数据查询操作

一、表达式

1、定义

操作数与操作符组合的式子。

- 操作数：可以是常量、变量、函数的返回值、另一个查询语句返回的值；
- 操作符：就是运算符，包括算术运算符、赋值运算符、比较运算符、逻辑运算符、字符匹配运算等。

2、运算符

1）算术运算符

符号	表达式	作用
+	x1 + x2	加
-	x1 - x2	减
*	x1 * x2	乘
/	x1 / x2	除
DIV	x1 DIV x2	除
%	x1 % x2	求余
MOD	x1 MOD x2	求余

```
select 1+1,1/2;
```

2）赋值运算符

=（比较运算符也是=）

3）比较运算符

符号	表达式	作用
=	x1 = x2	判断是否相等，不能判断等于NULL
<> 或 !=	x1<>x2 或 x1!=x2	判断是否不相等
<=>	x1<=>x2	判断是否相等，可以判断是否等于NULL
>、>=	x1>x2、x1>=x2	判断是否大于、大于等于
<、<=	x1<x2、x1<=x2	判断是否小于、小于等于
IS NULL、IS NOT NULL	x1 IS NULL、x1 IS NOT NULL	判断是否为NULL、或不为NULL
BETWEEN AND、NOT BETWEEN AND	X1 BETWEEN m AND n	判断是否在m与n范围内
IN或NOT IN	x1 IN (值1,...,值n)	判断是否在某个固定范围中
LIKE、NOT LIKE	x1 LIKE 表达式	判断是否匹配（模拟匹配、结合通配符）
REGEXP	x1 REGEXP 正则	判断是否正则匹配

注意：运算结果只能为真（1）或假（0）

```
#匹配username是否以字符t开始
select username,username REGEXP '^t' from student;

#区别=与<=>
select username,address=null from student;
select username,address<=>null from student;
```

4）逻辑运算符

符号	形式	作用
&&、AND	与	并且
||、OR	或	或者
!、NOT	非	取反
XOR	异或	不同为真

5）字符匹配运算

- BETWEEN...AND：如果操作数在某个范围之内，那么就为 TRUE
- IN：如果操作数等于表达式列表中的一个，那么就为 TRUE
- like：如果操作数与一种模式相匹配，那么就为 TRUE
    - %：零个或任意个字符
    - _：任意一个字符；
    - []：指定一个字符、字符串或范围，从中选择一个匹配对象；
    - [^]：所匹配的字符为指定字符以外的一个字符

6）优先级

优先级	运算符	优先级	运算符
1	!	8	|
2	~	9	=（比较）、<=>、<、<=、>=、!=、<>、IN、IS NULL、LIKE、REGEXP
3	^	10	BETWEEN AND、CASE、WHEN、THEN、ELSE
4	*、/、DIV、%、MOD	11	NOT
5	+、-	12	&&、AND
6	>>、<<	13	||、OR、XOR
7	&	14	=（赋值）、:=

注意：可以通过（）提高优先级

二、基本查询

1、基本语法

```
select [distinct] [顶部条数| 聚合函数] 字段集合|常量|表达式|函数|*
from  表名
[where 查询条件语句集合]
[group by 分组字段列表]
[having 过滤条件语句集合] 分组查询条件
[order by 排序字段集合 [asc | desc]]
[LIMIT [<offset>,] <row count>]

# 执行顺序: FROM -> WHERE -> GROUP BY -> SELECT -> ORDER BY -> LIMIT
```

2、查询全部字段数据（*）

```
select * from 表名 ;
```

3、局部字段查询

```
select 字段1,...,字段n from 表名 ;
```

4、前N条数

```
# 在MySQL中，top N无效，使用 limit 实现
select top N | top n percent ... from 表名 ;
select * from  表名 limit N


select * from  表名 limit [offset,] rows

# 分页查询
select * from  表名 limit (当前页-1) * 每页记录数,每页记录数
```

