

**数据库原理与技术**

**适用班级：软件设计师**

**主 讲：刘琳芳**

**网 址：**[**www.bitpx.com**](http://www.bitpx.com)

**E – Mail：bitpx@163.com**

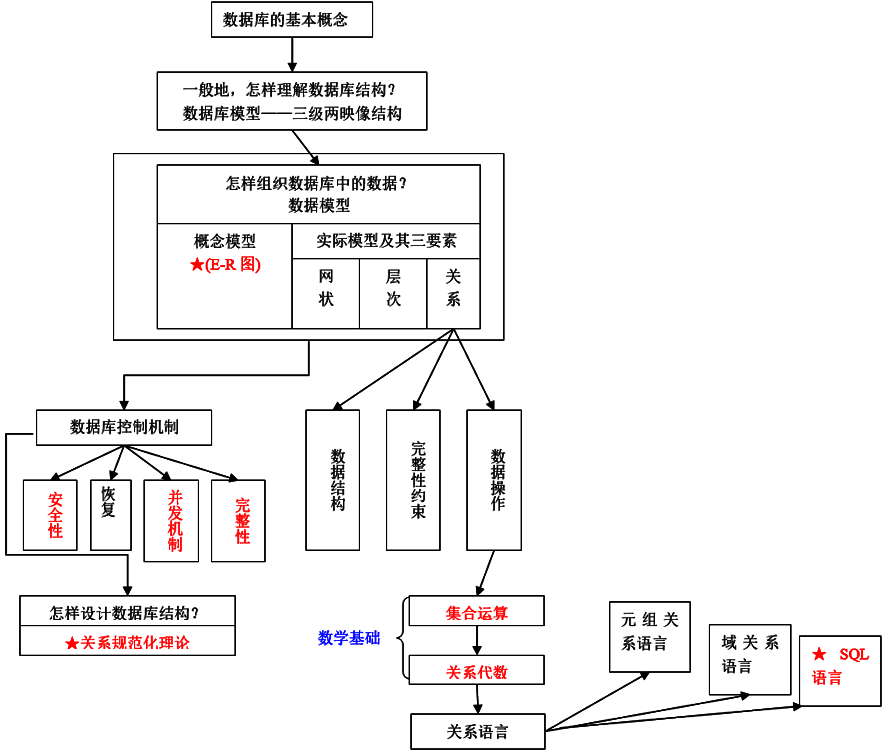
分值说明：早上试题考6-10分

下午试题考15分(数据库分析和设计)

比特培训

贵州·贵阳

本部分知识体系结构示意图



# 数据库管理系统的功能和特征

## 数据库系统概述

### 数据、数据库、数据库管理系统、数据库系统

#### 数据(Data)

数据是数据库中存储的基本对象。数据在大多数人头脑中的第一个反映就是数字，例如93、1000、99.5等。其实数字只是最简单的一种数据，是数据的一种传统和狭义理解。在数据库技术中，数据的种类很多，文本(text)、图形(graph)、图像(image)、音频(audio)、视频(video)、学生的档案记录、货物的运输情况等，凡是计算机能处理的一切元素都称为数据。

#### 数据库(DataBase，简称DB)

数据库，顾名思义，是存放数据的仓库。严格地讲，数据库是长期存储在计算机内、有组织的、可共享的大量数据的集合。数据库中的数据按一定的数据模型组织、描述和存储、具有较小的冗余度、较高的数据独立性和易扩展性，并可为各种用户共享。

#### 数据库管理系统(DataBase Management System，DBMS)

数据库管理系统是位于用户和操作系统之间的一层数据管理软件。数据库管理系统和操作系统一样是计算机的基础软件，也是一个大型复杂的软件系统。比如SqlServer、Oracle、Access、等都是比较常见的数据库管理系统。

数据库管理系统主要功能包括以下几个方面:

##### 数据定义功能

##### 数据组织、存储和管理

##### 数据操纵功能

##### 数据库的事务管理和运行管理

##### 数据库的建立和维护功能

##### 其他功能

#### 数据库系统(DataBase System，DBS)

数据库系统是指计算机系统中引入数据库后的系统，一般由**数据库、硬件、软件(OS、DBMS及开发工具、应用程序)和人员(系统分析员和数据库设计人员、应用程序员、最终用户、数据库管理员)**四个部分组成。应当指出的是，数据库的建立、使用和维护等工作只靠一个DBMS远远不够，还要有专门的人员来完成，这些人被称为数据库管理员(DabaBase Administrator,DBA)。数据库系统如图1.1和图1.2所示。

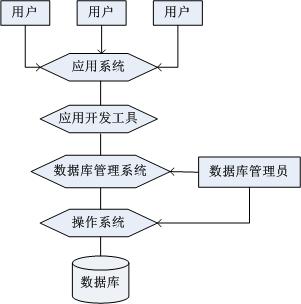


图.1数据库系统

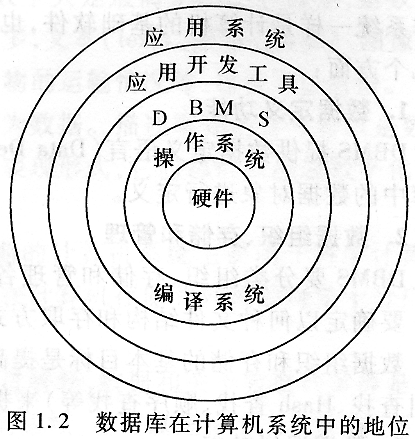


图.2数据库在计算机系统中的地位

#### 数据字典

数据字典中存放各类数据的描述，数据库管理系统对应程序的操作都要通过数据字典来进行(用于描述数据库结构的、特定的、系统级层次上的一种特殊数据库，即描述数据库结构的数据库)。

例如，存在如下图所示的数据库，该数据库中包含两张用户所创建的表格Dept和Emp，如表1.1和表1.2所示。

表1.1 Dept表



表1.2 Emp表



DBMS为了对该数据库进行管理，则必须建立一份针对该数据库的数据字典，其结构可以简略描述如下：

Table级别描述部分：

|  |  |  |  |
| --- | --- | --- | --- |
| TableName | ColumnCount | RowCount | ………… |
| Dept | 3 | 3 | ………… |
| Emp | 4 | 4 | ………… |
| ………… | ………… | ………… | ………… |

列级别描述部分：

|  |  |  |
| --- | --- | --- |
| TableName | ColumnName | ………… |
| Dept | Dept# | ………… |
| Dept | Dname | ………… |
| Dept | Budget | ………… |
| Emp | Emp# | ………… |
| Emp | Ename | ………… |
| Emp | Dept# | ………… |
| Emp | Salary | ………… |
| ………… | ………… | ………… |

数据字典对于每一个DBMS都是必须的，但是我们不一定能够在软件环境中找到，因为对于普通的开发者而言，它是“透明的”。

●数据库系统从广义的角度看，应该由(17)组成。数据字典中存放(18)，数据库管理系统对应用程序的操作都要通过数据字典来进行。

(17)A.数据库、硬件、软件和人员 B.数据库、硬件、数据库管理系统和软件

C.数据库、软件和人员 D.数据库、数据库管理系统和人员

(18)A.数据库管理系统软件 B.数据定义语言DDL C.数据操纵语言DML D.数据库体系结构的描述

## 数据模型

数据模型是我们用来抽象、标识和处理现实世界中的数据和信息的工具，数据模型就是现实世界的模拟。

### 分类的数据模型

#### 概念模型

也称为信息模型，它是按用户的观点来对数据和信息建模，主要用于数据库设计。

#### 逻辑模型

它是按照计算机系统的观点对数据建模，主要用于DBMS的实现。

#### 物理模型

是对数据最低层的抽象，它描述数据在系统内部的表示方式和存取方式，在磁盘或磁带上的存储方式和存取方式，是面向计算机系统的。物理模型的具体实现是DBMS的任务，数据库设计人员要了解和选择物理模型，一般用户不必考虑物理级的细节。

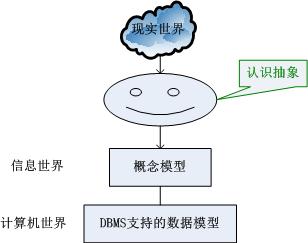


图1.3现实世界中客观对象的抽象过程

### 数据模型的组成要素

数据模型的组成要素包括三个部分：**数据结构、数据操作、数据约束。**

#### 数据结构

数据模型中的数据结构主要描述数据的类型、内容、性质以及数据间的联系等。数据结构是数据模型的基础，数据操作和约束都建立在数据结构上。不同的数据结构具有不同的操作和约束。

#### 数据操作

数据模型中数据操作主要描述在相应的数据结构上的操作类型和操作方式。

#### 数据的完整性约束

数据模型中的数据约束主要描述数据结构内数据间的语法、词义联系、他们之间的制约和依存关系，以及数据动态变化的规则，以保证数据的正确、有效和相容。

### 概念数据模型

概念模型用于信息世界的建模，是现实世界到信息世界的第一层抽象，是数据库设计人员进行数据库设计的有力工具，也是数据库设计人员和用户之间进行交流的语言，因此概念模型一方面应该具有较强的语义表达能力，能够方便、直接地表达应用中的各种语义知识，另一方面它还简单、清晰、易于用户理解。

#### 信息世界中的基本概念

##### 实体(Entity)

**客观存在并相互区别的事物称为实体。**实体可以是具体的人、事、物，也可以是抽象的概念或联系，例如，一个职工、一个学生、一个部门、一门课、学生的一次选题、部门的一次订货、老师与院系的工作关系(即某位老师在某院系工作)等都是实体。

##### 属性(Attribute)

实体所具有的某一特性称为属性。一个实体可以由若干个属性来刻画。例如，学生实体可以由学号、姓名、性别、出生年月、所在院系、入学时间等属性组成。

##### 码(Key)

唯一标识实体的属性集称为码。例如学号是学生实体的码。

##### 域(Domain)

属性的取值范围称为该属性的域。例如，学号的域为8位整数，姓名的域为字符串集合，学生年龄的域为整数，性别的域为(男，女)。

##### 实体型(Entity Type)

具有相同属性的实体必然具有共同的特征和性质。用实体名及其属性名集合来抽象和刻画同类实体，称为实体型。例如，学生(学号，姓名，性别，出生年月，所在院系，入学时间)就是一个实体型。

##### 实体集(Entity Set)

同一个类型实体的集合称为实体集。例如，全体学生就是一个实体集。

##### 联系(Relationship)

在现实世界中，事物内部以及事物之间是有联系的，这些联系在信息世界中反映为实体(型)内部的联系和实体(型)之间的联系。

实体内部的联系通常是指组成实体的各属性之间的联系；实体之间的联系通常是指不同实体集之间的联系。

#### 概念模型的一种表示方法：实体-联系方法

概念模型的表示方法很多，其中最著名最为常用的是P.P.S.Chen于1976年提出的实体—联系方法(Entity-Relationship Approach)。该方法用E-R图(E-R Diagram)来描述现实世界的概念模型，E-R方法也称为E-R模型。

E-R图提供了表示实体型、属性和联系的方法：

* + 实体型：用矩形表示，矩形框内写明实体名。
  + 属性：用椭圆形表示，并用无向边将其与相应的实体型连接起来。

例如，学生实体具有学号、姓名、性别、出生年月、系、入学时间等属性，用E-R图表示如图1.4所示。

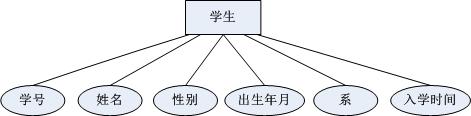


图.4学生实体和属性

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

需要注意的是，如果一个联系具有属性，则这些属性也要用无向边与该联系连接起来。

#### 两个实体型之间的联系

两个实体型之间的联系可以分为三种：

##### 一对一联系(1：1)

如果对于实体集A中的每一个实体，实体集B中至多有一个(也可以一个都没有)实体与之联系，反之亦然，则称实体集A与实体集B具有一对一联系，记为1：1。

例如：学校里面，一个班级只有一个正班长，而一个班长只在一个班中任职，则班级与班长之间具有一对一的联系。

##### 一对多联系(1：n)

如果对于实体集A中的每一个实体，实体集B中有n个实体(n≥0)与之联系，反之，对于实体集B中的每一个实体，实体集A中至多只有一个实体与之联系，则称实体集A与实体集B具有一对多联系，记为1：n。

例如：学校里面，一个班级有若干名学生，而每个学生只在一个班级中学习，则班级与学生之间具有一对多的联系。

##### 多对多联系(m：n)

如果对于实体集A中的每一个实体，实体集B中有n个实体(n≥0)与之联系，反之，对于实体集B中的每一个实体，实体集A中也有m个实体(m≥0)与之联系，则称实体集A与实体集B具有多对多联系，记为m：n。

例如：学校里面，一门课程同时有若干名学生选修，而每个学生可以同时选修多门课程，则课程与学生之间具有多对多的联系。

实际上，一对一联系是一对多联系的特例，而一对多联系是多对多联系的特例。

可以用图形来表示两个实体型之间的这三种联系，如图1.5所示。

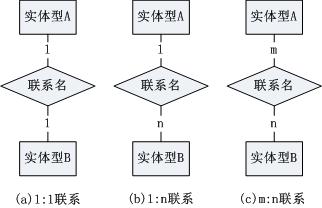


图1.5两个实体型之间的三类联系

#### 两个以上的实体型之间的联系

一般地，两个以上的实体型之间也存在着一对一、一对多、多对多联系。

例如，对于课程、教师与学生实体型之间的联系。一门课程可有一个教师讲授，每一个教师讲授多门课程，一个学生选修多门课程，一门课程可让多个学生选修。教师与学生没有直接联系，只能通过课程产生间接联系，如图1.6所示。

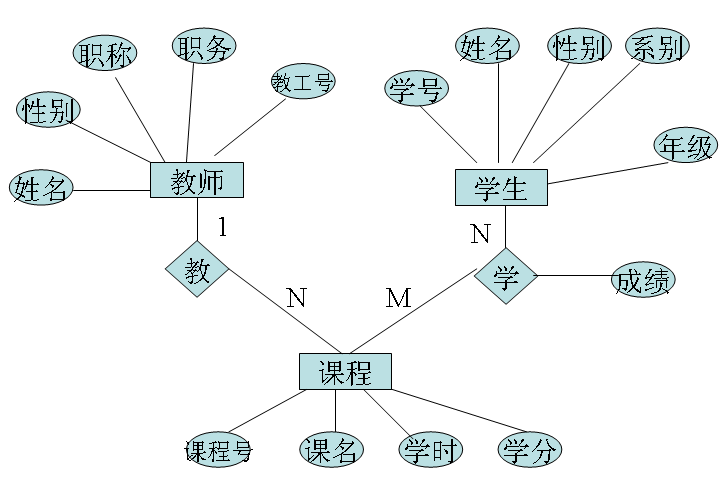


图.63个实体型之间的联系示例

#### 单个实体型内的联系

同一个实体集内的各实体之间也可以存在一对一、一对多、多对多的联系。例如，职工实体型内部具有领导和被领导的联系，即某一职工(干部)“领导”若干名职工，而一个职工仅被另外一个职工(领导)直接领导，因此这是一对多的联系，如图1.7所示。

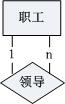
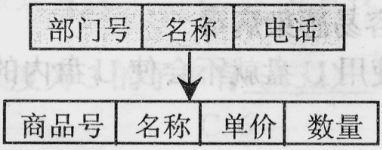


图1.7单个实体型之间一对多的联系示例

●某商场的部门和商品两个实体之间的关系如下图所示。假设每个部门负责销售若干种商品，每种商品只能由一个部门负责销售，那么部门和商品之间存在着\_\_(14)\_\_的联系。



(14)A.一对一 B.一对多 C.多对多 D.一对多对多

### 逻辑数据模型

目前，数据库领域中的逻辑数据模型有：

层次模型(Hierarchical Model)

网状模型(Network Model)

* + 关系模型(Relational Model)

面向对象模型(Object Oriented Model)

对象关系模型(Object Relational Model)

其中层次模型和网状模型统称为非关系模型。非关系模型的数据库在20世纪70年代至80年代初非常流行，在数据库系统产品中占据了主导地位，现在已逐渐被关系模型的数据库系统取代。我们主要学习关系模型的数据库。

### 物理数据模型

又称为物理模型，是面向计算机物理表示的模型，描述了数据在储存介质上的组织结构，它不但与具体的DBMS有关，而且还与操作系统和硬件有关。每一种逻辑数据模型在实现时都有其对应的物理数据模型。DBMS为了保证其独立性与可移植性，大部分物理数据模型的实现工作由系统自动完成，而设计者只设计索引、聚集等特殊结构。

## 数据库系统结构

### 数据库系统的三级模式结构

#### 模式

模式即概念模式，是数据库中全体数据的逻辑结构和特征的描述，是所有用户的公共数据视图。一个数据库只有一个概念模式。

#### 外模式

外模式也称子模式或用户模式，是数据库用户(包括程序员和最终端用户)能够看见和使用的局部数据的逻辑结构和特征的描述，是数据库用户的数据视图，外模式主要描述用户视图的各个记录的组成、相互关系、数据项的特征、数据的安全性和完整性约束性条件。是与某一应用有关的数据逻辑表示。一个数据库可以有多个外模式。但一个应用只能使用一个外模式。

#### 内模式

内模式是数据物理结构和存储方式的描述，是数据在数据库内部的表示方式。一个数据库只有一个内模式。内模式定义的是存储记录的类型、存储域的表示、存储记录的物理顺序指引元、索引和存储路径等数据的存储组织。

与此相对应，人们数据库系统划分为三个抽象级：用户级，概念级，物理级。

##### 用户级数据库

用户级数据库对应于外模式，是最接近于用户的一级数据库，是用户看到和使用的数据库，又称用户视图。用户级数据库主要由外部记录组成，用户的所有操作都是针对用户视图进行的。

##### 概念级数据库

概念级数据库对应于概念模式，介于用户级和物理级之间，是所有视图的最小并集，是数据库管理员看到和使用的数据库，又称DBA视图。

##### 物理级数据库

物理级数据库对应于内模式，是数据库的的底层表示，它描述数据的实际存储组织，是最接近物理层的级，有称内部视图，最接近于物理存储的级。

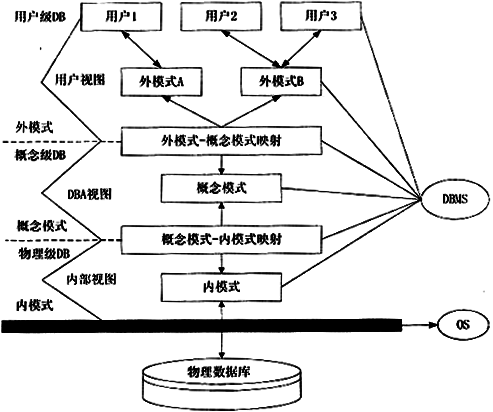


图.8数据库系统的三级模式结构

### 数据库的二级映像功能与数据独立性

#### 外模式/模式映像

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

#### 模式/内模式映像

数据库中只有一个模式，也只有一个内模式，所以模式/内模式映像是唯一的，它定义了数据全局逻辑结构与存储结构之间的对应关系。当数据的物理存储改变时，应用程序不需改变,物理独立性是指用户的应用程序与存储在磁盘上的数据库中的数据是相互独立的。

#### 聚簇索引

也叫簇类索引、聚集索引、聚类索引，簇集索引，是一种对磁盘上实际数据重新组织以按指定的一个或多个列的值排序。每张表只能建一个聚簇索引，但该索引可以包含多个列(组合索引)，就像电话簿按姓氏和名字进行组织一样。汉语字典也是聚簇索引的典型应用，在汉语字典里，索引项是字母+声调，字典正文也是按照先字母再声调的顺序排列。

聚簇索引对于那些经常要搜索范围值的列特别有效。使用聚簇索引找到包含第一个值的行后，便可以确保包含后续索引值的行在物理相邻。例如，如果应用程序执行的一个查询经常检索某一日期范围内的记录，则使用聚集索引可以迅速找到包含开始日期的行，然后检索表中所有相邻的行，直到到达结束日期。这样有助于提高此类查询的性能。同样，如果对从表中检索的数据进行排序时经常要用到某一列，则可以将该表在该列上聚簇(物理排序)，避免每次查询该列时都进行排序，从而节省成本。

●(2018年11月).数据库系统中的视图、存储文件和基本表分别对应数据库系统结构中的51。

(51) A.模式、内模式和外模式 B外模式、模式和内模式

C.模式、外模式和内模式 D.外模式、内模式和模式

●(2017年11月)采用三级结构/两级映像的数据库体系结构，如果对数据库的一张表创建聚簇索引改变的是数据库的 (51) 。

(51) A.用户模式 B.外模式 C.模式 D.内模式

●(2016.5)数据的物理独立性和逻辑独立性分别是通过修改(51)来完成的。

(51)A.外模式与内模式之间的映像、模式与内模式之间的映像

B.外模式与内模式之间的映像、外模式与模式之间的映像

C.外模式与模式之间的映像、模式与内模式之间的映像

D.模式与内模式之间的映像、外模式与模式之间的映像

### 大数据

1.大数据产生的背景

大数据(Big Data) 是指"无法用现有的软件工具提取、存储、搜索、共享、分析和处理

的海量的、复杂的数据集合"。大数据产生的背景主要有以下4 个方面。

(1)数据来源和承载方式的变革。由于物联网、云计算、移动互联网等新技术的发展，用户在线的每一次点击、每一次评论、每-个视频点擂，都是大数据的典型来源:而遍布地球各个角落的于机、PC 、平板电脑及传感器成为数据来源和承载方式。可见，只有大连接与大交互，才有大数据。

