数据库系统概述

一、有关概念

1．数据

2．数据库（DB）

3．数据库管理系统（DBMS）

桌面 DBMS { Access, VFP }

客户机/服务器型 DBMS { SQL Server, Oracle, MySQL, DB2 }

4．数据库系统（DBS）

{ 数据库（DB）, 数据库管理系统（DBMS）, 开发工具, 应用系统 }

二、数据管理技术的发展

1．数据管理的三个阶段

	人工管理	文件系统	数据库系统
数据能否保存	不能保存	可以保存	可以保存
数据面向的对象	某一应用程序	某一应用程序	整个应用系统
数据的共享程度	无共享，一组数据只能对应一个应用程序。	共享性差，一个数据文件只能对应一个应用程序。	共享性高
数据的独立性	不独立，它是应用程序的一部分。	独立性差	数据库与应用系统完全分开

概念模型

一、模型的三个世界

1．现实世界

2．信息世界：即根据需求分析画概念模型（即 E-R 图），E-R 图与 DBMS 无关。

3．机器世界：将 E-R 图转换为某一种数据模型，数据模型与 DBMS 相关。

注意：信息世界又称概念模型，机器世界又称数据模型

二、实体及属性

1．实体：客观存在并可相互区别的事物。

2．属性：

3．关键词（码、key）：能唯一标识每个实体又不含多余属性的属性组合。

一个表的码可以有多个，但主码只能有一个。

例：借书表（学号，姓名，书号，书名，作者，定价，借期，还期）

规定：学生一次可以借多本书，同一种书只能借一本，但可以多次续借。

4．实体型：即二维表的结构

例 student(no，name，sex，age，dept)

5．实体集：即整个二维表

三、实体间的联系：

1．两实体集间实体之间的联系

- 1：1 联系
- 1：n 联系
- m：n 联系

2．同一实体集内实体之间的联系

- 1：1 联系
- 1：n 联系
- m：n 联系

四、概念模型（常用 E-R 图表示）

实体型：

属性：

联系：

说明：① E-R 图作为用户与开发人员的中间语言。

② E-R 图可以等价转换为层次、网状、关系模型。

举例：

学校有若干个系，每个系有若干班级和教研室，每个教研室有若干教员，其中有的教授和副教授每人各带若干研究生。每个班有若干学生，每个学生选修若干课程，每门课程有若干学生选修。用 E-R 图画出概念模型。

数据模型

一、层次模型：用树型结构表示实体之间的联系。

① 每个结点代表一个实体型。

② 只能直接处理一对多（含一对一）的实体关系。

③ 查找层次数据库中的记录，速度较慢。

二、网状模型：用图结构表示实体之间的联系。

① 每个结点代表一个实体型。

② 可以处理多对多的实体关系。

③ 查找网状数据库中的记录，速度最快。

三、关系模型：用二维表表示实体之间的联系。

1．重要术语：

关系：一个关系就是一个二维表；

元组：二维表的一行，即实体；

关系模式：在实体型的基础上，注明主码。

关系模型：指一个数据库中全部二维表结构的集合。

2．特点：

① 关系模型是建立在严格的数学理论的基础上的；

② 关系模型的存取路径对用户透明；

③ 查找关系数据库中的记录，速度最慢。

小结：数据有三种类型，DBMS 就有三种类型，DB 亦有三种类型。

数据库系统结构

一、数据库系统的**体系结构**

① 单机结构：

DBMS、数据库、开发工具、应用系统安装在一台计算机上。

② C/S 结构：局域网结构

客户机：装开发工具、应用系统

服务器：装 DBMS、数据库

③ B/S 结构：Internet 结构

服务器：装 DBMS、数据库、开发工具、应用系统

客户机：装 IE 即可

三、 数据库系统的**模式结构**

1. 三级模式

① 模式：是数据库中全体数据的逻辑结构和特征的描述。

➲ 模式只涉及数据库的结构；

➲ 模式既不涉及应用程序，又不涉及数据库结构的存储；

② 外模式：是模式的一个子集，是与某一个应用程序有关的逻辑表示。

特点：一个应用程序只能使用一个外模式，但同一个外模式可为多个应用程序使用。

③ 内模式：描述数据库结构的存储，但不涉及物理记录。

2. 两级映象

① 外模式/模式映象：保证数据库的逻辑独立性；

② 模式/内模式映象：保证数据库的物理独立性；

3. 两级映象的意义

① 使数据库与应用系统完全分开，数据库改变时，应用系统不必改变。

② 数据的存取完全由 DBMS 管理，用户不必考虑存取路径。

数据库管理系统

1. DBMS 的功能：负责对数据库进行统一的管理与控制。

① 数据定义：即定义数据库中各对象的结构

② 数据操纵：包括对数据库进行查询、插入、删除、修改等操作。

③ 数据控制：包括安全性控制、完整性控制、并发控制、数据库恢复。

2. DBMS 的组成：

- DDL 语言
- **DML 语言**
- **DCL 语言**
- 实用程序

注意：

① SQL 集 DDL，DML，DCL 功能于一体；

② 所有应用程序通过 SQL 语句才能访问数据库

一、 基本概念

1．码：能唯一标识元组的属性集。

2．候选码：一个属性集既能唯一标识元组，且又不含有多余属性，一个关系模式可以有多个候选码。

3．主码：任选候选码中的一个。

4．主属性：主码中包含的各个属性。

5．非主属性：不包含在主码中的各个属性。

6．外码：设 F 是关系 R 的一个属性，不是 R 的主码，但却是另一个关系 S 的主码，则称 F 是关系 R 的外码。

例：student (sno, sname, ssex, sage, sdept)

Sc (sno, cno, grade)

Sc 的主码为：（sno,cno）；外码为：sno

关系的数学定义

一、域(domain)

1．定义：域是一组具有相同类型的值的集合。

2．域的基数：域中所含数据的个数。

二、笛卡尔积

1．定义：给定一组域 D1,D2,D3，则 D1×D2×D3 称为笛卡尔积。

2．笛卡尔积 D1×D2×D3 对应一个二维表，所含元组的个数等于各个域的基数之积。

三、关系

1．定义：笛卡儿积的一部分元组称为关系。

2．关系的目（或度）：一个关系所含属性的个数。

3．关系的性质

任意两个元组不能完全相同，但属性名允许重复。

四、关系的完整性

1．实体完整性：指关系的所有主属性都不能取空值。

注意：实体完整性不仅仅是主码整体不能取空值。

2．参照完整性：指一个关系外码的取值必须是相关关系中主码的有效值或空值。

例：班级(班名,人数)

学生(学号,姓名,性别,密码,班名)

在学生表中，班名的取值必须是班级表[班名]的值或空值。

关系代数

一、传统的集合运算

设关系 R、S 的结构完全相同，则：

R∪S：由属于 R 或属于 S 的元组组成。

R∩S：由既属于 R 又属于 S 的元组组成。

R－S：由属于 R 而不属于 S 的元组组成。

思考：（R∩S）∪（R－S）=?

