



北京交通大学  
BEIJING JIAOTONG UNIVERSITY



# 软件系统分析与设计 System Analysis & Design

M210007B [03]

Haiming Liu

Monday, May 30, 2022

# 课程计划

1. 软件工程基础/软件生命周期
2. 软件过程模型
3. 软件项目管理
4. 系统分析方法与问题定义
5. 需求分析与需求获取
6. 用例建模与用例描述
7. 数据模型

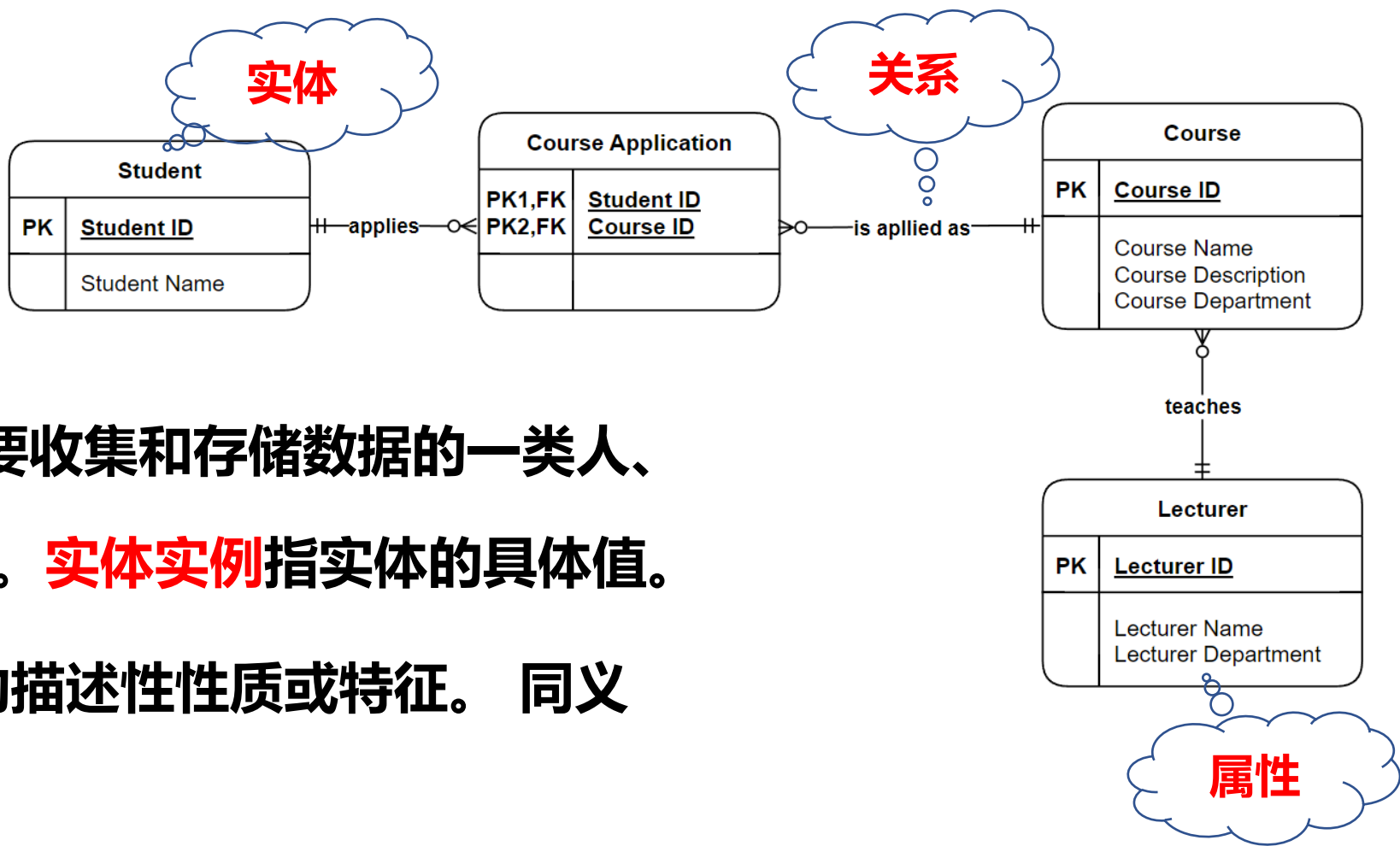
# 数据建模

## Data Modeling

1. 一种组织和记录系统数据的技术，有时也称为数据库建模
2. 实体关系图(Entity Relationship Diagram, **ERD**) 是一种利用符号记法按照数据描述的实体和关系来刻画数据的数据模型

# 举个栗子

## An ERD Example

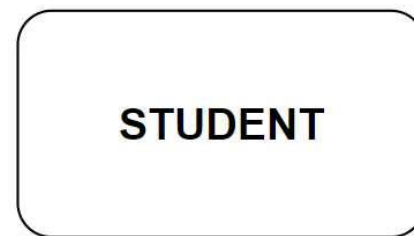


- **实体(Entity):** 是我们需要收集和存储数据的一类人、地点、对象、事件或概念。 **实体实例**指实体的具体值。
- **属性(Attribute):** 实体的描述性性质或特征。 同义词包括要素、性质和域。
- **关系(Relationship):** 存在于一个或多个实体之间的业务联系。

# 实体(Entity)

实体的分类包括：

- 人：“客户”、“部门”、“学生”等。可以表示个人、小组或组织。
- 地点：“销售区域”、“建筑物”、“校园”等。
- 对象：“图书”、“工具”、“飞机”等。
- 事件：“应用”、“取消”、“分类”、“预订”等。
- 概念：“时间段”、“资格”、“课程”等。



