第 1 章

数据库原理概述

数据库原理及技术课程设计的良好开展，取决于对数据库系统原理知识的掌握。本章将简要地介绍这些必须要较好掌握的原理性、概念性知识，为即将展开的课程设计做好知识储备。

1.1 数据库与数据库系统

1. 数据库

数据库(DataBase，DB)系统的处理对象是信息或表示信息的数据，信息(information)与数据(data)可以这样简单定义。

(1) 信息：向人们(或机器)提供关于现实世界新的事实的知识。

(2) 数据：用以载荷信息的可标识的物理符号，如：数字、符号、语言文字、声音、图形、图像、视频以及事物间发生的各种关联描述等。数据是数据库中存储的基本对象，不同形式的数据都可以以二进制形式转化到计算机数据库中的。

数据库从字面意思来说就是存放数据的仓库。具体而言就是长期存放在计算机内的有组织的可共享的数据集合，数据库中的数据按一定的数据模型描述、组织和储存，具有尽可能小的冗余度和较高的数据独立性和易扩展性。

在表面上，人们认为数据库内含有单位、企业或组织的形形色色、多种多样的直观信息，如文本、表格、图形、图像、声音信息等；在逻辑上，人们同样认为数据库中含有众多单位、企业或组织所需要的复杂而多样的各种逻辑数据；在实际使用中，数据往往以列表、记录或单个数据项的显示形式出现；在物理上，数据库实际上是由存放在一个或多个磁盘上的若干个物理文件组成的，其复杂的物理数据组织方式往往是透明、不公开的。

2. 数据库系统

数据库系统(DataBase System，DBS)是指在计算机系统中引入数据库及其管理系统后的系统，其构成主要有数据库(及相关硬件)、操作系统、数据库管理系统及其开发工具、应用系统、数据库管理员、系统分析设计人员和各个级别的普通用户这几部分。其中在数据库的建立、使用和维护的过程要有专门的人员来完成，这些人就被称为数据库管理员(DataBase Administrator，DBA)。

数据库系统及其组成可以用图 1-1 和图 1-2 来表示与说明。

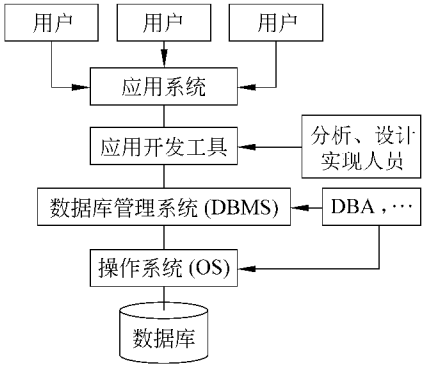

图 1-1　数据库系统及其组成

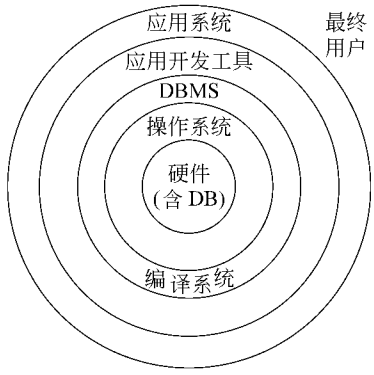

图 1-2　数据库在计算机系统中的地位

其中数据库管理系统(DataBase Management System,DBMS)是数据库系统的核心组成部分,是对数据进行管理的大型系统软件,用户对数据库的操作都是由数据库管理系统来控制执行而完成的。不同的数据库管理系统一般都包括以下几个功能。

(1) 数据定义：DBMS 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(包括表、视图、索引、存储过程、数据结构和有关的约束规则等)进行定义。

(2) 数据操纵：DBMS 提供数据操纵语言(Data Manipulation Language,DML),通过 DML 操纵数据实现对数据库的一些基本操作,如查询、添加、删除和修改等。其中,国际标准数据库操作语言——SQL 语言,就是 DML 的一种。

(3) 数据库的运行管理：这一功能是数据库管理系统的核心所在。DBMS 通过对数据库在建立、运用和维护时提供统一管理和控制,以保证数据安全、正确、有效地正常运行。DBMS 运行管理主要通过数据库安全性控制、完整性控制、并发性控制和系统备份与恢复等方面来体现的。

(4) 数据库的建立和维护等辅助功能：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、重组织功能和性能监视、分析功能等。

当前流行着的中、大型数据库系统有 Oracle、Informix、Sybase、INGRES、DB2(IBM)、MS SQL Server、Interbase(INPRISE,原 BROLAND)、MySQL、PostgreSQL 等,微型或小型数据库系统有 Dbase、Foxbase、Foxpro、VFP 系列、Access 等。这些系统实际上就是指管理相应数据库的数据库管理系统软件。学习与掌握这些数据库管理系统软件主要是以掌握以上四方面相似性的功能来要求的。

很显然,数据库应用系统的开发与实现,至少要求对某一数据库管理系统软件及某一应用系统开发工具要有全面而深入的了解。

1.2　数 据 模 型

数据库应用系统面对的是现实世界的现实问题,要利用运行于计算机上的数据库应用系统软件来对应管理或解决现实世界的问题,显然要借助多种类型的模型并经历若干次认

识抽象或变换过程才能实现。图 1-3 所示的三个世界与两种模型的关系图说明了这种状况。数据库应用系统的设计、开发与实现过程必然要经历或涉及三个世界与两种模型，数据库应用系统作为机器（或计算机）世界中的应用软件，它主要针对的是某 DBMS 支持的借助数据模型组织起来的数据库。为此，掌握好数据模型的相关知识对应用系统的实现是至关重要的。

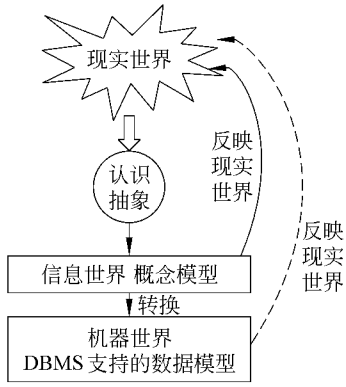

图 1-3　三个世界与两种模型

模型这个概念，人们并不陌生，它是对现实世界特征的模拟和抽象。数据模型也是一种模型，它能实现对现实世界信息或数据特征的抽象。现有的数据库系统均是基于某种数据模型的。因此，了解数据模型的基本概念是学习数据库的基础。

数据模型可分成两个不同的层次：

（1）概念模型，也称信息模型，它是按用户的观点来对数据和信息建模。

（2）数据模型，主要包括层次模型、网状模型、关系模型、面向对象模型等，它是按计算机系统的观点对数据建模的。

为了把现实世界中的具体事物抽象、组织为某一 DBMS 支持的数据模型，人们常常首先将现实世界问题抽象到信息世界，再从信息世界转换（或数据化）到机器世界。也就是说，首先把现实世界中的客观对象抽象为不依赖于具体的计算机系统的某一种信息结构，这种信息结构也不是某个具体 DBMS 支持的数据模型，而是概念级的模型；然后再把概念模型转换为计算机上某一 DBMS 支持的数据模型。而无论是概念模型还是数据模型，反过来都要能较好地刻画与反映现实世界，并与现实世界时时保持一致。这一过程如图 1-3 所示。三个世界中术语对照见表 1-1。

