高等学校计算机应用规划教材

# SQL Server 2019 数据库教程

主　编　于晓鹏
副主编　于　萍　于　淼
　　孙启隆　齐长利

清华大学出版社

北　京

## 内 容 简 介

本书从 SQL Server 2019 的基本概念出发，由浅入深地讲述了该数据库系统的安装过程、服务器的配置技术、Transact-SQL 语言、系统安全机制、数据库管理、各种数据库对象的管理，以及索引技术、数据更新技术、数据完整性技术、数据复制技术、数据互操作性技术、性能监视和调整技术、并发性技术等内容。在讲述 SQL Server 的各种技术时，运用了丰富的实例，注重培养学生解决问题的能力并快速掌握 SQL Server 的基本操作技术。

本书内容丰富、结构合理、思路清晰、语言简练流畅、实例翔实。每章正文结合所讲述的关键技术和难点，精选极富价值的示例；每章末尾都安排了有针对性的习题，以巩固所学基本概念，培养学生的实际动手能力，增强对基本概念的理解和实际应用能力。

本书主要面向数据库初学者，可作为高等院校的数据库课程教材，也可作为数据库培训班的培训教材，还可作为数据库应用程序开发人员的参考资料。



图书在版编目(CIP)数据

SQL Server 2019 数据库教程 / 于晓鹏　主编. —北京：清华大学出版社，2020.6
高等学校计算机应用规划教材
ISBN 978-7-302-55439-4

Ⅰ. ①S…　Ⅱ. ①于…　Ⅲ. ①关系数据库系统－高等学校－教材　Ⅳ. ①TP311.132.3

中国版本图书馆 CIP 数据核字(2020)第 082351 号

**责任编辑**：王　定
**封面设计**：孔祥峰
**版式设计**：思创景点
**责任校对**：成凤进
**责任印制**：丛怀宇

**出版发行**：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
**印 装 者**：三河市君旺印务有限公司
**经　　销**：全国新华书店
**开　　本**：185mm×260mm　　**印　　张**：16.75　　**字　　数**：387 千字
**版　　次**：2020 年 7 月第 1 版　　**印　　次**：2020 年 7 月第 1 次印刷
**定　　价**：58.00 元

产品编号：077584-01

# 前　言

信息技术的飞速发展大力推进了社会的进步，也逐渐改变了人类的生活、工作和学习方式。数据库技术和网络技术是信息技术中的两大重要支柱。自 20 世纪 70 年代以来，数据库技术的发展使得信息技术的应用从传统的计算方式转变到现代化的数据管理方式。在当前热门的信息系统开发领域，如管理信息系统、企业资源计划、供应链管理系统和客户关系管理系统等，都可以看到数据库技术应用的影子。

作为一个关系型数据库管理系统，SQL Server 不断采纳新技术来满足用户日益增长和变化的需要，产品的功能越来越强大，用户使用起来越来越方便，系统的可靠性也越来越高，从而使得该产品的应用越来越广泛。在我国，SQL Server 的应用已经深入银行、邮电、电力、铁路、气象、公安、军事、航天、税务、教育等众多行业和领域。SQL Server 为用户提供了完整的数据库解决方案，可以帮助用户建立自己的商务体系，增强用户对外界变化的敏捷反应能力，以提高用户的竞争力。

本书从 SQL Server 2019 的基本概念出发，由浅入深地讲述了该系统的安装过程、服务器的配置技术、Transact-SQL 语言、系统安全机制、数据库管理、各种数据库对象的管理，以及索引技术、数据更新技术、数据完整性技术、数据复制技术、数据互操作性技术、性能监视和调整技术、并发性技术等内容。

在讲述 SQL Server 的各种技术时，运用了丰富的实例，注重培养学生解决问题的能力并快速掌握 SQL Server 的基本操作技术。

本书内容丰富、结构合理、思路清晰、语言简练流畅、实例翔实。在每一章的正文中，结合所讲述的关键技术和难点，精选极富价值的示例。每一章末尾都安排了有针对性的习题，以巩固所学基本概念，培养学生的实际动手能力，增强对基本概念的理解和实际应用能力。

本书主要面向数据库初学者，可作为高等院校的数据库课程教材，也可作为数据库培训班的培训教材，还可作为数据库应用程序开发人员的参考资料。

本书由吉林师范大学于晓鹏主编，负责全书的策划、编写、统稿和定稿工作。参与本书编写的还有于萍、于淼、孙启隆、齐长利等，其中，第 1、2 章由于晓鹏编写，第 3～5 章由于萍编写，第 6～8 章由于淼编写，第 9～10 章由齐长利编写，第 11～12 章由孙启隆编写。

由于编者水平有限，时间仓促，本书难免存在疏漏之处，敬请读者指正。

本书课件、习题参考答案下载地址：

课件

习题参考答案

编　者

2020 年 3 月

# 目　录

# 第 1 章　数据库基础

随着科学技术和社会经济的飞速发展，人们掌握的信息量急剧增加，要充分地开发和利用这些信息资源，就必须有一种新技术，能对大量的信息进行识别、存储、处理与传播。随着计算机软硬件技术的发展，20 世纪 60 年代末，数据库技术应运而生，并从 20 世纪 70 年代起得到了迅速的发展和广泛的应用。

数据库是数据管理的有效技术，是计算机科学的重要分支。如今，信息资源已成为各个部门的重要财富和资源，建立一个满足各级部门信息处理要求的行之有效的信息系统也成为一个企业或组织生存和发展的重要条件。因此，作为信息系统核心和基础的数据库技术得到越来越广泛的应用，从小型单项事务处理系统到大型信息系统，从联机事务处理(On-Line Transaction Processing，OLTP)到联机分析处理(On-Line Analysis Processing，OLAP)，从一般企业管理到计算机辅助设计与制造(Computer Aided Design，CAD/Compter Aided Manufacturing，CAM)、计算机集成制造系统(Computer Integrated Manufacturing System，CIMS)、电子政务(e-Government)、电子商务(e-Commerce)、地理信息系统(Geographic Information System，GIS)等，越来越多的应用领域采用数据库技术来存储和处理信息资源。特别是随着互联网的发展，广大用户可以直接访问并使用数据库，例如通过网络订购图书、日用品、机票、火车票，通过网上银行转账存款取款、检索和管理账户等。数据库已经成为每个人生活中不可缺少的部分。

数据库技术主要研究如何科学地组织和存储数据，如何高效地获取和处理数据。数据库技术作为数据管理的最新技术，目前已被广泛应用于各个领域。如今，数据库的建设规模、数据库信息量的大小和使用频度，已经成为衡量一个国家信息化程度的重要标志。

## 1.1　数据库系统基本概念

本节主要介绍有关数据库的信息、数据、数据处理、数据库、数据库管理系统和数据库系统等概念。

### 1.1.1　信息

信息是人脑对现实世界中的客观事物及事物之间联系的抽象反映，它向人们提供了关于现实世界实际存在的事物及其联系的有用知识。

### 1.1.2　数据

数据是人们用各种物理符号，把信息按一定格式记载下来的有意义的符号组合。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应就是数字，例如 93、

1000、99.5、−330.86、￥60、$726 等。其实数字只是最简单的一种数据，是对数据的传统和狭义的理解。广义的理解认为数据的种类很多。例如，文本(text)、图形(graph)、图像(image)、音频(audio)、视频(video)、学生的档案记录、货物的运输情况等都是数据。

可以对数据做如下定义：描述事物的符号记录被称为数据。描述事物的符号可以是数字，也可以是文字、图形、图像、音频、视频等。数据有多种表现形式，它们都可以经过数字化后存入计算机。

在现代计算机系统中，数据的概念是广义的。早期的计算机系统主要用于科学计算，处理的数据是数值型数据，如整数、实数、浮点数等。现在计算机存储和处理的对象十分广泛，表示这些对象的数据也随之变得越来越复杂。

数据的表现形式还不能完全表达其内容，需要经过解释，数据和关于数据的解释是不可分的。例如，88 是一个数据，它可以是一个学生某门课的成绩，也可以是某个人的体重，还可以是某个班的学生人数。数据的解释是指对数据含义的说明，数据的含义称为数据的语义，数据与其语义是不可分的。

在日常生活中，人们可以直接用自然语言(如汉语)来描述事物。例如，日常生活中我们这样描述某校软件工程专业一位学生的基本情况：张三，男，1999 年 5 月生，吉林省四平市人，2018 年入学。这在计算机中常常如下描述：

(张三，男，199905，吉林省四平市，软件工程，2018)

即把学生的姓名、性别、出生年月、出生地、所在专业、入学时间等组织在一起，构成一个记录。这里的学生记录就是描述学生的数据，这样的数据是有结构的。记录是计算机中表示和存储数据的一种格式或一种方法。

### 1.1.3　数据处理

数据处理是指对各种形式的数据进行收集、整理、加工、存储和传播的一系列活动的总和。其目的之一是从大量的原始数据中提取出对人们有价值的信息，作为行动和决策的依据；目的之二是借助计算机科学地保存和管理大量的复杂数据，以便人们能利用这些信息资源。

### 1.1.4　数据库

数据库(Database，DB)，顾名思义，是存放数据的仓库。只不过这个仓库是在计算机存储设备上，而且数据是按一定的格式存放的。

人们收集并抽取出一个应用所需要的大量数据之后，应将其保存起来，以供进一步加工处理，抽取有用信息。在科学技术飞速发展的今天，人们的视野越来越广，数据量急剧增加。过去人们把数据存放在文件柜里，现在人们借助计算机和数据库技术科学地保存和管理大量复杂的数据，以便能方便而充分地利用这些宝贵的信息资源。

严格地讲，数据库是长期储存在计算机内、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、描述和储存，具有较小的冗余度(redundancy)、较高的数据独立性(data independency)和易扩展性(scalability)，并可为各种用户共享。

概括地讲，数据库数据具有永久存储、有组织和可共享 3 个基本特点。

### 1.1.5　数据库管理系统

数据库管理系统(Database Management System，DBMS)是一种系统软件，介于应用程序和操作系统之间，用于帮助人们管理输入计算机中的大量数据。例如，用于创建数据库，向数据库中存储数据，修改数据库中的数据，从数据库中提取信息，等等。具体来说，一个数据库管理系统应具备如下功能。

