

# SQL Server 2008数据库 应用与开发教程 (第二版)

- ◆ 数据库建立与维护
- ◆ 表的建立与维护
- ◆ 表数据操作
- ◆ 安全与权限
- ◆ T-SQL语言基础
- ◆ 查询和视图
- ◆ 索引
- ◆ 数据库完整性
- ◆ 存储过程、触发器
- ◆ 数据备份、恢复和报表
- ◆ SQL高级应用
- ◆ 实验



卫 琳 主编  
李 妍 周飞菲 王军委 副主编

清华大学出版社



高等学校计算机应用规划教材

# SQL Server 2008 数据库应用 与开发教程(第二版)

卫琳 主 编

李妍 周飞菲 王军委 副主编

docin 豆丁  
www.docin.com

清华大学出版社

北 京

## 内 容 简 介

本书全面介绍了 SQL Server 2008 系统的体系架构和功能。全书共 13 章, 内容包括 SQL Server 2008 基础知识、T-SQL 语言、数据库和表、SELECT 查询、视图、索引和游标、存储过程与触发器、数据库的备份与还原、安全与权限等。读者可以通过实例了解如何利用 SQL Server 2008 开发实际的应用系统。此外, 本书还配有习题和上机实验, 有助于读者对所学知识的理解与掌握。

本书结构清晰, 实例丰富, 图文并茂, 浅显易懂, 既可作为大学本科、高职高专院校的数据库应用课程教材, 也可作为初学者学习数据库的参考书以及数据库应用系统开发人员的技术参考书。

本书对应的电子教案、习题答案和实例源代码可以从 <http://www.tupwk.com.cn/downpage/index.asp> 网站下载。

本书封面贴有清华大学出版社防伪标签, 无标签者不得销售。

版权所有, 侵权必究。侵权举报电话: 010-62782989 13701121933

### 图书在版编目(CIP)数据

SQL Server 2008 数据库应用与开发教程(第二版)/卫琳 主编. —北京: 清华大学出版社, 2011.6

(高等学校计算机应用规划教材)

ISBN 978-7-302-24453-0

I. S… II. 卫… III. 关系数据库—数据库管理系统, SQL Server 2008—高等学校—教材 IV. TP311.138

中国版本图书馆 CIP 数据核字(2011)第 262480 号

责任编辑: 胡辰浩(huchenhao@263.net) 袁建华

装帧设计: 孔祥丰

责任校对: 蔡 娟

责任印制:

出版发行: 清华大学出版社

地 址: 北京清华大学学研大厦 A 座

<http://www.tup.com.cn>

邮 编: 100084

社 总 机: 010-62770175

邮 购: 010-62786544

投稿与读者服务: 010-62776969, c-service@tup.tsinghua.edu.cn

质 量 反 馈: 010-62772015, zhiliang@tup.tsinghua.edu.cn

印 刷 者:

装 订 者:

经 销: 全国新华书店

开 本: 185×260 印 张: 21 字 数: 521 千字

版 次: 2011 年 6 月第 1 版 印 次: 2011 年 6 月第 1 次印刷

印 数: 1~5000

定 价: 35.00 元

---

产品编号:



[www.docin.com](http://www.docin.com)



# 前 言

Microsoft 公司推出的 SQL Server 2008 数据库管理系统是大型关系数据库系统中的佼佼者,它基于成熟而强大的关系模型,能够满足各种类型的企事业单位构建网络数据库的需求,具有操作简单、功能齐全、安全可靠等特点,是目前各类院校的学生学习大型数据库管理系统的首选数据库产品。

本书共 13 章,第 1 章介绍了 SQL Server 2008 的基础知识;第 2 章介绍了利用管理工具建立与维护数据库的方法;第 3 章介绍了利用管理工具建立与维护数据表的方法;第 4 章介绍了表数据操作语句;第 5 章介绍了安全与权限;第 6 章介绍了 T-SQL 语言基础;第 7 章介绍了查询和视图;第 8 章介绍了索引;第 9 章介绍了数据完整性;第 10 章介绍了存储过程和触发器;第 11 章介绍了数据备份、恢复和报表;第 12 章介绍了 SQL 高级应用;第 13 章是实验。本书注重循序渐进、由浅入深、举一反三、理论联系实际。书中的文字说明、图形显示、程序语句(所有程序都运行通过)、习题和上机实验比例恰当,且配有电子教案和实例源代码,既便于教师教学,又方便学生学习,具有很强的实用性。

本书可作为大学本科、高职高专院校的数据库应用课程的教材,也可作为初学者学习数据库的参考书以及数据库应用系统开发人员的技术参考书。

本书由卫琳主编,李妍、周飞菲、王军委为副主编。其中第 1、8、9、12、13 章由卫琳执笔,第 2、6、10、11 章由李妍与王军委执笔,第 3、4、5、7 章由周飞菲执笔,全书由卫琳总纂。在本书编写过程中,石育澄、李斌、沈定军等参与了部分资料的收集整理工作,同时,张聪、陈文远、姚培娟、屠卫、王晖、薛正元、郭东东等付出了努力,在此一并向他们表示感谢。

在本书出版过程中,清华大学出版社的老师对书稿提出了许多宝贵意见。同时,在编写本书的过程中参考了相关文献,在此向这些文献的作者深表感谢。由于时间较紧,书中难免有错误与不足之处,恳请专家和广大读者批评指正。我们的联系方式为信箱:huchehao@263.net,电话 010-62796045。

作 者

2011 年 5 月

# 目 录

第 1 章 SQL Server 2008 简介..... 1	
1.1 SQL Server 2008 概述..... 1	
1.1.1 SQL Server 2008 的体系结构..... 1	
1.1.2 数据库和数据库对象..... 2	
1.1.3 SQL Server 2008 的新增特点..... 5	
1.2 SQL Server 2008 的安装..... 6	
1.2.1 系统版本..... 7	
1.2.2 SQL Server 2008 的运行环境要求..... 8	
1.2.3 Microsoft SQL Server 2008 的安装..... 10	
1.3 SQL Server 2008 的配置..... 20	
1.3.1 注册服务器..... 21	
1.3.2 配置服务器选项..... 23	
1.4 SQL Server 2008 常用的管理工具..... 29	
1.4.1 SQL Server Management Studio..... 29	
1.4.2 Business Intelligence Development Studio..... 30	
1.4.3 SQL Server Profiler..... 31	
1.4.4 SQL Server Configuration Manager..... 32	
1.4.5 Database Engine Tuning Advisor..... 33	
1.4.6 实用工具..... 34	
1.5 习题..... 35	
第 2 章 数据库的建立与维护..... 37	
2.1 数据库的组成..... 37	
2.1.1 系统数据库..... 38	
2.1.2 用户数据库..... 38	
2.1.3 示例数据库..... 39	
2.1.4 数据库文件及其文件组..... 39	
2.1.5 数据库对象..... 40	
2.2 数据库的创建..... 41	
2.2.1 使用 SSMS 图形界面创建数据库..... 42	
2.2.2 使用 Transact-SQL 语句创建数据库..... 45	
2.2.3 数据库的查看和修改..... 47	
2.2.4 删除数据库..... 51	
2.2.5 数据库操作..... 53	
2.2.6 复制和移动数据库..... 54	
2.3 习题..... 56	
第 3 章 表的建立与维护..... 57	
3.1 表概述..... 57	
3.1.1 什么是表..... 57	
3.1.2 表的数据类型..... 60	
3.1.3 创建表..... 60	
3.2 列数据类型..... 64	
3.2.1 数据类型的分类..... 64	
3.2.2 数字数据类型..... 64	
3.2.3 字符数据类型..... 66	
3.2.4 日期时间数据类型..... 66	
3.2.5 二进制数据类型..... 68	
3.2.6 其他数据类型..... 68	
3.2.7 数据类型的选择..... 68	
3.3 列的其他属性..... 69	
3.3.1 默认约束..... 69	
3.3.2 空值约束..... 69	

3.3.3 IDENTITY 的应用.....	70	5.2 管理用户.....	96
3.4 向表中添加数据.....	70	5.2.1 管理对 SQL Server 实例的 访问.....	96
3.5 查看表.....	72	5.2.2 管理对 SQL Server 数据库 的访问.....	101
3.5.1 查看表中的有关信息.....	72	5.3 管理角色.....	103
3.5.2 查看表中存储的数据.....	72	5.3.1 管理实例角色.....	103
3.5.3 查看表与其他数据对象的 依赖关系.....	73	5.3.2 管理数据库角色.....	104
3.6 修改表.....	73	5.3.3 管理应用程序角色.....	106
3.6.1 使用 SSMS 图形化界面 修改表.....	73	5.4 管理架构.....	107
3.6.2 使用 T-SQL 语句修改表.....	73	5.4.1 认识架构.....	107
3.6.3 使用 INSERT SELECT 语句.....	74	5.4.2 使用默认架构.....	108
3.6.4 使用 SELECT INTO 语句.....	75	5.5 管理权限.....	109
3.7 删除表.....	75	5.5.1 管理对 SQL Server 实例和 数据库的访问.....	109
3.8 习题.....	77	5.5.2 管理对表和列的访问.....	110
<b>第 4 章 表数据操作.....</b>	<b>79</b>	5.5.3 管理对可编程对象的 访问.....	112
4.1 概述.....	79	5.6 习题.....	116
4.2 界面操作表数据.....	80	<b>第 6 章 T-SQL 语言基础.....</b>	<b>117</b>
4.2.1 插入记录.....	81	6.1 T-SQL 语言概述.....	117
4.2.2 删除记录.....	81	6.1.1 T-SQL 语言的发展过程和 特点.....	118
4.2.3 修改记录.....	82	6.1.2 T-SQL 语言的分类.....	119
4.3 命令操作表数据.....	82	6.1.3 T-SQL 语法规则.....	119
4.3.1 使用 INSERT 语句插入表 数据.....	82	6.2 附加的语言元素.....	121
4.3.2 使用 DELETE 或 TRUNCATE 语句删除 数据.....	86	6.2.1 标识符.....	121
4.3.3 使用 UPDATE 语句修改 数据.....	89	6.2.2 常量.....	122
4.4 习题.....	92	6.2.3 变量.....	123
<b>第 5 章 安全与权限.....</b>	<b>93</b>	6.3 运算符和表达式.....	124
5.1 安全与权限的基础知识.....	93	6.3.1 算术运算符.....	125
5.1.1 SQL Server 2008 安全机制的 总体策略.....	93	6.3.2 关系运算符.....	125
5.1.2 网络安全基础.....	94	6.3.3 逻辑运算符.....	126
		6.3.4 字符运算符.....	127
		6.3.5 位运算符.....	127
		6.3.6 运算符的优先顺序.....	128

6.3.7 表达式 .....	128	7.3 视图 .....	165
6.4 流程控制语句 .....	130	7.3.1 视图概述 .....	165
6.4.1 BEGIN...END 语句 .....	130	7.3.2 视图的创建 .....	166
6.4.2 IF...ELSE 语句 .....	130	7.3.3 视图的修改、查看和 重命名 .....	168
6.4.3 CASE 语句 .....	131	7.3.4 视图的删除 .....	171
6.4.4 GOTO 语句 .....	132	7.3.5 通过视图更改记录 .....	171
6.4.5 WHILE、BREAK 和 CONTINUE 语句 .....	132	7.4 习题 .....	172
6.4.6 WAITFOR 语句 .....	133	<b>第 8 章 索引</b> .....	<b>175</b>
6.4.7 RETURN 语句 .....	134	8.1 索引概述 .....	175
6.5 函数 .....	134	8.1.1 创建索引的原因 .....	175
6.5.1 数学函数 .....	134	8.1.2 创建索引应该考虑的 问题 .....	175
6.5.2 字符串函数 .....	135	8.2 索引的分类及特点 .....	176
6.5.3 日期和时间函数 .....	136	8.2.1 B-Tree 索引结构 .....	176
6.5.4 聚合函数 .....	138	8.2.2 聚集索引和非聚集索引 .....	176
6.5.5 用户自定义函数 .....	139	8.2.3 其他类型索引 .....	177
6.6 习题 .....	144	8.2.4 访问数据的方式 .....	177
<b>第 7 章 查询和视图</b> .....	<b>145</b>	8.2.5 索引的创建 .....	178
7.1 连接、选择和投影 .....	145	8.2.6 索引的维护和删除 .....	181
7.1.1 选择(Selection) .....	145	8.3 习题 .....	184
7.1.2 投影(Projection) .....	146	<b>第 9 章 数据完整性</b> .....	<b>185</b>
7.1.3 连接(JOIN) .....	147	9.1 数据完整性概述 .....	185
7.2 数据查询语句(SELECT 语句) .....	147	9.2 使用约束实施数据的 完整性 .....	186
7.2.1 SELECT 语句对列的 查询 .....	148	9.3 约束的概念和类型 .....	187
7.2.2 SELECT 语句中的条件 查询 .....	152	9.4 管理约束 .....	187
7.2.3 ORDER BY 子句的 使用 .....	155	9.4.1 PRIMARY KEY 约束 .....	187
7.2.4 GROUP BY 子句的 使用 .....	156	9.4.2 UNIQUE 约束 .....	190
7.2.5 表的连接(JOIN) .....	157	9.4.3 CHECK 约束 .....	192
7.2.6 子查询 .....	161	9.4.4 DEFAULT 约束 .....	193
7.2.7 SELECT 语句的其他 子句 .....	164	9.4.5 NULL 约束 .....	195
		9.4.6 FOREIGN KEY 约束 .....	196
		9.5 禁止约束 .....	199
		9.5.1 创建规则 .....	199

9.5.2 绑定规则 .....	200	11.3 自动化管理任务 .....	235
9.5.3 解除绑定 .....	200	11.3.1 多服务器管理 .....	235
9.5.4 删除规则 .....	201	11.3.2 SQL Server 2008 代理 服务配置 .....	236
9.6 默认 .....	201	11.4 分离和附加数据库 .....	239
9.6.1 创建默认 .....	201	11.4.1 分离和附加数据库的 操作 .....	240
9.6.2 绑定默认 .....	201	11.4.2 使用 T-SQL 进行分离和 附加数据库操作 .....	243
9.6.3 解除绑定 .....	201	11.5 报表服务 .....	245
9.6.4 删除默认 .....	202	11.6 习题 .....	250
9.6.5 使用 SSMS 图形化界面 管理默认值对象 .....	202		
9.7 使用自动编号 IDENTITY .....	202	<b>第 12 章 SQL 高级应用 .....</b>	<b>251</b>
9.8 习题 .....	203	12.1 事务(Transaction) .....	251
<b>第 10 章 存储过程和触发器 .....</b>	<b>205</b>	12.1.1 事务的由来 .....	251
10.1 存储过程 .....	205	12.1.2 事务的概念 .....	251
10.1.1 存储过程的基本概念 .....	205	12.1.3 事务的特性 .....	252
10.1.2 存储过程的类型 .....	206	12.1.4 事务的工作原理 .....	252
10.1.3 用户存储过程的创建与 执行 .....	207	12.1.5 事务的执行模式 .....	252
10.1.4 存储过程的查看、修改和 删除 .....	209	12.1.6 使用事务时的考虑 .....	254
10.2 触发器 .....	212	12.2 锁(Lock) .....	254
10.2.1 触发器概述 .....	212	12.2.1 事务的缺陷 .....	254
10.2.2 DML 触发器的创建和 应用 .....	213	12.2.2 锁的概念 .....	256
10.2.3 DDL 触发器的创建和 应用 .....	215	12.2.3 隔离性的级别 .....	256
10.2.4 查看、修改和删除 触发器 .....	217	12.2.4 锁的空间管理及粒度 .....	257
10.3 习题 .....	218	12.2.5 锁的类别 .....	258
<b>第 11 章 数据备份、恢复和报表 .....</b>	<b>219</b>	12.2.6 如何在 SQL Server 中 查看数据库中的锁 .....	259
11.1 数据库的导入导出 .....	219	12.2.7 死锁及其防止 .....	260
11.1.1 数据库的导出 .....	219	12.3 游标 .....	261
11.1.2 数据库的导入 .....	223	12.3.1 游标(Cursor)概述 .....	261
11.2 数据库的备份与还原 .....	226	12.3.2 声明游标 .....	262
11.2.1 数据库的备份 .....	226	12.3.3 打开游标 .....	263
11.2.2 数据库的还原 .....	232	12.3.4 读取游标 .....	265
		12.3.5 关闭游标 .....	266
		12.3.6 删除游标 .....	266
		12.4 创建分区 .....	267

12.4.1 分区概述 .....	267	第二单元 数据库的建立与 维护 .....	284
12.4.2 分区技术的分类 .....	267	第三单元 表的建立与维护 .....	286
12.4.3 创建分区函数 .....	268	第四单元 表数据操作 .....	287
12.4.4 创建分区方案 .....	269	第五单元 安全与权限 .....	288
12.4.5 创建分区表 .....	269	第六单元 T-SQL 语言基础 .....	306
12.4.6 管理分区 .....	270	第七单元 查询和视图 .....	307
12.4.7 使用向导创建分区表 .....	271	第八单元 索引 .....	310
12.5 SQL Server 服务体系 .....	273	第九单元 数据完整性 .....	311
12.5.1 集成服务 .....	274	第十单元 存储过程和触发器 .....	317
12.5.2 分析服务 .....	277	第十一单元 数据备份、恢复和 报表 .....	319
12.6 习题 .....	280	第十二单元 SQL 高级应用 .....	320
第 13 章 SQL Server 实验指导 .....	281	参考文献 .....	323
第一单元 SQL Server 2008 简介 .....	281		

# 第1章 SQL Server 2008简介

微软公司发布的 Microsoft SQL Server 2008 是一个典型的关系型数据库管理系统。以其功能强大、操作简便、安全可靠等特性，得到了广大用户的认可，应用也越来越广泛。SQL Server 2008 是一个重大的新产品版本，具有许多新特性和关键的改进，是至今为止最强大和最全面的 SQL Server 版本。该产品不仅可以有效地执行大规模联机事务，而且可以完成数据仓库电子商务应用等许多具有挑战性的工作。SQL Server 2008 不仅继承了微软产品的一贯特点，而且在性能、可靠性、可用性、可编程性、易用性等方面都远远胜过了 SQL Server 2005。本章主要介绍 SQL Server 2008 的特点、安装与配置，以及常用管理工具的功能和使用。

**本章的学习目标：**

- 了解 Microsoft SQL Server 2008 的重要特性与特点
- 了解 Microsoft SQL Server 2008 的安装方法
- 理解 SQL Server 体系结构的特点和数据库引擎的作用
- 理解数据库和组成数据库的各种对象的类型与作用
- 熟练掌握 SQL Server Management Studio 工具的使用
- 掌握 SQL Server 2008 常用管理工具的使用

## 1.1 SQL Server 2008 概述

SQL Server 是 Microsoft 开发的基于关系数据库模型的数据库管理系统。数据库是存储数据的仓库，是长期存储在计算机内的、有组织的、可共享的数据集合。数据库中的数据具有结构化、最低冗余度、较高的程序与数据独立性、易于扩充、易于编制应用程序等优点。数据库中的数据由数据库管理系统统一管理，数据库管理系统(Database Management System, 简称 DBMS)是专门用于管理数据库的计算机系统软件。数据库管理系统能够为数据库提供数据的定义、建立、维护、查询和统计等功能，并完成对数据完整性、安全性进行控制的功能，它位于用户与操作系统之间。数据库管理系统的主要功能包括：数据库定义功能、数据库存取功能、数据库管理功能和数据库建立与维护功能。

本节主要介绍 SQL Server 2008 的体系结构、数据库和数据库对象以及 SQL Server 2008 的特点。

### 1.1.1 SQL Server 2008 的体系结构

SQL Server 2008 的体系结构是对 SQL Server 的组成部分和这些组成部分之间的描述。



Microsoft SQL Server 2008 系统由 4 部分组成, 这 4 个部分被称为 4 个服务, 分别是数据库引擎、Analysis Services、Reporting Services 和 Integration Services。这些服务之间的关系如图 1-1 所示。

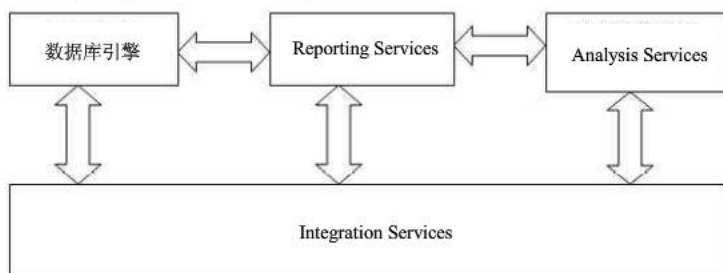


图 1-1 Microsoft SQL Server 2008 系统的体系结构图

- 数据库引擎是 Microsoft SQL Server 2008 系统的核心服务, 负责完成数据的存储、处理、查询和安全管理等操作。例如, 创建数据库、创建表、执行各种数据查询、访问数据库等操作, 都是由数据库引擎完成的。一般来说, 使用数据库系统实际上就是使用数据库引擎。数据库引擎也是一个复杂的系统, 它本身包含了许多功能组件, 例如, 复制、全文搜索、Service Broker 等。例如, 在商品库存管理系统中, 主要使用系统的数据库引擎来完成数据的添加、更新、删除、查询等操作。
- 分析服务(SQL Server Analysis Services, 简称为 SSAS)的主要作用是提供多维分析和数据挖掘功能。使用 Analysis Services, 用户可以设计、创建和管理包含来自于其他数据源数据的多维结构, 通过对多维数据进行多角度分析, 可以使管理人员对业务数据有更全面的理解。另外, 通过使用 Analysis Services, 用户可以完成数据挖掘模型的构造和应用, 实现知识的发现、表示和管理。
- 报表服务(SQL Server Reporting Services, 简称为 SSRS)为用户提供了支持 Web 方式的企业级报表功能。通过使用 Microsoft SQL Server 2008 系统提供的 Reporting Services, 用户可以方便地定义和发布符合需求的报表。无论是报表的布局格式, 还是报表的数据源, 用户都可以借助工具轻松地实现。这种服务极大地方便了企业的管理工作, 满足了管理人员对高效、规范管理的需求。例如, 在商品库存管理信息系统中, 使用 Microsoft SQL Server 2008 提供的 Reporting Services, 可以很方便地生成 Word、PDF、Excel 等格式的报表。
- 集成服务(SQL Server Integration Services, 简称 SSIS)是一个数据集成平台, 负责完成有关数据的提取、转换和加载等操作。例如, 对于 Analysis Services 来说, 数据库引擎是一个重要的数据源, 而如何将数据源中的数据经过适当的处理并加载到 Analysis Services 中, 以便进行各种分析处理, 这正是 Integration Services 所要解决的问题。重要的是, Integration Services 可以高效地处理各种各样的数据源, 例如, SQL Server、Oracle、Excel、XML 文档和文本文件等。

### 1.1.2 数据库和数据库对象

对 Microsoft SQL Server 2008 的体系结构有了了解之后, 下面介绍一下 Microsoft SQL

Server 2008 的数据库和数据库对象。

### 1. 数据库的类型和特点

Microsoft SQL Server 2008 系统提供了两种类型的数据库，即系统数据库和用户数据库。系统数据库存放 Microsoft SQL Server 2008 系统的系统级信息，例如，系统配置、数据库信息、登录帐户信息、数据库文件信息、数据库备份信息、警报和作业等。Microsoft SQL Server 2008 使用这些系统级信息管理和控制整个数据库服务器系统。用户数据库是由用户创建的、用来存放用户数据的数据库。Microsoft SQL Server 2008 系统的数据库类型如图 1-2 所示。

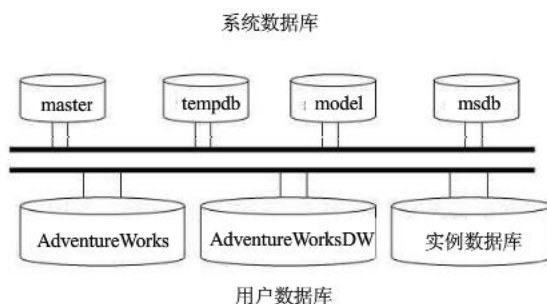


图 1-2 数据库类型示意图

Microsoft SQL Server 2008 安装成功之后，系统自动创建了 4 个系统数据库和多个用户示例数据库(如果在安装过程中选择了安装 Adventure Works 和 Adventure WorksDW 这两个用户示例数据库的话)。其中，4 个系统数据库分别是 master、model、msdb 和 tempdb 数据库。这些系统数据库的作用和特点如表 1-1 所示。

表 1-1 系统数据库的作用和特点

数 据 库	描 述
master	master 数据库是 SQL Server 的核心，如果该数据库被损坏，SQL Server 将无法正常工作。master 数据库记录了所有 SQL Server 系统级的信息，包括：登录帐户信息、系统配置信息、服务器配置信息、数据库文件信息以及 SQL Server 初始化信息等
tempdb	tempdb 是一个临时数据库，用于存储查询过程中的中间数据或结果。实际上，这是一个临时工作空间
model	model 数据库是其他数据库的模板数据库。当创建用户数据库时，系统自动把该模板数据库的所有信息复制到新建的数据库中。model 数据库是 tempdb 数据库的基础，对 model 数据库的任何改动都将反映在 tempdb 数据库中
msdb	msdb 数据库是一个与 SQL Server Agent 服务有关的数据库。该系统数据库记录有关作业、警报、操作员、调度等信息

### 2. 数据库对象

数据库是对象的容器，它不仅可以存储数据，而且能够使数据存储和检索以安全可靠的方式进行。数据库对象就是存储、管理和使用数据的不同结构形式。在 Microsoft SQL Server

在这些节点中，“数据库关系图”节点包含了数据库中的关系图对象，关系图对象用来描述数据库中表和表之间的关系。在数据库技术领域中，关系图有时也被称为 ER 图、ERD 图或 EAR 图等。

- “表”节点中包含了数据库最基本、最重要的对象——表。表实际用来存储系统数据和用户数据，是最核心的数据库对象。
- “视图”节点中包含了数据库中的视图对象。视图是一种虚拟表，用来查看数据库中的一个或多个表，视图是建立在表基础之上的数据库对象，它主要以 SELECT 语句形式存在。
- “同义词”节点中包含了数据库中的同义词对象。这是 Microsoft SQL Server 2008 系统新增的一种对象。同义词是数据库对象的另一个名称。通过使用同义词对象，可以大大简化对复杂数据库对象名称的引用。
- “可编程性”对象是一个逻辑组合，它包括存储过程、函数、触发器、程序集、类型、规则和默认值等对象。“存储过程”节点中包含了数据库中存储过程对象的信息。存储过程是封装了可重用代码的模块或例程。存储过程可以接受输入参数、向用户返回结果和消息、调用 Transact-SQL 语句并且返回输出参数。在 Microsoft SQL Server 2008 系统中，可以使用 Transact-SQL 或 CLR 语言来定义存储过程。
- 数据库中的函数对象包含在“函数”节点中。函数是接受参数、执行复杂操作并将结果以值的形式返回的例程。通过使用函数，可以达到模块化编程、执行速度更快、网络流量更少等效果。函数可以分为：表值函数、标量值函数、聚合函数和系统

函数。

- “数据库触发器”节点中包含了数据库中定义的触发器对象。触发器是一种特殊的存储过程，它在数据库服务器中发生指定的事件后自动执行。在 Microsoft SQL Server 2008 系统中，可以把触发器分为 DML 触发器和 DDL 触发器两大类。
- “程序集”节点中包含了数据库中的程序集对象。程序集是在 Microsoft SQL Server 2008 系统中使用的 DDL 文件，用于部署用 CLR 编写的函数、存储过程、触发器、用户定义聚合和用户定义类型等对象。
- “类型”节点中包含了系统数据类型、用户定义数据类型、用户定义类型和 XML 架构集合等对象类型。系统数据类型是 SQL Server 系统提供的数据类型，如，数值类型、日期类型等。用户定义数据类型是指用户基于系统数据类型定义的数据类型。用户定义类型扩展了 SQL Server 的类型系统，可以用于定义表中列的类型，或者 Transact-SQL 语言中的变量或例程参数的类型。XML 架构集合是一个类似数据库表的元数据实体，用于存储导入的 XML 架构。
- “规则”节点中包含了数据库中的规则对象。规则可以限制表中列值的取值范围，确保输入数据的正确性。实际上，规则是一种向后兼容的、用于执行与 CHECK 约束相同的功能。
- “默认值”节点中包含了数据库中的默认值对象。默认值对象也是一种完整性对象，它可以为表中的指定列提供默认值。
- “计划指南”节点中包含了数据库中的计划指南对象。计划指南是一种优化应用程序中查询语句性能的对象，它通过将查询提示或固定的查询计划附加到查询来影响查询的优化。这是 Microsoft SQL Server 2008 系统的新增功能。

除了上述节点表示的对象之外，数据库中的对象还包括消息类型、约定、队列、服务、路由、远程服务绑定、Broker 优先级等对象。

### 1.1.3 SQL Server 2008 的新增特点

Microsoft SQL Server 2008 与 Microsoft SQL Server 2005 相比，在性能、可靠性、实用性等方面都有了很大扩展和提高。与 Microsoft Visual Studio、Microsoft Office System 以及新的开发工具包(包括 Business Intelligence Development Studio)的紧密集成，使 Microsoft SQL Server 2008 与众不同。无论是开发人员、数据库管理员、信息工作者还是决策者，Microsoft SQL Server 2008 都可以为用户提供创新的解决方案，帮助用户从数据中获取更多的信息。

与以前的版本相比较，SQL Server 2008 具有以下新特性：

#### (1) 保护有价值的信息

透明的数据加密。SQL Server 2008 允许对整个数据库、数据文件和日志文件进行加密，而无需改动应用程序。进行加密可以使公司满足遵守规范及其关注数据隐私的要求。简单的数据加密的好处包括：使用任何范围或模糊查询搜索加密的数据、加强数据安全性以防止未授权的用户访问、还有对数据进行加密。这些可以在不改变已有应用程序的情况下进行。

可扩展的键管理。SQL Server 2008 为加密和键管理提供了一个全面的解决方案。它可以



支持第三方键管理和 HSM 产品, 以满足不断增长的需求。

审计。SQL Server 2008 通过 DDL 创建和管理审计, 同时通过提供更全面的数据审计来简化遵从性。允许组织回答常见的问题, 例如“检索什么数据”。

#### (2) 密钥管理

SQL Server 2008 为加密和密钥管理提供了一个全面的解决方案。为了满足不断发展的对数据中心信息的更强安全性的需求, 公司投资给供应商来管理公司内的安全密钥。SQL Server 2008 通过支持第三方密钥管理和硬件安全模块(HSM)产品为这个需求提供了很好的支持。

#### (3) 增强了审查

SQL Server 2008 使用户可以审查自己的数据, 从而提高了遵从性和安全性。审查不只包括对数据修改的所有信息, 而且还包括关于什么时候对数据进行读取的信息。SQL Server 2008 具有像服务器中加强的审查的配置和管理一样的功能, 这使得公司可以满足各种规范需求。SQL Server 2008 还可以定义每一个数据库的审查规范, 所以审查配置可以为每个数据库作单独的制定。为指定对象作审查配置使审查的执行性能更好, 配置的灵活性也更高。

#### (4) 确保业务连续性

增强的数据库镜像。SQL Server 2008 构建于 SQL Server 2005 之上, 并提供了更可靠的加强了数据库镜像的平台。新的特性包括:

- 页面自动修复。SQL Server 2008 通过请求获得一个从镜像合作机器上得到的出错页面的拷贝, 使主要的和镜像的计算机可以透明地修复数据页面上的 823 和 824 错误。
- 提高了性能。SQL Server 2008 压缩了输出的日志流, 以便使数据库镜像所要求的网络带宽达到最小。

#### (5) 加强了可支持性

SQL Server 2008 新增加了执行计数器, 它使得可以更细粒度地对数据库管理系统(Database Management System, DBMS)日志记录的不同阶段所耗费的时间进行计时。

SQL Server 2008 包括动态管理视图(Dynamic Management View)和对现有视图的扩展, 以此来显示镜像会话的更多信息。

#### (6) 热添加 CPU

为了在线添加内存资源而扩展 SQL Server 中已有的支持, 热添加 CPU 使数据库可以按需扩展。事实上, CPU 资源可以添加到 SQL Server 2008 所在的硬件平台上而不需要停止应用程序。

## 1.2 SQL Server 2008 的安装

安装和配置是使用 Microsoft SQL Server 2008 系统的开端工作, 正确地安装和配置系统是确保软件系统安全、健壮、高效运行的基础。

在安装 Microsoft SQL Server 2008 之前, 必须先做好一些准备工作。其中, 了解系统的版本特点和系统对安装环境的要求是最基本的准备工作。本节将详细介绍 Microsoft SQL Server 2008 系统的版本特点、对软硬平台的要求以及系统的安装操作。

### 1.2.1 系统版本

SQL Server 2008 提供了多个不同的版本，不同的应用需求，往往需要安装不同的版本。既有 32 位的版本，也有 64 位的版本；既有正式使用的服务器版本，也有满足特殊需要的专业版本。其中，服务器版本包括企业版和标准版，专业版本主要包括以下版本：工作组版(Workgroup)、开发人员版(Developer)、免费精简版(Express)、Web 版，以及免费的集成数据库 SQL Server Compact 3.5。

SQL Server 是微软应用平台中的一个关键组成部分，微软应用平台旨在帮助客户建立、运行和管理动态的业务应用。SQL Server 2008 提供了如下几个版本：

- **SQL Server 2008 企业版：**SQL Server 2008 企业版可以用作一个企业的数据库服务器，是一个全面的数据管理和商业智能平台，提供企业级的可扩展性、数据库、安全性，以及先进的分析和报表支持，从而运行关键业务应用。此版本可以整合服务器及运行大规模的在线事务处理。
- **SQL Server 2008 标准版：**SQL Server 2008 标准版可以用作一般企业的数据库服务器，它包括电子商务、数据仓库、业务流程等最基本的功能，是一个完整的数据管理和商业智能平台，提供业界最好的易用性和可管理性以运行部门级应用。
- **SQL Server 2008 工作组版：**SQL Server 2008 工作组版是一个可信赖的数据管理和报表平台，为各分支应用程序提供安全、远程同步和管理能力。此版本包括核心数据库的特点并易于升级到标准版或企业版。
- **SQL Server 2008 网络版：**SQL Server 2008 网络版是为运行于 Windows 服务器上的高可用性、面向互联网的网络环境而设计的。此版本为客户提供了必要的工具，以支持低成本、大规模、高可用性的网络应用程序或主机托管解决方案。
- **SQL Server 2008 开发版：**SQL Server 2008 开发版的主要用户是独立软件供应商、创建和测试数据库应用程序的开发人员、系统集成商等。使开发人员能够用 SQL Server 建立和测试任何类型的应用程序。此版本的功能与 SQL Server 企业版的功能相同，但不适用于普通的数据库用户。从功能上讲，该版本等价于企业版，但在并发查询等方面有着很大的性能限制。在此版本上开发的应用程序和数据库用户可以根据需要很容易地升级到企业版。
- **SQL Server 2008 Express 版本：**SQL Server 2008 Express 版是 SQL Server 的一个免费版本，提供核心数据库功能，包括 SQL Server 2008 所有的新数据类型。此版本旨在提供学习和创建桌面应用程序与小型服务器应用程序并可被 ISV 重新发布的免费版本。
- **SQL Server 移动版：**SQL Server 移动版是为开发者设计的一个免费的嵌入式数据库，旨在为移动设备、桌面和网络客户端创建一个独立运行并适时联网的应用程序。SQL Server 移动版可以在微软所有 Windows 的平台上运行，包括 Windows XP、Windows Vista 操作系统，以及 Pocket PC 和智能手机设备。SQL Server 2008 可以为关键业务应用提供可信赖的、高效的、智能的平台，支持基于策略的管理、审核、大规模数据仓库、空间数据、高级报告与分析服务等新特性。