表 1-1　现实世界、信息世界、机器世界/关系数据库间术语对照表

现实世界	信息世界	机器世界/关系数据库
	抽象 ⟷ 数据化	
事物	实体	记录/元组（或行）
若干同类事物	实体集	记录集（即文件）/元组集（即关系）
若干特征刻画的事物	实体型	记录型/二维表框架（即关系模式）
事物的特征	属性	字段（或数据项）/属性（或列）
事物之间的关联	实体型（或实体）之间的联系	记录型之间的联系/联系表（外码）
事物某特征的所有可能值	域	字段类型/域
事物某特征的一个具体值	一个属性值	字段值/分量
可区分同类事物的特征或若干特征	码	关键字段/关系键（或主码）

1.2.1　概念模型

信息世界是现实世界在人们头脑中的反映，概念模型是现实世界到机器世界的一个中间层次。概念模型针对于抽象的信息世界，该模型要反映信息世界中的一些基本概念，如实体(反映现实世界的事物)、属性(反映事物的特性)、码、域、实体型、实体集、联系(反映事物间的关联)等。

概念模型的表示方法很多，最常用又最简单的表示方法是实体-联系方法。该方法用E-R图来描述信息世界中的概念，E-R图主要提供了表示实体型、属性和联系及它们之间关系的能力。下面是E-R图的具体表示方法。

(1) 实体型：用矩形表示，矩形框内写明实体名。

(2) 属性：用椭圆表示，椭圆形内写明属性名，并用无向边将其与相应的实体或联系连接起来。

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

注意：

联系本身也可看成是一种实体型，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相连接。

1.2.2　数据模型

数据模型是机器世界中的模型，是数据库系统的核心和基础。各种机器上实现的DBMS软件都是基于某种数据模型的。数据模型是严格定义的一组概念的集合，这些概念精确地描述了系统的静态特性、动态特性和完整性约束。因此数据模型通常由分别反映系统这三方面的数据结构、数据操作和数据完整性约束三部分组成。

1) 数据结构

数据结构用于描述系统的静态特性，是所研究的对象类型的集合。其所研究的对象是数据库的组成部分，它们包括两类，一类是与数据类型、内容、性质有关的对象，例如网状模型中的数据项、记录，关系模型中的域、属性、实体关系等；一类是与数据之间联系有关的对象，例如网状模型中的关系型、关系模型中反映联系的关系等。

通常按数据结构的类型来命名数据模型。已有四种数据结构类型，它们是层次结构、网状结构、关系结构和面向对象结构，它们所对应的数据模型分别命名为层次模型、网状模型、关系模型和面向对象模型。

2) 数据操作

数据操作用于描述系统的动态特性，是指对数据库中各种对象及对象的实例允许执行的操作的集合，包括对象的创建、修改和删除，对对象实例的检索和更新(例如添加、删除和修改)两大类操作及其他有关的操作等。数据模型必须定义这些操作的确切含义、操作符号、操作规则(如优先级)以及实现操作的语言等。

3）数据完整性约束

数据完整性约束是一组完整性规则的集合。完整性规则是给定的数据模型中数据及其联系所具有的制约和储存规则，用以限定符合数据模型的数据库状态以及状态的变化，以保证数据的正确、有效、相容。

数据模型应该反映和规定本数据模型必须遵守的基本的通用的完整性约束条件。例如，在关系模型中，任何关系必须满足实体完整性和参照完整性两个条件。

此外，数据模型还应该提供自定义完整性约束条件的机制，以反映具体应用所涉及的数据必须遵守的特定的语义约束条件。例如，在学校的数据库中规定大学生入学年龄不得超过 40 岁，硕士研究生入学不得超过 45 岁，学生累计成绩不得有三门以上不及格等，这些应用系统数据的特殊约束要求，用户能在数据模型中自己来定义（所谓自定义完整性）。

数据模型的三要素紧密依赖相互作用形成一个整体（如图 1-4 所示），如此才能全面正确地抽象、描述来反映现实世界数据的全面特征。这里对基于关系模型的三要素示意图说明 3 点：

（1）内圈中表及表间连线，代表着数据结构；

（2）带操作方向的线段代表着动态的各类操作（包括数据库内的更新，数据库内外间的添加、删除、修改及查询等操作），代表着数据模型的数据操作要素；

（3）静态的数据结构及动态的数据操作要满足的制约条件（各椭圆示意）是数据模型的数据完整性约束条件。

还要说明的是图 1-4 是简单化、逻辑示意的图，数据模型的三要素在数据库中都是严格定义的一组概念的集合。在关系数据库可以简单理解为：数据结构是表结构定义及其他数据库对象定义的命令集；数据操作是数据库管理系统提供的数据操作（如操作命令、命令语法规定与参数指定等）命令集；数据完整性约束是各关系表约束的定义及动态操作约束规则等的集合。为此数据模型的三要素并不抽象，读者需细细领会。

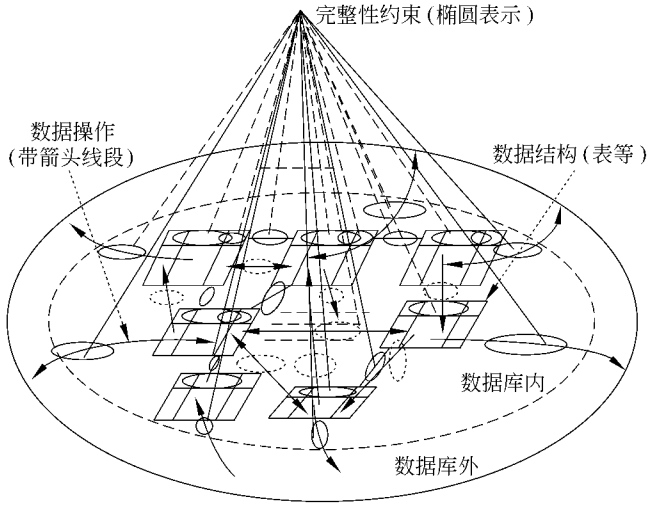

图 1-4　数据模型的三要素示意图

1.2.3　关系数据模型

关系模型是目前最重要的一种数据模型。美国 IBM 公司的研究员 E. F. Codd 于 1970 年发表题为“大型共享系统的关系数据库的关系模型”的论文，文中首次提出了数据库系统的关系模型。20 世纪 80 年代以来，计算机厂商新推出的 DBMS 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都是以关系方法为基础的。关系模型作为数据模型中最重要的一种模型，也有数据模型的三个组成要素。

1. 关系模型的数据结构

关系模型中数据的逻辑结构是一张二维表，它由行和列组成。每一行称为一个元组，每一列称为一个属性(或字段)。二维表这种简单的数据结构能够表达丰富的语义，能描述出信息世界的实体以及实体间的各种联系。

