



数据库原理及应用

主讲：陈安龙

chenanlong@uestc.edu.cn

电子科技大学

一、数据库设计与性能优化问题探讨

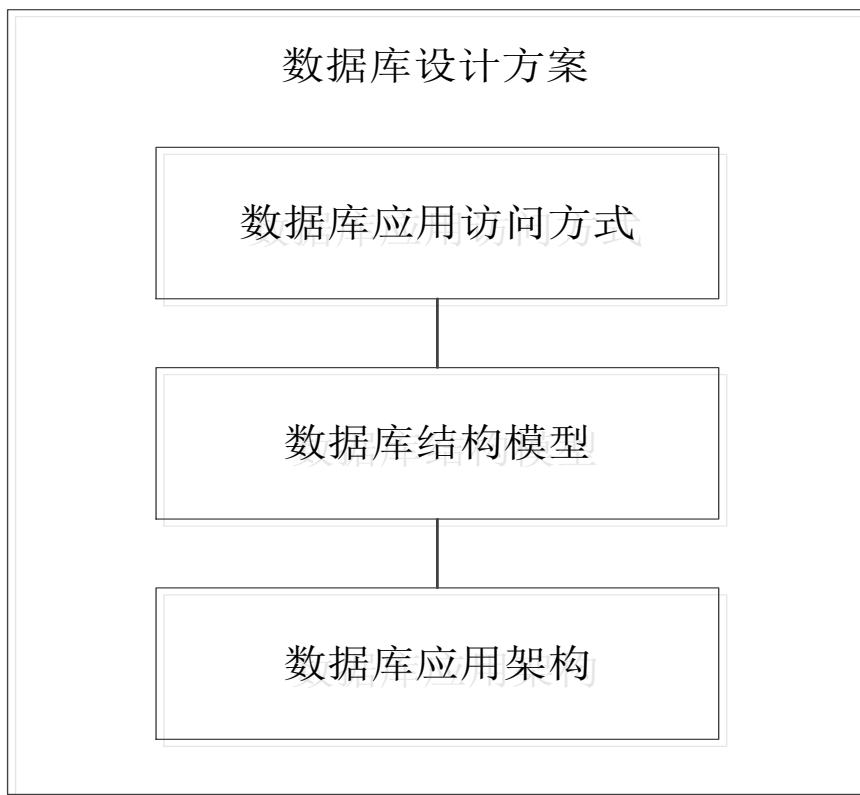
- 1、数据库设计如何影响数据库系统性能？
- 2、在数据库设计哪个阶段应考虑系统性能问题？
- 3、数据库哪些设计要素影响数据库系统性能？
- 4、数据库性能优化设计有哪些方法？

二、工程案例的数据库设计性能优化探讨

- 1、如何从数据库系统架构层面优化设计图书借阅管理系统？
- 2、如何从数据库逻辑层面优化设计图书借阅管理系统？
- 3、如何从数据库物理层面优化设计图书借阅管理系统？

一、数据库设计方案

数据库设计是数据库应用系统开发的重要内容。在实现数据库之前，必须有明确的设计方案。



1. 数据库应用架构设计

在不同应用需求场景中，数据库的应用架构方式是不同的。数据库应用架构可分为单用户结构、集中式结构、客户 / 服务器结构和分布式结构。

2. 数据库结构设计

数据库结构设计一般分为概念层、逻辑层、物理层设计，设计模型分别为概念数据模型、逻辑数据模型和物理数据模型。

3. 数据库应用访问方式设计

数据库应用对数据库访问可以有多种方式，如直接本地接口连接访问、基于标准接口连接访问、基于数据访问层框架连接访问。

数据库结构设计模型

概念数据模型（Concept Data Model, CDM）是一种面向用户的系统数据模型，它用来描述现实世界的系统概念化数据结构。使数据库设计人员在系统设计的初始阶段，摆脱计算机系统及DBMS的具体技术问题，集中精力分析业务数据以及数据之间的联系等，描述系统的数据对象及其组成关系。

逻辑数据模型（Logic Data Model, LDM）是在概念数据模型基础上，从系统设计角度描述系统的数据对象组成及其关联结构，并考虑这些数据对象符合数据库对象的逻辑表示。

物理数据模型（Physical Data Model, PDM）是在逻辑数据模型基础上，针对具体DBMS所设计的数据模型。它用于描述系统数据模型在具体DBMS中的数据对象组织、存储方式、索引方式、访问路径等实现信息。

数据库建模设计过程

数据需求分析

需求分析阶段

现实世界抽象为**概念模型**，由数据库设计人员和用户共同完成

系统概念数据建模

概念设计阶段

概念模型抽象为**逻辑模型**，由数据库设计人员完成

系统逻辑数据建模

逻辑设计阶段

逻辑模型抽象为**物理模型**，并转化为数据定义语句，即创建数据库对象的SQL语句。

系统物理数据建模

物理设计阶段

由程序员或DBA用SQL语句**创建数据库对象**。

数据库实现

系统实现阶段

数据库建模设计

1) 数据分析阶段

- 从现实业务获取数据表单、报表、查询、业务规则、数据更新的说明
- 分析系统的数据特征、数据类型、数据取值约束
- 描述系统的数据关系、数据处理要求
- 建立系统的数据字典

2) 数据库设计阶段

- 数据库模型结构设计（概念数据模型、逻辑数据模型、物理数据模型）
- 数据库索引、视图、查询设计
- 数据库表约束设计
- 数据库触发器、存储过程设计

3) 数据库实现阶段

- 数据库创建
- 数据模型物理实现

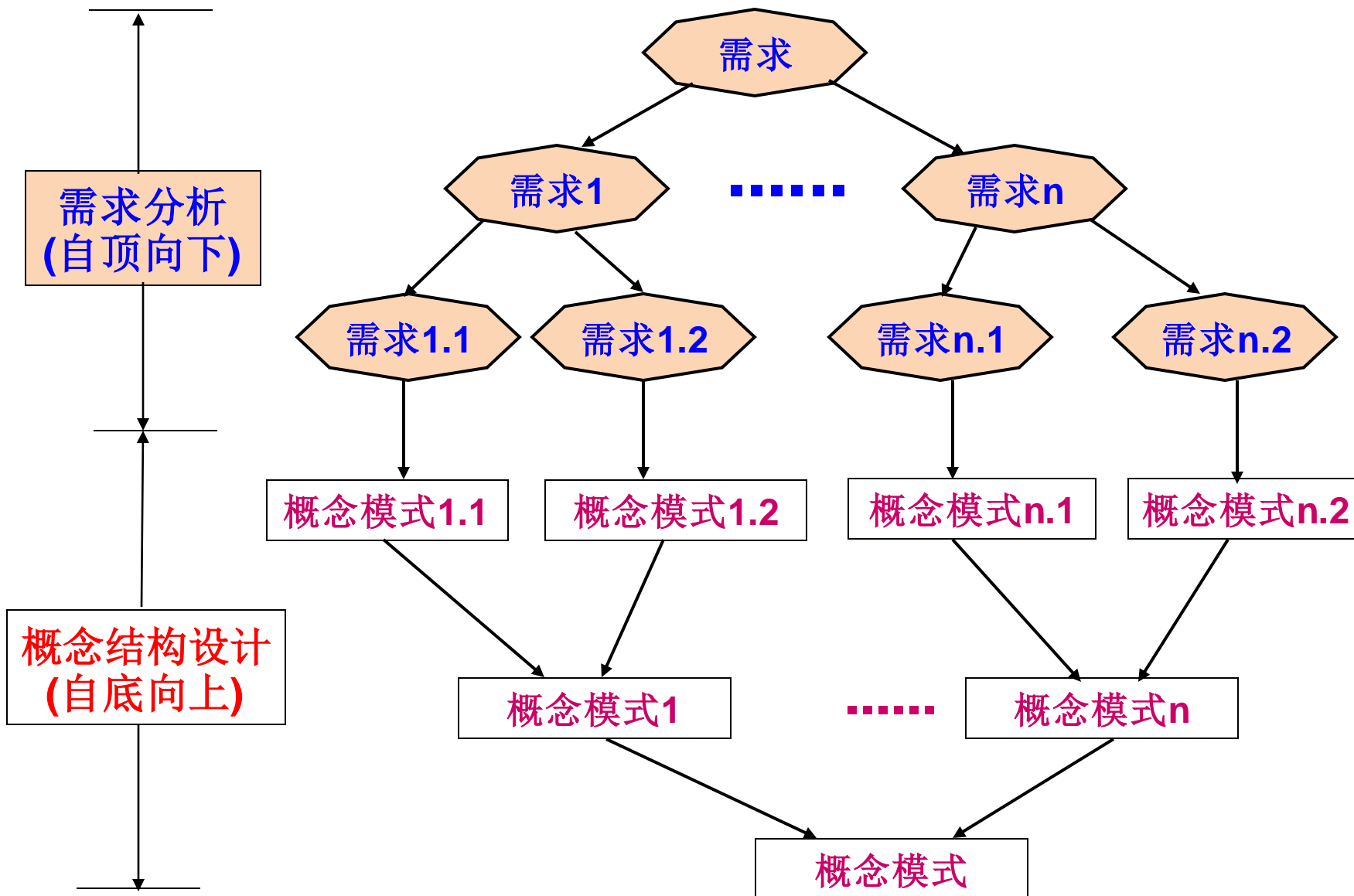
4) 数据库测试阶段

- 数据库数据上线
- 数据库系统测试

2. 设计策略

- 自底向上设计
- 自顶向下设计
- 自内向外设计
- 混合策略设计

自顶向下分析需求与自底向上设计概念结构



概念模型-----ER模型

➤ 概念模型的用途

- 概念模型用于信息世界的建模
- 是现实世界到机器世界的一个中间层次
- 是数据库设计的有力工具
- 数据库设计人员和用户之间进行交流的语言

➤ 对概念模型的基本要求

- 较强的语义表达能力
- 能够方便、直接地表达应用中的各种语义知识
- 简单、清晰、易于用户理解

ER模型及基本概念

➤ 实体(Entity)

- **概念：** 一个现实世界中有别于其它对象的对象。
- **注意：** 可以是具体的、也可以是抽象的。
- **示例：** 某某学生、某某老师、某门课程

ER模型及基本概念

➤ 属性 (Attribute)

- **概念：** 实体的特征或性质，即实体用若干属性来描述。
- **示例：** 学生的学号、姓名、生日、年龄、性别、住址等；
课程的课程号、课程名、学时、学分、开课学院等。
- **分类(按结构)：** 简单属性(不可再分)复合属性和子属性。
- **示例：** 复合—姓名(现用名、曾用名、英文名)；住址(省、市、区、街道、门牌号、邮政编码)。

ER模型及基本概念

➤ 域(Domain)

- **概念：** 属性的取值范围。
- **按域的取值分：** 单值、多值、导出和空值(NULL)等属性。
- **示例：** 多值—学位值(学士、硕士、博士)； 导出—生日导出年龄。

ER模型及基本概念

- **键**：用于唯一标识集合中的每个实体的一组属性。
示例：学生的**学号**；课程的**课程号**；选课的**学号及课程号**
- **键的分类**(按属性个数)：简单键、复合键。
- **候选键**(Candidate Key)：有多种选择作为键的属性或属性集，且属性集的任何属性都不可缺少，如缺少任意属性，就不能成为键。
- **主键**(Primary Key)：当存在多个候选键时，需选定一个作为实体的主键。是描述实体的唯一标识。**示例**：学生的身份证号、学号等。

ER模型及基本概念

➤ 实体型 (Entity Type)

- **概念:** 用实体名及其属性名集合来抽象和刻画同类实体称为实体型
- **示例:** 学生 (学号、姓名、生日、年龄、性别、住址)

➤ 实体集 (Entity Set)

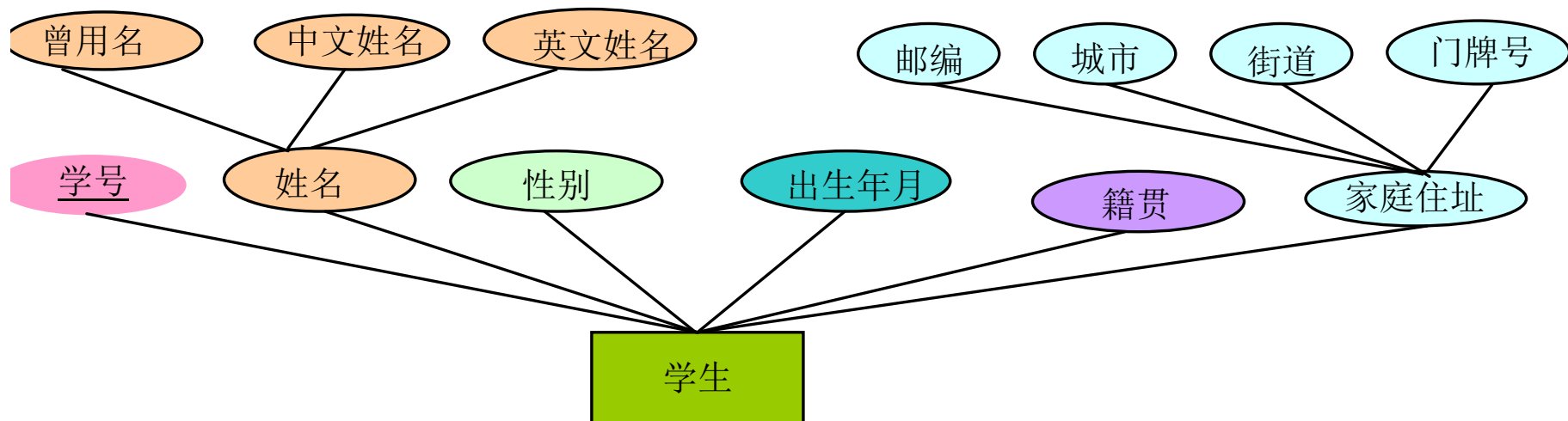
- **概念:** 同一类型实体的集合称为实体集
- **示例:** 一个学校所有学生构成的集合, 称为学生实体集

注意: 在不影响理解的情况下, 可以将实体、实体型、实体集都简称为实体

实体和属性的表示

ER模型可图形表示，传统的表示方式为：实体型用长方形；属性用椭圆；主键用下划线。

➤ 示例



主流数据库建模工具Power Designer

Power Designer是一种面向软件开发生命周期的建模工具，它提供软件需求模型、业务流程模型、数据库模型、面向对象模型、自定义模型的开发支持。

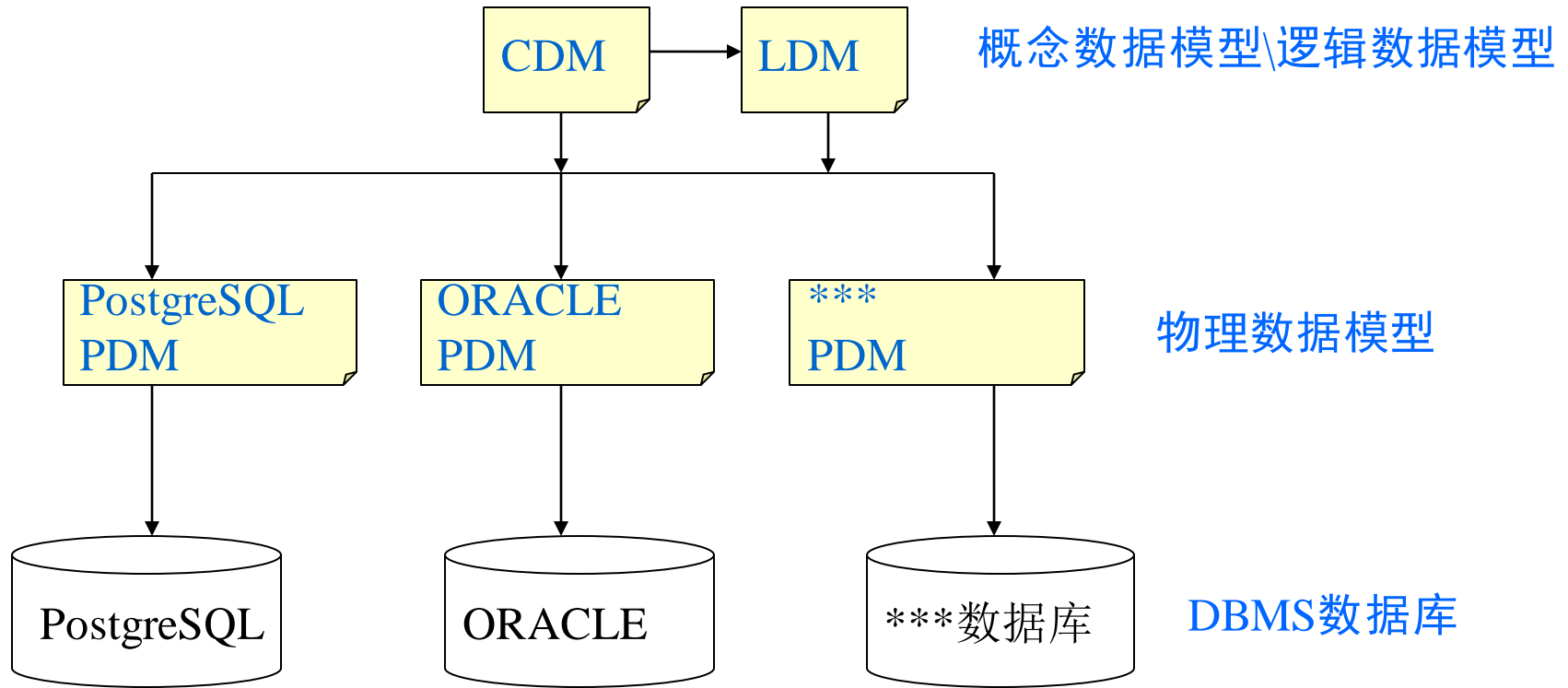
Power Designer的数据建模工具特点：

- 功能强大的软件开发生命周期建模工具
- 支持目前主流的数据库管理系统（如Oracle、SYBASE、SQL Server、DB2、MySQL、PostgreSQL等）
- 支持目前多种客户端开发工具
- 满足大、中、小型数据库建模设计

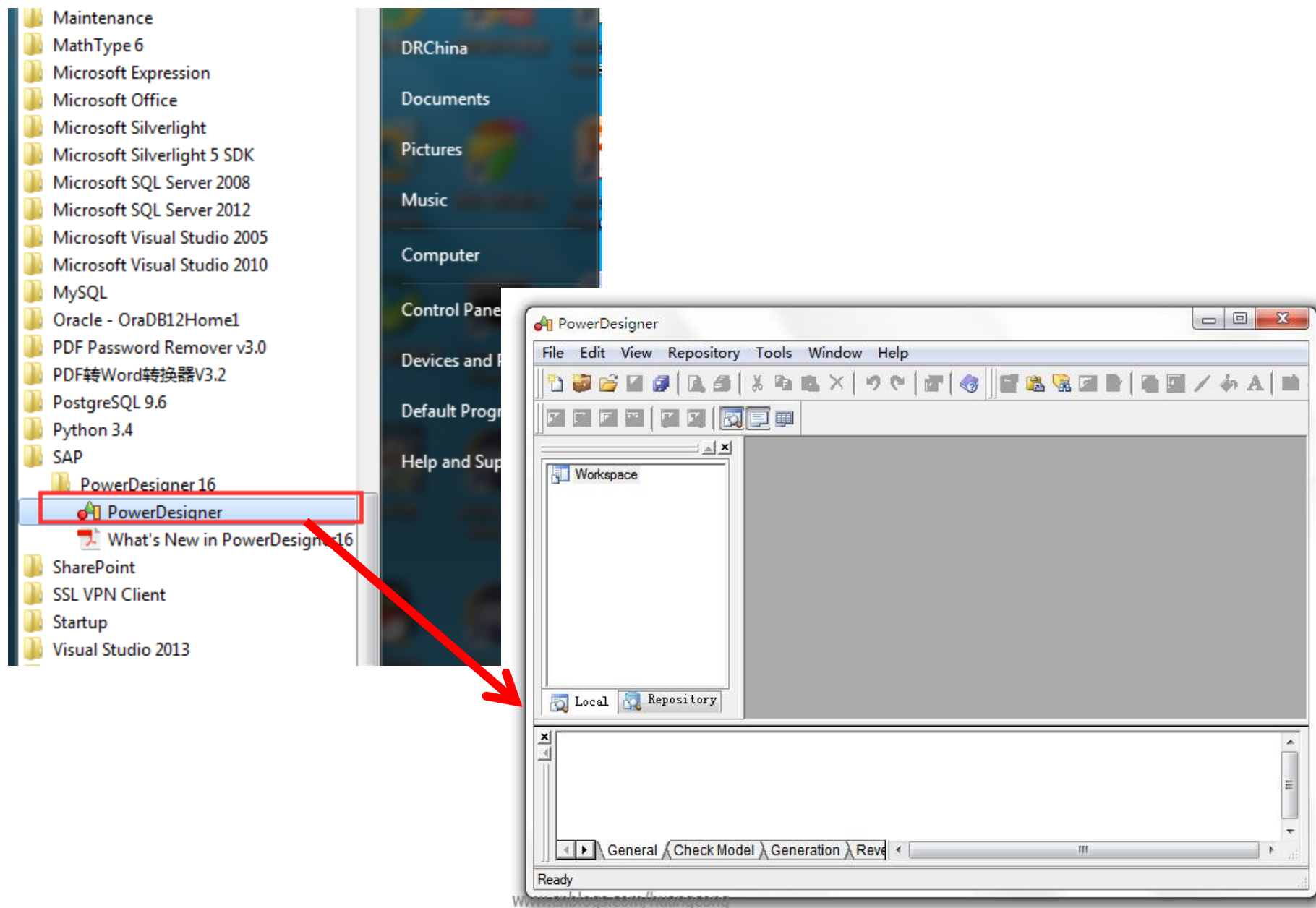
Power Designer可建立的数据模型

概念数据模型 Conceptual Data Model (CDM)	从用户角度所建模的系统数据对象及其关系，它帮助用户分析信息系统的数据结构关系。
逻辑数据模型 Logic Data Mode (LDM)	从系统分析员角度所建模的系统数据对象逻辑结构关系，它帮助开发人员分析信息系统的逻辑数据结构。
物理数据模型 Physical Data Model (PDM)	从系统设计人员角度所建模的系统数据物理存储及结构关系，它针对设计者具体定义信息系统的数据库表结构。

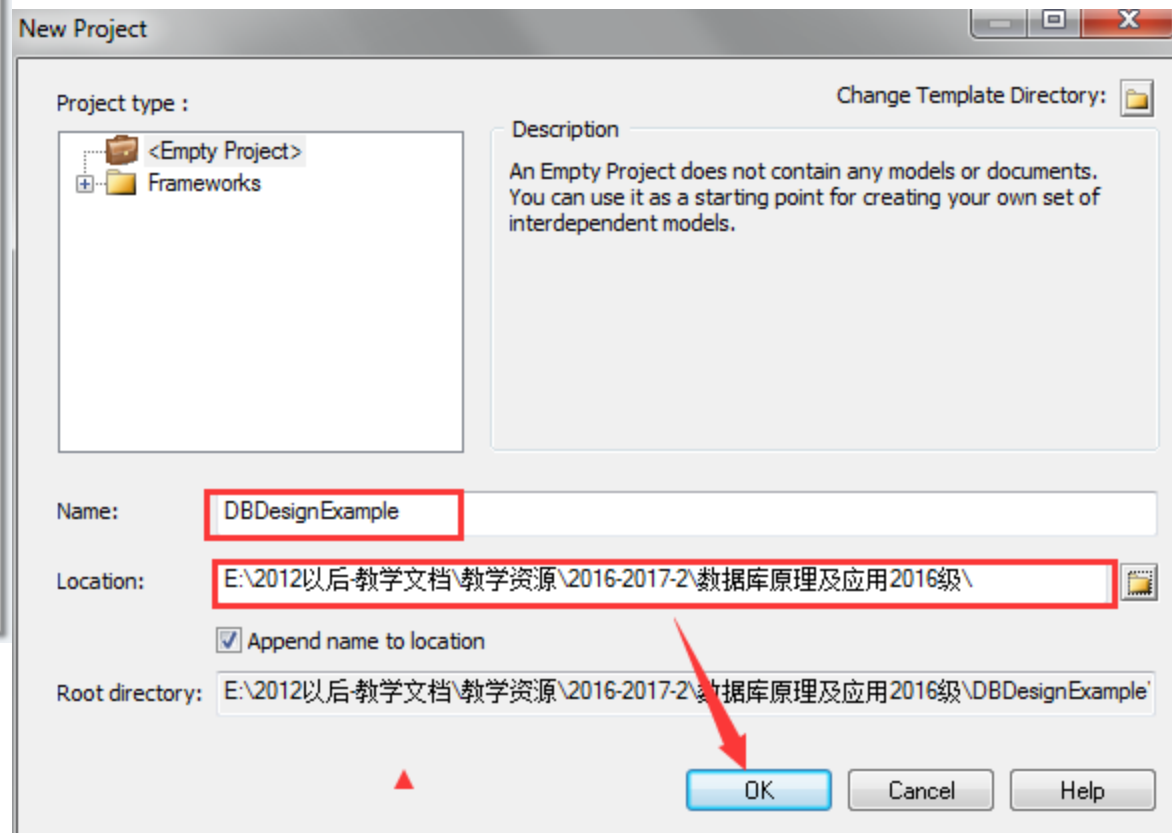
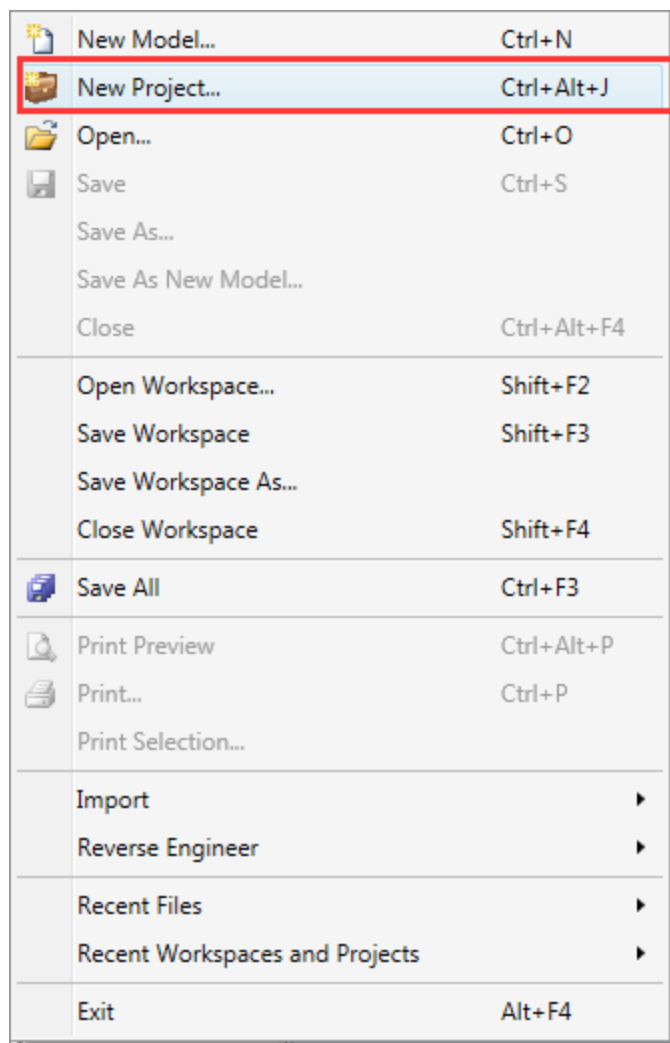
Power Designer各个数据模型之间的关系



