--

mysql系统体系结构

SQL 结构化查询语言

--------------

|SQL Layer|

--------------

| | |

-----------------------------

|storage engine layer |

-----------------------------

逻辑模块组成

总的来说，mysql可以看成是两层架构

第一层我们通常叫做sql layer,在mysql数据库系统处理底层数据之前的所有工作都是在这一层完成的，包括权限判断,query解析，执行计划优化，query cache（缓存)的处理等；

第二层就是存储引擎层，通常叫做storage engine layer,也就是底层数据存取操作实现部分，由多种存储引擎共同组成

--mysql存储引擎简介

共10种

1.MyISAM 查询引擎

--mysql5.5版本前重点掌握的版本之前默认的存储引擎，也是目前使用非常广泛的存储引擎之一，是ISAM的升级版本。用于读多写少的环境

2.innoDB 存储引擎简介

除了MyISAM外，使用最为广泛的存储引擎，由第三方公司开发，和mysql遵守相同的开源许可协议。

支持事务安全

数据多版本读取

锁定机制的改进

它有表空间的概念，但与oracle表空间相比又有较大的不同，首先，innodb的表空间分为两种形式，一种是共享表空间，也就是所有数据都被存放在同一个表空间（一个或多个数据文件）中，通过innodb\_data\_file\_path来指定，增加数据文件须要停机重启。另外一种是独享表空间，也就是每个表的数据和索引都存放在一个单独的.ibd文件中。

3.Memory 存储引擎

它是一个将数据存储在内存中的存储引擎。

memory存储引擎不会将任何数据存放到磁盘上，在磁盘上仅仅存放了一个表结构相关信息的.frm文件。所以一旦mysql崩溃或主机崩溃之后,memory的表就只剩下一个结构了。它实现的是页级锁定XAMP

此引擎所有数据都存放在内存中，那么它消耗的内存量可想而知。

----------

其他存储引擎介绍

4.merge

可以简单地理解其功能就是结构相同的MyISAM表通过一些特殊的包装对外提供一个单一的访问入口，从而减小应用复杂度的目的。

要创建merge表，不仅基表的结构要完全一致，包括字段的顺序，基表的索引也必须完全一致

merge表本身并不存储数据，仅仅是多个基表提供一个统一的存储入口。所以在创建merge表的时候，mysql只会生成两个较小的文件，一个是.frm的结构定义文件，还有一个.MRG文件，用于存放参与MERGE表的名称（包括所属数据库schema)

5.NDB cluster存储引擎简介

主要用于mysql cluster分布式集群环境。简单地说，mysql cluster就是在无共享存储设备的情况下实现的一种内存数据库cluster环境，主要是通过NDB cluster (简称NDB）存储引擎来实现的。

6.BDB存储引擎

BDB存储引擎全称为berkeleyDB存储引擎，和innoDB一样，也是第三方的，同样支持事务安全。

在锁定机制方面，和memory存储引擎一样，实现页级锁定。

7.federated 存储引擎

它所实现的功能和oracle的DBLINK基本相似，主要是用来提供对远程mysql服务器上面的数据的访问接口，如果我们使用源码编译来安装mysql,那么必須手工指定启用federated存储引擎，因为mysql默认是不启用该存储引擎的。

8.archive 存储引擎

主要用于通过较小的存储空间来存放过期的很少访问的历史数据。archive表不支持索引，包含一个.frm的结构定义文件，一个.ARZ的数据压缩文件，还有一个.ARM的元数据

信息文件。由于其所存放的数据的特殊性，archive表不支持删除，修尽管操作，仅支持插入和查询操作。

锁定机制为行级锁定。

9.blackhole存储引擎

就是一个‘黑洞’，就像/dev/null设备一样，不管写入任何信息，都是有去无回。

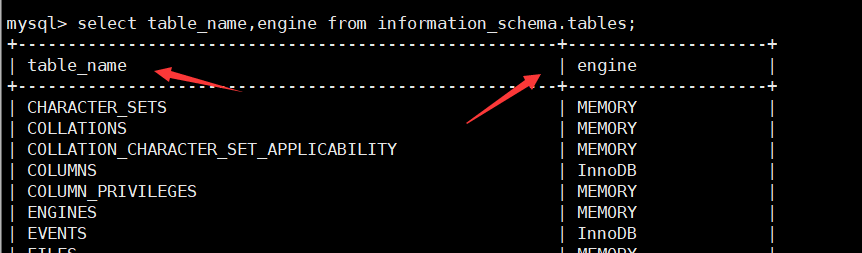
比如数据迁移过程，数据要经过一个中转的mysql服务器做一些相关的转换操作，然后再通过复制移植到新的服务器上面。如果中转服务器没有足够的空间，可以用blackhole引擎，它不记录任何数据（这样就不会占用空间了），但是会在binary log中记录下所有的query，把这些记录复制应用到新的服务器上。

10.CSV存储引擎

操作的就是一个标准的CSV文件，它不支持索引，主要用途就是有时候可能要通过数据库导出一份报表文件，而CSV文件是很多软件都支持的一种标准格式，所以可以先在数据库中建立一张CVS表，然后将生成的报表信息插入该表，就能得到一份CSV报表文件了

--查询数据库里所有的表及其对应的engine

mariadb[(none)] > select table\_name,engine from information\_schema.tables;



--查询表名为xxx的表的engine

mariadb[(none)] > select table\_name,engine from information\_schema.tables where table\_name="xxx";

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

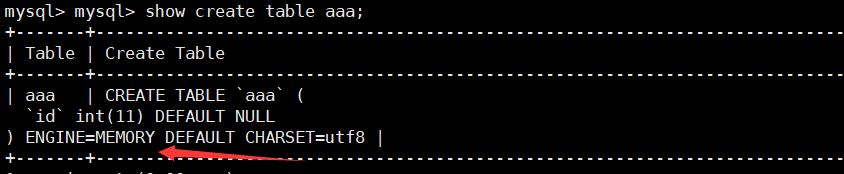
验证memory存储引擎的功能

mariadb[(none)] > create database aaa; --或者create schema aaa;

mariadb[(none)] > use aaa; --进入这个schema

mariadb[(none)] > create table aaa (id int) engine=memory; --如果不指定engine=memory则使用默认引擎

