☆☆☆

☆sqlserver 数据库操作大全——常用语句/技巧集锦/经典语句

☆☆☆

MS SQL 里没有括号时，运算进行的次序将是先乘后除再模后加减

减号(-)也有两种用途：1.作为负号使用 2.从某一列中减去另一列

and or not

如果一个 **where** 子句中同时出现这三个操作符

最先评估 not 然后是 and 然后是 or

coalesce 哪个不为空用哪个

coalesce(i.ProductID,d.ProductID)

@@rowcount

返回上一条语句影响的行数

SQL 判断某列中是否包含中文字符或者英文字符

select * **from** 表名 **where** 某列 like '%[吖-座]%'

select * **from** 表名 **where** 某列 like '%[a-z]%'

--数据操作，中英文对照

select --从数据库表中检索数据行和列

insert --向数据库表添加新数据行

delete --从数据库表中删除数据行

update --更新数据库表中的数据

--数据定义

create table --创建一个数据库表

drop table --从数据库中删除表

alter table --修改数据库表结构

create view --创建一个视图

drop view --从数据库中删除视图

create index --为数据库表创建一个索引

drop index --从数据库中删除索引

create proceduer --创建一个存储过程

drop proceduer --从数据库中删除存储过程

create trigger --创建一个触发器

drop trigger --从数据库中删除触发器

create schema --向数据库添加一个新模式

drop schema --从数据库中删除一个模式

create domain --创建一个数据值域

alter domain --改变域定义

drop domain --从数据库中删除一个域

--数据控制

grant --授予用户访问权限

deny --拒绝用户访问

revoke --解除用户访问权限

--事务控制

commit --结束当前事务

rollback --中止当前事务

set transaction --定义当前事务数据访问特征

--程序化 SQL

declare --为查询设定游标

explan --为查询描述数据访问计划

open --检索查询结果打开一个游标

fetch --检索一行查询结果

close --关闭游标

prepare --为动态执行准备 SQL 语句

execute --动态地执行 SQL 语句

describe --描述准备好的查询

----------SQL 中插入数据的技巧 ----------

插入少量数据时可以用：

insert into 表名(列名1，列名2...) **values** (值1，值2...)

插入大量数据时可以用：

insert into 要复制的表名 **select** * **from** 源表名

或：**insert into** 要复制的表名(列名1，列名2...) **select** (列名1，列名2...) **from** 源表名

insert select 语句要求你遵循如下规则：

SELECT 语句不能从被插入数据的表中选择行

INSERT INTO 中的列数必须与 **SELECT** 语句返回的列数相等

INSERT INTO 中的数据类型要与 **SELECT** 语句返回的数据类型相同

insert select 语句的另外一个用处对表进行备份：

select * **into** 临时表名 **from** 源表名

----------SQL 中 truncate table 和 delete 和 drop 的区别 ----------

truncate table 在功能上与不带 **Where** 子句的 **Delete** 语句相同,二者均删除表中的全部行

但 **truncate table** 比 **Delete** 速度快，且使用的系统和事务日志资源少。

Delete 语句每次删除一行，并在事务日志中为所删除的每行记录一项。

truncate table 通过释放存储表数据所用的数据页来删除数据，并且只在事务日志中记录页的释放。

truncate table 删除表中的所有行，但表结构及其列、约束、索引等保持不变。

新行标识所用的计数值重置为该列的种子。如果想保留标识计数值，请改用 **Delete**。

对于由 **FOREIGN KEY** 约束引用的表，不能使用 **truncate table**，而应使用不带 **Where** 子句的 **Delete** 语句。

由于 **truncate table** 不记录在日志中，所以它不能激活触发器。

truncate table 不能用于参与了索引视图的表。

truncate,**delete**,**drop** 的异同点：

注意:这里说的 **delete** 是指不带 **where** 子句的 **delete** 语句

相同点:**truncate** 和不带 **where** 子句的 **delete**, 以及 **drop** 都会删除表内的数据

不同点:

1.**truncate** 和 **delete** 只删除数据不删除表的结构(定义)

drop 语句将删除表的结构被依赖的约束(constrain),触发器(**trigger**),索引(**index**);

依赖于该表的存储过程/函数将保留,但是变为 invalid 状态.

2.**delete** 语句是 dml,这个操作会放到 **rollback** segement 中,事务提交之后才生效;

如果有相应的 **trigger**,执行的时候将被触发.

truncate,**drop** 是 ddl, 操作立即生效,原数据不放到 **rollback** segment 中,不能回滚. 操作不触发 **trigger**.

3.**delete** 语句不影响表所占用的 extent, 高水线(high w2atermark)保持原位置不动

显然 **drop** 语句将表所占用的空间全部释放

truncate 语句缺省情况下将空间释放到 minextents 个 extent,除非使用 reuse storage;

truncate 会将高水线复位(回到最开始).

4.速度,一般来说: **drop**> **truncate** > **delete**

5.安全性:小心使用 **drop** 和 **truncate**,尤其没有备份的时候.否则哭都来不及

使用上:

想删除部分数据行用 **delete**,注意带上 **where** 子句. 回滚段要足够大.

