MySQL中的CRUD操作及DML语句

# 对于数据的操作CRUD

CRUD：分别指Create、Read（Retrieve）、Update、Delete，简称增删改查。

**CRUD**是指在做计算处理时的**增加(Create)、读取(Retrieve)（重新得到数据）、更新(Update)和删除(Delete)**几个单词的首字母简写。主要被用在描述软件系统中数据库或者持久层的**基本操作功能**。

In computing, CRUD is an acronym for create, retrieve, update, and delete. It is used to refer to the basic functions of a database or persistence layer in a software system.

**C reate new records**

**R etrieve existing records**

**U pdate existing records**

**D elete existing records.**

在MySQL中，对于数据的操作包括：①创建（插入）数据记录（**Create**），②获取（查询）数据记录（**Retrieve**）③更新数据记录（**Update**）④删除数据记录（**Delete**）。

其中，插入数据记录、更新数据记录、删除数据记录属于**DML语句**，而查询数据记录属于**DQL语句**。

简单语法：

插入数据：**insert into** tbName(field1,field2,...) values(v11,v12,...),(v21, v22,...),...

更新数据：**update** tbName **set** field1 = v1,field2=v2 where condition;

删除数据：**delete from** tbName where field1=v1;

查询数据：**select** \* **from** tbName;

只有删除数据与查询数据语句中有‘**from’**，而插入数据与更新数据中没有‘**from’**。

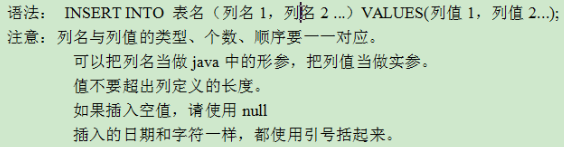
DML、DQL语句中都不带有‘table’，直接输入tbName即可。对于DDL语句，必须添加‘table’，然后在后面输入**tbName**。由于**DDL的操作对象是数据库或表格**，所以在语句中应该声明目标是**database**还是**table**。而DML与DQL语句只是针对表格中的数据记录操作，所以不用加‘table’关键字。

# 数据操作语言DML：Data Manipulation Language

DDL：就是用来**操作**数据库表格中的**数据（记录）**的语言；DDL命令包括：insert、update、delete，即增删改。DQL只是数据查询语言，Data Query Language。

# 插入数据：insert into tbName(f1,f2) values(v1,v2)

## 语法： insert into tbName(f1,f2,f3) values(v1,v2,v3),(v1,v2,v3),(v1,v2,v3)...



注意：

### 列名与列值必须意义对应，包括数据类型、个数、顺序；

### char或varchar数据类型，值不要超出规定的长度；

### 如果需要插入NULL，直接使用NULL即可；

### 插入date、time、datetime等数据类型值时，和插入字符串一样，用引号括起来，注意格式即可，yyyy-MM-dd hh:mm:ss

对于时间类型的值，虽然是以**字符串形式**插入的，但是插入的时候仍会检验插入的值是否合法，如不合法，则报错。也可以是**直接插入包含一串数字的字符串**。



## 插入完整的数据记录与部分数据记录

### 插入完整的数据记录的两种方式：

方式一：**insert into** tbName(f1,f2,...,fn) values(v1,v2,...vn);

方式二：**insert into** tbName values(v1,v2,...vn);

方式一中的f1、...fn代表表格的所有字段，vn与fn对应，可读性强，字段的顺序可以随意调整，灵活性比较强；方式二中的字段名省略，比较简单，但是可读性比较差，灵活性比较差，values中的值必须与表格的字段顺序、个数保持一致。为了提高可读性和灵活性，建议使用第一种方法。

### 插入部分数据记录：利用方式一，需要插入几个字段，就指定几个字段即可。

## 支持批量插入：直接在后面拼接values值就可以。

**可以一次插入多条数据**。**语法：values(v1,v2,v3),(v1,v2,v3),(v1,v2,v3)...**

**insert into** ds **values**(5,'aa','2016-11-17 17:21:32'),(6,'bb','2016-11-17 17:32:56'),(7,'cc','2016-11-18 21:43:59');

## 插入查询结果：insert into tb1Name(f1,f2,f3,...) (select f1,f2,f3,... from tb2Name);

将**查询语句的查询结果**作为插入数据的值**values**。实现数据复制的功能。

如果复制整个表格的内容到另一个新的表格中

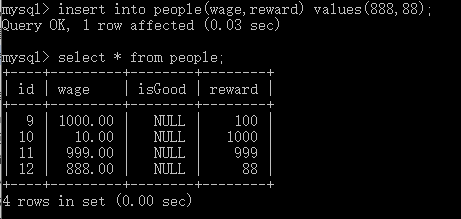
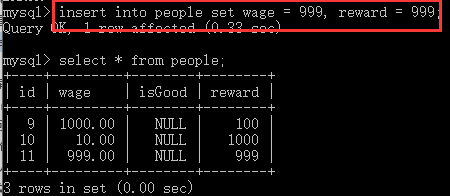
**create table newTbName (as) (select \* from tbName);** 利用**as**实现数据的复制，这里只是复制了数据和数据类型，对于数据的完整性并没有复制，比如自增、默认值等不会被复制。

