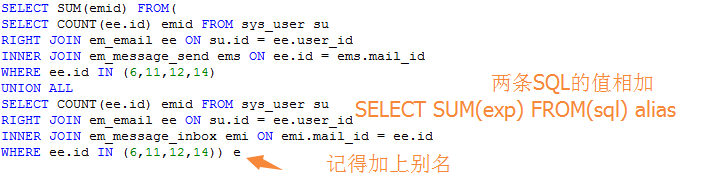
两个SQL的某个字段相加



**UNION和UNION ALL关键字**

**前者不显示重复行，默认进行排序运算，效率低**

**后者显示全部行，默认不进行排序运算，效率高**

[FIND\_IN\_SET函数](https://www.w3cschool.cn/wqf_database/wqf_database-oqa42m0w.html)



返回在集合中的索引位置（竖向发展)

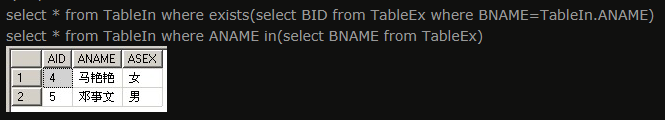
**Mysql 的不等于写法，<> 或者 !=**

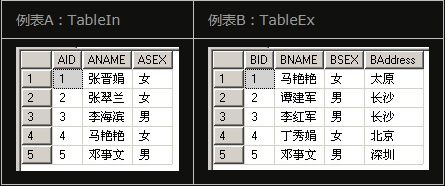
**SELECT 1 <> 2、SELECT 2 != 1**

**MySQL：**

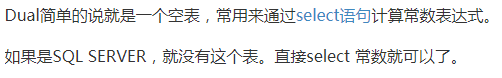
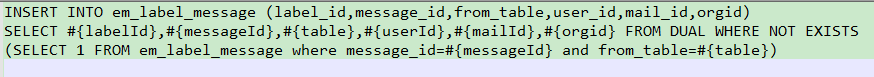
**[EXISTS关键字，存在/成立则返回数据](https://www.cnblogs.com/xuanhai/p/5810918.html)**

**[NOT EXISTS关键字，不存在/不成立则返回数据](https://www.cnblogs.com/xuanhai/p/5810918.html)**



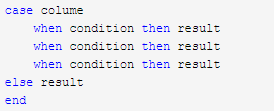


数据查重一步到位

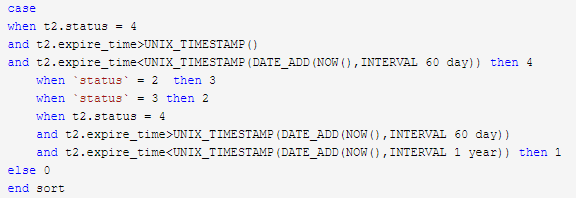


**CASE WHEN关键字，有点像switch case，用在select后边，**

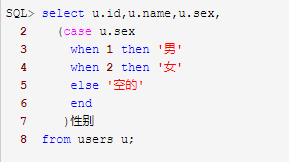
**语法结构：**



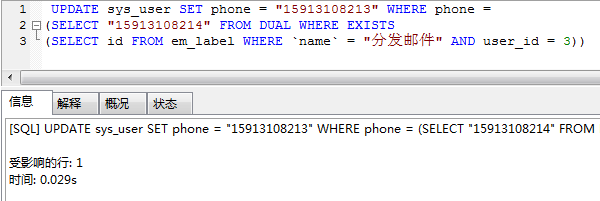
例子：



当满足某一条件时(condition)则执行then，并把结果返回给end



**MySQL，不能同时对同一张表进行读写操作**



查询、更新一步到位，但不能同时对一张表进行**读**和**写，因为（读|写）的过程中mysql会进行锁表操作**

**获取当前时间，NOW()，yyyy-MM-dd HH:mm:ss**

**获取当前日期，CURDATE()，yyyy-MM-dd**

**获取当前时间，CURTIME()，HH:mm:ss**

**获取时间戳，UNIX\_TIMESTAMP()，1540811128**

[**https://blog.csdn.net/qq\_37634366/article/details/79014815**](https://blog.csdn.net/qq_37634366/article/details/79014815)

****

拼接字符串

[CONCAT(str1,str2,str3,...)](https://www.cnblogs.com/northeastTycoon/p/5505637.html)

