mysql

主要参数

/etc/my.cnf MySQL主配置文件

/var/lib/mysql 数据库目录

3306 默认端口号

mysqld MySQL主进程名称

TCP 传输协议

msyql:mysql 进程所有者与所属组

/var/log/mysqld.log 错误日志文件

常用的SQL命令分类:

DDL数据定义语言(create,alter,drop)

DML数据操作语言(insert,update,delete)

DCL数据控制语言(grant,revoke)

DTL 数据事务语言(commit,rollback,savepoint)

首次进入mysql 需要查询自动生成的root密码

grep password /var/log/mysqld.log

修改root密码 若不修改则无法对数据库操作

方法一:

set global validate\_password\_policy=0; #只验证长度

set global validate\_password\_length=6; #修改密码长度,默认值是8个字符

alter user user() identified by "123456"; #修改登陆密码

方法二:

永久设置:

# vim /etc/my.cnf

#修改配置添加如下内容

[mysqld]

validate\_password\_policy=0

validate\_password\_length=6

###############################################################################

连接 MySQL 的方法:命令行、Web 网页、安装图形软件、编写脚本(PHP、java...)

连接MySQL数据库的命令语法格式:

mysql [-h服务器IP或域名 -u用户名 -p密码 数据库名称]

mysql -uroot -pchen -e 'show databases;' #非交互式对数据库进行操作

mysql> create database tts character set utf8mb4; #创建数据库并且定义整个库字符为万国字

mysql> drop database tts;

数据库命名规则

(数字、字母、下划线,不能纯数字;区分大小写;不能使用关键词或特殊符号)

\*\*特别重要不仅仅是数据库 也包括表中字段

创建数据表基本语法格式如下:

create table 数据库名称.数据表名称(

字段名1 数据类型(宽度) 约束条件,

字段名2 数据类型(宽度) 约束条件,

... ...

);

插入数据的语法格式:insert

into 数据库名称.数据表名称 values (值列表);

更新数据语法格式:update 数据库名称.数据表名称 set 字段=值 [where条件]

MySQL 数据类型

1. 字符类型

char(字符数) 固定长度,最大长度255字符,不够指定的字符数时自动在右边填补空格,超

出指定字符数则无法写入。

varchar(字符数) 可变长度,根据实际数据大小分配存储空间,超出指定字符数则无法写入。

text/blob 字符数大于65535时使用。

#超过了指定的 4 个字符,写入失败。

char(4)如果实际写入了 2 个字符,但是占用的空间是 4 个字符.

varchar(30)如果实际写入了 2 个字符,实际仅占用 2 个字符的空间.

2.数值类型 unsigned

类型 范围(有符号) 范围(无符号) 用途

tinyint -128~127 0~255 微小整数

smallint -32768~32767 0~65535 小整数

mediumint 中整数

\*int 大整数

bigint 极大整数

\*float 单精度浮点数(M,D)

#(M,D),其中 M 为总宽度,D 为小数位数,M 应大于 D

reate table school.num(

id tinyint,

age int(3),

score float(4,2));

3. 日期时间类型

datetime日期时间类型,范围1000-01-01 00:00:00:000000~9999-12-31 23:59:59.999999

timestamp日期时间类型,范围1970-01-01 00:00:00:000000~2038-01-19 03:14:07.999999

date日期类型,范围0001-01-01~9999-12-31

默认使用4位数字表示,当只用2位数字负值时:

01~69自动识别为2001~2069

70~99自动识别为1970~1999

year年份类型,范围1901-2155

time时间类型,范围HH:MM:SS

4. 枚举类型(选择类型)

enum(值1,值2,值3...) #单选项

set(值 1,值 2,值 3...)