下面通过图 1-5 所示的教师登记表，介绍关系模型中的相关的术语。

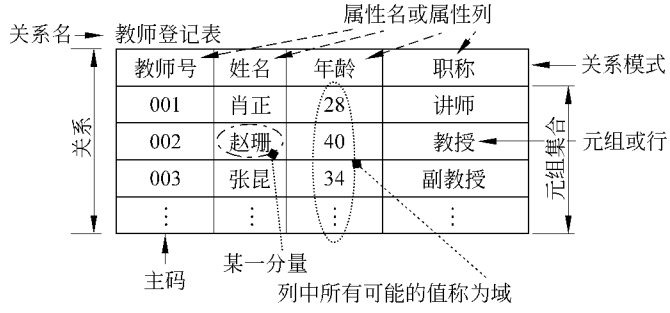

图 1-5　关系模型的数据结构及术语

- 关系：一个关系对应一张二维表，图 1-5 表示的就是一张教师登记表。
- 元组：二维表中的一行称为一个元组。
- 属性：二维表中的一列称为一个属性，对应每一个属性的名字称为属性名。如图 1-5 所示的表有四列，对应四个属性(教师号，姓名，年龄，职称)。
- 主码：如果二维表中的某个属性或最小属性组可以唯一确定一个元组，则称为主码，也称为关系键，如图 1-5 中的教师号，可以唯一确定一个教师，也就成为本关系的主码。
- 域：属性的取值范围称为域，如人的年龄一般在 1～120 岁之间，大学生的年龄属性的域是 14～38，性别的域是男和女两个值等。
- 分量：元组中的一个属性值。例如，教师号对应的值 001、002、003 都是分量。
- 关系模式：表现为关系名和属性的集合，是对关系的具体描述。一般表示为：关系名(属性 1，属性 2，…，属性 N)。在关系模型中，实体以及实体间的联系都是用关系来表示。例如教师、课程、教师与课程之间的多对多联系在关系模型中可以表示为：教师(教师号，姓名，年龄，职称)、课程(课程号，课程名，学分)、授课(教师号，课程号)。

2. 关系模型的数据操纵与完整性约束条件

关系模型对具体数据的操作主要包括查询、添加、删除和修改四类。这些操作必须满足关系的完整性约束条件，即实体完整性、参照完整性和用户定义完整性。

关系模型把对数据的存取路径向用户隐蔽起来，操作时用户只要指出“干什么”，不必详细说明“怎么干”，从而大大地提高了数据的独立性。

3. 关系模型的存储结构

在关系数据模型中，实体及实体间的联系都用表来表示。在数据库的物理组织中，表以文件形式存储，每一个表通常对应一个文件结构，也有多个表对应一个文件结构的。

关系模型具有以下特点：

- 较强的数学理论基础；
- 数据结构简单、清晰，用户易懂易用，表达能力强；
- 关系模型的存取路径对用户透明。因此数据库具有更高的数据独立性、更好的安全保密性，也简化了程序员的工作和数据库开发与建立的工作等，使其极具生命力而广泛沿用至今。

1.2.4　关系操作

关系模型给出了关系操作的能力，它利用基于数学的方法来表达关系操作。关系数据语言可以分成 3 类。

(1) 关系代数：用对关系的集合运算表达查询要求，如 ISBL。

(2) 关系演算：用谓词来表达查询要求的方式，可再分为两类。

① 元组关系演算：谓词变元的基本对象是元组变量，例如 APLHA、QUEL。

② 域关系演算：谓词变元的基本对象是域变量，例如 QBE。

(3) 关系数据语言，例如 SQL(Structured Query Language)。

关系代数、关系演算均是抽象的查询语言，这些抽象的语言与具体的 DBMS 中实现的实际语言并不完全一样。但它们能用作评估实际系统中查询语言能力的标准或基础。实际的查询语言除了提供关系代数或关系演算功能外，还提供了很多附加功能，例如集函数、关系赋值、算术运算等。

这些关系数据语言的共同特点是：语言具有完备的表达能力，表达时不需要指定存取路径，是非过程化的集合式操作语言，功能强，SQL 命令还能够嵌入到高级语言中使用。

基于关系(关系表是元组或记录或行的集合)集合运算的关系代数，能抽象而直观地表达对关系的各种操作，其中常用的关系操作有：选择(select)、投影(project)、连接(join)、除(divide)、并(union)、交(intersection)、差(difference)等查询(query)操作和添加(insert)、删除(delete)、修改(update)等更新操作。查询的表达能力是其中最主要的部分。

关系数据语言 SQL 是数据库操作的国际标准语言，它是一种介于关系代数和关系演算之间的语言。SQL 不但具有丰富的查询功能，而且具有数据定义、数据操纵和数据控制功能。它充分体现了关系数据语言的特点和优点。

SQL 语言是数据库应用系统中操作数据所必需的，其重要性不言而喻。SQL 语言及其

在数据库应用系统中的灵活应用本书将在后面再叙。

1.3　关系规范化设计理论和方法

面对一个现实问题，如何选择一个比较好的关系模式的集合？其中每个关系模式又由哪些属性组成？这就是关系规范化设计（又称数据库逻辑设计）主要关心的问题。

1.3.1　不合理的关系模式存在的问题

关系数据库设计时要遵循一定的规范化理论。只有这样才可能设计出一个较好的数据库来。关系数据库设计的关键所在是关系数据库模式的设计，也就是关系模式的设计。

实践证明关系模式的设计是有好坏之分的。某些不好的关系模式可能导致：数据冗余、插入异常、删除异常、修改异常等问题。产生这些问题的原因，直观地说，是因为关系中“包罗万象”，内容太杂了。一个好的关系模式不应该产生如此多的问题。

要设计的关系模式中的各属性是相互依赖、相互制约的，关系的内容实际上是这些依赖与制约作用的结果。关系模式的好坏也是由这些依赖与制约作用产生的。为此，在关系模式设计时，必须从实际出发，从语义上分析这些属性间的依赖关系，由此来做关系的规范化工作。

一般而言，规范化设计关系模式，是将结构复杂（即依赖与制约关系复杂）的关系分解成结构简单的关系，从而把不好的关系数据库模式转变为较好的关系数据库模式，这就是所谓的关系规范化。

1.3.2　规范化设计理论和方法

一个关系属性间的依赖情况有多种，一般主要根据属性间的函数依赖情况来判定关系是否具有某些不合适的性质。通常按属性间依赖情况来区分关系规范化的程度为第一范式、第二范式、第三范式、BC 范式和第四范式等。同时直观地描述如何将具有不合适性质的关系转换为更合适的形式。

