C语言数据库开发

李沐荪　编译

北京科海培训中心

一九九〇年十月

李沐荪　编译

C语言数据库开发

- 自己开发，不受商品数据库限制
- 步骤清楚，给出实例全部源程序
- 提供通用工具库
- 适用于PC机和多种C语言

北京科海培训中心
一九九〇年十月

C语言

目　录

第一章 引 言

本书的软件由许多函数（或功能块）组成，可以用在软件系统中，系统本身则是一些工具，可以当作应用软件的构成部分。这些函数能够支持大多数系统的通用要求，这样，在开发新的应用系统时，就能够节省不少的代码编写工作。本书的工具代表一层软件，介于专门的应用代码和硬件及操作系统之间。这样，它们就能支持较高层的通用系统要求，从而减少开发的代码量。使用已经证明可靠的实用代码——或软件工具——而不是每次都要另起炉灶，所编写的软件系统的可靠性就会有显著的改善。

编制程序应当考虑到软件的可移植性，充分使用自顶向下设计、自底向上开发、结构式编程、信息隐蔽等技术，尤其重要的，则是注意累积可重复使用的软件工具。这些方法都是非常有效的，只要认真按照这些原则去实施，必然得益不浅。

如果开发的软件工具可移植到各种类型的计算机，则累积软件工具，就可更快地出成果。有些程序员任务频繁，甚至不列程序清单，完成一项就放在一边，只凭自己的才智学识又去开始新任务。也许更好的办法是学会如何累积可重复使用的软件，为未来的应用打下基础。本书将为此提供例子。

有关C语言的著述不少，虽然各有长处，但是关于如何提供有用的C函数，则多数显得不足。有些书根本不注意这一点，而注意到这一点的也似乎并未解决问题。

典型的书不外先解释C语言，然后用一些例子说明如何使用各种特色。有些书中提供简单的函数，说明算法和语言的原则。这些函数往往和编译库中的相似，更多的则是一些没有实用价值的教科书例子。作者的讲授必须尽可能简炼。大型软件系统由于源文件卷帙浩繁，不仅印刷困难，解释清楚也很不容易。复杂的算法需要简单的说明，微妙之处，难以尽言。即使这样，本书仍打算提供全套有用的软件工具，可用于读者下一次要开发的软件系统中，便于使用，便于理解。书中公布的软件都可以实际应用，书中的知识与工具都可以用于开发数据库系统。

为使用本书，要求程序员懂得IBM PC及P-CDOS的软件开发环境。需要知道DOS的分层文件结构，以及与之相关的路径，文件目录和子目录。要知道PC-DOS批文件是如何工作的。这方面有不少好的教科书。如Peter Norton的IBM PC程序员指南（1985）和Ray Duncan的MS-DOS操作系统高等教程（科海技术参考资料之114，李沐苏编译）。当编写的程序超出标准C函数库时，这两本书是手头必备的

本书并不打算讲授如何用C语言编程；相反，假定读者已经是C程序员。不学会C语言的基本知识也许仍能装置书中的软件，但决不可能应用自如。所举的例子是一些功能很强而且十分有用的软件包，但它们本身不是这本书的目的。我们的目的是演示如何使用工具库。但还不尽于此。要想成为成功的程序员，有两点是关键：一是充分认识累积软件工具的必要性；二是收集可用的工具。如果书中收集的软件能够表明工具编制是好主意，则本书的第一个目的就算达到了，如果读者在自己的工作中能够用上这些工具，第二重目的也

已实现。

如果书中给出的函数能够使读者的个人软件工具库得到显著的扩充，可以认为本书已经达到目的。如果读者从来没有编过工具库，以本书作为开始，则本书取得的成就更高，因为它使读者开始进步，成为更好的程序员。

数 据 库 管 理

以后各章中将看到的数据库管理系统被写成通用性软件，可以移植到许多环境中，无需对所用的硬件或操作系统作不必要的考虑。书中的软件包曾工作于多种计算机，从Z80到80286，在四种操作系统之下有PC-DOS，CP/M-80，CP/M-86和TurboDOS。系统被汇集成单一的软件包，可以在IBM PC机中IBM PC-DOS下运行，但是可以移植到其他环境中。本书给出的是PC机版本。

各章内容提要

二至八章论述支持开发应用软件的数据库和数据库管理系统（DBMS）。这几章中包括了一个函数库，可以从应用程序中调用，它提供DBMS的大多数功能。具备了这些知识和工具库后，能编写以关系模式处理数据的软件系统。关系数据库原理在第四章中讨论。

第二章解释数据库原理，阐述构成数据库的各个数据成分。然后说明怎样用一种称为数据库图式（data base schema）的语言来描述数据库。读者将学会如何将数据库设计为逻辑上交连的文件，文件中含有数据元素，如何用称为关键字（key）的选定的数据元素来检索文件中的记录。然后将看到数据元素字典是所有关系数据定义的基础。

