数据库系统概论

An Introduction to Database System

第三章关系数据库标准语言 SQL（1）

第三章　关系数据库标准语言 SQL

3.1 SQL （Structured Query Language）

- ✓SQL 的产生与发展
- ✓SQL 的特点
- ✓SQL 的基本概念

3.2 SQL Server

- ✓SQL Server 的产生与发展
- ✓SQL Server 2008 的新特性
- ✓SQL Server 2008 的安装与配置
- ✓SQL Server 2008 的体系结构
- ✓SQL Server 2008 的管理和开发工具

SQL的产生与发展

- 1970年，美国IBM研究中心的E.F.Codd连续发表多篇论文，提出关系模型。
- 1972年，IBM公司开始研制关系数据库管理系统SYSTEM R，配制的查询语言称为SQUARE（Specifying Queries As Relational Expression）语言，在语言中使用了较多的数学符号。
- 1974年，Boyce和Chamberlin把SQUARE修改为SEQUEL（Structured English QUEry Language）语言。
- 后来SEQUEL简称为SQL（Structured Query Language），即“结构化查询语言”，SQL的发音仍为“sequel”。现在SQL已经成为一个标准 。

• SQL 标准的发展

- 1986年，美国国家标准协会（ANSI）公布了第一个SQL(Standard Query Language)标准，即SQL/86。
- 1987年，国际标准化组织（ISO）通过了该标准。
- 1989年，修改发布SQL/89。
- 1992年，修改发布SQL/1992。
- 1999年，SQL99。
- 2003年，SQL2003。
- 2006年，SQL2006。
- 2008年，SQL2008。

• SQL 的特点

- ✓**综合统一：** 集数据定义语言 DDL 、数据操纵语言 DML 、数据控制语言 DCL 功能于一体
- ✓**高度非过程化：** 无需了解存取路径，提高数据独立性
- ✓**面向集合的操作方式：** 无需说明按照哪种路径、如何循环等具体处理过程。操作对象、结果是元祖的集合
- ✓**以同一种语法结构提供多种使用方式：**
  - 独立语言：可以在终端键盘上直接键入 SQL 命令对数据库进行操作；
  - 嵌入式语言：嵌入到高级语言中，例如 C ， C++ ， Java 程序中，供程序员设计程序是使用。
- ✓**语言简洁，易学易用：** 核心只有 9 个动词，接近英语口语

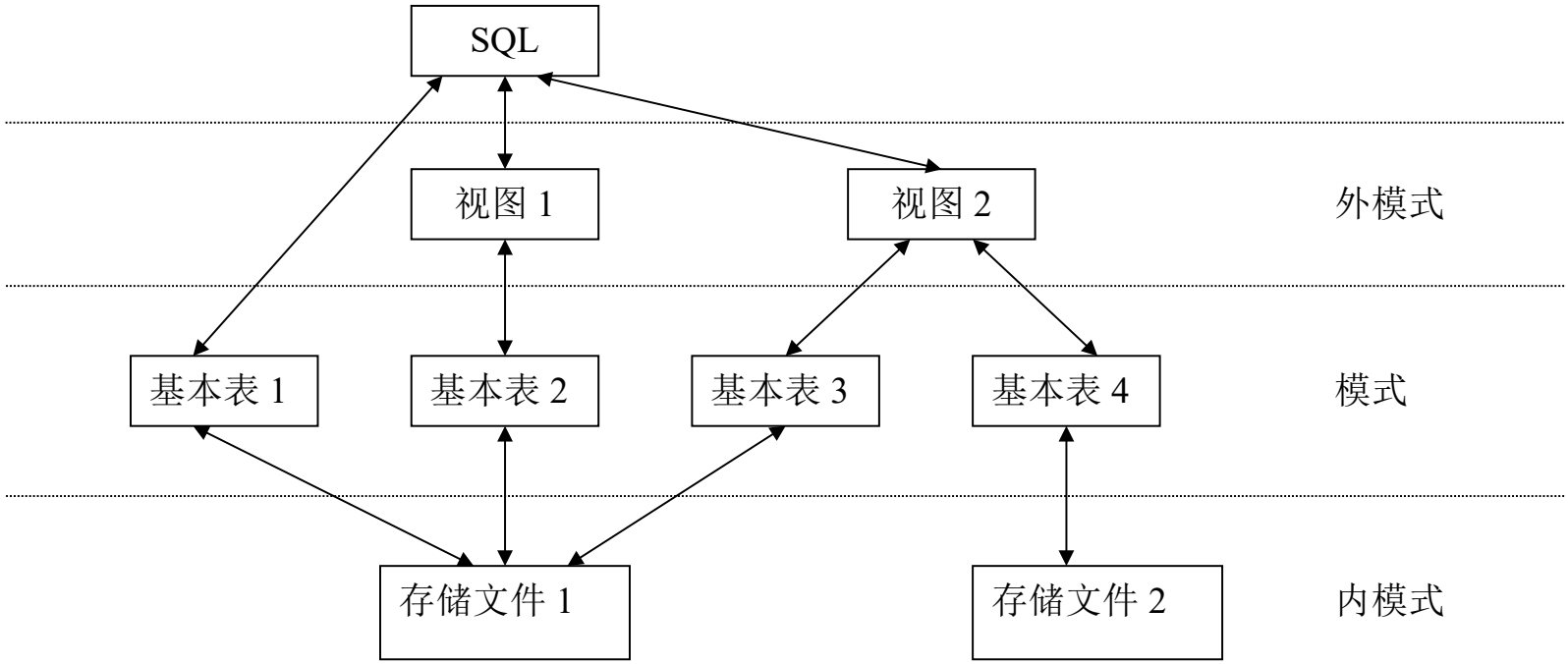
✓ 语言简捷，易学易用

表 3.1　SQL 语言的动词

SQL 功　能	动　　词
数 据 定 义	CREATE, DROP, ALTER
数 据 查 询	SELECT
数 据 操 纵	INSERT，UPDATE，DELETE
数 据 控 制	GRANT，REVOKE

