第3章 数据库设计

学习目标

- 理解数据库设计的特点；
- 掌握关系数据库设计的步骤，各个阶段的具体任务，特别是面向对象的需求分析、数据库的概念结构设计、逻辑结构设计的基本任务和设计的结果；
- 掌握 E-R 图的设计方法，E-R 图向关系模型的转换规则及关系模型的优化；
- 了解数据库物理结构设计的内容和方法，数据库的实施和维护；
- 掌握数据库设计的方法，根据实际应用需求，具备关系数据库设计的基本能力。

重点：需求分析、数据库的概念结构设计、数据库的逻辑结构设计。

难点：概念结构设计中的依赖实体集、强实体集、弱实体集、多值联系的建模。

在数据库领域内，通常把使用数据库的各类信息系统统称为数据库应用系统。数据库是数据库应用系统的重要组成部分，开发一个数据库应用系统一般都需要设计系统的数据库。数据库设计的好坏将直接影响着数据库的性能和程序编码的复杂程度，甚至影响整个数据库应用系统的稳定性。

3.1 数据库设计概述

如何设计第 1 章高校图书管理系统数据库结构？使用什么方法？按照什么流程完成数据库设计？有哪些步骤？每一步需要完成什么任务？这些问题将是本章需要解决的问题。

广义地讲，数据库设计是指设计整个数据库应用系统，包括数据库结构设计及其应用系统的设计；狭义地讲，数据库设计是指设计数据库本身，即设计数据库的各级模式并建立数据库，这是数据库应用系统设计的一部分。本章重点是讲解狭义的数据库设计。当然，设计一个“好”的数据库与设计一个“好”的一个数据库应用系统是密不可分的，一个好的数据库结构是应用系统的基础，在实际的系统开发中两者是密切相关、并行进行的。由于数据库应用系统结构复杂，应用环境多样，因此，设计时需要考虑的因素有很多。

3.1.1 数据库设计的基本任务和目标

在数据库设计之前，首先需要确定数据库应用系统需要解决哪些问题，达到什么目标。从本质上来讲，数据库设计实际上是指针对实际的应用问题和给定的应用环境，构造最优的数据库模式。通过对现实世界实际问题的分析，设计反映现实世界信息需求的数据库概念数据模型，并将其转换为逻辑数据模型和物理数据模型，最终建立能够服务现实世界实际问

题的数据库。

1. 基本任务

一个数据库应用系统的各部分能否紧密地结合在一起以及如何结合，关键在于数据库。只有对数据库进行合理的逻辑设计和有效的物理设计，才能有效地存储和管理数据，满足各种用户对数据的需求和对数据操作的要求，从而开发出完善而高效的数据库应用系统。因此，数据库设计的基本任务就是根据用户的信息需求、数据操作需求，设计一个结构合理、使用方便、效率较高的数据库。

信息要求指在数据库中应该存储和管理哪些数据对象；数据操作要求指对数据对象需要进行哪些操作，如增加、删除、修改、查询和统计等。通常，在充分了解了用户的信息需求、数据操作要求的基础上，结合计算机硬件、软件环境以及 DBMS 的特性，通过设计得到相应的数据模型，然后，根据数据模型创建数据库及其数据库应用系统，达到有效存储数据、实现不同用户数据操作要求的目的。

2. 设计目标

数据库应用系统是以数据为中心的，而数据库是根据数据模型建立的，因此，数据库设计的核心问题是建立一个什么样的数据模型。一般而言，模型的设计不仅要考虑系统数据的特性，而且要考虑数据库数据的存取效率、数据库存储空间的利用率、系统运行管理的效率等。具体而言，数据模型应当满足以下目标。

1）满足用户的应用要求

从系统开发的角度，设计数据库前首先需要通过采取多种调查方式了解用户的需求，用户的应用要求包括用户所需的全部数据以及支持用户所需要的系统必须完成的业务功能。由于用户往往不了解计算机和数据库，他们很难一次提出具体的完整要求，提出的需求一般具有模糊、片面、脱离实际等问题。因此，数据库设计人员必须学习相关用户现实环境的业务知识，充分理解各方面的要求与业务规则，并随时与用户进行沟通，了解情况，最终确定用户的应用要求，并得到用户的认可。

2）准确模拟现实世界

模型是对现实的抽象和模拟，是对现实世界所关心的应用环境（现实系统）的本质特征的一种抽象、简化和描述。数据库是一个组织信息流的反映，模拟越准确，就越能反映实际情况，用户对数据库就更加信赖。所谓“好”模型，就是既能反映现实世界的本质特征，又具有简单直观的表示形式。要准确地反映现实世界，除了需要依赖于数据模型本身的表达能力，还需要数据库设计者充分理解用户要求，掌握系统环境，熟练使用良好的软件工程规范与建模工具，发挥 DBMS 的特点，才能设计出一个高质量的模型。可以说，在一定的数据模型下，数据库设计的质量取决于设计者的水平。

3）能被某个 DBMS 所接受

数据库设计的结果是一个在确定的 DBMS 支持下能运行的数据模型，依照设计结果可以建立数据库。因此，在设计中必须充分掌握 DBMS 的特点，了解 DDL 和 DML、数据组织和数据存取方法、物理参数、安全性与完整性约束等，使设计结果扬长避短，充分发挥 DBMS 的特点。

4）具有良好的性能，较好的质量

一个数据库的性能应该从存取效率、存储效率以及方便维护与扩充、较好的安全性与完

整性、出现故障时恢复的难易程度等方面进行度量。而各个性能参数往往是相互冲突的、需要权衡得失、折中决定。

3.1.2 数据库设计的特点与方法

数据库设计涉及多学科的综合性技术,是一种技艺,如果缺乏科学理论和工程方法的支持,很难保证其设计的质量。

1. 数据库设计的特点

数据库建设是指数据库应用系统从分析、设计、实施到运行与维护的全过程。和一般的软件系统的设计、开发与维护有许多相同之处,也有其自身的一些特点。

1) 三分技术、七分管理、十二分基础数据

"三分技术、七分管理,十二分基础数据"是数据库建设的基本规律。要建设好一个数据库应用系统,开发技术固然重要,但是相比之下管理更为重要,管理不仅仅指项目的管理,而且更重要的是组织的业务管理。数据库结构的设计就是对组织中业务部门的数据以及各个业务部门之间数据联系的描述和抽象,而业务部门数据以及各个业务部门之间数据的联系是和各个部门的职能、整个组织的管理模式密切相关。因此,组织的业务管理对数据库结构的设计有着直接影响。

在数据库设计中,基础数据在数据库建设中的地位和作用更是不容忽视的,数据的收集、整理、组织和更新是数据库建设的重要环节。其中,基础数据的收集、入库是数据库建立初期工作量最大、最烦琐、最细致的工作;在数据库运行过程中需要不断地把新的数据加到数据库中,使数据库成为一个"活库"。如果数据库一旦成了"死库",系统也就失去了应用价值。试想,一个数据库应用系统如果不能及时更新"新"的数据,还有用户愿意去使用吗?

2) 结构设计和行为设计相结合

数据库设计要达到预期的目标,在整个设计过程中必须强调结构设计和行为设计相结合,这也是数据库设计的第二大特点。早期的结构化设计方法着重于数据处理过程的特性,即行为特性,一般把数据结构设计尽量推迟,这种方法对数据库应用系统的设计是不妥的。因此,要强调在数据库设计中把结构特性和行为特性结合起来,并将其作为数据库设计的重要特点。

结构设计是关键,因为数据库正是从分析用户的行为所涉及的数据汇总出来的。整个设计过程是一种"反复探寻、逐步求精"的过程,不能一蹴而就。数据库的逻辑模式设计要与事务设计结合起来,以支持全部事务处理的要求。因为,数据库并不是系统的全部,如果没有应用程序,只有数据存储,数据将是不可用的,如果没有期望实现的某个系统的行为,那么,数据库就失去了存在的价值。为了更有效地支持事务处理,还需要进行数据库的物理结构设计,以实现数据存取功能,数据库的子模式则是根据应用程序的需要而设计的。同时,在数据库库设计中,数据库设计者还应当具有战略眼光,考虑到当前、近期和远期某个时间段的对系统的需求,设计的系统应当完全满足用户当前和近期对系统的数据需求,对远期的数据需求有相应的处理方案。数据库系统设计者应充分考虑到系统可能的扩充与改变,使设计出的应用系统具有较长的生命力。

2. 数据库设计方法

数据库常常是在投入运行一段时间后又不同程度地发现各种问题,因而不得不进行修

改，甚至重新进行设计，增加了系统维护的代价。为此，人们努力探索，提出了各种各样的数据库设计方法，以及多种数据库设计的准则和规程，这些设计方法被称为规范化的设计方法。

1）新奥尔良方法

新奥尔良（New Orleans）方法是规范设计法中比较著名的一种方法。该方法把数据库设计分为四个阶段：需求分析（分析用户要求）、概念设计（信息分析和定义）、逻辑设计（设计实现）和物理设计（物理数据库设计），同时，采用一些辅助手段实现每一个过程。其后，许多科学家对此进行了改进，认为数据库设计应分六个阶段进行，这六个阶段分别是需求分析、概念结构设计、逻辑结构设计、物理结构设计、数据库实施以及数据库运行和维护。

2）基于 E-R 模型的数据库设计方法

在数据库设计的不同阶段，具有不同的具体实现方法，基于 E-R 模型的数据库设计方法设计数据库的概念模型，是数据库概念设计阶段广泛采用的方法。

3）基于 3NF 的设计方法

基于 3NF（第三范式）的设计方法是一种结构化的设计方法，它以关系数据理论为指导来设计数据库的逻辑数据模型，是设计关系数据库时在逻辑阶段可以采用的一种有效方法。

4）对象定义语言（object definition language，ODL）方法

ODL 方法是面向对象的数据库设计方法，该方法用面向对象的概念和术语描述数据库的结构，通过 UML（unified modeling language）的类图（对象模型）表示数据对象的汇集以及它们之间的联系。UML 是一种面向对象的、通用的、标准化的可视化图形建模语言，它提供了面向对象分析与设计方法建模结果的一种通用的符号表示。尽管其概念基于面向对象技术，但所得到的对象模型既可以用于设计关系数据库，也可以设计面向对象数据库以及对象关系数据库。

数据库工作者和数据库开发商一直在研究和开发数据库设计工具，经过多年的努力，数据库设计工具已经实用化和商品化。如 Oracle 公司开发的 Designer 2000、Sybase 公司推出的 Power Designer 等，都是比较成熟的数据库设计工具软件。这些数据库设计工具软件可以辅助数据库设计人员完成数据库设计过程中的很多任务，大大减轻了他们的工作量。目前，数据库设计工具的重要性已经被越来越多的人认识到，已经普遍应用于大型数据库设计中。

3.1.3 数据库设计步骤

按照规范化设计的方法，并遵循软件工程的思想与方法以及数据库设计的特点，考虑数据库及其应用系统开发的全过程，将数据库设计划分为六个阶段。在数据库设计之前，首先要选定参加设计的人员。

1. 准备工作

参与数据库设计的人员包括系统分析员、数据库设计人员、应用开发人员、DBA 和用户代表。其中：系统分析员和数据库设计人员是数据库设计的核心人员，他们自始至终都参与数据库设计，其水平将决定数据库系的质量；用户和 DBA 在数据库设计中也起着举足轻重的作用，他们主要参加需求分析和数据库的运行和维护，他们的积极参与不但能加速数据库设计，而且也是决定数据库设计质量的重要因素；应用开发人员负责编制程序和准备软/硬件环境，一般在系统实施阶段参与进来。

如果所设计的系统比较复杂,还应考虑是否需要使用数据库设计工具以及选用何种工具,以提高数据库设计质量并减少工作量。

2. 数据库设计的基本步骤

数据库设计可划分为六个阶段,如图 3-1 所示。

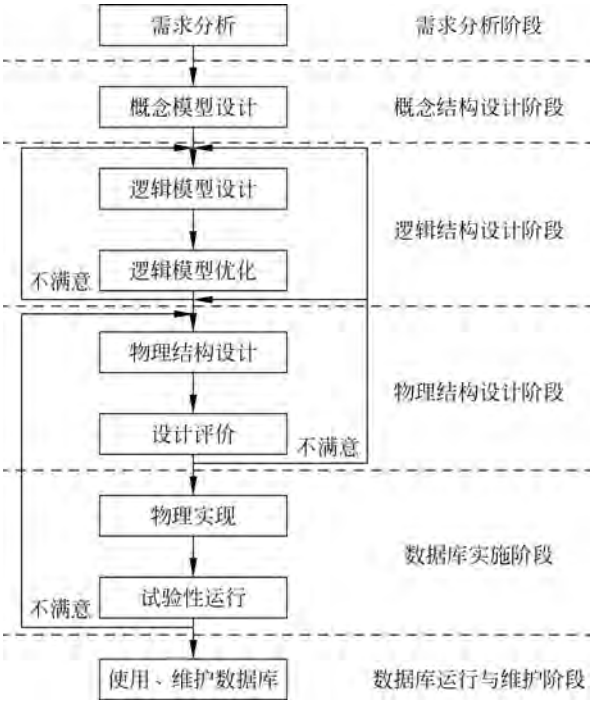

图 3-1 数据库设计步骤

在数据库设计过程中,需求分析和概念结构设计是独立于任何 DBMS 进行,逻辑结构设计与物理结构设计与选用的 DBMS 密切相关。

1) 需求分析阶段

获取需求是整个设计过程的基础。进行数据库设计时首先必须准确了解与分析用户的需求,弄清系统要达到的目标和实现的功能。面向对象方法通过用例模型描述系统功能需求。为了满足用户功能需求,还需要获取关于问题域本质内容的对象、对象的特征以及对象之间存在哪些关系和操作,从而确定系统的对象模型。建立对象模型的目的在于描述系统的构成方式,而不是系统如何协作运行。因此,对象模型设计是采用面向对象方法最终生成数据库的关键。

2) 概念结构设计阶段

概念结构设计的主要任务是根据系统分析建立的业务对象模型(实体对象),形成一个独立于具体 DBMS 的概念模型。此步骤即设计 E-R 模型。

3) 逻辑结构设计阶段

逻辑结构设计阶段的主要任务是将概念结构转换为某个 DBMS 所支持的数据模型,对于关系数据库来说,就是将 E-R 模型转化为关系模型,最终生成表,并确定表中的列,根据数据存取的性能要求优化关系模型。

4）物理结构设计阶段

数据库物理结构设计的主要任务是，为逻辑数据模型选取一个最适合应用环境的物理结构，包括数据存储结构和存取方法。真正实现规划好的数据库，是将一个满足用户信息需求的已确定的逻辑结构转换为一个有效的、可实现的物理数据库结构的过程。

5）数据库实施阶段

在数据库实施阶段中，系统设计人员要运用 DBMS 提供的数据操作语言，如 SQL 以及宿主语言，根据数据库的逻辑设计和物理设计的结果建立数据库、编制与调试应用程序、组织数据入库并进行系统试运行。

6）数据库运行和维护阶段

数据库应用系统经过试运行后即可投入正式运行。在数据库系统运行过程中，必须不断地对其结构性能进行评价、调整和修改。

设计一个完善的数据库应用系统不能一蹴而就，它往往是上述六个阶段的不断往复。需要指出的是，这六个设计步骤既是数据库设计的过程，也包括了数据库应用系统的设计过程。事实上，如果不了解应用环境对数据的操作要求或没有考虑如何去实现这些操作要求，不可能设计出一个良好的数据库结构。

本章主要讨论关于数据特性的描述以及如何在整个设计过程中参照操作要求来完善模型设计等问题，关于操作特性的设计方法和原理，需参照软件工程、信息系统分析与设计课程的相关知识，在此不做介绍。

3. 数据库设计过程中的各级模式

在数据库设计的不同阶段形成数据库的三层模型和数据库的各级模式之间的关系，如图 3-2 所示。

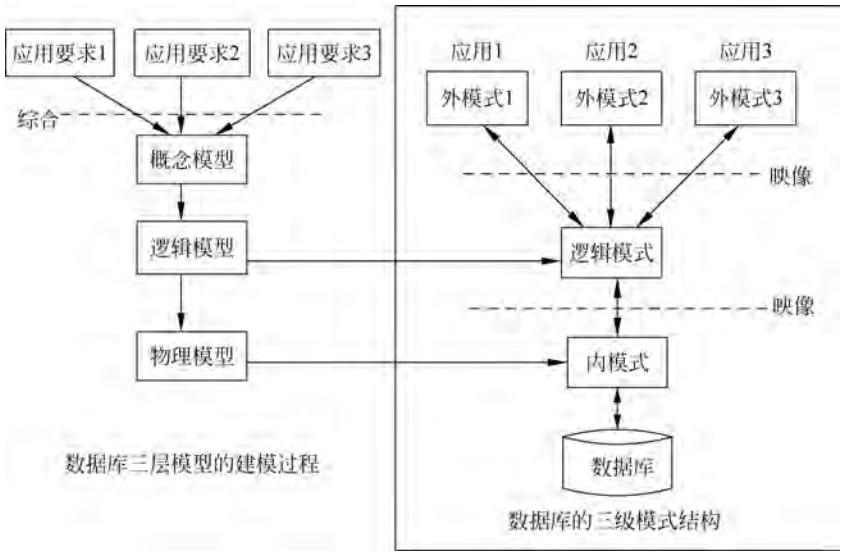

图 3-2　数据库的三层模型和各级模式之间的关系

图 3-2 表明，在需求分析阶段，设计的中心工作是综合不同的应用需求；在概念结构设计阶段，要形成与计算机硬件无关的、与各个 DBMS 产品无关的概念模型（即 E-R 图）；在

逻辑设计阶段,要完成模式和外模式的设计工作,即系统设计者首先要将 E-R 图转换成具体的数据库产品支持的逻辑模型,形成数据库逻辑模式,然后根据用户处理的要求、安全性的考虑,在模式的基础上再建立必要的视图,形成各应用的外模式;在物理结构设计阶段,要根据 DBMS 特点和处理的需要,将逻辑模型转换为 DBMS 所支持的物理模型,进行物理存储安排,建立索引,得出数据库的内模式。

本章以图 3-1 所示的数据库设计步骤的设计过程为主线,以高校图书管理系统中的数据库设计为例,讨论数据库设计各阶段的设计内容、设计方法和工具。

3.2 需求分析

需求分析就是正确理解用户需求并表达用户的需求。正确理解需求并表达用户需求是成功开发系统的基础,作为"基地"的需求分析是否能够准确地反映用户的实际要求、是否做得充分与准确,将直接影响到后面各个阶段的设计,也决定着在其上构建数据库"大厦"的速度与质量,并影响到系统的设计是否合理和实用。也就是说,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工作重做。

3.2.1 需求分析的任务

系统开发的首要步骤是进行需求调研,了解系统所属组织的业务流程,并从用户那里提取准确的、必要的系统需求,然后以用户可以理解的方式描述需求,以便使需求得到用户的认可。

面向对象分析的主要工作是把问题域中的事物抽象为系统中的对象,并建立一个用面向对象概念表达的系统模型。面向对象的系统开发过程中,作为客户方和开发方契约的用例建模是面向对象方法分析用户需求的常用方法。传统的需求调研要么以组织的不同业务为基础,要么以组织现有的职能部门为基础确定系统的边界,划分系统功能,这种划分容易带来系统边界的不清晰和依赖关系复杂等问题。而用例方法完全站在系统外部用户的角度来描述系统功能,并指出各功能的参与者。从参与者的角度来看,他们所关心的是系统能够提供哪些服务,并不关心系统内部是如何完成所提供的功能的,这也是用例方法的基本思想。

因此,面向对象方法需求分析的主要任务如下。

1. 获取需求

系统开发的关键问题之一是获取用户的需求。分析用户需求,建立对需求的准确认识,形成对需求的规范化描述,是采用面向对象系统分析的基础。首先进行需求调研,确定系统边界,识别系统的参与者和用例,并确定系统中用例之间的关系以及各参与者和用例之间的联系,从用户的角度通过用例模型来理解用户需求。

