**第八章 备份与恢复**

**1.概述**

**1).按照备份的方法分：**

**host backup 热备**

在数据库运行中直接备份，对正在运行的数据库没有任何影响。

**cold backup 冷备**

在数据库停止的情况下进行备份，这种备份最为简单，一般只需要拷贝相关的数据库物理文件即可。

**warm backup 温备**

备份同样是在数据库运行时进行，但是会对当前数据库的操作有所影响，例如加一个**全局锁**以保证备份数据的一致性。

**2).按照备份后文件的内容分：**

**逻辑备份**

备份后的文件内容是可读的，通常是文本文件，内容一般是SQL语句，或者是表内的实际数据

如mysqldump和select \* into outfile的方法。

这类方法的好处是可以看到导出文件的内容。一般使用于数据库的升级，迁移工作，但是恢复所需要的时间往往比较长。

**裸文件备份**

拷贝数据库的物理文件，数据库既可以处于运行状态（如ibbackup、xtrabackup这类工具），也可以处于停止状态。这类备份的恢复时间往往较逻辑备份短很多。

**3).按照备份数据库的内容来分：**

**完全备份**

对数据库进行完整的备份。

**增量备份**

**日志备份**

对于mysqldump备份工具来说，可以通过添加--single-transaction选项来获得innodb存储引擎的一致性备份，务必加。

每个公司根据自己的备份策略**编写一个备份的应用程序**，这个程序可以方便地设置备份的方法以及**监控备份的结果**，并且可以通过第三方接口实时地通知DBA，这样才能真正地做到24\*7的备份监控。

任何时候都需要做好远程**异地备份**，也就是**容灾的防范**。只是同一机房的两台机器的备份是远远不够的。如果一个机房可能出现各种问题，这时远程异地备份就显得至关重要了。

**2.冷备**

对于innodb存储引擎的冷备非常简单，只需要备份MySQL数据库的frm文件、共享表空间文件、独立表空间文件(\*.ibd)、重做日志文件。另外，建议定期备份MySQL数据库的配置文件my.cnf，这样有利于恢复操作。

**优点：**

**备份简单**，只要拷贝相关文件即可。

备份文件易于在**不同操作系统、不同MySQL版本**上进行恢复。

**恢复相当简单**，只需要把文件恢复到指定位置即可。

**恢复速度快**，不需要执行任何SQL语句，也不需要重建索引。

**缺点：**

innodb存储引擎冷备的文件通常比逻辑**文件大**很多，因为表空间中存放着很多其他数据，如undo段、插入缓冲等信息。

冷备并**不总是可以轻易地跨平台**。操作系统、MySQL的版本，文件大小写敏感和浮点数格式会成为问题。

**3. 逻辑备份**

**1)mysqldump**

语法

**mysqldump [arguments] > file\_name**

**mysqldump --all-databases > dump.sql**

**mysqldump --databases db1 db2 db3 > dump.sql**

**mysqldump --single-transaction test > test\_backup.sql**

--single-transaction来保证备份的一致性，备份出的是文本文件，通过命令cat就可以查看文件的内容

文件内容就是表结构和数据，所有这些都是用SQL语句表示的。

参数：

**--single-transaction**：在备份开始前，先执行start transaction命令，以此来获得备份的一致性，当前该参数只对innodb存储引擎有效。

当启用该参数并进行备份时，确保没有其他任何的DDL语句执行，因为一致性读并不能隔离DDL语句。

**--lock-tables(-l)**：在备份中，以此锁住每个架构下的所有表。一般用于 myisam存储引擎，备份时只能对数据库进行读取操作，不过备份依然可以保证一致性。对于innodb存储引擎，不需要使用该参数，用--single-transaction即可，并且--lock-tables和--single-transaction是互相排斥的，不能同时使用。如果MySQL数据库中既有myisam，又有innodb，那么这时只能选择--lock-tables了。另外，--lock-tables选项是依次对每个架构中的表上锁，因此只能保证每个架构下表备份的一致性，而不能保证所有架构下表的一致性。

