# SQL Server基础

## 安装

### 什么是实例

实例是在内存中运行的应用程序副本。

### 什么是服务

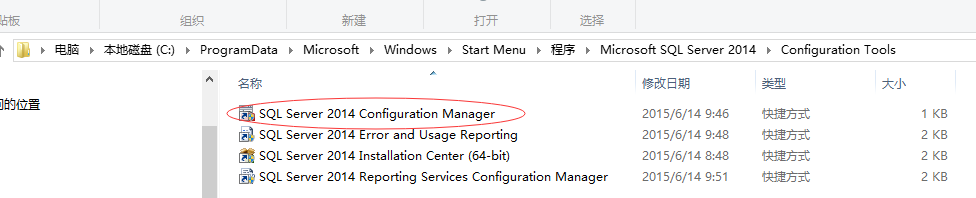
服务是在操作系统一开机就自动运行的帮助维护系统正常运行的一类程序。

### windows身份验证

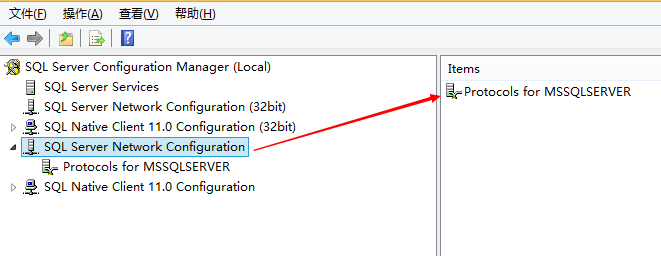
即登录windows系统时的账号

## 配置管理器

### 打开配置管理器

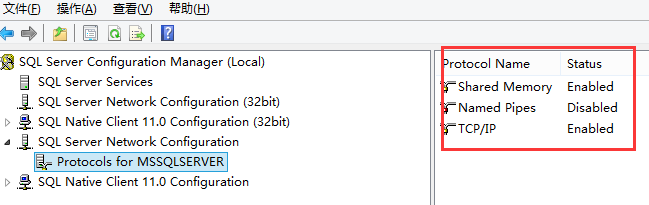


## 实例的协议



在这里对安装时创建的实例的协议（protocol）进行控制。

每一个实例有三种协议，这些协议代表了进程间通信的不同方式。



### Shared Memory

专门针对本机的连接。速度是最快的。

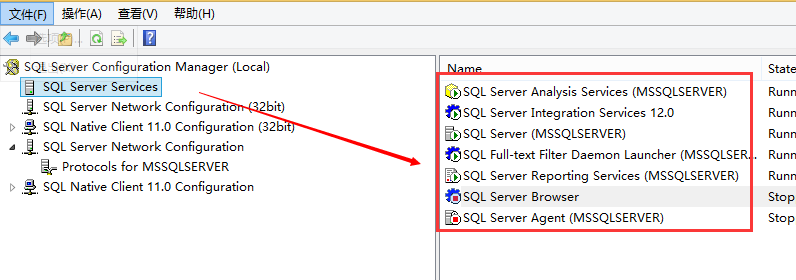
### Named Pipes

用在局域网中。

### TCP/IP

当数据库是服务在网络上的时候，需要TCP/IP协议支持。

## SQL Server 服务



### SQL Server Browser

用浏览器管理SQL Server，一般不用。

### SQL Server Agent

数据库代理，实质上就是委托。在生产中，为其配置自动执行任务的时间，就可以进行任务自动处理。比如数据库自动备份。

### SQL Server Reporting Services

数据库报表服务。数据统计结果输出。对数据进行统计、计算的服务。

### SQL Server Analysis Services

分析服务。

数据库的操作可以分为2类

* 联机事务处理（OLTP）
* 联机分析处理（OLAP）

SQL Server Analysis Services就是用来分析，预测的

### SQL Full-text Filter Daemon Launcher

全文检索。

### SQL Server Integration Services 12.0

数据库集成服务。

### SQl Server（实例名）

数据引擎，对数据的增删改查等操作，都是它提供的。大括号中的是实例名称。

## 连接数据库

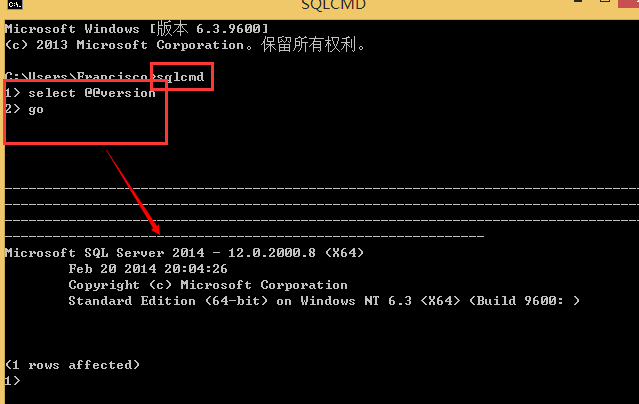
需要开启SQL Server服务，并根据连接网络类型，打开相应的协议。

### 控制台连接

在控制台中，输入sqlcmd，即可以进行SQL语句的输入了

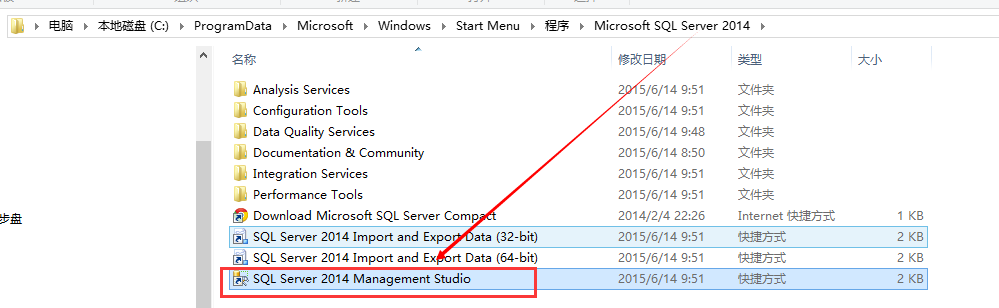


例如，输入：“select @@version;”,再输入：go执行命令，即可输出当前数据库的版本

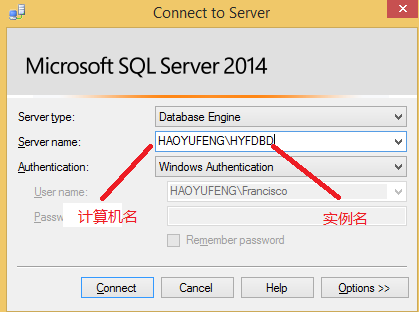


### 图形界面连接

1. 打开SQL Server 2014 Management Studio



1. 设置server name。 Server name： 计算机名\实例名如果是默认实例，只需要输入计算机名或者输入”.”、”localhost”、或者是ip地址（本机为127.0.0.1,如果登陆不上，可能是hosts文件问题）。



## 验证方式

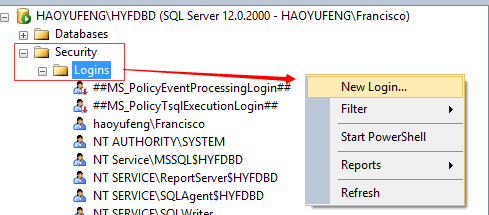
混合验证和windows验证。

登陆windows系统后就可以登陆。开发时用windows验证就行。

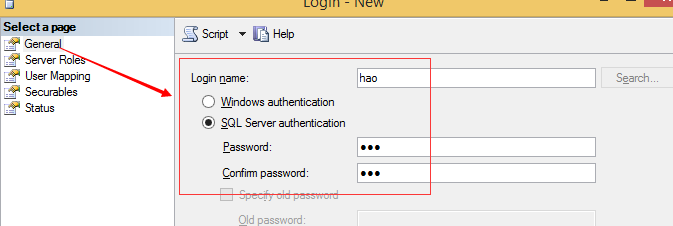
混合验证：连接网络时用的方式。输入登录名和密码即可登陆。

## 新建账号

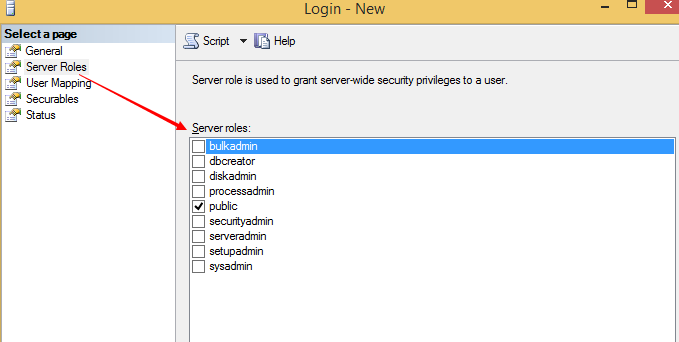
1. 在安全性→登陆上右键选择“新建登陆”



