DBA1-03

**MYSQL存储引擎**

**工作原理**

**查询缓存：客户端之前查询所留下来的数据信息**

**MYSQL存储引擎**

**作为可插拔式的组件提供**

– MySQL 服务软件自带的功能程序,处理表的处理器

– 不同的存储引擎有不同的功能和数据存储方式

**存储引擎功能组件**

连接池 sql接口 分析器 优化器 查询缓存 存储引擎 文件系统 管理工具

**默认的存储引擎**

– MySQL 5.0/5.1 ---> MyISAM

– MySQL 5.5/5.6 ---> InnoDB

**修改表的存储引擎**

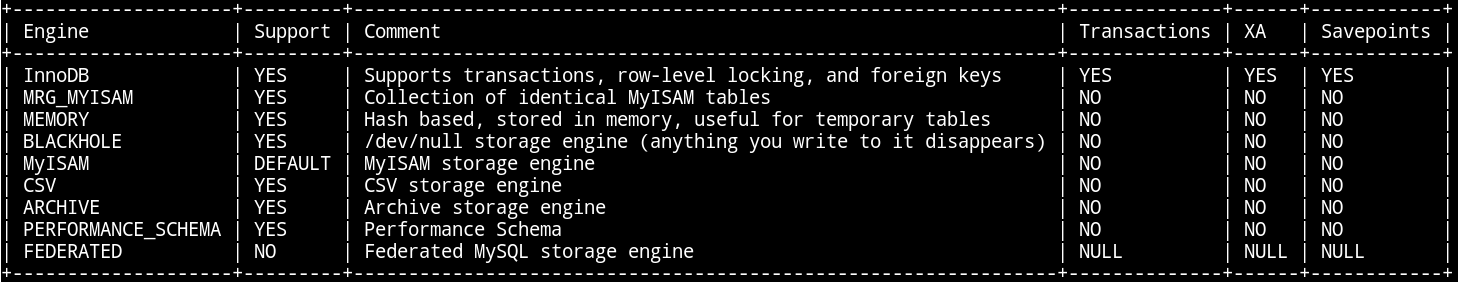
建表时手动指定

– 未指定时,使用默认存储引擎

– SHOW CREATE TABLE xxx\G ;可确认

**列出可用的存储引擎类型**

mysql> show engines;或mysql> show engines\G;



**设置默认存储引擎:修改 /etc/my.cnf 配置文件**

[root@client ~]# vim /etc/my.cnf

[mysqld]

default-storage-engine=myisam

**存储引擎特点**

**Myisam 存储引擎**

**• 主要特点**

**– 支持表级锁**

**– 不支持事务、事务回滚、外键**

**• 相关的表文件**

[root@client ~]# ls /var/lib/mysql/库名/

.frm结尾的存放的是表结构

.MYD结尾的存放表里的数据

.MYI结尾的存放索引信息

**InnoDB 存储引擎**

**• 主要特点**

– 支持行级锁定

– 支持事务、事务回滚、支持外键

**• 相关的表文件**

– xxx.frm 、 xxx.ibd

– ibdata1

– ib\_logfile0 事务日志文件

– ib\_logfile1

**总结：存储引擎的选用**

查询操作多的表适合使用myisam存储引擎

— 可以节省系统资源

西操作多的表适合使用innodb存储引擎

— 可以加大并发访问量

**MySQL 锁机制**

• 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:对整个页面( MySQL 管理数据的基本存储单

位)进行加锁。

• 锁类型

– 读锁(共享锁):支持并发读。

– 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。

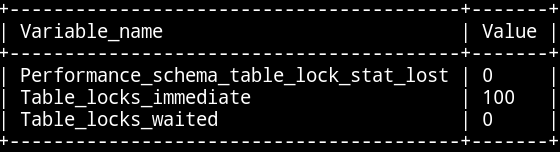
**锁的作用：**

解决对表的并发访问的冲突问题

**查看当前的锁状态**

— 检查Table\_lock开头的变量，%作通配符

mysql> show status like "%Table\_lock%"; //查看当前



mysql> show status; //查看所有的锁状态

**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

– 事务操作是相互隔离不受影响的。

• Durability :持久性

– 数据一旦提交,不可改变,永久改变表数据

**查看提交状态：（on——打开，off——关闭）**

mysql> show variables like "autocommit";

**关闭自动提交：**

mysql> set autocommit=off;

**数据回滚：**

mysql> rollback;

**提交数据：**

mysql> commit;