第三章讨论如何起草设计数据库。首先收集对数据存储的要求，然后列出必须存取的数据项清单。

第四章解释DBMS本身，它是支持数据的软件系统，提供函数以说明数据库的结构，根据用户的调用存取记录，保持记录索引的完整性和连续性。本书处理DBMS的方法主要包含在本章中，并假定读者具有中等程度的C语言句法知识。

第五章讨论如何使用C语言定义数据。章中介绍的技术允许改变数据格式而不必考虑大多数的数据处理软件。

第六章介绍“廉价数据库管理系统”，称为Cdata。在Cdata中，用第五章介绍的技术来规定数据库。然后讨论Cdata的数据操作语言函数。这些函数用来向数据库存放或从数据库读取记录。

第七章给出一系列实用函数，用来操作数据库中的数据。这些函数包括将数据送入文件记录的程序、根据索引关键数据元素读取记录的程序、以及根据从数据库读出的数据记录形成报告的程序。第七章还包括一个图式汇编程序，用来从文本文件读取数据库图式的描述，并产生第六章描述的Cdata数据定义语言。

第八章给出一个小而全的例子——这是一个软件系统，利用Cdata DBMS生成顾问帐单系统。

最后，第九章解释怎样用一种编译器生成本书的软件。每一种编译器都配有PC-DOS

批文件，用来生成工具库函数和应用程序。

C 编 译 器

本书的软件曾在IBM PC机上用九种不同的C编译器装置过：

- Aztec C
- Computer Innovtions C86
- DataLight C
- ECO-C88
- Lattice C
- Microsoft C
- Turbo C
- Wizard C

下面的讨论不打算比较编译器的优劣。它们都能工作，都是优良的产品，当然各有优缺点。只要有其中的一种，就可以使用本书的软件，就能工作。如果有其他不同的编译器。只要它支持全部C语言，转换将不会有困难。从前，C程序之间的转换是非常麻烦的；没有统一的语言标准，编译器各不相同，大多数程序员在移植性问题上没有什么经验。但是这一问题已经随着时间和经验而消失。本书中的软件经过多次修改，形成了比较一致的可移植的代码。目前，编译逐渐向标准语言靠拢，程序员也更加注意编写可移植的代码。

本书中的软件可以用九种不同的编译器工作。但IBM PC上至少还有七种C编译器可用：

- C-systems C
- DeSmet C
- Digital Research C
- High C
- Mix C
- QC88
- Whitesmiths C

可能会奇怪，为什么一个计算机需要16种不同的编译器（还有差不多同样数量的解释器（Interpreter）服务于同一语言。还有没有其他语言有这么多编译器，都畅销，都服务于同一计算机，同一操作系统？C语言的深受程序员欢迎以及它天生与个人计算机结构的兼容性，是如此大量编译器成功的关键。各大软件公司选定C为他们的开发语言更助长了对C的狂热。用C语言写的软件要比用其他语言写的更容易转移到其他计算机和操作系统。这种移植性是C语言最受赞扬的特性，也是C语言备受欢迎的主要原因之一。软件公司希望能够迅速对计算机工业的发展趋势作出响应。可移植软件给予软件公司这一能力；C语言给予他们可移植的软件。

小 结

本书中的函数是有用的，富有启发性和有趣的。但远胜于一切的是一种享受。为同行

程序员开发工具是一种理想的工作。长期从事编程工作会认识各方面的用户，但多数程序员并不了解用户的生活，然而，他们却深知程序员圈内的苦乐。有些人已经历了几代的计算机，反复地发现和再发现编程技术的原理与实践。希望读者在学习和使用书中的程序时，能感受到作者在开发它们时同等的乐趣。

第二章　数据库基础知识

以下的三章将讨论数据库结构的原理与设计，以及数据库管理系统（DBMS）。如果读者是一位经验丰实的数据库分析家，可能想跳过这几章，但最好还是全部读一遍。虽然三章都不详尽，但合在一起能帮助读者理解书中其余部分的软件，因为它们揭示了数据库理论的一种观点，可能和读者自己的不同。软件工业的许多“标准”是通过大众使用，而不是通过究研应用，而一再修改定义的。当一个技术名词，例如“关系数据库”，流行时，可能已失去最初定义时的意义。广告，非正式的新闻媒介，外行的插入都可能导致一个名词违反原意而重新定义。

系统设计或开发者的任务是复杂的。他，或她的目的是产生有用的自动化系统。为此，必须对系统部件作出有条理而且严格的定义。一个自动化系统包括许多元素或子系统，例如需求分析、硬件、软件、用户和操作员文献，维护与培训。

本书是讨论软件的。软件包括程序和信息；好的程序需要信息，以便处理，而信息需经有效的处理才是有用的。为使信息便于处理，必须组织为严格而且一致的格式，这就是数据库。

本章定义数据库，用一个小型人事档案库作为例子。第三章解释数据库设计过程，并用人事档案库的设计为例，说明设计步骤。第四章解释数据库管理系统如何将数据库设计编入管理它的软件系统中。

