# 数据库服务器

目录

[数据库服务器 2](#_Toc31766)

[第一章 简介 2](#_Toc17762)

[1、什么是数据库? 2](#_Toc5321)

[2、互联网上的常用数据库 2](#_Toc931)

[3、mariadb数据库简介 2](#_Toc17176)

[4、mariadb与mysql的区别： 2](#_Toc14113)

[第二章 Mariadb 3](#_Toc127)

[（1）安装mariadb数据库 3](#_Toc15847)

[（2） 数据库连接及安全初始化 3](#_Toc23381)

[（3） 更改密码 4](#_Toc23429)

[（4） mariadb忘记root密码 4](#_Toc1337)

[（5） 提高mariadb的安全性 4](#_Toc11842)

[（6） 数据库的管理 5](#_Toc4591)

[「1」建立scott数据库快速建立脚本 5](#_Toc5886)

[「2」更改字符集 8](#_Toc21053)

[「3」查数据 9](#_Toc28974)

[「4」创建表 9](#_Toc21118)

[「5」修改表 10](#_Toc9103)

[「6」删除表 10](#_Toc779)

[「7」外部表 10](#_Toc6012)

[「8」在linux环境备份数据库 11](#_Toc168)

[「9」用户和权限 12](#_Toc21529)

[第三章 mariadb主从复制 12](#_Toc26004)

[第四章 mycat读写分离 13](#_Toc9134)

1. **简介**

**1、什么是数据库?**

数据库是“按照数据结构来组织、存储和管理数据的仓库”。是一个长期存储在计算机内的、有组织的、可共享的、统一管理的大量数据的集合。

数据库是以一定方式储存在一起、能与多个用户共享、具有尽可能小的冗余度、与应用程序彼此独立的数据集合，可视为[电子化]的文件柜——存储电子文件的处所，用户可以对文件中的数据进行新增、查询、更新、删除等操作

**2、互联网上的常用数据库**

Oracle

SQL Server

mysql

DB2

**3、mariadb数据库简介**

自甲骨文公司收购MySQL后，其在商业数据库与开源数据库领域市场的占有份额都跃居第一，这样的格局引起了业内很多的人士的担忧，因为商业数据库的老大有可能将MySQL闭源。为了避免Oracle将MySQL闭源，而无开源的类MySQL数据库可用，MySQL社区采用分支的方式来避开这个风险。MariaDB数据库就这样诞生了，MariaDB是一个向后兼容，可能在以后替代MySQL的数据库产品，其官方地址为：https://mariadb.org/ 。mariadb和mysql几乎是一样的。首先，mariadb就是由mysql的创始人负责维护的。而mariadb就是mysql创始人女儿的名字。使用方法和mysql相同

**4、mariadb与mysql的区别：**

1. MariaDB不仅仅是Mysql的一个替代品，MariaDB包括的一些新特性使它优于MySQL
2. MariaDB跟MySQL在绝大多数方面是兼容的，对于开发者来说，几乎感觉不到任何不同。

* 目前MariaDB是发展最快的MySQL分支版本，新版本发布速度已经超过了Oracle官方的MySQL版本。
* MariaDB 是一个采用Aria存储引擎的MySQL分支版本， 这个项目的更多的代码都改编于 MySQL 6.0

1. 通过全面测试发现，MariaDB的查询效率提升了3%-15%，平均提升了8%，而且没有任何异常发生；

* 以qp为单位，吞吐量提升了2%-10%。

MariaDB虽然被视为MySQL数据库的替代品，但它在扩展功能、存储引擎以及一些新的功能改进方面都强过MySQL。而且从MySQL迁移到MariaDB也是非常简单的：

1、数据和表定义文件（.frm）是二进制兼容的

2、所有客户端API、协议和结构都是完全一致的

3、所有文件名、二进制、路径、端口等都是一致的

4、所有的MySQL连接器，比如PHP、Perl、Python、Java、.NET、MyODBC、Ruby以及MySQL C connector等在MariaDB中都保持不变