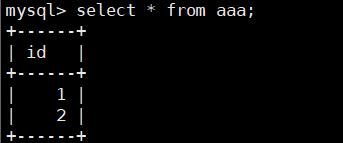
mariadb[(none)] > show create table aaa; --查看创建这个表的语句



mariadb[(none)] > insert into aaa values(1); --对aaa表插入数据整数1

mariadb[(none)] > insert into aaa values(2);

mariadb[(none)] > select \* from aaa;



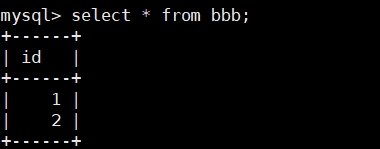
mariadb[(none)] > create table bbb (id int); --使用默认引擎

mariadb[(none)] > show create table bbb; --查看创建这个表的语句

mariadb[(none)] > insert into bbb values(1);

mariadb[(none)] > insert into bbb values(2);

mariadb[(none)] > select \* from bbb;



重启数据库

# systemctl restart mariadb

# mysql -p

mariadb[(none)] > select \* from aaa.aaa; --发现没有数据了

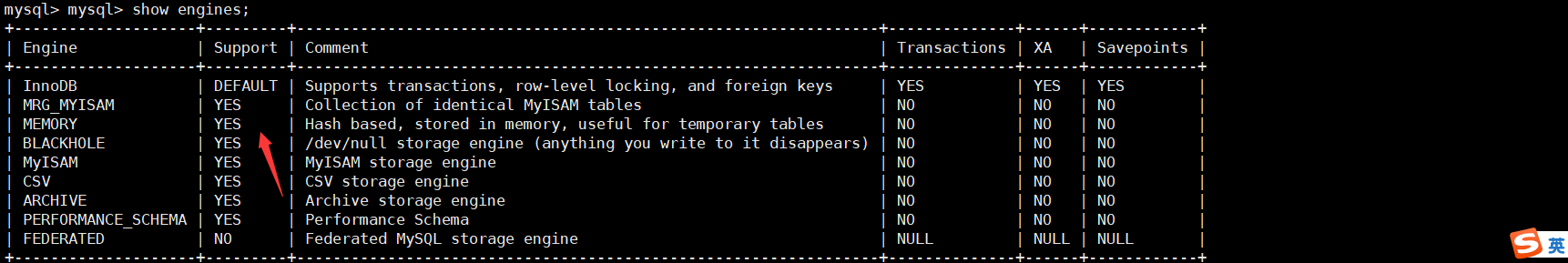
mariadb[(none)] > select \* from aaa.bbb; --发现数据还在

=======================================================

存储引擎相关的操作：

mariadb[(none)] > show engines;

里面的default表示默认引擎



YES表示支持的引擎，也就是可以在数据库里创建这种引擎的表

no表示禁用

--如何把引擎的no改为yes，把yes改为default

# vim /etc/my.cnf --修改配置文件在mysqld参数组下加上下面两句

federated --把federated引擎由no改为yes

skip-archive --把archive引擎由yes改为no

default-storage-engine=myisam --默认引擎改为myisam

--重启数据库验证

# systemctl restart mariadb

--登陆数据库

# mysql

# 再查看默认的数据库引擎是否发生改变

mariadb[(none)] > show engines;

--把aaa表改成innodb的存储引擎

mariadb[(none)] > alter table aaa engine=innodb;

--查看一个表的创建信息（就包含属于哪种引擎)

mariadb[(none)] > show create table aaa;

=================================================================================

mysql物理文件组成

日志文件

1. 错误日志 error log

记录了运行过程中所有较为严重的警告和错误信息，以及mysql server每次启动和关闭的详细信息。

在默认情况下，系统记录错误日志的功能是关闭的，错误信息被输出到标准错误输出（stderr),如果要开启系统记录错误日志的功能，须要在启动时开启--log-error选项，或者在配置文件种加入：log-error=filename

error log默认存放在数据目录下，以hostname.err命名。但是可以使用命令:--log-error[=file\_name]修改其存放目录和文件名

/usr/bin/mysqld\_safe --defaults-file=/etc/my.cnf & --使用这条命令启动mysql，第一句就会显示错误日志的路径

备:mysql启动不成功，看错误日志报什么错。

2,查询日志：query log

查询日志记录mysql中所有的query，可通过"general\_log \ general\_log\_file[=file\_name]来打开此日志。

由于记录了所有的query，包括所有的select,体积比较大，开启后对性能也有较大的影响，所以请大家慎用此功能。一般只在跟踪一些特殊的query性能问题时才会短暂打开此功能。

默认的查询日志文件名为hostname.log

# vim /etc/my.cnf --在[mysqld]里加上下面两句，重启mysql服务

general\_log

general\_log\_file=/var/log/mariadb/mariadb\_query.log

3,二进制日志： binary log& binary log index(类似oracle里的redo日志)

在通过"--log-bin[=file\_name]"打开记录的功能之后，mysql会将所有修改数据库的操作以二进制形式记录到日志文件中，还包括每一条query所执行的时间，所消耗的资源，以及相关的事务信息，所以binlog是事务安全的。

rpm版：

# vim /etc/my.cnf

[mysqld]