#多选项

################################################################

约束条件

1.常用约束条件:

条件约束 功能描述

null 允许为空,默认设置

not null 不允许为空

key 索引类型

default 设置默认值,缺省为NULL

修改表结构

add 添加字段

modify 修改字段类型

change 修改自定名称 \*\* 同时可以对字段类型也修改

drop 删除字段

rename 修改表名称

add 添加字段

alter table 表名 add 字段名 类型(宽度) 约束条件; 默认在最后添加

alter table info add age int(3) not null default 18;

通过after字段名可以将新添加的字段放到某个字段后面,或者first直接放到第一列

alter table info add age int(3) not null default 18 after name;

alter table info add age int(3) not null default 18 first;

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

modify 修改字段类型

alter table 表名 modify 字段名 类型(宽度) 约束条件;

alter table info modify age varchar(4) null default 19;

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

change 修改自定名称 \*\* 同时可以对字段类型也修改

alter table 表名 change 原字段名 新字段名 类型(宽度) 约束条件;

alter table info change age nianling int(4) nut null default 20;

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

drop 删除字段

alter table 表名 drop 字段名 ;

alter table info drop age ;

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

rename修改字段名称

alter table 表名 rename 新的数据表名称

##############################################################################

MySQL键值

/var/lib/mysql/mysql 中文件 .frm 结构 .MYD 数据 .MYI 索引

什么是索引:就是对数据表中的若干字段进行排序的方法,类似于对一本书做目录,有了目

录就可以快速定位数据的具体位置。

索引的优点:

➢ 通过创建唯一性索引,可以保证数据库表中每一行数据的唯一性

➢

可以加快数据的检索速度

索引的缺点:

➢ 当对表中的数据进行增加、删除和修改的时候,索引也要动态的维护,会降低数据库

的写速度

➢

索引需要占额外的物理空间

\*\*\*做索引 可以使查询速度变快 但是会导致写入速度变慢

键值的类型

INDEX 普通索引 #最常用

UNIQUE 唯一索引 #不怎么用 他的功能主键都有

FULLTEXT 全文索引

PRIMARYKEY 主键 #常用

FOREIGNKEY 外键 #常用

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

1. INDEX普通索引

使用说明

 一个表中可以有多个INDEX字段

 字段的值允许有重复,切可以赋NULL值

 经常把做查询条件的字段设置为INDEX字段

 INDEX字段的KEY标志是MUL

新建一个表格并创建索引:

create table school.info(

id int(6) not null,

name varchar(5),

sex enum('male','female'),

age int(3) default 1,

index(id),index(name));

在已有的数据表中创建或删除索引。

语法格式:create index 索引名称 on 数据表(字段名称)

create index age on school.info(age);

删除索引: drop index 索引名称 on 数据表(字段名称) 其中索引名称必须通过查询获得

show index from XX\G

drop index name on school.info;

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

2. primary key主键索引

注意事项

 一个表中只能有一个primary key字段

 对应的字段值不允许有重复,且不允许赋NULL值

 如果有多个字段都作为PRIMARY KEY,称为复合主键,必须一起创建。

 主键字段的KEY标志是PRI

 通常与AUTO\_INCREMENT连用

 经常把表中能够唯一标识记录的字段设置为主键字段[记录编号字段]

新建一个表格并创建主键:

create table school.student(

stu\_id char(9),

name char(5),

primary key(stu\_id));

或

create table school.student2(

id char(9) primary key,

name char(5));

对已经存在的数据表创建主键索引

ALTER TABLE 数据表名称 add primary key(字段名称);

alter table school.student3 add primary key(id);

删除数据表中的主键索引

alter table school.student3 drop primary key;

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

复合主键索引(多个字段做主键)

创建符合主键索引(姓名和单位,不能重复,但是单独的姓名或单位可以重复)

create table school.book(

姓名 char(20),

单位 char(20),

技能 int(100), primary key(姓名,单位));

#提示:单独的姓名重复是可以的

自动添加属性(可以自动将数据自动加1)

create table school.demo(

id int(100) auto\_increment primary key,

name char(10));

#必须为 int 类型

#当同时有自增长和主键时,一定要通过 alter 将自增长删除后才可以删除主键,无法直接删除主键

删除添加属性

alter table 表名 modify 类型(宽度);

alter table 表明 change 原字段名 原字段名 类型(宽度);

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

3. foreign key外键

什么是外键?

