面向 21 世纪课程教材

Textbook Series for 21st Century

数据库系统概论学习指导与习题解答

王　珊

高等教育出版社

内 容 提 要

本书是《数据库系统概论》的配套辅导和补充教材。

本书分为三大部分。第一部分是每章基本知识点的讲解、习题参考解答与解析,模拟试卷及答案;第二部分是三个附录,是《数据库系统概论》的补充教材;第三部分是一张随书的“数据库辅助教学软件”光盘,其中包括电子教案、学生课堂复习与练习解答等部分。

本书是应广大读者的要求,为了配合“数据库系统概论”课程的学习而编写的。通过习题的练习和参考答案,读者可以进一步加深对数据库系统基本概念的理解、对基本技术的运用和对基本知识的掌握。

本书可以作为高等学校计算机相关专业的数据库课程参考书,特别是可以作为《数据库系统概论》一书的补充教材。

前　言

本书是《数据库系统概论》的配套辅导和补充教材。应广大读者的要求,为了配合“数据库系统概论”课程的学习,对原书习题给出参考答案。希望读者通过习题的练习,进一步加深对数据库系统基本概念的理解、对基本技术的运用和对基本知识的掌握。

本书分为三大部分:

第一部分是第一至第十章基本知识点的讲解,每一章的习题解答与解析,以及模拟试卷和答案;

第二部分是三个附录;

第三部分是一张随书的“数据库辅助教学软件”光盘。

第一部分按照《数据库系统概论》一书的章节给出第一至第十章的基本知识点。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,给出了每一章的难点,希望读者学习时更加用心。然后给出了每一章后面的习题解答和部分解析。我特别要提醒读者的是,习题的解答常常不是惟一的,对于问答题是如此,对于程序题、设计题也是如此。读者切不要死记硬背习题答案。这也是我多年来不大愿意写这类书的原因。希望读者能够理解书上的解答后给出自己的正确答案。通过习题的练习,复习和掌握书上的内容,进一步加深对数据库系统基本概念的理解、对基本技术的运用和对基本知识的掌握,提高分析问题和解决问题的能力。

第二部分是三个附录,是《数据库系统概论》一书的补充和参考资料。

附录 A 介绍数据库领域的三位图灵奖获得者。他们是:1973 年图灵奖得主,查尔斯·巴赫曼(Charles W. Bachman)——“网状数据库之父”;1981 年图灵奖得主,埃德加·科德(Edgar Frank Codd)——“关系数据库之父”;1998 年图灵奖得主,詹姆斯·格雷(James Gray)——数据库技术和“事务处理”专家。

附录 B SQL99。我在 1999 年编写《数据库系统概论》第三版时 SQL99 还没有正式公布。现在把 SQL99 作为本书的附录向读者做一个概要的介绍。SQL99 比 SQL/92 增加的新特征非常多。限于篇幅,只介绍 SQL99 对传统的关系数据模型扩展的部分和对于面向对象的主要扩展和支持部分。

附录 C 数据库基准测试 TPC-C。数据库基准(Benchmark)测试是针对数据库管理系统(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。

第三部分是“数据库辅助教学软件”光盘。

这是《数据库系统概论》及其复习参考资料的多媒体教学辅助软件,集多媒体信息于一体。光盘包括电子教案、学生课堂复习与课后练习解答等。

电子教案是原教科书的提炼。它包括《数据库系统概论》前十章的内容,为使用《数据库系统概论》教材的教师,进行计算机投影教学提供方便。

学生课堂复习与课后练习解答,目的在帮助学生课后复习及完成作业。它包括:作者及内容介绍、课程学习、重点与难点、课后练习及答案、多套模拟考试及详细解答,内容生动。学生可以自由地选择自己需要的学习内容,练习已学过的知识,复习总结并进行模拟考试,检查学习成果,是学生课后辅导的好帮手。

本书第一、第二部分由王珊教授执笔,陈红教授和许多研究生参与了习题的初步解答。第三部分“数据库辅助教学软件”由王珊教授和朱青副教授共同策划,朱青副教授负责总体设计和实施,信息学院的倪泳智、温利华、张望、李凌云、杜贵彬等同学参加了软件开发。“数据库辅助教学软件”是在以前版本的基础上重新设计重新开发的。麻占全老师、常瑞君同学曾经在前面版本的设计和开发过程中付出了辛勤的劳动。在此一并向以上老师和学生表示诚挚的谢意。

本书在习题解答和解析中难免有错误和不足,“数据库辅助教学软件”从内容到界面也存在许多不足。希望读者多提宝贵意见和建议,以便在今后的版本中改进提高。

王 珊

2003 年 6 月于中国人民大学

目　录

第一章　绪　　论

有关数据库的教材、专著、书籍很多。《数据库系统概论》(以下简称《概论》)的定位是计算机专业和相关专业的大学本科教材,是“学生学习数据库课程的第一本书”。

由于读者刚刚步入数据库技术的新领域,开始学习《数据库系统概论》这本书,为了给读者一个“什么是数据库”的大致概念,在《概论》第一章中给出一个比较概要也比较全面的介绍,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》主要讲解数据库系统的哪些方面。

一、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情况、数据库技术产生和发展的背景、数据库系统的组成以及数据库技术的主要研究领域。

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,从而为以后的学习打好扎实的基础。

本章讲解的数据库基本概念和基本知识是学习后续各个章节的基础。

读者在刚学习本章时有些概念可能较为抽象,不太容易理解,但仍然需要尽可能地去把握其核心思想。只有这样,才能在以后各个章节的学习中,使得这些概念逐步得到更好的理解,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了如下分类:

① 需要了解的:数据管理技术的产生和发展过程、数据库系统的优点和好处、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、DBA 的职责、数据库技术的主要研究领域等。

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

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,所以把层次和网状数据库的内容加以精简和压

缩后放在第一章介绍。

这两类数据库系统可以划为第一代数据库系统,具有重要的地位,读者必须了解。

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。国外早期开发的采用层次或网状数据库的许多应用系统现在仍然在实际运行之中。

② 需要牢固掌握的:概念模型的基本概念及其主要建模方法——E-R 方法;关系数据模型的相关概念、数据库系统三级模式和两层映像的体系结构、数据库系统的逻辑独立性和物理独立性等。

③ 需要举一反三的:通过 E-R 方法描述现实世界的概念模型。

④ 难点:本章的难点是需要掌握数据库领域大量的基本概念。有些概念的确比较抽象,但不要紧,随着学习的逐渐推进,在后续章节中,这些抽象的概念会逐渐变得清晰、具体起来。此外,数据模型及数据库系统的体系结构也是本章的难点。

下面给出书后习题的参考答案。这里只给出答案的要点,读者可以根据自己的理解给出自己正确的答案。同时我们对一些问题进行了解析,辅助读者理解有关的概念和技术。

二、习题解答和解析

1. 试述数据、数据库、数据库系统、数据库管理系统的概念。

答

(1) 数据(Data):描述事物的符号记录称为数据。数据的种类有数字、文字、图形、图像、声音、正文等。数据与其语义是不可分的。

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。现代计算机能存储和处理的对象十分广泛,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。500 这个数字可以表示一件物品的价格是 500 元,也可以表示一个学术会议参加的人数有 500 人,还可以表示一袋奶粉重 500 克。

(2) 数据库(DataBase,简称 DB):数据库是长期储存在计算机内的、有组织的、可共享的数据集合。数据库中的数据按一定的数据模型组织、描述和储存,具有较小的冗余度、较高的数据独立性和易扩展性,并可为各种用户共享。

解析

简单地讲,数据库中的数据具有永久储存、有组织和可共享三个特点。

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的。

(3) 数据库系统(DataBase Sytem,简称 DBS):数据库系统是指在计算机系统中引入数据库后的系统构成,一般由数据库、数据库管理系统(及其开发工具)、应用系统、数据库管理员构成。

解析

数据库系统和数据库是两个概念。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。但是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。

(4) 数据库管理系统(DataBase Management Sytem,简称 DBMS):数据库管理系统是位于用户与操作系统之间的一层数据管理软件,用于科学地组织和存储数据、高效地获取和维护数据。DBMS 的主要功能包括数据定义功能、数据操纵功能、数据库的运行管理功能、数据库的建立和维护功能。

解析

DBMS 是一个大型的复杂的软件系统,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和 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、Sybase 公司的 Sybase 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。

2 使用数据库系统有什么好处?

答

使用数据库系统的好处是由数据库管理系统的特点或优点决定的。

使用数据库系统的好处很多,例如,可以大大提高应用开发的效率,方便用户的使用,减轻数据库系统管理人员维护的负担,等等。

为什么有这些好处,可以结合第 5 题来回答。

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。用一个通俗的比喻,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应用逻辑的设计,而不必为数据管理的许许多多复杂的细节操心。

还有,当应用逻辑改变,数据的逻辑结构也需要改变时,由于数据库系统提供了数据与程序之间的独立性,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。

使用数据库系统可以减轻数据库系统管理人员维护系统的负担。因为DBMS在数据库建立、运用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、安全性、多用户并发控制、故障恢复等,都由DBMS执行。

总之,使用数据库系统的优点是很多的,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。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。

3. 试述文件系统与数据库系统的区别和联系。

答

文件系统与数据库系统的区别是:

文件系统面向某一应用程序,共享性差,冗余度大,数据独立性差,记录内有结构,整体无结构,由应用程序自己控制。

数据库系统面向现实世界,共享性高,冗余度小,具有较高的物理独立性和一定的逻辑独立性,整体结构化,用数据模型描述,由数据库管理系统提供数据的安全性、完整性、并发控制和恢复能力。

读者可以参考书中表1.1中的有关内容。

文件系统与数据库系统的联系是:

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。

解析

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。但是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的文件系统来实现的。

读者可以参考《概论》第十一章“数据库管理系统”,或者进一步学习有关数据库管理系统实现的课程(第十一章只是DBMS实现技术的概述)来对本题有深入的理解和全面的解答。因为DBMS的实现与操作系统中的文件系统是紧密相关的,例如,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。

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。

答

(1) 适用于文件系统而不是数据库系统的应用例子

数据的备份、软件或应用程序使用过程中的临时数据存储一般使用文件比较合适。早期功能比较简单、比较固定的应用系统也适合用文件系统。

(2) 适用于数据库系统而非文件系统的应用例子

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。例如,一个工厂的管理信息系统(其中会包括许多子系统,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、人事管理系统等),学校的学生管理系统,人事管理系统,图书馆的图书管理系统,等等,都适合用数据库系统。

希望读者能举出自己了解的应用例子。

5. 试述数据库系统的特点。

答

数据库系统的主要特点有:

(1) 数据结构化

数据库系统实现整体数据的结构化,这是数据库的主要特征之一,也是数据库系统与文件系统的本质区别。

解析

注意这里的"整体"两个字。在数据库系统中,数据不再针对某一个应用,而是面向全组织,具有整体的结构化。不仅数据是结构化的,而且数据的存取单位即一次可以存取数据的大小也很灵活,可以小到某一个数据项(如一个学生的姓名),大到一组记录(成千上万个学生记录)。而在文件系统中,数据的存取单位只有一个:记录,如一个学生的完整记录。

(2) 数据的共享性高,冗余度低,易扩充

数据库的数据不再面向某个应用而是面向整个系统,因此可以被多个用户、多个应用以多种不同的语言共享使用。由于数据面向整个系统,是有结构的数据,不仅可以被多个应用共享使用,而且容易增加新的应用,这就使得数据库系统弹性大,易于扩充。

解析

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"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,可以说数据是某个应用的"私有资源"。

所谓"弹性大"是指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,只需做很少的改动。

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。

数据库管理系统的模式结构和二级映像功能保证了数据库中的数据具有很高的物理独立性和逻辑独立性。

解析

所谓“独立性”指的是相互不依赖。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,从而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、数据的完整性检查、并发控制和数据库恢复。

解析

DBMS 数据控制功能包括四个方面:

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的泄密和破坏;

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;

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)。

下面可以得到“什么是数据库”的一个定义:

数据库是长期存储在计算机内有组织的大量的共享的数据集合,它可以供各种用户共享,具有最小冗余度和较高的数据独立性。DBMS 在数据库建立、运用和维护时对数据库进行统一控制,以保证数据的完整性、安全性,并在多用户同时使用数据库时进行并发控制,在发生故障后对系统进行恢复。

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的数据库为中心的新阶段。

6. 数据库管理系统的主要功能有哪些?

答

(1) 数据库定义功能;

(2) 数据存取功能;

(3) 数据库运行管理;

(4) 数据库的建立和维护功能。

7. 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。

一般地讲,数据模型是严格定义的概念的集合。这些概念精确描述了系统的静态特性、动态特性和完整性约束条件。因此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:是所研究的对象类型的集合,是对系统静态特性的描述。

(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:是一组完整性规则的集合。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、相容。

解析

数据模型是数据库系统中最重要的概念之一。必须通过《概论》的学习真正掌握数据模型的概念和作用。

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:一类是概念模型,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;另一类是数据模型,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、操纵数据库中的数据,一般需要有严格的形式化定义和一组严格定义了语法和语义的语言,并有一些规定和限制,便于在机器上实现。

8. 试述概念模型的作用。

答

概念模型实际上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,是现实世界到信息世界的第一层抽象,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。

9. 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:惟一标识实体的属性集称为码。

实体联系图(E-R 图):提供了表示实体型、属性和联系的方法:

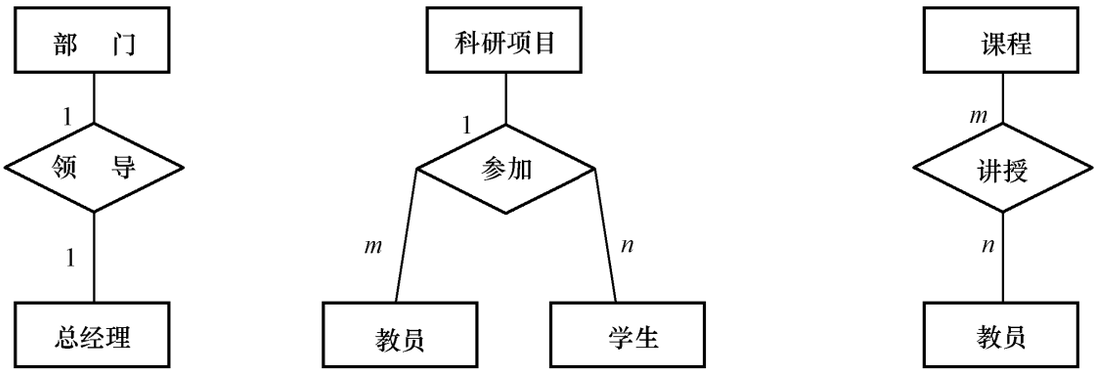
实体型:用矩形表示,矩形框内写明实体名;

属性:用椭圆形表示,并用无向边将其与相应的实体连接起来;

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

10. 试给出 3 个实际部门的 E-R 图,要求实体型之间具有一对一、一对多、多对多各种不同的联系。

答



一对一联系　　一对多联系　　多对多联系

解析

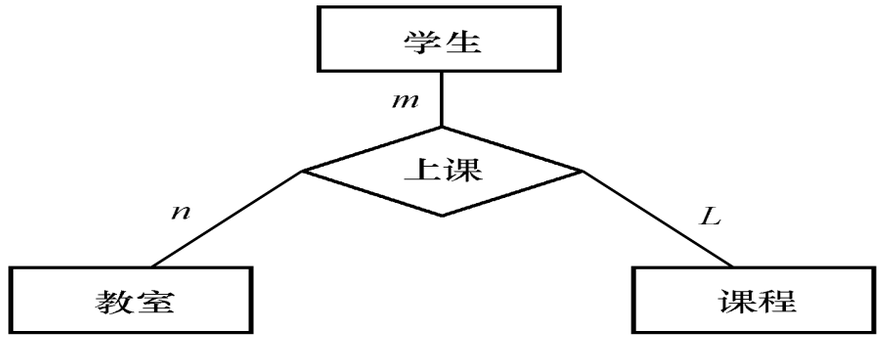
部门和总经理是两个实体。一个部门或者一个公司只有一个总经理,一个总经理只在一个公司工作,不能兼任两个以上公司的总经理,所以部门和总经理两个实体之间是一对一的联系。

一个科研项目可以由多个教员和多个学生承担。如果规定一个教员只能参加一个项目,一个学生也只能参加一个项目。按照这样的语义,科研项目和教员、学生三者之间是一对多的联系。

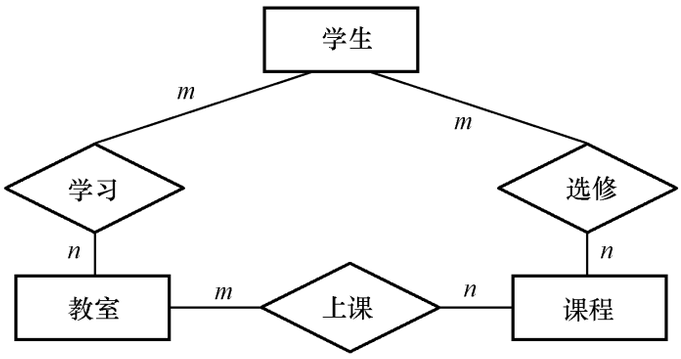
如果一门课程可以由多个教员讲授,一个教员可以讲授多门课程,则课程和教员之间是多对多的联系。

11. 试给出一个实际部门的 E-R 图,要求有三个实体型,而且 3 个实体型之间有多对多联系。3 个实体型之间的多对多联系和三个实体型两两之间的三个多对多联系等价吗?为什么?

答

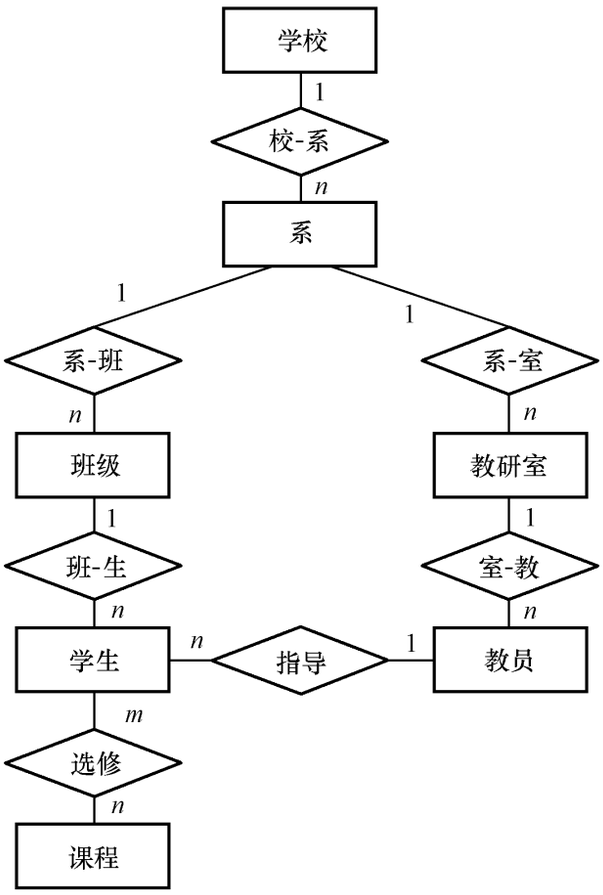


3 个实体型之间的多对多联系和 3 个实体型两两之间的 3 个多对多联系是不等价,因为它们拥有不同的语义。3 个实体型两两之间的三个多对多联系如下图所示。



12 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



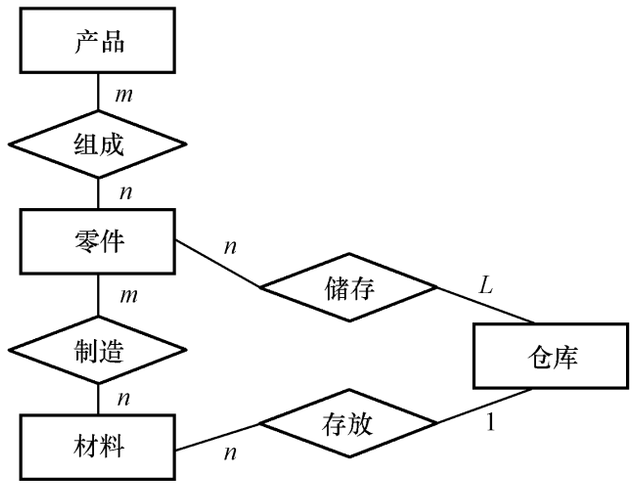
解析

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。再如,有的教授和副教授每人各带若干研究生,而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。

13. 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

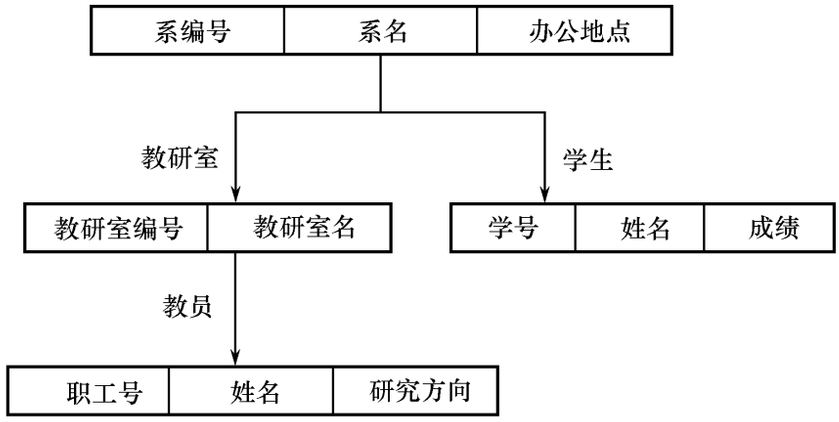
对实体之间联系的语义描述有时不是直截了当的,需要从对现实世界的整体描述中进行分析,导出实体之间的某种联系。就如本题中,“零件和仓库的联系”就要从以下描述中分析:“零件按所属的不同产品分别放在仓库中”。因为一个产品由多种零件组成的,所以一个仓库中存放多种零件;反过来一种零件是放在一个仓库还是多个仓库中呢?因为一种零件可以用在多种产品上,这些零件按所属的不同产品分别放在仓库中,于是知道一种零件可以放在多个仓库中,所以零件和仓库之间是多对多的联系。

“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。

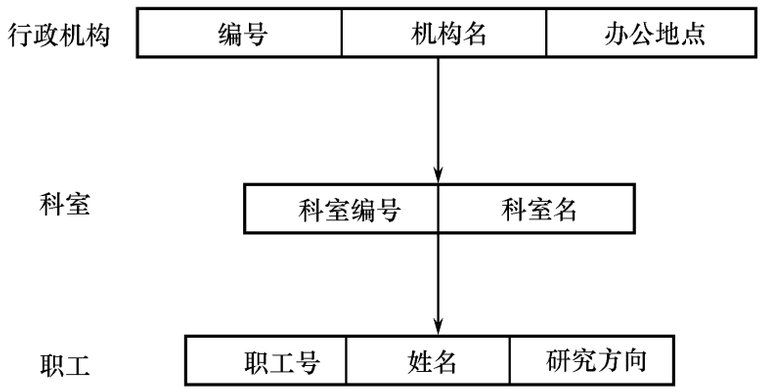
14. 试述层次模型的概念,举出三个层次模型的实例。

答

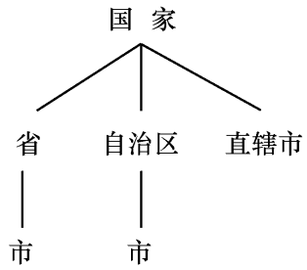
(1) 教员学生层次数据库模型



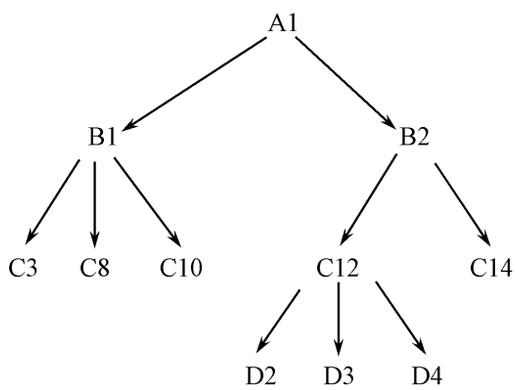
(2) 行政机构层次数据库模型



(3) 行政区域层次数据库模型

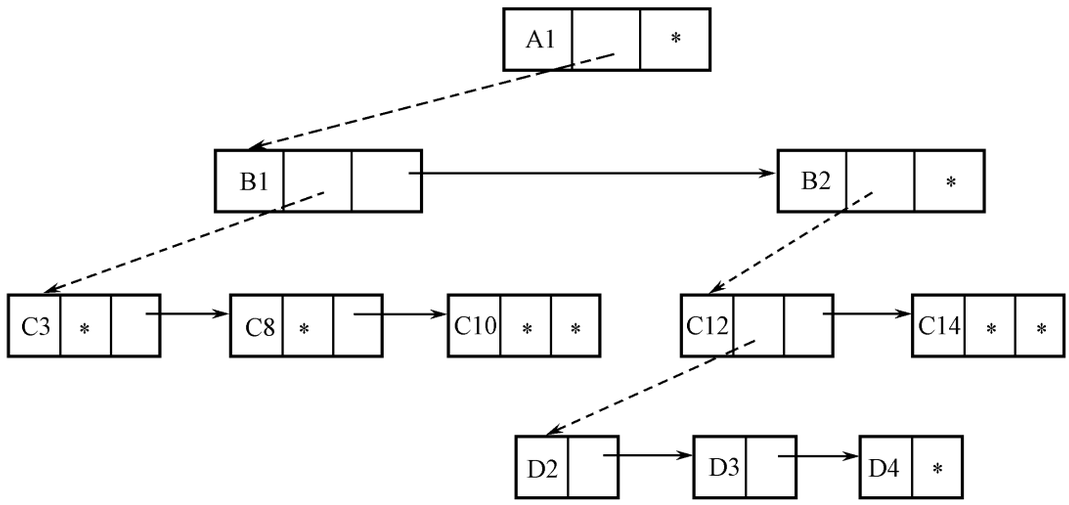


15. 今有一个层次数据库实例,试用子女 - 兄弟链接法和层次序列链接法画出它的存储结构示意图。



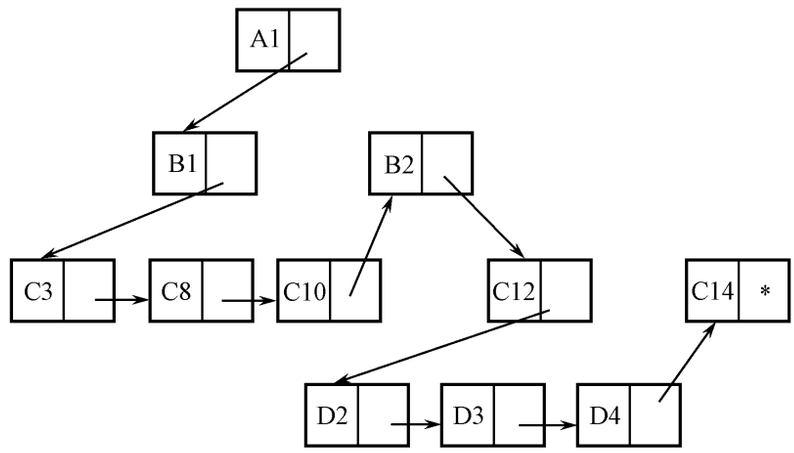
答

子女 - 兄弟链接法:



图中虚数表示子女链,记录结构的第二部分存放子女指针;实线表示兄弟链,记录结构的第三部分存放兄弟指针。星号“ * ”表示空指针。

层次序列链接法 :



16. 试述网状模型的概念,举出三个网状模型的实例。

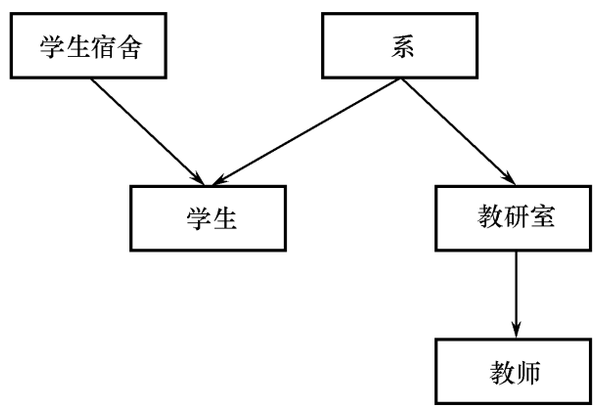
答

满足下面两个条件的基本层次联系集合为网状模型。

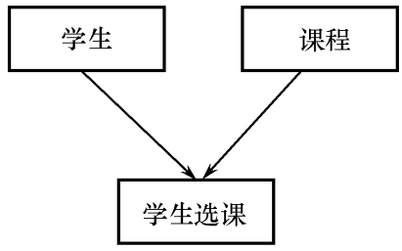
(1) 允许一个以上的结点无双亲;

(2) 一个结点可以有多于一个的双亲。

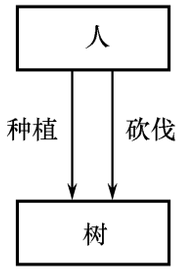
实例 1:



实例 2:



实例 3:



17. 试述网状、层次数据库的优缺点。

答

层次模型的优点主要有:

(1) 模型简单,对具有一对多层次关系的部门描述非常自然、直观,容易理解,这是层次数据库的突出优点;

(2) 用层次模型的应用系统性能好,特别是对于那些实体间联系是固定的且预先定义好的应用,采用层次模型来实现,其性能优于关系模型;

(3) 层次数据模型提供了良好的完整性支持。

层次模型的缺点主要有:

(1) 现实世界中很多联系是非层次性的,如多对多联系、一个结点具有多个双亲等,层次模型不能自然地表示这类联系,只能通过引入冗余数据或引入虚拟结点来解决;

(2) 对插入和删除操作的限制比较多;

(3) 查询子女结点必须通过双亲结点。

网状数据模型的优点主要有:

(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。

网状数据模型的缺点主要有:

(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) 其 DDL、DML 语言复杂,用户不容易使用。

由于记录之间联系是通过存取路径实现的,应用程序在访问数据时必须选择适当的存取路径。因此,用户必须了解系统结构的细节,加重了编写应用程序的负担。

18. 试述关系模型的概念,定义并解释以下术语:

(1) 关系 (2) 属性 (3)域 (4)元组

(5) 主码 (6) 分量 (7)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) 主码:表中的某个属性组,它可以惟一确定一个元组;

(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)

19. 试述关系数据库的特点。

答

关系数据模型具有下列优点:

(1) 关系模型与非关系模型不同,它是建立在严格的数学概念的基础上的。

(2) 关系模型的概念单一,无论实体还是实体之间的联系都用关系表示,操作的对象和操作的结果都是关系,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,其中最主要的缺点是,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,增加了开发数据库管理系统的难度。

20. 试述数据库系统三级模式结构,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。(参见书上图 1.29)

外模式,亦称子模式或用户模式,是数据库用户(包括应用程序员和最终用户)能够看见和使用的局部数据的逻辑结构和特征的描述,是数据库用户的数据视图,是与某一应用有关的数据的逻辑表示。

模式,亦称逻辑模式,是数据库中全体数据的逻辑结构和特征的描述,是所有用户的公共数据视图。模式描述的是数据的全局逻辑结构。外模式涉及的是数据的局部逻辑结构,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。

数据库系统的三级模式是对数据的三个抽象级别,它把数据的具体组织留给 DBMS 管理,使用户能逻辑抽象地处理数据,而不必关心数据在计算机中的表示和存储。

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层映像:外模式/ 模式映像和模式/ 内模式映像。正是这两层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。

21. 定义并解释以下术语:模式、外模式、内模式、DDL、DML

答

模式、外模式、内模式的解释参见第 20 题。

DDL:数据定义语言,用来定义数据库模式、外模式、内模式的语言。

DML: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。

22. 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

数据与程序的逻辑独立性:当模式改变时(例如增加新的关系、新的属性、改变属性的数据类型等),由数据库管理员对各个外模式/ 模式的映像做相应改变,可以使外模式保持不变。应用程序是依据数据的外模式编写的,从而应用程序不必修改,保证了数据与程序的逻辑独立性,简称数据的逻辑独立性。

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/ 内模式映像做相应改变,可以使模式保持不变,从而应用程序也不必改变,保证了数据与程序的物理独立性,简称数据的物理独立性。

数据库管理系统在三级模式之间提供的两层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。

23. 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。

24. DBA 的职责是什么?

答

负责全面地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 监督和控制数据库的使用和运行;

⑤ 改进和重组数据库系统。

25. 系统分析员、数据库设计人员、应用程序员的职责是什么?

答

系统分析员负责应用系统的需求分析和规范说明,系统分析员要和用户及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、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。在很多情况下,数据库设计人员就由数据库管理员担任。

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。

第二章　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。

一、基本知识点

关系模型和关系数据库是《数据库系统概论》一书的重点,在全书中占有较大的篇幅(包括第二、三、四、五章)。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:关系数据库理论产生和发展的过程,关系数据库产品的发展及沿革;关系演算的概念;域关系演算语言不包括在本科教学大纲内。

② 需要牢固掌握的: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(包括抽象的语言及具体的语言);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)、元组关系演算语言 ALPHA 及域关系演算语言 QBE 等,能够使用这些语言完成各种数据操纵。

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学习。同时,要注意把握由具体语言到抽象语言的原则,即通过对具体语言如 ALPHA 和 QBE 的学习过渡到对抽象的关系演算的把握。

二、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:

关系数据语言
- 关系代数语言 例如 ISBL
- 关系演算语言
  - 元组关系演算语言 例如 APLHA,QUEL
  - 域关系演算语言 例如 QBE
- 具有关系代数和关系演算双重特点的语言 例如 SQL

这些关系数据语言的共同特点是:具有完备的表达能力;是非过程化的集合操作语言;功能强;能够嵌入高级语言中使用。

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 $D_1, D_2, \cdots, D_n$,这些域中可以有相同的。这组域的笛卡儿积为

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

其中每一个元素 $(d_1, d_2, \cdots, d_n)$ 叫做一个 n 元组(n-tuple)或简称元组(Tuple)。元素中的每一个值 d_i 叫做一个分量(Component)。

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

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

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) 主码,候选码,外部码

答

候选码:若关系中的某一属性组的值能惟一地标识一个元组,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)。

外部码: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。

基本关系 R 称为参照关系(Referencing relation),基本关系 S 称为被参照关系(Referenced relation)或目标关系(Target relation)。关系 R 和 S 可以是相同的关系。

(3) 关系模式,关系,关系数据库

答

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,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:在域 D_1, D_2,…, 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,…, D_n)

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。关系数据库的型也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. 试述关系模型的完整性规则。在参照完整性中,为什么外部码属性的值也可以为空?什么情况下才可以为空?

