南京大学 NANJING UNIVERSITY

奋进行动 团结奋斗 争先进位 坐言起行 应势而动

数 据 管 理 基 础

第7章 数据库设计

（逻辑结构设计）

智能软件与工程学院

数据库的逻辑结构设计

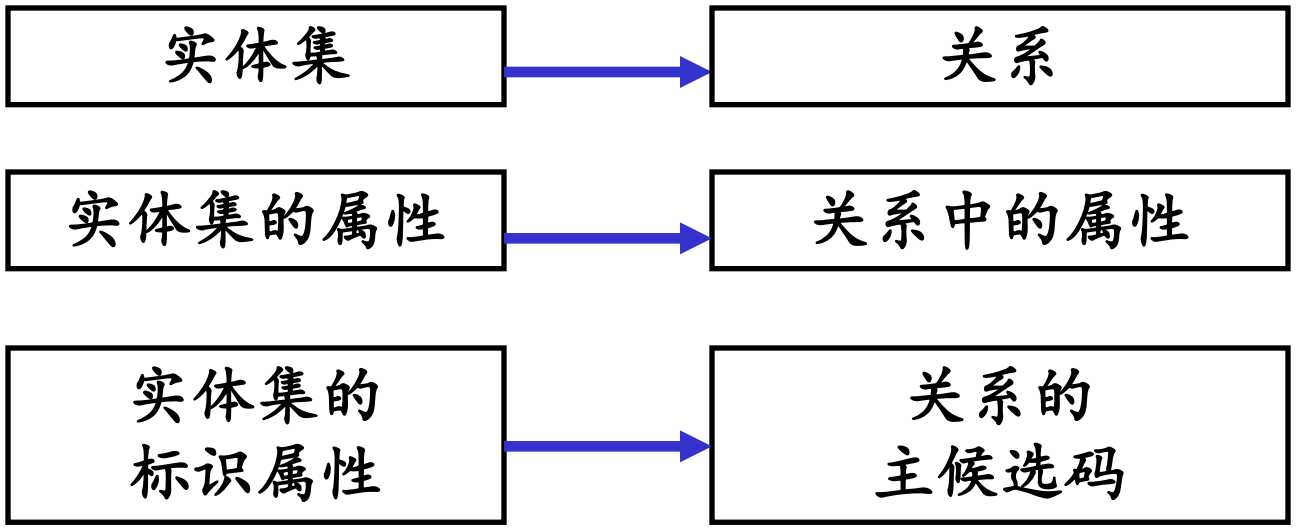
- 基本任务
  - 将前一阶段设计得到的概念数据模型(EE-R模型)转换成用户所选择的数据库管理系统(DBMS)支持的逻辑数据模型。
  - 由于在概念设计阶段选择的是EE-R模型，而目前最流行的是关系数据库系统，因此，数据库的逻辑设计的基本任务就是：
    - 将EE-R模型转换成等价的关系数据库模式

数据库的逻辑结构设计

- 基本方法
  - 每个‘实体集’被转换成一个关系
  - 每个‘联系’通常也被转换成一个关系

数据库的逻辑结构设计

- 实体集的转换方法
  - 每个实体集被转换成一个关系（模式）



  - 关系及其属性的命名采用原实体集及其属性的名称

数据库的逻辑结构设计

- 联系的转换方法
  - 在一般情况下，每个联系也被转换成一个关系模式，联系名被用作转换得到的关系模式的关系名，该关系模式中的属性由两部分组成：
    - 联系自身所具有的属性
    - 与该联系相关的实体集的标识属性
  - 来自于相关实体集的标识属性也是相关实体集转换得到的关系模式的主候选码，因此它们也是联系转换得到的关系模式中的外候选码

数据库的逻辑结构设计

- 在从概念数据模型（EE-R）到逻辑数据模型（关系模型）的转换过程中，还可能存在一些需要处理的问题。
  1. 命名与属性域的处理
  2. 非原子属性处理
  3. 联系的转换
  4. ISA联系的转换
  5. 规范化
  6. **RDBMS性能调整**
  7. 约束条件设置
  8. 关系视图的设计

逻辑设计的基本方法

1．命名与属性域的处理

- 关系及属性的命名
  - 尽量采用在EE-R模型中原有的名称
  - 可以重新命名，但要避免命名的冲突现象
    - 在同一个数据库模式中，关系名具有唯一性
    - 在同一个关系模式中，属性名具有唯一性

- 属性域的定义
  - 根据DBMS的选型进行必要的数据类型转换

转换规则：实体集到关系的转换

- 每个实体集都被单独转换成一个独立的关系。

E-R concepts	Relational Database
entity name	table name
attribute name	column name
the single-valued attributes	columns of the table
the composite attributes	multiple simple columns
an identifier for the entity	a candidate key for the table
primary identifier	primary key
entity instance	row

- 转换后得到的table和column的命名，可以直接使用对应实体的实体名和属性名；
- composite attribute将被替换为它的所有成员属性；
- 在转换后的table中，如果存在column的‘重名’，可以对column进行重命名。

2. 非原子属性的处理

- 组合属性 (composite attribute)
- 多值属性 (multi-valued attribute)
  - 多值属性的例子

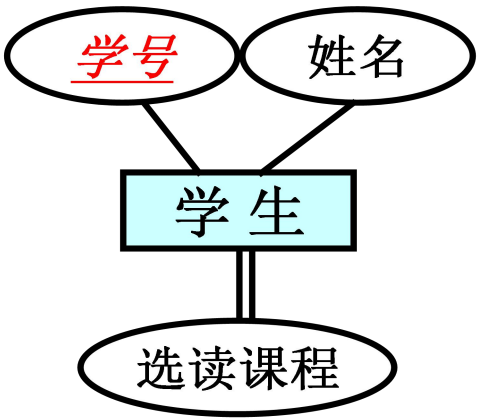


表 9-1 之 EE-R 图

学号	姓名	选读课程
S1307	王承志	Database Operating System Computer Network

表 9-1 学生实体

非原子属性的处理

- 多值属性的处理方法一
  - 关系模式不变，但原有关系的一个元组将被纵向展开成多个元组（如下表所示，以满足1NF的要求）
  - 在上述转换过程中，虽然实体集(或关系模式)中的属性没有增加，但转换得到的关系模式的主候选码由原实体集的主标识符属性和该多值属性联合构成（红色的属性名集合）。

学号	姓名	选读课程
S1307	王承志	Database
S1307	王承志	Operating System
S1307	王承志	Computer Network

‘学生’关系

非原子属性的处理

- **多值属性的处理方法二**
  - 每一个多值属性，都将被分解出去，和原实体集的主标识符属性联合起来单独组成一个新的关系。
  - 新关系的主候选码由原实体集的主标识符属性和该多值属性联合构成。

