## 通用SQL数据库的查询语句：

（注释：文中Transact-SQL查询只包括选择列表、FROM子句和WHERE子句。）

#### 一、 简单查询

简单的Transact-SQL查询只包括选择列表、FROM子句和WHERE子句。它们分别说明所查询列、查询的表或视图、以

及搜索条件等。例如，下面的语句查询testtable表中姓名为“张三”的nickname字段和email字段。

SELECT nickname,email FROM testtable WHERE name= '张三 '

**(一) select选择列表**

选择列表(select\_list)指出所查询列，它可以是一组列名列表、星号、表达式、变量(包括局部变量和全局变量)

等构成。

1、**选择所有列** 例如，下面语句显示testtable表中所有列的数据:

SELECT \* FROM testtable

2、**选择部分列** 并指定它们的显示次序查询结果集合中数据的排列顺序与选择列表中所指定的列名排列顺序相同。

SELECT nickname,email FROM testtable

3、**更改列标题** 在选择列表中，可重新指定列标题。定义格式为: 列标题=列名 列名 列标题

如果指定的列标题不是标准的标识符格式时，应使用引号定界符，例如，下列语句使用汉字显示列标题:

SELECT 昵称=nickname,电子邮件=email FROM testtable

4、**删除重复行** SELECT语句中使用ALL或DISTINCT选项来显示表中符合条件的所有行或删除其中重复的数据行，

默认为ALL。使用DISTINCT选项时，对于所有重复的数据行在SELECT返回的结果集合中只保留一行。

5、**限制返回的行数** 使用TOP n [PERCENT]选项限制返回的数据行数，TOP n说明返回n行，而TOP n PERCENT时，

说明n是表示一百分数，指定返回的行数等于总行数的百分之几。

SELECT TOP 2 \*FROM testtable

SELECT TOP 20 PERCENT \* FROM testtable

**(二)FROM子句**

FROM子句指定SELECT语句查询及与查询相关的表或视图。在FROM子句中最多可指定256个表或视图，它们之间用逗

号分隔。在FROM子句同时指定多个表或视图时，如果选择列表中存在同名列，这时应使用对象名限定这些列所属的表或视图。

例如在usertable和citytable表中同时存在cityid列，在查询两个表中的cityid时应使用下面语句格式加以限定:

SELECT username,citytable.cityid FROM usertable,citytable WHERE

usertable.cityid=citytable.cityid

在FROM子句中可用以下两种格式为表或视图指定别名: 表名 as 别名 表名 别名

例如上面语句可用表的别名格式表示为:

SELECT username,b.cityid FROM usertable a,citytable b

WHERE a.cityid=b.cityid

SELECT不仅能从表或视图中检索数据，它还能够从其它查询语句所返回的结果集合中查询数据。

SELECT a.au\_fname+a.au\_lname FROM authors a,titleauthor ta (SELECT title\_id,title

FROM titles WHERE ytd\_sales> 10000 ) AS t WHERE a.au\_id=ta.au\_id AND

ta.title\_id=t.title\_id

此例中，将SELECT返回的结果集合给予一别名t，然后再从中检索数据。

**(三) 使用WHERE子句设置查询条件**

WHERE子句设置查询条件，过滤掉不需要的数据行。例如下面语句查询年龄大于20的数据:

SELECT \* FROM usertable WHERE age> 20

WHERE子句可包括各种条件运算符:

比较运算符(大小比较):> 、> =、=、 <、 <=、 <> 、!> 、! <

范围运算符(表达式值是否在指定的范围):BETWEEN⋯AND⋯ NOT BETWEEN⋯AND⋯

列表运算符(判断表达式是否为列表中的指定项):IN (项1,项2⋯⋯)

NOT IN (项1,项2⋯⋯)

模式匹配符(判断值是否与指定的字符通配格式相符):LIKE、NOT LIKE

空值判断符(判断表达式是否为空):IS NULL、NOT IS NULL

逻辑运算符(用于多条件的逻辑连接):NOT、AND、OR

1、范围运算符例:age BETWEEN 10 AND 30相当于age> =10 AND age <=30

2、列表运算符例:country IN ( 'Germany ', 'China ')

3、模式匹配符例:常用于模糊查找，它判断列值是否与指定的字符串格式相匹配。可用于char、varchar、text、

ntext、datetime和smalldatetime等类型查询。

可使用以下通配字符:

百分号%:可匹配任意类型和长度的字符，如果是中文，请使用两个百分号即%%。

下划线\_:匹配单个任意字符，它常用来限制表达式的字符长度。

方括号[]:指定一个字符、字符串或范围，要求所匹配对象为它们中的任一个。[^]:其取值也[] 相同，

但它要求所匹配对象为指定字符以外的任一个字符。

例如:

限制以Publishing结尾，使用LIKE '%Publishing '

限制以A开头:LIKE '[A]% '

限制以A开头外:LIKE '[^A]% '

4、空值判断符例WHERE age IS NULL

5、逻辑运算符:优先级为NOT、AND、OR

**(四)查询结果排序**

使用ORDER BY子句对查询返回的结果按一列或多列排序。ORDER BY子句的语法格式为:

ORDER BY {column\_name [ASC &brvbar;DESC]} [,⋯n]

其中ASC表示升序，为默认值，DESC为降序。ORDER BY不能按ntext、text和image数据类型进行排序。

例如: SELECT \* FROM usertable ORDER BY age desc,userid ASC

另外，可以根据表达式进行排序。

#### 二、联合查询

UNION运算符可以将两个或两个以上上SELECT语句的查询结果集合合并成一个结果集合显示，即执行联合查询。

UNION的语法格式为:

select\_statement UNION [ALL] selectstatement [UNION [ALL] selectstatement][⋯n]

其中selectstatement为待联合的SELECT查询语句。

ALL选项表示将所有行合并到结果集合中。不指定该项时，被联合查询结果集合中的重复行将只保留一行。

联合查询时，查询结果的列标题为第一个查询语句的列标题。因此，要定义列标题必须在第一个查询语句中定义。要 对联合查询结果排序时，也必须使用第一查询语句中的列名、列标题或者列序号。在使用UNION 运算符时，应保证每 个联合查询语句的选择列表中有相同数量的表达式，并且每个查询选择表达式应具有相同的数据类型，或是可以自动将它 们转换为相同的数据类型。在自动转换时，对于数值类型，系统将低精度的数据类型转换为高精度的数据类型。

在包括多个查询的UNION语句中，其执行顺序是自左至右，使用括号可以改变这一执行顺序。例如:

查询1 UNION (查询2 UNION 查询3)

#### 三、连接查询

通过连接运算符可以实现多个表查询。连接是关系数据库模型的主要特点，也是它区别于其它类型数据库管理系统

的一个标志。 在关系数据库管理系统中，表建立时各数据之间的关系不必确定，常把一个实体的所有信息存放在一个表 中。当检索数据时，通过连接操作查询出存放在多个表中的不同实体的信息。连接操作给用户带来很大的灵活性，他们可 以在任何时候增加新的数据类型。为不同实体创建新的表，尔后通过连接进行查询。连接可以在SELECT 语句的FROM子句 或WHERE子句中建立，似是而非在FROM子句中指出连接时有助于将连接操作与WHERE子句中的搜索条件区分开来。所以， 在Transact-SQL中推荐使用这种方法。

SQL-92标准所定义的FROM子句的连接语法格式为:

FROM join\_table join\_type join\_table [ON (join\_condition)]

其中join\_table指出参与连接操作的表名，连接可以对同一个表操作，也可以对多表操作，对同一个表操作的连接

又称做自连接。join\_type 指出连接类型，可分为三种:内连接、外连接和交叉连接。内连接(INNER JOIN)使用比较运算符 进行表间某(些)列数据的比较操作，并列出这些表中与连接条件相匹配的数据行。根据所使用的比较方式不同，内连接又

分为等值连接、自然连接和不等连接三种。外连接分为左外连接(LEFT OUTER JOIN或LEFT JOIN)、右外连接(RIGHT

OUTER JOIN或RIGHT JOIN)和全外连接(FULL OUTER JOIN或FULL JOIN)三种。与内连接不同的是，外连接不只列出

与连接条件相匹配的行，而是列出左表(左外连接时)、右表(右外连接时)或两个表(全外连接时)中所有符合搜索条

件的数据行。

交叉连接(CROSS JOIN)没有WHERE 子句，它返回连接表中所有数据行的笛卡尔积，其结果集合中的数据行数等于第一 个表中符合查询条件的数据行数乘以第二个表中符合查询条件的数据行数。连接操作中的ON (join\_condition) 子句指出 连接条件，它由被连接表中的列和比较运算符、逻辑运算符等构成。