5、mysql-client包在MariaDB服务器中也能够正常运行

6、共享的客户端库与MySQL也是二进制兼容的

1. **Mariadb**

**（1）安装mariadb数据库**

apt install mariadb-server #安装mariadb服务端程序  
systemctl restart mariadb.service #启动程序  
systemctl enable mariadb.service #设为开机自启动

1. **数据库连接及安全初始化**

**连接数据库**

mysql工具是MySQL官方提供的连接工具，用户可以通过mysql连接到mysqld上进行一系列的SQL操作。

mysql工具有两种模式：交互模式和命令行模式。交互模式指令需要连接到mysql服务器下达，命令行模式通过特定（-e）参数读取shell命令行传递的指令到服务器。

命令格式：  
mysql [OPTIONS] [database]  
 -h, --help 显示帮助信息  
 -e, --execute=name 执行指定命令，命令行模式  
 -u, --user=name 指定连接的用户  
 -h, --host=name 指定主机名  
 -p, --password 指定密码  
 -P, --port 指定端口

示例：

（1）使用root账号连接服务器，无密码登录  
mysql -u root  
（2）使用root账号连接服务器，使用密码登录  
mysql -u root -p'passwd'

1. **更改密码**

mysqladmin -u root password '123456'  
mysql -u root -p'123456'  
  
在linux中  
修改密码  
mysqladmin -u root -p password '123456' #需输入root原始密码  
  
在数据库中修改  
set password=password('uos'); #对当前用户设置密码，立刻生效  
set password for uos@'localhost'=password('123456'); #对任意用户设置密码  
use mysql  
update user set password=password('uos') where user='root' and host='localhost'; #修改密码后需要flush privileges;刷新权限表，或重启服务

1. **mariadb忘记root密码**

vim /etc/mysql/mariadb.conf.d/50-server.cnf  
[mysqld]  
skip-grant-tables \#在\[mysqld\]下添加此字段，进入但用户模式  
systemctl restart mariadb.service  
输入mysql命令无需密码直接进入数据库  
use mysql  
update user set password=password(\'uos\') where user=\'root\';  
从5.5.7版本中mysql数据库中就开始引入plugin这项配置，用来进行用户密码验证  
update user set plugin=\'\' where user=\'root\';  
停止mariadb服务，删除配置文件的skip-grant-tables选项，重启mariadb服务，使用新密码进入

1. **提高mariadb的安全性**

1、数据库的起始状态设定信息是不安全的，需要做以下设定：

mysql\_secure\_installation  
Enter current password for root (enter for none): \#输入当前root的密码，密码为空直接回车即可  
Change the root password? \[Y/n\] \#是否修改root的密码  
Remove anonymous users? \[Y/n\] \#是否删除匿名用户  
Disallow root login remotely? \[Y/n\] \#是否不允许root远程登录  
Remove test database and access to it? \[Y/n\] \#是否删除测试数据库test  
Reload privilege tables now? \[Y/n\] \#是否重新加载授权信息  
  
systemctl restart mariadb

2、默认情况下，数据库的网络接口时打开的，为了安全起见，需要关闭此接口，改配置文件

vim /etc/mysql/mariadb.conf.d/50-server.cnf  
[mysqld] #在msqld模块写入  
skip-networking=1  
  
systemctl restart mariadb #重启系统数据库  
netstat -anplut | grep mysql #查询不到任何监听端口

ls -l /usr/share/mysql/ #有4个规模的配置文件模板

vim /etc/mysql/my.cnf #配置文件

ls -l /var/lib/mysql #每个数据库会在此文件夹下建立一个单独目录

1. **数据库的管理**

SQL语言可以理解成向服务端提问的语言，SQL语言按照功能可以分为几种子语言：  
 数据定义语言（DDL）：创建和管理数据库，包括创建修改表，定义索引，管理约束条件。DDL操作对象为数据库内部的对象。  
 数据操纵语言（DML）：查询和更新数据库中的数据,用于添加，删除，更新，查询。DML的操作对象是表的内部数据。而不会涉及到表的定义，结构的修改。  
 数据控制语言（DCL） ：对用户的权限控制  
 数据查询语言（dql）：对数据库查询