一个实体

实体实例是实体的具体值：

例：实体STUDENT可以有以下实例  
May、Mary、Joe、Martin等

# 属性(Attribute)

**STUDENT**

一个实体

**STUDENT**

Name

- Last Name
- First Name
- Middle Name

Address

- Street Address
- City
- Province
- Country

Postal Code

Phone Number

Data of Birth

Gender

Race

Major

Grade Point Average

属性和组合属性

# 属性(Attribute)

域 (Domain) 是属性的一个参数，定义了这个属性可以取的合法值

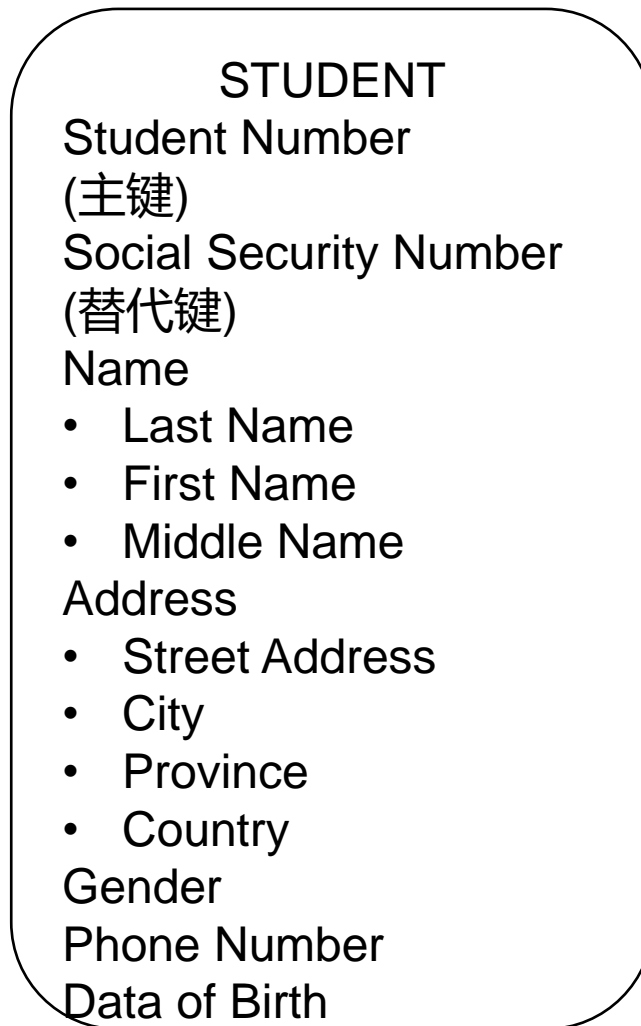
逻辑数据类型	逻辑业务含义	域(Domain)	例子
NUMBER	任何数、实数和整数	对于整数，指定范围 对于实数，指定范围和精度	{10~99} {1.0~799.999}
TEXT	一个字符串，包括数字。当数字包含在TEXT属性中时，意味着我们不希望进行哪些数字的算术或比较运算	TEXT（属性的最大长度） <b>实际值通常是无限的，但是用户可以指定某个较小的范围</b>	TEXT(30)
MEMO	同TEXT一样，但具有不确定的大小。某些业务系统要求能够附加潜在的长注解 信息到一个给定的数据库记录中	不可应用	不可应用
DATE	任何格式的日期	MMDDYYYY格式的变量	MMDDYYYY MMYYYY
TIME	任何格式的时间	对于AM/PM时间: HHMMT 或 对于军队时间: HHMM	HHMMT HHMM
YES/NO	只能取这两个值中的一个值的属性	{YES, NO}	{ON, OFF}
VALUE SET	一个有限值集合。在大多数情况下应该建立一个编码方案	{值1, 值2, ..., 值n}或 {标识代码及含义的表}	{M=Male F=Female}
IMAGE	任何图形或图像	不可应用	不可应用



# 属性(Attribute)

**标识符 (Domain) 是根据一个或多个属性的数据值唯一地标识每个实例。每个实体必须具有一个标识符或键**

- 候选键(Candidate Key): 是一组可以作为一个实体的主键的键, 也称为候选标识符。
- 主键(Primary Key): 最常用来唯一地确定一个实体实例的候选键。
- 替代键(Alternate Key): 没有被选中作为主键的任何候选键。也称为次键。
- 子集准则(Subsetting Criteria): 是一个属性其有限的取值范围把所有的实体实例分成了有用的子集。也称为反向条目。

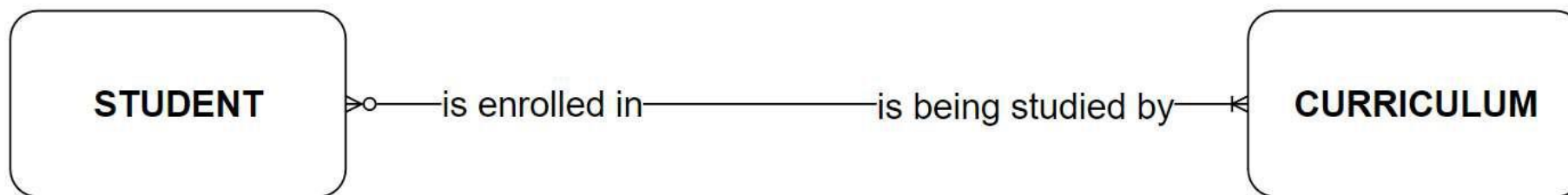




# 关系(Relationship)

实体STUDENT和CURRICULUM之间的关系：

- 一个学生可以选修一门或多门课程
- 一门课程正被零个、一个或者多个学生学习

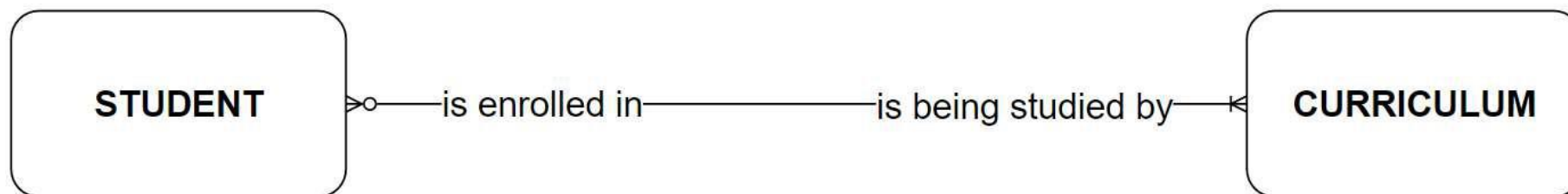


数据关系复杂度

- 基数(Cardinality)
- 度数(Degree)

