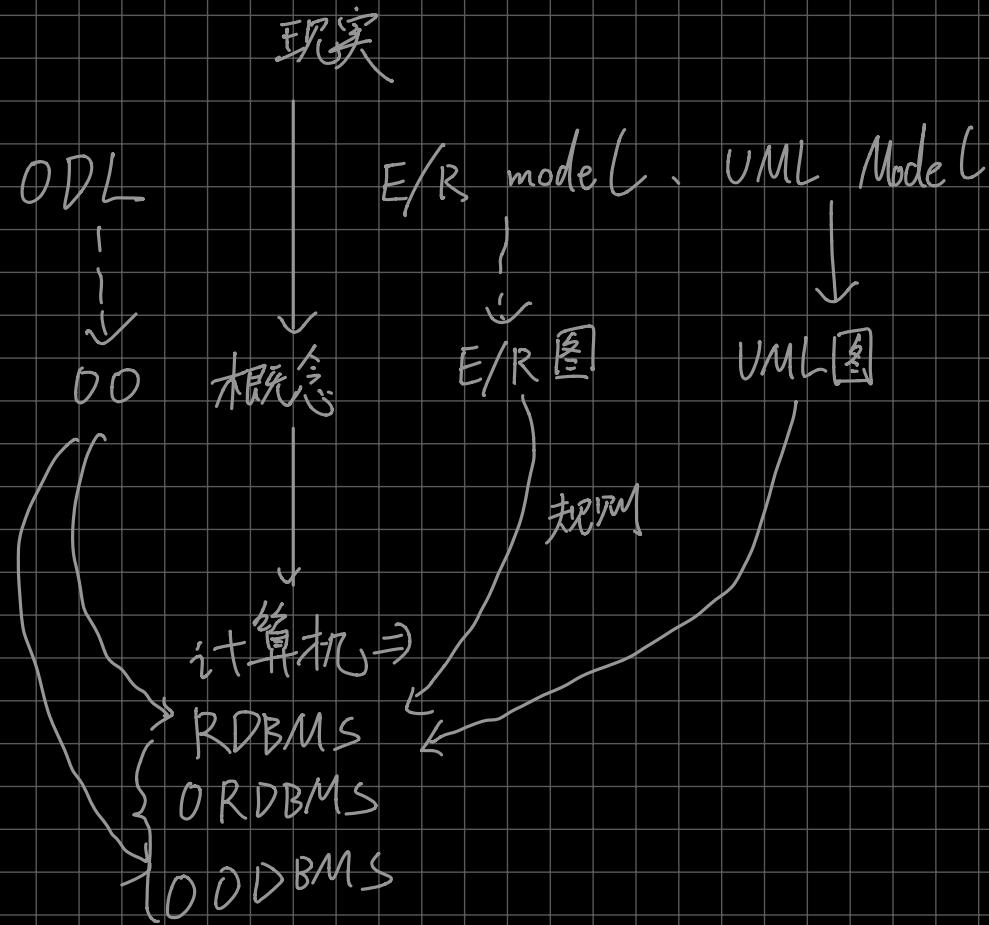


# Chapter 4 : Advanced Model



Key Point:

I. E/R Model

(convert E/R Model to Relations.)

II. UML

(convert UML to Relations.)

III. ODL

(convert ODL to Relations.)

# Lecture 6 ER Model

Concepts:

Model Components

实体

联系

多路联系

转换规则

二路联系

属性：转化可消除

角色：联系与实体间的  
边

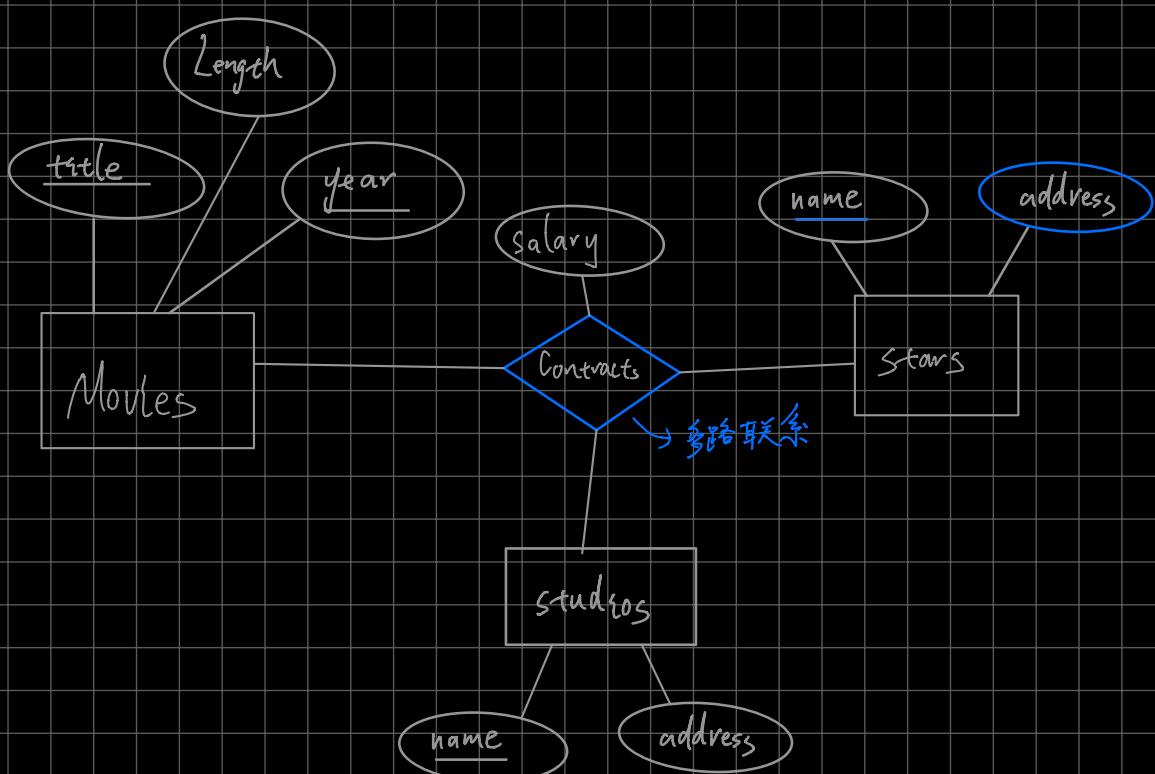
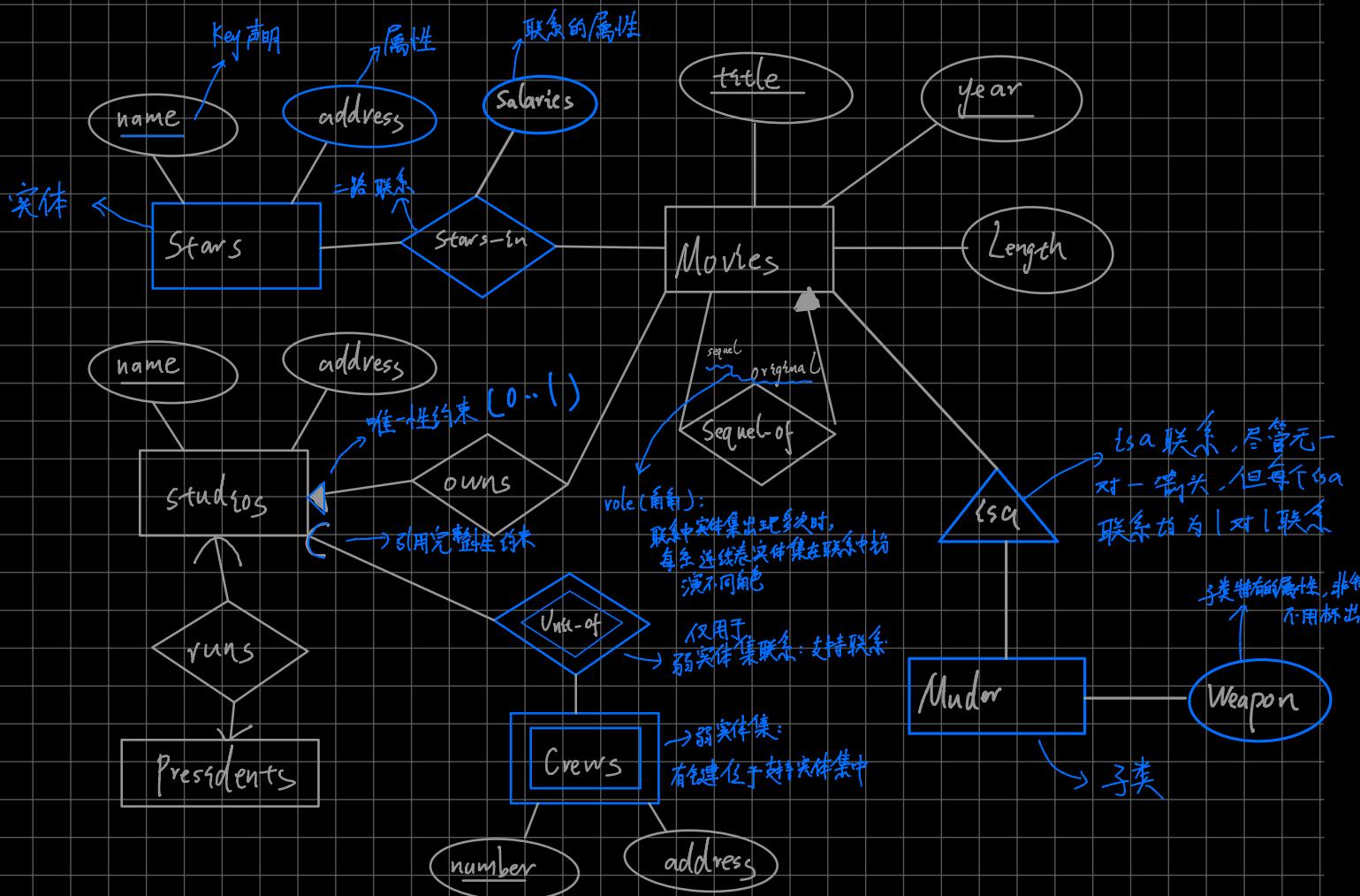
箭头 — 多对一 ( $\rightarrow$ ) : 表最多只有一个 (可能为0)  
引用完整性 ( $\rightarrow$ ) : 表 A  $\rightarrow$  B, 表 A 存在  
在表 B 必须存在。

