数据库原理

教 案

编写：数据库课程组

**目录**

[第一章 绪论 1](#_Toc504028869)

[1.1 授课内容 1](#_Toc504028870)

[1.2 教学目标要求 1](#_Toc504028871)

[1.2.1 能力目标 1](#_Toc504028872)

[1.2.2 知识目标 1](#_Toc504028873)

[1.3 课件分析 1](#_Toc504028874)

[1.4 教学方法 1](#_Toc504028875)

[1.5 教学过程 2](#_Toc504028876)

[1.5.1 课前补充 2](#_Toc504028877)

[1.5.2 上节回顾 2](#_Toc504028878)

[1.5.3 作业点评 2](#_Toc504028879)

[1.5.4 导入新课 2](#_Toc504028880)

[1.5.5 讲授新课 2](#_Toc504028881)

[1.5.6 课时小结 11](#_Toc504028882)

[1.5.7 课外补充 11](#_Toc504028883)

[第二章 关系数据库 12](#_Toc504028884)

[2.1 授课内容 12](#_Toc504028885)

[2.2 教学目标要求 12](#_Toc504028886)

[2.2.1 能力目标 12](#_Toc504028887)

[2.2.2 知识目标 12](#_Toc504028888)

[2.3 课件分析 12](#_Toc504028889)

[2.4 教学方法 13](#_Toc504028890)

[2.5 教学过程 13](#_Toc504028891)

[2.5.1 课前补充 13](#_Toc504028892)

[2.5.2 上节回顾 13](#_Toc504028893)

[2.5.3 作业点评 13](#_Toc504028894)

[2.5.4 导入新课 13](#_Toc504028895)

[2.5.5 讲授新课 13](#_Toc504028896)

[2.5.6 课时小结 20](#_Toc504028897)

[2.5.7 课外补充 20](#_Toc504028898)

[第三章 关系数据库标准语言SQL 21](#_Toc504028899)

[3.1 授课内容 21](#_Toc504028900)

[3.2 教学目标要求 21](#_Toc504028901)

[3.2.1 能力目标 21](#_Toc504028902)

[3.2.2 知识目标 21](#_Toc504028903)

[3.3 课件分析 21](#_Toc504028904)

[3.4 教学方法 22](#_Toc504028905)

[3.5 教学过程 22](#_Toc504028906)

[3.5.1 课前补充 22](#_Toc504028907)

[3.5.2 上节回顾 22](#_Toc504028908)

[3.5.3 作业点评 22](#_Toc504028909)

[3.5.4 导入新课 22](#_Toc504028910)

[3.5.5 讲授新课 23](#_Toc504028911)

[3.5.6 课外补充 56](#_Toc504028913)

[第四章 数据库安全性 57](#_Toc504028914)

[4.1 授课内容 57](#_Toc504028915)

[4.2 教学目标要求 57](#_Toc504028916)

[4.2.1 能力目标 57](#_Toc504028917)

[4.2.2 知识目标 57](#_Toc504028918)

[4.3 课件分析 57](#_Toc504028919)

[4.4 教学方法 58](#_Toc504028920)

[4.5 教学过程 58](#_Toc504028921)

[4.5.1 课前补充 58](#_Toc504028922)

[4.5.2 上节回顾 58](#_Toc504028923)

[4.5.3 作业点评 58](#_Toc504028924)

[4.5.4 导入新课 58](#_Toc504028925)

[4.5.5 讲授新课 58](#_Toc504028926)

[4.5.6 课时小结 66](#_Toc504028927)

[4.5.7 课外补充 67](#_Toc504028928)

[第五章 数据库完整性 68](#_Toc504028929)

[5.1 授课内容 68](#_Toc504028930)

[5.2 教学目标要求 68](#_Toc504028931)

[5.2.1 能力目标 68](#_Toc504028932)

[5.2.2 知识目标 68](#_Toc504028933)

[5.3 课件分析 68](#_Toc504028934)

[5.4 教学方法 69](#_Toc504028935)

[5.5 教学过程 69](#_Toc504028936)

[5.5.1 课前补充 69](#_Toc504028937)

[5.5.2 上节回顾 69](#_Toc504028938)

[5.5.3 作业点评 69](#_Toc504028939)

[5.5.4 导入新课 69](#_Toc504028940)

[5.5.5 讲授新课 69](#_Toc504028941)

[5.5.6 课时小结 72](#_Toc504028942)

[5.5.7 课外补充 72](#_Toc504028943)

[第六章 关系数据理论 73](#_Toc504028944)

[6.1 授课内容 73](#_Toc504028945)

[6.2 教学目标要求 73](#_Toc504028946)

[6.2.1 能力目标 73](#_Toc504028947)

[6.2.2 知识目标 73](#_Toc504028948)

[6.3 课件分析 73](#_Toc504028949)

[6.4 教学方法 74](#_Toc504028950)

[6.5 教学过程 74](#_Toc504028951)

[6.5.1 课前补充 74](#_Toc504028952)

[6.5.2 上节回顾 74](#_Toc504028953)

[6.5.3 作业点评 74](#_Toc504028954)

[6.5.4 导入新课 74](#_Toc504028955)

[6.5.5 讲授新课 74](#_Toc504028956)

[6.5.6 课时小结 80](#_Toc504028957)

[6.5.7 课外补充 80](#_Toc504028958)

[第七章 数据库设计 81](#_Toc504028959)

[7.1 授课内容 81](#_Toc504028960)

[7.2 教学目标要求 81](#_Toc504028961)

[7.2.1 能力目标 81](#_Toc504028962)

[7.2.2 知识目标 81](#_Toc504028963)

[7.3 课件分析 81](#_Toc504028964)

[7.4 教学方法 82](#_Toc504028965)

[7.5 教学过程 82](#_Toc504028966)

[7.5.1 课前补充 82](#_Toc504028967)

[7.5.2 上节回顾 82](#_Toc504028968)

[7.5.3 作业点评 82](#_Toc504028969)

[7.5.4 导入新课 82](#_Toc504028970)

[7.5.5 讲授新课 82](#_Toc504028971)

[7.5.6 课时小结 96](#_Toc504028972)

[7.5.7 课外补充 96](#_Toc504028973)

[第八章 数据库编程 97](#_Toc504028974)

[8.1 授课内容 97](#_Toc504028975)

[8.2 教学目标要求 97](#_Toc504028976)

[8.2.1 能力目标 97](#_Toc504028977)

[8.2.2 知识目标 97](#_Toc504028978)

[8.3 课件分析 97](#_Toc504028979)

[8.4 教学方法 97](#_Toc504028980)

[8.5 教学过程 97](#_Toc504028981)

[8.5.1 课前补充 97](#_Toc504028982)

[8.5.2 上节回顾 98](#_Toc504028983)

[8.5.3 作业点评 98](#_Toc504028984)

[8.5.4 导入新课 98](#_Toc504028985)

[8.5.5 讲授新课 98](#_Toc504028986)

[8.5.6 课时小结 109](#_Toc504028987)

[8.5.7 课外补充 109](#_Toc504028988)

[第十章 数据库恢复技术 110](#_Toc504028989)

[10.1授课内容 110](#_Toc504028990)

[10.2 教学目标要求 110](#_Toc504028991)

[10.2.1 能力目标 110](#_Toc504028992)

[10.2.2 知识目标 110](#_Toc504028993)

[10.3 课件分析 110](#_Toc504028994)

[10.4 教学方法 110](#_Toc504028995)

[10.5 教学过程 111](#_Toc504028996)

[10.5.1 课前补充 111](#_Toc504028997)

[10.5.2 上节回顾 111](#_Toc504028998)

[10.5.3 作业点评 111](#_Toc504028999)

[10.5.4 导入新课 111](#_Toc504029000)

[10.5.5 讲授新课 111](#_Toc504029001)

[10.5.6 课时小结 116](#_Toc504029002)

[10.5.7 课外补充 117](#_Toc504029003)

[第十一章 数据库并发控制 118](#_Toc504029004)

[11.1 授课内容 118](#_Toc504029005)

[11.2 教学目标要求 118](#_Toc504029006)

[11.2.1 能力目标 118](#_Toc504029007)

[11.2.2 知识目标 118](#_Toc504029008)

[11.3 课件分析 118](#_Toc504029009)

[11.4 教学方法 118](#_Toc504029010)

[11.5 教学过程 119](#_Toc504029011)

[11.5.1 课前补充 119](#_Toc504029012)

[11.5.2 上节回顾 119](#_Toc504029013)

[11.5.3 作业点评 119](#_Toc504029014)

[11.5.4 导入新课 119](#_Toc504029015)

[11.5.5 讲授新课 119](#_Toc504029016)

[11.5.6 课时小结 135](#_Toc504029017)

[11.5.7 课外补充 135](#_Toc504029018)

第一章 绪论

## 1.1 授课内容

授课科目：数据库原理

授课内容：绪论

授课类型：讲授

授课时间：4学时

主讲教师：

## 1.2 教学目标要求

### 1.2.1 能力目标

1. 对数据库系统有一个初步的认识；
2. 理解数据库的产生和发展过程；
3. 掌握数据库系统中的一些基本概念。

### 1.2.2 知识目标

1. 掌握数据库的4个基本概念；
2. 掌握概念模型、逻辑和物理模型的定义；
3. 掌握数据模型的组成要素；
4. 掌握数据库系统的三级模式、两级映像。

## 1.3 课件分析

概 述： 本章主要介绍数据库系统概述，通过讲授数据库的4个基本概念（数据、数据库、数据库管理系统、数据库系统）来认识数据库技术的基本内容；通过数据管理技术的发展历史，达到理解数据库系统的特点；通过介绍两类数据模型（概念模型、逻辑物理模型）引出数据管理的概念，接着介绍数据模型的三要素（数据结构、数据的操作、数据的完整性约束条件），并依据此三要素讲解三类常见的数据模型，最主要的是讲解什么是关系模型；依据数据库的独立性特点，介绍数据库系统的三级模式结构和两级映像功能；最后，介绍数据库系统的组成。

教学重点：4个基本概念、数据模型的概念以及组成要素、三级模式两级映像

教学难点： 数据库系统的结构

## 1.4 教学方法

讲授法。

## 1.5 教学过程

### 1.5.1 课前补充

（0分钟）

### 1.5.2 上节回顾

（0分钟）

### 1.5.3 作业点评

（0分钟）

### 1.5.4 导入新课

（3分钟）

本节课是本门课程的导论性课程，首先要让学生明确为什么要开设本门课程，是因为其重要性以及应用的广泛性在日常生活中的体现，包括同学们常用的学分制系统、医院的挂号系统、银行的各种业务系统、火车票的查询和订票系统等。其次是学生学会这门课程之后可以做什么，比如能够作为DBA管理单位的数据库、可以开发动态网站，还可以进行其他与数据库有关的应用。

### 1.5.5 讲授新课

1.1 引言

1.1.1 数据、数据库、数据库管理系统、数据库系统

1． 数据（data）

高级语言的数据,如C语言中各种类型数据(常量、变量)：int，long，char，float，struct,FILE，……（着重文件类型数据说明）

定义：1）数据是描述事物的符号记录，2）数据与其语义是不可分的，需要经过语义解释。

2． 数据库（database，简称DB）

定义：数据库是长期存储在计算机内、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、描述和存储，具有较小的冗余度，较高的数据独立性和易扩展性，并为各种用户共享。（P1）

目前流行数据库的数据模型：关系模型，数据结构为表。

数据库的存储：若干表一个库，一张表一个库（文件系统为一个文件一个库）

对数据库的操作：在数据库管理系统的支持下，可进行数据查询（select）、数据定义(create drop alter)、数据操纵(insert update delete)、数据控制(grant revoke)

3) 从定义上数据库与文件比较：

对照数据库说明：长期存储在计算机内、有组织的（组织简单）、可共享（共享性差）的数据集合。冗余度大，数据独立性低。

3． 数据库管理系统（database management system ,简称DBMS）（P7）

定义（功能解释）：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。

对照文件系统：提供的操作简单，主要包括：建立文件、删除文件、读元素、写元素、打开文件、关闭文件等。

用户与文件系统及数据库的接口：文件系统：系统调用

数据库管理系统：SQL（数据查询、数据定义、数据操纵、数据控制）。

4． 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（及其应用开发工具）、应用系统、数据库管理员和用户构成。

5． 本课程的主要研究内容、意义及学习方法

主要内容 关系数据库及SQL、关系数据库设计（设计理论、具体设计）、数据保护。

意义：非常实用

学习方法：大量的上机实践是学好本课程的先决条件**。**

**1.1.2 数据管理的技术的发展**

数据管理：是指如何对数据进行分类、组织、编码、存储、检索和维护，是数据处理的中心问题。

随着计算机硬件和软件的发展，数据管理技术经历了三个发展阶段：人工管理、文件系统、数据库管理系统。

1． 人工管理阶段的特点：

背景：硬件，没有直接存取的存储设备；软件，没有操作系统及管理数据的软件；数据处理方式为批处理）

（1）数据不保存

（2）数据需要由应用程序自己管理，没有相应的软件负责数据的管理工作。应用程序中不仅要规定数据的逻辑结构，而且要设计物理结构（包括存储结构、存取方法、输入方式等）。（PASCAL语言文件处理：应用程序中只考虑逻辑结构）

（3）数据不共享：数据面向应用，一组数据只能对应一个程序。因此程序与程序之间有大量的冗余数据。

（4）数据不具有独立性：数据的逻辑结构或物理结构发生变化后，必须对应用程序做相应的修改。

2． 文件系统阶段的特点：

背景：硬件，已有直接存取的存储设备；软件，操作系统中已经有了专门的数据管理软件，一般成为文件系统；数据处理方式，不仅有了批处理而且能够联机实时处理）

（1）数据可以长期保存。

（2）由专门的软件即文件系统进行数据管理。应用程序按名存取数据，大大节省了程序的编写与维护工作量。

（3）数据共享性差。一个文件基本上对应于一个应用程序。不同应用程序具有部分相同数据时也必许建立各自的文件，不能共享相同的数据，数据冗余度大。相同数据重复存储、各自管理，给数据的修改与维护带来了困难，容易造成数据的不一致性。

（4）数据的独立性低。数据与程序间缺乏独立性：一旦数据的逻辑结构改变，必须修改应用程序修改文件结构定义。而应用程序的改变（应用程序改用不同的高级语言）也将引起文件数据结构的改变。

3． 数据库系统阶段

产生背景：以文件系统作为数据管理手段已经不能满足应用的需要（多用户、多应用共享数据），于是出现了数据库技术及统一管理数据的专门软件数据库管理系统。特点：

（1）数据结构化，存取方式灵活

数据库系统实现了整体数据的结构化：在描述数据时不仅要描述数据本身，还要描述数据之间的联系。文件系统尽管其记录内部有某种结构，但记录之间没有联系。

（2）数据共享性好，冗余度低

数据库系统从整体角度看待和描述数据，数据不再面向某个应用而是面向整个系统。这样既可以大大减少数据冗余，节约存储空间，又能避免数据之间不相容与不一致性。

（3）数据独立性高

数据库系统提供了两方面的映象功能，从而使数据既具有物理独立性又具有逻辑独立性。

数据与程序间的逻辑独立性：数据的总体逻辑结构与某类应用所涉及的局部逻辑结构之间的映象或转换功能，保证了当数据的总体逻辑结构改变时，通过对映象的相应改变可以保持数据的局部逻辑结构不变，由于应用程序是依据数据的局部逻辑结构编写的，所以应用程序不必修改。

数据与程序间的物理独立性：数据的存储结构与逻辑结构之间的映象或转换功能，保证了当数据的存储结构（或物理结构）改变时，通过对映象的相应修改可以保持数据的逻辑结构不变，从而应用程序也不必修改。

（4）数据由DBMS统一管理和控制

统一管理有结构的数据，使数据可以有灵活的使用方式。数据库系统弹性大，易于扩充。

有以下几方面的控制功能：

（1）数据的安全性：指保护数据，防止不合法使用数据造成数据的泄密和破坏，使每个用户只能按规定对某些数据以某种方式进行访问和处理。

（2）数据的完整性：指数据的正确性、有效性和相容性。即将数据控制在有效的范围内，或要求数据间满足一定的关系。

（3）并发控制：当多个用户的并发进程同时存取、修改数据时，可能会相互干扰而得到错误的结构，并使数据的完整性遭到破坏，因此对多用户并发操作加以控制和协调。

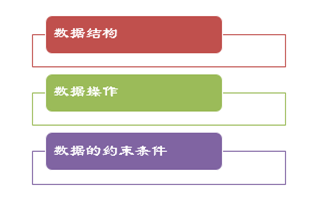
（4）数据库恢复：硬件、软件、操作员失误、故意破坏等造成的故障，DBMS必须将数据库从错误状态恢复到某一已知的正确状态。

**1.2 数据模型**

**1.2.1数据模型的作用、组成要素、概念**

1．数据模型的作用：通俗地说是计算机世界对现实世界信息处理的模拟。具体为抽象、表示和处理现实世界的数据和信息。

2．数据模型的组成要素：



（1）数据结构：

现实世界的信息抽象、表示为计算机世界的数据结构。

数据结构：为所研究对象类型的集合。一类对象与数据类型、内容、性质有关，另一类对象与数据间的联系有关。

例 现实世界的“学生选修课程处理”涉及的信息，抽象、表示为计算机世界的数据结构涉及的对象有：

与数据本身有关对象：学生、课程

与数据间联系有关对象：选课

数据结构（关系数据模型为表，数据及其联系都用表表示）

学生（学号、姓名、性别、系别、年龄、籍贯）（加属性类型说明）

课程（课程号、课程名、学分）（加属性类型说明）

选课（学号、课程号、成绩）（加属性类型说明）

数据结构在数据模型中的地位

数据模型中的数据结构是刻画一个数据模型性质的最重要方面，人们通常按数据结构类型来命名数据模型。

数据库中的数据结构有三种：关系结构、层次结构、网状结构。关系模型的重要性。

（2）数据操作

现实世界的信息对应计算机世界的数据结构，现实世界信息的处理对应计算机世界的数据操作。主要有检索和更新（包括插入、删除、修改）两大操作。

数据操作：是指对数据库中各种对象（型）的实例（值）允许执行的操作集合，包括操作及有关操作规则。

关系模型数据库操作语言为SQL，主要功能：数据查询（SELECT），数据定义（CREATE，DROP，ALTER），数据操纵（INSERT，UPDATE，DELETE），数据控制（GRANT，REVOTE）。

（3）数据的约束条件

是一组完整性规则的集合，用以限定符合数据模型的数据库状态，以保证数据的正确、有效和相容。

关系模型完整性约束：



3．文件管理系统支持一种简单（或简陋）的数据模型

（通过文件系统支持的简单数据模型来理解数据库系统支持的数据模型，仅仅为了说明问题，通常不称文件系统支持数据模型）

文件数据结构及对文件这种数据结构所进行操作集合是文件管理系统所支持的一种简单数据模型。

将现实世界的信息抽象为文件，对信息的处理通过文件操作实现。

例 学生选课程文件结构（学号、姓名、性别、系别、年龄、籍贯，课程号、课程名、学分，成绩）

文件系统不提供完整性约束功能，由应用程序实现。数据结构及对数据的操作与数据库中的数据模型相比显得落后。

4．数据模型的概念

任何一种数据模型都是严格定义的概念的集合。这些概念必须能够精确地描述系统的静态特性、动态特性和完整性约束条件。

5．任何一种数据库管理系统（DBMS）都基于某种数据模型（或支持某种数据模型）

如关系模型要求：

数据结构：为表，

操作：SQL

约束：实体完整性、参照完整性、用户自定义的完整性

**1.2.2概念模型**

现实世界的信息抽象表示为计算机世界某个数据模型数据结构的桥梁

1．现实世界的信息转换到机器世界信息的处理过程：

（1）现时世界到信息世界（概念模型）：

常用E—R图来描述现实世界的概念模型

（2）信息世界转换为机器世界（概念模型到数据模型的数据结构）

2． 信息世界的基本概念（结合关系模型考虑）

（1）实体：客观存在并可相互区别的事物。

可以是具体的人、事、物，也可以是抽象的概念或联系（如学生的一次选课）。

（2）属性：实体所具有的某一特性。一个实体由若干个属性来刻画。

（3）码：唯一标识实体的属性集。

（4）实体型：用实体名及其属性名集合来抽象和刻划同类实体。

学生：学号、姓名、性别、系别、年龄、籍贯

病人：病历号、病人姓名、年龄、性别

（5）实体集：同型实体的集合

（6）联系：实体内部（实体的各属性间）的联系和实体之间的联系。

**1.2.3关系数据模型**

1． 关系数据模型的数据结构

关系模式的数据结构是一张二维表，它由行和列组成。

一些概念：

关系：表

元组：表行

属性：表列

主码：某个属性组，它可以唯一确定一个元组。

域：属性的取值范围

分量：元组的一个属性值

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）

关系模型中实体与实体间的联系都是用表（关系）来表示的 （与概念模型的实体型及联系对应）

学生（）

课程（）

学生选课（）

关系模型要求关系必须是规范化的，即关系模式必须满足一定的规范条件。最基本一条为每个分量必须是不可分的数据项

2． 关系数据模型的操纵与完整性约束

3． 关系数据模型的存储结构

一个表对应一个DBF文件（DBASE，Foxpro等）

多个表放在一个DB文件中(SYBASE SQL Anywhere,SQL Server等)

**1.3数据库系统结构**

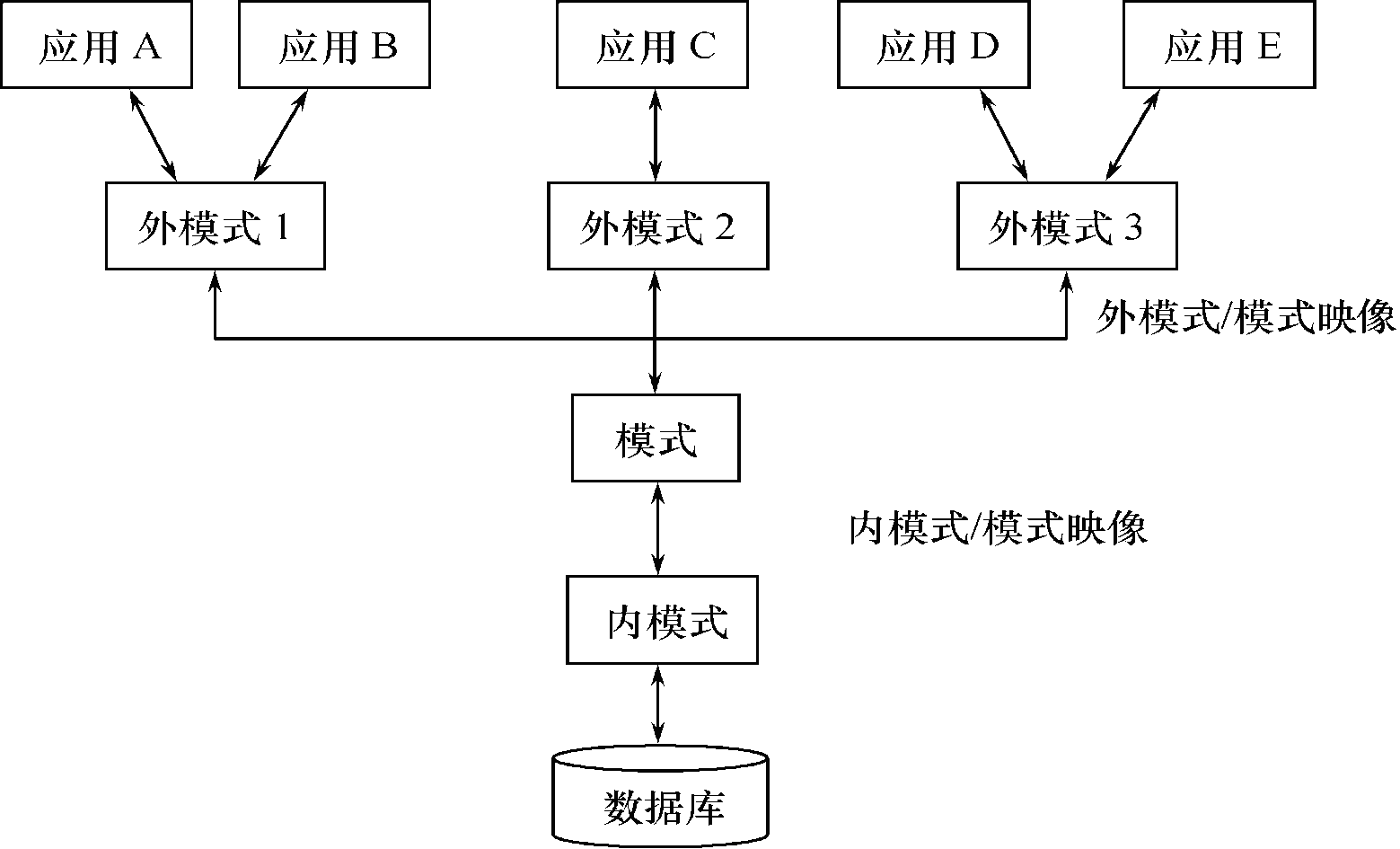
1．关系数据库系统的模式结构

模式：也称逻辑模式，是数据库中全体逻辑结构和特征的描述，是所有用户的公共数据视图。

外模式：也称子模式或用户模式，他是数据库用户（包括应用程序员和最终用户）看见和使用的局部数据的逻辑结构和特征的描述

内模式：内模式也称存储模式，它是数据物理结构和存储结构的描述，是数据在数据库内部的表示方式。

数据库系统的模式结构：数据库系统是由外模式、模式、内模式三级构成。



2．数据库的二级映像功能与数据独立性

外模式/模式映像

它定义了外模式与模式之间的对应关系，保证了数据与程序的逻辑独立性。

模式/内模式映像

它定义了全局逻辑结构与存储结构之间的对应关系，保证了数据与程序的物理独立性。

**1.4数据库系统的组成**

1．硬件平台及数据库

数据库系统对硬件资源的要求

足够大的内存，存放操作系统、数据库管理系统的核心模块、数据缓冲区和应用程序

足够大的磁盘或磁盘阵列等设备存放数据库，有足够大的磁带（或光盘）作数据备份

要求系统有较高的通道能力，以提高数据传送率

2．软件

DBMS

操作系统

与数据库接口的高级语言及其编译系统

以DBMS为核心的应用开发工具

为特定应用环境开发的数据库应用系统

3．人员

数据库管理员

系统分析员

数据库设计人员

应用程序员

(最终用户)

