第 1 章　SQL Server 概述

1.1　SQL

SQL（Structured Query Language），即结构化查询语言，　是关系数据库的标准语言，　SQL 是一个通用的、　功能极强的关系数据库语言。　其功能并不仅仅是查询。　当前，几乎所有的关系数据库管理系统软件都支持　SQL，许多软件厂商对　SQL 基本命令集还进行了不同程度的扩充和修改。

1.2　SQL 的特点

SQL 之所以能够为用户和业界所接受，并成为国际标准，是因为它是一个综合的、功能极强同时又简洁易学的语言。　SQL 集数据查询（Data Query）、数据操纵（Data Manipulation）、数据定义（ Data Definition ）和数据控制（ Data Control ）功能于一体，主要特点包括：

综合统一： SQL 集数据定义语言　DDL 、数据操纵语言　DML 、数据控制语言　DCL 的功能于一体，语言风格统一，可以独立完成数据库生命周期中的全部活动，包括定义关系模式，插入数据，建立数据库；对数据库中的数据进行查询和更新；数据库重构和维护；数据库安全性、完整性控制等一系列操作要求。

高度非过程化：非关系数据模型的数据操作语言是“ 面向过程 ”的语言，用“ 过程化 ”语言完成某项请求，必须指定存取路径。而用　SQL 进行数据操作，只要提出“ 做什么 ”，而无须指明“ 怎么做 ”　，因此无需了解存取路径。存取路径的选择以及 SQL 的操作过程由系统自动完成。　这不但大大减速轻了用户负担，　而且有利于提高数据独立性。

面向集合的操作方式：非关系数据模型采用的是面向记录的操作方式，操作对象是一条记录。 SQL 采用集合操作方式，不仅操作对象、查找结果可以是元组的集合，而且一次插入、删除、更新操作的对象也可以是元组的集合。

以同一种语法结构提供多种使用方式：　SQL 既是独立的语言，又是嵌入式语言。

语言简洁，易学易用： SQL 功能极强，但由于设计巧妙，语言十分简洁，完成核心功能只用了　9 个动词，如　1 所示。 SQL 接近英语口语，因此容易学习，容易使用。

表 1 SQL 的动词

SQL 功能	动词
数据查询	SELECT
数据定义	CREATE 、 DROP 、 ALTER
数据操纵	INSERT 、 UPDATE 、 DELETE
数据控制	GRANT 、 REVORE

1.3 SQL Server 的结构

一、 SQL Server 是一个单一进程多线程的关系型数据库

二、 SQL Server 是以 Client/Server 为设计结构

三、 Client/Server 数据库的两个 API

四、 SQL Server 支持在客户端以 Net-Library 或 ODBC 存取服务器端

五、支持分布式数据库结构

1.4 SQL Server 的性能

表 2 SQL Server 的性能

对象	范围
数据库	32767 个数据库，最小为 1MB，最大为 1TB
表	每个数据库最多有 20 亿个表，每列的最大字节数为 8060（文本和图像列除外）
列	每表最多 1024 列
索引	每表一个簇式索引， 249 个非簇式索引， 一个复合索引最多有 16 个索引关键字
触发器	每表最多有 3 个触发器，分别用于 insert、update 和 delete
存储过程	一个存储过程可以有 1024 个参数和最多 32 级嵌套
用户连接	32767 个
锁定及打开的对象	20 亿
打开的数据库	32767 个

1.5 SQL 分布管理框架介绍

1.6 关系数据库模型简介

关系数据库是一种所有用户可见数据都按表的形式组织起来的表， 且所有库操作都针对这些表的数据库，关系数据模型是以集合论中的关系（ relation）概念为基础发展起来的数据模型。

当前实际的数据库系统中所支持的主要模型有：

层次模型

网状模型

关系模型

在层次数据模型中，要查找一个记录必须从根记录开始，按给定条件沿一个层次路径查找所需要的记录。 在网状数据模型中， 要查找语句中不但要说明查找的对象， 而且还要规定存取的路径，操作语句也比较繁琐， 而关系数据库，通过关系，按给定的选择条件选出符合条件的元组，比较灵活。