log-bin=mysql-bin

# systemctl restart mariadb 有时候会报错。需要打开server\_id=1这样的参数

# ls /data --产生二进制日志

mysql-bin.000001

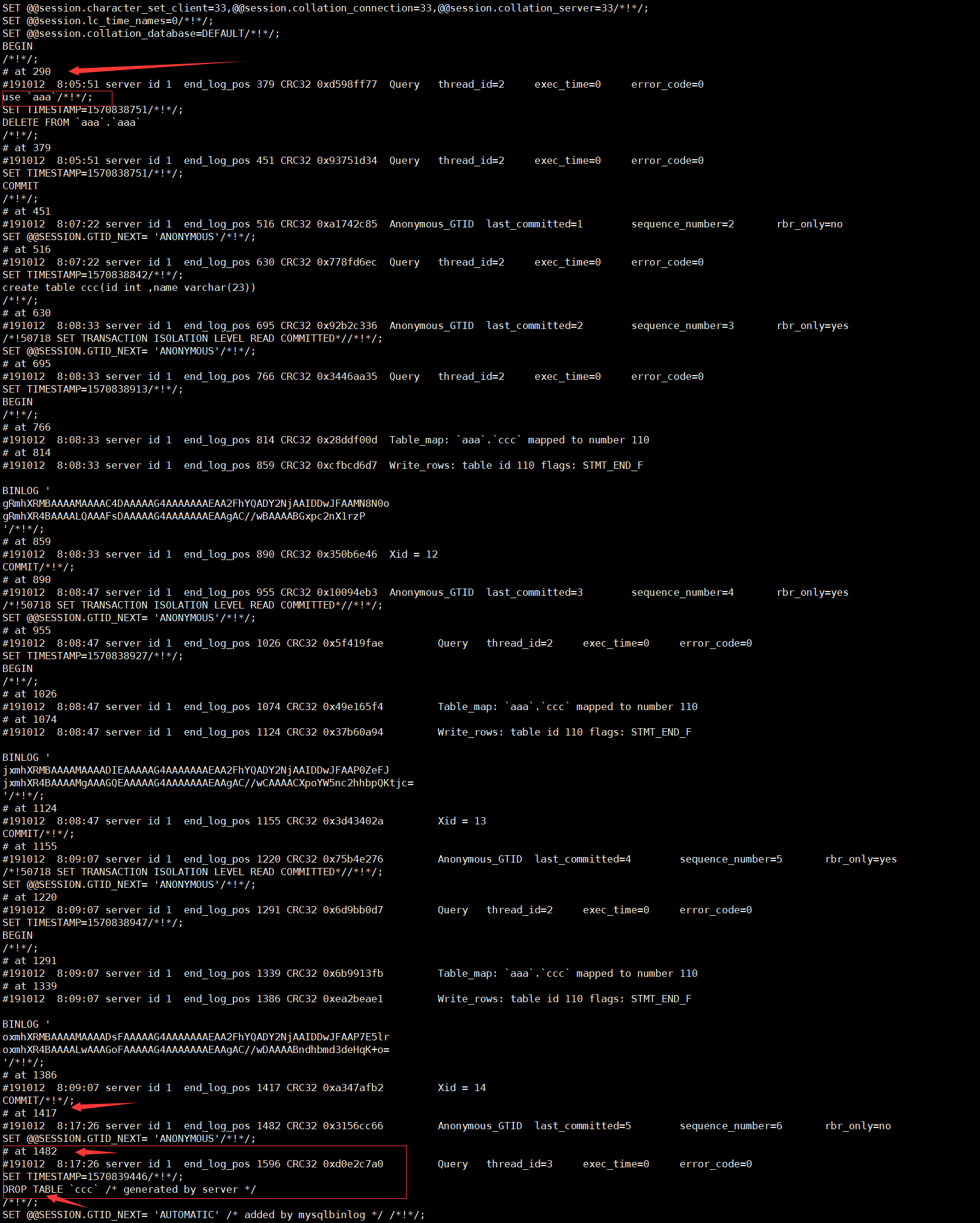
# mysqlbinlog /data/mysql-bin.000001 --不能用cat查看，只能使用mysqlbinlog命令查看此日志信息

330 845

实验:

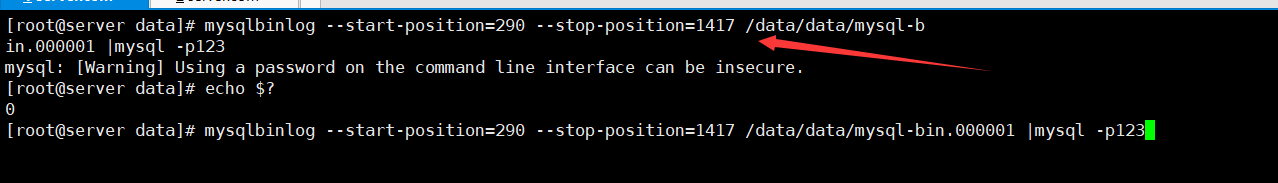
在aaa库里建了一个ccc表，插入了3条数据，然后drop table ccc模拟误操作删除了这张表

# mysqlbinlog /data/mysql-bin.000001 --查看二进制日志文件

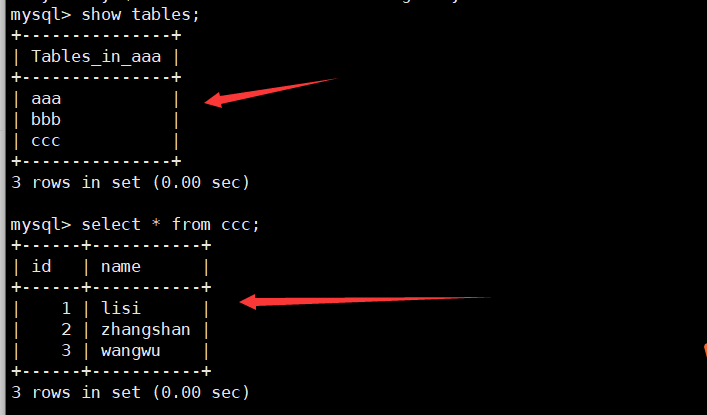


恢复方法

# mysqlbinlog --start-position=290 --stop-position=1417 /data/data/mysql-bin.000001 |mysql -p123



会发现数据已经恢复了，



4.慢查询日志：slow query log

此日志中记录的是执行时间较长的query,也就是常说的slow query,通过设--log-slow-queries[=file\_name]来打开此功能并设置记录位置和文件名，默认文件名为hostname-slow.log,默认目录也是数据目录。

慢查询日志采用的是简单的文本格式，可以通过各种文件编辑器查看其中的内容。其中记录了语句执行的时刻，执行所消耗的时间，执行用户，连接主机等相关信息。

mysql还提供了专门用来分析慢查询日志的工具程序mysqlslowdump,以帮助数据库管理人员解决可能存在的性能问题。

5.innoDB引擎的在线redo日志：innoDB redo log

innoDB是一个事务安全的存储引擎，其事务安全性主要是通过在线redo日志和记录在表空间的undo信息来保证的。redo日志中记录了innoDB所做的所有物理变更和事务信息，通过redo日志和undo信息，innoDB保证了在任何情况下的事务安全性。innoDB的redo日志同样默认存放在数据目录下，可以通过innodb\_log\_group\_home\_dir来更改设置日志的存放位置，通过innodb\_log\_files\_in\_group设置日志的数量。

innodb\_log\_group\_home\_dir=

innodb\_log\_files\_in\_group=4

－－－－－－－－－－－－－－－－－－－－－－－－

数据文件

不同的存储引擎有各自不同的数据文件，存放位置也有区别。多数存储引擎的数据文件都存放在和myisam数据文件位置相同的目录下，但是每个数据文件的扩展名却各不一样。如myisam用“.MYD"为扩展名，innoDB用".ibd",archive用".arc",CSV用'.CSV',等等。

1。.frm 文件

不论是什么存储引擎，每一个表都会有一个以表命名的.frm文件，与表相关的元数据（meta)信息都存放在此文件中，包括表结构的定义信息等。

所有的.frm文件都存放在所属数据库的文件夹下面。

2。.MYD文件

myisam存储引擎专用，存放myisam表的数据（data)。每一个myisam表都会有一个.MYD文件与之呼应，同样存放在所属数据库的文件夾下，和.frm文件在一起

create table a (id int);

insert into a values(1);

insert into a select \* from a;

......