答

关系模型的完整性规则是对关系的某种约束条件。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。

其中实体完整性和参照完整性是关系模型必须满足的完整性约束条件,被称做是关系的两个不变性,应该由关系系统自动支持。

(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:

1) 或者取空值(F 的每个属性值均为空值);

2) 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,外部码属性的值可以为空,它表示该属性的值尚未确定,但前提条件是该外部码属性不是其所在关系的主属性。

例如,在下面的“学生”表中,“专业号”是一个外部码,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(<u>学号</u>,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”虽然也是一个外部码属性,但它又是“课程”表的主属性,所以不能为空,因为关系模型必须满足实体完整性。

课程(课程号,课程名,学分)

选修(学号,课程号,成绩)

5. 设有一个 SPJ 数据库,包括 S、P、J、SPJ 四个关系模式:

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	500

试分别用关系代数、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

$$\pi_{SNO}(\sigma_{JNO=J1}(SPJ))$$

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

$$\pi_{SNO}(\sigma_{JNO=J1 \wedge PNO=P1}(SPJ))$$

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

$$\pi_{SNO}(\pi_{SNO,PNO}(\sigma_{JNO=J1}(SPJ)) \quad \pi_{PNO}(\sigma_{COLOR=红}(P)))$$

ALPHA 语言

RANGE P PX

GET W (SPJ.SNO): v PX(PX.PNO = SPJ.PNO ∧SPJ.JNO = J1 ∧PX .COLOR = 红)

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

$\pi_{JNO}(J) - \pi_{JNO}(\pi_{SNO}(\sigma_{CITY=\text{天津}}(S)) \quad \pi_{SNO,PNO,JNO}(SPJ) \quad \pi_{PNO}(\sigma_{COLOR=\text{红}}(P)))$

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,$\pi_{JNO}(J)$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):¬ v SPJX(SPJX .JNO=J.JNO∧
                    v SX ( SX.SNO=SPJX .SNO∧SX .CITY= 天津 ∧
                    v 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: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: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。

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。

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。

直到所有 JNO 都检查完。

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,没有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没有了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

$$\pi_{JNO,PNO}(SPJ) \div \pi_{PNO}(\sigma_{SNO='S1'}(SPJ))$$

解析

上面公式中除号前的部分是所有工程与该工程所用的零件,除号后的部分是 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,语言表达也很简单。

ALPHA 语言(类似于《概论》P67 例 14)

```
RANGE SPJ SPJX
```

```
        SPJ SPJY
        P    PX
GET W(J.JNO): " PX(v SPJX(SPJX.PNO = PX.PNO∧SPJX.SNO = S1 )
                    v 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。

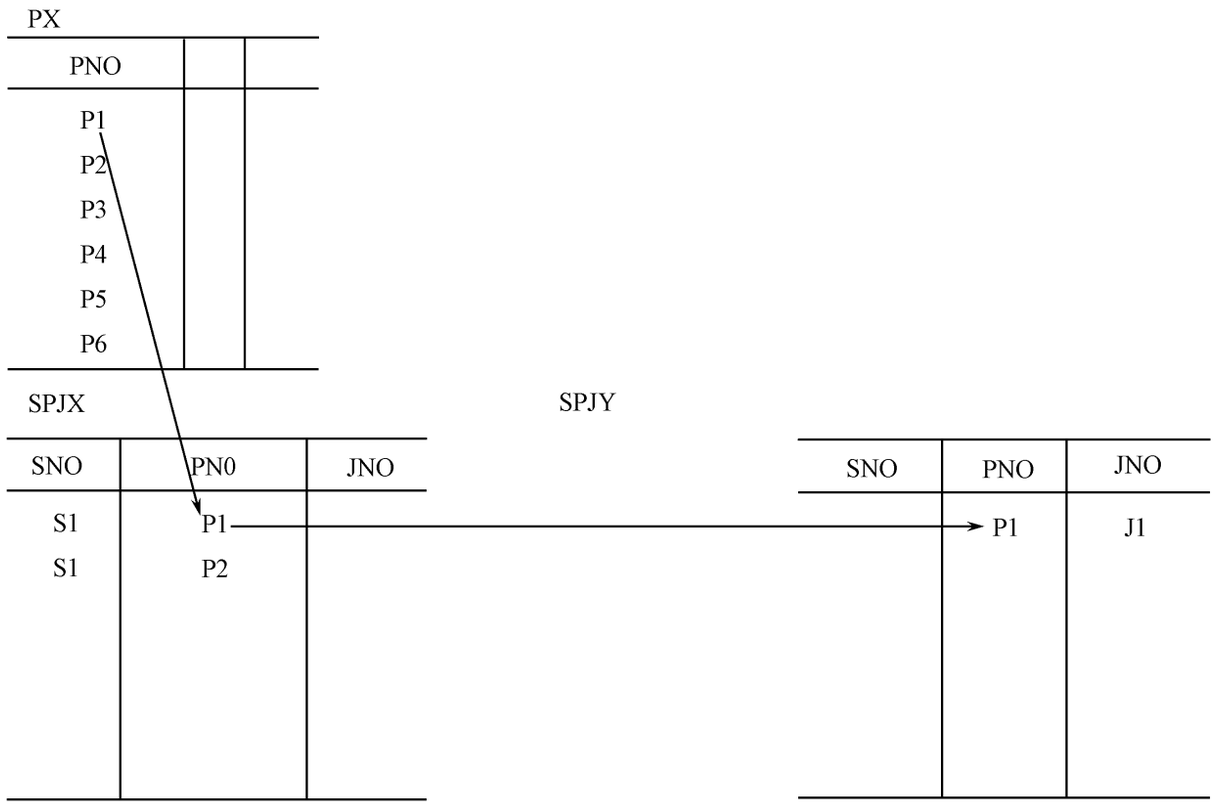
2) 解题思路是:要找的是满足给定条件的工程号 JNO。因此,对工程表 J 中的每一个 JNO(例如 J1),进行以下一组操作:

① 对零件 PX 中的所有零件,依次对每一个零件,进行以下检查;

② 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;

③ 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一步一步地分析,就能掌握解题方法,从而达到举一反三的要求。



QBE(不要求)。

6. 试述等值连接与自然连接的区别和联系。

答

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

θ为“ = ”的连接运算,称为等值连接。

它是从关系 R 与 S 的笛卡儿积中选取 A、B 属性值相等的那些元组。即等值连接为

$$R_{A=B}S=\{t_r t_s \mid t_r\in R\wedge t_s\in S\wedge t_r[A]=t_s[B]\}$$

自然连接(Natural join)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必须是相同的属性组,并且在结果中把重复的属性列去掉。即若 R 和 S 具有相同的属性组 B,则自然连接可记为

$$R\ S=\{t_r t_s \mid t_r\in R\wedge t_s\in S\wedge t_r[B]=t_s[B]\}$$

7. 关系代数的基本运算有哪些?如何用这些基本运算来表示其他的运算?

答

在八种关系代数运算中,并、差、笛卡儿积、投影和选择五种运算为基本的运算。其他三种运算,即交、连接和除,均可以用五种基本运算来表达。

交运算: $R\cap S=R-(R-S)$

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

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

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。

三、小 结

关系数据库系统是目前使用得最为广泛的数据库系统,1970 年以后开发的数据库系统几乎都是基于关系的。可以说,关系模型的提出是数据库发展史上最为重要的成就。

本章对关系模型的核心概念进行了阐述,并重点讲述了关系数据结构、关系操作集合以及关系完整性约束的概念。对以上三个部分的掌握是学习后续章节的基础。

学习本章的关键在于理解关系模型的核心与本质,而不要仅仅局限于理解相关概念的字面意思。读者应该达到能够对所学知识灵活运用的程度。

第三章　关系数据库标准语言 SQL

第三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,内容十分丰富,是学习关系数据库概念和技术的重要部分。

一、基本知识点

关系模型和关系数据库是《概论》的重点,第三章又是重点中的重点,是全书中篇幅最大的一章,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 语言发展的过程,从而进一步了解关系数据库技术和 RDBMS 产品的发展过程。

② 需要牢固掌握的:掌握 SQL 语言的特点、SQL 语言与非关系模型(层次模型、网状模型)数据语言的不同,从而体会 SQL 语言之所以能够为用户和业界所接受,并成为国际标准的原因;体会面向过程的语言和 SQL 语言的区别和优点;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、减轻用户负担、提高用户生产率的原因。

③ 需要举一反三的:熟练而正确地使用 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。

二、习题解答和解析

1. 试述 SQL 语言的特点。

答

(1) 综合统一。SQL 语言集数据定义语言 DDL、数据操纵语言 DML、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”,而无需指明“怎么做”,因此无需了解存取路径,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;作为嵌入式语言,它能够嵌入到高级语言程序中,供程序员设计程序时使用。

(5) 语言简捷,易学易用。

解析

详细的可参考《概论》上 3.1.1。注意不要仅仅背这些特点,关键是要通过具体的练习、使用 SQL 语句来理解这些特点。

2. 试述 SQL 的定义功能。

答

SQL 的数据定义功能包括定义表、定义视图和定义索引。

SQL 语言使用 CREATE TABLE 语句建立基本表,ALTER TABLE 语句修改基本表定义,DROP TABLE 语句删除基本表;使用 CREATE INDEX 语句建立索引,DROP INDEX 语句删除索引;使用 CREATE VIEW 语句建立视图,DROP VIEW 语句删除视图。

3. 用 SQL 语句建立第二章习题 5 中的 4 个表。

答

对于 S 表:S(SNO,SNAME,STATUS,CITY);

建 S 表

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

对于 P 表:P(PNO,PNAME,COLOR,WEIGHT);

建 P 表

```
CREATE TABLE P
    (PNO CHAR(3),
     PNAME CHAR(10),
```

```
    COLOR CHAR(4),
    WEIGHT INT);
```

对于 J 表:J(JNO,JNAME,CITY);

建 J 表

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

对于 SPJ 表: SPJ(SNO,PNO,JNO,QTY);

建 SPJ 表

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

4. 针对上题中建立的 4 个表试用 SQL 语言完成第二章习题 5 中的查询。

答

读者可以对比 SQL 语言、关系代数、ALPHA 语言、QBE 语言,体会各种语言的优点。

(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                          /*这是两表连接查询*/
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

读者可以对比第二章习题 5 中用 ALPHA 语言来完成该查询的解答。如果大家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。

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

这里的第一种解法是使用多重嵌套查询,第二种方法的子查询是一个多表连接。

注意:从 J 表入手,以包含那些尚未使用任何零件的工程号。

```
SELECT JNO
FROM  J
WHERE NOT EXISTS
      (SELECT *
       FROM SPJ
       WHERE SPJ.JNO=J.JNO
          AND SNO IN                /*天津供应商的 SNO*/
              (SELECT 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
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》P113 例 44)。

```
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      /*用别名将父查询与子查询中的 SPJ 表区分开*/
       WHERE SNO = S1
          AND NOT EXISTS
              (SELECT *
               FROM SPJ SPJY
               WHERE SPJY.PNO = SPJX.PNO
                  AND SPJY.JON = SPJZ.JNO));
```

解析

本查询的解析可以参考第二章第 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(" y)p→q 为真。即,对于所有的零件 y,满足逻辑蕴涵 p→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:

(" y)p→q≡¬ (v y(¬ (p→q))≡¬ (v y(¬ (¬ p∨q)≡¬ v y(p∧¬ 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。

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,请做必要的修改。

```
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有值*/
                                   /*并且属性列要和表定义中的次序对应*/
```

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》3.5.4。

8. 所有的视图是否都可以更新？为什么？

答

不是。视图是不实际存储数据的虚表,因此对视图的更新,最终要转换为对基本表的更新。因为有些视图的更新不能惟一有意义地转换成对相应基本表的更新,所以,并不是所有的视图都是可更新的,如《概论》3.5.1 中的视图 S_G(学生的学号及他的平均成绩)

```
CREAT VIEW S_G(Sno,Gavg)
    AS SELECT Sno,AVG(Grade)   /*设 SC 表中"成绩"列 Grade 为数字型*/
       FROM SC
       GROUP BY Sno;
```

要修改平均成绩,必须修改各科成绩,而我们无法知道哪些课程成绩的变化导致了平均成绩的变化。

9. 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》3.5.3 中的例 1。

若视图的属性来自集函数、表达式,则该视图肯定是不可以更新的,如《概论》3.5.3 中的 S_G 视图。

10. 试述某个你熟悉的实际系统中对视图更新的规定。

答

(略)

解析

不同的系统对视图更新的规定是不同的,读者必须了解你所用系统对视图更新的规定。

11. 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 的供应情况。

答

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) 找出三建工程项目使用的各种零件代码及其数量。

```
SELECT PNO, QTY
FROM V_SPJ;
```

(2) 找出供应商 S1 的供应情况。

```
SELECT PNO, QTY      /* S1 供应三建工程的零件号和对应的数量 */
FROM V_SPJ
WHERE SNO = S1 ;
```

12. 针对习题 3 建立的表,用 SQL 语言完成以下各项操作:

(1) 把对表 S 的 INSERT 权限授予用户张勇,并允许他再将此权限授予其他用户。

答

```
GRANT INSERT
ON TABLE S
TO 张勇
WITH GRANT OPTION;
```

(2) 把查询 SPJ 表和修改 QTY 属性的权限授给用户李天明。

答

```
GRANT SELECT, UPDATE(QTY)
ON TABLE SPJ
TO 李天明;
```

13. 在嵌入式 SQL 中是如何区分 SQL 语句和主语言语句的？

答

在 SQL 语句前加上前缀 EXEC SQL,SQL 语句的结束标志则随主语言的不同而不同。

例如,在 PL/1 和 C 中,以分号(;)结束,在 COBOL 中,以 END-EXEC 结束。

14. 在嵌入式 SQL 中是如何解决数据库工作单元与源程序工作单元之间通信的？

答

数据库工作单元与源程序工作单元之间的通信主要包括:

(1) SQL 通信区 SQLCA,用来向主语言传递 SQL 语句的执行状态信息,使主语言能够根据此信息控制程序流程。

(2) 主变量(Host Variable):

1) 用来实现主语言向 SQL 语句提供参数;

2) 将 SQL 语句查询数据库的结果交主语言进一步处理。

(3) 游标(Cursor),解决集合性操作语言与过程性操作语言的不匹配,通过游标逐一获取记录,并赋给主变量,交由主语言进一步处理。

详细解释参见《概论》3.7.2。

15. 在嵌入式 SQL 中是如何协调 SQL 语言的集合处理方式和主语言的单记录处理方式的？

答

用游标来协调这两种不同的处理方式。游标区是系统为用户开设的一个数据缓冲区,存放 SQL 语句的执行结果,每个游标区都有一个名字。用户可以通过游标逐一获取记录,并赋给主变量,交由主语言进一步处理。

三、大 作 业

使用某一个 RDBMS 产品进行 SQL 语言的练习。

你可以使用学校已经安装的 RDBMS 产品,也可以从网上下载一些免费的 RDBMS 软件。最好使用主流的产品,为今后的实际工作积累知识和技术。

练习内容:

1. 了解或学会安装 RDBMS 的步骤;
2. 了解具体产品的功能和特点;
3. 先使用产品自带的样本数据库进行练习;
4. 再用 SQL 语句建立第二章习题 5 中的 4 个表;
5. 使用 SQL 语句尽量多的向 4 个表中插入数据;
6. 完成本章练习题,用实际数据检查你的执行结果是否有错;
7. 使用 SQL 语句建立第三章中的学生 - 课程数据库:

学生 - 课程数据库中包括 3 个表:

学生表:Student(Sno,Sname,Ssex,Sage,Sdept)

课程表:Course(Cno,Cname,Cpno,Ccredit)

学生选课表:SC(Sno,Cno,Grade)

使用 SQL 语句尽量多地向 3 个表中插入数据;

8. 完成第三章书上的例题,用实际数据检查你的执行结果是否有错。

第四章　关系系统及其查询优化

第四章进一步讲解关系数据库的基本概念。简单介绍关系数据库管理系统查询优化的重要概念和实现技术。

查询优化是 RDBMS 的内部实现技术,对于一般用户应该是透明的。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、不同的数据库状况优化效果也不一样,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。希望用户能够写出“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 Turning)”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。

一、基本知识点

本章进一步讲解关系模型和关系系统这两个基本概念。希望读者知道这是两个不同的、又是紧密相关的概念。

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:关系系统的定义和分类;全关系系统的十二条准则。

② 需要牢固掌握的:最小关系的系统、关系上完备的系统和全关系型的关系系统等基本概念;什么是关系系统的查询优化。

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。

二、习题解答和解析

1. 试给出各类关系系统的定义:最小关系系统;关系上完备的系统;全关系型的关系系统。

答

最小关系系统:

一个系统可定义为最小关系系统,当且仅当它:

(1) 支持关系数据库(关系数据结构),从用户观点看,关系数据库由表构成,并且只有表这一种结构;

(2) 支持选择、投影和(自然)连接运算,对这些运算不必要求定义任何物理存取路径。

关系上完备的系统:

这类系统支持关系数据结构和所有的关系代数操作(或者功能上与关系代数等价的操作)。

全关系型的关系系统:

这类系统支持关系模型的所有特征。即不仅是关系上完备的而且支持数据结构中域的概念,支持实体完整性和参照完整性。

解析

(1) 通过本题,同学要清楚知道不同的关系系统支持关系模型的程度是不同的。

(2) 最小关系系统是指一个 RDBMS 最起码的条件。如果一个数据库厂商声称他的 DBMS 是关系的,那么它必须满足这两个最基本的要求。否则,就不是 RDBMS。例如表式系统、倒排表系统就不能算关系系统。

(3) 关系数据模型是由数据结构、关系操作和完整性约束条件这三部分组成,是按照这三部分内容来考察一个关系系统,并进行分类的。

(4) 《概论》上图 4.1 很直观地给出了不同的系统支持关系模型的程度,读者可以用这个图帮助理解和记忆。

*2 试述全关系型系统应满足的十二条准则,以及十二条基本准则的实际意义和理论意义。

答

关系模型的奠基人 E.F.Codd 具体地给出了全关系型的关系系统应遵循的十二条基本准则。从实际意义上看,这十二条准则可以作为评价或购买关系型产品的标准。从理论意义上看,它是对关系数据模型具体而又深入的论述,是从理论和实际紧密结合的高度对关系型 DBMS 的评述。

准则 0 一个关系型的 DBMS 必须能完全通过它的关系能力来管理数据库。

准则 1 信息准则。关系型 DBMS 的所有信息都应在逻辑一级上用一种方法即表中的值显式地表示。

准则 2 保证访问准则。依靠表名、主码和列名的组合,保证能以逻辑方式访问关系数据库中的每个数据项(分量值)。

准则 3 空值的系统化处理。全关系型的 DBMS 应支持空值的概念,并用系

统化的方式处理空值。

准则 4 基于关系模型的动态的联机数据字典。数据库的描述在逻辑级上应该和普通数据采用同样的表示方式,使得授权用户可以使用查询一般数据所用的关系语言来查询数据库的描述信息。

准则 5 统一的数据子语言准则。

准则 6 视图更新准则。所有理论上可更新的视图也应该允许由系统更新。

准则 7 高级的插入、修改和删除操作。

准则 8 数据物理独立性。无论数据库的数据在存储表示或存取方法上作任何变化,应用程序和终端活动都保持逻辑上的不变性。

准则 9 数据逻辑独立性。当对基本关系进行理论上信息不受损害的任何改变时,应用程序和终端活动都保持逻辑上的不变性。

准则 10 数据完整性的独立性。关系数据库的完整性约束条件必须是用数据库语言定义并存储在数据字典中的,而不是在应用程序中加以定义的。

准则 11 分布独立性。关系型 DBMS 具有分布独立性。

准则 12 无破坏准则。如果一个关系系统具有一个低级(指一次处理一个记录)语言,则这个低级语言不能违背或绕过完整性准则。

解析

不要求读者背熟全关系型系统应满足的这十二条准则,而是要理解每一条准则的含义。当在选择或购买 RDBMS 时可以按照这些准则的内容来衡量和评价实际的 RDBMS 产品。要学以致用,把这些准则具体化。

3. 试述查询优化在关系数据库系统中的重要性和可能性。

答

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:

这是因为:

(1) 优化器可以从数据字典中获取许多统计信息,例如关系中的元组数、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^+ 树索引还是 HASH 索引或惟一索引或组合索引)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。

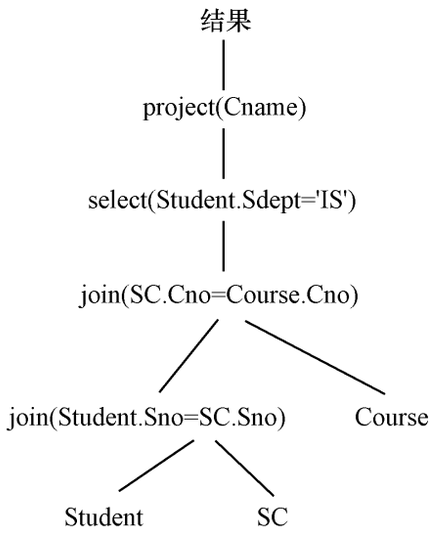
(4) 优化器中包括了很多复杂的优化技术,这些优化技术往往只有最好的程序员才能掌握。系统的自动优化相当于使得所有人都拥有这些优化技术。

4. 对学生 - 课程数据库有如下的查询:

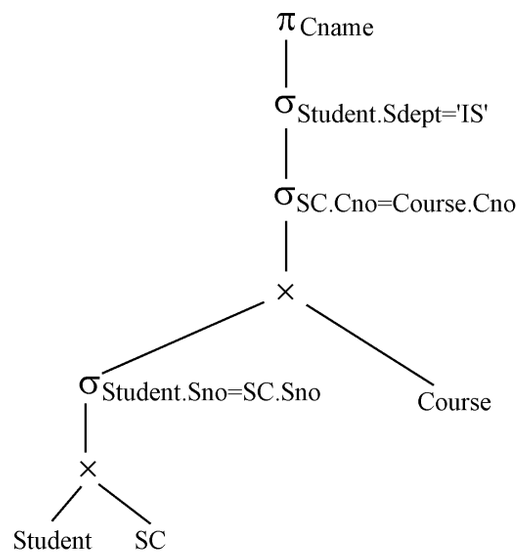
```
SELECT Cname
FROM Student,Course,SC
WHERE Student.Sno = SC.Sno
    AND SC.Cno = Course.Cno
    AND Student.Sdept = 'IS';
```

此查询要求信息系学生选修了的所有课程名称。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。

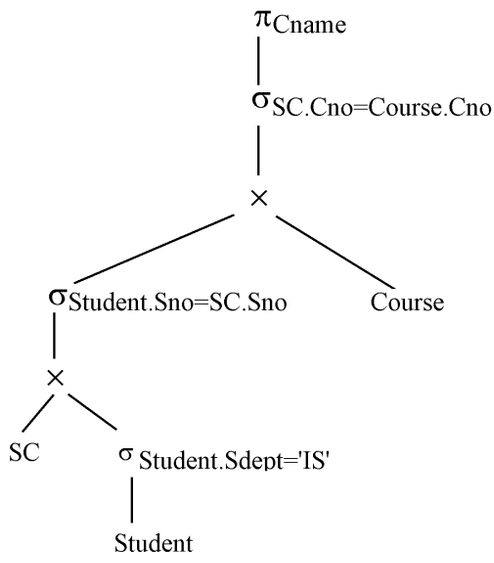
答



最初的语法树



关系代数语法树



优化后的语法树

5. 试述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)涉及物理优化了。

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。这是最基本的也是很有效的优化策略。

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。可以把视图计算出来,称为视图的实体化,计算结果称为实体化视图。

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。许多 RDBMS 提供了多种连接算法供优化子系统选择。

有时需要在执行这些连接算法前对关系进行预处理。如对于索引连接算法,有时要在连接属性上建立索引;对于排序合并算法,要对连接的两个关系首先进行排序,然后执行连接。这就是在执行连接前对关系的预处理。

6. 试述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,选择代价最小的。

解析

为了帮助同学进一步理解查询优化的概念,在这里粗略地画出 SQL 查询处理工作的框图。

查询处理工作的第一步是对 SQL 等高级查询语言所表示的查询进行扫描、语法分析和有效性检查。扫描器从查询语句表达式中识别出语言符号,如 SQL 关键字、属性名和关系名等。

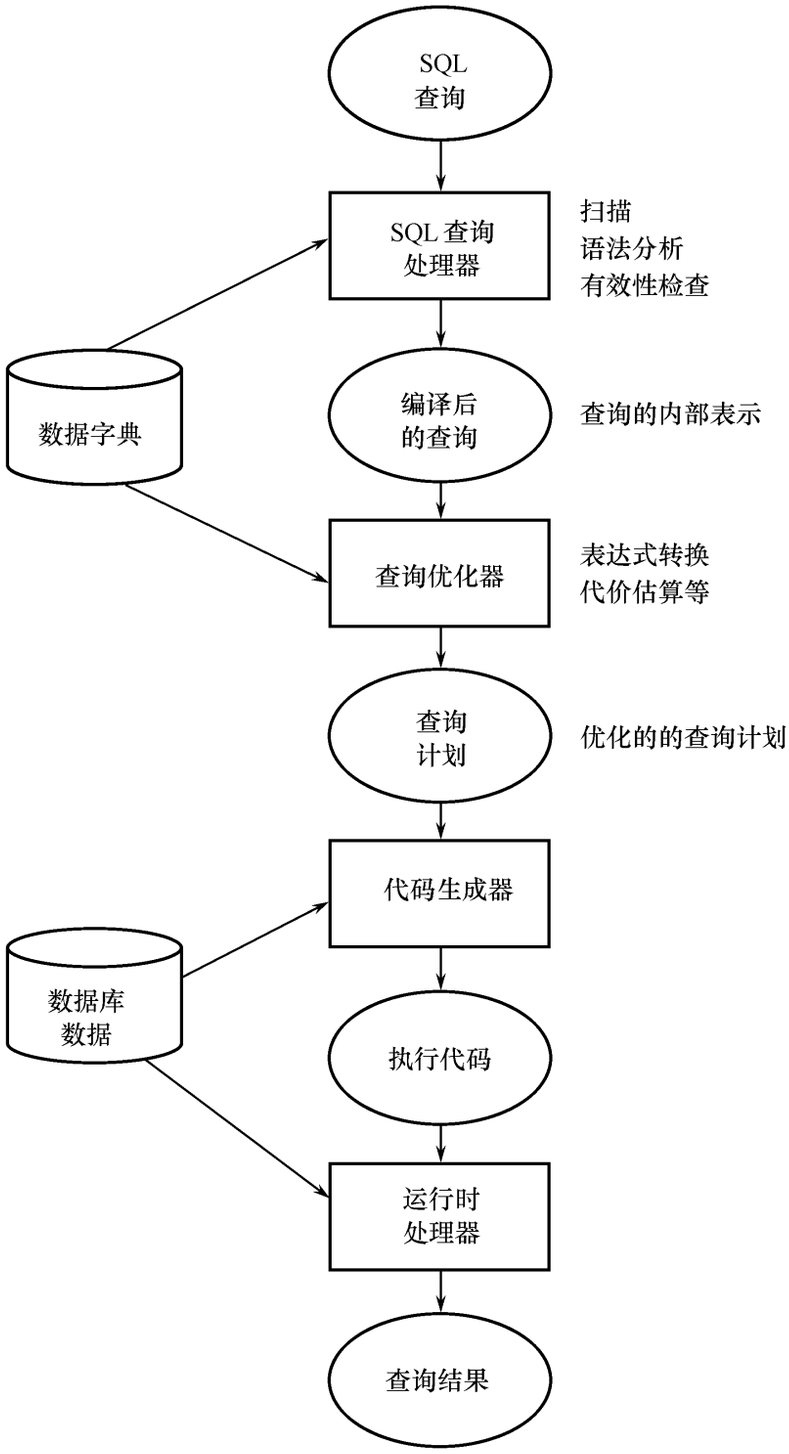
语法分析程序对查询进行语法检查,判断查询表达式是否符合查询语言的语法规则。另外,还必须检查查询的有效性,即根据被查询数据库的模式检查所有的属性名和关系名是否有效,属性名和关系名是否是有语义内涵的名字。随后建立查询的内部表示。

查询的内部表示一般用查询树来表达。

查询优化器采用表达式转换、代价估算等优化方法,选择较优的适当的查询处理策略,并生成查询计划。

代码生成器的任务是生成执行这个查询计划的代码。

运行时处理器则负责执行查询代码,并生成查询结果。如果运行时发生了错误,运行时处理器将返回一条错误消息。


SQL
查询
SQL 查询
处理器
扫描
语法分析
有效性检查
数据字典
编译后
的查询
查询的内部表示
查询优化器
表达式转换
代价估算等
查询
计划
优化的的查询计划
代码生成器
数据库
数据
执行代码
运行时
处理器
查询结果

第五章　关系数据理论

第五章讲解关系数据理论。这是关系数据库的又一个重点。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,关系数据理论是我们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,有的书把它放在数据库设计部分介绍以强调它对数据库设计的指导作用。

一、基本知识点

本章讲解关系数据理论,内容理论性较强,分为基本要求部分(《概论》5.1～5.3)和高级部分(《概论》5.4)。前者是计算机大学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:四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:各个级别范式的关系及其证明。

二、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) 要从属性间实际存在的语义来确定他们之间的函数依赖,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 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 $X \rightarrow Y$,$(Y \setminus X)$,$Y \not\rightarrow X$,$Y \rightarrow Z$,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(Candidate key)。若候选码多于一个,则选定其中的一个为主码(Primary key)。

解析

1) 这里我们用函数依赖来严格定义码的概念。在第二章中我们只是描述性地定义码(可以复习 2.2.1):若关系中的某一属性组的值能惟一地标识一个元组,则称该属性组为候选码(Candidate key)。

2) 因为码有了严格定义,在学习了《概论》5.3 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(Foreign key),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∈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∈1NF,并且每一个非主属性都完全函数依赖于 R 的码,则 R∈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 \ Y)使得 X→Y,(Y ↛X) Y→Z,成立,则称 R < U, F > ∈3NF。

BCNF:关系模式 R < U, F > ∈1NF。若 X→Y 且 Y \ X 时 X 必含有码,则 R < U, F > ∈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 4NF BCNF 3NF 2NF 1NF(《概论》上图 5.2)。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→→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 > ∈1NF,如果对于 R 的每个非平凡多值依赖 X→→Y(Y \ X), X 都含有码,则称 R < U, F > ∈4NF。

解析

对于多值依赖的定义有多种。《概论》上定义 5.9 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。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,有没有全码存在?

答

关系模式: 学生 S(S#,SN,SB,DN,C#,SA)

班级 C(C#,CS,DN,CNUM,CDATE)

系 D(D#,DN,DA,DNUM)

学会 P(PN,DATE1,PA,PNUM)

学生 - 学会 SP(S#,PN,DATE2)

其中,S#—学号,SN—姓名,SB—出生年月,SA—宿舍区

C#—班号,CS—专业名,CNUM—班级人数,CDATE—入校年份

D#—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—学会人数,

DATE2—入会年份

每个关系模式的极小函数依赖集:

S:S#→SN,S#→SB,S#→C#,C#→DN,DN→SA

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

/ * 因为每个专业每年只招一个班 */

D:D#→DN,DN→D#,D#→DA,D#→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:(S#,PN)→DATE2

S 中存在传递函数依赖: S#→DN, S#→SA, C#→SA

/ * 因为 S#→C#,C#→DN, DN→SA */

C 中存在传递函数依赖:C#→DN

/ * 因为 C#→CS,CS→DN */

(S#,PN)→DATE2 和(CS,CDATE)→C# 均为 SP 中的函数依赖,是完全函数依赖。

关系	候选码	外部码	全码
S	S#	C#,DN	无
C	C#,(CS,CDATE)	DN	无

D	D # 和 DN	无	无
P	PN	无	无
SP	(S # ,PN)	S # ,PN	无

解析

读者应该根据题目中给出的有关语义写出关系模式中的数据依赖,有些依赖可以按照实际情况写出,也许题目中并没有明显指出。例如,按照实际情况,系名和系号是一一对应的,因此有 D # →DN,DN→D # 。

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→ Z, X→ Y,则有 X→ YZ

(2) 伪传递规则:由 X→ Y, WY→ Z 有 XW→ Z

(3) 分解规则: X→ Y, Z Y,有 X→ Z

证明

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

(2) 已知 X→ Y,据增广律得 XW→ WY,因为 WY→ Z,所以 XW→ WY→ Z,通过传递律可知 XW→ Z。

(3) 已知 Z Y,根据自反律知 Y→ Z,又因为 X→ Y,所以由传递律可得 X→ 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∈ X, y∈ Y, z∈ 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→→ 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》5.2.7 节中定义 5.9 是等价的。

证明

设 $Y_{xz} = Y_{xz'}$对于每一组(x, z, z′)都成立,现证其能推出定义 5.9 的条件:

设 s、t 是关系 r 中的两个元组, s[X] = t[X],由新定义的条件可知对于每一个 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 5.9 的条件成立;

如果定义 5.9 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 5.9 的条件是等价的,所以新定义和定义 5.9 是等

价的。

5. 试举出3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→→S。由于 C 与 S 的完全对称性,必然有 M→→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→→S,I→→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→→D,R→→P 成立。

*6. 试证明《概论》上给出的关于 FD 和 MVD 公理系统的 A_4,A_6 和 A_8。

证明

A4:若 X→→Y, V W U,则 XW→→YV

设 Z = U - X - Y

已知 X→→Y,设 r 是 R 上的任一关系,s、t∈r,且 t[X] = s[X],则存在元组 p、q∈r,使 p[X] = q[X] = t[X],而 p[Y] = t[Y], p[Z] = s[Z], q[Y] = s[Y], q[Z] = t[Z]。

设 t[XW] = s[XW],我们以上构造的元组 p 和 q,是某部分属性在 s 和 t 上翻转而成,所以 p[W] = q[W],可知 p[XW] = q[XW],同理 p[YV] = t[YV](由 V W 知 t[V] = s[V]), q[YV] = s[YV], p[U - YV - XW] = s[U - YV - XW](因为 U - YV - XW Z), q[U - YV - XW] = t[U - YV - XW]。所以 XW→→YV。

A6:若 X→→Y, Y→→Z 则 X→→Z - Y

由 $Y\rightarrow\rightarrow Z$ 容易证得 $Y\rightarrow\rightarrow Z-Y$。

设 $R_1=U-X-Y$, $R_2=U-Y-Z$, $R_3=U-X-Z+Y$。

已知 $X\rightarrow\rightarrow Y$,设 r 是 R 上的任一关系,s、t∈r,且 $t[X]=s[X]$,则存在元组 p、q∈r,使 $p[X]=q[X]=t[X]$,而 $p[Y]=t[Y]$, $p[R_1]=s[R_1]$, $q[Y]=s[Y]$, $q[R_1]=t[R_1]$。

对元组 t、p,已知 $t[Y]=p[Y]$, $t[X]=p[X]$,由 $Y\rightarrow\rightarrow Z-Y$ 知:存在元组 m∈r,使 $m[Z-Y]=p[Z-Y]$, $m[R_2]=t[R_2]$。因为 $(Z-Y)(R_1$,又 $p[R_1]=s[R_1]$,所以 $m[Z-Y]=s[Z-Y]$。因为元组 p 和 s 在除属性 Y 之外的属性上值相等,所以 $m[R_2]=t[R_2]$,另外元组 m 是由元组 t 和 p 交换某些属性上的值而产生的,而 t 和 p 在属性 X 上值相等,显然 $m[X]=t[X]$,所以 $m[U-(Z-Y)]=t[U-(Z-Y)]$,即 $m[R_3]=t[R_3]$。

对元组 s、q,同理可知 $s[Y]=q[Y]$,存在元组 n,使 $n[Z-Y]=t[Z-Y]$,即 $n[R_3]=s[R_3]$。

综上所述,对 t、s∈r, $t[X]=s[X]$,存在元组 m、n∈r,使 $m[X]=n[X]=t[X]$,而 $m[Z-Y]=s[Z-Y]$, $m[R_3]=t[R_3]$, $n[Z-Y]=t[Z-Y]$, $n[R_3]=s[R_3]$。

A8:若 $X\rightarrow\rightarrow Y$, $W\rightarrow Z$, $W\cap Y=\Phi$, Z Y,则 $X\rightarrow Z$。

设 r 是 R 上的任一关系,对任意 s、t∈r,若 $t[X]=s[X]$,设 $R_1=U-X-Y$,则根据 $X\rightarrow\rightarrow Y$ 知:存在元组 p、q∈r,使 $p[X]=q[X]=t[X]$,而 $p[Y]=t[Y]$, $p[R_1]=s[R_1]$, $q[Y]=s[Y]$, $q[R_1]=t[R_1]$。因为 $W\cap Y=\Phi$,所以 $s[W]=p[W]$,又 $W\rightarrow Z$,所以 $s[Z]=p[Z]$;因为 Z Y,且 $p[Y]=t[Y]$,所以 $p[Z]=t[Z]$;所以可得 $t[Z]=s[Z]$,即 $X\rightarrow Z$。

*7. 设关系模式为 R(U,F), X,Y 为属性集, X,Y U。

证明: (1) X X_F^+

(2) $(X_F^+)_F^+=X_F^+$

(3) 若 X Y 则 X_F^+ Y_F^+

(4) $U_F^+=U$

解析

1. 要证明 $(X_F^+)_F^+=X_F^+$,只要证明 X_F^+ $(X_F^+)_F^+$,并且 $(X_F^+)_F^+$ X_F^+。

而 X_F^+ $(X_F^+)_F^+$ 是显然的,因此只要证明 $(X_F^+)_F^+$ X_F^+。

2. 这里的证明要用集合论的基本知识,读者应该复习一下有关集合论中的有关概念和证明方法。

证明

(1) 因为 $X\rightarrow X$,所以 X X_F^+ (根据 X_F^+ 的定义)。

(2) 下面求证$(X_F^+)_F^+ \quad X_F^+$。

任意 $A\in(X_F^+)_F^+$,(由题意知)存在 $B\in X_F^+$,使 $B\rightarrow A$ 能由 F 根据 Armstrong 公理导出,而从 $B\in X_F^+$ 可知 $X\rightarrow B$ 能由 F 根据 Armstrong 公理导出,根据公理中的传递律可知 $X\rightarrow A$ 能由 F 根据 Armstrong 公理导出,所以 $A\in X_F^+$,因此$(X_F^+)_F^+ \quad X_F^+$。

所以$(X_F^+)_F^+ = X_F^+$。

(3) 对任意 $A\in X_F^+$,可知 $X\rightarrow A$ 能由 F 根据 Armstrong 公理导出,因为 X Y,由自反律可以得 $Y\rightarrow X$,由传递律得 $Y\rightarrow A$,所以 $A\in Y_F^+$。$X_F^+ \quad Y_F^+$ 得证。

(4) 下面证明 $U_F^+ \quad U$,即证 U 由 F 据 Armstrong 公理推出的集合仍属于 U。

解析

要证明 $U_F^+ = U$,只要证明 $U \quad U_F^+$ 并且 $U_F^+ \quad U$。$U \quad U_F^+$ 是显然的。

证明

自反律: $Y \quad U$, $U\rightarrow Y$ 为 F 所蕴含,显然 U 由 F 据 Armstrong 公理的自反律推出的 Y 仍属于 U;

增广律: $U\rightarrow Y$ 为 F 所蕴含,且 $Z \quad U$,则 $UZ\rightarrow YZ$ 为 F 所蕴含, $YZ \quad U$;

传递律: $U\rightarrow Y$ 和 $Y\rightarrow Z$ 都为 F 所蕴含,则 $U\rightarrow Z$ 为 F 所蕴含, $Z \quad U$。

*8. 设关系模式为 R(U, F),若 $X_F^+ = X$,则称 X 相对于 F 是饱和的。定义饱和集 $\Phi_F = \{X \mid X = X_F^+\}$, 试证明 $\Phi_F = \{X_F^+ \mid X \quad U\}$。

解析

(1) 读者首先要理解饱和集定义的含义:饱和集 Φ_F 是这样一个集合,集合中的元素 X 都要满足条件 $X = X_F^+$

(2) 现在要证明 $\Phi_F = \{X_F^+ \mid X \quad U\}$,即要证明 Φ_F 中的元素是由 U 中所有属性 X 的 X_F^+ 组成。弄清这两点,下面的证明就容易了。

(3) 要证明 $\Phi_F = \{X_F^+ \mid X \quad U\}$,根据集合论,只要证明 $\Phi_F \quad \{X_F^+ \mid X \quad U\}$,并且$\{X_F^+ \mid X \quad U\} \quad \Phi_F$。

证明

(1) 证$\Phi_F \quad \{X_F^+ \mid X \quad U\}$。

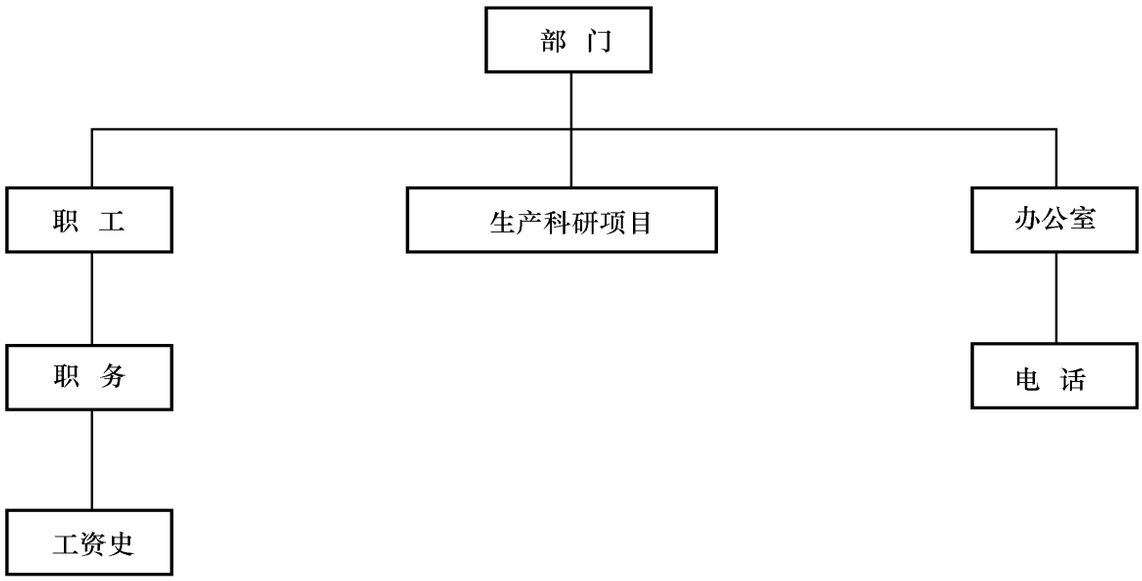
对任意 $A\in\Phi_F$,由已知条件得 $A = A_F^+$,因为 $A \quad U$, $A = A_F^+$,所以 $A\in\{X_F^+ \mid X \quad U\}$。

(2) 证 $\{X_F^+ \mid X \quad U\} \quad \Phi_F$。

对任意 $A\in\{A_F^+ \mid A \quad U\}$,因为$(A_F^+)_F^+ = A_F^+$(见习题 7),令 $B = A_F^+$,有 $B_F^+ = B$ 所以 $B\in\Phi_F$即 $A_F^+ \in\Phi_F$, $A\in\Phi_F$

得证。

9. 《概论》上图 5.12 表示一个公司各部门的层次结构,重画如下。



对每个部门,数据库中包含部门号(惟一的)D#、预算费(BUGET)以及此部门领导人员的职工号 E#(惟一的)信息。

对每一个部门,还存有关于此部门的全部职工、生产与科研项目以及办公室的信息。

职工信息包括:职工号、他所参加的生产与科研项目号(J#)、他所在办公室的电话号码(PHONE#)。

生产科研项目包含:项目号(惟一的)、预算费。

办公室信息包含办公室房间号(惟一的)、面积。

对每个职工,数据库中有他曾担任过的职务以及担任某一职务时的工资历史。

对每个办公室包含此办公室中全部电话号码的信息。

请给出你认为合理的数据依赖,把这个层次结构转换成一组规范化的关系。

提示:此题可分步完成,第一步先转换成一组 1NF 的关系,然后逐步转换为 2NF,3NF,BCNF。

答

(1) 首先画出一些重要的函数依赖,所有这些函数依赖都是根据习题的文字说明和语义假设导出。

语义假设如下:

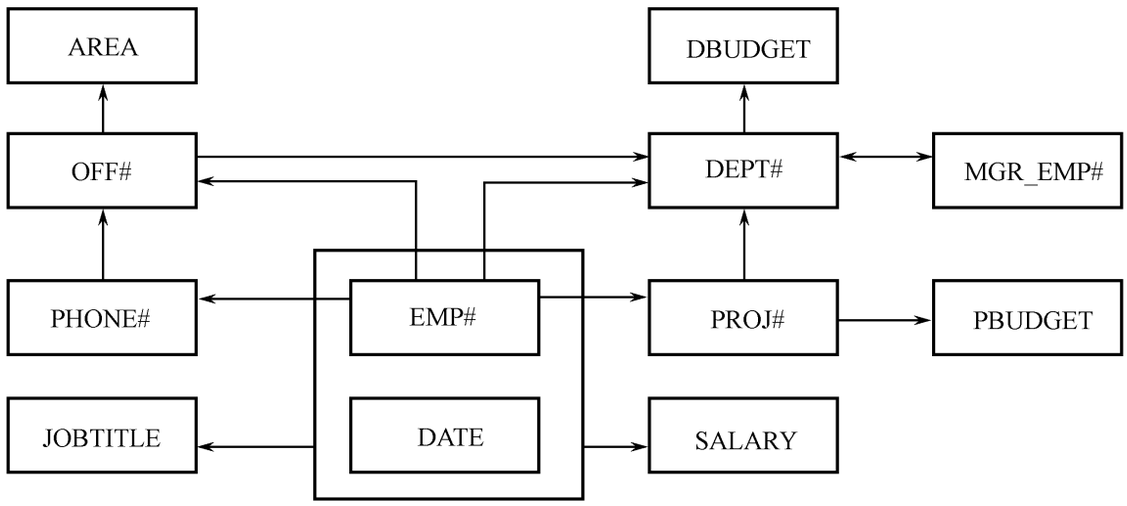
1) 一个职工不能同时成为多个部门的领导人;

2) 一个职工不能同时在多个部门就职;

3) 一个职工不能同时参加多个生产项目;

4) 一个职工不能同时在两个不同的办公室办公;

5) 一个职工不能同时拥有两部或两部以上的电话;



6) 一个生产项目不能同时分配给多个部门;

7) 一个办公室不能同时分配给多个部门;

8) 部门号、职工号、项目号、办公室号码及电话号码是全局惟一的。

(2) 先按照图 5.12 设计一组关系模式,它们都是属于 1NF 的。

DEPT(DEPT#,DBUDGET,MGR_EMP#)

PRIMARY KEY(DEPT#)

DEPT# 和 MGR_EMP# 都是侯选码,把 DEPT# 作为主码。

F={ DEPT# → DBUDGET, DEPT# → MGR_EMP#, MGR_EMP# → DEPT# }

EMP1(EMP#,DEPT#,PROJ#,OFF#,PHONE#)

PRIMARY KEY(EMP#)

F={EMP# → DEPT#,EMP# → PROJ#,EMP# → OFF#,EMP# → PHONE#, PHONE# → OFF#,OFF# → DEPT#,PROJ# → DEPT#}

JOB(EMP#,JOBTITLE)

PRIMARY KEY(EMP#,JOBTITLE)

F={ EMP#,JOBTITLE → EMP#,EMP#,JOBTITLE → JOBTITLE }

SALHIST(EMP#,JOBTITLE,DATE,SALARY)

PRIMARY KEY(EMP#,DATE)

F={ EMP#,DATE → JOBTITLE,EMP#,DATE → SALARY }

PROJ(PROJ#,DEPT#,PBUDGET)

PRIMARY KEY(PROJ#)

F={ PROJ# →DEPT#,PROJ# → PBUDGET }

OFFICE (OFF#,DEPT#,AREA)
PRIMARY KEY (OFF#)
F={ OFF# →DEPT#,OFF# →AREA }

PHONE (PHONE#,OFF#)
PRIMARY KEY (PHONE#)
F={ PHONE# → OFF# }

(3) 现在来分析一下这7个关系模式,发现:SALHIST (EMP#,DATE,JOBTITLE,SALARY)的属性包含了JOB(EMP#,JOBTITLE)的属性,所以JOB(EMP#,JOBTITLE)可以消去。

EMP1中OFF#和DEPT#都传递函数依赖于主码(EMP#)。OFF#通过PHONE#,DEPT#通过PROJ#或OFF#(然后通过PHONE#)传递依赖于{EMP#),所以可以把EMP1(EMP#,DEPT#,PROJ#,OFF#,PHONE#)分解成下面4个3NF的关系模式:

EMP (EMP#,PROJ#,PHONE#)
PRIMARY KEY (EMP#)
X (PHONE#,OFF#)
PRIMARY KEY (PHONE#)
Y (PROJ#,DEPT#)
PRIMARY KEY (PROJ#)
Z (OFF#, DEPT#)
PRIMARY KEY (OFF#)

然而,X就是PHONE,Y是PROJ的投影,Z是OFFICE的投影,所以X、Y、Z都可以消去。

最后可以得到下面6个关系模式,所有这些关系模式都是属于3NF的,进一步发现他们也是BCNF的。

DEPT(DEPT#,DBUDGET,MGR_EMP#)
PRIMARY KEY (MGR_EMP#)
EMP (EMP#,PROJ#,PHONE#)
PRIMARY KEY (EMP#)
SALHIST (EMP#,DATE,JOBTITLE,SALARY)
PRIMARY KEY (EMP,DATE)
PROJ (PROJ#,DEPT#,PBUDGET)

PRIMARY KEY (PROJ#)

OFFICE (OFF#, DEPT#, AREA)

PRIMARY KEY (OFF#)

PHONE (PHONE#, OFF#)

PRIMARY KEY (PHONE#)

10. 在一个订货系统的数据库中,存有顾客、货物和订货单的信息。

每个顾客包含顾客号 CUST#(惟一的)、收货地址 ADDRESS(一个顾客可有几个地址)、赊购限额 CREDLIM、余额 BAL 以及折扣 DISCOUNT。

每个订货单 ORD# 包含顾客号 CUST#、收货地址 ADDRESS、订货日期 DATE、订货细则 LINE#(每个订货单有若干条),每条订货细则内容为货物号 ITEM# 以及订货数量 QTYORD。

每种货物包含货物号 ITEM#(惟一的)、制造厂商 PLANT#、每个厂商的实际存货量 QTYOH、规定的最低存货量 DANGER 和货物描述 DESCN。

由于处理上的要求,每个订货单 ORD# 的每一订货细则 LINE# 中还应有一个未发货量 QTYOUT(此值初始时为订货数量,随着发货将减为零)。

为这些数据设计一个数据库,如第 9 题那样,首先给出合理的数据依赖。

答

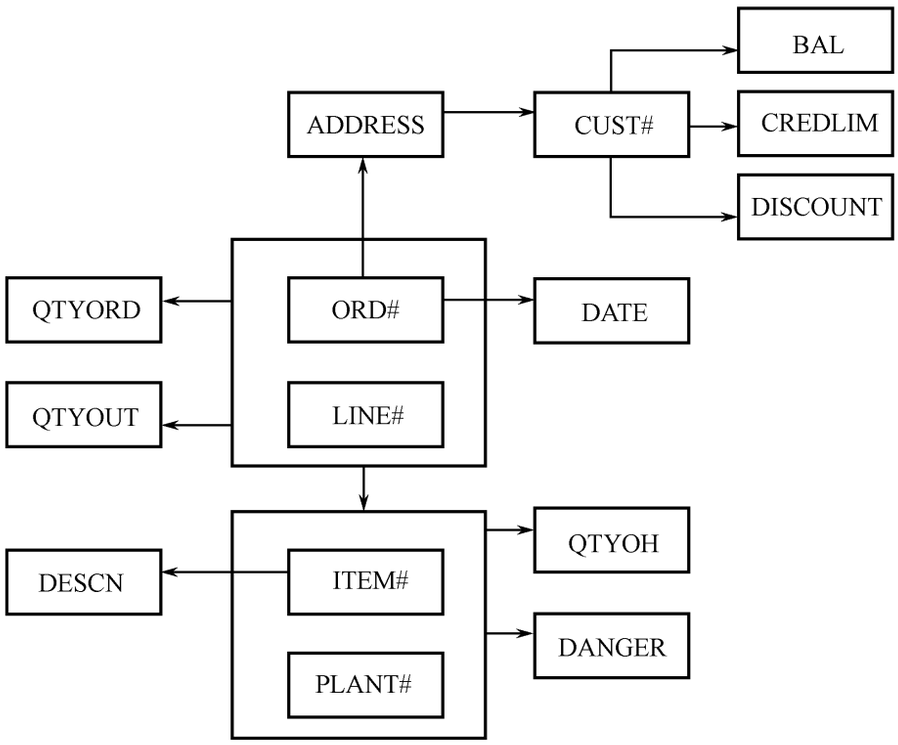
其语义假设如下:

(1) 任何两个顾客的收货地址都不相同;

(2) 每一个订单都有一个惟一的订单号码。

(3) 每个订单的订单细则在这个订单里有一个惟一的编号。

函数依赖图如下:



相应的 BCNF 关系模式如下:

CUST (CUST#, BAL, CREDLIM, DISCOUNT)
PRIMARY KEY (CUST#)
SHIPTO (ADDRESS, CUST#)
PRIMARY KEY (ADDRESS)
ORDHEAD (ORD#, ADDRESS, DATE)
PRIMARY KEY (ORD#)
ORDLINE (ORD#, LINE#, ITEM#, QTYORD, QTYOUT)
PRIMARY KEY (ORD#, LINE#)
ITEM (ITEM#, DESCN)
PRIMARY KEY (ITEM#)
IP (ITEM#, PLANT#, QTYOH, DANGER)
PRIMARY KEY (ITEM#, PLANT#)

11. 设在第 10 题中实际上只有很少量的顾客(例如 1%),却有多个发货地址,由于这些少数的而又不能忽视的情形使得不能按一般的方式来处理问题。你能发现第 10 题答案中的问题吗?能设法改进吗?

答

如果 99% 的顾客只有一个收货地址,则把地址放在与 CUST 不同的关系模式中,在实际处理订货过程时的效率是很低的。因此我们可以对这个问题进行改进。对于每个顾客,指定一个合法收货地址作为主地址,则对于 99% 的顾客,该地址就是他的惟一地址。

关系模式 CUST 的定义修改如下:

CUST (CUST#, ADDRESS, BAL, CREDLIM, DISCOUNT)
PRIMARY KEY (CUST#)

99% 的只有一个收货地址的顾客,则 CUST#(顾客号)→ ADDRESS(地址)。

对于其他 1% 的顾客,建立关系模式 SECOND(代替原来的关系模式 SHIPTO):

SECOND(ADDRESS, CUST#)
PRIMARY KEY (ADDRESS)

这样,CUST 存放主地址,而 SECOND 中存放所有的第二地址(和相应的顾客号),这两个关系变量都是属于 BCNF 的。

该方法具有如下优点:

(1) 对于 99% 的顾客的处理变得简单(当然更有效)了;

(2) 如果输入订单时把收货地址省略了,则可以用主地址作为默认地址。

总的说来,把特殊情况分离开来是个有效的方法,它可以充分利用两者的优

点,既达到简化处理的目的,又使设计的关系模式达到 BCNF。

12 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。

答

(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。√

R(X,Y)如果 X→→ 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→B 在 R 上成立,关系 R(A,B,C)等于其投影 R_1(A,B)和 R_2(A,C)的连接。×

当 A→B 在 R 上成立,关系 R(A,B,C)等于其投影 R_1(A,B)和 R_2(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→→ B 在 R 上成立,关系 R(A,B,C)等于其投影 R_1(A,B)和 R_2(A,C)的连接。(参见《概论》上定理 5.6)

(5) 若 R.A→R.B,R.B→R.C,则 R.A→R.C √

(6) 若 R.A→R.B,R.A→R.C,则 R.A→R.(B, C) √

(7) 若 R.B→R.A,R.C→R.A,则 R.(B, C)→R.A √

(8) 若 R.(B, C)→R.A,则 R.B→R.A,R.C→R.A ×

反例:关系模式 SC(S#,C#,G),(S#,C#)→G,但是 S#↛G,C#↛G。

第六章　数据库设计

第六章讲解数据库设计的方法和步骤。

和许多《教科书》不同,《数据库系统概论》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《教科书》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,并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。关系数据理论是我们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,才能够完成一个实际部门的数据库应用系统设计的全过程。

一、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,缺乏解决实际问题的能力,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。

二、习题解答和解析

1. 试述数据库设计过程。

答

这里只概要列出数据库设计过程的六个阶段:

(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、运行和维护。

设计一个完善的数据库应用系统往往是上述六个阶段的不断反复。

解析

希望读者能够认真阅读《概论》6.1 的内容,了解并掌握数据库设计过程(P205～P206)。

2. 试述数据库设计过程各个阶段上的设计描述。

答

各阶段的设计要点如下:

(1) 需求分析:准确了解与分析用户需求(包括数据与处理)。

(2) 概念结构设计:通过对用户需求进行综合、归纳与抽象,形成一个独立于具体 DBMS 的概念模型。

(3) 逻辑结构设计:将概念结构转换为某个 DBMS 所支持的数据模型,并对其进行优化。

(4) 数据库物理设计:为逻辑数据模型选取一个最适合应用环境的物理结构(包括存储结构和存取方法)。

(5) 数据库实施:设计人员运用 DBMS 提供的数据语言、工具及宿主语言,根据逻辑设计和物理设计的结果建立数据库,编制与调试应用程序,组织数据入库,并进行试运行。

(6) 数据库运行和维护:在数据库系统运行过程中对其进行评价、调整与修改。

解析

这是进一步了解数据库设计的具体内容。设计描述是指在各个阶段体现设计内容,描述设计结果的各种文档、程序。读者可以参考《概论》上图 6.3(P208)。

3. 试述数据库设计过程中结构设计部分形成的数据库模式。

答

数据库结构设计的不同阶段形成数据库的各级模式,即:

(1) 在概念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》上图 6.4(P209)。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。

4. 试述数据库设计的特点。

答

数据库设计既是一项涉及多学科的综合性技术又是一项庞大的工程项目。其主要特点有:

(1) 数据库建设是硬件、软件和干件(技术与管理的界面)的结合。

(2) 从软件设计的技术角度看,数据库设计应该和应用系统设计相结合,也就是说,整个设计过程中要把结构(数据)设计和行为(处理)设计密切结合起来。

详细的可以参考《概论》上 6.1.2(P204)。

5. 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。

详细的可以参考《概论》上 6.2(P209～P210)。

6. 数据字典的内容和作用是什么？

答

数据字典是系统中各类数据描述的集合。数据字典的内容通常包括:

(1) 数据项;

(2) 数据结构;

(3) 数据流;

(4) 数据存储;

(5) 处理过程五个部分。

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和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、完善。

(详细参考《概论》上 6.2.3。注意,数据库设计阶段形成的数据字典与第十一章 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)。

7. 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四种:

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。

详细参考《概论》上 6.3(P213～P216)。

8. 什么叫数据抽象?试举例说明。

答

数据抽象是对实际的人、物、事和概念进行人为处理,抽取所关心的共同特性,忽略非本质的细节,并把这些特性用各种概念精确地加以描述,这些概念组成了某种模型。

如"分类"这种抽象是:定义某一类概念作为现实世界中一组对象的类型。这些对象具有某些共同的特性和行为。它抽象了对象值和型之间的"is member of"的语义。在 E-R 模型中,实体型就是这种抽象。例如在学校环境中,李英是老师,表示李英是教师类型中的一员,则教师是实体型,李英是教师实体型中的一个实体值,具有教师共同的特性和行为:在某个系某个专业教学,讲授某些课程,从事某个方向的科研。

9. 试述数据库概念结构设计的重要性和设计步骤。

答

重要性:数据库概念设计是整个数据库设计的关键,将在需求分析阶段所得到的应用需求首先抽象为概念结构,以此作为各种数据模型的共同基础,从而能更好地、更准确地用某一 DBMS 实现这些需求。

设计步骤:概念结构的设计方法有多种,其中最经常采用的策略是自底向上方法,该方法的设计步骤通常分为两步:第 1 步是抽象数据并设计局部视图,第 2 步是集成局部视图,得到全局的概念结构(如《概论》图 6.9 所示,P216)。

10. 什么是 E-R 图?构成 E-R 图的基本要素是什么?

答

E-R 图为实体 - 联系图,提供了表示实体型、属性和联系的方法,用来描述现实世界的概念模型。

构成 E-R 图的基本要素是实体型、属性和联系,其表示方法为:

(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)。

解析

E-R 图的概念是在第一章中讲解的,读者可以复习《概论》1.2.2 概念模型的内容(P15～P20)。

11. 为什么要视图集成?视图集成的方法是什么?

答

在对数据库系统进行概念结构设计时一般采用自底向上的设计方法,把繁杂的大系统分解子系统。首先设计各个子系统的局部视图,然后通过视图集成的方式将各子系统有机地融合起来,综合成一个系统的总视图。这样,设计清晰,由简到繁。由于数据库系统是从整体角度看待和描述数据的,因此数据不再面向某个应用而是整个系统。因此必须进行视图集成,使得数据库能被全系统的多个用户、多个应用共享使用。

一般说来,视图集成可以有两种方式:

(1) 多个分 E-R 图一次集成,如《概论》图 6.25(a)所示(P224);

(2) 逐步集成,用累加的方式一次集成两个分 E-R 图,如《概论》图 6.25(b)所示。

无论采用哪种方式,每次集成局部 E-R 图时都需要分两步走:

(1) 合并,解决各分 E-R 图之间的冲突,将各分 E-R 图合并起来生成初步 E-R图;

(2) 修改和重构,消除不必要的冗余,生成基本 E-R 图。

12. 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(《概论》图 6.31):

(1) 将概念结构转换为一般的关系、网状、层次模型;

(2) 将转换来的关系、网状、层次模型向特定 DBMS 支持下的数据模型转换;

(3) 对数据模型进行优化。

13. 试述把 E-R 图转换为 DBTG 模型和关系模型的转换规则。

答

E-R 图向 DBTG 模型的转换规则:

(1) 每个实体型转换为记录型,实体的属性转换为记录的数据项;

(2) 实体型之间 1:n(n≥1)的联系转换为一个系,没有任何联系的实体型转换为奇异系;

(3) K(K≥2)个实体型之间多对多的联系,引入一个连结记录,形成 K 个实体型和连结记录之间的 K 个系。连结记录的属性由诸首记录的码及联系属性所组成;

(4) 同一实体型内的 1:n,n:m 联系,引入连结记录,转换为两个系。

解析

根据我国实际情况,网状,层次数据库系统已很少使用,因此《概论》第三版把它们删去了,有关的主要概念放在第一章数据模型中介绍。对于 DBTG 模型的许多概念也介绍得很简单。本题的内容已经超出了书上的内容,读者只要了解就可以了。但是,下面 E-R 图向关系模型的转换规则要求同学必须掌握,并且能够举一反三。

答

E-R 图向关系模型的转换规则:

一个实体型转换为一个关系模式。实体的属性就是关系的属性,实体的码就是关系的码。

对于实体间的联系则有以下不同的情况:

(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。如果转换为一个独立的关系模式,则与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性,每个实体的码均是该关系的候选码。如果与某一端实体对应的关系模式合并,则需要在该关系模式的属性中加入另一个关系模式的码和联系本身的属性。

(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。如果转换为一个独立的关系模式,则与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性,而关系的码为 n 端实体的码。

(3) 一个 m:n 联系转换为一个关系模式。与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性,各实体码的组合组成该关系的码,或码的一部分。

(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。与该多元联系相连的各实体的码以及联系本身的属性均转换为关系的属性,而关系的码为各实体码的组合。

(5) 具有相同码的关系模式可以合并。

*14. 你能给出由 E-R 图转换为 IMS 模型的转换规则吗?

答

E-R 图向 IMS 模型的转换规则:

(1) 每个实体型转换为记录型,实体的属性转换为记录的数据项;

(2) 实体型之间 1:n(n≥1)的联系转换记录型之间的有向边;

(3) 实体型之间 m:n(m>1,n>1)的联系则分解成一对多联系,再根据(2)转换;

(4) K(K≥2)个实体型之间多对多的联系,可先转换成多对两个实体型之间的联系,再根据(3)转换。

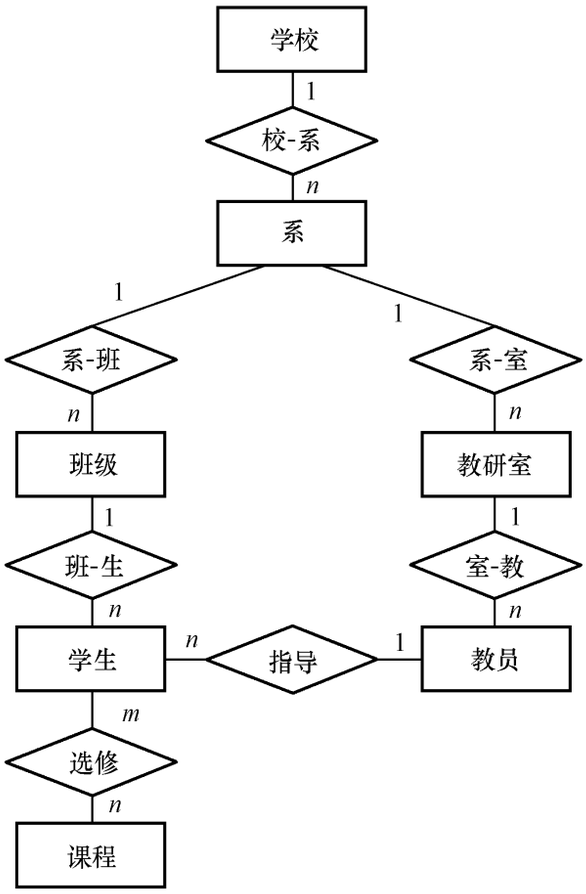
解析

IMS是IBM公司的层次数据库管理系统。IMS模型是层次模型。E-R图向IMS模型转换的另一种方法是,先把E-R图转换为网状模型,再利用IMS逻辑数据库LDB的概念来表示网状模型。详细方法这里从略。

15. 试把第一章习题12和习题13中的E-R图转换为DBTG模型、IMS模型、关系模型。

答

下面是第一章习题12的E-R图:



各实体的属性为:(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

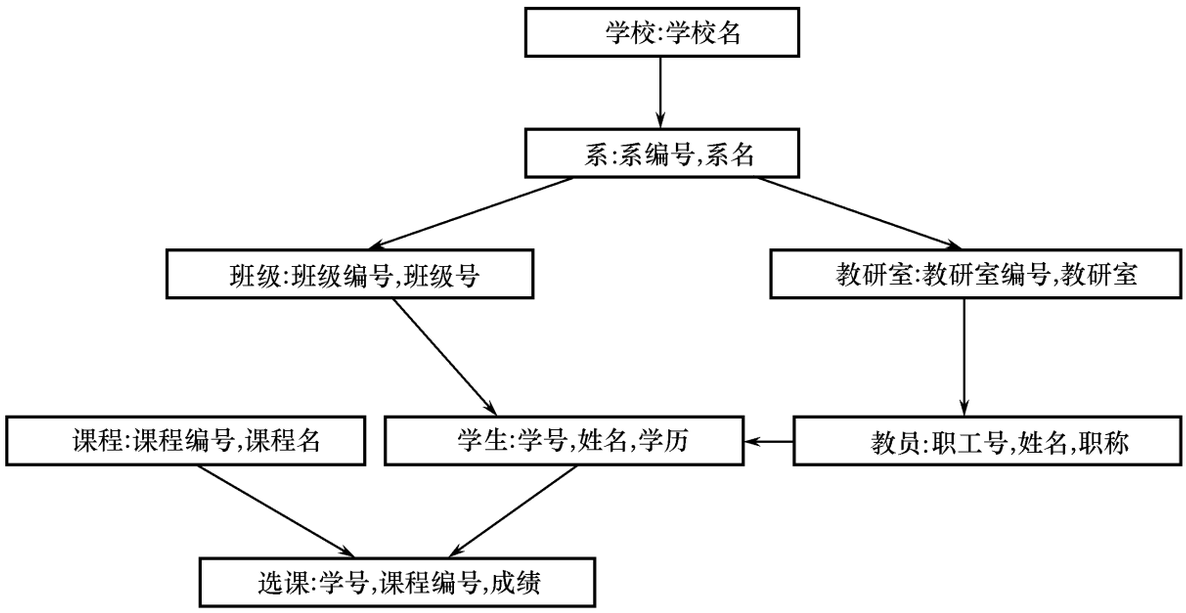
教员:职工号,姓名,职称

各联系的属性为:

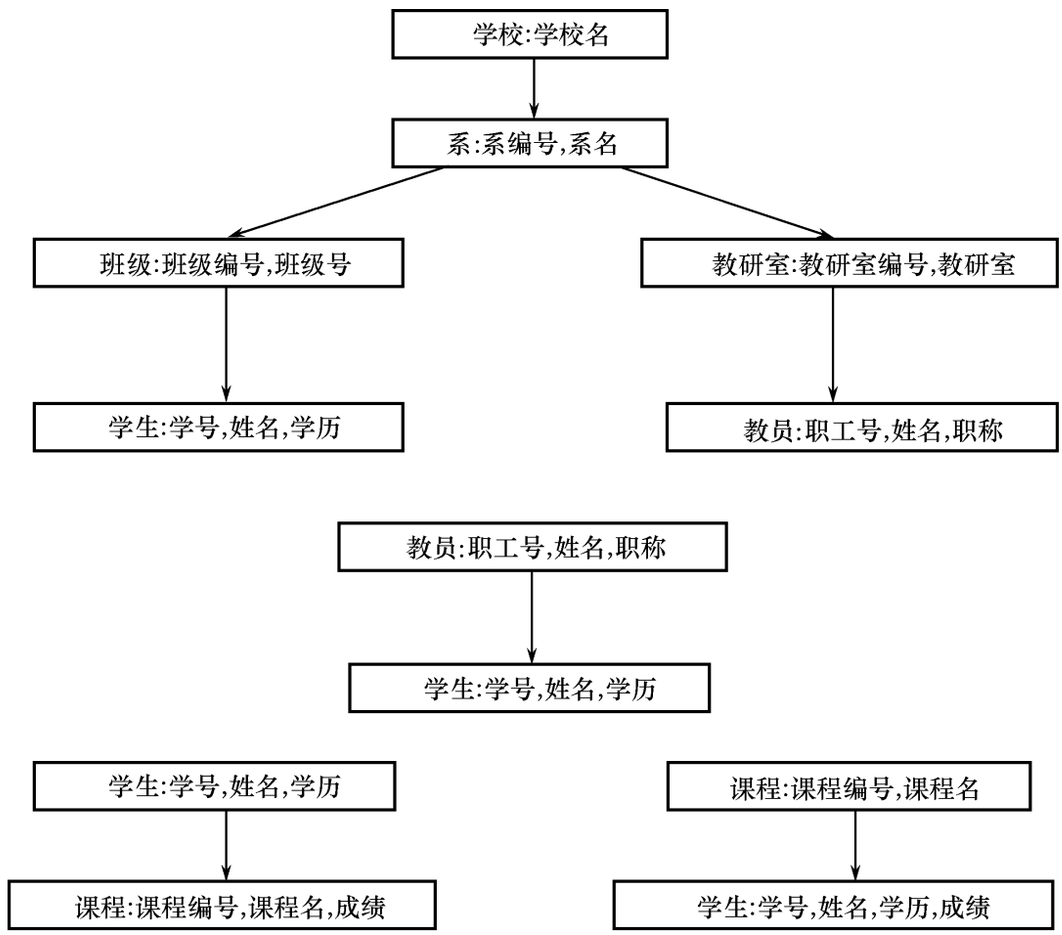
选修课:成绩

其他联系无属性。

该E-R图的DBTG模型为:



其 IMS 模型为:



其关系模型为:

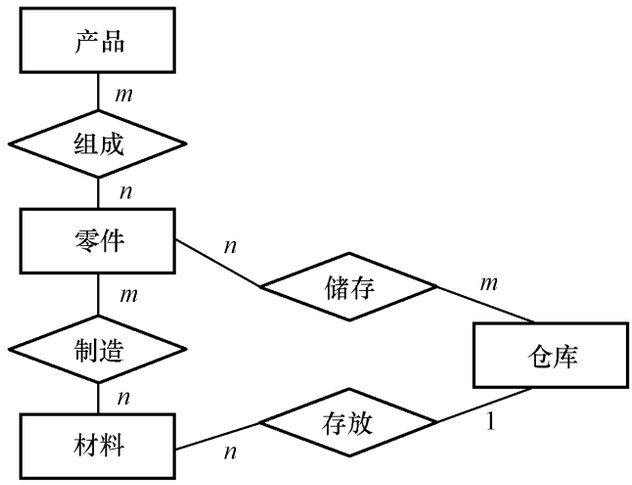
系(<u>系编号</u>,系名,学校名)

班级(<u>班级编号</u>,班级名,系编号)

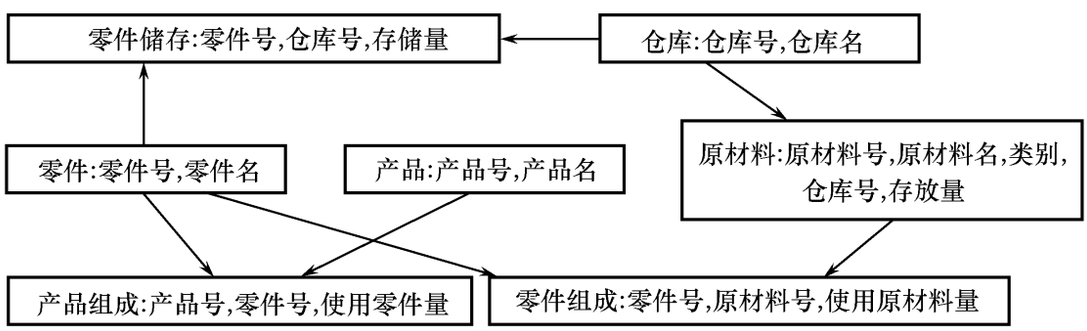
教研室(<u>教研室编号</u>,教研室,系编号)

学生(<u>学号</u>,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 13 的 E-R 图:



各实体的属性为:(方便起见,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
其 DBTG 模型:



对应的 IMS 模型从略。
对应的关系模型为(其中有下横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)

16. 试述逻辑设计阶段中运用 LRA 方法优化模型的方法和步骤。

答

(从略,此题不作要求)

17. 试用规范化理论中有关范式的概念分析习题 15 中你设计的关系模型中各个关系模式的码,它们属于第几范式?会产生什么更新异常?

答

习题 15 中设计的两个关系数据库的各个关系模式的码都用下划线注明,这些关系模式都只有一个码,且都是惟一决定的因素,所以都属于 BCNF。不会产生更新异常现象。

18. 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的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》上 P231～232 数据模型的优化)。

19. 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,评价的重点是时间效率和空间效率。

详细参考《概论》上 6.5.1。

20. 你能给出关系数据库物理设计的主要内容吗?例如 Oracle 数据库物理设计的内容。

答

关系数据库物理设计的内容主要包括:

(1) 为关系模式选择存取方法;

(2) 设计关系、索引等数据库文件的物理存储结构。

详细参考《概论》上 6.5.2 和 6.5.3。

有关 Oracle 数据库物理设计的内容请参考其技术资料,根据上述内容进行总结。

21. 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工作。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。

详细参考《概论》上 6.6.1。

22. 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。

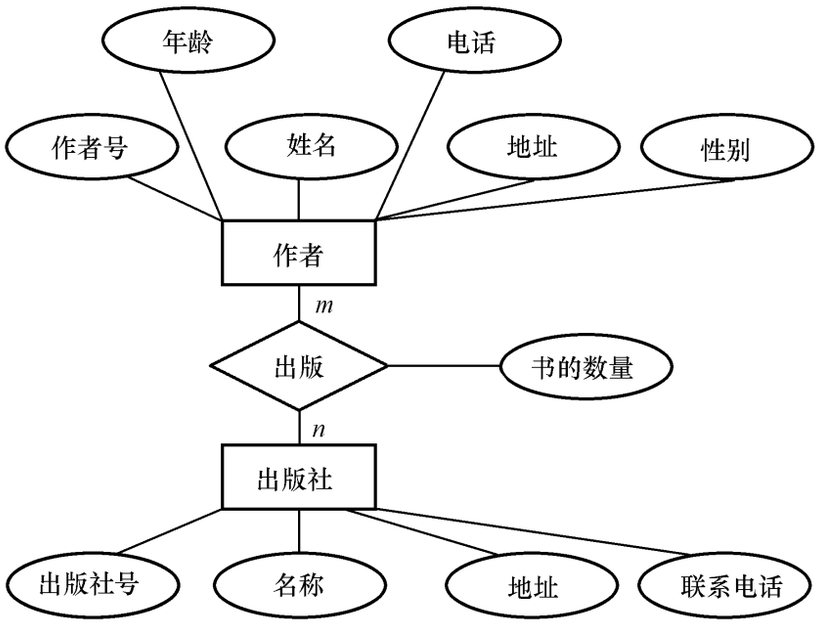
进行数据库的再组织和重构造的原因:数据库运行一段时间后,由于记录不断增、删、改,会使数据库的物理存储情况变坏,降低了数据的存取效率,数据库性能下降,这时 DBA 就要对数据库进行重组织。DBMS 一般都提供用于数据重组织的实用程序。

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。

23. 现有一局部应用,包括两个实体:“出版社”和“作者”,这两个实体是多对多的联系,请读者自己设计适当的属性,画出 E-R 图,再将其转换为关系模型(包括关系名、属性名、码和完整性约束条件)。

答

E-R 图为:



关系模型为:

作者(<u>作者号</u>,姓名,年龄,性别,电话,地址)

出版社(<u>出版社号</u>,名称,地址,联系电话)

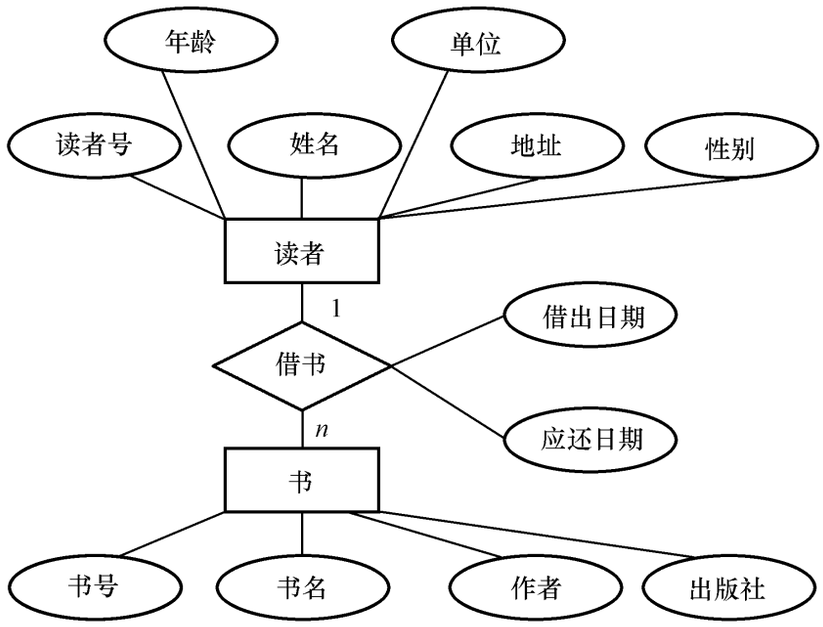
出版(<u>作者号,出版社号</u>,书的数量)

出版关系的主码<u>作者号,出版社号</u>分别参照作者关系的主码<u>作者号</u>和出版社关系的主码<u>出版社号</u>。

24. 请设计一个图书馆数据库,此数据库中对每个借阅者保存读者记录,包括:读者号,姓名,地址,性别,年龄,单位。对每本书存有:书号,书名,作者,出版社。对每本被借出的书存有读者号、借出日期和应还日期。要求:给出 E-R 图,再将其转换为关系模型。

答

E-R 图为:



关系模型为:

读者(读者号,姓名,地址,性别,年龄,单位)

书(书号,书名,作者,出版社)

借书(读者号,书号,借出日期,应还日期)

三、大 作 业

完成一个实际部门的数据库应用系统设计全过程。

包括:需求调查、数据库设计、数据库建立、数据输入、应用系统设计和开发、用户界面的设计和实现等。

要求:

1. 人员:5～6 人。

2. 分工:每个人担任不同的角色,包括系统分析人员、系统设计人员、数据库设计人员、应用系统开发人员、测试人员等。分工协作,共同完成设计和开发任务,从而培养团队精神。

3. 选择一个合适的 DBMS 产品或者使用学校提供的 DBMS 产品,选择合适的应用开发工具,按照设计的结果建立数据库、开发应用系统、输入数据、调试运行你们的系统。

4. 要求写出完整的实验报告,包括:

需求调查报告、系统分析报告、数据库设计报告、应用系统设计报告、数据库实施计划、系统测试计划、系统测试报告、用户使用手册等文档。

5. 向老师和其他小组运行演示开发的数据库应用系统,提交所有文档。

6. 集体讨论、互相学习,指出各自的特点和不足,交流开发过程中的收获和体会。

第七章　数据库恢复技术

《概论》从第七章起用四章的篇幅讨论 DBMS 中重要的事务处理技术、数据库完整性和安全性问题。事务处理技术主要包括数据库恢复技术和并发控制技术。本章讨论数据库恢复的概念和常用技术。

一、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,他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。什么是数据库镜像功能。

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的策略和方法。

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)和 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中的 DBMS 产品提供的恢复技术、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。

二、习题解答和解析

1. 试述事务的概念及事务的 4 个特性。

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。

2. 为什么事务非正常结束时会影响数据库数据的正确性,请列举一例说明之。

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,这时数据库就处于一种不正确的状态,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. 数据库中为什么要有恢复子系统?它的功能是什么?

答

因为计算机系统中硬件的故障、软件的错误、操作员的失误以及恶意的破坏是不可避免的,这些故障轻则造成运行事务非正常中断,影响数据库中数据的正

确性,重则破坏数据库,使数据库中全部或部分数据丢失,因此必须要有恢复子系统。

恢复子系统的功能是:把数据库从错误状态恢复到某一已知的正确状态(亦称为一致状态或完整状态)。

4. 数据库运行中可能产生的故障有哪几类?哪些故障影响事务的正常执行?哪些故障破坏数据库数据?

答

数据库系统中可能发生各种各样的故障,大致可以分以下几类:

(1) 事务内部的故障;

(2) 系统故障;

(3) 介质故障;

(4) 计算机病毒。

事务故障、系统故障和介质故障影响事务的正常执行;介质故障和计算机病毒破坏数据库数据。

5. 数据库恢复的基本技术有哪些?

