取同一数据后,T_1 由于某种原因被撤销,这时 T_1 已修改过的数据恢复原值,T_2 读到的数据就与数据库中的数据不一致,则 T_2 读到的数据就为"脏"数据,即不正确的数据。

避免不一致性的方法就是并发控制。常用的并发控制技术包括封锁方法、时间戳方法、乐观控制方法和多版本并发控制方法等。

3. 什么是封锁?基本的封锁类型有几种?试述它们的含义。

答:

封锁就是事务 T 在对某个数据对象例如表、记录等操作之前,先向系统发出请求,对其加锁。加锁后事务 T 就对该数据对象有了一定的控制,在事务 T 释放它的锁之前,其他的事务不能更新或读取此数据对象。

基本的封锁类型有两种:排他锁(简称 X 锁)和共享锁(简称 S 锁)。

排他锁又称为写锁。若事务 T 对数据对象 A 加上 X 锁,则只允许 T 读取和修改 A,其他任何事务都不能再对 A 加任何类型的锁,直到 T 释放 A 上的锁。这就保证了其他事务在 T

释放 A 上的锁之前不能再读取和修改 A。

共享锁又称为读锁。若事务 T 对数据对象 A 加上 S 锁，则事务 T 可以读 A 但不能修改 A，其他事务只能再对 A 加 S 锁，而不能加 X 锁，直到 T 释放 A 上的 S 锁。这就保证了其他事务可以读 A，但在 T 释放 A 上的 S 锁之前不能对 A 做任何修改。

4. 如何用封锁机制保证数据的一致性？

答：

DBMS 在对数据进行读写操作之前首先对该数据执行封锁操作，例如图中事务 T_1 在对 A 进行修改之前先对 A 执行 Xock(A) 即对 A 加 X 锁。这样，当 T_2 请求对 A 加 X 锁时就被拒绝，T_2 只能等待 T_1 释放 A 上的锁后才能获得对 A 的 X 锁，这时它读到的 A 是 T_1 更新后的值，再按此新的 A 值进行运算。这样就不会丢失 T_1 的更新。

T_1	T_2
① Xlock A	
获得	
② 读 $A=16$	
	Xlock A
	等待
③ $A \leftarrow A-1$	等待
写回 $A=15$	等待
Commit	等待
Unlock A	
④	获得 Xlock A
	读 $A=15$
	$A \leftarrow A-1$
⑤	写回 $A=14$
	Commit
	Unlock A

DBMS 按照一定的封锁协议对并发操作进行控制，使得多个并发操作有序地执行，就可以避免丢失修改、不可重复读和读“脏”数据等数据不一致性。

5. 什么是活锁？试述活锁的产生原因和解决方法。

答：

如果事务 T_1 封锁了数据 R，事务 T_2 又请求封锁 R，于是 T_2 等待。T_3 也请求封锁 R，当 T_1 释放了 R 上的封锁之后系统首先批准了 T_3 的请求，T_2 仍然等待。然后 T_4 又请求封锁 R，当 T_3 释放了 R 上的封锁之后系统又批准了 T_4 的请求……T_2 有可能永远等待，如下图所示。这就是活锁的情形。活锁的含义是该等待事务等待时间太长，似乎被锁住了，实际上可能被激活。

T_1	T_2	T_3	T_4
lock R			
	lock R		
	等待	Lock R	
Unlock	等待		Lock R
	等待	Lock R	等待
	等待		等待
	等待	Unlock	等待
	等待		Lock R
	等待		

活锁产生的原因：当一系列封锁不能按照其先后顺序执行时，就可能导致一些事务无限期等待某个封锁，从而导致活锁。

避免活锁的简单方法是采用先来先服务的策略。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批准申请队列中第一个事务获得锁。

6. 什么是死锁？请给出预防死锁的若干方法。

答：

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

T_1	T_2
lock R_1	
	Lock R_2
Lock R_2	
等待	
等待	Lock R_1
等待	等待

防止死锁的发生其实就是要破坏产生死锁的条件。预防死锁通常有两种方法：

① 一次封锁法

要求每个事务必须一次将所有要使用的数据全部加锁，否则就不能继续执行。

② 顺序封锁法

预先对数据对象规定一个封锁顺序，所有事务都按这个顺序实行封锁。

7. 请给出检测死锁发生的一种方法，当发生死锁后如何解除死锁？

答：

数据库系统一般采用的方法是允许死锁发生，DBMS 检测到死锁后加以解除。

DBMS 中诊断死锁的方法与操作系统类似，一般使用超时法或事务等待图法。

超时法是指如果一个事务的等待时间超过了规定的时限，就认为发生了死锁。

DBMS 并发控制子系统检测到死锁后，就要设法解除。通常采用的方法是选择一个处理死锁代价最小的事务，将其撤销，释放此事务持有的所有锁，使其他事务得以继续运行下去。

8. 什么样的并发调度是正确的调度？

答：

可串行化的调度是正确的调度。

可串行化的调度的定义：多个事务的并发执行是正确的，当且仅当其结果与按某一次序串行地执行它们时的结果相同，称这种调度策略为可串行化的调度。

9. 设 T_1、T_2、T_3是如下的三个事务，设 A 的初值为 0。

$$T_1: A:=A+2;$$

$$T_2: A:=A*2;$$

$$T_3: A:=A*A;(A \leftarrow A^2)$$

① 若这三个事务允许并行执行，则有多少可能的正确结果，请一一列举出来

答：

A 的最终结果可能有 2、4、8、16。

因为串行执行次序有 $T_1\ T_2\ T_3$；$T_1\ T_3\ T_2$；$T_2\ T_1\ T_3$；$T_2\ T_3\ T_1$；$T_3\ T_1\ T_2$；$T_3\ T_2\ T_1$。

对应的执行结果是 16；8；4；2；4；2。

② 请给出一个可串行化的调度，并给出执行结果

答：

T_1	T_2	T_3
Slock A		
$Y=A=0$		
Unlock A Xlock A		
$A=Y+2$	Slock A	
写回 $A(=2)$	等待	
Unlock A	等待	
	等待	
	$Y=A=2$	
	Unlock A	
	Xlock A	
		Slock A
	$A=Y*2$	等待
	写回 $A(=4)$	等待
	Unlock A	等待
		$Y=A=4$
		Unlock A
		Xlock A
		$A=Y*Y$
		写回 $A(=16)$
		Unlock A

最后结果 A 为 16,是可串行化的调度。

③ 请给出一个非串行化的调度,并给出执行结果。

答:

T_1	T_2	T_3
Slock A		
$Y=A=0$		
Unlock A		
	Slock A	
	$Y=A=0$	
Xlock A		
等待	Unlock A	
$A=Y+2$		
写回 $A(=2)$		Slock A
Unlock A		等待
		$Y=A=2$
		Unlock A
		Xlock A
	Xlock A	
	等待	$Y=Y**2$
	等待	写回 $A(=4)$
	等待	Unlock A
	$A=Y*2$	
	写回 $A(=0)$	
	Unlock A	

最后结果 A 为 0,为非串行化的调度。

④ 若这三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

答:

T_1	T_2	T_3
Slock A		
$Y=A=0$		
Xlock A		
$A=Y+2$	Slock A	
写回 $A(=2)$	等待	
Unlock A	等待	
	$Y=A=2$	
	Xlock A	
Unlock A	等待	Slock A
	$A=Y*2$	等待
	写回 $A(=4)$	等待
	Unlock A	等待
		$Y=A=4$
	Unlock A	
		Xlock A
		$A=Y**2$
		写回 $A(=16)$
		Unlock A
		Unlock A

⑤ 若这三个事务都遵守两段锁协议，请给出一个产生死锁的调度。

答：

T_1	T_2	T_3
Slock A $Y=A=0$		
	Slock A $Y=A=0$	
Xlock A 等待		
	Xlock A 等待	
		Slock A $Y=A=0$ Xlock A 等待

10. 今有三个事务的一个调度 $r_3(B)\ r_1(A)\ w_3(B)\ r_2(B)\ r_2(A)\ w_2(B)\ r_1(B)\ w_1(A)$，该调度是冲突可串行化的调度吗？为什么？

答：

是冲突可串行化的调度。

$Sc1=r_3(B)\ r_1(A)\ w_3(B)\ r_2(B)\ r_2(A)\ w_2(B)\ r_1(B)\ w_1(A)$，交换 $r_1(A)$ 和 $w_3(B)$，得到

$$r_3(B)\ w_3(B)\ r_1(A)\ r_2(B)\ r_2(A)\ w_2(B)\ r_1(B)\ w_1(A)$$

再交换 $r_1(A)$ 和 $r_2(B)\ r_2(A)\ w_2(B)$，得到

$$Sc2=r_3(B)\ w_3(B)\ r_2(B)\ r_2(A)\ w_2(B)\ r_1(A)\ r_1(B)\ w_1(A)$$

由于 Sc2 是串行的，而且两次交换都是基于不冲突操作的，所以

$$Sc1=r_3(B)\ r_1(A)\ w_3(B)\ r_2(B)\ r_2(A)\ w_2(B)\ r_1(B)\ w_1(A)$$

是冲突可串行化的调度。

11. 试证明，若并发事务遵守两段锁协议，则对这些事务的并发调度是可串行化的。

证明：

首先以两个并发事务 T_1 和 T_2 为例进行证明，对于多个并发事务的情形可以类推。

根据可串行化定义可知，事务不可串行化只可能发生在下列两种情况：

① 事务 T_1 写某个数据对象 A，T_2 读或写 A；

② 事务 T_1 读或写某个数据对象 A，T_2 写 A。

下面称 A 为潜在冲突对象。

设 T_1 和 T_2 访问的潜在冲突的公共对象为 $\{A_1,A_2,\cdots,A_n\}$。

不失一般性，假设这组潜在冲突对象中 $X=\{A_1,A_2,\cdots,A_i\}$ 均符合情况 1。

$Y=\{A_{i+1},\cdots,A_n\}$ 符合情况 2。

$\forall x \in X$，T_1 需要 Xlock x　　ⓐ

T_2 需要 Slock x 或 Xlock x ⓑ

① 如果操作ⓐ先执行，则 T_1 获得锁，T_2 等待。

由于遵守两段锁协议，T_1 在成功获得 X 和 Y 中全部对象及非潜在冲突对象的锁后，才会释放锁。

这时如果$\exists w \in X$ 或 Y，T_2 已获得 w 的锁，则出现死锁；否则，T_1 在对 X、Y 中对象全部处理完毕后，T_2 才能执行。这相当于按 T_1、T_2 的顺序串行执行。根据可串行化定义，T_1 和 T_2 的调度是可串行化的。

② 操作ⓑ先执行的情况与①对称。

因此，若并发事务遵守两段锁协议，在不发生死锁的情况下，对这些事务的并发调度一定是可串行化的。

证毕。

12. 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锁协议。

答：

T_1	T_2
Slock B	
读 $B=2$	
$Y=B$	
Unlock B	
Xlock A	
	Slock A
	等待
$A=Y+1$	等待
写回 $A=3$	等待
Unlock A	等待
	Slock A
	读 $A=3$
	$X=A$
	Unlock A
	Xlock B
	$B=X+1$
	写回 $B=4$
	Unlock B

13. 考虑如下的调度，说明这些调度集合之间的包含关系。

① 正确的调度。

② 可串行化的调度。

③ 遵循两阶段封锁(2PL)的调度。

④ 串行调度。

答：

③ 遵循两阶段封锁(2PL)的调度 ⊂ ①正确的调度=②可串行化的调度

④ 串行调度 ⊂ ①正确的调度

14. 考虑 T_1和 T_2两个事务。

T_1:R(A);R(B);$B=A+B$;W(B);　T_2:R(B);R(A);$A=A+B$;W(A)

① 改写 T_1和 T_2,增加加锁操作和解锁操作,遵循两阶段封锁协议。

② 说明 T_1和 T_2的执行是否会引起死锁,给出 T_1和 T_2的一个调度说明之。

答：

①

T_1	T_2
Slock A	Slock B
R(A)	R(B)
Xlock B	Xlock A
R(B)	R(A)
$B=A+B$	$A=A+B$
W(B)	W(A)
Unlock A	Unlock B
Unlock B	Unlock A

② 可能产生死锁,如下面的调度所示：

T_1	T_2
Slock A	
R(A)	
	Slock B
	$R(B)$
Xlock B	
	Xlock A

15. 为什么要引进意向锁？意向锁的含义是什么？

答：

引进意向锁是为了提高封锁子系统的效率。

原因是:在多粒度封锁方法中,一个数据对象可能以两种方式加锁——显式封锁和隐式封锁(有关概念参见《概论》第 11.7.1 小节)。因此系统在对某一数据对象加锁时,不仅要检

查该数据对象上有无(显式和隐式)封锁与之冲突,还要检查其所有上级结点和所有下级结点,看申请的封锁是否与这些结点上的(显式和隐式)封锁冲突。显然,这样的检查方法效率很低。为此引进了意向锁。

意向锁的含义是:对任一结点加锁时,必须先对它的上层结点加意向锁。引进意向锁后,系统对某一数据对象加锁时不必逐个检查与下一级结点的封锁冲突了。

16. 试述常用的意向锁:IS 锁、IX 锁、SIX 锁,给出这些锁的相容矩阵。

答:

IS 锁:如果对一个数据对象加 IS 锁,表示它的后裔结点拟(意向)加 S 锁。例如,要对某个元组加 S 锁,则要首先对关系和数据库加 IS 锁

IX 锁:如果对一个数据对象加 IX 锁,表示它的后裔结点拟(意向)加 X 锁。例如,要对某个元组加 X 锁,则要首先对关系和数据库加 IX 锁。

SIX 锁:如果对一个数据对象加 SIX 锁,表示对它加 S 锁,再加 IX 锁,即 SIX=S+IX。

相容矩阵:

T_1 \ T_2	S	X	IS	IX	SIX	—
S	Y	N	Y	N	N	Y
X	N	N	N	N	N	Y
IS	Y	N	Y	Y	Y	Y
IX	N	N	Y	Y	N	Y
SIX	N	N	Y	N	N	Y
—	Y	Y	Y	Y	Y	Y

11.3 补充习题

1. 问答题

(1) 意向锁中为什么存在 SIX 锁,而没有 XIS 锁。

(2) 完整性约束是否能够保证数据库在处理多个事务时处于一致状态。

2. 综合题

考虑下面的三级粒度树,根结点是整个数据库 D,包括关系 R1、R2、R3,分别包括元组 r1,r2,...,r100,r101,...,r200 和 r201,...,r300,使用具有意向锁的多粒度封锁方法,对于下面的操作说明产生加锁请求的锁类型和顺序。

(1) 读元组 r50;

（2）读元组 r90 到 r210；

（3）读 R2 的所有元组并修改满足条件的元组；

（4）删除所有元组。

11.4　补充习题答案

1. 问答题

（1）

答：

如果对数据对象加 SIX 锁，表示对它加 S 锁，再加 IX 锁，即对数据对象加 S 锁，后裔结点拟加 X 锁。

X 锁与任何其他类型的锁都不相容，如果数据对象被加上 X 锁，后裔结点不可能被以任何锁的形式访问，因此 XIS 锁没有意义。

（2）

答：

完整性约束能够保证操作后的数据满足某种约束条件，并不能使多个事务被正确调度，无法保证数据库处于一致状态。

2. 综合题

（1）D 上加 IS 锁；R1 上加 IS 锁；r50 上加 S 锁。

（2）D 上加 IS 锁；R1 上加 IS 锁，R2 上加 S 锁，R3 上加 IS 锁；r90 到 r100 上加 S 锁，r201 到 r210 上加 S 锁；

（3）D 上加 IS 锁和 IX 锁；R2 上加 SIX 锁；

（4）D 上加 IX 锁；R1、R2、R3 上加 X 锁。

第二部分

实验指导

本书在前一版的基础上进一步加强了上机实验。

本书针对《概论》第1章~第11章的内容设计了11个实验，每个实验包括若干实验项目，共26个实验项目，其中必修实验项目12个，选修实验项目14个。

第二部分首先介绍了数据库课程实验环境建设、实验数据准备的技术和方法，然后对每一章的实验，讲解了所包括的每个实验项目的实验目的、实验内容和要求、实验重点和难点，并给出了较为详细的实验报告示例，最后列出了SQL语言实验中一些常见问题及解答。

一 绪 论

数据库系统概论课程的教学内容包括基础知识、关系数据库、数据库设计、数据库管理系统(DBMS)和数据库新技术等,其知识体系按研究范畴可以分为理论、方法、技术和应用4个方面,按学习的深度和广度可以分为使用、管理和设计等三个层次(如图1所示)。由此可见,数据库系统概论是一门理论性、技术性和实践性都很强的课程,其教学目标不但要使学生比较全面地了解数据库系统的基本概念和原理,更要培养学生的数据库实际应用能力。

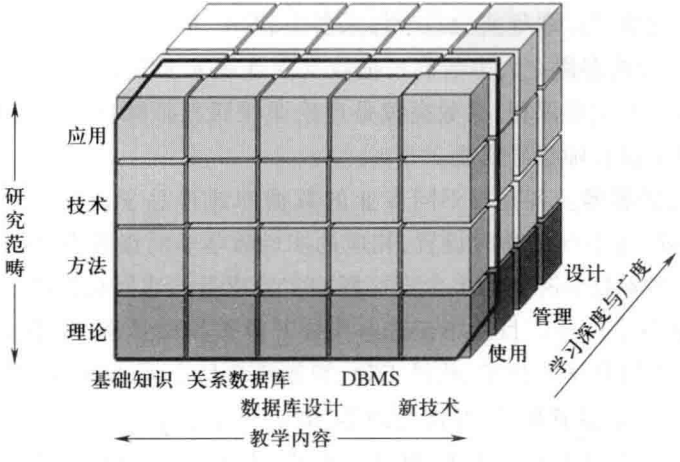


图1 数据库管理系统知识体系结构示意图

数据库实验课程针对数据库知识点设置相应的实验,锻炼学生的实际动手能力,启发学生对所学数据库知识的深入思考,达到理论联系实际的教学效果。实验类型通常分为验证性、设计性、综合性和创新性4种实验,表1从实验要求、培养学生能力和难易程度三方面对比分析了上述4种实验类型的异同。本部分以验证性和设计性实验为主,以综合性实验为辅,没有涉及创新性实验。

表 1 4 种实验类型对照表

区别 / 实验类型	实验要求	培养学生能力	难易程度
验证性实验	针对已学知识，以验证实验结果、巩固和加强课堂知识内容为目的的重复性实验	实验操作能力	容易
设计性实验	给定实验目的和要求，学生自行设计方案、选择实验条件并加以实现	独立解决实际问题能力	较难
综合性实验	涉及本课程或相关课程的综合知识和方法的实验	综合分析及查阅文献等能力	较难
创新性实验	运用多学科知识，结合教师科研项目，在教师指导下完成的实验	培养科学思维方式和研究方法，提高科学探索能力	难

1. 国内数据库课程实验教学现状

目前国内各高校数据库课程实验教学体系各不相同，主要体现在以下几个方面：

① 实验课开设方式多样。大多数高校把数据库课程实验设置为课内实验，也有部分高校设置为单独的数据库实验课程；多数高校是理论课授课教师兼任实验课指导教师，也有部分高校安排专门的实验教师指导实验。

② 实验学时设置各异。各高校不同专业的数据库课程总学时设置各不相同，有 72 学时、68 学时和 54 学时等不同的学时设置，相应的实验教学学时设置有 36 学时、20 学时和 12 学时等多种安排。实验教学时数的多少直接影响实验项目的选择和设置。

③ 开课高校和专业众多。目前不仅普通高校开设数据库课程，也有众多高职院校开设该门课程。除了计算机科学与技术、软件工程、智能科学与技术、网络工程等计算机类专业开设数据库课程外，还有信息管理类、经济管理类、电信类等专业开设该门课程。不同类型的高校和专业对学生具有不同的培养目标和培养模式，对数据库课程实验教学的要求也各不相同。

④ 实验项目设置不尽相同。由于各高校和专业的特点不同，数据库课程实验教学的实验项目设置也不尽相同。大部分院校开设数据库安装与配置、SQL 语言、视图、完整性语言、安全性语言、数据库设计等实验，而与 DBA 管理和 DBMS 系统内核相关的实验因难度较大开设的院校较少。

⑤ 实验平台多种多样。目前各院校主要选择某一种 DBMS 作为实验平台，例如 Oracle、SQL Server 等国外商用 DBMS，人大金仓 KingbaseES 等国内商用 DBMS，MySQL 等开源

DBMS,或者 Access 个人数据库系统。

通过调查了解国内部分高校数据库课程实验教学现状和教学改革情况,分析各种数据库课程实验指导手册,结合我们多年的数据库课程实验教学经验,针对目前数据库课程实验教学过程中存在的不足,我们对下面几方面问题进行了思考探索:

① 实验环境建设。目前大多数高校数据库实验教学环境是单机环境,即在每台学生机上预先安装配置好一套独立的 DBMS,每个学生使用独立的 DBMS 进行实验。单机环境维护简单,但没有体现数据库共享和并发访问的特点,也没有体现网络时代的特点。因此,如何建设一个网络化的数据库课程实验教学环境,甚至让学生能够参与到数据库课程实验环境建设中来,而不是仅仅使用 DBMS,将进一步锻炼学生的动手操作能力。

② 实验数据准备。数据库课程实验教师通常提前准备好一个数据量较少的实验数据库,学生做实验时导入该数据库即可做实验。数据量较少的数据库无须建立索引,无须查询优化,都可以表现出很好的性能。而"数据库是有组织、可共享的长期存储在计算机内的大量数据的集合",只有当数据库数据规模达到一定程度,才能体现出数据库系统的特点和优势。另外,准备实验数据的过程也是一个培养和锻炼学生分析问题和解决问题的极好机会。因此,如何合理安排实验项目,充分调动学生的积极性,准备一个数据规模较大的实验数据库,是需要重视和研究的问题。

③ 实验项目设置。目前不少数据库实验教学大纲都规定了非常具体明确的实验内容,相应的实验指导手册甚至给出了详细的实验步骤,学生照着指导手册一步一步地"依照葫芦画瓢"就可以顺利完成实验。如何按照启发式教学方式、方法来灵活设置实验项目和制定实验大纲,让学生拥有发挥的空间和创新的动力,是需要认真探索的问题。

④ 实验平台。数据库课程讲授的是通用的数据库知识,通常不依赖于某一种特定的 DBMS。不同的 DBMS 具有各自的特点,甚至在某些具体特性方面差别还比较大。而数据库实验教学过程中通常选择某一种特定的 DBMS 作为实验教学平台。如何建立一个由多种 DBMS 组成的开放的数据库实验教学环境,也是需要认真思考的问题。

2. 本书实验指导的特点

本书首先介绍数据库课程实验环境建设和实验数据准备的技术和方法,然后详细介绍了数据库课程实验设置,并针对每个实验给出了较为详细的实验报告示例,最后介绍了数据库课程实验考核标准和实验评价方法,并给出了数据库课程实验中一些常见问题及其解答。

本书以验证性实验为主,也设置部分设计性实验和综合性试验。实验教师可以根据具体情况设置少量的创新性实验。实验指导部分的具体特点是:

① 配套《概论》教材,参照《高等学校计算机科学与技术专业实践教学体系与规范》,设置数据库课程实验及其实验项目。

② 阐述了数据库实验环境的建设方法。

③ 介绍了数据库实验数据准备的有关技术和方法。

④ 全部实验以 TPC-H 标准数据库模式为基础，给出了各个实验项目的实验报告示例。

⑤ 实验报告示例不是简单地示范 SQL 语言或者 DBMS 的功能，而是比较综合实用，可以更好地培养学生实际应用 DBMS 的能力。

⑥ 实验代码以国产数据库管理系统金仓数据库(KingbaseES)为基础，但又不局限于该 DBMS，其实验示例代码针对其他具体的 DBMS 稍做改造也同样可以运行。

二 实验环境建设

数据库课程的实验环境主要分为单机实验环境、网络实验环境和开放实验环境三种。

1. 单机环境

单机实验环境是每台计算机都安装一套完整的 DBMS,每个学生都使用配置相同的独立的 DBMS 做实验,互不影响。这是目前用得较多的实验环境。单机环境的优点是安装配置和管理都比较简单,缺点是一旦需要改变 DBMS 配置或者升级 DBMS,需要重新安装所有的计算机,升级维护的工作量比较大。

单机实验环境难以体现 DBMS 共享和并发的特点,也难以完成一些比较复杂的数据库课程实验。

2. 网络环境

在一台服务器上安装一套 DBMS,每台学生用的计算机(简称学生机)只安装 DBMS 的客户端。网络实验环境要求在服务器上为每个学生创建一个用户名,并分配创建数据库或者数据库模式等权限。学生通过客户端访问服务器上的 DBMS,创建各自独立的实验数据库或者数据库模式进行实验,以免互相干扰。网络实验环境的缺点是安装配置和管理相对复杂一些,对服务器的性能要求比较高,需要高效地支持几十个甚至上百个学生同时做实验,并发访问 DBMS。

网络环境的优点是升级维护的工作量比较小,只需升级维护好服务器即可。该实验环境可以很好地体现 DBMS 共享和并发的特点,可以完成比较复杂的数据库课程实验,特别是可以支持多人共同参与的实验。

3. 开放平台

在服务器上安装多个 DBMS,在学生机安装多个 DBMS 的客户端程序,或者安装一套通用的 DBMS 客户端程序,学生可以选择任意一个 DBMS 进行实验。开放实验环境的安装配置和维护管理更为复杂,要求实验教师能够比较熟练地使用多种 DBMS。开放实验环境对学生的要求也比较高,不同类型的 DBMS 之间存在的差异容易导致学生实验时出现混淆。

开放平台实验环境的最大优点是不限定某一种 DBMS 作为实验平台,让学生享有充分的选择权和灵活性,也使得学生可以比较不同 DBMS 的优缺点,更好地理解和掌握数据库基本知识。

三 实验数据准备

实验数据集越大，实验数据质量越好，数据库课程实验效果就越好，越能激发学生利用数据库知识探索实验数据奥秘的主动性和积极性，从而越能提高数据库实验课程的总体教学效果。

实验数据需要根据所选择或者所设计的数据库模式来进行准备。实验数据准备有三种方式，第一种方式是利用现有数据集作为实验数据集；第二种方式是收集和整理各种来源的数据以生成真实的实验数据集，例如从 Web 上抓取相关行业或领域的数据集，分析整理以生成一个真实的数据集；第三种方式是利用数据库模拟数据生成工具生成模拟的实验数据集。下面具体介绍每种实验数据准备的方法。

1. 利用现有数据集

根据所选择或者所设计的数据库模式来搜索是否有现成的可用数据集，或者根据现有可用数据集设计相应的数据库模式作为实验数据库模式。

针对每个行业或者领域，在网络上都可以找到大量免费的可用数据集。例如学术论文数据库 DBLP（http://www.informatik.uni-trier.de/~ley/db/）、国际电影数据库 IMDB（http://www.imdb.com/）、从维基百科 WikiPedia 抽取结构化信息构成的数据集 DBpedia（http://dbpedia.org/About）等。各个数据集都以不同的格式提供，例如 DBLP 以 XML 格式提供、IMDB 以具有一定格式的 Text 文件提供，DBPEDIA 以 N-TRIPLE 形式提供。有的数据集可以找到相应的代码或者工具把原始格式转换成某种数据库的格式，以便装载到数据库系统中去，例如 IMDB 可以用 JMDB 工具（http://www.jmdb.de/）导入到 MYSQL 数据库中。有的数据集需要编写相应的解析程序把原始数据转换成期望的数据库格式。有的数据集，例如DBPEDIA，非常庞大。因此，对于现有数据集也需要仔细考察，有选择地加以利用。

现有数据集通常是真实的数据集，数据量比较大，是比较理想的实验数据集。但是现有数据集通常没有提供直接导入数据库的备份数据格式，数据转换和装载入库的工作量比较大。

2. 抓取网络数据,构造实验数据集

抓取网络数据构造实验数据集,这是目前很流行的一种实验数据采集方式。利用现有的抓取工具,如 Webharvest(http://web-harvest.sourceforge.net/)和 Tika(http://tika.apache.org/),或者针对具体网站编写相应的数据抓取程序,可以采集到大量的网络数据,以便构造一个真实的实验数据集。

抓取网络数据需要花费大量的时间和精力,需要编写程序解析抓取到的数据,然后转换成相应的数据库格式,工作量很大。

3. 模拟数据生成工具

模拟数据生成工具分为三类:

① 数据库辅助设计工具,如 PowerDesigner 可以产生测试数据。

② 针对特定数据库模式的专用测试数据产生工具,如针对 TPC-H 数据库模式的 DBGEN(http://www.tpc.org/tpch/),可以参考 http://www.geniiius.com/blog/generate-test-data-using-dbgen/文章产生模拟数据并导入数据库中。

③ 通用测试数据产生工具,如:

DBMONSTER(http://sourceforge.net/projects/dbmonster/)、

Datagenerator(http://sourceforge.net/projects/datagenerator/)、

TestDataBuilder(http://sourceforge.net/projects/testdatabuilder/)等。

模拟数据生成工具可以生成任意规模的实验数据,适合作为性能测试数据集。但是模拟数据的可读性差,不利于初学者理解数据含义以及数据之间的联系。

4. 本书所用实验数据集生成方法

本书采用事务处理性能委员会(Transaction Processing Performance Council,TPC)为决策支持基准测试而制定的TPC-H(http://www.tpc.org/tpch/)标准数据库模式作为所有实验示例的数据库模式,采用抓取网络数据和模拟生成部分数据的综合方法生成实验数据,得到了一个数据规模较大且可读性较强的实验数据集。

如果编写程序从网站上抽取结构化数据,不仅需要花费较多的时间和精力,也分散学生学习数据库课程的注意力。我们设计了一种无须编程就能收集整理真实数据的方法。具体做法是:首先从网上找到具有大量真实数据的网站,把其网页数据复制到 Word 文件中,并经过简单的编辑处理成有规律的文本数据,然后利用 Word 把文本数据转换成表格数据,利用表格交换列、排序等功能整理表格数据,再利用 Word 把表格数据转换成 TPC-H 数据库模式需要的格式文本数据,最后利用 DBMS 批量加载数据的功能导入真实的数据集。

利用这种方法,我们收集整理了大量的零件数据(part)、供应商数据(supplier),国家(nation)代码数据、地区(region)代码数据,常用人名作为顾客(customer)数据。数据集统计情况表如表 2 所示。

表 2　本书使用的 TPC-H 数据集统计情况表

序号	基本表名	记录数(条)	备注
1	Region	6	导入数据
2	Nation	150	导入数据
3	Part	65755	导入数据
4	Supplier	30810	导入数据
5	Customer	661396	导入数据
6	PartSupp	30000	随机生成数据,可以生成更多记录
7	Orders	21100	随机生成数据,可以生成更多记录,生成 1 000 条记录花费约 46 s
8	Lineitem	22026	随机生成数据,可以生成更多记录,生成 1 000 条记录花费约 4 s

设计模拟数据生成方法,我们利用 RAND( )随机函数设计了一系列的 INSERT 语句和存储过程,多次循环执行,随机生成供应情况(PartSupp)、订单(Orders)和订单明细(Lineitem)等实体或者实体联系表的模拟数据。

已经整理好的实验数据集放在“数据库系统概论”精品课程网站 http ://www.chinadb.org 上,读者可以直接下载使用。下载请点击链接 http ://www.chinadb.org/dbtestdata.rar。

实验数据收集、整理和生成的过程实际上也是数据库原理知识的实际应用过程,该过程非常有趣,能够吸引学生,调动学生学习数据库的兴趣,锻炼学生分析和解决数据库应用问题的能力。因为在生成实验数据的过程中会遇到许多意想不到的问题,例如生成数据的过程中如何去除大量数据中的重复记录,如何按照一定的原则补充和完善数据记录中的空值等,每一个问题的解决都会让学生学到处理实际问题的方法和技术。

我们鼓励老师和同学在具有了一定数据库知识后自己动手准备实验数据集。为此,下面给出在金仓数据库(KingbaseES)中生成 TPC-H 测试数据库的过程。

① 在 KingbaseES 中创建 TEST 数据库,然后创建 SALES 模式,并创建 TPC-H 数据库模式,参见实验 1.1 示例报告。

② 导入数据。

```
/ * 把 D 驱动器 tpchdata 目录中数据文件分别导入数据到 Region、Nation、Part、Supplier 和 Customer 等表中 * /
```

```
COPY Sales.Region
    FROM 'D:\tpchdata\region.csv' WITH DELIMITER AS ',';
COPY Sales.Nation
    FROM 'D:\tpchdata\nation.csv' WITH DELIMITER AS ',';
COPY Sales.Part(partkey,name,mfgr,type,retailprice)
    FROM 'D:\tpchdata\part.csv' WITH DELIMITER AS ',';
COPY Sales.Supplier(suppkey,name,address,phone)
    FROM 'D:\tpchdata\supplier.csv' WITH DELIMITER AS ',';
COPY Sales.Customer(custkey,name)
    FROM 'D:\tpchdata\customer.csv' WITH DELIMITER AS ',';
```

③ 随机设置顾客的国籍。

```
/*首先创建存储过程 setCustomerNations(),然后执行该存储过程*/
CREATE OR REPLACE PROCEDURE Sales.setCustomerNations() AS
DECLARE
    nationCount INT;
    res RECORD;  /*定义游标结果的变量 res 为记录类型*/
    CURSOR mycursor FOR SELECT custkey FROM Sales.Customer;
BEGIN
    SELECT COUNT(*) INTO nationCount FROM Sales.Nation;/*计算 Nation 表中的记录个数*/
    OPEN mycursor;
    LOOP
        FETCH mycursor INTO res;/*从游标中取出的结果保存在 res 变量中*/
        IF mycursor%NOTFOUND THEN
            EXIT;
        END IF;
        UPDATE Sales.Customer /*随机设置顾客的国籍*/
        SET nationkey=(SELECT nationkey
                       FROM Sales.Nation LIMIT 1
                       OFFSET MOD(CAST(RANDOM()*1000 AS INTEGER),nationCount))
        WHERE custkey=res.custkey;
    END LOOP;
    CLOSE mycursor;
END;
CALL sales.setCustomerNations();/*调用存储过程设置顾客的国籍*/
```

④ 随机设置供应商的国籍。

```
/*首先创建存储过程 setSupplierNations(),然后执行该存储过程*/
CREATE OR REPLACE PROCEDURE sales.setSupplierNations() AS
```

```
DECLARE
nationCount INT;
res RECORD;                    /*定义游标结果为记录类型变量*/
CURSOR mycursor FOR SELECT suppkey FROM Sales.Supplier;
BEGIN
    SELECT COUNT(*) INTO nationCount FROM Sales.Nation;/*计算 Nation 表中的记录个数*/
    OPEN mycursor;
    LOOP
        FETCH mycursor INTO res;/*从游标中取出的结果保存在 res 记录类型的变量中*/
        IF mycursor%NOTFOUND THEN
            EXIT;
        END IF;
        UPDATE Sales.Supplier                    /*随机设置供应商的国籍*/
        SET nationkey=(SELECT nationkey
            FROM Sales.Nation
            LIMIT 1
            OFFSET MOD(CAST(RANDOM()*1000 AS INTEGER),nationCount))
        WHERE suppkey=res.suppkey;
    END LOOP;
    CLOSE mycursor;
END;
CALL sales.setSupplierNations();      /*调用存储过程设置供应商的国籍*/
```

⑤ 随机生成 PartSupp 数据。

```
/*随机生成零件供应联系表 PartSupp 记录的存储过程:参数 p_partsuppCount 为需要产生的记录数*/
CREATE OR REPLACE PROCEDURE sales.insert_PartSupp(p_partsuppCount INT) AS
DECLARE
    supplierCount INT;
    partCount INT;
    NewPartKey INT;
    NewSuppKey INT;
    tmp INT;
BEGIN
    SELECT COUNT(*) INTO partCount FROM Sales.Part;/*计算 Part 表的记录数*/
    SELECT COUNT(*) INTO supplierCount FROM Sales.Supplier;/*计算 Supplier 表的记录数*/
    /*循环生成 PartSupp 记录:首先随机生成 partkey 和 suppkey,若 PartSupp 中不存在相应的记
    录,则增加一条新的记录*/
    FOR i IN 1..p_partsuppCount LOOP
```

```
        NewPartKey := (SELECT partkey          /* 随机获得一个合法的零件编号 */
                FROM Sales.Part
                LIMIT 1
                OFFSET
                MOD(CAST(RANDOM() * 100 * partCount AS INTEGER),partCount));
        NewSuppKey := (SELECT suppkey          /* 随机获得一个合法的供应商编号 */
                FROM Sales.Supplier
                LIMIT 1
                OFFSET
                MOD( CAST( RANDOM( ) * 100 * supplierCount AS INTEGER) ,supplierCount) );
        SELECT partkey INTO tmp                /* 查询将生成的零件供应记录是否已经存在 */
        FROM Sales.PartSupp
        WHERE partkey=NewPartKey AND suppkey=NewSuppKey;
          IF SQL%NOTFOUND THEN
            INSERT INTO Sales.PartSupp         /* 插入新的零件供应记录 */
            VALUES(NewPartKey,NewSuppKey,CAST(RANDOM() * 1000 AS INTEGER),CAST(RAN-
            DOM() * 10000 AS REAL));
          END IF;
        /* 循环 200 次,插入 200 条记录,就提交一次事务,以节省内存,提高执行效率 */
        IF (MOD(p_partsuppCount,200)= 0 ) THEN
              COMMIT;
        END IF;
    END LOOP;
END;
CALL Sales.insert_PartSupp(30000);          /* 执行存储过程 */
```

⑥ 随机生成 oders 数据。

```
/* 随机生成订单表 Orders 记录的存储过程:参数 p_orderCount 为需要产生的记录数 */
CREATE OR REPLACE PROCEDURE sales.insert_Orders( p_orderCount INT) AS
DECLARE
    customerCount INT;
    existingMaxOrderKey INT;
    NewOrderKey INT;
    NewCustKey INT;
    tmp INT;
    i INT;
BEGIN
    SELECT COUNT( * ) INTO customerCount FROM Sales.Customer;/* 计算 Supplier 表的记录数 */
```

```
    SELECT MAX(orderkey) INTO existingMaxOrderKey FROM Sales.Orders;
                                                    /* 计算 Orders 表已有的最大 orderkey */
    IF existingMaxOrderKey IS NULL THEN
        existingMaxOrderKey = 0;
    END IF;
    /* 循环产生订单记录:首先找到已有订单中的最大订单号,在此基础上生成新的订单记录号 */
    FOR i IN 1..p_orderCount LOOP
        NewOrderKey := existingMaxOrderKey+i;
        NewCustKey = (SELECT custkey      /* 随机获得一个客户编号 */
                    FROM Sales.Customer
                    LIMIT 1
                    OFFSET
                    MOD(CAST(RANDOM() * 100 * customerCount AS INTEGER),
                        customerCount));
        INSERT INTO Sales.Orders(orderkey,custkey,orderdate)   /* 插入一条订单记录 */
            VALUES(NewOrderKey,NewCustKey,
            DATEADD('day',MOD( CAST( RANDOM() * 1000 AS INTEGER),365),
            CURRENT_DATE));
    /* 循环 200 次,插入 200 条记录,就提交一次事务,以节省内存提高执行效率 */
    IF (MOD(p_orderCount,200) = 0 ) THEN
        COMMIT;
    END IF;
    END LOOP;
END;
CALL sales.insert_Orders(30000);        /* 执行存储过程 */
```