让当前表字段的值在另一个表中某个字段值的范围内选择。

使用外键的条件:

表存储引擎必须是innodb(默认就是)

字段的类型必须一致

被参照字段必须是索引类型中的一种(primary key)

创建外键的语法:

foreign key(表A的字段名称)

references 表B(字段名称)

on update casecade

#同步更新

on delete casecade

#同步删除

create table press.book(

书名 char(20),

作者 char(10),

foreign key(作者) references press.author(姓名)

on update cascade on delete cascade);

删除外键时,需要先查看外键的名称:

show create table press.book\G;

alter table press.book drop foreign key XXX(外键名)

###############################################################################

修改导出导入文件存放路径

1.查看默认存放路径:

mysql> show variables ; 当前服务器所有变量

mysql> show variables like '%secure%' //导入导出的默认文件路径(/var/lib/mysql-files/)

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

| Variable\_name | Value |

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

| require\_secure\_transport | OFF |

| secure\_auth | ON |

| secure\_file\_priv |/var/lib/mysql-files/| //默认存放路径与变量名称

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

2.修改主配置文件(/etc/my.cnf)

# vim /etc/my.cnf

[mysqld]

secure\_file\_priv="/myload" //添加此行 定义存放路径

:wq

# mkdir /myload //创建配置文件中定义的检索目录

# chown mysql /myload //由于数据库是由mysql用户身份运行的

chmod o+w /myload //也可以将文件其他人身份赋予写的权限(默认有rx)

3.重启服务并检查

#systemctl restart mysqld

#mysql -uroot -pchen

mysql > show variables like '%secure\_file\_priv%' //核查是否生效

4.数据导入：把系统文件的内容存储到数据库的表里

1 命令格式

mysql> load data infile "目录名/文件名"

into table 库名.表名 fields terminated by "分隔符" lines terminated by "\n";

2 注意事项？

3 数据导入步骤

建库

建表

拷贝文件到检索目录下

导入数据

查看表记录

4 例子

mysql> create database db3;

mysql> create table db3.user(

name char(50) , password char(1) ,uid int , gid int , comment varchar(150) , homedir char(60) , shell char(50)

);

mysql> system cp /etc/passwd /myload/

mysql> load data infile "/myload/passwd" into table db3.user fields terminated by ":" lines terminated by "\n" ;

mysql> alter table db3.user add id int primary key auto\_increment first;

mysql> select \* from db3.user ;

mysql> select \* from db3.user where id = 1;

5 数据导出 :把表记录存储到系统文件里

命令格式：

sql查询命令 into outfile “目录名/文件名";

sql查询命令 into outfile “目录名/文件名"

fields terminated by “分隔符”;

sql查询命令 into outfile “目录名/文件名”

fields terminated by “分隔符”

lines terminated by “\n” ;

注意事项？

例子：

mysql> select \* from db3.user where id <= 2 into outfile "/myload/user1.txt" ;

mysql> select \* from db3.user where id <= 2 into outfile "/myload/user2.txt"

-> fields terminated by "#";

mysql> select \* from db3.user where id <= 2 into outfile "/myload/user3.txt" fields terminated by "#" lines terminated by "!!!";

mysql> select name,uid,shell from db3.user where id <= 3 into outfile "/myload/user4.txt";

mysql> system ls /myload

mysql> system cat /myload/user1.txt

2 管理表记录

2.1 插入表记录 insert into

插入1条表记录给所以字段赋值

insert into 库.表 values(值列表)；

插入多条表记录给所以字段赋值

insert into 库.表 values(值列表),(值列表)....;

插入1条表记录给指定字段赋值

insert into 库.表(字段名列表) values(值列表);

插入多条表记录给指定字段赋值

insert into 库.表(字段名列表) values(值列表),(值列表)....;

mysql> insert into db3.user(name)values("bob");

mysql> insert into db3.user(name,uid,homedir) values("bob",19,"/bin/bash");

mysql> insert into db3.user values (23,"jerry","x",2000,2000,"student user","/home/jerry","/bin/bash");

2.2 查询表记录 select

select 字段名列表 from 库.表 where 条件 ；

mysql> select \* from db3.user;

mysql> select name,uid ,password from db3.user;

mysql> select name,uid ,password from db3.user where shell="/sbin/nologin";