• SQL 语言的基本概念

- 基本表（ Base Table ）和视图（ View ）
- SQL 语言支持数据库的三级模式结构



SQL 语言支持的关系数据库的三级逻辑结构

SQL 语言的基本概念：视图与基本表

1．基本表（Base Table）：

一个关系就对应一个基本表

2．视图（View)：

一个或几个基本表导出的表。本身不独立存储在数据库，即数据库中只存放视图的定义而不存放视图对应的数据。数据仍存放在导出视图的基本表中。因此，**视图是虚表，实际并不存在**，只有定义存放在数据字典中。

用户可在视图上再定义视图，就像在基本表上定义视图一样，因为视图也是关系。

数据库产品

- 数据库产品是由专门开发 DBMS 的厂商提供的。
- 当前，数据库市场上的常见数据库产品包括甲骨文公司的 Oracle 系统， IBM 公司的 DB2 系统和 Informix 系统，赛贝斯公司的 Sybase ASE 系统，微软公司的 Microsoft SQL Server 系统和 Access 系统，以及 MySQL 公司的开源数据库系统等。
- 本书重点讲述微软公司的 Microsoft SQL Server 系统

Microsoft SQL Server 简史

- 通常，把 Microsoft SQL Server 简称为 SQL Server 。
- 严格地说， SQL Server 和 Microsoft SQL Server 是不同的， Microsoft SQL Server 是由微软公司开发的 SQL Server 系统。
- 但是，最早的 SQL Server 系统并不是微软开发出来的，而是由赛贝斯公司推出的。

列表

- 1987 年，赛贝斯公司发布了 Sybase SQL Server 系统
- 1988 年，微软公司、 Aston-Tate 公司参加到了赛贝斯公司的 SQL Server 系统开发中
- 1990 年，微软公司希望将 SQL Server 移植到自己刚刚推出的 Windows NT 系统中
- 1993 年，微软公司与赛贝斯公司在 SQL Server 系统方面的联合开发正式结束
- 1995 年，微软公司成功地发布了 Microsoft SQL Server 6.0 系统
- 1996 年，微软公司又发布了 Microsoft SQL Server 6.5 系统
- 1998 年，微软公司又成功地推出了 Microsoft SQL Server 7.0 系统
- 2000 年，微软公司迅速发布了与传统 SQL Server 有重大不同的 Microsoft SQL Server 2000 系统
- 2005 年 12 月，微软公司艰难地发布了 Microsoft SQL Server 2005 系统
- 2008 年 8 月，微软公司发布了 Microsoft SQL Server 2008 系统

SQL Server 2008 系统主要特点

- 2008 年 8 月，微软公司发布了 Microsoft SQL Server 2008 系统，其代码名称是 Katmai 。
- 该系统在安全性、可用性、易管理性、可扩展性、商业智能等方面有了更多的改进和提高，对企业的数据存储和应用需求提供了更强大的支持和便利。

---------- 参见微软官方网络课程

SQL Server 2008 的安装与配置

- 各版本比较：

- ✓ **SQL Server 2008 企业版**是一个全面的数据管理和业务智能平台，为关键业务应用提供了企业级的可扩展性、数据仓库、安全、高级分析和报表支持。这一版本将为你提供更加坚固的服务器和执行大规模在线事务处理。
- ✓ **SQL Server 2008 标准版**是一个完整的数据管理和业务智能平台，为部门级应用提供了最佳的易用性和可管理特性。
- ✓ **SQL Server 2008 工作组版**是一个值得信赖的数据管理和报表平台，用以实现安全的发布、远程同步和对运行分支应用的管理能力。 这一版本拥有核心的数据库特性，可以很容易地升级到标准版或企业版。
- ✓ **SQL Server 2008 Web 版**是针对运行于 Windows 服务器中要求高可用、面向 Internet Web 服务的环境而设计。这一版本为实现低成本、大规模、高可用性的 Web 应用或客户托管解决方案提供了必要的支持工具。
- ✓ **SQL Server 2008 开发者版**允许开发人员构建和测试基于 SQL Server 的任意类型应用。这一版本拥有所有企业版的特性，但只限于在开发、测试和演示中使用。基于这一版本开发的应用和数据库可以很容易地升级到企业版
- ✓ **SQL Server 2008 Express 版**是 SQL Server 的一个免费版本，它拥有核心的数据库功能，其中包括了 SQL Server 2008 中最新的数据类型，但它是 SQL Server 的一个微型版本。这一版本是为了学习、创建桌面应用和小型服务器应用而发布的，也可供 ISV 再发行使用。
- ✓ **SQL Server Compact 版**是一个针对开发人员而设计的免费嵌入式数据库，这一版本的意图是构建独立、仅有少量连接需求的移动设备、桌面和 Web 客户端应用。 SQL Server Compact 可以运行于所有的微软 Windows 平台之上，包括 Windows XP 和 Windows Vista 操作系统，以及 Pocket PC 和 SmartPhone 设备。 **---------- 参见： SQL Server 2008 安装手册**

