数据库分析

整理需求和分析

决定必须有什么信息

决定the components of that information之间的relation

Conceptual概念化的 database design 需要data model

data model是一组概念，描述了

1.数据以及数据之间的关系

2.data semantics架构 与定义域constraints

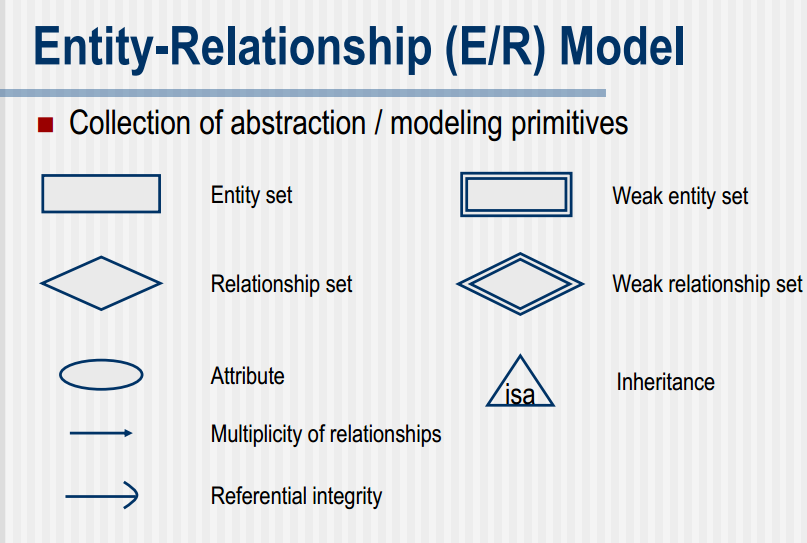
E/R entity relationship model

ER model是一个数据库建模中的图形化graphical approach

他脱胎于modeling application

多用于database design

ER language并没有统一标准



Entity 一个现实世界中的object,并且与其他object能做出区分

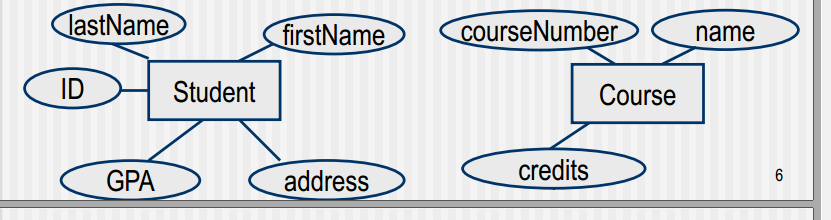
Entity Set:一组相同的entities

在一个ENTITY set内，所有Entities必须要有相同的set of attributes

例如下面的图，整个Student叫做 entity set, 然而 studentyyf 就是entity

在ODL中

Object对应着Entity， Class对应着Entity Set



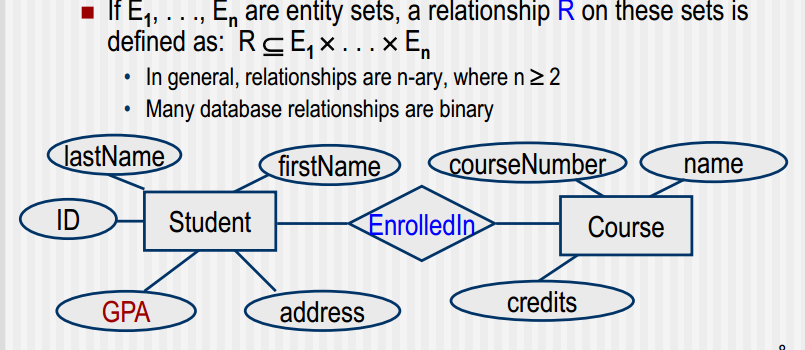
Relationships ：是Entities之间的连接

一个RElation会连接多个ENTITY

记作

类似Enrolledin的叫做Binary

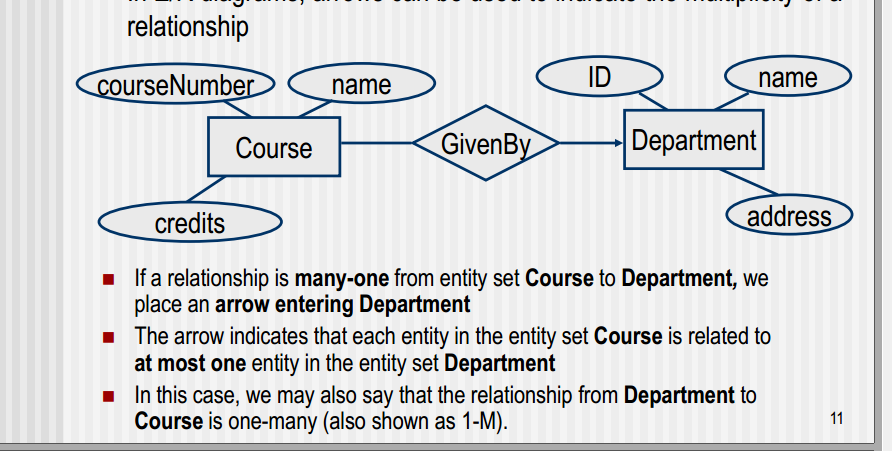
Relationship set: 是一组相同type的relationships