1. 函数依赖

定义 1.1　设关系模式 $R(U,F)$，U 是属性全集，F 是 U 上的函数依赖集，X 和 Y 是 U 的子集，如果对于 $R(U)$ 的任意一个可能的关系 r，对于 X 的每一个具体值，Y 都有唯一的具体的值与之对应，则称 X 函数决定 Y，或 Y 函数依赖于 X，记 $X\rightarrow Y$。这里称 X 为决定因素，Y 为依赖因素或被决定因素。当 Y 函数不依赖于 X 时，记作：$X\nrightarrow Y$。当 $X\rightarrow Y$ 且 $Y\rightarrow X$ 时，则记作：$X\leftrightarrow Y$。

函数依赖有三种分类分别是：一是平凡的函数依赖与非平凡的函数依赖；二是完全函数依赖和部分函数依赖；三是传递函数依赖和非传递函数依赖（或直接函数依赖）。

2. 函数依赖的基本性质

（1）投影性，根据平凡的函数依赖的定义可知，一组属性函数决定它的所有可能的

子集。

(2) 扩张性，若 $X \to Y$ 且 $W \to Z$，则 $(X,W) \to (Y,Z)$。

说明：扩张性实现了两函数依赖决定因素与被决定因素分别合并后仍保持决定关系。

(3) 合并性，若 $X \to Y$ 且 $X \to Z$ 则必有 $X \to (Y,Z)$。

说明：决定因素相同的两函数依赖，它们的被决定因素合并后，函数依赖关系仍保持。

(4) 分解性，若 $X \to (Y,Z)$，则 $X \to Y$ 且 $X \to Z$。很显然，分解性为合并性的逆过程。

说明：决定因素能决定全部，当然也能决定全部中的部分。

由合并性和分解性，很容易得到以下事实：$X \to A_1, A_2, \cdots, A_n$ 成立的充分必要条件是 $X \to A_i (i=1,2,\cdots,n)$ 成立。

3. 码

定义 1.2　设 K 为 $R(U,F)$ 中的属性或属性集，若 $K \xrightarrow{f} U$ 则 K 为 R 的候选码(或候选关键字或候选键)。若候选码多于一个，则选定其中的一个为主码(或称主键)。

包含在任何一个候选码中的属性，叫做主属性。不包含在任何候选码中的属性称为非主属性或非码属性。

4. 范式

规范化的基本思想是消除关系模式中的数据冗余，消除数据依赖中的不合适的部分，解决数据插入、删除与修改时发生的异常现象。这就要求关系数据库设计出来的关系模式要满足一定的条件。我们把关系数据库的规范化过程中，为不同程度的规范化要求设立的不同的标准或准则称为范式(normal form)。满足最低要求的叫第一范式(1NF)。在第一范式中满足进一步要求的为第二范式(2NF)，其余依此类推。R 为第几范式就可以写成 $R \in xNF$(x 表示某范式名)。

从范式来讲，主要是由 E. F. Codd 先做的工作。从 1971 年起，Codd 相继提出了关系的三级规范化形式，即第一范式、第二范式、第三范式(3NF)。1974 年，Boyce 和 Codd 共同提出了一个新的范式概念，即 Boyce-Codd 范式，简称 BCNF。1976 年 Fagin 提出了第四范式(4NF)，后来又有人定义了第五范式(5NF)。至此在关系数据库规范中建立了一系列范式。

当把某范式看成是满足该范式的所有关系模式的集合时，各个范式之间的集合关系可以表示为：5NF⊂4NF⊂BCNF⊂3NF⊂2NF⊂1NF。

一个低一级范式的关系模式，通过模式分解可以转换为若干个高一级范式的关系模式的集合，这种过程就叫规范化。

5. 第一范式

第一范式(first normal form)是最基本的规范化形式，其定义为：

定义 1.3　如果关系模式 R 所有的属性均为简单属性，即每个属性都是不可再分的，则称 R 属于第一范式，简称 1NF，记作 $R \in 1NF$。

在关系数据库系统中只讨论规范化的关系，凡是非规范化的关系模式必须转化成规范化的关系。在非规范化的关系中去掉组合性属性就能转化成规范化的关系。

6. 第二范式

定义 1.4　如果关系模式 $R \in 1NF$，$R(U,F)$ 中的所有非主属性都完全函数依赖于任意一个候选关键字，则称关系 R 是属于第二范式，简称 2NF，记作 $R \in 2NF$。

从定义可知，满足第二范式的关系模式 R 中，不可能有某非主属性对某候选关键字存在部分函数依赖。

2NF 规范化是指把 1NF 关系模式通过投影分解，消除非主属性对候选关键字的部分函数依赖，转换成 2NF 关系模式的集合的过程。分解时遵循的原则是“一事一地”，让一个关系只描述一个实体或实体间的联系。如果多于一个实体或联系，则进行投影分解。

7. 第三范式

定义 1.5　如果关系模式 $R \in 2NF$，$R(U,F)$ 中所有非主属性对任何候选关键字都不存在传递函数依赖，则称 R 是属于第三范式，简称 3NF，记作 $R \in 3NF$。

第三范式具有如下性质：如果 $R \in 3NF$，则 R 也是 2NF，反之则不然。

2NF 的关系模式解决了 1NF 中存在的一些问题，但 2NF 的关系模式在进行数据操作时，仍然存在一些问题的，需要继续分解以达到较高范式等级。

3NF 规范化是指把 2NF 关系模式通过投影分解，消除非主属性对候选关键字的传递函数依赖，而转换成 3NF 关系模式集合的过程。根据“一事一地”原则可继续将只属于 2NF 的关系模式规范为 3NF。

关系模式规范化到 3NF 后，所存在的异常现象已经基本消失。但是，3NF 只限制了非主属性对码的依赖关系，而没有限制主属性对码的依赖关系。如果发生了这种依赖，仍有可能存在数据冗余、插入异常、删除异常和修改异常。这时，则需对 3NF 进一步规范化，消除主属性对码的依赖关系，向更高一级的范式 BCNF 转换。

8. BC 范式

定义 1.6　如果关系模式 $R \in 1NF$，且所有的函数依赖 $X \rightarrow Y$（Y 不包含于 X，即 $Y \not\subseteq X$），决定因素 X 都包含了 R 的一个候选码，则称 R 属于 BC 范式，记作 $R \in BCNF$。

由 BCNF 的定义可以得到以下结论，一个满足 BCNF 的关系模式有：

(1) 所有非主属性对每一个候选码都是完全函数依赖；

(2) 所有的主属性对每一个不包含它的候选码都是完全函数依赖；

(3) 没有任何属性完全函数依赖于非码的任何一组属性。

由于 $R \in BCNF$，按定义排除了任何属性对候选码的传递依赖与部分依赖，所以 $R \in 3NF$。但若 $R \in 3NF$，则 R 未必属于 BCNF。