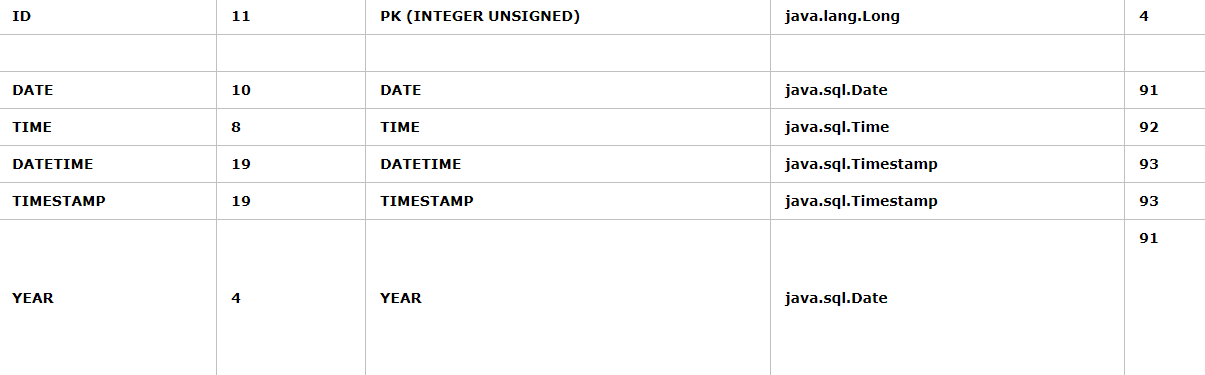
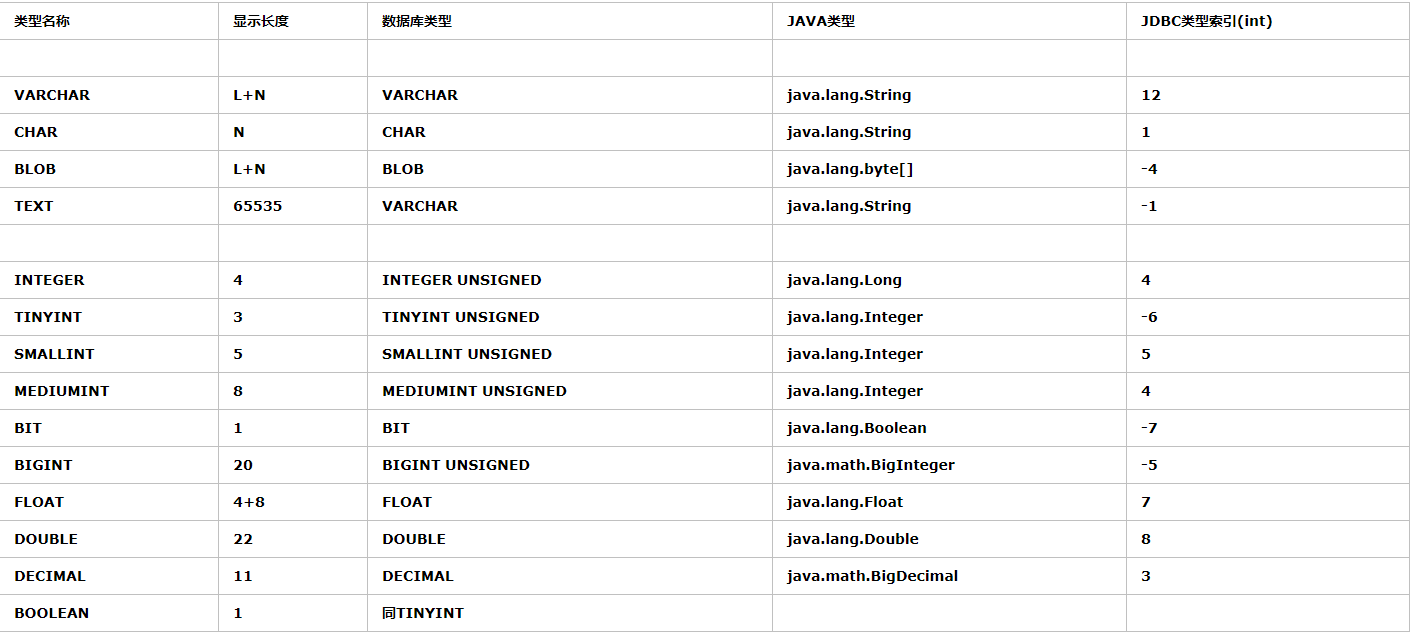


[MySQL模糊查询，](http://www.runoob.com/mysql/mysql-like-clause.html)

左模糊，%a 右模糊，a%，全模糊，%a%



[MySQL的bit类型在java中对应Boolean值](https://www.cnblogs.com/binyue/p/4730054.html)





[修改MySQL数据类型（MODIFY）](https://www.cnblogs.com/freeweb/p/5210762.html)

ALTER TABLE 表名 MODIFY COLUMN 字段名 类型【NULL / NOT NULL】

[MODIFY与ALTER区别](https://blog.csdn.net/qq_32953185/article/details/60594783)



ALTER TABLE 表名 相应的功能(MODIFY、ADD、SET...)

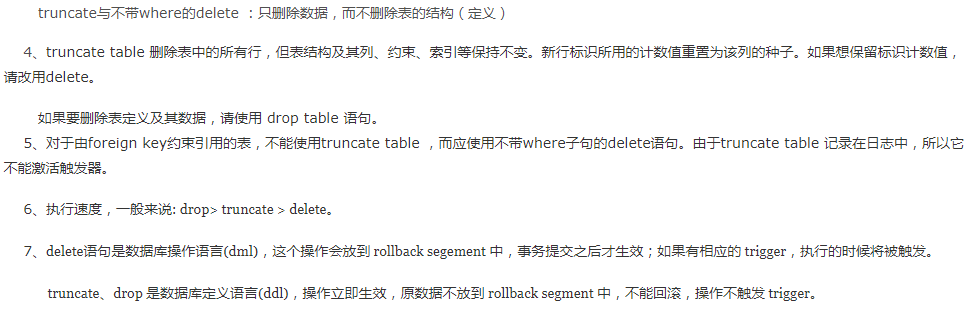
清空表数据

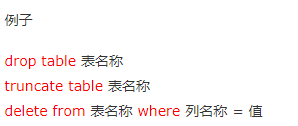
DELETE FROM table\_name

删除表

DROP TABLE table\_name







<https://blog.csdn.net/ninicaoyuan/article/details/76505482>

做成一张临时表，别名!

SELECT \* FROM (SELECT NOW()) a

表不能同时进行查询和更改的操作，但是把查询做成一张临时表就可以了。

UPDATE sys\_user SET orgname ="A组" WHERE id IN (SELECT \* FROM(SELECT id FROM sys\_user WHERE depid = 85 AND orgid = 2 AND ustate = 1) a)

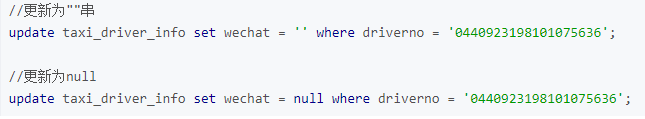
相当于查询一张临时表和更新一张表。

<https://blog.csdn.net/younglao/article/details/76570187>

更新字段为NULL或者""

UPDATE table\_name SET properties=”” WHERE xx=xx

UPDATE table\_name SET properties=NULL WHERE xx=xx



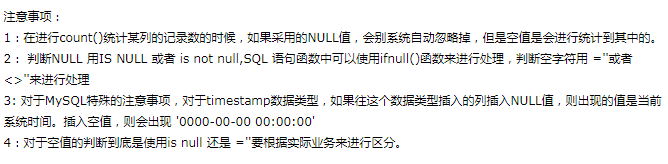
[截取空字符串 TRIM()](https://www.jb51.net/article/48473.htm)

如去除前后空格



[NULL与空字符串的区别](https://www.cnblogs.com/wzmenjoy/p/4244590.html)

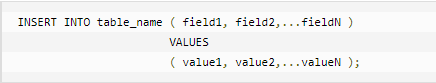
NULL不同于Java会占用空间不相关索引，反之空字符””不会占用空间

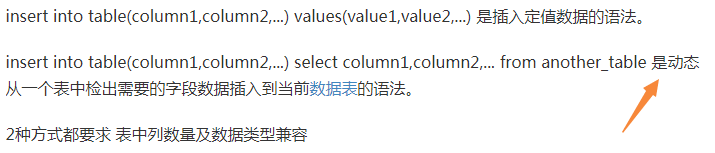


[单行注释、多行注释](https://www.cnblogs.com/JiangLe/p/6897403.html)