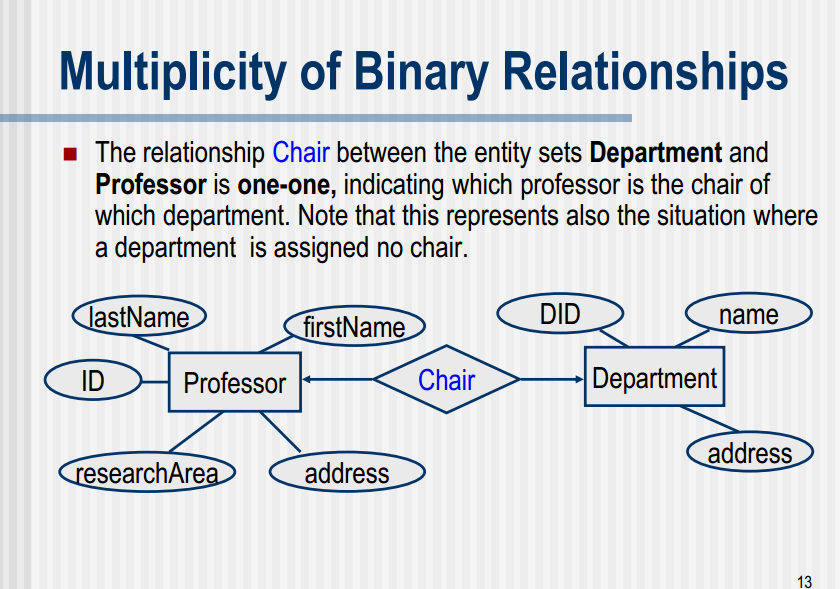
EnrolledIn 这个relationship 是manytomany

.

在ER图表张晓红，箭头可以用来表示relationship的multiplicity



左边是Many右边是1



双向的，箭头指向的是one<s所以这个关系是one to one

注意了，箭头指向的是at most one，所以也有可能没有Professor

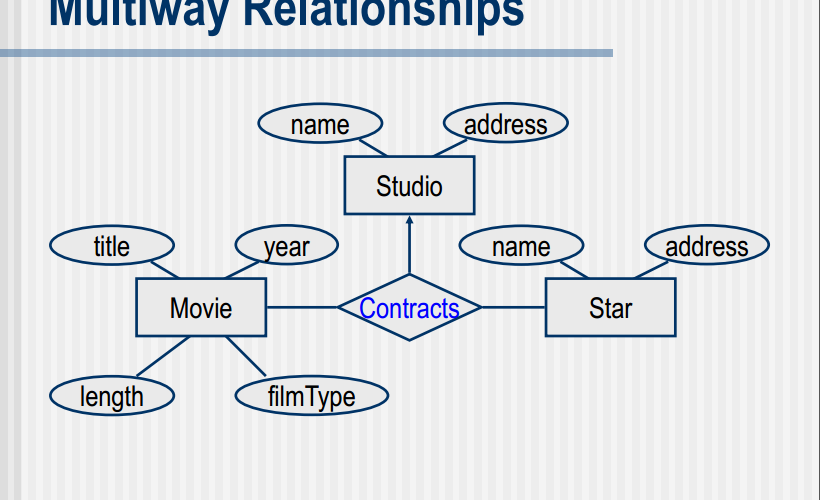
Multiway Relationships

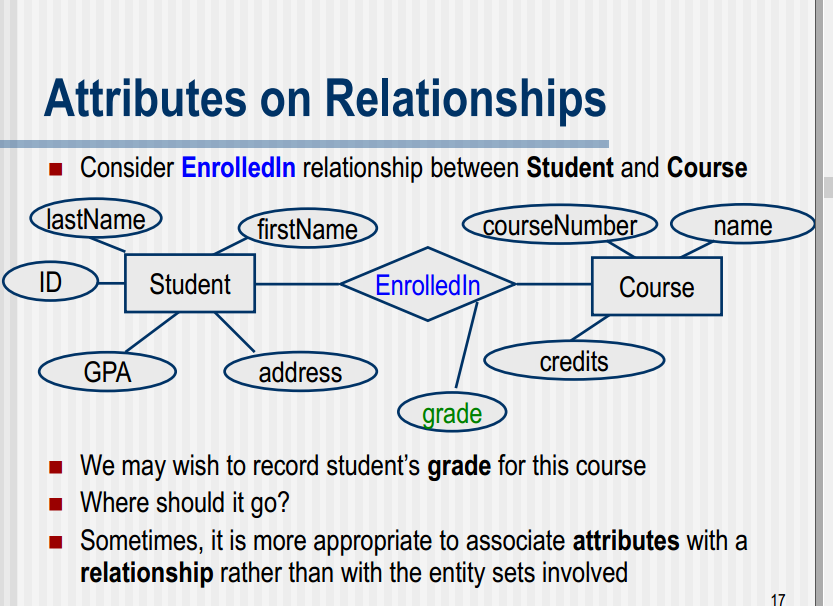
ODL：一种Object oriented 表示法， 只允许描述binary relationships

