基本常用查询

--select

select \* from student;

--all 查询所有

select all sex from student;

--distinct 过滤重复

select distinct sex from student;

--count 统计

select count(\*) from student;

select count(sex) from student;

select count(distinct sex) from student;

--top 取前N条记录

select top 3 \* from student;

--alias column name 列重命名

select id as 编号, name '名称', sex 性别 from student;

--alias table name 表重命名

select id, name, s.id, s.name from student s;

--column 列运算

select (age + id) col from student;

select s.name + '-' + c.name from classes c, student s where s.cid = c.id;

--where 条件

select \* from student where id = 2;

select \* from student where id > 7;

select \* from student where id < 3;

select \* from student where id <> 3;

select \* from student where id >= 3;

select \* from student where id <= 5;

select \* from student where id !> 3;

select \* from student where id !< 5;

--and 并且

select \* from student where id > 2 and sex = 1;

--or 或者

select \* from student where id = 2 or sex = 1;

--between ... and ... 相当于并且

select \* from student where id between 2 and 5;

select \* from student where id not between 2 and 5;

--like 模糊查询

select \* from student where name like '%a%';

select \* from student where name like '%[a][o]%';

select \* from student where name not like '%a%';

select \* from student where name like 'ja%';

select \* from student where name not like '%[j,n]%';

select \* from student where name like '%[j,n,a]%';

select \* from student where name like '%[^ja,as,on]%';

select \* from student where name like '%[ja\_on]%';

--in 子查询

select \* from student where id in (1, 2);

--not in 不在其中

select \* from student where id not in (1, 2);

--is null 是空

select \* from student where age is null;

--is not null 不为空

select \* from student where age is not null;

--order by 排序

select \* from student order by name;

select \* from student order by name desc;

select \* from student order by name asc;

--group by 分组

按照年龄进行分组统计

select count(age), age from student group by age;

按照性别进行分组统计

select count(\*), sex from student group by sex;

按照年龄和性别组合分组统计，并排序

select count(\*), sex from student group by sex, age order by age;

按照性别分组，并且是id大于2的记录最后按照性别排序

select count(\*), sex from student where id > 2 group by sex order by sex;

查询id大于2的数据，并完成运算后的结果进行分组和排序

select count(\*), (sex \* id) new from student where id > 2 group by sex \* id order by sex \* id;

--group by all 所有分组

按照年龄分组，是所有的年龄

select count(\*), age from student group by all age;

--having 分组过滤条件

按照年龄分组，过滤年龄为空的数据，并且统计分组的条数和现实年龄信息

select count(\*), age from student group by age having age is not null;

按照年龄和cid组合分组，过滤条件是cid大于1的记录

select count(\*), cid, sex from student group by cid, sex having cid > 1;

按照年龄分组，过滤条件是分组后的记录条数大于等于2

select count(\*), age from student group by age having count(age) >= 2;

按照cid和性别组合分组，过滤条件是cid大于1，cid的最大值大于2

select count(\*), cid, sex from student group by cid, sex having cid > 1 and max(cid) > 2;

Ø 嵌套子查询

    子查询是一个嵌套在select、insert、update或delete语句或其他子查询中的查询。任何允许使用表达式的地方都可以使用子查询。子查询也称为内部查询或内部选择，而包含子查询的语句也成为外部查询或外部选择。

# from (select … table)示例

将一个table的查询结果当做一个新表进行查询

select \* from (

select id, name from student where sex = 1

) t where t.id > 2;

上面括号中的语句，就是子查询语句（内部查询）。在外面的是外部查询，其中外部查询可以包含以下语句：

     1、 包含常规选择列表组件的常规select查询

     2、 包含一个或多个表或视图名称的常规from语句

     3、 可选的where子句

     4、 可选的group by子句

     5、 可选的having子句

# 示例

查询班级信息，统计班级学生人生

select \*, (select count(\*) from student where cid = classes.id) as num

from classes order by num;

# in, not in子句查询示例

查询班级id大于小于的这些班级的学生信息

select \* from student where cid in (

select id from classes where id > 2 and id < 4

);