# 关系(Relationship)

基数(Cardinality): 一个实体相对于另一个实体的某个具体值的最小和最大具体值数量



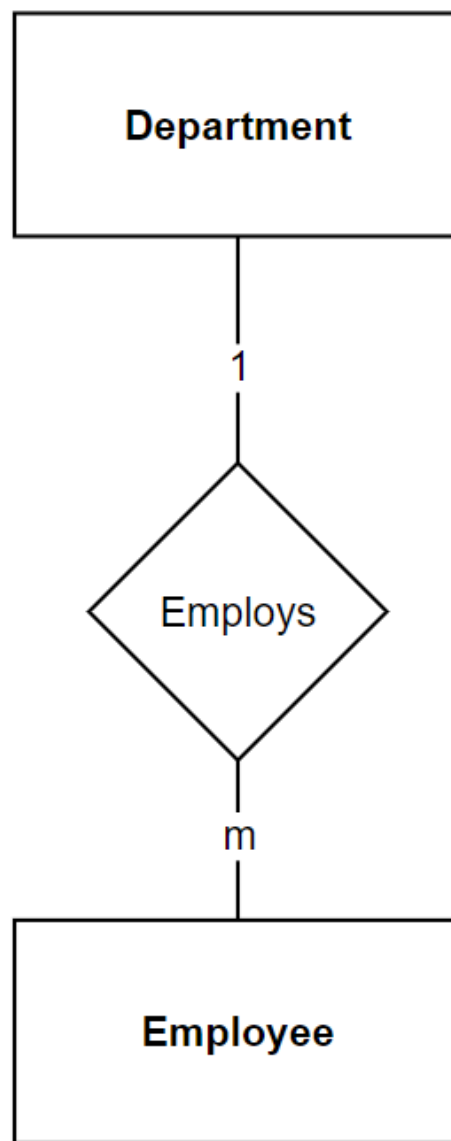
对应课程的每个实例都必然存在学生的一个实例吗? **NO!**

对应学生的每个实例都必然存在课程的一个实例吗? **YES!**

对应学生的每个实例存在多少个课程实例? **MANY!**

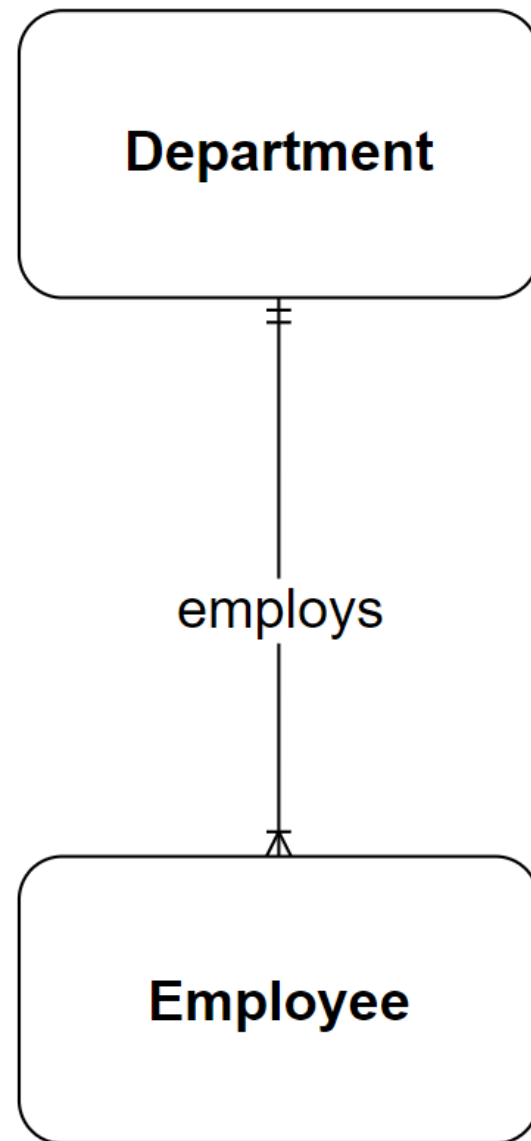
对应课程的每个实例存在多少个学生实例? **MANY!**

# 实体关系图符号 Entity Relationship Diagram Notations



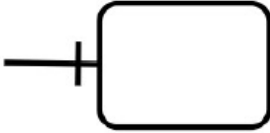
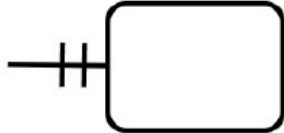
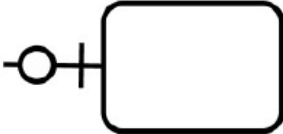
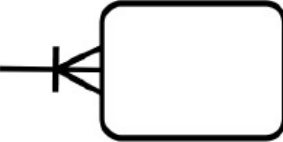
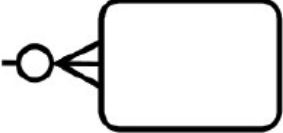
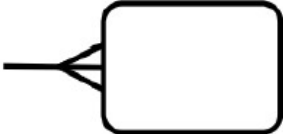
Chen

VS



Crow's Foot

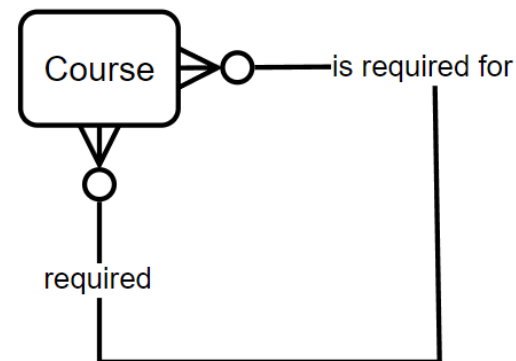
# 关系(Relationship)

基数含义	最小实例数	最大实例数	图形化符号
正好一个(一个且仅一个)	1	1	 或 
零个或一个	0	1	
一个或多个	1	多个(>1)	
零个、一个或多个	0	多个(>1)	
大于一个	>1	>1	

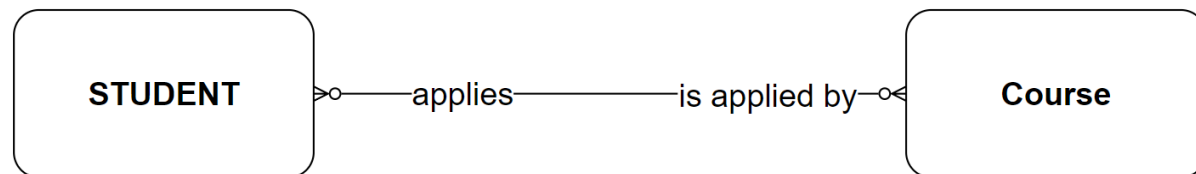
# 关系(Relationship)

度数(Degree): 关系的度数是参与那个关系的实体数量

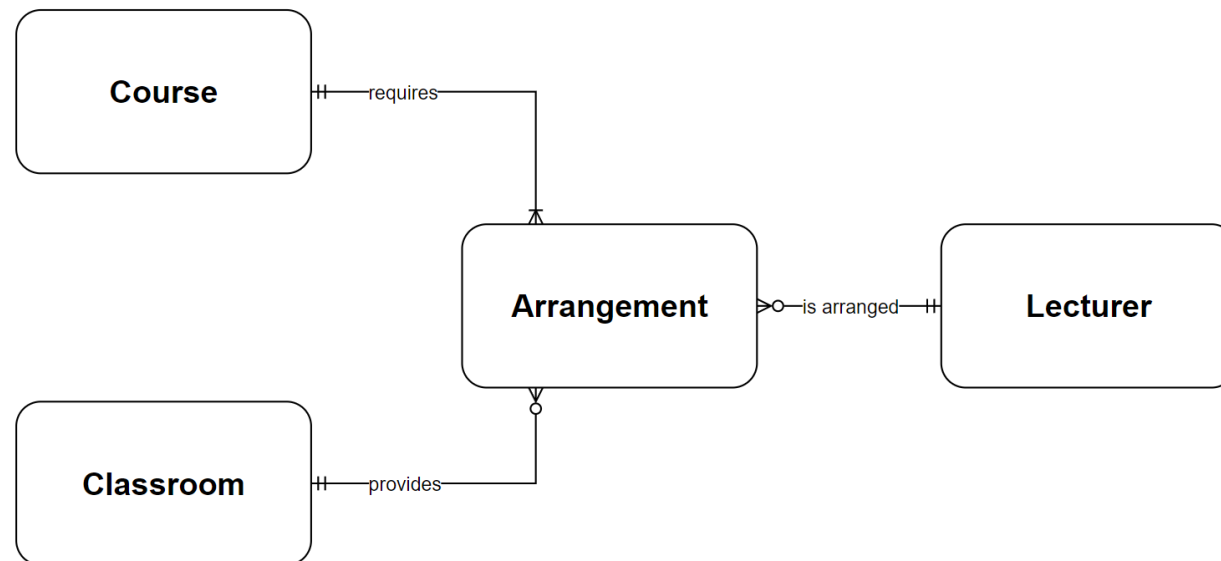
- 递归关系(Recursive Relationship): degree = 1