[MySQL插入时不带values](https://zhidao.baidu.com/question/454609546018269885.html)





Mapper中使用模糊查询

LIKE关键字配合使用

1、直接在传入xml前进行拼接(“\_”或者”%”)

LIKE #{field}

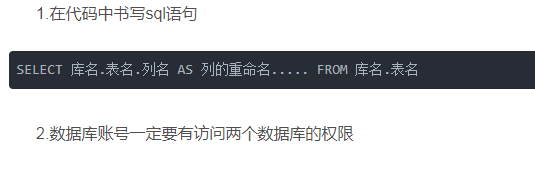
2、xml中使用CONCAT进行拼接

LIKE CONCAT("%", #{field}, "%")

3、使用单引号进行拼接

LIKE '%field%'

[MySQL跨库查询](https://blog.csdn.net/SimpleCow/article/details/80772813)



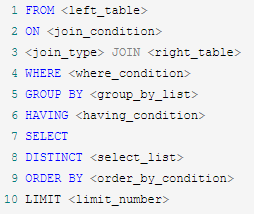
MySQL

LIMIT 条数

LIMIT(offset偏移值， limit条数)

SQL优化，简单查，不连表，数据多影响效率

SQL解析顺序



<https://www.cnblogs.com/annsshadow/p/5037667.html>

索引，加快对表中记录的查找或排序

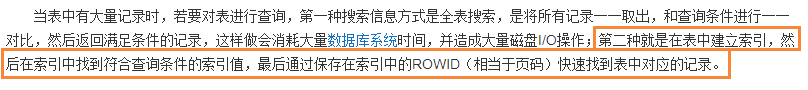
·概念：<https://baike.baidu.com/item/%E7%B4%A2%E5%BC%95/5716853?fr=aladdin#2>

在关系型数据库中，索引是对一列或多列的值进行排序的**一种存储结构。**

这种结构是**逻辑指针清单，**这份清单由**某一列或多列的集合**和**相应的指向这些值的数据页**共同组成。

存储结构=逻辑指针清单=加索引列的集合+指向这些值的数据页(指针)

数据库使用索引以找到特定值，然后顺指针找到包含该值的行。

数据查询流程：

类型：



主键索引是唯一索引的特定类型。

存储类型：

BTREE、HASH

两者的区别，效率上HASH > BTREE，后者，从根到枝。

HASH的弊端

Hash 索引仅仅能满足"=","IN"和"<=>"查询，不能使用范围查询。

经过一个计算后，不能保证和计算前的值一致。一次定位

Hash 索引无法被用来避免数据的排序操作。

经过一个计算后，不能保证和计算前的值一致。一次定位

Hash 索引不能利用部分索引键查询。

因为计算当前HASH索引时是通过组合索引键(Key)计算的，所以不能利用部分索引键(Key)单独查询

Hash 索引在任何时候都不能避免表扫描。

因为HASH运算结果和相对应的指针信息都放在一个HASH表中，由于不同索引键存在相同的HASH值，所以即便是找到满足某个HASH值的数据的记录条数，也要找完HASH表才能返回。

Hash 索引遇到大量Hash值相等的情况后性能并不一定就会比B-Tree索引高。

对于选择性比较低的索引键，如果创建Hash索引，那么将会存在大量记录指针信息存于同一个Hash值相关联。这样要定位某一条记录时就会非常麻烦，会浪费多次表数据访问，而造成整体性能底下

·存储类型：<https://www.cnblogs.com/vicenteforever/articles/1789613.html>

MySQL相对存储引擎支持的索引类型



·MySQL官方文档：<https://dev.mysql.com/doc/refman/5.6/en/create-index.html>



·博文：<https://www.cnblogs.com/whgk/p/6179612.html>

查看索引

SHOW INDEX FROM table\_name(表名)

返回字段解释：

· Table

表的名称。

· Non\_unique

如果索引不能包括重复词，则为0。如果可以，则为1。

· Key\_name

索引的名称。

· Seq\_in\_index

索引中的列序列号，从1开始。

· Column\_name

列名称。

· Collation

列以什么方式存储在索引中。在MySQL中，有值‘A’（升序）或NULL（无分类）。

· Cardinality

索引中唯一值的数目的估计值。通过运行ANALYZE TABLE或myisamchk -a可以更新。基数根据被存储为整数的统计数据来计数，所以即使对于小型表，该值也没有必要是精确的。基数越大，当进行联合时，MySQL使用该索引的机 会就越大。

· Sub\_part

如果列只是被部分地编入索引，则为被编入索引的字符的数目。如果整列被编入索引，则为NULL。

· Packed

指示关键字如何被压缩。如果没有被压缩，则为NULL。

· Null

如果列含有NULL，则含有YES。如果没有，则该列含有NO。

· Index\_type

用过的索引方法（BTREE, FULLTEXT, HASH, RTREE）。

[创建索引](https://www.cnblogs.com/a-du/p/7117837.html)

ALTER TABLE 表名 ADD INDEX 索引名 (要建索引的列)

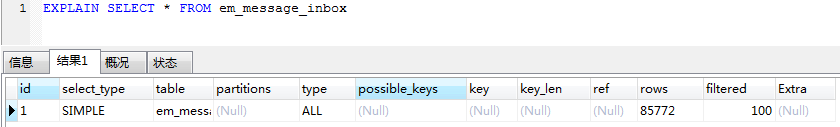
[查看进程](http://www.cnblogs.com/zengkefu/p/6527273.html)

SHOW PROCESSLIST;

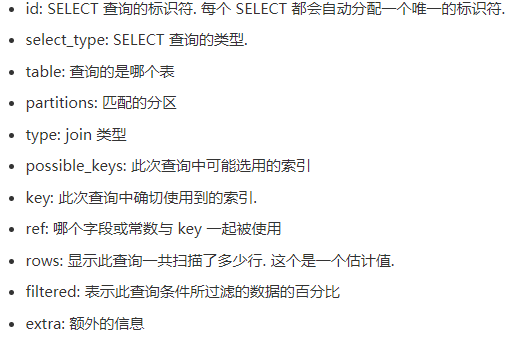
[结束进程](http://www.cnblogs.com/zengkefu/p/6527273.html)

KILL thread\_id(进程id)