BCNF 规范化是指把 3NF 的关系模式通过投影分解转换成若干 BCNF 关系模式的集合。

一个模式中的关系模式如果都属于 BCNF，那么在函数依赖范畴内，它已实现了彻底的分离，已消除了插入和删除异常。

在数据库课程设计中，应对所设计系统的关系模式加以范式判定，就一般而言，所设计关系模式应达到第三或以上范式等级要求。

1.4　数据库设计

1.4.1　数据库设计概述

1. 数据库设计的任务

数据库设计是指根据用户需求设计数据库结构及应用系统的过程。具体地说，数据库设计是指对于给定的应用环境，构造最优的数据库模式，建立数据库及其应用系统，使之能有效地存储数据，满足用户的信息和处理要求，也就是把现实世界中的数据，根据各种应用处理的要求，加以合理组织，能利用已有的 DBMS 来建立能够实现系统目标的数据库及其应用系统。数据库设计的优劣将直接影响信息系统的质量和运行效果。因此，设计一个结构优化的数据库是对数据进行有效管理与正确利用的前提和保证。

2. 数据库设计的内容

数据库设计内容包括数据库的结构设计和数据库的行为设计两个方面的。

数据库的结构设计是指根据给定的应用环境，进行数据库的模式设计或子模式的设计。它包括数据库的概念设计、逻辑设计和物理设计，即设计数据库框架或数据库结构。数据库结构是静态的，稳定的，一经形成后通常情况下是不容易也不需要改变的，所以结构设计又称为静态模式设计。

数据库的行为设计是指数据库用户的行为和动作。在数据库系统中，用户的行为和动作指用户对数据库的操作，这些要通过应用程序来实现，所以数据库的行为设计就是操作数据库的应用程序的设计。即设计应用程序、事务处理等，所以行为设计是动态的，行为设计又称为动态模式设计。

3. 数据库设计的特点

数据库设计既是一项涉及多学科的综合性技术，又是一项庞大的软件工程项目，具有如下特点：

- 数据库建设是硬件、软件和干件（技术和管理的界面）的结合；
- 数据库设计应该与应用系统设计相结合，也就是说要把行为设计和结构设计密切结合起来，是一种“反复探寻，逐步求精的过程”。

首先从数据模型开始设计，以数据模型为核心进行展开，将数据库结构设计和行为设计相结合，建立一个完整、独立、共享、冗余小和安全有效的数据库及其应用系统。

1.4.2　数据库设计的步骤

由于数据库设计是一项工程技术，需要科学理论和工程方法作为指导，否则，工程的质量很难保证。为了使数据库设计更合理、更有效，人们通过努力探索，提出了各种各样的数据库设计方法，其中运用了软件工程的思想和方法，提出的数据库规范化设计方法一直沿用至今。按照规范化的设计方法以及数据库应用系统开发过程，数据库的设计过程可分为以下六个设计阶段（见图 1-6）：需求分析、概念结构设计、逻辑结构设计、物理结构设计、数据

库的实施、数据库运行与维护。

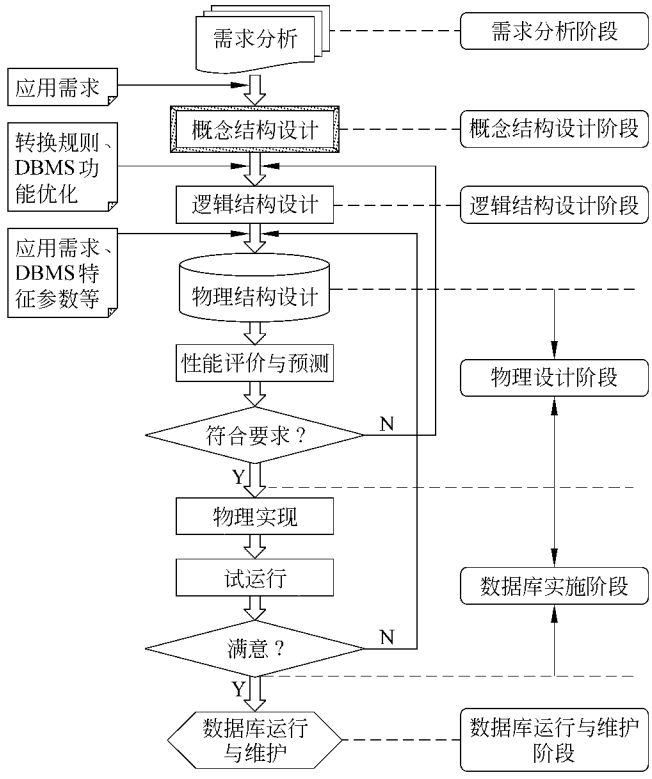

图 1-6　数据库设计步骤

在数据库设计中，前两个阶段是面向用户的应用要求，面向具体的问题，中间两个阶段是面向数据库管理系统，最后两个阶段是面向具体的实现方法。前 4 个阶段可统称为“分析和设计阶段”，后面两个阶段统称为“实现和运行阶段”。

在数据库设计之前，首先必须选择参加设计的人员，包括系统分析人员、数据库设计人员和程序员、用户和数据库管理员。系统分析和数据库设计人员是数据库设计的核心人员，他们将自始至终参加数据库的设计，他们的水平决定了数据库系统的质量。用户和数据库管理员在数据库设计中也是举足轻重的人物，他们主要参加需求分析和数据库的运行维护，他们的积极参与不但能加速数据库的设计，而且也是决定数据库设计是否成功的重要因素，程序员是在系统实施阶段参与进来，分别负责编制程序和准备软硬件环境。

如果所设计的数据库应用系统比较复杂，还应该考虑是否需要使用数据库设计工具和 CASE 工具等，以提高数据库设计的质量并减少设计工作量。

以下是数据库设计 6 个步骤的具体内容：

1. 需求分析阶段

需求分析是指准确了解和分析用户的需求，这是最困难、最费时、最复杂的一步，但也是最重要的一步。它决定了以后各步设计的速度和质量。

1）需求分析的任务

需求分析的任务是通过详细调查现实世界要处理的对象(组织、部门、企业等)，通过充分对原系统的工作概况的了解，明确用户的各种需求(数据需求、完整性约束条件、事物处理和安全性要求等)，然后在此基础上确定新系统的功能，新系统必须充分考虑到今后可能的扩充和变化，不能只是仅仅按当前应用需求来设计数据库及其功能要求。

数据库需求分析的任务主要包括“数据或信息”和“处理”两个方面：

(1) 信息要求指用户需要从数据库中获得信息的内容与性质。由信息要求可以导出各种数据要求；

(2) 处理要求指用户有什么处理要求(如响应时间、处理方式等)，最终要实现什么处理功能。

具体而言，需求分析阶段的任务包括以下方面：调查、收集、分析用户需求，确定系统边界；编写系统需求分析说明书等。

2）需求分析的方法

调查了解了用户的需求以后，还需要进一步分析和表达用户的需求，用于需求分析的方法有很多种，主要的方法有自顶向下和自底向上两种，其中结构化分析方法(Structured Analysis，SA)是一种简单实用的方法。SA 方法是从最上层的系统组织入手，采用自顶向下、逐层分解的方法分析系统。