(1) 数据定义功能。数据库管理系统提供数据定义语言(Data Definition Language，DDL)，用户通过它可以方便地对数据库中的数据对象的组成与结构进行定义。

(2) 数据组织、存储和管理。数据库管理系统要分类组织、存储和管理各种数据，包括数据字典、用户数据、数据的存取路径等。用户要确定以何种文件结构和存取方式在存储器上组织这些数据，如何实现数据之间的联系。数据组织和存储的基本目标是提高存储空间利用率和方便存取，提供多种存取方法(如索引查找、hash 查找、顺序查找等)来提高存取效率。

(3) 数据操纵功能。数据库管理系统还提供数据操纵语言(Data Manipulation Language，DML)，用户可以使用它操纵数据，实现对数据库的基本操作，如查询、插入、删除和修改等。

(4) 数据库的事务管理和运行管理。数据库在建立、运用和维护时由数据库管理系统统一管理和控制，以保证事务的正确运行，保证数据的安全性、完整性、多用户对数据的开发使用及发生故障后的系统恢复。

(5) 数据库的建立和维护功能。包括数据库初始数据的输入、转换功能，数据库的转储、恢复功能，数据库的重组织功能和性能监视、分析功能等。这些功能通常是由一些实用程序或管理工具完成的。

(6) 其他功能。包括数据库管理系统与网络中其他软件系统的通信功能，一个数据库管理系统与另一个数据库管理系统或文件系统的数据转换功能，异构数据库之间互访和互操作功能，等等。

数据库管理系统在计算机系统中的地位如图 1-1 所示。它运行在一定的硬件和操作系统平台上。人们可以使用一定的开发工具，利用 DBMS 提供的功能，创建满足实际需求的数据库应用系统。

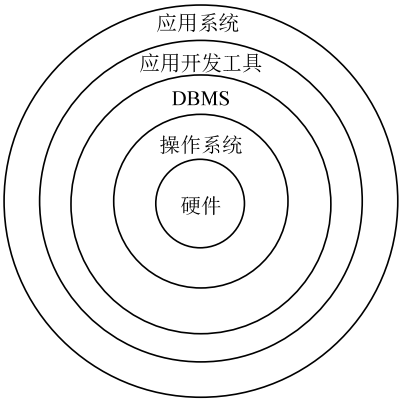


图 1-1　数据库管理系统在计算机系统中的地位

根据对信息的组织方式的不同，数据库管理系统可以分为关系、网状和层次 3 种类型。目前使用最多的数据库管理系统是关系型数据库管理系统(RDBMS)。例如，SQL Server、Oracle、Sybase、Visual FoxPro、DB2、Informix、Ignres 等都是常见的关系数据库管理系统。

### 1.1.6　数据库系统

数据库系统(Database System，DBS)是由数据库、数据库管理系统(及其应用开发工具)、应用程序和数据库管理员(Database Administrator，DBA)组成的存储、管理、处理和维护数据的系统。应当指出的是，数据库的建立、使用和维护等工作只靠一个数据库管理系统远远不够，还要有专门的人员来完成，这些人被称为数据库管理员。

在数据库系统中，数据库提供数据的存储功能，数据库管理系统提供数据的组织、存取、管理和维护等基础功能，数据库应用系统根据应用需求使用数据库，数据库管理员负责全面管理数据库系统。

## 1.2　数据管理技术的发展

数据库技术是应数据管理任务的需要而产生的。数据管理是指对数据进行分类、组织、编码、存储、检索和维护，它是数据处理的中心问题。

在应用需求的推动下，在计算机硬件、软件发展的基础上，数据管理技术经历了人工管理、文件系统、数据库系统 3 个阶段。

### 1.2.1　人工管理阶段

20 世纪 50 年代中期以前，计算机主要用于科学计算。当时的硬件状况是：外存只有纸带、卡片、磁带，没有磁盘等直接存取存储设备；软件状况是：没有操作系统，没有管理数据的专门软件；数据处理方式是批处理。批处理(Batch)是一种简化的脚本语言，它应用于 DOS 和 Windows 系统中，是由 DOS 或者 Windows 系统内嵌的命令解释器(通常是 COMMAND.COM 或者 CMD.EXE)解释运行，类似于 UNIX 中的 Shell 脚本。批处理文件具有.bat 或者.cmd 的扩展名，其最简单的例子是逐行书写在命令行中用到的各种命令。更复杂的情况，它需要使用 if、for、goto 等命令控制程序的运行过程，如同 C、BASIC 等高级语言一样。如果需要实现更复杂的应用，利用外部程序是必要的，这包括系统本身提供的外部命令和第三方提供的工具或者软件。批处理程序虽然是在命令行环境中运行，但不仅能使用命令行软件，任何 32 位的 Windows 程序都可以放在批处理文件中运行。人工管理数据具有如下特点。

(1) 数据不保存。由于当时计算机主要用于科学计算，一般不需要将数据长期保存，只是在计算某一课题时将数据输入，用完就撤走。它不仅对用户数据如此处置，对系统软件有时也是这样。

(2) 应用程序管理数据。数据需要由应用程序自己设计、说明(定义)和管理，没有相应

的软件系统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。因此，程序员负担很重。

(3) 数据不共享。数据是面向应用程序的，一组数据只能对应一个程序。当多个应用程序涉及某些相同的数据时必须各自定义，无法互相利用、互相参照，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，数据完全依赖于应用程序，称之为数据缺乏独立性，这就加重了程序员的负担。

在人工管理阶段，应用程序与数据之间的一一对应关系如图 1-2 所示。

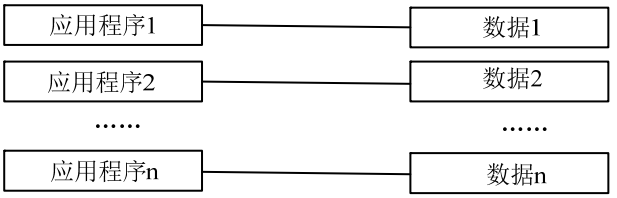


图 1-2　人工管理阶段应用程序与数据之间的对应关系

### 1.2.2　文件系统阶段

20 世纪 50 年代后期到 60 年代中期，这时硬件方面已有了磁盘、磁鼓等直接存取存储设备；软件方面，操作系统中已经有了专门的数据管理软件，一般称其为文件系统；处理方式上不仅有了批处理，而且能够联机实时处理。

#### 1. 文件系统优点

使用文件系统管理数据具有如下优点。

(1) 数据可以长期保存。由于计算机大量用于数据处理，数据需要长期保留在外存上反复进行查询、修改、插入和删除等操作。

(2) 由文件系统管理数据。由专门的软件即文件系统进行数据管理，文件系统把数据组织成相互独立的数据文件，利用“按文件名访问，按记录进行存取”的管理技术，提供了对文件进行打开与关闭、对记录读取和写入等存取方式。

#### 2. 文件系统的缺点

文件系统实现了记录内的结构性。但文件系统仍存在以下缺点。

(1) 数据共享性差，冗余度大。在文件系统中，一个(或一组)文件基本上对应于一个应用程序，即文件仍然是面向应用的。当不同的应用程序具有部分相同的数据时，也必须建立各自的文件，而不能共享相同的数据，因此数据的冗余度大，浪费存储空间。同时由于相同数据的重复存储、各自管理，容易造成数据的不一致性，给数据的修改和维护带来了困难。

(2) 数据独立性差。文件系统中的文件是为某一特定应用服务的，文件的逻辑结构是针对具体的应用来设计和优化的，因此要想对文件中的数据再增加一些新的应用会很困难。而且，当数据的逻辑结构改变时，应用程序中文件结构的定义必须修改，应用程序中对数

据的使用也要改变，因此数据依赖于应用程序，缺乏独立性。可见，文件系统仍然是一个不具有弹性的无整体结构的数据集合，即文件之间是孤立的，不能反映现实世界事物之间的内在联系。

在文件系统阶段，应用程序与数据之间的关系如图 1-3 所示。

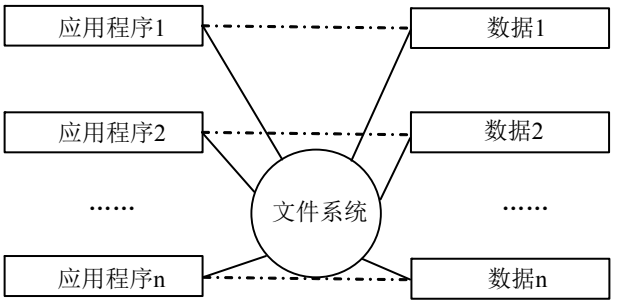


图 1-3　文件系统阶段应用程序与数据之间的关系

### 1.2.3　数据库系统阶段

20 世纪 60 年代后期以来，计算机管理的对象规模越来越大，应用范围越来越广，数据量急剧增长，同时多种应用、多种语言互相覆盖地共享数据集合的要求越来越强烈。

这时硬件已有大容量磁盘，硬件价格下降；软件则价格上升，为编制和维护系统软件及应用程序所需的成本相对增加；在处理方式上，联机实时处理要求更多，并开始提出和考虑分布处理。在这种背景下，以文件系统作为数据管理手段已经不能满足应用的需求，于是为解决多用户、多应用共享数据的需求，使数据为尽可能多的应用服务，数据库技术便应运而生，出现了统一管理数据的专门软件系统——数据库管理系统。

用数据库系统来管理数据比文件系统具有明显的优点，从文件系统到数据库系统标志着数据管理技术的飞跃。数据库系统阶段使用数据库技术来管理数据。数据库技术发展至今已经是一门非常成熟的技术，它克服了文件系统的不足，并增加了许多新功能。在这一阶段，数据由数据库管理系统统一控制，数据不再面向某个应用而是面向整个系统，因此数据可以被多个用户、多个应用共享，概括起来具有以下主要特征。