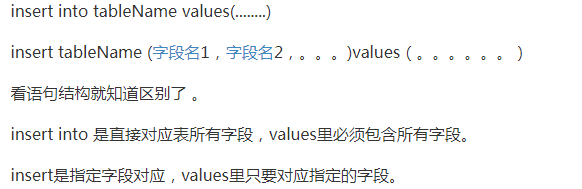
MySQL性能分析关键字，EXPLAIN



[各列含义：](https://segmentfault.com/a/1190000008131735)



[insert 和 insert into 的区别](https://zhidao.baidu.com/question/1301831163955190139.html)



MySQL查看表结构

DESC table\_name

显示表结构、字段类型等信息

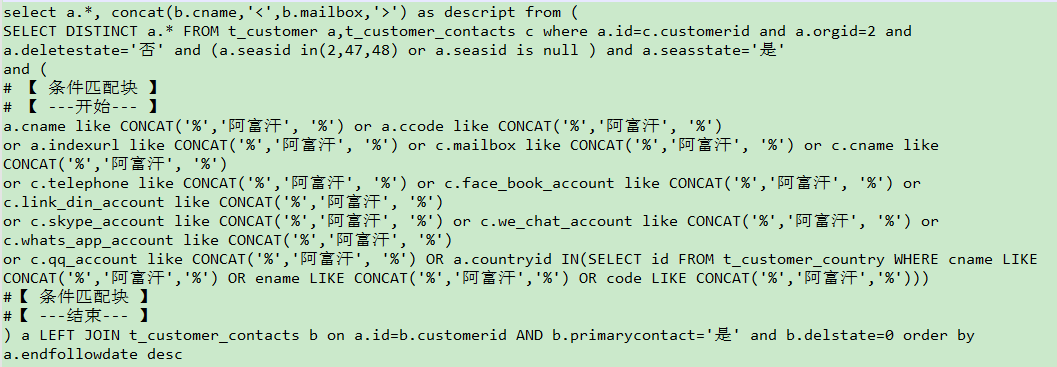
<https://www.cnblogs.com/zhangyuhang3/p/6873895.html>

MySQL的IN关键字，

可使用NULL来填补

善用开发工具的功能，比如Navicat Premium的语法高亮、提示等功能(可较容易的查看括号的开闭)

MySQL复杂查询时如何构建查询体，利用子查询是其中一种方案。



[MySQL的DETE\_FORMAT()函数](https://www.w3cschool.cn/mysql/func-date-format.html)

MySQL返回Map类型时，时间格式的返回需要处理，

2018-12-30 00:00:01:001

DATE\_FORMAT(str,"%Y-%m-%d %H-%i-%s")

具体时间格式：

<https://www.w3cschool.cn/mysql/func-date-format.html>

MySQL对指定时间增加一天(减少用DATE\_SUB())



type的取值以及该函数的具体用法：

<https://www.w3cschool.cn/mysql/func-date-add.html>

MySQL时间存储格式

<https://blog.csdn.net/qq_28483283/article/details/81873054>

<https://www.w3cschool.cn/mysql/mysql-data-types.html>

DATETIME 和 TIMESTAMP间的区别

虽然都是可以表示年月日 时分秒，但是，TIMESTAMP可以涉及时区、时间戳。

DATE\_FORMAT(tcf.followtime, "%Y-%m-%d")=DATE\_FORMAT("2018-12-29 00:00:00", "%Y-%m-%d")

遇到的一个问题，数据库中查询正常，但返回时，数据会缩减一天。

目前解决方式，先格式化再进行匹配

[MySQL添加和删除字段](https://blog.csdn.net/mjshldcsd/article/details/78295305)

添加：

ALTER TABLE table\_name ADD field\_name field\_type;

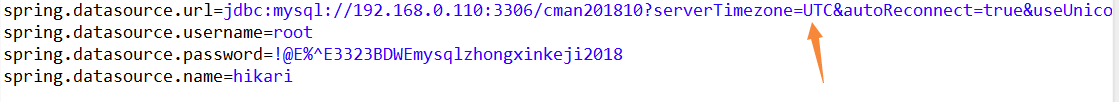
删除：

ALTER TABLE table\_name DROP field\_name;

CONVERT\_TZ(dt,from\_tz,to\_tz)

从一个时区的时间转换成另一个时区的时间(datetime类型)

如果在连接MySQL配置信息的文件中设置了时区(ServerZone)，那么查询出来的结果都是以设置的时区为准



这时候可以使用CONVERT\_TZ(dt,from\_tz,to\_tz)函数来进行转换

<https://www.w3cschool.cn/sql/82rg1ozi.html>

MySQL查询使用索引查询

SELECT \* FROM 表名 USE INDEX(索引名) WHERE 条件

查看表索引：

SHOW INDEX FROM 表名

MySQL字符串转数字

<https://blog.csdn.net/yaoyao9565/article/details/52790081/>

MySQL中CONVERT()函数



<https://www.cnblogs.com/chenqionghe/p/4675844.html>

MySQL字符串截取

一.从**左**开始截取字符串

用法：left(str, length)，即：left(被截取字符串， 截取长度)

二.从**右**开始截取字符串

用法：right(str, length)，即：right(被截取字符串， 截取长度)

三.截取**特定长度**的字符串

用法：

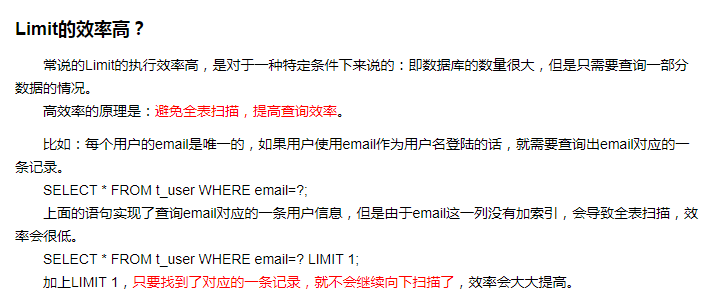
* substring(str, pos)，
  + 即：substring(被截取字符串， 从第几位开始截取)
* substring(str, pos, length)，
  + 即：substring(被截取字符串，从第几位开始截取，截取长度)

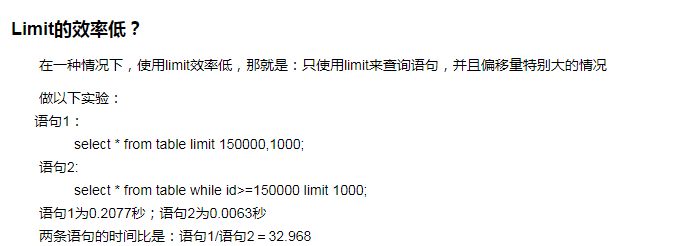
四.按**关键字**进行读取

用法：substring\_index(str, delim, count)，即：substring\_index(被截取字符串，关键字，关键字出现的次数)