属性 — 键 (Key)  
一般

子类：isa 结构为树形结构，只有 1 个根实体  
集，在实体集树中实体被允许有代表。  
(而面向对象方法认为实体只存在于 1 个类中)

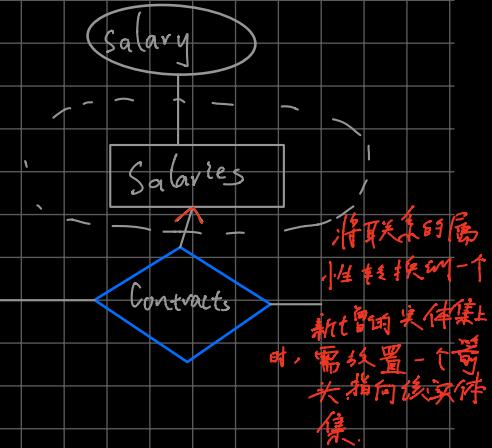
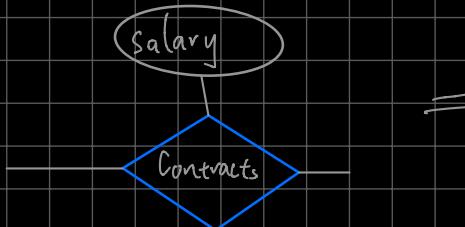
弱实体集

