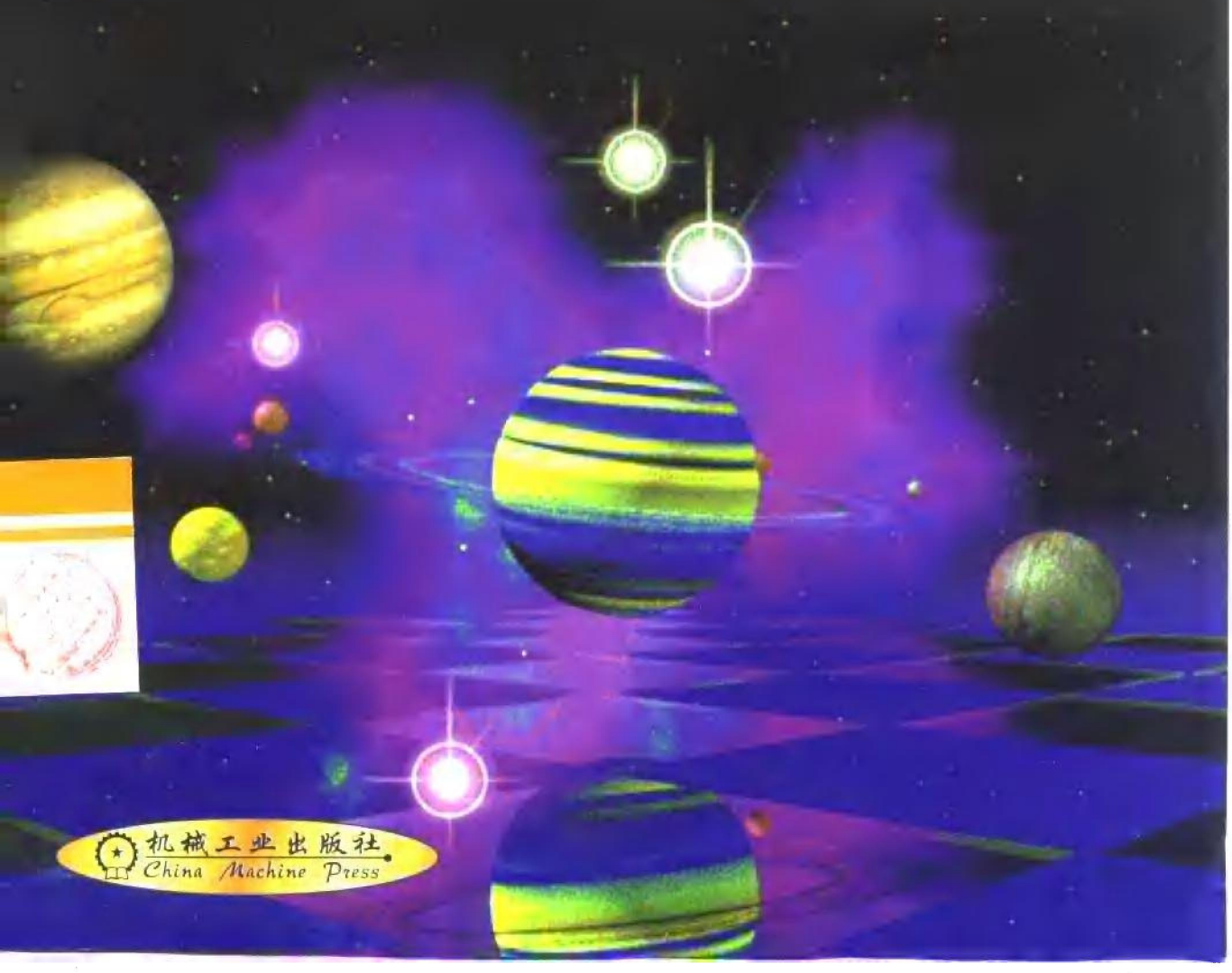
计算机应用二次开发丛书

C++Builder 数据库开发

清宏计算机工作室 编著



机械工业出版社
China Machine Press

计算机应用二次开发丛书

C++ Builder 数据库开发

清宏计算机工作室 编著

机 械 工 业 出 版 社

Borland C++ Builder 4.0 是美国 Inprise 公司（原名为 Borland 公司）最新推出的快速程序开发（Rapid Application Developing，简称 RAD）工具，它具有 RAD 环境下的 C++全部功能，其重要特点是有强大的数据库和网络应用程序开发能力，而且能快速、简单地实现。

本书的重点是数据库应用程序开发。全书分三大部分：C++ Builder 数据库编程基础、C++ Builder 通用数据库编程、C++ Builder 客户/服务器开发，由 11 章组成。主要内容包括关系数据库基本概念、C++ Builder 数据库引擎、设计数据库应用程序、结构化查询语句、使用 Data Access 控件组、使用 Data Control 控件组、使用 Data Cube 控件组、创建快速报表、客户/服务器系统、客户/服务器开发环境的构造、C++ Builder 客户/服务器编程。

本书是从事 Borland C++ Builder 应用和开发程序员的参考书。

图书在版编目（CIP）数据

C++Builder 数据库开发/清宏计算机工作室编著.—北京：机械工业出版社，2000．3

（计算机应用二次开发丛书）

ISBN 7-111-07862-4

Ⅰ.C… Ⅱ.清… Ⅲ.关系数据库-数据库管理系统，Builder

Ⅳ.TP311.138

中国版本图书馆 CIP 数据核字（2000）第 02575 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）