SA 方法把每个系统都抽象成图 1-7 的形式。图 1-7 只是给出了最高层次抽象的系统概貌，要反映更详细的内容，可将处理功能分解为若干个子系统，每个子系统还可以继续分解，直到把系统工作过程表示清楚为止。在处理功能逐步分解的同时，它们所用的数据也逐级分解，形成有若干层次的数据流图。

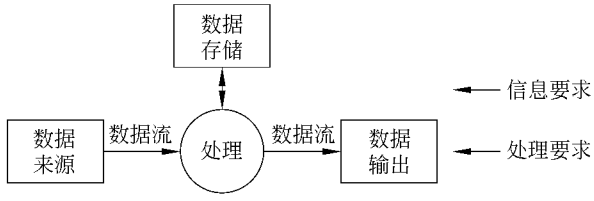

图 1-7　系统最高层数据抽象图

数据流图表达了数据和处理过程的关系。在 SA 方法中，处理过程的处理逻辑常常借助判定表和判定树来描述。系统中的数据则借助数据字典(Data Dictionary，DD)来描述。

2. 概念结构设计阶段

概念结构设计是指对用户的需求进行综合、归纳与抽象，形成一个独立于具体 DBMS 的概念模型，是整个数据库设计的关键。在需求分析阶段所得到的应用要求应该首先抽象为信息世界的结构，才能更好地，更准确地用某一 DBMS 实现这些需求。

人们提出了许多概念模型，其中最著名、最简单实用的一种是 E-R 模型，它将现实世界的信息结构统一用属性、实体以及实体间的联系来描述。

1）概念结构的设计方法

设计概念结构的 E-R 模型可采用 4 种方法。

(1) 自顶向下　首先定义全局概念结构的框架,然后逐步细化。

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

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

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

其中最常用的方法是自底向上,即自顶向下地进行需求分析,再自底向上地设计概念模式结构。

2) 概念结构设计的步骤

对于自底向上的设计方法来说,概念结构的步骤分为两步:

(1) 进行数据抽象,设计局部 E-R 模型。

选择好一个局部应用之后,就要对每个局部应用逐一设计分 E-R 图,也称局部 E-R 图。将各局部应用涉及的数据分别从数据字典中抽取出来,参照数据流图,标定各局部应用中的实体、实体的属性、标识实体的键,确定实体之间的联系及其类型(1∶1,1∶n,m∶n)。

实际上实体和属性是相对而言的,往往要根据实际情况进行必要的调整,在调整时要遵守两条原则:

① 属性不能再具有需要描述的性质。即属性必须是不可分的数据项,不能再由另一些属性组成;

② 属性不能与其他实体具有联系,联系只发生在实体之间。

符合上述两条特性的事物一般作为属性对待。为了简化 E-R 图的处置,现实世界中的事物凡能够作为属性对待的,应尽量作为属性。

(2) 集成各局部 E-R 模型,形成全局 E-R 模型。

各个局部视图即分 E-R 图建立好后,还需要对它们进行合并,集成为一个整体的概念数据结构即全局 E-R 图。也就是视图的集成,视图的集成有两种方式:一是一次集成法:一次集成多个分 E-R 图,通常用于局部视图比较简单时;二是逐步累积式:首先集成两个局部视图(通常是比较关键的两个局部视图),以后每次将一个新的局部视图集成进来。

不管用哪种方法,集成局部 E-R 图的都分为两个步骤:

① 合并分 E-R 图,生成初步 E-R 图。

这个步骤将所有的局部 E-R 图综合成全局概念结构。全局概念结构不仅要支持所有的局部 E-R 模型,而且必须合理地完成一个完整、一致的数据库概念结构。由于各个局部应用所面向的问题不同且由不同的设计人员进行设计,所以各个分 E-R 图之间必定会存在许多不一致的地方,我们称之为冲突。因此合并分 E-R 图时并不能简单地将各个分 E-R 图画到一起,而是必须着力消除各个分 E-R 图中不一致的地方,以形成一个能为全系统中所有用户共同理解和接受的统一概念模型。合理消除各分 E-R 图的冲突是合并分 E-R 图的主要工作与关键所在。E-R 图中的冲突有 3 种:属性冲突、命名冲突与结构冲突。

② 消除不必要的冗余,设计基本 E-R 图。

在初步的 E-R 图中,可能存在冗余的数据和冗余的实体间联系,冗余的数据是指可由

基本数据导出的数据，冗余的联系是指可由其他联系导出的联系。冗余数据和冗余联系容易破坏数据库的完整性，给数据库维护增加困难，当然并不是所有的冗余数据与冗余联系都必须加以消除，有时为了提高某些应用的效率，不得不以冗余信息作为代价。设计数据库概念模型时，哪些冗余信息必须消除，哪些冗余信息允许存在，需要根据用户的整体需求来确定。我们把消除不必要的冗余后的初步 E-R 图称为基本 E-R 图。采用分析的方法来消除数据冗余，以数据字典和数据流图为依据，根据数据字典中关于数据项之间逻辑关系的说明来消除冗余。

3. 逻辑结构设计阶段

逻辑结构设计是指将概念模型转换成某个 DBMS 所支持的数据模型，并对其进行优化。

1）逻辑结构设计的任务和步骤

概念结构是各种数据模型的共同基础。为了能够用某一 DBMS 实现用户需求，必须将概念结构进一步转化为相应的数据模型，这正是数据库逻辑结构设计所要完成的任务。

一般的逻辑结构设计分为以下 3 个步骤（见图 1-8）。

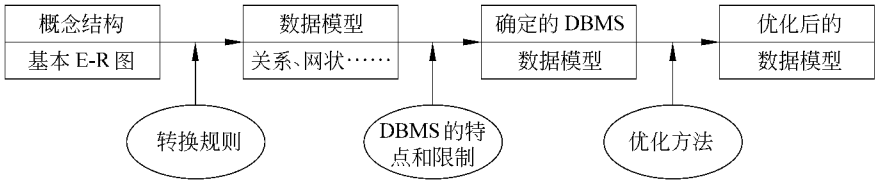

图 1-8　逻辑结构设计 3 步骤

- 将概念结构转化为一般的关系、网状、层次模型。
- 将转化来的关系、网状、层次模型向特定 DBMS 支持下的数据模型转换。
- 对数据模型进行优化。

2）将 E-R 图转化为关系模式

概念设计中得到的 E-R 图是由实体、属性和联系组成的，而关系数据库逻辑设计的结果是一组关系模式的集合，所以将 E-R 图转换为关系模型实际上是将实体、属性和联系转换成关系模式。在转换过程中要遵守以下原则：