### 1.5.6 课时小结

本章主要内容：

1. 数据库系统概述（4个基本概念、数据管理技术的产生和发展、数据库系统的特点）；

2. 数据模型；

3. 数据库系统结构；

4. 数据库系统的组成。

### 1.5.7 课外补充

无

# 第二章 关系数据库

## 2.1 授课内容

授课科目：数据库原理

授课内容：关系数据库

授课类型：讲授

授课时间：6学时

主讲教师：

## 2.2 教学目标要求

### 2.2.1 能力目标

1. 掌握关系数据库的基本概念、基本理论；
2. 能够充分理解关系模型的三要素；
3. 能够运用所学关系代数知识表达各类运算；
4. 熟悉关系演算的概念。

### 2.2.2 知识目标

1. 掌握关系数据结构及定义；
2. 掌握关系的操作，重点掌握关系的基本操作；
3. 掌握关系的三个完整性约束条件；
4. 掌握常用的关系代数表达方法。

## 2.3 课件分析

概 述： 本章是沿用第一章关系模型的三要素为主线讲解：首先是第一要素数据结构，主要讲的是关系数据结构及其形式化定义，重点介绍笛卡尔积、关系、候选码、主码、主属性、非主属性等基本概念；其次第二要素数据的操作，主要讲的是关系模型中能够进行的操作有哪些，重点介绍并、交、差、选择、投影五种基本操作；第三要素也是本章重点讲解实体完整性、参照完整性、用户定义的完整性三个约束规则，其中参照完整性引入重要的外码概念；最后依次介绍传统的集合运算和专门的关系运算。

教学重点：关系数据结构中的基本概念、关系的基本操作、关系的完整性、专门的关系运算。

教学难点： 参照完整性规则、外码、专门的关系运算。

## 2.4 教学方法

讲授法。

## 2.5 教学过程

### 2.5.1 课前补充

（0分钟）

### 2.5.2 上节回顾

（3分钟）

将第一章中的部分概念进行回顾，尤其重点回顾数据模型的概念，数据模型三要素。

### 2.5.3 作业点评

（0分钟）

### 2.5.4 导入新课

（3分钟）

1970年，E.F.Codd关于关系数据模型的论文被列为从1958年以来的四分之一世纪中具有里程碑意义的研究论文之一，正是有了严谨的理论基础，关系数据库才得以取代层次模型、网状模型成为到目前为止运用最广泛的数据库产品。本章在第一章的基础之上，较深入地介绍关系模型。按照数据模型的三个要素，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，前三节对这三部分进行分别介绍，后两节分别介绍关系代数、关系演算等内容。

### 2.5.5 讲授新课

**2.1 关系数据库概述**

1.关系数据库系统：支持关系模型的数据库系统。

2.关系模型的组成：关系数据结构、关系操作集合、关系的完整性约束条件三部分。

3.关系数据结构：表

特点：简单的数据结构表达丰富的语义，描述现实世界的实体以及实体间的联系（例）

4.关系操作

采用集合操作：操作对象与操作结果为集合，

5.完整性约束

允许定义三类完整性：实体完整性、参照完整性、用户自定义完整性。

关系系统自动支持的完整性：实体完整性、参照完整性（关系的两个不变性）。

**2.2关系数据结构（关系、关系模式、关系数据库）**

1．关系

例1：有表结构（关系模式）：学生（学号、性别）

如果，学号为子界类型 D1=[1..100], 性别为枚举类型D2=（男，女），学生表的最大取值(最大表)：1002=200个元组

例2：有表结构：学生（学号、姓名、性别、系别、年龄、籍贯）

每个列的类型：int,char(8),bolean,char(8),int,char(10)

最大表元组数：实际应用中的具体表：最大表的有意义的子集

（1）域：是一组具有相同数据类型值的集合。表中的每列都与一域（表的数据类型）相关连

（2）笛卡儿积（本质上为最大元组表）

定义：给定一组域D1，D2，….，Dn，这些域可以完全不同，也可以部分或全部相同。D1，D2，….，Dn的笛卡儿积为：

D1╳D2╳…. ╳Dn={（d1,d2,…,dn）∣di∈Di,i=1,2,…,n}

其中每一个元素（d1,d2,…,dn）叫作一个n元组，或简称为元组。元素中的每一个值di叫作一个分量。

若Di（i=1,2,…,n）为有限集，其基数为mi(i=1,2,…,n)，则D1╳D2╳…. ╳Dn的基数为m1m2……mn

（3）关系

定义：D1╳D2╳…. ╳Dn的子集叫作在域D1，D2，….，Dn上的关系，用R（D1，D2，…，Dn）表示。R表示关系的名字，n是关系的目或度

n=1单元关系

n=2二元关系

关系为笛卡儿积的子集，表行——元组，表列——属性（n目关系有n个属性）。

侯选码：若关系中某一属性组的值能唯一标识一个元组，则称该属性组为侯选码

主码：若一个关系中有多个侯选码，则选其中一个作为主码。

主属性：主码的诸属性

非主属性：不包含在任何侯选码中的属性

全码：侯选码为关系的所有属性组

关系的三种类型：基本表（实际存在表）、查询表（查询结果）、视图表（虚表由基本表与其他试图表导出）

基本关系具有6条性质：

① 列是同质的（Homogeneous）

② 不同的列可出自同一个域

其中的每一列称为一个属性

不同的属性要给予不同的属性名

③ 列的顺序无所谓, 列的次序可以任意交换

④ 任意两个元组的候选码不能相同

⑤ 行的顺序无所谓，行的次序可以任意交换

 ⑥ 分量必须取原子值

2．关系模式

定义：关系的描述称为关系模式。它可形式化地表示为R（U，D，DOM，F）。R为关系名，U组成该关系的属性名集合，D属性组所来自的域，DOM属性向域的映象集合，F属性间数据的依赖关系集合。

关系模式通常简记为R（U）或R（A1，A2，…，An），域名及属性向域的映象常常直接说明为属性的类型、长度

型与值：模式是型，关系（表）是模式的值

3．关系数据库

含义：在一个给定的现实世界的领域中，相应于所有实体及实体之间的联系的关系的集合构成一个关系数据库。

关系数据库的型和值：型即关系数据库模式，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。关系模式的值也称为关系数据库，是这些关系模式在某一时刻对应的关系的集合。关系数据库模式与关系数据库通常称为关系数据库。

**2.3关系操作**

常用的操作

查询：选择、投影、连接、除、并、交、差等；更新：增、删、改；**五种基本运算：选择、投影、并、差、笛卡尔积**。其中查询的表达能力是关系操作中的最主要部分。

关系模型中关系操作能力的早期抽象表示（即抽象查询语言，领会语言的含义）：

（1）关系代数语言：用代数方式表示，即用关系的运算来表达查询要求的方式。

（2）关系演算语言：用逻辑方式表示，即用谓词来表达查询要求的方式。

元组关系演算：谓词变元的基本对象是元组变量。

域关系演算：谓词变元的基本对象为域变量。

（3）关系代数、关系演算（元组关系演算、域关系演算）三种语言在表达能力上是等价的。

早期抽象查询语言（关系代数、关系演算语言）的作用：作为评估实际系统中查询语言能力的标准。与具体DBMS实现的实际语言不完全一样（提供许多附加功能，如集函数、关系赋值、算术运算等）

关系数据库的标准语言（SQL—structured query language结构化查询语言）是集数据查询、DDL数据定义、DML数据操纵、DCL数据控制于一体的语言。具有关系代数与关系演算双重特点的语言。

**2.4关系完整性**

1.实体完整性

要点：主属性不能取空值。

现实世界的语义要求：一个基本关系通常对应现实世界的一个实体集，现实世界的实体是可以区分的，即它们具有某种唯一标识，在关系模型中以主码作为唯一标识。

2.参照完整性

关系与关系之间存在着引用关系：因为现实世界的实体之间存在某种联系，而关系模型中实体及实体间的联系都是用关系来描述的。

要点：

外码——基本关系R的非码属性（组）F与基本关系S的主码Ks相对应，则属性（组）F为R的外码。参照关系——R，被参照关系——S

参照完整性规则——基本关系R的外码F与基本关系S的主码Ks相对应，则R上的每个元组在F上的取值要求：或者取空值；或者等于S中的某个元组的主码值。

3.用户自定义完整性

反映某一具体应用所涉及的数据必须满足的语义要求。

通常有三类：非空约束（not null）、唯一约束(unique)、检查约束(check)

2.4 关系代数

1.何为关系代数：是抽象的查询语言，它用关系运算来表达查询。主要研究关系（表）运算，运算对象是关系，运算结果也是关系。

2.关系运算分两类：

（1）传统的集合运算：并、交、差、广义笛卡尔积。运算只涉及行，将关系看成是行（元组）的集合，。

（2）专门的关系运算：选择、投影、连接、除。运算既涉及行也涉及列。

3.传统的集合运算（着重掌握语义）

设关系R、S

R

|  |  |
| --- | --- |
| A | B |
| a1 | b1 |
| a1 | b2 |

S

|  |  |
| --- | --- |
| A | B |
| a1 | b2 |
| a2 | b2 |

（1）并（U）

R ∪ S={ t | t∈R ∨ t∈S } 并由属于R或属于S的元组组成。两集合元组并在一起，去掉重复元组。

关系R、S要求具有相同的目n（都具有n个属性），相应的属性取自同一个域（同类型数据）

（2）差（—）

R — S={ t | t∈R ∧﹁t∈S } 差由属于R并不属于S的元组组成。

关系R、S要求具有相同的目n（都具有n个属性），相应的属性取自同一个域（同类型数据）

（3）交（∩）

R ∩ S={ t | t∈R ∧ t∈S } 由属于R且属于S的元组组成。

关系R、S要求具有相同的目n（都具有n个属性），相应的属性取自同一个域（同类型数据）

（4）笛卡尔积

R ╳ S={trts | tr∈R ∧ ts∈S}

关系R、S分别为m、n目k1．k2个元组，结果为m+n目，元组个数为k1k2个。

例： R

|  |  |
| --- | --- |
| 学号 | 姓名 |
| 1 | A |
| 2 | B |

S

|  |
| --- |
| 课 程 |
| 数据库 |
| 操作系统 |

R ╳ S

|  |  |  |
| --- | --- | --- |
| 学号 | 姓名 | 课程 |
| 1 | A | 数据库 |
| 1 | A | 操作系统 |
| 2 | B | 数据库 |
| 2 | B | 操作系统 |

（广义笛卡儿积与笛卡儿积的比较：）

4. 专门关系运算（运算结果着重掌握语义）

（1）选择

σF（R）={t| t∈R ∧F(t)=“T”} 选择关系R中满足逻辑表达式F为真的元组。

P48 例1．查询信息系的全体学生 例2．查询年龄小于20的元组

例1 σsdept='is'（student）

例2 σsage<20（student）

逻辑表达式F的基本形式为：X1 θ Y1[φ X2 θ Y2]…

θ比较运算符，φ逻辑运算符﹁∧∨，Xi、Yi属性名或常量名或简单函数。

（2）投影

ПA(R)={t[A]| t∈R } A为R中的属性列

例3：学生关系在姓名和系两个属性上的投影。例4：在学生关系中有那些系

例3 Пsname,sdept(student) 例4：Пsdept(student)

注：投影后的重复行应该消除

（3）连接∞

R ∞ S={trts | tr∈R ∧ ts∈S ∧ tr[A] θ ts[B] }

AθB

从广义笛卡尔积R ╳ S中选择满足比较运算表达式AθB的元组，A、B分别为R、S上度数相等且可比的属性组。

例4：R ∞ S

c<d

等值连接：上述连接的特例，θ取=

自然连接：书写格式在连接符下不加比较表达式，比较分量必须是相同的属性组，连接结果去掉重复列。

（4）除（÷）

例6 设关系R（A，B，C），S（B，C，D），求R÷S，

R÷S为新关系P（A），P是R中满足下列条件的元组在A属性列上的投影：元组在A上分量值a的象集BCa包含S在BC上投影的集合。

象集：关系R（X，Z），X和Z为属性组。我们定义，当t[X]=x时，x在R中的象集为：Zx={t[Z]| t∈R ,t[X]=x}，表示R中属性组X上值为x的诸元素在Z 上分量的集合。

R÷S：R（X，Y），S（Y，Z），其中X，Y，Z为属性组

R（X，Y）X=A，Y=BC

S（Y，Z）Y=BC，Z=D

R÷S的结果为新关系P（X），P是R中满足下列条件的元组在X属性列上的投影：元组在X上分量值x的象集Yx包含S在Y上投影的集合。

R ÷ S = {tr[X]| tr∈R ∧Пy(S) 包含于tr[X]的象集}

例：至少选修了课程表中所有课程的学生

Пsno,cno(student\_course) ÷Пcno(course)

假设学生课程数据库中有表：（理解含义）

Student（Sno学号，sname姓名，sex性别，age年龄，sdept所在系），

Course（Cno,Cname,Cpno,Ccredit），

SC(Sno,Cno,Grade)

例8：（错误）

例9：查询至少选修了一门其直接先行课为6号课程的学生姓名

1. 先行课为6号的课程，涉及课程表。P1=σcpno='6'（coruse）

2. 选了上述课程的学生学号P2：P1∞SC 或Пsno(P1∞SC )

3. 根据学号求姓名：P2∞ПSno,Sname(Student),投影出Sname

例10：查询选修了全部课程的学生号码和姓名

1. 所有课程的课程号P1：Пcno(Course )

2. 选修了所有课程的学生号P2：ПSno,Cno(SC)÷P1

3. 学生姓名、学生号:P2∞ПSno,Sname(Student)

### 2.5.6 课时小结

本章主要内容：

1. 关系数据结构及形式化定义；

2. 关系操作；

3. 关系的完整性；

4. 关系代数；

5. 关系演算。

### 2.5.7 课外补充

无

# 第三章 关系数据库标准语言SQL

## 3.1 授课内容

授课科目：数据库原理

授课内容：关系数据库标准语言SQL

授课类型：讲授+练习

授课时间：20学时

主讲教师：

## 3.2 教学目标要求

### 3.2.1 能力目标

1. 掌握SQL语言的基本语法；
2. 掌握用命令方式创建和管理数据库、数据表的方法；
3. 掌握用select查询数据库的方法；
4. 掌握视图的应用方法。

### 3.2.2 知识目标

1. 知道SQL的特点及概念；
2. 学会数据库的定义和删除，表的定义、删除、表结构的修改；
3. 掌握视图的创建方法、视图的定义与删除；
4. 熟练掌握数据操纵语言如插入、更新、删除；
5. 熟练掌握各种查询语句；
6. 理解和掌握索引的设计、创建、使用和维护等功能。

## 3.3 课件分析

概 述： 结构化查询语言是关系数据库的标准语言，也是一个通用的、功能极强的关系数据库语言。其功能不仅仅是查询，而是包括数据库模式创建、数据库数据的插入与修改、数据库安全性完整性定义与控制等一系列功能。

SQL可以分为数据定义、数据查询、数据更新、数据控制四大部分。人们有时把数据更新成为数据操纵，或把数据查询与数据更新合称为数据操纵。本章系统而详尽地讲解了前面三部分的内容。

本章在讲解SQL的同时，进一步讲解了关系数据库系统的基本概念，使关系数据库的许多概念更加具体、更加丰富。

SQL是关系数据库语言的工业标准。目前，大部分数据库管理系统产品都能支持SQL92，本章要求学生理解和掌握关系数据库标准SQL语言，能够熟练使用SQL语言完成各种数据库操作和管理任务，包括使用DDL语句创建、更改和删除数据库、模式和基本表；使用SQL查询语句完成各类查询操作；使用DML语句完成各类更新操作；理解视图的作用。

教学重点：

1.数据定义（表、视图、索引的定义）。表定义与完成性约束定义，数据定义与数据字典，索引能加快查询速度的原因，对视图操作与对基本表操作，视图及外模式

2.数据更新操作与数据完整性维护，数据更新操作与数据库一致性维护问题

3.SQL查询（单表查询、连接查询、嵌套查询、集合查询）相关关系代数表达

教学难点：涉及多表的连接查询

## 3.4 教学方法

问题教学法、讲授法。

## 3.5 教学过程

### 3.5.1 课前补充

（0分钟）

### 3.5.2 上节回顾

（0分钟）

### 3.5.3 作业点评

（0分钟）

### 3.5.4 导入新课

（3分钟）

[SQL](https://baike.baidu.com/item/SQL)即结构化查询语言(Structured Query Language)，是一种特殊目的的编程语言，是一种数据库查询和程序设计语言，用于存取数据以及查询、更新和管理关系数据库系统；同时也是数据库脚本文件的扩展名。SQL语句无论是种类还是数量都是繁多的，很多语句也是经常要用到的，SQL查询语句就是一个典型的例子，无论是高级查询还是低级查询，SQL查询语句的需求是最频繁的。

本章分10次讲述，每次2学时，每次主要讲述以下内容介绍如下：

第一次：介绍安装SQL Server的注意事项，讲解SQL的发展史和SQL的特点，数据库的创建和删除以及模式的创建和删除。

第二次：表的创建、修改和删除。

第三次：索引的创建和删除，以及数据定义部分的习题课并数据定义部分的实验。

第四次：单表查询（若干列、若干元组、对查询结果排序、使用集函数、对查询结果分组）。

第五次：连接查询（等值与非等值连接查询、自身连接查询、外连接查询、复合条件连接）。

第六次：嵌套查询（带IN谓词查询、带比较运算符查询）。嵌套查询（带ANY或ALL谓词查询、带EXISTS谓词子查询），集合查询。

第七次：数据查询部分的习题课，并完成相关实验。

第八次：数据更新，更新操作与数据库的一致性问题（通过数据库管理系统的事务功能解决，以及程序员如何编程解决）。

第九次：视图的创建、删除和更新。

第十次：本章小结以及完成数据更新和视图的相关实验。

### 3.5.5 讲授新课

**3.1 标准SQL语言**

* + 1. **SQL的特点**
* **SQL的特点**

1. **综合统一**
2. **高度非过程化**
3. **面向集合的操作方式**
4. **两种使用方式，统一的语法结构**
5. **简洁易学**
   * 1. **核心功能的9个动词**

|  |  |
| --- | --- |
| SQL功能 | 动词 |
| 数据定义 | CREATE,DROP,ALTER |
| 数据查询 | SELECT |
| 数据操纵 | INSERT,UPDATE,DELETE |
| 数据控制 | GRANT,REVOTE |

各个DBMS产品在实现标准SQL语言时各有差别，一般都作了某些扩充

**3.2数据定义**

数据定义的基本对象为表、索引、视图（导出表）

|  |  |  |  |
| --- | --- | --- | --- |
| 操作对象 | 操作方式 | | |
| 创建 | 删除 | 修改 |
| 表 | CREATE TABLE | DROP TABLE | ALTER TABLE |
| 索引 | CREATE INDEX | DROP INDEX |  |
| 视图 | CREATE VIEW | DROP VIEW |  |

1. 基本表独立存在，不同的RDBMS产品，表的物理存储会有不同的方式：若干基本表存储于一个文件；一个表存储与一个文件。（小型DBMS如DBASE，一个基本表对应一个存储文件）
2. 索引依附于基本表。其作用是为了加快数据的访问速度。两类：聚簇索引与非聚簇索引
3. 视图为导出表（基于基本表的虚表），只存储视图定义，不存放视图对应的数据，数据存放在导出视图的基本表中。（对视图操作最终通过视图的定义变为对基本表操作）
   * 1. **数据库的定义和删除**

create database <database\_name*>*

drop database <database\_name*>*

* + 1. **模式的定义和删除**

CREATE SCHEMA <**模式名**> AUTHORIZATION <**用户名**> **[<表定义> |<视图定义>** |**<授权定义>**]

* + 如果没有指定模式名，则模式名隐含为用户名
  + 权限：使用该命令，用户必须具有DBA权限，

或获得了DBA授权CREATE SCHEMA 的权限

* DROP SCHEMA <模式名> <CASCADE | RESTRICT>
* CASCADE(级联)

删除模式的同时把该模式中所有的数据库对象全部删除

* RESTRICT(限制)

如果该模式中定义了下属的数据库对象（如表、视图等），则拒绝该删除语句的执行

当该模式中没有任何下属的对象时才能执行

* + 1. **定义、删除、修改基本表**
* **基本数据类型**

****

* **定义基本表**

CREATE TABLE <表名>

<列名> <数据类型> [ <列级完整性约束条件> ]

[，<列名> <数据类型> [ <列级完整性约束条件>] ] …

[，<表级完整性约束条件> ] ）;

* + <列级完整性约束条件>：涉及相应属性列的完整性约束条件
  + <表级完整性约束条件>：涉及一个或多个属性列的完整性约束条件
* **完整性规则**

常用完整性约束

* 主码约束： PRIMARY KEY
* 唯一性约束：UNIQUE
* 非空值约束：NOT NULL
* 参照完整性约束： FOREIGN KEY
* **列级约束与表级约束的区别**

如果完整性约束涉及到该表的多个属性列，必须定义在表级上，否则既可以定义在列级也可以定义在表级

用SC表说明

create table student\_course

( sno char(6) ,

cno char(6),

grade int,

primary key(sno,cno)

);

* **约束的定义的形式**

在不同的数据库管理系统产品上某些地方有所不同，需要查相关产品的资料。

外键约束

student(no, name, sex , age , dept) //外码 dept

department(dept, dept\_name, location) //主码

create table student

(no char(6) primary key，

name char(6),

sex char(1),

age integer,

dept char(8) references department,//(dept),

//列级约束// constraint f\_k foreign key (dept) references department,//(dept\_code), // 表级约束

);

create table department

(

dept char(8) primary key,

dept\_name char(20),

location char(20),

);

* **修改基本表**

ALTER TABLE <表名>

[ ADD [COLUMN]<新列名> <数据类型> [ 完整性约束 ] ]

[ ADD <表级完整性约束>]

[ DROP [COLUMN]<列名>[CASCADE|RESTRICT] ]

[ DROP CONSTRAINT<完整性约束名[CASCADE|RESTRICT] ]

[ ALTER COLUMN <列名> <数据类型> ]；

* + <表名>：要修改的基本表
  + ADD子句：增加新列和新的完整性约束条件
  + DROP子句：删除指定的完整性约束条件
  + ALTER COLUMN子句：用于修改列名和数据类型

举例：

drop table student;

create table student

(

dept\_code char(8) primary key,

name char(20) unique,

sex char(1)

);

//增加新列

alter table student add age integer not null;

alter table student add age integer not null unique;

//删除约束

alter table student drop unique(name);

//修改已经存在的列

alter table student modify sex char(2) not null;//char(1)改为char(2),并加上约束

删除基本表（一般用得不多）

drop table<表名>；

drop table student;

* **删除基本表**

DROP TABLE <表名>［RESTRICT| CASCADE］；

* + RESTRICT：删除表是有限制的

欲删除的基本表不能被其他表的约束所引用

如果存在依赖该表的对象，则此表不能被删除

* + CASCADE：删除该表没有限制

在删除基本表的同时，相关的依赖对象一起删除

* + 1. **建立与删除索引**
* **索引的作用**
  + 建立索引是加快表查询速度的有效手段。
  + 原理示例图：student(no,name,sex,age)

索引 表

003 赵六 男 52岁….