责任编辑：边 萌 李铭杰 封面设计：姚 毅

责任印制：何全君

三河市宏达印刷厂印刷·新华书店北京发行所发行

2000 年 2 月第 1 版第 1 次印刷

787mm×1092mm 1/16·25.25 印张·607 千字

0 001--5000 册

定价：39.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

本社购书热线电话（010）68993821、68326677-2527

丛书前言

随着计算机技术的飞速发展，计算机应用及开发得到迅速推广，多种编程开发语言不断为人们掌握和使用。当前，有一大批从事计算机应用开发的人员对于某一种或某些开发工具的掌握已经达到了一定水准，但他们要进一步提高开发能力，就迫切地需要高层次的开发参考书。为了满足这些计算机开发人员的需求，进一步提高他们的技术水平，推动我国计算机事业的发展，我们特地组织编写了这套“计算机应用二次开发丛书”，适于中高级计算机开发人员学习。

本套丛书主要以常用的开发工具为基础，吸收成功的开发经验和技巧，结合开发原理，向读者介绍了如何有效地利用这些基本开发工具，开发实用性较强的应用程序。使读者不仅知道该怎样做，还知道为什么这样做，达到触类旁通、举一反三的目的。

本丛书重点明确、图文并茂、实用性强，每章除了讲解开发原理之外，还附有大量实例，这些实例绝大多数都是作者在开发工作中的实际成果或其中的一部分，因此，适用性非常强。

由于本丛书的读者定位比较高，适于已经具备一定开发能力的读者阅读。因此在使用这套书之前，要求读者对相应的开发工具已经初步了解。

在这套书中，我们首先推出了《C++ Builder 数据库开发》、《C++ Builder 多媒体开发》、《Delphi 数据库开发》和《AutoCAD 工程二次开发》4 本书。此后，我们还会相继推出其它图书，以满足广大计算机开发人员的需求。

为了把这套丛书编写得更好，我们真诚地希望广大读者提出宝贵意见和建议。

编 者 的 话

Inprise（Borland）公司的拳头产品 Delphi 自问世以来，就一直以其强大的数据库开发功能扮演着“VB 杀手”的角色，在 Microsoft 产品一统天下的开发工具领域内独树一帜，吸引了大批从事数据库开发工作的程序员。而 Inprise(Borland)公司全新推出的Borland C++ Builder 不仅秉承了 Delphi 数据库开发的强大功能，而且对目前流行的面对对象的设计方法和 C++语言进行了集成，从而结束了长期以来广大 C++程序员没有快速应用开发（RAD）产品可用的尴尬局面，成为当今最热门的开发工具之一。

C++ Builder 的最大特点就是面向对象和可视化编程。在面向对象领域，Inprise（Borland）公司一直处于世界领先水平。C++ Builder 4.0 面向对象的优越性主要体现在三个方面：一是编程语言最接近 ISO C++标准；二是全面支持基于组件的应用程序开发；三是借助于对象库可实现代码重用。C++ Builder 的组件库中提供了 100 多个组件，其中包含了 Windows 95 用户界面所用到的全部组件；还包含了两组开发数据库程序的专用组件，可使用简单的鼠标拖点方式进行各种数据库连接，并可将数据库中的数据以各种符合要求的不同方式出现在用户界面中，同时还提供了一组 Internet 组件和一组 ActiveX 组件。使用 C++ Builder 这种可视化开发工具，仅需几周时间学习便可着手进行开发工作。另外，利用鼠标拖点进行应用程序设计的开发方式可以节省大量的时间，大部分代码可以自动生成，将开发人员从繁重的代码编写工作中解放出来。而且，使用 C++ Builder 这种以组件为中心的可视化开发工具编写的应用程序，可在长期的系统维护工作中得心应手、轻松自如。

更让人兴奋的是 C++ Builder 提供了所有 VCL 组件的原始程序代码，可以随意修改它们以符合实际需要。此外，Borland C++ Builder 本身也提供了简单的用户界面，借助这些可视化组件特性可以轻松地生成新的组件。

C++ Builder 具有强大的数据库处理能力。C++ Builder 的数据感知组件中有 20 多个可以直接使用。在许多情况下，甚至不需要编写任何的程序代码便可以产生一个复杂的应用程序。C++ Builder 提供的数据感知组件具有很高的实用性，但是如果需要更高的灵活性，也可以直接调用 Borland 数据库引擎 BDE/IDAPI 以实现更高的追求。

此外，Borland C++ Builder 提供了强大的 Borland 数据库引擎，这是一种非常成熟的数据库连接技术。它提供了三种访问数据库的方式：一是可以直接存取 dBase、FoxBase、Foxpro、Paradox 等文件型数据库生成的 DB、DBF 文件；二是提供了一个标准的 ODBC 接口，通过这个接口可以存取任何一种支持 ODBC 的数据库；三是它提供了一个高效的 SQL Links 数据库驱动程序，允许直接存取 Oracle、Informix、SyBase、MS SQL Server、DB2 和 Borland 的 InterBase。而 SQL Links 是一种速度非常快的数据库驱动程序，使用它可以拥有效率非常高的数据库存取能力。

C++ Builder 4.0 在传统的单层数据库应用和双层数据库应用体系的基础上，率先引入了多层数据库应用模型，极大地拓展了 C++ Builder 的数据库应用空间。通过 C++ Builder 4.0 提供的多层分布式应用服务（MIDAS）可以轻松地开发出高可靠性、高效率、高负载的分布式数据处理系统。

本书全面深入地介绍了运用 Borland C++ Builder 4.0 进行数据库开发的各方面知识。为了加深读者对本书内容的理解，书中提供了大量实例，这些程序都已通过调试运行，而且很多是直接从实际项目中精选出来的。

本书适合于已了解 C++并具有 C++ Builder 基本知识的读者，对 C++ Builder 的程序员来说，本书是对相关手册和帮助文件的一个重要补充，也是一本非常有益的工具书。