- offset：起始值，默认从0开始
- rows：查询记录数

5、条件查询

```
select 字段集合 | * from 表名 where 条件 ;
```

查询条件可以是：

- 带比较运算符和逻辑运算符的查询条件
- 带 BETWEEN AND 关键字的查询条件
- 带 IS NULL 关键字的查询条件

- 带 IN、NOT IN 关键字的查询条件
- 带 LIKE 关键字的查询条件

6、别名（as）

```
# 为字段指定别名
select 字段 [as] 别名   ...

# 为表名指定别名
select *|字段集合 from 表名 as 别名
```

7、字符连接（+）

```
# 注：在MySQL中，+表示的是运算符，不能用于字符串连接
select 1 + 1;

# 把字符尝试转换为数值，如果转换成功，则得到对应的数值，如果转换失败，则为0
select 1 + '1' ;
select 1 + '你' ;

# null + 任何数据 = null
select 1 + null ;

# 实现字符连接，可以使用concat函数
```

8、排序查询

```
# 单字段排序,默认为ASC
order by 字段 ASC | DESC

# 多字段排序
# 注意：在对多个字段进行排序时，排序的第一个字段必须有相同的值，才会对第二个字段进行排序。如果第一个字段数据中所有的值都是唯一的，MySQL 将不再对第二个字段进行排序。
order by 字段1 ASC | DESC,字段2 ASC | DESC
```

9、聚合函数

- sum() : 求和
- avg() : 平均值
- max() : 最大值
- min() : 最小值
- count() : 记录数

注：

- 也叫分组函数，一般用于分组查询
- 以上五个分组函数都忽略null值，除了count(*)；
- sum和avg一般用于处理数值型，max、min、count可以处理任何数据类型；
- 如果count字段，则字段值为NULL的情况，将忽略统计

10、筛选重复数据（去重）

```
select distinct *|字段集合 from ...
```

注意：如果DISTINCT 关键字后有多个字段，则会对多个字段进行组合去重，也就是说，只有多个字段组合起来完全是一样的情况下才会被去重。

11、分组查询

```
select 字段，聚合函数
from 表名
group by 字段
【WITH ROLLUP】
having 条件
```

- select后面跟着的字段必须满足两个条件，要么分组字段、要么使用聚合函数；
- 分组查询一般结合聚合函数一起使用；
- where是分组前条件，having是分组后条件。

```
SELECT 性别,COUNT(*) FROM `学生信息`
GROUP BY 性别
WITH ROLLUP;

# 解决NULL输出：coalesce(arg1,arg2,arg3,...)
# 如果arg1非空，取值arg1，否则取值arg2;继续判断arg2是否为空，依此类推。如果参数都为NULL，则返回NULL
SELECT coalesce(NULL,100) ;

SELECT coalesce(性别,'总人数'),COUNT(*) FROM `学生信息`
GROUP BY 性别
WITH ROLLUP;
```

12、模糊查询

```
... where 字段 like  值 ...
```

其中，值一般结合以下三个通配符使用

- _（下划线）：表示任意的1个字符（长度不能为0）
- %：表示任意的0个或多个字符
- []：表示某范围（不支持？）
- [^]：表示某范围以外

13、正则查询

```
... where 字段 regexp  正则表达式 ...
```

14、CASE WHEN

```
CREATE TABLE test_user
(
    id int primary key auto_increment ,
    name varchar(50) not null ,
    gender tinyint default 1 ,
    country_code smallint
) engine=InnoDb default charset=utf8;

insert into test_user(name,gender,country_code) values ('清风',1,100) ;
insert into test_user(name,gender,country_code) values ('玄武',2,100) ;
insert into test_user(name,gender,country_code) values ('Kobe',1,110) ;
insert into test_user(name,gender,country_code) values ('John Snow',1,200) ;



select id,name,gender,
  (
    case gender
      when 1 then '男'
      when 2 then '女'
      else '未知'
    end
  ) 性别,
  country_code
from test_user;
```