答

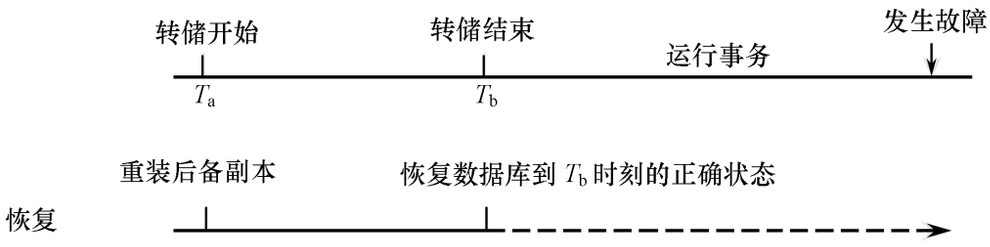
数据转储和登录日志文件是数据库恢复的基本技术。

当系统运行过程中发生故障,利用转储的数据库后备副本和日志文件就可以将数据库恢复到故障前的某个一致性状态。

6. 数据库转储的意义是什么?试比较各种数据转储方法。

答

数据转储是数据库恢复中采用的基本技术。所谓转储即 DBA 定期地将数据库复制到磁带或另一个磁盘上保存起来的过程。当数据库遭到破坏后可以将后备副本重新装入,将数据库恢复到转储时的状态。



静态转储:在系统中无运行事务时进行的转储操作,如上图所示。静态转储简单,但必须等待正运行的用户事务结束才能进行。同样,新的事务必须等待转储结束才能执行。显然,这会降低数据库的可用性。

动态转储:指转储期间允许对数据库进行存取或修改。动态转储可克服静态转储的缺点,它不用等待正在运行的用户事务结束,也不会影响新事务的运

行。但是,转储结束时后援副本上的数据并不能保证正确有效。因为转储期间运行的事务可能修改了某些数据,使得后援副本上的数据不是数据库的一致版本。

为此,必须把转储期间各事务对数据库的修改活动登记下来,建立日志文件(log file)。这样,后援副本加上日志文件就能得到数据库某一时刻的正确状态。

转储还可以分为海量转储和增量转储两种方式。

海量转储是指每次转储全部数据库。增量转储则指每次只转储上一次转储后更新过的数据。从恢复角度看,使用海量转储得到的后备副本进行恢复一般说来更简单些。但如果数据库很大,事务处理又十分频繁,则增量转储方式更实用更有效。

7. 什么是日志文件?为什么要设立日志文件?

答

(1) 日志文件是用来记录事务对数据库的更新操作的文件。

(2) 设立日志文件的目的是:进行事务故障恢复;进行系统故障恢复;协助后备副本进行介质故障恢复。有关日志文件的具体作用,参考《概论》7.4.2。

8. 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。

9. 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?介质故障恢复?)

答

事务故障的恢复:

事务故障的恢复是由 DBMS 自动完成的,对用户是透明的。

DBMS 执行恢复步骤是:

(1) 反向扫描文件日志(即从最后向前扫描日志文件),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;

(3) 继续反向扫描日志文件,做同样处理;

(4) 如此处理下去,直至读到此事务的开始标记,该事务故障的恢复就完

成了。

答

系统故障的恢复:

系统故障可能会造成数据库处于不一致状态:一是未完成事务对数据库的更新可能已写入数据库;二是已提交事务对数据库的更新可能还留在缓冲区,没来得及写入数据库。因此恢复操作就是要撤销(UNDO)故障发生时未完成的事务,重做(REDO)已完成的事务。

系统的恢复步骤是:

(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) 对撤销队列中的各个事务进行 UNDO 处理。

进行 UNDO 处理的方法是,反向扫描日志文件,对每个 UNDO 事务的更新操作执行逆操作,即将日志记录中"更新前的值"(Before Image)写入数据库。

(3) 对重做队列中的各个事务进行 REDO 处理。

进行 REDO 处理的方法是:正向扫描日志文件,对每个 REDO 事务重新执行日志文件登记的操作。即将日志记录中"更新后的值"(After Image)写入数据库。

解析

在第(1)步中如何找出 REDO 队列和 UNDO 队列?请大家思考一下。

下面给出一个算法:

1) 建立两个事务队列:

① UNDO-LIST:需要执行 undo 操作的事务集合;

② REDO-LIST:需要执行 redo 操作的事务集合。

两个事务队列初始均为空。

2) 从日志文件头开始,正向扫描日志文件:

① 如有新开始(遇到 Begin Transaction)的事务 Ti,把 Ti 暂时放入 UNDO-LIST 队列;

② 如有提交的事务(遇到 End Transaction)Tj,把 Tj 从 UNDO-LIST 队列移到 REDO-LIST 队列;直到日志文件结束。

答

介质故障的恢复:

介质故障是最严重的一种故障。

恢复方法是重装数据库,然后重做已完成的事务。具体过程是:

(1) DBA 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),使数据库恢复到转储时的一致性状态;

(2) DBA 装入转储结束时刻的日志文件副本;

(3) DBA 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。

解析

(1) 假定采用的是静态转储,因此第(1)步装入数据库后备副本便可以了。

(2) 如果采用的是动态转储,第(1)步装入数据库后备副本还不够,还需同时装入转储开始时刻的日志文件副本,经过处理后才能得到正确的数据库后备副本。

(3) 第(2)步重做已完成的事务的算法是:

1) 正向扫描日志文件,找出故障发生前已提交的事务的标识,将其记入重做队列;

2) 再一次正向扫描日志文件,对重做队列中的所有事务进行重做处理。即将日志记录中"更新后的值"写入数据库。

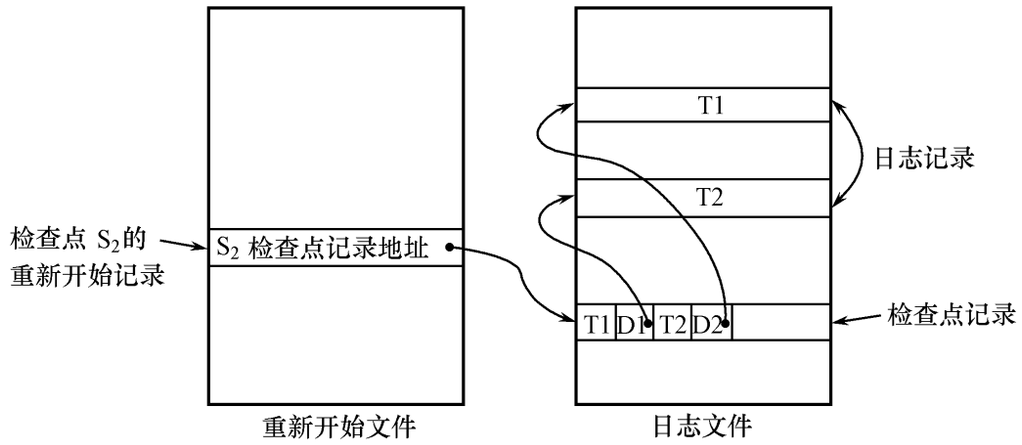
10. 什么是检查点记录?检查点记录包括哪些内容?

答

检查点记录是一类新的日志记录。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(如图中的 T1、T2);

② 这些事务的最近一个日志记录的地址(如图中的 D1、D2)。



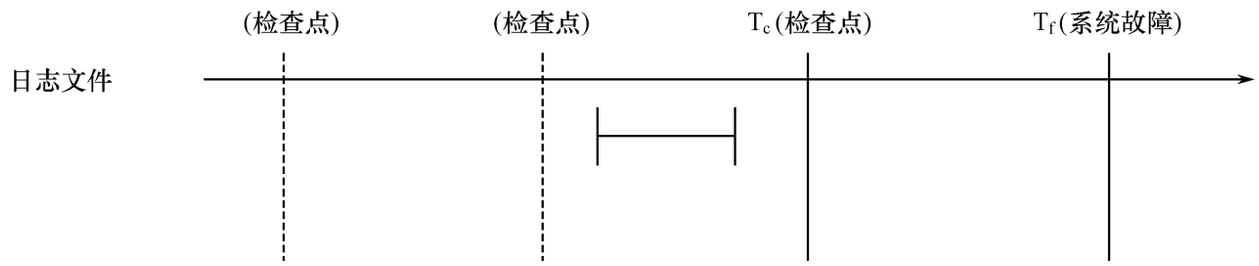
11. 具有检查点的恢复技术有什么优点?试举一个具体的例子加以说明。

答

利用日志技术进行数据库恢复时,恢复子系统必须搜索日志,确定哪些事务需要 REDO,哪些事务需要 UNDO。一般来说,需要检查所有日志记录。这样做有两个问题:一是搜索整个日志将耗费大量的时间;二是很多需要 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_1 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。

12 试述使用检查点方法进行恢复的步骤。

答

(1) 从重新开始文件(见第 11 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:

1) UNDO-LIST: 需要执行 undo 操作的事务集合;

2) 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:

① 如有新开始的事务 T_i,把 T_i 暂时放入 UNDO-LIST 队列;

② 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。

13 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。

数据库镜像的用途有:

一是用于数据库恢复。当出现介质故障时,可由镜像磁盘继续提供使用,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。

二是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它锁进行修改时,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。

*14. 试述你了解的某一个实际的 DBMS 产品中采用的恢复策略。

答

下面简单介绍一下 Oracle 的恢复技术:

Oracle 中恢复机制也采用了转储和登记日志文件两个技术。

Oracle 向 DBA 提供了多种转储后备副本的方法,如文件拷贝、利用 Oracle 的 Export 实用程序、用 SQL 命令 Spool 以及自己编程实现等。相应地,Oracle 也提供了多种重装后备副本的方法,如文件拷贝、利用 Oracle 的 Import 实用程序、利用 SQL*LOADER 以及自己编程实现等。

在 Oracle 早期版本(V.5)中,日志文件以数据块为单位,也就是说,Oracle 的恢复操作是基于数据块的,不是基于操作的。Oracle 中记录数据库更新前的旧值的日志文件称为数据库前像文件(Before Image,简称 BI 文件),记录数据库更新后的新值的日志文件称为数据库的后像文件(After Image,简称 AI 文件)。BI 文件是必须配置的,AI 文件是可以任选的。

Oracle7 为了能够在出现故障时更有效地恢复数据,也为了解决读"脏"数据问题,提供了 REDO 日志文件和回滚段(Rollback Segment)。REDO 日志文件中记录了被更新数据的前像和后像。回滚段记录更新数据的前像,设在数据库缓冲区中。在利用日志文件进行故障恢复时,为减少扫描日志文件的遍数,Oracle7 首先扫描 REDO 日志文件,重做所有操作,包括未正常提交的事务的操作,然后再根据回滚段中的数据,撤销未正常提交的事务的操作。

详细技术希望读者自己设法了解 Oracle 最新版本的介绍,例如通过 INTERNET 访问 Oracle 公司的网站。也可以了解其他 DBMS 厂商的产品情况。

*15. 试用恢复的基本技术设计一个恢复子系统,给出这个子系统的恢复策略,包括:

(1) 当产生某一类故障时如何恢复数据库的方法;

(2) 日志文件的结构;

(3) 登记日志文件的方法;

(4) 利用日志文件恢复事务的方法;

(5) 转储的类型;

(6) 转储的后备副本和日志文件如何配合使用。

解析

这是一个大作业,可以综合复习和运用学到的知识。读者可以参考《概论》上 7.4.2、7.5、7.6,设计一个恢复子系统。

例如,日志文件的结构可以以记录为单位,也可以以数据块为单位。不同的日志文件结构,登记的日志内容、日志文件恢复事务的方法也就不同了。

对于研究生,还应该上机模拟实现设计的恢复子系统。

第八章　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。读者学习本章一定要做到概念清楚。

为此要认真读书,特别是读书上的例题。读书时要自己动手先做一做例题,体会一下是否掌握了有关概念。

一、基本知识点

数据库是一个共享资源,当多个用户并发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(丢失修改、不可重复读、读“脏数据”)及其确切含义;封锁的类型;不同封锁类型的(例如 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。

二、习题解答和解析

1. 在数据库中为什么要并发控制?

答

数据库是共享资源,通常有许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。

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:丢失修改、不可重复读和读"脏"数据。

(1) 丢失修改(Lost Update)

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(Non - Repeatable Read)

不可重复读是指事务 T_1 读取数据后,事务 T_2 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:详见《概论》8.1(P266)。

(3) 读"脏"数据(Dirty Read)

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和技术就是并发控制。最常用的技术是封锁技术。也可以用其他技术,例如在分布式数据库系统中可以采用时间戳方法来进行并发控制。

3. 什么是封锁?

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此数据对象。

封锁是实现并发控制的一个非常重要的技术。

4. 基本的封锁类型有几种?试述它们的含义。

答

基本的封锁类型有两种:排它锁(Exclusive Locks,简称 X 锁)和共享锁(Share Locks,简称 S 锁)。

排它锁又称为写锁。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。

5. 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,例如下图中事务 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
③ A←A - 1	等待
写回 A = 15	等待
Commit	等待
Unlock A	等待
④	获得 Xlock A
	读 A = 15
	A←A - 1
	写回 A = 14
⑤	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。

6. 什么是封锁协议?不同级别的封锁协议的主要区别是什么?

答

在运用封锁技术对数据加锁时,要约定一些规则。例如,在运用 X 锁和 S 锁对数据对象加锁时,要约定何时申请 X 锁或 S 锁、何时释放封锁等。这些约定或者规则称为封锁协议(Locking Protocol)。对封锁方式约定不同的规则,就形成了各种不同的封锁协议、不同级别的封锁协议,例如《概论》8.3 中介绍的三级封锁协议,三级协议的主要区别在于什么操作需要申请封锁,何时申请封锁以及何时释放锁(即持锁时间的长短)。

一级封锁协议:事务 T 在修改数据 R 之前必须先对其加 X 锁,直到事务结束才释放。

二级封锁协议:一级封锁协议加上事务 T 在读取数据 R 之前必须先对其加

S 锁,读完后即可释放 S 锁。

三级封锁协议:一级封锁协议加上事务 T 在读取数据 R 之前必须先对其加 S 锁,直到事务结束才释放。

7. 不同封锁协议与系统一致性级别的关系是什么?

答

不同的封锁协议对应不同的一致性级别。

一级封锁协议可防止丢失修改,并保证事务 T 是可恢复的。在一级封锁协议中,对读数据是不加 S 锁的,所以它不能保证可重复读和不读"脏"数据。

二级封锁协议除防止了丢失修改,还可进一步防止读"脏"数据。在二级封锁协议中,由于读完数据后立即释放 S 锁,所以它不能保证可重复读。

在三级封锁协议中,无论是读数据还是写数据都加长锁,即都要到事务结束时才释放封锁。所以三级封锁协议除防止了丢失修改和不读"脏"数据外,还进一步防止了不可重复读。

下面的表格清楚地说明了封锁协议与系统一致性的关系。

	X 锁		S 锁		一致性保证		
	操作结束释放	事务结束释放	操作结束释放	事务结束释放	不丢失修改	不读"脏"数据	可重复读
一级封锁协议		√			√		
二级封锁协议		√	√		√	√	
三级封锁协议		√		√	√	√	√

8. 什么是活锁?什么是死锁?

答

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	·	·

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,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。

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

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

9. 试述活锁的产生原因和解决方法。

答

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。

10. 请给出预防死锁的若干方法。

答

在数据库中,产生死锁的原因是两个或多个事务都已封锁了一些数据对象,然后又都请求已被其他事务封锁的数据加锁,从而出现死等待。

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。

不过,预防死锁的策略不大适合数据库系统的特点,具体原因可参见《概论》8.4。

11. 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允许死锁发生,DBMS 检测到死锁后加以解除的方法。

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: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。超时法实现简单,但有可能误判死锁,事务因其他原因长时间等待超过时限时,系统会误认为发生了死锁。若时限设置得太长,又不能及时发现死锁发生。

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,将其撤消,释放此事务持有的所有锁,使其他事务得以继续运行下去。当然,对撤销的事务所执行的数据修改操作必须加以恢复。

12. 什么样的并发调度是正确的调度?

答

可串行化(Serializable)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。

13. 设 T_1,T_2,T_3 是如下的 3 个事务:

T_1:A:= A + 2;

T_2:A:= A * 2;

T_3:A:= A * * 2; (A A^2)

设 A 的初值为 0。

(1) 若这 3 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		
	Slock A	
A = Y + 2	等待	
写回 A(= 2)	等待	
Unlock A	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 * *2
	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 3 个事务都遵守两段锁协议,请给出一个不产生死锁的可串行化调度。

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 3 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

14. 试述两段锁协议的概念。

答

两段锁协议是指所有事务必须分两个阶段对数据项加锁和解锁。

在对任何数据进行读、写操作之前,首先要申请并获得对该数据的封锁;

在释放一个封锁之后,事务不再申请和获得任何其他封锁。

“两段”的含义是,事务分为两个阶段:

第一阶段是获得封锁,也称为扩展阶段,在这阶段,事务可以申请获得任何数据项上的任何类型的锁,但是不能释放任何锁;

第二阶段是释放封锁,也称为收缩阶段,在这阶段,事务释放已经获得的锁,但是不能再申请任何锁。

15. 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,存在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。$Y = \{A_{i+1}, \cdots, A_n\}$符合所情况(2)。

"x∈X,T_1 需要 Xlock x ①

T_2 需要 Slock x 或 Xlock x ②

1) 如果操作①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v w∈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②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。

16. 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。

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

Unlock A	等待
	等待
	Slock A
	读 A = 3
	X = A
	Unlock A
	Xlock B
	B = X + 1
	写回 B = 4
	Unlock B

17. 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。该封锁子系统支持多种封锁粒度。

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》8 7.1)。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。

例如事务 T 要对某个元组加 X 锁，则首先要对关系和数据库加 IX 锁。换言之，对关系和数据库加 IX 锁，表示它的后裔结点——某个元组拟(意向)加 X 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。例如，事务 T 要对关系 R 加 X 锁时，系统只要检查根结点数据库和 R 本身是否已加了不相容的锁(如发现已经加了 IX，则与 X 冲突)，而不再需要搜索和检查 R 中的每一个元组是否加了 X 锁或 S 锁。

18. 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

19. 理解并解释下列术语的含义:封锁、活锁、死锁、排它锁、共享锁、并发事务的调度、可串行化的调度、两段锁协议。

答

(略,已经在上面有关习题中解答)

*20. 试述你了解的某一个实际的 DBMS 产品的并发控制机制。

答

(略,参见《概论》8.8 节,简单介绍了有关 Oracle 的并发控制机制。)

第九章　数据库安全性

《概论》第九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。

一、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;统计数据库的安全性问题。

② 需要牢固掌握的: TDI/ TCSEC 标准的主要内容;C2 级 DBMS、B1 级 DBMS 的主要特征;实现数据库安全性控制的常用方法和技术有哪些;数据库中的自主存取控制方法和强制存取控制方法。

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: MAC 机制中确定主体能否存取客体的存取规则,读者要理解并掌握存取规则为什么要这样规定,特别是规则(2)。

二、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. 数据库安全性和计算机系统的安全性有什么关系?

答

安全性问题不是数据库系统所独有的,所有计算机系统都有这个问题。只是在数据库系统中大量数据集中存放,而且为许多最终用户直接共享,从而使安全性问题更为突出。

系统安全保护措施是否有效是数据库系统的主要指标之一。

数据库的安全性和计算机系统的安全性,包括操作系统、网络系统的安全性是紧密联系、相互支持的,

3. 试述可信计算机系统评测标准的情况,试述 TDI/ TCSEC 标准的基本内容。

答

各个国家在计算机安全技术方面都建立了一套可信标准。目前各国引用或制定的一系列安全标准中,最重要的是美国国防部(DoD)正式颁布的《DoD 可信计算机系统评估标准》(Trusted Computer System Evaluation Criteria,简称 TCSEC,又称桔皮书)。(详细介绍参见《概论》9.1.2)。

TDI/ TCSEC 标准是将 TCSEC 扩展到数据库管理系统,即《可信计算机系统评估标准关于可信数据库系统的解释》(Trusted Database Interpretation 简称 TDI,又称紫皮书)。在 TDI 中定义了数据库管理系统的设计与实现中需满足和用以进行安全性级别评估的标准。

TDI 与 TCSEC 一样,从安全策略、责任、保证和文档四个方面来描述安全性级别划分的指标。每个方面又细分为若干项。这些指标的具体内容,参见《概论》9.1.2。

4. 试述 TCSEC(TDI)将系统安全级别划分为 4 组 7 个等级的基本内容。

答

根据计算机系统对安全性各项指标的支持情况,TCSEC(TDI)将系统划分为四组(division)7 个等级,依次是 D、C(C1,C2)、B(B1,B2,B3)、A(A1),按系统可靠或可信程度逐渐增高。

安全级别	定 义
A1	验证设计(Verified Design)
B3	安全域(Security Domains)
B2	结构化保护(Structural Protection)
B1	标记安全保护(Labeled Security Protection)
C2	受控的存取保护(Controlled Access Protection)
C1	自主安全保护(Discretionary Security Protection)
D	最小保护(Minimal Protection)

这些安全级别之间具有一种偏序向下兼容的关系,即较高安全性级别提供

的安全保护包含较低级别的所有保护要求,同时提供更多或更完善的保护能力。

各个等级的基本内容为:

D 级　D 级是最低级别。一切不符合更高标准的系统,统统归于 D 组。

C1 级　只提供了非常初级的自主安全保护。能够实现对用户和数据的分离,进行自主存取控制(DAC),保护或限制用户权限的传播。

C2 级　实际是安全产品的最低档次,提供受控的存取保护,即将 C1 级的 DAC 进一步细化,以个人身份注册负责,并实施审计和资源隔离。

B1 级　标记安全保护。对系统的数据加以标记,并对标记的主体和客体实施强制存取控制(MAC)以及审计等安全机制。

B2 级　结构化保护。建立形式化的安全策略模型并对系统内的所有主体和客体实施 DAC 和 MAC。

B3 级　安全域。该级的 TCB 必须满足访问监控器的要求,审计跟踪能力更强,并提供系统恢复过程。

A1 级　验证设计,即提供 B3 级保护的同时给出系统的形式化设计说明和验证以确信各安全保护真正实现。

各个等级的基本内容请参见《概论》9.1.2。特别是《概论》上表 9.2 列出了各安全等级对安全指标的支持情况。希望读者掌握《概论》上的内容,这里就不重复了。

5. 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:

(1) 用户标识和鉴别:该方法由系统提供一定的方式让用户标识自己的名字或身份。每次用户要求进入系统时,由系统进行核对,通过鉴定后才提供系统的使用权。

(2) 存取控制:通过用户权限定义和合法权检查确保只有合法权限的用户访问数据库,所有未被授权的人员无法存取数据。例如 C2 级中的自主存取控制(DAC),B1 级中的强制存取控制(MAC)。

(3) 视图机制:为不同的用户定义视图,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,从而自动地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,DBA 可以利用审计跟踪的信息,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,从而使得不知道解密算法的人无法获知数据的内容。

具体内容请参见《概论》9.2。

6. 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。

解析

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。

7. SQL 语言中提供了哪些数据控制(自主存取控制)的语句？请试举几例说明它们的使用方法。

答

SQL 中的自主存取控制是通过 GRANT 语句和 REVOKE 语句来实现的。如:

```
GRANT SELECT, INSERT ON Student
TO 王平
WITH GRANT OPTION;
```

就将 Student 表的 SELECT 和 INSERT 权限授予了用户王平,后面的“WITH GRANT OPTION”子句表示用户王平同时也获得了“授权”的权限,即可以把得到的权限继续授予其他用户。

```
REVOKE INSERT ON Student FROM 王平 CASCADE;
```

就将 Student 表 的 INSERT 权限从用户王平处收回,选项 CASCADE 表示,如果用户王平将 Student 的 INSERT 权限又转授给了其他用户,那么这些权限也将从其他用户处收回。

8. 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 权力。

```
GRANT SELECT ON 职工,部门
TO 王明;
```

(2) 用户李勇对两个表有 INSERT 和 DELETE 权力。

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

(3) * 每个职工只对自己的记录有 SELECT 权力。

```
GRANT SELECT ON 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。读者应该了解你使用的 DBMS 产品的扩展语句。

(4) 用户刘星对职工表有 SELECT 权力,对工资字段具有更新权力。

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

(5) 用户张新具有修改这两个表的结构的权力。

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

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

```
GRANT ALL PRIVILIGES ON 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、平均工资的权力,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 部门工资
TO 杨兰;
```

9. 把习题 8 中(1)～(7) 的每一种情况,撤销各用户所授予的权力。

答

(1)

```
REVOKE SELECT ON 职工,部门
FROM 王明;
```

(2)

```
REVOKE INSERT,DELETE ON 职工,部门
FROM 李勇;
```

(3)

```
REOVKE SELECT ON 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 职工
FROM 刘星;
```

(5)

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

(6)

```
REVOKE ALL PRIVILIGES ON 职工,部门
FROM 周平;
```

(7)

```
REVOKE SELECT ON 部门工资
FROM 杨兰;
DROP VIEW 部门工资;
```

10. 为什么强制存取控制提供了更高级别的数据库安全性?

答

强制存取控制(MAC)是对数据本身进行密级标记,无论数据如何复制,标记与数据是一个不可分的整体,只有符合密级标记要求的用户才可以操纵数据,从而提供了更高级别的安全性。

11. 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(Label)。敏感度标记被分成若干级别,例如绝密(Top Secret)、机密(Secret)、可信(Confidential)、公开(Public)等。主体的敏感度标记称为许可证级别(Clearance Level),客体的敏感度标记称为密级(Classification Level)。

12. 举例说明 MAC 机制如何确定主体能否存取客体。

答

假设要对关系变量 S 进行 MAC 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)

S#	SNAME	STATUS	CITY	CLASS
S1	Smith	20	London	2
S2	Jones	10	Paris	3
S3	Clark	20	London	4

假设用户 U1 和 U2 的许可证级别分别为 3 和 2,则根据规则 U1 能查得元组 S1 和 S2,可修改元组 S2;而 U2 只能查得元组 S1,只能修改元组 S1。

解析

这里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等于客体的密级时才能写相应的客体。

13. 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,DBA 可以根据审计跟踪的信息,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。

14. 统计数据库中存在何种特殊的安全性问题?

答

统计数据库允许用户查询聚集类型的信息,如合计、平均值、最大值、最小值等,不允许查询单个记录信息。但是,人们可以从合法的查询中推导出不合法的信息,即可能存在隐蔽的信息通道,这是统计数据库要研究和解决的特殊的安全性问题。

*15. 试述你了解的某一个实际的 DBMS 产品的安全性措施。

答

不同的 DBMS 产品以及同一产品的不同版本的安全措施各不相同,仁者见仁,智者见智,请读者自己了解。《概论》上 9.4 简单介绍了有关 Oracle 数据库的安全性措施。

第十章　数据库完整性

《概论》第十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。

一、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反应。

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。

二、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?可分为哪几类?

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。一般可以分为六类:静态列级约束、静态元组约束、静态关系约束、动态列级约束、动态元组约束、动态关系约束。

静态列级约束是对一个列的取值域的说明,包括以下几个方面:

(1) 对数据类型的约束,包括数据的类型、长度、单位、精度等;

(2) 对数据格式的约束;

(3) 对取值范围或取值集合的约束;

(4) 对空值的约束;

(5) 其他约束。

静态元组约束就是规定组成一个元组的各个列之间的约束关系,静态元组约束只局限在单个元组上。

静态关系约束是在一个关系的各个元组之间或者若干关系之间常常存在各种联系或约束。常见的静态关系约束有:

(1) 实体完整性约束;

(2) 参照完整性约束;

(3) 函数依赖约束。

动态列级约束是修改列定义或列值时应满足的约束条件,包括下面两方面:

(1) 修改列定义时的约束;

(2) 修改列值时的约束。

动态元组约束是指修改某个元组的值时需要参照其旧值,并且新旧值之间需要满足某种约束条件。

动态关系约束是加在关系变化前后状态上的限制条件,例如事务一致性、原子性等约束条件。

详细内容可以参见《概论》10.1中的介绍。

4. DBMS的完整性控制机制应具有哪些功能?

答

DBMS的完整性控制机制应具有三个方面的功能:

(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) 违约反应: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以下几个方面:

(1) 外码是否可以接受空值。

(2) 删除被参照关系的元组时的考虑,这时系统可能采取的作法有三种:

1) 级联删除(CASCADES);

2) 受限删除(RESTRICTED);

3) 置空值删除(NULLIFIES)。

(3) 在参照关系中插入元组时的问题,这时系统可能采取的作法有:

1) 受限插入;

2) 递归插入。

(4) 修改关系中主码的问题。一般是不能用 UPDATE 语句修改关系主码的。如果需要修改主码值,只能先删除该元组,然后再把具有新主码值的元组插入到关系中。如果允许修改主码,首先要保证主码的惟一性和非空,否则拒绝修改。然后要区分是参照关系还是被参照关系。

详细讨论可以参见《概论》10.2。

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:

定义每个模式的主码;定义参照完整性;定义职工年龄不得超过 60 岁。

答

```
CREATE TABLE DEPT
        (Deptno NUMBER(2),
        Deptname VARCHAR(10),
        Manager VARCHAR(10),
        PhoneNumber Char(12)
        CONSTRAINT PK_SC PRIMARY KEY (Deptno));
CREATE TABLE EMP
        (Empno NUMBER(4),
        Ename VARCHAR(10),
        Age NUMBER(2),
        CONSTRAINT C1 CHECK (Aage < = 60),
```