查询不是班的学生信息

select \* from student where cid not in (

select id from classes where name = '2班'

)

in、not in 后面的子句返回的结果必须是一列，这一列的结果将会作为查询条件对应前面的条件。如cid对应子句的id；

# exists和not exists子句查询示例

查询存在班级id为的学生信息

select \* from student where exists (

select \* from classes where id = student.cid and id = 3

);

查询没有分配班级的学生信息

select \* from student where not exists (

select \* from classes where id = student.cid

);

exists和not exists查询需要内部查询和外部查询进行一个关联的条件，如果没有这个条件将是查询到的所有信息。如：id等于student.id；

# some、any、all子句查询示例

查询班级的学生年龄大于班级的学生的年龄的信息

select \* from student where cid = 5 and age > all (

select age from student where cid = 3

);

select \* from student where cid = 5 and age > any (

select age from student where cid = 3

);

select \* from student where cid = 5 and age > some (

select age from student where cid = 3

);

Ø 聚合查询

1、 distinct去掉重复数据

select distinct sex from student;

select count(sex), count(distinct sex) from student;

2、 compute和compute by汇总查询

对年龄大于的进行汇总

select age from student

where age > 20 order by age compute sum(age) by age;

对年龄大于的按照性别进行分组汇总年龄信息

select id, sex, age from student

where age > 20 order by sex, age compute sum(age) by sex;

按照年龄分组汇总

select age from student

where age > 20 order by age, id compute sum(age);

按照年龄分组，年龄汇总，id找最大值

select id, age from student

where age > 20 order by age compute sum(age), max(id);

compute进行汇总前面是查询的结果，后面一条结果集就是汇总的信息。compute子句中可以添加多个汇总表达式，可以添加的信息如下：

     a、 可选by关键字。它是每一列计算指定的行聚合

     b、 行聚合函数名称。包括sum、avg、min、max、count等

     c、 要对其执行聚合函数的列

     compute by适合做先分组后汇总的业务。compute by后面的列一定要是order by中出现的列。

3、 cube汇总

cube汇总和compute效果类似，但语法较简洁，而且返回的是一个结果集。

select count(\*), sex from student group by sex with cube;

select count(\*), age, sum(age) from student where age is not null group by age with cube;

cube要结合group by语句完成分组汇总

Ø 排序函数

   排序在很多地方需要用到，需要对查询结果进行排序并且给出序号。比如：

   1、 对某张表进行排序，序号需要递增不重复的

   2、 对学生的成绩进行排序，得出名次，名次可以并列，但名次的序号是连续递增的

   3、 在某些排序的情况下，需要跳空序号，虽然是并列

基本语法

排序函数 over([分组语句] 排序子句[desc][asc])

排序子句 order by 列名, 列名

分组子句 partition by 分组列, 分组列

# row\_number函数

根据排序子句给出递增连续序号

按照名称排序的顺序递增

select s.id, s.name, cid, c.name, row\_number() over(order by c.name) as number

from student s, classes c where cid = c.id;

# rank函数函数

根据排序子句给出递增的序号，但是存在并列并且跳空

顺序递增

select id, name, rank() over(order by cid) as rank from student;

跳过相同递增

select s.id, s.name, cid, c.name, rank() over(order by c.name) as rank

from student s, classes c where cid = c.id;

# dense\_rank函数

根据排序子句给出递增的序号，但是存在并列不跳空

不跳过，直接递增

select s.id, s.name, cid, c.name, dense\_rank() over(order by c.name) as dense

from student s, classes c where cid = c.id;

# partition by分组子句

可以完成对分组的数据进行增加排序，partition by可以与以上三个函数联合使用。

利用partition by按照班级名称分组，学生id排序

select s.id, s.name, cid, c.name, row\_number() over(partition by c.name order by s.id) as rank

from student s, classes c where cid = c.id;

select s.id, s.name, cid, c.name, rank() over(partition by c.name order by s.id) as rank

from student s, classes c where cid = c.id;

select s.id, s.name, cid, c.name, dense\_rank() over(partition by c.name order by s.id) as rank

from student s, classes c where cid = c.id;