- 二维关系(Binary Relationship): degree = 2

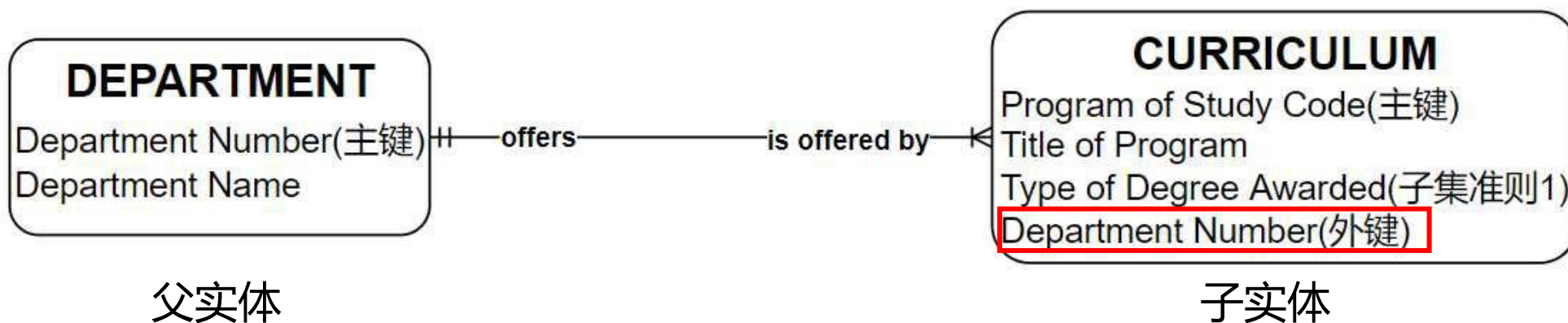


- 三维关系(Ternary Relationship): degree = 3



# 关系(Relationship)

外键 (Foreign Keys) : 是一个实体的主键, 它被复制到另一个实体以确定一个关系实例

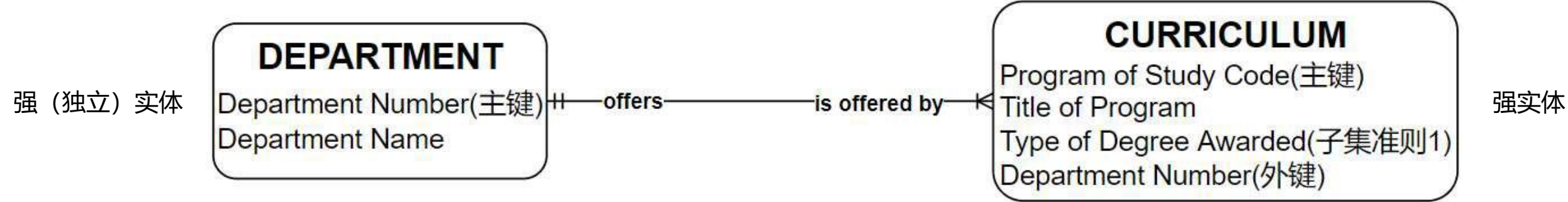


**父实体**: 将一个或多个属性复制给另一个实体的数据实体, 称为孩子。在一对多关系中, 父级是 “一” 侧的实体。

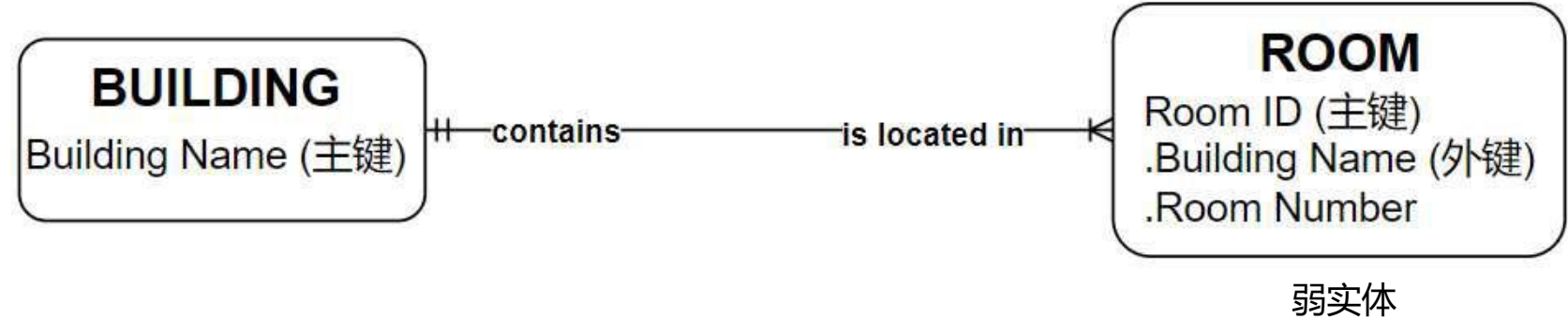
**子实体**: 从另一个实体派生一个或多个属性的数据实体, 称为父母。在一对多关系中, 孩子是 “多” 方的实体。

# 关系(Relationship)

非确定性关系(Nonidentifying Relationship): 每个参与关系的实体都有各自的独立主键的关系, 是一个实体的多个实例与另一个实体的多个实例相关联的关系。