(2) 全球数据量出现爆炸式增长。由于视频监控、智能终端、网络商店等快速普及，使得全球数据量出现爆炸式增长。许多研究表明，未来数年数据量会呈现指数增长o 根据麦肯锡全球研究院(MGO 估计，全球企业2010 年在硬盘上存储了超过7EB (IEB 等于109GB) 的新数据，而消费者在PC 和笔记本等设备上存储了超过6EB 的新数据。据IDC( Intemet Data Center)预测，到2020 年，全球以电子形式存储的数据量将达32ZB。

(3) 大数据己经成为一种自然资源。许多研究者认为:大数据是"未来的新石油"，己成为二种新的经济资产类别。一个国家拥有数据的规模、活性及解释运用的能力将成为综合国力的重要组成部分。

(4) 大数据日益重要，不被利用就是成本。大数据作为一种数据资产当仁不让地成为现代商业社会的核心竞争力，不被利用就是企业的成本。因为，数据资产可以帮助和指导企业对整个业务流程进行有效的运营和优化，帮助企业做出最明智的决策。

2. 大数据的特征

大数据(Big Data) 是指"无法用现有的软件工具提取、存储、搜索、共享、分析和处理的海量的、复杂的数据集合"。业界通常用"4V" 来概括大数据的特征。

大量化(Volume) 指数据体量巨大。随着IT 技术的迅猛发展，数据量级己从TB (1012 字节)发展至PB 乃至ZB ，可称海量、巨量乃至超量。当前，典型个人计算机硬盘的容量为TB量级，而一些大企业的数据量己经接近EB 量级。

多样化(Variety〉指数据类型繁多。相对于以往便于存储的以文本为主的结构化数据，非结构化数据越来越多，包括网络日志、音频、视频、图片、地理位置信息等，这些多类型的数据对数据的处理能力提出了更高的要求。

价值密度低(Value) 指大量的不相关信息导致价值密度的高低与数据总量的大小成反比。以视频为例，一部一小时的视频，在连续不间断的监控中，有用数据可能仅有一两秒。因此，如何通过强大的机器算法更迅速地完成数据的价值"提纯"，如何对未来趋势与模式的可预测分析、深度复杂分析(机器学习、人工智能VS 传统商务智能咨询、报告等)，成为目前大数据背景下待解决的难题。

快速化(Velocity )指处理速度快。大数据时代对时效性要求很高，这是大数据区分于传统数据挖掘的最显著特征。因为，在大数据环境下数据流通常为高速实时数据流，而且需要快速、持续的实时处理:处理工具也在快速演进，软件工程及人工智能等均可能介入。

3. 理解大数据

大数据不仅仅是指海量的信息，更强调人类对信息的筛选、处理，保留有价值的信息，即让大数据更有意义，挖掘其潜在的"大价值"这才是对大数据的正确理解。为此，有许多问题需要研究与解决。

(1)高并发数据存取的性能要求及数据存储的横向扩展问题。目前，多从架构和并行等方面考虑解决。

(2) 实现大数据资源化、知识化、普适化的问题，解决这些问题的关键是对非结构化数据的内容理解。

(3) 非结构化海量信息的智能化处理问题，主要解决自然语言理解、多媒体内容理解、机器学习等问题。

大数据时代主要面临三大挑战:软件和数据处理能力、资源和共享管理以及数据处理的可信力。

(1) 软件和数据处理能力。应用大数据技术，提升服务能力和运作效率，以及个性化的服务，比如医疗、卫生、教育等部门。

(2) 资源和共享管理。应用大数据技术，提高应急处置能为和安全防范能力。

(3) 数据处理的可信力。需要投资建立大数据的处理分析平台，实现综合治理、业务开拓

4. 大数据产生的安全凤险

2012 年瑞士达沃斯论坛上发布的《大数据大影响》报告称，数据己成为-种新的经济资产类别，就像货币或黄金一样。因此，也带来了更多安全风险。

(1)大数据成为网络攻击的显著目标。在互联网环境T ，大数据是更容易被"发现"的大目标。这些数据会吸引更多的潜在攻击者，如数据的大量汇集，使得黑客成功攻击→次就能获得更多数据，无形中降低了黑客的攻击成本，增加了"收益率"。

(2) 大数据加大了隐私泄露风险。大量数据的汇集不可避免地加大了用户隐私泄露的风险，因为数据集中存储增加了泄露风险;另外，二些敏感数据的所有权和使用权并没有明确界定，很多基于大数据的分析都未考虑到其中涉及的个体隐私问题。

(3)大数据威胁现有的存储和安防措施。大数据存储带来新的安全问题，数据大集中的后果是复杂多样的数据存储在-起，很可能出现将某些生产数据放在经营数据存储位置的情况，致使企业安全管理不合规。大数据的大小也影响到安全控制措施能否正确运行。如果安全防护于段的更新升级速度无法跟上数据量非线性增长的步伐，就会暴露大数据安全防护的漏洞。

(4) 大数据技术成为黑客的攻击于段。在企业用数据挖掘和数据分析等大数据技术获取商业价值的同时，黑客也在利用这些大数据技术向企业发起攻击。黑客会最大限度地收集更多的有用信息，例如社交网络、邮件、微博、电子商务、电话和家庭住址等信息，大数据分析使黑客的攻击更加精准。

(5) 大数据成为高级可持续攻击的载体。传统的检测是基于单个时间点进行的基于威胁特征的实时匹配检测，而高级可持续攻击(APT) 是二个实施过程，无法被实时检测。此外，大数据的价值低密度性使得安全分析工具很难聚焦在价值点上，黑客可以将攻击隐藏在大数据中，给安全服务提供商的分析制造很大困难。黑客设置的任何二个会误导安全厂商目标信息提取和检索的攻击，都会导致安全监测偏离应有方向。

(6) 大数据技术为信息安全提供新支撑。当然，大数据也为信息安全的发展提供了新机遇。大数据正在为安全分析提供新的可能性，对于海量数据的分析有助于信息安全服务提供商更好地刻画网络异常行为，从而找出数据中的风险点。对实时安全和商务数据结合在→起的数据进行预防性分析，可识别钓鱼攻击，防止诈骗和阻止黑客入侵。网络攻击行为总会留下蛛丝马迹，这些痕迹都以数据的形式隐藏在大数据中，利用大数据技术整合计算和处理资源有助于更有针对性地应对信息安全威胁，有助于找到攻击的源头。

# 关系模型

## 关系数据结构定义及形式

从用户的观点看，关系模型由一组关系组成。每个关系的数据结构是一张规范化的二维表。现以学生登记表为例，介绍关系模型中的一些术语。

例如，表2.1的关系可描述为：学生(学号，姓名，年龄，性别，系名，年级)。

表2.1关系模型的数据结构

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 学号 | 姓名 | 年龄 | 性别 | 系名 | 年级 |
| 2005004 | 小aaa | 19 | 女 | 计算机系 | 2005 |
| 2005005 | 小bbb | 18 | 男 | 法学系 | 2005 |
| 2005006 | 小ccc | 20 | 女 | 数学学院 | 2005 |
| … | … | … | … | … | … |

关系模型要求关系必须是规范化的，即要求关系必须满足一定的规范条件，这些规范条件中最基本的一条就是，**关系的每一个分量必须是一个不可分的数据项，也就是说，不允许表中还有表。**表2.2中工资和扣除是可分的数据项，工资又分为基本工资、津贴和职务工资，扣除又分为房租和水电。因此表2.2的表就不符合关系模型要求，但可以转化为表2.3所示的二维表。

表2.2一个工资表(表中有表)实例

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 职工编号 | 姓名 | 职称 | 工资 | | | 扣除 | | 实发 |
| 基本 | 补贴 | 奖励 | 房租 | 水电 |
| 0001 | 刘明 | 教授 | 5000 | 1000 | 1000 | 800 | 200 | 6000 |
| … | … | … | … | … | … | … | … | … |

表2.3一个工资表实例(符合关系模型规范)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 职工编号 | 姓名 | 职称 | 基本工资 | 补贴 | 奖励 | 扣除房租 | 扣除水电 | 实发 |
| 0001 | 刘明 | 教授 | 5000 | 1000 | 1000 | 800 | 200 | 6000 |
| … | … | … | … | … | … | … | … | … |

关系(Relation)：一个关系对应通常说的一张表，如图2.1所示的这张学生登记表；关系(实际上就是一张二维表)可以有三种类型：基本关系(通常又称为基本表或基表)、查询表和视图表。基本表是实际存在的表，它是实际存储数据的逻辑表示。查询表是查询对应的表。视图表是由基本表或其他视图表导出的表，是虚表，不对应实际存储数据。

元组(Tuple)：表中的一行即为一个元组；

属性(Attribute)：表中的一列即为一个属性，给每一个属性起一个名称即属性名。如这张表有6列，对应6个属性(学号，姓名，年龄，性别，系名和年级);

侯选码：若关系中的某一属性(组)的值能唯一的标识一个元组，则称该属性组为侯选码(侯选键)。若一个关系有多个候选码，则选定其中一个为主码(主键)。

主属性：能够充当主码的诸属性称为主属性。不包含在任何候选码中的属性称为非码属性(非主属性)。在最简单的情况下，候选码只包含一个属性。在最极端的情况下，关系模型的所有属性组是这个关系模式的候选码，称为全码。

域(Domain)：属性的取值范围，如人的年龄一般在1~150岁之间，大学生年龄属性的域是(14~38)，性别的域是(男，女)，系名的域是一个学校所有系名的集合；

分量：元组中的一个属性值；

关系模式的定义——“型与值”的定义

关系——对现实世界中某一实体及其拥有联系的某种取值——值；

关系模式——对关系的描述与抽象——型

一个关系模式应该是一个五元组。它可以形式化地完整表示为：R(U，D，DOM，F)。

其中R为关系名，U为组成该关系的属性名集合(U={ID,学号，姓名，班级，性别，身份，密码，……，是否交过卷})，D为属性组U中属性所来自的域，DOM为属性向域的映像集合，F为属性间数据的依赖关系集合(如上面属性中：学号→姓名)。

关系模式通常可以简记为R(A1，A2，…An)，其中R为关系名，A1，A2，….，An为属性名(如上面的表可以简写为：R(ID，学号，姓名，班级，性别，身份，密码，…，是否交过卷))。

关系数据库：在一个给定的现实世界领域中，相对于所有实体以及实体之间的联系的关系集合构成一个关系数据库。

可以把关系和现实生活中的表格所使用的术语做一个粗略的对表(如表2.4所示)。

表2.4术语对比

|  |  |
| --- | --- |
| 关系术语 | 对应普通表格术语 |
| 关系名 | 表名 |
| 关系模式 | 表头(表格的列描述) |
| 关系 | (一张)二维表 |
| 元组 | 记录或行 |
| 属性 | 列 |
| 属性名 | 列名 |
| 属性值 | 列值 |
| 分量 | 一条记录中的一个列值 |
| 非规范关系 | 表中有表(大表中嵌套小表) |

●(2015.11)若关系R(H，L，M，P)的主键为全码(ALL-key)，则关系R的主键应(51)。

(51)A.为HLMP B.在集合{H，L，M，P}中任选一个

C.在集合{HL，HM，HP，LM，LP，MP}中任选一个 D.在集合(HLM，HLP，HMP，LMP)中任选一个

## 关系操作

关系模型中常用的关系操作包括查询(Query)操作和插入(Insert)、删除(Delete)、修改(Update)操作四个部分。

关系操作的特点是集合操作方式，即操作的对象和结果都是集合。

## 关系的完整性

关系模型的完整性规则是对关系的某种约束条件。也就是说关系的值随着时间变化时应该满足一些约束条件。这些约束条件实际上是现实世界的要求。任何关系在任何时刻都要满足这些语义约束，以防止数据库出现存储异常问题，维护数据库数据完整性等。

关系模型中有三类完整性约束：实体完整性、参照完整性和用户自定义完整性。

#### 实体完整性(Entity Integrity)

**规则：**若属性(指一个或一组属性)A是基本关系R(一个二维表中)的主属性，则A不能取空值且不能重复。

注意：实体完整性是针对单张实体表而言的。例如在学生信息表中，学生关系模型表示为：

学生(学号，姓名，性别，专业号，年龄)

学号是学生实体表的主属性(在关系模型中用下划线表示)，所以学号一列的值不能为空、不能相同。

#### 参照完整性(Referential Integrity)

现实世界中的实体之间往往存在某种联系，在关系模型中实体及实体间的联系都是用关系来描述的。这里我们通过一个例子来学习参照完整性。

[例1] 学生实体和专业实体可以用下面的关系来表示，其中主码用下划实线表示，外码用下划线虚表示：

学生(学号，姓名，性别，专业号，年龄)

专业(专业号、专业名)

以上两个关系表，一个是学生表，存储学生的相关信息，另外一个专业表，存储专业相关的信息。在学生表中，属性“学号”能唯一地标识学生表中的一个学生(对应一行数据，即一个元组)，故“学号”为学生表中的主键(又称主码、主属性等)。而在专业表中，属性“专业号”也能唯一地标识一行专业数据，故“专业号”为专业表中的主键。

在以上两张表中存在着属性的引用，即学生关系引用了专业关系的主码“专业号”。从实际出发，学生关系中的“专业号”值必须是确实存在的专业表的专业号中，即专业关系中有该专业的记录。也就是说，学生关系中的某个属性(专业号)的取值需要参照专业关系中的属性(专业号)取值，这就形成了参照关系。在这一参照关系中，“专业号”是专业关系的主键，所以专业表为又被称为主键表，而“专业号”在学生表中被称为外键(用下划虚线表示)，因此学生表被称为外键表。

**规则：**若属性(或属性组)F是基本关系R的外码，它与基本关系S的主码K(属性或属性组)相对应，则对于R中的每个元组在F上的值必须为：

* + 或者取空值(F中的每个属性值均为空值)；
  + 或者等于S中某个元组的主键码值。

例如，对于例1中学生关系中每个元组的“专业号”属性只能取下面的两类值：

##### 空值，表示尚未给学生分配专业；

##### 非空值，这时该值必须是专业关系中某个元组的“专业号”值，表示该学生不可能分配到一个不存在的专业中。即外键的值必定存在于主键表的主键值中。

注意：不仅两个或两个以上的关系间可以存在引用关系，同一关系内部属性间也可能存在引用关系。如：学生(学号，姓名，性别，专业号，年龄，班长)中，“学号”属性是主码，“班长”属性表示该学生所在班级的班长的学号，它引用了本关系“学号”属性，即“班长”必须是确定存在的学生的学号。

#### 用户自定义完整性(User-defined Integrity)

任何关系数据库系统都应该支持实体完整性和参照完整性，除此，还应该支持用户自定义的完整性。例如在学生关系中，姓名不能取值为空，性别只能取值为男或女，年龄取值范围为5~50岁等都为用户自定义完整性。

●数据库管理技术是在文件系统的基础上发展起来的。数据模型的三要素是数据结构、数据操作和\_\_(21)\_\_。建立数据库系统的主要目标是减少数据冗余，提高数据的独立性，并集中检查\_\_(22)\_\_。

供选择的答案：

(21)A．数据安全 　B．数据兼容 C．数据约束条件 D．数据维护

(22)A．数据操作性　B．数据兼容性 C．数据完整性　 D．数据可维护性

## E-R图向关系数据库转换

概念模型是从用户的角度出发而对信息世界进行数据建模，其中主要的一个环节是经过对信息世界问题的分析而画出其对应的E-R图。E-R图是与具体的逻辑模型无关的，而我们在数据库设计的过程当中，必要的一个步骤是把E-R图转换为一种逻辑模型，然后再进行数据库的物理模型设计。本节讲解概念模型中的E-R图向逻辑模型中的关系数据库的转换过程，因为如今关系数据库在逻辑模型中占主流地位。

根据E-R图中实体型之间的关系，E-R图向关系数据库的转换遵循以下原则：

#### 一个实体型转换为一个关系模型，实体的属性就是关系的属性，实体的键就是关系的键。

如图2.1是一个实体图(假设属性1为主属性，即为码)，假设主属性为属性1，则将其转换为关系模式如下：



图2.1实体型A

对应的关系模式为：实体型A(属性1，属性2，属性3)

#### 一个联系是否要转换为一个关系模式，这取决于与该联系相关的实体之间的对应关系，不同的对应关心，遵循不同的转换原则：

##### 两个实体型之间的关系为1：1关系

这种情况下，可把相关的实体型合并成一个实体型，然后再转换为对应的关系模式，实体型之间的联系不用转换为关系模式。即是1：1的对应关系至少需要一张关系表就够了。

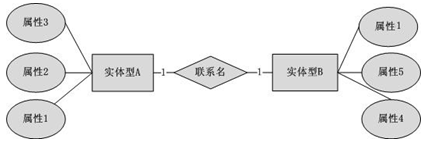


图2.2对1联系图

在图2.2中，实体型A和实体型B之间的对应关系为1：1，这种情况下中间的联系名不用转换为关系模式，而直接把实体型A和实体型B合并且转换为一个关系模式，这个关系模式名可为实体型A，也可以为实体型B，还可起名为其他同等意义的名字。

当然也可以把两个实体型单独转换为对应的关系模式，这种情况下有两个关系模式，但是由于这个两个关系模式的主键是一样的，所以可以合并为一个关系模式。故图2.2对应的关系模式为(假设此E-R图中属性1为主属性)：

实体型A(属性1，属性2，属性3，属性4，属性5)

**总之，把实体型之间对应关系为1：1的E-R图转换为对应的关系模式的时候，至少需要一个关系模式，即至少需要一张二维关系表，联系不必转换为关系模式。**

##### 两个实体型之间的关系为1：m关系

如图2.3所示，实体型A与实体型B的关系为1：m的关系，假设“属性1”为实体型A的码，“属性4”为实体型B的码。

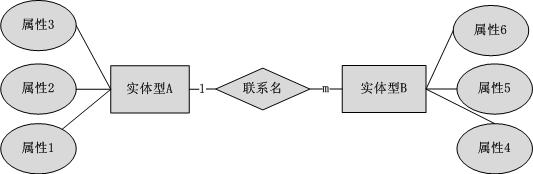


图2.31对多联系图

这种情况下，实体型A和实体型B分别单独转换为两个关系模式。除此，由于要体现两个关系模式之间的“1：m”的参照关系，所以还要把“1：m”联系中“1”端的主键复制到“m”一端的关系模式里面去并作为外键，而对于这个参照关系来说，实体型B转换的关系模式称为外键表，而实体型A转换的关系表称为主键表。**至于两个实体型之间的联系无需转为为关系模式。所以在两个实体型之间的对应关系为1对多的情况下，把E-R转换为关系模式至少需要2张关系二维表。**如图2.3对应的关系模式为：

* + 实体型A(属性1，属性2，属性3)
  + 实体型B(属性4，属性5，属性6，属性1)

在以上的关系模式“实体型B”中，**“属性4”是主键，“属性1”是外键，**参照到主键表“实体型A”表的主键“属性1”。

##### 两个实体型之间的关系为m：n关系

如图2.4所示，实体型A与实体型B的关系为m：n的关系，假设“属性1”为实体型A的码，“属性4”为实体型B的码。联系自己也有一个属性“属性7”。

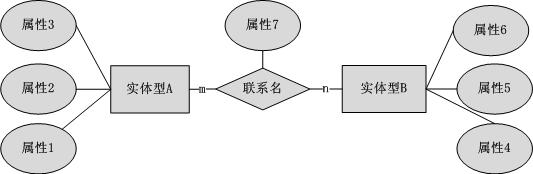


图2.4多对多联系图

由第(1)、(2)可知，1对1、1对多的E-R图可轻松地转换为对应的关系模式。在关系数据库中，目前也只能实现二维表之间关系为1对1和一对多这两种情况，不能直接实现多对多的关系。故m：n的实体关系图不能直接转换为关系模式，转换以后的关系模式如下：

* + 实体型A(属性1，属性2，属性3)
  + 实体型B(属性4，属性5，属性6)
  + 联系名(属性1，属性4，属性7)

可以看出，“联系名”这个关系模式除了自己的属性“属性7”外，还新加入了“实体型A”和“实体型B”的主键，所以有三个属性，而新加入的属性组合(属性1，属性4)就构成了“联系名”这个关系模式的主键，或者是主键的一部分，而“属性1”单个属性在“联系名”关系模式中是外键，对应到“实体型A”这个主键表的主键“属性1”，“属性4”也是对应到主键表“实体型B”主键的外键，同时在这个两个一对多的参照关系中，“实体型A”和“实体型B”这两个关系二维表为主键表，而“联系名”为外键表。