SQL Server 2008 系统的体系结构

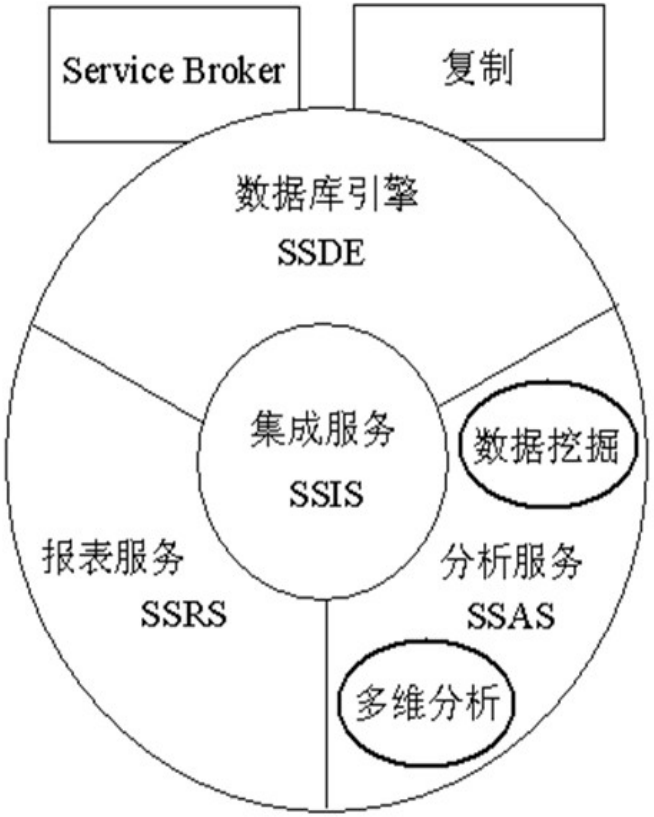


图 1-3　Microsoft SQL Server 2008 系统的体系结构示意图

数据库引擎（Database Engine）： 用于储存、处理和保护数据的核心服务。

分析服务（Analysis Services）：
为商业智能应用程序提供联机分析处理和数据挖掘功能。

集成服务（SQL Server Integration Services）：
用于清理、聚合、合并、复制数据的转换以及管理 SSIS 包。提供生产并调试 SSIS 包的图形向导工具，以及用于执行 FTP 操作、电子邮件消息传递等工作流功能。

报表服务（SQL Server Reporting Services）：
基于服务器的报表平台。

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、数据库的属性、登录账户、数据库文件、数据库备份、警报、作业等信息。 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、用来存放用户数据和对象的数据库。

SQL server 2008 的管理工具

- 主要包括
- **SQL Server Management Studio**：用于编辑和执行查询，以及启动标准向导任务
- **SQL Server** 配置管理器：管理服务器和客户端网络配置设置 **SQL Server Profiler**：提供用于监视 **SQL Server** 数据库引擎实例或 **Analysis Services** 实例的图形用户界面
- 数据库引擎优化顾问：可以协助创建索引、索引视图
- ---------- 见各菜单项

系统数据库

- **master** 是最重要的系统数据库，它记录了 SQL Server 系统级的所有信息，这些系统级的信息包括服务器配置信息、登录账户信息、数据库文件信息、 SQL Server 初始化信息等等，这些信息影响整个 SQL Server 系统的运行。
- **model** 是一个模板数据库。该数据库存储了可以作为模板的数据库对象和数据。当创建用户数据库时，系统自动把该模板数据库中的所有信息复制到用户新建的数据库中，使得新建的用户数据库初始状态下具有了与 model 数据库一致的对象和相关数据，从而简化数据库的初始创建和管理操作。
- **msdb** 是与 SQLServerAgent 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，这些信息可以用于自动化系统的操作。
- **tempdb** 是一个临时数据库，用于存储查询过程中所使用的中间数据或结果。实际上，它只是一个系统的临时工作空间。

数据库文件分类

- 数据库以文件的形式存放在磁盘上，根据其作用不同，可分为以下三种类型：
- 1 ）主数据文件（ primary file)
- 是数据库的关键文件，用来存储数据和数据库的启动信息。每个数据库必须有且仅有一个主数据文件，其扩展名为 .MDF 。

- 2）辅助数据文件（secondary file)
- 用来存放数据。使用它可以扩展存储空间。若只使用主数据文件，则文件的最大容量受一个磁盘容量的限制。但使用辅助数据文件就可以将他们放到不同的磁盘上，不再受一个磁盘的限制。
- 每个数据库可有 0 个或多个辅助数据文件，其扩展名为 .NDF 。