## 1.2.2 SQL Server 2008 的运行环境要求

Microsoft SQL Server 2008 系统的不同版本对硬件、操作系统和 Internet 网络等环境的要求也不尽相同。本节将介绍各种版本的硬件环境要求、操作系统要求、Internet 要求以及网络要求等。值得注意的是, Microsoft SQL Server 2008 系统对 32 位平台和 64 位平台的要求也是不同的。

### 1. 硬件环境要求

如表 1-2 所示的要求适用于所有 SQL Server 2008 安装。

表 1-2 SQL Server 2008 安装的环境要求

组 件	要 求
框架 2	SQL Server 安装程序安装该产品所需的以下软件组件: .NET Framework 3.5 SP1 SQL Server Native Client SQL Server 安装程序支持文件
软件 2	SQL Server 安装程序要求使用 Microsoft Windows Installer 4.5 或更高版本 安装了所需的组件后, SQL Server 安装程序将验证要安装 SQL Server 2008 R2 的计算机是否也满足成功安装所需的所有其他要求
网络软件	SQL Server 2008 64 位版本的网络软件要求与 32 位版本的要求相同 支持的操作系统都具有内置网络软件。独立的命名实例和默认实例支持以下网络 协议: Shared memory Named Pipes TCP/IP VIA 注意: 故障转移群集不支持 Shared memory 和 VIA
虚拟化	在以 Windows Server 2008 SP2 Standard、Enterprise 和 Datacenter 版本中的 Hyper-V 角色运行的虚拟机环境中支持 SQL Server 2008 R2。虚拟机必须运行本主题稍后部分中列出的特定 SQL Server 2008 R2 版本所支持的操作系统 除了父分区所需的资源以外, 还必须为每个虚拟机(子分区)的 SQL Server 2008 R2 实例提供足够的处理器资源、内存和磁盘资源。具体要求在本主题的稍后部分中列出 在 Windows Server 2008 SP2 上的 Hyper-V 角色中, 最多可以为运行 Windows Server 2008 SP2 32 位或 64 位版本的虚拟机分配 4 个虚拟处理器。最多可以为运行 Windows Server 2003 32 位版本的虚拟计算机分配 2 个虚拟处理器。对于承载其他操作系统的虚拟计算机, 最多可以为虚拟计算机分配一个虚拟处理器 注意: 建议在关闭虚拟机之前先关闭 SQL Server 2008 R2。
Internet 软件	所有的 SQL Server 2008 R2 安装都需要使用 Microsoft Internet Explorer 6 SP1 或更高版本。 Microsoft 管理控制台(MMC)、SQL Server Management Studio、Business Intelligence Development Studio、Reporting Services 的报表设计器组件和 HTML 帮助都需要 Internet Explorer 6 SP1 或更高版本



(续表)

组 件	要 求
硬盘	磁盘空间要求随所安装的 SQL Server 2008 组件不同而发生变化
驱动器	从磁盘进行安装时需要相应的 CD 或 DVD 驱动器
显示器	SQL Server 2008 R2 图形工具需要使用 VGA 或更高分辨率: 分辨率至少为 1 024×768 像素
其他设备	指针设备: 需要 Microsoft 鼠标或兼容的指针设备

(1) 以下 .NET Framework 版本是必需的:

- Windows Server 2003(64 位)IA64 上的 SQL Server 2008 R2: .NET Framework 2.0 SP2
- SQL Server Express: .NET Framework 2.0 SP2
- SQL Server 2008 R2 的所有其他版本: .NET Framework 3.5 SP1

安装 .NET Framework 需要重新启动操作系统。如果安装 Windows Installer 也需要重新启动操作系统, 安装程序将等到 .NET Framework 和 Windows Installer 组件都完成安装后才重新启动。

注意:

.NET Framework 2.0 SP2 不作为单独的下载提供, 用户需要安装其中包括 .NET Framework 2.0 SP2 的 .NET Framework 3.5 SP1。

(2) SQL Server 安装程序将不安装 SQL Server Express、SQL Server Express with Tools 和 SQL Server Express with Advanced Services 所需的以下组件。因此, 在运行 SQL Server 安装程序之前, 必须手动安装这些组件:

- SQL Server Express: .NET Framework 2.0 SP2 和 Windows Installer 4.5。在 Windows Vista SP2 上, 请使用 .NET Framework 3.5 SP1。
- SQL Server Express with Advanced Services: .NET Framework 3.5 SP1、Windows Installer 4.5 和 Windows PowerShell 1.0。
- SQL Server Express with Tools: .NET Framework 3.5 SP1、Windows Installer 4.5 和 Windows PowerShell 1.0。

(3) 与所有虚拟化技术一样, 在 Windows Server 2008 SP2 Hyper-V 虚拟计算机上运行的 SQL Server 2008 R2 将比在具有相同物理资源的物理计算机上运行的慢。

## 2. 处理器、内存和操作系统要求

Reporting Services: 从 SQL Server 2008 R2 开始, Reporting Services 组件不再支持运行 Windows Server 2003 或 Windows Server 2003 R2 的基于 Itanium 的服务器。Reporting Services 继续支持其他 64 位操作系统, 包括 Windows Server 2008 for Itanium-Based Systems 和 Windows Server 2008 R2 for Itanium-Based Systems。如果要在 Windows Server 2003 或 Windows Server 2003 R2 的基于 Itanium 的系统版本上从具有 Reporting Services 的 SQL Server

2008 安装升级到 SQL Server 2008 R2, 则必须首先升级操作系统。

**WOW64:** WOW64 是 Windows 64 位版本中的一个功能, 使用该功能可以以 32 位模式在本机运行 32 位应用程序。尽管基础操作系统是在 64 位操作系统上运行, 但应用程序可以以 32 位模式工作。仅对于 SQL Server 的独立实例才支持 WOW64。SQL Server 故障转移群集安装不支持 WOW64。对于在支持的 64 位操作系统上安装的 SQL Server 64 位版本, WOW64 中支持管理工具。有关支持的操作系统的详细信息, 请在下面各节中选择 SQL Server 2008 R2 的版本。在支持的 64 位操作系统上, SQL Server 32 位版本可以安装到 64 位服务器的 Windows on Windows (WOW64) 32 位子系统中。

其他操作系统备注: Windows Server 2008 SP2 服务器核心或 Windows Server 2008 R2 服务器核心安装不支持 SQL Server 2008 R2。SQL Server Standard for Small Business 还支持在 SQL Server Standard 部分列出的操作系统。Windows Small Business Server 64 位 x64 版本还支持 Windows Server 2008 R2 64 位 x64 Standard 版本所支持的 SQL Server 版本。

### 1.2.3 Microsoft SQL Server 2008 的安装

上一节介绍了 Microsoft SQL Server 2008 对环境的要求, 下面以 Microsoft SQL Server 2008 标准版为例, 介绍如何安装 Microsoft SQL Server 2008 数据库系统。

#### 1. 安装 SQL Server 2008 标准版之前应该注意的问题

##### (1) 增强物理安全性

物理和逻辑隔离是构成 SQL Server 安全的基础。如果要增强 SQL Server 安装的物理安全性, 则需要执行以下任务:

- 将服务器置于专门的机房, 未经授权的人员不得入内。
- 将数据库的宿主计算机置于受物理保护的场所, 最好是上锁的机房, 房中配备水灾检测和火灾检测监视系统以及灭火系统。
- 将数据库安装在公司 Intranet 的安全区域中, 并且不能将 SQL Server 直接连接到 Internet。
- 定期备份所有数据, 并将备份存储在远离工作现场的安全位置。

##### (2) 使用防火墙

防火墙对于协助确保 SQL Server 安装的安全十分重要。若要使防火墙发挥最佳效用, 则要遵循以下指南:

- 在服务器和 Internet 之间放置防火墙。启用防火墙, 如果防火墙处于关闭状态, 请将其开启; 如果防火墙处于开启状态, 请不要将其关闭。
- 将网络分成若干个安全区域, 区域之间用防火墙分隔。先阻塞所有通信流量, 然后有选择地只接受所需的通信。
- 在多层环境中, 使用多个防火墙创建屏蔽子网。
- 如果在 Windows 域内部安装服务器, 需要将内部防火墙配置为允许使用 Windows 身份验证。

- 如果应用程序使用分布式事务处理,则必须要将防火墙配置为允许 Microsoft 分布式事务处理协调器(MS DTC)在不同的 MS DTC 实例之间进行通信。还需要将防火墙配置为允许在 MS DTC 和资源管理器(如 SQL Server)之间进行通信。

### (3) 隔离服务

隔离服务可以降低风险,防止已受到危害的服务被用于危及其他服务。若要隔离服务,需要考虑如下原则:在不同的 Windows 帐户下运行各自的 SQL Server 服务。对每个 SQL Server 服务,尽可能使用不同的低权限 Windows 或本地用户帐户;配置安全的文件系统。

使用正确的文件系统可提高安全性。对于 SQL Server 的安装,应执行以下几项任务:

- 使用 NTFS 文件系统。NTFS 是安装 SQL Server 的首选文件系统,因为它比 FAT 文件系统更加稳定且更容易恢复。NTFS 还可以使用安全选项,例如,文件和目录访问控制列表(ACL)和加密文件系统(EFS)文件加密。在安装期间,如果检测到 NTFS,SQL Server 将对注册表项和文件设置相应的 ACL。请不要对这些权限做任何更改。SQL Server 的未来版本可能不支持在具有 FAT 文件系统的计算机上进行安装。
- 对关键数据文件使用独立磁盘冗余阵列(RAID)。

### (4) 禁用 NetBIOS 和服务器消息块

外围网络中的服务器应禁用所有不必要的协议,包括 NetBIOS 和服务器消息块(SMB)。

NetBIOS 使用以下端口:

- UDP/137(NetBIOS 名称服务)
- UDP/138(NetBIOS 数据报服务)
- TCP/139(NetBIOS 会话服务)

SMB 使用以下端口:

- TCP/139
- TCP/445

Web 服务器和域名系统(DNS)服务器不需要 NetBIOS 和 SMB。在这些服务器上,禁用这两个协议可以减轻由用户枚举带来的威胁。

## 2. 安装 Microsoft SQL Server 2008 标准版的步骤

安装 SQL Server 2008 标准版的步骤如下:

(1) 插入 SQL Server 安装介质,双击根目录中的 setup.exe 文件。如果是从网络共享进行安装,则导航到共享中的根文件夹,然后双击 setup.exe。此时将启动 Microsoft SQL Server 2008 安装程序,如图 1-4 所示。

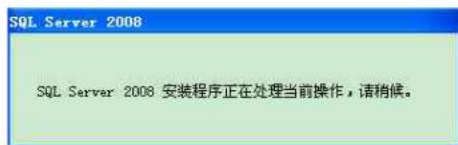


图 1-4 Microsoft SQL Server 2008 安装对话框

(2) 如果检测到需要安装必备组件,则出现安装必备组件的选项。单击“确定”按钮安

装必备组件,单击“取消”按钮将退出 SQL Server 2008 安装。如果出现.NET Framework 3.5 SP1 安装对话框,需要选中相应的复选框,接受.NET Framework 3.5 SP1 许可协议。单击“下一步”按钮继续。当.NET Framework 3.5 SP1 安装完成后,单击“完成”按钮。

(3) Windows Installer 4.5 也是必需的,并且可能由安装向导进行安装。如果系统提示重新启动计算机,则重新启动计算机,然后再次启动 SQL Server 2008 setup.exe。

(4) 必备组件安装完成后,安装向导会立即启动 SQL Server 安装中心,如图 1-5 所示。若要创建 SQL Server 2008 的全新安装,在左侧选择“安装”选项,然后单击“全新安装或向现有安装添加功能”链接。



图 1-5 SQL Server 安装中心

(5) 系统配置检查器将在计算机上运行发现操作。单击“确定”按钮继续。此时,系统已经为安装创建了安装日志文件。

(6) 接着出现“安装程序支持文件”对话框,如图 1-6 所示,单击“安装”按钮开始安装。

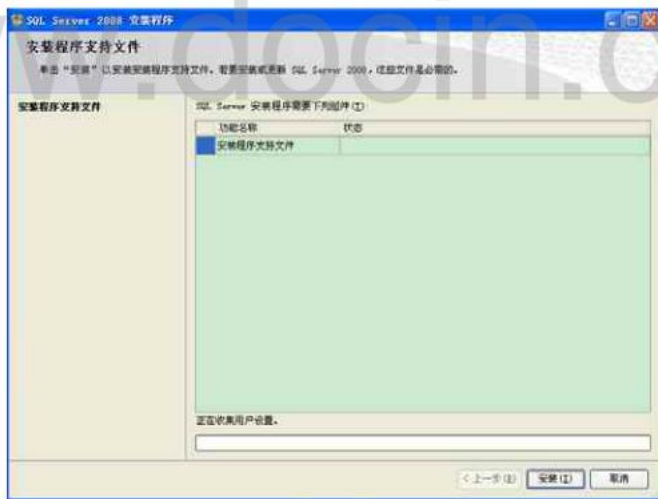


图 1-6 “安装程序支持文件”对话框



(7) 接着出现“安装程序支持规则”对话框，如果 1-7 所示。这里必须更正所有的失败后才能进行安装。



图 1-7 “安装程序支持规则”对话框

(8) 单击“下一步”按钮进入“安装类型”页，选择“执行 SQL Server 2008 的全新安装”单选按钮，如果 1-8 所示。



图 1-8 “安装类型”页

(9) 单击“下一步”按钮，出现“产品密钥”页，选择相应的单选按钮，以指示是安装免费版本的 SQL Server，还是拥有该产品生产版本的 PID 密钥，如图 1-9 所示。

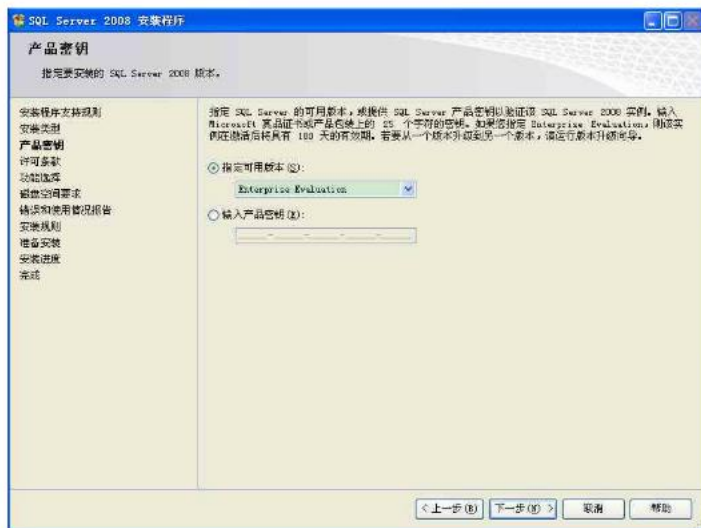


图 1-9 “产品密钥”页

(10) 单击“下一步”按钮，在“许可条款”页上阅读许可协议，然后选中相应的复选框接受许可条款和条件。如图 1-10 所示，单击“下一步”按钮继续。若要结束安装程序，可单击“取消”按钮。



图 1-10 “许可条款”页

(11) 单击“下一步”按钮继续，在“功能选择”页上选择要安装的组件。选择功能名称后，右侧窗格中会显示该组件的说明，可以选中任意一些复选框，如图 1-11 所示。

用户也可以使用此页底部的字段为共享组件指定自定义目录。若要更改共享组件的安装路径，可更新对话框底部字段中所提供的路径名，或单击“浏览”按钮指定另一个安装目录。默认安装路径为 C:\Program Files\Microsoft SQL Server\。

(12) 单击“下一步”按钮，在“实例配置”页中指定是安装默认实例还是命名实例，如

图 1-12 所示。

- 实例 ID: 默认情况下, 使用实例名称作为实例 ID 的后缀。这用于标识 SQL Server 实例的安装目录和注册表项。默认实例和命名实例的默认方式都是如此。对于默认实例, 实例名称和实例 ID 后缀均为 MSSQLSERVER。若要使用非默认的实例 ID, 则选中“命名实例”单选按钮, 并提供一个值。
- 实例根目录: 默认情况下, 实例根目录为 C:\Program Files\Microsoft SQL Server\。若要指定一个非默认的根目录, 则单击“浏览”按钮以找到一个安装目录。
- 已安装的实例: 该表格将显示运行安装程序的计算机上的 SQL Server 实例。如果计算机上已经安装了一个默认实例, 则必须安装 SQL Server 2008 的命名实例。



图 1-11 “功能选择”页

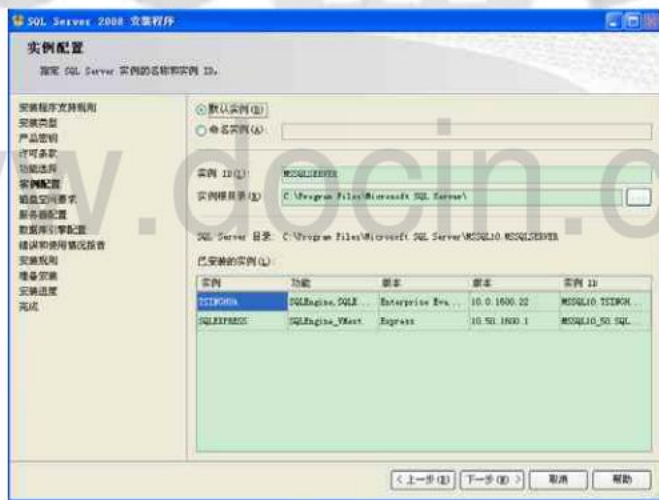


图 1-12 “实例配置”页

(13) 单击“下一步”按钮, 出现“磁盘空间要求”页, 计算出指定的组件所需的磁盘空间, 与可用磁盘空间进行比较, 如图 1-13 所示。





图 1-13 “磁盘空间要求”页

本主题中的其余工作流取决于要安装的功能。

(14) 在“服务器配置”页的“服务帐户”选项中指定 SQL Server 服务的登录帐户。此页上配置的实际服务取决于前面选择安装的功能，如图 1-14 所示。

用户可以为所有的 SQL Server 服务分配相同的登录帐户，也可以单独配置各个服务帐户。还可以指定服务是自动启动、手动启动还是禁用。Microsoft 建议对各服务帐户进行单独配置，以便为每项服务提供最低特权，即向 SQL Server 服务授予它们完成各自任务所需的最低权限。

如果要为此 SQL Server 实例中的所有服务帐户指定相同的登录帐户，则在页面底部的字段中提供凭据。

安全说明：请不要使用空密码，建议使用强密码。

为 SQL Server 服务指定登录信息后，单击“下一步”按钮继续。

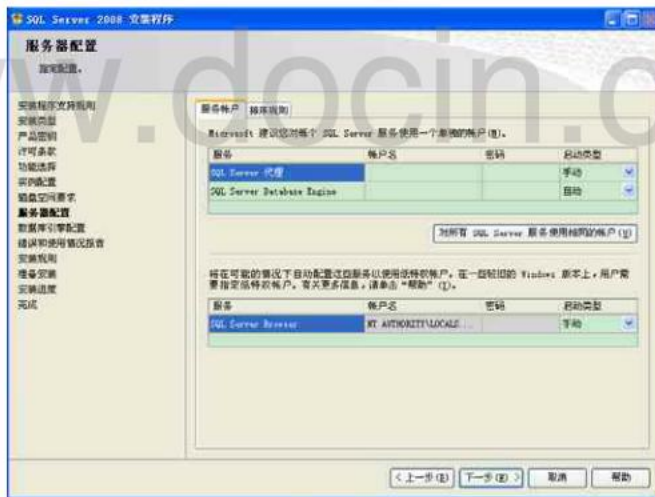


图 1-14 “服务器配置 - 服务帐户”页

(15) 使用“服务器配置”页的“排序规则”选项卡为数据库引擎和 Analysis Services 指定非默认的排序规则。

(16) 单击“下一步”按钮，使用“数据库引擎配置”页的“帐户设置”选项卡指定以下事项，如图 1-15 所示。

**身份验证模式：**为 SQL Server 实例选择 Windows 身份验证或混合模式身份验证。如果选择“混合模式”身份验证，则必须为内置 SQL Server 系统管理员帐户提供一个强密码。在设备与 SQL Server 成功建立连接之后，用于 Windows 身份验证和混合模式身份验证的安全机制是相同的。

**SQL Server 管理员：**必须为 SQL Server 实例指定至少一个系统管理员。如果要添加用以运行 SQL Server 安装程序的帐户，可单击“添加当前用户”按钮。如果要向系统管理员列表中添加帐户或从中删除帐户，可单击“添加”或“删除”按钮，然后编辑将拥有 SQL Server 实例的管理员特权的用户、组或计算机的列表。



图 1-15 “数据库引擎配置 - 帐户设置”页

(17) 在“数据目录”选项卡中指定非默认的安装目录。若要安装到默认目录，直接单击“下一步”按钮。

(18) 在 FILESTREAM 选项卡中对 SQL Server 实例启用 FILESTREAM。然后单击“下一步”按钮继续。

(19) 使用“Analysis Services 配置”页的“帐户设置”选项卡指定拥有 Analysis Services 的管理员权限的用户或帐户。必须为 Analysis Services 指定至少一个系统管理员。如果要添加用以运行 SQL Server 安装程序的帐户，可以单击“添加当前用户”按钮。若要向系统管理员列表中添加帐户或从中删除帐户，可单击“添加”或“删除”按钮，然后编辑将拥有 Analysis Services 的管理员特权的用户、组或计算机的列表。

完成对该列表的编辑后,单击“确定”按钮。验证配置对话框中的管理员列表。单击“下一步”按钮继续。

(20) 使用“Analysis Services 配置”页的“数据目录”选项卡指定非默认的安装目录,如图 1-16 所示。如果要安装到默认目录,直接单击“下一步”按钮继续。

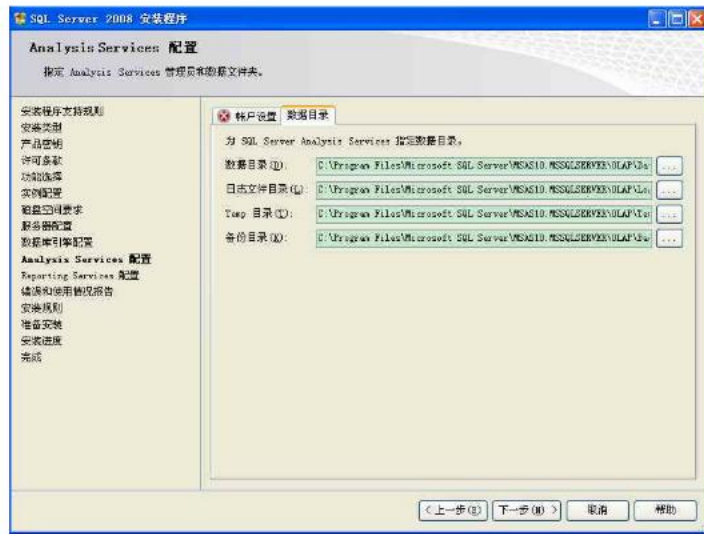


图 1-16 “Analysis Services 配置 - 数据目录”页

(21) 使用“Reporting Services 配置”页指定要创建的 Reporting Services 安装的类型,如图 1-17 所示。包括以下选项:

- 安装本机模式默认配置
- 安装 SharePoint 集成模式默认配置
- 安装并不配置报表服务器



图 1-17 “Reporting Services 配置”页



(22) 在“错误和使用情况报告”页上指定是否要发送到 Microsoft 以帮助改善 SQL Server 的信息。默认情况下,用于错误报告和使用情况的选项处于启用状态,如图 1-18 所示。

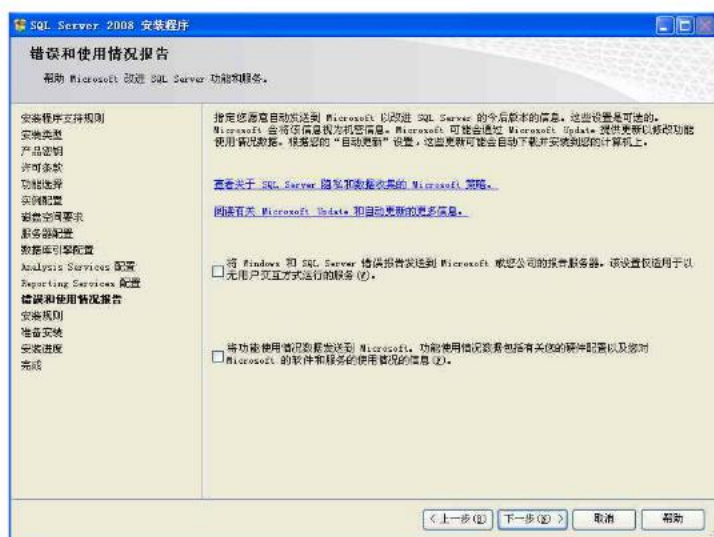


图 1-18 “错误和使用情况报告”页

(23) 系统配置检查器将再运行一组规则来针对用户指定的 SQL Server 功能验证计算机配置。

(24) “准备安装”页显示安装期间指定的安装选项的树视图。单击“安装”按钮开始安装,如图 1-19 所示。

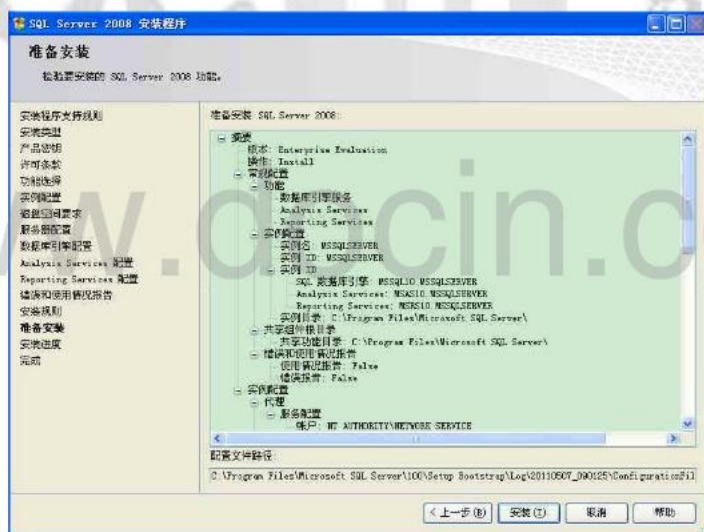


图 1-19 “准备安装”页

(25) 在安装过程中,“安装进度”页会提供相应的状态,以便在安装过程中监视安装进度,如图 1-20 所示。

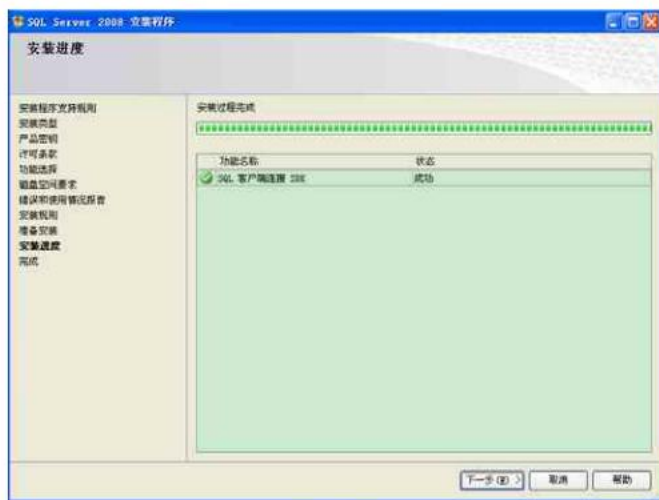


图 1-20 “安装进度”页

(26) 安装完成后，“完成”页会提供指向安装摘要日志文件以及其他重要说明的链接。单击“关闭”按钮完成 SQL Server 的安装，如图 1-21 所示。

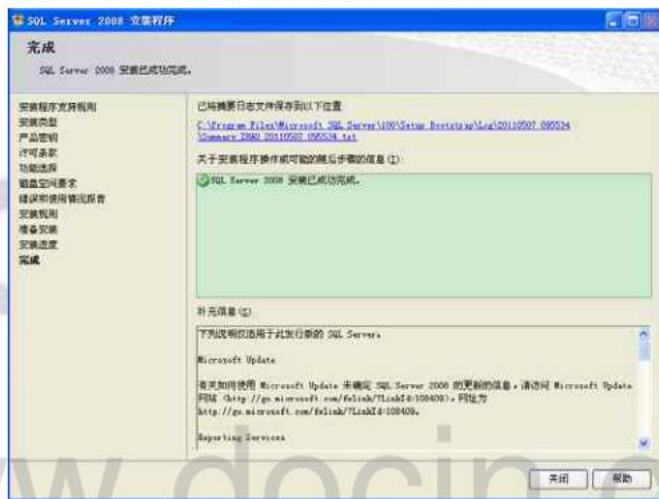


图 1-21 “完成”页

(27) 如果安装程序指示要重新启动计算机，请立即重新启动。安装完成后，请务必阅读来自安装向导的消息。

### 1.3 SQL Server 2008 的配置

SQL Server 2008 是运行于网络环境下的数据库管理系统，它支持网络中不同计算机上的多个用户同时访问和管理数据库资源。服务器是 SQL Server 2008 数据库管理系统的核心，它为客户提供网络服务，使用户能够远程访问和管理 SQL Server 数据库。配置服务器的过程

就是为了充分利用 Microsoft SQL Server 系统资源，而设置 Microsoft SQL Server 服务器默认行为的过程。合理地配置服务器，可以加快服务器响应请求的速度、充分利用系统资源、提高系统的工作效率。本节将重点讲述如何注册服务器以及如何配置服务器选项。

### 1.3.1 注册服务器

为了管理、配置和使用 Microsoft SQL Server 2008 系统，必须使用 Microsoft SQL Server Management Studio 工具注册服务器。注册服务器就是为 Microsoft SQL Server 客户机/服务器系统确定数据库所在的一台机器，该机器将作为服务器，为客户端的各种请求提供服务。

#### 1. 注册服务器

使用 Microsoft SQL Server Management Studio 工具注册服务器的操作步骤如下：

(1) 启动 Microsoft SQL Server Management Studio 工具，选择“视图”|“已注册的服务器”命令或者按快捷键 Ctrl+Alt+G，在打开的“已注册的服务器”窗口中选中“数据库引擎”图标。

(2) 选择“数据库引擎”节点下面的“本地服务器组”，单击鼠标右键，从弹出的快捷菜单中选择“新建服务器注册”命令，即可打开如图 1-22 所示的“新建服务器注册”对话框。选择“常规”选项卡，可以在该选项卡中输入将要注册的服务器名称。在“服务器名称”下拉列表框中，既可以输入服务器名称，也可以从列表中选择一个服务器名称。从“身份认证”下拉列表框中可以选择身份验证模式，在此选择“Windows 身份验证”。

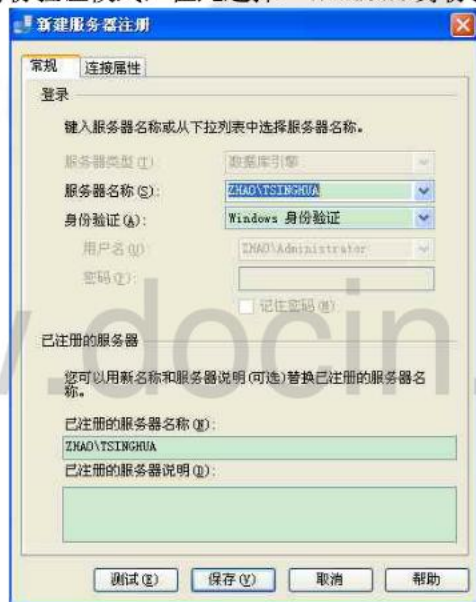


图 1-22 “新建服务器注册”对话框的“常规”选项卡

(3) 选择“连接属性”选项卡，如图 1-23 所示，在该选项卡中可以设置连接到的数据库、网络以及其他连接属性。在“连接到数据库”下拉列表框中可以指定用户将要连接到的数据库名称。如果选择“<默认值>”选项，即表示连接到 Microsoft SQL Server 系统中当前用户的默认数据库。



图 1-23 “新建服务器注册”对话框的“连接属性”选项卡

如果选择“<浏览服务器>”选项，则会出现如图 1-24 所示的“查找服务器上的数据库”对话框，在该对话框中可以指定该服务器注册连接的数据库。



图 1-24 “查找服务器上的数据库”对话框

(4) 然后单击“测试”按钮，可以对当前设置的连接属性进行测试。如果出现如图 1-25 所示的“新建服务器注册”消息框，则表示连接属性的设置是正确的。

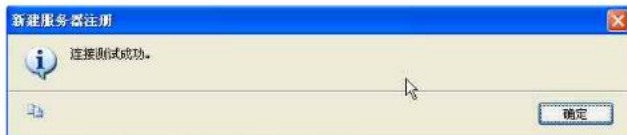


图 1-25 “新建服务器注册”消息框



(5) 完成连接属性设置后,单击“保存”按钮,即可完成连接属性的设置。接着单击“保存”按钮,完成新建服务器注册的操作。新注册的服务器名称将出现在列表中。

## 2. 修改服务器的注册

在查询分析器中,在需要修改的服务器名称上单击鼠标右键,从弹出的快捷菜单中选择“编辑 SQL Server 注册属性”命令,在打开的对话框中,用户可以修改身份验证模式和服务器组等属性。

## 3. 删除服务器

在需要删除的服务器名称上单击鼠标右键,从弹出的快捷菜单中选择“删除”命令。在弹出的“确认删除”对话框中单击“是”按钮即可完成删除操作,如图 1-26 所示。



图 1-26 “确认删除”对话框

需要说明以下几点:

(1) 第一次运行 SQL Server 2008 查询分析器时,它将自动注册本地 SQL Server 2008 所有已经安装的实例。

(2) 如果有一个已注册的 SQL Server 实例,接着又安装了更多的 SQL Server 实例,则只注册第一个 SQL Server 实例。可以启动注册服务器向导或使用“已注册的 SQL Server 属性”对话框来注册其他的服务器。

(3) 如果连接到远程服务器有困难,可以使用客户端网络工具来配置对该服务器的访问。

### 1.3.2 配置服务器选项

在 Microsoft SQL Server 2008 中,可以使用 Microsoft SQL Server Management Studio、sp\_configure 系统存储过程、SET 语句等方式来配置服务器。下面,以使用 Microsoft SQL Server Management Studio 配置服务器为例来说明如何配置服务器选项。

配置服务器选项的步骤如下:

(1) 在 Microsoft SQL Server Management Studio 中的“对象资源管理器”中右击需要配置的服务器名称,从弹出的快捷菜单中选择“属性”命令,打开如图 1-27 所示的“服务器属性”对话框。该对话框包含 8 个选项页,通过这 8 个选项页可以查看或设置服务器的常用选项值。

“常规”选项页如图 1-27 所示,该选项页列出了当前服务器的产品名称、操作系统名称、平台名称、版本号、使用的语言、当前服务器的最大内存数量、当前服务器的处理器数量、当前 SQL Server 安装的根目录、服务器使用的排序规则以及是否已经群集化等信息。

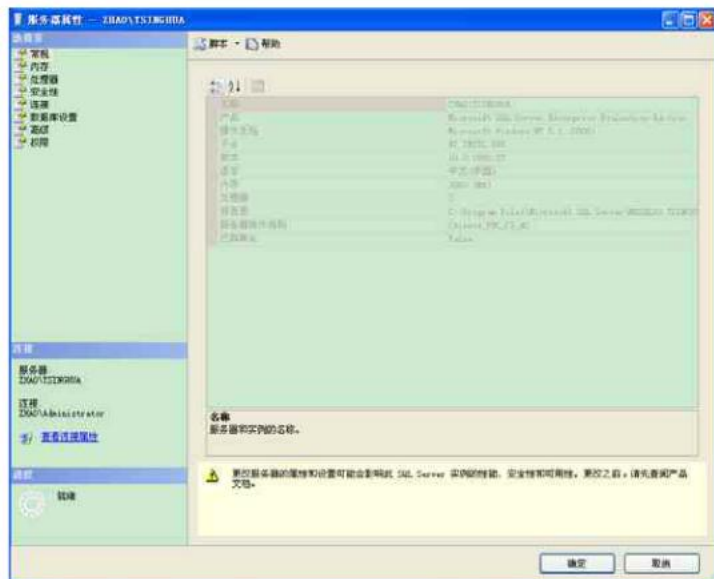


图 1-27 “服务器属性”对话框的“常规”选项页

(2) “服务器属性”对话框的“内存”选项页如图 1-28 所示，在该选项页中可以设置与内存管理相关的选项。



图 1-28 “服务器属性”对话框的“内存”选项页

“使用 AWE 分配内存”复选框表示在当前服务器上使用 AWE 技术执行超大物理内存，从理论上讲，32 位地址最多可以映射 4GB 内存，但是，通过使用 AWE 技术，SQL Server 系统可以使用远远超过 4GB 的内存空间。一般情况下，只有在使用大型数据库系统时才选中该复选框。

