第5章 数据库设计

数据库设计就是从用户的需求出发构造和设计数据库结构的过程，也就是为特定的应用环境构造最优的数据模型。本章主要介绍数据库设计的过程与方法，重点讨论基于 E-R 模型的数据库的概念结构设计和基于关系型数据库的逻辑结构设计问题。

5.1 数据库设计的步骤

数据库设计过程可分为需求分析、概念结构设计、逻辑结构设计、物理结构设计、数据库实施、数据库运行与维护 6 个阶段。其设计过程和步骤如图 5.1 所示。

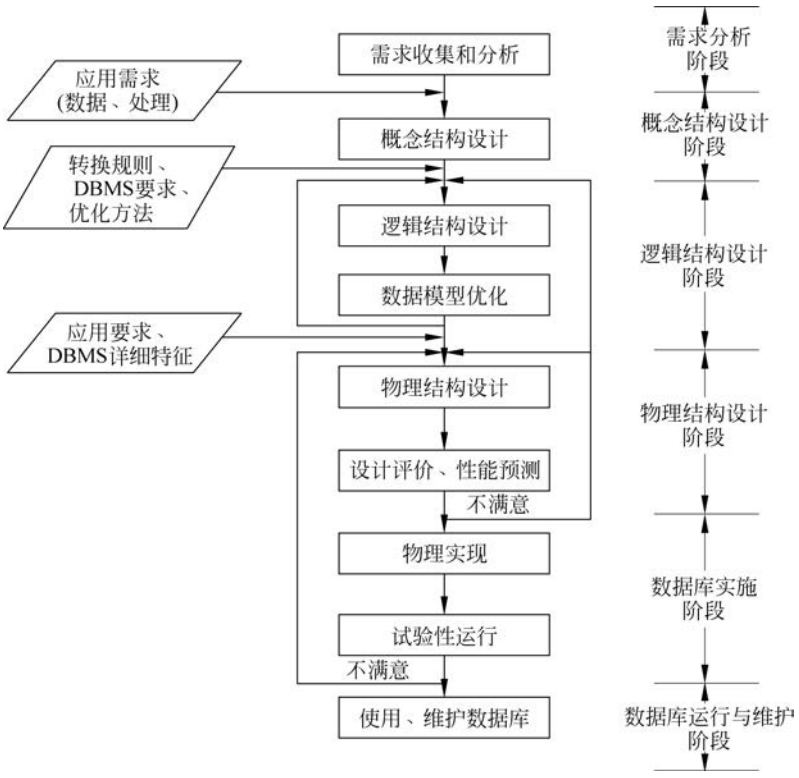

图 5.1 数据库设计过程和步骤

1. 需求分析阶段

需求分析是数据库设计的第一步，是最重要最耗时的一个环节。需求分析就是要准确地了解并分析用户对系统的需求（包括数据与处理），弄清系统要达到的目标和实现的功能。需求分析是否做得充分与准确，决定着在其上构建数据库“大厦”的速度与质量。

2. 概念结构设计阶段

概念结构设计是整个数据库设计的关键。概念结构设计通过综合、归纳与抽象用户需求，形成一个独立于任何具体 DBMS 的概念模型。

3. 逻辑结构设计阶段

逻辑结构设计是将概念结构转换成某个 DBMS 所支持的逻辑数据模型，如关系逻辑模型，并对其进行优化。

4. 物理结构设计阶段

数据库物理结构设计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。

5. 数据库实施阶段

在该阶段，系统设计人员运用 DBMS 提供的数据库语言（如 SQL）和宿主语言，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. 数据库运行与维护阶段

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。

5.2 需求分析

需求分析是数据库设计的第一步，主要是需要准确了解并分析用户对系统的需要和要求，弄清系统要达到的目标和实现的功能。简单地讲就是分析用户对数据和处理的要求。在需求分析阶段，系统分析员将分析结果用数据流图（DFD）和数据字典（DD）表示出来。需求分析的结果是否能够准确地反映用户的实际要求，将直接影响到后面的各个阶段的设计，并影响到系统的设计是否合理与实用。

5.2.1 需求分析的任务

需求分析的主要任务是详细调查现实世界中要处理的对象，充分了解原系统（手工系统或计算机系统）的概况和发展前景，明确用户的各种需求，收集支持系统目标的基础数据及

其处理方法,确定新系统的功能和边界。

调查是系统需求分析的重要手段,只有通过对用户的调查研究,才能得出需要的信息。调查的重点是“数据”和“处理”,其目的是获得数据库所需数据和数据处理要求。调查的内容包括以下 3 项。

1. 数据库中的信息内容

数据库中的信息内容是指数据库中需存储哪些数据,它包括用户将数据库中直接获得或间接导出的信息的内容和性质。

2. 数据处理内容

数据处理内容包括:用户要完成什么数据处理功能;用户对数据处理响应时间的要求;数据处理的工作方式。

3. 数据安全性和完整性要求

数据安全性和完整性要求主要是指:数据的保密措施和存取控制要求;数据自身的或数据间的约束限制。

5.2.2 需求分析的方法

要进行需求分析,应当先对用户进行充分的调查,弄清楚他们的实际要求,与用户达成共识,然后再分析和表达这些需求。下面是调查用户需求的具体步骤。

1. 了解现实世界的组织结构情况

包括了解所涉及的行政组织结构,弄清楚所设计的数据库系统与哪些部门相关,各部门的职责是什么。

2. 了解相关部门的业务活动情况

弄清了与数据库系统相关的部门后,就要深入这些部门了解其业务活动情况。通过调查了解:各部门需要输入和使用什么数据;各部门如何加工处理这些数据;各部门需要输出什么信息;输出到什么部门;输出数据格式是什么。

3. 确定新系统的边界

对前面调查结果进行初步分析后,要确定出数据库系统的边界。即搞清楚:哪些功能由计算机完成;哪些功能将来准备让计算机完成;哪些功能或活动由人工完成。由计算机完成的功能就是新系统应该实现的功能。

在调查过程中,可以根据不同的问题和条件,使用不同的调查方法。常用的调查方法有以下 6 种:

(1) 跟班作业。数据库设计人员亲身参加业务工作,了解业务活动情况。

(2) 开调查会。数据库设计人员通过与用户座谈来了解业务活动情况及用户需求。

(3) 请专业人员介绍。

(4) 询问。对某些调查中的问题,可以找专人咨询。

(5) 设计调查表请用户填写。

(6) 查阅现实的数据记录。查阅与原系统有关的数据记录。

调查了解用户需求后,还需要进一步分析和表达用户的需求。分析和表达用户需求的方法有很多,结构化分析方法是一种简单实用的方法。结构化分析方法从最上层的系统组织机构入手,采用自顶向下、逐层分解的方式分析系统。在进行逐层分解的同时,所用的数据也逐级分解,形成若干层次的数据流图。

数据流图表达了数据和处理过程的关系。在结构化分析方法中,处理过程的处理逻辑通常使用判定表或判定树来描述。而系统中的数据则使用数据字典来描述。有关数据流图和数据字典的相关概念内容超出本课程范围,请参考"软件工程"课程的内容。

5.3 概念结构设计

数据库的概念结构设计是将系统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。概念结构设计的结果是数据库的概念模型。它是整个数据库设计的关键。

概念结构能够真实、充分地反映现实世界,包括事物和事物之间的联系,是对现实世界模拟的一个真实模型;概念结构易于理解、易于更改,容易对概念模型进行修改与扩充;更重要的是概念结构容易向各种数据模型转换(如关系数据模型)。概念结构是各种数据模型的共同基础和抽象表达,它比数据模型更独立于计算机、更加抽象,从而更加稳定可靠。

