注意:若要修改 MySQL 服务的默认字符集,可以更改服务器的 my.cnf 配置文件,添加

character\_set\_server=utf8 配置,然后重启数据库服务。

create databases tts **character set utf8mb4**;

什么是数据:

视频,音频,图片,文本

专业术语:

DB (DataBase) 数据库

DBMS (DataBase Management System) 数据库管理系统

--用来操纵和管理数据库的服务软件

DBS (DataBase System) 数据库系统

--数据库系统:即 DB + DBMS

--指安装了数据库并整合了数据库软件的计算机系统

关系型数据库系统

oracle

mysql

sql server

db2

nosql: redis mencache mongodb

LANG=C

部署mysql数据库服务

1. 解包
2. 安装软件包
3. 启动mysql服务
4. 查看服务进程和端口
5. 查看初始密码
6. 使用初始密码登录
7. 修改登录密码
8. 断开连接,退出数据库
9. 使用新密码登录
10. 修改密码策略
11. 根据新密码策略设置密码
12. 使用新密码登录

设置密码

grant select on Contacts.\* to Raikon@localhost identified by‘atenorth’;

过滤初始密码

grep "password" /var/log/mysqld.log

2019-06-06T01:59:05.924348Z 1 [Note] A temporary password is generated for root@localhost: B5XgQLFAt;dm

用初始密码登录

mysql -hlocalhost -uroot -p'B5XgQLFAt;dm'

修改密码

alter user root@"localhost" identified by "123qqq...A";

修改密码策略

查看变量

mysql> show variables like "%password%";

修改密码策略

mysql> set global validate\_password\_policy=0;

修改密码长度

mysql> set global validate\_password\_length=6;

修改新密码

mysql> alter user root@"localhost" identified by "tarena";

修改配置文件 /etc/my.cny

[mysqld]

validate\_password\_policy=0

validate\_password\_length=6

连接MYSQL服务

连接方式

\*客户端连接MySQL服务的方法

---命令行

---web页面

---安装图形软件

---编写脚本(php java python...)

\*使用mysql命令

--mysql -h服务器IP -u用户名 -p密码 [数据库名]

--quit exit 退出

数据存储流程

客户端把数据存储到数据库服务器上的步骤

1,连接数据库服务器

2,建库 //类似与文件夹

3,建表 //类似于文件

4,插入记录 //类似与文件内容

5,断开连接

mysql管理环境

\*SQL命令使用规则

SQL命令不区分字母大小写(密码 变量值除外)

每条sql以; 结尾

\c 终止sql命令

\*常用的SQL命令分类

--管理数据库使用SQL(结构化查询语言)

1,DDL 数据定义语言 如:create alter drop

2,DML 数据操作语言 如:insert update delete

3,DCL 数据控制语言 如:grant 授予 revoke 撤销

4,DTL 数据事物语言 如:commit rollback savepoint

库管理命令

仅可以使用数字、字母、下划线、不能纯数字

区分字母大小写,具有唯一性

不可使用指令关键字、特殊字符

show full processlist; //显示连接用户

select user(); //显示连接用户

show database; //显示已有的库

use 库名; //切换库

select database(); //显示当前所在的库

create database 库名; //创建新库

show tables; //显示已有的表

drop database 库名; //删除库

表管理命令

1.建表

create table 库名.表名(字段名1 类型(宽度),

字段名2 类型(宽度),

...

)DEFAULT CHARSET=utf8; 指定中文字符集,可以给字段赋值中文

2.查看表结构

desc 库名.表明;

3.删除表

drop table 库名.表名;

4.记录管理命令

记录 类似于文件里的行

查看表记录

select \* from 库名.表名; //查看表记录

insert into 库名.表名 values(值列表); //插入表记录

mysql> insert into aaa values ("1","tom","2","boy","sleep");

一,常见信息种类

信息种类:

**数值型**: 体重、身高、成绩、工资

**字符型**:姓名、工作单位、通信地址

**枚举型**:兴趣爱好、性别、专业

**日期时间型**:出生日期、注册时间

***1,***字符类型

1)定长: char (字符个数)

最大字符个数255

不够指定字符个数时在右边用空格补全

字符个数超出时,无法写入数据

create table db1.t2 (

name char(5),

homedir char(50)

);

2)变长varchar(字符个数)

按数据实际大小分配存储空间

字符个数超出时,无法写入数据

大文本类型 : text / blob

字符数大于65535存储使用

create table db1.t2 (

name char(5),

email varchar(30)

);

***2,***数值类型

整数型

--只能存储整数



**tinyint smallint mediumint int bigint unsigned**

浮点型



**float double**

create table db1.t3(pay float,id double);

create table db1.t4(pay float(5,2),id double (4,2));

**3,日期时间类型**

日期时间: **datetime**

范围: 1000-01-01 00:00:00 ~ 9999-12-31 23:59:59

格式:yyyymmddhhmmss

日期时间: **timestamp** (未赋值时自动以当前系统时间赋值)

范围:1970-01-01 00:00:00 ~ 2083-01-19 00:00:00

格式:yyyymmddhhmmss

create table db1.t9(

meetting datetime,

party timestamp

) ;

mysql> insert into db1.t9 values (20200711094000,20200711095000);

日期 date

范围:0001-01-01 ~ 9999-12-31

格式:yyyymmdd

年 year

范围: 1901 ~ 2155

格式: yyyy

01~69 视为2001~2069

70~99 视为 1970~1999

时间 time

格式 HH:MM:SS

时间函数

可以使用时间函数给字段赋值



**4,枚举类型**

1) 单选enum

仅能选择一个值,字段值必须在列表选择

create table db1.t5(

name char(5),

sex enum(“boy”,”girl”,”no”)

);

1. 多选set

选择一个或多个值,字段值必须在列表选择

create table db1.t5(

name char(5),

likes set (“eat”,”game”,”music”,”money”)

);

**day02**

2.1约束条件

作用:限制给表字段赋值

查看字段的约束条件: desc 库名.表名;

约束条件包括那些:

null 允许为空(默认设置)

not null 不允许为null(空)

key 键值类型

default 设置默认值,缺省为NULL

extra 额外设置

2.2修改表结构

命令格式 table 库.表 执行动作

add 添加新字段

drop 删除已有字段

modify 修改字段类型

change 修改字段名

rename 修改表名

添加新字段add

alter table db1.t10 add email varchar(50) default "stu@tedu.cn";

alter table db1.t10 add stu\_num char(9) **first**;

alter table db1.t10 add age tinyint unsigned not null default 19 **after name**;

删除已有字段drop

alter table db1.t10 drop email, drop stu\_num;

修改字段类型modify

alter table db1.t10 modify name varchar(15) default **""**;

alter table db1.t10 modify age tinyint unsigned not null default 19 after sex;

修改字段名change

alter table db1.t10 change likes **loves** set('eat','game','piao','du','film')

修改表名rename

1.

alter table db1.t10 rename db1.**classinfo**;

RENAME TABLE `haha`.`llll` TO `haha`.`ooo` ;

2.

mysql> rename table a to aa;

二、MySQL键值

1.键值类型:

普通索引 index

唯一索引 unique

主 键 primary key \*\*\*

外 键 foreign key \*\*\*

全文索引 fulltext

2.作用:约束如何给字段赋值

3.什么是索引(索引介绍)

类似于书的目录,对表中的字段进行排序,

索引类型包括:Btree、B+tree(二叉树)、hash

1. 为什么要使用索引(索引的优缺点)

优点:通过创建唯一性索引,

可以保证数据库表中每一行数据的唯一性,

可以加快数据的查询速度

缺点:当对表中的数据进行增加、删除和修改的时候,

索引也要动态的调整,降低了数据的维护速度

索引需要占用物理空间

5.使用索引:

使用规则:

一个表中可以有多个index字段

字段的值允许重复,且可以赋NULL值

通常把做为查询条件的字段设置为index字段

index字段标志是MUL

1.)查看:

desc 库.表 key

show index from 表名 \G;

create table t3(

name char(10),

age tinyint,

sex enum("m","w"),

index(name),index(age)

);

mysql> desc t3;

mysql> show index from t3\G;

1. )创建:

(1).新建索引

create table db1.t22(

class char(9),

name char(15),

age int,

index(name),index(age)

);

desc db1.t22;mysql> desc db1.t22;

+------- +----------+-----------+-----+-----------+-------+

| Field | Type | Null | Key | Default | Extra |

+------- +-------------+----------+-----+-------------+-------+

| class | char(9) | YES | | NULL | |

| name | char(15) | YES |**MUL** | NULL | |

| age | int(11) | YES | **MUL** | NULL | |

+-------+--------------- +------+-----+--------------+-------+

3 rows in set (0.00 sec)

show index from db1.t22\G;

mysql> show index from db1.t22\G;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1. row \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Table: t22

Non\_unique: 1

**Key\_name: name //索引名**

Seq\_in\_index: 1

**Column\_name: name //默认索引名和字段名同名**

Collation: A

Cardinality: 0

Sub\_part: NULL

Packed: NULL

Null: YES

Index\_type: BTREE

Comment:

Index\_comment:

(2.)在已有的表里创建索引

添加索引字段

create index xixi on stuinfo(name);

1. )删除索引字段:

drop index 索引名 on 表名;

先查看索引名

desc t22;

show index from t22\G;

再删除

drop index xixi on t22;

三、primary key主键

1.使用规则

字段值不允许重复,且不允许赋NULL值

一个表中只能有一个primary key字段

多个字段都作为主键,称为复合主键,必须一起创建.

**主键字段的标志是PRI**

通常把表中唯一标识记录的字段设置为主键

[记录编号字段]

2.创建复合主键

mysql> create table db1.t24( **clientip** char(15),

**serport** int, status enum("yes","no"),

primary key(**clientip,serport**));

mysql> desc db1.t24;

mysql> insert into db1.t24 values ("1.1.1.1",22,"no");

mysql> insert into db1.t24 values ("1.1.1.2",25,"no");

mysql> insert into db1.t24 values ("1.1.1.2",25,"no");

mysql> select \* from db1.t24;

3.与**auto\_increment**连用

自增长

赋值时 字段的值字段加1

mysql> create table db1.t25(

**id int primary key auto\_increment**,

name char(15),

age tinyint,

sex enum("boy","girl")

);

mysql> desc db1.t25;

mysql> insert into db1.t25(name,age,sex) values("bob",19,"boy");

mysql> insert into db1.t25(name,age,sex) values("tom",19,"boy");

mysql> insert into db1.t25(name,age,sex) values("jerr",20,"boy");

mysql> select \* from db1.t25;

+----+------+------+------+

| id | name | age | sex |

+----+------+------+------+

|  **1** | bob | 19 | boy |

| **2** | tom | 19 | boy |

|  **3** | jerr | 20 | boy |

+----+------+------+------+

3 rows in set (0.00 sec)

primary key主键

id字段会自动增长,

id也可以自己赋值,

mysql> insert into db1.t25 values (6,"bob",19,"girl");

id会从最大值开始增长

1. 删除primary key主键.

必须同时删除所有复合主键

有**auto\_increment**自增长时,先修改为空,再删除

**mysql> alter table db1.t24 drop primary key;**

mysql> alter table db1.t25 modify id int;

mysql> alter table db1.t25 drop primary key;

1. foreign key 外键

外键作用:

限制给字段赋值

插入记录时,字段值在另一个表字段值范围内选择

**使用规则**

1.表存储引擎必须是innodb

2.字段类型要一致

3.被参照字段必须要是索引类型的一种(通常是主键primary key)

**创建外键**

create table 表名(

字段名列表,

**foreign key**(字段名) **references**  表名(字段名) //指定外键

**on update cascade** //同步更新

**on delete cascade** //同步删除

)**engine=innodb**; //指定存储引擎

创建员工表(yg\_id,name)

mysql> create table db1.yg( yg\_id int **primary key auto\_increment**,

name char(20)

)**engine=innodb**;

创建工资表

mysql>create table db1.gz(

->gz\_id int,

-> pay float(7,2),

-> foreign key(**gz\_id**) references **yg(yg\_id)**

-> on update cascade

-> on delete cascade

-> )engine=innodb;

mysql> desc db1.gz;

mysql> show create table db1.gz\G;

修改或删除员工表yg\_id 工资表gz\_id会一起改变

mysql> update db1.yg set yg\_id=4 where yg\_id=3;

mysql> select \* from db1.yg;

mysql> select \* from db1.gz;

mysql> delete from db1.yg where yg\_id=1;

mysql> select \* from db1.yg;

mysql> select \* from db1.gz;

删除

mysql> delete from db1.gz where pay=5000;

显示创建表的命令

mysql> **show create table gzb\G;**

删除外键

先查看创建表命令找出外键名

**CONSTRAINT `gzb\_ibfk\_1`**

gzb\_ibfk\_1

**alter table gzb drop foreign key gzb\_ibfk\_1;**

Day03

一、数据导入导出(导入完数据后再设置id自增长值)

1.修改导入导出默认文件检索路径

mkdir /myload

chown mysql /myload

vim /etc/my.cnf

secure\_file\_priv="/myload"

systemctl restart mysld

查看

show variables like “secure\_file\_priv”;

查看是否修改为 “/myload”

+--------------------------+----------------+

| Variable\_name | Value |

+--------------------------+----------------+

| secure\_file\_priv | /myload/ |

+--------------------------+----------------+

2,作用:

把系统文件的内容存储到数据的表里

3,命令格式:

mysql> load data infile “目录名/文件名”

into table 库名.表名

fields terminated by “分隔符”

lines terminated by “\n”;

创建表(根据文件的内容创建表格,**看PPT注意事项**)

mysql> create table db3.user (

name char(50),

password char(1),

uid int,

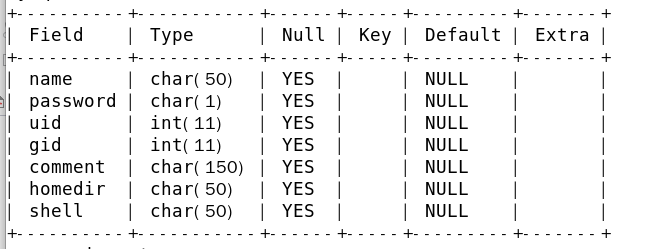
gid int,

comment char(150),

homedir char(50),

shell char(50));

mysql> desc db3.user;