想删除表,当然用 **drop**

想保留表而将所有数据删除. 如果和事务无关,用 **truncate** 即可.

如果和事务有关,或者想触发 **trigger**,还是用 **delete**.

如果是整理表内部的碎片,可以用 **truncate** 跟上 reuse stroage,再重新导入/插入数据

----------好句收藏----------

1.**UNION** 将返回两个查询的结果并去除其中的重复部分

```
SELECT NAME FROM SOFTBALL
UNION
SELECT NAME FROM FOOTBALL
```

distinct (去掉完全重复的记录)

select distinct * **from** 表名

with ties (并列显示完全重复的记录)

select Top 10 **with** ties * **from** 表名

2.**UNION** ALL 不去掉重复的记录

SELECT NAME FROM SOFTBALL

UNION ALL

SELECT NAME FROM FOOTBALL

3.**INTERSECT** 返回两个表中共有的行

SELECT * **FROM** FOOTBALL

INTERSECT

SELECT * **FROM** SOFTBALL

4.MINUS (相减)

返回的记录是存在于第一个表中但不存在于第二个表中的记录

SELECT * **FROM** FOOTBALL

MINUS

SELECT * **FROM** SOFTBALL

5.in(满足括号里任意一个条件即可)

SELECT * **FROM** FRIENDS **WHERE** STATE IN('CA','CO','LA')

6.BEWTEEN (满足区间)

SELECT * **FROM** PRICE **WHERE** WHOLESALE BETWEEN 0.25 AND 0.75

7.连接(||)

可以将两个字符串连接起来

SELECT (NAMEa || NAMEb) **as** 新列名 **FROM** FRIENDS

格式化连接

SELECT (NAMEa || ',' || NAMEb) **as** 新列名 **FROM** FRIENDS --在连接的数据中间加上自定义字符串

8.STARTING **WITH** (它的作用与 like 相似,用之前要测试 sql 解释器是否支持此语法)

SELECT 列名1，列名2... **FROM** 表名 **WHERE** 列名 STARTING **WITH**('Ca')

ORDER BY (排序)

升序：**SELECT** * **FROM** 表名 **ORDER BY** 列名

降序：**SELECT** * **FROM** 表名 **ORDER BY** 列名 **DESC**

技巧：假如你已经知道了你想要进行排序的列是表中的第一列的话，那么你可以用 **ORDER BY** 1 来代替输入列的名字

GROUP BY (分组查询)

SELECT 列名 **FROM** 表名 **GROUP BY** 列名

HAVING (对你需要进行分组的数据进行限制)

SELECT 列名1，AVG(列名2) **FROM** 表名 **GROUP BY** 列名 **HAVING** AVG(列名3)>66

---表的不等值联合

不等值联合则是在 **WHERE** 子句中使用除了等号以外的其它比较运算符

例句：**SELECT** O.**NAME**,O.PARTNUM,P.PARTNUM **FROM** ORDERS **as** O,PART **as** P **WHERE** O.PARTNUM > P.PARTNUM

---表的自我联合

WHERE F.PARTNUM = S.PARTNUM AND F.DESCRIPTION <> S.DESCRIPTION

----------日期函数----------

VARIANCE (返回某一列数值的方差)

例句：**SELECT** VARIANCE(列名) **AS** 新列名 **FROM** 表名 --列必须为 int 或 double 等数值类型

STDDEV (返回某一列数值的标准差)

例句：**SELECT** STDDEV(列名) **AS** 新列名 **FROM** 表名 --列必须为 int 或 double 等数值类型

ADD_MONTHS (该函数的功能是将给定的日期增加一个月)

例句：**SELECT** ADD_MONTHS(ENDDATE,1) **AS** 新列名 **FROM** 表名 --ENDDATE 为 datetime 类型

ADD_MONTHS(ENDDATE,1)可以使用于 **where** 条件

LAST_DAY (可以返回指定月份的最后一天是几号)

例句：**SELECT** LAST_DAY(ENDDATE) **AS** 新列名 **FROM** 表名

DISTINCT (得到唯一的结果,就是去掉重复的结果)

例句：**SELECT DISTINCT** 列名 **FROM** 表名

MONTHS_BETWEEN (得到给定的两个日期中有多少个月)

例句1：**SELECT** MONTHS_BETWEEN(ENDDATE,STARTDATE) **AS** 新列名 **FROM** 表名

例句2：**SELECT** * **FROM** 表名 **WHERE** MONTHS_BETWEEN(DATETIME1,DATETIME2)>0

SYSDATE (将返回系统的日期和时间)