描述概念模型的有力工具是 E-R 模型,也称为 E-R 图。相关 E-R 模型的基本概念请参看 1.2.2 节。本节主要是利用 E-R 模型来描述数据库的概念结构设计。

5.3.1 概念结构设计的步骤

1. 概念结构设计的方法

概念结构设计通常有 4 种方法。

(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,最终得到一个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。这是经常采用的方法,即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) 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,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) 混合策略的设计方法。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中设计的各局部概念结构。

最常用的策略是自底向上方法,即自顶向下地进行需求分析,然后再自底向上地设计数据库概念结构,其方法如图 5.2 所示。

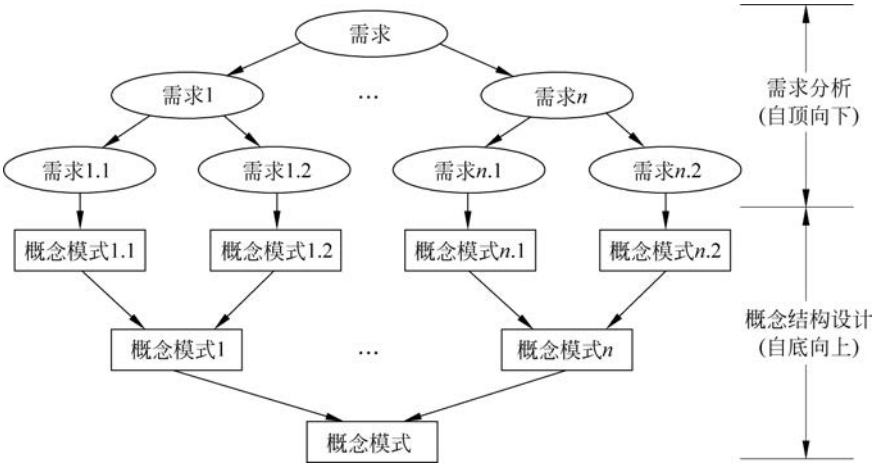

图 5.2 自顶向下需求分析与自底向上概念结构设计的方法

2. 概念结构设计的步骤

自底向上的概念结构设计方法通常分为两步：第一步是抽象数据并设计局部视图，第二步是集成局部视图，从而得到全局的概念结构。这种自底向上概念结构设计步骤如图 5.3 所示。

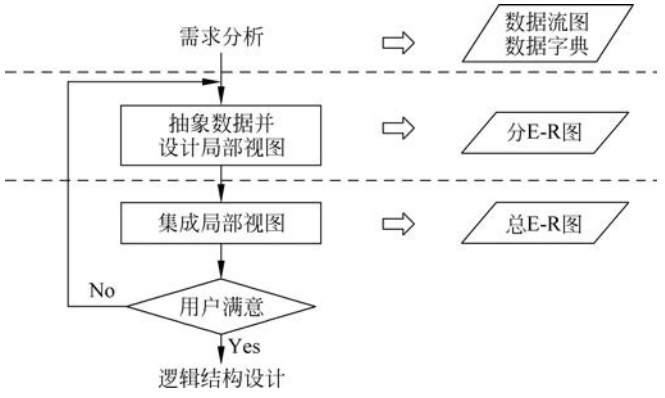

图 5.3 自底向上概念结构设计步骤

5.3.2 设计局部的 E-R 模型

概念结构是对现实世界的一种抽象表达。抽象就是抽取现实世界中的人、事、物和概念的共同特性，忽略非本质的细节，并把这些共同特性用各种概念精确地加以描述，形成某种模型。

1. 3 种数据抽象方法

一般数据抽象有 3 种基本方法，分别是分类、聚集和概括。利用数据抽象方法可以对现实世界抽象，得出概念模型的实体集及属性。

1) 分类

分类就是定义某一类概念作为现实世界中一组对象的类型，这些对象具有某些共同的

特性和行为。分类抽象了对象值和型之间的“成员(is member of)”的语义。在 E-R 模型中的实体型就是分类抽象。例如,张晨是学生,表示张晨是学生中的一员,具有学生们共同的特性和行为:在某班学习某专业知识,并选修某些课程。图 5.4 是学生分类示意图。

2) 聚集

聚集是定义某类型的组成部分,它抽象了对象内部类型和对象内部成分之间的“组成部分(is part of)”的语义。在 E-R 模型中,若干属性聚集组成了实体型。例如,把实体集“学生”的“学号”“姓名”“专业”等属性聚集为实体型“学生”,如图 5.5 所示。

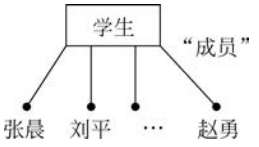

图 5.4　学生分类示意图

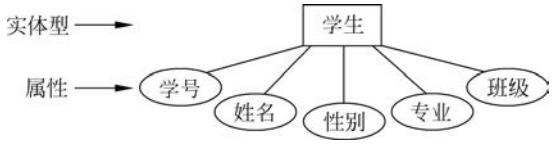

图 5.5　学生属性聚集实例

事实上现实世界的事物是复杂的,某些类型的组成部分可能仍然是一个聚集,这是一种更复杂的聚集。即某一类型的成分仍是一个聚集,如图 5.6 所示。

3) 概括

概括定义了类型之间的一种子集联系,它抽象了类型之间的“所属(is subset of)”的语义。例如,学生是一个实体型,本科生、研究生也是实体型,而且本科生和研究生都是学生的子集。把学生称为超类(Superclass),本科生、研究生称为学生的子类(Subclass)。

基本 E-R 模型中不支持概括抽象,为了表达这种特殊的抽象概念,本书引入了增强型的实体联系(Enhanced Entity-Relationship,EER)模型,扩充基本 E-R 模型的表达内容。在 E-R 模型中允许定义超类实体型和子类实体型,使用双竖边的矩形框表示子类,用直线加小圆圈表示超类与子类的联系,如图 5.7 所表示的学生和本科生、研究生之间的概括抽象联系。

图 5.6　复杂的聚集

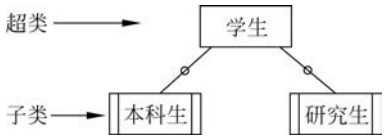

图 5.7　概括

概括的一个重要性质是继承性。超类与子类之间具有继承性的特点,即子类实体继承超类实体中定义的所有属性抽象。例如,本科生、研究生继承了学生类型的属性,当然本科生或研究生也可以有属于自己的某些特殊属性。又例如,技术人员和干部是职工的子类,即技术人员和干部都具有职工的属性,也可有自己的特殊属性。

2. 设计局部 E-R 图

概念结构设计是利用抽象机制对需求分析阶段收集到的数据进行分类、组织(聚集),形成实体集、实体的属性和标识实体的主关键字,确定实体集之间的联系类型(一对一、一对多、多对多联系),从而设计分 E-R 图。下面讲述设计分 E-R 图的具体做法。

1) 选择局部应用

选择局部应用就是根据系统的具体情况,在多层的数据流图中选择一个适当层次的数据流图,作为设计分 E-R 图的出发点,并使数据流图中的每个部分都对应一个局部应用。

选择局部应用之后,就可以对每个局部应用逐步设计分 E-R 图。

2) 逐步设计分 E-R 图

设计某层分 E-R 图前,局部应用的数据流图应该已经设计好,其所涉及的数据应该已经收集在对应的数据字典中。设计分 E-R 图时,需要根据局部应用的数据流图中标定的实体集、属性和主关键字,并结合数据字典中的相关描述内容,确定 E-R 图中的实体、实体之间的联系及类型。