15、行转列

```
create table grade
(
    id int(10) not null auto_increment ,
    user_name varchar(20),
    course varchar(20),
    score float ,
    primary key(id)
) engine=InnoDb default charset=utf8 ;

insert into grade(user_name,course,score) values ('张三','数学',34) ;
insert into grade(user_name,course,score) values ('张三','语文',58) ;
insert into grade(user_name,course,score) values ('张三','英语',58) ;
insert into grade(user_name,course,score) values ('李四','数学',45) ;
insert into grade(user_name,course,score) values ('李四','语文',87) ;
insert into grade(user_name,course,score) values ('李四','英语',45) ;
insert into grade(user_name,course,score) values ('王五','数学',76) ;
insert into grade(user_name,course,score) values ('王五','语文',34) ;
insert into grade(user_name,course,score) values ('王五','英语',89) ;

SELECT user_name,
  max(case course when '数学' then score else 0 end) 数学,
  max(case course when '语文' then score else 0 end) 语文,
  max(case course when '英语' then score else 0 end) 英语
from grade GROUP BY user_name ;
```

三、多表查询

> 如果我们要查询的数据分布在不同的表时，那么需要连接多张表进行多表查询。

1、等值查询

```
select 字段集合 from 表1,表2,...,表n
where 条件 (主外键)
```

注意：

- 条件一般是主键和外键的关联（可能包含筛选数据的条件）；
- 一般给各表取别名，提高阅读性、性能以及解决字段冲突

```
create table 教师表
(
  编号 int identity(1,1) primary key not null ,
  姓名 char(30) ,
  性别 char(2) check(性别='男' or 性别='女') default '男',
  专业 char(30)
)


create table 学生表
(
  学号 int identity(1,1) primary key not null ,
  姓名 char(30) ,
  性别 char(2) check(性别='男' or 性别='女') default '男',
  身高 float ,
  学分 float ,

  教师编号 int foreign key references 教师表(编号)
)


insert into 教师表 values ('张三','男','计算机')
insert into 教师表 values ('李四','男','日语')
insert into 教师表 values ('王五','女','英语')


insert into 学生表 values ('学生一','男',1.5,50,1)
insert into 学生表 values ('学生二','女',2.5,60,1)
insert into 学生表 values ('学生三','男',3.5,70,2)
insert into 学生表 values ('学生四','女',4.5,80,2)

select * from 教师表
select * from 学生表
```

2、内连接

```
-- 功能同等值连接
-- 好处：连接条件与筛选条件分离，简洁明了
select 字段集合 from 表1 [inner] join 表2 on 条件(主外键|相同数据类型)
where 条件
```

3、外连接

3.1、左外连接

以第一张表为基础向第二张表匹配，匹配成功则正常显示，匹配不成功，则第二张表以NULL值显示

```
select 字段集合 from 表1 left [outer] join 表2 on 条件
```

3.2、右外连接

以第二张表为基础向第一张表匹配，匹配成功则正常显示，匹配不成功，则第一张表以NULL值显示

```
select 字段集合 from 表1 right [outer] join 表2 on 条件
```

3.3、完全外连接

左、右表的数据都要显示，如果能连接，则正常显示;如果不能连接，则以NULL值显示

```
# select 字段集合 from 表1 full [outer] join 表2 on 条件
# MySQL不支持，可以使用UNION实现

SELECT * from 表1 LEFT JOIN 表2 on 条件;
UNION
SELECT * from 表1 RIGHT JOIN 表2 on 条件;
```

4、交叉连接

```
-- 语法一
select 字段集合 from 表1 [inner] join 表2 #sql server 使用关键字 cross

-- 语法二
select 字段集合 from 表1,表2
```

5、自连接

```
-- 语法一
select * from 表1 as 别名 join 表1 as 别名 on 条件

-- 语法二
select * from 表1 as 别名, 表1 as 别名 where 条件
```

6、联合查询

把多个查询的结果合并在一起

```
SELECT f1,f2 FROM T_1
UNION [ALL]
SELECT f1,f2 FROM T_2
```

注意:

- 两个查询的字段个数必须相同;
- 字段类型也要相同或兼容;
- union代表去重，union all代表不去重

四、子查询

1、概念

一条查询语句中被嵌套到另一条完整的查询语句中，其中被嵌套的查询语句，称之为子查询或内查询，而在外面的查询语句，称为主查询或外查询。用于子查询的关键字主要包括 in、not in、=、!=、exists、not exists 等。

