数据库技术系列

DATABASE PROFESSIONALS LIBRARY

SQL Server 2000 提高

北京希望电子出版社 总策划
吴 豪 编 著

红旗出版社

北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

数据库技术系列

DATABASE
PROFESSIONAL S
LIBRARY

SQL Server 2000 提高

北京希望电子出版社 总策划
吴 豪 编 著

红旗出版社

北京希望电子出版社
Beijing Hope Electronic Press
www.bhp.com.cn

图书在版编目（CIP）数据

SQL Server 2000 提高 / 吴豪编著.
—北京：红旗出版社，2005.4
ISBN 7-5051-1041-1
Ⅰ.S… Ⅱ.吴… Ⅲ.关系数据库—数据库管理系统，SQL Server 2000 Ⅳ.TP311.138
中国版本图书馆 CIP 数据核字（2004）第 102821 号

内 容 简 介

本书在《SQL Server 2000 基础》上，进一步介绍了 SQL Server 2000 数据库管理与开发的高级知识。

本书由 18 章和两个附录组成。主要内容包括范式化设计、SQL Server 存储与索引结构、事务与锁、高级查询、XML 简介、XML 集成、全文检索、英文查询、分布式查询与事务、数据转换服务、高级 DTS、大容量复制程序（bcp）、高级设计、分析服务、数据库安全性、性能测试、管理概览等内容。将各个知识点与具体实例结合，以印证知识使用的方法。另外，在附录部分提供了 SQL Server 2000 的系统和操作函数。

本书适合于 SQL Server 数据库管理员和高级开发者，重点面向开发者。结合《SQL Server 2000 基础》，对初学者的继续提高也有很大帮助。

本书部分素材请从 www.b-xr.com 下载。

需要本书或技术支持的读者，请与北京中关村 083 信箱（邮编：100080）发行部联系，电话：010-82702660，82702658，62978181（总机）转 103 或 238 传真：010-82702698 E-mail：tbd@bhp.com.cn。

系 列 名	数据库技术系列
书 名	SQL Server 2000 提高
编 著	吴 豪
责 任 编 辑	但明天 雷 铎
出 版 发 行	红旗出版社 北京希望电子出版社
地 址	红旗出版社 北京市沙滩北街 2 号（100727） 电话：（010）64037138 北京希望电子出版社 北京市海淀区上地三街 9 号金隅嘉华大厦 C 座 610
经 销	各地新华书店 软件连锁店
排 版	希望图书输出中心 王春桥
印 刷	北京媛明印刷厂
版次 / 印次	2005 年 4 月第 1 版 2005 年 4 月第 1 次印刷
开本 / 印张	787×1092 1/16 37.875 印张 890.7 千字
印 数	1-5000 册
书 号	ISBN 7-5051-1041-1
定 价	52.00 元

前　言

SQL Server 2000 在 SQL Server 6.5 和 SQL Server 7.0 的基础上有了飞跃的发展，增加了不少功能：XML 集成、分布式分区视图、索引视图、INSTEAD OF 触发器、用户自定义函数、两阶段数据抽取——用于数据转换服务（DTS）、数据挖掘等。作为目前应用广泛的一种数据库，它不仅采用了合理的体系结构，而且利用了优越的可视化开发环境，加上其优秀的编辑界面，使其很多性能又远远超过了 Oracle 及其他数据库，从而为广大用户所青睐。目前有不少应用软件利用了 SQL Server 2000 作为后台存储数据库，如部分医院信息管理系统，银行、证券管理系统等。为了使越来越多的用户逐步深入掌握 SQL Server 2000 这种优秀的数据库，笔者编写了数据库技术系列这套丛书，与大家分享。

笔者是一位具有多年数据库研发经验的资深系统分析员和软件工程师，用过 SQL Server 和 Oracle 等大型数据库，开发过医院信息管理系统和远程医疗视频会议等大型管理系统。在学习和研究这些数据库的过程中，笔者深深地感受到学习过程的艰辛，为把推敲再三的心得殷实详尽地记录下来，为后学者铺出一条广阔的道路，笔者与同事一起系统地编写了这套丛书。

