DBMS中的关系模型和Codd规则介绍

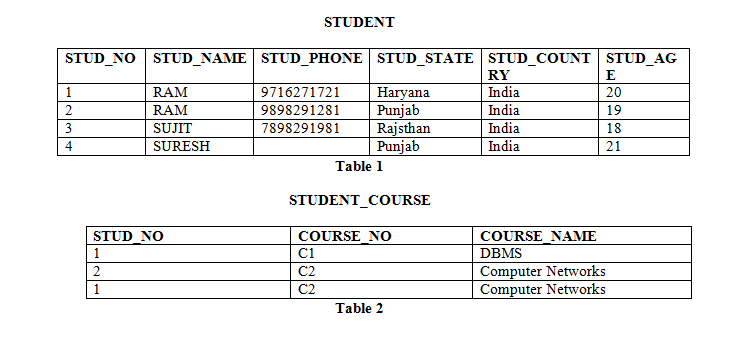
**关系模式：**模式表示关系的结构。例如; STUDENT关系的关系模式可以表示为：  
STUDENT（STUD\_NO，STUD\_NAME，STUD\_PHONE，STUD\_STATE，STUD\_COUNTRY，STUD\_AGE）

**关系实例：**关系在特定时间实例上存在的一组值称为关系实例，如表1和表2所示。

**属性：**每个关系都是根据某些性质定义的，每个性质都称为属性。例如，STUD\_NO，STUD\_NAME等是关系STUDENT的属性。

**属性的域：**属性可以在关系中采用的可能值称为其域。例如，STUD\_AGE的域可以是18到40。  
**元组：**关系的每一行称为元组。例如; 下面给出的学生关系有4个元组。

**NULL值：**某些元组的某些属性的值可能是未知的，缺失的或未定义的，它们由NULL表示。关系中的两个NULL值被认为彼此不同。  
表1和表2表示具有两个关系STUDENT和STUDENT\_COURSE的关系模型。

[](https://media.geeksforgeeks.org/wp-content/uploads/image7.png)

**Codd规则**

E.F. Codd提出了Codd规则，应该通过关系模型来满足。

1. **基础规则：**对于任何宣传为或声称是关系数据库管理系统的系统，该系统必须能够完全通过其关系功能来管理数据库。
2. **信息规则：**关系模型中存储的数据必须是表中某些单元格的值。
3. **保证的访问规则：**每个数据元素都必须可以通过表名称，其主键和要确定其值的属性名称进行访问。
4. **NULL值的系统处理：**数据库中的NULL值必须仅与丢失，未知或不适用的值相对应。
5. **活动在线目录：**数据库的结构必须存储在一个在线目录中，授权用户可以查询该目录。
6. **全面的数据子语言规则：**数据库应通过支持定义，操作和事务管理操作的语言进行访问。
7. **视图更新规则：**为各种目的而创建的不同视图应由系统自动更新。
8. **高级别的插入，更新和删除规则：**关系模型应该在每个关系级别上支持插入，删除，更新等操作。另外，应支持诸如并集，交点和减号之类的设置操作。
9. **物理数据独立性：**对表物理位置的任何修改都不应在应用程序级别上强制进行修改。
10. **逻辑数据独立性：**对表的逻辑或概念模式所做的任何修改均不应在应用程序级别上强制进行修改。例如，将两个表合并到一个表中不应影响应用程序难以访问它。
11. **完整性独立性：**在数据库级别修改的完整性约束不应强制在应用程序级别进行修改。
12. **分发独立性：**终端用户不应看到各个位置的数据分发。
13. **非颠覆规则：**对数据的低级别访问不应能够绕过完整性规则来更改数据。