实际上,实体和属性之间并没有可以截然划分的界限。但是,在现实世界中具体的应用环境常常对实体和属性已经作了大体的自然划分。在数据字典中的"数据结构""数据流""数据存储"都是若干属性的聚合,它体现了自然划分的意义。设计 E-R 图时,可以先从自然划分的内容出发定义得到最初的 E-R 图,再进行必要的调整。

为了简化 E-R 图,在调整中应当遵循的一条原则:现实世界的事物能作为属性对待的尽量作为属性对待。而实体与属性之间并没有形式上的截然区分,但可依据两条基本准则:

- "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。
- "属性"不能与其他实体具有联系,即 E-R 图中所表示的联系必须是实体间的联系,而不能有属性与实体之间的联系。

凡满足上述两条基本准则的事物,一般均可作为属性对待。但有时在一些特殊复杂的应用环境下,"属性"是否可分割或细分难以解决,必须结合具体的应用环境来综合判断。在具体分析问题时,可通过对属性进行仔细的划分与分类来逐步判断。下面探讨属性的分类类别,并处理这些属性。

(1) 简单属性和复合属性。

简单属性是不可再分的属性,如性别、年龄。复合属性可以再划分为更小的部分。例如,地址属性可再分解为省份、城市、街道和邮编等子属性,而街道又可分为街道名、门牌号两个子属性。对于简单属性可以直接当成某实体的属性,但对于复合属性是当成属性还是实体,还必须结合具体情况而定。若需要进一步表达其信息内容,则需要细化复合属性的组成信息,从而需要当成实体(包括子属性)。

例如,职工是一个实体,职工号、姓名、年龄和职称是职工的属性。如果职称没有与工资、福利挂钩,就没有必要进一步描述的特性,则职称可作为职工实体集的一个属性对待。如果不同的职称有着不同的工资、住房标准和不同的附加福利,则职称作为一个实体来考虑就比较合适。如图 5.8 所示是"职称"上升为实体,其结果是综合考虑第一条基本准则与复合属性的表达。

再如,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的工作,则根据上述第一条准则可知,病房应作为一个实体,如图 5.9 所示。

(2) 单值属性和多值属性。

单值属性是指同一实体的属性只能取一个值。例如,同一个员工只能有一个性别,性别就是员工实体的一个单值属性。多值属性是指同一个实体的某些属性可能对应一组值,比如一个员工可能有多个电话号码。对于单值属性在数据库中表达是必需的,而多值属性会带给数据库产生冗余数据,也会造成数据异常、数据不一致和完整性等问题。因此需要对多

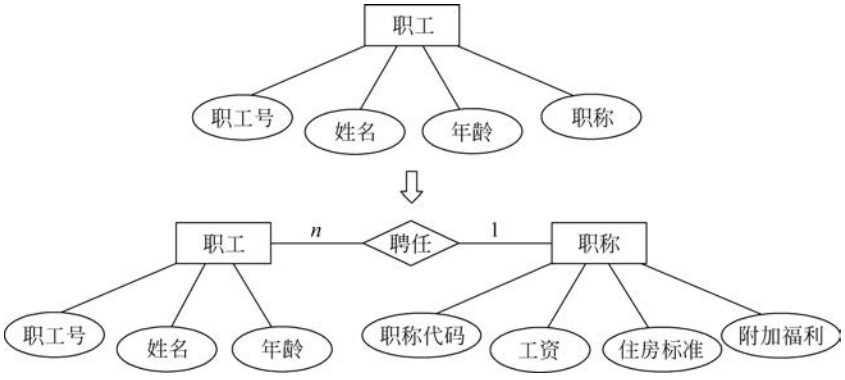

图 5.8　“职称”由属性上升为实体

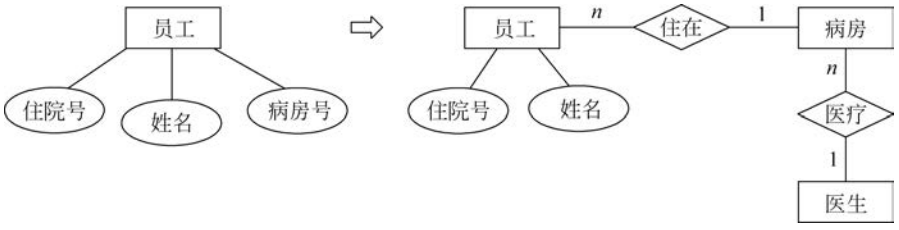

图 5.9　病房作为一个实体

值属性进行变换，有以下两种变换方法。

① 将原来的多值属性用新的单值属性表示。

例如，员工的联系电话可用办公电话、移动电话等进行分解，分解后的员工的结构，如图 5.10 所示。

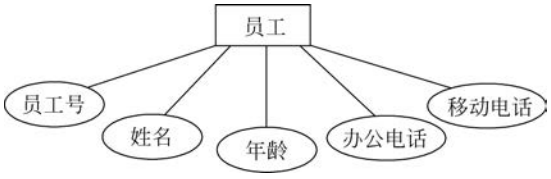

图 5.10　员工属性(多值属性变换表示——新增属性)

② 将原来的多值属性用一个新的实体型表示。

对于多值属性也可以用一个新的实体型来表示，这个新的实体型和原来的实体型之间是一对多联系，新的实体依赖原来的实体而存在，因此称新的实体为弱实体。在 E-R 图中，弱实体用双线矩形框表示，与弱实体相关的联系用双菱形框表示。如图 5.11 所示为使用弱实体表示的员工的联系电话。

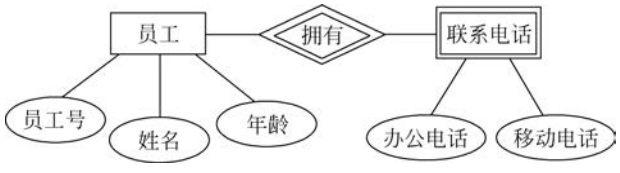

图 5.11　员工属性(多值属性变换表示——弱实体)

5.3.3 设计全局的 E-R 模型

各子系统的分 E-R 图设计好后,下一步就可以将所有的分 E-R 图综合成一个系统的总 E-R 图,即设计全局 E-R 图模型。设计全局 E-R 图有两种方法:一种方法是多个分 E-R 图一次集成为总 E-R 图,如图 5.12(a)所示;另一种方法是逐步集成,用累加的方法一次集成两个分 E-R 图,如图 5.12(b)所示。

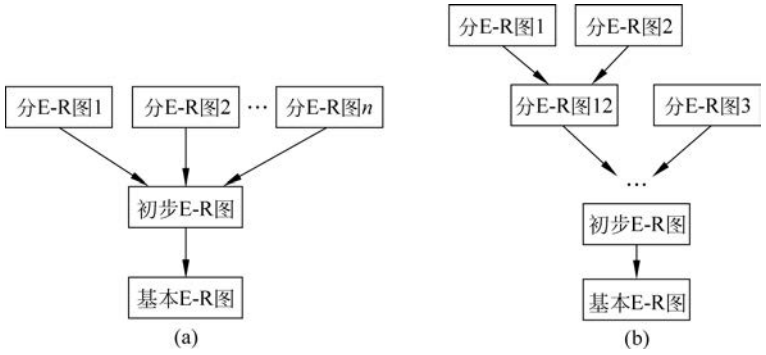

图 5.12 全局 E-R 图模型设计的两种方式

第一种方法将多个分 E-R 图一次集成比较复杂,做起来难度较大;第二种方法逐步集成,由于每次只集成两个分 E-R 图,因而可以有效地降低复杂度。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 E-R 图。进行 E-R 图合并时,要解决各分 E-R 图之间的冲突问题,并将各分 E-R 图合并起来生成初步 E-R 图。