(1) 数据库能够根据不同的需要按不同的方法组织数据，最大限度地提高用户或应用程序访问数据的效率。

(2) 数据库不仅能够保存数据本身，还能保存数据之间的相互联系，保证了对数据修改的一致性。

(3) 在数据库中，相同的数据可以共享，从而降低了数据的冗余度。

(4) 数据具有较高的独立性，数据的组织和存储方法与应用程序相互独立，互不依赖，从而大大降低了应用程序的开发代价和维护代价。

(5) 提供了一整套的安全机制来保证数据的安全、可靠。

(6) 可以给数据库中的数据定义一些约束条件来保证数据的正确性(也称完整性)。

在数据库系统阶段，应用程序和数据库之间的关系如图 1-4 所示。

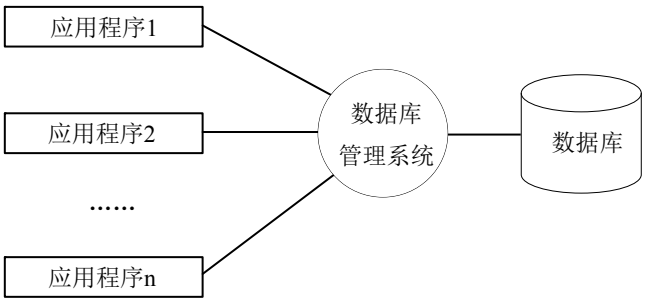


图 1-4　数据库系统阶段应用程序和数据库之间的关系

## 1.3　数 据 模 型

数据库技术是计算机领域中发展最快的技术之一。数据库技术的发展是沿着数据模型的主线推进的。模型，特别是具体模型，对人们来说并不陌生。一张地图、一组建筑设计沙盘、一架精致的航模飞机都是具体的模型，一眼望去就会使人联想到真实生活中的事物。

模型是对现实世界中某个对象特征的模拟和抽象。例如，航模飞机是对生活中飞机的一种模拟和抽象，它可以模拟飞机的起飞、飞行和降落，它抽象了飞机的基本特征——机头、机身、机翼、机尾。

数据模型(Data Model)也是一种模型，它是对现实世界数据特征的抽象。也就是说，数据模型是用来描述数据、组织数据和对数据进行操作的。

数据库是某个企业、组织或部门所涉及数据的综合，它不仅要反映数据本身的内容，而且要反映数据之间的联系。由于计算机不可能直接处理现实世界中的具体事物，所以人们必须事先把具体事物转换成计算机能够处理的数据。在数据库技术中，使用数据模型来抽象表示现实世界中的数据和信息。

现实世界中的数据要进入数据库中，需要经过人们的认识、理解、整理、规范和加工。可以把这一过程划分成 3 个主要阶段，即现实世界阶段、信息世界阶段和机器世界阶段。现实世界中的数据经过人们的认识和抽象，形成信息世界；信息世界中用概念模型来描述数据及其联系，概念模型按用户的观点对数据和信息进行建模，不依赖于具体的机器，独立于具体的数据库管理系统，是对现实世界的第一层抽象；根据所使用的具体机器和数据库管理系统，需要对概念模型进行进一步转换，形成在具体机器环境下可以实现的数据模型，称为逻辑模型。对现实世界数据的抽象过程，如图 1-5 所示。

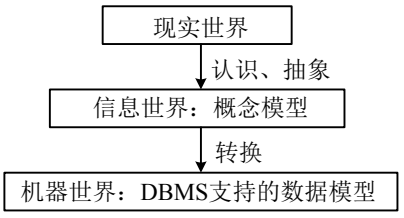


图 1-5　对现实世界数据的抽象过程

### 1.3.1　现实世界

在现实世界阶段，把现实世界中客观存在并可以相互区分的事物称为实体。实体可以是实际存在的东西，也可以是抽象的。例如，学生、课程、零件、仓库、项目、案件、选课等都是实体。

每一个实体都具有一定的特征。例如，对于“学生”实体，它具有学号、姓名、性别、生日等特征；对于“零件”实体，它具有名称、规格型号、生产日期、单价等特征。

具有相同特征的一类实体的集合构成实体集。例如，所有的学生构成“学生”实体集；所有的课程构成“课程”实体集；所有的部门构成“部门”实体集等。

在一个实体集中，用于区分实体的特征被称为标识特征。例如，对于学生实体，学号可以作为其标识特征，因为通过不同的学号可以区分不同的学生实体，而性别则不能作为其标识特征，因为通过性别“男”或“女”并不能识别出具体是哪个学生。

### 1.3.2　信息世界

人们对现实世界的对象进行抽象，并对其进行命名、分类，在信息世界用概念模型来对其进行描述。信息世界涉及的主要概念如下。

#### 1. 实体(Entity)

客观存在并可相互区别的事物被称为实体。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、学生的一次选课、部门的一次订货、教师与院系的工作关系(即某位教师在某院系工作)等都是实体。

#### 2. 属性(Attribute)

实体所具有的某一特性被称为属性。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，属性组合(201315121，张山，男，199505，计算机系，2013)即表征了一个学生。

#### 3. 域(Domain)

属性的取值范围被称为该属性的域。例如，姓名的域为字符串集合；年龄的域为不小于零的整数；性别的域为(男，女)。

#### 4. 码(Key)

唯一标识实体的属性集被称为码。例如，学号是学生实体的码。

#### 5. 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，被称为实体型。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。

### 6. 实体集(Entity Set)

同一类型实体的集合被称为实体集。例如，全体学生就是一个实体集。

### 7. 联系(Relationship)

现实世界中的事物之间通常是有联系的，这些联系在信息世界中反映为实体内部的联系和实体之间的联系。实体内部的联系通常指组成实体的各属性之间的联系；实体之间的联系通常指不同实体集之间的联系。这些联系总的来说可以划分为一对一联系、一对多(或多对一)联系以及多对多联系。

(1) 一对一联系。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(表示为 1∶1)。

例如，“班级”是一个实体集，“班长”也是一个实体集。如果按照语义，一个班级只能有一个班长，而一个班长只能管理某一个班级，则“班级”和“班长”实体集之间的联系就是一对一的联系。这种关系可以用图 1-6 来表示，这里把班级和班长之间的关系称为“管理”关系。

(2) 一对多联系。如果实体集 A 与实体集 B 之间存在联系，并且对于实体集 A 中的任意一个实体，在实体集 B 中可以有多个实体与之对应；而对于实体集 B 中的任意一个实体，在实体集 A 中至多只有一个实体与之对应，则称实体集 A 到实体集 B 的联系是一对多的联系(表示为 1∶n)。

例如，“部门”是一个实体集，“职工”也是一个实体集。如果按照语义，一个部门可以有多个职工，而一个职工只能归属于一个部门，则“部门”实体集和“职工”实体集的联系就是一对多的联系，如图 1-7 所示。

(3) 多对多联系。如果实体集 A 与实体集 B 之间存在联系，并且对于实体集 A 中的任意一个实体，在实体集 B 中可以有多个实体与之对应；而对于实体集 B 中的任意一个实体，在实体集 A 中也可以有多个实体与之对应，则称实体集 A 与实体集 B 的联系是多对多的联系(表示为 m∶n)。

例如，“学生”是一个实体集，“课程”也是一个实体集，“学生”实体集和“课程”实体集的联系就是多对多的联系。因为一个学生可以学习多门课程，而一门课程又可以有多个学生来学习，它们之间的关系如图 1-8 所示。这里把课程和学生之间的关系称为“选修”关系。

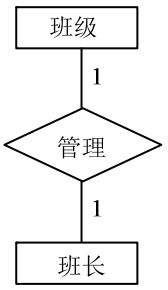

图 1-6　一对一联系

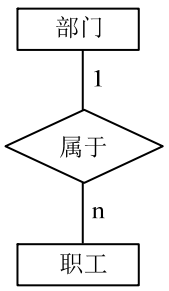

图 1-7　一对多联系

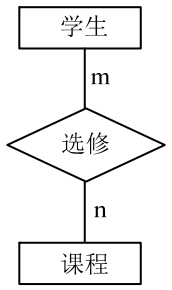

图 1-8　多对多联系

两个以上的实体之间也存在一对一、一对多和多对多的联系，这里不再介绍。

### 8. 概念模型

概念模型是对信息世界的建模，因此，概念模型能够方便、准确地表示出上述信息世界中的常用概念。概念模型有多种表示方法，其中，最常用的是“实体-联系方法”(Entity Relationship Approach)，简称 E-R 方法。E-R 方法用 E-R 图来描述现实世界的概念模型，E-R 图提供了表示实体、属性和联系的方法，具体如下。

(1) 实体型。用矩形表示，在矩形内写明实体名。如图 1-9 所示为学生实体和课程实体。

(2) 属性。用椭圆形表示，并用无向边将其与实体连接起来。例如，学生实体及其属性的 E-R 图表示如图 1-10 所示。

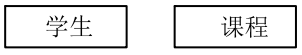


图 1-9　实体的表示

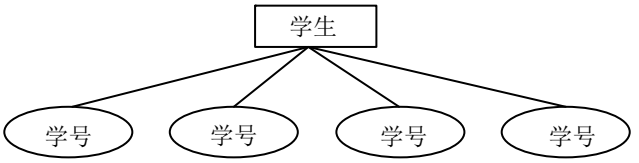


图 1-10　学生实体及其属性

(3) 联系。用菱形表示，在菱形框内写明联系的名称，并用无向边将其与有关的实体连接起来，同时在无向边旁标上联系的类型。例如，图 1-6、图 1-7 和图 1-8 分别表示了一对一、一对多和多对多的联系。需要注意的是，联系本身也是一种实体型，也可以有属性。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。例如，图 1-11 表示了学生实体和课程实体之间的联系，即“选修”联系，每个学生选修某一门课程会产生一个成绩，因此，“选修”联系有一个属性“成绩”，学生和课程实体之间是多对多的联系。