插入几百W行记录，就会发现.MYD很大了

3。.MYI文件

也是myisam存储引擎专用，存放myisam表的索引相关信息。对于myisam存储引擎来说，可以被缓存(cache)的内容主要就是来源于.MYI文件中。每一个myisam表对应一个.MYI文件，其存放的位置和.frm及.MYD一样

4。.ibd文件和ibdata文件

存放innoDB的数据文件（包括索引）。innoDB存储引擎有两种表空间方式：独享表空间和共享表空间。

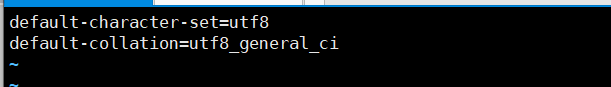
独享表空间存储方式使用.ibd文件来存放数据，且每个表一个.ibd文件，文件存放在和myisam数据相同的位置。

共享存储表空间使用.ibdata文件，所有表共同使用一个（或多个，自行配置）ibdata文件。

mysql5.6里每个innodb引擎的表会对应一个 表名.ibd 文件，用于存放数据的

5，每个schema里会有一个记录字符集的文件叫db.opt

# cat /data/aaa/db.opt --每一个schema目录下会有一个db.opt文件，记录字符集信息



==================================================================================

mysql 数据库的帮助

1，官档 dev.mysql.com

2，man 文档

3，命令行下的帮助文件 --oracle命令行没有，需要去查官方文档

mariadb [(none)] > help;

mariadb [(none)] > help create table;

mariadb[(none)] > ? contents

--分了很多类

Account Management

Administration

Data Definition

Data Manipulation

Data Types

Functions

Functions and Modifiers for Use with GROUP BY

Geographic Features

Language Structure

Storage Engines

Stored Routines

Table Maintenance

Transactions

Triggers

mariadb[(none)] > ? Account Management --一级一级的向内部查

mariadb[(none)] > ? CREATE USER

sql语句

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

sql 语句/语言 （结构化查询语言,structure query language) ，在不同数据库都是通用的，但会有少许不同

PL/SQL

最早是由ANSI定义的标准 sql-86

DQL(data query language) 数据查询语言 select --查询数据

DDL(data define language) 数据定义语言 create drop alter truncate rename --对表操作的命令

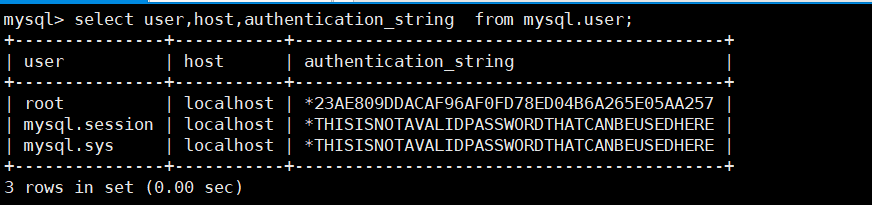
DML(data manipulate language) 数据操纵语言 insert update delete --对表内容操作的命令

DCL(data control language) 数据控制语言 grant revoke --授权

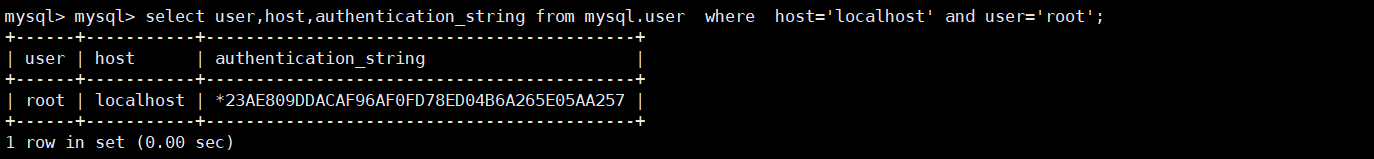
select 查询格式

select [ \* / 字段] from 表名 where 表字段 = 值 or/and 表字段 = 值

select host,user,password from mysql.user;



select host,user,password from mysql.user where host='127.0.0.1' and user='root';



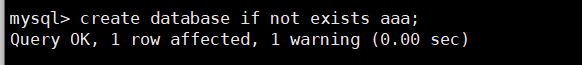
DDL

mariadb[(none)] > drop database/schema aaa; --删除前面测试时已经创建的aaa库

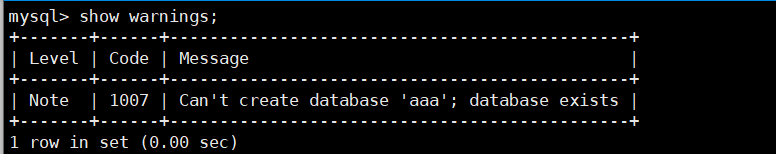
mariadb[(none)] > create database/schema aaa; --创建一个叫aaa的库，也就是数据目录下创建了一个aaa的目录，owner为mysql.mysql，并在其内有一个db.opt的默认字符集文件

mariadb[(none)] > show create database aaa; --可以用show查看创建库或者表的相关参数

mariadb[(none)] > create database if not exists aaa; --如果不存在则创建，如果存在，则有一个警告



mariadb[(none)] > show warnings; --使用此命令可以查看上一句命令的警告，注意只能查看上一句



# 修改表名

mariadb[(none)] > rename table ccc to ab2;

# 截断表数据，截断后表数据不可恢复 相当于把数据清空了

mariadb[(none)] > truncate table ab2;

