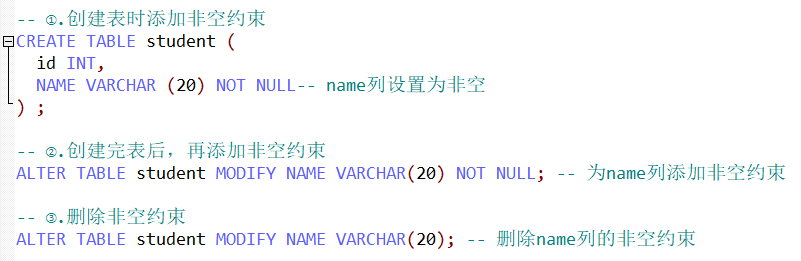
一.数据库表的约束

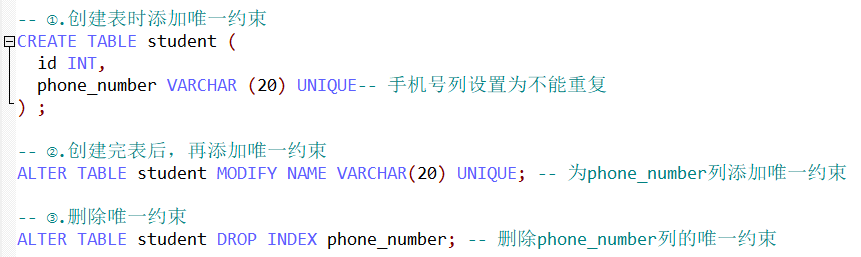
数据库表的约束，是对表中的列进行条件限制，确保不符合条件、不正确的数据无法插入到表中，从而达到对表中的数据进行限制的目的，保证了数据的正确性、有效性和完整性。一般在创建表的时候对表中的列加以约束，数据库表的约束一般有以下四种：

* 非空约束：NOT NULL
* 唯一约束：UNIQUE
* 主键约束：PRIMARY KEY
* 外键约束：FOREIGN KEY

1.非空约束 NOT NULL：某一列的值不能为 null



2.唯一约束 UNIQUE：某一列的值不能重复

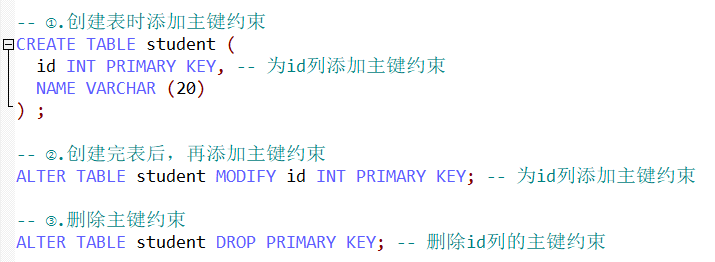


注意：唯一约束的列可以有 null 值，但是只能有一条记录为 null

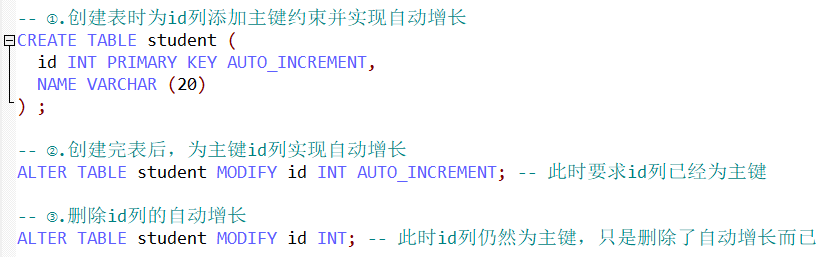
3.主键约束 PRIMARY KEY

如果表中某个列的数据，能唯一标识表中的每一条记录，那么就可以把该列作为表的主键，即：为该列添加上主键约束。所以表的主键就是用来唯一标识表中记录的，表中的主键具有如下特点：

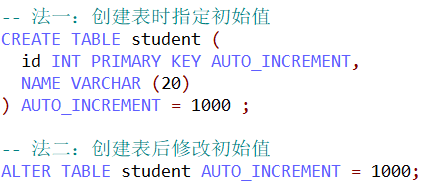
* 作为主键的列，必须满足：非空且唯一
* 主键的数目在一张表中，只能有一个，不能出现多个主键。但是主键可以是单列，也可以是多列(即使用两列来作为每条记录的唯一标识)
* 通常不使用业务字段作为主键，我们会单独给每张表设计一个名为 id 的字段，以此来作为表的主键