数　据　库

定义：

数据库是互相关联，支持共同目标的自动化数据文件的整体集合。

数据库中的文件由**数据元素**（data elements）构成——如帐号、日期、数量、名字、地址、帽子尺寸以及各种可识别的数据项。数据元素由数据库组织、存放和读出。为设计这样的数据库，需要懂得什么是文件、数据元素，如何使用它们。

对于设计者来说，数据意味着自动化信息：就是说，信息采取一种能被计算机接受，进行自动化处理的格式。信息首先必须化为机器可读的形式，才能自动化。

计算机数据的最小数据成分是比特（bit），其值为0或1的二进制元素。程序员有时也考虑比特，但作为数据库设计者，将从更高的层次来看数据。比特用来构成字节(byte)或字符，字符用来构成数据元素。数据文件包含由数据元素组成的记录，数据库由数据文件组成。花些时间来理解数据库中数据的层次是必要的，以后常要用到这方面的概念。从最高级开始，层次如下：

1. 数据库
2. 文件
3. 记录

4. 数据元素

5. 字符（字节）

6. 比特（位）

在设计的本阶段，前五层是我们感兴趣的。数据库包含记录文件，记录包含由字符组成的数据元素。

图 2.1是数据库的比拟——办公用的文件抽屉。抽屉代表数据库，文件夹代表文件，夹中的文献代表记录，文献中的信息代表数据元素。有些数据库甚至是用卷柜、抽屉、文件夹在屏幕上代表数据库的部件。

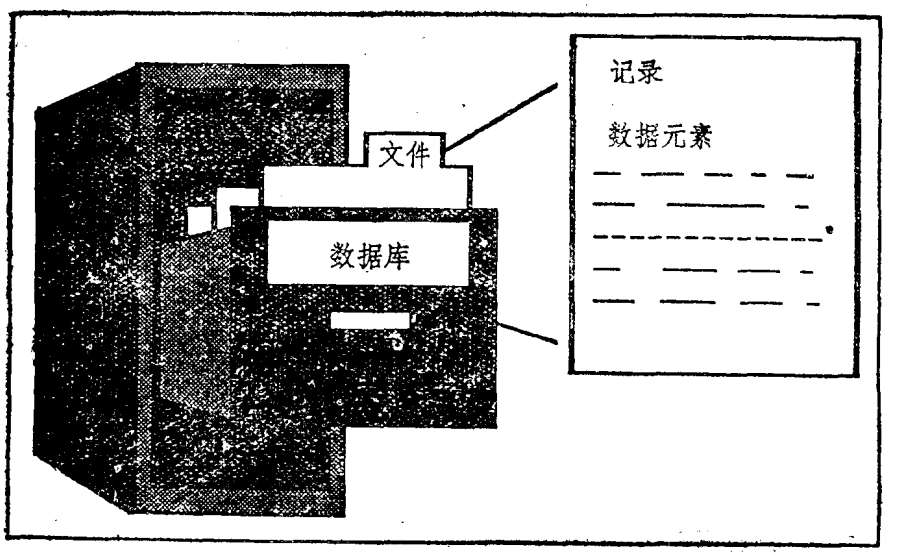

图 2.1 数据库的比拟

然而这种比拟并不完善。文件抽屉中的纸记录通常并无严格规定。当然，档案管理员会努力整理，但控制只取决于人的注意力。在自动化数据库中，数据库和软件中备有控制。每一项数据完成特定任务，信息不作不必要的重复。凡是必需的，必须具备，凡是不合适的，禁止入内。文件中的记录格式必须一致，同一文件中所有记录的数据元素必须相同。

为设计数据库，必须懂得数据结构及其相互关系。下面由自动化数据的最底层——数据元素开始来逐步深入理解。

数 据 元 素

数据库中装着各项信息，信息项所含值需要存放地点，数据的存贮点称为**数据元素**。

定义：

数据元素是文件中存放信息项的地点，信息项可以唯一地根据它的目的与内容识别。

数据元素与数据库所支持的应用程序在功能上相关联。数据元素的例子为日期、帐号、数量，名字、地址、电话号码和命中率平均值等。它们和应用目的的功能关系是很重

要的：一个数据元素之所以被记录到数据库中，是因为它对系统所支持的功能有关。例如，雇员的工资是工资系统中的有关数据元素，而现有数量是库存管理系统中的有关项。任一数据元素放在另一系统中就错了。

数据元素是信息瞬时值的暂存处。不要把数据元素（放置数据的地点）和**数据值**（数据元素的当前值）混为一谈。雇员记录将永远含雇员工资数据元素，但数据元素对不同雇员可以有不同的值。对某个雇员的值也可因时而变。