例句：**SELECT DISTINCT** SYSDATE **FROM** 表名

----------数学函数----------

ABS() 函数返回给定数字的绝对值

CEIL() 返回与给定参数相等或比给定参数在的最小整数

FLOOR() 返回与给定参数相等或比给定参数在的最大整数

MOD(A,B) 返回 A 与 B 相除后的余数

SIGN() 如果参数的值为负数返回-1 ,如果参数的值为正数返回1 ,如果参数为零返回零

SQRT() 该函数返回参数的平方根,由于负数是不能开平方的所以不能将该函数应用于负数

----------字符函数----------

CHR() 该函数返回与所给数值参数等当的在 ASCLL 码字符,返回的字符取决于数据库所依赖的字符集

例句：**SELECT** CHR(列名) **FROM** 表名

CONCAT() 与||符号相同，表示将两个字符串连接起来

例句：**SELECT** CONCAT(列名1,列名2) **FROM** 表名

INITCAP() 该函数将参数的第一个字母变为大写,此外其它的字母则转换成小写

例句：**SELECT** INITCAP(列名) **as** 新列名 **FROM** 表名

LOWER() 将参数全部转换为小写字母

UPPER() 将参数全部转换为大写字母

LENGTH() 将返回指定字符串的长度

----------转换函数----------

TO_CHAR() 将一个数字转换为字符型

TO_NUMBER() 将一个字符串型数字转换为数值型

---其它函数

GREATEST() 将会返回在字母表中最靠后的字符开头的字符串,函数是返回几个表达式中最大的；

例句：**SELECT** GREATEST('ALPHA','BRAVO','FOXTROT','DELTA','FP') **FROM** 表名

LEAST() 函数是返回几个表达式中最小的！

例句：**SELECT DISTINCT** LEAST(34,567,3,45,1090) **FROM** 表名

USER() 函数将返回当前使用数据库的用户的名字

例句：**SELECT DISTINCT** USER **FROM** 表名

----------条件语句后用的关键字----------

EXISTS

从子查询中返回的行数至少有一行时，EXIST 返回为 **true**。返回为空时，EXIST 返回为 **false**。

例句：

SELECT NAME FROM ORDERS

WHERE EXISTS(**SELECT** * **FROM** ORDERS **WHERE NAME** ='MOSTLY HARMLESS')

ANY/SOME ANY 与 SOME 具有同样的功能

ANY 与子查询中的每一行与主查询进行比较,并对子查询中的每一行返回一个 **TRUE** 值

区别：

IN 只相当于多个等号的作用,IN 不能用于大于或小于的判断。

而 ANY 和 SOME 则可以使用其它的比较运算符如大于或小于。

例句：

SELECT NAME FROM ORDERS **WHERE NAME** > ANY

(**SELECT NAME FROM** ORDERS **WHERE NAME** ='JACKS BIKE')

ALL 关键字的作用在于子查询中的所有结果均满足条件时它才会返回 **TRUE**，ALL 常起双重否定的作用。

例句：

SELECT NAME FROM ORDERS **WHERE NAME** <> ALL

(**SELECT NAME FROM** ORDERS **WHERE NAME** ='JACKS BIKE')

----------局部变量和全局变量----------

局部变量必须以“@”开头，而且必须先用 **DECLARE** 命令说明后才可使用。

语法：**DECLARE** @变量名 变量类型

局部变量赋值必须使用 **SELECT** 或 **SET** 命令来设定变量的值

语法： **SELECT** @局部变量=变量值

SET @局部变量=变量值

全局变量不是由用户的程序定义的，它们是在服务器级定应义的。

只能使用预先说明及定义的变局变量。

引用全局变量时，必须以“@@”开头。

局部变量的名称不能与全局变量的名称相同、否则会在应用中出错

----------事务-视图-索引----------

事务是一种机制，用以维护数据库的完整性。

事务有4个属性：原子性（Atomicity）、一致性（Consistency）、隔离性（**Isolation**）以及持久性（Durability），也称作事务的 ACID 属性。

原子性：事务内的所有工作要么全部完成，要么全部不完成，不存在只有一部分完成的情况。

一致性：事务内的然后操作都不能违反数据库的然后约束或规则，事务完成时有内部数据结构都必须是正确的。

隔离性：事务直接是相互隔离的，如果有两个事务对同一个数据库进行操作，比如读取表数据。

任何一个事务看到的所有内容要么是其他事务完成之前的状态，要么是其他事务完成之后的状态。

一个事务不可能遇到另一个事务的中间状态。

持久性：事务完成之后，它对数据库系统的影响是持久的，即使是系统错误，重新启动系统后，该事务的结果依然存在。

事务的模式

a、 显示事务

显示事务就是用户使用 T-SQL 明确的定义事务的开始（**begin transaction**）和提交（**commit transaction**）或回滚事务（**rollback transaction**）

b、 自动提交事务

自动提交事务是一种能够自动执行并能自动回滚事务，这种方式是 T-SQL 的默认事务方式。

例如在删除一个表记录的时候，如果这条记录有主外键关系的时候，删除就会受主外键约束的影响，那么这个删除就会取消。

可以设置事务进入隐式方式：**set** implicit_transaction **on**;

c、 隐式事务

隐式事务是指当事务提交或回滚后，SQL Server 自动开始事务。因此，隐式事务不需要使用 **begin transaction** 显示开始，

只需直接失业提交事务或回滚事务的 T-SQL 语句即可。

使用时，需要设置 **set** implicit_transaction **on** 语句，将隐式事务模式打开，下一个语句会启动一个新的事物，再下一个语句又将启动一个新事务。

