

# 第一部分 数据库管理的原则

## 第1章 数据库、DBMS原理和关系型模型

本章要点：

了解数据库

了解DBMS

了解RDBMS

### 1.1 了解数据库

在过去的许多年里，有许多关于“数据库”这个名词的定义。数据库是一个服务于一个核心目标的数据的有组织的集合。数据库中的数据是有组织的，从某种意义上说，数据库中存储的数据采用一种不变的方式被存储、格式化、存取以及显示。因为数据库不含有无关的或冗余的数据，它可以适用于一个核心目标。一本电话簿就是一个很好的数据库例子，它包含有关的数据（名字），让人们能够查找电话号码；它不包含无关的数据，如某人的电话机的颜色；它只贮存那些与它的目标相关的信息。最常见的，一个数据库的目标是商务应用，但是也可能贮存科学、军事或其他数据，这些数据通常不能当作商务数据看待。因此，有商业数据库、科学数据库、军事数据库以及其他的数据库等等。另外，数据不仅能根据它的应用分类，还能根据它的格式分类，现代数据库包括多种类型的数据。例如，现在数据库贮存图像、图表、声音、视频或包括两种或多种类型的复合文档，已经是很普通的事了。

当讨论数据库特别是数据库设计时，通常称数据库所服务的核心目标为它的业务（business），而不管它属于什么特殊的领域，如太空、生物医学或其他。此外，在实际生活中，你会发现数据库通常有它明确的业务应用。

在早些年，编写程序实现自动数据处理（Automatic Data Processing，ADP）要求的程序员发现，在每次运行中，他们需要频繁地贮存数据，这就是通常所说的对永久内存的需要，即从程序的一次运行到下一次运行时，需要将数据存留或贮存。这个基本的需要成为数据库发展的开端。其次的需要——简单的数据存储，也促进了数据库的产生。在线归档和历史数据就是一对特别的例子。尽管文件、目录和文件系统能满足多数普通的数据存储需要（包括索引变化），但数据库可以做文件系统能做的工作并且还可做更多的。

现代数据库通常为上级组织或企业的部门，或它们的小型组织单位实现存贮处理需求。因此，你用术语“企业范围（enterprisewide）”来指代整个组织的商务范围，用“部门范围（departmentwide）”来指代一个部门级的范围，用“工作组（workgroup）”指一个部门内的一些单位。通常，在部门范围和工作组级上查找数据库。

偶尔，你会查找服务于企业范围的数据库，例如工资单和人事数据库，但是它们在数量上远远少于那些较小的数据库。实际上，当几个部门的数据库放在一起或合并成为一个大数

据库时，这就是建立一个数据仓库（Data Warehouse，DW）的本质。那些充当大型数据库的数据源的小型数据库，称为可操作数据库。然而，这不是新东西，一个操作数据库就是产生数据的数据库，多年来一直被叫做成品数据库；只有在建立数据仓库的前提下，你才会发现成品数据库是指可操作数据库或有时是指可操作的数据存贮。随着因特网技术的出现，数据库和数据仓库现在常常作为前端 Web浏览器的后端使用。

当工作组数据库合并起来以满足一个大型部门需要时，结果通常相当于一个数据中心（Data Mart，DW），一个DM就是一个部门规模的数据仓库。和术语“数据库”那样，术语“数据仓库”也会产生多种定义。然而，当你把几个小型数据库集成为一个大型数据库，为一个较广泛的组织服务时，如果该数据库存储历史数据、提供决策支持、提供数据汇总、提供只读数据，并且实质上充当所有向它提供数据的相关成品数据库的数据接收器，那么它通常被看作是一个数据仓库。

另外，如果一个数据库增大的原因是因为该数据库是一个长时期贮存数据的历史数据库（例如一个人口普查数据库），或因为它必须贮存数据的类型（例如一个图像数据库），或者因为它必须采用的贮存数据的频率（例如一个卫星遥测数据库），那么它通常被称为一个非常大的数据库（Very Large Database，VLDB）。