然而实际上我们有时候需要描述n-ary (multi-way) relationship,

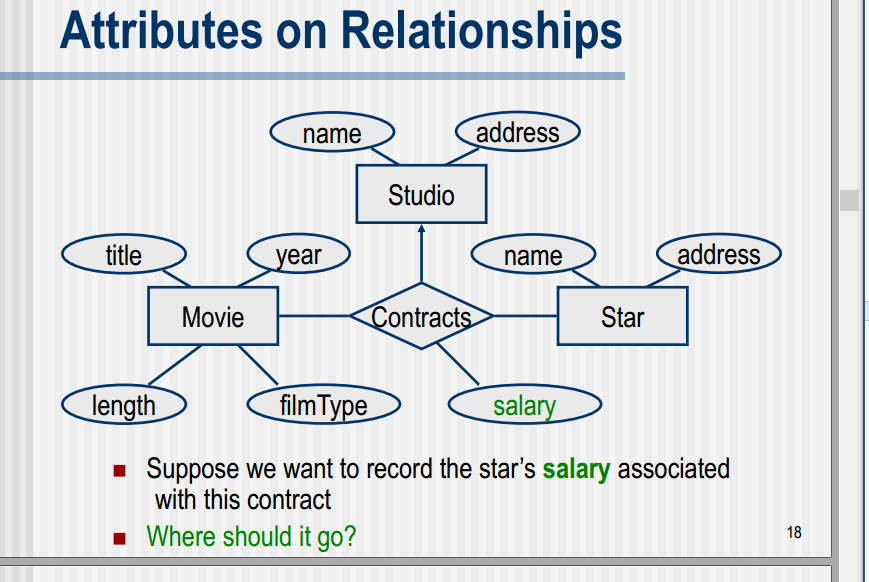
而ERmodel允许的

一个diamond 链接代表着n-ary(包括binary)

