WHERE

SELECT proc\_id FROM customer WHERE cus\_num <> 1003;

查找customer 中 cus\_num 不等于1003的

SELECT proc\_id FROM customer WHERE cus\_num NOT IN （1002,1003;

查找customer 中 cus\_num 不在1002,1003中的

SELECT proc\_id FROM customer WHERE cus\_email IS NULL;

查找customer 中 cus\_email 是空的proc\_id

AND/OR

计算次序：

SELECT

pro\_id , pro\_name

FROM product

WHERE (ven\_id = 1002 OR ven\_id = 1003 ) AND pro\_pay >=1000;

判断顺序是：（） > AND > OR

ORDER BY

ORDER BY pro\_name: 按名字的字母顺序排列

ORDER BY pro\_num ：按数字大小排列

ORDER BY

Like 模糊匹配

SELECT pro\_id ,pro\_name FROM product WHERE ven\_name LIKE ‘jet%’;

SELECT pro\_id ,pro\_name FROM product WHERE ven\_name LIKE ‘% .vent %’;

%:jet后接任意字符，匹配任意多个（0~n） 但JET…不匹配

\_: 只匹配单个字符

注意：使用通配符时，尽量不要全表搜索，应该结合其他的限制一起搜索

正则表达式：

SELECT pro\_id ,pro\_name FROM product WHERE ven\_info regexp ‘1000 ‘;

匹配所有ven\_info 含有1000的

SELECT pro\_id ,pro\_name FROM product WHERE ven\_info regexp ‘.000 ‘;

匹配所有ven\_info 含有X000的，可以有2000,11000,3000

**(like 与 regexp 的区别)**

**Like：匹配的是整个列，如果匹配的文本在列中，like不会找到他**

**Regexp：在列中 进行匹配，被匹配的再列中会找到并返回**

正则匹配不区分大小写

SELECT pro\_id ,pro\_name FROM product WHERE ven\_info regexp ‘[1,2,3]TOM ‘;

SELECT pro\_id ,pro\_name FROM product WHERE ven\_info regexp ‘[1-9]TOM ‘;

SELECT pro\_id ,pro\_name FROM product WHERE ven\_info regexp ‘\\- ‘;

匹配特殊字符前面加\\ 如\\# \\. \\^ \\\

[\\f](file:///\\\\f) 换页

[\\n](file:///\\\\n) 换行

[\\r](file:///\\\\r) 回车

\*匹配0-n个

+匹配1-n 个

？匹配0-1 个

匹配字符类型：

预定义的字符集

[:digit:] = [0-9]

[:lower:] = [a-z]

[0-9]{3}出现连续的三个数字

拼接字段：concatenate 只有mysql需要用其他的东西

Concat()函数

SELECT Concat(vend\_name,’(‘,vend\_city,’)’) FROM venders ORDER BY vend\_name

将vend\_name 与 vend\_city 合并，concat()需要一个或多个指定的串，，各个串之间用,分离

RTrim()删除函数右侧多余的空格

SELECT Concat(RTrim(vend\_name),’(‘,RTrim(vend\_city),’)’)

数据处理函数：

Upper()

Lower()

LTrim()

RTrim()

Right()

Left()

时间处理函数，运算函数

SELECT count(\*) AS num\_proc FROM products WHERE id=1003;计数

GROUP BY 创建分组：

GROUP BY必须在WHERE后ORDER BY 之前

SELECT ven\_id , count(\*) FROM products GROUP BY ven\_id;

以ven\_id来进行分组

1001 7

1002 1

1003 6

1004 2

SELECT ven\_id , count(\*) FROM products GROUP BY ven\_id HAVING count(\*) >=2;

1001 7

1003 6

使用子查询

Select嵌套查询

1.将联合查询转化为两个嵌套的查询

Select cus\_id

FROM order

WHERE order\_num IN (select order\_num

from ordertimes

where pro\_id = ‘TN2

);