- 3）事务日志文件（transaction log)
- 用于保存恢复数据库所需的事物日志信息，即存放数据库的修改信息。

（如： insert ,update,delete…..) 都会记录在其中。当数据库被破坏时，可利用事务日志文件来恢复数据库的内容。

每个数据库有一个或多个事务日志文件，其扩展名为 .LDF 。

- 创建数据库时，一个数据库至少由一个主数据文件和一个事务日志文件组成

- 主数据文件名为 “ 数据库名 - DATA.MDF ”
- 事务日志文件名为 “ 数据库名 - LOG.LDF ”

3.3 SQL 数 据 定 义

SQL 的数据定义功能包括模式定义、表定义、视图和索引定义。（SQL 通常不提供修改模式定义、修改视图定义和修改索引定义的操作。用户如果想修改这些对象，只能先将它们删除掉，然后再重建。）

表 3.2 SQL 的数据定义语句

操作对象	操作方式		
	创建	删除	修改
模式	CREATE SCHEMA	DROP SCHEMA	
表	CREATE TABLE	DROP TABLE	ALTER TABLE
视图	CREATE VIEW	DROP VIEW	
索引	CREATE INDEX	DROP INDEX	

3.3.1 模式的定义与删除

定义模式

CREATE SCHEMA < 模式名 >
AUTHORIZATION < 用户名 >

如果没有指定 < 模式名 > ，则隐含为 < 用户名 > 。

要创建模式，调用该命令的用户必须具有 **DBA** 权限，或者获得了 **DBA** 授予的 **CREATE SCHEMA** 的权限。

例：定义一个学生 - 课程模式 **S-T**

CREATE SCHEMA “S-T” AUTHORIZATION WANG;

例：为用户 **WANG** 定义一个模式 **S-T**

CREATE SCHEMA AUTHORIZATION WANG

(< 模式名 > 隐含为 **WANG**)

3.3.1 模式的定义与删除

二、删除模式

DROP SCHEMA < 模式名 > <CASCADE| RESTRICT>

其中 CASCADE 和 RESTRICT 两者必选其一。

CASCADE （级联）： 在删除模式的同时把该模式中所有的数据库对象全部一起删除

RESTRICT （限制）： 表示如果该模式中已经定义了下属的数据库对象（如表、视图等），则拒绝该删除语句的执行。

例： DROP SCHEMA ZHANG CASCADE

该语句删除了模式 ZHANG，同时该模式中已经定义的表 TAB1 也被删除了。

- 补充：一个 SQL 模式由模式名和模式拥有者的用户名或帐号来确定并包含模式中每一个元素（基本表、视图、索引等）。大多数的 DBMS 不用“ SQL 模式”这个名词而是采用“数据库”（ DATABASE ）这个名词。

创建、修改、删除数据库

两种方法

- 企业管理器向导

- 查询分析器语句

CREATE DATABASE < 数据库名 >

……

DROP DATABASE < 数据库名 >

……

ALTER DATABASE < 数据库名 >

……

Create database 语句格式

- Create database 数据库名

[on [primary]

{ ([name= 数据文件的逻辑名称 ,]

filename=‘ 数据文件的物理名称’

[,size= 数据文件的初始大小]

[,maxsize= 数据文件的最大容量]

[,filegrowth= 数据文件的增长量] ） } [, …n]

- [Log on
{ ([name = 事务日志文件的逻辑名称 ,]
filename = ‘ 事务日志文件的物理名称’
[,size = 事务日志文件的初始大小]
[,maxsize = 事务日志文件的最大值]
[,filegrowth = 事务日志文件的增长量])} [,…n]]

- 例：创建一个名为 StuTestDatabase

的数据库，包含一个主数据文件和一个事务日志文件。主数据文件的逻辑文件名为 StuData，物理文件名为 StuData_data.mdf，初始容量大小为 10MB，最大容量为 50MB，每次增长量为 25 %。

事务日志文件的逻辑名为 StuLog，物理文件名为 StuData_log.ldf，初始容量大小为 10MB，最大容量不受限制，每次增长量为 2MB. 要求这两个文件都存放在“ d:\data”目录下。注意：权限、附加。。。。。。查询的保存。。。。。。

在查询分析器的编辑窗口输入以下代码：

```
CREATE  DATABASE StuTestDatabase
ON PRIMARY
      (NAME=StuData,
FILENAME='d:\data\StuData_data.mdf',
      SIZE=10MB,
      MAXSIZE=50MB,
      FILEGROWTH=25%)
LOG ON
```

- (NAME=StuLog,
- FILENAME='d:\data\StuData_log.ldf',
- SIZE=10MB,
- MAXSIZE=UNLIMITED,
- FILEGROWTH=2MB)

查看数据库

- 利用 sp_helpdb 函数可以动态查看当前服务器中的数据库或指定的数据库。
- 语法如下：
- sp_helpdb [[@dbname=]'name']

重命名数据库

- 利用 sp_renamedb 函数可以动态重命名指定的数据库。
- 语法如下：
- sp_renamedb [[@dbname=]'old_name'],
- [@newname=]'new_name'

查看数据库文件

- 利用 sp_helpfile 函数可以动态查看逻辑文件的信息。若不指定文件名，则显示数据库中所有文件的信息。语法如下：
- sp_helpfile [[@dbname=]'name']
- 例：查看 student 中逻辑文件信息
- use student
- go
- sp_helpfile student
- 例：查看 student 中所有文件信息
- use student
- go
- sp_helpfile

