5. 表的约束

真正约束字段的是数据类型，但是数据类型约束很单一，需要有一些额外的约束，更好的**保证数据的合法性**，从业务逻辑角度保证数据的正确性。比如有一个字段是email，要求是唯一的。

表的约束很多，这里主要介绍如下几个：`null/not null,default, comment, zerofill, primary key, auto_increment, unique key`。

5.1 空属性

- 两个值：null（默认的）和not null(不为空)
- 数据库默认字段基本都是字段为空，但是实际开发时，尽可能保证字段不为空，因为数据为空没办法参与运算。

```
mysql> select null;
+------+
| NULL |
+------+
| NULL |
+------+
1 row in set (0.00 sec)

mysql> select 1+null;
+--------+
| 1+null |
+--------+
|   NULL |
+--------+
1 row in set (0.00 sec)
```

案例：

创建一个班级表，包含班级名和班级所在的教室，如果班级没有名字，你不知道你在哪个班级，如果教室名字可以为空，就不知道在哪上课。

```
mysql> create table myclass(
    -> class_name varchar(20) not null,
    -> class_room varchar(10) not null);
Query OK, 0 rows affected (0.02 sec)

mysql> desc myclass;
+------------+-------------+------+-----+---------+-------+
| Field      | Type        | Null | Key | Default | Extra |
+------------+-------------+------+-----+---------+-------+
| class_name | varchar(20) | NO   |     | NULL    |       |
| class_room | varchar(10) | NO   |     | NULL    |       |
+------------+-------------+------+-----+---------+-------+

//插入数据时,没有给教室数据插入失败:
```

```
mysql> insert into myclass(class_name) values('class1');
ERROR 1364 (HY000): Field 'class_room' doesn't have a default value
```

5.2 默认值

默认值：某一种数据会经常性的出现某个具体的值，可以在一开始就指定好，在需要真实数据的时候，用户可以选择性的使用默认值。

```
mysql> create table tt10 (
    -> name varchar(20) not null,
    -> age tinyint unsigned default 0,
    -> sex char(2) default '男'
    -> );
Query OK, 0 rows affected (0.00 sec)

mysql> desc tt10;
+-------+---------------------+------+-----+---------+-------+
| Field | Type                | Null | Key | Default | Extra |
+-------+---------------------+------+-----+---------+-------+
| name  | varchar(20)         | NO   |     | NULL    |       |
| age   | tinyint(3) unsigned | YES  |     | 0       |       |
| sex   | char(2)             | YES  |     | 男      |       |
+-------+---------------------+------+-----+---------+-------+
```

默认值的生效：数据在插入的时候不给该字段赋值，就使用默认值

```
mysql> insert into tt10(name) values('zhangsan');
Query OK, 1 row affected (0.00 sec)

mysql> select * from tt10;
+----------+------+------+
| name     | age  | sex  |
+----------+------+------+
| zhangsan |    0 | 男   |
+----------+------+------+
```

5.3 列描述

列描述：comment，没有实际含义，专门用来描述字段，会根据表创建语句保存，用来给程序员或DBA来进行了解。

```
mysql> create table tt12 (
    -> name varchar(20) not null comment '姓名',
    -> age tinyint unsigned default 0 comment '年龄',
    -> sex char(2) default '男' comment '性别'
    -> );
```

通过desc查看不到注释信息：

```
mysql> desc tt12;
+-------+---------------------+------+-----+---------+-------+
| Field | Type                | Null | Key | Default | Extra |
+-------+---------------------+------+-----+---------+-------+
| name  | varchar(20)         | NO   |     | NULL    |       |
| age   | tinyint(3) unsigned | YES  |     | 0       |       |
| sex   | char(2)             | YES  |     | 男      |       |
+-------+---------------------+------+-----+---------+-------+
```

通过show可以看到：

```
mysql> show create table tt12\G
*************************** 1. row ***************************
       Table: tt12
Create Table: CREATE TABLE `tt12` (
  `name` varchar(20) NOT NULL COMMENT '姓名',
  `age` tinyint(3) unsigned DEFAULT '0' COMMENT '年龄',
  `sex` char(2) DEFAULT '男' COMMENT '性别'
) ENGINE=MyISAM DEFAULT CHARSET=gbk
1 row in set (0.00 sec)
```