<https://www.cnblogs.com/heyonggang/p/8117754.html>

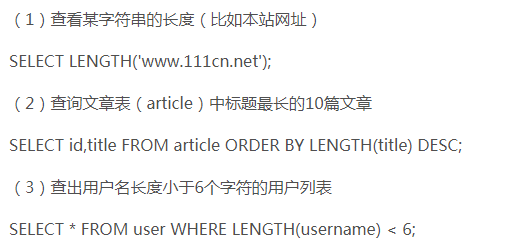
MySQL的LIMIT详解





<https://www.cnblogs.com/acm-bingzi/p/msqlLimit.html>

MySQL字符长度函数(LENGTH(str))



<https://blog.csdn.net/weixin_42056745/article/details/80772573>

Mysql获取上个月，上上个月，利用存储过程在当前时间进行相减

// 查询当前月份

SELECT　DATE\_FORMAT(DATE\_SUB(NOW(), INTERVAL 0 MONTH), “%m”)

// 查询上个月月份

SELECT　DATE\_FORMAT(DATE\_SUB(NOW(), INTERVAL 1 MONTH), “%m”)

// 查询上上个月月份

SELECT　DATE\_FORMAT(DATE\_SUB(NOW(), INTERVAL 0 MONTH), “%m”)

<https://blog.csdn.net/zhangshuhao0101/article/details/79020260>

**更爱字段名称，使用change关键字**

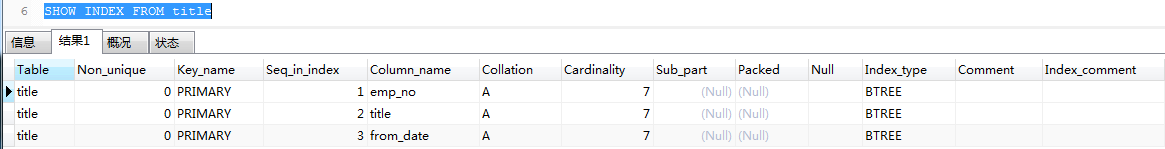
**ALTER TABLE table\_name CHANGE old\_column\_name new\_column\_name 原来的类型 是否为空 注释;**

[**Mysql修改字段类型，修改字段名**](https://blog.csdn.net/u010002184/article/details/79354136)

**MySQL的优化主要分为结构优化（Scheme optimization）和查询优化（Query optimization），这里以结构优化范畴为例**

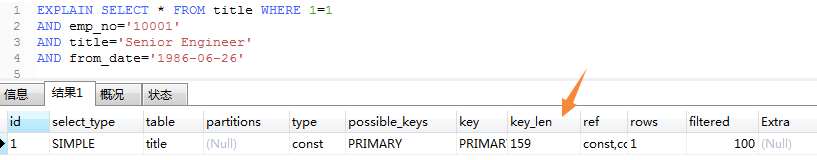
**高效使用索引的首要条件是知道什么样的查询会使用到索引，这里列举七种情况**

**这里存在组合的聚集索引，PEIRMARY KEY(`emp\_no`,`title`,`from\_date`);**

****

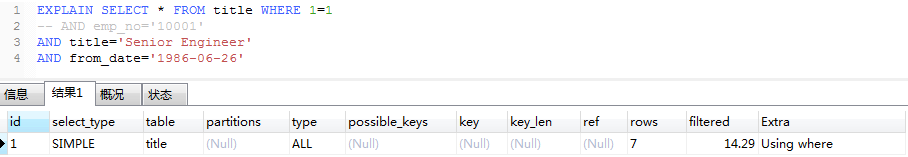
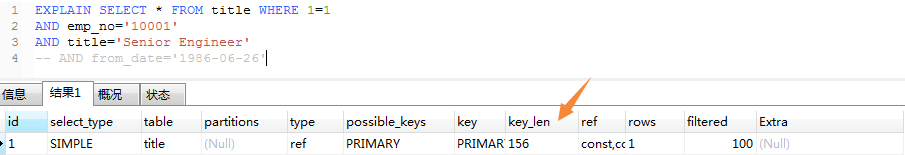
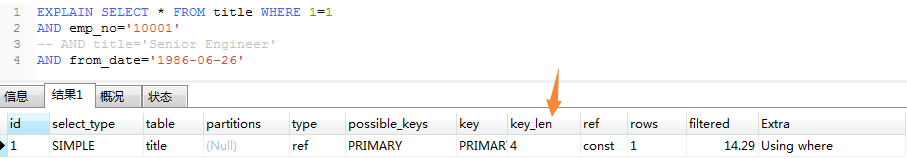
**情况一，全列匹配**

**所有列进行精确匹配时（这里指“=”或“IN”）索引可以被用到，这里有一点需要注意，理论上索引对顺序是敏感的，但是由于MySQL的查询优化器会自动调整where子句的条件顺序以使用适合的索引，例如我们将where中的条件顺序颠倒结果还是一样：**

****

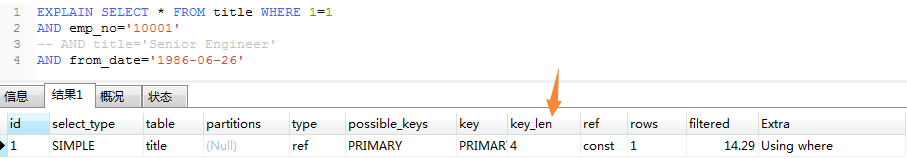
**情况二，最左前缀匹配，从上边知道组合索引的顺序是，<emp\_no,title,from\_date>**