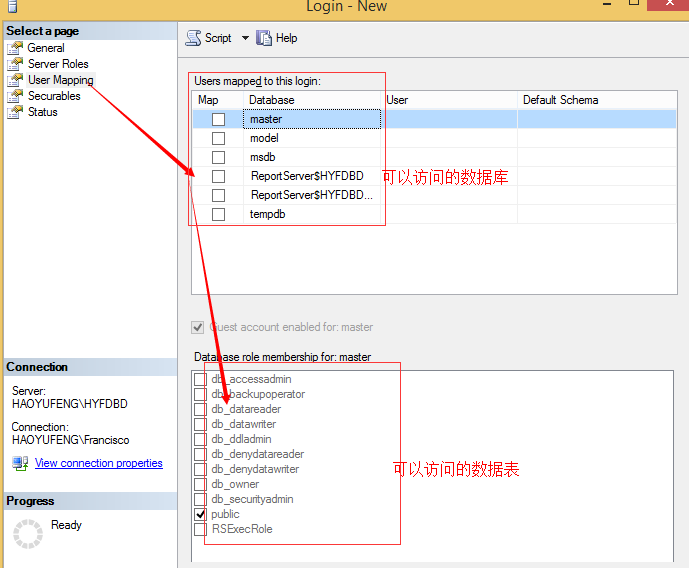
1. 新建账号密码



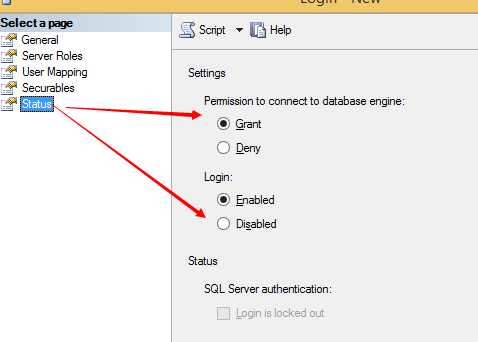
1. 设置权限



1. 数据库和数据表的访问权限



1. 是否可以登录和访问数据库引擎



1. 最后点ok确定即可。

# 数据库基础

DB：DataBase

数据库（数据库文件）：数据的仓库

数据库管理系统（DBMS，Database management system）

数据系统：用某种编程语言对数据库进行完整操作的完整的整体。例如对网站访问量就行统计的系统就是一个数据系统。

数据库管理员（DBA database administrator）:对数据进行管理（快速存储，检索，上架等，存储）的的人。

## 什么是数据库

数据库就是存储数据的仓库

随着电商的发展，仓库也越来越多的被大家所知道；

## 数据库技术的基本发展

结绳记事→文件存储→计算机人工管理（程序员自己编写，无法进行数据交换）--文件管理（数据冗余、数据关联）--关系型数据库（应用程序管理数据）--SQL标准。

对于数据不仅需要存储，更重要的是将数据进行存储以后怎么才能方便快捷的查询、修改、分析。

数据库特点：海量存储、查找速度快、并发性问题控制、安全性、数据完整性(存储的数据是正确的，真实的)。

高级应用：BI，专家系统，人工智能。

## 关系型数据库E-R

当前主流的数据库，都是基于E-R模型的数据库。

E-R模型指的是实体-关系模型。典型的特点是：一个实体有多个列（属性），一个行构成一个实体的实例；实体与实体之间可以有多种对应关系。

实体与类相似，只是这里只关注描述对象的数据，而不关心对象的行为（函数）。

## 数据库关系图

### 对应关系

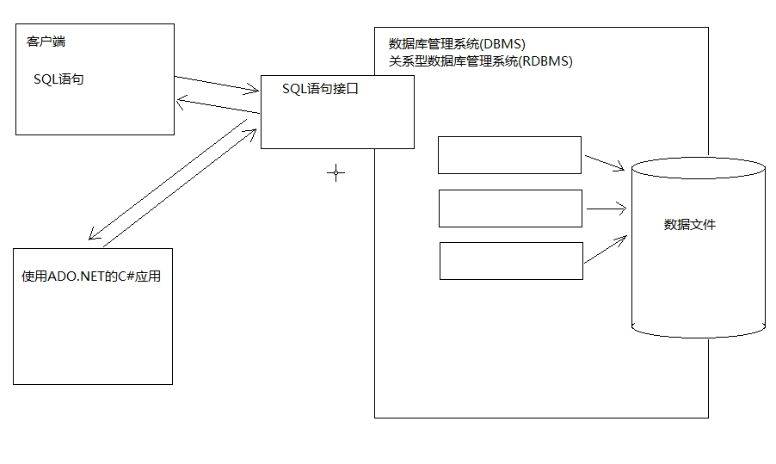
1对1的关系：这种数据存储到双方任一表里即可

1对多的关系：将关系的数据存储到多的一方的表里。

多对多的关系：单独建立一个表，用于存储关系。

根据关系创建出来的列的存储信息，需要满足第3NF。

## 数据库系统的基本模型

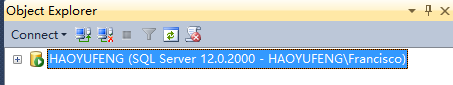


平常用的SQL Server management studio属于基本模型中的客户端。

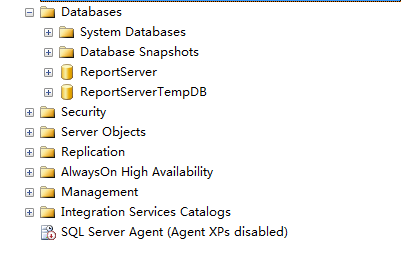
ADO.NET也是通过SQL语句接口通过数据库管理系统与数据文件进行沟通的。

## 对象资源管理器（Object Explorer）

对象资源管理器中的每一个根节点就是一个数据库的实例



其中的每一个节点都叫数据库对象



## 必会术语

Table（表）：关系数据库中的关系指的就是表。不同的货物要放到不同的货物架，将这种区域叫做“表”。不同的表根据放的数据不同进行空间的优化，找起来也方便。

列（column）：每一个字段就是一个列，也就是实体（E-R）的成员。

数据字典：描述数据的数据被称为数据字典（例如name=’hyf’，name就是描述’hyf’数据的数据，属于字典范畴）。

行（rows）：行中多个列的信息合起来确定一个实体。

## 数据库设计的范式（规范）

必须遵守3NF

1. 列的原子性，即列不可再拆分
2. 表中不能描述多个信息，不能有数据冗余



上表中，第一个表中描述了多个信息（产品与厂商），属于数据冗余。

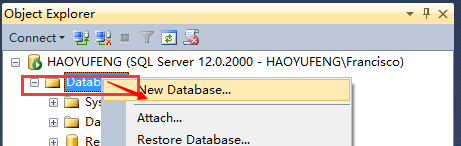
1. 引用其它表的主键信息

# 数据库基本操作

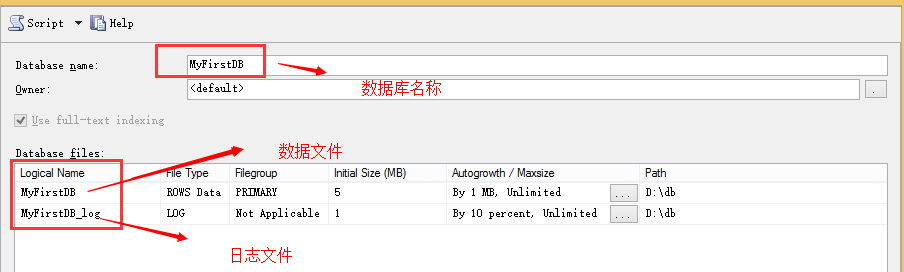
## 创建数据库文件

### 使用SSMS创建

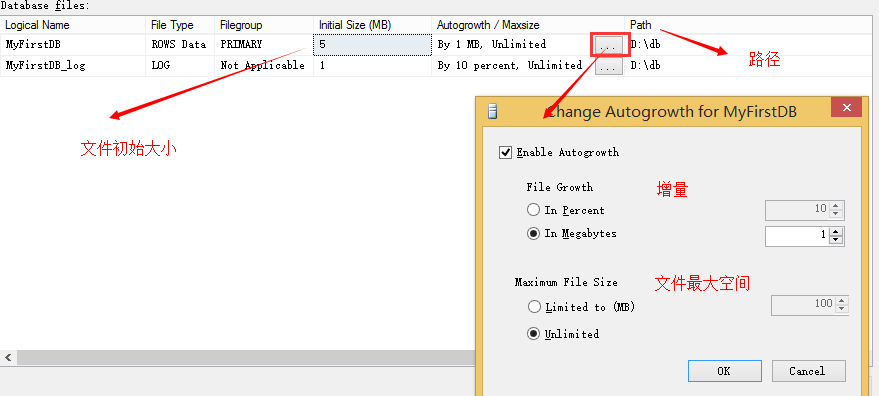
1. 找到数据库节点，右键新建



1. 输入数据库的名字（自动的填充两个文件，一个是数据文件（mdf），一个是日志文件（ldf））



1. 配置数据文件和日志文件的大小、最大限制、增量等信息

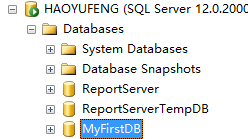


1. 确定

在设置的路径下有了两个文件