由此可见，一个“多对多”的E-R图转换为对应的关系模式，至少需要三个二维关系表，E-R图中的联系要转换为相应的关系模式。

●(2011.11)E-R图转换为关系模型时，对于实体E1与E2间的多对多联系，应该将(56)。

(56)A.E1的码加上联系上的属性并入E2 B.E1的码加上联系上的属性独立构成一个关系模式

C.E2的码加上联系上的属性独立构成一个关系模式D.E1与E2码加上联系上的属性独立构成一个关系模式

●(2012.5)E-R模型向关系模型转换时，三个实体之间多对多的联系m:n:p应该转换为一个独立的关系模式，且该关系模式的关键字由(49) 组成。

(49)A．多对多联系的属性 B．三个实体的关键字C．任意一个实体的关键字 D．任意两个实体的关键字

●(2015.5)部门、员工和项目的关系模式及它们之间的E-R图如下所示，其中，关系模式中带实下划线的属性表示主键属性。图中：

部门(部门代码，部门名称，电话)

员工(员工代码，姓名，部门代码，联系方式，薪资)

项目(项目编号，项目名称，承担任务)



若部门和员工关系进行自然连接运算，其结果集为(54)元关系。由于员工和项目关系之间的联系类型为(55)，所以员工和项目之间的联系需要转换成一个独立的关系模式，该关系模式的主键是(56)。

(54)A.5 B.6 C.7 D.8

(55)A.1对1 B.1对多 C.多对1 D.多对多

(56)A.(项目名称，员工代码)B.(项目编号，员工代码)C.(项目名称，部门代码)D.(项目名称，承担任务)

## 关系数据模型的操作

### 传统集合运算

#### 并

设关系R和S具有相同的关系模式，R和S的并是由属于R或属于S的元组组成的集合，记为R∪S。形式定义如下：R∪S≡{t|t∈R∨t∈S}式中t是元组变量(下同)。显然，R∪S=S∪R。

#### 差

关系R和S具有相同的关系模式，R和S的差是由属于R但不属于S的元组组成的集合，记为R-S。形式定义如下：R－S≡{t|t∈R∧tS}

#### 交

关系R和S具有相同的关系模式，R和S的交是由既属于R又属于S的元组组成的集合，记为R∩S。形式定义如下：R∩S≡{t|t∈R∧t∈S}

#### 笛卡儿积

设关系R和S元数(实际上就是表中的字段个数，即是属性的个数)分别为r和s。R和S的笛卡儿积是一个r+s元的元组集合，每个元组的前r个分量来自R的一个元组，后s个分量来自S的一个元组，记为R×S。形式定义如下：R×S≡{t|t=<tr,ts>∧tr∈R∧rs∈S}

若R有m个元组，S有n个元组，则R×S有m\*n个元组。

例如：设关系R和S如图2.5所示，则R∪S与R∩S如图2.6所示，R-S和S-R如图2.7所示，R×S如图2.8所示。

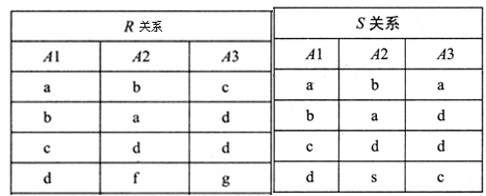


图2.5关系R和S

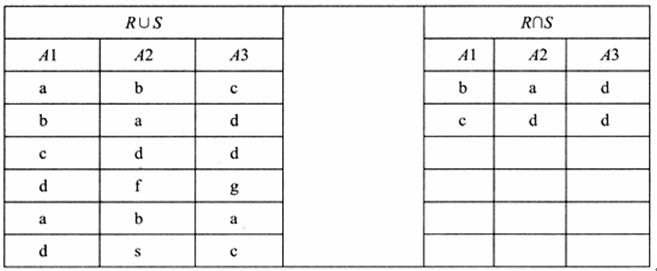


图2.6并集与交集

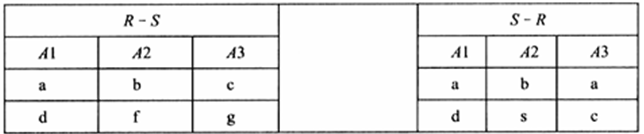


图2.7集合之差

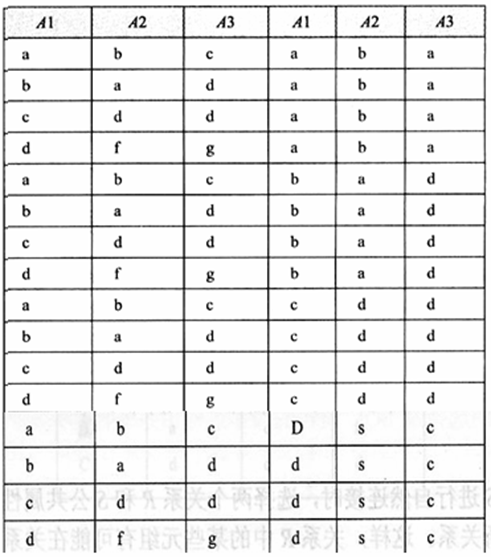


图2.8笛卡尔积

●若课程域D1={数据库，操作系统，面向对象}，教师域D2=｛王强，刘亦｝，学生域D3＝｛刘海华，张刚，昊琨，梁谦｝，则D1×D2×D3\_\_(45)\_\_。在关系代数运算中，D1×D2的结果有\_\_(46)\_\_个元组。

(45)A.可以用一个二维表来表示 B.不能用一个二维表来表示

C.不是一个集合也不是一个关 D.不是一个集合但它是一个关系

(46)A. 3　 B. 4　 　C. 5　 　D. 6

### 关系运算

关系运算主要有投影、选择、连接、除法和外连接。关系运算以集合运算为理论基础。

#### 选择(selection)

选择操作在关系R中选择满足给定条件的所有元组，记为：σF(R) ≡{t|t∈R∧F(t)=true}

其中F表示选择条件，是一个逻辑表达式(逻辑运算符+算术表达式)。选择运算是从行的角度进行的运算，而投影是从列得角度来进行运算。例如：对关系R作选择运算：σA1=b(R)，如图2.9所示。

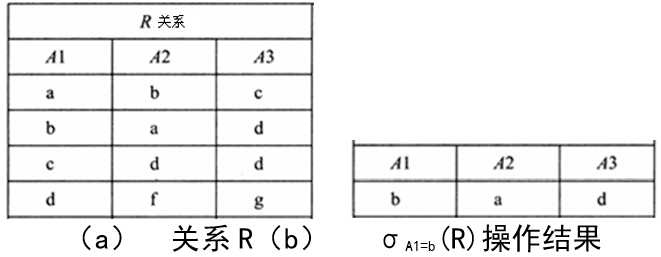


图.9对关系R作选择运算举例

#### 投影(projection)

投影操作从关系R中选择出若干个属性列组成新的关系，该操作对关系进行垂直分割，消去某些列，并重新安排列的顺序，再删去重复元组。记为：

πA(R)≡{t[A]|t∈R} 其中A是R的属性列。例如：对关系R求投影操作：πA1，A2(R)或π1，2(R)

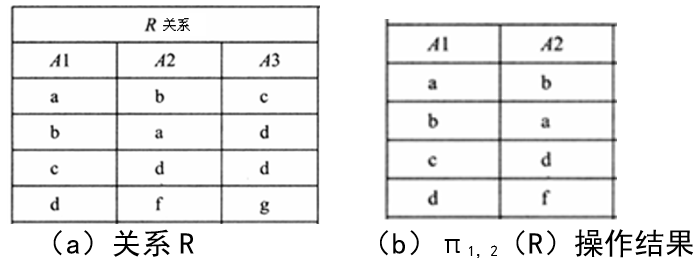
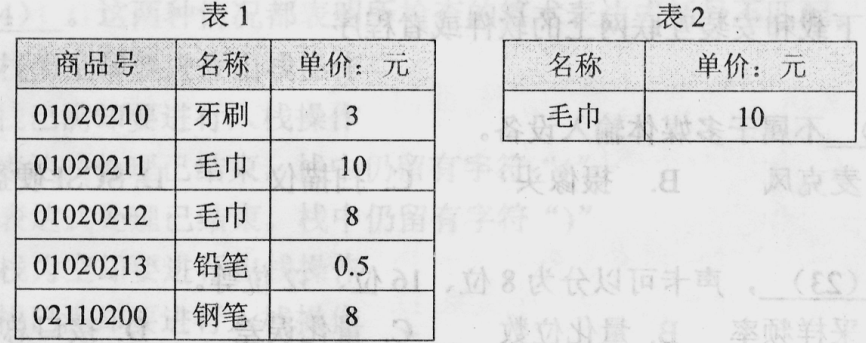


图.10对关系R作投影操作举例

●在关系代数运算中，若对表1按 (15)，可以得到表2。

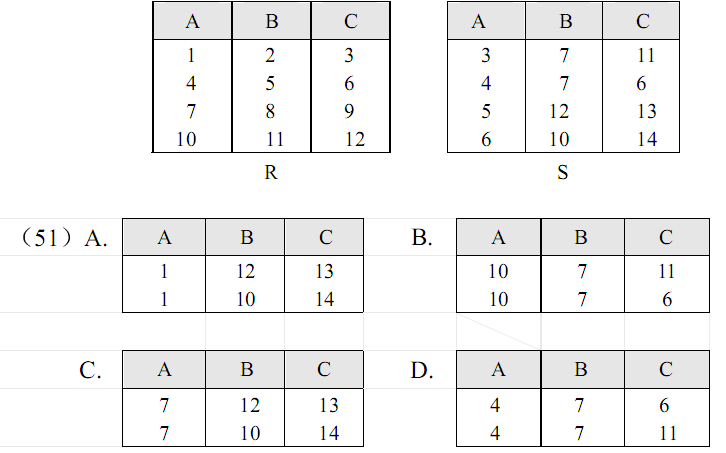


(15)A.商品号都进行选择运算　　 B.名称、单价进行投影运算

　　 C.名称等于“毛巾”进行选择运算，再按名称、单价进行投影运算

　 　D.商品号等于“01020211”进行选择运算，再按名称、单价进行投影运算

●关系R、S如下图所示，关系代数表达式(51)



#### 连接(Join)

也称θ连接，是从两个关系的笛卡儿积中选取属性间满足一定条件的元组，记为；

RS={*t*r*t*s|*t*r∈R∧*t*s∈S∧*t*r[A]θ*t*s[B]}≡σAθB(R×S)

其中AB分别为R和S上度数相等且可比的属性组。θ为比较运算符。(**请牢牢记住 即先求笛卡尔集，然后选择。)**

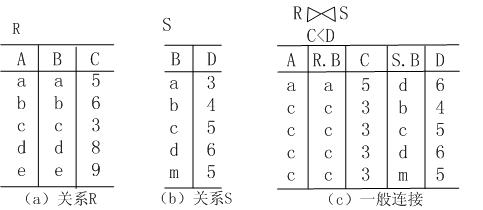
**等值连接**：当θ为“=”的连接，称为等值连接，

记为：RS={*t*r*t*s|*t*r∈R∧*t*s∈S∧*t*r[A]=*t*s[B]}，A=B

**自然连接:** 如果两个关系中进行比较的分量是相同的属性组，并且在结果中把重复的属性列去掉，则称为自然连接，记为：

RS≡{*t*r*t*s|*t*r∈R∧*t*s∈S∧*t*r[A]=*t*s[B]}

例：设图2.11(a)(b)分别为关系R和S。



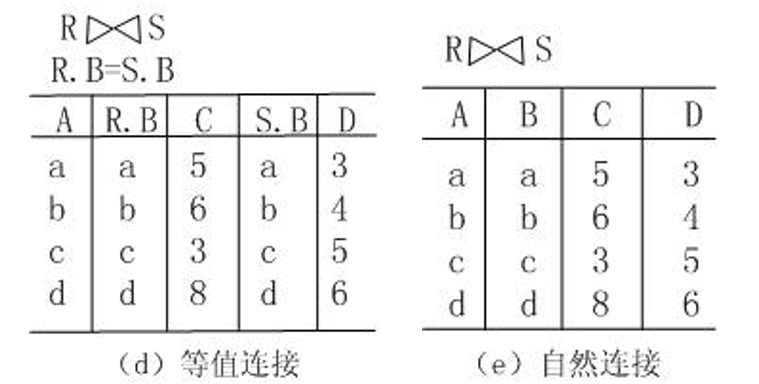


图.11连接运算举例

●注意:θ连接与RS(自然连接)是有区别的。自然连接要把其中一个关系中的相同的属性列去掉，即自然连接得到的关系的属性的个数<r+s，而θ连接得到的关系的属性的个数＝r+s。

如果把舍弃的元组也保存在结果关系中，而在其他属性上填空值(null),那么这种连接就叫做外连接(outer join)。

左外连接：R和S进行自然连接时，以R为主进行操作，不能匹配的字段值记为NULL，如图2.12(a)。

右外连接：R和S进行自然连接时以S为主进行操作，不能匹配的字段值记为NULL，如图2.12(b)

外连接：R与S的左外连接与右外连接的并集，如图2.12(c)。

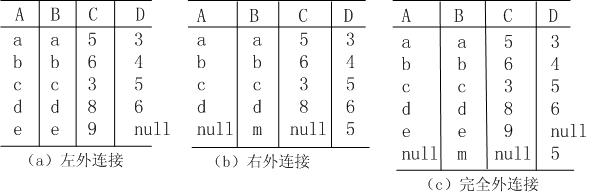


图.12外连接运算举例

●(2019.5)给定关系R(A,B,C,D,E)和关系S(A,C,E,F,G),对其进行自然连接运算RS后其结果集的属性列为 (51) 。

(51)A.R.A,R.C,R.E,S.A,S.C,S.E B.R.A,R.B，R.C,R.D,R.E,S.F，S.G

C.R.A,R.B,R. C，R.D，R.E，S.A，S.C，S.E D.R.A，R.B,R. C，R.D，R.E，S.A，S.C，S.E，S.F，S.G

●(2018.11)给定关系R(A,B,C,D)和S(C,D,E)，若关系R与S进行自然连接运算，则运算后的元组属性列数为(55) ；关系代数表达式π1,4(σ2=5 (R ⋈S))与(56)等价。

(55) A.4 B.5 C.6 D.7

(56) A.πA,D(σC=D(RXS)) B. πR.A,R.D(σR.B=S.C(RXS))

C.πA,R.D(σR.C=S.D(RXS)) D. πR.A,R.D(σR.B=S.E(RXS))

●(2012.11)若对关系R(A，B，C，D)进行∏1,3(R)运算，则该关系运算与 (52) 等价，表示 (53) 。

(52)A．∏A=1,C=3(R) B．∏A=1ΛC=3(R) C．∏A,C(R) D．∏A=1∨ C=3(R)

(53)A．属性A和C的值分别等于1和3的元组为结果集

B．属性A和C的值分别等于l和3的两列为结果集

C．对R关系进行A=l、C=3的投影运算 D．对R关系进行属性A和C的投影运算

●(2016.11)给定关系R(A,B,C,D)和关系S(A,C,E,F)，对其进行自然连接运算RS后的属性列为\_(54)\_个，与(RS)等价的关系代数表达式为\_(55)\_。

(54)A.4 B.5 C.6 D.8

(55)A.(RS) B.((RS)) C.(RS) D.((RS))

●(2016.11)下列查询B=“大数据”且F=“开发平台”，结果集属性列为A、B、C、F的关系代数表达式中，查询效率最高的\_(56)\_。

(56)A.((RS)) B.(((R)S))

C.((R(S)) D.(((R)(S)))

#### 除运算(division)

R与S的除运算得到一个新关系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Y，包含S在Y上投影的集合。

例如：图2.13所示：

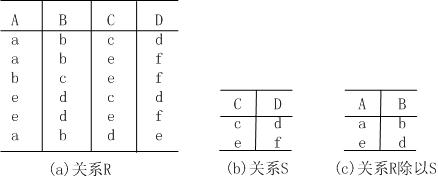


图2.13除运算举例

∵(A,B)两列可以的取值为：{(a,b)、(b,c)、(e,d)}**(原象集合)**

又∵(a,b)的**象集**为：{(c,d) (e,f)(d,e)}=K1

(b,c) 的**象集**为：{(d,f)}=K2

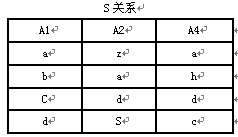
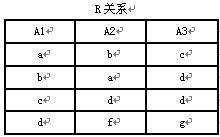
(e,d) 的**象集**为：{(c,d) (e,f)}=K3

同时S关系中(C,D)两列的投影值为：{(c,d) (e,f)}=K

可以看出： 也有 

∴:={K1,K3}的**原象**

●关系R、S如下表所示，R÷(Al,A2(σ1<3(S)))的结果为(40)，左外联接，右外联接和完全外联接的元组个数分别为(41)。



(40)**A.**{d} B. {c，d} C.{c，d，8} D. {(a,b)，(b，a)，(c，d)，(d，f)}

(41) A.2，2 B.2，2，6 **C.**4，4，6 D.4，4，4

●在关系代数运算中，\_(23)\_运算结果的关系模式与原关系模式相同。供选择的答案：

(23)A．并 B．笛卡儿积 C．投影 D．自然连接

# 关系数据库标准语言SQL

## SQL结构化查询语言简介

SQL(structured Qquery Language),即结构化查询语言，是关系数据库的标准语言，它的使用方式有两种，即它可以单独使用，称为交互式语言,也可以用在高级语言编写程序中使用,称为嵌入式语言,这时相应的高级语言称为宿主语言.从SQL数据的体系结构角度来看,用户可以用SQL语言的语句,对基本表和视图进行查询等操作。

SQL的特点主要体现在以下几方面:

##### 集数据定义语言DDL、数据操纵语言DML、数据控制语言DCL的功能于一体，语言风格统一。

##### 存取路径的选择及SQL语句的操作过程由系统自动完成，减轻了用户负担，提高了数据的独立性。因此它是非过程化语言。

##### 采用集合的操作方式。

##### 既是自含式语言(联机交互)，又是嵌入式语言(宿主语言)。

##### 语言简捷，易学易用，只有10个动词(SELECT,CREATE,DROP,ALTER,INSERT,UPDATE,DELETE,GRANT,REVOKE，MODIFY)。

SQL语言的一次查询就是一个表。dBaseII, dBaseIII, ORACLE,INGRES,INFORES,INFROMIX等数据库系统中,除dBaseII外,都提供SQL语言用户界面。

## 数据定义

### 学生-课程数据库



图.1学生-课程完整实体E-R图

经过分析，上图对应关系模型(关系的主码用下划线表示)如下：

* + 学生表：Student(Sno，Sname，Ssex，Sage，Sdept)
  + 课程表：Course(Cno，Cname，Cpno，Ccredit)
  + 学生选课表：SC(Sno，Cno，Grade)

数据结构如表3.1(a)(b)(c)所示。

表3.1学生-课程数据库 (a)学生表Student

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 学号(SNo) | 姓名(SName) | 性别(SSex) | 年龄(SAge) | 所在系(SDept) |
| 200215121 | 李晨 | 男 | 20 | CS |
| 200215122 | 刘晨 | 女 | 19 | CS |
| 200215123 | 王敏 | 女 | 18 | MA |
| 200215124 | 张立 | 男 | 19 | IS |

(b)课程表Course

|  |  |  |  |
| --- | --- | --- | --- |
| 课程号(CNo) | 课程名(CName) | 先行课(CPreNo) | 学分(CCredit) |
| 1 | 数据库 | 5 | 4 |
| 2 | 数学 |  | 2 |
| 3 | 信息系统 | 1 | 4 |
| 4 | 操作系统 | 6 | 3 |
| 5 | 数据结构 | 7 | 4 |
| 6 | 数据处理 |  | 2 |
| 7 | PASCAL语言 | 6 | 4 |

(c)选课表(SC)

|  |  |  |
| --- | --- | --- |
| **学号(SNo)** | **课程号(CNo)** | **成绩(Grade)** |
| 200215121 | 1 | 92 |
| 200215121 | 2 | 85 |
| 200215121 | 3 | 88 |
| 200215122 | 2 | 90 |
| 200215122 | 3 | 80 |

在表3.1的三张表中，Sno是Student表的主键，Cno是Course表的主键，而(Sno，Cno)构成了SC表的主键。三张表之间存在两个参照关系，一个存在于Student和SC表之间，另外一个存在于Course和SC表之间。

### 基本表的定义、删除与修改

#### 定义基本表

SQL语言使用动词CREATE定义基本表，其具体语法格式如下：

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>]

);

上式中除了[……]可有可无外，其它的都是必须要的。

例如，建立，学生表student，它由学号Sno、姓名Sname、性别Ssex、年龄Sage、所在系Sdept 5个属性组成。其中学号不能为空，值是惟一的，并且姓名取值也不能为空。

CREATE TABLE Student

(

Sno CHAR(9) PRIMARY KEY,--在列级上定义主键,也即是定义实体完整性

Sname CHAR(20) NOT NULL, ---->"NOT NULL"为用户自定义完整性

Ssex CHAR(2),

Sage INT，

Sdept CHAR(15),

);