开始事务： **begin transaction**

提交事务： **commit transaction**

回滚事务： **rollback transaction**

创建视图：**create view** 视图名 **as** <**select** 语句>

删除视图：**drop view** 试图名

视图定义中的 **select** 语句中不能包括下列：

1.**order by** 子句，除非 **select** 语句的选择列有 **top** 子句

2.**into** 关键字

3.引用临时表或变量

创建索引：**create unique** 【clustered | nonclustered】 **index** 索引名 **on** 表名(列名) 【**with** fillfactor=x】

unique 可选，指定唯一索引

clustered , nonclustered 可选，指定是聚集索引或非聚集索引

fillfactor 可选，表示填充因子，指定一个0-100的值，该值指示索引页填满的空间所占的百分比

删除索引：**drop index** 表名.索引名

下面的表总结了何时使用聚集索引或非聚集索引(很重要)。

动作描述	使用聚集索引	使用非聚集索引
外键列	应	应
主键列	应	应
列经常被分组排序(**order by**)	应	应
返回某范围内的数据	应	不应
小数目的不同值	应	不应
大数目的不同值	不应	应
频繁更新的列	不应	应
频繁修改索引列	不应	应
一个或极少不同值	不应	不应

----------操作数据库----------

SQL 分类：

DDL 类型包括数据库、表的创建，修改，删除，声明—数据定义语言(**CREATE**，**ALTER**，**DROP**，**DECLARE**)

DML 类型包括数据表中记录的查询，删除，修改，插入—数据操纵语言(**SELECT**，**DELETE**，**UPDATE**，**INSERT**)

DCL 类型包括数据库用户赋权，废除用户访问权限， 提交当前事务，中止当前事务—数据控制语言(**GRANT**，**REVOKE**，**COMMIT**，**ROLLBACK**)

首先,简要介绍基础语句：

1、说明：创建数据库

```
CREATE DATABASE db1(db1代表数据库，可自命名)
on       primary        --默认属于 primary 主文件组，可省略
(
 --数据文件的具体描述
 name='MySchool_data',          --主数据文件的逻辑名称
 filename='D:\project\MySchool_data.mdf', --主数据文件的物理名称
 size=5MB,                      --主数据文件的初始大小
 maxsize=100MB,                  --主数据文件增长的最大值
 filegrowth=15%                 --主数据文件的增长率
)
log on
(
 --日记文件的具体描述，各参数含义同上
 name='MySchool_log',
 filename='D:\project\MySchool_data.ldf',
 size=2MB,
```

filegrowth=1MB

)

2、说明：删除数据库

drop database db1(db1代表数据库，可自命名)

3、说明：备份 sql server

--- 创建 备份数据的 device

USE master

EXEC sp_addumpdevice 'disk', 'testBack', 'c:\mssql7backup\MyNwind_1.dat'

--- 开始 备份

BACKUP **DATABASE** pubs **TO** testBack

4、说明：创建新表

create table tb1

(

Id **int** not null **primary key**, --设置为主键

one **int** identity(1,1), --设为标识列

name varchar not null, --非空

phone nvarchar(100), --可以为空

...

)

根据已有的表创建新表：

A：**create table** tab_new like tab_old (使用旧表创建新表)

B：**create table** tab_new **as select** col1,col2… **from** tab_old definition **only**

5、说明：

删除新表：**drop table** tb1

use　MySchool　--将当前数据库设置为 MySchool

if　exists(**select** * **from** MySchool **where name**='Student') --exist 是查询语句，检测某个查询是否存在

drop table Student

6、说明：

增加一个列：**Alter table** 表名 **add** 字段名 字段类型 字段说明/约束

添加带主键及约束的语法：

alter table　表名

add constraint　约束名　约束类型　具体的约束说明

--添加主键约束(将 StudentNo 作为主键)

alter table Student

add constraint PK_stuNo **primary key** (StudentNo)

--添加唯一约束(身份证号唯一)

alter table Student

add constraint UQ_stuID **unique** (身份证号列名)

--添加默认约束(如果地址不填，默认为“地址不详”)

alter table Student

add constraint DF_stuAddress **default** ('地址不详') **for** Address

--添加检查约束(要求出生日期在1980年1月1日之后)

alter table Student

add constraint CK_stuBornDate **check** (BornDate >= '1980-01-01')

--添加外键约束(主表 Student 和从表 Result 建立关系，关联列为 StudentNo)

alter table Result

add constraint FK_stuNo

foreign key (stuNo) **references** Student(stuNo)

删除列的语法：

你删除的时候会提示你，有默认约束依赖该字段，那么你需要先删除默认约束（错误提示里会有默认约束名），再删除字段：

ALTER TABLE 表名 **DROP CONSTRAINT** 默认约束名

GO

ALTER TABLE 表名 **DROP COLUMN** 字段名

GO

删除约束：

alter table Student

drop constraint 约束名

例句：

alter table Student

add constraint PK_stuNo

7、说明：

添加主键：**Alter table** tabname **add primary key**(ID)(设置某字段为主键，ID 可自由设置，主键数据不可重复）

说明：

删除主键：**Alter table** tabname **drop primary key**(ID)（删除某字段主键）

8、说明：

创建索引：**create** [**unique**] **index** idxname **on** tabname(col….)

