RDBMS1\_DAY01

## 1 构建MySQL服务

### 1.1 相关概念

### 1.2 部署mysql数据库服务192.168.4.50

1 安装mysql软件

2 启动mysql服务

3 服务相关参数

相关目录与文件

主配置文件： /etc/my.cnf

数据库目录： /var/lib/mysql/

默认端口号： 3306

进程名： mysqld

传输协议： TCP

进程所有者： mysql

进程所属组： mysql

错误日志文件： /var/log/mysql.log

4 使用初始密码连接服务 #初始密码在/var/log/mysql.log

5 修改登陆密码 #alter user root@”localhost” identified by “密码”

6 使用修改后的密码登陆

### 1.3 修改密码策略

0或者LOW 长度；

1或者MEDIUM 长度；数字、小写/大写，特殊字符

2或者STRONG 长度；数字、小写/大写，特殊字符；字典文件

]# mysql -hlocalhost -uroot -p123qqq...A

mysql> show variables like "%password%"; #模糊查询含有password的变量

mysql> set global validate\_password\_policy=0 ; 临时修改密码策略

mysql> set global validate\_password\_length=6 ; 临时修改 长度

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

mysql> exit

]# mysql -hlocalhost -uroot -ptarena

mysql> exit

]# vim /etc/my.cnf 永久修改

[mysqld]

validate\_password\_policy=0

validate\_password\_length=6

:wq

]# systemctl restart mysqld

]# mysql -hlocalhost -uroot -ptarena

mysql> show variables like "%password%";

## 2 、数据库基本管理

2.1 库管理命令

2.2 表管理命令

mysql> create database db1;

mysql> create table db1.stuinfo(name char(15) , homeaddr char(20) );

mysql> use db1;

mysql> show tables;

mysql> desc db1.stuinfo;

mysql> select \* from db1.stuinfo;

mysql> insert into db1.stuinfo values("bob","usa"),("lilei","china");

mysql> select \* from db1.stuinfo;

mysql> update db1.stuinfo set homeaddr="beijing";

mysql> select \* from db1.stuinfo;

mysql> delete from db1.stuinfo;

mysql> drop table db1.stuinfo;

mysql> show tables;

mysql> create table 学生表2(姓名 char(10) ,年龄 int )DEFAULT CHARSET=utf8;

mysql> show create table 学生表2 ;

mysql> insert into 学生表2 values("张三丰",100);

mysql> insert into 学生表2 values("张翠山",30);

mysql> select \* from 学生表2;

## 3 MySQL 数据类型

### 3.1 数据类型的种类？

字符类型、数值类型

### 3.2 字符类型 char / varchar

mysql> create table t2 (name char(5) , email varchar(30) );

mysql> insert into db1.t2 values ("bob","bob@tedu.cn");

大文本类型：text/blob (字符数大于65535时使用)

### **3.3 数值类型**

只能存储整数 有符号（负数） 无符号（正数） 使用无符号存储 unsigned

整型类型

微小整型 tinyint -128~127 0~255

小整型 smallint -32768~32767 0 ~ 65535

中整数 mediumint

大整数 int

极大整数 bigint

mysql> create table t3(

-> name char(5),

-> age tinyint

-> );

mysql> create table t4(

-> name char(5),

-> age tinyint unsigned

-> );

mysql> create table t4 ( name char(5), age tinyint unsigned);

mysql> insert into t4 values ("tom",-1);

mysql> insert into t4 values ("tom",256);

mysql> insert into t4 values ("tom",56);

mysql> insert into t4 values ("lucy",11.88);

mysql> insert into t4 values ("lili",11.45);

mysql> select \* from t4;

浮点型

单精度 float

双精度 double

小数的组成 整数.小数 总位数

1024.88 6

float

float(7,2)

xxxxx.xx

99999.99

0.00

-99999.99

float(5,2)

xxx.xx

999.99

-999.99

mysql> create table t5 ( name char(5) , gz float(7,2) );

mysql> desc t5;

mysql> insert into t5 values ("bob" , 100000.88);

mysql> insert into t5 values ("bob" , 10000.88);

mysql> insert into t5 values ("bob" , 99999.99);

mysql> insert into t5 values ("bob" , -99999.99);

mysql> insert into t5 values ("bob" , -99999.99);

mysql> insert into t5 values ("bob" , 11.99);

mysql> insert into t5 values ("bob" , 11.23);

mysql> insert into t5 values ("bob" , 23);

mysql> select \* from t5;

### 3.4日期时间类型

日期时间：datetime timestamp

时间：time date

日期：year

mysql> create table t7 (

-> name char(10),

-> your\_start year,

-> up\_time time ,

-> birthday date ,

-> party datetime

-> );

mysql> desc t7;

mysql> insert into t7 values ("bob",1990,083000,20191120,20190707213045);

mysql> select \* from t7;

使用时间函数获取数据给类型赋值

mysql> select curtime(); #获取当前系统时间

mysql> select curdate(); #获取当前系统日期

mysql> select now(); #获取当前系统日期和时间

mysql> select year( now() ); #获取年

mysql> select month( now() ); #获取月

mysql> select day( now() ); #获取日

mysql> select year(20211008);

mysql> insert into t7 values ("bom",2000, curtime(), curdate(), now() );

mysql> insert into t7

values ("jerry",year(20230928), curtime(), curdate(), now() );

mysql> select \* from t7;

当使用2位数赋值给year类型赋值时：

01~69视为 2001~2069

70~99视为 1970~1999

mysql> insert into t7 values ("jim",99,now(),now(),now());

mysql> insert into t7 values ("mack",69,now(),now(),now());

mysql> select \* from t7;

当未给timestamp字段赋值时，自动以当前系统时间赋值，而datetime 值为 NULL (空)

mysql> create table t8 (

-> meetting datetime ,

-> party timestamp

-> );

mysql> insert into t8 values ( now() , now() );

mysql> insert into t8(meetting) values(20201028223058);

mysql> insert into t8(party) values(20181128223058);

mysql> select \* from t8;

### 3.5枚举类型

（字段值 只能在列举的范围里选择）

enum 单选

set 多选

mysql> create table t9 (

-> name char(10) ,

-> sex enum ("boy","girl","no"),

-> likes set ("eat","drink","game","happy")

-> );

mysql> desc t9;

mysql> insert into t9 values ("bob","man","book,it");

mysql>

mysql> insert into t9 values ("bob","boy","book,it");

mysql>

mysql> insert into t9 values ("bob","boy","eat,happy,game");

mysql> select \* from t9;

++++RDBMS1\_DAY02

## 1、表结构

### 1.1 约束条件

:作用是限制如何给字段赋值

NULL 是否允许为空null 默认允许 null

不允许为空 not null

不定义时没有键值

key 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

DEFAULT 默认值：不给字段赋值使用默认值赋值

默认不定义时 是 null

定义 default 值 （要与字段类型匹配）

Extra 额外设置 默认都没有额外设置

create database db2;

create table db2.t1(

name char(10) not null default "",

age tinyint unsigned default 19,

sex enum("m","w") not null default "m"