R×S：设 R 有 m 个属性，K1 个元组；S 有 n 个属性，K2 个元组，则 R×S 含有(m+n)个属性，(K1×K2)个元组。

二、专门的关系运算

1．选择：从关系 R 中选择满足条件的元组。记为：$_{F}(R)$

2．投影：从关系 R 中选择若干属性组成新的关系，并把新关系的重复元组去掉。

记为： $\prod_{A}(R)$

3．条件连接：将两关系按一定条件连接成一个新关系，记为： $R\underset{F}{\infty} S=\sigma_{F}(R\times S)$

说明：条件连接：两关系可以没有公共属性，若有公共属性，则新关系含有重复属性。

4．自然连接：将两关系按公共属性连接成一个新的关系，并把新关系的重复属性去掉。

记为： $R\infty S$

说明：① 自然连接：两关系至少有一个公共属性。

② 对于 R 的每个元组，S 都从第一个元组开始判断，若两元组的公共属性值相同，则产生一个新元组添加到新关系中，最后把新关系中的重复属性去掉。

5．除：给定关系 R（x，y）和 S（y，z），则 R÷S=P（x），其中 x，y，z 为属性组。

求解过程：

① 求 R 中 x 可以取哪些值，并求各值的象集。

② 求 S 在属性组 y 上的投影 K。

③ 检查每个象集是否包含 K

注：除不是一个必须的运算，可以由其它运算符代替。

例：设有关系 R，S 如下图，求 R÷S。

R

A	B	C
a1	b1	c2
a2	b3	c7
a3	b4	c6
a1	b2	c3
a4	b6	c6
a2	b2	c3
a1	b2	c1

S

B	C	D
b1	c2	d1
b2	c1	d1
b2	c3	d2

解：在关系 R 中，A 可以取四个值，a1，a2，a3，a4。

a1 的象集为{（b1，c2），（b2，c3），（b2，c1）}

a2 的象集为{（b3，c7），（b2，c3）}

a3 的象集为{（b4，c6）}

a4 的象集为{（b6，c6）}

S 在（B，C）上的投影 K 为{（b1，c2），（b2，c3），（b2，c1）}

显然只有 a1 的象集包含 K，故 R÷S={a1}

结论：如何写关系代数表达式？

答：① 查询涉及多个关系时，一般使用 $\infty\rightarrow\sigma\rightarrow\prod$ 。

② 查询涉及“否定”时，一般用差运算。

③ 查询涉及“全部”时，一般用除运算。

④ 查询涉及“至少”时，一般用×

关系数据库规范化理论

函数依赖

一、有关概念：

R 表

XH	KH	KM	XM	DZ	CJ

961	C1	OS	高明	D1	70
962	C2	DBS	高飞	D2	72
962	C4	AI	高飞	D2	80
962	C1	OS	高明	D1	75
963	C1	OS	高明	D1	90

1．函数依赖：

任给 R（U），U 为属性集，x、y 为 U 的子集，如果对于 x 的每个值，y 有唯一确定的值与之对应，则称 x 决定 y，或 y 函数依赖于 x。记为：x→y。

例：KH→KM

XM→DZ

(XH,KH) →CJ

KH→(KM,XM)

2．完全函数依赖：

若 x→y，且对于 x 的所有真子集 x′，都有 x′ y，则称 x 完全决定 y，或 y 完全函数依赖于 x。记为：$x \xrightarrow{f} y$。

例 1：(XH,KH) →CJ

但 XH→CJ

KH →CJ

故 $(XH,KH) \xrightarrow{f} CJ$

例 2：KH→KM

则 $KH \xrightarrow{f} KM$

结论：若 x→y，且 x 只包含一个属性，则 $x \xrightarrow{f} y$。

3．部分函数依赖：

若 x→y，且存在 x 的一个真子集 x′，满足 x′ →y，则称 x 部分决定 y，或 y 部分函数依赖于 x。记为：$x \xrightarrow{p} y$。

例 1：(KH,KM)→XM

但 KM→XM

则 $(KH,KM) \xrightarrow{p} XM$

例 2：(XH,KH)→DZ

但 KH→DZ

则 $(XH,KH) \xrightarrow{p} DZ$

4．传递函数依赖：

若 x→y, y→z, 但 y→ / x, 则 $x \xrightarrow{t} z$

例：KM→XM, XM→DZ, 但 XM→ / KM

二、平凡函数依赖与非平凡函数依赖

设 x→y，如果 y 是 x 的子集，则该依赖是平凡的。如：

Sno,sname→sno

如果 y 中至少有一个属性不在 x 中，则该依赖是非平凡的。如：

Sno, sname→sname, sdept

如果 y 中没有一个属性在 x 中，则该依赖为完全非平凡的。

三、函数依赖的推理规则

设有关系 R，x、y、z 为 R 的一个属性集，则有：

①自反律：若 $y \quad x$，则 x→y。

②增广律：若 x→y，则 xz→yz。

③传递律：若 x→y，y→z，则 x→z。

注意传递律与传递函数依赖的区别。

④合并律：若 x→y，x→z, 则 x→yz。

⑤分解律：若 x→yz，则 x→y，x→z。

关系模式的规范化

一、问题提出

R 表

XH	KH	KM	XM	DZ	CJ
961	C1	OS	高明	D1	70
962	C2	DBS	高飞	D2	72
962	C4	AI	高飞	D2	80
962	C1	OS	高明	D1	75
963	C1	OS	高明	D1	90

答：存在问题

① 数据冗余大；

② 修改麻烦；

③ 插入异常：应该插入到 DB 中的数据插不进去。

如：新开课程没有学生选修时，新开课程的课程号、课程名插不进去。

④ 删除异常：不应该删除的数据被删掉。

如选修某门课的学生毕业了，在删除学生信息的同时，把课程信息也删除掉。

结论：一个好的关系模式应满足：

① 冗余应尽可能少；

② 应尽可能避免插入、删除异常；

③ 消去关系中不合适的属性依赖关系。

二、范式

① 什么叫范式？

指一个关系的非主属性函数依赖于主码的程度。

② 什么叫关系规范化？

指一个关系从低级范式向高级范式的转换过程。

③ 应用：关系规范化理论应用在逻辑结构设计阶段。

三、关系模式的规范化

1．第一范式（1NF）

① 定义：若关系 R 的所有属性不能再分，则 R∈1NF

② 存在问题

③ 原因：存在非主属性对主码的部分依赖。

④ 解决办法：消除非主属性对主码的部分依赖，

将关系 R 一分为二，将满足完全依赖的属性集组成一个关系；将满足部分依赖的属性集组成另一个关系；

R1 表

XH	KH	CJ
961	C1	70
962	C2	72
962	C4	80
962	C1	75
963	C1	90

R2 表

KH	KM	XM	DZ
C1	OS	高明	D1
C2	DBS	高飞	D2
C4	AI	高飞	D2

R1 主码：（XH,KH）

R2 主码：KH

2．第二范式（2NF）

① 定义：若关系 R∈1NF，且它的每个非主属性都完全依赖于主码，则称 R∈2NF。

② 存在问题：

- 冗余大： R1 必要冗余，R2 冗余可以修改。
- 修改麻烦
- 插入异常：如新来的教师没有上课，则该教师的信息就没办法插入 R2 表中。
- 删除异常：若某位教师只授一门课，当该门课不开时，该教师的信息亦被删除。