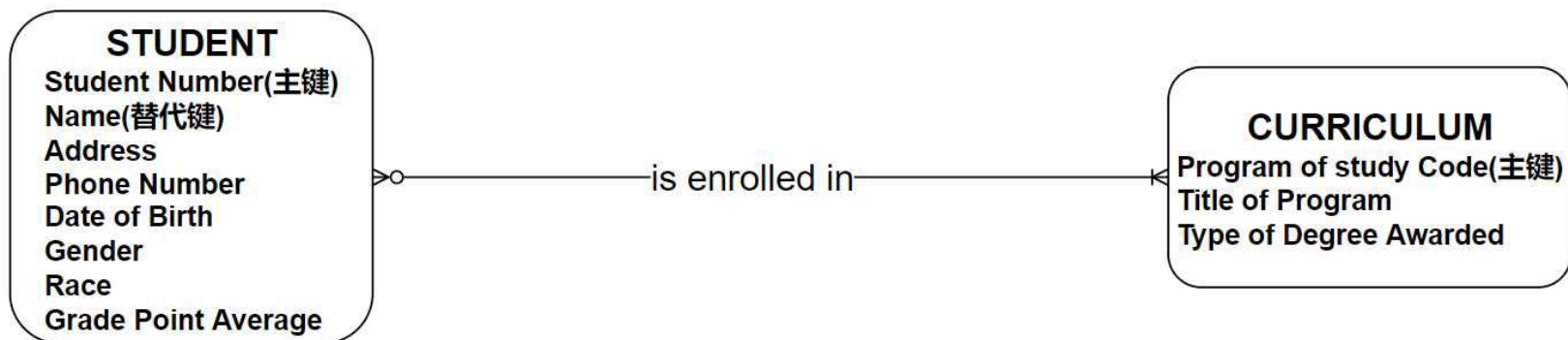


确定性关系(Identifying Relationship): 父实体的键也是子实体的主键的一部分的关系





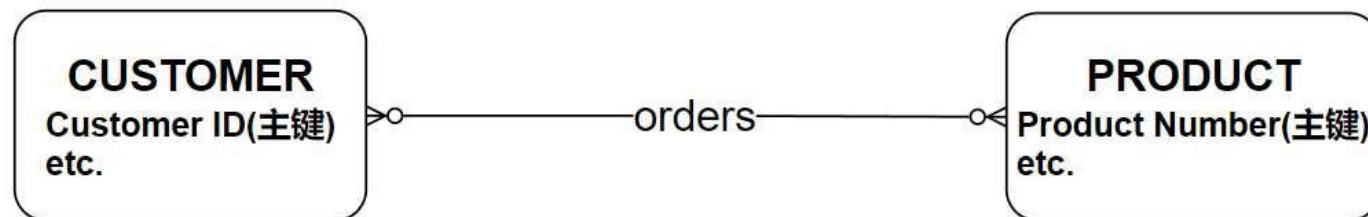
# 解决非确定性关系



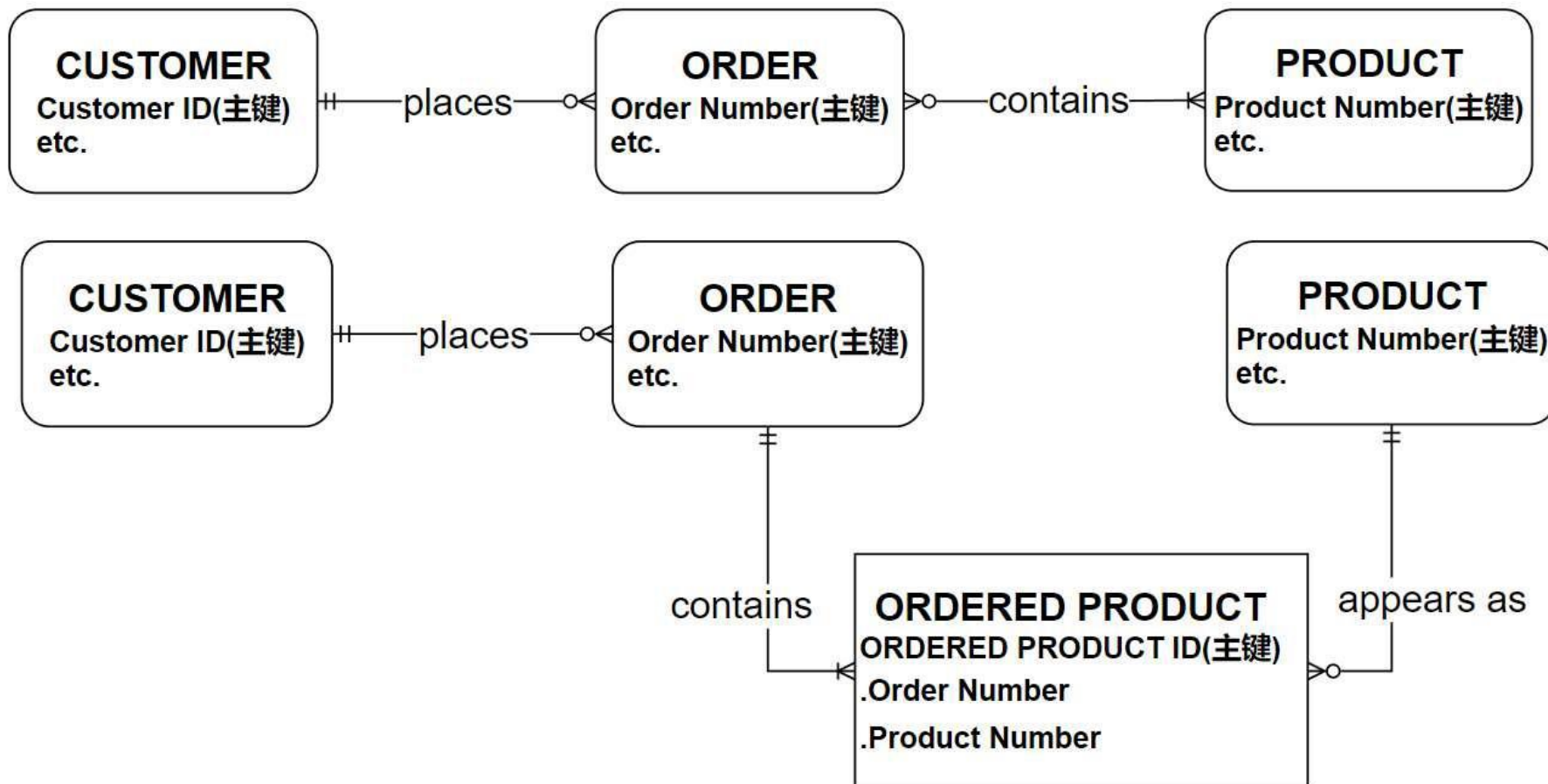
通过关联实体解决，所有关联实体都是弱实体



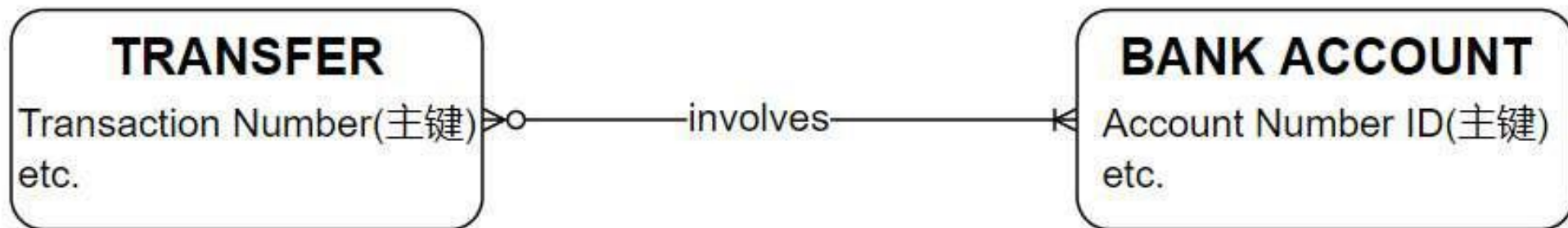
# 解决非确定性关系



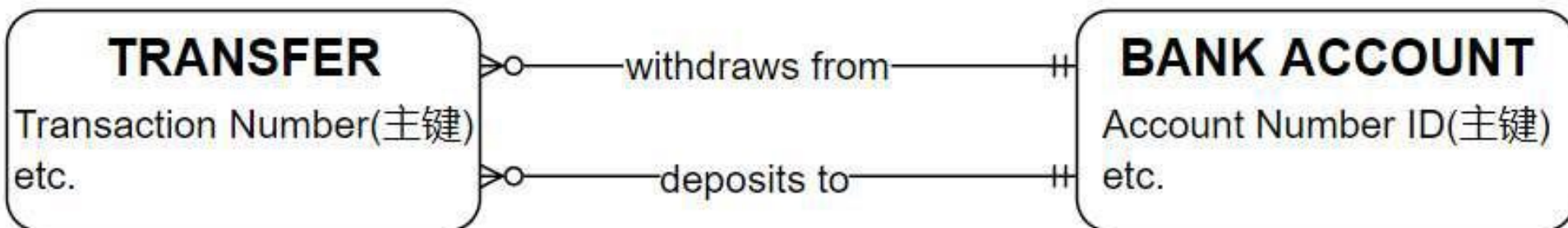
通过识别基本业务实体来解决



# 解决非确定性关系



通过引入独立的关系进行分解



# 案例分析——学生信息管理系统

1. 需要存储有关STUDENT、COURSE、LECTURE、ACADEMIC

2. STUDENT ID的值唯一地标识一个学生。当然COURSE ID的值唯一地标识一位讲师，一门COURSE只由一位LECTURE讲授

3. 对于一个STUDENT，我们需要知道STUDENT的名字。对于一门COURSE名称描述和COURSE DEPARTMENT。对于LECTURELECTURE DEPARTMENT