删除索引：**drop index** idxname

注：索引是不可更改的，想更改必须删除重新建。

9、说明：

创建视图：**create view** viewname **as select** statement

删除和修改视图

alter view yourviewname **as**...

drop view yourviewname **as**...

加密视图

alter view yourviewname **with** encryption **as**...

加密了之后连你自己也看不到原代码了

10、说明：几个简单的基本的 sql 语句

选择：**select** * **from** table1 **where** Id=1(Id=1为条件语句，根据自己情况自定义）

插入：**insert into** table1(field1,field2) **values**(value1,value2)

删除：**delete from** table1 **where** 范围

更新：**update** table1 **set** field1=value1 **where** 范围

查找：**select** * **from** table1 **where** field1 like '%value1%' ---like 的语法很精妙，查资料!

排序：**select** * **from** table1 **order by** field1,field2 [**desc**]

总数：**select** count * **as** totalcount **from** table1

求和：**select** sum(field1) **as** sumvalue **from** table1

平均：**select** avg(field1) **as** avgvalue **from** table1

最大：**select max**(field1) **as** maxvalue **from** table1

最小：**select min**(field1) **as** minvalue **from** table1

11、说明：几个高级查询运算词

A： **UNION** 运算符

UNION 运算符通过组合其他两个结果表(例如 TABLE1 和 TABLE2)并消去表中任何重复行而派生出一个结果表。

当 ALL 随 **UNION** 一起使用时(即 **UNION** ALL)，不消除重复行。两种情况下，派生表的每一行不是来自 TABLE1 就是来自 TABLE2。

B： **EXCEPT** 运算符

EXCEPT 运算符通过包括所有在 TABLE1 中但不在 TABLE2 中的行并消除所有重复行而派生出一个结果表。

当 ALL 随 **EXCEPT** 一起使用时 (**EXCEPT** ALL)，不消除重复行。

C： **INTERSECT** 运算符

INTERSECT 运算符通过只包括 TABLE1 和 TABLE2 中都有的行并消除所有重复行而派生出一个结果表。

当 ALL 随 **INTERSECT** 一起使用时 (**INTERSECT** ALL)，不消除重复行。

注：使用运算词的几个查询结果行必须是一致的。

12、说明：使用外连接

A、left outer join：

左外连接(左连接)：结果集几包括连接表的匹配行，也包括左连接表的所有行。

SQL: **select** a.a, a.b, a.c, b.c, b.d, b.f **from** a LEFT **OUT** JOIN b **ON** a.a = b.c

B：right outer join:

右外连接(右连接)：结果集既包括连接表的匹配连接行，也包括右连接表的所有行。

C：**full** outer join：

全外连接：不仅包括符号连接表的匹配行，还包括两个连接表中的所有记录。

其次，大家来看一些不错的 sql 语句

1、说明：复制表(只复制结构,源表名：a 新表名：b) (Access 可用)

法一：**select** * **into** b **from** a **where** 1<>1 （仅用于 SQlServer）

法二：**select top** 0 * **into** b **from** a

2、说明：拷贝表(拷贝数据,源表名：a 目标表名：b) (Access 可用)

insert into b(a, b, c) **select** d,e,f **from** b;

3、说明：跨数据库之间表的拷贝(具体数据使用绝对路径) (Access 可用)

insert into b(a, b, c) **select** d,e,f **from** b in '具体数据库' **where** 条件

例子：..**from** b in '"&Server.MapPath(".")&"\data.mdb" &"' **where**..

4、说明：子查询(表名1：a 表名2：b)

select a,b,c **from** a **where** a IN (**select** d **from** b) 或者: **select** a,b,c **from** a **where** a IN (1,2,3)

5、说明：显示文章、提交人和最后回复时间

select a.title,a.username,b.adddate **from table** a,(**select max**(adddate) adddate **from table where table**.title=a.title) b

6、说明：外连接查询(表名1：a 表名2：b)

select a.a, a.b, a.c, b.c, b.d, b.f **from** a LEFT **OUT** JOIN b **ON** a.a = b.c

7、说明：在线视图查询(表名1：a)

```
select * from (SELECT a,b,c FROM a) T where t.a > 1;
```

8、说明：between 的用法,between 限制查询数据范围时包括了边界值,not between 不包括

```
select * from table1 where time between time1 and time2
```

```
select a,b,c, from table1 where a not between 数值1 and 数值2
```