⑦ 随机生成 Lineitem 数据。

```
/* 随机生成订单明细表 Lineitem 记录的存储过程:参数 p_orderCount 为需要产生订单明细的订单数 */
CREATE OR REPLACE PROCEDURE sales.insert_Lineitem(p_orderCount INT) AS
    DECLARE
        NewOrderKey INT;
        NewPartKey INT;
        NewSuppKey INT;
        L_orderloop INT;
        L_linenumber   INT;
        L_existingMaxLinenumber INT;
        partsuppCount INT;
        lineitemCount   INT;
```

```
    CURSOR mycursor FOR SELECT orderkey FROM sales.Orders;
BEGIN
    SELECT COUNT( * ) INTO partsuppCount FROM Sales.PartSupp;/ * 计算 PartSupp 的记录数 * /
    OPEN mycursor;
    IF mycursor%ISOPEN THEN
      FOR L_orderloop IN 1..p_orderCount LOOP
       / * 获取 Orders 订单表的记录,如果没有记录,或者是已经产生足够的订单明细了,
       就退出循环 * /
       FETCH mycursor INTO NewOrderKey;
       EXIT WHEN (mycursor%NOTFOUND);
       / * 计算 Lineitem 表中当前订单的最大的 linenumber,在此基础上产生新的 linenum-
       ber 号 * /
       SELECT MAX(linenumber) INTO L_existingMaxLinenumber FROM Sales.Lineitem
       WHERE orderkey = NewOrderKey;
       IF L_existingMaxLinenumber IS NULL THEN
          L_existingMaxLinenumber = 0;
       END IF;
       / * 随机设置每个订单产生的订单明细记录条数:最多产生 3 条明细记录 * /
       lineitemCount := MOD(CAST(RANDOM( ) * 1000 AS INTEGER),3);
       FOR L_linenumber IN 1..lineitemCount LOOP/ * 循环产生当前订单的订单明细记录 * /
       / * 随机生成订单明细记录中所购买的对应零件供应联系表中的记录号 * /
              SELECT partkey,suppkey INTO NewPartKey,NewSuppKey
               FROM sales.partsupp LIMIT 1 OFFSET MOD(CAST(RANDOM( ) * 1000000
                    AS INTEGER),partsuppCount );
              INSERT INTO sales.Lineitem (orderkey,partkey,suppkey,linenumber,quantity,
                                          discount,extendedprice)      / * 插入订单明细
                                          记录 * /
               VALUES(NewOrderKey,NewPartKey,NewSuppKey,L_linenumber+L_existing-
                      MaxLinenumber,CAST(RANDOM( ) * 100 AS INTEGER),RAN-
                      DOM( ),0);
      END LOOP;
       / * 循环 200 次,插入 200 条记录,就提交一次事务,以节省内存提高执行效率 * /
       IF (MOD(p_orderCount,200) = 0 ) THEN
           COMMIT;
       END IF;
      END LOOP;
    CLOSE mycursor;
```

```
        END IF;
END;
CALL sales.insert_Lineitem(20000);        /* 执行存储过程 */
UPDATE Sales.Lineitem                  /* 设置订单明细记录中的销售价格 extendedprice */
SET extendedprice=quantity * Part.retailprice
    FROM Sales.Part WHERE Sales.Part.partkey=Sales.Lineitem.partkey;
```

四 数据库课程实验

表 3 列出数据库课程实验总体情况：共设置 11 个实验、26 个实验项目，其中必修 12 个实验项目，选修 14 个实验项目。本表还列出了各实验项目对应的章节内容，以便复习实验内容。各类实验项目可以自由裁剪组合。

表 3 数据库课程实验一览表

序号	实验名称	开设类别	难易程度	实验项目数（必修/选修）	实验学时（必修/选修）	对应《概论》章节
实验 1	数据库定义与操作语言	必修	适中	5/1	10/2	第 3 章
实验 2	安全性语言	必修	适中	1/1	2/2	第 4 章
实验 3	完整性语言	必修	适中	2/1	4/2	第 5 章
实验 4	触发器	必修	适中	1/0	2/0	第 5 章
实验 5	数据库设计	必修	适中	1/0	4/0	第 7 章
实验 6	存储过程	必修	适中	1/2	2/4	第 8 章
实验 7	数据库应用开发	选修	难	0/2	0/8	第 8 章
实验 8	数据库设计与应用开发大作业	必修	适中	1/0	8/0	第 3~8 章
实验 9	数据库监视与性能优化	选修	较难	0/3	0/12	第 9 章
实验 10	数据库恢复技术	选修	适中	0/3	0/6	第 10 章
实验 11	并发控制	选修	适中	0/1	0/2	第 11 章
合计				12/14	32/38	

说明：实验学时仅列出课堂学时，部分实验（如实验 8）还需学生利用较多的课外时间来完成。

实验 1 数据库定义与操作语言实验

数据库定义与操作语言实验包含 6 个实验项目(参见表 4),其中有 5 个必修实验项目,1 个选修实验项目。实验项目 1.1~1.5 为设计性实验,实验项目 1.6 为验证性。

表 4 "数据库定义与操作语言"实验项目一览表

序号	实验项目名称	开设类别	难易程度	建议实验学时	对应教材章节
实验 1.1	数据库定义实验	必修	适中	2	第 3 章 3.3
实验 1.2	数据基本查询实验	必修	适中	2	第 3 章 3.4
实验 1.3	数据高级查询实验	必修	较难	2	第 3 章 3.4
实验 1.4	数据更新实验	必修	适中	2	第 3 章 3.5
实验 1.5	视图实验	必修	适中	2	第 3 章 3.7
实验 1.6	索引实验	选修	适中	2	第 3 章 3.3.3

实验 1.1 数据库定义实验

1. 实验目的

理解和掌握数据库 DDL 语言,能够熟练地使用 SQL DDL 语句创建、修改和删除数据库、模式和基本表。

2. 实验内容和要求

理解和掌握 SQL DDL 语句的语法,特别是各种参数的具体含义和使用方法;使用 SQL 语句创建、修改和删除数据库、模式和基本表。掌握 SQL 语句常见语法错误的调试方法。

3. 实验重点和难点

实验重点:创建数据库、基本表。

实验难点:创建基本表时,为不同的列选择合适的数据类型,正确创建表级和列级完整性约束,如列值是否允许为空、主码和外码等。注意:数据完整性约束可以在创建基本表时定义,也可以先创建表然后定义完整性约束。由于完整性约束的限制,被引用的表要先创建。

4. 实验报告示例

实 验 报 告				
题目:数据库定义实验	姓名		日期	
本实验建立 TPC-H 数据库模式(如图 2 所示)。TPC-H 数据库模式由零件表(Part)、供应商表(Supplier)、零件供应商联系表(Partsupp)、顾客表(Customer)、国家表(Nation)、地区表(Region)、订单表(Orders)和订单明细表(Lineitem)8 个基本表组成。				

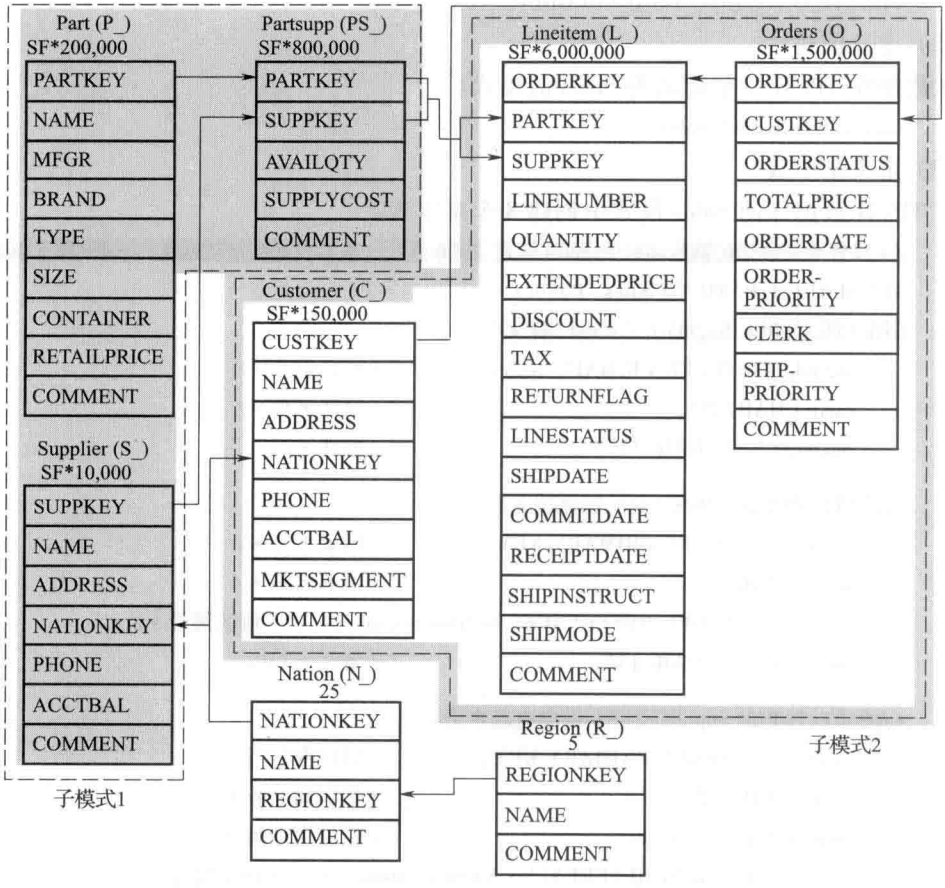


图 2　TPC-H 数据库模式图

TPC-H 数据库模式又可以分为以下两个子模式。

子模式 1:零件供应商子模式,包括 Part、Supplier 和 PartSupp 三个基本表,类似学生、课程和选课数据库模式,该子模式还可以增加 Nation 和 Region 两个表。该模式中 Part 和 Supplier 之间是多对多类型的联系。

子模式 2:顾客订单子模式,包括 Customer、Orders 和 Lineitem 三个表,该子模式也可以增加 Nation 和 Region 两个表。该模式中各实体之间主要是一对多类型的联系。

后续各个实验项目的示例基于 TPC-H 数据模式,或者是基于上述两个子模式。

(1) 定义数据库

采用中文字符集创建名为 TPCH 的数据库。

```
CREATE DATABASE TPCH ENCODING = 'GBK';
```

(2) 定义模式

在数据库 TPCH 中创建名为 Sales 的模式。

```
CREATE SCHEMA Sales;
```

(3) 定义基本表

在 TPCH 数据库的 Sales 模式中创建 8 个基本表。

```
/*设置当前会话的搜索路径为 Sales 模式、Public 模式,基本表就会自动创建在 Sales 模式下。*/
SET SEARCH_PATH TO Sales, Public;
CREATE TABLE Region ( /*地区表*/
    regionkey INTEGER PRIMARY KEY,          /*地区编号*/
    name CHAR(25),                          /*地区名称*/
    comment VARCHAR(152)                    /*备注*/) ;

CREATE TABLE Nation ( /*国家表*/
    nationkey INTEGER PRIMARY KEY,          /*国家编号*/
    name CHAR(25),                          /*国家名称*/
    regionkey INTEGER REFERENCES Region(regionkey), /*地区编号*/
    comment VARCHAR(152)                    /*备注*/) ;

CREATE TABLE Supplier ( /*供应商基本表*/
    suppkey  INTEGER PRIMARY KEY,           /*供应商编号*/
    name CHAR(25),                          /*供应商名称*/
    address VARCHAR(40),                    /*供应商地址*/
    nationkey INTEGER REFERENCES Nation(nationkey), /*国家编号*/
    phone CHAR(15),                         /*供应商电话*/
    acctbal REAL,                           /*供应商账户余额 account balance*/
    comment VARCHAR(101)                    /*备注*/) ;

CREATE TABLE Part ( /*零件基本表*/
    partkey INTEGER PRIMARY KEY,            /*零件编号*/
    name VARCHAR(55),                       /*零件名称*/
    mfgr CHAR(25),                          /*制造厂*/
    brand CHAR(10),                         /*品牌*/
    type VARCHAR(25),                       /*零件类型*/
    size INTEGER,                           /*尺寸*/
    container CHAR(10),                     /*包装*/
    retailprice REAL,                       /*零售价格*/
```

```
    comment VARCHAR(23)                          /*备注*/);

CREATE TABLE PartSupp ( /*零件供应联系表*/
    partkey INTEGER REFERENCES Part(partkey),        /*零件编号*/
    suppkey INTEGER REFERENCES Supplier(suppkey),    /*供应商编号*/
    availqty INTEGER,                                /*可用数量*/
    supplycost REAL,                                 /*供应价格*/
    comment VARCHAR(199),                            /*备注*/
    PRIMARY KEY(partkey,suppkey)                     /*定义主码,表级约束*/);

CREATE TABLE Customer ( /*顾客表*/
    custkey INTEGER PRIMARY KEY,                     /*顾客编号*/
    name VARCHAR(25),                                /*姓名*/
    address VARCHAR(40),                             /*地址*/
    nationkey INTEGER REFERENCES Nation(nationkey),  /*国籍编号*/
    phone CHAR(15),                                  /*电话*/
    acctbal REAL,                                    /*账户余额*/
    mktsegment CHAR(10),                             /*市场分区*/
    comment VARCHAR(117)                             /*备注*/);

CREATE TABLE Orders ( /*订单表*/
    orderkey INTEGER PRIMARY KEY,                    /*订单编号*/
    custkey INTEGER REFERENCES Customer(custkey),    /*顾客编号*/
    orderstatus CHAR(1),                             /*订单状态*/
    totalprice REAL,                                 /*订单总金额*/
    orderdate DATE,                                  /*订单日期*/
    orderpriority CHAR(15),                          /*订单优先级别*/
    clerk CHAR(15),                                  /*记账员*/
    shippriority INTEGER,                            /*运输优先级别*/
    comment VARCHAR(79)                              /*备注*/);
/*其中 totalprice = SUM(Lineitem.extendedprice×(1-Lineitem.discount)×(1+ Lineitem.tax)) */

CREATE TABLE Lineitem ( /*订单明细表 Lineitem*/
    orderkey INTEGER REFERENCES Orders(orderkey),    /*订单编号*/
    partkey INTEGER REFERENCES Part(partkey),        /*零件编号*/
    suppkey INTEGER REFERENCES Supplier(suppkey),    /*供应商编号*/
    linenumber INTEGER,                              /*订单明细编号*/
    quantity REAL,                                   /*数量*/
```

```
    extendedprice REAL,                    /* 订单明细价格 */
    discount REAL,                         /* 折扣[0.00, 1.00] */
    tax REAL,                              /* 税率[0.00, 0.08] */
    returnflag CHAR(1),                    /* 退货标记 */
    linestatus CHAR(1),                    /* 订单明细状态 */
    shipdate DATE,                         /* 装运日期 */
    commitdate DATE,                       /* 委托日期 */
    receiptdate DATE,                      /* 签收日期 */
    shipinstruct CHAR(25),                 /* 装运说明,如 deliver in person */
    shipmode CHAR(10),                     /* 装运方式,如空运、陆运、铁运和海运等 */
    comment VARCHAR(44),                   /* 备注 */
    PRIMARY KEY(orderkey,linenumber),
    FOREIGN KEY (partkey,suppkey) REFERENCES PartSupp(partkey,suppkey) ) ;
    /* 其中订单明细价格 extendedprice = quantity × Part.retailprice */
```

实验总结：

① 初学者可以先定义零件供应商子模式，包括 Part、Supplier 和 PartSupp 三个基本表，类似学生、课程和选课数据库模式。

② 初学者可以先不定义实体完整性和参照完整性，待讲完有关概念后再在实验 3 中练习。

5. 思考题

（1）SQL 语法规定，双引号括定的符号串为对象名称，单引号括定的符号串为常量字符串，那么什么情况下需要用双引号来界定对象名呢？请实验验证。

（2）数据库对象的完整引用是“服务器名.数据库名.模式名.对象名”，但通常可以省略服务器名和数据库名，甚至模式名，直接用对象名访问对象即可。请设计相应的实验验证基本表及其列的访问方法。

实验 1.2 数据基本查询实验

1. 实验目的

掌握 SQL 程序设计基本规范，熟练运用 SQL 语言实现数据基本查询，包括单表查询、分组统计查询和连接查询。

2. 实验内容和要求

针对 TPC-H 数据库设计各种单表查询 SQL 语句、分组统计查询语句；设计单个表针对

自身的连接查询，设计多个表的连接查询。理解和掌握 SQL 查询语句各个子句的特点和作用，按照 SQL 程序设计规范写出具体的 SQL 查询语句，并调试通过。

说明：简单地说，SQL 程序设计规范包含 SQL 关键字大写、表名、属性名、存储过程名等标识符大小写混合、SQL 程序书写缩进排列等编程规范。

3. 实验重点和难点

实验重点：分组统计查询、单表自身连接查询、多表连接查询。

实验难点：区分元组过滤条件和分组过滤条件；确定连接属性，正确设计连接条件。

4. 实验报告示例

实 验 报 告				
题目：数据基本查询实验	姓名		日期	

（1）单表查询（实现投影操作）

查询供应商的名称、地址和联系电话。

```
SELECT name,address,phone
FROM Supplier;
```

（2）单表查询（实现选择操作）

查询最近一周内提交的总价大于 1 000 元的订单的编号、顾客编号等订单的所有信息。

```
SELECT *
FROM Sales.Orders      /* 模式名.表名，Sales 模式中的 Orders 表 */
WHERE CURRENT_DATE - orderdate < 7 AND totalprice > 1000;   /* 选择条件 */
```

（3）不带分组过滤条件的分组统计查询

统计每个顾客的订购金额。

```
SELECT C.custkey, SUM(O.totalprice)   /* 对每个组，作用聚集函数 SUM */
FROM Customer C, Orders O
WHERE C.custkey = O.custkey
GROUP BY C.custkey;   /* 按照 C.custkey 分组 */
```

（4）带分组过滤条件的分组统计查询

查询订单平均金额超过 1 000 元的顾客编号及其姓名。

```
SELECT C.custkey, MAX(C.name) /* 分组属性和聚集函数才能出现在 SELECT 子句 */
FROM Customer C, Orders O
WHERE C.custkey = O.custkey
GROUP BY C.custkey;   /* 按照 C.custkey 分组 */
HAVING AVG(O.totalprice) > 1000;   /* 按照条件对组进行过滤，只输出满足条件的组 */
```

(5) 单表自身连接查询

查询与"金仓集团"在同一个国家的供应商编号、名称和地址信息。

```
SELECT F.suppkey, F.name, F.address
FROM   Supplier F, Supplier S   /* Supplier 表的自身连接 */
WHERE F.nationkey = S.nationkey AND S.name = '金仓集团';
```

(6) 两表连接查询(普通连接)

查询供应价格大于零售价格的零件名、制造商名、零售价格和供应价格。

```
SELECT P.name, P.mfgr, P.retailoprice, PS.supplycost
FROM Part P, PartSupp PS   /* 两表连接,给表起个别名,简化表达 */
WHERE P.retailprice > PS.supplycost;   /* 限定条件 */
/* 说明:上述连接语句是从两个表的笛卡儿积中选出满足限定条件的元组,得到的结果可能
不是同一个商品的有关值,所以应改为下面的自然连接 */
```

(7) 两表连接查询(自然连接)

查询供应价格大于零售价格的零件名、制造商名、零售价格和供应价格。

```
SELECT P.name, P.mfgr, P.retailoprice, PS.supplycost
FROM Part P, PartSupp PS   /* 两表连接,给表起个别名,简化表达 */
WHERE P.partkey = PS.partkey   /* 连接条件 */
        AND P.retailprice > PS.supplycost; /* 限定条件 */
```

(8) 三表连接查询

查询顾客"苏举库"订购的订单编号、总价及其订购的零件编号、数量和明细价格。

```
SELECT O.orderkey, O.totalprice, L.partkey, L.quantity, L.extendedprice
FROM Customer C, Orders O, Lineitem L
WHERE C.custkey = O.custkey AND O.orderkey = L.orderkey AND C.name='苏举库';
```

实验总结:

① 正确理解数据库模式结构,才能正确设计数据库查询;

② 连接查询是数据库 SQL 查询中最重要的查询,连接查询的设计要特别注意,不同的查询表达,其查询执行的性能会有很大差别。

5. 思考题

(1) 不在 GROUP BY 子句出现的属性,是否可以出现在 SELECT 子句中? 请举例并上机验证。

(2) 请举例说明分组统计查询中的 WHERE 和 HAVING 有何区别?

(3) 连接查询速度是影响关系数据库性能的关键因素。请讨论如何提高连接查询速度,

并进行实验验证。

实验 1.3　数据高级查询实验

1. 实验目的

掌握 SQL 嵌套查询和集合查询等各种高级查询的设计方法等。

2. 实验内容和要求

针对 TPC-H 数据库，正确分析用户查询要求，设计各种嵌套查询和集合查询。

3. 实验重点和难点

实验重点：嵌套查询。

实验难点：相关子查询、多层 EXIST 嵌套查询。

4. 实验报告示例

实　验　报　告				
题目：数据高级查询实验	姓名		日期	

(1) IN 嵌套查询

查询订购了“海大”制造的“船舶模拟驾驶舱”的顾客。

```
SELECT   custkey, name
FROM   Customer
WHERE custkey IN (SELECT O.custkey
                  FROM Orders O, Lineitem L, PartSupp PS, Part P   /*四表连接*/
                  WHERE   O.orderkey = L.orderkey AND
                          L.partkey = PS.partkey AND
                          L.suppkey = PS.suppkey AND
                          PS.partkey = P.partkey AND
                          P.mfgr = '海大' AND P.name = '船舶模拟驾驶舱');
/*试比较上述查询与下列查询有何区别？下列查询中 Lineitem 表直接与 Part 表连接。*/
/*请见实验总结*/

SELECT   custkey, name
FROM   Customer
WHERE custkey IN (SELECT O.custkey
                  FROM Orders O, Lineitem L, Part P    /*三表连接*/
                  WHERE   O.orderkey = L.orderkey AND
                          L.partkey = P.partkey AND
                          P.mfgr = '海大' AND P.name = '船舶模拟驾驶舱');
```

(2) 单层 EXISTS 嵌套查询

查询没有购买过“海大”制造的“船舶模拟驾驶舱”的顾客。

```
SELECT  custkey, name
FROM  Customer C
WHERE NOT EXISTS(SELECT O.custkey /*购买“海大”制造的“船舶模拟驾驶舱”的顾客*/
                 FROM  Orders O, Lineitem L, PartSupp PS, Part P
                 WHERE  C.custkey = O.custkey AND  /*此条件为相关子查询条件*/
                        O.orderkey = L.orderkey AND
                        L.partkey = PS.partkey AND
                        L.suppkey = PS.suppkey AND
                        PS.partkey = P.partkey AND
                        P.mfgr = '海大' AND P.name = '船舶模拟驾驶舱');
```

(3) 双层 EXISTS 嵌套查询

查询至少购买过顾客“张三”购买过的全部零件的顾客姓名。

```
SELECT CA.name  /*查找 CA 客户,其不存在张三购买过而 CA 客户没有买过的零件*/
FROM Customer CA
WHERE NOT EXISTS
   (SELECT *   /*张三购买过而 CA 客户没有买过的零件*/
    FROM Customer CB, Orders OB, Lineitem LB
    WHERE CB.custkey = OB.custkey AND
          OB.orderkey = LB.orderkey AND
          CB.name = '张三' AND
          NOT EXISTS(SELECT *   /*CA 客户与 CB 客户都购买过的零件*/
                     FROM Orders OC, Lineitem LC
                     WHERE CA.custkey = OC.custkey AND
                     OC.orderkey = LC.orderkey AND
                     LB.suppkey = LC.suppkey AND
                     LB.partkey = LC.partkey ) );
```

(4) FROM 子句中的嵌套查询

查询订单平均金额超过 1 万元的顾客中的中国籍顾客信息。

```
SELECT  C.*
FROM Customer C,(SELECT custkey     /*子查询生成的临时派生表为 B 表*/
                 FROM  Orders
                 GROUP BY custkey
                 HAVING AVG(totalprice) > 10000) B, Nation N
```

```
WHERE   C.custkey = B.custkey AND /* B 表成为主查询的查询对象 */
        C.nationkey = N.nationkey AND   N.name = '中国';
```

(5) 集合查询(交)

查询顾客“张三”和“李四”都订购过的全部零件的信息。

```
SELECT P.*
FROM   Customer C,Orders O, Lineitem L, PartSupp PS, Part P
WHERE   C.custkey = O.custkey AND O.orderkey = L.orderkey AND
        L.suppkey = PS.suppkey AND L.partkey = PS.partkey AND
        PS.partkey = P.partkey AND C.name = '张三';
INTERSECTION
SELECT P.*
FROM Customer C,Orders O, Lineitem L, PartSupp PS, Part P
WHERE   C.custkey = O.custkey AND O.orderkey = L.orderkey AND
        L.suppkey = PS.suppkey AND L.partkey = PS.partkey AND
        PS.partkey = P.partkey AND C.name = '李四';
```

(6) 集合查询(并)

查询顾客“张三”和“李四”订购的全部零件的信息。

```
SELECT P.*
FROM   Customer C,Orders O, Lineitem L, PartSupp PS, Part P
WHERE   C.custkey = O.custkey AND O.orderkey = L.orderkey AND
        L.suppkey = PS.suppkey AND L.partkey = PS.partkey AND
        PS.partkey = P.partkey AND C.name = '张三';
UNION
SELECT P.*
FROM   Customer C,Orders O, Lineitem L, PartSupp PS, Part P
WHERE   C.custkey = O.custkey ANDO.orderkey = L.orderkey AND
        L.suppkey = PS.suppkey AND L.partkey = PS.partkey AND
        PS.partkey = P.partkey AND C.name = '李四';
```

(7) 集合查询(差)

顾客“张三”订购过而“李四”没订购过的零件的信息。

```
SELECT P.*
FROM   Customer C,Orders O, Lineitem L, PartSupp PS, Part P
WHERE   C.custkey = O.custkey AND O.orderkey = L.orderkey AND
        L.suppkey = PS.suppkey AND L.partkey= PS.partkey AND
```

```
        PS.partkey = P.partkey AND C.name = '张三';
EXCEPT
SELECT P.*
FROM Customer C,Orders O, Lineitem L, PartSupp PS, Part P
WHERE  C.custkey = O.custkey AND O.orderkey = L.orderkey AND
        L.suppkey = PS.suppkey AND L.partkey=PS.partkey AND
        PS.partkey = P.partkey AND C.name = '李四';
```

实验总结：

通过分析图 2 中 TPC-H 数据库模式可知，Lineitem 表是通过 Partsupp 表跟 Part 表联系的，所以，“(1) IN 嵌套查询”中第一个查询是正常的查询表达方式。而由于 partkey 是 Part 的主码，第二个查询也能得出相同的结果。因此，生成 Lineitem 记录时利用 PartSupp 表保证供应商和零件的一致性，而查询 Lineitem 时可以直接和 Part 相连接。同样，也可以直接和 Suppliers 相连接。

5. 思考题

(1) 试分析什么类型的查询可以用连接查询实现，什么类型的查询用只能用嵌套查询实现？

(2) 试分析不相关子查询和相关子查询的区别。

实验 1.4 数据更新实验

1. 实验目的

熟悉数据库的数据更新操作，能够使用 SQL 语句对数据库进行数据的插入、修改、删除操作。

2. 实验内容和要求

针对 TPC-H 数据库设计单元组插入、批量数据插入、修改数据和删除数据等 SQL 语句。理解和掌握 INSERT、UPDATE 和 DELETE 语法结构的各个组成成分，结合嵌套 SQL 子查询，分别设计几种不同形式的插入、修改和删除数据的语句，并调试成功。

3. 实验重点和难点

实验重点：插入、修改和删除数据的 SQL。

实验难点：与嵌套 SQL 子查询相结合的插入、修改和删除数据的 SQL 语句；利用一个表的数据来插入、修改和删除另外一个表的数据。

4. 实验报告示例

实 验 报 告				
题目:数据更新实验	姓名		日期	

(1) INSERT 基本语句(插入全部列的数据)

插入一条顾客记录,要求每列都给一个合理的值。

```
INSERT INTO Customer
VALUES(30, '张三', '北京市', 40, '010-51001199', 0.00, 'Northeast', 'VIP Customer');
```

(2) INSERT 基本语句(插入部分列的数据)

插入一条订单记录,给出必要的几个字段值。

```
INSERT INTO Lineitem(orderkey, Linenumber, partkey, suppkey, quantity, shipdate)
VALUES (862,ROUND(RANDOM() * 100,0), 479, 1, 10, '2012-3-6');
/* RANDOM()函数为随机小数生成函数,ROUND()为四舍五入函数 */
```

(3) 批量数据 INSERT 语句

① 创建一个新的顾客表,把所有中国籍顾客插入到新的顾客表中。

```
CREATE TABLE NewCustomer AS SELECT * FROM Customer WITH NO DATA;
/* WITH NO DATA 子句使得 SELECT 查询只生成一个结果模式,不查询出实际数据 */
INSERT INTO NewCustomer  /* 批量插入 SELECT 语句查询结果到 NewCustomer 表中 */
SELECT C.*
FROM Customer C, Nation N
WHERE C.nationkey = N.nationkey AND N.name = '中国';
```

② 创建一个顾客购物统计表,记录每个顾客及其购物总数和总价等信息。

```
CREATE TABLE ShoppingStat
(custkey   INTEGER ,
    quantity REAL,
    totalprice   REAL);
INSERT INTO ShoppingStat
SELECT C.custkey,Sum(L.quantity), Sum(O.totalprice)   /* 对分组后的数据求总和 */
FROM Customer C, Orders O, Lineitem L
WHERE C.custkey = O.custkey AND O.orderkey = L.orderkey
GROUP BY C.custkey
```

③ 倍增零件表的数据,多次重复执行,直到总记录数达到 50 万为止。

```
INSERT INTO Part
SELECT partkey + (SELECT COUNT(*) FROM Part),
```

```
        name, mfgr, brand, type, size, container, retailprice, comment
    FROM Part;
```

说明：

① 该方法是迅速生成大量数据的一个有效方法，但其缺点是要求主码为数值类型，并且除了主码外其他属性上没有 Unique 约束，因此该方法生成的数据具有很大的重复性。

② 生成 50 万记录需要执行上述 INSERT 语句的次数，需要根据 Part 表的初始记录数人工计算。

③ 执行上述 INSERT 语句的速度会一次比一次慢，因此需要耐心等候。

(4) UPDATE 语句(修改部分记录的部分列值)

"金仓集团"供应的所有零件的供应成本价下降 10%。

```
UPDATE PartSupp
SET supplycost = supplycost * 0.9
WHERE suppkey = (SELECT suppkey        /*找出要修改的那些记录*/
                 FROM Supplier
                 WHERE name = '金仓集团');
```

(5) UPDATE 语句(利用一个表中的数据修改另外一个表中的数据)

利用 Part 表中的零售价格来修改 Lineitem 中的 extendedprice，其中 extendedprice = Part.retailprice * quantity。

```
UPDATE Lineitem L
SET L.extendedprice = P.retailprice * L.quantity
FROM Part P
WHERE L.partkey = P.partkey
/*Lineitem 表也可以直接与 Part 表相连接，而不需通过 PartSupp 连接*/
```

(6) DELETE 基本语句(删除给定条件的所有记录)

删除顾客张三的所有订单记录。

```
DELETE FROM Lineitem    /*先删除张三的订单明细记录*/
WHERE orderkey IN (SELECT orderkey
                   FROM Orders O, Customer C
                   WHERE O.custkey = C.custkey AND C.name ='张三');
DELETE FROM  Orders   /*再删除张三的订单记录*/
WHERE custkey = (SELECT custkey
                 FROM Customer
                 WHERE name ='张三');
```

实验总结：

① 正确地设计和执行数据更新语句，确保正确地录入数据和更新数据，才能保证查询出来的数据正确。

② 当更新数据失败时，一个主要原因是更新数据时违反了完整性约束。

5. 思考题

（1）请分析数据库模式更新和数据更新 SQL 语句的异同。

（2）请分析数据库系统除了 INSERT、UPDATE 和 DELETE 等基本的数据更新语句之外，还有哪些可以用来更新数据库基本表数据的 SQL 语句？

实验 1.5　视图实验

1. 实验目的

熟悉 SQL 语言有关视图的操作，能够熟练使用 SQL 语句来创建需要的视图，定义数据库外模式，并能使用所创建的视图实现数据管理。

2. 实验内容和要求

针对给定的数据库模式，以及相应的应用需求，创建视图和带 WITH CHECK OPTION 的视图，并验证视图 WITH CHECK OPTION 选项的有效性。理解和掌握视图消解执行原理，掌握可更新视图和不可更新视图的区别。

3. 实验重点和难点

实验重点：创建视图。

实验难点：可更新的视图和不可更新的视图之区别，WITH CHECK OPTION 的验证。

4. 实验报告示例

实　验　报　告				
题目：视图实验	姓名		日期	

（1）创建视图（省略视图列名）

创建一个“海大汽配”供应商供应的零件视图 V_DLMU_PartSupp1，要求列出供应零件的编号、零件名称、可用数量、零售价格、供应价格和备注等信息。

```
CREATE VIEW V_DLMU_PARTSUPP1 AS /* 由 SELECT 子句目标列组成视图属性 */
SELECT P.partkey,P.name,PS.availqty,P.retailprice,PS.supplycost,P.comment
FROM   Part P,PartSupp PS,Supplier S
WHERE P.partkey=PS.partkey AND S.suppkey=PS.suppkey AND S.name='海大汽配';
```

(2) 创建视图(不能省略列名的情况)

创建一个视图 V_CustAvgOrder,按顾客统计平均每个订单的购买金额和零件数量,要求输出顾客编号,姓名,平均购买金额和平均购买零件数量。

```
CREATE VIEW V_CustAvgOrder(custkey,cname,avgprice,avgquantity) AS
SELECT C.custkey,MAX(C.name),AVG(O.totalprice),AVG(L.quantity)
FROM  Customer C, Orders O, Lineitem L
WHERE C.custkey = O.custkey AND L.orderkey = O.orderkey
GROUP BY C.custkey;
```

(3) 创建视图(WITH CHECK OPTION)

使用 WITH CHECK OPTION,创建一个“海大汽配”供应商供应的零件视图 V_DLMU_PartSupp2,要求列出供应零件的编号、可用数量和供应价格等信息。然后通过该视图分别增加、删除和修改一条“海大汽配”零件供应记录,验证 WITH CHECK OPTION 是否起作用。

```
CREATE VIEW V_DLMU_PartSupp2
AS
SELECT partkey, suppkey, availqty, supplycost
FROM PartSupp
WHERE suppkey = (SELECT suppkey
                 FROM Supplier
                 WHERE name = '海大汽配')
WITH CHECK OPTION;

INSERT INTO V_DLMU_PartSupp2
VALUES(58889,5048,704,77760);

UPDATE V_DLMU_PartSupp2
SET supplycost = 12
WHERE suppkey = 58889;

DELETE FROM V_DLMU_PartSupp2
WHERE suppkey = 58889;
```

(4) 可更新的视图(行列子集视图)

创建一个“海大汽配”供应商供应的零件视图 V_DLMU_PartSupp4,要求列出供应零件的编号、可用数量和供应价格等信息。然后通过该视图分别增加、删除和修改一条“海大汽配”零件供应记录,验证该视图是否是可更新的,并比较上述“(3) 创建视图”实验任务与本任务结果有何异同。

```
CREATE VIEW V_DLMU_PartSupp3
AS
SELECT partkey, suppkey, availqty, supplycost
FROM PartSupp
WHERE suppkey = (SELECT suppkey
FROM Supplier
WHERE name = '海大汽配');

INSERT INTO V_DLMU_PartSupp3
VALUES(58889,5048,704,77760);

UPDATE V_DLMU_PartSupp3
SET supplycost = 12
WHERE suppkey = 58889;

DELETE FROM V_DLMU_PartSupp3
WHERE suppkey = 58889;
```

(5) 不可更新的视图

(2)中创建的视图是可更新的吗？通过 SQL 更新语句加以验证，并说明原因。

```
INSERT INTO V_CustAvgOrder
VALUES(100000,NULL,20,2000);
```

(6) 删除视图(RESTRICT/CASCADE)

创建顾客订购零件明细视图 V_CustOrd，要求列出顾客编号、姓名、购买零件数、金额，然后在该视图的基础上，再创建(2)的视图 V_CustAvgOrder，然后使用 RESTRICT 选项删除视图 V_CustOrd，观察现象并解释原因。利用 CASCADE 选项删除视图 V_CustOrd，观察现象并检查 V_CustAvgOrder 是否存在，解释原因？

```
CREATE VIEW V_CustOrd(custkey,cname,qty,extprice)
AS
SELECT C.custkey, C.name, L.quantity, L.extendedprice
FROM Customer C,Orders O,Lineitem L
WHERE C.custkey = O.custkey AND O.orderkey = L.orderkey;

CREATE VIEW V_CustAvgOrder(custkey,cname, avgqty, avgprice)
AS
SELECT custkey,MAX(cname),AVG(qty),AVG(extprice)
FROM V_CustOrd    /*在视图 V_CustOrd 上再创建视图*/
GROUP BY custkey;
```

```
DROP VIEW V_CustOrd RESTRICT;
DROP VIEW V_CustOrd CASCADE;
```

实验总结：

5. 思考题

（1）请分析视图和基本表在使用方面有哪些异同，并设计相应的例子加以验证。

（2）请具体分析修改基本表的结构对相应的视图会产生何种影响？

实验 1.6 索引实验

1. 实验目的

掌握索引设计原则和技巧，能够创建合适的索引以提高数据库查询、统计分析效率。

2. 实验内容和要求

针对给定的数据库模式和具体应用需求，创建唯一索引、函数索引、复合索引等；修改索引；删除索引。设计相应的 SQL 查询验证索引有效性。学习利用 EXPLAIN 命令分析 SQL 查询是否使用了所创建的索引，并能够分析其原因，执行 SQL 查询并估算索引提高查询效率的百分比。要求实验数据集达到 10 万条记录以上的数据量，以便验证索引效果。

3. 实验重点和难点

实验重点：创建索引。

实验难点：设计 SQL 查询验证索引有效性。

4. 实验报告示例

实 验 报 告				
题目：索引实验	姓名		日期	

（1）创建唯一索引

在零件表的零件名称字段上创建唯一索引。

```
CREATE UNIQUE INDEX Idx_part_name ON Part (name);
```

（2）创建函数索引（对某个属性的函数创建索引，称为函数索引）

在零件表的零件名称字段上创建一个零件名称长度的函数索引。

```
CREATE INDEX Idx_part_name_fun ON Part (LENGTH(name));
```

(3) 创建复合索引(对两个及两个以上的属性创建索引,称为复合索引)

在零件表的制造商和品牌两个字段上创建一个复合索引。

```
CREATE UNIQUE INDEX Idx_part_mfgr_brand ON Part (mfgr, brand);
```

(4) * 创建聚簇索引

在零件表的制造商字段上创建一个聚簇索引。

```
CREATE UNIQUE INDEX Idx_part_mfgr ON Part (mfgr);
CLUSTER Idx_part_mfgr ON Part;
/* 有关聚簇索引的概念在《概论》第 7 章 7.5.2 小节"关系模式存取方法选择"中介绍 */
```

(5) 创建 Hash 索引

在零件表的名称字段上创建一个 Hash 索引。

```
CREATE INDEX Idx_part_name_hash ON Part USING HASH (name);
```

(6) 修改索引名称

修改零件表的名称字段上的索引名。

```
ALTER INDEX Idx_part_name_hash RENAME TO Idx_part_name_hash_new;
```

(7) 分析某个 SQL 查询语句执行时是否使用了索引

```
EXPLAIN SELECT * FROM part WHERE name = '零件';
```

(8) * 验证索引效率

创建一个函数 TestIndex,自动计算 SQL 查询执行的时间。

```
CREATE FUNCTION TestIndex(p_partname CHAR(55)) RETURN INTEGER
AS  /* 自定义函数 TestIndex():输入参数为零件名称,返回 SQL 查询的执行时间 */
DECLARE
    begintime   TIMESTAMP;
    endtime   TIMESTAMP;
    durationtime INTEGER;
BEGIN
    SELECT CLOCK_TIMESTAMP() INTO begintime; /* 记录查询执行的开始时间 */
    PERFORM * FROM Part WHERE name =p_partname;
                                        /* 执行 SQL 查询,不保存查询结果 */
    SELECT CLOCK_TIMESTAMP() INTO endtime; /* 记录查询执行的结束时间 */
    SELECT DATEDIFF('ms', begintime , endtime) INTO durationtime;
    RETURN durationtime;          /* 计算并返回查询执行时间,时间单位为毫秒 ms */
END;

 /* 查看当零件表 Part 数据规模比较小,并且无索引时的执行时间 */
 SELECT TestIndex('零件名称');
```

```
INSERT INTO Part  /* 不断倍增零件表的数据,直到 50 万条记录 */
SELECT partkey + (SELECT COUNT( * ) FROM Part),
    name, mfgr, brand, type, size, container, retailprice, comment
FROM Part;

/* 查看当零件表 Part 数据规模比较大,但无索引时的执行时间 */
SELECT TestIndex('零件名称');

CREATE INDEX part_name ON Part(name); /* 在零件表的零件名称字段上创建索引 */

/* 查看零件表 Part 数据规模比较大,有索引时的执行时间 */
SELECT TestIndex( );
```