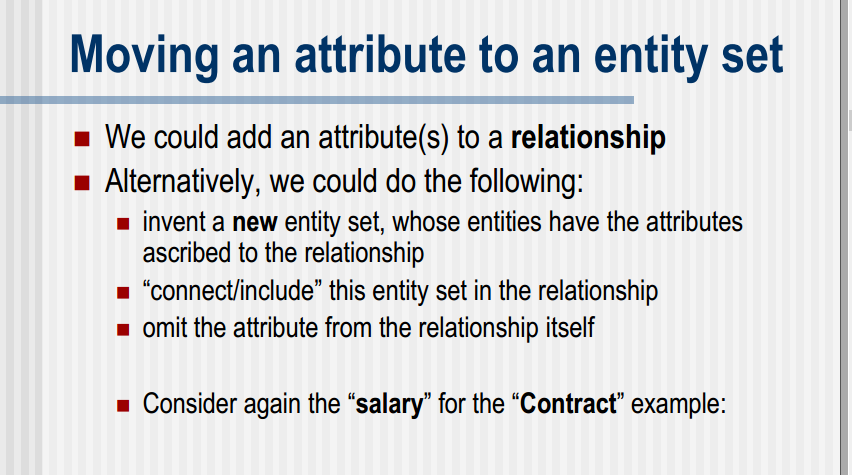




有的时候把attribute连接在relationship上更恰当

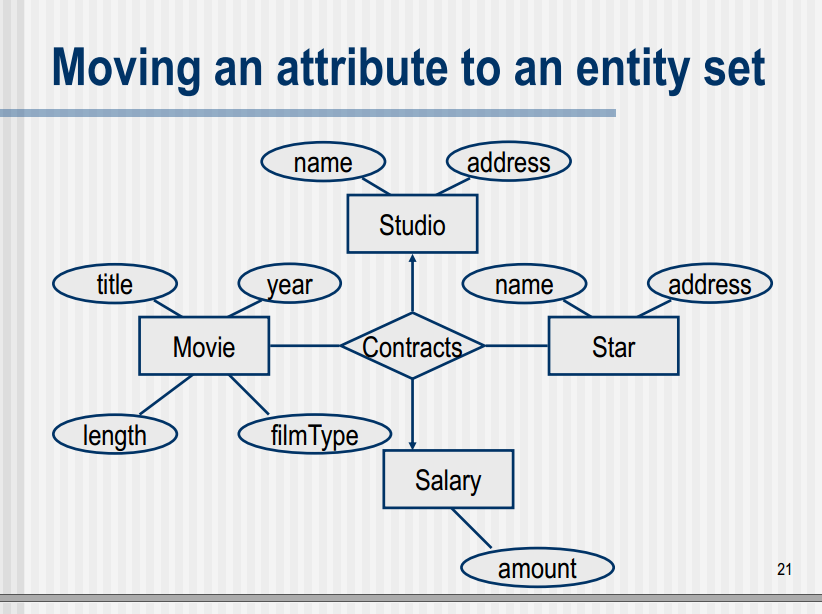


salary与Contract这个链接 关系更紧密



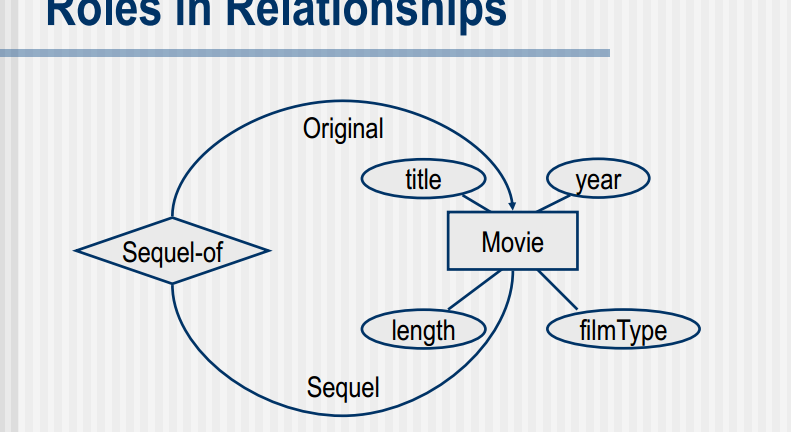
我们可一个给relationship加上attributes

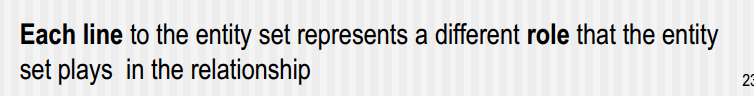
但是作为替代，我们也可以创造一个新entity，然后把这个attribute连接到新entity上



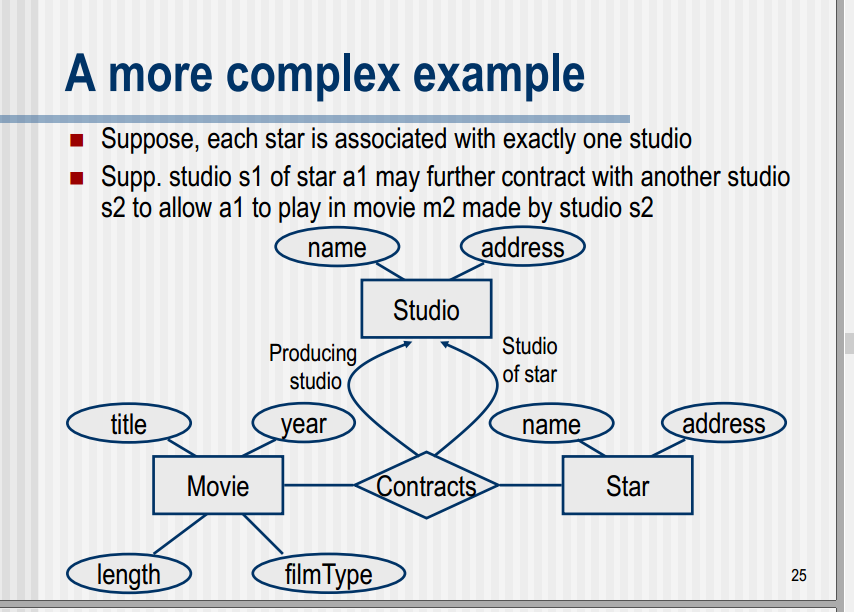
一个entity set也可能同时出现两次或多次在一个relationship中

例如，我们想捕捉两个movie的relationship,一个是另一个的续集





每一条，Seque-of可能代表着不同的意思， （而不是上下切开看）， 有可能movie1是movie 2 的original，也有可能moview 1 是movie 2的sequel