如果要设置服务器可以使用的内存范围,可以通过“最小服务器内存(MB)”和“最大服务器内存(MB)”两个文本框来完成设置。

如果希望指定索引占用的内存,可以设置“创建索引占用的内存”文本框来完成。当“创建索引占用的内存”文本框中的值为0时,表示系统动态为索引分配内存。查询也需要耗费内存,在“每次查询占用的最小内存(KB)”文本框中可以设置查询所占内存的大小,默认值为1024。

需要说明的是,该选项页上还有两个单选按钮,即“配置值”和“运行值”。“配置值”指的是当前设置了但是还没有真正起作用的选项值;“运行值”则指的是当前系统正在使用的选项值。在对某个选项进行设置之后,选中“运行值”单选按钮可以查看该设置是否立即生效。如果这些设置不能立即生效,则必须重新启动服务器才能使设置生效。

(3) “服务器属性”对话框的“处理器”选项页如图1-29所示,在该选项页中,可以设置与服务器的处理器相关的选项。只有服务器上安装了多个处理器时,“处理器关联”和“I/O 关联”才有意义。



图 1-29 “服务器属性”对话框的“处理器”选项页

在 Windows 操作系统中,有时为了执行多个任务,需要在不同的处理器之间进行移动,以便处理多个线程。但是,这种移动会使处理器缓存不断地重新加载数据,从而降低 Microsoft SQL Server 系统的性能。如果事先将每个处理器分配给特定的线程,则可以避免处理器缓存重新加载数据,从而提高 Microsoft SQL Server 系统的性能。线程与处理器之间的这种关系称为处理器关联。



“最大工作线程数”选项用于设置 Microsoft SQL Server 进程的工作线程数。如果客户端比较少,可以为每一个客户端设置一个线程;如果客户端很多,则可以为这些客户端设置一个工作线程池。当该值为 0 时,表示系统动态分配线程。最大线程数受服务器硬件的限制,例如,当服务器的 CPU 个数低于 4 个时,32 位机器的最大可用线程数为 256,64 位机器的最大可用线程数为 512。

“提升 SQL Server 的优先级”复选框表示设置 SQL Server 进程的优先级高于操作系统上的其他进程。一般情况下,选中“使用 Windows 纤程(轻型池)”复选框,可以通过减少上下文的切换频率来提高系统的吞吐量。

(4) “服务器属性”对话框的“安全性”选项页如图 1-30 所示,在该选项页中,可以设置与服务器身份认证模式、登录审核等安全性相关的选项。



图 1-30 “服务器属性”对话框的“安全性”选项页

在该选项页中可以修改系统的身份验证模式。通过设置登录审核功能,可以将用户的登录结果记录在错误日志中。如果选中“无”单选按钮,表示不对登录过程进行审核;选中“仅限失败的登录”单选按钮,表示只记录登录失败的事件;选中“失败和成功的登录”单选按钮,表示无论是登录失败事件还是登录成功事件都将记录在错误日志中,以便对这些登录事件进行跟踪和审核。这种登录审核仅仅是对登录事件的审核。

如果要对执行某条语句的事件进行审核、对使用某个数据库对象的事件进行审核,可以选中“启用 C2 审核跟踪”复选框,该选项可以在日志文件中记录对语句、对象访问的事件。

如果选中“启用服务器代理帐户”复选框,则需要指定代理帐户名称和密码。注意:如果服务器代理帐户的权限过大,有可能被恶意用户利用,形成安全漏洞,危及系统的安全。

因此,服务器代理帐户应该只具有执行既定工作所需的最低权限。

所有权链接通过对某个对象的权限进行设置,允许对多个对象的访问进行管理,但是,这种所有权链接是否可以跨数据库,可以通过“跨数据库所有权链接”复选框来进行设置。

(5) “服务器属性”对话框的“连接”选项页如图 1-31 所示,在该选项页中可以设置与连接服务器相关的选项和参数。

“最大并发连接数(=0 无限制)”数值框用于设置当前服务器允许的最大并发连接数。并发连接数是同时访问服务器的客户端数量,这种限制受到技术和商业两方面的限制。其中,技术上的限制可以在这里设置,商业上的限制是通过合同或协议来确定的。将该选项设置为 0,表示从技术上不对并发连接数进行限制,理论上允许有无数多的客户端同时访问服务器。



图 1-31 “服务器属性”对话框的“连接”选项页

在 Microsoft SQL Server 系统中,查询语句执行时间的长短是通过查询调控器进行限定的。如果在“使用查询调控器防止查询长时间运行”下面的文本框中指定一个正数,那么查询调控器将不允许查询语句的执行时间超过这个设定值;如果指定为 0,则表示不限制查询语句的执行时间。另外,可以通过设置“默认连接选项”中的列表清单来控制查询语句的执行行为。

如果要设置与远程服务器连接有关的操作,可以选中“允许远程连接到此服务器”复选框,设置“远程查询超时值(秒,0=无超时)”文本框和“需要将分布式事务用于服务器到服务器的通信”复选框。

(6) “服务器属性”对话框的“数据库设置”选项页如图 1-32 所示,在该选项页中,可以设置与创建索引、执行备份和还原等操作相关的选项。

(7) “服务器属性”对话框的“高级”选项页如图 1-33 所示,在该选项页中,可以设置有关服务器的并行操作行为、网络行为等。



图 1-32 “服务器属性”对话框的“数据库设置”选项页

<b>并行</b>	
并行的开闭值	5
查询等待值	-1
锁	0
最大并行度	0
<b>网络</b>	
网络数据包大小	4096
远程登录超时值	20
<b>杂项</b>	
两位年份截止	2049
默认全文语言	2052
默认语言	Simplified Chinese
自动扫描存储过程	False
游标阈值	-1
允许触发器触发其他触发器	True
最大文本复制大小	65536

图 1-33 “服务器属性”对话框的“高级”选项页

(8) “服务器属性”对话框的“权限”选项页如图 1-34 所示, 在该选项页中, 可以设置和查看当前 SQL Server 实例中登录名或角色的权限信息。



图 1-34 “服务器属性”对话框的“权限”选项页

至此, 完成服务器的配置。



## 1.4 SQL Server 2008 常用的管理工具

Microsoft SQL Server 2008 系统提供了大量的管理工具,通过这些管理工具,可以对系统快速、高效地管理。这些管理工具主要包括:SQL Server Management Studio、Business Intelligence Development Studio、SQL Server Profiler、SQL Server Configuration Manager、Database Engine Tuning Advisor 以及大量的命令行和实用工具。本节将介绍这些工具的主要作用及其特点。

### 1.4.1 SQL Server Management Studio

Microsoft SQL Server Management Studio 是 Microsoft SQL Server 2008 提供的一种新的集成环境。Microsoft SQL Server 2008 将服务器管理和业务对象创建合并到以下两种集成环境中:SQL Server Management Studio 和 Business Intelligence Development Studio。这两种环境使用解决方案和项目来进行管理与组织,同时还提供了完全集成的源代码管理功能,能够与 Visual Studio 2008 集成。

SQL Server Management Studio 是一个集成环境,用于访问、配置、控制、管理和开发 SQL Server 的所有工作。实际上,SQL Server Management Studio 组合了大量的图形工具和丰富的脚本编辑器,大大方便了技术人员和数据库管理员对 SQL Server 系统的各种访问,它是 SQL Server 2008 中最重要的管理工具组件。SQL Server Management Studio 将 SQL Server 2000 的查询分析器和服务管理器的各种功能组合到一个单一环境中。此外,SQL Server Management Studio 还提供了一种新环境,用于管理分析服务(Analysis Services)、集成服务(Integration Services)、报表服务(Reporting Services)和 XQuery。此环境为开发者提供了一个熟悉的体验环境,为数据库管理人员提供了一个单一的实用工具,使用户能够通过易用的图形工具和丰富的脚本完成任务。

SQL Server 管理平台不仅能够配置系统环境和管理 SQL Server,而且由于它能够以层叠列表的形式来显示所有的 SQL Server 对象,因而,所有 SQL Server 对象的建立与管理都可以通过它来完成。通过 SQL Server Management Studio 可以完成的操作有:管理 SQL Server 服务器;建立与管理数据库;建立与管理表、视图、存储过程、触发程序、角色、规则、默认值等数据库对象以及用户定义的数据类型;备份数据库和事务日志、恢复数据库;复制数据库;设置任务调度;设置报警;提供跨服务器的拖放控制操作;管理用户帐户;建立 T-SQL 命令语句。

要打开 Microsoft SQL Server 2008 SQL Server Management Studio,可以通过“开始”菜单,选择 Microsoft SQL Server 2008 程序组中的“SQL Server Management Studio”。

要使用 SQL Server Management Studio,首先必须在对话框中注册。在“服务器类型”、“服务器名称”、“身份验证”选项中分别输入或选择正确的信息(默认情况下不用选择,因为在安装时已经设置完毕),然后单击“连接”按钮即可登录到 SQL Server Management Studio,如图 1-35 所示。



图 1-35 SQL Server Management Studio 主界面

SQL Server Management Studio 的工具组件包括: 已注册的服务器、对象资源管理器、解决方案资源管理器、模板资源管理器、摘要页。如果要显示某个工具, 需要选择“视图”下拉菜单中相应的工具名称即可。

“查询分析器”是以前版本中 Query Analyzer 工具的替代物, 使用“查询分析器”可以编写和执行 T-SQL 语句, 并且可以迅速查看这些语句的执行结果, 以便分析和处理数据库中的数据。与 Query Analyzer 工具总是工作在连接模式下不同的是, “查询分析器”既可以工作在连接模式下, 也可以工作在断开模式下。另外, “查询分析器”还支持彩色代码关键字、可视化地显示语法错误、允许开发人员运行和诊断代码等功能。这是一个非常实用的工具, 在 SQL Server Management Studio 工具栏中, 单击工具栏左侧的“新建查询”按钮即可打开查询分析器, 如图 1-36 所示。可以在其中输入要执行的 T-SQL 语句, 然后单击“执行”按钮, 或按 Ctrl+E 组合键执行此 T-SQL 语句, 查询结果将显示在结果窗口中。



图 1-36 SQL Server Management Studio 查询分析器界面

#### 1.4.2 Business Intelligence Development Studio

SQL Server 2008 商业智能开发平台(SQL Server 2008 Business Intelligence Development Studio)是一个集成开发环境, 用于开发商业智能构造(如多维数据集、数据源、报告和 Integration Services 软件包), 如图 1-37 所示。SQL Server 2008 商业智能开发平台包含了一些项目模板, 这些模板可供开发特定构造的上下文。

在商业智能开发平台中开发项目时，可以将其作为某个解决方案的一部分进行开发，而该解决方案独立于具体的服务器。例如，可以在同一个解决方案中包括 Analysis Services 项目、Integration Services 项目和 Reporting Services 项目。在开发过程中，可以将对象部署到测试服务器中进行测试，然后将项目的输出结果部署到一个或多个临时服务器或生产服务器上。



图 1-37 SQL Server 2008 商业智能开发平台界面

SQL Server 2008 商业智能开发平台可用于开发商业智能应用程序。如果要实现使用 SQL Server 数据库服务的解决方案，或者要管理并使用 SQL Server、Analysis Services、Integration Services 或 Reporting Services 的现有解决方案，则应当使用 SQL Server Management Studio；如果要开发并使用 Analysis Services、Integration Services 或 Reporting Services 的方案，则应当使用 SQL Server 2008 商业智能开发平台。

### 1.4.3 SQL Server Profiler

SQL Server 分析器(SQL Server Profiler)是一个图形化的管理工具，用于监督、记录和检查 SQL Server 2008 数据库的使用情况。对于系统管理员来说，它是一个连续、实时地捕捉用户活动情况的间谍。

可以通过多种方法来启动 SQL Server Profiler，以支持在各种情况下收集跟踪输出。例如，可以通过“开始”菜单启动 SQL Server Profiler。SQL Server Profiler 启动以后，选择“文件”|“新建跟踪”命令，打开如图 1-38 所示的“跟踪属性”窗口。

在“常规”选项卡中，可以设置跟踪名称和跟踪提供程序名称、类型，所使用的模板，保存的位置，以及是否启用跟踪停止时间等。

在“事件选择”选项卡中，可以设置需要跟踪的事件和事件列，如图 1-39 所示。





图 1-38 SQL Server Profiler 的“常规”选项卡



图 1-39 SQL Server Profiler 的“事件选择”选项卡

SQL Server Profiler 是用于捕获来自服务器的 SQL Server 2008 事件的工具，这些事件保存在一个跟踪文件中，可以在以后对该文件进行分析，也可以在试图诊断某个问题时，用它来重播某一系列的步骤。SQL Server Profiler 可以支持如下多种活动：

- 逐步分析有问题的查询，以便找到问题的原因。
- 查找并诊断执行速度慢的查询。
- 捕获导致某个问题的一系列 T-SQL 语句，然后利用所保存的跟踪，在某台测试服务器上复制此问题，接着，在该测试服务器上诊断问题。
- 监视 SQL Server 的性能以便优化工作负荷。
- 使性能计数器与诊断问题关联。

SQL Server Profiler 还支持对 SQL Server 实例上执行的操作进行审核。审核将记录与安全相关的操作，方便安全管理员以后复查。

#### 1.4.4 SQL Server Configuration Manager

SQL Server Configuration Manager(SQL Server 配置管理器)用于管理与 SQL Server 相关联的服务、配置 SQL Server 使用的网络协议以及从 SQL Server 客户端计算机管理网络连接配置。可以通过“开始”菜单来启动 SQL Server Configuration Manager，如图 1-40 所示。



图 1-40 SQL Server Configuration Manager 的界面

SQL Server 配置管理器是一个 Microsoft 管理控制台管理单元，它集成了以下工具的功能：服务器网络实用工具、客户端网络实用工具和服务管理器。通过设置“控制面板”|“管理工具”|“计算机管理”组件，也可以实现对 SQL Server Configuration Manager 的操作，“计算机管理”窗口如图 1-41 所示。



图 1-41 “计算机管理”窗口

### 1.4.5 Database Engine Tuning Advisor

Database Engine Tuning Advisor(数据库引擎优化顾问)工具可以帮助用户分析工作负荷、提出创建高效率索引的建议等。借助于数据库引擎优化顾问，用户不必详细地了解数据库的结构，就可以选择和创建最佳的索引、索引视图、分区等。Database Engine Tuning Advisor 的主界面如图 1-42 所示。

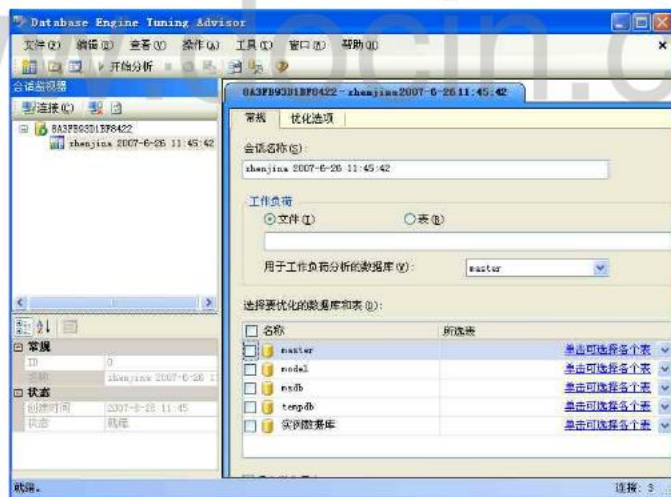


图 1-42 “Database Engine Tuning Advisor” 界面



在 SQL Server 2008 系统中, 使用数据库引擎优化顾问工具可以执行如下操作:

- 通过使用查询优化器分析工作负荷中的查询, 推荐数据库的最佳索引组合。
- 为工作负荷中引用的数据库推荐对齐分区和非对齐分区。
- 推荐工作负荷中引用的数据库的索引视图。
- 分析所建议的更改将产生的影响, 包括索引的使用、查询在工作负荷中的性能等。
- 推荐为执行一个小型的问题查询集而对数据库进行优化的方法。
- 允许通过指定磁盘空间约束等选项对推荐进行自定义。
- 提供对所给工作负荷的建议执行效果的汇总报告。

#### 1.4.6 实用工具

Microsoft SQL Server 2008 系统不仅提供了大量的图形化工具, 而且还提供了大量的命令行实用工具。通过这些命令, 可以与 Microsoft SQL Server 2008 进行交互, 但不能在图形界面下运行, 只能在 Windows 命令提示符下输入命令及参数执行(即相当于 DOS 命令)。这些命令行实用工具包括: bcp、dta、dtexec、dtutil、nscontrol、osql、rs、rsconfig、rskeymgmt、sac、sqlcmd、sqlmaint、sqlservr、sqlwb、tablediff 等, 这些工具的功能如下。

- bcp 实用工具可以在 Microsoft SQL Server 2008 实例和用户指定格式的数据文件之间进行数据复制。
- dta 实用工具是数据库引擎优化顾问的命令提示符版本。通过该工具, 用户可以在应用程序和脚本中使用数据库引擎优化顾问功能, 从而扩大了数据库引擎优化顾问的作用范围。
- dtexec 实用工具用于配置和执行 Microsoft SQL Server 2008 Integration Services (SSIS) 包。使用 dtexec 可以访问所有 SSIS 包的配置信息和执行功能, 这些信息包括连接、属性、变量、日志、进度指示等。
- dtutil 实用工具主要用于管理 SSIS 包, 这些管理操作包括验证包的存在性以及包进行复制、移动、删除等操作。
- nscontrol 实用工具与 Microsoft SQL Server 2008 Notification Services 服务有关, 用于管理、部署、配置、监视和控制通知服务, 并提供了创建、删除、使能、修复和注册等与通知服务相关的命令。
- osql 实用工具用来输入和执行 T-SQL 语句、系统过程、脚本文件等。该工具通过 ODBC 与服务器进行通信, 实际上, 在 Microsoft SQL Server 2008 系统中, sqlcmd 实用工具可以代替 osql 实用工具。
- rs 实用工具与 Microsoft SQL Server 2008 Reporting Services 服务有关, 用于管理和运行报表服务器的脚本。
- rsconfig 实用工具也是与报表服务相关的工具, 用来对报表服务连接进行管理。
- rskeymgmt 实用工具也是与报表服务相关的工具, 用来提取、还原、创建、删除对称密钥。

- sac 实用工具与 Microsoft SQL Server 2008 外围应用设置相关, 可用来导入、导出这些外围应用设置, 方便了多台计算机上的外围应用设置。
- sqlcmd 实用工具可以在命令提示符下输入 T-SQL 语句、系统过程和脚本文件。实际上, 该工具是作为 osql 实用工具和 isql 实用工具的替代工具而新增的, 它通过 OLE DB 与服务器进行通信。
- sqlmaint 实用工具可以执行一组指定的数据库维护操作, 这些操作包括 DBCC 检查、数据库备份、事务日志备份、更新统计信息、重建索引并且生成报表, 以及把这些报表发送到指定的文件或电子邮件帐户。
- sqlservr 实用工具的作用是在命令提示符下启动、停止、暂停、继续 Microsoft SQL Server 的实例。
- sqlwb 实用工具可以在命令提示符下打开 SQL Server Management Studio, 并且可以与服务器建立连接, 打开查询、脚本、文件、项目、解决方案等。
- tablediff 实用工具用于比较两个表中的数据是否一致, 对于排除复制过程中出现的故障非常有用。

## 1.5 习 题

1. 简述 SQL Server 2008 系统中主要数据库对象的特点。
2. SQL Server 2008 数据库管理系统产品分为哪几个版本, 它们各有什么特点?
3. SQL Server 2008 包含哪些组件, 其功能各是什么?
4. 安装 SQL Server 2008 之前应该做什么准备工作?
5. SQL Server 2008 支持哪两种身份验证?
6. 如何注册和启动 SQL Server 服务器?
7. 熟悉查询分析器的功能与使用。

www.docin.com

## 第3章 表的建立与维护

数据库创建之后，数据库中使用最频繁的就是表。表是存储数据的地方，是一种结构化的文件，是一种重要的数据库对象。我们可以通过创建不同的表来存储不同的数据，把表管理好也就管理好了数据库。其他数据，如视图、索引等，都是依附于表对象而存在的。

本章主要介绍 SQL Server 2008 数据库中表的创建及维护，以及基于表的各种操作(创建、修改、删除等)、数据完整性以及表的索引和视图等内容。

**本章的学习目标：**

- 掌握用 SSMS 和 T-SQL 创建表
- 熟悉各种列数据类型
- 掌握列的各种属性
- 掌握用 SSMS 和 T-SQL 修改、删除表
- 熟悉保持数据完整性的各种约束
- 掌握根据完整性规则的要求使用 SSMS 和 T-SQL 设置表的主键、外键和约束等

### 3.1 表 概 述

在使用数据库的过程中，接触最多的莫过于表。表是存储数据的地方，是一种结构化的文件，可以用来存储一些特定数据类型的数据，是数据库中最重要的部分。

#### 3.1.1 什么是表

在关系数据库中，每一个关系都表现为一张表。表是用来存储数据和操作数据的逻辑结构，关系数据库中的所有数据都表现为表的形式，由列和行组成，如图 3-1 所示。关系数据库由表、具体查询等对象组成，而查询等对象又是通过表来呈现的。在使用数据库时，绝大多数时间都是在与表打交道。表是数据库的基础，只有创建了表才能存储数据库记录信息。

EmployeeID	NationalIDNumber	ContactID	LoginID	ManagerID	Title
1	14417087	1209	adventure-work...	16	Production Tech...
2	253022076	1000	adventure-work...	6	Marketing Assist...
3	509647174	1002	adventure-work...	11	Engineering Man...
4	112457891	1290	adventure-work...	3	Senior Tool Desi...
5	401605026	1009	adventure-work...	253	Tool Designer
6	24756624	1020	adventure-work...	109	Marketing Manager
7	309750752	1070	adventure-work...	21	Production Sup...
8	696627818	1071	adventure-work...	185	Production Tech...
9	495256006	1005	adventure-work...	3	Design Engineer
10	912846026	1076	adventure-work...	185	Production Tech...
11	998220632	1006	adventure-work...	3	Design Engineer
12	248797967	1001	adventure-work...	109	Vice President of...
13	844073625	1072	adventure-work...	185	Production Tech...
14	233069930	1067	adventure-work...	21	Production Sup...
15	132674023	1073	adventure-work...	185	Production Tech...
16	440466106	1068	adventure-work...	21	Production Sup...
17	505209017	1074	adventure-work...	185	Production Tech...
18	494170342	1069	adventure-work...	21	Production Sup...
19	9839517	1075	adventure-work...	185	Production Tech...
20	443668905	1129	adventure-work...	175	Production Tech...
21	277173473	1231	adventure-work...	149	Production Cont...
22	839460180	1170	adventure-work...	197	Production Tech...
23	687689941	1175	adventure-work...	197	Production Tech...

图 3-1 表

在 Microsoft SQL Server 2008 中, 表有不同的类型, 每种类型的表都有其自身的作用和特点。具体可把表分为 4 种类型, 即普通表、分区表、临时表和系统表。

- 普通表又称标准表(基表或表), 就是通常被提到的数据库中存储数据和使用的表, 也是最重要、最基本的表。其他类型的表都是有特殊用途的表, 它们往往是在特殊应用环境下, 为了提高系统的使用效率而派生出来的表。
- 分区表是将数据水平划分为多个单元的表, 这些单元可以分散到数据库中的多个文件组中, 实现对单元中数据的并行访问。如果表中的数据量非常大, 并且这些数据经常被不同的使用方式来访问, 就需要建立分区表。分区表的特点在于可方便地管理大型表, 提高对表中数据的使用效率。
- 临时表就是临时创建的、不能永久生存的表。临时表又可分为本地临时表和全局临时表。临时表被创建之后, 可以一直存储到 SQL Server 实例断开连接为止。本地临时表只对创建者是可见的, 全局临时表则在创建之后对所有用户和连接都是可见的。
- 系统表与基表的主要区别在于, 系统表存储了有关 SQL Server 服务器的配置、数据库设置、用户和表对象的描述等系统信息。通常, 只能由 DBA 来使用系统表。

表是关系模型中表示实体的方式, 是用来组织和存储数据、具有行列结构的数据库对象。表是由定义的列数和可变的行数组成的逻辑结构。它是一个二维结构, 每一列称为一个字段, 每列的标题称为列名, 它是一组相同数据类型的值。在表中不必对行进行排序。要对结果集进行排序, 需要在相应的 SQL 语句中显示指定排序。在每个列和行的相交处是一个称为值的特定数据项。

表中的一行包含一个对象、事件或关系的数据。表中的所有行包含了类似的对象、事件或关系的数据, 表可以包含几百或者几千行, 表中的行是没有特定顺序的; 表中的一列包含关于表的每一行的一类数据, 每列都有一个列名, 每列都有一个数据类型, 最重要的数据类型是文本、数据和日期, 在表中一些列能接受 NULL 值, 其他的列则不接受 NULL 值。每列

在表中都有一个位置。

行和列相交的地方称为一个单元，它是表的最小部分，不能将它拆分成更小的部分。理论上一个单元只能包含单个数据，即单个信息单元。而实际上，有时一个单元可以包含几条信息。在数据库表中，一个单元可以包含以下内容中的一个：

- 一个单词
- 一个字母
- 一个数字
- 一个日期数据
- 一个 NULL 值，它表示该单元中没有数据

一个列是一个单元集合，这些单元有相同的数据类型，并且表现了相同类型的信息。一个行也是一个单元集合，它们一起表现了同一对象的信息。

在设计表时，目标是使用最少的表、每个表中包含最少的列来达到设计要求。合理的表结构，可以提高整个数据库的数据查询效率。为了设计出高质量的存储数据的表，在设计表时，应该考虑如下几点因素。

- 表需要存储的数据对象，并绘制相应的 E-R 图。

E-R 图，又称实体联系图。它是描述数据库中所有实体以及实体之间关系的图形，是辅助设计关系模型的工具。实际上，表就是关系模型，也应该对应模型中的实体，是存储数据的对象。在设计表时，应该考虑包含多少个表，每个表需要包含哪些数据，表与表之间是否存在关系。

- 表中需要包含的列，每一列的数据类型、精度。

在创建表时，确定表中的内容。每一个表都包含了多个列，每一列都有一个数据类型，数字数据类型的列还需要确定列的精度。通过充分分析，使表中列的数量尽可能地少。

- 表中哪些列允许空值或不允许空值。

在创建表时，列允许空值，表示该列可以不包含任何数据，如果允许列包含空值，表示可以不为该列输入具体的数据；反之，必须为该列提供具体数据。空值列的数量应尽可能少。

- 表中是否设置主键，在何处设置主键。

在创建表时，主键是唯一确定每一行数据的方式，是一种数据完整性对象。主键通常是一个列，也可能是多个列的组合。一个表中最多只能有一个主键，一般情况下，应该为每个表都指定主键，用来确定行数据的唯一性。

- 表中是否设置约束、默认值等，以及在何处设置。

在创建表时，约束、默认值等都是数据完整性对象，用来确保表中的数据质量。对表中数据的查询操作，只能在满足定义的约束、默认值、规则等条件下，才能执行成功。

- 表中是否设置外键以及在何处设置。

在 E-R 图中，需要绘制出实体之间的关系。在设计表时，实体之间的关系需要借助主键-外键对应来实现。因此，该因素也是确保 E-R 图完整实施的一个重要方面。



- 表中是否设置索引、在何处设置以及设置什么样的索引。

索引同表一样,也是一种数据库对象,是加快对表中数据检索的手段,是提高数据库使用效率的一种重要途径。在哪些列上设置或不设置索引,是设置聚集索引、非聚集索引,还是全文索引等。对了解到的因素要认真分析,达到更高的质量要求(索引的内容,将在第六章进行详细的讲解)。

### 3.1.2 表的数据类型

如表 3-1 所示列出了 SQL Server 2008 系统中常用的数据类型,不同的数据类型可以为表中的每个列限定取值范围,实现数据的域完整性控制。

**注意:**

在数据类型中,固定长度的数据类型比相应可变长度类型处理速度要快,如: char(n)比 varchar(n)处理速度快。

表 3-1 SQL Server 中常用的数据类型

种 类		数 据 类 型
数字	整数	int, bigint, smallint, tinyint
	精确数值	decimal, numeric
	近似数值	float, real
	货币	money, smallmoney
日期和时间		datetime, smalldatetime
字符	Non-Unicode	char, varchar, varchar(max), text
	Unicode	nchar, nvarchar, nvarchar(max), ntext
二进制		binary, varbinary, varbinary(max)
图像		image
全局标识符		uniqueidentifier
XML		xml
特殊		bit, cursor, timestamp, sysname, table, sql_variant

### 3.1.3 创建表

创建表可以有两种方法来实现:一种是通过图形界面的方法(即使用 SQL Server Management Studio)创建,另一种是通过 Transact-SQL 语句进行创建。下面就对这两种方法分别进行介绍。

#### 1. 使用 SQL Server Management Studio 创建表

(1) 打开 SQL Server Management Studio,在“对象资源管理器”窗格中,展开服务器,展开“数据库”,然后展开“StuInfo”数据库。

(2) 右击“表”节点,从弹出的快捷菜单中选择“新建表”命令,如图 3-2 所示。



图 3-2 “新建表”命令

(3) 打开表设计器窗口，根据需要创建表结构，创建好之后，单击工具栏上的“保存”按钮。在弹出的“选择名称”对话框中，输入表名“student”，如图 3-3 所示。

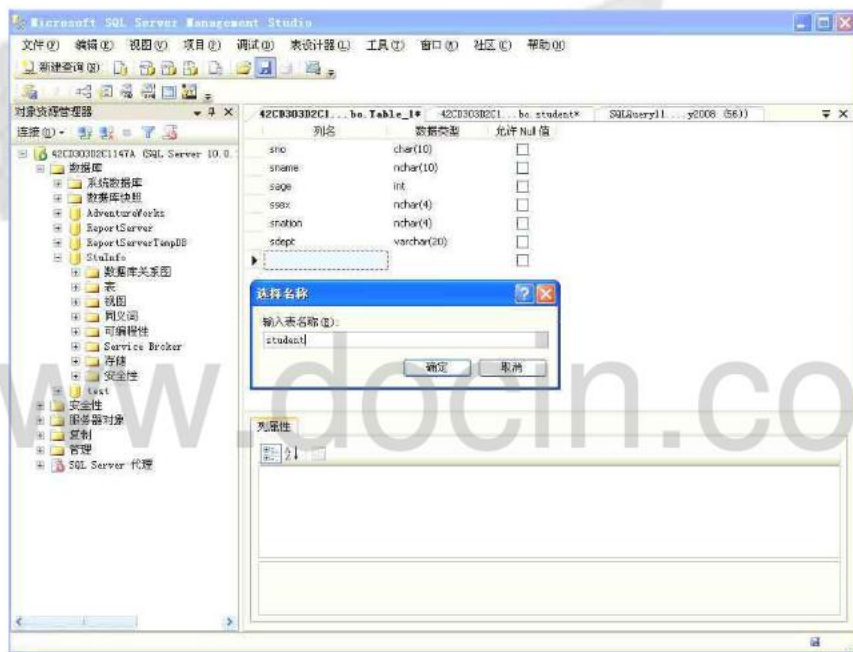


图 3-3 创建表并保存

(4) 关闭当前设计器窗口，完成表的创建。在新创建的 student 表中录入相应数据即可。如图 3-4 所示。

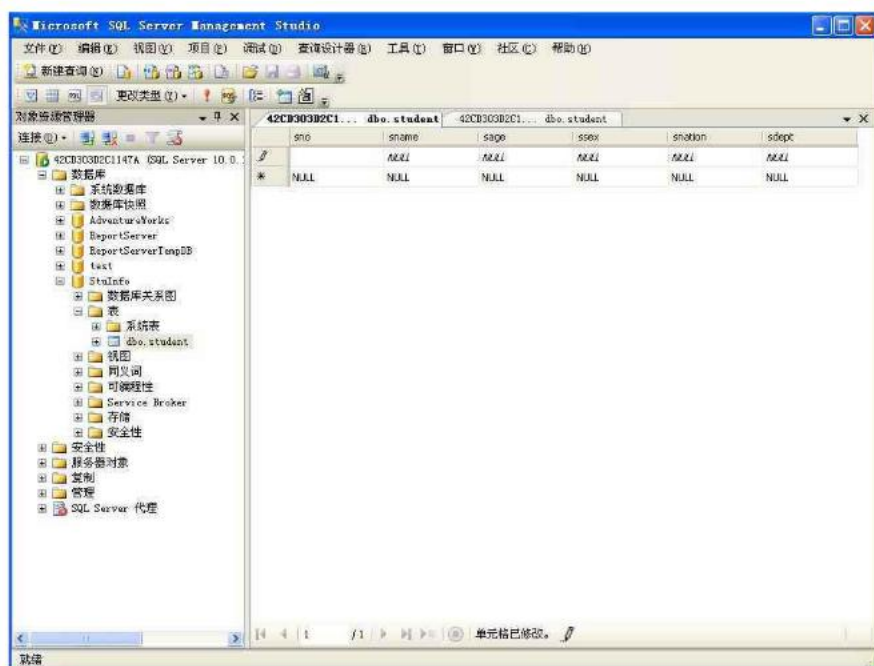


图 3-4 在 student 表中录入数据

## 2. 使用 T-SQL 语句创建表

使用 CREATE TABLE 命令创建表非常灵活，它允许对表设置几种不同的选项，包括表名、存放位置和列的属性等。其完整的语法格式如下：

```
CREATE TABLE [database_name].[owner].[owner].table_name
(
    {<column_definition>|<table_constraint>
    |column_name AS computed_column_expression}[,...n]
)
[ON {filegroup|DEFAULT}]
[TEXTIMAGE_ON {filegroup|DEFAULT}]
<column_definition>::={column_name data_type}
[[DEFAULT constant_expression][IDENTITY[(seed,increment)]]]
[<column_constraint>][,...]
```

该语句中各参数的说明如下：

- database\_name 是要在其中创建表的数据库名称。如果没有指定数据库，database\_name 默认为当前数据库。
- owner 是新数据表的所有者名，若此项为空，则默认为新表的创建者在当前数据库中的用户名。
- table\_name 是新数据表的名称。

- `column_name` 是新数据表中的字段名。字段名在表内唯一。如果字段名中包含空格,则需要将字段名用方括号括起来。
- `ON{filegroup|DEFAULT}` 指定存储表的文件组。如果指定 `filegroup`, 则数据表将存放在指定的文件组中; 如果指定 `DEFAULT`, 则数据表将存储在默认文件组中。
- `data_type` 指定了字段的数据类型, 可以是系统数据类型或用户自定义的数据类型。
- `column_constraint` 是字段约束。
- `IDENTITY` 指定该字段为标识字段。`seed` 定义标识字段的起始值, 起始值是装入表的第一个记录时使用的值。`increment` 定义标识增量, 标识增量是指该字段值相对前一条记录标识字段的增量值。
- 使用 SSMS 图形化界面创建检查约束。

