列属性完整性

说在前面的话：列属性，字段在某些情况下会有一些特殊的属性，哈哈专业术语很有逼格。大白话翻译就是，你设置的 字段（数据库表中要填写信息的要求），有一些特殊性质，比如说，不能没有信息，不能重复信息，自动增长信息，可以不填信息但要给个提示，解释一下信息是干嘛的。完整性还用解释么？哎呀，就是就是逼格词罢了，大白话啊，就是你在写字段的时候要考虑好，这个字段的属性和其他表的关系，再去定义。

作者给的建议：数据库的版本不同，可能在属性上的定义会有一些差距，作者在查资料学习中也遇到矛盾，有的说可行有的说不可行。可不可行自己试试就知道，多在终端机上实际写写练习练习，网上的资料终究是参考。

列属性

空属性

null和　not null

解析:null 字段非强制填数据。

　　not null　字段强制填数据。

增改查

曾(create)

```
create table `表名`(
`字段名`数据类型（字符宽度）not null,
`字段名`数据类型（字符宽度）
);
```

解析：字段非强制填数据后面不用写关于空属性关键字。

删

```
alter table `表名` modify 字段名 字段类型(长度);
```

空属性是个二元对立的属性要么强制要么不强制，只能改不存在删的问题,所谓的删就是改成非强制罢了。

改（alter)

```
alter table `表名` modify 字段名 字段类型(长度) null;
```

```
alter table `表名` modify 字段名 字段类型(长度) not null;
```

查（show）

```
show create table `表名`;
```

解析：MySQL语句显示。

```
desc `表名`;
```

解析：表格形式显示。

注意：modify 和 change 的功能类似，但是 modify 不能修改字段名,字段顺序。

详见例图，基本命令文档关于表的操作中。

默认值

default

解析：“默认值约束（Default Constraint）”，用来指定某列的默认值。在表中插入一条新记录时，如果没有为某个字段赋值，系统就会自动为这个字段插入默认值。

大白话解析：可以不填数据但要给个提示。

比如说：收集同学的手机号，有的同学不想给你，防止尴尬，你设个默认值：“暂时未知”，这样当同学不想给你手机号时你就可以填**default**，在表中就显示暂时未知。

增删改查

曾（create）

```
create table `表名`(
`字段名`数据类型（字符宽度）default '默认值内容'
);
```

删

```
alter table `表名` modify 字段名 字段类型(长度);
```

改（alter）

```
alter table `表名` modify 字段名 字段类型(长度) default '要改的内容';
```

查（show）

同上，空属性。

注意：not null 和 default 是两个独立的约束，不可以用在一个字段上。

自动增长

auto_increment

解析：自增编号，当设定编号字段，使用 auto_increment 属性可使编号自动递增（递增多少可调的但是真的不常用，有兴趣可自行查询），默认递增 +1。

注意和特点

在使用该属性要确定：

1.使用该属性的数据必须是正整数不能为负数，不能为小数。

2.该属性必须在主键（primary key）的属性基础上才能使用。

曾（create）

```
create table `表名`(
`字段名`数据类型（字符宽度）primary key auto_increment
);
```

删

```
alter table `表名` modify 字段名 字段类型(长度);
```

改（alter）

```
alter table `表名` modify 字段名 字段类型(长度) auto_increment;
```

查（show）

同上空属性

主键 PRI

primary key

解析：确定数据的存在性，唯一性，绝对性，标识性是表中最重要的属性，它所定义的 字段属性在表中，还是表之间关系中起着至关重要的作用。 主键的功能及优点，就好比身份证号的功能和优点。尽量用数字作主键，方便查询。

注意和特点

1.定义成主键属性的字段数据不能为空（null），如果设置了自动增长则可填。

2.定义成主键属性的字段数据不能重复。

3.一张表只能有一个主键，但是主键可以有多个字段组成。

4.主键在本表确定唯一性方便查找数据，也可以和其他有关系的表进行关联提供查询数据的依据。

5.设置成主键尽量不要改动

综上所述我们在定义主键的时候一定要考虑好，选择符合需求以及条件的字段。

曾

```
create table `表名`(
`字段名`数据类型（字符宽度）primary key
);
```

注意：其他属性放在前面 primary key 一定要放在最后面，注释的前面。

删

```
alter table `表名` drop primary key;
```

注意:删除主键时，如果还设置自动增长需要先删自动增长。

改

前期未设置主键后期添加

```
alter table `表名` add primary key (`字段名`);
```

查

同上空属性

复合主键

多个字段组成一个主键，不常用，但是遇到极端情况还是需要使用的。

注意:

当我们要创建复合主键时，不能直接通过上面讲解新建主键的方法，那样会报错。

新建一个复合主键

要被选则成为复合主键的字段在新建时不要有任何附加属性**（如果有请先删掉其他属性）**。

然后再用我们上面学到的后期添加主键的方法添加

```
alter table `字段名` add primary key (`字段名`, `字段名`);
```

例：新建一张表

```
mysql> create table `t_04`(
    -> id int(4) unsigned auto_increment primary key,
    -> name varchar(30) not null,
    -> phone varchar(20) default '暂时未知'
    -> );
Query OK, 0 rows affected (0.05 sec)

mysql> desc  `t_04`
    -> ;
+-------+-----------------+------+-----+----------+----------------+
| Field | Type            | Null | Key | Default  | Extra          |
+-------+-----------------+------+-----+----------+----------------+
| id    | int(4) unsigned | NO   | PRI | NULL     | auto_increment |
| name  | varchar(30)     | NO   |     | NULL     |                |
| phone | varchar(20)     | YES  |     | 暂时未知 |                |
+-------+-----------------+------+-----+----------+----------------+
3 rows in set (0.00 sec)
```

要把 id和 phone 改为复合主键，但是他们有自增长和默认值属性要删掉，并且 id是主键也要删掉该属性。

```
mysql> alter table `t_04` modify phone varchar(20);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

```
mysql> alter table `t_04` modify id int(4) unsigned;
Query OK, 0 rows affected (0.11 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> alter table `t_04` drop primary key;
Query OK, 0 rows affected (0.09 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc `t_04`;
+-------+-----------------+------+-----+---------+-------+
| Field | Type            | Null | Key | Default | Extra |
+-------+-----------------+------+-----+---------+-------+
| id    | int(4) unsigned | NO   |     | NULL    |       |
| name  | varchar(30)     | NO   |     | NULL    |       |
| phone | varchar(20)     | YES  |     | NULL    |       |
+-------+-----------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

创建完成

```
mysql> alter table `t_04` add primary key (id,phone);
Query OK, 0 rows affected (0.10 sec)
Records: 0  Duplicates: 0  Warnings: 0

mysql> desc `t_04`;
+-------+-----------------+------+-----+---------+-------+
| Field | Type            | Null | Key | Default | Extra |
+-------+-----------------+------+-----+---------+-------+
| id    | int(4) unsigned | NO   | PRI | NULL    |       |
| name  | varchar(30)     | NO   |     | NULL    |       |
| phone | varchar(20)     | NO   | PRI | NULL    |       |
+-------+-----------------+------+-----+---------+-------+
3 rows in set (0.00 sec)

mysql>
```

删除方法同主键相同

唯一键 UNI

unique

解析：保证数据的唯一性，不重复。

注意与特点

1.可设置多个唯一键的字段。

2.唯一键可以为空（null），可以使用默认值。

3.唯一键只作用与本表，与其他表无关联。

曾

```
create table `表名`(
`字段名`数据类型（字符宽度）unique,
`字段名`数据类型（字符宽度）unique
);
```

删

```
alter table `表名` drop index 字段名;
```

改

给字段添加唯一键

```
alter table `表名` add unique (字段名);
```

也可以用 change 语句修改字段属性添加。

查

同上

注释

内注释

单行注释

#

```
 SELECT 1+1;     # This comment continues to the end of line
```

--

```
SELECT 1+1;     -- This comment continues to the end of line
```

需要特别注意这种注释--后面要加一个空格

可多行可单行

/* */

单行注释

```
SELECT 1 /* this is an in-line comment */ + 1;  /* this is an in-line comment
*/
```

可在在代码中注释

多行注释（直接在终端演示写代码块不方便）

```
mysql> SELECT 1+
    -> /*
   /*> this is a
   /*> multiple-line comment
   /*> */
    -> -1
    -> ;
+--------+
| 1+

-1 |
+--------+
|      0 |
+--------+
1 row in set (0.00 sec)
```

代码注释（实际开发必须写）

comment

代码注释要放在一行的最后面

```
create table `表名`(
`字段名`数据类型（字符宽度）commnent'注释内容',
`字段名`数据类型（字符宽度）commnent'注释内容'
);
```

外键

foreign key　　需要去了解，在实际开发中，尤其是 并发项目　是禁止使用外键的！

解析：外键用来建立主表与从表的关联关系，为两个表的数据建立连接，约束两个表中数据的一致性和完整性。对于两个具有关联关系的表而言，相关联字段中主键所在的表就是主表（父表），外键所在的表就是从表（子表）。