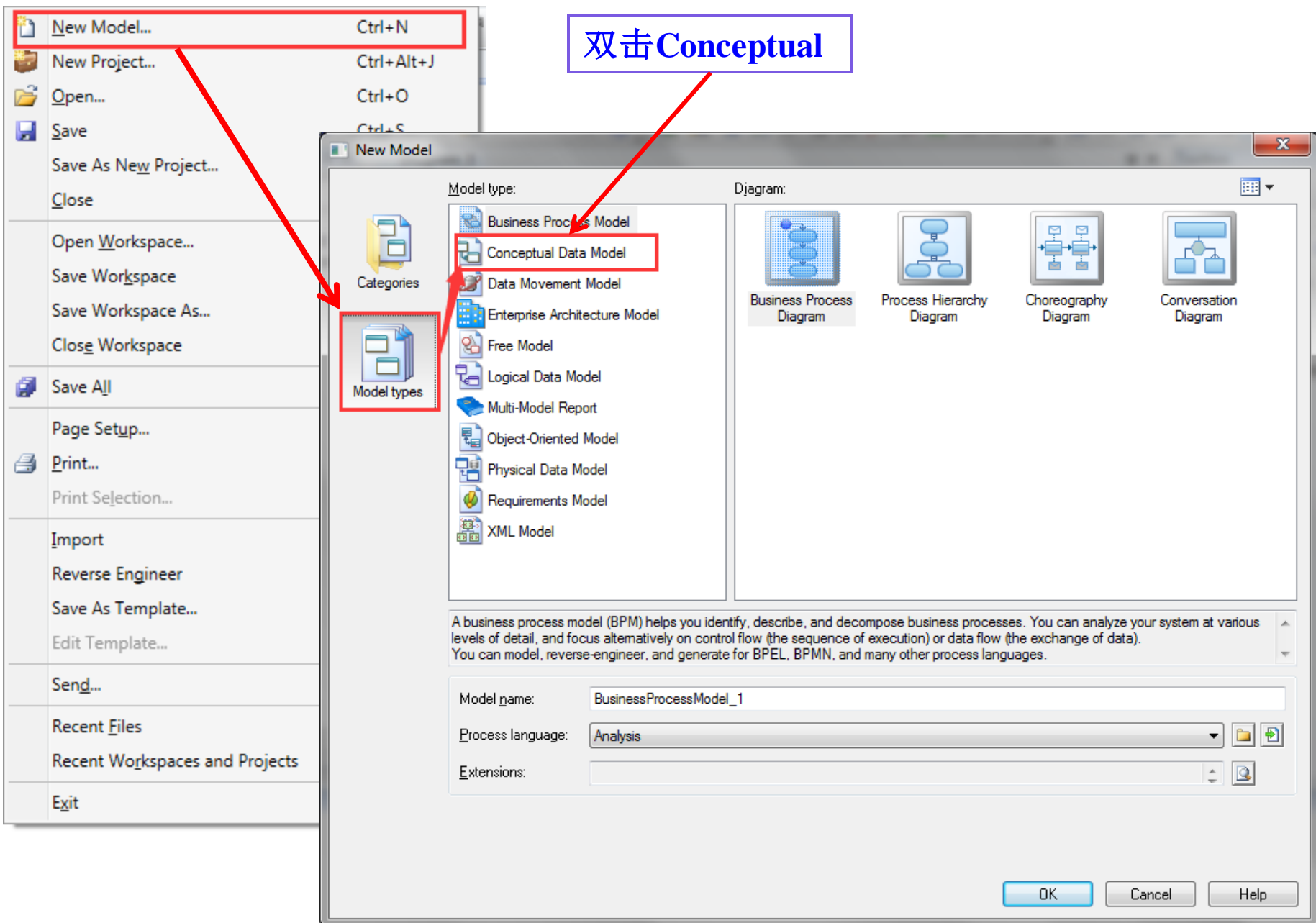
使用PowerDesigner工具进行数据库建模



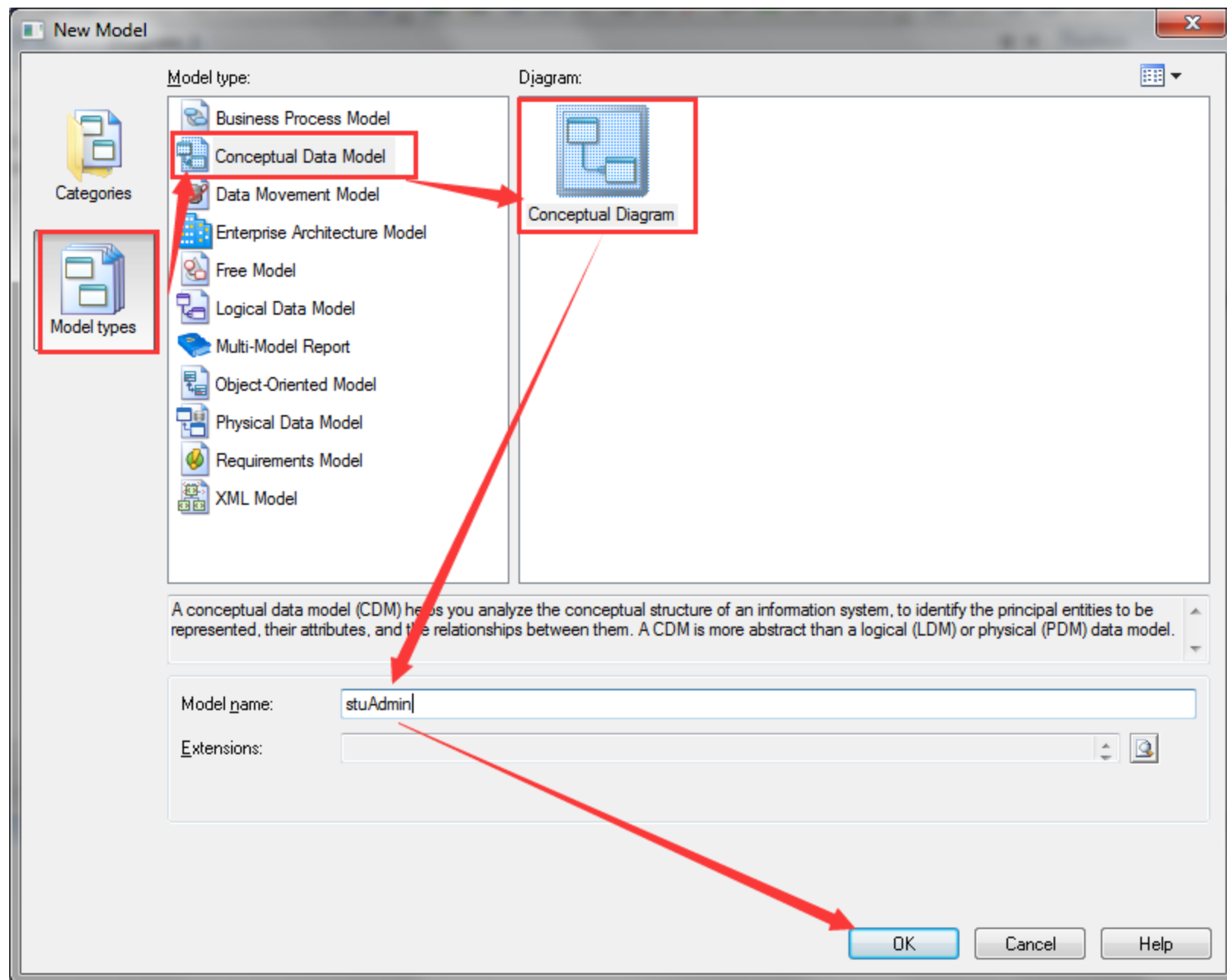
(1) 创建工程



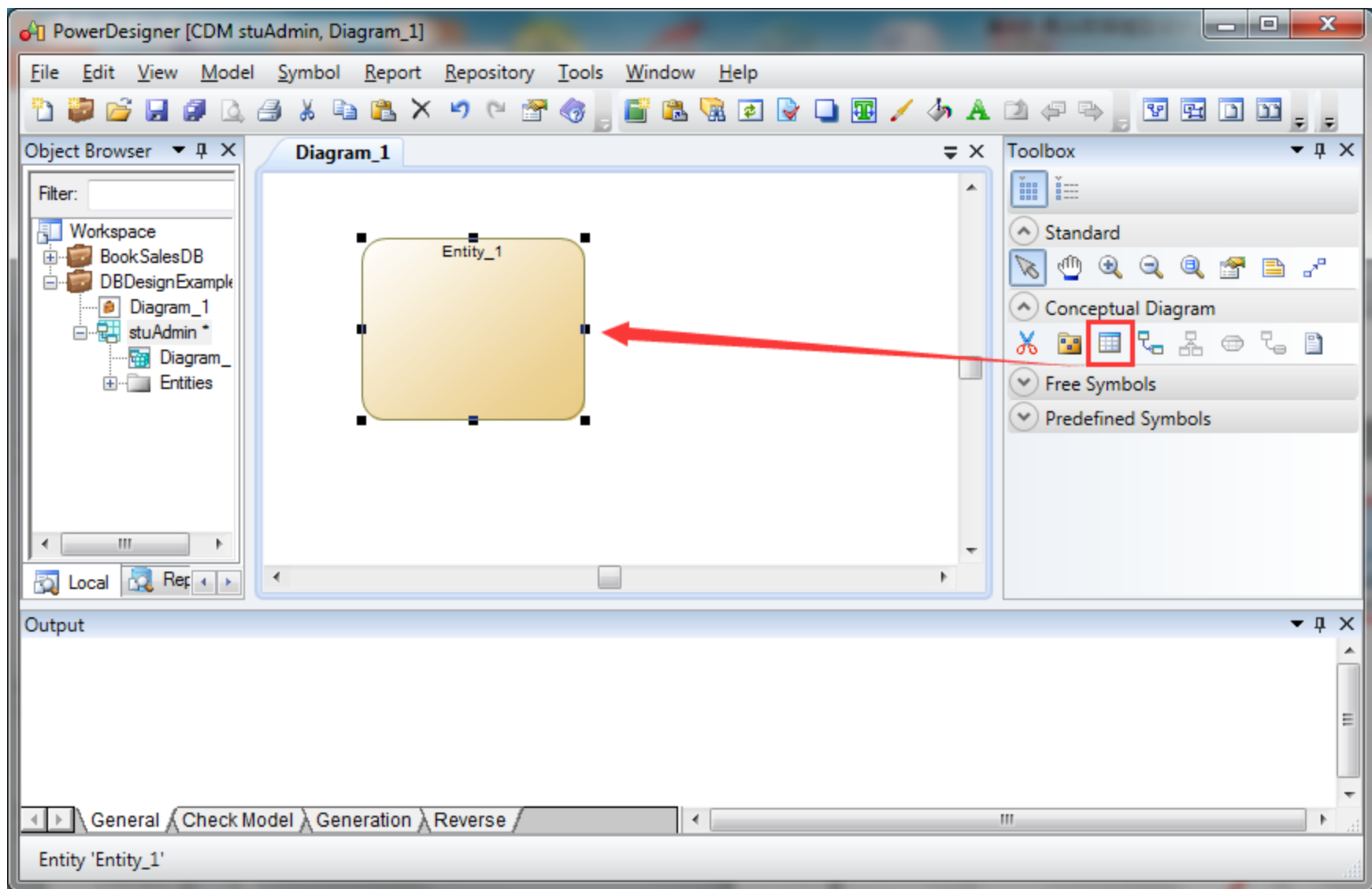
(2) 创建数据库模型



(2) 创建数据库模型



(3) 创建实体（或实体型）



(3) 给实体命名

The image shows a software window titled "Entity Properties - 学生 (student)". It contains several tabs: "General", "Attributes", "Identifiers", "Subtypes", "Notes", and "Rules". The "General" tab is selected. In this tab, there are input fields for "Name:" (containing "学生") and "Code:" (containing "student"), both of which are highlighted with red rectangular boxes. Below these is a "Comment:" field containing the text "学生的基本信息". Further down, there is a "Stereotype:" dropdown menu, a "Number:" field with a checked "Generate" checkbox, a "Parent entity:" dropdown menu set to "<None>", and a "Keywords:" field. At the bottom of the window, there are buttons for "More >>", "OK", "Cancel", "Apply", and "Help".

Entity Properties - 学生 (student)

General Attributes Identifiers Subtypes Notes Rules

Name: 学生

Code: student

Comment: 学生的基本信息

Stereotype:

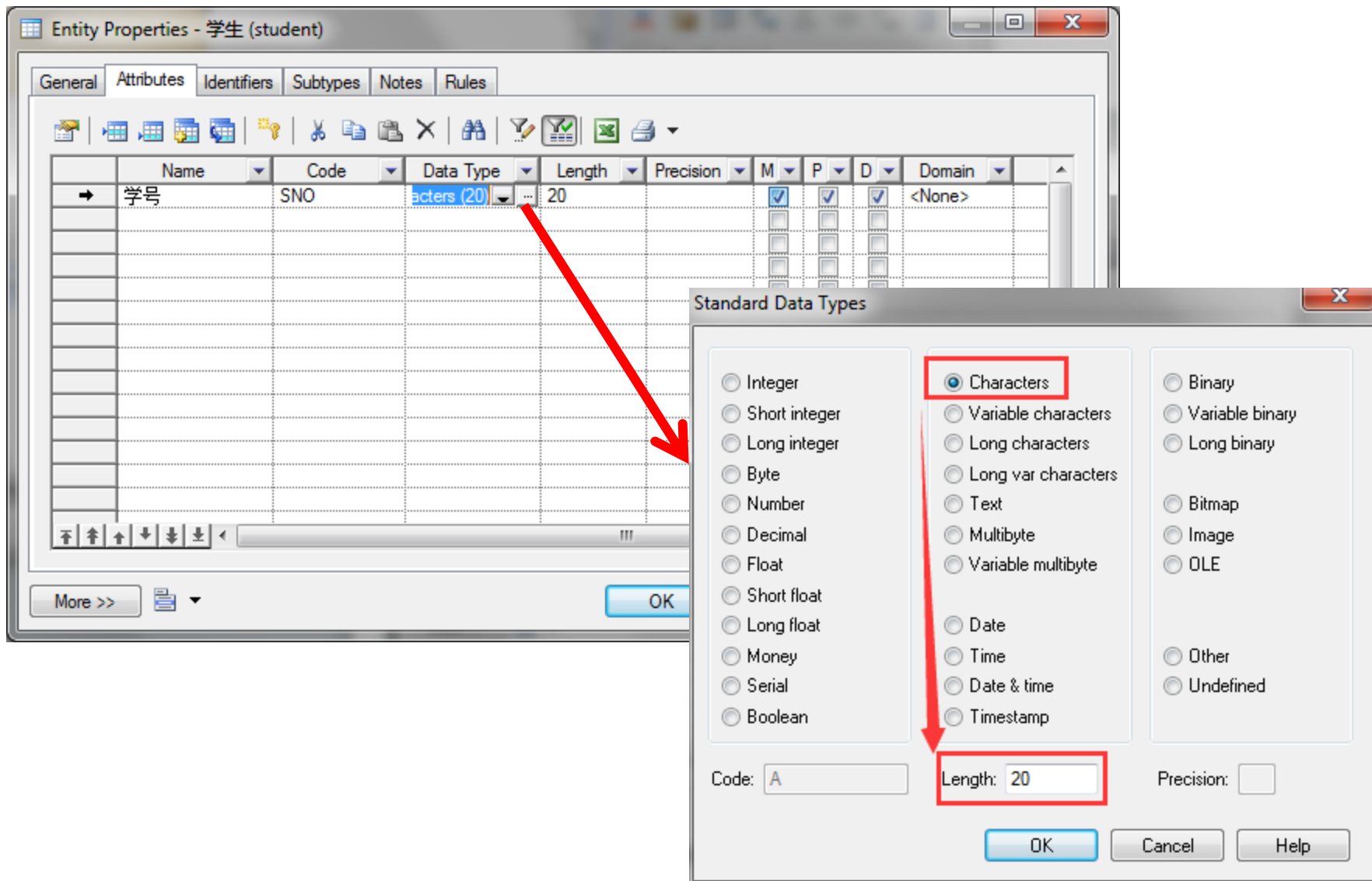
Number: ☒ Generate

Parent entity: <None>

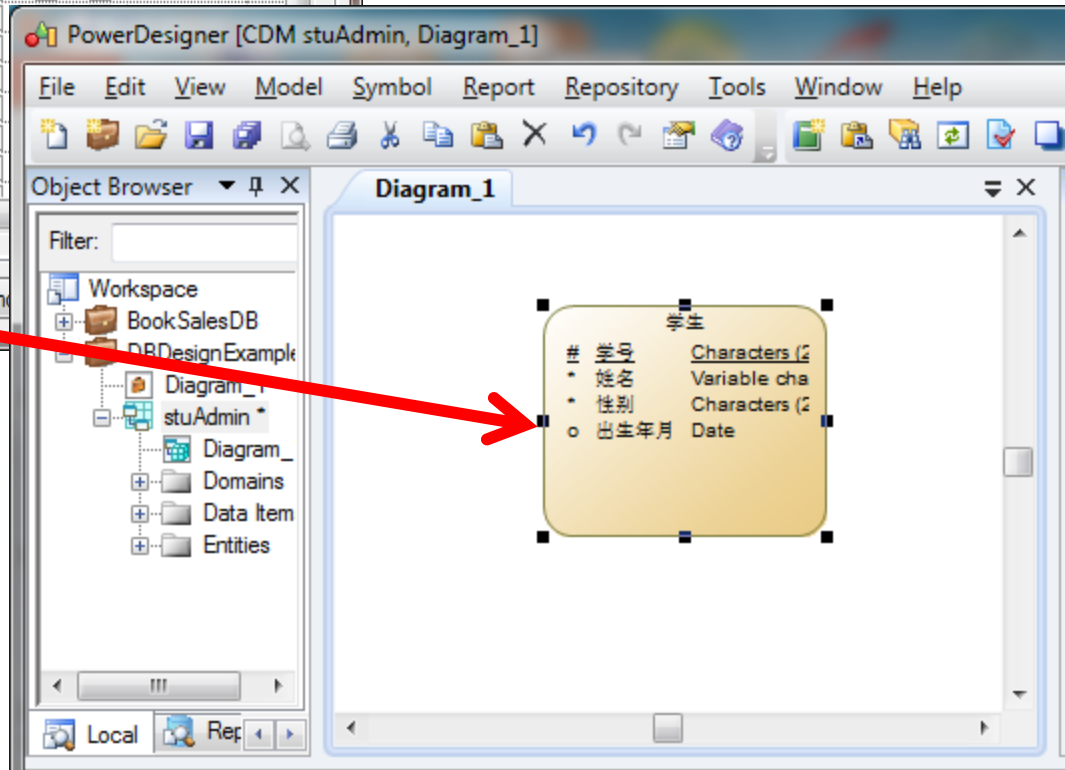
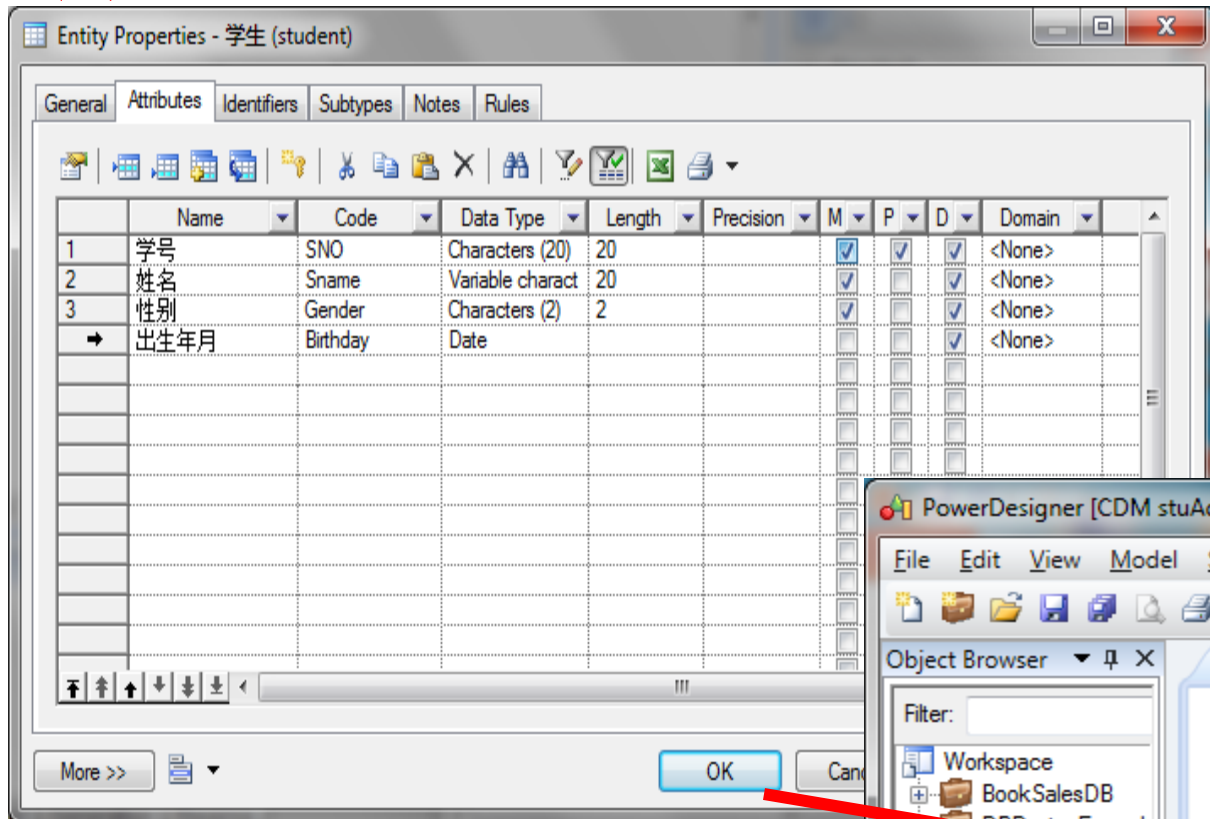
Keywords:

More >> OK Cancel Apply Help

(3) 给实体添加属性



(3) 属性添加完成



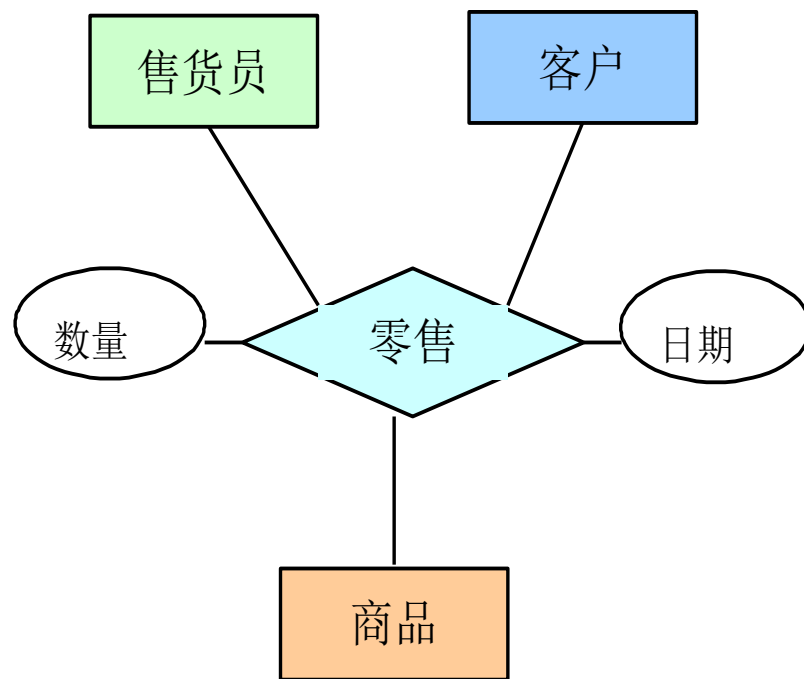
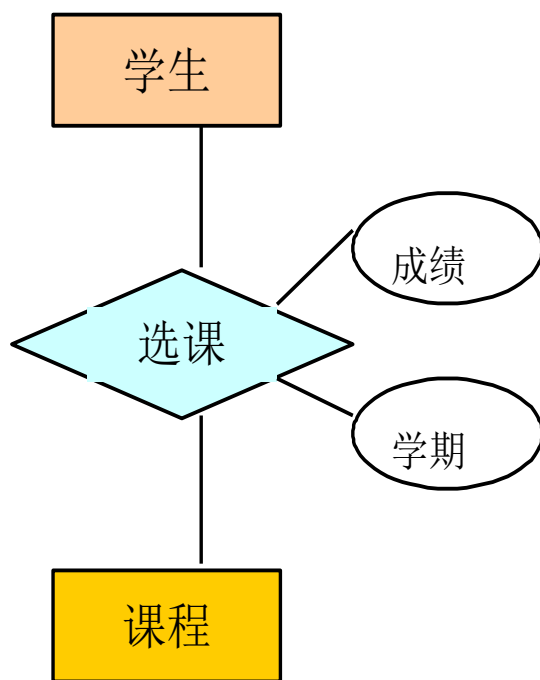
ER模型及基本概念