在“表设计器”中, 如: SC 表。右键单击某列(Grade 列), 从弹出的快捷菜单中选择“CHECK 约束”命令, 或者单击工具栏中的“管理 CHECK 约束”按钮, 打开“CHECK 约束”对话框, 单击“添加”按钮, 在“表达式”文本框中输入检查(CHECK)表达式“Grade between 0 and 100”, 然后进行其他选项的设置, 如图 3-5 所示。最后, 单击“关闭”按钮完成设置。

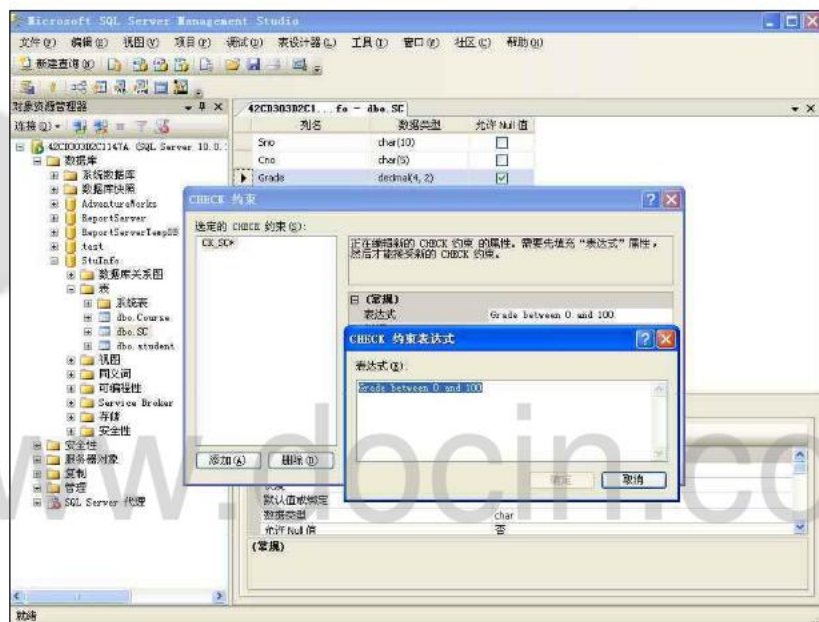


图 3-5 设置 CHECK 约束

- 用 Transact-SQL 语句创建检查约束

其语法形式如下:

```
CONSTRAINT constraint_name CHECK (logical_expression)
CHECK (logical_expression)
```

例【3-1】创建一个 SC 表, 定义 SCORE 的取值范围为 0 到 100 之间。

T-SQL 语句如下:

```
CREATE TABLE SC  
( SNO CHAR (5),  
  CNO CHAR (5),  
  SCORE NUMERIC(5,1) CHECK (SCORE>=0 AND SCORE <=100))
```

## 3.2 列数据类型

在计算机中,按照数据描述信息的含义,将数据分为不同的种类,称为数据类型。

数据类型在数据库中的作用非常关键。对数据类型的选择将影响应用数据库的应用程序的功能和性能。表创建完成后,涉及数据类型的结构更改将花费时间来完成,甚至一些对数据类型的更改会导致数据的丢失:如减少字段的长度,将列类型从 `nchar(10)` 更改为 `INT` 类型。因此,深入学习数据类型,理解他们的基本原理是非常重要的。在使用数据类型建立表时,也要格外谨慎。

在本节中,我们将学习 SQL Server 中的数据类型,它们中的一些可以定义为表中列的数据类型,而另一些则用在 T-SQL 程序中。

### 3.2.1 数据类型的分类

SQL Server 提供了两类数据类型:一类是用户定义数据类型,用户可以根据需要创建自己的数据类型,会对数据组织带来一定的好处;另一类是基本数据类型,是 SQL Server 系统提供的数据类型。本书只介绍基本数据类型。

SQL Server 的基本数据类型,可以分为数字数据类型、字符数据类型、日期和时间数据类型、二进制数据类型以及其他数据类型,如表 3-2 所示。用户使用时一般按照需求选择满足需要的最小的数据类型。

表 3-2 SQL Server 中常用的数据类型

种 类	数 据 类 型
数字数据类型	int, bigint, smallint, tinyint, decimal, numeric, float, real, money, smallmoney, bit
字符数据类型	char, varchar, nchar, nvarchar, text, ntext
日期和时间	datetime, smalldatetime, date, time, datetime2, datetimeoffset
二进制数据类型	binary, varbinary, image
其他数据类型	uniqueidentifier, xml, timestamp, sql_variant

### 3.2.2 数字数据类型

使用数字数据的类型被称为数字数据类型。这些数据类型的数字可用于各种数值运算。



数字数据类型包含两大类 11 种类型，具体分类如表 3-3 所示。

表 3-3 数字数据类型

数字数据类型		表示符号	用途
精确数字类型	整数数据类型	int、bigint、smallint、tinyint	存储数字数据，直接进行数据运算
	位数据类型	bit	用于一些条件逻辑判断
	货币数据类型	money、smallmoney	存储货币数据
近似数字类型	十进制小型	decimal、numeric	存储带固定精度和位数的小数
	浮点数据类型	float、real	存储范围非常大的数字，用于表示指定数的大致数据值

说明：

- 所有的数值都有精度，精度是指有效数字位数，小数位数是指小数点右边的数字个数。例如，数字 123456.789 的精度是 9，小数位数是 3；
- 数字数据类型的长度是存储此数值所占用的字节数。例如，int 数据类型最多可以存储 10 位数，用 4 个字节来存储；
- 在选择整数数据类型时，默认情况下应该考虑使用 int 数据类型，如果确认将要存储的数据可能很大或很小，那么可以考虑使用 bigint 型或 smallint 型。只有当将要存储的数据不超过 255 且都是正数时，才能使用 tinyint 数据类型；
- 位型数据(bit)，只存储 0、1 或 NULL，长度为 1 字节，常用于一些条件逻辑判断。特别的，由于在表示逻辑数据时，常用 TRUE 和 FALSE，因此，TRUE 和 FALSE 也可以存储到位型中，但这时需要按照字符格式存储 TRUE 和 FALSE 数据；
- 货币数据类型 money 和 smallmoney 的差别在于存储字节的大小和取值范围不同。它们的精度和小数位数是确定的。他们与其他数据类型的不同之处在于：
  - 它们表示货币数据值，因此可以在数字前面加上“\$”作为货币符号；
  - 它们的小数位数最多为 4 位，即可以表示货币单位的万分之一；
  - 当小数位数超过 4 位时，自动按四舍五入进行处理。
- 十进制小型(decimal 和 numeric)，精度高，而且没有误差。事实上这两种数据类型在功能上是等价的，只是名称不同而已。在 SQL Server 2008 系统中，把这两种数据类型作为完全相同的一种数据类型来对待。

decimal 数据类型由整数部分和小数部分构成，它的语法格式如下：

decimal(p, s)

其中 p 表示数字的精度，s 表示数字的小数位数，s 的默认值为 0。数据的存储长度随精度变化。

- 浮点数据类型(real 和 float)通常用于科学计算，数据用科学计数法表示。这种数据类型的优点是能够存储范围非常大的数字。但是浮点数据类型容易产生舍入误差。在对数据的精度要求并不是绝对严格时，可以考虑使用 real 或者 float 型数据。

### 3.2.3 字符数据类型

字符型数据用于存储字符串,其中包括字母、数字和其他符号。注意:术语字符串意味着一个或者多个组合在一起的字符。

T-SQL 提供的字符数据类型如表 3-4 所示。

表 3-4 字符数据类型

字符类型	表示符号	用法
ANSI 字符	char	存储指定长度的字符串
	varchar	存储可变长度字符
	text	存储大量非 Unicode 字符
Unicode 字符	nchar	用于存储 Unicode 字符
	nvarchar	与 varchar 类似,存储可变长度 Unicode 字符
	ntext	与 text 类似,存储大量 Unicode 字符数据

说明:

- char 类型存储字符数据时,每一个字符占一个字节大小。使用 char 类型时,应该给定数据的最大长度,定义格式为:char(n)
  - 如果没有指定 n 的大小,则默认值是 1,最长可以容纳 8000 个字符;
  - 如果实际数据的字符长度大于给定的最大长度,那么超过的字符将会被截断;若实际数据的字符长度小于给定的最大长度,则多余的字节被系统用空格填充。
- varchar 的使用方式与 char 基本相同。它们的区别在于:当列中字符长度基本一致时(如学号、姓名等),可以使用 char 类型;当数据长度相差较大时(如备注),使用 varchar 类型可以节省存储空间;
- 当数据有可能涉及到英语之外的其他语言时,应该使用 Unicode 字符编码。每一个 Unicode 字符需要两个字节存储,因此,Unicode 格式比 ANSI 格式有更大的字符集范围:ANSI 字符集仅有 256 个字符,Unicode 字符集则有 65536 个不同的字符。使用 Unicode 字符可以在表的一个列中同时出现中文、英文、法文等,而不会出现编码冲突。通常使用以下方式来表示 Unicode 字符:

N'Unicode 字符'

- nchar 和 nvarchar 分别用于存储固定长度和可变长度的 Unicode 字符数据;
- text 和 ntext,分别对应存储长度很大的 ANSI 字符和 Unicode 字符数据。微软公司建议,用户应该避免使用 text 和 ntext 数据类型,而使用 varchar 和 nvarchar 存储大文本数据。

### 3.2.4 日期时间数据类型

日期时间数据类型用于存储日期和时间信息。

在 SQL Server 2008 之前的版本中, 日期时间类型只包括 `datetime` 和 `smalldatetime` 两种类型。它们的差别在于其表示的日期和时间范围不同、时间精度也不同。其中最常用的是 `datetime` 型数据。

- `datetime` 型数据长度为 8 字节, 其中前 4 个字节用于存储日期, 后 4 个字节用于存储时间。由于历史原因的限制, `datetime` 支持的日期范围从 1753 年 1 月 1 日至 9999 年 12 月 31 日, 时间精确度是 3.33 毫秒;
- 由于存在各种各样的表示日期的习惯, 输入 `datetime` 数据的值是一件比较棘手的事情。如 '11/10/01' 对于不同的人有不同的理解方式。当将其作为 `datetime` 型数据输入时, SQL Server 将根据当前系统的语言设置进行转换。系统的语言由登录 SQL Server 时的默认语言决定, 也可以使用 SET DATEFORMAT 语句, 指定字符 D、M 和 Y 的组合。
  - 设置格式为: SET DATEFORMAT {format | @format\_var};
  - 参数 format 或 @format\_var 是日期的顺序。有效的形式包括 MDY、DMY、YMD、YDM、MYD 和 DYM, 其中 Y 表示年, M 表示月, D 表示日。默认情况下, 日期格式为 MDY;
  - 在 SQL Server 2008 中, 可以使用的时间格式是 HH:MI:SS.mmm。其中 HH 表示小时, MI 表示分钟, SS 表示秒, mmm 表示千分之一秒;
  - 存在一种标准输入格式, 不受 DATEFORMAT 设置的影响, 不会产生歧义。它的中间部分没有分隔符, 形式为: '[YY]YYMMDD[ HH:MI[:SS][.mmm]]'。如: '20111001'、'111001' 和 '20111001 20:50:30.988' 都是这种格式的具体应用。在输入 `datetime` 类型数据时, 我们推荐使用这种格式。

在 SQL Server 2008 系统中, 日期时间数据类型最大转变就是在 `datetime` 和 `smalldatetime` 两种类型的基础上又引入了 4 种日期时间数据类型, 分别为 `date`、`time`、`datetime2` 和 `datetimeoffset`。下面分别介绍:

- `date` 数据类型只存储日期型数据类型, 不存储时间数据, 取值范围从 0001-01-01 到 9999-12-31。引入 `date` 类型, 克服了 `datetime` 类型中既有日期又有时间的缺陷, 使得对日期的查询更加方便;
- `time` 数据类型与 `date` 数据类型类似, 如果只想存储时间数据而不需要存储日期部分, 就可以利用 `time` 数据类型, 取值范围从 00: 00: 00.0000000 到 23: 59: 59.9999999;
- `datetime2` 数据类型是一种日期时间混合的数据类型, 不过其时间部分秒数的小数部分可以保留不同位数的值, 比 `datetime` 数据类型的取值范围要广, 可以存储从公元元年 1 月 1 日到 9999 年 12 月 31 日的日期。用户可以根据自己的需要设置不同的参数来设定小数位数, 最高可以设定到小数点后七位(参数为 7), 也可以不要小数部分(参数为 0), 依此类推;
- `datetimeoffset` 数据类型用于存储与特定的日期和时区相关的日期和时间。这种数据类型的日期和时间存储为协调世界时(Coordinated Universal Time, UTC)的值, 然后, 根据与该值相关的时区, 定义要增加或减去的时间数。`datetimeoffset` 类型是由年、月、日、小时、分钟、秒和小数秒组成的时间戳结构。小数秒的最大小数位数为 7。

### 3.2.5 二进制数据类型

二进制数据类型用于表示位数据流,包括 binary(固定长度)、varbinary(可变长度)和 image 三种。

- binary 用于存储固定长度的二进制数据;
- varbinary 用于存储可变长度的二进制数据,如果存储的二进制大于 8000 字节,就必须使用 varbinary(max)数据类型;
- image 数据类型用于存储图像信息,在 SQL Server 2008 中,只有在数据的字节数超过了 8KB 的情况下,才使用 image 数据类型,其他情况应使用 varbinary(max)代替,其中 max 最大可以达到  $2^{31}-1$  字节;
- 当二进制数据存储到表中时,可以使用 SELECT 语句来检索。但检索结果是以 16 进制数据格式来显示。

### 3.2.6 其他数据类型

除了上述数据类型外,SQL Server 还提供了其他几类常用数据类型,使用这些数据类型可以完成特殊数据对象的定义、存储和使用:

- timestamp 时间戳数据类型与时间、日期无任何关系。timestamp 的值是二进制,表明数据库中的数据修改发生的相对顺序。实现 timestamp 数据类型最初是为了支持 SQL Server 恢复算法。每一个数据库都有一个时间戳计数器,当对该数据库中包含 timestamp 列的表执行插入或更新操作时,计数器值就会增加。这样,可以轻易地确定表中的某个数据行的值是否在上次读取后发生了更新:如果发生了更新,则该时间戳计数器的值也发生了变化。可以使用 @@DBTS 函数返回数据库的时间戳值;
- sql\_variant 类型:用于存储 SQL Server 2008 支持的各种数据类型(不包括 text、ntext、image、timestamp 和 sql\_variant 类型)的值。数据列类型为 sql\_variant 的列可能包含不同类型的行,一般在不能准确确定将要存储的数据类型时,使用这种数据类型。该数据类型可以用在列、变量、用户定义的函数等返回值中;
- uniqueidentifier 唯一标识符类型,是一个具有 16 字节的全局唯一性标志符,用来确保对象的唯一性。可以在定义列或者变量时使用该数据类型,主要目的是在合并复制和事务复制中确保表中数据行的唯一性。例如,在客户标识号列使用这种数据类型可以区别不同的客户。uniqueidentifier 数据类型的初始值可以通过 NEWID 函数获得;
- xml 类型:xml 数据类型可以用来保存整个 XML 文档,允许用户存储和处理 XML 数据。用户可以像使用 int 数据类型一样使用 xml 数据类型。另外,xml 数据类型还提供了一些高级功能,比如借助 Xquery 语法执行搜索。

### 3.2.7 数据类型的选择

通过以上介绍可以看到,不同的数据类型有其特定的用途。在实际使用中,应慎重选择数据类型,需要综合考虑如下因素:



- 数据的使用情况：数据是用于数学运算，表示日期或者时间还是存储文本信息？不同的使用情况决定了不同数据类型的选择；
- 数据的大小：选择的数据类型能否存放期望存储的最大值。例如，选择整型数据，要根据数据的取值决定使用 INT 型数据还是 BIGINT 型数据；选择字符型数据，需要考虑实际需要的最大字符长度；
- 正确的存储信息：例如，使用 integer 数据类型存储货币值，将造成小数部分丢失，这将导致不正确的值；
- 非英语字符：在 SQL Server 2008 系统中，如果某些列需要存储非英语字符(如中文字符)，建议最好使用 NCHAR、NVARCHAR 数据类型。

### 3.3 列的其他属性

#### 3.3.1 默认约束

默认约束是指用户在进行插入操作时，没有显示地为列提供数据，那么系统将把默认值赋给该列。默认值约束所提供的默认值可以为常量、函数、系统函数、空值等，表中的每一列只能定义一个默认约束，对于具有 IDENTITY 属性和 timestamp 数据类型的字段，不能使用默认约束，同时，定义的默认值长度不允许大于对应字段所允许的最大长度。

创建默认约束的具体操作如下：

打开设计表的对话框，单击要设置的字段，在列属性的“常规”选项区中，在“默认值或绑定”文本框中输入该字段的默认值，保存表的修改。例如，“性别”列只能为“男”或“女”，若设置其默认值为“男”，则当用户没有输入数据时，系统将自动添加“男”字符串，如图 3-6 所示。



图 3-6 创建默认约束

#### 3.3.2 空值约束

空值约束即是否允许该字段的值为 NULL，即空值。主键列不允许为空值，否则就失去了唯一标识的意义。

创建空值约束的操作如下:

打开“表”对话框,选中需要设置空值约束的字段,选中“允许空”复选框,然后保存表的修改即可,如图 3-7 所示。



图 3-7 空值约束

### 3.3.3 IDENTITY 的应用

identity 表示的是最近一次向具有 identity 属性(即自增列)的表插入数据时对应的自增列的值,是系统定义的全局变量。比如有个表 A,它的自增列是 id,当向 A 表中插入一行数据后,如果插入数据后自增列的值自动增加至 101,则通过 select @@identity 得到的值就是 101。使用 @@identity 的前提是在进行 insert 操作之后,执行 select @@identity 的时候连接没有关闭,否则得到的将是 NULL 值。

在发出 CREATE SEQUENCE 语句以创建新序列或对现有序列发出 ALTER SEQUENCE 语句时,可以更改序列的属性。例如,如果要创建一个名为 id\_values 的序列,其最小值为 0、最大值为 1000、使用每个 NEXT VALUE 表达式使值递增 2,并且在达到最大值时返回到其最小值,那么使用以下语句: CREATE SEQUENCE id\_values START WITH 0 INCREMENT BY 2 MAXVALUE 1000 CYCLE。

SQL SERVER 2008 中的标识值获取函数 IDENTITY(标识)列,也有很多人称之为自增列,在 SQL Server 2008 中,标识列通过 IDENTITY 来定义,下面是与获取最后插入记录的标识值有关的函数的一个示例说明。在 SQL Server 中,可以使用 SCOPE\_IDENTITY()、@@IDENTITY、IDENT\_CURRENT()来取得最后插入记录的值,它们的区别在于:SCOPE\_IDENTITY()返回插入到同一作用域中的 IDENTITY 列内的最后一个 IDENTITY 值。

创建表时指定标识列,标识列可用 IDENTITY 属性建立,因此在 SQL Server 中,又称标识列为具有 IDENTITY 属性的列或 IDENTITY 列。可使用函数 IDENT\_CURRENT,用法为: SELECT IDENT\_CURRENT('表名')。需要注意的是:当包含标识列的表刚刚创建,为经过任何插入操作时,使用 IDENT\_CURRENT 函数得到的值为标识列的种子值,这一点在开发数据库应用程序的时候尤其应该注意。

## 3.4 向表中添加数据

首先需要在需要添加数据的表上单击鼠标右键,从弹出的快捷菜单中选择“打开表”命令,出现如图 3-8 所示的“表”对话框,在其中可向打开的表添加数据。本节向各个表中添加的数据如表 3-5~表 3-7 所示。

学号	姓名	性别	出生日期	入学日期	院系名称	备注
20050101	张利	男	1986-2-3 0:00:00	2005-8-1 0:00:00	计算机系	
20050102	唐杰	男	1985-3-6 0:00:00	2005-8-1 0:00:00	计算机系	
20050103	王娟	女	1986-5-30 0:00:00	2005-8-1 0:00:00	计算机系	
20050201	陈胜利	男	1987-8-9 0:00:00	2005-8-1 0:00:00	企管系	
20050202	赵杉	女	1986-6-5 0:00:00	2005-8-1 0:00:00	企管系	
20050301	吕启明	男	1985-6-23 0:00:00	2005-8-1 0:00:00	国贸系	

图 3-8 “表”对话框

表 3-5 学生表

学 号	姓 名	性 别	出 生 日 期	入 学 日 期	院 系 名 称	备 注
20050101	张利	男	1986.02.03	2005.08.01	计算机系	
20050102	唐杰	男	1985.03.06	2005.08.01	计算机系	
20050103	王娟	女	1986.05.30	2005.08.01	计算机系	
20050201	陈胜利	男	1987.08.09	2005.08.01	企管系	
20050202	赵杉	女	1986.06.05	2005.08.01	企管系	
20050301	吕启明	男	1985.06.23	2005.08.01	国贸系	

表 3-6 课程表

课 程 号	课 程 名	学 分	备 注
01001	数据库原理及应用	4	
02001	市场营销	4	
01002	操作系统	4	
02002	消费心理学	3	

表 3-7 选课表

学 号	课 程 号	分 数
20050101	01001	85
20050101	01002	90
20050102	01002	82
20050201	02001	78
20050201	02002	80
20050301	02002	60

向表中添加的数据不符合列约束(检查约束)的规定时,例如,向选课表中的“分数”字段输入大于 100 或小于 0 的数值时,将出现如图 3-9 所示的错误信息。单击“确定”按钮,然后根据提示进行修改。单击错误的行,按 Esc 键取消该行,重新添加信息。



图 3-9 出错信息



## 3.5 查看表

在数据库中创建表之后,有时需要查看表的有关信息、表中存储的数据及表与其他数据库对象之间的依赖关系。

### 3.5.1 查看表中的有关信息

打开指定的数据库,在需要查看的表上单击鼠标右键,从弹出的快捷菜单中选择“属性”命令,打开“表属性”对话框,如图 3-10 所示。“常规”选项页中显示了该表格的定义,包括存储结构、当前的连接以及名称等属性,该选项页中显示的属性不能修改。

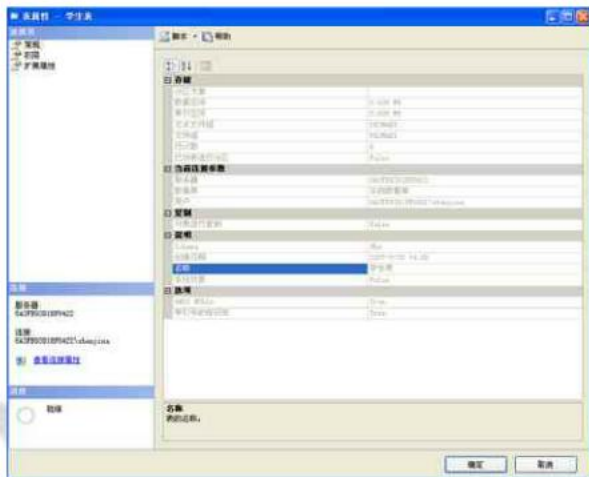


图 3-10 表属性

### 3.5.2 查看表中存储的数据

在表“课程表”上单击鼠标右键,从弹出的快捷菜单中选择“打开表”命令,显示“课程表”中的数据,如图 3-11 所示。

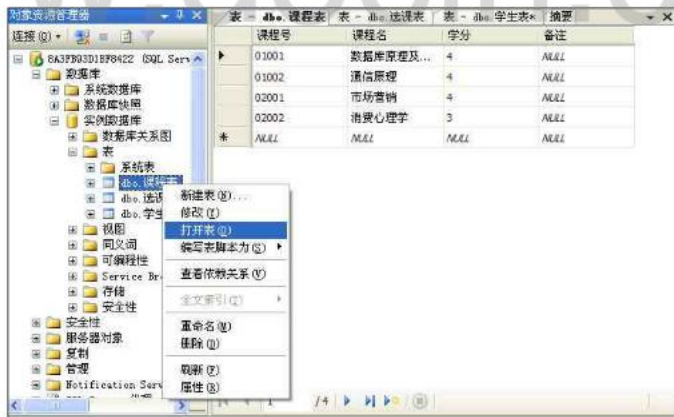


图 3-11 显示表中的数据



### 3.5.3 查看表与其他数据对象的依赖关系

在要查看的表上单击鼠标右键，从弹出的快捷菜单中选择“查看依赖关系”命令，打开“对象依赖关系”对话框，该对话框显示了该表依赖的其他数据对象和依赖于此表的依赖对象。例如，查看“选课表”的依赖对象，如图 3-12 所示，该对话框显示了选课表依赖于课程表与学生表。

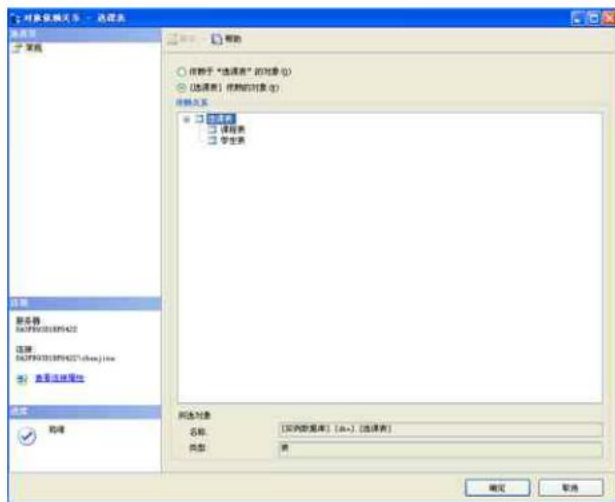


图 3-12 表的对象依赖关系

## 3.6 修 改 表

完成了表的创建及设计之后，如果发现表的名称、结构、字段属性和约束等相关设置不合理，可以对表进行修改。下面分别介绍如何使用 SQL Server Management Studio 以及使用 T-SQL 语句对表进行修改。

### 3.6.1 使用 SSMS 图形化界面修改表

在“对象资源管理器”窗口中，展开“数据库”节点，再展开具体的数据库，然后展开“表”节点，用右键单击要修改的表，从弹出的快捷菜单中选择“设计”命令，打开“表设计器”，即可对表的定义进行修改，方法同创建表。

### 3.6.2 使用 T-SQL 语句修改表

可以使用 ALTER TABLE 语句修改表定义，其语法格式如下：

```
ALTER TABLE 表名
(ALTER COLUMN 列名 列定义,
ADD 列名 数据类型 约束,
.....
DROP 列名,
```

```
.....  
ADD CONSTRAINT 约束名 约束,  
.....)
```

例【3-2】在 AdventureWorks 数据库中, 在 Sales 下创建名为 CustomerOrders 的表; 创建 OrderID、OrderDate、CustomerID、Notes 列, 并定义其数据类型分别为 int、datetime、int、nvarchar(200))和可空性(Notes 列为空), 指定 OrderID 列为标志列。

T-SQL 语句如下:

```
CREATE TABLE Sales.CustomerOrders  
( OrderID int identity(1,1),  
  OrderDate datetime,  
  CustomerID int,  
  Notes nvarchar(200) NULL );
```

例【3-3】基于例 3-2 定义的 Sales.CustomerOrders 表, 添加列和更改列的可空性。

```
-- 修改 Sales.CustomerOrders 表的定义  
-- 在表中添加名为SalesPersonID 的列, 并定义其数据类型为int, 指其不可空  
ALTER TABLE Sales.CustomerOrders  
ADD SalesPersonID int NOT NULL  
GO  
-- 修改 Sales.CustomerOrders 表的定义  
-- 修改 Notes 列, 并定义其数据类型为nvarchar(200), 指定其不可空  
ALTER TABLE Sales.CustomerOrders  
ALTER COLUMN Notes nvarchar(200) NOT NULL  
GO
```

### 3.6.3 使用 INSERT SELECT 语句

INSERT SELECT 语句可以将其他数据源中的行添加到数据库中现有的表当中。这种方法用于插入的数据是不确定(一般都多于一条)。具体方法是: 通过 SELECT 语句生成的结果集, 再结合 INSERT 语句, 就可以把结果集插入到指定的表中。

使用 INSERT SELECT 语句比使用多个单行的 INSERT 语句效率要高得多。使用 INSERT SELECT 语句时要注意以下原则: 必须保证接收新值的表中的列的数据类型与源表中相应列的数据类型一致; 插入新行的表是否存在于数据库中; 必须确定是否存在默认值, 或所有被忽略的值是否允许空值。

INSERT SELECT 语句的语法格式如下:

```
INSERT INTO table_name  
SELECT column_list  
FROM table_list  
WHERE search_condition
```

SELECT 查询语句用于标注输入表中的列, 通过 SELECT-FROM-WHERE 查询语句块生成结果集, 将其添加到 INSERT 语句指定的数据库的表中。

例【3-4】求每个学生的平均成绩，并按学号、姓名、平均成绩存入学生信息数据库。首先创建一个表：

```
CREATE TABLE AG(  
    学号 CHAR(8)PRIMARY KEY,  
    姓名 CHAR(8),  
    平均成绩 SMALLINT)  
GO;
```

再求每个学生的平均成绩并把其插入已创建的表中：

```
INSERT INTO AG(学号, 姓名, 平均成绩)  
SELECT SC.学号, MIN(姓名), AVG(成绩)  
FROM Student, SC  
WHERE Student.学号 = SC.学号  
GROUP BY SC.学号;
```

### 3.6.4 使用 SELECT INTO 语句

SELECT INTO 语句与 INSERT SELECT 语句不同的是：它可以将其他数据源中的任何查询结果或导入的数据都添加到数据库的新表中。这种方法用于从不同的数据源中找到不同的数据集，放在通过该语句一开始创建的临时表中，用该表进行查询比在多表或多数据库查询执行起来更加容易、方便。

使用 SELECT INTO 语句时要注意如下原则：必须保证 SELECT INTO 语句中指定创建的新表在数据库是唯一的；可以创建本地临时表或全局临时表。本地临时表的创建需要在表名前加一个#符号，全局临时表的创建需要在表名前加两个##符号。

SELECT INTO 语句的语法格式如下：

```
SELECT select_list  
INTO new_table  
FROM table_list  
WHERE search_condition
```

例【3-5】统计每个学生未通过课程的门数，将其保存到临时表#stuexam 表。显示系名、学号、姓名、未通过门数，并按系名排序。

T-SQL 语句如下：

```
SELECT MIN(系名), SC.学号, MIN(姓名), COUNT(*) 未通过门数  
INTO #stuexam  
FROM Student, SC  
WHERE Student.学号=SC.学号 AND 成绩 < 60  
GROUP BY SC.学号  
ORDER BY 系名;
```

## 3.7 删除表

删除表时，表的结构定义、数据、全文索引、约束等都将永久地从数据库中删除，原先

存放表及其索引的存储空间可用来存放其他表。如果要删除通过外键约束和主键约束相关的外键表和主键表,必须首先删除外键表;若要删除外键约束中引用的主键表而不删除外键表,则必须删除外键表的外键约束。

### 1. 使用 SSMS 图形化界面删除表

在“对象资源管理器”窗口中,展开“数据库”节点,再展开所选择的具体数据库,然后展开“表”节点,用右键单击要删除的表,从弹出的快捷菜单中选择“删除”命令,打开“删除对象”窗口,如图 3-13 所示。

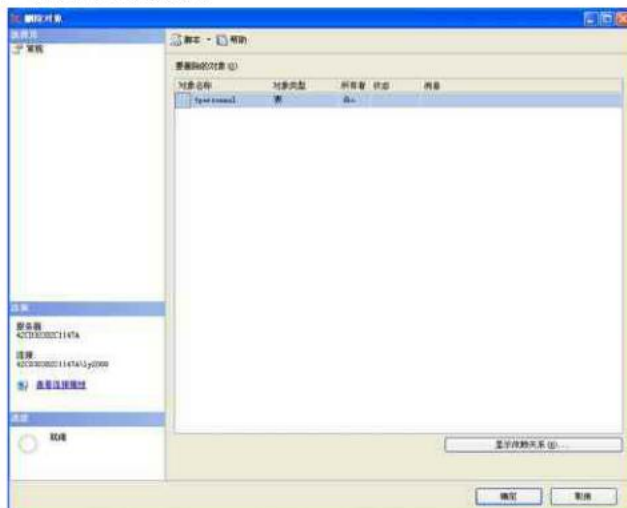


图 3-13 用 SSMS 图形化界面删除表

### 2. 使用 T-SQL 语句删除表

使用 DROP TABLE 语句删除表,如图 3-14 所示。其 T-SQL 语句格式如下:

DROP TABLE 表名

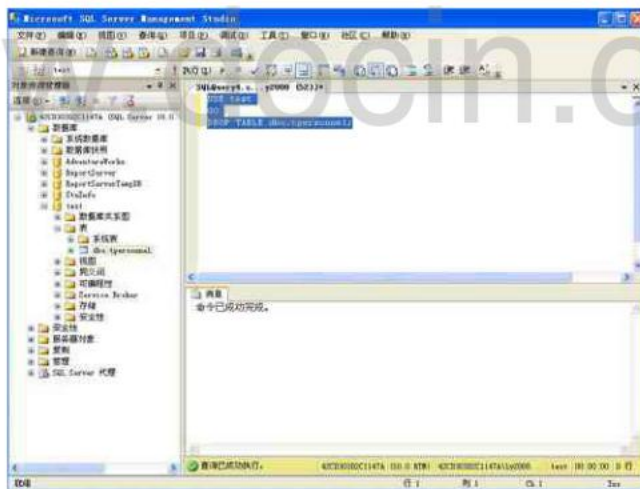


图 3-14 用 T-SQL 语句删除表



例【3-6】在 AdventureWorks 数据库中，删除 CustomerOrders 表，它的架构是 Sales。

```
USE AdventureWorks
```

```
GO
```

```
DROP TABLE Sales.CustomerOrders
```

### 3.8 习 题

1. 在一个表上可以定义\_\_\_\_\_个 CHECK 约束。
2. 创建表的语句是：\_\_\_\_\_。
3. 表和表之间的关系是通过\_\_\_\_\_实现的。
4. 删除表“employ”中的“employdate”列所使用的语句是什么？
5. 为表“employ”删除主键约束的语句是什么？



# 第5章 安全与权限

安全性问题对于绝大多数的数据库应用程序来说都是至关重要的。SQL Server 2008 在数据库平台的安全模型上有了显著增强，由于提供了更为精确和灵活的控制，数据安全更为严格。SQL Server 2008 旨在通过数据库加密、更加安全的默认设置、加强的密码政策和细化许可控制以及加强的安全模型等特性，为企业数据提供最高级别的安全性。本章将主要讲述数据库开发人员在开发安全的数据应用时应该掌握的基本安全与权限的概念。

**本章的学习目标：**

- 了解 SQL Server 2008 的安全策略
- 掌握 SQL Server 2008 的网络配置
- 掌握对 Microsoft SQL Server 实例、数据库访问的管理
- 掌握对实例、数据库、应用程序的角色管理
- 了解数据库架构及其作用
- 掌握对 SQL Server 实例、数据库、表和列访问的权限管理
- 掌握对可编程对象访问的权限管理

## 5.1 安全与权限的基础知识

### 5.1.1 SQL Server 2008 安全机制的总体策略

在 SQL Server 2008 中，数据的安全保护由 4 个层次构成，如图 5-1 所示。SQL Server 2008 主要对其中的 3 个层次提供安全控制。下面分别对每个层次进行介绍。

(1) 远程网络主机通过 Internet 访问 SQL Server 2008 服务器所在的网络，这由网络环境提供某种保护机制。

(2) 网络中的主机访问 SQL Server 2008 服务器，首先要求对 SQL Server 进行正确配置，其内容将在下一节中介绍；其次是要求拥有对 SQL Server 2008 实例的访问权——登录名，其内容将要在 5.2.1 小节中介绍。

(3) 访问 SQL Server 2008 数据库，这要求拥有对 SQL Server 2008 数据库的访问权——数据库用户，其内容将要在 5.2.2 小节中介绍。