**--lock-all-tables(-x)**：在备份过程中，对所有架构中的所有表上锁。这可以避免之前提及的--lock-talbes参数不能同时锁住所有表的问题。

**--add-drop-database**：在create database前先运行drop database。这个参数要和--all-databases 或者--databases选项一起使用。默认情况下，导出的文本文件中并不会有create database，除非你指定了这个参数。

**--master-data[=value]**：通过该参数产生的备份转存文件主要用来简历一个slave replication。

当value=1时，转存文件中记录change master语句

mysqldump --single-transaction --add-drop-database --master-data=1 --database test > test\_backup.sql

cat test\_backup.sql

...

change master to **master\_log\_file='master01-bin.000005', master\_log\_pos=8908**;

...

当value=2时， change master语句被写成SQL注释。

mysqldump --single-transaction --add-drop-database --master-data=2 --database test > test\_backup.sql

cat test\_backup.sql

默认的值为空。

**--master-data会自动忽略--lock-tables选项。如果没有使用--single-transaction选项，则会自动使用--lock-all-tables选项。**

**--events (-E)**：备份事件调度器。

**--routine(-R)**：备份存储过程和函数。

**--triggers**：备份触发器。

**--hex-blob**：将binary,varbinary,blob,bit类型备份为十六进制的格式。mysqldump导出的文件一般是文本文件，但是，如果导出的数据中有上述这些类型，文本文件模式下可能有些字符不可见，若添加--hex-bob选项，结果会以十六进制的方式显现.

大多数DBA喜欢用select ... into outfile的方式来导出一张表，但是通过mysqldump一样可以完成工作，而且可以一次完成多张表的导出，并且保证导出数据的一致性。

**2). select ... into outfile**

语法：

select [col1,col2...]

into

outfile 'file\_name'

[ {fields | columns}

[terminated by "string"]

[[optionally]enclosed by 'char']

[escaped by 'char']

]

[ lines

[starting by 'string']

[terminated by 'string']

]

from table where ....

**file\_name表示导出的文件，但文件所在的路径的权限必须是mysql:mysql，否则MySQL会报告没有权限导出。**

**3) 逻辑备份的恢复**

mysqldump的恢复操作比较简单，因为备份的文件就是导出的SQL语句，一般只需要执行这个文件就可以了

mysql -uroot -p < test\_backup.sql

如果在导出时包含了创建和删除数据库的SQL语句，则必须确保删除架构时架构目录下没有其他与数据库无关的文件。

因为逻辑备份的文件是由sql语句组成的，所以也可以通过source命令来执行导出的逻辑备份文件

mysql> source /home/mysql/test\_backup.sql;

问题：

通过mysqldump 可以恢复数据库，但是常发生的一个问题是mysqldump可以导出存储过程、触发器、事件、数据，但是却**不能导出试图**。 因此，如果你的数据库中还使用了试图，那么在用mysqldump备份完数据库后还需要导出试图的定义，或者保存试图定义的frm文件，并在恢复时进行导入，这样才能保证mysqldump数据库的完全恢复。

**4) load data infile**

若是通过mysqldump --tab或者select into outfile导出的数据需要恢复时，这时需要通过load data infile命令来进行导入.

mysql>load data infile '/home/mysql/a.txt' into table a;

为了加快innodb存储引擎的导入，你可能希望导入过程忽略对外键的检查，因此可以使用如下方式：

mysql> set @@foreign\_key\_checks=0;

mysql>load data infile '/home/mysql/a.txt' into table a;

mysql>set @@foreign\_key\_checks=1;

另外可以针对指定的列进行导入，如将数据导入到a、b，而c列等于a+b：

mysql>create table b(a int,b int, c int,primary key(a))engine=innodb;

mysql>load data infile '/home/mysql/a.txt' into table b fields terminated by "," (a,b) set c = a+b;

load data infile 命令可以用来导入数据，但同时可以完成对Linux操作系统的监控。如果需要监控CPU的使用情况，可以通过加载/proc/stat来完成。首先需要建立一张监控CPU的表cpu\_stat

mysql> create database dba;

mysql> create table if not exists dba.cpu\_stat( id bigint auto\_increment primary key, value char(25) not null, user bigint, nic bigint, system bigint, idle bigint,iowait bigint, irq bigint, softirq bigint, steal bigint, guest bigint, other bigint, time dtetime );

cp /proc/stat /var/lib/mysql-files/

mysql> load data infile '/proc/stat'