E-R concepts	Relational Database
the plural multi-valued attribute	name for the new table
primary identifier	column for the new table
the multi-valued attribute	column for the new table
primary identifier + the multi-valued attribute	primary key
entity instance + an element of the multi-valued attribute	row

学号	姓名
S1307	王承志
S1307	王承志
S1307	王承志

‘学生’关系

学号	选读课程
S1307	Database
S1307	Operating System
S1307	Computer Network

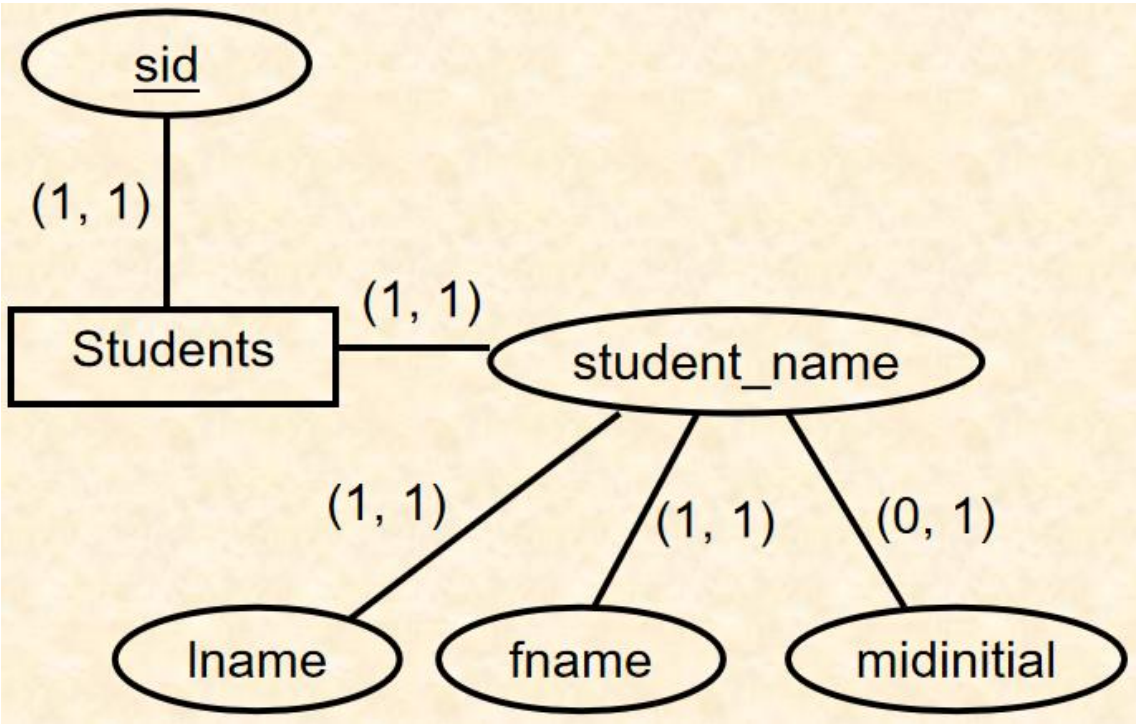
‘选读课程’关系

非原子属性的处理

- **组合属性的处理**

  将一个组合属性横向展开成多个属性

  - 例：如下图所示的Students实体集



  - 转换得到的关系模式为：

    Students (sid, lname, fname, midinitial)

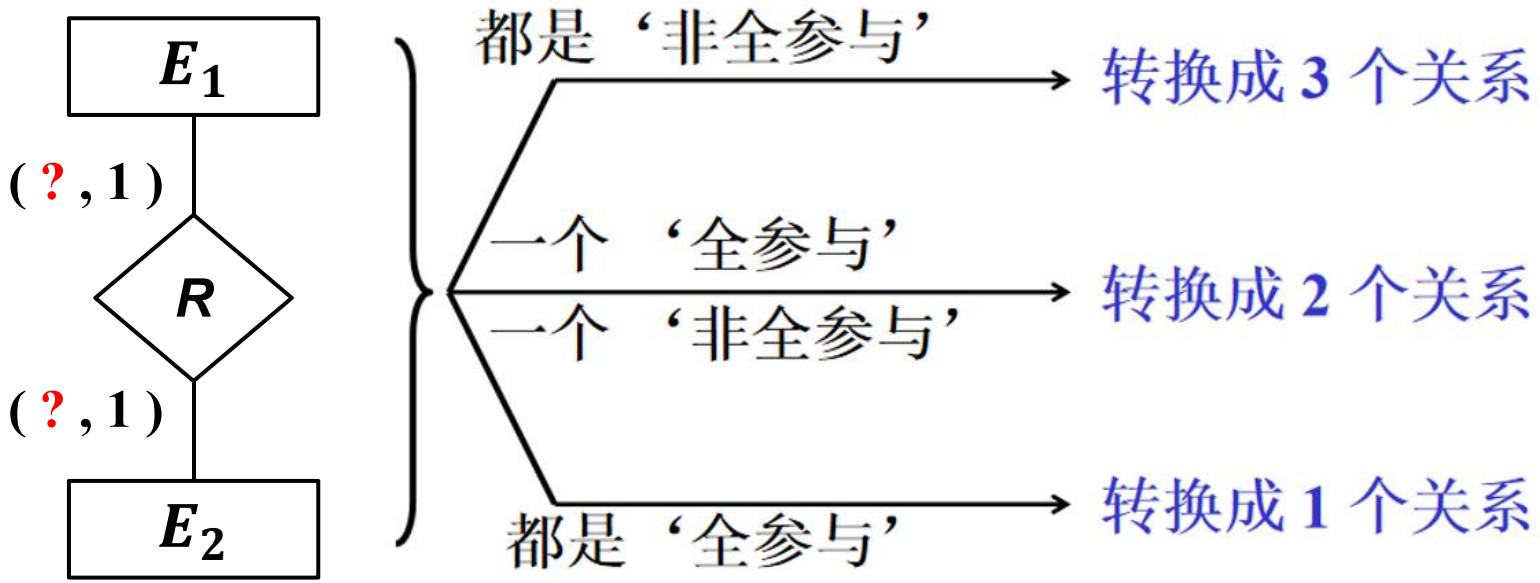
逻辑设计的基本方法

3. 联系的转换

- 一般情况下，一个联系可以被转换成一个关系。
- 但是在有些情况下,联系也可被归并到相关联的实体所对应的关系模式中去，即将联系与某个(或几个)相关联的实体集共同转换成一个关系模式。

- 下面以两个实体集之间的二元联系为例，介绍在不同情况下其逻辑设计的方法。
  - 两个实体集之间的二元联系，可以根据该联系的函数关系，以及每个实体参与该联系的参与方式将其转换成1、2或3个关系模式
  - 根据实体在联系中的最小参与基数，可以将参与方式分为
    - min_card(E, R) = 1
      - 称实体E在联系R中是‘强制参与’(mandatory participation)，又被称为是‘全参与’
    - min_card(E, R) = 0
      - 称实体E在联系R中是‘非强制参与’，又被称为是‘可选参与’(optional participation)，或‘非全参与’