005 杨小 男 23岁

002 高卫 男 32岁

004 吕红 女 21岁

006 杨阳 男 23岁

001 李敏 女 33岁

…

001

002

003

004

005

006

…

004

* + - * + 索引中记录索引项与其记录实际存放地址。
        + 在查找某一记录（004号吕红）时，DBMS先通过索引文件找到相应记录的物理地址，通过物理地址找到记录内容。
        + 实际的索引结构往往采用B+树结构。
  + 索引依附于基本表
  + 索引的使用，通过查询过程说明

select no,name,age from student where no=006;

* **索引的类别：唯一索引、聚簇索引。**
  + 唯一索引：每个索引值只对应唯一的数据记录

例：CREATE UNIQUE INDEX in\_name ON student(name);

* + SQL SERVER在带PRIMARY KEY约束或UNIQUE约束时自动创建一个唯一索引
* 聚簇索引：索引项的顺序与表中记录的物理顺序一致的索引组织。
* 显然，一个基本表上最多只能建立一个聚簇索引。（因物理顺序只有一个）
* 经常更新的列不宜建立聚簇索引：导致记录物理顺序的变更，代价较大。
* **建立索引的一般格式**

CREATE [UNIQUE][CLUSTER]INDEX <索引名>

ON <表名>（<列名>[<次序>][，<列名>[<次序>]…]）;

索引一经建立，就由系统使用和维护，不需要用户干预。

如：Select from student where sno=’98001’;系统自动根据索引优化查找。

* **是否需要创建索引的考虑**

索引是有用的（可以加速数据访问速度。如没有索引，在查找一个主题的信息时，必须一次一页地扫描整个文本），但索引要耗用磁盘空间，并引起系统开销和增加维护的费用。在使用索引时要考虑以下事实和要点：

* 修改一个索引列上的数据时，要更新相关的索引
* 维护索引需要时间和资源，所以不要创建不经常使用的索引
* 小表上的索引没有多少好处。
* **删除索引**

如果数据增加、删除、修改频繁，系统需要花费许多时间来维护索引，这时需要删除一些不必要的索引。

DROP INDEX <索引名>；

* + 1. **查询单表**
* **基本语法**

SELECT [ALL|DISTINCT]〈目标列表达式〉[,〈目标列表达式>] …

FROM 〈表名或视图名〉[，〈表名或视图名〉] …

[WHERE <条件表达式>]

[GROUP BY 〈列名〉[，〈列名〉]…

[HAVING <内部函数表达式>] ]

[ORDER BY 〈列名〉 [ASC│DESC] [，〈列名〉[ASC│DESC]]…]

* **SELECT语句的含义**
* 根据WHERE子句中的条件表达式，从FROM子句中的基本表或视图中找出满足条件的元组
* 按SELECT子句中的目标字段，选出元组中的分量形成结果表
* GROUP BY子句将结果按字段分组，每个组产生结果表中的一个元组
* 通常在每组中作用库函数，分组的附加条件用HAVING短语给出只有满足内部函数表达式的组才予输出
* 如果有ORDER BY子句，则结果表要根据指定的字段按升序或降序排列
* **选择表中的若干列**
* 查询指定列

select sno,sname from student;

select sname,sno,sdept from student;

* 查询全部列

（1）依表中的列序

select \* from student;

select \* from sc;

（2）依用户指定列序

* 查询经计算的列

a. 计算列

select sname,1999-sage from student;

select sname,'生日:',1999-sage，islower(sdept) from student;

目标列表达式：

属性列

有关表达式：算术表达式、字符串常量、函数等。

b. 列的别名表示

select sname,1999-sage "生日" from student;

* **选择表中的若干元组**

消除查询结果重复元组

select sno from sc;//重复元组，等价select all sno from sc;

select distinct sno from sc;//唯一，缺省为all

* **查询满足条件的元组**

条件：where <条件表达式>

* 比较大小

select grade from sc where not grade<60;//演示去掉not情况

比较运算符有：=，> < >= <= != , <>, !>, !<, not+前述比较运算符

* 确定范围between…..and 和not between…..and

select sname,sage from student where sage between 20 and 30;

select sname,sage from student where sage not between 20 and 30;

* 确定集合：in、not in

select sname,sage,sdept from student where sdept not in ('信工系');//或in ('信工系');

select sname,sage from student where sage in (20,21);

* 字符匹配
* %任意长度字符串(可以0个字符)

select sname,sage,sdept from student where sdept not like '化工%';

select sname,sage,sdept from student where sdept like '%系';//或not like '%系';

* \_任意单个字符

select sname,sage,sdept from student where sname like 'a\_';//或not like 'a\_';

* 转义符ESCAPE’<换码字符>’

select cno,cname from course where cname like '%\\_%' escape'\';

select cno,cname from course where cname like '%\_%';

* 涉及空值is null, is not null

select \* from course where ccredit is null; //或ccredit is not null;

* 多重条件

用and或 or 连接多个查询条件，考虑优先级

in 与 or 的关系

select sname from student where sdept = '信息系’ and sage<21;

* **对查询结果排序**
* 查询的输出结果
* 查询语句中不指定查询结果的显示顺序：

不能保证结果集显示顺序（SQL SERVER）。书中指方便顺序（通常元组在表中的先后顺序）

查询语句中指定查询结果显示顺序：按指定顺序显示

举例

select \* from sc

select \* from sc order by sno desc, cno; //缺省为升序asc

|  |  |  |
| --- | --- | --- |
| **Sno** | **Cno** | **Grade** |
| 97003 | 1 | null |
| 97003 | 3 | 95 |
| 97002 | 2 | 89 |
| 97002 | 3 | 89 |
| 97001 | 1 | 90 |

|  |  |  |
| --- | --- | --- |
| **Sno** | **Cno** | **Grade** |
| 97001 | 1 | 90 |
| 97002 | 2 | 89 |
| 97002 | 3 | 89 |
| 97003 | 3 | 95 |
| 97003 | 1 | null |

select \* from sc where grade is not null

select \* from sc

where grade is not null

order by sno ,grade desc

//select cno,sno,grade from sc where cno=3 order by grade desc;

//select cno,sno,grade from sc where cno in (1,2,3) order by grade desc;//空值排在最前面

//select cno,sno,grade from sc where cno in (1,2,3) order by grade;//空值排在最后面，缺省表示升序asc

select sno,cno,grade from sc where cno in (1,2,3) order by sno asc,grade desc;//学号升序，同一学号成绩降序。

* **使用集函数**
* 统计元组个数count(\*)

count(\*)

count(distinct \*) //语句执行错误（没有指定列名）

//select “行数”=count(\*) from student;

//或select count(\*) “行数” from student;

//或select count(\*) as “行数” from student;

//select count(distinct \*) from student; //书上语句执行错误（没有指定列名）

* 统计一列中值的个数

count(distinct <列名>)

count( <列名>)

select count(sno) from sc;//5个

select count(sno) from sc;//5个

select count(distinct sno) from sc;//3个

select count(distinct sno) “选课学生数“ from sc;//别名

选课学生数

3

* 计算一列值的平均值

select avg(grade) from sc;//空值不计算

计算1号课程的平均成值

* select avg(grade) from sc where cno=1;//空值不计算
* 计算一列最大值、最小值、平均值、总和
* 查询某一课程号的最高分与最低分

select max(grade), min(grade) from sc where cno=1;

* 查询所有课程的最高分与最低分

select max(grade) “最大”,min(grade) “最小”, avg(grade)”平均”,sum(grade)”总和” from sc;

select max(distinct grade) “最大”,min(distinct grade) “最小”, avg(distinct grade)”平均”,sum(distinct grade)”总和” from sc;

* **对查询结果分组**

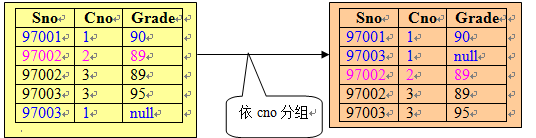
例 查询各课程的课程号与相应的选修人数

select cno,count(sno) from sc group by cno;

select cno,avg(grade) from sc group by cno;

* 不加集函数分组select cno from sc group by cno;

select cno,sno from sc group by cno;//执行出错,分组条件应改为group by cno,sno



说明：分组集函数的作用对象为组

|  |  |
| --- | --- |
| Cno | count(sno) |
| 1 | 2 |
| 2 | 1 |
| 3 | 2 |

不分组集函数作用于整个查询结果

|  |  |
| --- | --- |
| Cno | Avg(Grade) |
| 1 | 90 |
| 2 | 89 |
| 3 | 92 |

分组后只输出满足条件的组

求每门课的平均成绩：

|  |  |
| --- | --- |
| Cno | Avg(Grade) |
| 1 | 90 |
| 2 | 89 |
| 3 | 92 |

* **where与having子句的区别；**

where作用于基本表

having作用于分组

* + 1. **连接查询**
* **等值与非等值连接查询**
* 等值连接：例：查询每个学生及其选修课程情况。

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| student | Sno | Sname | Ssex | Sage | Sdept |  | sc | Sno | Cno | Grade |
|  | 97001 | 李勇 | F | 22 | 信工系 |  | 97001 | 1 | 90 |
| 97002 | 刘晨 | F | 21 | 信工系 | 97002 | 2 | 89 |
| 97003 | 王名 | F | 20 | 信工系 | 97002 | 3 | 89 |
| 97004 | 张立 | M | 22 | 信工系 | 97003 | 3 | 95 |
| 97005 | Ee | M | 21 | 信工系 | 97003 | 1 | null |
| 97101 | Ee | M | 18 | 体育部 |  |  |  |

结果表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Student  Sno | Sname | Ssex | Sage | Sdept | SC  Sno | Cno | Grade |
| 97001 | 李勇 | F | 22 | 信工系 | 97001 | 1 | 90 |
| 97002 | 刘晨 | F | 21 | 信工系 | 97002 | 2 | 89 |
| 97002 | 刘晨 | F | 21 | 信工系 | 97002 | 3 | 89 |
| 97003 | 王名 | F | 20 | 信工系 | 97003 | 3 | 95 |
| 97003 | 王名 | F | 20 | 信工系 | 97003 | 1 | null |

执行过程：1）选表1中的第一个元组；2）与表2中的所有元组比较，满足连接条件（等值）的元组进行拼接，形成结果表中的一个元组；3）表1中选下一元组重复步骤2）直到结束。

select student.\*,sc.\*

from student,sc

where student.sno=sc.sno;//等值

* 非等值连接

where 后为其它比较运算符：>,<,<>,>=,<=等。

笛卡儿积

例：select student.\*, sc.\* from student,sc ;

自然连接

例：select student.sno, sname, ssex, sage, sdept, cno,grade

from student,sc

where student.sno=sc.sno;

表名前缀的作用：避免混淆。属性名在不同表中如为唯一，则可以省略表名前缀。

cno只出现在sc表中，所以可不加前缀。Sno则不行。

* **自身连接**

例：查询每一门课的间接先修课号（先修课的先修课）

|  |  |  |  |
| --- | --- | --- | --- |
| Cno | Cname | Cpno | Ccredit |
| 1 | '计算机语言\_pascal' | 6 | null |
| 2 | '操作系统\_dos' | 1 | 1 |
| 3 | '数据库原理' | 2 | 2 |
| 4 | '编译原理' | 1 | 1 |

select f.cno,second.cpno

from course f , course second /\*取两个别名f与second\*/

where f.cpno=second.cno;

结果：

First.cno, Second.Cpno

(用first出错，为保留字)

思考题：每门课的先修课程名？

select cno,cname from course

where cno in (

select second.cpno

from course f , course second /\*取两个别名f与second\*/

where f.cpno=second.cno);

* **外连接**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| student | Sno | Sname | Ssex | Sage | Sdept |  | sc | Sno | Cno | Grade |
|  | 97001 | 李勇 | F | 22 | 信工系 |  | 97001 | 1 | 90 |
| 97002 | 刘晨 | F | 21 | 信工系 | 97002 | 2 | 89 |
| 97003 | 王名 | F | 20 | 信工系 | 97002 | 3 | 89 |
| 97004 | 张立 | M | 22 | 信工系 | 97003 | 3 | 95 |
| 97005 | Ee | M | 21 | 信工系 | 97003 | 1 | null |
| 97101 | Ee | M | 18 | 体育部 |  |  |  |

学生表、学生选课表 的等值连接结果表：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Student  Sno | Sname | Ssex | Sage | Sdept | SC  Sno | Cno | Grade |
| 97001 | '李勇' | 'f' | 22 | '信工系' | 97001 | 1 | 90 |
| 97002 | '刘晨' | 'f' | 21 | '信工系' | 97002 | 2 | 89 |
| 97002 | '刘晨' | 'f' | 21 | '信工系' | 97002 | 3 | 89 |
| 97003 | '王名' | 'f' | 20 | '信工系' | 97003 | 3 | 95 |
| 97003 | '王名' | 'f' | 20 | '信工系' | 97003 | 1 | null |

从连接操作的结果可看出，只有满足条件的元组才能作为结果输出

对学生表来说，没有选课的学生（97004．97005．97101）学其基本情况信息没有出现在结果表中。如希望这些信息出现在基本表中，如下表：

学生表、学生选课表 的等值连接的结果表：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Student  Sno | Sname | Ssex | Sage | Sdept | SC  Sno | Cno | Grade |
| 97001 | '李勇' | 'f' | 22 | '信工系' | 97001 | 1 | 90 |
| 97002 | '刘晨' | 'f' | 21 | '信工系' | 97002 | 2 | 89 |
| 97002 | '刘晨' | 'f' | 21 | '信工系' | 97002 | 3 | 89 |
| 97003 | '王名' | 'f' | 20 | '信工系' | 97003 | 3 | 95 |
| 97003 | '王名' | 'f' | 20 | '信工系' | 97003 | 1 | null |
| 97004 | 'dd' | 'm' | 22 | '信工系' | null | null | null |
| 97005 | 'ee' | 'm' | 21 | '信工系' | null | null | null |
| 97101 | 'ee' | 'm' | 18 | '体育部' | null | null | null |

select student.sno, sname, ssex, sage, sdept, cno,grade

from student left outer join sc

on student.sno=sc.sno;

* **left outer join左连接，**

on用where代替将出错

* **right outer join右连接**
* **复合条件连接**

含义：where子句中有多个查询条件

例：查询选修了2号课程且成绩在90分以上所有学生的学号、姓名

学生表、学生选课表 的等值连接结果表：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Student  Sno | Sname | Ssex | Sage | Sdept | SC  Sno | Cno | Grade |
| 97001 | '李勇' | 'f' | 22 | '信工系' | 97001 | 1 | 90 |
| 97002 | '刘晨' | 'f' | 21 | '信工系' | 97002 | 2 | 89 |
| 97002 | '刘晨' | 'f' | 21 | '信工系' | 97002 | 3 | 89 |
| 97003 | '王名' | 'f' | 20 | '信工系' | 97003 | 3 | 95 |
| 97003 | '王名' | 'f' | 20 | '信工系' | 97003 | 1 | null |

select student.sno,sname

from student,sc

where student.sno=sc.sno and sc.cno=3 and sc.grade>90;

例：查询每个学生（学号，姓名）选修的课程名及其成绩

select student.sno,sname,cname,grade from student,sc,course where student.sno=sc.sno and sc.cno=course.cno;

* + 1. **嵌套查询**
* **带有IN条件的子查询**

例：查询与“刘晨”在同一个系学习学生的学号、姓名、系

第一步：刘晨所在的系；

select sdept

from student

where sname='刘晨'；

第二步：查询与“刘晨”在同一个系学习学生的学号、姓名、系

select sno,sname,sdept

from student

where sdept in (select sdept from student where sname='刘晨'); //

select sdept from student where sname='刘晨'为子查询

查询结果包括刘晨本人

DBMS处理：第一步求解子查询，确定刘晨所在的系；第二步求解父查询

如果查询结果中不包括刘晨本人，where中加条件：and sname<>'刘晨'

本查询确切知道内层查询结果是单值，in可以用比较运算符=代替。（根据题目需要也可以用其它比较符）

本例的查询可以通过表的自连接查询来完成（见P88）

例 查询选修了课程名为‘数据库原理’的学生学号和姓名。

|  |  |  |  |
| --- | --- | --- | --- |
| Cno | Cname | Cpno | Ccredit |
| 1 | '计算机语言\_pascal' | 6 | null |
| 2 | '操作系统\_dos' | 1 | 1 |
| 3 | '数据库原理' | 2 | 2 |
| 4 | '编译原理' | 1 | 1 |

已知条件：课程名为‘数据库原理’（课程表course）

查找：的学生学号和姓名（学生表student）

步骤分析：1）根据‘数据库原理’在course表中查找cno；2）根据cno在sc表中求sno；3）根据sno在student表中求sname。

1）Select cno from course where cname='数据库原理'

select sno from sc where cno in (1)

select sno,sname from student where sno in (2);

语句合并：

select sno,sname

from student where sno in

(select sno from sc where cno in

(Select cno from course where cname='数据库原理))

注：

本例查询可以通过连接查询实现（见P90）

本查询明确知道最内层的查询结果为单值，所以in也可以用比较运算符=代替，同上例

用连接查询实现

select student.sno, sname from student, sc, course

where student.sno=sc.sno and

sc.cno=course.cno and

course.cname='数据库原理'; //课本上语句执行出错，因为sno前无student。

* **2．带有比较运算符的子查询**

在上例中已经讲述

* **3．带ANY或ALL的子查询**

1）运算符及语义

|  |  |
| --- | --- |
| ANY或ALL运算符 | 等价的集函数或IN |
| > ANY | >MIN |
| <ANY | <MAX |
| >= ANY | >=MIN |
| <= ANY | <=MAX |
| = ANY | IN |
| != ANY 或 <> ANY | ---- |
| > ALL | > MAX |
| < ALL | < MIN |
| >= ALL | >= MAX |
| <= ALL | <= MIN |
| = ALL | --- |
| != ALL 或 <> ALL | NOT IN |

例：查询其他系中比“信工系”任一学生年龄小的学生名单

SELECT Sname,Sage

from student

where sage <ANY

(SELECT sage from student where sdept=’信工系’)

and sdept <>’信工系’ /\*红部分不加情况\*/

order by sage desc;

注：

\*蓝部分等价：sage <

(SELECT max(sage) from student where sdept=’信工系’)

\*书中错误：>不正确

例：查询其他系中比“信工系”所有学生年龄都小的学生名单

SELECT Sname,Sage

from student

where sage <All

(SELECT sage from student where sdept=’信工系’)

and sdept <>’信工系’ /\*红部分不加情况\*/

order by sage desc;

注：

\*蓝部分等价：sage <

(SELECT min(sage) from student where sdept=’信工系’)

需要特别指出：子查询中不能使用排序子句，排序子句永远只能对最终查询结果排序

* **4．带有EXISTS和NOT EXISTS关键字的相关子查询**

例：查询所有选修了1号课程的学生姓名

学生表：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sno | Sname | Ssex | Sage | Sdept |
| 97001 | '李勇' | 'f' | 22 | '信工系' |
| 97002 | '刘晨' | 'f' | 21 | '信工系' |
| 97003 | '王名' | 'f' | 20 | '信工系' |
| 97004 | '张立' | 'm' | 22 | '信工系' |
| 97005 | 'ee' | 'm' | 21 | '信工系' |
| 97101 | 'ee' | 'm' | 18 | '体育部' |

课程表：

|  |  |  |
| --- | --- | --- |
| Sno | Cno | Grade |
| 97001 | 1 | 90 |
| 97002 | 2 | 89 |
| 97002 | 3 | 89 |
| 97003 | 3 | 95 |
| 97003 | 1 | null |

思考：

\*方法1：用前述的不相关子查询实现（子查询与父查询不相关）（IN、 比较、 ANY或ALL）

1）根据cno求学号集合

SELECT Sno FROM SC WHERE Cno=1

2）SELECT sname FROM student

WHERE sno in (2))

即：SELECT sname FROM student

WHERE sno in

(SELECT Sno FROM SC WHERE Cno=1)

\*方法2：用相关的子查询实现：

1）每个学生{从学生（Student）表中依次取每个学生的学号(Sno)}

2）选修了1号课程（SC表中存在1）中的学号选修了课程号1的元组）

SELECT Sname

FROM Student

* 不相关子查询：先执行子查询，后执行外查询
* 相关子查询：子查询的的查询条件依赖于外层父查询的某个属性值，外查询的结果集为满足子查询的行。

（例中揭示：对外层查询的每一元组，根据它与内层查询的相关属性值处理内查询，内层的查询结果非空，则取此元组放入结果表中）

* + 1. **集合查询**
* **并（UNION）**

例：查询信工系学生及年龄不大于19的学生

SELECT \* FROM Student WHERE Sdept=’信工系’

Union

SELECT \* FROM Student WHERE Sage<=19;

同：select \* from yj\_student where sdept='j' or sage<=19;

例：选修了1号或选修了2号课程的学生

* **交**

例：查询信工系学生与年龄不大于19的学生的交集

SELECT \* FROM Student

WHERE Sdept=’信工系’ AND Sage<=20;

例：查询选修了2号与选修了3号课程的学生交集（既选修了2号课程又选修了3号课程）

SELECT Sno FROM SC

WHERE cno=2 AND

sno IN (SELECT Sno FROM SC WHERE Cno=3);

如下实现是否正确：

SELECT Sno FROM SC

WHERE cno=2 AND Cno=3

* **差**

例：查询信工系学生与年龄不大于20的学生差集

SELECT \* FROM Student

WHERE Sdept=’信工系’ AND Sage>20;

例：查询选修了2号与选修了3号课程的学生差集（选修了2号但没选修3号课程的学生）

SELECT Sno FROM SC

WHERE cno=2 AND

sno NOT IN (SELECT Sno FROM SC WHERE Cno=3);

**3.3 数据更新**

数据更新包括插入、修改、删除三条语句

* **插入数据**
* 两种形式

插入单个元组（两种情况），

插入子查询结果

* 数据的完整性

插入操作时，插入数据必须满足完整性约束条件

* 插入单个元组

表名后不加属性：每属性值需要全部列出。

insert into sc values(97006,1,null);

表名后加属性：值与属性需要一一对应，未列出属性自动取空值。

insert into sc(sno,cno) values(97006,1);

成绩自动取空值

* 插入子查询

将子查询结果的一组元组插入指定表中。

例：对每个系，求学生的平均年龄，并把结果存入数据库

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sno | Sname | Ssex | Sage | Sdept |
| 97001 | '李勇' | 'f' | 22 | '计算机' |
| 97002 | '刘晨' | 'f' | 21 | '计算机' |
| 97003 | '王名' | 'f' | 20 | '信工系' |
| 97004 | '张立' | 'm' | 22 | '信工系' |
| 97005 | 'ee' | 'm' | 21 | '信工系' |
| 97101 | 'ee' | 'm' | 18 | '体育部' |

//1） 结果数据库表

drop table dept\_avgage;

create table dept\_avgage

( sdept char(20) primary key,

avgage int,

);

//2） 按系分组求平均年龄

insert

into dept\_avgage//(sdept,avgage)

select sdept,avg(sage)

from student

group by sdept;

一般格式：P99

insert

into <表名>[(<属性列1>[,<属性列2>]…)]

子查询；

* **修改数据**
* 修改某一元组

例：将97001学生的年龄改为18

update student //<表名>

set sage=18 //<列名>=<表达式>[,<列名>=<表达式>]…

where sno=97001; //[WHERE<条件>];

* 修改多个元组

例：将信工系学生转为电机系学生

update student

set sdept='电机系'

where sdept=’信工系’;

例：将学生年龄增加1

update student

set sage=sage+1;

一般格式：P99

UPDATE <表名>

SET <列名>=<表达式>[,<列名>=<表达式>]…

[WHERE<条件>];

* 带子查询的修改语句

例：将计算机系全体学生成绩置0

update sc

set grade=0

where sno in

(select sno

from student

where student.sno=sc.sno and student.sdept='计算机');

update yj\_sc

set grade=0

where '计算机' =

(select sdept

from yj\_student

where yj\_student.sno=yj\_sc.sno);

修改操作与数据库的一致性

出现数据不一致性举例

update 语句一次只能修改一个表，要修改多个表需要执行多次操作。

* **删除数据**
* 删除一个元组的值

delete

from student

where sno=97001;

* 删除多个元组

//删除若干元组

delete

from student

where sdept='信工系';

//删除所有元组

delete

from sc;

* 带子查询删除

删除“计算机”所有学生的选课记录

delete

from sc

where '计算机'=

(select sdept

from student

where student.sno=sc.sno); //SC中所删除学号所在的系

**3.4 视图**

* **定义视图**
* 建立(定义)视图

例1 建立信工系学生的视图

drop view is\_student;

create view is\_student /\*(sno\_1,sname\_2,sage\_3)\*/

as

select sno,sname,sage

from student

where sdept='信工系';

//建立视图的一般格式：

create view <视图名>[(<列名>[，<列名>]…)]

as <子查询>

[with check option];

说明：

\*子查询可以是任意复杂的select语句，但通常不允许含有order by 子句和distinct短语。

\*with check option 保证更新、插入、删除时满足视图定义中的谓词条件

\*省略组成视图的各个属性列名

* 视图属性列隐含由子查询中的SELECT子句目标列中的诸字段组成。

\*下列三种情况必须明确指定组成视图的所有列名

\*\*其中某个列不是单纯的属性名，而是集函数或列表达式。

\*\*多表连接时选出了几个同名列作为视图的字段。

\*\*需要在视图中为某个列名用新的更合适的名字。(个人喜好)

\*视图是一个虚表

\*\*从一个或多个基本表（或视图）导出的表。

\*\*在数据库（数据字典）中只存放视图的定义，而不存放视图所对应的数据，这些数据仍然放在原来的基本表中。

\*\*定义视图时，并不执行其中的SELECT语句。

* **查询视图**

select \* from is\_student;

select sno,sname,sage from is\_student

where sage<18;

\*执行过程

进行有效性检查：检查涉及的表、视图等是否存在于数据库中，

2）转换成对基本表的查询：从数据字典中取出查询涉及到视图的定义，将定义中的子查询和用户对视图的查询结合起来。

3）执行经过修正的查询

select sno,sname,sage from is\_student

where sage<18;//修正后的查询语句为：

select sno,sname,sage

from student

where sdept='信工系' and sage<18;

\*用户就可以像对基本表进行查询一样对视图查询

* **建立在基本表上的单表视图**
* 建立在基本表上的单表视图有三种形式

行列子集视图、带表达式视图、分组视图

//行列子集视图的概念（重要）

从单个基本表中导出，只是去掉基本表的某些行和某些列，但保留了码。（前例 建立信工系学生的视图）

* 带表达式视图

定义一个反映学生出生年份的视图

create view birth\_st(sno,sname,sbirth)

as

select sno,sname,1996-sage

from student;

select \* from birth\_st;