定义：

数据值是数据元素中存贮的信息。

数据元素字典

数据库设计常常以数据元素表的设计开始。不管是否这样开始，最终总是要建立起一张的。下表是一个小型人事系统数据库的数据元素表例子。它说明了数据库设计的基础——**数据元素字典**。

数据元素名	数据类型	长度
empoyee_no	numeric（数字）	5
employee_name	alphanumeric（字符数字）	25
data_hired	date（日期）	6
salary	currency（货币）	10
dept_no	numeric（数字）	5
dept_name	alphanumeric（字符数字）	25
proj_no	numeric（数字）	5
proj_name	alphanumeric（字符数字）	25
start_date	date（日期）	6
completion_date	date（日期）	6
labor_budget	numeric（数字）	5
hours_charged	numeric（数字）	5

数根库字典是本书数据库管理工具应用的核心。必须对所有出现在屏幕、报告、数据文件中的数据项在数据库字典中定义。例外的是常数，和屏幕及报告中报告头和输入提示所用的文字信息。不需要为这类信息定义数据元素，虽然有些数据库管理系统不区分这类信息和真正的数据元素。

定义：

数据库元素字典是一张数据元素表，至少包括数据库中每个数据元素的名字，数据类型和长度。

数据元素类型

上示数据元素词典包括元素的名字、类型和长度。类型来自下表：

日期	Date
货币	Currency
数字	Numeric
字母数字	Alphanumeric

这些不是唯一可用的，但是在商业应用中，可用以上的类型之一来说明任何数据元

素。其他类数据元素类型（generic data element types）有电话号码、邮政编码、州、社会保障号码等。系统支持类数据元素类型的程度不相同。有一种最流行（最昂贵）的大型机数据库管理系统只允许定义数据元素为数字或字母数字，不允许其他专门类型（generic type）。通常，类数据元素类型不进一步定义不能充分说明数据。有些类型隐含形式和功能，但不严格。日期或电话号码的隐含格式看似明显，但这两种简单类型的格式可能有多种变化。考虑下面的例子：

电话号码	日期
(202) 555-1212	01 102186
PE6-5000	3 Jan 64
Dial'M'for Myrtle	The Ides of March

数据元素名

数据元素字典中的数据元素名用来使程序和用户能识别数据元素。选择的名字应当能够说明它们所代表的数据元素的功能和目的。如果读者习惯于一般程序员爱用的密码式或缩写数据元素名，最好重新考虑，改变这种习惯。在利用本书的工具生成的数据库管理系统中，用户和程序员看到的数据元素名是相同的。它必须既可被人读，也可被机器读。

C语言数据识别符被用作数据元素名。识别符的前31个字符是独一的，足以写出能充分说明问题的名字。第5章解释如何应用这一技术。

数据值的表达

数据库中的数据元素往往根据计算机或语言的数据类型来保留存贮空间或时间。在同一文件中，可能混合使用二进制整数，浮点数字符串和位标志。在本书中使用的数据存贮方法是将所有数据元素值存贮为以零结束的字符串。这是有好处的。可以用系统的实用程序（如TYPE,DUMP等）在屏幕上显示和观察文件数据。使用ASCII字符串使数据格式和处理数据的程序可以移植，计算机内部数据格式不同不会产生影响。不必耽心语言或操作系统会把由某些位组成的模式（例如，文件末尾标志）看成非法。

文　件

数据库中含有因共同目的而发生关系的一组文件。不要混淆**文件**和**记录**。文件是记录的集合。记录格式相同，内容各异；因此文件中的记录都有相同的数据元素，但不同的数据元素值。

定义：

文件是记录的集合，记录中含有相同格式的相同数据元素。

可以根据文件中数据元素表推出记录的格式。

数据与数据库支持的系统目的相关；文件的组织为数据提供功能性存贮。文件之间的相互关系依据于文件目的的功能关系。为说明一个文件，应赋予名字，并提供数据元素清单，名字应和文件目的相关，正如数据元素名应能描述元素中的数据。如果设计雇员文件，取名如"EMPLOYEES"是有意义的。下面是如何说明文件的例子：

EMPLOYEES:

employee_no,employee_name,date_hired,salary

从数据元素字典中可以查到数据元素的长度，不难求出EMPLOYEES中元素长度之和等于46个字符。C语言在字符串终了时加零字节，因此每个数据元素长度应再加上一个字符，总数变为50，这就是EMPLOYEE文件的记录长度。

数据库图式

当设计数据库时，必须为文件和每个文件的格式准备正式的叙述。准备这一叙述的技术随开发人员乐用的风格不同，但这类叙述至少必须包含文件、其数据元素、每个文件中用于识别特定记录的数据元素。以及文件间的隐含关系。数据库的这种描术称为**图式**(schema)