这是一套 SQL Server 进阶的系列丛书，更是一组经验的结晶，可满足不同层次读者的切实需要。本书适合于 SQL Server 数据库设计领域的初学者和高级开发者，但重点面向开发者。这意味着，为了简明扼要，那些适合于数据库管理员而不适合于开发者的内容有时一带而过，甚至避而不谈，但对开发者有影响或在开发过程中需要考虑的管理因素，会在部分章节对相关内容进行简要阐述。对于初学者或具有一定关系数据库经验而不熟悉 SQL Server 的人来说，使用这套丛书可以迅速利用 SQL Server 进行数据库开发，设计出自己需要的数据库结构；对于具有一定基础的中级水平的开发人员，可以利用本书提供的内容，融会贯通、灵活运用，并利用本丛书提供的最新特点和使用方法，帮助自己提高开发和运用工具的水平；对于高级开发人员，这是一套不可多得的参考书，书中讨论的内容表明：本版软件包含了更多内容，SQL Server 将以一个全新的面孔展现出精彩的世界。

本丛书目的在于使读者阅读以后能在前进的道路上掌握更高级的技术，假设你已经是一个有经验的开发者（并不一定在数据库方面）。为了彻底理解本书，虽然需要懂得编程的基础知识，如变量、数据类型、程序化设计等，但并不要求以前就看过有关查询的内容。

从头到尾通读全书，就可以很好地掌握本书介绍的相关内容。本书侧重于帮助读者利用 SQL Server 2000 成功地开发应用程序，并为备战微软认证考试的人提供帮助，因此笔者会简要介绍一些力求更完美的内容。

这套书不是万能工具书，但它能够提供有关 SQL Server 2000 开发的许多信息，如 DTS、英文查询和联机分析处理（OLAP）等。

内容结构

本套丛书分为《SQL Server 2000 基础》和《SQL Server 2000 提高》两本（以下简称《基础》和《提高》）。

为了使读者迅速利用 SQL Server 进行数据库开发，设计出自己需要的数据库结构，笔

者从数据库设计的角度出发，先介绍如何使用 SQL Server 2000 必备的工具，然后讨论如何进行数据库设计和开发应用，最后就高级数据库设计中可能遇到的各种问题做详细剖析。在《基础》中，笔者介绍了 SQL Server 2000 的历史背景、数据对象、常用工具、Transact-SQL、联接、创建和修改表、约束、视图、脚本和批处理、存储过程、用户自定义函数、触发器及游标等内容。在《提高》中，笔者先讨论了如何进行范式化设计，并就数据库设计中不得不讨论的索引结构、事务和锁展开探讨，再结合目前广为应用的 XML 技术，探讨了如何将 SQL Server 2000 与 XML 结合起来，并就如何利用 XML 技术进行高级查询、全文检索、英文查询做了深入介绍，同时就如何设计合理的分布式分区视图展开描述，最后就数据转换服务和数据库之间的数据复制等内容做了详细讨论。

运行环境

本书讨论的所有内容几乎都有例子，而且列出了相应的代码。为了能够更好地运行本书提供的示例，需要配备：

- SQL Server 2000
- Windows NT 4.0 和 SP5 补丁，最好使用 Windows 2000（Windows 9X 也可以工作，但在某些方面会失败）