# E/R 图：

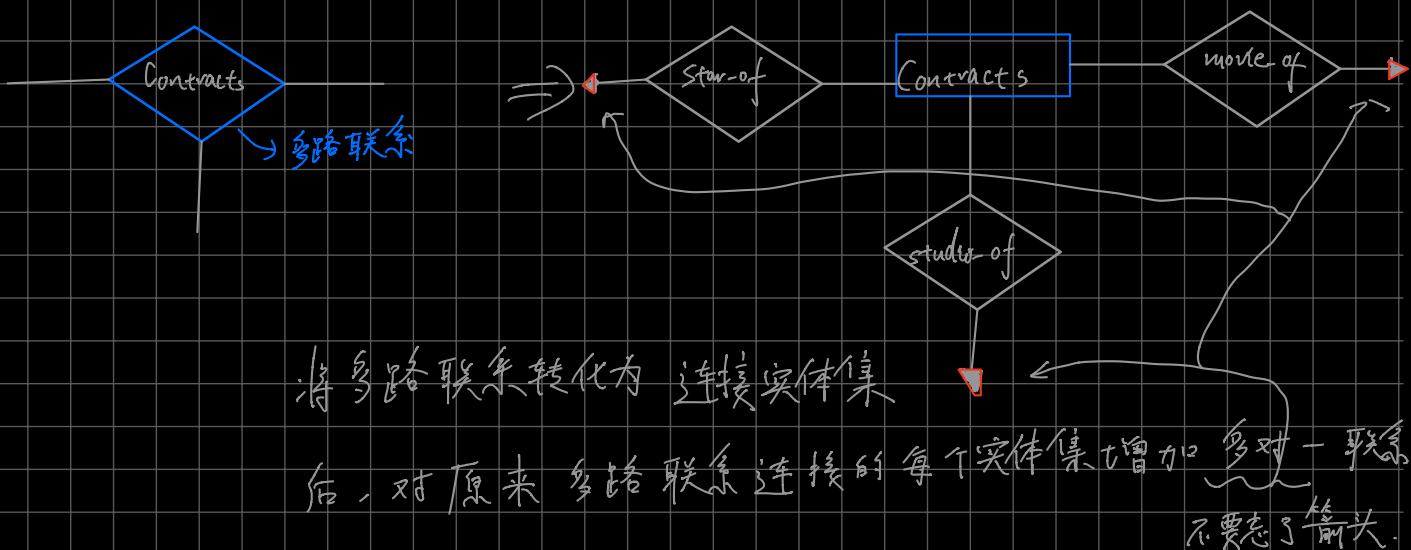


# 转换：

- ① 去除联系的属性：增添实体集



- ② 多路联系到二路联系：



# Good Design：

- ① 忠实地表达现实世界

- ② 选择合适的联系、属性

- ③ 避免冗余

同一件事表达了两次

同一件事间接表达

同一件事表达过度复杂

模型转化: E/R Model  $\xrightarrow{\text{transform}}$  Relations:

How:

实体集 relations 属性为 ① 实体集 Relations  
属性 / ② 弱实体集属性 +  
支持实体集 Key 属性

relations 属性为:  
关系所对应的实体集的  
Key 属性  $\triangle$  支持关系  
不需要生成 relation.  
 $\triangle$  不为 is-a 生成 relation

relations 属性为:  
子类 (is-a 关系) ① 根的 key 属性与该  
子类实体集本身的属性  
② 层次中每个可能的子集  
的属性.  
③ 为所有实体仅创建一个 relation -  
实体若没有该属性则以空值替代.

E/R 模型:

$\triangle$  在 many-one 联系中,  
many 的实体集形成的 relations  
属性中包含 one 的实体集属性.  
则该联系不用再写.

$\triangle$  在 one-one 联系中, 一方  
如上则该联系不用再写.