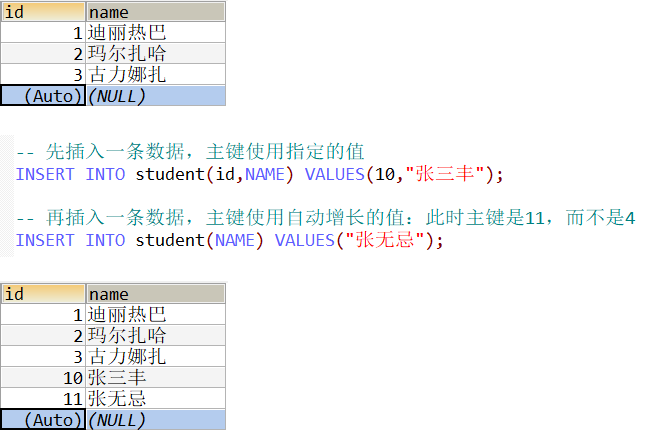
#.主键约束的自动增长 AUTO\_INCREMENT：主键的值如果让我们自己添加很有可能重复，通常希望在每次插入新记录时，数据库会自动生成主键字段的值，并依次自动增长。这时我们可以为主键添加自动增长来实现上述效果，并且自动增长只能用在主键上。



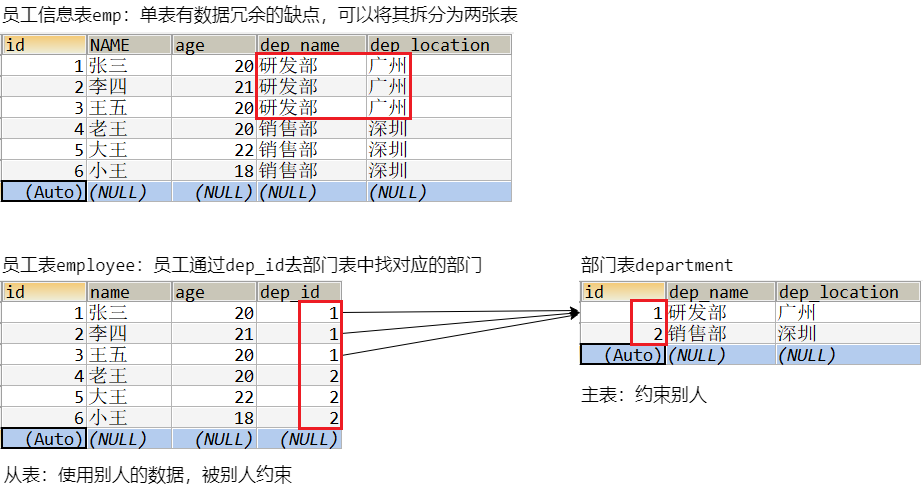
①.默认地主键自动增长的开始值是1，如果希望修改起始值，可采用以下方法



②.主键的自动增长是根据上一条记录的数值自动增长的



4.外键约束 FOREIGN KEY



当我们把员工信息表 emp 拆分为两张表后：在员工表 employee 的 dep\_id 列输入不存在的部门 id，数据依然可以添加；在部门表department 中删除一条数据，员工表并没有发生变化。即拆分后的两张表并没有任何联系，员工表的 dep\_id 列只是单纯地引用了部门表 id 列的数据，这两张表还是独立存在的。要想使两张表之间产生联系，就必须使员工表的 dep\_id 列与部门表的 id 列关联起来。此时就可以通过外键约束来实现这两个字段的关联：给员工表 dep\_id 列加上外键约束使其关联部门表的主键 id 列，此时 dep\_id 列就称为员工表的外键。这时员工表、部门表之间就产生了联系，二者并不是独立存在的个体了，这种联系就体现在：不能随意删除主表中的数据、不能在从表中添加不存在的外键数据，从而保证了数据的正确性。

所以当一张表的某个字段引用了另一张表的主键数据的时候：前一张表称为从表，后一张表称为主表。通过给这个字段加上外键约束使其关联主表的主键，使从表与主表之间产生联系，此时这个字段就称为这张从表的外键。