③ 原因：存在非主属性对主码的传递依赖。

KH→XM，XM→DZ，但 ~~XM~~ KH

$\therefore KH \xrightarrow{t} DZ$

[传递依赖必须有两个非主属性]

④ 解决办法：将 R2 一分为二

R21 表

KH	KM	XM
C1	OS	高明
C2	DBS	高飞
C4	AI	高飞

R22 表

XM	DZ
高明	D1
高飞	D2

R21 主码：KH

R22 主码：XM

3．第三范式（3NF）

① 定义：若关系 R∈2NF，且它的每个非主属性都不传递依赖于主码，则称 R∈3NF。

② 规范化过程

非规范关系

↓使每个属性都不能再分

1NF

↓消去非主属性对主码的部分依赖

2NF

↓消去非主属性对主码的传递依赖

3NF

4．结论

① 若 R∈1NF，且主码只含一个属性，则 R 一定为 2NF。

② 若 R∈2NF，且只有 0~1 个非主属性，则 R 一定为 3NF。

③ 3NF 一般控制了数据冗余，一般避免了操作异常。

④ 范式并非越高越好，适可而止。

数据库设计

一、数据库设计的步骤

① 需求分析：了解分析用户的需要、要求。

② 概念结构设计：根据需求分析的结果画概念模型（即 E-R 图）。

③ 逻辑结构设计：将 E-R 图转换为某一种数据模型，并优化。

④ 物理结构设计

⑤ 数据库实施

⑥ 数据库运行与恢复

概念结构设计

一、局部 E-R 图设计

1．确定局部范围

通常把系统涉及的各个部门或各个主要功能作为局部。

2．确定实体与属性

① 属性是不能再分的数据项；

② 联系只发生在两实体之间；

③ 原则上，能够作为属性，就不要作为实体。

二、合并成总体 E-R 图

1．消除各局部 E-R 图的冲突问题。

2．按公共实体名合并，生成初步 E-R 图。

3．消除冗余的属性和冗余的联系，生成总体 E-R 图。

逻辑结构设计

一、联系的属性和主码

（1）联系的属性：必须包含相关联的各实体型的主码。

（2）联系的主码

1：1 联系：可以是相关联的任一实体型的主码。

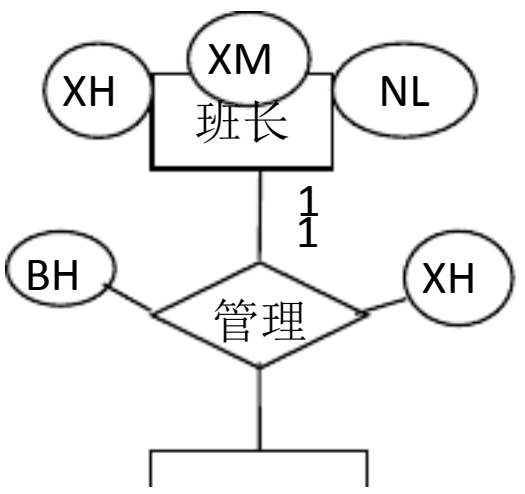
1：n 联系：必须是 n 方实体型的主码。

m：n 联系：必须是相关联的各实体型的主码之和。

二、E-R 图向关系模型的转换

（1）把每个实体型转换为一个关系模式。

（2）1：1 联系：可以消化到相关联的任一实体型对应的关系模式中。

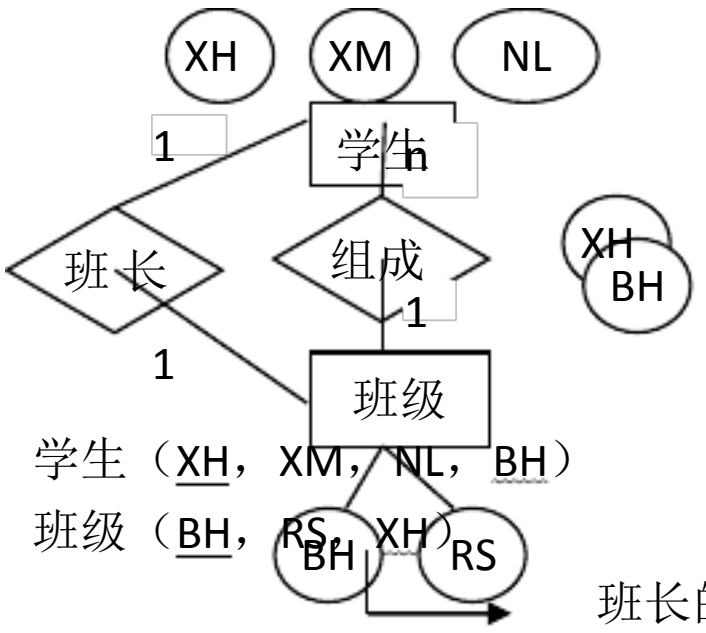


班长(XH， XM， NL，BH)