关系数据库是应用数学方法来处理数据库数据的，与层次模型和网状模型相比，有很大改进。表现在：

面向集合的处理，可以一次操作多个行

数据的逻辑独立性，使得应用程序不随数据库的改变而改变

数据的自动导航，数据的访问路径由数据库优化器决定，方便了用户操作

关系模型的 RDBMS 的基础，它包括三部分：

（1） 关系模型的数据结构

关系模型的数据结构为单一的数据结构 —由行和列组成的二维表，任意两行互不相同，

列值是不可分的数据项，行和列的次序可任意。

（2）　关系模型的完整性

关系模型的完整性包括实体完整性、参照完整性和用户定义的完整性。实体完整性是指用主键来惟一标志表中行和列，　主键的任一属性不能为空。　参照完整性指外键或者为空，　或者等于它所参照的主键的某个值。用户定义的完整性指对某一具体的数据库的约束条件。

（3）　关系模型的数据操作

关系模型的操作表达能力非常强大，　定义了很多的操作，　其中主要有选择、　投影、集合、连接等操作。

第 2 章　SQL Server 的安装及基本操作

一、 SQL Server 2000 的硬件和软件安装要求

二、 SQL Server 的安装内容

每一个 SQL Server 都包含两种类型的数据库：　系统数据库和用户数据库。　系统数据库中存储有 SQL Server 的整体信息，　SQL Server 使用系统数据库操作和管理系统。用户数据库是用户自己创建的数据库。在安装　SQL Server 时，　SQL Server 安装程序创建系统数据库和示例数据库。

（1）　master 数据库

（2）　model 数据库

（3）　msdb 数据库

（4）　tempdb 数据库

（5）　pubs 示例数据库

三、 SQL Server 的安全性

四、安装　SQL Server

五、 SQL 群组中各项目的作用

六、 SQL Server 服务器端的操作

七、 SQL Server 客户端的操作界面

第 3 章　数据库的基本操作

3. 1 文件和文件组

3. 1. 1 文件

SQL Server2000 使用一组操作系统文件映射数据库。　数据库中的所有数据和对象　（如表、存储过过程、触发器和视图）都存储在下列的操作系统文件中。

主要数据文件：该文件包含数据库的启动信息，并用于存储数据。每个数据库都有一个主要数据文件。

次要数据文件：这些文件含有不能置于主要数据文件中的所有数据。如果主文件可以包含数据库中的所有数据，　那么数据库就不需要次要数据文件。　有些数据库可能足够大故需要多个次要数据文件，或使用位于不同磁盘驱动器上的辅助文件将数据扩展到多个磁盘。

事务日志：这些文件包含用于恢复数据库的日志信息。每个数据库都必须至少有一个日志文件

3. 1. 2 文件组

3. 2 创建数据库

3. 2. 1 用企业管理器以图形化界面建立数据库

3. 2. 2 在查询分析器窗口中用　T-SQL 命令创建数据库

```
CREATE DATABASE tsing_DB
ON
PRIMARY (NAME=tsing_DBData,
FILENAME=    'C:\Program Files\Microsoft SQL Server\MSSQL\Data\tsing_DB.mdf      ',
SIZE=25MB,
MAXSIZE=50MB,
FILEGROWTH=2MB)
LOG ON
(NAME=tsing_DBLog,
FILENAME=    'C:\Program Files\Microsoft SQL Server\MSSQL\Data\tsing_DB.ldf      ',
SIZE=10MB,
MAXSIZE=20MB,
FILEGROWTH=25%)
```

3. 2. 3 事务日志

在 SQL Server2000 中，数据库必须至少包含一个数据文件和一个事务日志文件。　　数据和事务日志信息从不混合在同一文件中，并且每个文件只能由一个数据库使用。

SQL Server 使用各数据库的事务日志来恢复事务。　　事务日志是数据库中已发生的所有修改和执行每次修改的事务的一连串记录。

3. 2. 4 查看数据库信息

(1)用企业管理器以图形化界面查看数据库信息

(2)在查询分析器窗口中用　T-SQL 命令查看数据库信息

```
EXEC sp_helpdb database_name
```

3. 3 管理数据库

3. 3. 1 打开数据库

```
use database_name
```

3. 3. 2 增加数据库容量

3. 3. 3 查看目前数据库选项设定及修改

3. 3. 4 缩减数据库容量

3. 3. 5 更改数据库名称

```
EXEC sp_renamedb OldName,NewNAME
GO
```

3. 3. 6 查看目前 SQL Server 上共有几个数据库

```
use master
go
select name from sysdatabases
```

