# MySQL

## 登录数据库

### 1.1登录

明文登录：mysql -u用户名 -p密码

密文登录：mysql -u用户名 -p 回车 然后输入密码

### 1.2密码重置

a. 停止mysql服务器 运行输入services.msc 停止mysql服务

b. 在cmd下,输入mysqld --console --skip-grant-tables 启动服务器,出现一下页面,不要关闭该窗口

c. 新打开cmd,输入mysql -uroot 不需要密码

use mysql;

update user set password=password('abc') WHERE User='root';

d. 关闭两个cmd窗口

## 二、 数据库操作语言

### 2.1 操作数据库

创建数据库

create database 数据库名;

CREATE DATABASE mydb;

查看有哪些数据库

show databases;

SHOW DATABASES;

使用数据库。

use 数据库名;

USE mydb;

删除数据库

drop database 数据库名;

DROP DATABASE mydb;

### 2.2 操作数据表

#### 创建表的操作

create table 表名(

字段名 数据类型(长度) [约束],

字段名 数据类型(长度) [约束],

字段名 数据类型(长度) [约束]

);

字段名其实就是列名

常用的数据类型。

int 整数

double 小数

varchar 字符串(可变字符)

date 日期，只有年月日，没有时分秒。

注意：常用的数据类型中，varchar必须指定长度，其他的可以不指定。

[约束] 是可选的。可以加，也可以不加。

每一个字段都可以有约束。

#### 查看表

查看该数据库下一共有哪些表。

show tables;

查看表的详情。可以查看到表结构的一些信息。

desc 表名;

#### 删除表

删除表。

drop table 表名;

#### 修改表结构

（1）添加列

alter table 表名 add 列名 类型(长度) [约束];

#为分类表添加一个新的字段为分类描述 varchar(20)

ALTER TABLE category ADD `desc` VARCHAR(20);

（2）修改列的数据类型及约束

alter table 表名 modify 列名 类型(长度) 约束;

#为分类表的描述字段进行修改，类型varchar(50) 添加约束 not null

ALTER TABLE category MODIFY `desc` VARCHAR(50) NOT NULL;

（3）修改列名和数据类型和约束

alter table 表名 change 旧列名 新列名 类型(长度) 约束;

#为分类表的分类名称字段进行更换更换为 snamesname varchar(30)

ALTER TABLE category CHANGE cname snamesname VARCHAR(30);

（4）删除列

alter table 表名 drop 列名;

#删除分类表中snamename这列

ALTER TABLE category DROP snamename;

（5）修改表名

rename table 表名 to 新表名;

#为分类表category改名成 category2

RENAME TABLE category TO category2;

（6）修改表的字符集

alter table 表名 character set 字符集;

#6, 为分类表 category 的编码表进行修改，修改成 gbk

ALTER TABLE category CHARACTER SET gbk;

#### 向表中添加数据( 重要 )

# 全列添加。

# insert into 表名 (列1,列2,列3) values (值1,值2,值3);

# 指定列添加

# insert into 表名 (列1,列3) values (值1,值3);

# 全列值添加。

# insert into 表名 values (值1, 值2, 值3);

# 批量添加

# insert into 表名 values (值1, 值2, 值3),(值1, 值2, 值3),(值1, 值2, 值3);

#### 删除表中数据（ 重要 ）

删除语句：

delete from 表名; 会把所有的数据都给删除。

如果想要删除指定的，可以使用条件进行筛选

delete from 表名 where 条件;

#### 修改表中数据 （ 重要 ）

update 表名 set 列=值,列=值 where 条件;

where条件用作筛选。可以省略。如果省略，会修改所有的数据。

## 三、约束

### 3.1 主键约束

/\*

主键约束 primary key

可以保证该列的数据唯一性，并且不能为NULL。

主键列并没有特殊的含义，只是做一个标识，这个标识因为是主键所以唯一的。

主键不能是业务数据。

一个表中最多只能有一个主键。

从设计上来说，一个表应该有一个主键。

\*/

可以在创建表的时候直接添加. 方式一。

CREATE TABLE person(

-- PRIMARY KEY代表主键约束，保证这列的值不重复，且不能为NULL

p\_id INT PRIMARY KEY,

firstname VARCHAR(20),

lastname VARCHAR(20),

address VARCHAR(20),

city VARCHAR(20)

);

-- 可以在创建表时，在constraint区域添加 方式二。

-- constraint区域指的是创建表的时候最后一个字段的后面

-- [constraint 名称] primary key (列);

DROP TABLE person;

CREATE TABLE person(

p\_id INT,

firstname VARCHAR(20),

lastname VARCHAR(20),

address VARCHAR(20),

city VARCHAR(20),

-- constraint区域，是最后一个字段的后面.