随着时间的发展，由于磁盘容量的增大和价钱的下降、均衡多处理技术机器的出现、冗余廉价磁盘阵列（Redundant Array of Inexpensive Disks，RAID）技术的发展、数据库软件的增多或规模化，判定一个数据库是否为 VLDB的条件不断变化。当前，一个常用的准则是任何 100GB或100GB以上的数据库就可以被看作是一个 VLDB；而就在几年前，10GB就被看作是分隔点了。

## 1.2 理解DBMS

数据库管理系统（Database Management System，DBMS）就是管理一个数据库的软件，它充当所有数据的知识库，并对它的存储、安全、一致性、并发操作、恢复和访问负责。DBMS有一个数据词典（有时被称为系统目录），其中贮存着它拥有的每个事物的数据，例如名字、结构、位置和类型，这种关于数据的数据也被称为元数据（metadata）。在一条数据的生存周期里（从它的创建到删除），这条数据的逻辑和物理信息都被记录在数据词典中。数据库系统管理员（Database Administrator，DBA）应该熟悉DBMS的数据词典；在数据库的整个生命周期内，数据词典为他或她服务。

### 1.2.1 数据的安全

在成品数据库中，安全性一直是一个重点，在开发或测试数据库中也常如此。这通常不是有没有安全性的问题，而是有多大安全性的问题。除了操作系统和网络安全设施外，一个DBMS通常提供几层安全措施。最常见的情形是，DBMS要求用户登录用户帐户的口令，口令被验证为真，才能访问数据库。

DBMS也提供其他的机制，例如组、角色、权限以及简档，这些都提供更加精良的安全措施。提供这些安全级不仅是为了加强安全性，而且为了建立商业安全策略。例如，只有一个经过验证属于航空组的用户才能存取航空数据；又例如，只有经过验证拥有操作者角色的用户才能备份数据库。

### 1.2.2 维护和实施完整性

数据的完整性是指它的一致性和正确性。对于一致的数据，在它所有的出现的场合中，必须以相同的方式建模和实现，对于正确的数据，它必须是正确、精确和有意义的。

DBMS维护完整性的一个方法是在一条数据项发生改变的过程中进行锁定。数据库通常是在数据库页级或行级锁定数据。锁定偶尔也允许并发操作，在下面你将涉及到这种情形。

DBMS实施完整性的另一个方法是：如果一条数据贮存在多个地方，那么把变化复制到这条数据上。DBMS实现完整性的最后一个方法是通过监视输入的或改变的数据值，使它们全部符合要求的规格（例如：一个范围检查）。

如果接着的是合适的建模和实现过程（在后面的第2章“逻辑数据库的设计和标准化”中讨论），DBMS会自动地帮助实施这种完整性，例如，通过一个触发器或约束条件，这两个概念在第28章“PL/SQL基础”中被定义和说明。没有完整性，数据是没有价值的。有了完整性，数据就是信息。完整性不仅能增强数据，它还给予数据以价值。

当DBMS提供多用户存取时，它必须管理并发操作。那就是说，当几个人同时访问相同的数据库（特别是同一条数据）时，DBMS必须保证这种并发访问能够以某种方式实现。并发可以广义上定义为同时发生，即两个或多个用户在同一时期访问同一数据。

DBMS实现并发操作的方法倒不太复杂，但背后的实际程序却很复杂。本质上说，当两个或多个人只想简单地浏览一下同一个数据而不改变它时，一切还好。但当至少一个人想改变数据而其他人想浏览或也想改变数据时，DBMS必须贮存多个备份。当每个人都完成改动后，DBMS必须将所有改变的备份保存为一条正确的数据。

前面已提到并发管理的一种方式加锁。通常来讲，锁越精密（越小），并发性就越好（那就是说，更多的用户无需等待即可同时访问）。行通常比最小的数据库页或块都小，因此，行级锁可较好地管理短小、杂乱数据的处理，块级锁可较好地管理长的、连续数据的处理。