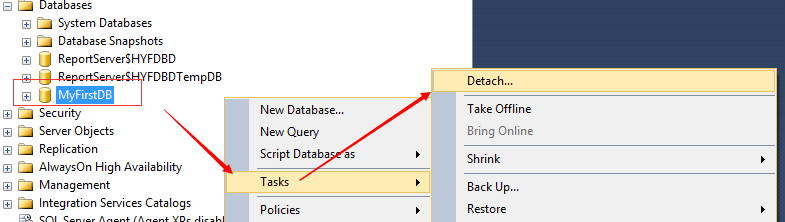
同时，在数据库节点下有了一个新的对象“MyFirstDB”



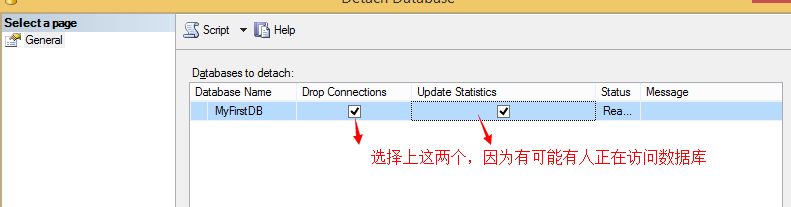
## 分离数据库

数据库进行拷贝时，需要先分离数据库。

1. 在想要拷贝的数据库上，右击选择任务→分离



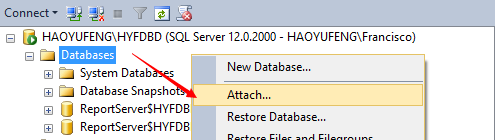
1. 关闭连接与更新统计信息



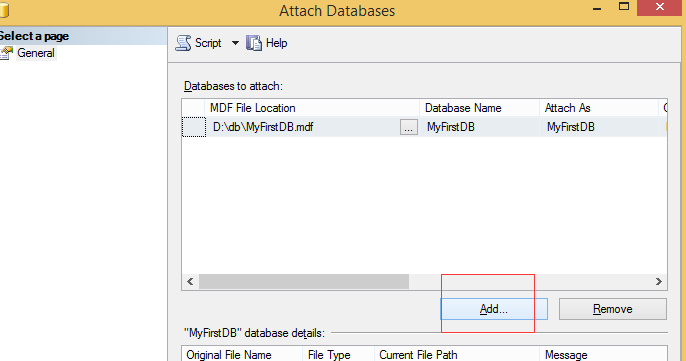
1. 点确定即可。

## 附加数据库

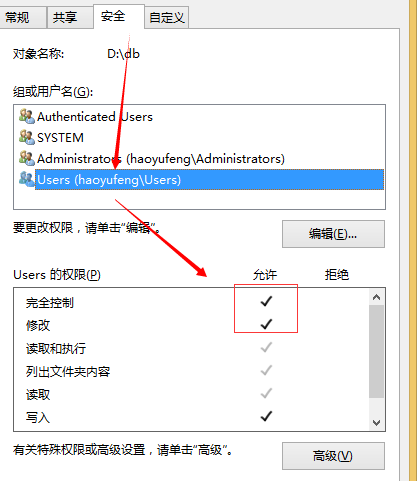
1. 在数据库上右键选择附加，



1. 添加数据库文件



1. 点确定即可把数据库文件添加进来。
2. 如果在附加数据库时没有成功，提示访问权限，可以在要附加的数据库所在文件夹上右击，选择属性，进行权限操作，如下：



## 创建数据库表

表是存储数据的地方

在数据库中存储的数据是描述某一个事物的特征（与类一样，描述事物特征的变量叫字段，数据库中列也叫字段）。

创建表，就是在设计表结构，就是说表应该有什么列（也叫字段），存储什么类型的数据。

## 数据类型

数字类型：int、bigint、money、numeric、bit

时间类型: datetime、date、time

字节类型: image、

字符串类型：

基于ASCII编码的：char 、varchar。每个字符站1个字节

基于Unicode编码：nchar、nvarchar。每个字符站2个字节

为了数据快速检索，字符串大小是规定好了的(如字符串大小不确定，数据库需要一个个字符进行扫描、才能确定什么时候结束)。

char[10]: 存储10个字节的字符串，如果提供的字符串长度不够10个字节后面自动填充空格。

varchar[10]: 存储10个字节的字符串，如果提供的字符串长度不够10个字节，有几个字节就存储几个字节。

字符串类型的使用注意：

char[数字]：这个数字可以取到8000。nchar[数字]：这个数字可以取到4000.

如果高出了这个范围，可以使用varchar[max]、nvarchar[max]，这个类型会在单独的地方存储数据，容量在2G左右。

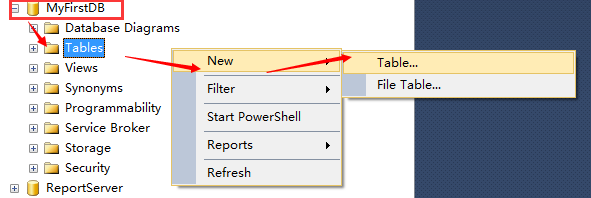
文本类型：存储内容大，则可使用text, ntext。不过已经过时了，推荐使用varchar[max]、nvarchar[max]。

## 创建表

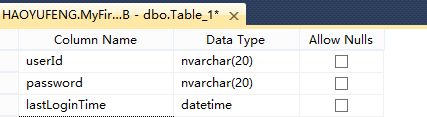
创建表和约束的语句形式非常多，一般熟练使用一种即可。

### 使用SSMS创建

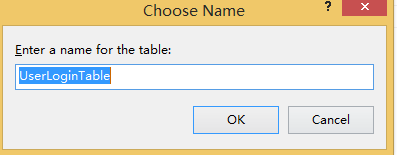
1. 在要创建表的数据库节点下找到表，右键选择新建，在弹出的菜单中选择表。



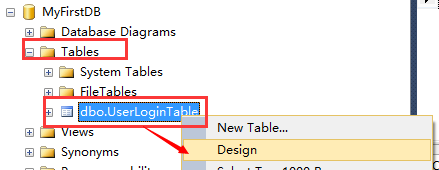
1. 在表中输入字段名及数据类型（定义列）



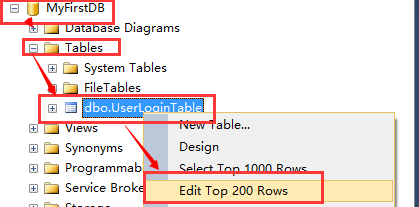
1. ctrl+S保存表



1. 如果想要修改表的字段，可以在表上右键选择设计



1. 如果要为表添加数据，则可以在表上右击，选择edit top 200 Rows



# 约束

数据库约束用为了保证数据的完整性（正确、真实）而实现的一套机制。

## 非空约束

作用：不允许为空。

## 主键约束（PK）

作用：数据行的唯一标识、不能重复的列才能当主键。没有特殊原因表都要设主键。

主键有两种选用策略：

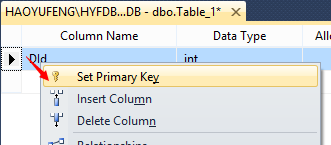
业务主键和逻辑主键

业务主键：使用有业务意义的字段做主键，比如身份证号、银行账户等。

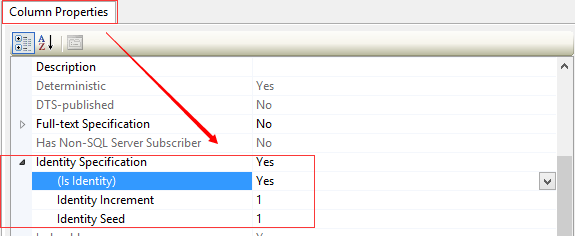
逻辑主键：使用没有任何业务意义（没有描述实体）的字段做主键，完全给程序看的，业务人员不会看的数据。因为很难保证业务主键不会重复（身份证号码重复）、不会变化（账号升位），因此推荐逻辑主键。