);

mysql> desc db2.t1;

mysql> insert into db2.t1(name)values("bob");

mysql> insert into db2.t1 values("lucy",21,"w");

mysql> select \* from db2.t1;

mysql> insert into db2.t1 value(null , null , null );

mysql> insert into db2.t1 value(null, null , null );

mysql> insert into db2.t1 value("null", null , null );

mysql> insert into db2.t1 value("null", null , "w" );

mysql> select \* from db2.t1;

### 1.2 修改表结构

命令 alter table 库名.表名 执行动作；

执行动作：

add 添加新字段

add 字段名 类型 [约束条件]；

add 字段名 类型 [约束条件] after 字段名；

add 字段名 类型 [约束条件] first;

mysql> alter table db2.t1 add email varchar(50);

mysql> desc db2.t1;

mysql> select \* from db2.t1;

mysql> alter table t1 add likes set("eat","drink","game","happy")

not null default "eat,drink" after age;

mysql> desc db2.t1;

mysql> select \* from db2.t1;

mysql> alter table db2.t1 add class char(7) default "nsd1905" first;

mysql> desc db2.t1;

mysql> select \* from db2.t1;

modify 修改字段类型

alter table 库名.表名 modify 字段名 类型(宽度) 约束条件 [after 字段名|first ];

mysql> alter table db2.t1 modify

age tinyint unsigned Default 19 after class;

mysql> alter table db2.t1 modify

name varchar(15) not null default "" ;

change 修改字段名

alter table 库名.表名 change 源字段名 新字段名 类型(宽度) 约束条件；

mysql> alter table db2.t1 change email mail varchar(50) ;

mysql> desc db2.t1;

drop 删除字段

alter table 库名.表名 drop 字段名;

mysql> alter table db2.t1 drop mail;

mysql> desc db2.t1;

mysql> select \* from db2.t1;

rename 修改表名

alter table 库名.表名 rename 库名.新表名；

mysql> use db2;

mysql> show tables;

mysql> alter table db2.t1 rename db2.stuinfo;

mysql> show tables;

mysql> select \* from stuinfo;

## 2、MySQL 键值

### 2.1 键值种类

普通索引 index \*\*\*

唯一索引 unique

主 键 primary key \*\*\*

外 键 foreign key \*\*\*

全文索引 fulltext

### 2.2 索引介绍 ?

--类似于书的目录

--对表中字段值进行排序

--索引类型包括：Btree、B+tree、hash

--索引优点：通过创建唯一索引，可以保证数据库中表中每一行数据的唯一性、加快数据查询的速度。

--索引缺点：当对表中的数据进行增加、删除和修改时，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。

### 2.3 普通索引index 的使用

1 使用规则?

--一个表中可以有多个index字段

--字段值允许重复，且可以赋null值

--通常把作为查询条件的字段设置为index字段

--index字段标志是MUL

2 创建索引：

1 建表时创建索引

create table 库.表(

字段列表，

index(字段名)，

index(字段名)

);

mysql> create table db2.t2 (

-> name char(10),

-> age int,

-> class char(7),

-> email char(30),

-> index(name),index(age)

-> );

2 在已有表里创建索引

create index 索引名 on 表名(字段名);

mysql> desc db2.stuinfo;

mysql> create index xxx on db2.stuinfo(name);

mysql> create index sex on db2.stuinfo(sex);

mysql> desc db2.stuinfo;

mysql> show index from db2.stuinfo\G;

3 查看索引字段

基本查看 desc 表名；key列

mysql> desc db2.t2;

详细查看 show index from 表名 \G;

mysql> show index from db2.t2 \G;

4 删除索引

drop index 索引名 on 库.表名;

mysql> drop index age on db2.t2;

mysql> desc db2.t2;

mysql> show index from db2.t2 \G;

### 2.4 主键primary key 的使用

1 使用规则 ?

--字段不允许重复，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

--主键字段的标识是PRI

--主键通常与auto\_increment连用（自增长）

--通常把表中唯一标识记录的字段设置为主键

2 创建主键

2.1 建表时创建主键

--primary key （字段名）

mysql> create table t3( name char(10) primary key , age int );

mysql> create table t4( name char(10) , age int , primary key(name) );

mysql> desc t3;

mysql> desc t4;

mysql> insert into t3 values ("bob", 19);

mysql> insert into t3 values ("bob", 21);

mysql> insert into t3 values (null, 17);

mysql> insert into t3 values ("tom", 17);

mysql> select \* from t3;

2.2 在已有表里创建主键

alter table 库.表 add primary key (字段名);

mysql> desc db2.stuinfo;

mysql> alter table db2.stuinfo add primary key (name);

mysql> desc db2.stuinfo;

2.3 复合主键:表中多列一起做主键

插入记录时多列的值不允许同时重复

create table db2.t5(

class char(7),

name char(10),

money enum("no","yes"),

primary key(class,name)

);

desc db2.t5;

mysql> insert into db2.t5 values("nsd1905","bob","yes");

mysql> insert into db2.t5 values("nsd1905","bob","yes"); 报错

mysql> insert into db2.t5 values("nsd1905","bob","no"); 报错

mysql> insert into db2.t5 values("nsd1906","bob","yes");

mysql> insert into db2.t5 values("nsd1906","tom","no");

mysql> select \* from db2.t5;

2.3 删除主键

alter table 库.表 drop primary key;

mysql> alter table db2.t5 drop primary key;

mysql> desc db2.t5;

2.4 在已有表里创建主键

alter table 库.表 add primary key(字段名)；

alter table 库.表 add primary key(字段名列表)；

mysql> alter table db2.t5 add stu\_num char(9) first;

mysql> delete from db2.t5;

mysql> select \* from db2.t5;

mysql> show tables;

mysql> desc db2.t5;

mysql> alter table db2.t5 add primary key (stu\_num , class , name);

mysql> desc db2.t5;

2.5 主键与auto\_increment 连用

自增长

create table db2.t6(

stu\_num int primary key auto\_increment,

name char(10),

age int

);

mysql> desc db2.t6;

mysql> insert into db2.t6 (name,age) values("bob",19);

mysql> insert into db2.t6 (name,age) values("tom",19);

mysql> insert into db2.t6 (name,age) values("lucy",19);

mysql> select \* from db2.t6;

mysql> insert into db2.t6 values(9,"alice",18);

mysql> select \* from db2.t6;

mysql> insert into db2.t6 (name,age) values("jerry",18);

mysql> select \* from db2.t6;

mysql> delete from db2.t6;

mysql> insert into db2.t6 (name,age) values("jerry",18);

mysql> select \* from db2.t6;

mysql> insert into db2.t6 values(null, "lilei",17);

mysql> select \* from db2.t6;

2.3 foreign key 外键

作用：向表中插入记录时，字段的值在另外一张表字段值里选择。

外键使用规则？

--表存储引擎必须是innodb

--字段类型要一致

--被参照的字段必须是索引类型的一种（primary key）

create table xxx (........)engine=innodb;