班级（BH，RS）

（3）1：n 联系：可以消化到 n 方实体名对应的关系模式中。

例：一个班级有多名学生，每名学生只能属于一个班级。每一个班级有一名班长，他是学生中的一员。



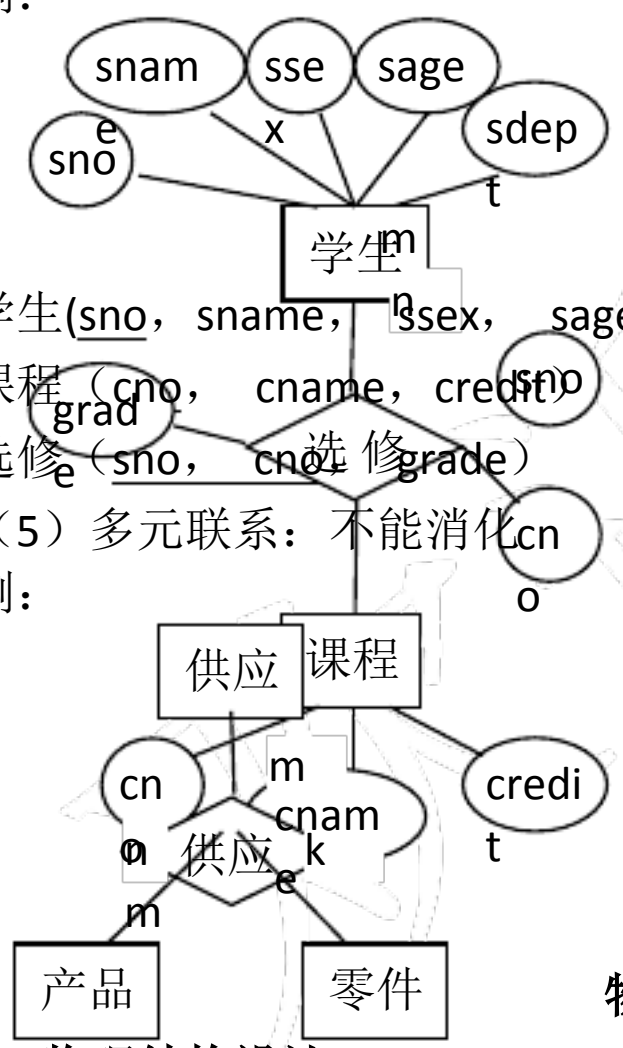
学生（XH，XM，NL，BH）

班级（BH，RS，XH）

班长的学号

（4）m：n 联系：必须转换为一个关系模式，并且不能消化。

例：



物理结构设计与数据库实施

1．物理结构设计

在逻辑设计的基础上，为每个关系模式选择合适的存储结构与存储方式。

选择存储结构：即决定每个表的记录顺序。

选择存取方式：即决定为哪些属性建立非聚集索引，以便加快查找速度。一般把经常查询的属性名指定为非聚集索引。

2．数据库实施

主要工作：

定义数据库结构；

组织数据入库；

编写应用程序；

数据库试运行；

SQL Server 2005 系统入门

一、运行环境

最低处理器速度：600MHZ（推荐 1GB 或者更快）

最小内存：512MB（推荐 1GB 或者更大）

可用硬盘空间：1.6GB

二、SQL Server 2005 的主要组件

1．服务：用于启动 SQL Server 2005 服务器

单击“开始”菜单，选择“控制面板”|“管理工具”|“服务”选项，将“SQL Server（MSSQLSERVER）”选项置为“自动”，即随操作系统的启动而自动启动。安装时默认为“自动”。

2．配置管理器（Configuration Manager）：用于启动 SQL Server 2005 服务器

注意：“服务”窗口中显示的是操作系统中全部的服务程序，而“配置管理器”窗口中显示的仅仅是与 SQL Server 2005 有关的服务程序。

3．SQL Server Management Studio（简称 SSMS，管理工作室）

组合了对象资源管理器、查询编辑器的功能。对象资源管理器用于管理数据库服务器中的所有数据库对象；

4．SSMS 查询编辑器：

主要用于输入、执行和保存 Transact-SQL 命令

创建和使用数据库

一、数据库文件

1．一个数据库至少有一个主要数据文件和一个事务日志文件。如果数据库很大，可以使用一个主要数据文件、多个次要数据文件和多个事务日志文件。

主数据文件(.mdf)
次数据文件(.ndf) } 用于存放数据库的各类

事务日志文件(.ldf)：用来记录对数据库对象的所有更新操作。

2．系统数据库

Master 数据库、Model 数据库

二、创建数据库

1．打开数据库：

Use 数据库名

2．删除数据库：

Drop database 数据库名

三、修改数据库

1．分离和附加数据库

2．备份和还原数据库

3．数据的导入和导出

数据库的备份和恢复

1．将数据库备份到一个备份文件中：

Backup database 数据库名 to disk=’路径\文件名’

2．将备份文件恢复到数据库中：

Restore database 数据库名 from disk=’路径\文件名’

创建和使用表

数据类型

1．字符型

char(n)： 定长字符型，n 表示字符数，取值为 1~8000。若缺省 n，则默认为 1。

varchar(n)：变长字符型

Text：可以存储任意长的字符串

说明：标准字符型：每个英文字母、数字算 1 个字符，每个汉字算 2 个字符，每个字符占 1 个字节。

2．统一码字符型

每个英文字母、数字、汉字算 1 个字符，每个字符占 2 个字节。

nchar(n)： n 表示字符数，取值为 1~4000。

nvarchar(n)

ntext：可以存储任意长的统一码字符串。

例：设某表的结构如下：

no char(6), name nchar(6)

则 no 可以赋予 3 个汉字，name 可以赋予 6 个汉字

3．整型

	取值范围	所占字节数
Bigint		8
Int		4
Smallint	-32768~ +32767	2
Tinyint	0~255	1
bit	0,1	即逻辑型

4．实型

① 精确数值型

Decimal(p,s)

Numeric(p,s)

精度 P：表示全部数字的位数（不计小数点和正负号）

S：表示小数位数，若缺省 S，则默认为 0

P-S：表示整数位数

② 近似数值型

Real 精确到 7 位有效数字

Float 精确到 15 位有效数字

5．贷币型

Money：占 8 个字节，保留 4 位小数

Smallmoney：占 4 个字节，保留 4 位小数

字符串常量：由单引号括住的字符序列

贷币型常量：可以是一个实型常量

6．日期时间型

Datetime、Smalldatetime

① 若省略日期部分，则默认为：’1900-1-1’
若省略时间部分，则默认为：’00:00:00’

② 日期格式：’yyyy-mm-dd’
时间格式：’hh:mm:ss’

使用对象资源管理器创建和管理表

一、创建表

1．在一个表中只能定义一个标识字段。

2．只有整型和精确数值型（decimal、numeric）字段才能指定为标识字段。

3．把某个字段指定为公式字段。

二、设置约束

1．主键约束：

每个表中只能有一列或一个组合被指定为主键，主键中的各个列不能为空值。

2．外键约束：

创建外键约束，就是定义两个表的永久关联，这两个表分别称为主键表、外键表。外键表中外键的值只能是主键表中主键的有效值或空值。

3．唯一性约束：

主键约束与唯一性约束的区别是：

（1） 在一个表中只能定义一个主键约束，但可定义多个唯一性约束；

（2）指定为主键约束的列不能取空值，但指定为唯一性约束的列允许取空值。

4．唯一性约束：

用于限制输入到一列或多列的值的范围，保证数据库的数据完整性。

5．默认值约束：

T-SQL

一、SQL 语言的特点

①SQL 语言集数据定义、数据查询、数据操纵、数据控制的功能于一体。

	动词
数据定义	Create、drop
数据查询	select
数据操纵	Insert、delete、update
数据控制	Grant、revoke

②所有的 DBMS 都支持 SQL 语言。

T-SQL 基础

一、创建和使用数据库

1．创建数据库

create database 数据库名

2．使用数据库

Use 数据库名

3．删除数据库

drop database 数据库名

二、 定义表

欢迎阅读

1．创建表

create table 表名（属性名 类型，…，属性名 类型）

①指定标识字段：identity(标识种子，标识增量)