(2) 修改和重构初步 E-R 图。修改和重构初步 E-R 图的目的是要消除不必要的实体集冗余和联系冗余,生成基本 E-R 图。

1. 合并分 E-R 图,生成初步 E-R 图

由于各个局部应用所面对的问题是不同的,且通常是由不同的设计人员进行不同局部的 E-R 图设计,这样导致各个分 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。因此合并分 E-R 图时,并不能简单地将各个分 E-R 图画在一起,而是必须先消除各个分 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。合理消除各个分 E-R 图的冲突是进行合并的主要工作和关键所在。

各分 E-R 图之间的冲突主要有 3 类:属性冲突、命名冲突和结构冲突。

1) 属性冲突

属性冲突有两种情况:属性域冲突和属性取值单位冲突。

(1) 属性域冲突,即属性值的类型、取值范围或取值集合不同。例如,对于零件号属性,不同部门可能会采用不同的编码形式,且定义的类型也各不相同,有的部门把它定义为整型,有的则定义为字符型。又如关于年龄,某些部门以出生日期来表示,另一些部门用整数表示职工年龄。这些属性域冲突需要各个部门之间协商解决。

（2）属性取值单位冲突。例如，对于零件的重量，不同部门可能分别用公斤、斤或千克来表示，结果造成数据统计错误。这类冲突同样需要各部门协商解决。

2）命名冲突

命名冲突主要有两种：同名异义冲突和异名同义冲突。

（1）同名异义冲突，即不同意义的对象在不同的局部应用中使用相同的名字。

（2）异名同义冲突，即意义相同的对象在不同的局部应用中有不同的名字。例如，对于科研项目，财务科称为项目，科研处称为课题，生产管理处称为工程。

命名冲突可能发生在实体或联系上，也可能发生在属性一级上。其中，属性的命名冲突更常见。处理命名冲突同样需要通过协商讨论加以解决。

3）结构冲突

结构冲突存在3种情况。

（1）同一对象在不同应用中具有不同的抽象。例如，职工在某局部应用中被当成实体对象，而在另一局部应用中被当成属性对待。解决这种冲突的方法通常是把属性变换为实体或把实体变换为属性，使同一对象具有相同的抽象。

（2）同一实体对象在不同分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顺序不完全相同。这类冲突是由于不同的局部应用所关心的实体对象的不同侧面而造成的。解决方法是使该实体对象的属性取各分E-R图中属性的并集，再适当调整属性的次序，使之兼顾各种局部应用。

（3）实体之间的联系在不同的分E-R图中为不同的类型。此类冲突解决方法是根据应用的语义对实体联系的类型进行综合或调整。设有实体集E1、E2和E3；在一个分E-R图中E1和E2是多对多联系，在另一分E-R图中E1和E2则又是一对多联系，这就是联系类型不同的情况；在某一个E-R图中E1和E2发生联系，而在另一个E-R图中E1、E2和E3三者之间发生联系，这就是联系涉及的对象不同的情况。

解决结构冲突的方法是根据应用的语义对实体联系的类型进行综合或调整。如图5.13所示就是一个综合E-R图的实例。一个分E-R图中零件与产品之间的“构成”联系是多对多联系，另一个分E-R图中产品、零件与供应商三者之间存在多对多的“供应”联系，显然这两个联系相互不包含，合并时需要把它们综合起来表示。

2. 消除不必要的冗余，设计基本E-R图

在初步E-R图中可能存在冗余的数据和实体间冗余的联系。冗余数据就是可由基本数据导出的数据。而冗余联系则是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，应当加以消除。消除了冗余的初步E-R图就称为基本E-R图。消除冗余的主要方法是用分析方法消除冗余和用规范化理论消除冗余。

1）用分析方法消除冗余

分析方法是消除冗余的主要方法。分析方法消除冗余是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。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，在数据库中设计了一些数据冗余或冗余联系。也就是说，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要求来确定。如果希望存在某些冗余，则应该把数据字典中

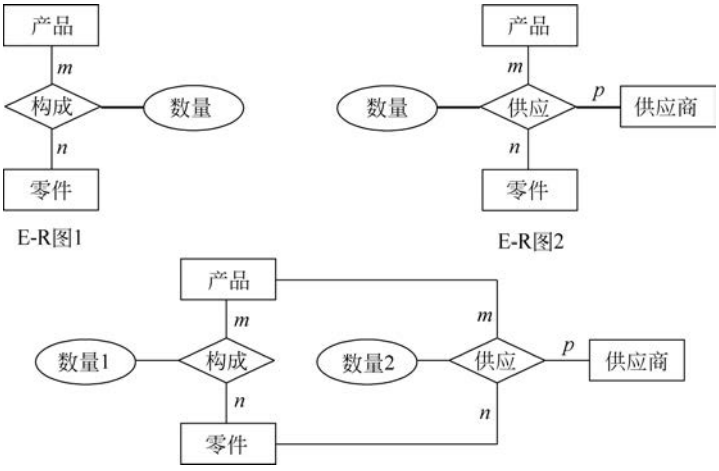

图 5.13 合并两个分 E-R 图时的综合

数据关联的说明作为完整性约束条件。

例如,在图 5.14 中,如果 $Q_3=Q_1\times Q_2$,且 $Q_4=\sum Q_5$,则 Q_3 和 Q_4 是冗余数据,因此 Q_3 和 Q_4 可以被消去。而消去 Q_3,产品与材料间的多对多的冗余联系也应当被消去。但若物种部门经常需要查询各种材料的库存量,如果每次都要查询每个仓库中此种材料的库存,再求和,则查询效率非常低下,因此应保留 Q_4,同时把"$Q_4=\sum Q_5$"定义为 Q_4 的完整性约束条件。每当 Q_5 被更新,就会触发完整性检查例程,以便对 Q_4 做相应的修改。

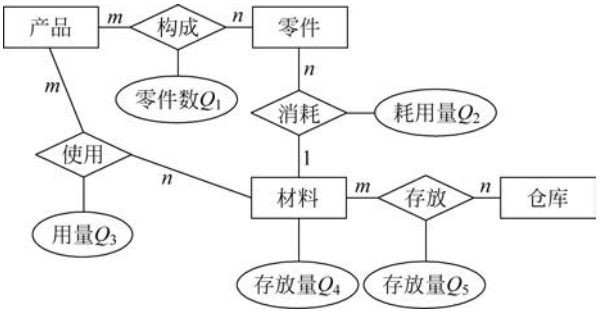

图 5.14 消除冗余的实例

2) 用规范化理论消除冗余

规范化理论中,函数依赖的概念提供了消除冗余的形式化工具。具体方法描述如下。

(1) 确定分 E-R 图实体之间的数据依赖。实体之间的一对一、一对多、多对多的联系可以用实体的主关键字之间的函数依赖来表示。

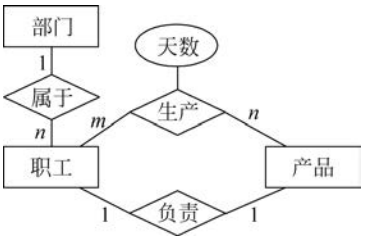

图 5.15 人事管理系统的分 E-R 图

如图 5.15 中,部门和职工之间一对多的联系可表示为职工号→部门号;职工和产品之间多对多的联系可表示为(职工号,产品号)→工作天数。这样可以得到函数依赖集 F_L。