- 子查询都放在小括号内;
- 子查询可以放在from后面、select后面、where后面、having后面，但常用于where后面;
- 子查询优先于主查询执行，子查询所得的结果服务于主查询;
- 子查询根据查询结果的行数不同分为以下两类:
  - 单行子查询：子查询的结果只有一行
    - 一般结合单行操作符使用：>、<、=、 <>、 >=、 <=
  - 多行子查询：子查询的结果有多行
    - 一般结合多行操作符使用：any、all、in、not in

2、三种实现:

2.1、[NOT] IN

```
... where 字段 [NOT] IN (子查询|数值集合)
```

案例

```
create table t1
(
  n int
);
```

```
create table t2
(
  n int
);

insert into t1 values (2);
insert into t1 values (3);

insert into t2 values (1);
insert into t2 values (2);
insert into t2 values (3);
insert into t2 values (4);


select * from t2 where n in (1,3);
select * from t2 where n =1 || n=3;

select * from t2 where n not in (1,3);
select * from t2 where !(n = 1 || n = 3);
```

2.2、比较运算符 [any | some | all]

```
... where 字段 比较运算符 [any | some | all] (子查询)
--如:=any,>any,>=all
```

说明

- 其中，ANY与SOME相同，对子查询的结果进行逻辑或运算;而All则对子查询的结果进行逻辑与运算；
- ＝any 或 =some 等同于 in
- =any 或 =some 后面必须要是子查询，不能是具体某些数据；而in语句后面可以是子查询，也可以是具体的某些数据

案例

```
-- all : 逻辑与(同时满足条件)
SELECT * FROM T2 WHERE N>all (SELECT N FROM T1)
SELECT * FROM T2 WHERE N>2 && N>3

-- some、any：逻辑或(满足其中之一)
SELECT * FROM T2 WHERE N>any (SELECT N FROM T1)
SELECT * FROM T2 WHERE N>some (SELECT N FROM T1)
SELECT * FROM T2 WHERE N>2 || N>3

-- =any、=some、in相同
SELECT * FROM T2 WHERE N=SOME (SELECT N FROM T1);
SELECT * FROM T2 WHERE N=ANY (SELECT N FROM T1);
SELECT * FROM T2 WHERE N IN (SELECT N FROM T1);

-- 思考：以下查询的结果是?
```

```
SELECT * FROM T2 WHERE N <> any(SELECT * FROM T1);
```

2.3、[NOT] EXISTS

```
-- 判断子查询是否有查询结果，有返回true，否则返回false
... where [NOT] EXISTS (子查询)
```

3、特殊使用

子查询也可以使用在SELECT、INSERT、UPDATE或DELETE语句中。

五、常用函数

1、字符函数

函数	功能
CONCAT(s1, s2, ..., sn)	字符串拼接，将s1, s2, ..., sn拼接成一个字符串
LOWER(str)	将字符串全部转为小写
UPPER(str)	将字符串全部转为大写
LPAD(str, n, pad)	左填充，用字符串 pad 对 str 最左边进行填充，直到长度为 n 个字符长度
RPAD(str, n, pad)	右填充，用字符串 pad 对 str 最右边进行填充，直到长度为 n 个字符长度
TRIM(str)	去掉字符串头部和尾部的空格
LTRIM(str)	去左边空格
RTRIM(str)	去右边空格
SUBSTRING(str, start, len)	返回从字符串str从start位置起的len个长度的字符串
REPLACE(str,a,b)	用字符串 b 替换字符串 str 中所有出现的字符串 a
INSERT(str,x,y,instr)	将字符串 str 从第 x 位置开始，y 个字符长的子串替换为字符串 instr
LENGTH(str)	获取字节个数
LEFT(str,x)和RIGHT(str,x)	分别返回字符串最左边的x个字符和最右边的x个字符。 如果第二个参数是 NULL，那么将不返回任何字符串

```
# 任何字符串与 NULL 进行连接的结果都将是 NULL
SELECT concat('aaa','bbb','ccc') ,concat('aaa',NULL);

SELECT substring('beijing2008',8,4),substring('beijing2008',1,7);

SELECT INSERT('beijing2008you',12,3, 'me') ;

SELECT LEFT('beijing2008',7),LEFT('beijing',NULL),RIGHT('beijing2008',4);

SELECT lpad('2008',20,'beijing'),rpad('beijing',20,'2008');
```

