Day04 MHA集群 MySQL视图

一、MHA高可用集群中管理节点的配置（host56）

1.删除mysql56，准备好MHA集群环境

1）准备一主多从

·在作为 master的51 52 53 开启半同步复制

·安装依赖的perl软件包

·配置ssh服务 root用户无密码登录

·数据库服务器之间无密码登录

·管理主机56可以无密码登录所有数据库服务器

2.配置MHA集群

1）配置数据节点（51-55）

1-1 查看mysql51-55上的IO线程和SQL线程，检测主从同步是否配置完整，发现mysql52主库未指定为51

1-2 通过binlog日志将52后面的内容同步到51上;配置51为52的主库。

1-3 安装mha-node软件包

# yum -y install perl-\*

# rpm -ivh mha4mysql-node-0.56-0.el6.noarch.rpm

perl(DBD::mysql)依赖失败

# yum -y perl-DBD-mysql

# tar -xf mha4mysql-manager-0.56.tar.gz

# ls mha4mysql-manager-0.56

# cd mha4mysql-manager-0.56

# perl Makefile.PL

# yum -y install perl-ExtUtils-\*

# perl Makefile.PL

# yum -y install perl-CPAN-\*

# perl Makefile.PL

# make install

添加./bin目录至 $PATH

2）配置管理节点（56）

2-1 安装软件包

2-2 定义目录命令目录

2-3 修改配置文件

2-4 测试配置文件

# masterha\_check\_ssh --conf=/etc/mha\_manager/app1.cnf

# masterha\_check\_repl --conf=/etc/mha\_manager/app1.cnf

2-5启动服务

# masterha\_check\_status --conf=/etc/mha\_manager/app1.cnf

# masterha\_manager --conf=/etc/mha\_manager/app1.cnf \

--remove\_dead\_master\_conf --ignore\_lastfailover

# masterha\_check\_status --conf=/etc/mha\_manager/app1.cnf

1）修改故障切换脚本里定义vip地址

2）手动把脚本中定义vip地址部署在当前的主库51

3）测试高可用集群配置

在数据库服务器上添加访问数据时连接用户webuser

> create database db13;

> grant all on db13.\* to webuser@"%"

-> identified by "123456";

3-1 客户端连接VIP地址访问数据库

# mysql -h192.168.4.100 -uwebuser -p123456

3-2 测试高可用集群

·把主机51上的数据库服务停止, 客户端仍然连接VIP地址访问数据库

·把宕机的数据库服务器51再添加到当前的集群里

> change master to master\_host="192.168.4.52",

-> master\_user="repluser",

-> master\_password="123456",

-> master\_log\_file="master52.000001",

-> master\_log\_pos=154;

> start slave;

> show slave status;

MHA

Master Master Manager MMM 主主结构 有一个致命的缺点

二、MySQL视图

1.准备环境

1) 把/etc/passwd文件的内容存储到db9库下的user表里

2) 添加新字段id 存储记录的行号（在所有字段的前边）

3) 创建视图v1 结构及数据user表的字段、记录一样

4) 创建视图v2 只有user表shell是/bin/bash用户信息

5) 分别对视图表和基表执行insert update delete操作

6) 删除视图v1和v2

1) 把/etc/passwd文件的内容存储到db9库下的user表里

2) 添加新字段id 存储记录的行号（在所有字段的前边）

EXAMPLE:

> create database db9;

> use db9;

> create table user(

-> name char(20),

-> uid int(2),

-> gid int(2),

-> comment char(150),

-> homedir char(150),

-> shell char(30)

-> );

> system ls /var/lib/mysql-files

> system cp /etc/passwd /var/lib/mysql-files/

> system ls /var/lib/mysql-files

passwd

> load data infile "/var/lib/mysql-files/passwd" into table user

-> fields terminated by ":"

-> lines terminated by "\n";

> alter table user

-> add

-> id int (2)

-> primary key auto\_increment

-> first;

3) 创建视图v1 结构及数据user表的字段、记录一样

2.视图介绍和基本使用

1) 什么是视图(view)

- 虚拟表

- 内容与真实的表相似，包含一系列带有名称的列和行

- 视图并不在数据库中以存储的数据的形式存在

- 行和列的数据来自定义视图时查询所引用的基表，并且在具体引用视图时动态生成

- 更新视图的数据，就是更新基表的数据

- 更新基表数据，视图的数据也会跟着改变

2) 视图优点

·) 简单

- 使用视图的用户完全不需要关心视图中的数据是通过什么查询得到的

- 视图中的数据对用户来说已经是过滤好的符合条件的结果集

·) 安全

- 用户只能看到视图中的数据

·) 数据独立

- 一旦视图的结构确定了，可以屏蔽表结构变化对用户的影响

3) 使用视图的限制

·) 不能在视图上创建索引

·) 在视图的FROM子句中不能使用子查询

·) 以下情形中的视图是不可更新的

- 包含以下关键字的SQL语句：聚合函数(SUM、MIN、MAX、COUNT等)、Distinct、Group by、Having、Union或Union all