测试：

mysql> set autocommit=off; //关闭提交功能

mysql> create table db3.t4(id int)engine=innodb;

//创建表设置存储引擎Innodb

mysql> insert into db3.t4 values(666); //插入表数据

mysql> commit; //提交保证插入成功

mysql> select \* from db3.t4; //此时查看时有表数据的

mysql> delete from db3.t4; //删除表数据

//此终端查看确定表数据已经删除，新开终端连接数据库查看此表数据，可以看到表数据并未删除

mysql> rollback; //回到源终端进行数据回滚

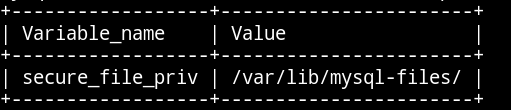
mysql> select \* from db3.t4; //再次查看表数据

//可以发现之前插入的表数据被成功回滚恢复

**数据导入导出**

**默认使用目录：**/var/lib/mysql-files/

**查看方法：**mysql> show variables like "secure\_file\_priv";



修改导入数据文件的目录

[root@client ~]# vim /etc/my.cnf

[mysqld]

secure\_file\_priv=/mydir //在配置文件中加入

[root@client ~]# mkdir /mydir //创建目录

[root@client ~]# chown mysql /mydir //修改归属

[root@client ~]# ls -ld /mydir //查看属性

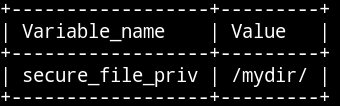
drwxr-xr-x. 2 mysql root 6 10月 17 14:06 /mydir

[root@client ~]# systemctl restart mysqld //重起服务

[root@client ~]# getenforce //确保selinux状态为关闭

Permissive

mysql> show variables like "secure\_file\_priv"; //查看使用目录



**数据导入：**

• 基本用法

– 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

要求：将本地用户的记录导入 db3库的 user 表里

1. **建库建表**

mysql> create database db3;

mysql> create table user(

-> name char(5),

-> password char(1),

-> uid int(1),

-> gid int(1),

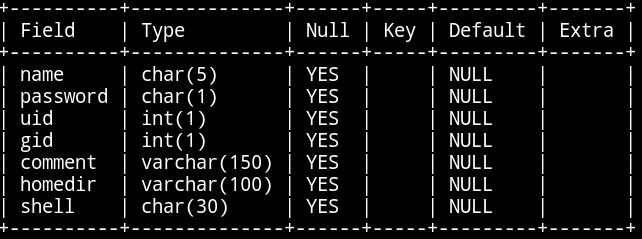
-> comment varchar(150),

-> homedir varchar(100),

-> shell char(30)

-> );

mysql> desc db3.user;



**2. 将要导入的文件拷贝到默认目录下：**

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

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

1. **将拷贝的文件进行导入到表中**

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

-> into table db3.user

-> fields terminated by":"

-> lines terminated by"\n";

1. **查看导入的数据**

mysql> select \* from db3.user;

**5. 为了方面查询，向表中加入主键自动增长字段id**

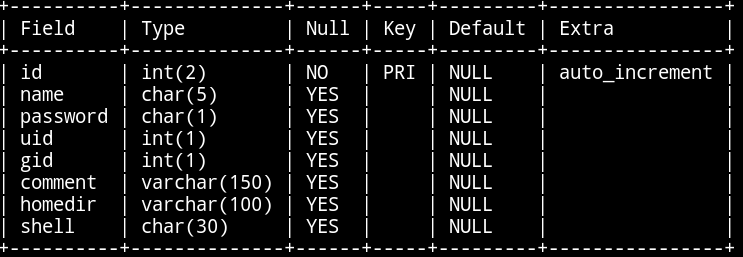
mysql> alter table db3.user

-> add

-> id int(2) primary key auto\_increment first;

mysql> select \* from db3.user;

mysql> desc db3.user;



**数据导出：**

• 基本用法

– SELECT 查询 .. ..