②指定公式字段：属性名 as 表达式

例：create table student

(no int identity(1,1),

name char(6),

chi smallint,

mat smallint,

score as chi+mat)

2．删除表

drop table 表名,…，表名

三、select 语句

select */表达式表

[into 新表]

from 表名,…,表名

[where 条件]

[group by 属性名]

[having 条件]

[order by 属性名][Asc/Desc]

1．Select 子句

① *代表所有属性名

② 若一个属性名来自多个表，则属性名前须冠以表名，格式为：表名．属性名

③ 设置表达式的别名：

表达式 As 别名

④ 限制查询结果的记录行数：

all 返回全部记录

top n 返回前面 n 号记录

distinct 表示取消重复行

说明：top n 只能放在关键字 select 的后面；

all、distinct 只能放在关键字 select 或聚合函数的后面。

2．Where 子句

① in 的格式：属性名 in (常量，…，常量)

② like 的格式：属性名 like 通配字符串

通配符有：% 表示 0 个或多个字符

\- 表示 1 个字符

③ 在 Where 子句中指定连接：

Where 表名 1. 属性名=表名 2. 属性名

3．Order by 子句

order by 属性名 1 [Asc/Desc], 属性名 2 [Asc/Desc]

4．聚合函数

① sum(属性名)：纵向求数值型属性之和。

② avg(属性名)

③ count(*)　　返回表的记录行数（含重复行）。

count(属性名) 返回指定列中取非 NULL 值的单元格数目。

count(distinct 属性名) 返回指定列中取非 NULL 值、非重复的单元格数目。

④ max(属性名)

⑤ min(属性名)

5．Group by 子句

使用 Group by 子句时，Select 子句只能使用分组项字段和聚合函数

例：以性别为分组项，求每一组的平均年龄。

```
Select    ssex, avg(sage) as 平均年龄
From student
Group by ssex
```

6．Having 子句

① Having 子句只能跟在 Group by 子句之后，且只能使用聚合函数和分组项字段。

② where 子句放在 Group by 子句之前，甚至可以没有 Group by 子句；且不能包含聚合函数。

例：以系别为分组项，查询学生平均年龄大于 19 岁的系的系名，平均年龄。

```
Select    sdept,avg(sage) as 平均年龄
From student
Group by sdept
Having avg(sdept)>19
```

7．into 子句

功能：将查询结果保存到新的基表中。

一、　查询的分类

- 单表查询
- 连接查询
- 嵌套查询

1．连接查询：在 where 子句中指定连接

where 表名 1.属性名=表名 2.属性名

2．嵌套查询

① 嵌套查询的特点

- 每级查询的 from 子句一般只包含一个表名。
- 一个嵌套查询总可以分解为若干个单表查询，总可以改写成连接查询。
- 若查询结果显示的属性名来自一个表，才可以写成嵌套查询。
- 子查询不能使用 order by 子句，order by 只能用于最顶层的查询。

② 在 where 子句中指定子查询

where 属性名 [not] in（子查询）：子查询返回一列多行。

where 属性名=（子查询）：子查询返回一列一行。

where　[not] exists（子查询）：子查询返回多列多行。

五、数据操纵

1．insert 语句

（1）每次插入一条记录

insert into 表名[(属性名表)] values(表达式表)

（2）插入子查询的结果

insert into 表名[(属性名表)]

子查询

例：insert into student

select * from student1

2．update 语句

update 表名 set 属性名=值，…，属性名=值 [where 条件]

缺省 where 子句，默认为更新全部记录。

3．delete 语句

delete from 表名 [where 条件]

T-SQL 程序设计基础

一、常量：

字符型：由单引号括住，例：'china'

整型

实型

日期型：由单引号括住的具有日期或时间意义的序列，格式为：

'yyyy-mm-dd'， 'hh:mm:ss'

二、变量：

局部变量：由用户定义和赋值，以@ 开头。

全局变量：由系统定义和赋值，以@@ 开头。

1. 声明局部变量

Declare 变量名 类型

例：declare @a int,@b char(5)

注：不要把局部变量声明为 text、ntext、image

2. 给局部变量赋值

① 使用 set 语句：

Set 变量名=表达式

② 使用 select 语句

Select 变量名=表达式,……，变量名=表达式

[from 表名]

若表达式中含有属性名，则必须使用 from 子句。

例：declare @a char(5),@b char(6)

Select @a='95001' ,@b='王名'

或：select @a=sno,@b=sname

From student

3. 输出表达式的值：

①使用 print 语句

Print 表达式

②使用 select 语句

Select 表达式，……，表达式

[from 表名]

例：A）select @a,@b

B）select sno,sname

From student

C）select sno as 学号,sname as 姓名

From student

等价于：select 学号=sno, 姓名=sname

From student

4．局部变量的作用域：

只能在声明它的批处理中使用。

例：
```
Use stud
Go
Declare @a int
Set @a=5
Print @a
Go
Declare @a char(5)
Set @a='张三 '
Print @a
```

三、运算符：

算术运算符（*、/、% → +、-）

↓

关系运算符（>、>=、<、<=、=、<>）

↓

逻辑运算符（not → and → or）

批处理

1．什么叫批处理？

一个脚本由一个或多个批处理组成，批处理以 GO 作为结束标志。

2．批处理是脚本的编译单位，当一个批处理中的某个语句出现编译错误，则批处理中的任何语句均无法执行。

3．当一个批处理中的某个语句出现运行错误，则批处理中当前语句和它之后的语句将无法执行。

流程控制语句

一、begin…end 语句

Begin

语句 1

：

语句 n

End

二、if-else 语句

1.格式：if 逻辑表达式

语句 1

[else

语句 2]

2．当逻辑表达式包含子查询时，子查询必须用括号括住。

属性名 in (子查询)　　子查询返回一列多行

属性名 = (子查询)　　子查询返回一列一行

Exists(子查询)　　子查询返回多列多行

三、case 表达式

Case

When 逻辑表达式 1　then 结果 1

When 逻辑表达式 2　then 结果 2

……

[else 结果 n]

End

说明：case 表达式不是语句，不能单独执行。

例：use stud

Select 姓名=sname,系别=

Case

When sdept= 'CS' then '计算机科学系'

When sdept= 'IS' then '信息系统系'