修改数据库

- 使用 **ALTER DATABASE** 命令可对数据库进行如下修改：

✓增加或删除数据文件

✓改变数据文件的大小和增长方式

✓改变日志文件的大小和增长方式

✓增加或删除日志文件

✓增加或删除文件组

修改数据库

- 语法格式：
- **Alter database database_name**
- **add file<filespec>**
- **add log file <filespec>**
- **remove file <filespec>**
- **modify file <filespec>**

修改数据库 举例

- 例 1：修改数据库文件 TEST1，将主数据文件的最大大小改为 100MB，增长方式改为按每次 5MB 增长。
- Alter database TEST1
- modify file

```
(
name = ‘test1_data’,
maxsize = 100MB,
filegrowth = 5MB
)
```

修改数据库 举例

- 例 2：为数据库文件 TEST1 增加数据文件 TEST1BAK 。
- Alter database TEST1
- add file

(

name = ‘TEST1BAK’,

filename = ‘D:\data\TEST1BAK.ndf’,

size = 50MB,

maxsize = 100MB,

filegrowth = 5%

)

修改数据库 举例

- 例 3：删除例 2 中所建文件 TEST1BAK

```
alter database TEST1
remove file TEST1BAK
```

- 例 4：从数据库 TEST1 中删除一个日志文件
- 注意：不能删除主日志文件

```
alter database TEST1
remove file TEST1_LOG2
```

删除数据库

- 语法格式：
- drop database database_name,[,…n][;]
- 例：
- 删除数据库 TEST2
- drop database TEST2
- 一般使用带有判断条件的数据库动态删除
- If Exists(select * from dbo.sysobjects where name='db_modsql')
- Drop database db_modsql
- Else
- Print ' 要删除的数据库不存在'
- dbo.sysobjects 也可以写成 sys.databases

3.3.2 基本表的定义、删除与修改

- 企业管理器向导创建

- 查询分析器语句创建
  - 一、定义表语句的格式
  - 二、数据类型
  - 三、完整性约束
  - 四、修改表的格式
  - 五、删除表的格式

3.3.2 基本表的定义、删除与修改

- 企业管理器向导创建

- 查询分析器语句创建
  - 一、定义表语句的格式
  - 二、数据类型
  - 三、完整性约束
  - 四、修改表的格式
  - 五、删除表的格式

3.3.2 基本表的定义格式

一、定义基本表

CREATE TABLE < 表名 >

（ < 列名 > < 数据类型 >[< 列级完整性约束条件 >]
[，< 列名 > < 数据类型 >[< 列级完整性约束条件 >]] …
[，< 表级完整性约束条件 >] ）；

- **< 表名 >**：所要定义的基本表的名字
- **< 列名 >**：组成该表的各个属性（列）
- **< 列级完整性约束条件 >**：涉及相应属性列的完整性约束条件
- **< 表级完整性约束条件 >**：涉及一个或多个属性列的完整性约束条件

一、创建表语句格式

数据类型 [(长度)] [NULL | NOTNULL]

[IDENTITY(初始值，步长值）]

其中列约束的格式为：

[CONSTRAINT 约束名] PRIMARY KEY[(列名)]

指定列值必须是唯一的，且不能为空值。

[CONSTRAINT 约束名] UNIQUE [(列名)]

指定列值的所有值不能重复，且可以为空值。

[CONSTRAINT 约束名] [FOREIGEN KEY [(外键列)]]
REFERENCES 引用表名 (引用列)

[CONSTRAINT 约束名] CHECK (检查表达式）

验证列值服从某些规则，即规定改列的取值范围

[CONSTRAINT 约束名] DEFAULT 默认值。

[例] 建立一个“学生”表 Student，它由学号 Sno、姓名 Sname、性别 Ssex、年龄 Sage、所在系 Sdept 五个属性组成。其中学号不能为空，值是唯一的，并且姓名取值也唯一。

```
CREATE TABLE Student
(
   Sno    CHAR(5)  NOT NULL  UNIQUE,
   Sname  CHAR(20)  UNIQUE,
    Ssex     CHAR(1) ,
    Sage     INT,
    Sdept    CHAR(15)
);
```

Sno	Sname	Ssex	Sage	Sdept

↑	↑	↑	↑	↑
字符型	字符型	字符型	整数	字符型
长度为5	长度为20	长度为1		长度为15
不能为空值				

在 SQL Server 中创建表有如下限制：

① 每个数据库里最多有 20 亿个表；
② 每个表上最多可以创建一个聚集索引和 249 个非聚集索引；
③ 每个表最多可以包含 1024 个字段；
④ 每条记录最多可以占用 8060 字节，但不包括 text 字段和 image 字段。

二、数据类型

- 当用 SQL 语句定义表时，需要为表中的每一个字段设置一个数据类型，用来指定字段所存放的数据是整数、字符串、货币或是其它类型的数据。
- SQL SERVER 的数据类型有很多种，分为以下 9 类：

1. 整数数据类型：依整数数值的范围大小，有Bigint, int , smallint, tinyint 四种。

- **Bigint: 存放 -2^{63}~$2^{63}-1$ 数值范围内的整型数据**
- **Int: 存放 -2^{31}~$2^{31}-1$ 数值范围内的整型数据**
- **Smallint: 存放 -2^{15}~$2^{15}-1$ 数值范围内的整型数据**
- **Tinyint：存放 0~255 数值范围内的整型数据**