INTO OUTFILE “ 目录名 / 文件名”

FIELDS TERMINATED BY “ 分隔符”

LINES TERMINATED BY “\n”;

• 注意事项

– 导出的内容由 SQL 查询语句决定

– 禁用 SElinux

方式1：不改变分隔符，一切默认

mysql> select \* from teadb.t3 into outfile "/mydir/a.txt";

方式2：改变分隔符，指定换行符

mysql> select \* from teadb.t3 into outfile "/mydir/b.txt" fields terminated by "##" lines terminated by "!!!";

查看：

[root@client mysql]# cd /mydir

[root@client mydir]# ls

a.txt b.txt

[root@client mydir]# cat b.txt

**管理表记录**

1. **增加表记录**

格式 1 :给所有字段赋值

– INSERT INTO 表名 VALUES ( 字段 1 值, .. .. ,字段 N 值 ) ,( 字段 1 值, .. .. ,字段 N 值 ) ,( 字段 1 值, .. .. ,字段 N 值 ) ,...;

mysql> insert into user values

(43,"bob","x",3003,3003,"test user","/home/bob","/shell/bash"),

(44,"lisa","x",3005,3005,"test user","/home/bob","/shell/bash"),

(45,"lucy","x",3008,3008,"test user","/home/bob","/shell/bash");

格式 2 ,给指定字段赋值

– INSERT INTO 表名 ( 字段 1,.. .., 字段 N) VALUES

( 字段 1 值,字段 2 值,字段 N 值 ) ,

( 字段 1 值,字段 2 值,字段 N 值 ) ,...;

mysql> insert into user (name,uid,shell) values

("tom",2008,"/bin/bash"),

("jim",2005,"/bin/bash"),

("jack",2018,"/bin/bash");

//没有指定字段默认输出NULL，为空值

**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**

**– 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**

1. **查询表记录**

• 格式 1

– SELECT 字段 1, .. .., 字段 N FROM 表名 ;

**显示表的所有记录**

mysql> select \* from user;

**查看每行指定列的值**

mysql> select name,shell,homedir from user;

• 格式 2

– SELECT 字段 1, .. .., 字段 N FROM 表名

WHERE 条件表达式 ;

**加条件查询**

mysql> select name,shell,homedir from user where shell="/bin/bash";

**给字段定义别名称**

mysql> select name username ,uid user\_id from user;

• 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

1. **更新表记录**

语法格式

• 格式 1 ,更新表内的所有记录

– UPDATE 表名 SET

字段 1= 字段 1 值 ,

字段 2= 字段 2 值 ,

字段 N= 字段 N 值 ;

mysql> update user set password="a",comment="student";

• 格式 2 ,只更新符合条件的部分记录

– UPDATE 表名

SET

字段 1= 字段 1 值 ,

字段 2= 字段 2 值 ,

字段 N= 字段 N 值 ;

WHERE 条件表达式 ;

mysql> update user set password="x" where name="root";

• 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

1. **删除表记录**

语法格式

• 格式 1 ,仅删除符合条件的记录

– DELETE FROM 表名 WHERE 条件表达式 ;

mysql> delete from user where name="bob";

• 格式 2, 删除所有的表记录

– DELETE FROM 表名 ;

mysql> delete from db3.t4;

**匹配条件**

1. **基本匹配条件**

**数值比较**

• 字段类型必较数据数值类型

|  |  |
| --- | --- |
| 类 型 | 类 型 |
| = | 等于 |
| >、>= | 大于、大于或等于 |
| <、<= | 小于、小于或等于 |
| != | 不等于 |

mysql> select name,uid,gid from user where uid=gid;

mysql> select name,uid,gid from user where uid<=10;

mysql> select \* from user where id=3;

**字符比较 / 匹配空 / 非空**

• 字段类型比较

|  |  |
| --- | --- |
| 类型 | 用途 |
| = | 用途 |
| ！= | 不相等 |
| IS NULL | 匹配空 |
| IS NOT NULL | 非空 |

mysql> select \* from user where id=3;

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

mysql> select name,shell from user where shell !="/bin/bash";

**逻辑比较**

• 多个判断条件时使用

既有与又有或，默认先做与，再做或

|  |  |
| --- | --- |
| 类型 | 用途 |
| OR | 逻辑或 |
| AND | 逻辑与 |
| ！ | 逻辑非 |
| （） | 提高优先级 |

匹配名为root并且uid=3的行

mysql> select name,uid from user where name="root" and uid=3;

匹配名为root或者uid=3的行

mysql> select name,uid from user where name="root" or uid=3;

匹配名为root 或者 名为bin并且uid=3的行

mysql> select name,uid from user where name="root" or name="bin" and uid=3;

匹配名为root或bin，并且uid=3的行

mysql> select name,uid from user where (name="root" or name="bin") and uid=3;

mysql> select name,uid from user where (name="root" or name="bin") and uid=1;