Query OK, 0 rows affected (0.00 sec)

--库改名的话没有相关命令，可以直接在系统上mv来修改

========================================================

数据类型

数值类型

tinyint -128-127 0-255 (unsigned)

smallint 0-65535(unsigned)

mediumint

int 整数的11位数

bigint 大表字段值

浮点数型

float 单精度

double 双精度

定点数

decimal (M,D) M=整数位＋小数位 D＝小数位

字符类型 最高支持255个字符

char[M] 定长，M代表长度 10 char(10) abc

varchar[M] 变长，M代表长度限制 10 varchar(10) abc

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

char是一种固定长度的类型，varchar则是一种可变长度的类型

char(M)类型的数据列里，每个值都占用M个字节，如果某个长度小于M，MySQL就会在它的右边用空格字符补足．

（在检索操作中那些填补出来的空格字符将被去掉）在varchar(M)类型的数据列里，每个值只占用刚好够用的字节再加上一个用来记录其长度的字节（即总长度为L+1字节）

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

mysal> create table ab(v varchar(4),c char(4));

query ok ,0 rows affected(0.02 sec)

mariadb[(none)] >insert into abc values('ab ','ab ');

query ok ,1 row affected(0.00 sec);

mysql->select \* form abc

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

| v | c |

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

| ab | ab |

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

1 row in set (0.00 sec)

时间类型

date 年月日

year 年

time 时分秒

datetime 年月日 时分秒

timestamp 用于经常插入或者更新系统时间

============================================================

mariadb[(none)] > use aaa;

mariadb[(none)] > create table emp (ename varchar(20),sex char(1),hiredate date,sal decimal(10,2),deptno tinyint(1)); --创建表

mariadb[(none)] > create table emp

-> (ename varchar(20),

-> sex char(1),

-> hiredate date,

-> sal decimal(10,2),

-> deptno tinyint(1)); --可以分开写，比较容易阅读，但注意这只是一行

mariadb[(none)] > show create table emp; --查看创建表的参数，引擎和默认的字符集

mariadb[(none)] > alter table emp engine=memory; --修改表的引擎

mariadb[(none)] > alter table emp default charset=utf8; --修改表的字符集

mariadb[(none)] > rename table emp to emp1; --把emp表改名为emp1表

mariadb[(none)] > drop table emp;

mariadb[(none)] > create table emp (ename varchar(20), sex char(1), hiredate date, sal decimal(10,2), deptno tinyint(1)) engine=memory default charset=utf8; --也可以在创建表时直接指定非默认的引擎或字符集

mariadb[(none)] > create table dept (deptno tinyint(1),deptname varchar(30),location varchar(50)); --再创建一个部门表，注意deptno这一列和emp表的deptno列是对应的

============================================================

alter 命令修改表结构

add 新增字段，默认在最下面

first 新增字段，默认在最前面

after 指定新增到某个字段后面

drop 删除字段

modify 修改字段数据类型

change 修改字段名

mariadb[(none)] > alter table emp add age tinyint(1); --增加一个列，默认增加到最后

mariadb[(none)] > alter table emp drop age; --删除某一列

mariadb[(none)] > alter table emp add manager varchar(30) after hiredate; --在hiredate这列后面增加一列，使用after关键字

mariadb[(none)] > alter table emp add manager varchar(30) first; --把一列加到最前面，使用first关键字

mariadb[(none)] > alter table emp modify manager varchar(40); --使用modify修改一列的数据类型

mariadb[(none)] > alter table emp change manager boss varchar(30); --修改列名要使用change去修改

--modify和change的区别，都可以修改数据类型，但change要写原列名;只有change可以修改列名，modify不可以

练习：通过上面的命令，把emp表多加三列，成为下面的格式

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

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

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

| empno | int(11) | YES | | NULL | |

| ename | varchar(20) | YES | | NULL | |

| sex | char(1) | YES | | NULL | |

| birthday | date | YES | | NULL | |

| hiredate | date | YES | | NULL | |

| sal | decimal(10,2) | YES | | NULL | |

| deptno | tinyint(1) | YES | | NULL | |

| managerno | int(11) | YES | | NULL | |

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

mariadb[(none)] > alter table emp add empno int first;

mariadb[(none)] > alter table emp add birthday date after sex;

mariadb[(none)] > alter table emp add managerno int;

mariadb[(none)] > truncate table emp; --截断表，清空了表内的所有数据，但是表的结构还在

========================================================================

DML insert delete update

mariadb[(none)] > insert into aaa.emp values (1,'boss','m','1964-08-08','1995-01-01','20000','1','1');

--插入数据，注意在mysql里数字可以不用加引号，别的都加引号

mariadb[(none)] > insert into emp values

-> (1,'boss','m','1964-08-08','1995-01-01','20000','1','1'),

->(2,'zhangsan','m','1967-04-05','1995-04-11','15000','2','1');

Query OK, 2 rows affected, 0 warning (0.02 sec)

Records: 2 Duplicates: 0 Warnings: 0

--一次插入多条记录

还可以插入特定的列（非所有列），那么没有插入的就成了空值（空值不是0，它做任何运算结果还是空值)

mariadb[(none)] > insert into emp (ename,sex) values ('lisi','m');

mariadb[(none)] > create table emp1 (empno tinyint,ename varchar(20),sex char(1),birthday date,hiredate date,sal decimal(10,2),deptno tinyint(1),managerno tinyint);

下面针对上面的新表结构，写成一个脚本，直接插入9列数据

--也可以直接写成一行的脚本，使用mysql工具直接导入进去

# vim emp1.sql --创建一个文件，复制粘贴上我下面这一段

#!/bin/bash

insert into aaa.emp1 values (1,'boss','m','1964-08-08','1996-01-01','20000','1','1'),(2,'zhangsan','m','1967-04-05','1995-04-11','15000','2','1'),(3,'lisi','f','1973-01-28','1998-11-21','13000','3','1'),(4,'wangwu','f','1975-06-03','1999-12-12','12000','4','1'),(5,'maliu','m','1982-08-18','2001-07-03','8000','2','2'),(6,'tianqi','f','1983-02-15','2002-11-01','7000','2','2'),(7,'mark','m','1984-08-12','2003-10-02','6500','3','3'),(8,'john','m','1985-09-14','2005-04-03','6000','3','3'),(9,'mm','f','1990-06-08','2008-09-13','4000','4','4');

