笔记

1.原子性:数据库表中每一个列是不可分割的单位数据项,

2.主键:每一行的数据,有唯一的主键,非主键只能对主键依赖

3:外键

部署笔记

```
1.切记吧dockerfile单独复制出来
2.做镜像
docker build -t bbs .

docker run -d -p 5000:80 --name mybbs bbs
```

前端部署

```
worker_processes  1;
events {
    worker_connections  1024;
}


http {
    include       mime.types;
    default_type  application/octet-stream;


    sendfile        on;
    #tcp_nopush     on;

    #keepalive_timeout  0;
    keepalive_timeout  65;

    #gzip  on;

    server {
```

```
        listen       80;
        server_name  localhost;

        location / {
           root /data/;
           index index.html index.htm;
        }

        #error_page  404              /404.html;

        # redirect server error pages to the static page /50x.html
        #
        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   html;
        }


    }



}
```

```
先把上面的这个配置 /home/bbsui/nginx.cnf
在/home/bbsui/html/

docker run -d -p 8090:80  -v /home/nginx/:/var/log/nginx/ \
-v /home/bbsui/nginx.conf:/etc/nginx/nginx.conf -v /home/bbsui/html/:/data/ --name nginx  nginx

```

```
 动态api

userinfo  dal
userinfo service
   增删改查
 userinfo api
 增 删 改和查 也放出来,
```

```
 发现webapi 里面除了创建一个service 其他就调service

 get getuserinfo()
 {
var sercice=  new userservice();
 sercice.getuserinfo();
 }

 可以直接把sercice 变成webapi ;
```

mysql 读写分离

```
 ###主库

 docker run -d -p 3306:3306 -v
/home/mysql/conf:/etc/mysql/conf.d -v
/home/mysql/data:/var/lib/mysql -e
MYSQL_ROOT_PASSWORD=123456 --name mysql01 mysql:5.7

 # 进入之前的主库
docker exec -it mysql01 /bin/bash
cd /etc/mysql
# 安装vim
apt-get update
apt-get install vim
#修改配置
vim my.cnf
```

```
[mysqld]
## 同一局域网内注意要唯一
server-id=100
## 开启二进制日志功能，可以随便取（关键）
log-bin=master-bin
binlog-format=ROW

```

```
service mysql restart
docker restart mysql01
```

创建主数据库同步账号

```
docker exec -it mysql01 /bin/bash
mysql -uroot -p123456
CREATE USER 'slave'@'%' IDENTIFIED BY '123456';
GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'slave'@'%';
```

配置从库

```
 ###从库

  docker run -d -p 13306:3306 -e MYSQL_ROOT_PASSWORD=123456 --name mysql02 mysql:5.7


 把主库的数据库备份传输到从库

```

```
docker exec -it mysql02 /bin/bash
cd /etc/mysql
# 安装vim
apt-get update
apt-get install vim
#修改配置
vim my.cnf
```

```
[mysqld]
## 设置server_id,注意要唯一
server-id=101
## 开启二进制日志功能，以备Slave作为其它Slave的Master时使用
log-bin=mysql-slave-bin
## relay_log配置中继日志
relay_log=mysql-relay-bin
read_only=1
```

重新启动

```
service mysql restart
docker restart mysql02
```

开启Master-Slave主从复制

```
docker exec -it mysql01 /bin/bash
mysql -uroot -p123456

```

进入Master库mysql客户端：输入`show master status`查看Master状态：

```
show master status;
```

```
docker inspect --format='{{.NetworkSettings.IPAddress}}' mysql-master

docker exec -it mysql02 /bin/bash
mysql -uroot -p123456
# 设置同步
change master to master_host='192.168.3.204', master_user='slave', master_password='123456', master_port=3306, master_log_file='master-bin.000001', master_log_pos=617, master_connect_retry=30;
#验证
start slave;
show slave status \G
Slave_IO_Running 和 Slave_SQL_Running是查看主从是否运行的关键字段，默认为NO，表示没有进行主从复制。
使用start slave;开启主从复制过程，然后再次查询主从同步状态show slave status \G;
```

