mysql:

数据类型：

整形：int bit

------> bit 只能存储 0或1 二进制 显示不了的

小数：decimal

-------->decimal(5,2) 表示存5位数 小数占2位数

字符串：varchar char text

-------->char 表示存固定的字符串， char(3）如果 填充“ab” 会补上够上空格 “ab空格"

-------->varchar 表示可变长度字符， varchar(3) 填充 "ab" 就只填充"ab"

-------->text 表示存储大文本，当字符大于4000时推荐使用

日期：date time datetime

枚举类型：enum

---------> enum(“男”,“女”) 1为男 2为女

对于图片、音频、视频等文件，不存储在数据库中，而是上传到某个服务器中，然后，在表中存储这个文件的保存路径

约束：

主键primary key：物理上存储顺序

非空not null：此字段不允许为空值，不能填null

唯一unique：此字段的值不允许重复

默认default：当不填写值时或（填default），使用默认值

自动增长：auto\_increment

无符号：unsigned

外键foreign key：对关系字段进行约束，当为关系字段时，会到关联表中查询此值是否存在，如果存在则填写成功，不存在，填写失败抛出异常,会影响数据的增删改查，还会降低数据库的性能，不太推荐使用 可以在逻辑层上实现



命令行操作mysql

连接数据库：mysql -u用户 -p密码

退出数据库：quit 或 exit

Sql语句需要结尾需要加上（；）

显示当前MySQL版本：select version（）；

显示当前时间：select now（）；

；

对数据库操作：

查看具有的数据库的语句：show databases

创建数据库：create database 数据库名字 charset=utf8；

查看创建的数据库信息语句：show create database 数据库名字

删除数据库：drop database 数据库名字

使用数据库：use 数据库名字

查看当前在使用那个数据库：select database（）；

对表的操作：

查看当前数据库中具有的表：show tables;

创建表：create table 表名 （列名 类型 约束，列名 类型 约束）；

查看表设计信息：desc 表名；

删除表：drop table 表名；

修改表-添加字段：alter table 表名 add 列名 类型；

修改表-修改字段：alter table 表名 modify 列名 类型 约束；

（将具有的字段的 类型约束 更改了）

修改表-修改字段：alter table 表名 change列名 列名新名 类型 约束；

（将具有的字段 的名字 类型 约束 更改了）

增加外键：alter table 表A add foreign key（列名） references 表B(列名)

将表A的某个字段设置为外键 关联上表b（列名）

创建外键两个表中的字段类型 一定要一样

插入数据时 如果关联的表没有这个信息 就插入不进去

修改表-删除字段：alter table 表名 drop 列名；

删除外键： alter table 表名 drop foreign key 外键名称

show create table 表名； 如果有外键 里面 会有外键名称

数据的增删改查：

若字段设置约束：auto\_increment 自动递增 可以为0 null default 自动增加上去

全部插入数据：insert into 表名 values（） 这种插入数据要全填表中的所有字段

有些字段可为空：不写自动填为默认（空） 非空：一定要填

部分数据插入：insert into 表名 （列名，列名……） values(值……)

多行插入：insert into 表名 （列名……） values(值……)，（值……）

修改表中的数据：update 表名 set 列名=值，列名=值…… 所有都改

修改表中的数据：update 表名 set 列名=值，列名=值…… where 条件；

查询数据：select \* from 表名； 查询所有数据

Select \* from 表名 where 条件；

指定列查询：select 列名，列名…… from 表名；

设置别名： select 列名 as 别名，列名 as 别名 from 表名；

指定列查询：select 表名.列名，表名.列名 from 表名；

给表取别名：select 别名.列名，别名.列名 from 表名 as 别名；

消除重复行： select distinct 列名 from 表名

条件查询：

Where 条件: < > <= >= != = and(且) or(或)

not（条件） -🡪 不在这个条件内 不加括号 只否定前面的条件

模糊查询： 列名 like “数据配下面的符号”

% （替换1次或多次）

\_ (替换1次)

Name like “小%” 或 “小\_”

列名 rlike “正则表达式”