# ntile平均排序函数

将要排序的数据进行平分，然后按照等分排序。ntile中的参数代表分成多少等分。

select s.id, s.name, cid, c.name,

ntile(5) over(order by c.name) as ntile

from student s, classes c where cid = c.id;

Ø 集合运算

操作两组查询结果，进行交集、并集、减集运算

1、 union和union all进行并集运算

--union 并集、不重复

select id, name from student where name like 'ja%'

union

select id, name from student where id = 4;

--并集、重复

select \* from student where name like 'ja%'

union all

select \* from student;

2、 intersect进行交集运算

--交集（相同部分）

select \* from student where name like 'ja%'

intersect

select \* from student;

3、 except进行减集运算

--减集（除相同部分）

select \* from student where name like 'ja%'

except

select \* from student where name like 'jas%';

Ø 公式表表达式

查询表的时候，有时候中间表需要重复使用，这些子查询被重复查询调用，不但效率低，而且可读性低，不利于理解。那么公式表表达式可以解决这个问题。

我们可以将公式表表达式（CET）视为临时结果集，在select、insert、update、delete或是create view语句的执行范围内进行定义。

--表达式

with statNum(id, num) as

(

select cid, count(\*)

from student

where id > 0

group by cid

)

select id, num from statNum order by id;

with statNum(id, num) as

(

select cid, count(\*)

from student

where id > 0

group by cid

)

select max(id), avg(num) from statNum;

Ø 连接查询

1、 简化连接查询

--简化联接查询

select s.id, s.name, c.id, c.name from student s, classes c where s.cid = c.id;

2、 left join左连接

--左连接

select s.id, s.name, c.id, c.name from student s left join classes c on s.cid = c.id;

3、 right join右连接

--右连接

select s.id, s.name, c.id, c.name from student s right join classes c on s.cid = c.id;

4、 inner join内连接

--内连接

select s.id, s.name, c.id, c.name from student s inner join classes c on s.cid = c.id;

--inner可以省略

select s.id, s.name, c.id, c.name from student s join classes c on s.cid = c.id;

5、 cross join交叉连接

--交叉联接查询，结果是一个笛卡儿乘积

select s.id, s.name, c.id, c.name from student s cross join classes c

--where s.cid = c.id;

6、 自连接（同一张表进行连接查询）

--自连接

select distinct s.\* from student s, student s1 where s.id <> s1.id and s.sex = s1.sex;

Ø 函数

1、 聚合函数

max最大值、min最小值、count统计、avg平均值、sum求和、var求方差

select

max(age) max\_age,

min(age) min\_age,

count(age) count\_age,

avg(age) avg\_age,

sum(age) sum\_age,

var(age) var\_age

from student;

2、 日期时间函数

select dateAdd(day, 3, getDate());--加天

select dateAdd(year, 3, getDate());--加年

select dateAdd(hour, 3, getDate());--加小时

--返回跨两个指定日期的日期边界数和时间边界数

select dateDiff(day, '2011-06-20', getDate());

--相差秒数

select dateDiff(second, '2011-06-22 11:00:00', getDate());

--相差小时数

select dateDiff(hour, '2011-06-22 10:00:00', getDate());

select dateName(month, getDate());--当前月份

select dateName(minute, getDate());--当前分钟

select dateName(weekday, getDate());--当前星期

select datePart(month, getDate());--当前月份

select datePart(weekday, getDate());--当前星期

select datePart(second, getDate());--当前秒数

select day(getDate());--返回当前日期天数

select day('2011-06-30');--返回当前日期天数

select month(getDate());--返回当前日期月份

select month('2011-11-10');

select year(getDate());--返回当前日期年份

select year('2010-11-10');

select getDate();--当前系统日期

select getUTCDate();--utc日期

3、 数学函数

select pi();--PI函数

select rand(100), rand(50), rand(), rand();--随机数

select round(rand(), 3), round(rand(100), 5);--精确小数位

--精确位数，负数表示小数点前

select round(123.456, 2), round(254.124, -2);

select round(123.4567, 1, 2);