➤ 联系（Relationship）

- **概念：**反映为实体内部的联系和实体之间的某种关系。
- **实体内部**的联系通常是指组成实体的各属性之间的联系
- **实体之间**的联系通常是指不同实体集之间的联系
- **示例：**选课是学生与课程之间的联系。
- **联系的属性：**联系也可有描述属性，记录联系的信息而非实体的信息。
- **示例：**选课的成绩和修课学期；零售的商品数量。
- **联系的识别：**联系由参与的实体唯一确定。
- **示例：**选课(学号、课程号)

ER模型及基本概念

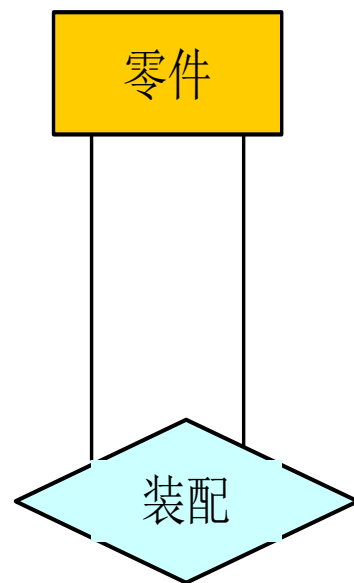
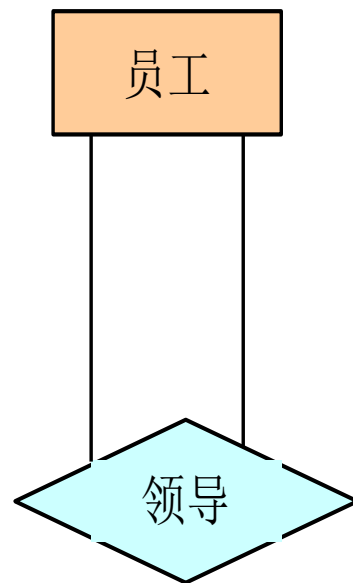
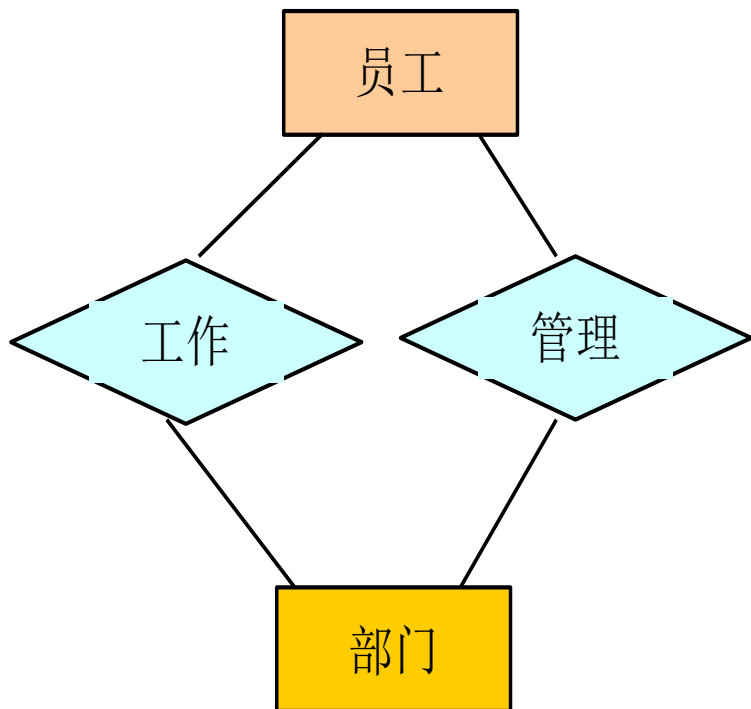
- 联系的阶，是指一个联系关联实体的个数。
 - **概念：** 一个关联的n个实体的联系，称该联系为n元联系
 - **示例：** 选课(二元联系)； 零售(三元联系)
 - **注：** 在ER图中，联系用菱形表示。



ER模型及基本概念

➤ 特殊类型的联系

- 两个实体之间可能有多个不同的联系；
- 一个联系所关联的是同一个实体集中的两个实体
- 示例：



实体间联系的类型

1. 一对一联系 (one-to-one, 1:1)

- **定义:** 设联系R关联实体A和B。如果对应A中的每个实体, B中有且仅有一个实体与之关联, 则称R是一对一联系, 简记作1:1联系。
- **示例:** 一个班级只有一个正班长, 一个班长只在一个班中任职



实体间联系的类型

2. 一对多联系 (one-to-many, 1:N)

- **定义:** 如果对应A中的每个实体, B中有n个实体($n \geq 0$)与之关联, 则称R是一对多联系型, 简记作1:N联系
- **示例:** 一个班级中有若干名学生, 每个学生只在一个班级学习



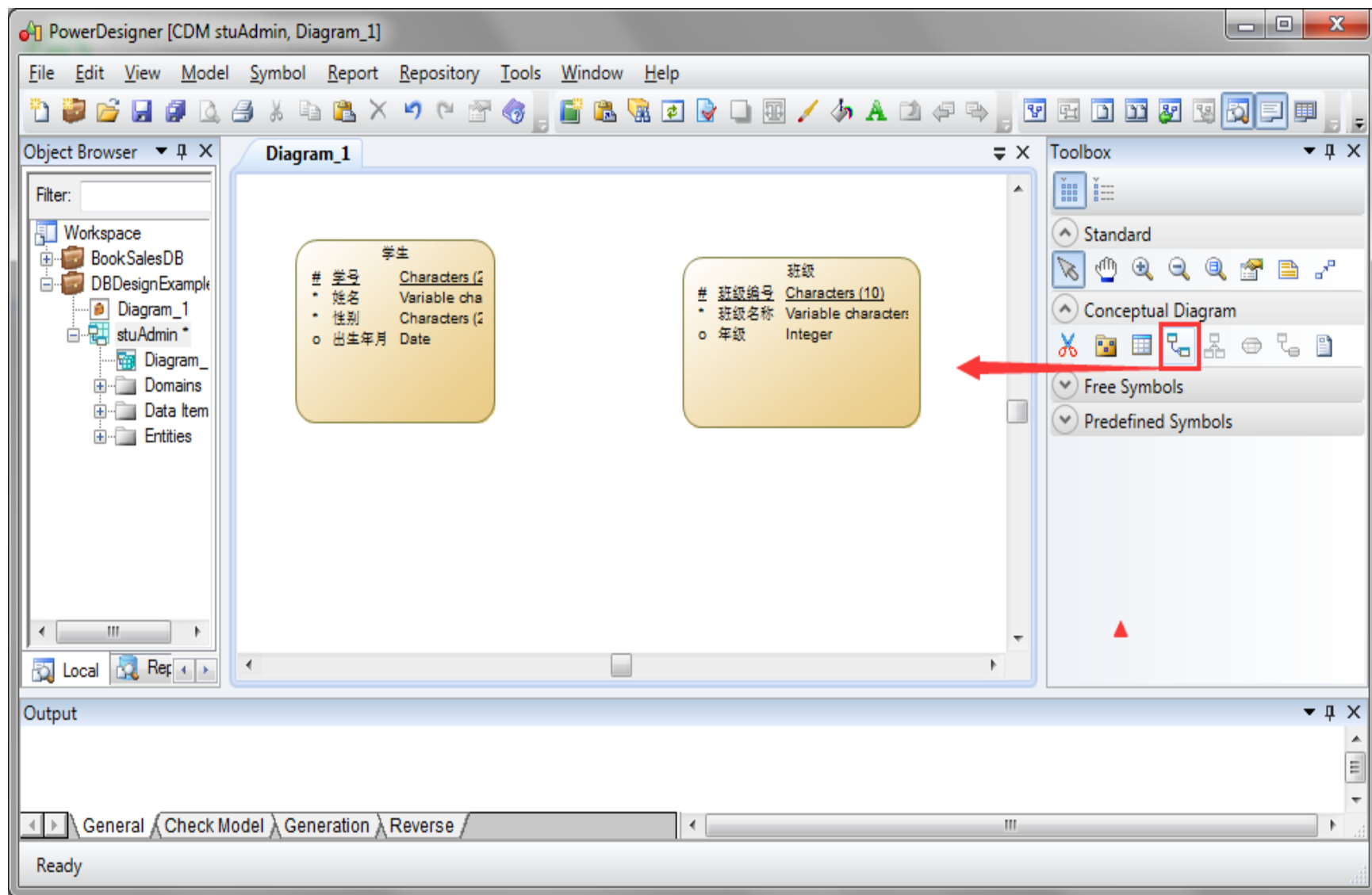
实体间联系的类型

3. 多对多联系 (many-to-many, M:N)

- **定义:** 如果对应A中的每个实体, B中有n个实体($n \geq 0$)与之关联, 如果对应B中的每个实体, A中有m个实体($m \geq 0$)与之关联, 则称R是**多对多联系**, 简记作**M:N**联系
- **示例:** 一门课程同时有若干个学生选修, 一个学生可以同时选修多门课程



(4) 添加联系



(4) 修改联系类型

Relationship Properties - 学生属于班级 (StuINClass)

Entity 1: 班级 Entity 2: 学生

General

Name: 学生属于班级

Code: StuINClass

Comment:

Stereotype:

Entity 1: 班级

Entity 2: 学生

☒ Generate

Keywords:

More >> OK Cancel

Relationship Properties - 学生属于班级 (StuINClass)

Entity 1: 班级 Entity 2: 学生

General

Each 班级 may have one or more 学生.
Each 学生 must have one and only one 班级.

Cardinalities

☐ One - one ☒ One - many ☐ Many - one ☐ Many - many

Dominant role: <None>

班级 到 学生

Role name:

☐ Dependent ☐ Mandatory Cardinality: 0..n

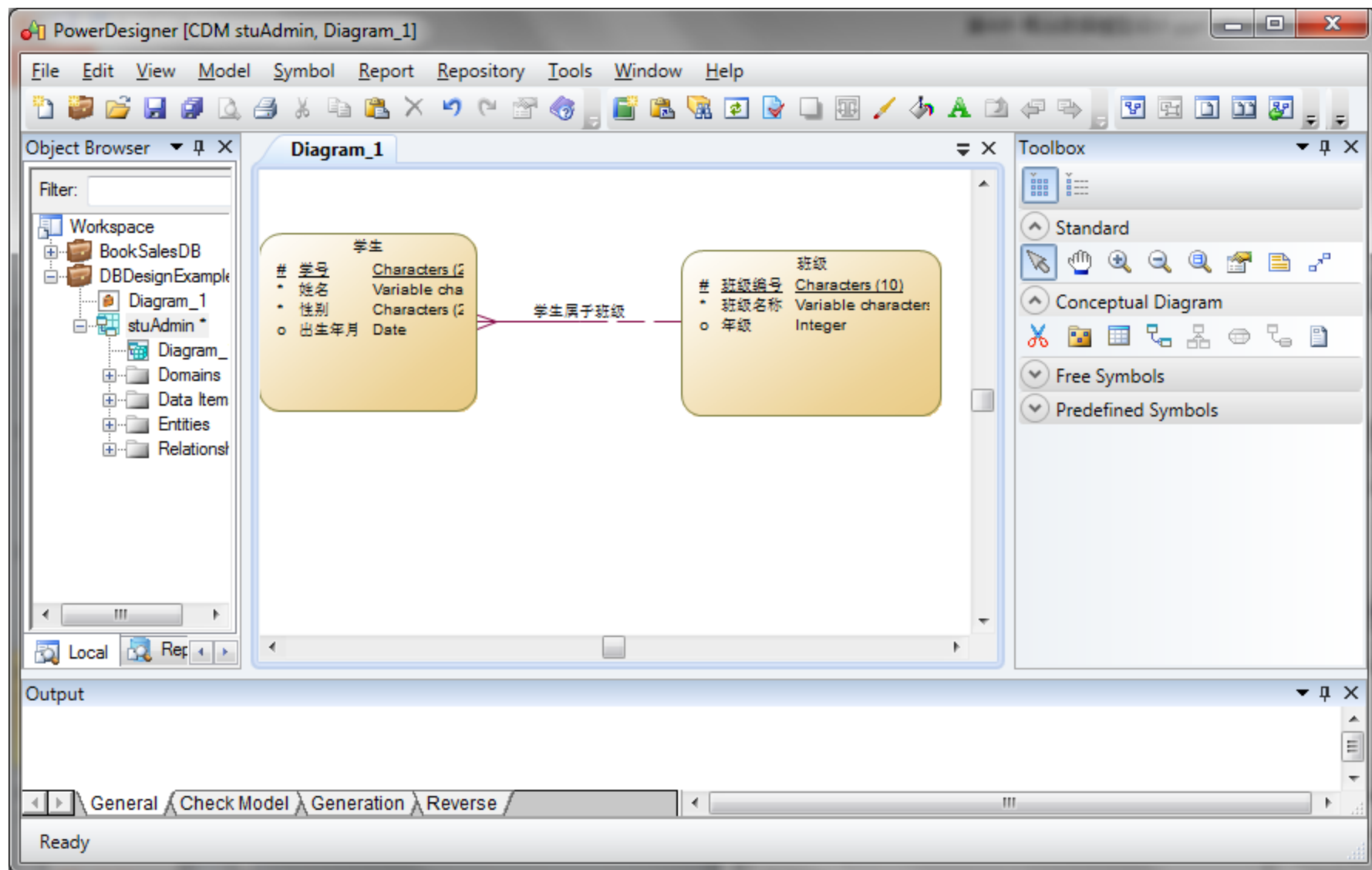
学生 到 班级

Role name:

☐ Dependent ☒ Mandatory Cardinality: 1..1

More >> OK Cancel Apply Help

(4) 修改联系类型



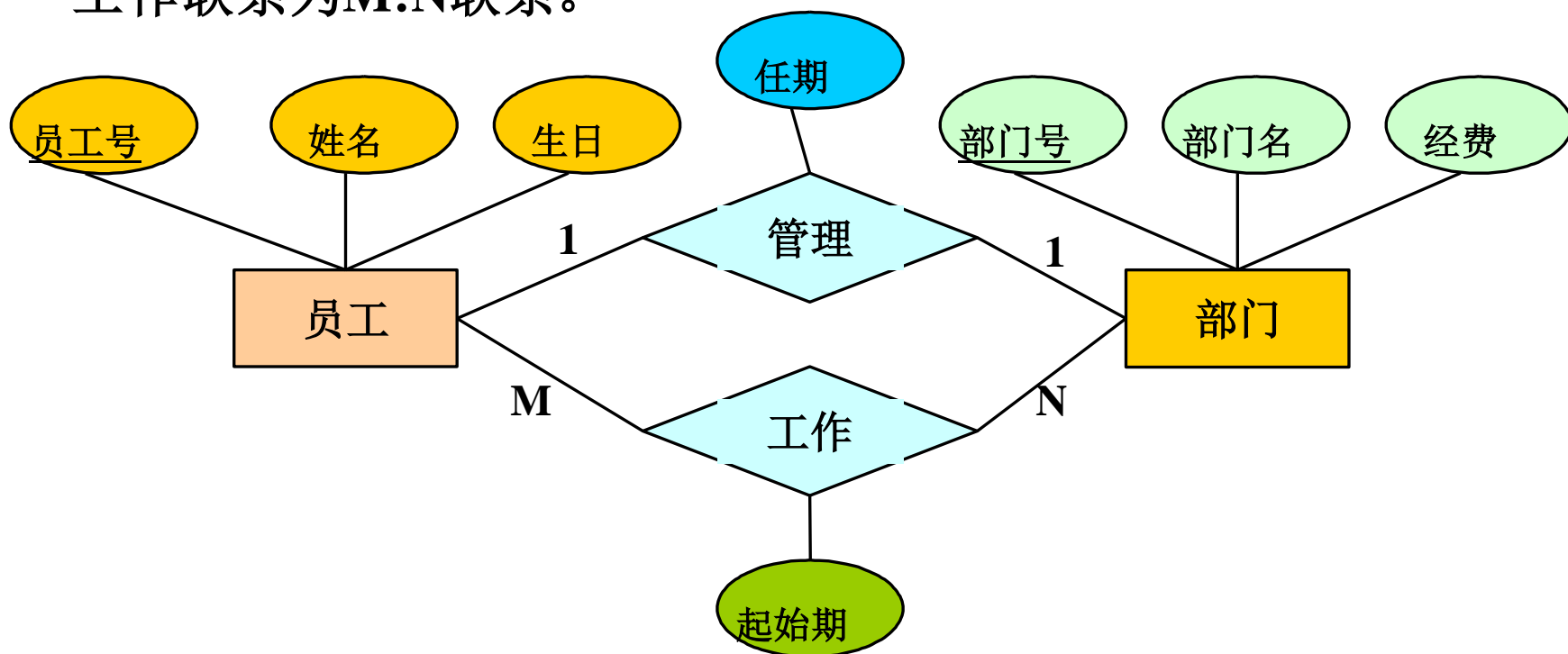
实体间的联系蕴含了某种完整性约束

➤ 两个实体间的联系

- 实体之间某种联系,体现了实体之间的某种约束.
- 约束关系是由实际应用所赋予的语义决定
- 前面所述的实体间的三种联系类型,实际上指的是
一些约束,分别为: 一对一约束、一对多约束和多
对多约束。

➤ 示例：如果规定：

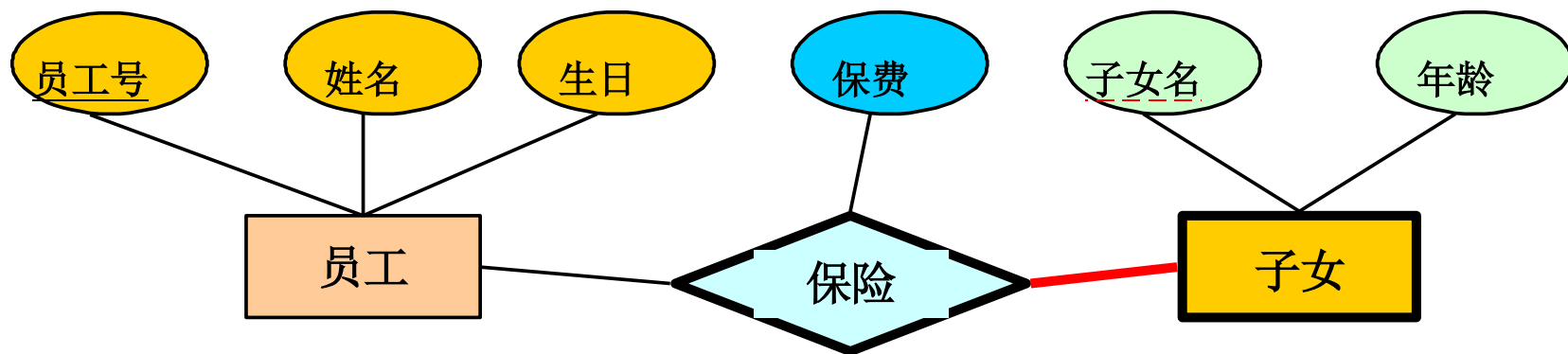
- (1)一个部门只有一个经理，而一个经理只能管理一个部门，则该管理联系为1:1联系。
- (2)一个员工可以是多个部门的经理，而一个部门最多只能有一个经理，则该管理联系为1:N联系。
- (3)一个员工可以在多个部门工作，而一个部门有多个员工，则该工作联系为M:N联系。



弱实体(Weak Entities)

- 前面所讲的实体总存在键。但实际情况中，并不总是如此。
- 概念：不存在键的实体，称为弱实体。不同弱实体的属性值可能完全相同，因此，难以区别。为此，弱实体型需要与一般的实体相关联。
- 识别实体型与识别联系：假如联系R关联弱实体A和一般实体B，A的弱实体可以通过与实体B相结合来加以区别，则B称为弱实体A的识别实体，R称为弱实体A的识别联系。

示例：弱实体和识别联系用粗线条，部分键加虚下划线



➤ 弱实体(Weak Entities)

- 识别实体与弱实体必须参与的是**1:n联系**，该联系即为该弱实体的识别联系
- 弱实体型必须**完全参与**识别联系。
- **部分键 (Partial Key)**：弱实体的某些属性与识别实体的键共同区分弱实体。这些弱实体属性称为弱实体的部分键。

ER模型描述概念分层

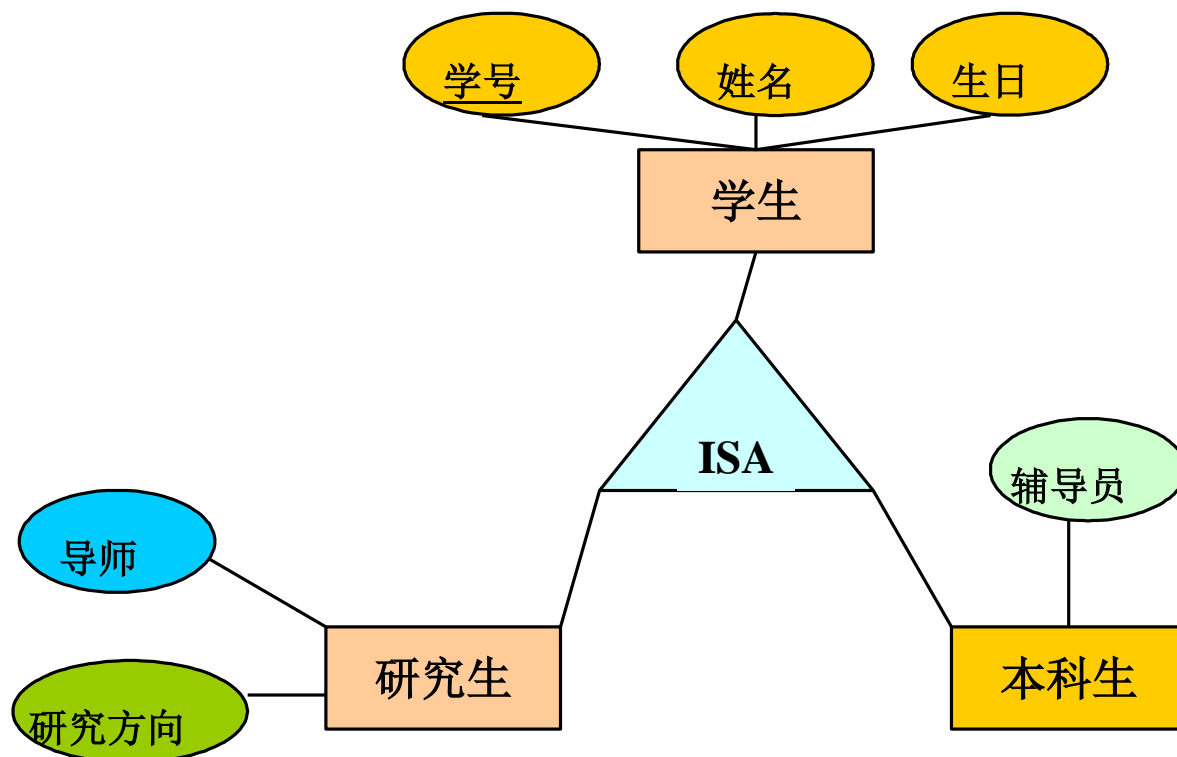
在某些应用中，需要将实体集划分为若干子类，分类后形成层次关系，最上层为超类（**Superclass**），下层即为子类。