例如，创建课程表Course，如下：

CREATETABLE Course ---->创建课程表Course，主键表

(

Cno INTNOTNULLUNIQUE,

Cname VARCHAR(50)NOTNULL, ---->"NOT NULL"为用户自定义完整性

Cpno INTNOTNULL, ---->数据类型应该和Cno一样,根据表3.1中b表，应该定义为null，但后边有修改

Ccredit SMALLINTNOTNULL,

PRIMARY KEY(Cno) ---->在表级上定义实体完整性,因有此定义，--所以Cno INT NOT NULL UNIQUE中可省略NOT NULL UNIQUE

);

例如，创建课程表选课表，如下：

CREATE TABLESC ---->创建外键表

(

Sno CHAR(9) , ---->外键，要与对应的主键表Student主键Sno同数据类型，名称可不同

Cno INT, ---->外键，要与对应的主键表Course主键Cno同数据类型，名称可不同

Grade SMALLINT,

PRIMARY KEY(Sno,Cno),--字段组合作为主键，只能在表级别上定义

FOREIGN KEY (Sno) REFERENCES Student(Sno),--定义参照完整性

FOREIGN KEY (Cno) REFERENCES Course(Cno)--定义参照完整性

);

#### 数据类型

表3.2SQL Server 2008 R2数据类型

|  |  |  |  |
| --- | --- | --- | --- |
| 数据类型名称 | 类别 | 大小(字节) | 数据的本质 |
| int | integer | 4 | -2 147 483 648~2 147 483 647的全部数值 |
| smallint | integer | 2 | -32 768~32 767的全部数值 |
| datetime | date/time | 8 | 表示日期和时间的数据类型，从1753年1月1日到9999年12月31日，精确度是1/300s |
| char | character | 固定的 | 固定长度字符数据。如果值的长度比设定的长度短，则将添补空格直至达到设定的长度。数据是非Unicode字符数据。可指定的最大长度是8 000字符 |
| varchar | Character | 变化的 | 可变长度的字符数据。不会向值中填入空格。数据是非Unicode字符数据。可指定的最大长度是8 000字符，不过，可以使用max关键字，表明该数据是非常大的字符数据(最大存储大小是231字节) |

#### 修改基本表

格式：--修改表结构

ALTER TABLE <表名>

[ADD<新列名><数据类型> [完整性约束]]

[ADD CONSTRAINT<约束名><完整性约束>]

[DROP<完整性约束>]

[DROP column<列名>]

[ALTER COLUMN<列名><数据类型>];

例如，向Student表增加“入学时间”列，其数据类型为日期型，SQL命令如下：ALTER TABLE Student ADD Scome Date datetime;

#### 删除基本表

格式--删除表

DROP TABLE <表名>>;

注意：基本表一旦删除，表中的数据、表上建立的索引和视图都将自动被删除。

例如，要删除Student表的命令为：

DROP TABLE Student;

## 数据查询

SQL提供了SELECT语句进行数据库的查询，该语句具有灵活的使用方式和丰富的功能。其一般格式为：

--查询一般格式

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> [ HAVING <条件表达式> ]]

[ ORDER BY <列名> [ ASC | DESC ]];

### 单表查询

#### 选择表中的若干列

##### 查询指定列

查询全体学生的学号和姓名，结果如图3.2所示。

SELECT Sno,Sname FROM Student;

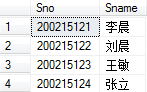


图.2选择Sno、Sname



图.3选择所有列

查询全体学生的详细信息，结果如图3.3所示。

SELECT \* FROM Student;

等价于：

SELECT Sno,Sname,Ssex,Sage,Sdept FROM Student;

##### 查询经过计算的值

查询全体学生的学号及其出生的年份, YEAR(GetDate())求出当前年份，结果如图3.4所示。

SELECT Sno,YEAR(GetDate())-Sage AS BornYear FROM Student;

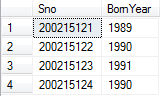


图.4计算的值

#### 查询表中的若干元组

##### 消除取值重复的行

查询选修了课程的学生学号，使用DISTINCT进行筛选，ALL表示全部(ALL可以省略)

SELECT ALL(Sno) FROM SC; ---->结果见图3.5所示

SELECT DISTINCT(Sno)FROM SC; ---->结果见图3.6所示

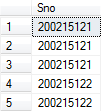


图3.5使用ALL



图.6使用DISTINCT

##### 查询满足条件的元组

查询满足指定条件的元组可以通过WHERE子句实现。WHERE子句常用的查询条件如表3.3所示。

表3.3常用的查询条件

|  |  |  |  |
| --- | --- | --- | --- |
| 查询条件 | 谓词 | 查询条件 | 谓词 |
| 比较 | =,>,<,>=,<=,!=,<>,!>,!< | 字符匹配 | LIKE,NOT LIKE |
| 确定范围 | BETWEEN AND, NOT BETWEEN AND | 空值 | IS NULL,  IS NOT NULL |
| 确定集合 | IN,NOT IN | 多重条件 | AND,OR,NOT |

查询计算机科学系全体学生的名单

SELECT Sname FROM Student WHERE Sdept='CS';

查询考试成绩有不及格的学生的学号

SELECT DISTINCT(Sno) FROM SC WHERE Grade<60;

查询年龄不在20~23岁之间的学生详细情况

SELECT \* FROM Student WHERE Sage NOT BETWEEN 20 AND 23;

查询CS、MA、IS三个系的学生姓名和性别

SELECT Sname,Ssex FROM Student WHERE Sdept IN ('CS','MA','IS');

查询学号为200215121的学生的详细情况

SELECT \* FROM Student WHERE Sno LIKE '200215121';

--与下行等价

SELECT \* FROM Student WHERE Sno = '200215121';

查询所有姓刘的学生的详细信息

SELECT \* FROM Student WHERE Sname LIKE '刘%';

查询姓刘且全名为3个汉字的学生的详细信息

SELECT \* FROM Student WHERE Sname LIKE '刘\_\_';

查询所有不姓刘的学生姓名

SELECT Sname FROM Student WHERE Sname NOT LIKE '刘%';

查询以“DB\_”开头，且倒数第3个字符为i的课程的详细情况

SELECT \* FROM Course WHERE Cname LIKE 'DB\\_%i\_\_' ESCAPE '\';

查询CS系年龄在20岁以下的学生姓名

SELECT Sname FROM Student WHERE Sdept='CS' AND Sage<20;

#### ORDER BY子句

ORDER BY子句对查询结果按照一个或多个属性列的升序(ASC)或降序(DESC)排列，缺省值为升序。

查询选修了3号课程的学生的学号及其成绩，查询结果按分数的降序排序

SELECT Sno,Grade FROM SC WHERE Cno=3 ORDER BY Grade DESC;

查询全体学生情况，查询结果按所在系的系号升序排列，同一系中的学生按年龄降序排列

SELECT \* FROM Student ORDER BY Sdept ASC,Sage DESC;

#### 聚集函数

为了进一步方便用户，增强检索功能，SQL提供了许多聚集函数，主要有：

COUNT( [ DISTINCT | ALL] \* )统计元组个数

COUNT( [ DISTINCT | ALL] <列名> ) 统计一列中值的个数

SUM( [ DISTINCT | ALL] <列名> ) 计算一列中的总和(此列必须是数值型)

AVG( [ DISTINCT | ALL] <列名> ) 计算一列中的平均值(此列必须是数值型)

MAX( [ DISTINCT | ALL] <列名> )求一列中的最大值(此列必须是数值型)

MIN( [ DISTINCT | ALL] <列名> )求一列中的最小值(此列必须是数值型)

查询学生总人数

SELECT COUNT(\*) FROM Student;

查询选修了课程的学生人数

SELECT COUNT(DISTINCT(Sno)) FROM SC;

计算1号课程的平均成绩

SELECT AVG(Grade) FROM SC WHERE Cno='1';--与下句等价

SELECT SUM(Grade)/COUNT(\*) FROM SC WHERE Cno='1';-

查询选修1号课程的学生最高分数

SELECT MAX(Grade) FROM SC WHERE Cno='1';

#### GROUP BY子句

GROUP BY子句将查询按某一列或多列的值分组，值相等的为一组。对查询结果分组的目的是为了细化聚集函数的作用对象。如果为对查询结果分组，聚集函数将作用于整个查询结果。分组后的聚集函数将作用于每一个组，即每一组都有一个函数值。

求各个课程号及相应的选课人数

SELECT Cno,COUNT(Sno) FROM SC GROUP BY Cno;--与下句等价

SELECT Cno,COUNT(\*) FROM SC GROUP BY Cno;

该语句对查询结果按Cno的值分组，所有具有相同Cno值的元组为一组，然后对每一个分组作聚集函数COUNT计算,以求得该组的学生人数。结果如图3.7所示。



图3.7分组查询

查询选修了2门以上课程的学生学号

SELECT Sno FROM SC GROUP BY Sno HAVING COUNT(\*)>2;

WHERE子句和HAVING短语的区别在于作用对象不同。WHERE子句作用于基本表或视图，从中选择满足条件的元组。HAVING短语作用于组，从中选择满足条件的组。

### 多表查询

若一个查询同时涉及到两个以上的表，则称之为多表查询，又叫连接查询，包括等值连接查询、自然连接查询、非等值连接查询、自身连接查询、外连接查询和复合条件查询等。

#### 等值连接

查询每个学生及选修课程的情况，结果如图3.8所示。

SELECT Student.\*,SC.\*

FROM Student,SC

WHEREStudent.Sno=SC.Sno;



图.8学生及选修课情况

#### 自然连接

在等值连接中把目标列中重复的属性列去掉则为自然连接。

查询每个学生及选修课程的情况，结果如图3.9所示。

SELECT Student.\*,SC.Cno,Grade

FROM Student,SC

WHERE Student.Sno=SC.Sno;



图.9自然连接

●(2018.5)给定关系及R(A,B,C,D,E)与S(B,C,F,G)，那么与表达式((R⋈S))等价的SQL语句如下：SELECT (52) FROM R.S WHERE (53) ;

(52)A.R.B,D,F,G B.R.B,E,S.C,F,G C.R.B,R.D,S.C,F D.R.B,R.C,S.C,F

(53)A.R.B = S.B ORR.C = S.C ORR.B < S.G B.R.B = S.B OR R.C = S.C OR R.B <S.C

C.R.B = S.B AND R.C = S.C AND R.B <S.G D.R.B = S.B AND R.C = S.C AND R.B< S.C

#### 自身连接

在同一个表之间进行连接操作，则为自身连接。

查询每一门课的间接先修课(先修课的先修课)

在Course表中，只有每门课的直接先修课信息，而没有先修课的先修课。要得到这个信息，必须先对一门课找到其先修课，再按此先修课的课程号，查找它的先修课程。这就是要将Course表与其自身连接。为此，要为Course表取两个别名，一个是FIRST,另一个是SECOND，见表3.4和

表3.5所示。表3.4FIRST表(Course表)

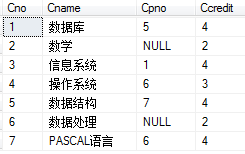
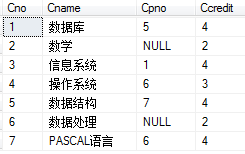


表3.5SECOND表(Course表)



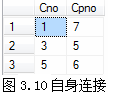


图3.10自身链接

SELECT FIRST.Cno,SECOND.Cpno

FROM CourseASFIRST,Course AS SECOND

WHERE FIRST.Cpno=SECOND.Cno

AND SECOND.Cpno IS NOT NULL; ---->结果如

图3.10所示。

#### 外连接

有时想以Student表为主题列出每个学生的基本情况及选修课情况。若某个学生没有选课，仍把舍弃的Student元组保存在结果关系中，而在SC表的属性上填空NULL，这时就需要使用外连接(OUTER JOIN，其中“OUTER”关键字可省略)。

查询每个学生的基本情况及选修课情况(需要左外连接)

SELECT Student.\*,SC.Cno,Grade

FROM Student

LEFT OUTER JOIN SC

ON Student.Sno=SC.Sno;

查询结果如图3.11所示。左外连接列出左边关系中所有的元组，右外连接列出右边关系中的所有的元组。



图.11左外连接

#### 复合连接查询

当WHERE子句中有多个连接条件时则称为复合条件连接。

查询选修了2号课程且成绩在90分以上的所有学生

SELECT Student.Sno,Sname

FROM Student,SC

WHERE Student.Sno=SC.Sno

AND SC.Cno='2' AND SC.Grade>80; ---->结果如图3.12所示



图.12复合连接

查询每个学生的学号、姓名、选修的课程名及成绩

SELECT Student.Sno,Sname,Cname,Grade

FROM Student,Course,SC

WHERE Student.Sno=SC.Sno AND SC.Cno=Course.Cno;

### 嵌套查询

在SQL语言中，一个SELECT-FROM-WHERE语句称为一个查询块。将一个查询块嵌套在另一个查询块的WHERE子句或HAVING短语的条件中的查询称为嵌套查询。

例如：查询选修了2号课程的学生姓名

SELECT Sname /\*外层查询或父查询\*/

FROM Student

WHERE Sno IN

(SELECT Sno /\*内层查询或子查询\*/

FROM SC

where Cno=’2’);

本例中，下层查询块SELECT Sno FROM SC WHERE Cno=’2’是嵌套在上层查询块SELECT Sname FROM Sno的WHERE条件中的。上层的查询块称为外层查询或父查询，下层查询称为内查询或子查询。

SQL语言多层嵌套查询。即一个子查询中还可以嵌套其他子查询。**需要特别指出的是，子查询的SELECT语句中不能使用ORDER BY子句，ORDER BY子句只能对最终查询结果排序**。

嵌套查询使我们可以用多个简单查询构成复杂的查询，从而增强SQL的查询能力。以层层嵌套的方式构造程序正是SQL中“结构化”的含义所在。

#### 带有IN的子查询

在嵌套查询中，子查询的结果往往是一个集合，所以谓词IN是嵌套查询中最经常使用的谓词。

查询与“刘晨”在同一个系学习的学生(这里假定“刘晨”的情况)

先分步来完成此查询，然后再构造嵌套查询。

第一步：确定“刘晨”所在的系，查询结果为：CS

SELECT Sdept FROM Student WHERE Sname='刘晨';

第二步：查询所有CS系学习的学生。

SELECT Sno,Sname,Sdept FROM Student WHERE Sdept='CS';

第三步：将第一步查询嵌入第二步查询的条件中，构造嵌套查询如下：

SELECT Sno,Sname,Sdept FROM Student

WHERE Sdept

IN

(SELECT Sdept FROM Student WHERE Sname='刘晨');

本例子中，子查询的查询条件不依赖于父查询，称为不相关子查询。一种求解方法是由里向外处理，即先执行子查询，子查询的结果用于建立其父查询的查找条件。得到如下的语句：

SELECT Sno,Sname,Sdept FROM Student WHERE Sdept IN ('CS');

查询选修了课程名为“信息系统”的学生学号和姓名

SELECT Sno,Sname

FROM Student

WHERE Sno IN

(

SELECT Sno

FROM SC

WHERE Cno IN

(

SELECT Cno

FROM Course

WHERE Cname='信息系统'

)

);

本查询同样可以用连接查询实现：

SELECT Student.Sno,Sname

FROM Student,SC,Course

WHERE Student.Sno=SC.Sno AND

SC.Cno=Course.Cno AND Course.Cname='信息系统';

#### 带有比较运算符的子查询

带有比较运算符的子查询是指父查询与子查询之间用比较运算符进行连接。当用户能确切知道内层查询返回的是单值时，可以用>,<,=,>=,<=,!=或<>等比较运算符。

查询与“刘晨”在同一个系学习的学生

由于一个学生只可能在一个系学习，也就是说内查询的结果是一个值，因此可以用=代替IN。

SELECT Sno,Sname,Sdept

FROM Student

WHERE Sdept =

(SELECT Sdept FROM Student WHERE Sname='刘晨');

找出每个学生超过他选修课程平均成绩的课程号。注意：如200215121学生选修了1、2、3号课程，则平均成绩为此3个课程的平均成绩。

SELECT Sno,Cno

FROM SC x

WHERE Grade >=(SELECT AVG(Grade)

FROM SC y

WHERE y.Sno=x.Sno);

x是表SC的别名，又称为元组变量，可以用来表示SC的一个元组。内层查询是求一个学生所有选修课程的平均成绩，至于是哪个学生的平均成绩要看参数x.Sno的值，而该值与父查询相关的，因此这类查询称为相关子查询。

这个语句的一种可能的执行过程是：

第一步：从外层查询中取出SC的一个元组x，将元组x的Sno值(200215121)传送给内存查询。

SELECT AVG(Grade)

FROM SC y

WHERE y.Sno='200215121';

第二步：把第一步执行的结果88(近似值)代替内层查询，得到外层查询：

SELECT Sno,Cno

FROM SC x

WHERE Grade>=88 AND Sno='200215121';

得到结果如图3.13所示。



图3.13学号200215121对应查询结果

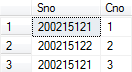


图.14最终查询结果

然后外层查询取出下一个元组重复上述第一步、第二步的处理过程，直到外层的SC元组全部处理完毕。结果如图3.14所示。

求解相关子查询不能像求解不相关子查询那样，一次将子查询求解出来，然后求解父查询。内层查询由于与外层查询有关，因此必须反复求值。

#### 带有ANY或ALL谓词的子查询

子查询返回单值时可以用比较运算符，但返回多值时要用ANY(有的系统用SOME)或ALL谓词修饰符。而使用ANY或ALL谓词时则必须同时使用比较运算符。其语义为：

* + > ANY 大于子查询结果中的某个值
  + > ALL 大于子查询结果中的所有值
  + < ANY 小于子查询结果中的某个值
  + < ALL 小于子查询结果中的所有值
  + >= ANY 大于等于子查询结果中的某个值
  + >= ALL 大于等于子查询结果中的所有值
  + <= ANY 小于等于子查询结果中的某个值
  + <= ALL 小于等于子查询结果中的所有值
  + = ANY 等于子查询结果中的某个值
  + =ALL 等于子查询结果中的所有值(通常没有实际意义)
  + !=(或<>)ANY 不等于子查询结果中的某个值
  + !=(或<>)ALL 不等于子查询结果中的任何一个值

查询其他系中比计算机科学系所有学生年龄都小的学生姓名及年龄

SELECT Sname,Sage

FROM Student

WHERE Sage<ALL

( SELECT Sage

FROM Student

WHERE Sdept = 'CS')

AND Sdept<>'CS';

本查询也可以用聚集函数来实现，如下：

SELECT Sname,Sage

FROM Student

WHERE Sage<

( SELECT MIN(Sage)

FROM Student

WHERE Sdept = 'CS')

AND Sdept<>'CS';

#### 带有EXISTS谓词的子查询

EXISTS谓词--存在量词∃，带有EXISTS谓词的子查询不返回任何数据，只产生逻辑真值“true”或逻辑假值“false”。若内层查询结果非空，则返回真值。若内层查询结果为空，则返回假值。

由EXISTS引出的子查询，其目标列表达式通常都用\* ，因为带EXISTS的子查询只返回真值或假值，给出列名无实际意义

查询所有选修了1号课程的学生姓名。

SELECT Sname

FROM Student

WHERE EXISTS

(SELECT \*

FROM SC

WHERE Sno=Student.Sno AND Cno= '1');

求解过程的思路分析：

本查询涉及Student和SC关系。在Student中依次取每个元组的Sno值，用此值去检查SC关系。 若SC中存在这样的元组，其Sno值等于Student.Sno值，并且其Cno= '1'，则取此Student.Sname送入结果关系。

本例也可以使用连接查询如下：

SELECT Sname

FROM Student, SC

WHERE Student.Sno=SC.Sno AND SC.Cno= '1';

查询没有选修1号课程的学生姓名。

SELECT Sname

FROM Student

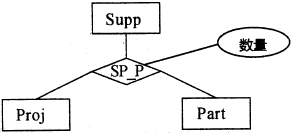
WHERE NOT EXISTS

(SELECT \*

FROM SC

WHERE Sno = Student.Sno AND Cno='1');

●(2017.5)在某企业的工程项目管理系统的数据库中供应商关系Supp、项目关系Proj和零件关系Part的E-R模型和关系模式如下:



Supp(供应商号,供应商名,地址,电话)，Proj(项目号,项目名,负责人,电话)，Part(零件号,零件名)，其中，每个供应商可以为多个项目供应多种零件，每个项目可由多个供应商供应多种零件。SPP需要生成一个独立的关系模式，其联系类型为(54)。给定关系模式SPP(供应商号,项目号,零件号,数量)查询至少供应了3个项目(包含3项)的供应商，输出其供应商号和供应零件数量的总和，并按供应商号降序排列。

SELECT供应商号，SUM(数量)FROM(55)

GROUPBY供应商号

(56)

ORDERBY供应商号DESC;

(54)A.\*:\*:\* B.1:\*:\* C.1:1:\* D.1:1:1

(55)A.Supp B.Proj C.Part D.SPP

(56)A.HAVING COUNT(项目号)>2 B.WHERE COUNT(项目号)>2

C.HAVING COUNT(DISTINCT(项目号))>2 D.WHERE COUNT(DISTINCT(项目号))>3

## 数据更新

### 插入数据

SQL的数据插入语句INSERT通常有两种形式。一种是插入一个元组，另一种是插入子查询结果。后者可以一次插入多个元组。

DBMS在执行修改语句、插入语句时会检查修改操作是否破坏表上已定义的完整性规则：

* + 实体完整性
  + 参照完整性
  + 用户定义的完整性
    - NOT NULL约束
    - UNIQUE约束
    - 值域约束

#### 插入元组

* 语句格式

INSERT INTO <表名> [(<属性列1>[，<属性列2 >…)]