When sdept= 'MA' then '数学系'

End

四、while 语句

1．格式：

While 逻辑表达式

Begin

语句组

End

2．专用于循环体的语句：

Break　　强制退出 while 语句，执行其后续语句。

Continue　返回 while 语句的入口。

Break、Continue 必须放在循环体内，并常与 if-else 语句结合使用。

二、其它语句

1．Return 语句

格式：Return (整数值)

功能：用于存储过程或批处理中，功能是退出所在的存储过程或批处理。

说明：当用于存储过程时，若没有为 Return 指定整数值，则默认为 0。

2．存储过程

① 创建存储过程

Create procedure 存储过程名 [@形参名 类型]

As　SQL 语句序列

② 执行存储过程

Exec　存储过程名 [常量| @ 变量名]:

③ 删除存储过程

Drop procedure 存储过程名

④ 说明：存储过程不返回值，或者只能返回整数值；而函数可返回任意类型的值。

3．Waitfor 语句

函数

1．创建函数

create function 函数名(@形参名 类型)　returns 类型

as

begin

函数体

End

注：① 函数体最后一条语句必须是 return 语句。

② 两类函数：标量函数、内嵌表值函数

2．执行函数

Exec　@变量名 1=函数名 常量| @ 变量名

Print 函数名 (常量| @ 变量名)

3．删除函数

Drop function 函数名

附加练习题 9

一、定义表：

1．SQL server 中建立一个数据库 stu。

2．使用企业管理器中创建如下三个表。

student 表

Sno	sname	ssex	sage	sdept
95001	李勇	男	20	CS
95002	刘晨	女	19	IS
95003	王名	女	18	MA
95004	张立	男	18	CS

Course 表

cno	cname	credit
1	数据库原理	4
2	操作系统	3
3	Java 程序设计	3
4	汇编语言	2

Sc 表

Sno	Cno	grade
95001	1	85
95001	2	88
95001	3	82
95002	1	90
95002	3	80

二、查询表：

1．查询全体学生的姓名及出生年份。

2．查询年龄小于 19 岁的男学生的学号，姓名。

3．查询选修了课程的学生人数。

[提示：select count(distinct sno) from sc]

4．查询所有选修过课的学生的学号、姓名。

5．查询数学系（MA）、计算机科学系（CS）的学生的姓名和性别。

6．查询年龄最大的 2 名学生情况

7．查询姓“刘”的学生情况。

8．以性别为分类项，查询每一类的平均年龄。

9．查询选修了课程名为“操作系统”的学生的学号、姓名。

10．查询年龄低于平均年龄的所有学生的姓名、系别

三、操纵表：

1． 在企业管理器中创建一个新表 student1：

Sno	sname	ssex	sage	sdept
95005	王虹	女	19	CS
95006	万亮	男	20	MA

2．将 student1 表插入到 student 表的后面。

[提示：insert into student select * from student1]

3．将 (95020，陈冬，男，18，IS) 插入到 student 表中。

4．删除 student1 表中的所有记录，使之成为空表。

5．把 student 表中所有男生记录复制到空表 student1 中。

6．把 student 表中所有女生记录复制到新表 student2 中。

7．将所有学生的年龄增加 1 岁。

8．将计算机科学系全体学生的成绩置 0。

[提示：update sc set grade=0

where sno in (select sno from student where sdept='CS')

9．删除学号为 95020 的学生记录。

10．删除计算机科学系所有学生的选课记录。

T-SQL 高级应用

一、查询的分类

- 单表查询
- 连接查询
- 嵌套查询

一、 连接查询：在 where 子句中指定连接

1．内连接：where 表名 1.属性名=表名 2.属性名

2．自身连接：给一个表取两个别名，where 子句的格式为：

where 别名 1.属性名=别名 2.属性名

3．左外连接：where 表名 1.属性名*=表名 2.属性名

意义：查询结果包含了表 1 的全部记录和表 2 满足条件的记录。

4．右外连接：where 表名 1.属性名=*表名 2.属性名

意义：查询结果包含了表 2 的全部记录和表 1 满足条件的记录。

三、连接查询：在 from 子句中指定连接

1．内连接：from 表名 1 inner join 表名 2 on 条件

例：select *

from student,sc

where student.sno=sc.sno

等价于 select *

from student inner join sc

on student.sno=sc.sno

2．左外连接：from 表名 1 left outer join 表名 2 on 条件

3．右外连接：from 表名 1 right outer join 表名 2 on 条件

4．完全外连接：from 表名 1 full [outer] join 表名 2 on 条件

5．交叉连接

四、嵌套查询

事务处理

1．什么叫事务？

事务是用户定义的一组操作序列。

① 事务是并发控制的基本单位。

② 一个事务包含的诸操作要么都执行，要么都不执行。

2． 事务的属性

- 原子性：指事务中包含的诸操作要么都执行，要么都不执行。
- 一致性：事务必须使数据库从一个一致性状态变到另一个一致性状态。
- 隔离性：一个事务的执行不能被其他事务干扰。
- 持久性

3．显式定义事务

begin transaction [事务名]

：

commit/rollback [transaction 事务名]

当未显式指定事务，则一个 SQL 语句就是一个事务。

4．在事务内设置保存点

begin transaction

save transaction 保存点名

rollback transaction 保存点名

功能：将保存点到 rollback 子句之间的 SQL 操作回滚掉。

数据的锁定

一、并发操作与数据不一致性

1．数据不一致性包括三类

丢失修改：指事务 1 与事务 2 从数据库中读入同一数据并修改，事务 2 的提交结果破坏事务 1 提交的结果，导致事务 1 的修改被丢失。

不可重复读：指事务 1 读取数据后，事务 2 执行更新操作，使事务 1 无法再现前一次读取结果。

读脏数据：指事务 1 修改某一数据后，事务 2 读取该数据，事务 1 由于某种原因被撤销，这时数据又恢复到原值，事务 2 读到的数据与数据库中的数据不一致，称为“脏”数据。

2．产生数据不一致性的原因

并发操作破坏了事务的隔离性。

二、并发控制的目标、方法

1．目标：确保 DB 中的数据一致性。

2．并发事务正确性的原则

几个事务的并发执行是正确的，当且仅当其结果与任何一个串行执行的结果相同。

3． 并发控制的方法

DBMS 一般采用“封锁”技术，保证并发操作的可串行化。

二、 封锁（Locking）

1． 什么叫封锁？

SQL Server 自动强制封锁，并且会将封锁粒度控制在合适的级别，用户不必考虑封锁问题。

2． 封锁类型

- 排它锁（X 锁）：事务 T 对数据 A 加 X 锁，其它事务不能再对 A 加锁，即其它事务不能读取和修改 A。
- 共享锁（S 锁）：事务 T 对数据 A 加 S 锁，其它事务只能再对 A 加 S 锁，即其它事务只能读 A，不能修改 A。