示例：研究生和本科生都是学生的子类。

表示：研究生**ISA**（is a）学生、本科生**ISA**学生。**ISA**为这种类型层次的联系。

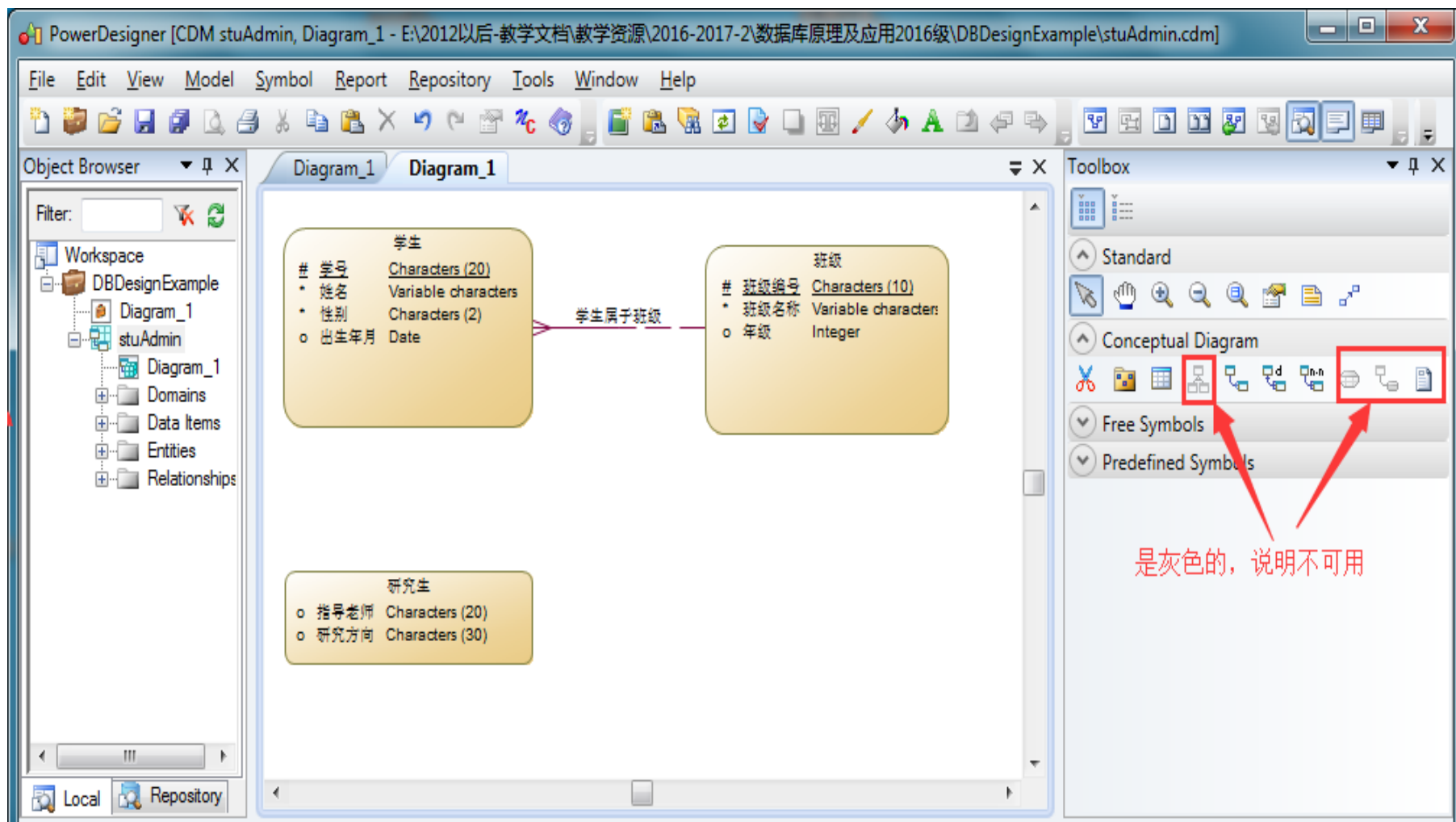
子类属性：除可继承超类属性外还可有自己独特的属性。

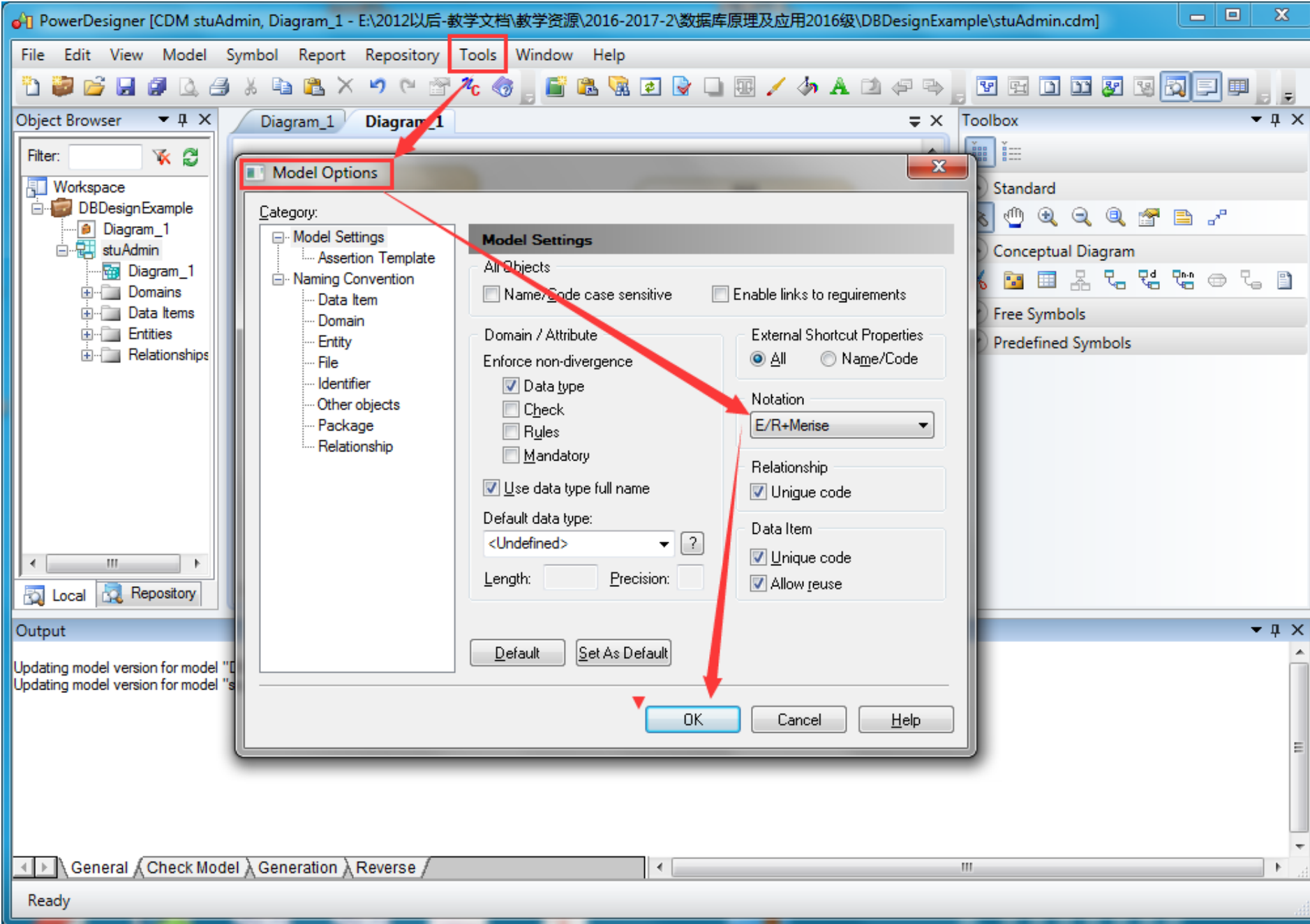
图示：

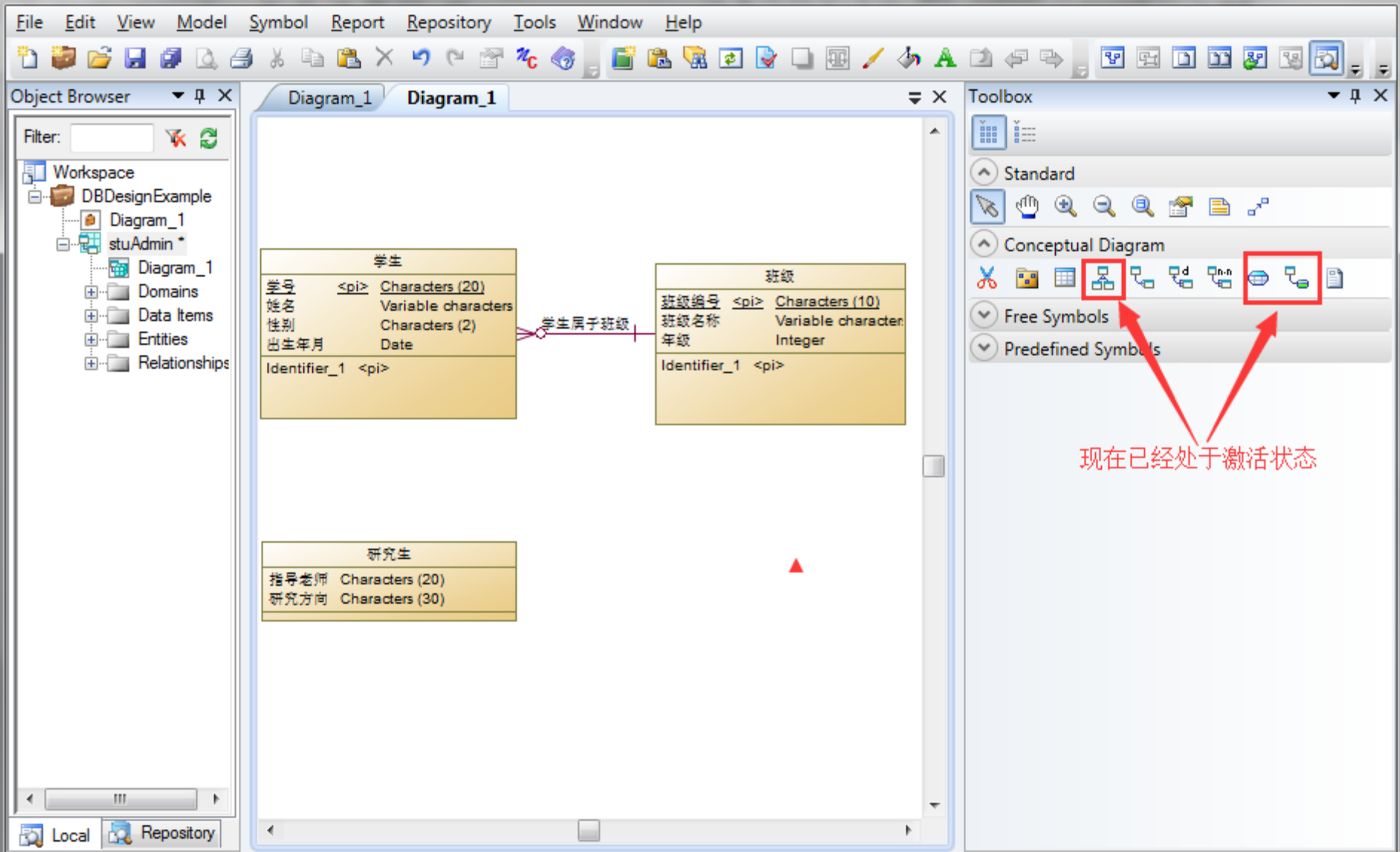


注意：有时还可按其他标准分类，可根据管理的需要来定。

示例：员工分资深员工 (**Senior Employee**) 与非资深 (**Junior**) 员工。



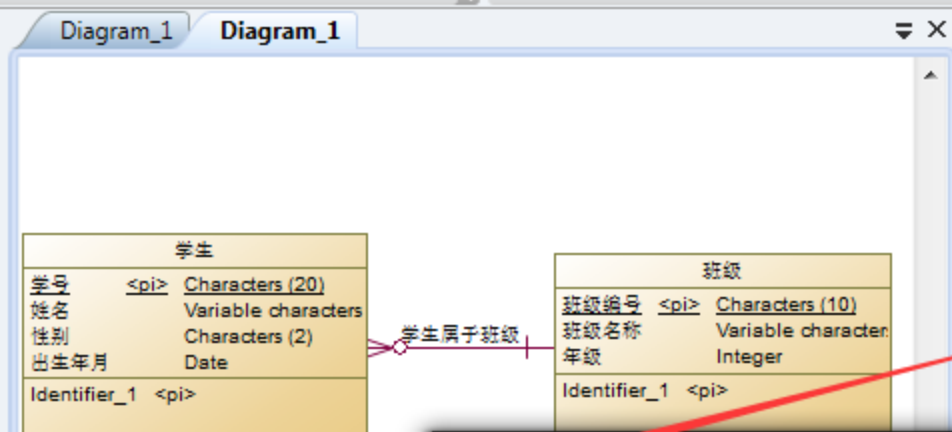




Object Browser

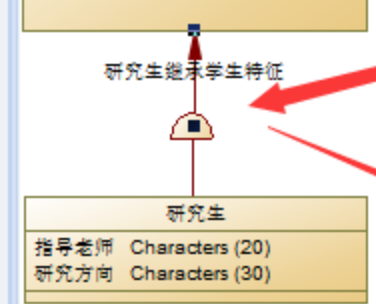
Filter:

- Workspace
 - DBDesignExample
 - Diagram_1
 - stuAdmin
 - Diagram_1
 - Domains
 - Data Items
 - Entities
 - Relationships
 - Inheritances



Toolbox

- Standard
- Conceptual Diagram
- Free Symbols
- Predefined Symbols



Inheritance Properties - 研究生继承学生特征 (graduateInStu)

General Generation Children Definition Rules

Name: 研究生继承学生特征

Code: graduateInStu

Comment:

Stereotype:

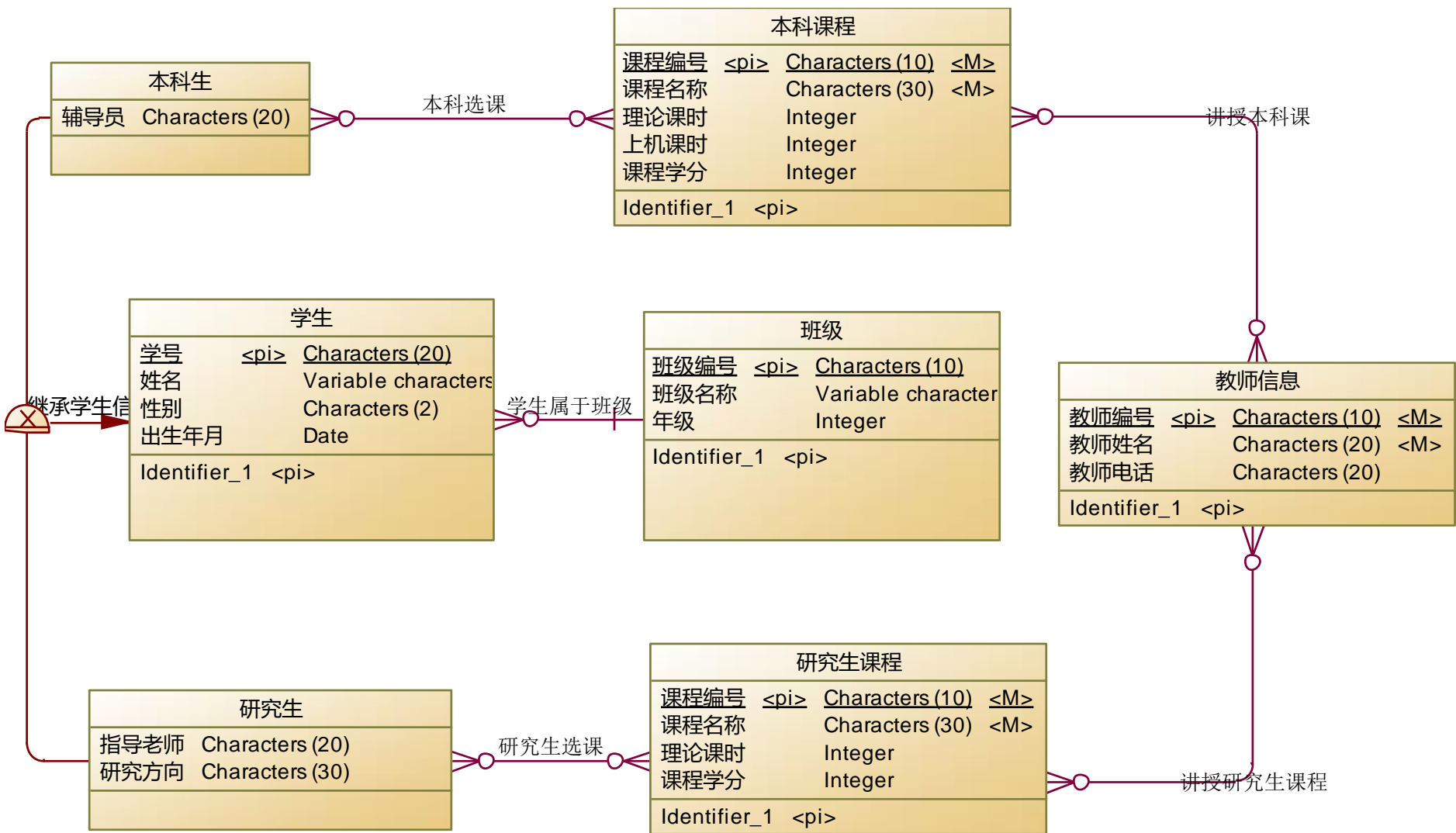
Parent: 学生

☒ Mutually exclusive children

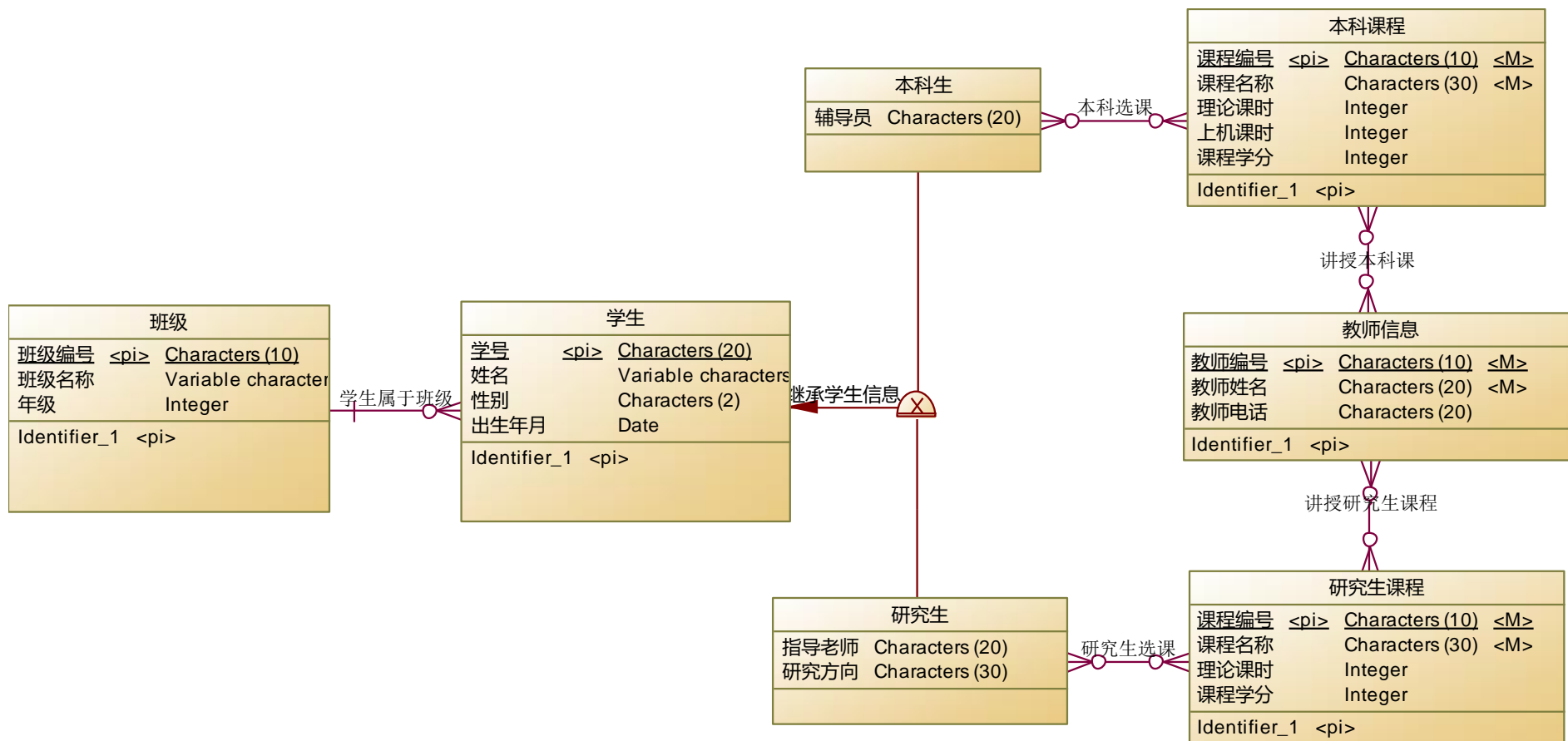
☒ Complete

Keywords:

More >> OK Cancel Apply Help



注意：调整布局的逻辑清晰美观



三、E-R 模型的设计实例

设计一个企业职工管理数据库，主要功能有：

人事管理（人事部门）

工资管理（财务部门）

项目管理（科研部门）

第1步 设计局部 E-R 模型

(1) 确定局部范围

可以按部门划分。

(2) 确定实体集

人事部门：职工、部门、职务

财务部门：职工、工资

科研部门：职工、项目

(3) 确定实体集的属性

人事部门：职工（职工号、姓名、性别、出生日期、工资）

部门（部门号、名称、电话、负责人）

职务（代号、名称、津贴）

财务部门：职工（职工号、姓名、性别、出生日期、职务）

工资（工资号、基本工资、津贴、保险、实发工资）

科研部门：职工（职工号、姓名、性别、出生日期、职务）

项目（项目号、名称、起始日期、鉴定日期）

(4) 确定联系集

人事部门：职工与部门的联系（分工）

职工与职务的联系（担任）

财务部门：职工与工资的联系（领取）

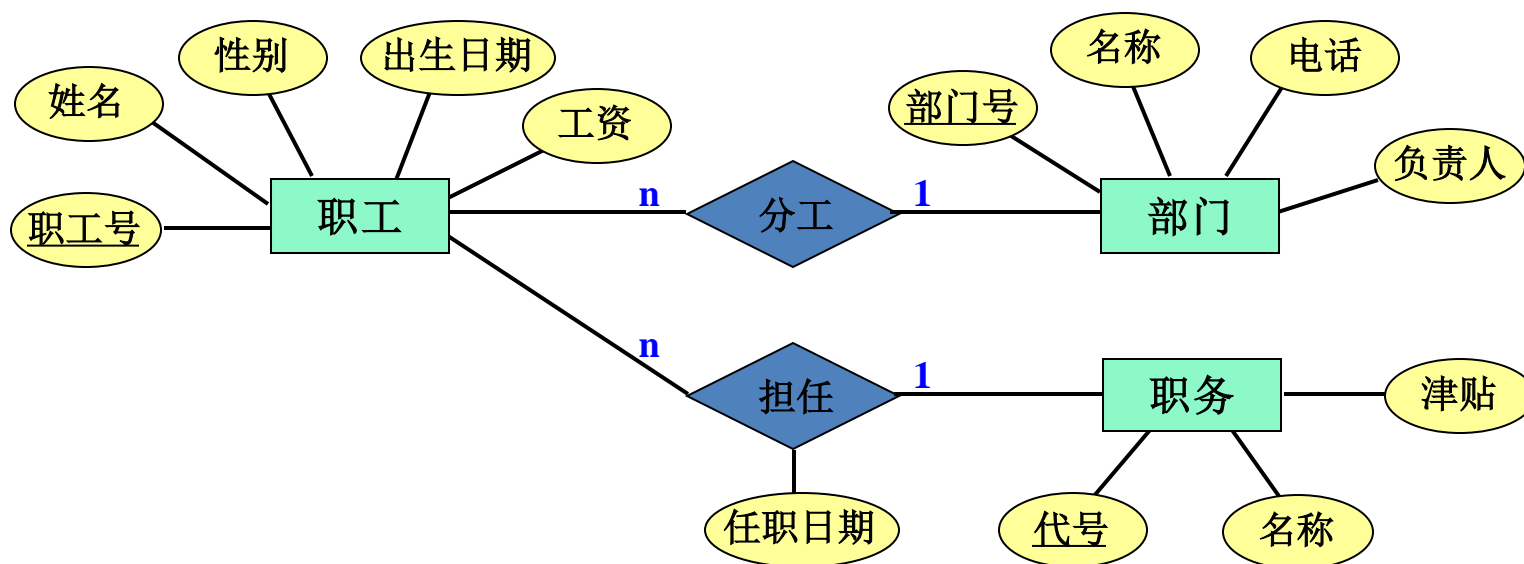
科研部门：职工与项目的联系（参与）

(5) 确定联系集的属性

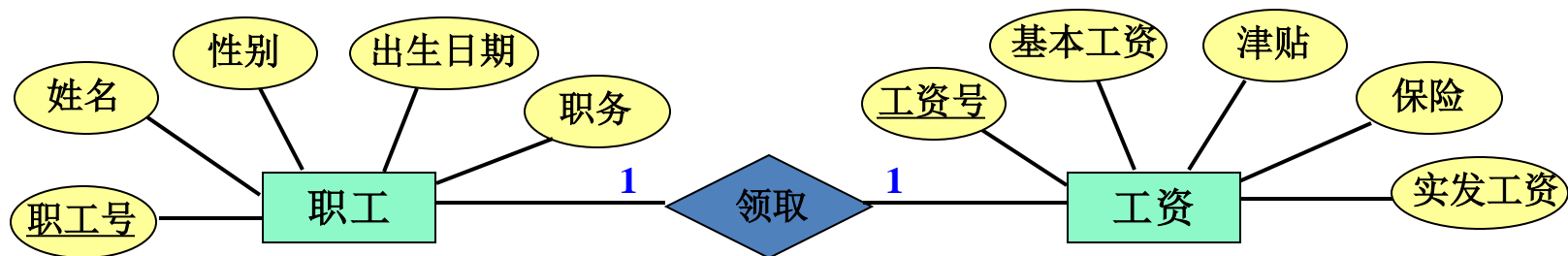
人事部门：职工与职务的联系有一个属性（任职时间）。

(6) 画出各局部的 E-R 模型

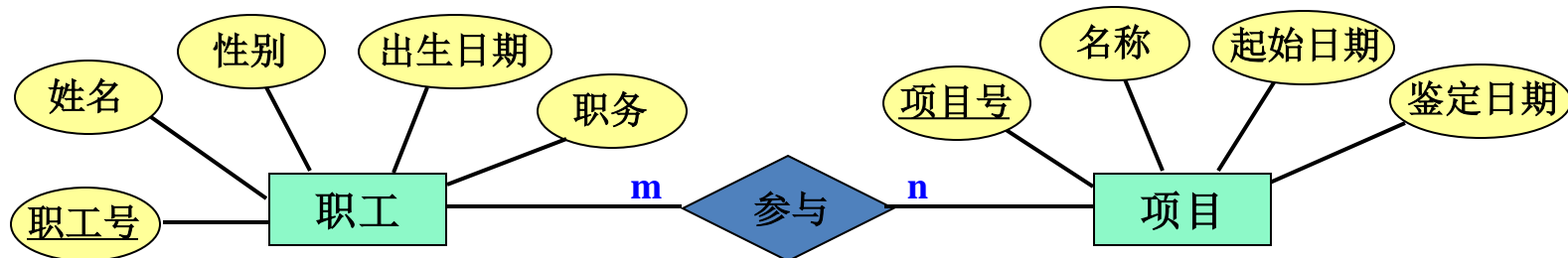
人事管理的局部 E-R 模型



工资管理的局部 E-R 模型



项目管理的局部 E-R 模型



2、组合局部 E-R 模型为全局 E-R 模型

(1) 消除各局部 E-R 模型之间的冲突

- 命名冲突：包括同名异义或异名同义等。
- 属性冲突：包括属性的数据类型、取值范围等。
- 结构冲突

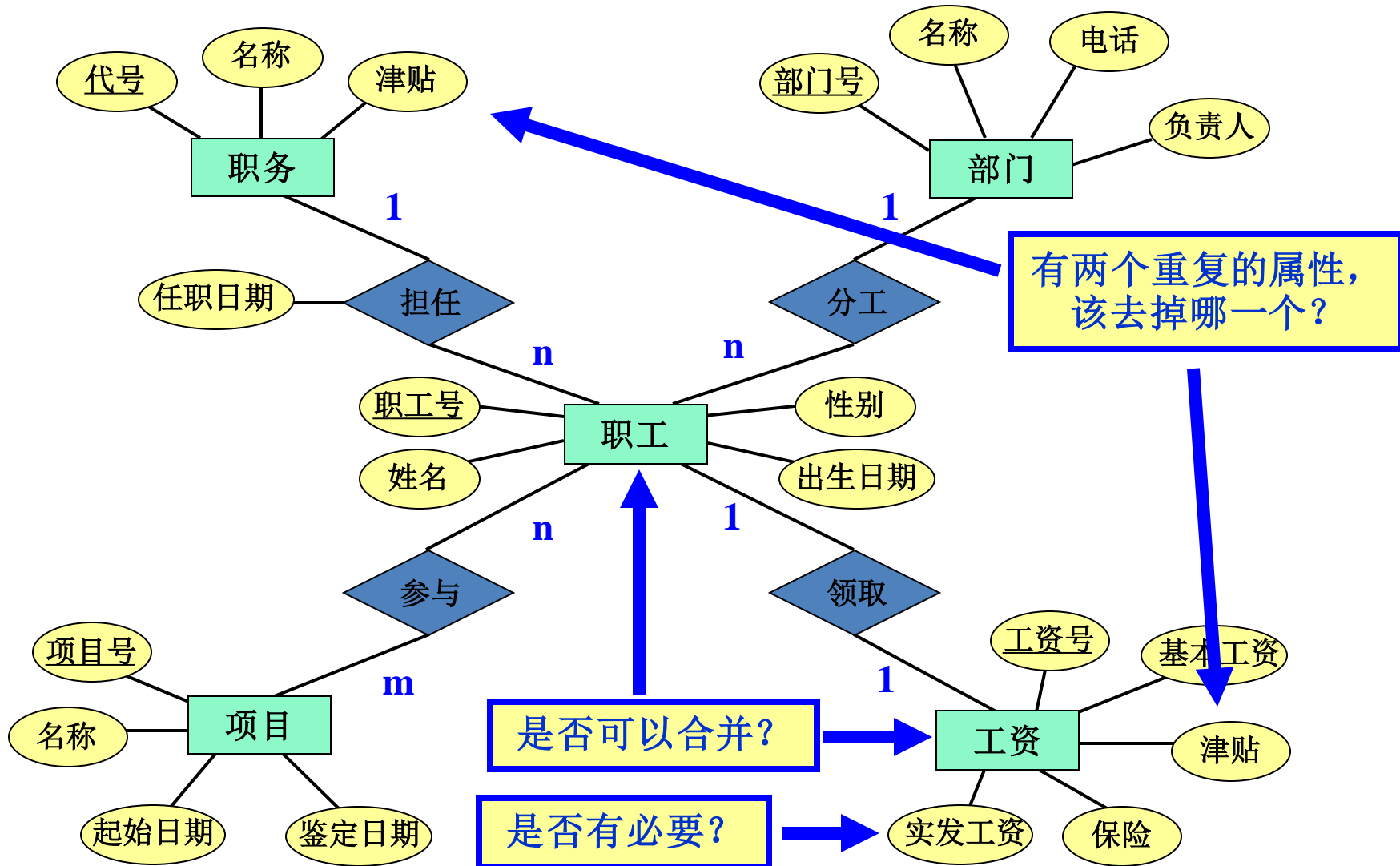
例如：在工资管理中，工资是一个实体，而在人事管理中，工资却是一个属性，合并前应去掉该属性。

在人事管理中，职务是一个实体，而在工资和项目管理中，职务却是一个属性，合并前应去掉该属性。

(2) 确定公共实体

如 职工实体。

(3) 局部 E-R模型以公共实体为中心，两两合并。



3、消除冗余，优化全局 E-R 模型

(1) 实体和联系尽量减少

1:1 联系的或具有相同键的两个实体集根据实际情况可以合并。如 职工和工资。

(2) 属性尽量减少

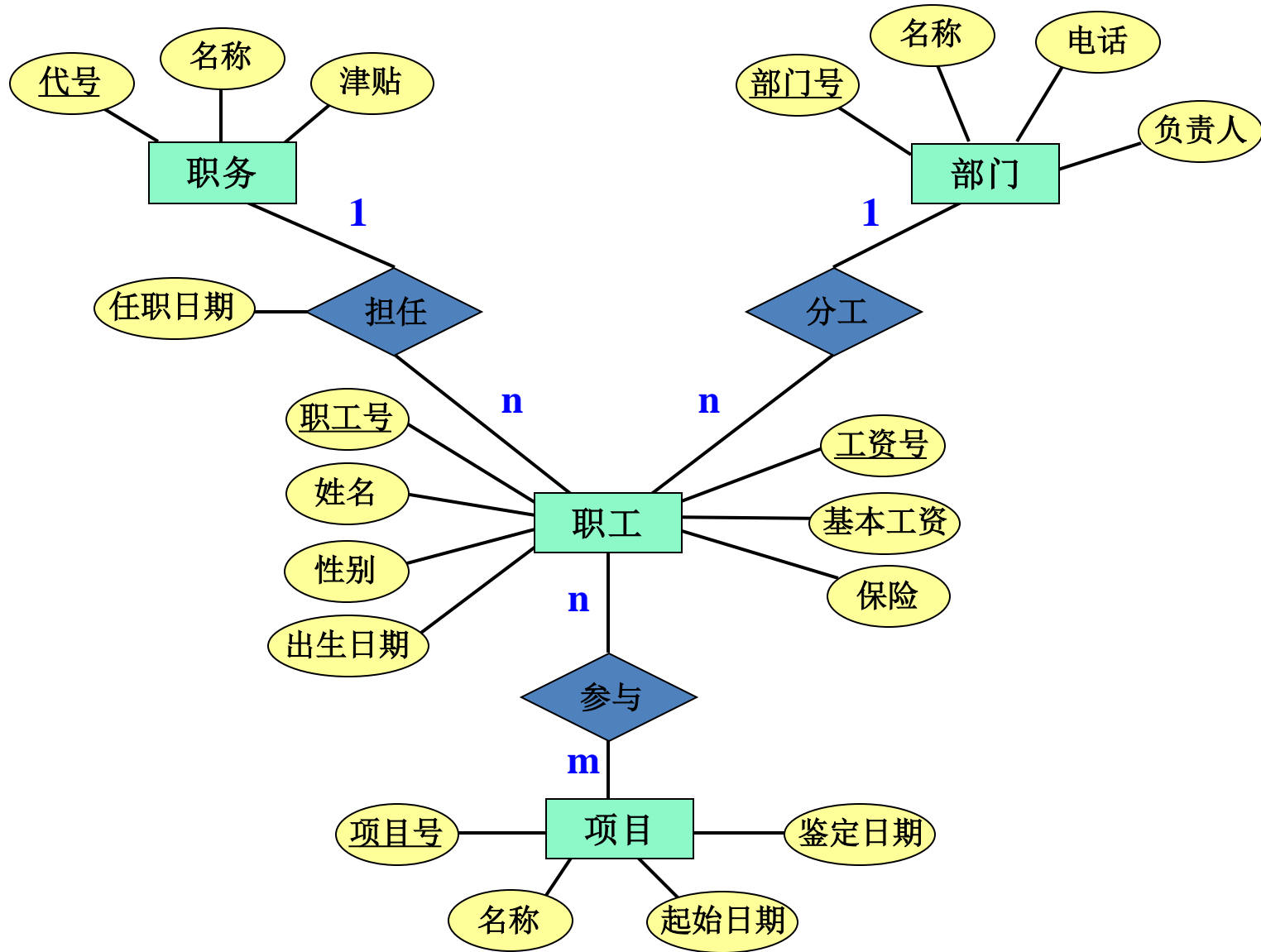
去除冗余的属性。

如 工资和职务两个实体都有津贴属性；

工资实体的实发工资属性可以由其他属性计算出来；

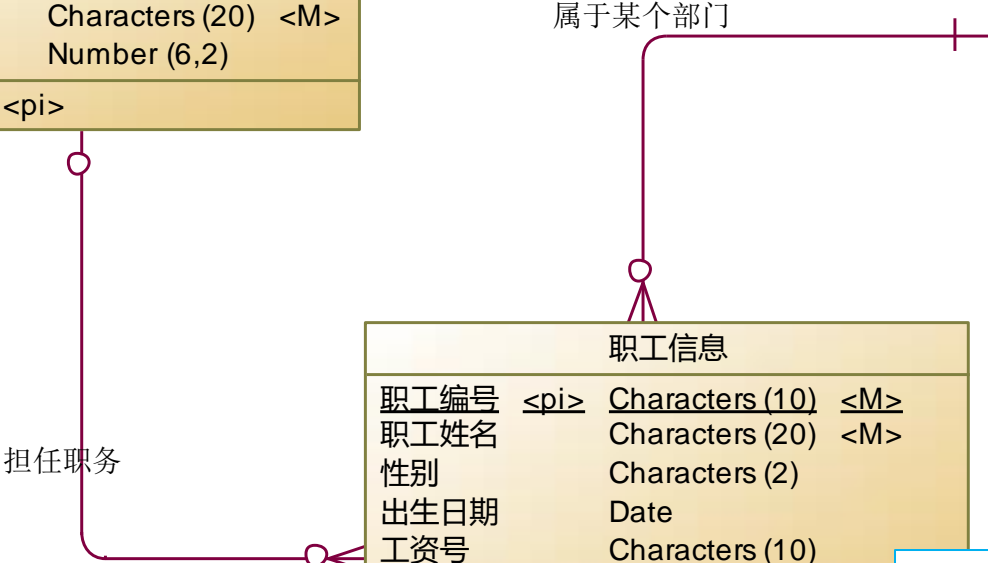
(3) 实体间的联系没有冗余

优化后的全局 E-R 模型



职务信息			
职务编号	<pi>	Characters (10)	<M>
职务名称		Characters (20)	<M>
职务津贴		Number (6,2)	
Identifier_1 <pi>			

部门信息			
部门编号	<pi>	Characters (10)	<M>
部门名称		Characters (30)	<M>
部门电话		Characters (20)	
负责人		Characters (20)	
Identifier_1 <pi>			



请思考：

工资号和职工编号可以取之一吗？

在实际应用中能满足需要吗？

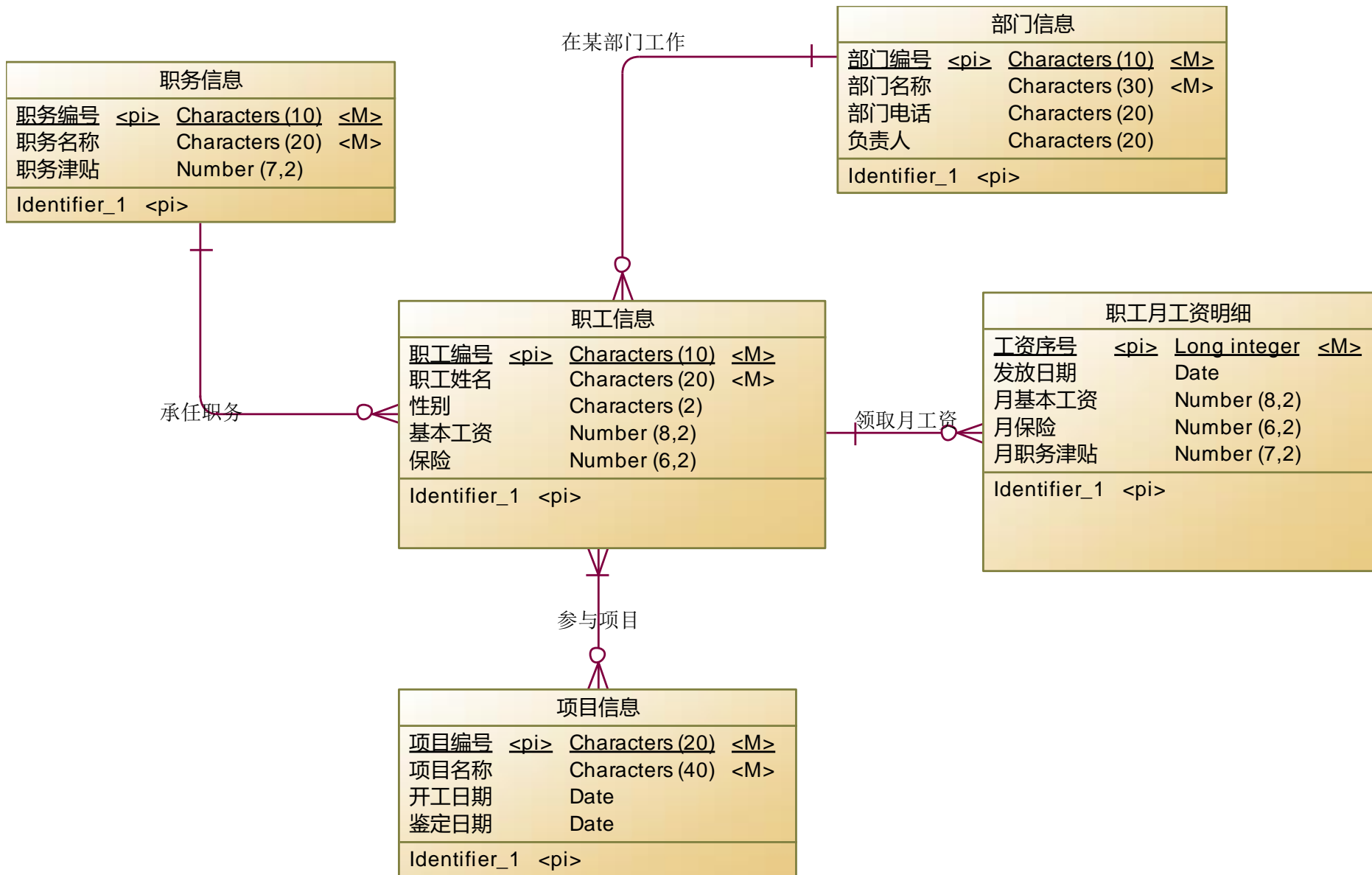
职工的涨工资如何处理？

每个月津贴、保险都不变吗？

职工信息			
职工编号	<pi>	Characters (10)	<M>
职工姓名		Characters (20)	<M>
性别		Characters (2)	
出生日期		Date	
工资号		Characters (10)	
基本工资		Number (8,2)	
保险		Number (5,2)	
Identifier_1 <pi>			

项目信息			
项目编号	<pi>	Characters (20)	<M>
项目名称		Characters (40)	<M>
开工日期		Date	
鉴定日期		Date	
Identifier_1 <pi>			

改进后 E-R 模型



三、E-R模型向关系模型转化

目前的数据库软件产品还不支持**概念模型**，不能直接对其进行编程，因此必须将**概念模型**转化为**关系模型**。

几个概念：

- 独立实体和弱实体

如 学生和学生家长。

- 基本属性和导出属性

如 出生日期和年龄，基本工资、津贴、保险和实发工资。

1、独立实体的转化

一个独立实体直接转化为一个**关系表**。

2、弱实体的转化

一个弱实体也可以直接转化为一个关系表，
且与对应的独立实体建立**依赖关系**。

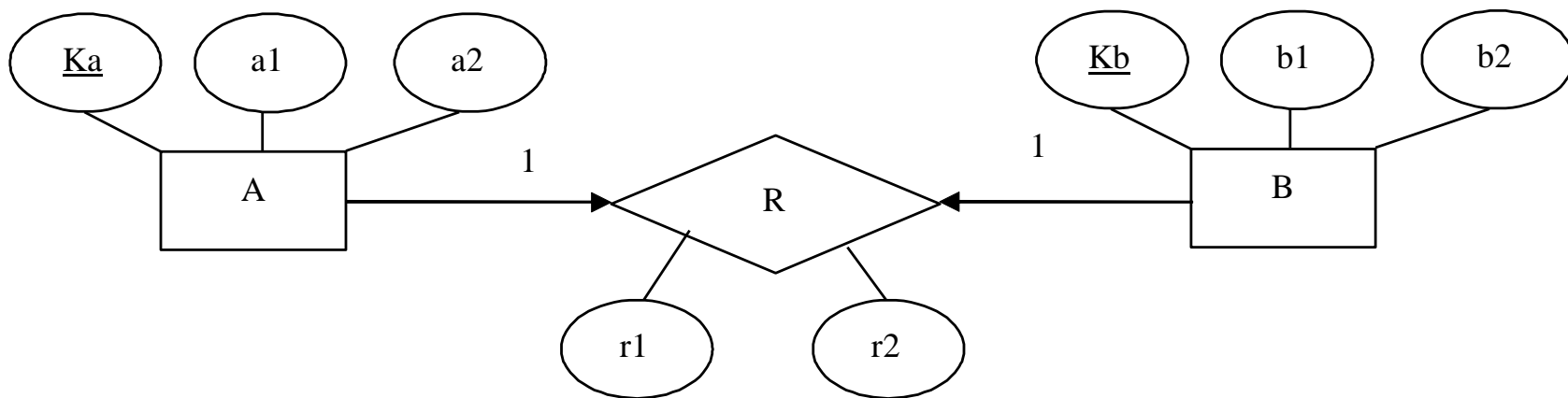
3、1:1 联系的转化

在 1:1 联系的两个实体的关系表中添加对方的键作为**外键**。

4、1:n 联系的转化

在 1:n 联系的 n 方实体的关系表中添加 1 方实体的键作为**外键**。

1:1联系的例子



转换结果为:

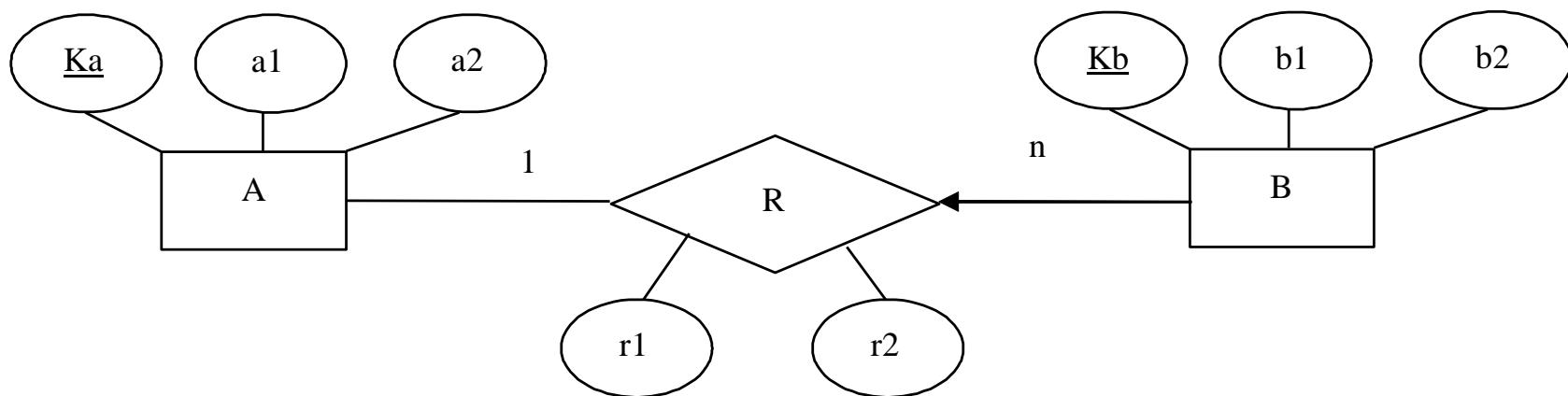
A (Ka,a1,a2), **Ka为主键**

B (Kb,b1,b2), **Kb为主键**

R (Ka, Kb, r1, r2), **Ka, Kb均为外键**

由于有两个键约束，故Ka, Kb都能单独作为R的主键，如果指定Ka为主键，则Kb就为候选键；否则，如指定Kb为主键，则Ka就为候选键。

1:n联系的例子



转换结果为:

A (Ka,a1,a2), **Ka**为主键

B (Kb,b1,b2), **Kb**为主键

R (Ka, Kb, r1, r2), **Kb**为主键, 另外, **Ka, Kb**均为外键

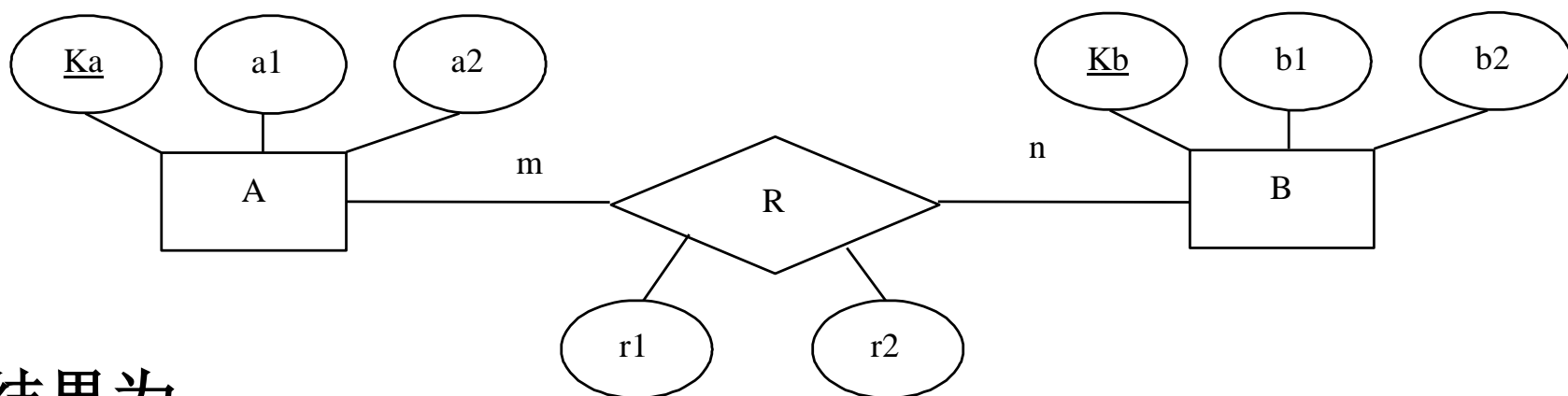
5、m : n 联系的转化

需要单独建立一张关系表，分别用两个实体的键作为**外键**。

6、多元联系的转化

需要单独建立一张关系表，分别用该联系的全部实体的键作为**外键**。

m:n联系的例子



转换结果为:

A (Ka,a1,a2), **Ka**为主键

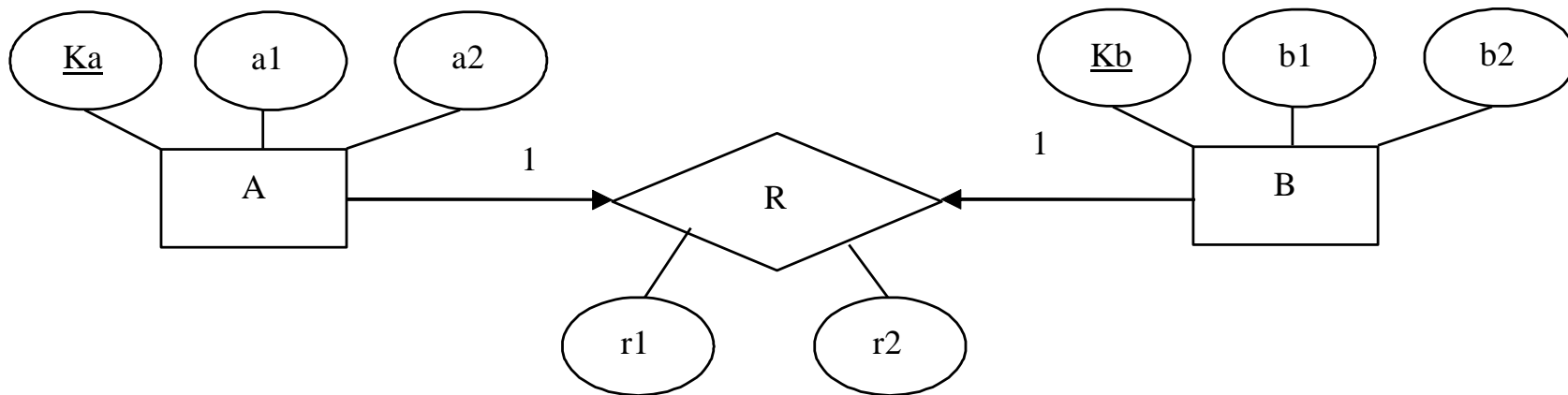
B (Kb,b1,b2), **Kb**为主键

R (Ka,Kb, r1, r2), **Ka**和**Kb**组合为主键, 另外**Ka**,**Kb**均为外键

7、自身联系的转化

需要单独建立一张关系表，用该实体的键作为**外键**，并反映出实体内部的联系。

1:1联系的转化



结果一（将描述属性移动到A实体，同时将Kb拷贝过去）：

A (Ka,a1,a2,r1,r2,Kb), **Ka为主键,Kb为外键**

B (Kb,b1,b2), **Kb为主键**

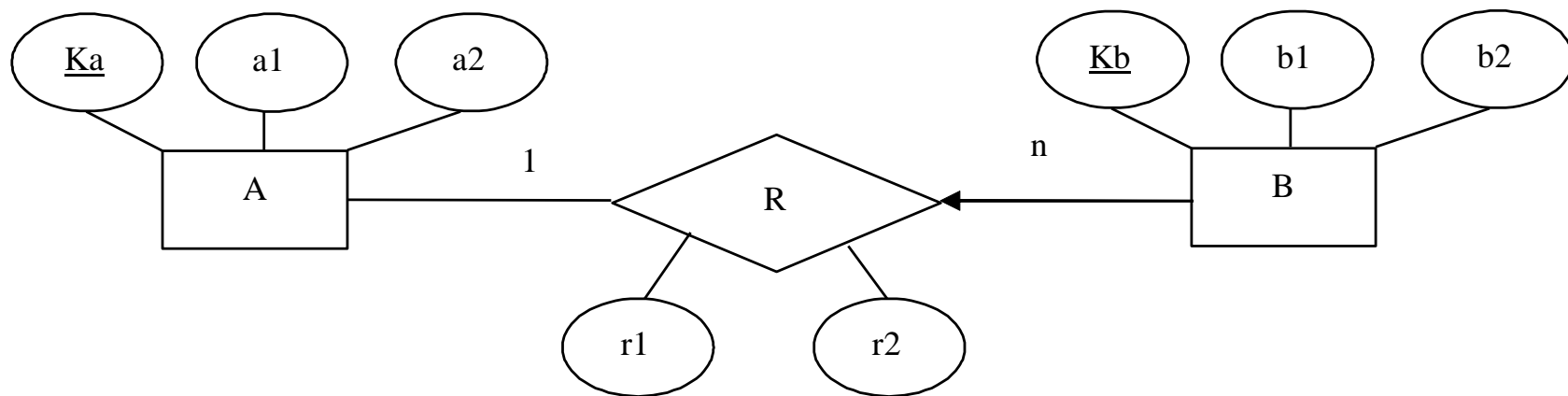
或：

结果二（将描述属性移动到B实体，同时将Ka拷贝过去）：

A (Ka,a1,a2), **Ka为主键**

B (Kb,b1,b2,r1,r2,Ka), **Kb为主键,Ka为外键**

1:n联系的转化

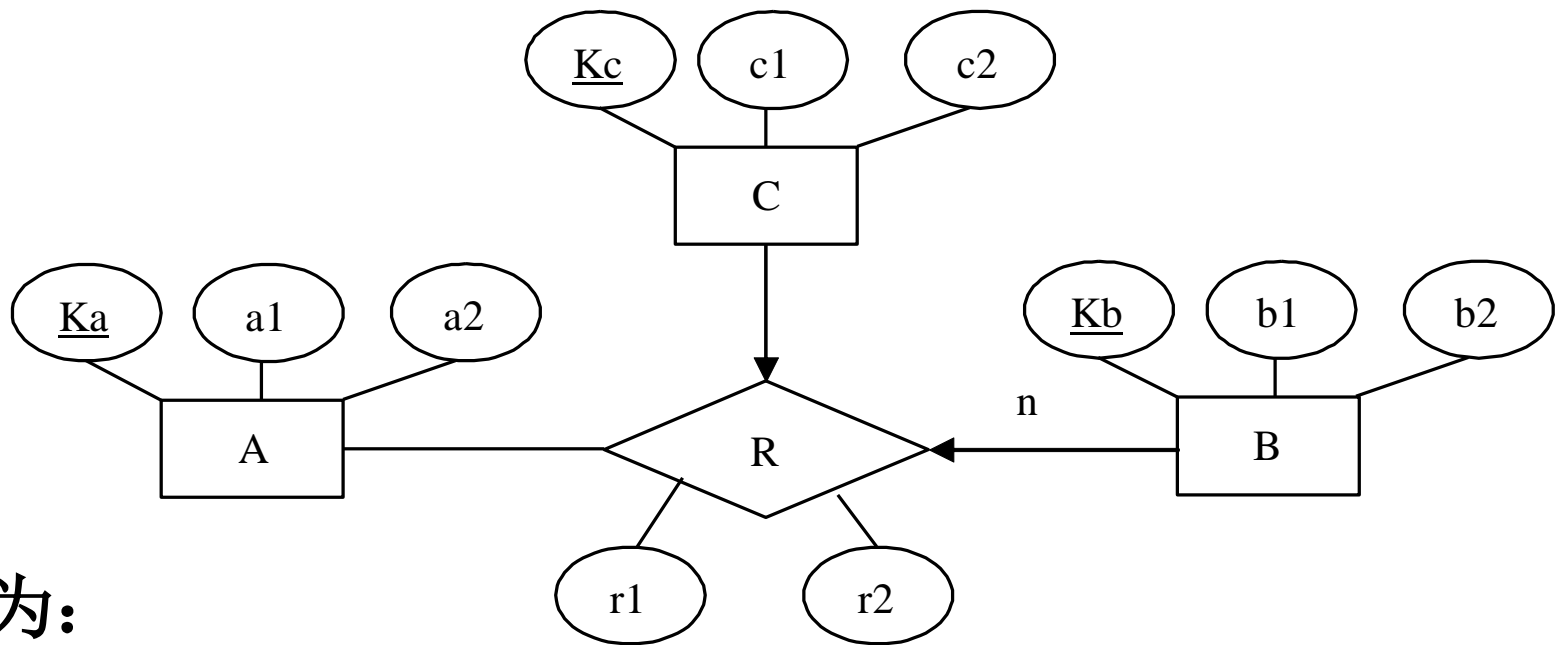


结果为（联系的属性只能向B方移动）：

A (Ka,a1,a2), **Ka为主键**

B (Kb,b1,b2,r1,r2,Ka), **Kb为主键, Ka为外键**

ER模型向关系模型转换示例



转换结果为:

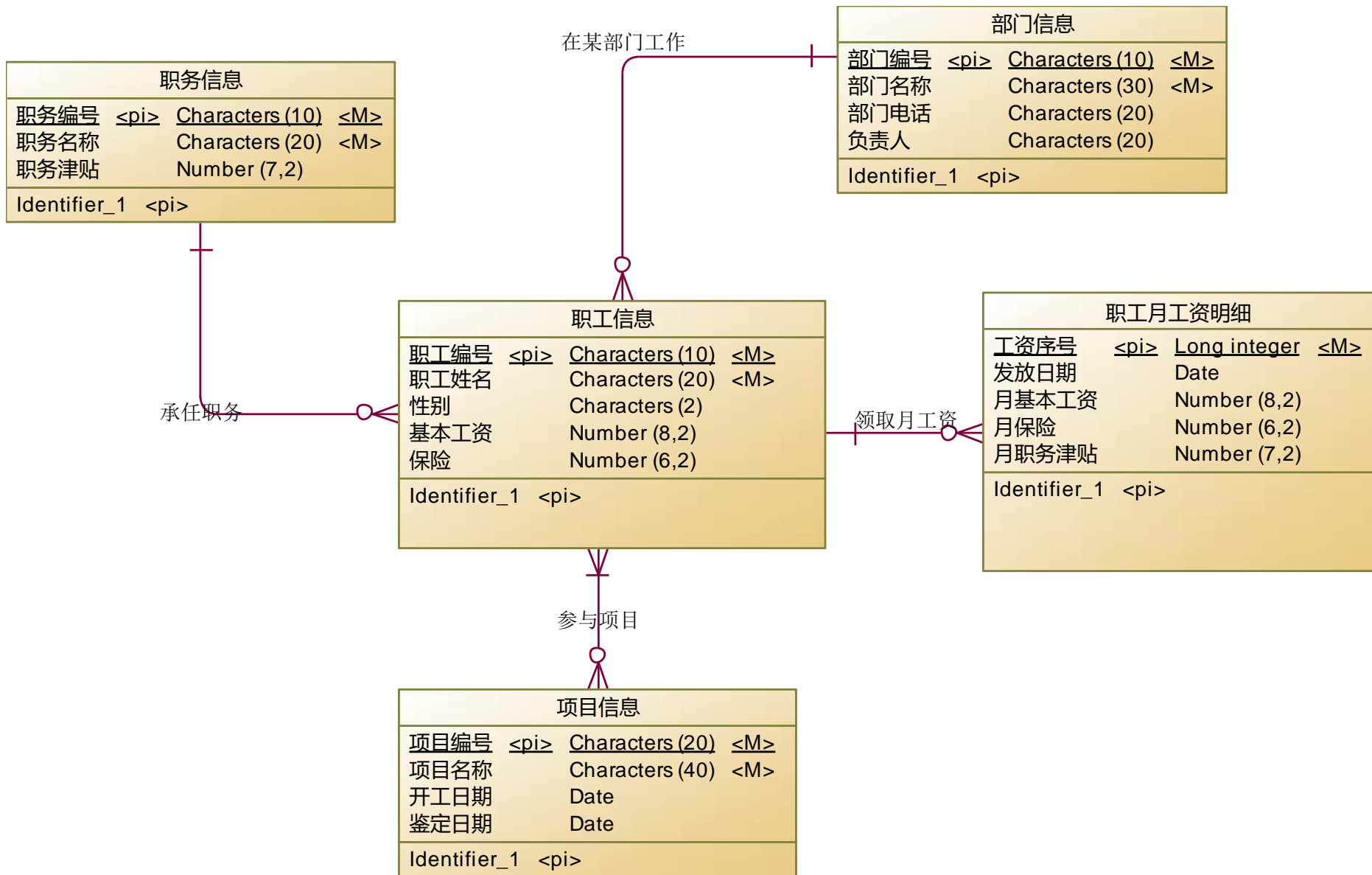
A (Ka,a1,a2), **Ka**为主键

B (Kb,b1,b2), **Kb**为主键

C (Kc,c1,c2), **Kc**为主键

R (Ka, Kb, Kc, r1, r2), 设**Kb**为主键, 则**Kc**为候选键, **Ka**, **Kb**和**Kc**均为外键

改进后 E-R 模型



通过手工转化的关系模式如下

由实体转化的关系表

职务（职务代号,名称,津贴）

部门（部门号,名称,电话,负责人）

职工（职工号,姓名,性别,出生日期,工资号,基本工资,保险）

项目（项目号,名称,起始日期,鉴定日期）

月工资明细（工资序号,发放日期,月基本工资,月保险,月津贴）

由联系转化的关系表

担任（职工号,职务代号,担任日期）

分工（职工号,部门号）

参与（职工号,项目号）

领取工资（职工号,工资序号）

对1:1和1:n联系合并后的关系模式

职务（职务代号,名称,津贴）

部门（部门号,名称,电话,负责人）

职工（职工号,姓名,性别,出生日期,工资号,基本工资,保险,
职务代号,担任日期,部门号）

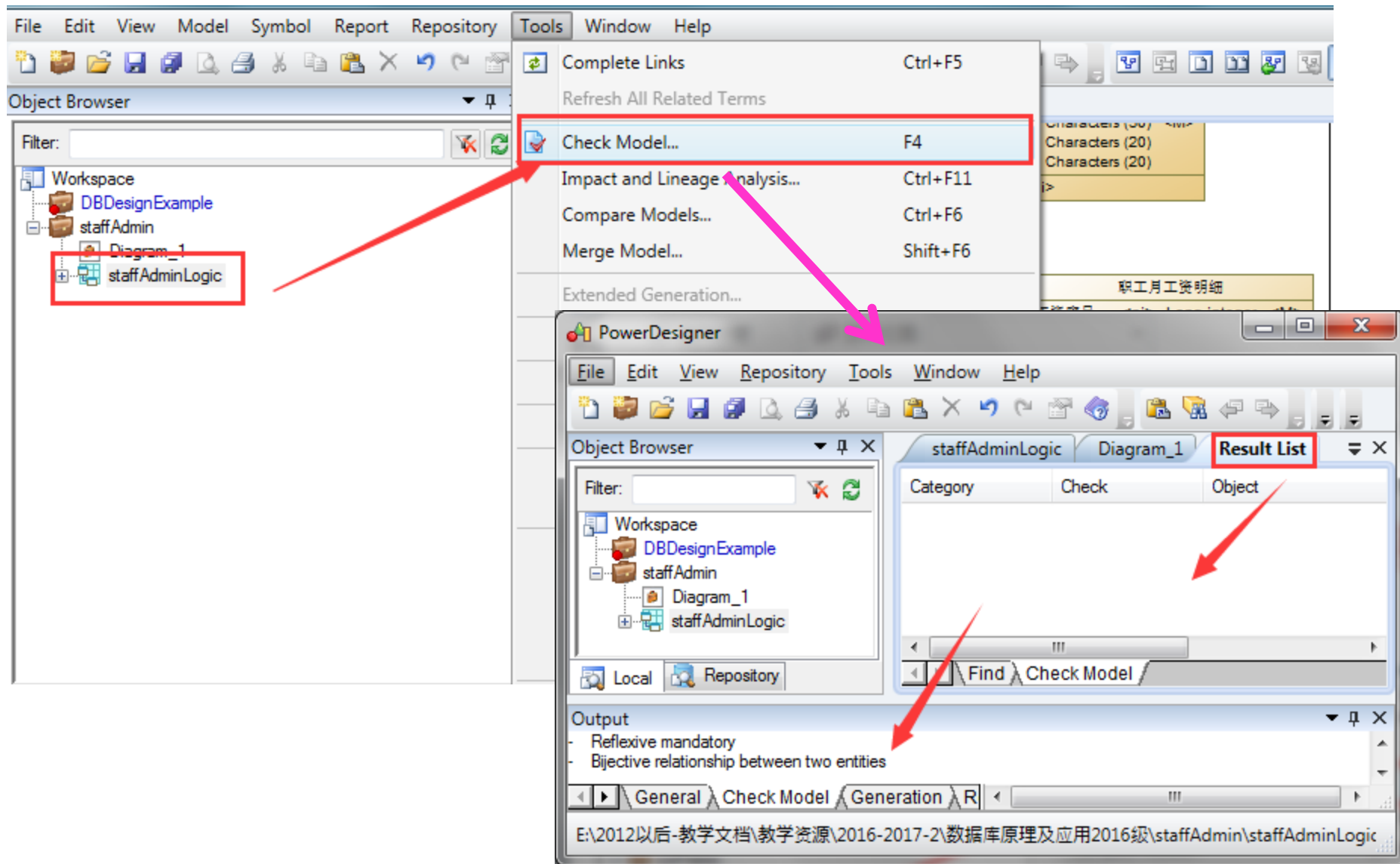
月工资明细（工资序号,发放日期,月基本工资,月保险,月津贴,职工号）

项目（项目号,名称,起始日期,鉴定日期）

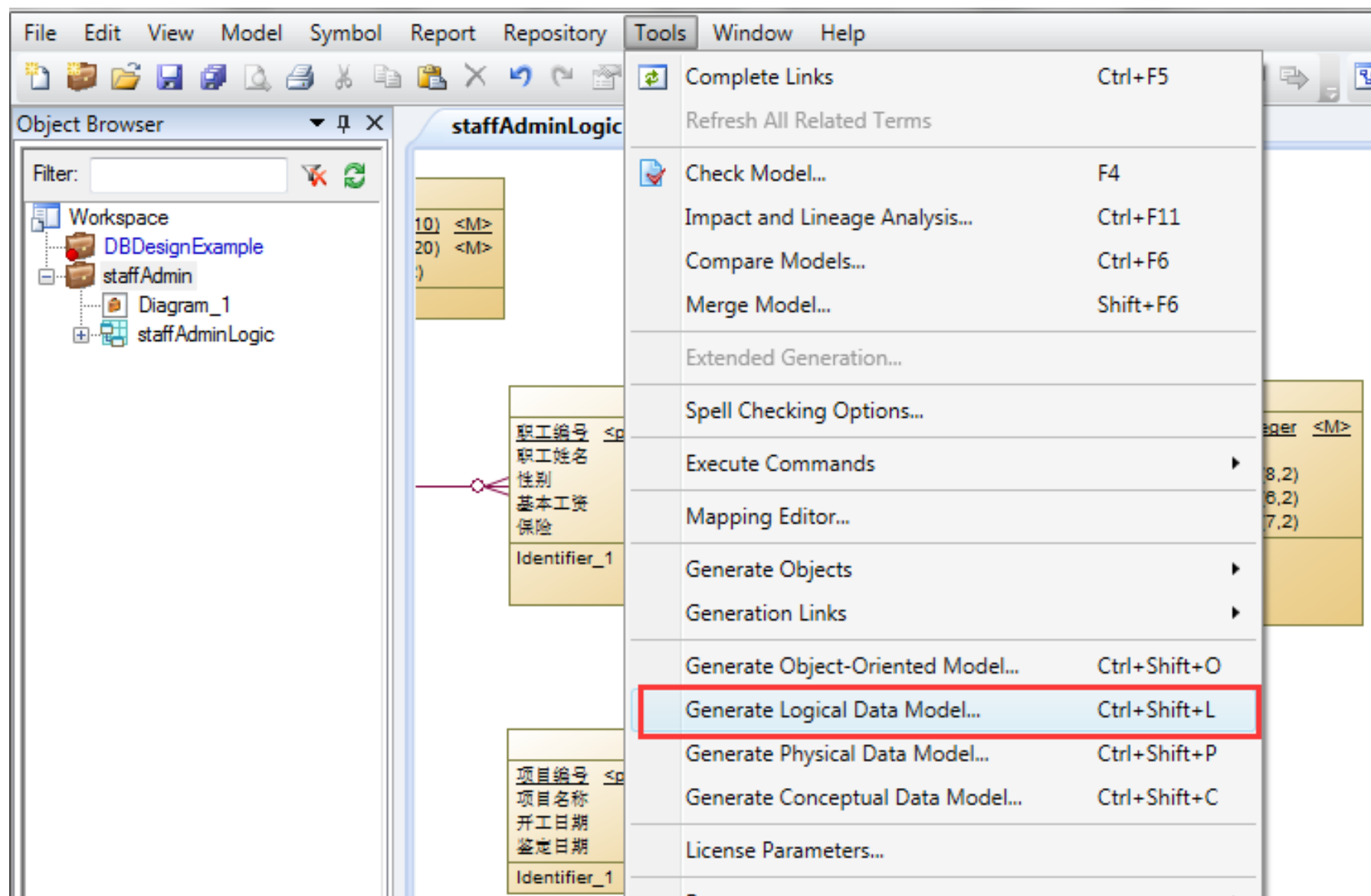
参与（职工号,项目号）

通过PowerDesigner将概念模型转化为逻辑模型

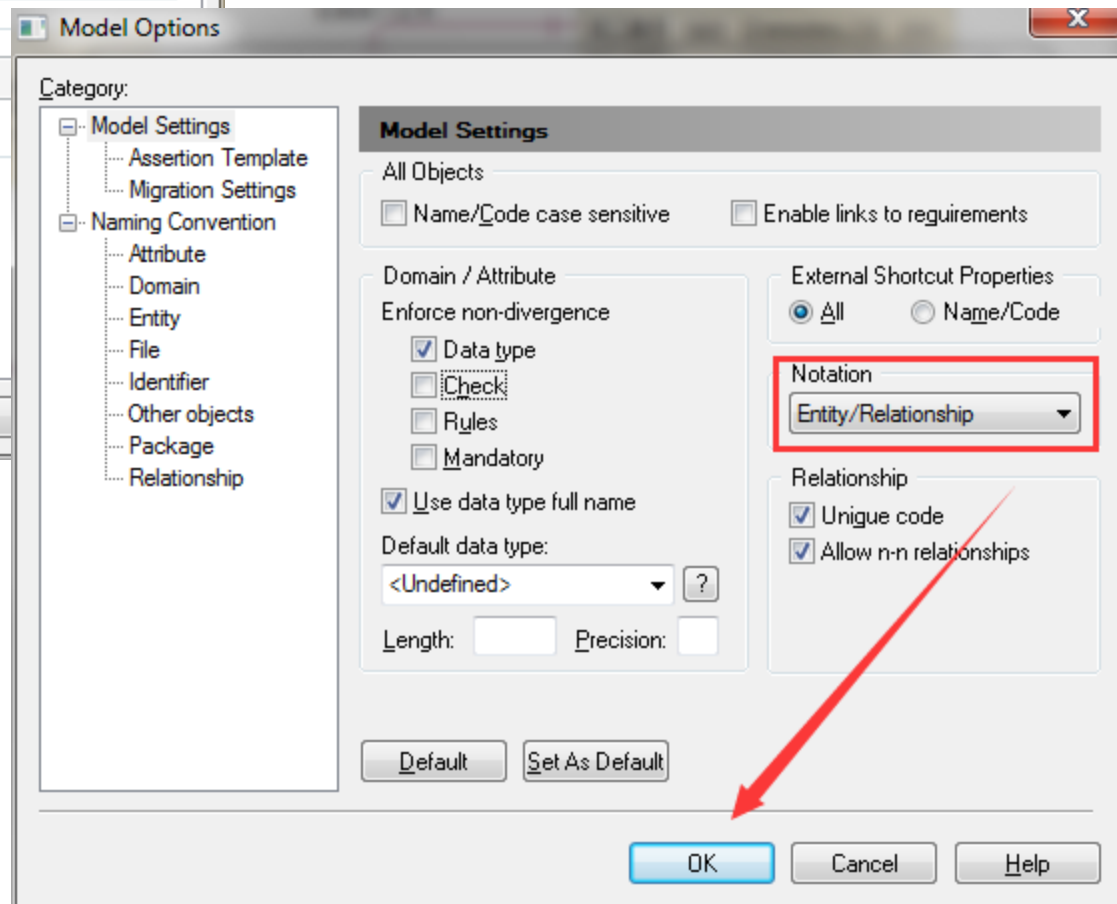
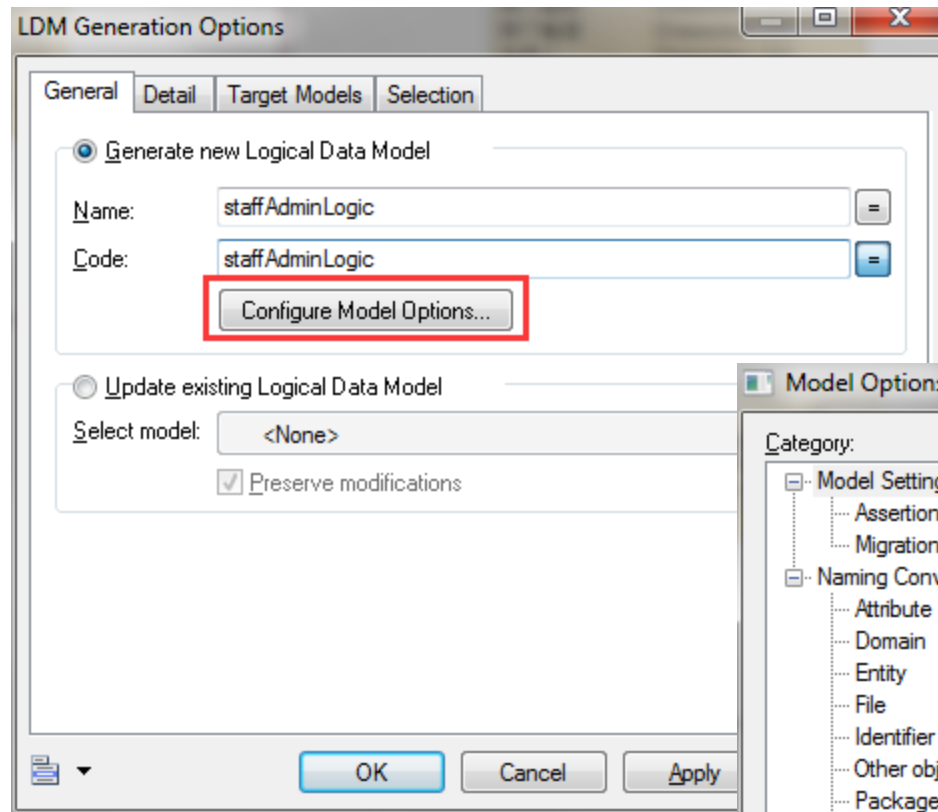
检查模型的正确性