VALUES (<常量1> [，<常量2>] … );

* 功能: 将新元组插入指定表中。
* INTO子句
  + 指定要插入数据的表名
  + 没有指定属性列：表示要插入的是一条完整的元组，且属性列属性与表定义中的顺序一致
  + 若指定属性列，属性列的顺序可与表定义中的顺序不一致
  + 若指定部分属性列：插入的元组在其余属性列上取空值
* VALUES子句
  + 提供的值必须与INTO子句匹配
    - 值的个数
    - 值的类型

将一个新学生记录:(学号：200215125；姓名：陈冬；性别：男；所在系：IS；年龄：18岁)插入到Student表中。

INSERT INTO Student VALUES ('200215125','陈冬','男',18,'IS');

插入一条选课记录( ‘200215125’，'1 ')，新插入的记录在Grade列上取空值

INSERT INTO SC(Sno,Cno) VALUES ('200215125','1');--与下行等价

INSERT INTO SC VALUES ('200215125','1',NULL);

### 修改数据

* 语句格式

UPDATE <表名>

SET <列名>=<表达式>[，<列名>=<表达式>]…

[WHERE <条件>]；

* 功能：修改指定表中满足WHERE子句条件的元组
* SET子句：指定修改方式、要修改的列、修改后取值。
* WHERE子句：指定要修改的元组、缺省表示要修改表中的所有元组
* 三种修改方式
  + 修改某一个元组的值
  + 修改多个元组的值
  + 带子查询的修改语句

将学生200215121的年龄改为22岁

UPDATE Student SET Sage=22 WHERE Sno='200215121';

将所有学生的年龄增加1岁

UPDATE Student SET Sage=Sage+1;

将计算机科学系全体学生的成绩置零

UPDATE SC

SET Grade=0

WHERE 'CS'=

(SELECT Sdept

FROM Student

WHERE Student.Sno=SC.Sno);等价于下行：

UPDATE SC

SET Grade=0

WHERE sno in

(SELECT sno

FROM Student

WHERE sdept=’cs’);

### 删除数据

* 语句格式

DELETE FROM <表名> [WHERE <条件>]；

功能：删除指定表中满足WHERE子句条件的元组。

* WHERE子句
  + 指定要删除的元组
  + 缺省表示要修改表中的所有元组
* 三种删除方式
  + 删除某一个元组的值
  + 删除多个元组的值
  + 带子查询的删除语句

删除学号为20021525的学生记录

DELETE FROM Student WHERE Sno='200215128';

删除所有的学生选课记录

DELETE FROM SC;

删除计算机科学系所有学生的选课记录

DELETE FROM SC

WHERE 'CS'=

(SELECT Sdept

FROM Student

WHERE Student.Sno=SC.Sno);等同于下行：

DELETE FROM SC

WHERE sno in

(SELECT sno

FROM Student

WHERE sdept='cs');

DBMS在执行插入、删除、更新语句时必须保证数据库一致性，必须遵循实体完整性、参照完整性、用户自定义完整性，必须有事务的概念和原子性、检查和保证完整性。

## 视图

基本表是实际存在的表，它是实际存储数据的逻辑表示。视图是从一个或几个基本表(或视图)导出的表。它与基本表图不同，是一个虚表。数据库中只存放视图的定义，而不存放视图对应的数据，这些数据仍然存放在原来的基本表中。所以基本表中的数据发生变化，从视图中查询出来的数据也就随之发生变化。所以，视图就像一个窗口，透过它可以看到数据中自己感兴趣的数据及其变化。

视图一经定义，就可以和基本表一样被查询、被删除。也可以在一个视图上再定义新的视图，但对实体的更新(增、删、改)操作则有一定的限制。

在DBMS中，视图的作用如下：

##### 简化用户的操作；

##### 使用户能从多种角度看待同一个数据；

##### 对重构数据库提供了一定程度的逻辑独立性；

##### 能够对机密数据提供安全保护。

### 定义视图

#### 建立视图

* 语句格式

CREATE VIEW <视图名> [(<列名> [，<列名>]…)]

AS <子查询>

[WITH CHECK OPTION]；

* 其中:
* (<列名> [，<列名>]…)为可选项，省略时，视图的列名由子查询的结果决定。
* 在子查询中不许使用ORDER BY 子句和DISTINCT短语
* WITH CHECK OPTION 通过视图进行增删改操作时，不得破坏视图定义中的谓词条件(即子查询中的条件表达式)

建立信息系学生的视图。

CREATE VIEW View\_IS\_Student

AS

SELECT Sno,Sname,Ssex,Sage,Sdept

FROM Student

WHERE Sdept= 'IS';

建立信息系学生的视图，并要求透过该视图进行的更新操作只涉及信息系学生。

CREATE VIEW View\_IS\_Student

AS

SELECT Sno,Sname,Ssex,Sage,Sdept

FROM Student

WHERE Sdept= 'IS'

WITH CHECK OPTION;

由于在定义View\_IS\_Student视图时加上了子句，以后对该视图进行插入、修改和删除操作时，DBMS会自动加上Sdept= 'IS'这个条件。

建立信息系选修了1号课程的学生视图。

CREATE VIEW VIEW\_IS\_S1(Sno,Sname,Grade)

AS

SELECT Student.Sno,Sname,Grade

FROM Student,SC

WHERE Sdept= 'IS' AND

Student.Sno=SC.Sno AND

SC.Cno= '1';

建立信息系选修了1号课程且成绩在90分以上的学生的视图。

CREATE VIEW VIEW\_IS\_S2

AS

SELECT Sno,Sname,Grade

FROM VIEW\_IS\_S1

WHERE Grade>=90;

可以看到，视图VIEW\_IS\_S2是建立在VIEW\_IS\_S1视图上的。

#### 删除视图

* 语句格式：

DROP VIEW <视图名>；

* 说明：
* 该语句从数据字典中删除指定的视图定义；
* 由该视图导出的其他视图定义仍在数据字典中，但已不能使用，必须显式删除；
* 删除基表时，由该基表导出的所有视图定义都必须显式删除。

删除视图VIEW\_IS\_S1

DROP VIEW VIEW\_IS\_S1;

### 查询视图

视图定义以后，用户就可以像对基本表一样对视图进行查询了。

在信息系学生的视图中找出年龄小于岁的学生。

SELECT Sno,Sage FROM View\_IS\_Student WHERE Sage<20;

View\_IS\_Student视图的定义(视图定义例)：

CREATE VIEW View\_IS\_Student

AS

SELECT Sno,Sname,Ssex,Sage,Sdept

FROM Student

WHERE Sdept= 'IS';

用视图消解法，转换后的查询语句为：

SELECT Sno,Sage

FROM Student

WHERE Sdept='IS' AND Sage<20;

查询信息系选修了1号课程的学生

SELECT a.Sno,Sname

FROM View\_IS\_Student as a,SC

WHERE a.Sno =SC.Sno AND SC.Cno= '1';

本查询涉及视图View\_IS\_Student(虚表)和基本表SC，通过这两个表的连接来完成请求。

### 更新视图

更新视图是指通过视图来插入(INSERT)、删除(DELETE)修改(UPDATE)数据。

由于视图是不实际存储数据的虚表，因此对视图的更新，最终要转换为对基本表的更新。像查询视图那样，对视图的更新操作也是通过视图消解，转换为对基本表的更新操作。

为防止用户通过视图对数据进行增加、删除、修改时，有意无意地对不属于视图范围内的基本表数据进行操作，可在定义视图时加上WITH CHECK OPTION子句。这样在视图上增删改数据时，DBMS会检查视图定义中的条件，若不满足条件，则拒绝执行该操作。

将信息系学生视图View\_IS\_Student中学号200215124的学生姓名改为“刘辰”。

UPDATE View\_IS\_Student

SET Sname= '刘辰'

WHERE Sno= '200215124';

用视图消解法，转换后的更新语句为：

UPDATE Student

SET Sname= '刘辰'

WHERE Sno= '200215124' AND Sdept= 'IS';

向信息系学生视图View\_IS\_Student中插入一个新的学生记录，学号为200215136，姓名为赵新，性别为男，年龄为20岁。

INSERT INTO View\_IS\_Student(Sno,Sname,Ssex,Sage,Sdept)VALUES('200215136','赵新','男',20,'is');

用视图消解法，转换后的插入语句为：

INSERT INTO Student(Sno,Sname,Ssex,Sage,Sdept)VALUES('200215129','赵新','男',20,'IS' );

删除视图View\_IS\_Student中学号为的记录。

DELETE FROM View\_IS\_StudentWHERE Sno= '200215129';

用视图消解法，转换为对基本表的删除：

DELETEFROM StudentWHERE Sno= '200215129' AND Sdept='is';

从上面的视图操作中，可以看出视图能简化用户的操作，使用户能以多种角度看待同一数据等。

●(2013.11)在数据库系统中，视图是一个(54)。

(54) A.真实存在的表，并保存了待查询的数据 B.真实存在的表，只有部分数据来源于基本表

C.虚拟表，查询时只能从一个基本表中导出

D.虚拟表，查询时可以从一个或者多个基本表或视图中导出

●(2015.11)数据库系统通常采用三级模式结构：外模式、模式和内模式。这三级模式分别对应数据库的(51)。

(51)A.基本表、存储文件和视图 B.视图、基本表和存储文件

C.基本表、视图和存储文件 D.视图、存储文件和基本表

## SQL自定义函数

SQL中自定义函数总共有三种, 根据返回值的类型可以分为标量函数(返回单个值)和表值函数(返回一个表，表值函数又有二种，内联表格值函数和多句表格值函数,不同数据库管理系统中函数的定义语法有一些细微差别.

1. 标量值函数。

语法如下：

CREATE FUNCTION 函数名( @参数名称 参数数据类型)

RETURNS 返回值类型

[WITH ENCRYPTION] ---->如果指定了encryption则函数被加密

[AS] ----> []选项为可选，在实现时可有可无

[BEGIN]

function\_body ---->函数体

RETURN 表达式; ---->返回

[END]

不论是何种数据库管理系统，以上定义函数的格式基本上是一致的，下例中分别使用SqlServer数据库和MySql数据库中语法进行演示。

**例题：给定教师关系Teacher(T\_no,T\_name,Dept\_name,Tel),其中属性T\_no、T\_\_name、Dept\_name和Tel的含义分别为教师号、教师姓名、学院名和电话号码。用SQL创建一个“给定学院名求该学院的教师数”的函数如下：**

在SqlServer数据库中，实现题目要求的函数定义如下：

Create function Dept\_count ( **@**Dept\_name varchar(20)) ---->创建名为'Dept\_count'的标量函数

returns integer ---->(54) 返回值类型为整型integer

begin ---->开始

declare @d\_count integer; ---->(55) 使用declare关键字定义整型变量@d\_count

select @d\_count=count(\*) ---->将查询出来的统计值保存到变量

from Teacher ---->查询Teacher表

where Teacher.Dept\_name=@Dept\_name; ---->根据部门名称进行查询

return @d\_count ---->返回统计值，即教师个数

end ---->结束

若在MySql数据库中，则可定义如下：

Create function Dept\_count (Dept\_name varchar(20)) ---->创建名为'Dept\_count'的标量函数

returns integer ---->(54) 返回值类型为整型integer

begin ---->--开始

declare d\_count integer; ---->(55) 使用declare关键字定义整型变量d\_count

select count(\*) into d\_count ----->将查询出来的统计值保存到变量

from Teacher ----->查询Teacher表

where Teacher.Dept\_name=Dept\_name; ---->根据部门名称进行查询

return d\_count; ---->返回统计值，即教师个数

end ---->结束

可见，在SqlServer中，变量名称前必须有符号“@”，而MySql中则不需要。

定义的标量函数可采取如下的方式调用：

SELECT Dept\_count('计算机学院') as result; ---->查询计算机学院的教师人数

●(2018.5)给定教师关系Teacher(T\_no,T\_name,Dept\_name,Tel),其中属性T\_no、T\_\_name、Dept\_name和Tel的含义分别为教师号、教师姓名、学院名和电话号码。用SQL创建一个“给定学院名求该学院的教师数”的函数如下：

Create function Dept\_count(Dept\_name varchar(20))

(54)

begin

(55)

Select count(\*) into d\_count

From Teacher

whereTeacher.Dept\_name=Dept\_name

returnd\_count

end

(54)A.returns integer B.returns d\_count integer C.declare integer D.declare d\_count integer

(55)A.returns integer B.returns d\_count integer C.declare integer D.declare d\_count integer

1. 表格值函数。函数返回的不是一个独立的标量值，而是一个表格。

* 内联表格值函数。语法如下：

create function 函数名(参数) ---->定义函数

returns table ---->返回类型为表格

[with Encryption] ---->如果指定了encryption则函数被加密

[AS] ----> []选项为可选，在实现时可有可无

[BEGIN]

function\_body ---->函数体

RETURN 表达式; ---->返回

[END]

如需要根据部门名称查询教师清单，在Sql Server中创建函数如下：

create function GetTeacherDetailsByTypeValue(@TypeValue varchar(50))

returns table ---->返回的表格列由查询的结果构成

as

begin

return (select t.T\_no,t.T\_name,t.Dept\_name,t.Tel ---->返回的表格列由T\_no,t.T\_name,t.Dept\_name,t.Tel构成

from Teacher t

where [t.Type=@TypeValue](mailto:t.Type=@TypeValue));

end

可通过如下语句调用函数：

select \* from dbo.GetTeacherDetailsByTypeValue('计算机学院')

* 多句表格值函数

create function 函数名(参数)--创建函数

return @dt table(列的定义) ----> dt为执行后返回的表格名称，即@dt为表格变量，(列的定义)为表格中列的属性、名称等定义

[with Encryption] ---->如果指定了encryption则函数被加密

[as]

[begin]

SQL语句

[end]

如需要根据部门名称查询教师清单，在Sql Server中创建函数如下：

create function GetTeacherInfoByDptName(@Dept\_name varchar(50))

returns @dt table(T\_no varchar(20),T\_name varchar(50))---->返回的表格dt包含T\_no和 T\_name两列，

[as]

[begin]

insert into @dt select T\_no,T\_name from Teacher where Dept\_name=@Dept\_name;--将查询结果插入@dt中

return

[end]

使用如下语句调用：

select \* from GetTeacherInfoByDptName('计算机学院') ---->查询计算机学院的教室编号和姓名

## 关系演算

关系演算(把数理逻辑的谓词演算引入到关系运算中)，该理论也应掌握。掌握的深度要求比关系代数略低一些。

#### 元组演算表达式的一般形式为：{t|P(t)}

比如，设有三个关系：

* + S(S#,SNAME,AGE,SEX)
  + SC(S#,C#,GRADE)
  + C(C#,CNAME,TEACHER)

则{t|(u)(C(u)∧C[3]='LIU'∧t[1]=u[1]∧t[2]=u[2])}可以这样理解：t是所求元组，对于t，存在元组U,U是关系C的一个元组并且C的第三个分量是“LIU"，t的第一个分量等于U的第一个分量,t的第二个分量等于U的第二个分量。

综上，题意：检索LIU老师所授课程的课程号(第一个分量)、课程名(第二个分量)。

#### 域关系演算

域演算表达式的一般形式为：{t1...tk|P(t1...,tk)}。含义为：这是一个域集合，其中每个域变量的取值关系满足公式P所规定的条件。设有三个关系：

* + S(S#,SNAME,AGE,SEX)
  + SC(S#,C#,GRADE)
  + C(C#,CNAME,TEACHER)

比如，{t1t2|(u1)(u2)(C(u1u2u3)∧u3='LIU'∧t1=u1∧t2=u2)}可以这样理解：所求结果有t1、t2两个域，存在u1,u2与它们分别相等(u1,u2为关系C中两个域，对应的U3为'LIU')。

综上，题意：检索LIU老师所授课程的课程号(第一个域)、课程名(第二个域)。

##### 检索LIU老师所授课程的课程号、课程名。

πC#,CNAME(σTEACHER='LIU'(C))

{t|(existu)(C(u)∧C[3]='LIU'∧t[1]=u[1]∧t[2]=u[2])}

##### 检索年龄大于23岁的男学生的学号与姓名。

πS#,SNAME(σAGE＞'23'∧SEX='男'(S))

{t|(existu)(S(u)∧u[3]＞'23'∧u[4]='男'∧t[1]=u[1]∧t[2]=u[2])}

##### 检索学号为S3学生所学课程的课程名与任课教师名。

πCNAME,TEACHER(σS#='S3'(SCjoinC))

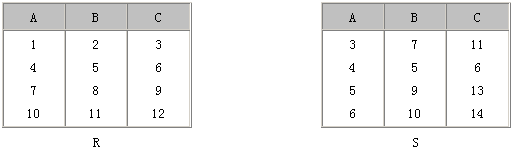
{t|(existu)(existv)(SC(u)∧C(v)∧u[1]='S3'∧v[1]=u[2]∧t[1]=v[2]∧t[2]=v[3])}

##### 检索至少选修LIU老师所授课程中一门课程的女学生的姓名。

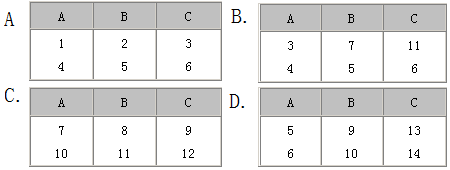
πSNAME(σSEX='女'∧TEACHER='LIU'(SjoinSCjoinC))

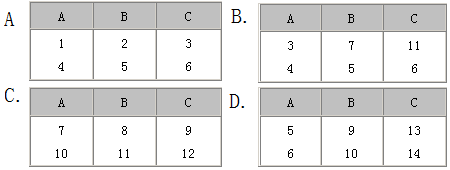
{t|(existu)(existv)(existw)(S(u)∧SC(v)∧C(w)∧u[4]='女'∧v[1]=u[1]∧v[2]=w[1]∧w[3]='LIU'∧t[1]=u[2])}

●关系 R、S 如下图所示，元组演算表达式说明: gc06_2s06的结果为(47)。

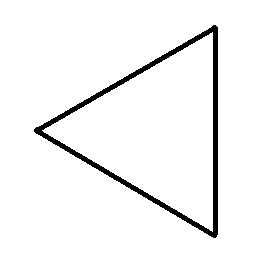
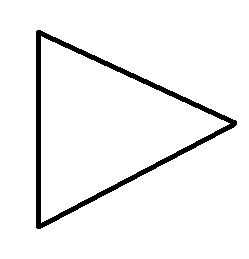
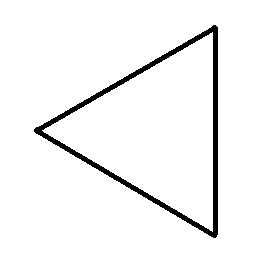
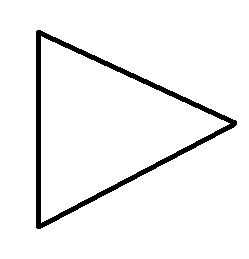


(47)





●设学生 S、课程 C、学生选课 SC 的关系模式分别为：

S(Sno，Sname，Sage，Saddr)、C(Cno，Cname，Pcno)以及 SC(Sno，Cno，Grade)与关系代数表达式πSno，Sname，Grade(σcname='数据库'(SSCC))等价的元组演算表达式为：

{\_\_(35)\_\_S(u)∧SC(v)∧C(w)∧\_\_(36)\_\_∧\_\_(37)\_\_}

(35)A.(存在u)( existv)( existw) B.( existu)( 存在v)( existw) C.(existu)( existv)( existw) D.( existu)( existv)( 存在w)

(36)A.u[1]=v[1]∧v[1]=w[l]∧w[1]='数据库' B.u[1]=v[2]∧v[2]=w[l]∧w[3]='数据库'

C.u[1]=v[1]∧v[2]=w[l]∧w[2]='数据库' D.u[2]=v[2]∧v[1]=w[2]∧w[2]='数据库'

(37)A.t[1]=u[1]∧t[2]=u[2]∧t[3]=v[3] B.t[1]=u[1]∧t[2]=u[2]∧t[3]=v[2]

C.t[1]=u[1]∧t[2]=w[1]∧t[3]=v[2] D.t[1]=u[1]∧t[2]=w[2]∧t[3]=v[3]

●某数据库中有供应商关系S和零件关系P，其中：供应商关系模式S(Sno,Sname,Szip,City)中的属性分别表示：供应商代码、供应商名、邮编、供应商所在城市；零件关系模式P(Pno，Pname，Color，Weight，City)中的属性分别表示：零件号、零件名、颜色、重量、产地。要求一个供应商可以供应多种零件，而一种零件可以由多个供应商供应。请将下面的SQL语句空缺部分补充完整。

CREATE TABLE SP(Sno CHAR(5),

Pno CHAR(6),

Status CHAR(8),

Qty NUMERIC(9),

(33)(Sno,Pno),

(34)(Sno),

(35)(Pno);