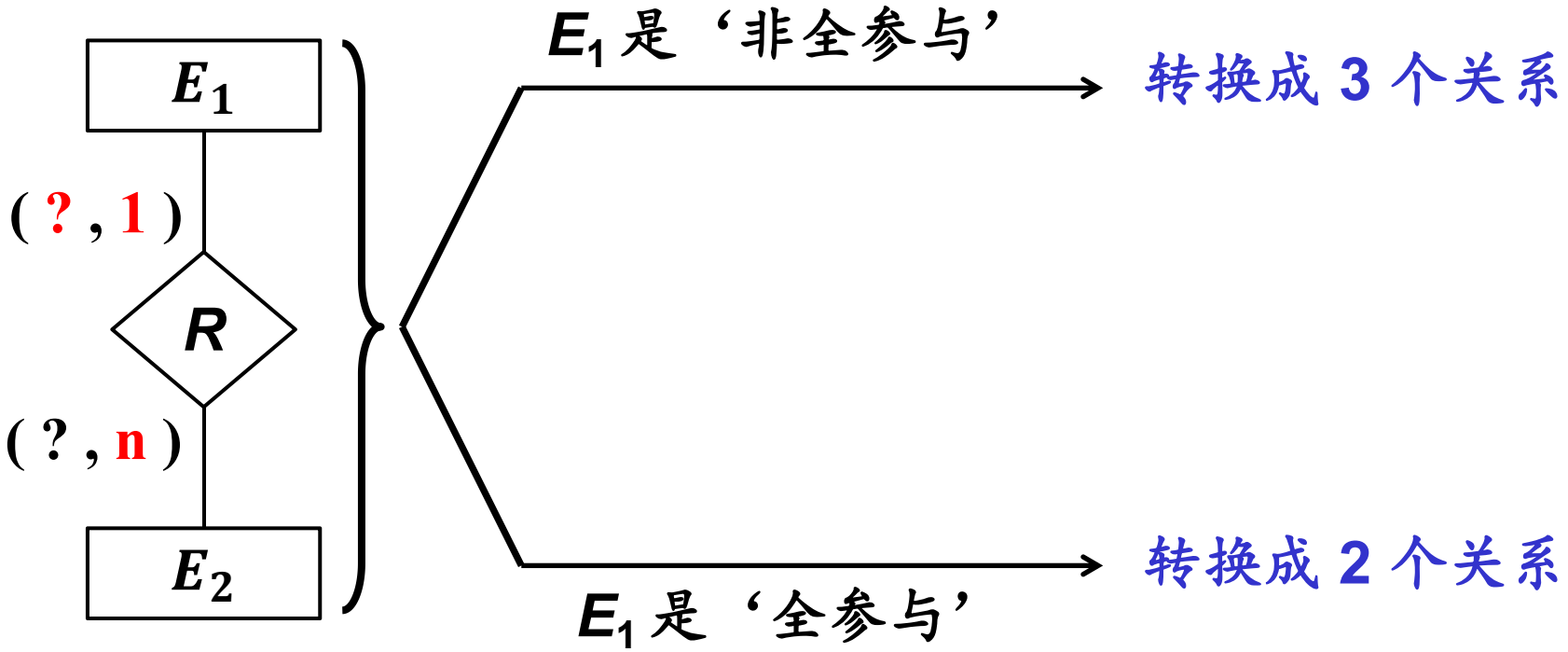
联系的转换



‘一对一’联系的转换方式

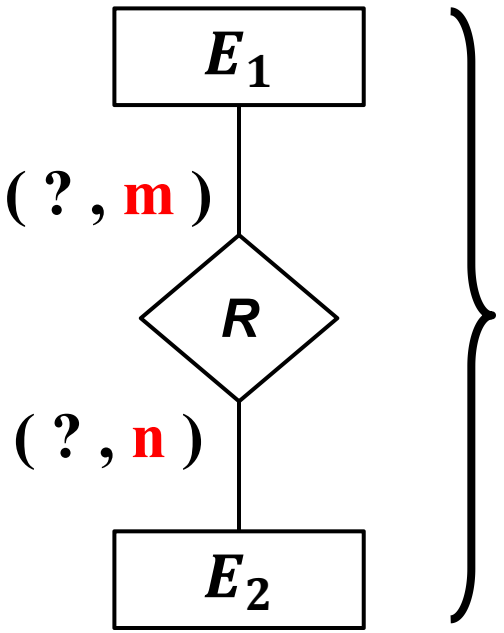
（？代表 0 或 1）

联系的转换



‘多对一’ 联系的转换方式

联系的转换



转换成 3 个关系

‘多对多’ 联系的转换方式

“一对一”二元联系的转换

① **一对一：都是‘非全参与’（图9.12）**

- 将被转换成三个关系模式。其中，k1、k2分别是实体集E_1和E_2的标识符(identifier)

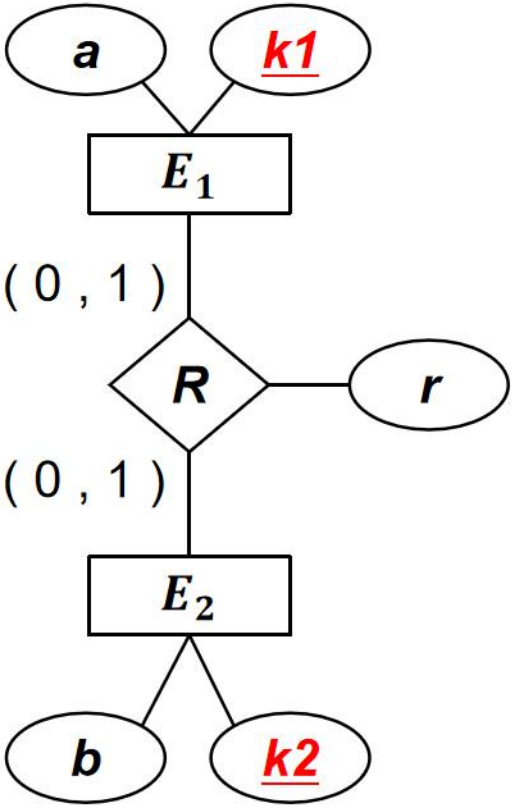


图9.12　一对一联系

$E_1(k1, a)$

- 候选码是 k1

$E_2(k2, b)$

- 候选码是 k2

$R(k1, k2, r)$

- 两个候选码： k1 或 k2
- k1 和 k2 也是关系R的两个外候选码

“一对一”二元联系的转换

② 一对一：E_1 是‘全参与’，E_2 是‘非全参与’