4. STUDENT申请0、1或多个COURSE APPLICATIONS

5. 一门COURSE由一个或多个COURSE APPLICATIONS

6. COURSE APPLICATION标识单个STUDENT申请单个COURSE COURSE ID标识课程。它们共同确定了一个且只有一个COURSE

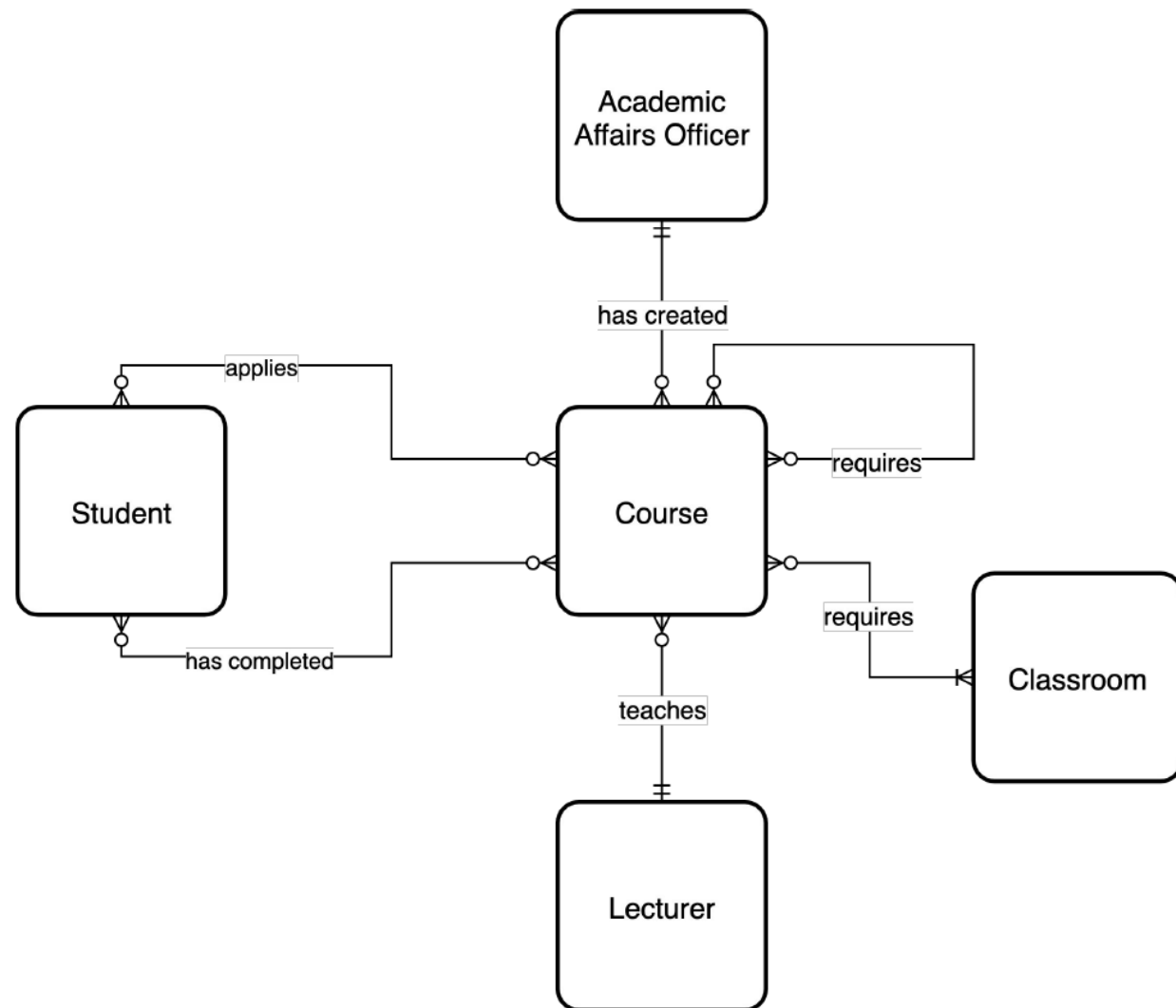
7. LECTURE教授零门、一门或多门COURSES

8. 等等
- | Use Case ID            | SAD-001  |       |      |                        |                  |                    |                  |  |                    |
|------------------------|--|-------|------|------------------------|------------------|--------------------|------------------|--|--------------------|
| 用例名                    | 申请课程   |       |      |                        |                  |                    |                  |  |                    |
| 描述                     | 这个用例描述了学生通过系统申请课程的事件。学生从开放课程列表中选择一门课程。学生可以获得课程信息、剩余可选数以及是否完成了预先要求的课程。一旦学生提交申请，创建一个申请并等待批准。   |       |      |                        |                  |                    |                  |  |                    |
| 优先级                    | 高  |       |      |                        |                  |                    |                  |  |                    |
| 主要参与者                  | 学生   |       |      |                        |                  |                    |                  |  |                    |
| 其他参与者                  | 学生   |       |      |                        |                  |                    |                  |  |                    |
| 前置条件                   | 学生必须登录系统进行选课   |       |      |                        |                  |                    |                  |  |                    |
| 触发器                    | 当学生选择申请课程时启动用例   |       |      |                        |                  |                    |                  |  |                    |
| 典型事件过程                 | <table><tr><th>参与者动作</th><th>系统响应</th></tr><tr><td>第1步：学生从打开的课程列表中选择一门课程。</td><td>第2步：系统响应，显示课程信息。</td></tr><tr><td>第3步：学生浏览课程信息并选择申请。</td><td>第4步：系统为学生生成课程申请。</td></tr><tr><td></td><td>第5步：系统向学生显示申请是否成功。</td></tr></table> | 参与者动作 | 系统响应 | 第1步：学生从打开的课程列表中选择一门课程。 | 第2步：系统响应，显示课程信息。 | 第3步：学生浏览课程信息并选择申请。 | 第4步：系统为学生生成课程申请。 |  | 第5步：系统向学生显示申请是否成功。 |
| 参与者动作                  | 系统响应   |       |      |                        |                  |                    |                  |  |                    |
| 第1步：学生从打开的课程列表中选择一门课程。 | 第2步：系统响应，显示课程信息。   |       |      |                        |                  |                    |                  |  |                    |
| 第3步：学生浏览课程信息并选择申请。     | 第4步：系统为学生生成课程申请。   |       |      |                        |                  |                    |                  |  |                    |
|                        | 第5步：系统向学生显示申请是否成功。   |       |      |                        |                  |                    |                  |  |                    |
| 替代事件过程                 | 替代第3步：课程没有空席或没有完成所有预修课程。学生无法申请课程。终止用例。   |       |      |                        |                  |                    |                  |  |                    |
| 结论                     | 当学生收到生成应用程序的确认信息时，此用例就结束了。   |       |      |                        |                  |                    |                  |  |                    |
| 后置条件                   | 申请已生成并等待批准。  |       |      |                        |                  |                    |                  |  |                    |
| 业务规则                   | 学生不是休学状态。  |       |      |                        |                  |                    |                  |  |                    |
| 实现约束和说明                | - 用例必须24 * 7小时在课程申请期间提供给学生。<br>- 频率：预计此用例每天将执行 10,000 次。 它应该支持多达 50 名学生的并发量。   |       |      |                        |                  |                    |                  |  |                    |
| 假设                     | - 开放课程列表中的课程可供申请<br>- 学生可以在获得批准之前取消申请  |       |      |                        |                  |                    |                  |  |                    |
| 开放问题                   | N/A  |       |      |                        |                  |                    |                  |  |                    |