比较上述各次执行时间,可计算出索引提高查询效率的百分比。

实验总结:

5. 思考题

在一个表的多个字段上创建的复合索引,与在相应的每个字段上创建的多个简单索引有何异同? 请设计相应的例子加以验证。

实验 2 安全性语言实验

安全性实验包含两个实验项目(参见表 5),其中 1 个为必修,1 个为选修。自主存取控制实验为设计性实验项目,审计实验为验证性实验项目。本节目前没有设置强制存取控制和数据加密等方面的实验项目。

表 5 "安全性语言"实验项目一览表

序号	实验项目名称	开设类别	难易程度	建议实验学时	对应教材章节
实验 2.1	自主存取控制实验	必修	适中	2	第 4 章 4.2
实验 2.2	审计实验	选修	适中	2	第 4 章 4.4

实验 2.1　自主存取控制实验

1. 实验目的

掌握自主存取控制权限的定义和维护方法。

2. 实验内容和要求

定义用户、角色，分配权限给用户、角色，回收权限，以相应的用户名登录数据库验证权限分配是否正确。选择一个应用场景，使用自主存取控制机制设计权限分配。可以采用两种方案。

方案一：采用 SYSTEM 超级用户登录数据库，完成所有权限分配工作，然后用相应用户名登录数据库以验证权限分配正确性；

方案二：采用 SYSTEM 用户登录数据库创建三个部门经理用户，并分配相应的权限，然后分别用三个经理用户名登录数据库，创建相应部门的 USER、ROLE，并分配相应权限。

下面的实验报告示例采用了实验方案一。验证权限分配之前，请备份好数据库；针对不同用户所具有的权限，分别设计相应的 SQL 语句加以验证。

3. 实验重点和难点

实验重点：定义角色，分配权限和回收权限。

实验难点：实验方案二实现权限的再分配和回收。

4. 实验报告示例

实 验 报 告				
题目：自主存取控制实验	姓名		日期	

设有一个企业，包括采购、销售和客户管理等三个部门，采购部门经理 David，采购员 Jeffery；销售部门经理 Tom，销售员 Jane；客户管理部门经理 Kathy，职员 Mike。该企业一个信息系统覆盖采购、销售和客户管理等三个部门的业务，其数据库模式为 TPC-H 数据模式。针对此应用场景，使用自主存取控制机制设计一个具体的权限分配方案。

（1）创建用户

① 为采购、销售和客户管理等三个部门的经理创建用户标识，要求具有创建用户或角色的权利。

```
CREATE USER David WITH CREATEROLE PASSWORD '123456';
CREATE USER Tom WITH CREATEROLE PASSWORD '123456';
CREATE USER Kathy WITH CREATEROLE PASSWORD '123456';
/*注意：CREATE USER 语句不是 SQL 标准，因此不同的 RDBMS 语法和内容相差甚远。
这里采用的是 Kingbase 创建用户语句*/
```

② 为采购、销售和客户管理等三个部门的职员创建用户标识和用户口令。

```
CREATE USER Jeffery WITH PASSWORD '123456';
CREATE USER Jane WITH PASSWORD' 123456';
CREATE USER Mike WITH PASSWORD '123456';
```

(2) 创建角色并分配权限

① 为各个部门分别创建一个查询角色,并分配相应的查询权限。

```
CREATE ROLE PurchaseQueryRole;
GRANT SELECT ON TABLE Part TO PurchaseQueryRole;
GRANT SELECT ON TABLE Supplier TO PurchaseQueryRole;
GRANT SELECT ON TABLE PartSupp TO PurchaseQueryRole;

CREATE ROLE SaleQueryRole;
GRANT SELECT ON TABLE Order TO SaleQueryRole;
GRANT SELECT ON TABLE LineItem TO SaleQueryRole;

CREATE ROLE CustomerQueryRole;
GRANT SELECT ON TABLE Customer TO CustomerQueryRole;
GRANT SELECT ON TABLE Nation TO CustomerQueryRole;
GRANT SELECT ON TABLE Region TO CustomerQueryRole;
```

② 为各个部门分别创建一个职员角色,对本部门信息具有查看、插入权限。

```
CREATE ROLE PurchaseEmployeeRole;
GRANT SELECT,INSERT ON TABLE Part TO PurchaseEmployeeRole;
GRANT SELECT,INSERT ON TABLE Supplier TO PurchaseEmployeeRole;
GRANT SELECT,INSERT ON TABLE PartSupp TO PurchaseEmployeeRole;

CREATE ROLE SaleEmployeeRole;
GRANT SELECT,INSERT ON TABLE Order TO SaleEmployeeRole;
GRANT SELECT,INSERT ON TABLE LineItem TO SaleEmployeeRole;

CREATE ROLE CustomerEmployeeRole;
GRANT SELECT,INSERT ON TABLE Customer TO CustomerEmployeeRole;
GRANT SELECT,INSERT ON TABLE Nation TO CustomerEmployeeRole;
GRANT SELECT,INSERT ON TABLE Region TO CustomerEmployeeRole;
```

③ 为各部门创建一个经理角色,相应角色对本部门的信息具有完全控制权限,对其他部门的信息具有查询权。经理有权给本部门职员分配权限。

```
CREATE ROLE PurchaseManagerRole WITH CREATEROLE;
GRANT ALL ON TABLE Part TO PurchaseManagerRole;
```

```
GRANT ALL ON TABLE Supplier TO PurchaseManagerRole;
GRANT ALL ON TABLE PartSupp TO PurchaseManagerRole;
GRANT SaleQueryRole TO PurchaseManagerRole;
GRANT CustomerQueryRole TO PurchaseManagerRole;

CREATE ROLE SaleManagerRole WITH CREATEROLE;
GRANT ALL ON TABLE Order TO SaleManagerRole;
GRANT ALL ON TABLE LineItem TO SaleManagerRole;
GRANT PurchaseQueryRole TO SaleManagerRole;
GRANT CustomerQueryRole TO SaleManagerRole;

CREATE ROLE CustomerManagerRole WITH CREATEROLE;
GRANT ALL ON TABLE Customer TO CustomerManagerRole;
GRANT ALL ON TABLE Nation TO CustomerManagerRole;
GRANT ALL ON TABLE Region TO CustomerManagerRole;
GRANT PurchaseQueryRole TO CustomerManagerRole;
GRANT SaleQueryRole TO CustomerManagerRole;
```

(3) 给用户分配权限

① 给各部门经理分配权限。

```
GRANT PurchaseManagerRole TO David WITH ADMIN OPTION;
GRANT SaleManagerRole TO Tom WITH ADMIN OPTION;
GRANT CustomerManagerRole TO Kathy WITH ADMIN OPTION;
```

② 给各部门职员分配权限。

```
GRANT PurchaseEmployeeRole TO Jeffery;
GRANT SaleEmployeeRole TO Jane;
GRANT CustomerEmployeeRole TO Mike;
```

(4) 回收角色或用户权限

① 收回客户经理角色的销售信息查看权限。

```
REVOKE SaleQueryRole FROM CustomerManagerRole;
```

② 回收 MIKE 的客户部门职员权限。

```
REVOKE CustomerEmployeeRole FROM Mike;
```

(5) 验证权限分配正确性

① 以 David 用户名登录数据库,验证采购部门经理的权限

```
SELECT * FROM Part;
DELETE * FROM Order;
```

② 回收 MIKE 的客户部门职员权限

```
SELECT * FROM Customer;
SELECT * FROM Part;
```

实验总结：

在进行权限分配之后，针对不同用户所具有的权限，设计并执行若干 SQL 语句，验证权限分配是否有效。

5. 思考题

（1）请分析 WITH CHECK OPTION、WITH GRANT OPTION 和 WITH ADMIN OPTION 有何区别与联系。

（2）请结合上述实验示例分析使用角色进行权限分配有何优缺点。

实验 2.2 审计实验

1. 实验目的

掌握数据库审计的设置和管理方法，以便监控数据库操作，维护数据库安全。

2. 实验内容和要求

打开数据库审计开关。以具有审计权限的用户登录数据库，设置审计权限，然后以普通用户登录数据库，执行相应的数据操纵 SQL 语句，验证相应审计设置是否生效，最后再以具有审计权限的用户登录数据库，查看是否存在相应的审计信息。

3. 实验重点和难点

实验重点：数据库对象级审计，数据库语句级审计。

实验难点：合理地设置各种审计信息。一方面，为了保护系统重要的敏感数据，需要系统地设置各种审计信息，不能留有漏洞，以便随时监督系统使用情况，一旦出现问题，也便于追查；另一方面，审计信息设置过多，会严重影响数据库的使用性能，因此需要合理设置。

4. 实验报告示例

实 验 报 告				
题目：审计实验	姓名		日期	

（1）审计开关

① 显示当前审计开关状态。

```
SHOW AUDIT_TRAIL;
```

② 打开审计开关。

```
SET AUDIT_TRAIL TO ON;
```

(2) 数据库操作审计

① 对客户信息表上的删除操作设置审计。

```
AUDIT DELETE ON Sales.Customer BY ACCESS;
/* BY ACCESS 审计方式表示系统对每个设置的审计操作都要进行记录;BY SESSION 审计方式表示对于每次会话中涉及的同类审计操作,系统只记录最早的一次。 */
```

② 以普通用户登录,执行 SQL 语句。

```
DELETE Sales.Customer WHERE custkey = 1011;
```

③ 查看数据库对象审计信息,验证审计设置是否生效。

```
SELECT * FROM SYS_AUDIT_OBJECT;
```

(3) 语句级审计

① 对表定义的更改语句 ALTER 设置审计。

```
AUDIT ALTER TABLE BY ACCESS;
```

② 查看数据库所有语句级审计设置,验证审计设置是否生效。

```
SELECT * FROM SYS_STMT_AUDIT_OPTS;
```

③ 以普通用户登录,执行 SQL 语句,验证审计设置是否生效。

```
ALTER TABLE Customer ADD COLUMN tt INT;
```

④ 查看所有审计信息。

```
SELECT * FROM SYS_AUDIT_TRAIL;
```

实验总结:

① 在 KingbaseES 系统中,只有审计管理员(SAO)才能进行对象级、语句级、系统权限级等审计权限设置。通过系统视图 SYS_STMT_AUDIT_OPTS 查看所有数据库中语句级审计设置,通过系统视图 SYS_AUDIT_OBJECT 查看数据库对象的审计设置。

② 通过系统视图 SYS_AUDIT_STATEMENT 可以查询所有关于 GRANT、REVOKE、AUDIT、NOAUDIT, 以及 ALTER SYSTEM 语句的审计结果;通过系统视图 SYS_AUDIT_OBJECT 可以查询数据库中所有对象的审计结果;通过系统视图 SYS_AUDIT_TRAIL 可以查看所有审计记录。

③ 审计语句不是标准 SQL 语句,所以不同的系统语句格式和语法不尽相同。

5. 思考题

试着设计一个例子,分析数据库审计对数据库性能的影响情况。

实验 3 完整性语言实验

完整性语言实验包含三个实验项目(参见表 6),其中 2 个必修项目,1 个选修项目。该实验的各个实验项目均为验证性实验项目。

表 6 "完整性语言"实验项目一览表

序号	实验项目名称	开设类别	难易程度	建议实验学时	对应教材章节
实验 3.1	实体完整性实验	必修	适中	2	第 5 章 5.1
实验 3.2	参照完整性实验	必修	适中	2	第 5 章 5.2
实验 3.3	用户自定义完整性试验	选修	适中	2	第 5 章 5.3

实验 3.1 实体完整性实验

1. 实验目的

掌握实体完整性的定义和维护方法。

2. 实验内容和要求

定义实体完整性,删除实体完整性。能够写出两种方式定义实体完整性的 SQL 语句:创建表时定义实体完整性、创建表后定义实体完整性。设计 SQL 语句验证完整性约束是否起作用。

3. 实验重点和难点

实验重点:创建表时定义实体完整性。

实验难点:有多个候选码时实体完整性的定义。

4. 实验报告示例

实 验 报 告				
题目:实体完整性实验	姓名		日期	

(1) 创建表时定义实体完整性(列级实体完整性)

定义供应商表的实体完整性。

```
CREATE TABLE Supplier(
    suppkey INTEGER CONSTRAINT PK_supplier PRIMARY KEY,
    name CHAR(25),
    address VARCHAR(40),
    nationkey INTEGER,
```

```
    phone CHAR(15),
    acctbal REAL,
    comment VARCHAR(101) );
```

(2) 创建表时定义实体完整性(表级实体完整性)

定义供应商表的实体完整性。

```
CREATE TABLE Supplier (
    suppkey INTEGER,
    name CHAR(25),
    address VARCHAR(40),
    nationkey INTEGER,
    phone CHAR(15),
    acctbal REAL,
    comment VARCHAR(101),
    CONSTRAINT PK_supplier PRIMARY KEY(suppkey));
```

(3) 创建表后定义实体完整性

定义供应商表。

```
CREATE TABLE Supplier (
    suppkey INTEGER,
    name CHAR(25),
    address VARCHAR(40),
    nationkey INTEGER,
    phone CHAR(15),
    acctbal REAL,
    comment VARCHAR(101) );

ALTER TABLE Supplier     /*再修改供应商表,增加实体完整性*/
ADD CONSTRAINT PK_Supplier PRIMARY KEY(suppkey);
```

(4) 定义实体完整性(主码由多个属性组成)

定义供应关系表的实体完整性。

```
CREATE TABLE PartSupp (
    partkey INTEGER,
    suppkey INTEGER,
    availqty INTEGER,
    supplycost REAL,
    comment VARCHAR(199),
    PRIMARY KEY(partkey,suppkey)) ;
```

```
    /* 主码由多个属性组成,实体完整性必须定义在表级 */
```

(5) 有多个候选码时定义实体完整性

定义国家表的实体完整性,其中 nationkey 和 name 都是候选码,选择 nationkey 作为主码,name 上定义唯一性约束。

```
CREATE TABLE nation (
    nationkey INTEGERCONSTRAINT PK_nation PRIMARY KEY,
    name CHAR(25)UNIQUE,
    regionkey INTEGER,
    comment VARCHAR(152) );
```

(6) 删除实体完整性

删除国家实体的主码。

```
ALTER TABLE nation DROP CONSTRAINT PK_nation;
```

(7) 增加两条相同记录,验证实体完整性是否起作用

```
/* 插入两条主码相同的记录就会违反实体完整性约束 */
INSERT INTO Supplier (suppkey,name,address,nationkey,phone,acctbal,comment )
        VALUES (11,'test1','test1',101,'12345678',0.0,'test1');
INSERT INTO Supplier (suppkey,name,address,nationkey,phone,acctbal,comment )
        VALUES (11,'test2','test2',102,'23456789',0.0,'test2');
```

实验总结:

5. 思考题

(1) 所有列级完整性约束都可以改写为表级完整性约束,而表级完整性约束不一定能改写成列级完整性约束。请举例说明。

(2) 什么情况下会违反实体完整性约束,DBMS 将做何种违约处理? 请用实验验证。

实验 3.2 参照完整性实验

1. 实验目的

掌握参照完整性的定义和维护方法。

2. 实验内容和要求

定义参照完整性,定义参照完整性的违约处理,删除参照完整性。写出两种方式定义参照完整性的 SQL 语句:创建表时定义参照完整性、创建表后定义参照完整性。

3. 实验重点和难点

实验重点:创建表时定义参照完整性。

实验难点:参照完整性的违约处理定义。

4. 实验报告示例

实 验 报 告				
题目:参照完整性实验	姓名		日期	

(1) 创建表时定义参照完整性

先定义地区表的实体完整性,再定义国家表上的参照完整性。

```
CREATE TABLE region(
  regionkey INTEGER PRIMARY KEY,
  name CHAR(25),
  comment VARCHAR(152));

CREATE TABLE nation(
  nationkey INTEGER PRIMARY KEY,
  name CHAR(25),
  regionkey INTEGER REFERENCES Region(regionkey),   /*列级参照完整性*/
  comment VARCHAR(152));
```

或者:

```
CREATE TABLE nation(
  nationkey INTEGER PRIMARY KEY,
  name CHAR(25),
  regionkey INTEGER,
  comment VARCHAR(152),
  CONSTRAINT FK_Nation_regionkey FOREIGN KEY (regionkey) REFERENCES Region(region-
  key));      /*表级参照完整性*/
```

(2) 创建表后定义参照完整性

定义国家表的参照完整性。

```
CREATE TABLE nation (
  nationkey INTEGER PRIMARY KEY,
  name CHAR(25),
  regionkey INTEGER,
  comment VARCHAR(152));

ALTER TABLE Nation
```

```
    ADD CONSTRAINT FK_Nation_regionkey
    FOREIGN KEY(regionkey) REFERENCES Region(regionkey );
```

(3) 定义参照完整性(外码由多个属性组成)

定义订单项目表的参照完整性。

```
CREATE TABLE PartSupp (
  partkey INTEGER,
  suppkey INTEGER,
  availqty INTEGER,
  supplycost REAL,
  comment VARCHAR(199),
  PRIMARY KEY(partkey,suppkey)) ;

CREATE TABLE Lineitem (
  orderkey INTEGER REFERENCES Orders(orderkey),
  partkey INTEGER REFERENCES Part(partkey),
  suppkey INTEGER REFERENCES Supplier(suppkey),
  linenumber INTEGER,
  quantity REAL,
  extendedprice REAL,
  discount REAL,
  tax REAL,
  returnflag CHAR(1),
  linestatus CHAR(1),
  shipdate DATE,
  commitdate DATE,
  receiptdate DATE,
  shipinstruct CHAR(25),
  shipmode CHAR(10),
  comment VARCHAR(44),
  PRIMARY KEY(orderkey,linenumber),
  FOREIGN KEY (partkey,suppkey) REFERENCES PartSupp(partkey,suppkey));
```

(4) 定义参照完整性的违约处理

定义国家表的参照完整性，当删除或修改被参照表记录时,设置参照表中相应记录的值为空值。

```
CREATE TABLE nation (
  nationkey INTEGER PRIMARY KEY,
```

```
    name CHAR(25),
    regionkey INTEGER,
    comment VARCHAR(152),
    CONSTRAINT FK_Nation_regionkey FOREIGN KEY (regionkey) REFERENCES Region(regionkey)
    ON DELETE SET NULL ON UPDATE SET NULL );
```

(5) 删除参照完整性

删除国家表的外码

```
ALTER TABLE nation DROP CONSTRAINT FK_Nation_regionkey;
```

(6) 插入一条国家记录,验证参照完整性是否起作用

```
/* 插入一条国家记录,如果'1001'号地区记录不存在,违反参照完整性约束。 */
INSERT INTONation(nationkey,name,regionkey,comment )
VALUES (1001,'nation1',1001,'comment1');
```

实验总结:

5. 思考题

对于自引用表,例如课程表(课程号、课程名、先修课程号,学分)中的先修课程号引用该表的课程号,请完成如下任务:

(1) 写出课程表上的实体完整性和参照完整性。

(2) 在考虑实体完整性约束的情况下,试举出几种录入课程数据的方法。

实验 3.3　用户自定义完整性实验

1. 实验目的

掌握用户自定义完整性的定义和维护方法。

2. 实验内容和要求

针对具体应用语义,选择 NULL/NOT NULL、DEFAULT、UNIQUE、CHECK 等,定义属性上的约束条件。

3. 实验重点和难点

实验重点:NULL/NOT NULL, DEFAULT。

实验难点:CHECK。

4. 实验报告示例

实 验 报 告				
题目:自定义完整性实验	姓名		日期	

(1) 定义属性 NULL/NOT NULL 约束

定义地区表各属性的 NULL/NOT NULL 属性。

```
CREATE TABLE region (
    regionkey INTEGER NOT NULL PRIMARY KEY,
    name CHAR(25) NOT NULL,
    comment VARCHAR(152)NULL ) ;
```

(2) 定义属性 DEFAULT 约束

定义国家表的 regionkey 的缺省属性值为 0 值,表示其他地区。

```
CREATE TABLE nation (
    nationkey INTEGER PRIMARY KEY,
    name CHAR(25),
    regionkey INTEGER DEFAULT 0,
    comment VARCHAR(152),
    CONSTRAINT FK_Nation_regionkey FOREIGN KEY (regionkey) REFERENCES Region(re-
    gionkey));
```

(3) 定义属性 UNIQUE 约束

定义国家表的名称属性必须唯一的完整性约束。

```
CREATE TABLE nation (
    nationkey INTEGER PRIMARY KEY,
    name CHAR(25) UNIQUE,
    regionkey INTEGER,
    comment VARCHAR(152));
```

(4) 使用 CHECK

使用 CHECK 定义订单项目表中某些属性应该满足的约束。

```
CREATE TABLE Lineitem (
    orderkey INTEGER REFERENCES Orders(orderkey),
    partkey INTEGER REFERENCES Part(partkey),
    suppkey INTEGER REFERENCES Supplier(suppkey),
    linenumber INTEGER,
    quantity REAL,
    extendedprice REAL,
```

```
    discount REAL,
    tax REAL,
    returnflag CHAR(1),
    linestatus CHAR(1),
    shipdate DATE,
    commitdate DATE,
    receiptdate DATE,
    shipinstruct CHAR(25),
    shipmode CHAR(10),
    comment VARCHAR(44),
    PRIMARY KEY(orderkey,linenumber),
    FOREIGN KEY (partkey,suppkey) REFERENCES PartSupp(partkey,suppkey),
    CHECK(shipdate < receiptdate ),          /*装运日期<签收日期*/
    CHECK(returnflag IN ('A','R','N') ) );    /*退货标记为A或R或N*/
```

(5) 修改 Lineitem 的一条记录验证是否违反 CHECK 约束

```
UPDATE sales.Lineitem SET shipdate = '2015-01-05', receiptdate = '2015-01-01'
WHERE orderkey = 5005 AND linenumber = 1;
```

实验总结：

5. 思考题

(1) 请分析哪些完整性约束只针对单个属性,哪些完整性约束可以针对多个属性？哪些只针对一个表,哪些针对多个表？

(2) 对表中某一列数据类型进行修改时,要修改的列是否必须为空列？

实验 4　触发器实验

触发器实验只有一个必修实验项目(参见表 7),该实验项目为设计型实验项目。

表 7　"触发器"实验项目一览表

实验项目名称	开设类别	难易程度	建议实验学时	对应教材章节
触发器实验	必修	较难	2	第 5 章 5.7

1. 实验目的

掌握数据库触发器的设计和使用方法。

2. 实验内容和要求

定义 BEFORE 触发器和 AFTER 触发器。能够理解不同类型触发器的作用和执行原理，验证触发器的有效性。

3. 实验重点和难点

实验重点：触发器的定义。

实验难点：利用触发器实现较为复杂的用户自定义完整性。

4. 实验报告示例

实 验 报 告				
题目：触发器实验	姓名		日期	

（1）AFTER 触发器

① 在 Lineitem 表上定义一个UPDATE 触发器，当修改订单明细（即修改订单明细价格 extendedprice、折扣 discount、税率 tax）时，自动修改订单 Orders 的 TotalPrice，以保持数据一致性。

```
total price = totalprice+extendedprice * (1-discount) * (1+tax) )
CREATE OR REPLACE TRIGGER TRI_Lineitem_Price_UPDATE
AFTER UPDATE OF extendedprice,discount,tax ON Lineitem
  FOR EACH ROW
  AS
  DECLARE
    L_valuediff REAL;
BEGIN
  /* 订单明细修改后,计算订单含税折扣价总价的修正值 */
  L_valuediff = NEW.extendedprice * (1-NEW.discount) * (1+NEW.tax) -
              OLD.extendedprice * (1-OLD.discount) * (1+OLD.tax);
              /* 更新订单的含税折扣价总价 */
  UPDATE Orders SET totalprice = totalprice + L_valuediff
  WHERE orderkey = NEW.orderkey;
END;
```

② 在 Lineitem 表上定义一个INSERT 触发器，当增加一项订单明细时，自动修改订单 Orders 的 TotalPrice，以保持数据一致性。

```
CREATE OR REPLACE TRIGGER TRI_Lineitem_Price_INSERT
```

```
AFTER INSERT ON Lineitem
    FOR EACH ROW
    AS
    DECLARE
      L_valuediff REAL;
BEGIN
  L_valuediff = NEW.extendedprice * (1 - NEW.discount) * (1 + NEW.tax);
                /* 增加订单明细项后,计算订单含税折扣价总价的修正值 */
  UPDATE Orders SET totalprice = totalprice + L_valuediff
                /* 更新订单的含税折扣价总价 */
  WHERE orderkey = NEW.orderkey;
END;
```

③ 在 Lineitem 表上定义一个DELETE 触发器,当删除一项订单明细时,自动修改订单 Orders 的 TotalPrice,以保持数据一致性。

```
CREATE OR REPLACE TRIGGER TRI_Lineitem_Price_DELETE
AFTER DELETE ON Lineitem
    FOR EACH ROW
    AS
    DECLARE
      L_valuediff REAL;
BEGIN
      L_valuediff = - OLD.extendedprice * (1 - OLD.discount) * (1 + OLD.tax);
                /* 删除订单明细项后,计算订单含税折扣价总价的修正值 */
      UPDATE Orders SET totalprice = totalprice + L_valuediff
                /* 更新订单的含税折扣价总价 */
WHERE orderkey = NEW.orderkey;
END;
```

④ 验证触发器 TRI_Lineitem_Price_UPDATE。

```
/* 查看 1854 号订单的含税折扣总价 totalprice */
SELECT totalprice
FROM Orders
WHERE orderkey = 1854;
/* 激活触发器:修改 1854 号订单第一个明细项的税率,该税率增加 0.5% */
UPDATE Lineitem SET tax = tax + 0.005
WHERE orderkey = 1854 AND linenumber = 1;
```

```
/* 再次查看 1854 号订单的含税折扣总价 totalprice 是否有变化,如有变化,则是触发器起作用了,否则触发器没有起作用 */
SELECT totalprice
FROM Orders
WHERE orderkey = 1854;
```

(2) BEFOR 触发器

① 在 Lineitem 表上定义一个BEFORE UPDATE 触发器,当修改订单明细中的数量(quantity)时,先检查供应表 PartSupp 中的可用数量 availqty 是否足够。

```
CREATE OR REPLACE TRIGGER TRI_Lineitem_Quantity_UPDATE
BEFORE UPDATE OF quantity ON Lineitem
    FOR EACH ROW
    AS
    DECLARE
        L_valuediff  INTEGER;
        L_availqty  INTEGER;
BEGIN
        /* 计算订单明细项修改时,订购数量的变化值 */
        L_valuediff = NEW.quantity - OLD.quantity;
        /* 查询当前订单明细项对应零件供应记录中的可用数量 */
        SELECT availqty INTO L_availqty
        FROM PartSupp
        WHERE partkey = NEW.partkey AND suppkey = NEW.suppkey;

    IF (L_availqty - L_valuediff  >= 0) THEN
          BEGIN
            /* 如果可用数量可以满足订单订购数量,则提示 ENOUGH */
            RAISE NOTICE 'Available quantity is ENOUGH';
            /* 修改当前订单明细项对应零件供应记录中的可用数量 */
            UPDATE PartSupp
            SET availqty = availqty - L_valuediff
            WHERE partkey = NEW.partkey AND suppkey = NEW.suppkey;
          END;
    ELSE
        /* 如果可用数量不能满足订单订购数量,则更新过程异常中断 */
        RAISE EXCEPTION 'Available quantity is NOT ENOUGH';
    END IF;
```

```
END;
```

② 在 Lineitem 表上定义一个BEFORE INSERT 触发器，当插入订单明细，先检查供应表 PartSupp 中的可用数量 availqty 是否足够。

```
CREATE OR REPLACE TRIGGER TRI_Lineitem_Quantity_UPDATE
BEFORE INSERT ON Lineitem
    FOR EACH ROW
    AS
    DECLARE
        L_valuediff   INTEGER;
        L_availqty   INTEGER;
BEGIN
        L_valuediff = NEW.quantity;      /*获得插入订单明细项的订购数量*/
        /*查询当前订单明细项对应零件供应记录中的可用数量*/
        SELECT availqty INTO L_availqty
        FROM PartSupp
        WHERE partkey = NEW.partkey AND suppkey = NEW.suppkey;

        IF (L_availqty - L_valuediff  >= 0) THEN
              BEGIN
                    /*如果可用数量可以满足订单订购数量,则提示 ENOUGH*/
                    RAISE NOTICE 'Available quantity is ENOUGH';
                    /*修改当前订单明细项对应零件供应记录中的可用数量*/
                    UPDATE PartSupp
                    SET availqty = availqty - L_valuediff
                    WHERE partkey = NEW.partkey AND suppkey = NEW.suppkey;
              END;
        ELSE
              /*如果可用数量不能满足订单订购数量,则插入过程异常中断。*/
              RAISE EXCEPTION 'Available quantity is NOT ENOUGH';
        END IF;
END;
```

③ 在 Lineitem 表上定义一个 BEFORE DELETE 触发器，当删除订单明细时，该订单明细项订购的数量要归还对应的零件供应记录。

```
CREATE OR REPLACE TRIGGER TRI_Lineitem_Quantity_UPDATE
BEFORE DELETE ON Lineitem
    FOR EACH ROW
```

```
    AS
    DECLARE
        L_valuediff  INTEGER;
        L_availqty   INTEGER;
BEGIN
        /* 获得删除订单明细项的订购数量 */
        L_valuediff = -OLD.quantity;
        /* 修改当前订单明细项对应零件供应记录中的可用数量 */
        UPDATE PartSupp
        SET availqty = availqty - L_valuediff
        WHERE partkey = NEW.partkey AND suppkey = NEW.suppkey;
END;
```

④ 验证触发器 TRI_Lineitem_Quantity_UPDATE。

```
/* 查看 1854 号订单第 1 个明细项的零件和供应商编号、订购数量、可用数量 */
SELECT L.partkey, L.suppkey, L.quantity, PS.availqty
FROM Lineitem L, PartSupp PS
WHERE L.partkey = PS.partkey AND L.suppkey = PS.suppkey AND
      L.orderkey = 1854 AND L.linenumber = 1;
/* 激活触发器:修改 1854 号订单第 1 个明细项的订购数量 */
UPDATE Lineitem SET  quantity = quantity + 5
WHERE orderkey = 1854  AND linenumber = 1;

/* 再次查看 1854 号订单第 1 个明细项的相关信息,以验证触发器是否起作用 */
SELECT L.partkey, L.suppkey, L.quantity, PS.availqty
FROM Lineitem L, PartSupp PS
WHERE L.partkey = PS.partkey AND L.suppkey = PS.suppkey AND
      L.orderkey = 1854 AND L.linenumber = 1;
```

(3) 删除触发器

删除触发器 TRI_Lineitem_Price_UPDATE。

```
DROP TRIGGER TRI_Lineitem_Price_UPDATE ON Lineitem;
```

实验总结:

5. 思考题

试设计一个 AFTER 触发器，当 Lineitem 表中的 quantity 变化时，自动计算 Lineitem 表中的 extendedprice 值，同时也要修改 PartSupp 中的 availqty 值（提示：extendedprice = quantity * Part.retailprice）。

实验 5　数据库设计实验

数据库设计和应用开发实验是一个大的实验项目，共包括 4 个实验，即实验 5、实验 6、实验 7 和实验 8。实验 5 只有一个必修实验项目（参见表 8），针对第 7 章数据库设计中数据库概念设计、逻辑设计和物理设计等内容进行实验。实验 6 和实验 7 针对第 8 章内容，实验 8 则是一个综合性的数据库设计与应用开发大作业。

表 8　“数据库设计”实验项目一览表

实验项目名称	开设类别	难易程度	建议实验学时	对应教材章节
数据库设计	必修	适中	4	第 7 章

1. 实验目的

掌握数据库设计基本方法及数据库设计工具。

2. 实验内容和要求

掌握数据库设计基本步骤，包括数据库概念结构设计、逻辑结构设计，物理结构设计，数据库模式 SQL 语句生成。能够使用数据库设计工具进行数据库设计。

3. 实验重点和难点

实验重点：概念结构设计、逻辑结构设计。

实验难点：逻辑结构设计。逻辑结构设计虽然可以按照一定的规则从概念结构转换而来，但是由于概念结构通常比较抽象，较少考虑更多细节，因此转换而成的逻辑结构还需要进一步调整和优化。逻辑结构承接概念结构和物理结构，处于核心地位，因而是数据库设计的重点，也是难点。

4. 实验报告示例

实　验　报　告				
题目：数据库设计实验	姓名		日期	
设计一个采购、销售和客户管理应用数据库。其中，一个供应商可以供应多种零件，一种零件也可以有多个供应商。一个客户订单可以订购多种供应商供应的零件。客户和供				

应商都分属不同的国家，而国家按世界五大洲八大洋划分地区。请利用 PowerDesigner 或者 ERwin 等数据库设计工具设计该数据库。

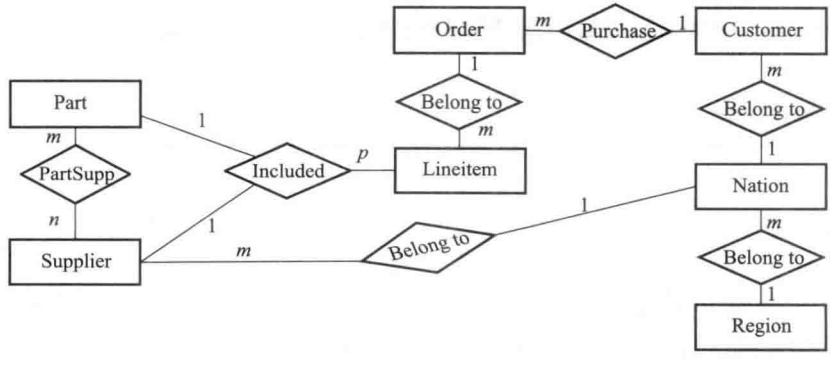
（1）数据库概念结构设计

识别出零件 Part、供应商 Supplier、客户 Customer、订单 Order、订单项 Lineitem、国家 Nation、地区 Region 等 7 个实体。每个实体的属性、码如下。

- 零件 Part：零件编号 partkey、零件名称 name、零件制造商 mfgr、品牌 brand、类型 type、大小 Size、零售价格 retailprice、包装 container、备注 comment。主码：零件编号 partkey。
- 供应商 Supplier：供应商编号 suppkey、供应商名称 name、地址 address、国籍 nation、电话 phone、备注 comment 等。主码：供应商编号 suppkey。
- 客户 Customer：客户编号 custkey、客户名称 name、地址 address、电话 phone、国籍 nation、备注 comment。主码：客户编号 custkey。
- 订单 Order：订单编号 orderkey、订单状态 status、订单总价 totalprice、订单日期 orderdate、订单优先级 orderpriority、记账员 clerk、运送优先级 shippriority、备注 comment。主码：订单编号 orderkey。
- 订单项 Lineitem：订单项编号 linenumber、所订零件号 partkey、所订零件供应商号 suppkey、零件数量 quantity、零件总价 extendedprice、折扣 discount、税率 tax、退货标记 returnflag 等。主码：订单项编号 linenumber。
- 国家 Nation：国家编号 nationkey、国家名称 name、所属地区 region、备注 comment。主码：国家编号 nationkey。
- 地区 Region：地区编号 regionkey、地区名称 name、备注 comment。主码：地区编号 regionkey。

根据实际语义，分析实体之间的联系，确定实体之间一对一，一对多和多对多联系。

实体-联系图（E-R 图）如下：



（2）数据库逻辑结构设计

按照数据库设计原理中概念结构转化成逻辑结构的规则，每个实体转换成一个关系，多对多的联系也转换成一个关系。因此，根据上述 E-R 图设计数据库逻辑结构（参见实验 1.1 数据库定义中的图 2 TPC-H 数据库模式图）。

（3）数据库物理结构设计

数据库物理结构首先根据逻辑结构自动转换生成，然后根据应用需求设计数据库的索引结构、存储结构。

（4）数据库模式 SQL 语句生成

生成 KingbaseES 数据库管理系统的 SQL 语句参见实验 1.1 数据库定义。

实验总结：

5. 思考题

试选择一个应用，练习数据库设计。

实验 6　存储过程实验

存储过程实验包含三个实验项目（参见表 9），其中 2 个必修实验项目，1 个选修实验项目，均为设计性实验项目。

表 9　“存储过程”实验项目一览表

序号	实验项目名称	开设类别	难易程度	建议实验学时	对应教材章节
实验 6.1	存储过程实验	必修	较难	2	第 8 章 8.2,8.3
实验 6.2	自定义函数实验	选修	较难	2	第 8 章 8.2,8.3
实验 6.3	游标实验	选修	较难	2	第 8 章 8.3

实验 6.1　存储过程实验

1. 实验目的

掌握数据库 PL/SQL 编程语言，以及数据库存储过程的设计和使用方法。

2. 实验内容和要求

存储过程定义,存储过程运行,存储过程更名,存储过程删除,存储过程的参数传递。掌握 PL/SQL 编程语言和编程规范,规范设计存储过程。

3. 实验重点和难点

实验重点:存储过程定义和运行。

实验难点:存储过程的参数传递方法。

4. 实验报告示例

实 验 报 告				
题目:存储过程实验	姓名		日期	

(1) 无参数的存储过程

① 定义一个存储过程,更新所有订单的(含税折扣价)总价。

```
/*即根据订单明细表,计算每个订单的总价,更新 ORDER 表*/
CREATE OR REPLACE PROCEDURE Proc_CalTotalPrice()
AS
BEGIN
  UPDATE Orders SET totalprice =
        (SELECT SUM(extendedprice * (1 - discount) * (1 + tax))
         FROM Lineitem
         WHERE Orders.orderkey = Lineitem.orderkey);
END;
```

② 执行存储过程 Proc_CalTotalPrice()

```
CALL Proc_CalTotalPrice();
```

(2) 有参数的存储过程

① 定义一个存储过程,更新给定订单的(含税折扣价)总价。

```
CREATE OR REPLACE PROCEDURE Proc_CalTotalPrice4Order(okey INTEGER)
AS
BEGIN
  UPDATE Orders SET totalprice =
        (SELECT SUM(extendedprice * (1 - discount) * (1 + tax))
         FROM Lineitem
         WHERE Orders.orderkey = Lineitem.orderkey AND
               Lineitem.orderkey = okey);
END;
```

② 执行存储过程。

```
CALL Proc_CalTotalPrice4Order(5365);  /*带参数的调用*/
```

(3) 有局部变量的存储过程

① 定义一个存储过程,更新某个顾客的所有订单的(含税折扣价)总价。

```
CREATE OR REPLACE PROCEDURE Proc_CalTotalPrice4Customer(p_custname CHAR(25))
AS
DECLARE
   L_custkey INTEGER;
BEGIN
  SELECT custkey INTO L_custkey   /*查找给定客户名对应的客户编号*/
  FROM Customer
  WHERE name =TRIM(p_custname);
                /*TRIM是系统函数,截去字符串前后空格*/
  UPDATE Orders SET totalprice =/*修改指定客户编号的所有订单的总价*/
         (SELECT SUM(extendedprice * (1 - discount) * (1 + tax))
          FROM Lineitem
          WHERE Orders.orderkey = Lineitem.orderkey AND
                Orders.custkey = L_custkey);
END;
```

② 执行存储过程 Proc_CalTotalPrice4Customer()。

```
CALL Proc_CalTotalPrice4Customer('陈楷丰');
```

③ 查看存储过程执行结果。

```
SELECT * FROM Orders
WHERE custkey = (SELECT custkey FROM Customer WHERE name = '陈楷丰');
```