* 分组视图

例：将学生的学号及其平均成绩定义为一个视图

create view s\_g(sno,gavg)

as

select sno,avg(grade)

from sc

group by sno;

select \* from s\_g;

* **建立在基本表上的多表视图（普通多表、带表达式、分组）**

例：建立信工系选修了1号课程的学生视图，视图属性包括sno,sname,grade

drop view is\_student1;

create view is\_student1(sno, sname, sage)

as

select student.sno,sname,grade

from student,sc

where sdept=’计算机’ and student.sno=sc.sno

and sc.cno=1;

* **建立在视图上的视图**

drop view is\_student2;

create view is\_student2//(sno, sname, sage)

as

select sno,sname,grade

from is\_student

where grade>=90;

* **删除视图**

drop view <视图名>;

* **查询视图**

查询的执行过程为（前例已述）

效性检查：检查涉及的表、视图等是否存在于数据库中，

转换成对基本表的查询：从数据字典中取出查询涉及到视图的定义，将定义中的子查询和用户对视图的查询结合起来。

执行经过修正的查询

一般情况下视图查询的转换是直截了当的

例：在信息系学生的视图中，找出年龄小于20岁的学生

SELECT Sno,Ssge

From is\_student

Where sage<20

Is\_student 视图定义中的子查询

SELECT Sno,Sname,Ssge

From student

Where sdept=’信工系’

修正后对基本表的查询

SELECT Sno,Ssge

From student

Where sdept=’信工系’ and sage<20

在有些情况下转换不能直接进行

例 在S\_G视图上查询平均成绩在90份以上学生的学号和成绩P106

create view s\_g(sno,gavg)

as

select sno,avg(grade)

from sc

group by sno;

以下查询：

select \* from S\_G where gavg>=90;

将被转换为：

select sno,avg(grade)

from sc

where avg(grade)>=90

group by sno; //where子句中是不能用集函数作为条件表达式，所以此语句不能被正确执行。

正确的查询语句应该是：

select sno,avg(grade)

from sc

group by sno

having avg(grade)>=90;

//目前多数关系数据库系统不能正确转换成这种方式，但SQL anywhere能正确转换成这种形式，即上述查询语句能正确执行。

注意要点

一般来说DBMS对行列子集视图的查询均能直接进行正确转换，其它视图不一定能保证，对这类视图进行查询时应尽量避免视图中的特殊属性出现在查询中。实际应用时需要查有关资料，或直接在所选的DBMS产品上做尝试。

* **更新视图**

目前各个关系数据库系统产品一般都只允许更新行列子集视图，而且不同的系统还有进一步的规定，使用时需要查相关的资料。

\*with check option 保证更新（插入、修改、删除）时满足视图定义中的谓词条件（子查询中的条件表达式）

drop view is\_student;

create view is\_student //属性考虑

as

select sno,sname,sage//,sdept//分两种情况考虑

from student

where sdept='信工系';

// with check option;

//

插入

视图中包含系列（sdept）

insert into is\_student values(97008,'wwww',19,'信工系');//有无 with check option 都能执行

insert into is\_student values(97009,'wwww',19,'体育部');//无——执行，有不执行

视图中不包含系列（sdept）

insert into is\_student values(95029,'wwww',19);

在SQL ANYWHERE环境中（与教材情况不同）

\*无with check option --执行，但不能自动加上‘信工系’

\*有with check option不执行

书上的插入原理

insert into is\_student values(97008,'wwww',19);

\*执行过程：

转换为对基本表的更新

insert into student values(97008,'wwww',19,'信工系');

//在有些数据库产品中不能实现（在SQL Anywhere 中信工系没有自动加上）

修改

sno sname sage sdept

97009 wwww 19 体育部

95029 wwww 19

2001 a 16 计算机

2002 b 19 计算机

2003 aaaaa 20 信电

2004 d 15 信电

2005 e 20 化工

update yj\_is\_student

set sname='aaaaa'

where sno=2001;

同前转换为对基本表的修改

\*执行过程

（1）进行有效性检查：检查涉及的表、视图等是否存在于数据库中，

（2）转换成对基本表的更新：从数据字典中取出该语句所涉及到视图的定义，将定义中的子查询和用户对视图的更新操作结合起来。

（3）执行经过修正的更新操作

前例转换后的更新语句为：

update student

set sname='aaaaa'

where sno=2001 and sdept='信工系';

不管有无with check option，都只能修改信工系学生记录

删除元组

delete from is\_student

where sno=2003;//能被删除

delete from is\_student

where sno=2001;//不能被删除

转换后的删除语句为

delete from student

where sno=2003 and sdept=’信工系’;//能被删除

//视图更新时注意事项

目前各个关系数据库一般只允许对行列子集视图进行更新。一般对行列子集视图都可以执行修改和删除操作，如果基本表中所有不允许空值的列都出现在视图中，则也可以对其执行插入操作。

不同系统对视图的更新有进一步的规定，不同系统实现方法的不同，具体规定也不同。

### 3.5.6 课外补充

无

# 第四章 数据库安全性

## 4.1 授课内容

授课科目：数据库原理

授课内容： 数据库安全性

授课类型：讲授+练习

授课时间：4学时

主讲教师：

## 4.2 教学目标要求

### 4.2.1 能力目标

1. 对数据库安全机制有一定程度的了解；
2. 能够灵活运用数据库安全机制一定程度上保护数据库的安全；
3. 能够使用数据库的授权机制合理为数据库使用者分配权限；
4. 数据库出现安全问题能快速准确地定位到问题的根源。

### 4.2.2 知识目标

1. 了解数据库安全包括哪些内容；
2. 掌握数据库安全模型；
3. 掌握用户身份识别、自主存取控制、授权和回收、数据库角色管理相关的理论和语法；
4. 了解视图机制对数据库安全的保护；
5. 了解审计、数据加密等数据库安全保护措施。

## 4.3 课件分析

概 述： 数据库管理系统能提供统一的数据保护功能来保证数据的安全可靠和正确有效是数据库的一大特点，本章围绕着数据库的安全性介绍了数据库安全的标准、数据库安全控制模型、以及视图机制有效保护数据库数据安全性的原理。在数据库安全模型中重点介绍用户的身份识别、存取控制、自主存取控制、授权与回收、数据库角色管理、强制存取控制等六个保障数据库安全的环节以及其中涉及到的SQL语法，CREATE LOGIN、CREATE USER、CREATE ROLE、GRANT、REVOKE等。

教学重点：数据库安全模型、身份识别、自主存取控制、授权、数据库角色管理。

教学难点：授权、数据库角色管理。

## 4.4 教学方法

问题教学法、讲授法。

## 4.5 教学过程

### 4.5.1 课前补充

（0分钟）

### 4.5.2 上节回顾

（0分钟）

### 4.5.3 作业点评

（0分钟）

### 4.5.4 导入新课

（3分钟）

在现实生活中，人们在使用数据库时经常会出现一些非法使用数据库的情况，比如：编写合法的程序绕过数据库管理系统及其授权机制，通过合法的一些操作得到数据推导出一些保密的数据，恶意损坏数据库等等，这些非法使用数据的情况有可能是无意的也有可能是恶意的，不管怎么，这些非法的使用都可能给数据库的安全带来隐患，所以了解数据库的安全机制，了解各DBMS对数据库安全的保障措施是从事数据库开发和使用人员的必修课。本章就围绕着SQLServer2008来讨论数据库的安全性问题和措施。

### 4.5.5 讲授新课

**4.1 数据库安全性概述**

数据库的一大特点是数据共享，数据共享必然导致数据库的安全性问题。数据库中的数据共享，必须在DBMS统一的严格的控制下，只允许有合法使用权限的用户访问允许他存取的数据。

数据库的安全性：是指保护数据库，防止因用户非法使用数据库造成数据泄密、更改或破坏。数据库系统的安全保护措施是否有效是数据库系统的主要性能指标之一。合法用户合法访问到机密数据后，能否对这些数据保密，不属于课程的讨论范围。属于法律、政策、伦理、道德方面的问题。

数据库的不安全因素主要包括以下几个方面：

1. 非授权的用户对数据库的恶意存取和破坏。

2. 数据库中重要或敏感的数据被泄露。

3. 安全环境的脆弱性。

计算机以及信息技术方面哟一系列的安全标准，最具影响力的是TCSEC（Trusted Computer System Evaluation Criteria）和CC（Common Criteria），TCSEC又称为桔皮书。TCSEC扩展到数据库管理系统定义了数据库管理系统的设计和实现中需要满足和用以进行安全性级别评估的标准以及等级如下表所示：

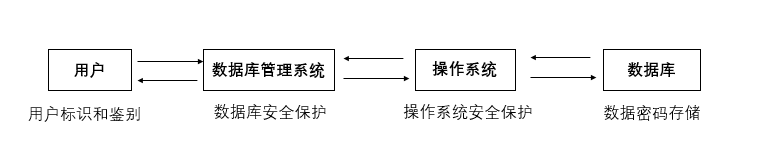
|  |  |
| --- | --- |
| 安全级别 | 定义 |
| A1 | 验证设计 |
| B3 | 安全域 |
| B2 | 结构化保护 |
| B1 | 标记安全保护 |
| C2 | 受控的存取保护 |
| C1 | 自主安全保护 |
| D | 最小保护 |

D级是最低级别；C1级只提供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；C2级是安全产品的最低档，提供受控的存取保护；B1级标记安全保护，对标记的主体和客体实施强制存取控制以及审计等安全机制；B2级结构化保护，建立形式化的拿权策略模型，并对系统内的所有主体和客体实施强制存取控制和自主存取控制；B3级安全域，必须满足访问监控器的要求，设计跟踪能力更强，并提供系统恢复过程；A1级验证设计，提供B3级的保护，同事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。

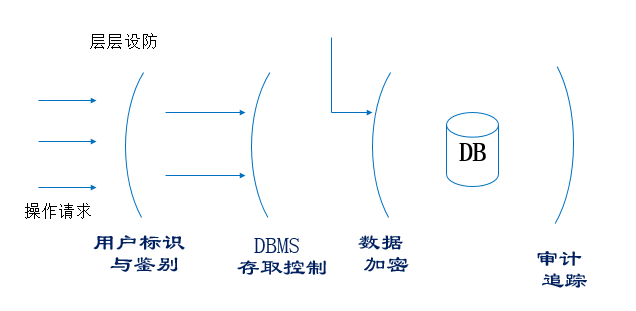
**4.2 数据库安全性控制**

在一般的计算机系统中，安全措施是一级一级层层设置的。

一种常用的计算机安全模型为：



数据库安全保护的控制层次如下图所示：



4.2.1 用户身份鉴别

用户身份鉴别是系统提供的最外层安全保护措施。只让合法用户进入系统。

系统提供一定的方式让用户标识自己的名字或身份。系统内部记录着所有用户的标识。每次用户要求进入系统时，由系统将用户提供的身份标识与系统内部记录的合法用户标识进行核对，通过鉴定后才提供机器使用权。为了进一步核实用户，在输入用户标识后，系统常常要求用户输入口令。口令通常以\*形式显示。为避免用户名与口令被别人窃取，可以使用更复杂的方法：如可采用如下方法，预先约定计算过程或函数，鉴定用户身份时系统提供一随机数，用户根据随机数及约定函数计算结果，作为密码。（随机数不同密码不同）

SQLServer2008数据库管理系统中创建合法用户包括两个步骤：

1. 创建登录名

创建登录名的基本语法：

CREATE LOGIN <登录名>

[ {

WITH PASSWORD = ‘’ [HASHED][MUSTCHANGE]

,DEFAULT\_DATABASE = <数据库>

} |

{

FROM

WINDOWS

[WITH DEFAULT\_DATABASE = <数据库>]

|CERTIFICATE <证书名>

|ASYMMETRIC KEY <不对称密钥名>

}

]

例：创建一个SQL Server验证模式的登录名

CREATE LOGIN zhangsan

WITH PASSWORD = ‘abc123!’

2. 创建用户

创建用户的基本语法：

CREATE USER <用户名>

[ {{FOR | FROM}

LOGIN <登录名>

| CERTFICATE <证书名>

| ASYMMETERIC KEY <密钥名>

}

[WITHOUT LOGIN]

[WITH DEFAULT\_SCHEMA = <架构名>]

]

例：创建具有默认架构的数据库用户

CREATE USER zhangsan FOR LOGIN zhangsan

WITH DEFAULT\_SCHEMA = student;

4.2.2 存取控制

存取控制机制主要包括定义用户权限和合法权限检查两部分。定义用户权限是将用户权限登记到数据字典中（用户对某一对象的操作权力称为权限）；合法权限检查是指每次用户每次发起存取数据库的操作请求后，数据库管理系统查找数据字典，根据安全规则进行合法权限检查，如果用户的操作请求超出了定义的权限，系统将拒绝执行此操作。

定义用户权限和合法权限检查机制合在一起组成了数据库管理系统的存取控制子系统。C2级的数据库管理系统支持自主存取控制（DAC），B1级的数据库管理系统支持强制存取控制（MAC）。

4.2.3 自主存取控制

用户权限由两个要素组成：数据对象和操作类型，定义一个用户的存取权限就是要定义这个用户可以在哪些数据库对象上进行哪些类型的操作。在数据库系统中定义存取权限称为授权（authorization）。

关系数据库中涉及到的数据对象和其上的操作类型之间的对应关系如下表所示：

|  |  |  |
| --- | --- | --- |
| 对象类型 | 对象 | 操作类型 |
| 数据库模式 | 模式 | CREATE SCHEMA |
| 基本表 | CREATE TABLE，ALTER TABLE |
| 视图 | CREATE VIEW |
| 索引 | CREATE INDEX |
| 数据 | 基本表和视图 | SELECT，INSERT，UPDATE，DELETE，REFERENCES，ALL PRIVILEGES |
| 属性列 | SELECT，INSERT，UPDATE，REFERENCES，ALL PRIVILEGES |

授权粒度：可以定义的数据对象范围。授权粒度越细，即可以定义的数据对象范围越小，授权子系统就越灵活，能够提供的安全性就越完善。另一方面数据字典变大变复杂，系统定义与检查权限的开销也会相应地增大。

关系数据库中，授权的数据对象粒度包括表、属性列、行（记录）。

例：一个授权表(授权粒度为表和列)

|  |  |  |
| --- | --- | --- |
| 用户名 | 数据对象名 | 允许的操作类型 |
| 张明  李青  李青  李青  李青  王楠  王楠 | 关系student  关系student  关系course  关系sc  列sc.grade  列sc.sno  列sc.cno  ……. | SELECT  ALL  ALL  select  UPDATE  SELECT  Select  ……. |

例：与数据值有关的授权（授权粒度为表、列、记录）

|  |  |  |  |
| --- | --- | --- | --- |
| 用户名 | 数据对象名 | 允许的操作类型 | 存取谓词 |
| 张明  李青  李青  李青  李青  王楠  王楠 | 关系student  关系student  关系course  关系sc  列sc.grade  列sc.sno  列sc.cno  ……. | SELECT  ALL  ALL  select  UPDATE  SELECT  Select  ……. | Sdept=’计算机’ |

数据库管理系统一般都提供了存取控制语句进行存取权限定义。SQL中使用GRANT和REVOKE语句向用户授权和收回对数据的操作权限。

1. GRANT的一般语法格式：

GRANT <权限>[,<权限>]...

[ON <对象类型> <对象名>]

TO <用户>[,<用户>]...

[WITH GRANT OPTION];

例：把查询Student表权限授给用户U1。

GRANT SELECT ON TABLE Student TO U1;

例：把对Student表和Course表的全部权限授予用户U2和U3;

GRANT ALL PRIVILEGES ON TABLE Student, Course TO U2, U3;

2. REVOKE的一般语法格式：

REVOKE <权限>[,<权限>]...

[ON <对象类型> <对象名>]

FROM <用户>[,<用户>]...;

例：把用户U4修改学生学号的权限收回。

REVOKE UPDATE(Sno) ON TABLE Student FROM U4;

例：把用户U5对SC表的INSERT权限收回。

REVOKE INSERT ON TABLE SC FROM U5 CASCADE;

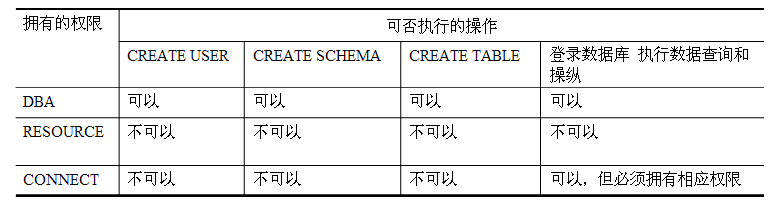
3. 创建数据库模式的权限

创建数据库模式一类的数据库对象的授权是由数据库管理员在创建用户时实现的。其一般的语法格式：

CREATE USER <username>

[WITH] [DBA | RESOURCE | CONNECT]

创建用户的命令中如果没有指定创建的新用户的权限，默认用户拥有CONNECT权限，拥有CONNECT权限的用户不能穿件新用户，不能创建模式，也不能创建基本表，只能登录数据库。下图展示了三类权限与所对应的的操作。



4.2.5 数据库角色

数据库角色是被命名的一组与数据库操作相关的权限，角色是权限的集合。

1. 创建角色的语法格式：CREATE ROLE <角色名>。

2. 给角色授权的语法格式：

GRANT <权限>［，<权限>］…

ON <对象类型>对象名

TO <角色>［，<角色>］…

3. 将一个角色授予其他的角色或用户：

GRANT<角色1>[,<角色2>]…

TO <角色3>[,<用户1>]…

[WITH ADMIN OPTION]

在SQLServer2008中使用存储过程将用户加入到某一角色：exec sp\_addrolemember r1,u1；

4. 角色权限的收回：

REVOKE <权限>［，<权限>］…

ON <对象类型> <对象名>

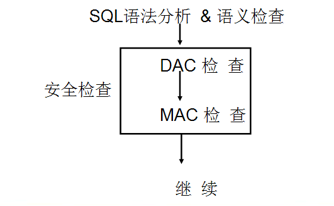
FROM <角色>［，<角色>］…

4.2.6 强制存取控制

强制存取控制是指系统为保证更高程度的安全性，按照TDI/TCSEC标准中安全策略的要求所采取的强制存取检查手段。用户不能直接感知或者进行强制存取控制。适合对数据有严格而固定密级分类的部门。比如军事、政府等。

强制存取控制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。

DAC与MAC共同构成了DBMS的安全机制，实现MAC首先要实现DAC。DAC+MAC的安全检查示意图如下图所示。



**4.3 视图机制**

定义视图：关系系统中，为不同用户定义不同的视图，把要保密的数据对无权存取这些数据的用户隐藏起来，从而自动地对数据提供一定程度的安全保护。

视图机制更重要的功能在于提供数据的独立性，其安全保护功能不太精细，远远达不到应用系统的要求。

实际应用中视图机制与授权机制配合使用，即由视图屏蔽掉部分保密数据，在视图上进一步定义存取权限。

**4.4 审计**

前述的安全性措施均为强制性措施，将用户的操作限制在安全性范围内。

审计追踪使用一个专用文件或数据库，系统自动将用户对数据库的所有操作记录在上面，利用审计追踪信息，就能重现导致数据库现有状况的一系列事件，以找出非法存取数据的人。（作为一种预防手段，检测可能的不法行为）

审计很费时间和空间，所以DBMS往往将其作为可选特征，允许DBA根据应用对安全性的要求，灵活地打开或关闭审计功能。审计功能主要用于安全性要求较高的部门。

**4.5 数据加密**

对于高度敏感的数据，可以采用数据加密技术，以密码的形式存储和传输数据。用户正常检索数据时，首先要提供密码钥匙，由系统进行译码后，才能得到可识别的数据。

数据加密与解密也是比较费时的操作，而且数据加密与解密程序会占用大量的系统资源，此功能通常也作为可选特征，允许用户自由选择，只对高度机密数据加密。

数据加密的基本思想是根据一定的算法将原始数据——明文变换为不可直接识别的格式——密文，从而使得不知道解密算法的人无法获知数据的内容。

数据加密主要包括存储加密和传输加密。

存储加密一般提供透明和非透明两种存储加密方式。透明加密是内核级加密保护方式，对用户完全透明；非透明加密则是通过多个加密函数实现。

传输加密是在客户/服务器结构中，数据库用户与服务器之间的信息传输采用加密处理。常用的传输加密方式是链路加密和端到端的加密。

**4.6 其他安全性保护**

推理控制：避免用户利用其能够访问数据推知更高密级的数据，是为了解决强制存取控制未解决的问题而提出的。

隐蔽信道：也是为了解决强制存取控制未解决的问题而提出的一种安全保护措施，比如解决利用未被强制存取控制的SQL执行后反馈的信息进行间接的信息传递。

数据隐私：是指控制不愿被他人知道或者他人不便知道的个人数据的能力。数据隐私已经成为数据库应用中心的数据保护模式。

### 4.5.6 课时小结

本章主要内容：

1. 数据库安全概述；

2. 数据库安全的一般模式；

3. 用户身份识别、自主存取控制、授权与回收、数据库角色管理、强制存取控制；

4. 视图机制、审计、数据加密、其他的安全保护。

### 4.5.7 课外补充

无

# 第五章 数据库完整性

## 5.1 授课内容

授课科目：数据库原理

授课内容：完整性

授课类型：讲授+练习

授课时间：2学时

主讲教师：

## 5.2 教学目标要求

### 5.2.1 能力目标

1. 掌握完整性的概念；
2. 掌握约束的定义；
3. 掌握完整性检查机制；
4. 掌握违背完整性约束条件时关系数据库管理系统应采取的动作等。

### 5.2.2 知识目标

1. 掌握用数据定义语言实现的完整性；
2. 掌握约束的定义；
3. 掌握完整性检查机制；
4. 掌握违背完整性约束条件时关系数据库管理系统应采取的动作等。

## 5.3 课件分析

概 述： 数据库完整性是为了保证数据库中存储的数据是正确的。所谓正确是指符合现实世界语义的。本章讲解了关系数据库管理系统完整性实现的机制，包括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关系数据库管理系统应采取的动作等。

在关系系统中，最重要的完整性约束是实体完整性和参照完整性，其他完整性约束条件则可以归入用户定义的完整性。

数据库完整性的定义一般由SQL的数据定义语言来实现。他们作为数据库模式的一部分存入数据字典，在数据库数据修改时关系数据库管理系统的完整性检查机制将按照数据字典中定义的这些约束进行检查。

完整性机制的实施会影响系统性能。因此，许多数据库管理系统对完整性机制的支持比对安全性的支持要晚得多。

教学重点：

1．数据定义语言来实现数据库完整性的定义

2．数据库完整性检查机制

教学难点：数据库完整性的定义

## 5.4 教学方法

问题教学法、讲授法。

## 5.5 教学过程

### 5.5.1 课前补充

（0分钟）

### 5.5.2 上节回顾

（0分钟）

### 5.5.3 作业点评

（0分钟）

### 5.5.4 导入新课

（3分钟）

[数据库完整性](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93%E5%AE%8C%E6%95%B4%E6%80%A7)（Database Integrity）是指数据库中数据在逻辑上的一致性、正确性、[有效性](https://baike.baidu.com/item/%E6%9C%89%E6%95%88%E6%80%A7)和相容性。数据库完整性由各种各样的完整性约束来保证，因此可以说数据库完整性设计就是数据库完整性约束的设计。数据库完整性约束可以通过[DBMS](https://baike.baidu.com/item/DBMS)或应用程序来实现，基于DBMS的完整性约束作为模式的一部分存入数据库中。

### 5.5.5 讲授新课

**5.1 数据库完整性性**

数据库的完整性是指数据的正确性和相容性。数据库是否完整性关系到数据库系统能否真实反映现实世界。

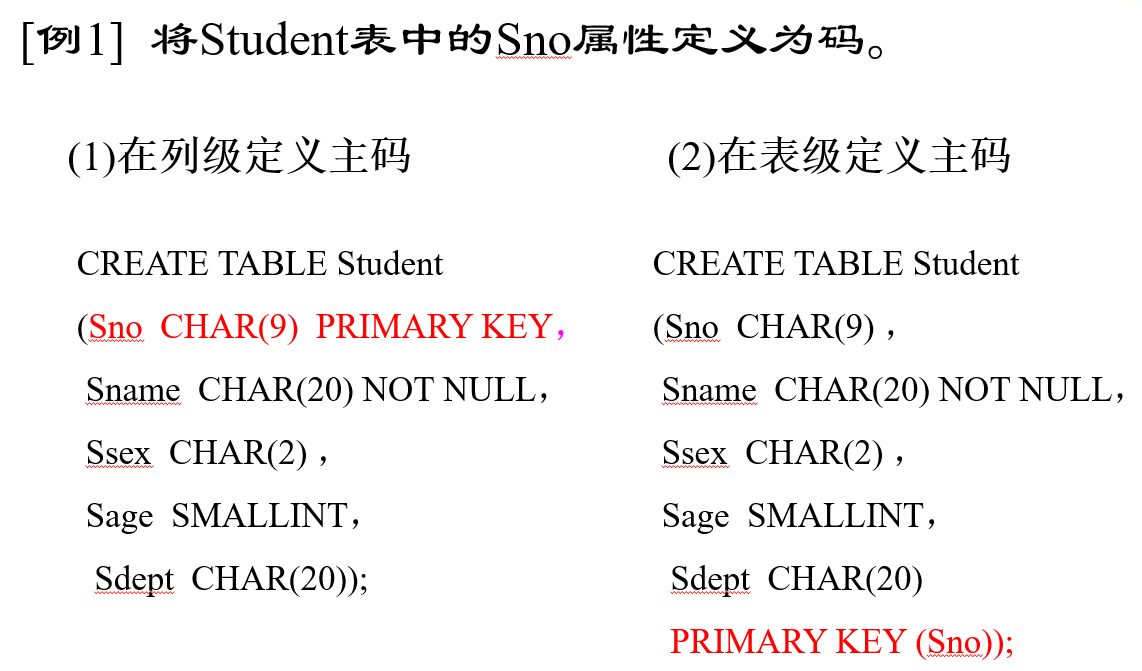
数据库的完整性约束条件：加上数据库数据之上的语义约束条件。

数据库中的数据是否正确有效相容，依据是是否满足完整性约束条件。

数据库安全性与数据库完整性间的联系与区别：安全性与完整性是数据库保护的两个不同方面。安全性是防止用户非法使用数据库，包括恶意破坏数据和越权存取数据。完整性则是防止合法用户使用数据库时向数据库中加入不合语义的数据。