- 常量视图、Join、From一个不能更新的视图

- Where子句的子查询引用了From子句中的表

- 使用了临时表，视图不可更新

4) 视图的基本使用

·) 创建视图

·语法格式

- create view 视图名称 as SQL查询;

- create view 视图名称（字段名列表） as SQL查询;

> use db9;

> show tables;

> create view v1 as select \* from user;

> desc v1;

> create view v2 as

-> select name,uid from user limit 3;

>select \* from v2;

> create view v3(vname,vuid) from name,uid from user limit 3;

·) 查看视图

·查看当前库下所有表的状态信息

- show table status;

- show table status where comment="view"\G

·) 查看创建视图的具体命令

- show create view 视图名;

·tips : 在视图表中不定义字段名话，默认使用基表的字段名，若定义字段名的话，视图表中的字段必须和基表的字段个数相等

·) 使用视图

·查询记录

- Slelect 字段名列表 from 视图名 where 条件;

·插入记录

- Insert into 视图名(字段名列表) values(字段值列表);

·更新记录

- Update 视图名 set 字段名=值 where 条件;

·删除记录

- Delete from 视图名 where 条件;

注意：对视图操作即是对基本表操作，反之亦然

3.视图进阶

1) 创建视图完全格式

·) 完全命令格式

·命令格式

- CREATE

[OR REPLACE]

[ALGORITHM = { UNDEFINED | MERGE | TEMPTABLE }]

[DEFINER = { user | CURRENT\_USER }]

[SQL SECURITY { DEFINER | INVOKER }]

VIEW view\_name [ (column\_list) ]

AS selet\_statement

[WITH [ CASCADED | LOCAL ] CHECK OPTION]

·) 设置字段别名

·命令格式

- 视图中的字段名不可以重复 所以要定义别名

Create view 视图名

as

select 表别名.源字段名 as 字段别名 from 源表名 表别名 left join 源表名 表别名 on 条件;

- 关联查询的视图，默认不允许修改视图字段的值

- 创建视图时必须给视图中的字段定义别名的情况

快速建表

> create table user2 select name,uid,shell from user limit 5;

> create table info select name,uid,shell from user limit 10;

> select info.\*,user.name from info left join user2 where

-> info.uid = user2.uid;

> create view v2

-> as

-> select a.name as aname , b.name as bname , a.uid as auid , b.uid as buid from user2 a left join info b;

> create view v2

-> as

-> select a.name as aname , b.name as bname , a.uid as auid , b.uid as buid from user a left join info b on a.uid=b.uid;

2) 重要选项说明

·) OR REPLACE

·语法格式

- 创建时，若视图已存在，会替换已有的视图

- Create or replace view 视图名 as select 查询;

·) ALGORITHM

·定义处理视图的方式

- ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}

·MERGE（替换方式）

- 视图名直接使用视图的公式替换掉，把视图公式合并到了select中。

·TEMPTABLE（具体化方式）

- 先得到视图的执行结果，该结果形成一个中间结果暂时存在内存中，之后，外面的select语句就调用了这些中间结果。

·UNDEFINED（未定义）

- ALGORITHM选项的值时UNDEFINED表示使用的是MERGE替换方式。

·) WITH [ LOCAL | CASCADED ] CHECK OPTION

- 定义对视图表里的数据做操作时的限制方式。(当视图是根据另一个视图定义时，对视图更新/删除/插入)

- LOCAL仅检查当前视图的限制。（满足视图本身的限制即可）

- CASCADED同时要满足基表的限制（默认值）。

EXAMPLE:

> create table user2 select \* from user;

> create view v1 as select \* from user2 where uid <=30 with check option;

> create view v2 as select \* from v1 where uid >=20 with check option;

> select \* from v1 where name="mysql";

> select \* from v2 where name="mysql";

> update v2 set uid=19 where name="mysql";

> update v2 set uid=39 where name="mysql";

> update v2 set uid=29 where name="mysql";

> create view v3 as select \* from v1 where uid>=20 with local check option;

> show create view v3;

> select \* from v3;

> update v3 set uid=10 where name="mysql";

> update v3 set uid=35 where name="mysql";

> select \* from v3 where name="mysql";

> select \* from v1 where name="mysql";

//对视图表做操作，基表会被更改，因此，user2中mysql的uid更改为35，而不符合v1的限制，因此v1中不显示mysql字段，而v3基表为v1，因此v3中也不显示mysql字段

> create view v4 as select \* from user2 limit 3;

> update v4 set uid=666 where name="root";

//此时不会允许修改，因为修改后不满足"前三行"的约束条件