拷贝/etc/password到/myload下

第一种:在命令行输入

[root@mysql1 ~]# cp /etc/passwd /myload/

第二种:在数据库输入

mysql> **system** cp /etc/passwd /myload

mysql> system ls /myload

passwd

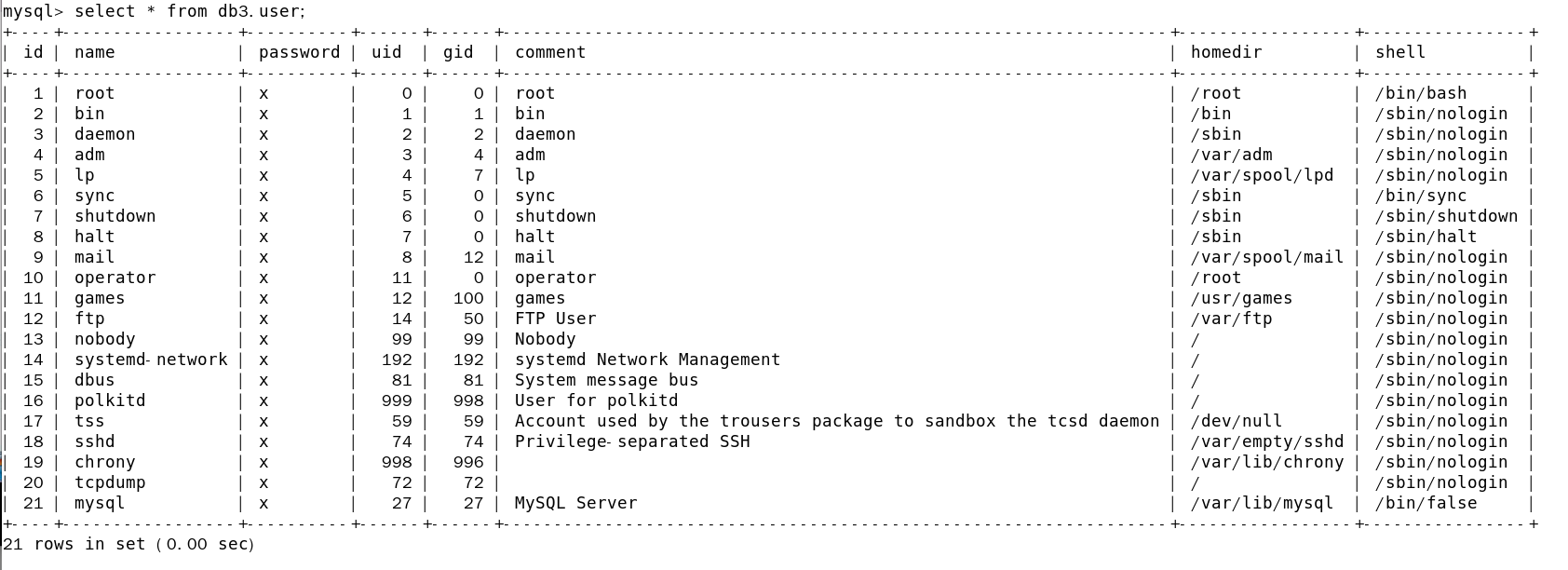
导入数据

mysql> **load data infile** "/myload/passwd" **into table** db3.user **fields terminated by ":" lines terminated by "**\n**";**

添加id行号

mysql> alter table db3.user

add id int primary key auto\_increment first;



4,数据导出:

格式1

select命令 into outfile “目录名/文件名”;

mysql> **select \* from db3.user into outfile "/myload/user1.txt";**

格式2

select命令 into outfile “目录名/文件名”;

fields terminated by “分隔符”;

mysql> **select \* from db3.user where id<=3 into outfile "/myload/user2.txt" fields terminated by "###";**

格式3

select命令 into outfile “目录名/文件名”;

fields terminated by “分隔符”;

lines terminated by “\n”; (\n:换行符)

mysql> **select name,uid,shell from db3.user where id<=3 into outfile "/myload/user3.txt" fields terminated by "#" lines terminated by "???";**

管理表记录 (db3.user)

1,增加表记录

格式1:添加一条记录,给所有字段赋值

insert into 表名 values (字段值列表);

mysql> insert into user values

(22,"bob","x",3000,3000,"test user","/home/bob","/bin/bash");

格式2:添加N条记录,给所有字段赋值

insert into 表名

values

(字段值列表),(字段值列表),(字段值列表);

mysql> insert into user values

(25,"lucy","x",3001,3001,"test user","/home/bob","/bin/bash"),

(26,"tom","x",3002,3002,"test user","/home/bob","/bin/bash"),

(27,"jack","x",3003,3003,"test user","/home/bob","/bin/bash");

格式3:添加一条记录,给指定字段赋值

insert into 表名(字段名列表) values (字段值列表);

mysql> insert into user (name,uid) values

("jerry","4000");

格式4:添加N条记录,给指定字段赋值

insert into表名 (字段名列表)

values

(字段值列表),

(字段值列表),

(字段值列表);

mysql> insert into user(name,uid) values

("mack","4001"),("lili","4002"),("dachui","4003");

注意事项:

查询记录

格式1: 查看所有记录

select 字段1, .. .., 字段N from 库名.表名;

mysql> select id,name,shell from user;

格式2:条件查询

select 字段1, .. .., 字段N from 库名.表名;

where 条件表达式;

mysql> select \* from user where name=bob;

mysql> select id from user where name=bob;

mysql> select \* from user where id<=5; //列出前五行

更新记录表

格式1:批量更新

update 库名.表名

set 字段名=值,字段名=值,字段名=值,......;

mysql> update user set password="x",comment=”student user”;

格式2:条件匹配更新

update 库名.表名

set 字段名=值,字段名=值,字段名=值,......;

where 条件表达式;

mysql> select id,name,password from user where id<=5;

删除记录

格式1:条件匹配删除内容

delete from 库名.表名 where 条件表达式;

mysql> delete from user where name="dachui";

格式2:删除所有记录

delete from 库名.表名;

mysql>delete from user;

匹配条件

3.1 基本匹配条件(适用于select \ updata \ delete)

3.1.1数值比较

3.1.2字符比较

3.1.3范围内比较

3.1.4 逻辑比较

3.1.5空\非空

3.1.6 去掉重复数据

3.2 高级匹配条件(适用于select \ updata \ delete)

3.2.1 模糊匹配

3.2.2 正则匹配

3.2.3 四则匹配

3.2.4 提高优先级()

3.3 操作查询结果(适用于select)

3.1.1数值比较

字段必须是数值类型

= > >= < <= !=

mysql> select \* from user where uid=3;

mysql> select \* from user where id=5;

mysql> select name,uid,gid from user where uid=gid;

mysql> select name,uid,gid from user where uid!=gid;

3.1.2字符比较

= !=

mysql> select name,shell from user where shell !="/bin/bash";

mysql> select name,shell from user where name ="root";

3.1.3范围内比较

类型 比较

in(值列表) 在...里...

not in (值列表) 不在...里..

between数字and数字 在...之间..

distinct字段名 去重显示

mysql> select name from user where

-> name **in** ("mysql","apache","adm","sync");

mysql> select name,uid from user

-> where uid in (3,5,6,7,8);

mysql> select name,shell from user

-> where shell **not in**("/bin/bash","/sbin/nologin");

mysql> select name,uid,shell from user

-> where uid **between** 50 and 100;

3.1.4 逻辑比较

逻辑匹配

多个判断条件时使用

逻辑与 and 或 && //多个判断条件必须同时成立

逻辑或 or 或 || //多个判断条件,某个成立即可

逻辑非 ! 或 not //

mysql> select \* from user

-> where

-> name="root" and uid=0 and shell="/bin/bash";

mysql> select name,shell from user where name ="root";

mysql> select \* from user where

name="root" && uid=0 && shell="/bin/bash";

-\*-\*----------\*--------\*-

3.1.5空\非空

is null

is not null

mysql> insert into user(name) values(null);

mysql> select id from user where name is null;

mysql> select id from user where name is not null;

3.1.6 去掉重复数据**distinct**

mysql> select shell from user where uid>=100;

mysql> **select distinct shell from user where uid>=100;**

mysql> select **distinct** shell from user;

3.2.1 模糊匹配

where 字段名 **like**  ‘通配符’

\_ 表示1个字符

% 表示0~n个字符

匹配名字为四个字符

mysql> select name from user where name **like '\_ \_ \_ \_';**

mysql> select name from user where name **like '\_\_\_\_';**

匹配名字四个字符以上

mysql> select name from user where name **like '\_ \_ %\_ \_';**

匹配以a开头的名字

mysql> select name from user where name **like 'a%';**

3.2.2 正则匹配

where 字段名  **regexp** ‘正则表达式’

正则元字符 ^ $ .(一个字符) [] \*(任意次数) |

***以r开头或者以y结尾的中间任意多个字符***

mysql> select name from user where name **regexp '^r.\*y$';**

***uid 为四个字符的***

mysql> select name,uid from user where uid **regexp '^....$';**

***匹配名字中含有数字的***

mysql> insert into user(name) values ("yaya9"),("7yaya"),("ya8ya")

mysql> select name,uid from user where name **regexp '[0-9]';**

3.2.3 四则匹配 + - \* / %

uid和gid 相加

mysql> select name,uid,gid, uid+gid zonghe from user

where name="bin";

uid和gid 相加除2(会保留四位小数)

mysql> select name,uid,gid, (uid+gid)/2 zonghe from user

where name="bin";

gid加1

mysql> update user set gid=gid+1;

设置年龄(age)默认值19

mysql> alter table user add age tinyint unsigned default 19 after name;

计算root出生年份(2019-age)

mysql> select name,age, 2019-age chusheng from user

where name="root";

3.2.4 提高优先级 **()**

mysql> **select name,uid,gid, (uid+gid)/2 zonghe from user**

**where name="bin";**

3.3 操作查询结果(适用于select)

MySQL内置数据统计函数

**avg(字段名) //统计字段平均值**

**sum(字段名) //统计字段之和**

**min(字段名) //统计字段最小值**

**max(字段名) //统计字段最大值**

**count(字段名) //统计字段值个数**

3.3.1聚集函数

mysql> select min(uid) from user;

3.3.2查询结果排序(**默认升序排序**)

SQL查询 order by 字段名 [ asc | desc ];

**asc 升序排序**

**desc 将序排序**

mysql> select name,uid from user

-> where uid >=10 and uid <=500 order by uid;

mysql> select name,uid from user

-> where uid >=10 and uid <=500 order by uid **desc**;

3.3.2查询结果分组

SQL查询 group by 字段名;

mysql> select shell from user group by shell;

mysql> select shell from user where uid <=500;

mysql> select shell from user where uid <=500 **group by shell**;

3.3.3查询结果过滤

SQL查询  **having** 条件表达式;

mysql> select name from user where id<=20;

过滤id小于20的adm用户

mysql> select name from user where id<=20 having name="adm";

3.3.4限制查询结果显示行数

SQL 查询 limit 数字;

SQL 查询 limit 数字1,数字2;

**起始行是0,如果输出第二行就是1,第三行2表示**

显示user表id,名字,密码,前三行

mysql> select id,name,password from user where id<=20 limit 3;

显示user表id,名字,密码,从第三行开始的三行;

mysql> select id,name,password from user where id<=20 **limit 2,3;**

4常见的MySql管理工具



真机拷软件包

[student@room9pc01mysql]$ scp phpMyAdmin-2.11.11-all-languages.tar.gz root@192.168.4.50:/root/

day04

一,用户授权:在数据库服务器上添加新的连接账户,默认只允许数据库管理员本机登陆

mysql -h192.168.4.50 -ubob -p123

grant授权

\*授权:添加用户并设置权限

命令格式

grant 授权列表 on 库名 to 用户名@”客户端地址”

identified by “密码”

with grant option; // 有授权权限,可选项

grant all on db4.\* to yaya@”%” identified by “123”;

all 给所有权限

\* db4库下的所有表

% 所有网段

权限列表

all 所有权限

usage 无权限

select,update,insert 个别权限

select,update(字段1,.. ..,字段N) 指定字段

库名

mysql> grant all on \*.\* to mydba @"%" identified by "123qqq...A" with grant option;

**登录用户使用相关命令**

查看主机名

mysql> select @@hostname;

查看当前登录的用户

mysql> select user();

查看访问权限

mysql> show grants;

mysql> desc mysql.db;

mysql> select user, host from mysql.user;

mysql.db表,

mysql> select user,host,db,select\_priv from mysql.db;

查看mysql.db表,

mysql> select host,db,user from mysql.db;

查看用户one,库(db)是db4de,

mysql> select \* from mysql.db where db="db4" and host='localhost' and user="one"\G;

mysql> update mysql.db set Delete\_priv="N" where user="one" and host="localhost" and db="db4";

授权用户连接后修改密码

mysql> set password=password("654321");

管理员重置授权用户连接密码

mysql> set password for mydba@"%"=password("123456")

管理员查看有哪些用户

mysql> select user,host from mysql.user;

管理员查询用户权限

mysql> show grants for mydba@"%";

mysql> drop user one@"%";

刷新

mysql> flush privileges;

1.6撤销权限

删除用户的权限

命令: revoke 权限列表 on 库名.表名 from 用户名@”客户端地址”;

1,撤销授权权限(使用户不能)

撤销----------->WITH GRANT OPTION权限

mysql> select user,host from mysql.user;

mysql> show grants for mydba@"%";

mysql> revoke grant option on \*.\* from mydba@"%";

mysql> show grants for mydba@"%";

2,撤销对数据的访问权限

mysql> revoke delete,update on \*.\* from mydba@"%";

mysql> show grants for mydba@"%";

1.7 root密码

1.7.1 恢复数据库管理员本机登录密码

1,停止Mysql服务程序

2,跳过授权表启动mysql服务程序

3,修改root密码

4,正常方式启动mysql服务程序

2, vim /etc/my.cnf

[mysqld]

skip-grant-tables

secure\_file\_priv="/xing"

#validate\_password\_policy=0

#validate\_password\_length=6

]# systemctl restart mysqld

3, mysql> update mysql.user

set authentication\_string=password("123qqq...A")

where user="root" and host="localhost";

mysql> flush privileges;

4, vim /etc/my.cnf

[mysqld]

skip-grant-tables

secure\_file\_priv="/xing"

#validate\_password\_policy=0

#validate\_password\_length=6

]# systemctl restart mysqld

5, [root@mysql1 ~]# mysql -uroot -p123qqq...A

1.7.2重置数据库root密码 (需要知道操作系统的密码)

[root@mysql1 ~]# mysqladmin -uroot -p password "123456" 新密码

Enter password: 输入旧密码

mysqladmin: [Warning] Using a password on the command line interface can be insecure.

Warning: Since password will be sent to server in plain text, use ssl connection to ensure password safety.