**5.2 实体完整性**

实体完整性定义：关系模型的实体完整性。



CREATE TABLE中用PRIMARY KEY定义，单属性构成的码有两种说明方法：1.定义为列级约束条件；2.定义为表级约束条件。对多个属性构成的码只有一种说明方法：定义为表级约束条件。

实体完整性检查和违约处理：插入或对主码列进行更新操作时，RDBMS按照实体完整性规则自动进行检查。包括：

1. 检查主码值是否唯一，如果不唯一则拒绝插入或修改。

2. 检查主码的各个属性是否为空，只要有一个为空就拒绝插入或修改。

检查记录中主码值是否唯一的一种方法是进行全表扫描。

索引：为了避免全表扫描，一般会在主码上建立索引。

**5.3 参照完整性**

关系模型的参照完整性定义，在CREATE TABLE中用FOREIGN KEY短语定义哪些列为外码，用REFERENCES短语指明这些外码参照哪些表的主码 。

例如 关系SC中一个元组表示一个学生选修的某门课程的成绩，（Sno，Cno）是主码。Sno，Cno分别参照引用Student表的主码和Course表的主码。

CREATE TABLE SC

( Sno CHAR(9) NOT NULL，

Cno CHAR(4) NOT NULL，

Grade SMALLINT，

PRIMARY KEY (Sno， Cno)， /\*在表级定义实体完整性\*/

FOREIGN KEY (Sno) REFERENCES Student(Sno)，

FOREIGN KEY (Cno) REFERENCES Course(Cno)

);

参照完整性和违约处理



拒绝(NO ACTION)执行

默认策略

级联(CASCADE)操作，设置为空值（SET-NULL），对于参照完整性，除了应该定义外码，还应定义外码列是否允许空值。

**5.4 用户定义完整性**

属性上的约束条件的定义

CREATE TABLE时定义

列值非空（NOT NULL）

列值唯一（UNIQUE）

检查列值是否满足一个布尔表达式（CHECK）

属性上的约束条件检查和违约处理

插入元组或修改属性的值时，RDBMS检查属性上的约束条件是否被满足

如果不满足则操作被拒绝执行

元组上的约束条件的定义

在CREATE TABLE时可以用CHECK短语定义元组上的约束条件，即元组级的限制

同属性值限制相比，元组级的限制可以设置不同属性之间的取值的相互约束条件

元组上的约束条件检查和违约处理

插入元组或修改属性的值时，RDBMS检查元组上的约束条件是否被满足

如果不满足则操作被拒绝执行

**5.4 完整性约束命名子句**

完整性约束命名子句：CONSTRAINT 约束。

基本语法如下：

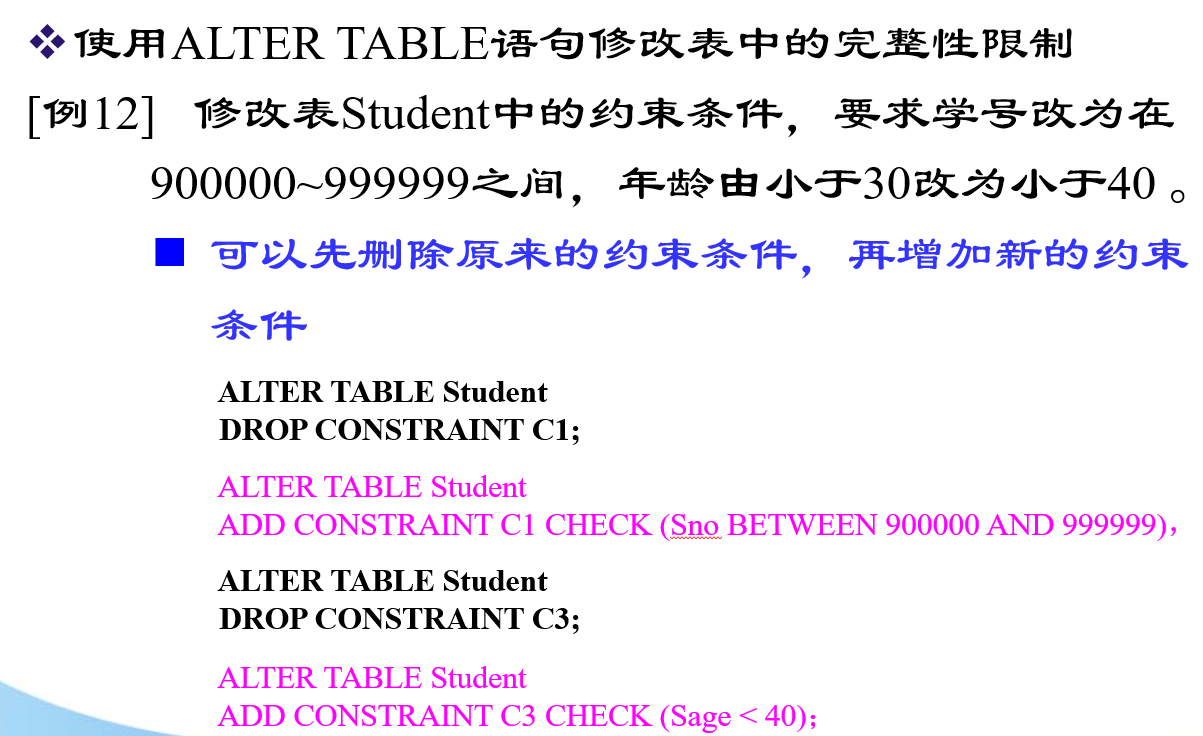
CONSTRAINT <完整性约束条件名>

［PRIMARY KEY短语

|FOREIGN KEY短语

|CHECK短语］

修改表中的完整性限制



### 5.5.6 课时小结

本章主要内容：

1. 数据库完整性概念；

2. 数据库完整性约束的定义以及检查；

3. 违背完整性约束条件时关系数据库管理系统采取的动作。

### 5.5.7 课外补充

无

# 第六章 关系数据理论

## 6.1 授课内容

授课科目：数据库原理

授课内容：关系数据理论

授课类型：讲授

授课时间：4学时

主讲教师：

## 6.2 教学目标要求

### 6.2.1 能力目标

1. 能够正确判断当前关系数据库产品是否有冗余和异常；
2. 能够依据关系数据理论对关系数据模式进行分解；
3. 能够熟练掌握数据依赖的公理系统；
4. 了解模式分解的理论和实际意义。

### 6.2.2 知识目标

1. 掌握函数依赖的概念；
2. 掌握范式的概念；
3. 掌握Amstrong公理系统；
4. 掌握闭包的概念，掌握利用闭包求关系模式候选码的算法；
5. 熟悉模式分解算法，熟悉最小依赖集概念。

## 6.3 课件分析

概 述： 本章内容为关系数据理论，从一个关系数据模型的例子引入，分析其是否有插入异常、删除异常、更新异常和数据冗余过大等问题，通过分析这些问题出现的原因，引入函数依赖的概念，通过五种函数依赖，对其进行模式分解，一步步解决以上异常和冗余的问题，期间重点讲解范式的概念；又讲解了数据依赖的公理系统，为以上的模式分解提供必要的理论支撑；最后讲解闭包的概念，并通过属性相对于函数依赖的闭包，来求得关系模式的候选关键字。

教学重点：函数的五种依赖，范式的概念，1NF、2NF、3NF，Amstrong公理，属性相对于函数依赖的闭包，求候选关键字。

教学难点：Amstrong公理系统，属性闭包的算法。

## 6.4 教学方法

讲授法。

## 6.5 教学过程

### 6.5.1 课前补充

（0分钟）

### 6.5.2 上节回顾

（5分钟）

第一章中部分概念回顾：什么是候选码、主码、主属性、非主属性等；关系模式的定义，R（U，D，DOM，F），各部分的含义。

### 6.5.3 作业点评

（0分钟）

### 6.5.4 导入新课

（5分钟）

通过前五章的学习，对于关系数据库的基本结构、相应的操作、以及使用规则都有了明确的认识。通过SQL语言也能够熟练的建立数据库、数据表、并对已有数据方便的进行增删改查。但是数据库在创建的过程中，是否符合相应规范，是否有相应的理论能够使我们建立的数据模型科学合理，在之前的课程中并没有提及，本章将讨论关系数据理论，在规范创建关系数据库的同时，对之前所学内容有一个更深入的认识。

### 6.5.5 讲授新课

**6.1 问题的提出**

[例1]建立一个描述学校教务的数据库：

学生的学号（Sno）、所在系（Sdept）、系主任姓名（Mname）、课程号（Cno）、 成绩（Grade）

单一的关系模式 ： Student <U、F>， U ＝｛Sno, Sdept, Mname, Cno, Grade｝

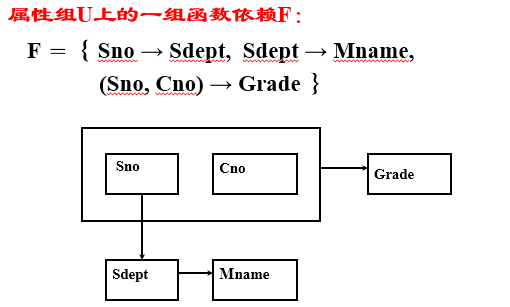
学校数据库的语义：

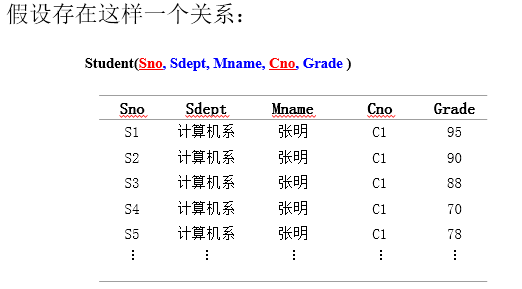
1. 一个系有若干学生， 一个学生只属于一个系；

2.一个系只有一名主任；

3.一个学生可以选修多门课程， 每门课程有若干学生选修；

4.每个学生所学的每门课程都有一个成绩。





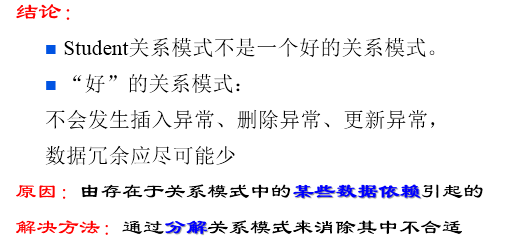
请问：该关系模式好吗？

1.系名、系主任名重复出现 数据冗余太大

2.“张明”退休，李四接替 更新异常

3.一个新系刚成立，尚无学生 插入异常

4.一个系的学生全部毕业 删除异常



**6.2 函数依赖**

函数依赖是关系模式内属性间最常见的一种依赖关系。

1.关系中属性间函数依赖举例

学生关系属性的集合U={Sno,Sdept,Mname,Cname,Grade}

现实世界的已知事实：

1）一个学生只属于一个系（含义：学号确定后，学生所在的系也就被唯一确定了。类似于数学中的函数y=f(x)，x=Sno,y=Sdept。我们说Sno函数决定Sdept,或者说,Sdept函数依赖于Sno, 记做sno→sdept；反过来一个系有若干学生（解释系不能决定学生）

2）一个系只有一名系主任（含义：系确定后系主任名被唯一确定了，类似于数学中的函数，我们说Sdept函数决定Mname,或者说,Mname函数依赖于Sdept。记做Sdept→Mname）

3）一个学生可以选修多门课，每门课程有若干学生选修（含义：学号与课程号间够不成决定被决定关系）

4）每个学生所学的每门课程都有一个成绩，含义：(sno,cname) →Grade

属性组U上的一组函数依赖F={ sno→sdept, Sdept→Mname,(sno,cname) →Grade } 2.函数依赖定义

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，他们在X上的属性值相同，而在Y上属性值不同，则称“X函数决定Y”或“Y函数依赖于X”,记作X→Y

思考题：

1）码是否一定函数决定非码属性？非码属性间是否存在函数依赖

2）SNo→Sdept成立 (Sno,Cno) →Sdept是否也成立？

3）找出以下几个表的中的函数依赖

SCG（Sno,Sname,Ssex,Sage,Sdept，Cno,Cname,Cpno,Ccredit,Grade）

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

课程表Course(Cno,Cname,Cpno,Ccredit)

学生选课SC(Sno,Cname,Grade)

函数依赖的几点说明：

函数依赖是所有关系实例(关系模式值的每一状态)均要满足的约束条件。

1） 函数依赖是语义范畴。只能根据语义确定函数依赖。如：在没有同名的情况下“姓名→年龄”成立

2） 数据库设计者可以对现实世界作强制的规定。

3） 若X→Y，则X称为这个函数依赖的决定属性集

4） 若X→Y,Y→X则记为 X Y

5） 若Y不函数依赖于X，则记为X→Y

3.平凡函数依赖与非平凡函数依赖定义

在关系模式R(U)中，对于U的子集X和Y，如果X→Y，但Y不是X的子集，则称X→Y是非平凡函数依赖。若Y是X的子集，则称X→Y为平凡函数依赖。

对于任一关系模式，平凡函数依赖都是必然成立的，它不反映新的语义。

4.完全函数依赖与部分函数依赖

在关系模式R(U)中，如果X→Y, 并且对于X的任何一个真子集X'都有X'→Y，则称Y完全函数依赖于X，记作X→Y。若X→Y，但Y不完全函数依赖于X，则称Y部分函数依赖于X，记X→Y。

F

P

5.传递函数依赖

在关系模式R(U)中，如果X→Y，Y→Z，且Y不是X的子集，X不函数依赖于Y，则称Z传递函数依赖于X。

Std(Sno,Sdept,Mname) 有Sno→Sdept, Sdept→Mname，Mname传递函数依赖于 Sno

6.码

设K为关系模式R(U，F)中属性或属性组。若U完全依赖于K，则K称为R的一个侯选码。若关系模式中有多个侯选码，则选定一个作为主码。

**6.3 范式**

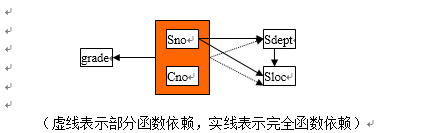
1NF：关系模式R的所有属性都是不可分割的基本数据项，则R∈1NF,(非主属性函数依赖于码)

不满足1NF的举例：学生(学号，姓名，年龄，**入学毕业年月**)

1NF是关系模式的起码要求

第一范式举例：

SLC(Sno,Sdept,Sloc,Cno,Grade)



属性函数依赖情况：

插入异常：未选课的学生不能插入，因为码值部分为空（原因是对码的部分函数依赖造成）。（解决问题的办法：把部分函数以来部分分解出来）。

删除异常：某一学生的选课信息全部删除后，学生的其他信息也被删除，否则码值部分为空（不允许）（原因是对码的部分函数依赖造成）。（解决问题的办法：把部分函数以来部分分解出来）。

数据冗余度大：一个学生选修了多门课程，同一学生的基本信息（Ssept,Sloc）需要保存多次。（原因是对码的部分函数依赖造成）。（解决问题的办法：把部分函数以来部分分解出来）。

修改复杂：修改某学生基本信息时，如果其选修多门课程，同一信息需要重复修改多次。（原因是对码的部分函数依赖造成）（解决问题的办法：把部分函数以来部分分解出来）。

2NF：满足第一范式，非主属性完全函数依赖于码。

前面的关系模式不满足第二范式，分解为下面两个模式后，部分函数依赖被消除：

SC(sno,cno,grade)

SL(Sno,Sdept,Sloc)

分解成第二范式后SC(sno,cno,grade)，SL(Sno,Sdept,Sloc)。以上4个问题在一定程度上得到了解决。

1）SL关系中可以插入尚未选修的学生

2）删除学生选课关系，只涉及SC关系，不涉及SL中的学生基本信息。

3）由于学生选课信息于学生基本信息分开存放，不论该学生选修了几门课，Sdept与Sloc值都只存储一次。

4）某学生转系，只需要修改相关关系中的一个元组。

第二范式SL中存在传递函数依赖，使得SL仍然存在上述4个问题。

1）插入异常：系刚成立，无在校学生，无法存入系信息。（解决问题的办法：把与主码无关的函数部分分解出来独立构成关系模式，即消除对主码的传递函数依赖）

2）删除异常：某系的全部学生毕业了，系的信息也丢失了。（解决问题的办法：把与主码无关的函数部分分解出来独立构成关系模式，即消除对主码的传递函数依赖）

3）数据冗余度大：每一个系的学生都住在同一个地方，关于系的住处信息却要重复出现。

4）修改复杂：当某系学生调整住处时，需要修改所有学生的Sloc属性值。

造成上述4个问题的主要原因：非主属性传递函数依赖于主码。

解决办法：将传递函数依赖关系分解出来。

3NF：满足第2范式，且主属性既不部分函数依赖于码，也不传递函数依赖于码

SL(Sno,Sdept,Sloc) 分解为

SD(sno,sdept)

DL(Sdept,Sloc)

规范化过程：

规范化过程即为关系模式的不断分解过程。

规范化的实质：概念的单一化。

规范化的目的：

不断解决关系插入异常、删除异常、数据冗余度大、修改复杂问题。

3NF是工程上的标准，在工程应用上，达到第三范式一般情况下就可以了

**6.4 关系模式的规范化**

1.第一范式到第三范式的分解过程：分解关系模式消除非主属性对码的部分函数依赖，分解关系模式消除非主属性对码的传递函数依赖。

2.关系模式的分解方法并不唯一，只有能够保证分解后的关系模式与原关系模式等价的方法才有意义。

1）分解具有无损连接性：分解后不能丢失信息，即分解前的关系与分解后关系的自然连接结果相等。

2）分解后应能保持函数依赖。分解前的函数依赖F被分解后关系模式函数依赖Fi所逻辑蕴含

3）分解既保持函数依赖，又保持无损连接。

3.第3范式是工业标准

若要求分解既具有无损连接，又保持函数依赖，那么模式分解一定能达到3NF，但不一定能达到BCNF。

习题：

第3题

学生（学号，姓名，出生年月，系名，班号，宿舍区）

班级（班号，专业名，系名，人数，入校年份）

系（系名，系号，系办公室地址，人数）

学会（学会名，成立年份，地点，人数）

学生参加学会（学号，学会名，入会年份）

只考虑与1．2．3NF有关的函数依赖（非主属性部分或完全函数依赖于码，非主属性传递函数依赖于码）

学生（学号，姓名，出生年月，系名，班号，宿舍区）

分解为：学生（学号，姓名，出生年月，班号，系） 系宿舍（系名。宿舍区）

班级（班号，专业名，系名，人数，入校年份）

分解为：班级（班号，专业名，人数，入校年份） 专业系（专业名，系名）

系（系号，系名，系办公室地址，人数）

学会（学会名，成立年份，地点，人数）

学生参加学会（学号，学会，入会年份）

### 6.5.6 课时小结

本章主要内容：

1. 函数依赖；

2. 范式；

3. Amstrong公理系统；

4. 属性相对于函数依赖集的闭包；

5. 模式分解。

### 6.5.7 课外补充

无

# 第七章 数据库设计

## 7.1 授课内容

授课科目：数据库原理

授课内容： 数据库设计

授课类型：讲授+练习

授课时间：4学时

主讲教师：

## 7.2 教学目标要求

### 7.2.1 能力目标

1. 掌握数据库设计的基本流程；
2. 掌握数据库E\_R模型的绘制；
3. 掌握E\_R模型向关系型数据库模型的转换。

### 7.2.2 知识目标

1. 需求分析的过程、方法及结果表达；
2. 概念结构设计的方法与步骤；
3. 逻辑结构设计（E\_R图向数据模型的转换原则）；
4. 物理结构设计的主要工作和过程。

## 7.3 课件分析

概 述： 本章主要介绍基于关系型数据库管理系统的数据库设计的技术和方法。广义地讲数据库的设计是指数据库及其应用系统的设计；狭义地讲，是设计数据库本身，即设计数据库的各级模式并建立数据库。狭义的数据库设计的基本步骤包括：需求分析、概念结构设计、逻辑结构设计、物理结构设计、数据库实施、数据库运行和维护六个阶段。其中的需求分析和概念设计可以独立于任何数据库管理系统开展，而逻辑结构和物理结构的设计与选择的数据库管理系统密切相关。本章重点介绍六个阶段中的概念结构设计和逻辑结构设计的方法和工具。

教学重点：概念结构设计、逻辑结构设计。

教学难点： 概念模型向逻辑模型的转换。

## 7.4 教学方法

问题教学法、讲授法。

## 7.5 教学过程

### 7.5.1 课前补充

（0分钟）

### 7.5.2 上节回顾

（0分钟）

### 7.5.3 作业点评

（0分钟）

### 7.5.4 导入新课

（3分钟）

数据库课程的学习就是为了今后的使用，在数据库产品的使用过程中，数据库的设计是很重要的一部分内容，在数据库领域内，通常把使用数据库的各类信息系统称为数据库应用系统。比如各类信息管理系统、办公自动化系统等。这些系统的基础都是数据库，这些系统的数据库都是如何设计并维护的呢？本节课我们就来深入学习数据的设计和一系列的数据库工作内容。

### 7.5.5 讲授新课

**7.1 数据库设计简介**

数据库设计的定义：数据库设计是指对于一个给定的应用环境，构造优化的数据库逻辑模式和物理结构，并据此建立数据库及其应用系统，使之能够有效地存储和管理数据，满足各种用户的应用需求，包括信息管理要求和数据操作要求。狭义地讲数据库设计是指数据库本身的设计，即数据库各级模式的设计和数据库的建立。

狭义的数据库设计的主要步骤：

1.需求分析：准确了解与分析用户需求（包括数据与处理）。

是最困难、最耗时的一步。作为地基的需求分析是否做得充分与准确，决定了在其上构建数据库大厦的速度与质量。做得不好，甚至会导致整个数据库设计返工重做。

2.概念结构设计阶段：

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.逻辑结构设计阶段：

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.数据库物理设计阶段

为逻辑数据模型选取一个最合适的应用环境的物理结构（包括存储结构和存取方法）

5.数据库实施阶段

设计人员运用DBMS提供的数据语言及其宿主语言，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.数据库运行和维护阶段

数据库应用系统经过试运行后即可投入正式运行。运行过程中必须不断地对其进行评价、调整与修改。

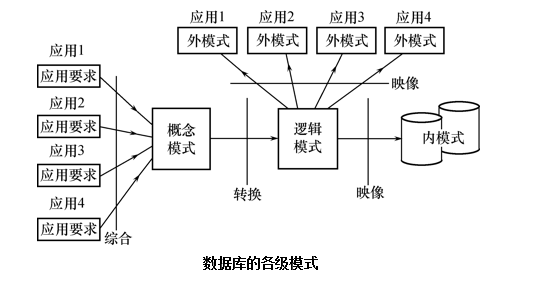
设计一个完善的数据库应用系统是不可能一蹴而就的，往往是上述6个阶段的不断反复过程。

数据库设计的特点：

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。数据库建设的基本规律是“三分技术，七分管理，十二分基础数据”，在数据库的建设中技术固然重要，但是更重要的是管理，不仅仅包括数据库建设本身作为一个项目的管理还包括对企业的业务管理。企业的业务管理更加复杂，也更加重要，对数据库结构的设计有直接影响。十二分基础数据则是强调数据的收集、整理、组织和不断更新也是数据库建设中的一个重要环节。

数据库设计设计到的各方面的知识和技术包括：计算机基础知识、软件工程的原理和方法、程序设计的方法和技巧、数据库的额基本知识、数据库设计技术、应用领域的知识。

数据库设计的各级模式：



**7.2 需求分析**

**1. 需求分析的过程**

(1) 调查组织机构总体情况：调查这个组织由哪些部门组成，各部门的职责是什么等，为分析信息流程做准备。

(2) 熟悉业务活动情况：调查各部门输入和使用的数据，数据的加工和处理，输出信息，输部门，输出的结果格式等。是调查的重点。

(3) 明确用户需求：在熟悉业务活动的基础上，协助用户明确对新系统的各种要求，包括信息要求、处理要求、安全性与完整性要求。调查重点。

(4) 确定系统边界：对调查的结果进行初步分析，确定整个系统中，哪些由计算机完成，哪些将来由计算机完成，哪些由手工完成。由计算机完成的功能就是新系统应该实现的功能。