2.3 修改记录字段值

update 库.表 set 字段名=值，字段名=值【 where 条件 】；

mysql> update db3.user set password="a" ;

mysql> update db3.user set password="x" where name="root";

2.4 删除表记录 （删除行）

delete from 库名.表名 【where 条件表达式】;

mysql> delete from db3.user where name="bob";

3 匹配条件 （适用于select update delete）

3.1 基本匹配条件

1 数值比较 = != > >= < <=

mysql> select name from db3.user where uid = 0 ;

mysql> select name,uid,gid from db3.user where uid = gid;

mysql> select name,uid,gid from db3.user where uid != gid;

mysql> select \* from db3.user where id <= 10;

2 字符比较 = !=

mysql> select name ,shell from db3.user where shell != "/bin/bash";

mysql> select name ,shell from db3.user where shell = "/bin/bash";

mysql> select name from db3.user where name="mysql";

3 空 is null

非空 is not null

mysql> select name , shell from db3.user where shell is not null;

mysql> select name , shell from db3.user where shell is null;

mysql> insert into db3.user(name)values (null),("null"),("");

mysql> select id ,name from db3.user;

mysql> select id , name from db3.user where name="null";

mysql> select id , name from db3.user where name="";

mysql> select id , name from db3.user where name is null;

4 逻辑匹配 （多个条件）

逻辑与 and 多个条件必须都成立

逻辑或 or 多个条件成立即可

逻辑非 ！/ not 取反 != is not null

mysql> select \* from db3.user where uid=0 or name="bin" ;

mysql> select \* from db3.user where uid=0 and name="bin" ;

5 范围内匹配/去重显示

in (值列表) 在…里…

not in (值列表) 不在…里…

between 数字 and 数字 在…之间…

distinct 字段名 去重显示

mysql> select name ,uid from db3.user where uid in (1 , 9 ,3 ,7);

mysql> select name from db3.user where name in ("demo","sync","bin","root");

mysql> select name , shell from db3.user

where shell not in ("/bin/bash","/sbin/nologin");

mysql> select name ,uid from db3.user where uid between 10 and 40 ;

mysql> select distinct gid from db3.user where id <= 10;

3.2 高级匹配条件

1、 like 模糊查询

where 字段名 like '通配符'

\_ 表示1个字符

% 表示0~n个字符

mysql> select name from db3.user where name like '\_\_\_' ;

mysql> select name from db3.user where name like 'a\_\_' ;

mysql> select name from db3.user where name like 'a%' ;

mysql> select name from db3.user where name like '%a%' ;

mysql> select name from db3.user where name like '\_\_%\_\_' ;

mysql> select name from db3.user where name like '%a%' ;

2、正则表达式

where 字段名 regexp '正则表达式’

元字符 ^ $ . [ ] \* |

mysql> insert into db3.user(name)values("9yaya"),("ya7ya"),("yaya6");

mysql> select name from db3.user where name regexp '^a|t$' ;

mysql> select name from db3.user where name regexp '^[0-9]';

mysql> select name from db3.user where name regexp '[0-9]'

mysql> select name ,uid from db3.user where uid regexp '..';

mysql> select name ,uid from db3.user where uid regexp '^..$';

mysql> select name ,uid from db3.user where uid regexp '^...$';

3 四则运算 + - \* / %

mysql> alter table db3.user add age tinyint unsigned default 10 after name;

mysql> select name ,uid , gid , uid+gid zh from db3.user;

mysql> select name ,age , 2019 - age csnf from db3.user where name="root";

mysql> select name ,uid from db3.user where uid % 2 = 0 ;

mysql> update db3.user set uid=uid+1 where uid <= 10;

mysql> update db3.user set uid=uid-1 where uid <= 10;

mysql> select name , uid , gid , (uid+gid)/2 pjz from db3.user where name = "bin";

