% 替代 0 个或多个字符

\_ 替代一个字符

一般通配符用 LIKE 使用

[charlist] list中的任何单一字符，如【a-z】或【a,b,c】

[^charlist]、[!charlist] 不在list中的任何单一字符

正则表达式用 RLIKE 使用

/////////////////////////////////////////////////////////////////////////

ALTER 列【新增、删除、修改】

UPDATE 行【修改】

/////////////////////////////////////////////////////////////////////////

where 和 having 之后都是筛选条件，但是有区别的：

1.where在group by前， having在group by 之后

2.聚合函数（avg、sum、max、min、count），不能作为条件放在where之后，但可以放在having之后【引入having的原因】

where：对表中原有记录进行筛选

having：对聚合函数操作后得到的临时记录名称进行筛选。

（order by也可以使用聚合函数操作后得到的临时记录名称）

【case

when 。。。 then 。。。

end】 as Name 【case--end】表示对某列数据进行分段，并将其新列命名为Name

case…end 可视为自定义的函数，与一般函数不同的是，函数得到的结果是多个（分段）

对某一段的记录条数计数： sum（case when 。。。 then 1 else 0）

/////////////////////////////////////////////////////////////////////////

EXISTS 运算符用于判断查询子句是否有记录，如果有一条或多条记录存在返回 True，否则返回 False

SELECT column\_name(s)

FROM table\_name

WHERE EXISTS

(SELECT column\_name FROM table\_name WHERE condition);

/////////////////////////////////////////////////////////////////////////

UCASE(\*) 函数,把字段的值转换为大写

LCASE(\*) 函数,把字段的值转换为小写

NOW() 函数,返回当前系统的日期和时间

/////////////////////////////////////////////////////////////////////////

查询时简化表名（from+真实表名+别名）：

select 表别名.name from 实际表名 表别名;

如：

select A.name from TableA A;

/////////////////////////////////////////////////////////////////////////

1.表与表的合并：

INNER JOIN （两个表的各取一部分进行合并）

LEFT JOIN （第一个表在左边作为基础）

RIGHT JOIN （第一个表在右边作为基础）

FULL JOIN （两个表完全合并，SQL暂不支持，可以通过【select a.\*,b.\* on ...】实现）

on和where的区别：

where 后的语句必须为真才返回（可能会过滤掉基础表的记录）

on 后的语句都会返回，不满足条件则临时表中该项值为空（on后条件为假时可体现left join和right join的区别；【a，b，null 和 null，b，c】）

【...A XXX JOIN B...】和 FROM 【...FROM A,B WHERE...】区别：

1.如查询信息在A、B中都存在，结果无差异

2.如A、B中某表无满足的记录，JOIN 语句仍会返回A表的记录

所以，使用on是为了完全合并两张表，不符合连接要求的记录也需要返回

注：

A inner join B； 取交集。

A left join B； 取 A 全部，B 没有对应的值为 null

A right join B； 取 B 全部 A 没有对应的值为 null

A full outer join B； 取并集，彼此没有对应的值为 null

对应条件在 on 后面填写

2.两次查询的合并

UNION （不显示重复的结果）

UNION ALL （显示重复的结果）

注：

1.两次查询结果的返回项，类型必须一致（多个项时需按顺序对应一致）

2.ORDER BY 只能当前 SQL 查询结果进行排序，如要对 union all 出来的结果进行排序，需要先做集合

select aa.\* from

(select country,name from websites where country = 'CN'

union all

select country,app\_name from apps where country='CN' )

aa order by aa.name;

3.复制表的内容

//SELECT INTO 语句从一个表复制数据，然后把数据插入到另一个新表中【要求目标表不存在】

SELECT \*

INTO table2

FROM table1;

//可用于创建新的空表,只需要添加促使查询没有数据返回的 WHERE 子句

SELECT \*

INTO newtable

FROM table1

WHERE 1=0;

//INSERT INTO SELECT 语句从一个表复制数据，然后把数据插入到一个已存在的表中【要求目标表存在】

INSERT INTO table2

SELECT \* FROM table1;

分组查询 GROUP BY ：

作为查出的记录的关键字，即不会出现两次

如后面有两个列名，则表示两个列名组合成关键字，相同组合不会出现两次

================================数据库使用优化=================================

1. EXPLAIN语句提供有关MySQL如何执行语句的信息。

EXPLAIN适用于SELECT、DELETE、INSERT、REPLACE和UPDATE语句

显示sql执行时要访问的表

rows 字段表示要扫描的行数

extra 字段：

Using index==在索引中就可查找所需数据

Using temporary==使用临时表, 出现于排序, 分组和多表 join, 建议优化

Using filesort==不能通过索引顺序达到排序，建议优化

type字段表示执行效率，效率从低到高如下



1. SQL语句中IN包含的值不应过多, 能用between就不用in了；再或者使用连接来替换。
2. SELECT语句务必指明字段名称，非必要不使用select \*
3. 当只需要一条数据的时候，使用limit 1
4. 如果排序字段没有用到索引，就尽量少排序
5. 如果限制条件中其他字段没有索引，尽量少用or。or两边的字段中，如果有一个不是索引字段，而其他条件也不是索引字段，会造成该查询不走索引的情况。很多时候使用union all或者是union（必要的时候）的方式来代替“or”会得到更好的效果
6. 尽量用union all代替union。union进行唯一性过滤操作，这就会涉及到排序，增加大量的CPU运算，加大资源消耗及延迟
7. IN适合于外表大而内表小的情况；EXISTS适合于外表小而内表大的情况
8. 避免在where子句中对字段进行null值判断，对于null的判断会导致引擎放弃使用索引而进行全表扫描。
9. 不建议使用%前缀模糊查询，LIKE“%name”或者LIKE“%name%”，这种查询会导致索引失效而进行全表扫描，必须使用可以创建并使用全文索引。
10. 避免在where子句中对字段进行表达式操作。如select user\_id,user\_project from user\_base where age\*2=36; 【age\*2】
11. 必要时可以使用force index来强制查询走某个索引