上面的脚本导入的方法

方法1:

# mysql -p < emp.sql

方法2:

mariadb[(none)] > source /path/emp.sql

mariadb[(none)] > select \* from aaa.emp;

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

| empno | ename | sex | birthday | hiredate | sal | deptno | managerno |

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

| 1 | boss | m | 1964-08-08 | 1995-01-01 | 20000.00 | 1 | 1 |

| 2 | zhangsan | m | 1967-04-05 | 1995-04-11 | 15000.00 | 2 | 1 |

| 3 | lisi | f | 1973-01-28 | 1998-11-21 | 13000.00 | 3 | 1 |

| 4 | wangwu | f | 1975-06-03 | 1999-12-12 | 12000.00 | 4 | 1 |

| 5 | maliu | m | 1982-08-18 | 2001-07-03 | 8000.00 | 2 | 2 |

| 6 | tianqi | f | 1983-02-15 | 2002-11-01 | 7000.00 | 2 | 2 |

| 7 | mark | m | 1984-08-12 | 2003-10-02 | 6500.00 | 3 | 3 |

| 8 | john | m | 1985-09-14 | 2005-04-03 | 6000.00 | 3 | 3 |

| 9 | mm | f | 1990-06-08 | 2008-09-13 | 4000.00 | 4 | 4 |

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

9 rows in set (0.00 sec)

把dept表也插入记录，方便下面的实验

mariadb[(none)] > insert into dept values

-> (1,'manager','beijing'),

-> (2,'it','shenzhen'),

-> (3,'sale','shanghai'),

-> (4,'services','guangzhou');

Query OK, 4 rows affected (0.00 sec)

Records: 4 Duplicates: 0 Warnings: 0

insert into dept values (1,'manager','beijing'), (2,'it','shenzhen'), (3,'sale','shanghai'), (4,'services','guangzhou');

mariadb[(none)] > select \* from dept;

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

| deptno | deptname | location |

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

| 1 | manager | beijing |

| 2 | it | shenzhen |

| 3 | sale | shanghai |

| 4 | services | guangzhou |

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

4 rows in set (0.00 sec)

===============================================================

update

把wangwu性别改成m

mariadb[(none)] > update emp set sex='m' where ename='wangwu';

wangwu的工资加500

mariadb[(none)] > update emp set sal=sal+500 where ename='wangwu';

2号部门的所有员工工资涨10%

mariadb[(none)] > update emp set sal=sal\*1.1 where deptno=2;

mark由3号部门换成2号部门，同时工资加1000

mariadb[(none)] > update emp set deptno=2,sal=sal+1000 where ename='mark'; --正确写法

工资就john和mark涨10%,其它人不涨

mariadb[(none)] > update emp set sal=sal\*1.1 where ename='john' or ename='mark'; --and or逻辑操作符只能在where语句之后，where后接一个条件

mariadb[(none)] > update emp set sal=sal\*1.1 where ename in ('john','mark');

工资都涨10%，john和mark犯错误，就他们不涨

mariadb[(none)] > update emp set sal=sal\*1.1 where ename != 'john' and ename!='mark';

mariadb[(none)] > update emp set sal=sal\*1.1 where ename not in ('john','mark');

delete --删除指定行的数据

delete from emp where xxxx;

delete from emp where empno=9;

delete from user where host='192.168.88.132'; 删除user表里的host字段，且ip=192.168.88.132

delete from user where user='mysql'; 删除user表里的mysql 用户

逻辑条件and or not在where之后起作用

--------------------------------------------------------------

DQL

select

mariadb[(none)] > select \* from emp; --\* 代表查看所有列

mariadb[(none)] > select ename,sal from emp; -- 只查看其中几列

mariadb[(none)] > select distinct deptno from emp; --distinct 去除重复行

查询所有男员工的姓名和工资

mariadb[(none)] > select ename,sal from emp where sex='m';

查询工资大于8000的员工的所有信息

mariadb[(none)] > select \* from emp where sal>8000;

查询工资在4000到8000之间的员工的所有信息（包含4000和8000的)

mariadb[(none)] > select \* from emp where sal>=4000 and sal<=8000;

mariadb[(none)] > select \* from emp where sal between 4000 and 8000;

查询入职时间在2001那年的员工的所有信息

mariadb[(none)] > select \* from emp where hiredate>="2001-01-01" and hiredate<="2001-12-31";

mariadb[(none)] > select \* from emp where hiredate like "2001%";

mariadb[(none)] > select \* from emp where year(hiredate)=2001;

mariadb[(none)] > select \* from emp where substr(hiredate,1,4)=2001;

mariadb[(none)] > select \* from emp where hiredate regexp '2001.\*';

－－－－－－－－－－－－－－－－－－－－－－－－－－

排序

order by 表字段 asc --升序

order by 表字段 desc --降序

limit 数值 取值

limit 第几行,取多少行

limit 2,3 默认是从第0行开始计数，这里的意思是从第二行开始，取三行

排序操作统计工资总额，最大工资，最小工资，平均工资

mariadb[(none)] > select \* from emp order by sal; --以工资排序，默认升序排序

mariadb[(none)] > select \* from emp order by sal asc; --加不加asc都是升序（ascend)

mariadb[(none)] > select \* from emp order by sal desc; --desc表示降序排序(descend)

mariadb[(none)] > select \* from emp order by sex,sal;

--先按性别排，再按工资排。结果是女的都在一起，以工资从小到大排。男的都在一起，以工资从小到大排。

mariadb[(none)] > select \* from emp order by sex desc,sal desc;

找出工资最低的三个人的姓名和工资

mariadb[(none)] > select \* from emp order by sal limit 3;

找出工资最高的三个人的姓名和工资

mariadb[(none)] > select \* from emp order by sal desc limit 3;

找出工资最低的女员工的姓名和工资

mariadb[(none)] > select ename,sal from emp where sex='f' order by sal limit 1;

--where和order by一起使用时，where在前，order by在后

找出工资从高到低第三到第五的人的姓名和工资

mariadb[(none)] > select ename,sal from emp order by sal desc limit 2,3;

