主要有如下几种约束：

主键约束：形式： primary key (字段名);
含义（作用）：使该设定字段的值可以用于“唯一确定一行数据”，其实就是“主键”的意思。

唯一约束：形式： unique key (字段名);
含义（作用）：使该设定字段的值具有“唯一性”，自然也是可区分的。

外键约束：形式： foreign key (字段名) references 其他表名(对应其他表中的字段名);
含义（作用）：使该设定字段的值，必须在其谁定的对应表中的对应字段中已经有该值了。

非空约束： 形式： not null，其实就是设定一个字段时写的那个“not null”属性。

默认约束： 形式： default XX 值；其实就是设定一个字段时写的那个“default 默认值”属性

检查约束： 形式： check（某种判断语句），比如：

```
create table tab1 (
    age tinyint,
    check (age>=0 and age <100)    /*这就是检查约束*/
)
```

其实，主键约束，唯一约束，外键约束，只是“同一件事情的 2 个不同角度的说法”，他们同时也称为“主键索引”，“唯一索引”，“外键索引”。

表选项列表

表选项就是，创建一个表的时候，对该表的整体设定，主要有如下几个：

charset = 要使用的字符编码，

engine = 要使用的存储引擎（也叫表类型），

auto_increment = 设定当前表的自增长字段的初始值，默认是 1

comment = ‘该表的一些说明文字’

说明：

1，设定的字符编码是为了跟数据库设定的不一样。如果一样，就不需要设定了：因为其会自动使用数据库级别的设定；

2，engine（存储引擎）在代码层面，就是一个名词：**InnoDB, MyIsam**, BDB, archive, Memory。默认是 InnoDB。

什么叫做存储引擎？

存储引擎是将数据存储到硬盘的“机制”。其实，也就几种机制（如上名字所述）；

不同的存储引擎，其实主要是从 2 个大的层面来设计存储机制：

1，尽可能快的速度；

2，尽可能多的功能；

选择不同的存储引擎，就是上述性能和功能的“权衡”。

- 修改表
  - 修改表是指修改表的结构或特性。理论上创建一个表能做到的事情，修改表也能做到。修改表有二三十项修改项，包括增删改字段，等等。举例如下：
  - 添加字段：alter table 表名 add [column] 新字段名 字段类型 [字段属性列表]；
  - 修改字段（并可改名）：alter table 表名 change [column] 旧字段名 新字段名 新字段类型 [新字段属性列表]；
  - 修改字段（只改属性）：alter table 表名 modify [column] 字段名 新字段类型 [新字段属性列表]；
  - 修改字段名：没有单纯修改字段名这个功能！
  - 删除字段：alter table 表名 drop [column] 字段名；
  - 添加普通索引：alter table 表名 add key [索引名] (字段名1[，字段名2,...])；
  - 添加唯一索引(约束)：alter table 表名 add unique key (字段名1[，字段名2,...])；
  - 添加主键索引(约束)：alter table 表名 add primary key (字段名1[，字段名2,...])；
  - 添加外键索引(约束)：alter table 表名1 add foreign key (字段1，[，字段名2,...]) references 表名2(字段1，[，字段名2,...])；
  - 添加字段默认值(约束)：alter table 表名 alter [column] 字段名 set default 默认值；
  - 删除字段默认值(约束)：alter table 表名 alter [column] 字段名 drop default；
  - 删除主键：alter table 表名 drop primay key；#每一个表最多只能有一个主键
  - 删除外键：alter table 表名 drop foreign key 外键名；
  - 删除索引：alter table 表名 drop key 索引名；
  - 修改表名：alter table 表名 rename [to] 新表名；
  - 修改表选项：alter table 表名 选项名1=选项值1，选项名2=选项值2，...；
- 删除表: **drop table [if exists] 表名**；

其他表的相关语句：

- 显示当前数据库中的所有表: show tables；
- 显示某表的结构: desc 表名； 或：describe 表名；
- 显示某表的创建语句：show create table 表名；
- 重命名表：rename table 旧表名 to 新表名；
- 从已有表复制表结构：create table [if not exists] 新表名 like 原表名;

- 删除表: **drop table [if exists] 表名**；
- 其他表相关语句：
  - 显示所有表: show tables；
  - 显示某表的结构: desc 表名； 或：describe 表名；
  - 显示某表的创建语句：show create table 表名；
  - 重命名表：rename table 旧表名 to 新表名；
  - 从已有表复制表结构：create table [if not exists] 新表名 like 原表名;

  - 创建索引：create [unique | fulltext] index 索引名 on 表名(字段名1[，字段名2,...])。这里省略unique或fulltext，那就是普通索引。实际上此创建统内部映射为一条“alter table”的添加索引语句。
  - 删除索引：drop index 索引名 on 表名。实际上，此语句同样被映射为一条“alter table”的删除索引语句。

什么叫视图？

类比：什么叫做函数？

就是一段代码，我把它“封装”起来，并给一个名字，以后，要使用（执行）该段代码，就方便了：用该函数名就可以了。

视图：

就是一个 select 语句（通常比较复杂），我们给其一个名字（视图名），以后，要使用（执行）该 select 语句，就方便了：用该视图名就可以了。

语法形式：

create view 视图名 as select 语句；

举例：

create view v1 as

select id, f1, name, age, email, p_id, f3 from 表 1 where id > 7 and id < 100 or f1 < 1000 and age > 10

使用视图：

基本上，当做一个表用就了！

```
mysql> create  view v1 as
    -> select id, pid, sp_id, lastchanged from ipos_spkcb;
Query OK, 0 rows affected (0.00 sec)
```

语法形式：

create view 视图名 【（字段名 1，字段名 2，字段名 3，....）】 as select 语句；

删除视图：

drop view 【if exists】 视图名；