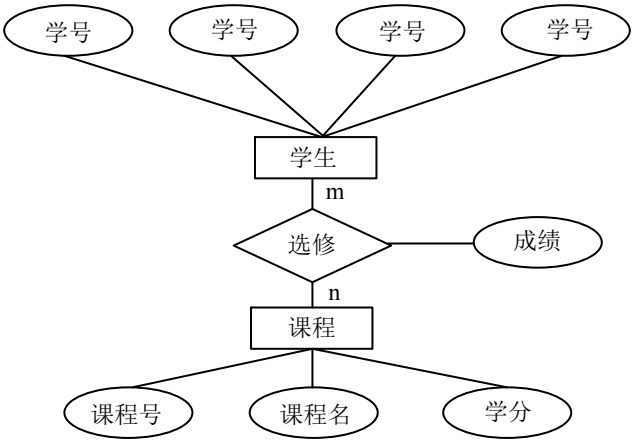


图 1-11　学生实体及课程实体之间的联系

用 E-R 图表示的概念模型独立于具体的 DBMS 所支持的数据模型，是各种数据模型的共同基础，因此比数据模型更一般、更抽象、更接近现实世界。

## 1.3.3　机器世界

当信息进入计算机后，则进入机器世界范畴。概念模型是独立于机器的，需要转换成

具体的 DBMS 所能识别的数据模型才能将数据和数据之间的联系保存到计算机上。在计算机中可以用不同的方法来表示数据与数据之间的联系，把表示数据与数据之间的联系的方法称为数据模型。数据库领域常见的数据模型有以下 4 种。

(1) 层次模型(Hierarchical Model)。

(2) 网状模型(Network Model)。

(3) 关系模型(Relational Model)。

(4) 面向对象模型(Object Oriented Model)。

其中，关系模型是目前使用最广泛的数据模型，其他数据模型在 20 世纪 70 年代至 80 年代初比较流行，现在已经逐步被关系模型的数据库系统所取代。因此，本书将只讨论关系模型。需要注意的是，以下介绍的关系模型是用户能够直接看到的数据模型，实际上关系模型是可以在某种 DBMS 的支持下用某种语言进行描述的，通过 DBMS 提供的功能实现对其进行存储和实施各种操作。把支持关系模型的数据库管理系统称为关系数据库管理系统，简称 RDBMS。

# 1.4　关系数据库

关系数据库，是创建在关系模型基础上的数据库，借助于集合代数等数学概念和方法来处理数据库中的数据。现实世界中的各种实体以及实体之间的各种联系均用关系模型来表示。关系模型是由埃德加 • 科德(E. F. Codd)于 1970 年首先提出的，并配合“科德十二定律”。现如今虽然对此模型有一些批评意见，但它还是数据存储的传统标准。标准数据查询语言 SQL 就是一种基于关系数据库的语言，这种语言执行对关系数据库中数据的检索和操作。

## 1.4.1　关系模型

关系模型是最重要的一种数据模型。关系数据库系统采用关系模型作为数据的组织方式。

1970 年，美国 IBM 公司 San Jose 研究室的研究员埃德加 • 科德首次提出了数据库系统的关系模型，开创了数据库关系方法和关系数据理论的研究，为数据库技术奠定了理论基础。由于埃德加 • 科德的杰出工作，他于 1981 年获得 ACM 图灵奖。

20 世纪 80 年代以来，计算机厂商新推出的数据库管理系统几乎都支持关系模型，非关系系统的产品也大都加上了关系接口。数据库领域当前的研究工作也都是以关系方法为基础。关系模型由关系数据结构、关系操作集合和关系的完整性约束 3 部分组成。

### 1. 关系数据结构

关系模型与以往的模型不同，它是建立在严格的数学概念的基础上的。这里只简单勾画关系模型。从用户观点看，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。下面以学生登记表(如表 1-1 所示)为例，介绍关系模型中的一些术语。

表 1-1　学生登记表

| 学号 | 姓名 | 性别 | 生日 | 入学成绩 | 专业 | 政治面貌 | 籍贯 | 民族 |
|---|---|---|---|---|---|---|---|---|
| 2015410101 | 刘　聪 | 男 | 1996-02-05 | 487 | 计算机科学与技术 | 党员 | 吉林 | 汉族 |
| 2015410102 | 王腾飞 | 男 | 1997-12-03 | 498 | 计算机科学与技术 | 团员 | 辽宁 | 回族 |
| 2015410103 | 张　丽 | 女 | 1996-03-09 | 482 | 计算机科学与技术 | 团员 | 黑龙江 | 朝鲜族 |
| 2015410201 | 李云霞 | 女 | 1996-06-15 | 456 | 软件工程 | 党员 | 河北 | 汉族 |
| 2015410202 | 马春雨 | 女 | 1997-12-11 | 487 | 软件工程 | 团员 | 吉林 | 汉族 |

(1) 关系(Relation)。一个关系对应于一张二维表，每个关系都有一个关系名。如表 1-1 所示的学生登记表可以取名为“学生信息”。

(2) 元组(Tuple)。表中的一行称为一个元组，对应于存储文件中的一个记录。

(3) 属性(Attribute)。表中的一列称为一个属性，给每个属性起一个名字，并称其为属性名。属性对应于存储文件中的字段。

(4) 候选码(Candidate Key)。如果在一个关系中，存在多个属性(或属性组合)都能用来唯一标识该关系的元组，这些属性(或属性组合)被称为该关系的候选码(或候选关键字)。例如，假设以上“学生登记”关系中的姓名没有重名现象，则学号和姓名都是候选码。

(5) 主码(Primary Key)。在一个关系的若干个候选码中指定作为码的属性(或属性组合)称为该关系的主码(或主关键字)。例如，可以将以上“学生信息”关系的学号作为该关系的主码。

(6) 主属性(Primary Attribute)。包含在候选码中的属性称为主属性。例如，学号和姓名(假设无重名)都是主属性。

(7) 非主属性(Nonprimary Attribute)。不包含在任何候选码中的属性称为非码属性或非主属性。例如，性别和生日都是非主属性。

(8) 关系模式(Relation Schema)。对关系的描述称为关系模式，一般表示如下：

关系名(属性 1，属性 2，…，属性 n)

例如，上面的关系模式可描述为：

学生登记(学号，姓名，性别，生日，入学成绩，专业，政治面貌，籍贯，民族)

(9) 全码(All-key)。如果一个关系模型的所有属性一起构成这个关系的码，则称其为全码。

(10) 域(Domain)。域是一组具有相同数据类型的值的集合。属性的取值范围来自某个域。如人的年龄一般为 1~120 岁，大学生年龄属性的域是(15~45 岁)，性别的域是(男，女)，系名的域是一个学校所有系名的集合。

(11) 分量(Component)。元组中的一个属性值称为分量，如表 1-1 中的“张丽”。

可以把关系和现实生活中的表格所对应的术语做一个粗略的对比，如表 1-2 所示。

表 1-2　术语对比

| 关 系 术 语 | 一般表格术语 |
| --- | --- |
| 关系名 | 表名 |
| 关系模式 | 表头(表格的描述) |
| 关系 | (一张)二维表 |
| 元组 | 记录或行 |
| 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 | 中有表(大表中嵌有小表) |

在关系模型中，实体和实体之间的联系都是用关系来表示的。例如，图 1-11 所表示的概念模型中的学生、课程和选修关系可以表示为以下 3 个关系模式：

- 学生信息(学号，姓名，性别，年龄)
- 课程(课程号，课程名，学分)
- 选修(学号，课程号，成绩)

### 2. 关系操作集合

关系操作主要包括查询、插入、修改和删除数据，这些操作的操作对象和操作结果都是关系，也就是元组的集合。

### 3. 关系的完整性约束

关系的完整性约束主要包括 3 类：实体完整性，参照完整性和用户定义的完整性。其中，实体完整性和参照完整性是关系模型必须满足的完整性约束条件，用户定义的完整性是指针对具体应用需要自行定义的约束条件。

(1) 实体完整性。一个基本关系通常对应于现实世界的一个实体集。例如，学生关系对应于学生的集合。现实世界中的实体是可区分的，即它们具有某种唯一性标识。相应地，关系模型中以主码作为唯一性标识。主码中的属性即主属性不能取空值。所谓空值就是“不知道”或“无意义”的值。如果主属性取空值，就说明存在某个不可标识的实体，即存在不可区分的实体，这与现实世界的应用环境相矛盾，因此这个实体一定不是一个完整的实体。这就是实体的完整性规则。

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

(2) 参照完整性。在关系模型中，实体及实体间的联系都是用关系来描述的，这就需要在关系与关系之间通过某些属性建立起它们之间的联系。

例如，对于以下 3 个关系模式：

- 学生信息(学号，姓名，性别，年龄)
- 课程(课程号，课程名，学分)
- 选修(学号，课程号，成绩)

“学生信息”关系的主码是学号，“课程”关系的主码是课程号，而“选修”关系的主码是(学号，课程号)，“选修”关系中的学号必须是一个在“学生信息”关系中存在的学号，而“选修”关系中的课程号也必须是一个在“课程”关系中存在的课程号。

参照完整性定义：设 F 基本关系 R 的一个或一组属性，但不是关系 R 的码，如果 F 与基本关系 S 的主码 Ks 相对应，则称 F 是基本关系 R 的外码(Foreign Key)，并称基本关系 R 为参照关系(Referencing Relation)，基本关系 S 为被参照关系(Referenced Relation)。关系 R 和 S 不一定是不同的关系。

选修关系的“学号”属性与学生关系的主码“学号”相对应；选修关系的“课程号”属性与课程关系的主码“课程号”相对应，因此“学号”和“课程号”属性是选修关系的外码。这里学生关系和课程关系均为被参照关系，选修关系为参照关系。

参照完整性规则就是定义外码与主码之间的引用规则。

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

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

按照参照完整性规则，“学号”和“课程号”属性也可以取两类值：空值或目标关系中已经存在的值。但由于“学号”和“课程号”是选修关系中的主属性，按照实体完整性规则，它们均不能取空值，所以选修关系中的“学号”和“课程号”属性实际上只能取相应被参照关系中已经存在的主码值。

(3) 用户定义的完整性。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，这是关系模型所要求的。除此之外，不同的关系数据库系统根据其应用环境的不同，往往还需要一些特殊的约束条件。用户定义的完整性就是针对某一具体关系数据库的约束条件，它反映某一具体应用所涉及的数据必须满足的语义要求。例如，某个属性必须取唯一值、某个非主属性不能取空值等。如在学生关系中，若按照应用的要求学生不能没有姓名，则可以定义学生姓名不能取空值;某个属性(如学生的年龄)的取值范围可以定义在 15～30 之间等。