(4) 访问 SQL Server 2008 数据库中的表和列，这要求拥有对表和列的访问权——权限，其内容将要在 5.5.2 小节中介绍。

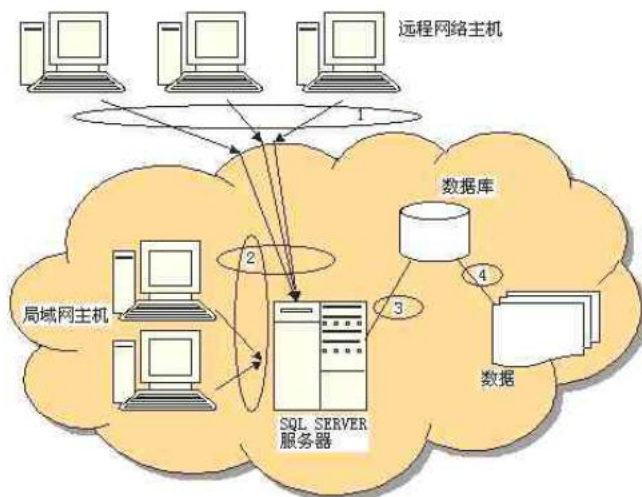


图 5-1 SQL Server 2008 的安全层次结构示意图

为了对登录和数据库用户进行管理，SQL Server 提供了角色的概念，5.3 节将介绍对固定服务器角色、数据库角色、应用程序角色的管理。

SQL Server 2008 实现了 ANSI 中有关架构的概念，一个数据库对象通过由 4 个命名部分所组成的结构来引用：<服务器>.<数据库>.<架构>.<对象>，其内容将在 5.4 节中介绍。

权限的管理不仅涉及数据库中的表和列，还涉及对 SQL Server 实例和数据库的访问、对可编程对象的访问，其内容将在 5.5 节中介绍。

### 5.1.2 网络安全基础

SQL Server 2008 是第一个基于 Microsoft Trustworthy Computing initiative 开发的 SQL Server 版本。Trustworthy Computing initiative 的宗旨之一就是 Secure by Default(默认安全)。在实现这一原则的过程中，SQL Server 2008 禁用了一些网络选项，以尽量保证 SQL Server 环境的安全性。

SQL Server 是一个设计用于在服务器上运行，能够接受远程用户和应用程序访问的数据库管理系统。在运行 SQL Server 的本地计算机上也可以对 SQL Server 进行本地访问。但实际应用中一般不这样做。因此，正确配置 SQL Server，以使其能够接受远程计算机的安全访问是非常重要的。

为了可以远程访问 SQL Server 实例，需要一种网络协议来建立到 SQL Server 服务器的连接。为了避免系统资源的浪费，只需要激活自己需要的网络连接协议即可。

在默认安装中，SQL Server 禁用了许多功能特性，以减少数据库系统被攻击的可能性。例如，SQL Server 2008 在默认情况下并不允许远程访问(企业版除外)，所以要用“SQL Server 外围应用配置器”工具来启用远程访问。

可以通过如下操作来配置远程访问，启用远程访问连接。

(1) 从“开始”菜单中选择“所有程序”|“Microsoft SQL Server 2008”| SQL Server Management Studio 命令，启动 SSMS，如图 5-2 所示。

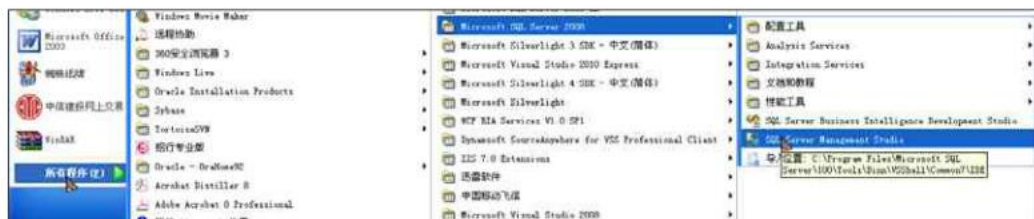


图 5-2 通过开始菜单启动 SSMS

- (2) 在“连接到服务器”对话框中，指定属性值，单击“连接”按钮，如图 5-3 所示。
- (3) 右击服务器节点，从弹出的快捷菜单中选择“方面”命令，如图 5-4 所示。



图 5-3 “连接到服务器”对话框



图 5-4 选择“方面”命令

- (4) 在打开的“查看方面”对话框中，选择“外围应用配置器”选项，如图 5-5 所示。



图 5-5 “查看方面”对话框



在该对话框中,可以设置“外围应用配置器”相关的选项,设置完成后,单击“确定”按钮即可。

数据库存储了重要的信息,应当对数据库服务器进行很好的网络安全保护,以防止未经授权的外部访问。当需要 SQL Server 能够通过 Internet 供用户或者应用程序访问时,应该保证网络环境提供了某种保护机制,例如,防火墙或者 IDS(入侵检测系统)。

除了通过 SSMS 进行设置之外,还可以通过存储过程 sp\_config 进行设置。

在“查询分析器”中执行: "exec sp\_config 'remote admin connections 1'"

提示已经从"0"修改为"1".即: "从本地链接,修改为允许远程链接。

## 5.2 管理用户

连接到 SQL Server 实例的时候,必须提供有效的认证信息。数据库引擎会执行两步有效性验证过程: 第一步,数据库引擎会检查用户是否提供了有效的、具备连接到 SQL Server 实例权限的登录名; 第二步,数据库引擎会检查登录名是否具备连接数据库访问许可。

SQL Server 2008 定义了人员、组或进程作为请求访问数据库资源的实体。实体可以在操作系统、服务器和数据库级进行指定,并且实体可以是单个实体或集合实体。例如,一个 SQL 登录名是在 SQL Server 实例级的实体,一个 Windows 组则是 Windows 级的集合实体。

### 5.2.1 管理对 SQL Server 实例的访问

针对 SQL Server 实例访问,SQL Server 2008 支持两种身份验证模式: Windows 身份验证模式和混合身份验证模式。

- 在 Windows 身份验证模式下,SQL Server 依靠操作系统来认证请求 SQL Server 实例的用户。由于已经通过了 Windows 的认证,因此用户不需要在连接字符串中提供任何认证信息。
- 在混合身份验证模式下,用户既可以使用 Windows 身份验证模式,也可以使用 SQL Server 身份验证模式来连接 SQL Server。在后一种情况下,SQL Server 根据现有的 SQL Server 登录名来验证用户的凭据。使用 SQL Server 身份验证需要用户在连接字符串中提供连接 SQL Server 的用户名和密码。

#### 1. 选择身份验证模式

可以通过如下步骤在 SQL Server Management Studio 中配置身份验证模式。

(1) 在“开始”菜单中选择“所有程序”|“Microsoft SQL Server 2008”|“SQL Server Management Studio”命令,打开“连接到服务器”对话框,如图 5-6 所示。



图 5-6 “连接到服务器”对话框

(2) 在“连接到服务器”对话框中，单击“连接”按钮，连接到服务器，如图 5-7 所示。

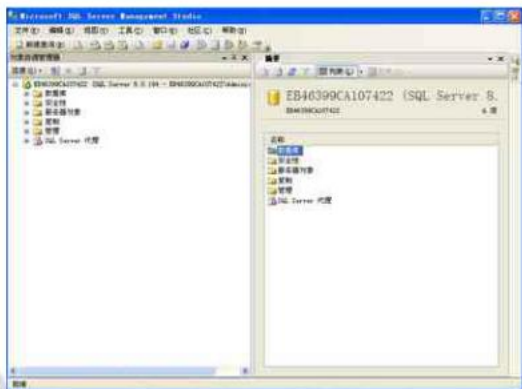


图 5-7 连接到服务器

(3) 在“对象资源管理器”中，在 SQL Server 实例名上单击鼠标右键，从弹出的快捷菜单中选择“属性”命令，打开“服务器属性”对话框，如图 5-8 所示。

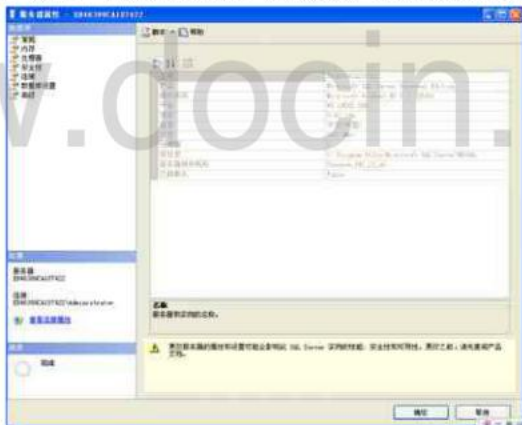


图 5-8 “服务器属性”对话框

(4) 在左边的“选择页”列表框中，选择“安全性”选项，打开“安全性”选项页，如图 5-9 所示。

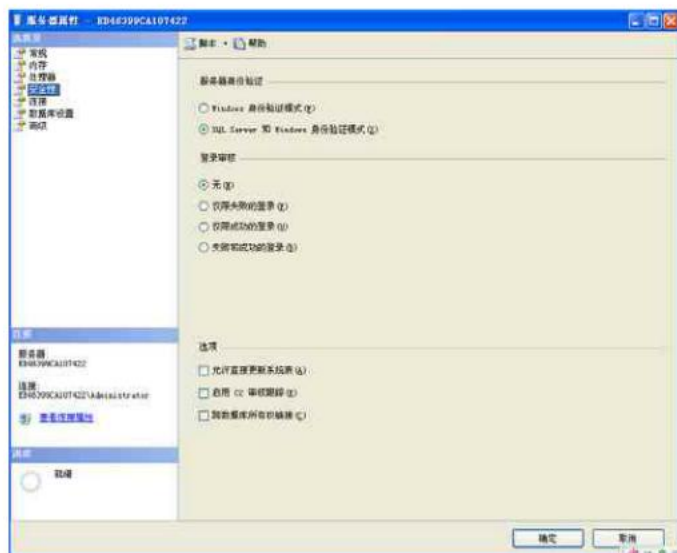


图 5-9 服务器身份验证

(5) 在“服务器身份验证”选项区中设置身份验证模式,如图 5-9 所示。更改身份验证模式后,需要重新启动 SQL Server 实例才能使其生效。

一般情况下,“Windows 身份验证模式”是推荐的身份验证模式。在 Windows 身份验证模式下,连接链路中没有密码信息,并且可以在集中的企业存储方案(例如 Active Directory)中管理用户帐户信息,使用操作系统已有的所有安全特性。然而,Windows 身份验证模式在某些情况下并不是最好的选择。例如,在需要为不属于自己操作系统环境的用户(例如外部供应商),或者所用操作系统与 Windows 安全体系不兼容的用户,提供访问授权的时候,则需要采用混合身份验证模式,并使用 SQL Server 登录名以便使这些用户可以连接到 SQL Server。

在 SQL Server 2008 中,不能指定 SQL Server 的身份验证模式,然而可以在 SQL Server 实例中设置安全特性来限制大多数 Windows 用户对 SQL Server 实例的访问。

## 2. 授权 Windows 用户及组连接到 SQL Server 实例

为 Windows 用户或者 Windows 组创建登录名,以允许这些用户连接到 SQL Server。默认情况下,只有本地 Windows 系统管理员组的成员和启动 SQL 服务的帐户才能访问 SQL Server。

### 注意:

可以删除本地系统管理员组对 SQL Server 的访问权限。

可以通过指令或者通过 SQL Server Management Studio 来创建登录名,以便授权用户对 SQL Server 实例的访问。下面的代码将授权 Windows 域用户 ADMINMAIN\tomlogin 对 SQL Server 实例的访问:

```
CREATE LOGIN [ADMINMAIN\tomlogin] FROM WINDOWS;
```

### 注意:

使用 SQL Server Management Studio 创建登录名时,SQL Server Management Studio 将执行与上面类似的 T-SQL 语句。

在安装 SQL Server 2008 实例时，安装程序会创建如表 5-1 所示的 Windows 登录名。

表 5-1 默认的 Windows 登录名

Windows 登录名	描 述
BUILTIN\Administrators	为已安装 SQL Server 实例的计算机的本地系统管理员组建立的登录名，这个登录名对于运行 SQL Server 不是必须的
<Servername>\SQLServerMSFTEUser\$ <Servername>\$MSSQLSERVER	为 Windows 组 SQLServerMSFTEUser\$<Servername> \$MSSQLSERVER 建立的登录名，这个组的成员具有作为 SQL Server 实例全文搜索服务的登录帐户应有的必要特权，该帐户对于运行 SQL Server 2005 全文搜索服务是必须的
<Servername>\SQLServerMSSQLUser\$ <Servername>\$MSSQLSERVER	为 Windows 组 SQLServerMSSQLUser\$<Servername> \$MSSQLSERVER 建立的登录名，这个组的成员具有作为 SQL Server 实例登录帐户应有的必要特权，该帐户对于运行 SQL Server 2008 是必须的，因为在实例建立后，使用本地服务帐户作为其服务帐户时，它就是 SQL Server 的服务帐户
<Servername>\SQLServerSQLAgentUser\$ <Servername>\$MSSQLSERVER	为 Windows 组 SQLServerSQLAgentUser\$<Servername> \$MSSQLSERVER 建立的登录名，这个组的成员具有作为 SQL Server 实例的 SQL Server Agent 的登录帐户应有的必要特权，该帐户对于运行 SQL Server 2008 Agent 服务是必须的

**注意：**

可以在不通知 SQL Server 的情况下，从操作系统中删除一个映射到 Windows 登录名的 Windows 用户或组。SQL Server 2008 不检查这种情况，需要定期检查 SQL Server 实例以便检查出这种孤立的登录名。可以通过使用系统存储过程 sp\_validatelogins 来执行检查。

### 3. 授权 SQL Server 登录名

在混合身份验证模式下，可以创建并管理 SQL Server 登录名。

创建 SQL Server 登录名时，要为登录名设置一个密码。当用户连接到 SQL Server 实例时必须提供密码。创建 SQL Server 登录名时，可以为其指定一个默认的数据库和一种默认语言。当应用程序连接到 SQL Server 但没有指定连接到哪个数据库和使用何种语言时，SQL Server 将为这个连接使用登录名的默认属性。

**注意：**

SQL Server 2008 使用自签名的认证方式加密登录时的网络通信包，以避免对登录信息的非授权访问(然而，一旦登录过程结束，并且登录被确认，SQL Server 将以明文的形式发送后续的所有信息)。可以通过 Secure Sockets Layer (SSL)和 Internet Pro 两种途径来实现安全而机密的网络通信。

可以通过指令或者通过 SQL Server Management Studio 来创建 SQL Server 登录名，以便授权用户对 SQL Server 实例的访问。以下代码将创建一个名为 Marylogin 的 SQL Server 登录



名, 并指定该登录名的默认数据库为“实例数据库”。

```
CREATE LOGIN Marylogin  
WITH PASSWORD = '674an7$52',  
DEFAULT_DATABASE = 实例数据库;
```

SQL Server 2008 安装进程在 SQL Server 2008 安装过程中创建了一个 SQL Server 登录名 sa。sa 登录名始终都会创建, 即使安装时选择的是 Windows 身份验证模式。

**注意:**

虽然不能删除 sa 登录名, 但可以通过改名或者禁用的方式避免用户通过该帐户对 SQL Server 进行非授权访问。

可以运行如下 T-SQL 语句以便通过 sql\_logins 目录视图来获取有关 SQL Server 登录名的信息。

```
SELECT *  
FROM sys.sql_logins;
```

#### 4. 实施密码策略

SQL Server 2008 能够对 SQL Server 登录名执行操作系统的密码实施策略。如果在 Windows 2003 服务器版上运行 SQL Server, SQL Server 将使用 NetValidatePasswordPolicy API (应用程序接口)来控制如下 3 点:

- 密码的复杂性
- 密码的生存周期
- 帐户锁定

如果在 Windows 2000 服务器版上运行 SQL Server, SQL Server 会使用 Microsoft Baseline Security Analyzer (MBSA)提供的本地密码复杂性规则来执行如下密码规则:

- 密码不能为空或者 NULL
- 密码不能为登录名
- 密码不能为机器名
- 密码不能为 Password, Admin, 或者 Administrator

可以使用如下 Transact-SQL 语句打开密码实施策略:

```
CREATE LOGIN Marylogin  
WITH PASSWORD = '674an7$52' MUST_CHANGE,  
CHECK_EXPIRATION = ON,  
CHECK_POLICY = ON;
```

#### 5. 拒绝用户访问

在某些情况下, 例如, 一个用户离开了组织, 需要拒绝一个特定的登录名对数据库的访

问。如果这个拒绝是临时的，可以通过禁用该登录名来完成。并不需要将该登录名从实例中删除。通过禁用访问，为数据库用户保留了登录名属性及其与数据库用户之间的映射关系。重新启用登录名时，可以像以前一样使用同样的登录名属性。执行如下 ALTER 语句可以启用或禁用登录名 Marylogin:

```
-- Disable the login
ALTER LOGIN Marylogin DISABLE;
-- Enable the login
ALTER LOGIN Marylogin ENABLE;
```

可以通过查询 sql\_logins 目录视图来检查被禁用的登录名，T-SQL 语句如下:

```
-- Query the system catalog view
SELECT * FROM sys.sql_logins
WHERE is_disabled=1;
```

**注意:**

在 SQL Server Management Studio 中，禁用的登录名有一个红色的箭头作为标记。该箭头显示在登录名的图标中。用户可以在“对象资源管理器”面板下的“安全性/登录名”中看到。

另一方面，如果需要从实例中删除一个登录名，可以使用 DROP LOGIN 语句。如下语句将删除登录名 Marylogin:

```
DROP LOGIN Marylogin;
```

**注意:**

删除登录名时，SQL Server 2008 不会删除与其映射的数据库用户。另外，删除与 Windows 用户或组映射的登录名并不能保证该用户或者该组的成员不能访问 SQL Server，该用户可能仍然属于有合法登录名的其他 Windows 组。

## 5.2.2 管理对 SQL Server 数据库的访问

对于需要进行数据访问的应用程序来说，仅仅授权其访问 SQL Server 实例是不够的。在授权访问 SQL Server 实例之后，还需要对特定的数据库进行访问授权。

可以通过创建数据库用户，并且将数据库登录名与数据库用户映射来授权对数据库的访问。为了访问数据库，除了服务器角色 sysadmin 的成员，所有数据库登录名都要在自己要访问的数据库中与一个数据库用户建立映射。sysadmin 角色的成员与所有服务器数据库上的 dbo 用户建立有映射。

### 1. 创建数据库用户

可以使用 CREATE USER 语句创建数据库用户。以下 Transact-SQL 示例创建了一个名为 Peterlogin 的登录名，并且将它与“实例数据库”中的 Peteruse 用户进行了映射。

```
-- Create the login Peterlogin
CREATE LOGIN Peterlogin WITH PASSWORD='234$7hf8';
-- Change the connection context to the database 实例数据库.
USE 实例数据库;
GO
-- Create the database user Peteruse,
-- mapped to the login Peterlogin in the database 实例数据库.
CREATE USER Peteruse FOR LOGIN Peterlogin;
```

## 2. 管理数据库用户

可以通过执行如下语句来检查当前的登录名是否可以登录到指定的数据库:

```
SELECT HAS_DBACCESS('实例数据库');
```

可以通过查询目录视图 `sys.database_principals` 来获取数据库用户的信息。

如果要临时禁用某个数据库用户对数据库的访问, 可以通过取消该用户的 `CONNECT` 授权来实现。如下语句将撤销用户 `Peteruse` 的 `CONNECT` 授权:

```
-- Change the connection context to the database 实例数据库.
USE 实例数据库;
GO
-- Revoke connect permission from Peteruse
-- on the database 实例数据库.
REVOKE CONNECT TO Peteruse;
```

可以使用 `DROP USER` 语句来删除一个数据库用户。

注意:

SQL Server 2008 不允许删除一个拥有数据库架构的用户。本章稍后将详细介绍架构的知识。

## 3. 管理孤立用户

孤立用户是指当前 SQL Server 实例中没有映射到登录名的数据库用户。在 SQL Server 2008 中, 用户所映射的登录名被删除后, 它就变成了孤立用户。为了获取孤立用户的信息, 可以执行如下语句:

```
-- Change the connection context to the database 实例数据库.
USE 实例数据库;
GO
-- Report all orphaned database users
EXECUTE sp_change_users_login @Action='Report';
```

SQL Server 2008 允许用户使用 `WITHOUT LOGIN` 子句来创建一个没有映射到登录名的用户。用 `WITHOUT LOGIN` 子句创建的用户不会被当作孤立用户, 这一特性在需要改变一个模块的执行上下文时非常有用。本章后续小节将详细介绍执行上下文。下面的语句将创建

一个没有映射到登录名的用户 Peteruse:

```
-- Change the connection context to the database 实例数据库.  
USE 实例数据库;  
GO  
-- Creates the database user Peteruse in the 实例数据库 database  
-- without mapping it to any login in this SQL Server instance  
CREATE USER Peteruse WITHOUT LOGIN;
```

#### 4. 启用 Guest 用户

当一个没有映射到用户的登录名试图登录到数据库时, SQL Server 将尝试使用 Guest 用户进行连接。Guest 用户是一个默认创建的没有任何权限的用户。可以通过为 Guest 用户授予 CONNECT 权限以启用 Guest 用户, 代码如下:

```
-- Change the connection context to the database 实例数据库.  
USE 实例数据库;  
GO  
-- Grant Guest access to the 实例数据库 database.  
GRANT CONNECT TO Guest;
```

在启用 Guest 用户时一定要谨慎, 因为这会给数据库系统环境的安全带来隐患。

## 5.3 管理角色

### 5.3.1 管理实例角色

前面介绍了如何允许一个用户访问 SQL Server 实例, 但尚未讨论这些登录名在 SQL Server 中具有哪些权限。由于用户需要访问某些数据, 因而需要创建具有执行管理任务权限的登录名。SQL Server 在实例级提供了服务器角色来实现这一任务。服务器角色是固定的, 不能在实例级创建角色。如表 5-2 所示为 SQL Server 2008 创建的固定服务器角色。

表 5-2 固定服务器角色

固定服务器角色	描 述
bulkadmin	可以运行 BULK INSERT 语句
dbcreator	可以新建、更新、删除和恢复数据库
diskadmin	可以管理磁盘文件
processadmin	可以结束进程
securityadmin	可以管理登录名和分配权限
Serveradmin	可以更改服务选项和关闭服务器
setupadmin	可以管理已连接的服务器并执行系统存储过程
sysadmin	可以在服务器上执行任何操作



通过查询系统函数 IS\_SRVROLEMEMBER, 可以查看当前用户是否属于一个服务器角色。如果实际登录名属于 sysadmin 服务器角色, 那么如下 Transact-SQL 语句将返回 1, 否则返回 0:

```
SELECT IS_SRVROLEMEMBER ('sysadmin');
```

可以使用系统存储过程 sp\_addsrvrolemember 为现有的服务器角色添加一个登录名。如下 T-SQL 语句将在 sysadmin 服务器角色中添加登录名 Marylogin:

```
EXECUTE sp_addsrvrolemember 'Marylogin', 'sysadmin';
```

可以使用存储过程 sp\_dropsrvrolemember 将一个登录名从服务器角色中删除。如下 T-SQL 语句将删除 sysadmin 服务器角色中的登录名 Marylogin:

```
EXECUTE sp_dropsrvrolemember 'Marylogin', 'sysadmin';
```

### 5.3.2 管理数据库角色

一旦创建了数据库用户, 随之而来的便是管理这些用户权限。可以通过将用户加入到一个数据库角色或者为其赋予更细的权限来管理用户。

#### 1. 创建数据库角色

数据库角色是数据库级的主体, 可以使用数据库角色来为一组数据库用户指定数据库权限。SQL Server 2008 为数据库创建了一套默认的数据库角色, 如表 5-3 所示。

表 5-3 默认的数据库角色

数据库角色	描 述
db_accessadmin	可以管理对数据库的访问
db_backupoperator	可以备份数据库
db_datareader	可以读取所有用户表中的所有数据
db_datawriter	可以在所有用户表中添加、删除和更新数据
db_ddladmin	可以执行任何 DDL(数据定义语言)命令
db_denydatareader	不能读取所有用户表中的所有数据
db_denydatawriter	不能在所有用户表中添加、删除和更新数据
db_owner	可以执行所有的配置和维护行为
db_securityadmin	可以修改数据库角色成员并管理权限
public	一个特别的数据库角色, 所有的数据库用户都属于 public 角色, 不能将用户从 public 角色中移除

可以根据特定的权限需求在数据库中加入角色来对数据库用户进行分组。如下一组 T-SQL 语句创建了一个名称为 Auditors 的数据库角色, 并在这个新角色中添加了数据库用户 Peter:

```
-- Change the connection context to the database 实例数据库.
USE 实例数据库;
GO
-- Create the role Auditorsrole in the database 实例数据库.
CREATE ROLE Auditorsrole;
GO
-- Add the user Peteruse to the role Auditorsrole
EXECUTE sp_addrolemember 'Auditorsrole', 'Peteruse';
```

## 2. 管理数据库角色

可以通过查询系统函数 IS\_MEMBER 来判断当前数据库用户是否属于某个数据库角色。

例如，下面的语句可以判断当前用户是否属于角色 db\_owner：

```
-- Change the connection context to the database 实例数据库.
USE 实例数据库;
GO
-- Checking if the current user belongs to the db_owner role
SELECT IS_MEMBER ('db_owner');
```

**注意：**

也可以使用 IS\_MEMBER 系统函数来判断当前数据库用户是否属于某个特定的 Windows 组，如下所示：

```
-- Change the connection context to the database 实例数据库.
USE 实例数据库;
GO
-- Checking if the current user belongs to the Managers group
-- in the ADVWORKS domain
SELECT IS_MEMBER ('[ADMINMAIN\tomlogin]');
```

通过使用 sp\_droprolemember 系统存储过程，可以从一个数据库角色中删除数据库用户。如果要删除一个数据库角色，可以使用 DROP ROLE 语句。如下语句将数据库用户 Peteruse 从数据库角色 Auditorsrole 中移除，然后删除了 Auditorsrole 角色：

```
-- Change the connection context to the database 实例数据库.
USE 实例数据库;
GO
-- Drop the user Peteruse from the Auditorsrole
EXECUTE sp_droprolemember 'Auditorsrole', 'Peteruse';
-- Drop the Auditorsrole from the current database
DROP ROLE Auditorsrole;
```

**注意：**

SQL Server 2008 不允许删除含有成员的角色。在删除一个数据库角色之前，必须先删除该角色下的所有用户。

### 5.3.3 管理应用程序角色

应用程序角色是特殊的数据库角色，允许用户通过特定的应用程序获取特定数据。应用程序角色不包含任何成员，而且在使用之前，需要在当前连接中将其激活。激活一个应用程序角色后，当前连接将丧失它所具备的特定用户权限，只获得应用程序角色所拥有的权限。

#### 1. 创建应用程序角色

可以使用 CREATE APPLICATION ROLE 语句来创建一个应用程序角色。示例如下：

```
-- Change the connection context to the database 实例数据库.
USE 实例数据库
GO
-- Create the Application role FinancialRole
-- in the current database
CREATE APPLICATION ROLE FinancialRole
WITH PASSWORD = '09$9ik985';
```

#### 2. 使用应用程序角色

应用程序角色在使用之前必须先激活。可以通过 sp\_setapprole 系统存储过程来激活应用程序角色。在连接关闭或执行系统存储过程 sp\_unsetapprole 之前，被激活的应用程序角色将一直保持激活状态。应用程序角色旨在由客户的应用程序使用，但同样可以在 T-SQL 批处理中使用它们。如下语句将调用存储过程激活应用程序角色 FinancialRole，然后再解除该操作。

```
-- Change the connection context to the database 实例数据库.
USE 实例数据库;
GO
-- Declare a variable to hold the connection context.
-- We will use the connection context later
-- so that when the application role is deactivated
-- the connection recovers its original context.
DECLARE @context varbinary (8000);
-- Activate the application role
-- and store the current connection context
EXECUTE sp_setapprole 'FinancialRole',
'09$9ik985',
@fCreateCookie = true,
@cookie = @context OUTPUT;
-- Verify that the user's context has been replaced
-- by the application role context.
SELECT CURRENT_USER;
-- Deactivate the application role,
-- recovering the previous connection context.
EXECUTE sp_unsetapprole @context;
GO
-- Verify that the user's original connection context
```

```
-- has been recovered.  
SELECT CURRENT_USER;  
GO
```

### 3. 删除应用程序角色

如果要删除应用程序角色，可以使用 `DROP APPLICATION ROLE` 语句。示例如下：

```
-- Change the connection context to the database 实例数据库.  
USE 实例数据库;  
GO  
-- Drop the application role FinancialRole  
-- from the current database  
DROP APPLICATION ROLE FinancialRole;
```

## 5.4 管理架构

SQL Server 2008 实现了 ANSI 中有关架构的概念。架构是一种允许用户对数据库对象进行分组的容器对象。架构对如何引用数据库对象有很大的影响。在 SQL Server 2008 中，一个数据库对象通过 4 个命名部分所组成的结构来引用，如下所示：

<服务器>.<数据库>.<架构>.<对象>

使用架构的一个好处是它可以将数据库对象与数据库用户分离，可以快速地从数据库中删除数据库用户。在 SQL Server 2008 中，所有的数据库对象都隶属于架构，在对数据库对象或者对其存在于数据库应用程序中的相应引用没有任何影响的情况下，可以更改并删除数据库用户。这种抽象的方法允许用户创建一个由数据库角色拥有的架构，以使多个数据库用户拥有相同的对象。

### 5.4.1 认识架构

可以使用 `CREATE SCHEMA` 语句来创建数据库架构。创建数据库架构时，可以在调用 `CREATE SCHEMA` 语句的事务中创建数据库对象并指定权限。以下示例将创建一个名称为 `Admnschema` 的架构，并将数据库用户 `Peteruse` 指定为该架构的所有者。随后，在这个架构下创建了一个名为 `Student` 的表。同时为数据库角色 `public` 授予 `select` 权限。具体代码如下：

```
-- Change the connection context to the database 实例数据库.  
USE 实例数据库;  
GO  
-- Create the schema Admnschema with Peteruse as owner.  
CREATE SCHEMA Admnschema  
AUTHORIZATION Peteruse;  
GO  
-- Create the table Student in the Admnschema.
```



```
CREATE TABLE Adminschema.Student (  
    StudentID int,  
    StudentDate smalldatetime,  
    ClientID int);  
GO  
-- Grant SELECT permission on the new table to the public role.  
GRANT SELECT ON Adminschema.Student  
TO public;  
GO
```

可以使用 DROP SCHEMA 语句删除一个架构。SQL Server 2008 不允许删除其中仍含有对象的架构。可以通过目录视图 sys.schemas 来获取架构的信息。如下语句将查询 sys.schemas 目录视图以获取架构信息:

```
SELECT *  
FROM sys.schemas;
```

下面的代码演示了如何查询现有架构所拥有的数据库对象、删除数据库对象, 以及删除架构的过程:

```
-- Change the connection context to the database 实例数据库.  
USE 实例数据库  
GO  
-- Retieve informationn about the Adminschema.  
SELECT s.name AS 'Schema',  
       o.name AS 'Object'  
FROM sys.schemas s  
INNER JOIN sys.objects o  
ON s.schema_id=o.schema_id  
WHERE s.name='Adminschema';  
GO  
-- Drop the table Student from the Adminschema.  
DROP TABLE Adminschema.Student;  
GO  
-- Drop the Adminschema.  
DROP SCHEMA Adminschema;
```

#### 5.4.2 使用默认架构

当一个应用程序引用一个没有限定架构的数据库对象时, SQL Server 将尝试在用户的默认架构中找到该对象。如果对象没有在默认架构中, SQL Server 将尝试在 dbo 架构中查找这个对象。如下语句演示了如何创建一个架构并将其指定为某个数据库用户的默认架构:

```
-- Create a SQL Server login in this SQL Server instance.  
CREATE LOGIN Marylogin
```

```
WITH PASSWORD=' 674an7$52 ';  
GO  
-- Change the connection context to the database 实例数据库.  
USE 实例数据库;  
GO  
-- Create the user Peteruse in the 实例数据库 database  
-- and map the user to the login Marylogin  
CREATE USER Peteruse  
FOR LOGIN Marylogin;  
GO  
-- Create the schema Adminschema, owned by Roseuse.  
CREATE SCHEMA Adminschema  
AUTHORIZATION Roseuse;  
GO  
-- Create the table Student in the newly created schema.  
CREATE TABLE Adminschema.Student (  
StudentID int,  
StudentDate smalldatetime,  
ClientID int);  
GO  
-- Grant SELECT permission to Peteruse on the new table.  
GRANT SELECT ON Adminschema.Student TO Peteruse;  
GO  
-- Declare the Adminschema as the default schema for Peteruse  
ALTER USER Peteruse  
WITH DEFAULT_SCHEMA= Adminschema;
```

## 5.5 管理权限

对数据的访问是通过 SQL Server 2008 的权限层次结构进行管理的。可以通过 GRANT、DENY 和 REVOKE 语句来管理这个权限层次结构。

- GRANT: 允许一个数据库用户或角色执行所授权限指定的操作。
- DENY: 拒绝一个数据库用户或角色的特定权限, 并且阻止它们从其他角色中继承这个权限。
- REVOKE: 取消先前被授予或拒绝的权限。

### 5.5.1 管理对 SQL Server 实例和数据库的访问

#### 1. 控制登录操作

SQL Server 2008 提供了一个精细的权限结构, 能够更出色地控制登录操作。可以通过 GRANT、DENY 和 REVOKE 语句来控制权限, 通过 sys.Server\_permissions 目录视图来获取有关服务权限的信息。

如下语句将为登录名 Marylogin 授予创建和执行 SQL Server Profiler 跟踪的权限：

```
GRANT ALTER TRACE TO Marylogin;
```

用户可以通过使用 fn\_my\_permissions 函数来了解自己的权限。如下语句将显示用户的权限：

```
SELECT * FROM fn_my_permissions (NULL, 'SERVER');
```

## 2. 为数据库角色授予权限

除了使用固定数据库角色之外，还可以为数据库角色授予小粒度的数据库权限。下面的语句将为数据库用户 Peteruse 授予 BACKUP DATABASE(备份数据库)权限：

```
-- Change the connection context to the database 实例数据库.  
USE 实例数据库;  
GO  
-- Grant permissions to the database user Peteruse  
-- to backup the database 实例数据库.  
GRANT BACKUP DATABASE TO Peteruse;
```

## 5.5.2 管理对表和列的访问

### 1. 更改对表的访问

用户对表所拥有的有效权限控制着用户对表的访问行为。可以通过管理表的权限来控制数据库用户对表的访问。如表 5-4 所示的就是可以管理的表的权限。可以对数据库用户或角色指定这些权限。

表 5-4 表的权限

权 限	描 述
ALTER	可以更改表属性
CONTROL	提供所有权之类的权限
DELETE	可以从表中删除行
INSERT	可以向表中插入行
REFERENCES	可以通过外键引用其他表
SELECT	可以在表中选择行
TAKE OWNERSHIP	可以取得表的所有权
UPDATE	可以在表中更新行
VIEW DEFINITION	可以访问表的元数据

可以使用 GRANT 语句授权数据库用户或者角色对表的访问。如下语句将授予用户 Peteruse 对表 Adminschema.Student 的 SELECT、INSERT 和 UPDATE 权限：

```
-- Change the connection context to the database 实例数据库.  
USE 实例数据库;
```

```
GO
-- Grant some permissions to Peteruse on the Adminschema.Student table.
GRANT SELECT,INSERT,UPDATE
ON Adminschema.Student
TO Peteruse;
```

限制对表的访问，有两种不同的情况。如果已经为用户授予了表的这种权限，则应该使用 REVOKE 语句清除之前授予的权限。示例如下：

```
-- Change the connection context to the database 实例数据库.
USE 实例数据库;
GO
-- Revoke SELECT permissions from Peteruse on the Adminschema.Student table
REVOKE SELECT
ON Adminschema.Student
TO Peteruse;
```