无论哪种连接都不能对text、ntext和image数据类型列进行直接连接，但可以对这三种列进行间接连接。例如:

SELECT p1.pub\_id,p2.pub\_id,p1.pr\_info FROM pub\_info AS p1 INNER JOIN pub\_info AS

p2 ON DATALENGTH(p1.pr\_info)=DATALENGTH(p2.pr\_info)

**(一)内连接**

内连接查询操作列出与连接条件匹配的数据行，它使用比较运算符比较被连接列的列值。内连接分三种:

1、等值连接:在连接条件中使用等于号(=)运算符比较被连接列的列值，其查询结果中列出被连接表中的所有列，包括其中 的重复列。

2、不等连接: 在连接条件使用除等于运算符以外的其它比较运算符比较被连接的列的列值。

这些运算符包括> 、>=、 <=、 <、!> 、! <和 <> 。

3、自然连接:在连接条件中使用等于(=)运算符比较被连接列的列值，但它使用选择列表指出查询结果集合中所包括的列， 并删除连接表中的重复列。 例，下面使用等值连接列出authors和publishers表中位于同一城市的作者和出版社:

SELECT \* FROM authors AS a INNER JOIN publishers AS p ON a.city=p.city

又如使用自然连接，在选择列表中删除authors 和publishers 表中重复列(city和state):

SELECT a.\*,p.pub\_id,p.pub\_name,p.country FROM authors AS a INNER JOIN publishers AS

p ON a.city=p.city

**(二)外连接**

内连接时，返回查询结果集合中的仅是符合查询条件( WHERE 搜索条件或 HAVING 条件)和连接条件的行。而采用外连 接时，它返回到查询结果集合中的不仅包含符合连接条件的行，而且还包括左表(左外连接时)、右表(右外连接时)或两 个边接表(全外连接)中的所有数据行。如下面使用左外连接将论坛内容和作者信息连接起来:

SELECT a.\*,b.\* FROM luntan LEFT JOIN usertable as b ON a.username=b.username

下面使用全外连接将city表中的所有作者以及user表中的所有作者，以及他们所在的城市:

SELECT a.\*,b.\* FROM city as a FULL OUTER JOIN user as b ON a.username=b.username

**(三)交叉连接**

交叉连接不带WHERE 子句，它返回被连接的两个表所有数据行的笛卡尔积，返回到结果集合中的数据行数等于第一个 表中符合查询条件的数据行数乘以第二个表中符合查询条件的数据行数。例，titles表中有6类图书，而publishers表 中有8家出版社，则下列交叉连接检索到的记录数将等。于6\*8=48行。

SELECT type,pub\_name FROM titles CROSS JOIN publishers ORDER BY type

## 一些异常精妙的"SQL"语句

◆复制表(只复制结构,源表名：a 新表名：b)

SQL: select \* into b from a where 1<>1

◆拷贝表(拷贝数据,源表名：a 目标表名：b)

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

◆显示文章、提交人和最后回复时间

SQL: select a.title,a.username,b.adddate from table a,

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

◆说明：外连接查询(表名1：a 表名2：b)

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

◆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

◆两张关联表，删除主表中已经在副表中没有的信息

SQL: delete from info where not exists

( select \* from infobz where info.infid=infobz.infid )

## SQL Server数据库标识列注意事项

SQL Server中，经常会用到Identity标识列，这种自增长的字段操作起来比较方便。但它有时还会有点麻烦。

示例一 ：当表中被删除了某些数据的时候，自增长列的编号就不再是一个连线的数列。：

SET IDENTITY\_INSERT [TABLE] [ON|OFF]

允许将显式值插入表的标识列中，当设置为ON时，这时可能在INSERT操作时手工指定插入到标识列中的编号，同时必须在操作完成后，将IDENTITY\_INSERT还原成OFF，否则下次插入的时候必须指定编号，那不然就无法完成INSERT操作。

示例二：当表中的记录被全部删除，但此时标识列的值越来越大的时候，如果不加以重置，它还会无休止的增长。

DBCC CHECKIDENT（TABLE, [RESEED|NORESEED], [1])

将把指定表的种子值强制重设为1。然而，你可能不想将种子重设为1，在这种情况下，你可以用你想用的种子值替代第三个参数。有时候你可能想知道当前的种子，而不是想重设种子,这时你就要用到NORESEED，而不用再去顾忌第三个参数。