假设每一个star 链接当前 exactly one studio

然而，s1 的star a1也许与另一个studio签合同，这样在另一个studio拍的片里可以用他

这样就是两类studio关系，

Studio of star-> 我属于的事务所

producing studio-> 制片方

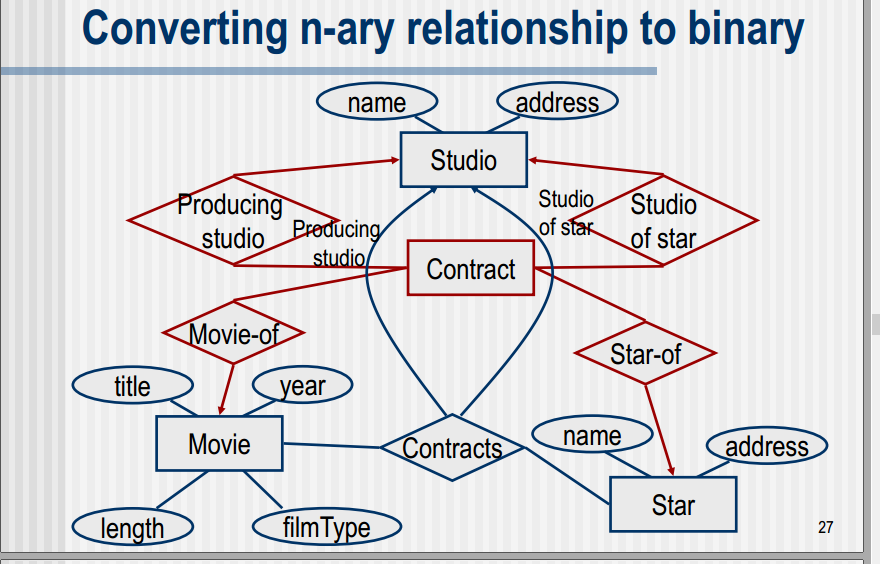
任意n-ary 多向relationship R可以转化为一组binary reashinships 同时不会流失信息

流程

1.创建一个新Entity,叫做connecting entity set,他的Entity可以被看作 这个Relation R的 tuple（object）

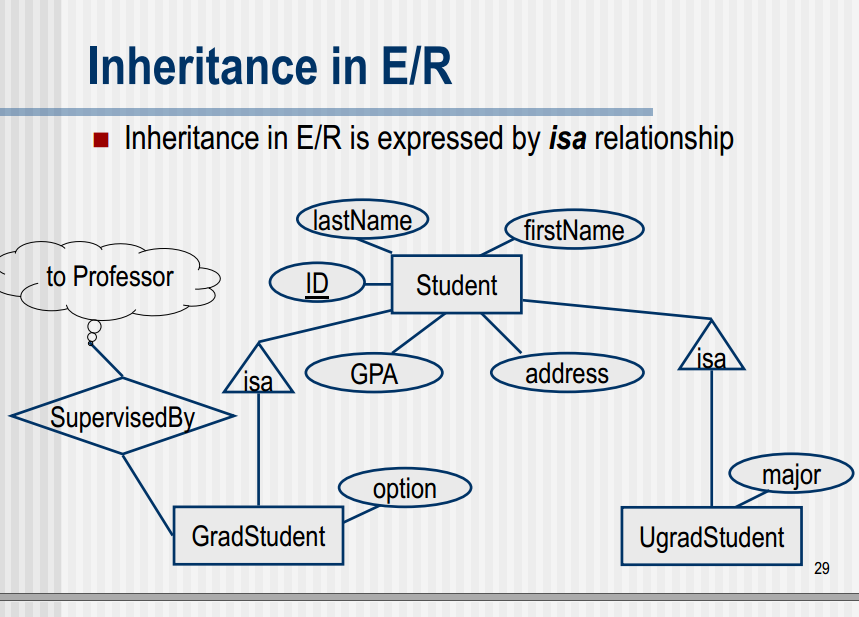
2.创建binary entities，每一个都和entity sets 一一相连

3. 当然有的entity set扮演多个角色（就相当于Movie续集例子），那么要和每一个role都要链接



这里就是contractt被拆分成一个新的entity，他应该指向MOVIE,STAR,STUDIO

把这些关系具体描述



ISA代表着继承

尖头是父类

然而ER与object oriented有些许的区别

在OO里，一个object必然是一个class的member

而在ER里，一个entity是拥有他自己本身的属性，然后通过isa又得到了父类的属性，通过这些component组成了一个entity

（实际上没有特别大的区别相当于继承,OO是继承了父类属性，ER是连接父类属性创造一个object）

Constraints

现实世界中有很多方面是不能用ODL或者ER表示出来的

这些额外的信息通常以constraints 限制的形式出现

有时，对这些附加信息的建模超出了class、entity sets、attributes和relationships施加的结构和类型约束

A classification of Constraints Constraint的分类

Key K：属于Entity set R的属性的一种，但是他是独一无二的

Single-value constraints, Single-value constraints require that a value be unique in certain contexts.

他需要在某种上下文情况下是独一无二的，例如院长位置,或者many to one 关系

Referential integrity constraints：require that a value referred to actually exists in the database.