5.4 zerofill

刚开始学习数据库时，很多人对数字类型后面的长度很迷茫。通过show看看tt3表的建表语句：

```
mysql> show create table tt3\G
    ***************** 1. row *****************
           Table: tt3
    Create Table: CREATE TABLE `tt3` (
      `a` int(10) unsigned DEFAULT NULL,
      `b` int(10) unsigned DEFAULT NULL
    ) ENGINE=MyISAM DEFAULT CHARSET=gbk
    1 row in set (0.00 sec)
```

可以看到int(10),这个代表什么意思呢？整型不是4字节码？这个10又代表什么呢？其实没有zerofill这个属性，括号内的数字是毫无意义的。a和b列就是前面插入的数据，如下：

```
mysql> select * from tt3;
    +------+------+
    | a    | b    |
    +------+------+
    |    1 |    2 |
    +------+------+
```

但是对列添加了zerofill属性后，显示的结果就有所不同了。修改tt3表的属性：

```
mysql> alter table tt3 change a a int(5) unsigned zerofill;
```

对a列添加了zerofill属性，再进行查找，返回如下结果：

```
mysql> select * from tt3;
+-------+------+
| a     | b    |
+-------+------+
| 00001 |    2 |
+-------+------+
```

这次可以看到a的值由原来的1变成00001，这就是zerofill属性的作用，如果宽度小于设定的宽度（这里设置的是5），自动填充0。要注意的是，这只是最后显示的结果，在MySQL中实际存储的还是1。为什么是这样呢？我们可以用hex函数来证明。

```
mysql> select a, hex(a) from tt3;
+-------+--------+
| a     | hex(a) |
+-------+--------+
| 00001 | 1      |
+-------+--------+
```

可以看出数据库内部存储的还是1,00001只是设置了zerofill属性后的一种格式化输出而已。

5.5 主键

主键：primary key用来唯一的约束该字段里面的数据，不能重复，不能为空，一张表中最多只能有一个主键；主键所在的列通常是整数类型。

案例：

- 创建表的时候直接在字段上指定主键

```
mysql> create table tt13 (
->  id int unsigned primary key comment '学号不能为空',
->  name varchar(20) not null);
Query OK, 0 rows affected (0.00 sec)

mysql> desc tt13;
+-------+-----------------+------+-----+---------+-------+
| Field | Type            | Null | Key | Default | Extra |
+-------+-----------------+------+-----+---------+-------+
| id    | int(10) unsigned | NO   | PRI | NULL    |       | <= key 中 pri表示该字段是主键
| name  | varchar(20)     | NO   |     | NULL    |       |
+-------+-----------------+------+-----+---------+-------+
```

在创建表的时候，在所有字段之后，使用primary key(主键字段列表)来创建主键，如果有多个字段作为主键，可以使用复合主键。

```
mysql> create table tt14(
-> id int unsigned,
-> course char(10) comment '课程代码',
-> score tinyint unsigned default 60 comment '成绩',
-> primary key(id, course) -- id和course为复合主键
```

```
-> );
Query OK, 0 rows affected (0.01 sec)

mysql> desc tt14;
+--------+---------------------+------+-----+---------+-------+
| Field  | Type                | Null | Key | Default | Extra |
+--------+---------------------+------+-----+---------+-------+
| id     | int(10) unsigned    | NO   | PRI | 0       |       | <= 这两列合成主键
| course | char(10)            | NO   | PRI |         |       |
| score  | tinyint(3) unsigned | YES  |     | 60      |       |
+--------+---------------------+------+-----+---------+-------+
```

- 当表创建好以后，可以再次追加主键

```
alter table 表名 add primary key(字段列表)
```

- 主键约束：主键对应的字段中不能重复，一旦重复，操作失败。

```
mysql> insert into tt13 values(1, 'aaa');
Query OK, 1 row affected (0.00 sec)
```

```
mysql> insert into tt13 values(1, 'aaa');
ERROR 1062 (23000): Duplicate entry '1' for key 'PRIMARY'
```