**「1」建立scott数据库快速建立脚本**

SCOTT是数据库内部的一个示例用户，缺省口令为tiger，下面有表emp, dept等，这些表和表间的关系演示了关系型数据库的一些基本原理

vim /scott.sql  
create database scott;  
   
use scott  
create table dept(  
 -- 部门编号  
 deptno int unsigned auto\_increment primary key,  
 -- 部门名称  
 dname varchar(15) ,  
 -- 部门所在位置  
 loc varchar(50)   
)engine = InnoDB;  
   
create table emp(  
 -- 雇员编号  
 empno int unsigned auto\_increment primary key,  
 -- 雇员姓名  
 ename varchar(15) ,  
 -- 雇员职位  
 job varchar(10) ,  
 -- 雇员对应的领导的编号  
 mgr int unsigned ,  
 -- 雇员的雇佣日期  
 hiredate date ,  
 -- 雇员的基本工资  
 sal decimal(7,2) ,  
 -- 奖金  
 comm decimal(7,2) ,  
 -- 所在部门  
 deptno int unsigned ,  
 foreign key(deptno) references dept(deptno)  
)engine = innodb;  
   
create table salgrade(  
 -- 工资等级  
 grade int unsigned ,  
 -- 此等级的最低工资  
 losal int unsigned ,  
 -- 此等级的最高工资  
 hisal int unsigned   
)engine=innodb;  
   
create table bonus(  
 -- 雇员姓名  
 ename varchar(10),  
 -- 雇员职位  
 job varchar(9),  
 -- 雇员工资  
 sal decimal(7,2),  
 -- 雇员资金  
 comm decimal(7,2)  
)engine=innodb;  
   
INSERT INTO dept VALUES (10,'ACCOUNTING','NEW YORK');  
INSERT INTO dept VALUES (20,'RESEARCH','DALLAS');  
INSERT INTO dept VALUES (30,'SALES','CHICAGO');  
INSERT INTO dept VALUES (40,'OPERATIONS','BOSTON');  
   
INSERT INTO emp VALUES (7369,'SMITH','CLERK',7902,'1980-12-17',800,NULL,20);  
INSERT INTO emp VALUES (7499,'ALLEN','SALESMAN',7698,'1981-2-20',1600,300,30);  
INSERT INTO emp VALUES (7521,'WARD','SALESMAN',7698,'1981-2-22',1250,500,30);  
INSERT INTO emp VALUES (7566,'JONES','MANAGER',7839,'1981-4-2',2975,NULL,20);  
INSERT INTO emp VALUES (7654,'MARTIN','SALESMAN',7698,'1981-9-28',1250,1400,30);  
INSERT INTO emp VALUES (7698,'BLAKE','MANAGER',7839,'1981-5-1',2850,NULL,30);  
INSERT INTO emp VALUES (7782,'CLARK','MANAGER',7839,'1981-6-9',2450,NULL,10);  
INSERT INTO emp VALUES (7788,'SCOTT','ANALYST',7566,'87-7-13',3000,NULL,20);  
INSERT INTO emp VALUES (7839,'KING','PRESIDENT',NULL,'1981-11-17',5000,NULL,10);  
INSERT INTO emp VALUES (7844,'TURNER','SALESMAN',7698,'1981-9-8',1500,0,30);  
INSERT INTO emp VALUES (7876,'ADAMS','CLERK',7788,'87-7-13',1100,NULL,20);  
INSERT INTO emp VALUES (7900,'JAMES','CLERK',7698,'1981-12-3',950,NULL,30);  
INSERT INTO emp VALUES (7902,'FORD','ANALYST',7566,'1981-12-3',3000,NULL,20);  
INSERT INTO emp VALUES (7934,'MILLER','CLERK',7782,'1982-1-23',1300,NULL,10);  
   
INSERT INTO salgrade VALUES (1,700,1200);  
INSERT INTO salgrade VALUES (2,1201,1400);  
INSERT INTO salgrade VALUES (3,1401,2000);  
INSERT INTO salgrade VALUES (4,2001,3000);  
INSERT INTO salgrade VALUES (5,3001,9999);

调用系统资源

mysql -uroot -p  
system ls -l /root  
source /scott.sql

**「2」更改字符集**

更改字符集  
查看Linux的字符集用locale命令  
status #查看mariadb属性  
Server characterset: utf8mb4  
Db characterset: utf8mb4  
Client characterset: utf8mb4  
Conn. characterset: utf8mb4  
  