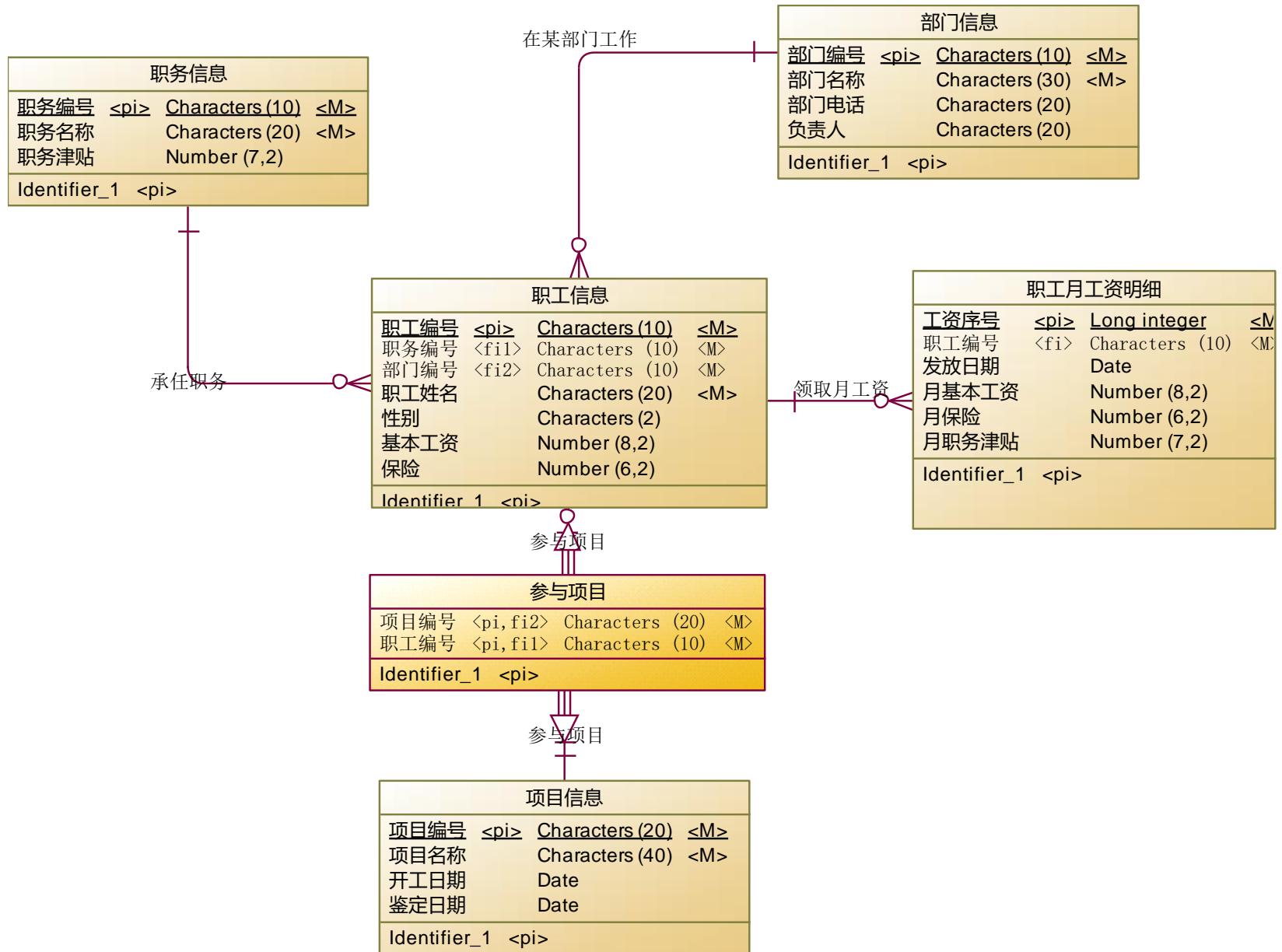
通过PowerDesigner将概念模型转化为逻辑模型



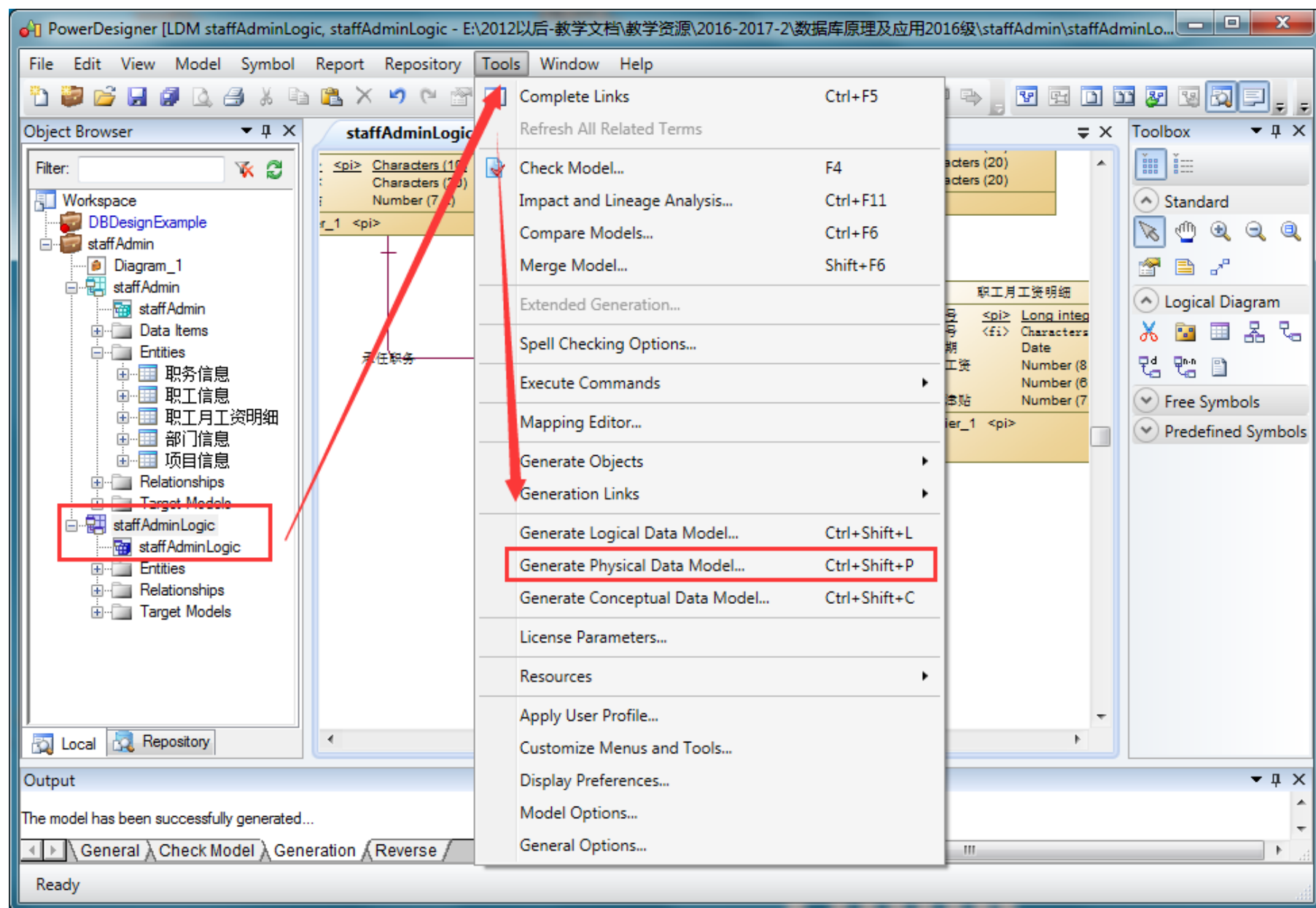
通过PowerDesigner将概念模型转化为逻辑模型



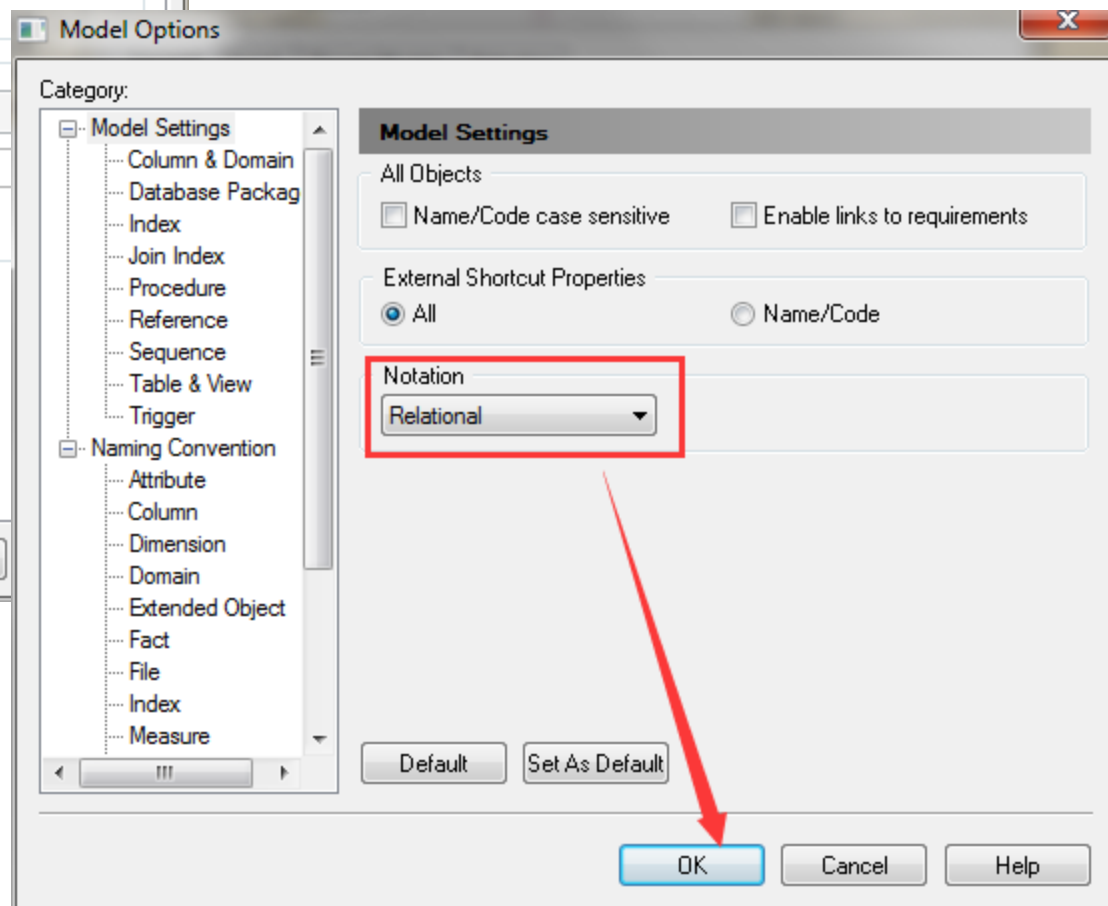
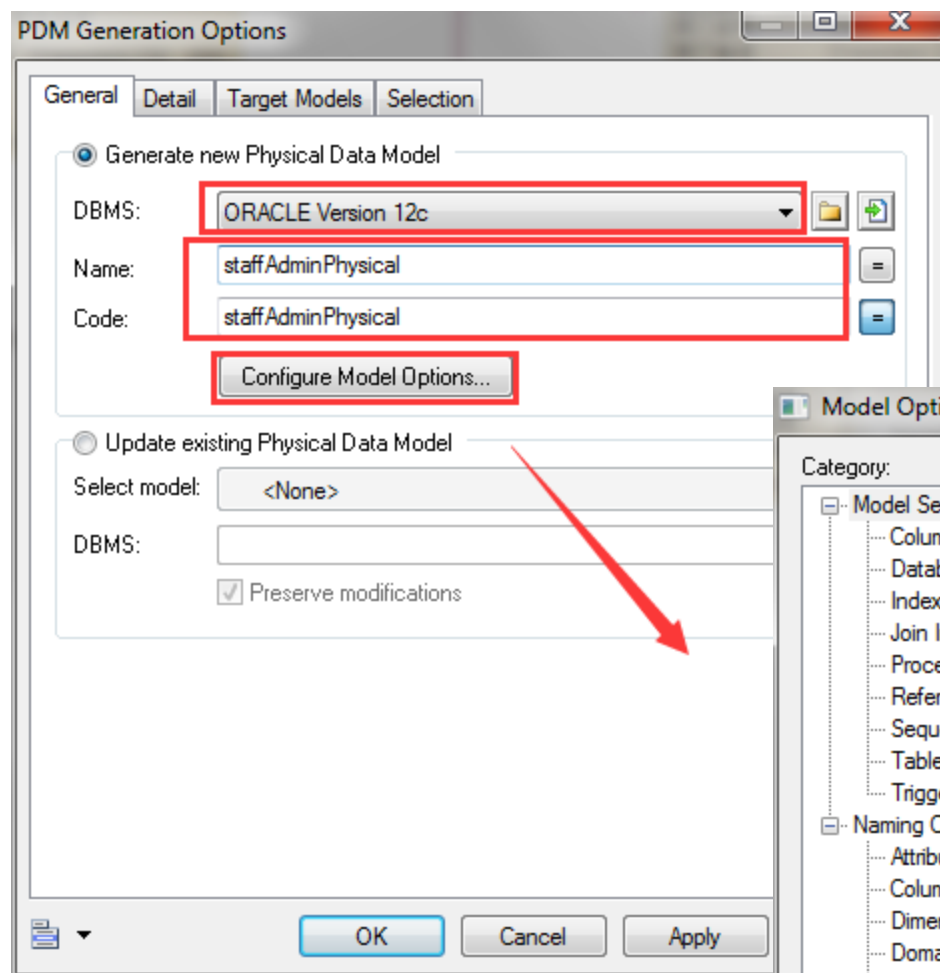
通过PowerDesigner转化的逻辑模型



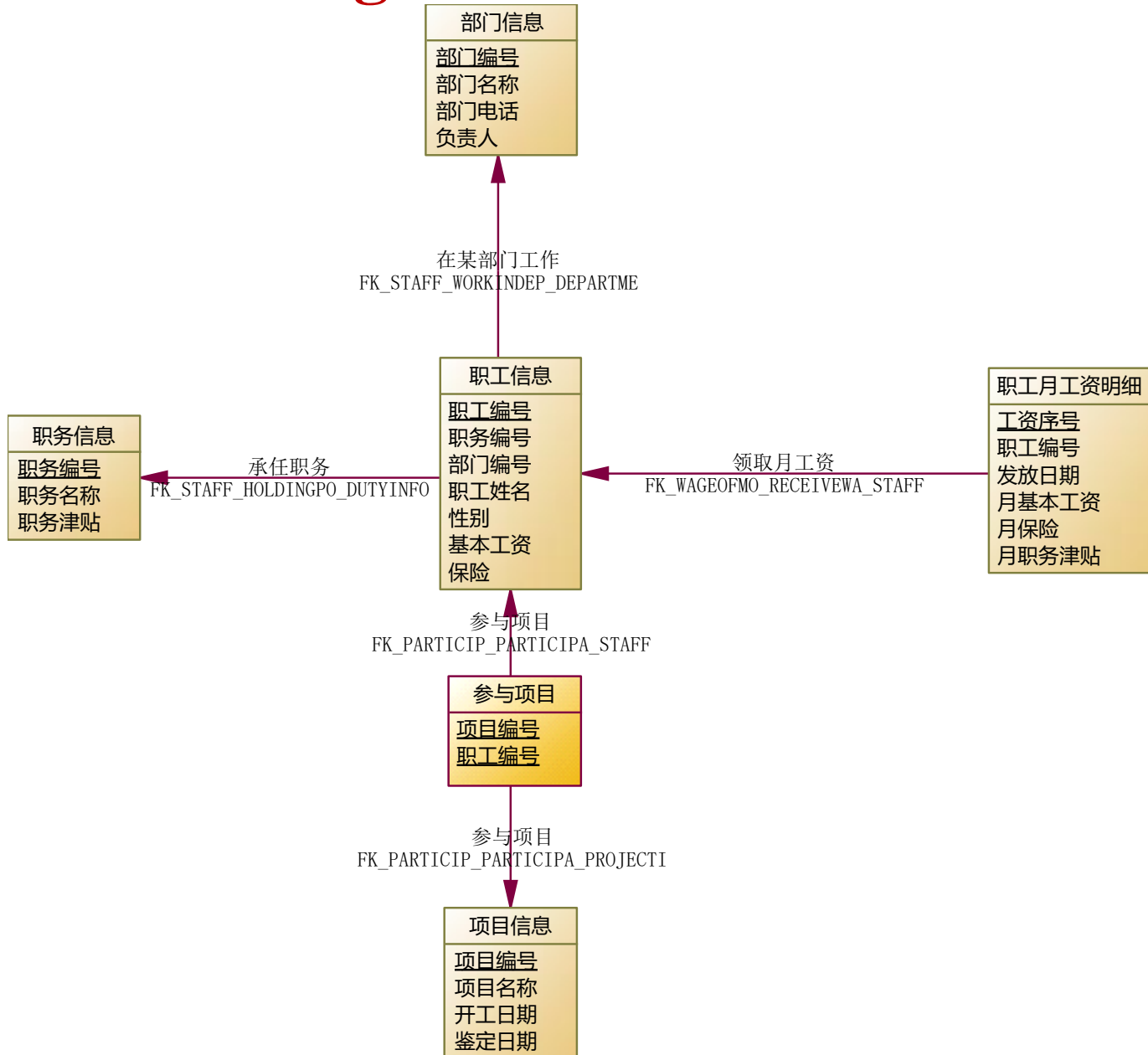
通过PowerDesigner转化物理模型



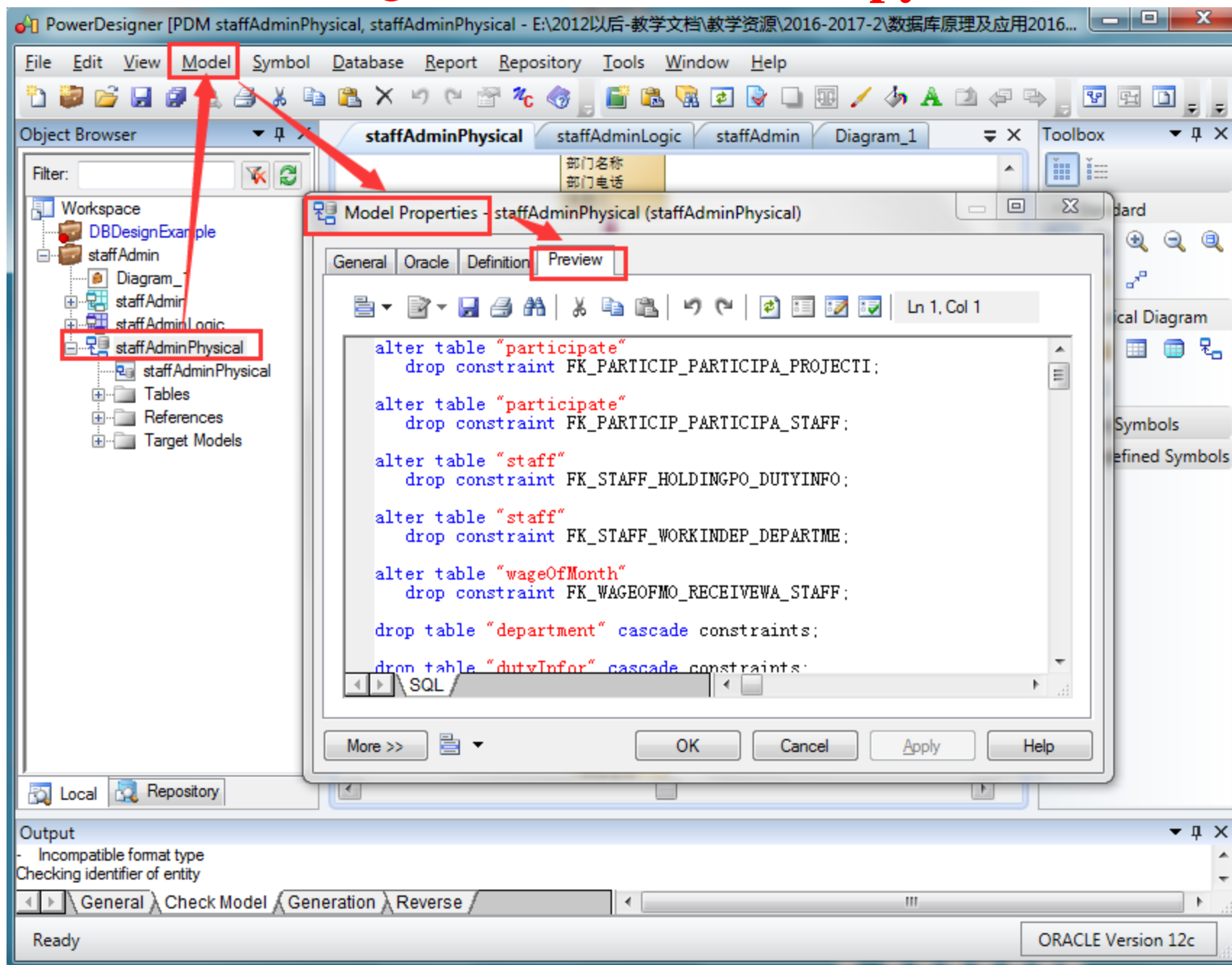
通过PowerDesigner转化物理模型



通过PowerDesigner转化的物理模型如下



由物理模型生成SQL语句如下，可以copy到文本编辑器。



为什么要对关系模式进行优化？

如何对关系模式进行优化？

一、实例-假设有如下表

教师学生关系模式

Tid	Tname	Dlevel	Course	Sid	Sname	Semester	Score
101	罗晓	教授	编译技术	S201	张珍	2010-2011-2	90
102	杨勋	副教授	数据库	S201	张珍	2010-2011-1	80
102	杨勋	副教授	数据库	S202	刘景	2010-2011-1	85
102	杨勋	副教授	数据库	S203	张柳	2010-2011-1	83
103	邓英超	讲师	C语言	S204	李秀	2008-2009-1	88
101	罗晓	教授	编译技术	S205	傅伟相	2010-2011-2	70

- 假设数据语义：
- (1) 教师可以在不同学期上同一门课程；
 - (2) 一个教师可为多位学生上课，而一个学生可选多门课程；
 - (3) 同一门课，一个学生在某学期只能选一个教师。

根据上述语义（**Tid, Course, Sid, Semester**）作为该模式的主键

假设教师、学生、课程信息没有和其它表中存储

上述教师学生模式可能存在的问题

(1) 插入异常(Insert Anomaly)

如果教师新来工作，由于还没排课，学生为空，由于主键属性不能为null，导致而不能插入教师信息

(2) 删除异常>Delete Anomaly)

- ① 删除时删掉了其他信息；
- ② 删除一个元组却删除了多个元组。

(3) 冗余(Redundancy)

表现： ① 某种信息在关系中存储多次；

(4) 更新异常(Update Anomaly)

表现： ① 更新一条记录却要求更新多个记录。

解决方案:

除了1:1联系的实体可以包含在一个表中, 其它实体应或联系单独建立关系表中。

将上表中的两个实体及联系分解, 形成三张关系表。

教师信息

Tid	Tname	Dlevel
101	罗晓	教授
102	杨勋	副教授
103	邓英超	讲师

学生课程信息

学生信息

Sid	Sname
S201	张珍
S202	刘景
S203	张柳
S204	李秀
S205	傅伟相

Tid	Course	Sid	Semester	Score
101	编译技术	S201	2010-2011-2	90
102	数据库	S201	2010-2011-1	80
102	数据库	S202	2010-2011-1	85
102	数据库	S203	2010-2011-1	83
103	C语言	S204	2008-2009-1	88
101	编译技术	S205	2010-2011-2	70

分析为何存在这些问题

数据语义在关系模式中的体现

具体表现：在关系模式的属性间的依赖关系，此即数据依赖。

数据依赖（Data Dependency）：指通过关系模式某些属性的取值能够决定另一些属性的取值。

数据依赖分类：函数依赖、多值依赖和连接依赖

数据依赖决定因素：由现实系统中属性间相互联系的语义决定。

异常现象产生的根源：关系模式中属性间存在的这些依赖关系。

根源的体现及解决：一般来讲，关系必须含有主键和候选键。主键值决定其他属性值，候选键的值不能重复。如果将各种数据集中于一个模式中，一般都会造成异常。**解决异常的方法，是利用规范化理论，对关系模式进行相应的分解，以消除这些异常。**

规范化就是对所有的属性进行重新组合，使关系的结构更简洁、更规范。

规范化的目的是：

优化关系模式，提高数据管理的效率。

三、规范化的几个概念

1、属性的几个概念

(1) 简单属性和复合属性

关系模型只支持简单属性。

简单属性	复合属性	不支持！			
职工号	姓名	工资			
		基本工资	津贴	保险	实发工资

(2) 单值属性和多值属性

关系模型只支持单值属性。

单值属性	多值属性	不支持！	
学号	姓名	选修课程	
99101	张三	计算机网路	
		数据库原理	
		机械制图	
99102	李四	计算机网络	
		嵌入式系统	
99103	王五	计算机网络	

(3) 基本属性和导出属性

如出生日期和年龄；

基本工资、津贴、保险和实发工资，等等。

(4) 属性之间的联系

- **1 : 1**

如 学号和联系电话。

- **1 : n**

如 班号和学号。

- **m : n**

如 学号和课程号。

2、键的几个概念

- 单键

由一个属性组成的键称为**单键**。

- 多键

由关系表中的多个属性组成的键称为**多键**。

- 全键

由关系表中的全部属性组成的键称为**全键**。

3、函数依赖

函数依赖是属性之间的约束关系。

定义：设X、Y是关系表R的属性（组），如果对于R的所有元组都有：X的每一具体值都只有一个Y的值与之对应，则称X**函数决定**Y，或Y**函数依赖**于X，记作 $X \rightarrow Y$ 。

换句话说，如果知道了X的值，就可以在表中确定与之对应的Y的值（只有一个）。

- 若Y不函数依赖于X，记作 $X \nrightarrow Y$ 。
- 若 $X \rightarrow Y$ ，且 $Y \rightarrow X$ ，记作 $X \leftrightarrow Y$ 。

函数依赖等价定义

假设 R 是一关系模式， U 是 R 的属性集合， $X、Y \subseteq U$ ， r 是 R 的一个关系实例，元组 $t \in R$ 。则用 $t[X]$ 表示元组 t 在属性集合 X 上的值。 XY 表示 X 和 Y 的并集。

函数依赖定义： 设 R 是一个关系模式， U 是 R 的属性集合， X 和 Y 是 U 的子集。对于 R 的任意实例 r ， r 中任意两个元组 t_1 和 t_2 ，如果 $t_1[X] = t_2[X]$ 则 $t_1[Y] = t_2[Y]$ ，那么称 X 函数地确定 Y ，或 Y 函数地依赖于 X ，记作： $X \rightarrow Y$ ， X 称为决定子(Determinant)。

函数依赖关心的问题： 是一个或一组属性的值决定其他属性的值。

$U=\{X1,X2,X3,Y1,Y2,Z\}$ $X=\{X1,X2,X3\}$ $Y=\{Y1,Y2\}$

	X1	X2	X3	Y1	Y2	Z
t_1	a1	b1	c1	d1	e1	...

t_2	a1	b1	c1	d1	e1	...
t_3	a2	b2	c2	d2	e2	...

t_4	a2	b2	c2	d2	e2	...

$$Y=F(X)$$

其中: $X=[x1,x2,x3]$, $Y=[Y1,Y2]$

例

学号	姓名	班号	学院	课程号	课程名	联系电话
14101	张三	99141	自动化	A101	高等数学	136xxxxxxxx
14101	张三	99141	自动化	A204	计算机网络	136xxxxxxxx
14122	李四	99141	自动化	A101	高等数学	130xxxxxxxx
14213	王五	99142	自动化	A101	高等数学	138xxxxxxxx
08113	赵六	99081	计算机	A107	大学物理	137xxxxxxxx
08218	钱七	99082	计算机	A204	计算机网络	133xxxxxxxx