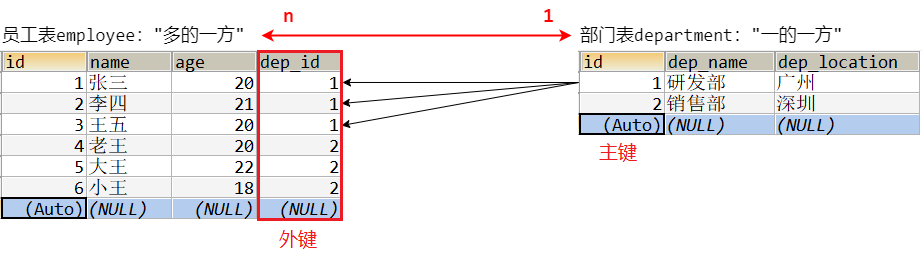
#.外键约束的级联操作：在给员工表加上外键约束后，此时再修改、删除部门表 id 列的数据，就会提示无法进行修改、删除。所以想要在修改、删除主表的主键时，同时更新、删除从表的外键值，就要给从表的外键列加上级联操作。级联操作包括：级联更新 ON UPDATE CASCADE、级联删除 ON DELETE CASCADE (二者可同时使用，也可单独使用)



二.数据库的设计

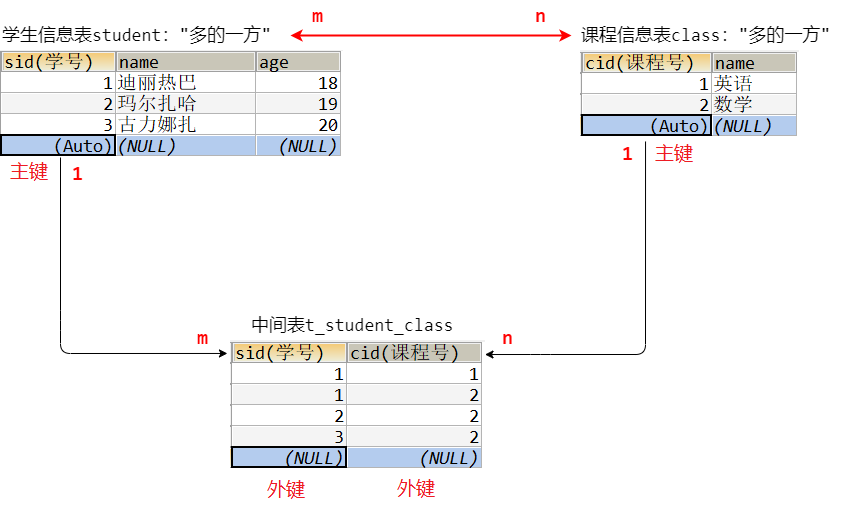
1.多表之间的关系

(1).一对多的关系：员工表和部门表的关系



一对多关系的建表原则：在"多的一方"建立外键，关联"一的一方"的主键

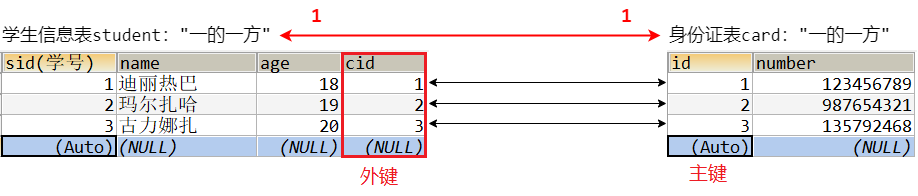
(2).多对多的关系：学生表和课程表的关系



多对多关系的建表原则：

多对多关系在建表时需要借助第三张中间表，用来保存这两张表之间的多对多对应关系。中间表至少包含两个字段，这两个字段均作为中间表的外键，分别关联两张表的主键，此时这两张表和中间表之间又形成了一对多的关系。(联合主键：这两个字段可以联合起来作为中间表的联合主键)