查询供应了“红”色零件的供应商号、零件号和数量(QTY)的元组演算表达式为：

{t|u(v)(w)(36u[1]=v[1] v[2] =w[1] w[3]=‘红'37)}

(33)A.FOREIGN KEY B.PRIMARY KEY C.FOREIGN KEY(Sno)REFERENCES S

D.FOREIGN KEY(Pno) REFERENCES P

(34)A.FOREIGN KEY B. PRIMARY KEY C.FOREIGN KEY(Sno) REFERENCES S

D.FOREIGN KEY(Pno) REFERENCES P

(35)A.FOREIGN KEY B. PRIMARY KEY C.FOREIGN KEY(Sno)REFERENCES S

D.FOREIGN KEY(Pno) REFERENCES P

(36)A.S(U)SP(V)P(W) B. SP(U)S(V)P(W)

C. P(U)S(P)S(W) D. S(U)(V)SP(W)

(37)A.t[1]=U[1]T[2]=W[2]T[3]=V[4] B. t[1]=V[1]T[2]=U[2]T[3]=U[4]

C. t[1]=W[1]T[2]=U[2]T[3]=V[4] D. t[1]=U[1]T[2]=V[2]T[3]=V[4]

# 模式分解、函数依赖与范式

## 函数依赖的引入

对于给定学生关系S(Sno,Sname,Sdept)，如果有：Sno->Sname ,Sno->Sdept，则称Sno函数决定Sname；Sno函数决定Sdept，或称Sname，Sdept函数依赖于Sno。

例如，对于给定学生关系S(Sno，Sdept，Mname，Cno，Grade)，Sno为学号，Sdept所在系，Mname为系主任姓名，Cno为课程号，Grade为成绩，则有如下函数依赖：Sno->Sdept；Sdept->Mname；(Sno，Cno)->Grade；如表4.1所示。

表4.1学生关系S

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sno | Sdept | Mname | Cno | Grade |
| S1 | 计算机系 | 李明 | C1 | 68 |
| S2 | 计算机系 | 李明 | C1 | 90 |
| S3 | 计算机系 | 李明 | C1 | 70 |
| S4 | 计算机系 | 李明 | C1 | 88 |

可以看出，给定的这一关系模式存在如下几个问题：

#### 数据余度太大

比如每一个系的系主任姓名重复出现，重复出现次数与该系所有学生的所有课程成绩出现次数相同，如上表，浪费大量的存储空间。

#### 更新异常

当更新数据库中的数据时，系统要付出很大的代价来维护数据库的完整性，否则会面临数据不一致的危险。比如，某系更换系主任后，必须修改与该系学生有关的一个元组。

#### 插入异常

如果一个系刚成立，尚无学生，就无法把这个系及其系主任的信息存入数据库。

#### 删除异常

如果一个系的学生全部毕业了，在删除该系学生信息的同时，把这个系及系主任的信息也丢掉了。

结论：S关系模式不是一个好模式。一个“好”的模式应当不会发生插入，删除，更新异常，故数据冗余应尽可能少。

## 规范化

### 函数依赖

#### 函数依赖

设R(U)是属性U上的一个关系模式，X和Y是U的子集，r是R的任一关系，如果对于r中的任意两个元组u,v,只要有u[X]=v[X],就有u[Y]=v[Y],则称X函数决定Y，或称Y函数依赖于X，记为：X→Y。

函数依赖不是指关系模式R的某个或某些关系满足的约束条件，而是指R的一切关系都要满足的约束条件。

若X→Y，Y→X，则记作X←→Y；

若Y函数不依赖于X,则记作XY。

#### 非平凡的函数依赖

如果X→Y，但Y不是X的真子集，则称X→Y是非平凡函数依赖，一般情况下总是讨论非平凡的函数依赖。

#### 平凡的函数依赖

如果X→Y，但Y是X的真子集，则称X→Y是平凡函数依赖。

#### 完全函数依赖

在R(U)中，如果X→Y成立，并且对于X的任何一个真子集X，都有X→Y不成立，则称Y完全函数依赖于X，记作XY。

#### 部分函数依赖

如果X→Y，但Y不完全函数依赖于X，则称Y对X部分函数依赖,记作XY。

#### 传递函数依赖

在R(U)中，如果X→Y，Y不是X的真子集，且YX，Y→Z，则称Z对X传递函数依赖。

例如：在关系Student(Sno，Sname，Ssex，Sage，Sdept)中，存在有：SnoSname；SnoSsex；SnoSage；SnoSdept；但SnameSage；SageSsex等；在关系SC(Sno，Cno，Grade)中，存在有：(Sno，Cno)Grade，但SnoGrade；SnoCno；CnoGrade等。

#### 码

设X为关系模式R<U,F>中的属性或属性组合，如果有X→U在关系模式R(U)上成立，并且不存在X的任意真子集X使X→U成立，那么称X是R的候选码。若候选码多于一个，则选定其中一个为主码(Primary key)。

#### 主属性和非主属性

包含在任何一个候选码中的属性为主属性，不包含在任何码中的属性称为非主属性或非码属性，最极端的情况是整个属性值是码，成为全码。

#### 外码

关系模式R中属性或属性组X并非R的码，但是是另一个关系模式的码，则称X是R的外部码(Foreign key)，也称外码。

### 推理规则

定义：对于满足一组函数依赖F的关系模式R<U,F>,其任何一个关系r，若函数依赖X→Y都成立(即r中的任意两元组t，s，若t[X]=s[X]，则t[Y]=s[Y]),则称F逻辑蕴含X→Y。

设U是关系模式R的属性集，F是R上成立的只涉及到U中属性的函数依赖集合，则有以下三条推理规则：

自反律：若YXU,则X→Y为F所蕴含。

注意：由此规则可推出任何集合都函数依赖于其自身，即X→X，Y→Y，AB→AB。

增广律：若X→Y在R上成立，且ZU，则XZ→YZ在R上成立。

传递律：若X→Y和Y→Z在R上成立，则X→Z在R上也成立。

注意：XZ，YZ等写法表示X∪Z，Y∪Z。

根据上述三条规则还可以推出其他三条常用的推理规则：

* + 合并规则：若 X→Y和X→Z，则X→YZ在R上成立。
  + 分解规则：若X→Y在R上成立，且ZY，则X→Z在R上成立。
  + 伪传递原则：若X→Y和WY→Z在R上成立，则WX→Z在R上成立。

在关系模式R(U,F)中被F逻辑蕴含的函数依赖全体叫做F的闭包，记作F＋。

设F为属性集U上的一组函数依赖，X是U的子集，那么相对于属性集X的闭包用X＋表示，它是一个从F集使用推理规则推出的所有满足X→A的属性A的集合：

X＋={属性A|X→A在F＋中}如果G＋＝F＋，就说函数依赖集F覆盖G(F是G的覆盖，或G是F的覆盖)，或F与G相等。

如果函数依赖集F满足下列条件，则称F为一个极小函数依赖集，也称为最小依赖集或最小覆盖。

F中任一函数依赖的右部仅含有一个属性；

F中不存在这样的函数依赖X→A，使得F与F－{X→A}等价；

F中不存在这样的函数依赖X→A，X有真子集Z使得F－{X→A}∪{Z→A}与F等价。

●(2019.5)假设关系R<U, F >,U={A1,A2,A3,A4}, F ={A1A3→A2,A1A2→A3,A2→A4}那么在关系R中 (52) ，和候选关键字中必定含有属性 (53)

(52)A.有1个候选关键字A2A3 B.有1个候选关键字A2A4

C.有2个候选关键字A1A2和A1A3 D.有2个候选关键字A1A2和A2A3

(53)A.A1,其中A1A2A3主属性，A4为非主属性 B.A2,其中A2A3A4主属性，A1为非主属性

C.A2A3,其中A2A3为主属性，A1A4為为非主属性 D.A2A4,其中A2A4为主属性，A1A3为非主属性

●(2018.11)设有关系模式R(A1,A2,A3,A4,A5,A6)，函数依赖集F={ A1, -> A3 , A1,A2->A4 , A5 A6 -> A1 , A3 A5 ->A6 , A2 A5 ->A6 }。关系模式R的一个主键是(53)，从函数依赖集F可以推出关系模式R (54)

(53)A . A1, A4, B. A2 A5 C. A3 A4, D. A4,A5

(54) A.不存在传递依赖，故R为1 NlF B.不存在传递依赖，故R为2NF

C.存在传递依赖，故R为3NF D.每个非主属性完全函数依赖于主键，故R为2NF

●(2018.5)给定关系模式及R<U,F>，其中U为属性集,F是U上的一组函数依赖,那么Armstrong公理系统的伪传递律是指(51)。

(51)A.若则为所蕴涵

B.若则为所蕴涵

C.若为所蕴涵,则为所蕴涵

D.若所蕴涵,且，则XZ→YZ为F所蕴涵

●给定关系模式R(U，F)，U={A，B，C，D，E}，F={B→A，D→A，A→E，AC→B}，其属性AD的闭包为(18)，其候选关键字为(19)。 (两个重要算法，求闭包，找码)

(18) A. ADE B. ABD C. ABCD D. ACD

(19) A. ABD B.ADE C.ACD D.CD

### 范式

什么是范式？为什么要引入范式？

范式是指符合某一种级别的关系模式的类型(标准)。引入范式即是希望在设计数据库结构的过程中，力图使设计的成果符合某种范式的范式化要求，从而达到消除异常，减少数据冗余度的目的。

#### 第一范式(1NF)

如果关系模式R的每个属性值都是不可分的原子值，那么称R是第一个范式，R是规范化的关系。

例如：SLC(Sno，Sdept，Sloc，Cno，Grade)，其中Sloc为学生住处，假设每个系的学生住在同一个地方。由于关系SLC中每个属性都是不可再分的原子属性，故关系SLC满足第一范式。

在关系SLC中，存在以下函数依赖：

SnoSdept；完全函数依赖

SnoSloc；完全函数依赖

(Sno，Cno)Grade；完全函数依赖

(Sno，Cno)Sdept；部分函数依赖

(Sno，Cno)Sloc；部分函数依赖

SdeptSloc(假设每个系的学生均住同一个地方)

SnoSdept，SdeptSloc，故SnoSlo是传递函数依赖

在关系SLC中，码，主属性和非主属性分别表示为：

* + 码：(Sno，Cno)，主属性：Sno，Con。
  + 非主属性：Sdept，Sloc，Grade。

满足第一范式的关系虽然是规范化的关系，但由于非主属性对候选码除了完全函数依赖外，还存在部分函数依赖和传递函数依赖，故第一范式的关系存在一序列的问题。

表4.2SLC关系

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Sno | Sdept | Sloc | Cno | Grade |
| S1 | 计算机系 | B栋 | C1 | 80 |
| S1 | 计算机系 | B栋 | C2 | 90 |
| S1 | 计算机系 | B栋 | C3 | 75 |
| S1 | 计算机系 | B栋 | C4 | 95 |
| S2 | 数学系 | C栋 | C1 | 75 |
| S2 | 数学系 | C栋 | C2 | 95 |
| S2 | 数学系 | C栋 | C3 | 98 |
| S2 | 数学系 | C栋 | C4 | 92 |

在表4.2中，SLC关系存在如下问题：

##### 插入异常：如果要插入一个学生Sno=S3，Sdept=“数学系”，Sloc=“C栋”，但该生还未选课，即这个学生无Cno值，这样的元组数据就不能插入，因为(Sno、Cno)为主键，主键字段不能为空(实体完整性要求)。

##### 删除异常：假设学生S1不选择任何一门课程了，即学生S1对应的Cno列值都要被删除，但由于(Sno、Cno)为主键，主键字段是不能为空的(实体完整性要求)，要删除学生S1对应的Cno值，就必须连同Cno值对应的元组数据也删除才会删除成功，从而造成删除异常，即将不应删除的信息也一并删除了。

##### 修改异常：如果某个学生从数学系转到计算机系，本来只需要修改元组中的Sdept分量值，但由于关系中还有系的住处Sloc属性，学生转系时需要改变住处，还必须修改元组中Sloc分量值。另外，如果某个学生选修了多门课程，则Sdept、Sloc两列就会重复存储多次，不仅存储冗余度大，而且必须无遗漏地修改多个元组中的全部Sdept和Sloc信息，造成修改的复杂性。

##### 数据冗余度大：如果某个学生选修了K门课程，则Sdept、Sloc两列就会重复存储K次，造成数据存储冗余度大。

**总结：**SLC(满足1NF要求)不是一个好的关系模式，因为非主属性对候选码不仅有完全函数依赖，同时还包含部分函数依赖或传递函数依赖，解决的办法是将SLC分解为两个关系模式。

#### 第二范式2NF

若关系模式R1NF，且每个非主属性完全函数依赖于候选键，那么称R是2NF模式。

将上述SLC关系分解为两个新的关系SL和SC：

* + SC(Sno，Cno，Grade)
  + SL(Sno，Sdept，Sloc)

在SC关系中(Sno，Cno)Grade(码：(Sno，Cno)；主属性：Sno，Cno；非主属性：Grade)，在SC关系中，只有完全函数依赖，没有部分函数依赖和传递函数依赖。

在SL关系中SnoSdept，SdeptSloc，可得SnoSloc。其中，码是Sno，主属性是Sno，非主属性有：Sdept、Sloc。在此关系中，不仅仅包含完全函数依赖，还存在传递函数依赖。

表4.3SL关系

|  |  |  |
| --- | --- | --- |
| Sno | Sdept | Sloc |
| S1 | 计算机系 | B栋 |
| S2 | 计算机系 | B栋 |
| S3 | 计算机系 | B栋 |
| S4 | 计算机系 | B栋 |
| S5 | 数学系 | C栋 |

由于SL关系中不仅仅存在非主属性对候选码完全函数依赖，同时还存在传递函数依赖，故SL关系仍然存在如下问题：

##### 插入异常。如果要插入一个新系“历史系”，住处为“D栋”，但由于新成立的系还没有任何学生，故Sno列无具体学号值。此时该元组数据是不能插入关系SL中的，因为主键字段不能为空(实体完整性约束)。

##### 删除异常。在表4.3中，如果要删除学生S2，则必须将其对应的整个元组数据删除(S2为主键Sno的分量值，根据完整性约束要求不能为空)，这导致将应该保留的Sdept和Sloc信息也一并删除了。

##### 更新异常。在表4.3中，如果学生S1要转系，则不仅仅要修改Sdept列值，同时也要修改Sloc列值，而且由于数据冗余，必须无遗漏地修改多个元组中的全部Sdept和Sloc信息，造成修改的复杂性。

##### 数据冗余大。在表4.3中，“计算机系”和“B栋”重复存储了多次，该系的学生越多，冗余度就越大。

总结：SL(满足2NF)包含非主属性对候选码的传递函数依赖，其仍不是一个好的关系模式，可将SL关系分解为两个关系模式从而去掉非主属性对候选码的传递函数依赖关系。

#### 第三范式(3NF)

如果关系模式R1NF，且每个非主属性都不传递依赖于R的候选键，也不部分依赖于R的候选键，则称R是3NF。

例如：将上述的SL关系再一次分解成两个新的关系SD，DL。

* + SD(Sno，Sdept)
  + DL(Sdept，Sloc)

在关系SD中码是Sno，主属性是Sno，非主属性是Sdept。只有完全函数依赖SnoSdept，没有部分函数依赖和传递函数依赖。

在关系DL关系中，码是Sdept，主属性是Sdept，非主属性是Sloc。只有完全函数依赖SdeptSloc，没有部分函数依赖和传递函数依赖。

总结：由于SD，DL关系模式(满足3NF)，只有非主属性对候选码的完全函数依赖，没有部分函数依赖和传递函数依赖，已经减少了插入异常、删除异常和修改异常、数据冗余度大等问题，但可能还存在主属性对候选码的部分函数依赖或传递函数依赖，所以3NF范式也并未彻底解决各种异常问题。

#### 扩充第三范式(BCNF)

若关系模式是1NF，且每个主属性都不传递依赖于R的候选键，也不部分依赖于R的候选键，那么称R是BC范式。

例如，在关系STJ(S，T，J)中；S为学生；T为教师；J为课程。假如：

##### 每一教师只教一门课程。

##### 每门课程有若干教师教，某一学生选定某门课程，就确定了一个固定的教师。

根据以上假设条件，于是有(S，J)T；(S，T)J；TJ。其中，候选码：(S，J)和(S，T)；主属性：S，T，J；无非主属性。

STJ关系一定是3NF，但TJ，T是属于候选码(S，T)的主属性，故J是属于候选码(S，J)的主属性，故存在主属性J对候选码(S，T)的部分函数依赖，故STJ关系只是3NF范式，并非BCNF范式。STJ关系仍然存在插入异常，删除异常，更新异常，数据冗余度大等问题。

将上述STJ关系分解成ST，TJ两个新关系：

* + ST(S,T)码：ST
  + TJ(T,J)码：TJ

ST，TJ两个新关系既满足3NF的条件，同时不存在主属性对候选码的部分函数依赖或传递函数依赖，所以是BCNF范式，可解决STJ关系(符合３NF)所存在的问题。

满足一下条件的必属于BCNF范式：

* + 如果R只有一个候选码，则R必属于BCNF。
  + 如果R有多个候选码，但多个候选码不存在相交，则必属于BCNF。
  + 如果R有多个候选码，而且多个候选码间存在交集，但不存在属性(非主属性和主属性)对候选码的传递依赖和部分依赖，则R必属于BCNF。

上述四种范式之间有如下联系：1NF2NF3NFBCNF。总结如下：

* + 在1NF中，包含非主属性对候选码的完全函数依赖、部分函数依赖和传递函数依赖。
  + 在1NF基础上去除非主属性对候选码的部分函数依赖就变成2NF，2NF中有非主属性对候选码的完全函数依赖和传递函数依赖。
  + 在2NF中去掉非主属性对候选码的传递函数依赖就变成3NF，3NF中非主属性对候选码只有完全函数依赖。
  + 在1NF、2NF、3NF中都不考虑主属性对候选码是否有依赖关系，只考虑非主属性对候选码的依赖关系。
  + 在3NF中去掉主属性对候选码的部分函数依赖和传递函数依赖就变成了BCNF范式。

#### 第四范式(4NF)--了解其存在就行

定义：4NF关系模式R∈1NF，若对于R的每个非平凡多值依赖X→→Y且Y不是X的子集时，X必含有码，则关系模式R∈4NF。

注意：如果只考虑函数依赖，关系模式最高的规范化程度是BCNF，如果考虑多值依赖，关系模式最高的规范化程度是4NF。

●(2015年11月)设有关系模式R(A1，A2，A3，A4，A5，A6)，其中：函数依赖集F：{A1→A2，A1A3→A4，A5A6→A1，A2A5→A6，A3A5→A6}，则(55)是关系模式R的一个主键，R规范化程度最高达到(56)。

(55)A.A1A4 B.A2A4 C.A3A5 D.A4A5

(56)A.1NF B.2NF C.3NF D.BCNF

●(2017年11月)某企业的培训关系模式R(培训科目，培训师，学生，成绩，时间，教室)，R的函数依赖集F={培训科目->->培训师，(学生，培训科目)->成绩，(时间，教室)->培训科目，(时间，培训师)->教室，(时间，学生)->教室}。关系模式R的主键为 (52) ，其规范化程度最高达到 (53) 。

(52)A.(学生，培训科目) B.(时间，教室) C.(时间，培训师) D.(时间，学生)

(53)A. 1NF B. 2NF C. 3NF D. BCNF

## 模式分解

### 模式分解的3个定义

关系模式R(U,F)的一个分解是指，ρ={R1(U1,F1),R2(U2,F2),……, Rn(Un,Fn)},其中：U= U1 U U2 U……U Un，并且没有UiUj，1<=i,j<=n,Fi是在Ui上的投影，Fi={X→Y| X→YF+XYUi}。

对一个给定的模式进行分解，使得分解后的模式与原来的模式等价的3种情况如下：

* + 分解具有无损连接性
  + 分解要保持函数依赖
  + 分解既要有无损连接性，又要保持函数依赖

### 分解的无损连接性

定理：关系模式R(U,F)的一个分解，ρ={R1(U1,F1), R2(U2,F2)}具有无损连接的充分必要条件：

U1U2 → U1－U2 F+ 或者 U1U2 → U2－U1 F+

如果分解简单用上法，复杂则用下法：

算法：设关系模式R(A1，A2，…，An)，R的一个分解ρ={R1,R2,…,Rk},无损连接分解的判断方法如下:

#### 构造一张k行n列的表格，每列对应一个属性Aj(1jn),每行对应一个模式Ri(1ik)。

如果Aj在Ri中，那么在表格的第i行第j列处填上aj，否则填上符号bij。

#### 把表格看成模式R的一个关系，反复检查F中每个FD(函数依赖式)在表格中是否成立，若不成立，则修改表格中的元素。修改方法如下：对于F中一个FD“X－>Y”，如果表格中有行在X分量上相等，在Y分量上不相等，那么把对应行在Y分量上改成相等。如果Y的分量中有一个是aj,那么另一个也改成aj；如果没有aj，那么用其中的一个bij替换另一个(尽量把ij改成较小的数)，一直到表格不能修改为止。