由于水平有限，书中难免存在错误与不足，欢迎广大读者批评指教。

编　　者

第 1 章　关系数据库的基本概念

1.1　数据库概念

在这一节中将简要地介绍一些与数据库相关的基本概念。

首先在系统地介绍关系数据库的基本概念之前，我们首先应该理解的一个问题是：什么是数据库。

所谓数据库就是一个通用化的综合性的数据集合。它可以供多种用户共享并且可以达到最小的数据冗余度和较高的数据与程序的独立性。

发展数据库技术正是由于它能够有效地进行数据处理。数据处理就是指对各种形式的数据进行收集、存储、运算（加工）、传播（分发）等一系列活动的总和。它的最终目的就是从大量的客观实际中的数据提取、抽象、推导出对人们有价值的信息，作为行为行动和决策的准则或依据，并且有效地保存和管理复杂数据，从而方便人们使用宝贵的信息资源。数据处理的中心问题就是数据管理。数据管理就是指有效地对数据分类、组织、编码、存储、检索和维护。

数据库管理系统（Data Base Management System 简称 DBMS）就是可使多种程序并发地使用数据库的一个软件系统，它能够有效及时地处理数据，并保证数据的完整性和安全性。

1.1.1　数据库的特点

- 面向全组织的复杂的数据结构。
- 具有最小的数据冗余度，易扩充。
- 具有较高的数据和程序的独立性。
- 统一的数据控制功能。
- 数据的最小存取单位是数据项。

1.1.2　数据模型

数据模型是数据库系统中用于提供信息表示和操作手段的形式构架。数据模型一般分为两个层次：概念模型和数据模型（这里的数据模型指具体的模型，如网状模型、层次模型、关系模型）。前者是从用户的角度对数据和信息建立模型，后者是从计算机的角度对数据建立模型。

数据模型是严格定义的概念的集合。这些概念精确地描述了系统的静态特性、动态特性和完整性约束条件。数据模型一般由三个部分组成：数据结构、数据操作、完整性约束。这就是所谓的数据模型的三要素。

数据结构指所研究的对象类型的集合。对象是数据库的组成成分。一般分成两类数据对象：一类是与数据类型、内容、性质有关的对象,我们可以称之为数据对象；另一类是与数据之间联系有关的对象，我们可以称之为关系对象。

数据操作指对数据库中各种对象的实例允许执行的操作的集合，包括操作及操作规则。数据库的数据操作主要有检索和更新两大类。

数据的约束条件是完整性规则的集合。完整性规则是数据之间相互关系所存在的制约条件，用于限定数据状态和数据变化，从而保证数据的准确、有效、相容。

1.1.3 数据库系统的三级模式结构

三级模式结构是一般数据库系统普遍采用的体系结构。它由外模式、模式和内模式组成。

外模式是从用户角度看到的数据视图。

模式是数据库中全体数据的逻辑结构和特征的描述。

内模式是数据在数据库系统中的表示。

数据库系统的三级模式是三个数据抽象级别，它将对数据的具体组织交给 DBMS 管理，从而用户可以抽象地处理数据。

为了实现三个抽象层次数据的转换，一般提供两个映像：内模式/模式映像，模式/外模式映像。

1.2 关 系 模 型

关系模型是数据模型中最重要的模型。它是建立在数学概念基础上的。通俗地讲，在关系模型中，数据从用户的角度来看就是一张二维表。下面非形式地介绍一下关系模型中的一些基本概念：

关系对应于一张表。

表中的一行成为一个元组。

表中的一列成为属性，每一列的名字成为属性名。

主码就是表中的一个属性组，它们的值能唯一标识一个元组。

域指属性的取值范围。

分量指元组中的一个属性值。

关系模式是对关系的描述，用关系名{属性名 1，属性名 2，…，属性名 n}来表示。

关系模型的特点就是：

- 概念单一简单，实体与实体之间的联系都用单一的关系来表示。
- 每个分量是不可分割的。
- 具有严格理论基础的集合运算。

1.2.1 数学定义

定义 1 域（Domain）—— 值的集合。

例如，整数，{1，2，3，4}，A 公司中所有年龄 20 岁到 30 岁的职工的集合，长度等于 6 的字符串的集合等等都可以是域。域可以是有限集，也可以是无限集。

定义 2 n 元组（n-tuple）—— 给定一组域 $D_1, D_2, \ldots, D_n$ 的笛卡儿积为 $D_1 * D_2 * \ldots * D_n = \{(d_1, d_2, \ldots, d_n) \mid d_i \in D_i, i=1,2,\ldots, n\}$，其中每个元素 $(d_1, d_2, \ldots, d_n)$ 称为一个 n 元组，简称元组。元组中每个值 d_i 称为一个分量（component）。域 $D_1, D_2, \ldots, D_n$ 可以是相同的。

笛卡儿积可以表示为一个二维表。

定义 3　基数（Cardinal number）—— 若 Di（i = 1，2，…，n）为有限集，其基数分别为 m_i（i = 1，2，…，n），则 $D_1 * D_2 * \ldots * D_n$ 的基数为 $m = \Pi m_i$

定义 4　关系（Relation）——$D_1 * D_2 * \ldots * D_n$ 的子集称为域 $D_1, D_2, \ldots, D_n$ 上的关系。用 R（$D_1, D_2, \ldots, D_n$）表示，其中 R 表示关系名，n 是关系的目或度（Degree）。每个元素是关系中的元组，通常用 t 表示。当 n = 1 时，称为单元关系；当 n = 2 时，称为二元关系。

定义 5　属性（Attribute）—— 关系是一个二维表，表的每行对应一个元组，表的每列对应一个域。由于域可以相同，为了加以区分，对每列起一个名字，称为属性，n 目关系有 n 个属性。