你一个Value如果指代的是某个object,某个entity，那么这个Object或entity必须在数据库中存在，避免dangling pointers

Domain constraints specify what set of values an attribute can take. ，某个attribute必须在一定范围以内，定义域

General constraints are arbitrary constraints that should hold in the database. 任意在database中需要保持的限制

**Constraints are part of the schema of a database.**

Keys

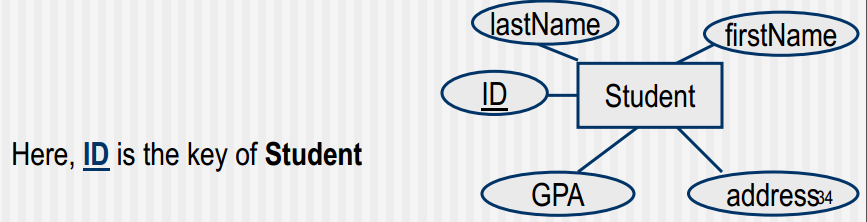
SuperKey: unique，用来区别一个Entity

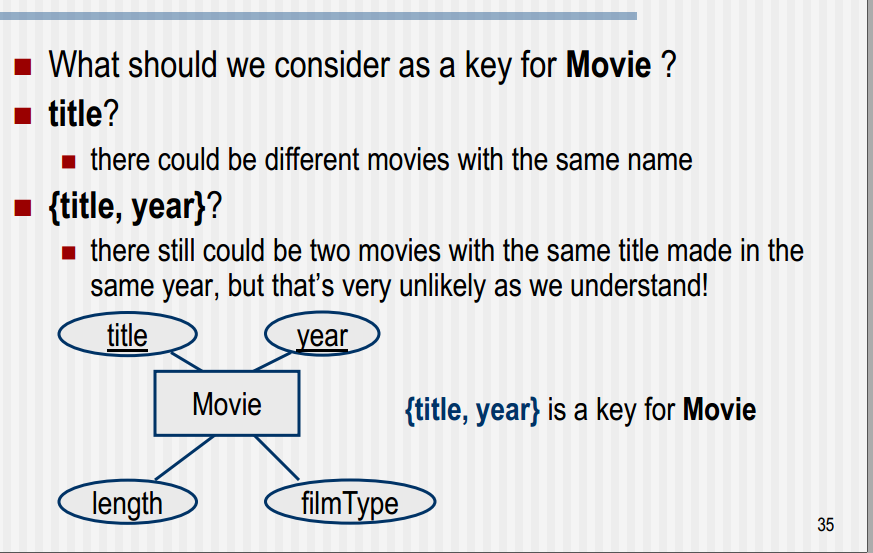
Minimal superkey: 也叫做Candidate key， 可以当做primary key因为他们也是独一无二的，