创建外键的命令格式：

create table 库.表 (

字段列表，

foreign key(字段名) references 库.表(字段名)

on update cascade on delete cascade

)engine=innodb;

use db2;

create table ygb(

yg\_id int primary key auto\_increment,

name char(20),

sex enum("boy","girl")

)engine=innodb;

insert into db2.ygb(name,sex)values("bob","boy");

insert into db2.ygb(name,sex)values("lucy","girl");

select \* from db2.ygb;

create table gzb(

gz\_id int,

pay float(7,2),

foreign key(gz\_id) references ygb(yg\_id) on update cascade

on delete cascade )engine=innodb;

show create table gzb;

insert into gzb values(1,10000);

insert into gzb values(2,20000);

insert into gzb values(3,25000); 报错

update ygb set yg\_id=8 where yg\_id=2;

select \* from ygb;

select \* from gzb;

delete from ygb where yg\_id=8;

select \* from ygb;

select \* from gzb;

alter table gzb add primary key(gz\_id);

删除外键

show create table gzb;

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

show create table gzb;

insert into gzb values(8,20000);

insert into gzb values(7,20000);

select \* from gzb;

select \* from ygb;

Mysql\_day3

## 一.数据导入与导出:

### 1.1默认检索路径:

1.1.1查看:

mysql> show variables like "%file%" #模糊查询所有’file’内置变量

mysql> show variables like "%secure\_file\_priv%";#查询默认检索路径

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

| Variable\_name | Value |

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

| secure\_file\_priv | /var/lib/mysql-files/ |

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

1.1.2 修改:

vim /etc/my.cnf #MySQL主配置文件

[mysqld]

secure\_file\_priv="/myload" #修改默认检索路径

mkdir /myload #新建定义的检索路径

chown mysql /myload #对新建的检索路径文件修改权限

systemctl restart mysqld #重启mysql服务

mysql> show variables like "%secure\_file\_priv%";#查询默认检索路径

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

| Variable\_name | Value |

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

| secure\_file\_priv | /myload/ |

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

### 1.2数据导入:

把系统文件的内容存储到数据库的表里;(批量操作)

1.2.1命令格式

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”;

1.2.2注意事项:

1.字段要与文件一致

2.表字段类型和字段个数要与文件匹配

3.导入数据时指定文件的绝代路径

1.2.3数据导入步骤

1.建库

create database db3;

2.建表

mysql>create table db3.user (

-> name char(50),

-> passwd char(1),

-> uid int,

-> gid int,

-> comment varchar(150),

-> homedir char(50),

-> shell char(50));

3.拷贝文件到检索目录下

system cp /etc/passwd /myload/

#在mysql命令行中执行系统命令,命令前需加system

4.导入数据

load data infile "/myload/passwd"

into table db3.user

fields terminated by ":"

lines terminated by "\n" ;

5.查看表记录

select \* from db3.user;

6.添加主键及主键自增长

alter table db3.user

add id int primary key auto\_increment first;

### 1.3数据导出:

把表记录存储到系统文件里;(批量操作)

1. 3.1命令格式:

1.Sql查询命令 into outfile “目录/文件名”;

2.Sql查询命令 into outfile “目录/文件名”

fields terminated by “分隔符”;

3.Sql查询命令 into outfile “目录/文件名”

fields terminated by “分隔符”;

Lines terminated by “\n”;

1.3.2注意事项:

1.导出数据的行由查询结果决定.

2.导出的是表记录,不包括字段名.

3.自动创建存储的数据文件.

4.存储数据文件具有唯一性.

1.3.3具体步骤:

mysql> select \* from db3.user where id <=2

-> into outfile "/myload/user1.txt";

mysql> select \* from db3.user where id <=2

-> into outfile "/myload/user1.txt"

-> fields terminated by "#";

mysql> select \* from db3.user where id <=2

-> into outfile "/myload/user1.txt"

-> fields terminated by "#";

-> lines terminated by "!!!";

mysql> select name,id,shell from db3.user where id <=3

-> into outfile "/myload/user4.txt";

## 二,管理表记录

### 2.1增加表记录

2.1.1.语法格式:

1.插入一条表记录给所有字段赋值

Insert into 库名.表名 values (字段值列表);

2.插入多条表记录给所有字段赋值

Insert into 库名.表名 values (字段值列表),(字段值列表)....;

3.插入一条表记录给指定字段赋值

Insert into 库名.表名(字段值列表) values (字段值列表);

4.插入多条表记录给指定字段赋值

Insert into 库名.表名(字段值列表)

values (字段值列表),(字段值列表)....;

2.1.2 具体案例:

1. Insert into db3.user values(

22,“bob”,”x”,2000,2000,”studentuser”,

”/home/bob”,”/bin/bash”);

2.Insert into db3.user values(

22,“bob”,”x”,2000,2000,”studentuser”,

”/home/bob”,”/bin/bash”),(字段值列表);

3.Insert into db3.user(name,passwd) values(“bob”,”x”);

4.Insert into db3.user(name,passwd) values(“bob”,”x”),

(字段值列表)....;

2.1.3 注意事项:

1. 字段值要与字段类型相匹配.
2. 字符类型的字段,要用””号引起来.
3. 依次给所有字段赋值时,字段名可以省略.
4. 只给部分字段赋值时,必须明确写出对应的字段名称.
5. 没有赋值的字段使用默认值或自增长赋值.

### 2.2查询表记录

2.2.1 语法格式

1. 查看所有记录

select 字段1,.....,字段N from 库名.表名；

1. 条件查询

select 字段1,.....,字段N from 库名.表名

where 条件表达式；

2.2.2 具体案例

1. select name,uid from db3.user;
2. select name,uid from db3.user where id <=3;

2.2.3 注意事项

\*表示所有表字段

查看当前库表记录时库名可以省略

字段列表决定显示列个数

条件决定显示行个数

### 2.3更新表记录

2.3.1语法格式

1. 批量修改：

update 库名.表名

set 字段名＝值，字段名＝值，字段名＝值，．．．．．．；

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；

2.3.2注意事项：

1. 字段值要与字段类型相匹配.

　　2.字符类型的字段,要用””号引起来.

