6.19 存储引擎 数据导入 导出 匹配条件

。

一．存储引擎

默认的存储引擎是innodb

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

查看存储引擎的类型

mysql> show engines; 或 show engines\G

查看使用的存储引擎

Show create table xxx\G

1. 设置默认的存储引擎（修改匹配文件/etc/my.cnf）

default-storage-engine=xxxx

存储引擎的特点

生产环境中常用的是myisam和innodb

Myisam特点

一个表使用3个文件存储

t1.frm t1.myd t1.myl

表结构 数据 索引

不支持外健 和事务 和事务回滚

支持表级锁 （给一张表加锁）

Innodb特点

一个表中使用2个文件存储

t2.frm 表结构

t2.ibd 数据和索引

支持行级锁定（只给被访问的行加锁）

支持事务、事务回滚、支持外键

事务：一次sql操作从开始到结束的过程（连接数据库、操作数据、断开连接）

事务回滚：

执行过程中任意一步操作失败，恢复原有操作

事务日志文件：（只有innodb有事务日志文件，myisam没有事务日志）

Ibdata 1 未提交的sql

ib\_logfile0 ib\_logfile1 已经提交的sql

执行查询操作多的表，适合用myisam存储引擎，节省资源

执行写操作的表，适合用innodb存储引擎，加大访问并发量

Mysql锁机制

锁：解决并发访问的冲突问题

锁粒度：行锁 表锁 页级锁

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。

查看当前的锁状态

– 检查 Table\_lock 开头的变量, % 作通配符事务特性 (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

列：

mysql> show variables like "autocommit"; // 查看提交状态

mysql> set autocommit=off; //改成手动上传

mysql> create table t4(name int(2))engine=innodb;

mysql> insert into t4 values(44);

mysql> rollback ; // 数据回滚，在未提交前，可以恢复数据，提交后。不能恢复

mysql> commit; // 提交数据

二．数据导入、导出

查看在数据库中默认的目录

mysql> show variables like "secure\_file\_priv";

可以修改默认目录，修改配置文件/etc/my.conf

[mysqld]

secure\_file\_priv="/myload"

在数据库中拷贝目录要加system

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

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

在数据导入之前，要先在数据库中把表创建出来，格式要一样(创建的表要和文件的格式一样)

mysql> create table user(

-> name char(10),

-> password char(10),

-> uid smallint(2),

-> gid smallint(2),

-> aa char(100),

-> bb char(150),

-> shell char(50),

-> index(name)

-> );

导入数据：

格式：

LOAD DATA INFILE “ 目录名 / 文件名”

INTO TABLE 表名

FIELDS TERMINATED BY “ 分隔符”

LINES TERMINATED BY “\n”;

mysql> load data infile "/var/lib/mysql-files/passwd" into table user fields terminated by ":" lines terminated by "\n";

/var/lib/mysql-files/默认路经 将/var/lib/mysql-files/passwd/下的内容导入user表中，分隔符为”:”

出现报错：有可能是你的长度不够或者类型不可以，把类型smallint换成int

Select \* from user where id=30; //查看id为30的行

导出：表记录存储到文件里

格式：

SELECT 查询 .. ..

INTO OUTFILE “ 目录名 / 文件名” //文件名自定义

FIELDS TERMINATED BY “ 分隔符”

LINES TERMINATED BY “\n”;

mysql> select \* from user

-> into outfile

-> "/var/lib/mysql-files/user.txt";

//把user中的所有导出到/user.txt中

mysql> select id,name,password from user where id <=3 into outfile "/var/lib/mysql-files/user4.txt";

//把id小于3的导出到user4.txt中

mysql> select id,name,password from user where id <=3 into outfile "/var/lib/mysql-files/user5.txt"

-> fields terminated by ";" lines terminated by "####";

//把id小于3的导出到user5.txt下，分隔符为“；” 换行为“#####”

导出user表中uid小于100的用户记录

mysql> select \* from userdb.user where uid<100 into outfile "/

myload/user2.txt";

管理表记录(增、删、改、查)

增：

Insert into 库.表 values(字段值列表) //给所有值赋值

Insert into 表名（字段1....字段N）values（字段1值、2值） //指定字段赋值

在往表里增加信息时，不指定id，如果之前删除过，会默认从最大值开始加

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

查：

Select 字段1 ....字段N from 表名； //查找所有

Select 字段名列表 from 库.表 where 条件； //查找指定条件的

mysql> select id,name,shell,bb from user where id=3;