注意与特点

1.主表删除某条记录时，从表中与之对应的记录也必须有相应的改变。

2.一个表可以有一个或多个外键。

3.外键可以为空值，若不为空值，则每一个外键的值必须等于主表中主键的某个值（这里说的是从表里的外键）。

4.外键必须是主表定义的主键。

5.外键中列的数据类型必须和主表主键中对应列的数据类型相同。

曾

在创建表时添加外键

```
create table `表名`(
字段名 类型(宽度)字段属性,
字段名 类型(宽度)字段属性,
外键的字段名 类型(宽度),
foreign key (外键的字段名) references `主表名`(主表主键字段名)
);
```

解析：创建外键的字段名 类型(宽度)，要与主表你要关联的主键名字类型宽度要相同。

foreign key　插入外键

references　参考的意思，这里用于定向你要关联的主表名字和字段。所以后边写`主表名`(主表主键字段名)

例：我们用水果店买水果案例来带入（订单数，多少钱，什么水果）

新建一张名为水果的主表

```
mysql> create table `fruit`(
    -> id int(4) primary key,
    -> name varchar(30) comment'水果名'
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql> desc `fruit`;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| id    | int(4)      | NO   | PRI | NULL    |       |
| name  | varchar(30) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

建立一张名为水果订单测从表，当中需要引用水果的主键id进行关联

注意：关联语句不能使用comment注释。

```
mysql> create table `fruit_order`(
    -> order_id int(11) primary key comment'订单数为主键id',
    -> money decimal(10,4) comment'订单所花钱数',
    -> id int(4) comment'新建外键名以及类型和主表主键一致',
    -> foreign key (id) references fruit(id)
    -> );
Query OK, 0 rows affected (0.05 sec)
```

查

```
show create table `表名`
```

只能使用该方法查询，外键只能靠SQL语句才能看到。

例：接上面水果店案例我们开始查询。

```
mysql> show create table `fruit_order`;
+-------------+---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| Table       | Create Table                                                                                                                                                                                                                          |
+-------------+---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| fruit_order | CREATE TABLE `fruit_order` (
  `order_id` int(11) NOT NULL COMMENT '订单数为主键id',
  `money` decimal(10,4) DEFAULT NULL COMMENT '订单所花钱数',
  `id` int(4) DEFAULT NULL COMMENT '新建外键名以及类型和主表主键一致',
  PRIMARY KEY (`order_id`),
  KEY `id` (`id`),
  CONSTRAINT `fruit_order_ibfk_1` FOREIGN KEY (`id`) REFERENCES `fruit` (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=gbk |
+-------------+---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

红框语句，显示是我们外键创建成功。

删

注意： 我们在删除外键时，要删的是外键真正的名字，这个名字是系统创建的。接上面水果店从表查询例图

CONSTRAINT 后面所接的 `fruit_order_ibfk_1` 这条语句才是外键真正的名字。下面是删除语句语法。

```
alter table `从表名` drop foreign key 真正的外键名;
```

改

后期添加从表，从表里外键设置也要满足与主表主键相同的约束。

注意： 尽量避免后期添加外键这种情况，在创建表的结构时就应该规划好。当数据库已经开始录入数据，这时再改工作量会超级的大。

```
alter table `从表明` add foreign key (外键字段名) references 主表名(主键字段名);
```