例如，给出三个域：

D_1 = 教师集合（TEACHER）= { 张平，李军，赵明 }

D_2 = 课程集合（COURSE）= { 高等代数，空间几何 }

D_3 = 参考书集合（REFERENCE）= {《高等代数论》，《空间几何论》}

则笛卡儿积为 $D_1 * D_2 * D_3$ = {（张平，高等代数，《高等代数论》），（张平，高等代数，《空间几何论》），（张平，空间几何，《高等代数论》），（张平，空间几何，《空间几何论》），（李军，高等代数，《高等代数论》），（李军，高等代数，《空间几何论》），（李军，空间几何，《高等代数论》），（李军，空间几何，《空间几何论》），（赵明，高等代数，《高等代数论》），（赵明，高等代数，《空间几何论》），（赵明，空间几何，《高等代数论》），（赵明，空间几何，《空间几何论》）}

$D_1 * D_2 * D_3$ 共有 3*2*2 = 12 个元组，这 12 个元组的总体可列成一个二维表。但是并不是每个元组都有现实意义。如果张平只会教高等代数，则（张平，空间几何，《高等代数论》），（张平，空间几何，《空间几何论》）是没有意义的，因为张平不能教空间几何。另外，高等代数这门课程的参考书只能选择《高等代数论》，所以像（张平，高等代数，《空间几何论》）这样的元组也没有现实意义。在实际的数据库开发中，应该只讨论有意义的部分，二维表中也应该只包含有意义的元组。

现在假设张平只会教高等代数，赵明只会教空间几何，李军既能教高等代数也能教空间几何。高等代数的参考书是《高等代数论》，空间几何的参考书是《空间几何论》。下面从上述的笛卡儿积中取出有意义的子集作为一个关系，关系名为“COURSE SETTING（课程设置）”。这个关系可以表示为

COURSE SETTING（TEACHER，COURSE，REFERENCE），其中 TEACHER 表示教该课程的教师，COURSE 表示课程名，REFERENCE 表示该课程使用的参考书。在该例子中，课程和参考书是一一对应的，即一个课程对应一部参考书并且一部参考书只对应一个课程；课程和教师是一对多的，即一个课程可以有多个授课教师，如表 1-1 所示。

表 1-1　COURSE SETTING

TEACHER	COURSE	REFERENCE
张平	高等代数	《高等代数论》
李军	高等代数	《高等代数论》
李军	空间几何	《空间几何论》
赵明	空间几何	《空间几何论》

假如，D_3 = 参考书集合（REFERENCE）

= {《高等代数论》，《高等代数导论》，《空间几何论》}，高等代数这门课程有两本参考书《高等代数论》和《高等代数导论》，在数据库中，是否可以将关系 COURSE SETTING 定义成如表 1-2 所示的形式呢？

表 1-2 COURSE SETTING

TEACHER	COURSE	REFERENCE	
		FIRST	SECOND
张平	高等代数	《高等代数论》	《高等代数导论》
李军	高等代数	《高等代数论》	《高等代数导论》
李军	空间几何	《空间几何论》	
赵明	空间几何	《空间几何论》	

在数据库中，我们要求关系的每个分量必须是不可分的数据项，并把这样的关系称为规范化的关系，简称为范式（Normal Form）。上面的关系中 REFERENCE 这个分量可以取两个值，这不符合规范化的要求，因此这样的关系在数据库中是不允许的。应该改为如表 1-3 所示。

表 1-3 COURSE SETTING

TEACHER	COURSE	REFERENCE
张平	高等代数	《高等代数论》
张平	高等代数	《高等代数导论》
李军	高等代数	《高等代数论》
李军	高等代数	《高等代数导论》
李军	空间几何	《空间几何论》
赵明	空间几何	《空间几何论》

在该关系中，课程和参考书不再是一一对应的，而是一对多对应的。

另外，域名和属性名可以不一致，特别是当多个属性取自同一个域时，由于属性名不能相同，这时必然会出现域名和属性名不同的情况。在上面的例子中，假如将第三个域 D_3 的名字改为 BOOK，即 D_3 = 参考书集合（BOOK）= {《高等代数论》，《空间几何论》}，表 COURSE SETTING 中仍用 REFERENCE 这个名字，这样就需要在模式中给出映像，说明这个属性来自某某域，如 DOM（REFERENCE）= BOOK。

总之，数据库中的关系有以下性质：

- 列是同质的（Homogeneous），即每一列中的分量是同一类型的数据，来自同一个域。
- 不同的列可出自同一个域，每一列称为属性，要给予不同的属性名。
- 列的顺序无所谓，即列的顺序可以任意交换。
- 任意两个元组不能完全相同。
- 行的顺序无所谓，即行的顺序可以任意交换。
- 每个分量必须是不可分的数据项。

1.2.2 关系模型

关系模型由三部分组成：数据结构、关系操作集合、关系的完整性。下面分别讨论这三个方面。

1. **单一的数据结构——关系** 在关系模型中，无论是实体还是实体之间的联系均由

单一的结构类型即关系来表示。下面介绍码、关系模式、关系数据库等基本概念。

(1) 码（Key）　关系中的某一属性组，如果它的值能唯一地表示一个元组，则称该属性组为候选码（Candidate key）。

若关系中有多个候选码，则选定其中一个作为主码（Primary key）。主码的诸属性称为主属性。

(2) 关系模式　关系的描述称为关系模式。它包括关系的诸属性名、属性域的映像、属性间数据的依赖关系等。通常，一个关系是由赋予它的元组的语义来确定的，元组的语义实际上是一个 n 目谓词（n 是属性集中属性的个数）。凡使该 n 目谓词为真的笛卡儿积中的元素（或者说凡符合元组语义的元素）的全体就构成了该关系模式的关系。不同时刻，关系模式也可能有所变化。某个时刻对应的关系模式的内容成为相应模式的状态，它是元组的集合，成为关系。关系模式和关系常常统称为关系，可从上下文中加以区别。