#### 4. 关系的性质

上面提到，在用户观点下，关系模型中数据的逻辑结构是一张二维表，但并不是所有的二维表都是关系，关系数据库对关系是有一定限制的，归纳起来有以下几个方面。

(1) 表中的每一个数据项必须是单值的，每一个属性必须是不可再分的基本数据项。这是关系数据库对关系最基本的限制。例如，表 1-3 就是一个不满足该要求的表，因为工资不是最小的数据项，它还可以再分解为基本工资、薪级工资和津贴。

(2) 每一列中的数据项具有相同的数据类型，来自同一个域。

(3) 每一列的名称在一个表中是唯一的。

(4) 列次序可以是任意的。

(5) 表中的任意两行(即元组)不能相同。

(6) 行次序可以是任意的。

表 1-3　职工工资表

| 职工编号 | 姓　名 | 工　资 | | |
|---|---|---|---|---|
| | | 基本工资 | 薪级工资 | 津　贴 |
| 01 | 李元太 | 3000 | 2000 | 1000 |
| 02 | 张志民 | 2800 | 1800 | 900 |
| 03 | 王之之 | 3500 | 2300 | 1200 |

### 1.4.2　关系数据库的规范化理论

数据库设计的问题可以简单描述为：如果要把一组数据存储到数据库中，如何为这些数据设计一个合适的逻辑结构呢？如在关系数据库系统中，针对一个具体问题，应该构造几个关系？每个关系由哪些属性组成？使数据库系统无论是在数据存储方面，还是在数据操纵方面都有较好的性能，这就是关系数据库规范化理论要研究的主要问题。

E-R 模型的方法讨论了实体与实体之间的数据联系，而关系规范化理论主要讨论实体内部属性与属性之间的数据的联系，其目标是要设计一个“好”的关系数据库模型。

#### 1. 问题的提出

设有以下“学生”关系：

学生(学号，姓名，性别，所在系，系主任，课程号，课程名称，成绩)

该关系表示了学生选修各门课程的成绩信息，记录内容如表 1-4 所示。

表 1-4　“学生”关系

| 学　号 | 姓　名 | 性　别 | 所在系 | 系主任 | 课程号 | 课程名称 | 成　绩 |
|---|---|---|---|---|---|---|---|
| 2015410101 | 刘　聪 | 男 | 计科 | 王冬 | 101 | C 语言 | 78 |
| 2015410102 | 王腾飞 | 男 | 计科 | 王冬 | 102 | Java 语言 | 89 |
| 2015410103 | 张　丽 | 女 | 计科 | 王冬 | 103 | 操作系统 | 85 |
| 2015410201 | 李云霞 | 女 | 软件 | 刘迎 | 104 | 数据库 | 79 |
| 2015410201 | 李云霞 | 女 | 软件 | 刘迎 | 101 | C 语言 | 90 |
| 2015410201 | 李云霞 | 女 | 软件 | 刘迎 | 102 | Java 语言 | 78 |

可以看出，“学生”关系的主码应为(学号，课程号)。该关系存在以下问题。

(1) 数据冗余。所谓数据冗余，就是数据的重复出现。当一个学生选修多门课程时，学生信息重复出现，导致数据冗余的现象。例如，以上关于李云霞的信息就出现了 3 次；同样，一门课程有多个学生选修，也导致该课程信息的冗余。例如，C 语言课程被刘聪和李云霞选修了，因此，其课程名称出现了 2 次。显然，数据冗余会导致数据库存储性能的下降，同时还会带来数据的不一致性问题。

(2) 不一致性。由于存在数据冗余，因此，如果某个数据需要修改，则可能会因为其多处存在而导致在修改时未能全部修改过来而产生数据的不一致。例如，假设有 40 名学生选修了 C 语言这门课，则在关系表中就会有 40 条记录包含课程名称 C 语言，如果该课程

名称需要改成 C 语言程序设计，则可能会只修改了其中的一些记录，而其他记录没有修改。这就是数据的不一致，也叫更新异常。

(3) 插入异常。如果新生刚刚入校，还没有选修课程，则学生信息就无法插入表中，因为课程号为空，而主码为(学号，课程号)，根据关系模型的实体完整性规则，主码不能为空或部分为空，因此无法插入新生数据，这就是插入异常。又如，学校计划下学期开一门新课计算机组成原理，该课程信息也不能马上添加到表中，因为还没有学生选修该课程，无法知道学生的信息。简单地说，插入异常就是该插入的数据不能正常插入。

(4) 删除异常。当学生毕业时，需要删除相关的学生记录，于是就会删除对应的课程号、课程名信息。这就是删除异常。例如，在学生关系表中要删除学生记录(2015410103，张丽，女，计科，王冬，103，操作系统，85)，则会丢失课程号为 103、课程名为操作系统的课程信息。简单来说，删除异常就是不该删除的数据被异常地删除了。一个系的所有学生都毕业了，则这个系的系主任信息也将被删除。

为了克服以上问题，可以将学生关系分解为如下 4 个关系。

- 学生基本信息(学号，姓名，性别，所在系)　　主码为学号
- 系信息(所在系，系主任)　　主码为所在系
- 课程(课程号，课程名称)　　主码为课程号
- 选修(学号，课程号，成绩)　　主码为(学号，课程号)

首先，这样分解后的关系在一定程度上解决了数据冗余。例如，一门课程被 100 个学生选修，则该课程名称在课程关系中只会出现一次(在选修关系中只需要存储这 100 名学生的学号和该课程的课程号及成绩信息，但课程名称不会重复出现)。数据的不一致性是由于数据冗余产生的，解决了数据的冗余问题，不一致性问题就自然解决了。

其次，由于学生基本信息和课程信息是分开存储的，如果新生刚刚入校，也可以将新生信息插入学生基本信息关系中，只是在选修关系中没有该学生的相应成绩记录，因此不存在插入异常问题。

同样，当学生毕业时，要删除相关的学生信息，则只需要删除学生基本信息关系中的相关记录和选修关系中的相关成绩记录，不会删除课程信息，因此解决了删除异常的问题。为什么对学生关系进行以上分解之后，可以消除所有异常呢?这是因为学生关系中的某些属性之间存在数据依赖，这种数据依赖会造成数据冗余、插入异常、删除异常等问题。数据依赖是对属性间数据的相互关系的描述。

### 2. 函数依赖

函数依赖是数据依赖的一种描述形式。

**定义 1.1** 设 R(U)是属性集 U 上的关系模式，X、Y 是 U 的子集。若对于 R(U)的任意一个可能的关系 r，r 中不可能存在两个元组在 X 上的属性值相等，而在 Y 上的属性值不等，则称 X 函数确定 Y 或 Y 函数依赖于 X，记作 X→Y。

简单地说，如果属性 X 的值决定属性 Y 的值(如果知道 X 的值就可以获得 Y 的值)，则属性 Y 函数依赖于属性 X。

函数依赖和别的数据依赖一样是语义范畴的概念，只能根据语义来确定一个函数依赖。例如，“姓名→年龄”这个函数依赖只有在该部门没有同名人的条件下成立。如果允许有同名人，则年龄就不再函数依赖于姓名。

设计者也可以对现实世界作强制性规定，例如规定不允许同名人出现，因而使“姓名→年龄”函数依赖成立。这样当插入某个元组时这个元组上的属性值必须满足规定的函数依赖，若发现有同名人存在，则拒绝插入该元组。

注意，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 的一切关系均要满足的约束条件。

下面介绍一些术语和记号。

(1) X→Y，但 Y⊈X，则称 X→Y 是非平凡的函数依赖。

(2) X→Y，但 Y⊆X，则称 X→Y 是平凡的函数依赖。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。若不特别声明，总是讨论非平凡的函数依赖。

(3) 若 X→Y，则 X 称为这个函数依赖的决定属性组，也称为决定因素(Determinant)。

(4) 若 X→Y，Y→X，则记作 X←→Y。

(5) 若 Y 不函数依赖于 X，则记作 X↛Y。

例如，设有以下关系模式：

商品(商品名称，价格)

如果知道商品名称，就可以知道该商品的价格，也就是说，不存在商品名称相同而价格不同的记录，则可以说，价格函数依赖于商品名称，即商品名称→价格。

又如，设有以下关系模式：

学生(学号，姓名，性别，所在系，系主任，课程号，课程名称，成绩)

“学生”关系中有唯一的标识号学号，每个学生有且只有一个所在系，则学号决定所在系的值，因此，所在系函数依赖于学号，也就是学号→所在系。

同样可以看出学号与课程号共同决定一个成绩，因此成绩函数依赖于属性组(学号，课程号)，也就是(学号，课程号)→成绩。

同样存在非平凡函数依赖：(学号，课程号)→成绩。存在平凡函数依赖：(学号，课程号)→学号；(学号，课程号)→课程号。

**定义 1.2** 在 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：

$$X \xrightarrow{F} Y$$

若 X→Y，但 Y 不完全函数依赖于 X，则称 Y 对 X 部分函数依赖(partial functional dependency)，记作：

$$X \xrightarrow{P} Y$$

如表 1-4 关系所示，(学号，课号) $\xrightarrow{F}$成绩是完全函数依赖，(学号，课号) $\xrightarrow{P}$姓名是部分函数依赖，因为学号→姓名成立，而学号是(学号，课号)的真子集。

**定义 1.3** 在 R(U)中，如果 X→Y(Y⊈X)，Y↛X，Y→Z，Z⊈Y，则称 Z 对 X 传递函数依赖(transitive functional dependency)，记为 $X \xrightarrow{传递} Z$。

设有如下关系：

学生(学号，姓名，性别，所在系，系主任，课程号，课程名称，成绩)