这就是并发性和完整性是如何结合的。当一个人想查看或更改一条数据时，这个人就是执行数据库的一个事务。

### 1.2.3 理解事务

作为DBMS标准的一部分，DBMS有一个事务管理器（transaction manager），它的目的是管理并发操作和确保事务的完整性。事务管理器的工作是艰巨的，因为它必须允许许多人同时访问相同的数据，并且在访问之后要把数据放回到数据库中，就好象在某个时间上只有一个人存取数据，他完成工作后另一个人才工作，这样确保数据的正确性。DBMS解决数据的多个备份的基本方案就在这中间。如果（并且是只有）数据是串行的，那么在保持数据的准确性的同时也进行了事务处理。简单地说，DBMS必须重新整理所有改变，以使得它们的最终结果仿佛发生在一个文件中。

事务是并发或工作的单位。不能产生比一个事务更小或更少的东西，也就是说，没有人能够只完成数据改变工作的一部分。全部事务必须都是原子的，所以每个单独的事务要么完成，要么不完成。直到20世纪现代物理发展起来以前，原子一直被当作物质的最小单位。同样，事务也是并发的最小单位，它要么全有，要么全无。一个被完成的事务可以说是被提交了，没有完成的事务则是被回滚了。

DBMS用事务作为恢复的单位来控制恢复，正常完成、手工要求中止以及意料之外的退

出都要求DBMS重新访问数据的多个备份来提交或回滚数据。为了回滚或前滚，DBMS保持了一个事务日志。回滚是一个撤销操作。前滚是一个重做操作，例如，由于一个硬件或软件错误，一个已提交的事务无法将它所做的操作从内存中存储到磁盘就会发生前滚，DBMS只是简单地重做这个操作。因此，在DBMS中事务恢复的关键是一个事务必须是原子的，而且在必要时可以做、不做或重做。

### 1.2.4 与数据库通信

如果你不能和一个DBMS对话，那么这个DBMS就不是很好的。你可能会问怎样和DBMS对话。可以通过一种存取或查询语言访问数据库。结构化查询语言（Structured Query Language, SQL）是当今主要的查询语言，它主要用于管理主流类型的DBMS——关系型DBMS（RDBMS）。所有与数据库相关的通信往来都将通过DBMS完成，为了做这件事，你可以使用SQL或其他类似的东西。数据库系统管理员（DBA）使用查询语言来建立并维护数据库，用户使用查询语言来访问数据库并查看或更改数据。

## 1.3 理解RDBMS

1970年，E. F. Codd创立了关系模式的概念。在RDBMS（例如DB2）产生之前，层次（IMS）和网状（IDMS）模式是常见的。在这些模式之前，使用平面文件（操作系统文件不一定平面）来建立数据库，并且使用第三代语言（3GL）访问例程。实际上，一些专用系统仍然是按这种方式建立的，只是进行了修改或根本没有变化。在大型机和微机中依然存在着许多这样的遗留数据库。CODASYL（数据系统语言协会）是数据库任务组（Database Task Group, DBTG）创建的一种数据库标准，这是一种基于COBOL的网络数据库标准，并且IDMS是一个厂商的实现。但是，从70年代起，RDBMS已经逐渐地控制了市场，如Oracle、Sybase、Informix和Ingres。

最近，面向对象（Object-Oriented, OO）的DBMS已经成为最为突出的数据库管理系统，并找到了许多适当的应用环境，如在CAD/CAM、工程、多媒体等等。面向对象DBMS适于在这些领域中应用，因为在一个几乎非事务性的环境中，它们具有控制复型数据类型实力。由于竞争，RDBMS厂商为了提供包括文本、音频、图像和视频数据类型的面向对象/多媒体性能，已经制造了商业可用的通用服务器。Oracle的Universal Server就是一个例子。另外，用户定义的数据类型或可扩展类型，已经被扩大或增加到核心数据库服务器中，Oracle 8就提供了这样的性能。类似这样的RDBMS产品被认为是混合的，然而它们明显比以前的RDBMS更具有主流性。

此外，多维数据库（Multi-Dimensional Database, MDD）也分享了部分市场份额，这些数据库为带有许多必须被多维存取或列表的变量（例如行为科学数据）的应用提供了高度索引化的数据。在传统的RDBMS中，这几乎是不可能实现的，数据库只允许单独使用。再者，为和MDD竞争，RDBMS供应商提供了一些他们自己的层次产品，这些产品提供超级索引化的数据，并使用了特殊的技术，例如位映射索引。Oracle的Express就是一个多维数据库的例子。