4、 元数据

select col\_name(object\_id('student'), 1);--返回列名

select col\_name(object\_id('student'), 2);

--该列数据类型长度

select col\_length('student', col\_name(object\_id('student'), 2));

--该列数据类型长度

select col\_length('student', col\_name(object\_id('student'), 1));

--返回类型名称、类型id

select type\_name(type\_id('varchar')), type\_id('varchar');

--返回列类型长度

select columnProperty(object\_id('student'), 'name', 'PRECISION');

--返回列所在索引位置

select columnProperty(object\_id('student'), 'sex', 'ColumnId');

5、 字符串函数

select ascii('a');--字符转换ascii值

select ascii('A');

select char(97);--ascii值转换字符

select char(65);

select nchar(65);

select nchar(45231);

select nchar(32993);--unicode转换字符

select unicode('A'), unicode('中');--返回unicode编码值

select soundex('hello'), soundex('world'), soundex('word');

select patindex('%a', 'ta'), patindex('%ac%', 'jack'), patindex('dex%', 'dexjack');--匹配字符索引

select 'a' + space(2) + 'b', 'c' + space(5) + 'd';--输出空格

select charIndex('o', 'hello world');--查找索引

select charIndex('o', 'hello world', 6);--查找索引

select quoteName('abc[]def'), quoteName('123]45');

--精确数字

select str(123.456, 2), str(123.456, 3), str(123.456, 4);

select str(123.456, 9, 2), str(123.456, 9, 3), str(123.456, 6, 1), str(123.456, 9, 6);

select difference('hello', 'helloWorld');--比较字符串相同

select difference('hello', 'world');

select difference('hello', 'llo');

select difference('hello', 'hel');

select difference('hello', 'hello');

select replace('abcedef', 'e', 'E');--替换字符串

select stuff('hello world', 3, 4, 'ABC');--指定位置替换字符串

select replicate('abc#', 3);--重复字符串

select subString('abc', 1, 1), subString('abc', 1, 2), subString('hello Wrold', 7, 5);--截取字符串

select len('abc');--返回长度

select reverse('sqlServer');--反转字符串

select left('leftString', 4);--取左边字符串

select left('leftString', 7);

select right('leftString', 6);--取右边字符串

select right('leftString', 3);

select lower('aBc'), lower('ABC');--小写

select upper('aBc'), upper('abc');--大写

--去掉左边空格

select ltrim(' abc'), ltrim('# abc#'), ltrim(' abc');

--去掉右边空格

select rtrim(' abc '), rtrim('# abc# '), rtrim('abc');

6、 安全函数

select current\_user;

select user;

select user\_id(), user\_id('dbo'), user\_id('public'), user\_id('guest');

select user\_name(), user\_name(1), user\_name(0), user\_name(2);

select session\_user;

select suser\_id('sa');

select suser\_sid(), suser\_sid('sa'), suser\_sid('sysadmin'), suser\_sid('serveradmin');

select is\_member('dbo'), is\_member('public');

select suser\_name(), suser\_name(1), suser\_name(2), suser\_name(3);

select suser\_sname(), suser\_sname(0x01), suser\_sname(0x02), suser\_sname(0x03);

select is\_srvRoleMember('sysadmin'), is\_srvRoleMember('serveradmin');

select permissions(object\_id('student'));

select system\_user;

select schema\_id(), schema\_id('dbo'), schema\_id('guest');

select schema\_name(), schema\_name(1), schema\_name(2), schema\_name(3);

7、 系统函数

select app\_name();--当前会话的应用程序名称

select cast(2011 as datetime), cast('10' as money), cast('0' as varbinary);--类型转换

select convert(datetime, '2011');--类型转换

select coalesce(null, 'a'), coalesce('123', 'a');--返回其参数中第一个非空表达式

select collationProperty('Traditional\_Spanish\_CS\_AS\_KS\_WS', 'CodePage');

select current\_timestamp;--当前时间戳

select current\_user;

select isDate(getDate()), isDate('abc'), isNumeric(1), isNumeric('a');

select dataLength('abc');

select host\_id();

select host\_name();

select db\_name();

select ident\_current('student'), ident\_current('classes');--返回主键id的最大值

select ident\_incr('student'), ident\_incr('classes');--id的增量值

select ident\_seed('student'), ident\_seed('classes');

select @@identity;--最后一次自增的值

select identity(int, 1, 1) as id into tab from student;--将studeng表的烈属，以/1自增形式创建一个tab

select \* from tab;

select @@rowcount;--影响行数

select @@cursor\_rows;--返回连接上打开的游标的当前限定行的数目

select @@error;--T-SQL的错误号

select @@procid;