(3) 关系数据库　关系数据库有型和值的概念。关系数据库的型即数据库描述，它包括若干域的定义以及在这些域上定义的若干关系模式。数据库的值是这些关系模式在某一时刻对应的关系的集合。数据库的型亦称为数据库的内涵（Intention）。数据库的值亦称为数据库的外延（Extention）。

关系模式是稳定的，而关系是随时间不断变化的，因为数据库中的数据在不断变化。

2. **关系操作**　关系模型给出了关系操作的能力和特点，但不对 DBMS（数据库管理系统）的语言作具体的语法要求。关系语言的特定是高度的非过程化，这也是它的优点。用户不必请求 DBA（数据库管理员）为他建立特殊的存取路径，存取路径的选择是由 DBMS 的优化机制来完成的。此外，用户也不必求助于循环、递归来完成数据操作。

早期的关系操作能力是用两种方式来表示的：代数方式和逻辑方式，即关系代数和关系演算。这两种方式是等价的。

关系模型中，关系操作的能力可用关系代数来表示。下面列出常用的几种：

- θ 选择（Theta select）
- 投影（Project）
- θ 连接（Theta join）
- 除（Divide）
- 并（Union）
- 交（Intersection）
- 差（Set difference）

其中 θ 表示大于、小于、等于、不等于、大于或等于、小于或等于这些比较运算符中的一种。

关系操作方式的特定是集合操作，即操作的对象和结果是集合。这种操作方式也成为一次一集合（set-at-a-time）的方式。非关系型的数据操作方式则为一次一记录（record-at-a-time）的方式。

3. **关系模型的三类完整性**　关系模型的三类完整性是实体完整性、参照完整性和用户定义的完整性。

实体完整性和参照完整性是关系模型必须满足的完整性约束条件，应该由关系系统自

动支持。

(1) 实体完整性（Entity Integrity） 如果属性 A 是基本关系 R 的主码组成成分（主属性），则属性 A 不能取空值。

关系数据库中有各种关系，如基本关系（常称为基本表）、查询表、视图表等。基本表是实际存在的表，它是实际存储数据的逻辑表示。查询表是查询的结果所对应的表。视图表是由基本表或视图表导出的表，是虚表，不对应实际存储的数据。实体完整性是针对基本关系而言的。

对于实体完整性说明如下：

① 一个基本关系通常对应现实世界的一个实体集。如学生关系对应学生的集合。

② 现实世界中实体是可区分的，即他们具有某种唯一性标识。

③ 关系模型中由主码作为唯一性标识。

④ 主码不能取空值（即“不知道”或“无意义”的值）。因为主码取空值说明某个不可标识的实体，而这和第二点矛盾，即不存在这样的实体，从而取名“实体完整性”

(2) 参照完整性（Referential Integrity） 若基本关系 R 中含有与另一个基本关系 S 的主码 Ks 相对应的属性组 F（F 称为 R 的外部码），则对于 R 中的每个元组在 F 上的值必须为：

① 或者取空值（F 的每个属性值均为空值）。

② 或者等于 S 中某个元组的主码值。

关系 S 的主码 Ks 和 F 定义在同一个（或一组）域上。基本关系 R、S 不一定是不同的关系。

例如，职工关系 EMP（ENO，ENAME，DNO）和部门关系 DEPT（DNO，DNAME）是两个基本关系。EMP 的主码是 ENO，DEPT 的主码是 DNO。在 EMP 中，DNO 是它的外部码。EMP 中的每个元组在 DNO 上的值允许有两种可能：

- 取空值。这表明该职工尚未分配到某个部门；
- 若非空值，则 DNO 的值必须是 DEPT 中某个元组中的 DNO 值。表明该职工不可能分配到一个不存在的部门中，即被参照的关系 DEPT 中一定存在一个元组，它的主码值等于该关系 EMP 中的外部码值，这就是参照完整性。

(3) 用户定义的完整性 实体完整性和参照完整性可用于任何关系数据库系统。而用户定义的完整性则是针对某个具体的数据库的约束条件，由应用环境决定，它反映某个具体应用所涉及的数据必须满足的语义要求。关系模型应提供定义和检验这类完整性的机制，以便用统一的系统的方法处理它们而不是由应用程序来承担这一功能。

1.3 表 结 构

正如上面所述，表的数学概念是笛卡儿积的子集。从用户观点来讲，表是存储和操作某一类关系数据的逻辑结构。表具有表名。

表中的一行称为一个元组，也就是通常所说的一条记录，一列称为一个属性，也可简单地称为一个字段。同一个表中的属性具有不同的属性名，即字段名必须是唯一的。属性

的值取自域，在关系模型中要定义属性到域的映射。表中的每个元素都必须是不可分的数据项。

表中的行顺序和列顺序可以随意安排。能够唯一标识一个元组（即一条记录）的属性组（字段组合）称为候选码，确定一个候选码作为该表的主码。主码是唯一的，即在一个表里不可能出现两个主码相同的元组。

一个表可以有也可以没有外部码。

1.4　数据库引擎

数据库引擎是应用程序与数据库交互的接口，负责提交、处理数据库访问请求并且返回处理结果，在客户端，通常由一系列用于进行数据库访问的数据结构和基于这些数据结构的方法（函数库）组成，例如 BDE（Borland Database Engine）就是一个高效的数据库访问接口。在服务器端，从响应和处理数据库访问请求的方式来看，数据库引擎有两种基本的体系结构。