(4) 有输出参数的存储过程

① 定义一个存储过程,更新某个顾客的所有订单的(含税折扣价)总价。

```
CREATE OR REPLACE PROCEDURE Proc_CalTotalPrice4Customer2(p_custname CHAR(25),
OUT p_totalprice REAL)
AS
DECLARE
   L_custkey INTEGER;
BEGIN
   SELECT custkey INTO L_custkey   /*查找给定客户名对应的客户编号*/
   FROM Customer
   WHERE name = trim(p_custname);
```

```
        RAISE NOTICE 'custkey is %', L_custkey;      /* 提示客户编号信息 */

        UPDATE Orders SET totalprice =       /* 修改指定客户编号的所有订单的总价 */
                (SELECT SUM(extendedprice * (1 - discount) * (1 + tax))
                 FROM Lineitem
                 WHERE Orders.orderkey =
                        Lineitem.orderkey AND Orders.custkey = L_custkey);
               /* 查找指定客户编号的所有订单的总价的和,并通过输出参数输出该值 */
        SELECT SUM(totalprice) INTO p_totalprice
        FROM Orders WHERE custkey = L_custkey;
    END;
```

② 执行存储过程 Proc_CalTotalPrice4Customer2()

```
/* 有输出参数的存储过程,调用的时候,生成临时表,以结果集的形式显示出来 */
CALL Proc_CalTotalPrice4Customer2('陈楷丰',null);
```

③ 查看存储过程执行结果

```
/* 检查下列 SQL 语句执行结果与上述存储过程执行结果是否一致 */
SELECT SUM(totalprice)
FROM Orders
WHERE custkey = (SELECT custkey
                 FROM Customer
                 WHERE name = '陈楷丰');
```

(5) 修改存储过程

① 修改存储过程名 Proc_CalTotalPrice4Order 为 CalTotalPrice4Order。

```
ALTER PROCEDURE Proc_CalTotalPrice4Order RENAME TO
CalTotalPrice4Order;
```

② 编译存储过程 CalTotalPrice4Order。

```
ALTER PROCEDURE CalTotalPrice4Order(okey INTEGER) COMPILE;
```

(6) 删除存储过程

删除存储过程 CalTotalPrice4Order。

```
DROP PROCEDURE CalTotalPrice4Order;
```

实验总结:

存储过程、用户自定义函数可以通过 CALL 和 SELECT 语句调用。需要说明的是:

① 存储过程、用户自定义函数如果带有 OUT 或 INOUT 参数,则参数对应位置在调用时必须使用 NULL 或其他常量占位。运行所得是一个结果集,结果集由一条或多条

RECORD 组成，每条 RECORD 中字段的顺序是 OUT 或 INOUT 参数对应的字段在前，最后返回 RETURN 语句对应的字段。

② SELECT 调用，就是执行普通的 SELECT 语句。对于存储过程，不能和其他任何常量、函数、存储过程等一并构成表达式使用，只能单独作为一个表达式出现在 SELECT 语句中。对于用户自定义函数，如果没有 OUT 或 INOUT 参数，可以和其他常量、变量、对象名如字段名等组合成表达式使用。带有 OUT 或 INOUT 参数的函数不可以参与表达式的计算。

5. 思考题

（1）试总结几种调试存储过程的方法。

（2）存储过程中的 SELECT 语句与普通的 SELECT 语句格式有何不同？执行方法有何不同？

实验 6.2　自定义函数实验

1. 实验目的

掌握数据库 PL/SQL 编程语言以及数据库自定义函数的设计和使用方法。

2. 实验内容和要求

自定义函数定义，自定义函数运行，自定义函数更名，自定义函数删除，自定义函数的参数传递。掌握 PL/SQL 和编程规范，规范设计自定义函数。

3. 实验重点和难点

实验重点：自定义函数的定义和运行。

实验难点：自定义函数的参数传递方法。

4. 实验报告示例

实 验 报 告				
题目：自定义函数实验	姓名		日期	

（1）无参数的自定义函数

① 定义一个自定义函数，更新所有订单的（含税折扣价）总价，并返回所有订单的总价之和。

```
/*该自定义函数与实验 6.1 中 Proc_CalTotalPrice()存储过程类似，区别在于该自定义函数具有一个 REAL 类型的返回值。*/
```

```
CREATE OR REPLACE FUNCTION FUN_CalTotalPrice( )
RETURN REAL
AS
DECLARE
    res  REAL;
BEGIN
    UPDATE Orders SET totalprice =     /* 更新所有订单的含税折扣价总价 */
           (SELECT SUM(extendedprice * (1-discount) * (1+tax))
            FROM Lineitem
            WHERE Orders.orderkey=Lineitem.orderkey);
     SELECT SUM(totalprice) INTO res   /* 计算所有订单的含税折扣价总价之和 */
     FROM Orders;
     RETURN res;      /* 返回总价之和 */
END;
```

② 执行自定义函数 FUN_CalTotalPrice( )。

```
SELECT FUN_CalTotalPrice( );
/* 执行自定义函数,其返回值以结果集的方式返回和显示。 */
```

(2) 有参数的自定义函数

① 定义一个自定义函数,更新并返回给定订单的总价。

```
CREATE OR REPLACE FUNCTION FUN_CalTotalPrice4Order(p_okey INTEGER)
RETURN REAL
AS
DECLARE
    res REAL;
BEGIN UPDATE Orders SET totalprice =    /* 更新给定编号的订单的含税折扣价总价 */
    (SELECT SUM(extendedprice * (1-discount) * (1+tax))
     FROM Lineitem
     WHERE Orders.orderkey=Lineitem.orderkey AND Lineitem.orderkey=p_okey);
     SELECT totalprice INTO res          /* 查找给定订单的总价 */
     FROM Orders;              WHERE orderkey=p_okey;
     RETURN res;       /* 返回给定订单的总价 */
END;
```

② 执行自定义函数 FUN_CalTotalPrice4Order( )

```
/* 更新并返回 5365 号订单的总价 */
        CALL FUN_CalTotalPrice4Order(5365);
```

(3) 有局部变量的自定义函数

① 定义一个自定义函数,计算并返回某个顾客的所有订单的总价。

```
CREATE OR REPLACE FUNCTION FUN_CalTotalPrice4Customer(p_custname CHAR(25))
RETURN REAL
AS
DECLARE
   L_custkey   INTEGER;        /*局部变量 L_custkey */
   res   REAL;
BEGIN
   SELECT custkey INTO L_custkey /* 查找给定客户名的客户编号 */
   FROM Customer
   WHERE name=trim(p_custname);

   RAISE NOTICE 'custkey is %',L_custkey; /* 提示客户编号信息 */

   /*更新指定客户编号的所有订单的含税折扣价总价 */
   UPDATE Orders SET totalprice=
          (SELECT SUM(extendedprice * (1-discount) * (1+tax))
           FROM Lineitem
           WHERE Orders.orderkey=Lineitem.orderkey AND
                 Orders.custkey=L_custkey);
   /* 计算指定客户编号的所有订单的含税折扣价总价之和 */
   SELECT SUM(totalprice) INTO res
   FROM Orders
   WHERE custkey=L_custkey;
   RETURN res;  /*返回总价之和*/
END;
```

② 执行自定义函数 FUN_CalTotalPrice4Customer()。

```
SELECT  FUN_CalTotalPrice4Customer('陈楷丰');
```

(4) 有输出参数的自定义函数

① 定义一个自定义函数,计算并返回某个顾客的所有订单的总价。

/*该函数定义一个输入参数 p_custname,一个输出参数 p_totalprice,还有一个返回值类型 REAL,通过输出参数的定义,该函数可以返回两个或者两个以上的值。该函数与 FUN_CalTotalPrice4Customer ()基本类似,区别只在于该函数多了一个输出参数。*/

```
CREATE OR REPLACE FUNCTION FUN_CalTotalPrice4Customer2(p_custname
CHAR(25),OUT p_totalprice REAL)
```

```
    RETURN REAL
    AS
    DECLARE
      L_custkey  INTEGER;
      res  REAL;
    BEGIN
      SELECT custkey INTO L_custkey
      FROM Customer
      WHERE name=trim(p_custname);

      RAISE NOTICE 'custkey is %',L_custkey;

      UPDATE Orders SET totalprice=
            (SELECT SUM(extendedprice * (1-discount) * (1+tax))
             FROM Lineitem
             WHERE Orders.orderkey=Lineitem.orderkey AND
                   Orders.custkey=L_custkey);

      SELECT SUM(totalprice) INTO p_totalprice
      FROM Orders WHERE custkey=L_custkey;

      Res:= p_totalprice;
      RETURN res;
    END;
```

② 执行自定义函数 FUN_CalTotalPrice4Customer2()。

```
SELECT FUN_CalTotalPrice4Customer2('陈楷丰',null);
```

(5) 修改自定义函数

① 修改自定义函数名 FUN_CalTotalPrice4Order 为 CalTotalPrice4Order。

```
ALTER FUNCTIONFUN_CalTotalPrice4Order RENAME TO CalTotalPrice4Order;
```

② 编译自定义函数 CalTotalPrice4Order。

```
ALTER FUNCTION CalTotalPrice4Order(okey INTEGER) COMPILE;
```

(6) 删除自定义函数

删除自定义函数 CalTotalPrice4Order。

```
DROP FUNCTION CalTotalPrice4Order;
```

实验总结：

5. 思考题

（1）试分析自定义函数与存储过程的区别与联系。

（2）如何使得自定义函数可以返回多个值？如何利用？

实验 6.3　游标实验

1. 实验目的

掌握 PL/SQL 游标的设计、定义和使用方法，理解 PL/SQL 游标按行操作和 SQL 按结果集操作的区别和联系。

2. 实验内容和要求

游标定义、游标使用。掌握各种类型游标的特点、区别与联系。

3. 实验重点和难点

实验重点：游标定义和使用。

实验难点：游标类型。

4. 实验报告示例

实 验 报 告				
题目：游标实验	姓名		日期	

（1）普通游标

① 定义一个存储过程，用游标实现计算所有订单的总价。

```
CREATE OR REPLACE PROCEDURE ProcCursor_CalTotalPrice( )
AS  /* 定义存储过程,循环计算和更新每个订单的总价 totalprice */DECLARE
   L_orderkey    INTEGER;
   L_totalprice   REAL;
   CURSOR mycursor FOR    /* 定义下面 SELECT 语句的游标 */
      SELECT orderkey,totalprice
      FROM Orders;
BEGIN
   OPEN mycursor;    /* 打开游标 */
   LOOP   /* 对每个订单记录循环,修改其 totalprice 属性值 */
      FETCH mycursor INTO L_orderkey,L_totalprice; /* 获取游标当前记录 */
      IF mycursor%NOTFOUND THEN
         EXIT;      /* 游标找不到记录,则退出 */
      END IF;
```

```
        /* 计算当前订单所有明细项目的含税折扣价格总和 */
        SELECT SUM(extendedprice * (1-discount) * (1+tax)) INTO L_totalprice
        FROM Lineitem
        WHERE orderkey = L_orderkey;
        /* 修改当前订单的 totalprice 属性值 */
        UPDATE Orders SET totalprice = L_totalprice
        WHERE orderkey = L_orderkey;
    END LOOP;
    CLOSE mycursor;  /* 关闭游标 */
END;
```

② 执行存储过程 ProcCursor_CalTotalPrice()。

```
CALL  ProcCursor_CalTotalPrice();
```

(2) REFCURSOR 类型游标

① 定义一个存储过程,用游标实现计算所有订单的总价。

```
/* 该存储过程与 ProcCursor_CalTotalPrice()存储过程类似,唯一区别就是所定义的游标类型不同 */
CREATE OR REPLACE PROCEDURE ProcRefCursor_CalTotalPrice()
AS
DECLARE
        L_orderkey   INTEGER;
        L_totalprice  REAL;
        mycursor REFCURSOR;          /* 定义 REFCURSOR 类型游标 */
BEGIN
        /* 打开 REFCURSOR 类型游标,游标记录集为 Orders 订单表所有记录 */
        OPEN mycursor FOR SELECT orderkey,totalprice FROM Orders;
        LOOP
            FETCH mycursor INTO L_orderkey,L_totalprice;
            IF mycursor%NOTFOUND THEN
              EXIT;
            END IF;
            SELECT SUM(extendedprice * (1-discount) * (1+tax))  INTO L_totalprice
            FROM Lineitem
            WHERE orderkey = L_orderkey;

            UPDATE Orders SET totalprice = L_totalprice
            WHERE orderkey = L_orderkey;
```

```
    END LOOP;
    CLOSE mycursor;
  END;
```

② 执行存储过程 ProcRefCursor_CalTotalPrice()。

```
CALL ProcRefCursor_CalTotalPrice();
```

(3) 记录变量与游标

① 定义一个存储过程,用游标实现计算所有订单的总价。

```
/*该存储过程与 ProcCursor_CalTotalPrice()存储过程类似,唯一区别就是所定义的游标结果类型为记录类型的变量*/
  CREATE OR REPLACE PROCEDURE ProcRecCursor_CalTotalPrice()
  AS
  DECLARE
    L_totalprice   REAL;
    res RECORD;                    /*定义游标结果为记录类型变量*/
    CURSOR mycursor FOR
        SELECT orderkey,totalprice
        FROM Orders;
    BEGIN
      OPEN mycursor;
      LOOP
        /*从游标中取出的结果保存在 res 记录类型的变量中*/
        FETCH mycursor INTO res;
        IF mycursor%NOTFOUND THEN EXIT;
        END IF;
        /*从记录变量 res 中获得当前订单的订单编号,计算当前订单的所有明细项目的含税折扣价总和*/
        SELECT SUM(extendedprice * (1-discount) * (1+tax)) INTO L_totalprice
        FROM Lineitem
        WHERE orderkey=res.orderkey;

        UPDATE Orders SET totalprice=L_totalprice
        WHERE orderkey =res.orderkey;
      END LOOP;
      CLOSE mycursor;
  END;
```

② 执行存储过程 ProcRecCursor_CalTotalPrice( )

```
CALL ProcRecCursor_CalTotalPrice( );
```

(4) 带参数的游标

① 定义一个存储过程,用游标实现计算指定国家的用户订单的总价。

```
/* 该存储过程与 ProcCursor_CalTotalPrice( )存储过程类似,区别在于:该存储过程带有一个参数、所定义的游标为带参数的游标、游标结果定义为记录类型。 */
CREATE OR REPLACE PROCEDURE ProcParaCursor_CalTotalPrice( p_nationname CHAR(20) )
AS
DECLARE
  L_totalprice   REAL;
  res    RECORD;          /* 定义游标结果为记录类型变量 */
  /* 定义一个带参数的游标,查询指定国家名称的顾客的订单及其总价 */
  CURSOR mycursor( c_nationname CHAR(20) ) FOR
          SELECT O.orderkey, O.totalprice
          FROM Orders O, Customer C, Nation N
          WHERE O.custkey = C.custkey AND c.nationkey = N.nationkey AND
                  TRIM( N.name) = TRIM( c_nationname) ;
BEGIN
  /* 打开游标:把存储过程的参数 p_nationname 传递给游标的参数 c_nationname */
  OPEN mycursor( p_nationname) ;
  LOOP
      FETCH mycursor INTO res;
      IF mycursor % NOTFOUND THEN
          EXIT;
      END IF;
      SELECT SUM( extendedprice * (1-discount) * (1+tax) ) INTO L_totalprice
      FROM Lineitem
      WHERE orderkey = res.orderkey;

      UPDATE Orders SET totalprice = L_totalprice
      WHERE orderkey = res.orderkey;
    END LOOP;
  CLOSE mycursor;
END;
```

② 执行存储过程 ProcParaCursor_CalTotalPrice( )。

```
CALL ProcParaCursor_CalTotalPrice('中国');      /* 计算中国用户订单的总价 */
```

实验总结：

① REFCURSOR 类型的游标定义一个游标引用变量，只是在打开该类型游标时才指定具体的 SELECT 语句以便产生游标的结果集。因此 REFCURSOR 实质上是定义了一个动态游标，可以灵活方便地根据程序运行时情况动态设置游标的 SELECT 查询结果集。

② 从任务(1)可以看出，游标可以实现对数据库记录逐条处理，而不是整个结果集一起处理，因此游标是在 PL/SQL 语言中实现过程化处理的核心功能。

③ 从任务(3)看出，记录变量对于游标结果记录的处理很方便，通过记录变量可以直接访问记录的每个属性，而无须为记录的每个属性定义相应的变量。

④ PL/SQL 语言中%TYPE、%ROWTYPE 和 RECORD 等数据类型的区别与联系如下：

- %TYPE 提供一个表字段数据类型的变量，例如在 users 表里面有一个字段叫 user_id，要声明一个和 users.user_id 类型相同的变量，可以写为 user_id users.user_id%TYPE。缺点是：通过使用 %TYPE，必须知道所引用的结构的数据类型。优点是：如果被引用项的表字段数据类型变化了也不需要修改 PL/SQL 的过程体定义。
- %ROWTYPE，一个复合类型变量，叫做行变量。这样的一个变量可以保存一次 SELECT 命令结果的完整一行，只要命令的字段集匹配该变量声明的类型。一个行变量可以声明为和一个现有的表或者视图的行类型相同，方法是使用 table_name%ROWTYPE 表示。在一个行类型的变量中，只可以访问用户定义的表中行的属性。
- RECORD 记录变量类似行类型变量，但是它们没有预定义的结构，而是在 SELECT 命令中获取实际的行结构，这点和 Oracle 不同，在 KingbaseES 的 PL/SQL 的过程体中，正是这种预先没有确定结构的特性为用户提供了极大的灵活性。

5. 思考题

试分析说明 REFCURSOR 类型游标的优点。

实验 7　数据库应用开发实验

数据库应用开发实验分为 2 个选修实验项目(参见表 10)。该实验项目可以任选其一，也可以单独作为一个大作业，或者与数据库设计实验一起作为一个大作业布置给学生，让学生充分利用课外时间完成该大作业。该实验项目为综合型实验项目。

表 10 “数据库应用开发”实验项目一览表

序号	实验项目名称	开设类别	难易程度	建议实验学时	对应教材章节
实验 7.1	基于 ODBC 的数据库应用开发实验	选修	较难	4	第 8 章 8.4
实验 7.2	基于 JDBC 的数据库应用开发实验	选修	较难	4	第 8 章 8.6

实验 7.1 基于 ODBC 的数据库应用开发实验

1. 实验目的

掌握基于 ODBC 驱动的数据库应用开发方法。

2. 实验内容和要求

设置 ODBC 驱动数据源,基于 ODBC 驱动的数据库连接方法,实现数据库数据操纵等应用开发常见功能。

3. 实验重点和难点

实验重点:基于 ODBC 驱动的数据库连接方法、数据库数据操纵功能等。

实验难点:不同的数据库应用开发工具具有不同的开发框架和模式。能够较为熟练地使用所选择的应用开发工具,是实现本实验的难点。

4. 实验报告示例

实 验 报 告				
题目:基于 ODBC 的数据库应用开发实验	姓名		日期	
在本实验中,以 VC++6.0 开发环境、KingbaseES 和 SQL Server 数据库为例,实现一个完整的示例程序,其源代码 ODBCTest.c 附在思考题之后。该程序实现把 KingbaseES 的 Samples 数据库中 S-C 模式下的 Student 表数据复制到 SQL Server 的 SamplesBackup 数据库中相同模式下的 Student 表中。为简单起见,也可以在 KingbaseES 中创建 SamplesBackup 数据库,建立 S-C 模式和 Student 表,然后实现上述复制功能。该程序的框架如下。				

（1）配置 ODBC 驱动

配置 KingbaseES ODBC 驱动数据源，共有两种方法：

方法一：运用数据源管理工具来进行配置。打开 Winodws 的“开始/设置/控制面板/管理工具/数据源(ODBC)”，出现“ODBC 数据源管理器”界面，点击“添加”按钮，出现“创建数据源”界面，选择希望安装数据源的驱动程序类别为“KingbaseES ODBC Driver”，出现“建立新的数据源到 KingbaseES”界面，然后设置好数据源的名称、描述信息及服务器的名称等参数即可。同样的方法，可以设置 SQL Server 数据库中的数据源。

方法二：使用 Driver Manager 提供的 ConfigDsn 函数来增加、修改或删除数据源。这种方法特别适用于在应用程序中创建的临时使用的数据源。

（2）基于 ODBC 驱动的数据库连接方法

```
/* Step 1 :定义句柄和变量 */
/* Step 2 :初始化环境 */
/* Step 3 :建立连接 */
```

（3）基于 ODBC 驱动的数据库数据操纵方法

```
/* Step 4 :初始化语句句柄 */
/* Step 5 :两种方式执行语句 */
    /*预编译带有参数的语句 */
    /*直接执行 SQL 语句*/
/* Step 6:处理结果集并执行预编译后的语句*/
```

（4）中断基于 ODBC 驱动的数据库连接

```
/* Step 7 中止处理*/
```

实验总结：

5. 思考题

（1）试修改源程序 ODBCTest.c，实现 TPC-H 数据库 SALES 模式下所有表数据的复制。

（2）请调查目前比较流行的软件开发环境在基于 ODBC 驱动开发数据库应用方面各有什么优缺点？

VC++6.0 环境中编程实现访问数据库的源程序 ODBCTest.c：

```
#include <stdlib.h>
#include <stdio.h>
#include <windows.h>
#include <sql.h>

#include <sqlext.h>
#include <sqltypes.h>
#define SNO_LEN 30
#define NAME_LEN 50
#define DEPART_LEN 100
#define SSEX_LEN 5
int main( )
{
    /* Step 1 :定义句柄和变量 */
    //以 king 开头的表示的是连接 KINGBASEES 的变量
    //以 server 开头的表示的是连接 SQLSERVER 的变量
     SQLHENV kinghenv,serverhenv;       //环境句柄
     SQLHDBC kinghdbc,serverhdbc;       //连接句柄
     SQLHSTMT kinghstmt,serverhstmt;    //语句句柄
     SQLRETURN ret;                     //结果返回集
     SQLCHAR sName[ NAME_LEN ],sDepart[ DEPART_LEN ],sSex[ SSEX_LEN ],
             sSno[ SNO_LEN ];
     SQLINTEGER sAge;
     SQLINTEGER cbAge = 0,cbSno = SQL_NTS,cbSex = SQL_NTS,
                cbName = SQL_NTS,cbDepart = SQL_NTS;
      /* Step 2 :初始化环境 */
      //分配环境句柄
      ret = SQLAllocHandle( SQL_HANDLE_ENV,SQL_NULL_HANDLE,&kinghenv) ;
      ret = SQLAllocHandle( SQL_HANDLE_ENV,SQL_NULL_HANDLE,&serverhenv) ;
      //设置管理环境的属性
      ret = SQLSetEnvAttr( kinghenv,SQL_ATTR_ODBC_VERSION,( void * )SQL_OV_ODBC3,0) ;
      Vret = SQLSetEnvAttr( serverhenv,SQL_ATTR_ODBC_VERSION,( void * )SQL_OV_ODBC3,0) ;
      /* Step 3 :建立连接 */
      //分配连接句柄
      ret = SQLAllocHandle( SQL_HANDLE_DBC,kinghenv,&kinghdbc) ;
      ret = SQLAllocHandle( SQL_HANDLE_DBC,serverhenv,&serverhdbc) ;
      ret = SQLConnect( kinghdbc,                    //连接 KingbaseES
                  "KingbaseES ODBC",SQL_NTS,
```

```
                    "SYSTEM",SQL_NTS,
                    "MANAGER",SQL_NTS);
if(!SQL_SUCCEEDED(ret))                    //连接失败时返回错误值
    return-1;
ret=SQLConnect(serverhdbc,                 //连接 SQLServer
                    "SQLServer",SQL_NTS,
                    "sa",SQL_NTS,
                    "sa",SQL_NTS);
if(!SQL_SUCCEEDED(ret))
    return-1;                              //连接失败时返回错误值
/* Step 4 :初始化语句句柄 */
ret=SQLAllocHandle(SQL_HANDLE_STMT,kinghdbc,&kinghstmt);
ret=SQLSetStmtAttr(kinghstmt,SQL_ATTR_ROW_BIND_TYPE,(SQLPOINTER)SQL_BI
    ND _BY_COLUMN,SQL_IS_INTEGER);//设置语句选项
ret=SQLAllocHandle(SQL_HANDLE_STMT,serverhdbc,&serverhstmt);
/* Step 5 :两种方式执行语句 */
/*预编译带有参数的语句 */
//需要多次执行插入,因此预先声明插入语句
ret=SQLPrepare (serverhstmt,"INSERT INTO STUDENT
                (SNO,SNAME,SSEX,SAGE,SDEPT) VALUES (?,?,?,?,?)"
                SQL_NTS);
if(ret == SQL_SUCCESS || ret == SQL_SUCCESS_WITH_INFO)
{   ret=SQLBindParameter(serverhstmt,1,SQL_PARAM_INPUT,     //绑定参数
    SQL_C_CHAR,SQL_CHAR,SNO_LEN,0,sSno,0,&cbSno);
    ret=SQLBindParameter(serverhstmt,2,SQL_PARAM_INPUT,
    SQL_C_CHAR,SQL_CHAR,NAME_LEN,0,sName,0,&cbName);
    ret=SQLBindParameter(serverhstmt,3,SQL_PARAM_INPUT,
    SQL_C_CHAR,SQL_CHAR,2,0,sSex,0,&cbSex);
    ret=SQLBindParameter(serverhstmt,4,SQL_PARAM_INPUT,
    SQL_C_LONG,SQL_INTEGER,0,0,&sAge,0,&cbAge);
    ret=SQLBindParameter(serverhstmt,5,SQL_PARAM_INPUT,
    SQL_C_CHAR,SQL_CHAR,DEPART_LEN,0,sDepart,0,&cbDepart);
}
/*执行 SQL 语句*/
ret=SQLExecDirect(kinghstmt,"SELECT * FROM STUDENT",SQL_NTS);
if(ret == SQL_SUCCESS || ret == SQL_SUCCESS_WITH_INFO)
    {//将结果集中的属性列一一绑定至变量
```

```
    ret=SQLBindCol(kinghstmt,1,SQL_C_CHAR,sSno,SNO_LEN,&cbSno);
    ret=SQLBindCol(kinghstmt,2,SQL_C_CHAR,sName,NAME_LEN,&cbName);
    ret=SQLBindCol(kinghstmt,3,SQL_C_CHAR,sSex,SSEX_LEN,&cbSex);
    ret=SQLBindCol(kinghstmt,4,SQL_C_LONG,&sAge,0,&cbAge);
    ret=SQLBindCol(kinghstmt,5,SQL_C_CHAR,sDepart,DEPART_LEN,&cbDepart);
    }
/* Step 6 :处理结果集并执行预编译后的语句 */
while ((ret=SQLFetch(kinghstmt)) !=SQL_NO_DATA_FOUND)
    {
      if(ret== SQL_ERROR)                    //错误处理
          printf("Fetch error\n");
      else ret=SQLExecute(serverhstmt);       //执行语句
    }
/* Step 7 :中止处理 */
SQLFreeHandle(SQL_HANDLE_STMT,kinghstmt);
SQLDisconnect(kinghdbc);
SQLFreeHandle(SQL_HANDLE_DBC,kinghdbc);
SQLFreeHandle(SQL_HANDLE_ENV,kinghenv);
SQLFreeHandle(SQL_HANDLE_STMT,serverhstmt);
SQLDisconnect(serverhdbc);
SQLFreeHandle(SQL_HANDLE_DBC,serverhdbc);
SQLFreeHandle(SQL_HANDLE_ENV,serverhenv);
return 0;
}
```

实验 7.2 基于 JDBC 的数据库应用开发实验

1. 实验目的

掌握基于 JDBC 驱动的数据库应用开发方法。

2. 实验内容和要求

基于 JDBC 驱动的数据库连接方法,实现数据库数据操纵等应用开发常见功能。

3. 实验重点和难点

实验重点:基于 JDBC 驱动的数据库连接方法、数据库数据操纵功能等。

实验难点:不同的数据库应用开发工具具有不同的开发框架和模式。能够较为熟练地使用所选择的应用开发工具,是实现本实验的难点。

4. 实验报告示例

<table>
<tr><td colspan="5">实 验 报 告</td></tr>
<tr><td>题目：基于 JDBC 的数据库应用开发实验</td><td>姓名</td><td></td><td>日期</td><td></td></tr>
<tr><td colspan="5">在本实验中，以 Eclipse 开发环境和 KingbaseES 数据库为例，实现一个完整的示例程序，其源代码 JDBCTest.java 附在思考题之后。该程序实现把 KingbaseES 的 Samples 数据库中 S-C 模式下的 Student 表数据复制到 SQL Server 的 SamplesBackup 数据库中相同模式的相同表中。为简单起见，也可以在 KingbaseES 中创建 SameplesBackup 数据库，建立 S-C 模式和 Student 表，然后实现上述复制功能。该程序的框架如下：
（1）基于 JDBC 驱动的数据库连接方法
/* Step 1 定义句柄和变量 */
/* Step 2 初始化环境 */
/* Step 3 :建立连接 */
（2）基于 JDBC 驱动的数据库数据操纵方法
/* Step 4 :初始化语句句柄 */
/* Step 5 :两种方式执行语句 */
/* 预编译带有参数的语句 */
/* 直接执行 SQL 语句 */
/* Step 6:处理结果集并执行预编译后的语句 */
（3）中断基于 JDBC 驱动的数据库连接
/* Step 7 中止处理 */</td></tr>
<tr><td colspan="5">实验总结：</td></tr>
</table>

5. 思考题

（1）试修改源程序 JDBCTest.java，实现 TPC-H 数据库 SALES 模式下所有表的复制。

（2）请调查目前比较流行的软件开发环境在基于 JDBC 驱动开发数据库应用方面各有哪些优缺点？

Eclipse 环境中编程实现访问数据库的源程序 JDBCTest.java：

```
import java.sql.Connection;
```

```
import java.sql.DriverManager;
import java.sql.Statement;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.Properties;

public class JDBCTest {
    public static void main(String[ ] args) {

        /* Step 1 :定义句柄和变量 */
        String sSno,sName,sSex,sDepart;
        Integer sAge;

        /* Step 2 :初始化环境 */
        /* Step 3 :建立连接 */

        String driverName="com.kingbase.Driver";                    //连接源数据库
        String src_url="jdbc:kingbase: //localhost:54321/SAMPLES";
        Properties src_info=new Properties( );
        src_info.put ("user","SYSTEM");
        src_info.put ("password","MANAGER");
        src_info.put ("EscapeProcessing","false");

        String dst_url="jdbc:kingbase://localhost:54321/SameplesBackup";  //连接目标数据库
        Properties dst_info=new Properties( );
        dst_info.put ("user","SYSTEM");
        dst_info.put ("password","MANAGER");
        dst_info.put ("EscapeProcessing","false");

        try{
            Class.forName( driverName);
            Connection src_conn=DriverManager.getConnection( src_url,src_info);
            Connection dst_conn=DriverManager.getConnection( dst_url,dst_info);

            /* Step 4 :初始化语句句柄 */
            Statement src_stmt=src_conn.createStatement( );
            PreparedStatement dst_prestmt=dst_conn.prepareStatement ("INSERT INTO \"S-C\".
                                                                    STUDENT (SNO, SNAME,
                                                                    SSEX,SAGE,SDEPT) VAL-
                                                                    UES (?,?,?,?,?)");
```

```
/* Step 5 :两种方式执行语句 */
/* 预编译带有参数的语句 */
/* 直接执行 SQL 语句 */
ResultSet rs = src_stmt.executeQuery( "SELECT * FROM \"S-C\".Student" );

/* Step 6 :处理结果集并执行预编译后的语句 */
while ( rs.next( ) )
  {
    sSno = rs.getString( "SNO" );
    sName = rs.getString( "SNAME" );
    sSex = rs.getString( "SSEX" );
    sAge = rs.getInt( "SAGE" );
    sDepart = rs.getString( "SDEPT" );

    dst_prestmt.setString( 1, sSno );
    dst_prestmt.setString( 2, sName );
    dst_prestmt.setString( 3, sSex );
    dst_prestmt.setInt( 4, sAge );
    dst_prestmt.setString( 5, sName );

    dst_prestmt.executeUpdate( );

    System.out.println( sSno+sName+sSex+sDepart+sAge );
  }
/* Step 7 :中止处理 */
  rs.close( );
  src_stmt.close( );
  dst_prestmt.close( );
  src_conn.close( );
  dst_conn.close( );
}
catch ( Exception e )
{
    System.out.println( e.getMessage( ) );
}
  }
}
```

实验 8 数据库设计与应用开发大作业

实验 8 是数据库设计与应用开发大作业(参见表 11),是一个综合型的实验项目。

表 11 "数据库事务管理"实验项目一览表

实验项目名称	开设类别	难易程度	建议实验学时	对应教材章节
数据库设计与应用开发大作业	必修	较难	8	第 3-8 章

1. 实验目的

掌握综合运用数据库原理、方法和技术进行数据库应用系统分析、设计和开发的能力。

2. 实验内容和要求

为某个部门或单位开发一个数据库应用系统,具体内容包括:对某个部门或单位业务和数据进行调查,系统分析,系统设计,数据库设计,数据库创建和数据加载,数据库应用软件开发,系统测试,系统分析设计和开发文档撰写,软件、文档和数据库提交,数据库应用系统运行演示和大作业汇报。

能够针对某个部门或单位的应用需求,通过系统分析,从数据库数据和应用系统功能两方面进行综合设计,实现一个完整的数据库应用系统。同时培养团队合作精神。要求 5~6 位同学组成一个开发小组,每位同学承担不同角色(如项目管理员、DBA、系统分析员、系统设计员、系统开发员、系统测试员)。撰写系统设计和开发文档;提交系统文档、数据库应用软件和数据库。每个小组进行 60 分钟的报告和答辩,讲解设计方案,演示系统运行,汇报分工与合作情况。

3. 实验重点和难点

实验重点:数据库设计,数据库应用软件开发。

实验难点:综合运用系统分析与设计方法,从数据和功能两方面协调设计一个完整的数据库应用系统。熟练掌握和运用一个主流数据库应用开发工具进行数据库应用软件开发。

4. 实验报告示例

实 验 报 告				
题目:数据库设计与应用开发大作业	姓名		日期	
在"数据库系统概论"精品课程网站上有一些大作业实验报告示例,可供读者参考学习。				

数据库设计和应用开发步骤主要包括：

① 系统调查

② 系统分析

③ 系统设计

④ 数据库设计

⑤ 数据库创建和数据加载

⑥ 数据库应用软件的功能设计和开发

⑦ 数据库应用系统测试

⑧ 分析设计和开发文档撰写

⑨ 应用软件、数据库和文档提交

⑩ 应用系统演示和汇报

实验总结：

5. 思考题

① 数据库应用系统的功能设计对数据库设计有何影响？

② 如何协调数据库应用系统的功能设计与数据库设计？

③ 数据库应用开发中使用的程序设计语言的数据类型与数据库的数据类型如何互操作？

实验 9　数据库监视与性能优化

数据库监视与性能优化实验包括三个选修实验项目(参见表 12),为验证性与设计性相结合的实验项目。

表 12　“数据库监视与性能优化”实验项目一览表

序号	实验项目名称	开设类别	难易程度	建议实验学时	对应教材章节
实验 9.1	数据库查询性能调优实验	选修	难	4	第 9 章
实验 9.2	数据库性能监视实验	选修	难	4	第 9 章
实验 9.3	数据库系统配置参数调优实验	选修	难	4	第 9 章

实验 9.1 数据库查询性能调优实验

1. 实验目的

理解和掌握数据库查询性能调优的基本原理和方法。

2. 实验内容和要求

学会使用 EXPLAIN 命令分析查询执行计划、利用索引优化查询性能、优化 SQL 语句,以及理解和掌握数据库模式规范化设计对查询性能的影响。能针对给定的数据库模式,设计不同的实例验证查询性能优化效果。

3. 实验重点和难点

实验重点:利用索引优化查询性能、优化 SQL 语句。

实验难点:数据库模式规范化设计对查询性能的影响。

4. 实验报告示例

实 验 报 告				
题目:数据库查询性能调优	姓名		日期	

(1) 使用 EXPLAIN 命令查看查询执行计划

查看 Part、PartSupp、Supplier 三个表连接查询的查询执行计划。

```
EXPLAIN  SELECT *
FROM Part P,Partsupp PS,Supplier S
WHERE P.partkey=PS.partkey AND PS.suppkey=S.suppkey
        AND P.name='发动机'
ORDER BY S.acctbal desc,S.name,P.partkey;
/*该 SQL 语句是要查询零件名为"发动机"的零件、供应该零件的供应商以及供应等详细信息,
并按照供应商的账户余额(降序)、供应商名称(升序)、零件编号(升序),排序输出结果*/
```

(2) 利用索引优化查询性能

建立索引,优化 SQL 查询性能。

```
CREATE INDEX IDX_part_name ON Part(name); /*在 Part 表的 name 属性上建立索引*/

EXPLAIN  SELECT *
FROM Part P,Partsupp PS,Supplier S
WHERE P.partkey=PS.partkey AND PS.suppkey=S.suppkey
        AND P.name='发动机'
ORDER BY S.acctbal desc,S.name,P.partkey;
```

比较在 Part 表的 name 上有索引和无索引时，两种执行计划有何异同，并实际执行该查询，验证有索引和无索引时此查询语句的执行性能。

(3) 优化 SQL 语句

① IN 与 EXISTS 查询

```
SELECT *
FROM Orders
WHERE orderkey IN(SELECT orderkey
                  FROM  Lineitem
                  WHERE partkey IN(SELECT partkey
                                   FROM  Part
                                   WHERE name='发动机'));
```

一般地，使用 EXISTS 查询效率要高于 IN 查询，改写 SQL 语句如下：

```
SELECT *
FROM Orders O
WHERE EXISTS(SELECT *
             FROM Lineitem L
             WHERE O.orderkey=L.orderkey AND
                   EXISTS(SELECT *
                          FROM Part P
                          WHERE P.partkey=L.partkey AND name='发动机'));
```

比较两种执行计划，并实际测试执行性能哪种情况好。

② 尽可能使用不相关子查询，避免使用相关子查询。不相关子查询一般比相关子查询执行效率要高，在可能的情况下，改写相关子查询为不相关子查询。

查找这样的订单，其总价大于该顾客所购商品的平均总价。

相关子查询：

```
SELECT *
FROM Orders O1
WHERE O1.totalprice > (SELECT AVG(O2.totalprice)
                       FROM Orders O2
                       WHERE O2.custkey=O1.custkey)
```

不相关子查询：

```
SELECT *
FROM Orders O1,(SELECT O2.custkey,AVG(O2.totalprice) AS avgprice
FROM Orders O2
GROUP BY O2.custkey) AVG1
```

```
/*子查询生成临时派生表,取名为 AVG1*/
    WHERE O1.custkey=AVG1.custkey AND O1.totalprice > AVG1.avgprice;
```

比较两种执行计划,并实际测试执行性能哪种情况好。

(4) 数据库模式规范化设计对查询性能的影响

分析 TPC-H 数据库模式中是否存在不规范化的设计。该设计在海量数据的情况下查询效率怎样? 如何在设计上进一步提高海量数据的查询效率?