然而，如果用户隶属于某个具备此权限的角色，则用户可能依然具备通过 REVOKE 语句取消的权限。在这种情况下，需要使用 DENY 语句来拒绝该用户的访问。示例如下：

```
-- Change the connection context to the database 实例数据库.
USE 实例数据库;
GO
-- Deny DELETE permission to Peteruse on the Adminschema.Student table,
-- regardless of what permissions this user might
-- inherit from roles.
DENY DELETE
ON Adminschema.Student
TO Peteruse;
```

## 2. 提供对列的单独访问

SQL Server 2008 提供了授予或拒绝访问单独列的权限，这个特性提供了灵活的拒绝访问机制，例如，保护某些列上的机密数据。如表 5-5 所示的是可以管理的列权限。

表 5-5 列权限

权 限	描 述
SELECT	可以选择列
UPDATE	可以更新列
REFERENCE	可以通过外键引用列

授权对列的访问，也使用 GRANT 语句。以下示例为 Peteruse 用户授予了在表 Adminschema.Student 的 StudentDate 和 ClientID 列上 SELECT 和 UPDATE 的权限。

```
-- Change the connection context to the database 实例数据库.
USE 实例数据库;
```



```
GO
-- Grant SELECT and UPDATE permissions to Peteruse
-- on some specific columns of the Adminschema.Student table
GRANT SELECT,UPDATE (
StudentDate,
ClientID)
ON Adminschema.Student
TO Peteruse;
```

取消对列的访问授权，与取消对表的访问授权类似，可以使用 REVOKE 语句来实现，但如果要阻止一个用户获得某种权限，则需要使用 DENY 语句。

```
-- Change the connection context to the database 实例数据库.
USE 实例数据库;
GO
-- Revoke previosly granted or denied permissions
-- from Peteruse on the StudentDate column.
REVOKE UPDATE (StudentDate)
ON Adminschema.Student
TO Peteruse;
```

### 5.5.3 管理对可编程对象的访问

可编程对象，如存储过程及用户定义的函数，具有自己的安全上下文。数据库用户需要获得授权才能执行存储过程、函数和程序集。一旦数据库引擎检查了执行可编程对象的权限，就会在可编程对象内部对其所执行的操作进行权限检查。当数据库对象按顺序相互访问时，该访问顺序将形成一个所有权链。

#### 1. 管理存储过程的安全性

在各种数据库对象中，存储过程是数据库开发人员最常使用的数据库对象。与其他数据库对象一样，存储过程也是需要保护的对象。用户需要具备执行操作的权限，就像创建一个存储过程一样，用户需要具备相应的权限来执行一个存储过程。如表 5-6 所示是可为存储过程授予的权限。

表 5-6 存储过程权限

权 限	描 述
ALTER	可以更改存储过程属性
CONTROL	可以提供所有权之类的权限
EXECUTE	可以执行存储过程
TAKE OWNERSHIP	可以取得存储过程的所有权
VIEW DEFINITION	可以查看存储过程的元数据

在执行一个存储过程时，SQL Server 会检查当前数据库用户是否具有该存储过程的

EXECUTE 权限。下面的语句将为数据库用户 Peteruse 授予存储过程 dbo.uspGetBillOfMaterials 的 EXECUTE 权限：

```
-- Change the connection context to the database 实例数据库.
USE 实例数据库;
GO
-- Grant EXECUTE permission to Peteruse on a stored procedure.
GRANT EXECUTE ON dbo.uspGetBillOfMaterials
TO Peteruse;
```

同样地,如果要阻止一个用户执行某个存储过程,可以取消或者拒绝该用户的 EXECUTE 权限。

## 2. 管理用户定义函数的安全性

用户定义函数和存储过程一样,也是可编程对象。主要存在两种类型的用户定义函数:只返回单一值的标量函数和返回一个表数据类型值的表值函数。根据用户定义函数类型的不同,可以对函数授予 EXECUTE 或 SELECT 权限,如表 5-7 所示。

表 5-7 用户定义函数权限

权 限	描 述
ALTER	可以更改函数属性
CONTROL	可以提供所有权之类的权限
TAKE OWNERSHIP	可以取得函数的所有权
VIEW DEFINITION	可以查看函数的元数据
SELECT	可以选择表值函数所返回的数据(只对表值函数有效)
EXECUTE	可以执行用户定义函数(只对标量函数有效)

对于执行表值函数,SQL Server 将检查用户是否拥有此函数所返回的表的 SELECT 权限。可以采用与为表授予 SELECT 权限相同的方式来为表值函数授予 SELECT 权限。如下语句将授予数据库用户 Peteruse 对用户定义函数 dbo.ufnGetContactInformation 的 SELECT 权限:

```
-- Change the connection context to the database 实例数据库.
USE 实例数据库;
GO
-- Grant permission to Peteruse to execute a user defined function.
GRANT SELECT ON dbo.ufnGetContactInformation
TO Peteruse;
```

### 注意:

表值函数还有另外一种类型,叫做内联函数。内联函数在功能上等同于视图,但是它支持参数。从安全角度来讲,这种类型的函数等同于视图。

对于执行标量函数,数据库用户需要在函数上具备 EXECUTE 权限。可以采用与为存储

过程授予 EXECUTE 权限相同的方式来为标量函数授予 EXECUTE 权限。如下语句将授予数据库用户 Peteruse 对用户定义函数 dbo.ufnGetContactInformation 的 EXECUTE 权限:

```
-- Change the connection context to the database 实例数据库.  
USE 实例数据库;  
GO  
-- Grant Peteruse permission to execute a user defined function.  
GRANT EXECUTE ON dbo.ufnGetStock  
TO Peteruse;
```

### 3. 管理程序集的安全性

SQL Server 2008 提供了在数据库引擎内部包含 .NET 程序集(引用 .dll 文件的对象), 并在存储过程及函数中调用这些程序集的能力。可以为程序集分配与存储过程一样的权限, 这些权限如表 5-6 所示。

#### (1) 权限集

创建一个程序集时, 需要指定一个权限集。权限集指定了程序集在 SQL Server 中所授予的一个代码访问权限的集合。权限集具有如下 3 种不同的类型。

- **SAFE 类型:** 程序集执行的代码不能访问外部系统资源。SAFE 类型是最受限制的权限集合, 并且是默认的类型。
- **EXTERNAL\_ACCESS 类型:** 程序集可以访问外部系统资源。
- **UNSAFE 类型:** 程序集可以执行非托管代码。

对于不需要访问外部资源的程序集, 推荐使用 SAFE 类型的权限集。

#### (2) 执行一个程序集

当一个应用程序尝试访问程序集中的对象时, 数据库引擎会检查当前用户是否具有该程序集的 EXECUTE 权限。如下语句将授予数据库用户 Peteruse 对程序集的 EXECUTE 权限:

```
-- Change the connection context to the database 实例数据库.  
USE 实例数据库;  
GO  
-- Grant Peteruse permission to execute an assembly.  
GRANT EXECUTE ON <AssemblyName>  
TO Peteruse;
```

通过为一个程序集授予 EXECUTE 权限, 可以为数据库用户授予对程序集中所有对象的 EXECUTE 权限。

### 4. 管理所有权链

所有权链是数据对象互相访问的顺序。例如, 在一个存储过程中, 向一个表中插入一行数据, 存储过程称为调用对象, 表称为被调用对象。SQL Server 遍历这个链中的链接时, 与单独访问数据库对象时的方式不同, 数据库引擎会以另一种方式评估对对象的访问权限。

在一个链中访问对象时, SQL Server 首先会比较对象的所有者与调用对象的所有者。如果

两个对象的所有者相同, 则不评估被引用对象的权限。这个特性对管理对象权限非常有用。例如, 假设数据库用户 Peteruse 创建了一个名称为 Person.SupplierContacts 的表, 并在一个名称为 Person.InsertSupplierContacts 的存储过程中向 PersonSupplierContacts 表中插入了行。由于这两个数据对象具有同样的所有者 Peteruse, 因此, 只需授予其他用户对存储过程 Person.InsertSupplierContacts 的 EXECUTE 权限, 以允许其他用户在访问表 PersonSupplierContacts 时, 依然具有 EXECUTE 权限。

#### 注意:

所有权链提供了一种强大的封装算法。一个数据可以被设计成只通过充分文档化的公共接口(例如存储过程和用户定义函数)来对外提供数据访问, 这些存储过程和用户定义函数隐藏了数据设计实现的复杂性。数据库开发人员可以充分利用所有权链, 在拒绝所有用户对数据库中所有表的访问的同时, 仍然可以允许其访问数据。

### 5. 管理执行上下文

执行上下文由连接到相应会话的用户、登录名或者由执行(调用)相应模块的用户或登录名确定。在 SQL Server 2008 进行对象权限检查时, 登录名和用户令牌为其提供了所需的信息。在 SQL Server 2008 中, 可以使用 EXECUTE AS 语句来更改执行上下文。这一操作称为切换执行上下文。

#### (1) 运行 EXECUTE AS

EXECUTE AS 语句允许显式地定义当前连接的执行上下文。可以用 EXECUTE AS 更改当前连接的登录名或者数据库用户。上下文的变化在另一个上下文变更发生前、连接关闭前或者一个 REVERT 语句执行前始终是有有效的。如下语句使用了 EXECUTE AS 语句为数据库用户 Peteruse 更改执行上下文:

```
-- Change the connection context to the database 实例数据库.
USE 实例数据库;
GO
-- Change the execution context to the user Peteruse.
EXECUTE AS USER=' Peteruse ';
-- The following statement will be executed under Peteruse 's credentials.
TRUNCATE TABLE dbo.ErrorLog;
```

由于用户 Peteruse 没有 truncate 表的权限, 因此, 上述代码将会产生一个错误。而去下 truncate 表的语句则能成功执行:

```
-- Change the execution context back to the original state
REVERT;
-- Now the following statement will be executed under
-- the original execution context.
TRUNCATE TABLE dbo.ErrorLog;
```

#### (2) 管理上下文切换

除了控制批处理(批处理是包含一个或多个 Transact-SQL 语句的组, 这个组从应用程序

一次性地发送到 SQL Server 执行,就像前面的 TRUNCATE TABLE 示例一样)的执行上下文,还可以控制存储过程 and 用户定义函数的执行上下文。在这些模块中切换上下文时,可以控制在这些存储过程或者函数中使用哪个用户帐户来访问它所引用的数据库对象。为此,只需要对 EXECUTE AS 语句进行如下改动即可。

- **CALLER:** 存储过程或者用户定义函数内的语句都在模块调用者的上下文中执行。
- **SELF:** 所有语句在创建或者更改存储过程或者用户定义函数的用户的上下文中执行。
- **OWNER:** 所有语句在存储过程或者用户定义函数的当前所有者的上下文中执行。
- **<User>:** 所有语句在指定数据库用户或者登录名的上下文中执行。

以下示例将切换上下文到数据库用户 **dbo** 的上下文中以创建一个存储过程。然后,为数据库用户 **Peteruse** 授予这个新建存储过程的 EXECUTE 权限,并更改上下文以测试存储过程的执行:

```
-- Create a stored procedure to execute statements
-- as dbo.
CREATE PROCEDURE dbo.usp_TruncateErrorLog
    WITH EXECUTE AS 'dbo'
AS
    TRUNCATE TABLE dbo.ErrorLog;
GO
-- Grant permissions to execute this procedure to Peteruse.
GRANT EXECUTE ON dbo.usp_TruncateErrorLog TO Peteruse
-- Change the execution context of this batch to Peteruse.
EXECUTE AS [USER=]'Peteruse'
-- Execute the stored procedure.
EXECUTE dbo.usp_TruncateErrorLog
```

由此可见,在不能通过授权来使用户执行某些操作时,尤其适合使用上下文切换。

## 5.6 习 题

1. 简述 SQL Server 2008 的安全层次?
2. 对 SQL Server 实例访问,SQL Server 2008 支持哪几种身份验证模式?
3. 在 SQL Server 2008 中有几类角色?
4. 什么是架构,架构有什么用处?
5. 如何管理 SQL Server 2008 的权限层次结构?
6. 管理对可编程对象的访问主要涉及哪些可编程对象?



## 第7章 查询和视图

在第二章我们学习了使用 SQL Server 2008 的图形用户界面方式来创建和操作数据库与表,本章我们将学习使用 T-SQL 语句来完成相同的任务。与图形用户界面方式相比,用 T-SQL 命令方式更为灵活。

根据 T-SQL 语言的功能特点,我们从以下角度探讨 T-SQL 的高级应用: T-SQL 作为数据定义语言(DDL)、作为数据查询语言(SELECT)和作为数据操纵语言(DML)的不同语法与应用。通过这些学习,使读者进一步掌握 T-SQL 这一综合的、通用的、功能强大的关系数据库语言。

**本章学习目标:**

- 掌握 T-SQL 作为数据定义语言的语法与应用
- 掌握 WHERE、ORDER BY、GROUP BY、HAVING 子句的使用
- 掌握基本的多表查询
- 掌握内连接、外连接、交叉连接和联合查询的使用
- 掌握多行和单值子查询的使用
- 掌握嵌套子查询的使用
- 了解表的视图

### 7.1 连接、选择和投影

SQL Server 2008 是一种关系数据库管理系统,在关系数据库中,必须提供一种对二维表进行运算的机制。这种机制除了包括传统的集合运算中的并、交、差、广义笛卡尔积以外,还包括专门的关系运算中的选择、投影和连接。

#### 7.1.1 选择(Selection)

选择是单目运算,它是按照一定的条件,从关系 R 中选择出满足条件的行为作为结果返回。选择运算的操作对象是一张二维表,其运算结果也是一张二维表。选择运算的记号为  $\sigma_F(R)$ ,其中  $\sigma$  是选择运算符,下标 F 是一个条件表达式, R 是被操作的表。

选择运算符的含义是:在关系 R 中选择满足给定条件的诸元组。

$$\sigma_F(R) = \{t \mid t \in R \wedge F(t) = \text{'真'}\}$$

F: 选择条件,是一个逻辑表达式,基本形式如下:

$$[ (X_1 \theta Y_1) ] [\varphi [ (X_2 \theta Y_2) ] ] \dots$$

$\theta$ : 比较运算符(>, ≥, <, ≤, =或<>)

X1, Y1 等: 属性名、常量、简单函数; 属性名也可以用它的序号来代替;

$\phi$ : 逻辑运算符( $\wedge$ 或 $\vee$ )

[]: 表示任选项

...: 表示上述格式可以重复下去。

例【7-1】设有一个学生-课程数据库, 包括学生关系 Student, 查询信息系(IS 系)全体学生。

$\sigma \text{ Sdept} = \text{'IS'} (\text{Student})$

或  $\sigma 5 = \text{'IS'} (\text{Student})$

结果如表 7-1 所示。

表 7-1 STUDENT 表

学 号	姓 名	性 别	年 龄	系别 Sdept
95001	李勇	男	20	CS
95002	刘晨	女	19	IS
95003	王敏	女	18	MA
95004	张立	男	19	IS

### 7.1.2 投影(Projection)

投影也是单目运算, 该运算从表中选出指定的属性值组成一个新表, 记为:  $\Pi A(R)$ 。

其中 A 是属性名(即列名)表, R 是表名。

投影运算符的含义是: 从 R 中选择出若干属性列组成新的关系。

$$\pi A(R) = \{ t[A] \mid t \in R \}$$

A: R 中的属性列。

投影操作主要是从列的角度进行运算, 但投影之后不仅取消了原关系中的某些列, 而且还可能取消某些元组(避免重复行)。

例【7-2】查询学生的姓名和所在系。

即求 Student 关系上学生姓名和所在系两个属性上的投影。

$\pi \text{ Sname, Sdept}(\text{Student})$

或  $\pi 2, 5(\text{Student})$

结果如表 7-2 所示。

表 7-2 投影结果

Sname	Sdept
李勇	CS
刘晨	IS
王敏	MA
张立	IS

### 7.1.3 连接(JOIN)

把两个表中的行按着给定的条件拼接而形成的新表。

连接也称为  $\theta$  连接, 连接运算的含义: 从两个关系的笛卡尔积中选取属性间满足一定条件的元组。

A 和 B: 分别为 R 和 S 上度数相等且可比的属性组

$\theta$ : 比较运算符

连接运算从 R 和 S 的广义笛卡尔积  $R \times S$  中选取(R 关系)在 A 属性组上的值与(S 关系)在 B 属性组上值满足比较关系的元组。

$\theta$  为 "=" 的连接运算称为等值连接, 等值连接的含义: 从关系 R 与 S 的广义笛卡尔积中选取 A、B 属性值相等的那些元组。

一般的连接操作是从行的角度进行运算。

## 7.2 数据查询语句(SELECT 语句)

查询功能是 T-SQL 的核心, 通过 T-SQL 的查询可以从表或视图中迅速、方便地检索数据。

T-SQL 查询最基本的方式是 SELECT 语句, 其功能十分强大。它能够以任意顺序、从任意数目的列中查询数据, 并在查询过程中进行计算, 甚至能够包含来自其他表中的数据。

本节我们选取 stuinfo 数据库作为示例数据库, 以下例句均省略打开数据库的 USE stuinfo 语句。

stuinfo 数据库中有 3 张数据表: student、course 和 sc。它们的结构分别如表 7-3、表 7-4 和表 7-5 所示。

表 7-3 student 表的结构

字段名称	数据类型	长度(字节)	NULL	备注
sno	char	10	not null	主键, 学号
sname	nchar	10	not null	学生姓名
sage	int	4	null	年龄
ssex	nchar	4	null	性别
snation	nchar	10	null	民族
sdept	nchar	20	null	所属院系
snativeplace	nchar	6	null	籍贯
spoliticalstatus	nchar	6	null	政治面貌

表 7-4 course 表的结构

字段名称	数据类型	长度(字节)	NULL	备注
cno	char	5	not null	主键, 课程号
cname	nvarchar	20	not null	课程名称
ccredit	nchar	4	null	学分
cteacher	nchar	10	null	任课教师

表 7-5 sc 表的结构

字段名称	数据类型	长度(字节)	NULL	备 注
sno	char	10	not null	学号
cno	char	5	not null	课程号
grade	decimal	4	null	成绩, 2 位小数位

SELECT 语句的完整语法格式如下:

```
SELECT <列名选项>
FROM <表名>|<视图名称>
[WHERE <查询条件>|<联接条件>]
[GROUP BY <分组表达式>[HAVING <分组统计表达式>]]
[ORDER BY <排序表达式>[ASC|DESC]]
```

从上述语法可以看出, SELECT 查询语句中共有 5 个子句, 其中, SELECT 和 FROM 语句为必选子句, 而 WHERE、ORDER BY 和 GROUP BY 子句为可选子句, 要根据查询的需要进行选用。下面对 SELECT 语法中各参数进行如下说明:

- SELECT 子句: 用来指定查询返回的列, 各列在 SELECT 子句中的顺序决定了它们在结果表中的顺序;
- FROM 子句: 用来指定数据来源的表;
- WHERE 子句: 用来限定返回行的搜索条件;
- GROUP BY 子句: 用来指定查询结果的分组条件;
- ORDER BY 子句: 用来指定结果的排序方式。

SELECT 语句可以写在一行中。但对于复杂的查询, SELECT 语句随着查询子句的增加不断增长, 一行很难写下, 此时, 可以采用分行的写法, 即每个子句分别在不同的行中。需要注意的是, 子句与子句之间不能使用符号分隔。

下面, 我们按照由简到繁、由易到难的顺序, 讨论 SELECT 语句的基本结构和主要功能。

### 7.2.1 SELECT 语句对列的查询

对列的查询实质上是对关系的“投影”操作。在很多情况下, 用户只对表中的一部分列感兴趣, 可以使用 SELECT 子句来指明要查询的列, 并根据需要改变输出列显示的顺序。

T-SQL 中对列的查询是通过对 SELECT 子句中的列名选项进行设置完成的, 具体格式如下:

```
SELECT [ALL|DISTINCT] [TOP n [PERCENT]]
{ *|表的名称.*|视图名称.*                      /*选择表或视图中的全部列*/
  | 列的名称|列的表达式 [[AS] 列的别名]          /*选择指定的列*/
}, ...n]
```

#### 1. 查询一个表中的全部列

选择表的全部列时, 可以使用星号 “\*” 来表示所有的列。

例【7-3】检索 students 表、course 表和 sc 表中的所有记录。

T-SQL 语句如下：

```
USE StuInfo
SELECT * FROM student
SELECT * FROM course
SELECT * FROM sc
```

执行结果如图 7-1 所示。

也可以将 3 个表的操作合并为一条语句，使结果在一个表中显示，T-SQL 语句如下：

```
SELECT student.*, course.*, sc.* FROM student, course, sc
```

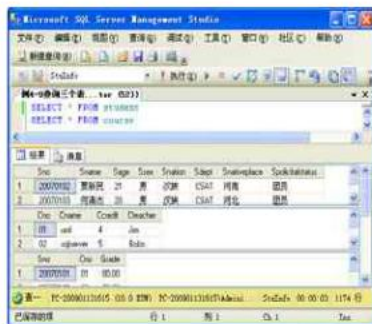


图 7-1 查询 3 个表的全部信息

需要注意的是：如果有大量数据要返回，或者数据是通过网络返回的，为了防止返回的数据比实际需要的多，则不应该使用星号“\*”，而应该指定所需的列名。

## 2. 查询一个表中部分列

如果查询数据时只需要选择一个表中的部分列信息，则在 SELECT 后给出需要的列即可，各列名之间用逗号分隔。

例【7-4】检索 student 表中学生的部分信息，包括学号、学生姓名和所属院系。

T-SQL 语句如下：

```
SELECT sno, sname, sdept FROM student
```

执行结果如图 7-2 所示。

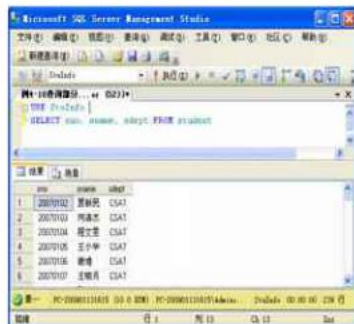


图 7-2 查询部分列信息



### 3. 为列设置别名

通常情况下,当从一个表中取出列值时,该值与列的名称是联系在一起的。如上例中从 student 表中取出学号与学生姓名,取出的值就与 sno 和 sname 有联系。当希望查询结果中的列使用新的名字来取代原来的列名称时,可以使用如下方法:

- 在列名之后使用 AS 关键字来更改查询结果中的列标题名,如 sno AS 学号;
- 直接在列名后使用列的别名,列的别名可以带双引号、单引号或不带引号。

例【7-5】检索 student 表中学生的 sno、sname、sage 和 sdept,指定结果中各列的标题分别为学生学号、学生姓名、年龄和所属院系。

T-SQL 语句如下:

```
SELECT sno as 学生学号, sname 学生姓名, sage '年龄', sdept "所属院系"
FROM student
```

执行结果如图 7-3 所示。



图 7-3 为列设置别名

### 4. 计算列值

使用 SELECT 语句对列进行查询时,SELECT 后面可以跟列的表达式。也就是说,使用 SELECT 语句不仅可以查询原表中已有的列,而且还可以通过计算得到新的列。

例【7-6】查询 sc 表中的学生成绩,并显示折算后的分数。(折算方法:原始分数\*1.2)

T-SQL 语句如下:

```
SELECT sno, grade AS 原始分数, grade*1.2 AS 折算后分数 FROM sc
```

执行结果如图 7-4 所示。

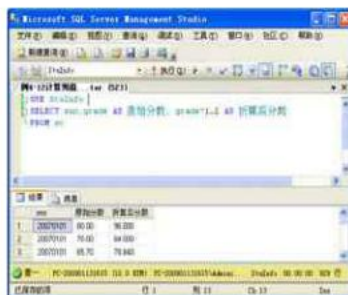


图 7-4 计算列值

### 5. 消除结果中的重复项

在一张完整的数据库表中不可能出现两个完全相同的记录,但由于我们在查询时经常只涉及表的部分字段,这样,就有可能出现重复的行,可以使用 **DISTINCT** 短语来避免这种情况。

关键字 **DISTINCT** 的含义是对结果集中的重复行只选择一个,以保证行的唯一性。

例【7-7】从 **student** 表中查询所有的院系信息,并去掉重复信息。

T-SQL 语句如下:

```
SELECT DISTINCT sdept FROM student
```

执行结果如图 7-5 所示。可以看到, **student** 表一共只有 4 个专业。

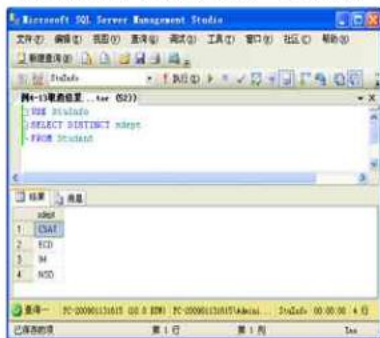


图 7-5 取消结果重复项

与 **DISTINCT** 相反,当使用关键字 **ALL** 时,将保留结果中的所有行。在省略 **DISTINCT** 和 **ALL** 的情况下, **SELECT** 语句默认为 **ALL**。

### 6. 限制结果返回的行数

如果 **SELECT** 语句返回的结果行数非常多,而用户只需要返回满足条件的前几条记录,则可以使用 **TOP n [PERCENT]** 可选子句。其中, **n** 是一个正整数,表示返回查询结果的前 **n** 行。如果使用 **PERCENT** 关键字,则表示返回结果的前 **n%** 行。

例【7-8】查询 **student** 表中的前 10 个学生的信息。

T-SQL 语句如下:

```
SELECT TOP 10 * FROM student
```

执行结果如图 7-6 所示,只返回了 10 个学生的信息。



图 7-6 限制结果返回的行数

## 7.2.2 SELECT 语句中的条件查询

条件查询是用的最多的一种查询方式,通过在 WHERE 子句中设置查询条件可以挑选出符合要求的数据、修改某一记录、删除某一记录。条件查询的本质是对表中的数据进行筛选,即关系运算中的“选择”操作。

在 SELECT 语句中,WHERE 子句必须紧跟在 FROM 子句之后,其基本格式如下:

WHERE <查询条件>

其中,查询条件的使用主要有以下几种情况。

### 1. 使用比较运算符

使用上一章介绍的比较运算符来比较表达式值的大小,包括:=(等于)、>(大于)、<(小于)、>=(大于等于)、<=(小于等于)、!=(不等于)、<>(不等于)、!<(不小于)、!>(不大于)。运算结果为 TRUE 或者 FALSE。

例【7-9】在 student 表中查询信息系(IM)的学生。

T-SQL 语句如下:

```
SELECT * FROM student WHERE sdept='IM'
```

执行结果如图 7-7 所示,显示的全是信息系的学生。



Sno	Sname	Sage	Sex	Sdept	Sstatus	Snativeplace	Sgrade
1	付建王	20	女	信息系	在职	河南	1
2	孙建王	21	女	信息系	在职	河南	1
3	孙建王	21	女	信息系	在职	河南	1
4	王保	20	女	信息系	在职	河南	1
5	南长	21	男	信息系	在职	河南	1
6	孙建王	22	女	信息系	在职	河南	1
7	李洋洋	20	男	信息系	在职	河南	1

图 7-7 使用比较运算符

### 2. 使用逻辑运算符

逻辑运算符包括 AND、OR 和 NOT,用于连接 WHERE 子句中的多个查询条件。当一条语句中同时含有多个逻辑运算符时,取值的优先顺序为:NOT、AND 和 OR。

例【7-10】在 student 表中查询年龄小于 18 或者大于 22,并且籍贯是河南的学生信息。

T-SQL 语句如下:

```
SELECT * FROM student  
WHERE (sage<18 or sage>22) and snativeplace='河南'
```

执行结果如图 7-8 所示,显示满足条件的有 3 名同学。





图 7-8 使用逻辑运算符

### 3. 使用 LIKE 模式匹配

在查找记录时,如果不是很适合使用算术运算符和逻辑运算符,则可能要用到更高级的技术。例如,当不知道学生全名而只知道姓名的一部分时,就可以使用 LIKE 搜索学生情况。

LIKE 是模式匹配运算符,用于指定一个字符串是否与指定的字符串相匹配。使用 LIKE 进行匹配时,可以使用通配符,即可以使用模糊查询。

T-SQL 中使用的通配符有“%”、“\_”、“[ ]”和“[^]”。通配符用在要查找的字符串的旁边。它们可以一起使用,使用其中的一种并不排斥使用其他的通配符。

- “%”用于字符串的末尾或开始处,代表 0 个或任意多个字符。如果要查找姓名中有“a”的教师,可以使用“%a%”,这样会查找出姓名中任何位置包含字母“a”的记录;
- “\_”代表单个字符。使用“\_a”,将返回任何名字为两个字符且第二个字符是“a”的记录;
- “[ ]”允许在指定值的集合或范围中查找单个字符。如,要搜索名字中包含介于 a-f 之间的单个字符的记录,则可以使用 LIKE “[a-f]”;
- “[^]”与 “[ ]”相反,用于指定不属于范围内的字符。如, [^abcdef] 表示不属于 abcdef 集合中的字符。

例【7-11】在 students 表中查询姓“赵”的学生信息。

T-SQL 语句如下:

```
SELECT * FROM student WHERE sname like N'赵%'
```

执行结果如图 7-9 所示。



图 7-9 使用 LIKE 模式匹配

#### 4. 确定范围

T-SQL 中与范围相关的关键字有两个：BETWEEN 和 IN。

当要查询的条件是某个值的范围时，使用 BETWEEN...AND...来指出查询范围。其中，AND 的左端给出查询范围的下限，AND 的右端给出查询范围的上限。

例【7-12】在 sc 表中，查询成绩在 60 到 80 分之间的学生情况。

T-SQL 语句如下：

```
SELECT * FROM sc WHERE grade between 60 and 80
```

执行结果如图 7-10 所示，可以看到，使用 BETWEEN 查询，结果包含两个端点的值。在本例中，包含了 60 分和 80 分的学生信息。

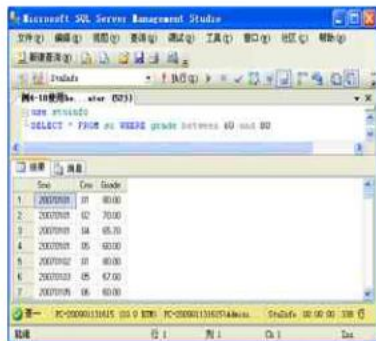


图 7-10 使用 BETWEEN 确定范围

关键字 IN 用于表示查询范围属于指定的集合。集合中列出所有可能的值，当表中的值与集合中的任意一个值匹配时，即满足条件。

例【7-13】在 student 表中查询 IM 系和 CAST 系的同学情况。

T-SQL 语句如下：

```
SELECT * FROM student WHERE sdept IN('NCSAT','NIM')
```

该语句等价于：

```
SELECT * FROM student WHERE Sdept='NCSAT' or Sdept='NIM'
```

执行结果如图 7-11 所示。

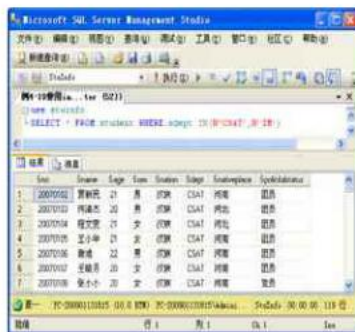


图 7-11 使用 IN 确定范围



### 5. 涉及空值 NULL 的查询

值为“空”并非没有值，而是一个特殊的符号“NULL”。一个字段是否允许为空，是在建立表的结构时设置的。要判断一个表达式的值是否为空值时，可以使用 IS NULL 关键字。

例【7-14】查询缺少单科成绩的学生的信息。

T-SQL 语句如下：

```
SELECT * FROM sc WHERE grade IS NULL
```

执行结果如图 7-12 所示。

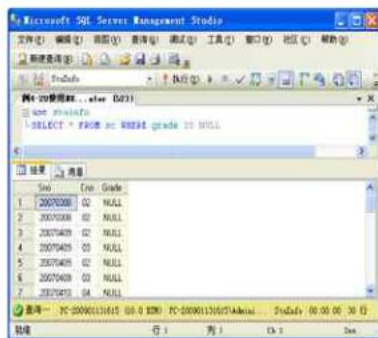


图 7-12 涉及 NULL 的查询

### 7.2.3 ORDER BY 子句的使用

利用 ORDER BY 子句可以对查询的结果按照指定的字段进行排序。

ORDER BY 子句的语法格式如下：

```
ORDER BY 排序表达式 [ASC|DESC]
```

其中，ASC 代表升序，DESC 表示降序，默认为升序排列。对数据类型为 TEXT、NTEXT 和 IMAGE 的字段不能使用 ORDER BY 进行排序。

例【7-15】查询 student 表中全体女学生的情况，要求结果按照年龄降序排列。

T-SQL 语句如下：

```
SELECT * FROM student WHERE ssex='女' ORDER BY sage DESC
```

执行结果如图 7-13 所示。



图 7-13 查询结果排序

## 7.2.4 GROUP BY 子句的使用

对数据进行检索时,经常需要对结果进行汇总统计计算。在 T-SQL 中,通常使用聚合函数和 GROUP BY 子句来实现统计计算。

### 1. 聚合函数

聚合函数用于处理单个列中所选的全部值,并生成一个结果值。

常用的聚合函数如表 7-6 所示。

表 7-6 常用聚合函数

函 数 名 称	说 明
COUNT([DISTINCT ALL] 列名称[*])	统计符合条件的记录的个数
SUM([DISTINCT ALL] 列名称)	计算一列中所有值的总和,只能用于数值类型
AVG([DISTINCT ALL] 列名称)	计算一列中所有值的平均值,只能用于数值类型
MAX([DISTINCT ALL] 列名称)	求一列值中的最大值
MIN([DISTINCT ALL] 列名称)	求一列值中的最小值

说明:

如果使用 DISTINCT,则表示在计算时去掉重复值,而 ALL 则表示对所有值进行运算,默认值为 ALL。

例【7-16】统计查询 student 表中学生的总人数。

T-SQL 语句如下:

```
SELECT COUNT(*) FROM student
```

例【7-17】查询选修 01 课程学生的最高分,最低分和平均分。

T-SQL 语句如下:

```
SELECT MAX(grade) as '01 课程最高分',MIN(grade) as '01 课程最低分',AVG(grade) as '01 课程平均分'
FROM sc WHERE cno='01'
```

以上两例的执行结果如图 7-14 所示。

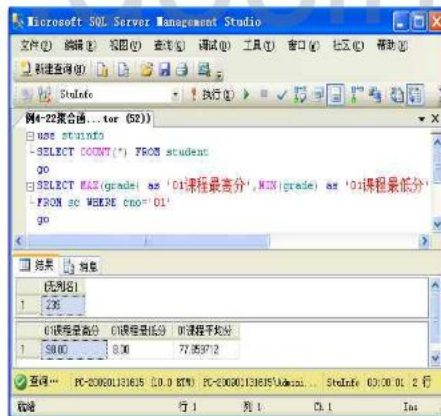


图 7-14 使用聚合函数



在 T-SQL 中, 连接查询有两大类实现形式: 一类是使用等值连接形式, 另一类是使用关键字 JOIN 连接形式。

### 1. 等值连接

等值连接的连接条件是在 WHERE 子句中给出的, 只有满足连接条件的行才会出现在查询结果中。这种形式又称为连接谓词表示形式, 是 SQL 语言早期的连接形式。

等值连接的连接条件格式如下:

表名 1. 字段名 1 = 表名 2. 字段名 2

例【7-20】从 student 表和 sc 表中, 查询所有不及格的学生的学号、学生姓名、所属院系、所选的课程号和成绩。

T-SQL 语句如下:

```
SELECT student.sno,sname,sdept,cno,grade
FROM student,sc
WHERE student.sno=sc.sno and grade<60
```

执行结果如图 7-16 所示。

说明:

- 本例中, WHERE 子句既有查询条件(Grade<60), 又有连接条件(student.Sno=sc.Sno);
- 连接条件中的两个字段称为连接字段, 它们必须具有一致的数据类型。如本例中连接字段分别为 student 表的 sno 字段和 sc 表的 sno 字段;
- 在单表查询中, 所有的字段都来自于同一张表, 故在 SELECT 语句中不需要特别说明。但是在多表查询中, 有的字段(如 sno 字段)在几个表中都出现了, 引用时就必须说明其来自哪个表, 否则就可能引起混乱, 造成语法错误;
- 连接条件中使用的比较运算符可以是 <、<=、=、>、>=、!=、<>、!<和!>。当比较运算符为“=”时, 就是等值连接。