客户端的字符集要和数据库的字符集一致，不一致有可能乱码  
vim /etc/mysql/mariadb.conf.d/50-server.cnf  
character-set-server = utf8  
#collation-server = utf8  
  
systemctl restart mariadb.service  
MariaDB [(none)]> status #重启后变更为新字符集  
Server characterset: utf8  
  
查看数据库的字符集  
create database scott;  
drop database scott; #删除数据库  
source /scott.sql #使用脚本重建数据库  
show create database scott; #在更改字符集之后建立，所以是字符集是utf8  
show create table emp;  
更改字符集  
alter database scott charset utf8;  
alter table emp charset utf8;

select database(); #查看当前进入的数据库  
select user(); #查看当前登录用户  
show databases; #查看数据库  
use mysql #进入mysql数据库  
show tables; #查看此数据库的表，test没表，可以看其他数据库的  
desc user; #查看表结果  
show variables like 'innodb%'; #查看环境变量

有时命令输错停在>界面出不来，一般是少些了'或者；补全即可退出，实在不行可以CTRL+D在重新登录

**「3」查数据**

select \* from emp;  
select ename,sal from emp;  
select ename,(sal+200)\*3 as bonus from emp; #支持算数表达式加减乘数和括号等  
select ename,sal,comm,sal+ifnull(comm,0) as income from emp; #空值参与算术运算为空值，建议用ifnull函数转换  
select distinct deptno from emp; #用distinct去除重复结果  
select ename,sal,deptno from emp where deptno=30;  
select \* from emp where ename like '\_\_\_TT'; #\_代表任意单个字符  
select \* from emp where ename like '%LL%'; #%代表任意字符  
select ename,sal,deptno from emp where deptno=30 and sal>2000; #与  
select ename,sal,deptno from emp where deptno=30 or sal>2000; #或  
select ename,sal,deptno from emp where not sal>2000; #非  
select \* from emp order by sal; #默认升序，降序加desc  
select empno,ename,sal,deptno from emp order by deptno,sal desc; #多列排序  
select concat(ename,'\'s sal is ',sal) from emp; #支持多种函数，不一一举了  
select count(\*) from emp;  
select deptno,sum(sal),min(sal),max(sal),avg(sal) from emp group by deptno;  
select deptno,avg(sal) from emp group by deptno having avg(sal)<2000;  
select ename,dname from emp,dept; #笛卡尔乘积  
select ename,dname from emp,dept where emp.deptno=dept.deptno;  
select ename,sal,grade from emp join salgrade on sal between losal and hisal;  
select y.ename yuangong,j.ename jingli from emp y,emp j where y.mgr=j.empno;  
select ename,dname,sal,grade from emp,dept,salgrade where emp.deptno=dept.deptno and emp.sal between salgrade.losal and salgrade.hisal;  
select ename,sal from emp where sal=(select max(sal) from emp); #子查询找到工资最高薪  
select ename from emp where empno not in (select mgr from emp where mgr is not null); #in结果集不能有空值，否则结果为空

创建数据库

MariaDB [(none)]> create database uosdata;

**「4」创建表**

创建表并建立字段

格式：create table 数据表名 (字段名 字段类型);  
  
MariaDB [uos]> create table uostable(id int,name varchar(10),mail varchar(30));  
int类型表示正常大小的整数（数字数据类型）。  
char类型（字符串数据类型）表示包含指定长度的空格的右侧带有固定长度的字符串。 M表示字符的列长度，取值范围为0〜255，缺省值为1。  
varchar类型（字符串数据类型）表示一个可变长度字符串，M范围（最大列长度）为0到65535。

**「5」修改表**

create table uos1 like uostable; \#没有数据，只是复制了表的结构  
insert into uos1 select \\* from uostable; \#复制数据，表不存在无法复制  
create table uos2 as select \\* from uostable; \#新建表，并复制整个表结构+数据

查看表结构

格式：desc 数据表名;  
  
MariaDB [uos]> desc uos;