### 添加主键约束

在要设置主键的字段上右击，选择设置主键即可：



在列属性中可以设置标识规范来使主键自动填充值。



下面的是SQL语句设置。

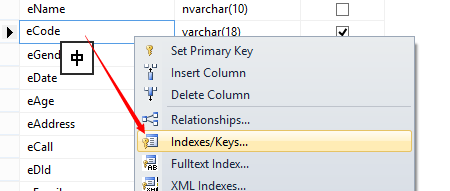
|  |
| --- |
| --增加主键约束  alter table TStudent  add constraint PK\_TStudent\_stuID primary key(stuId); |

## 唯一约束（UQ）

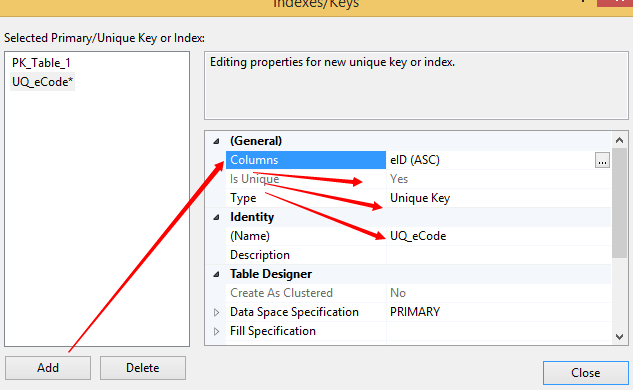
Unique constraint：唯一，允许为空，但只能出现一次。无法做主键的不能重复的字段，则可设置为唯一约束。典型是身份证号、编号。

### 添加唯一约束

1. 在要添加唯一约束的字段上右击，选择索引/键



2. 点击添加添加键，修改类型为Unique key，唯一性选择yes，并命名。



## 默认约束（DF）

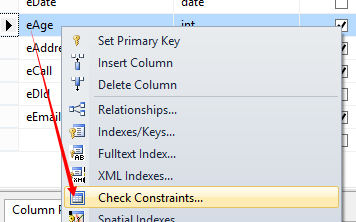
Default constraint：默认填充的值。

## 检查约束（CK）

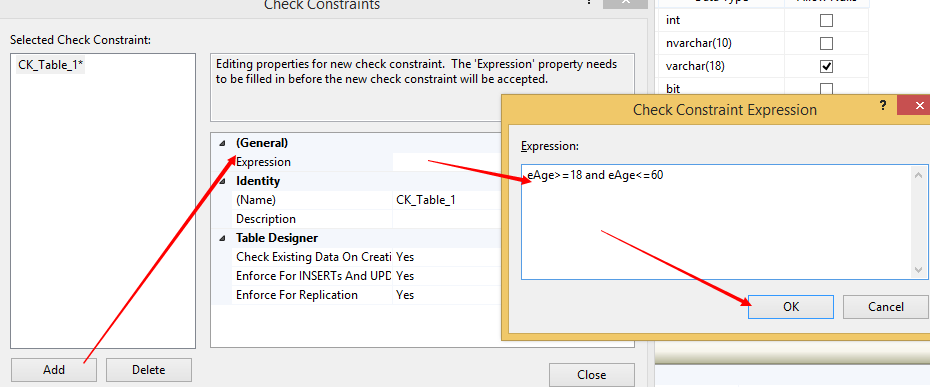
Checked constraint：范围以及格式限制。

### 添加检查约束

1. 在要添加检查约束的字段上右击，选择检查约束



1. 选择添加，在表达式中输入表达式



## 外键约束（FK）

Foreign constraint：外键约束。表现表之间的关系。

外键其实就是引用，因为主键实现了实体的完整性，外键实现了引用的完整性，

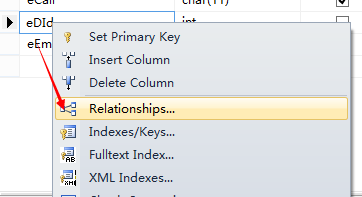
外键只是一个约束，保证数据的有效性。只要表中有关系列即可表与表的关系。

外键需要建立在关系列（存储了2个表的关系）上。

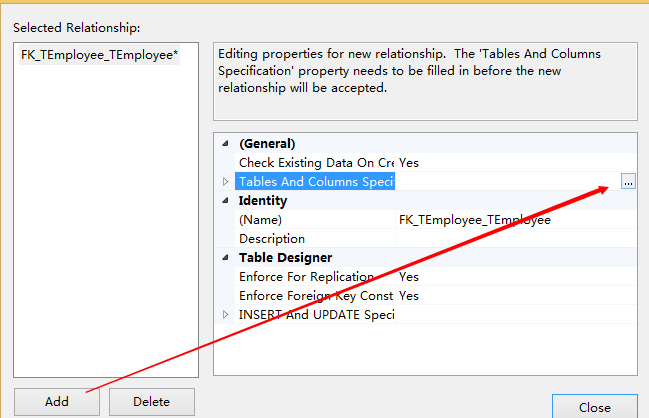
（国内工作中基本不做外键。增加外键约束时，设置级联更新、级联删除）

### 创建外键约束

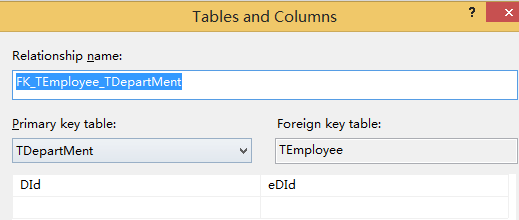
1. 在关系列上右击，选择关系



1. 选择增加之后在表和列的范式右侧的按钮上点击



1. 关系列所在的表作为外键表，从中选择要对应的字段，主键表为发生关系的实体所在的表，之后从中选择要对应的字段。在下图中，外键表Temployee的字段eDId和主键表TDpartment的字段Did之间是表面对应的。



# SQL语句入门

Sql全名是结构化查询语言（structured Query language），是关系数据库管理系统的 标准语言。

不同的DBMS都认识SQL语句。

SQL不区分大小写，字符串用单引号，在数据库中两个’’表示一个’（转义）。

SQL主要分DDL(数据库定义语言：建表，建库等语句)，DML（数据库操作语言：增删改查）和DCL（数据库控制语言：授权、取消权限）。

## DDL

有三个语句：create、alter、drop

用来创建：

* 数据库database
* 表table
* 视图view
* 存储过程proc
* 触发器trigger
* 索引index

## 数据库

### 创建数据库

|  |
| --- |
| --创建数据库  create database SchoolDB  --创建数据库文件  on primary  (  name=N'SchoolDB', filename=N'D:\db\SchoolDB.mdf', size=5120kb,maxsize=unlimited,filegrowth =1024kb  )  --创建日志文件  log on  (  name=N'SchoolDB\_log',filename=N'D:\db\SchoolDB\_log.ldf', size=2048kb,maxsize=2048gb,filegrowth =10%  ) |

### 删除数据库

|  |
| --- |
| --删除数据库  drop database SchoolDB |

### 选择数据库

|  |
| --- |
| --选择数据库，创建数据表前，需要先选择要添加数据表的数据库  use SchoolDB |

## 数据表

### 创建数据表

|  |
| --- |
| create table TblClass  (  cId int identity(1,1) primary key not null,--创建一个主键约束，唯一标识  cName nvarchar(10) not null,--创建一个非空约束的字符串  cDescription nvarchar(100) default(N'这是一个充满活力的班级')--创建一个默认约束的字符串  ) |

### 删除表

|  |
| --- |
| --删除表  drop table TblClass |

### 修改表

也就是为其添加约束，参考下一节。

### 清空表

Truncate清空表数据，将所有数据删除的执行效率比delete高，删除后，再插入数据时，标识会恢复种子值。

|  |
| --- |
| --清空表  truncate table tblClass |

## 列

### 修改列类型

|  |
| --- |
| --修改列类型  alter table TblClass  alter column cDescription nvarchar(150) |

### 添加列

|  |
| --- |
| --添加列  alter table TblClass  add snum int |

### 删除列

|  |
| --- |
| --删除列  alter table TblClass  drop column snum |

## 约束

约束的语法：

|  |
| --- |
| alter table 表格名称 add constraint 约束名称 增加的约束类型 （列名） |

### 检查约束

|  |
| --- |
| --添加检查约束  alter table TblClass  add constraint CK\_TblClass\_cName check (len(cName)>2); |

### 主键约束

|  |
| --- |
| --主键字段必须为非空  alter table tbltime add constraint PK\_Tbltime\_tId primary key(tId) |

### 唯一性约束

|  |
| --- |
| alter table tbltime add constraint U\_Tbltime\_tName unique(tname) |