通常,系统需求可以分为两类:功能性需求和非功能性需求。

(1) 功能性需求。功能性需求描述的是系统预期能够提供的功能或服务,包括系统需要哪些输入、对输入做出什么反应以及对系统具体行为的描述。面向对象方法是通过用例模型表达系统的功能性需求,其基本思想是考察系统边界以外的、与系统进行交互的参与者

对每一项系统功能的使用情况，通过描述每一类参与者对系统功能的使用情况，便可定义系统的功能需求。用例把系统看作“黑盒”，用例模型用来表示系统和参与者之间的交互，表达了用户对系统的功能需求，为面向对象的分析提供了良好的基础。

(2) 非功能性需求。非功能性需求是系统特性和系统性能的需求。随着软件规模的不断扩大和应用环境的日趋复杂，确定非功能性需求的指标需要考虑的因素越来越多，其主要指标包括可靠性、可用性、响应时间、并发性、吞吐量和可移植性等，通常要靠一些技术手段才能得以实现，因此也称为技术需求。如果非功能性需求得不到满足，可能会造成系统部分功能或者整个系统都无法使用。

2. 确定对象及对象间的关系

以用户的观点对系统进行了用例分析后，还需要对用户需求进行深入研究。为了满足用户需求，应该从所研究问题领域中抽象出哪些对象来构成系统，获取关于问题域本质内容的对象、对象的特征以及对象之间存在哪些关系和操作，确定系统的逻辑结构，针对不同的问题选择不同的抽象层次，构造问题的对象模型，展示对象和类如何组成系统(静态模型)，使该模型能够精确反映所要解决的“实质”问题。

面向对象方法中的所有对象都是通过类来描述的，其核心工作是分析和设计对象以及类，这是一个迭代的过程。类作为一个整体，贯穿从分析到实现的整个系统开发过程，但在每一个阶段其抽象层次是不同的。本书只讨论系统分析阶段所研究问题领域中有意义的概念类(概念类就是现实环境中存在的事物或发生的事件，如图书、读者就是图书管理系统的最重要的事物，借书、还书就是最重要的事件)、概念类之间的关系以及概念类的属性以及类的主要操作，其他细节及与实现有关的问题可以推迟到系统设计阶段再考虑。

3.2.2 用例建模

对于规模较大的系统，为了控制一次分析所考虑的范围，按照人们认识问题由简单到复杂的原则，也就是运用粒度控制的原则，可以采用自顶向下或自底向上的方法，把功能联系紧密的用例加以分组形成若干个主题(每一个主题就是一个局部应用)，每个主题还可以进一步划分成子主题，形成主题的层次化结构，直到底层的一个主题的功能相对独立，便于理解。对于一般的小系统而言，无须划分。

使用“用例”方法描述系统需求的过程称作用例建模。用例建模通常由用例图和用例的详细描述(用例规约)组成。用例图的模型元素包括参与者、用例、用例与参与者之间关系以及用例之间的关系，图 3-3 给出了如何在用例图中表示一个用例以及各模型元素的表示法，其中矩形框表示系统的边界。

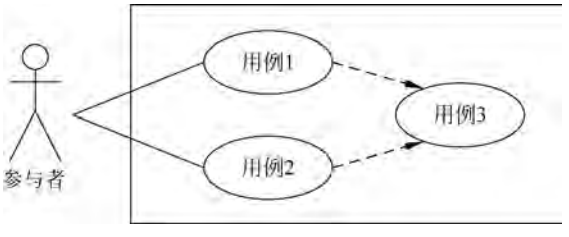

图 3-3 用例图的表示法

下面给出用例建模的主要步骤。

1. 确定系统的参与者

通俗地讲,参与者就是定义系统的使用者。对于系统来说,都存在着一些与系统打交道的事物。所谓参与者是存在于系统之外并与系统进行交互的人或其他系统。边界之内的所有人和事物都不是参与者。参与者以某种方式参与用例的执行过程,通过向系统输入或请求系统输入某些事件来触发系统的执行。执行系统某些功能的参与者可能有多个,根据他们在与系统进行交互时的不同职责,参与者可以划分为主要参与者和次要参与者、发起参与者和参加参与者。参与者由参与用例时所担当的角色来表示。

2. 确定需求用例

面向对象方法中,通过用例描述系统需求。用例是外部可见的系统功能单元,通常由系统用户来响应需求,是系统运行的活动。确定参与者后,可以根据参与者来确定系统的用例,主要是看每个参与者如何使用系统或者参与者需要系统提供什么样的服务,从而使得识别用例变得相对容易。

3. 构造用例图

识别了参与者和用例后,就可以构造用例图。用例图中除了表示参与者和用例之间的通信关联关系外,还需要表示用例和用例之间的包含(include)、扩展(extend)和泛化关系以及参与者和参与者之间的泛化关系。利用这些关系可以调整初始的用例模型,把一些公共的信息抽取出来重用,使得用例模型更易于维护。

4. 用例规约

用例图只是在总体上描述了系统所提供的服务,使我们对系统的功能有了一个整体的认识,但每个用例的细节并没有在用例图中表述出来。因此,还需对每个用例进行详细的描述,用例规约是以文档的形式详述用例,描述的是系统响应参与者的操作所依据的内部操作顺序,从而说明用例"做什么"的细节,一般采用自然语言表达,方便相关人员对需求的深入理解。目前没有一个标准的用例规约文档,但一般情况下用例规约应该包含用例名、参与者、前置条件、事件流和后置条件。

3.2.3 对象模型

尽管面向对象方法中的核心概念是对象,但对象是由它们所属的类来描述和创建的。一个类可以有很多对象实例,只要把对象所属的类描述清楚,由这个类创建的每个对象的属性、操作以及之间的关系就确定了。因此,将对象的描述方式上升为一种建模思想,就是在类的抽象层次建立以"类"为直接构造成分的系统模型,通过类以及类之间的关系描述系统中所有对象及其相互之间的关系。可以说,类是面向对象方法开发系统的基石,只有将系统的应用需求建立为对象模型(类图),系统的状态和行为才能变成可观察的。

获取用户的需求后,面向对象分析的主要工作是建立问题领域的对象模型。对象模型描述了现实环境中"类与对象"以及它们之间的关系,表示了系统的静态数据结构。因此,对象模型设计的主要任务是根据实际需求,通过分析找出对象和类,明确它们的含义和责任,确定属性和操作,并确定它们之间的关系,建立问题领域的对象模型,保证业务执行逻辑能够被对象很好地完成。

对于复杂的大系统来说,对象模型中类非常多,类之间的关系错综复杂,这样对模型的

理解和阅读都非常困难。为了控制系统的复杂性，当考虑系统的全局时，首先着眼于系统的中高层次、大粒度的概念，进行宏观思考，在考虑各部分细节时，则集中于一个局部，围绕一个主题(控制一次分析所考虑的范围)进行微观思考。具体分析时，采用自底向上的方法，首先对相关的类进行归并，建立各个局部的对象模型，再由建立的若干对象模型构成整个系统的逻辑层次结构。

对象的发现建立在获取需求的基础上，可以从用例模型中的每一个用例的事件流中获得概念类，这些概念类经过完善后将形成对象模型中的类，因此，此阶段的对象模型也称作为分析层次类图。

下面阐述构造对象模型的主要步骤。

1. 识别对象和类

对象模型中所说的对象，不是一个个具体的对象，而是一组具有相似属性和服务的抽象对象——类。对象模型中所说的类分为两种：一种是对象类，另一种是抽象类。对象类是指含有对象实例的类，抽象类是指不含对象实例的类。对象模型中所说的类基本都是对象类。一个对象类都应包含类名、属性和服务三部分，通过一个矩形框来表示，矩形框被分为三部分：上部分表示对象类名，中间部分表示对象类的属性，下部分表示该对象类提供的操作。

类与对象在所研究问题领域中是客观存在的。系统分析师的主要任务就是通过分析找出这些类和对象。实际中，首先找出所有候选的类与对象，然后从候选类与对象中筛选掉不恰当的或不必要的。

发现对象和类的常用方法有名词短语法、用例驱动法和通用类模式法。

1) 名词短语法

名词短语法依据需求陈述与用例描述中出现的名词或名词短语来提取实体对象，并将它们作为候选的概念类或属性，通过筛选明显无意义的名词或名词短语，确定最终的对象类。

2) 用例驱动法

在使用了用例模型获取了系统需求后，把用户的需求落实到不同的用例中，用例规约又描述了用例中对象同系统的交互，每个用例也可采用动态模型中的顺序图和协作图进行补充说明用例中对象出现的步骤。因此，通过用例驱动法可以归纳并发现候选的对象类。

3) 通用类模式法

通用类模式法是一种从对象的一般分类理论中发现问题域的候选类。候选对象可以从以下几个方面进行识别。

(1) 人员类。人员类是人或组织在系统中扮演的角色，而不是具体的人，如图书馆的读者、图书管理员。

(2) 组织类。在系统中发挥一定作用的组织机构，如企业中的部门、学校中的院系。

(3) 实体类。需求分析包括的可以感知的物理实体或抽象的概念实体，如图书馆的图书、学校中的课程。

(4) 事件类。需求分析所涉及的重要事件，也就是在系统中可能发生的事件或交易，如图书馆系统中的每次借书还书事件、银行系统中的客户提款事件、顾客在商场的购物事件。事件是指一个状态的改变或一个活动的发生。

(5) 业务规则或政策。系统中经常使用的业务规则或政策的文字描述。

上述各种方法都有各自的优缺点,因此,实际识别对象和类时要各种方法综合使用。

2. 确定属性

属性是对象所具有的共同性质,描述对象静态特征的一个数据项。确定属性是为了在类中存储必要的描述数据,通过属性可以对类与对象的结构有更深入、更具体的认识。

在分析阶段,不可能找到对象类的所有属性,但需要确定每个对象类的基本的、重要的属性,以后再逐渐把其余属性增添进去。属性的确定既与问题领域有关,也和系统的任务有关,需要考虑的是与具体应用之间相关的属性,而不是考虑那些超出系统边界范围的属性。当然,实际确定属性时还需要借助于一些常识才能分析出需要的属性。

1) 发现属性的方法

与发现类与对象类似,可以通过需求描述中的名词短语的方法发现属性,也可以向用户提出问题,采用交谈的方式帮助发现对象的属性。针对每一类对象提出并回答以下问题,从而启发系统分析师从各种不同的角度发现对象的属性。

(1) 该类对象有哪些一般特征?例如,图书馆的图书一般需要描述的特征有书名、作者、出版日期、单价、出版社等。

(2) 在所研究的问题领域中对象还需具备什么特定的描述?如图书,为了了解图书的库存情况,除基本特征外还需描述图书的库存数量。

(3) 该对象在系统中的职责是什么?对象的有些属性,只有在明确了对象的职责时才能决定是否需要。例如,图书管理员需要向用户提醒还书功能,或者读者还书时,图书管理员需要查看是否超期,与该职责相关的就需要定义读者类别的借阅期限、读者借阅图书的借书日期。

(4) 建立这个对象需要长期保存和管理哪些信息?随着时间的推移,问题领域中的类始终保存稳定,但对象的特性却可能改变,但从历史来看这些特性还有用途,需要长期保存。

(5) 对象可能处于什么状态?是否需要增加描述对象状态的属性。对象的状态不同,可能执行的操作也不同。例如,图书馆中的图书就有入库、在库、破损等状态。

2) 确定属性需要注意的问题

属性的确定工作往往需要反复多次才能完成,但属性的修改通常并不影响系统的结构。在确定属性时应该注意以下问题。

(1) 仅定义与系统目标和任务相关的属性。对象的特征有很多方面,但只有在所研究的问题领域中有意义的属性才予以保留。例如,读者的属性有政治面貌、体重、身高、身体状况,但在图书管理系统中为读者设置这些属性显然没有意义。

(2) 误把对象当作属性。如果某个业务实体的独立存在比它的值更重要,则应该把它作为一个对象而不是对象的属性。例如,图书所属的属性“出版社”,如果还需要强调出版社的地址、邮编、负责人、联系方式等信息,则出版社可以当作一个独立的对象,有自己的属性和服务。同一个实体在不同的应用领域中应该作为对象还是属性,需要具体分析才能确定。

(3) 误把关联类的属性当作对象的属性。如果某个性质依赖于某个关联的存在,则该性质是关联类的属性,不应该把它作为相互关联的两个对象类的属性。特别是在多对多的关联中,关联类属性非常明显。例如,借书日期和还书日期这两个属性是依赖于读者是否借阅图书,也就是说,读者对象是否与图书对象具有一个“借阅”关联实例。因此,可以创建一个关联类借阅记录,把借书日期和还书日期作为关联类“借阅记录”的属性,而不能作为读者

或图书的属性。

(4)“属性”不能再具有需要描述的性质。“属性”必须是不可分割的数据项,不能包含其他属性。也就是说,属性不能包含一个内部结构,不能是另外一些属性的聚集。例如,如果把“地址”作为出版社的属性,那么就不能试图区分省、市、街道等。

(5) 存在不一致的属性。类应该是简单而且一致的,如果得出一些看起来与其他属性毫不相干的属性,则应该考虑把该类分解为两个不同的类。

(6) 如果一个属性能从模型中其他的属性推导得到,则应该取消。

属性的表示通常使用属性字典,属性字典是所有对象类的所有属性的数据类型和数据范围的定义。

3. 确定对象间的关系

一个系统中与其他对象没有任何关系的对象是无任何意义的,对象只有在与其他对象的相互联系、相互依赖中才具有实际意义。因此,对象之间存在着一定的关系。只有建立了对象之间的各种关系,系统中所有的对象才能构成一个有机的整体。由于对象是类的实例,因此,类与类之间的关系也就是对象与对象之间的关系。根据现实世界中的事物之间实际存在的各种关系,面向对象方法在分析阶段主要有三种表现形式表示对象之间的关系:关联关系、一般和特殊关系、整体和部分关系。

1) 关联关系

关联关系描述的是类和类之间的连接,表示的是对象类的实例之间的相互依赖、相互作用的关系。关联关系建立了对象类之间的逻辑连接关系。根据关联所涉及的类的数量,可以分为一元关联、二元关联和多元关联。最常见的是两个类之间的关联,即二元关联,有时也需要定义多个类之间的关联,即多元关联。下面以二元关联为例,讨论关联的设计。

(1) 关联的表示法。类之间的连接关系在对象模型中一般使用一条连接对象类的直线(关联线)表示。二元关联的表示法如图 3-4 所示。

图 3-4　二元关联的表示法

(2) 关联的语义。关联的语义包含关联名称和关联角色。关联名称用于标记关联,建立了两个类的关联后,可以在关联线上的中部确定关联的名称;关联角色是关于关联关系中一个类对另一个类所表现出的职责,用于反映该端的对象在关联中扮演什么角色,在关联线的两端分别表明角色名称。两者不必同时使用,如果在关联上没有标出角色名,则隐含地用类的名称作为角色名。

(3) 关联的多重性。关联的多重性表示参与关联的对象实例的数量约束。确切地说,就是表示一端的多少个对象可以和另一端的一个对象产生关联。多重性可以用来表示一个取值范围、特定值、无限定的范围或一组离散值。用“..”分隔开的区间表示,其格式为 minimum..maximum,其中,minimum 和 maximum 都是整数,成对地显示在关联的尾部。多重性的值和含义如表 3-1 所示。

表 3-1 多重性值的含义

修饰符	语义	修饰符	语义
1..1 或 1	表示 1 个对象	0..* 或 *	表示 0 到多个对象
0..1	表示 0 到 1 个对象	1..*	表示 1 到多个对象

(4) 关联的类型。根据发生关联关系的类之间所涉及的对象实例的数目,可以将关联关系分为一对一关联、一对多关联和多对多关联。

一对一关联是指关联两端的数量约束都是 1,即每一端的一个对象实例都只和另一端的一个对象实例相关联,如图 3-5 所示的班级类和班长类之间的关联。

图 3-5 一对一关联示意图

一对多关联指关联两端数量约束一端是 1,另一端是 * 。数量约束是 1 的类的一个对象实例可以和数量约束是 * 的类的多个对象实例相关联,数量约束是 * 的类的一个对象实例仅和数量约束是 1 的类的一个对象实例关联,如图 3-6 所示的班级类和学生类之间的关联。

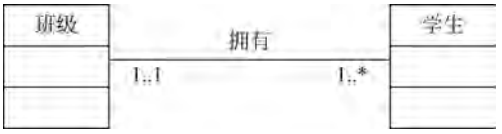

图 3-6 一对多关联示意图

多对多关联是指关联两端的数量约束都是 * ,即任何一端类的一个对象实例都可以和另一端类的多个对象实例相关联,如图 3-7 所示的学生类和课程类之间的关联。

图 3-7 多对多关联示意图

(5) 带有属性和操作的关联。在实际应用中,两个类之间的关联可能还需要描述更多的信息,即关联有可能有自己的属性或操作,对此,需要引入一个关联类来进行描述。例如,在用一个选课关联表示学生选修课程时,还需要给出选修的成绩等信息,于是可以引入“选课”关联类,成绩就是“选课”关联类的属性。

关联类附属于关联,通过其中的属性和操作来描述在关联上需要附加的更多信息。其表示方法是通过一条虚线悬挂在一个关联的连接线上,它的上、中、下三部分表示关联类的名称、属性和操作。关联类的表示如图 3-8 所示。

然而,关联类并不是面向对象方法的基本概念,在应用中,当认为一个关联上还需要增加一些信息时,运用面向对象的观点,可以分析出这些信息究竟描述了一种什么事物。把这

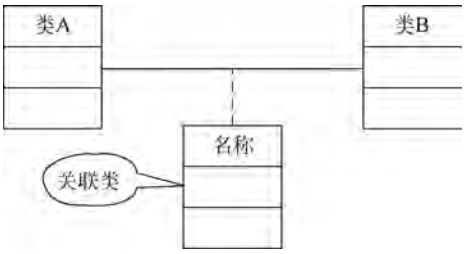

图 3-8 关联类的表示示意图

种事物抽象为对象，用类来表示，并分别建立新增加的类和原先的类之间的新的关联。也就是把关联类定义成一个普通的类来处理。图 3-7 所示学生类和课程类之间的关联选课，转换之后如图 3-9 所示。

图 3-9 关联类作为普通类表示的示意图

2）一般和特殊关系

一般和特殊关系是指一个一般类和它的特殊类之间的二元关系。在面向对象方法中，“继承”“分类”和“泛化”反映的是对象之间的一般特殊关系。一般类包含特殊类的共有属性和服务，特殊类不仅包含各自特殊的属性和服务，而且可以继承一般类共有的属性和服务。

由一组具有一般和特殊关系的类所形成的结构称为一般和特殊结构。

（1）一般和特殊关系的表示法。一般和特殊结构的表示法如图 3-10 所示。特殊类和一般类用一个带箭头实线相连，其中，末端的小三角形指向一般类，从它引出的线条可以分出数量不等的分支，分别连接到每个特殊类。

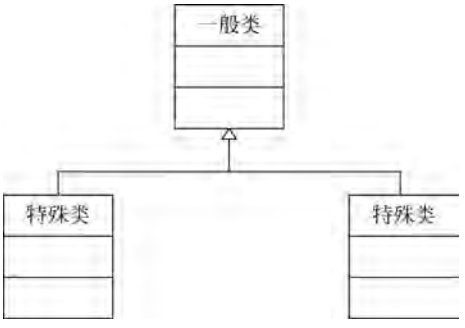

图 3-10 一般类和特殊类的表示法

（2）一般和特殊关系的语义。其语义是“is a kind of”或“is a set of”，中文含义是“…….是一种……”“……是一类……”。一般和特殊结构是所研究问题领域中各事物之间的客观存在的一种结构，建立这种结构，将使对象模型更清晰地反映问题领域中事物的“分类”关系，从而简化对系统的认识。

(3) 如何建立一般和特殊关系。可以使用两种方式建立一般和特殊关系。一是自底向上抽象出多个类含有的共同属性和操作,构成一个在概念上包含原先类的一般类,从而形成类的一般特殊结构。例如,系统中定义了“本科生”和“研究生”两个类,把它们的共性提取出来,构成一个“学生”类,则“学生”类就是一般类,而“本科生”和“研究生”就是“学生”类的两个特殊类,它们不仅可以继承“学生”类的属性,还可以有自己的属性。二是自顶向下把现有类细化成更具体的子类,也就是说,如果一个类的某些属性和操作只适合这个类的一部分对象,而不是全部对象,说明可以从这个类中划分出一些特殊类,建立一般特殊结构。同样,“学生”类如果有“导师”属性,而“导师”属性只能适合学生中的研究生。因此,应在“学生”类之下建立“研究生”这个特殊类。