1.4.1　多进程数据库引擎

其特点是依靠多个独立的处理进程来完成数据库操作。其结构如图 1-1 所示。在这种体系结构下，每个用户提交数据库访问请求都启动了数据库引擎本身的不同实例。为了支持多用户访问同一个数据集，数据库处理进程和其它全局管理程序同时工作以协调不同用户间的同步和互斥。这种数据库引擎下的应用程序使用一种专门的内部进程通信（Internal Process Communication，简称为 IPC）设备进行通信，例如 Microsoft Windows 下的动态数据交换（DDE）就是一种 IPC。

多进程数据库引擎的典型使用是在大型机数据库系统中。因为大型机的多重虚拟操作系统（Multiple Vitual System，简称 MVS）为应用程序提供了 IPC 设备，所以在基于 MVS 的机器上很流行。

多进程数据库引擎的代表是 Oracle 公司的 Oracle Server。Oracle Server 配置了 16 种不同的程序来执行不同的任务。用户每次连接到 Oracle 数据库时就启动了 Oracle 数据库处理程序的不同实例。数据库访问请求被传送到哪个进程，该进程就负责处理这一请求并返回结果集、执行加锁与解锁和执行其它数据访问功能。

典型情况下，这种数据库引擎与其它类型相比会消耗较多的系统资源，但是相对来说更容易扩展到大型的平台。

多进程数据库引擎有两个很重要的优点：

- 一个数据库可以同时支持多个用户，在网络上提供数据集中化。
- 通过在物理机器上增加更多的 CPU 来扩展数据库规模。

不过其缺点也是很明显的，一是占用的系统资源较多，成本较高；二是在较低的硬件配置下效率不高。

1.4.2　单进程、多线程数据库引擎

其结构如图 1-2 所示。其特点是在一个处理进程内创建多个线程来响应多个数据库访问请求，通常用于 SQL Server。这种结构依赖于同一应用程序内的多线程并发工作，而不是为