**看下使用到索引第一列、第一二列（由于title不存在而无法和左前缀连接），第二三列的情况（这种不使用索引）**

****

**因此，最左前缀匹配，突出点，从最左边开始计算**

**情况三，查询条件用到了索引中列的精确匹配，但是中间某个条件未提供。**

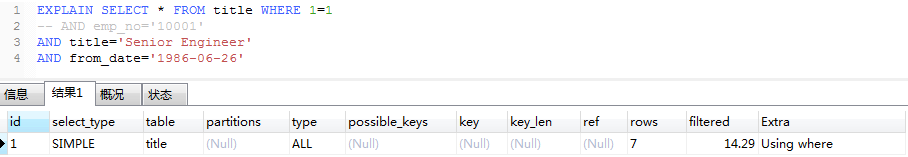
****

**造成原因是查询只用到了索引的第一列，而后面的from\_date虽然也在索引中，但是由于title不存在而无法和左前缀连接。所以两种填补方式，**

**1、emp\_no与from\_date建立辅助索引**

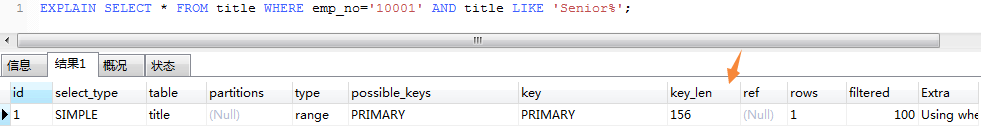
**2、title字段进行填补，如使用IN关键字进行匹配，使用场景，当title列值较少时**

**情况四：查询条件没有指定索引第一列**

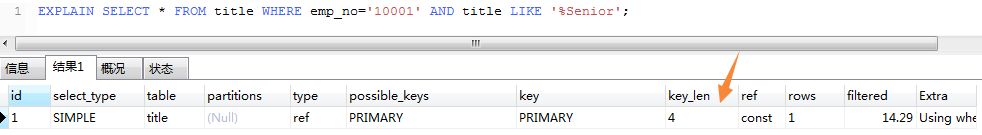
****

**不是最左前缀，因此用不到索引**

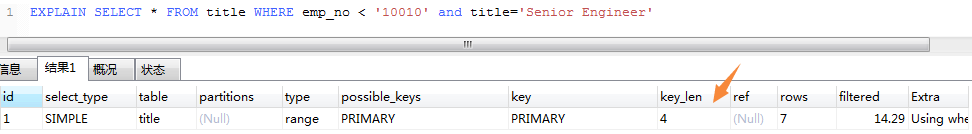
**情况五：匹配某列的前缀字符串。**

****

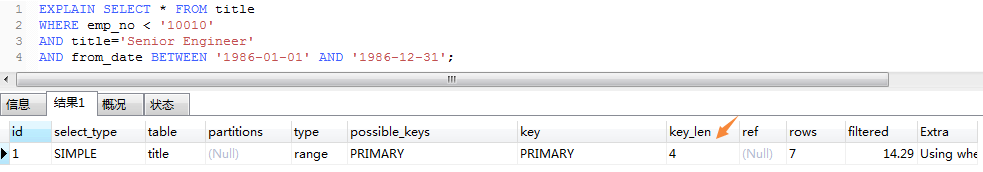
**可以看到，此时有使用到索引，如果通配符%不出现在开头，则可以用到索引，但根据具体情况不同可能只会用其中一个前缀**

****

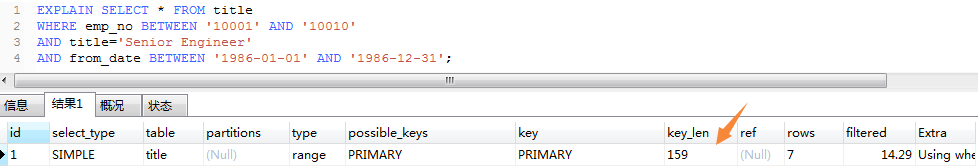
**情况六：范围查询**

****

**范围列可以用到索引（必须是最左前缀），但是范围列后面的列无法用到索引。同时，索引最多用于一个范围列，因此如果查询条件中有两个范围列则无法全用到索引。**

****

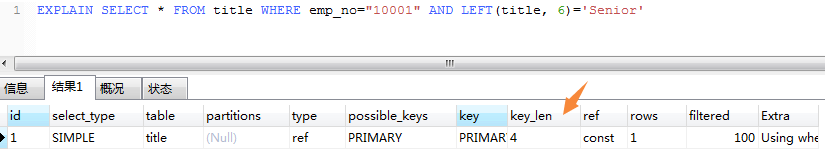
**可以看到索引对第二个范围索引无能为力。这里特别要说明MySQL一个有意思的地方，那就是仅用explain可能无法区分范围索引和多值匹配，因为在type中这两者都显示为range。同时，用了“between”并不意味着就是范围查询，例如下面的查询：**

****

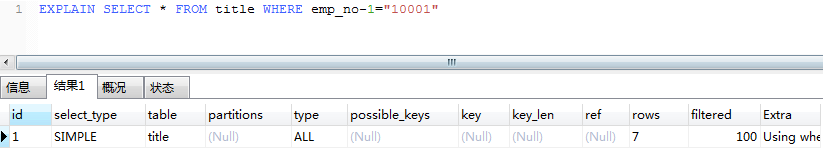
**看起来是用了两个范围查询，但作用于emp\_no上的“BETWEEN”实际上相当于“IN”，也就是说emp\_no实际是多值精确匹配。可以看到这个查询用到了索引全部三个列。因此在MySQL中要谨慎地区分多值匹配和范围匹配，否则会对MySQL的行为产生困惑。**