一般特殊关系是利用继承机制共享性质,对系统中的类进行组织。利用这种关系,可以从系统中的类抽象出适用于整个领域的可复用类。但应避免过度细化,如果建立过深的继承层次,增加了系统的理解程度,或者一般类划分出过多的特殊类,使系统中的类太多,从而增加系统的复杂性。

3) 整体和部分关系

整体和部分关系是指一个类的某些对象实例是另一个类的某些对象实例的组成部分,这种关系在客观世界中非常常见。例如,计算机系统由主机、显示器、键盘等组成,一个大学由许多学院组成,一个社团包含指导老师和成员。在需求分析中,“组成”“包含”“是…….部分”等经常设计为这种关系。根据整体和部分联系的强弱程度,整体和部分关系可分为聚合和组成关系。

聚合泛指所有的整体部分关系,整体和部分相互独立,一个部分可以是多个整体的组成部分,整体消失,部分实例依然存在。例如,一个学生可以同时是多个社团的成员。

组合专指紧密、固定的整体部分关系,是一种联系更强的聚合关系,又称为强聚合。整体拥有各个部分,在某一时刻部分只可能是一个整体的组成部分而且部分的存在依赖于整体,随着整体的创建而创建,随着整体的消亡而消亡。例如,一篇文章有摘要、关键词、正文、参考文献组成,如果文章不存在了,就不会有组成该文章的摘要等部分。

(1) 整体和部分关系的表示法。整体和部分结构的表示法如图 3-11 所示,在整体对象类和部分对象类之间画一条连接线,把靠近整体的一端画成菱形,其中,空心菱形表示聚合关系,实心菱形表示组成关系。和关联关系类似,也可以在连接线的两端标识关系的多重性,表明关系双方对象实例的数量约束。

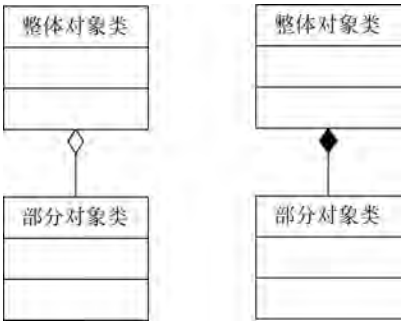

图 3-11 整体和部分关联的表示法

(2) 整体和部分关系的语义。整体和部分的语义是“is a part of”或“has a”,中文含义是“……是……一部分”“有一个”。整体部分结构是把一个复杂事物看成是由若干比较简单的事物组成,能够简化对复杂事物的描述,使系统模型更清晰地描述所研究问题领域中事物之间的组成关系。

(3) 如何建立整体和部分关系。问题域中的事物,彼此之间都有可能存在整体部分关系。实际应用中,组织结构和它的下级组织、部门可以很好地表达它们之间的整体部分关系,团体与它的成员之间也是,物理上的整体事物和它的组成部分如果都需要作为系统中的对象,也可以建立它们之间的整体和部分关系。

在系统中采用关联、一般和特殊、整体和部分的形式将不同对象连接起来,促进对象之间的合作。这些关系在对象模型中是最必要的,尤其在持久的对象模型中是最为必要的一种联系。在分析确定对象之间的关系中,不必花费过多的精力去区分关联、聚合、组合等关系,建立不同的关系只是为了使对象模型更加明确,在分析阶段使用普通的关联关系即可。况且,问题领域的主体是对象,识别对象比识别关联更为重要,总之,要识别出问题领域中有关键作用的对象间的关系,而不是现实中的全部关系。

4. 确定服务

对象的服务是直接体现系统功能性需求的成分,“确定服务”的目标是将事物的动态特征抽象为对象和类的服务。通过审查用例模型图中所描述的每一项用户功能要求和对象在问题域中具有哪些行为,可以明确各个对象应该提供哪些服务。对象类的服务分为两大类:一类称作常规性服务或辅助性服务,主要包括每个对象类都有的创建对象服务、设置对象属性值的服务、获取对象属性值的服务、删除对象服务等;另一类称作为功能性服务或需求性服务,它反映了该类对象实例所具有的特殊功能。在分析阶段,确定服务主要是确定功能性服务,即只考虑对象固有行为的操作。常规性服务通常在实现阶段才具体考虑。

3.2.4 需求分析案例

图书馆是高校图书资料的情报中心,是为教学和科研服务的学术性机构。图书馆在正常运营中总是面对大量的业务信息。假设某大学图书馆需要开发一个图书管理系统,为了简化问题,现只考虑图书管理的核心业务图书信息、读者信息以及由两者相互作用产生的借书、还书等信息。为了简化问题,没有考虑实际系统可能还有的其他功能需求,如读者登录、读者预定图书、超期罚款处理、借书卡挂失、续借等,也不考虑同一个读者多次借阅同一本书的情况(或者读者每次还书时,图书管理员清除读者的借阅信息)。

1. 系统功能需求

经过需求调查,主要实现以下功能。

1) 图书管理

图书管理员对馆内的图书按类别进行统一编号,并登记图书的主要信息,包括图书编号、书名、类别、作者、单价、出版社、出版日期和库存量等。所有图书由图书编号唯一标识。图书管理员、读者可以随时查询图书的信息。

2) 读者管理

图书管理员为读者办理借书证,建立读者信息,包括读者卡号、姓名、性别、单位、办卡日

期、卡状态和读者类别等,所有读者由借书卡号唯一标识。读者包括教师、研究生、本科生,根据读者的不同身份,读者具有不同的借阅权限,不同类别的读者一次借阅图书册数、借阅期限、借阅超期时的罚款金额不同。图书管理员可以随时查询读者的信息,读者也可以查询自己的信息。

3) 借书管理

系统能够判断读者的借阅证是否有效,能够记录读者的借书信息、借阅日期等流通信息。读者借书时,图书管理员能够查看该读者的借书卡,查询读者的借阅信息,统计读者已借书的数量及是否有超期的图书等,如果没有借书超量或超期的情况,办理借书手续。读者也可以查询自己的借阅情况。

4) 还书管理

读者还书时,图书管理员判断图书的合规性(如图书是否破损)、查询流通记录、修改流通状态等流通记录,并自动填写还书日期。如果读者未超期,则删除读者的借阅记录;如果读者超期,则按超期天数给出读者超期罚款金额。

2. 系统用例模型

1) 确定参与者和用例

参与者代表的是使用者在与系统交互时所扮演的角色,而不是某个具体用户,根据参与者的定义和参与者的确定方法(参考信息系统分析与设计、UML 相关课程知识),可以识别出系统最重要的参与者有读者、图书管理员。

实践表明,通过参与者来识别用例是非常有用的,面对一个大系统,要列出用例清单常常非常困难,而首先列出参与者清单,再对每个参与者列出它的用例,从而使问题变得容易。对读者来说,主要系统用例有查询图书、查询读者(本人)信息、查询借阅情况;对图书管理员来说,主要系统用例有办理借书证、借书处理、还书处理、查询图书信息、查询读者信息、查询借阅信息、统计馆藏图书。

2) 建立用例图

识别了参与者和用例,并确定了它们之间的关系后,就可以构造系统的用例图。用例图是描述参与者和用例之间关系的图形。在 UML 中,用类似小人符号表示参与者,用椭圆表示用例,用矩形框表示系统边界。系统用例图如图 3-12 所示。

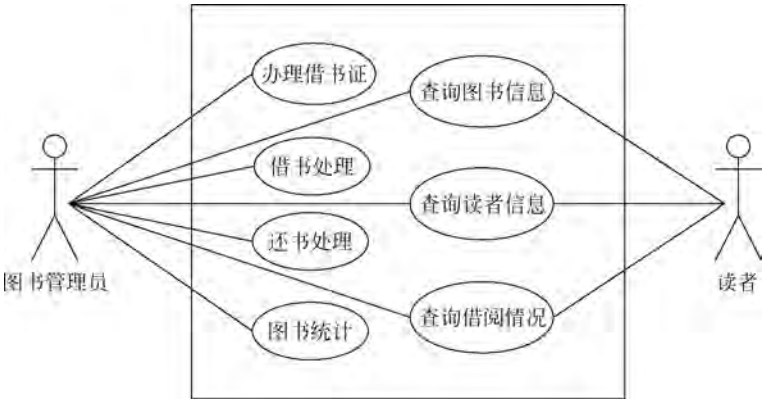

图 3-12 图书管理系统用例图

3）用例规约

用例规约是以文档的形式详细描述用例，描述用例也称为用例场景，即进行的业务事件以及如何同系统进行交互以完成任务的文字描述。没有描述的用例就像一本书的目录，只知道目录标题，并不知道这些目录标题对应的内容。以“借书处理”用例为例，说明用例规约的书写，如表 3-2 所示。

表 3-2 “借书处理”用例说明

用例名称	借书处理	
用例描述	图书管理员将读者选择的图书进行借书处理	
参与者	图书管理员（主要参与者），读者（次要参与者）	
前置条件	图书管理员已经登录系统，并被授权	
后置条件	存储借书记录，更新图书库存数量	
主要事件流	参与者动作	系统响应
	（1）图书管理员将读者借书卡提供给系统	（2）系统验证读者类别身份和借书条件
	（3）图书管理员将读者所借图书输入系统	（4）系统记录读者借书记录，并修改图书的库存数量、累加读者的借书数量
	（5）重复（3）～（4），直到图书管理员确认全部图书登记完毕	（6）用例结束
备选事件流	备选流 1：非法读者，系统显示错误，用例结束 备选流 2：系统验证读者借书数量已达上限，系统显示错误，并拒绝借阅，用例结束	
业务实体	图书管理员、读者、图书、借阅单	

3. 系统对象模型

1）识别系统的对象和类

考虑到借阅过程中，图书管理员只是一个执行者，为了简化问题，在对象模型中，不考虑图书管理员。根据用例模型和确定类的方法，可以得出高校图书管理系统所涉及的类有读者类、图书类、读者类别类、借阅单类。其中，没有把读者的“借书卡”抽象为类，因为“借书卡”对象中除了“借书卡号”属性外，其余属性都与“读者”对象相同，因此，在“读者”类中增加“读者卡号”属性；也没有把“读者”类抽象为一般类，把“教师”类、“研究生”类和“本科生”类作为“读者”类的三个特殊类。如果把读者、教师、本科生、研究生抽象为四大类，并建立一般和特殊的关系结构，虽然一般类和特殊类之间在概念上是不同的，但在图书管理系统中，对教师、研究生、本科生这三类人员的管理没有什么不同，只是借书册数和借书期限不同，从系统功能的角度分析也没有必要做这样的区分，又由于教师、本科生、研究生这三类人员的可借阅天数、可借阅数量、超期罚款额不同，因此，用“读者类别”和“读者”这两个类就足够了。“借阅单”类是读者和图书类之间的关联，由于此关联需要描述借书日期、还书日期等特性，因此，把借阅关联作为一个关联类或一个普通类“借阅单”来处理。

2）确定属性

根据需求描述可以获取各个对象类的属性如下。

读者类别的属性：类别编号，类别名称，可借阅天数，可借阅数量，超期罚款额；

读者的属性：读者卡号，姓名，性别，单位，办卡日期，卡状态；

图书的属性：图书编号，书名，类别，作者，出版社，出版日期，单价，库存数量；

借阅单的属性：借书日期，还书日期。

3）确定关系

两个类之间需不需要建立关联，完全取决于用户的业务需要，根据图书管理系统的实际业务需求，建立了读者类别和读者类之间的一对多的“拥有”关联，读者和图书之间建立了多对多的“借阅”关联，把“借阅”关联转换为一个普通类“借阅单”，然后建立了“读者”类和“借阅单”之间的一对多关联和“图书”类和“借阅单”类之间的一对多关联。

4）确定服务(操作)

通过分析对象在问题域中所呈现的行为以及对象所履行的系统责任来发现和定义对象的每个操作。对象提供的操作应尽可能准确地反映该操作所提供的功能，从各种不同的角度尽可能把所有可能的操作找到，然后确定哪些操作是真正有用的。通过分析高校图书管理系统各对象的主要操作如下。

读者对象主要操作：办理借书证，查询；

图书对象主要操作：入库，查询，统计；

读者类别对象主要操作：查询；

借阅单对象主要操作：借书，还书，查询。

通过以上分析，确定的高校图书管理系统的对象模型如图 3-13 所示。

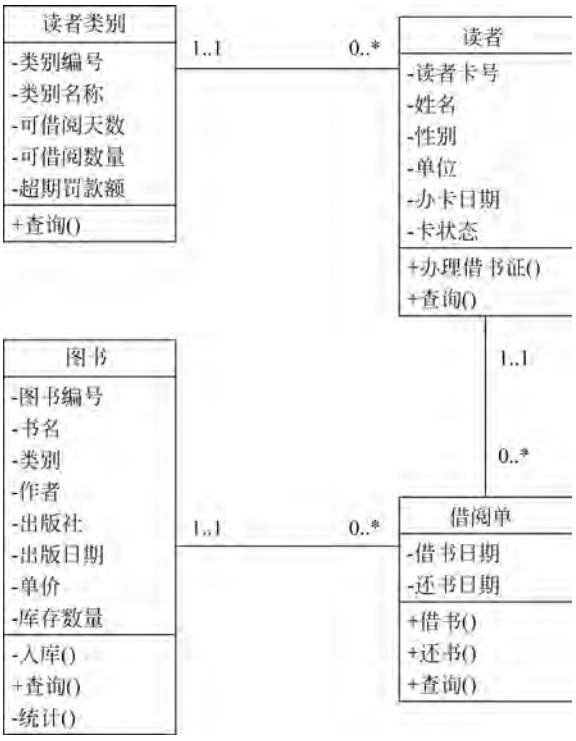

图 3-13　图书管理系统类图

5）属性字典和服务说明

属性字典是说明对象模型属性的主要工具。通过属性字典可以对对象模型中所有对象类的所有属性的数据类型和数据范围定义。表 3-3 给出了图 3-13 所示的高校图书管理系

统对象模型的部分属性字典。在此省略对所有对象类的服务说明。

表 3-3 属性字典

属 性	类	定义/范围	备 注
类别编号	读者类别	2 个字符	
类别名称		10 个字符	
可借阅天数		1B	
可借阅数量		1B	
超期罚款额		4B	
读者卡号	读者	10 个字符	
姓名		16 个字符	
性别		1 个字符	男、女
单位		30 个字符	
办卡日期		3B	yyyy-mm-dd
卡状态		5 个字符	
图书编号	图书	8 个字符	
书名		40 个字符	
类别		16 个字符	
作者		16 个字符	
出版社		20 个字符	
借书日期	借阅单	3B	yyyy-mm-dd
还书日期	借阅单	3B	yyyy-mm-dd

3.3 数据库概念结构设计

数据库的概念结构设计是将系统分析得到的用户需求抽象为反映用户观点的信息结构的过程，也就是将现实事物以不依赖于任何数据模型的方式加以描述，目的在于以符号的形式正确地反映现实事物及事物与事物间的联系。设计的结果是数据库的概念模型，即 E-R 模型。由于它是从现实世界的角度进行抽象和描述，所以，它与计算机硬件、数据库逻辑结构和支持数据库的 DBMS 无关。在数据库设计中应重视概念结构设计，它是整个数据库设计的关键，是为计算机存储数据做准备工作。

3.3.1 概念结构设计概述

只有将系统的应用需求抽象为信息世界的结构，也就是概念结构后，才能转换为机器世界的数据模型，并用 DBMS 实现这些需求。概念结构设计的目标在用例图和类图的基础上，产生反映企业组织信息需求的数据库概念结构，即概念模型。

1. 概念数据模型

概念模型最常用的表示方法就是实体联系(E-R)方法，其设计的内容就是如何根据应用需求设计系统的 E-R 模型。

1) 实体联系方法

实体联系方法采用了实体集、联系集和属性三个基本概念分别描述现实世界中事物、联

系及其特征。由于此方法简单、实用,得到了非常普遍的应用,是目前描述信息结构最常用的方法。该方法使用的工具称作 E-R 图,E-R 图描述的结果称为 E-R 模型或概念模型。

E-R 模型是一种非常广泛的数据建模工具,它通过将现实世界中人们所关心的事物及其联系建模为实体、实体的属性和实体之间的联系,并通过 E-R 图进行描述,具有很强的语义表达能力。

目前还没有具体的数据库管理系统支持 E-R 模型,但有支持 E-R 模型的数据库设计工具,这种设计工具可以把 E-R 模型直接转换为具体的数据库管理系统上的数据模型,并生成建立数据库的目标代码,甚至可以直接建立数据库。如 Computer Association 公司的 Erwin 工具、Sybase 公司的 Power Designer 工具等。

2) 概念模型的表示

E-R 模型为数据库建模提供了三个基本的语义概念:实体集、属性和联系的表示方法。

(1) 实体集的表示。用长方形表示实体集,长方形内标明实体集名。

(2) 属性的表示。用椭圆形表示实体集的属性,并用线段将其与相应的实体集连接起来。属性下的下画线表示此属性为该实体的主标识符。例如,读者具有读者卡号、姓名、性别、单位四个属性,其实体属性如图 3-14 所示。

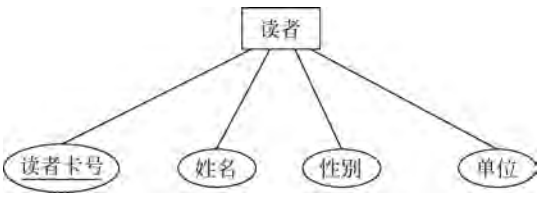

图 3-14 读者实体及属性

(3) 联系的表示。联系指实体集之间存在的相互关联关系,用菱形表示实体集之间的联系,菱形内写明联系名,并用线段分别与有关实体集连接起来,同时在线段旁标出联系的类型。

3) 实体集之间联系的类型

为了建立现实世界的完整模型,常常需要对联系分类,根据一个实体集中的实体可以和多少个另一类实体集中的实体相联系,可将联系分为如下几种。

(1) 二元联系。二元联系是指两个实体集之间的联系。两个实体集之间的关联关系可以概括为一对一、一对多、多对多三种。

① 一对一联系(1∶1):两个实体集 A 和 B,对于实体集 A 中的每一个实体,在实体集 B 中至多有一个(也可以没有)实体与之联系;反之,对于实体集 B 中的每一个实体,实体集 A 也至多有一个实体与之联系,则实体集 A 与实体集 B 之间的联系是一对一的联系,可写作 1∶1,如图 3-15(a)所示。

② 一对多联系(1∶n):两个实体集 A 和 B,对于实体集 A 的每一个实体,实体集 B 中有一个或多个实体与之联系,而实体集 B 中的每一个实体,实体集 A 中至多有一个实体与之联系,则实体集 A 与实体集 B 之间的联系是一对多的联系,可写作 1∶n,如图 3-15(b)所示。

③ 多对多联系(m∶n):两个实体集 A 和 B,对于实体集 A 的每一个实体,实体集 B

中有任意多个实体与之联系；而对于实体集 B 中的每一个实体，实体集 A 中也有任意多个实体与之联系，则实体集 A 与实体集 B 之间的联系是多对多联系，可写作 m：n，如图 3-15(c)所示。

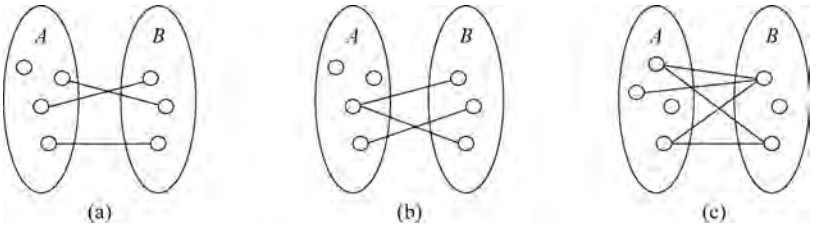

图 3-15　二元联系的类型

例如，学校里的单位、单位负责人两个实体，其语义为一个单位有一名单位负责人，而一名负责人只能管理一个单位的工作，则单位与负责人之间具有一对一的联系，E-R 图如图 3-16(a)所示。同样，针对班级和学生两个实体，其语义为一个班级有多名学生，而一名学生只能属于一个班级，则班级和学生之间具有一对多的联系，E-R 图如图 3-16(b)所示。一名学生可以选修多门课程，而一门课程可以被多名学生选修，则学生和课程两个实体之间具有多对多的联系，E-R 图如图 3-16(c)所示。

