一　数据库存储引擎

１，mysql服务，体系结构

管理工具：命令

连接池：

Sql组件：

解析器：

优化器：

查询缓存：缓存曾经查找过的数据

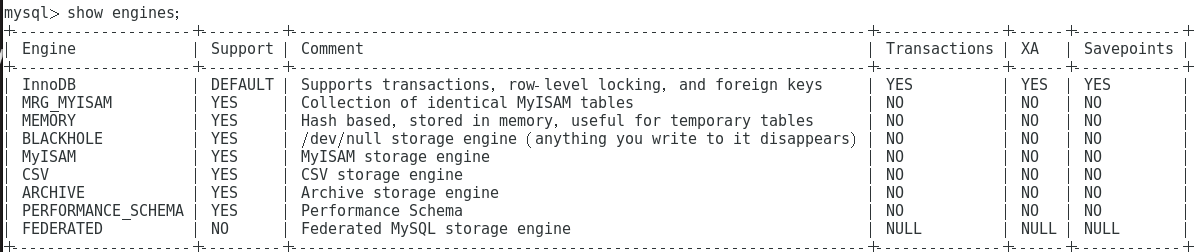
存储引擎：

文件系统：

２，mysql工作原理

－－存储引擎

mysql> show engines;　　　　#显示当前mysql的存储引擎



常见的存储引擎：innodb，myisam

[root@mysql-50 ~]# vim /etc/my.cnf

[mysqld]

default-storage-engine=MyISAM

－－mysql锁机制

1. 锁粒度

表级锁：一次直接对整张表进行加锁

行级锁：只锁定某一行  
页级锁：对整张页面（mysql管理数据的基本存储单位）进行加锁

1. 类型锁

读锁（共享锁）：支持并发锁

写锁（互斥锁，排它锁）：是独占锁，上锁期间其他线程不能读表或写表

1. 查看当前的锁状态

mysql> show status like "table\_lock%";

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

| Variable\_name | Value |

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

| Table\_locks\_immediate | 102 |

| Table\_locks\_waited | 0 |

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

2 rows in set (0.00 sec)

－－事务

１）定义

　　从连接数据库到断开数据库服务的过程称为事务

２）事务特性（ACID）

Atomic:原子性

　　事务的整个操作是一个整体，不可分割，要么全部成功，要么全部失败．

Consistency:一致性

　　事务操作的前后，表中的记录没有变化

Isolation:隔离性

　　事务操作是互相隔离不受影响的

Durability :持久性

　　数据一旦提交，不可改变，永久改变表数据

－－Myisam:

１）特点

存储数据库文件格式：

.frm：存储表结构

.MYI：存储字段排队(索引信息)

.MYD:存储数据

特点：

支持表级锁

不支持事务，事务回滚，外键

－－Innodb

１）特点

存储数据库文件格式：

.frm:存储表结构

.Ibd：存储数据和索引

事务日志文件：

ibdata1

ib\_logfile0

ib\_logfile1

特点：

支持行级锁

支持事务，事务回滚，外键

提示：

经常用于查询访问多的表，适合使用myisam存储引擎，节省系统资源

写访问量大的表，适合使用innodb存储引擎，访问量大

３，数据导入导出

－－修改默认使用目录

mysql> show variables like "%secure\_file%"; #查看默认导入数据目录

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

| Variable\_name | Value |

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

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

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

1 row in set (0.00 sec)

[root@mysql-50 ~]# vim /etc/my.cnf　　　　　#修改默认目录

[mysqld]

secure\_file\_priv="/mydata"

[root@mysql-50 ~]# mkdir /mydata　　　　　#新建目录

[root@mysql-50 ~]# chown mysql /mydata/　　#赋予mysql权限

[root@mysql-50 ~]# systemctl restart mysqld　　#重启数据库

mysql> show variables like "%secure\_file%";　　　#验证

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

| Variable\_name | Value |

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

| secure\_file\_priv | /mydata/ |

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

1 row in set (0.01 sec)

－－数据导入

用法：

Load data infile “目录名/文件名”

into table 表名

fields terminated by “分隔符”

Lines terminated by “\n”;

注意事项：

字段分隔符要与文件内的一致

指定导入文件的绝对路径

导入数据的表字段类型要与文件字段匹配

禁用selinux保护机制

例子：

mysql> create table user(

-> name char(30),

-> password char(1),

-> uid int,

-> gid int ,

-> comment char(150),

-> homedir char(150),

-> shell char(50)

-> );

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

mysql> system ls /var/lib/mysql-files

passwd

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

-> into table db3.user

-> fields terminated by ":"

-> lines terminated by "\n";

－－数据导出

基本用法：

Sql查询　　into outfile “目录名/文件名”