则有学号→所在系，所在系→系主任成立，所以学号 $\xrightarrow{传递}$ 系主任。

这里加上条件 Y↛X，是因为如果 Y→X，则 X←→Y，实际上是 $X \xrightarrow{直接} Z$，是直接函数依赖而不是传递函数依赖。

### 3. 范式和规范化

规范化理论用于改造关系模式，通过分解关系模式来消除其中不合适的数据依赖，以解决数据冗余、插入异常、删除异常等问题。所谓规范化，就是用形式更为简洁、结构更加规范的关系模式取代原有关系的过程。要设计一个好的关系，必须使关系满足一定的约束条件，这种约束条件已经形成规范，分成几个等级，一级比一级要求更严格。满足最低一级要求的关系称为属于第一范式(Normal Form，NF)，在此基础上如果进一步满足某种约束条件，达到第二范式标准，则称该关系属于第二范式，以此类推，直到第五范式。显然，满足较高范式条件的关系必须满足较低范式的条件。一个较低的范式，可以通过关系的无损分解转换为若干个较高级范式的关系，这一过程被称为关系的规范化。

(1) 第一范式(1NF)。

**定义 1.4** 如果一个关系模式 R 的所有属性都是不可分的基本数据项，则 R 属于 lNF。记为：R∈1NF。

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

例如，表 1-4 的“学生”关系满足第一范式。

记为：

学生∈1NF

表 1-3 的职工工资表，其中工资分为 3 项：基本工资、薪级工资和津贴，所以不满足第一范式条件。

记为：

职工工资∉1NF

满足第一范式的关系模式不一定就是一个好的关系模式。例如，对于表 1-4 的“学生”关系，它存在数据冗余、插入异常、删除异常等问题。

(2) 第二范式(2NF)。

**定义 1.5** 若关系模式 R 是 lNF，并且每个非主属性都完全函数依赖于 R 的码，则 R 属于 2NF。记为：R∈2NF。

例如，有如下关系：

学生(学号，姓名，性别，所在系，系主任，课程号，课程名称，成绩)

已知该关系的码是(学号，课程号)，因此，学号、课程号是主属性，姓名、性别、所在系、系主任、课程名称、成绩是非主属性。该关系存在以下部分函数依赖：

(学号，课程号) $\xrightarrow{P}$姓名，(学号，课程号) $\xrightarrow{P}$性别，(学号，课程号) $\xrightarrow{P}$课程名称。

也就是存在非主属性对码的部分函数依赖，因此该关系不是 2NF。

记为：

学生$\notin$2NF

改进的方法是对该关系进行分解，生成若干关系，以消除部分函数依赖。实际上，这里就是把描述不同主题的内容分别用不同的关系来表示，形成以下 3 个关系：

- 学生基本信息(学号，姓名，性别，所在系，系主任)　　主码为学号
- 课程(课程号，课程名称)　　主码为课程号
- 选修(学号，课程号，成绩)　　主码为(学号，课程号)

可以看出，在这 3 个关系中不存在部分函数依赖，因此问题得到了解决。

但是学生基本信息表仍然存在删除异常、冗余等问题，如计科系增加一个学生，系主任的名字就出现一次，这是由于存在传递依赖造成的。

(3) 第三范式(3NF)。

**定义 1.6** 如果关系模式 R 是第二范式，且每个非主属性都不传递函数依赖于主码，则 R 属于 3NF。记为：R$\in$3NF。

也可以说，如果关系 R 的每一个非主属性既不部分函数依赖于主码，也不传递函数依赖于主码，则 R 属于 3NF。

如下关系：

学生基本信息(学号，姓名，性别，所在系，系主任)　　主码为学号

存在学号 $\xrightarrow{传递}$系主任，所以该关系不属于第三范式。

改进的方法是对该关系进行分解，生成若干关系，以消除传递依赖。实际上，这里就是把描述不同主题的内容分别用不同的关系来表示，形成以下 2 个关系：

- 学生信息(学号，姓名，性别，所在系)　　主码为学号
- 专业信息(系名称，系主任)　　主码为系主任

这里所在系和系名称是异名同义。

可以看出，分解后的关系解决了以上的插入异常、删除异常、数据冗余的问题。

在对关系进行规范化的过程中，一般要将一个关系分解为若干个关系。实际上，规范化的本质是把表示不同主题的信息分解到不同的关系中，如果某个关系包含两个或两个以上的主题，就应该将它分解为多个关系，使每个关系只包含一个主题。但是，在分解关系之后，关系数目增多，需要注意建立起关系之间的关联约束(参照完整性约束)。关系变得更加复杂，对关系的使用也会变得复杂，因此并不是分解得越细越好。一般来说，用户的

目标是第三范式(3NF)数据库，因为在大多数情况下，这是进行规范化功能与易用程度的最好平衡点。在理论上和一些实际使用的数据库中，有比 3NF 更高的等级，如 BCNF、4NF、5NF 等，但其对数据库设计的关心已经超过了对功能的关心，本书只讨论到 3NF。

# 1.5　数据库系统的体系结构

数据库系统的体系结构是数据库系统的一个总的框架。尽管实际的数据库系统软件产品多种多样，它们支持的数据模型也不一定相同，使用不同的数据库语言，建立在不同的操作系统之上，数据的存储结构也各不相同，但是绝大多数数据库系统在总的体系结构上都具有三级模式的结构特征。

## 1.5.1　数据库系统的三级模式结构

数据库系统的三级模式结构由外模式、模式和内模式组成，如图 1-12 所示。这三级模式是对数据的 3 个抽象级别，它把数据的具体组织留给 DBMS 管理，使用户能逻辑地、抽象地处理数据，而不必关心数据在计算机中的表示和存储。为了实现这 3 个抽象层次的联系和转换，数据库系统在这三级模式中提供了两层映像：外模式/模式映像，模式/内模式映像。

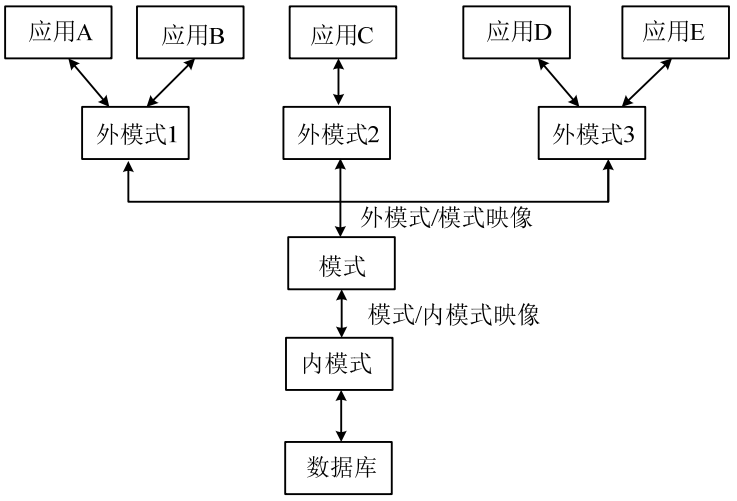


图 1-12　数据库系统的三级模式结构

### 1. 外模式

外模式也称子模式或用户模式，是数据库用户看到的数据视图，它是与某一应用有关的数据的逻辑表示。

外模式通常是模式的子集，它是各个用户的数据视图。由于不同的用户其需求不同，看待数据的方式不同，对数据的要求不同，使用的程序设计语言也可以不同，因此不同用

户的外模式描述是不同的。即使对模式中的同一数据，在外模式中的结构、类型、长度、保密级别等都可以不同。

数据库管理系统提供外模式数据描述语言(外模式 DDL)来描述外模式。

#### 2. 模式

模式也被称为逻辑模式，是数据库中全体数据的逻辑结构和特性的描述，是所有用户的公共数据视图。它是数据库系统模式结构的中间层，既不涉及数据的物理存储细节和硬件环境，也与具体的应用程序和开发工具无关。

模式实际上是数据库数据在逻辑级上的视图，一个数据库只有一个模式，数据库模式以某种数据模型为基础，综合考虑了所有用户的需求，并将这些需求有机地整合成一个逻辑整体。

模式不仅要定义数据的逻辑结构，而且要定义与数据有关的安全性、完整性要求；不仅要定义数据记录内部的结构，而且要定义这些数据项之间的联系，以及不同记录之间的联系。

数据库管理系统提供模式数据描述语言(模式 DDL)来描述模式。

#### 3. 内模式

内模式是全体数据库数据的内部表示或者低层描述，用来定义数据的存储方式和物理结构。

内模式通常用内模式数据描述语言(内模式 DDL)来描述和定义。

### 1.5.2　数据库的二级映像与数据的独立性

#### 1. 外模式/模式映像

对应于同一个模式，可以有任意多个外模式。外模式/模式的映像定义某一个外模式和模式之间的对应关系。当模式改变时，外模式/模式的映像要做相应的改变(由 DBA 负责)以保证外模式保持不变。

#### 2. 模式/内模式映像

模式/内模式映像定义数据的逻辑结构和存储结构之间的对应关系，它说明逻辑记录和字段在内部是如何表示的。这样，当数据库的存储结构改变时，可相应修改模式/内模式的映像，从而使模式保持不变。

正是由于上述这二级映像功能，才使得数据库系统中的数据具有较高的逻辑独立性和物理独立性。数据库这种多层次的结构体系可进一步阐述如下。

(1) 在定义一个数据库的各层次结构时，全局逻辑结构(模式)应该首先定义，因为它独立于数据库的其他所有结构描述。

(2) 内模式(存储模式)是依赖于全局逻辑结构的，其目的是具体地将数据库模式中所定义的全部数据及其联系进行适当的组织并加以存储，以实现较高的时空运行效率。存储模式独立于任何一个用户的局部逻辑结构描述(外模式)。

(3) 用户的外模式独立于存储模式和存储设备，它必须在数据库的全局逻辑结构描述(模式)的基础上定义。子模式一旦被定义，则除非模式结构的变化使得子模式中的某些数据无法再从数据库中导出，否则子模式将不必改变。通过调整外模式/模式映像可实现这一点。这就是子模式对于模式的相对独立性，即逻辑数据独立性。

(4) 应用程序是在子模式的数据结构上编制的，因此，它必然依赖于特定的子模式。但是，在一个完善的数据库系统中，它是独立于存储设备和存储模式的，并且只要数据库全局逻辑模式的变化不导致其对应的子模式的改变，则应用程序也是独立于数据库模式的。