每个任务创建不同的应用程序。从理论上来说，这种机制将提供更大的可移植性，因为数据库系统本身要管理各个任务的调度、内存和磁盘的访问等等。

它的优点是对系统资源的占用较少，对硬件配置要求较低，在一般的配置下也能发挥较高的效率，但是扩展性较差。

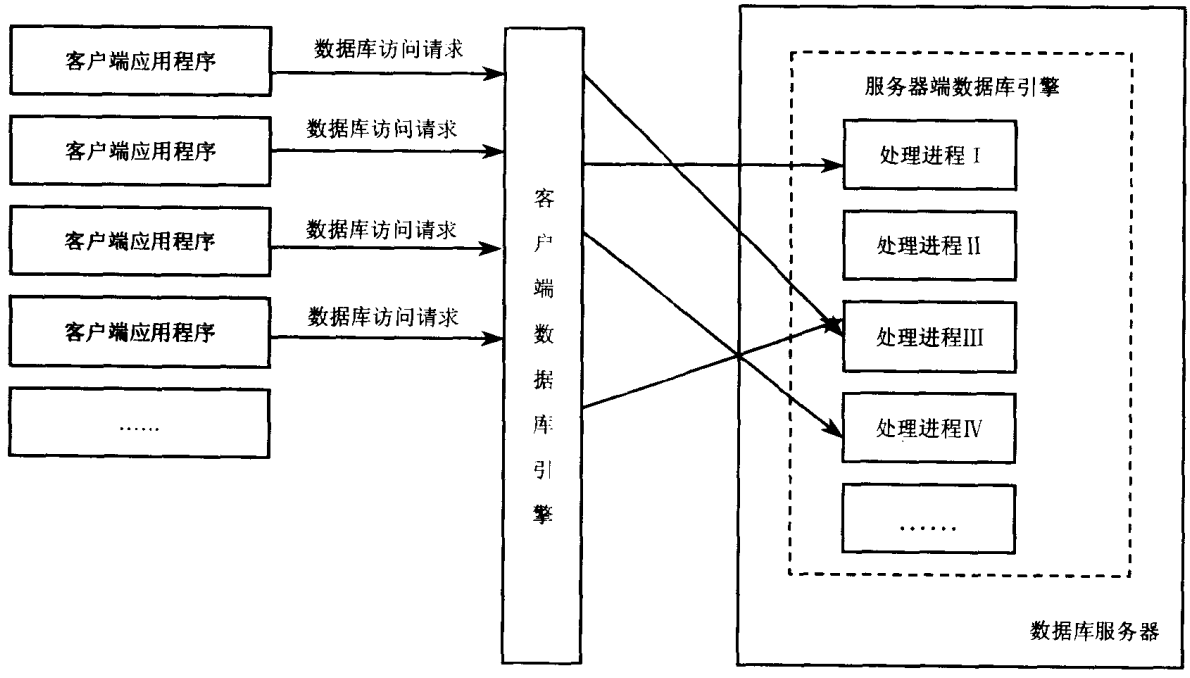


图 1-1 多进程数据库引擎

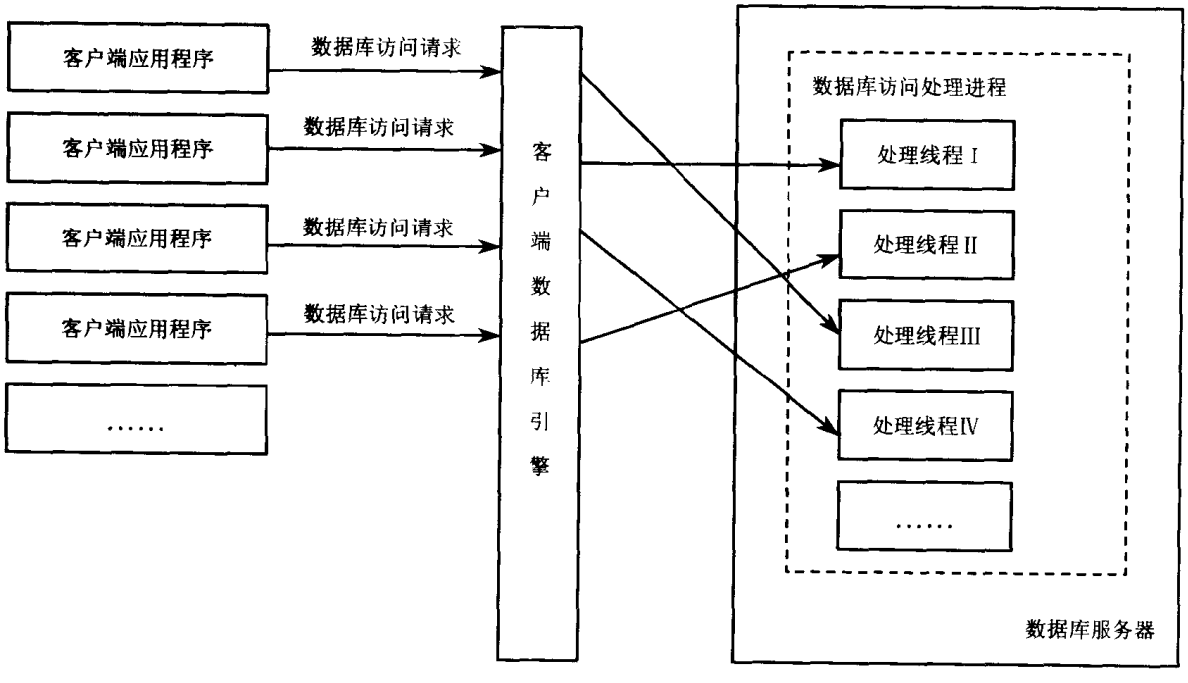


图 1-2 单进程、多线程数据库引擎

1.5　设计关系数据库

1.5.1　数据库设计的概述

人们在总结信息资源开发、管理和服务的各种手段时，认为最有效的是数据库技术。数据库的应用已越来越广泛。从小型的单项事物处理到大型的信息系统大都采用先进的数据库技术来保证系统数据的整体性、完整性和共享性。

数据库设计是研制数据库及其应用系统的技术，是数据库在应用领域中主要的研究课题。数据库设计是指对于一个给定的应用环境，构造最优的数据库模式，建立数据库及其应用系统，使之能够有效地存储数据，满足各种用户的需求（信息要求和处理要求）。数据库设计通常是在一个通用的 DBMS 支持下进行的，即利用现成的 DBMS 为基础。

从使用者的角度看，信息系统是提供信息，辅助人们对环境进行控制和进行决策的系统。数据库是信息系统的核心和基础。它把信息系统中大量的数据按一定的模型组织起来，提供存储、维护、检索数据的功能，使信息系统可以方便、及时、准确地从数据库中获得所需的信息。一个信息系统的各个部分能否紧密地结合在一起以及如何结合，关键在数据库。因此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开发出完善而高效的信息系统。数据库设计是信息系统开发和建设的重要组成部分。

大型数据库的设计是一项庞大的工程，属于软件工程的范畴，必须将软件工程的原理和方法应用到数据库建设中来。对于从事数据库设计的人员来讲，应该具备多方面的技术和知识，主要有：

- 计算机科学基础知识和程序设计技术
- 数据库基础知识和数据库设计技术
- 软件工程的原理和方法
- 应用领域的知识

1.5.2　数据库设计的特点

数据库设计既是一项涉及多学科的综合性技术又是一项庞大的工程项目。数据库设计质量的好坏直接影响系统中各个处理过程的性能和质量。有人讲“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。技术与管理的界面（称之为“干件”）十分重要。数据库建设是硬件、软件和干件的结合。这是数据库设计的特点之一。

数据库设计应该和应用系统设计相结合。也就是说，整个设计过程中要把结构（数据）设计和行为（处理）设计密切结合起来。这是数据库设计的特点之二。

在数据库设计中应该把结构特征和行为特征相结合，既要致力于数据模型和建模方法的研究，着重结构特性的设计，又要对行为设计提供指导。

1.5.3　数据库设计的方法

数据库设计好似一种技艺而不是工程技术，缺乏科学的理论和工程原则的支持，很难保证质量。如果数据库投入使用后才发现问题，就不得不进行修改，维护代价昂贵。由此可见，数据库设计十分重要。数据库设计方法运用软件工程的思想和方法，提出了各种设计准则和规程，都属于规范设计法。

规范设计法中比较著名的有新奥尔良（New Orleans）方法，它将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（数据库分析和定义）、逻辑设计（设计实现）和物理设计（物理数据库设计）。其后，S.B.Yao 等又将数据库设计分为五个步骤。

基于 E-R 模型的数据库设计方法、基于 3NF（第三范式）的设计方法，基于抽象语言规范的设计方法等等，是在数据库设计的不同阶段上支持事项的具体技术和方法。

规范设计法从本质上看仍然是手工设计方法，其基本思想是过程迭代和逐步求精。

计算机辅助数据库设计，目前还是在数据库设计的某些过程中模拟某一规范设计方法，并以人的知识或经验为主导，通过人机交互方式实现设计中的某些部分。

从目前技术条件来看，按照一定的设计规程，用工程化方法设计数据库是最实用的方法。

1.5.4 数据库设计的步骤

按规范设计的方法将数据库设计分为以下六个阶段（如图 1-3 所示）：

- 需求分析
- 概念结构设计
- 逻辑结构设计
- 数据库物理设计
- 数据库的实施
- 数据库运行和维护

这个图表是从数据库应用系统设计和开发的全过程来考察数据库设计的问题。因此它既是数据库也是应用系统的设计过程。

在设计过程中努力把数据库设计和系统其它成分的设计紧密结合，把数据和处理的需求收集、分析、抽象、设计、实现在各个阶段同时进行，相互参考，相互补充，以完善两方面的设计。按照这个设计过程，结构特性设计部分形成了数据库的各级模式，如图 1-4 所示。