(2) 求函数依赖集 F_L 的最小覆盖 G_L,差集为 $D=F_L-G_L$。逐一考察差集 D 中的函数依赖,确定是否是

冗余的联系，如果是冗余的联系则删除。由于规范化理论受泛关系假设的限制，因此应注意两个问题。

① 冗余的联系一定在差集 D 中，而 D 中的联系不一定是冗余的。

② 当实体之间存在多种联系时，需要将实体之间的联系在形式上加以区分。如图 5.15 中，部门和职工之间可能有另一个一对一的联系，则可表示为：负责人.职工号→部门号，部门号→负责人.职工号。

5.4 逻辑结构设计

数据库的概念结构设计的结果是使用 E-R 图来表示的，它是一种用户数据要求的图形化形式。如前所述，E-R 图独立于任何数据模型，也独立于任何一个 DBMS。逻辑结构设计的任务就是把概念模型结构转换成某个具体的 DBMS 所支持的数据模型。

理论上设计数据库逻辑结构的步骤应该是：首先选择最适合的数据模型，并按转换规则将概念模型转换为所选定的数据模型，然后从支持选定的数据模型的各个 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。但实际情况通常先给定了计算机环境和 DBMS，再进行数据库逻辑结构设计。因此在概念模型向逻辑模型转换时，就要考虑适合给定的 DBMS 的问题。

当前 DBMS 一般只支持关系、网状、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。

通常将概念模型转换为逻辑模型的过程分为 3 步进行，如图 5.16 所示为概念模型向逻辑模型转换的步骤。

(1) 把概念结构模型转换为一般的数据模型，如关系、网状、层次模型。

(2) 将一般的数据模型转换成特定的 DBMS 所支持的数据模型。

(3) 通过优化方法将数据模型进行优化。

图 5.16 逻辑结构设计的 3 个步骤

当前大多数设计的数据库应用系统都采用支持关系数据模型的 RDBMS，因此本节只介绍 E-R 图转换为关系数据模型的转换原则与方法。

5.4.1 E-R 模型向关系模型的转换

将 E-R 图转换成关系模型需要解决两个问题：一个是如何将实体型和实体型间的联系转换为关系模式；另一个是如何确定这些关系模式的属性和主关键字。关系模型的逻辑结构是一组关系模式，而 E-R 图表示的概念模型则是由实体型、实体的属性和实体型间的联系 3 个要素组成。将 E-R 图转换为关系模型实际上就是需要将实体型、实体的属性和实体型之间的联系转换为相应的关系模式。下面是概念模型转换为关系模型的基本转换规则。

1. 实体型的转换规则

概念模型中的一个实体型转换为关系模型中的一个关系,实体的属性就是关系的属性,实体的主关键字就是关系的主关键字,关系的结构是关系模式。

2. 实体型之间联系的转换规则

从概念模型向关系模型转换时,实体型之间的联系按照以下规则转换。

1) 一对一(1∶1)联系的转换方法

一个1∶1联系可以转换为一个独立的关系模式,也可以与任意一端的实体型所对应的关系模式合并。若将其转换为一个独立的关系模式,则与该联系相连的各实体的主关键字以及联系本身的属性均转换为关系的属性,且每个实体的主关键字均为该关系的候选关键字。若将其与某一端实体型所对应的关系模式合并,则需要在被合并关系中增加属性,新增属性为联系本身的属性和与此联系相关的另一个实体型的主关键字。

【例 5.1】 将如图5.17所示是一对一联系的E-R图转换为关系模型。

解答:该例有3种方法可供选择(关系模式中标有下画线的属性为主关键字)。

图5.17 二元1∶1联系转换为关系的实例

方法1:将1∶1联系"负责"形成独立的关系。转换后的关系模型为:

职工(<u>职工号</u>,姓名,年龄);

产品(<u>产品号</u>,产品名,价格);

负责(<u>职工号</u>,<u>产品号</u>)。

方法2:将"负责"与"职工"两关系合并。转换后的关系模型为:

职工(<u>职工号</u>,姓名,年龄,产品号);

产品(<u>产品号</u>,产品名,价格)。

方法3:将"负责"与"产品"两关系合并。转换后的关系模型为:

职工(<u>职工号</u>,姓名,年龄);

产品(<u>产品号</u>,产品名,价格,职工号)。

上面的3种方法中不难发现,方法1中的关系多,增加了系统的复杂性;方法2中由于并不是每个职工都负责产品,这样会造成产品号属性空值较多;综合比较,方法3较合理。

2) 一对多(1∶n)联系的转换方法

一个1∶n联系的转换有两种转换方法:一种方法是将联系转换为一个独立的关系,其关系的属性由与该联系相连的各实体的主关键字以及联系自身的属性组成,而该关系的主关键字为n端实体的主关键字;另一种方法是与n端对应的关系模式合并,即在n端实体中增加新属性,新属性由联系对应的1端实体的主关键字和联系自身的属性构成,新增属性后原关系的主关键字不变。

【例 5.2】 将如图5.18所示的一对多联系的E-R图转换为关系模型。

解答:该例有两种方法可供选择(关系模式中标有下画线的属性为主关键字)。

方法 1：将 1∶n 联系“存储”形成独立的关系。转换后的关系模型为：

仓库(仓库号,地点,面积)；

产品(产品号,产品名,价格)；

存储(仓库号,产品号,数量)。

方法 2：将“存储”与 n 端关系“产品”合并。转换后的关系模型为：

仓库(仓库号,地点,面积)；

产品(产品号,产品名,价格,仓库号,数量)。

比较上面两种转换方法可以发现，方法 1 使用的关系多，但对存储变化大的场合比较适用；相反，方法 2 中关系少，它适合存储变化比较小的应用场合。

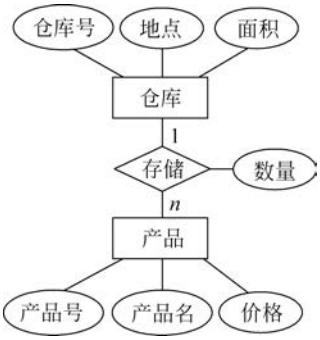

图 5.18　二元 1 ∶ n 联系转换为关系的实例

【例 5.3】 将图 5.19 所示的同一实体集内的一对多联系的 E-R 图转换为关系模型。

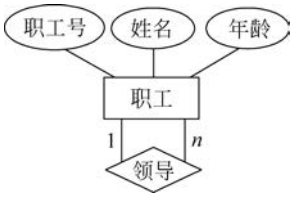

图 5.19　实体集内部 1∶n 联系转换为关系的实例

解答：该例有两种方法可供选择(关系模式中标有下画线的属性为主关键字)。

方法 1：转换为两个关系模式。

职工(职工号,姓名,年龄)；

领导(领导工号,职工号)。

方法 2：转换为一个关系模式。

职工(职工号,姓名,年龄,领导工号)。

在方法 2 中由于同一关系中不能有相同的属性名，所以将领导的职工号改为领导工号。两种方法相比，方法 2 关系少，且能充分表达，所以采用方法 2 更好。

3) 多对多(m ∶ n)联系的转换方法

此情况只有一种转换方法，一个 m ∶ n 联系转换为一个关系。转换方法是：与该联系相连的各实体的主关键字以及联系自身的属性均转换为关系的属性，各实体的主关键字组成新关系的主关键字或主关键字的一部分，新关系的主关键字为多属性构成的组合关键字。

