数据库3

**mysql存储引擎**（查询访问多的表使用myisam存储引擎，节省系统资源。

写操作多的表使用innodb存储引擎，并发访问量大）

mysql服务软件自带的功能程序，处理表的处理器

锁类型：

select读锁（共享锁）

update delete insert 写锁（排它锁）

锁粒度：

表级锁：一次对整张表进行加锁

行级锁：只锁定一行

页级锁：对整个页面进行加锁

innodb（mysql5.5-5.7默认存储引擎）：

支持事务，事务回滚，支持行级锁,外键

相关表文件

表名.frm(表结构)

表名.ibd（表数据）

ibdata1

ib\_logfile0

ib\_logfile1

myisam（mysql5.0-5.1默认存储引擎）：

支持表级锁，不支持事务，事务回滚，外键

相关表文件

表名.frm（表结构）

表名.MYI（）

表名.MYD(存储表内数据)

mysql>show engines;列出可用的存储引擎类型

可修改默认存储引擎

vim /etc/my.cnf

default-storage-engine=存储引擎名

事务特性（ACID）：

原子性，不可分割，要么成功，要么失败

一致性，事务操作前后，表内数据无变化（没有提交之前）

隔离性，事务操作互相不影响（没有commit之前）

持久性，数据一旦提交，不可改变，表内数据永久改变（commit之后）

mysql>show variables like “autocommit”;

#查看更改数据时，自动提交是否开启

mysql>set autocommit=off;

#关闭自动提交

mysql>rollback;

#在提交之前，都可以进行回滚

mysql>commit;

#提交

**数据导入导出**

**数据导入**

mysql>load data infile “目录/文件名” into table 库.表 fields terminated by “分隔符” lines terminated by “\n”;

步骤：

建表（根据需要导入文件字段建表）

mysql> show variables like "%secure\_%";

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

| Variable\_name | Value |

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

| require\_secure\_transport | OFF |

| secure\_auth | ON |

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

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

将文件导入之前，需先将文件复制到/var/lib/mysql-files目录下，才能进行导入操作

例如：

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

#添加第一行前面添加字段，设置类型

**数据导出**

mysql>sql查询命令 into outfile “目录/文件名” fields terminated by “分隔符” lines terminated by “\n”;

mysql> select \* from db3.user where id<=8 into outfile "/var/lib/mysql-files/a.txt" fields terminated by ":";

#将表前八行导出到文件，目录仍为/var/lib/mysql-files/，分隔符任意,默认使用tab键分隔

存放文件路径可改变，可在配置文件里修改（/etc/my.cnf/）

vim /etc/my.cnf

secure\_file\_priv=”目录”

可自行创建目录，但目录的所有者必须为mysql

**管理表记录**

添加表记录

insert into 表名 values(字段值列表)或者

insert into 表名（字段名列表） values(字段值列表)

mysql>insert into db3.user values(42,"tom","x",2000,2000,"test user","/home/tom","/bin/bash"),

-> (43,"jack","x",2001,2001,"test user","/home/jack","/bin/bash"),

-> (44,"lucy","x",2002,2002,"test user", "/home/lucy","/sbin/nologin");

#添加表记录，依次给所有字段赋值时，字段名可省略，只给部分字段赋值时，必须明确写出对应的字段名，字段值要与字段类型相匹配

查询表记录

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

#查询满足条件的表记录，where 条件

更新表记录

mysql> update db3.user set password="f",comment="student";

#批量修改

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

#对符合条件的记录进行修改，where 条件

删除表记录

mysql> delete from db3.user where id>=43;