```
Job VARCHAR(9),
Sal NUMBER(7,2),
Deptno NUMBER(2),
CONSTRAINT FK_DEPTNO
    FOREIGN KEY (Deptno)
    REFERENCES DEPT(Deptno));
```

7. 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》10.2中相应部分。

*8. 试述你了解的某一个实际的DBMS产品的完整性控制策略。

答

不同的DBMS产品以及同一产品的不同版本的完整性控制策略各不相同,读者要去了解某一个具体的DBMS产品的完整性控制策略。

《概论》10.3简单介绍了有关Oracle数据库的完整性控制策略。

第十一章　数据库管理系统

习题解答和解析

1. 试述 DBMS 的基本功能？

答

DBMS 主要是实现对共享数据有效地组织、管理和存取。DBMS 的基本功能有:

(1) 数据库定义功能;

(2) 数据存取功能;

(3) 数据库运行管理功能;

(4) 数据组织、存储和管理功能;

(5) 数据库的建立和维护功能;

(6) 其他功能。

具体内容请参见《概论》11.2。

2. 试述 DBMS 四种进程组织方案,并分析各种方案的特点和优缺点。

答

DBMS 四种进程组织方案是:

N 方案:N 个 DB 用户应用程序对应 N 个用户进程,DBMS 作为应用程序的子程序被连入用户应用程序中。因此这种方案也称为连入式方案。数据库系统中共有 N 个进程。

2N 方案:每个用户进程有一个 DBMS 进程为之服务,因此 N 个用户进程就有 N 个 DBMS 进程,共 2N 个进程。

N+1 方案:N 个用户进程仅有一个 DBMS 进程为它们服务,因此共 N+1 个进程。

N+M 方案:用 M 个 DBMS 进程为 N 个用户进程提供服务,一般 M<N。

这四个方案各自的特点和优缺点请参见《概论》11.2.1、11.2.2、11.2.3、11.2.4。

3. 理解并解释下列术语的含义:进程、任务、"轻权"进程、线程、线索。

答

进程:操作系统中的核心概念,进程是程序的一次执行过程。进程既是资源分配的最小单位也是操作系统调度的基本单位。

任务:在单处理器系统中任务与进程是类似的概念。随着多处理器系统以及并行计算技术的发展,进程概念被进一步细划为任务(Task)与线程(Thread)的概念。任务是申请资源的基本单位,而线程是调度和运行的基本单位。

“轻权”进程:线程又被称为“轻权”或“轻量”进程。

线索:是数据库系统中的概念,它借鉴了操作系统中“线程”的概念:整个DBMS可以看作是一个Task,当有一个用户申请数据库服务时,Task分配多个Thread为之服务,多个Thread并行工作,共享资源。一般地讲,DBMS中的线索是DBMS的一个执行流。

4. 什么是DBMS的多线索机制?有什么优点?

答

DBMS借鉴了操作系统中“线程”的概念和技术,在DBMS的实现中采用多线索机制。

一般地讲,DBMS中的线索是DBMS的一个执行流,它服务于整个DBMS系统或DBMS中的某个用户;DBMS服务器响应客户请求是通过为每个用户创建线索(而不是创建进程)来完成的。DBMS的各个线索能在逻辑上并行执行;它们共存于一个服务器进程中,共享DBMS的所有资源,如数据库缓冲区和CPU时间;线索是DBMS的调度单位,服务器进程能按一定的调度算法调度用户请求。与进程相比,线索具有以下优点:

(1) 线索比进程占用较少的系统资源,如内存;

(2) 线索调度比较灵活,可控制性强;

(3) 线索切换开销较小;

(4) 线索间通信简便。

有关DBMS的多线索机制可参见《概论》11.2.5。

5. DBMS由哪些主要的程序模块组成?

答

DBMS主要的程序模块有:

(1) 数据定义模块;

(2) 数据操纵模块;

(3) 数据库运行管理方面的程序模块;

(4) 数据库组织、存储和管理方面的程序模块;

(5) 数据库建立、维护和其他方面的程序模块。

DBMS程序模块组成请参见《概论》11.3.1。

6. DBMS工作过程是什么?给出DBMS插入一个记录的活动过程,画出类似

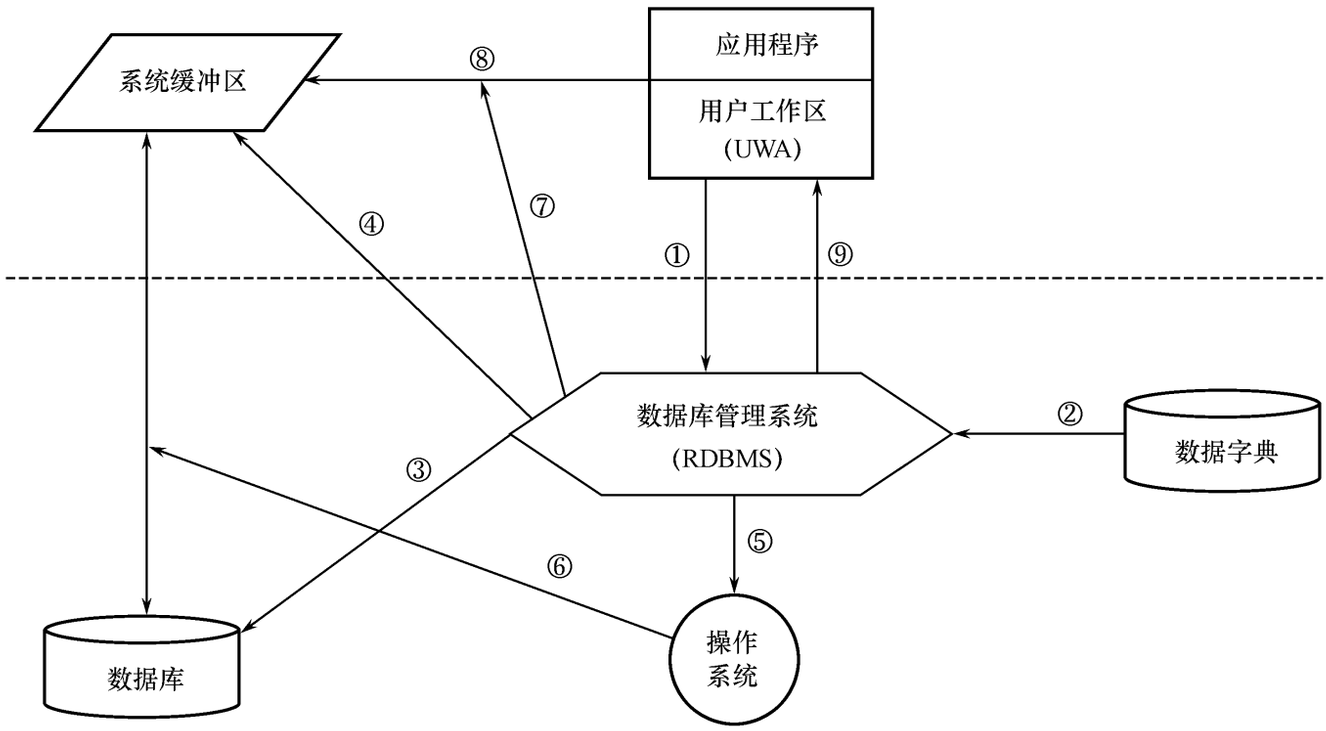
《概论》上图 11.8 的活动过程示意图。

答

(1) 用户 A 通过应用程序向 DBMS 发出调用数据库数据的 INSERT 命令。命令中给出了一个关系名和插入的元组值。

(2) DBMS 先对命令进行语法检查、语义检查和用户存取权限检查:语义检查的具做法是,DBMS 读取数据字典,检查是否存在该关系及相应的字段,值的类型是否正确;检查该用户是否具有该关系上执行 INSERT 的权限;检查通不过就拒绝执行 INSERT 命令,返回有关的错误信息。

(3) DBMS 查看存储模式,找到新记录应插入的位置和应在的页面 P。



(4) DBMS 在缓冲区中找到一个空页。

(5) DBMS 根据(3)的结果,向操作系统发出读取物理页面 P 的命令。

(6) 操作系统执行读操作,将数据页 P 读入系统缓冲区的空页处。

(7) DBMS 根据插入命令和数据字典的内容将数据转化成内部的记录格式。

(8) DBMS 将数据记录写入到系统缓冲区的页 P 中。

(9) DBMS 将执行事务提交,把状态信息,如成功或不成功的指示、例外状态信息等返回给应用程序。

/ * 这里没有考虑多用户并发控制的问题 */

7. DBMS 的语言翻译层是如何处理一个 DDL 语句的?

答

语言翻译处理层首先要对 DDL 语句进行语法检查、语义检查和用户权限检查。语义检查的内容具体做法是,DBMS 读取数据字典,检查是否存在与该语句

中的表,或视图,或索引等要创建的对象名相同的对象名,检查该用户是否具有创建数据库对象的权限。然后把 DDL 语句翻译成内部表示,把它存储在系统的数据字典中。例如新建立一个表,就要把关系名、建立者、属性个数、记录长度等信息记入数据字典中。

解析

DBMS 的恢复子系统和审计子系统会把与该语句有关的日志记录和审计记录自动记录到系统的日志文件和审计文件中。

8. 试述 DBMS 的语言翻译层处理一个 DML 语句的大致过程。

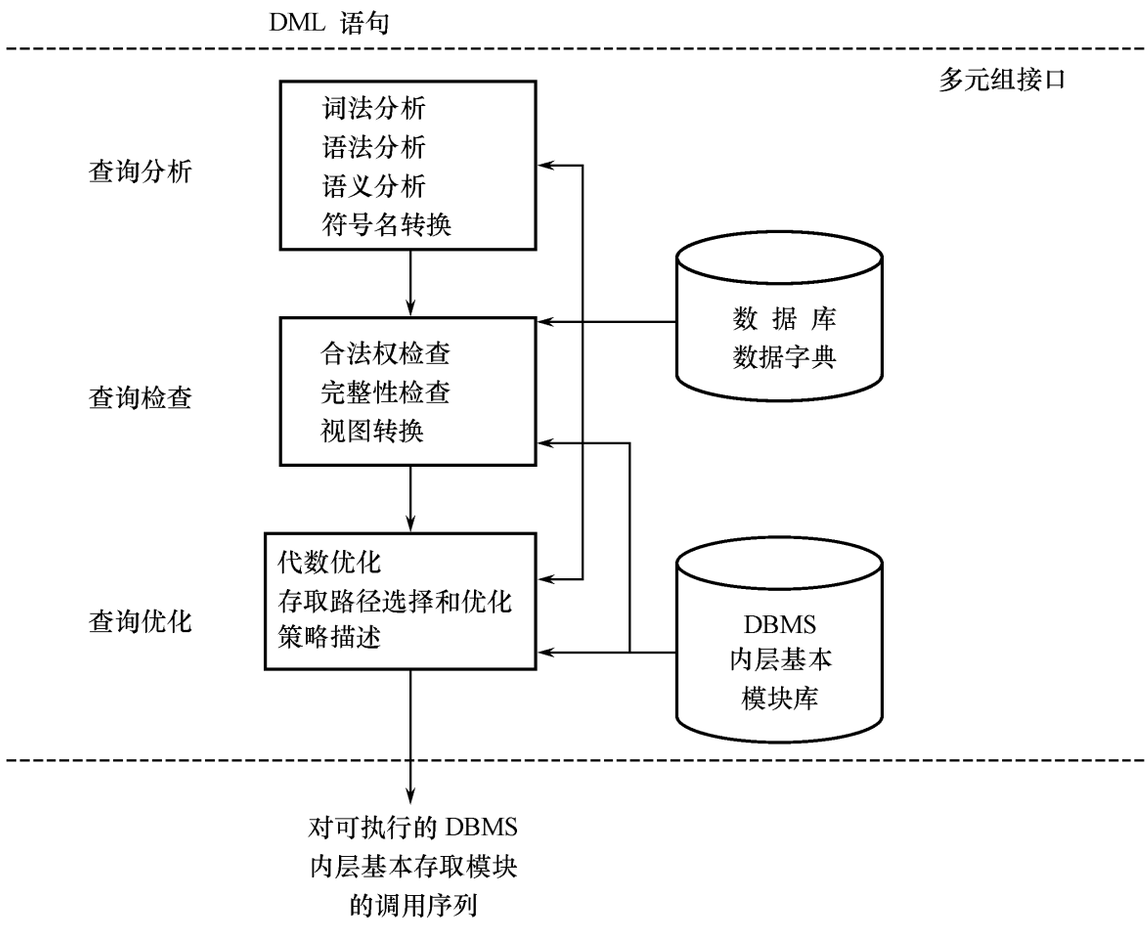
答

首先,对 DML 语句进行词法分析和语法分析,并把外部关系名、属性名转换为内部名。词法和语法分析通过后生成语法分析树。

接着,根据数据字典中的内容进行查询检查,包括审核用户的存取权限、视图转换和完整性检查。

然后,对查询进行优化。优化分为两类,一类为代数优化,另一类为存取路径优化。并把选中的查询执行方案描述出来。

DBMS 语言翻译层处理一个 DML 语句的过程称为一个逐步束缚的过程,如下图所示。



详细解释参见《概论》11.4.1。

9. 什么是处理 DML 语句的解释方法和预编译方法？试述二者的区别、联系，比较各自的优缺点。

答

解释执行 DML 语句的方法是：执行语句前，该语句都以原始字符串的形式保存。当执行到该语句时，才利用解释程序去完成束缚的全部过程，同时予以执行。

解释方法的优点是：应变性强，能适应在解释过程中发生的数据结构、存储结构等的变化，因此能保持较高的数据独立性。缺点是：每执行一次 DML 语句时都要经过所有解释步骤，尤其当这样的语句位于一个循环体内时，就要多次重复解释一个 DML 语句，显然效率比较低。

预编译方法是：在用户提交 DML 语句之后对它进行翻译处理，保存产生的可执行代码。当需要运行时，取出保存的可执行代码加以执行。

预编译方法的优点是：效率高。但是，使用这种方法会遇到这样的问题：在束缚过程中进行优化所依据的条件可能在运行前已不存在，导致已作出的应用规划在执行时不再有效。为了解决这类问题，可以采用自动重编译技术。

10. 试述数据存取层主要的子系统及其功能。

答

数据存取层中包括记录存取子系统、事务管理子系统、封锁子系统、恢复子系统、存取路径维护子系统、排序/合并模块等。主要功能有：

(1) 记录存取、事务管理子系统：记录存取子系统提供按某个属性值直接取一个元组和顺序取一个元组的存取原语；事务管理子系统提供定义和控制事务的操作。

(2) 封锁子系统，执行并发控制。

(3) 恢复子系统：主要是日志登记子系统把事务开始、滚回、提交，对元组的插入、删除、修改，对索引记录的插入、删除、修改等每一个操作作为一个日志记录存入日志文件中，对不同的故障恢复策略执行恢复。

(4) 控制信息管理模块：该模块利用专门的数据区(内存中)登记不同记录类型以及不同存取路径的说明信息(取自数据字典)和控制信息。

(5) 存取路径维护子系统：对数据执行插入、删除、修改操作的同时要对相应的存取路径进行维护。

(6) 排序/合并子系统：在语言翻译处理层中，描述性语言表达的集合级操作被转换成一系列的对数据存取层所提供的存取原语的调用。为了得到用户所要求的有序输出，为了加速关系运算(如自然连接)的中间步骤，为了提高效率，常常需要对关系元组重新排序。这一工作由排序/合并子系统来完成

这些子系统的具体功能请参见《概论》11.5.2。

11. 在操作系统中也有并发控制问题,为什么 DBMS 还要并发控制机制?

答

操作系统提供的封锁机制和 DBMS 的封锁机制在封锁对象、封锁对象的状态、封锁的粒度及封锁的类型上存在很大的差别,操作系统的封锁机制不能直接应用在 DBMS 中,DBMS 必须重新设计,来满足复杂的封锁需求。

12. 试比较 DBMS 与操作系统的封锁技术。

答

	操作系统	数据库管理系统
封锁对象	单一,系统资源(包括 CPU、设备、表格等)	多样,数据库中各种数据对象(包括用户数据、索引(存取路径)、数据字典等)
封锁对象的状态	静态,确定,各种封锁对象在封锁表中占有一项,封锁对象数是不变的	动态,不确定,封锁对象动态改变着,常常在执行前不能确定,一个封锁对象只有当封锁时才在封锁表中占据一项
封锁的粒度	不变,由于封锁对象单一、固定,封粒度不会改变	可变,封锁可加到或大或小的数据单位上,封锁粒度可以是整个数据库、记录或字段
封锁的类型	单一,排它锁	多样,一般有共享锁(S Lock)、排它锁(X Lock)或其他类型的封锁,随系统而异

详细解释请参见《概论》11.5.2。

13. DBMS 中为什么要设置系统缓冲区?

答

设立系统缓冲区的原因:

一是为了把存储层以上的 DBMS 各系统成分和实在的外存设备隔离,外存设备的变更不会影响这些系统,使 DBMS 具有设备独立性。

二是为了提高效率。DBMS 利用系统缓冲区滞留数据。当需要读取数据时系统首先到缓冲区中查找。只有当缓冲区中不存在该数据时才真正从外存读入该数据所在的页面。当要写回一元组到数据库中时,系统并不把它立即写回外存,仅把该元组所在的缓冲区页面作一标志,表示可以释放。只有当该用户事务结束或缓冲区已满需要调入新页时才按一定的淘汰策略把缓冲区中已有释放标志的页面写回外存。这样可以减少 I/O 次数,提高系统效率。

14. 数据库中要存储和管理的数据内容包括哪些方面?

答

数据库中存储四个方面的数据:

(1) 数据描述,即数据的外模式、模式、内模式;

(2) 数据本身;

(3) 数据之间的联系;

(4) 存取路径。

这四个方面的数据内容都要采用一定的方式组织、存储起来。

*15. 请给出缓冲区管理中的一个淘汰算法,并上机实现(提示:你首先要设计缓冲区的数据结构,然后写出算法)。

解析

先设计缓冲区数据结构,该结构可以包括两大部分:

(1) 缓冲区控制的记录表,用于登记缓冲区使用情况,分别记录已使用的缓冲区页面和未使用的缓冲区页面(可以用缓冲区指针或缓冲区页面序号来代表)。

(2) 缓冲区页面的结构,包括读入缓冲区的物理块的块号、该页面被使用的次数、该页面最近被使用的时戳。

再设计缓冲区的各种管理算法。包括读/写缓冲区页面、查找缓冲区页面、登记缓冲区记录、缓冲区的淘汰算法等。其中页面淘汰算法,可以设计不同的页面淘汰策略,常用的如"最近最少用到的页面淘汰"策略。

*16. 请写出对一个文件按某一个属性的排序算法(设该文件的记录是定长的),并上机实现。若要按多个属性排序,你能写出改进的算法吗?

解析

这是一个外排序的问题。

算法思想:使用合并排序的方法实现,即将文件分为若干个部分,每部分先使用内排序的算法实现排序,再将排好序的各部分进行合并。

文件分为多少部分,即多路合并的路数由文件大小和可供内排序用的内存大小决定,这是外排序的优化问题。

*17. 请给出 B+ 树文件的创建和维护(增、删、改)算法并上机实现

解析

设 B+ 树叶结点上仅存放索引项[码值,TID],你首先要设计索引项、B 树叶页和非叶页的数据结构,然后写出算法。

解析

习题 15、16、17 可作为学生 DBMS 课程的实习课题。这些模块是 DBMS 中必不可少的基本模块。

第十二章　数据库技术新发展

习题解答和解析

1. 试述数据库技术的发展过程。

答

(1) 数据模型是数据库系统的核心和基础。数据库技术的三个发展阶段应该按照数据模型的进展来界定。按照数据模型的进展,数据库技术可以相应地分为三个发展阶段。

(2) 数据模型的发展经历了格式化数据模型(包括层次数据模型和网状数据模型)、关系数据模型两个阶段,发展到以面向对象数据模型为代表的非传统数据模型的阶段。

(3) 读者可以从每一代数据库系统的主要特征、代表性系统、主要成就、优点和不足来了解数据库技术的发展过程。

层次数据库系统和网状数据库系统的数据模型虽然分别为层次模型和网状模型,但实质上层次模型是网状模型的特例。它们都是格式化模型。它们从体系结构、数据库语言到数据存储管理均具有共同特征,是第一代数据库系统。

关系数据库系统支持关系模型。关系模型不仅简单、清晰,而且有关系代数作为语言模型,有关系数据理论作为理论基础。因此,关系数据库系统具有形式基础好、数据独立性强、数据库语言非过程化等特色,标志着数据库技术发展到了第二代。

第二代数据库系统的数据模型虽然描述了现实世界数据的结构和一些重要的相互联系,但是仍不能捕捉和表达数据对象所具有的丰富而重要的语义,因此尚只能属于语法模型。

第三代的数据库系统将以更加丰富的数据模型和更强大的数据管理功能为特征,从而满足传统数据库系统难以支持的新的应用要求。

2. 当前数据库技术发展的主要特征是什么?

答

新一代数据库技术的特点是:

(1) 面向对象的方法和技术对数据库发展的影响最为深远

数据库研究人员借鉴和吸收了面向对象的方法和技术,提出了面向对象数据模型(简称对象模型)。该模型克服了传统数据模型的局限性,促进了数据库技术在一个新的技术基础上继续发展。

(2) 数据库技术与多学科技术的有机结合

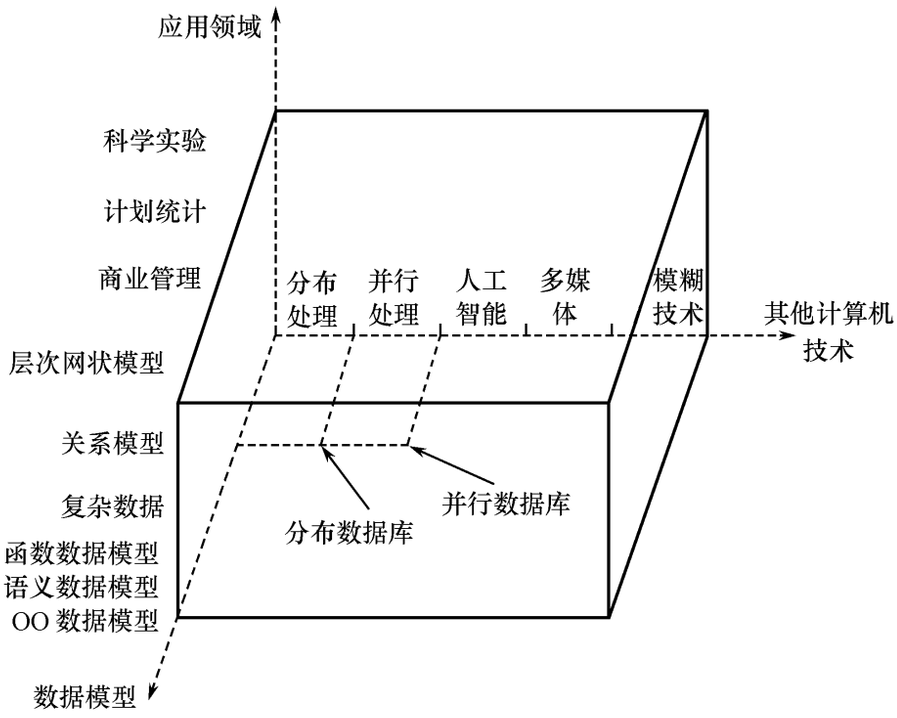
计算机领域中其他新兴技术的发展对数据库技术产生了重大影响。传统的数据库技术和其他计算机技术,如网络通信技术、人工智能技术、面向对象程序设计技术、并行计算技术、移动计算技术等的互相结合、互相渗透,使数据库中新的技术内容层出不穷。

(3) 面向应用领域的数据库技术的研究

在传统数据库系统基础上,结合各个应用领域的特点,研究适合该应用领域的数据库技术,如数据仓库、工程数据库、统计数据库、科学数据库、空间数据库、地理数据库等,这是当前数据库技术发展的又一重要特征。

解析

可以用一个三维空间的视图,比较清晰地从数据模型、其他计算机技术、应用领域 3 个方面描述新一代数据库系统及其相互关系。



3. 试述第一、二代数据库系统的主要成就。

答

第一代数据库系统指层次和网状数据库系统,其代表是:

(1) 1969 年 IBM 公司研制的层次模型的数据库管理系统 IMS。

(2) 美国数据库系统语言协商会 CODASYL 下属的数据库任务组 DBTG 对数据库方法进行了系统的研究、探讨,于 20 世纪 60 年代末 70 年代初提出了若干

DBTG 报告。DBTG 报告确定并建立了数据库系统的许多概念、方法和技术。DBTG 所提议的方法是基于网状结构的。它是数据库网状模型的典型代表。在 DBTG 方法和思想的指引下数据库系统的实现技术不断成熟,开发了许多商品化的数据库管理系统,它们都是基于网状模型或层次模型的。

可以说,层次数据库是数据库系统的先驱,而网状数据库则是数据库概念、方法、技术的奠基。它们是数据库技术中研究得最早的两种数据库系统。

支持关系数据模型的关系数据库系统是第二代数据库系统。

20 世纪 70 年代是关系数据库理论研究和原型开发的时代,其中以 IBM San Jose 研究室开发的 System R 和 Berkeley 大学研制的 INGRES 为典型代表。经过大量的高层次的研究和开发取得了一系列的成果,主要是:

(1) 奠定了关系模型的理论基础,给出了人们一致接受的关系模型的规范说明。

(2) 研究了关系数据语言,有关系代数、关系演算、SQL 语言及 QBE 等。这些描述性语言一改以往程序设计语言和网状、层次数据库系统中数据库语言的风格,以其易学易懂的优点得到了最终用户的喜爱,为 20 世纪 80 年代数据库语言标准化打下了基础。

(3) 研制了大量的 RDBMS 的原型,攻克了系统实现中查询优化、并发控制、故障恢复等一系列关键技术。不仅大大丰富了 DBMS 实现技术和数据库理论,更重要的是促进了 RDBMS 产品的蓬勃发展和广泛应用。

在计算机领域中把 20 世纪 70 年代称为数据库时代。20 世纪 80 年代几乎所有新开发的系统均是关系的。关系数据库系统从实验室走向了社会,数据库技术日益广泛地应用到企业管理、情报检索、辅助决策等各个方面,成为实现和优化信息系统的基础和基本技术。

4. 第三代数据库系统的主要特点是什么?

答

经过多年的研究和讨论,对第三代数据库系统的基本特征已有了共识。

(1) 第三代数据库系统应支持数据管理、对象管理和知识管理

除提供传统的数据管理服务外,第三代数据库系统将支持更加丰富的对象结构和规则,应该集数据管理、对象管理和知识管理为一体。由此可以导出,第三代数据库系统必须支持 OO 数据模型。

(2) 第三代数据库系统必须保持或继承第二代数据库系统的技术

第三代数据库系统应继承第二代数据库系统已有的技术,如第二代数据库系统的非过程化数据存取方式和数据独立性。不仅能很好地支持对象管理和规则管理,而且能更好地支持原有的数据管理,支持多数用户需要的即席查询等。

(3) 第三代数据库系统必须对其他系统开放

数据库系统的开放性表现在:支持数据库语言标准;在网络上支持标准网络协议;系统具有良好的可移植性、可连接性、可扩展性和可互操作性等。

5. 试述数据模型在数据库系统发展中的作用和地位。

答

(1) 数据模型是数据库系统的核心和基础。

(2) 数据库的发展集中表现在数据模型的发展。

6. 请用实例阐述数据库技术与其他学科的技术相结合的成果。

答

数据库技术与其他学科的内容相结合,是新一代数据库技术的一个显著特征,涌现出各种新型的数据库系统(如图所示)。例如:

(1) 数据库技术与分布处理技术相结合,出现了分布式数据库系统;

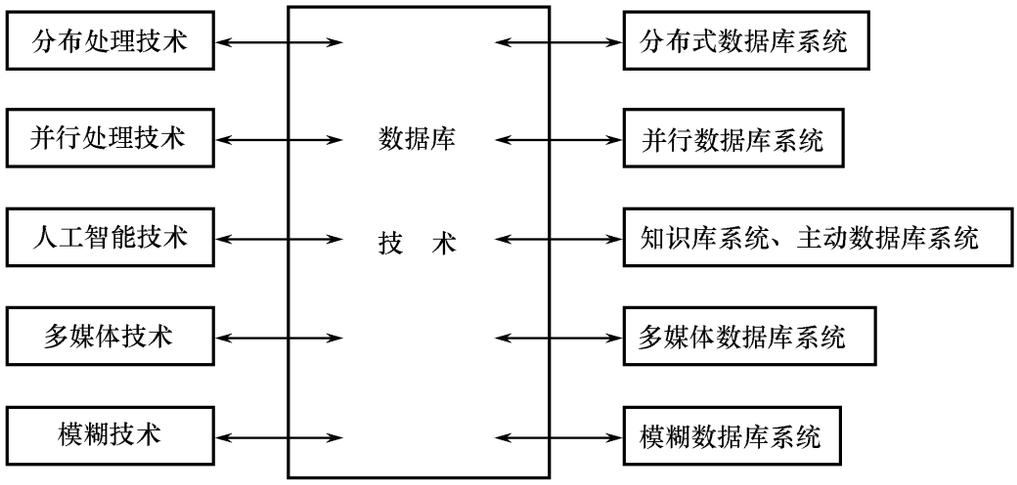
(2) 数据库技术与并行处理技术相结合,出现了并行数据库系统;

(3) 数据库技术与人工智能技术相结合,出现了知识库系统和主动数据库系统;

(4) 数据库技术与多媒体技术相结合,出现了多媒体数据库系统;

(5) 数据库技术与模糊技术相结合,出现了模糊数据库系统;

等等。



7. 请阐述以下数据库系统的主要概念、研究的主要问题及其发展过程:分布式数据库系统、并行数据库系统、主动数据库系统、多媒体数据库系统、模糊数据库系统。

答

下面仅仅给出有关概念,它们研究的主要问题及其发展过程请参见《概论》相关内容。

分布式数据库系统:分布式数据库是由一组数据组成的,这组数据分布在计算机网络的不同计算机上,网络中的每个结点具有独立处理的能力(称为场地自

治),可以执行局部应用。同时,每个结点也能通过网络通信子系统执行全局应用。参见《概论》12.2.2与14.1.1。

并行数据库系统:并行数据库系统是在并行机上运行的具有并行处理能力的数据库系统。并行数据库系统是数据库技术与并行计算技术相结合的产物。参见《概论》12.2.2与15.1。

主动数据库系统:主动数据库是相对于传统数据库的被动性而言的。主动数据库能根据数据库的当前状态,主动适时地做出反应,执行某些操作,向用户提供有关信息。主动数据库是传统数据库技术与人工智能技术、面向对象技术相结合的产物。参见《概论》12.2.2。

多媒体数据库系统:可实现对格式化和非格式化的多媒体数据的存储、管理和查询的数据库系统。参见《概论》12.2.2。

模糊数据库系统:存储、组织、管理和操作模糊数据的数据库系统。参见《概论》12.2.2。

8. 试述数据仓库的产生背景。

答

(1) 数据库技术的发展和广泛应用使许多部门、企业积累了大量的原始数据,这些数据是宝贵的资源。

(2) 对这些数据的分析和利用可以了解企业运行的情况,发现存在的问题,预测未来的趋势。

(3) 数据库系统作为数据管理的先进技术已经成功用于事务处理。但是它对分析处理的支持一直不能令人满意,具体表现在:

1) 分析处理时性能低;

2) 分析的数据对象分散,而且不一致,即缺乏对数据的清洗、集成能力;

3) 事务处理系统不具备动态集成的能力;

4) 系统缺乏对历史数据的有效组织和存储能力,而分析方法必须以大量的历史数据为依托;

5) 在事务处理系统中存储的是细节数据,不适合进行分析处理,而事务处理系统又不具备对数据的综合能力。

总之,DSS对数据在空间和时间的广度上都有了更高的要求,而事务处理环境难以满足这些要求。在事务型环境中直接构建分析型应用是一种失败的尝试。数据仓库正是为了构建这种新的分析处理环境而出现的一种数据存储和组织技术。但是数据仓库的主要驱动力并不是过去的缺点,而是市场商业经营行为的改变,市场竞争要求捕获和分析事务级的业务数据。详细参见《概论》12.2.3。

9. 数据仓库数据的基本特征是什么?

答

4 个基本特征是:

(1) 数据仓库的数据是面向主题的;

(2) 数据仓库的数据是集成的;

(3) 数据仓库的数据是不可更新的;

(4) 数据仓库的数据是随时间不断变化的。

参见《概论》12 2 3。

10. 什么是联机分析处理?什么是数据挖掘?

答

联机分析处理 OLAP 是以海量数据为基础的复杂分析技术。OLAP 支持各级管理决策人员从不同的角度、快速灵活地对数据仓库中的数据进行复杂查询和多维分析处理,并且能以直观易懂的形式将查询和分析结果提供给决策人员,以方便他们及时掌握企业内外的情况,辅助各级领导进行正确决策,提高企业的竞争力。

数据挖掘是从超大型数据库(VLDB)或数据仓库中发现并提取隐藏在内的模式的过程,这些模式是有效的、新颖的、有潜在使用价值的和易于理解的。目的是帮助决策者寻找数据间潜在的关联,发现经营者忽略的要素,而这些要素对预测趋势、决策行为也许是十分有用的信息。

详细参见《概论》12 2 3。

11. 基于数据库技术的 DSS 解决方案是什么?

答

基于数据库技术的 DSS 的解决方案是:

DW + OLAP + DM→DSS 的可行方案

数据仓库、联机分析处理和数据挖掘是作为三种独立的信息处理技术出现的。数据仓库用于数据的存储和组织,OLAP 集中于数据的分析,数据挖掘则致力于知识的发现。由于这三种技术内在的联系性和互补性,将它们结合起来是一种新的 DSS 构架,是 DSS 有效而可操作的整体解决方案。

详细参见《概论》12 2 3。

12. 什么是工程数据库?

答

工程数据库是一种能存储和管理各种工程设计图形和工程设计文档,并能为工程设计提供各种服务的数据库。

主要应用于 CAD/ CAM,CIM,CASE 等工程应用领域。

工程数据库中,由于传统的数据模型难以满足工程应用的要求,需要运用新的模型技术,如扩展的关系模型、语义模型、面向对象的数据模型。

工程数据库管理系统的功能与传统数据库管理系统有很大不同,详细参见《概论》12.2.3。

13. 什么是统计数据库?

答

统计数据库是一种用来对统计数据进行存储、统计、分析的数据库系统。

统计数据具有层次型特点,但并不完全是层次型结构。统计数据也有关系型特点,但关系型也不完全满足需要。统计数据具有一些特殊的性质,例如:

(1) 分类属性和统计属性;

(2) 多维性;

(3) 分类属性的层次结构;

(4) 微数据和宏数据之分。

统计数据库中常用的操作有:抽样、邻近搜索、估计与插值、转置、聚集及复杂的分析操作。这些操作不同于关系数据库中传统的查询、增加、删除、修改操作。人们希望能从 DMBS 一级来支持以上的数据特性和操作,因此,研究和发展了统计数据库技术。

统计数据库在安全性方面有特殊的要求,要防止某些用户在统计数据库中利用对统计数据(如综合数据)的合法查询推导出该用户无权了解的某一个体的具体数据。已在《概论》9.3 节中做了介绍。

14. 什么是空间数据库?

答

空间数据库系统是描述、存储和处理空间数据及其属性数据的数据库系统。空间数据是用于表示空间物体的位置、形状、大小和分布特征等诸方面信息的数据。

空间数据的特点是不仅包括物体本身的空间位置及状态信息,还包括表示物体的空间关系(即拓扑关系)的信息。

空间数据库是随着地理信息系统(GIS)的开发和应用而发展起来的数据库新技术。目前,空间数据库系统不是独立存在的系统,它是和应用紧密结合,大多数作为地理信息系统的基础和核心的形式出现。

空间数据库的研究涉及计算机科学、地理学、地图制图学、摄影测量与遥感、图像处理等多个学科。空间数据库技术研究的主要内容包括:

(1) 空间数据模型;

(2) 空间数据查询语言;

(3) 空间数据库管理系统;

等等。详细参见《概论》12.2.3。

第十三章　面向对象数据系统

习题解答和解析

1. 面向对象程序设计的基本思想是什么?它的主要特点是什么?

答

面向对象程序设计的基本思想是封装和可扩展性。

封装的特点:

面向对象程序设计就是把数据结构和数据结构上的操作算法封装在一个对象之中。

对象是以对象名封装的数据结构和可施加在这些数据上的私有操作。对象的数据结构描述了对象的状态,对象的操作是对象的行为。

面向对象程序设计中,操作名列在封装对象的界面上,当其他对象要启动它的某个操作时,以操作名发一条消息,该对象接受消息,操作动作起来,完成对私有数据的加工。当一个面向对象的程序运行完毕时,各对象也就达到了各自的终态。输入、输出也由对象自己完成。

这种全封装的计算实体给软件带来了模块性、安全性等显著优点。因为它基本没有数据耦合,对象间没有因操作而产生的边界效应,出了错可以很快找到原因,所以易于维护和修改。

可扩展性的特点:

面向对象程序设计的可扩展性体现在继承性和行为扩展两个方面。

因为对象具有一种层次关系。每个对象可以有子对象。子对象可以继承父对象(及其祖先对象)的数据结构和操作,继承的部分就可以重用。

另一方面子对象还可以增加新的数据结构和新的操作。新增加的部分就是子对象对父对象发展的部分。

面向对象程序设计的行为扩展是指可以方便地增加程序代码来扩展对象的行为而不会影响该对象上的其他操作。

详细参见《概论》13.1。

2. 定义并解释 00 模型中以下核心概念: 对象与对象标识、封装、类、类层次。

答

(1) 对象与对象标识 OID

现实世界的任一实体被模型化为一个对象,每个对象有一个惟一的标识,称为对象标识。

(2) 封装

每一个对象是其状态与行为的封装,其中状态是该对象一系列属性值的集合,而行为是在对象状态上操作的集合,操作也称为方法。

(3) 类

共享同样属性和方法集的所有对象构成了一个对象类简称类,一个对象是某一类的一个实例。类的属性的定义域可以是任何类,即可以是基本类也可以是包含属性和方法的一般类,也还可以是这个类自身。

(4) 类层次

在一个面向对象数据库模式中,可以定义一个类(如 C1)的子类(如 C2),类 C1 称为类 C2 的超类(或父类)。子类(如 C2)还可以再定义子类(如 C3)。这样,面向对象数据库模式的一组类形成一个有限的层次结构,称为类层次。

详细参见《概论》13.2.1。

3. OO 模型中对象标识与关系模型中的"码"有什么区别?

答

对象标识具有永久持久性。一个对象一经产生,系统就给它赋于一个在全系统中惟一的对象标识符,直到它被删除。对象标识是由系统统一分配的,用户不能对对象标识符进行修改。对象标识是稳定的,独立于值的,它不会因为对象中某个值的修改而改变。

关系模型中的"码"是值标识,不具有永久持久性,只具有程序内持久性。码是由用户建立的,用来区分关系的不同元组。

详细参见《概论》13.2.2。

4. 请给出一个 OO 数据库类层次(包括各类对应的属性)实例。

答

以工厂数据库为例:

各类对应的属性为:

人:身份证号、姓名、年龄、性别、住址

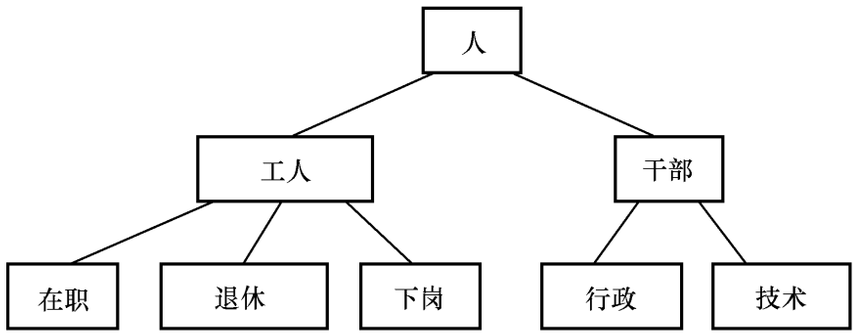
工人:工龄、工种、级别、工资

干部:参加工作时间

在职工人:所在车间

退休工人:退休年份

下岗工人:下岗时间、是否再就业



工厂数据库的类层次结构图

行政干部:职务、级别、办公室地址

技术干部:职称、任职时间

5. 举例说明超类和子类的概念。

答

在第 4 题例子中:

工人和干部是人的子类,在职工人、退休工人、下岗工人是工人的子类,行政干部、技术干部是干部的子类;

人是工人和干部的直接超类。人也是在职工人、退休工人、下岗工人、行政干部、技术干部的间接超类。

6. 什么是单继承?什么是多重继承?继承性有什么优点?

答

若一个子类只能继承一个超类的特性(包括属性、方法和消息),这种继承称为单继承;若一个子类能继承多个超类的特性,这种继承称为多重继承。

继承性优点:

(1) 它是建模的有力工具,提供了对现实世界简明而精确的描述;

(2) 它提供了信息重用机制,由于子类可以继承超类的特性,这就可以避免许多重复定义。

详细参见《概论》13.2.4。

7. 什么是操作的重载?在 OODB 中为什么要滞后联编?

答

在 OO 模型中对于同一个操作,可以按照类的不同,重新定义操作的实现,这称为操作的重载。这样,同一个操作名就与不同的实现方法、不同的参数相联系。

为了提供这个功能,OODBMS 不能在编译时就把操作名联编到程序上,必须在运行时根据实际请求中的对象类型和操作来选择相应的程序,把操作名与它联编上(即把操作名转换成该程序的地址),这个推迟的转换称为滞后联编。

详细参见《概论》13.2.5。

8. 什么是 OODB 模式演进？为什么面向对象数据库模式的修改要比关系模式的修改复杂得多？

答

面向对象数据库模式是类的集合。模式为适应需求的变化会随着时间而变化,这称为模式演进。模式演进包括创建新的类,删除旧的类,修改类的属性和操作等。

面向对象数据库模式的修改要比关系模式的修改复杂的原因是:

(1) 模式改变频繁

使用 OODB 系统的应用通常需要频繁地改变 OODB 数据库模式。例如 OODB 经常运用于工程设计环境中,设计环境特征之一就是不断变化。

(2) 模式修改复杂

OO 模型具有很强的建模能力和丰富的语义,包括类本身的语义、类属性之间和类之间丰富的语义联系,这使得模式修改操作的类型复杂多样。

(3) OODB 中模式演进往往是动态的,使得实现技术更加复杂。

详细参见《概论》13.4。

9. 什么是对象-关系数据库？它的主要特点是什么？常用的实现方法有哪些？

答

对象-关系数据库系统是将关系数据库系统与面向对象数据库系统两方面的特征相结合,不仅支持核心的面向对象数据模型,而且支持传统数据库系统所具有的特征。

主要特点有:

(1) 具有原来关系数据库的各种特点;

(2) 扩充数据类型;

(3) 支持复杂对象;

(4) 支持继承的概念;

(5) 提供通用的规则系统。

实现对象—关系数据库系统的方法主要有以下五类。

(1) 从头开发对象—关系 DBMS。

(2) 在现有的关系型 DBMS 基础上进行扩展。扩展方法有两种:

1) 对关系型 DBMS 核心进行扩充,逐渐增加对象特性。

2) 不修改现有的关系型 DBMS 核心,而是在现有关系型 DBMS 外面加一个包装层。

(3) 将现有的关系型 DBMS 与其他厂商的对象-关系型 DBMS 连接在一起,

使现有的关系型 DBMS 直接而迅速地具有了对象 - 关系特征。连接方法主要有两种:

① 关系型 DBMS 使用网关技术与其他厂商的对象 - 关系型 DBMS 连接。

② 将对象 - 关系型引擎与关系型存储管理器结合起来,即以关系型 DBMS 作为系统的最底层,对象 - 关系型系统作为上层。

(4) 将现有的 OO 型 DBMS 与其他厂商的对象 - 关系型 DBMS 连接在一起,使现有的面向对象型 DBMS 直接而迅速地具有了对象 - 关系特征。

(5) 扩充现有的面向对象的 DBMS,使之成为对象 - 关系型 DBMS。

详细参见《概论》13.5。

第十四章　分布式数据库系统

习题解答和解析

1. 什么样的数据库系统是分布式数据库系统？《概论》图 14.1 的系统配置在什么情况下只能算分散的数据库系统？在什么条件下才是分布式数据库系统？

答

分布式数据库是由一组数据组成的，这组数据分布在计算机网络的不同计算机上，网络中的每个结点具有独立处理的能力(称为场地自治)，可以执行局部应用。同时，每个结点也能通过网络通信子系统执行全局应用。

分布式数据库定义的要点：分布性、逻辑整体性、自治性和协作性。

解析

(1) 分布性，数据库中的数据不是存储在同一场地上，这就可以和集中式数据库相区别。

(2) 逻辑整体性，这些数据逻辑上是互相联系的，是一个整体，逻辑上如同集中数据库。

(3) 自治性，分布数据库中每个结点上的 DBMS 具有独立处理的能力(如果没有连入网络，也是一个完整的 DBMS)。

(4) 协作性，分布数据库中各个结点上的 DBMS 能相互协调，执行全局应用。

答

《概论》图 14.1 中，如果用户既可以通过客户机对本地服务器中的数据库执行局部应用，也可以对两个或两个以上结点中的数据库执行全局应用，这样的系统是分布式数据库系统。不支持全局应用的系统不能称为分布式数据库系统，即只是分散的数据库系统。

详细说明参见《概论》14.1.1。

2. 分布式数据库系统有什么特点？

答

分布式数据库系统是在集中式数据库系统技术的基础上发展起来的，但不

是简单地把集中式数据库分散地实现,它是具有自己的性质和特征的系统。

(1) 数据独立性:除了数据的逻辑独立性与物理独立性外,还具有数据分布独立性亦称分布透明性。

(2) 集中与自治相结合的控制结构:各局部的 DBMS 可以独立地管理局部数据库,具有自治的功能。同时又有集中控制机制,协调各局部 DBMS 的工作,执行全局应用。

(3) 数据可以适当冗余以提高系统的可靠性、可用性和性能。

(4) 全局的一致性、可串行性和可恢复性。分布式数据库系统中各局部数据库应满足集中式数据库的一致性、并发事务的可串行性和可恢复性。除此以外还应保证数据库的全局一致性、全局并发事务的可串行性和系统的全局可恢复性。

详细说明参见《概论》14.1.2。

3. 试述研制分布式数据库系统的目的和动机。

答

研制分布式数据库系统的目的和动机,主要包括技术和组织两方面。

(1) 适应部门分布的组织结构,降低费用;

(2) 提高系统的可靠性和可用性;

(3) 充分利用数据库资源,提高数据库的利用率和共享程度;

(4) 逐步地扩展系统处理能力和系统规模。

详细说明参见《概论》14.1.3。

4. 试述分布式数据库系统的模式结构。

答

分布式数据库系统的模式结构可以分为两大部分:集中式数据库系统的模式结构和分布式数据库系统增加的模式级别,其中包括:

(1) 全局外模式,它们是全局应用的用户视图,是全局概念模式的子集;

(2) 全局概念模式,它定义分布式数据库中数据的整体逻辑结构,使得数据如同没有分布一样;

(3) 分片模式,定义片段以及全局关系到片段的映像;

(4) 分布模式,定义片段的存放结点,分布模式的映像类型确定了分布式数据库是冗余的还是非冗余的。

详细可参考《概论》图 14.3 分布式数据库系统的模式结构。

5. 什么是数据分片?有几种分片方式?数据分片的目的是什么?有什么优点?

答

数据分片就是将数据表按照一定条件划分成若干子集,每个子集称为一个

片段。

分片的方式有多种,水平分片和垂直分片是两种基本的分片方式,混合分片和导出分片是较复杂的分片方式。

水平分片是指按一定的条件将关系表按行(水平方向)分为若干不相交的子集,每个子集为关系的一个片段。

垂直分片是指将关系按列(垂直方向)分为若干子集。垂直分片的各个片段都要包含关系的码。这样才能从各个片段重构原来的关系。

导出分片是指导出水平分片,即水平分片的条件不是本身属性的条件而是其他关系的属性的条件。

混合分片是指按上述三种分片方式得到的片段继续按另一种方式分片。

数据分片的优点是:数据不是按照关系而是按片段来存放,有利于更好地根据用户需求来组织数据的分布,也有利于控制数据的冗余度。

6. 试述分布透明性的内容。

答

分布透明性包括分片透明性、位置透明性和局部数据模型透明性。

分片透明性指用户或应用程序只对全局关系进行操作而不必考虑关系的分片。当分片模式改变了,由于全局模式到分片模式的映像,全局模式不变,应用程序不必改写。

位置透明性指用户或应用程序不必了解片段的存储场地,当存储场地改变了,由于分片模式到分布模式的映像,应用程序不必改变。同时,若片段的重复副本数目改变了,数据的冗余度改变了,用户也不必关心如何保持各副本的一致性,这就是重复副本的透明性。

局部数据模型透明性指用户或用户程序不必了解局部场地上使用的是哪种数据模型。

7. 什么是同构型 D-DBMS?什么是异构型 D-DBMS?

答

D-DBMS 的同构和异构可以有三级:硬件级、操作系统级和局部 DBMS 级。其中最主要的是局部 DBMS 这一级,因为硬件和操作系统的不同将由通信软件处理和管理。所以,同构型 D-DBMS 定义为:在分布数据库系统中若每个结点的局部数据库具有相同的 DBMS,则成为同构型 D-DBMS;若各结点的局部数据库具有不同的 DBMS,则成为异构型的 D-DBMS。

详细说明参见《概论》14.2.4。

8. 设在《概论》14.2.3 的分布式数据库系统例子中,还有全局关系 SC(SNO,CNO,G),它具有两个导出分片 SC_A,SC_B,分别存储理学院和文学院学生的选课记录。SC_A 存放在场地 4,SC_B 存放在场地 5。今有一个稍复杂的查询,

从终端输入一个课程号,查找选修该课程的学生学号和姓名,并把它们显示在屏幕上。请写出具有不同层次分布透明性(类比例子中的三种情况)的应用程序。不必给出细节,只需写出算法思想。

情况1 若系统具有分片透明性,则