1. 若不使用where限定条件，会更新所有记录字段值．
2. 限定条件时，只更新匹配条件的记录的字段值

### 2.4删除表记录

2.4.1条件匹配删除表记录

delete from 库名.表名　 where 　条件表达式；

2.4.2删除所有表记录

delete from 库名.表名；

## 三，匹配条件

（适用于select update delete）

### 3.1 基本匹配条件

3.1.1 数值比较

= != <= >= < >

3.1.2 字符比较／空／非空

= != null not null

3.1.3 逻辑匹配

3.1.4 范围内匹配／去重显示

in　(值列表)　　 在．．．．．．这里

not in (值列表) 不在．．．．．．这里

between 数字　　　and 　数字　 在．．．之间．．．

distinct 字段名 去重显示

### 3.2高级匹配条件

3.2.1模糊匹配

用法

where　 字段名　like ‘通配符’

\_　表示一个字符

% 表示０～n个字符

#查询user表中name以s结尾并且s前有任意４个字符

mysql> select \* from db3.user where name like '\_\_\_\_s';

3.2.2 正则表达式

正则元字符：　^ $ . [ ] \* |

#查询user表中name为４个字符的

mysql> select \* from db3.user where name regexp '^....$';

#查询user表中name中以数字开头的

mysql> select \* from db3.user where name regexp '^[0-9]';

#查询user表中name中以数字开头结尾的

mysql> select \* from db3.user where name regexp '^[0-9]$';

#查询表中用户以r开头以c结尾的

select name from userdb.user where name regexp '^r.\*c$';

3.2.3四则运算－－字段必须是数值类型

+ - \* % / ()

select uid,gid,uid+gid zh from db3.user;

select name,age,2019-age csnf from db3.user

where name ='root';

select \* from db3.user where uid % 2 = 0;

update db3.user set gid=gid+1 where gid <=10;

select name,uid,gid,(uid+gid)/2 pjz from db3.user

where name ='halt';

### 3.3操作查询结果

3.3.1聚集函数

1.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) #最小值

max(字段名) #最大值

count(字段名) #统计字段值个数

select avg(gid) from db3.user;

select max(gid) from db3.user;

select sum(gid) from db3.user;

select count(\*) from db3.user;

select count(name) from db3.user

where shell != "/bin/bash";

3.3.2查询结果排序

1. 用法

Sql查询 order by 字段 asc/desc

select \* from db3.user where uid <=100 order by shell ;

select \* from db3.user where uid <=100 order by shell desc;

3.3.3查询结果分组

1. 用法

Sql查询 group by 字段名

select uid from db3.user group by uid ;

select shell from db3.user where uid <=100 group by shell ;

3.3.4查询结果过滤

Sql查询 having 条件

select name from db3.user where uid <=3000 having name ="mysql";

3.3.5限制查询结果显示行数

1.用法

SQL查询 limit 数字; #显示查询结果前多少条记录

SQL查询 limit 数字1,数字2; #显示指定范围内的查询记录

数字1: 起始行(0表示第1行)

数字2: 总行数

select \* from db3.user where id <=10 limit 0,2;

select \* from db3.user where id <=10 limit 3,2;

## 四，MySQL管理工具

Mysql\_day4

## 一，用户授权

### 1.1用户授权

1.1.1 什么是用户授权?

在数据库服务器上添加客户端连接时使用的用户,并设置访问权限

1.1.2 为什么学习用户授权?

默认只允许数据库管理员root用户本机登陆

1.1.3 配置用户授权:授权命令格式

mysql> grant 权限列表 on 数据库名 to 用户名@”客户端地址” identified by “密码” [ with grant option ];

with grant option:允许授权客户端拥有授权权限

例如:

grant select,update(name) on userdb.user to yaya108@"%" identified by "userweb888" ;

权限列表:

个别权限: select,insert,update,delete

所有权限: all

无权限: usage

指定字段权限:update(字段1,字段2)

命令 权限

usage 无权限

SELECT 查询表记录

INSERT 插入表记录

UPDATE 更新表记录

DELETE 删除表记录

CREATE 创建库、表

DROP 删除库、表

RELOAD 有重新载入授权 必须拥有reload权限，才可以执行flush [tables | logs | privileges]

SHUTDOWN 允许关闭mysql服务 使用mysqladmin shutdown 来关闭mysql

PROCESS 允许查看用户登录数据库服务器的进程 （ show processlist; ）

FILE 导入、导出数据

REFERENCES 创建外键

INDEX 创建索引

ALTER 修改表结构

SHOW DATABASES 查看库

SUPER 关闭属于任何用户的线程

CREATE TEMPORARY TABLES 允许在create table 语句中使用 TEMPORARY关键字

LOCK TABLES 允许使用 LOCK TABLES 语句

EXECUTE 执行存在的Functions,Procedures

REPLICATION SLAVE 从主服务器读取二进制日志

REPLICATION CLIENT 允许在主/从数据库服务器上使用 show status命令

CREATE VIEW 创建视图

SHOW VIEW 查看视图

CREATE ROUTINE 创建存储过程

ALTER ROUTINE 修改存储过程

CREATE USER 创建用户

EVENT 有操作事件的权限

TRIGGER, 有操作触发器的权限

CREATE TABLESPACE 有创建表空间的权限

库名

所有库所有表: \*.\*

一个库: 库名.\*

一张表: 库名.表名

用户名

授权时自定义 要有标示性

存储在mysql库的user表里

客户端地址

所有主机: %

网段内的所有主机: 192.168.4.%

1台主机: 192.168.4.1

数据库服务器本机: lsocalhost

应用实例:

grant all on db4.\* to yaya@"%" identified by "123qqq...A";

相关命令:

查询当前连接用户: select user();

查询当前连接用户权限: show grants;

管理员查看已有授权用户权限: show grants for 用户名@”客 户端地址”;

授权用户连接后修改连接密码: set password=password(“密码”);

管理员重置授权用户连接密码: set password for 用户名@”客户端地址” =password(“密码”);

删除授权用户(必须有管理员权限): drop user 用户名@”客户端地址”;

授权库mysql:记录授权信息

user表 记录已有的授权用户及权限

db表 记录已有授权用户对数据库的访问权限

tables\_priv表 记录已有授权用户对表的访问权限

Colu mns\_priv表 记录已有授权用户对字段的访问权限

\G:查询结果竖向显示;

撤销权限:删除已有授权用户的权限

命令格式:

mysql> revoke 权限列表 on 库名.表 from 用户名@”客户端地址”;

例如:

撤销root用户的更新,删除权限:

mysql> revoke update ,delete from root@”%”;

撤销root用户的所有权限:

mysql> revoke all from root@”%”;

### 1.2 root密码

1.2.1重置root密码

#]mysqladmin -hlocalhost -uroot -p password "新密码"

Enter password:旧密码

1.2.2 回复root密码

#]vim /etc/my.cnf

[mysqld]

skip-grant-tables

wq

#]systemctl restart mysqld

#]mysql

mysql >update mysql.user set authentication\_string=password(“密 码”) where user=”root” and host=”localhost”;

mysql >flush privileges;

mysql >exit;

## 二，完全备份

### 2.1数据备份方式

2.1.1物理备份:冷备: cp,tar,......