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

改：

Update 库.表 set 字段名=值，字段名=值 where 条件

Update user set password=”A”; //把user表中的password全部改成A

Update user set password=”x” where id=1; //id为1的password改为1

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

删：

Delete 库.表 where 条件；

mysql> delete from user where id>=40;

//删除40后面的行

DELETE FROM 表名 ; //删除所有

三． 匹配条件（select、update、delete）

1. 数值比较

=、>、>=、< 、<=、！=

Ls null 匹配空

Ls not null 非空

列：

mysql> update user set password="a",bb="/student" where uid<10; //uid小于10的password、bb的修改，后面的值要用双引号括起来

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

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

mysql> select name from user where uid is null;

mysql> update user set uid=4008 where uid is null; //查找用户的uid是4008的

mysql> select name,uid from user where name="yaya";

给字段设置为空，前题条件下，必须允许赋空值

mysql> update user set gid=not null where id=3;

2.逻辑比较（多个比较条件）

逻辑与 and 多个条件必须同时成立

逻辑或 or 多个条件某一个条件成立即可

逻辑非 ！ Not 取反

() 提高优先级

例：

mysql> select name,uid from user where name="root" and uid=0 and shell="/bin/bash"

mysql> select name,uid from user where name="root" or uid=7 or shell="/sbin/nologin";

mysql> select name,uid from user where name="root" and (uid=0 or uid=3);

3.范围内匹配 / 去重显示

匹配范围内的任意一个值即可

In ( 值列表 ) 在...里...

Not in ( 值列表 ) 不在...里...

Between 数字 1 and 数字

2 在...之间...

DISTINCT 字段名 去重显示 不显示字段的重复值

mysql> select name from user where name in ("aa","bb","cc","dd"); //匹配括号里的任意一个

mysql> select name,shell from user where shell not in ("/bin/bash","/sbin/nologin"); //匹配不等于括号里的解释器

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

mysql> select distinct shell from user; //不显示重复的值

mysql> select distinct shell from user where uid<10;

4.高级查询条件

1. 模糊匹配

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

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

//匹配r开头，t结尾的，中间两个字符

mysql> select name,uid from user where name like '\_\_\_' and uid>10;

//匹配uid大于10且是三个字符的

mysql> select name,uid from user where name like 'a%t';

//匹配以a开头，t结尾，中间匹配任意多个字符

5.正则匹配

WHERE 字段名 REGEXP ' 正则表达式‘ ^ $ . [ ] \*

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

//查找user表中uid为四个值的

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

//查找user表中name中包含数字的

mysql> select name from user where name regexp 't$';

//查找user表中以t结尾的

6.四则运算

字段必须是数值类型

+ 加法

- 减法

\* 乘法

/ 除法

% 取余数(求模)

mysql> alter table user add age tinyint(2) unsigned default 18 after name; //添加字段age，默认值是18

mysql> select name,age, 2018-age as aa from user where name="root"; //aa为自定义名字

mysmysql> select name,age from user;

mysql> select name,age from user;

mysql> select name,uid,gid,(uid+gid)/2 aaa from user; //求uid和gid的平均值

mysql> select name,uid,gid,(uid+gid)/2 aaa from user where uid>=10 and gid <=30;

7.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 max(uid) from user;

mysql> select count(uid) from user;

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

//查找shell是/bin/bash的人，统计个数

操作查询结果

1. 查询结果排序

通常是数值类型字段

SQL 查询

ORDER BY 字段名

[ asc | desc ]

列：

mysql> select name,id from user order by id;

//从小到大，不指定是默认是从小到大（asc）

mysql> select name,id from user order by id desc;

//加上desc，倒叙，从大到小

1. 查询结果分组

通常是字符类型字段

SQL 查询

group by 字段名

mysql> select shell from user where uid>=10 and uid<=1000 group by shell;

1. 查询结果过滤

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

式;

HAVING 条件表达

mysql> select name from user having name="aa"

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

mysql> select shell from user where uid>=10 and uid<=1000 group by shell having shell="sync";

4.限制查询结果显示行数

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

mysql> select \* from user limit 0,1;

mysql> select \* from user limit 9,5;

mysql> select \* from user limit 3,3;

mysql> select \* from user where uid <=10 order by uid desc limit 3;