- 删除主键

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

```
mysql> alter table tt13 drop primary key;
mysql> desc tt13;
+-------+------------------+------+-----+---------+-------+
| Field | Type             | Null | Key | Default | Extra |
+-------+------------------+------+-----+---------+-------+
| id    | int(10) unsigned | NO   |     | NULL    |       |
| name  | varchar(20)      | NO   |     | NULL    |       |
+-------+------------------+------+-----+---------+-------+
```

5.6 自增长

auto_increment：当对应的字段，不给值，会自动的被系统触发，系统会从当前字段中已经有的最大值+1操作，得到一个新的不同的值。通常和主键搭配使用，作为逻辑主键。

自增长的特点:

- 任何一个字段要做自增长，前提是本身是一个索引（key一栏有值）
- 自增长字段必须是整数
- 一张表最多只能有一个自增长

案例：

```
mysql> create table tt21(
    -> id int unsigned primary key auto_increment,
    -> name varchar(10) not null default ''
    -> );

mysql> insert into tt21(name) values('a');

mysql> insert into tt21(name) values('b');

mysql> select * from tt21;
+----+------+
| id | name |
+----+------+
|  1 | a    |
|  2 | b    |
+----+------+
```

在插入后获取上次插入的 AUTO_INCREMENT 的值（批量插入获取的是第一个值）

```
mysql > select last_insert_id();
+------------------+
| last_insert_id() |
+------------------+
|                1 |
+------------------+
```

索引：**

在关系数据库中，索引是一种单独的、物理的对数据库表中一列或多列的值进行排序的一种存储结构，它是某个表中一列或若干列值的集合和相应的指向表中物理标识这些值的数据页的逻辑指针清单。索引的作用相当于图书的目录，可以根据目录中的页码快速找到所需的内容。

索引提供指向存储在表的指定列中的数据值的指针，然后根据您指定的排序顺序对这些指针排序。数据库使用索引以找到特定值，然后顺指针找到包含该值的行。这样可以使对应于表的SQL语句执行得更快，可快速访问数据库表中的特定信息。

5.7 唯一键

一张表中有往往有很多字段需要唯一性，数据不能重复，但是一张表中只能有一个主键：唯一键就可以解决表中有多个字段需要唯一性约束的问题。

唯一键的本质和主键差不多，唯一键允许为空，而且可以多个为空，空字段不做唯一性比较。

案例：

```
mysql> create table student (
    -> id char(10) unique comment '学号，不能重复，但可以为空',
    -> name varchar(10)
    -> );
Query OK, 0 rows affected (0.01 sec)
```

```
mysql> insert into student(id, name) values('01', 'aaa');
Query OK, 1 row affected (0.00 sec)

mysql> insert into student(id, name) values('01', 'bbb'); --唯一约束不能重复
ERROR 1062 (23000): Duplicate entry '01' for key 'id'
mysql> insert into student(id, name) values(null, 'bbb'); -- 但可以为空
Query OK, 1 row affected (0.00 sec)

mysql> select * from student;
+------+------+
| id   | name |
+------+------+
| 01   | aaa  |
| NULL | bbb  |
+------+------+
```