2,将一个查询中的结果作为输出的一部分

SELECT pro\_id ,pro\_name ,

(SELECT count(\*) FROM order WHERE cus\_id = ser\_id ) AS order

FROM product

ORDER BY pro\_id

内部链接与外部l链接

内部链接：

Inner join on

SELECT C.\* ,o.order, oi.date

FROM customer AS C, order AS 0,order\_date AS oi

WHERE C.id = ‘FB’ AND oi.num=o.num

C.\*表示第一个被匹配的列，后面相同的列都不会再显示

left join(左联接) 返回包括左表中的所有记录和右表中联结字段相等的记录

right join(右联接) 返回包括右表中的所有记录和左表中联结字段相等的记录

inner join(等值连接) 只返回两个表中联结字段相等的行

select \* from A

inner join B

on A.aID = B.bID

返回A.aID 与 B.bID 都有的行

Left join

返回 A.aID 全部行与B.bID相同数量的行 ，匹配的显示，不足的为NULL

Right join 相反

语法：FROM table1 LEFT JOIN table2 ON table1.field1 compopr table2.field2

说明：table1, table2参数用于指定要将记录组合的表的名称。

field1, field2参数指定被联接的字段的名称。且这些字段必须有相同的数据类型及包含相同类型的数据，但它们不需要有相同的名称。

compopr参数指定关系比较运算符："="， "<"， ">"， "<="， ">=" 或 "<>"。

外部链接

用于一个表中的行与另外表中的行相关，但有时需要部分没有关联的行

SELECT customer, cus\_id order\_num FROM custmers

LEFT OUTER JOIN orders ON

Cus\_id = order\_id;

组合查询：

UNION 联合两个select查询语句，将两个独立的查询结果联合输出

注意：必须是相同的列，表达式，函数，数据类型必须兼容

UNION自动取消重复行

UNION ALL 不取消重复行

全文本搜索：

Match()

AGAINST()

常用的SQL命令：

COMMIT

它把上一个COMMIT或ROLLBACK命令之后的全部事务都保存到数据库。

这个命令的语法是：  commit [work];(可以直接commit就行)

#禁止自动提交  
set autocommit=0;

CREATE INDEX

创建一索引：

|  |  |
| --- | --- |
| 1  2  3 | ALTER TABLE table\_name ADD INDEX index\_name (column\_list)  ALTER TABLE table\_name ADD UNIQUE (column\_list)  ALTER TABLE table\_name ADD PRIMARY KEY (column\_list) |

CREATE INDEX可对表增加普通索引或UNIQUE索引。

|  |  |
| --- | --- |
| 1  2 | CREATE INDEX index\_name ON table\_name (column\_list)  CREATE UNIQUE INDEX index\_name ON table\_name (column\_list) |

可利用ALTER TABLE或DROP INDEX语句来删除索引。类似于CREATE INDEX语句，DROP INDEX可以在ALTER TABLE内部作为一条语句处理，语法如下。

[?](https://www.jb51.net/article/73372.htm)

|  |  |
| --- | --- |
| 1  2  3 | DROP INDEX index\_name ON talbe\_name  ALTER TABLE table\_name DROP INDEX index\_name  ALTER TABLE table\_name DROP PRIMARY KEY |

可利用ALTER TABLE或DROP INDEX语句来删除索引。类似于CREATE INDEX语句，DROP INDEX可以在ALTER TABLE内部作为一条语句处理，语法如下。

|  |  |
| --- | --- |
| 1  2  3 | DROP INDEX index\_name ON talbe\_name  ALTER TABLE table\_name DROP INDEX index\_name  ALTER TABLE table\_name DROP PRIMARY KEY |

5．查看索引

|  |  |
| --- | --- |
| 1  2 | mysql> show index from tblname;  mysql> show keys from tblname; |

CREATE ROLE

Role

Mysql8.0的新特性：

## -------------------------------------------------- **创建role，并赋予role对t1的select权限**

mysql> create role app\_read;

mysql> grant select on DB.t1 to app\_read;

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

## **创建用户，并赋予用户role**

## create user tom@localhost identified by 'qqq123'

grant app\_read to tom@localhost;