如果X、Y是 1 : 1 的联系，则 $X \leftrightarrow Y$ 。

如 学号 \leftrightarrow 联系电话，即知道了学号，就可以在表中确定其联系电话；同样地，知道了联系电话，也可以在表中确定其学号。

如果X、Y是 n : 1 的联系，则 $X \rightarrow Y$ 。

如 学号 \rightarrow 班号，即知道了学号，就可以在表中确定其班号；相反地，如果知道了班号，却无法确定学号。

如果X、Y是 m : n 的联系，则X和Y不存在函数依赖关系。

如 学号和课程号没有函数依赖关系。即知道了学号，无法在表中确定课程号；同样地，如果知道了课程号，也无法确定学号。

学号	姓名	班号	学院	课程号	课程名	联系电话
14101	张三	99141	自动化	A101	高等数学	136xxxxxxxx
14101	张三	99141	自动化	A204	计算机网络	136xxxxxxxx
14122	李四	99141	自动化	A101	高等数学	130xxxxxxxx
14213	王五	99142	自动化	A101	高等数学	138xxxxxxxx
08113	赵六	99081	计算机	A107	大学物理	137xxxxxxxx
08218	钱七	99082	计算机	A204	计算机网络	133xxxxxxxx

问题：班号和学院是否有函数依赖关系？

课程号和课程名是否有函数依赖关系？

课程号和联系电话是否有函数依赖关系？

(1) 平凡函数依赖和非平凡函数依赖

定义：设 X 、 Y 是关系表 R 的属性（组），且 $X \rightarrow Y$ ，若 $Y \subseteq X$ ，则称为**平凡依赖**，否则称为**非平凡依赖**。

如 $(\text{学号}, \text{姓名}) \rightarrow \text{姓名}$ ，而 $\text{姓名} \subseteq (\text{学号}, \text{姓名})$ ，因此，这就是**平凡依赖**。即知道了学号、姓名，就可以确定姓名，这是再平凡不过的道理。

(2) 完全函数依赖和部分函数依赖

定义：设 X 、 Y 是关系表 R 的属性（组），且 $X \rightarrow Y$ ，若 X 存在某个子集 X_1 ，使 $X_1 \rightarrow Y$ 成立，则称 Y **部分依赖**于 X ，否则称 Y **完全依赖**于 X 。

如 $(\text{学号}, \text{姓名}) \rightarrow \text{班号}$ ，而 $\text{学号} \rightarrow \text{班号}$ ，因此，班号 **部分依赖**于 $(\text{学号}, \text{姓名})$ 。

学号	姓名	班号	学院	课程号	课程名	联系电话
14101	张三	99141	自动化	A101	高等数学	136xxxxxxxx
14101	张三	99141	自动化	A204	计算机网络	136xxxxxxxx
14122	李四	99141	自动化	A101	高等数学	130xxxxxxxx
14213	王五	99142	自动化	A101	高等数学	138xxxxxxxx
08113	赵六	99081	计算机	A107	大学物理	137xxxxxxxx
08218	钱七	99082	计算机	A204	计算机网络	133xxxxxxxx

(3) 传递函数依赖和非传递函数依赖

定义：设 X 、 Y 、 Z 是关系表 R 的属性（组），若 $X \rightarrow Y$ ， $Y \rightarrow Z$ ，且 $Y \not\rightarrow X$ ， Y 不属于 X ，则称 X **传递决定** Z ，或 Z **传递依赖于** X ，否则称 Z **非传递依赖于** X 。

如 学号 \rightarrow 班号，班号 \rightarrow 学院，因此，学院**传递依赖于**学号，或 学号**传递决定**学院。

学号	姓名	班号	学院	课程号	课程名	联系电话
14101	张三	99141	自动化	A101	高等数学	136xxxxxxxx
14101	张三	99141	自动化	A204	计算机网络	136xxxxxxxx
14122	李四	99141	自动化	A101	高等数学	130xxxxxxxx
14213	王五	99142	自动化	A101	高等数学	138xxxxxxxx
08113	赵六	99081	计算机	A107	大学物理	137xxxxxxxx
08218	钱七	99082	计算机	A204	计算机网络	133xxxxxxxx

Armstrong公理系统

问题提出：在关系模式的规范化处理过程中，不仅要知道一个给定的函数依赖集合，还要知道由给定的函数依赖集合所蕴涵（或**推导出**）的所有函数依赖的集合。为此，需要有效而完备的公理系统，Armstrong公理系统即是这样的系统。

Armstrong公理：为从已知的函数依赖推导出其他的函数依赖，Armstrong提出了一套推理规则，称为Armstrong公理(**Armstrong's Axioms**)。

Armstrong公理包含如下三条推理规则：

- (1) 自反律(**Reflexivity**)：若 $Y \subseteq X \subseteq U$ ，则 $X \rightarrow Y$ 。
- (2) 增广律(**Augmentation**)：若 $X \rightarrow Y$ ， $Z \subseteq U$ ，则 $XZ \rightarrow YZ$ 。
- (3) 传递律(**Transitivity**)：若 $X \rightarrow Y$ 和 $Y \rightarrow Z$ ，则 $X \rightarrow Z$ 。

引理 1： Armstrong公理是正确的，即由已知函数依赖，根据Armstrong公理所推导的函数依赖总是成立的。

引理 2： 如下三条推理规则是正确的：

- (1) 合并规则(**Union**)：如果 $X \rightarrow Y$ ， $X \rightarrow Z$ ，则 $X \rightarrow YZ$ 。
- (2) 伪传递规则(**Pseudo Transitivity**)：

如果 $X \rightarrow Y$ ， $YW \rightarrow Z$ ，则 $XW \rightarrow Z$ 。

(3) 分解规则(**Decomposition**)：如果 $X \rightarrow Y$ ， $Z \subseteq Y$ ，则 $X \rightarrow Z$ 。
或：如 $X \rightarrow YZ$ ，则 $X \rightarrow Y$ ， $X \rightarrow Z$ 。

多值依赖 (MultiValued Dependency, 缩写为MVD)

设 $R(U)$ 是属性集 U 上的关系模式, X, Y, Z 是 U 的子集, 且 $Z = U - X - Y$, 多值依赖 $X \twoheadrightarrow Y$ 成立当且仅当对 $R(U)$ 的任一关系 r , 任给的一对 (x, z) 值有一组 Y 的值, 这组值仅仅取决于 x 值而与 z 值无关。

称 X 多值决定 Y 或 Y 多值依赖于 X 。

例如, 在关系模式 $TEACH$ 中有 $C \twoheadrightarrow T$

直观上看, 若 $X \twoheadrightarrow Y$, 则 X 的一个值唯一决定一组 Y 值, 且这组值与 X, Y 之外的属性值无关

课程C	教员T	参考书B
物理	李洋	普通物理学
物理	李洋	光学原理
物理	李洋	物理习题集
物理	王海	普通物理学
物理	王海	光学原理
物理	王海	物理习题集
数学	王海	数学分析
数学	王海	微分方程
数学	王海	高等代数
数学	白云	数学分析
数学	白云	数学分析
.....

多值依赖的另一等价定义：

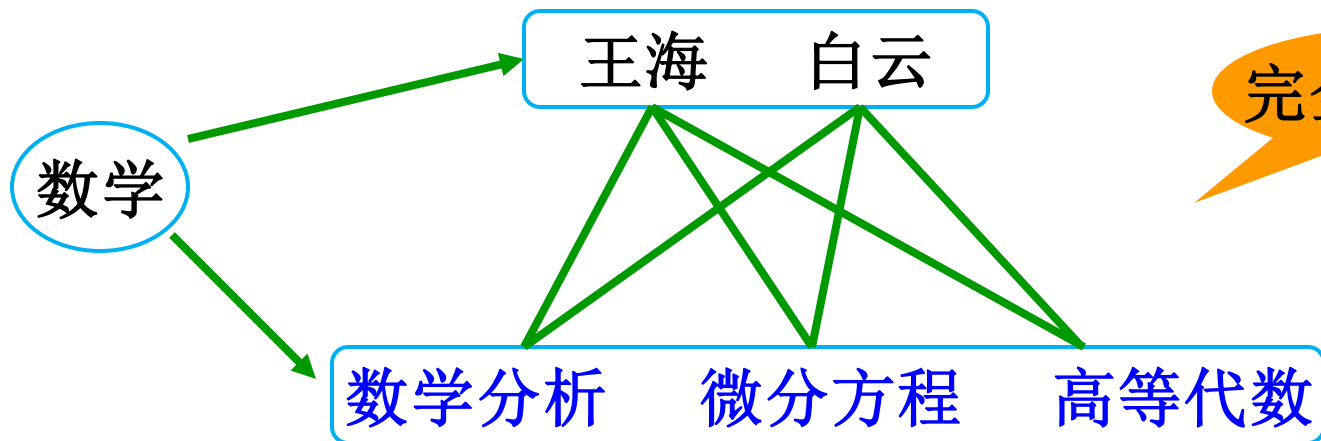
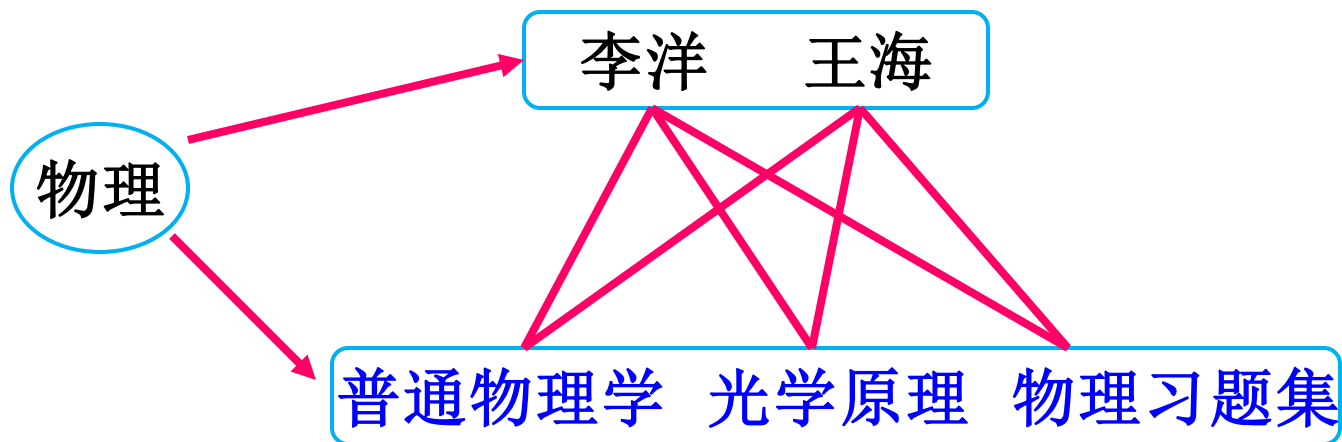
多值依赖 $X \twoheadrightarrow Y$ 成立当且仅当对 $R(U)$ 的任一关系 r ，若存在元组 s 、 t 使得 $s[X]=t[X]$ ，则必存在元组 w 、 $v \in r$ （ w 、 v 可以与 s 、 t 相同），使得 $w[X]=v[X]=t[X]$ ，而 $w[Y]=t[Y]$ ， $w[Z]=s[Z]$ ， $v[Y]=s[Y]$ ， $v[Z]=t[Z]$ 。

	X	Y	Z
s		s[Y]	s[Z]
t		t[Y]	t[Z]
w		w[Y]	w[Z]
v		v[Y]	v[Z]

交换 s 、 t 的Y值
所得新元组仍在 r 中

左图直观显示，
 x 决定一组 y 值，
这组值与 z 无关

由前面例子，可看出X、Y、Z之间有下列关系：



完全二分图

多值依赖的性质：

- (1) 对称性：若 $X \twoheadrightarrow Y$, $Z = U - X - Y$, 则 $X \twoheadrightarrow Z$ 。
- (2) 函数依赖可看成是多值依赖的特例：若 $X \rightarrow Y$, 则 $X \twoheadrightarrow Y$
- (3) 若 $U = XY$ (表示 $X \cup Y$) , 则 $X \twoheadrightarrow Y$ 显然成立。

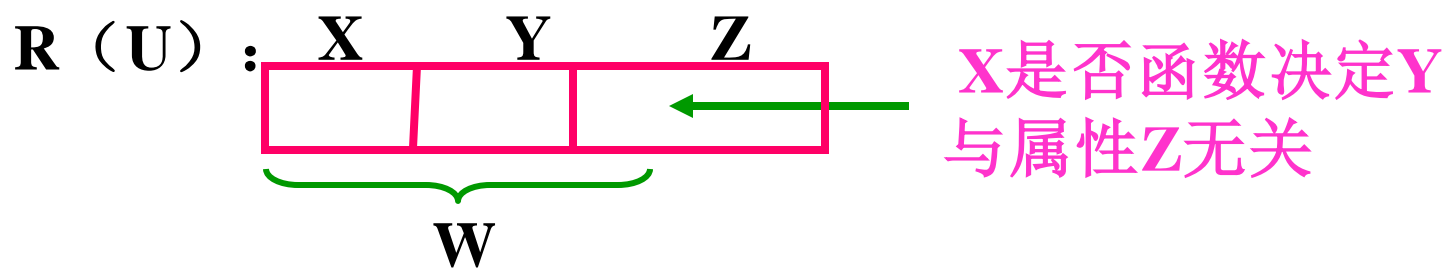
(这种多值依赖无任何实际意义, 故称为 平凡的多值依赖)

多值依赖与函数依赖的区别

- (1) 函数依赖 $X \rightarrow Y$ 的有效性仅取决于 X 、 Y , 与 X 、 Y 之外的属性无关:

$X \rightarrow Y$ 在 $\pi_{XY}(R)$ 上成立 $\iff X \rightarrow Y$ 在 $\pi_W(R)$ 上成立

其中 W 满足 $XY \subseteq W \subseteq U$ (U 是关系模式 R 的属性集)。

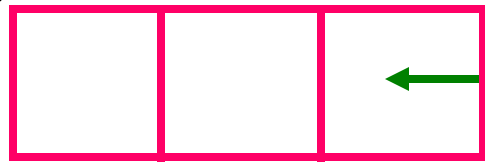


多值依赖 $X \twoheadrightarrow Y$ 的有效性与 X 、 Y 之外的属性范围有关：

若 $X \twoheadrightarrow Y$ 在 U 上成立，则在 W ($XY \subseteq W \subseteq U$) 上也成立，
但反之不然。

可缩小范围但不一定能扩大范围

$R(U) : \quad X \quad Y \quad Z$



X 是否多值决定 Y
与属性 Z 密切相关

W

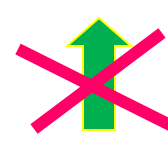
$X \twoheadrightarrow Y$ 在 U 上成立

$R(W) : \quad X \quad Y$



W

$X \twoheadrightarrow Y$ 在 W 上也成立

 反之
不成立

三、关系模式规范化理论

1、第一范式（1NF）

定义：如果关系表R不存在**复合属性**及**多值属性**，即：属性是不可再分，则称R满足第一范式，记作 $R \in 1NF$ 。

对于不满足1NF的表，其解决办法：

- ① 将复合属性用各子属性代替，称为**简单属性**
- ② 将含有多值属性的表分解成两张表，一张表由**主键和简单属性**构成，另外一张表由**多值属性和主键**。

例如：

职工号	姓名	工资			
		基本工资	津贴	保险	实发工资

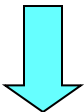
转化为



职工号	姓名	基本工资	津贴	保险	实发工资

学号	姓名	选修课程
99101	张三	计算机网路
		数据库原理
		机械制图
99102	李四	计算机网络
		嵌入式系统
99103	王五	计算机网络

转化为



学号	选修课程
99101	计算机网路
99101	数据库原理
99101	机械制图
99102	计算机网络
99102	嵌入式系统
99103	计算机网络

学号	姓名
99101	张三
99102	李四
99103	王五

2、第二范式（2NF）

定义：如果关系表R满足**1NF**，且所有非主属性都**完全依赖于**任一候选键，则称R满足第二范式，记作 $R \in 2NF$ 。

即R中不存在非主属性对键的**部分函数依赖**。

看看以下关系表是否满足2NF？

学号	姓名	班号	学院	课程号	课程名	成绩
14101	张三	99141	自动化	A101	高等数学	80
14101	张三	99141	自动化	A204	计算机网络	83
14122	李四	99141	自动化	A101	高等数学	91
14213	王五	99142	自动化	A101	高等数学	76
08113	赵六	99081	计算机	A107	大学物理	88
08218	钱七	99082	计算机	A204	计算机网络	69

其中：学号和课程号构成主键

如果不是2NF,如何解决？

INF转化为2NF

- ① 将部分依赖的非主属性和它所依赖的属性构成新的模式
- ② 将完全依赖的非主属性与候选键构成新模式

将上述的表分解为如下三张表

学号	姓名	班号	学院
14101	张三	99141	自动化
14122	李四	99141	自动化
14213	王五	99142	自动化
08113	赵六	99081	计算机
08218	钱七	99082	计算机

学号	课程号	成绩
14101	A101	80
14101	A204	83
14122	A101	91
14213	A101	76
08113	A107	88
08218	A204	69

课程号	课程名
A101	高等数学
A204	计算机网络
A107	大学物理

可以写成如下关系模式：

- (1) 学生表 (学号, 姓名, 班号, 学院)
- (2) 课程表 (课程号, 课程名)
- (3) 选课表 (学号, 课程号, 成绩)

3、第三范式（3NF）

定义：如果关系表R满足1NF，且所有非主属性都**非传递依赖于R**的任一候选键，则称R满足第三范式，记作 $R \in 3NF$ 。

即R中不存在非主属性对键的**传递函数依赖**。

推论：若R不存在非主属性，则一定满足3NF。

看看以下关系表是否满足3NF？

学号	姓名	班号	学院
14101	张三	99141	自动化
14122	李四	99141	自动化
14213	王五	99142	自动化
08113	赵六	99081	计算机
08218	钱七	99082	计算机

如果不是，如何解决？

如何将1NF转化3NF

- ① 如果不满足2NF，则先按照前面方法转化2NF，然后继续下一步；
- ② 如果满足2NF，则将传递依赖的属性及其中间属性移出构成新表；将不存在传递依赖的属性及候选键构成新表。

将上述的表分解为如下两张表

学号	姓名	班号
14101	张三	99141
14122	李四	99141
14213	王五	99142
08113	赵六	99081
08218	钱七	99082

班号	学院
99141	自动化
99142	自动化
99081	计算机
99082	计算机

可以写成如下关系模式：

- (1) 学生表 (学号, 姓名, 班号)
- (2) 班级表 (班号, 学院)

4、改进的3NF (BCNF)

定义： 如果关系表R满足1NF，且R的任一函数依赖关系的**左部**都是R的一个**候选键**，则称R满足BCNF，记作 $R \in \text{BCNF}$ 。

即R中不存在**主属性**对键的**传递函数依赖或部分依赖**。

推论1： R中所有**非主属性**对所有键都是**完全依赖**。

推论2： R中所有主属性对不包含它们的键都是完全依赖。

推论3： R中没有哪个属性完全依赖于**非键属性**。

定理： 若R满足BCNF，则一定满足3NF，但满足3NF并不一定就满足BCNF。

BCNF范式示例

- 假设一个教师只能讲一门课程，一门课有多个教师讲，一个学生可以选多门不同的课

学生	教师	课程
丁一	袁老师	数据库
马二	李老师	计算机网络
马二	袁老师	数据库
张三	王老师	计算机网络
张三	赵老师	数据库
李四	袁老师	数据库
王五	李老师	计算机网络

候选键有：（学生，教师）

（学生，教师）→课程

教师→课程

如果不是BCNF，如何解决？

BCNF范式的规范化

- ① 如果不满足3NF，则先转化为3NF，然后继续下一步；
- ② 如果满足3NF，则将部分依赖的主属性和它所依赖的主属性构成新表；然后将左端的候选键构成新表。

将上述的表分解为如下两张表

学生	教师
丁一	袁老师
马二	李老师
马二	袁老师
张三	王老师
张三	赵老师
李四	袁老师
王五	李老师

教师	课程
袁老师	数据库
李老师	计算机网络
王老师	计算机网络
赵老师	数据库

可以写成如下关系模式：

- (1) 学生表 (学生, 教师)
- (2) 授课表 (教师, 课程)

思考：能按照下列两种分解吗？

- (1) 选课表 (学生, 课程)
- (2) 授课表 (教师, 课程)

或者：(3) 选课表 (学生, 课程)
(4) 师生表 (学生, 教师)

5、第四范式（4NF）

定义：如果关系表R满足第一范式，且R的任一非平凡的多值依赖 $X \twoheadrightarrow Y$ （X不包含Y），X含有键，则称R满足第四范式，记作 $R \in 4NF$ 。

若 $R \in 4NF$ ，则必有 $R \in BCNF$ 。

若 $R \in BCNF$ ，则不一定有 $R \in 4NF$ 。

若R中没有**非平凡多值依赖**，则必有 $R \in 4NF$ 。

看看以下关系表是否满足4NF？

仓库	保管员	货品
101仓库	王xx	洗衣粉
101仓库	王xx	香皂
101仓库	陈xx	洗衣粉
101仓库	陈xx	香皂
102仓库	刘xx	微波炉
102仓库	刘xx	电吹风
102仓库	李xx	微波炉
102仓库	李xx	电吹风
102仓库	张xx	微波炉
102仓库	张xx	电吹风

如何解决？

4NF范式的规范化

- 对于 $U=X+Y+Z$ ，如果有 $X \twoheadrightarrow Y$ ，则将 U 分解 XY 和 XZ 两张表

将上述的表分解为如下两张表

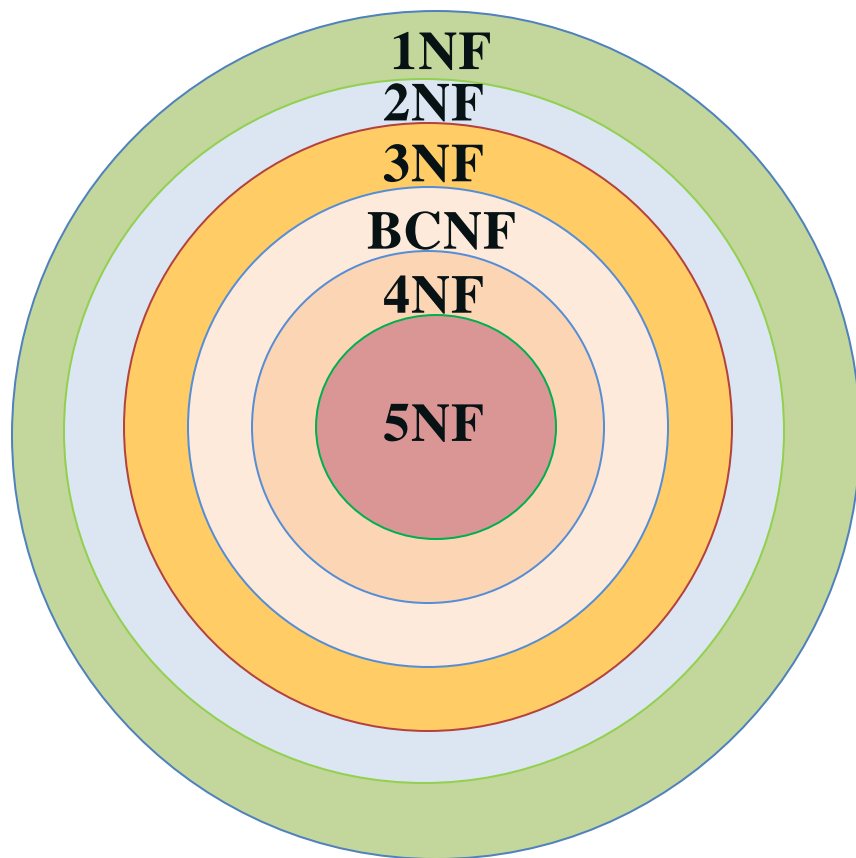
仓库	保管员
101仓库	王xx
101仓库	陈xx
102仓库	刘xx
102仓库	李xx
102仓库	张xx

仓库	货品
101仓库	洗衣粉
101仓库	香皂
102仓库	微波炉
102仓库	电吹风

可以写成如下关系模式：

- (1) 管理表（仓库, 保管员）
- (2) 存储表（仓库, 货品）

四、规范化程度



规范化的过程

对关系模式分解，把一个低一级关系模式分解成若干个高一级的关系模式。

规范化与操作效率

片面追求高级的模式，会使数据库操作效率降低
通常情况，满足3NF就达到基本规范要求。

规范化过程：

1NF 去除**复合属性**和**多值属性**；

2NF 去除非主属性对键的**部分函数依赖**；

3NF 去除非主属性对键的**传递函数依赖**；

BCNF 去除主属性对键的**传递函数依赖**；

4NF 去除**非平凡多值依赖**。

三、反规范化处理

规范化减少了数据冗余，易于保证数据的完整性，但规范化也会导致数据库性能降低，因此，在利用规范化设计数据库时要平衡两者的关系。

规范化带来结构的完整和精确性，但同时也可能带来负面的效果。也正是基于此，人们提出了反规范化设计的基本思想。

所谓的反规范化，就是适当降低甚至抛弃范式约束，不再要求一个表只表述其表自身，而是适当冗余性添加带有某种依赖关系的数据。

反规范化处理的主要手段有如下2种：

（1）增加冗余列或派生列

如果应用系统的常用操作需要关联其他表中的数据，则在进行表设计时，应直接将该列融入当前表中，使其冗余存在，称为冗余列。

（2）表的合并和分割

执行反规范化设计，表的数量往往也就会减少，而这就降低了表连接运算的压力，可以有力提升性能。

但反规范化的使用也会带来以下问题：

- （1）数据冗余的存在
- （2）降低了数据库的完整性

反规范化是把双刃剑，并不具有普遍意义，需要就事论事，用不好会伤及自身