范围查询： 列名 in（）

in(10,20,30) not in() 非连续范围（有10，20，30）

Between 18 and 30 连续范围（18到30）

Not between 18 and 30

判断空： 列名 is null

列名 is not null

排序: 查询语句 order by 列名； 默认从小到大排序

Order by 列名 asc; 从小到大

Order by 列名 desc； 从大到小

多字段： Order by 列名 asc，列名 asc；

首先按照第一个字段数据排序，数值相同，再按照后面字段排序

聚合函数： select 聚合函数（\*或列名） from 表名 where 条件，

Select 聚合函数（\*或类名） as 别名 from 表名 where 条件；

Count（\*） 查询出来的数据 有多少条

Max（列名） 查询出来的数据 列名中最大的值

Min（列名） 查询出来的数据 列名中最小的值

Sum（列名） 查询出来的数据 列名求和

Avg（列名） 查询出来的数据 列名平均值

Round(数值，保留几位小数) 四舍五入

聚合函数之间 ：+ - \* /

Groug\_concat(列名，“ ”，列名) 用于分组中使用 写出各组中的列名数据

原理上不分组，select 列名 不能跟 聚合函数 混合查询：

Select 列名，聚合函数 from 表名 --->失败

分组： 分组使用的列名 最好具有标识性 比如性别：男女

select 列名（gender） from 表名 group by 列名（gender）；

性别分组

查询出来的数据 先分组，各组数据再聚合函数

（聚合计算 各组中的数据）

Select 列1名，聚合函数 from 表名 where 条件 group by 列1名

Having 条件： 查询出来的数据基础上再判断条件

分页：查询语句 limit 呈现个数； limit 填再sql语句最后

查询语句 limit 第n个开始（显示n+1），显示几个； 0开始为1

连接查询：多张表关联 将表1和表2 每一条数据组合起来

内连接查询： 表1 inner join 表2 on 表1.列名= 表2.列名 交集

Select \* from表1 Inner join 表2 on 表1.列名 = 表2.列名；

Select 表1.\*，表2.列名 form 表1 inner join 表2；

如果表名太长 可以取别名

左连接查询： 表1 left join 表2 on 表1.列名=表2.列名

交集 和 表1.列名 数据多余不跟表2.列名相同的

如：表1.cls\_id字段：1，2，5 表2.id字段：1，2

那么 3是多余的

显示表1.列名=3的数据 表2.为NULL

| 9 | 程坤 | 27 | 184.00 | 男 | 5 | | NULL | NULL |

右连接查询： 表1 right join 表2 on 表1.列名= 表2.列名

交集 和 表2.列名 数据多余不跟表1.列名相同

如：表1.cls\_id字段：1，2 表2.id字段：1，2，6

那么 5是多余的

| NULL | NULL | NULL | NULL | NULL | NULL | NULL | 6 | 傻逼班 |

Having 条件：可以在查询出来的数据基础上，再次条件判断

内连接查询 having 条件

自关联查询：同一张表中 俩个字段相互关联 （表自己合成）

比如： id name p\_id 存储省市区 没有上级填NULL

市区 p\_id 关联上 id

Select \* from 表 as 别名inner join 表 as 别名 on 条件 having 条件

子查询：select 里面 嵌套 slect ：先执行嵌套的select语句

嵌套语句 查询出的值一个

Select \* from 表名 where 列名 = （select语句）

表1 inner join （select语句） as 别名 on

将表A的某个字段 插入 表B 中的某个字段：

Insert into 表名 values（）：

Insert into 表名 （列名……） values（……）

Insert into 表B (列名) select语句（查出某个字段数据）

将表A与表B关联连接 更新某个字段的信息：

Update 表名 set 列名=值，列名=值……;

Update 表名 set 列名=值，列名=值…… where 条件；

Update 表A inner join 表B on 条件 set 表A.字段 = 表B.字段

物理删除数据：delete from 表名； 删除表中所有数据

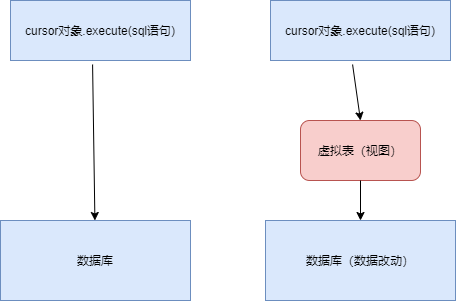
delete from 表名 where 条件；

（auto\_increment = 值） 删除了 也不要往回插

逻辑删除：设置一个字段 进行标记

Mysql 高级：

视图：理论上，有一个程序，如果 数据库进行了修改（增加表或者修改表 等），那么，之前写的程序里面，cursor对象.execute(sql语句)也要相应的改动sql语句的。但是 可以进行创建一张虚拟表,也就是创建视图，那么程序里面的sql语句，就不需要变动了。