### 外键约束

语法：

|  |
| --- |
| alter table 表名 add constraint 约束名称 约束类型 (列名) references 被引用的表名称（列名） |

|  |
| --- |
| alter table orders  add constraint FK\_Orders\_CustomerId --添加约束 名称  foreign key (customerId) references customer(customerId) --外键约束，外键列名，被引用列名 |

创建表时创建约束

|  |
| --- |
| foreign key(CustomerId) references CustomerInfo(CustomerId) |

#### 级联动作

外键和其他类型键的一个重要区别是：外键是双向的，即不仅是限制子表的值必须存在于父表中，还在每次对父表操作后检查子行(这样避免了孤行)。SQLServer的默认行为是在子行存在时“限制”父行被删除。然而，有时会自动删除任何依赖的记录，而不是防止删除被引用的记录。同样在更新记录时，可能希望依赖的记录自动引用刚刚更新的记录。比较少见的情况是，你可能希望将引用行改变为某个已知的状态。为此，可以选择将依赖行的值设置为NULL或者那个列的默认值。

　　这种进行自动删除和自动更新的过程称为级联。这种过程，特别是删除过程，可以经过几层的以来关系(一条记录依赖于另一条记录，而这另一条记录又依赖其他记录)。在SQLServer中实现级联动作需要做的就是修改外键语法-只需要在添加前面加上ON子句。例如：

|  |
| --- |
| alter table orders  add constraint FK\_Orders\_CustomerId --添加约束 名称  foreign key (customerId) references customer(customerId) --外键约束，外键列名，被引用列名  on update no action --默认　　修改时不级联更新子表  on delete cascade --删除时级联删除依赖行 |

　当在进行级联删除时，如果一个表级联了另一个表，而另一个表又级联了其他表，这种级联会一直下去，不受限制，这其实是级联的一个危险之处，很容易一个不小心删掉大量数据。

　　级联动作除了no action，cascade之外，还有set null和set default。后两个是在SQLServer2005中引入的，如果要兼容到SQLServer2000的话，要避免使用这两个级联动作。但是他们的才做是非常简单的：如果执行更新而改变了一个父行的值，那么子行的值将被设置为NULL，或者设置为该列的默认值(不管SET NULL还是SET DEFAULT)。

### 默认约束

语法：

|  |
| --- |
| alter table 表名称 add constraint 约束名称 约束类型 默认值） for 列名 |

|  |
| --- |
| alter table tbltime add constraint DF\_Tbltime\_tName default getdate() for tDate |

### 非空约束

|  |
| --- |
| alter table tbltime  alter column tDate date not null |

### 删除约束

|  |
| --- |
| alter table UserInfo  drop constraint DF\_\_UserInfo\_\_UserGe\_\_0F975522 |

## 数据的增删改查

### 增加数据

语法：

|  |
| --- |
| insert into 表名 (列1,列2,列3…) values(值1,值2,值3…); |

例如：

|  |
| --- |
| insert into UserLoginTable(userId,[password],lastLoginTime) values ('haoyufeng','hyf123','2015-6-22 12:12:12.000'); |

在SQL中换行被忽略，字符串用单引号括起来。

下面的语句是非标准的插入语句

|  |
| --- |
| --插入部分列。要求所有非空列都列出来，允许为空、有默认值的列不需列出  insert into TblClass(cName) values('Java');  --插入所有列: 省略列名部分，要求除了标识列都要插入数据，而且要按照定义的列的顺序插入  insert into TblClass values('.net','这是一个充满创意的班级');  --强制为标识列插入数据W  set identity\_insert Tblclass on  insert into tblclass(cid,cname) values(5,'ios');  --关闭手动插入标识列  set identity\_insert tblclass off  --插入单引号。需要转义（''）  insert into TblClass(cName) values('i''am')  --强制插入中文  insert into tblclass(cName) values(N'表示我是中文编码'); |

### 查看数据

语法：

|  |
| --- |
| select 列1,列2,列3… from 数据表; |

例如：

|  |
| --- |
| select userId,[password],lastLoginTime from [dbo].[UserLoginTable]; |

查看所有数据

|  |
| --- |
| select \* from UserLoginTable; |

#### 对列的限制(as)

|  |
| --- |
| --取所有列  select \* from TblClass  --取指定列  select tCid,tCName from TblClass  --为列起别名  select tCid as 编号, tCName as 名称 from TblClass  --为列起别名  select 编号=tCid, 名称=tCName from TblClass |

### 删除数据

删除数据时一般不做物理删除（一一旦删除，不可恢复），建议使用逻辑删除，即为表加一个表示“是否删除”的bit类型的列。

语法：

|  |
| --- |
| delete from 数据表 where 条件 |

例如：

|  |
| --- |
| delete from UserLoginTable where userId='haoyufeng'; |

### 修改数据

语法：

|  |
| --- |
| update 表名 set 列名1=值1, 列名2=值2 …where 条件 |

例如：

|  |
| --- |
| --更新所有行  update TblClass set cDescription='hello';  --更新特定行  update TblClass set cDescription='已经毕业，就业还不错' where cId=3; |

# 数据库高级检索

## 筛选

### 对行的限制(top,distinct)

关键字：

|  |
| --- |
| --关键字top：表示取前n个数据--------------------------------  select top 2 \* from TblClass --取前2条数据  select top 2 percent \* from TblClass --取前2%条数据 |
| --关键字distinct：去掉重复项，是对整个结果集进行重复处理，而不是针对某一个列  select distinct tCid,tCDescription from TblClass |

子句：

|  |
| --- |
| --where子句  select \* from TblTeacher where tTid>5 --比较运算符：> < >= <= != <>  select \* from TblTeacher where tTid>5 and tTid<8 --逻辑运算符： and or not  select \* from TblTeacher where tTid>5 and tTid<8 or LEN(tTname)>2;--运算符优先级。比较优先于逻辑  select \* from TblTeacher where tTid between 5 and 8 -- 一个连续的范围空间 [5,8] |

### where…in（不连续空间）

取一个不连续的范围。

|  |
| --- |
| --检索cId为1，4和6的行  select \* from TblClass where cId in (1,4,6) |

### where …like（模糊查询）

\_:表示任意一个字符

%:任意多个字符

[]: 一个连续的区间

^:放在[]中表示非

|  |
| --- |
| --查询cDescription中包含毕业两个字符的信息  select \* from TblClass where cDescription like '%毕业%'  --查询cDescription中以h开头并且是2个字符的信息  select \* from TblClass where cDescription like 'h\_'  --查询cDescription 中包含%号的信息。使用转义：[%]包起来  select \* from TblClass where cDescription like '%[%]%'  --查询cDescription 中包含4-8之间的信息  select \* from TblClass where cDescription like '%[4-8]'  --查询cDescription 中不包含4-8之间的信息  select \* from TblClass where cDescription like '%[^4-8]' |

### where … is null(空值判断)

|  |
| --- |
| --判断为空  select \* from TblClass where cDescription is null  --判断不为空  select \* from TblClass where cDescription is not null |

## order by（排序）

|  |
| --- |
| --升序排序，asc可以省略，默认就是升序排列  select \* from TblClass order by cName asc  --降序排列  select \* from TblClass order by cName desc  --按照多列排序：默认按照cName降序，cName相同，则按照cId升序  select \* from TblClass order by cName desc , cId |

## 聚合函数

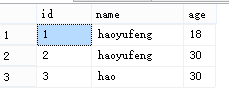
聚合就是把多行合成一行。有五个常用的聚合函数

|  |
| --- |
| --找出sEnglish字段中的最高值  select MAX(sEnglish) from TblSorce  --找出sEnglish字段中的最小值  select MIN(sEnglish) from TblSorce  --找出sEnglish字段中的平均值  select AVG(sEnglish) from TblSorce  --找出sEnglish字段中的总和  select SUM(sEnglish) from TblSorce  --求sEnglish字段的数量,不会包含null的数据  select COUNT(sEnglish) from TblSorce |

## group by（分组）

对前一个数据筛选的结果进行重组，使用某个字段或多个字段进行分类，然后按照这些字段组合数据。需要注意的是，好像是按照字段重新得到了新表。

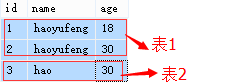
例如，在下面的表有三个数据



按照名字进行分组，并检索

|  |
| --- |
| select name from PersonTable group by name; |

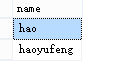
在上面的语句中，PersonTable group by name按照名字把表进行了分组，每一个不相同的名字形成一个新的小表，因此上面的数据分为了2个表:



在检索时，每一个小表只能显示为一行，因此，只能显示group by中包含的字段，如果显示age，一个为18，一个为30，就不知道显示那个了，

|  |
| --- |
| select name from PersonTable group by name; |

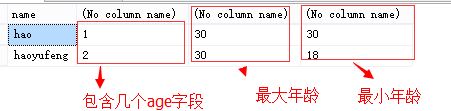
上面语句的检索结果：



想要显示age等信息就需要用聚合函数了

|  |
| --- |
| select name, count(age),MAX(age),MIN(age) from PersonTable group by name; |

显示结果如下：



### 案例：学士表，计算学生出现的次数

|  |
| --- |
| --创建学生表  create table TStudent  (  stuId int identity(1,1) not null,  stuName nvarchar(10) not null,  stuGender char(1) null,  stuAge int null  );  --增加主键约束  alter table TStudent  add constraint PK\_TStudent\_stuID primary key(stuId);  --增加学生数据  insert into TStudent(stuName,stuGender,stuAge) values('张三','M',18);  insert into TStudent(stuName,stuGender,stuAge) values('张三','F',19);  insert into TStudent(stuName,stuGender,stuAge) values('李四','M',18);  insert into TStudent(stuName,stuGender,stuAge) values('王五','M',18);  insert into TStudent(stuName,stuGender,stuAge) values('赵六','F',18);  --count统计学生出现的次数  select TStudent.stuName ,count(\*) from TStudent group by stuName; |

### 案例： 统计产品销量

|  |
| --- |
| --创建订单表  create table TOrder  (  orderId int identity(1,1) not null, --订单ID  productId int not null, --产品ID  custId int not null, --用户ID  unitPrice money not null, --单价  quantity int not null, --数量  discount numeric(4,3) not null --折扣  );  --创建主约束  alter table TOrder  add constraint PK\_TOrder\_orderID primary key(orderID);  --添加数据  insert into TOrder(productId,custId,unitPrice,quantity,discount) values  (1,1,19.5,11,0.75),  (2,1,20,11,0.75),  (3,2,50,100,0.85),  (2,1,45.4,3,0.75),  (1,2,41.3,12,1),  (1,3,11,10,0.65)  ;  --查询各个商品售出多少  select productID, sum(quantity)as total from TOrder group by productId;  --查询卖出总价大于300元的商品  select productId, sum(unitPrice\*quantity\*discount) from TOrder group by productId having sum(unitPrice\*quantity\*discount)>300  --统计各个用户销售金额  select custId, sum(quantity\*unitPrice\*discount)as total from TOrder group by custId; |

### 分组与聚合函数

当用集合函数时，想要同时显示其它的列时，只有出现在分组中的列，才能出现在查询结果。

|  |
| --- |
| select custId, sum(quantity\*unitPrice\*discount)as total from TOrder group by custId; |

如上表:只有分组中存在的列custId，才能与集合函数一起出现在查询结果中。

## 分组中的筛选（having）

在分组后，对结果集进行筛选。

|  |
| --- |
| --按照sEnglish进行分组，从结果集中找到 sEnglish>60的组  select sEnglish from TblSorce group by sEnglish having sEnglish>60 |

## 完整的select语句 及执行顺序

|  |
| --- |
| select distinct top n percent 列名 from 表名 where 条件 group by … having … order by … |

执行顺序：

* from 表： 确定去哪个表中寻找数据。
* where 条件：在表中检索出哪些行。
* group by 列： 在where中拿出的数据基础上分组。
* having 筛选条件：对分组中的数据进行筛选。
* select 列：
* distinct ：上面的列
* order by 列：
* top：top最后计算。

综合语句练习

|  |
| --- |
| select distinct top 1 percent sEnglish as 英语成绩  from TblSorce  where ScoreId < 5  group by sEnglish  having sEnglish>=60  order by 英语成绩 desc |

分析执行顺序：

1. from TblSorce
2. where ScoreId < 5
3. group by sEnglish
4. having sEnglish>=60
5. select distinct sEnglish as 英语成绩（分成2步）
6. order by 英语成绩 desc
7. top 1 percent

## 聚合查询（常用来做汇总）

将多个结果集合并成一个结果集

前提：列数相同，并且对应的类型（至少能隐式转换）相同。

判断标准：以第一个结果集为准

结果: 结果集的列名以第一个为准，自动按照主键排序。去掉重复，相当于默认应用了distinct（union all 保留所有结果，不去掉重复，也不排序 ）

|  |
| --- |
| -- 联合需要select子句的列数相同、类型相同  SELECT Name, Age FROM dbo.T\_Person1  UNION  SELECT Name, Age FROM dbo.T\_Person  -- 显示名字与年龄，最后显示总年龄  SELECT Name ,age FROM dbo.T\_Person1  UNION  SELECT '年龄',SUM(Age) FROM dbo.T\_Person1 |

### 案例：查询每位老师的信息，包括姓名、工资，并且在最后一行加上平均工资和最高工资。

|  |
| --- |
| --查询每位老师的信息，包括姓名、工资，并且在最后一行加上平均工资和最高工资  select tTName,tTSalary from TblTeacher  union all  select '平均工资',AVG(tTSalary) from TblTeacher  union all  select '最高工资',MAX(tTSalary) from TblTeacher |

### 案例：添加多行数据

|  |
| --- |
| insert into T\_UserInfo(Uname,UPassword)  select 'Uname','UPassword'  union  select 'n','233' |

## 连接查询

当结果中的数据分布于多个表中时，需要将多个表连接到一起进行查询，这种查询就是连接查询。

### 内连接（inner join … on）

匹配两个表中都有的数据，把两个表中的字段显示到一个结果集中。

|  |
| --- |
| --查询：显示学生姓名及对应的班级名称  select tClassName,tSName  from TblClass  inner join tblstudent on TblClass.tClassId=TblStudent.tSClassId  --查询：学生的姓名及英语数学分数  select tSName,tEnglish,tMath  from TblStudent  inner join tblscore on TblStudent.tSId=TblScore.tSId |

#### 案例

|  |
| --- |
| --查询所有英语及格的学生姓名、年龄及成绩  --1：结果数据：姓名tblstudent.tsname，年龄sblstudent.tsage，成绩tblscore.tenglish  --2：条件：英语成绩>60  --3：需要使用连接查询，使用内连接，关系：tblstudent.tsid=tblscore.tsid  select \* from TblStudent  select \* from TblScore  select tsname,tsage,tenglish  from TblStudent  inner join TblScore on TblStudent.tSId=TblScore.tSId  where tEnglish>60  --查询所有没有参加考试的学生的姓名  --1、结果集：姓名tblstudent.tsname  --2、条件：学生编号不在tblscore表中出现 not in()  select tsname from TblStudent  where tSId not in(select tSId from TblScore)  --分组使用多个属性  --查询：每个班级的各地区有多少人  select tSClassId,tSAddress,COUNT(\*)  from TblStudent  group by tSAddress,tSClassId  ----找出各班中最多人的城市名称  --1、结果集：班级编号tsclassid,城市名称tsaddress  --2、条件：各班级：group by tsclassid,tsaddress  select tsclassid,COUNT(\*)  from TblStudent  group by tSClassId  --计算出各员工的工龄：强调，给2年的员工发奖金  select \*  from TblEmployee  where DATEDIFF(DAY,EmpJoinDate,GETDATE())+1>=730  --计算两个日期之间相差\*年\*月\*天  select \* from TblStudent  --求两个日期之间差“\*年\*月\*天”  --设定:一个月按30天算  --测试数据一：2012-12-31，2013-1-1  --如果日期差值小于30天,则直接输出相差天数  select DATEDIFF(day,'2012-12-31','2013-1-1')  --测试数据二：2012-10-14，2013-3-12  --如果日期差值大于30天,并且小于12个月,则输出相差月数及天数  select DATEDIFF(DAY,'2012-10-14','2013-3-12')/30,  DATEDIFF(DAY,'2012-10-14','2013-3-12')%30  --测试数据三：1994-02-03,2013-10-13  --如果日期差值大于30天，并且大于12个月，则输出相差年数月数天数  select DATEDIFF(DAY,'1994-02-03','2013-10-13')/360,  DATEDIFF(DAY,'1994-02-03','2013-10-13')%360/30,  DATEDIFF(DAY,'1994-02-03','2013-10-13')%30  select DATEDIFF(YEAR,'1994-2-3','2013-10-13'),  DATEDIFF(MONTH,'1994-2-3','2013-10-13')%12,  DATEDIFF(DAY,'1994-2-3','2013-10-13')%30 |