**情况七：查询条件中含有函数或表达式**

**如果查询条件中含有函数或表达式，则MySQL不会为这列使用索引（虽然某些在数学意义上可以使用）。**

****

**可以看到，第二个索引没使用到，因为它使用了函数left<按指定字数截取字符>**

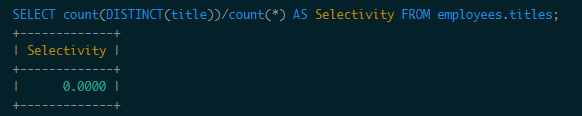
****

**由于查询条件是一个表达式，MySQL无法为其使用索引。**

**索引建立的必要，参考**

**1、表记录的数量。如表记录达到2000或者3000记录时。**

**2、索引的选择性。计算公式：索引的选择性=不重复的列值/记录总数，如**

****

**title的选择性不足0.0001（精确值为0.00001579），所以实在没有什么必要为其单独建索引。**

**总言之就是越接近1就越有建立索引的必要。**

**[MySQL索引背后的数据结构及算法原理](http://blog.codinglabs.org/articles/theory-of-mysql-index.html)**

**key\_len的长度计算公式，这里以VARCHAR类型为例：**

**VARCHAR(10)变长字段且允许NULL：**

**10 \* ( character set：utf8=3,gbk=2,latin1=1)+1(NULL)+2(变长字段)  
VARCHAR(10)变长字段且不允许NULL：**

**10 \*( character set：utf8=3,gbk=2,latin1=1)+2(变长字段)**

**CHAR(10)固定字段且允许NULL：**

**10 \* ( character set：utf8=3,gbk=2,latin1=1)+1(NULL)  
CHAR(10)固定字段且不允许NULL：**

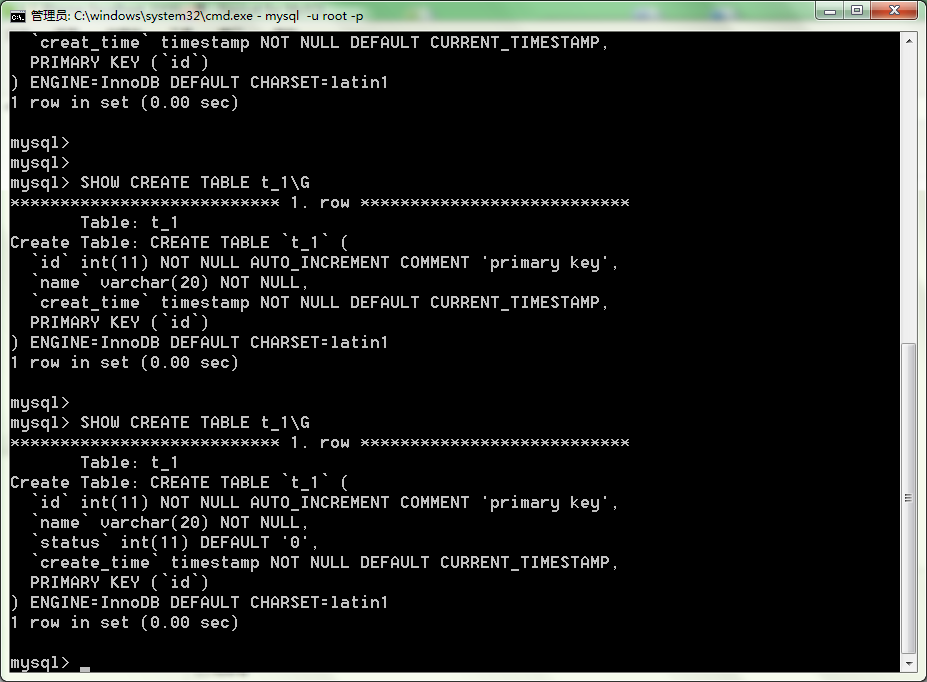
**10 \* ( character set：utf8=3,gbk=2,latin1=1)**

[**explain之key\_len计算**](https://www.cnblogs.com/gomysql/p/4004244.html)

**查看mysql的字符编码设置**

**show variable like ‘charter%’;**

**建表**



**CREATE TABLE table\_name(**

**字段数据类型 是否为空 是否要默认值 注释**

**..**

**主键设置**

**) 存储引擎 字符集设置...等等设置**

# [MySQL timestamp NOT NULL插入NULL的问题](https://www.cnblogs.com/suifu/p/5823052.html)

**timestamp NULL值的控制由mysql内部的一个系统变量**

explicit\_defaults\_for\_timestamp进行控制（5.6版本引入）

该参数能有效对传入的timestamp值进行控制，如果不开启，插入NULL值时不报错插入后时间为当前时间。

可以在使用

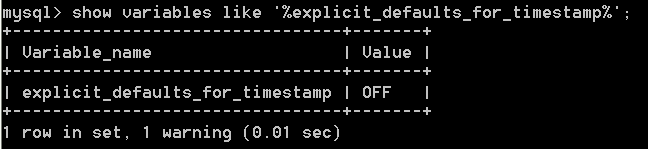
SET explicit\_defaults\_for\_timestamp=1

SET explicit\_defaults\_for\_timestamp=0

进行控制

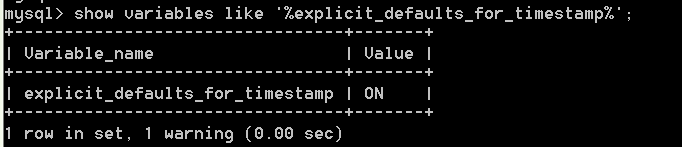
进行查看

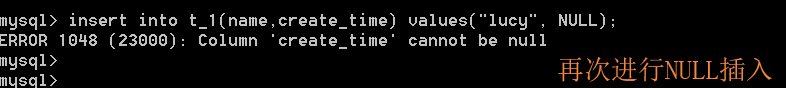
show variables like '%explicit\_defaults\_for\_timestamp%';











explicit\_defaults\_for\_timestamp除此之外还有一个功能。

如果一行数据中某些列被更新了，如果这一行中有timestamp类型的列，那么么这个timestamp列的数据

也会被自动更新到 更新操作所发生的那个时间点；这个操作是由**explicit\_defaults\_for\_timestamp**这个变更控制的

## [mysql explicit\_defaults\_for\_timestamp 变量的作用](https://www.cnblogs.com/JiangLe/p/6956865.html)

# [MySQL修改表、字段、库的字符集及字符集说明](https://www.cnblogs.com/qiumingcheng/p/10336170.html)

修改字段的字符集：

ALTER TABLE tbl\_name CHANGE c\_name c\_name CHARACTER SET character\_name [COLLATE ...];

如：ALTER TABLE logtest CHANGE title title VARCHAR(100) CHARACTER SET utf8 COLLATE utf8\_general\_ci;

修改数据库字符集：

ALTER DATABASE db\_name DEFAULT CHARACTER SET character\_name [COLLATE ...];

把表默认的字符集和所有字符列（CHAR,VARCHAR,TEXT）改为新的字符集：

ALTER TABLE tbl\_name CONVERT TO CHARACTER SET character\_name [COLLATE ...]