需要说明的是，联系也可以拥有属性，联系上的属性因联系发生而需要记录、存储的信息，联系和它所有的属性构成了联系的一个完整描述。因此，联系与属性间也有关系。例如，学生和课程之间的选课联系具有一个属性“成绩”。在 E-R 图中如果联系具有属性，则该属性也用椭圆表示，仍需要用线段将属性与其联系连接起来。联系的属性必须在 E-R 图上标出，如图 3-16(c)所示。

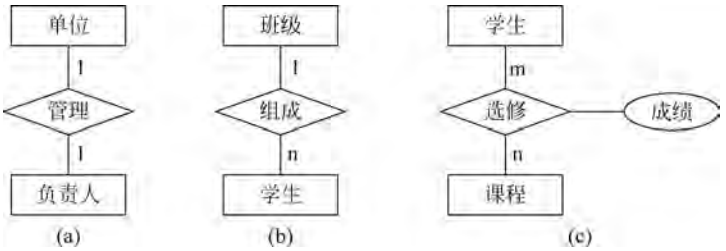

图 3-16　联系的例子

实际上，一对一联系是一对多联系的特例，而一对多联系又是多对多联系的特例。

需要注意的是，实体之间的联系类型并不取决于实体本身，而是取决于现实世界的管理方法，或者说取决于实际语义。同样两个实体，如果有不同的语义，则可以得到不同的联系类型，也就是说，给定的各实体之间可以有多种不同的联系，即多个不同的联系集可以定义在一些相同的实体集上，称之为实体之间的多联系。因此，有时在 E-R 图中，根据具体应用环境，两个实体集合之间的联系可以不只一个。例如，教师、学生两个实体集之间不仅存在授课联系集，也存在指导联系集(指导硕士研究生)，E-R 图如图 3-17 所示。

(2) 多元联系。多元联系指多个实体集之间的联系，即两个以上的实体集之间存在的联系，其联系类型为一对一、一对多和多对多三种。

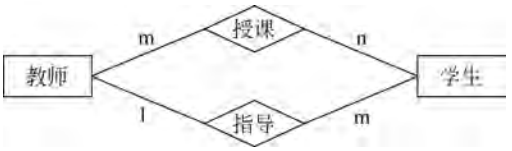

图 3-17　两个实体集之间的不同联系

数据库系统中大多数联系都是二元的,也会涉及三个及三个以上实体之间的联系。例如,有三个实体:客户、订单和图书,其语义为:一个客户可以订购多个订单,一个订单可以包含多本图书,而一个订单只能属于一个客户,可以包含多本图书。每个客户订购都有订购日期。因此,对客户、订单和图书三个实体来说,就是多个实体之间的一对多联系,联系命名为订购。E-R 图如图 3-18(a)所示。

再如供应商、项目与零件三个实体,其语义为:每个供应商可以向多个项目供应多种零件,每个项目可以使用多个供应商供应的多种零件,每种零件可以由多个供应商供应给不同的项目,每个供应商向项目供应零件应提供供应数量。则供应商、项目与零件之间的联系就为三个实体之间的多对多联系,联系命名为供应,E-R 图如图 3-18(b)所示。

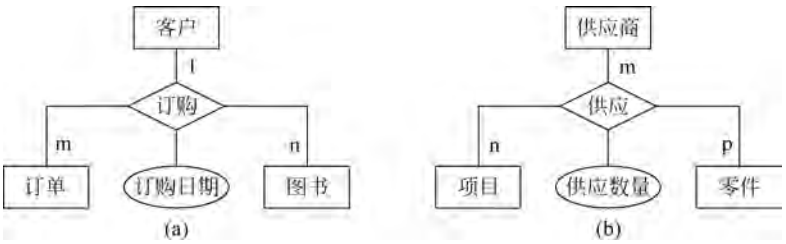

图 3-18　多元联系

(3) 递归联系。递归联系指实体集内部的联系,通常指用于组成实体的各元组之间的联系,也称为自身联系。实际上,在一个实体集的实体之间也可以存在一对多或多对多的联系。例如,学生是一个实体集,学生中有班长,而班长自身也是学生。学生实体集内部具有领导与被领导的联系,即某一个班长领导若干名学生,而一个学生仅被一个班长所管理,这种联系是一对多的递归联系,E-R 关系如图 3-19 所示。

图 3-19　同一实体集内部的一对多联

2. 概念结构设计的作用

在早期的数据库设计中,需求分析后,直接设计 DBMS 支持的关系模式,这样,注意力往往被牵扯到更多的细节限制方面,不能集中在最重要的信息组织结构和处理上。当外界环境发生变化时,设计结果就难以适应变化。

将概念结构从数据库设计中独立出来,对数据库设计人员来说,仅从用户的角度看待数据并处理需求和约束,可以使数据库设计各阶段的任务相对单一化,从而可以控制设计的复

杂性，便于组织管理。其作用主要体现在以下几个方面。

1）能真实地描述现实世界

概念结构能真实地反映现实世界事物和事物之间的联系，能满足用户对数据的处理要求，是现实世界的一个真实模型。

2）易于理解

用 E-R 图来描述概念模型非常接近人的思维，是对现实世界的真实反映，容易被人们所理解。用户不需要掌握计算机等专业知识，也能够理解概念结构，利用概念结构，用户可以与设计者交换意见，参与到数据库的设计过程中。因此，概念结构是数据库设计人员与用户交互的最有效的工具。

3）是各种数据模型的共同基础

概念模型最终要转换成不同 DBMS 所支持的数据模型，并且很容易就能向普遍使用的关系模型转换，所以是各种数据模型的基础。

（4）有利于修改和扩充

由于现实世界的应用环境和应用要求会发生变化，发生变化时只需要改变概念模型，易于更改的概念模型更有利于扩充。

3. 概念结构设计的方法

概念模型是数据模型的前身，它比数据模型更独立于机器、更抽象，也更加稳定。概念结构设计的方法有如下四种。

1）自顶向下的设计方法

首先定义全局概念结构的框架，然后逐步细化为完整的全局概念结构。

2）自底向上的设计方法

首先定义各局部应用的概念结构，然后将它们集成起来，得到全局概念结构。

3）逐步扩张的设计方法

首先定义最重要的核心概念结构，然后向外扩充，生成其他概念结构，直至完成总体概念结构。

4）混合策略设计的方法

采用自顶向下与自底向上相结合的方法。混合策略设计的方法首先用自顶向下策略设计一个全局概念结构的框架，然后以它为主干，通过自底向上策略设计各局部概念结构。

最常采用的策略是自底向上的方法，即先自顶向下地进行需求分析，然后再自底向上地设计概念结构。

3.3.2 概念结构设计的任务

概念结构设计是根据分析阶段所得到的对象模型形成信息世界的实体、属性和实体标识符，确定实体之间的联系类型。在设计时，其注意力主要集中在怎样表达用户对信息的需求上，暂不考虑如何实现的问题。概念结构设计的一般步骤如下。

（1）站在全局的角度，设计面向全局应用的整个系统的总体初步框架。

（2）根据需求分析划分的局部应用，设计局部 E-R 图。

（3）将局部 E-R 图合并，消除冗余和可能的矛盾，得到系统的全局 E-R 图，完成概念模型的设计。

(4) 审核和验证全局 E-R 图。

1. 局部 E-R 图设计

局部应用的信息需求是构造全局概念模式的基础。因此,需要从单个应用的需求出发,为每个数据的观点与使用方式相似的用户建立一个相应的局部概念结构。设计局部 E-R 图的具体步骤如下。

1) 划分用户组

划分用户组是设计分 E-R 图的前提,其实质是确定局部范围,一般将数据要求和处理要求接近的用户分成一组,同时,为了控制局部 E-R 图的复杂性,需要考虑用户组的规模,一般实体个数宜为 5～9。

2) 确定实体、属性

实体和属性,只能根据对客观世界的理解和思维习惯与数据的逻辑关系来划分。实际上,实体和属性之间并不存在形式上可以截然划分的界限。但是,为了简化 E-R 图,区分实体属性时,应当遵循的一条原则是:"现实世界的事物能作为属性对待的尽量作为属性对待。"在给定的应用环境中,可遵循如下的基本准则来划分。

(1) 属性不能再具有需要描述的性质。"属性"必须是不可分割的数据项,不能包含其他属性。也就是说,属性不能是另外一些属性的聚集。

(2) 属性不能与其他实体具有联系。在基本的 E-R 建模中,所有联系必都须是实体间的联系,而不能有属性与实体之间的联系。

(3) 对于同一个对象,如果需要进一步描述该对象,并需要处理该对象与其他实体间的联系,可以考虑将该对象作为实体;如果对象只是用来描述另一个实体,则将该对象抽象为属性。

例如,图书和出版社可以建模为两种形式:第一种形式中,出版社作为图书的一个属性存在,图书是一个实体,其属性包括图书编号、书名、作者和出版社;在第二种形式中,出版社作为一个单独的实体存在。其主要区别在于将出版社作为实体来建模可以描述关于出版社的额外信息,如出版社名称、地址、联系人等,这样比将其建模为一个属性更具有通用性。如果强调这种通用性,那么,将出版社作为实体就是合适的方式。如果不需要考虑出版社的其他信息,就没有必要把出版社作为一个实体对待,则出版社可以作为图书实体的一个属性对待,如图 3-20 所示。

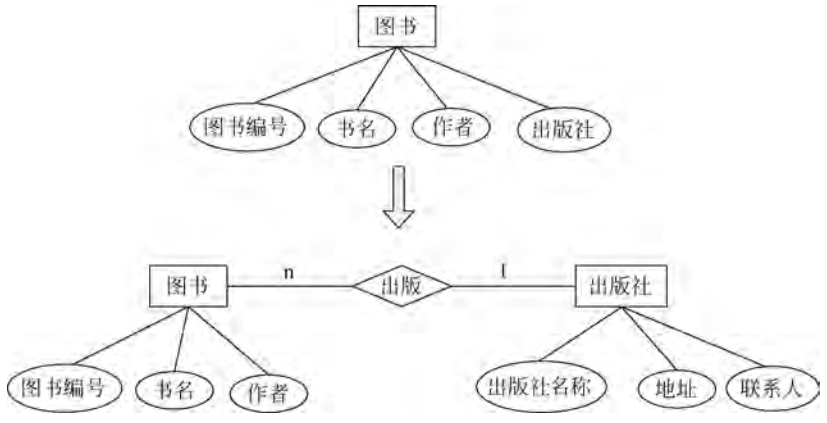

图 3-20 "出版社"由属性上升为实体的示意图

又如，在医院中，一个病人只能住在一个病房，病房号可以作为病人实体的一个属性。但如果病房还要与医生实体发生联系，即一个医生负责几个病房的病人工作，则根据第(2)条准则，病房应作为一个实体，如图 3-21 所示。

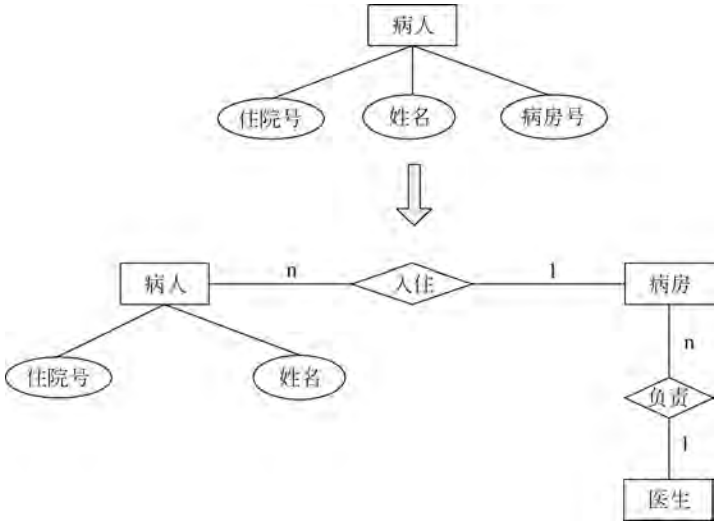

图 3-21　与“医生”产生联系的“病房”由属性转换为实体示意图

选择实体还是属性，应避免出现以下两个错误。

(1) 将一个实体的主标识符作为另一个实体的属性，而不是使用联系。

(2) 将相关实体的主标识符作为联系的属性。

3) 确定实体之间的联系、属性及其联系类型

不仅实体具有属性，而且联系也可以有属性。在确定实体和属性的同时，要通过分析确定实体之间的联系以及联系的属性，并根据语义确定联系的类型。例如，教师和课程两个实体具有语义：一个教师可以讲授多门课程，一门课程只能由一名教师讲授，则教师和课程的联系类型为 1∶n；如果语义是一个教师可以讲授多门课程，一门课程可以被多名教师以不同的开课班讲授，则两者的联系类型为 m∶n。

如果 E-R 图比较复杂，为了使 E-R 图简洁明了，常将图中实体的属性省略，而着重反映实体之间的联系。

4) 设计局部 E-R 图

在确定了实体、联系以及它们的属性之后，各个局部 E-R 图的设计就水到渠成了。

2. 全局 E-R 图设计

一个局部 E-R 图只反映了部分应用，面向的是部分用户的观点，并且各局部 E-R 图全局 E-R 图的设计就是把设计好的各局部 E-R 图综合(合并)成一个系统的总 E-R 图。合并时可以有两种方法：一种方法是多个局部 E-R 图一次集成；另一种方法是逐步集成，用累加的方法每次集成两个局部 E-R 图。无论采用哪种方法，在每次集成局部 E-R 图时，都要分两步进行。

1) 合并局部 E-R 图，生成初步全局 E-R 图

各局部 E-R 图进行合并时，要解决各局部 E-R 图之间的冲突问题，并将各局部 E-R 图

合并起来生成初步全局 E-R 图。

由于各个局部应用所面向的问题是不同的,而且通常是由不同的设计人员进行不同的视图设计,这样就会导致各个局部 E-R 图之间必定会存在许多不一致的地方,即产生冲突问题。如果各个局部 E-R 图存在冲突,就不能简单地把它们画到一起,必须先消除各个局部 E-R 图之间的不一致,形成一个能被全系统所有用户共同理解和接受的统一的概念模型,再进行合并。

各局部 E-R 图之间的冲突主要有三类:属性冲突、命名冲突和结构冲突。

(1) 属性冲突。属性冲突包括域冲突和属性取值单位冲突两种情况。

属性值的类型、取值范围或取值集合的不同都属于属性域冲突。例如,在大学里的教务管理系统、毕业设计管理系统、学籍管理系统等不同的应用中,对于学生的属性学号,可能会采用不同的编码形式,而且定义的类型各不相同,有的定义为整型,有的定义为字符型,这都需要各个应用或部门之间协商解决。

属性取值单位冲突,也就是同一属性在不同的应用中采用不同的度量单位,这样将会给数据统计造成错误。

(2) 命名冲突。命名冲突主要有同名异义冲突和异名同义冲突两种。

同名异义冲突是指不同意义的对象在不同的局部应用中具有相同的名字,即两个同名的实体、属性在不同的局部 E-R 图中含义是不同的。

异名同义冲突是指意义相同的对象在不同的局部应用中有不同的名字,即在不同的局部 E-R 图中含义相同的实体、属性,其命名不同。

命名冲突主要通过协商和调整解决。

(3) 结构冲突。结构冲突有以下三种情况。

① 同一对象在不同的应用中具有不同的抽象。例如,高校的人事管理局部应用中,工资可能作为教职工的一个属性对待,而在财务管理局部应用中,为了描述工资各方面的细节,如基本工资、补贴、实发工资等特性,则把工资作为实体对待。这就是抽象冲突。

② 同一实体在不同局部 E-R 图中的属性组成不一致,即所包含的属性个数和属性排列次序不完全相同,解决这类冲突的方法是使该实体的属性取各局部 E-R 图中属性的并集,再适当调整属性的次序,兼顾到各种应用。

③ 实体之间的联系在不同的局部 E-R 图中联系的名称、属性不同或呈现不同的类型,此类冲突解决方法是根据应用的语义对联系名称进行统一、对属性进行合并、对联系的类型进行综合或调整。

例如,对于高校,在学籍管理系统的局部应用中,学生的属性设计为学号、姓名、性别、出生日期、籍贯等,其实体属性图如图 3-22(a)所示;而在学校医院管理系统的局部应用中,学生的属性设计为学生学号、学生姓名、性别、年龄、身高、健康状态等,其实体属性如图 3-22(b)所示。

由图 3-22 可以看出,学生实体属性存在异名同义的命名冲突和结构冲突。异名同义的冲突是指在学籍管理系统中的学生的主标识符取名为学号,而在学校医院管理系统中学生的主标识符取名为学生学号,但两者描述的含义相同,需统一命名为学号。同样,姓名和学生姓名也属于异名同义的冲突,统一取名为姓名。在图 3-22(a)中学生属性有 5 个,

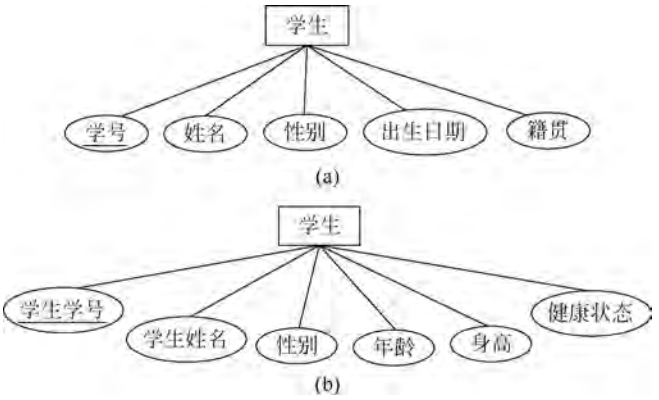

图 3-22　学生实体属性图

图 3-22(b)学生属性有 6 个，显然，因为属性个数不相同而存在结构冲突，应将学生实体的属性进行合并。经分析、调整消除冲突后，得到的学生实体属性图如图 3-23 所示。

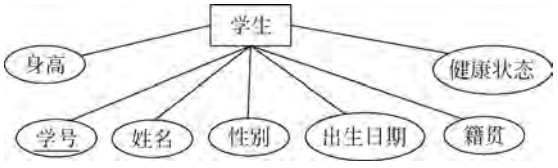

图 3-23　调整后的学生实体属性图

2）优化初步 E-R 图

初步 E-R 图优化指根据数据库应用系统的需求，在初步确定的全局 E-R 图的基础上，利用需求分析的结果，通过检测全局 E-R 图的数据冗余、联系冗余，从而消除相应的冗余数据，并最终形成独立于具体 DBMS 的整体概念结构的过程。

优化的目标是在全面准确地反映用户需求的基础上，使得系统尽量满足以下要求。

（1）属性尽可能少，即组成每一个实体的属性的个数尽量少。

（2）实体尽量少，即组成概念结构的实体的个数尽量少。

（3）联系尽量少，即组成概念结构的联系的个数尽量少。

优化指修改和重构初步 E-R 图，消除冲突的初步全局 E-R 图中可能存在冗余数据和实体间冗余的联系。冗余数据主要包括冗余属性和冗余实体。所谓冗余属性是指重复存在或可由基本数据导出的数据。例如，学生的出生日期和年龄都作为学生实体的属性，年龄就可以通过学生的出生日期获得，因此，优化时只保留其一；商品实体有单价、数量、总价属性，但根据分析，总价属性可由单价和数量属性导出，故应消除。冗余实体是指使用一个实体代替两个或多个实体（即合并实体），或者在极端的情况，可以利用多个实体导出的实体，对于合并实体，一般是指两个实体或多个实体具有相同的主标识符，则可以合并为一个实体，对于可由多个实体导出的实体，则可以直接消除。冗余的联系可由其他联系导出。