- 可以将联系 R 与实体集 E_1 合并，将图9.12转换成二个关系模式。

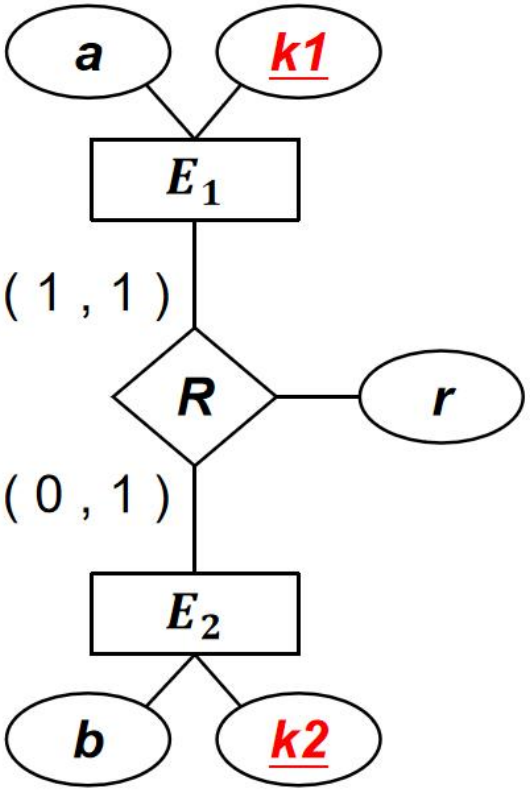


图9.12 一对一联系

$E_1(k1, a, k2, r)$

- 候选码是 k1
- k2是关系 E_1 的外候选码

$E_2(k2, b)$

- 候选码是 k2

“一对一”二元联系的转换

③ **一对一：都是‘全参与’**

- 将三者全部合并，将图9.12转换成一个关系模式。

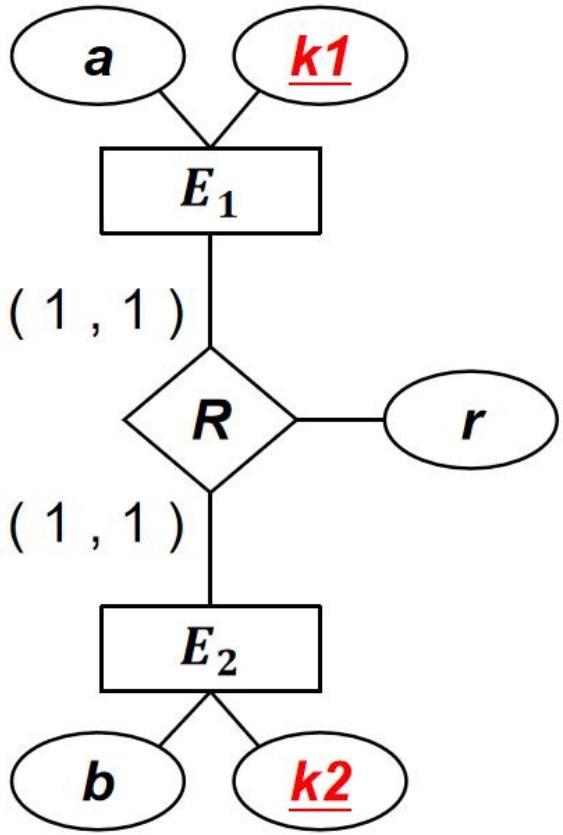


图9.12 一对一联系

$E(k1, a, k2, b, r)$

- ***k1, k2是该关系的两个候选码***

“一对多” 二元联系的转换

④ **一对多：(单值参与) 多端 E_2 是‘全参与’**

- 可以将联系 R 与实体集 E_2 合并，将图9.13 转换成 2 个关系模式

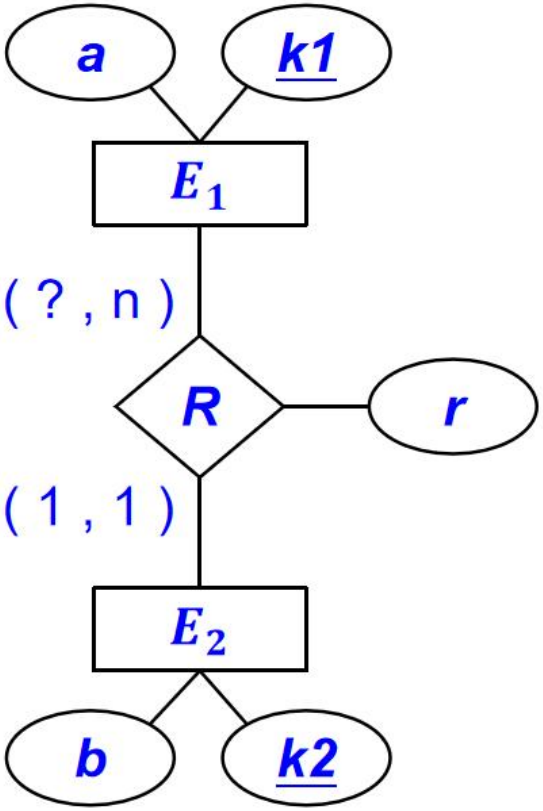


图9.13 一对多联系

$E_1(k1, a)$

- *候选码是 k1*

$E_2(k2, b, k1, r)$

- *候选码是 k2*
- *k1 是关系E2的外候选码*

“一对多” 二元联系的转换

⑤ **一对多：(单值参与) 多端 E_2 是‘非全参与’**

- 必须将图9.13 转换成3个关系模式

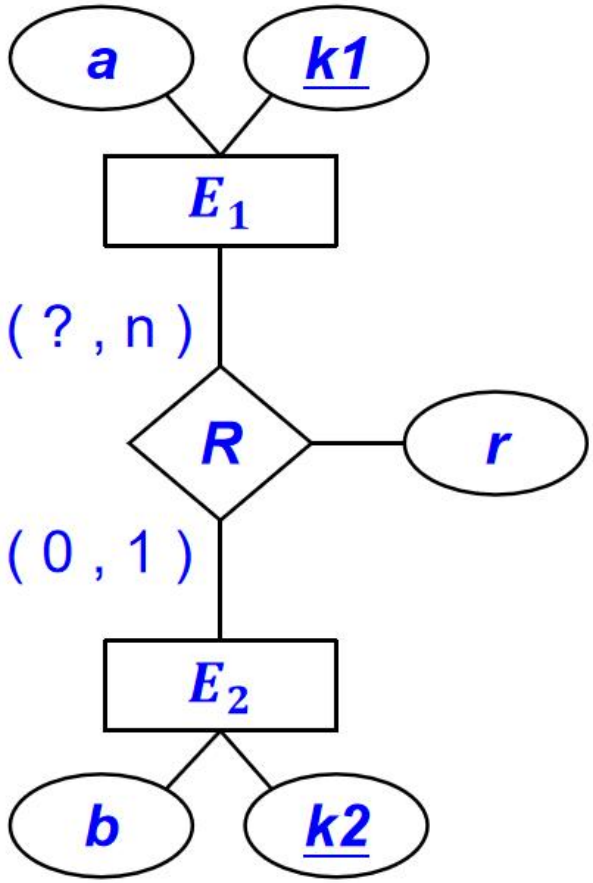


图9.13 一对多联系

$E_1(k1, a)$

- *候选码是* ***k1***

$E_2(k2, b)$

- *候选码是* ***k2***

$R(k2, k1, r)$

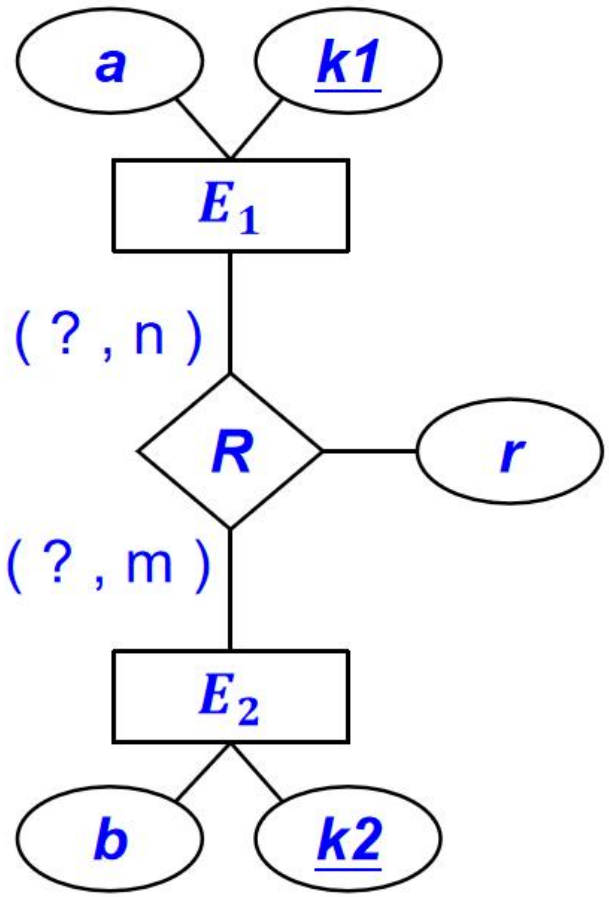