9、说明：in 的使用方法

```
select * from table1 where a [not] in (‘值1’,’值2’,’值4’,’值6’)
```

10、说明：两张关联表，删除主表中已经在副表中没有的信息

```
delete from table1 where not exists ( select * from table2 where table1.field1=table2.field1 )
```

11、说明：四表联查问题：

```
select * from a left inner join b on a.a=b.b right inner join c on a.a=c.c inner join d on a.a=d.d where .....
```

12、说明：日程安排提前五分钟提醒

SQL: select * from 日程安排 where datediff('minute',f 开始时间,getdate())>5

13、说明：一条 sql 语句搞定数据库分页

```
select top 10 b.* from (select top 20 主键字段,排序字段 from 表名 order by 排序字段 desc) a,表名 b where b.主键字段 = a.主键字段 order by a.排序字段
```

14、说明：前10条记录

```
select top 10 * form table1 where 范围
```

15、说明：选择在每一组 b 值相同的数据中对应的 a 最大的记录的所有信息(类似这样的用法可以用于论坛每月排行榜,每月热销产品分析,按科目成绩排名,等等.)

```
select a,b,c from tablename ta where a=(select max(a) from tablename tb where tb.b=ta.b)
```

16、说明：包括所有在 TableA 中但不在 TableB 和 TableC 中的行并消除所有重复行而派生出一个结果表

```
(select a from tableA ) except (select a from tableB) except (select a from tableC)
```

17、说明：随机取出10条数据

```
select top 10 * from tablename order by newid()
```

18、说明：随机选择记录

```
select newid()
```

19、说明：删除重复记录

```
Delete from tablename where id not in (select max(id) from tablename group by col1,col2,...)
```

20、说明：列出数据库里所有的表名

```
select name from sysobjects where type='U'
```

21、说明：列出表里的所有的

```
select name from syscolumns where id=object_id('TableName')
```

22、说明：列示 type、vender、pcs 字段，以 type 字段排列，case 可以方便地实现多重选择，类似 select 中的 case。

```
select type,sum(case vender when 'A' then pcs else 0 end),sum(case vender when 'C' then pcs else 0 end),
        sum(case vender when 'B' then pcs else 0 end)
```

FROM tablename

group by type

显示结果：

type vender pcs

电脑 A 1

电脑 A 1

光盘 B 2

光盘 A 2

手机 B 3

手机 C 3

23、说明：初始化表 table1

TRUNCATE TABLE table1

24、说明：选择从10到15的记录

select top 5 * **from** (**select top** 15 * **from table order by** id **asc**) table_别名 **order by** id **desc**

随机选择数据库记录的方法(使用 Randomize 函数，通过 SQL 语句实现)

对存储在数据库中的数据来说，随机数特性能给出上面的效果，但它们可能太慢了些。

你不能要求 ASP“找个随机数”然后打印出来。实际上常见的解决方案是建立如下所示的循环：

Randomize

```
RNumber = Int(Rnd*499) +1

While Not objRec.EOF

If objRec("ID") = RNumber THEN

... 这里是执行脚本 ...

end if

objRec.MoveNext

Wend
```

这很容易理解。首先，你取出1到500范围之内的一个随机数(假设500就是数据库内记录的总数)。

然后，你遍历每一记录来测试 ID 的值、检查其是否匹配 RNumber。满足条件的话就执行由 **THEN** 关键字开始的那一块代码。

假如你的 RNumber 等于495，那么要循环一遍数据库花的时间可就长了。

虽然500这个数字看起来大了些，但相比更为稳固的企业解决方案这还是个小型数据库了，

后者通常在一个数据库内就包含了成千上万条记录。这时候不就死定了?

采用 SQL，你就可以很快地找出准确的记录并且打开一个只包含该记录的 recordset，如下所示：

```
Randomize

RNumber = Int(Rnd*499) + 1

SQL = "SELECT * FROM Customers WHERE ID = " & RNumber

set objRec = ObjConn.Execute(SQL)

Response.WriteRNumber & " = " & objRec("ID") & " " & objRec("c_email")
```

不必写出 RNumber 和 ID，你只需要检查匹配情况即可。只要你对以上代码的工作满意，你自可按需操作“随机”记录。

Recordset 没有包含其他内容，因此你很快就能找到你需要的记录这样就大大降低了处理时间。

再谈随机数

现在你下定决心要榨干 Random 函数的最后一滴油，那么你可能会一次取出多条随机记录或者想采用一定随机范围内的记录。

把上面的标准 Random 示例扩展一下就可以用 SQL 应对上面两种情况了。

为了取出几条随机选择的记录并存放在同一 recordset 内，你可以存储三个随机数，然后查询数据库获得匹配这些数字的记录：

```
SQL = "SELECT * FROM Customers WHERE ID = " & RNumber & " OR ID = " & RNumber2 & " OR ID = " & RNumber3
```