```
Scanf( % s ,SCnumber);          /* 从终端读入课程号到变量 SCnumber 中 */
EXEC SQL SELECT Sno,Sname INTO :SNO, :NAME
                                /* SNO,NAME 为程序变量 */
FROM SC,Student                 /* 在全局关系 SC,Student 中查找 */
WHERE SC.Cno =:SCnumber AND SC.Sno = Student.Sno;
Printf( % s,% s ,SNO,NAME);  /* 把 SNO,NAME 输出在屏幕上 */
```

情况2 若系统具有位置透明性,但不具有分片透明性,则

```
Scanf( % s ,SCnumber);
EXEC SQL SELECT Sno,Sname INTO :SNO, :NAME
      FROM SC_A,S_A
      WHERE SC_A.Cno =:SCnumber AND SC_A.Sno=S_A.Sno;
If(!FOUND){
EXEC SQL SELECT Sno,Sname INTO :SNO, :NAME
      FROM SC_B,S_B
      WHERE SC_B.Cno =:SCnumber AND SC_B.Sno=S_B.Sno;
}
Printf( % s,% s ,SNO,NAME);
```

情况3 若系统只具有局部数据模型透明性,不具有位置透明性(当然也就不具有分片透明性),则

```
Scanf( % s ,SCnumber);
EXEC SQL SELECT Sno,Sname INTO :SNO,:NAME
      FROM SC_A AT Site4, S_A AT Site1
      /* 先在场地 4 的片段 SC_A 和场地 1 的片段 S_A 中查找 */
      WHERE SC_A.Cno =:SCnumber AND SC_A.Sno=S_A.Sno;
If(!FOUND){
EXEC SQL SELECT Sno,Sname INTO :SNO,:NAME
      FROM SC_B AT Site5,S_B AT Site2
      /* 再在场地 5 的片段 SC_B 和场地 2 的片段 S_B 中查找*/
      /*也可以在场地 5 的片段 SC_B 和场地 3 的片段 S_B 中查找*/
      WHERE SC_B.Cno =:SCnumber AND SC_B.Sno=S_B.Sno;
}
Printf( % s,% s ,SNO,NAME);
```

9. 对《概论》14.3.1 节的例子中介绍的六种策略改用下列估算值后分别计

算通信时间:

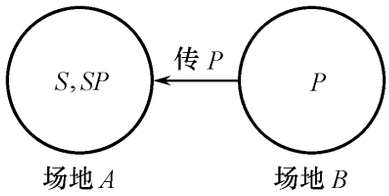
红色零件数 = 1000,

北京供应商的装运单 = 10000。

答

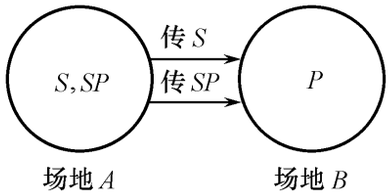
策略 1　把关系 P 传送到场地 A,在 A 地进行查询处理,所以

$T[1] = 1 + 10^5 \times 100/10^4 = 10^3$ 秒(16.7 分)



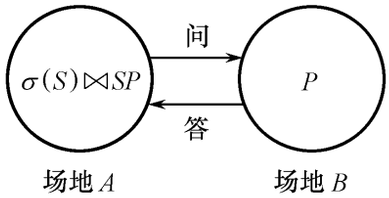
策略 2　把关系 S、SP 传到场地 B,在 B 地执行查询处理,所以

$T[2] = 2 + (10^4 + 10^6) \times 100/10^4 \approx 10\ 100$ 秒(2.8 小时)



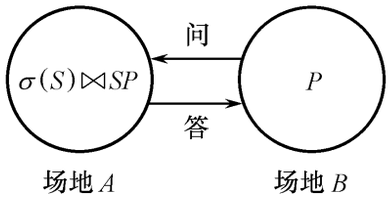
策略 3　在场地 A 连接关系 S 和 SP,选出城市为北京的元组(10^4 个)然后对这些元组中的每个元组的 Pno,询问场地 B 看此零件是否红色。所以共问答 10^4 次,由于不是传送数据,只是消息的问答,所以

$T[3] = 2 \times 10^4$s (5.6 小时)



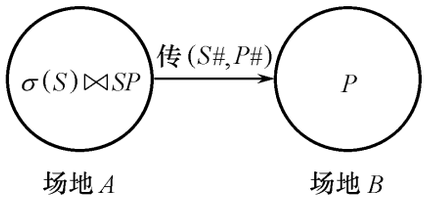
策略 4　在场地 B 选出红色零件的元组(10^3 个),然后对每一个元组逐一检查 A 站,看北京供应商的装运单中是否有这个零件装运单(若有则选出 Sno)每做这样一次检查包括 2 次消息,共问一答 10^3 次,所以

$T[4] = 2 \times 10^3$s (33.3 分)



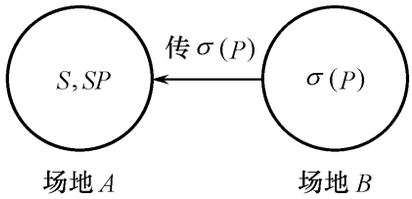
策略 5　在场地 A 选出北京的供应商的装运单把结果送到场地 B,在场地 B 完成最后处理,所以

$T[5] = 1 + (10^4 \times 100)/10^4 \approx 101$ 秒 (1.68 分)



策略 6　在场地 B 的关系 P 中选出红色的元组(1 000 个),把结果送到场地 A 完成最终处理。所以

$T[6] = 1 + (1\ 000 \times 100)/10^4 = 11$ 秒



10. 设关系 R、S 存储在不同的场地,已知:

card(R) = 100, size(R) = 50

card(S) = 50, size(S) = 5

今要计算 $R \underset{A=B}{} S$,其中 A, B 分别为 R, S 上的属性,并且

val(A[R]) = 50, size(A) = 3

val(B[S]) = 50, size(B) = 3

若 $R' = R \underset{A=B}{<} S$, card(R') = 0.2 * card(R)

$S' = S < R$, card(S') = 0.8 * card(S)

请比较采用以下不同策略时的通信代价:

(1) 在 R 所在场地执行连接操作,利用半连接方法;

(2) 在 S 所在场地执行连接操作,利用半连接方法;

(3) 在 R 所在场地执行连接操作,不用半连接方法;

(4) 在 S 所在场地执行连接操作,不用半连接方法。

哪种情况代价最省(不考虑把结果传送到某一场地的代价)?

答

在 R 所在场地执行半连接送到 S 所在场地执行连接的总代价为

$C_{SJ} = 2C_0 + C_1(size(B) \times val(B[S]) + size(R) \times card(R'))$

不用半连接,直接把 R 送到 S 所在场地执行连接的代价为

$C_{JN} = C_0 + C_1 \times size(R) \times card(R)$

(1): $C_{SJ} = 2C_0 + C_1(size(A) \times val(A[R]) + size(S) \times card(S'))$

$= 2C_0 + C_1(3 \times 50 + 5 \times 50 \times 0.8) = 2C_0 + 350\ C_1$

(2): $C_{SJ} = 2C_0 + C_1(size(B) \times val(B[S]) + size(R) \times card(R'))$

$= 2C_0 + C_1(3 \times 50 + 50 \times 100 \times 0.2) = 2C_0 + 1150\ C_1$

(3): $C_{JN} = C_0 + C_1 \times size(S) \times card(S)$

$= C_0 + C_1 \times 5 \times 50 = C_0 + 250\ C_1$

(4): $C_{JN} = C_0 + C_1 \times size(R) \times card(R)$

$= C_0 + C_1 \times 50 \times 100 = C_0 + 5000\ C_1$

所以(3)代价最省。

11. 试述下列概念:两段提交协议(2PC);分布事务的原子性;全局死锁。

答

2PC:2PC 把一个分布事务的事务管理分为协调者和参与者。

2PC 的第一阶段:协调者向所有参与者发出“准备提交”信息。如果某个参与者准备提交,就回答“就绪”信息,否则回答“撤销”信息。参与者在回答前,应把有关信息写入自己的日志中。协调者在发出准备提交信息前也要把有关信息写入自己的日志中。如果在规定时间内协调者收到了所有参与者“就绪”的信息,则将作出提交的决定,否则将作出撤销的决定。

2PC 的第二阶段:协调者将有关决定的信息先写入日志,然后把这个决定发送给所有的参与者。所有参与者收到命令之后首先往日志中写入“收到提交(或

撤销)”决定的信息,并向协调者发送“应答(ACK)”消息,最后执行有关决定。协调者收到所有参与者的应答消息后,一个事务的执行到此结束,有关日志信息可以脱机保存。

分布事务的原子性:分布事务的原子性就应该是:组成一个全局事务的所有子事务要么一致地全部提交,要么一致地全部回滚。

全局死锁:全局事务执行时发生的涉及两个以上场地上的死锁。

12 在分布式数据库系统中,对多副本的封锁有几种解决方法?

答

处理多副本的封锁可采取如下几种方法:

(1) 对写操作,要申请对所有副本的 X 锁;对于读操作,只要申请对某个副本的 S 锁。

(2) 无论是写操作还是读操作都要对多数(大于半数)副本申请 X 锁或 S 锁。

(3) 规定某个场地上的副本为主副本,所有的读/写操作均申请对主副本的封锁。

第十五章　并行数据库系统

习题解答和解析

1. 什么是并行数据库系统？

答

并行数据库系统是在并行机上运行的具有并行处理能力的数据库系统。并行数据库系统是数据库技术与并行计算技术相结合的产物。

2. 试述并行数据库系统的研制目标。

答

并行数据库系统该实现如下目标:

(1) 高性能

并行数据库系统通过将数据库管理技术与并行处理技术有机结合,发挥多处理机结构的优势,提供比相应的大型机系统更高的性能价格比和可用性。

(2) 高可用性

并行数据库系统可通过数据复制来增强数据库的可用性。

(3) 可扩充性

系统通过增加处理和存储能力来平滑地扩展性能,应具有线性伸缩比和线性加速比。

具体说明可参考《概论》15.2。

3. 什么是并行数据库系统的伸缩比和加速比？

答

线性伸缩比是指当任务扩大 N 倍,系统处理和存储能力也扩大 N 倍时系统性能不变。

线性伸缩比 = (小任务在小系统上的运行时间)/ (大(N 倍)任务在大系统上的运行时间) = 1

线性加速比是指当任务不变而系统处理和存储能力扩大 N 倍时,系统性能也提高 N 倍。

线性加速比 = (小系统上执行一个任务的时间)/ (大(N 倍)系统上执行同一任务的时间) = N

4. 并行数据库系统有哪几种体系结构？试比较它们的特点。

答

从硬件结构来看,根据处理机与磁盘、内存的相互关系可以将并行计算机分为三种基本的体系结构:

共享内存结构(SM);

共享磁盘结构(SD);

无共享资源结构(SN)。

此外还有混合结构,即整个系统是 Shared _ Nothing 结构而每个结点是 Shared _ Memory 结构。这种结构综合了 SM 与 SN 的优点。

(1) SM 并行结构

SM 并行结构由多个处理机、一个共享内存(主存储器)和多个磁盘存储器构成。多处理机和共享内存由高速通讯网络连接,每个处理机可直接存取一个或多个磁盘,即所有内存与磁盘为所有处理机共享。

(2) SD 并行结构

SD 并行结构由多个具有独立内存的处理机和多个磁盘构成。每个处理机都可以读/写任何磁盘。

(3) SN 并行结构

SN 并行结构由多个处理结点构成。每个处理结点具有自己独立的处理机、内存和磁盘存储器。多个处理机结点由高速通信网络连接。

并行数据库系统体系结构的详细说明可参考《概论》15.3.1、15.3.2、15.3.3。

三种并行数据库系统体系结构的比较如《概论》上表 15.1:

	Shared _ Memory	Shared _ Disk	Shared _ Nothing
性能	最佳	较佳	较佳
可用性	低	较高	高
可扩充性	差	较好	好
负载均衡	易作到	易作到	难作到
实现技术	容易	较复杂	复杂
成本	高	较低	低
处理机数	数十个	数百个	数千个
规模	中小系统	中小系统	大系统

5. 阐述并行数据库系统中并行粒度的概念。

答

并行粒度是用来刻画查询执行的并行程度,有四种并行粒度:不同用户事务间的并行性、同一事务内不同查询间的并行性、同一查询内不同操作间的并行性

和同一操作内的并行性。

事务(Transaction)			事务内 intra -	事务间 inter -
查询(Query)		查询内 intra	查询间 inter -	
操作(Operation)	操作内 intra -	操作间 inter		
并行粒度	细			粗

具体说明可参考《概论》15.4.1。

6. 举例说明水平并行和垂直并行的概念。

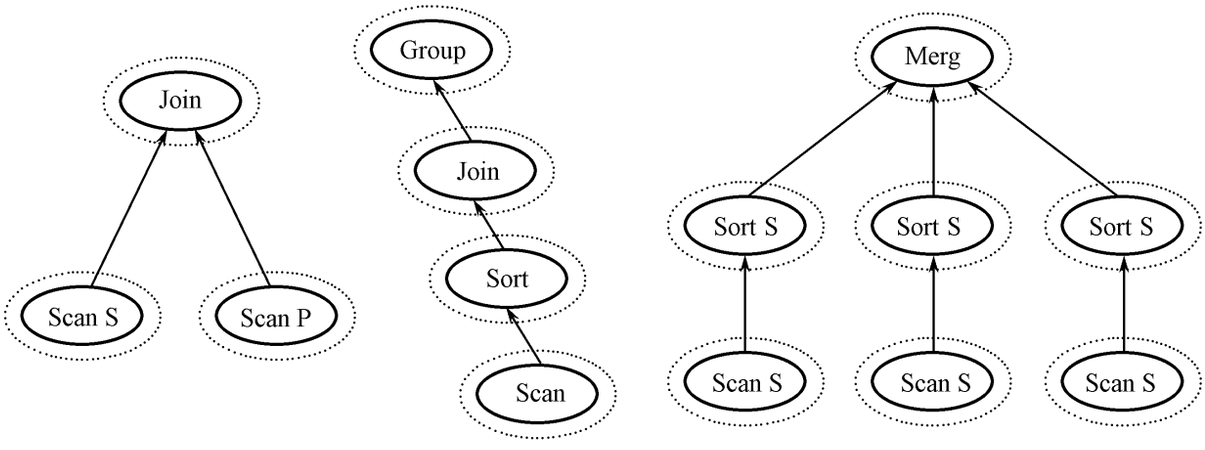
答

水平并行化指:互相独立的多个操作或者一个操作内互相独立的多个子操作分别由不同的处理机并行执行的形式。

如果两个操作 OP1 和 OP2 之间没有数据的依赖关系,这两个操作可以独立地并行执行。例如,图(a)中对不同表的扫描操作 Scan S 和 Scan P,由不同的处理机并行执行就是水平并行化的实例。

如果把操作的输入数据分解为多个子集,该操作就可以分成多个独立的子操作,每个子操作只处理一部分数据,这种针对单个操作的并行被称为操作内并行。例如,图(c)中的 Scan S 和 Sort S 都被分解成 3 个子操作,每个子操作只对部分数据进行扫描和排序。

如果一个操作的输出流是另一个操作的输入流,这两个操作之间就存在着流水线并行性,操作间流水线并行又被直观地称为垂直并行。垂直并行化也就是指存在流水线方式依赖关系的操作分别由不同处理机并行执行的形式。例如,图(b)中的 4 个操作都可以采用流水线方式并行执行。



(a) 操作间独立并行 (b) 操作间流水线并行 (c) 操作内并行

详细说明可参考《概论》15.4.2。

7. 并行数据库系统中并行查询优化的必要性和困难何在？

答

查询优化始终是数据库管理系统的重要组成部分,查询优化的目标在于提高执行效率。由于并行数据库环境中存在多个处理机,并行查询优化应尽可能地使每个操作并行处理,充分利用系统资源提高并行度来达到提高系统性能的目的。

并行查询优化面临的两大困难在于:

(1) 执行计划的搜索空间十分庞大;

(2) 执行时的某些系统参数比如 CPU 数目、内存大小在优化时是未知的。

具体说明可参考《概论》15.4.4。

8. 试述数据划分在并行查询处理中的重要性。

答

数据划分是并行查询处理的重要基础。研究和实践表明,数据划分对于并行数据库系统的性能具有很大的影响。

通过将每个关系的数据划分为小的片段,并把这些小片段均匀地分布在系统的多个磁盘驱动器上可以降低数据的聚集度,使得每个操作能够由多个处理机来承担,从而减少查询的响应时间并提高整个系统的吞吐量。若负载不均,往往会造成多个处理机结点能力的浪费。所以使用正确的数据分布算法以达到负载均衡是并行数据库中数据分布的关键问题。

具体说明可参考《概论》15.5.1。

9. 并行数据库系统中有哪几种常用的数据划分方法？

答

划分数据时可以依据一个属性的值,也可以同时依据多个属性的值,前者称为一维数据划分,后者则称为多维数据划分。

一维数据划分方法相对比较简单,常用的数据划分方法有:

(1) 轮转法;

(2) Hash 法;

(3) 值域划分法。

此外,还有用户定义的划分法、模式划分法、Hybrid_Range 划分法等。

具体说明可参考《概论》15.5.1。

10. 试述并行数据库系统与分布式数据库系统的区别。

答

分布式数据库系统与并行数据库系统特别是与 SN 结构的并行数据库系统具有很多相似点:

(1) 它们都是用网络连接各个数据处理结点;

(2) 整个网络中的所有结点构成一个逻辑上统一的整体;

(3) 用户可以对各个结点上的数据进行透明存取。

分布式数据库系统和并行数据库系统的应用目标和具体实现方法不同,使得它们具有很大的不同:

(1) 应用目标不同

并行数据库系统的目标是充分发挥并行计算机的优势,利用各个处理机结点并行地完成任务,提高系统的整体性能。

分布式数据库系统的目标是实现场地自治和数据的全局透明共享,而不要求利用网络中的各个结点来提高系统处理性能。

(2) 实现方式不同

在并行数据库系统中各结点间采用高速网络互连,结点间的数据传输代价相对较低,因此当某些结点处于空闲状态时,可以将工作负载过大的结点上的部分任务通过高速网传送给空闲结点处理,从而实现系统的负载平衡。

在分布式数据库系统中,各结点间一般采用局域网或广域网相连,网络带宽较低,点到点的通信开销较大,因此在查询处理时一般应尽量减少结点间的数据传输量。

(3) 各结点的地位不同

在并行数据库系统中,不存在全局应用和局部应用的概念,各结点是非独立的。而在分布式数据库系统中,各结点除了能通过网络协同完成全局事务外,更重要的是各结点具有场地自治性。

具体说明可参考《概论》15.6。

附录 A　数据库系统概论课程模拟试卷

模拟试卷一

一、判断题

判断下列模式分别属于哪个范式(最高范式)并说明理由。

1. R({A,B,C},{(A,C)→B ,(A,B)→C,B→C})

2. R({S#,SD,SL,SN},{S#→SD,S#→SN,S#→SL,SD→SL})

二、判断题

判断下题中给出的命题是否正确,若不对,请给出你认为正确的答案。

如一组事务是按一定顺序执行的,则称这组事务是可串行的。

三、简答题

1. 在数据库中为什么要有并发控制?

2. 试述数据库中完整性的概念、类型及你所了解的系统完整性检查方法。

3. 什么是数据模型?试述其组成部分。

4. 什么是数据库系统的三级模式结构?这种体系结构的优点是什么?

5. 什么是日志文件?简述用日志文件恢复事务的过程。

四、求解题

某医院病房计算机管理中需要如下信息:

科室:科名,科地址,科电话,医生姓名

病房:病房号,床位号,所属科室名

医生:姓名,职称,所属科室名,年龄,工作证号

病人:病历号,姓名,性别,诊断,主管医生,病房号

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

完成如下设计:

(1) 涉及该计算机管理系统的 E - R 图;

(2) 将该 E - R 图转换为关系模型的结构;

(3) 指出转换结果中每个关系模式的候选码。

五、求解题

设有关系模式 R(C,T,S,N,G),其中 C 代表课程,T 代表教师的职工号,S 代表学生号,N 代表学生的姓名,G 代表分数(成绩)。其函数依赖集 F = {C→T,CS→G,S→N},即每一门课由一名教师讲授,每个学生每门课只有一个成绩,学生的学号决定学生的姓名。试求:

1. 该关系模式的候选码(应根据候选码的定义,并给出所求的过程);

2. 将该模式分解成既符合 BCNF,又具有无损连接的若干关系模式(要求给出过程);

3. 将 R 分解成 R_1(C,T,S,G)和 R_2(C,S,N,G)试说明它们各符合第几范式。

六、问答题

图书流通数据库中的 3 个关系:读者关系、图书关系、借书关系,它们所含的属性及码分别为:

READER(CARDNO,SNAME,DEPT),KEY = CARDNO

BOOKS(BCALLNO, TITLE, AUTHOR, BOOKNO, PUBHOU, PRICE), KEY = BCALLNO

LOANS(CARDNO, BCALLNO,DATE),KEY = (CARDNO, BCALLNO,)

其中:CARDNO——借书证号

SNAME——姓名

DEPT——单位

BOOKNO——图书登记号(一本书对应一个图书登记号,例如《数据库系统概论》有一个图书登记号 RD DB 1801)

DATE——借书日期

BCALLNO——索书号(借出一本书有一个索书号,例如图书馆中有 60 本《数据库系统概论》,有 60 个索书号,例如从 RD DB 1801 001 到 RD DB 1801 060)

TITLE——书名

AUTHOR——作者

PUBHOU——出版单位

PRICE——价格

要求用关系代数和 SQL 分别表示如下查询:

1. 查询借阅“数据库”的读者姓名;

2. 找出 94.1.1 前被借出的书的书名和作者;

3. 作者“王平”所著“操作系统”书共借出几本?(仅用 SQL)

模拟试卷一参考答案

一、判断题

1. 1NF。

由题目可知,关系的侯选码为(A, C)和(A, B)。B→C 表明存在对码的部分依赖,所以这只能是 1NF。

2. 2NF。

由题目可知,关系的码为 S# 。这里存在对码的传递依赖。

二、判断题

错误。

根据可串行化的定义,多个事务并发执行时,当且仅当其执行的结果与这一组事务按某一次序串行地执行结果相同,才能称这种调度策略为可串行化。各种调度的策略会产生不同的结果,但未必与串行的结果相同,所以它们不都是可串行的。

三、简答题

1.

数据库是一个共享资源,它允许多个用户同时存取修改同一个数据。若系统对并行操作不加控制,就可能产生错误的结果,如存取和存储不正确的数据,破坏数据库一致性等。并发控制的目的,就是要以正确的方式调度并发操作,避免造成各种不一致性,使一个事务的执行不受另一个事务的干扰。

2.

数据库的完整性是指数据的正确性和相容性,为了防止不合语义的数据进入数据库。完整性的类型一般可以分为六类:静态列级约束、静态元组约束、静态关系约束、动态列级约束、动态元组约束、动态关系约束。

系统完整性检查方法有多种,例如,在一条语句执行完后立即检查是否违背完整性约束,即立即执行完整性检查。有时完整性检查延迟到整个事务执行结束后再进行,检查正确方可提交,即延迟执行约束完整性检查。

3.

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。不同的数据模型是提供给我们模型化数据和信息的不同工具。根据模型应用的不同目的,可以将模型分成两类或两个层次:

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;

另一是数据模型,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、操纵数据库中的数据。一般需要有严格的形式化定义和一组严格定义了语法和语义的语言,并有一些规定和限制,便于在机器上实现。

一般地讲,数据模型是严格定义的概念的集合。这些概念精确地描述系统的静态特性、动态特性和完整性约束条件。因此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) 数据结构是所研究的对象类型的集合,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、相容。

4.

数据库系统的三级模式结构由外模式、模式和内模式组成。

外模式,亦称子模式或用户模式,是数据库用户看到的数据视图。

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。

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。

模式描述的是数据的全局逻辑结构。外模式涉及的是数据的局部的逻辑结构,通常是模式的子集。

这种体系结构的优点:数据库系统的三级模式是对数据的三个抽象级别,它把数据的具体组织留给 DBMS 管理,使用户能逻辑抽象地处理数据,而不必关心数据在计算机中的表示和存储。而为了能够在内部实现这 3 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层映像:外模式/ 模式映像和模式/ 内模式映像。正是这两层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。

5.

日志文件是用来记录事务对数据库的更新操作的文件。

用日志文件恢复事务(即事务故障的恢复)的过程如下:

(1) 反向扫描文件日志(从最后向前扫描日志文件),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。如果日志记录中是插入操作,则做删除操作;若日志记录中是删除操作,则做插入操作;若是修改操作,则用修改前值代替修改后值。

(3) 继续反向扫描日志文件,查找该事务的其他更新操作,并做同样处理。

(4) 如此处理下去,直至读到此事务的开始标记,事务故障恢复就完成了。

四、求解题

(1) 本题的 E-R 图如下所示

(2) 对应的关系模型结构如下:

科室(科名,科地址,科电话);

病房(病房号,床位号,科室名);

医生(工作证号,姓名,职称,科室名,年龄);

病人(病历号,姓名,性别,诊治,主管医生,病房号)。

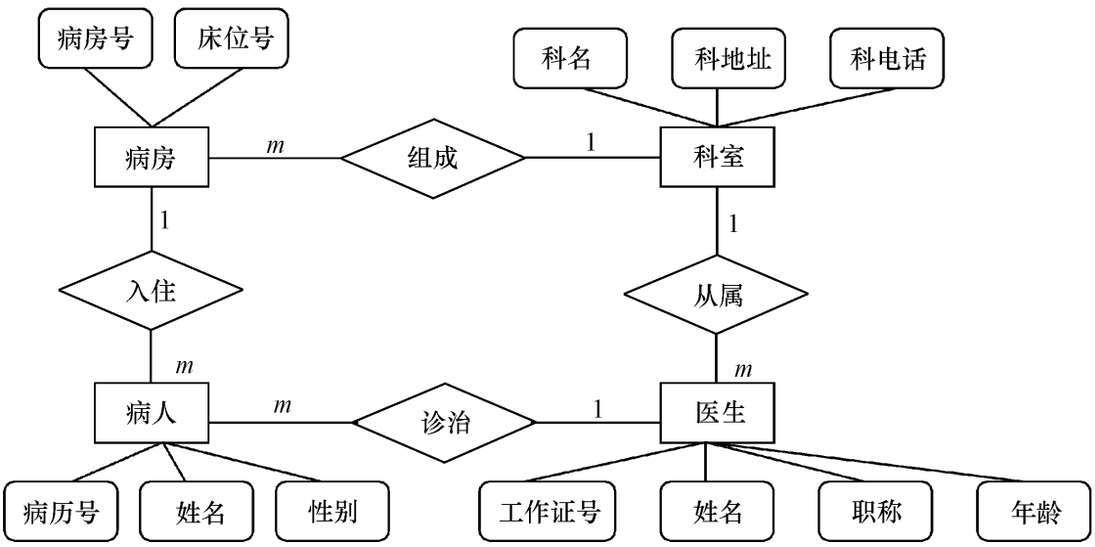
(3) 每个关系模式的候选码如下:

科室的候选码是科名;

病房的候选码是病房号 + 床位号;

医生的候选码是工作证号;

病人的候选码是病历号。



五、求解题

1.

只有一个码 CS

求解过程:令 $U=\{C,S,T,N,G\}$, $CF_F^+=\{C,T\}$, $S_F^+=\{S,N\}$, $CS_F^+=\{C,S,T,G,N\}=U$;所以只有一个码 CS。

2.

分解成 $R_1(C,T)$ $R_2(S,N)$ $R_3(C,S,G)$

求解过程:按照“分解法”,步骤依次为

step1:因为 C→T 不满足 BCNF,所以令 $U_{11}=\{C,T\}$,$U_{12}=\{C,S,N,G\}$;

step2:因为 S→N 不满足 BCNF,所以令 $U_{21}=\{S,N\}$,$U_{22}=\{C,S,G\}$;

step3:因为 CS→G 满足 BCNF,算法停止,$U_{31}=\{C,S,G\}$;

U_{11},U_{21},U_{31}即为分解结果。

3.

R_1 与 R_2 都为 1NF,因为都存在非主属性对码的部分函数依赖。

R_1 的码是 CS,CS→T ,而 R_1 中有 C→T,是部分函数依赖。

同样,R_2 的码是 CS,CS→N ,而 R_2 中有 S→N,是部分函数依赖。

六、问答题

1.

```
SELECT SNAME
FROM READER, BOOKS, LOANS
WHERE READER.CARDNO = LOANS.CARDNO AND
      LOANS.BCALLNO = BOOKS.BCALLNO AND
          BOOKS.TITLE =  数据库
```

$$\sigma_{books.title=\text{数据库}}(reader \underset{reader.cardno=loans.cardn}{} loans \underset{loans.bcallno=books.bca}{} books)$$

2.

```
SELECT DISTINCT (TITLE, AUTHOR)
FROM BOOKS, LOANS
WHERE BOOKS.BCALLNO = LOANS.BCALLNO AND DATE < 940101
```

$$\sigma_{data<940101}(loans \underset{books.bcallno=loans.bcallno}{} books)$$

3.

```
SELECT COUNT ( * )
FROM LOANS, BOOKS
WHERE BOOKS.BCALLNO =LOANS.BCALLNO AND
      TITLE =  操作系统 AND AUTHOR =  王平
```

模拟试卷二

一、选择题

1. 五种基本关系代数运算是【】

A. ∪, -, ×, π和σ

B. ∪, -, ∞, π和σ

C. ∪, ∩, ×, π和σ

D. ∪, ∩, ∞, π和σ

2. 下列聚集函数中不忽略空值 (null) 的是【】

A. SUM (列名)

B. MAX (列名)

C. COUNT (*)

D. AVG (列名)

3. 设关系模式 R (A, B, C), F 是 R 上成立的 FD 集, F = { B→ C}, 则分解 ρ = { AB, BC}

A. 是无损联接, 也是保持 FD 的分解

B. 是无损联接, 但不保持 FD 的分解

C. 不是无损联接, 但保持 FD 的分解

D. 既不是无损联接, 也不保持 FD 的分解

4. 在数据库设计中, 将 E-R 图转换成关系数据模型的过程属于【】

A. 需求分析阶段

B. 概念设计阶段

C. 逻辑设计阶段

D. 物理设计阶段

5. DBMS 中实现事务持久性的子系统是【】

A. 安全性管理子系统

B. 完整性管理子系统

C. 并发控制子系统

D. 恢复管理子系统

6. 当关系 R 和 S 自然联接时, 能够把 R 和 S 原该舍弃的元组放到结果关系中的操作是【】

A. 左外联接

B. 右外联接

C. 外部并

D. 外联接

二、名词解释

1. 关系模型中的实体完整性、参照完整性

2. 二段锁协议,可串行化调度

三、简答题

1. 什么是数据模型及其要素?

2. 文件系统的特点及其主要缺点是什么?

3. 什么是数据库恢复?简述数据库恢复的基本技术。

四、求解题

在供应商、零件数据库中有以下 3 个关系模式:

供应商:S(SNO,SNAME,CITY,STATUS)

零件:P(PNO,PNAME,WEIGHT,COLOR,CITY)

供应货:SP(SNO,PNO,QTY)

各属性的含义可由属性名体现,不再重复,供应货关系 SP 表示某供应商 SNO,供应了 PNO 零件,数量为 QTY。

用 SQL 语言完成以下操作:

1. 求供应红色零件的供应商名字;

2. 求北京供应商的号码、名字和状况(STATUS);

3. 求零件 P2 的总供应量;

4. 把零件 P2 的重量增加 5,颜色该为黄色。

五、问答题

已知关系模式 R < U,F > , U = {A,B,C,D,E,G} F = {AC→B,CB→D,A→BE,E→GC}

求:AB,BC,AC 是否为关系 R 的候选码?

六、证明题

试证由关系模式中全部属性组成的集合为候选码的关系是 3NF,也是 BCNF。

七、综合题

现有如下关系模式:

其中,Teacher(Tno, Tname, Tel, Dpartment, Bno, Bname, BorrowDate, RDate, Backup)。

Tno—教师编号,

Tname—教师姓名,

Tel—电话,

Department—所在部门,

Bno—借阅图书编号,

Bname—书名,

BorrowDate—借书日期,

RDate—还书日期,

Backup—备注

该关系模式的属性之间具有通常的语义,例如,教师编号函数决定教师姓名,即教师编号是惟一的,图书编号是惟一的,等等。

1. 教师编号是候选码吗?

2. 说明上一题判断的理由是什么。

3. 写出该关系模式的主码。

4. 该关系模式中是否存在部分函数依赖?如果存在,请写出其中两个。

5. 说明要将一个 1NF 的关系模式转化为若干个 2NF 关系,需要如何做?

6. 该关系模式最高满足第几范式?并说明理由。

7. 将该关系模式分解为 3NF。

八、综合题

假设某商业集团数据库中有一关系模式 R 如下:

R (商店编号,商品编号,商品库存数量,部门编号,负责人)

如果规定:

(1) 每个商店的每种商品只在该商店的一个部门销售;

(2) 每个商店的每个部门只有一个负责人;

(3) 每个商店的每种商品只有一个库存数量。

试回答下列问题

(1) 根据上述规定,写出关系模式 R 的基本函数依赖;

(2) 找出关系模式 R 的候选码;

(3) 试问关系模式 R 最高已经达到第几范式?为什么?

(4) 如果 R 不属于 3NF,请将 R 分解成 3NF 模式集。

模拟试卷二参考答案

一、选择题

1. A　　2. C　　3. A　　4. C　　5. D　　6. D

二、名词解释

1.

(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:

1) 或者取空值(F 的每个属性值均为空值);

2) 或者等于 S 中某个元组的主码值。

2.

(1) 两段锁协议是指所有事务必须分两个阶段对数据项加锁和解锁。

1) 在对任何数据进行读、写操作之前,首先要申请并获得对该数据的封锁;

2) 在释放一个封锁之后,事务不再申请和获得任何其他封锁。

“两段”的含义是,事务分为两个阶段:

第一阶段是获得封锁,也称为扩展阶段。在这阶段,事务可以申请获得任何数据项上的任何类型的锁,但是不能释放任何锁。

第二阶段是释放封锁,也称为收缩阶段。在这阶段,事务释放已经获得的锁,但是不能再申请任何锁。

(2) 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,我们称这种调度策略为可串行化的调度。

三、简答题

1.

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。

一般地讲,数据模型是严格定义的概念的集合。这些概念精确地描述系统的静态特性、动态特性和完整性约束条件。因此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:是所研究的对象类型的集合,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、相容。

2.

特点:数据可以长期保存,把数据组织成相互独立的数据文件,利用"按文件名访问,按记录进行存取"的技术,可以对文件进行修改、插入和删除的操作。实现了记录内的结构性,但整体无结构。应用程序和数据有一定的独立性,程序员不必过多考虑物理细节,节省了维护程序的工作量。

缺点:(1) 数据共享性差,冗余度大;(2) 数据独立性差。

3.

把数据库从错误状态恢复到某一已知的正确状态(即一致状态或完整状态),就是数据库恢复。

数据库恢复的基本技术是数据转储和登录日志文件。即根据存储在系统别处的冗余信息来恢复数据库系统。转储即 DBA 按照一定的策略将数据库复制到磁带或另一个磁盘上保存起来的过程。日志文件是用来记录事务对数据库的所有更新操作的文件,包括数据库内部的更新操作。不同数据库系统采用的日志文件格式是不同的。

当系统运行过程中发生故障,利用转储的数据库后备副本和日志文件就可以将数据库恢复到故障前的某个一致性状态。

四、求解题

1.

```
SELECT SNAME
FROM S
WHERE SNO IN
      (SELECT SNO
      FROM P,SP
      WHERE P.COLOR= 红色 AND P.PNO=SP.PNO);
```

2.

```
SELECT SNO,SNAME,STATUS
FROM S
WHERE S.CITY= 北京
```

3.

```
SELECT SUM(QTY)
FROM SP
WHERE PNO= P2
```

4.