3.3 操作查询结果

1 聚集函数(对数据做统计的命令)

avg(字段名) //统计字段平均值

sum(字段名) //统计字段之和

min(字段名) //统计字段最小值

max(字段名) //统计字段最大值

count(字段名) //统计字段值个数

mysql> select avg(gid) from db3.user;

mysql> select max(uid) from db3.user;

mysql> select min(uid) from db3.user;

mysql> select sum(uid) from db3.user;

mysql> select count(name) from db3.user where shell != "/bin/bash";

mysql> select count(\*) from db3.user;

2 查询结果排序 order by 字段 asc/desc ;

mysql> select name ,uid from db3.user where uid <= 1000 order by uid ;

mysql> select name ,uid from db3.user where uid <= 1000 order by uid desc;

3 查询结果分组 group by 字段名；

mysql> select shell from db3.user where uid <= 100 group by shell ;

mysql> select shell from db3.user group by shell ;

4 查询结果过滤 having 条件；

mysql> select name from db3.user where uid <= 3000;

mysql> select name from db3.user where uid <= 3000 having name="mysql";

5 限制查询结果显示行数 limit

只显示查询结果的头几行 limit 数字 ；

mysql> select \* from db3.user where id <= 10 ;

mysql> select \* from db3.user where id <= 10 limit 1;

mysql> select \* from db3.user where id <= 10 limit 2;

只显示查询结果指定行 limit 起始行，行个数

mysql> select \* from db3.user where id <= 10 limit 0,2;

mysql> select \* from db3.user where id <= 10 limit 1,2;

mysql> select \* from db3.user where id <= 10 limit 3,3;

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

\*\*mysql 登录状态下 使用shell 命令

\*\*mysql system XXX

############################################################################################

管理员拥有的权限:

查看mysql数据库中的所有用户

mysql.user 记录已有的授权用户及权限

mysql.db 记录已有授权用户 对数据库 的访问权限 !!若要1个用户有多个库权限 只能逐个加入 因为db表中一个库为一条记录 若要添加只能insert 或用grant 逐一授权

mysql.tables\_priv 记录已有授权用户 对表 的访问权限 !!若要修改某个用户的权限 可以通过 update 来实现

mysql.columns\_priv 记录已有授权用户 对字段 的访问权限

select distinct(可选 不加显示root) concat('User: ''',user,'''@''',host,''';') as query from mysql.user; 查看授权用户及客户端登录地址

select concat('User: ',db,'.',User,'''@''',Host,''';') as query from mysql.db; 查看授权数据库 用户 客户端登录地址

show grants for 用户名@"客户端地址"; 管理员查看已授权用户权限

set password for 用户名@"客户端地址" = password("密码"); 管理员重置授权用户连接密码

客户端拥有的权限:

select user(); 显示登录及客户端地址

show grants; 用户显示自身访问权限

set password=password("密码"); 用户连接后修改连接密码

with grant option用于对象授权时，被授予的用户也可把此对象权限授予其他用户或角色，不同的是但管理员收回用with grant option授权的用户对象权限时，权限会因传播而失效，如：grant select on 表名 to A with grant option;，A用户把此权限授予B，但管理员收回A的权限时，B的权限也会失效，但管理员不可以直接收回B的SELECT ON TABLE 权限。

如果授权用户已经存在 那么可以通过 grant 权限 on 库.表 to用户 赋予权限

如果授权用户对象没有给mysql 的库 那么 没办法给新用户用grant赋予权限

!!!因为 grant命令就是在mysql.user 等表中加入数据

修改配置文件中 skip-grant-tables 中 与 validate\_password\_policy 等密码策略冲突 必须注释等测试后 才能跳过

备份方式

物理备份

-冷备份:cp tar ...

缺点1 跨平台性差 系统操作类型不同 比如linux保存的文件系统 xfs 等格式 到Windows 上保存的文件系统不同!

缺点2 数据量大 拷贝数据时间长

缺点3 时效性差 只记录物理备份前的数据

逻辑备份

mysqldump //备份命令

缺点 会造成锁表 如果数据量大 会造成长时间无法访问此表

-A -B 2个命令中是含有创建库的命令的 所以恢复的时候不带库名

如果没有 -A -B 的参数 恢复命令时 必须带上库名并且此库必须存在

mysql //恢复命令

----