第三范式在一定程度上减少了不必要的冗余,提高了数据库的查询效率,但是如果数据量大且需要大量联合查询的时候,第三范式设计又可能会影响查询效率。TPC-H 中存在的不规范的设计如下:

① Orders 表中订单的 totalprice 由 Lineitem 表中该订单的所有订单项的 extendedprice、discount 和 tax 等属性计算得出,即 totalprice = SUM(Lineitem.extendedprice×(1-Lineitem.discount)×(1+Lineitem.tax)

该项设计虽然不规范,但大大提高了客户所有订单总金额的查询效率。例如,查询所有购物总金额大于 20 万的客户编号及其购物总金额:

```
SELECT custkey,SUM(totalprice)
FROM Orders
GROUP BY custkey
HAVING SUM(totalprice) > 200000.00;
```

如果不用 Totalprice 字段,SQL 查询语句则为:

```
SELECT O.custkey,SUM(L.extendedprice * (1-L.discount) * (1+L.tax)) AS sumprice
FROM Orders O,Lineitem L
WHERE O.orderkey=L.orderkey
GROUP BY O.custkey
HAVING sumprice > 200000.00;
```

② Lineitem 表中订单明细价格 extendedprice 由 quantity 和 Part 表中的 retailprice 得出,即 extendedprice = quantity× Part.retailprice。

该项设计虽然不规范,但大大提高了客户订单总金额的查询效率。例如,查询所有购物零售总金额(折扣和加税之前的价格,即 extendedprice)大于 1 万的订单对应的客户编号及金额:

```
SELECT O.custkey,SUM(L.extendedprice) AS sumextprice
FROM Orders O,Lineitem L
WHERE O.orderkey=L.orderkey
GROUP BY O.custkey
HAVING sumextprice > 10000.00;
```

如果不使用 extendedprice,SQL 查询则为:

```
SELECT O.custkey,SUM(L.quantity * P.retailprice) AS sumextprice
FROM Orders O,Lineitem L,PartSupp PS,Part P
WHERE O.orderkey=L.orderkey AND L.partkey=PS.partkey
      AND L.suppkey=PS.suppkey AND PS.partkey=P.partkey
GROUP BY O.custkey
HAVING sumextprice > 10000.00;
```

实验总结:

5. 思考题

对实验 1.3 中的查询,使用不同的 SQL 语句来表达,比较它们的查询效率,体会并总结查询优化的技巧和方法。

实验 9.2　数据库性能监视实验

1. 实验目的

了解所使用的 DBMS 提供的数据库性能监视功能,学习数据库查询性能监视的基本原理和方法。

2. 实验内容和要求

使用 KingbaseES 的数据库性能监视工具,通过标准统计视图和统计访问函数查看数据库系统收集到的性能统计信息、ANALYZE 更新数据库统计信息,通过专门工具监视系统系性能。希望能够熟悉数据库系统有关性能统计信息的标准视图和统计访问函数,了解如何通过系统收集到的性能数据监视系统性能。

3. 实验重点和难点

实验重点:通过标准统计视图和统计访问函数查看数据库系统收集到的性能统计信息。

实验难点:分析性能统计信息,找出存在的性能问题。

4. 实验报告示例

实　验　报　告				
题目:数据库性能监视实验	姓名		日期	
(1) 查看系统标准统计视图和统计访问函数 ① KingbaseES 部分标准统计视图				

sys_stat_activity：每个服务器进程一行，显示数据库、进程、用户、当前查询等信息。只有在打开stats_command_string参数时，才能得到报告当前查询的相关信息的各个字段。

sys_stat_database：每个数据库一行，显示数据库与该数据库连接的活跃服务器进程数、提交的事务总数以及在该数据库中回滚数目的总数、读取的磁盘块的总数，以及缓冲区命中的总数。

sys_stat_all_tables：当前数据库中每个表一行，显示模式和表信息、发起的顺序扫描的总数、顺序扫描抓取的有效数据行的数目、发起的索引扫描的总数、索引扫描抓取的有效数据行的数目，以及插入、更新和删除的行的总数。

sys_stat_all_indexes：对当前数据库的每个索引，显示输入索引所在表和索引，模式、表和索引名，包括使用了该索引的索引扫描总数、索引扫描返回的索引记录的数目、使用该索引的简单索引扫描抓取的有效的表中数据行数。

sys_statio_all_tables：当前数据库中每个表一行，显示表、模式和表名，包含从该表中读取的磁盘块总数、缓冲区命中的次数、在该表上所有索引的磁盘块读取和缓冲区命中总数等信息。

sys_statio_all_indexes：当前数据库中每个索引一行，显示表和索引OID、模式、表和索引名、该索引的磁盘块读取和缓冲区命中的数目。

sys_statio_all_sequences：当前数据库中每个序列对象一行，显示序列的OID、模式和序列名、序列磁盘读取和缓冲区命中的数目。

② KingbaseES部分统计访问函数

sys_stat_get_db_numbackends(oid),integer,数据库活跃的服务器进程数目

sys_stat_get_db_xact_commit(oid),bigint,数据库中已提交的事务数量

sys_stat_get_db_xact_rollback(oid),bigint,数据库中回滚的事务数量

sys_stat_get_tuples_inserted(oid),bigint,插入表中的元组数量

sys_stat_get_tuples_updated(oid),bigint,在表中已更新的元组数量

sys_stat_get_tuples_deleted(oid),bigint,从表中删除的元组数量

sys_stat_get_blocks_fetched(oid),bigint,表或者索引的磁盘块被存取的数量

sys_backend_pid(),integer,附着在当前会话上的服务器进程ID

sys_stat_get_backend_pid(integer),integer,给出的服务器进程的进程号

sys_stat_get_backend_dbid(integer),oid,指定服务器进程的数据库ID

sys_stat_get_backend_userid(integer),oid,指定服务器进程的用户ID

sys_stat_get_backend_activity(integer),text,服务器进程的当前活跃查询

sys_stat_get_backend_client_port(integer),integer,连接到给定服务器的客户端口

sys_stat_reset(),boolean,重置所有当前收集的统计

(2) 查看数据库管理系统当前活动情况

① 查看当前进程数

```
SELECT COUNT( * )
```

```
    FROM (SELECT sys_stat_get_backend_idset() AS backendid) AS s;
    /* sys_stat_get_backend_idset() 函数为:获得服务器后台进程集合 */
```

② 查看当前进程的详细活动

```
SELECT * FROM sys_stat_activity;
```

③ 查看正在进行的 SQL 操作

```
SELECT sys_stat_get_backend_pid(s.backendid) AS procpid,
       sys_stat_get_backend_activity(s.backendid) AS current_query
FROM (SELECT sys_stat_get_backend_idset() AS backendid) AS s ;
```

(3) 更新数据库统计信息

① 用 ANALYZE 更新数据库统计信息

```
ANALYZE;
```

② 用 ANALYZE 更新数据表的统计信息

```
ANALYZE Sales.Orders;
```

实验总结:

5. 思考题

试总结所用的 DBMS 的性能监视工具的功能和使用方法。

实验 9.3　数据库系统配置参数调优实验

1. 实验目的

了解数据库系统级参数和连接级参数的配置和调优的基本原理和方法。了解用户可以通过修改这些参数设置来调整系统运行时配置,以优化系统性能。

2. 实验内容和要求

熟悉和了解数据库各级参数的作用以及配置,包括系统级参数配置和调优、数据库级参数配置和调优、会话(连接)级参数配置和调优。

3. 实验重点和难点

实验重点:数据库级参数配置和调优、连接级参数配置和调优。

实验难点:系统级参数配置和调优。

4. 实验报告示例

实 验 报 告				
题目:数据库系统配置参数调优实验	姓名		日期	

(1) 显示参数的命令

① 显示 XXX 参数。

```
SHOW  shared_buffers;  /* 专门的显示系统会话值的命令,可以显示所有参数的值 */
```

② 显示 xxx 参数。

```
SELECT  shared_buffers;  /* 只能显示一个参数的值 */
```

(2) 设置会话级参数

① 设置提交延迟时间参数为 10 s,对本次会话有效。

```
SET SESSION COMMIT_DELAY to 10;
```

② 设置提交延迟时间参数为 10 s,对本次提交事务有效。

```
SET LOCAL COMMIT_DELAY to 10;
```

(3) 修改数据库默认参数

设置 test 数据库的密码长度为 10 个字符,该设置覆盖数据库的默认设置,对该数据库上启动的每个会话都有效。

```
ALTER DATABASE test SET password_length TO 10;
```

(4) 设置系统级或全局级参数配置

① 关闭自动清理存储空间开关,自动修改数据库系统配置文件,该修改将在重启数据库服务器后才生效。

```
ALTER SYSTEM SET autovacuum = off
```

② 设置数据库监听的地址,自动修改数据库系统配置文件,该修改将在重启数据库服务器后才生效。

```
ALTER SYSTEM SET listen_addresses = '127.0.0.1'
```

(5) 优化系统级参数

优化 KingbaseES 数据库管理系统的 shared_buffers 参数

```
ALTER SYSTEM SET shared_buffers = 80000;
```

该参数是很重要的参数,数据库管理系统通过 shared_buffers 与内核和磁盘打交道,一方面,该参数应该设置得足够大来应付通常的表访问操作,让更多的数据缓存在 shared_buffers 中。通常设置为实际 RAM 的 10%,比如设置 80 000 个缓冲区(每个缓冲区一般为

8 KB,8 万个缓冲区大小为 625 MB)。将所有的内存都给 shared_buffers 将导致没有内存来运行程序。另一方面,它应该足够小,以避免操作系统因缺少内存而将页面换进换出。

(6) 优化会话级参数

① 优化 KingbaseES 数据库管理系统的 work_mem 参数。

```
SET work_mem = 61440;
```

在执行排序操作时,系统会根据 work_mem 的大小决定是否将一个大的结果集拆分为几个小的和 work_mem 差不多大小的临时文件。显然,拆分的结果会降低排序的速度。因此增加 work_mem 有助于提高排序的速度。通常设置为实际 RAM 的 2%~4%,还要根据需要排序结果集的大小而定,比如 61 440(60 MB)。

② 优化 KingbaseES 数据库管理系统的 temp_buffers 参数。

```
SET temp_buffers = 2048;  /* 设置临时缓冲区为 2 GB */
```

该参数称之为临时缓冲区,用于数据库会话访问临时表数据,系统默认值为 8 MB。可以在单独的 session 中对该参数进行设置,尤其是需要访问比较大的临时表时将会有显著的性能提升。

实验总结:

5. 思考题

试通过一个实际例子,测试不同的 temp_buffers 大小对临时表查询性能的影响。

实验 10　数据库恢复技术实验

数据库恢复技术实验包括三个选修实验项目(参见表 13)。该实验项目为验证性实验项目。

表 13　"数据库恢复技术"实验项目一览表

序号	实验项目名称	开设类别	难易程度	建议实验学时	对应教材章节
实验 10.1	事务实验	选修	适中	2	第 10 章
实验 10.2	数据库备份实验	选修	适中	2	第 10 章
实验 10.3	数据库恢复实验	选修	适中	2	第 10 章

实验 10.1 事务实验

1. 实验目的

掌握数据库事务管理的基本原理以及事务的编程方法。

2. 实验内容和要求

设计几个典型的事务应用,包括显式事务、事务提交、事务回滚、隐式事务等。

3. 实验重点和难点

实验重点:显式事务的编写。

实验难点:把事务的编写和存储过程的设计与使用结合起来。

4. 实验报告示例

实 验 报 告				
题目:事务实验	姓名		日期	

(1) 显式事务的编写

① 创建一个事务,当用户购买零件时,插入订单明细和订单记录,修改供应基本表以保持数据一致性。

```
BEGIN TRANSACTION;          /* 明显写出事务开始和后面的事务提交 */
INSERT INTO Orders(orderkey,custkey,orderstatus,totalprice,   /* 插入订单记录 */
                   orderdate,orderpriority,clerk,shippriority,comment)
VALUES(1021,3,'P',0.0,CURRENT_DATE,'Common','Mike',1,null);
INSERT INTO Lineitem(orderkey,partkey,suppkey,          /* 插入订单明细记录 */
        linenumber,quantity,extendedprice,discount,tax,returnflag,linestatus,
        shipdate,commitdate,receiptdate,shipinstruct,shipmode,comment)
VALUES(1021,479,1,1,2,0.0,0.3,0.08,'F','C',DATEADD('day',3,
        CURRENT_DATE),DATEADD('day',3,CURRENT_DATE),
        CURRENT_DATE,'BOX','GENERAL',NULL);
                    /* 修改订单明细项目对应的零件供应记录中的可用数量 */
UPDATE PartSupp SET availqty=availqty-2
WHERE partkey =479 AND suppkey =1;
COMMIT TRANSACTION;                    /* 明显写出事务开始和事务提交 */
```

② 创建一个事务,当用户撤销某个用户购买记录时,删除订单明细(假设只有一项订单明细)和订单记录,然后修改供应基本表以保持数据一致性。

```
BEGIN TRANSACTION;
   DELETE FROM Lineitem          /* 先删除订单明细记录 */
   WHERE orderkey=1021 AND linenumber=1;
```

```
    DELETE  FROM  Orders              /*再删除订单记录*/
    WHERE  orderkey=1021;
    /*修改订单明细记录对应的零件供应记录中的可用数量*/
    UPDATE PartSupp  SET  availqty=availqty+2
    WHERE partkey=479 AND suppkey=1;
    COMMIT TRANSACTION;
```

③ 创建一个存储过程，当用户撤销某个用户购买记录时，修改供应基本表，删除订单明细（可以是多项）和订单记录，保持数据一致性。

```
    /*该存储过程就是一个事务*/
    /*创建存储过程 PRC_DeleteOrder：删除输入的订单。首先逐个删除订单明细记录，最后删除
  订单记录 */
    CREATE OR REPLACE PROCEDURE PRC_DeleteOrder(P_orderkey INTEGER)
    AS
    DECLARE          /*定义一个带参数的游标，检索给定订单号的订单明细记录*/
      CURSOR Cursor_Lineitem(L_orderkey INTEGER)
          FOR SELECT * FROM Sales.Lineitem WHERE orderkey=L_orderkey;
      res Lineitem%ROWTYPE;              /*定义订单明细记录类型变量 res*/
    BEGIN
      OPEN Cursor_Lineitem(P_orderkey);
    /*打开游标：把存储过程的参数传递给游标的参数*/
    LOOP                        /*针对给定订单的明细记录循环，逐个删除订单明细记录 */
      FETCH Cursor_Lineitem INTO res;          /*从游标获取当前订单明细记录 */
      IF Cursor_Lineitem%NOTFOUND THEN
          EXIT;
      END IF;
      /*修改零件供应表中对应的可用数量 */
      UPDATE Sales.PartSupp SET availqty=availqty+res.quantity
      WHERE partkey=res.partkey AND suppkey=res.suppkey;
      /*删除当前订单明细记录 */
      DELETE FROM  Sales.Lineitem
      WHERE orderkey=P_orderkey AND linenumber=res.linenumber;
    END LOOP;
    CLOSE Cursor_Lineitem;          /* 关闭游标 */
    DELETE FROM  Sales.Orders     /* 删除订单记录 */
    WHERE orderkey=P_orderkey;
```

```
    END;
```

(2) 显式事务的编写(带有回滚)

创建一个事务,当用户购买零件时,插入订单明细(假设只购买一项明细)和订单记录,修改供应基本表,以保持数据一致性。

```
    /* 创建存储过程,完成用户购买零件事务 */
    CREATE OR REPLACE PROCEDURE PRC_Purchase4OneItem(p_orderkey INTEGER,p_custkey
INTEGER,p_partkey INTEGER,p_suppkey INTEGER,p_purchaseqty INTEGER)
    AS
    DECLARE
      L_availqty  INTEGER;
      BEGIN
      BEGIN TRANSACTION
      /* 插入订单记录 */
      INSERT INTO Sales.Orders(orderkey,custkey,orderstatus,totalprice,
              orderdate,orderpriority,clerk,shippriority,comment)
      VALUES(P_orderkey,P_custkey,'P',0.0,CURRENT_DATE,
              'Common','Mike',1,null);
      /*插入订单明细记录 */
      INSERT INTO Sales.Lineitem(orderkey,partkey,suppkey,linenumber,
            quantity,extendedprice,discount,tax,returnflag,linestatus,
            shipdate,commitdate,receiptdate,shipinstruct,shipmode,comment)
      VALUES(P_orderkey,P_partkey,P_suppkey,1,P_purchaseqty,0.0,0.3,0.08,
              'F','C',  DATEADD('day',3,CURRENT_DATE),
              DATEADD('day',3,CURRENT_DATE),
              CURRENT_DATE,'BOX','GENERAL',NULL);

      /*查询订单明细记录对应的零件供应记录中的可用数量*/
      SELECT availqty INTO L_availqty
      FROM Sales.PartSupp
      WHERE partkey=P_partkey AND suppkey=P_suppkey;
      IF (L_availqty > P_purchaseqty)  THEN
        /*如果可用数量大于订购数量,则修改零件供应记录,提交事务*/
        UPDATE Sales.PartSupp  SET  availqty=availqty-P_purchaseqty
        WHERE partkey=P_partkey AND suppkey=P_suppkey;
        COMMIT TRANSACTION;
      ELSE
```

```
    ROLLBACK TRANSACTION;                    /* 可用数量不够,回滚事务 */
    END IF;
END;

CALL PRC_Purchase4OneItem (1021,3,479,1,20);      /* 执行存储过程,购买零件 */

/* 查看 Orders、Lineitem 和 PartSupp 三个表中相应记录,验证存储过程的事务执行正确性 */
SELECT  * FROM Orders WHERE orderkey = 1021;
SELECT * FROM Lineitem WHERE orderkey = 1021;
SELECT * FROM PartSupp WHERE partkey = 479 AND suppkey = 1;

CALL PRC_DeleteOrder(1021);                    /* 执行存储过程,删除指定订单 */
CALL PRC_Purchase4OneItem (1021,3,479,1,40000);/* 重新执行存储过程,购买零件 */

/* 再次查看 Orders、Lineitem 和 PartSupp 三个表中相应记录,验证存储过程的事务执行正确性 */
SELECT  * FROM Orders WHERE orderkey = 1021;
SELECT * FROM Lineitem WHERE orderkey = 1021;
SELECT * FROM PartSupp WHERE partkey = 479 AND suppkey = 1;
```

5. 思考题

试分析存储过程与事务的联系和区别。

实验 10.2　数据库备份实验

1. 实验目的

掌握数据库数据转储备份的方法。

2. 实验内容和要求

了解数据转储备份的概念。学习实际使用的数据库系统(如 Kingbase)中数据库逻辑备份、物理备份 、增量备份和完全备份的概念和使用方法。利用数据库管理系统提供的备份工具实现各种数据库备份策略。

说明:数据库备份也称数据库转储。物理备份是复制数据库物理文件;逻辑备份将数据库对象的定义和数据导出到指定文件中;完全备份是备份整个数据库;增量备份是只备份上次备份以来有变化的数据。

3. 实验重点和难点

实验重点:逻辑备份,物理备份。

实验难点:增量备份。

4. 实验报告示例

实 验 报 告				
题目:数据备份实验	姓名		日期	

实际使用的数据库管理系统所提供的备份功能、备份命令都不尽相同。我们以 Kingbase 为例进行实验。

KingbaseES 提供逻辑备份有三种备份模式:全库备份、模式备份、表备份。

全库备份是指备份单个数据库中所有的用户可备份的对象;模式备份是指备份用户指定的模式和模式所包含的对象;表备份是指备份指定的表和表的数据。

KingbaseES 提供了图形界面的逻辑备份还原工具 javatools.bat JDump,并提供了命令行的逻辑备份工具 sys_dump 和 exp。sys_dump 是 KingbaseES 专有的逻辑备份工具,而 exp 是兼容 Oracle 的逻辑备份工具,exp 的使用依赖于配置的 Kingbase 服务,所以需要在使用前配置 sys_service.conf 文件。

KingbaseES 提供物理全系统备份,指对“全系统”进行备份,备份结果形成一个备份集。备份机制是通过复制 KingbaseES 系统,以统一的格式形成一个全系统的拷贝,以备恢复到一个一致的数据状态。物理全系统备份支持两种类型:物理全系统联机备份和物理全系统脱机备份。KingbaseES 提供了图形界面的物理备份还原工具 javatools.bat JBackup,并提供了命令行的物理备份还原工具 sys_backup。

本实验使用 KingbaseES 命令行工具 sys_dump 和 sys_backup 备份数据库。打开命令窗口,设置 KingbaseES 的可执行文件所在的目录为当前路径,完成如下所有实验。本书假设 KingbaseES 安装后的可执行文件所在目录为 c:\Kingbase\ES\V7\bin。

(1) 逻辑备份整个数据库

① 备份为二进制文件。

```
/*-F c | p:  c 表示备份文件为二进制格式,p 表示备份文件为 SQL 文件,不指明该参数默认为二进制格式;-f 参数指定备份文件名*/
    sys_dump-h localhost-p 54321-U SYSTEM-W MANAGER-F c-f d:\dumpfile.dmp TPCH
```

② 备份为 SQL 文件。

```
    sys_dump-h localhost-p 54321-U SYSTEM-W MANAGER-F p-f d:\dumpfile.sql  TPCH
```

(2) 逻辑备份多个表

备份 TPC-H 的 Orders,Lineitem,PartSupp 等多个表。

```
    /*-t 参数指定备份的表名*/
    sys_dump-h localhost-p 54321-U SYSTEM-W MANAGER-t Sales.Orders  -f d:\dumpfile.dmp
TPCH
```

```
sys_dump-h localhost-p 54321-U SYSTEM-W MANAGER-t Sales.Lineitem  -f d:\dumpfile.dmp
TPCH
sys_dump-h localhost-p 54321-U SYSTEM-W MANAGER-t Sales.PartSupp  -f d:\dumpfile.dmp
TPCH
```

(3) 逻辑备份数据库中指定的模式

备份数据库 TPCH 的 SALES 模式。

```
/*-n 参数指定备份模式名*/
sys_dump-h localhost-p 54321-U SYSTEM-W MANAGER-n Sales  -f d:\dumpfile.dmp TPCH
```

(4) 利用 COPY TO 命令备份数据

备份数据库 TPCH 中 SALES 模式的 Lineitem 表到物理文件 lineitem.csv 中。

```
/*该命令不是 KingbaseES 的命令行工具,需要在 KingbaseES 的查询分析器中执行。FORCE QUOTE
comment:强制复制表中 comment 属性的所有非空数值,两边使用 QUOTE 指定的标记符*/
COPY Sales.Lineitem TO 'd:\lineitem.csv' WITH CSV QUOTE '"' FORCE QUOTE comment;
```

(5) 联机全系统物理备份

设置数据库管理系统为归档模式,然后联机进行全系统的物理备份数据库 TPC-H。

① 首先找到数据库管理系统的配置文件,如 KingbaseES 数据库的配置文件为 Data 目录下的 Kingbase.conf,找到其中 log_archive_start 参数设置为 on,同时设置 log_archive_dest 为一个已经存在的存放归档日志的文件夹名称,然后启动数据库服务器。

② 利用BACKUP 命令联机备份数据库 TPC-H。

```
/*BACKUP 命令不是 KingbaseES 的命令行工具,需要在查询分析器中执行。NAME 参数指定备份
集的名称,TYPE FULL 参数指定为全系统备份,FILEPATH 指定备份数据存放的文件夹,该文件夹可以存
放多个备份集成员。*/
BACKUP NAME week1 TYPE FULL FILEPATH 'd:\Kingbase\archive';
```

(6) 脱机全系统物理备份

停止数据库服务器运行,然后脱机进行全系统的物理备份数据库 TPC-H。

① 利用 KingbaseES 控制管理器停止数据库服务器运行。

② 脱机物理备份数据库 TPC-H。

```
/*-b 参数指定脱机备份,-D 参数指定要备份的数据目录,-n 参数指明备份名,-p 参数指定备份集
路径*/
sys_backup-b-Dc:\Kingbase\ES\V7\data-n offline_backup-P d:\Kingbase\backup
```

(7) 物理联机增量备份

先做一次全系统物理备份作为基准备份,然后修改数据,做增量备份。

① 完全备份,得到完全备份集 week1。

```
/* BACKUP 命令不是 KingbaseES 的命令行工具,需要在查询分析器中执行。 */
BACKUP NAME week1 TYPE FULL FILEPATH 'd:\Kingbase\archive';
```

② 插入数据,修改数据库目录。

```
INSERT INTO Sales.Customer(custkey,name,address,nationkey,phone,acctbal,mktsegment,comment) VALUES(999888,'李四',null,null,null,null,null,null);
```

③ 联机增量备份(其基础备份为 TPC-H_base1),得到增量备份集 diff1。

```
/* FILEPATH 指定增量备份数据存放的文件夹 */
BACKUP NAMEtpch_diff1 FILEPATH 'd:\kingbase\archive\' TYPE DIFFERENTIAL INCREMENT;
```

(8) 物理脱机增量备份

停止数据库服务器运行,然后脱机进行全系统的物理增量备份数据库 TPC-H。

① 利用 KingbaseES 的控制管理器停止数据库服务器。

② 进行脱机完全备份,如果没有停止服务器,就进行脱机完全备份,会报错。

```
/* -b 参数指定脱机备份,-D 参数指定要备份的数据目录,-n 参数指明备份名,-p 参数指定备份集路径 */
sys_backup-b-D c:\Kingbase\ES\V7\data-n offline_backup_full-P d:\Kingbase\backup
```

③ 利用 KingbaseES 控制管理器启动服务器,连接数据库后增加数据

```
CREATE TABLE Sales.Tab(col1 INT,col2 TEXT);
INSERT INTO Sales.Tab VALUES(1,'one');
```

④ 利用 KingbaseES 控制管理器停止数据库服务器。

⑤ 进行脱机差异增量备份,该增量备份的基础备份是第二步的完全备份或差异增量备份。

```
sys_backup-b-n offline_backup_increment2-D c:\Kingbase\ES\V7\data-P d:\Kingbase\backup-t differential
```

实验总结:

5. 思考题

请阅读所用的数据库系统联机帮助,设计一个物理脱机增量备份方案。

实验 10.3　数据库恢复实验

1. 实验目的

掌握数据库逻辑恢复和物理恢复的方法。

2. 实验内容和要求

设计数据库恢复策略，实现数据库恢复，包括数据库逻辑恢复、物理恢复、增量恢复和完全恢复等。数据库恢复也称数据库还原。

3. 实验重点和难点

实验重点：逻辑恢复，物理恢复。

实验难点：增量恢复。

4. 实验报告示例

实 验 报 告				
题目：数据恢复实验	姓名		日期	

"逻辑还原"是指利用备份把数据库从一个快照转化到另一个快照的过程（数据库在某个时刻的一个状态称为一个快照）。

KingbaseES 逻辑还原是将备份文件中的数据库对象和数据还原到指定数据库。逻辑还原时，若不指定还原对象，则对备份文件中的所有备份对象进行还原。逻辑还原方式可以有三种选择：

- 还原整个备份文件。
- 还原指定对象（表、索引、存储过程、触发器）。
- 根据对象列表文件还原。

KingbaseES 提供了图形界面的逻辑备份还原工具 javatools.bat JDump，并提供了命令行工具 sys_restore 和 imp。sys_restore 是 KingbaseES 专有的逻辑备份工具，而 imp 是兼容 Oracle 的逻辑还原工具。imp 的使用依赖于配置的 Kingbase 服务，所以需要在使用前配置 sys_service.conf 文件。

"物理还原"指的是通过备份集和归档日志将数据库转化为一致性状态的过程。还原时可以采用两种策略：

- 只使用增量备份的备份集进行恢复。
- 使用增量备份的备份集+归档日志+尾日志进行恢复。增量备份的脱机恢复对用户来说是透明的。

系统会自动判断是完全备份还是增量备份。如果是增量备份，系统会依次恢复本备份

集到基础备份间的所有备份集。增量恢复成功后,可以是一个不完整恢复,此时将会为恢复后数据目录中的日志文件生成新的时间线,这点和完全备份是一致的。

KingbaseES 提供了图形界面的物理备份还原工具 javatools.bat JBackup,并提供了命令行工具 sys_backup-r,该命令可以使用联机和脱机中的完全备份和增量备份进行脱机恢复。

本实验使用 KingbaseES 命令行工具 sys_restore 和 sys_backup-r 还原数据库。打开命令窗口,设置 KingbaseES 的可执行文件的所在目录为当前路径,完成如下所有实验。本书假设 KingbaseES 安装后的可执行文件所在目录为 c:\Kingbase\ES\V7\bin。

(1) 逻辑还原整个数据库

从二进制备份文件还原整个数据库。

```
/*-d 参数指明数据库名,-clean 参数指明还原之前删除已经存在的数据库对象*/
sys_restore-h localhost-p 54321-U SYSTEM-W MANAGER-clean-d TPCH d:\dumpfile.dmp
```

(2) 逻辑恢复多个表

恢复数据库 TPC-H 的 Orders,Lineitem,PartSupp 等多个表。

```
/*-t 参数指定要恢复的表名*/
sys_restore-h localhost-p 54321-U SYSTEM-W MANAGER-clean-d TPCH-t Sales.Orders d:\
dumpfile.dmp
sys_restore-h localhost-p 54321-U SYSTEM-W MANAGER-clean-d TPCH-t Sales.Lineitem d:\
dumpfile.dmp
sys_restore-h localhost-p 54321-U SYSTEM-W MANAGER-clean-d TPCH-t Sales.PartSupp d:\
dumpfile.dmp
```

(3) 逻辑恢复指定模式

恢复 TPCH 数据库的 SALES 模式。

```
/*-d 参数指明数据库名,实质与恢复整个数据库的命令一致*/
sys_restore-h localhost-p 54321-U SYSTEM-W MANAGER-clean-d TPCH d:\dumpfile.dmp
```

(4) 利用 COPY FROM 命令恢复数据

从物理文件 lineitem.csv 恢复 TPCH 数据库 SALES 模式中的 Lineitem 表。

```
/*该命令需要在 KingbaseES 的查询分析器中执行。WITH CSV 指明输入数据文件格式;
QUOTE:指明 CSV 文件中所有非空数值两边使用的标记符,默认为双引号。*/
COPY Sales.Lineitem FROM 'd:\lineitem.csv' WITH CSV QUOTE '"';
```

说明:执行 COPY FROM 命令时,DBMS 进行数据完整性检查。如果要恢复的表已存有数据,可能由于数据重复引起完整性检查失败,从而导致 COPY FROM 命令执行失败。解决的办法之一是,如确认表中已有数据已失效,可以先执行 TRUNCATE Sales.Lineitem 命令清空该表中的数据,然后执行上述 COPY FROM 命令。

(5) 全系统恢复物理数据库

① 利用联机和脱机中的完全备份进行全系统恢复物理数据库 TPCH。

使用命令行工具 sys_backup,全系统恢复物理数据库 TPCH 到新路径。

```
/* -r 指定使用物理全系统恢复功能;-P 指明完全备份集所在的路径,该路径包含备份集的名称;-N: 指定恢复数据库数据到指定的目的路径。 */
sys_backup-r-P d:\Kingbase\archive\week1-N c:\Kingbase\data
```

说明:如果目的路径是备份前数据所在路径,则要删除该路径之后再恢复,否则由于该路径下有数据而非空导致恢复失败;如果目的路径是一个新路径,则要修改 KingbaseES 安装路径中 config 目录下 instance.conf 文件中的 data_dir 和 log_dir 为该新路径,以使 KingbaseES 启动时从该新路径中装载数据库数据。

② 利用联机和脱机中的增量备份进行全系统恢复物理数据库 TPCH。

使用命令行工具 sys_backup,全系统恢复物理数据库 TPCH 到新路径。

```
/* -r 指定使用物理全系统恢复功能;-P 指明增量备份集所在的路径,该路径包含备份集的名称;-N: 指定恢复数据库数据到指定的目的路径。 */
sys_backup-r-P d:\Kingbase\archive\tpch_diff2-N c:\Kingbase\data1
```

说明:该命令实际上与利用完全备份集进行恢复是一样的,只是-P 参数指明的是最后一次增量备份集所在的路径。系统先自动找到增量备份对应的最后一次完全备份做全系统物理数据库恢复,然后按照增量备份集的时间顺序依次恢复各个增量备份集。

③ 使用归档的日志和尾日志文件,恢复备份名为 ONLINE_B2 的备份到新位置

```
/* -P 指明备份集所在的路径,该路径包含备份集的名称;-N: 指定恢复数据库数据到指定的目的路径;-A: 指明归档日志保存的位置;-D: 恢复时,是尾日志所在的数据目录的路径值。尾日志通常在 KingbaseES 系统运行的数据目录中,所以在基于归档日志的恢复中,需要指定这个目录 */
sys_backup-r-P d:\Kingbase\archive\week2-N c:\Kingbase\recover\data-A d:\Kingbase\archive
-D c:\Kingbase\ES\V7\data
```

实验总结:

5. 思考题

请阅读所用的数据库系统联机帮助,设计一个物理脱机增量恢复方案。

实验 11 并发控制实验

并发控制为 1 个实验项目(参见表 14),为选修的设计性实验项目。

表 14 "并发控制"实验项目一览表

实验项目名称	开设类别	难易程度	建议实验学时	对应教材章节
并发控制实验	选修	适中	2	第 11 章

1. 实验目的

掌握数据库并发控制的基本原理及其应用方法。

2. 实验内容和要求

验证并发操作带来的数据不一致性问题,包括丢失修改、不可重复读和读"脏"数据等情况。要求通过取消查询分析器的自动提交功能,创建两个不同的用户,分别登录查询分析器,同时打开两个客户端;通过 SQL 语言设计具体例子展示不同的封锁级别的应用场景,验证各种封锁级别的并发控制效果,以进一步理解封锁技术是如何解决事务并发导致的问题。

3. 实验重点和难点

实验重点:并发操作带来的数据不一致性问题。

实验难点:设计具体的例子演示各种封锁级别。

4. 实验报告示例

实 验 报 告				
题目:并发控制实验	姓名		日期	

DBMS 通常提供四级隔离级别:读未提交(read uncommitted)、读已提交(read committed)、可重复读(repeatable read)、可串行化(serializable)。

四级隔离级别与三级封锁协议大致对应关系如下:

- 读未提交隔离级别提供最大的事务并发度,但仅能避免丢失修改,对应一级封锁协议。
- 读已提交隔离级别提供的事务并发度减弱,能够避免丢失修改和脏读,对应二级封锁协议。
- 可重复读隔离级别提供的事务并发度进一步减弱,能够避免丢失修改、脏读和不可重复读,对应增强的二级封锁协议。

• 可串行化隔离级别提供最小的事务并发度，能够避免所有的事务并发控制问题，对应三级封锁协议。

read commited 是 KingbaseES 默认的事务隔离级别。运行在该隔离级别的事务中，查询语句只能看到该查询开始执行之前提交的数据，而不会看到任何未提交的数据或查询执行期间并发的其他事务提交的数据，但是该查询可以看到本事务中查询之前执行的数据更新。

serializable 提供了更加严格的事务隔离级别。在该事务隔离级别下，事务好像没有并发，是串行执行的。运行在 serializable 隔离级别的事务中，查询语句只能看到该事务开始执行之前提交的数据，而不会看到任何未提交的数据或事务执行期间并发的其他事务提交的数据。

repeatable read 向上兼容 serializable。read uncommitted 向上兼容 read commited。

(1) 实验准备

① 数据准备：给 Sales 模式下的 PartSupp 表增加一条记录。

```
/*插入零件供应记录 */
INSERT INTO Sales.PartSupp(partkey,suppkey,availqty,supplycost,comment)
VALUES(1,1,0,0.0,null);
/*设置指定零件供应记录的可用数量为 0 */
UPDATE Sales.PartSupp
SET availqty=0
WHERE partkey=1 AND suppkey =1;
```

② 创建两个超级用户：SYSTEM1 和 SYSTEM2。

```
CREATE USER SYSTEM1 WITH SUPERUSER PASSWORD 'MANAGER';
CREATE USER SYSTEM2 WITH SUPERUSER PASSWORD 'MANAGER';
```

③ 分别以 SYSTEM1 和 SYSTEM2 两个用户登录查询分析器。

假设 SYSTEM1 用户登录打开的查询分析称为客户端 A，SYSTEM2 用户登录打开的查询分析器称为客户端 B。设置客户端 A 和客户端 B 的当前数据库为 TPCH。取消两个查询分析器的自动提交事务的功能，改为显式事务提交，即需要执行 COMMIT 命令提交事务。

```
SHOW TRANSACTION ISOLATION LEVEL;
```

(2) 在“读已提交”隔离级别下，验证丢失修改的情况。

① 查看 Sales 模式下 PartSupp 表的 availqty 值为 0。

分别在客户端 A 和客户端 B 执行如下 SQL 语句。

```
/*查看零件供应记录 */
SELECT availqty FROM Sales.PartSupp WHERE partkey =1 AND suppkey=1;
```

② 在客户端 A 执行如下 SQL 语句修改 PartSupp 表的 availqty 值。

```
UPDATE Sales.PartSupp SET availqty =availqty-5
WHERE partkey=1 AND suppkey=1;
/*查看 PartSupp 表的 availqty 值为修改后的值*/
SELECT availqty FROM Sales.PartSupp WHERE partkey =1 AND suppkey=1;
```

③ 在客户端 B 执行如下 SQL 语句也修改 PartSupp 表的 availqty 值。

```
UPDATE Sales.PartSupp SET availqty =availqty-5
WHERE partkey=1 AND suppkey=1;
/*查看 PartSupp 表的 availqty 值为修改后的值*/
SELECT availqty FROM Sales.PartSupp WHERE partkey =1 AND suppkey=1;
```

④ 在客户端 A 执行如下 SQL 语句提交事务：

```
COMMIT;
```

⑤ 在客户端 B 执行如下 SQL 语句提交事务：

```
COMMIT;
```

⑥ 在客户端 A 和 B 分别执行如下 SQL 语句，查看 PartSupp 表的 availqty 值。

```
/*看到修改后且已提交的 availqty 值*/
SELECT * FROM Sales.PartSupp WHERE partkey =1 AND suppkey=1;
```

上述操作过程可以用下面的表格示意表达：

T_A 终端 A 提交的事务	T_B 终端 B 提交的事务
Read(availqty)=22	
	Read(availqty)=22
Update(availqty)=availqty-5	
Read(availqty)=17	
	Update(availqty) = availqty-5
	等待...
Commit	
	Update(availqty) = availqty-5
	执行完成
	Read(availqty)=12
	Commit;

(3) 在“读已提交”隔离级别下，验证避免脏读的情况

① 查看 Sales 模式下 PartSupp 表的 availqty 值为 0。

分别在客户端 A 和客户端 B 执行如下 SQL 语句。

```
/* 查看零件供应记录 */
SELECT availqty FROM Sales.PartSupp WHERE partkey = 1 AND suppkey = 1;
```

② 在客户端 A 执行如下 SQL 语句,修改 PartSupp 表的 availqty 值。

```
UPDATE Sales.PartSupp SET availqty = 22 WHERE partkey = 1 AND suppkey = 1;
/* 查看 PartSupp 表的 availqty 值为修改后的值 */
SELECT availqty FROM Sales.PartSupp WHERE partkey = 1 AND suppkey = 1;
```

③ 在客户端 B 执行如下 SQL 语句,查看 PartSupp 表的 availqty 值。

```
/* 看不到 A 客户端修改后但还未提交的 availqty 值 */
SELECT availqty FROM Sales.PartSupp WHERE partkey = 1 AND suppkey = 1;
```

④ 在客户端 A 执行如下 SQL 语句提交事务:

```
COMMIT;
```

⑤ 在客户端 B 执行如下 SQL 语句,查看 PartSupp 表的 availqty 值:

```
/* 看到 A 客户端修改后且已提交的 availqty 值 */
SELECT * FROM Sales.PartSupp WHERE partkey = 1 AND suppkey = 1;
```

上述操作过程可以用下面的表格示意表达:

T_A 终端 A 提交的事务	T_B 终端 B 提交的事务
Read(availqty) = 0	
	Read(availqty) = 0
update(availqty) = 22	
Read(availqty) = 22	
	Read(availqty) = 0
Commit	
	Read(availqty) = 22

(4) 在“读已提交”隔离级别下,验证出现不可重复读的现象

① 客户端 A 执行如下 SQL 语句,读取指定零件的 availqty 值。

```
SELECT availqty FROM Sales.PartSupp WHERE partkey = 1 AND suppkey = 1;
```

② 客户端 B 执行 SQL 语句,修改指定零件的 availqty 值。

```
UPDATE Sales.PartSupp SET availqty = 33 WHERE partkey = 1 AND suppkey = 1;
```

③ 客户端 A 再次读取指定零件的 availqty 值。

```
/* 发现这次读取的值与上次读取的值一样,因为客户端 B 还没有提交 */
SELECT availqty FROM Sales.PartSupp WHERE partkey = 1 AND suppkey = 1;
```

④ 客户端 B 提交事务。

```
COMMIT;
```

⑤ 客户端 A 再次读取指定零件的 availqty 值。

```
/* 发现这次读取的值与上次读取的值不一样了,发生了不可重复现象 */
SELECT availqty FROM Sales.PartSupp WHERE partkey = 1 AND suppkey = 1;
```

⑥ 客户端 A 提交事务。

```
COMMIT;
```

(5) 在“读已提交”隔离级别下,验证出现“幻影”的现象

① 客户端 A 执行如下 SQL 语句,读取指定供应商的零件供应记录。

```
SELECT * FROM Sales.PartSupp WHERE suppkey = 1;
```

② 客户端 B 执行 SQL 语句,插入指定供应商新的零件供应记录。

```
INSERT INTO Sales.PartSupp(partkey, suppkey, availqty, supplycost, comment) VALUES(2, 1, 0,
0.0, null);
```