总之，冗余的存在容易破坏数据库的完整性，给数据库维护增加困难，应当消除。消除冗余的方法可以借助需求分析的结果采用分析方法，也可以用第 4 章的规范化理论。其中分析方法是消除冗余的主要方法，消除了冗余的初步 E-R 图就称为基本的 E-R 图。

在实际应用中,并不是要将所有的冗余数据与冗余联系都消除。有时为了提高数据查询效率、减少数据存取次数,在数据库时专门设计了一些数据冗余或联系冗余。因而在设计数据库结构时,冗余数据的消除或存在,需要根据用户的整体需要来确定。如果希望存在某些冗余,则应在数据字典中进行说明,并把保持冗余数据的一致作为完整性约束条件。

重组 E-R 图是指对于消除数据冗余后的全局 E-R 图,由于消除了不必要的冗余属性、冗余实体和冗余联系,因此需要根据应用系统的整体需求,再对全局 E-R 图进行整体统一的调整、重新组合和重新构造,从而形成优化的全局 E-R 图,即系统的整体概念结构。

3. 验证全局 E-R 图

经过优化的概念结构只有通过相关人员必要的审核和验证、确保无误后,才能作为下一阶段数据库逻辑结构设计的依据。验证全局 E-R 图时,确保满足下列条件。

(1) 全局 E-R 图必须具有一致性,不存在相互矛盾的表达。

(2) 全局 E-R 图能准确地反映原来每个局部应用的局部 E-R 图结构,包括实体、属性和联系。

(3) 能够满足需求分析阶段所确定的所有需求。

(4) 全局 E-R 图必须征求用户和相关人员的意见,需要经过评审、修改、优化,再确定作为数据库的概念结构,提交给用户。

综上所述,设计概念结构时,需要根据系统的应用需求和用户的最终要求,选择局部概念结构的范围,设计局部 E-R 图,合并局部 E-R 图和消除全局 E-R 图不必要的冗余,并将设计的结果向用户进行演示和解释,听取用户的意见,检查由此设计的数据库是否提供了用户所需要的全部信息。只有经过反复评审、修改和优化,才能保证所设计的概念数据模型是合理的。


视频讲解

3.3.3 概念结构设计案例

根据需求分析,利用数据库概念结构设计的方法,可以得出高校图书管理系统的数据库的各实体属性图如图 3-24～图 3-26 所示,E-R 模型如图 3-27 所示。

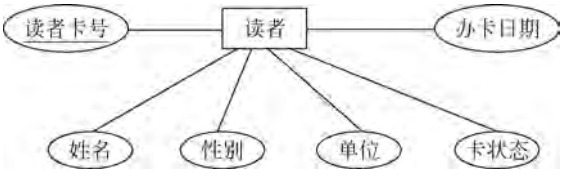

图 3-24 读者实体属性图

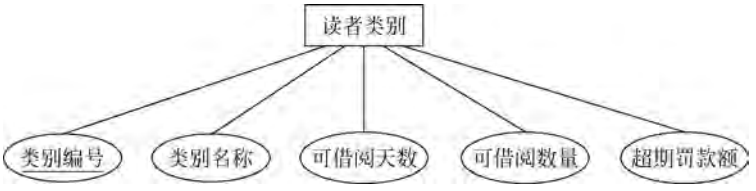

图 3-25 读者类别实体属性图

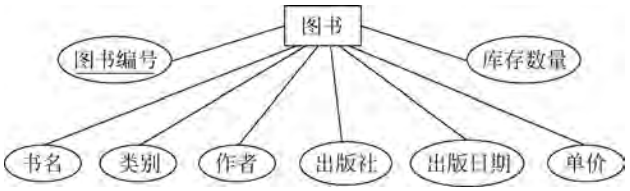

图 3-26 图书实体属性图

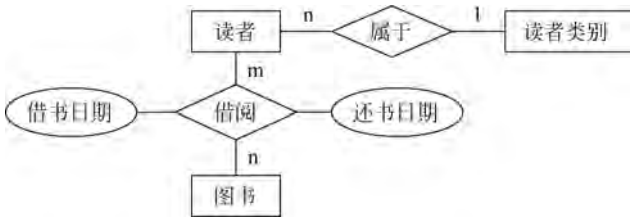

图 3-27 高校图书管理系统实体联系图

3.3.4 概念结构设计的其他问题

3.3.1 节讨论概念模型时，介绍了 E-R 方法及其基本内容。虽然已经满足数据库概念结构设计的基本要求，但在解决实际问题时，E-R 模型一些扩充的特征可以更恰当地反映数据库设计的需求。为此，对 E-R 模型中所涉及的其他一些概念做必要的补充。

1. 弱实体集与强实体集

实体是客观存在并可相互区分的客观事物或抽象事件。其中，可区分强调实体的唯一标识特征，即通过实体的标识符来区分不同的实体。在现实世界中，还存在一类实体，不能独立存在，其属性不足以标识实体特征，必须依赖于其他实体的存在而存在，这样的实体集合称为弱实体集(依赖实体集)。被依赖的实体集称为强实体集(主实体集)，强实体集与其弱实体集之间的联系称为弱联系集。例如，如图 3-28 所示的大学选课系统中，“课程”实体集和“开课班”实体集的联系，由于一学期一门课程选修的人数比较多，需要一次同时开设多个“开课班”供学生选修，则“开课班”实体集的存在是依赖于“课程”实体集的，在“开课班”实体集中，假设“开课班号”的值是某门课程所开设的开课班的序号，则不同课程的“开课班号”可能有相同的值(当然，也可以假设“开课班”的“开课班号”的值是全局唯一的，则“开课班”就是强实体集)。因此“，开课班号”就不能唯一标识“开课班”中的不同实体，而“开课班”实体集中的开课学年、开课学期、开课时间、开课地点等属性(假设一个开课班只有一个上课时间和上课地点)也不能唯一标识。因此，“课程”是强实体集，“开课班”是弱实体集。

既然弱实体集属性不足以标识自身，那么，如何标识弱实体集呢？由于弱实体集是依赖于强实体集而存在，能否考虑用所依赖的强实体集的主标识符来标识其中的实体呢？假设每门课程的“开课班号”可用来区分属于该门课程的不同“开课班”，即每门课程不存在两个具有相同“开课班号”的“开课班”，即可以使用“课程”强实体集中的主标识符“课程号”与“开课班号”结合来唯一标识弱实体集“开课班”。因此，弱实体集的标识是由强实体集中的主标识符与其自身的一个属性(部分标识符)共同构成。其中，弱实体的部分标识符使用虚下画线表示。

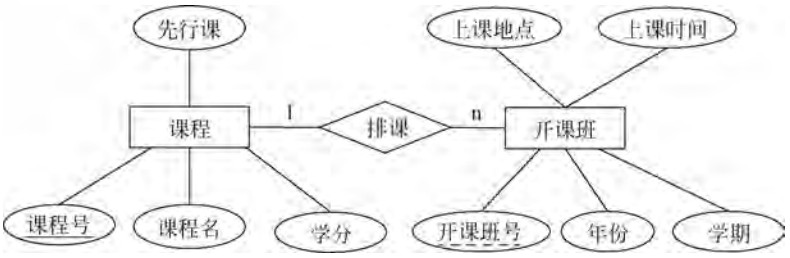

图 3-28 “开课班”弱实体集依赖于“课程”强实体集

对于弱实体集,必须满足以下限制。

(1) 强实体集和弱实体集的联系类型只能是一对多或一对一的联系。

(2) 弱实体集中的每个实体都参与到联系集中至少一个联系中。

2. 依赖实体集

对于图书管理系统的借阅业务,伴随着读者借阅图书业务的发生,会产生借阅单。如果将“借阅单”作为实体集对待,则“借阅单”实体集和“图书”实体集之间会存在多对多的“图书借阅”联系集,联系的属性包括借阅数量等,如图 3-29 所示。因此,“借阅单”实体集的存在是依赖于“图书借阅”联系集的存在。也就是说,没有“图书借阅”联系,就没有“借阅单”实体,即“借阅单”实体集与“图书借阅”联系集之间存在依赖约束,“借阅单”是依赖实体集。因此,所谓依赖实体集是指联系中一种实体的存在依赖于该联系集中联系的存在,将依赖于联系集而存在的实体集称为依赖实体集。

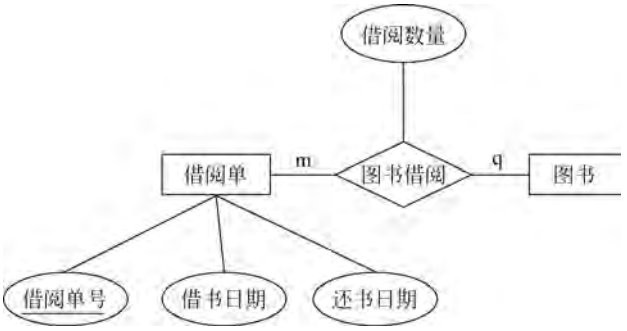

图 3-29 依赖于联系集的实体集

由此可以得出,依赖有以下两种约束情况。

(1) 实体集与实体集之间的依赖约束,即弱实体集依赖于强实体集。

(2) 实体集与联系集之间的依赖约束,即实体的存在依赖于联系集中的联系。

所谓依赖约束是指联系中一种实体的存在依赖于其他实体集中实体的存在或依赖于该联系集中联系。根据以上分析可以看出,依赖于联系集而存在的实体集一般是指伴随着业务发生形成的原始单据,具有独立的业务处理需求,是一个业务实体,即强实体,如入库单、存款单、销售单等。在 E-R 模型设计时,一般将依赖于业务而发生的一些单据直接建模为依赖实体集,同时,建立与所依赖的联系集的关联。

3. 多值联系的建模

多值联系指在同一个给定的联系集中,相关联的相同实体之间可能存在多个联系。如

图 3-27 所示的实体集读者和图书之间的多对多联系集，表示一个读者可以借阅多本图书，同一本图书可以在不同时间被多个读者借阅，借阅联系的属性有借书日期、还书日期。如果考虑一个读者在不同的时间可以借阅同一本图书，此时，E-R 模型存在以下问题。

当一个读者多次借阅同一本图书时(同一个读者在不同时间借阅了同一本图书)，联系集中无法标识一个联系。也就是说，“借阅”联系不仅是一个多对多联系，而且是一个多值联系。为了唯一标识多值联系中的多个联系，即解决如图 3-27 所示的类似“借阅”联系的多值联系，可以将多值联系“借阅”建模为一个弱实体集或依赖实体集“借阅单”。如果建模为弱实体集，则“借阅单”依赖于与它关联的“读者”实体集和“图书”实体集；如果建模为依赖实体集，则“借阅单”依赖于与它关联的“拥有”和“包含”联系集。也就是说，多值联系的建模问题可转换为依赖约束的建模问题。在实际的 E-R 建模应用中，需要根据实际业务的语义，判断是将多值联系建模为弱实体集还是依赖实体集。图 3-30 所示是将“借阅单”建模为依赖实体集，从而解决了读者借阅图书的多值联系。

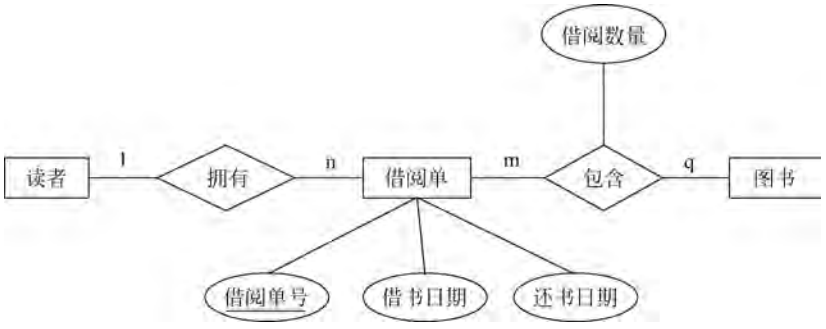

图 3-30 多值依赖的建模解决方法 1

因此，在数据库概念结构设计时，如果两个实体(可推广到多个实体)之间的联系非常复杂，它们之间就可能存在多对多的联系。处理多对多联系的方法，是在它们之间插入第三个实体(弱实体或依赖实体)，使原来的多对多联系化解为一对多联系。当然，也可以根据实际情况，例如，考虑到数据库应用系统的查询效率，可以建立多个实体之间的多元联系。图 3-31 同样解决了图 3-27 中读者和图书之间的多对多联系。

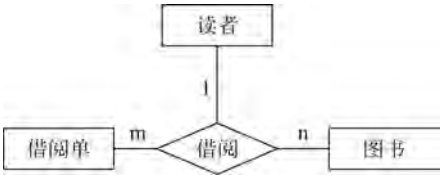

图 3-31 多值依赖的建模解决方法 2

4. 实体的子类型和超类型

实体的子类型和超类型类似于对象模型中对象之间的一般和特殊关系。如果一个实体类型的全部实体也属于另一个实体类型，并且具有自己的特殊特征，则前者称为子类，后者称为超类。例如，前面提到的研究生、本科生都属于学生实体的子集，研究生不仅拥有学生的所有特性，而且也可定义自己的属性，如导师、研究方向等。因此，实体类型“学生”称为实

体类型“研究生”“本科生”的超类,而“研究生”“本科生”就是实体类型“学生”的子类。子类自动继承超类的属性,子类也可有自己特殊的属性。

一个实体类型可以有子类,而子类也可以有自己的子类,这样便构成了实体类型的层次结构。在数据库建模中,实体集可以根据多个不同特征进行分类(自顶向下的设计过程),对高层实体集进行分类而产生若干低层实体集,也可以把具有相同属性的多个低层实体集进行综合(自底向上的设计过程),构成一个高层次的实体集。从一个实体集分出特化的实体集是为了强调低一层次的实体集之间的区别,这些低一层次的实体集可以有自己的属性,也可以参加某些联系,而高层次的实体集是提取低层次实体集的共性,隐藏之间的区别,从而简化模型。

E-R 模型使用实体集的继承和 ISA 联系来描述实体集特殊化和泛化的概念。如图 3-32 所示,描述了学生实体集的层次关系。

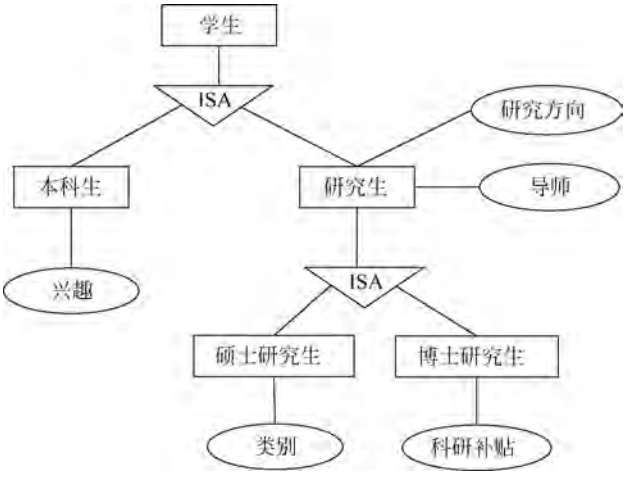

图 3-32 学生实体集的层次关系

5. 多元联系和二元联系

数据库中的联系一般都是二元联系。不过在很多场景下需要使用多元联系来表示几个实体集之间的相互关联关系。如图 3-18 所示的供应商、项目、零件之间的联系就是典型的多元联系。此时多元联系比通过建立相关实体之间的相邻二元联系集合具有更为精确的描述能力。图 3-18(b)所示的三元联系可以反映出某供应商对某个项目提供某种零件的信息。如果建模为如图 3-33 所示的实体间的相邻二元联系,则无法体现某供应商向某个项目供应某种零件的语义,从而无法实现相同的目的。因此,多元实体集之间的联系不能简单地用多个二元实体集之间的联系代替,一定要根据管理需求中多个实体间联系的语义描述,以及所涉及的实体,来确定是构造多元联系还是二元联系。

事实上,一个多元联系集也可以用一组不同的二元联系集代替。考虑一个抽象的三元联系集 R,它将实体集 A、B、C 联系起来。用实体集 E(为了表示联系集而创建的依赖实体集或弱实体集)替代联系集 R,并建立三个联系集:R_A(联系 E 和 A)、R_B(联系 E 和 B)、R_C(联系 E 和 C)。如果联系集 R 有属性,则将其属性赋给实体集 E,并为 E 建立一个特殊的标识属性(因为每个实体集都应该至少有一个属性,以区别实体集中的不同成员)。因此,图 3-18 所示的三元联系即可转换为图 3-34 所示的三个二元联系。

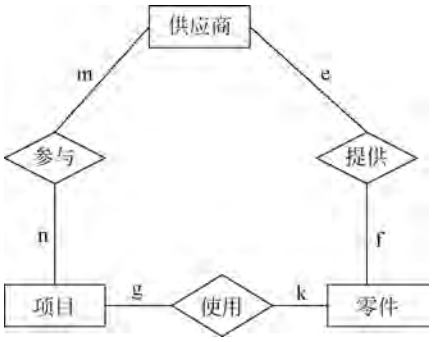

图 3-33 多元联系转换为实体间的相邻二元联系

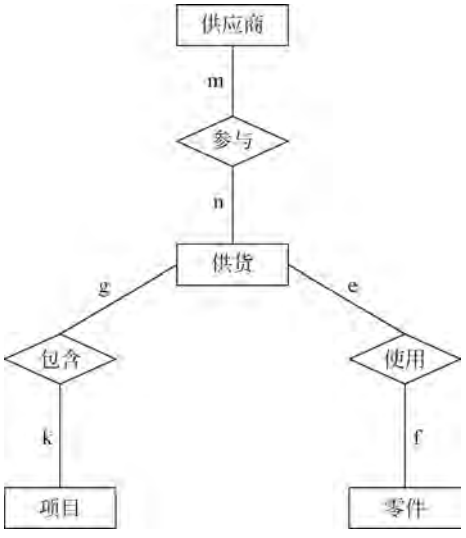

图 3-34 多元联系转换为等价的二元联系

需要注意的是,将多元联系转换为二元联系时,可能会带来如下问题。

(1) 为了表示联系集而创建的实体集可能需要额外的标识属性,增加了设计的复杂度及空间占用。

(2) 多元联系可以更清楚地表示多个实体对同一联系的共同参与,而转换为二元联系后,在 E-R 图中丢失了直接表示多个实体之间存在的多元联系手段。

(3) 在多元联系转换为二元联系时,需要注意保持原多元联系上的联系语义。在转换过程中,对多元联系上的约束可能无法转换为二元联系上的约束(即无法有效保留约束)。

在多元联系中,若只考虑两两实体集间的联系,可以采用多个二元联系来实现。

3.4 数据库逻辑结构设计

E-R 图表示的概念模型是用户数据要求的形式化,只是描述了数据库的概念模式。正如前面所述,它是独立于任何一种数据模型的概念信息结构,也不为任何一个 DBMS 所支持。为了能够用某一具体的数据库管理系统实现用户需求,使计算机能处理模型中的信息,

被关系数据库所接受,必须进行信息化,将概念模型落实,进一步转换为更为具体的数据模型,即将 E-R 模型转换为关系数据库所支持的逻辑模式——关系模式。

3.4.1 逻辑结构设计的任务

从理论上讲,设计数据库逻辑结构的步骤应该是:首先,考虑实现数据库的数据库管理系统所支持的数据模型是什么;然后,按转换规则将概念模型转换为选定的数据模型,再从支持这种数据模型的各个 DBMS 中选出最佳的 DBMS,根据选定的 DBMS 的特点和限制对数据模型做适当修正。但实际情况常常是先给定了计算机和 DBMS,再进行数据库逻辑模型设计。由于设计人员并无选择 DBMS 的余地,所以,在概念模型向逻辑模型设计时就要考虑到适合给定的 DBMS 的问题。现行的 DBMS 一般只支持关系、网状或层次模型中的某一种,即使是同一种数据模型,不同的 DBMS 也有其不同的限制,提供不同的环境和工具。