Fields terminated by “分隔符”　　　　#可以省略

Lines terminated by ”\n”;　　　　　　#可以省略

注意事项：

导出的内容有sql查询语句决定

导出的是表中的记录，不包括字段名

禁用selinux

例子：

mysql> select \* from db2.girlfriend into outfile "/mydata/girlfriend.txt";

４，sql语句

－－insert

格式：

Insert into 表名 values(字段值列表);

Insert into 表名 values(字段值列表1),(字段值列表2)....(字段值列表n);

Insert into 表名(字段名列表) values(字段值列表);

Insert into 表名(字段名列表) values(字段值列表1),(字段值列表2)....(字段值列表n);

注意事项：  
　　字段值要与字段类型相匹配

　　对于字符类型的字段，要用双引或单引括起来

　　依次给所有字段赋值时，字段名可以省略

　　只给一部分字段赋值时，必须明确写出对应的字段名称

例子：

mysql> insert into db2.girlfriend values("黄蓉",17,90,100864521,"女",curdate(),now(),"唱歌,跳舞");

－－select

格式：

Select 字段１,....,字段n from 表名;

Select 字段１,....,字段n from 表名 where 条件表达式;

注意事项：

使用\*可匹配所有字段

指定表名时，可采用库名.表名的形式

例子：

－－update

格式:

Update 表名　set 字段１=字段１值，set 字段２=字段２值，set 字段３=字段３值;

Update 表名　set 字段１=字段１值，set 字段２=字段２值，set 字段３=字段３值 where 条件表达式;

注意事项：

字段值要与字段类型相匹配

对于字符类型的字段，要用双或单引号括起来

若不使用where限定条件，会更新所有记录

限定条件时，只更新匹配条件的记录

－－delete

格式：

Delete from 表名 where 条件表达式; #仅删除符合条件的记录

Delete from 表名;　　　　　　　　　　　 #删除表中的所有记录

－－数值比较



－－字符比较



－－逻辑匹配

多个判断条件时使用



－－范围内匹配/去重显示

匹配范围内的任意一个值即可



－－高级匹配条件

1. 模糊查询

Where 字段名 like ‘通配符’

\_：匹配单个字符

%：匹配0~n个字符

mysql> select \* from girlfriend where name like '吴%';

| name | age | weight | phone | sex | birthday | firstsee | loves |

| 吴珊珊 | 17 | 90 | 1008611 | 女 | 2018-09-25 | 2018-11-17 08:54:21 | 唱歌,吃,睡 |

| 吴敏 | 19 | 90 | 17585217559 | 女 | 2018-09-25 | 2018-11-17 08:58:38 | 唱歌,吃,睡 |

mysql> select \* from girlfriend where name like '\_\_\_';

| name | age | weight | phone | sex | birthday | firstsee | loves |

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

| 吴珊珊 | 17 | 90 | 1008611 | 女 | 2018-09-25 | 2018-11-17 08:54:21 | 唱歌,吃,睡 |

| 小龙女 | 16 | 90 | 1008612 | 女 | 2018-08-25 | 2018-11-17 08:55:20 | 减肥,吃,睡 |

1. 正则表达式

Where 字段名　regexp ‘正则表达式’

正则元字符：^ $ . [] |

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

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

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

| name | age | weight | phone | sex | birthday | firstsee | loves |

| 貂蝉 | 18 | 95 | 1758521752 | 女 | 2018-09-20 | 2018-11-17 08:44:41 | 跳舞 |

提示：1个汉字占3个字符

1. 四则运算



－－操作查询结构

1. 聚集函数

Mysql内置数据统计函数

Avg(字段名)　　　　　　　#统计字段平均值

Sun(字段名)　　　　　　　＃统计字段之和

Min（字段名）

Max(字段名)

Count(字段名)

1. 查询结果排序

Order by 字段名 [asc | desc];

Asc：升序

Desc:降序

mysql> select \* from girlfriend where weight <=90 order by age;

mysql> select \* from girlfriend order by age desc;

1. 查询结果分组

Group by 字段名;

mysql> select weight from girlfriend group by weight;

1. 限制查询结果显示行数

基本用法：

Sql查询 limit n; //显示查询结果前n条记录

Sql查询 limit n,m; //显示指定范围内的查询记录

Sql查询 where 条件查询　limit 3;　//显示查询结果前3调记录

Sql查询　where 条件查询 limit 3,3; //从第４条开始，共显示３条

1. 查询结果过滤

Sql 查询　　having 条件表达式;

Sql 查询　where 题哦啊键　having 条件表达式;

Sql 查询　group by　字段名　having 条件表达式;

mysql> select \* from girlfriend where weight=90 having age=17;　　#效率高

mysql> select \* from girlfriend where weight=90 and age=17;　　　　#效率低