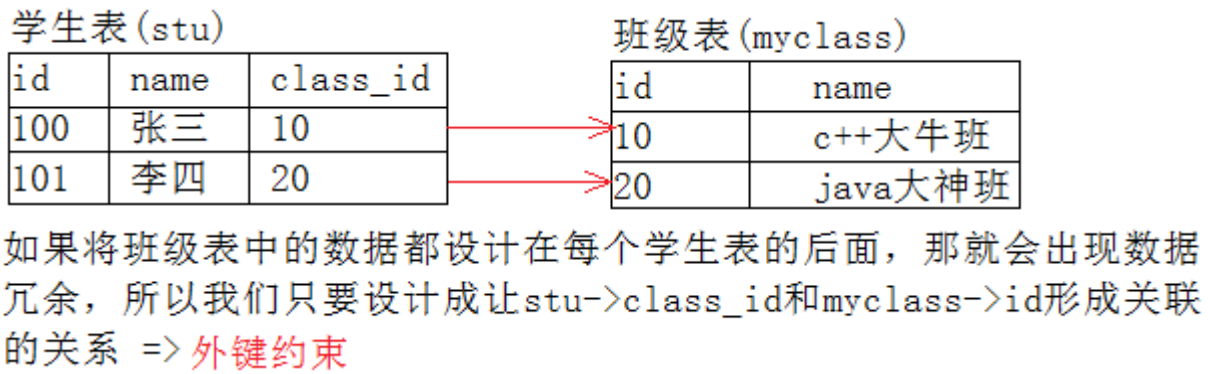
5.8 外键

外键用于定义主表和从表之间的关系：外键约束主要定义在从表上，主表则必须是有主键约束或unique约束。当定义外键后，要求外键列数据必须在主表的主键列存在或为null。

语法：

```
foreign key (字段名) references 主表(列)
```

案例：



对上面的示意图进行设计：

- 先创建主键表

```
create table myclass (
        id int primary key,
        name varchar(30) not null comment'班级名'
    );
```

- 再创建从表

```
create table stu (
        id int primary key,
        name varchar(30) not null comment '学生名',
        class_id int,
        foreign key (class_id) references myclass(id)
    );
```

- 正常插入数据

```
mysql> insert into myclass values(10, 'C++大牛班'),(20, 'java大神班');
Query OK, 2 rows affected (0.03 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

```
mysql> insert into stu values(100, '张三', 10),(101, '李四',20);
Query OK, 2 rows affected (0.01 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

- 插入一个班级号为30的学生，因为没有这个班级，所以插入不成功

```
mysql> insert into stu values(102, 'wangwu',30);
ERROR 1452 (23000): Cannot add or update a child row:
a foreign key constraint fails (mytest.stu, CONSTRAINT stu_ibfk_1
FOREIGN KEY (class_id) REFERENCES myclass (id))
```

- 插入班级id为null，比如来了一个学生，目前还没有分配班级

```
mysql> insert into stu values(102, 'wangwu', null);
```

5.9 综合案例

有一个商店的数据，记录客户及购物情况，有以下三个表组成：

- 商品goods(商品编号goods_id，商品名goods_name, 单价unitprice, 商品类别category, 供应商provider)
- 客户customer(客户号customer_id,姓名name,住址address,邮箱email,性别sex，身份证card_id)
- 购买purchase(购买订单号order_id,客户号customer_ id,商品号goods_ id,购买数量nums)

要求：

- 每个表的主外键
- 客户的姓名不能为空值
- 邮箱不能重复
- 客户的性别(男，女)

SQL:

```
-- 创建数据库
create database if not exists bit32mall
default character set utf8 ;

-- 选择数据库
```

```
use bit32mall;

-- 创建数据库表
-- 商品
create table if not exists goods
(
    goods_id  int primary key auto_increment comment '商品编号',
    goods_name varchar(32) not null comment '商品名称',
    unitprice  int  not null default 0  comment '单价，单位分',
    category  varchar(12) comment '商品分类',
    provider  varchar(64) not null comment '供应商名称'
);

-- 客户
create table if not exists customer
(
    customer_id  int primary key auto_increment comment '客户编号',
    name varchar(32) not null comment '客户姓名',
    address  varchar(256)  comment '客户地址',
    email  varchar(64)  unique key comment '电子邮箱',
    sex  enum('男','女') not null comment '性别',
    card_id char(18) unique key comment '身份证'
);

-- 购买
create table if not exists purchase
(
    order_id  int primary key auto_increment comment '订单号',
    customer_id int comment '客户编号',
    goods_id  int  comment '商品编号',
    nums  int  default 0 comment '购买数量',
    foreign key (customer_id) references customer(customer_id),
    foreign key (goods_id) references goods(goods_id)
);
```