③ 客户端 A 再次读取指定供应商的零件供应记录。

```
/* 发现这次读取的值与上次读取的值一样,因为客户端 B 还没有提交 */
SELECT * FROM Sales.PartSupp WHERE suppkey = 1;
```

④ 客户端 B 提交事务。

```
COMMIT;
```

⑤ 客户端 A 再次读取指定供应商的零件供应记录。

```
/* 发现这次读取的值与上次读取的值不一样了,发生了“幻影”现象 */
SELECT * FROM Sales.PartSupp WHERE suppkey = 1;
```

⑥ 客户端 A 提交事务。

```
COMMIT;
```

(6) 在“可串行化”隔离级别下,验证出现“幻影”的现象

修改 KingbaseES 的 data 文件下的 kingbase.conf 文件,设置 default_transaction_isolation 为 'serializable',重新启动数据库,然后重新执行本实验中(2)、(3)和(4)等三项任务。在“可串行化”隔离级别下,避免脏读、避免不可重复读和避免“幻影”。

实验总结:

不可重复读分为三种情况,一是由于修改导致不可重复读;二是由于插入新记录导致不可重复读,第二次读取时会多出一些新记录;三是由于删除记录导致不可重复读,即第二次读取时有些旧记录“消失”了。后面两种情况称为“幻影”现象。

5. 思考题

试分析在“读已提交”隔离级别下,设计一个应用程序以避免“不可重复读”和“幻影”。

五 实验考核标准和评价方式

1. 实验考核标准

数据库课程实验考核内容主要分为:实验过程、实验代码和实验报告三部分(实验考核评分标准表参见表 15)。实验过程考核主要考查学生实验课程出勤、实验态度、实验任务完成情况、实验过程中分析问题和解决问题的能力等方面;实验代码考核主要是考查学生编写的 SQL 代码规范性、正确性和代码质量等方面;实验报告考核主要是考查实验报告内容完整性、正确性和规范性等方面。

数据库课程实验通常由教师选择实验手册中的若干个小实验组成,实验成绩通常占课程总成绩的 30%~50%,实验过程、实验代码和实验报告可分别占三分之一的分数。考核标准以实验成绩 100 分计算,最终实验成绩可以根据其占课程总成绩的百分比折算。

表 15 实验考核评分标准表

考核内容	评分项	评分标准	分数	备注
1. 实验过程	1.1 实验课程出勤	全勤 缺勤 2 次以下 缺勤 3 次以上	10 8 6	
	1.2 实验态度	积极认真 一般 态度不端正	10 7 5	
	1.3 实验任务完成情况	全部完成 缺项 1 次 缺项 2 次以上	10 8 6	
	1.4 分析问题和解决问题的能力	良好 一般 较差	10 7 6	

续表

考核内容	评分项	评分标准	分数	备注
2. 实验代码	2.1 代码规范性	符合 SQL 编程规范 基本规范 不够规范	10 8 6	
	2.2 代码正确性	运行结果全部正确 错误 2 项以下 错误 3 项以上	10 7 5	
	2.3 代码质量	质量较高 一般 质量较差	10 7 5	
3. 实验报告	3.1 内容完整性	内容完整不缺项 缺 1 项 缺 2 项以上	10 8 5	
	3.2 内容正确性	内容全部正确 错误 1 项 错误 2 项以上	10 8 6	
	3.3 规范性	内容符合实验报告撰写规范 基本规范,字体格式等不统一 不规范	10 8 5	

表 15 中实验考核标准可以针对每个小实验评定分数,然后加权求和得到总体实验成绩;也可以针对所有实验总体完成情况一次性评定分数。

2. 实验评价方式

实验评价从不同角度来看有多种方式。例如,

① **教师评价**:教师是传统的最主要的评价主体。实验课教师全程跟踪和监督每个学生的实验情况,依据实验考核标准,作出详细的评价。

② **学生互相评价**:学生互相之间是很了解各自试验情况的。通过学生之间互相匿名评价,引入监督机制,可以更好地促进实验成绩公平公正的评价。

③ **学生演示汇报和答辩**:在大作业实验结束阶段,每个小组进行 60 分钟的报告和答辩,

制作幻灯片，讲解设计方案，演示系统运行，汇报分工与合作情况。回答老师和同学们的提问。根据整个小组实验情况进行评价，给出总体实验成绩，然后根据小组中每个成员的完成情况，评价每个小组成员的实验成绩。

上述各种评价方式可以按需组合，形成各具特色的实验评价方式。

六 SQL 语言实验常见问题解答

SQL 语言是数据库实验的核心内容。SQL 语言实验的常见错误分为两种:语法错误和逻辑错误。对于语法错误,DBMS 会给出相应的错误信息,有些错误信息比较直观,容易理解,而有些错误信息并不直观,有时还比较费解。对于逻辑错误,SQL 语句本身是可以执行并得出结果的,只不过这样的结果并不是用户所期望的,是不正确的结果。因此,这里列出 SQL 语言有关实验过程中常见的错误及其解答,以便帮助初学者更好地理解数据库知识,更快地提高数据库实验水平和实践开发能力。

(1) 单引号和双引号导致的错误。例如执行:

```
SELECT *
FROM Sales.Customer
WHERE name="李四";
```

会出现错误信息:

ERROR:列"李四"不存在。

答:英文双引号括起来的字符串表示列名,英文单引号括起来的字符串表示字符串常量。上述 SQL 中"李四"改为'李四'即可。

(2) 中文单引号或双引号字符导致的错误。例如执行:

```
SELECT *
FROM Sales.Customer
WHERE name='李四';
```

会出现错误信息:

ERROR:列 "'李四'" 不存在。

答:把中文单引号改为英文的单引号即可。SQL 查询分析器一般都是语法制导编辑的,字符串常量一般显示为红色,如果没有变色,即为非字符常量。还有中文的分号、中文的单括号、中文的加号和减号经常会导致语法错误。因此,应该学会快速辨别中文标点符号和英文标点符号显示时的区别。

(3) 当前模式不清楚导致的错误。例如执行:

```
SELECT *
FROM Customer
WHERE name='李四';
```

会出现错误信息:

ERROR:关系"CUSTOMER"不存在。

答:该问题可能导致的原因有:

① 关系 Customer 确实不存在。

② 当前数据库设置错误,需要正确设置当前数据库。

③ 当前模式设置错误;更正方法:需要用

```
SET SEARCH_PATH TO Sales,Public;
```

设置当前搜索模式为 Sales 模式。或者直接在 SELECT 语句中,在 Customer 前加上关系所属的模式名,如 Sales.Customer。

(4) 与 CHAR 固定长度字符类型字段自动补齐空格有关的错误。例如执行:

```
SELECT CONCAT(NAME,'AA')
FROM Sales.Nation
WHERE name LIKE '中国';
```

会查找不到记录。

答:因为 Nation 的 name 字段是 CHAR(25)固定长度字符类型,不足 25 个字符的国名,系统自动在国名的后面补空格。因此,WHERE 条件换成 TRIM(name) LIKE '中国'就可以找到一条记录,而换成 name LIKE '中国%'就可以找到记录。

(5) 多个表中相同列名引起歧义而导致的错误。例如执行:

```
SELECT name
FROM Sales.Supplier S,Sales.partsupp PS,Sales.Part P
WHERE S.suppkey=PS.suppkey AND PS.partkey=P.partkey;
```

会出现错误信息:

ERROR:列引用 "NAME" 存在歧义。

答:因为 Supplier 和 Part 两个表中都存在 name 字段,因此在 SELECT 子句中应该指明输出哪个表中的 name 字段值。修改为:SELECT S.name 即可。

(6) GROUP BY 子句中未出现的列名不能输出导致的错误。例如执行:

```
SELECT C.name,SUM(totalprice)
FROM Sales.Orders O,Sales.Customer C
WHERE O.custkey=C.custkey
GROUP BY O.custkey;
```

会出现错误信息:

ERROR:列 "C.NAME" 必须出现在 GROUP BY 子句或者聚集函数中。

答:带有 GROUP BY 的 SQL 语句中,只有出现在 GROUP BY 中的字段,以及聚集函数才能在 SELECT 子句中输出。因此,上述 SQL 语句可以改为

```
SELECTMAX(C.name),SUM(totalprice)
FROM Sales.Orders O,Sales.Customer C
WHERE O.custkey=C.custkey
GROUP BY O.custkey;
```

或者:

```
SELECTC.name,SUM(totalprice)
FROM Sales.Orders O,Sales.Customer C
WHERE O.custkey=C.custkey
GROUP BY O.custkey,C.name;
```

(7) 集合和标量做相等比较导致的错误。例如,查找张姓顾客的订单信息:

```
SELECT *
FROM Orders
WHERE custkey=(SELECT custkey
               FROM Sales.Customer
               WHERE name like '张%');
```

会出现错误信息:

ERROR:作为一个表达式使用的子查询返回多行结果。

答:上述 SQL 子查询返回多行记录的集合,不能直接和 custkey 标量值进行相等比较,把相等比较运算法=改为 IN 集合运算符即可。

(8) 同时删除多个表数据的错误。例如,删除 3 号供应商及其零件供应记录:

```
DELETE FROM Sales.Supplier,Sales.PartSupp
WHERE suppkey=3;
```

会出现错误信息:

ERROR:语法错误 在 "," 附近。

答:DELETE 语句一次只能删除一个表的数据。上述 SQL 语句修改为两条 DELETE 语句,并且要先删除依赖表 Sales.PartSupp 中的数据,后删除被依赖表 Sales.Supplier 中的数据。INSERT 和 UPDATE 语句也是一次只能插入一个表或者更新一个表的数据。

(9) WHERE 条件逻辑错误。例如查找中国供应商的信息:

```
SELECT *
FROM Sales.Supplier
WHERE(SELECT nationkey
FROM Sales.Nation
WHERE TRIM(name)='中国');
```

该 SQL 语句会查找出所有记录,与查询要求不符。

答:应该修改 WHERE 子句为 nationkey IN(SELECT...)。

(10) 关于别名引用的错误。例如查找中国供应商的名称:

```
SELECT COUNT( * )
FROMSales.Supplier S,Sales.Nation N
WHERESales.Supplier.nationkey=N.nationkey AND N.name='中国';
```

该 SQL 语句执行出现错误信息:

ERROR:WHERE 子句中对表"SUPPLIER"的无效引用。

HINT:可能你原本意图引用表的别名 "S"。

答:一旦在 FROM 子句中给某个表定义了别名,则在该 SQL 语句中各个引用该表名的地方,都应该以其别名代替。上述 SQL 语句的修改方法,或者是把 WHERE 子句中的 Sales.Supplier.nationkey 改为 S.nationkey,或者是把 FROM 子句中 Sales.Supplier 的别名 S 去掉。

(11) 修改数据类型的错误。例如,把 Part 零件表中的零件型号属性(type)改为整数类型:

```
ALTER TABLE Sales.Part ALTER COLUMN type TYPE INTEGER;
```

该 SQL 语句执行出错:

ERROR:无效的整型输入:"HP 862XL …"。

答:必须是 Part 表 type 属性中所有已存在的值都能合法自动地转换成整数,该类型数据才能转换成整数,如其中有一条记录的 type 属性值不能自动转换成整数,整个 SQL 语句也不会执行成功。处理方法之一,该属性 type 不能修改其数据类型为整数类型,放弃修改;处理方法之二,如确实需要修改成整数类型,可以先自动或者手工处理属性 type 所有的值成可自动转换成整数类型的数值,然后执行上述属性数据类型修改 SQL 语句。

(12) 除数为零导致出错的问题。例如随机返回指定编号供应商的一条供应记录:

```
/ * 创建函数:随机获取指定供应商的一条供应记录 * /
CREATE OR REPLACEFUNCTION Sales.RandomPartSupp(p_suppkey INTEGER)
RETURN SETOF Sales.PartSupp AS
DECLARE
  L_partsuppCount INT;
  res Sales.PartSupp%ROWTYPE;
BEGIN
  SELECT COUNT( * )INTO L_partsuppCount  / * 计算 PartSupp 表中指定供应商编号的记录个数 * /
  FROM Sales.PartSupp
  WHERE suppkey=p_suppkey;      / * 输入参数 p_suppkey:指定的供应商编号 * /

  SELECT partkey,suppkey,availqty,supplycost,comment INTO res  / * 随机定位一条记录 * /
  FROM Sales.PartSupp
```

```
    WHERE suppkey=p_suppkey      /* 输入参数 p_suppkey:指定的供应商编号 */
    LIMIT 1
    OFFSET MOD(CAST(RANDOM() * 1000 AS INTEGER),L_partsuppCount);
    RETURN NEXT res;                /* 返回随机选定的记录 */

    RETURN;
END;
CALL Sales.RandomPartSupp(7667);   /* 调用函数:随机返回编号为 7667 供应商的一条零件供
                                      应记录 */
```

该调用函数的 SQL 语句执行出错：

ERROR:除以 0。

答:因为编号为 7667 的供应商没有零件供应记录,则 L_partsuppCount 变量值为 0,所以求余函数 MOD()的除数为零,导致出错。如任意选取一个有零件供应记录的供应商编号,例如 2077,再执行上述函数就不会出错：

CALL Sales.RandomPartSupp(2077)。

(13) 改变已有自定义函数返回类型导致的出错。例如指定编号供应商的零件平均零售价格：

```
/* 创建函数:求指定编号供应商所供应零件的平均零售价格 */
CREATE OR REPLACE FUNCTION Sales.AvgRetailPrice4Supplier(p_suppkey INTEGER)
RETURN  INTEGER  AS
DECLARE
  L_avgretailprice DOUBLE;
BEGIN
  SELECT AVG(P.retailprice) INTO L_avgretailprice
  FROM Sales.Part P,Sales.PartSupp PS,Sales.Supplier S
  WHERE P.partkey=PS.partkey ANDPS.suppkey=S.suppkey AND S.suppkey=p_suppkey;
  RETURN L_avgretailprice;
END;
 /* 修改上述函数返回值的类型,重新创建函数,出错 */
CREATE OR REPLACEFUNCTION Sales.AvgRetailPrice4Supplier(p_suppkey INTEGER)
RETURN DOUBLE AS  ...(此处省略代码,与上述函数定义相同)

/* 调用函数,测试函数功能 */
CALL Sales.AvgRetailPrice4Supplier(2077);
```

该调用函数的 SQL 语句执行出错：

ERROR:不能改变已有函数的返回类型。

HINT:首先使用 DROP FUNCTION。

答:CREATE OR REPLACE FUNCTION 语句创建用户自定函数时,不能改变已有函数的返回类型。如果有必要,先要用 DROP FUNCTION 删除已有函数,然后重新创建同名函数。

(14) 插入或更新数据超过属性取值范围的错误。例如插入一条新的零件记录:

```
INSERT INTO Sales.Part(partkey,name,mfgr,brand,type,size,container,retailprice,comment)
VALUES(123456789,'轴承','中国辽宁省大连市瓦房店新世纪特殊轴承专造新技术有限责任公司
    ',null,null,null,null,1990.00,null);
```

该 SQL 语句执行出错:

ERROR:用于类型字符(25)的值过长。

答:mfgr 属性数据类型为 CHAR(25),上述制造商名称有 29 个字符,超过定义长度。处理办法之一是修改 Part 表 mfgr 属性数据类型,增加字符类型的长度,如

```
ALTER TABLE Sales.Part ALTER COLUMN mfgr TYPE CHAR(40);
```

然后执行上述插入语句。处理办法之二是简化上述制造商的名称,如改为"中国瓦房店新世纪特殊轴承专造新技术有限责任公司"。类似的错误还有"integer 越界"等。

(15) 两个字符串串接运算引起的错误。例如以指定格式的字符串输出零件名称和制造商信息:

```
SELECT name+'('+mfgr+')'
FROM Sales.PART
WHERE mfgr LIKE '北京%';
```

该 SQL 语句执行出错:

ERROR:操作符不唯一:CHARACTER VARYING+" UNKNOWN"。

HINT:无法确定一个最匹配的操作符。你可能需要使用显式类型转换。

答:KingbaseES 的字符串串接运算不能使用"+"运算符,需要使用 CONCAT 字符串串接函数。正确的 SQL 语句如下:

```
SELECT CONCAT(CONCAT(CONCAT(name,'('),TRIM(mfgr)),')')
FROM Sales.Part WHERE mfgr LIKE '北京%'。
```

(16) 插入数据违反外码约束导致的错误。例如插入一条零件供应记录:

```
INSERT INTO Sales.PartSupp(partkey,suppkey,availqty,supplycost,comment)
VALUES(123456,123456,1000,1990.00,null);
```

该 SQL 语句执行出错:

ERROR:向表" PARTSUPP" 中插入和更新违反了外码约束" PARTSUPP_PARTKEY_FKEY"。

答:由于创建 PartSupp 表时,定义了约束名为"PARTSUPP_PARTKEY_FKEY"的外码,规定 PartSupp 表中 partkey 须引用 Part 表中已存在记录的 partkey 值。在 Part 表中不存在编号

为 123456 的零件记录，因此上述插入语句违反了外码约束，即违反了参照完整性，导致 SQL 语句执行错误。

（17）删除数据违反外码约束导致的错误。例如删除一条零件记录：

```
DELETE FROMSales.Part WHERE partkey = 46777;
```

该 SQL 语句执行出错：

ERROR：在表"PART"上的更新或删除操作破坏了外码约束"PARTSUPP_PARTKEY_FKEY"在表"PARTSUPP"。

答：因为在 PartSupp 表中存在编号为 46777 的零件供应记录，删除 Part 表中该编号零件记录，将破坏 PartSupp 表中的外码约束，因此不能删除该记录。如果创建 PartSupp 表中的外码约束 PARTSUPP_PARTKEY_FKEY 时定义了 ON DELTE CASCADE，则上述 DELETE 语句是可以执行成功的。

（18）违反 CHECK 完整性约束导致的错误。例如插入一条顾客记录：

```
ALTER TABLE Sales.Customer ADD CONSTRAINT CK_Customer_m ktsegment
CHECK( mktsegment IN('亚太区','北美区','欧洲区','非洲区'));
INSERT INTO Sales.Customer( custkey,name,mktsegment)
VALUES( 1234567,' Mike','南美区');
```

该插入 SQL 语句执行出错：

ERROR：关系"CUSTOMER"的新增行违反了 CHECK 约束"CK_CUSTOMER_MKTSEGMENT"。

答：因为在 CHECK 短语制定的列值中没有南美区。

（19）违反实体完整性约束导致的错误。例如插入一条顾客记录：

```
INSERT INTO Sales.Customer( custkey,name,mktsegment)
VALUES( 1,' Mike','北美区');
```

该插入 SQL 语句执行出错：

ERROR：重复的码违背了主码约束 "CUSTOMER_PKEY"。

答：因为顾客编号 custkey 为主码，已有一条编号为 1 的顾客记录，不能再插入一条相同编号的顾客记录。否则违反了实体完整性约束。

（20）已有重复记录，不能设置主码的问题。例如，给表建立实体完整性约束：

```
CREATE TABLE Sales.TestPk( a INT,b INT);
INSERT INTO Sales.TestPK VALUES(1,1);
INSERT INTO Sales.TestPK VALUES(1,2);
ALTER TABLE Sales.TestPK ADD CONSTRAINT PK_TestPK_a PRIMARY KEY(a);
```

该 SQL 语句执行出错：

ERROR：不可以建立 UNIQUE 索引。

答：因为在 a 属性上建立实体完整性约束，就要在 a 属性上建立 UNIQUE 索引。但是，

TestPK 表中已有两条记录在 a 属性有重复值 1,则不能建立 UNIQUE 索引,因此也不能在 a 属性建立实体码完整性约束。处理办法之一:删除在 a 属性上具有重复值的记录;处理办法之二:修改具有重复值的 a 属性值,使之在 a 属性上不再具有重复值。当表 TestPK 的记录数比较少的时候,上述两种办法都可以手工完成;但是当表 TestPK 中具有成千上万条记录时,手工方法都是难以完成上述两种处理方法的,需要采用一定的算法编写存储过程自动删除或者修改重复记录才能实现。读者可以考虑编写相应的存储过程实现删除一个表中的重复记录。

(21) 已有数据,修改属性完整性约束不能满足的错误。如修改 Part 表中的 type 属性为非空:

```
ALTER TABLE Sales.Part ALTER COLUMN type SET NOT NULL;
```

该 SQL 语句执行出错:

ERROR:列"TYPE"包括空值。

答:因为 type 属性上已有空值的记录,因此修改 type 属性为非空不成功。处理办法就是把 type 属性上的 NULL 值全部改写为某个特殊的非空值,或者某个缺省的非空值(例如 UPDATE Sales.Part SET type ='UNKNOWN' WHERE type IS NULL;),然后再执行上述 ALTER TABLE 语句即可。

(22) 有依赖对象(如视图)的基本表,不能更改其属性的数据类型,也不能删除属性。

```
CREATE VIEW Sales.V_PartType AS
SELECT partkey,name AS partname,brand AS partbrand,type AS parttype
FROM Sales.Part;                    /* Sales.Part 上建立了视图 */
ALTER TABLE Sales.Part ALTER COLUMN brand INTEGER;
```

该 SQL 语句执行出错:

ERROR:不能使用视图或者规则修改一个列的类型。

```
ALTER TABLE Sales.Part DROP COLUMN brand;
```

该 SQL 语句执行出错:

视图 Sales.V_PARTTYPE 依赖于表 SALES.PART 上的列 BRAND。

答:因为在 Part 表 brand 等属性上定义了视图 V_PartType,所以系统不允许修改这些列的数据类型,或者删除这些列;但可以重命名这些列名(如 ALTER TABLE Sales.Part RENAME COLUMN brand TO partbrand;)。当使用 CASCADE 参数来删除被依赖的属性,如 ALTER TABLE Sales.Part DROP COLUMN brand CASCADE;则连同依赖对象(如视图 Sales.V_PartType)也一起删除。

(23) 调用系统函数、存储过程或者自定义函数缺少参数的错误。

```
SELECT SUBSTRING('this is a test');
```

该 SQL 语句执行出错:

ERROR:函数 SYS_CATALOG.SUBSTRING("UNKNOWN")不存在。

HINT:不能根据所提供的名称和参数类型找到与之匹配的函数。你可以增加显式的类型转换。

答:错误显示是该函数不存在,实际上是因为缺少了参数。正确的调用方法如下:

```
SELECT SUBSTRING('this is a test',6,2); SELECT SUBSTRING('this is a test',6);
```

(24) 创建表时,定义多个主码的错误。例如:

```
CREATE TABLE Sales.PartSuppTest(
  partkey INT PRIMARY KEY,
  suppkey INT PRIMARY KEY,
  availqty INT,
  supplycost REAL,
  comment VARCHAR(199));
```

该 SQL 语句执行出错:

ERROR:表 "PARTSUPP" 不允许指定多个主码。

答:因为该表的主码由两个属性构成,故主码约束不能定义成列级完整性约束,必须定义为表级完整性约束。正确的形式如:

```
CREATE TABLE Sales.PartSupp(
  partkey INT,
  suppkey INT,
  availqty INT,
  supplycost REAL,
  comment VARCHAR(199),
PRIMARY KEY(partkey,suppkey));
```

(25) 插入数据由于数据类型不对导致的错误。例如插入一条顾客记录:

```
INSERT INTO Sales.Customer(custkey,name,mktsegment,acctbal)
VALUES(123456,'Mike',0.0,'北美区');
```

该插入 SQL 语句执行出错:

ERROR:用于类型 real 的无效输入语法:"北美区"。

答:acctbal 属性是 real 类型,应该输入数字类型值,而不是字符串值,即使是字符串类型的值,也应该可自动转换成数值类型的字符串。正确的写法:

```
INSERT INTO Sales.Customer(custkey,name,mktsegment,acctbal)
VALUES(234567890,'Mike','北美区',0.00);
```

(26) 有计算列时创建视图省略视图属性名的问题。例如:

```
CREATE VIEW Sales.V_PartTotalValue AS
SELECT P.partkey,PS.availqty * P.retailprice  /*这是一个计算列*/
FROM Sales.Part P,Sales.PartSupp PS
```

```
WHERE P.partkey=PS.partkey;
```

该 SQL 语句执行不会出错,系统会自动给计算列命名,并显示为?COLUMN?。

答:创建视图时,虽然系统会自动给计算列命名,但是该计算列较难引用。处理办法之一是给计算列重命名:

```
DROP VIEW Sales.V_PartTotalValue;
CREATE VIEW Sales.V_PartTotalValue AS
    SELECT P.partkey,PS.availqty * P.retailprice AS totalvalue
    FROM Sales.Part P,Sales.PartSupp PS
    WHERE P.partkey=PS.partkey;
```

处理方法之二是直接给视图属性命名:

```
DROP VIEW Sales.V_PartTotalValue;
CREATE VIEW Sales.V_PartTotalValue(partkey,totalvalue) AS
    SELECT P.partkey,PS.availqty * P.retailprice
    FROM Sales.Part P,Sales.PartSupp PS
    WHERE P.partkey=PS.partkey;
```

(27) WHERE 和 HAVING 条件混淆导致的问题。例如:

```
SELECT PS.suppkey,SUM(PS.availqty * P.retailprice)
FROM Sales.Part P,Sales.PartSupp PS
WHERE P.partkey=PS.partkey AND SUM(PS.availqty * P.retailprice)>1000000;
GROUP BY PS.suppkey
```

该 SQL 语句执行出错:

ERROR:在 WHERE 子句中不能使用聚集函数。

答:聚集函数一般用在分组统计 SQL 语句中,并且只能出现在 SELECT 子句和 HAVING 子句中。WHERE 子句是元组过滤条件,而 HAVING 子句是分组过滤条件。因此,上述 SQL 语句正确的写法是:

```
SELECT PS.suppkey,SUM(PS.availqty * P.retailprice)
FROM Sales.Part P,Sales.PartSupp PS
WHERE P.partkey=PS.partkey
GROUP BY PS.suppkey
HAVINGSUM(PS.availqty * P.retailprice)>1000000;
```

(28) 批量导入数据时,有属性缺少数据。例如:

```
COPY Sales.Part FROM 'D:\tpchdata\part.csv' WITH DELIMITER AS',';
```

该 SQL 语句执行出错:

ERROR:列"CONTAINER"缺少数据。

答:该错误信息表明 part.csv 中没有 container 属性的数据。因此,该 COPY 语句应指明需要导入数据的属性列表,如:

```
COPYSales.Part( partkey,name,mfgr,type,retailprice)
FROM 'D:\tpchdata\part.csv' WITH DELIMITER AS',';
```

(29) 丢失条件导致的逻辑错误。如把 1 号供应商供应的 1 号零件改为 2 号供应商供应:

```
UPDATE Sales.PartSupp
SET suppkey=2
WHERE suppkey=(SELECT suppkey
                FROM Sales.PartSupp
                WHERE suppkey=1 AND partkey=1);
```

该 SQL 语句执行不会出错,但是丢失了条件 partkey=1。

答:上述 UPDATE 语句有逻辑错误,WHERE 条件中的嵌套 SQL 子查询,得出结果为 1,因此 WHERE 条件实际上为 suppkey=1,丢失了 partkey=1 的条件。正确的写法是:

```
UPDATE Sales.PartSupp
SET suppkey=2
WHERE suppkey=1 AND partkey=1;
```

附录

附录 A 介绍数据库领域 4 位图灵奖获得者的事迹。

2015 年 3 月 25 日，ACM 宣布迈克尔·斯通布雷克（Michael Stonebraker）获得 2014 年度图灵奖，我们马上整理了他的贡献和传奇事迹，奉献给读者。

附录 B 介绍数据库基准测试 TPC-C 和 TPC-H。

一个大型通用的 DBMS 软件必须经过严格的测试，数据库基准（Benchmark）测试是研究和开发 DBMS 的重要技术，附录 B 的内容可以作为《概论》的补充资料，也可供数据库研究、开发与应用的技术人员参考。

附录 A　数据库领域图灵奖获得者

数据库技术是计算机科学技术中发展最快的领域之一，也是应用最广的技术之一，它是计算机信息系统与应用系统的核心技术和重要基础。

数据库技术从 20 世纪 60 年代中期产生，至今仅仅 40 多年的历史，却已经历了三代演变，造就了查尔斯·巴赫曼、埃德加·科德、詹姆斯·格雷和迈克尔·斯通布雷克 4 位图灵奖得主。

这里向大家介绍这 4 位图灵奖得主的风采和成就。

A.1　1973 年图灵奖获得者：查尔斯·巴赫曼*——“网状数据库之父”

20 世纪 60 年代中期以来，数据库技术的形成、发展和成熟使得计算机数据处理技术跃上了一个新台阶，从而极大地推动了计算机的普及与应用。1973 年的图灵奖首次授予在数据库方面作出杰出贡献的先驱查尔斯·巴赫曼（Charles W.Bachman）。

为了说明巴赫曼的功绩，下面先简要回顾一下计算机数据处理的历史。

计算机在 20 世纪 40 年代诞生之初仅用于科学与工程计算，不能用于数据处理，因为当时的计算机还只能处理数字，不能处理字母和符号。此外，当时的计算机也还没有数据处理所需要的大容量存储器。20 世纪 50 年代初发明了字符发生器（character generator），使计算机具有了能显示、存储与处理字母及各种符号的能力，高速磁带机成功地用于计算机作为存储器，这是计算机得以进入数据处理领域具有决定意义的两大技术进展。但是磁带只能顺序读写，速度也慢，不

---

* A.1～A.3 内容主要摘录自吴鹤龄教授和崔林老师所编写的《ACM 图灵奖（1966—2001）——计算机发展史的缩影》（增订版）。The ACM Turing Award（1966-2001）：An Epitomized History of Computing in the Twentieth-Century，高等教育出版社，2002 年 9 月。

是理想的存储设备。1956 年,IBM 公司和 Remington Rand 公司先后实验成功磁盘存储器方案,推出了商用磁盘系统。磁盘不但转速快、容量大,还可以随机读写,为数据处理提供了更加理想的大容量、快速存储设备。有了这些硬件的支持,计算机数据处理便日益发展起来。

但是,初期的数据处理软件只有文件管理。数据文件和应用程序一一对应,造成数据冗余、数据不一致性和数据依赖。所谓数据依赖就是编写程序依赖于具体数据,也就是数据缺乏与程序的独立性。以 COBOL 这种常用的商用语言为例,程序员必须在数据部的文件节(data division,file section)中详细说明文件中各数据项的类型和长度、格式,在设备环境部的输入/输出节(environment division,input-output section)中还要通过 SELECT 语句和 ACCESS MODE 语句严格规定文件的组织方式和存储方式。根据这些具体规定,程序员再在过程部(procedure division)中用一系列命令语句进行导航式的操作,才能使系统完成预期的数据处理任务。应用程序与数据的存储、存取方式密切相关这种状况给程序的编制、维护都造成很大的麻烦。

后来出现了文件管理系统(File Management System,FMS)作为应用程序和数据文件之间的接口,一个应用程序通过 FMS 可以和若干文件打交道,在一定程度上增加了数据处理的灵活性。但这种方式仍以分散的、互相独立的数据文件为基础,数据冗余、数据不一致性、处理效率低等问题仍不可避免。这些缺点在较大规模的系统中尤为突出。以美国在 20 世纪 60 年代初制定的阿波罗登月计划为例,阿波罗飞船由大约 200 万个零部件组成,分散在世界各地制造生产。为了掌握计划进度及协调工程进展,阿波罗计划的主要合约者 Rock-well 公司曾研制、开发了一个基于磁带的零部件生产计算机管理系统。系统共用了 18 盘磁带,虽然可以工作,但效率极低。18 盘磁带中 60%是冗余数据,维护十分困难。这个系统的状况曾一度成为实现阿波罗计划的重大障碍之一。

针对上述问题,计算机学术界和工业界纷纷开展研究,为改革数据处理系统进行探索与试验,其目标主要就是突破文件系统分散管理的弱点,实现对数据的集中控制,统一管理,其成果就是出现了一种全新的高效的数据管理技术——数据库技术。

Rockwell 公司与 IBM 公司合作,在当时新推出的 IBM 360 系列上研制成功了世界上最早的数据库管理系统之一 IMS(Information Management System)。IMS 为保证阿波罗飞船 1969 年顺利登月作出了贡献。IMS 是基于层次模型的。几乎同时,巴赫曼在通用电气公司主持设计与实现了网状的数据库管理系统 IDS(Integrated Data System)。

巴赫曼 1924 年 12 月 11 日出生于美国堪萨斯州的曼哈顿。1948 年在密歇根州立大学取得工程学士学位,1950 年在宾夕法尼亚大学取得硕士学位。20 世纪 50 年代在 Dow 化工公司工作,1961—1970 年在通用电气公司任程序设计部门经理,1970—1981 年在 Honeywell 公司任总工程师,同时兼任 Cullinet 软件公司的副总裁和产品经理。Cullinet 公司当时在美国很有名气,它是 1978 年第一家在纽约股票交易所上市的软件公司。当时微软公司在新墨西哥州的阿尔伯克基成立不久,鲜为人知。微软的股票是 1986 年上市的,比 Cullinet 晚 8 年之久。但 Cullinet 最

终被 CA 公司购并。1983 年巴赫曼创办了自己的公司 Bachman Information System, Inc.。

巴赫曼在数据库方面的主要贡献有以下两个方面：

其一是在通用电气公司任程序设计部门经理期间，主持设计与开发了最早的网状数据库系统 IDS。IDS 于 1964 年推出后，成为最受欢迎的数据库产品之一，而且它的设计思想和实现技术被后来的许多数据库产品所效仿。

其二是巴赫曼积极推动与促成了数据库标准的制定，这就是美国数据系统语言委员会 CODASYL 下属的数据库任务组 DBTG 提出的网状数据库模型以及数据定义和数据操纵语言(即 DDL 和 DML)的规范说明。DBTG1971 年推出了第一个正式报告——DBTG 报告，成为数据库历史上具有里程碑意义的文献。该报告中基于 IDS 的经验所确定的方法称为 DBTG 方法或 CODASYL 方法；所描述的网状模型称为 DBTG 模型或 CODASYL 模型。DBTG 曾希望美国国家标准委员会(ANSI)接受 DBTG 报告为数据库管理系统的国家标准，但是没有成功。1971 年报告之后，又推出了一系列新版本，如 1973 年、1978 年、1981 年和 1984 年的修改版本。DBTG 后来改名为 DBLTG(DataBase Language Task Group，数据库语言工作小组)。

DBTG 首次确定了数据库的三层体系结构，明确了数据库管理员的概念，规定了其作用与地位。DBTG 系统虽然是一种方案而非实际的数据库系统，但它所提供的基本概念却具有普遍意义，不但国际上大多数网状数据库管理系统，如 IDMS、PRIME DBMS、DMS 170、DMS Ⅱ 和 DMS 1100 等都遵循或基本遵循 DBTG 模型，而且对后来产生和发展的关系数据库技术也有很重要的影响，其体系结构也遵循 DBTG 的三级模式(虽然名称有所不同)。DBTG 的系统结构如图 3 所示，主要包括模式、子模式、物理模式、数据操纵和数据库管理系统等几个部分。

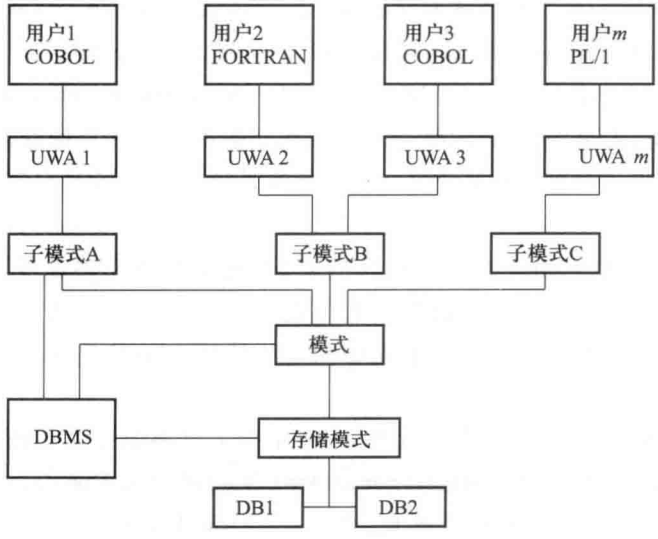


图 3　DBTG 的系统结构

模式是对数据库整体数据逻辑结构的描述，它对应数据库的逻辑层，由数据库管理员借助模式数据描述语言建立。子模式是某一用户对其所关心的那部分数据的数据结构的描述，对应于数据库的外层或用户视图，是由该用户自己或委托数据库管理员借助子模式数据描述语言加以定义的。物理模式也叫存储模式，是对数据库数据的存储组织方式的描述，对应于数据库的物理层，由数据库管理员通过数据存储描述语言（Data Storage Description Language，DSDL）加以定义。DSDL 是 DBTG 报告的 1978 年版本提出的，之前的报告用的名称叫数据介质控制语言（Data Media Control Language，DMCL）。

数据库可由多个用户、多个应用共享，数据库应用程序利用数据操纵语言实现对数据库数据的操纵。但一个应用程序必须引用某一模式的某一子模式（也就是说它操作的数据限于某一用户视图中的数据）。DML 语句可以嵌在主语言（如当时的 COBOL、FORTRAN，现在的 C、C++等）中，在数据库管理系统的控制下访问数据库中的数据，并通过一个称为用户工作区的缓冲区与数据库通信，完成对数据库的操作。数据库管理系统的其他功能包括维护数据库中数据的一致性、完整性、安全性和一旦出现故障情况下的恢复，以及在多个应用程序同时存取同一个数据单元时并发控制等，以避免出现“脏数据”或“丢失更新”等异常现象。

由此可见，模式数据描述语言是建立数据库的工具，数据操纵语言是操作数据库、存取数据库中数据的工具，而数据库管理系统则是执行这些操作并负责维护与管理数据库的工具，它们各司其职，完成数据库系统整个生命期中的一切活动。

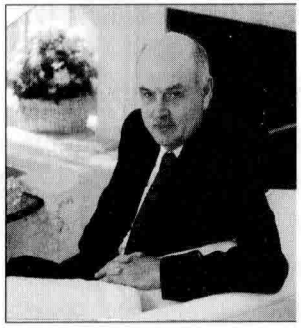
由于以上两个方面的杰出贡献，巴赫曼被理所当然地公认为“网状数据库之父”或“DBTG 之父”，在数据库技术的产生、发展与推广应用方面都发挥了巨大的作用。

在数据库的文档中，有一种描述网状数据库模型的数据结构图。这种图解技术是巴赫曼发明的，通常称为“巴赫曼图”（Bachman diagram）。此外，在担任 ISO/TC 97/SC-16 主席时，巴赫曼还主持制定了著名的“开放系统互连”标准（Open System Interconnection，OSI）。OSI 对计算机、终端设备、人员、进程或网络之间的数据交换提供了一个标准规程，实现 OSI 对系统之间达到彼此互相开放具有重要意义。巴赫曼也是建立在美国波士顿的计算机博物馆的创始人之一。

20 世纪 70 年代以后，由于关系数据库的兴起，网状数据库逐渐受到冷落。但随着面向对象技术的发展，有人认为网状数据库有可能重新受到人们的青睐。但无论这个预言是否能够实现，巴赫曼作为数据库技术先驱的历史作用和地位是学术界和产业界普遍认可的。

1973 年 8 月 28 日，在亚特兰大举行的 ACM 年会上巴赫曼接受了图灵奖。他发表了题为“作为导航员的程序员”（The Programmer as Navigator）的图灵奖演说，刊载于 1974 年 11 月的 Communications of ACM 第 653～658 页，也可见《ACM Turing Award Lectures——The First 20 Years：1966—1985》，ACM，269～286 页（《前 20 年的 ACM 图灵奖演说集》）。

A.2　1981 年图灵奖获得者:埃德加·科德——“关系数据库之父”



在数据库技术发展的历史上,1970 年是发生伟大转折的一年。这一年的 6 月,IBM 圣约瑟研究实验室的高级研究员埃德加·科德(Edgar Frank Codd)在 Communications of ACM 上发表了题为“A Relational Model of Data for Large Shared Data Banks”(大型共享数据库的关系数据模型)一文。ACM 后来在 1983 年把这篇论文列为从 1958 年以来的 1/4 世纪中具有里程碑式意义的最重要的 25 篇研究论文之一。因为它首次明确而清晰地为数据库系统提出了一种崭新的模型,即关系模型。