#### 若修改的最后一张表格中有一行全为a，即a1，a2，a3，。。。，，an，那么p相对于F是无损连接分解，否则是有损连接分解。

●设关系模式 R为 R(H，I，J，K，L)，R 上的一个函数依赖集为 F={H→J，J→K，I→J，JL→H}，分解 (38) 是无损联接的。关系模R (A,B,C,D,E)中的关系代数表达式σ5<'2'(R) 等价于 SQL 语句 (39)。

(38) A.ρ={HK，HI，IJ，JKL，HL} B.ρ={HIL，IKL，IJL}

C.ρ={HJ，IK，HL} D.ρ={HI，JK，HL}

解析：对于A选项,构建表格如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | H | I | J | K | L |
| HK | a1 | b12 | b13 | a4 | b15 |
| HI | a1 | a2 | b23 | b24 | b25 |
| IJ | b31 | a2 | a3 | b34 | b35 |
| JKL | b41 | b42 | a3 | a4 | a5 |
| HL | a1 | b52 | b53 | b54 | a5 |

对于H→J，由于H列有三行相同，都为a1，所以把J列对应的b13，b23，b53改成J列的b13，得到下表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | H | I | J | K | L |
| HK | a1 | b12 | b13 | a4 | b15 |
| HI | a1 | a2 | b13 | b24 | b25 |
| IJ | b31 | a2 | a3 | b34 | b35 |
| JKL | b41 | b42 | a3 | a4 | a5 |
| HL | a1 | b52 | b13 | b54 | a5 |

对于J→K，由于J列上第一行、第二行、第五行对应相等，第三行、第四行也对应相等，所以修改第K列：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | H | I | J | K | L |
| HK | a1 | b12 | b13 | a4 | b15 |
| HI | a1 | a2 | b13 | a4 | b25 |
| IJ | b31 | a2 | a3 | a4 | b35 |
| JKL | b41 | b42 | a3 | a4 | a5 |
| HL | a1 | b52 | b13 | a4 | a5 |

对于I→J，由于I列上有两行相等，所以把J列对应的元素进行修改，都改为a3。对于JL→H，J列二、三、四行相等，但L列没有对应行相等，所以H列无法修改，得到下表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | H | I | J | K | L |
| HK | a1 | b12 | b13 | a4 | b15 |
| HI | a1 | a2 | a3 | a4 | b25 |
| IJ | b31 | a2 | a3 | a4 | b35 |
| JKL | b41 | b42 | a3 | a4 | a5 |
| HL | a1 | b52 | b13 | a4 | a5 |

到此，FD中的所有函数依赖式都使用完毕，此时表中没有一行全为a，所以A选项为有损连接。

对于B选项,构建表格如下：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | H | I | J | K | L |
| HIL | a1 | a2 | b13 | b14 | a5 |
| IKL | b21 | a2 | b23 | a4 | a5 |
| IJL | b31 | a2 | a3 | b34 | a5 |

对于H→J，由于H列没有相等的行，所以J列元素无法修改。对于J→K，由于J列也没有相等的行，所以K列元素也无法修改。对于I→J，由于I列全为a2，所以把J列的元素全改为a3，得到下表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | H | I | J | K | L |
| HIL | a1 | a2 | a3 | b14 | a5 |
| IKL | b21 | a2 | a3 | a4 | a5 |
| IJL | b31 | a2 | a3 | b34 | a5 |

对于JL→H，由于J列和L列都分别相等，所以把H列的所有元素修改为a1，得到下表：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | H | I | J | K | L |
| HIL | a1 | a2 | a3 | b14 | a5 |
| IKL | a1 | a2 | a3 | a4 | a5 |
| IJL | a1 | a2 | a3 | b34 | a5 |

到此结束，可以看到表中第二行元素全为a，所以b选项为无损连接。B 答案正确。

同理可证C、D为有损连接。

(39)A. SELECT\* FROM R WHERE E<'2’ B. SELECT B,E FROM R WHERE B<'2’

C. SELECT B,E FROM R HAVING E<'2’ D. SELECT\* FROM R WHERE '5'<B

### 保持函数依赖

设关系模式R(A1，A2，…，An)，R的一个分解，={R1(U1,F1), R2(U2,F2),…,Rk(Uk,Fk)},如果F+=(F1F2……Fk)+,则分解保持函数依赖。

●关系模式R(U,F)，其中U=(W,X,Y,Z)，F={WX→Y,W→X,X→Z,y→w}。关系模式R的候选码是(45)，(46)是无损连接并保持函数依赖的分解。

(45)A.W和Y　　 B.WY　　 C.WX　 D.WZ  
(46)A.ρ={R1(WY)，R2(XZ)} B.ρ={R1(WZ)，R2(XY)}　C.ρ={R1(WXY)，R2(XZ)} D．ρ={R1(WX)，R2(YZ)}

●给定关系R(A1，A2，A3，A4)上的函数依赖集F={A1A3->A2,A2->A3},R候选关键字为\_(36)\_。分解R={(A1,A2)，(A1,A3)}\_(37)\_。供选择的答案：

(36)A.A1A3 B.A1A2A3 C.A1A3A4 D.A2和A1A3

(37)A.是无损联接的 B.是保持函数依赖的C.既是无损联接又保持函数依赖D.既是有损联接又不保持函数依赖

# 数据库的安全性

为了保证数据库中数据的安全可靠和正确有效，数据库管理系统(DBMS)提供数据恢复、并发控制、数据完整性保护与数据安全性保护等功能

本节将从安全性、完整性、并发控制、恢复四个方面一一讲述，重点是完整性、并发控制。

## 安全性——防止非法用户使用数据库

数据库的安全性是指保护数据库，防止因用户非法使用数据库造成数据泄露、更改或破坏。数据库系统中的数据共享不能是无条件的共享，而必须是在DBMS统一的严格的控制之下，只允许有合法使用权限的用户访问允许他存取的数据。

常用的数据库系统四级安全模型：

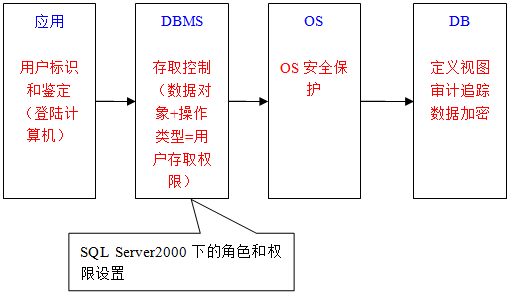


图.1数据库系统四级安全模型

注：图5.1中“应用”、“DBMS”、“OS”、“DB”表示安全级别；方框中剩余的表示保证此安全级别的常用方法。

在上述四级安全模型中，数据库安全所关心的主要是DBMS的存取控制机制，最重要的一点就是确保只授权给有资格的用户访问数据库的权限，同时令所有未被授权的人员无法接近数据。存取控制机制主要包括两部分：

* + 定义用户权限，并将用户权限登记到数据字典中。
  + 合法权限检查

每当用户发出存取数据库的操作请求后(包括操作类型、操作对象等)，DBMS查找数据字典，根据安全规则进行合法权限检查，若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一起组成了DBMS的安全子系统。

用户权限由两个要素组成，数据库对象和操作类型，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，这种定义成为授权(Authorization)。

#### GRANT语句

GRANT语句的一般格式为

GRANT <权限>[，<权限>]…

ON <对象名><对象名>[，<对象类型><对象名>]

TO <用户>[,<用户>]…

[WITH GRANT OPTION]

其语义为：对指定操作对象的指定操作权限授予指定的用户。发出该语句的可以是DBA，也可以是数据库属主，也可以是拥有该权限的用户,接受权限的用户可以是一个或多个具体用户，也可以是 PUBLIC，即全体用户。

如果指定了WITH GRANT OPTION子句，则获得某种权限的用户还可以把这种权限再授予其他用户，若没有指定WITH GRANT OPTION子句，则获得某种权限的用户只能使用该权限，不能传播该权限。

把查询Student表的权限授予用户U1

GRANT SELECT ON TABLE Student TO U1；

把对Student表和Course表的全部操作权限授予用户U2和U3.

GRANT ALLPRIVILEGES ON TABLE Student,Course TO U2，U3；

把对表SC的查询权限授予所有用户

GRANT SELECT ON TABLE SC TO PUBLIC；

把查询Student表和修改学生学号权限授给用户U4.

GRANT UPDATE(Sno)，SELECT ON TABLE Student TO U4；

把对表SC的INSERT权限授予U5用户，并允许将此权限再授予其他用户。

GRANTINSERT ON TABLE SC TOU5 WITH GRANT OPTION

#### REVOKE语句

REVOKE语句的一般格式为

REVOKE <权限>[，<权限>]…

ON <对象类型><对象名>[，<对象类型><对象名>]

FROM <用户>[,<用户>]…[CASCADE | RESTRICT]

把用户U4修改学生学号的权限收回。

REVOKE UPDATE(Sno) ON TABLE Student FROM U4

收回所有用户对表SC的查询权限。

REVOKE SELECT ON SC Student FROM PUBLIC

把用户U5对于SC表的INSERT 权限收回。

REVOKE INSERT ON TABLE SC FROM U5 CASCADE

CASCADE(级联)，即将用户U5的INSERT 权限收回的时候必须级联收回，不然系统将拒绝执行该命令。

●(2019.05)要将部门表Dept中name列的修改权限赋予用户Ming,并允许Ming将该权限授予他人。实现该要求的SQL语句如下：

GRANT UPDATE(name) ON TABLE Dept TO Ming (54) : 授予

(54)A.FOR ALL B.CASCADE C.WITH GRANT OPTION D.WITH CHECK OPTION

●(**2011.11**)将Students表的插入权限赋予用户UserA，并允许其将该权限授予他人，应使用的SQL语句为：

GRANT (51) TABLE Students TO UserA (52)；

(51)A.UPDATE B.UPDATE ON C.INSERT D.INSERT ON

(52)A.FOR ALL B.PUBLIC C.WITH CHECK OPTION D.WITH GRANT OPTION

●(2016.11)在数据库系统中,一般由DBA使用DBMS提供的授权功能为不同用户授权，其主要目的是为了保证数据库的\_(51)\_。

(51)A.正确性 B.安全性 C.一致性 D.完整性

## 完整性-防止合法用户“破坏性”使用数据库

完整性是指防止合法用户使用数据库时向数据库中加入不和语义的数据。也就是说，完整性措施的防范对象是不和语义的数据。

为了达到此目的，DBMS大都提供了一种机制来检查数据库中的数据，看其是否符合语义规定的条件。这些加在数据库数据之上的语义约束条件成为数据库完整性约束条件，它们作为模式的一部分存入数据库中。而DBMS中检查数据时候满足完整性条件的机制成为完整性检查。完整性约束条件是完整性控制机制的核心。

### 完整性控制机制的三个功能

定义功能，即提供定义完整性约束条件的机制。

检查功能，即检查用户发出的操作请求是否违背了完整性约束条件。

如果发现用户的操作请求使数据违背了完整性约束条件，则采取一定的动作来保证数据的完整性；处理的方法分为两类—立即执行的约束和延迟执行的约束。

### 完整性约束的分类

实体完整性

参照完整性

用户自定义完整性

## 并发控制机制—解决数据库的数据一致性问题

当多个用户程序并发存取统一数据时，若对并发操作不加控制就可能会存取和存储不正确的数据，破坏数据库的一致性。

#### 不提供并发控制机制可能会造成的问题

##### 丢失修改

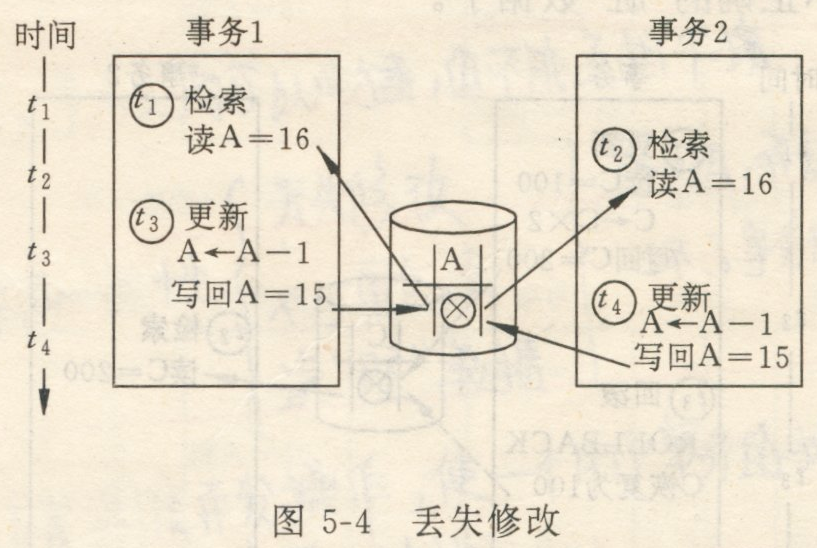


图.2丢失修改

两个事物T1和T2读入同一数据并修改，T2提交的结果破坏了T1提交的结果，导致T1的修改被丢失，如图5.2所示。

##### 不可重复读

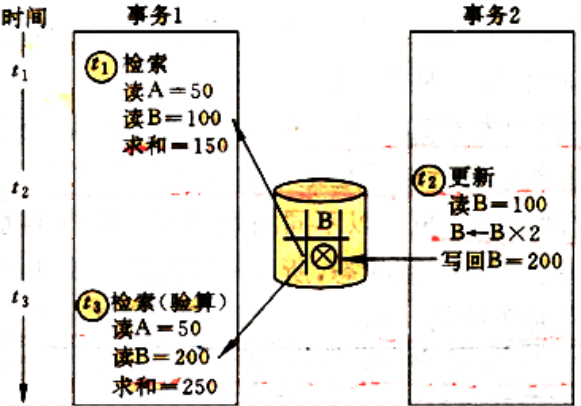


图.3不可重复读

指事务T1读取数据后，事务T2执行更新操作，使T1无法再现前一次读取结果，如图5.3所示。

##### 读脏数据

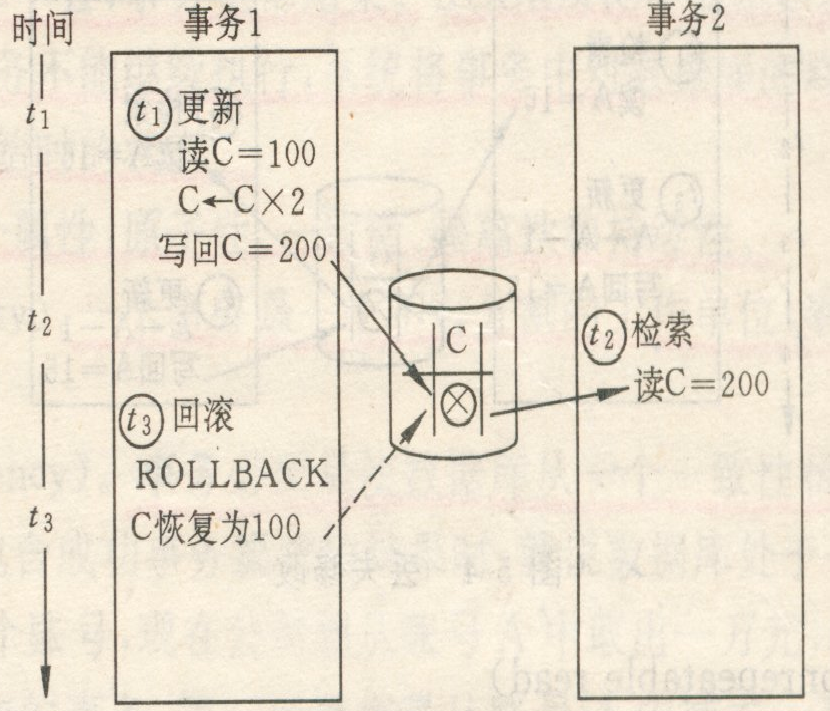


图.4脏数据

如图5.4，事务T1修改了某一数据，并将其写回磁盘，事务T2读取同一数据后，T1由于某种原因被撤销，这时T1已修改过的数据恢复原值，T2读到的数据就与数据库中的数据不一致，则T2读到的数据就为“脏”数据，即不正确的数据。

#### DBMS怎处理并发控制机制？

DBMS的并发机制是以事务为单位进行的。事务是数据库的逻辑工作单位，它是用户定义的一组操作序列。在SQL语言中，定义事务的语句有三条：

Begin Transaction

Commit

Rollback

事务具有以下的特性。

##### 原子性(Ａtomicity):数据库的逻辑工作单位。

##### 一致性(Consistency):使数据库从一个一致性的状态变到另一个一致性状态。

##### 隔离性(Isoation):不能被其他事务干扰。

##### 持续性(永久性)(Durability)：一旦提交，改变就是永久性的。

#### 怎样解决存在的问题—封锁机制

处理并发控制的主要方法是采用封锁技术。有两种封锁：排它型封锁(简称X封锁，又称写锁)和共享性封锁(简称S封锁，又称读锁)。

##### X锁：若事务T对数据对象A加上了X锁，则只允许T读取和修改A，其他任何事物都不能再对A加任何类型的锁，直到T释放A上的X锁为止。

##### S锁：若事务T对数据对象A加上了S锁，则其他事务只能对A加上S锁，即保证了在T释放A上的S锁之前不能对A做任何的修改。

#### 封锁机制的副作用—死锁和活锁

##### 活锁

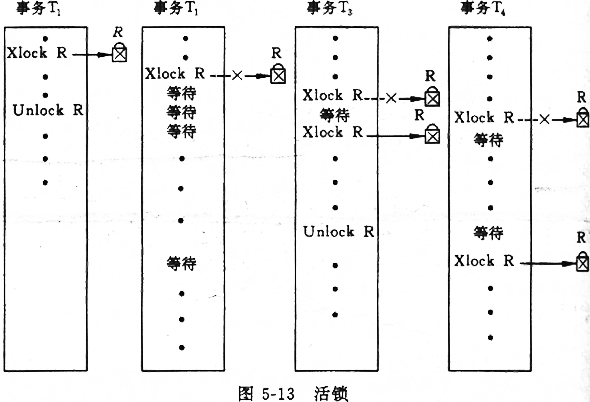


图.5活锁

解决措施—FCFS方法，当多个事务请求封锁同一数据对象时，封锁子系统按请求封锁的先后次序对事务排队，数据对象上的锁一旦释放就批准申请队列中第一个事务获得锁。

##### 死锁

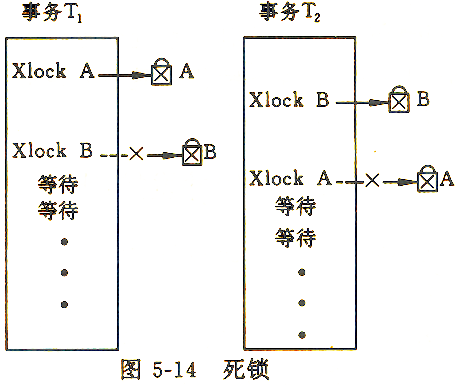


图.6死锁

目前解决死锁的方法有两类：(类似操作系统的内容)

死锁的预防——一次封锁法、顺序封锁法

死锁的诊断与解除

## 恢复机制

把数据库从错误状态恢复到某一已知正确状态的功能，称为数据库的恢复。

数据库的故障可以分为事务内部故障、系统故障、介质故障和计算机病毒造成的故障等。

●(2019.5)若事务T1对数据D1加了共享锁，事务T2T3分别对数据D2和数据D3加了排它锁， 则事务 (55) 。

(55)A.T1对数据D2D3加排它锁都成功，T2T3对数据D1加共享锁成功

B.T1对数据D2D3加排它锁都失败，T2T3对数据D1加排它锁成功

C.T1对数据D2D3加共享锁都成功，T2T3对数据D1加共享锁成功

D.T1对数据D2D3加共享锁都失败，T2T3对数据D1加共享锁成功

●(2017.5)若事务T1对数据D1加了共享锁，事务T2、T3分别对数据D2、D3加了排它锁，则事务T1对数据(51);事务T2对数据(52)。

(51)A.D2、D3加排它锁都成功 B.D2、D3加共享锁都成功

C.D2加共享锁成功，D3加排它锁失败 D.D2、D3加排它锁和共享锁都失败

(52)A.D1、D3加共享锁都失败 B.D1、D3加共享锁都成功

C.D1加共享锁成功，D3加排它锁失败 D.D1加排它锁成功，D3加共享锁失败

## 分布式数据库

## 分布式数据库系统由松散分布在多个物理站点组成，这些站点不共享物理部件，每个站点都可以参与到事务的执行中，这些事务所要访问的数据可以位于一个站点上，也可以位于几个站点上，和集中式数据库系统的区别就是集中式数据库系统的数据集中存放于一个地方，而分布式数据存放于多个不同的地方。

1.同构和异构分布式数据库

在同构分布式数据库系统中，所有站点都使用相同的数据库管理系统软件，他们彼此了解，共同合作处理用户的请求。为使事务处理能在多个站点间进行，数据库管理系统软件还必须和其他站点合作来交换和事务有关的信息。

异构分布式数据库系统中，不同站点可能使用不同的模式和不同的数据库管理系统软件，站点之间可能彼此并不了解，在合作处理事务的过程中，它们可能仅提供有限的功能，并且软件的差异成为访问多站点事务的一个障碍。