二,数据完全备份

2.1数据备份相关概念

恢复数据库管理员密码

重置数据库管理员密码

物理备份50

[root@mysql1 ~]# cp -r /var/lib/mysql /root/mysql.bak

[root@mysql1 ~]# scp -r /root/mysql.bak [root@192.168.4.51:/tmp/](mailto:root@192.168.4.51:/tmp/)

物理恢复51

systemctl stop mysqld

ls /var/lib/mysql

rm -rf /var/lib/mysql

cp -r /tmp/mysql.bak/ /var/lib/mysql

ls /var/lib/mysql

ls -ld /var/lib/mysql

chown -R mysql:mysql /var/lib/mysql

ls -ld /var/lib/mysql

ls -l /var/lib/mysql

systemctl restart mysqld

mysql -uroot -p123456

mysql> show databases;

mysql> select \* from db3.user;

mysql> use db1

mysql> show tables;

数据备份策略

1,完全备份

备份所有数据

192.168.4.50

mysqldump -uroot -p123456 -A > allbak.sql

ls \*.sql

mysqldump -uroot -p123456 db3 > db3.sql

vim db3.sql

mysqldump -uroot -p123456 -B db2 db3 > db3.sql

scp db3.sql root@192.168.4.51:/tmp/

scp db3.sql root@192.168.4.52:/tmp/

2,完全恢复

命令

mysql -uroot -p密码 [库名] < 目录/xxx.sql

192.168.4.52

mysql> create database db3;

]# mysql -uroot -p密码 db3 < /tmp/db3.sql

mysql> select \* from db3.user;

备份所有新产生的数据:

2,增量备份

备份上次备份后的数据

2.1 binlog日志的使用

[root@mysql1 ~]# vim /etc/my.cnf

[mysqld]

log-bin

server\_id=50

[root@mysql1 ~]# systemctl restart mysqld

[root@mysql1 ~]# ls /var/lib/mysql/\*-bin.\*

2.2启用日志

手动生成新的日志文件：(**四种方法**)

]# mysqldump -uroot -p123456 --flush-logs db3 > /dbbak/db3.sql

]# mysql -uroot -p123456 -e "flush logs"

]# systemctl restart mysqld

mysql> flush logs

2.3清理日志

删除binlog日志

1.删除50-bin.000004之前的日志文件 (001----004都没了)

mysql> purge master logs to "50-bin.000004";

2.全删

mysql> reset master;

2,4自定义日志存储目录及日志文件名

]# mkdir /mylog

]# chown mysql /mylog

]# vim /etc/my.cnf

[mysqld]

server\_id=50

log-bin=/mylog/plj

:wq

]# systemctl restart mysqld

]# ls /mylog

mysql> show master status;

2.5使用binlog日志恢复数据

格式: ]# mysqlbinlog binlog日志文件 | mysql -uroot -p密码

192.168.4.50

mysql> show master status;

mysql> create database db6;

mysql> create table db6.t1(id int);

mysql> insert into db6.t1 values(99999);

mysql> insert into db6.t1 values(99998);

mysql> insert into db6.t1 values(99997);

mysql> select \* from db6.t1;

mysql> show master status;

]# scp /mylog/plj.000001 root@192.168.4.51:/root/

192.168.4.51

]# mysqlbinlog /root/plj.000001 | mysql -uroot -pabc123

mysql> select \* from db6.t1;

3,差异备份

备份完全备份后的新产生的数据

4,修改binlog日志文件记录sql命令的格式：

查看默认的记录格式：

mysql> show variables like "%binlog%";

mysql> show variables like "binlog\_format";

修改binlog日志格式

]# vim /etc/my.cnf

[mysqld]

binlog\_format="mixed"

:wq

]# systemctl restart mysqld

mysql> show variables like "binlog\_format";

mysql> show master status;

mysql> insert into db6.t1 values(123);

mysql> insert into db6.t1 values(456);

mysql> insert into db6.t1 values(789);

mysql> insert into db6.t1 values(1010);

mysql> select \* from db6.t1;

mysql> delete from db6.t1 where id>1000;

mysql> delete from db6.t1;

mysql> drop db6.t1;

]# mysqlbinlog /mylog/plj.000002

]# mysqlbinlog /mylog/plj.000002 | grep -i insert

]# mysqlbinlog /mylog/plj.000002 | grep -i delete

]# mysqlbinlog /mylog/plj.000002 | grep -i drop

日志文件如何区分记录的多条sql命令 ? 偏移量 和 命令执行时间

格式: ]# mysqlbinlog 选项 binlog日志文件 | mysql -uroot -p密码

--start-position=数字 //起始偏移量

--stop-position=数字 //结束偏移量

--start-datetime="yyyy-mm-dd hh:mm:ss" //起始时间

--stop-datetime="yyyy-mm-dd hh:mm:ss" //结束时间

192.168.4.50 ]# scp /mylog/plj.000002 root@192.168.4.51:/root/

使用binlog日志文件恢复部分数据

格式: ]# mysqlbinlog 选项 binlog日志文件 | mysql -uroot -p密码

]# mysqlbinlog /mylog/plj.000002

192.168.4.51]#

mysqlbinlog --start-position=293 --stop-position=1724 /root/plj.000002 | mysql -uroot -pabc123

192.168.4.51 mysql> select \* from db6.t1 ;

**day05**

物理备份 逻辑备份 mysqldump binlog

**一、percona介绍**

**linux系统与widows系统文件系统不同,无法跨平台备份**

环境准备： 把50主机的存储引擎 修改为innodb

删除所有的自定义库 只保留初始的4个库

准备软件包

libev-4.15-1.el6.rf.x86\_64.rpm

percona-xtrabackup-24-2.4.7-1.el7.x86\_64.rpm

1. 安装percona软件192.168.4.50/51:

]# rpm -ivh libev-4.15-1.el6.rf.x86\_64.rpm

]# yum -y install percona-xtrabackup-24-2.4.7-1.el7.x86\_64.rpm

]# rpm -ql percona-xtrabackup-24

]# innobackupex --help

]# man innobackupex

innobackupex命令格式： innobackupex <选项>

常用选项：PPT上

显示创建表的命令

mysql> show create table db3.user\G;

.......

ENGINE=InnoDB AUTO\_INCREMENT=40 DEFAULT CHARSET=latin1

(innoDB存储引擎)

显示数据库软件支持有那些存储引擎;

mysql> show engines;

yes支持no不支持

2.1

**完全备份**

没有指定库默认备份所有库, --databases

没有指定主机名默认本地主机, --host

没有指定端口号,默认3306, --port

不用系统时间命名文件存储的目录名 --no-timestamp

[root@50 ~]# innobackupex --user root --password 123456

/allback 用系统时间命名文件存储的目录名

[root@50 ~]# innobackupex --user root --password 123456

/allback --no-timestamp 不用系统时间命名文件存储的目录名

**完全恢复**

1,把需要恢复的数据拷到需要恢复数据的主机上

2,stop mysqld服务,在需要恢复数据的主机上

3,清空/var/lib/mysql/\*

4,准备恢复数据

5,恢复数据

6,递归设置所有者,所属组为mysql

7,启动服务mysqld

把50主机数据拷贝到51主机上(**51主机密码也是50主机的**)

1. [root@50 ~]# scp -r /allback/ root@192.168.4.51:/root

[root@51 ~]# cat /root/allback/xtrabackup\_checkpoints

2. [root@51 ~]# systemctl stop mysqld

3. [root@51 ~]# rm -rf /var/lib/mysql/\*

[root@51 ~]# ls /var/lib/mysql/

4. [root@51 ~]# innobackupex --apply-log /root/allback/

5. [root@51 ~]# innobackupex --copy-back /root/allback/

[root@51 ~]# ls /var/lib/mysql

[root@51 ~]# ls /var/lib/mysql -l

6. [root@51 ~]# chown -R mysql:mysql /var/lib/mysql

[root@51 ~]#ls /var/lib/mysql -l

7. [root@51 ~]# systemctl restart mysqld

[root@51 ~]# mysqld -uroot -p123456

mysql> show databases;

mysql> select \* from db3.user2;

mysql> select cuont(\*) from db3.user2;

**2.2恢复单张表**

user2.frm 存表结构的

user2.ibd 表空间文件(存数据的)

操作步骤:

1,删除表空间

2,导出表信息

3,拷贝表信息文件到数据库目录下

4,修改表信息文件所有者及组用户为mysql

5,导入表空间

6,删除数据库目录下的表信息文件

7,查看表记录

50主机:

误删除数据库操作

mysql> delete from db3.user2;

恢复步骤

1, mysql> alter table db3.user2 discard tablespace; //清空表里的数据

2, [root@mysql1 ~]# innobackupex --apply-log --export /allback/

3, [root@mysql1 ~]# cp

/allback/db3/user2.{cfg,exp,ibd} /var/lib/mysql/db3/

4, [root@mysql1 ~]# chown mysql:mysql /var/lib/mysql/db3/user2.\*

5, mysql> alter table db3.user2 import tablespace; //恢复表里的数据

6, [root@mysql1 ~]# rm -rf /var/lib/mysql/db3/user2.{cfg,exp}

7, mysql> select \* from db3.user2;

**2.3增量备份与恢复**

增量备份

增量备份时,必须有一次备份,通常是完全备份

周一完全备份,周二~周日增量备份

**2.3.1增量备份**

完全备份(目录可以不用创建)

~]#innobackupex --user root --password 密码

/fullbak --no-timestamp

~]# cat /fullbak/xtrabackup\_checkpoints

写入数据作为备份

mysql> insert into db3.user2 values(1,"jack");

增量备份

~]#innobackupex --user root --password 密码

--incremental /new1dir

--incremental-basedir=/fullbak --no-timestamp

增量备份

~]#innobackupex --user root --password 密码

--incremental /new2dir

--incremental-basedir=/new1dir --no-timestamp

[root@mysql1 ~]# cat /fullbak/xtrabackup\_checkpoints

backup\_type = full-backuped //完全备份

from\_lsn = 0 //起始

to\_lsn = 3431636 //结束

last\_lsn = 3431645

compact = 0

recover\_binlog\_info = 0

[root@mysql1 ~]# cat /new1dir/xtrabackup\_checkpoints

backup\_type = incremental //增量备份

from\_lsn = 3431636 //起始

to\_lsn = 3436331 //结束

last\_lsn = 3436340

compact = 0

recover\_binlog\_info = 0

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**2.3.2增量恢复**

1, systemctl stop mysqld

2, rm -rf /var/lib/mysql/\*

3, 完全恢复

innobackupex --apply-log --redo-only /root/fullbak

4, 恢复增量

innobackupex --apply-log --redo-only /root/fullbak

--incremental-dir=/root/nwe1dir

innobackupex --apply-log --redo-only /root/fullbak/

--incremental-dir=/root/new2dir

5, 拷贝文件

innobackupex --copy-back /root/fullbak

6, chown -R mysql:mysql /var/lib/mysql/

7, systemctl start mysqld

cat /root/fullbak/xtrabackup\_checkpoints

backup\_type = log-applied

from\_lsn = 0

to\_lsn = 3436331

last\_lsn = 3436340

compact = 0

recover\_binlog\_info = 0

最后last\_lsn = 3436340 是new2dir last\_lsn = 3436340

总和

对数据库备份命令 cp/tar mysqldump binlog innobackupex

编写备份脚本

周一完全备份,周二至周日增量备份

00 06 \* \* 1 /root/allback.sh

00 07 \* \*2-7 /root/newback.sh

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**day01**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**mysql 主从同步**

主从同步介绍

实现数据自动同步的服务结构

主服务器:接受客户端访问连接

从服务器:自动同步主服务器数据

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

构建思路:

O1: 配置主库

启用binlog日志,授权用户,查看binlog日志信息

O2: 配置从库

确保与主服务器数据一致

设置server\_id, 指定主库信息,启动slave程序

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**配置主服务器**

在51主机上操作

[root@51 ~]# vim /etc/my.cnf

server\_id=51

log-bin=db51

[root@51 ~]# systemctl restart mysqld

授权用户<replication slave>客户端地址允许从库连接

mysql> grant replication slave on \*.\* to repluser@"%" identified by "123qqq...A";

查看日志信息

mysql> show master status\G;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**配置从服务器**

在52主机操作

[root@52 ~]# vim /etc/my.cnf

server\_id=52

[root@52 ~]# systemctl restart mysqld

在51主机上操作(备份数据,,从51拷贝到52)

[root@51 ~]# mysqldump -uroot -p123456 --master-data db3 > /db3.sql

[root@51 ~]# scp /db3.sql root@192.168.4.52:/root/

在52主机操作

创建需要恢复的库

mysql> create databases db3;

从库恢复数据

[root@52 ~]# mysql -uroot -p123456 db3 < /root/db3.sql

查看备份数据相应的日志名及偏移量编号

[root@52 ~]# vim /root/db3.sql

CHANGE MASTER TO

MASTER\_LOG\_FILE='db51.000001', MASTER\_LOG\_POS=441;

指定主库信息

mysql> change master to master\_host="192.168.4.51", //主库IP

master\_user="repluser", //主库授权用户

master\_password="123qqq...A", //授权用户密码

master\_log\_file="db51.000001", //主库日志文件

master\_log\_pos=441; //日志偏移量

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

启动slave start\stop

mysql> start slave;

mysql> show slave status\G;

查看SLave状态

确认Slave\_IO 线程、Slave\_SQL线程都是 YES 状态

Slave\_IO\_Running: Yes

Slave\_SQL\_Running: Yes

Slave\_IO\_Running: Yes //IO线程已运行

Slave\_IO: 复制Master主机 binlog 日志文件里的SQL命令到本机的relay-log

文件里.

Slave\_SQL\_Running: Yes //SQL线程已运行

Slave\_SQL:执行本机relay-log文件里的SQL语句,实现与Master数据一致

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

如果查看slave状态是NO

把数据库恢复为独立数据库服务器

]#cd cat /var/lib/mysql/

]#rm -rf master.info relay-log.info host52-relay-bin.\*

host52(主机名mysql3)

]#systemctl restart mysqld

重新**配置从服务器**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

查看日志信息

show master status\G;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**主从同步结构**

基本应用

**1,单向复制:**

主<--------从

扩展应用

**2,一主多从:**

从<--------主-------->从

**3,链式复制:**

主<--------从(主从)<--------从

**4,互为主从:**

主<-------->主

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**3,链式复制:**

{53主机主}<--------{54主机:从}1从(是主的从,是2从的主)<--------2从

**1从配置文件**

[root@53 ~]# systemctl stop mysqld

[root@53 ~]# cd /var/lib/mysql

[root@53 mysql]# rm -rf master,into

[root@53 mysql]# rm -rf relay-log.info

[root@53 mysql]# rm -rf \*-relay-bin\*

[root@53 mysql]# vim /etc/my.cnf

#server\_id

[root@53 mysql]# systemctl restart mysqld

[root@53 mysql]# mysql -u root -p123456

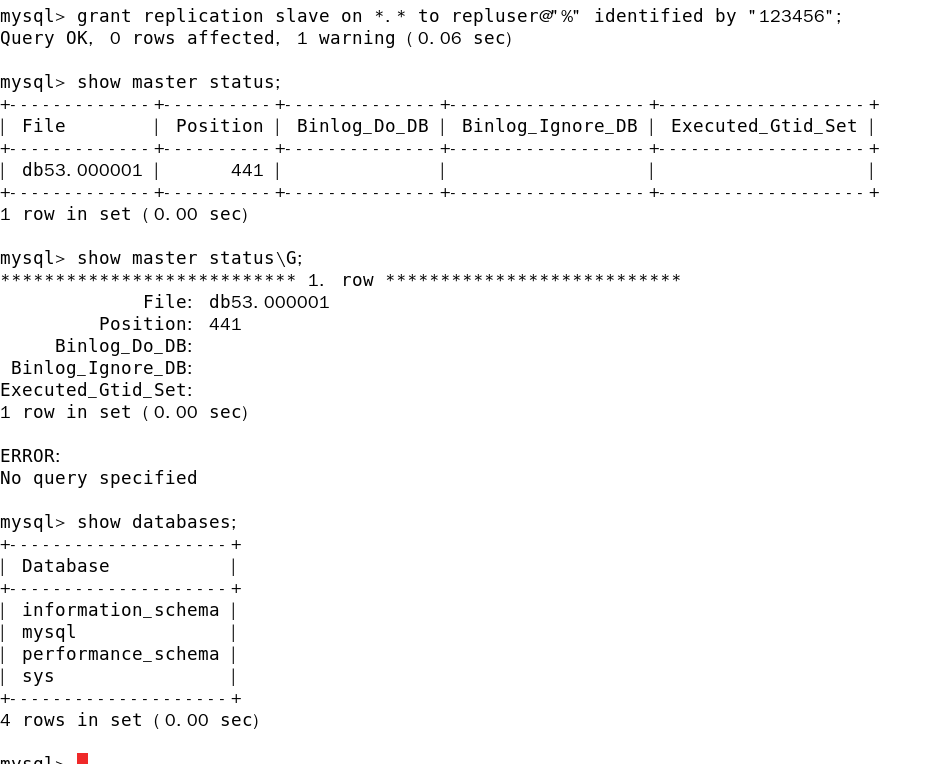
mysql> drop database pzk; 删除多余的库

[root@53 mysql]# vim /etc/my.cnf

server\_id=53

log-bin=db53

[root@53 mysql]# systemctl restart mysqld



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**[root@54 ~]**# vim /etc/my.cnf