向数据表插入数据

格式：insert into 数据表名(id,name,mail) values(1,'uos1','uos1\@uos.com');  
  
insert into uostable(id,name,mail) values(1,'uos1','uos1@uos.com');  
insert into uostable values(2,'uos2','uos2@uos.com'); #加入全列数据可以不写列名  
insert into uostable(id,name) values(3,'uos3'); #不是全列必须写列名  
insert into uostable values(4,'uos4','uos4@uos.com'),(5,'uos5','uos5@uos.com'),(6,'uos6','uos6@uos.com');

更新表

update uostable set name=\'test\' where id=2;

修改表结构

alter table uostable add newlist varchar(20); #增加newlist列  
alter table uostable drop newlist; #删除newlist列  
alter table uostable add firstlist varchar(20) first; #增加firstlist列到第一列  
alter table uostable add afterid varchar(30) after id; #增加afterid列到id列后面

**「6」删除表**

delete from uostable where id=4; #要加上where约束，没有where删除整个表  
delete from uos1; #DML操作，清除表数据，保留表结果  
truncate uos2; #DDL操作，清除表数据，保留表结果，更彻底，降低高水位线  
drop table uos1; #清除表结构和数据

**「7」外部表**

建立测试用外部表

vim /uos.txt  
  
1,uos1,uos1@uos.com  
  
2,uos2,uos2@uos.com  
  
3,uos3,uos3@uos.com  
  
4,uos4,uos4@uos.com

将外部表导入数据库

create database uosdatabase;  
create table uostable(id int(4),name varchar(10),email varchar(20));  
load data infile '/uos.txt' into table uostable fields terminated by ',' lines terminated by '\n';

将数据库导出为外部表

select \* from uostable into outfile '/var/lib/mysql/uosdatabase/uostable.txt' fields terminated by ',' lines terminated by '\n';

通过外部表导入scott数据库

将scott数据库的3个文本文件拷贝到根目录

vim /scott.emp.txt  
7369,'SMITH','CLERK',7902,800,200,20  
7499,'ALLEN','SALES',7698,1600,300,30  
7521,'WARD','SALES',7698,1250,500,30  
7566,'JONES','MANAG',7839,2975,100,20  
  
vim /scott.dept.txt  
10,'ACCOUNTING','NEW YORK'  
20,'RESEARCH','DALLAS'  
30,'SALES','CHICAGO'  
40,'OPERATIONS','BOSTON'  
  
vim /scott.salgrade.txt  
1,700,1200  
2,1201,1400  
3,1401,2000  
  
create database scott;  
use scott  
create table emp(empno int(4),ename varchar(10),job varchar(9),mgr int(4) default null,sal int(7),comm int(7) default null,deptno int(2));  
load data infile '/scott.emp.txt' into table emp fields terminated by ',' lines terminated by '\n';  
create table dept(deptno int(2),dname varchar(14),loc varchar(13));  
load data infile '/scott.dept.txt' into table dept fields terminated by ',' lines terminated by '\n';  
create table salgrade(grade int,losal int,hisal int);  
load data infile '/scott.salgrade.txt' into table salgrade fields terminated by ',' lines terminated by '\n';

将scott数据库导出为外部表

use scott  
select \* from emp into outfile '/var/lib/mysql/scott/scott.emp.txt' fields terminated by ',' lines terminated by '\n';

**「8」在linux环境备份数据库**

mysqldump -u root -p scott > /scott.dump #备份数据库  
mysqldump -u root -p scott emp > /scott.emp.dump #备份数据库中的表  
mysqldump -u root -p scott dept salgrade > /scott.dept+salgrade.dump #备份数据库中的多个表

在linux环境还原数据库

>drop database scott;  
#如果报错，删除/var/lib/mysql/scott/scott.emp.txt文件  
>create database uos; #不建立数据库，无法导入整库,随意命名  
mysql -u root -p uos < /scott.dump #还原数据库  
>drop database uos;  
>create database uos; #重建数据库用于测试还原表  
mysql -u root -p uos < /scott.emp.dump #还原数据库中的表  
mysql -u root -p uos < /scott.dept+salgrade.dump #还原数据库中的多个表