定义：

图式是数据库的表达方式，表明它存贮的文件，每个文件中的数据元素，用于识别记录的关键数据元素，以及文件之间的关系。

在本书的讨论中用到的小型数据库是一个关系数据库。记录索引以及文件与文件关系所用技术取自数据库管理的关系理论。还有其他数据模式，第四章将讨论可用的数据模式。因为本书涉及数据库管理软件，又因为这些软件支持关系模式，讨论将偏重于这一方面。

数据文件的说明

如前所述，数据库是文件的集合。如果懂得什么构成文件，就可以考虑数据库的部件。数据库最简单的形式是一组共同目的的文件。人事系统的例子可加扩充来阐明这一点。数据库（现在取名为PERSONNEL）可能由下列文件组成，包括以前定义的文件EMPLOYEES在内：

PERSONNEL：

EMPLOYEES：employee_no,employee_name,date_hired,salary

DEPARTMENTS：dept_no,dept_name

PROJECTS：proj_no,proj_name,start_date,completion_date,labor_budget

将图式译成数据库管理软件系统通常涉及用一种语言向数据库管理软件说明图式。这种语言有时称为**数据定义语言**。在本书的数据库定义方法中，利用C语言的方便建立一种数据定义语言，如第四章所述。

为完成数据库设计，图式中还含有两项数据构造：文件索引键的说明和文件之间的关系。

关键数据元素

每个文件中的记录必须能被唯一地识别，从而使系统能找到并读出。需要一种方法告知计算机要读取哪一个记录，计算机也需要知道如何找到要求的记录。有些文件管理系统给出记录在文件中的位置，亦即记录号，来读取记录，但这种方法不是很完善的。不应当去记住记录的存放点，只需要知道它是在文件中，並可供使用，就应当解决问题。然而，有理由要求用户记住记录中数据的某些方面。只要告诉计算关于某些方面，计算机就能搜寻含有给定值的记录，把它找出来。

文件的定义中应当规定文件的关键字元素（一个或几个）。文件的**关键字**(key)逻辑上指向它索引的记录。以后也简称关键字为关键。

因为关键字是进入文件的索引，数据库管理工具就采用这一方法。对于索引数据元素来说，一个文件可以赋予一个以上的关键，这意味着可以根据几个数据元素值来读取一个记录。即使如此，文件通常还是有一个唯一的关键能够区分文件中所有的记录。这一关键称为主关键（primary key）因为文件中只允许一个记录可以含主关键的某个给定值。

定义：

文件中的**主关键数据元素**是一个数据元素，它来用唯一地说明和定位所需的记录。关键可以是一个以上数据元素的组合。

在文件EMPLOYEES中，数据元素“employee_no”是主关键。索引文件的非主关键是副关键（secondary keys），可以有一个以上的记录含有相同值、在人事数据库PERSONNEL中，开始仅须识别各文件的主数据元素。为此，下面再次给出了数据库图式，各文件的主关键元素加了下划线。

PERSONNEL:

EMPLOYEES: employee_no, employee_name date_hired, salary

DEPARTMENTS: dept_no, dept_name

PROJECTS: proj_no, proj_name, start_date, completion_date, labor_budget

如果知道雇员编号，就可以找到雇员记录。同理，部门记录以部门编号(dept-no)为关键，任务编号（proj-no）索引任务文件。

文件间关系

人事数据库PERSONNEL的图式並未示出文件是怎样关联的。维持文件之间的关系的能力是数据库管理系统的强力功能之一。当一个文件中的数据记录与另一个文件中的数据记录关联时，文件之间就发生了关系。当雇员被分配任务时，雇员某甲（在EMPLOYEES文件中）可能与曼海顿计划（在PROJECTS文件中）相关。为使文件之中存在关系，並不要求所有雇员都被分配任务，也不要求所有任务都配有雇员。可以有一批雇员，有一批任务，却没有分配任何任务，但是只要数据库设计提供了向雇员分配任务的潜在可能性，则关系还是存在的。在人事数据库PERSONNEL例子中，没有示出文件关联的潜在性，但下面将把这一能力加入数据库图式中。

在数据库中，可用三种方式文件相关：

- 一对一
- 多对一
- 多对多

考虑教师与学生记录之间的关系。当学生聘请全日制辅导教师以便更好地学习某科时，辅导教师文件和学生文件之间发生一对一关系；一个辅导教师只有一名学生，一名学生只有一位辅导教师·支持教师—学生关系的数据库被称为支持两个文件之间的一对一关系。虽然数据库有许多学生和许多教师，关系是一对一的因为每名教师或学生只有另一个对方。

在小学里，一位教师有多名学生，而每个学生仅有一位教师，这构成一对多关系。

在中学或大学中，一位教师教多名学生，每个学生有多名教师·这是多对多关系的一个例子。

图2.2利用这一例子说明如何画出师生间的关系。为设计师生关系数据库，可以用一张类似图2.2三个部分之一的图表开始。每个椭园代表一个文件，连接箭头代表文件之间的关系。单箭头指向“对一”关系；双箭头指向“对多”关系。