## 三种数据库利用 SQL语句进行高效果分页

在程序的开发过程中，处理分页是大家接触比较频繁的事件，因为现在软件基本上都是与数据库进行挂钓的。但效率 又是我们所追求的，如果是像原来那样把所有满足条件的记录全部都选择出来，再去进行分页处理，那么就会多多的浪费 掉许多的系统处理时间。为了能够把效率提高，所以现在我们就只选择我们需要的数据，减少数据库的处理时间，以下就 是常用SQL分页处理：

1、SQL Server、Access数据库

这都微软的数据库，都是一家人，基本的操作都是差不多，常采用如下分页语句：

PAGESIZE：每页显示的记录数 CURRENTPAGE：当前页号

数据表的名字是：components索引主键字是：id

select top PAGESIZE \* from components where id not in

(select top (PAGESIZE\*(CURRENTPAGE-1))id from components order by id)order by id

如下列：

select top 10 \* from components where id not in

(select top 10\*10 id from components order by id) order by id

从101条记录开始选择，只选择前面的10条记录

2、Oracle数据库

因为Oracle数据库没有Top关键字，所以这里就不能够像微软的数据据那样操作，这里有两种方法：

(1)、一种是利用相反的。

select \* from components where id not in(select id from components where

rownum<=(PAGESIZE\*(CURRENTPAGE-1))) and rownum<=PAGESIZE order by id;

如下例：

select \* from components where id not in(select id from components where rownum<=100)

and rownum<=10 order by id;

从101到记录开始选择，选择前面10条。

(2)、使用minus，即中文的意思就是减去。

select \* from components where rownum <=(PAGESIZE\*(CURRENTPAGE-1)) minus

select \* from components where rownum <=(PAGESIZE\*(CURRENTPAGE-2));

如例：select \* from components where rownum<=10 minus select \* from components where rownum<=5;.

(3)、一种是利用Oracle的rownum，这个是Oracle查询自动返回的序号，一般不显示，但是可以通过select rownum from [表名]看到，注意，它是从1到当前的记录总数。

select \* from (select rownum tid,components.\* from components where rownum<=100) where tid<=10;

## 数据库三大范式五大约束

#### 三大范式

**第一范式（1NF）：**

数据表中的每一列(字段)，必须是不可拆分的最小单元，也就是确保每一列的原子性。满足第一范式是关 系模式规范化的最低要求

两列的属性相近或相似或一样，尽量合并属性一样的列，确保不产生冗余数据。

**第二范式（2NF）：**

满足1NF后要求表中的所有列，每一行的数据只能与其中一列相关，即一行数据只做一件事。

**第三范式（3NF）：**

满足2NF后，要求：表中的每一列都要与主键直接相关，而不是间接相关（表中的每一列只能依赖于主键）。 数据不能存在传递关系，即没个属性都跟主键有直接关系而不是间接关系。

1. 第二范式与第三范式的本质区别：在于有没有分出两张表。

第二范式是说一张表中包含了多种不同实体的属性，那么必须要分成多张表，第三范式是要求已经分好了多张表的话，一张表中只能有另一张标的ID，而不能有其他任何信息，（其他任何信息，一律用主键在另一张表中查询）。

2.必须先满足第一范式才能满足第二范式，必须同时满足第一第二范式才能满足第三范式。

三大范式只是一般设计数据库的基本理念，可以建立冗余较小、结构合理的数据库。如果有特殊情况，当 然要特殊对待，数据库设计最重要的是看需求跟性能，需求>性能>表结构。所以不能一味的去追求范式建立数 据库。

#### 数据库中的五大约束包括：

1.主键约束（Primay Key Coustraint） 唯一性，非空性；

2.唯一约束 （Unique Counstraint）唯一性，可以空，但只能有一个；

3.默认约束 (Default Counstraint) 该数据的默认值；

4.外键约束 (Foreign Key Counstraint) 需要建立两表间的关系；

5.非空约束（Not Null Counstraint）:设置非空约束，该字段不能为空。

## 有关 SQL语句的优化技术

操作符优化

◆IN 操作符

用IN写出来的SQL的优点是比较容易写及清晰易懂，这比较适合现代软件开发的风格。但是用IN的SQL性能总 是比较低的，从ORACLE执行的步骤来分析用IN的SQL与不用IN的SQL有以下区别：ORACLE试图将其转换成多个表的 连接，如果转换不成功则先执行IN里面的子查询，再查询外层的表记录，如果转换成功则直接采用多个表的连接方式 查询。由此可见用IN的SQL至少多了一个转换的过程。一般的SQL都可以转换成功，但对于含有分组统计等方面的 SQL就不能转换了。