（1）一个实体转换为一个关系模式。

- 关系的属性：实体的属性。
- 关系的键：实体的键。

（2）一个 $m:n$ 联系转换为一个关系模式。

- 关系的属性：与该联系相连的各实体的键以及联系本身的属性。
- 关系的键：各实体键的组合。

（3）一个 $1:n$ 联系可以转换为一个关系模式。

- 关系的属性：与该联系相连的各实体的码以及联系本身的属性。
- 关系的码：n 端实体的键。

说明：一个 $1:n$ 联系也可以与 n 端对应的关系模式合并，这时需要把 1 端关系模式的

码和联系本身的属性都加入到 n 端对应的关系模式中。

(4) 一个 1∶1 联系可以转换为一个独立的关系模式。

- 关系的属性：与该联系相连的各实体的键以及联系本身的属性。
- 关系的候选码：每个实体的码均是该关系的候选码。

说明：一个 1∶1 联系也可以与任意一端对应的关系模式合并，这时需要把任一端关系模式的码及联系本身的属性都加入到另一端对应的关系模式中。

(5) 3 个或 3 个以上实体间的一个多元联系转换为一个关系模式。

- 关系的属性：与该多元联系相连的各实体的键以及联系本身的属性。
- 关系的码：各实体键的组合。

3) 关系模式的优化

数据库逻辑设计的结果不是唯一的。为了进一步提高数据库应用系统的性能还应该根据应用需要适当地修改、调整数据模型的结构，也就是对数据库模型进行优化，关系模型的优化通常是以规范化理论为基础，方法如下：

(1) 确定数据依赖，按需求分析阶段所得到的语义，分别写出每个关系模式内部各属性之间的数据依赖以及不同关系模式属性之间的数据依赖。

(2) 对于各个关系模式之间的数据依赖进行极小化处理，消除冗余的联系。

(3) 按照数据依赖的理论对关系模式逐一进行分析，考查是否存在部分函数依赖、传递函数依赖、多值依赖等，确定各关系模式分别属于第几范式。

(4) 按照需求分析阶段得到的各种应用对数据处理的要求，分析对于这样的应用环境这些模式是否合适，确定是否要对它们进行合并或分解。需求时对关系模式进行必要的分解或合并，以提高数据操作时间效率和存储空间的利用率。

4) 关系模式的评价与改进

在初步完成数据库逻辑结构设计之后，在进行物理设计之前，应对设计出的逻辑结构(这里为关系模式)的质量和性能进行评价，以便改进。

(1) 模式的评价，对模式的评价包括设计质量的评价和性能评价两个方面。

(2) 数据模式的改进，根据对数据模式的性能估计，对已生成的模式进行改进。如果因为系统需求分析、概念结构设计的疏忽导致某些应用不能支持，则应该增加新的关系模式或属性。如果因为性能考虑而要求改进，则可使用合并或分解的方法。

4. 物理设计阶段

物理设计是指为逻辑数据模型选取一个最适合应用环境的物理结构(包括存储结构和存取方法)，然后对该存储模式进行性能评价、修改设计，经过多次反复，最后得到一个性能较好的存储模式。

1) 确定物理结构

物理设计不仅依赖于用户的应用要求，而且依赖于数据库的运行环境，即 DBMS 和设备特性。数据库物理设计内容包括记录存储结构的设计、存储路径的设计、记录集簇的设计等。

(1) 记录存储结构的设计。逻辑模式表示的是数据库的逻辑结构，其中的记录称为逻辑记录，而存储记录则是逻辑记录的存储形式，记录存储结构的设计就是设计存储记录的结构形式，它涉及不定长数据项的表示，数据项编码是否需要压缩和采用何种压缩，记录间互

联指针的设置以及记录是否需要分割以节省存储空间等在逻辑设计中无法考虑的问题。

(2) 关系模式的存取方法选择。数据库系统是多用户共享的系统，对同一个关系要建立多条存取路径才能满足多用户的多种应用要求。物理设计的第一个任务就是要确定选择哪些存取方法，即建立哪些存取路径。DBMS 常用存取方法有：索引方法，目前主要是 B^+ 树索引方法；聚簇(cluster)方法；HASH 方法。

2) 评价物理结构

和前面几个设计阶段一样，在确定了数据库的物理结构之后，要进行评价，重点是时间和空间的效率。如果评价结果满足设计要求，则可进行数据库实施，实际上，往往需要经过反复测试才能优化物理设计。

5. 数据库实施阶段

数据库实施是指根据逻辑设计和物理设计的结果，建立数据库，编制与调试应用程序，组织数据入库，进行测试和试运行的过程。

(1) 建立实际数据库结构：确定了数据库的逻辑结构与物理结构后，就可以用所选用的 DBMS 提供的数据定义语言(Data Description Language，DDL)来严格描述数据库结构。

(2) 装入数据：数据库结构建立好后，就可以向数据库中装载数据了。组织数据入库是数据库实施阶段最主要的工作。数据装载方法有人工方法与计算机辅助数据入库方法两种。

(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。在数据库实施阶段，当数据库结构建立好后，就可以开始全面编制与调试数据库的应用程序。调试应用程序时由于数据入库尚未完成，可先使用模拟数据。

(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，就可以开始数据库的试运行。数据库试运行也称为联合调试，其主要工作包括：

① 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；

② 性能测试：测量系统的性能指标，分析是否符合设计目标。

数据库物理设计阶段在评价数据库结构估算时间、空间指标时，作了许多简化和假设，忽略了许多次要因素，因此结果必然很粗糙。数据库试运行则是要实际测量系统的各种性能指标(不仅是时间、空间指标)，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。

(5) 整理文档：在程序的编制和试运行中，应将发现的问题和解决方法记录下来，将它们整理存档为资料，供以后正式运行和改进时参考，全部的调试工作完成之后，应该编写应用系统的技术说明书，在系统正式运行时给用户，完整的资料是应用系统的重要组成部分。

6. 数据库运行与维护阶段

数据库运行与维护是指对数据库系统实际正常运行使用，并时时进行评价、调整与修改。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，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。

对数据库经常性的维护工作主要是由 DBA 完成的，包括 3 个方面的内容即数据库的转

储和恢复，数据库的安全性、完整性控制，数据库性能的监督、分析和改进。

（1）数据库的安全性、完整性：DBA 必须根据用户的实际需要授予不同的操作权限，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 需要根据实际情况修改原有的安全性控制。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。

（2）监视并改善数据库性能：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。

（3）数据库的重组织和重构造：为什么要重组织数据库？因为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。因此要对数据库进行重新组织，即重新安排数据的存储位置，回收垃圾，减少指针链，改进数据库的响应时间和空间利用率，提高系统性能。DBMS 一般都提供了供重组织数据库使用的实用程序，帮助 DBA 重新组织数据库。

数据库的重组织，并不改变原设计的逻辑和物理结构，而数据库的重构造则不同，它是指部分修改数据库的模式和内模式。

由于数据库应用环境发生变化，增加了新的应用或新的实体，取消了某些旧的应用，有的实体与实体间的联系也发生了变化等，使原有的数据库设计不能满足新的需要，必须要调整数据库的模式和内模式。例如，在表中增加或删除某些数据项，改变数据项的类型，增加或删除某个表，改变数据库的容量，增加或删除某些索引等。当然数据库的重构也是有限的，只能做部分调整与修改。如果应用变化太大，重构也无济于事，说明此数据库应用系统的生命周期已经结束，应该设计新的数据库应用系统了。