# 上下文数据模型

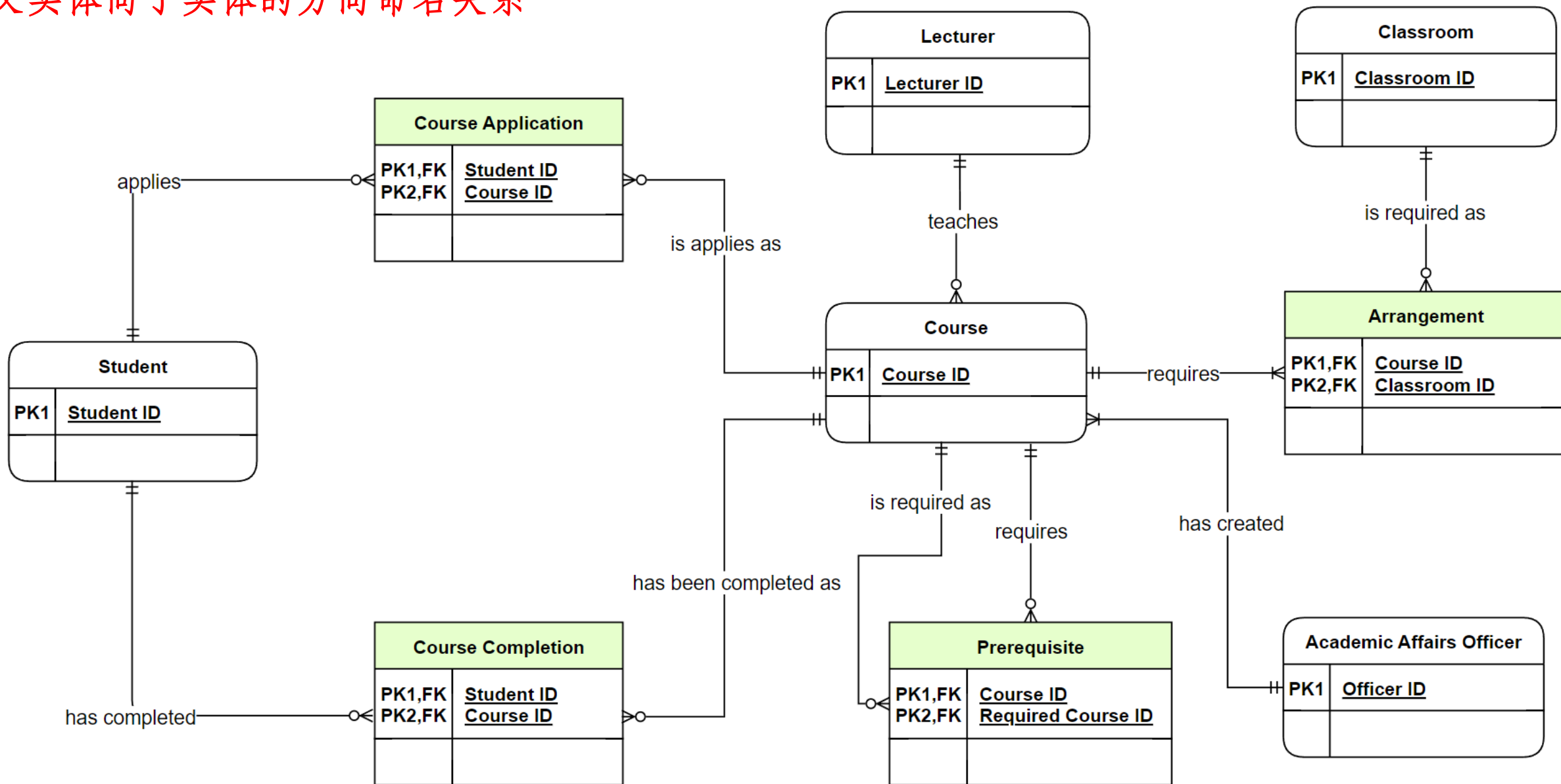
## Context Data Model

- 上下文数据模应该包括基本业务实体以及它们之间的自然关系。
- 关系使用动词短语命名，和实体名称共同组成一个简单的业务语句



# 基于键的数据模型 Key-Based Data Model

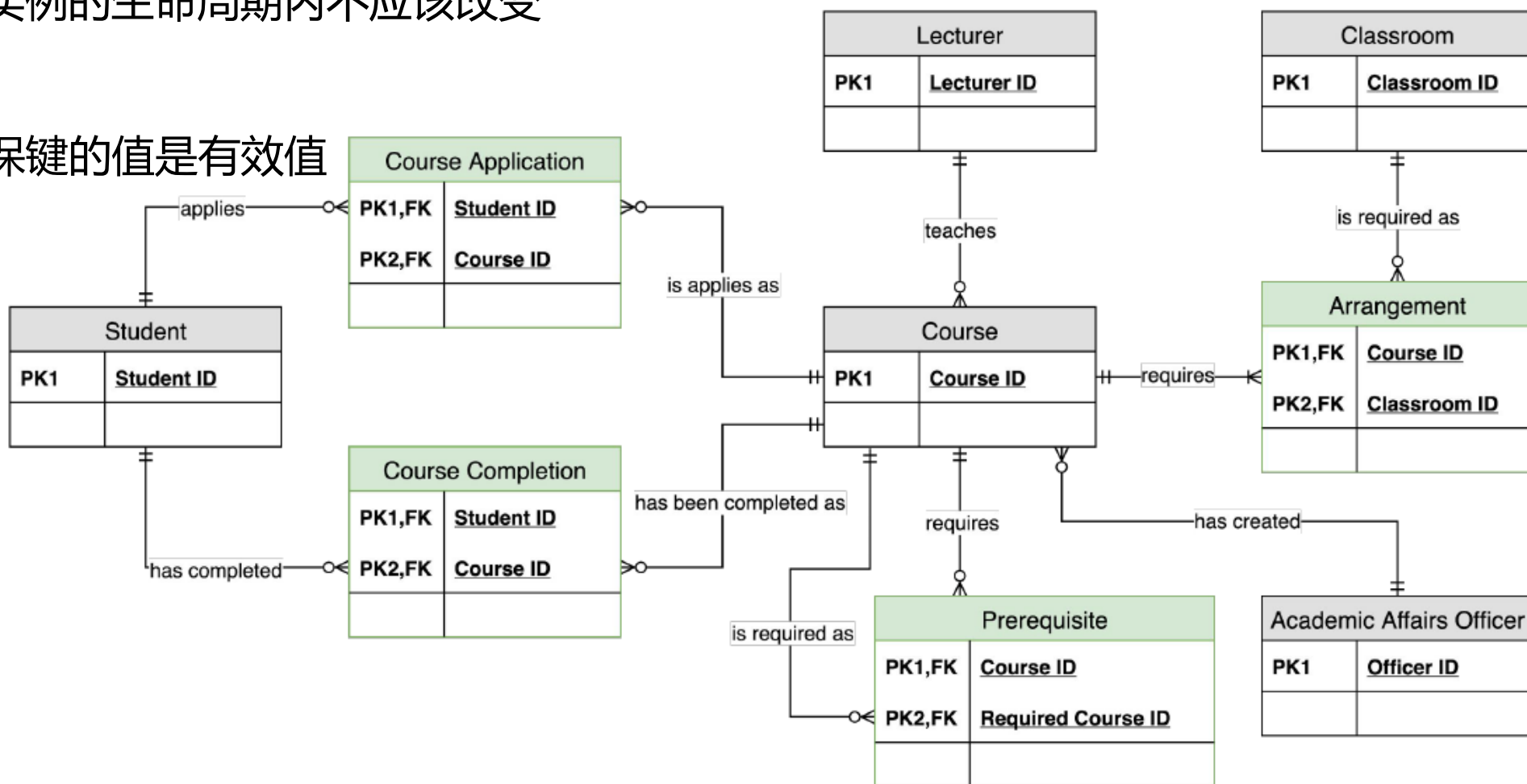
从父实体向子实体的方向命名关系



# 基于键的数据模型

## Key-Based Data Model

- 从父实体向子实体的方向命名关系
- 确定每个实体的主键和替代键，遵循以下原则：
  - 键的值在每个实体实例的生命周期内不应该改变
  - 键的值不能为空
  - 必须进行控制以确保键的值是有效值