需求分析任务（上述4步概括，也可直接用上述4点回答）：通过详细调查现实世界要处理的对象，充分了解原系统（手工系统或计算机系统）的工作概况，明确用户的各种需求，然后在此基础上确定新系统的功能。新系统必须充分考虑今后可能的扩充和改变不能仅仅按当前应用的需求来设计数据库。其重点是调查、收集与分析用户在数据管理中的信息要求、处理要求、安全性与完整性要求。

用户的信息要求：用户要从数据库中得到哪些信息，这些信息的具体内容和性质，从中确定数据库中应存储哪些数据。

用户的处理要求：用户要完成什么样的处理功能，对某种处理要求的响应时间，涉及的数据，处理方式是联机还是批处理。

**2. 调查方法**

跟班作业：通过亲生参加业务工作来了解业务活动的情况。此法可以比较准确理解用户的需求，但比较耗费时间。

开调查会：通过与用户座谈来了解业务活动情况及用户需求。座谈时，参加者之间可以相互启发。

请专人介绍：

询问：对某些调查中的问题，可以找专人询问。

设计调查表请用户填：如果调查表设计得合理，此方法很有效，也易于为用户接受。

查阅记录：查阅与原系统有关的数据记录

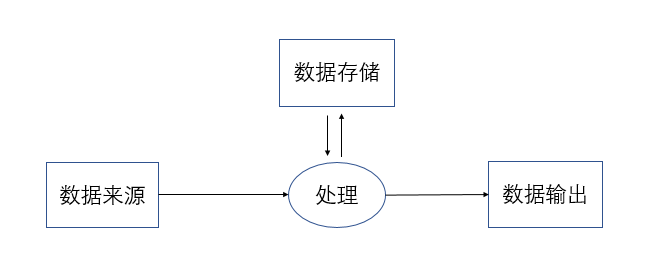
**3. 用户和设计人员对设计工作的最后结果共同承担责任**

让用户积极参与和配合调查，设计人员应该和用户取得共同的语言，帮助不熟悉计算机的用户建立数据库环境下的共同概念

在众多分析和表达用户需求的方法中，结构化分析方法（structured analysis ,SA方法）是一种最为简单实用的方法。

SA方法用自顶向下，逐步分解的方式分析系统，用数据流图、数据字典描述系统。

即设计人员首先需要把任何一个系统抽象为下图形式，再



将处理功能的具体内容分解为若干子功能，在把每个字功能继续分解，直到把系统的工作过程表达清楚为止。在处理功能分解的同时，他们所用的数据也逐级分解，形成若干层次的数据流图。

数据流图表示数据与处理间的关系。数据字典则详尽描述系统中的数据。对数据库设计来说，数据字典是进行详细的数据收集和数据分析所获得的主要结果。在数据字典中的内容在数据库设计过程中还要不断修改、充实、完善。

**4. 需求分析举例**

例：学校管理系统，经可行性分析和初步需求调查，抽取出该系统的最高层数据流图P188，共3个子系统教师管理子系统，学生管理子系统，后勤管理子系统。每个子系统分配一个开发小组。

学生管理子系统包括学籍管理和课程管理。其数据流图见P188

**5. 数据字典**

数据字典通常由数据项、数据结构、数据流、数据存储和处理过程组成。

(1) 数据项：不可分割的数据单位

数据项描述={数据项名，数据项的含义说明，别名，数据类型，长度，取值范围，取值含义，与其他数据项的逻辑关系}

例：库存数量范围、含义

(2) 数据结构

数据结构描述={数据结构名，含义说明，组成：{数据项或数据结构}}

(3) 数据流

可以是数据项，但更一般的情况是数据结构。表示某一处理过程的输入或输出数据。

数据流描述={数据流名，说明，数据流来源，数据流去向，组成：{数据结构}，平均流量，高峰期流量}

平均流量：单位时间（每天、每周、每月等）里的传输次数。高峰期流量：高峰时期的数据流量。

(4) 数据存储：处理过程中要存储的数据

数据存储描述={数据存储名，说明，输入数据流，输出数据流，数据量（每次存储多少数据），存取频度（每天或每小时或每月存取几次），存取方式（批处理还是联系处理，是检索还是更新，是顺序存取还是随机存取）}

(5) 处理过程：数据字典中只描述处理过程的说明性信息。具体处理逻辑一般用判定表与判定树来描述。

处理过程描述={处理过程名，说明，输入：{输入流}，输出：{输出流}，处理：{简要说明处理过程的功能及处理要求}}

说明：数据字典以能将数据描述清楚为度。

**7.3概念结构设计**

**1. 最常用的方法**

自底向上设计概念结构法，通常分两部：抽象数据并设计局部视图，集成局部视图

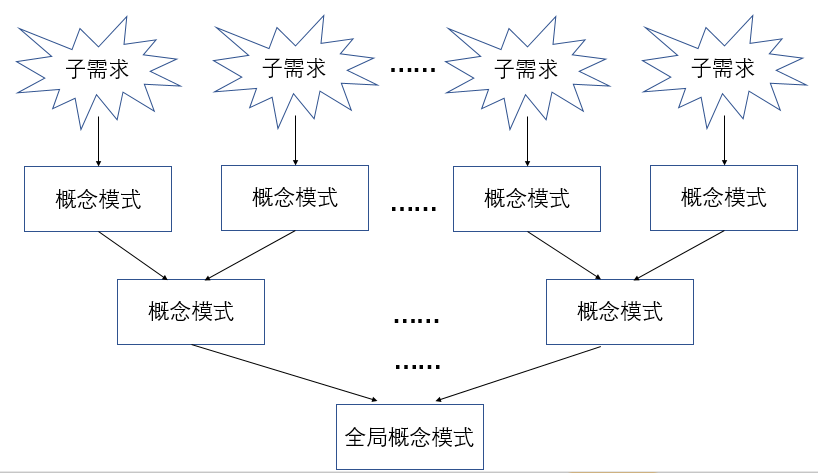


图 概念结构设计策略

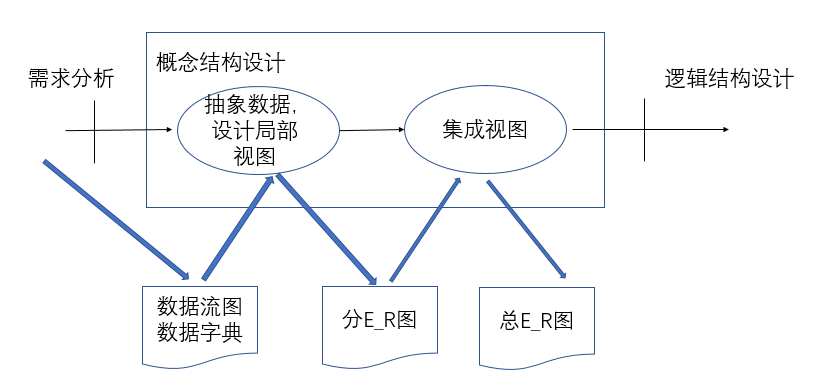


图 概念结构设计

**2. 数据抽象与局部视图设计**

选择局部应用

根据系统的具体情况，在多层数据流图中选择一个适当层次的数据流图，让这组图中每一部分对应局部应用，设计分E­­—R图。

往往以中层数据流图作为设计分E\_R图的依据，因它较好反映系统中各局部应用子系统的组成。如果局部应用比较复杂，可以从更下层的数据流图入手。

从图6-5 图6-6（a）入手设计学生管理子系统的分E-R图

逐一设计分E\_R图

\*\*将收集在数据字典中局部应用所涉及的数据抽取出来，参照数据流图标识局部应用中的实体、实体属性、标识实体的码，确定实体间的联系及其类型（１:１，１:n，1:m）。

实体抽象：将一组具有某些共同特性和行为的对象抽象为一个实体。对象与实体间是“is member of ”关系。

属性抽象：对象类型的组成成分可以抽象为实体的属性。组成成分与对象类型间的关系是“is part of”关系

\*\*有时实体与属性之间很难有截然划分的界限，同一事物，在一种应用环境中作为“属性”，在另一种应用环境中就必须作为实体。

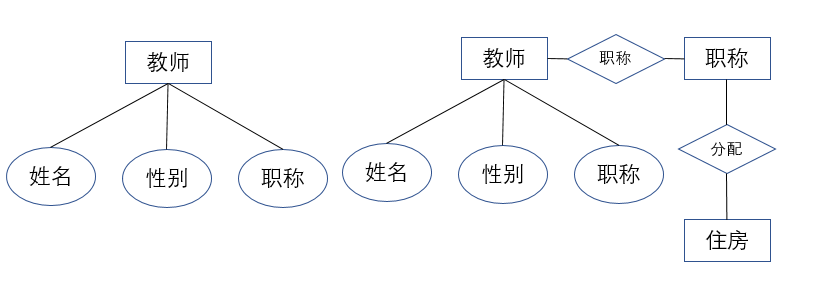
例：学校的系。有些环境下只作为属性描述，而在另一些环境中作为实体描述

\*\*确定属性准则（考虑到）：

属性不能再具有需要描述的性质。即属性必须是不可分的数据项，不能再由另外一些属性组成。

属性不能与其他实体有联系。联系只发生在实体间。

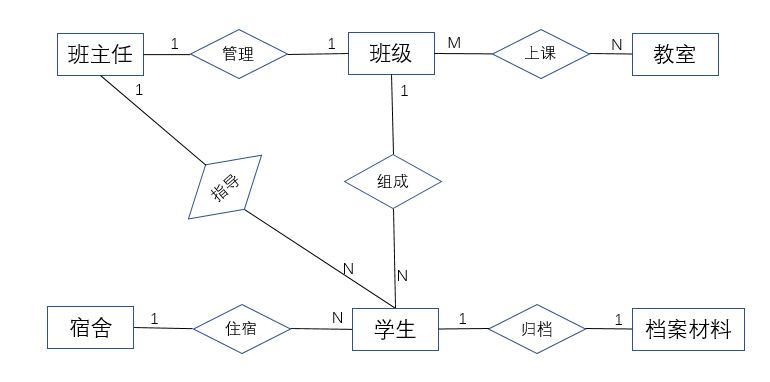
例子：职称



为了简化E\_R图的处理，现实世界中的事物凡能够作为属性的，应尽量作为属性。

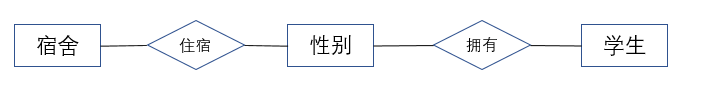
例：设计学籍管理局部应用的分E-R图，可用相同方法设计其他局部应用的分E-R图

\*\*学籍管理局部应用的分E-R图草图



\*\*学籍管理局部应用的分E-R图草图调整，得到分E-R图

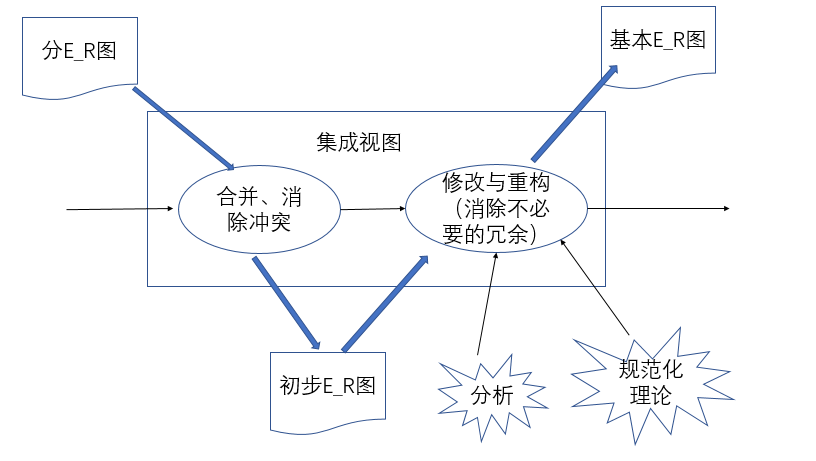
性别与宿舍分配有关，根据属性准则2应调整为实体（是否描述全）。



数据存储“学生登记表”，有用部分已经转入学生档案中，所以不必作为实体了（是否重复描述）。

\*\*学籍管理局部应用的分E-R得到分E-R图的所有实体属性P195，

**3. 视图集成**



合并分E\_R图，初步生成E\_R图：

1) 消除冲突。各分E\_R图之间的冲突主要有三类：属性冲突，命名冲突，结构冲突。

属性冲突包括：属性域冲突，例：学号类型不同分E\_R图中分别被说明为整形或字符型；属性的取值单位冲突。

命名冲突包括：同名异义、异名同义。命名冲突在实体、联系和属性上都可能发生。其中属性命名冲突更为常见。通过讨论、协商等行政手段加以解决。

结构冲突，同一对象在不同的应用中具有不同的抽象。如课程在某一局部应用中被当作实体，而在另一局部应用中被当作属性。用属性准则加以统一。

同一实体在不同应用局部应用中所包含的属性不完全相同，或属性的排列次序不同。解决方法为取属性的并集。

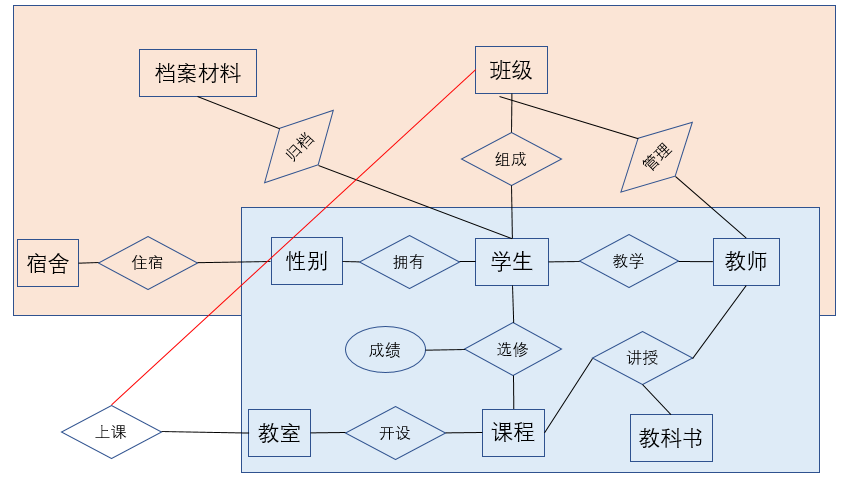
实体间的联系在不同的局部应用中呈现不同的类型。

解决方法：根据应用的语义对实体联系的类型进行综合或调整

例：学生管理子系统中学籍管理与课程管理局部视图分E\_R图存在的冲突。

学籍管理中的“班主任”与课程管理的“教师”在一定程度上属于异名同义。统一为 教师（职工号，姓名，性别，职称，是否为优秀班主任），班主任改为教师后，将两种联系（指导与教学）也综合为教学联系，性别在两个局部应用中具有不同的抽象。学籍管理中为实体，课程管理中为属性。根据属性准则进行合并。学生实体的属性的组成与次序在两个不同的分E-R图中都存在差异，应将所有属性综合，并重新调整次序。

解决上述冲突后得到的学生管理子系统的初步E-R图为：（P198 图6-16）



2）修改与重购，生成基本Ｅ-Ｒ图

目的：消除冗余的数据和冗余的实体间的联系（冗余容易破坏数据库的完整性，给数据维护增加困难）

如：

工资单（基本工资，各种补贴，应扣房租水电，实发工资）

实发工资 = 基本工资 + 各种补贴 – 应扣房租水电 （在数据字典中说明）

学生（学号，姓名，出生年月，年龄，所在系，年级，平均成绩）

年龄 = 当前年份 – 出生年月

平均成绩由学生选课联系中的成绩属性推算出

上课联系可由其他联系推算出

冗余的消除方法：主要为分析法，其分析依据是用数据字典中关于数据项之间逻辑关系的说明来消除冗余。

并不是所有冗余数据与冗余联系都必须加以消除：为了提高某些应用效率，不得不以冗余信息作为代价。如需要经常查询学生的平均成绩，每次读都需要计算效率就太底，保留该冗余数据能提高效率。（重点）

冗余数据的一致性维护：触发器。任何一科成绩修改或学生学了新的科目并有了成绩后，就触发该触发器去修改该学生的平均成绩属性值。（重点）

**7.4 逻辑结构设计**

主要工作：1．E\_R图向数据模型转换；2．数据模型优化；3．设计用户子模式

**1. E\_R图向数据模型转换**

\*将ER图转换为关系模型：将实体、实体属性、和实体间的联系转换为关系模式

\*转换的一般原则：

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。

一个m:n联系转换为一个关系模式。属性：由与该联系相连的实体码及联系本身的属性组成。码与该联系相连的实体码的组合。如学生选课联系。

一个1:n联系可以转换为独立的关系模式，也可以与n端对应的关系模式合并。如学生“组成”班级联系，可以转换成：

组成（学号，班级号）（独立的关系模式，码为n端实体的码）

或

学生（学号，姓名，出生年月，所在系，年级，班级号，平均成绩）（与多端关系模式合并，）

两种表示方法达到同样的目的：学生由组成班级情况。但后一种情况能减少系统中表的个数，更常用。

1：1联系可以转换为独立的关系模式，也可以与任意一端对应的关系模式合并。

例：教师“管理”班级联系（反映了班主任与班级的对应关系），可以转换为

独立的关系模式

管理（职工号，班级号） 或 管理（职工号，班级号）

与任一端合并

班级（班级号，学生人数，职工号）

或 教师（职工号，姓名，性别，职称，班级号，是否优秀班主任）

注：基于效率考虑，有时联系与某一端合并效率更高。如要经常查询某个班级的班主任名，则管理联系与教师关系合并更好些。原因是能减少连接操作。

三个或三个以上实体间的联系转换为一个关系模式。与该多元联系相连的各实体的吗以及联系本身的属性均转换为关系的属性。关系的码为实体码的组合。

如：授课（课程号，职工号，书号）

自联系即同一实体集的实体间联系，也按上述方法处理。

具有相同码的关系模式可合并：两个关系模式具有相同的主码，可以考虑将他们合并为一个关系模式。

例：拥有（学号，性别）

学生（学号，姓名，出生年月，所在系，年级，班级号，平均成绩）

合并为：学生（学号，姓名，性别，出生年月，所在系，年级，班级号，平均成绩）

例：依照上述的7个转换规则，学生管理子系统中的18个实体和联系可以转换为下列关系模型

实体（9个）：有档案材料，班级，宿舍，性别，学生，教师，教室，课程，教科书

联系（9个）：归档，组成，管理，住宿，拥有，教学，选修，讲授，开设

实体：

有档案材料（档案号，…..）

班级（班级号，学生人数）

宿舍（宿舍编号，地址，人数）4性别

性别（性别） 考虑书上属性“宿舍楼”是否显多余？

学生（学号，姓名，出生年月，所在系，年级，平均成绩）1档案号，2班级号，5性别，

教师（职工号，姓名，性别，职称，是否为优秀班主任）3班级号

教室（教师编码，地址，容量）

课程（课程，课程名，学分）9教室号

教科书（书号，书名，价钱）

联系：（其中只有6．7．8三个需要独立关系模式描述）

归档：1：1归并到 学生实体（档案号）

组成：学生“组成”班级n：1，归并到“学生”（班级号）

管理：教师“管理”班级1：1，归并到“教师”（班级号）

住宿：性别“住宿”宿舍1：n，归并到“宿舍”（性别）

拥有：学生“拥有”性别n：1，归并到“学生”（性别）

教学：学生与教师间的关系m：n，独立关系模式：教学（职工号，学号）

选修：选修（学号，课程号，成绩）

讲授：讲授（课程号，职工号，书号）

开设：课程“开设”教室n：1，归并到“课程”（教室号）

**2. 数据模型优化**

\*适当修改、调整数据模型的结构。通常以规范化理论为指导。

**3. 设计用户子模式**

（自看）

使用更符合用户习惯的别名

针对不同级别的用户定义不同的外模式，以满足系统对安全性的要求

简化用户对系统的使用

**7.5 数据库的物理设计**

**1. 确定数据库的物理结构**

数据库的物理设计：对于设计好的逻辑数据模型选择一个最符合应用要求的物理结构。物理结构指：数据库在实际的物理设备上的存储结构和存取方法称为数据库的物理结构。

没有通用的物理设计方法可循，原因有

数据库的物理设计完全依赖于给定的硬件环境和数据库产品的。

可能用到的数据库产品多种多样；不同的数据库产品所提供的物理环境、存储结构和存取方法有很大的区别，能供设计人员使用的设计变量、参数范围也很不相同。

一般的设计内容和原则主要有

确定数据的存储安排

此问题主要是从提高系统性能考虑。例如

将表和索引分别放在不同的磁盘上，在查询时，由于两个磁盘驱动器分别在工作，因而可以保证物理读写速度较快

将比较大的表分别放在两个磁盘上，可以加快存取速度，特别是在多用户环境下

将日志文件和数据库对象（表、索引等）分别放在不同的磁盘可以改进系统的性能。

各系统所提供的对数据进行物理安排的手段、方法差异很大，因此设计人员应仔细了解给定的DBMS在这方面提供了什么方法，再针对应用环境的要求，对数据进行适当的物理安排。

设计数据的存取路径

在关系数据库中，主要是指确定如何建立索引。例如：次码索引的建立，组合索引的建立，其它类型索引。

确定系统配置

数据库产品一般提供了一些存储分配参数，供设计人员和DBA对数据库进行物理优化。初始情况下，系统为这些变量赋予合理的缺省值，但这些值不一定适应不同的应用环境，在进行物理设计时，需要对这些值重新赋值以改善系统性能。

这些配置变量通常包括：同时使用数据库的用户数；同时打开数据库的对象数；使用的缓冲区长度、个数；数据库大小；锁的数目等，这些参数值影响存取时间和存储分配，在物理设计时要根据应用环境确定这些参数值，以使系统性能最优。

系统配置参数在系统运行时需要做进一步调整，以期切实改进系统性能。

**2. 评价数据库的物理结构**

数据库的物理结构设计过程中需要对时间效率、空间效率、维护代价和各种用户要求进行权衡、其结果可以产生多种方案。数据库设计人员必须对这些方案进行细致的评价，从中选择一个较优的方案作为数据库的物理结构。

数据库物理结构的评价完全依赖于所选用的关系数据库管理系统，主要从定量估算各种方案的存储空间、存取时间和维护代价入手，对估算结果进行权衡、比较，选择出一个较优的、合理的物理结构。如果该结构不符合用户需求，则需要修改设计。

**7.6 数据库实施**

1. 用DDL定义数据结构；

2. 组织数据入库；

3. 编制与调试应用程序；

4. 数据库试运行：在数据库的试运行过程中一般的做法是：先输入小批量数据做调试用，待试运行基本合格后再大批量输入数据，逐步增加数据量，逐步完成运行评价。在数据库试运行阶段由于系统的不稳定因此要做好数据库的转出和恢复的工作。

**7.7数据库运行和维护**

1. 数据库的转储和恢复：是数据库正常运行后最重要的维护工作之一，数据库管理员要针对不同的应用需求定制不同的转出计划，以保证一旦发生故障能尽快将数据库恢复到某种一致性状态。

2. 数据库的安全性与完整性控制：由于应用环境的变化，对安全性的要求也会发生变化，比如有的数据原来是机密的，现在是公开的，数据库中用户的密级也可能会发生变化，这些都需要数据库管理员不断修正，以满足用户要求。

3. 数据库性能的监督、分析和改进：目前有一些关系数据库管理系统提供了检测系统性能参数的工具，数据库管理员可以利用这些工具方便地得到系统运行过程中一些列性能参数的值。数据库管理员应仔细分析这些数据，判断当前系统运行状况是否为最佳，应当做哪些改进，例如调整系统物理参数或者对数据库进行重组织或者重构造等。

4. 数据库的重组织和重构造：数据库运行一段时间后由于记录不断增、删、改，将会使数据库的物理存储情况变坏，降低数据的存取效率，是数据库性能下降，这时，数据库管理员就要对数据库进行重组或者部分重组。

### 7.5.6 课时小结

本章主要内容：

1. 什么是数据库设计；

2. 数据库设计的一般步骤；

3. 概念设计（E\_R图）；

4. 逻辑结构设计（关系型逻辑结构设计）：E\_R图向关系模型转换的规则；

5. 物理结构设计；

6. 数据库的实施、运行和维护。

### 7.5.7 课外补充

无

# 第八章 数据库编程

## 8.1 授课内容

授课科目：数据库原理

授课内容： 数据库编程

授课类型：讲授+练习

授课时间：4学时

主讲教师：

## 8.2 教学目标要求

### 8.2.1 能力目标

1. 掌握嵌入式SQL语句的使用的优点；
2. 掌握数据库编程的主要应用场景；
3. 掌握存储过程、自定义函数、游标、触发器的定义和使用。

### 8.2.2 知识目标

1. 嵌入式SQL语句的使用；
2. 存储过程的定义及使用；
3. 自定义函数的定义和使用；
4. 游标的定义和使用；
5. 触发器的定义和使用。

## 8.3 课件分析

概 述： 本章主要介绍嵌入式SQL语句的使用，包括存储过程、游标、自定义函数、触发器的定义及使用；介绍ODBC编程方式及优点。

教学重点：存储过程、触发器的定义和使用。

教学难点： 游标。自定义函数的定义和使用。

## 8.4 教学方法

问题教学法、讲授法，类比法。

## 8.5 教学过程

### 8.5.1 课前补充

（0分钟）

### 8.5.2 上节回顾

（10分钟）

复习第三章SQL的基本语法。

### 8.5.3 作业点评

（0分钟）

### 8.5.4 导入新课

（5分钟）

数据库编程是数据库课程中非常重要的一部分内容，标准的SQL是非过程化的查询语言，具有操作统一、面向集合、功能丰富、使用简单等多项优点。SQL的特点之一是在交互式和嵌入式两种不同的使用方式下其语法结构基本一致。嵌入式SQL是将SQL嵌入到程序设计语言中过程，SQL可以被嵌入到多种程序设计语言中，比如：C、C++、Java等。在宿主语言中嵌入SQL语言来完成程序对数据库中数据的操作已经成为应用系统开发中不可或缺的一部分，不同的宿主语言会采用不同的数据库管理方式，目前主要存在的有：开放数据库互连（Open Data Base Connectivity，ODBC），Java数据库连接（Java Data Base Connectivity,JDBC），过程化SQL（Procedural Language/SQL，PL/SQL），嵌入式SQL（Embedded SQL，ESQL）等。本章内容主要介绍数据库编程中常用的过程化SQL和嵌入式SQL。