通常把概念模型向逻辑模型的转换过程分为以下三步进行。

第一步:把概念模型转换成一般的数据模型。

第二步:将一般的数据模型转换成特定的 DBMS 所支持的数据模型。

第三步:通过优化方法将其转换为优换的数据模型。

概念模型向逻辑模型的转换步骤,如图 3-35 所示。

图 3-35 逻辑结构设计的步骤

目前设计的数据库系统大都采用支持关系模型的 RDBMS,因此,逻辑结构设计的任务就是把概念结构设计好的基本 E-R 图转换为与选用的某个具体的 DBMS 所支持的数据模型相符合的逻辑结构,针对关系型 DBMS 产品,逻辑结构设计的主要工作就是将概念模型转换为关系数据库所支持的关系模式。因此,关系数据库逻辑结构设计是指定义数据库中所包含的各关系模式的结构,包括各关系模式的名称、每个关系模式中各属性的名称、取值范围及约束、主码和外码等,设计的结果是一组关系模式。

3.4.2 概念模型转换为关系模型的方法

概念模型向关系模型转换需要解决的问题包括以下两个。

(1) 如何将实体集和实体集之间的联系转换为关系模式;

(2) 如何确定这些关系模式的属性、主码和外码。

关系模型的逻辑结构是一组关系模式的集合,将 E-R 图转换为关系模型实际上就是将实体集、属性以及联系集转换为相应的关系模式。这种转换要遵循一定的原则进行。

1. 实体集的转换规则

1) 强实体集转换方法

E-R 模型中的每一个强实体集转换为一个关系模式,该关系模式的名称为强实体集的名称,其属性就是原实体集中的属性,主码就是原实体集的标识符。

2）弱实体集转换方法

由于弱实体集的各实体需借助强实体集的主标识符进行标识。因此，弱实体集转换的关系模式的属性由弱实体集本身的描述属性与所依赖的强实体集的主标识符构成，主码由所依赖的强实体集标识符和弱实体集的标识符构成，外码是强实体集标识符，关系模式名就是弱实体集的名称。如图 3-28 所示的“开课班”弱实体集转换的关系模式为

开课班(课程号,开课班号,年份,学期,上课地点,上课时间)

其中，课程号和开课班号共同构成开课班关系模式的主码，课程号作为外码。

2．实体集之间联系集的转换规则

E-R 模型向关系模型转换时，实体集之间联系集转换为关系模式有以下不同的情况。

1）联系类型 1∶1 的转换方法

联系类型为 1∶1 联系有两种转换方法。

（1）将联系转换为一个独立的关系模式。由联系转换的关系模式的属性是由联系本身的属性与参与该联系的各实体集的标识符构成，且每个实体集的标识符均可作为该关系模式的候选码，每个实体集的标识符均是外码。

（2）联系不单独转换为一个关系模式。将 1∶1 联系与任意一端实体集所对应的关系模式合并，则需要在被合并关系中增加属性，其新增的属性为联系本身的属性与联系相关的另一个实体集的标识符，新增属性后原关系模式的主码不变，增加的另一端实体集的标识符为关系模式的外码。

【例 3-1】 图 3-36 所示为某国内旅游管理信息系统数据库中涉及的旅游团和保险实体的 E-R 图，两实体的联系类型为一对一联系，将 E-R 图转换为关系模式。

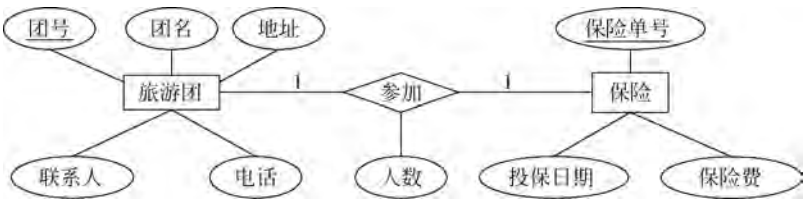

图 3-36　联系类型 1∶1 转换为关系模式的实例

解：（1）将两个实体集转换为两个关系模式：分别是旅游团关系模式和保险关系模式，其中关系模式中标有下画线的属性为候选码。

旅游团(团号,团名,地址,联系人,电话)

保险(保险单号,投保日期,保险费)

（2）将联系转换为关系模式，有三种方案。

方案 1　将联系形成一个独立的关系模式：

参加(团号,保险单号,人数)

方案 2　将参加与旅游团两个关系模式进行合并，则关系模式为

旅游团(团号,团名,地址,联系人,电话,人数,保险单号)

方案 3　将参加与保险两个关系模式进行合并，则关系模式为

保险(保险单号,投保日期,保险费,人数,团号)

将上面三种方案进行比较,不难发现在方案 1 中,由于关系多,增加了系统的复杂性;在方案 2 中,由于并不是每个旅游团都参加保险,这样就会造成方案 2 旅游团关系中的保险单号属性的 NULL 值过多;相比较起来,方案 3 比较合理。最终图 3-36 转换的关系模式集合为

旅游团(团号,团名,地址,联系人,电话)
保险(保险单号,人数,投保日期,保险费,人数,团号)

由此可以看出,保险关系模式不仅描述了保险单的信息,同时描述了保险和旅游团之间的关系,此联系是通过保险关系模式的外码(团号)属性联系起来。

2) 联系类型 1∶n 的转换方法

当 E-R 模型在向关系模型转换时,实体之间的 1∶n 联系可以有两种转换方法:

(1) 将联系转换为一个独立的关系模式。关系模式的属性由与该联系相连的各实体集的标识符以及联系本身的属性组成,关系模式的主码为 n 端实体集的标识符,每个实体集的标识符均是关系模式的外码。

(2) 联系不单独转换为一个关系模式。将联系和 n 端实体集对应的关系模式合并,需要在 n 端实体集对应的关系模式中增加联系的属性和 1 端实体集的标识符,新增属性后原关系模式的主码不变,外码则是 1 端实体集的标识符。

【例 3-2】 图 3-37 给出了国内旅游管理信息系统数据库中涉及的旅游团和旅客实体的 E-R 图,两实体的联系类型为一对多联系,将 E-R 图转换为关系模式。

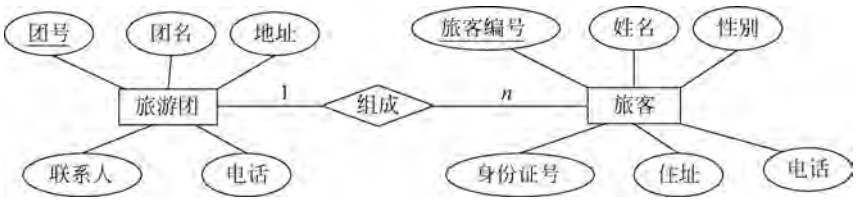

图 3-37 联系类型 1∶n 转换为关系模式的实例

解:(1) 将两个实体转换为两个关系模式:分别是旅游团关系模式和旅客关系模式,其中关系模式中标有下画线的属性为主码。

旅游团(团号,团名,地址,联系人,电话)

旅客(旅客编号,姓名,性别,身份证号,地址,电话)

(2) 将联系转换为关系模式,有两种方案。

方案 1 将联系形成一个独立的关系模式:

组成(旅客编号,团号)

方案 2 将联系"组成"和旅客关系模式进行合并,则关系模式为

旅客(旅客编号,姓名,性别,身份证号,地址,电话,团号)

比较以上两种方案可以发现,方案 1 中使用的关系多;方案 2 中使用的关系少,特别适用于旅游团旅客变化小的应用场合。为了降低系统的复杂性,一般情况下采用第二种方案。最终图 3-37 转换的关系模式集合为

旅游团(团号,团名,地址,联系人,电话)
旅客(旅客编号,姓名,性别,身份证号,地址,电话,团号)

同样，两个关系模式的联系是通过旅客关系模式的外码“团号”联系起来。因此，如果两个实体之间是一对多的联系，且联系没有自身的属性，则联系不必转换为一个独立的关系模式，只需要将一方实体集的标识符加入多方实体集转换的关系模式中。

【例 3-3】 图 3-38 是同一实体集内部的一对多联系，将其转换为关系模式。

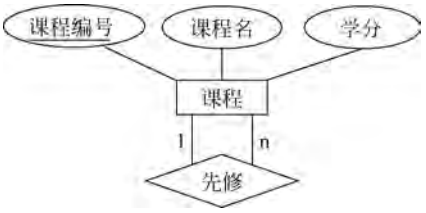

图 3-38　同一实体集内部的一对多联系

解：(1) 方案 1　转换为两个关系模式：

课程(课程编号，课程名，学分)

先修(先修课程号，课程编号)

(2) 方案 2　转换为一个关系模式：

课程(课程编号，课程名，学分，先修课程号)

其中，由于同一关系中不能有相同的属性名，故将课程的课程编号改为先修课程号。以上两种方案相比较，第二种方案的关系少，且能充分表达原有的数据联系语义，所以，采用第二种方案会更好些。

3) 联系类型 m∶n 的转换方法

一个 m∶n 联系转换为一个独立关系模式。关系模式的属性由参与 m∶n 联系的各实体集的标识符以及联系本身的属性构成，新关系的主码为各实体集的标识符组合(该关系模式的主码为多属性构成的组合码，也称多属性码)，每个实体集的标识符均为该关系模式的外码。

【例 3-4】 图 3-39 给出了高校学生选课系统数据库所涉及的学生和课程的 E-R 图，两实体的联系类型为多对多联系，将 E-R 图转换为关系模式。

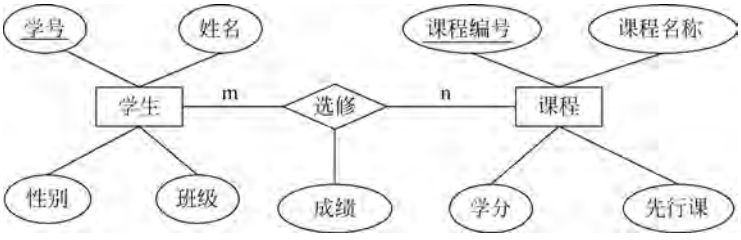

图 3-39　联系类型 m∶n 转换为关系模式的实例

解：根据实体间多对多的转换规则，转换的关系模式为

学生(学号，姓名，性别，班级)

课程(课程编号，课程名称，学分，先行课)

选修(学号，课程编号，成绩)

关系模式中标有下画线的属性为候选码，在此也作为主码。其中，联系“选修”关系模式

的候选码是学号和课程编号的组合,学号、课程编号分别作为选修关系模式的外码,必须满足关系的参照完整性约束。此题能否采用其他方案呢?

假如对联系选修不建立独立的关系模式,而是和学生或课程关系进行合并,则转换的关系模式为

学生(学号,姓名,性别,班级)
课程(课程编号,学号,课程名称,学分,先行课,成绩)

或

学生(学号,课程编号,姓名,性别,班级,成绩)
课程(课程编号,课程名称,学分,先行课)

这样,对于多个学生选修相同的课程或一个学生选修多门课程,分别存在相同课程、同一学生的信息重复存储多次的情形,造成数据存储冗余太大,给维护带来不便。

由此可以看出,对于实体之间 m∶n 的联系类型,联系必须转换为一个独立的关系模式。

【例 3-5】 图 3-40 是同一实体集内部的多对多联系,将其转换为关系模式。

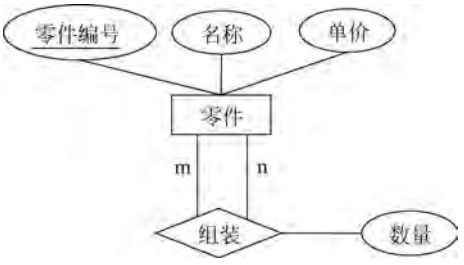

图 3-40 同一实体集内部的多对多联系

解:根据实体间多对多的转换规则,转换的关系模式为

零件(零件编号,名称,单价)

组装(组装件号,零件号,数量)

其中,组装件号为组装后的复杂零件号。由于同一个关系中不允许存在相同的属性名,因此,取名为组装件号。

4) 三个或三个以上实体集之间多元联系的转换方法

(1) 一对多的多元联系转换为关系模型的方法:按 1∶n 联系的转换规则进行转换,即可将与联系相关的其他实体集的标识符和联系自身的属性作为新属性加入 n 端实体集中。

(2) 多对多的多元联系转换为关系模型的方法:按 m∶n 联系的转换规则进行转换,联系必须转换一个独立的关系模式,该关系的属性为多元联系相连的各实体的标识符及联系本身的属性,主码为各实体转换为各自关系模式主码的组合。

【例 3-6】 图 3-41 所示的 E-R 图给出了某公司网上采购系统数据库涉及的供应商、商品和采购员的三个实体之间的多对多的多元联系,将其 E-R 图转换为关系模式。

根据多实体间联系的转换规则,三个实体转换为三个关系模式,联系转换为一个独立的关系模式:

采购员(采购员编号,姓名,联系电话)

供应商(供应商号,供应名称,地址)

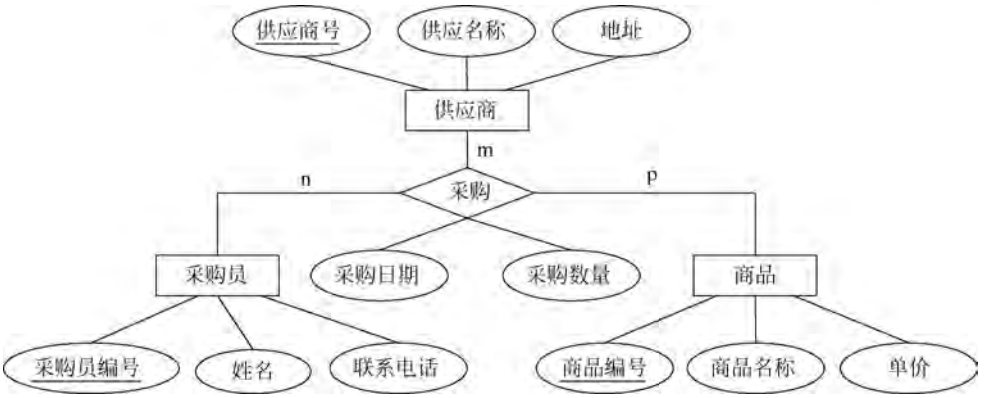

图 3-41　多实体之间多对多联系实例

商品(商品编号,商品名称,单价)

采购(采购员编号,供应商号,商品编号,采购日期,采购数量)

其中,关系模式中标有下画线的属性为候选码,也即主码。

3. 子类型和超类型的转换规则

将子类型和超类型转换为关系模式有两种方法。

(1) 超类实体和子类实体分别转换为单独的关系模式。其中,超类实体对应关系模式的属性为超类实体的属性(即公共属性),而各子类实体对应的关系模式的属性由该子类的特有的属性和超类实体的标识符属性组成,子类实体对应关系模式的主码、外码均是超类实体的标识符。

(2) 只将子类实体转换为关系模式。其属性包含超类实体的全部属性和子类的特有属性,其主码就是超类实体的标识符。

其中,方法(1)是通用的方法,任何时候都可以采用。方法(2)的缺陷之一是不能表示超类实体的其他子类实体。如果一个超类实体均是各子类实体时,其公共属性的值重复存储多次。因此,方法(2)应该有条件地使用。

在 E-R 模型转换为关系模型过程中,除了要遵循以上介绍的转换原则外,还应注意以下两点。

(1) 命名和属性域的处理。关系模式的命名,可以采用 E-R 图中原来的命名,也可以另行命名。命名应有助于对数据的理解和记忆,同时,应尽可能避免重名,具体的 DBMS 一般只支持有限的几种数据类型,而 E-R 图是不受这个限制的。如果 DBMS 不支持 E-R 图中的某些属性的域,则应做相应的调整。

(2) 非原子属性的处理。E-R 模型中允许非原子属性,这不符合关系的第一条性质——满足第一范式的条件。解决的办法是对 E-R 图中出现的非原子属性进行展开。

E-R 模型和关系模型相对于物理存储文件来说,都是对现实世界的抽象逻辑表示,例如,在数据库的三层模式中,概念模式和逻辑模式就是一回事。因此,两种模型采用类似的设计原则,可以将 E-R 图设计转换为关系的设计。将数据库的表示从 E-R 图转换为表的形式是由 E-R 图产生关系数据库设计的基础。

3.4.3 关系模型优化

数据库逻辑结构设计的结果不是唯一的。得到由 E-R 模型转换的初步关系模式后,为了提高数据库应用系统的性能,还应当适当地修改,调整关系模式的结构,这指关系模式的优化。关系模型优化通常以规范化理论为指导(参照第 4 章内容),在此主要介绍优化的具体方法和子模式的设计问题。

1. 优化方法

1) 确定数据依赖

用数据依赖分析表示数据项之间的联系,写出每个数据项之间的数据依赖,即根据需求分析阶段得到的语义,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) 消除冗余的联系

对于各个关系模式之间的数据依赖进行最小化处理,消除冗余的联系。

3) 分析数据依赖

按照数据依赖的理论,逐一分析构造的关系模式,检查是否存在部分函数依赖、传递函数依赖、多值依赖等。如果存在部分函数依赖、传递函数依赖,则要消除其中的依赖关系,确定关系模式分别属于第几范式。

4) 判断是否合并或分解

结合实际应用环境、数据库的规模及不同应用对数据处理的要求,分析这些关系模式是否适合具体的应用环境,确定是否需要对关系模式进行关系的合并或分解。如果两个关系模式的主码相同,则可以进行合并,而为了数据库的完整性,对于某些不满足 BCNF 的关系模式,可以考虑进行分解或合并,以提高数据操作效率和存储空间的利用率。

模式分解分为水平分解和垂直分解。

(1) 水平分解是指把原来关系的元组依据其使用的频率分为若干个子集合,定义每个子集合为一个子关系。大体上,水平分解适用于两种情形。

第一种情形:满足“80/20 原则”的应用。“80/20 原则”的含义是在一个大关系中,把经常被使用的数据分解出来,形成一个子关系模式,这部分数据大约占 20%。这样可以减少查询的数据量,提高整个系统的响应速度。

第二种情形:并发事务进程存取不相交的数据。如果关系 R 上具有 n 个事务,而且多数事务存取的数据不相交,则 R 可以分解为小于或等于 n 个子关系,使每个事务存取的数据对应一个关系。这样可以提高整个系统的并发响应效率。

(2) 垂直分解指根据属性的依赖程度,把关系模式 R 的属性分解为若干个子集合,形成若干个子关系模式。其一般原则是把经常在一起使用的属性从 R 中分解出来形成一个子关系模式。事实上,进行关系模式规范化的过程是利用垂直分解方法消除不必要的数据依赖,提高关系模式的范式等级。通过垂直分解可以提高某些事务的效率,但也可能使另一些事务不得不执行连接操作,从而降低了效率。因此,是否进行垂直分解取决于分解后 R 上的所有事务的总效率是否得到了提高。具体进行分解时,可以采用相应的分解算法确定分解的方案。当然,如果关系模式比较简单,也可以直接根据直观观察确定分解的方案,不管采用哪种方法,必须保证分解的无损连接性和函数依赖保持性。

需要注意的是，在应用规范化理论进行优化关系模式时，一定要分析应用环境，千万不要盲目追求高范式，这是因为并不是规范化程度高的关系模式就越优，由于规范化程度高，数据分离程度就越低，这对于查询操作会带来不利的影响。例如，当查询涉及两个或多个关系模式的信息时，系统就必须经常使用连接运算，而连接运算恰恰是一种代价高昂的运算。可以说，关系模型低效的主要原因就是由连接运算引起的，如果这种操作频繁出现，必将大大降低处理的速度，这时可以考虑将几个关系进行合并。尽管在理论上非 BCNF 的关系模式会存在不同程度的更新异常或数据冗余，但是如果在具体的应用中，该关系主要用于查询，不执行更新操作，那么更新异常等问题在实际应用中就不会产生影响。

因此，对于一个具体的应用来说，关系模式到底规范化到什么程度，需要权衡响应的时间和潜在的问题这两者的利弊决定。规范化理论为数据库逻辑设计的设计人员判断关系模式的优劣提供了理论标准，可用来预测模式可能出现的问题，使数据库设计工作有了严格的理论基础。