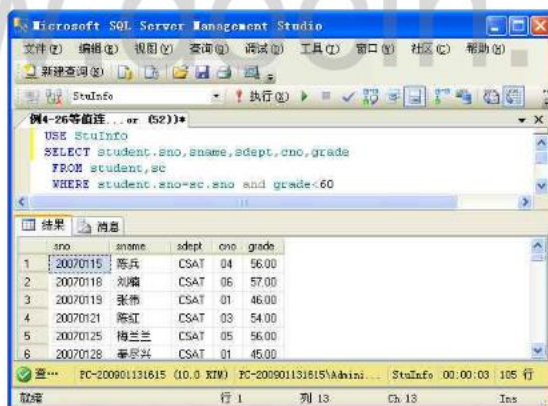


图 7-16 等值连接



## 2. JOIN 关键字连接多个表

T-SQL 扩展了连接的形式,引入了 JOIN...ON 关键字连接形式,从而使表的连接运算能力得到了增强。

JOIN...ON 关键字放在 FROM 子句中,其命令格式如下:

```
FROM <表名 1> [INNER] | {LEFT|RIGHT|FULL} [OUTER] JOIN <表名 2> ON <连接条件>
```

这种连接形式通过 FROM 给出连接类型,用 JOIN 表示连接,用 ON 短语给出连接条件。

JOIN 提供了多种类型的连接方法:内连接、外连接和交叉连接。它们之间的区别在于:从相互关联的不同表中选择用于连接的行时所采用的方法不同。下面分别介绍这几种连接的使用方法。

- 内连接 INNER 查询

内连接是最常见的一种连接,又称为普通连接或自然连接,它是系统默认的形式,在实际使用中可以省略 INNER 关键字。

例【7-21】上例也可以改写成如下形式实现。

```
SELECT student.sno,sname,sdept,cno,grade
FROM student JOIN sc ON student.sno=sc.sno
WHERE grade<60
```

使用 JOIN...ON 连接词替换了上例中 WHERE 子句中的连接条件。内连接与等值连接效果相同,仅当两个表中都至少有一行符合连接条件时,内连接才返回行。

- 外连接 OUTER 查询

外连接是指连接关键字 JOIN 后面表中指定列连接在前一表中指定列的左边或者右边,如果两个表中指定列没有匹配行,则返回空值。

外连接的结果不但包含满足连接条件的行,还包含相应表中的所有行。外连接有 3 种形式,其中的 OUTER 可以省略:

(1) 左外连接(LEFT OUTER JOIN 或 LEFT JOIN): 包含左边表的全部行(不管右边的表中是否存在与它们匹配的行),以及右边表中全部满足条件的行。

(2) 右外连接(RIGHT OUTER JOIN 或 RIGHT JOIN): 包含右边表的全部行(不管左边的表中是否存在与它们匹配的行),以及左边表中全部满足条件的行。

例【7-22】分别用左外连接和右外连接查询 student 表和 sc 表中的学生的 Sno、Cno、Sname 和 Grade。比较查询结果的区别并分析。

左外连接 T-SQL 语句如下:

```
SELECT student.sno,cno,sname,grade
FROM student LEFT JOIN sc ON sc.sno=student.sno
```

右外连接 T-SQL 语句如下:



```

SELECT student.sno,cno,sname,grade
FROM student RIGHT JOIN sc ON sc.sno=student.sno

```

执行结果如图 7-17 所示。

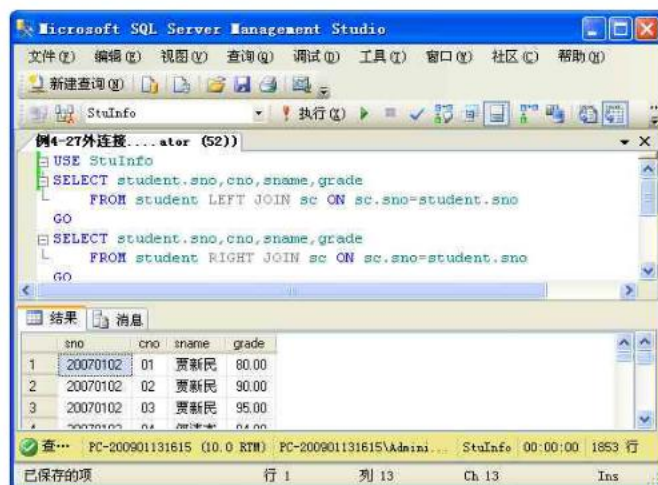


图 7-17 外连接

可以看到, 两者的运行结果不完全相同, 左连接以 student 表为准, 包含 student 表中的所有数据, 而只在 sc 表中存在的数据将不出现在查询结果中; 右连接以 sc 表为准, 在 student 表中不可能出现的 Sno 为 NULL 的数据, 在查询结果中就显示出来了。

(3) 全外连接(FULL OUTER JOIN 或 FULL JOIN): 包含左、右两个表的全部行, 不管另外一边的表中是否存在与它们匹配的行, 即全外连接将返回两个表的所有行。

在现实生活中, 参照完整性约束可以减少对全外连接的使用, 一般情况下, 左外连接就足够了。但当在数据库中没有利用清晰、规范的约束来防范错误数据的情况下, 全外连接就变得非常有用, 可以用它来清理数据库中的数据。

#### • 交叉连接(CROSS JOIN)

交叉连接即两个表的笛卡儿积, 其返回结果是由第一个表的每行与第二个表的所有行组合后形成的表, 因此, 数据行数等于第一个表中符合查询条件的数据行数乘以第二个表中符合查询条件的数据行数。交叉连接关键字 CROSS JOIN 后不跟 ON 短语引出的连接条件。

例【7-23】交叉连接 student 和 sc 两表, 查看新表的行数。

T-SQL 语句如下:

```

SELECT * FROM student
SELECT * FROM sc
SELECT student.*,sc.* FROM student CROSS JOIN sc

```

执行结果如图 7-18 所示, 可以看到, student 表有 239 行, sc 表有 929 行, 因此, 两表相乘有 222031 行。



图 7-18 交叉连接

## 7.2.6 子查询

在 SELECT 查询语句中，子查询也称为嵌套查询，是一个嵌套在 SELECT 语句中的查询语句。处于内层的查询称为子查询，处于外层的查询称为父查询。任何允许使用表达式的地方都可以使用子查询。T-SQL 语句支持子查询，正是 SQL 结构化的具体体现。

子查询 SELECT 语句必须放在括号中，使用子查询的语句实际上执行了两个连续查询，而且查询的结果作为第二个查询的搜索值。可以用子查询来检查或者设置变量和列的值，或者用子查询来测试数据行是否存在于 WHERE 子句中。需要注意的是：ORDER BY 子句只能对最终查询结果排序，即在子查询中的 SELECT 语句中不能使用 ORDER BY 子句。

### 1. 使用 IN 关键字的子查询

由于子查询的结果是记录的集合，故常用谓词 IN 来实现。

IN 谓词用于判断一个给定值是否在子查询的结果集中。当父查询表达式与子查询的结果集中的某个值相等时，返回 TRUE，否则返回 FALSE。同时，还可以在 IN 关键字之前使用 NOT，表示表达式的值不在查询结果集中。

例【7-24】查询至少有一门课程不及格的学生信息。

T-SQL 语句如下：

```
SELECT * FROM student WHERE sno IN
    (SELECT sno FROM sc WHERE grade<60)
```

执行结果如图 7-19 所示。

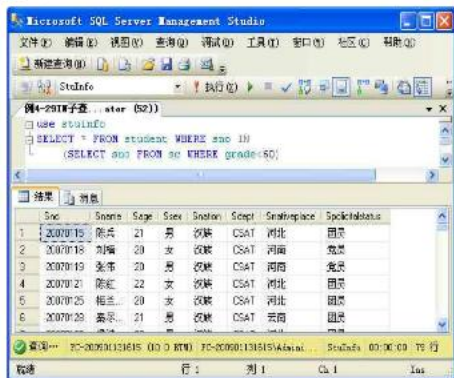


图 7-19 使用 IN 的子查询

在执行包含子查询的 SELECT 语句时,系统先执行子查询,产生一个结果表。在本例中,系统先执行子查询,得到所有不及格学生的 sno,再执行父查询,如果 student 表中某行的 sno 值等于子查询结果表中的任意一个值,则该行就被选择。

## 2. 使用比较运算符的子查询

使用带有比较运算符的子查询,是当用户能够确切知道子查询返回的是单值时,可以在父查询的 WHERE 子句中,使用比较运算符进行比较查询。这种查询可以认为是 IN 子查询的扩展。

例【7-25】从 sc 表中查询王小华同学的考试成绩信息,显示 sc 表的所有字段。

T-SQL 语句如下:

```
SELECT * FROM sc WHERE sno=
(SELECT sno FROM student WHERE sname=N'王小华')
```

执行结果如图 7-20 所示,我们通过如下查询语句:



图 7-20 使用比较运算符的子查询

```
SELECT * FROM student WHERE sname=N'王小华'
```

能够验证,查询结果显示的是王小华同学的考试成绩信息。

## 3. 带有 ANY、SOME 或 ALL 关键字的子查询

使用 ANY、SOME 或 ALL 关键字对子查询进行限制。

ALL 代表所有值,ALL 指定的表达式要与子查询结果集中的每个值都进行比较,当表达式与每个值都满足比较的关系时,才返回 TRUE,否则返回 FALSE。

SOME 或 ANY 代表某些或者某个值,表达式只要与子查询结果集中的某个值满足比较的关系时,就返回 TRUE,否则返回 FALSE。

例【7-26】查询考试成绩比王小华同学高的学生信息。

在上题的基础上,我们进一步进行查询嵌套:如果使用 ANY,则查询结果是只要比王小华同学任一成绩高的学生信息;使用 ALL,则查询结果是比王小华同学所有的成绩都要高的学生信息。



T-SQL 语句如下:

```
SELECT * FROM sc WHERE grade >ALL
(SELECT grade FROM sc WHERE sno=
(SELECT sno FROM student WHERE sname=N'王小华'))

go

SELECT * FROM sc WHERE grade >ANY
(SELECT grade FROM sc WHERE sno=
(SELECT sno FROM student WHERE sname=N'王小华'))

go
```

执行结果如图 7-21 所示, 由上例的查询结果我们知道, 王小华的最高成绩为 98, 最低成绩为 60。因此, 使用 ALL 关键字, 只显示高于 98 分的学生成绩, 只有 4 条记录; 而使用 ANY 关键字, 则显示了所有高于 60 分的学生成绩, 共有 791 条记录。

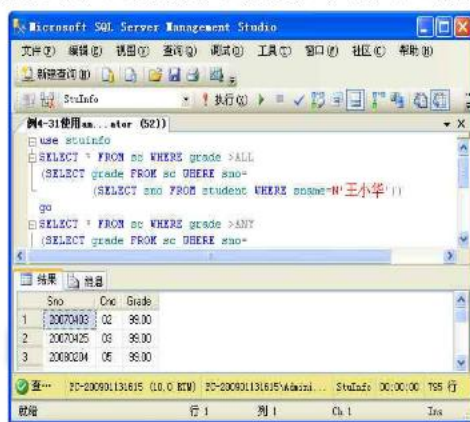


图 7-21 使用 ALL 和 ANY 的查询

#### 4. 使用 EXISTS 的子查询

EXISTS 称为存在量词, 在 WHERE 子句中使用 EXISTS, 表示当子查询的结果非空时, 条件为 TRUE, 反之则为 FALSE。EXISTS 前面也可以使用 NOT, 表示检测条件为“不存在”。

EXISTS 语句与 IN 非常类似, 它们都根据来自子查询的数据子集来测试列的值。不同之处在于, EXISTS 使用联接将列的值与子查询中的列联接起来, 而 IN 不需要联接, 它直接根据一组以逗号分隔的值进行比较。

例【7-27】查询没有选修 01 课程的学生信息。

T-SQL 语句如下:

```
SELECT * FROM student WHERE NOT EXISTS
(SELECT * FROM sc WHERE sno=student.sno AND cno='01')
```

执行结果如图 7-22 所示。

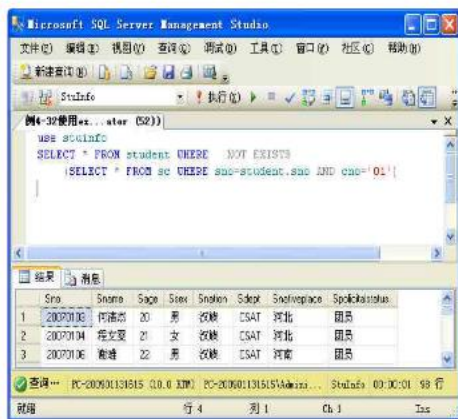


图 7-22 使用 EXISTS 的查询

## 7.2.7 SELECT 语句的其他子句

除了上面介绍的 SELECT 语句的主要子句之外, SELECT 还有 2 个常用的子句: INTO 子句和 UNION 子句, 使用它们可以实现更为完备的功能。

### 1. INTO 子句

SELECT 语句使用 INTO 子句来表明查询结果的去向。如果需要将查询得到的结果存入新的数据表中, 就需要使用 INTO 语句: INTO <新表名>, 来创建新表, 存储记录。

例【7-28】将 student 表中所有年龄为 19 岁的学生信息存入新表 age19 中。

T-SQL 语句如下:

```
SELECT * INTO age19 FROM student WHERE sage=19
```

```
GO
```

```
SELECT * FROM age19
```

执行结果如图 7-23 所示。

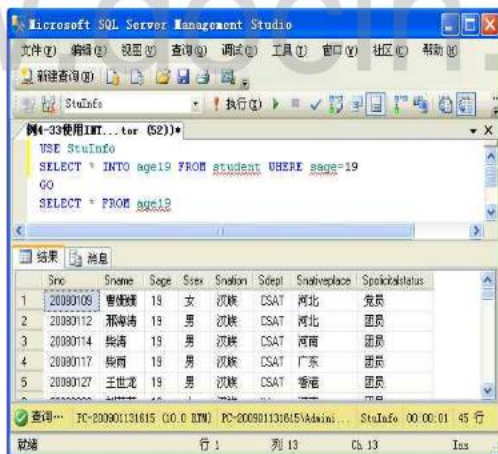


图 7-23 使用 INTO 子句



## 2. UNION 子句

T-SQL 支持集合的并(UNION)运算, 执行联合查询。需要注意的是: 参与并运算操作的两个查询语句, 其结果应该具有相同的字段个数, 以及相同的对应字段的数据类型。

默认情况下, UNION 将从结果集中删除重复的行。如果使用 ALL 关键字, 那么结果中将包含所有行而不删除重复的行。

例【7-29】查询 CSAT 专业的女学生和 IM 专业的男学生信息。

T-SQL 语句如下:

```
SELECT * FROM student where sdept='CSAT' and ssex='女'  
UNION  
SELECT * FROM student where sdept='IM' and ssex='男'
```

执行结果如图 7-24 所示。

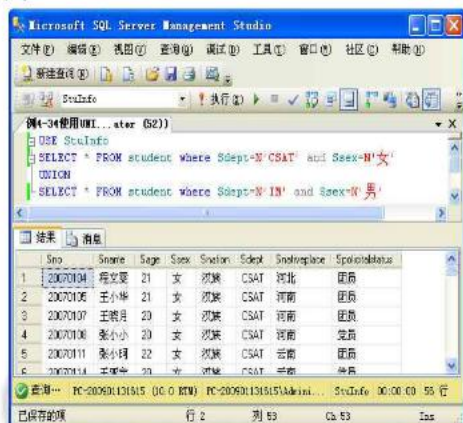


图 7-24 使用 UNION 子句

## 7.3 视图

如果我们经常需要从多个表中获取特定列的数据, 并需要将这些数据组织在一起使用, 有没有什么好办法能够完成这项工作呢? 答案就是视图。

### 7.3.1 视图概述

视图是一种数据库对象, 它是从一个或多个表或视图中导出的虚表, 即它可以从一个或多个表中的一个或多个列中提取数据, 并按照表的组成行和列来显示这些信息。

视图与真实的表也有很多类似的地方。例如, 视图也是由若干个字段(列)和一些记录组成的, 在某些条件满足时, 还可以通过视图来插入、更改和删除数据。当对通过视图看到的数据进行修改时, 相应的基本表中的数据也会发生改变, 同时, 若本表中的数据发生改变时, 这种改变也会自动反映到视图中。

视图在数据库中存储的是视图的定义, 而不是查询的数据。通过这个视图的定义, 对视图的查询最后转化为对基本表的查询。

视图中的数据并不是实际地以视图结构存储在数据库中的,而是在视图所引用的表中。视图被定义后,便存储在数据库中,通过视图看到的数据只是存放在数据库表中的数据,其结构和数据都是建立在对表的查询的基础上的。

使用视图不仅可以简化数据操作,还可以提高数据库的安全性,使用视图有很多优点,这些优点如下:

(1) 视图是作为一个数据库对象而存在数据库中的,便于管理和维护,而且视图象表一样还可以用在查询语句中,从而简化了检索数据的操作。

(2) 可以定制允许用户查看哪些数据,让用户通过视图来访问表中的特定字段和记录,而不对用户授予直接访问数据库表的权限。

(3) 可以针对不同的用户定义不同的视图,在用户视图上不包括机密数据字段,从而自动提供对机密数据的保护。

(4) 可以使用视图将数据导出到其他的应用程序。

(5) 允许用户以不同的方式查看数据,即使在用户同时使用相同的数据时也可如此。

视图是从现有基表中抽取若干子集组成的用户“特殊表”。定义一个视图时,只是把其定义存放在系统数据中,而并不直接存储视图对应的数据,直到用户使用视图时才去取得对应的数据。

视图可以根据其所包含的内容灵活的命名。在定义了一个视图之后,就可以把它当作表来引用。虽然视图可以作为数据库对象存储在磁盘上,但它并不创建所包含的行和列的永久复制。在每次接受访问时,视图都是从基表中提取所包含的行与列,所以,视图永远依赖于基表。当对视图进行操作时,也可能影响到基表。当通过视图修改数据时,实际上是在改变基表中的数据。相反地,基表数据的改变也会自动反映在由基表产生的视图中。由于逻辑上的原因,有些视图可以修改对应的基表,有些则不能。视图也可以用作安全机制,允许用户通过视图访问数据,但不授予用户直接访问基表的权限。

使用视图的优点和作用主要体现在:视图让用户能够只关心他感兴趣的某些特定数据和所负责的一些特定任务,而那些不需要的或者无用的数据则不在视图中显示;视图可以简化用户对数据的操作方式;可以将经常使用的连接、投影和选择查询定义为视图,在每次执行相同的查询时,不必重新编写这些复杂的查询语句,只要一条简单的查询视图语句即可;视图可以让不同的用户以不同的方式看到不同或者相同的数据集;有时由于表中的数据量很大,在设计时可以将表进行水平或垂直分割,但表的结构变化会对应用程序产生不良影响,而使用视图则可以重新组织数据,使外模式保持不变,原有的应用程序仍可以通过视图来重载数据;视图还增强了数据的安全性,通过视图用户只能查看和修改他们有权限或所能看到的数据,其他数据或表既不可见也不可访问。

### 7.3.2 视图的创建

在 SQL Server 2008 系统中,只能在当前数据库中创建视图,创建视图时,SQL Server 会首先验证视图定义中所引用的对象是否存在。视图的名称必须符合命名规则。因为视图的外形和表的外形是一样的,所以在给视图命名时,要使用一种能与表区分开的命名机制,使人很容易分辨出视图与表,如在视图名称之前使用 V\_ 作为前缀。

创建视图时应该注意如下几点：必须是 sysadmin、db\_owner、db\_ddladmin 角色的成员，或拥有创建视图权限；只能在当前数据库中创建视图，在视图中最多只能引用 1024 列；如果视图引用的基表或者视图被删除，则该视图将不能再被使用；如果视图中的某一列是函数、数学表达式、常量或者与来自多个表的列名相同，则必须为列定义名称；不能在规则、默认、触发器的定义中引用视图；当通过视图查询数据时，SQL Server 要检查以确保语句中涉及的所有数据库对象存在；视图的名称必须遵循标识符的规则，是唯一的。

创建视图的方法主要有两种：一是在 SQL Server Management Studio(SSMS)中使用现有命令和功能，通过图形化工具进行创建；二是通过 T-SQL 语句中的 CREATE VIEW 命令进行创建。本节将对这两种创建视图的方法分别进行阐述。

### 1. 使用 SQL Server Management Studio 创建视图

(1) 在“开始”菜单中选择“程序”|Microsoft SQL Server 2008|SQL Server Management Studio 命令，打开 SQL Server Management Studio 窗口，并使用 Windows 或 SQL Server 身份验证建立连接。

(2) 在“对象资源管理器”中展开服务器，然后展开“数据库”节点，双击“AdventureWorks”数据库将其展开。

(3) 右击“视图”节点，从弹出的快捷菜单中选择“新建视图”命令，如图 7-25 所示。

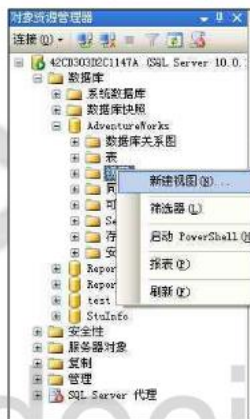


图 7-25 “新建视图”命令

(4) 打开“视图设计器”窗口，并弹出“添加表”对话框，如图 7-26 所示。



图 7-26 “添加表”对话框



(5) 选择要定义的视图所需的表、视图、函数后,通过单击字段左边的复选框选择需要的字段,如图 7-27 所示。

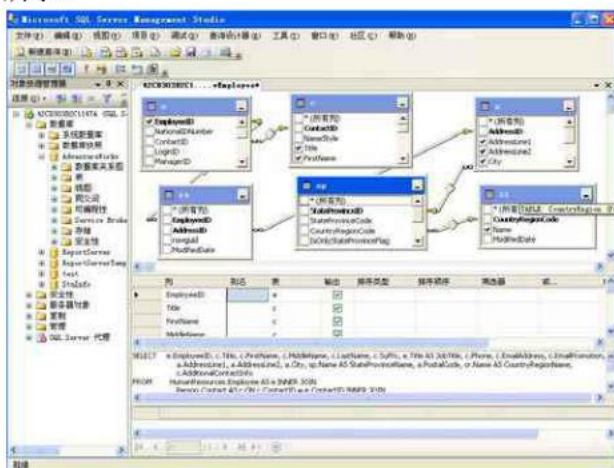


图 7-27 “视图”设计窗口

(6) 单击工具栏中的“保存”按钮,或者从菜单中选择保存选项保存视图,输入视图名,即可完成视图的创建。

## 2. 使用 Transact-SQL 语句中的 CREATE VIEW 命令创建视图

使用 CREATE VIEW 语句创建视图的基本语法格式如下:

```
CREATE VIEW [ schema_name.] view_name [ (column [,... ,n] ) ]
[WITH [ENCRYPTION] [SCHEMABINDING] [VIEW_METADATA] ]
AS select_statement [ ; ]
[ WITH CHECK OPTION ]
```

需要注意的是:在视图定义中,SELECT 子句中不能包含下列内容: COMPUTE 或 COMPUTE BY 子句; INTO 关键字; ORDER BY 子句,除非 SELECT 语句中的选择列表中有 TOP 子句; OPTION 子句; 引用临时表或表变量。

例【7-30】选择表 s 和 sc 中的部分字段(name,age,sex,cno,score)和记录来创建一个视图,限制表 s 中的记录是计算机系的记录集合,视图定义为 view\_s。

T-SQL 语句如下:

```
CREATE VIEW view_s
AS
SELECT s.name,s.age,s.sex,sc.cno,sc.score
FROM s,sc
WHERE s.sno=sc.sno AND s.dept='计算机';
```

### 7.3.3 视图的修改、查看和重命名

创建完视图之后,就可以对视图进行修改、查看和重命名等操作。

## 1. 视图的修改

### (1) 使用 SSMS 图形化界面修改视图

在 SSMS 图形化界面中, 右击要修改的视图, 从弹出的快捷菜单中选择“修改”命令, 出现原视图。该视图与创建视图时相同, 可以按照创建视图的方法修改视图。

### (2) 使用 T-SQL 语句修改视图

首先必须拥有使用视图的权限, 然后才能使用 ALTER VIEW 语句, 该语句的语法格式如下:

```
ALTER VIEW [ schema_name.]view_name [ (column [ ,... ,n ] ) ]
[WITH [ENCRYPTION] [SCHEMABINDING] [VIEW_METADATA] ]
AS select_statement [ ; ]
[ WITH CHECK OPTION ]
```

例【7-31】修改视图 V\_employees(number,name,age), 在该视图中增加一个新的字段 employees.salary, 并且定义一个新的字段名称 e\_salary。

T-SQL 语句如下:

```
ALTER VIEW dbo.V_employees(number,name,age,e_salary)
AS
SELECT number,name,age,salary
FROM employees;
```

## 2. 查看视图信息

### (1) 使用 SSMS 图形化界面查看视图

在 SSMS 中, 右击某个视图的名称, 从弹出的快捷菜单中选择“选择前 1000 行”或“编辑前 200 行”命令, 在 SQL Server 图形化界面中就会显示该视图输出的相应数据, 如图 7-28 所示。

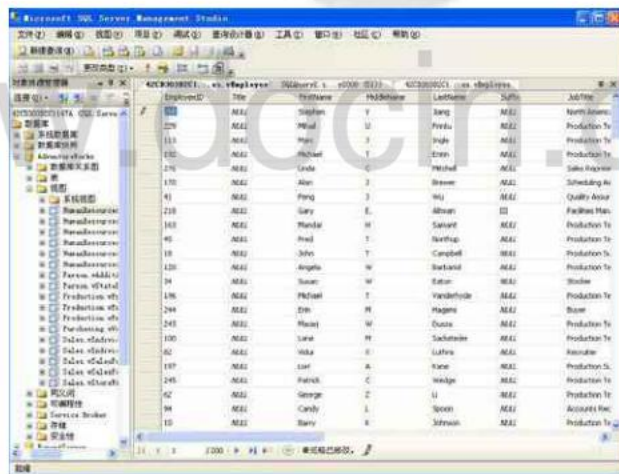


图 7-28 “编辑前 200 行”输出数据

### (2) 捕获视图信息

在 sys.Views 视图中, 每个视图的对象在该 sys.Views 中对应一行数据。可以使用 sys.Views



如在 AdventureWorks 数据库中查看所有视图信息，可以输入如下 T-SQL 语句：

执行结果如图 7-29 所示。

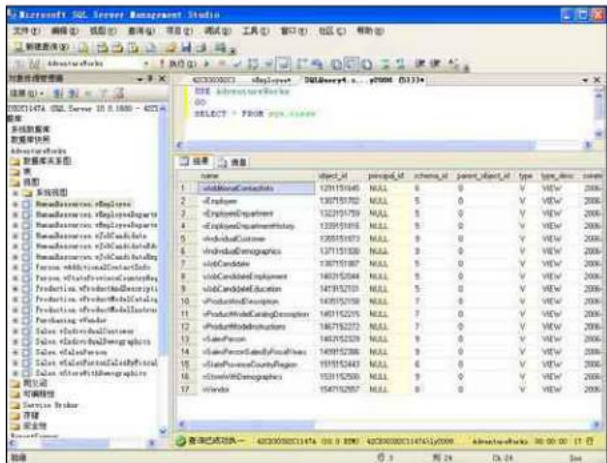


图 7-29 获取视图信息

```
USE AdventureWorks
GO
SELECT * FROM sys.all_sql_modules
```

执行结果如图 7-30 所示。

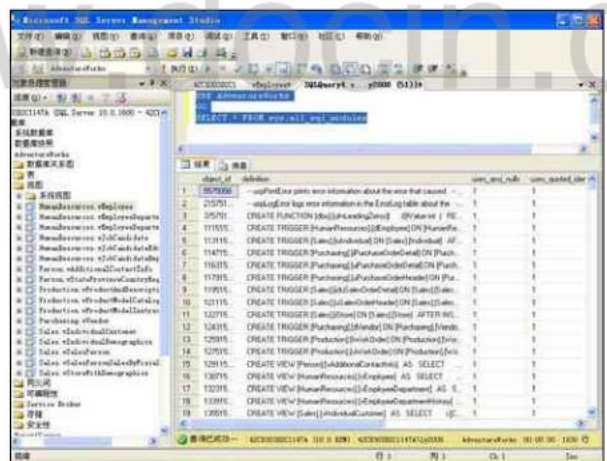


图 7-30 获取视图定义信息

### 3. 重命名视图

#### (1) 使用 SSMS 图形化界面重命名视图

在 SSMS 中, 选择要修改名称的视图, 并右击该视图, 从弹出的快捷菜单中选择“重命名”命令; 或者在视图上再次单击, 也可以修改视图的名称。此时, 该视图的名称将变成可输入状态, 可以直接输入新的视图名称。

#### (2) 使用系统存储过程重命名视图

可是使用系统存储过程 `sp_rename` 来重命名视图, 语法格式如下:

```
sp_rename old_name,new_name
```

例【7-32】把视图 `v_all` 重命名为 `v_part`。

T-SQL 语句如下: `sp_rename v_all,v_part`

### 7.3.4 视图的删除

对于不再需要的视图, 可以通过 `DROP VIEW` 语句把视图的定义从数据库中删除。删除视图, 就是删除其定义和赋予它的全部权限。在 `DROP VIEW` 语句中, 可以同时删除多个不再需要的视图。

`DROP VIEW` 语句的基本语法格式如下:

```
DROP VIEW view_name
```

例【7-33】同时删除视图 `v_student` 和 `v_teacher`。

T-SQL 语句如下:

```
DROP view v_student,v_teacher
```

可以使用该命令同时删除多个视图, 只需在要删除的各视图名称之间用逗号隔开即可。

### 7.3.5 通过视图更改记录

在 SQL Server 2008 中, 对视图中的数据进行修改, 其实就是对其基表中的数据进行修改。这是由视图本身的性质决定的, 因为视图就是一个虚拟表, 它并不存储数据, 数据只存于基表中。如果满足一些限制条件, 可以通过视图自由地插入、更新和删除数据。

使用视图修改数据时, 需要注意以下几点: 修改视图中的数据时, 不能同时修改两个或者多个基表; 不能修改那些通过计算得到的字段; 如果在创建视图时指定了 `WITH CHECK OPTION` 选项, 那么使用视图修改数据库信息时, 必须保证修改后的数据满足视图定义的范围; 执行 `UPDATE`、`DELETE` 命令时, 所删除与更新的数据必须包含在视图的结果集中; 当视图引用多个表时, 无法使用 `DELETE` 命令删除数据, 如果使用 `UPDATE` 命令则应与 `INSERT` 操作一样, 被更新的列必须属于同一个表。

通过视图插入、更新与删除数据的步骤如下: 在“开始”菜单中选择“程序”|Microsoft SQL Server 2008|SQL Server Management Studio 命令, 打开 SQL Server Management Studio 窗口, 并使用 Windows 或 SQL Server 身份验证建立连接; 单击“新建查询”按钮, 新建一个查询窗口。可以在该查询窗口中输入相应的语句进行操作。

### 1. 添加数据行

例【7-34】创建一个基于表 `employees` 的新视图 `v_employees`。

T-SQL 语句如下：

```
CREATE VIEW v_employees(number, name, age, sex, salary)
AS
SELECT number, name, age, sex, salary
FROM employees
```

例【7-35】通过视图 `v_employees` 添加一条新的数据行，各列的值分别为 10100、李子言、25、男、3600。

T-SQL 语句如下：

```
INSERT INTO v_employees
Values(10100, '李子言', 25, '男', 3600);
```

### 2. 修改数据行

例【7-36】创建一个基于表 `employees` 的视图 `v_employees`，然后通过该视图修改表 `employees` 中的记录，把名字“张三”改为“张然”。

T-SQL 语句如下：

```
CREATE VIEW v_employees
AS
SELECT * FROM employees
UPDATE v_employees
SET name='张然'
WHERE name='张三';
```

### 3. 删除数据行

例【7-37】利用视图 `v_employees` 删除表 `employees` 中姓名为“李惠”的记录。

T-SQL 如下：

```
DELETE FROM v_employees
WHERE name='李惠';
```

## 7.4 习 题

1. 简述 SELECT 语句的基本语法。
2. 简述 SELECT 语句中的 FROM、WHERE、GROUPBY 以及 ORDERBY 子句的作用。

3. 简述 WHERE 子句可以使用的搜索条件及其意义。
4. 举例说明什么是内连接、外连接和交叉连接？
5. INSERT 语句的 VALUES 子句中必须指明哪些信息，必须满足哪些要求？
6. 用 T-SQL 语句创建数据库“销售管理”，并在其中创建 3 张数据表“商品”，“部门”和“销售”。表结构如下：  
商品(商品号 char(6)，商品名称 nchar(10)，单价 money)  
部门(部门号 char(6)，部门名称 nchar(10)，部门经理 nchar(8)，电话 char(12))  
销售(部门号 char(6)，商品号 char(6)，数量 int，金额 money)
7. 用 T-SQL 语句为上一题中创建的每张表至少添加 10 条记录。
8. 使用数据库 stuinfo，进行如下操作：
  - (1) 将任课教师 Daad 改为 Forster；
  - (2) 增加课程 vc++，课程号为 07，学分为 6，任课教师为 Emily；
  - (3) 删除学生贾新民的相关记录；
  - (4) 查询所有学生的姓名及年龄；
  - (5) 查询所有考试不及格的学生的学号、姓名和成绩；
  - (6) 查询年龄在 20-22 岁之间的学生姓名、年龄、所属院系和政治面貌；
  - (7) 查询所有姓李的学生的学号、姓名和性别；
  - (8) 查询名字中第 2 个字为“华”字的女学生的姓名、年龄和所属院系；
  - (9) 查询所有选了 3 门课以上的学生的学号、姓名、所选课程名称及成绩；
  - (10) 查询每个同学各门课程的平均成绩和最高成绩，并按照降序排列输出学生姓名、平均成绩和最高成绩；
  - (11) 查询所有学生都选修了的课程号和课程名。

# 第9章 数据完整性

为了防止数据库中出现不符合语义的数据，维护数据的完整性，数据库管理系统提供了一种机制来检查数据库中的数据是否满足语义规定的条件，加在它们之上的语义约束条件就是数据库语义约束条件。

数据的完整性是指数据库中数据的一致性与正确性。在 SQL Server 2000 中，可以通过约束、默认、规则、触发器等来达到保证数据完整性的目的。本章将介绍如何通过约束、默认与规则实现数据的完整性。

**本章的学习目标：**

- 操纵数据时的问题和解决机制
- 约束的基本概念和类型
- 管理 DEFAULT 约束
- 管理 CHECK 约束
- 管理主键约束
- 管理 UNIQUE 约束
- 管理外键约束
- 禁止约束的场景

## 9.1 数据完整性概述

数据完整性的含义包括以下内容：

- (1) 数值的完整性，是指数据类型与取值的正确性。
- (2) 表内数据不相互矛盾。
- (3) 表间数据不相互矛盾，指数据的关联性不被破坏。

数据完整性有不同的分类方法。

- (1) 按照数据完整性的功能可以将其分为 4 类：

实体完整性，要求表中每一条记录(每一行数据)是唯一的，即它必须至少拥有一个唯一标识以区分不同的数据行。实现方法有：主键约束 PRIMARY KEY、唯一性约束 UNIQUE、唯一索引 UNIQUE INDEX、标识 IDENTITY 等。