-- constraint p primary key (p\_id)

-- 可以省略名称

-- constraint primary key (p\_id)

-- 最简单写法

PRIMARY KEY (p\_id)

);

-- 可以在创建表之后，使用alter语句添加 第三种方式

-- alter table 表名 add [constraint 名称] primary key (列);

CREATE TABLE person(

p\_id INT,

firstname VARCHAR(20),

lastname VARCHAR(20),

address VARCHAR(20),

city VARCHAR(20)

);

-- 添加主键

ALTER TABLE person ADD PRIMARY KEY (p\_id);

-- 删除主键

-- alter table 表名 drop primary key;

ALTER TABLE person DROP PRIMARY KEY;

/\*

联合主键。

可以把表中的多个字段共同当做一个主键。

constraint区域添加一下格式

[constraint 名称] primary key (列,列);

-- 联合主键，保证所有的值都不重复。

\*/

CREATE TABLE person(

p\_id INT,

firstname VARCHAR(20),

lastname VARCHAR(20),

address VARCHAR(20),

city VARCHAR(20),

-- constraint区域添加联合主键。

PRIMARY KEY (firstname, lastname)

)

### 3.2 自动增长列

/\*

自动增长列。auto\_increment

可以自动去维护数据。

如果我们在添加这一列数据的时候，没有给出这列数据的值，那么这个数据就会自动加1.

auto\_increment 一定要对整数列使用。

一般都会对主键加上自动自动增长，让他自己去维护。

\*/

-- 创建person表对主键添加自动自动增长约束

CREATE TABLE person (

-- primary key 代表这是主键。 auto\_increment 代表这列数据会自动增长。

p\_id INT PRIMARY KEY AUTO\_INCREMENT,

pname VARCHAR(20),

address VARCHAR(20)

);

-- delete from 表名; 删除所有的数据。 不会重置自增长序列。

DELETE FROM person;

-- truncate table 表名; 删除所有数据，会重置自增长序列。 原因是因为这个是先删除整张表，然后再新建表。

TRUNCATE TABLE person;

### 3.3 非空约束

/\*

非空约束 NOT NULL

保证这列的数据不能出现NULL值。

可以出现'' 'null'

\*/

-- 在创建表的时候直接添加 --------- 添加方式一

-- 创建person表。 p\_id 主键自增。 pname 非空约束。 address

CREATE TABLE person(

p\_id INT PRIMARY KEY AUTO\_INCREMENT,

-- not null代表是非空约束

pname VARCHAR(20) NOT NULL,

address VARCHAR(20)

);

-- 在创建表之后使用alter语句修改表结构进行添加 -- 方式二

DROP TABLE person;

CREATE TABLE person(

p\_id INT PRIMARY KEY AUTO\_INCREMENT,

pname VARCHAR(20),

address VARCHAR(20)

);

-- alter语句修改表结构添加约束

ALTER TABLE person MODIFY pname VARCHAR(20) NOT NULL;

-- 删除这个非空约束

-- 也是使用alter语句修改表结构

ALTER TABLE person MODIFY pname VARCHAR(20);

### 3.4 唯一约束

/\*

唯一约束：UNIQUE

保证这列的数据是唯一的，不能重复。

主键在一个表中最多只能有一个。

而唯一约束可以有多个。

唯一约束不能限制多个NULL值。

\*/

-- 创建person表。 p\_id主键，并且自动增长。 pname，唯一约束。 address..

CREATE TABLE person(

p\_id INT PRIMARY KEY AUTO\_INCREMENT,

-- UNIQUE代表唯一约束，这列的数据不能出现重复的

pname VARCHAR(20) UNIQUE,

address VARCHAR(20)

);

-- 可以在创建表的constraint区域添加 方式二

-- [constraint 名称] unique (列)

CREATE TABLE person(

p\_id INT PRIMARY KEY AUTO\_INCREMENT,

pname VARCHAR(20),

address VARCHAR(20),

-- constraint区域添加唯一约束

UNIQUE (pname)

);

-- 创建完成表之后，使用alter语句添加唯一约束 第三种方式

CREATE TABLE person(

p\_id INT PRIMARY KEY AUTO\_INCREMENT,

pname VARCHAR(20),

address VARCHAR(20)

);

-- 使用alter语句添加唯一约束

-- alter table 表名 add [constraint 名称] UNIQUE (列名);