【例 5.4】 将图 5.20 所示的多对多二元联系的 E-R 图转换为关系模型。

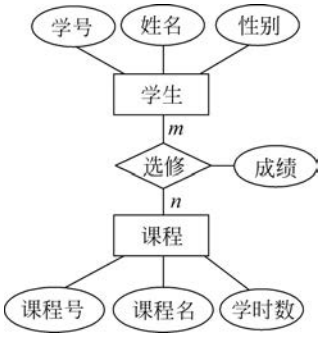

图 5.20　二元 m ∶ n 联系转换为关系的实例

解答：该例转换的关系模型为(关系模式中标有下画线的属性为主关键字)。

学生(学号,姓名,性别)；

课程(课程号,课程名,学时数)；

选修(学号,课程号,成绩)。其中，学号和课程号分别为该关系“选修”的两个外关键字。

4) 3 个或 3 个以上实体间的多元联系的转换方法

分两种情况采用不同方法处理。

(1) 对于 1∶n 的多元联系，转换为关系模型的方法是修改 1 端的实体对应的关系，即将与联系相关的其他实体的主关键字和联系自身的属性

作为新属性加入1端实体中。

(2) 对于 $m:n$ 的多元联系,转换为关系模型的方法是新建一个独立的关系,该关系的属性为多元联系的各实体的主关键字以及联系自身的属性,新关系的主关键字为各实体主关键字的组合。

【例 5.5】 将图 5.21 所示的多个实体间的多对多联系的 E-R 图转换为关系模型。

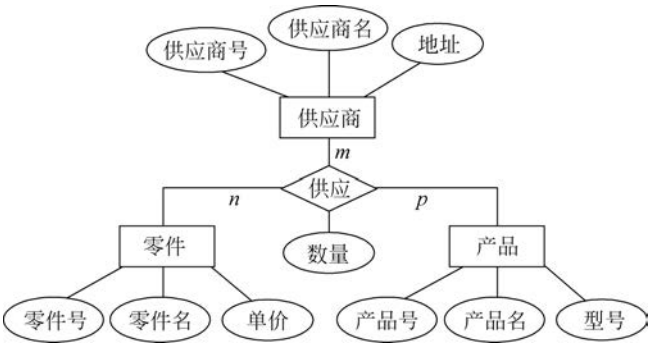

图 5.21 多实体间多对多联系转换为关系模型的实例

解答:该例转换的关系模型为(关系模式中标有下画线的属性为主关键字):

供应商(供应商号,供应商名,地址);

零件(零件号,零件名,单价);

产品(产品号,产品名,型号);

供应(供应商号,零件号,产品号,数量)。

其中,供应关系中的供应商号、零件号、产品号分别是该关系的3个外关键字。

3. 关系合并规则

在关系模型中,具有相同主关键字的关系模式可根据情况合并为一个关系。

5.4.2 关系数据模型的优化

从概念模型 E-R 图转换为逻辑模型,设计得出的结果并不是唯一的。为了提高数据库应用系统的性能,需要对所得的逻辑模型进行适当修改、调整其结构,也就是数据模型的优化。关系数据模型的优化是以规范化理论为指导原则,优化方法如下所述。

(1) 确定数据依赖:按照需求分析的语义,分别写出各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) 依据数据依赖理论对关系模式进行分析,分析是否存在部分函数依赖、传递函数依赖、多值依赖等,确定各关系模式属于第几范式。

(4) 按照需求分析要求分析关系模式是否合适,确定是否对某些模式进行合并或分解。

通常我们希望关系模式达到第三范式 3NF 甚至 BCNF 就比较良好了。但并不是规范化程度越高越好,有时根据需要可以对部分模式降低其范式,又称为反规范化。有时第二范式甚至第一范式也许是合适的。

(5) 对关系模式进行必要的分解,提高数据操作的效率和存储空间的利用率。常用的

两种分解方法是水平分解和垂直分解。

水平分解是把关系的元组分为若干子集合，定义每个子集合为一个子关系，以提高系统的效率。通常把经常使用的数据分解出来形成一个子关系。垂直分解是把关系模式 R 的属性分解为若干子集合，形成若干子关系模式。垂直分解的原则是经常一起使用的属性从关系模式 R 中分离出来形成一个子关系模式。垂直分解需要确保无损连接性和保持函数依赖，即保证分解后的关系具有无损连接性和保持函数依赖性。

5.4.3 设计用户子模式

用户子模式又称为外模式。关系型数据库管理系统(RDBMS)中提供的视图(View)就是根据用户子模式设计的。由于用户子模式与模式是相对独立的，设计用户子模式时只考虑用户对数据的使用需求、习惯及安全性要求，而不用考虑系统的时间效率、空间效率、易维护等问题。设计用户子模式应注意以下问题。

(1) 使用符合用户习惯的别名。

在合并分 E-R 图时应消除命名冲突，这在设计数据库整体结构时非常必要。但有时命名统一后会使某些用户感觉别扭，用视图机制定义子模式可以有效解决此问题。必要时可对子模式中的关系和属性名重新命名，使其与用户习惯一致，以便使用。

(2) 对不同级别的用户定义不同的子模式。

视图可以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子模式，可保证系统的安全性。

例如，假设关系模式：产品(产品号，产品名，规格，单价，生产车间，生产负责人，产品成本，产品合格率，质量等级)。可在该关系上建立两个视图，即：

给一般顾客建立视图：产品 1(产品号，产品名，规格，单价)；

给产品销售部门建立视图：产品 2(产品号，产品名，规格，单价，车间，生产负责人)。

建立视图后，产品 1 中包含了允许一般顾客查询的产品属性；产品 2 中包含允许销售部门查询的产品属性；生产领导部门则可以利用产品关系查询产品的全部属性数据。这样可以防止用户非法访问本来不允许查询的数据，保证了系统的安全性。

(3) 简化用户对系统的使用流程。

利用子模式可以简化使用、方便查询。在实际应用中经常会使用比较复杂的查询，包括多表连接、限制、分组、统计等。为了方便用户，可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，避免了每次查询都要对其进行重复描述，大大简化了使用流程。

5.5 物理结构设计

数据库的物理结构设计的目的是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。数据库的物理结构是指数据库在物理设备上的存储结构与存取方法，它依赖于选定的数据库管理系统。数据库物理结构设计通常分两步进行。

(1) 确定数据的物理结构，在关系型数据库中主要就是确定数据库的存取方法和存储结构。

(2) 对物理结构进行评价,评价的重点是时间和空间效率。

如果评价的结构满足原设计要求,则可以进行物理实施;否则应该重新设计或修改物理结构,有时甚至需要返回到逻辑结构设计阶段修改数据模型。

5.5.1 物理设计的主要内容

由于不同的数据库产品提供的物理环境、存取方法和存储结构各不相同,能够供给设计人员使用的设计变量、参数范围也各不相同,所以数据库的物理设计根本没有通用的设计方法,只能给出一般的设计内容和设计原则供数据库设计者参考。

数据库设计人员希望自己设计的数据库物理结构能够满足事务在数据库上运行时响应时间少、存储空间利用率高和事务吞吐率大的要求。因此,设计人员首先应该对要运行时事务进行详细分析,获得选择物理数据库设计所需的参数;其次,应该充分了解选定的 DBMS 的功能、提供的环境与工具等内部特征,尤其是存取方法和存储结构。