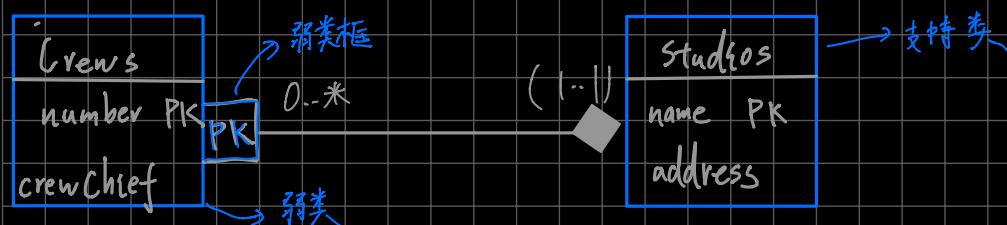
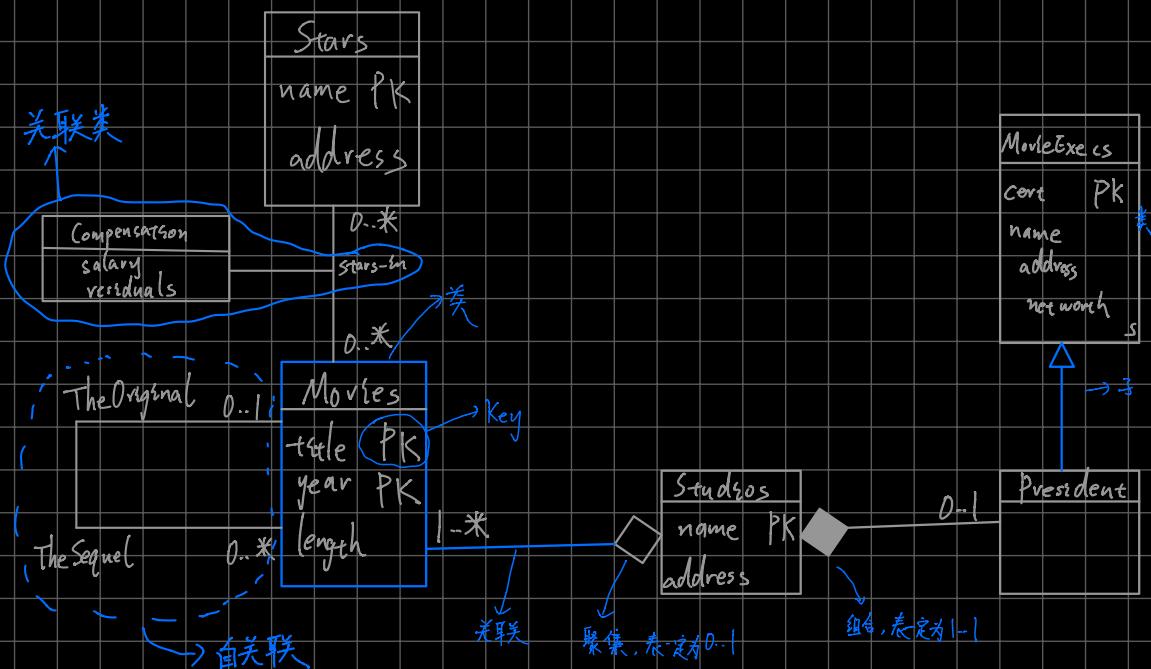
# Lecture 7: UML and ODL

UML (Unified Modeling Language)

Compared with E/R Model.

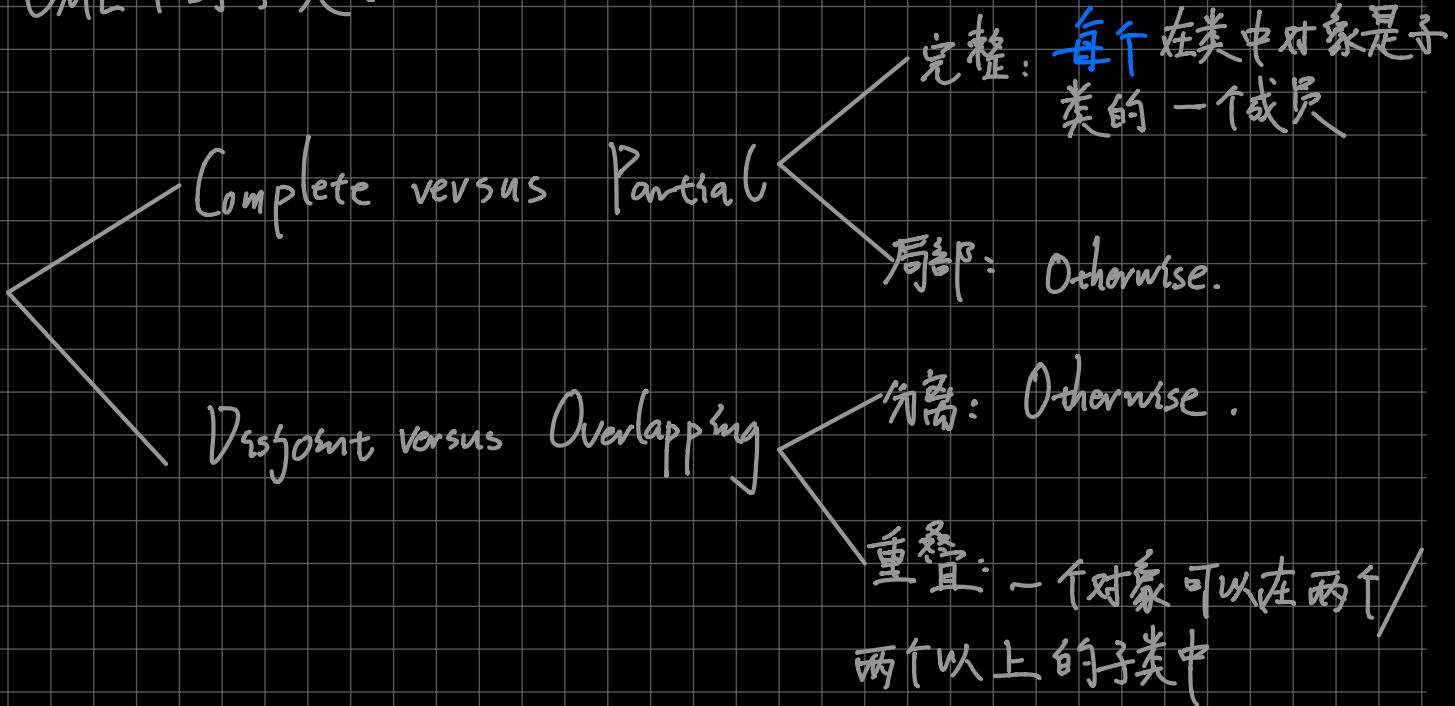
| UML              | E/R Model |
|------------------|-----------|
| 类                | 实体集       |
| 关联               | 二元联系      |
| 聚集 (Aggregation) | 多对一联系     |
| 组合 (composition) | 一对一面联系    |
| 子类               | 子类        |
| 自关联              | 角色        |
| 关联类              | 联系的属性     |
| 弱类               | 弱实体集      |
| 支持类              | 支持实体集     |
| 支持组合             | 支持联系      |