ALTER TABLE person ADD UNIQUE (pname);

-- 删除唯一约束

-- alter table 表名 drop index 约束名字;

ALTER TABLE person DROP INDEX pname;

## 四、查询语句【非常重要】

### 4.1 简单查询

简单查询

select 列名,列名 from 表名;

如果要查询所有的列。可以使用\*

select \* from 表名;

### 4.2 条件查询

运算符：用作条件的筛选。

> < >= <= = != <>

between ... and ...：在区间范围内。比如 between 1 and 10。代表1-10之间。包含头也包含尾。

in(...)： 包含其中之一。 比如in(1,3,5)。只要包含1,3,5其中之一就可以。

like：用作模糊查询。

%：0个或1个或多个任意字符。

\_：1个任意字符。

IS NULL：判断是否为空。

AND：与，相当于java中的&。必须所有条件全部满足。

OR： 或，相当于java中的|。 满足一个条件就可以。

NOT: 非。 相当于java中的!.取反

条件查询的格式：

select 列名 from 表名 where 条件;

### 4.3 排序查询

可以对查询后的结果根据指定的字段进行排序。

格式：

select \* from 表名 order by 要排序的字段 [asc | desc];

asc：代表升序排序

desc：代表降续排序。

这两个单词可以都不写，如果都不写默认升序排序。

### 4.4 分组查询

group by 要分组的字段;

分组查询一定要结合聚合函数去查询。

where 要在分组之前，对原始数据进行条件筛选过滤

having 用在分组之后，对筛选之后的数据再进行条件过滤

where 和having 只有使用位置不同，作用一样，都是用来进行条件过滤

### 4.5 聚合函数

聚合函数。

用来纵向查找。

count(列)： 统计该列有多少条数据。不会统计NULL值。

max(列)：求该列数据的最大值。

min(列)： 求改列数据的最小值。

sum(列)： 求该列数据的和。

avg(列)： 求该列所有数据的平均值。

聚合函数加在要查询的字段的位置。

建议在sql语句中使用关键字的地方使用``包裹

## 五、多表

### 5.1 多对一（一对多）

比如：部门和员工，客户和订单，分类和商品

建表原则： 多对一关系中，需要在从表中建立外键列，外键列指向主表中的主键

### 5.2 多对多

比如：学生和课程、用户和角色

建表原则：建立中间表，中间表至少两个字段，这两个字段当做外键指向各一方的主键

### 5.3 一对一

比如：丈夫和妻子

一般一对一关系新建一张表即可

## 六、多表查询【非常重要】

### 6.1 交叉连接查询

交叉连接查询

select \* from 表A, 表B;

可以同时查询表A和表B

会产生笛卡尔积，结果是错误的。

### 6.2 内连接查询

隐式内连接：

select \* from 表A, 表B where 条件;

显式内连接：

select \* from 表A inner join 表B on 条件;

一定要加条件，否则会产生笛卡尔积.

inner 可以省略

只会查询关联的数据

### 6.3 外连接查询

左外连接：

select \* from 表A left outer join 表B on 条件;

会查询左表中所有的数据。右表只会查询关联的数据.

右外连接

select \* from 表A right outer join 表B on 条件;

查询右边表中的所有数据，左表只会查询关联的数据

outer也可以省略

# JDBC

## 一、概念

jdbc是java访问数据库的一套规范。

## 二、开发步骤

1. 注册驱动

2. 获取连接

3. 获取sql语句执行者对象

4. 执行sql

5. 处理结果集

6. 释放资源

## 三、注册驱动两种方式

第一种：DriverManager.registerDriver(new Driver());

这种方式会注册两次，不推荐

第二种：Class.forName("com.mysql.jdbc.Driver");

只会注册一次

## 四、JDBC代码示例

public class Demo01JDBC {

public static void main(String[] args) throws SQLException {

//1. 注册驱动。

Class.forName("com.mysql.jdbc.Driver");

//2. 获取连接

String url = "jdbc:mysql://127.0.0.1:3306/mydb"; //url

String username = "root";//用户名

String password = "sorry";// 密码

Connection con = DriverManager.getConnection(url, username, password);

//3. 通过连接对象获取sql语句执行者对象。

Statement st = con.createStatement();

//4. 使用执行者对象执行sql语句

String sql = "insert into category (cname) values ('床上用品')";

int row = st.executeUpdate(sql);

System.out.println("row:" + row);

//5. 处理结果集 (只有查询要处理)

// 不需要，因为现在是做的添加的操作。

//6. 释放资源

st.close();

con.close();

}

}