2. 用户子模式的设计

把 E-R 模型转换为全局的逻辑模型后，还应根据局部应用的需求，结合具体的 DBMS 的特点设计用户子模式。用户子模式也称外模式，是用户看到的数据模式，各类用户有各自的子模式。关系数据库管理系统中提供的视图是根据用户子模式设计的。设计用户子模式时只考虑用户对数据的使用要求、习惯及安全性要求，不用考虑系统时间效率、空间效率、易维护等问题。

用户子模式设计时应注意以下问题。

1）使用更符合用户习惯的别名

在合并各分 E-R 图时应消除命名的冲突，以使数据库系统中同一关系和属性具有唯一的名字。这在设计数据库整体结构时是非常必要的。但命名统一后会使某些用户在使用上感到别扭，用户子模式（视图）的方法可以有效地解决该问题。必要时，可以设计视图重新定义命名，使其与用户习惯一致，以方便用户的使用。

2）对不同级别的用户可以定义不同的子模式

由于视图能够对表中的行和列进行限制，所以它还具有保证系统安全性的作用。对不同级别的用户定义不同的视图，可以保证系统的安全性。

例如，假设有关系模式：

图书（图书编号，书名，类别，作者，单价，出版日期，出版社，地址，出版社负责人，联系电话，库存数量，读者卡号，借书日期，还书日期）

对一般读者来说，在借阅图书时，首先需要了解相关图书的信息，为一般读者建立视图：

图书 1 （图书编号，书名，类别，作者，单价，出版日期，出版社，库存数量）

对图书馆的图书管理员来说，需要查询图书的详细借阅及归还情况，为图书借阅部门建立视图：

图书 2 （图书编号，书名，类别，读者卡号，借书日期，还书日期）

对图书馆负责采购图书的采购部门来说，需要了解出版社的详细信息，为采购部门建立视图：

图书 3 （出版社，地址，出版社负责人，联系电话）

在建立视图后，图书 1 视图中包含了允许一般读者查询的图书属性；图书 2 视图中包

含允许图书管理员查询图书借阅的属性；而对图书馆的采购部门，则可以利用图书 3 查询出版社的全部属性数据。这样，既方便了使用，又可以防止用户非法访问本来不允许他们查询的数据，保证了系统的安全性。

3）简化用户对系统的使用

实际中，某些局部应用经常要使用某些复杂的查询，这些查询包括多表连接、限制、分组、统计等。为了方便用户，可以将这些复杂查询定义为视图，用户每次只对定义好的视图进行查询，避免每次查询都要对其进行重复描述，大大简化了用户的使用。


视频讲解

3.4.4 逻辑结构设计案例

由 3.3.3 节设计的高校图书管理系统的 E-R 图，可以得到如表 3-4 所示的高校图书管理系统的一组关系模式及相关信息。表中的一行为一个关系模式。

表 3-4 图书管理系统数据库的关系模式信息

数据性质	关系名	属性	说明
实体	读者类别	类别编号，类别名称，可借阅天数，可借阅数量，超期罚款额	别编号为与读者关系合并后的新增属性
实体	读者	读者卡号，姓名，性别，单位，办卡日期，卡状态，类别编号	
实体	图书	图书编号，书名，类别，作者，出版社，出版日期，单价，库存数量	
1∶n 联系	属于	读者卡号，类别编号	与读者关系合并
m∶n 联系	借阅	读者卡号，图书编号，借书日期，还书日期	

其中，带有下画线的属性为关系的码。

根据概念模型向关系模型的转换规则，三个实体转换为三个关系模式，1 个 m∶n 联系转换为 1 个关系模式，1 个 1∶n 联系形成的关系模式与相应的实体形成的关系模式进行合并，因此，经过优化后，该数据库可以设计为四个关系模式，分别是：

读者类别(类别编号，类别名称，可借阅天数，可借阅数量，超期罚款额)

读者(读者卡号，姓名，性别，单位，办卡日期，卡状态，类别编号)

图书(图书编号，书名，类别，作者，出版社，出版日期，单价，库存数量)

借阅(读者卡号，图书编号，借书日期，还书日期)

3.5 数据库的物理结构设计

数据库最终要存储在物理设备上。所谓物理结构设计，是指对于给定的逻辑数据模型，选取一个最适合应用环境的物理结构的过程。数据库的物理结构主要指数据库在物理设备上的存储结构与存取方法，它依赖于给定的计算机系统。设计的主要任务是选择合适的存储结构、存取方法和存取路径。

针对关系数据库系统而言，数据库的物理结构设计相对简单，主要解决的问题是物理存储结构的设计和数据库存取方法的设计，其中，物理结构设计解决数据文件中记录的存储问题，使应用要访问的记录尽量存储在同一磁盘块上；数据库存取方法的设计，解决如何快速

地找到所需的记录，使得磁盘 I/O 操作次数最少。数据库的物理结构设计主要任务是使用 SQL 创建数据库，定义数据库的三级模式结构；其主要的设计目标有两个。

(1) 提高数据库的性能，特别是满足主要应用的性能要求；

(2) 有效地利用存储空间。

这两个目标中，第一个目标更为重要，因为性能仍然是当今数据库系统的薄弱环节。

一般来说，物理结构设计与 DBMS 的功能、性能、DBMS 提供的物理环境和可利用的工具、应用环境及数据存储设备的特性都有密切关系。数据库用户通过 DBMS 使用数据库，物理结构设计比起逻辑结构设计更加依赖 DBMS，数据库设计者只能在 DBMS 提供的手段范围内，根据需求和实际条件适当地选择，同时，必须仔细阅读 DBMS 的有关手册，充分了解其限制条件，充分利用其提供的各种手段完成物理结构的设计。

不同的计算机系统提供的物理环境、存储结构和存取方法是不相同的，没有通用的物理结构设计方法可供遵循，这里只介绍物理结构设计要考虑的因素及设计内容。

3.5.1 影响物理结构的主要因素

数据库的物理结构主要由以下因素决定。

1) 应用处理需求

在进行数据库物理结构设计前，应先弄清楚应用的处理需求，如吞吐量、平均响应时间、系统负荷等，这些需求直接影响着物理结构设计方案的选择，而且会随着应用环境的变化而变化。

2) 数据的特性

数据的特性主要指数据的结构、关系之间的联系、数据的检索频度等。数据本身的特性对数据库物理结构设计也会有较大影响，例如，关系中每个属性值的分布和记录的长度与个数也会影响到数据库的物理存储结构和存取方法的选择。在数据库设计阶段，数据特性很难准确估计，会随数据库状态的改变而变化。

3) 数据的使用特性

数据的使用特性包括各个用户的应用所对应的数据视图、各种应用的处理频度、使用数据的方法、对系统的重要程度，这些是对时空效率进行平衡和优化的主要依据。一般来说，物理结构设计不能均等地考虑每一个用户，必须将用户分类，以保证重点用户的重点应用。

4) 可用性要求

数据库的可用性要求指适应用户的要求，维护数据库逻辑上、物理上的完整性的能力。人们都希望数据库有较高的可用性，但为此必须付出较大的代价，所以必须权衡得失。

5) 应用环境

从整个计算机系统来说，数据库的应用仅是其负荷的一部分，数据库的性能不仅决定于数据库的设计，而且与 OS、DBMS、网络的运行环境有关，受到计算机硬件资源的制约。DBMS 的特性主要指 DBMS 的功能、提供的物理环境和工具，特别是存储结构和存取方法。由于每一种 DBMS 都有自己的特点和不足，只有真正了解 DBMS 的特点，才能设计出一个充分发挥 DBMS 特色的物理结构。

数据库的物理结构设计可以分为两步进行：

(1) 确定数据的物理结构，即确定数据库的存储结构和存取方法。

(2) 对物理结构进行评价,对物理结构评价的重点是时间和效率。

3.5.2 物理结构设计的任务

关系数据库的物理结构设计一般比层次、网状数据库的物理设计简单。数据库的存储管理主要由 DBMS 完成,既不用选择存取路径,又不用确定内部存储结构,关系数据库管理系统一般把数据的内部存储结构、存取路径完全向用户隐藏,一切都由 DBMS 自动进行,并且系统的内部都有优化程序,能够在进行路径选择时,先对各种路径的成本进行估算,然后自动选择最佳的路径,这恰恰是关系数据库系统的优越性之一。

关系数据库物理结构设计的任务主要是,数据库设计者利用 DBMS 提供的功能,使用 SQL 描述数据库三级模式结构,通过定义数据库、数据表、视图及索引等确定数据库的文件组织结构、文件存储结构和数据存取方法,并在创建数据库时制定数据库的存储策略,包括确定日志、备份等的存储安排和存储结构,确定系统配置等。本节重点介绍数据库的存储结构和数据存取方法。

1. 确定数据库的存储结构

存储结构的设计主要包括文件物理存储结构的设计和逻辑模式存储结构的设计。针对设计的存储结构,还需要合理安排数据的存放位置及各类文件的物理存储设备。确定存储结构和数据的存放位置要综合考虑存取时间、存储空间利用率和维护代价三方面的因素。这三个方面常常相互矛盾,需要进行权衡,选择一个折中方案。

1) 文件的物理存储结构

文件的物理存储结构即文件中的记录组织方式,是数据库物理结构设计的基础,它包括记录和块存储在磁盘上的方式,以及记录和块之间相互联系的方法。目前,文件中组织记录的常用方法有顺序文件、链式文件、B+树文件、散列文件和堆文件。

(1) 顺序文件。顺序文件指把数据表的元组按照某个属性或属性组的值的次序依次存放到存储介质上依次相连的存储块中的存储方式。其优点是结构简单,容易实现,检索效率高,不需要额外的开销,因为对每个文件要求存放在存储介质上的连续物理块中;其缺点是存储空间利用率不高,并且用户创建文件时要给出文件的大小,不利于文件的动态增加和修改。因此,顺序文件适合变化不大的顺序访问的文件。

(2) 链式文件。链式文件指用链表结构,并按照链表结点的地址把元组动态存放到存储介质上的不一定连续的存储块中的存储方式。链式文件结构不要求连续存放,其优点是存储空间利用率高,用户创建文件时不必指出文件的大小,文件容易动态扩充和修改。由于每个文件有一个索引表,而索引表也由物理块存储,因此存储开销大,需要额外的外存空间。

(3) B+树文件。磁盘的 I/O 操作次数对索引的使用效率至关重要。B+树作为关系数据库系统中使用最为广泛的一种索引结构,其查询效率高效、稳定。B+树文件结构中,非叶子结点仅用于索引,不保存数据记录;叶子结点存放若干条数据记录,而不是存放索引项和指向记录或物理文件块的指针,叶子结点按照索引项的大小从小到大顺序链接构成一个有序链表。这样,在磁盘页大小相同的情况下,树的深度更小,所需访问的磁盘 I/O 次数更少,从而保证 B+树索引结构具有良好的查询、插入和删除性能。由于 B+树结点大小一般等于磁盘块大小,因此将会造成空间浪费的缺点。

(4) 散列文件。散列文件是基于散列函数和桶的存储方式,把记录按照某属性(组)的

值，依据一个散列函数计算其存放的位置桶号(块号)。虽然可以不通过索引就能够进行数据访问，但只适用于定长记录文件和按照记录随机查找的访问方式，而且需要解决散列函数的定义和冲突问题等。

(5) 堆文件。堆文件也称无序记录文件，磁盘上存储的记录是无序的，记录可存储于任意有空间的位置，更新效率高，但检索效率低。

2) 逻辑模式的存储结构

根据操作系统提供的文件物理结构，DBMS 提供了关于数据库、基本表、索引文件、日志文件等文件的存储结构。因此，针对具体的应用，需要进一步设计和设置相应文件的存储结构。

针对关系数据库来说，像数据库、基本表、索引文件、日志文件等文件的存储结构，通常使用 DBMS 的默认存储结构或做进一步设置。例如，在 SQL Server 2019 中，创建数据库时，可以设置数据库的大小限制、数据库的增长限制等，同时，可以设置日志文件的大小限制和日志文件的增长限制等。

逻辑模式的存储结构设计主要包括存储的关系模式，关系模式的数据项，数据项的类型、宽度，是否是主键、外键，是否是索引等。

3) 确定数据的存放位置

为了减少访问磁盘的 I/O 操作次数、提高系统性能，应该根据应用情况将数据的易变部分与稳定部分、经常存取部分和存取频率较低部分分开存放。对于有多个磁盘的计算机，可以采用下面几种存取位置的分配方案。

(1) 将表和索引放在不同的磁盘上，这样在查询时，两个磁盘驱动器并行工作，可以提高物理 I/O 读写的效率。

(2) 将比较大的表分别放在两个磁盘上，以加快存取速度，这在多用户环境中特别有效。

(3) 将日志文件、备份文件与数据库对象(表、索引等)放在不同的磁盘上，以改进系统的性能。

(4) 对于经常存取或存取时间要求高的对象(如表、索引)应放在高速存储器(如硬盘)上，对于存取频率小或存取时间要求低的对象(如数据库的数据备份和日志文件备份等，只在故障恢复时才使用)，如果数据量很大，可以存放在低速存储设备上。

4) 选取存储介质

存储介质指用于存储系统各类文件的物理存储设备。通用的存储介质主要包括磁盘、磁带、光盘、磁盘阵列 RAID、网络存储等。

针对应用系统的具体需求，通过物理结构设计，根据目前常用存储介质的特征和性能指标，综合考虑应用需求、存储介质的容量、存取速度、费用、接口等因素，选择适用于应用需求的存储介质以及由存储介质组成的存储系统。

5) 确定系统配置

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

系统配置变量很多,如同时使用数据库的用户数、同时打开的数据库对象数、数据库的大小等,这些变量影响存取时间和存储空间的分配,在物理设计时就要根据应用环境确定这些变量,以使系统性能最佳。

在物理设计时对系统配置变量的调整只是初步的,在系统运行时还要根据系统实际运行情况做进一步调整,以期切实改进系统性能。

2. 关系模式存取方法的选择

存取方法是用户存取数据库数据的方法和技术。存取方法指对文件采取的存取操作方法,即进行增加、删除、修改操作时如何快速地"存",进行检索查询操作时如何快速地"取"。一种文件组织可以采取多种存取方法进行访问,存取方法的选择将直接影响数据的存取速度和吞吐量,需要考虑的因素主要包括访问类型(指定属性值的查询还是属性值范围的查询)、访问时间、插入删除时间和空间开销等。

为了给用户提供快速高效的数据库共享系统,DBMS 需要提供支持多种存取方法的存取机制,使应用系统根据需要选择最佳性能的存取方法,实现对数据库的快速访问。

关系数据库常用的存取方法主要有索引方法、聚簇方法和 Hash 方法等。

1) 索引方法的选择

实现数据库快速访问的最有效方法是使用索引机制。索引机制指对数据库的数据表,根据查询需要,按照查询数据对应的关键属性,为数据表建立相应的用于快速检索的索引文件(索引文件是用于存储索引表的文件,索引表是由索引属性值与指向索引属性值所在数据页的物理地址的指针构成)。在执行查询操作时,先从索引文件中找到查询的元组在数据表中的地址,再根据这个地址,从数据表中直接取出元组数据。这种先查询索引文件,再从数据表中取值的检索机制称为索引机制。

索引是加到数据库内部的特殊数据结构,定义在数据表的基础之上,有助于无须检查所有记录而快速定位所需记录的一种辅助存储结构。几乎所有的关系数据库管理系统都建立有索引功能,选择索引方法实际上就是根据应用要求确定对关系的哪些属性列建立索引、建立多少个索引、哪些属性列建立组合索引、哪些索引建立唯一索引等。即索引的选择应考虑两个问题:一是对什么关系建立索引,二是选择哪个或哪些属性作为索引码。

选择索引方法的基本原则如下。

(1) 如果一个属性经常在查询条件中出现,则考虑在这个属性上建立索引;如果一组属性经常在查询条件中出现,则考虑在这组属性上建立联合(复合)索引。

(2) 如果一个属性经常作为最大值和最小值等聚集函数的参数,则考虑在这个属性上建立索引。

(3) 如果一个属性经常在连接操作的连接条件中出现,则考虑在这个属性上建立索引;同理,如果一组属性经常在连接操作的连接条件中出现,则考虑在这组属性上建立索引。

(4) 对经常需要执行查询、连接、统计操作而又记录较多的关系建立索引,经常进行插入、删除、修改操作或记录较少的关系和很少作为查询条件的列可避免建立索引。关系上定义的索引数要适当,并不是越多越好,因为系统维护索引要付出代价,更重要的是随着关系中数据的变化,索引需要维护更新以反映数据的变化。

B 树索引是最常用的存取方法,而 B+树基于 B 树做了改进,和 B 树索引的综合指标相

比，B+树索引在查询性能上更稳定、更高效。因此，B+树已经被广泛地应用到 DBMS 中。

注意：索引文件需要配合数据文件一起使用，才能进行快速检索，就像一本字典，要实现字的查询，那么字典索引与字典正文必须一起使用。其中，字典索引相当于索引表，字典正文相当于数据表。因此，索引文件单独使用没有任何意义。

2）聚簇方法的选择

在物理结构设计中，为了改善性能、提高处理效率，特别是为了提高某个属性或属性组的查询速度，把这个属性或属性组上具有相同值的元组集中存放在连续的物理块上的处理称为聚簇，这个属性或属性组称为聚簇码。

聚簇索引指按照关键属性对数据表建立索引时，数据表中的相应元组在磁盘上按照索引的排序方式进行存储，使索引的顺序与数据表中相应元组的物理顺序始终保持一致的索引过程。可以说，聚簇索引中的索引通常定义在聚簇码上，根据索引关键属性的值直接找到数据的物理存储位置，将相同聚簇码值的元组集中存放，减少访问磁盘的次数，从而达到快速检索数据的目的。

聚簇不仅适用于单个数据表，也适用于经常进行连接操作的多个数据表，即把多个连接表的元组按连接属性聚集存放，从而提高连接操作的效率。

因为数据记录只能有一种排序存储方式，所以一个数据表只能有一个聚簇索引，但一个数据库可以建立多个聚簇索引。选择聚簇方法就是确定需要建立多少个聚簇，确定每个聚簇中包括哪些属性。聚簇设计分两步进行。

（1）根据规则确定候选聚簇，设计候选聚簇的原则如下。

① 对经常在一起进行连接操作的关系可以建立聚簇。

② 如果一个关系的一组属性经常出现在相等、比较条件中，则此单个关系可建立聚簇。

③ 如果一个关系的一个（或一组）属性上的值重复率很高，则此单个关系可建立聚簇。

④ 如果关系的主要应用是通过聚簇码进行访问或连接，而其他属性访问关系的操作很少时，可以使用聚簇。尤其当 SQL 语句中包含与聚簇有关的 ORDER BY、GROUP BY、UNION、DISTINCT 等子句或短语时，使用聚簇特别有利，可以省去对结果集的排序操作。反之，当关系较少利用聚簇码操作时，最好不要使用聚簇。

（2）从候选聚簇中去除不必要的关系，检查候选聚簇，取消其中不必要关系的方法如下。

① 从聚簇中删除经常进行全表扫描的关系。

② 从聚簇中删除更新操作远多于连接操作的关系。

③ 不同的聚簇中可能包含相同的关系，一个关系可以在某一个聚簇中，但不能同时加入多个聚簇。要从多个聚簇方案（包括不建立聚簇）中选择一个较优的方案，其标准是在这个聚簇上运行各种事务的总代价最小。

（3）建立聚簇应注意的问题如下。

① 聚簇虽然提高了某些应用的性能，但是建立与维护聚簇的开销是相当大的。

② 对已有的关系建立聚簇，将导致关系中的元组移动其物理存储位置，这样会使关系上原有的索引无效，要想使用原索引就必须重建原有索引。

③ 当一个元组的聚簇码值改变时，该元组的存储位置也要做相应移动，所以聚簇码值

应当相对稳定,以减少修改聚簇码值所引起的维护开销。

需要注意的是,聚簇索引虽然可以提高数据查询的效率,但是会改变数据的物理存储位置,而且会导致数据表的原有索引失效,同时维护费用很大,因此需要谨慎使用。

3) Hash 方法的选择

Hash 方法又称散列方法。虽然索引技术能够提高数据的访问效率,但是索引的维护费用比较高,查询数据需要首先访问索引文件,才能在主文件中定位记录。基于散列技术的文件组织方式,可以不通过索引就能够进行数据访问。此外,还提供了构造散列索引的方法,使对数据的访问更加快速有效。