3． 封锁粒度

封锁对象可以是属性列、元组、关系、整个数据库。封锁对象的大小称为封锁粒度。

封锁粒度越小，并发度越高，但并发控制的开销越大。

4． 封锁协议

① 事务 T 在修改数据 A 之前，必须对其加 X 锁，直到事务结束才释放。

② 事务 T 在读取数据 A 之前，必须对其加 S 锁，直到事务结束才释放。

遵循封锁协议，可以解决三种数据不一致性问题：

- 丢失修改
- 不可重复读
- 读“脏”数据

四、死锁和活锁

封锁技术可以解决并发操作的不一致性问题，但也带来新的问题，即死锁和活锁。

1．死锁：

① 定义：两个事务已经各自锁定一个数据，但是又要访问被对方锁定的数据，造成了循环等待，称为死锁。

② 避免死锁的方法：

顺序封锁法：若规定封锁顺序为 A，B，则 T1，T2 只能先封锁 A，再封锁 B。

2．活锁：

① 定义：若多个事务请求封锁同一个数据时，其中的某个事务总处于等待状态，则称为活锁。

② 避免活锁的方法：先来先服务

使用游标

一、游标的概念

1． 每个表均有一个游标，它可以指向表的任意一条记录。

2． 移动游标的方法：

①在触发器或存储过程中，使用 SQL 语句定义和使用游标。

②在前台应用程序中，使用主语言实现对游标的移动。

二、 Transact-SQL 游标的使用：

1. 声明游标

Declare 游标名 cursor

[forward_only / scroll][global /local]

[read_only]

for select 语句

- forward_only(只进游标)：只能进行 next 操作，缺省为 forward_only。
- scroll （滚动游标）

[global /local]：缺省为：global

2．打开游标

open 游标名

打开游标时，游标指向查询结果集的第一条记录之前。

3．提取游标

fetch [next / prior / first /last] from 游标名

[into 局部变量名表]

① 缺省 into 子句：移动游标，并显示当前记录的内容。

② 含 into 子句：移动游标，并将当前记录各属性值依次赋给局部变量。

③ 缺省游标移动方式，则为 next。

4．关闭游标

close 游标名

5．释放游标

declaredeclare 游标名

例：--打印 student 表中全体男生的平均年龄。（ 不得使用聚合函数。）

```
set nocount on
declare @x int, @s int,@n decimal(4,1)
select @n=0, @s=0
declare pm cursor
scroll
for select sage    from student    where ssex='男'
open pm
fetch next from pm    into @x
while @@FETCH_STATUS=0
begin
select @n=@n+1, @s=@s+@x
fetch next from pm    into @x
end
print '全体男生的平均年龄为:'+str(@s/@n,4,1)
close pm
deallocate pm
```

三、 有关全局变量

1．@@FETCH_STATUS：返回上一个 Fetch 语句的执行状态，若 Fetch 语句成功执行，则返回 0，否则返回-1。

2．@@rowcount：返回受上一个语句（包括 select、insert、delete、update）所影响的行数。

3．set nocount on：关闭影响行数信息。

索引

一、索引的概念：

① 如果把数据表比作一本书，那么表的索引就是这本书的目录。可见，索引使用户能快速访问数据表的特定信息。

② 索引包括两项：索引字段值、原记录号

③ 索引必须依附于某个基本表，不能单独存在。

二、索引的类型：

聚集索引：影响数据表的记录顺序

非聚集索引：不会影响数据表的记录顺序

注：一个表只能建立一个聚集索引，但可以建立若干个非聚集索引。

三、创建索引

1．自动创建索引：

。如果在数据表的某个属性设置主键约束或唯一约束，则系统将在这些属性上自动创建唯一索引。

。自动创建的索引随约束的存在而存在，随约束的消失而消失。

2．使用 SQL 语句创建索引

Create [unique] [clustered/nonclustered] index 索引名

On 表名（属性名[asc/desc], 属性名[asc/desc]）

注：①若未指定 clustered，则创建非聚集索引；

②若未指定排序方式，则为 ASC；

③text,ntext 类型的字段不能指定为索引字段。

四、删除索引：

Drop index 索引名，…，索引名

思考题：创建主键时，如果使主键字段值不影响数据表的记录顺序？

视图

一、视图的特点：

①视图只有结构，没有记录，是虚表；

②一个视图总对应着一个 select 语句；

③对视图的查询、更新，实际上是对基本表的查询、更新。

二、定义视图：

1．创建视图：

Create view 视图名 [(属性名，…，属性名)]

As 子查询

[with check option]

说明：视图的属性个数必须与子查询中 select 子句的表达式个数相同。

2．删除视图：

Drop view 视图名，…，视图名

三、查询视图：

select */表达式表

from 视图名,…,视图名

[where 条件]

[group by 属性名]

[order by 属性名][Asc/Desc]

四、操纵视图：

1．向视图插入一条记录

insert into 视图名[(属性名表)] values(表达式表)

2．修改视图中的数据

update 视图名 set 属性名=值，…，属性名=值 [where 条件]

缺省 where 子句，默认为更新全部记录。

3．删除视图中的记录

delete from 视图名 [where 条件]

数据库完整性

一、在创建表时指定约束

1．主键约束

[constraint 约束名]

Primary key [Clustered/Nonclustered] [(属性名，…，属性名)]

说明：

① 每个约束都有一个约束名，约束名通常由系统自动给出。

② 列级约束：只牵涉到一个属性的约束，它放在相关属性的后面，且省略属性名表。

表级约束：牵涉到多个属性的约束。

③ 创建主键约束、唯一性约束时可以指定聚集（clustered）或非聚集(nonclustered)。

④ 主键约束默认为聚集的，唯一性约束默认为非聚集的。

⑤ 一个表最多只能创建一个约束是聚集的，聚集约束会影响数据表的记录号顺序。

2．外键约束

[constraint 约束名]

Foreign key[(属性名，…，属性名)]

References 主键表名(属性名，…，属性名)

例：创建如下两表

Student (sno, sname, ssex, sage, sdept)

Sc (sno, cno, grade)

```
Create table student
 (sno char(5) primary key,
    Sname char(6),
    Ssex char(2),
    Sage int,
    Sdept char(10))
    Create table sc
    ( sno char(5) foreign key references student(sno),
      Cno char(2),
      Grade int,
      Primary key(sno,cno)
    )
```

注意：两表关联的方式：

①临时关联：where 表名 1．属性名=表名 2．属性名

②永久关联：创建外键约束

3．唯一性约束

[constraint 约束名]

Unique [Clustered/Nonclustered] [(属性名，…，属性名)]