周边语法

```
#重置主从
stop SLAVE;
RESET SLAVE;
```

配置从库2

```
 ###从库
docker run -d -p 23306:3306 -e MYSQL_ROOT_PASSWORD=123456 --name mysql03 mysql:5.7
#########把主库的数据库备份传输到从库

```

```
docker exec -it mysql03 /bin/bash
cd /etc/mysql
# 安装vim
apt-get update
apt-get install vim
#修改配置
vim my.cnf
```

```
[mysqld]
## 设置server_id,注意要唯一
server-id=102
## 开启二进制日志功能，以备Slave作为其它Slave的Master时使用
log-bin=mysql-slave-bin
## relay_log配置中继日志
relay_log=mysql-relay-bin
read_only=1
```

重新启动

```
service mysql restart
docker restart mysql03
```

```

docker exec -it mysql03 /bin/bash
mysql -uroot -p123456
# 设置同步 注意 偏移量
change master to master_host='192.168.3.204',
master_user='slave', master_password='123456',
master_port=3306, master_log_file='master-bin.000001',
master_log_pos=7198, master_connect_retry=30;
#验证
start slave;
show slave status \G
Slave_IO_Running 和 Slave_SQL_Running是查看主从是否运行的关键字段，默认为NO，表示没有进行主从复制。
使用start slave;开启主从复制过程，然后再次查询主从同步状态show slave status \G;
```

分区处理

```
 show plugins;

 ALTER TABLE PostReplys
  partition by range(PostId)
  (
    partition p1 values less than (100000),
```

```
     partition p2 values less than (200000),
    partition p3 values less than maxvalue
  );

  ALTER TABLE PostReplys
  partition by range(PostId)
  (
    partition p1 values less than (100000),
     partition p2 values less than (200000),
    partition p3 values less than maxvalue
  );

select PARTITION_NAME as "分区",TABLE_ROWS as "行数"
from information_schema.partitions where
 table_name="PostReplys";

CREATE UNIQUE  INDEX index_replys ON PostReplys
(id,PostId)

EXPLAIN SELECT * from PostReplys   where id=1

SHOW INDEX FROM PostReplys

SHOW INDEX FROM Posts


```

字段修改

```
alter table Posts alter column Up set default 0;
alter table Posts alter column Down set default 0;

update Posts set Up=0 where Up is null;
update Posts set Down=0 where Down is null;


alter table PostReplys alter column Up set default 0;
alter table PostReplys alter column Down set default
0;

update PostReplys set Up=0 where Up is null;
update PostReplys set Down=0 where Down is null;
```

问题的本质

帖子或者回复的内容id

用户id ,一个用户对于一个postid,要么是点赞,要么是点踩 互斥的

set

一个postid 对应两个set ,一个点踩,一个是点赞

多次赞只会存储一个相同userid , 多次踩同样的道理

set 太复杂--不行换,大道至简

zset ---怎么使用zset

还差点???

之前是一个postid 对应了两个 set

postid 对应一个zset id

postid

userid　(1是顶,-1是踩)

通过最大值,和最小能查询count(0,1) 这边不就是所有

count(-1,0) 踩数,count(0,1)

,我需要判断

需要查看我们的当前的点赞数量和点踩数量,

写日志

第一种,些本地,挂载logstash ,异步收集...

第二种,es 直接写规范的日志

发消息,然后消费端,消费日志

1

10000000000000000000

压缩--- 分块 -- zset hash 尽量让postid score (大大大大)

postid score (大大大大)切片

hash(postid)%50 就是50个分块

还有一种--- 算法 post +score 从大值变小值,但是不影响排序功能

1 100000 1

2 98000 0.98

可以吧大值变小值,但是不能回来---

要刷盘,吧redis数据刷到mysql

分批刷盘,分页---

假设用一个

postid

userid@1

userid@0

后面统计-- 所有的拿出来,然后在自己的代码中判断...XXX 不行

postid赞	postid踩
userid	userid