8、 配置函数

set datefirst 7;--设置每周的第一天，表示周日

select @@datefirst as '星期的第一天', datepart(dw, getDate()) AS '今天是星期';

select @@dbts;--返回当前数据库唯一时间戳

set language 'Italian';

select @@langId as 'Language ID';--返回语言id

select @@language as 'Language Name';--返回当前语言名称

select @@lock\_timeout;--返回当前会话的当前锁定超时设置（毫秒）

select @@max\_connections;--返回SQL Server 实例允许同时进行的最大用户连接数

select @@MAX\_PRECISION AS 'Max Precision';--返回decimal 和numeric 数据类型所用的精度级别

select @@SERVERNAME;--SQL Server 的本地服务器的名称

select @@SERVICENAME;--服务名

select @@SPID;--当前会话进程id

select @@textSize;

select @@version;--当前数据库版本信息

9、 系统统计函数

select @@CONNECTIONS;--连接数

select @@PACK\_RECEIVED;

select @@CPU\_BUSY;

select @@PACK\_SENT;

select @@TIMETICKS;

select @@IDLE;

select @@TOTAL\_ERRORS;

select @@IO\_BUSY;

select @@TOTAL\_READ;--读取磁盘次数

select @@PACKET\_ERRORS;--发生的网络数据包错误数

select @@TOTAL\_WRITE;--sqlserver执行的磁盘写入次数

select patIndex('%soft%', 'microsoft SqlServer');

select patIndex('soft%', 'software SqlServer');

select patIndex('%soft', 'SqlServer microsoft');

select patIndex('%so\_gr%', 'Jsonisprogram');

10、 用户自定义函数

# 查看当前数据库所有函数

--查询所有已创建函数

select definition,\* from sys.sql\_modules m join sys.objects o on m.object\_id = o.object\_id

and type in('fn', 'if', 'tf');

# 创建函数

if (object\_id('fun\_add', 'fn') is not null)

drop function fun\_add

go

create function fun\_add(@num1 int, @num2 int)

returns int

with execute as caller

as

begin

declare @result int;

if (@num1 is null)

set @num1 = 0;

if (@num2 is null)

set @num2 = 0;

set @result = @num1 + @num2;

return @result;

end

go

调用函数

select dbo.fun\_add(id, age) from student;

--自定义函数，字符串连接

if (object\_id('fun\_append', 'fn') is not null)

drop function fun\_append

go

create function fun\_append(@args nvarchar(1024), @args2 nvarchar(1024))

returns nvarchar(2048)

as

begin

return @args + @args2;

end

go

select dbo.fun\_append(name, 'abc') from student;

# 修改函数

alter function fun\_append(@args nvarchar(1024), @args2 nvarchar(1024))

returns nvarchar(1024)

as

begin

declare @result varchar(1024);

--coalesce返回第一个不为null的值

set @args = coalesce(@args, '');

set @args2 = coalesce(@args2, '');;

set @result = @args + @args2;

return @result;

end

go

select dbo.fun\_append(name, '#abc') from student;

# 返回table类型函数

--返回table对象函数

select name, object\_id, type from sys.objects where type in ('fn', 'if', 'tf') or type like '%f%';

if (exists (select \* from sys.objects where type in ('fn', 'if', 'tf') and name = 'fun\_find\_stuRecord'))

drop function fun\_find\_stuRecord

go

create function fun\_find\_stuRecord(@id int)

returns table

as

return (select \* from student where id = @id);

go

select \* from dbo.fun\_find\_stuRecord(2);

http://www.cnblogs.com/hoojo/archive/2011/07/16/2108129.html