1.数据备份及恢复

a)备份操作

Mkdir /dbbak

cp -r /var/lib/mysql/ /dbbak/

tar -zcvf /dbbak/mysql.tar.gz /var/lib/mysql/\*

scp -r mysql root@192.168.4.51:/root/

b)恢复操作

systemctl stop mysqld

rm -rf /var/lib/mysql

cp -r /root/mysql /var/lib/

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

systemctl restart mysqld

2.1.2逻辑备份:

mysqldump #备份命令

mysql #恢复命令

1. 数据备份策略
   1. 完全备份:备份所有数据
   2. 增量备份:备份上次备份后,所有新产生的数据
   3. 差异备份:备份完全备份后,所有新产生的数据
2. 完全备份命令格式:

mysqldump -uroot -p密码 库名 > 目录/xxx.sql

例如:

mysqldump -uroot -p123456 -A > /dbbak/all.sql

mysqldump -uroot -p123456 db3 > /dbbak/db3.sql

mysqldump -uroot -p123456 db3 user > /dbbak/db3\_user.sql

mysqldump -uroot -p123456 -B mysql db3 > /dbbak/twodb.sql

1. 备份时库名的表示方式
   1. 所有库: --all-databases 或 -A
   2. 单个库: 数据库名
   3. 单张表: 数据库名 表名
   4. 多个库: -B 数据库1 数据库2

4.注意事项:无论备份还是恢复,都需要验证用户权限

5.完全恢复命令格式:(备份多个或全部库时不需要加库名)

mysql -uroot -p密码 库名 < 目录/xxx.sql

例如:恢复db3库

mysql -uroot -p123456 db3 < /opt/db3.sql

6.完全备份缺点:

锁表(恢复备份时数据无法读取写入),当有增量数据时,数据库崩亏后,增量数据无法恢复.

## 三，增量备份:

### 3.1日志概述

1.什么是binlog日志

a)也称作二进制日志

b)MySQL服务日志文件的一种

c)记录除查询之外的所有SQL命令

d)可用于数据备份和恢复

e)配置mysql主从同步的必要条件

1. 启用日志

server\_id=数字 #指定ID值(1-255)

log\_bin=[目录名/文件名] #启用biglog日志

max\_binlog\_size=数值m #指定日志文件容量,默认1G

]# vim /etc/my.cnf

[mysqld]

......

log\_bin

server\_id=50

]# systemctl restart mysqld

1. binlog相关文件

主机名-bin.index 索引文件

主机名-bin.000001 第1个二进制日志

主机名-bin.000002 第2个二进制日志

......

3.2 使用binlog日志文件恢复数据

命令格式]# mysqlbinlog 目录/文件 | mysql -u用户 -p密码

3.3 自定义日志文件名和存储目录

]# vim /etc/my.cnf

[mysqld]

server\_id=50

log\_bin=/mylog/plj

:wq

]# mkdir /mylog

]# chown mysql /mylog

]# systemctl restart mysqld

3.4查看正在使用的日志文件 show master status;

3.5手动创建新的日志文件

]# systemctl restart mysqld

mysql> flush logs;

]# mysql -uroot -pabc123...Q -e 'flush logs'

]# mysqldump -uroot -pabc123...Q --flush-logs db3 > /dbbak/db3.sql

3.6删除已有的日志文件

a)删除指定编号之前的binlog日志文件:

mysql> purge master logs to 'binlog文件名';

b)删除所有binlog日志,重建新日志

mysql> reset master ;

3.7修改日志记录格式

查看日志当前记录模式:mysql> show variables like ‘%binlog\_format%’

日志三种记录模式

1.statement 报表模式 2.row 行模式 3.mixed 混合模式

]# vim /etc/my.cnf

[mysqld]

binlog\_format="mixed" #修改为混合模式

:wq

]# systemctl restart mysqld

3.8日志文件如何区分记录多条sql命令（日志记录sql命令的方式）

a)偏移量

--start-position=数字 --stop-position=数字

b)时间点

--start-datetime="yyyy-mm-dd hh:mm:ss" --stop-datetime="yyyy-mm-dd hh:mm:ss"

3.9读取日志指定范围的sql命令恢复数据

1. 基本思路
   1. 使用mysqlbinlog提取历史SQL操作
   2. 通过管道交给mysql命令执行
2. 命令格式

mysqlbinlog 日志文件 | mysql -u用户 -p密码

1. 例如:

mysqlbinlog --start-position=325 --stop-position=1708 /opt/plj.000002 | mysql -uroot -pabc123...Q

Day05.数据备份与恢复

## 一,percona软件

1.1mysql备份工具

1. 常用的mysql备份工具
   1. 物理备份缺点

---跨平台性差

---备份时间长、冗余备份、浪费存储空间

1. mysqldump备份缺点

---效率较低、备份和还原速度慢、锁表

---备份过程中,数据插入和更新操作被阻塞

1.2安装percona

[root@room9pc17 mysql]# scp percona-xtrabackup-24-2.4.7-1.el7.x86\_64.rpm root@192.168.4.50:/root/

[root@room9pc17 mysql]# scp libev-4.15-1.el6.rf.x86\_64.rpm root@192.168.4.50:/root/

[root@host50 ~]# rpm -ivh libev-4.15-1.el6.rf.x86\_64.rpm

[root@host50 ~]# yum -y install percona-xtrabackup-24-2.4.7-1.el7.x86\_64.rpm

1.3常用选项

--host #主机名

--user #用户名

--port #端口号

--password #密码

--databases #数据库名

--no-timestamp #不用日期命名备份文件存储的子目录名

--redo-only #日志合并

--apply-log #准备恢复数据

--copy-back #拷贝数据

--incremental目录名 #增量备份

--incremental-basedir #增量备份时,指定上一次备份数据的存储目录名

--incremental-dir #准备恢复数据时,指定增量备份数据存储目录名

--export #导出表信息

Import #导入表空间

--databases=”库名” #一个库

--databases=”库1 库2” #多个库

--databases=”库1.表” #一张表

1.4命令格式

完全备份

]# innobackupex --user 用户名 --password 密码 备份目录名--no-timestamp

完全恢复

innobackupex --apply-log 目录名 #准备恢复数据

innobackupex --copy-back 目录名 #恢复数据

数据完全恢复步骤:

停止数据库服务

清空数据库目录

准备恢复数据

拷贝备份数据到数据库目录下

修改目录的所有者/组用户为mysql

启动服务

管理员登陆查看数据

使用完全备份文件恢复单张表的所有数据步骤：

删除表空间文件

在备份文件里，导出表信息

把导出表信息文件拷贝到数据库目录下

导入表信息

删除表信息文件

查看表记录

使用完全备份文件恢复单张表的相关命令:

> alter table 库名.表名 discard tablespace; #删除表空间

]# innobackupex --apply-log --export 数据完全备份目录 #导出表信息

]# cp 数据完全备份目录/数据库名目录/表名.{cfg,exp,ibd} 数据库目录/库名目录 #拷贝表文件信息

]# chown mysql:mysql 数据库目录/库名 #修改权限

> alter table 库名.表名 import tablespace; #导入表空间

> select \* from 库名.表名 ; #查看表记录

]# rm -rf 数据库目录/库名/表名.{cfg,exp} #删除表信息文件

## 二，增量备份与恢复

### 2.1增量备份

2.1.1周一 首次备份，备份所有数据

[root@host50 ~]# innobackupex --user root --password abc123...Q /fullbak --no-timestamp

2.1.2周二 增量备份

插入信息记录

[root@host50 ~]# innobackupex --user root --password abc123...Q --incremental /new1dir --incremental-basedir=/fullbak --no-timestamp

2.1.3周三 增量备份......