### 外连接

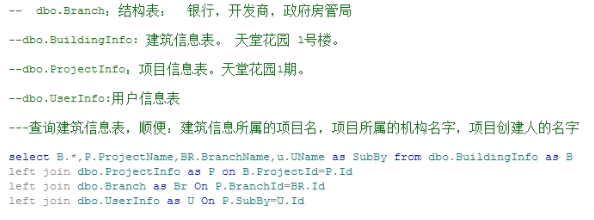
#### 左连接（left join）

左表中的信息全部出现，右表中的信息必须能够匹配才出现。

未匹配项使用null填充。

|  |
| --- |
| select \* from Department left join Employee on department.DId=employee.edid |

##### 多表查询案例



#### 右连接（right join）

右表中的信息全部出现，左表中的信息必须能够匹配才出现。

未匹配项使用null填充。

|  |
| --- |
| select \* from Department right join Employee on department.DId=employee.edid |

#### 全连接（full outer join）

两个表中的信息全部出现，可以省略为full join

|  |
| --- |
| select \* from Department  full outer join  Employee on department.DId=Employee.edid |



### 自连接

特点：关系列引用本表的主键列信息。

如右图, 关系列OrgParent引用的都是本表中的主键信息。

下面的该图的SQL代码：

|  |
| --- |
| create table OrgStructure  (  Orgid int identity(1,1) not null primary key,  OrgDepartment nvarchar(20) not null,  OrgParent int  )  select \* from OrgStructure  insert into OrgStructure values('宇宙大学',0)  insert into OrgStructure values('电子信息工程学院',1)  insert into OrgStructure values('工商管理学院',1)  insert into OrgStructure values('音乐学院',1)  insert into OrgStructure values('建筑学院',1)  insert into OrgStructure values('医学院',1)  insert into OrgStructure values('电子工程系',2)  insert into OrgStructure values('计算机科学系',2)  insert into OrgStructure values('工商管理系',3)  insert into OrgStructure values('市场营销系',3)  insert into OrgStructure values('会计学系',3)  insert into OrgStructure values('信息管理系',3)  insert into OrgStructure values('声乐系',4)  insert into OrgStructure values('作曲系',4)  insert into OrgStructure values('音乐教育系',4)  insert into OrgStructure values('建筑系',5)  insert into OrgStructure values('城市规划系',5)  insert into OrgStructure values('临床医学系',6)  insert into OrgStructure values('中医学系',6)  insert into OrgStructure values('护理学系',6) |

连接的表都是同一个表，可以由内连接，外连接各种组合

|  |
| --- |
| create table groups(  gid int identity(1,1) primary key not null,  gname nvarchar(10),  gparent int  )  select \* from groups  insert into groups values('.net部门',3)  --当前表的含义：第一级分类：总部，第二级分类：分公司，第三级分类：部门  --查询部门名称  select gname from groups as department  --查询公司名称  select gname from groups as company  --查询部门名称及对应的公司名称  --需要从部门表和公司表连接查询  select department.gname,company.gname  from groups as department  inner join groups as company  on department.gparent=company.gid |

## 子查询

查询可以嵌套使用，这叫子查询

# 常用函数

## 类型转换

### cast

语法：

|  |
| --- |
| cast(值 as 类型) |

例如：

|  |
| --- |
| -- 两个值相加，需要类型相同。  update TblClass set cDescription = cDescription + CAST(cId as varchar(2)) |

### convert

语法：

|  |
| --- |
| convert(目标类型, 值) |

例如：

|  |
| --- |
| select CONVERT(float,'123.456')  --select支持格式化.  --日期格式转换为字符串，103表示格式为：dd/mm/yy.  select convert(varchar(10),GETDATE(),103) |

## 字符串函数

|  |
| --- |
| use SchoolDB  --求字符串的长度  select len('abcd')  --字符串转大写  select UPPER('abcd')  --字符串转小写  select LOWER('ABCD')  --字符串左侧的空格去掉  select LTRIM(' abcd')  --字符串右侧的空格去掉  select RTRIM('abcd ')  --字符串两侧的空格去掉  select LTRIM(RTRIM(' abcd '))  --截取前2个字符  select LEFT('abcd',2)  --截取后2个字符  select RIGHT('abcd', 2)  --从索引3开始截取四个字符。索引从1开始，而不是0.结果是cdef  select SUBSTRING('abcdef',3,4) |

# T-SQL

## case（判断）

用于在select字句中进行判断。有两种格式：判断相等和判断不等。

### 判断不想等

|  |
| --- |
| select tSName,  --tEnglish。当英语成绩低于60时显示为不及格，大于60显示以字符串显示成绩。没有成绩时显示为缺考  ISNULL(case when tEnglish<60 then '不及格'else CAST(tEnglish as nvarchar(10)) end ,'缺考')  from TblStudent  left join  tblscore  on tblstudent.tsid=tblscore.tsid |

#### 案例：流水账

|  |
| --- |
| --有一个财务流水表  --MoneyFlow fid fTitle fMoney  create table MoneyFlow  (  fid int identity(1,1) primary key not null,  ftitle nvarchar(10),  fmoney money  )  select \* from MoneyFlow  insert into MoneyFlow values('发工资',1000)  insert into MoneyFlow values('奖金',500)  insert into MoneyFlow values('捡钱',200)  insert into MoneyFlow values('请客',-1500)  insert into MoneyFlow values('洗脚',-3000)  select fid,ftitle,case when fmoney>0 then fmoney else 0 end as '收入',  case when fmoney<0 then ABS(fmoney) else 0 end as '支出'  from MoneyFlow |

### 判断相等

|  |
| --- |
| --将性别显示成男和女  select \*,case egender when 0 then '男' when 1 then '女' end  from Employee |

## 变量

|  |
| --- |
| --定义变量并初始化  declare @username nvarchar(10) = 'hyf'  --变量赋值  set @username='hao'  --输出  print @username  --变量赋值2  select @username ='yu'  --输出2  select @username |

### 全局变量

全局变量用于系统维护，不需要我们管理，用于查看信息。

|  |
| --- |
| --全局变量  select @@VERSION  select \* from Employee  insert into Employee(eName,eCode,edid) values('zjy',14,4)  select @@IDENTITY  insert into Employee values('yhb')  print @@error  print @@rowcount |

## 条件语句

|  |
| --- |
|  |

## 循环语句

|  |
| --- |
| declare @number int =1  declare @sum int =0;  --循环  while(@number<=100)  begin  set @sum+=@number  set @number+=1  end  select @sum |

## 开窗函数

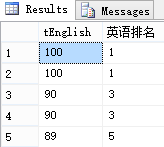
几个行作为一个区，就称为一个窗，能够进行按行划区的函数就是开窗函数。

### 排序

使用over子句与rank()结合使用，可以进行排名

|  |
| --- |
| select tEnglish ,rank() over(order by tenglish desc) as '英语排名' from tblscore |

效果如下：



### 分页

|  |
| --- |
| --分页,要求3条数据显示一页  select \* from(  select \*,row\_number() over(order by oprice desc) as num  from TblOrders  ) t1  where num between 4 and 6 |

### 划区

下面的代码按照代码分区，之后进行sum

|  |
| --- |
| select \*,SUM(ocount) over(partition by oname)  from TblOrders |

## 视图（view）

用于储存一个select语句。视图并不是一个存储数据的容器，而只是存储了一个查询。

好处：使用方便，安全（看不到表的名称、结构等信息）

使用：主要用于查询，与查询表的语法一样。

语法：

|  |
| --- |
| create view 视图名  as  select 语句 |

执行视图语句（相当于执行select 语句）:

|  |
| --- |
| select \* from 视图名 |

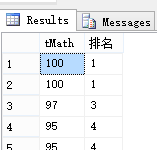
创建视图

|  |
| --- |
| create view VMath  as  select tMath,rank()over(order by tmath desc) as '排名'  from TblScore |

执行：

|  |
| --- |
| select \* from VMath |

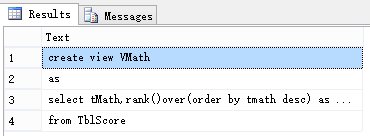
结果：



### 查看视图中的代码

|  |
| --- |
| exec sp\_helptext vmath |

结果：



## 事务（transaction）

保证多个操作全部成功，否则全部失败，这样的机制就是事务。

数据库中的事务：代码全部成功则提交，如果有某一条语句失败则回滚，整体失败。

事务操作：

begin tran：开始事务

commit tran: 提交，没错后执行

rollback tran：回滚，出错后执行

|  |
| --- |
| --声明变量用来判断是否出错  declare @errorNum int = 0  --开始事务  begin tran  --要执行的sql语句  insert into Orders values ('五粮液',20,300.5)  --变量赋值  set @errorNum+=@@ERROR  insert into Orders values ('茅台',20,300.5)  set @errorNum+=@@ERROR  --判断是否出错  if(@errorNum>0)  begin  --如果出错，回滚  rollback tran  end  --没有出错，提交事务，执行代码  commit tran |