------------------------------------------------

聚合和分组操作：

count(\*) 统计参数

distinct 去重数据

group by 分组

sum() 计算总和，把列的值相加

max() 显示该列最大的值

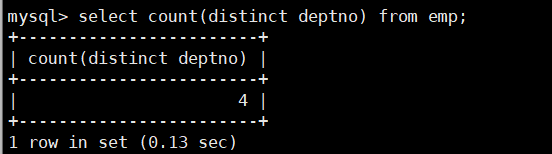
min() 显示该列最小的值

avg() 显示该平均值

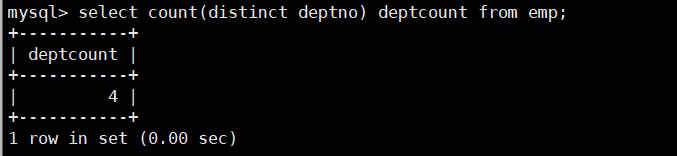
having 使用聚合函数和group by 分组后，再进行过滤使用的命令

mariadb[(none)] > select count(\*) from emp; --统计记录条数

mariadb[(none)] > select count(distinct deptno) from emp; --统计某个字段，并去重



mariadb[(none)] > select count(distinct deptno) deptcount from emp; --别名



统计每个部门的人数

mariadb[(none)] > select deptno,count(\*) from emp group by deptno;

mariadb[(none)] > select ename,deptno,count(\*) from emp group by deptno;

--这种在mysql里可以查，但无意义，在oracle里属于错误语法。因为前面select的列名除了count(),max(),min(),avg()，sum()等外，别的列名都必须在group by里

统计男、女员工各有多少人

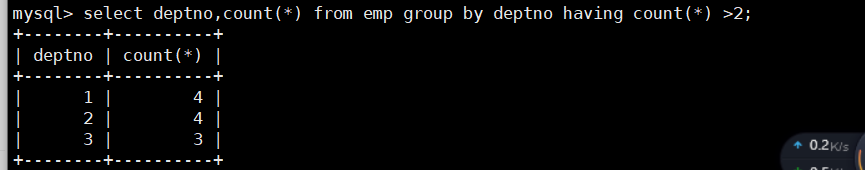
mariadb[(none)] > select sex,count(\*) from emp group by sex;

统计每个部门里男女员工各有多少个

mariadb[(none)] > select deptno,sex,count(\*) from emp group by deptno,sex;

查找部门人数大于2的部门号和人数

mariadb[(none)] > select deptno,count(\*) from emp group by deptno having count(\*) >2;



求每个部门的工资总额，最大工资，最小工资，平均工资

mariadb[(none)] > select deptno,sum(sal),max(sal),min(sal),avg(sal) from emp group by deptno;

------------------------------------------

表链接（多表查询)

mariadb[(none)] > create table emp (empno int(1),ename char(20),sex char(1),birthday date,hiredate date,sal decimal(10,2),deptno tinyint(1),managerno tinyint);

mariadb[(none)] > create table emp1 (deptno tinyint(1),dname char(20),loc char (5));

# vim 1.sql

#!/bin/bash

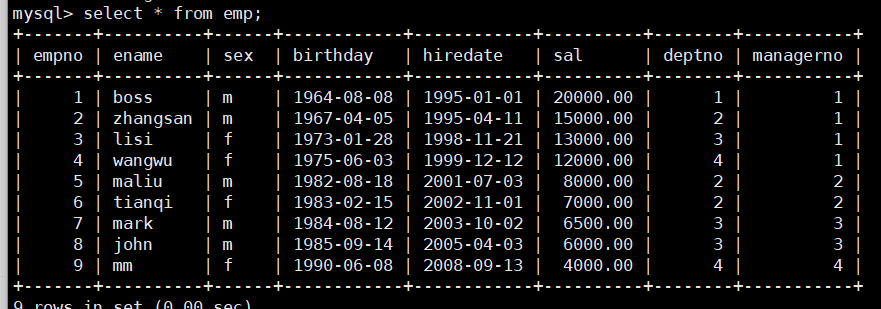
insert into aaa.emp values (1,'boss','m','1964-08-08','1995-01-01','20000','1','1'),(2,'zhangsan','m','1967-04-05','1995-04-11','15000','2','1'),(3,'lisi','f','1973-01-28','1998-11-21','13000','3','1'),(4,'wangwu','f','1975-06-03','1999-12-12','12000','4','1'),(5,'maliu','m','1982-08-18','2001-07-03','8000','2','2'),(6,'tianqi','f','1983-02-15','2002-11-01','7000','2','2'),(7,'mark','m','1984-08-12','2003-10-02','6500','3','3'),(8,'john','m','1985-09-14','2005-04-03','6000','3','3'),(9,'mm','f','1990-06-08','2008-09-13','4000','4','4');

# vim 2.sql

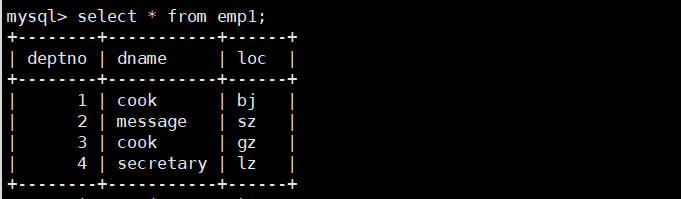
#!/bin/bash

insert into aaa.emp1 values ('1','cook','bj'),('2','message','sz'),('3','cook','gz'),('4','secretary','lz');

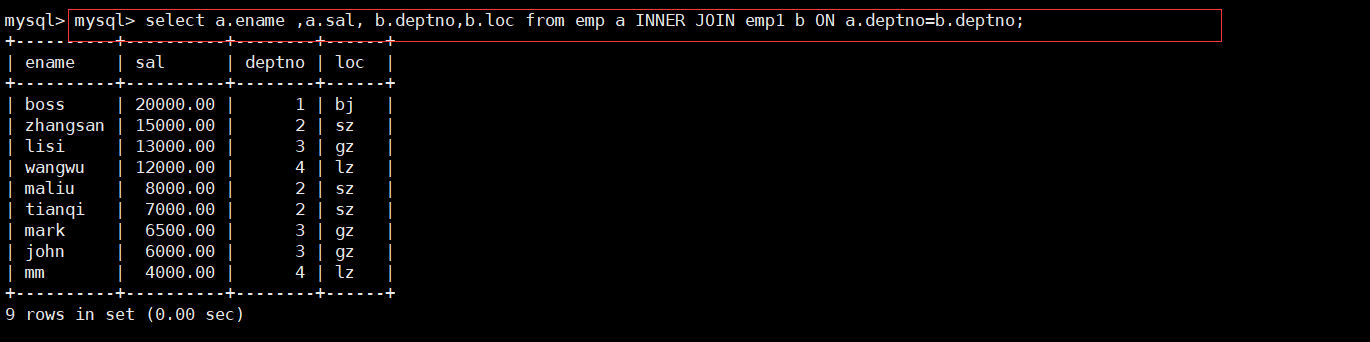
mariadb[(none)] > select \* from emp;



mariadb[(none)] > select \* from emp1;



mariadb[(none)] > select a.ename,a.sal,b.deptno,b.loc from emp a INNER JOIN emp1 b ON a.deptno = b.deptno;