图2.3用箭头符号示出PESONNEL数据库的关系。

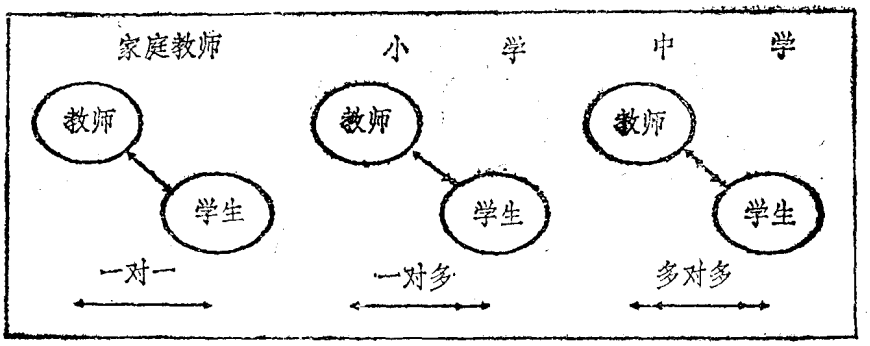

图 2.2　三个关系

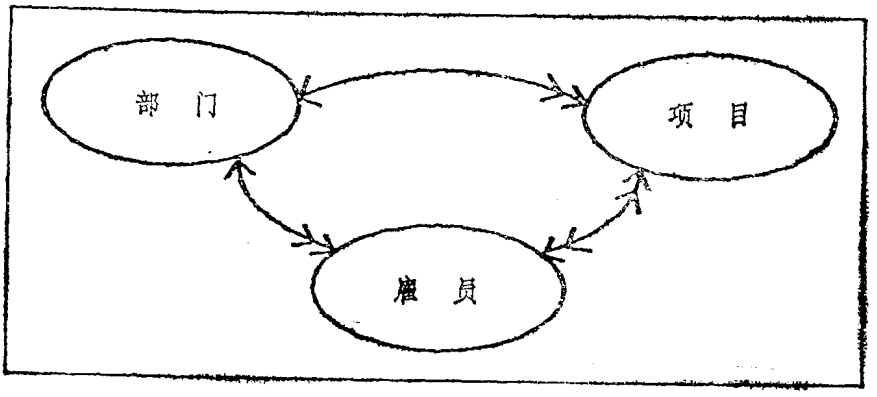

图 2.3　RESONNEL数据库

每个部门只承担一项任务，每项任务只属于一个部门；因此DEPARTMENTS文件和PROJECTS文件之间关系是一对一的。

一个部门雇用多名雇员，一名雇员只能为一个部门工作，因此DEPARTMENTS文件对EMPLOYEES文件的关系是一对多。反过来，EMPLOYEES文件和DEPARTMENTS文件的关系是多对一。

一个任务可被派给多名雇员，一名雇员可以承担多项任务，所以EMPLOYEES文件和PROJECTS文件之间的相互关系是多对多。

描述这些关系的技术是不复杂的。一个文件有一个主关键数据元素。如果另外一个文件与该文件有多对一关系；则另外的文件将第一个文件的主关键数据元素包括在它的数据元素中。下面是PERSONNEL数据库图式，黑体字表示由于文件间存在关系而收入的关键元素：

PERSONNEL：

EMPLOYEES: employee_no, employee_name, date_hired, salary, **dept-no**

DEPARTMENTS: dept_no, dept_name

PROJECTS: proj_no, proj_name, start_date, completion_date, labor_budget, **dept_no**

注意关键元素dept_no被加入了文件EMPLOYEES和文件PROJECTS中。结果是这两个文件和DEPARTMENTS文件之间形成多对一关系。假定部门和任务之间是一对一关系。可以用两种技术支持这一关系：一是将两个记录连接（物理上结合）为一个，二是利用软件向PROJECTS文件加入记录，强行造成一对一关系。如果部门已被分配另一任务，添加记录是不允许的。

然而两种方法都有缺点。第一种方法要求所有部门记录为任务数据保留空间，即使部门并无任务。第二种方法要求应用软件强迫实现关系。这种方法存着潜在的由于编程不小心而造成丧失数据完整性的可能；如果数据库设计中包括了程序不能破坏的内在完整性约束，效果会好些。究竟选用哪一种办法取决于手头的问题，以及每种方法针对应用的相对指标。