3. 3. 7 数据库的删除

```
DROP DATABASE database_name [,database_name…]
```

或

```
sp_dbremove database_name
```

或

通过图形化界面进行数据库的删除。

第 4 章　数据库中表的建立

4.　1　表的概念

表是由数据记录按一定的顺序和格式构成的数据集合，不同表的集合构成数据库。表是数据库的对象之一，表中包含有多条记录，而记录中有多个字段，字段是描述事物的属性。

SQL Server 对表格有如下限制：

每个数据库里最多有　20 亿个表

每个表格仅可以有　1 个簇索引，　249 个非簇索引

每个表格最多有　1024 个字段

表格中每条记录最大为　8060 字节，但不包括　text 字段和　image 字段，因为它们有独立的存储页链。

数据库逻辑设计包含了数据规范化和各表格之间关联的建立，　在 SQL Server 中有　3 个影响执行速度效率的主要因素：

数据库逻辑设计

索引设计

查询命令设计

4.　2 SQL Server 的数据类型

4.　3　数据库中表的操作

一、数据库中表的创建

例 4-1　在 tsing_DB1 数据库中创建一张用于学生信息管理的表　stud_info，表中包括的学生信息分别为：学生学号、学生姓名、出身年月、性别、家庭地址、电话号码、邮政编码、所在系代号、　每月生活补贴。　用户可以在　SQL 查询分析器窗口中的查询页下输入以下内容：

```
CREATE TABLE tsing_DB1.dbo.stud_info
(stud_id int CONSTRAINT perid_chk NOT NULL PRIMARY KEY,
name nvarchar(5) NOT NULL,
birthday datetime,
gender nchar(1),
address nvarchar(20),
telcode char(12),
zipcode char(6) CONSTRAINT zip_chk CHECK(zipcode LIKE　'[0-9] [0-9] [0-9] [0-9] [0-9] [0-9]'),
DeptCode tinyint CONSTRAINT DeptCode_chk CHECK(DeptCode<100),
Salary money DEFAULT 260)
```

例 4-2　在 tsing_DB1 数据库中创建一张学生成绩表，以方便学生成绩的统计。该表包含的字段依次为：学年年度、学生学号、数学成绩、英语成绩、计算机成绩、化学成绩和物理成绩。

```
CREATE TABLE tsing_DB1.dbo.stud_score
(year int NOT NULL,
stud_id int NOT NULL,
math_score numeric(4,1) CHECK(math_score>=0 and math_score<=100),
engl_score numeric(4,1) CHECK(engl_score >=0 and engl_score <=100),
comp_score numeric(4,1) CHECK(comp _score >=0 and comp _score <=100),
chem_score numeric(4,1) CHECK(chem_score >=0 and chem_score <=100),
```

phys_score numeric(4,1) CHECK(phys_score >=0 and phys_score <=100),

CONSTRAINT pk_chk PRIMARY KEY(year,stud_id)

)

例 4-3 带有参照性约束的表的创建

在数据库 tsing_DB2 中创建一个用于学校实验室仪器管理的试验仪器管理表，表名为 device_manage，表中包括的信息有：仪器编号、仪器名称、仪器所属试验室的实验室名称、同种仪器的数量、该仪器的购买价格及该仪器的供货商。

CREATE TABLE tsing_DB2.dbo.device_name

(

dev_id varchar(15) CONSTRAINT pk_chk NOT NULL PRIMARY KEY ,

dev_name varchar(20) NOT NULL,

lab_id varchar(20) NOT NULL,

dev_qty int,

unit_price money,

supply_id varchar(15)

)

创建一张用于记录每次试验中试验仪器使用情况的表 device_use。该表中记录的信息有：实验名称、 实验地点、实验日期、试验学生学号以及该学生使用的仪器编号。我们假设每次实验中每个学生用且只用同一台仪器。 但该仪器必须为实验仪器管理表中所记录的仪器， 即表 device_use 中的仪器编号必须参照表 device_manage 中的仪器编号。 该表创建的命令如下：

CREATE TABLE tsing_DB2.dbo.device_use

(

experiment_name varchar(20),

experiment_lab varchar(20),

experiment_date datetime,

stud_id int,

dev_id varchar(15) CONSTRAINT fk_chk REFERENCES device_manage(dev_id)