散列方法是一种基于散列函数和桶的检索模式,按照这种模式组织的文件称为散列文件,散列文件支持快速存取。如果用该方法存取文件,必须指定文件的一个或一组域为查询的关键字,该域常称为 Hash 域,然后定义一个 Hash 域上的函数,即散列函数,利用散列函数计算出存储数据的桶的地址,再根据桶地址访问桶中的数据。

桶是散列方法的基本存储单位,指在存储介质上用于存储多个元组的存储空间,可以是存储介质上一个或者多个存储块。散列函数是该方法的核心内容,实现索引关键属性值到桶地址值之间的映射。

选择散列方法的规则如下。

如果一个关系的属性主要出现在相等连接条件、相等比较条件中,而且满足下列两个条件之一,则该关系可以选择散列方法。

(1) 一个关系的大小可以预知,而且不变。

(2) 关系的大小动态改变,而且 DBMS 提供了动态散列方法。

每一种存取方法都可以实现关系中数据的快速检索。由于每一种存取方法都有其自身的优点和缺点,因此针对实际问题,对于关系是否建立索引、建立什么索引、建立多少个索引等的选择是一个复杂的问题。通常根据应用需求并结合索引机制自身的优点进行综合考虑,并最终选择合理的索引机制。

3. 物理结构的评价

物理设计过程中需要对时间效率、空间效率、维护代价和各种用户要求进行权衡,其结果可能会产生多种设计方案。数据库设计人员必须对这些方案进行详细地评价,从中选择一个较优的方案作为数据库的物理结构。

评价物理结构的方法完全依赖于选用的 DBMS,主要是从定量估算各种方案的存储空间、存取时间和维护代价入手,对估算结果进行权衡和比较,选择出一个较优、合理的物理结构。如果该结构不符合用户需求,则需要修改设计。

因此,为了更好地进行数据库的物理结构设计,数据库设计者必须了解选用的 DBMS 的存储管理技术。为了提高系统的性能,物理结构设计应该根据应用系统的处理要求,生成不同的物理存储文件。


视频讲解

3.5.3 物理结构设计案例

本案例数据库的物理结构设计主要针对逻辑模式的存储结构,主要任务是依据逻辑结构设计的结果,确定数据库模式在 DBMS 中存储的逻辑结构。由 3.4.4 节逻辑结构设计阶

段得到的高校图书管理数据库的关系数据库模式，在 SQL Server 2019 环境下，使用 T-SQL 定义图书管理数据库的三级模式结构，包括数据库名称，各关系模式的名称，属性的名称、数据类型、长度，该属性是否允许为空值、是否是主码、是否为索引项及约束条件等，确定视图名称、属性以及索引名称等。其中，名称的定义要符合标识符的命名规则，为了便于数据库的操作和维护，一般用英文单词代替，本书为了阅读方便，名称命名使用的是中文。表 3-5～表 3-8 详细列出了图书管理数据库中各关系模式的设计情况。

表 3-5 读者类别表

列 名	数据类型	长 度	列级约束	表级约束	备 注
类别编号	nvarchar	2	唯一，非空	主键	
类别名称	nvarchar	10	非空		
可借阅天数	tinyint	1	非空		
可借阅数量	tinyint	1	非空		
超期罚款额	smallmoney	4	非空		

表 3-6 读者表

列 名	数据类型	长 度	列级约束	表级约束	备 注
读者卡号	nvarchar	10	唯一，非空	主键	
姓名	nvarchar	16	非空		索引项(升序)
性别	nvarchar	1	非空		男，女
单位	nvarchar	30	非空		
办卡日期	date	3	非空		yyyy-mm-dd
卡状态	nvarchar	5	非空		
类别编号	nvarchar	2	非空	外键	

表 3-7 图书表

列 名	数据类型	长 度	列级约束	表级约束	备 注
图书编号	nvarchar	8	唯一，非空	主键	
书名	nvarchar	40	非空		索引项(升序)
类别	nvarchar	16	非空		
作者	nvarchar	16	非空		
出版社	nvarchar	20	非空		
出版日期	date	3	非空		
单价	smallmoney	4	非空		
库存数量	tinyint	1			

表 3-8 借阅表

列 名	数据类型	长 度	列级约束	表级约束	备 注
读者卡号	nvarchar	10	非空	主属性，外键	
图书编号	nvarchar	8	非空	主属性，外键	
借书日期	date		非空		yyyy-mm-dd
还书日期	date		非空		yyyy-mm-dd

3.6 数据库的实施

对数据库的物理设计进行初步评价以后,就可以进行数据库的实施了。数据库实施阶段的主要任务是根据数据库逻辑结构和物理结构设计的结果,在实际的计算机系统中建立数据库的结构、装载数据、测试程序、对数据库的应用系统进行试运行等。具体工作是:设计人员用 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设计和物理设计结果严格描述出来,使数据模型成为 DBMS 可以接受的源代码;再经过调试产生目标模式,完成建立定义数据库结构的工作;最后组织数据入库,并运行应用程序进行调试。

1. 建立数据库的结构

利用给定的 DBMS 提供的命令,建立数据库的模式、子模式和内模式。对关系数据库来说,就是创建数据库和建立数据库中包含的各个基本表、视图、索引。此部分内容将在第 6 章详细介绍。

2. 数据的装载和应用程序的编制调试

数据的装载是数据库实施阶段最主要的工作,一般数据库系统中,数据量都很大,而且来源部门中各个不同的单位,分散在各种不同的单据或原始凭证中,数据的组织方式、结构和格式都与新设计的数据库系统有相当的差距,数据载入就是要将各种源数据从各个局部应用中抽取出来,输入到计算机后再进行分类转换,综合成符合新设计的数据库结构的形式,最后输入数据库。因此,数据转换和组织数据入库工作是一件耗费大量人力、物力的工作。

为提高数据输入工作的效率和质量,应该针对具体的应用环境设计一个数据录入子系统,由计算机完成数据入库的任务。为了防止不正确的数据输入到数据库内,应当采用多种方法多次地对数据检验。现有的 DBMS 一般都提供不同的 DBMS 之间数据转换的工具,若原有系统是数据库系统,就可以利用新系统的数据转换的工具,先将原系统中的表转换成新系统中相同结构的临时表,再将这些表中的数据分类、转换,综合成符合新系统的数据模式,插入相应的表中。

数据库应用程序的设计应该与数据库设计同时进行,因此,在组织数据入库时,还要调试应用程序。

需要注意的是,装载数据时,一般是分期分批地组织数据入库,先输入小批量数据进行调试,等系统试运行结束基本合格后,再大批量输入数据,逐步增加数据量,逐步完成运行评价。

3. 数据库的试运行

在原有系统的部分数据输入到数据库后,就可以开始对数据库系统进行联合调试,从而进入到数据库的试运行阶段。其主要工作如下。

1) 测试应用程序功能

实际运行数据库应用程序,执行对数据库的各种操作,测试应用程序功能是否满足设计要求,如果应用程序的功能不能满足设计要求,则需要对应用程序部分进行修改、调整,直到达到设计要求为止。

2) 测试系统的性能指标

测试系统的性能指标,分析其是否符合设计目标。由于对数据库进行物理设计时考虑的性能指标是估计的,和实际系统运行有一定的差距,因此必须在试运行阶段实际测试和评

价系统性能指标。

在此阶段由于系统还不稳定，软、硬件故障随时都可能发生。同时，系统的操作人员对新系统还不熟悉，误操作也不可避免。因此，在数据库试运行时，应首先调试运行 DBMS 的恢复功能，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，尽量减少对数据库的破坏。

3.7 数据库的运行和维护

数据库试运行合格后，即可投入正式运行了，这标志着数据库开发工作基本完成。但是，由于应用环境在不断变化，数据库运行过程中物理存储也会不断变化，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。数据库的维护工作包括以下四项。

1）数据库的转储和恢复

数据库的转储和恢复是系统正式运行后最重要的维护工作之一。数据库管理员要针对不同的应用要求制定不同的转储计划，以保证一旦发生故障尽快将数据库恢复到某种一致的状态，并尽可能减少对数据库的破坏。

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。例如，有的数据原来是机密的，现在变成可以公开查询的，而新加入的数据又可能是机密的了。系统中用户的密级也会变化。这些都需要 DBA 根据实际情况修改原有的安全性控制。同样，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、分析和改造

在数据库运行过程中，监督系统运行、对监测数据进行分析并找出改进系统性能的方法是 DBA 的又一项重要任务。目前，有些 DBMS 产品提供了监测系统性能参数的工具，DBA 可以利用这些工具方便地得到系统运行过程中一系列性能参数的值。DBA 应仔细分析这些数据，判断当前系统运行状态是否是最佳，应当做哪些改进，如调整系统物理参数或对数据库进行重组织、重构造等。

4）数据库的重组与重构

数据库运行一段时间后，由于记录不断地增加、删除和修改，会使数据库的物理存储情况变坏，降低数据的存取效率，数据库的性能下降。这时，DBA 就要对数据库进行重组织或部分重组织（只对频繁增删的表进行重组织）。数据库的重组指在不改变数据库逻辑结构和物理结构的情况下，删除数据库存储文件的废弃空间及碎片空间的指针链，使数据库记录在物理上紧连。DBMS 一般都提供数据重组用的实用程序。在重组的过程中，按原设计要求重新安排存储位置、回收垃圾等，以提高系统性能。

数据库的重构指当数据库的逻辑结构不能满足当前数据处理的要求时，对数据库的模式和内模式进行修改。由于数据库应用环境发生变化，如增加了新的应用或新的实体、取消了某些应用、有的实体与实体间的联系发生了变化等，使原有的数据库设计不能满足新的需求，需要调整数据库的模式和内模式。又如，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。

一个好的数据库设计,不仅可以为用户提供所需的全部信息,而且必须提供准确、快速、安全的服务,数据库的管理和维护相对才会简单。对数据库应用系统来说,数据库是基础,只有把系统的数据库设计好,才可能实现一个实用的数据应用系统,否则,整个系统也是一个失败的系统。

习题 3

1. 选择题

(1) 用例图是用于数据库设计中(　　)阶段的工具。

A. 概要设计　　B. 逻辑设计　　C. 程序编码　　D. 需求分析

(2) 获取用户的需求后,面向对象分析的主要工作是建立问题领域的对象模型,对象模型的关键是确定(　　)。

A. 对象所属的类　　B. 对象的属性

C. 对象的操作　　D. 对象之间的关系

(3) 概念结构设计是通过对用户需求进行综合、归纳与抽象,形成一个独立于具体DBMS的(　　)。

A. 数据模型　　B. 概念模型　　C. 层次模型　　D. 关系模型

(4) 数据库设计人员和用户之间沟通信息的桥梁是(　　)。

A. 程序流程图　　B. 实体联系图　　C. 模块结构图　　D. 数据结构图

(5) 数据库设计的概念设计阶段,表示概念结构的常用方法和描述工具是(　　)。

A. 层次分析法和层次结构图　　B. 用例分析法和用例图

C. 实体联系方法　　D. 结构分析法和模块结构图

(6) 在关系数据库设计中,设计关系模式是数据库设计中(　　)阶段的任务。

A. 逻辑设计阶段　　B. 概念设计阶段

C. 物理设计阶段　　D. 需求分析阶段

(7) 在数据库设计中,将E-R图转换成关系数据模型的过程属于(　　)。

A. 需求分析阶段　B. 逻辑设计阶段　C. 概念设计阶段　D. 物理设计阶段

(8) 在关系数据库设计中,对关系进行规范化处理,使关系达到一定的范式,如达到3NF,这是(　　)阶段的任务。

A. 需求分析阶段　B. 概念设计阶段　C. 物理设计阶段　D. 逻辑设计阶段

(9) 设计子模式属于数据库设计的(　　)。

A. 需求分析阶段　B. 概念设计阶段　C. 物理设计阶段　D. 逻辑设计阶段

(10) 数据库设计可划分为六个阶段,每个阶段都有自己的设计内容,“为哪些关系,在哪些属性上建什么样的索引”这一设计内容应该属于(　　)设计阶段。

A. 概念结构设计　B. 逻辑结构设计　C. 物理结构设计　D. 全局结构设计

(11) 数据库设计中,确定数据库存储结构,即确定关系、索引、聚簇、日志、备份等数据

的存储安排和存储结构，这是数据库设计的(　　)。

A. 需求分析阶段　B. 逻辑设计阶段　C. 概念设计阶段　D. 物理设计阶段

(12) 数据库物理设计完成后，进入数据库实施阶段，(　　)一般不属于实施阶段的工作。

A. 建立库结构　B. 系统调试　C. 加载数据　D. 扩充功能

(13) 在数据库设计中，子类与超类存在着(　　)。

A. 相容性联系　B. 调用的联系　C. 继承性的联系　D. 一致性联系

(14) 当同一个实体集内部实体之间存在着一个 m：n 的联系时，根据 E-R 模型转换成关系模型的规则，转换成关系的数目为(　　)。

A. 1　B. 2　C. 3　D. 4

(15) 在 E-R 模型中，如果有四个不同的实体型，三个 m：n 联系，根据 E-R 模型转换为关系模型的规则，转换为关系的数目是(　　)。

A. 4　B. 5　C. 6　D. 7

(16) 从 E-R 图导出关系模型时，如果实体间的联系是 m：n 的，下列说法中正确的是(　　)。

A. 将 n 方码和联系的属性纳 m 方的属性中

B. 将 m 方码和联系的属性纳入 n 方的属性中

C. 增加一个关系表示联系，其中纳入 m 方和 n 方的码

D. 在 m 方属性和 n 方属性中均增加一个表示级别的属性

(17) 下列有关 E-R 模型向关系模型转换的叙述中，不正确的是(　　)。

A. 一个实体型转换为一个关系模式

B. 一个 1：1 联系可以转换为一个独立的关系模式，也可以与联系的任意一端实体所对应的关系模式合并

C. 一个 1：n 联系可以转换为一个独立的关系模式，也可以与联系的任意一端实体所对应的关系模式合并

D. 一个 m：n 联系转换为一个关系模式

(18) 对数据库的物理设计优劣评价的重点(　　)。

A. 时空效率　B. 动态和静态性能

C. 用户界面的友好性　D. 成本和效益

(19) 一个仓库可以存放多种零件，每种零件只能存放在一个仓库中，仓库和零件之间为(　　)的联系。

A. 一对一　B. 一对多　C. 多对多　D. 多对一

(20) 当一个客户向同一个银行申请多笔贷款时，则贷款联系集转化关系模式的主码应该是(　　)。

A. 客户编号　B. 客户编号＋银行编号

C. 银行编号　D. 客户编号＋银行编号＋贷款日期

2. 简答题

(1) 简述数据库设计的主要步骤和每一个阶段的具体任务。

(2) 简述面向对象方法需求分析阶段的主要工作。

(3) 试述数据库概念结构设计的重要性和设计步骤。

(4) 什么是 E-R 图,构成 E-R 图的基本要素是什么?

(5) 局部 E-R 图合并为全局 E-R 图时主要存在哪些冲突?如何解决?

(6) 什么是关系数据库的逻辑结构设计?试述其设计步骤。

(7) 影响数据库物理结构设计的主要因素有哪些?

3. 设计题

(1) 一个企业的职工信息管理数据库要求提供下述服务。

① 可随时查询各个部门的部门编号、部门名称、部门负责人等信息,以及现有职工的工号、姓名、性别、出生日期、所在部门等信息。并约定一个部门有多名职工,一名职工来自一个部门,一个部门只有一个负责人。

② 可随时查询企业的岗位设置情况,包括岗位编号、岗位名称、基本工资等,并约定一个岗位有多名职工,一名职工只能属于一个岗位。

③ 该数据库还可描述职工参加培训课程的情况,包括培训课程的课程号、课程名、学时以及培训成绩。并约定,在不冲突的情况下,一名职工可培训多门课程,同一门课程可同时安排多名职工进行培训。

根据以上情况,完成如下工作:

① 试为该企业的职工信息管理系统设计一个 E-R 模型,并在图上注明属性、实体主标识符、联系类型。

② 设计此数据库的关系模式,要求满足 3NF 范式以上,并标识主码和外码。

(2) 一个学校的师资管理数据库要求提供下述服务。

① 可随时查询各个院系的系编号、系名称、系负责人等信息,以及现有教师的教师工号、教师姓名、年龄、职称、所在教研室、聘用日期等信息。并约定一个系有多名教师,一名教师从属于一个系,一个系只有一个负责人。

② 可随时查询各个教师的任课情况,包括所授课程的课程编号、课程名、学分。并约定一个教师可教多门课程,每门课程可由多位教师担任。

③ 该数据库还可描述教师研究课题的情况,包括课题的课题号、课题名、计划完成日期、实际完成日期及研究课题的最终成果。并约定,一个教师可研究多个课题,一个课题可由多个教师完成。

根据以上情况,完成如下工作:

① 试为该学校师资管理部门设计一个 E-R 模型,并在图上注明属性、主标识符、联系类型。

② 将 E-R 模型转换成满足 3NF 的关系模式,并标识主码和外码。

(3) 某医院病房需要设计一个数据库系统来管理该医院病房的业务信息,涉及如下信息。

科室:科名,科地址,科电话,医生姓名;

病房:病房号,床位号,所属科室名;

医生:工作证号,姓名,职称,年龄,所属科室名;

病人:病历号,姓名,性别,诊断,主管医生,病房号。

其中,一个科室有多个病房、多名医生;一个病房只能属于一个科室;一名医生只能属

于一个科室，但可负责多个病人的诊治；一个病人的主管医生只有一个。

根据以上情况，完成如下工作：

① 设计该系统的 E-R 模型，并在图上注明属性、联系类型、主标识符。

② 将 E-R 模型转换成满足 3NF 的关系模式，并标识主码和外码。

(4) 在一个软件开发公司中，有来自不同部门的程序员，他们共同完成软件整体开发。现要求从该软件开发公司中的数据库中提供下述服务。

① 可随时查询各个部门的编号、名称、部门负责人。一个部门有多个程序员，一个程序员只能属于一个部门。

② 可随时查询各个办公室情况，包括办公室编号、地点、电话。一个部门可有多个办公室，一个办公室只能属于一个部门。

③ 可随时查询每个程序员的信息，包括程序员编号、姓名、年龄、性别、职称、入职时间。

④ 该数据库还可提供软件项目的信息，包括项目编号、项目名称、版权等信息。在程序设计工作中，一位程序员可以参与多个项目，一个项目是由多位程序员共同完成。对每位程序员参与某个项目的设计要记录其开始时间及结束时间。

根据以上情况，完成如下工作：

① 试为该数据库设计一个 E-R 模型，并在图上注明属性、联系类型、主标识符。

② 将 E-R 模型转换成满足 3NF 的关系模式，并标识主码和外码。

(5) 有一田径运动会组委会需建立数据库系统进行管理，要求反映如下信息。

裁判员数据：姓名，年龄，性别，等级；

运动员数据：号码，姓名，年龄，性别，比赛成绩；

运动项目数据：名称，比赛时间，比赛地点，最高纪录。

其中，每个裁判员只能裁判一个运动项目。每个运动员可以参加多个运动项目，取得不同运动项目的比赛成绩。

根据以上情况，完成如下工作：

① 设计该系统的 E-R 模型，并在图上注明属性、主标识符、联系类型。

② 将 E-R 模型转换成满足 3NF 的关系模式，并标识主码和外码。

(6) 一个学校的社团管理数据库要求反映如下信息。

学生的属性有：学号，姓名，出生年月，班号，宿舍号；

班级的属性有：班号，班长，专业名，人数；

学院的属性有：学院编号，学院名称，办公地点；

学生社团的属性有：社团名称，成立年份，地点；

其中，一个学院有若干个班级，每个班级有若干学生。每个学生可以参加若干社团，每个社团有若干学生。学生参加某个社团有一个入会年份。

根据以上情况，完成如下工作。

① 设计该系统的 E-R 模型，并在图上注明属性、主标识符、联系类型；

② 将 E-R 模型转换成满足 3NF 的关系模式，并标识主码和外码。