约　定

本书采用许多不同文本方式和显示方式，用于帮助区分不同内容。其中：

- 对于需要特别注意和需要提示的内容，相应的文字使用楷体，并在段落前标以“注意：”或方框字样。例如：

注意： Personal 版本虽然能与其他工具协调使用，但建议不要这样使用，因为微软不支持这种用法。

- 新代码或重要代码的格式设为：

```
SELECT CustomerID,ContactName,Phone
FROM Customers
```

- 以前见过的代码或者无须讨论的代码格式为：

```
SELECT ProductName FROM Products
```

- 背景信息、附加信息和参考内容的字体格式如下：

20 世纪 80 年代，美国国家标准化组织（American National Standards Institute，ANSI）制订了 SQL 语言规范，ANSI-SQL 诞生了，这是 RDBMS 计算领域的关键时刻。

由于本书篇幅有限，加之时间仓促，难免有疏漏、不足之处，诚请广大读者提出宝贵意见并批评、指正。

目 录

第 1 章　范式化设计

如果本书是你第一次真正涉足数据库，那么对迄今为止怎样和为什么要构造表可能仍然感到困惑。

本书讨论的内容基本都在 OLTP（Online Transaction Processing，联机事务处理）环境下进行，书中将讨论 OLTP 和其更具面向分析的 OLAP（Online Analytical Processing，在线分析处理）之间的差别。需要指出的是，在大部分例子中，会看到最常见的 OLTP 数据库表的优化设计，这种方式的数据库设计问题属于第三范式下数据库布局的范畴。

那么，什么是“范式”呢？简单地说，范式就是将数据分成逻辑、非重复的格式，并且容易组合成一个整体。除了范式化（它将数据变成标准格式）之外，我们还要验证 OLTP 和 OLAP 数据库的功能。此外，还要在本章介绍实现“约束”的许多例子。

本章内容对部分读者而言，可能是最难掌握的章节之一。建议仔细研读本章，充分掌握其中的细节，从而为今后设计自己的数据库打下坚实的基础。

1.1　表

在《SQL Server 2000 基础》（以下简称《基础》）中，已经对表进行了大量的阐述，这些内容似乎超出了基础知识的范围，但还是简要复习一下什么是表。

数据库表不是无关紧要的，而是数据库的核心对象。表是具有相同通用属性（attributes）的实例数据集合，这些数据实例组织为数据行（row）和列（column）。一张表应该提供实际的数据集合，而且与其他表有关联（relationships）信息。各种实体（表）和关系（它们怎样协同工作）的描述通常指一个实体关系图表（或者叫 ER 图），有时将 ER 图写为 ERD（Entity- Relationship diagram）。

通过各种关系连接两张或多张表，可以根据需要在这些表的组合数据中临时创建其他表（在《基础》的第 4 和第 5 章中已经对此进行过深入讨论）。关系实体集以成组的方式存放在数据库中。

1.2　实现数据范式化

范式是现代 OLTP 数据库设计的基础模型，它首先沿用了关系数据库的概念。范式和关系数据库的概念源自 E.F.Codd（IBM 公司）1969 年的专著。当时，Codd 首先提出了数据库“由一系列可以返回表的非过程化操纵的无序表组成”的概念，这种概念在当时并没有引起计算机领域的研究人员的重视。在随后的几十年里，经过像 Sybase 和 Oracle 等公司的研究和发展，以及 ANSI 的不懈努力，最终形成了真正的关系数据库产品，从而使关系数据库最终变成当今占主流地位的数据库产品。

实现数据范式化的关键在于：

- 次序不是很重要。
- 一张基表应能以非过程化方法与其他表相关（Codd 称表为关系）。
- 通过关联这些基表，应该能创建一个满足需要的虚拟表。

范式化是关系数据库设计的一个分支。

在编程过程中，范式化的概念已经成为被滥用的甚至是被误解的概念之一。人人都认为自己理解了这一概念，而且许多人至少在学术中做到了这一点。不幸的是，不少数据库设计者对此像走过场——在某种程度上，他们像“真正”的数据库设计师。实际上，他们也仅知道什么是范式。范式其实就是大型数据库设计的一张图纸，有时需要实现数据的范式，但有时需要再次故意使数据不满足范式的要求。即使在范式化处理过程中，通常也可有多种途径得到技术上的范式化数据库。