- *候选码是* ***k2***
- ***k1是关系R的外候选码***

❑ 注意：在这里不能合并关系E_1与关系R

“多对多” 二元联系的转换

⑥ **多对多**

- 将转换成3个关系模式



多对多联系

$E_1(k1, a)$

- 候选码是 k1

$E_2(k2, b)$

- 候选码是 k2

$R(k1, k2, r)$

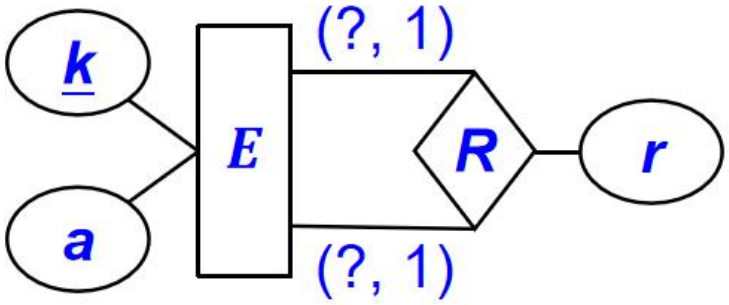
- (k1, k2)共同构成关系R的候选码
- k1和 k2是关系R的两个外候选码

联系的转换

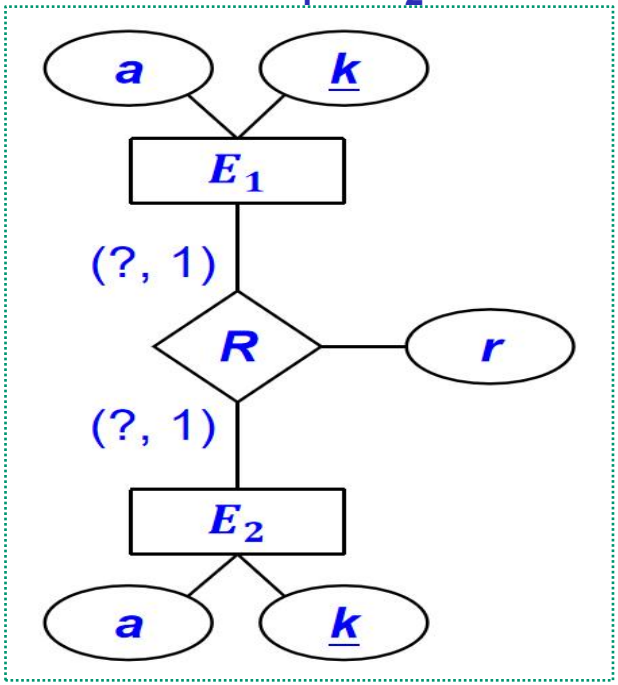
- **多个实体集之间的多元联系**
  - 每个实体集转换成一个关系模式
  - 联系被单独转换成一个关系模式
    - 其属性包括：
      a) 联系自身的属性
      b) 参与联系的每个实体集对应关系的主候选码
  - 转换得到的关系模式的候选码一般由所有参与联系的实体集所对应关系模式的候选码联合组成

- **单个实体集内部的联系**
  - 首先，将该联系转换成两个实体集之间的二元联系
  - 再按照二元联系的处理方式转换成关系模式
  - 将两个实体集转换得到的关系模式合并为一个关系

单个实体集内部的联系（一对一）



根据在联系R中所担当的角色，将实体集E划分为E_1和E_2两个实体集



- 转换成三个关系模式

  $E_1(k, a)$

  $E_2(k, a)$

  $R(k1, k2, r)$

其中：

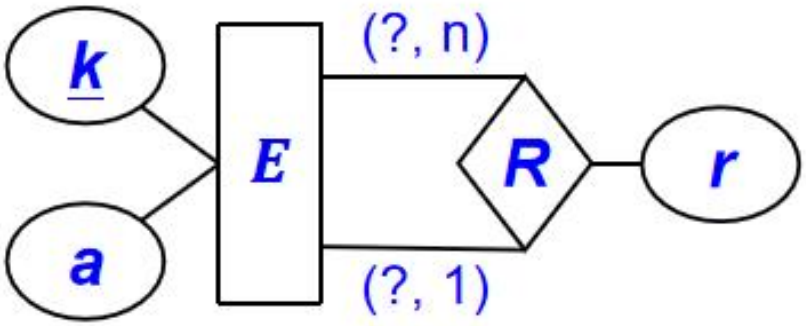
- *k1*和*k2*是R的两个候选码
- *k1*和*k2*也是R的两个外候选码，且*k1*对应E_1的主候选码k，*k2*对应E_2的主候选码k