Example:



△ UML 无需弱类原因：  
每个对象均有自己的对象标识，因而可以区分每个对象。引用指针指向对象的指针。

## UML 中的子类：



## 模型转化：

类 → Relation

关联 → 以联接各个类的键为列的 Relation

聚集、组合 → 从多的那端的类构建或 Relation

子类  
① : E/R 模式  
② : 面向对象方法  
③ : 使用空值方法 } → Relation

△ How to choose:

① 层次的每一层都是分离的 / 既是完整的又是分离的 ⇒ 面向对象方法

② 层次很大 && 在某些层上是重叠的 ⇒ E/R 模式

# ODL ( Object Definition Language )

## 屬性

# Components

**联系**：必须是二元，必须是成对的  
联接的两个类声明联系，同时要声明双向  
联系。

## 方 法

## 结构 = 可异类型

集合：不能重复. 无序

包：可重复，无序

# 会連表 = 有序

数组：有序  
字典：可弄类型，无序

## Example:

## → 属性关键字

联系  
关键字

```
1) class Movie {
2)     attribute string title;
3)     attribute integer year;
4)     attribute integer length;
5)     attribute enum Genres
6)         {drama, comedy, sciFi, teen} genre;
7)     relationship Set<Star> stars
8)         inverse Star::starredIn;
9)     relationship Studio ownedBy
10)        inverse Studio::owns;
11)    };
12) class Star {
13)     attribute string name;
14)     attribute Struct Addr
15)         {string street, string city} address;
16)     relationship Set<Movie> starredIn
17)         inverse Movie::stars;
18)    };
19) class Studio {
20)     attribute string name;
21)     attribute Star::Addr address;
22)     relationship Set<Movie> owns
23)         inverse Movie::ownedBy;
24)    };

```

向联系关键字  
反向联系必须成对出现.

集合类型  
单类

class Cartoon extends Movie: Vedio {  
  ~~类名~~ 建字      多重继承

子產義建

多  
重  
統  
計

```
class Movie (key(title, year)) {
```

2

ODL中键值可选，每个对象均有 一个对象 ID

模型转化：

类、联系等转化与 E/R Model 转化相同

复杂类型属性类转换：

Method A：先将所有属性放到 relation 中，做规范化  
分解，消除 BCNF / 4NF 之后。

Method B：

创建一个新属性 (count, position)  
来表示对数标识并作为键

△ ODL 中，将联系与它们的逆配对后，只创建一个  
relation。