插入信息记录

[root@host50 ~]# innobackupex --user root --password abc123...Q --incremental /new2dir --incremental-basedir=/new1dir --no-timestamp

### 2.2增量恢复

2.2.1具体步骤

停止服务

清空数据库目录

准备恢复数据

合并数据

拷贝数据到数据库目录下

修改数据库目录的所有者和组用户为mysql

启动服务

查看数据

2.2.2具体案例

]# sytemctl stop mysqld

]# rm -rf /var/lib/mysql/\*

]# innobackupex --apply-log --redo-only /opt/fullbak

]# innobackupex --apply-log --redo-only /opt/fullbak --incremental-dir=/opt/new1dir

]# innobackupex --apply-log --redo-only /opt/fullbak --incremental-dir=/opt/new2dir

]# rm -rf /opt/new1dir/

]# rm -rf /opt/new2dir/

]# innobackupex --copy-back /opt/fullbak

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

]# systemctl start mysqld

RDBMS2\_DAY01

## 一、mysql主从同步

### 1.1 主从同步介绍？

实现数据自动同步的服务结构

主服务器:接收客户端访问连接

从服务器:自动同步主服务器数据

### 1.2 工作过程？

master:启用binlog日志

slave:

Slave\_IO:复制master主机binlog日志文件里的SQL命令到主机的relay\_log文件里.

Slave\_SQL:执行本机relay\_log文件里的SQL语句,实现与master数据一致.

### 1.3 配置mysql主从同步

1.3.1 拓扑结构

1.3.2 配置主服务器

思路:启用binlog日志,重启服务,授权用户,查看binlog日志信息

[root@host51 ~]# vim /etc/my.cnf

[mysqld]

server\_id=51

log\_bin=master51

:wq

[root@host51 ~]# systemctl restart mysqld

[root@host51 ~]# ls /var/lib/mysql/master51.\*

[root@host51 ~]# mysql -uroot -p123456

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

mysql> show master status;

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

| File | Position | Binlog\_Do\_DB | Binlog\_Ignore\_DB | Executed\_Gtid\_Set |

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

| master51.000001 | 441 | | | |

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

1 row in set (0.00 sec)

1.3.3 配置从服务器

思路:设置server\_id

确保与主服务器数据一致

指定主库信息

启动slave程序

查看状态信息

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

[mysqld]

server\_id=52

:wq

[root@52 ~]# systemctl restart mysqld

[root@51 ~]# mysqldump -uroot -p123qqq...A --master-data db5 > /root/db5.sql

[root@51 ~]# scp /root/db5.sql root@192.168.4.52:/opt/

[root@52 ~]# mysql -uroot -p123qqq...A -e 'create database db5'

[root@52 ~]# mysql -uroot -p123qqq...A db5 < /opt/db5.sql

[root@52 ~]# mysql -uroot -p123qqq...A -e 'use db5 ; show tables'

[root@52 ~]# grep master51 /opt/db5.sql

CHANGE MASTER TO MASTER\_LOG\_FILE='master51.000001', MASTER\_LOG\_POS=441;

[root@52 ~]# mysql -uroot -p123qqq...A

mysql> change master to master\_host="192.168.4.51" , master\_user="repluser",

-> master\_password="123qqq...A" , master\_log\_file="master51.000001",master\_log\_pos=441;

mysql> start slave ;

mysql> show slave status\G;

Master\_Host: 192.168.4.51

Slave\_IO\_Running: Yes

Slave\_SQL\_Running: Yes

相关文件

master.info 主库信息

relay-log.info 中继日志信息

主机名-relay-bin.xxxxxx 中继日志

主机名-relay-bin.index 索引文件

1.4 排错

Last\_IO\_Error: IO线程报错信息

Last\_SQL\_Error:SQL线程报错信息

stop slave ;

根据提示解决问题

start slave;

show slave status\G;

终极解决办法：

删除相关文件的4种文件后 重启数据库服务，重新指定主服务器信息。

1.5 验证配置

1.5.1 添加授权用户（给客户端连接使用）

[root@host51 ~]# mysql -uroot -p123qqq...A

mysql> create database bbsdb ;

mysql> grant all on bbsdb.\* to yaya108@"%" identified by "123qqq...A";

mysql> grant all on db5.\* to yaya108@"%" identified by "123qqq...A";

1.5.2 客户端使用授权用户连接主服务器51 对数据做访问

[root@mysql50 ~]# mysql -h192.168.4.51 -uyaya108 -p123qqq...A

mysql> show grants;

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

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

mysql> select \* from bbsdb.a;

+------+

| id |

+------+

| 100 |

+------+

mysql>

1.5.3 在从服务器本机能够查看到和主服务器一样的数据

[root@mysql52 ~]# mysql -uroot -p123qqq...A -e 'select \* from bbsdb.a'

## 二、主从同步模式

### 2.1主从同步结构模式

2.1.1 主从同步结构模式 ?

2.1.2 配置mysql一主多从结构

要求：把数据库服务器53 也配置为51的从服务器

配置从服务器 192.168.4.53 步骤如下:

[root@host53 ~]# vim /etc/my.cnf

[mysqld]

server\_id=53

:wq

[root@host53 ~]# systemctl restart mysqld

[root@host51 ~]# mysqldump -uroot -p123qqq...A --master-data -B db5 bbsdb > /root/bbsdb.sql

[root@host51 ~]# ls /root/\*.sql

/root/bbsdb.sql

[root@host51 ~]# scp /root/bbsdb.sql root@192.168.4.53:/root/

[root@host53 ~]# mysql -uroot -p123456 -e 'create database bbsdb'

[root@host53 ~]# mysql -uroot -p123456 bbsdb < /root/bbsdb.sql

[root@host53 ~]# mysql -uroot -p123456 -e 'use bbsdb ;show tables'

[root@host53 ~]# grep master51 /root/bbsdb.sql

CHANGE MASTER TO MASTER\_LOG\_FILE='master51.000003', MASTER\_LOG\_POS=1315;

[root@host53 ~]# mysql -uroot -p123456

mysql> change master to master\_host="192.168.4.51" , master\_user="repluser" , master\_password="123qqq...A" , master\_log\_file="master51.000003" , master\_log\_pos=1315;

mysql> start slave ;

mysql> show slave status\G;

Master\_Host: 192.168.4.51

Slave\_IO\_Running: Yes

Slave\_SQL\_Running: Yes

2.1.3 配置mysql主从从结构

2.1.4 配置mysql主主结构(2台数据库服务器互为主从)