[mysqld]

server\_id=54

log-bin=db54

log\_slave\_updates //允许级联复制

[root@54 ~]# systemctl restart mysqld

mysql> grant replication slave on \*.\* to repluser@"%" identified by "123456";

mysql> change master to

master\_host="192.168.4.53",

master\_user="repluser",

master\_password="123qqq...A",

master\_log\_file="db53.000001",

master\_log\_pos=441;

mysql> start slave;

mysql> show slave status\G;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**[root@55 ~]**# vim /etc/my.cnf

[mysqld]

server\_id=55

[root@55 ~]# systemctl restart mysqld

mysql> grant replication slave on \*.\* to repluser@"%" identified by "123456";

mysql> change master to

master\_host="192.168.4.54",

master\_user="repluser",

master\_password="123qqq...A",

master\_log\_file="db54.000001",

master\_log\_pos=441;

mysql> start slave;

mysql> show slave status\G;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**2.4复制模式(mysql主从同步复制模式)**

2.4.1分类

**异步复制模式**

**全同复制模式(组同步)**

**半同复制模式**

**2.4.2配置半同步复制模式**

<主53-------主从54-------从55>

模式配置

默认模式have\_dynamic\_loading

2.4.2.1,命令行配置(**立即生效**)

1,查看是否允许动态加载模块

2,

2,写进配置文件永久生效

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1,查看加载模块

mysql> show variables like "have\_dynamic\_loading";

+-------------------------------+-------+

| Variable\_name | Value |

+-------------------------------+-------+

| have\_dynamic\_loading | YES |

+-------------------------------+-------+

查看插件

mysql>select plugin\_name,plugin\_status from information\_schema.plugins where plugin\_name like "%semi%";

2,命令行加载插件

mysql> install plugin rpl\_semi\_sync\_master soname "semisync\_master.so";

mysql> install plugin rpl\_semi\_sync\_slave soname "semisync\_slave.so";

rpl\_semi\_sync\_master 主模块

rpl\_semi\_sync\_slave 从模块

3,查看插件

mysql>select plugin\_name,plugin\_status from information\_schema.plugins where plugin\_name like "%semi%";

4,查看加载插件状态

mysql> show variables like "rpl\_semi\_sync\_%\_enabled";

+------------------------------------------------+---------+

| Variable\_name | Value |

+------------------------------------------------+---------+

| rpl\_semi\_sync\_master\_enabled | OFF |

| rpl\_semi\_sync\_slave\_enabled | OFF |

+-------------------------------------------------+-------+

5,命令行启用半同步复制

mysql> set global rpl\_semi\_sync\_master\_enabled=1;

mysql> set global rpl\_semi\_sync\_slave\_enabled=1;

查看状态

mysql> show variables like "rpl\_semi\_sync\_%\_enabled";

+------------------------------------------------+---------+

| Variable\_name | Value |

+------------------------------------------------+---------+

| rpl\_semi\_sync\_master\_enabled | ON |

| rpl\_semi\_sync\_slave\_enabled | ON |

+-------------------------------------------------+-------+

6,修改配置文件(设置永久启动半同步复制)

vim /etc/my.cnf

第一种

plugin-load=rpl\_semi\_sync\_master=semisync\_master.so

rpl\_semi\_sync\_master\_enabled=1

plugin-load=rpl\_semi\_sync\_slave=semisync\_slave.so

rpl\_semi\_sync\_slave\_enabled=1

第二种

plugin-load=”rpl\_semi\_sync\_master=semisync\_master.so;rpl\_semi\_sync\_slave=semisync\_slave.so”

rpl\_semi\_sync\_master\_enabled=1

rpl\_semi\_sync\_slave\_enabled=1

7,重启服务

[root@mysql4 ~]# systemctl restart mysqld

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**day02**

读写分离(解决主库压力)

**一、数据读写分离**

MySQL读写分离

1.1数据读写分离介绍

把客户端访问时的**查询请求**和**写入请求**分别给**不同**的数据库服务器出路

1.2数据读写分离优点

减轻**主**服务器的**工作压力**,提高**从**服务器的**硬件利用率**

1.3配置数据读写分离(实现数据读写分离的方式)

1.程序实现 程序员写代码

2.部署服务器实现 在服务器部署数据读写分离服务

1.4拓扑结构

1.5环境准备(部署maxscale(中间件)服务)

. 中间件: mycat mysql-proxy maxscale

下载地址https://downloads.mariadb.com/files/MaxScale

50 做客户端能够ping通57主机即可

克隆一台新的虚拟机 配置IP地址为192.168.4.57不需要安装mysql数据服务

**配置mysql一主一从 同步结构 51主服务器 52从服务器**

拷贝软件给57主机

cd /linux-soft/03/mysql/

scp maxscale-2.1.2-1.rhel.7.x86\_64.rpm root@192.168.4.57:/root/

1.6配置代理服务器 192.168.4.57

1.6.1装包 (安装提供代理服务器软件)

1.6.2修改代理服务器的主配置文件

1.6.3配置数据库服务器 (根据主配置文件的设置做配置)

1.6.4启动服务器

1.6.5查看服务器状态

排错

1.6.6测试配置

在客户端连接代理服务器57访问数据

验证数据读写分离

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

6.1

[root@localhost ~]# rpm -ivh maxscale-2.1.2-1.rhel.7.x86\_64.rpm

[root@mysql7 ~]# rpm -ql maxscale //查看安装清单

提供管理的相关命令

[root@mysql7 ~]# max

maxadmin maxbinlogcheck maxpasswd

maxavrocheck maxkeys maxscale

6.2

备份配置文件(防止配错)

[root@localhost ~]# cp /etc/maxscale.cnf /root/

查看maxscale日志文件,初始状态是空的

这服务运行后线程的数量

9 [maxscale]

10 threads=auto //根据CPU的核数创建对应个数的线程,(自动开启的线程)

定义数据库服务器(5yy,p复制出server2)

18 [server1]

19 type=server

20 address=192.168.4.51 //主服务器

21 port=3306

22 protocol=MySQLBackend

23 [server2]

24 type=server

25 address=192.168.4.52 //从服务器

26 port=3306

27 protocol=MySQLBackend

监视数据库51,52

35 [MySQL Monitor]

36 type=monitor

37 module=mysqlmon

38 servers=server1, server2 //监视server1,server2服务器是否运行,哪 个是主,哪个是从服务器

39 user=maxacalemon //监视51,52时用的用户

40 passwd=123qqq...A //监视时用的用户密码

41 monitor\_interval=10000

不定义只读服务(注释掉#)(用代理服务器自动分离)

52 #[Read-Only Service]

53 #type=service

54 #router=readconnroute

55 #servers=server1

56 #user=myuser

57 #passwd=mypwd

58 #router\_options=slave

读写服务的配置

63 [Read-Write Service]

64 type=service 类型:服务类型

65 router=readwritesplit 调用的模块:读写分离的模块

66 servers=server1, server2 读写分离在那些主机执行

67 user=maxscalerouter 路由用户()

68 passwd=123qqq...A 路由用户密码

69 max\_slave\_connections=100%

管理服务

75 [MaxAdmin Service]

76 type=service 管理服务的类型

77 router=cli 管理服务的模块

不定义只读服务(注释掉#)

85 #[Read-Only Listener]

86 #type=listener

87 #service=Read-Only Service

88 #protocol=MySQLClient

89 #port=4008

91 [Read-Write Listener]

92 type=listener

93 service=Read-Write Service

94 protocol=MySQLClient //调用的功能模块

95 port=4006 //读写分离端口号

97 [MaxAdmin Listener]

98 type=listener

99 service=MaxAdmin Service

100 protocol=maxscaled

101 socket=default

102 port=4016 //管理服务的端口号,自己写的端口(默认有,不用默认的自己添加)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

删除有#号的行

[root@57 ~]# sed -i '/#/d' /etc/maxscale.cnf

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

在主、从数据库服务器创建授权用户

51数据库服务器:

mysql> grant replication slave,replication client on \*.\* to maxscalemon@"%" identified by "123qqq...A";

mysql> grant select on mysql.\* to maxscalerouter@"%" identified by "123qqq...A";

52数据库服务器:

mysql>grant select on mysql.\* to maxscalerouter@"%" identified by "123qqq...A";

启动服务

57数据库:启动服务、查看端口号、

[root@57 ~]# maxscale -f /etc/maxscale.cnf

[root@57 ~]# ps -C maxscale

[root@57 ~]# netstat -utnlp | grep maxscale

启动、

maxscale -f /etc/maxscale.cnf

停止服务

killall -9 maxscale

killall -9 19081

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

有报错查看报错日志

[root@57 ~]# cat /var/log/maxscale/maxscale.log

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

测试配置

在代理服务器本机访问管理

[root@57 ~]# maxadmin -uadmin -pmariadb -P4016

MaxScale> list servers

Servers.

-----------+------------------------+---------+------------------+----------------------

Server | Address | Port | Connections | Status

------------+------------------------+---------+------------------+----------------------

server1 | 192.168.4.51 | 3306 | 0 | Master, Running

server2 | 192.168.4.52 | 3306 | 0 | Slave, Running

------------+-----------------------+---------+------------------+-----------------------

51数据库服务器:

mysql> grant all on db7.\* to yaya66@"%" identified by "123qqq...A";

mysql> create database db7;

52数据库服务器:

mysql> select user from mysql.user;

mysql> show databases;

连接57数据库,查看权限,查看库,创建表,插入数据

50数据库:

[root@51 ~]# ping -c 2 192.168.4.57

[root@51 ~]# which mysql

[root@51 ~]# mysql -h192.168.4.57 -P4006 -uyaya66 -p123qqq...A

mysql> show grants;

mysql> show databases;

mysql> user db7;

mysql> create table db7.a(id int);

mysql> insert into db7.a values (1001);

mysql> select \* from db7.a;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

验证57服务器的功能

1 在主服务器本机向db7.a表插入数据A

2 在从服务器本机向db7.a表插入数据B

3 在主服务器本机查看db7.a表会有数据A

4 在从服务器本机查看db7.a表会有数据A和B

5 登录57主机查看db7.a表会有数据A和B

6 登录57主机向db7.a表插入数据C

7 在主服务器本机向db7.a表插入数据A和C

8 在从服务器本机查看db7.a表会有数据A和B和C

总结:

主服务器在db7.a表有数据A

从服务器在db7.a表有数据A、B

登录57主机在db7.a表,插入写入数据C会到主服务器,从服务器会同步主服务器数据,

登录57主机在db7.a表,查看数据是从服务器提供数据A、B、C;

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**多实例概述**

在一台物理主机上运行多个数据库服务

优点:

节约成本

提高硬件利用率

主配置文件: /etc/my.cnf

每个实力要有独立的数据库目录和监听端口

每个实力要有独立的实例名和独立的sock文件

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

57主机:

[root@57 ~]# killall -9 maxscale

[root@57 ~]# systemctl stop mysqld

[student@room9pc01~]$scp

/linux-soft/03/mysql/mysql-5.7.20-linux-glibc2.12-x86\_64.tar.gz

root@192.168.4.57:/root/

[root@57 ~]# tar -zxvf mysql-5.7.20-linux-glibc2.12-x86\_64.tar.gz

[root@57 ~]# mv mysql-5.7.20-linux-glibc2.12-x86\_64 /usr/local/mysql

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[root@57 ~]#vim /ect/my.cnf

[mysqld\_multi] //启用多实例

mysqld = /usr/local/mysql/bin/mysqld\_safe //指定进程文件路径

mysqladmin = /usr/local/mysql/bin/mysqladmin //指定管理命令路径

user = root //指定进程用户

[mysqld1] //实力进程名称

port = 3307 //端口号

datadir = /dir1 //数据库目录,要手动创建

socket = /dir1/mysql1.sock //指定sock文件的路径和名称

pid-file = /dir1/mysql1.pid //进程pid号文件位置

log-error = /dir1/mysql1.eer //错误日志位置

[mysqld2]

port = 3308

datadir = /dir2

socket = /dir2/mysql2.sock

pid-file = /dir2/mysql2.pid

log-error = /dir2/mysql2.eer

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

管理多实例

启动服务,停止服务

[root@57 mysql]# vim /etc/profile

export PATH=/usr/local/mysql/bin:$PATH

[root@57 mysql]#

echo 'export PATH=/usr/local/mysql/bin:$PATH' >> /etc/profile

[root@57 mysql]# source /etc/profile

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[root@57 mysql]# /usr/local/mysql/bin/mysqld\_multi start 1

.....................

localhost: tW!k)&iFW8va

[root@57 mysql]# mysql -uroot -p'tW!k)&iFW8va' -S /dir1/mysql1.sock

mysql> alter user root@"localhost" identified by "123456";

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

[root@57 mysql]# /usr/local/mysql/bin/mysqld\_multi start 2

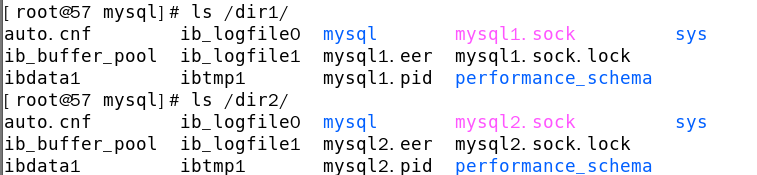
..................

localhost: &58e)oHsEdKA

[root@57 mysql]# mysql -uroot -p'&58e)oHsEdKA' -S /dir2/mysql2.sock

mysql> alter user root@"localhost" identified by "123456";

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

登录多实例数据库

[root@57 mysql]# mysql -u root -p123456 -S /dir1/mysql1.sock

[root@57 mysql]# mysql -u root -p123456 -S /dir2/mysql2.sock

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

授权用户

[root@57 mysql]# mysql -u root -p123456 -S /dir2/mysql2.sock

mysql> grant all 0n db6.\* to yaya66@"%" identified by "123456";

[root@50 ~]# mysql -h192.168.4.57 -P3308 -uyaya66 -p123456

mysql> show grants;

mysql> show databases;

mysql> create database db6;

mysql> create table db6.a (id int);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

启用多实例服务的binlog日志

[root@57 ~]#vim /ect/my.cnf

.........

[mysqld2]

server\_id=2

log-bin=mysqld2

port = 3308

datadir = /dir2

socket = /dir2/mysql2.sock

pid-file = /dir2/mysql2.pid

log-error = /dir2/mysql2.eer

[root@57 mysql]# mysqld\_multi --user=root --password=123456 stop 2

[root@57 mysql]# mysqld\_multi start 2

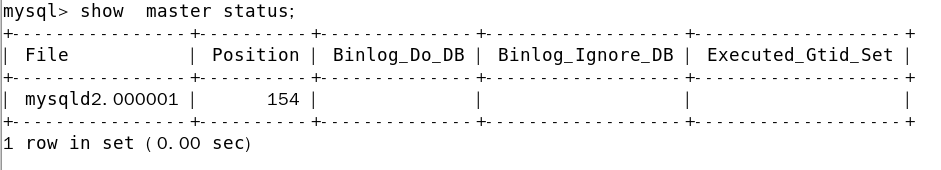
[root@57 mysql]# netstat -utnlp | grep :3308

[root@57 mysql]# ls /dir2/mysqld2.\*

/dir2/mysqld2.000001 /dir2/mysqld2.index

[root@57 mysql]# mysql -uroot -p123456 -S /dir2/mysql2.sock

mysql> show master status;



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**day03**

**部署坏境**

**50客户机,无服务器**

**56分片服务器,无服务器**

**53数据库服务器,db1**

**54数据库服务器,db2**

**55数据库服务器,db3**

[root@50 ~]# systemctl stop mysqld

[root@56 ~]# systemctl stop mysqld

[student@room9pc01 ~]$ scp

maxscale-2.1.2-1.rhel.7.x86\_64.rpm root@192.168.4.56:/root/

[root@53 ~]# mysql -u root -p123456 -e "show master status"

mysql: [Warning] Using a pas................line interface can be insecure.