范式化是一种理论，而且仅是一种理论。一旦选择了是否采用范式化策略，就决定了你的数据库拥有的内容——满怀希望地期待它可能是你设计出的最好数据库。不要沉迷于书本知识，最好根据实际情况进行处理。这里介绍的内容只是告诉你相关的概念。为了得到最佳的解决方案，你需要掌握这些概念，但并不要求你按照笔者的想法去设计自己的数据库。下面开始讨论范式的形式和作用。

我们首先从 6 种范式形式说起。对那些已经接触过数据库和范式化设计的人来说，或许你对有 6 种范式感到惊奇。你可能听到完整的范式化设计通常归结为 3 种形式——难道不止这 3 种吗？本书重点讨论常用的 3 种形式，但为了后续的需要，还将对其他 3 种形式做简要介绍。

我们已经看到了怎样创建主键，并看到了在表中使用主键的原因，这就是：如果你只想返回一行，就需要能惟一定义该行。范式化的概念在很大程度上是围绕行的主键定义和主键所依赖的列展开的。在范式化设计中，最常见得说法是：

键，乃至整个键，什么也不是，就是一个键。

这是对第三种范式的最简短解释。如果有人说所有的列仅依赖于整个键（不多也不少），那么他所说的就是第三范式。

下面来看看各种标准范式，了解每种范式是怎样工作的。

1.2.1 准备工作

在设法把数据转换成第一范式化格式之前，还需要进行适当的准备。用关系数据库意义上的词来说，在将表考虑为实体之前，需要考虑以下几件事情：

- 表应该描述一个实体。
- 所有行必须惟一，而且必需有一个主键。
- 列和行的顺序不受影响。

这些内容有些非常明显，而有些则不明显，我们将在范式化设计过程中详细讨论这些内容。首先要定义实体，然后仔细检查并定义所有明显的实体。

面向对象的编程方法（OOP）通常会在对象模型中把最符合逻辑的上层实体比作对象。

下面考虑一个非常简单的模型——再次考虑库房管理模型。开始，我们不考虑各种变化，甚至不考虑拥有的列。相反，我们只考虑系统中将要定义的基本实体。

首先，考虑最基本的处理。在处理过程中，想为每一个原子体创建一个实体，从而能够维护其中的数据。

假设第一个要传递的实体是入库单。

如果 NI 是一位具有丰富范式化设计经验的系统设计师，那么你的第一印象可能是从开始就产生一些实体。但现在只采用这个实体，看看范式化处理进程如何给我们提供所需要的结果。

假定 NI 已经掌握了有关实体的概念，下一步要指出哪个列产生主键。记住，主键为每一行提供了惟一标识。

仔细看看列的清单并提供键的候选人（key candidates）。候选键的清单应该包括惟一标识实体中每一行的所有列。另外，规则可以指出哪一列能成为主键（这就是许多人在设计数据库时包含了相同基本信息的原因之一）。如果找不到候选键，就要采用组合的方法来形成一个候选键。表 1-1 是我们创建一张入库记录 IMaster。

表 1-1　入库记录表 IMaster

中文字段名	英文字段
入库编号	INo
入库日期	IDate
厂家编号	ISource
厂家名称	FacName
联系人	Connect
联系电话	Telephone
联系地址	Address
入库方式	IMode
入库设备	IstrumItem

因为这是一张入库表，一个入库编号就意味着一张入库单，所以要用 INo 作为主键。

现在已经拥有了基本实体，即入库记录表 IMaster，这个实体并不关心列的顺序（根据传统，表的组成通常将第一列作为主键，但从技术角度来讲，并不需要这样做）。在表的基本组成中，没有哪一列关心行的顺序。表面上，表描述的对象仅是一个实体。下面准备开始范式化进程（实际上，我们已经开始了）。