推荐方案：在业务密集的SQL当中尽量不采用IN操作符。

◆NOT IN操作符

此操作是强列推荐不使用的，因为它不能应用表的索引。

推荐方案：用NOT EXISTS 或（外连接+判断为空）方案代替

◆<> 操作符（不等于）

不等于操作符是永远不会用到索引的，因此对它的处理只会产生全表扫描。

推荐方案：用其它相同功能的操作运算代替，如 a<>0 改为 a>0 or a<0 a<>’’ 改为 a>’’

◆IS NULL 或IS NOT NULL操作（判断字段是否为空）

判断字段是否为空一般是不会应用索引的，因为B树索引是不索引空值的。

推荐方案：用其它相同功能的操作运算代替，如a is not null 改为 a>0 或a>’’等。

不允许字段为空，而用一个缺省值代替空值，如业扩申请中状态字段不允许为空，缺省为申请。建立位图索引（ 有分区的表不能建，位图索引比较难控制，如字段值太多索引会使性能下降，多人更新操作会增加数据块锁的现象）

◆> 及 < 操作符（大于或小于操作符）

大于或小于操作符一般情况下是不用调整的，因为它有索引就会采用索引查找，但有的情况下可以对它进行优 化，如一个表有100万记录，一个数值型字段A，30万记录的A=0，30万记录的A=1，39万记录的A=2，1万记录的 A=3。那么执行A>2与A>=3的效果就有很大的区别了，因为A>2时ORACLE会先找出为2的记录索引再进行比较，而 A>=3时ORACLE则直接找到=3的记录索引。

◆LIKE操作符

LIKE操作符可以应用通配符查询，里面的通配符组合可能达到几乎是任意的查询，但是如果用得不好则会产生性 能上的问题，如LIKE ‘%5400%’ 这种查询不会引用索引，而LIKE ‘X5400%’则会引用范围索引。

◆UNION操作符

UNION在进行表链接后会筛选掉重复的记录，所以在表链接后会对所产生的结果集进行排序运算，删除重复的记录

再返回结果。实际大部分应用中是不会产生重复的记录，最常见的是过程表与历史表UNION。如：

select \* from gc\_dfys union select \* from ls\_jg\_dfys

这个SQL在运行时先取出两个表的结果，再用排序空间进行排序删除重复的记录，最后返回结果集，如果表数据 量大的话可能会导致用磁盘进行排序。

推荐方案：采用UNION ALL操作符替代UNION，因为UNION ALL操作只是简单的将两个结果合并后就返回。

select \* from gc\_dfys union all select \* from ls\_jg\_dfys

◆SQL书写的影响

同一功能同一性能不同写法SQL的影响 如一个SQL在

A程序员写的为 Select \* from zl\_yhjbqk

B程序员写的为 Select \* from dlyx.zl\_yhjbqk（带表所有者的前缀）

C程序员写的为 Select \* from DLYX.ZLYHJBQK（大写表名）

D程序员写的为 Select \* from DLYX.ZLYHJBQK（中间多了空格）

以上四个SQL在ORACLE分析整理之后产生的结果及执行的时间是一样的，但是从ORACLE共享内存SGA的原理， 可以得出ORACLE对每个SQL 都会对其进行一次分析，并且占用共享内存，如果将SQL的字符串及格式写得完全相同则 ORACLE只会分析一次，共享内存也只会留下一次的分析结果，这不仅可以减少分析SQL的时间，而且可以减少共享内 存重复的信息，ORACLE也可以准确统计SQL的执行频率。

◆WHERE后面的条件顺序影响

WHERE子句后面的条件顺序对大数据量表的查询会产生直接的影响，如

Select \* from zl\_yhjbqk where dy\_dj = '1KV以下' and xh\_bz=1

Select \* from zl\_yhjbqk where xh\_bz=1 and dy\_dj = '1KV以下'