**范围内匹配 / 去重显示**

• 匹配范围内的任意一个值即可

|  |  |
| --- | --- |
| 类型 | 用途 |
| IN（值列表） | 在...里... |
| Not in（直接表） | 不在...里... |
| Between数字1 and 数字2 | 在...之间... |
| DISTINCT 字段名 | 去重显示 |

mysql> select name from user where

->name in("mysql","daemon","apache","adm");

mysql> select name from user where uid in (10,20,30,40);

Empty set (0.00 sec)

mysql> select name,uid from user where uid not in (10,20,30,40);

mysql> select id,name,shell from user where id between 10 and 20;

mysql> select distinct shell from user; //去重查询

mysql> select distinct shell from user where uid<=1000;

**2. 高级匹配条件**

**模糊匹配**

• 基本用法

– WHERE 字段名 LIKE ' 通配字串 '

– 通配符 \_ 匹配单个字符

– % 匹配 0~N 个字符

• 示例

mysql> select \* from user where name like 'J%' or name like '%y';

mysql> select name from user where name like '\_\_\_';

mysql> select name from user where name like 'r\_\_t';

mysql> select name from user where name like 'r%t';

mysql> select id,name from user where name like '%';

mysql> select id,name from user where name="" or name is null;

mysql> select name from user where name like '\_\_%\_\_';

.......

**正则匹配**

• 基本用法

– WHERE 字段名 REGEXP ' 正则表达式‘

– ^ $ . [ ] \*

• 示例

显示以名字中包含数字的

mysql> select name from user where name regexp '[0-9]';

显示uid为三位数的

mysql> select name,uid from user where uid regexp '^...$';

显示以字母xxx开头

mysql> select name,uid from user where name regexp '^a|^r';

mysql> select name,uid from user where name regexp '^(a|r|c)';

**四则运算**

• 运算操作

– 字段必须是数值类型

|  |  |
| --- | --- |
| 类型 | 用途 |
| + | 加法 |
| - | 减法 |
| \* | 乘法 |
| / | 除法 |
| % | 取余数（求模） |

显示uid小与10的行

mysql> select \* from user where id <=10;

给满足uid小与10的行的uid加1

mysql> update user set uid=uid+1 where id<=10;

在name后面加入age字段

mysql> alter table user add age tinyint(2) unsigned default 19 after name;

通过计算显示root的出生日期

mysql> select name,age,2018-age start from user where name="root";

**聚集函数**

• MySQL 内置数据统计函数

– avg( 字段名 ) : 求平均值

– sum( 字段名 ) : 求和

– min( 字段名 ) : 统计最小值

– max( 字段名 ) : 统计最大值

– count( 字段名 ) : 统计个数

mysql> select avg(age) from user;

mysql> select avg(uid) from user where shell !="/bin/bash";

mysql> select sum(age) from user;

mysql> select min(uid) from user;

mysql> select max(uid) from user;

mysql> select count(name) from user where shell!="/bin/bash";

**操作查询结果**

1. **排序（默认升序排）**

• 基本用法

– SQL 查询 ORDER BY 字段名[ asc | desc ]

mysql> select name,uid from user where uid regexp '...$' order by uid;

mysql> select name,uid from user where uid regexp '^...$' order by

uid asc; //正序

mysql> select name,uid from user where uid regexp '^...$' order by uid desc; //反序

1. **分组**

• 基本用法

– SQL 查询 group by 字段名

mysql> select gid from user group by gid;

mysql> select gid from user where id<=20 group by gid;

1. **过滤**

• 基本用法

– SQL 查询 HAVING 条件表达式;

– SQL 查询 where 条件 HAVING 条件表达式;

– SQL 查询 group by 字段名 HAVING 条件表达式;

在查寻结果中过滤id=45的行

mysql> select id,name from user where name like '%' having id=45;

在查询结果中过滤名字是空的行

mysql> select id,name from user where name like '%' having name is null;

在查询结果中过滤名字为“”的行

mysql> select id,name from user where name like '%' having name="";

//分两次查询

mysql> select id,name from user where name like '%' and name="";

//在整张表中进行查询

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 N ; 显示查询结果前 N 条记录

– SQL 查询 where 条件查询 LIMIT N , M ; 显示指定范围内的查询记录

示例：

显示查询结果的前10行

mysql> select id,name,shell from user limit 10;

显示查询结果的9+6行

mysql> select id,name,shell from user limit 9,6;

mysql> select name,shell from user where shell!="/bin/bash" limit 3;

mysql> select name,shell from user where shell!="/bin/bash" limit 3,5;