创建视图:create view 视图名as select语句

相当于把select语句查询出来的数据，虚拟创建成一张表。

使用 show tables； 就可以查看所有表（视图是一张虚拟表）

视图只是用来方便查询数据，不允许进行增删改操作

删除视图：drop view 视图名

视图会关联slect 查询表的数据， 如果 表数据进行了更改，视图里面数据也会更改，视图只是虚拟表，并不会占数据空

事务： 是一个操作系列，这些操作要么都执行，要么都不执行，如果有一个操作失败，之前的所有操作都要回滚，一个不可分割的工作单位，

事务只有表引擎为（innoDB）才有。

比如：银行转账工作，从一个帐号扣款并使另一个帐号增款，这两个操作要么都执行，要么都不执行。

事务是数据库维护数据一致性的单位，每个事务结束时，都能保持数据库的一致。

四大特性（简称ACID）:原子性（atomicity）

一致性（consistency）

隔离性（isolation）:两个事务操作同一行

数据，会上锁，后来

的等待阻塞。

持久性（Durability）

Mysql客户端也默认使用了事务，事务里面的操作执行完后，要调用

Commit（）函数进行提交，如果，中间产生问题，rollback（）进行回滚。只是平时使用sql语句，客户端自动调用了commit（）。

Python使用pymsql 执行数据增删改时，要调用commit（）默认开启了事务

事务启动： begin; 或者 start transaction；

中间 ： sql语句操作

提交事务（事务结束）： commit; 或 rollback;回滚事务

中间sql语句有失误，只要没有执行提交commit，可以进行回滚：rollback；

索引： 是一种特殊文件（innoBD数据表上的索引是表空间的一个组成部分），

他们包含着对数据表里所有记录的引用指针

通俗：索引就是书的目录，只是目录写的是所有记录的指针

索引目的：提高查询效率

开启运行时间检测：set profiling=1；

执行的sql语句

查看执行的时间：show profiles；

如果是字符串类型列名后带上长度，其他类型，直接写上列名

创建索引：create index 索引名 on 表名（列名或列名（长度））

为表创建索引，关联上的列名，where条件 就用这个字段列名

查看索引：show index from 表名; ----> 表名为索引关联的表

删除索引：drop index 索引名 on 表名；

主键和外键创建：也创建了该列名关联的索引

注意：一般情况，是你的表数据比较多，且那个字段经常要使用，才 为表的这个字段去建立索引，索引是要占据空间的。索引也不是建立越多越

好，建立多了，占空间，且会影响updat或insert的速度。

账户管理： mysql自带一个mysql数据库，在这里进行帐号管理；

使用mysql数据库：Use mysql

用户表：desc user；

主要字段说明：host 表示允许访问的主机

User 表示用户名

Authentication\_string 表示密码（加密）

创建用户和授权：数据库.表名（只能对这个该数据库表进行具有的权限操作）

grant 权限列表 on 数据库名.表名 to ‘用户名’@‘主机ip’ identified by ‘密码’;

权限列表：select uptade insert delete

all privileges(所有权限都能用)

数据库.\*:操作所有表

主机ip：%（允许用户用任何主机登入访问该mysql数据库）

Localhost （只能本地登入）

修改用户权限：

首先：grant 权限列表 on 数据库.表名 to ‘用户名’@‘主机ip’ with grant option;

最后：flush privileges； --->刷新权限

修改密码：

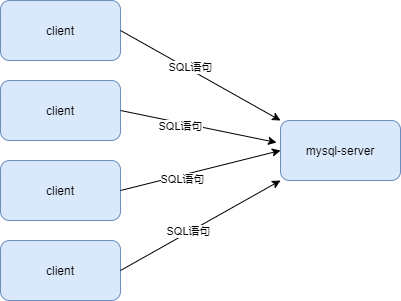
首先：update user set authentication\_string = password(‘新密码’) where user=‘用户名’

最后：flush privileges； --->刷新权限

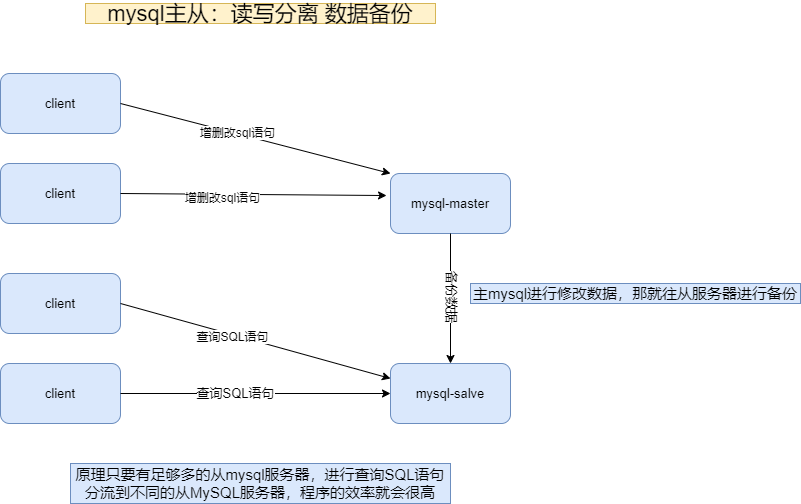
删除用户： drop user ‘用户名’@‘主机ip’；

Mysql 主从： 读写分离

原来：所有客户端都为一台mysql服务器操作



现在进行mysql主从：



配置主从同步的步骤：

1. 在主服务器上，必须开启二进制日志机制和配置一个独立的id
2. 在每一个从服务器上，配置一个唯一的id，创建一个用来专门复制主服务器数据的帐号
3. 在开始复制进程前，在主服务器上记录二进制文件的位置
4. 如果在开始复制之前，数据库中已经有数据，就必须先创建一个数据快照（可以使用mysqldump导出数据库，或者直接复制数据文件）
5. 配置从服务器要连接的主服务器的ip地址和登陆授权，二进制日志文件名和位置

Mysql数据备份：终端上执行命令 用户（root 需要权限）

备份：mysqldump -u用户 -p 数据库名 > xxx.sql

mysqldump -u用户 -p --all-databases --lock-all-tables > xxx.sql

--databases:所有数据库

--lock-all-tables：执行操作时锁住所有表，防止操作修改数据

恢复：

首先：连接上mysql，创建一个数据库，退出mysql。

最后：mysql -u用户 -p 创建的数据库名 < xxx.sql