以上两个SQL中dy\_dj 及xh\_bz 两个字段都没进行索引，所以执行的时候都是全表扫描，第一条SQL的 dy\_dj = '1KV以下'条件在记录集内比率为99%，而xh\_bz=1的比率只为0.5%，在进行第一条SQL的时候99%条记录都 进行dy\_dj及xh\_bz的比较，而在进行第二条SQL的时候0.5%条记录都进行dy\_dj及xh\_bz的比较，以此可以得出第 二条SQL的CPU占用率明显比第一条低。

◆查询表顺序的影响

在FROM后面的表中的列表顺序会对SQL执行性能影响，在没有索引及ORACLE没有对表进行统计分析的情况下 ORACLE会按表出现的顺序进行链接，由此因为表的顺序不对会产生十分耗服务器资源的数据交叉。

（注：如果对表进行了统计分析，ORACLE会自动先进小表的链接，再进行大表的链接）

◆SQL语句索引的利用 采用函数处理的字段不能利用索引，如：

substr(hbs\_bh,1,4)=’5400’，优化处理：hbs\_bh like ‘5400%’

trunc(sk\_rq)=trunc(sysdate)， 优化处理： sk\_rq>=trunc(sysdate) and sk\_rq<trunc(sysdate+1)

进行了显式或隐式的运算的字段不能进行索引，如：

ss+20>50，优化处理：ss>30

条件内包括了多个本表的字段运算时不能进行索引，如：

qc\_bh||kh\_bh=’5400250000’， 优化处理：qc\_bh=’5400’ and kh\_bh=’250000’

## 一些sql模板

（注释：数据库字典包括表结构、索引和主键.外键.约束.视图.函数.存储过程.触发器。）

--------------------

--行操作 |

--------------------

--增加

insert into 表名[(字段1,字段2...)] values (值1,值2...);

insert into 表名[(字段1,字段2...)] values (值1,值2...), (值1,值2...);

insert into 表名[(字段1,字段2...)] select (字段1,字段2...) from 表名2 [where 条件];

--查询

select distinct \* as name from 表名 where 条件 group by 字段　 having 条件

order by 字段 desc limit 数量

--(关键字执行顺序from-where-group by-having-select-distinct-order by-limit　　)

--函数操作 concat() 连接字符串 concat\_ws() 第一个参数为分隔符,来进行字符串拼接

-- group\_concat()　　 和group by一起使用,分完组后可以显示其他字段数据,用自动用逗号隔开

-- count() 计数 max() 最大值 min() 最小值 avg() 平均值 sum() 求和

select \* from 表名 where name regexp "re";

--改值

update 表名 set 字段1 = 值1, 字段2 = 值2, where 条件;

--删除行

delete from 表名 where 条件;

--删除表中所有行 清空表内容，保留表

truncate table 表名;

--------------------

--列操作 |

--------------------

--查询列

select \* from SYS.COLUMNS WHERE OBJECT\_ID = OBJECT\_ID('tablename') AND NAME = 'columnname'

--新增

alter table tablename add columnname type default val

--重命名

exec sp\_rename 'tablrname.columnname','columnname\_new'

--修改类型

alter table tableName alter column columnName type

--删除列

alter table tableName drop column columnName

--------------------

--表操作 |

--------------------

--创建表

CREATE TABLE table\_name (

field1 datatype,

field2 datatype,

field3 datatype

primary key (field1,field2)

) character set 字符集 collate 校验规则 engine 存储引擎;

--查表

show tables

--表结构

desc tablename

--删除表

drop table [if exists] 表名;

--修改表名称

alter table 表名 rename [to] 新名称;

--------------------

--数据库操作 |

--------------------

--新建库

create database schoolInfo;

--删除库

drop database <数据库名>

--------------------

--数据库连接 |

--------------------

--数据库的最大连接数量配置 0表示无限制

SELECT value\_in\_use FROM sys.configurations c WHERE c.name = 'user connections';

--查询当前的数据库连接数量

select count(distinct(login\_time)) from sys.sysprocesses

--设置数据库最大的连接数 需要重启数据库生效

exec sp\_configure 'show advanced options', 1 --显示数据库高级配置

GO

exec sp\_configure 'user connections', 30

GO

RECONFIGURE WITH OVERRIDE

GO

--查询用户连接数据库最大数量

SELECT @@MAX\_CONNECTIONS

--获取自上次启动 SQL Server服务 以来连接或试图连接的次数

SELECT @@CONNECTIONS

--------------------

--数据库特殊表 |

--------------------

--插入表 inserted

--删除表 deleted

--配置表sys.configurations

--索引

CREATE INDEX index\_name

ON table\_name (column\_name)