)

二、数据库中表的删除

USE tsing_DB1

Go

Drop table stud_info

Go

三、数据库中表的修改

(1) ALTER 语句

例 向表 stud_info 中加入字段名为 dormitory 的一列， 以记录该学生的宿舍号。 该列的属性为 nvarchar(20),默认值为“ 16 号楼 ”。

ALTER TABLE tsing_DB1.dbo.stud_info

ADD dormitory nvarchar(20) NULL DEFAULT('16 号楼 ')

ALTER TABLE tsing_DB1.dbo.stud_info

DROP COLUMN dormitory

(2) INSERT 语句

通过 INSERT 语句，可以向表中添加新的记录或在记录中插入部分字段的数据。

（A）整行数据插入

例 下面的例子将向 tsing_DB1 中的 stud_info 表插入 2 行记录

```
use tsing_DB1
go
INSERT stud_info
Values(0811, 张源 ', '12/15/1979 ', '1 ', '北京市海淀区 ', '010-64572345 ', '100080 ',3,260)
Go
INSERT stud_info
Values(970890, 赵明 ', '8/5/1979 ', '1 ', '上海市浦东区 ', '021-64897232 ', '201700 ',2,260)
go
```

在添加记录过程中应注意输入数据的格式，不同的数据类型有不同的输入格式。子句 go 用来执行一条独立的完整的 SQL 命令。

（B）插入部分数据

可以在 INSERT 子句中指定列名，values 子句中的数据项与指定的列名相对应，未列出的列必须具有允许 NULL 、timestamp、IDENTITY 、DEFAULT 四种定义的条件之一。这样就能向表中插入部分列的数据。跳过的列将以默认值或者用 NULL 填充。

```
INSERT stud_info （stud_id,name, gender）
Values(970890, 赵明 ', '1 ')
```

(C) 查询结果的插入

在 INSERT 语句中可用 SELECT 从同一张表或其他表中插入行，插入的表的结构必须与 SELECT 查询返回的结果相容。通过 SELECT 子句可以插入多行。

```
Use tsing_DB1
CREATE TABLE tsing_DB1.dbo.stud_info1
(stud_id int CONSTRAINT perid_chk NOT NULL PRIMARY KEY,
name nvarchar(5) NOT NULL,
birthday datetime,
gender nchar(1),
address nvarchar(20),
telcode char(12),
zipcode char(6) CONSTRAINT zip_chk CHECK(zipcode LIKE '[0-9] [0-9] [0-9] [0-9] [0-9] [0-9] '),
DeptCode tinyint CONSTRAINT DeptCode_chk CHECK(DeptCode<100),
Salary money DEFAULT 260)
go
INSERT into stud_info1(stud_id,name)
Select stud_id,name from stud_info
```

(3)UPDATE 语句

use tsing_DB1
UPDATE stud_info
SET name= '李四 ', gender= '2 '
WHERE stud_id=970890

(4)DELETE 语句
通过 DELETE 子句将从表中删除一行或多行记录
DELETE FROM stud_info
WHERE stud_id=970890

第 5 章 数据检索

在介绍了表的创建并介绍了必要情况下对表的修改之后，我们就可以对表的内容进行查询等操作，以发挥数据库应有的提供信息的作用。查询主要是根据用户提供的限定条件进行，查询的结果返回一张能满足用户要求的表，查询主要由 SELECT 来完成，在创建好数据库及数据库下的表并输入数据后，用户就可以对数据进行检索了。

5. 1 SELECT 语句

T-SQL 语言中数据的检索是通过 SELECT 语句及与其他一系列子句配合来完成。

例 5-1 查询数据库 tsing_DB2 中表 device_manage 中的所有信息

use tsing_DB2
select * from device_manage

例 5-2 把表 stud_score 中 comp_score 字段名在查询结果的显示中改为 computer_score, 以便于阅读，本查询要求列出学年度（ year)、学号（ stud_id)、计算机科目（ computer#）及计算机科目的成绩（表中字段名为 comp_score,在输入时改为 computer_score）等信息。

Select year,stud_id,computer#= 'computer score ', computer_score=comp_score
From stud_score