2、数学函数

函数	功能
CEIL(x)	向上取整，返回大于 x 的最大整数
FLOOR(x)	向下取整，返回小于 x 的最大整数，和 CEIL 的用法刚好相反
MOD(x, y)	返回x % y的模
RAND()	返回0~1内的随机数
ROUND(x, y)	求参数x的四舍五入值，保留y位小数
TRUNCATE(x,y)	返回数字 x 截断为 y 位小数的结果

```
# 如果不写 y，则默认 y 为 0，即将 x 四舍五入后取整
select ROUND(1.1),ROUND(1.1,2),ROUND(1,2);
select RAND(),RAND();

# 模数和被模数任何一个为 NULL 结果都为 NULL
select MOD(15,10),MOD(1,11),MOD(NULL,10);

select CEIL(-0.8),CEIL(0.8);
select FLOOR(-0.8), FLOOR(0.8);

select ROUND(1.235,2),TRUNCATE(1.235,2);
```

3、日期函数

函数	功能
CURDATE()	返回当前系统日期
CURTIME()	返回当前系统时间
NOW()	返回当前系统日期+时间
YEAR(date)	获取指定date的年份
MONTH(date)	获取指定date的月份
DAY(date)	获取指定date的日期
STR_TO_DATE(str,format)	将字符转换成日期
DATE_FORMAT(date,format)	将日期转换成字符
DATE_ADD(date,INTERVAL expr type)	返回一个日期/时间值加上一个时间间隔expr后的时间值。其中 INTERVAL 是间隔类型关键字，expr 是一个表达式，这个表达式对应后面的类型，type 是间隔类型，如：year,month,day等
DATEDIFF(date1，date2)	返回起始时间date1和结束时间date2之间的天数
TIMESTAMPDIFF(type ,datetime_expr1,datetime_expr2)	datetime_expr2与datetime_expr1的时间差，type 是间隔类型，如：year,month,day等

```
select str_to_date('2021年03月09日','%Y年%m月%d日');

select date_format(now(),'%Y年%m月%d日');

# 两天后
select now() 当前时间,date_add(now(),INTERVAL 2 day) ;

select DATEDIFF('2021-05-01',now());

select  TIMESTAMPDIFF(MONTH, '1985-06-24', '1985-08-24') ;
```

4、流程控制函数

函数	功能
IF(value, t, f)	如果value为true，则返回t，否则返回f
IFNULL(value1, value2)	如果value1不为空，返回value1，否则返回value2
CASE WHEN [val1] THEN [res1] ... ELSE [default] END	如果val1为true，返回res1，... 否则返回default默认值
CASE [expr] WHEN [val1] THEN [res1] ... ELSE [default] END	如果expr的值等于val1，返回res1，... 否则返回default默认值

1）if：处理双分支

```
# 如果value是真，返回t；否则返回f
if(value,t,f)


select if(5>1,'你好','我好');
```

2）IFNULL(value1,value2)：如果 value1 不为空返回 value1，否则返回 value2

```
select IFNULL('你好','我好') ;

select IFNULL(NULL,'我好') ;
```

3）case语句：处理多分支

```
# 情况1：处理等值判断
# 如果 expr 等于 value，返回 result，否则返回 default
CASE expr WHEN value THEN result ... ELSE default END


# 情况2：处理条件判断
# 如果 value 是真，返回 result，否则返回 default
CASE WHEN [value] THEN [result] ... ELSE [default] END



# 案例
create table salary
(
  userid int,
  salary decimal(9,2)
);

insert into salary values(1,1000),(2,2000), (3,3000),(4,4000),(5,5000), (1,null);

select userid,salary,if(salary>2000,'high','low') from salary;
```

```
select ifnull(salary,0) from salary;


select case when salary<=2000 then 'low' else 'high' end from salary;
select case salary when 1000 then 'low' when 2000 then 'mid' else 'high' end from salary;
```

5、其他函数

函数	功能
version()	版本
database()	当前库
user()	当前连接用户

```
select version() ;

select database() ;

select user() ;
```