# 泛化层次体系 Generalized Hierarchies

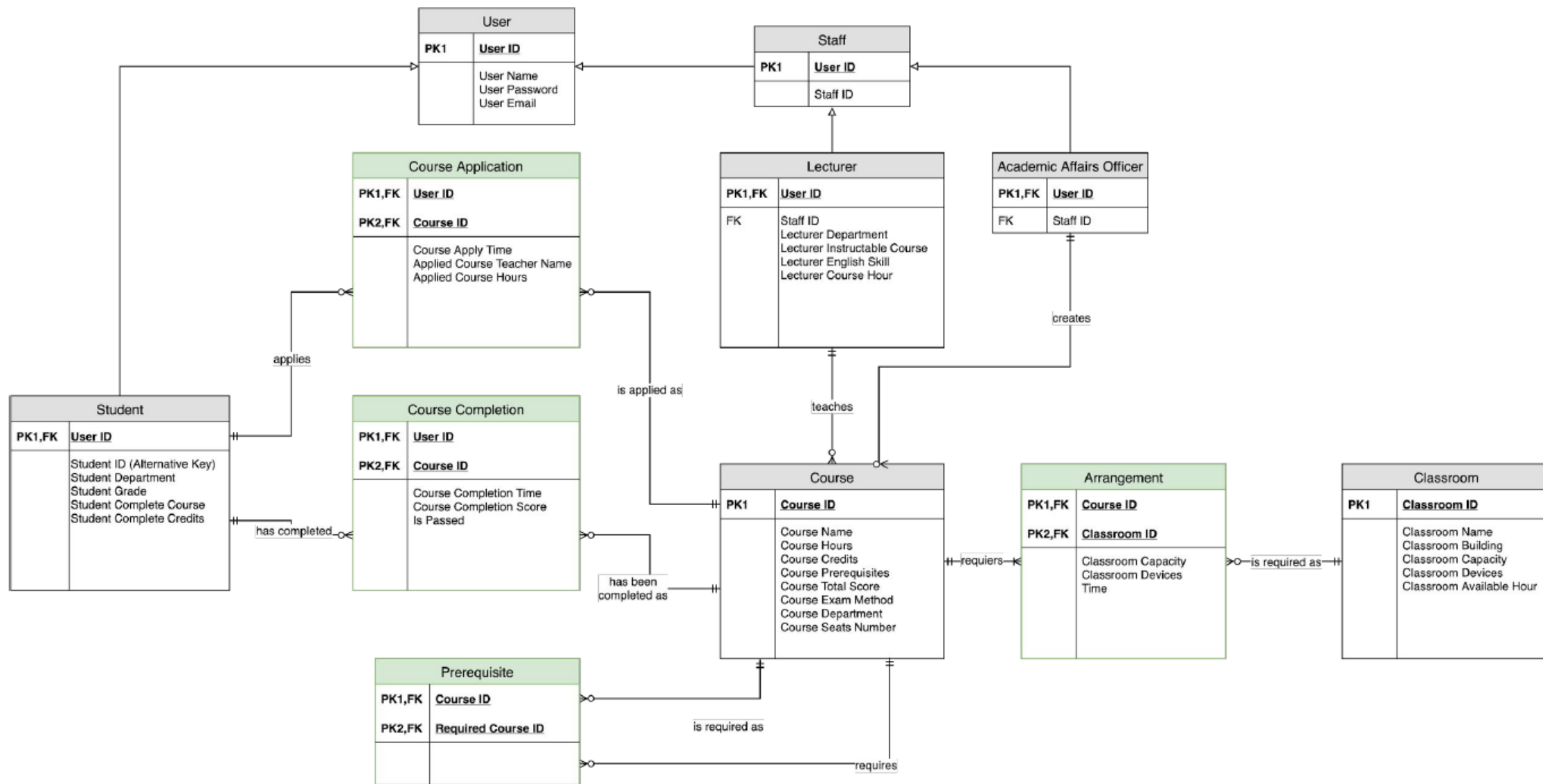
对于两个实体，若有几个属性是公共的，可以提取这些公共属性到一个实体超类中

- ① 子类(subtype)继承超类(supertype)的键
- ② 超类是一个实体，其实例存储了一个或多个实体子类的公共属性，也称抽象类或父类



# 具有完整属性的数据模型

## The Fully Attributed Data Model



# 如何构造数据模型

## How to Construct Data Models

1. 获取实体
2. 构造上下文数据模型，确立项目范围
3. 绘制基于键的数据模型
4. 泛化层次体系
5. 构造一个具有完整属性的数据模型

# 分析数据模型

## Analyzing the Data Model

### 好的数据模型的标准

- 简单的(simple)
- 无冗余的
- 灵活的，对未来的需求具有可适应性

### 规范化（改进数据模型）

- 第一范式(1NF)：实体的所有属性值都是原子值不可再分。
- 第二范式(2NF)：实体的所有非主键属性的值都依赖于主键。
- 第三范式(3NF)：实体的非主键属性的值不依赖于任何其他非主键属性

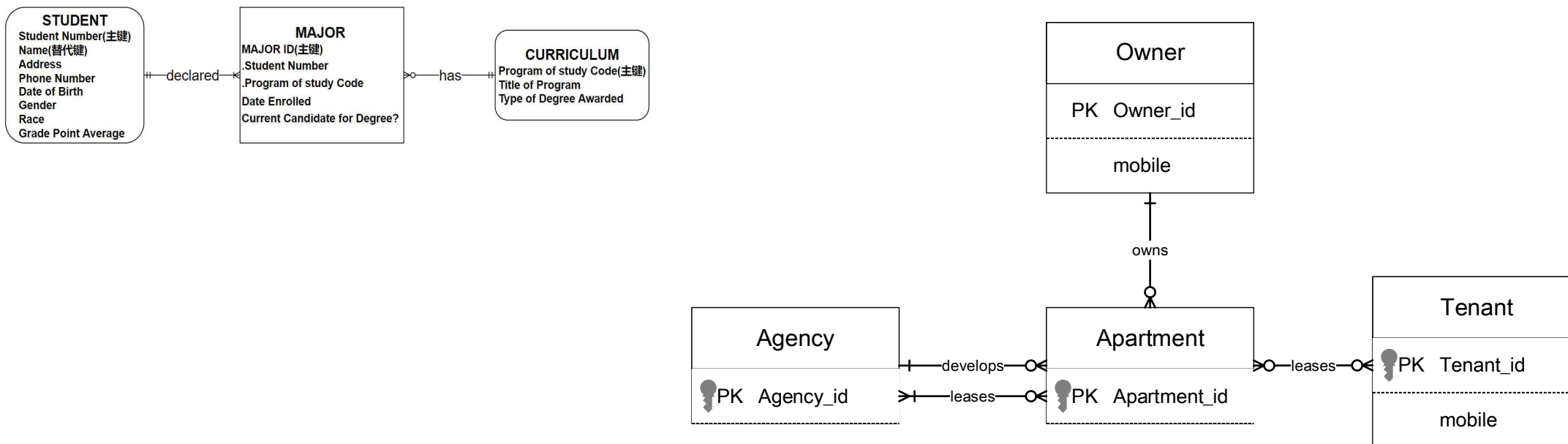
# 练习 Practice

【腾讯文档】解决所有非特定关系

<https://docs.qq.com/form/page/DSGlqQm92ZkF3d3FQ>

- 公寓业主授权公寓中介出租其公寓。
- 租户（租用公寓居住的客户）通过中介租赁公寓。
- 代理、承租人和业主签订租赁合同，约定租赁的起止日期。
- 不需要考虑业主可以出售公寓或业主可以将公寓委托给其他机构等情况。

根据给出的ER图和文字描述，构建基于键的数据模型，解决所有非特定关系。



# 本节内容

## Readings

### 《系统分析与设计方法》

- 第8章 数据建模与分析
- 第14章 数据库设计

- 关键词：数据建模；实体关系图；非特定关系；上下文数据模型；基于键的数据模型