- E_1和E_2其实是同一个关系，联系R也可以被合并到E_1或E_2对应的关系中去，合并后的关系模式如下：

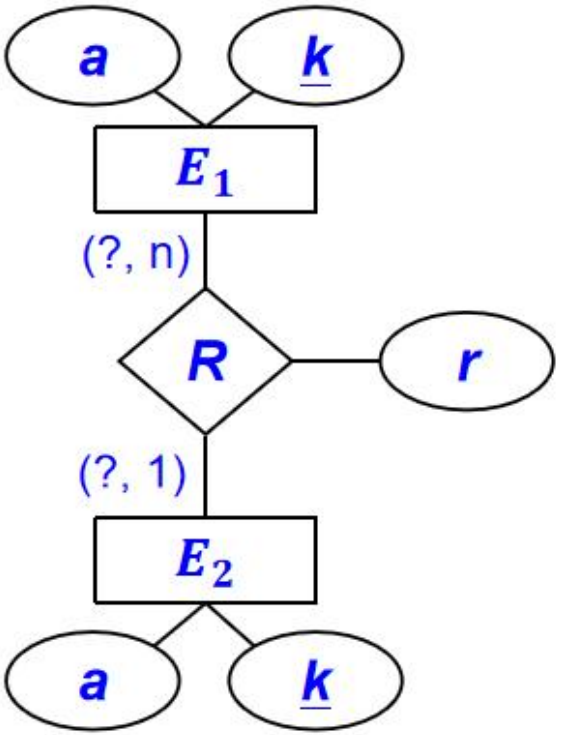
  $E(k, a, k', r)$

- 在最终方案中，必须区分清楚候选码k和外候选码k'在联系R中的角色！

单个实体集内部的联系（一对多）



根据在联系R中所担当的角色，将实体集E划分为E_1和E_2两个实体集



- 转换成两个关系模式（联系R被合并到‘单值参与’端）

  $E_1(k, a)$

  $E_2(k, a, k1, r)$

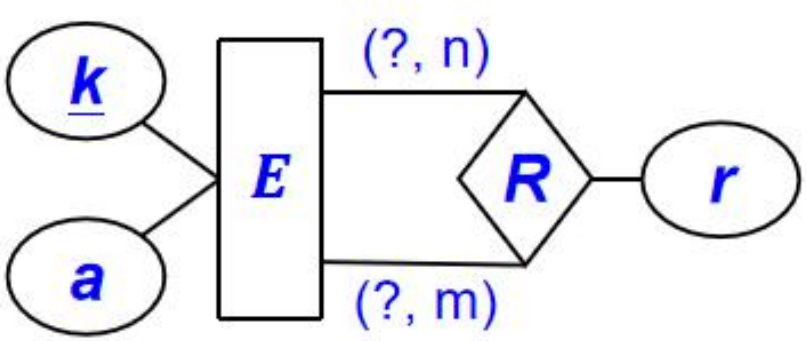
  其中：k1对应E_1的主候选码k

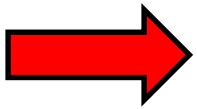
- 只需要保留关系E_2，得到最终的模式设计方案如下：

  $E(k, a, k', r)$

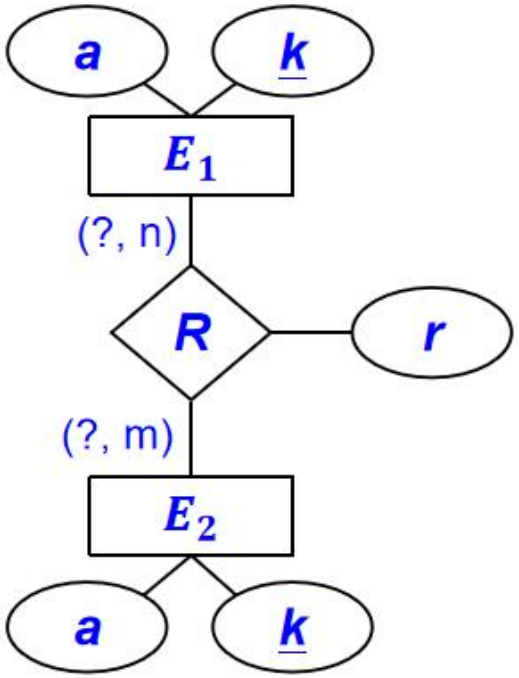
- 在最终方案中，候选码k对应‘单值参与’端，外候选码k'对应‘多值参与’端。

单个实体集内部的联系（多对多）





根据在联系R中所担当的角色，将实体集E划分为E_1和E_2两个实体集



- 转换成三个关系模式

  $E_1(k, a)$

  $E_2(k, a)$

  $R(k1, k2, r)$

其中：

- (k1,k2) 是R的主候选码
- k1和k2是关系R的两个外候选码，且k1对应E_1的主候选码k，k2对应E_2的主候选码k

- E_1和E_2只需保留一个，最终构成两个关系的设计方案

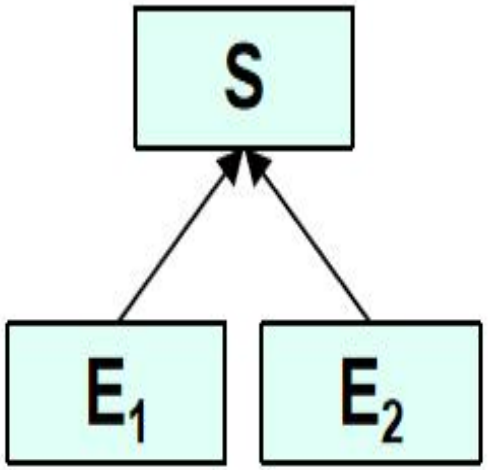
  $E(k, a)$

  $R(k1, k2, r)$

- 在最终方案中，必须区分清楚关系R中的两个外候选码$k1$和$k2$在联系中的角色！

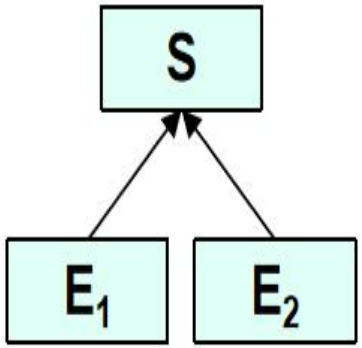
4. ISA联系的转换

- 假设超实体集S与其两个子实体集E_1和E_2构成如右图所示的继承关系。其中：
  - 实体集S的属性有$(k,A_1,A_2,\ldots,A_n)$，k是其标识属性
  - 实体集E_1的属性有$(B_{11},B_{12},\ldots,B_{1i})$
  - 实体集E_2的属性有$(B_{21},B_{22},\ldots,B_{2j})$