## 利用insert into tbName set f1=v1,f2=v2; 插入某一列或某些列的值

语法：**insert into** tableName **set** field1=value1,field2 = value2,...;

insert into people set wage = 999, reward = 999;

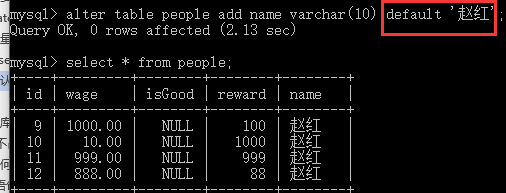
等价于 insert into people(wage,reward) values(999,999);



## 设置默认值：default

在**定义字段时或添加字段或修改字段**属性时利用default value定义默认值。

**alter table** people **add** name varchar(10) **default '赵红'**;



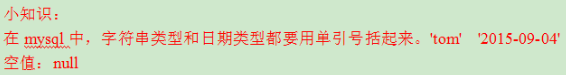
利用not null 限定该字段不能为null，即不能为空。

alter table people add name varchar(10) **not null**;

若添加一个数据时，该字段没有值的话，就会报错。

利用**auto\_increment** 限定该字段自增；利用primary key 限定该字段为主键。

## NULL与‘’



### 在数据库中一般都使用单引号，尽量不要采用双引号；（双引号可以使用）；

### NULL不占用内存，而‘’会占用内存。（Java中也一样）

### 无论任何数据类型，都可以设置为NULL，但是有些字段是有限制的，NOT NULL。

修改某个字段为NULL：

**update** tbname **set** field=NULL where id=1；

# 更新数据记录Update the Records:（简单）

## 语法： update tbName set f1=v1,f2=v2 where conditions;

注意，update-set-where，没有from。

## 一般修改数据都是对某条或某些数据记录的某个或某些字段进行更新，所以一般配合where 进行条件选择，否则会把所有的记录都进行了修改（谨慎操作）。

### 不添加where语句：update tbName set f1=v1;更新所有的数据记录。

### 添加where语句:update tbName set f1=v2 where condition;更新where选定的特定数据记录。

## 可以对一个或多个字段进行修改

**update users set f1=v1,f2=v2,f3=v3 where username in(‘henan’,’hubei’);**

# 删除数据记录Delete the Records：删除数据记录

## 语法 : delete from tbName where conditions;

**删除的是整个记录，**不能删除某个字段的值，所以delete与from之间不加\*或字段名。删除某个字段的值，对应的是update，更新某个字段为null即可。

## 一般删除的是某条或部分数据记录，所以一定要配合where condition筛选。

使用delete命令时，一定要慎重，注意使用where，否则删除了整个表的内容。如**delete from tableName表示清空表格。**同**UPDATE**一样，一般与WHERE一起使用，否则将整个表中的所有记录都会删除掉，谨慎操作。

**delete from tableName; #清空该表的所有内容**

区别于：**drop table tableName**; #删除这个表格（数据和表格都被删除）

delete from tableName where id = 10;#删除id为10的那一整行

# TRUNCATE语句（truncate table tbName）

补充：**truncate** 英 [trʌŋ'keɪt; 'trʌŋ-] 美 [trʌŋ'ket]

adj. 截短的；被删节的 vt. 把…截短；缩短；使成平面

## 语法： truncate [table] tablename; table可以省略

**注意：drop table tbName; 这里的table不可以省略。**

**所以为了统一，建议drop和truncate 中间都加上table。**

## 功能：truncate : 删除表中的所有的记录（数据）；

truncate的功能可以理解为两个功能的组合。一是利用drop table tbName删除整个表格，然后是根据原表格的结构创建（create）一个空表。

## delete、drop、truncate的区别

### delete是DML语句，操作的是数据记录，是用来删除某条或某些数据记录，操作对象是数据记录。

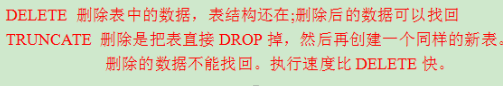
delete from tbName; 功能：删除了所有的数据记录

### drop是DDL语句，是用来删除表格，操作对象是表格。不针对具体的数据。

**drop table** tbName; 删除tbName表格

### truncate也是DDL语句，首先利用drop table tbName;删除表格，然后再利用create table创建一个和tbName结构相同的表格。

### 可以看出，truncate table tbName;与delete from tbName;作用类似，都是清空表格。但是，有所区别。



* + - 1. delete 可以删除某一条或某些特定的数据记录，但是truncate只可以删除所有的记录（数据）。
      2. delete删除的数据在一定条件下**可以恢复**，但是truncate table tbName;不可以恢复。
      3. 由于delete一条一条记录删除，操作较慢，truncate内部直接利用drop table tbName;直接把整个表删除，显然**truncate的操作速度比delete更快**。
      4. 对于有**自增的id字段**时候，delete from tbName;执行后，添加新的数据记录，此时id值会在原来的值的基础上继续增加；但是truncate table tbName;之后，id值从1开始增加。