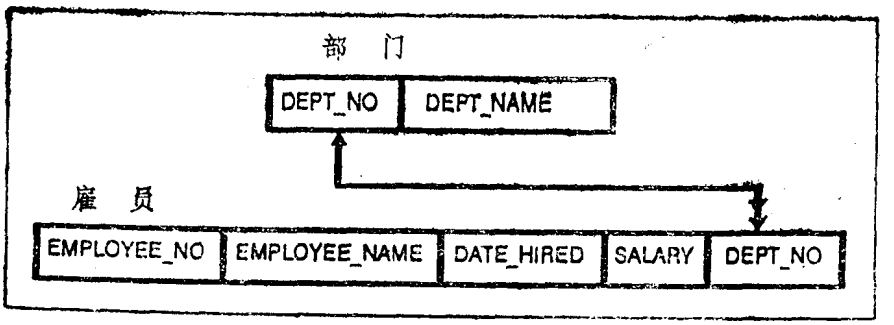

图 2.4 DEPARTMENTS和EMPLOYEES:一对多

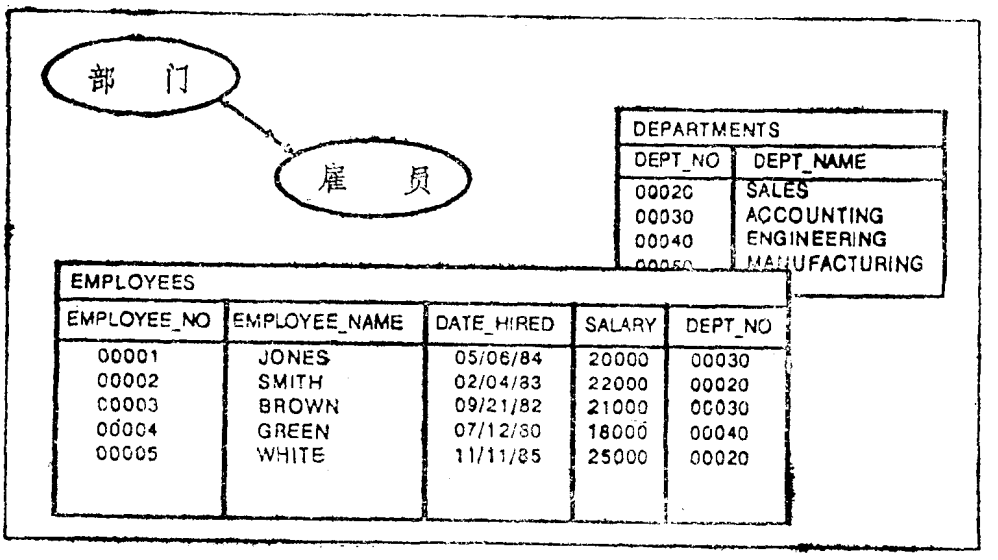

图 2.5 形成一对多关系

图2.4和2.5示如何形成一对多关系。当dept_no——这是DEPARTMENTS文件的主关键——被放入EMPLOYEES记录中作为一个数据元素时，两个文件就发生了关系。此后，每个EMPLOYEES记录可以指向一个DEPARTMENTS记录，又因为可以有多个EMPLOYEES记录指向一个DEPARTMENTS记录，这种设计支持一对多关系。图2.5示出两个文件和记录值。注意有两个EMPLOYEES记录（employee_no的00002和00005）含有同一个dept_no（0020），另一方面，没有EMPLOYEES记录能够和多个DEPARTMENTS记录相关。

还有一个关系有待建立。PROJECTS文件和EMPLOYEES文件之间有多对多关系，它不像其它关系那样容易建立。初看上去，可能会认为可以把proj_no数据元素放在EMPLOYEES文件中，把employee_no数据元素放在PROJECTS文件中，但这是不行的，因为一个任务可以指向一名雇员，而后者又指向另一个任务。相反，必须建立一个附加的文件，其主要功能就是维持关系。这称为**连接记录文件**（file of connector records），因为新文件中的记录实现其他两个文件之间记录的逻辑连接。然而通常除了起连接其他两个文件的作用以外，还可以完成外加的功能。常常存在这样的信息，只和关系有关，但作用于两个成员之一时，是无意义的。例如，在雇员与任务之间的关系中，可能要记录每个雇员花费在每项任务上的时间。经过扩充后的图式，附有雇员——任务分配新文件，看上去如下：

PERSONNEL:

EMPLOYEES:	employee_no, employee_name, date_hired, salary, dept_no
DEPARTMENTS:	dept_no, dept_name
PROJECTS:	proj_no, proj_name, start_date, completion_date, labor_budget, dept_no
ASSIGMENTS:	employee_no, proj_no, hours_charged

注意employee_no和proj_no数据元素有下划线。这两项连接在一起成为文件的主关键，因为两项都需要，才能唯一地确定一个记录。文件可以含同一雇员的多个记录和同一任务的多个记录，但只有一个记录可以有employee_no和proj_no的特定值的组合。