primary key , Candidate key中挑选出来一个，其他的作为alternates备选项

给他们加下划线代表着key

Alternate keys没有特殊符号





可以两个attribute或多个组成PK

single-value限制

在ER模型中

ATTRIBUTE是ATOMIC //单数的

你可以用一个箭头表示一对N对等关系

但是只有relationships允许存在一对多

attribute只能有一个值， //例如 学生有所上的课，你不能在这里面填很多课，而是要新建一个表

对于一个single-value attribute中

又有两种情况

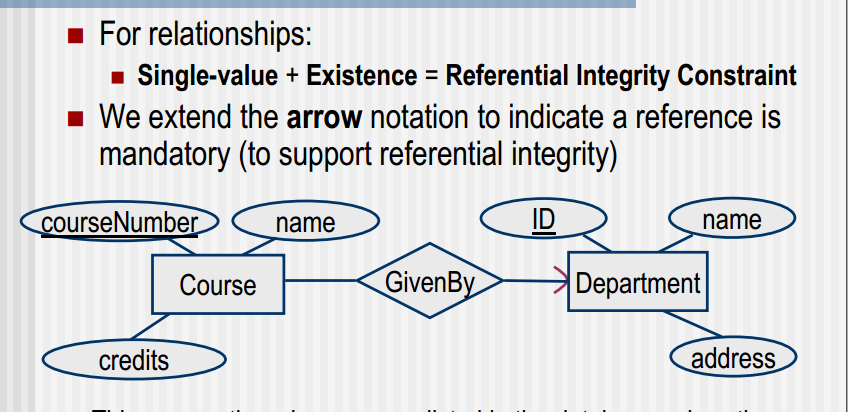
Not null： 例如Key,或part of key

以及可以为null

Referential Integrit Constraint,

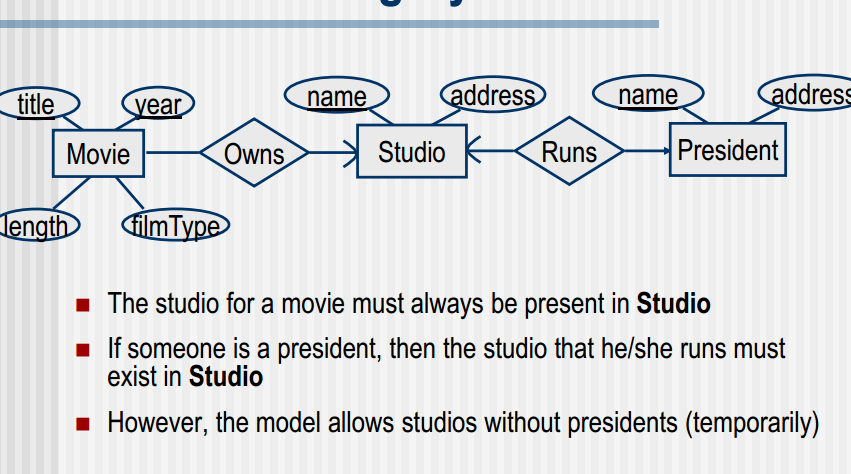
引用完整性约束

对于single-value attribute要引用我们必须保证引用物存在的关系，例如foreign key



我们延长圆化箭头符号来代表这个reference必须是建立在existence的基础上的

例如，除非我们创立了这个department，不然不能把课添加到database里



没有studio，就没有President或者MOVIE

没有President与Movie，可以有Studio

Domain Constraint

定义域限制

让Attribute的值限定在一定范围内

ER模型不允许实施Domain Constraint

ODL 允许使用types 来限制控制attribute的可能值 //决定是INT还是CHAR

ODL不允许控制attribute的值的范围

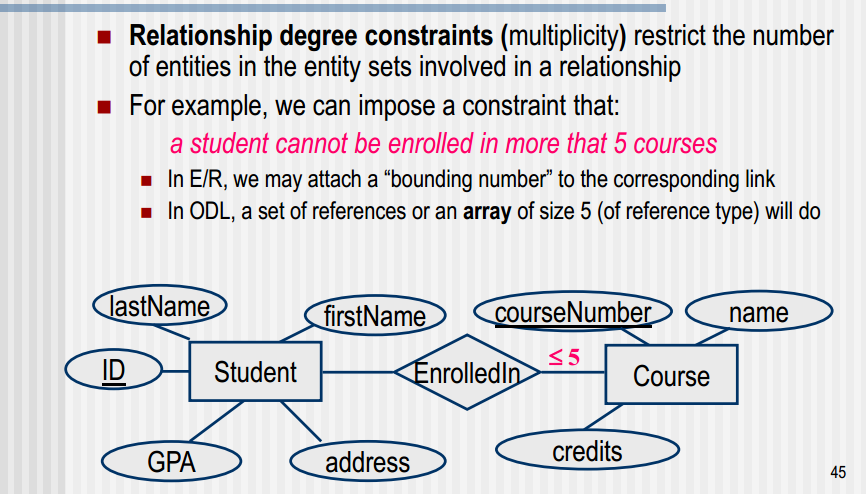
Relationship degree constraints(multiplicity)限制