- 有三种不同的转换方式可供选择，可分别转换为：1个关系、2个关系、3个关系

ISA联系的转换



- 实体集S的属性有 $\{k, A_1, A_2, ..., A_n\}$，k是其标识属性
- 实体集E_1的属性有 $\{B_{11}, B_{12}, ..., B_{1i}\}$
- 实体集E_2的属性有 $\{B_{21}, B_{22}, ..., B_{2j}\}$

转换方式1：每一个实体集都将被单独转换为一个关系。

$S(\underline{k}, A_1, A_2, ..., A_n)$

$E_1(\underline{k}, B_{11}, B_{12}, ..., B_{1i})$

$E_2(\underline{k}, B_{21}, B_{22}, ..., B_{2j})$

转换方式2：只有最底层的叶子结点才会被转换为关系，并继承所有超实体集中的所有属性。

$E_1(\underline{k}, A_1, A_2, ..., A_n, B_{11}, B_{12}, ..., B_{1i})$

$E_2(\underline{k}, A_1, A_2, ..., A_n, B_{21}, B_{22}, ..., B_{2j})$

转换方式3：所有实体集被转换为单个关系，其中包含所有实体集中的属性。

$E(\underline{k}, A_1, A_2, ..., A_n, B_{11}, B_{12}, ..., B_{1i}, B_{21}, B_{22}, ..., B_{2j})$

- 请仔细体会这三种不同转换方案之间的区别！

数据库的逻辑结构设计

- 在从概念数据模型（EE-R）到逻辑数据模型（关系模型）的转换过程中，还可能存在一些需要处理的问题。
  1. 命名与属性域的处理
  2. 非原子属性处理
  3. 联系的转换
  4. ISA联系的转换

  5. 规范化
     - 对转换得到的关系模式进行关系规范化设计，确保所有关系模式至少满足第三范式(3NF)

  6. RDBMS性能调整
  7. 约束条件设置
  8. 关系视图的设计

逻辑设计的基本方法

6. 关系数据库管理系统（DBMS）

- 为满足数据库系统在性能、存储空间等方面的要求及其它限制条件所做的模式调整与修改。包括：

(1) 逆规范化

- 减少关系的连接运算次数，提高系统性能

(2) 关系的分割

(3) 尽量使用快照

逻辑设计的基本方法

(2) 关系的分割

- 调整每个关系的大小，提高在单个关系上的数据存取效率。
- 常用分割方法：
  - 水平分割
    - 将一个关系的元组集合划分为若干个不相交的子集，每个子集对应一个子关系模式；
  - 垂直分割
    - 将一个关系模式纵向分解成若干个子关系模式。（不同于规范化设计中的模式分解）

逻辑设计的基本方法

(3) 尽量使用快照

- 快照(snapshot)

```
create snapshot <snapshot-name> as <select-statement>;
```

  - 通过创建‘快照’，可以用一条select查询语句来为数据库中的某张表、或者为某个数据查询请求的执行结果创建一个不可修改的、持久化的数据备份。
  - 在执行create snapshot命令时，DBMS将执行对应的查询语句<select-statement>，并将查询执行结果以‘表’的形式持久存储在数据库中。
  - 在实际应用中，应用程序可以通过访问快照直接使用保存在数据库中的查询结果，不必再去执行对应的查询命令，有效提高了应用访问效率。特别适合于业绩统计、库存盘点、实时计费等应用场景。

- 快照的维护：DBMS提供了两种快照结果的更新方式
  - 由DBMS周期性地自动刷新快照的结果数据
  - 由用户通过执行快照刷新命令来手工刷新快照的结果数据

逻辑设计的基本方法

7. 约束条件的设置

- 完整性约束
- 安全性约束
- 数据类型约束
- 数据量的约束
- 重新定义每个表的候选键、主键及外键

8. 关系视图的设计

- 在关系模式基础上所设计的直接面向操作用户的视图，即用户的‘外模式’，它可以根据用户的需求随时定义或重新定义，一般 RDBMS 均通过‘视图’(view)来提供关系视图的功能。

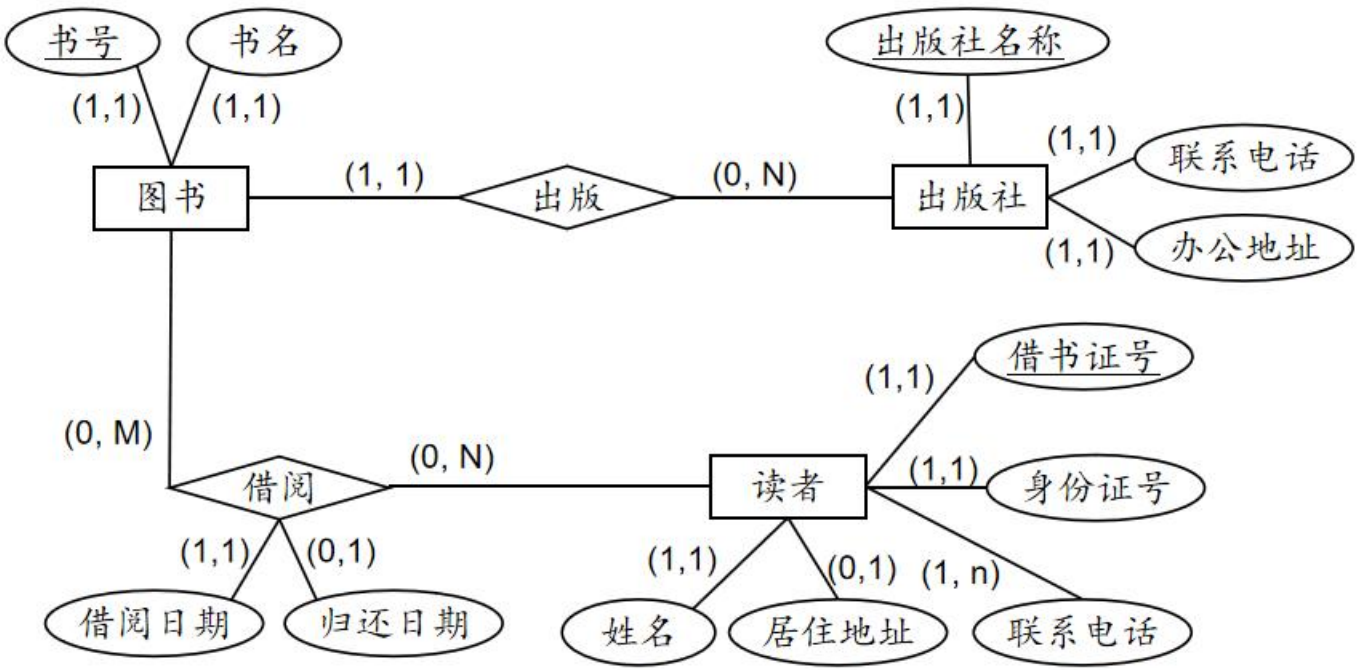
- 关系视图的作用
  - 提供数据的逻辑独立性
  - 能满足不同用户的不同数据需求
  - 提供一定的数据安全、保密功能