### 显式事务

sql server 默认采用显式事务，即不需要明确调用就自动提交。如执行一个插入语句：

|  |
| --- |
| insert into Orders values ('茅台',20,300.5) |

sql server自动提交执行了事务。当别的用户进行查询时可以即时查询到。

### 隐式事务

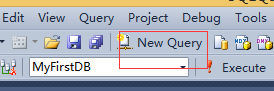
需要手动提交。设置语句为：

|  |
| --- |
| set implicit\_transactions on |

之后插入一条语句，并执行

|  |
| --- |
| insert into Orders values ('茅台',20,300.5) |

新建一个查询窗口：



进行查询：



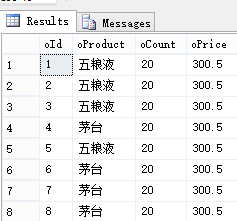
会发现一直在执行查询，没有结果。这是因为事务还没有提交，其他用户进行查询时一直在等待状态。



提交事务：

|  |
| --- |
| commit tran |

发现新建sql窗口中的查询结果显示出来了：



## 存储过程

就是一个函数，用于存储一段代码

好处：

* 完成代码的封装，实现代码的重用
* 安全
* 方便应用程序与数据库间通信，不用传大量sql语句过程，而只用传一个存储过程名称过来,简单方便。

自定义存储过程，语法：

|  |
| --- |
| create proc 存储过程名称  参数列表  as  自定义代码段 |

调用执行：

|  |
| --- |
| exec 存储过程名称 参数值 |

比如，去除字符串两边的空格：

|  |
| --- |
| --创建存储过程  create procedure MyTrim  --参数列表，多个参数以逗号隔开  @str varchar(10)  as  --自定义代码段  declare @str1 varchar(10)  set @str1= LTRIM(RTRIM(@str))  select @str1  --执行存储过程  exec MyTrim 'abc |

### 带输出值的存储过程

在参数列表中，将变量声明为output，在调用时也要明确指定参数为output。

|  |
| --- |
| --带输出参数的存储过程  --求两个数的和  create proc sum1  @num1 int,@num2 int ,@result int output  as  set @result=@num1+@num2  --声明变量,用来接收输出值  declare @r int  exec sum1 3,10,@r output  select @r |

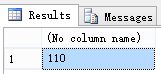
### 带默认值的存储过程

|  |
| --- |
| --带默认值的存储过程  create proc sum2  @num1 int,@num2 int =100--默认值为100  as  set @num1 += @num2  select @num1 |

执行存储过程，使用默认值：

|  |
| --- |
| exec sum2 10 |

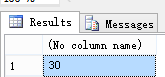
结果：



执行存储过程，不使用默认值：

|  |
| --- |
| exec sum2 10,20 |

结果：



## 索引

索引就像书的目录，找起来非常快捷方便。会把重复的数据放到一组中，这样寻找时先找到组，再查找速度就很快。

分类：

* 聚焦索引：与实际存储位置一样，一个表只能有一个聚焦索引；
* 非聚焦索引:逻辑上的一个划分

为什么快了？

建立索引，就分了区，排了序，找起来就快了。

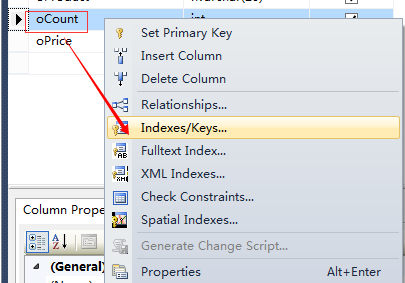
是否要为列建立索引呢?根据实际的sql语句进行分析，列出现在where中的次数较多，时间损耗较大，则为这个列建立索引。

注意：不是越多越好，因为每建立一个列索引，都要排序存储一次，会让数据库文件变大。

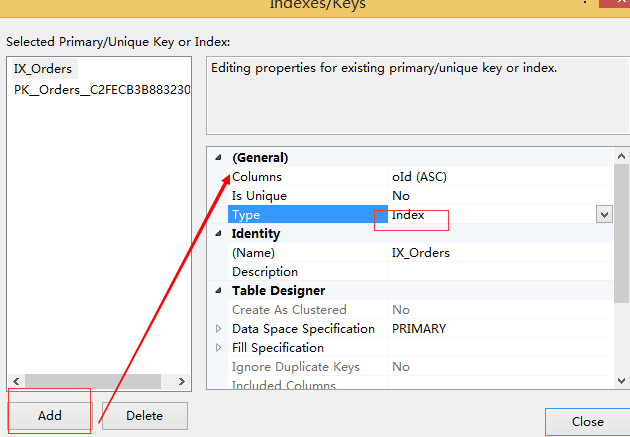
## 创建索引

### 使用SSMS创建

1. 在列上右击选择”索引/键”



1. 点击添加，在常规下修改要创建索引的列、是否唯一的、类型选择索引。



### 使用sql语句创建

语法：

|  |
| --- |
| create [unique] index 索引名  on 表名(列名) |

例如：

|  |
| --- |
| create index IX\_myOrder  on orders(oProduct) |

## 触发器

作用：对表进行增、删、改时，自动进行一个操作。

根据触发机制不同，分为:

* after触发器：在触发源执行完成后执行触发器中的代码
* instead of 替换触发器：在触发源执行前执行触发器中的代码，同时触发源的操作被废弃。

应用背景：

当有关联操作的时候。比如，下订单时，创建中的商品数量需要减少；退票时，总的票量要增加。

关键：确定要将触发器建立在哪个表上

触发源: insert update delete

临时表：inserted,deleted。

触发器效率低，综合考虑使用存储过程还是触发器那个效率高。

### 创建触发器

语法：

|  |
| --- |
| create trigger 名称  on 表名  [after][instead of] [insert|delete|update]  as  begin  ….  end |

### ater触发器

例如，在orders表中插入一个数据，会在bak\_orders表中自动备份。首先在orders表中创建触发器。

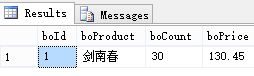
|  |
| --- |
| create trigger bak\_order  on orders  after insert  as  begin  insert into Bak\_Orders(boProduct,boCount,boPrice)  select oProduct,oCount,oPrice from inserted  end |

插入数据：

|  |
| --- |
| insert into Orders values ('剑南春',30,130.45) |

bak\_orders表自动插入数据：

|  |
| --- |
| select \* from bak\_orders |



### instead of触发器

更新orders表中的数据时，rders表中的数据放弃更新去更新bak\_orders标准的数据:

|  |
| --- |
| create trigger update\_order  on orders  instead of update  as  begin  update Bak\_Orders set boCount=(select ocount from inserted),boprice=(select ocount from inserted) where boProduct in (select oProduct from inserted)  end |

更新数据：

|  |
| --- |
| update Orders set oCount=300,oPrice=300 where oProduct='剑南春' |

结果：



## 游标

可以逐条操作表中的数据，针对每一行数据进行操作。

缺点:性能低，慎重使用。

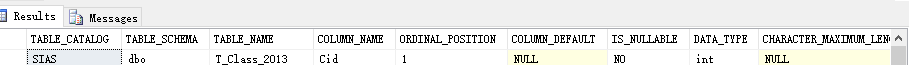
例如，想要为order表中每一行的ocount字段增加一个随机数，设计到每一行的操作就需要要用游标。方法如下：

|  |
| --- |
| --1.定义游标  declare myCurosr cursor for select oproduct,ocount,oprice from orders  --2.打开游标  open myCurosr  --3.在访问游标中的某行数据时,需要声明变量  declare @product nvarchar(10)  declare @count int  declare @price float  --4.通过游标从结果集中取数据,  ------取得是第一行的数据。（游标是一行行取数据）  fetch myCurosr into @product,@count,@price  --5.循环执行，修改每一行的数据。  while(@@FETCH\_STATUS=0)--@@FETCH\_STATUS=0表示游标指向下一行有数据。  begin  set @count = @count + rand()\*100--数量加上随机数。  --5.1. 把新值更新回去。  update Orders set oCount = @count where oProduct=@product and oPrice=@price  --5.2. 取下一行  fetch myCurosr into @product,@count,@price  end  --6.关闭游标  close myCurosr  --7.释放游标  deallocate myCurosr |

# 元数据

## INFORMATION\_SCHEMA.COLUMNS

获取当前数据库的字段信息



这是个存储过程，可以用sp\_helptext查看该存储过程：

|  |
| --- |
| sp\_helptext "INFORMATION\_SCHEMA.COLUMNS" |

Data Source=.;Initial Catalog=SIAS;User Id=sa;Password=123

# 总结