例 5-3 把学生的数据成绩 math_score进行输出，且输出值在原有值的基础上提高了 5%。

Select year,stud_id,new_score=math_score*1.05
From stud_score

在对记录值进行数学运算时，请注意运算符的优先级。

先乘除（ *、/、%)、后加减（ +、-)，相同优先级时，表达式采用从左到右的计算顺序。采用（）明确优先级，能有效减少失误。

在查询中也可以使用总计函数返回数据，其语法如下：

SELECT caption=function_name(col_name)

总计函数据共有 6 个，它们分别是：

SUN （）：求总和
AVG （）：求平均值
MIN （）：求最小值
MAX （）：求最大值
COUNT （）：传回非 NULL 值的字段数目
COUNT （*）：传回符合查询条件的数目

例 5-4 下例通过 AVG （）函数返回每年度学生的平均成绩。

Select year,average=avg(comp_score)
From stud_score
Group by(year)
Order by(year)

注： Group by(year) 子句的意思是根据入学年限　(year)分组。

Order by(year) 子句的意思是根据入学年限排序。

例 5-5 查询 stud_score 表中成绩字段 comp_score 中的总成绩、最高分、最低分、最高分和最低分的差额。

```
SELECT
Sum=sum(comp_score),max=max(comp_score),min=min(comp_score),sub=max(comp_score)-min(comp_score)
FROM stud_score
```

5. 2 带条件的检索

在检索过程中，经常需要根据一定的条件对数据进行过滤，这就是带条件的检索，带条件的检索主要用 WHERE 、HAVING 、GROUP BY 等子句。

5. 2. 1 WHERE 子句

在检索信息时可以通过 WHERE 子句指定检索的条件， 而且 SQL Server 还提供了 NOT 、OR、 AND 三种逻辑运算符，其中 NOT 的优先级最高，其次是 AND ， OR 的优先级最低。

例 5-6 查询满足如下条件的学生， 其中计算机课成绩不大于数学成绩， 而且计算机成绩大于化学成绩，或者化学成绩大于数学成绩的学生。

```
USE tsing_DB1
SELECT year,stud_id,comp_score
FROM stud_score
WHERE (comp_score<=math_score AND comp_score>chem_score)
        OR
        Chem_score>math_score
```

5. 2. 2 WHERE 子句中条件的指定

SQL Server 支持的运算符如下：

（1）逻辑运算符： AND 、OR、 NOT

（2）比较符： =、！=、 >、 >=、<、 <=

IN、 NOT IN 、 ANY 、ALL 判断是否是集合的成员

BETWEEN AND 判断列值是否满足指定的区间

LIKE 匹配模式

Is [not] null 测试空值

例 5-8 查询 stud_info 中的姓氏为‘ 陈 ’的学生姓名。

```
USE tsing_DB1
SELECT name
FROM stud_info
WHERE name LIKE 陈%’
```

注： %是字符匹配符，能匹配任意长度的字符串。还有一种字符匹配符‘ _ ’，只能匹配一个字符。此外，字符匹配符 [] 可以匹配对指定范围（例如 [a～f] ）或集合（例如 [abcdef] ）中的任何单个字符；字符匹配符 [^] 可以匹配不属于指定范围的任何单个字符。

例 5-9 查询 stud_score 表中学年年度在 1997（含 1997）到 1998（含 1998）的学生信息。

```
USE tsing_DB1
SELECT year,stud_id,comp_score
FROM stud_score
WHERE year BETWEEN 1997 AND 1998
```

通过 BETWEEN AND 子句可以指定一个范围，指定的范围包含区间的两个端点。

例 5-11 查询数据库 tsing_DB2 中表 device_manage 中 supply_id 字段值为 null 的设备信息。

```
USE tsing_DB2
SELECT dev_id,dev_name,lab_id
FROM device_manage
WHERE supply_id is null
```

注：使用 in、any 子名用于判定元素是否在集合中。

例 5-12 查询数据库 tsing_DB1 中表 stud_score 中学生数学成绩为 80、85 或 90 的学生的学号、学年年度和该学年的数学成绩。

```
USE tsing_DB1
SELECT year,stud_id,math_score
FROM stud_score
WHERE math_score in (80,85,90)
```

在信息查询中，还可以根据另一张表的查询结果来确定查询条件。

例 5-13

```
use tsing_DB2
SELECT dev_id,dev_name,lab_id FROM device_manage
WHERE dev_id=
( SELECT dev_id FROM device_use WHERE experimen_lab is null)
```