置空和级联

该操作是关于外键数据的更新与删除

该操作只能在表新建的时候添加

置空操作用于删除数据

```
on delete set null
```

主表数据删除后从表对应的外键数据会变成 NULL

级联操作用于更新数据

```
on update cascade
```

主表数据更新后从表数据也会对应改变

设置 置空级联 操作语法

创建从表时添加

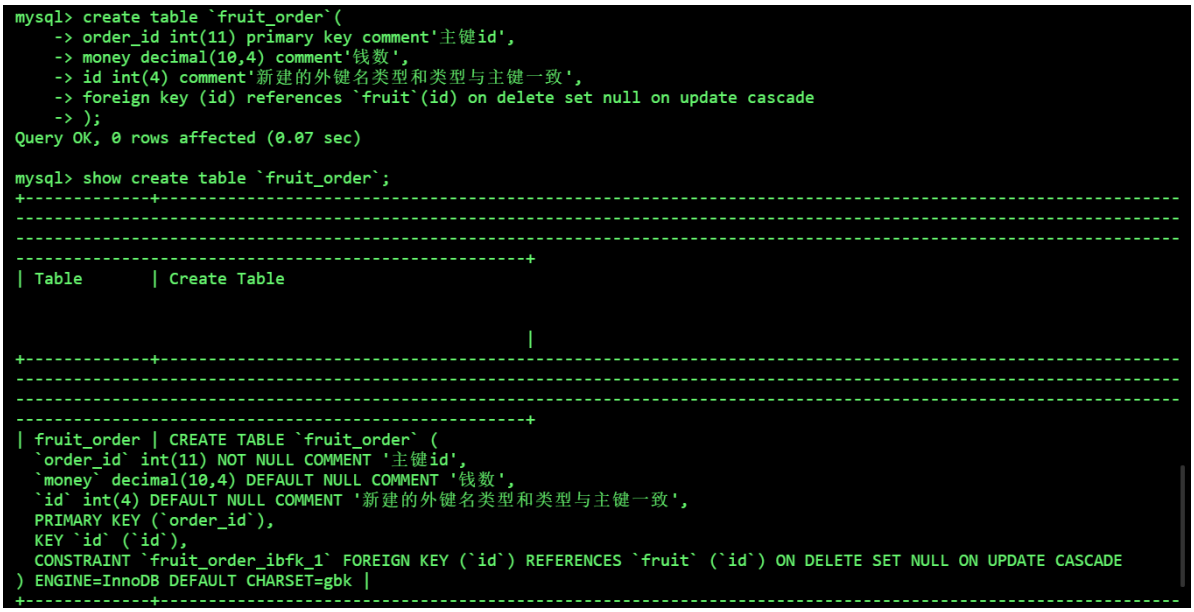
```
create table `表名`(
字段名 类型(宽度)字段属性,
字段名 类型(宽度)字段属性,
外键的字段名 类型(宽度),
foreign key (外键的字段名) references `主表名`(主表主键字段名) on delete set null on
update cascade
);
```

on update cascade 在更新时级联

on delete set null 在删除时置空

例

接上面水果店案例，但是我们在这之前得将从表删除重新新建。



接下来我们往主表和从表中插入数据

主表数据

```
mysql> select *from `fruit`;
+----+------+
| id | name |
+----+------+
|  1 | 苹果 |
|  2 | 香蕉 |
|  3 | 西瓜 |
|  4 | 榴莲 |
+----+------+
4 rows in set (0.00 sec)
```

从表数据

```
mysql> select * from `fruit_order`;
+----------+------------+------+
| order_id | money      | id   |
+----------+------------+------+
|        1 |   111.1100 |    1 |
|        2 |   222.2200 |    2 |
|        3 |   333.3330 |    3 |
|        4 |   444.4400 |    4 |
|        5 |   555.5000 |    1 |
|        6 |   212.3100 |    2 |
|        7 |  2212.3000 |    3 |
|        8 |   256.3900 |    4 |
|        9 |   345.1200 |    1 |
|       10 |  1231.2400 |    2 |
|       11 |   983.4500 |    3 |
|       12 | 87345.0000 |    4 |
+----------+------------+------+
12 rows in set (0.00 sec)
```

主表删除数据西瓜后从表数据变化

```
mysql> delete from `fruit` where id='3';
Query OK, 1 row affected (0.01 sec)

mysql> select * from `fruit_order`;
+----------+------------+------+
| order_id | money      | id   |
+----------+------------+------+
|        1 |   111.1100 |    1 |
|        2 |   222.2200 |    2 |
|        3 |   333.3330 | NULL |
|        4 |   444.4400 |    4 |
|        5 |   555.5000 |    1 |
|        6 |   212.3100 |    2 |
|        7 |  2212.3000 | NULL |
|        8 |   256.3900 |    4 |
|        9 |   345.1200 |    1 |
|       10 |  1231.2400 |    2 |
|       11 |   983.4500 | NULL |
|       12 | 87345.0000 |    4 |
+----------+------------+------+
12 rows in set (0.00 sec)
```

主表改变苹果id为10从表的变化

```
mysql> update `fruit` set id='10' where name='苹果';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> select * from `fruit_order`;
+----------+------------+------+
| order_id | money      | id   |
+----------+------------+------+
|        1 |   111.1100 |   10 |
|        2 |   222.2200 |    2 |
|        3 |   333.3330 | NULL |
|        4 |   444.4400 |    4 |
|        5 |   555.5000 |   10 |
|        6 |   212.3100 |    2 |
|        7 |  2212.3000 | NULL |
|        8 |   256.3900 |    4 |
|        9 |   345.1200 |   10 |
|       10 |  1231.2400 |    2 |
|       11 |   983.4500 | NULL |
|       12 | 87345.0000 |    4 |
+----------+------------+------+
12 rows in set (0.00 sec)
```