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

**删除一个权限：**

revoke insert on t1 from app\_read

**修改role权限，也需要重新登录，先exit退出**

grant insert on t1 to app\_read;

**查看role**

show tables like '%role%'

限[系统](http://www.2cto.com/os/" \t "/root/文档\\x/_blank)分为系统权限与对象权限。系统权限( database system privilege )可以让用户执行特定的命令集。例如，create table权限允许用户创建表，grant any privilege 权限允许用户授予任何系统权限。对象权限( database object privilege )可以让用户能够对各个对象进行某些操作。例如delete权限允许用户删除表或视图的行，select权限允许用户通过select从表、视图、序列(sequences)或快照(snapshots)中查询信息。

CREATE TABLE（AS）

创建克隆表actor\_as

create table actor\_as as select \* from actor;

查看表索引>show index from actor\_as \G

查看表信息>show table status like 'actor\_as'\G

查看表结构>desc actor\_as

a、create table like方式会完整地克隆表结构，但不会插入数据，需要单独使用insert into或load data方式加载数据  
b、create table as  方式会部分克隆表结构，完整保留数据  
c、create table as select .. where 1=0 会克隆部分表结构，但不克隆数据。

CREATE VIEW

CREATE USER

创建一个用户：

create user test identified by '123456';

授权用户：

grant all privileges on \*.\* to 'test'@'%'identified by '123456' with grant option;

修改指定用户的密码：

update mysql.user set password=password('新密码') where User="test" and Host="localhost";

删除用户：

delete from user where User='test' and Host='localhost';

GRANT

INSERT

**INSERT SELECT**

[Insert select 带选择复制一张表到另一张表](https://www.cnblogs.com/accumulater/p/6096729.html)

1.INSERT INTO SELECT语句

　　语句形式为：Insert into Table2(field1,field2,...) select value1,value2,... from Table1 [where column =value][]为可选内容要求目标表Table2必须在由于目标表Table2已经存在，所以我们除了插入源表Table1的字段外，还可以插入常量。示例如下：

　　insert into tjjrmx(yybh,xh,tjxmbh,jg,sfyx,zhxmbh,tjksbh,jcrq,jcys,ts,ckfw,disporder)  
      　　　　　　select '24',xh,tjxmbh,jg,sfyx,zhxmbh,tjksbh,jcrq,jcys,ts,ckfw,disporder from tjjrmx where yybh = 5

2.SELECT INTO FROM语句

      语句形式为：SELECT vale1, value2 into Table2 from Table1

      要求目标表Table2不存在，因为在插入时会自动创建表Table2，并将Table1中指定字段数据复制到Table2中。示例如下：

3.从一个数据库到另一个数据库

       语句形式为：insert into 数据库名.框架名.表名(列名) select (列名) from 数据库名.框架名.表名 where 条件

       例如：insert into MyEmp.dbo.tjdjb(yybh)  
　　 select yybh from MyCmd.dbo.tjdjb where djrq='2009-10-15' and yybh = '11'

SAVEPOINT

#测试保存点savepoint  
savepoint point1;  
update test  
set PROD\_ID=1;  
rollback to point1;  #回到保存点point1

**autocommit模式：在开启情况下，对于每条statement来说，都会自动形成一个commit，也就是会即时对开始和结束一个事务。所以，当出现rollback to savepoint出现这个错误时，第一步是检查autocommit模式是否有开启。**

**select @@autocommit; //查看是否有自动提交**

**set autocommit = 0 ; //设置关闭自动提交**

**>start transaction //事务开始**

**select \* from ordertotals;**

**start transaction;**

**delete from ordertotals;select \* from ordertotals;**

**rollback;select \* from ordertotals;**

**Commit; //事务结束**

**>start transaction;**

**delete from orderitems where order\_num = 20010;**

**delete from orders where order\_num = 20010;**

**>commit;**

隐含事务关闭 当 COMMIT 或 ROLLBACK 语句执行后，事务会自动关闭（将来的更改会隐含提交）。

SQL句子的常用格式

GROUP BY

HAVING

ORDER BY