解释：

从a读取ename sal字段

从b读取deptno loc字段

deptno字段在a和b中有重叠，所以通过该字段来进行关联

emp a 表示赋值a=emp

emp1 b 表示赋值b=emp1

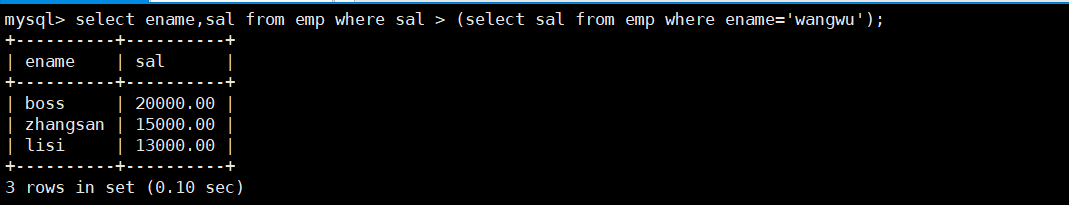
a.deptno = b.deptno 表示从两个表中选择哪个字段进行关联

－－－－－－－－－－－－－－－－－－－－－－－－

子查询：

查出比wangwu工资高的人的姓名和工资

mariadb[(none)] > select ename,sal from emp where sal > (select sal from emp where ename='wangwu');



----------------------------------------------------------------

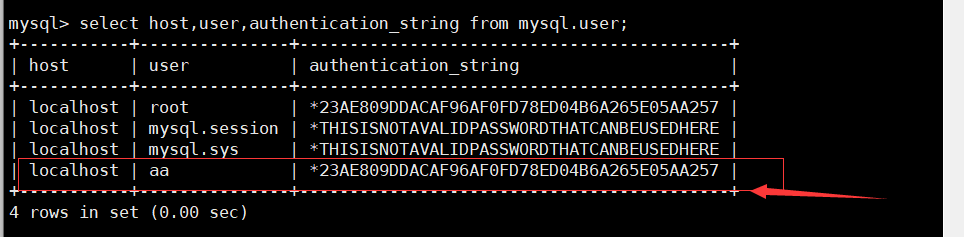
DCL

grant 授权

--授权使aa@localhost用户，对aaa库的emp表拥有select和insert权限

mariadb[(none)] > grant select,insert on aaa.emp to 'aa'@'localhost' identified by '123';

mariadb[(none)] > flush privileges; --刷新权限



--授权aaa所有权限并能操作所有库，所有表

grant all on \*.\* to 'aaa'@'localhost' identified by '123'

--授权ccc所有权限并能操作aaa库下的所有表

grnat all on aaa.\* 'ccc'@'localhost' identified by '123';

revoke 权限回收

--回收aa@localhost用户对aaa库的emp的select权限

mariadb[(none)] > revoke select on aaa.emp from 'aa'@'localhost';

mariadb[(none)] > flush privileges;

授权帐号能远程登陆

mariadb[(none)] > grant all on \*.\* to 'aaa'@'%' identified by '123'; --all代表所有权限，\*.\*代表所有库的所有表 %代表所有IP

mariadb[(none)] > flush privileges;

mariadb[(none)] > grant all on \*.\* to 'bb'@'192.168.224.11' identified by '123'; --all代表所有权限，\*.\*代表所有库的所有表

mariadb[(none)] > flush privileges;

代表192.168.224.11这台主机可以通过bb用户来登陆mysql

客户端测试

# mysql -u bb -h 192.168.224.10 -p123

-u 指定登陆时用户名

-h 指定登陆远程主机IP

-p 指定密码

-P 指定端口，如果是使用默认的3306就不用改端口

-S 指定sock文件路径

---------------------------------------------------------

函数：

mariadb[(none)] > select user(); --查看当前操作的登陆用户

mariadb[(none)] > select database(); --查看当前进入的数据库

mariadb[(none)] > select version(); --查看数据库版本

password() 函数

mariadb[(none)] > show variables like '%password%';

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

| Variable\_name | Value |

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

| old\_passwords | ON | --这里ON了，表示使用的是兼容老密码格式的16位

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

字符函数：

concat --连接函数

left --左截取

right --右截取

length --统计字段值长度

substr --截取函数

upper --将结果转成大写

lower --将结果转成小写

例：

mariadb[(none)] > select concat(ename,' salary is ', sal) from emp; --连接2个字符串

mariadb[(none)] > select left(ename,2) from emp; --取ename的左两个字符

mariadb[(none)] > select right(ename,2) from emp; --取ename的右两个字符

mariadb[(none)] > select length(ename) from emp; --长度函数

mariadb[(none)] > select substr(ename,2,2) from emp; --从第二个字符开始截取，截取两个字符

mariadb[(none)] > select upper(ename) from emp; --大写函数

mariadb[(none)] > select lower(ename) from emp; --小写函数

mariadb[(none)] > select upper(lower(ename)) from emp; --函数的嵌套

数学函数

mod --取余

round --四舍五入

例：

mariadb[(none)] > select mod(10,3);

mariadb[(none)] > select round(4.444444); --四舍

mariadb[(none)] > select round(4.544444); --五入

mariadb[(none)] > select round(4.544444,2); 四舍五入后，取两位小数点

日期函数

mariadb[(none)] > select current\_time();

mariadb[(none)] > select current\_date();

mariadb[(none)] > select current\_timestamp();

mariadb[(none)] > select now();

mariadb[(none)] > select year(now());

mariadb[(none)] > select month(now());

mariadb[(none)] > select day(now());

mariadb[(none)] > select hour(now());

mariadb[(none)] > select minute(now());

mariadb[(none)] > select second(now());