“关系”是数学中的一个基本概念,由集合中的任意元素所组成的若干有序偶对(ordered pair)表示,用以反映客观事物间所存在的一定关系,如数之间的大小关系、一个组织中的成员之间的领导与被领导关系、商品流通中的购销关系、产品零部件之间的装配关系,等等。在自然界和社会中,关系是无处不在的。在计算机学科中,关系的概念也十分普遍。计算机的逻辑设计、编译程序设计、算法分析和程序结构、信息检索等,都应用了关系的概念。而用关系的概念来建立数据模型,用以描述、设计与操纵数据库,则是科德 1970 年的这篇论文的创举。

由于关系模型简单明了,有坚实的数学基础,一经提出,立即引起学术界和产业界的广泛重视和响应,从理论与实践两个方面都对数据库技术产生了强烈的冲击。在关系模型提出之前,已经存在多年的基于层次模型和网状模型的数据库产品很快走向衰败,一大批关系数据库系统很快被开发出来并迅速商品化,占领了大部分市场份额。其交替速度之快,除旧布新之彻底是软件史上所罕见的。基于 20 世纪 70 年代中后期和 80 年代初期这一令世人瞩目的成就,1981 年的图灵奖很自然地授予了这位“关系数据库之父”。

科德原是英国人,1923 年 8 月 19 日生于英格兰中部濒临大西洋的港口城市波特兰(Portland)。第二次世界大战爆发以后,年轻的科德应征入伍,在皇家空军服役。1942—1945 年间任机长,参与了许多惊心动魄的空战,为反法西斯战争立下了汗马功劳。“二战”结束以后,科德进了牛津大学学习数学,于 1948 年取得学士和硕士学位后,远渡大西洋到美国谋求发展。先在 IBM 公司取得一个职位,为 IBM 初期的计算机之一 SSEC(Selective Sequence Electronic Calculator)编制程序,为其个人的计算机生涯奠定了基础。1953 年,他应聘到加拿大渥太华的 Computing Device 公司工作,出任加拿大开发导弹项目的经理。1957 年他重返美国 IBM,任“多道程序设计系统”(Multiprogramming Systems)的部门主任,其间参加了 IBM 第一台科学计算机 701,第一台大型晶体管计算机 STRETCH 的逻辑设计。

STRETCH 完成于 1961 年,最多可重叠执行 6 条连续的指令,是后来流水线方式的原型,因而被认为是第一台流水线计算机。它还采用交换器和多道程序技术、用多个存储器交叉工作等许多创新技术,因而在计算机发展史上有重要意义和影响。科德在 STRETCH 的研制中主持了第一个有多道程序设计能力的操作系统的开发。1959 年 11 月,他在 Communications of ACM 上发表的介绍 STRETCH 的多道程序操作系统的文章,是这方面最早的学术论文之一。

尤其难能可贵的是,科德在工作中发觉自己缺乏硬件方面的知识,影响了在这些重大工程中发挥更大的作用,遂于 20 世纪 60 年代初毅然决定重返大学校园(当时他已年近 40),到美国密歇根大学进修计算机与通信专业,并于 1963 年获得硕士学位,1965 年获得博士学位。这使他的理论基础更加扎实,专业知识更加丰富,加上他在此之前十几年丰富实践经验的积累,终于在 1970 年迸发出智慧的闪光,为数据库技术开辟了一个新时代。

由于数据库是计算机各种应用的基础,关系模型的提出不仅为数据库技术的发展奠定了基础,同时也为计算机的普及应用提供了强大的动力。在科德提出关系模型以后,IBM 投巨资开展关系数据库管理系统的研究,其 System R 项目的研究成果极大地推动了关系数据库技术的发展。在此基础上推出的 DB2 和 SQL 等产品成为 IBM 的主流产品。System R 本身虽然只是原型,但鉴于其作用与影响,ACM 把 1988 年的"软件系统奖"授予了 System R。获奖的开发小组 6 个成员中就包括后来在 1998 年荣获图灵奖的格雷(J.Gray)。这一年的软件系统奖还破例同时奖励了两个软件系统,另一个得奖软件也是关系数据库管理系统,即 INGRES。INGRES 是加州大学伯克利分校的斯通勃莱克(M.Stonebracker)等人研制的,后来由美国关系技术公司 RTI 商品化。

1970 年以后,科德继续致力于完善和发展关系理论。1972 年,他提出了关系代数和关系演算,定义了关系的并、交、差、投影、选择、连接等各种基本运算,为日后成为标准的结构化查询语言(SQL)奠定了基础。科德还创办了一个研究所:关系研究所(The Relational Institute)和一个公司:Codd & Associations,进行关系数据库产品的研发与销售。科德本人则是美国国内和国外许多企业的数据库技术顾问。1990 年,他编写出版了专著《The Relational Model for Database Management》,Version 2,Addison-Wesley(《数据库管理的关系模型(第 2 版)》),全面总结了他几十年的理论探索和实践经验。

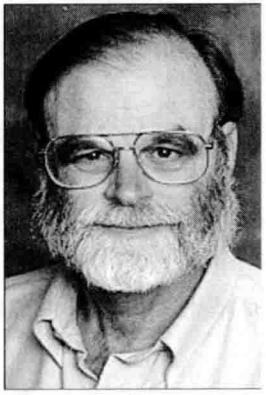
科德是美国工程院院士。

向科德颁发图灵奖的仪式是 1981 年 11 月 9 日在洛杉矶召开的 ACM 年会上举行的。由 ACM 主席邓宁(P.Denning)亲自授奖并致词。科德发表了题为"Relational Database:A Practical Foundation for Productivity"(关系数据库:提高生产率的实际基础)的演说,刊于 1982 年 2 月的 Communications of ACM,109—117 页,或见《ACM Turing Award Lectures——The First 20 Years:1966—1985》,ACM,391—410 页(《前 20 年的 ACM 图灵奖演说集》)。演说中,科德说明了他当初提出关系模型的动机,强调了数据操纵语言既要有交互能力,又要能嵌入主语言

程序的重要性,这是信息系统特殊的应用方式所决定的。

科德于 1984 年从 IBM 退休,2003 年谢世。

A.3　1998 年图灵奖获得者:詹姆斯·格雷——数据库技术和“事务处理”专家



1998 年度的图灵奖授予了声誉卓著的数据库专家詹姆斯·格雷(James Gray),或称吉姆·格雷(Jim Gray,Jim 是 James 的昵称)。

这是图灵奖诞生 32 年的历史上,继数据库技术的先驱查尔斯·巴赫曼(Charles W.Bachman,1973)和关系数据库之父埃德加·科德(Edgar F.Codd,1981)之后,第三位因在推动数据库技术的发展中做出重大贡献而获此殊荣的学者。

格雷生于 1944 年,在美国著名的加州大学伯克利分校计算机科学系获得博士学位。其博士论文是有关优先文法语法分析理论的。学成以后,他先后在贝尔实验室、IBM、Tandem、DEC 等公司工作,研究方向转向数据库领域。

在 IBM 期间,他参与和主持过 IMS、System R、SQL/DS、DB2 等项目的开发,其中除 System R 仅作为研究原型,没有成为产品外,其他几个都成为 IBM 在数据库市场上颇具影响力的产品。

在 Tandem 期间,格雷对该公司的主要数据库产品 ENCOMPASS 进行了改进与扩充,并参与了系统字典、并行排序、分布式 SQL、Nonstop SQL 等项目的研制工作。

在 DEC,他仍然主要负责数据库产品的技术工作。格雷进入数据库领域时,关系数据库的基本理论已经成熟,但各大公司在关系数据库管理系统(RDBMS)的实现和产品开发中,都遇到了一系列技术问题。主要是在数据库的规模愈来愈大,数据库的结构愈来愈复杂,又有愈来愈多的用户共享数据库的情况下,如何保障数据的完整性、安全性、并发性,以及一旦出现故障后,数据库如何从故障中恢复。这些问题如果不能圆满解决,无论哪个公司的数据库产品都无法进入实用,最终不能被用户所接受。正是在解决这些重大的技术问题,使 DBMS 成熟并顺利进入市场的过程中,格雷以他的聪明才智发挥了十分关键的作用。

目前,各 RDBMS 解决上述问题的主要技术手段和方法如下:

① 数据库的操作划分为“事务”(或“事务元”,transaction)的一个个原子单位。事务是事务处理的基本执行单位,即一个事务中的操作要么全部被执行,要么全部都不执行,即实行所谓 all or none 的原则。一个事务一般以一个“开始”语句(begin)启动,先从数据库中取出一些数据,然后进行所需的处理,最后以“提交”语句(commit)结束。如事务中发生异常,则用“异常终止”语句(abort)或“回退”语句(rollback)撤销本事务执行过程中对数据库已做的

所有更新(即所谓 undo),将数据库恢复到事务开始时的正确状态,以保障数据的完整性、一致性。

② 用户在对数据库发出操作请求时,系统对不同粒度(granularity)的数据元素(字段、记录以至整个文件)"加锁"(locking)。加锁的数据被暂时禁止其他用户访问(这里仅是一种简化的解释,实际上,根据用户对数据请求的不同性质,加锁的数据如何对待另一用户的请求有多种复杂的情况。例如,如果加锁的数据将被修改,那是绝对禁止其他用户访问的;而如果加锁的数据只用于读出,则其他用户的读出请求还将是允许的。这由所谓"锁相容性矩阵"lock compatibility matrix 来管理和控制)。操作完成后"解锁"(unlocking)。这一机制用以既保持事务之间的"并发性",又保证数据的"完整性"。

③ 建立系统运行日志(log),记载各事务的起点、终点以及在事务中被更新过的页面的改前状态和改后状况(before image 和 after image),以便在系统出现故障使数据库遭到破坏时,能根据定期或不定期为数据库所作的备份(backup)加上日志中的信息将数据库恢复到系统故障前的正确状态,同时又能保留最后一次备份以来对数据库所作的修改。

④ 对数据库的任何更新分两阶段提交(two-phase commit)。这是基于一个事务可能同时涉及两个不同的数据库系统而必需的,这在分布式系统中尤为重要。

上述及其他各种方法可总称为"事务处理技术"(transaction processing technique)。格雷在事务处理技术上的创造性思维和开拓性工作,使他成为该技术领域公认的权威。他的研究成果反映在他发表的一系列论文和研究报告之中,最后结晶为一部厚厚的专著《Trasaction Processing Concepts and Techniques》Morgan Kaufmann Publishers,1993。另一作者为德国斯图加特大学的 A.Reuter 教授。事务处理技术虽然诞生于数据库研究,但对于分布式系统,Client/Server 结构中的数据管理与通信,对于容错和高可靠性系统同样具有重要的意义。

格雷的另一部著作是《The Benchmark Handbook:for Database and Transaction Processing Systems》。第 1 版于 1991 年出版,第 2 版于 1993 年出版,也是 Morgan Kanfmann 出版社出版的。格雷还是该出版社"数据管理系统丛书"的主编。

格雷在数据库学术界十分活跃。国际上定期或不定期举行的一些重要的数据库学术会议如 VLDB、SIGMOD 上都能见到他的身影,听到他的声音。除了在公司从事研究开发外,他还兼职在母校伯克利、斯坦福大学、布达佩斯大学从事过教学和讲学活动。1992 年,VLDB 杂志(The VLDB Journal)创刊,他出任主编。

格雷是 1988 ACM 授予 IBM 的 System R 以软件系统奖的 6 位得奖人之一,其他 5 人是 Donald Chamberlin、Raymond Lorie、Gianfranco Putzolu、Patricia、Selinger 和 Irving Traiger。

正是由于格雷在数据库技术方面的声誉,软件业中的"巨无霸"微软公司在 1993 年决定进入大型关系数据库市场时,不惜用种种手段把格雷从 DEC 公司挖过来。格雷不喜欢微软公司总部所在的多雨潮湿的西雅图,愿意留在阳光灿烂的旧金山,微软公司特地在旧金山开辟了第二个微软研究院——"湾区研究中心"(Bay Area Research Center,BARC),安排格雷任

该研究院主管。格列果然不负众望,领导研制小组开发出 MS SQL Server 7.0,成为微软公司历史上一个里程碑式的版本,而且也成为当今关系数据库市场上的佼佼者。

格雷是在 1999 年 5 月 4 日于美国亚特兰大举行的 ACM 全国会议上接受图灵奖的。他发表了"What Next? ——A Dozen Remaining IT Problems"(信息技术今后的目标)演说,纵论了信息技术发展中有关的几个方向性问题。后来,该文经修改后在 SIGMOD 会上以"What Next? ——A dozen IT Research Goals"为题再次发表。

格雷的演说在对计算技术的发展作总结性回顾时认为,英国数学家巴贝奇(Charles Babage,1791—1871)在 19 世纪所梦想和追求的计算机今天已经基本实现;美国数学家布什(Vannever Bush ,1890—1974,曾任罗斯福总统的科学顾问)20 世纪 40 年代所设想的"梅米克斯"MEMEX,即"记忆延伸器"(MEMory Extender),当前已接近实现;而图灵所提出的智能机器离实现还有一段距离,目前的计算机还难以通过"图灵测试"。

为了实现上述三位科学巨人的理想,格雷呼吁美国政府要重视对 IT 技术研究的长期支持,认为其重要意义不亚于 200 年前杰弗逊(Thomas Jefferson,1743—1826,"独立宣言"的起草者,美国第三任总统,1801—1809 在位)决定用 1500 万美元从法国政府手中买回路易斯安娜领地(Louisiana Territory,这是位于密西西比河和洛矶山脉之间,北至加拿大,南达墨西哥湾的大块土地,面积达 2 070 000 km^2)这一称为 Louisiana Purchase 的著名历史事件,然后又派出以刘易斯上尉(Captain Meriwether Lewis)和克拉克(William Clark)为首的"发现军团(corps for discovery)"到北美洲西部探险直至太平洋海岸,为最终形成美国如今的版图奠定了基础。

格雷认为,一个好的 IT 长期目标应具有以下 5 个关键性质:

① 可理解性:目标应能简单表述并被人理解。

② 有挑战性:如何达到目标不是很明显。

③ 用途广泛:不只对计算机科学家有用,而且可以使大多数人受益。

④ 可测试性:可以检查项目进展并知道目标是否已经达到。

⑤ 渐进性:中间有若干里程碑意义的成果,以检查项目进展情况并鼓舞研究人员继续工作。

在以上论点支持下,格雷提出了 IT 技术的长远研究目标如下:

① 规模可伸缩性(scalability)。

② 通过图灵测试。

③ 语音到文本的转换(speech to text)。

④ 文本到语音的转换(text to speech)。

⑤ 机器视觉,能像人一样识别物体和运动。

⑥ 个人的"梅米克斯",可记录人所看到和听到的一切,需要时可以快速检索出来。

⑦ 世界的"梅米克斯",即建立文本、音乐、图像、艺术、电影的"大全"(corpus),可回答有

关的任何提问,像人类专家那样快而好的做索引、做文摘。

⑧ 虚拟现实。

⑨ 无故障系统(trouble-free systems)。

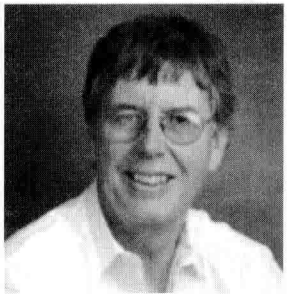
⑩ 安全系统(secure systems)。

⑪ 高可用系统(always UP)。

⑫ 自动程序设计(automatic programming)。

格雷于 2007 年 1 月 28 日失踪。在失踪前的数年中,他从事 Scalability 这一长期目标的研究。他是微软公司"规模可伸缩的服务器研究小组"(Scalable Servers Research Group)的高级研究员。该项目已有若干研究成果在网上公布。

A.4 2014 年图灵奖获得者:迈克尔·斯通布雷克——现代主流数据库系统架构的奠基人



2015 年 3 月 25 日,ACM 宣布 MIT 计算机和人工智能实验室研究员迈克尔·斯通布雷克(Michael Stonebraker)由于他在现代数据库系统概念和实践方面做出的奠基性贡献而获得 2014 年度图灵奖,并将获得 Google 首次资助的 100 万美元奖金。

Michael Stonebraker 生于 1943 年 10 月 11 日,是著名的数据库学者。他于 1965 年在普林斯顿大学获得电气工程专业的科学和工程学士学位,并分别于 1966 年和 1971 年在密歇根大学获得电气工程专业的科学和工程硕士学位和计算机信息与控制专业的博士学位。

1971 年开始,他在加州大学伯克利分校计算机系任教 29 年;2001 年从伯克利退休后,他成为 MIT 计算机科学与人工智能实验室的兼职教授。

Michael Stonebraker 于 1974 年获得 SIGMOD 第一届 Edgar F.Codd 创新奖,1992 年获得著名的 ACM 系统软件奖,1994 年获得 ACM SIGMOD 最佳年度创新奖。此外,他于 1994 年成为 ACM 会士,1997 年当选为美国国家工程院院士,2005 年被授予 IEEE 冯·诺依曼奖章。

作为数据库系统架构的先行者,Stonebraker 在 40 多年的科研生涯中不断设计、实现和推广新的数据库架构和管理技术,并在工业界和商业市场中连续取得了巨大成功。Stonebraker 在数据库领域的研究工作可以大致分为三个阶段:

① 1971 年至 2000 年为第一阶段,从事关系数据库的体系架构与实现技术研究。

从教之初,Stonebraker 领导的 Ingres 项目和其后继的 Postgres 项目为现代数据库系统奠定了系统架构基础。1974 年,Stonebraker 在加州大学伯克利分校启动了 Ingres 项目,在低端机器上实现了关系数据模型[15]。其中 B 树、主从备份、实例化视图、利用查询重写进行访问

控制、利用触发器机制进行完整性检查等关键技术被广泛成功应用到实际系统中。系统架构上的成功让 Ingres 项目收获了商业上的成功。

1979 年,Stonebraker 基于 Ingres 数据库创建了 Relational Technology 公司,1982 年创建了 Ingres 公司,Stonebraker 在该公司工作到 1991 年,之后公司由 ASK 收购,1994 年 ASK/Ingres 又被 CA Computer Associates 收购。

Postgres 于 1985 年启动。它实现了对象关系数据模型的系统架构,允许用户定义复杂的数据类型及其上的操作,进而形成了当前被广泛采用的 PostgreSQL 数据库。1990 年 Stonebraker 为了商业化 Postgres 成立了 Illustra 公司,该公司后来被 Informix 并购。现代数据库 EnterpriseDB、Aster Data System 和 Greenplum 等就是基于 PostgreSQL 的系统架构开发的。

针对分布式数据库,Stonebraker 在组织开发分布式数据库系统 Distributed Ingres 中,对分布式查询处理和事务一致性协议做出了持续性的技术贡献。接着,他主导了非常有影响力的基于 Postgres 并行版本的 XPRS 项目。在这个项目中,Stonebraker 探索了“share nothing”可扩展的数据库系统架构,并成为该架构的早期推动者。目前,数据库厂商提供的大数据解决方案将此系统架构奉为圭臬。

1992 年,Stonebraker 主导了 Mariposa 项目,探索了联邦数据库的系统架构。由于不同组织机构具有不同的数据存储和处理的收费方式,联邦数据库的系统架构基于一个经济模型。这使得传统的查询优化算法能够以经济模型为目标,服务于数据在联邦数据库中的存储、复制和移动。这个项目催生了 Cohera Corporation 公司,其联邦数据库的架构由于支持企业间的目录管理应用而在 2001 年被 PeopleSoft 公司购买,2004 年又被 Oracle 收购。

② 2001 年至 2008 年为第二阶段,在 One size does not fit all 的理念下,进行了一系列新型数据库系统的体系架构设计与产品开发。

2001 年 Stonebraker 任教 MIT 后意识到数据库需要不同的架构应对不同的应用问题,因此他提出了著名的观点“one-size does not fit all”[16],并以此为指引开启了一系列数据库新架构的研究。他首先启动了针对流数据的 Aurora 项目。面对新的数据模型和查询语言,Aurora 项目设计了新的系统架构,一方面以股票市场、传感器等数据源随时产生、推送的数据为输入,另一方面以流的形式将处理结果发送给用户。2003 年,Stonebraker 成立 StreamBase Systems 公司,把此架构商业化。

2005 年,Stonebraker 启动 C-Store 项目,为数据仓库设计了并行的、share nothing 的、列存储的数据库架构。这样的架构比传统的行存储架构使用更少的 I/O 并达到更好的压缩比。同年,Stonebraker 成立 Vertica 公司商业化 C-Store 的技术架构。

2006 年,针对数据集成时需要部署一系列中间转换器为网站及其服务提供查询接口的需求,Stonebraker 启动了 Morpheus 项目,设计了新的数据集成系统的架构。该架构允许查询多个中间转换器并加以集成和合并,因此可以为多种新的服务提供统一的视图。2009 年 Stonebraker 利用 Morpheus 带来的技术成立了本地搜索公司 Goby。

2007 年,针对 OLTP 系统高吞吐量的需求,Stonebraker 启动了 H-store 项目,设计了分布式内存 OLTP 系统。新系统在分布式内存架构上的成功使得 Stonebraker 在 2009 年成立了 VoltDB 公司。

2008 年,针对科学研究、空间地理、金融等领域广泛使用多维数组的需求,Stonebraker 启动了 SciDB 项目,设计了数组数据库系统架构。新的架构具有 4 点特性:

- 以内建数组的方式支持多维数据的组织、存储和检索;
- 内建数组实现了对多维数据高效的投影和连接操作;
- 具有支持复杂分析和预测模型的基础数据结构矩阵;
- 使用可以横向扩展的分布式大规模并行处理架构。Stonebraker 成立了 Paradigm4 公司推广 SciDB。

③ 2009 年至今为第三阶段,大数据系统的体系架构设计与实践。

云计算时代,新的系统架构,如 Google 公司提出的 Google File System、MapReduce 等分布式文件系统和并行处理框架应运而生,被业界广泛采用。但 Stonebraker 在系统架构变革中一直坚持自己的判断。

2009 年,Stonebraker 率先为 MapReduce 和并行数据库设计性能测评[17],并撰文[18]指出并行数据库模型与 MapReduce 模型并无高下之分。他认为不同的系统架构适合不同的任务,MapReduce 适合那些只对原始数据处理一遍的任务,而数据库适合对数据集进行反复频繁的查询处理任务。他预计未来 MapReduce 将被并行数据库取代。实际上,到 2014 年,Google 开发出 MapReduce 的替代产品后,也声明由于 MapReduce 不能处理公司面对的数据量而放弃 MapReduce。

2010 年,Stonebraker 分析指出了 NoSQL 架构中存在的缺陷,包括牺牲了事务的强一致性,以及日志、锁、栓和缓冲区管理 4 个主要的性能瓶颈,只有克服了上述 4 个瓶颈,才有可能取代 SQL 数据库。否则像 NoSQL 数据库那样依然采用传统的硬盘管理、多线程技术和弱化的事务是不能取代 SQL 数据库的。通过对 NoSQL 运动的反思,Stonebraker 倡导新的数据库架构 New SQL,即实现 SQL 接口、支持事务、具备非阻塞的并发控制机制和 share nothing 的分布式并行架构。

大数据时代,Stonebraker 也撰文探讨数据管理系统的未来走向[19]。首先,他建议研究者应建立统一的数据管理中心,因为这样会大大降低数据管理中的设计、建造和维护成本。他展示了自己对在大数据上进行简单的如 count、sum 等聚集操作的 SQL 分析的技术走向。他认为未来的技术需要在支持弹性可扩展的同时保证系统 100%线上可用。同时,他认为未来的数据库将全部采用列存储框架。

针对数据挖掘、机器学习等大数据上的复杂分析,他以探讨股票的涨跌相关性为例,认为这些回归、统计操作可以视为矩阵上的乘法、转置等线性代数上的运算。进一步分析了目前存在 3 种技术路线:

- 现有统计软件，它们仅支持线性代数操作，不能扩展到多节点，不具备处理超出内存大小的数据集等数据管理功能；
- 数据库，它们虽然支持数据管理功能，但使用数据库模拟矩阵运算需要进行连接后的分组操作，对稠密矩阵的 I/O 和 CPU 代价都过高；
- 现有数据库厂商提供的用户自定义函数，用它们实现线性代数操作不具备操作符内部的并行实现。

因此，他认为应该使用 SciDB 等数组数据库。当然，这要求用户学习和 SQL 稍有不同的语言。

最后，他提醒大家应该从 1990 年起建造数据仓库的经验中吸取教训，要充分意识到通过数据收集、模式匹配、实体识别、数据清洗等一系列过程建立一个可供所有分析者分析的数据池的复杂性[22]。

在持续开发数据库的系统架构过程中，Stonebraker 培养了一大批技术专家，Daniel Abadi（Hadapt 联合创始人）、Michael J.Carey（UC Irvine 教授，美国工程院院士，ACM Fellow）、Robert Epstein（Sybase 创始人）、Diane Greene（VMWare 创始人）、Paula Hawthorn（Britton-Lee 创始人，曾任 Informix 研发副总）、Marti Hearst（UC Berkeley 教授，ACM 会士）、Gerald Held（曾任 Oracle 研发副总）、Joseph M.Hellerstein（UC Berkeley 教授，ACM 会士）、AnantJhingran（IBM 信息管理部的副总兼 CTO）、Mike Olson（曾任 Sleepycat 和 Cloudera 的 CEO）、Margo Seltzer（哈佛大学教授，BerkeleyDB 的作者）、Dale Skeen（Tibco 副总，Vitria 创始人），等等。

Michael Stonebraker 在 40 余年的科研和系统开发生涯中不断重复“大学科研项目→研发原型系统→创办公司进行成果转化→公司被收购成果商业化”的传奇。他提出了大量的数据库新概念，开发了许多数据库原型和产品，成立了许多数据库公司，也培养了一大批数据库领域的技术专家。他获得 2014 年 ACM 图灵奖的确实至名归。

附录 B 数据库基准测试 TPC-C 和 TPC-H

数据库基准(Benchmark)测试是针对 DBMS 的性能测试。一个大型通用的 DBMS 软件必须经过严格的测试,包括功能测试、SQL 标准符合性测试、性能测试、稳定性测试、极限测试和综合应用测试等。其中性能测试的一个重要内容是数据库基准测试。

什么是数据库基准测试呢?基准测试是通过运行标准的评测程序获得某个 DBMS 软件执行预定任务的性能特征。性能评测程序也称为性能基准程序。目前最主要的是 TPC 数据库基准测试。

B.1 数据库基准的发展历史

在近 30 年中,涌现出不少数据库测试基准。下面介绍曾经使用过的几个比较著名的数据库基准。

1. Wisconsin 基准

第一个被广泛使用的测试关系数据库系统性能的基准是 Wisconsin(威斯康星)基准。威斯康星大学的 DeWitt 和 Carolyn 等用 SQL 语句描述了基准使用的查询。

Wisconsin 基准要点包括:

① 定义了 4 个关系,大小固定。

② 提供测试基本关系操作性能的 32 个 SQL 语句。

③ 把执行时间作为性能标准。

④ 易于理解。

⑤ 没有定义更新操作,而且是单用户基准,不测试并发和恢复。

2. AS^{3}AP 基准

Dina Bitton、Cyril Orji 和 Carolyn Turbyfill 后来提出了 AS^{3}AP 基准,这是一个更加完整的关系数据库基准测试。

AS^{3}AP Wisconsin 基准要点包括:

① 5 个关系表。所有关系通过生成文件装入数据。

② 设定了时间限制,然后测量在限定时间内系统可以处理的最大数据库规模。

③ 增加了批处理和交互查询的混合测试,测试分单用户和多用户两部分。

④ 定义了更加复杂的衡量尺度。

3. Set Query 基准

威斯康星和 AS^{3}AP 定义的查询针对的是简单关系查询,O'Neil 指出决策支持应用需要更复杂、更有效的测试基准。所以,他提出了一个大小可变的数据库来测试决策支持系统的性能。Set Query 基准要点包括:

① 测试信息系统和决策支持系统,主要针对查询,特别是集合查询。

② 增加了集合查询,共执行 69 个查询功能语句。

③ 衡量测试结果的尺度是性能价格比。

④ 缺乏对多用户环境的测试。

Set Query 有些事务响应时间很长,选择一个可接受的标准值作为各个系统比较的依据有点困难。

另外还有一些营利性商业基准组织,较有名的两个是 Neal Nelson Associates 和 AIM Technology。这些组织拥有各自的基准,可以为开发者和客户提供有价值的性能测量建议。

B.2 TPC 简介

对于不同的数据库测试基准,测试基准开发者、计算机硬件厂商和数据库厂商都有各种争议。所以 1988 年 8 月 10 日,34 家软硬件开发者创立了事务处理性能委员会(Transaction Processing Performance Council ,TPC)。TPC 是一个非盈利性组织,目的是定义事务处理和数据库系统领域的基准,定义计算系统性能和报告性能结果的方法,向业界发布客观的、经过证实的 TPC 性能数据。

1. 一些重要的 TPC 测试基准

TPC 委员会设计了不少性能测试基准,其中包括 TPC-A、TPC-B、TPC-C、TPC-D、TPC-E、TPC-H、TPC-R、TPC-W、TPC-APP 等。下面简要介绍这些测试基准:

① TPC-A。这是 TPC 委员会于 1989 年 11 月公布的第一个测试标准。

② TPC-B。是面向事务批处理的测试程序。由于 TPC-A 模拟了端到端的用户环境,受到了服务器厂商和数据库厂商的反对,他们认为批处理模式更能代表产品的性能。所以 TPC 在 1990 年 8 月公布了 TPC-B,放弃了网络及用户延迟部分,而采用了批处理方式,但测试模型仍然采用了 TPC-A 的银行交易。

无论是 TPC-A 还是 TPC-B,都为科学地测试 DBMS 的事务处理性能提供了方法和技术,但是它们设计的应用背景比较简单,只强调数据的修改操作,不能充分体现应用领域数据操作的特点,也不能全面测试事务处理系统性能。

③ TPC-C。这是最成功的目前广泛使用的数据库基准测试。TPC-A、TPC-B 已经失去应用价值而被废弃了。B.3 将详细介绍 TPC-C 的相关内容。

④ TPC-D。1994 年 4 月,TPC 公布了测试标准 TPC-D,主要用来测试决策支持系统(DSS)。进行 TPC-D 测试要比 TPC-C 复杂,而且测试费用也高得多。

⑤ 后来 TPC-D 分成了两个基准测试 TPC-H 和 TPC-R,TPC-D 随即被废弃了。TPC-H 主要针对随机复杂查询的决策支持,TPC-R 针对商务报表的决策支持。

⑥ TPC-W。这是针对 Web E-Commerce 应用的基准。于 2000 年 2 月公布。2005 年被废弃。

⑦ TPC-APP。用于测试 Web 电子商务应用系统的性能基准,于 2005 年 8 月公布。

为了使基准更加公正有效,TPC 仍然在不断更新各种基准的版本,开发新的基准。

综上所述,TPC-A、TPC-B、TPC-D、TPC-W 已经被废弃。目前常用的是 TPC-C、TPC-H、TPC-R 和 TPC-APP。

TPC-C 是针对联机事务处理(Online Transaction Processing,OLTP)领域的基准。而 TPC-H、TPC-R 是针对决策支持系统(Decision Support System,DSS)领域的基准。由于它们针对的应用环境不同,所以它们之间存在许多差异。

2. TPC 测试的意义

进行 TPC 测试给最终用户和生产厂商都能带来许多好处,例如,

- 提供了比较不同系统性能差异的客观方法;
- 提供了比较不同系统性能价格比的客观方法;
- 为客户提供了整套系统而非处理器等单个部件性能的评价方法;
- 厂商通过评测改进产品,从而为客户提供了性价比更高的产品。

不论是计算机硬件厂商还是软件厂商,都十分重视 TPC 测试,不断努力提高自己的 TPC 测试结果。

任何一个基准都不可能衡量所有应用情况下计算机系统的性能。不同领域的系统特征有很大的差异,所以应该采用不同的基准。尽管这样,不同领域的基准测试仍应该满足一些共同的标准,Jim Gray 在 1993 年出版的《数据库和事务处理性能手册》中给出了这些共同的标准。

- 相关性:必须测量在执行该领域内的典型操作时,系统性能和价格/性能的峰值。
- 轻便性:便于在许多不同的系统和结构中实现。
- 规模灵活:既可应用于小的计算机系统,也可应用于大的计算机系统。
- 简单性:易于理解。

B.3　TPC-C

TPC 委员会设计的基准中最著名的是 TPC-C。TPC-C 已经成为 OLTP 性能测试的工业标准。

1. TPC-C 的应用环境

TPC-C 模拟了一个完整的处理订单的应用环境。

TPC-C 定义的应用环境是一个虚拟的应用,模拟批发商向客户供货的场景,包括批发商、仓库、货物、销售地区、客户 5 个实体。这些批发商具有地理分散的销售地区及相应的仓库,有众多的用户。TPC-C 模拟了 5 种不同事务,包括下订单、根据订单交付货物、记录用户支付订单的情况、检查订单处理的情况和监督仓库的库存水平;而且包括了联机执行事务和对延迟执行的事务进行排队的混合情况。

TPC-C 定义的数据库包括了 9 张表,表中的记录取值范围较广,进行 TPC-C 测试时可以灵活地选择数据库的大小。

虽然 TPC-C 基准描述的是一个批发供应商的活动,但是并没有局限于任何一种特定商业领域,而是代表了普遍的商业环境特点,尤其是管理、销售和分销商业环境。

TPC-C 详细定义了各种事务的执行比例、数据的输入时间和返回结果后的思考时间、写磁盘间隔、检查点间隔等。

TPC-C 具有如下特点:

- 定义完整,从表模式到测试结果的提交方式等各个方面都有完整的定义;
- 设计科学,宏观设计和技术细节都有理论依据和实践背景;
- 结构复杂,包括被测试系统、测试驱动系统以及网络连接系统;
- 代价昂贵,进行 TPC-C 测试需要大量的硬件和软件的支持,将会耗费大量的人力资源。

2. TPC-C 商务模型

TPC-C 的商务模型如图 4 所示,描述了供货商(也就是批发商)公司(Company)、公司所管理的仓库(Warehouse)、销售地区(District)以及顾客(Customer)的分层结构。一个供货商经营一定数目的仓库,每个仓库为 10 个地区供货,每个地区有 3 000 名顾客。另外,每个仓库存储了公司的 100 000 种商品,用来满足客户订单的需要。

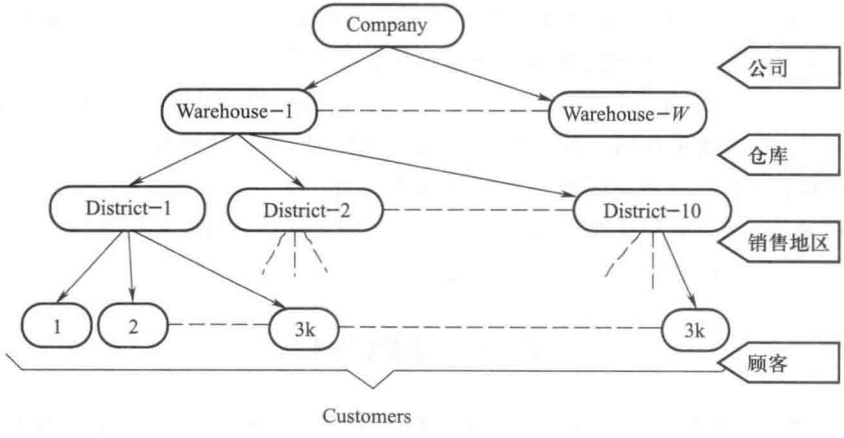


图 4 TPC-C 商务模型

该商务模型所模拟应用是可扩展的。如果公司业务扩展,就加入一个或几个新的仓库和相应的销售地区。而每个仓库覆盖的销售地区仍为 10 个,每个地区的顾客仍为 3 000 个。仓库存储的商品数也不变,仍为 100 000 个。

3. TPC-C 的逻辑数据库模式

TPC-C 的逻辑数据库模式如图 5 所示。椭圆框表示一个数据库表,其中的数字表示表的行数,即表的元组数。这些数字都包含了一个因子 W(仓库的数目),描述数据库缩放比例。Warehouse 表示规模的基本单位,其他表的基数是其基数的函数。例如:W = 2,表示该公司有 2 个仓库,2×10 个销售地区,2×10×3 000 个顾客。

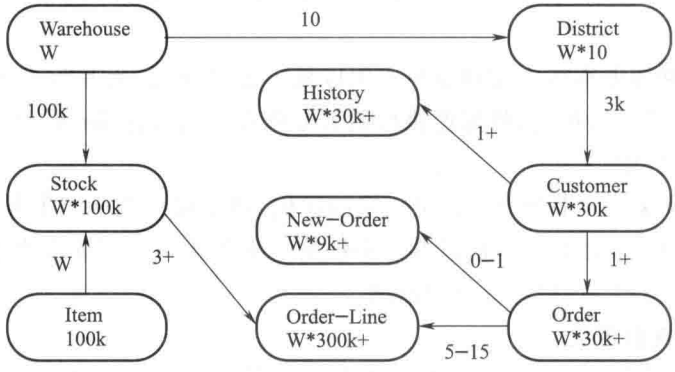


图 5　TPC-C 逻辑数据库模式

箭头表示基本表之间的联系,箭头旁边的数字表示联系的基数,即每个父结点平均的子结点数目)。加号“+”表示该数字会随行数的增加或删除而发生变化。

图 4 所对应的 9 张数据库表为:仓库(Warehouse)、库存(Stock)、货物(Item)、地区(District)、顾客(Customer)、新订单(New-Order)、库存订单表(Order-Line)、顾客订单表(Order)和历史纪录(History)。

4. TPC-C 的 5 种事务

TPC-C 是针对联机事务处理的。TPC-C 对以上的数据库表设计了 5 种商业事务,定义了处理每种事务的具体步骤。

① 建立新订单事务(new order)。该事务的功能是模拟客户建立货物订单行为,在数据库中建立一张完整的订单表。新订单事务是一个典型的执行频率高、具有严格响应时间要求的读写事务,并且还通过模拟用户的输入错误来引起事务的回滚。

② 支付事务(payment)。该事务的功能是模拟客户付款行为,修改客户、地区和仓库的账目和销售额。支付事务对客户的收支平衡进行更新,以反映该地区的支付情况和仓库的销售统计。该事务执行频率高,而且响应时间的要求也很严格。

③ 查询订单状态事务(order status)。该事务的功能是模拟客户察看当前自己的订单处理的情况。客户在相应的表格里查询相关的信息。系统建立的是一个只读事务,具有较低的执行频率,对于响应时间的要求也不是很严格。

④ 交付事务(delivery)。该事务的功能是模拟公司的发货行为,一次完成 10 张订单的发货。在一个读写事务内,对每一个订单进行处理(交付)。在同一个事务内作为一组(或一批)交付的订单数目是由具体实现确定的。该事务包括了数据库的读写操作,执行频率较低,是一个批处理事务,响应时间的限制不太苛刻。

⑤ 查询库存水平事务(stock level)。该事务的功能是查询库存当前状况,查找那些库存量低于规定阈值的货物,了解库存水平。这是一个只读事务,执行频率低,对于响应时间和一致性要求不是很高。

注意,以上 5 种商业事务中交付事务必须以延迟方式执行,而其他事务都是以交互方式执行。延迟执行方式的特征是,将延迟执行的事务放入队列,把控制转交给客户端,并把执行信息记录在结果文件中。

TPC-C 还要求这 5 种事务应该满足一定的比例,即在提交的所有事务请求中,新订单事务 45%、支付事务 43%、查询订单状态事务 4%、交付事务 4%、查询库存水平事务 4%。

需要强调的是,TPC-C 只有多用户测试。

5. TPC-C 的性能尺度

TPC-C 定义了度量性能和性能价格比的尺度,评价指标主要有两个:

① tpmC(transactions per minute 的简称,C 指 TPC 中的 C 基准程序)。表示系统最大处理能力或者最大吞吐量,即每分钟系统处理新订单个数。需要强调的是,TPC-C 测试时 5 种业务按照规定的比例同时发生,并且对每种业务都有响应时间的限制,tpmC 计算的只是处理新订单业务的有效数量,而不是所有业务之和,它代表的是一个峰值,即最大处理新订单数。