数据库设计者在确定数据存取方法时,必须清楚 3 种相关信息。

(1) 数据库查询事务的信息,包括查询所需的关系、查询条件所涉及的属性、连接条件所涉及的属性、查询的投影属性等信息。

(2) 数据库更新事务的信息,包括更新所需的关系、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信息。

(3) 每个事务在各关系上运行的频率和性能要求。例如,某事务必须在 5 秒内完成,这对于存取方法的选择有着直接影响。

关系型数据库物理设计的内容主要指选择存取方法和存储结构,包括确定关系、索引、聚簇、日志、备份等的存储安排和存储结构,确定系统配置等。

5.5.2 关系型数据库的存取方法

由于数据库是为多用户共享的系统,对于同一个关系需要提供多条存取路径才能满足多用户共享数据的要求。数据库物理设计的任务之一就是确定建立哪些存取路径和选择哪些数据存取方法。存取方法是快速存取数据库中数据的技术。关系型数据库中常用的存取方法分为 3 类:索引方法、聚簇方法和 HASH 方法。

1. 索引存取方法的选择

选择索引存取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、哪些属性列建立组合索引、哪些索引需要设计为唯一索引等。选择索引方法的基本原则如下。

(1) 若一个属性经常出现在查询条件中,则考虑在此属性列上建立索引;若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组合索引。

(2) 若一个属性经常作为最大值和最小值等聚集函数的参数,则考虑在此属性上建立索引。

(3) 若一个(或一组)属性经常在连接操作的连接条件中出现,则考虑在此属性上建立索引。

(4) 关系上定义的索引数要适当,并不是越多越好。因为系统为了维护索引需要付出

代价，查找索引也要付出代价。

例如，若一个关系的更新频率很高，这个关系上定义索引就不能太多。因为更新一个关系时，必须对这个关系上有关的索引做相应的修改。

2. 聚簇存取方法的选择

为了提高某个属性(或属性组)的查询速度，把该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。

1) 建立聚簇的必要性

聚簇功能可以大大提高按聚簇码进行查询的效率。例如，假设要查询计科系的所有学生名单，若计科系学生人数为 200，在极端情况下，这 200 名学生对应的数据元组分布在 200 个不同的物理块上。尽管对学生关系已按所在系别建立了索引，由索引很快找到计科系学生的元组标识，避免了全表扫描。但是再由元组标识去访问数据块时就要存取 200 个物理块，执行 200 次 I/O 操作。如果将同一个系的学生元组集中存放，则每读一个物理块就可得到多个满足查询条件的元组，从而减少了访问磁盘的次数。

2) 建立聚簇的基本原则

一个数据库可以建立多个聚簇，但一个关系只能加入一个聚簇。选择聚簇存取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关系。聚簇设计时可分为两步进行：先根据规则确定候选聚簇，再从候选聚簇中去除不必要的关系。

设计候选聚簇的原则是：

(1) 对经常在一起进行连接操作的关系可以建立聚簇。

(2) 若一个关系的一组属性经常出现在相等、比较条件中，则该单个关系可以建立聚簇。

(3) 若一个关系的一个或一组属性上的值重复效率很高，则此单个关系可以建立聚簇。

(4) 若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 语句中包括有与聚簇有关的 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。否则当关系较少利用聚簇操作时，最好不要使用聚簇。

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括相同的关系，一个关系可以在某一个聚簇中，但不能同时加入多个聚簇。要从这多个聚簇方案中选择一个较优的，即在这个聚簇上运行各种事务的总代价最小。

3) 建立聚簇应该注意的问题

(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取位置，这样使关系上原有的索引无效，需要重建原索引。

(3) 当一个元组的聚簇码值改变时，该元组的存储位置也要相应移动，索引聚簇码值应当相当稳定，以减少修改聚簇码值引起的维护开销。

3. Hash 存取方法的选择

某些数据库管理系统提供了 Hash 存取方法。

若一个关系的属性主要出现在等值连接条件中或主要出现在相等比较选择条件中,且满足下面两个条件之一,则此关系可以选择 Hash 存取方法。

(1) 若一个关系的大小可预知且不变。

(2) 若关系的大小动态改变,且数据库管理系统提供了动态 Hash 存取方法。

5.6 数据库的实施

完成数据库物理结构设计并进行初步评价后,就可以进行数据库的实施了。数据库实施阶段的工作是:设计人员使用 DBMS 提供的数据定义语言和其他实用程序工具将数据库逻辑设计和物理设计的结果严格描述出来,当成 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;最后就是组织数据入库了。

数据库的实施阶段包括两项重要工作:一个是数据的载入,另一个是应用程序的编码和调试。

1. 数据的载入和应用程序的调试

组织数据入库是数据库实施阶段最主要的工作。由于一般数据库系统中数据量都很大,而且数据来源于部门中的各个不同单位,分散在各种数据文件、原始凭证或单据中,有大量的纸质文件需要处理,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。组织数据录入时,需要将各类源数据从各个局部应用中抽取出来,并输入计算机,再进行分类转换,综合成符合新设计的数据库结构的形式,最后输入数据库中。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。

为了防止不正确的数据输入数据库,应当采用多种方法多次地对数据检验。由于需要入库的数据格式或结构与系统的要求不完全一致,所以向数据库中输入数据时会发生错误,数据转换过程中也可能出错。因此,需要设计一个数据输入子系统来完成数据入库的工作。设计数据输入子系统时还要注意原有系统的特点,充分考虑老用户的习惯,这样才可以提高输入的质量。现在大多数 DBMS 都提供了不同的 DBMS 之间数据转换的工具,若原来是数据库系统,则要充分利用新系统的数据转换工具,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。

2. 数据库的试运行

在部分数据输入数据库后,就可以开始对数据库系统进行联合调试运行工作了,这也就是数据库的试运行。其主要工作内容如下。

(1) 实际运行数据库应用程序,执行对数据库的各种操作,测试应用程序的功能是否满足设计要求。当然如果功能不满足设计要求,则需要对应用程序部分进行修改、调整,直到符合要求为止。

(2) 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只是近似的估计,和实际系统肯定有一定的差距,因此,必须在试运行时进行实际测量和评价系统性能指标。

值得注意的是,有些参数的最佳值常常是经过运行调试后找到的。若测试的结果与设计目标不符,则需要返回到物理设计阶段,重新调整物理结果,修改系统参数。某些情况下甚至需要返回到逻辑结果设计阶段修改逻辑结构。

最后还必须强调两点。

数据库的试运行操作应分步进行。通常应该分期分批组织数据入库,先输入小批量数据做调试用,待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。

数据库的实施和调试是不可能一次完成的。在数据库试运行阶段,由于系统处于不稳定状态,软硬件故障随时都可能发生。另外,由于系统的操作人员对新系统还不熟悉,误操作在所难免。因此,在数据库试运行时,应该首先调试运行 DBMS 的恢复功能,做好数据库的转储和恢复工作,一旦发生故障就可以使数据库尽快恢复,尽量减少对数据库的破坏。

5.7 数据库的运行与维护

数据库试运行合格后,数据库开发工作基本完成,即可投入正式运行了。但由于应用环境不断变化,数据库运行过程中物理存储也不断变化,对数据库设计进行评价、调整与修改等维护工作是一个长期的任务。

在数据库运行阶段,对数据库经常性的维护工作主要是由数据库管理员 DBA 完成的。数据库的维护工作包括 4 个方面。

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能够尽快地将数据库恢复到某种一致的状态,并尽可能地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,对安全性的要求也发生变化。例如,有的数据原来是机密的,现在可以公开查询,而新增加的数据又可能是机密的。这些都需要数据库管理员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,也需要数据库管理员不断修正,以满足用户要求。