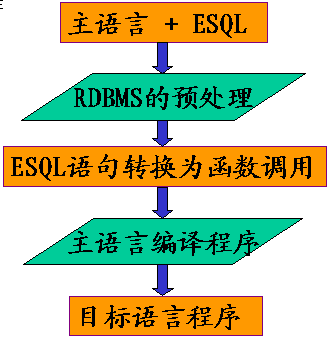
### 8.5.5 讲授新课

**8.1 嵌入式SQL**

**8.1.1 嵌入式SQL的处理过程**

嵌入式SQL是将SQL语句嵌入到程序设计语言中，被嵌入的程序设计语言被称为宿主语言。数据库管理系统一般会采用预编译的方法来处理嵌入式SQL，即数据库管理系统的预处理程序对源程序进行扫描识别出嵌入式SQL语句，把它们转换为主语言调用语句，以使得主语言编译程序能识别它们。

**8.1.2 嵌入式SQL语句与主语言间的通信**

****

主要包括：

向主语言传递SQL语句的执行状态，使主语言能够据此信息控制程序流程----SQLCA，主语言向SQL语句提供参数----主变量，将SQL语句查询数据库的结果交给主语言----主变量、游标。

T-SQL编程基础



**8.2 游标的定义和使用**

游标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，用户可以通过游标注意获取记录并赋值给变量，交由主语言程序进一步处理。

游标的使用包括以下几个步骤：

1.声明游标

声明游标的语法：

DECLARE <游标名> [INSENSITIVE] [SCROLL] CURSOR

FOR <SELECT语句>

[FOR {READ ONLY | UPDATE [OF <列名> [,...n]]}]

Insensitive指明游标只对基本表的副本操作，游标的任何操作不对基本表产生影响。SCROLL指定游标推进的方向，取值有（ FIRST、LAST、PRIOR、NEXT、RELATIVE、ABSOLUTE）。

例：声明一个游标，统计没有选修课程的学生的人数

declare num\_cursor cursor

for

select sno

from student

for READ ONLY

2.打开游标

OPEN <游标名>

例：open num\_cursor;

3.读取游标中的数据

基本语法：FETCH

[[NEXT|PRIOR|FIRST|LAST|ABSOLUTE {n|@nvar}

|RELATIVE {n|@nvar}]

FROM ]

{{[GLOBAL] <游标名>}|<@游标变量>}

[INTO @<变量名>[,...n]]

执行FETCH语句后，可通过@@FETCH\_STATUS全局变量返回游标当前的状态。@@FETCH\_STATUS 变量有三个不同的返回值：0代表执行成功，-1代表执行失败或者超出游标范围，-2代表数据不存在。

例：

declare num\_cursor cursor --声明

for select sno from student

for READ ONLY

Open num\_cursor; --打开

declare @sno varchar(10),@num int --声明变量

set @num = 0

fetch next from num\_cursor --取信息

into @sno

while @@fetch\_status = 0 --检测状态

begin

if not exists(select \*

from sc

where sno = @sno)

set @num = @num + 1

fetch next from num\_cursor

into @sno

end

select @num 未选课人数

CLOSE num\_cursor

DEALLOCATE num\_cursor

4.关闭游标

CLOSE [GLOBAL] <游标名>|@<游标变量>

5.释放游标

关闭游标只是关闭游标并没有释放游标所占用的数据结构，通过下面的语句可以释放游标所占用的资源。

DEALLOCATE [GLOBAL] <游标名>|@<游标变量>

**8.3 存储过程概述**

存储过程（Stored Procedure）是一组完成特定功能的SQL语句集，经编译后存储在数据库中，用户通过指定存储过程的名字并给出参数（如果该存储过程带有参数）来执行存储过。

存储过程的优点：存储过程已在服务器注册、存储过程具有安全特性、存储过程可以强制应用程序的安全性、存储过程允许模块化程序设计、存储过程可以减少网络通信流量。

存储过程的使用包括下面三个步骤：



1.存储过程的创建

创建存储过程的语法：

CREATE { PROC | PROCEDURE } [schema\_name.] procedure\_name

[ { @parameter data\_type }[ VARYING ] [ = default ] [ OUTPUT ] ] [ ,...n ]

[ WITH RECOMPILE | ENCRYPTION]

AS

{ <BEGIN>

<sql\_statement> [;][ ...n ]

<END>

}

例：从SC表中查询不及格课程超过3门的学生信息

create procedure myproc

as

select \*

from student

where sno in

(

select sno

from sc

where grade<60

group by sno

having count(\*)>3

);

存储过程中还可以带输出参数，例：查询指定学号学生的平均成绩，并将平均成绩返回

create proc proc\_avergrade

@sno varchar(10),

@savg int out --输出参数

as

begin

select @savg = avg(grade)

from sc

where sno = @sno

end

2.存储过程的执行

语法：EXEC|EXECUTE [ @return\_status = ] [schema\_name.]procedure\_name

[[@parameter =] {value | @variable [OUTPUT] | [ DEFAULT ]}][ ,...n ]

3.存储过程的删除

drop procedure {procedure\_name}[,…n]

**8.4 自定义函数**

SQLServer中支持三种自定义函数，分别是：标量函数、内嵌表值函数、多语句表值函数。使用自定义函数可以提高SQL的执行效率，减少网络流量，还可以对程序进行模块化设计。

1.标量函数

CREATE FUNCTION [ schema\_name.] function\_name

([{ @parameter\_name [ AS ] data\_type [ = default ] } [ ,...n ]])

RETURNS return\_data\_type

[ WITH <ENCRYPTION>|<SCHEMABINDING> [ ,...n ] ]

[ AS ]

BEGIN

function\_body

RETURN scalar\_expression

END

例：定义一个函数返回不带时间的日期

CREATE FUNCTION dbo.DateOnly(@date datetime)

RETURNS VARCHAR(12)

AS

BEGIN

RETURN CONVERT(VARCHAR(12),@DATE,101)

END

2.内嵌表值函数

基本语法：

CREATE FUNCTION [ schema\_name. ] function\_name

([{ @parameter\_name  data\_type [ = default ] } [ ,...n ]])

RETURNS TABLE

[ WITH <function\_option> [ ,...n ] ]

[ AS ]

RETURN ( select\_stmt )

例：定义一个函数返回学生的学号和姓名

CREATE FUNCTION dbo.Fun1()

RETURNS table

AS

RETURN

SELECT SNO,SNAME

FROM STUDENT

3.多语句表值函数

基本语法：**CREATE FUNCTION** [ schema\_name. ] function\_name

([{@parameter\_name  data\_type [ = default ] } [ ,...n ]])

RETURNS @return\_variable TABLE < table\_type\_definition >

[ WITH <function\_option> [ ,...n ] ]

[ AS ]

BEGIN

function\_body

RETURN

END

例：现有一个员工表test （字段省），一个部门表bm （字段省）。查询某部门的员工信息

**CREATE FUNCTION** fn\_salary ( @bm char(2) )  
RETURNS @salary table  
(姓名 varchar(10),部门名称 varchar(10),工资 numeric(8,2))

as

begin

insert @salary

select a.姓名,b.部门名称,a.工资

from test a left join bm b on a.部门=b.部门 and [a.部门=@bm](mailto:a.部门=@bm)

return

end

**8.5 触发器**

触发器（Trigger）是用户定义在关系表上的一类由事件驱动的特殊过程，是一种特殊的存储过程，它在执行事件时自动生效。SQLServer2008中的触发器分为DML触发器和DDL触发器。

触发器的作用：触发器可以对数据库进行级联修改、实现比CHECK约束更为复杂的限制、比较数据修改前后的差别、强制表的修改要合乎业务规则。

1.DML触发器

DML触发器是在对表进行插入、更新或删除操作时自动执行的存储过程。包括delete触发器、insert触发器、update触发器。触发器在触发时系统自动在内存中创建deleted和inserted表，临时保存删除掉的数据和新插入（或更新）的数据。

DML触发器的基本语法：

CREATE TRIGGER [ schema\_name . ]trigger\_name

ON { table | view }

[ WITH ENCRYPTION ]

{ FOR | AFTER | INSTEAD OF }

{ [ INSERT ] [ , ] [ UPDATE ] [ , ] [ DELETE ] }

[ NOT FOR REPLICATION ]

AS

begin sql\_statement [ ; ] end

例1：建立一个触发器，当向sc表中添加数据时，如果添加的数据与student表中的数据不匹配（没有对应的学号），则将此数据删除。

CREATE TRIGGER tr\_sc\_insert ON sc

FOR INSERT

AS

BEGIN

DECLARE @bh char(10)

Select @bh=Inserted.sno from Inserted

If not exists(select sno from student where student.sno=@bh)

Delete from sc where sno=@bh

END

例2：创建一个触发器，当插入或更新成绩列时，该触发器检查插入的数据是否处于设定的范围内（0，100）。

CREATE TRIGGER tr\_sc\_grade

ON sc

AFTER INSERT,UPDATE

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

DECLARE @score int;

SELECT @score=inserted.grade from inserted

IF (@score<0 or @score > 100)

BEGIN

RAISERROR ('成绩的取值必须在0到100之间', 16, 1)

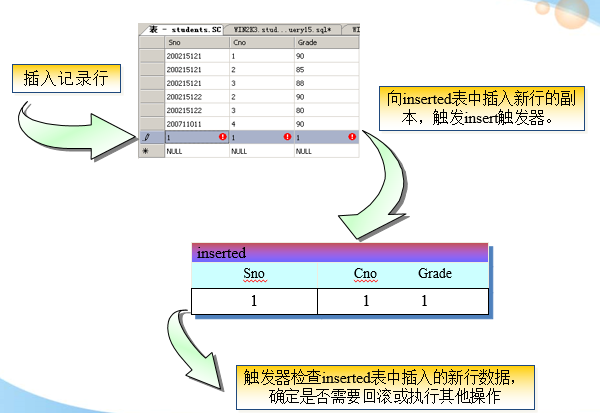
ROLLBACK TRANSACTION

END

END

Raiserror函数的作用是抛出一个错误， 第一个参数：错误的提示消息、第二个参数：错误的消息级别（0~18之间）、第三个参数：错误的状态号（1~127之间）。

Insert触发器执行的基本流程如下图所示：



例3： 创建一个修改触发器，该触发器防止用户修改表student的学号。

CREATE TRIGGER tr\_student\_sno ON student

AFTER UPDATE

AS

BEGIN

SET NOCOUNT ON;

if update(sno)

begin

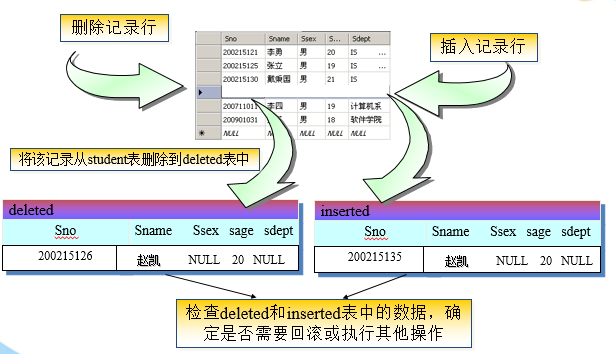
raiserror('不能修改学号',16,10)

rollback transaction

end

END

Update触发器执行的基本流程如下图所示：



### 8.5.6 课时小结

本章主要内容：

1. 数据库编程——嵌入式SQL、过程化SQL；

2. 数据库编程的基本语法；

3. 游标的定义和使用；

4. 存储过程的定义和使用；

5. 自定义函数的定义和使用；

6. 触发器的定义和使用。

### 8.5.7 课外补充

无

# 第十章 数据库恢复技术

## 10.1授课内容

授课科目：数据库原理

授课内容：数据库恢复技术

授课类型：讲授

授课时间：4学时

主讲教师：

## 10.2 教学目标要求

### 10.2.1 能力目标

1. 理解事务的基本概念和事务的特性；
2. 了解数据库系统常见的故障类型；
3. 掌握不同的数据库故障的解决方案；
4. 能解决数据库常见的故障问题。

### 10.2.2 知识目标

1. 数据库事务的基本概念，事务的四大特性；
2. 数据库故障的种类；
3. 数据库故障的恢复技术和恢复策略；
4. 数据库镜像的作用。

## 10.3 课件分析

概 述： 本章介绍数据库恢复技术。包括数据库运行中常见的故障类型，数据库恢复中最经常使用的技术——数据转储和登录日志文件。介绍数据库转储的分类和具体的应用，日志文件的内容及作用，登记日志文件所要遵循的原则。针对事务故障、系统故障和介质故障等不同故障的恢复策略和恢复方法。具有检查点的恢复技术。数据库镜像功能。

教学重点：常见的数据库故障及其恢复策略，数据转储和日志文件的使用。

教学难点：带检查点的数据库故障恢复技术。

## 10.4 教学方法

讲授法，类比法。

## 10.5 教学过程

### 10.5.1 课前补充

（0分钟）

### 10.5.2 上节回顾

（5分钟）

数据库编程，重点内容数据库中的SQL编程，游标、存储过程、自定义函数、触发器的定义、调用、销毁（删除）。

### 10.5.3 作业点评

（0分钟）

### 10.5.4 导入新课

（5分钟）

第10章、11章讨论数据库的事务处理技术，事务是一系列的数据库操作，是数据库应用程序的基本逻辑单元。事务处理技术主要包括数据库恢复技术和并发控制技术。数据库恢复机制和并发控制机制是数据库管理系统的重要组成部分。本章的内容重点讨论数据库恢复的概念和常用的技术。

### 10.5.5 讲授新课

**10.1 事务的概念**

在学习数据库恢复技术之前，首先要明确事务的基本概念及其特性。

事务(Transaction)是访问并可能更新数据库中各种数据项的一个程序执行单元(unit)。事务通常由高级数据库操纵语言或编程语言（如SQL，C++或Java）书写的用户程序的执行所引起，并用形如begin transaction和end transaction语句（或函数调用）来界定。

事务是由事务开始(begin transaction)和事务结束(end transaction)之间执行的全体操作组成。事务是由一系列操作序列构成的程序执行单元，这些操作要么都做，要么都不做，是一个不可分割的工作单位。例如：银行转帐。事务是恢复和并发控制的基本单位。

事务具有4个特性：原子性、一致性、隔离性、持久性。这四个特性通常称为ACID特性。

原子性（atomicity），一个事务是一个不可分割的工作单位，事务中包括的诸操作要么都做，要么都不做。

一致性（consistency），事务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。一致性与原子性是密切相关的。

隔离性（isolation），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并发的其他事务是隔离的，并发执行的各个事务之间不能互相干扰。

持久性（durability），持久性也称永久性（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有任何影响。

**10.2 数据库故障分类**

“数据故障恢复”和“完整性约束”、“并发控制”一样，都是数据库数据保护机制中的一种完整性控制。所有的系统都免不了会发生故障，有可能是硬件失灵，有可能是软件系统崩溃，也有可能是其他外界的原因，比如断电等等。运行的突然中断会使数据库处在一个错误的状态，而且故障排除后没有办法让系统精确地从断点继续执行下去。这就要求DBMS要有一套故障后的数据恢复机构，保证数据库能够回复到一致的、正确的状态去。而“数据故障恢复”正是这样一个机制。

数据库的故障主要包括：事务故障、系统故障、介质故障、计算度病毒。

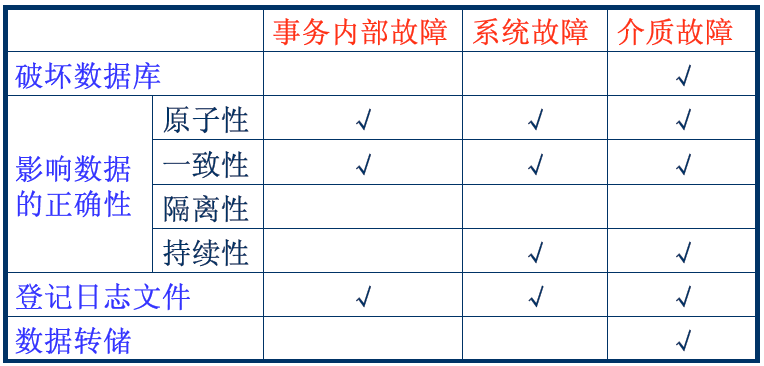
一个事务失败了，应该能够撤消该事务对数据库的影响。如果有其它事务读取了失败事务写入的数据，则该事务也应该撤消。事务故障是指事务由于运算溢出、事务发生死锁被撤消、更新违反完整性而引起的事务非正常结束。事务故障更多的是非预期的，大部分是不能由应用程序处理的。一般事务故障的恢复是撤销事务。

系统故障称为软故障，事务由于硬件错误、操作系统错误、DBMS软件错误、突然断电而引起的事务非正常结束。重新启动数据库时，必须撤消未完成的事务；对于已经完成的事务可能并未完全写入到数据库中，必须重新执行所有已经提交的事务。

介质故障称为硬故障，由于磁盘损坏、磁头碰撞、强磁场干扰等原因引起的外存故障。利用备份数据和备份日志重新建立数据库。

计算机病毒是一种人为的故障或者破坏，是一些恶作剧者研制的一种计算机程序，可以繁殖和传播。计算机病毒的危害极大，可以破坏系统文件或者盗取系统的数据。

各类故障对数据库的影响主要有两种，一种是数据库本身被破坏，一种导致是数据库中的数据可能不正确，这主要是因为事务的运行被非正常结束。



**10.3 数据库恢复技术**

数据库恢复是把数据库从错误状态恢复到某一正确状态的功能，从而确保数据库的一致性。恢复的基本原理是冗余，即数据库中任一部分的数据可以根据存储在系统别处的冗余数据来重建。

数据库恢复技术主要有两种：一种是数据转储，一种是登记日志文件。

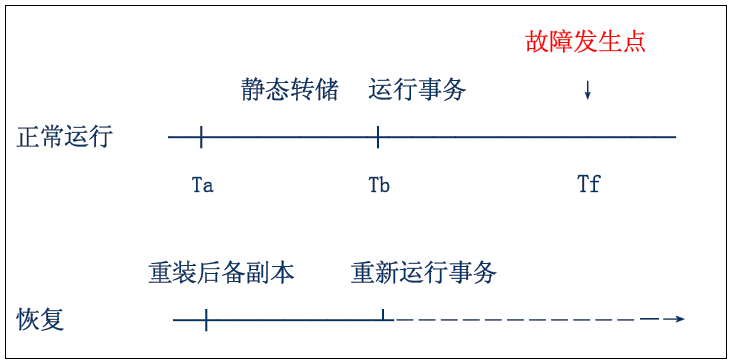
1．数据转储

转储即DBA定期将整个数据库复制到另一个磁盘上保存起来的过程，这些备用的数据文本称为后备副本或后援副本。

当数据库遭到破坏后可将后备副本重新装入，恢复到转储时状态。

数据转储按照转储的状态可以分为：静态转储、动态转储。按照转储的方式分为：海量转储、增量转储。

静态转储是指在系统中无运行事务时进行转储，转储过程中不允许对数据库进行任何的存取、修改。静态转储的优点是实现简单，缺点是降低了数据库的可用性。



动态转储是指转储的操作与用户事务并行，转储期间允许对数据库进行存取或修改。动态转储的最大优点是不用等待正在运行的用户事务结束，并且不影响新事务的执行。缺点是不能保证副本中数据的正确有效。动态转储的副本恢复数据库时需要联合日志文件一起完成。

海量转储是指每次转储全部的数据库，增强转储指只转储上次转储后更新过的数据。从恢复的角度看海量转储得到后备副本进行恢复更方便，如果数据库很大 ，事务处理很繁琐，增量转储更方便实用。

在实际的数据库维护中应该定期进行数据转储，制作后备副本，但是转储十分的耗费资源和时间也不能太频繁进行，所以DBA应该根据数据库使用情况确定适当的转储周期和转储方法。

2．登记日志文件

日志文件是用来记录事务对数据库的更新操作的文件，由系统自动记录。不同数据库采用的日志文件格式不一样。

日志内容包括：事务开始标记，事务标识符（标明那个事务）、操作标识符（增删改）、操作对象(记录内部标识)、旧记录值、新记录值，事务结束标记。

由于日志包含了所有数据库活动的完整记录，所以日志中存储的数据量会变得非常大。

日志的作用：事务故障恢复和系统故障恢复必须用日志；在动态转储方式中必须转存日志文件，由后援副本和日志结合有效恢复数据库；在静态转储方式中，利用日志文件可以恢复到故障前某一时刻的正确状态。

登记日志文件：DBMS在登记日志文件时，遵循两条原则：登记次序严格按照并发事务执行的时间次序；先写日志的原则（WAL）先写日志文件，后写数据库。

对于尚未提交的事务，在将DB缓冲区写到外存之前，必须先将日志缓冲区内容写到外存去。

如果先写DB，则可能在写的中途发生系统崩溃，导致内存缓冲区内容丢失，而外存DB处于不一致状态，由于日志缓冲区内容已丢失，导致无法对DB恢复。

日志记录将要发生何种修改，写入DB表示实际发生何种修改。

**10.4 数据库恢复策略**

基本恢复操作：对圆满事务所做过的修改操作应执行redo操作，即重新执行该操作，修改对象被赋予新记录值。

对夭折事务所做过的修改操作应执行undo操作，即撤消该操作，修改对象被赋予旧记录值。

数据库恢复的策略包括：

1．事务故障的恢复

利用日志文件撤消非正常终止的事务，事务故障发生后由系统自动完成。

反向扫描日志，查找该事务的更新操作，执行该更新操作的逆操作，继续查找该事务其他更新操作，直到该事务的开始标记。

2．系统故障的恢复

利用日志撤消未完成的事务，重做已经完成的事务，重启系统，由系统自动完成。

从头正向扫描日志文件，找出已经有结束标志的事务，将其事务标记为重做（REDO）队列；同时找出没有结束的事务，将其事务标记为撤消（UNDO）队列。正向扫描日志文件，对于每一个REDO事务重新执行。反向扫描日志文件，对于每一个UNDO事务，执行更新的逆操作。

系统故障恢复的策略复杂，一般很耗时。

3．介质故障的恢复

重新装入最近的数据副本和有关的日志文件副本，然后执行系统提供的恢复命令。

介质故障的恢复需要由DBA介入，重装数据库。

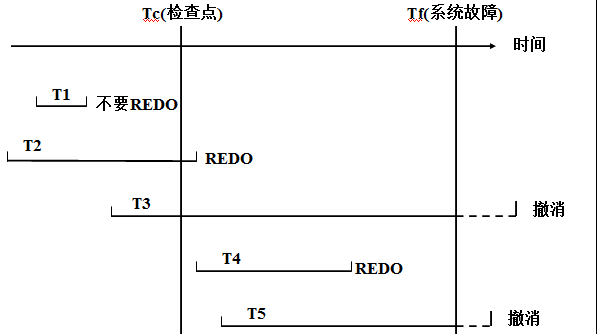
**10.5 带检查点技术的故障恢复**

利用日志进行系统恢复时，必须扫描日志，决定哪些事务需要REDO，哪些事务需要UNDO。原则上需要扫描整个日志来决定这一信息。这种方法有两种困难：一是搜索过程太耗时；二是很多REDO处理其更新已经写入数据库，尽管再做一次不会造成不良后果，但会使恢复过程变得更长。

为降低这种开销，引入检查点。系统周期性的执行检查点（checkpoint）,做如下操作：

将内存中所有日志记录输出到日志文件；将内存中所有修改了的数据block输出到数据文件；将一个日志记录（checkpoint）输出到日志文件。

故障发生后，检查日志来确定最近的检查点发生前开始执行的最近的事务清单。



从日志尾部由后向前扫描日志，找到第一个（checkpoint）记录，即日志文件中最后一个（checkpoint）记录。

由最后一个（checkpoint）记录得到该检查点发生时正在执行的事务清单1。

再扫描检查点之后的日志记录，得到事务清单2。

由事务清单1和事务清单2组成的事务清单，对不同事务采取不同的恢复策略。

制定如下规则：检查点执行过程中，不允许新的事务执行，必须等到所有事务正常结束，并且将修改的数据信息和日志信息分别写入数据库和日志文件中后才算检查点结束。

那么恢复时，只需找到最后一个检查点记录，并且从最后一个检查点记录开始进行恢复即可。

**10.6 数据库镜像**

根据DBA要求DBMS自动把整个数据库或其中的关键数据复制到另一个磁盘上，称为镜像。DBMS自动保证镜像数据与主数据的一致性。

一旦出现介质故障，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恢复。

实际应用中，往往只对关键数据和日志文件镜像，而不对整个数据库镜像。

ORACLE提供对控制文件和日志成员的镜像功能。

### 10.5.6 课时小结

本章主要内容：

1. 数据库事务的基本概念，事务的特性；

2. 常见的数据库故障的种类；

3. 数据库故障恢复技术和策略：数据库转储、登记日志文件；

4. 带检查点的数据库故障恢复技术；

5. 数据库镜像。

### 10.5.7 课外补充

无

# 第十一章 数据库并发控制

## 11.1 授课内容

授课科目：数据库原理

授课内容： 数据库并发控制

授课类型：讲授

授课时间：4学时

主讲教师：

## 11.2 教学目标要求

### 11.2.1 能力目标

1. 了解用户的并发操作带来的数据库问题；
2. 掌握数据库中解决并发问题的机制——封锁机制；
3. 掌握数据库中的调度策略；
4. 了解两段锁协议及其在数据库中的应用。