2. **精确数值类型：用来定义可带小数部分的数字，有 NUMERIC 和 DECIMAL 两种。二者相同，但建议使用 DECIMAL 。如： 123.0 、 8000.56**

Decimal(p,s): p 为精度，表示不算小数点，总输出的位数； s 为小数点后的位数。其取值范围为

-10^{38}~ $10^{38}-1$

- **3. 近似值型数据：**
- **该类型用于存储浮点数，包括 float 和 real 两种**
- **float(n)：其中 n 为精度，表示总输出位数。**
- **这种近似数值型数据不能确定所输出的数值精确度。**

4. **日期时间数据类型：**用来表示日期与时间，依时间范围与精确程度可分为 DATETIME 与 SMALLDATETIME 两种。

- Datetime：存放 1/1/1753~12/31/9999 的时间数据，精确到 0.001s 。
- Smalldatetime：存放 1/1/1900~6/6/2079 的时间数据，精确到分。

5. 字符型数据类型：用来存储汉字、英文字母、数字、标点、和各种符号，包括： CHAR, VARCHAR, TEXT 三种，如：“数据库”

Char(n): 按固定长度存储字符数据

Varchar (n): 按变长存储字符数据

Text： 可存储最大长度为 $2^{31}-1$ 个字节字符数据。

前两种字符型数据中，如果不指定 n 值，其默认值为 1 。

在输入字符型数据时要用单引号括起来。

- 6. 统一字符型数据（ unicode)
- 包括 3 种数据：定长字符型 nchar 、变长字符型 nvarchar 、和文本类型 ntext 。

Nchar(n): 按固定长度存储 n 个 Unicode 数据

Nvarchar (n): 存放可变长度的 n 个 Unicode 数据

Ntext : 存储最大长度为 $2^{30}-1$ 个字节 Unicode 数据

统一字符型数据 与 字符型数据 区别：

如： char(2) 最多可存放 2 个英文字母或 1 个汉字；

Nchar(2) 则最多存放 2 个汉字或 2 个英文字母或 1 个汉字 1 个英文字母

7. **二进制数据类型：**用来定义二进制码的数据。有： BINARY, VARBINARY ， IMAGE 三种，通常用十六进制表示：如： OX5F3C

BINARY （ n) ：按固定长度存储二进制数据。若输入的数据不足 n 个字节，补足后存储； 若超过 n 个字节，则截断后存储。总是用 n 个字节存储数据。

VARBINARY （ n): 按变长存储二进制数据。若输入数据不足 n 个字节，按实际长度存储；若超过 n 个字节，则截断个存储。

IMAGE ： 可存储最大长度为 $2^{31}-1$ 个字节的二进制数据。

- **8. 货币型数据**
- **用于存储货币数据，包括 money 和 smallmoney 两种。这些数据类型实际上都是带有 4 位小数的 decimal 类型数据。**
- **输入货币型数据时要在数据前加货币符号。若为负值，则在货币符号后面加符号。**

- 9. 位类型数据（ bit)
- 用于存储整数，只能取值 0 、 1 或 null（空值），可用于逻辑判断中。
- 在位类型的字段中输入 0 和 1 以外的任何值，系统都会作为 1 来处理。

三、常用的完整性约束

- 主键约束（ PRIMARY KEY)
- 唯一约束（ UNIQUE)
- 外键约束 (FOREIGN KEY)
- 检查约束 （ CHECK)
- 默认值约束（ DEFAULT)
- 空值约束（ NULL)

1. 主键约束 (primary key)

- 主键约束是用来保证表中记录唯一可区分的列。一个表可以通过一列或列组合的数据来唯一标识表中的每一条记录。
- 创建了主键约束的列有如下特点：
- 1）每一表仅能定义一个主键，主键值是表中记录的
- 标识。
- 2）主键列可以有一个或多个列组合而成。
- 3）主键值不可为空。
- 4）主键值不可重复。若主键是多列组成时，某一列
- 上的数据可以重复，但多列的组合值不能重复。
- 5） image 和 text 类型的列不能做主键。

2. 唯一约束 (unique)

- 规定一条记录的一个字段值或几个字段的组合值不得与其他记录的相同字段或字段组合的值重复，将这种限制成为“唯一约束”。
- 唯一约束是为了保证除主键列外的其他列的数据不重复。它可以由一个列或多个列组成。

唯一约束和主键约束的区别：

- 1）一个表中只能定义一个主键约束，但可以定义多个唯一约束。
- 2) 如果在某列创建了主键约束，表中记录在磁盘的存放顺序就以该列的值从小到大的顺序存放，而唯一约束则不改变记录的物理存放顺序，仅保证该列的值不重复。
- 3）定义了唯一约束的列数据可以为空值，而定义了主键约束的列数据不能为空值。

例：创建含有主键、唯一键字段的数据库表

- use db_mysql
- Create table db_table3
- (

  仓库标号　int primary key,

  仓库号　varchar(50) unique,

  城市　varchar(50) not null,

  面积　int

  );

3. 外键约束 (foreign key)