- 值域完整性, 限定表中输入数据的数据类型与取值范围。实现方法: 默认值约束 DEFAULT 或默认对象、核查约束 CHECK、外键约束 FOREIGN KEY、规则 RULE、数据类型、非空性约束 NOT NULL 等。
  - 引用完整性, 指对数据库进行添加、删除、修改数据时, 要维护表间数据的一致性。实现方法: 外键约束 FOREIGN KEY、核查约束 CHECK、触发器 TRIGGER、存储过程 PROCEDURE。
  - 用户定义的完整性, 用于实现用户特殊要求的数据规则或格式。实现方法: 默认值 DEFAULT、核查约束 CHECK、规则 RULE 等。
- (2) 按照在数据库中实现数据完整性的作用级别, 可以将其分为对象级与总体级。
- 对象级: 作用范围是数据库的某个表对象, 是在定义表的同时定义的, 删除表后则失去作用, 如约束。
  - 总体级: 是作为数据库的对象单独定义的, 因此单独存在于数据库中, 需要时可以绑定到数据库的表或用户定义的数据类型中, 如默认对象、规则。
- (3) 按照数据完整性的实施方法可分为: 约束、默认、规则、触发器、存储过程等。

## 9.2 使用约束实施数据的完整性

约束用于实现表内数据的完整性, 属于对象级。SQL Server 支持的约束有如下几种: 非空约束 NOT NULL、主键约束 PRIMARY KEY、唯一约束 UNIQUE、核查约束 CHECK、外键约束 FOREIGN KEY、默认值约束 DEFAULT。

约束可以在创建表时定义, 也可以在修改表时定义(即向已有的表中添加约束), 但同一个数据库中不同约束的名称不能相同。

按约束的作用范围可以分为两类, 在定义列时定义的约束, 仅作用于本列, 称为列级约束; 表中单独定义的约束, 称为表级约束。表级约束的声明与列的定义无关, 当一个约束作用于一个以上的列时, 必须使用表级约束。

SQL Server 2008 的表定义支持以下 4 个完整性约束, 分别是:

- 实体完整性: 将行定义为特定表的唯一实体, 可以通过设定主键、唯一、标识列等方法来实现;
- 域完整性: 它是对某列上可以使用的有效值的限制, 也称为列完整性, 即确定一个数据集对某一列是否有效和确定是否允许空值;
- 参照完整性: 它是用来维护相关数据表之间数据一致性的手段, 可使用外键、检查等来实现;
- 用户自定义完整性: 是由用户根据实际应用中的需要自行定义。

SQL Server 中的完整性约束如表 9-1 所示。

表 9-1 SQL Server 中约束

完整性类型	约 束 类 型	描 述
域	DEFAULT	指定列的默认值
	CHECK	指定列的允许值
	FOREIGN KEY	指定必须存在值的列
	NULL	指定是否允许为 NULL
实体	PRIMARY KEY	唯一标识每一行
	UNIQUE	防止非主键重复
引用	FOREIGN KEY	定义值与同一个表或另一个表的主键值匹配的一列或多列组合
	CHECK	指定根据同一个表中其他列的值可在列中接受的数据值

### 9.3 约束的概念和类型

约束是 SQL Server 提供的自动保持数据库完整性的一种方法，它通过限制字段中数据、记录中数据、表之间的数据来保证数据的完整性。

在 SQL Server 中，对于基本表的约束分为列约束和表约束。列约束是对某一个特定列的约束，包含在列定义中，直接跟在该列的其他定义之后，用空格分隔，不必指定列名；表约束与列定义相互独立，不包括在列定义中，通常用于对多个列一起进行约束，与列定义用“，”分隔，定义表约束时必须指出要约束的列的名称。

完整性约束的基本语法格式如下：

[CONSTRAINT constraint\_name(约束名)] <约束类型>

注意：

约束不指定名称时，系统会给定一个名称。

在 SQL Server 2008 中，共有 6 种约束：主键约束(primary key constraint)、唯一性约束(unique constraint)、检查约束(check constraint)、默认约束(default constraint)、外部键约束(foreign key constraint)和空值(NULL)约束。

### 9.4 管 理 约 束

本节详细研究 DEFAULT、CHECK、主键、UNIQUE、外键等约束的特点、创建方式、修改等内容。

#### 9.4.1 PRIMARY KEY 约束

一个表通常可以通过一个列或多个列组合的数据来唯一标识表中的每一行，这个列或列

的组合就称为表上的主键。创建表中的主键是为了保证数据的实体完整性。PRIMARY KEY (主键)约束用于定义基本表的主键,它是唯一确定表中每一条记录的标识符,其值不能为 NULL,也不能重复,以此来保证实体的完整性。

PRIMARY KEY 约束与 UNIQUE 约束类似,通过建立唯一索引来保证基本表在主键列取值的唯一性,但它们之间又存在着很大的区别:首先,一个表只能有一个 PRIMARY KEY 约束,但可以定义多个 UNIQUE 约束;其次,对于指定为 PRIMARY KEY 的一个列或多个列的组合,其中任何一个列都不能出现空值,而对于 UNIQUE 所约束的唯一键,则允许为空。

注意:

不能为同一个列或一组列既定义 UNIQUE 约束,又定义 PRIMARY KEY 约束; PRIMARY KEY 既可用于列约束,也可用于表约束。

通常在表中将一个列或多个列组合的数据设置成具有各不相同的值,以便能唯一地标识表中的每一行。这样的一系列或多列称为表的主键,通过它可以强制实现表的实体完整性,消除表的冗余数据。

主键具有如下特性:

- (1) 不重复性
- (2) 非空性
- (3) 唯一性

创建主键的方法有两种:使用 SSMS 操作法和 Transact-SQL 语句操作法。

- 使用 SSMS 图形化界面创建主键约束

在“对象资源管理器”窗口中,展开“数据库”节点下某一具体数据库,展开“表”节点,右键单击要创建主键的表,从弹出的快捷菜单中选择“设计”命令,这时“文档”窗口中将打开“表设计器”页,可对表进行进一步定义;选中表中的某列,单击鼠标右键,从弹出的快捷菜单中选择“设置主键”命令即可为表设置主键,如图 9-1 所示。

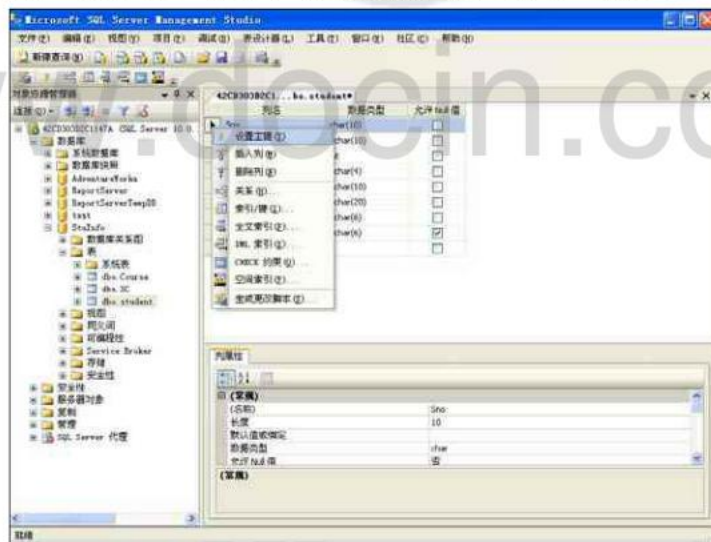


图 9-1 设置主键约束

- 创建表时定义主键约束

创建或更改表时可通过定义 PRIMARY KEY 约束来创建主键。

(1) 定义列级主键的语法

命令格式如下：

```
CREATE TABLE table_name
( column_name data_type
[ DEFAULT default_expression ][ IDENTITY [( seed , increment ) ] ]
[ [ CONSTRAINT constraint_name ]
PRIMARY KEY [ CLUSTERED | NONCLUSTERED ]
][,... n]
)
```

命令说明：

DEFAULT 为默认值约束的关键字,用于指定其后的 default\_expression 为默认值表达式。

IDENTITY [( seed,increment)]表示该列为标识列或称自动编号列。

CONSTRAINT constraint\_name 为可选项,关键字 CONSTRAINT 用于指定其后面的约束名称 constraint\_name。如省略本选项,则系统自动给出一个约束名。建议选择约束名以便于识别。

PRIMARY KEY 表示该列具有主键约束。

CLUSTERED | NONCLUSTERED 表示建立聚簇索引或非聚簇索引,省略此选项则系统默认为聚簇索引。如果没有特别指定本选项,且没有为其他 UNIQUE 唯一约束指定聚簇索引,则默认对该 PRIMARY KEY 约束使用 CLUSTERED。

(2) 定义表级主键

表级主键一般位于表定义中所有列定义之后,与列定义可以用逗号相隔,语法格式如下。

```
CREATE TABLE table_name
( column_name data_type [... n]
[ [ CONSTRAINT constraint_name ]
PRIMARY KEY [ CLUSTERED | NONCLUSTERED ]( column_name [... n] )
]
)
```

命令说明：

( column\_name [...n] )表示该表级主键可以作用于组合在一起的多列所构成的列组合。

- 更改表的主键约束

(1) 在现有表中添加一列,同时将其设置为主键,要求表中原先没有主键。

命令格式如下：

```
ALTER TABLE table_name
ADD column_name data_type
```

```
[DEFAULT default__expression] |[ IDENTITY [ ( seed , increment ) ] ]  
[ CONSTRAINT constraint_name ]  
PRIMARY KEY [ CLUSTERED | NONCLUSTERED ]
```

命令说明:

ALTER TABLE 只允许添加可包含空值或指定了 DEFAULT 定义的列。因为主键不能包含空值,所以需要指定 DEFAULT 定义,或指定 IDENTITY。其他说明与创建主键约束类似。

(2) 使表中现有的一列(或列组合)成为主键,要求表中原先没有主键。且备选主键列中的已有数据不得重复或为空。

命令格式如下:

```
ALTER TABLE table_name  
[ WITH CHECK | WITH NOCHECK ]  
ADD [ CONSTRAINT constraint_name ]  
PRIMARY KEY [ CLUSTERED | NONCLUSTERED ] (column_name [... n])
```

命令说明:

WITH CHECK 为默认选项,该选项表示将使用新的主键约束来检查表中已有数据是否符合主键条件;如果使用了 WITH NOCHECK 选项,则不进行检查。

ADD 指定要添加的约束。

(3) 将表的主键由当前列换到另一列。

一般先删除主键,然后在另一列上添加主键。

(4) 删除主键约束语法。

命令格式如下:

```
ALTER TABLE table_name  
DROP [CONSTRAINT] primarykey_name
```

命令说明:

primarykey\_name 表示要删除的主键名称,该名称是建立主键时定义的。如果建立主键时没有定义名称,则这里必须输入建立主键时系统自动给出的随机名称。

例【9-1】建立一个S表,包括: SNO、SNAME 和 SSEX 列,定义 SNO 为 S 表的主键。

T-SQL 语句如下:

```
CREATE TABLE S  
(SNO CHAR(5) PRIMARY KEY,  
SNAME CHAR(10) NOT NULL,  
SSEX CHAR(5))
```

## 9.4.2 UNIQUE 约束

一个表只能有一个主键,如果有多列或多个列组合需要实施数据唯一性,则可以采用唯



一约束。唯一约束与主键约束的主要区别在于：唯一约束用于非主键列，使之满足数据唯一性要求；唯一约束允许 NULL 值，而主键不允许；唯一约束可以在多列或多个列组合上分别设置，而主键只能在一列或一个列组合上设置。

唯一性约束用于指定一个或多个列的组合值具有唯一性，以防止在列中输入重复的值。定义了 UNIQUE 约束的那些列称为唯一键，系统自动为唯一键建立唯一索引，从而保证唯一键的唯一性。当使用唯一性约束时，需要考虑如下几个因素：使用唯一性约束的字段允许为空值；一个表中可以允许有多个唯一性约束；可以把唯一性约束定义在多个字段上；唯一性约束用于强制在指定字段上创建一个唯一性索引。

创建唯一性约束的方法有两种：通过 SSMS 可以完成创建和修改唯一性约束的操作；使用 Transact-SQL 语句完成唯一性约束的操作。

- 通过 SSMS 完成创建和修改唯一性约束的操作

右键单击表，从弹出的快捷菜单中选择“设计”命令，打开表设计页面，右击某列从弹出的快捷菜单中选择“索引/键”命令，如图 9-2 所示，或者单击工具栏中的“管理索引和键”按钮，为表创建唯一性索引。

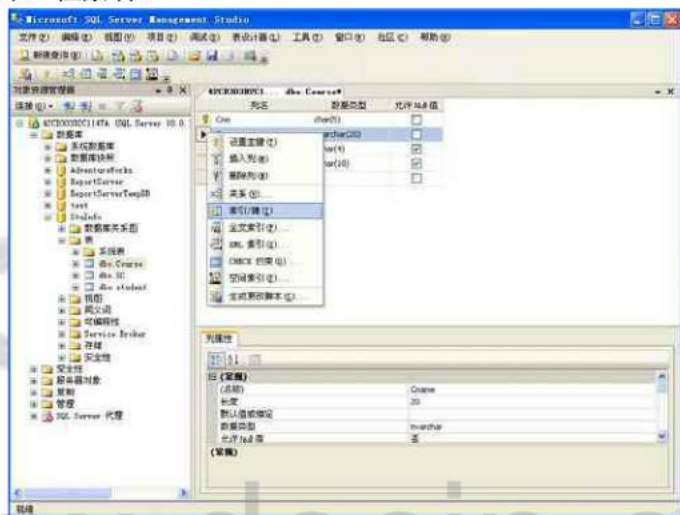


图 9-2 选择“索引/键”命令

在弹出的“索引/键”对话框中，单击“添加”按钮，添加新的主/唯一键或索引；在“常规”区域的“类型”中选择“唯一键”选项，再根据需要选中要创建的列名和排序规律等即可创建完成，如图 9-3 所示。



图 9-3 设置唯一键约束

- 创建表时用语句定义唯一约束

- (1) 定义列级唯一约束的语法

命令格式如下：

```
CREATE TABLE table_name
( column_name data_type
[ DEFAULT default_expression ] [ [ IDENTITY ( seed, increment ) ] ]
[ [ CONSTRAINT constraint_name ] UNIQUE [ CLUSTERED | NONCLUSTERED ] ]
[,... n]
)
```

显然，唯一约束与主键约束的定义十分相似，区别仅在于唯一约束采用关键字 **UNIQUE**，而主键约束采用关键字 **PRIMARY KEY**。

- (2) 定义表级唯一约束

例【9-2】在 AdventureWorks 数据库中，建立 HumanResources.Employee 表，其中 EmployeeID 和 NationalIDNumber 两列均不能为空，后者必须唯一。

```
CREATE TABLE [HumanResources].[Employee]
([EmployeeID] [int] NOT NULL, [NationalIDNumber] [nvarchar](15) NOT NULL UNIQUE)
```

### 9.4.3 CHECK 约束

对表中的某些列创建 **CHECK**(检查)约束是为了实施数据的域完整性约束。检查约束对输入列或者整个表中的值设置检查条件，以限制输入值，保证数据库数据的完整性和有效性。

使用检查约束时，应该注意以下几点：一个列级检查约束只能与限制的字段有关；一个表级检查约束只能与限制的表中字段有关；一个表中可以定义多个检查约束；每个 **CREATE TABLE** 语句中的每个字段只能定义一个检查约束；在多个字段上定义检查约束，则必须将检查约束定义为表级约束；当执行 **INSERT** 语句或者 **UPDATE** 语句时，检查约束将验证数据。

创建检查约束常用的操作方法有如下两种：使用 SSMS 图形化界面创建检查约束；使用 Transact-SQL 语句创建检查约束。

使用 Transact-SQL 语句创建检查约束的语法格式如下。

```
CREATE TABLE table_name
( column_name data_type
[ [ CONSTRAINT constraint_name ]
CHECK [ NOT FOR REPLICATION ] (check_criterial)[,... n]
]
[,... n]
)
```

命令说明:

- NOT FOR REPLICATION: 该选项表示在复制表时禁用检查约束。
- CHECK: 该关键字表示定义的约束为检查约束。
- check\_criterial 为检查准则, 一般是条件表达式, 与 WHERE 子句中的查询条件表达式类似。
- (check\_criterial)[... n]: 表示在一列上可以设置多个检查约束, 只要它们不相互矛盾。
- NOT FOR REPLICATION: 用于在复制与分发过程中使检查约束暂时失效, 而仅对用户的修改(而不是对复制进程)强加约束。

更改表的检查约束

(1) 在现有表中添加新的一列, 该列具有检查约束。

命令格式如下:

```
ALTER TABLE table_name
ADD column_name data_type
[CONSTRAINT constraint_name ]
CHECK [ NOT FOR REPLICATION ](check_criterial)[... n]
```

(2) 为表中现有的一列或几列添加检查约束。

命令格式如下:

```
ALTER TABLE table_name
[WITH CHECK | WITH NOCHECK ]
ADD [ CONSTRAINT constraint_name ]
CHECK [ NOT FOR REPLICATION ](check_criterial)[... n]
```

命令说明:

WITH CHECK | WITH NOCHECK: WITH CHECK 为默认选项, 表示将使用新的核查约束检查表中已有数据是否符合核查条件; 使用 WITH NOCHECK, 则不进行核查。

#### 9.4.4 DEFAULT 约束

在数据库中每一行记录的每一列都应该有一个值, 这个值也可以为空值。但有时添加数据时并不知道某列的具体值(或该列的值不能确定), 这时, 可以将该列定义为允许接受空值或给该列定义一个默认值。

默认约束指定在插入操作中如果没有提供输入值时, 则 SQL Server 系统会自动为该列指定一个值。默认约束可以包括常量、函数、不带变元的内建函数或者空值。

使用默认约束时, 应该注意以下几点: 每个字段只能定义一个默认约束; 如果定义的默认值长于其对应字段的允许长度, 那么输入到表中的默认值将被截断; 不能为带有 IDENTITY 属性或者数据类型为 timestamp 的字段添加默认约束; 如果字段定义为用户定义的数据类型, 而且有一个默认绑定到这个数据类型上, 则不允许该字段有默认约束。

创建默认约束常用的操作方法有如下两种：使用 SSMS 图形化界面创建默认约束和用 Transact-SQL 语句创建默认约束。

- 使用 SSMS 图形化界面创建默认约束

在 SSMS 图形化界面中，选中某一个具体的表，如：student 表。选中 student 表中的一列 sdept 列，在“表设计器”下方的“默认值或绑定”中，设定一个默认值。在以后的添加数据中，如果该列没有指定具体值，那么该值就为“计算机”，如图 9-4 所示。

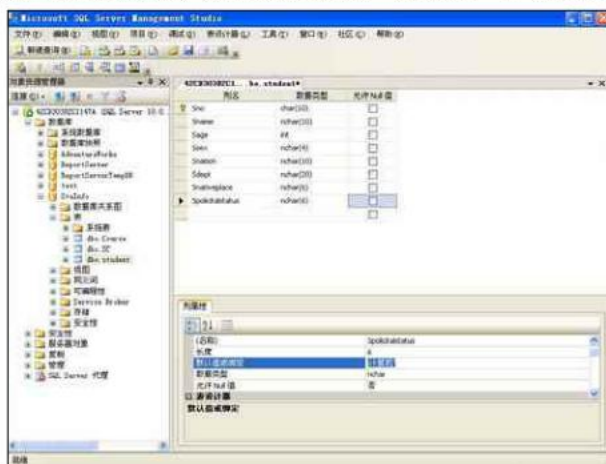


图 9-4 设置 DEFAULT 约束

- 创建表时定义默认值约束

命令格式如下：

```
CREATE TABLE table_name  
(  
    column_name data_type  
    [CONSTRAINT constraint_name]  
    DEFAULT constant_expression  
)
```

命令说明：

(1) DEFAULT 关键字用于指明其后的常量表达式 constant\_expression 为该列的默认值。常量表达式中除了允许使用常数、NULL 值和表达式作为默认值外，还可以使用 SQL Server 系统函数如 current\_user、system\_user、user、getdate() 等作为默认值。默认值的数据类型必须与列定义的数据类型一致，且不能与 CHECK 约束矛盾。

(2) DEFAULT 约束定义的默认值仅在执行 INSERT 操作插入数据时生效。

(3) 一列至多有一个默认值，其中包括 NULL 值。

(4) 具有 IDENTITY 属性或 TIMESTAMP 数据类型属性的列不能使用默认值，text 与 image 类型的列只能以 NULL 为默认值。

- 更改表的默认约束

在现有表中添加一个新列，该列具有默认值约束。



命令格式如下：

```
ALTER TABLE table_name
( ADD column_name data_type
[CONSTRAINT constraint_name]
DEFAULT constant_expression
)
```

- 为表中现有列添加默认约束

命令格式如下：

```
ALTER TABLE table_name
ADD [CONSTRAINT constraint_name]
DEFAULT constant_expression FOR column_name
```

命令说明：

默认表达式的值 `constant_expression` 用于关键字 `FOR` 所指定的列 `column_name`。

- 删除默认值约束的语法同删除主键

例【9-3】为 `student` 表中 `sdept` 字段创建默认约束。

T-SQL 语句如下：

```
CREATE TABLE student
(sno CHAR(5) PRIMARY KEY,
Sname CHAR(10) NOT NULL,
Sdept CHAR(10) DEFAULT (计算机))
```

9.4.5 NULL 约束

空值约束用于控制是否允许该字段的值为 `NULL`。`NULL` 值不是 0 也不是空白，更不是填入的“`NULL`”字符串，而是表示“不知道”、“不确定”或“没有数据”的意思。当某一字段的值一定要输入才有意义的时候，则可以设置为 `NOT NULL`。如主键列就不允许出现空值，否则就失去了唯一标识一条记录的作用。空值约束只能用于定义列约束。

创建空值(`NULL`)约束常用的操作方法也有如下两种。

- 使用 SSMS 图形化界面设置空值约束

在 SSMS 中，找到某一个具体的表，如：`SC` 表。单击鼠标右键，从弹出的快捷菜单中选择“设计”命令，打开此表。选中表中的其中一列 `Grade`，在该行的“允许 `Null` 值”的选项的复选框中选中，则表示允许该列为空，如图 9-5 所示。



列名	数据类型	允许 Null 值
Sno	char(10)	<input type="checkbox"/>
Cno	char(5)	<input type="checkbox"/>
Grade	decimal(4, 2)	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

图 9-5 设置空值约束



- 用 Transact-SQL 语句创建空值约束

其语法形式如下:

直接在相应的语句后面加上[ NULL | NOT NULL ]

例【9-4】创建一个 S 表, 包括 SNO、SNAME、AGE, 对 SAGE 字段进行 NULL 约束。

T-SQL 语句如下:

```
CREATE TABLE S
( SNO CHAR(10),
  SNAME VARCHAR(20),
  SAGE INT NULL )
```

#### 9.4.6 FOREIGN KEY 约束

FOREIGN KEY 约束是用于建立和加强两个表数据之间的链接的一列或多列。设置表中的外键约束是为了保证数据的参照完整性。SQL Server 2008 中的关系保证了表之间的连接, 用外键表(参照关系)中的外键引用主键表(被参照关系)中的主键。一旦创建了关系并将关系成功地加入到数据库中, 就能够保证数据的参照完整性。FOREIGN KEY 约束确保了同一个表或者不同表之间的引用完整性。

注意:

FOREIGN KEY 约束必须引用一个 PRIMARY KEY; 用户必须在应用表上具有 REFERENCES 权限; 一个表中最多可以有 31 个外键约束; 在临时表中, 不能使用外键约束; 主键和外键的数据类型必须严格匹配。

创建外键约束常用的操作方法有如下两种。

- 在 SSMS 图形化平台上添加外键约束

在 SC 表中选中一列, 如: sno 列, 如图 9-6 所示。右击该列, 从弹出的快捷菜单中选择“关系”命令, 就会弹出“外键关系”对话框, 如图 9-7 所示。单击“添加”按钮即可添加新的约束关系; 设置“在创建或重新启用时检查所有数据”为“是”; 对外键名称、强制外键约束和强制用于复制选项进行设置; 在“表和列规范”中设置表与列之间的参照关系, 如图 9-8 所示。

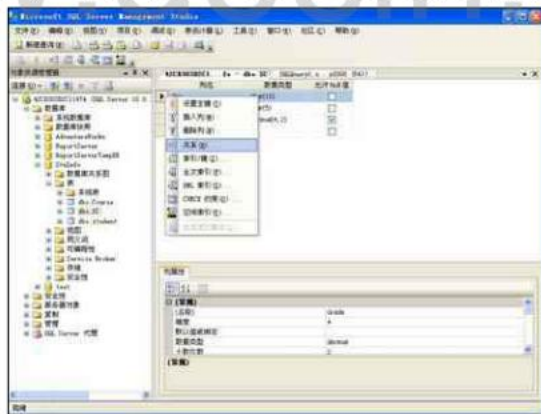


图 9-6 选择参照关系



图 9-7 设置外键约束



图 9-8 设置表和列

提示：如果“强制外键约束”和“强制用于复制”选项都设置为“是”，则能保证任何数据添加、修改或删除都不会违背参照关系。

• 创建表时定义外键约束

创建表时定义外键约束的语法格式如下：

```
CREATE TABLE table_name
(column_name data_type
[ [ CONSTRAINT constraint_name ]
FOREIGN KEY REFERENCES ref_table (ref_column)
[ ON DELETE { CASCADE|NO ACTION } ]
[ ON UPDATE { CASCADE|NO ACTION } ]
[ NOT FOR REPLICATION ]
] [... n]
[ [ CONSTRAINT constraint_name ]
FOREIGN KEY (column_name [... n])
REFERENCES ref_table (ref_column[,... n])
[ ON DELETE { CASCADE|NO ACTION } ]
[ ON UPDATE { CASCADE|NO ACTION } ]
```

```
[ NOT FOR REPLICATION ]
] [... n]
)
```

命令说明:

(1) 第 1 个外层方括号[]表示外键作用于单列(列级约束)时的选项,第 2 个外层方括号[]表示外键作用于列组合时的选项。

(2) 对于列级外键约束,关键字 FOREIGN KEY REFERENCES 用于指明其后的 ref\_table(ref\_column)为引用表名称及其列名,该列名所指定的列在引用表中必须为主键或唯一约束列。

(3) 对于列组合型外键约束,FOREIGN KEY 后面的(column\_name [... n])表示本表中的外键列组合,ref\_table(ref\_column[,... n])表示所引用的主键表的表名 ref\_table 及其主键或唯一约束列组合 ref\_column[,... n]。两表中外键与主键的列名顺序及其数据类型一定要相同。

(4) ON DELETE [CASCADE | NO ACTION] 表示在主键表中删除数据行时,级联删除外键表中外键所对应的数据行(CASCADE)或不做任何操作(NO ACTION)。对 ON UPDATE 则为级联修改或否。

(5) NOT FOR REPLICATION 选项表示在复制表时禁用外键。

- 更改现有表的外键约束

在现有表中添加新的一列及其外键约束的语法格式如下:

```
ALTER TABLE table_name
( ADD column_name data_type
[ [ CONSTRAINT constraint_name ]
FOREIGN KEY REFERENCES ref_table (ref_column)
[ ON DELETE { CASCADE | NO ACTION } ]
[ ON UPDATE { CASCADE | NO ACTION } ]
[ NOT FOR REPLICATION ]
] [... n]
[ [ CONSTRAINT constraint_name ]
FOREIGN KEY (column_name [... n])
REFERENCES ref_table (ref_column [... n])
[ ON DELETE { CASCADE | NO ACTION } ]
[ ON UPDATE { CASCADE | NO ACTION } ]
[ NOT FOR REPLICATION ]
] [... n]
)
```

- 对表中的某一列(或列组合)添加外键约束的语法格式如下:

```
ALTER TABLE table_name
[WITH CHECK | WITH NOCHECK]
ADD [ CONSTRAINT constraint_name ]
FOREIGN KEY (column_name [... n])
REFERENCES ref_table (ref_column [... n])
[ ON DELETE { CASCADE | NO ACTION } ]
[ ON UPDATE { CASCADE | NO ACTION } ]
[NOT FOR REPLICATION]
[,... n]
```

例【9-5】在 AdventureWorks 数据库中, 修改表, 添加外键约束, 使 Sales.SalesOrderHeader 表中的 CustomerID 列作为 Customer 表中的 CustomerID 列的外键。

```
ALTER TABLE [Sales].[SalesOrderHeader] WITH CHECK
ADD CONSTRAINT [FK_SalesOrderHeader_Customer_CustomerID] FOREIGN KEY([CustomerID])
REFERENCES [Sales].[Customer] ([CustomerID])
```

## 9.5 禁止约束

约束是作用于表的数据库对象, 离开了表, 约束将不再存在。除了约束之外, SQL Server 还提供了两个独立于表的数据库对象用以维护数据库的完整性, 即默认和规则。规则的作用类似于核查约束, 如果将一个规则绑定到指定列上, 则可以检查该列的数据是否符合规则的要求。

规则与核查约束的主要区别在于一列只能绑定一个规则, 但却可以设置多个核查约束。规则的优点是仅创建一次就可以绑定到数据库的多个表的列上, 使同一数据库中所有表的不同列共享。规则还可以绑定到同一数据库中一个以上的用户定义的数据类型上。

### 9.5.1 创建规则

创建规则的语法格式如下:

```
CREATE RULE rule_name AS rule_expression
```

命令说明:

(1) rule\_name 为要建立的规则的名称。

(2) rule\_expression 为规则表达式, 它与 WHERE 子句中的条件表达式类似, 可以使用比较表达式、逻辑表达式、LIKE 子句等, 所不同的是需要将 WHERE 子句中的列名改为一个以@开头并符合 SQL Server 命名规则的参数。将规则绑定到列上时, 该参数代表 INSERT 语句或 UPDATE 语句所输入的数据, 因此, 其参数名字可以随意。注意 rule\_expression 不能引用表中的列或其他数据库对象。

### 9.5.2 绑定规则

规则创建后,它仅仅是一个数据库对象,与其他数据库对象并没有联系。要将规则应用于一个表(或用户定义的数据类型)中,必须将其与表中的指定列(或用户定义的数据类型)相结合,称之为将规则绑定(bind)到列(或用户定义的数据类型),或称绑定规则。绑定规则需要使用系统存储过程 `sp_bindrule`,语法格式如下:

```
sp_bindrule rule_name, 'table_name.column_name'
```

或

```
sp_bindrule rule_name, 'user_defined_datatype' [, 'futureonly_flag']
```

命令说明:

(1) `table_name.column_name` 指明将 `rule_name` 所表示的规则绑定到指定表的指定列上,列名前面必须有表名项,如果仅有列名而没有表名,则系统视它为用户定义的数据类型。

(2) `user_defined_datatype` 指明将规则绑定到一个用户定义的数据类型上。

(3) `futureonly_flag` 为一个标志变量,它仅在将规则或默认绑定到用户定义的数据类型时使用。当表中的某一列具有尚未绑定规则的用户定义数据类型属性时,此时如果再将一个新规则绑定到上述用户定义数据类型,则 `futureonly_flag` 的作用为:

- `futureonly_flag` 取值为 `FUTUREONLY` 时,在将规则绑定到数据类型时,该绑定规则的用户定义数据类型仅对新表和表中新列起作用,对于此次绑定之前的那些具有该用户定义数据类型属性的列无效,即具有该用户定义数据类型属性的现有列不会继承新规则。
- `futureonly_flag` 取值 `NULL`(此为默认选项)时,新规则将绑定到具有用户定义数据类型属性的每一列,包括现有列和新建列。

可以多次将不同规则绑定到同一列,但只有最近一次绑定的规则生效。亦即对同一列,每绑定一次新规则,以前所绑定的旧规则将自动失效,一列只能有一个规则生效。

### 9.5.3 解除绑定

当表中不再需要规则时,可以将规则与列分离,称为解除绑定。解除绑定需要使用系统存储过程 `sp_unbindrule`,语法格式如下:

```
sp_unbindrule 'table_name.column_name'
```

或

```
sp_unbindrule 'user_defined_datatype' [, 'futureonly_flag']
```

命令说明:

`futureonly_flag` 的默认值为 `NULL`,当它取值为 `FUTUREONLY` 时,现有使用该数据类型的列值不会失去规则,仅对将来输入的数据不采纳规则。解除绑定后规则仍然存在于数据



库系统中，只不过它与表的联系没有了。

### 9.5.4 删除规则

删除规则的前提是解除数据库中所有表的绑定。删除规则的语法格式如下：

```
DROP RULE rule_name
```

## 9.6 默 认

与规则类似，默认值对象(简称默认)的优点也是仅创建一次就可以绑定到数据库的多个表的列或用户自定义数据类型中，使它们共享默认。

### 9.6.1 创建默认

创建默认的语法格式如下：

```
CREATE DEFAULT default_name AS default_expression
```

命令说明：

- (1) `default_name` 是符合 SQL Server 标识符规则的默认值名称。
- (2) `default_expression` 是常量，用于指出默认值的具体数值或字符串。

### 9.6.2 绑定默认

绑定默认的语法格式如下：

```
sp_bindefault default_name, 'table_name.column_name'
```

或

```
sp_bindefault default_name, 'user_defined_datatype' [, 'futureonly_flag']
```

命令说明：

- (1) `default_name` 为默认值数据库对象的名称。
- (2) 其他语法项目的用法与规则类似。

### 9.6.3 解除绑定

解除默认绑定的语法格式如下：

```
sp_unbindefault 'table_name.column_name'
```

或

```
sp_unbindefault 'user_defined_datatype' [, 'futureonly_flag']
```

### 9.6.4 删除默认

删除默认之前要先解除默认绑定。删除默认的语法格式如下：

```
DROP DEFAULT default_name
```

### 9.6.5 使用 SSMS 图形化界面管理默认值对象

与管理规则类似，在企业管理器中展开指定数据库节点，单击“默认”，在窗口右边可以看到当前数据库中的所有默认值对象。右击“默认”，从弹出的快捷菜单中选择“新建默认”命令，在弹出的窗口输入默认名称及其取值。或右击某个现有的默认值对象，从弹出的快捷菜单中选择“属性”命令，可以查看、修改或绑定该默认值对象。注意在修改之前需要解除绑定。

## 9.7 使用自动编号 IDENTITY

SQL Server 为自动进行顺序编号而引入了自动编号的 IDENTITY 属性。当需要对某一列输入的数据进行有规律的顺序编号且保证每个编号唯一时，最好是采用 IDENTITY，这样既方便准确，速度又快。具有 IDENTITY 属性的列称为标识列，其取值称为标识值，具有如下特点：

- (1) IDENTITY 列的数据类型只能为 tinyint、smallint、int、bigint、numeric、decimal。当为 numeric、decimal 类型时，不允许有小数位。
- (2) 当用户向表中插入新的一行记录时，不必也不能向具有 IDENTITY 属性的列输入数据，系统将自动在该列添加一个按规定间隔递增(或递减)的数据。
- (3) 每个表至多有一列具有 IDENTITY 属性，该列不能为空、不允许具有默认值、不能由用户更新。因此，IDENTITY 列常可以作为主键使用。
- (4) 使用 IDENTITY 时，可以设置种子(seed)与增量(increment)。“种子”表示系统为表中第一条记录添加的自动编号数字。“增量”表示相邻两条记录之间后一个自动编号数字减去前一个自动编号数字的数值差，正值表示后一数据大于前一数据，反之，表示后一数据小于前一数据。必须同时指定种子和增量，或者二者都不指定。如果二者都未指定，则取默认值(1,1)。当对表中数据进行删除操作后，在标识值之间可能会产生数量不等的差值。

设置 IDENTITY 属性的语法格式如下。

建立表时定义标识列。

命令格式如下：

```
CREATE TABLE table_name  
( column_name data_type  
[ IDENTITY [( seed, increment )] ] NOT NULL  
[,... n]  
)
```

## 9.8 习 题

1. 什么是数据的完整性？数据完整性的主要问题是什么？
2. 什么是域完整性、实体完整性和引用完整性？
3. 约束的作用和类型是什么？
4. DEFAULE 约束的特点是什么？
5. 为什么要引入 CHECK 约束？
6. 主键约束的作用和创建方式是什么？
7. UNIQUE 约束的使用场景是什么？
8. 外键约束的特点是什么？





