```
UPDATE P
SET WEIGHT= WEIGHT+5,COLOR= 黄色
WHERE PNO= P2
```

五、问答题

BC 不是候选码,AB、AC 是超码。

解析:分别求出 AB_F^+、AC_F^+、BC_F^+,

$AB_F^+ = U, AC_F^+ = U, BC_F^+ = \{B,C,D\}$,可以推出 BC 不是候选码;

进一步分析,$A_F^+ = U$,即 AB 和 AC 都不是侯选码的最小集,可以得出 AB 和 AC 是超码;候选码应该为 A。

六、证明题

证明:因为关系模式的候选码由全部属性组成,所以该关系中没有非主属性。因此满足关系 R 属于 3NF 的条件:每个非主属性既不部分依赖于码,也不传递依赖于码。

又因为它没有非主属性,关系模式的候选码= U,关系模式中的决定因素也是 U,满足关系属于 BCNF 的条件。

七、综合题

1. 教师编号 Tno 不是候选码。

2. 因为:教师编号→书名(Tno -> Bname)不成立,根据候选码的定义可知 Tno 不是候选码。

3. 该关系模式的主码是:(Bno ,Tno, BorrowDate)

4. 存在部分函数依赖,如: (Tno -> Department),(Bno -> Bname)

5. 找出其中存在的所有的码,找出非主属性对码的部分依赖,将该关系模式分解为两个或两个以上的关系模式,使得分解后的关系模式中均消除了非主属性对码的部分依赖。

6. 关系模式 teacher 最高满足 1NF,因为存在非主属性对码的部分函数依赖,实例如上面第 4 小题。

7. BK(Bno,Bname) F1 = {Bno -> Bname}

TH(Tno,Tname,Tel,Department) F2 = { Tno -> Tname,Tno -> Tel,Tno -> Department}

TBB(Tno, Bno,BorrowDate, Rdate,Backup) F3 = {(Tno,Bno,BorrowDate) -

> Rdate, (Tno, Bno, BorrowDate) -> Backup }

八、综合题

(1) 有 3 个函数依赖:

(商店编号,商品编号) →部门编号,(商店编号,部门编号) →负责人,

(商店编号,商品编号) →商品库存数量

(2) R 的候选码是 (商店编号,商品编号)。

(3) 因为 R 中存在着非主属性“负责人”对候选码 (商店编号、商品编号)的传递函数依赖,所以 R 属于 2NF,R 不属于 3NF。

(4) 将 R 分解成:R_1(商店编号,商品编号,商品库存数量,部门编号)

R_2(商店编号,部门编号,负责人)

模拟试卷三

一、简答题

1. 简述 DBMS 的主要功能。

2. 对如下关系 R,指出是否存在多值依赖 C→→HR ?为什么 ?

C	T	H	R	S	G
C1	T1	H1	R1	S1	G1
C1	T1	H2	R2	S1	G1
C1	T1	H1	R1	S2	G2

3. 简述关系系统的分类。

二、求解题

有一学校教学数据库,包括学生、课程、教师、学生成绩 4 个关系。

学生关系 S(SNO,SN,AGE,SEX),有属性:学号、姓名、年龄、性别;

课程关系 C(CNO,CN,PCNO),包括属性: 课程号、课程名、先修课课程号;

教师关系 T(ENO,EN,DEPT),包括属性: 职工号、姓名、系别;

学生成绩关系 SC(SNO,CNO,ENO,G),包括属性: 学生号、课程号、任课教师职工号和学生学习成绩。

请分别用关系代数与关系演算完成下列操作:

1. 求选修所有课程并且成绩为 A 的学生名;

2. 求选修了王平老师讲授的所有课程的学生名;

3. 求不选修信息系老师开设的所有课程的学生名。

三、求解题

某学校有若干系,每个系有若干学生,若干课程,每个学生选修若干课程,每门课有若干学生选修,某一门课可以为不同的系开设,今要建立该校学生选修课程的数据库。请你设计:

1. 关于此学校数据库的 E-R 图;

2. 并把该 E-R 图转换为关系模型。

四、证明题

试证明:在关系模型中,若 R∈BCNF,则 R∈3NF

五、问答题

供应商 - 零件 - 工程项目数据库由以下四个关系模式构成:

S(SNO, SNAME,STATUS,CITY)

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

J(JNO, JNAME, CITY)

SPJ(SNO, PNO, JNO, QTY)

供应商 S,零件 P 和工程项目 J 分别由供应商号(SNO),零件号(PNO)和工程项目号(JNO)惟一标识。供货 SPJ 是指由某个供应商向某个工程项目供应某些数量的某种零件。

请用 SQL 语言完成如下的操作:

1. 找出给北京的工程项目提供不同的零件号;

2. 将没有供货的所有工程项目从 J 中删除;

3. 查询提供全部零件的供应商名;

4. 查询这样的工程项目号:供给该工程项目的零件 P1 的平均供应量大于供给工程项目 J1 的任何一种零件的最大供应量;

5. 定义一个视图,它由所有这样的工程项目(工程项目号与所在城市名称)组成:它们由供应商 S1 供货且使用零件 P1。

六、问答题

设有如下两事务:

T1:读 B;A = B + 1;写回 A

T2:读 A; B = A + 1;写回 B

1. 若这两个事务并发执行,举例可能结果。并发事务执行是否正确的标准是什么?

2. 请给出一个可串行化的调度,并给出执行结果。

七、问答题

在关系数据库中为提高查询效率,在物理实现时,对存储结构有哪些考虑?

模拟试卷三参考答案

一、简答题

1.

数据库管理系统的主要功能有:

(1) 数据库定义功能;

(2) 数据操纵功能;

(3) 数据库运行管理;

(4) 数据库的建立和维护功能。

2.

不存在多值依赖:C→→HR

多值依赖的定义为: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→→Y 成立,当且仅当对 R(U)的任一关系 r,给定一对(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。

当 C 取值 C1,(T,S,G)取值(T1,S1,G1)时,得(H,R)一组值{(H1,R1),(H2,R2)};

当 C 取值 C1,(T,S,G)取值(T1,S2,G2),得(H,R)一组值{(H1,R1)};

即与多值依赖定义矛盾,所以不存在多值依赖:C→→HR。

3.

关系系统可以分为 (最小)关系系统、关系完备的系统和全关系的系统。

最小关系系统:

一个系统可定义为最小关系系统,当且仅当它:

(1) 支持关系数据库(关系数据结构)。

从用户观点看,关系数据库由表构成,并且只有表这一种结构。

(2) 支持选择、投影和(自然)连接运算,对这些运算不必要求定义任何物理存取路径。

关系上完备的系统:

这类系统支持关系数据结构和所有的关系代数操作(或者功能上与关系代数等价的操作)。

全关系型的关系系统:

这类系统支持关系模型的所有特征。即不仅是关系上完备的而且支持数据

结构中域的概念,支持实体完整性和参照完整性。

二、求解题

1.

(1) $\pi_{SN}(S \quad (\sigma_{G=A}(SC) \div \pi_{CNO}(C))$

(2) Range C CX
SC SCX
GET W (S.SN): "CXv SCX(SCX.SNO=S.SNO∧SCX.CNO=CX.CNO∧
SCX.G= A)

2.

(1) $\pi_{SN}(S \quad (SC \div \pi_{CNO}(\sigma_{EN=王平}(SC \quad T))))$

(2) RANGE C CX
T TX
SC SCX
SC SCY
GET W (S.SN): "CX(v SCXv TX(TX.ENO=SCX.ENO ∧TX.CN= 王平 ∧SCX.CNO=CX.CNO)→v SCY(SCY.SNO=S.SNO∧SCY.CNO=CX.CNO∧SCX.ENO=SCY.ENO))

3.

(1) $\pi_{SN}(S) - \pi_{SN}(\sigma_{dept=信息}(S \quad SC \quad T))$

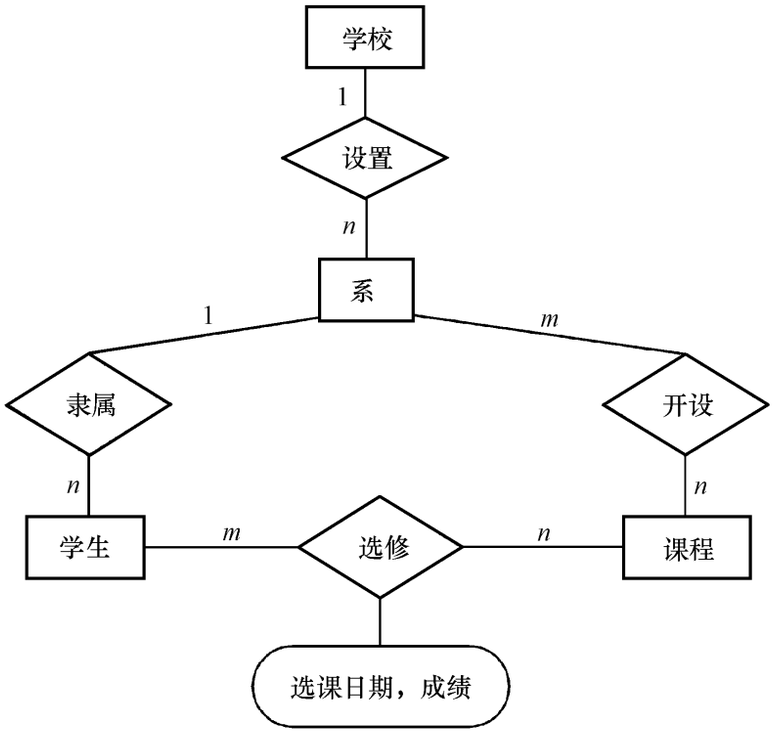
(2) RANGE T TX
SC SCX
GET W (S.SN):¬ v SCX (TX.ENO=SCX.ENO ∧TX.DEPT= 信息 ∧
SCX.SNO=S.SNO)

三、求解题

1. E - R 图中省略了各个实体的属性,图在下面

2.

(在数据库中要存放以下信息:
系:系名,系代号,系主任名,电话;
学生:学号,姓名,年龄,性别,所在系代号;
课程:课程号码、课程名称;
每个学生选修某门课的日期,成绩;
每个系开设的课程。)
学生关系:Student(Sno, Sname, Sage, Ssex, Sdept);
系关系:Dept(Dno, Dname, Dmanager, Dtelephone);
课程关系:Course(Cno, Cname);



学生选课关系:SC(Sno, Cno, Date, Grade);

系开设课程的关系:DC(Dno, Cno);

注:加横线的为码。

四、证明题

证明:3NF 定义: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 \ Y)使得 X→Y,(Y ↛X) Y→Z 成立则称 R < U, F > ∈3NF。

BCNF 定义:关系模式 R < U, F > ∈1NF。若 X→Y 且 Y \ X 时 X 必含有码,则 R < U, F > ∈BCNF。

采用反证法:

若 R∈3NF 不成立,则关系模式 R < U, F > 中存在这样的码 X,属性组 Y 及非主属性 Z(Z| Y)使得 X→Y,(Y →X 不成立) Y→Z 成立。

又,R∈BCNF,则,在 Y→Z 成立(Z \ Y)成立条件下,Y 必含有码,进而 Y→X。

与假设矛盾,所以 R∈3NF 成立。

五、问答题

1.

```
SELECT DISTINCT SPJ.PNO
FORM SPJ,J
WHERE SPJ.JNO=J.JNO AND J.CITY= 北京
```

2.

```
DELETE
```

```
FROM J
WHERE JNO NOT IN(
      SELECT JNO
      FROM SPJ);
```

3.

```
SELECT SNAME
FROM S
WHERE NOT EXISTS(
      SELECT *
      FROM P
      WHERE NOT EXISTS(
            SELECT *
            FROM SPJ
            WHERE SNO=S.SNO AND PNO=P.PNO));
```

4.

```
SELECT DISTINCT JNO
FROM SPJ
WHERE PNO= P1
GROUP BY JNO
HAVING AVG(QTY) >
      (SELECT MAX(QTY)
      FROM SPJ
      WHERE JNO= J1 );
```

5.

```
CREATE VIEW J_S1_P1
AS SELECT J.JNO,J.CITY
   FROM SPJ,J
   WHERE SPJ.JNO=J.JNO AND SPJ. SNO= S1  AND SPJ.PNO= P1
```

六、问答题

T1	T2
SLOCK B	
Y=B=2	SLOCK A
	X=A=2
UNLOCK B	
	UNLOCK A
XLOCK A	
A=Y+1	

写回 A(=3)

XLOCK B

B=X+1

写回 B(=3)

UNLOCK A

UNLOCK B

此例是不可串行化的调度。

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

T1、T2 串行执行地可能结果应该是 A=3、B=4 或 B=3、A=4,本题 T1、T2 并发执行的结果却是 A=3、B=3,所以不正确。

2

T1	T2
SLOCK B	
Y=B=2	
XLOCK A	
	Slock A
A=Y+1	等待
写回 A(=3)	等待
UNLOCK B	等待
UNLOCK A	等待
	X=A=3
	XLOCK B
	B=X+1
	写回 B(=4)
	UNLOCK A
	UNLOCK B

此例是可串行化的调度。

七、问答题

在关系数据库中为提高查询效率,要对存储结构进行优化,数据库查询物理优化的考虑包括:确定数据的存放位置和存储结构,包括确定关系、索引、聚簇、日志、备份等的存储安排和存储结构;确定系统配置等。

确定数据的存放位置:

为了提高系统性能,应该根据应用情况将数据的易变部分与稳定部分、经常存取部分和存取频率较低的部分分开存放。

确定系统配置:

DBMS产品一般都提供了一些系统配置变量、存储分配参数,供设计人员和DBA对数据库进行物理优化。初始情况下,系统都为这些变量赋予了合理的缺省值。但是这些值不一定适合每一种应用环境,在进行物理设计时,需要重新对这些变量赋值,以改善系统的性能。

模拟试卷四

一、选择题

1. 数据库与文件系统的根本区别在于（）

A. 提高了系统效率

B. 方便了用户使用

C. 数据的结构化

D. 节省了存储空间

2. 现有关系模式：

EMP(empno,ename,mgr,sal,workday)

DEPT(deptno,dname,loc)

在以下视图中，不可能更新的视图为（）。

A. 视图 V1，由 1970 年以后参加工作的雇员组成

B. 视图 V2，由部门号和各部门的平均工资组成

C. 视图 V3，由雇员姓名和其领导者姓名组成

D. 视图 V4，由薪金超出所有雇员平均薪金以上的雇员组成

3. 对由 SELECT—FROM—WHERE—GROUP—ORDER 组成的 SQL 语句，其在被 DBMS 处理时，各子句的执序次序为（）。

A. SELECT—FROM—GROUP—WHERE—ORDER

B. FROM—SELECT—WHERE—GROUP—ORDER

C. FROM—WHERE—GROUP—SELECT—ORDER

D. SELECT—FROM—WHERE—GROUP—ORDER

二、简答题

1. 试给出 BCNF 的定义，并说明满足 BCNF 的关系有哪些特性。

2. 在建立一个数据库应用系统时，为什么要首先调试运行 DBMS 的恢复功能？简述一下你所了解的数据库系统的恢复方法。

3. 试述关系数据库系统中视图(VIEW)的定义，引进 VIEW 的概念有什么优点。

4. 试述数据模型中完整性约束条件的概念，并给出关系模型中的完整性约束。

三、求解题

设有学生表 S(SNO,SN)(SNO 为学生号，SN 为姓名)和学生选修课程表 SC

(SNO,CNO,CN,G)(CNO 为课程号,CN 为课程名,G 为成绩),试用 SQL 语言完成以下各题

(1) 建立一个视图 V - SSC(SNO,SN,CNO,CN,G),并按 CNO 升序排序;

(2) 从视图 V - SSC 上查询平均成绩在 90 分以上的 SN、CN 和 G。

四、求解题

今有如下关系数据库:

S(SNO,SN,STATUS,CITY)

P(PNO,PN,COLOR,WEIGHT)

J(JNO,JN,CITY)

SPJ(SNO,PNO,JNO,QTY)

其中,S 为供应单位,P 为零件,J 为工程项目,SPJ 为工程订购零件的订单,其语义为:某供应单位供应某种零件给某个工程,请用 SQL 完成下列操作。

(1) 求为工程 J1 提供红色零件的供应商代号。

(2) 求使用 S1 供应的零件的工程名称。

(3) 求供应商与工程所在城市相同的供应商提供的零件代号。

(4) 求至少有一个和工程不在同一城市的供应商提供零件的工程代号。

五、问答题

假设存款余额 x = 1000 元,甲事务取走存款 300 元,乙事务取走存款 200 元,其执行时间如下:

甲事务	时间	乙事务
读 x	t1	
	t2	读 x
更新 x = x - 300	t3	
	t4	更新 x = x - 200

如何实现这两个事务的并发控制?

模拟试卷四参考答案

一、选择题

1. C

2. B,D

解析

因为 B 中视图 V2 的一个字段是来自集函数 AVG,所以不能更新;D 中视图 V4 含有内层嵌套,且涉及的表是导出该视图的基本表,所以也不能更新。

3. C

二、简答题

1.

关系模式 R < U, F > ∈1NF。若 X→Y 且 Y \ X 时 X 必含有码,则 R < U, F > ∈BCNF。

满足 BCNF 关系的特性有:

所有非主属性对每一个码都是完全函数依赖;

所有的主属性对每一个不包含它的码,也是完全函数依赖;

没有任何属性完全函数依赖于非码的任何一组属性。

2.

因为计算机系统中硬件的故障、软件的错误、操作员的失误以及恶意的破坏是不可避免的,这些故障轻则造成运行事务非正常中断,影响数据库中数据的正确性,重则破坏数据库,使数据库中全部或部分数据丢失,为了防止出现此类事件带来的灾难性后果,必须首先调试运行 DBMS 的恢复功能。即把数据库从错误状态恢复到某一已知的正确状态(亦称为一致状态或完整状态)的功能。

DBMS 一般都使用数据转储和登录日志文件实现数据库系统恢复功能。针对不同的故障,使用不同的恢复策略和方法。例如,对于事务故障的恢复是由 DBMS 自动完成的,对用户是透明的。

对于系统故障,也是由 DBMS 完成恢复操作,包括撤销(UNDO)故障发生时未完成的事务,重做(REDO)已完成的事务。DBA 的任务是重新启动系统,系统启动后恢复操作就由 DBMS 来完成了。

对于介质故障,则恢复方法是由 DBA 重装最新的数据库后备副本和转储结束时刻的日志文件副本,然后 DBA 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。

3.

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。

引进 VIEW 的优点有:

(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。

4.

数据模型应该反映和规定本数据模型必须遵守的基本的通用的完整性约束条件。数据模型还应该提供定义完整性约束条件的机制,以反映具体应用所涉及的数据必须遵守的特定的语义约束条件。

在关系模型中,任何关系必须满足实体完整性和参照完整性两个条件。这是关系数据模型必须遵守基本的通用的完整性约束条件。

三、求解题

(1)

```
CREATE VIEW V-SSC(SNO,SN,CNO,CN,G)
AS SELECT S.SNO,S.SN, CNO, SC. CN, SC.G
FROM S, SC
WHERE S.SNO=SC. SNO
ORDER BY CNO;
```

(2)

```
SELECT SN, CN, G
FROM V-SSC
GROUP BY SNO
HAVING AVG(G)>90;
```

四、求解题

(1)

```
SELECT DISTINCT SPJ.SNO
FROM SPJ,P
WHERE P.PNO=SPJ.PNO AND SPJ.JNO= J1  AND P.COLOR= 红 ;
```

(2)

```
SELECT J.JN
```

```
FROM J,SPJ
WHERE J.JNO=SPJ.JNO AND SPJ.SNO= S1 ;
```

(3)

```
SELECT DISTINCT SPJ.PNO
FROM S,J,SPJ
WHERE S.SNO=SPJ.SNO AND J.JNO=SPJ.JNO AND S.CITY=J.CITY;
```

(4)

```
SELECT DISTINCT SPJ.JNO
FROM S,J,SPJ
WHERE S.SNO=SPJ.SNO AND J.JNO=SPJ.JNO AND S.CITY< >J.CITY;
```

五、问答题

如果按照题中的顺序执行甲乙两个事务,则最后的 x 为 800,而不是正确的 500。为此,采用封锁的方法,将甲事务修改为:

```
WHILE(x上已有排他锁)
{
    等待
}
对 x加上排他锁
读 x
更新 x=x-300
释放排他锁
```

将乙事务修改为:

```
WHILE(x已有排他锁)
{
    等待
}
对 x加上排他锁
读 x
更新 x=x-200
释放排他锁
```

可以说明如下:

甲事务	时间	乙事务
XLOCK x	t1	
获得		
	t2	XLOCK x

		等待
更新 x = x - 300	t3	等待
x = 700		
Commit	t4	等待
UNLOCK x	t5	等待
	t6	获得 XLOCK x
	t7	更新 x = x - 200
		x = 500
	t8	Commit
	t9	UNLOCK x

附录 B[①]　数据库领域图灵奖获得者简介

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 20 世纪 60 年代中期产生到今天仅仅三十多年的历史,却已经历了三代演变,造就了 C. W. Bachman、E. F. Codd 和 James Gray 三位图灵奖得主。这里我们向大家介绍这三位图灵将得主的风采和成就。

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、发展和日趋成熟,使计算机数据处理技术跃上了一个新台阶,并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做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,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 年代诞生之初只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;高速磁带机又成功地用于计算机作存储器,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(file management)这种形式。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(data

---

① 本附录的内容主要摘录自吴鹤龄教授和崔林老师所编写的书:《ACM 图灵奖(1966—2001)——计算机发展史的缩影》(增订版)。

dependence)。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。拿 COBOL 这种常用的商用语言来说,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 FMS(File Management System)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、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,它们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 Rockwell 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,各国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 公司就与 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 年顺利登月做出了贡献。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 公司对中国人来说知之者不多,但这个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。当时微软在新墨西哥州的阿尔伯克基开张不久,鲜为人知。微软的股票是 1986 年上市

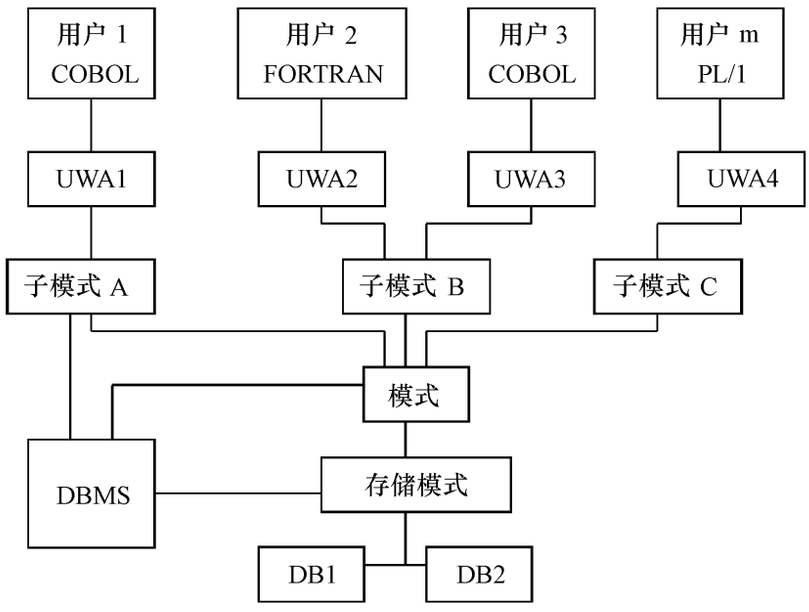
的,比 Cullinet 晚 8 年之久,但 Cullinet 最终被 CA 公司购并。1983 年巴赫曼创办了自己的公司 Bachman Information System, Inc.。

巴赫曼在数据库方面的主要贡献有两项:

第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。1971 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、1978、1981 和 1984 年的修改版本。DBTG 后来改名为 DBLTG(Data Base 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 DBA(Data Base Administrator)的概念,规定了 DBA 的作用与地位。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)。下面简要介绍一下 DBTG 的系统结构。



DBTG 的系统结构如上图所示,主要包括模式(schema)、子模式(subschema)、

物理模式(physical schema)、数据操纵(data manipulation)和数据库管理系统(DBMS,Data Base Management System)等几个部分。

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 DDL(Data Description Language)建立。子模式是某一用户对他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(user view),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(storage schema),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 DSDL(Data Storage Description Language)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 DMCL——Data Media Control Language)。

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库操纵语言 DML(Data Manipulation Language)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 UWA(User Work Area)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(consistency)、完整性(integrity)、安全性(security)和一旦出现故障情况下的恢复(recovery)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(concurrency)等,以避免出现“脏数据”(dirty data)或“丢失更新”(lose update)等异常现象。

由此可见,有关模式的数据描述语言 DDL 是建立数据库的工具,数据操纵语言 DML 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 DBMS 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,通常被称为“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,即 OSI(Open System Interconnection)。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有重要意义。巴赫曼也是建立在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承认的。

1973 年 8 月 28 日巴赫曼在亚特兰大举行的 ACM 年会上接受了图灵奖。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, 653～658 页,也可参见 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》)。

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 年以来的四分之一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”(relation)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,有坚实的数据学基础,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(Hierarchical Model)和网状模型(Network Model)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,占领了市场。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 年代初期这一十分引人注目的成就,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,年青的科德应征入伍,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以后,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,为他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 年科德重返美国 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。STRETCH 首次采用先行控制方式,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《ACM 通信》上发表的介绍 STRETCH 的多道程序操作系统的文章,是这方面的最早的学术论文之一。

尤其难能可贵的是,科德由于在工作中发觉自己缺乏硬件知识,影响了在这些重大工程中发挥更大的作用,在 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 年又获得博士学位。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的实践经验的丰富积累,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(relational algebra)和关系演算(relational calculus),定义了关系的并(union)、交(intersection)、差(difference)、投影(project)、选择(selection)、连接(join)等各种基本运算,为日后成为标准的结构化查询语言 SQL(Structured Query Language)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 第二版》),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。

科德已于 1984 年从 IBM 退休。

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 3 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

市场上有影响力的产品。

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(Integrity)、安全性(Security)、并发性(Concurrency),以及一旦出现故障后,数据库如何从故障中恢复(Recovery)。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(transaction processing)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)开始,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)。这一机制用以即保持事务之间的“并发性”,又保证数据的“完整性”。

3. 建立系统运行日志(log),记载各事务的始点、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做的备份(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做的修改。

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。格雷不喜欢微软总部所在的多雨的西雅图,愿意留在阳光灿烂的旧金山,微软特地在旧金山开辟第二个微软研究院叫“湾区研究中心”BARC(Bay Area Reserch Center),安排格雷任该研究院主管。格雷果然不负所望,领导一个研制小组开发出了 MS SQL Server 7.0,成为微软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 日于亚特兰大举行的 ACM 全国会议上接受图灵奖的。格雷发表了“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 的会上以“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做总结性回顾时认为,英国数学家巴贝奇(Charles Babage,1791—1871 年)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 年,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 3 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,美国第三任总统)决定用1 500万美元从法国政府手中买回路易斯安娜领地(Louisiana Territory)。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,而是对大多数人也有用;
4. 可测试性　以便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,以检查项目进展并鼓舞研究人员干下去。

在以上论点支持下,格雷提出的几个 IT 技术长期研究目标如下:

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,向人类专家那样快而好地做索引,做文摘;
8. 虚拟现实(格雷用了 TelePresence 这个词,参见对 1969 年图灵奖得主明斯基的介绍);
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)。

目前,格雷正在从事 Scalability 这一长期目标的研究。他是微软“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。

附录 C　SQL99

SQL99 也称为“SQL3”或者“SQL/3”,应该是 1999 年公布的 SQL 标准,但是一直推迟。我在 1999 年编写《数据库系统概论》第三版时 SQL99 还没有正式公布。《概论》第三版的第三章对 SQL 的介绍是以 SQL92 为基础的。现在我们把 SQL99 作为本书的附录给读者做一个概要的介绍,主要介绍 SQL99 的新特征。

C.1　SQL 历史

制定 SQL 标准的目的是要求不同厂商的数据库管理系统都遵从 SQL 标准,使得数据库应用系统在不同的数据库管理系统平台之间容易移植,提高数据库管理系统之间的互操作性。

SQL 标准从 1986 年公布以来随着数据库技术的发展不断发展,不断丰富。以下是 SQL 标准的进展过程。

标准	大致页数	发布日期
SQL/86		1986 年 10 月
SQL/89(FIPS 127 - 1)	120 页	1989 年
SQL/92	622 页	1992 年 7 月
SQL 调用接口 (CLI, Call Level Interface)	200 页	1995 年 9 月
SQL 永久存储模块 (PSM, Persistent Stored Module)	250 页	1996 年 11 月
SQL99		
SQL 框架	20 页	1999 年 8 月
SQL 基础部分	900 页	1999 年 8 月
SQL CLI	100 页	1999 年 11 月
SQL PSM	150 页	1999 年 8 月
SQL 宿主语言绑定(Language Bindings)	200 页	1999 年 8 月
SQL 外部数据的管理 (Management of External Data)	112 页	2000 年 7 月
SQL 对象语言绑定	238 页	2000 年 2 月

(Object Language Bindings)

SQL/4(正在进行之中)

以上全部内容

OLAP 150页 2000年7月

XA

SQL Temporal

可以发现,SQL标准的内容越来越多,SQL99合计超过1700页。SQL/86和SQL/89都是单个的文档。SQL/92和SQL99已经扩展为一系列开放的部分。

例如,SQL/92除了SQL基本部分外还增加了SQL调用接口、SQL永久存储模块。而SQL99则进一步扩展为框架、SQL基础部分、SQL调用接口、SQL永久存储模块、SQL宿主语言绑定、SQL外部数据的管理和SQL对象语言绑定等多个部分。

目前,大部分数据库管理系统产品都能支持SQL92,但是许多数据库系统只支持SQL99的部分特征,至今尚没有一个数据库系统能够完全支持SQL99。

C.2 SQL99概述

SQL99是SQL/92的超集,完全向上兼容,并且远远大于SQL/92。SQL99已经扩展为一系列开放的部分,主要有:

第一部分 SQL Framework

SQL99的框架概述。

第二部分 SQL Foundation

是SQL99的基本部分,也是最长的部分(900页),包括类型、模式、表、视图、查询和更新语句、表达式、安全模型、谓词、赋值规则、事务管理等标准的定义和说明。

第三部分 SQL/CLI(调用层接口)

主语言直接使用CLI存取数据库,因此无须预处理(即不要SQL语句预处理)。

第四部分 SQL/PSM(持久存储模块)

扩展SQL使其过程化。基本的SQL语言是高度非过程化的语言,没有类似C语言中的条件语句、循环语句。SQL/PSM增加了过程化语句功能,并且可以把它编译后存储起来供以后共享使用。

第五部分 SQL/Bindings

嵌入SQL与主语言的绑定、动态SQL的绑定。

SQL99 内容如此丰富又如此繁杂,本附录只能对 SQL99 的特征做一个十分概要的介绍。重点介绍 SQL99 比 SQL/ 92 增加的主要特征。对于 SQL99 的全面介绍,读者可以参考[4]。

SQL99 比 SQL/ 92 增加的新特征非常多。限于篇幅,我们重点把 SQL99 新特征的介绍分为“关系特征”和“面向对象的特征”两大部分。

C.3 关系特征

本节将介绍 SQL99 增加的“关系的特征”。也就是说,这部分特征是对传统的关系数据模型扩展的部分。这里介绍其中的六部分:新数据类型,新条件表达式,增强的语义,增强的安全性,完整性和决策支持。

C.3.1 新数据类型

SQL99 有四种新的数据类型。它们是:LOB、BOOLEAN、集合类型 ARRAY 和 ROW、用户定义的 DISTINCT 类型。

一、大对象 LOB(Large OBject)类型

LOB 可存储多达十亿字节的串。LOB 又分为两种:

BLOB(Binary Large OBject):二进制大对象,用于存储音频、图像数据。

CLOB(Character Large OBject):字符串大对象,用于存储长字符串数据。

LOB 类型的大小可直接在定义时指明(用 KB、MB 或 GB),如:

```
CREATE TABLE Booktable
(title VARCHAR(200),
book_id INTEGER,
summary CLOB(32K),          /*字符串大对象*/
book_text CLOB(20M),        /*字符串大对象*/
movie BLOB (2G)) ;          /*二进制大对象*/
```

LOB 类型数据是直接存储在数据库中,由 DBMS 维护,不是存在外部文件中。

LOB 类型可以像其他类型的数据一样被查询、提取、插入和修改。但是使用时要注意,必须为 LOB 提供足够大的缓冲区,而且还有很多限制,例如:

不能进行大于和小于比较,只能进行完全相等或不相等比较;

不能用于 Primary Key、Unique 和 foreign Key 中;

不能用于 GROUP BY 和 ORDER BY 子句中;

不能用于 UNION、INTERSECT 和 EXCEPT;

不能用于 Join 操作。

应用程序在操作 LOB 类型数据时用 LOB 定位器(LOB locator)来提取 LOB 数据。定位器类似游标机制,是存储在宿主变量中的值,4 字节长,是一种特殊的二进制值。应用程序通常并不传输整个 LOB 类型的值,而是通过 LOB 定位器访问大对象值。

例如,

```
EXEC SQL BEGIN DECLARE SECTION
    SQL TYPE IS BLOB_LOCATOR movie_loc;
                              /*声明 locator 变量 movie_loc */
EXEC SQL END DECLARE SECTION
EXEC SQL
    SELECT movie
    INTO :movie_loc          /* movie_loc 指向大对象 movie 的当前值 */
    FROM booktable
    WHERE title = 'Moby Dick';
```

对于 LOB 定位器:

由应用来声明 locator 变量,然后将其设置为指向某个大对象的当前值;

定位器(locator)可以在任何使用大对象数值的地方使用;

定位器还可以操作 ARRAY 类型和用户自定义类型(user-defined type)。

二、BOOLEAN 类型

第二种新数据类型是布尔类型 BOOLEAN,它支持三个真值:true、false 和 unknown。

传统的布尔操作符有 NOT、AND 和 OR。一般来说,标量表达式可以出现的位置就可以写布尔表达式。SQL99 还增加了两个新的操作符:EVERY(而不是 ALL)和 ANY。这两种操作的参数都是 BOOLEAN 类型,该 BOOLEAN 值通常是由一个表达式得到的。例如

在 WHERE 子句中的 WHERE EVERY(QTY > 200)或 WHERE ANY(QTY > 200)。

如果 QTY 列为空值,两个操作符都返回 unknown;

如果 QTY 列为非空,那么

当该列的每一个值都使(QTY > 200)为 true 时,EVERY 返回 true,否则为 false;

当该列的每一个值都使(QTY > 200)为 false 时,ANY 返回 false,否则为 true。

(即只要该列中的某一个值使(QTY > 200)为 true 时,ANY 就返回 true。)

三、集合类型:ARRAY 和 ROW

SQL99 新增集合类型,或者称为复合类型(composite type):ARRAY 和 ROW。ARRAY 类型允许在数据库的一列中存储数组。例如:

```
CREATE TABLE SALES
  (
    ITEM_NO CHAR(20),          /* 商品号*/
    QTY INTEGER ARRAY[12],     /* 整数数组,存放12个月的销售额*/
    PRIMARY KEY(ITEM_NO)
  );
```

这里的 QTY 列是 ARRAY 数组类型。一个 QTY 的值是一个包含 12 个元素的元组,其中每一个元素都是 INTEGER 类型的。SQL99 的数组只能是一维的,而且数组中的元素不能再是数组。

下面是向 SALES 表插入一个元组:

```
INSERT INTO SALES(ITEM_NO,QTY)
VALUES('T-shirt 2000',
       ARRAY[200,150,200,100,50,70,80,200,10,20,100,200]
       );
```

下面是查询语句,查找三月份销售额大于 100 的商品号:

```
SELECT ITEM_NO
FROM SALES                 /*从SALES表中选出满足下面条件的商品号*/
WHERE QTY[3]>100;          /* QTY数组第三个值大于100*/
```

ROW 类型与 ARRAY 类型相似,它允许在数据库的一列中存储结构化的值。例如:

```
CREATE TABLE employee (
   EMP_ID INTEGER,
   NAME ROW (              /* 姓名列为一个ROW类型 */
       GIVEN VARCHAR(30),
       FAMILY VARCHAR(30)),
   ADDRESS ROW (           /* 地址列为一个ROW类型 */
       STREET VARCHAR(50),
       CITY VARCHAR(30),
       PROVINCE CHAR(20))
   SALARY REAL);
```

下面是向 employee 表插入一个元组:

```
INSERT INTO employee(EMP_ID,ADDRESS);
VALUES(58556,ROW('和平大街59号','保定市','河北省'))
```

下面是查询语句,查找河北省的职工号码:

```
SELECT EMP_ID
```

```
FROM employee
WHERE ADDRESS. PROVINCE = 河北省 ;
```

四、DISTINCT 类型

SQL99 新加了一种 DISTINCT 类型。

在应用中,不同的字段经常定义成同一种数据类型。例如,职工的智商字段(IQ)和鞋号字段(SHOE_SIZE)都定义成 INTEGER 类型。表达式:

WHERE SHOE_SIZE > IQ 语法是对的,因为它们是同一种数据类型,可以相互比较。但是,这显然不符合实际应用的语义。

这时可将 SHOE_SIZE 和 IQ 字段定义成 DISTINCT 类型。

使用新的 CREATE TYPE 语句,用户可以定义一个 DISTINCT 类型,语法如下:

```
CREAT TYPE < type name >     / * 用户定义数据类型的名称 */
AS < built in scalar type name > FINAL
[ < cast option > ]              / * cast 选项 */
[ < method specification commalist > ];
```

本例中两字段类型可分别定义如下:

```
CREAT TYPE SHOE_SIZE_TYPE AS INTEGER FINAL;
                    / * 用户定义 SHOE_SIZE_TYPE 数据类型,它是整型的 */
CREAT TYPE IQ_TYPE AS INTEGER FINAL;
                    / * 用户定义 IQ_TYPE 数据类型,它是整型的 */
```

这样,我们定义了 SHOE_SIZE_TYPE 和 IQ_TYPE 两种数据类型,它们成为两种不同的数据类型。此时 SHOE_SIZE 只能与同类型数据值相比,而不能与其他任何数据类型相比,包括 INTEGER 类型。IQ 也一样。因此,表达式:WHERE SHOE_SIZE > IQ 就是非法的了。

然而,如果在定义类型时设置了选项 < cast option > ,下面用法也是合法的:

```
WHERE MY_SHOE_SIZE > CAST (MY_IQ AS SHOE_SIZE)
SET MY_IQ = MY_IQ * CAST(2 AS IQ)
```

另外,SQL99 还引入了另一种用户定义的数据类型,是结构类型。这将在后面面向对象特征部分讨论。

C.3.2 条件表达式

SQL99 引入了两个新的条件表达式:SIMILAR 和 DISTINCT。

一、SIMILAR

条件表达式 SIMILAR 用于字符串匹配,跟 LIKE 类似。即,检验一个给定的字符串是否与先前给定的模式相符合。SIMILAR 和 LIKE 的不同之处是前者支

持的范围更大。例如：

```
WHERE NAME SIMILAR TO (SQL - (86|89|92|99))|(SQL(1|2|3))
```

即变量 NAME 匹配所有的 SQL 标准(SQL - 86、SQL - 89、SQL - 92、SQL - 99，或者 SQL1、SQL2、SQL3)。

SIMILAR 可支持更多附加的特殊字符，不仅仅是 LIKE 中的“_”、“*”和“%”，还支持“+”、“-”和其他的一些字符。目的是尽可能支持在正规语言中出现的表达式的字符。应注意的是 SIMILAR 使用类似于 UNIX 的规则表达式，但并不等同于 UNIX 的规则表达式语法。

二、DISTINCT

条件表达式 DISTINCT 和 SQL/92 的 UNIQUE 相似。主要的区别是在 UNIQUE 中两个空值被认为是不相等的，因而满足 UNIQUE，返回 true。而 DISTINCT 谓词在比较两个空值时，认为他们相等，返回 false。

C.3.3 增强的语义

一、SQL99 扩展了对视图更新的支持

SQL99 允许对“UNION ALL”视图，以及一对一和一对多连接视图进行更新。对游标也做了同样的扩展。SQL99 主要依靠函数依赖(functional dependency)决定对哪些视图作更新以及如何修改底层的数据表。

使用 UNION 操作时，对于重复的行，只要有都会取消；使用 UNION ALL 时，对于重复的行，只要有都会保留。

这里举例说明怎么更新“UNION ALL”视图。

1. 创建下面的表：

```
CREATE TABLE Q1(product_no INT, sales INT, date DATE);
/* 某日 date 某个商品 product_no 的销售量为 sales */
CREATE TABLE Q2(product_no INT, sales INT, date DATE);
CREATE TABLE Q3(product_no INT, sales INT, date DATE);
CREATE TABLE Q4(product_no INT, sales INT, date DATE);
```

为了保证每个表只存放某个季度的数据，即表 Q1 存放一月份到三月份的数据，表 Q2 存放四月份到六月份的数据，依此类推，需要添加适当的约束：

```
ALTER TABLE Q1 ADD CONSTRAINT Q1_CHK_DATE
        CHECK (MONTH(date) IN (1, 2, 3));
ALTER TABLE Q2 ADD CONSTRAINT Q2_CHK_DATE
        CHECK (MONTH(date) IN (4, 5, 6));
ALTER TABLE Q3 ADD CONSTRAINT Q3_CHK_DATE
        CHECK (MONTH(date) IN (7, 8, 9));
```

```
ALTER TABLE Q4 ADD CONSTRAINT Q4_CHK_DATE
        CHECK (MONTH(date) IN (10, 11, 12));
```

2. 把"销售数据"插入表中:

```
INSERT INTO Q1 VALUES (5,6,  2001-01-02 ),
                      (8,100,  2001-02-28 );
INSERT INTO Q2 VALUES (3,10,  2001-04-11 ),
                      (5,15,  2001-05-19 );
INSERT INTO Q3 VALUES (1,12,  2001-08-27 );
INSERT INTO Q4 VALUES (3,14,  2001-12-29 ),
                      (2,21,  2001-12-12 );
```

3. 通过合并所有(UNION ALL)季度表创建全年的视图。

```
CREATE VIEW YEAR AS
        SELECT product_no, sales, date FROM Q1
UNION ALL
        SELECT product_no, sales, date FROM Q2
UNION ALL
        SELECT product_no, sales, date FROM Q3
UNION ALL
        SELECT product_no, sales, date FROM Q4;
```

我们可以通过视图查找全年的商品销售情况:

```
SELECT * FROM YEAR ORDER BY date, product_no;
PRODUCT_NO          SALES           DATE
---------   ---------   ----------
         5           6   01/02/2001
         8         100   02/28/2001
         3          10   04/11/2001
         5          15   05/19/2001
         1          12   08/27/2001
         2          21   12/12/2001
         3          14   12/29/2001
         7 record(s) selected.
```

4. 对视图 YEAR 更新和删除

因为在每个基本表上都定义了 Constraints,对视图 YEAR 的更新或删除就可以正确地转换为对某个基本表中某行的更新或删除。例如:

```
UPDATE YEAR SET sales = 20
WHERE product_no = 1 AND date =  2001-08-27 ;
```

则更新的是 Q3 中的一行。

```
DELETE FROM YEAR
WHERE product_no = 1 AND date = '2001-08-27';
```

则从 Q3 中删除同样一行:

如果要重新插入已删除的行:

```
INSERT INTO YEAR VALUES (1, 20, '2001-08-27');
```

在 SQL99 之前,是不能通过 UNION ALL 视图向基本表插入数据的。现在应用程序能通过 UNION ALL 视图插入数据:

```
INSERT INTO YEAR VALUES (1, 20, '2001-06-03'),
                        (2, 30, '2001-03-21'),
                        (2, 25, '2001-08-30');
```

二、SQL99 在事务管理中增加的新特征

SQL99 在事务管理中增加了若干新的特征:

1. 新增了 START TRANSACTION 语句,用于描述事务开始。

2. SQL99 支持 DECLARE CURSOR 中的 WITH HOLD 选项。这就使得 COMMIT 语句并不关闭游标,游标保持打开、定位状态,这样下一个 FETCH 操作将按顺序把游标指向下一个元组。

3. SQL99 支持保存点(savepoint)的概念。许多数据库管理系统都已实现了保存点。有了保存点,应用程序可以撤销在保存点以后的动作而不必撤销整个事务中的全部动作。

4. SQL99 允许使用 ROLLBACK TO SAVEPOINT 和 RELEASE SAVEPOINT 等命令。

三、SQL99 增加了递归查询的功能

在某些应用的数据中存在固有的层次联系。例如,材料单(Bill of Materials)中零件和部件之间的组成关系;一个部门各个组织之间的层次关系。对这些数据的某些查询如果不使用递归查询算法,就需要进行冗长的迭代编程。SQL99 提供了公共表表达式(Common Table Expression—WITH 子句)来实现递归。

1. 递归查询示例

下面以电视节目为例,说明递归查询的概念以及递归查询的过程,以帮助我们理解如何使用递归查询语句。图 C-1 描述了电视节目之间的层次关系。CCTV 下有新闻 NEWS、体育 SPORTS;NEWS 下面有国际新闻、经济、科技新闻等;SPORTS 下面的节目也具有类似的联系,他们形成了一个树结构。

这些数据及其联系可以用关系表(表 C-1)来表示和存储。

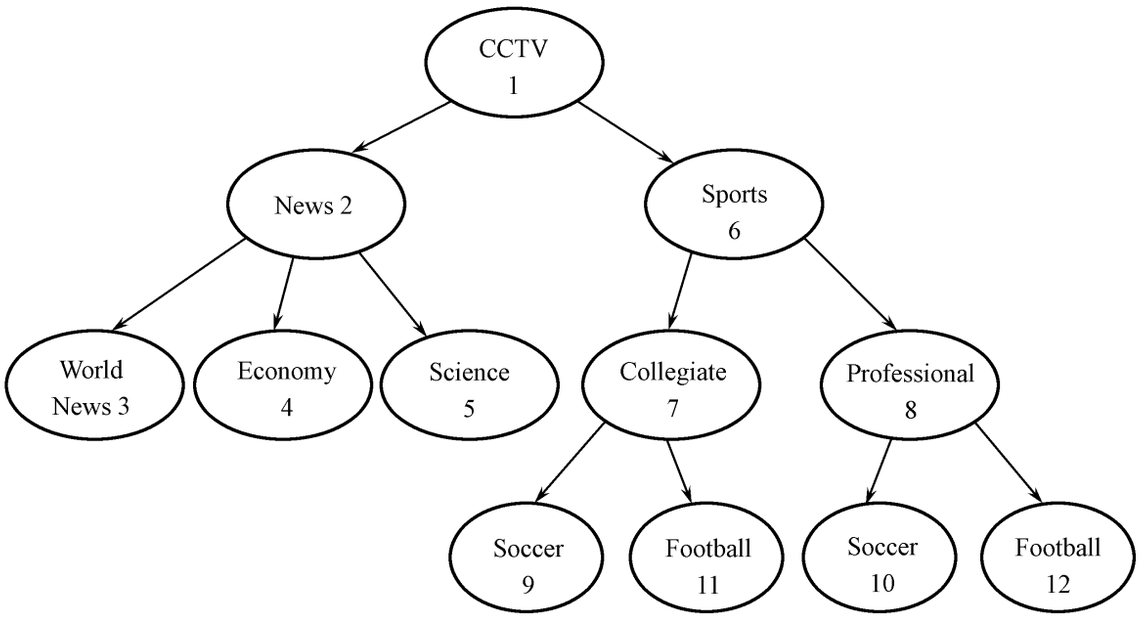


图 C-1 电视节目之间的层次关系

表 C-1 TV

Parent_id	Item_id	Item_Name
-1	1	CCTV
1	2	News
2	3	World News
2	4	Politics
2	5	Science
1	6	Sports
6	7	Collegiate
6	8	Professional
7	9	Soccer
8	10	Soccer
7	11	Football
8	12	Football

表中的 Parent_id 和 Item_id 列表示图中节点的父、子联系(根节点 CCTV 没有父节点,因此用 -1 表示)。

对于要通过遍历树来产生结果的查询,递归是非常有用的。例如:要查询给定节点‘Sports’的所有后代。表 C-2 是递归查询应该得到的结果。

表 C - 2 递归查询的结果

Parent _ id	Item _ id	Item _ Name
6	7	Collegiate
6	8	Professional
7	9	Soccer
8	10	Soccer
7	11	Football
8	12	Football

在 SQL99 中,我们可以使用下面的递归查询来得到要求的结果:

```
WITH RECURSIVE RPL(Parent_ID, Item_ID, Item_name) AS
    (SELECT ROOT.Parent_ID, ROOT.Item_ID, ROOT.Item_Name
    FROM TV ROOT
    WHERE ROOT.Parent_ID = 6
    UNION
    SELECT CHILD.Parent_ID, CHILD.Item_ID, CHILD.Item_Name
    FROM RPL PARENT, TV CHILD
    WHERE PARENT.Item_ID = CHILD.Parent_ID
    )
    SELECT DISTINCT Parent_ID, Item_ID, Item_Name
    FROM RPL
    ORDER BY Parent_ID, Item_ID, Item_Name
```

过程分析:

WITH 子句的 RPL 是一个虚拟表,有三列:Parent _ ID、Item _ ID 和 Item _ name。

WITH 子句内的第一个 SELECT 语句是初始化表。它只执行一次。它的结果形成虚拟表的初始内容以作为递归的种子。本示例中,是 从 TV 表中选出的 Parent _ ID 为 6 的若干行,如表 C - 3。

表 C - 3 虚拟表的初始内容

Parent _ id	Item _ id	Item _ Name
6	7	Collegiate
6	8	Professional

第二个 SELECT 语句可能执行多次。

将种子 RPL 作为输入传递给第二个 SELECT 语句,RPL 与 TV 执行 JOIN 操

作。将 JOIN 的结果添加(UNION ALL)到虚拟表的当前内容中,形成用于下一次传递的输入。只要虚拟表的行在增加,这个过程就会继续。

使用 UNION ALL 合并上述所有中间结果。

最后的 SELECT 允许我们在递归查询所产生的所有行中,去掉重复的结果行,并按照给定的列进行排序输出。

2. SQL99 中的递归语句

SQL99 提供了实现递归查询功能。

SQL99 把 SQL/92 表表达式的概念扩展为公共表表达式。

SQL99 允许对公共表表达式一次定义多次使用。公共表表达式在 SELECT 语句的开始部分采用 WITH 子句的形式定义,在使用公共表表达式的查询中可以多次使用它,并且公共表表达式还可以通过取别名连接到它本身,这两个特性使它在实现递归时很有用。

使用公共表表达式,避免了在每一次引用它时重新计算的代价。一般的,递归查询通常有三个部分组成:

一个公共表表达式形式的虚拟表;

一个初始化表;

一个与虚拟表进行完全内连接的辅助表。

你如果要写一个递归查询,首先写出递归查询表达式并分别起名字,然后把这些名字用到相关的查询表达式中。

C.3.4 增强的安全性

角色的概念其实早已在许多数据库产品中应用了,这次作为标准写入了 SQL99 中。即 SQL99 支持用户定义的角色。

定义角色的语句如下:

```
CREATE ROLE <rolename>
CREATE ROLE 护士              /* 建立了一个角色叫护士 */
CREATE ROLE 护士长            /* 建立了一个角色叫护士长 */
```

DBA 或用户可以把权限授给某个角色,如同授给某个用户一样。而且这些角色还可以将权限授给其他的用户或其他的角色。DBA 或用户可以把权限收回。DBA 或用户可以把角色授给某个用户或其他角色。这些语句如下:

```
GRANT/REVOKE <privileges> TO <roles>
GRANT/REVOKE <roles> TO <users and other roles>
```

例如:

```
GRANT 护士 TO 王平
        /*把护士这个角色授给用户王平,王平就拥有了护士具有的权限 */
```

```
GRANT 护士 TO 护士长
        /* 这个语句就把护士这个角色具有的权限授给了护士长 */
```

可以看出这种功能大大简化了对一组复杂权限的定义和授权机制,简化了RDBMS的安全管理。

C.3.5 增强的完整性

SQL99还增强了触发器功能。触发器的概念也已在许多数据库产品中使用多年了。当应用程序在特定表上执行特定操作时,数据库系统能按用户事先设计好的触发器,自动对特定的表执行触发器中指定的操作。例如,下面定义的触发器能够纪录对 employee 表工资字段进行修改的所有操作。

```
CREATE TRIGGER log_salupdate
                                    /* 创建一个触发器,名字为 log_salupdate */
BEFORE UPDATE OF salary             /* 在更新 employees 表的 salary 字段之前 */
ON employees
REFERENCING OLD ROW as oldrow       /* 把更新前的行称为 oldrow */
            NEW ROW as newrow       /* 把更新前的行称为 oldrow */
FOR EACH ROW                        /* 每做一次更新操作 */
INSERT INTO log_table               /* 在日志表 log_table 中插入: */
VALUES (CURRENT_USER,               /* 执行更新的用户 */
        oldrow.salary,              /* 更新前的工资 */
        newrow.salary)              /* 更新后的工资 */
```

C.3.6 决策支持

我们知道,在 SQL/92 之前的 RDBMS 主要提供的数据操纵是增、删、改和查询操作,是面向联机事务处理(OLTP)的操作。

随着数据库积累的数据越来越多,如企业中的产品数据、销售数据、客户数据及市场数据等。这些数据是宝贵的资源,其中隐含着丰富的信息和有用的知识,有可能对企业的决策产生重大影响。于是,人们提出了新的需求,希望利用数据库中的数据资源来帮助领导层进行决策。要进行决策支持,就要对大量数据进行分析,这就是面向联机分析处理(OLAP)的操作。

数据分析时用户的数据视图是多维数据模型,是面向分析的数据模型,用于给分析人员提供多种观察的视角和面向分析的操作。

多维数据模型的数据结构可以用一个多维数组来表示。一般的,多维数组用多维立方体 CUBE 来表示。多维立方体 CUBE 也称为超立方体。SQL99 中有了 CUBE 的概念。这表示 SQL99 增强了“决策支持”功能。

在多维数据模型中,数据按照多个维进行组织,每个维又具有多个层次,每个层次有多个层组成。多维数据模型使用户可以从不同的视角来观察和分析数据。常用的 OLAP 多维分析操作有切片(slice)、切块(dice)、旋转(pivot)、向上综合(rollup)、向下钻取(drill - down)等。通过这些操作,使最终用户能从多个角度多侧面观察数据、剖析数据,从而深入地了解包含在数据中的信息与内涵。

SQL99 通过增强和扩展 GROUP BY 子句的功能来支持 OLAP 分析操作。

SQL99 增加了:

GROUPING SETS、GROUPING ROLLUP 和 CUBE 等选项。

GROUPING SETS 选项允许用户精确指定进行哪些特定分组操作。

GROUP BY GROUPING SETS(A,B,C) ,可以分别按 A,B,C 分组输出结果。

GROUP BY GROUPING ROLLUP(A,B,C),则提供了向上综合(rollup)的功能。

GROUP BY GROUPING CUBE(A,B,C) 则提供了对 CUBE(A,B,C)的所有子集的分组操作。

C.4　面向对象的特征

C.3 中介绍了 SQL99 增加的“关系的特征”。SQL99 的最显著的特点是增加了对于面向对象概念(Object Oriented)的支持。这一节我们就来介绍这部分内容。

面向对象的概念早就被提出了,并且学术界和工业界都进行了大量的研究和开发。这里我们不对面向对象数据库系统做全面介绍,这部分的内容读者可以参考有关的教材和技术资料。例如我们在《数据库系统概论》的第 13 章就讲解了面向对象数据库系统的基本概念和一般特征。

SQL99 对 SQL/92 提供了面向对象的扩展,目的是为了满足很多应用程序日益迫切的需求。他们需要这些扩展来超越‘普通的”关系数据库的处理能力。

C.4.1　用户定义的结构化类型

SQL99 的一个重要特点是用户可以定义结构化的数据类型。B3.1 中我们介绍了 SQL99 新增加 DISTINCT 类型。DISTINCT 也是用户定义的数据类型。但它只限于 SQL 内置的那些数据类型,如整数、实数、字符串等,因而不是结构化的数据类型。

SQL99 允许用户通过 CREATE TYPE 语句定义自己的数据类型。

下面是一个结构类型的定义:

```
CREATE TYPE emp_type        /* 用户定义了一个 emp_type 类型 */
```

```
UNDER person_type              /* emp_type 是人员类型(person_type)的子类 */
AS( EMP_ID INTEGER,            /* 定义 emp_type 类型的结构 */
    SALARY REAL )              /* 定义 emp_type 类型的结构 */
    INSTANTIABLE               /* 该 emp_type 类型是可实例化的 */
    NOT FINAL                  /* 表示不是最后的"叶结点"类型 */
                               /* 它下面还可以定义子类型 */
    REF ( EMP_ID )
    INSTANCE METHOD            /* 定义该类型的实例可以使用的方法 */
    GIVE_RAISE
        (ABS_OR_PCT BOOLEAN,
         AMOUNT REAL)
    RETURNS REAL;
```

上例定义了一个雇员类型 emp_type。可以看到这不是一个简单的整数、字符串类型而是一个复杂结构。在这个定义中说明了雇员类型是一个人员类型(person_type)的子类。所以,雇员类型的属性有两部分,一部分是从人员类型继承来的,如姓名、住址等有关人的普通属性,另一部分是人员类型没有的而雇员才有的属性——雇员的 ID(EMP_ID)和工资(SALARY)。上例还声明了该类型是可实例化的(INSTANTIABLE),并且允许其拥有子类型(NOT FINAL,表示不是最后的"叶结点")。此外,还声明了任何关于该类型的引用 REF(REFERENCE)均通过 EMP_ID。最后,还定义了一个该类型的实例可以使用的方法 METHOD(GIVE_RAISE)。

下面再看几个用户定义的数据类型的例子:

```
CREATE TYPE POINT              /* 用户定义了一个 POINT(点)类型 */
    AS(x int,y int)            /* POINT 有两个属性 x,y */
    FINAL;                     /* POINT 不允许再定义子类型 */
CREATE TYPE LINE               /* 用户定义了一个 LINE(线)类型 */
    AS(Begin POINT,End POINT)
                               /* LINE 有两个属性 Begin 和 End,他们是 POINT 类型 */
    FINAL;
```

结构类型有许多重要特点,例如:

1) 它可以有一个或多个属性,每个属性可以是任何的 SQL 类型。这些类型包括 SQL 内置类型(如 INTEGER)、集合类型(如 ARRAY)或者结构类型本身。例如结构类型 LINE 的两个属性 Begin 和 End 就是结构类型 POINT。

2) 结构类型的所有行为通过方法(methods)、函数(functions)和过程(procedures)实现。在 SQL99 中函数和方法有很多不同。方法是和某个用户定义的类型紧密联系在一起的,而函数不是。

3) 函数和方法都可以用 SQL 语言或其他传统的程序设计语言(包括 JAVA)

来书写。

4) 结构类型的属性被封装起来,只能通过系统自动生成的运算符 observer 和 mutator 访问(可以不严格地分别称之为取值(get)和赋值(set))。然而,这些系统生成的函数不能被重载(所有其他的函数和方法都能被重载)。

5) 只能通过用户定义的函数比较它们的值。

6) 结构化类型可以参与类型继承。结构类型在类型继承过程中,子类型包括父类型的属性和方法,并可以增加自己的新的属性和方法。在 SQL99 中由根类型开始,与其相关的子类型、子类型的子类型构成一个类型层次(Type Hierarchies)。

结构类型的其他特点限于篇幅就不详细说明了。

SQL99 最重要的特征是可以处理数据类型。注意,这里用的是"处理"这个词。其含义是,SQL99 允许用户定义新的数据类型,定义新的数据类型上面的操作。

定义了的数据类型可以成为 RDBMS 所支持的系统数据类型,可以按照一定的规则供其他用户使用,用户就可以像使用整数、字符串那样使用新的数据类型。

C.4.2 函数和方法

函数和方法都可以用 SQL 语言或其他传统的程序设计语言(包括 JAVA)来书写。在 SQL99 中函数和方法有许多相同的地方,也有很多不同。

简单地说,方法是由 SQL 调用的与用户定义的类型关联的函数。它们的具体区别如下:

1) 方法是和某个用户定义的类型紧密联系在一起的,而函数不是。

2) 用户定义类型是与其相关联的方法的隐含的参数,也叫主题参数(subject parameter),而函数无此特点。

3) 函数的多态性(通过重载)是在编译时通过检查参数的个数和类型来选择最匹配的函数调用来实现的。方法的多态性通过动态联编实现,即根据主题参数在运行时刻选择最合适的方法。

4) 方法必须与和它相关联的结构化类型存储在同一计划(schema)中。而函数则无此限制。

C.4.3 参照类型(Reference Type)

SQL99 提供了一种特殊的类型:REF 类型——Reference Type(参照类型)。REF 类型也称为引用类型。一个给定的 REF 类型总是和某个特定的结构类型

相联系。下面我们通过例子来说明 REF 类型的有关概念。

SQL99 允许使用用户定义的类型来定义基表。例如

```
CREATE TYPE DEPT_TYPE                    /* 用户定义了一个类型 */
  AS (DEPT_NO CHAR(3),
      DNAME CHAR(25),
      BUDGET MONEY)
  REF IS SYSTEM GENERATED … ;
CREATE TABLE DEPT OF DEPT_TYPE    /* 使用用户定义的类型来定义基表 */
      (REF IS DEPT_ID SYSTEM GENERATED,
      PRIMARY KEY(DEPT_NO ),
        UNIQUE(DEPT_ID))… ;
```

说明:

1. 给定了结构类型 DEPT_TYPE 的定义,系统自动生成一个 REF 类型。

2. 基本表 DEPT 是用 DEPT_TYPE 类型定义的。REF 值就是 DEPT 表中各行的标识 DEPT_ID。DEPT_ID 是惟一的。

3. DEPT 表的主码是 DEPT_NO 。

一般的:

1. 在 CREATE TYPE 语句中的 REF IS SYSTEM GERERATED 表示 REF 类型的实际值是系统生成的。

REF 类型也可以有三种选项:

REF IS SYSTEM GENERATED	系统自动生成
REF USING < predefined type >	用户定义
REF (< list of attributes >)	从一组属性中导出

默认是系统自动生成的。

2. SQL99 的 CREATE TABLE 语句允许用某一结构类型来定义基表。这时,必须有一个列是 REF 类型的。其语法是:

```
REF IS < column name > SYSTEM GENERATED
```

这一列是用来标识基表各行的惟一 ID 的,概念上相当于对象标识(OID)。上面的例子中明确地指明了 DEPT_ID 列是惟一的。也可以不明确地指明,因为隐含是 UNIQUE 的。

3. 每一行的 ID 是在这行插入时赋值的,一直到该行删除时该 ID 才删除。

4. DEPT 表的主码是 DEPT_NO 。这里,DEPT 有 REF 值,在需要的情况下,我们可以将其作为主码,如下:

```
CREATE TABLE DEPT OF DEPT_TYPE
(REF IS DEPT_ID SYSTEM GENERATED,
 PRIMARY KEY(DEPT_ID),
```

```
 UNIQHE(DEPT_NO)
);
```

下面,我们进一步扩展上面的例子,再定义一个基表 EMP,如:

```
CREATE TABLE EMP
  (EMP_NO CHAR(5),
  ENAME CHAR(25),
  SALARY MONEY,
  DEPT_ID REF(DEPT_TYPE) SCOPE DEPT
        REFERENCES ARE CHECKED
        ON DELETE CASCADE,
  PRIMARY KEY(EMP_NO)
        ...
  );
```

下面给出一些解释:

一般来说,基表 EMP 将包含一个外码列 DEPT_NO ,该列参照部门表 DEPT 中的部门号。

这里没有给出外码的说明,而有一个参照列 DEPT_ID 的说明,通过“references”参照了部门。

SCOPE DEPT 指出了参照表。

REFERENCES ARE CHECKED 则表示可支持参照完整性。

ON DELETE …声明了一个删除规则(这里是 CASCADE),该删除规则跟一般的外码删除规则类似。

下面再考虑对这些表的查询。例如‘查找职工 E1 所在的部门号”:

```
SELECT EX.DEPT_ID—>DEPT_NO AS DEPT_NO
FROM EMP EX
WHERE EX.EMP_NO = E1 ;
```

注意: SELECT 子句中的表达式 EX.DEPT_ID—>DEPT_NO 返回 DEPT_ID 所标识的那一行中的 DEPT_NO 的值。

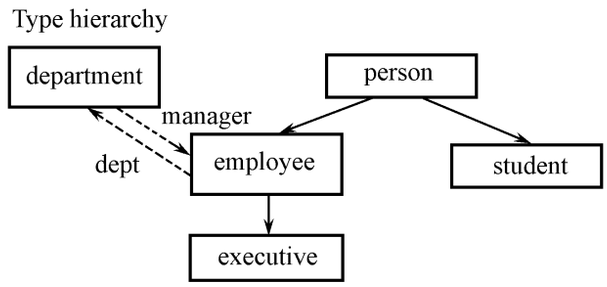
C.4.4 子表和超表

前面 C.4.1 中我们介绍了 SQL99 中结构化类型、子类型、子类型的子类型构成一个类型层次。当用户创建了结构化类型,并且在结构化类型下再创建子类型,那么这些子类型就会自动继承层次结构中在它们之上的类型(超类型)的属性。

SQL99 也支持子表和超表的概念。超表、子表、子表的子表也构成一个表层次结构。表层次和类型层次的概念十分相似。

如果一个基表是用结构类型来定义的,那么它可以有子表或/ 和超表。

例如:



C - 2 子表

```
CREATE TYPE person … NOT FINAL;
CREATE TYPE employee UNDER person …NOT FINAL;
CREATE TYPE student UNDER person …NOT FINAL;
CREATE TYPE executive UNDER employee …NOT FINAL;
```

类型 employee、student 是类型 person 的子类型,类型 executive 是 employee 的子类型。他们构成了一个类型层次,如图 C - 2 所示。

用这些结构类型可以定义以下的基本表:

```
CREATE TABLE people OF person (…)
CREATE TABLE emp OF employee UNDER people;
CREATE TABLE student OF students UNDER people;
CREATE TABLE manager OF executive UNDER emp;
```

这些表就构成了一个表层次。people 是 emp 表和 student 表的超表,emp 是 people 的子表,student 也是 people 的子表,manager 是 emp 的子表。

子表可以继承父表的列、约束条件、触发器等等,子表可以定义自己的新属性。

对某个表的查询其实是对该表和它所有子表中对象集合的查询。

例如:

```
SELECT NAME,ADDRESS
FROM PEOPLE
WHERE AGE >= 35 ;
```

这个查询是要找出 PEOPLE 表上年龄大于等于 35 岁的人,查询结果包括了 MANAGER、STUDENT、EMP 表上的所有人。

可以使用 ONLY“关闭”对子表的检索。办法是在 FROM 子句中使用 ONLY 将检索的对象限制为指定表中的对象,而不是该表和它的子表中的对象。

```
SELECT NAME,ADDRESS
FROM ONLY PEOPLE
```

```
WHERE AGE >= 35 ;
```

这个查询是只找出了 PEOPLE 表上年龄大于等于 35 岁的人,不包括 MANAGER、STUDENT、EMP 表上的人。

对于 INSERT、DELETE、UPDATE 等其他操作,也有同样的规则。

INSERT:向子表插入一行时一般情况下会在该子表的超表上也插入一行。例如正常的向 STUDENT 中插入一个新学生,则在 PEOPLE 表上也插入一个新行。

DELETE:从表删除一行时一般情况下会在该表的超表和子表上删除一行。

详细讨论请参考[3][4]。

C.5　主要 DBMS 产品对 SQL99 的新特征的支持

表 C-4 列出 ORACLE、DB2 和 SQL SERVER 三个数据库产品对 SQL99 新特征的支持。ORACLE 使用 oracle9i,DB2 使用 db2 7.2 版本,SQL SERVER 使用 SQL Server2000。

表 C-4

	SQL99 新特征		ORACLE	DB2	SQL SERVER
非面向对象的特征	新数据类型	LOB	S	S	SWV
		BOOLEAN	S	NS	NS
		集合类型	NS	NS	NS
		DISTINCT	NS	SWV	NS
	新条件表达式	SIMILAR 和 DISTINCT	NS	NS	NS
	增强的语义	递归查询	NS	S	NS
		事务管理	SWL	SWL	SWL
	增强的安全性和完整性	角色授权	SWV	SWV	SWL
		触发器	SWV	SWV	SWV
	决策支持(group by 中的选项)	SETS	NS	NS	NS
		ROLLUP	NS	SWV	SWV
		CUBE	NS	SWV	SWV
面向对象特征	用户定义的结构类型		S	S	NS
	结构化类型层次		S	S	NS
	类型表		S	S	NS
	REF 类型		S	S	NS

说明:

1. 数据库对 SQL99 新特征的支持级别如下:

完全支持(Supported,缩写为 S);

支持,但在语法和用法上略有变化(Supported, with variations,缩写为 SWV);

部分支持(Supported, with limitations,缩写为 SWL);

不支持(Not Supported,缩写为 NS)。

2. 表中显示不支持 SQL99 标准的某一特征,数据库厂商在部分功能上也许会有其他的语法或方法来实现类似的特征。

附录 D　数据库基准测试 TPC - C

数据库基准(Benchmark)测试是针对数据库管理系统(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。

D.1　数据库基准的发展历史

在近 20 年中,涌现出不少数据库测试基准。下面介绍曾经使用过的比较著名的几个数据库基准。

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 基准包括以下要点:

(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^3AP 基准

Dina Bitton, Cyril Orji 和 Carolyn Turbyfill 后来提出了 AS^3AP 基准,这是一个更加完整的关系数据库基准测试。

AS^3AP 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的关系查询,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,针对查询,特别是集合查询;

(2) 增加了集合查询,一共执行 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。他们拥有各自的基准,可以为开发者和客户提供很有价值的性能测量建议。

D.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 ,简记为 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 。

D.2.1 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 - H, TPC - R, TPC - W 等。下面简要介绍这些测试基准:

TPC - A 这是 TPC 委员会于 1989 年 11 月公布的第一个测试标准。

TPC - B 是面向成批事务处理的测试程序。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 都为科学的测试 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的特点,也不能全面测试 TP 系统性能。

TPC - C 是最成功的目前广泛使用的数据库基准测试。TPC - A, TPC - B 已经失去了应用价值而被废弃了。后面几节我们将详细介绍 TPC - C。

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。

后来 TPC - D 分成了两个基准测试 TPC - H 和 TPC - R。TPC - D 也就被废弃了。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。

TPC - W 最近又开发的针对 Web E-Commerce 应用的基准。

为了使基准更加公正有效,TPC 仍然在不断地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 已经被废弃。目前最重要和最常用的是 TPC - C、TPC - H、TPC - R 和 TPC - W。

D.2.2 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》[13]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:既可以用于小的计算机系统,也可以应用于大的计算机系统。

简单性:易于理解。

TPC - C 是针对联机事务处理(On_Line 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。由于他们针对的应用环境不同,所以它们之间存在许多差异。

下面我们详细介绍 TPC - C 测试基准。

D.3 TPC - C

TPC 委员会设计的基准中最著名的就是 TPC - C。TPC - C 已经成为 OLTP 性能测试的工业标准。

D.3.1 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,包括批发商,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、输入数据的键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 的特点是: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,消耗大量的人力资源。

D.3.2 TPC - C 商务模型

TPC - C 的商务模型如图 D - 1 所示,描述了供货商(也就是批发商)公司、公司所管理的仓库、销售地区以及顾客的分层结构。一个供货商经营一定数目的仓库(Warehouse),每个仓库为 10 个地区(District)供货,每个地区有 3 000 名顾客(Customer)。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。

该商务模型所模拟的应用是可扩展的。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。

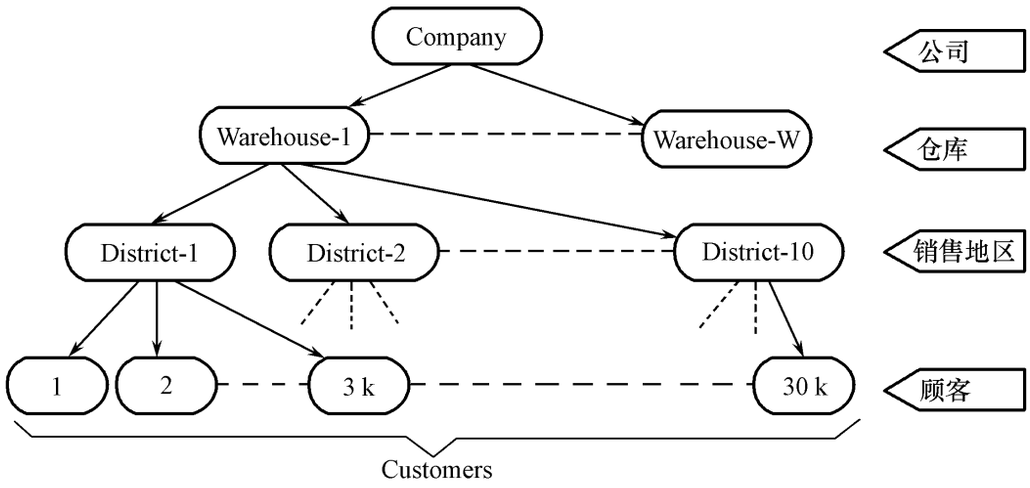


图 D - 1 TPC - C 商务模型

D.3.3 TPC - C 的逻辑数据库模式

TPC - C 的逻辑数据库模式如图 D - 2 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。其他表的基数是它的基数的函数。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。

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 D - 2 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)。

C.3.4 TPC - C 五种事务

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)

建立新订单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。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)

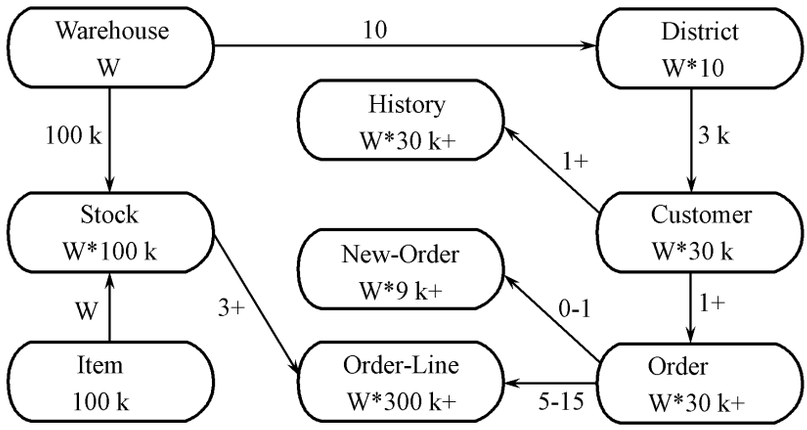


图 D - 2 TPC - C 逻辑数据库模式

支付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;这个事务执行频率高,而且响应时间要求很严格。

3. 查询订单状态事务(Order status)

查询订单状态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,响应时间的要求也不是很严格。

4. 交付事务(Delivery)

交付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)

查询库存水平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,响应时间要求和一致性要求不是很高。

注意,其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。

D.3.5 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:

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 测试时五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。

$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。

D.3.6 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 D - 3 描述了 TPC - C 事务的流程。

事务的流程通常是:

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显示事务输出结果信息之间的时间间隔。

4. 用户阅读处理结果,返回第一步执行下一个事务。

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。

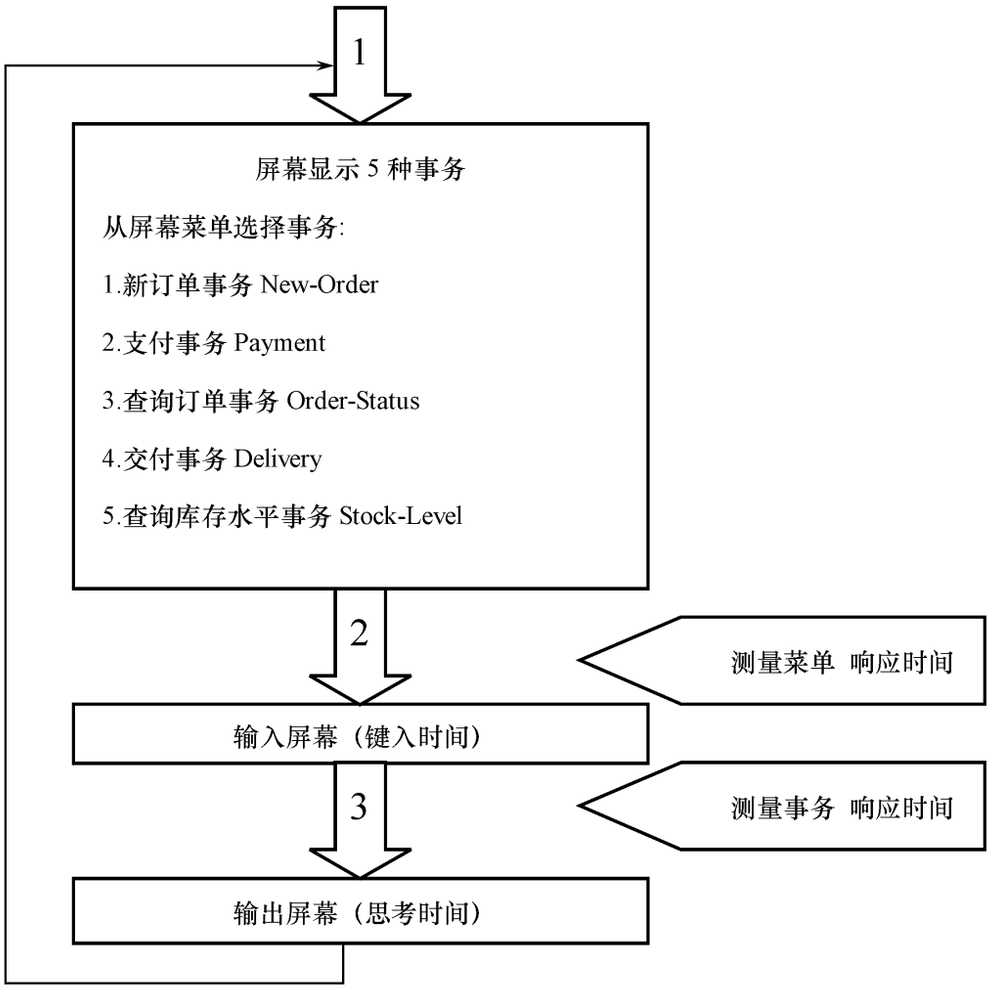


图 D - 3 TPC - C 事务流程图

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。

D.3.7 ACID 测试

ACID 特性指的是数据库事务必须具有的四个特性:原子性(Atomicity)、一致性(Consistency)、隔离性(Isolation)和持续性(Durability)。这个四个特性简称为 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

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。

随机选择一个仓库、地区和顾客,运行 payment 事务,rollback 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 针对四类数据不一致性,定义了 4 种隔离级别,如表 D - 1。

这四类数据不一致性是:

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。

隔离级别 L0:可以保证不丢失修改;

隔离级别 L1: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:在隔离级别 L1 的基础上还可以保证可重复读;

隔离级别 L3:在隔离级别 L2 的基础上还可以保证无幻影。

表 D - 1 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, Delivery 事务的组合,进行多个用户并发执行、提交和回滚操作。测试当资源竞争时,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 T_1;

(2) T_1 在接近 commit 前停止;

(3) 对于同一个顾客开始一个新的 payment 事务 T_2;

(4) 证明事务 T_2 等待;

(5) 回滚 T_1,完成 T_2;

(6) 检查数据库表中的有关字段(如帐户余额等)只被事务 T_2 修改。

TPC - C 在测试说明书中声明了: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,就要设计其他的隔离性测试方法,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 3 种情况下的永久性。

(1) 存储介质的失效;

(2) 系统运行中的崩溃(system crash),需要系统重起;

(3) 内存数据的部分丢失。

TPC - C 的测试过程:

(1) 运行 TPC - C 事务;

(2) 人为设置以上 3 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)。

参考文献和参考站点

1 Mattos N M. SQL3——新的SQL标准,新一代对象关系数据库. IBM 数据库通用技术公司, 1999

2 Eisenberg A, Melton J. SQL:1999, formerly known as SQL3. 1999

3 Mattos N M. SQL99, SQJ/MM, and SQLJ: An Overview of the SQL Standards. IBM Database Common Technology, 2000

4 Gulutzan P, Pelzer T. SQL-99 Complete, Really. Miller Freeman, Inc, 1999
该书全面描述SQL3标准,通过许多例子来说明SQL3的功能。对于SQL3来说,内容比较完整。该书的翻译本为:SQL-3参考大全.齐舒创作室翻译.北京:机械工业出版社, 2000

5 DATE C J. An Introduction to Database Systems, 7th Edition. Addison-Wesley Pub co., 1995

6 Kline K, Kline D. SQL in a Nutshell. O Reilly & Associates, 2000

7 Melton J. SQL:1999 A Tutorial. Oracle Corp., 1999

8 http://otn.oracle.com/products/oracle9i/content.html

9 http://www-3.ibm.com/software/data/db2/udb/features.html

10 http://msdn.microsoft.com/library

11 萨师煊,王珊.数据库系统概论.第三版.北京:高等教育出版社,2000

12 王珊等编著.数据仓库技术与联机分析处理.北京:科学出版社,1998

13 Gray J. Database and Transaction Processing Performance Handbook. Morgan Kaufmann, 1993

14 Jim Gray's Benchmark Handbook Online. http://archives.postgresql.org/pgsql-hackers/1998-07/msg00342.php

15 Transaction Processing Performance Council(TPC). TPC BENCHMARK™ C Standard Specification Revision 5.0. 2001

16 Transaction Processing Performance Council(TPC). TPC BENCHMARK™ H (Decision Support) Standard Specification Revision 1.5.0. 2002

17 Transaction Processing Performance Council(TPC). TPC BENCHMARK™ R (Decision Support) Standard Specification Revision 2.0.0.2002

18 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论文,2001

19 TPC委员会网站. http://www.tpc.org, info@tpc.org

20 SPEC. http://www.spec.org

21 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

22 吴鹤龄、崔林:《ACM图灵奖(1966-2001)——计算机发展史的缩影》(增订版).