- 特点如下：
- 1）外键列可以由是一个列或多个列组成。
- 2）外键列的取值可以为空，可以有重复值，但必须是它所引用列的列值之一。引用列必须是创建了主键约束或唯一约束的列。

[例] 建立一个“学生选课”表 SC，它由学号 Sno、课程号 Cno，修课成绩 Grade 组成，其中 (Sno, Cno) 为主码。

```
CREATE TABLE SC
(    Sno CHAR(5) ,
     Cno CHAR(3) ,
     Grade   int,
     Primary key (Sno, Cno),
     Foreign key (sno) References student (sno),
     Foreign key (cno) References course (cno)  );
```

表级完整性约束

4. 检查约束 (check)

- 检查约束是用来检查一个字段或多个字段的输入值是否满足指定的约束条件。
- 使用逻辑表达式来限制某列上可以接受的数值范围，将这种对字段输入值的限制成为“检查约束”。
- 例：

```
use test
Create table db_table7
(
        仓库编号  int primary key,
    仓库号  varchar(50) unique,
    面积  int check( 面积 >=600 and  面积 <=1800)
);
```

5. 默认值约束（ default)

- 在数据库中建立一个默认值并把该默认值绑定到表中某字段或用户定义数据类型时，如果用户在插入记录时没有明确提供该字段数值，系统便自动将默认值赋给该字段，这种对字段数值的限制被称为“默认值”约束。
- 在用户定义数据类型的情况下，如果使用默认值约束，则默认值被插入到使用这个自定义数据的所有字段中。
- 默认值可以是常量、内置函数或表达式。
- 例 : Use test
- Create table db_table10
- (
- 仓库编号 int primary key,
- 仓库号 varchar(50) unique,
- 城市 varchar(50) default '青岛'
-);

创建含有计算字段的数据库表

```
use test
create table db_table9
(
        职工编号  int primary key,
        职工号  varchar(50) unique,
        仓库号  varchar(50),
        基本工资  int check( 基本工资 >=800 and  基本工资 <=2100),
        加班工资  int,
        奖金  int,
        扣率  int,
        应发工资  as( 基本工资 + 加班工资 + 奖金 - 扣率 )
);
```

- 如果数据表中某字段具有默认值，那么向表中插入记录时，如果该字段的值没有填写，则最后显示为默认值；计算字段是由其他字段通过运算得来的，该字段中的数据不能进行输入。

创建含有自动编号字段的数据库表

```
use test
go
create table db_table11
(
        仓库编号   int identity(1,1) primary key,
        仓库号   varchar(50) unique,
        城市   varchar(50) default '青岛',
        面积   int check( 面积 >=300 and  面积 <=1800)
 );
```

6. 空值约束（ null ）

- 空值约束就是指尚不知道或不确定的数据值，它不等同于 0 或空格。

- 记

例： 试创建 customers 表（在 market 中），

录客户的基本信息。设置客户编号字段为主键约束，电话字段为唯一约束，姓名字段不能为空。

- USE market
- GO
- CREAT TABLE customers
- （客户编号 INT PRIMARY KEY,
- 姓名 VARCHAR (20) NOT NULL,
- 电话 VARCHAR (50) unique,
- 出生日期 DATETIME)

- 例 2 ：在 market 中创建 orders 表，记录客户定购货品的订单信息，设置订单号字段为自动编号，初始值为 1，递增量为 1，同时设置主键。设置客户编号字段为外键约束。数量字段检查约束为数量大于 0，同时不能为空。订货日期设为默认值，为系统日期。

- USE market
- GO

- **CREATE TABLE orders**
- **(订单号　　int IDENTITY(1,1) PRMARY KEY,**
- **　货品名称　varchar(20) NOT NULL FOREIGN KEY REFERENCES goods(货品名称），**

客户编号　int FOREIGN KEY

REFERENCES customers (客户编号），

数量　　int NOT NULL CHECK (数量 >0),

总金额　　money,

订货日期　datetime DEFAULT getdate ()　);

四、修改基本表

ALTER TABLE < 表名 >

[ADD < 新列名 > < 数据类型 > [完整性约束]]
[ADD CONSTRAINT < 完整性约束名 >< 约束类型 >]
[DROP COLUMN < 列名 >[,...n]]
[DROP CONSTRAINT < 完整性约束名 >]
[ALTER COLUMN < 列名 > < 数据类型 >] ;

- **< 表名 >：要修改的基本表**
- **ADD 子句：增加新列或新的完整性约束条件**
- **DROP 子句：删除指定的完整性约束条件**
- **ALTER COLUMN 子句：用于修改列名和数据类型**
- **DROP COLUMN 字句：用于从表中删除一个或多个字段**

例题

[例] 向 Student 表增加“入学时间”列，其数据类型为日期型。

- ALTER TABLE Student
- ADD Scome DATETIME
- 不论基本表中原来是否已有数据，新增加的列一律为空值（新添列可以设置为允许为空）。若新添列设置不允许为空时，必须给该列指定默认值， SQL Server 就将该默认值送给已存在数据的新添列，否则操作将失败。

[例] 将年龄的数据类型改为半字长整数。

- ALTER TABLE Student
- ALTER COLUMN Sname SMALLINT

  – 注：修改原有的列定义有可能会破坏已有数据

例题

[例] 删除学生姓名必须取唯一值的约束。

ALTER TABLE Student

DROP UNIQUE(Sname) ;