### 1.3.1 关系模型

你已经了解了DBMS的主要任务，为了进一步了解一个RDBMS是由什么构成的，你必须

先了解关系模型。下列情况出现在一个关系模型中：

数据的基础项是关系。

在这些表上的操作只产生关系（关系型闭合）。

什么是关系？这是一个描述两个集合的元素如何相互联系或如何一一对应的数学概念。因此，关系模型是建立在数学基础上的。然而，对你来说，关系只是一个带有一些特殊属性的表，一个关系模型把数据组织到表中，而且仅在表中。客户、数据库设计者、数据库系统管理员和用户都以同样的方式——即从表中——查看数据。那么，表就是关系模型的近义词。

一个关系型表有一组命名的属性（attribute）或列，以及一组元组（tuple）或行。有时列被称为域，行被称为记录，列和行的交集通常被叫做单元。列标示位置，有作用域或数据类型，例如字符或整数。行自己就是数据。表1-1有三列四行。

表1-1 汽车表

制 造	模型品牌	价 格
Toyota	Camry	\$25K
Honda	Accord	\$23K
Ford	Taurus	\$20K
Volkswagen	Passat	\$20K

一个关系表必须符合某些特定条件，才能成为关系模型的一部分：

贮存在单元中的数据必须是原子的。每个单元只能存贮一条数据，这也叫信息原则（Information Principle）。尽管在过去的数年中按某些违反这一条的方式已经建立了许多系统，但违反这一条将不能运用良好的设计原则。当一个单元包含多于一条的信息时，这叫做信息编码（information coding），一个很好的例子是一个车辆识别号码（Vehicle Identification Number, VIN）。如果它被存贮成一行，这将违犯信息原则，因为它包含了多条信息，例如产地、型号、出厂等等。在这样的情况下，是否采用违背理论的方案是一个设计的选择问题，尽管在多数情况下，结果证明这对数据的完整性是不利的。

贮存在列下的数据必须具有相同数据类型。

每行是唯一的（没有完全相同的行）。

列没有顺序。

行没有顺序。

列有一个唯一性的名称。

除了表和它们的属性，关系模型有它自己特殊的操作。不需要深入研究关系型数学，只需说明这些操作可能包括列的子集、行的子集、表的连接以及其他数学集合操作（如联合）等就足够了。真正要知道的事情是这些操作把表当作输入，而将产生的表作为输出。SQL是当前RDBMS的ANSI标准语言，它包含这些关系型操作。在SQL占主导地位之前，一种具有竞争性的语言是来自Ingres的QUEL或QUEry语言，另一种是UDL（统一数据语言，Unified Data Language）。ANSI（美国国家标准化组织）是一个具有广泛范围的标准实体，其中包括计算机软件语言（如SQL）的标准。

允许数据操作或数据处理的主要语句是SELECT、INSERT、UPDATE和DELETE。因此，这些数据处理操作中任何一个都是一个事务。

允许数据定义或结构化处理的基本语句是CREATE、ALTER和DROP。所有这些语句都可

以使用一组子句来替代, 这些子句可以利用多种变化来定义和存取组成数据库关系表的结构和数据。因此, SQL既是一种数据定义语言 (Data Definition Language, DDL), 也是一种数据操作语言 (Data Manipulation Language, DML)。统一的DDL和DML肯定比两种不同的语言和接口更有效、更有用。数据库系统管理员和用户可以通过完全相同的语言访问数据库。

关系模型要求的最后一件事是两个基础的完整性原则。它们是实体完整性原则 (entity integrity rule) 和引用完整性原则 (referential integrity rule)。首先, 让我们看看两个定义:

主键 (primary key) 是能唯一标识行的一列或一组列的集合。有时, 多个列或多组列可以被当作主键。

由多个列构成的主键被称为连接键 (concatenated key) 组合键 (compound key), 或者更常称为复合键 (composite key)。