假如你想选出10条记录(也许是每次页面装载时的10条链接的列表)，你可以用 BETWEEN 或者数学等式选出第一条记录和适当数量的递增记录。

这一操作可以通过好几种方式来完成，但是 **SELECT** 语句只显示一种可能(这里的 ID 是自动生成的号码)：

```
SQL = "SELECT * FROM Customers WHERE ID BETWEEN " & RNumber & " AND " & RNumber & "+ 9"
```

注意：以上代码的执行目的不是检查数据库内是否有9条并发记录。

随机读取若干条记录，测试过

Access 语法：**SELECT top** 10 * **From** 表名 **ORDER BY** Rnd(id)

Sql server:**select top** n * **from** 表名 **order by** newid()

mysql **select** * **From** 表名 **Order By** rand() Limit n

Access 左连接语法(最近开发要用左连接,Access 帮助什么都没有,网上没有 Access 的 SQL 说明,只有自己测试, 现在记下以备后查)

语法 **select** table1.fd1,table1,fd2,table2.fd2 **From** table1 left join table2 **on** table1.fd1,table2.fd1 **where** ...

使用 SQL 语句 用...代替过长的字符串显示

语法：

SQL 数据库：**select** case **when** len(field)>10 **then** left(field,10)+'...' **else** field **end as** news_name,news_id **from** tablename

Access 数据库：**SELECT** iif(len(field)>2,left(field,2)+'...',field) **FROM** tablename;

Conn.**Execute** 说明

Execute 方法

该方法用于执行 SQL 语句。根据 SQL 语句执行后是否返回记录集，该方法的使用格式分为以下两种：

1.执行 SQL 查询语句时，将返回查询得到的记录集。用法为：

Set 对象变量名=连接对象.**Execute**("SQL 查询语言")

Execute 方法调用后，会自动创建记录集对象，并将查询结果存储在该记录对象中，通过 **Set** 方法，将记录集赋给指定的对象保存，以后对象变量就代表了该记录集对象。

2.执行 SQL 的操作性语言时，没有记录集的返回。此时用法为：

连接对象.**Execute** "SQL 操作性语句" [, RecordAffected][, **Option**]

·RecordAffected 为可选项，此出可放置一个变量，SQL 语句执行后，所生效的记录数会自动保存到该变量中。通过访问该变量，就可知道 SQL 语句队多少条记录进行了操作。

·**Option** 可选项，该参数的取值通常为 adCMDText，它用于告诉 ADO，应该将 **Execute** 方法之后的第一个字符解释为命令文本。通过指定该参数，可使执行更高效。

·BeginTrans、RollbackTrans、CommitTrans 方法

这三个方法是连接对象提供的用于事务处理的方法。BeginTrans 用于开始一个事物;RollbackTrans 用于回滚事务;CommitTrans 用于提交所有的事务处理结果，即确认事务的处理。

事务处理可以将一组操作视为一个整体，只有全部语句都成功执行后，事务处理才算成功;若其中有一个语句执行失败，则整个处理就算失败，并恢复到处里前的状态。

BeginTrans 和 CommitTrans 用于标记事务的开始和结束，在这两个之间的语句，就是作为事务处理的语句。

判断事务处理是否成功，可通过连接对象的 Error 集合来实现，若 Error 集合的成员个数不为0，则说明有错误发生，事务处理失败。

Error 集合中的每一个 Error 对象，代表一个错误信息。

SQL 语句大全精要

DELETE 语句

DELETE 语句：用于创建一个删除查询，可从列在 **FROM** 子句之中的一个或多个表中删除记录，且该子句满足 **WHERE** 子句中的条件，可以使用 **DELETE** 删除多个记录。

语法：**DELETE** [**table**.*] **FROM table WHERE** criteria

语法：**DELETE** * **FROM table WHERE** criteria='查询的字'

说明：**table** 参数用于指定从其中删除记录的表的名称。

criteria 参数为一个表达式，用于指定哪些记录应该被删除的表达式。

可以使用 **Execute** 方法与一个 **DROP** 语句从数据库中放弃整个表。不过，若用这种方法删除表，将会失去表的结构。

不同的是当使用 **DELETE**，只有数据会被删除;表的结构以及表的所有属性仍然保留，例如字段属性及索引。

UPDATE

有关 **UPDATE**，急!!!!!!!!!!!

在 ORACLE 数据库中

表 A (ID ,FIRSTNAME,LASTNAME)

表 B(ID,LASTNAME)

表 A 中原来 ID,FIRSTNAME 两个字段的数据是完整的

表 B 中原来 ID,LASTNAME 两个字段的数据是完整的

现在要把表 B 中的 LASTNAME 字段的相应的数据填入到 A 表中 LASTNAME 相应的位置。两个表中的 ID 字段是相互关联的。

```
update a set a.lastname=(select b.lastname from b where a.id=b.id)
```

常用 sql 语句命令的作用

1. 查看数据库的版本

```
 select @@version
```

2. 查看数据库所在机器操作系统参数

```
exec master..xp_msver
```

3. 查看数据库启动的参数

```
sp_configure
```

4. 查看数据库启动时间

```
select convert(varchar(30),login_time,120) from master..sysprocesses where spid=1
```