1.2.2 第一范式

第一范式（The first normal form，简称 1NF）完全是定义消除数据重复并保证数据的原子性（atomicity，该数据是自包含而且是独立的）。它在高层创建主键，把重复数据移到新表中，再为这些表创建新键，还把数据分成由列组成的单行数据。

在更传统的平面设计中，经常会看见很多重复数据（因为多种信息都存放在一列中），这会带来很多问题：

- 磁盘存储非常昂贵，多次存储数据意味着浪费空间。目前数据存储的成本已经不再昂贵，所以这已经不是一个问题。
- 重复数据意味着要移动更多的数据，从而导致大的 I/O 流量。这意味着性能受阻，因为后台的大量数据块必须通过数据总线和网络移动，即使当今的数据传输技术发展速度非常快，在性能上也会有负面影响。
- 应该成为重复数据行之间的数据通常不一致，并产生某种自相矛盾或缺乏完整性的数据。
- 如果想从组合数据列中查找信息列，首先就得采用一种方式解析该列的数据（这种工作相当慢）。
- 通过前一节的介绍，我们知道该表中已经拥有很多列，你还可以再加入一些列。如果想了解入库单的所有内容，最好是从一个地方查询出所有结果。

下面探究一下这到底意味着什么，让我们看看表中的一些数据（这里有意删减了一些列，表 1-2）。

表 1-2 入库砍数据表

INo	IDate	ISource	FacName	……	IMode	IstrumItem
100	2004/01/06	0001	Linkstar	……	购买入库	卫星调制解调器 150 个，单价 50000.00 元 串口服务器 150 个，单价 3000.00 元 视频通信终端 200 套，单价 60000.00 元
101	2004/01/06	0002	创佳公司	……	购买入库	LNB（低噪声放大器）200 个，单价 3000.00 元，保质期 2 年 BUC（功率放大器）200 个，单价 20000.00 元，保质期 2 年
102	2004/02/13	0001	Linkstar	……	赠送入库	① SUN 工作站一台，估价 10 万元
103	2004/01/14	0004	新通公司	……	购买入库	① 网线 2 箱（600 米/箱），总价 1000.00 元

如果继续对该表实现范式化，还需要处理一系列问题。虽然我们已经拥有了一个功能主键（主键已经存在于关系系统中很久了），但在第一范式中仍然要进行适当的处理。

- 存在数据重复组（入库来源），需要将该数据分成不同的表。
- 入库设备（IstrumItem）列不包含本质上是原子的数据。

我们可以从该表中分离出几列，以形成新表 FacName 中（表 1-3 和表 1-4）。

表 1-3　分离入库单

ISource	FacName	……	IMode	IstrumItem
0001	Linkstar	……	购买入库	卫星调制解调器 150 个，单价 50000.00 元 串口服务器 150 个，单价 3000.00 元 视频通信终端 200 套，单价 60000.00 元
0002	创佳公司	……	购买入库	LNB（低噪声放大器）200 个，单价 3000.00 元，保质期 2 年 BUC（功率放大器）200 个，单价 20000.00 元，保质期 2 年
0001	Linkstar	……	赠送入库	① SUN 工作站一台，估价 10 万元
0004	新通公司	……	购买入库	① 网线 2 箱（600 米/箱），总价 1000.00 元

表 1-4　表 FacName

ISource	FacName	Connect	Telephone	Address
0001	Linkstar	卓娅	13006883113	北京市复兴路 33 号
0002	创佳公司	黄忠	010-82316945	北京市海淀路 24 号
0004	新通公司	张剑锋	020-43561234	上海市华山路 30 号