数据库设计者决定哪些列的组合能够最准确和有效地反映业务情形, 这并不意味着其他数据未被存贮, 只是那一组列被选作主键而已。

剩余有可能被选为主键的列被叫做候选键 (candidate key) 或替代键 (alternate key)。一个外键 (foreign key) 是一个表中的一列或一组列, 它们在其他表中作为主键而存在。一个表中的外键被认为是对另外一个表中主键的引用。实体完整性原则简洁地表明主键不能全部或部分地空缺或为空, 引用完整性原则简洁地表明一个外键必须为空或者与它所引用的主键当前存在的值相一致。

一个RDBMS就是一个建立在前面这些关系模型基础上的, 一般能满足所提到的全部要求的DBMS。但是, 在70年代末到80年代初, RDBMS开始销售的时候, SQL超越了本质为非关系型的系统, 受到普遍欢迎, 并被称作关系型。这引发了一些修正活动, 即 Codd十二条法则 (1985)。

### 1.3.2 Codd十二条法则

DBMS应该遵循Codd提出的十二条法则, 才能被分类到完全关系型:

1) 信息法则。信息表现为贮存在单元中的数据, 正如前面所讨论过的, 将 VIN作为一个单独的列使用, 违反了这条规则。

2) 授权存取法则。每一个数据项必须通过一个“表名 + 行主键 + 列名”的组合形式访问。例如, 如果你能用数组或指针访问一个列, 就违反这条规则。

3) 必须以一致的方式使用空值。如果由于缺少数字值, 空值 (Null) 被当作 0 来处理, 或者由于缺少字符值而被当作一个空格处理, 那么它就违反了这条规则。空值仅仅是指缺少数据而且没有任何数值。如果缺少的数据需要值, 软件提供商通常提供使用缺省值的能力满足这一目的。

4) 一个活跃的、在线数据字典应作为关系型表被存储, 并且该字典应该可以通过常规的数据存取语言访问。如果数据字典的任何部分贮存在操作系统文件里, 就违反了这条规则。

5) 除了可能的低级存取例程外, 数据存取语言必须提供所有的存取方式, 并且是存取的仅有方式。如果你能通过一个实用程序而不是一个 SQL 接口来存取支持一个表的文件, 就有可能违反了本规则。参见规则 12。

6) 所有能被更新的视图应当是可更新的。例如, 如果你能将三个表连结起来, 作为一个视图的基础, 但却不能更新这个视图, 则违反本规则。

7) 必须有集合级的插入、更新和删除。目前,大多数 RDBMS 提供商都在某种程度上提供了这种能力。

8) 物理数据的独立性。应用不能依赖于物理结构,如果一个支持某表的文件从一张盘移动到其他盘上或重新命名,不应该对应用产生影响。

9) 逻辑数据的独立性。应用不应依赖于逻辑结构。如果一个表必须被分成两个部分,那么应该提供一个视图,以把两段连接在一起,以便不会对应用产生影响。

10) 完整性的独立性。完整性规则应该贮存在数据字典中。主键约束、外键约束、检查约束、触发器等等都应该贮存在数据字典中。

11) 分布独立性。一个数据库即使被分布,也应该能继续工作。这是规则 8 的一个扩展,一个数据库不仅能在一个系统(本地地)分布,也能在通过系统的网络(远程地)分布。

12) 非破坏性法则。如果允许低级存取,一定不能绕过安全性或完整性规则,这些规则是常规的数据存取语言所遵守的,例如,一个备份或载入工具不能绕过验证、约束和锁来备份或载入数据。然而,软件供应商出于速度的原因,通常提供这些功能。那么,数据库系统管理员就有责任确保数据的安全性和完整性,如果瞬间出现问题,应该立即恢复。例如当载入 VLDB 时,可以临时禁止并重新打开约束检查。

如果一个 DBMS 能满足本章中讨论的所有基本原则(两个定义、六个属性、关系型操作以及两个完整性规则)和这十二条法则,那么它就可以被当作一个 RDBMS。Codd 用他的法则 0 总结了这一切:“对于一个有资格成为 RDBMS 的系统来说,该系统必须排他地使用它的关系型工具来管理数据库。”