要求：把数据库服务器192.168.4.56和192.168.4.57

2台服务器配置为互为主从结构。

### 2.2主从同步复制模式

2.2.1 主从同步复制模式

介绍？

2.2.2 配置半同步复制模式 （192.168.4.54）

命令行配置

1 查看是否允许动态加载模块

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

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

| Variable\_name | Value |

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

| have\_dynamic\_loading | YES |

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

1 row in set (0.00 sec)

mysql>

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";

3 查看模块是否加载

mysql> select plugin\_name, plugin\_status from information\_schema.plugins

where plugin\_name like '%semi%';

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

| plugin\_name | plugin\_status |

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

| rpl\_semi\_sync\_master | ACTIVE |

| rpl\_semi\_sync\_slave | ACTIVE |

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

2 rows in set (0.00 sec)

4 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;

5 查看是否启用半同步复制

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

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

| Variable\_name | Value |

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

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

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

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

永久配置 （/etc/my.cnf）

]# vim /etc/my.cnf

[mysqld]

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

:wq

[root@host54 ~]# systemctl restart mysqld

]# mysql -uroot -p123qqq...A

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

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

| Variable\_name | Value |

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

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

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

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

2 rows in set (0.00 sec)

RDBMS2\_DAY02

## 一、数据读写分离

### 1.1 数据读写分离 ？

原理:由mysql代理面向客户端提供服务

--收到SQL写请求时,交给master服务器处理

--收到SQL读请求时,交给slave服务器处理

构建思路:

1.1.1部署mysql一主一从结构

1.1.2部署mysql代理服务器

--装包,修改配置文件,启动服务

1.1.3测试配置

` --客户端连接代理服务访问数据

### 1.2 部署maxscale服务

1.2.1 安装软件

[root@57 ~]# netstat -utnlp | grep :3306

[root@57 ~]# systemctl stop mysqld ; systemctl disable mysqld

]# rpm -ivh maxscale-2.1.2-1.rhel.7.x86\_64.rpm

配置文件

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

/etc/maxscale.cnf

命令

maxadmin maxbinlogcheck maxpasswd

maxavrocheck maxkeys maxscale

日志文件

[root@57 ~]# ls /var/log/maxscale/

[root@57 ~]# rpm -qa | grep -i maxscale

maxscale-2.1.2-1.x86\_64

服务程序

[root@57 ~]# rpm -ql maxscale

1.2.2 修改主配置文件

[root@57 ~]# cp /etc/maxscale.cnf /root/ #备份,防止改错

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

[maxscale]

threads=auto #定义线程个数,auto为根据服务器CPU核数自适应

[server1] #定义数据库服务器

type=server

address=192.168.4.51 #master主机IP地址

port=3306

protocol=MySQLBackend

[server2] #定义数据库服务器

type=server

address=192.168.4.52 #slave主机IP地址

port=3306

protocol=MySQLBackend

[MySQL Monitor] #定义要监视的数据库节点

type=monitor

module=mysqlmon

servers=server1, server2 #主,从数据库的主机名

user=maxscalemon #监控用户

passwd=123qqq...A #监控用户密码

monitor\_interval=10000

[Read-Write Service] #定义读写分离的数据库节点

type=service

router=readwritesplit

servers=server1, server2 #主,从数据库的主机名

user=maxscalerouter #路由用户

passwd=123qqq...A #密码

max\_slave\_connections=100%

[MaxAdmin Service] #定义管理服务

type=service

router=cli

[Read-Write Listener] #定义读写分离服务端口号

type=listener

service=Read-Write Service

protocol=MySQLClient

port=4006

[MaxAdmin Listener] #定义管理服务端口号

type=listener

service=MaxAdmin Service

protocol=maxscaled

socket=default

port=4016

1.2.3 配置数据库服务器（ server1 和 server2 ）

maxscalemon 用户授权(监控用户)

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

maxscalerouter用户授权(路由用户)

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

1.2.4 启动服务

]# maxscale -f /etc/maxscale.cnf

]# killall -9 maxscale //停止服务命令

1.2.5 查看服务信息

]# netstat -utnlp | grep :4016

]# netstat -utnlp | grep :4006

]# netstat -utnlp | grep maxscale

]# ls /var/log/maxscale/ 日志文件目录

]# vim /var/log/maxscale/maxscale.log 查看日志内容

1.2.6 查看监控信息

[root@host57 ~]# maxadmin -uadmin -pmariadb -P4016

MaxScale>

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

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

MaxScale> exit

1.2.7 测试配置

1.2.7.1 在主服务器51 ， 添加客户端访问数据的连接用户

host51]# mysql -uroot -p123qqq...A

mysql> create database db7;

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

mysql> grant select , insert on db7.\* to plj99@"%" identified by "123qqq...A";

host52]# mysql -uroot -p123qqq...A -e 'show grants for plj99@"%"'

1.2.7.2 在客户端连接服务器57 访问数据

client50]# mysql -h192.168.4.57 -P4006 -uplj99 -p123qqq...A

mysql> show grants;

mysql> select \* from db7.a;

mysql> insert into db7.a values(100);

mysql> select \* from db7.a;

1.2.7.3 测试数据读写分离

在从服务器52本机向表里插入记录

]# mysql -uroot -p123qqq...A

mysql> insert into db7.a values(52);

mysql> select \* from db7.a;

在主服务器51本机查看表记录

mysql> select \* from db7.a;

在客户端50连接服务器57 访问数据

client50]# mysql -h192.168.4.57 -P4006 -uplj99 -p123qqq...A

## 二、mysql多实例

准备主机 克隆1台新的虚拟机配置Ip地址 eth0 192.168.4.58/24

软件 mysql-5.7.20-linux-glibc2.12-x86\_64.tar.gz

### 2.1 mysql多实例 介绍

概念:在一台物理主机上运行多个数据库服务

优点:节约运维成本,提高硬件利用率

### 2.2配置mysql多实例

步骤如下：

1 安装软件

依赖软件:libaio

PATH=/usr/local/mysql/bin:$PATH #临时修改

永久修改写入 /etc/baserc文件中

2 创建并编辑主配置文件

---每个实例要有独立:数据库目录,端口,socket文件,pid文件,错误日志文件

]# vim /etc/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] #实例进程名称,1表示实例编号

datadir=/dir1 #端口号

port=3307 #数据库目录,要手动创建

socket=/dir1/mysqld1.sock #指定sock文件的路径和名称

log-error=/dir1/mysqld1.err #进程pid 号文件位置

pid-file=/dir1/mysqld1.pid #错误日志位置

[mysqld2] #同上

datadir=/dir2

port=3308

socket=/dir2/mysqld2.sock

log-error=/dir2/mysqld2.err

pid-file=/dir2/mysqld2.pid

[root@host58 ~]#

手动创建数据库目录:

[root@host58 ~]# mkdir /dir1

[root@host58 ~]# mkdir /dir2

3 启动多实例服务

3.1 启动编号1的实例服务

]# mysqld\_multi start 1 #启动服务

]# ls /dir1

]# netstat -utnlp | grep :3307

]# mysql -uroot -p'zdkk%:q6HCe9' -S /dir1/mysqld1.sock #停止服务

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

mysql> exit;

]# mysql -uroot -p123456 -S /dir1/mysqld1.sock

]# mysqld\_multi --user=root --password=123456 stop 1

]# netstat -utnlp | grep :3307

3.2 启动编号2 的实例服务

]# mysqld\_multi start 2 #启动服务

]# ls /dir2

]# netstat -utnlp | grep :3308

]# mysql -uroot -p'zdkk%:q6HCe9' -S /dir2/mysqld2.sock #停止服务

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

mysql> exit;

]# mysql -uroot -p123456 -S /dir2/mysqld2.sock

]# mysqld\_multi --user=root --password=123456 stop 2

]# netstat -utnlp | grep :3308

RDBMS2\_DAY03

## 一．数据分片相关概念

### 1.1分库分表:

将存放在一台数据库服务器中的数据,按照特定的方式进行拆分,分散存放到多台数据库服务器,以达到分散单台服务器负载的效果.