在新表和旧表之间，需要注意：

- 在新表中必需有一个主键以确保每一行数据惟一。新表有两个候选键——厂家编号（ISource）和厂家名称（FacName）。厂家编号就是为了这种目的而创建的，它似乎仅是一个逻辑选择。
- 尽管从 IMaster 表中移出了有关厂家信息的数据，但仍然需要在新 IMaster 表中引用 FacName 表中的数据，这就是为什么在入库主记录表中仍然需要入库来源（主键）列的原因。建立引用时，将创建一个外部键约束，强迫所有入库单都拥有一个有效厂家编号。
- 对 Linkstar 有限公司而言，可以省略实例信息，这就是移动重复组中的数据的目的——只存储一次；这样既节省了空间又避免了冲突值。
- 只移动重复组数据。我们仍然可以看到多条记录的入库单日期相同，但它实际不在一个组中——它仅是该表中没有适当边界的相关随机数据。

前面已经处理了重复数据，下面准备讨论第一范式的第二种非常规行为。如果看一看 IstrumItem 列，就会看到实际有许多存在差别的数据，比如设备名称、设备规格和单价信息。

如果维持设备名称、规格和单价信息不变，那么它们就是原子信息，一旦把它们组合在一起，就不再是原子体了。

无论相信与否，有时事情就是以这种方式处理的。乍一看，似乎这是一件非常简单的事情。

下面继续分解——而且，要根据单价添加一些新信息。问题是，一旦将这些信息分解，主键就不再能惟一定义行信息了——行仍然惟一，但主键不再惟一。

现在，我们在表中添加流水号 IItemNo 作为主键的一部分，从而可以再次惟一定义行。因为每张入库单只要具备了入库单号和入库流水号，就能准确定位是哪一条记录。

注意： 也可以不创建另一列，而将产品编号作为主键的一部分，但这样最终会导致在同一张订单中不能让同一种产品出现两次。我们将在第 2 章中简要讨论基于多列上的键——组合键。

此时已经满足了第一范式的标准：没有数据重复组，而且所有列都是原子体。但是还有一些列不得不存在重复数据的问题，随后将会对此进行讨论。

1.2.3 第二范式

范式化的下一个阶段是第二范式（2NF），第二范式进一步减少了重复数据的范围（不一定分组）。

第二范式有两个规则：

- 表必须满足第一范式的规则（范式化是一个建筑块的处理过程——如果没有准备好前两种范式，就不可能拥有第三范式）。
- 每个列必须依赖整个键。

该例子存在一个问题——实际上，在该区域中有一对规则。再看一看第一范式下的 IMaster 表，它的每一列都依赖整个键吗？难道真需要这部分键吗？

答案是既是也不是。有几列仅依赖于 INo 列，而不是 IItemNo 列。它们是：Idate，Isource，Imode，IAuthor，Ismanager 和 IHandler。无论该入库单上有多少条记录，这几列对整个入库单而言都一样。处理这些问题需要创建另一张表，这要涉及头表（header）与明细表（detail）的概念。

实际上，一个实体有时仍然需要分成两个实体（两张表）。头表在关系上是明细表的父表，它包含的信息只需要保存一次，而明细表存储的信息则可能存放于多个实体中。头表通常保存了原始表名，而明细表则以头表开头并添加了一些明细信息（如 IDetail）。头表的每一条记录通常至少有一条明细记录，也可能有多条明细记录，这是下一节要了解的一种对应关系（一对多的对应关系）。

为了讨论相关内容，我们需要再次对该表进行分离。首先考虑明细表，将它称为入库单明细表（IDetail）：Ino（PK），IItemNo（PK），Item_code，Item_spec，units，batch_no，expire_date，firm_id，purchase_price，discount，retail_price，package_spec，quantity，package_units，sub_package_1，sub_package_units_1，sub_package_spec_1，sub_package_2，sub_package_units_2，sub_package_spec_2，invoice_no，invoice_date。

尽管可以把它当做两张表之间的新表，但还需要头表 Imaster，让它保存入库单名：INo（PK），Storage，import_date，supplier，account_receivable，account_payed，additional_fee，import_class，sub_storage，account_indicator，memos，operator。

这就是第二范式的例子，所有列都依赖于整个键，但是范式化设计过程还远未结束。