查看数据库服务器名和实例名

```
print 'Server Name.....：' + convert(varchar(30),@@SERVERNAME)
```

```
print 'Instance.....：' + convert(varchar(30),@@SERVICENAME)
```

5. 查看所有数据库名称及大小

```
sp_helpdb
```

重命名数据库用的 SQL

sp_renamedb 'old_dbname', 'new_dbname'

6. 查看所有数据库用户登录信息

sp_helplogins

查看所有数据库用户所属的角色信息

sp_helpsrvrolemember

修复迁移服务器时孤立用户时,可以用的 fix_orphan_user 脚本或者 LoneUser 过程

更改某个数据对象的用户属主

sp_changeobjectowner [@objectname =] 'object', [@newowner =] 'owner'

注意：更改对象名的任一部分都可能破坏脚本和存储过程。

把一台服务器上的数据库用户登录信息备份出来可以用 add_login_to_aserver 脚本

查看某数据库下,对象级用户权限

sp_helprotect

7. 查看链接服务器

sp_helplinkedsrvlogin

查看远端数据库用户登录信息

sp_helpremotelogin

8.查看某数据库下某个数据对象的大小

sp_spaceused @objname

还可以用 sp_toptables 过程看最大的 N(默认为50)个表

查看某数据库下某个数据对象的索引信息

sp_helpindex @objname

还可以用 SP_NChelpindex 过程查看更详细的索引情况

SP_NChelpindex @objname

clustered 索引是把记录按物理顺序排列的，索引占的空间比较少。

对键值 DML 操作十分频繁的表我建议用非 clustered 索引和约束，fillfactor 参数都用默认值。

查看某数据库下某个数据对象的的约束信息

sp_helpconstraint @objname

9.查看数据库里所有的存储过程和函数

use @database_name

sp_stored_procedures

查看存储过程和函数的源代码

sp_helptext '@procedure_name'

查看包含某个字符串@str 的数据对象名称

select distinct object_name(id) **from** syscomments **where** text like '%@str%'

创建加密的存储过程或函数在 **AS** 前面加 **WITH** ENCRYPTION 参数

解密加密过的存储过程和函数可以用 sp_decrypt 过程

10.查看数据库里用户和进程的信息

sp_who

查看 SQL Server 数据库里的活动用户和进程的信息

sp_who 'active'

查看 SQL Server 数据库里的锁的情况

sp_lock

进程号1--50是 SQL Server 系统内部用的,进程号大于50的才是用户的连接进程.

spid 是进程编号,dbid 是数据库编号,objid 是数据对象编号

查看进程正在执行的 SQL 语句

dbcc inputbuffer ()

推荐大家用经过改进后的 sp_who3过程可以直接看到进程运行的 SQL 语句

sp_who3

检查死锁用 sp_who_lock 过程

sp_who_lock

11.查看和收缩数据库日志文件的方法

查看所有数据库日志文件大小

dbcc sqlperf(logspace)

如果某些日志文件较大，收缩简单恢复模式数据库日志，收缩后@database_name_log 的大小单位为 M

backup log @database_name **with** no_log

dbcc shrinkfile (@database_name_log, 5)

12.分析 SQL Server SQL 语句的方法：

set statistics time {**on** | **off**}

set statistics io {**on** | **off**}

图形方式显示查询执行计划

在查询分析器->查询->显示估计的评估计划(D)-Ctrl-L　　或者点击工具栏里的图形

文本方式显示查询执行计划

set showplan_all {**on** | **off**}

set showplan_text { **on** | **off** }

set statistics profile { **on** | **off** }

13.出现不一致错误时，NT 事件查看器里出3624号错误，修复数据库的方法

先注释掉应用程序里引用的出现不一致性错误的表，然后在备份或其它机器上先恢复然后做修复操作

alter database [@error_database_name] **set** single_user

修复出现不一致错误的表

dbcc checktable('@error_table_name',repair_allow_data_loss)

或者可惜选择修复出现不一致错误的小型数据库名

dbcc checkdb('@error_database_name',repair_allow_data_loss)

alter database [@error_database_name] **set** multi_user

CHECKDB 有3个参数：

repair_allow_data_loss 包括对行和页进行分配和取消分配以改正分配错误、结构行或页的错误，以及删除已损坏的文本对象，这些修复可能会导致一些数据丢失。

修复操作可以在用户事务下完成以允许用户回滚所做的更改。

如果回滚修复，则数据库仍会含有错误，应该从备份进行恢复。

如果由于所提供修复等级的缘故遗漏某个错误的修复，则将遗漏任何取决于该修复的修复。

修复完成后，请备份数据库。

repai*_**st 进行小的、不耗时的修复操作，如修复非聚集索引中的附加键。

这些修复可以很快完成，并且不会有丢失数据的危险。

repair_rebuild 执行由 repai*_**st 完成的所有修复，包括需要较长时间的修复（如重建索引）。

执行这些修复时不会有丢失数据的危险。