[root@53 ~]# mysql -u root -p123456 -e "show slave status\G"mysql: [Warning] Using a password on the command line interface can be insecure.

[root@53 ~]# mysql -u root -p123456 -e "show databases"

只有四个库: information\_schema mysql performance\_schema sys

[root@54 ~]# mysql -u root -p123456 -e "show master status"

[root@54 ~]# mysql -u root -p123456 -e "show slave status\G"

[root@54 ~]# mysql -u root -p123456 -e "show databases"

[root@55 ~]# mysql -u root -p123456 -e "show master status"

[root@55 ~]# mysql -u root -p123456 -e "show slave status\G"

[root@55 ~]# mysql -u root -p123456 -e "show databases"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

数据分片概述

**MyCAT**

分库分表

<将存放在一台数据库服务器中的数据,按照特定方式进行拆分,

分散存放在多台数据库服务器中,

已达到分散单台服务器负载的效果.>

垂直分割

纵向切分

将单个数据库的多个表按业务类型分类,分散存储到不同的数据库

水平分割

横向切分

按照表中指定字段的分片规则,将表记录按行切分,分散存储到多个数据库中.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

MyCAT

软件介绍

mycat是基于java的分布式数据库系统中间件,为高并发环境的分布式存储提供解决方案

适合数据大量写入的存储需求

支持MySQL、Oracle、Sqlserver、Mongodb等

提供数据读写分离服务

提供数据分片服务

基于阿里巴巴Cobar进行研发的开源软件

分片规则

mycat支持提供10种分片规则:

1.枚举法 sharding-by-intfile

2.固定分片 rule1

3.范围约定 auto-sharding-long

4.求模法 mod-log

5.日期列分区发 sharding-long

6.通配区模 sharding-by-pattern

7.ASCII码求模通配 sharding-by-prefixpattern

8.编程指定 sharding-by-substring

9.字符串拆分hash解析 sharding-by-stringhash

10.一致性hash sharding-by-murmur

工作过程:

当mycat收到一个SQL命令时

解析SQL命令涉及到的表

然后看对表的配置,如果有分片规则,则获取SQL命令里的分片字段的值,

并匹配分片函数,获得分片列表

然后将SQL命令发往对应的分片服务器执行

最后收集和处理所有分片结果数据,并返回到客户端

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1.部署服务运行环境JDK

[root@56 ~]# yum -y install java-1.8.0-openjdk

[root@56 ~]# java -version

openjdk version "1.8.0\_161"

OpenJDK Runtime Environment (build 1.8.0\_161-b14)

OpenJDK 64-Bit Server VM (build 25.161-b14, mixed mode)

2.安装mycat软件

[root@56~]#tar -xf

Mycat-server-1.6-RELEASE-20161028204710-linux.tar.gz

[root@56 ~]# mv mycat/ /usr/local/mycat

71501

1. 目录结构

bin //mycat命令

catlet //扩展功能

conf //配置文件

lib //mycat使用的jar包

logs //mycat启动日志和运行日志

wrapper.log //mycat服务启动日志

mycat.log //记录SQL脚本执行后的报错内容

重要配置文件说明( ls /usr/local/mycat/conf/)

server.xml //设置连接账户及逻辑库

schema.xml //配置数据分片

rule.xml //分片规则

\*.txt //函数调用文件

4.修改配置文件

[root@56 ~]# vim /usr/local/mycat/conf/server.xml <使用默认配置>

[root@56 ~]# cp /usr/local/mycat/conf/schema.xml /root/

[root@56 ~]# cd /usr/local/mycat/conf/

[root@56 conf]# wc -l schema.xml

77 schema.xml

[root@56 conf]# sed -i '56,77d' schema.xml ---\

[root@56 conf]# sed -i '16,18d' schema.xml \

[root@56 conf]# sed -i '36,39d' schema.xml --删除注释的行

[root@56 conf]# wc -l schema.xml /

48 schema.xml ---/

<schema>..............</schema> 定义数据分片存储的表

<dataNode.........../> 定义数据节点

<dataHost...>......</datahost>定义数据库服务器IP地址及端口

[root@mysql6 conf]# vim schema.xml

配置文件分为三部分

1.在dn1,dn2后面添加dn3

2. <dataNode name="dn2" dataHost="localhost2" database="db2" />

<dataNode name="dn3" dataHost="localhost3" database="db3" />

1. ------------7yy<dataHost name..........</dataHost>,p



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

5.配置数据库服务器

1. 创建存储数据的库
2. 授权mycat服务连接用户admin

[root@53 ~]# mysql -u root -p123456 -e "create database db1"

[root@54 ~]# mysql -u root -p123456 -e "create database db2"

[root@55 ~]# mysql -u root -p123456 -e "create database db3"

[root@53 ~]# mysql -u root -p123456 -e 'grant all on \*.\* to admin@"%" identified by "123qqq...A"'

[root@54~]# mysql -u root -p123456 -e 'grant all on \*.\* to admin@"%" identified by "123qqq...A"'

[root@55 ~]# mysql -u root -p123456 -e 'grant all on \*.\* to admin@"%" identified by "123qqq...A"'

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 启动服务

安装命令

[root@56 conf]# yum -y install mariadb

测试是否能连接

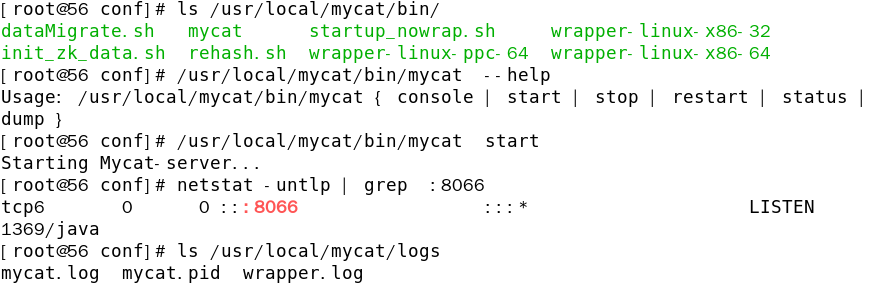
[root@56 conf]# mysql -h192.168.4.53 -uadmin -p123qqq...A

[root@56 conf]# mysql -h192.168.4.54 -uadmin -p123qqq...A

[root@56 conf]# mysql -h192.168.4.55 -uadmin -p123qqq...A

启动服务

[root@56 conf]# /usr/local/mycat/bin/mycat start



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

7.客户端连接

[root@50 ~]# mysql -h192.168.4.56 -P8066 -uroot -p123456

mysql> show databases;

TESTDB

mysql> show tables;

| company |

| customer |

| customer\_addr |

| employee |

| goods |

| hotnews |

| orders |

| order\_items |

| travelrecord |

mysql> desc company;

ERROR 1146 (42S02): Table 'db2.company' doesn't exist

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**8.分片规则**

枚举法-----------------sharding-by-intfile

字段值必须在列表范围内选择

8.1.修改枚举参数

[root@56 ~]# cd /usr/local/mycat/conf/

[root@56 conf]# vim schema.xml

<table name="employee" primaryKey="ID" dataNode="dn1,dn2,dn3"

rule="sharding-by-intfile" />

[root@56 conf]# vim rule.xml

<tableRule name="sharding-by-intfile">

<rule>

<columns>sharding\_id</columns>

<algorithm>hash-int</algorithm>

</rule>

</tableRule>

........................................................................................

<function name="hash-int"

class="io.mycat.route.function.PartitionByFileMap">

<property name="mapFile">partition-hash-int.txt</property>

</function>

[root@56 conf]# vim partition-hash-int.txt

10000=0

10010=1

10020=2

[root@56 conf]# /usr/local/mycat/bin/mycat stop

[root@mysql6 conf]# ../bin/mycat stop

Stopping Mycat-server...

Stopped Mycat-server.

[root@56 conf]# /usr/local/mycat/bin/mycat start

[root@mysql6 conf]# ../bin/mycat start

Starting Mycat-server...

8.2

根据(枚举法)sharding-by-intfile规则建表

[root@50 ~]# mysql -h192.168.4.56 -P8066 -uroot -p123456

mysql> use TESTDB;

mysql> create table employee (

ID int primary key auto\_increment,

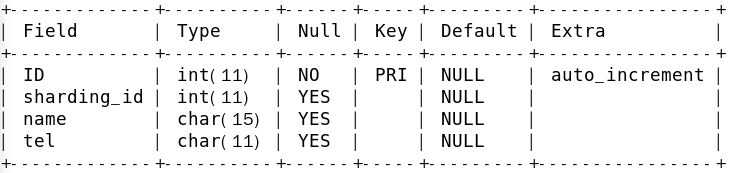
sharding\_id int,

name char(15),

tel char(11)

);

mysql> desc employee;

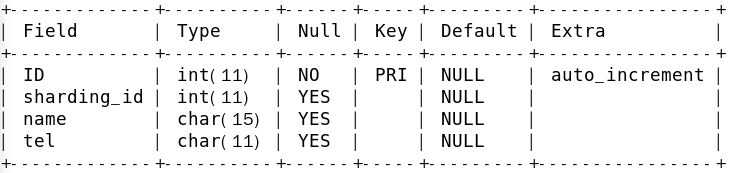


分别在53,54,55主机查看db1.employee表

[root@53 ~]# mysql -u root -p123456 -e 'desc db1.employee'

[root@54 ~]# mysql -u root -p123456 -e 'desc db2.employee'

[root@55 ~]# mysql -u root -p123456 -e 'desc db3.employee'



8.3

从客户端插入数据,

分别在53,54,55主机查看db1.employee表

[root@50 ~]# mysql -h192.168.4.56 -P8066 -uroot -p123456

mysql> use TESTDB;

mysql> insert into employee(sharding\_id,name,tel) values

(10000,"tom",8866);

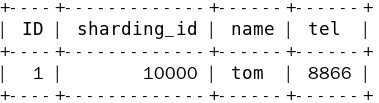
mysql> insert into employee(sharding\_id,name,tel) values

(10010,"jim",8866);

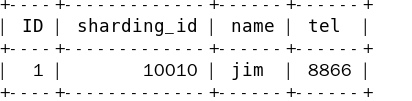
mysql> insert into employee(sharding\_id,name,tel) values

(10020,"bob",8866);

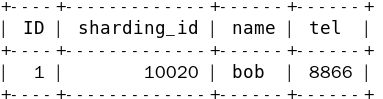
[root@53 ~]# mysql -u root -p123456 -e 'select \* from db1.employee'



[root@54 ~]# mysql -u root -p123456 -e 'select \* from db2.employee'



[root@55 ~]# mysql -u root -p123456 -e 'select \* from db3.employee'



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**9.mod-long**

**求模法**:**根据字段值与设定的数字求模结果存储数据**

[root@56 conf]# vim schema.xml

71508

<table name="hotnews" dataNode="dn1,dn2,dn3" rule="mod-long" />

[root@56 conf]# vim rule.xml

<tableRule name="mod-long">

<rule>

<columns>**id**</columns> 求模字段名

<algorithm>mod-long</algorithm>

</rule>

</tableRule>

<function name="mod-long"

class="io.mycat.route.function.PartitionByMod">

<!-- how many data nodes -->

<property name="count">**3**</property>求模数字

</function>

[root@56 conf]# /usr/local/mycat/bin/mycat stop

[root@56 conf]# /usr/local/mycat/bin/mycat start

9.2客户端连接分片服务器,建表,存储数据

mysql> use TESTDB;

mysql> create table hotnews(

id int,

title char(50),

comment char (200),

worker char(10)

);

mysql> insert into hotnews (id,title,comment,worker)

values

(7,"linux","linux web","nb"),

(3,"linux","vpn","wk"),

(11,"linux","ope","nb");

根据字段值与设定的数字求模结果存储数据,

3除以3结果余0,存储在db1库中

7除以3结果余1,存储在db2库中

11除以3结果余1,存储在db3库中

[root@53 ~]# mysql -u root -p123456 -e 'select \* from db1.hotnews'

| 3 | linux | vpn | wk |

[root@54 ~]# mysql -u root -p123456 -e 'select \* from db2.hotnews'

| 7 | linux | linux web | nb |

[root@55 ~]# mysql -u root -p123456 -e 'select \* from db3.hotnews'

| 11 | linux | ope | nb |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**10.不分片存储,所有数据存储在同一台服务器上**

[root@56 conf]# vim schema.xml

<table name="company" primaryKey="ID" type="global" dataNode="dn1,dn2,dn3" />

mysql> create table company (

id int primary key,

cname char (15),

addr char(20),

epnum int);

mysql> insert into company(id,cname,addr,epnum)

values (1,"tarena","bj",15000);

mysql> insert into company(id,cname,addr,epnum)

values (2,"sina","sh",2500);

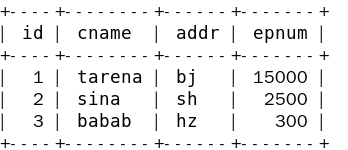
mysql> insert into company(id,cname,addr,epnum)

values (3,"babab","hz",300);

[root@53 ~]# mysql -u root -p123456 -e 'select \* from db1.company'

[root@54 ~]# mysql -u root -p123456 -e 'select \* from db2.company'

[root@55 ~]# mysql -u root -p123456 -e 'select \* from db3.company'



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

11.在mycat服务器上添加新库、新表、重启服务、客户端测试配置

11.1

[root@56 conf]# vim server.xml 添加新库

<user name="root">

<property name="password">123456</property>

<property name="schemas">TESTDB,GAMEDB</property>

11.2

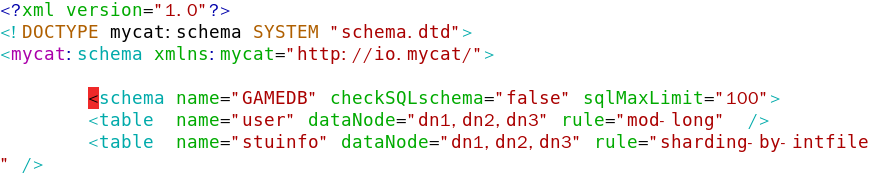
[root@56 conf]# vim schema.xml 添加新表

<schema name="GAMEDB" checkSQLschema="false" sqlMaxLimit="100">

<table name="user" dataNode="dn1,dn2,dn3" rule="mod-long" />

<table name="stuinfo" dataNode="dn1,dn2,dn3" rule="sharding-by-intfile" />

</schema>



11.3

[root@56 conf]# /usr/local/mycat/bin/mycat stop

[root@56 conf]# /usr/local/mycat/bin/mycat start

[root@56 conf]# netstat -untlp | grep :8066

tcp6 0 0 :::8066 :::\* LISTEN 2397/java

11.4

[root@50 ~]# mysql -h192.168.4.56 -P8066 -uroot -p123456

mysql> show databases;