如：ALTER TABLE logtest CONVERT TO CHARACTER SET utf8 COLLATE utf8\_general\_ci;

# [MySQL\_避免重复插入\_IGNORE](https://blog.csdn.net/u010003835/article/details/54381763)



还有个应用就是复制表,避免重复记录：

INSERT IGNORE INTO `table\_1` (`name`) SELECT `name` FROM `table\_2`;

[**ORDER BY**](https://blog.csdn.net/oYeYuanXinZhiZhu1/article/details/79079544)

1、普通排序

SELECT \* FROM table ORDERY BY col\_name;

2、多列排序

SELECT 列名1,列名2,.... FROM 表名ORDER BY 列名3,列名4;

先按列名3排序,若出现相同行，再按列名4排序，以此类推。

除了用列名还可以使用列的下标。SELECT 列名1,列名2,.... FROM 表名ORDER BY 3, 4;

3、指定排序方向

SELECT prod\_id,prod\_price,prod\_name FROM products ORDER BY prod\_price DESC; // 降序，默认ASC升序

4、顺序排序

SELECT prod\_price FROM products ORDER BY prod\_price DESC LIMIT 1;

[**mysqld与mysql区别**](https://zhidao.baidu.com/question/279891181.html)

**mysqld（**mysql daemon**）既服务，mysql服务器端程序。监听mysql端口， 如3306，因为客户端是通过连接服务器端来访问数据库，所以必先启动mysqld.exe才能进行mysql客户端的访问。**

**mysql则是客户端，可以对服务器端发起访问连接。**



# [MySQL 慢查询日志(Slow Query Log)](https://blog.csdn.net/leshami/article/details/39829605)

同大多数关系型数据库一样，日志文件是MySQL数据库的重要组成部分。MySQL有几种不同的日志文件，通常包括错误日志文件，二进制日志，通用日志，慢查询日志，等等。这些日志可以帮助我们定位mysqld内部发生的事件，数据库性能故障，记录数据的变更历史，用户恢复数据库等等。本文主要描述通用查询日志。

a、[错误日志](https://blog.csdn.net/leshami/article/details/39759849)：记录启动、运行或停止mysqld时出现的问题。

   b、[通用日志](https://blog.csdn.net/leshami/article/details/39779225)：记录建立的客户端连接和执行的语句。

   c、更新日志：记录更改数据的语句。该日志在MySQL 5.1中已不再使用。

   d、[二进制日志](https://blog.csdn.net/leshami/article/details/39801867)：记录所有更改数据的语句。还用于复制。

   e、慢查询日志：记录所有执行时间超过long\_query\_time秒的所有查询或不使用索引的查询。

   f、Innodb日志：innodb redo log

缺省情况下，所有日志创建于mysqld数据目录中。

   可以通过刷新日志，来强制mysqld来关闭和重新打开日志文件（或者在某些情况下切换到一个新的日志）。

   当你执行一个FLUSH LOGS语句或执行mysqladmin flush-logs或mysqladmin refresh时，则日志被老化。

   对于存在MySQL复制的情形下，从复制服务器将维护更多日志文件，被称为接替日志。

慢查询日志是将mysql服务器中影响数据库性能的相关SQL语句记录到日志文件，通过对这些特殊的SQL语句分析，改进以达到提高数据库性能的目的。

   通过使用slow\_query\_log[={0|1}]选项来启用慢查询日志。

所有执行时间超过long\_query\_time（默认10）秒的SQL语句都会被记录到慢查询日志。

   缺省情况下hostname-slow.log为慢查询日志文件安名，存放到数据目录，同时缺省情况下未开启慢查询日志。

   缺省情况下数据库相关管理型SQL(比如OPTIMIZE TABLE、ANALYZE TABLE和ALTER TABLE)不会被记录到日志。

   对于管理型SQL可以通过log-slow-admin-statements开启记录管理型慢SQL。

   mysqld在语句执行完并且所有锁释放后记入慢查询日志。记录顺序可以与执行顺序不相同。获得初使表锁定的时间不算作执行时间。

MySQL服务器按以下顺序记录SQL是否写入到慢查询日志

a. 查询必须不是一个管理型SQL，或者开启了log-slow-adminstatement选项

b. 查询超过long\_query\_time时间或者没有使用到索引

c. 查询必须满足min\_examined\_row\_limit选项，即行数少于min\_examined\_row\_limit的查询不记录

d.查询超过log\_throttle\_queries\_not\_using\_indexes设置则不记录，

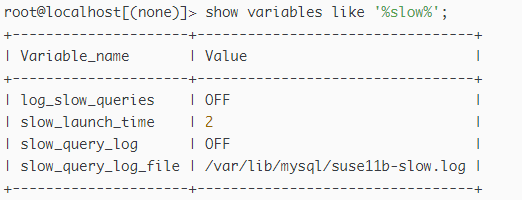
log\_throttle\_queries\_not\_using\_indexes指每分钟记录到日志的未使用索引的语句数目，超过这个数目后只记录语句数量和花费的总时间

选项分析：

|  |  |
| --- | --- |
| long\_query\_time | 设定慢查询的阀值，超出次设定值的SQL即被记录到慢查询日志，缺省值为10s |
| slow\_query\_log | 指定是否开启慢查询日志 |
| log\_slow\_queries | 指定是否开启慢查询日志(该参数要被slow\_query\_log取代，做兼容性保留) |
| slow\_query\_log\_file | 指定慢日志文件存放位置，可以为空，系统会给一个缺省的文件host\_name-slow.log |
| min\_examined\_row\_limit | 查询检查返回少于该参数指定行的SQL不被记录到慢查询日志 |
| log\_queries\_not\_using\_indexes | 不使用索引的慢查询日志是否记录到索引 |

是否开启慢日志查询

show variables like '%slow%';



慢查询时间阀值

show variables like '%long\_query\_time%';

开启慢日志查询

set global long\_query\_time=1; // 全局级别

set session long\_query\_time=1; // session级别

慢日志文件部分解析：

会出现操作SQL时间、用户、SQL执行时间、锁占有时间、原因...



格式化慢查询日志，mysqldumpslow命令



更多参数使用mysqldumpslow --help查看

COUNT(\*)、COUNT(1)

COUNT(\*)与COUNT(1)没什么太大区别，如果COUNT(\*)中的’\*’为NULL的话则不进行统计函数，返回’0’，COUNT(\*)相对于COUNT(1)进行过内部优化。但它们都是进行全表扫描，都会统计全为null值的记录。

导入导出数据库结构、数据（导出文件是不需要进行mysql客户端）

导出

mysqldump -u用戶名 -p密码 -d 数据库名 表名 > 脚本名

导出整个数据库结构和数据

mysqldump -h localhost -uroot -p123456 database > dump.sql

导出单个数据表结构和数据

mysqldump -h localhost -uroot -p123456 database table > dump.sql

导出整个数据库结构（不包含数据）

mysqldump -h localhost -uroot -p123456 -d database > dump.sql

导出单个数据表结构（不包含数据）

mysqldump -h localhost -uroot -p123456 -d database table > dump.sql

导入

mysql -h localhost -uroot -p123456 database < sqlfile // 没有进入mysql客户端进行

source sqlfile // 进入数据库后执行