主键约束与唯一约束的区别：

① 在一个表中只能定义一个主键约束，但可定义多个唯一性约束；

② 指定为主键约束的字段不能取 null 值，但指定为唯一性约束的字段允许取 null 值。

4．检查约束

[constraint 约束名]

Check (条件表达式)

例： create table sc

(sno char(5),

Cno char(2),

Grade int,

Primary key(sno,cno))

5．缺省约束

[constraint 约束名]

Default 常量

二、删除表中的约束

alter table 表名

drop constraint 约束名,...,约束名

注意：alter 语句后面只能跟着一个子句。

三、向表添加约束

alter table 表名

add constraint 约束名 约束定义,...,

constraint 约束名 约束定义

约束定义指：

Primary key [Clustered/Nonclustered] (属性组)

Foreign key(属性组) references 主键表名(属性组)

Unique [Clustered/Nonclustered] (属性组)

Check(条件表达式)

Default 常量 for 属性名

例：现有 Student 表与 SC 表，请为它们创建如下约束。

Student 表：

列名	数据类型	长度	约束
Sno	char	5	主键
Sname	char	8	唯一约束
Ssex	char	2	只能输入“男”或“女”
sage	tinyint	1	只能为 15~20
sdept	char	2	

SC 表：

列名	数据类型	长度	说明
Sno	char	5	外码，参照 student 表
Cno	Char	2	
grade	tinyint	1	

```
alter table student
add constraint PK_student primary key(sno),
        constraint IX_student unique(sname),
        constraint CK_student check(ssex in ('男','女')),
        constraint CK_student_1 check(sage>=15 and sage<=20)
alter table sc
add constraint PK_sc primary key(sno,cno),
        constraint FK_sc_student foreign key(sno) references student(sno)
```

默认对象

①默认对象与默认约束的功能类似。

②默认对象以单独的对象创建，可以绑定到数据库的所有表中。

③默认约束只能绑定到一个表中。

规则

①规则与 check 约束的功能类似。

②规则以单独的对象创建，可以绑定到数据库的所有表中。

③check 约束只能绑定到一个表中。

存储过程

1．什么叫存储过程？

将一组 SQL 语句，以一个名称存储在数据库中，就形成存储过程。

2．创建存储过程

Create proc 存储过程名[@形参名 类型][=常量][output]

As SQL 语句序列

说明：

① [=常量]：用于指定形参的默认值；[output]用来指定该形参值是可以返回的。

3．执行存储过程

Exec 存储过程名 [常量| @ 变量名[output] | default]： 执行无返回值的存储过程

Exec @变量名 1=存储过程名 [常量| @ 变量名[output] | default] ：

执行有返回值的存储过程

② 实参可以是常量，已赋值的局部变量，带 output 的局部变量，default。

③ 当实参为带 output 的局部变量时，形参必须为带 output 的局部变量。

例：

```
create  proc proc1
@a varchar(20),@b varchar(20)='张三',@c varchar(40) output
as
set @c=@a+@b
```

```
go
declare @result varchar(20)
exec proc1 'I am ',default,@result output
print @result
```

4．删除存储过程

Drop proc 存储过程名[，存储过程名]

5．查看存储过程

exec sp_helptext 存储过程名

触发器

一、维护数据完整性的措施：

创建约束 } 基于一个表创建

创建触发器

创建规则：以单独的对象创建，可以绑定到数据库的所有表中。

二、触发器类型：

1．after 触发器：

① 当对表实施某种操作以后，就自动触发包含该操作的 after 触发器，并执行 as 后面的 SQL 语句。

② 一个表可以建立多个 after insert、after delete、after update 触发器。

③ after 触发器只能基于表创建。

2． Instead of 触发器：

① 当对表实施某种操作之前，就自动触发包含该操作的 Instead of 触发器，并执行 as 后面的 SQL 语句。

② 一个表或视图只能建立一个 instead of insert、instead of delete、instead of update 触发器。

③ instead of 触发器基于表或视图创建。

三、创建触发器：

Create trigger 触发器名

On 表名|视图名

For | after | Instead of 触发操作

As SQL 语句序列

说明：①for 或 after，表示创建 after 触发器。

②触发操作是指 Insert、update、delete 中的一个或多个。

四、触发器的应用

1．两个临时表：inserted、deleted

当触发器被创建时，将生成两个临时表；当触发器被触发时，将向临时表插入有关记录。

①当执行 insert 语句后，新记录插入到 inserted 表中。

②当执行 delete 语句后，被删记录插入到 deleted 表中。

③当执行 update 语句后，原记录插入到 deleted 表中，新记录插入到 inserted 表中。

2．在 student 表中创建一个 after 触发器，限制 ssex 只能输入男或女。若输入其它，将回滚插入的记录。

```
CREATE TRIGGER trig1
  ON student
after INSERT
AS
if exists (select * from inserted    where ssex not in('男','女'))
begin
print      ' 性别只能输入男或女'
rollback
end
```

3．在 student 表中创建一个 after 触发器，监控删除的学生记录。若被删的学生在 SC 表有选修课，则回滚被删记录。

```
CREATE TRIGGER trig2 ON student
after    DELETE
AS
if exists(select * from deleted,sc    where deleted.sno=sc.sno)
begin
print ' 该学生已选修过课,不能删除!'
rollback
end
```

4．在 student 表中创建一个 after 触发器，限制 sage 的变动只能在 1 岁以内。

```
CREATE TRIGGER trig3 on student
after UPDATE
AS
if exists( select * from inserted,deleted
            where inserted.sno=deleted.sno and abs(inserted.sage-deleted.sage)>1)
begin
print ' 年龄变动不能超过 1 岁!'
rollback
end
```

五、删除触发器

Drop trigger 触发器名[,触发器名]

安全管理

一、两种身份验证模式：

- 仅 windows 模式：用户只能使用 windows 登录名登录 SQL Server
- 混合模式：用户可以使用 windows 登录名或 SQL Server 登录名登录 SQL Server

二、两种身份验证：

用户登录到 SQL Server 时，必须使用特定的登录名和密码标识自己。

- Windows 身份验证：用户登录到 SQL Server 时，使用操作系统当前的登录名和密码。
- SQL Server 身份验证：用户登录到 SQL Server 时，必须显式提供登录名和密码。

三、登录名

欢迎阅读

1．系统内置的登录名：

- Sa　　系统管理员，具有最高的管理权限
- 域名\administrator：由 Sa 授权，权限一般与 Sa 相同

2．两类登录名

- windows 登录名
- SQL Server 登录名

四、sa 的作用：

①sa 拥有对全部数据库的所有操作权限。

②sa 可以创建普通登录名，并可以把普通登录名指定为一个或多个数据库的用户。

③把一个登录名指定为数据库的 public 和 db_owner，则登录名对该数据库拥有全部权限。