5. 2. 3 HAVING 子句

HAVING 子句也能指定查询条件，查询表达式与 WHERE 子句中的类似。

例 5-15

```
USE tsing_DB1
SELECT year,total=sum(comp_score)
FROM stud_score
GROUP BY(year)
HAVING year<1998
```

5. 2. 4 COMPUTE BY 及 COMPUTE 子句

COMPUTE BY 子句可以通过 BY 指定字段进行分组计算，COMPUTE 子句则计算所有的字段值之和。

例 5-17 查询学号为 96**** 的学生并按入学年度分别计算他们的总分，最后计算出所有符合查询条件的学生的 comp_score 的总分。

```
USE tsing_DB1
SELECT stud_id,year,comp_score
FROM stud_score
WHERE stud_id LIKE '96%'
ORDER BY year
COMPUTE SUM(comp_score) BY year
COMPUTE SUM(comp_score)
```

5. 3 从多张表中查询数据

例 5-18 通过查询 stud_info 和 stud_score 两个表，以获得学号为 980814 的学生的姓名、性别以及其在 1998 年的数学科目的成绩。

```
SELECT stud_info.stud_id,stud_info.name,stud_info.gender,stud_score.math_score
FROM stud_info,stud_score
```

```
WHERE stud_info.stud_id=980814 AND stud_info.stud_id=stud_score.stud_id
  AND stud_score.year=1998
```

5. 4 UNION 操作

通过 UNION 操作可以把两个或两个以上的查询结果合并到一个结果集中。

例 5-21

```
SELECT stud_id,name
FROM stud_info
UNION
SELECT stud_id,name
FROM Add_stud_info
```

5. 5 子查询

可以在 INSERT、SELECT、UPDATE、DELETE 等地方嵌套 SELECT 查询子句。

例 5-23

```
UPDATE temp_stud_info
SET tpColumn= 'New'
WHERE stud_id
IN
(SELECT stud_id FROM add_stud_info)
```

第 6 章 实现 SQL Server 的数据完整性

6. 1 数据完整性的基本概念

6. 1. 1 数据完整性的类型

数据完整性是指存储在数据库中的数据的一致性和准确性。在 SQL Server 中数据的完整性可能会由于用户进行的各种数据操作（例如 INSERT、DELETE 和 UPDATE 操作）而遭到破坏。例如当用户将一个学号为空的学生记录插入到一张学籍管理的表中时（该表中学生记录的学号必须非空），就破坏了学籍表中的数据完整性。为了保证数据库的数据完整性，SQL Server 设计了多种数据完整性约束。在 SQLServer 中，数据完整性可以分为以下四种类型：

实体完整性（ entity integrity ）

实体完整性将行定义为特定表中的惟一实体，要求表中所有的行具有惟一的标志符。也就是说，表中的所有记录在某一字段上必须取值惟一，这一字段通常就是我们所说的“主键（primary key ）”。此外，实体完整性还可以通过索引、UNIQUE 约束或指定 IDENTITY 属性来实现。

域完整性（ domain integrity ）

域完整性是指数据库表中对指定列有效的输入值。强制域有效性的方法有：限制数据类型（通过数据类型）、格式（通过 CHECK 约束和规则）或可能值的范围（通过 FOREIGN KEY 约束、CHECK 约束、DEFAULT 约束、NOT NULL 定义和规则）。由于域完整性可以反映一定的业务规则，因些也被称为“商业规则（ business rule ）”。例如：学生成绩 <=100

引用完整性（ reference integrity ）

在输入或删除记录时，引用完整性可以用来保持所有表之间定义的关系。在 SQLServer 中，引用完整性为基于外键与主键之间或外键与惟一键之间的关系（通过 FOREIGN KEY 和 CHECK 约束）。引用完整性可以用来确保键值在所有表中一致。这样的一致性要求不能引用不存在的值，如果键值更改了，那么在整个数据库中，对该键值的所有引用要进行一致的更改。