GAMEDB

TESTDB

mysql> create table user(id int,name char(20),class char(9));

mysql> insert into user values (10,"bob","nsd1904");

mysql> insert into user(id,name,class)

values (10,"bob","nsd1904");

mysql> insert into user(id,name,class)

values (20,"tom","nsd1905");

mysql> insert into user(id,name,class)

values (30,"lucy","nsd1906");

根据字段值与设定的数字求模结果存储数据,

[root@53 ~]# mysql -u root -p123456 -e 'select \* from db1.user'

[root@54 ~]# mysql -u root -p123456 -e 'select \* from db2.user'

[root@55 ~]# mysql -u root -p123456 -e "select \* from db3.user"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

diff 文件名 文件名

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**day04**

1. MHA介绍
2. 环境准备

2.1在所有主机上,安装系统光盘里的per软件包

[student@room9pc01 ~]$ cd /linux-soft/03/mysql/

[student@room9pc01 mysql]$ for i in {1,2,3,7}

do

scp -r mha-soft-student/ root@192.168.4.5$i:/root/

done

2.2在所有主机上,安装共享目录里的perl软件包(1.2.3.7主机)

[root@51 ~]# yum -y install perl-\*

[root@51 ~]# cd mha-soft-student/

[root@51 mha-soft-student]# yum -y install perl-\*

2.3配置ssh无密码连接

2.3.1所有数据库服务器之间彼此无密码登录

51、

ssh-keygen

ssh-copy-id root@192.168.4.52

ssh-copy-id root@192.168.4.53

52、

ssh-keygen

ssh-copy-id root@192.168.4.51

ssh-copy-id root@192.168.4.53cd

53、

ssh-keygen

ssh-copy-id root@192.168.4.51

ssh-copy-id root@192.168.4.52

2.3.2管理主机57无密码ssh所有数据库服务器

ssh-keygen

ssh-copy-id root@192.168.4.51

ssh-copy-id root@192.168.4.52

ssh-copy-id root@192.168.4.53

1. 配置管理主机

3.1安装软件

[root@57 ~]# cd mha-soft-student/

[root@57 ~]# rpm -ivh mha4mysql-node-0.56-0.el6.noarch.rpm

[root@57 mha-soft-student]# tar -xf mha4mysql-manager-0.56.tar.gz

[root@57 mha-soft-student]# cd mha4mysql-manager-0.56/

[root@57 mha-soft-student]# perl Makefile.PL

[root@mysql7 mha4mysql-manager-0.56]# make

[root@mysql7 mha4mysql-manager-0.56]# make install

masterha\_check\_repl //检查MySQL复制状况

masterha\_check\_ssh //检查MHA的SSH配置状况

masterha\_manager //启动MHA

masterha\_check\_status //检测MHA运行状态

masterha\_stop //停止MHA

3.2编辑主配置文件

[root@57 mha4mysql-manager-0.56]# mkdir /etc/mha

[root@57 mha4mysql-manager-0.56]#

cp samples/conf/app1.cnf /etc/mha/app1.cnf

[root@57 ~]# vim /etc/mha/app1.cnf

[server default] //管理服务默认配置

manager\_workdir=/etc/mha //工作目录

manager\_log=/etc/mha/manager.log //日志文件

master\_ip\_failover\_script=/etc/mha/master\_ip\_failover //故障切换脚本

ssh\_user=root //访问ssh服务用户

ssh\_port=22 //ssh服务端口

repl\_user=repluser //同步数据授权用户

repl\_password=123456 //密码

user=root //监控用户

password=123456 //密码

[server1] //指定第1台数据库服务器

hostname=192.168.4.51 //服务器IP地址

candidate\_master=1 //竞选主机/竞争主库

[server2] //指定第2台数据库服务器

hostname=192.168.4.52

candidate\_master=1

[server3] //指定第3台数据库服务器

hostname=192.168.4.53

candidate\_master=1

3.3创建故障切换脚本

[root@57 mha-soft-student]# cp master\_ip\_failover /etc/mha

[root@57 mha-soft-student]# chmod +x /etc/mha/master\_ip\_failover

[root@57 mha-soft-student]# vim +35 /etc/mha/master\_ip\_failover

my $vip = '192.168.4.100/24'; # Virtual IP

1. 配置数据库服务器

4.1配置mysql 一主多从

配置主数据库服务器51

配置从数据库服务器52

配置从数据库服务器53

[root@51 ~]# mysql -u root -p123456

mysql> grant replication slave on \*.\* to repluser@"%" identified by "123456";

mysql> show master status;

52,53都做

mysql -u root -p123456

mysql> stop slave;

mysql>change master to

master\_host="192.168.4.51",

master\_user="repluser",

master\_password="123456",

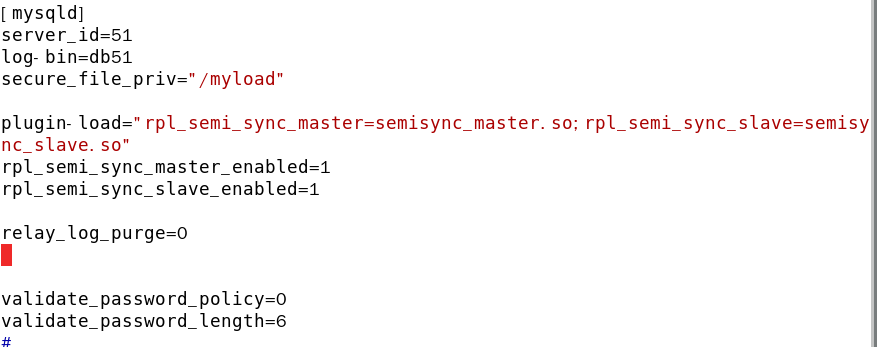
master\_log\_file="db51.000001",

master\_log\_pos=154;

mysql> start slave;

mysql> show slave status\G;

4.2根据管理主机的定义,配置数据库服务器



1.在2台从服务器上添加repluser用户

|||\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*52,53都做:\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|||

mysql> grant replication slave on \*.\* to repluser@"%" identified by "123456";

mysql> show grants for repluser@"%";

2.在2台**从**服务器上启用binlog日志文件

server\_id=52

log-bin=db52

3.在3台服务器上开启半同步复制模式

plugin-load="rpl\_semi\_sync\_master=semisync\_master.so;rpl\_semi\_sync\_slave=semisync\_slave.so"

rpl\_semi\_sync\_master\_enabled=1

rpl\_semi\_sync\_slave\_enabled=1

4.在3台服务器上,禁止删除中继日志文件

relay\_log\_purge=0

5.把vip地址部署在当前主服务器上(51上)

[root@51 ~]# ifconfig eth0:1 192.168.4.100 //配置vip地址

[root@51 ~]# ifconfig eth0:1 //查看eth0:1

1. 在3台服务器上,管理主机使用的root连接用户

<只在51做,52和53会同步>

mysql> grant all on \*.\* to root@"%" identified by "123456";

mysql> show grants for root@"%";

7.在3台服务器上,安装MHA\_node软件{**配置数据节点**}

[root@51 ~]# cd mha-soft-student/

[root@51 mha-soft-student]# rpm -ivh mha4mysql-node-0.56-0.el6.noarch.rpm

[root@52 ~]# cd mha-soft-student/

[root@52 mha-soft-student]# rpm -ivh mha4mysql-node-0.56-0.el6.noarch.rpm

[root@53 ~]# cd mha-soft-student/

[root@53 mha-soft-student]# rpm -ivh mha4mysql-node-0.56-0.el6.noarch.rpm

1. 测试配置

5.1测试SSH配置

[root@**57** ~]# masterha\_check\_ssh --conf=/etc/mha/app1.cnf

..........................

[info] All SSH connection tests passed successfully.

5.2测试主从同步

[root@**57** ~]# masterha\_check\_repl --conf=/etc/mha/app1.cnf

........................................

MySQL Replication Health is OK.

1. 启动管理服务

6.1启动服务

查看状态:

[root@57 mha]# masterha\_check\_status --conf=/etc/mha/app1.cnf

app1 is stopped(2:NOT\_RUNNING).

启动服务

[root@57 mha]# masterha\_manager --conf=/etc/mha/app1.cnf

--remove\_dead\_master\_conf --ignore\_last\_failover

--remove\_dead\_master\_conf //删除宕机主库的配置

--ignore\_last\_failover //忽略xxx.health文件

查看状态

[root@57 ~]# masterha\_check\_status --conf=/etc/mha/app1.cnf

app1 (pid:2829) is running(0:PING\_OK), master:192.168.4.51

6.2连接vip地址访问数据库服务器,

6.2.1在主服务器上添加授权用户

[root@51 ~]# mysql -u root -p123456 -e 'create database db9'

[root@51 ~]# mysql -u root -p123456 -e 'create table db9.a(id int)'

[root@51 ~]# mysql -u root -p123456 -e 'grant select,insert on db9.\* to yaya88@"%" identified by "123456"'

6.2.2客户端连接vip地址访问数据

[root@50 ~]#mysql -h192.168.4.100 -uyaya88 -p123456

mysql> select \* from db9.a;

mysql> insert into db9.a values(100);

mysql> select \* from db9.a;

100

1. 测试高可用集群

7.1停止主服务器51的数据库服务

[root@51 ~]# systemctl stop mysqld

7.2在2台从服务器上查看vip地址(查看到vip地址的是新选举的主服务器)

[root@52 ~]# ifconfig eth0:1 //有192.168.4.100

[root@53 ~]# ifconfig eth0:1 //无192.168.4.100

[root@53 ~]#mysql -uroot -p123456 -e "show slave status\G" | grep -i 192

Master\_Host: 192.168.4.52 //主库是52主机

[root@53 ~]# mysql -uroot -p123456 -e "show slave status\G" | grep -i yes

Slave\_IO\_Running: Yes

Slave\_SQL\_Running: Yes

7.3客户端依然连接vip地址，可以访问到数据

[root@50 ~]#ping -c 2 192.168.4.100

[root@50 ~]# mysql -h192.168.4.100 -uyaya88 -p123456

mysql> insert into db9.a.values (**200**);

mysql> select \* from db9.a;

[root@52 ~]# mysql -u root -p123456 -e 'select \* from db9.a;'

[root@53 ~]# mysql -u root -p123456 -e 'select \* from db9.a;'

都可以查看到**100,200**

7.4管理服务器,并查看状态

[root@57mha]# masterha\_manager --conf=/etc/mha/app1.cnf

--remove\_dead\_master\_conf --ignore\_last\_failover

[root@57 mha]# masterha\_check\_status --conf=/etc/mha/app1.cnf

app1 (pid:3549) is running(0:PING\_OK), master:192.168.4.52

八、修复故障服务器

8.1启动51主机的数据库服务器

启动mysql服务

与主服务器数据一致

指定主服务器信息

启动slave进程

查看状态信息

[root@52 ~]#mysqldump -uroot -p123456 --master-data

db9 > root/db9.sql

[root@52 ~]#scp /root/db9.sql 192.168.4.51:/root/

[root@51 ~]#systemctl start mysqld

[root@51 ~]#mysql -uroot -p123qqq...A db9 < /root/db9.sql

[root@51 ~]#grep db52 /root/db9.sql

MASTER\_LOG\_FILE='db52.000001', MASTER\_LOG\_POS=648;

[root@51 ~]# mysql -uroot -p123qqq...A

mysql> stop slave;

mysql> change master to

master\_host="192.168.4.52",

master\_user="repluser",

master\_password="123qqq...A",

master\_log\_file="db52.000001",

master\_log\_pos=648;

mysql> start slave;

mysql> show slave status\G;

[root@52 ~]#

8.2 配置管理服务器

修改主配置文件

测试集群环境

重启管理服务

查看服务状态

[root@57~]#vim /etc/mha/app1.cnf

[server1]

candidate\_master=1

hostname=192.168.4.51

[root@57~]#masterha\_stop --conf=/etc/mha/app1.cnf

Stopped app1 successfully

[root@57~]# masterha\_check\_ssh --conf=/etc/mha/app1.cnf

[root@57~]# masterha\_check\_repl --conf=/etc/mha/app1.cnf

[root@57~]#masterha\_manager --conf=/etc/mha/app1.cnf

--remove\_dead\_master\_conf --ignore\_last\_failover

[root@57~]#masterha\_check\_status --conf=/etc/mha/app1.cnf

配置步骤总结:

1)配置ssh密钥认证登录

2)配置MySQL一主多从

3)安装软件包

4)配置管理节点

5)启动管理服务

6)测试配置

7)测试故障转移

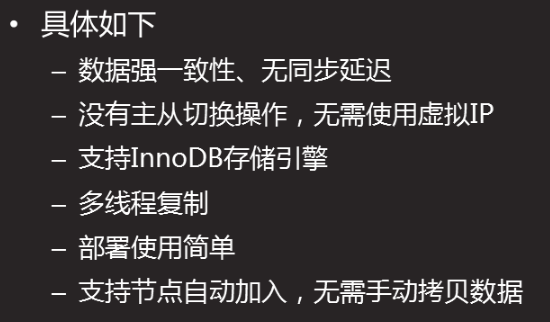
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**day05**

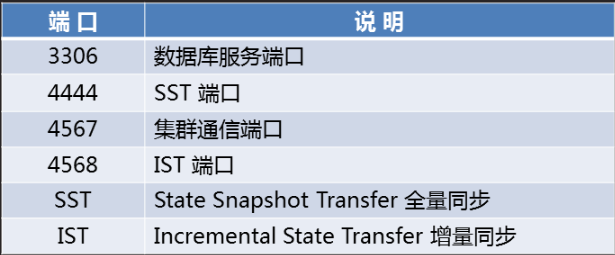
一、PXC介绍



PXC特点



环境部署





1. 更改主机名
2. 主机名绑定,做主机名映射

[root@my01 ~]# vim /etc/hosts

192.168.4.71 my01

192.168.4.72 my02

192.168.4.73 my03

[root@my01 ~]# scp /etc/hosts root@192.168.4.72:/etc/hosts

[root@my01 ~]# scp /etc/hosts root@192.168.4.73:/etc/hosts

[root@my01 ~]# ping -c 3 my01 (3台都可以ping通)

64 bytes from my01 (192.168.4.71): icmp\_seq=1 ttl=255 time=0.016 ms

3.装包

在线热备程序

percona-xtrabackup-24-2.4.13-1.el7.x86\_64.rpm

递归压缩程序

qpress-1.1-14.11.x86\_64.rpm

集群服务程序

Percona-XtraDB-Cluster-5.7.25-31.35-r463-el7-x86\_64-bundle.tar

真机拷贝软件包

三台装包

[root@my01 ~]# tar -xf

Percona-XtraDB-Cluster-5.7.25-31.35-r463-el7-x86\_64-bundle.tar

rpm -ivh libev-4.15-1.el6.rf.x86\_64.rpm

yum -y install percona-xtrabackup-24-2.4.13-1.el7.x86\_64.rpm

rpm -ivh qpress-1.1-14.11.x86\_64.rpm

yum -y install Percona-XtraDB-Cluster-\*.rpm

1. 配置服务

/etc/percona-xtradb-cluster.conf.d/ //目录

mysqld.cnf //数据库服务运行参数配置文件

mysqld\_safe.cnf //Percona Server5.7配置文件

wsrep.cnf //PXC集群配置文件

**每台**都改为主机末尾IP

[root@my01 ~]# vim /etc/percona-xtradb-cluster.conf.d/mysqld.cnf

server-id=71

**每台**都是用**默认**配置

[root@my01 ~]# vim /etc/percona-xtradb-cluster.conf.d/mysqld\_safe.cnf

**每台**都做

[root@my01 ~]# vim /etc/percona-xtradb-cluster.conf.d/wsrep.cnf

wsrep\_cluster\_address=gcomm://192.168.4.71,192.168.4.72,192.168.4.73

//集群成员列表

wsrep\_node\_address=192.168.4.71 //本机IP地址

wsrep\_cluster\_name=pxc-cluster //集群名称,3台必须相同

wsrep\_node\_name=my01 //本机主机名

wsrep\_sst\_auth="sstuser:123qqq...A" //SST数据同步授权用户

5.启动服务(在一台服务器执行)