由表 1-2 可以看出，设计一个数据库及其应用系统不可能一蹴而就的，它往往是上述各个阶段的不断反复。以上 6 个阶段是从数据库应用系统设计和开发的全过程来考察数据库设计的问题。因此，它既是数据库也是应用系统的设计过程。在设计过程中，努力使数据库设计和系统其他部分的设计紧密结合，把数据和处理的需求收集、分析、抽象、设计和实现在各个阶段同时进行，并相互参照、相互补充，以完善数据和处理两个方面的设计。按照这个原则，数据库各个阶段的设计可用图 1-9 描述。

表 1-2　数据库各个设计阶段的描述

设计各阶段	设 计 描 述	
	数　　据	处　　理
需求分析	数据字典，全系统中数据项、数据流 、数据存储的描述	数据流图和判定表（或判定树）、数据字典中处理过程的描述
概念结构设计	概念模型（E-R 图）；数据字典	系统说明书。包括新系统要求、方案和概图；反映新系统信息的数据流图
逻辑结构设计	某种数据模型，如关系模型	系统结构图、模块结构图
物理设计	存储安排、存取方法选择、存取路径建立	模块设计、IPO 表
数据库实施	编写模式、装入数据、数据库试运行	程序编码、编译联结、测试
数据库运行与维护	性能测试，转储/恢复数据库重组和重构	新旧系统转换、运行、维护（修正性、适应性、改善性维护）

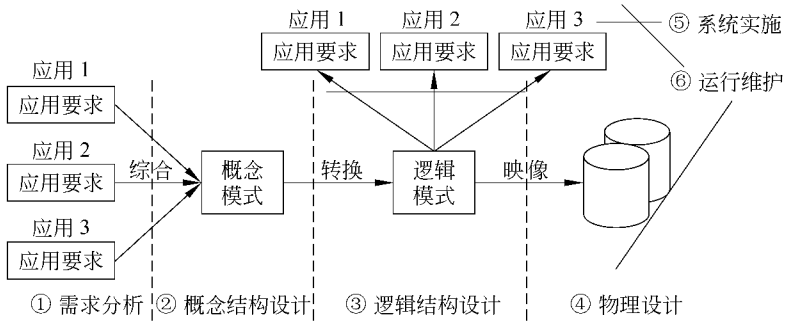

图 1-9　数据库设计过程与数据库各级模式

按照这样的设计过程，经历这些阶段能形成数据库的各级模式，如图 1-9 所示。需求分析阶段，综合各个用户的应用需求；在概念结构设计阶段形成独立于机器特点，独立于各个 DBMS 产品的概念模型，在本书中就是 E-R 图；在逻辑结构设计阶段将 E-R 图转换成具体的数据库产品支持的数据模型，如关系模型中的关系模式；然后根据用户处理的要求、安全性完整性要求等，在基本表的基础上再建立必要的视图（可认为是外模式或子模式）；在物理设计阶段，根据 DBMS 特点和处理性能等的需要，进行物理设计（如存储安排、建立索引等），形成数据库内模式；在数据库实施阶段，开发设计人员基于外模式，进行系统功能模块的编码与调试；设计成功的话就进入系统的运行与维护阶段。规范化 6 步骤的数据库应用系统开发设计过程在各课程设计案例中将有充分的体现，请读者对照阅读与领会。

1.5　数据库安全保护

数据库系统中的数据是由 DBMS 统一进行管理和控制的。为了适应和满足数据共享的环境和要求，DBMS 要保证数据库及整个系统的正常运转，防止数据意外丢失和不一致数据的产生，以及当数据库遭受意外破坏后能迅速地恢复正常，这就是数据库的安全保护。DBMS 对数据库的安全保护功能是通过四方面来实现的，即安全性控制、完整性控制、并发性控制和数据库恢复。

在进行课程设计时，在理解相关概念与知识的同时，更要努力实践数据库安全保护功能，努力使自己设计的应用系统及系统数据库具有安全性保障功能，保障其能可靠、安全、良好地运行。

1.5.1　数据库安全性概述

数据库的安全性是指保护数据库，以防止非法使用所造成数据的泄露、更改或破坏。安全性问题有许多方面，这里主要关心数据库系统本身安全性的问题。数据库本身的安全性内容，类似于整个计算机系统逐层设置安全控制的情况，其安全控制模型一般如图 1-10 所示。

根据图 1-10 的安全模型，当用户进入计算机系统时，系统首先根据输入的用户标识进

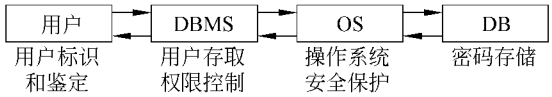

图 1-10　安全控制模型

行身份的鉴定，只有合法的用户才准许进入系统。

对已进入系统的用户，DBMS 还要进行存取控制，只允许用户进行合法的操作。

DBMS 是建立在操作系统之上的，安全的操作系统是数据库安全的前提。操作系统应能保证数据库中的数据必须由 DBMS 访问，而不允许用户越过 DBMS，直接通过操作系统或其他方式访问。

数据最后可以通过密码的形式存储到数据库中。能做到非法者即使得到了加密数据，也无法识别它的安全效果。

1. 视图机制

为不同的用户定义不同的视图，可以限制各个用户的访问范围。通过视图机制把要保密的数据对无权存取这些数据的用户隐藏起来，从而自动地对数据提供一定程度的安全保护。

2. 审计机制

前面介绍的各种数据库安全性措施，都可将用户操作限制在规定的安全范围内。但实际上任何系统的安全性措施都不是绝对可靠的，窃密者总有办法打破这些控制。对于某些高度敏感的保密数据，必须以审计作为预防手段。审计功能是一种监视措施，跟踪记录有关数据的访问活动。

1.5.2　数据库完整性概述

数据库的完整性是指保护数据库中数据的正确性、有效性和相容性，防止错误的数据进入数据库造成无效操作。

数据库的完整性和安全性是数据库保护的两个不同的方面。安全性是保护数据库，以防止非法使用所造成数据的泄露、更改或破坏，安全性措施的防范对象是非法用户和非法操作；完整性是防止合法用户使用数据库时向数据库中加入不符合语义的数据，完整性措施的防范对象是不合语义的数据。但从数据库的安全保护角度来讲，安全性和完整性又是密切相关的。

为了实现完整性控制，数据库管理员应向 DBMS 提出一组完整性规则，来检查数据库中的数据，看其是否满足语义约束。这些语义约束构成了数据库的完整性规则，这组规则作为 DBMS 控制数据完整性的依据。它定义了何时检查、检查什么、查出错误又怎样处理等事项。具体地说，完整性规则主要由以下 3 部分构成。

(1) 触发条件：规定系统什么时候使用规则检查数据。

(2) 约束条件：规定系统检查用户发出的操作请求违背了什么样的完整性约束条件。

(3) 违约响应：规定系统如果发现用户的操作请求违背了完整性约束条件，应该采取一定的动作来保证数据的完整性，即违约时要做的事情。