用户定义完整性（ user defined integrity ）

用户定义完整性使用户得以定义不属于其他任何完整性分类的特定业务规则。所有的完

整性类型都支持用户定义完整性（　　CREATE TABLE中的所有列级和表级约束、存储过程和触发器）。

6. 1. 2 强制数据完整性

了解了数据库中数据完整性的重要性后，接着让我们来了解　　SQL Server 中强制数据完整性的两种方法，这里的强制数据完整性实际上就是数据完整性的实现。

（1）　声明数据完整性

声明数据完整性是指定义数据标准规定数据必须作为对象定义的一部分，　　SQL Server 将自动确保数据符合标准。　　使用这种方法实现数据完整性简单而且不易出错。　　系统将实现数据完整性的要求直接定义在表上或列上。在　　SQL Server 中可以通过使用约束、默认和规则实现声明数据完整性。

（2）　过程定义数据完整性

过程定义数据完整性是指通过编写用来定义数据必须满足的标准和强制该标准的脚本来实现数据完整性。过程定义数据完整性通常用于复杂的商业逻辑中。在　　SQL Server 中可以通过使用触发器和存储过程来实现过程定义数据完整性。

6．2 定义约束

6．2．1 约束的类型

约束是一种强制数据完整性的标准机制。　　使用约束可以确保在字段中输入有效数据并维护各表之间的关系。　SQL Server 2000 支持下列五类约束。

（1）DEFAULT约束（默认约束）：当向数据库表中插入数据时，如果没有明确的提供输入值时，　SQL Server 自动为该列输入指定值。

（2）CHECK约束（检查约束）：通过逻辑表达式判断限制插入到列中的值。

（3）PRIMARYKET约束（主键约束）：不允许数据库表在指定的列上具有相同的值，且不允许有空值。

（4）FOREIGNKEY约束（外键约束）：定义数据库表中指定列上插入或更新的数值必须在另一张被参照表中的特定列上存在。

（5）UNIQUE约束（惟一约束）：不允许数据库表在指定列上具有相同的值，但允许有空值。

约束也被分为列约束和表约束两类。　　列约束是指只对某一列起作用的约束。　　当一个约束中包含了数据库表中一个以上的列时，称为表约束。

6. 2. 2 约束的创建

```
CREATE TABLE table_name
(column_name  date_type (NULL|NOT NULL)
[[CONSTRAINT constraint_name]
{
PRIMARY KEY [CLUSTERED|NONCLUSTERED]
|UNIQUE [CLUSTERED|NONCLUSTERED]
|[FOREIGN KEY] REFERENCES ref_table[(ref_column)]
|DEFAULT constant_expression
|CHECK(logical_expression)
}
][, …]
)
```

例 6-1　在 tsing_DB1 数据库中创建一张用于教师信息管理的表　　teacher ，表中包括的教师信息分别为：教师编号、教师姓名、性别、出身年月、所在系代号、职称、办公室电话号

码、科研方向以及工作状态，在创建时定义有列约束和表约束。

```
USE tsing_DB1
Go
CREATE TABLE tsing_DB1.dbo.teachers
 (
  TeacherID int NOT NULL,
  Name nvarchar(5) NOT NULL,
  Gender nchar(1) NULL,
  Birthday datetime NULL,
  DeptCode tinyint NOT NULL,
  Title nvarchar(5) NULL,
  TelCode char(8) NOT NULL,
  Aspect nvarchar(200) NULL,
 Status nvarchar(5) NOT NULL CONSTRAINT  DE_Status DEFAULT(  ‘在职’),
CONSTRAINT PK_Teacher PRIMARY KEY CLUSTERED(TeacherID),
CONSTRAINT FK_DeptCode FOREIGN KEY(DeptCode)
REFERENCES dbo.departments(DeptCode),
CONSTRAINT CK_TelCode CHECK(Telcode LIKE ‘627[0-9] [0-9] [0-9] [0-9] [0-9]  ’),
)
go
CREATE TABLE tsing_DB1.dbo.deparments
(
DepeCode tinyint NOT NULL Primary Key,
DeptName nchar(20) NOT NULL,
TelCode char(8) NULL
)
```

6. 2. 3 查看约束的定义信息

6. 2. 4 删除约束

6. 3 约束类型

6. 3. 1 DEFAULT约

6. 3. 2 CHECK约束

6. 3. 3 PRIMARY KEY约束

6. 3. 4 UNIQUE约束

6. 3. 5 FOREIGN KEY约束

6. 4 创建约束的其他选项

6. 5 使用默认

声明数据完整性可以使用约束、默认和规则实现。与在约束中所介绍的 DEFAULT约束一样，使用默认也可以实现当用户在向数据库表中插入一行数据时，如果没有明确给出某列的输入值时，则由 SQL Server 自动为该列输入默认值，但与 DEFAULT约束不同的是，默认是一种数据库对象，在数据库中只需定义一次后，就可以被一次或多次应用于任意表中的一列或多列，还可以用于用户定义的数据类型。