启动集群服务

[root@my01 ~]# systemctl start mysql@bootstrap.service

[root@my01 ~]# grep pass /var/log/mysqld.log

mysql> alter user root@"localhost" identified by "123456";

mysql> grant reload,lock tables,replication client,process on \*.\* to sstuser@"localhost" identified by "123qqq...A";

reload,

lock tables, 锁表权限

replication client, 查看从服务权限

process

在其他两台启动服务即可

[root@my02 ~]# systemctl start mysql

[root@my03 ~]# systemctl start mysql

[root@my01 ~]# netstat -untlp | grep :3306

[root@my01 ~]# netstat -untlp | grep :4567

1. 测试配置

6.1查看主机在集群中的状态

mysql> show status like "%wsrep%";

wsrep\_incoming\_addresses //成员列表

192.168.4.71:3306,192.168.4.72:3306,192.168.4.73:3306

wsrep\_cluster\_size 3 //集群服务器台数

wsrep\_cluster\_status Primary //集群状态

wsrep\_connected ON //连接状态

wsrep\_ready ON //服务状态

6.2客户端连接集群中的主机,访问数据

6.2.1 在任意一台主数据服务器上添加访问数据的用户(其他两台自动同步)

mysql> grant all on db10.\* to yaya10@"%" identified by "123456";

6.2.2在其他两台数据服务器上查看授权用户

[root@my02 ~]# mysql -u root -p123456 -e 'show grants for yaya10@"%"'

[root@my03 ~]# mysql -u root -p123456 -e 'show grants for yaya10@"%"'

6.2.3客户端50连接集群中的任意1台数据库服务器,访问数据

[root@50 ~]# mysql -h192.168.4.71 -uyaya10 -p123456

mysql> show grants;

mysql> create database db10;

mysql> create table db10.a(id int primary key auto\_increment, name char(10));

mysql> insert into db10.a (name) values ("bob");

mysql> insert into db10.a (name) values ("bob2");

mysql> insert into db10.a (name) values ("bob3");

mysql> select \* from db10.a;

+----+------+

| id | name |

+----+------+

| 1 | bob |

| 4 | bob2 |

| 7 | bob3 |

+----+------+

步长和服务器台数有关

6.2.4测试故障自动恢复

3台服务器,任何一台宕机都可以,

服务器运行后 自动同步宕机期间的数据

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

二、存储引擎

1.MySQL存储引擎

1.1存储引擎介绍

作为可插拔式的组件提供

MySQL服务软件自带的功能程序,处理表的处理器

不同的存储引擎有不同的功能和数据存储方式

1.2列出可用的存储引擎类型/查看存储引擎

show engines

1.3修改标的存储引擎

建表时手动指定

未指定使用默认存储引擎

show create table 表名\G;

[root@50 ~]# mysql -uroot -p123456

mysql> create database db10;

mysql> create table db10.c (name char(10) ) engine=memory;

mysql> show create table c;

1.4修改服务存储引擎

default-storage-engine=存储引擎

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

2.存储引擎的特点

2.1 myisam存储引擎

主要特点

支持表级锁

不支持事务、事务回滚、外键

表文件

表名.frm //表结构

表名.MYI //索引

表名.MYD //数据

2.2innodb存储引擎

主要特点

支持行级锁定

支持事务、事务回滚、外键

表文件

表名.frm

表名.ibd

事务日志文件

ibdata1

ib\_logfile0

ib\_logfile1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

3.MySQL锁机制

锁粒度

表级锁:对整张表加锁

行级锁:仅对被访问的行分别加锁

锁类型

读锁(共享锁):支持并发读

写锁(互斥锁、排它锁):是独占锁,上锁期间其他线程不能读表或写表

查看当前锁状态

show status like "table\_lock%";

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

4.事务特性(ACID)

Atomic**:原子性**

事物的整个操作是一个整体,不可分割,要么全成功,要么全部失败

Consistency:**一致性**

事务操作的前后,表中的记录没有变化

Isolation:**隔离性**

事务操作是互相隔离 不受影响的

Durability:**持久性**

数据域一旦提交,不可改变,永久改变 表数据

5.相关命令

mysql> show variables like “autocommit”; //查看提交状态

mysql> set autocommit=off; //关闭自动提交

mysql>rollback; //数据回滚

mysql>commit; //提交数据

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**day06**

一.、NoSQL数据库管理

1.RDBMS

**关系数据库**管理系统

Relational Database Management System

按照预先设置的组织结构,将数据存储在物理介质上,数据之间可以做关联操作

RDBMS软件

主流的RDBMS软件

MySQL、 MariaDBm、 Oracle、 DB2、 SQL Server;

2.NoSQL

NoSQL(NoSQL = Not Only SQL)

“不仅仅是SQL”, 泛指**非关系型数据库**, 不需要预先定义数据存储结构,

每条记录可以有不同的数据类型和字段个数

NoSQL软件

主流软件:

Memcached、**Redis**、MongoDB、CouchDB、Neo4j、FlockDB

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

二.、Redis

1.Redis介绍

Remote Dictionary Server(远程字典服务器)

是一款高性能的(key / Values)分布式内存数据库

支持数据持久化 (定期把内存里数据存储到硬盘)

支持多种数据类型String、list、hash ....

支持master - salve 模式数据备份

中文网站www.redis.cn

在登录状态输入密码

192.168.4.50:6350> auth 123456

1. 初始配置

/etc/redis/6379.conf //主配置文件

/var/log/redis\_6379.log //日志文件

/var/lib/redis/6379/ //数据库目录

/usr/local/bin/redis-server //服务启动程序

/usr/local/bin/redis-cli //命令行连接命名

1. 管理服务器

/etc/init.d/redis\_6379 stop //停止服务

/etc/init.d/redis\_6379 start //启动服务

ps -C redis-server //查看进程

netstat -utnlp | grep :6379 //查看端口

1. 连接服务

set key名 key值 //存储1个key值

mset key名列表 //存储多个key值

get key名 //获取key值

mget //获取多个key值

select 数据库编号0-15 //切换库

keys \* //显示所有key名

keys a //显示指定key名

exists key名 //测试key名是否存在

ttl key名 //查看key生存时间

expire key名 //设置key有效时间

type key名 //查看key类型

move key名 库编号 //移动key到指定库

del key名 数字 //删除指定的key

flushall //删除内存里所有key

flushdb //删除所在库的所有key

save //保存所有key到硬盘

shutdown //停止服务

[root@50 ~]# cp /etc/redis/6379.conf /root/6379.conf

[root@50 ~]# vim /etc/redis/6379.conf

70 bind 192.168.4.50

93 port 6350

137 daemonize yes

172 logfile /var/log/redis\_6379.log

187 databases 16

264 dir /var/lib/redis/6379

501 requirepass 123456

533 # maxclients 10000

565 # volatile-lru ->

[root@50 ~]# vim +43 /etc/init.d/redis\_6379

$CLIEXEC -h 192.168.4.50 -p 6350 -a 123456 shutdown

[root@50 ~]# /etc/init.d/redis\_6379 stop

[root@50 ~]# /etc/init.d/redis\_6379 start

[root@50 ~]#netstat -utnlp | grep redis-server

[root@50 ~]# redis-cli -h 192.168.4.50 -p 6350 -a 123456

192.168.4.50:6350> ping

PONG

......

......

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

配置PHP支持redis

[root@52 ~]# php -m //PHP支持的功能

[root@52 ~]# rpm -q php php-devel automake autoconf //查看依赖

[root@52 ~]#yum -y install php php-devel automake autoconf //安装依赖 [root@52 ~]#tar -zxf php-redis-2.2.4.tar.gz

[root@52 ~]#cd phpredis-2.2.4/

[root@52 ~]#phpize //生成配置文件php-config及 configure命令

[root@52 ~]#./configure --with-php-config=/usr/bin/php-config //配置

[root@52 ~]#make && make install

[root@52 ~]# vim /etc/php.ini

728 extension\_dir = "/usr/lib64/php/modules"

729 ; On windows:

730 extension = "redis.so"

[root@52 ~]# systemctl restart php-fpm

[root@52 ~]# php -m | grep -i redis

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

书写三个脚本:

[root@52 ~]# cd lnmp

[root@52 lnmp]# cp linkredis.php /usr/local/nginx/html/save.php

[root@52 ~]# cd /usr/local/nginx/html/

1.存脚本

[root@52 html]# vim save.php

<?php

$redis = new redis();

$redis->connect('192.168.4.50',6350);

$redis->auth('123456');

$redis->set('linux','redhat');

?>

2.取值脚本

[root@52 html]# vim get.php

<?php

$redis = new redis();

$redis->connect('192.168.4.50',6350);

$redis->auth('123456');

echo $redis->get('linux');

?>

3.存储脚本

[root@52 html]# vim likredis.php

<?php

$redis = new redis();

$redis->connect('192.168.4.50','6350');

$redis->auth('123456');

$redis->set('x','666666');

echo $redis->get('x');

?>

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

测试save.php

[root@52 html]# curl http://localhost/save.php //空值,没有输出

[root@52 html]# redis-cli -h 192.168.4.50 -p 6350 -a 123456

192.168.4.50:6350> keys \*

1) "linux"

2) "x"

3) "name"

192.168.4.50:6350> get linux

"redhat"

测试get.php

[root@52 html]# curl http://localhost/get.php

redhat

测试likredis.php

[root@52 html]# curl http://localhost/likredis.php

666666

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**day07**

集群环境

redis服务器:6台 ----------------------51----->56

管理主机:1台 -----------------------50

客户端:1台 -----------------------57



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**day08**

hash槽的个数0~16383个

1. 、部署管理主机

为什么要六台服务器,因为要选3台做主,每台都需要1台从,总共6台,

6台里不能有数据

部署ruby脚本运行环境

[root@51 ~]# rpm -q ruby

未安装软件包 ruby

[root@51 ~]# yum -y install ruby

[student@room9pc01 ~]$ cd /linux-soft/03/redis/

[student@room9pc01 redis]$ scp

redis-3.2.1.gem root@192.168.4.51:/root/

[root@51 ~]# gem install redis-3.2.1.gem

创建管理集群脚本

[root@51 ~]# cd /root/redis-4.0.8/src/

[root@51 src]# ls \*.rb

redis-trib.rb

[root@51 src]# mkdir /root/bin

[root@51 src]# cp redis-trib.rb /root/bin

[root@51 src]# chmod +x /root/bin/redis-trib.rb

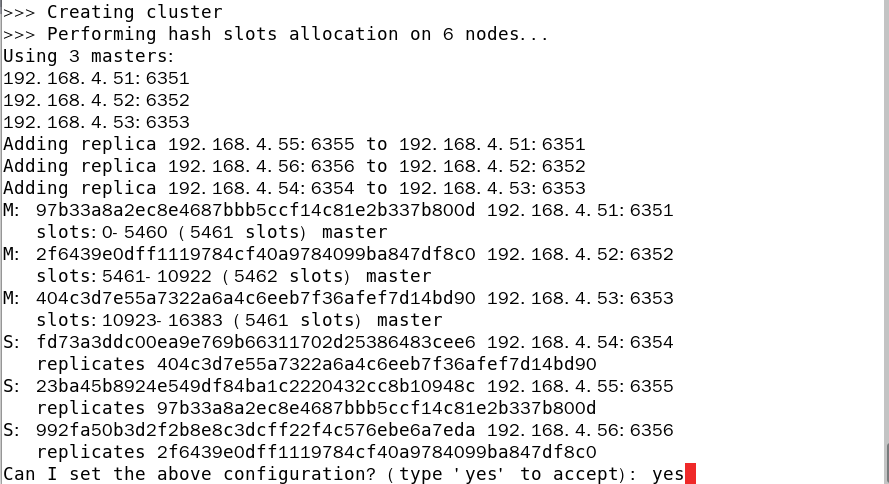
[root@51 src]# redis-trib.rb help

通常会把列表前3台作为主库,把槽(16384个)平均分给3台主服务器,从服务器没有槽.

[root@51 ~]# redis-trib.rb create --replicas 1

192.168.4.51:6351 192.168.4.52:6352 192.168.4.53:6353 192.168.4.54:6354 192.168.4.55:6355 192.168.4.56:6356

--replicas ------------------ 定义每台主库从库的个数



Using 3 masters: //选举出3个主库

[OK] All 16384 slots covered. //提示16384个槽分配完毕

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

在管理主机上查看

查看集群信息

[root@51 ~]# redis-trib.rb info 192.168.4.51:6351

检查集群主机角色

[root@51 ~]# redis-trib.rb check 192.168.4.51:6351

在每一台上,redis服务本机查看

[root@51 ~]# redis-cli -h 192.168.4.51 -p 6351

查看集群信息

192.168.4.51:6351> cluster info

查看集群节点信息

192.168.4.51:6351> cluster nodes

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

访问集群

在客户端可以连接集群中每一台redis服务器

[root@51 ~]# redis-cli -c -h192.168.4.52 -p6352

在**52**redis服务器**存**数据,在**任何**redis服务器可以**取**数据

192.168.4.52:6352> set school 123

192.168.4.53:6353> get school

-> Redirected to slot [8455] located at 192.168.4.52:6352

"123"

192.168.4.56:6356> get school

-> Redirected to slot [8455] located at 192.168.4.52:6352

"123"

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

二.、管理集群

1.集群存取数据的工作原理

变量名 和 crc16算法 做hash计算后 得到一个数字和 %16384 取余

2.测试集群的高可用功能

把**主**服务器的redis服务器**停**止,对应的**从**服务器会自动升级为**主**服务器

把**宕机**的redis服务器运行后,自动做**当前主**服务器的**从**库,并自动**同步**宕机期间产生的**数据**

2.1把53宕机后,从服务器54会自动升级为主服务器,54成为主库,但没有从服务器

[root@53 ~]# /etc/init.d/redis\_6379 stop

Stopping ...

Waiting for Redis to shutdown ...

[root@51 ~]# redis-trib.rb info 192.168.4.52:6352

192.168.4.52:6352 (2f6439e0...) -> 1 keys | 5462 slots | 1 slaves.

192.168.4.**54**:6354 (fd73a3dd...) -> 3 keys | 5461 slots | **0** slaves.

192.168.4.51:6351 (97b33a8a...) -> 0 keys | 5461 slots | 1 slaves.

[OK] 4 keys in 3 masters.

2.2把53宕机运行后,自动成为54主库的从服务器,并自动同步数据

[root@53 ~]# /etc/init.d/redis\_6379 start

Starting Redis server...

[root@51 ~]# redis-trib.rb info 192.168.4.51:6351

192.168.4.51:6351 (97b33a8a...) -> 0 keys | 5461 slots | 1 slaves.