下面将具体讨论数据库设计阶段的几个设计步骤：

1. **需求分析阶段** 需求收集和分析是数据库设计的第一阶段，明确地把它作为数据库设计的第一步十分重要。这一阶段收集到的基础数据和一组数据流图（Data Flow Diagram 简称为 DFD）是下一步设计概念结构的基础。

从数据库设计的角度考虑，需求分析阶段的目标是：对现实世界要处理的对象（组织、部门、企业等）进行详细调查，在了解原系统功能的过程中，收集支持系统目标的基础数据及其处理。

(1) 调查的重点是“数据”和“处理”，要获得用户对数据库的如下要求：

1) 信息要求。即在数据库中需存储哪些数据。

2) 处理要求。用户要完成什么处理功能，对某种处理要求的响应时间、处理方式等。

3) 安全性和完整性的要求。

(2) 具体做法

1) 了解组织机构情况。调查这个组织由哪些部门组成，各部门的职责是什么，为分析信息流程做准备。

2) 了解各部门的业务活动情况。调查各部门使用和输入什么数据，如何加工处理这

些数据，输出什么信息，输出到何部门，输出结果的格式是什么。

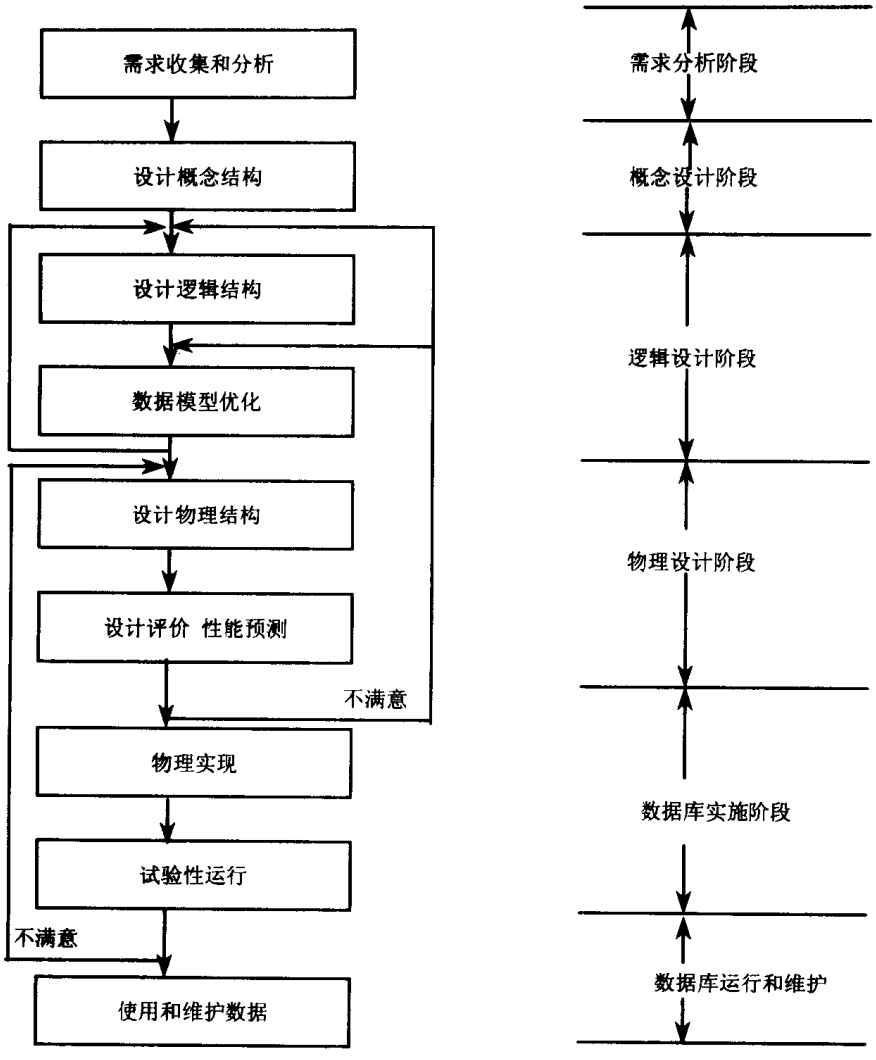


图 1-3　数据库设计阶段

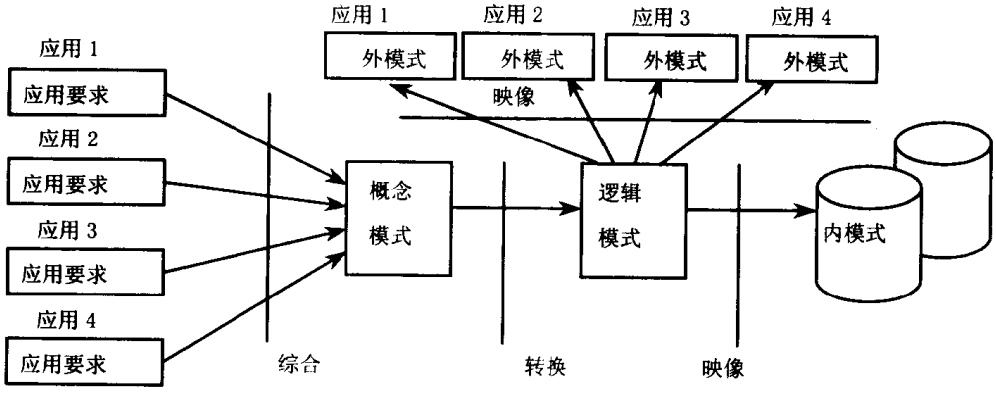


图 1-4　结构特性设计模式