创建默认的命令如下：

```
CREATE DEFAULT default_name
AS constant_expression
```

在默认被创建后，用户必须通过执行系统存储过程　　sp_binddefault　　将其定于列或用户自定义的数据类型上，　从而将默认用于数据库中任意表的一列或多列，　　以及用于用户自定义的数据类型，执行默认绑定的命令如下：

EXEC sp_binddefault default_name,

’table_name.[column_name[,　…]|user_datetype]　　’

例：在数据库　tsing_DB1　上创建默认　DeptCode_default　，并将其绑定在数据库表 Departments　中的　DeptCode 列上，从而实现各系默认电话为　　62780001（校总机）。

USE tsing_DB1

Go

CREATE DEFAULT DeptCode_default

AS ‘62780001’

Go

EXEC sp_binddefault DeptCode_default,　　’Departments.DeptCode　’

go

6. 6 使用规则

通过使用规则，　用户可以指定插入数据库表上列中的有效值，　　从而确保数据在指定的取值范围内，　并与特定的模式或特定数据库表中的实体匹配。　　规则也是一种数据库对象，　因此和默认一样在数据库中只需定义一次，就可以被一次或多次应用于任意表中的一列或多列，也可以用于用户定义的数据类型。

创建规则的命令如下：

CREATE rule rule_name

AS condition_expression

规则被创建后，　用户必须通过执行系统存储过程　　sp_bindrule　将其绑定于列或用户自定义的数据类型上，　从而将规则用于数据库中任意表的一列或多列，　　以及用于用户自定义的数据类型，执行规则绑定的命令如下：

EXEC sp_bindrule rule_name,

’table_name.[column_name[,　…]|user_datetype]　　’

例：在数据库　tsing_DB1　上创建规则　Code_rule ，并将其绑定在数据库表　stud_info　中的 stud_id　列上，从而使学生标识号　　stud_id　为 1 至 50 的自然数（包含　1 和 50）。

USE tsing_DB1

Go

CREATE RULE Code_rule

AS @stud_id>=1 AND @stud_id<=50

Go

EXEC sp_bindrule Code_rule,　　’stud_info.stud_id　　’

Go

6. 7 数据完整性强制方法的选择

第 7 章　索引及其应用

7.1　有关索引的基础知识

7. 2 创建索引的原因和选择索引列

7. 3 索引的分类

7. 4 建立可利用的索引

7. 4. 1 在查询分析器窗口中用　　SQL命令建立索引

例 7-1

```
USE Northwind
Go
CREATE CLUSTERED INDEX CustomerIndex
ON [dbo].[Test_Table]([CustomerID])
ON [PRIMARY]
```

7. 5 索引信息的查询

7. 6 更改索引的名称

7. 7 删除索引

7. 8 设置创建索引的选项

7. 9 索引的分析和维护

7. 10 使用索引优化向导

第 8 章　视图及其应用

8. 1 综述

8. 1. 1 视图的基本概念

视图可以被看成是虚拟表或存储查询。　　可通过视图访问的数据不作为独特的对象存储在数据库内。 数据库内存储的是　SELECT语句，SELECT语句的结果集构成视图所返回的虚拟表。在视图中查询的表被称为“ 基表 ”

8. 1. 2 使用视图的优点和缺点

8. 2 视图的创建和查询

例 8-1

```
USE pubs
Go
CREATE VIEW my_view1
AS
SELECT au_id,phone,job_id,job_desc
FROM authors,jobs
WHERE au_id LIKE ‘1%’AND job_desc LIKE ‘N%’
go
SELECT * FROM my_view1
```

8. 3 视图定义信息查询

8. 4 视图的修改和删除

例 8-4

```
ALTER VIEW my_view2
AS
SELECT au_id,phone,job_id,job_desc
FROM authors,jobs
WHERE au_id LIKE ‘1%’ AND job_desc LIKE ‘M%
```

例 8-5

```
DROP VIEW my_view2
```

8. 5 通过视图修改数据

第 9 章　存储过程及其应用

第 10 章　触发器及其应用