3. 数据库性能的监督、分析和改造

在数据库运行过程中,监督系统运行、对检测数据进行分析,并找出改进系统性能的方法。目前某些 DBMS 产品提供了监测系统性能的参数工具,数据库管理员可以利用这些工具方便得到系统运行过程中一系列性能参数值。数据库管理员应仔细分析这些数据,判断当前系统运行状况是否是最佳,应当做哪些改进。例如,调整系统物理参数,或对数据库进

行重组织或重构等。

4. 数据库的重组织与重构造

数据库系统在运行一段时间后,由于存在记录不断地增加、删除、修改等操作,使数据库的物理存储情况变坏,降低了数据的存取效率,数据库的性能下降。此时,数据库管理员需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)。DBMS 一般都提供了数据重组织的实用程序。在重组织过程中,按原设计要求重新安排存储位置、回收垃圾、减少指针链等,从而提高系统性能。

数据库的重组织并不修改原设计的逻辑结构和物理结构,而数据库的重构造是指部分修改数据库的模式和内模式,也就是需要修改逻辑结构或物理结构。当增加了新的应用或新的实体,或取消了某些应用,某些实体与实体间的联系发生了变化时,原来设计的数据库结构不能满足新的需求,此时需要调整数据库的模式和内模式。当然,数据库的重构造是有限的,只能做部分的修改,如果变化太大,而重构造还是不能满足需要,那么说明需要设计新的数据库应用系统了。

小结

本章主要讲解了数据库设计的方法和步骤,从需求分析到概念结构设计,再到逻辑结构设计和物理结构设计,最后是数据库的实施和数据库的运行与维护等。其中,重点与难点内容是数据库的概念结构设计和逻辑结构设计,也是数据库设计过程中最重要的两个环节。

需求分析是软件过程中至关重要的一步,也是数据库设计的第一步,是需要充分准确地分析用户对系统的需求。需求分析的成功与否对软件系统设计的好坏非常关键,因此,进行需求分析时必须严格按照软件工程思想进行,但这个阶段不是本课程的重点内容,有关需求分析的详细内容请参考软件工程相关课程。

数据库的概念结构设计是将用户需求抽象为概念模型,是整个数据库设计的关键。概念结构真实充分地反映现实世界,包括事物和事物之间的联系,而且易向各种数据模型转换(如关系数据模型)。利用 E-R 模型图描述数据库的概念模型。本节主要讲解了从局部 E-R 模型的设计到全局 E-R 模型的设计过程。在局部 E-R 模型设计中详细介绍了各种数据抽象方法的使用、简单属性与复合属性、单值属性与多值属性,以及弱实体等概念;在全局 E-R 模型设计中详细介绍了解决合并局部 E-R 模型过程中的 3 种冲突:属性冲突、命名冲突和结构冲突,从而形成初步 E-R 模型,并讲解了两种消除冗余方法,即分析方法和规范化理论法,最终设计出合理的基本 E-R 模型。

数据库的逻辑结构设计是将概念模型转换为逻辑模型,本章主要是利用基本 E-R 模型转换为关系数据模型。重点讲解了 E-R 模型向关系模型的转换规则,即实体型转换规则、实体型之间的联系转换规则以及关系合并规则。同时,还综合介绍了如何对转换后的关系模型进行合理适当的优化,其中值得加深理解与注意的是反规范化,即某些时候需要适当降低规范化程度。

数据库的物理结构设计是数据库在物理设备上的存储结构与存取方法,它依赖于选定的具体数据库管理系统。完成数据库物理结构设计并进行初步评价后,就可以进行数据库

的实施了。数据库的实施包括两项重要工作：一个是数据的载入，另一个是应用程序的编码和调试。而数据库正式运行后，需要由数据库管理员负责数据库的运行管理与维护工作。

习题 5

5.1　问答题

1. 试述数据库的设计过程。
2. 什么是数据库的概念结构？试述其特点和设计策略。
3. 什么叫数据抽象？试举例说明。
4. 什么是数据库的逻辑结构设计？试述其设计步骤。
5. 试述 E-R 模型转换为关系模型的转换规则。

5.2　设计题

1. 学校中有若干系，每个系有若干班级和教研室，每个教研室有若干名教师，其中有的教授和副教授每人各带若干研究生，每个班有若干名学生，每名学生选修若干课程，每门课程可由若干名学生选修。请用 E-R 图表达其概念模型，并将其转换为关系模型。

2. 如图 5.22 所示为某个教务管理数据库的 E-R 图，请将其转换为关系模型。

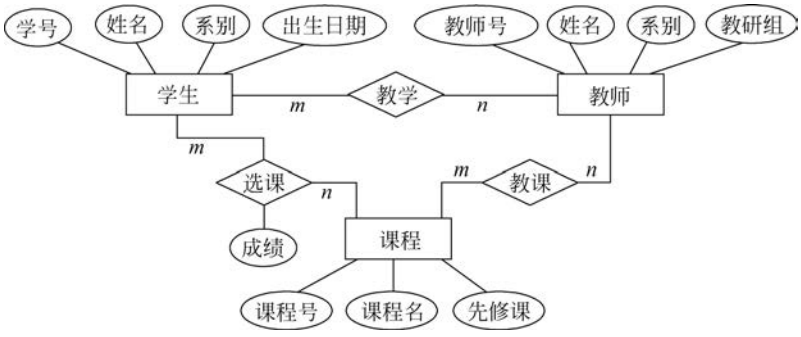

图 5.22　教学管理数据库 E-R 图

3. 如图 5.23 所示是一个销售业务管理的 E-R 图，请将其转换为关系模型。

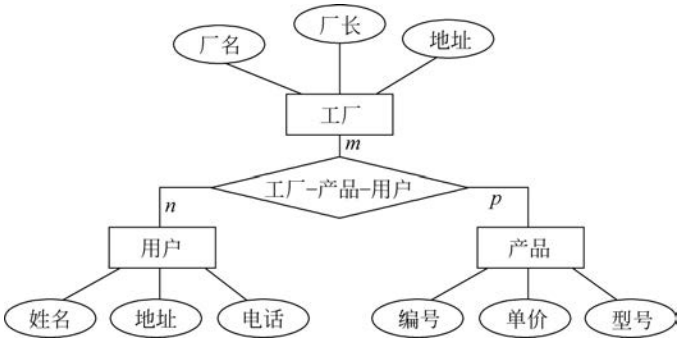

图 5.23　销售业务管理的 E-R 图

4. 现有一个局部应用，包括两个实体："出版社"和"作者"，它们是多对多的联系，请设计适当的属性并画出其 E-R 图，再将其转换为关系模型（包括关系名、属性名、关键字）。

5. 设计一个图书馆数据库,此数据库中对每名借阅者保存读者记录,包括读者号,姓名,地址,性别,出生日期,单位;对每本书存有书号、书名、作者、出版社;对每本被借出的书存有读者号、借出日期和应还日期。要求画出其 E-R 图,并将其转换为关系模型。

6. 设有一家百货商店,已知信息有:

(1) 每名职工的数据是职工号、姓名、地址和所在商品部。

(2) 每个商品部的数据有:它的职工、经理和它经销的商品。

(3) 每种经销的商品数有:商品名、生产厂家、价格、型号和内部商品代号(商店自己规定的)。

(4) 关于每个生产厂家的数据有:厂名、地址、向商店提供的商品价格。

请设计该百货商店的概念模型,再将概念模型转换为关系模型。注意,某些信息可用属性表示,其他信息可用联系表示。