### 11.2.2 知识目标

1. 用户的并发操作带来的并发问题；
2. 封锁机制，活锁和死锁；
3. 并发调度的可串行性；
4. 两段锁协议；
5. 封锁的粒度。

## 11.3 课件分析

概 述： 本章介绍数据库中的并发问题的产生原因，以及在数据库中解决并发问题常用的封锁机制，以及两段锁协议和锁的粒度确定。介绍数据库中并发的可串行性，只有可以串行化的并发调度策略才是正确的调度策略。

教学重点：封锁机制，活锁和死锁及其解决方案。

教学难点：可串行化的并发调度，两段锁协议。

## 11.4 教学方法

讲授法，案例法。

## 11.5 教学过程

### 11.5.1 课前补充

（0分钟）

### 11.5.2 上节回顾

（5分钟）

数据库运行过程中可能常见的四种故障是：事务故障、系统故障、介质故障、计算机病毒，当发生这些故障时为了保证数据库的正常运行，需要DBA进行数据库的故障恢复，常用的故障恢复手段有：数据转储、登记日志文件，数据库故障恢复的核心思路是冗余，采用冗余的数据存储，来保障数据库出现故障后能及时准确地恢复。

### 11.5.3 作业点评

（0分钟）

### 11.5.4 导入新课

（5分钟）

数据库是一个共享资源，可以供多个用户使用。允许多个用户同时使用同一个数据库的数据库系统成为多用户数据库系统。比如飞机票的订票系统，银行数据库系统等都是多用户数据库系统。在这些系统中同一时刻并发访问的人数可达到数百上千个。那么，数据库系统中应该如何来处理这些并发的访问呢？本章的内容就主要介绍数据库中对并发的控制策略和机制。

### 11.5.5 讲授新课

**11.1 并发控制概述**

数据库是多用户系统，在多个同时操作数据库时，可能会产生多个事务同时存取同一数据的情况，或者可能会存取和存储不正确的数据，破坏事务一致性和数据库的一致性。

事务是并发控制的基本单位，DBMS的并发控制是以事务为单位进行的。在第十章中数据库恢复技术中介绍了事务的基本概念和特性。事务的四大特性是：原子性、一致性、隔离性、持续性。

保证事务的ACID四个特性是事务处理的重要任务，而多事务并发对数据库操作可能会破坏事务的ACID特性，所以DBMS需要对并发操作进行正确的调度，以确数据库中数据的正确和合法性。

示例：飞机订票系统中的一个活动序列

① 甲售票点(甲事务)读出某航班的机票余额A，设A=16；

② 乙售票点(乙事务)读出同一航班的机票余额A，也为16；

③ 甲售票点卖出一张机票，修改余额A←A-1，所以A为15，把A写回数据库；

④ 乙售票点也卖出一张机票，修改余额A←A-1，所以A为15，把A写回数据库

结果明明卖出两张机票，数据库中机票余额只减少1。

事务的原子性属性用以保证数据库修改删除操作时的数据库一致性。例如某一学号为2001的学生信息同时出现在student表与sc表中，要删除该学生的信息，得同时删除student表及sc表中有关该学生的信息。

时刻 程序应完成的工作 系统故障

t1 删除学生表中2001号学生信息

t2 系统掉电或系统故障

t3 删除学生选课表中2001号学生信息

如银行转帐的例子，把甲银行的1万元钱转到乙银行

时刻 程序应完成的工作 系统故障

t1 甲银行减1万

t2 系统掉电或系统故障

t3 乙银行加1万

COMMIT; //成功修改，提交事务

事务的开始与结束控制

1.用户显式控制，即显式定义，在SQL语言中定义事务的语句有三条：

BEGIN TRANSACTION（开始）

COMMIT（提交事务的所有操作，即将事务中所有对数据的更新写回磁盘中的物理数据库中，事务正常结束）

ROLLBACK（发生故障，即系统将事务中对数据库的所有已完成的更新操作全部撤消，回滚到事务开始时的状态）

例如，要删除2001号学生的信息，得同时删除student表及sc表中有关该学生的信息。

BEGIN TRANSCATION（开始）

Delete from student where sno=’2001’;

If sqlca.sqlcode<>sucess then

ROLLBACK;

Endif;

Delete from sc where sno=’2001’;

If sqlca.sqlcode=sucess then

COMMIT;

else

ROLLBACK;

Endif;

2. 如果用户没有显式定义，则由DBMS按缺省规定自动划分事务，有些系统的缺省处理方法为，将每一个SQL语句都作为单独的一个事务处理。

通常情况下一个应用程序包括多个事务。

何为“事务”：是数据库的逻辑工作单位，它是用户定义的一组操作的序列。应该具有4个属性：原子性、一致性、隔离性和持续性。（回答事务的概念时，属性不可少）

原子性：一个事务是一个不可分割的工作单位，事务中包括的诸操作要么都做，要么都不做。

一致性：事务必须使数据库从一个一致性状态变到另一个一致性状态。

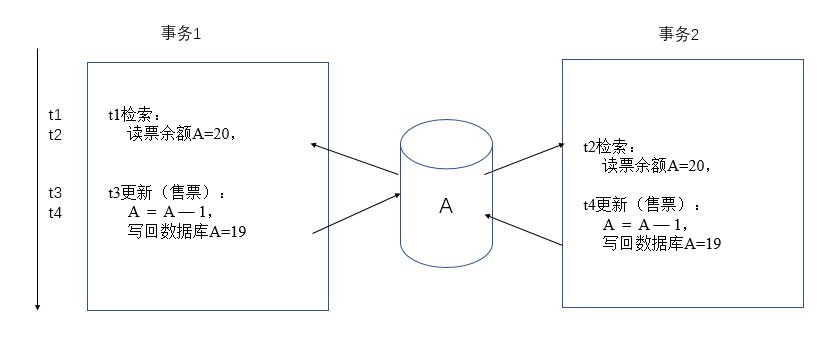
前面的修改操作与银行转帐的例子说明，原子性为数据的一致性提供了一定的保证。后面会看到在多个事务并发执行时，事务间常常出现相互干扰的现象，事务的原子性特点不能完全保证数据的一致性。

隔离性：并发执行的各个事务之间不能相互干扰。

（补充操作系统进程调度的概念，以分时系统为例）

并发事务对数据进行并发操作时，如对并发操作不进行合适控制，事务间就可能出现相互干扰，因而导致数据库数据的不一致性。

如：汽车（飞机等）售票系统，存在两个事务



2事务的提交结果破坏了1事务的提交结果。1、2事务的相互干扰，导致t3时刻的修改操作被丢失。

持续性：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有任何影响。

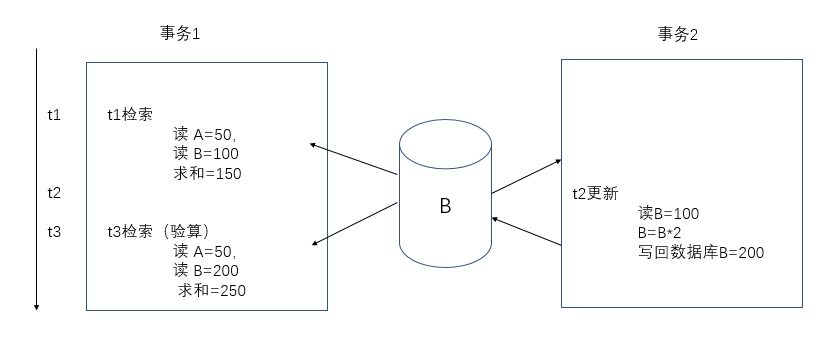
并发操作与数据的不一致性：前述（事务的隔离性中），并发事务对数据进行并发操作时，如对并发操作不进行合适控制，事务间就可能出现相互干扰，因而导致数据库数据的不一致性。

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。

丢失修改：多个事务读同一数据并修改，一个事务的提交结果破坏了另一事务的提交结果，导致修改被丢失。

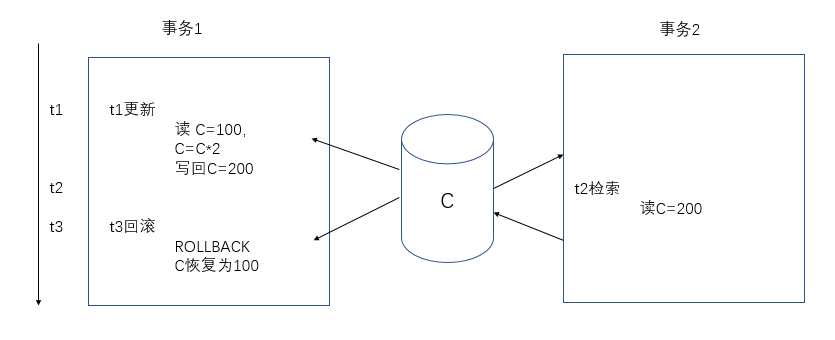
隔离性中的例子属于此种情况。不止两个事务的情况？

不可重复读：事务1读取数据后，事务2执行更新操作，使事务1无法再现前一次读取的结果。具体包括三种情况。



事务1读某一数据，事务2修改；事务1按一定条件从数据库中读某些数据记录后，事务2删除了其中某些记录，事务1再次按相同条件读取数据时，发现某些记录神秘消失（幻行现象，即看花眼了）；事务1按一定条件从数据库中读某些数据记录后，事务2插入了某些记录，事务1再次按相同条件读取数据时，发现多了某些记录（幻行现象）。

读脏数据：



事务1修改某一数据，并将其写回磁盘，事务2读取同一数据后，事务1由于某种原因被撤消，这时事务1修改过的数据恢复原值，事务2读到的数据就与数据库中的数据不一致，是不正确的数据，称其为“脏数据”。

并发控制：用正确的方式调度并发操作，使一个事务的执行不受其他事务的干扰，从而避免造成数据的不一致性。

思考：如果将所有事务串起来调度，能否避免并发操作引起的三类数据不一致性情况？

**11.2 封锁技术**

封锁是事务并发执行的一种调度和控制手段，它可以保证并发执行的事务间相互隔离、互不干扰，从而保证并发事务的正确执行。

所谓封锁（封锁的含义）是指事务T在对某个数据对象（例如表、记录等）操作前，先向系统发请求，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，其他事务不能更新此数据对象。

一个事务对某个数据对象加锁后究竟拥有什么样的控制由封锁的类型决定。基本的封锁类型有两种：排它锁（简称X锁）和共享锁（简称S锁）。

排它锁（Xlock）（又称写锁）：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，其它任何事务都不能再对A加任何类型锁，直到T释放A上的锁。（释放X锁前保证其他事务不能读取和修改A）

共享锁（Slock）（又称读锁）：若事务T对数据对象A加上S锁，则其它事务只能再对A加上S锁，而不能加X锁，直到T释放A上的S锁。（释放S锁前，保证其它事务不能修改A）。

封锁类型的相容矩阵：

|  |  |
| --- | --- |
| T2  T1 | X S — |
| X  S  — | N N Y  N Y Y  Y Y Y |

说明：

T1已经获得的数据对象上的锁的类型，其中横线表示没有加锁；

T2对同一数据对象发出的封锁请求，能被满足Y（相容），不能被满足N（冲突）。

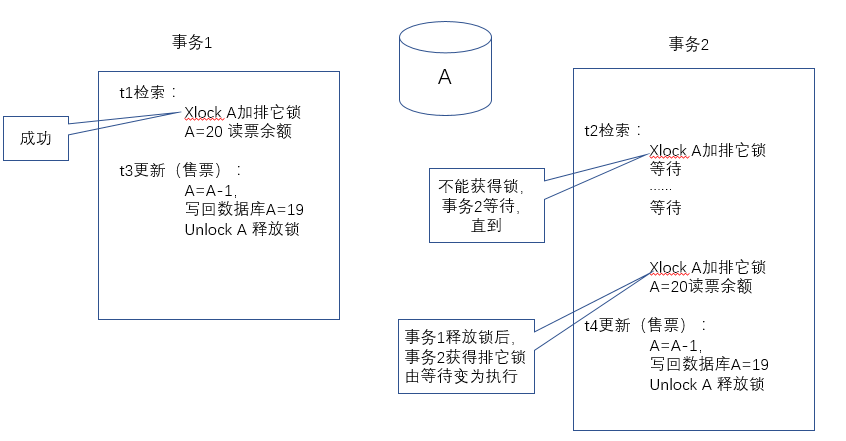
封锁的协议：

1. 一级封锁协议

内容：事务T在修改数据R之前必须先对其加X锁（何时加锁），直到事务结束才释放（何时释放持锁时间）。事务结束包括正常结束（commit）和非正常结束(rollback)。

一级封锁协议可以防止丢失修改。

如：汽车（飞机等）售票系统，存在两个事务（结合进程调度过程说明调度控制过程）

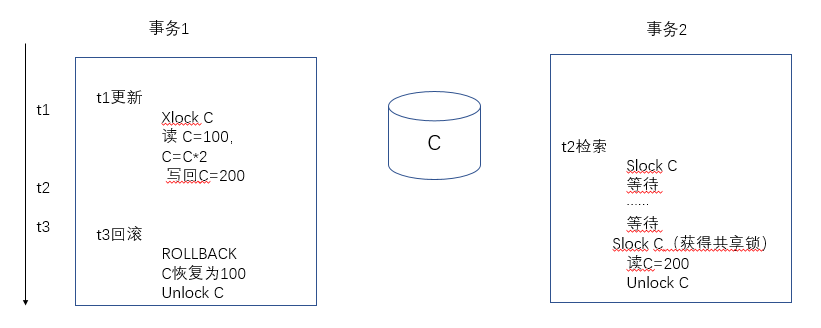


一级封锁协议的效果将修改操作串行化。

2. 二级封锁协议

内容：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，读完后即释放S锁。

二级封锁协议防止可避免读“脏数据”现象发生。

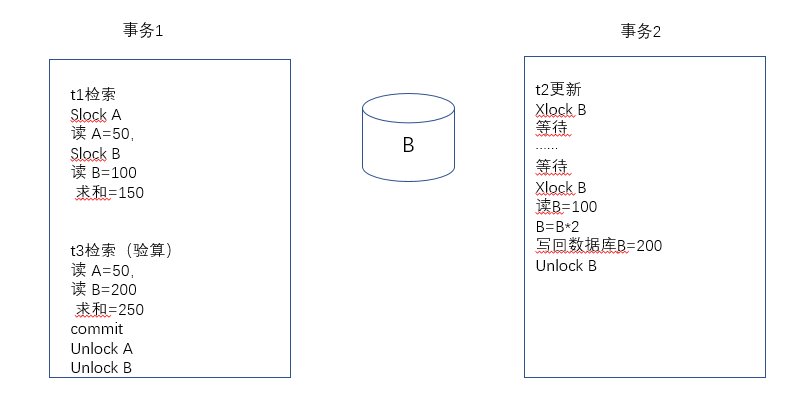


不能读取正在更新的数据，即读不到“脏数据”。但不能保证可重复读（在两次读期间可能发生数据的增删改操作）。

3. 三级封锁协议

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

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。数据重复读之间不可能被修改。



4. 不同封锁协议的比较

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 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 |  | V |  |  | V |  |  |
| 二级封锁协议 |  | V | V |  | V | V |  |
| 三级封锁协议 |  | V |  | V | V | V | V |

**11.3 活锁和死锁**

活锁

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 事务T1 | 事务T2 | 事务T3 | 事务T4 | …. |
| Xlock R  .  .  Unlock R | Xlock R  等待  .  .  .  .  等待  . | Xlock R  等待  Xlock R  .  Unlock R | Xlock R  等待  Xlock R  .  Unlock R |  |

活锁概念：指某些事务永远处于等待状态。

避免活锁的简单方法：先来先服务。

死锁

|  |  |
| --- | --- |
| 事务T1 | 事务T2 |
| Xlock A  ……  Xlock B  等待  .  .  .  .  等待 | Xlock B  ……  Xlock A  等待  .  .  .  .  等待 |

死锁概念：事务对锁的循环等待。

解决死锁的常用办法：

1.预防法：

顺序申请法：将封锁对象按顺序编号，事务在申请封锁时按编号顺序申请，这样能避免死锁的发生。

一次性申请法：事务在开始执行时将它需要的所有锁一次性申请完成，并在操作完成时一次性归还所有的锁。

2.死锁解除法：

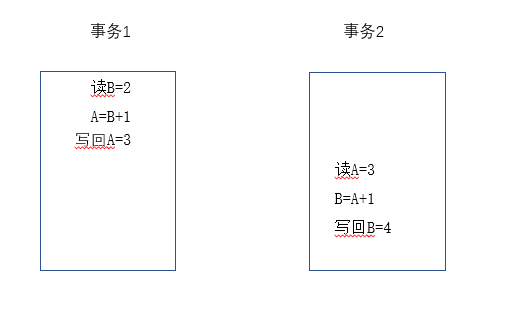
定时法：对每个锁设定一个时限，当事务等待此锁超过时限后认为已经产生死锁，此时调用解锁程序以解除死锁。

死锁检测法：系统内设定一个死锁检测程序，定时检查系统中是否产生死锁，一旦发现死锁调用死锁解除程序解锁。

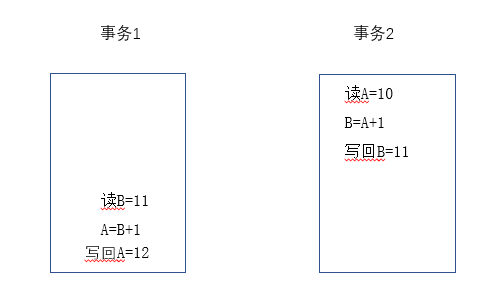
**11.4 并发操作的调度**

计算机系统对并行事务中的并行操作的调度是随机的，不同的调度可能产生不同的结果。但是将所有事务串起来的调度策略一定是正确的调度策略。

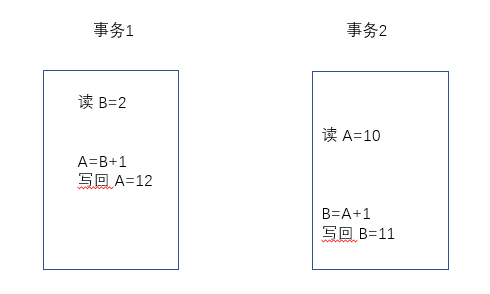
1. 串行调度：



2. 串行调度：



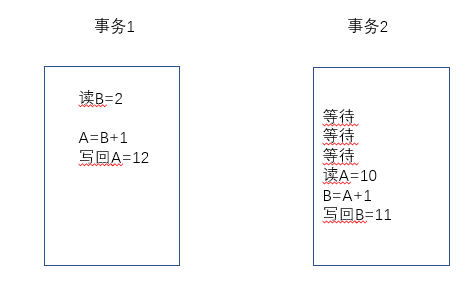
3. 不可串行化调度（发生干扰）：



A=B+1

写回A=12 （丢失修改）

4. 可串行化调度



可串行化的调度（即可串行性）是并行事务正确性的唯一准则。

可串行化的调度指：几个事务的并行执行是正确的，当且仅当其结果与事务按某一次序串行执行时的结果相同。

为了保证并行操作的正确性，DBMS的并行控制机制必须提供一定的手段来保证调度是可串行化的。

事务的并发执行：在多个应用、多个事务执行中有如下几种方法

1. 串行执行：以事务为单位，多个事务顺序执行，此种执行称为串行执行。串行执行能保证事务的正确执行。

2. 并行执行：以事务为单位，多个事务按一定调度策略同时执行，此种执行为并行执行。

3. 并发执行的可串行化：事务的并发执行并不能保证事务的正确性，因此需要采用一定的技术，使得并发执行时象串行一样（正确），此种执行称为并发事务的可串行化。而所采用的技术成为并发控制技术。

例子：事务T1从账号A（初值为20000）转10000到账号B（初值为20000），事务T2从账号A转10%的款项到账号B，其具体程序如下。

T1:

Read(A)

A:=A-10000

Write(A)

Read(B)

B:=B+10000

Write(B)

T2:

Read(A)

temp:=A\*0.1

Write(A)

Read(B)

B:=B+temp

Write(B)

串行执行之一（A与B的存款总额均为40000，保持了其一致性）

|  |  |
| --- | --- |
| T1 | T2 |
| Read(A)  A:=A-10000  Write(A)  Read(B)  B:=B+10000  Write(B) |  |
|  | Read(A)  temp:=A\*0.1  Write(A)  Read(B)  B:=B+temp  Write(B) |

串行执行之二（A与B的存款总额均为40000，保持了其一致性）

|  |  |
| --- | --- |
| T1 | T2 |
|  | Read(A)  temp:=A\*0.1  Write(A)  Read(B)  B:=B+temp  Write(B) |
| Read(A)  A:=A-10000  Write(A)  Read(B)  B:=B+10000  Write(B) |  |

并发执行可串行化（A与B的存款总额均为40000，保持了其一致性）

|  |  |
| --- | --- |
| T1 | T2 |
| Read(A)  A:=A-10000  Write(A) |  |
|  | Read(A)  temp:=A\*0.1  Write(A) |
| Read(B)  B:=B+10000  Write(B) |  |
|  | Read(B)  B:=B+temp  Write(B) |

并发执行（不正确）

|  |  |
| --- | --- |
| T1 | T2 |
| Read(A)  A:=A-10000 |  |
| 丢失T2修改 | Read(A)  temp:=A\*0.1  Write(A)  Read(B) |
| Write(A)  Read(B)  B:=B+10000  Write(B) | 丢失T1修改 |
|  | B:=B+temp  Write(B) |

**11.5 两段锁协议**

保证并行操作可串行性的封锁协议——两段锁协议。

三级封锁协议特点：可以保证事务并行执行的正确性；在事务中申请的所有锁必须在事务结束后才能释放。这意味着，在一个事务执行中必须把锁的申请与释放分为两个阶段，其中第一个阶段是申请并获得锁，也称为扩展阶段，第二个阶段是释放封锁，也称为收缩阶段。

两段锁协议规定在对任何数据进行读写操作之前，事务首先要获得对该数据的封锁，而且在释放一个封锁之后，事务不再获得其他封锁。

举例：

遵守两段锁协议事务的封锁序列：

Slock A … Slock B … Xlock C … Unlock A …Unlock B … Unlock C

不遵守两段锁协议事务的封锁序列：

Slock A … Unlock A… Slock B … Xlock C …Unlock B … Unlock C

思考题：

一级封锁协议是否满足两段协议规定？

二级封锁协议是否满足两段协议规定？

所有遵守两段锁协议的事务，其并行执行的结果一定是正确的。因为如所有并发执行的事务均执行两段锁协议，则对这些事务的所有并行调度策略都是可串行化的。

事务遵守两段锁协议是可串行化调度的充分条件，而不是必要条件

遵守两段锁协议

|  |  |
| --- | --- |
| 事务1 | 事务2 |
| Slock B  读B=2  Y:=B  Xlock B  A:=Y+1  写回A=3  Unlock B  Unlock A | Slock A  等待  …  等待  Slock A  读A=3  Y:=A  Xlock B  B:=Y+1  写回B=4  Unlock B  Unlock A |

不遵守两段锁协议

|  |  |
| --- | --- |
| 事务1 | 事务2 |
| Slock B  读B=2  Y:=B  Unlock B  Xlock B  A:=Y+1  写回A=3  Unlock A | Slock A  等待  …  等待  Slock A  读A=3  Y:=A  Unlock A  Xlock B  B:=Y+1  写回B=4  Unlock B |

**11.6 封锁的粒度**

封锁粒度指事务封锁的数据目标（对象）的大小。关系数据库中的封锁粒度一般有以下几种：属性（值）、属性（值）集、元组、表、物理页面、索引、关系数据库。

粒度大小直接影响数据库系统开销的大小，封锁粒度小系统开销大，封锁粒度大系统开销小。常用的封锁粒度有表和属性。

在封锁粒度的选择上，必须同时考虑封锁开销和并发度两个因素，适当选择封锁粒度。需要处理多个关系的大量元组的用户事务：以数据库为封锁单位；需要处理大量元组的用户事务：以关系为封锁单元；只处理少量元组的用户事务：以元组为封锁单位。

### 11.5.6 课时小结

本章主要内容：

1. 数据库并发问题的产生；

2. 封锁技术解决数据库并发问题；

3. 活锁和死锁及其解决思路；

4. 可串行化的并发调度；

5. 两段锁协议；

6. 封锁的粒度。

### 11.5.7 课外补充

无