图2.6和2.7是多对多关系的例子。在图2.6中，PROJECTS和EMPLOYEES文件与早先的图式相比没有改变，不支持任何关系。图式中加入了ASSIGNMENTS文件，因而能够计算每名雇员为每项任务支付的时间。此外，它支持PROSECTS和EMPLOYEES之间的多对多关系。ASSIGMENTS文件分别对PROJECTS和EMPLOYEES文件有多对一关系，因为它含有它们的主关键作为数据元素。所以给定一个ASSIGNMENTS记录，可以指向一个PROJECTS记录和一个EMPLOYEES记录。此外，不论是PROJECTS或EMPLOYEES中的一个记录可以被多个ASSIGNMENTS记录所指，多对多关系是通过这二个一对多关系建立的。由一个已知的EMPLOYEE记录，可以抽取含有对应的employee_no的ASSIGNMENTS记录，从而找到所有指向它的ASSIGNMENTS记录。然后可以利用得自ASSIGNMENTS记录的proj_no导出与EMPLOYEE相关的多个记录。反过来，与一个PROJECT相关的多个EMPLOYEES也可以通过相似的提取过程，通过ASSIGNMENTS文件求得。

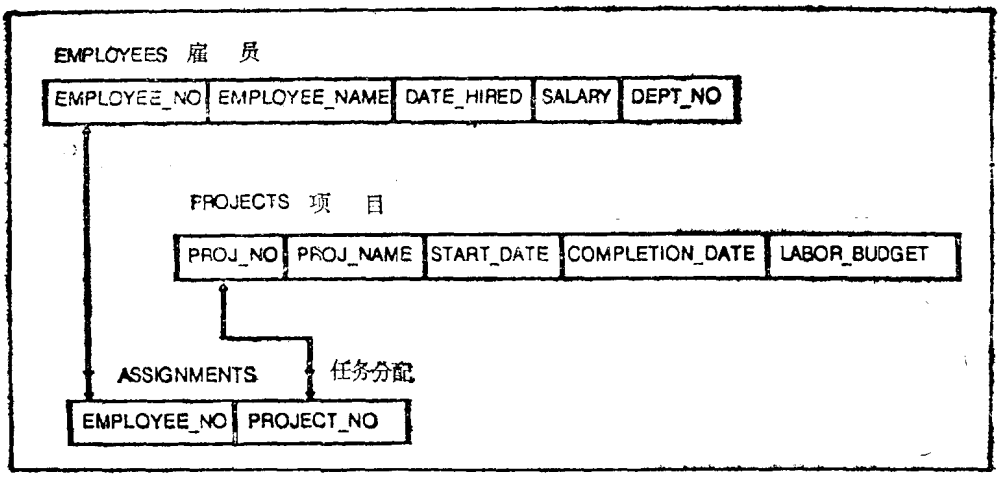

图 2.6 EMPLOYEES和PROJECTS:多对多

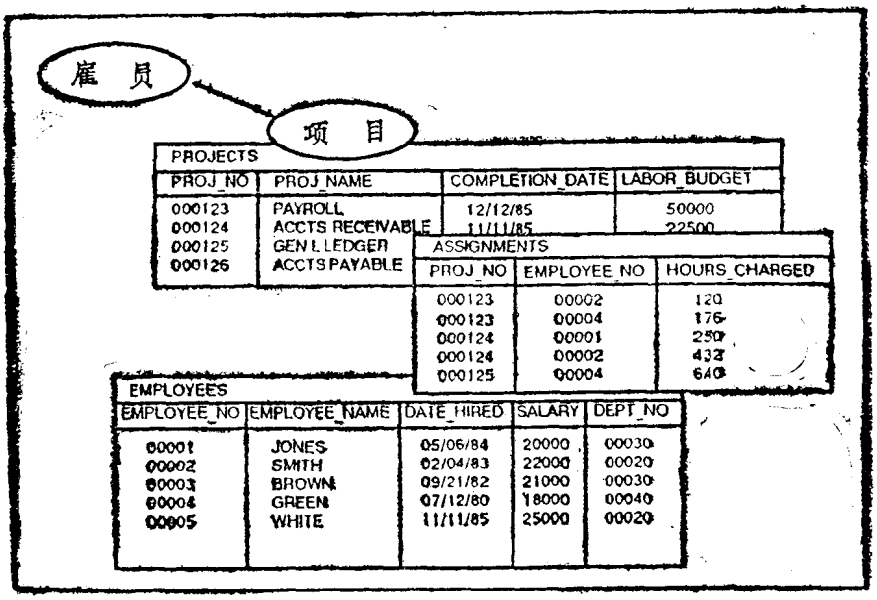

图 2.7 形成多对多关系

图2.7示出含数据值的三个文件。由图可见ASSIGNMENTS文件如何将00123 proj_no与employee_no00002和00004相关联，employee_no 0004与proj_no 00123和 00125相关联。就这样，通过ASSIGNMENTS文件的连接性能建立了一个多对多关系。

这三个关系是我们唯一需要考虑的。任何要存放在数据库中的信息都可以用三种方法之一存贮和维护。

小 结

本章描述了数据库並讨论其部件。下一章讲述常常被误解和叙述过度的如何起草设计数据库。