**「9」用户和权限**

mariadb的用户登录包括用户名和登录位置两部分。早期版本%就可以代表所有的连接，后期版本localhost表示本地登录，%表示tcp/ip的远程登录的所有ip。%也可以写具体主机或网段，比如192.168.200.10、192.168.100.%或者192.168.200.0/255.255.255.0，其他格式不识别。如果登录主机符合多个连接条件，默认连接第一个身份，设置权限的时候需要注意

select user(); #查看当前登录用户  
  
use mysql  
select host,user,password from user; #查看数据库所有登录范围、用户和密码，4个root不是同一个用户，可以单独设置密码  
create user uos@'localhost' identified by '123456'; #建立可以本地登录的用户uos并设置密码123456  
create user uos@'%' identified by '123456'; #建立可以远程登录的用户uos并设置密码123456  
create user test@'localhost' identified by '123456'; #建立可以本地登录的用户test并设置密码123456  
create user test@'192.168.200.10' identified by '123456'; #建立可以从192.168.200.10主机登录的用户test并设置密码123456  
grant all on \*.\* to uos@'localhost'; #授权本地登录的用户uos对所有数据库的所有表有所有权限  
grant select on scott.\* to test@'localhost'; #授权本地登录的用户test可以读取scott数据库的所有表  
grant all on \*.\* to uostest@'%' identified by '123456'; #可以建立用户、授权、设置密码一起做  
flush privileges; #刷新授权表  
show privileges; #查看所有权限  
show grants for uos@'localhost'; #查看用户权限  
show grants for test@'localhost';  
revoke select on scott.\* from test@'192.168.200.10'; #收回权限

1. **mariadb主从复制**

在uos1上：

apt install mariadb-server  
systemctl enable mariadb.service  
配置主服务器 vim /etc/mysql/mariadb.conf.d/50-server.cnf  
[mysqld]  
server\_id = 1  
log-bin = /var/lib/mysql/mysql-bin  
replicate-do-db = uos #同步的数据库uos  
  
重启mariadb数据库 systemctl restart mariadb  
  
登录mysql主服务器  
grant replication slave on \*.\* to 'uos'@'192.168.200.11' identified by '123'; #允许从服务器slave访问  
flush privileges; #刷新权限  
show master status \G #查看主master服务器状态

在uos2上：

apt install mariadb-server  
systemctl enable mariadb.service  
配置从服务器 vim /etc/mysql/mariadb.conf.d/50-server.cnf  
  
[mysqld]  
server\_id = 2  
relay-log = /var/lib/mysql/relay-bin  
replicate-do-db = uos  
  
重启mariadb数据库 systemctl restart mariadb  
  
登录mysql从服务器  
change master to master\_host='192.168.200.10',master\_user='root',master\_password='123',master\_log\_file='mysql-bin.000001',master\_log\_pos=524; #指定master主服务器的ip、用户名密码、起始log文件和偏移位置  
start slave; #启动slave从服务器  
show slave status \G #查看从slave服务器状态  
  
Slave\_IO\_Running: Yes  
Slave\_SQL\_Running: Yes #主从复制搭建成功  
  
测试：  
在uos1主库中插入数据  
use uos  
insert into dept values (50,'SALES','TOM');  
在uos2从库中查看  
select \* from uos.dept;

1. **mycat读写分离**

在主从同步实验的基础上

在uos1上

安装jre软件

www.oracle.com网站下载jre-8u251-linux-x64.tar.gz压缩包

cd /root  
tar -xvzf jre-8u251-linux-x64.tar.gz  
export JAVA\_HOME=/root/jre1.8.0\_251 #加入环境变量中  
export PATH=$JAVA\_HOME/bin:$PATH  
export CLASSPATH=.$JAVA\_HOME/lib/dt.jar:$JAVA\_HOME/lib/tools.jar  
  
下载mycat软件  
wget https://github.com/MyCATApache/Mycat-download/raw/master/1.5-RELEASE/Mycat-server-1.5.1-RELEASE-20161130213509-linux.tar.gz  
tar -xvzf Mycat-server-1.5.1-RELEASE-20161130213509-linux.tar.gz  
  