表示在 SQL2008 中不合法

sp_helpconstraint student

alter table student

drop constraint IX_student

- 例：删除 customers 表中“电子邮件”字段
- USE market
- ALTER TABLE customers
- DROP COLUMN 电子邮件

- 注意：在删除列时，必须先删除基于该列的索引和约束后，才能删除该列

3.3.2 建立与删除索引

索引是一种特殊类型的数据库对象，它与表有着密切的关系。索引用来提高表中数据的查询速度

- SQL Server 使用索引来实现表或视图中记录或字段的唯一性约束，对数据进行物理排序或把数据展开到多个文件组中，以此来提高检索效率。

- 查询右表编号为 003 的职工

编号	姓名	职业编号
006	Aa	01
002	Bb	08
005	Cc	03
007	Dd	05
003	Ee	09
001	Ff	15
004	gg	04

方法有二：全表查询，索引查询。

编号	指针地址
001	6
002	2
003	5
004	7
005	3
006	1
007	4

在编号字段上建立索引

索引分类

1. 唯一索引 和 非唯一索引

- 唯一索引要求所有数据行中任意两行中的被索引列不能存在重复值。也不能有两个 **null,**

  而非唯一索引则不存在这一限制。

例如：如果一个表中的一个字段或多个字段的组合在多行记录中具有 **null** 值（相同），则不能将这个字段或字段组合作为唯一索引键。

2. 聚集索引和非聚集索引

- 聚集索引改变记录的物理存储顺序，使之与索引列的顺序完全相同。
- 非聚集索引不改变表记录的存放顺序。

- 由于一个表中的数据只能按照一种顺序来存储，所以一个表中只能建立一个聚集索引。

  建立聚集索引后，更新该索引列上的数据时，往往导致表中的记录物理顺序的变更，代价较大，所以经常更新的列不适合建立聚集索引.

建立索引的方法

- 一、通过企业管理器
- 二、查询分析器中创建语句

3.3.2 建立与删除索引

建立索引

- DBA 或表的属主（即建立表的人）根据需要建立
- 有些 DBMS 自动建立以下列上的索引
  - PRIMARY KEY
  - UNIQUE

- 维护索引
  - DBMS 自动完成
- 使用索引
  - DBMS 自动选择是否使用索引以及使用哪些索引

一、建立索引语句格式

- **CREATE [UNIQUE]** [CLUSTERED]
- **INDEX < 索引名 > ON**
- **< 表名 >(< 列名 >[<ASC | DESC >] [, … n])**

  - 用 < 表名 > 指定要建索引的基本表名字
  - 索引可以建立在该表的一列或多列上，各列名之间用逗号分隔
  - 用 < 次序 > 指定索引值的排列次序，升序： **ASC** ，降序： **DESC** 。缺省值： **ASC**
  - **UNIQUE** 表明此索引的每一个索引值只对应唯一的数据记录
  - **CLUSTERED** 表示要建立的索引是聚簇索引

[例 14] 为学生 - 课程数据库中的 Student，Course，SC 三个表建立索引。其中 Student 表按学号升序建唯一索引，Course 表按课程号升序建唯一索引，SC 表按学号升序和课程号降序建唯一索引。

```
CREATE UNIQUE INDEX  Stusno ON Student(Sno)；
CREATE UNIQUE INDEX Coucno ON Course(Cno)；
CREATE UNIQUE INDEX SCno
        ON SC (Sno ASC，Cno DESC)；
```

建立索引 几点说明

- 唯一值索引
  - 对于已含重复值的属性列不能建 **UNIQUE** 索引
  - 对某个列建立 **UNIQUE** 索引后，插入新记录时 **DBMS** 会自动检查新记录在该列上是否取了重复值。这相当于增加了一个 **UNIQUE** 约束

建立索引 （续）

- **聚簇索引**
  - 建立聚簇索引后，基表中数据也需要按指定的聚簇属性值的升序或降序存放。也即聚簇索引的索引项顺序与表中记录的物理顺序一致

[例]：

CREATE CLUSTERED INDEX Stusname ON S(Sname)；

在 S 表的 Sname （姓名）列上建立一个聚簇索引，而且 Student 表中的记录将按照 Sname 值的升序存放

二、删除索引

DROP INDEX 表名 . 索引名 [, ... n] ;

– 删除索引时，系统会从数据字典中删去有关该索引的描述。

[例] 删除 **S** 表的 **Stusname** 索引。

DROP INDEX S. Stusname ;

习题

- 创建表 Student，满足如下要求：

✓ 学号为主键

✓ 姓名唯一且不空

✓ 性别为男或者女，默认为女

✓ 出生日期为日期型

创建表 PJCJ，满足如下要求：

✓ 课程号为主键，可自动编号

✓ 总成绩不空

✓ 人数不空，且大于 0

✓ 平均成绩为总成绩 / 人数，自动计算。

```
create table Student
(
        Sno int primary key,
        Sname char(6) unique not null,
        Sex char(2) default '女',
        Age datetime
);
create table PJCJ
(
        Cno int primary key identity(1,1),
        Score int not null,
        Number int not null check(Number>0),
        AverageScore as Score/Number
);
```