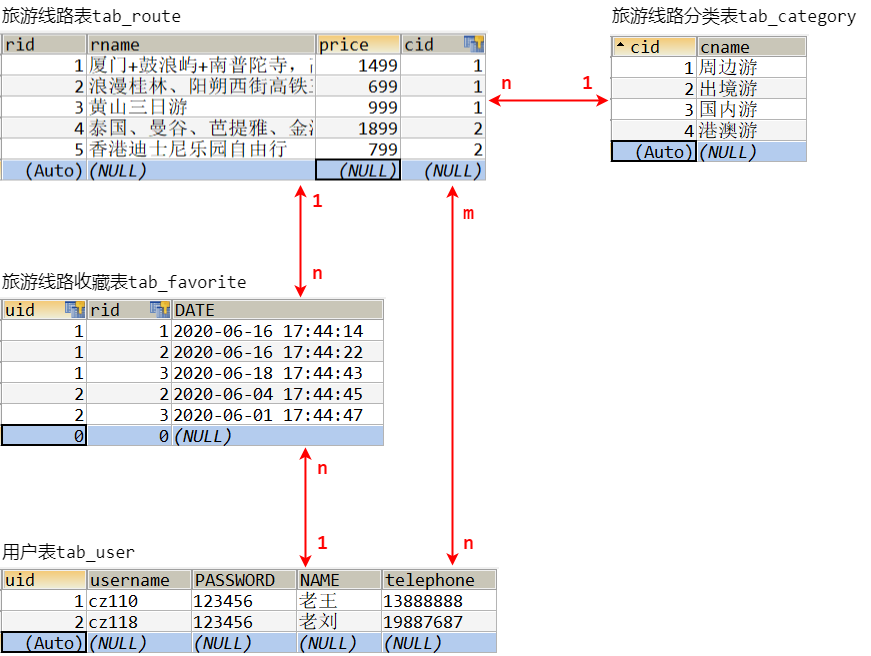
(3).一对一的关系：学生表和身份证表的关系



一对多关系的建表原则：可以在任意一方建立唯一外键，关联另一方的主键。(一对一的关系一般创建成一张表)

#.数据库设计的案例：

简单设计旅游网的数据库，旅游线路分类表、旅游线路表之间是一对多的关系，即：一个旅游线路分类下可以有多个旅游线路。用户表、旅游线路表是多对多的关系，即：一个用户可以收藏多个旅游线路，一个旅游线路也可以被多个用户所收藏，此时需要借助旅游线路收藏表来保存二者之间的多对多对应关系。(表的创建语句见：[参考\旅游网的建表语句.sql](http://参考/旅游网的建表语句.sql))



2.数据库设计的范式

        \* 概念：设计数据库时，需要遵循的一些规范。要遵循后边的范式要求，必须先遵循前边的所有范式要求

设计关系数据库时，遵从不同的规范要求，设计出合理的关系型数据库，这些不同的规范要求被称为不同的范式，各种范式呈递次规范，越高的范式数据库冗余越小。  
                目前关系数据库有六种范式：第一范式（1NF）、第二范式（2NF）、第三范式（3NF）、巴斯-科德范式（BCNF）、第四范式(4NF）和第五范式（5NF，又称完美范式）。

\* 分类：  
                1. 第一范式（1NF）：每一列都是不可分割的原子数据项  
                2. 第二范式（2NF）：在1NF的基础上，非码属性必须完全依赖于码（在1NF基础上消除非主属性对主码的部分函数依赖）  
                        \* 几个概念：  
                                1. 函数依赖：A-->B,如果通过A属性(属性组)的值，可以确定唯一B属性的值。则称B依赖于A  
                                        例如：学号-->姓名。 （学号，课程名称） --> 分数  
                                2. 完全函数依赖：A-->B， 如果A是一个属性组，则B属性值得确定需要依赖于A属性组中所有的属性值。  
                                        例如：（学号，课程名称） --> 分数  
                                3. 部分函数依赖：A-->B， 如果A是一个属性组，则B属性值得确定只需要依赖于A属性组中某一些值即可。  
                                        例如：（学号，课程名称） -- > 姓名  
                                4. 传递函数依赖：A-->B, B -- >C . 如果通过A属性(属性组)的值，可以确定唯一B属性的值，在通过B属性（属性组）的值可以确定唯一C属性的值，则称 C 传递函数依赖于A  
                                        例如：学号-->系名，系名-->系主任  
                                5. 码：如果在一张表中，一个属性或属性组，被其他所有属性所完全依赖，则称这个属性(属性组)为该表的码  
                                        例如：该表中码为：（学号，课程名称）  
                                        \* 主属性：码属性组中的所有属性  
                                        \* 非主属性：除过码属性组的属性  
                                          
                3. 第三范式（3NF）：在2NF基础上，任何非主属性不依赖于其它非主属性（在2NF基础上消除传递依赖）

三.事务