-> ignore into table dba.cpu\_stat

-> fields terminated by ' '

-> (@value,@val1,@val2,@val3,@val4,@val5,@val6,@val7,@val8,@val9,@val10)

-> set

-> value=@value,

-> user=if(@value not like 'cpu%',null,if(@value != 'cpu',ifnull(@val1,0),ifnull(@val2,0))),

-> nice=if(@value not like 'cpu%',null,if(@value != 'cpu',ifnull(@val2,0),ifnull(@val3,0))),

-> system=if(@value not like 'cpu%',null,if(@value != 'cpu',ifnull(@val3,0),ifnull(@val4,0))),

-> idle=if(@value not like 'cpu%',null,if(@value != 'cpu',ifnull(@val4,0),ifnull(@val5,0))),

-> iowait=if(@value not like 'cpu%',null,if(@value != 'cpu',ifnull(@val5,0),ifnull(@val6,0))),

-> irq=if(@value not like 'cpu%',null,if(@value != 'cpu',ifnull(@val6,0),ifnull(@val7,0))),

-> softirq=if(@value not like 'cpu%',null,if(@value != 'cpu',ifnull(@val7,0),ifnull(@val8,0))),

-> steal=if(@value not like 'cpu%',null,if(@value != 'cpu',ifnull(@val8,0),ifnull(@val9,0))),

-> guest=if(@value not like 'cpu%',null,if(@value != 'cpu',ifnull(@val9,0),ifnull(@val10,0))),

-> other=if(@value like 'cpu%',user+nice+system+idle+iowait+irq+softirq+steal+guest,@val1),

-> time=now();

mysql> select \* from cpu\_stat\G;

接着可以设置一个定时器来让MySQL数据库自动地运行上述load data infile语句，这样就会有每个时间点的CPU信息被记录到表cpu\_stat。

mysql> select

-> 100\*((new.user-old.user)/(new.other-old.other)) user,

-> 100\*((new.system-old.system)/(new.other-old.other)) system,

-> 100\*((new.idle-old.idle)/(new.other-old.other)) idle,

-> 100\*((new.iowait-old.iowait)/(new.other-old.other)) iowait,

-> 100\*((new.irq-old.irq)/(new.other-old.other)) irq,

-> 100\*((new.softirq-old.softirq)/(new.other-old.other)) softer,

-> 100\*((new.steal-old.steal)/(new.other-old.other)) steal,

-> 100\*((new.guest-old.guest)/(new.other-old.other)) guest,

-> new.time

-> from dba.cpu\_stat old,

-> dba.cpu\_stat new

-> where new.id-15=old.id

-> and old.value='cpu'

-> and new.value=old.value\G;

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

user: 0.0915

system: 0.1198

idle: 99.7420

iowait: 0.0341

irq: 0.0000

softer: 0.0010

steal: 0.0117

guest: 0.0000

time: 2019-02-18 15:20:08

1 row in set (0.00 sec)

同样，我们还可以对/proc/diskstat文件执行以上操作，这样就又可以对磁盘进行监控操作了。

**5) mysqlimport**

是MySQL数据库提供的一个命令行程序，从本质上来说，是load data infile的命令接口，而且大多数的选项都和load data infle 语法相同。

格式：

shell> mysqlimport [options] db\_name textfile1 [textfile2 ...]

与load data infile 不同的是，mysqlimport命令可以导入多张表的，并且通过use-threads参数来并发导入不同的文件。这里的并发是指并发导入多个文件，并不是指mysqlimport可以并发导入一个文件，这时有区别的，并且并发地对同一张表进行导入，效果一般都不会比串行的方式好。通过mysqlimport并发导入两张表。

] mysqlimport --use-threads=2 test /home/mysql/t.txt /home/mysql/t.txt /home/mysql/s.txt

mysql>show full processlist\G;

可以看到mysqlimport实际上是同时执行了2条load data infile语句来完成并发导入操作。