192.168.4.52:6352 (2f6439e0...) -> 1 keys | 5462 slots | 1 slaves.

192.168.4.54:63**54** (fd73a3dd...) -> 4 keys | 5461 slots | **1** slaves.

[root@51 ~]# redis-trib.rb check 192.168.4.51:6351

M: 97b33a8a2ec8e4687bbb5ccf14c81e2b337b800d 192.168.4.51:6351

slots:0-5460 (5461 slots) master

1 additional replica(s)

S: 404c3d7e55a7322a6a4c6eeb7f36afef7d14bd90 192.168.4.**53**:6353

slots: (0 slots) slave

replicates fd73a3ddc00ea9e769b66311702d25386483**cee6**

S: 23ba45b8924e549df84ba1c2220432cc8b10948c 192.168.4.55:6355

slots: (0 slots) slave

replicates 97b33a8a2ec8e4687bbb5ccf14c81e2b337b800d

S: 992fa50b3d2f2b8e8c3dcff22f4c576ebe6a7eda 192.168.4.56:6356

slots: (0 slots) slave

replicates 2f6439e0dff1119784cf40a9784099ba847df8c0

M: 2f6439e0dff1119784cf40a9784099ba847df8c0 192.168.4.52:6352

slots:5461-10922 (5462 slots) master

1 additional replica(s)

M: fd73a3ddc00ea9e769b66311702d25386483**cee6** 192.168.4.**54**:6354

slots:10923-16383 (5461 slots) master

1 additional replica(s)

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. 添加服务器

添加master服务器

不指定角色默认主服务器,需要手动分配hash槽

添加57为主库

[root@51~]#redis-trib.rb add-node 192.168.4.57:6357 192.168.4.52:6352

查看集群信息

[root@51 ~]# redis-trib.rb info 192.168.4.51:6351

如果没有平均分可以

重新平均分片

[root@51 ~]# redis-trib.rb rebalance 192.168.4.51:6351

重新分片

[root@51 ~]# redis-trib.rb reshard 192.168.4.51:6351

How many slots do you want to move (from 1 to 16384)? 4096

What is the receiving node ID? b36f0648ed05af32784acff11b1f0cdbcaf1a864

Source node #1:all

Do you want to proceed with the proposed reshard plan (yes/no)? yes

重新分片后,数据也会对应hash值重新分配

[root@51 ~]# redis-trib.rb info 192.168.4.51:6351

192.168.4.51:6351 (97b33a8a...) -> 0 keys | 4096 slots | 1 slaves.

192.168.4.57:6357 (b36f0648...) -> 2 keys | 4096 slots | 0 slaves.

192.168.4.52:6352 (2f6439e0...) -> 1 keys | 4096 slots | 1 slaves.

192.168.4.54:6354 (fd73a3dd...) -> 2 keys | 4096 slots | 1 slaves.

添加slave服务器

如果不指定主节点的id的,会把新节点随机添加为从节点最少的主库

[root@51 ~]# redis-trib.rb add-node --slave

192.168.4.58:6358 192.168.4.51:6351

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

1. **移除服务器**
   1. **移除slave主机**

从服务器没有hash槽,直接移除,指定从服务器的id值,即可

[root@51 ~]# redis-trib.rb del-node 192.168.4.51:6351

6f432052f93ea5fcde22591cea6b9271b3a8b91d //192.168.4.58的id值

**b)移除master主机**

释放占用的hash槽,移除主机

需要重新分片

[root@51 ~]# redis-trib.rb reshard 192.168.4.51:6351

How many slots do you want to move (from 1 to 16384)? 4096

What is the receiving node ID? 2f6439e0dff1119784cf40a9784099ba847df8c0

Source node #1:fd73a3ddc00ea9e769b66311702d25386483cee6

Source node #2:done

Do you want to proceed with the proposed reshard plan (yes/no)? yes

[root@51 ~]# redis-trib.rb del-node 192.168.4.51:6351

fd73a3ddc00ea9e769b66311702d25386483cee6 //54id值

**c)把移除的redis服务器再次添加到集群里**

c.1启动53主机服务

[root@53 ~]# /etc/init.d/redis\_6379 start

c.2群集重置(CLUSTER RESET)

192.168.4.53:6353> CLUSTER RESET

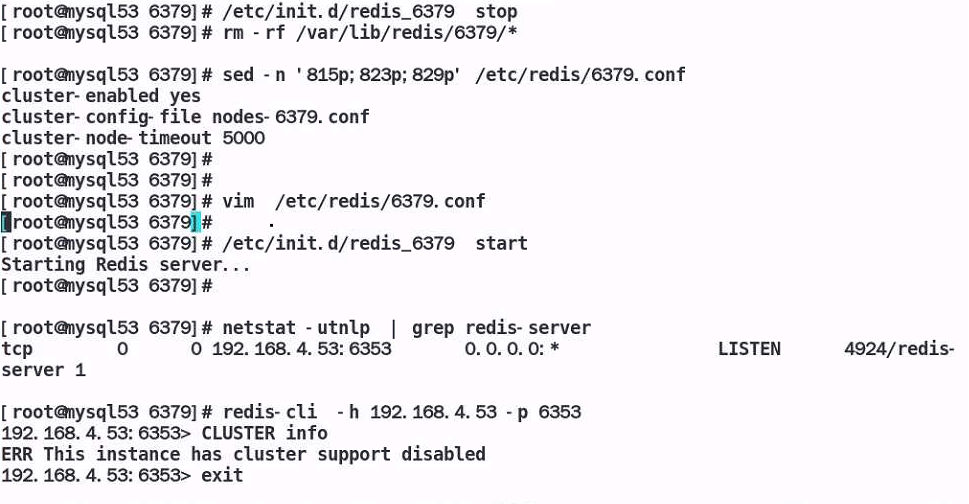
OK

c.3添加主机53主机到集群中里去

[root@51 ~]# redis-trib.rb add-node 192.168.4.53:6353 192.168.4.51:6351

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

把集群中的redis服务还原独立的服务器



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

三、redis主从复制

主服务器:接收客户端访问

从服务器:自动同步主服务器数据到本机

192.168.4.51:6351> **info replication**      //查看复制信息

# Replication

role:master            //是master 服务器

connected\_slaves:0 //从服务器个数零台

将主机192.168.4.52配置为192.168.4.51的从服务器

1）命令行配置（马上生效）

[root@52 ~]# redis-cli -h 192.168.4.52 –p 6352

192.168.4.52:6352> **slaveof** 192.168.4.51 6351 //指定主服务器ip地址与端口

OK

192.168.4.52:6352> info replication //查看复制信息

# Replication

role:slave //从服务器

master\_host:192.168.4.51        //主服务器ip地址

master\_port:6351 //主服务器端口

master\_link\_status:up //连接状态开启

master\_last\_io\_seconds\_ago:3

master\_sync\_in\_progress:0

2）永久配置（重新redis服务后，依然有效）

[root@52 ~]# vim /etc/redis/6379.conf

slaveof 192.168.4.51 6351 //在文件末尾添加或在原有配置项上修改都可以



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

四、配置带验证的主从复制

[root@51 ~]# vim +501 /etc/redis/6379.conf

requirepass 123456 //设置密码

[root@51 ~]# vim +43 /etc/init.d/redis\_6379 //修改脚本

$CLIEXEC -h 192.168.4.51 -p 6351 -a 123456 shutdown //添加密码

[root@51 ~]# /etc/init.d/redis\_6379 stop //停止服务

[root@51 ~]# /etc/init.d/redis\_6379 start //启动服务

[root@51 ~]# netstat -utnlp | grep :6351 //查看端口

1） 修改配置文件，设置主服务器连接密码。

[root@52 ~]## /etc/init.d/redis\_6379 stop //停止服务

[root@52 ~]# vim +289 /etc/redis/6379.conf

masterauth 123456 //设置密码

[root@redisA ~]# /etc/init.d/redis\_6379 start //启动服务

[root@redisA ~]# netstat -utnlp | grep :6351 //查看端口



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**day09**

**一、哨兵服务**

哨兵服务的介绍:

监视master服务器,发现master宕机后,将从服务器自动升级为主服务器.

监视主从服务结构

主配置文件: sentinel.conf

[root@52 ~]# vim /etc/sentinel.conf //创建配置文件

bind 0.0.0.0 //哨兵服务地址（表示本机所有网络接口）

port 26379

sentinel monitor server51 192.168.4.51 6351 1 //监视主服务器

sentinel auth-pass server51 123456 //主服务器密码

[root@52 ~]# redis-sentinel /etc/sentinel.conf //启动哨兵服务

测试高可用

停止主服务器51的redis服务器

在从服务器查看复制信息

52>info replication

查看哨兵服务配置文件,自动监视当前的主服务器

cat /etc/sentinel.conf

**二、持久化**: (redis服务把内存里的数据 保存到硬盘的方式)

2.1RDB介绍

Redis数据库文件,全称Redis DataBase

数据持久化方式之一

数据持久化默认方式

按照指定时间间隔,将内存中的数据集快照写入硬盘

快照术语叫Snapshot

恢复时,将快照文件直接读入内存

定义RDB文件名

dbfilename “dump.rdb”

2.2使用RDB文件恢复数据

[root@50 ~]# redis-cli -h 192.168.4.50 -p 6350 -a 123456

192.168.4.50:6350> keys \*

(empty list or set)

192.168.4.50:6350> set hh k

OK

[root@53 ~]# /etc/init.d/redis\_6379 stop

[root@50 ~]# scp /var/lib/redis/6379/dump.rdb

root@192.168.4.53:/var/lib/redis/6379/dump.rdb

[root@53 ~]# /etc/init.d/redis\_6379 start

[root@53 ~]# redis-cli -h 192.168.4.53 -p 6353

192.168.4.53:6353> keys \*

1) "hh"

3.1AOF介绍

Append Only File

追加方式记录写操作的文件

记录redis服务所有的写操作

不断的将新的写操作,追加到文件的末尾

默认没有启用

使用cat命令可以查看文件内容

AOF优点和缺点

优点:

可以灵活设置持久化方式

出现意外宕机时,尽可能丢失1秒的数据

缺点:

持久化文件的体积通常会大于RDB方式

执行fsync策略时速度可能会比RDB方式慢

3.2优化配置

appendonly yes //启用aof,默认no

........

appendfilename "appendonly.aof" //指定文件名

.......

# appendfsync always //实时记录,并完成磁盘同步

appendfsync everysec //每秒记录一次,并完成磁盘同步

# appendfsync no //写入aof,不执行磁盘同步

.........

auto-aof-rewrite-percentage 100 //首次重写触发值

auto-aof-rewrite-min-size 64mb //再次重写,增长百分比

3.3启用AOF

[root@50 ~]# redis-cli -h 192.168.4.50 -p 6350

192.168.4.50:6350> keys \*

1) "hh"

192.168.4.50:6350> set jj 9

OK

192.168.4.50:6350> set oo 88

OK

192.168.4.50:6350> config set appendonly yes //启用AOF服务

OK

192.168.4.50:6350> config rewrite //写进配置文件

OK

192.168.4.50:6350> save //保存配置

OK

192.168.4.50:6350> exit

[root@50 ~]# ls /var/lib/redis/6379/

appendonly.aof dump.rdb

[root@50 ~]# cat /var/lib/redis/6379/appendonly.aof

3.4使用AOF恢复数据

[root@53 ~]# vim /etc/redis/6379.conf

672 appendonly yes //修改配置文件启动AOF,默认关闭的

[root@53 ~]# /etc/init.d/redis\_6379 stop

[root@53 ~]# rm -rf /var/lib/redis/6379/\*

[root@53 ~]# ls /var/lib/redis/6379/

[root@50 ~]# scp /var/lib/redis/6379/appendonly.aof

root@192.168.4.53:/var/lib/redis/6379/appendonly.aof

[root@53 ~]# /etc/init.d/redis\_6379 start

[root@53 ~]# redis-cli -h 192.168.4.53 -p 6353

192.168.4.53:6353> keys \*

1) "oo"

2) "jj"

3.5修复AOF文件

把文件恢复到最后一次的正确操作

[root@53 ~]# redis-check-aof --fix /var/lib/redis/6379/appendonly.aof

AOF analyzed: size=108, ok\_up\_to=108, diff=0

AOF is valid

1. **数据的类型**
2. String字符串

字符串操作

set key value [EX seconds] [PX milliseconds] [NX|XX]

EX:秒

PX:毫秒

NX:变量存在则不赋值

XX:变量存在覆盖赋值

设置过期时间(-1永不过期,-2)

set a EX 10 set a PX 10

存储变量K值为zxc123===,修改第二个值为A.

192.168.4.50:6350> set k zxc123===

OK

192.168.4.50:6350> setrange k 2 A

(integer) 9

192.168.4.50:6350> get k

"zxA123==="

统计子串长度

192.168.4.50:6350> strlen k

 存在则追加，不存在则创建,返回key长度

192.168.4.50:6350> append k OK

(integer) 11

192.168.4.50:6350> get k

"zxA123===OK"

**setbit** 位存储(节省占用空间)

设置或清除特定偏移量上的位(bit)，value值可以为1或0，offset为0~2^32之间，key不存在，则创建新key

1G=1024M

1M=1024K

1K=1024B

1字节=8位

192.168.4.50:6350> setbit k 0 1

0:第0位

1:存储值为1

192.168.4.50:6350> setbit k 1 0

1:第1位

1:存储值为1

192.168.4.50:6350> bitcount k //统计1出现的个数

**getrange** 截取范围

192.168.4.50:6350> get k

"zxA123===OK"

192.168.4.50:6350> getrange k 6 8

"==="

incr key

j将key的值增加--------------主要应用为计数器

**2.List列表**

192.168.4.50:6350> **lpush** name tom jerry jack lili

(integer) 4

192.168.4.50:6350> **lrange** name 0 3

1) "lili"

2) "jack"

3) "jerry"

4) "tom"

192.168.4.50:6350> **lpop** name

"lili"

192.168.4.50:6350> **llen** name

(integer) 3

192.168.4.50:6350> lrange name 0 2

1) "jack"

2) "jerry"

3) "tom"

192.168.4.50:6350> **lindex** name **1**

"jerry"

192.168.4.50:6350> **lindex** name **-1**

"tom"

192.168.4.50:6350> **lset** name 0 A

192.168.4.50:6350> lrange name 0 2

1) "A"

2) "jerry"

3) "tom"

192.168.4.50:6350> **rpush** name alice

(integer) 4

192.168.4.50:6350> **rpop** name

"alice"

**3.Hash表**

一个变量有多个列,每列有不同的值

\*Redis hash

是一个string类型的field和value的映射表

一个key可对应多个field,一个field对应一个value

**将一个对象存储为hash类型,较于每个字段都存储成string类型更能节省内存**

192.168.4.50:6350> hset s baidu 'www.g.cn'

(integer) 1

192.168.4.50:6350> hget s baidu

"www.g.cn"

192.168.4.50:6350> hset s google 'www.b.cn'

(integer) 1

192.168.4.50:6350> hmget s

(error) ERR wrong number of arguments for 'hmget' command

192.168.4.50:6350> hmget s google baidu

1) "www.b.cn"

2) "www.g.cn"

192.168.4.50:6350> hvals s

1) "www.g.cn"

2) "www.b.cn"

192.168.4.50:6350> hdel s baidu

(integer) 1

192.168.4.50:6350> hvals s

1) "www.b.cn"