--触发器

CREATE TRIGGER trigger\_name

ON table\_name

[WITH ENCRYPTION]

FOR | AFTER | INSTEAD OF [DELETE, INSERT, UPDATE]

AS T-SQL语句

GO

--with encryption 表示加密触发器定义的sql文本

--delete,insert,update指定触发器的类型

--游标

DECLARE 游标名称 CURSOR FOR SELECT 字段1,字段2,字段3,... FROM 表名 WHERE ...  
 OPEN 游标名称  
 FETCH NEXT FROM 游标名称 INTO 变量名1,变量名2,变量名3,...  
 WHILE @@FETCH\_STATUS=0  
        BEGIN  
                  SQL语句执行过程... ...  
                  FETCH NEXT FROM 游标名称 INTO 变量名1,变量名2,变量名3,...  
        END  
 CLOSE 游标名称  
 DEALLOCATE 游标名称 (删除游标)

#### 进制转换

-----二进制转换十进制-----------------

select sum(data1)

from (select substr('1101', rownum, 1) \* power

(2, length('1101') - rownum) data1

from dual

connect by rownum <= length('1101'))

-----八进制转换十进制-----------------

select sum(data1) from (select substr('1101', rownum, 1) \* power(8, length('1101') - rownum) data1

from dual connect by rownum <= length('1101'))

-----十六进制转换十进制-----------------

select sum(data1)from (select (CASE upper(substr('2D', rownum, 1))

WHEN 'A' THEN '10'WHEN 'B' THEN '11'WHEN 'C' THEN '12'WHEN 'D' THEN '13'WHEN 'E' THEN '14'WHEN 'F' THEN '15'

ELSE substr('2D', rownum, 1) END) \* power(16, length('2D') - rownum) data1 from dual connect by rownum <= length('2D'))

#### 定时任务作业

if exists (select \* from dbo.sysobjects where id = object\_id(N'[dbo].[proc\_doc001\_refresh\_applydoc]') and OBJECTPROPERTY(id, N'IsProcedure') = 1)

drop procedure [dbo].[proc\_doc001\_refresh\_applydoc]

GO

create proc proc\_doc001\_refresh\_applydoc

@jobname varchar(100), --作业名称

@sql varchar(8000), --要执行的命令

@serverName sysname='', --job server名

@dbname sysname='', --默认为当前的数据库名

@freqtype varchar(6)='day', --时间周期,month 月,week 周,day 日

@fsinterval int=24, --相对于每日的重复次数

@time int=0 --开始执行时间,对于重复执行的作业,将从0点到23:59分

as

if isnull(@dbname,'')='' set @dbname=db\_name()

--创建作业

exec msdb..sp\_add\_job @job\_name=@jobname

--创建作业步骤

exec msdb..sp\_add\_jobstep @job\_name=@jobname,

@step\_name = '数据处理',

@subsystem = 'TSQL',

@database\_name=@dbname,

@command = @sql,

@retry\_attempts = 5, --重试次数

@retry\_interval = 5 --重试间隔

--创建调度

declare @ftype int,@fstype int,@ffactor int

select @ftype=case @freqtype when 'day' then 4

when 'week' then 8

when 'month' then 16 end

,@fstype=case @fsinterval when 1 then 0 else 8 end

if @fsinterval<>1 set @time=0

set @ffactor=case @freqtype when 'day' then 0 else 1 end

EXEC msdb..sp\_add\_jobschedule @job\_name=@jobname,

@name = '时间安排',

@freq\_type=@ftype , --每天,8 每周,16 每月

@freq\_interval=1, --重复执行次数

@freq\_subday\_type=@fstype, --是否重复执行

@freq\_subday\_interval=@fsinterval, --重复周期

@freq\_recurrence\_factor=@ffactor,

@active\_start\_time=@time --下午17:00:00分执行

if @servername=''

set @servername=@@servername

EXEC msdb..sp\_add\_jobserver @job\_name = @jobname,

@server\_name = @servername

go

--调用

--每天执行的作业

exec proc\_doc001\_refresh\_applydoc @jobname='update\_apply\_doc\_state2'

,@sql='insert into kk\_001 ( PDMCLSNAME, defineclsname) VALUES (''TEST'',''TEST'')'

,@servername=N'10.115.0.161'

,@dbname='2018SP1\_CHANGFA'

,@freqtype='day'

,@time='000000'