# 1.6　数据库系统设计简介

数据库系统的设计包括数据库的设计和数据库应用系统的设计。数据库设计是指设计数据库的结构特性，即为特定的应用环境构造最优的数据模型；数据库应用系统的设计是指设计出满足各种用户对数据库应用需求的应用程序。用户通过应用程序来访问和操作数据库。

按照规范设计的方法，考虑到数据库及其应用系统开发的全过程，将数据库设计分为以下 6 个阶段：①需求分析阶段；②概念结构设计阶段；③逻辑结构设计阶段；④物理结构设计阶段；⑤数据库实施阶段；⑥数据库运行和维护阶段。

需要指出的是，以上设计步骤既是数据库设计的过程，也包括数据库应用系统的设计过程。在设计过程中只有将这两方面有机地结合起来，互相参照、互为补充，才可以设计出性能良好的数据库应用系统。

## 1.6.1　需求分析阶段

需求分析阶段是数据库设计的第一步，也是最困难、最耗时的一步。需求分析的任务是要准确了解并分析用户对系统的要求，确定所要开发的应用系统的目标，收集和分析用户对数据与处理的要求。需求分析主要是考虑做什么，而不是考虑怎么做。需求分析做得是否充分、准确，将决定以后各设计步骤能否顺利进行。如果需求分析做得不好，会影响整个系统的性能，甚至会导致整个数据库设计的返工。

需求分析阶段需要重点调查的是用户的信息要求、处理要求、安全性与完整性要求。信息要求是指用户需要从数据库中获得信息的内容与性质，由用户的信息要求可以导出数据要求，即在数据库中需要存储哪些数据；处理要求包括对处理功能的要求、对处理的响应时间的要求、对处理方式(如批处理、联机处理)的要求等。

需求分析的结果是产生用户和设计者都能接受的需求说明书，作为下一步数据库概念结构设计的基础。

## 1.6.2　概念结构设计阶段

需求分析阶段描述的用户需求是面向现实世界的具体需求。将需求分析得到的用户需求抽象为信息结构即概念模型的过程就是概念结构设计。概念结构独立于支持数据库的

DBMS 和使用的硬件环境。

人们提出了多种概念结构设计的表达工具，其中，最常用、最有名的是 E-R 模型。

在概念结构设计阶段，首先要对需求分析阶段收集到的数据进行分类、组织，形成实体、实体的属性，标识实体的码，确定实体之间的联系类型(1∶1，1∶n，m∶n)，针对各个局部应用设计局部视图(如分 E-R 图)。各个局部视图建立好后，还需要对它们进行合并，通过消除各局部视图的属性冲突、命名冲突、结构冲突、数据冗余等，最终集成为一个全局视图(如整体的 E-R 图)。

概念结构具有丰富的语义表达能力，能表达用户的各种需求，反映现实世界中各种数据及其复杂的联系，以及用户对数据的处理要求等。由于概念结构独立于具体的 DBMS，因此易于理解，用它可以和不熟悉计算机的用户交换意见。

设计概念模型的最终目的是向某种 DBMS 支持的数据模型转换，因此，概念模型是数据库逻辑设计的依据，是整个数据库设计的关键。

### 1.6.3　逻辑结构设计阶段

逻辑结构设计的任务是将概念结构进一步转化为某一 DBMS 支持的数据模型，包括数据库模式和外模式。

在逻辑结构设计阶段，首先需要将概念结构转化为一般的关系、网状、层次模型。其次，将转化后的关系、网状、层次模型向特定 DBMS 支持下的数据模型转换，转换的主要依据是所选用的 DBMS 的功能及限制，没有通用规则。对于关系模型来说，这种转换通常都比较简单。最后，对数据模型进行优化。

对于 E-R 图向关系模型的转换，需要解决的问题是如何将实体、实体的属性和实体之间的联系转化为关系模型。

得到初步数据模型后，还应该适当地修改、调整数据模型的结构，以进一步提高数据库应用系统的性能。关系数据模型的优化通常以规范化理论为指导。

这一阶段可能还需要设计用户子模式(外模式)，即用户可直接访问的数据模式。前面已经提到，在同一系统中，不同用户可以有不同的外模式。外模式来自逻辑模式，但在结构和形式上可以不同于逻辑模式，所以它不是逻辑模式简单的子集。外模式的作用主要有：通过外模式对逻辑模式的屏蔽，为应用程序提供了一定的逻辑独立性；可以更好地适应不同用户对数据的需求；为用户划定了访问数据的范围，有利于数据的安全保密；等等。

定义用户外模式时应该更注重考虑用户的习惯与方便，主要包括以下 3 个方面。

(1) 使用更符合用户习惯的别名。

(2) 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。

(3) 如果某些局部应用中经常要使用很复杂的查询，为了方便用户，可以将这些复杂查询定义为外模式(视图)，以简化用户对系统的使用。

### 1.6.4　物理结构设计阶段

数据库的物理结构设计阶段用于为逻辑数据模型选取一个最适合应用环境的物理结

构，包括数据库在物理设备上的存储结构和存取方法。由于不同的数据库产品所提供的物理环境、存取方法和存储结构各不相同，供设计人员使用的设计变量、参数范围也各不相同，所以数据库的物理结构设计没有通用的设计方法可以遵循。

数据库设计人员都希望自己设计的物理数据库结构能满足事务在数据库上运行时响应时间短、存储空间利用率高和事务吞吐率大的要求。为此，设计人员需要对要运行的事务进行详细的分析，获得物理数据库设计所需要的参数，并且全面了解给定的 DBMS 的功能、所提供的物理环境和工具，尤其是存储结构和存取方法。在确定数据存取方法时，必须清楚以下 3 种相关信息。

(1) 数据库查询事务的信息，包括查询所需要的关系、查询条件所涉及的属性、查询连接条件所涉及的属性、查询结果所涉及的属性等。

(2) 数据更新事务的信息，包括被更新的关系、每个关系上的更新操作所涉及的属性、修改操作要改变的属性值等。

(3) 每个事务在各关系上运行的频率和性能要求。

关系数据库物理结构设计的内容主要有：为关系模式选择存取方法和存储结构，包括设计关系、索引等数据库文件的物理存储结构，确定系统配置参数等。

在初步完成物理结构的设计之后，还需要对物理结构进行评价，评价的重点是时间和空间效率。如果评价结果满足原设计要求，则可以进入物理实施阶段，否则，需要重新设计或修改物理结构，有时甚至要返回到逻辑设计阶段，修改数据模型。

### 1.6.5　数据库实施阶段

完成数据库物理结构设计之后，设计人员就要用 DBMS 提供的数据定义语言和其他实用程序将数据库逻辑设计和物理设计结果严格地描述出来，成为 DBMS 可以接受的源代码，再经过调试产生目标模式，就可以组织数据入库了，这便是数据库实施阶段，具体包括以下内容。

(1) 用所选用的 DBMS 提供的数据定义语言来严格描述数据库结构。

(2) 组织数据入库。数据库结构建立好后，即可向数据库中装载数据。组织数据入库是数据库实施阶段最主要的工作。对于小型系统，可以选择使用人工方法装载数据。对于中、大型系统，可以使用计算机辅助数据入库，如使用数据录入子系统提供录入界面，对数据进行检验、转换、综合、存储等。

需要装入数据库中的数据通常分散在各个部门的数据文件或原始凭证中，所以首先必须把需要入库的数据筛选出来。对于筛选出来的数据，其格式往往不符合数据库要求，还需要进行一定的转换，这种转换有时可能很复杂。最后才可以将转换好的数据输入计算机中。

(3) 编制与调试应用程序。数据库应用程序的设计应该与数据库设计并行进行。因此，在部分数据录入数据库中之后，就可以对应用程序进行调试了。调试应用程序时由于数据入库尚未完成，可以先使用模拟数据，模拟数据应该具有一定的代表性，足够测试系统的多数功能。应用程序的设计、编码和调试方法、步骤应遵循软件工程的规范。

(4) 数据库试运行。应用程序调试完成，并且已有一小部分数据入库后，即可开始数

据库的试运行。数据库试运行也称为联合调试，其主要工作如下。

- 功能测试：实际运行应用程序，执行对数据库的各种操作，测试应用程序的各种功能是否满足设计要求。
- 性能测试：测试系统的性能指标，分析其是否达到设计目标。如果结果不符合设计目标，则需要返回物理设计阶段，调整物理结构，修改参数。有时甚至需要返回逻辑设计阶段，调整逻辑结构。

需要注意的是，组织数据入库的工作量非常大，如果在数据库试运行后还要修改数据库设计，则可能需要重新组织数据入库。所以可以采用分期输入数据的方法，先输入小批量数据供前期的联合调试使用，待试运行基本合格后再输入大批量数据，逐步增加数据量，完成运行评价。

在数据库试运行阶段，系统还不稳定，硬件和软件的故障随时都可能发生。系统的操作人员对新系统还不熟悉，不可避免地会发生一些误操作。因此，必须首先做好数据库的转储和恢复工作，一旦发生故障，能使数据库尽快恢复，以减少对数据库的破坏。

### 1.6.6　数据库运行和维护阶段

数据库试运行结果符合设计目标后，数据库即可投入正式运行。数据库投入运行，标志着开发任务的基本完成和维护工作的开始。由于应用环境在不断变化，在数据库运行过程中，物理存储会不断变化，因此，对数据库设计进行评价、调整、修改等维护工作是一个长期的任务，也是设计工作的继续和提高。

在数据库运行阶段，对数据库经常性的维护工作主要是由 DBA 完成的。这一阶段的工作主要包括以下几点。

(1) 数据库的转储和恢复。转储和恢复是系统正式运行后最重要的维护工作之一。DBA 要针对不同的应用要求制订不同的转储计划，定期对数据库和有关文件进行备份。一旦系统发生故障，可以尽快对数据库进行恢复。