2.存储数据到数据库

将一个关系r要存储到数据库中，可以使用两种方法：

复制：系统维护这个关系的几个相同的副本，并把每个副本存储在不同的站点上。

分片：系统把关系划分为几个片，并且把每个片存储在不同的站点上。

3.透明性

分布式数据库系统的用户不需要知道数据的物理位置在哪里或者在特定的站点上数据时如何存储的，该特点称为数据的透明性，有几种形式：

分片透明：用户不需要知道一个关系是如何被分片的。

复制透明：在用户看来，每个数据对象逻辑上是唯一的，分布式系统可能为了提高系统性能或者数据可用性而复制对象，用户不需要关系什么数据对象被复制，也不必关心副本放在何处。

位置透明：用户不需要知道数据存放的物理位置，用户事务只提高数据标识符，分布式系统就能够找到任何数据。

●(2019.5)当某一场地故障时，系统可以使用其他场地上的副本而不至于使整个系统瘫痪。 这称为分布式数据库的 (56) 。

A.共享性 B.自治性 C.可用性 D.分布性

●(2018.11) 在分布式数据库中，(52) 是指用户或应用程序不需要知道逻辑上访问的表具体如何分块存储。

(52) A.逻辑透明 B.位置透明 C.分片透明 D复制透明

# 历年真题讲解

**2010年上半年**

●某销售公司数据库的零件P(零件号，零件名称，供应商，供应商所在地，库存量)关系如下表所示，其中同一种零件可由不同的供应商供应，一个供应商可以供应多种零件。零件关系的主键为(53)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 零件号 | 零件名称 | 供应商 | 供应商所在地 | 单价(元) | 库存量 |
| 010023 | P2 | S1 | 北京市海淀区58号 | 22.80 | 380 |
| 010024 | P3 | S1 | 北京市海滨区58号 | 280.00 | 1350 |
| 010022 | P1 | S2 | 陕西省西安市雁塔区2号 | 65.60 | 160 |
| 010023 | P2 | S2 | 陕西省西安市雁塔区2号 | 28.00 | 1280 |
| 010024 | P3 | S2 | 陕西省西安市雁塔区2号 | 260.00 | 3900 |
| 010022 | P1 | S3 | 北京市新城区65号 | 66.80 | 2860 |
| …… | …… | …… | …… | …… | …… |

查询各种零件的平均单价、最高单价与最低单价之间差距的SQL语句为：

SELECT零件号，(54)

FROM P

(55)；

(53)A.零件号，零件名称 B.零件号，供应商 C.零件号，供应商所在地 D.供应商，供应商所在地

(54)A.零件名称, AVG(单价)，MAX(单价)-MIN(单价) B.供应商，AVG(单价)，MAX(单价)-MIN(单价)

C.零件名称，AVG单价，MAX单价-MIN单价 D.供应商，AVG单价，MAX单价-MIN单价

(55)A.ORDER BY供应商 B.ORDER BY零件号 C.GROUP BY供应商 D.GROUP BY零件号

该关系存在冗余以及插入异常和删除异常等问题。为了解决这一问题需要将零件关系分解为(56)。

(56)A.P1(零件号，零件名称，单价)、P2(供应商，供应商所在地，库存量)

B.P1(零件号，零件名称)、P2(供应商，供应商所在地，单价，库存量)

C.P1(零件号，零件名称)、P2(零件号，供应商，单价，库存量)、P3(供应商，供应商所在地)

D.P1(零件号，零件名称)、P2(零件号，单价，库存量)、P3(供应商，供应商所在地)、P4(供应商所在地，库存量)

**2010年下半年**

●在某企业的营销管理系统设计阶段，属性“员工”在考勤管理子系统中被称为“员工”，而在档案管理子系统中被称为“职工”，这类冲突称为(51)冲突。

(51)A.语义 B.结构 C.属性 D.命名

●设有学生实体Students(学号，姓名，性别，年龄，家庭住址，家庭成员，关系，联系电话)，其中“家庭住址”记录了邮编、省、市、街道信息；“家庭成员，关系，联系电话”分别记录了学生亲属的姓名、与学生的关系以及联系电话。

学生实体Students中的“家庭住址”是一个(52)属性；为使数据库模式设计更

合理，对于关系模式Students (53)。

(52)A.简单 B.多值 C.复合 D.派生

(53)A.可以不作任何处理，因为该关系模式达到了3NF

B.只允许记录一个亲属的姓名，与学生的关系以及联系电话的信息

C.需要对关系模式Students增加若干组家庭成员、关系及联系电话字段

D.应该将家庭成员、关系及联系电话加上学生号，设计成为一个独立的实体

●设有关系模式R(课程，老师，学生，成绩，时间，教室)，其中函数依赖集F如下：

F=(课程→→教师，(学生，课程)→成绩，(时间，教室)→课程，(时间，老师)→教室，(时间，学生)→教室

关系模式R的一个主键是(54)，R规范化程度最高达到(55)。若将关系模式R分解为3个关系模式R1(课程，老师)、R2(学生，课程，成绩)、R3(学生，时间，教室，课程)，其中R2的规范化程度最高达到(56)。

(54)A.(学生，课程) B.(时间，教室) C.(时间，老师) D.(时间，学生)

(55)A.INF B.2NF C.3NF D.BCNF

(56)A.2NF B.3NF C.BCNF D.4NF

**2011年上半年**

●某医院数据库的部分关系模式为：科室(科室号，科室名，负责人，电话)、病患(病历号，姓名，住址，联系电话)和职工(职工号，职工姓名，科室号，住址，联系电话)。假设每个科室有一位负责人和一部电话，每个科室有若干名职工，一名职工只属于一个科室；一个医生可以为多个病患看病；一个病患可以由多个医生多次诊治。

科室与职工的所属联系类型为(51)，病患与医生的就诊联系类型为(52)。对于就诊联系最合理的设计是(53)，就诊关系的主键是(54)。

(51)A.1:1 B.1:n C.n:1 D.n:m

(52)A.1:1 B.1:n C.n:1 D.n:m

(53)A.就诊(病历号，职工号，就诊情况) B.就诊(病历号，职工姓名，就诊情况)

C.就诊(病历号，职工号，就诊时间，就诊情况) D.就诊(病历号，职工姓名，就诊时间，就诊情况)

(54)A.病历号，职工号 B.病历号，职工号，就诊时间

C.病历号，职工姓名 D.病历号，职工姓名，就诊时间

●给定关系模式R<U，F>，U={A,B,C}，F={AB→C，C→B}。关系R(55)。且分别有(56)。

(55)A.只有1个候选关键字AC B.只有1个候选关键字AB

C.有2个候选关键字AC和BC D.有2个候选关键字AC和AB

(56)A.1个非主属性和2个主属性 B.2个非主属性和1个主属性

C. 0个非主属性和3个主属性 D. 3个非主属性和0个主属性

**2012年上半年**

●某销售公司数据库的零件关系P(零件号，零件名称，供应商，供应商所在地，库存量)，函数依赖集F={零件号→零件名称，(零件号，供应商)→库存量，供应商→供应商所在地}。零件关系模式P属于 (54) 。

查询各种零件的平均库存量、最多库存量与最少库存量之间差值的SQL语句如下：

SELECT零件号，零件名称， (55)

FROM P

(56) ；

(54)A.INF B.2NF C.3NF D.4NF

(55)A.AVG(库存量) AS平均库存量，MAX(库存量)-MIN(库存量) AS差值

B.平均库存量AS AVG(库存量)，差值AS MAX(库存量)-MIN(库存量)

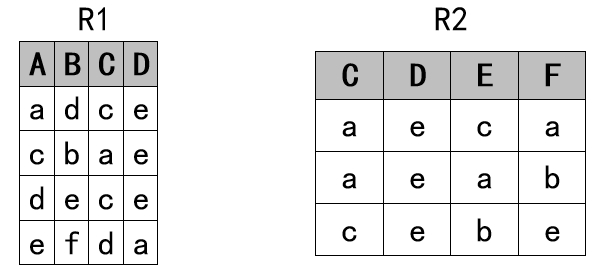
C.AVG库存量AS平均库存量，MAX库存量-MIN库存量AS差值

D.平均库存量AS AVG库存量，差值AS MAX库存量-MIN库存量

(56)A.ORDER BY供应商 B.ORDER BY零件号 C.GROUP BY供应商 D.GROUP BY零件号

**2012年下半年**

●关系R1和R2如下图所示：



若进行R1R2运算，则结果集为(51)元关系，共有(52)个元组。

(51)A．4 B．5 C．6 D．7

(52)A．4 B．5 C．6 D．7

●设有关系模式R(E，N，M，L，Q)，其函数依赖集为F={E—>N，EM—>Q，M—>L}。则关系模式R达到了(53)；该关系模式(54)。

(53)A.IN F B.2NF C.3NF D.BCNF

(54)A．无需进行分解，因为已经达到了3NF B．无需进行分解，因为已经达到了BCNF

C．尽管不存在部分函数依赖，但还存在传递依赖，所以需要进行分解

D．需要进行分解，因为存在冗余、修改操作的不一致性、插入和删除异常

●已知关系模式：图书(图书编号，图书类型，图书名称，作者，出版社，出版日期，ISBN)，图书编号唯一识别一本图书。建立“计算机”类图书的视图Computer-BOOK，并要求进行修改、插入操作时保证该视图只有计算机类的图书。实现上述要求的SQL语句如下：

CREATE (55)AS SELECT图书编号，图书名称，作者，出版社，出版日期

FROM图书

WHERE图书类型=‘计算机’(56)

(55)A．TABLE Computer-BOOK B．VIEW Computer-BOOK

C．Computer-BOOK TABLE D． Computer-BOOK VIEW

(56)A．FOR ALL B.PUBLIC C．WITH CHECK OPTIOND D．WITH GRANT OPTION

**2013年上半年**

●假设学生Students和教师Teachers关系模式如下所示：

students(学号，姓名，性别，类别，身份证号)

Teachers(教师号，姓名，性别，身份证号，工资)

其中，学生关系中的类别分为“本科生”和“研究生”两类。

a.查询在读研究生的教师的平均工资、最高与最低工资之间差值的SQL语句如下：

SELECT (56)

FROMStudents,TCachers

WHERE (57) ;

(56)A.AVG(工资)AS平均工资，MAX(工资)- MIN(工资)AS差值

B.平均工资AS AVG(工资)，差值AS MAX(工资)- MIN(工资)

C.AVG(工资)ANY平均工资，MAX(工资)-MIN(工资)ANY差值

D.平均工资ANY AVG(工资),差值ANY MAX(工资)- MIN(工资)

(57)A.Students.身份证号＝Teachers.身份证号

B.students.类别＝’研究生’

C.Students.身份证号＝Teachers.身份证号AND Students.类别=’研究生’

D.Studenis.身份证号＝Teachers.身份证号OR Students.类别＝’研究生’

b.查询既是女教师，又是研究生且工资大于等于3500元的身份证号和姓名的SQL语句如下：

(SELECT 身份证号，姓名

FROM Students

WHERE (58) )

INTERSECT

(SELECT身份证号，姓名

FROM Teachers

WHERE (58) )

(58)A.工资＞=3500 B.工资＞= '3500’

C.性别＝女AND类别＝研究生 D.性别＝’女’AND 类别 = ’研究生’

(59)A.工资＞=3500 B.工资＞='3500’

C.性别＝女AND类别＝研究生 D.性别＝’女’AND 类别=’研究生’

**2013年下半年**

●若有关系R(A，B，C，D，E)和S(B，C，F，G)，则R和S自然连结运算后的属性列有(51)个，与表达式等价的SQL语句如下：

SELECT (52) FROM (53) WHERE (54)；

(51)A.5 B.6 C.7 D.9

(52)A.A,R.C,F,G B. A,C,S.B,S.F C.A,C,S.B,S.C D. R,A,R.C,S.B,S.C

(53)A.R B. S C. RS D. R,S

(54)A.R.B=S.B AND R.C=S.C AND R.C<S.B B.R.B=S.B AND R.C=S.C AND R.C<S.F

C.R.B=S.B OR R.C=S.C OR R.C<S.B D.R.B=S.B OR R.C=S.C OR R.C<S.F

●分布式数据库系统中，(55)是指用户无需知道数据存放的物理位置。

(55)A.分片透明 B.复制透明 C.逻辑透明 D.位置透明

●计算机系统的软硬件故障可能会造成数据库中的数据被破坏，为了防止这一问题，通常需要(56)，以便发生故障时恢复数据库。

(56)A.定期安装DBMS和应用程序 B.定期安装应用程序，并将数据库做镜像

C.定期安装DBMS，并将数据库作备份

D.定期将数据库作备份，在进行事务处理时，需要将数据更新写入日志文件

**2014年上半年**

●为了保证数据库中数据的安全可靠和正确有效，系统在进行事务处理时，对数据的插入、删除或修改的全部有关内容先写入(51)；当系统正常运行时，按一定的时间间隔，把数据库缓冲区内容写入(52)；当发生故障时，根据现场数据内容及相关文件来恢复系统的状态。

(51)A．索引文件 B．数据文件 C．日志文件 D．数据字典

(52)A．索引文件 B．数据文件 C．日志文件 D．数据字典

●“当多个事务并发执行时，任一事务的更新操作直到其成功提交的整个过程对其他事务都是不可见的”，这一性质通常被称为事务的 (53)。

(53)A．原子性 B．一致性 C．隔离性 D．持久性

●假定某企业2014年5月的员工工资如下表所示：

2014年5月员工工资表

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 员工号 | 姓名 | 部门 | 基本工资 | 岗位工资 | 全勤奖 | 应发工资 | 扣款 | 实发工资 |
| 1001 | 王小龙 | 办公室 | 680.00 | 1200.00 | 100.00 | 1980.00 | 20.00 | 1960.00 |
| 1002 | 孙晓红 | 办公室 | 1200.00 | 1000.00 | 0.00 | 2200.00 | 50.00 | 2150.00 |
| 2001 | 赵眙珊 | 企划部 | 680.00 | 1200.00 | 100.00 | 1980.00 | 10.00 | 1970.00 |
| 2002 | 李丽敏 | 企划部 | 950.00 | 2000.00 | 100.00 | 3050.00 | 15.00 | 3035.00 |
| 3002 | 傅学君 | 设计部 | 800.00 | 1800.00 | 0.00 | 2600.00 | 50.00 | 2550.00 |
| 3003 | 曹海军 | 设计部 | 950.00 | 1600.00 | 100.00 | 2650.00 | 20.00 | 2630.00 |
| 3004 | 赵晓勇 | 设计部 | 1200.00 | 2500.00 | 0.00 | 3700.00 | 50.00 | 3650.00 |
| 4001 | 杨一凡 | 销售部 | 680.00 | 1000.00 | 100.00 | 1780.00 | 10.00 | 1770.00 |
| 4003 | 景昊星 | 销售部 | 1200.00 | 2200.00 | 100.00 | 3500.00 | 20.00 | 3480.00 |
| 4005 | 李建军 | 销售部 | 850.00 | 1800.00 | 100.00 | 2750.00 | 98.00 | 2652.00 |

套询人数大于2的部门和部门员工应发工资的平均工资的SQL语句如下：

SELECT(54)

FROM工资表

(55)

(56)；

(54)A．部门，AVG(应发工资)AS平均工资 B．姓名，AVG(应发工资)AS平均工资

C．部门，平均工资ASAVG(应发工资) D．姓名，平均工资ASAVG(应发工资)

(55)A．ORDERBY姓名 B．ORDERBY部门 C．GROUPBY姓名 D．GROUPBY部门

(56)A．WHERE COUNT(姓名)>2 B．WHERE COUNT(DISTINCT(部门))>2

C．HAVING COUNT(姓名)>2 D．HAVING COUNT(DISTINCT(部门))>2

**2014年下半年**

●在数据库逻辑结构设计阶段，需要(51)阶段形成的(52)作为设计依据。

(51)A.需求分析 B.概念结构设计 C.物理结构设计 D.数据库运行和维护

(52)A.程序文档、数据字典和数据流图 B.需求说明文档、程序文档和数据流图

C.需求说明文档、数据字典和数据流图 D.需求说明文档、数据字典和程序文档

●给定关系模式R(A，B，C，D)、S(C，D，E)，与π1,3,5(σ2=＇软件工程＇(R⋈S))等价的SQL语句如下：

SELECT(53)FROMR,SWHERE(54)；

(53)A.A，C，S.C B.A，B，E C.A，R.C，E D.A，R.C，S.D

(54)A.B=软件工程ORR.C=S.CANDR.D=S.D B.B=‘软件工程’ORR.C=S.CANDR.D=S.D

C.B=‘软件工程’ORR.C=S.CORR.D=S.D D.B=‘软件工程’ANDR.C=S.CANDR.D=S.D

下列查询B=“信息”且E=“北京”的A、B、E的关系代数表达式中，查询效率最高的是(55)。

(55)A.π1,2,7(σ2=‘信息'⋀3=5⋀4=6⋀7=‘北京'(R×S)) B.π1,2,7(σ3=5∧4=6(σ2=‘信息’(R)×σ5=‘北京’(S)))

C.π1,2,7(σ3=5∧4=6∧2=‘’(R×σ7=‘’(S))) D.π1,2,7(σ3=5∧4=6∧7=‘北京’(σ2=‘信息’(R)×S))

**2015年上半年**

●给定关系模式R(,,,)上的函数依赖集F={→,→}。若将R分解为ρ={(,),(,)}，则该分解是(52)的。

(52)A.无损联接且不保持函数依赖 B.无损联接且保持函数依赖

C.有损联接且保持函数依赖 D.有损联接且不保持函数依赖

**2015年下半年**

●在数据库逻辑设计阶段，若实体中存在多值属性，那么将E-R图转换为关系模式时，(52)，得到的关系模式属于4NF。

(52) A.将所有多值属性组成一个关系模式 B.使多值属性不在关系模式中出现

C.将实体的码分别和每个多值属性独立构成一个关系模式

D.将多值属性和其它属性一起构成该实体对应的关系模式

●在分布式数据库中有分片透明、复制透明、位置透明和逻辑透明等基本概念，其中：(53)是指局部数据模型透明，即用户或应用程序无需知道局部使用的是哪种数据模型；(54)．是指用户或应用程序不需要知道逻辑上访问的表具体是如何分块存储的。

(53)A.分片透明 B.复制透明 C.位置透明 D.逻辑透明

(54)A.分片透明 B.复制透明 C.位置透明 D.逻辑透明

2016年上半年

●关系规范化在数据库设计的(52)阶段进行。

(52)A.需求分析 B.概念设计 C.逻辑设计 D.物理设计

●若给定的关系模式为R，U={A,B,C}，F={AB->C,C->B}，则关系R(53)。

(53)A.有2个候选关键字AC和BC，并且有3个主属性

B.有2个候选关键字AC和AB，并且有3个主属性

C.只有1个候选关键字AC，并且有1个非主属性和2个主屬性

D.只有1个候选关键字AB，并且有1个非主属性和2个主属性

●某公司数据库中的元件关系模式为P(元件号,元件名称,供应商,供应商所在地,库存量)，函数依赖集F如下所示：F={元件号->元件名称，(元件号,供应商)->库存量,供应商->供应商所在地}，元件关系的主键为(54)，该关系存在冗余以及插入异常删除异常等问题。为了解决这一问题需要将元件关系分解为(55)，分解后的关系模式可以达到(56)。

(54)A.元件号，元件名称B.元件号，供应商C.元件号，供应商所在地D.供应商，供应商所在地

(55)A.元件1(元件号，元件名称，库存量)、元件2(供应商，供应商所在地)

B.元件1(元件号，元件名称)、元件2(供应商，供应商所在地，库存量)

C.元件1(元件号，元件名称)、元件2(元件号，供应商，库存量)、元件3(供应商，供应商所在地)

D.元件1(元件号，元件名称)、元件2(元件号，库存量)、元件3(供应商，供应商所在地)、元件4(供应商所在地，库存量)

(56)A.1NF B.2NF C.3NF D.4NF

**2016年下半年**

●给定关系模式R(U，F),其中：U为关系模式R中的属性集，F是U上的一组数依赖。假设U=｛，，，｝，F=｛，，｝，那么关系R的主键为\_(52)\_。函数依赖集F中\_(53)\_的是冗余的。

(52)A. B. C. D.

(53)A. B. C. D.

**2017年上半年**

●假设关系R<U,F>,U={A1,A2,A3}，F={A1A3→A2,A1A2→A3}，则关系R的各候选关键字中必定含有属性(53)。

(53)A.A1 B.A2 C.A3 D.A2A3

**2017年下半年**

●设关系模式R(U, F)，其中:U= {A, B, C, D, E} } F={A->B, DE->B, CB->E, E->A,B->D。 (54) 为关系模式R的候选关键字。分解 (55) 是无损连接，并保持函数依赖的。

(54)A. AB B. DE C. DB D. CE

(55)A. p={R1(AC), R2(ED), R3 (B)} B. p={R1 (AC), R2 (E), R3(DB)}

C. p={R1(AC), R2(ED), R3(AB)} D. p={R1 (ABC), R2 (ED), R3(ACE)}