配置server.xml文件 vim /root/mycat/conf/server.xml  
  
<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE mycat:server SYSTEM "server.dtd">  
<mycat:server xmlns:mycat="http://org.opencloudb/">  
 <system>  
 <property name="defaultSqlParser">druidparser</property>  
 </system>  
 <user name="root">  
 <property name="password">123</property>  
 <property name="schemas">uos</property>  
 </user>  
</mycat:server>  
  
配置schema.xml文件 vim /root/mycat/conf/schema.xml  
  
<?xml version="1.0"?>  
<!DOCTYPE mycat:schema SYSTEM "schema.dtd">  
<mycat:schema xmlns:mycat="http://org.opencloudb/" >  
  
 <schema name="uos" checkSQLschema="false" sqlMaxLimit="100" dataNode="db1"> </schema>  
 <dataNode name="db1" dataHost="localhost1" database="uos" />  
 <dataHost name="localhost1" maxCon="1000" minCon="10" balance="3 writeType="0" dbType="mysql" dbDriver="native" switchType="1" slaveThreshold="100">  
 <heartbeat>select user()</heartbeat>  
 <writeHost host="Master" url="192.168.200.10:3306" user="root" password="123" >  
 <readHost host="slave" url="192.168.200.11:3306" user="root" password="123" />  
 </writeHost>  
 </dataHost>  
</mycat:schema>  
  
<dataNode> 对应我们的在mysql中需要配置读写分离的数据库  
<writeHost> 和<readHost> 配置写服务器和读服务器  
url:服务器的ip地址和端口  
user:服务器上mysql的用户名  
password:服务器上mysql的用户密码  
心跳检测，方便我们查看读写分离情况  
  
balance="0", 不开启读写分离机制，所有读操作都发送到当前可用的writeHost 上。  
balance="1"，全部的 readHost 与 stand by writeHost 参与 select 语句的负载均衡，简单的说，当双主双从模式(M1 ->S1 ， M2->S2，并且 M1 与 M2 互为主备)，正常情况下， M2,S1,S2 都参与 select 语句的负载均衡。  
balance="2"，所有读操作都随机的在 writeHost、 readhost 上分发。  
balance="3"， 所有读请求随机的分发到 wiriterHost 对应的 readhost 执行,writerHost 不负担读压力，注意 balance=3 只在 1.4 及其以后版本有， 1.3 没有。  
  
writeType="0", 所有写操作发送到配置的第一个 writeHost，第一个挂了切到还生存的第二个writeHost，重新启动后已切换后的为准，切换记录在配置文件中:dnindex.properties .  
writeType="1"，所有写操作都随机的发送到配置的 writeHost。  
writeType="2"，没实现。-1 表示不自动切换1 默认值，自动切换2 基于MySQL 主从同步的状态决定是否切换  
  
配置log4j2.xml文件 vim /root/mycat/conf/log4j2.xml  
<level value="debug" /> #日志级别设置成debug   
  
配置wrapper.conf文件,vim /root/mycat/conf/wrapper.conf  
wrapper.java.command=/root/jre1.8.0\_251/bin/java  
wapper.java.additional.10=-Xmx512M  
wrapper.java.additional.11=-Xms64M  
  
uos1主库：  
use mysql  
update user set host = '%' where user = 'root';   
system restart mariadb

uos2从库：

修改MySQL的配置文件/etc/mysql/mariadb.conf.d/50-server.cnf，因为默认3306端口只允许本地访问的，注释掉这行

vi /etc/mysql/mariadb.conf.d/50-server.cnf  
//找到"bind-address = 127.0.0.1"这一行，前面添加#  
  
需进入数据库mysql,修改user表，如果没有密码设置新密码为123  
use mysql;  
UPDATE user SET password=PASSWORD('新密码') WHERE user='用户';  
flush privileges;  
  
systemctl restart mariadb

在uos1上

/root/mycat/bin/mycat start  
登录mycat  
mysql -u root -h127.0.0.1 -P8066 -p123 -Duos  
插入数据：  
insert into dept values (60,'SALES','JAM');  
查询数据：  
select \* from dept;  
  
查看mycat日志看结果  
tail -100 /root/mycat/logs/mycat.log