② $tpmC。表示处理一笔新订单业务所需的费用,即系统的性能价格比。在计算系统费用时,包括所有的软硬件费用,实际上是用户运行环境的总投资,单位是美元 $,而性能价格比则定义为总价格÷性能,单位是 $/tpmC。

就数据库基准测试来说,只考虑性能是不够的,还要考虑系统的性能价格比。因为对于用户来说,在满足性能要求的前提下,应该选择价格最低的系统。

还有一个重要指标是系统可用日期。系统可用日期指的是被测试系统中使用的所有硬件和软件可以在市场上购买到的最近日期。TPC 规定,系统可用日期不得比提交测试结果的日期晚 6 个月以上。

6. TPC-C 事务发生器 Drive 及其执行模式

Driver(external driver system)是一个 TPC-C 事务发生器,它模拟该商务模型里的各个地区的终端,以及每个使用这些终端的用户操作,图 6 描述了 TPC-C 事务的流程。

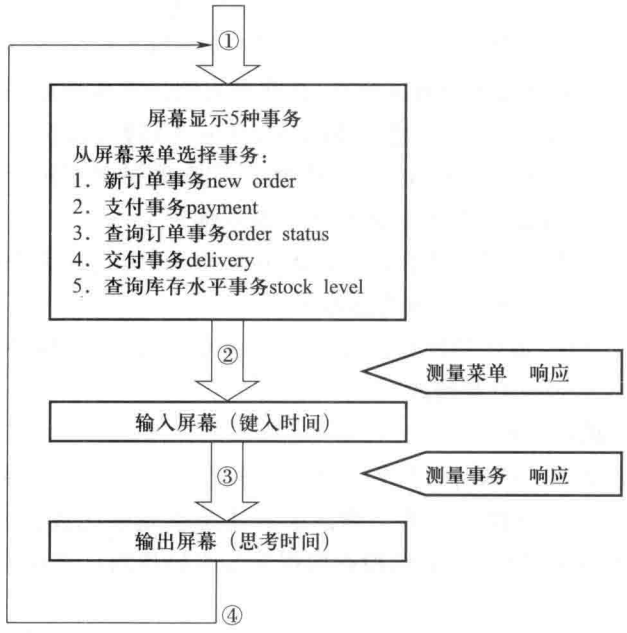


图 6　TPC-C 事务流程图

事务的流程通常如下所示：

① 首先屏幕显示菜单，上面显示 5 种事务，等待用户从中选择一种事务。

② 用户选择了一种事务之后，屏幕显示出该事务的输入屏幕。TPC-C 要求测量菜单响应时间，即用户键入选择之后到出现输入屏幕之间的时间间隔。

③ 用户输入数据，提交请求，被测试的系统处理该事务，然后显示事务的输出信息。TPC-C 要求测量事务响应时间，即用户提交请求之后到 ACID 显示事务输出结果之间的时间间隔。

④ 用户阅读处理结果，返回第 1 步执行下一个事务。

TPC-C 比较严格地模拟现实应用的情况，要求客户端有键入时间（keying time）和思考时间（think time），并对键入时间和思考时间有明确的要求。键入时间是模仿用户把数据录入到系统中所花费的时间。思考时间是模仿用户阅读处理结果所花费的时间。

最后，TPC-C 进行相关信息统计和计算，给出 tpmC 和 $tpmC 等测试结果。

TPC-C 没有定义明确的 SQL 语句，但是对 5 种事务进行了详细的定义。其中包括每种事务的输入数据应该满足的条件，并用自然语言详细描述了每种事务需要进行的各项操作以及这些操作的执行顺序，而且定义了每种事务的终端 I/O 需要满足的规格。

7. ACID 测试

ACID 特性指的是数据库事务必须具有的 4 个特性:原子性(Atomicity)、一致性(Consistency)、隔离性(Isolation)和持续性(Durability)。这 4 个特性简称为 ACID 特性。

TPC-C 是针对联机事务处理的测试基准,在获得被测试系统性能结果之外,必须测试被测试系统在执行事务的过程中是否满足事务的 ACID 基本特性,以保证测试结果的正确性。因此 TPC-C 测试中也制定了一系列 ACID 特性测试,主要测试内容包括:

(1) TPC-C 选择 payment 支付事务进行原子性测试

① 随机选择一个仓库、地区和顾客,运行 payment 事务,提交该事务后,检查修改后的仓库表、地区表和顾客表是否正确一致地被修改了。

② 随机选择一个仓库、地区和顾客,运行 payment 事务,回滚该事务后,检查修改后的仓库表、地区表和顾客表是否没有被修改。

(2) TPC-C 的一致性检查

针对 TPC-C 问题背景,定义了 12 个一致性测试条件。只要数据库满足这些一致性条件,就认为数据库是处于一致性状态的。测试的方法是首先确认数据库满足所有一致性条件,然后对数据库进行规定的操作,最后检查数据库是否仍然满足那些一致性条件。

(3) 隔离性测试

TPC-C 针对 4 类数据不一致性,定义了 4 种隔离级别,如表 16。

这 4 类数据不一致性是:

- P0 为 Dirty Write,丢失修改;
- P1 为 Dirty Read,读"脏"数据;
- P2 为 Non-repeatable Read,不可重复读;
- P3 为 Phantom,幻影。

4 种隔离级别是:

- 隔离级别 L0 为可以保证不丢失修改;
- 隔离级别 L1 为可以保证不丢失修改、还可以进一步防止读"脏"数据;
- 隔离级别 L2 为在隔离级别 L1 的基础上,还可以保证可重复读;
- 隔离级别 L3 为在隔离级别 L2 的基础上,还可以保证无幻影。

表 16 TPC-C 定义的 4 种隔离级别

隔离级别	P0:Dirty Write	P1:Dirty Read	P2:Non-repeatable read	P3:Phantom
0	Not Possible	Possible	Possible	Possible
1	Not Possible	Not Possible	Possible	Possible
2	Not Possible	Not Possible	Not Possible	Possible
3	Not Possible	Not Possible	Not Possible	Not Possible

TPC-C 使用 new order、payment、order status 和 delivery 事务的组合，进行多个用户并发执行、提交和回滚操作。测试在资源竞争时是否等待，以及其执行结果是否与顺序执行结果相同。

TPC-C 定义了 9 种隔离性测试。例如，其中一个测试是：并发执行 delivery 事务和 payment 事务，当 delivery 事务回滚的时候，delivery 事务和 payment 事务之间是否存在 write-write 冲突。测试步骤是：

① 开始一个 delivery 事务 T1；

② T1 在接近 commit 前停止；

③ 对于同一个顾客开始一个新的 payment 事务 T2；

④ 证明事务 T2 等待；

⑤ 回滚 T1，完成 T2；

⑥ 检查数据库表中的有关字段（如账户余额等）只被事务 T2 修改。

TPC-C 在测试说明书中声明：基准规定的隔离性测试是针对采用封锁机制的数据库系统的，如果采用了其他的并发控制技术，就要设计其他隔离性测试方法，但是必须公布相应的并发控制技术和测试方法。

（4）持续性测试

持续性也称永久性（permanence）。指一个事务一旦提交，它对数据库中数据的改变就应该是永久性的，接下来的其他操作或故障不应该对其执行结果有任何影响。

TPC-C 测试三种情况下的永久性。

- 存储介质的失效；
- 系统运行中的崩溃（system crash），需要系统重启；
- 内存数据的部分丢失。

TPC-C 的测试过程如下：

① 运行 TPC-C 事务；

② 人为设置以上三种情况下的故障，如掉电、操作系统重新启动等；

③ 进行系统恢复；

④ 针对成功和失败的事务，验证数据库中的数据是否正确，一致性是否满足，监测已提交事务所产生的结果是否保存，从而监测事务的永久性；

⑤ 转①。

B.4　TPC-H

TPC Benchmark H（TPC-H）是一个决策支持的测试基准，由一系列面向商务应用的查询和并发的数据修改组成，其选择的查询和组成数据库的数据在商业上都具有广泛的代表性并

且易于实现。该基准描述了决策支持系统的三个方面:分析大量的数据,执行高复杂度的查询,回答关键的商业问题。

1. TPC-H 的应用环境

TPC-C 基准模拟了商用环境的操作端,在操作端事务以实时方式执行,而 TPC-H 测试则模拟了商用环境的分析端,在分析端计算未来趋势、产生精炼的分析结果(如图 7 所示)。

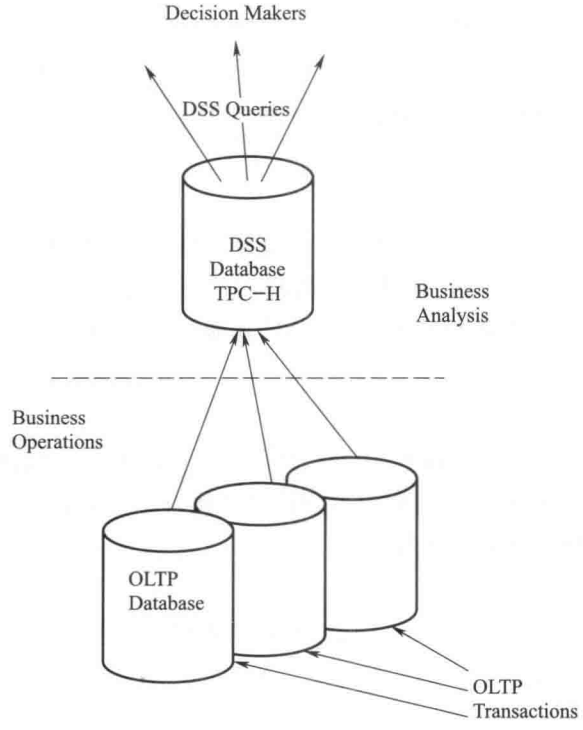


图 7 TPC-C 与 TPC-H 所对应的商用环境示意图

TPC-H 由一系列代表复杂商业分析应用的查询组成,这些查询给出了一个实际的应用环境,描绘了批发商活动和客户订购活动。TPC-H 不代表任何特定商业领域里的活动,但可以被应用到全球范围内任何需要管理或销售某种商品的行业(比如汽车租赁、食品销售、供应商等)中。

TPC-H 以测试系统利用率和操作复杂度为目的。因此,被选择的查询具有以下特征:非常复杂、选择各种各样的访问模式、带有随机特性、检查大部分可以获得的数据、互不相同、含有不同的查询参数且参数在执行时会变化。所选择的查询要为下面各类商业分析提供答案:定价和促销、供货和需求管理、利润和收入管理、顾客满意度研究、市场份额研究和配送管理。

虽然 TPC-H 的重点是信息分析，但该测试也要求系统满足数据库定期更新的需要，必须能够全天候支持对所有表的查询和更新。

2. TPC-H 商务模型

TPC-H 的商务模型如图 7 所示，描述了地区和国家、零件供应商和顾客、供应和销售的分层结构。一个供应商供应一定数量的零件，一类零件也有一定数量的供应商，一个顾客下一定数量的订单，每个订单可能具有若干个订单明细记录。地区和国家的数目是固定的，零件、供应商和顾客的数量是可伸缩的，也是相对稳定的，而供应和订单记录是动态的，也是记录数最多的。

图 8 所对应的 8 张数据库表为：零件（Part）、供应商（Supplier）、零件供应记录（PartSupp）、顾客（Customer）、订单（Orders）、订单明细记录（Lineitem）、国家（Nation）和地区（Region）。

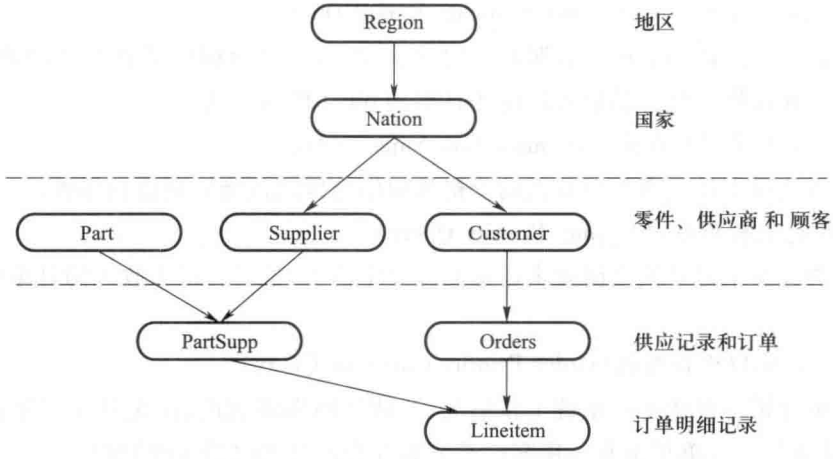


图 8　TPC-H 商务模型

3. TPC-H 的逻辑数据库模式

TPC-H 的逻辑数据库模式如图 9 所示。椭圆框表示一个数据库表，其中的数字表示表的基本行数，即表的基本元组数，SF（Scale Factor）为规模因子，描述数据库缩放比例。例如，SF 等于 2，则表示 Part 表有 2×200 000 个零件，Partsupp 表有 2×800 000 个零件供应记录，依次类推。箭头表示基本表之间的联系。

4. TPC-H 的 22 种查询

针对决策支持系统，TPC-H 设计了 22 种商业分析查询，定义了处理每种事务的具体步骤。下面概述每个查询的名称、含义和作用，如果读者对查询对应的 SQL 语句感兴趣，请参阅 TPC-H 标准规范。

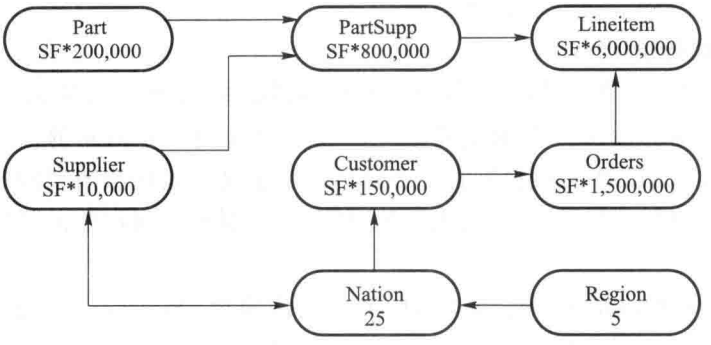


图 9 TPC-H 逻辑数据库模式

Q1. 价格统计报告查询(Pricing Summary Report Query)

该查询汇总和统计指定运送日期的已付款、已配送和已返回的所有零件的业务数量,该指定日期应该在数据库包含的最大的配送日期的 60-120 天以内。

Q2. 最小成本供应商查询(Minimum Cost Supplier Query)

该查询查找哪个供应商能以最低的价格供应给定区域内指定规格的零件。

Q3. 运送优先权查询(Shipping Priority Query)

该查询查找在指定日期之前尚未运送的订单中前 10 位具有最大收入的订单的优先权和潜在的收入。

Q4. 订单优先权检查查询(Order Priority Checking Query)

该查询可分析订单优先权系统工作情况,并估计顾客满意度,即统计指定年份和季度的订单数量,其中每个订单至少有一项零件是顾客在合同日期之后才收到的。

Q5. 当地供应商数量查询(Local Supplier Volume Query)

该查询统计指定国家和地区通过本地供应商和本地顾客获得的收入情况。该查询的统计数据可以为是否需要在指定国家和地区建立一个本地配送中心提供决策支持。

Q6. 预测收入变化查询(Forecasting Revenue Change Query)

该查询统计指定年份如果取消某些公司的折扣后收入增加的情况。这是一类“what if”查询,可以被用来寻找增加收入的途径。

Q7. 货运量查询(Volume Shipping Query)

该查询统计在两国之间的商品货运量,以帮助重新谈判货运合同,即查询在给定年份间,零件从一国供应商被配送给另一国的顾客,两国货运量总的收入情况。

Q8. 国家市场份额查询(National Market Share Query)

该查询统计在过去的两年中给定零件类型在指定国家和地区的市场份额变化情况。

Q9. 产品利润统计查询(Product Type Profit Measure Query)

该查询按国家和年份分组统计指定零件的盈利情况。

Q10. 退货报告查询(Returned Item Reporting Query)

该查询查找给定时间内有退货的顾客情况。

Q11. 重要库存识别查询(Important Stock Identification Query)

该查询查找给定国家的供应商库存中的重要零件,即那些库存价值占总库存价值比例比较大的零件。

Q12. 配送模式和订单优先级查询(Shipping Modes and Order Priority Query)

该查询分析和确定选择更便宜的配送模式是否会造成更多零件在合同日期之后被收到,从而对关键订单产生负面影响。

Q13. 顾客分布查询(Customer Distribution Query)

该查询查找顾客和订单之间的关系,即顾客按订单数量的分布情况,包括没有订单的顾客数,有一个订单的顾客数,有两个订单的顾客数,等等。

Q14. 促销效果查询(Promotion Effect Query)

该查询监视 TV 广告或者特别的促销活动带来的市场反应,即查询某一特定时间的收入中有多大的百分比是来自促销零件。

Q15. 顶级供应商查询(Top Supplier Query)

该查询确定顶级供应商以便给予奖励、更多订单,或是给予特别认证。所谓顶级供应商是在一个季度或一年内为总收入贡献最多的供应商。

Q16. 零件/供应商关系查询(Parts/Supplier Relationship Query)

该查询按零件品牌、类型和大小分组统计供应商数量。该查询可用来确定对于大量订购的零件是否有充足的供应商。

Q17. 小订单收入查询(Small-Quantity-Order Revenue Query)

该查询统计如果取消某些零件的小订单将导致平均年收入损失的情况。所谓小订单是指订购零件数比较少的订单。例如,首先确定给定品牌和包装类型的零件在过去 7 年中的每个订单的平均订购数量。如果某类零件的订购数量小于平均订购数量 20%的订单不再被接纳,统计平均年收入将会减少多少。

Q18. 大订单顾客查询(Large Volume Customer Query)

该查询按大订单分组统计顾客已订购的零件数,并根据订单总价排序顾客。所谓大订单是指其订购的零件数超过某个给定的阈值。

Q19. 折扣收入查询(Discounted Revenue Query)

该查询汇总统计订购指定品牌、包装和大小的零件的所有订单的折扣收入情况。

Q20. 潜在零件促销查询(Potential Part Promotion Query)

该查询查找有潜在零件促销机会的供应商,即那些拥有过剩零件的供应商。所谓过剩零

件是指其库存数量超过指定国家在指定年份运送总量的 50%。

Q21. 不能按时交货的供应商查询(Suppliers Who Kept Orders Waiting Query)

该查询查找在具有多个供应商的订单中,那些不能按照合同日期发货的供应商。

Q22. 全球销售机会查询(Global Sales Opportunity Query)

该查询确定可能购买零件的顾客的地理分布,即按国家分组统计近年来没有下过订单但还有账户余额的顾客情况。

5. TPC-H 的性能尺度

为了测量使用 TPC-H 基准的系统性能,测试者将执行:

① 功率测试(power test),测试单用户连接情况下系统执行用户查询的能力。按照如下公式计算当前数据库规模下的 TPC-H 查询处理能力:

$$\text{TPC - H Power@ Size} = \frac{3600 * SF}{\sqrt[24]{\prod_{i=1}^{i=22} QI(i,0) * \prod_{j=1}^{j=2} RI(j,0)}}$$

其中:

$QI(i,0)$ 为查询 Qi 执行时间(Timing Interval),时间单位为秒;

$RI(j,0)$ 为更新函数(Refresh Function)RFj 的执行时间,时间单位为秒;

Size 为当前数据库大小,SF(Scale Factor)为相应的规模因子;

说明:功率测试结果数是基于每小时的查询率(即乘以 3 600)。

② 吞吐量测试(throughput test),用来测量系统在最短时间内处理最多查询的能力,测试系统在多用户负载下系统的处理性能。按照如下公式来计算当前数据库规模下的 TPC-H 吞吐量:

$$\text{TPC-H Throughput@ Size} = (S * 22 * 3600)/T_S * SF$$

其中:

Ts 为测量间隔(从第一个查询流的第一个查询开始提交算起,到最后一个查询流的最后一个查询执行完输出所有结果为止的时间间隔);

Size 为当前数据库大小,SF 为相关的规模因子;

S 为查询流数量,其取值参见表 17。

表 17 吞吐量测试中规模因子和要求的最少查询流数

SF	*S*(Streams)
1	2
10	3
30	4

续表

SF	*S*(Streams)
100	5
300	6
1 000	7
3 000	8
10 000	9
30 000	10
100 000	11

TPC-H 定义三个主要度量：

① 性能度量(the performance metric)：合成的每小时查询性能(TPC-H composite Query-per-hour performance metric)，记为 QphH@ Size，计算公式如下：

$$\text{QphH@ Size} = \sqrt{\text{Power@ Size} * \text{Throughput@ Size}}$$

② 性价比度量(the price/performance metric)：每小时查询性能的价格(price-per-QphH@ Size)，记为 $/QphH@ Size，计算公式如下：

$$\text{TPC-H price-per-QphH@ Size} = \$/\text{QphH@ Size}$$

其中：

$ 为当前整个系统的价格；

QphH@ Size 为合成的每小时查询性能度量；

Size 为当前数据库大小。

③ 系统可获得日期(the availability date of the system)：一个待测系统(system under test)所有组件(components)都可以订购或者配送给客户的日期。

6. ACID 测试

在对基准的计时部分进行测试时，待测系统必须支持事务处理系统的 ACID 特性。由于 TPC-H 不是一个事务处理基准，没有包含任何 OLTP 事务，因此对 ACID 特性的评估需要在计时部分的测试之外进行，也要用一个特定的 ACID 事务来进行 ACID 测试。另外，在只读查询并发执行时，为了简化 ACID 特性是否有效的证明过程，TPC-H 定义了一个特定的 ACID 查询。

ACID 事务和 ACID 查询都使用一个事务函数以保证算法的正确性和结果的一致性。定义函数 trunc(n,p) 如下：

$$\mathrm{Trunc}(n,p)=\lfloor n*10^{p}\rfloor/10^{p}$$

其功能是截去数 n 到其第 p 位十进制数(例如:trunc(1.357,2)=1.35)。

ACID 事务的实现必须符合下列事务特征。

给定输入数据集合(O_KEY,L_KEY,[delta]),各项含义如下:

- 根据 Lineitem 表中的 L_ORDERKEY 分布情况随机选择 O_KEY;
- L_KEY 是从[1..M]中随机选择的,其中,

  M=SELECT MAX(L_LINENUMBER)FROM LINEITEM WHERE L_ORDERKEY=O_KEY;
- [delta]从[1.. 100]中随机选择。

```
BEGIN TRANSACTION
    Read O_TOTALPRICE from ORDERS into [ototal] where O_ORDERKEY=[o_KEY]
    Read
L_QUANTITY,L_EXTENDEDPRICE,L_PARTKEY,L_SUPPKEY,L_TAX,L_DISCOUNT
    into [quantity],[extprice],[pkey],[skey],[tax],[disc]
    where L_ORDERKEY=[o_key] and L_LINENUMBER=[l_key]
    Set [ototal]=[ototal]-
    trunc(trunc([extprice]*(1-[disc]),2)*(1+[tax]),2)
    Set [rprice]=trunc([extprice]/[quantity],2)
    Set [cost]=trunc([rprice]* [delta],2)
    Set [new_extprice]=[extprice]+[cost]
    Set [new_ototal]=trunc([new_extprice]*(1.0-[disc]),2)
    Set [new_ototal]=trunc([new_ototal]*(1.0+[tax]),2)
    Set [new_ototal]=[ototal]+[new_ototal]
    Update LINEITEM
    where L_ORDERKEY=[o_key] and L_LINENUMBER=[l_key]
    Set L_EXTENDEDPRICE=[new_extprice]
    Set L_QUANTITY=[quantity]+[delta]
    Write L_EXTENDEDPRICE,L_QUANTITY to LINEITEM
    Update ORDERS where O_ORDERKEY=[o_key]
    Set O_TOTALPRICE=[new_ototal]
    Write O_TOTALPRICE to ORDERS
    Insert Into HISTORY
    Values([pkey],[skey],[o_key],[l_key],[delta],[current_date_time])
COMMIT TRANSACTION
Return [rprice],[quantity],[tax],[disc],[extprice],[ototal] to driver
```

其中,HISTORY 表(其表结构如表 18 所示)只在 ACID 特性测试时需要;ACID 事务返回的值是旧值,也就是更新以前的值。

表 18　HISTORY 表结构

列名	数据类型要求	
H_P_KEY	identifier	外键引用 P_PARTKEY
H_S_KEY	identifier	外键引用 S_SUPPKEY
H_O_KEY	identifier	外键引用 O_ORDERKEY
H_L_KEY	integer	
H_DELTA	integer	
H_DATE_T	date and time to second	

ACID 查询必须依照下面的查询功能定义来实现。

提供输入数据：

根据 Lineitem 表的 L_ORDERKEY 的分布情况随机选择 O_KEY

```
SELECT SUM(trunc(
                trunc(L_EXTENDEDPRICE * (1-L_DISCOUNT),2) * (1+L_TAX),2))
                FROM LINEITEM
WHERE L_ORDERKEY=[o_key]
```

ACID 事务和 ACID 查询必须能用来证明一个待测试系统完全符合 ACID 特性。

尽管 ACID 事务和 ACID 查询不包括 TPC-H 数据库中所有的表，但是所有的表都必须支持 ACID 特性。

(1) 原子性测试

① 用随机输入数据来执行 ACID 事务、COMMIT 事务，然后检查 ORDERS、LINEITEM 和 HISTORY 表中适当的列被修改了。

② 用随机输入数据来执行 ACID 事务、ROLLBACK 事务，然后检查 ORDERS、LINEITEM 和 HISTORY 表中相应的列没有被修改。

(2) 一致性测试

TPC-H 数据库的一致性状态定义为所有通过(O_ORDERKEY=L_ORDERKEY)定义的 ORDERS 和 LINEITEM 都满足：

```
O_TOTALPRICE=SUM(trunc(trunc(L_EXTENDEDPRICE * (1-L_DISCOUNT),2) * (1+L_TAX),
        2))
```

为了验证 ORDERS 和 LINEITEM 表的一致性时，执行以下的步骤：

① 检查 ORDERS 和 LINEITEM 表在初始状态时是符合一致性定义的，其中必须具有随机的至少 10 个不同的 O_ORDERKEY 值对应的数据。

② 为每个查询和更新任务提交至少 100 个 ACID 事务用来产生测试报告，每个事务必须

随机地使用(O_KEY,L_KEY,DELTA)这些字段,确定所有上一步中 O_ORDERKEY 的值都会被这些事务所用到。

③ 再次检查第一步选定的 O_ORDERKEY 值对应的 ORDERS 和 LINEITEM 数据的一致性。

(3) 隔离性测试

定义下面这些术语:

T_1 = 一个 ACID 事务

T_2 = 一个 ACID 事务

T_3 = TPC-H 中第 1~第 22 的任何一个查询或者任何一个 ACID 查询

T_n = 任何事务,尽管是任何事务,T_n 也不会写"脏"数据

表 19 描述了 TPC-H 执行时必须满足的隔离级别需求。

表 19 TPC-H 执行时必须满足的隔离级别

序号	事务集合	现象	不能在下面的事务中出现	文字描述
1	$\{T_i,T_j\}$ $1<=i,j<=2$	P_0,P_1,P_2,P_3	T_i	两个事务之间满足第 3 级隔离级别
2	$\{T_i,T_n\}$ $1<=i<=2$	P_0,P_1,P_2	T_i	ACID 事务和其他事务满足第 2 级隔离级别
3	$\{T_i,T_3\}$ $1<=i<=n$	P_0,P_1	T_3	TPC-H 的 22 个查询与 ACID 事务及任何事务满足第 1 级隔离级别

其中,P_0 表示"写脏数据",P_1 表示"读脏数据",P_2 表示"不可重复读",P_3 表示"幻影"。

在常规的加锁方案中,隔离级测试可以采用下列方法来执行。实现其他隔离模式的系统需要执行其他的隔离级别测试方法。

下面 6 种测试是为了检查待测试系统是否支持需要的隔离级别,所有的隔离级测试都是通过随机选择一个数据集合(P_KEY,S_KEY,O_KEY,L_KEY,DELTA)来实现的。

① 隔离级测试 1

该测试是为了演示一个读写事务和一个只读事务的读写冲突,其中读写事务会提交。执行步骤如下:

a. 开始一个 ACID 事务 Txn1,随机选择 O_KEY、L_KEY 和 DELTA。

b. 在提交之前将事务 Txn1 挂起。

c. 开始一个 ACID 查询 Txn2,使用和第一步一样的 O_KEY(Txn2 将会试图读取 Txn1 修改过的元素)。

d. 检查 Txn2 没有读取到 Txn1 修改过的值。

e. Txn1 提交。

f. 此时 Txn2 应该完成了。

② 隔离级测试 2

该测试是为了演示一个读写事务和一个只读事务的读写冲突,其中读写事务会回滚。执行步骤如下:

a. 开始一个 ACID 事务 Txn1,随机选择 O_KEY、L_KEY 和 DELTA。

b. 在提交之前将事务 Txn1 挂起。

c. 开始一个 ACID 查询 Txn2,使用和第一步一样的 O_KEY(Txn2 将会试图读取 Txn1 修改过的元素)。

d. 检查 Txn2 没有读取到 Txn1 修改过的值。

e. Txn1 回滚。

f. 此时 Txn2 应该完成了。

③ 隔离级测试 3

该测试是为了演示两个更新事务中第一个事务提交时的写-写冲突,执行步骤如下:

a. 开始一个 ACID 事务 Txn1,随机选择 O_KEY、L_KEY 和 DELTA1。

b. 在即将提交之前将事务 Txn1 停止。

c. 开始另一个 ACID 事务 Txn2,使用和第一步一样的 O_KEY、L_KEY,另外随机选一个 DELTA2(Txn2 将会试图读取并修改 Txn1 修改过的元素)。

d. 检查 Txn2 处于等待中。

e. 让 Txn1 提交,Txn2 可以继续执行。

f. 检查 Txn2.L_EXTENDEDPRICE = Txn1.L_EXTENDEDPRICE + (DELTA2 * (Txn1.L_EXTENDEDPRICE / Txn1.L_QUANTITY))

④ 隔离级测试 4

该测试是为了演示两个更新事务中第一个事务回滚时的写-写冲突,执行步骤如下:

a. 开始一个 ACID 事务 Txn1,随机选择 O_KEY、L_KEY 和 DELTA1。

b. 在即将提交之前将事务 Txn1 停止。

c. 开始另一个 ACID 事务 Txn2,使用和第一步一样的 O_KEY、L_KEY,另外随机选一个 DELTA2(Txn2 将会试图读取并修改 Txn1 修改过的元素)。

d. 检查 Txn2 处于等待中。

e. 让 Txn1 回滚,Txn2 可以继续执行。

f. 检查 Txn2.L_EXTENDEDPRICE = Txn1.L_EXTENDEDPRICE

⑤ 隔离级测试 5

该测试是为了演示两个操作不同表的事务可以并发执行。执行步骤如下：

a. 开始一个 ACID 事务 Txn1，随机选择 O_KEY、L_KEY 和 DELTA1。

b. 在即将提交之前将事务 Txn1 停止。

c. 开始另一个 ACID 事务 Txn2，执行如下的操作。

d. 随机选择 PS_PARTKEY 和 PS_SUPPKEY 的值，返回 PARTSUPP 中与 PS_PARTKEY 和 PS_SUPPKEY 的值相等的所有元组。

e. 检查 Txn2 已经完成。

f. 允许 Txn1 继续执行完毕。检查 ORDERS、LINEITEM 和 HISTORY 表中相应的值已经更新了。

⑥ 隔离级测试 6

该测试是为了演示操作同一张表的任何一个只读事务的执行不会延迟在这些表上进行更新事务的执行。执行步骤如下：

a. 开始一个事务 Txn1，Txn1 对确认数据库执行 Q1 操作，它的[delta]值从[0..2159]中选择，便于这个查询能够执行足够长的时间。

说明：对[delta]选择值 0 将会使 Txn1 得到最大的执行时间。

b. 在 Txn1 完成之前，执行一个 ACID 事务 Txn2，随机选择 O_KEY、L_KEY 和 DELTA。

如果 Txn2 在 Txn1 之前结束，检查 ORDERS、LINEITEM 和 HISTORY 表中相应的值已经更新了。在这种情况下，这个测试只能完成步骤 a 和 b。如果 Txn2 在 Txn1 完成时没有结束，继续进行下面的步骤 c 和 d。

c. 确定 Txn1 仍然在执行，开始第三个事务 Txn3，它也对确认数据库执行 Q1 查询，但是它的[delta]值不同于第一步所开始的那个事务。

d. 检查 Txn2 在 Txn3 完成之前结束，并且 ORDERS、LINEITEM 和 HISTORY 表中相应的值已经更新了。

说明：有些情况下 Txn2 将会比 Txn1 提前结束，这时候不需要运行 Txn3 来检测是否符合隔离级需求。

(4) 持久性测试

提交特性的定义：当一个事务不但写入了对数据库中数据的更新，同时也对相应操作记录了日志时，我们说这个事务是已提交的。

说明 1：事务可以不向用户发出通知就提交，因为完整的消息机制不是 TPC-H 所必需的。

说明 2：尽管事务的操作执行顺序是无关紧要的，但是必须在事务的提交操作成功之后才能返回数据的实际值。

为了更好地进行持久性测试,在驱动器上必须有一个持久的记录文件,用来保存所有成功结束的事务的详细信息以及它们所返回的消息。当发生故障时,这个日志文件必须已经记录有所有已经提交的事务,除非事务的提交信息被故障所中断了。

持久日志文件仅在持久性测试中使用,它需要记录的内容如表 20 所示。

表 20　在持久性测试时持久日志文件记录的内容

字段	数据描述
P_KEY	与 P_PARTKEY 联系的外键
S_KEY	与 S_SUPPKEY 联系的外键
O_KEY	与 O_ORDERKEY 联系的外键
L_KEY	Integer
DELTA	Integer
DATE_T	时间

说明:如果驱动器也在 SUT 中,成功文件必须与 TPC-H 数据库隔离开来。例如,成功文件必须在 ACID 事务之外写入,而且如果成功文件的持久性也是由同样的 TPC-H 数据库的管理员来维护,则必须使用一个不同的日志文件。

测试负责人必须保证待测试系统从以下故障中恢复时可以保持数据库的已提交修改结果。

- TPC-H 数据库中任何一个存储表或者恢复日志的持久性媒体遭受到永久的无可挽救的故障,发生故障的媒体必须是随机的,而不能是预先设定好的。
- 瞬间的中断(系统崩溃/系统挂起)需要系统重启以恢复。
- 内存的全部或者部分故障(内容的丢失)。
- 待测试系统的电源故障:所有外部电源全部中断至一个不确定的时间。

持久性测试的目的是为了演示不管出现了上述哪种故障,已提交并且给驱动器发送了消息的事务所作的修改都会保存下来,恢复后的数据库同样也符合一致性条件。

当发生了上述提到的故障,执行下面的步骤:

a. 检查 ORDERS 和 LINEITEM 表初始时是符合一致性要求的,其中 O_ORDERKEY 的值是随机选取的 10 个不同的值。

b. 在每个执行流中异步提交至少 200 个 ACID 事务,每个事务随机选择(O_KEY,L_KEY,DELTA)的值。确定在第一步中所有的 O_ORDERKEY 值都会被这些事务所操作到。

c. 等到第二步中的至少 100 个 ACID 事务已经提交结束了,选择一种故障来实现。当故障发生时,必须符合以下条件:

- 至少一个事务正在运行中。

- 第二步中的整个任务中的所有 ACID 事务并没有完全执行完毕。

说明:这个目的是为了保证所有的执行流都正在不断地提交和执行事务时引发故障。如果在失败点时正在执行的事务数量少于执行流的数量,这样可以假设在提交一个事务和另一个事务的中间的非常小的间隔时间内随机中断了某些执行流。

d. 重新启动待测试系统,启用恢复过程。

e. 比较日志文件和 HISTORY 表的内容,判断日志中已提交事务和 HISTORY 中记录的修改是对应的,未提交事务的修改则没有出现在 HISTORY 中。计算日志文件中修改的数量和 HISTORY 中的修改数量是否相同。

说明:如果有不同的地方出现,只可能是因为事务已将修改写入到待测试系统中,但是还没将记录写入日志文件中,故障就出现了。

f. 重新检查 ORDERS 和 LINEITEM 表是否符合一致性。

参考文献

[1] 王珊，萨师煊. 数据库系统概论. 5 版. 北京：高等教育出版社，2014.

[2] 王珊，李翠平，李盛恩. 数据仓库和数据分析教程. 北京：高等教育出版社，2012.

[3] 王珊. 数据库系统概论(第四版)学习指导与习题解析. 北京：高等教育出版社，2008.

[4] SILBERSCHATZ A，KORTH H F，SUDARSHAN S. Database System Concepts. 6th ed. 杨冬青，李红燕，唐世渭，等，译. 北京：机械工业出版社，2012.

[5] ULLMAN J D，WIDOM J. A First Course in Database Systems. 3rd ed. 岳丽华，金培权，万寿红，等，译. 北京：机械工业出版社，2009.

[6] 谷长勇. 事务处理服务器的性能评价研究. 中国科学院博士学位研究生学位论文，2001.

[7] GRAY J. Database and Transaction Processing Performance Handbook. Morgan Kaufmann，1993.

[8] 王志英，蒋宗礼，杨波，等. 高等学校计算机科学与技术专业实践教学体系与规范. 北京：清华大学出版社，2008.

[9] CELKO J. SQL Programming Style. 米全喜，译. 北京：人民邮电出版社，2008.

[10] 北京人大金仓信息技术股份有限公司. KingbaseES V7.1.2 联机帮助. http://www.kingbase.com.cn/kingbase/newslist/list-186-1.html，2014.

[11] Standard Performance Evaluation Corporation(SPEC). www.spec.org.

[12] Transaction Processing Council(TPC). http://www.tpc.org.

[13] Transaction Processing Council(TPC). TPC BENCHMARK C Standard Specification(Revision 5.11). http://www.tpc.org/tpc_documents_current_versions/pdf/tpc-c_v5-11.pdf.2010.2.

[14] Transaction Processing Council(TPC). TPC BENCHMARK H(Decision Support)Standard Specification(Revision 2.17.0).http://www.tpc.org/tpc_documents_current_versions/pdf/tpch2.17.1.pdf.2014.

[15] HELD G D，STONEBRAKER M，WONG E.INGRES：a Relational Data Base System. In Proceedings of the May 19-22，1975. National Computer Conference and Exposition (AFIPS'75). ACM，New York，NY，USA：409-416.

[16] STONEBRAKER M，MADDEN S，ABADI D J，et al. The End of an Architectural Era：(it's Time for a Complete Rewrite). In Proceedings of the 33rd International Conference on Very large Data Bases(VLDB'07). VLDB Endowment，2007：1150-1160.

[17] PAVLO A，PAULSON E，RASIN A，et al. Comparison of Approaches to Large-Scale Data Analysis.

In Proceedings of the 35th SIGMOD International Conference on Management of Data. ACM Press, NewYork, 2009: 165-178.

[18] STONEBRAKER M, Abadi D, DeWitt D J, et al. MapReduce and Parallel DBMSs: Friends or Foes?. Commun. ACM 53(1), 2010: 64-71.

[19] Michael Stonebraker. What Does 'Big Data' Mean?. blog@ cacm, http://cacm.acm.org/blogs/blog-cacm/155468-what-does-big-data-mean/fulltext, 2012.

[20] Michael Stonebraker. What Does 'Big Data' Mean (Part 2)?. blog@ cacm, http://cacm.acm.org/blogs/blog-cacm/156102-what-does-big-data-mean-part-2/fulltext, 2012.

[21] Michael Stonebraker. Reseachers' Big Data Crisis. blog@ cacm, http://cacm.acm.org/magazines/2012/2/145391-researchers-big-data-crisis-understanding-design-and-functionality/abstract, 2012.

[22] Michael Stonebraker. Why the 'Data Lake' is Really a 'Data Swamp'. blog@ cacm, http://cacm.acm.org/blogs/blog-cacm/181547-why-the-data-lake-is-really-a-data-swamp/fulltext, 2014.