#满足条件的删除

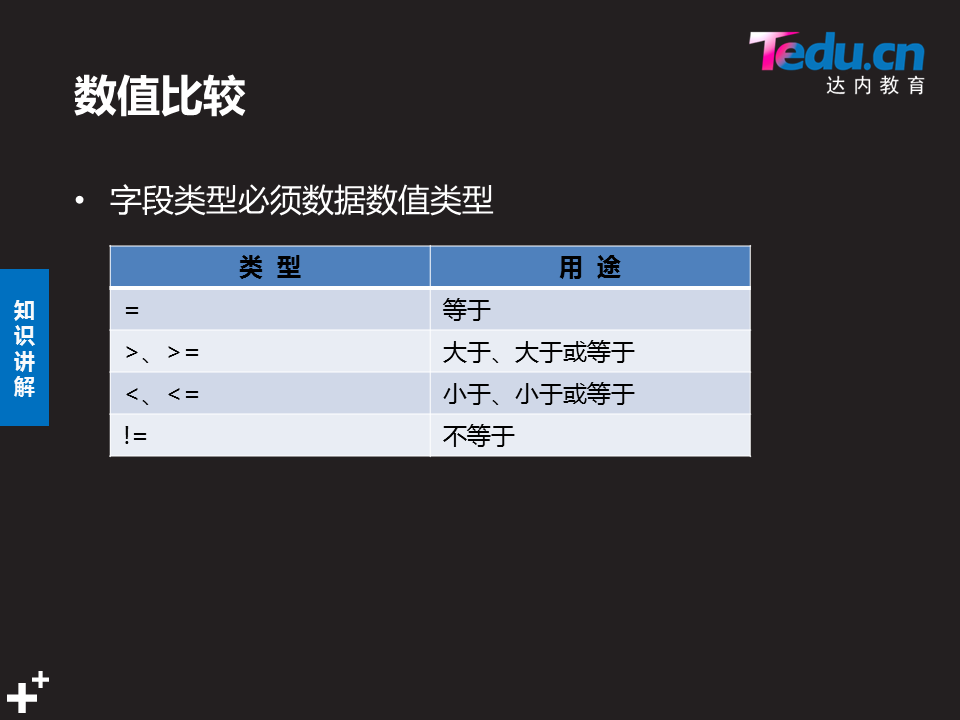
mysql>delete from db3.user;

#删除表内所有记录

**匹配条件**

1.基本匹配条件

字段类型必须为数值类型



字符比较



逻辑匹配（多个匹配条件）

and 逻辑与（同时满足多个条件）

or 逻辑或（满足任何一个条件）

范围内匹配

in(条件)

not in()

between 数值1 and 数值2

distinct 字段名 去重

mysql> select name from user where name in("root","mysql","sync");

#筛选root，mysql，sync 用户

mysql> select name,shell from user where shell not in("/sbin/nologin");

#筛选出不使用/sbin/nologin 解释器的用户

mysql> select name,uid from user where uid between 100 and 200;

#筛选出uid范围在100~200之间

mysql> select distinct shell from user where uid > 100;

#去掉重复的

高级匹配条件

模糊查询

关键字 like

“\_” 表示匹配单个字符

“%” 表示匹配0-n个字符

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

#查询name是四个字符的

mysql> select name from user where name like "%a%";

#查询name中包含a的

正则表达式

mysql> select name from user where name regexp "^r|c$";

四则运算符（对行）

+、-、\*、/（加减乘除）

聚集函数内置函数（对列）

min(字段名)统计取最小值

max(字段名)统计取最大值

sum(字段名)统计字段之和

avg(字段名)统计平均值

count(字段名)统计表记录个数

mysql> select min(uid) from user;

mysql> select max(uid) from user;

mysql> select avg(uid) from user;

mysql> select count(name) from user where uid>100;

查询结果排序

sql查询语句 order by 字段名 asc|desc

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

#降序

mysql> select name,uid from user order by uid asc;

#升序

查询结果分组

sql查询语句 group by 字段名

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

等同于

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

两者区别在于工作过程不一样（group by节省cpu）

查询结果过滤

sql查询语句 having 条件

mysql> select name from user where shell!="/bin/bash" having name="nobody";

等同于

mysql> select name from user where name="nobody" and shell!="/bin/bash";

两者区别：使用having（在上一个结果里面查询，范围缩小）更节省资源，而使用逻辑与，需查询两次（两个条件，全表查询，范围广）

限制查询结果显示行数

mysql> select name,uid,shell from user where uid<=10 limit 2,4;

#显示从第三行开始的四行（第一行行数为0，以此类推）

mysql> select name,uid,shell from user where uid<=10 limit 2;

#显示前两行