### 1.2水平分割:

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.

### 1.3垂直分割:

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.

## 二． MyCat软件介绍

基于Java的分布式数据库系统的中间件,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:

--枚举法

## 三． 配置数据分片

### 3.1 配置数据分片服务器

3.1.1 安装软件

3.1.2 配置文件说明

3.1.3 修改配置文件

3.1.3.1 配置客户端连接使用的用户及密码

]# vim server.xml (使用默认配置)

TESTDB

3.1.3.2 配置数据分片

]# cp schema.xml /opt/

]# sed -i '56,77d' schema.xml

]# sed -i '39,42d' schema.xml

]# sed -i '16,18d' schema.xml

]# sed -i '45,47d' schema.xml

]# vim schema.xml

### 3.2 配置数据库服务器

3.2.1 用户授权

3.2.2 创建存储数据库

[root@host53 ~]# mysql -uroot -p123qqq...A -e 'create database db1'

[root@host53 ~]# mysql -uroot -p123qqq...A -e 'grant all on \*.\* to admin@"%" identified by "123qqq...A"'

[root@host54 ~]# mysql -uroot -p123qqq...A -e 'create database db2'

[root@host54 ~]# mysql -uroot -p123qqq...A -e 'grant all on \*.\* to admin@"%" identified by "123qqq...A"'

[root@host55 ~]# mysql -uroot -p123qqq...A -e 'create database db3'

[root@host55 ~]# mysql -uroot -p123qqq...A -e 'grant all on \*.\* to admin@"%" identified by "123qqq...A"'

## 四． 启动mycat服务

[root@host56 conf]# /usr/local/mycat/bin/mycat -help

[root@host56 conf]# /usr/local/mycat/bin/mycat status

[root@host56 conf]# /usr/local/mycat/bin/mycat start

[root@host56 conf]# ps -C java

[root@host56 conf]# netstat -utnlp | grep :8066

[root@host56 conf]# ls /usr/local/mycat/logs/

## 五． 查看服务信息

查看进程

[root@host56 conf]# ps -C java

查看端口

[root@host56 conf]# netstat -utnlp | grep :8066

日志文件

[root@host56 conf]# ls /usr/local/mycat/logs/

## 六． 客户端50 连接分片服务器56 访问数据

[root@host50 ~]# mysql -h192.168.4.56 -P8066 -uroot -p123456

> show databases;

> use TESTDB;

> show tables;

> desc goods;

7 排错时间

自行排错时间

8 分片规则的使用

枚举法sharding-by-intfile: 字段值必须在列举范围内选择

]# vim schema.xml

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

rule="sharding-by-intfile" />

]# 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>

:wq

]# vim /usr/local/mycat/conf/partition-hash-int.txt

10000=0

10010=1

10020=2

:wq

]# /usr/local/mycat/bin/mycat stop

]# /usr/local/mycat/bin/mycat start

]# netstart -utnlp | grep ；8066

[root@host50 ~]# 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) , age tinyint ,sex enum("man","woman"));

mysql> insert into employee(sharding\_id , name , age , sex)

values(10000,"bob",19,"woman");

mysql> insert into employee(sharding\_id , name , age , sex)

values(10010,"bob",19,"woman");

mysql> insert into employee(sharding\_id , name , age , sex)

values(10020,"bob",19,"woman");

mysql> insert into employee(sharding\_id , name , age , sex)

values(10020,"jim",19,"woman");

mysql> insert into employee(sharding\_id , name , age , sex)

values(10020,"lucy",19,"woman");

mysql> select \* from employee;

求模法mod\_long: 根据字段值与设定的数字求模结果存储数据

]# vim schema.xml

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

:wq

]# vim rule.xml

<tableRule name="mod-long">

<rule>

<columns>id</columns>

<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>

</function>

:wq

[root@host56 ~]# /usr/local/mycat/bin/mycat stop

[root@host56 ~]# /usr/local/mycat/bin/mycat start

[root@host50 ~]# mysql -h192.168.4.56 -P8066 -uroot -p123456

mysql> use TESTDB;

Database changed

mysql> create table hotnews(

id int , title char(50),worker char(10), comment varchar(150));

insert into hotnews(id,title,worker,comment) values(7,"linux","redhat","linux-lb");

insert into hotnews(id,title,worker,comment) values(8,"linux","redhat","linux-lb");

insert into hotnews(id,title,worker,comment) values(9,"linux","redhat","linux-lb");

insert into hotnews(id,title,worker,comment) values(6,"linux","redhat","linux-lb");

type=global 数据不分片存储

]# vim schema.xml

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

:wq

[root@host50 ~]# mysql -h192.168.4.56 -P8066 -uroot -p123456

mysql> use TESTDB;

mysql> create table goods(ID int primary key ,name char(15),class char(9));

mysql> insert into goods(ID ,name , class)values(1,"tom","nsd1906");

mysql> insert into goods(ID ,name , class)values(2,"jim","nsd1906");

mysql> insert into goods(ID ,name , class)values(3,"max","nsd1906");

9 添加新库/新表

9.1 添加新库

]# vim server.xml

<user name="root">

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

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

<user name="user">

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

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

:wq

9.2 添加新表

]# vim schema.xml

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

<table name="company2" primaryKey="ID" type="global" dataNode="dn1,dn2,dn3" />

<table name="employee2" primaryKey="ID" dataNode="dn1,dn2,dn3"

rule="sharding-by-intfile" />

<table name="hotnews2" primaryKey="ID" autoIncrement="true" dataNode="dn1,dn2,dn3" rule="mod-long" />

</schema>

:wq

9.3 重启服务

[root@host56 ~]# /usr/local/mycat/bin/mycat stop

[root@host56 ~]# /usr/local/mycat/bin/mycat start

[root@host56 ~]# netstat -utnlp | grep :8066

9.4 测试配置

[root@host50 ~]# mysql -h192.168.4.56 -P8066 -uroot -p123456

mysql> show databases;

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

| DATABASE |

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

| BBSDB |

| TESTDB |

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