限制一个relationship中，entity的数量

例如一个student最多加入五门课

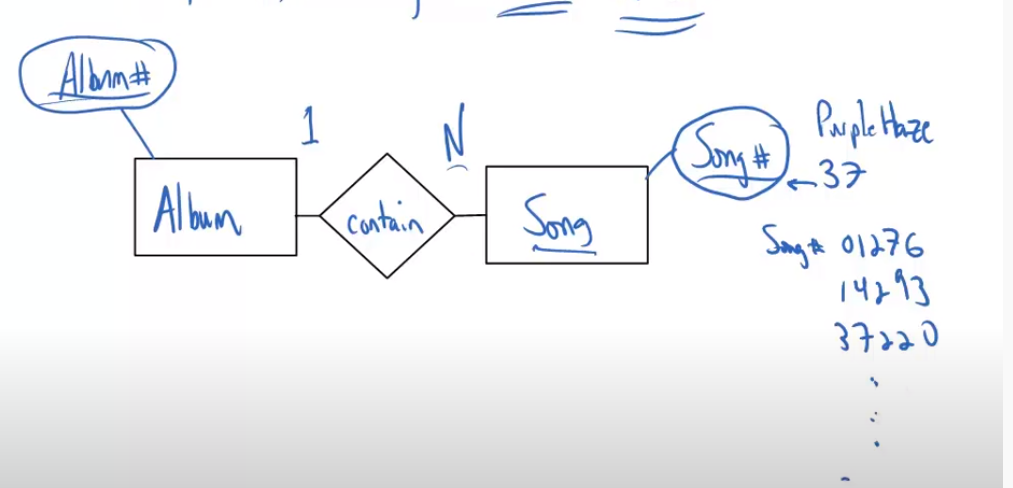
ER我们可以加个bounding number

ODL我们可以用array SIZE OF 5



Weak Entity

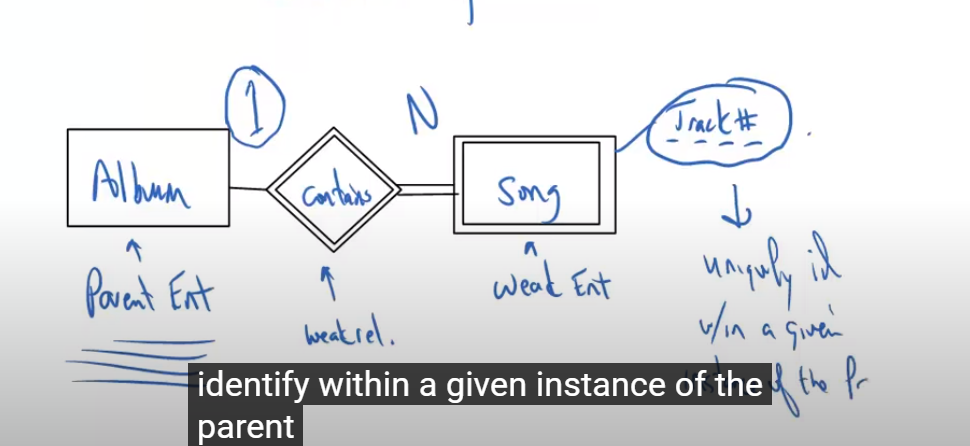
例如ALBUM-TRACK关系



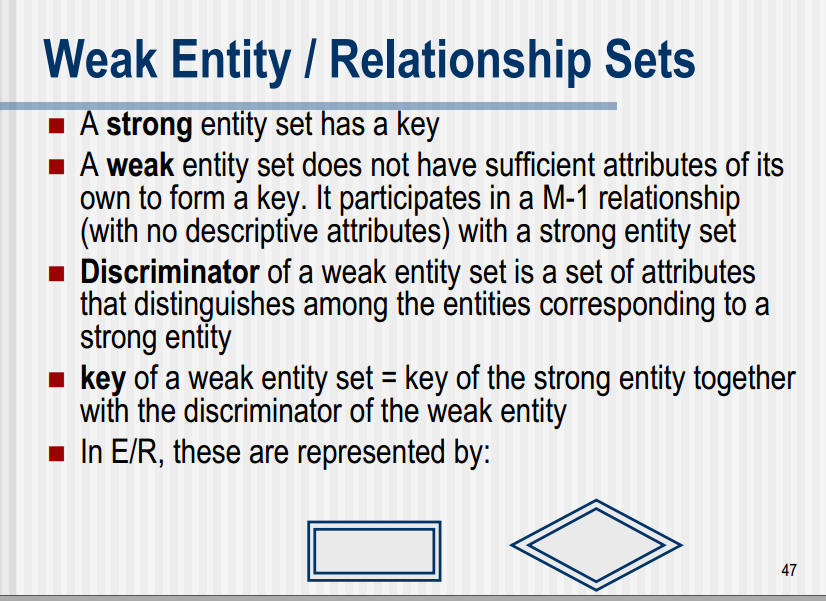
你可以把所有SONG放在一个大DATABASE里，每一个KEY都不同

而WEAK ENTITY

只是把每个SONG 分成TRACK 1 TRACK2 ，



在album里能区分这首歌



WEAK entity本身并没有一个完全体的KEY，他通过一个M-1 relationship与 Strong entity set结合（WEAK ENTITY需要依赖一个Strong entity）

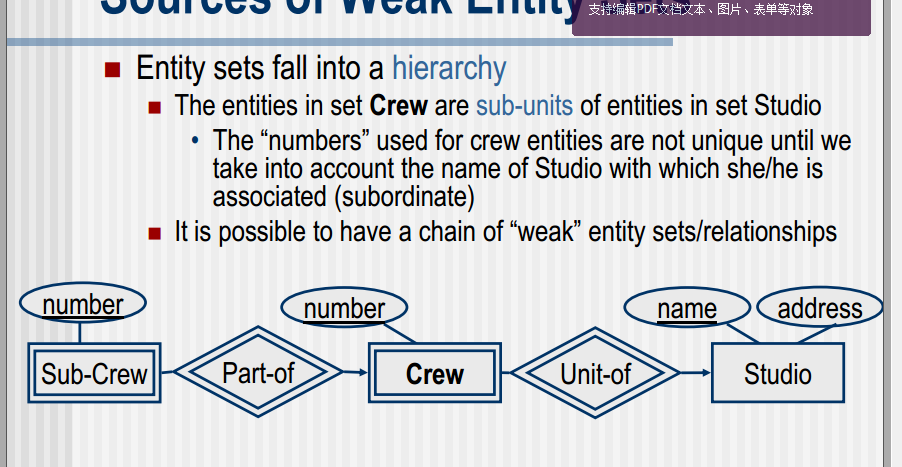
而他还需要discriminator，也就是区分着不同的weak entity的， 也就相当于专辑里的track

这样WEAK ENTITY SET的key实际上是他的strong entity的key+ discriminator

例如专辑#+TRACK#

Chain of weak entity

通过WEAK ENTITY，我们能实现hierarchy分层



weakentity->weak entity->strong，最终连到一个Strong entity

CES:

例如M TO N 中间那个ENTITTY

纳闷没有自己的属性，而是用连接的ENTITY的属性组成KEY

CES永远是WEAK