(2) 数据库的安全性、完整性控制。在数据库运行过程中，由于应用环境的变化，对安全性的要求也会发生变化，DBA 需要根据实际情况的变化修改原有的安全性控制，根据用户的实际需要授予不同的操作权限。由于应用环境的变化，数据库的完整性约束条件也会变化，也需要 DBA 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。在数据库运行过程中，DBA 必须监督系统运行，对监测数据进行分析，找出改进系统性能的方法。有些 DBMS 提供检测系统性能工具，可以利用该工具获取系统运行过程中一系列性能参数的值。通过仔细分析这些数据，判断当前系统是否处于最佳运行状态。如果不是，则需要通过调整某些参数来进一步改进数据库性能。

(4) 数据库的重组织和重构造。数据库运行一段时间后，由于记录的不断增、删、改，会使数据库的物理存储变坏，从而降低数据库存储空间的利用率和数据的存取效率，使数据库的性能下降。因此，需要对数据库进行重新组织(全部重组织或部分重组织)，以提高系统的性能。

# 习　题　1

## 一、单项选择题

1. (　　)是长期存储在计算机内的相互关联的数据的集合。

A. 数据库管理系统　　B. 数据库系统　　C. 数据库　　D. 文件

2. (　　)是位于用户与操作系统之间的一层数据管理软件。

A. 数据库管理系统　　B. 数据库系统

C. 数据库　　D. 数据库应用系统

3. 数据库管理系统能实现对数据库数据的添加、修改、删除等操作，这种功能被称为(　　)。

A. 数据定义功能　　B. 数据管理功能

C. 数据操纵功能　　D. 数据控制功能

4. 数据库管理系统(DBMS)是一种(　　)。

A. 数学软件　　B. 应用软件　　C. 操作系统　　D. 系统软件

5. 数据库系统不仅包括数据库本身，还要包括相应的硬件、软件和(　　)。

A. 数据库管理系统　　B. 数据库应用系统

C. 相关的计算机系统　　D. 各类相关人员

6. 数据库的建立、使用和维护只靠 DBMS 是不够的，还需要有专门的人员来完成，这些人员称为(　　)。

A. 高级用户　　B. 数据库管理员

C. 数据库用户　　D. 数据库设计员

7. 数据库(DB)、数据库系统(DBS)和数据库管理系统(DBMS)三者之间的关系是(　　)。

A. DBS 包括 DB 和 DBMS　　B. DBMS 包括 DB 和 DBS

C. DB 包括 DBS 和 DBMS　　D. DBS 就是 DB，也就是 DBMS

8. 在人工管理阶段，数据是(　　)。

A. 有结构的　　B. 无结构的

C. 整体无结构、记录内有结构的　　D. 整体结构化的

9. 在文件系统阶段，数据(　　)。

A. 无独立性　　B. 独立性差

C. 具有物理独立性　　D. 具有逻辑独立性

10. 产生数据不一致的根本原因是(　　)。

A. 数据存储量太大　　B. 没有严格地保护数据

C. 未对数据进行完整性控制　　D. 数据冗余

11. 在数据库中存储的是(　　)。

A. 数据　　B. 数据模型

C. 数据以及数据之间的联系　　D. 信息

12. 数据库不仅能够保存数据本身，还能保存数据之间的相互联系，保证了对数据修改的(　　)。

A. 一致性　　B. 独立性　　C. 安全性　　D. 共享性

13. 在数据库系统阶段，数据(　　)。

A. 没有独立性　　B. 具有一定的独立性

C. 具有高度独立性　　D. 独立性差

14. 数据库系统和文件系统的主要区别是(　　)。

A. 数据库系统复杂，而文件系统简单

B. 文件系统不能解决数据冗余和数据独立性问题，而数据库系统能够解决

C. 文件系统只能管理文件，而数据库系统还能管理其他类型的数据

D. 文件系统只能用于小型、微型机，而数据库系统还能用于大型机

15. 在数据管理技术的发展过程中，数据独立性最高的是(　　)阶段。

A. 数据库系统　　B. 文件系统　　C. 人工管理　　D. 数据项管理

16. 在用户观点下，关系模型中数据的逻辑结构是(　　)。

A. 一个 E-R 图　　B. 一张二维表　　C. 层次结构　　D. 网状结构

17. 在一个关系中如果有这样一个属性存在，它的值能唯一地标识关系中的每一个元组，这个属性被称为(　　)。

A. 候选码　　B. 数据项　　C. 主属性　　D. 主属性值

18. 关系模型结构单一，现实世界中的实体以及实体之间的各种联系均以(　　)的形式来表示。

A. 数据项　　B. 属性　　C. 分量　　D. 域

19. 以下关于关系的说法错误的是(　　)。

A. 一个关系中的列次序可以是任意的

B. 一个关系的每一列中的数据项可以有不同的数据类型

C. 关系中的任意两行(即元组)不能相同

D. 关系中行的次序可以是任意的

20. 关系规范化中的删除操作异常是指(　　)，插入操作异常是指(　　)。

A. 不该删除的数据被删除　　B. 不该插入的数据被插入

C. 应该删除的数据未被删除　　D. 应该插入的数据未被捕入

21. 关系数据库规范化是为解决关系数据库中的(　　)问题而引入的。

A. 插入、删除异常和数据冗余　　B. 查询速度

C. 数据操作的复杂性　　D. 数据的安全性和完整性

22. 数据依赖讨论的问题是(　　)。

A. 关系之间的数据关系　　B. 元组之间的数据关系

C. 属性之间的数据关系　　D. 函数之间的数据关系

23. 函数依赖是(　　)。

A. 对函数关系的描述　　B. 对元组之间关系的一种描述

C. 对数据库之间关系的一种描述　　D. 对数据依赖的一种描述

24. 规范化理论是关系数据库进行逻辑设计的理论依据。根据这个理论，关系数据库中的关系必须满足：每一个属性都是(　　)。

A. 不相关的　　B. 不可分解的　　C. 长度可变的　　D. 有关联的

25. 消除了非主属性对码的部分函数依赖的 lNF 的关系模式必定是(　　)。

A. 1NF　　B. 2NF　　C. 3NF　　D. 4NF

26. 2NF(　　)规范为 3NF。

A. 消除非主属性对码的部分函数依赖　　B. 消除非主属性对码的传递函数依赖

C. 消除主属性对码的部分函数依赖　　D. 消除主属性对码的传递函数依赖

27. 数据库的三级模式结构中，描述数据库中全体数据的全局逻辑结构和特征的是(　　)。

A. 外模式　　B. 内模式　　C. 存储模式　　D. 模式

28. 子模式是(　　)。

A. 模式的副本　　B. 模式的逻辑子集　　C. 多个模式的集合　　D. 存储模式

29. 数据库系统的数据独立性是指(　　)。

A. 不会因为数据的变化而影响应用程序

B. 不会因为系统数据存储结构与数据逻辑结构的变化而影响应用程序

C. 不会因为存储策略的变化而影响存储结构

D. 不会因为某些存储结构的变化而影响其他的存储结构

## 二、填空题

1. 对现实世界进行第一层抽象的模型，称为_____模型；对现实世界进行第二层抽象的模型，称为_____模型。

2. 在信息世界中，用_____来表示实体的特征。

3. _____是用来唯一标识实体的属性。

4. 实体之间的联系可以有_____、_____和_____ 3 种。

5. 如果在一个关系中，存在多个属性(或属性组合)都能用来唯一标识该关系的元组，这些属性(或属性组合)都称为该关系的_____。

6. 包含在_____中的属性称为主属性。

7. 关系模式一般表示为_____。

8. 关系模型由_____、_____和_____3 部分组成。

9. 关系模型允许定义的 3 类完整性约束是_____完整性、_____完整性和_____完整性。其中，_____完整性和_____完整性是关系模型必须满足的完整性约束条件。

10. 实体完整性要求主码中的主属性不能为_____。

11. 数据库设计过程的 6 个阶段是指_____、_____、_____、_____、_____、_____。

## 三、写出以下各缩写的英文含义和中文含义

1. DB：_____。
2. DBMS：_____。
3. RDBMS：_____。
4. DBS：_____。
5. DBA：_____。
6. NF：_____。
7. DDL：_____。

## 四、按题目要求回答问题

1. 设某商业集团数据库中有 3 个实体集：一是公司实体集，属性有公司编号、公司名、地址；二是仓库实体集，属性有仓库编号、仓库名、地址；三是职工实体集，属性有职工编号、姓名、性别。

设：公司与仓库之间存在隶属联系，每个公司管辖若干仓库，每个仓库只能属于一个公司管辖；公司与职工之间存在聘用联系，每个公司可聘用多个职工，每个职工只能在一个公司工作，公司聘用职工有聘期和工资。

试画出 E-R 图，并在图上注明属性、联系的类型。

2. 某体育运动锦标赛由来自世界各国运动员组成的体育代表团参赛各类比赛项目。假设如下。

- 对于每个代表团，包含的信息有：团编号，地区，住所。
- 对于每个运动员，包含的信息有：编号，姓名，年龄，性别。
- 对于每个比赛项目，包含的信息有：项目编号，项目名称，级别。
- 对于每一个比赛类别，包含的信息有：类别编号，类别名称，主管。

每个代表团有多个运动员，而每个运动员只属于一个代表团；一个运动员可以参加多个比赛项目，每个比赛项目有多个运动员参加；一种比赛类别中包含多个比赛项目，一个比赛项目只属于一种比赛类别。每个运动员参加某个比赛项目具有比赛时间和得分信息。

试为该锦标赛各个代表团、运动员、比赛项目、比赛类别设计 E-R 图，并在图上注明属性、联系的类型。

3. 设有如表 1-5 所示的关系 R。

表 1-5　R 关系

| 课 程 编 号 | 任 课 教 师 | 教 师 电 话 |
|---|---|---|
| 101 | 李云霞 | 131******** |
| 102 | 王腾飞 | 138******** |
| 103 | 张　丽 | 135******** |
| 104 | 李云霞 | 131******** |

(1) 关系 R 为第几范式？为什么？

(2) 关系 R 是否存在删除操作异常？若存在，说明是在什么情况下发生的。

(3) 将关系 R 分解为高一级范式，分析分解后的关系是如何解决分解前可能存在的删除操作异常的。