数据库逻辑设计案例

- 数据库逻辑设计的案例
  - 案例1：图书借阅管理
  - 案例2：CBA篮球联赛信息管理

案例1：图书借阅管理

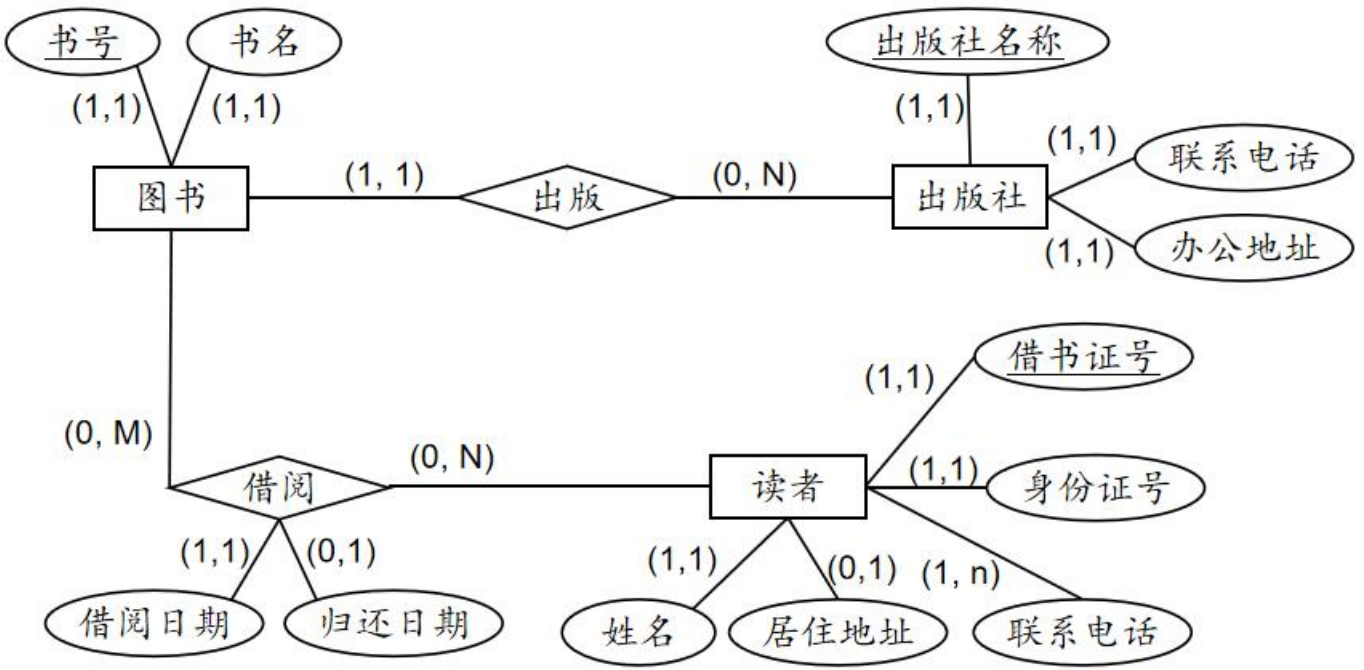
- 从ER模型向关系模型的转换



- 实体的转换（每个关系的候选码用下划线来表示）
  - 图书(书号, 书名)
  - 出版社(出版社名称, 联系电话, 办公地址)
  - 读者(借书证号, 身份证号, 姓名, 居住地址)
  - 读者电话(借书证号, 联系电话)

案例1：图书借阅管理

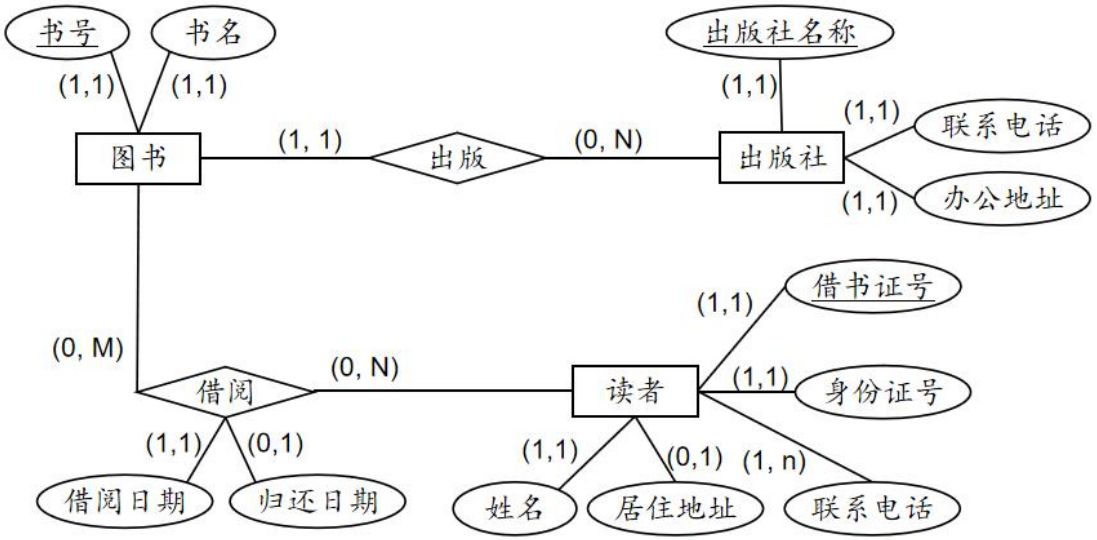
- 从ER模型向关系模型的转换：联系的转换



- ‘借阅’联系被转换为一个关系，‘出版’联系被合并到‘图书’关系中
  - 图书(书号, 书名, 出版社名称)
  - 出版社(出版社名称, 联系电话, 办公地址)
  - 读者(借书证号, 身份证号, 姓名, 居住地址)
  - 读者电话(借书证号, 联系电话)
  - 借阅(借书证号, 书号, 借阅日期, 归还日期)

案例1：图书借阅管理

- ER模型设计结果：



- 转换后的关系模型及其候选码的定义
  - 图书(书号, 书名, 出版社名称)
  - 出版社(出版社名称, 联系电话, 办公地址)
  - 读者(借书证号, 身份证号, 姓名, 居住地址)
  - 读者电话(借书证号, 联系电话)
  - 借阅(借书证号, 书号, 借阅日期, 归还日期)

- 思考题：
  ① 在ER模型中，有哪些模型信息没有表示出来？
  ② 各个关系上的候选码的定义是否正确？

案例1 图书借阅管理

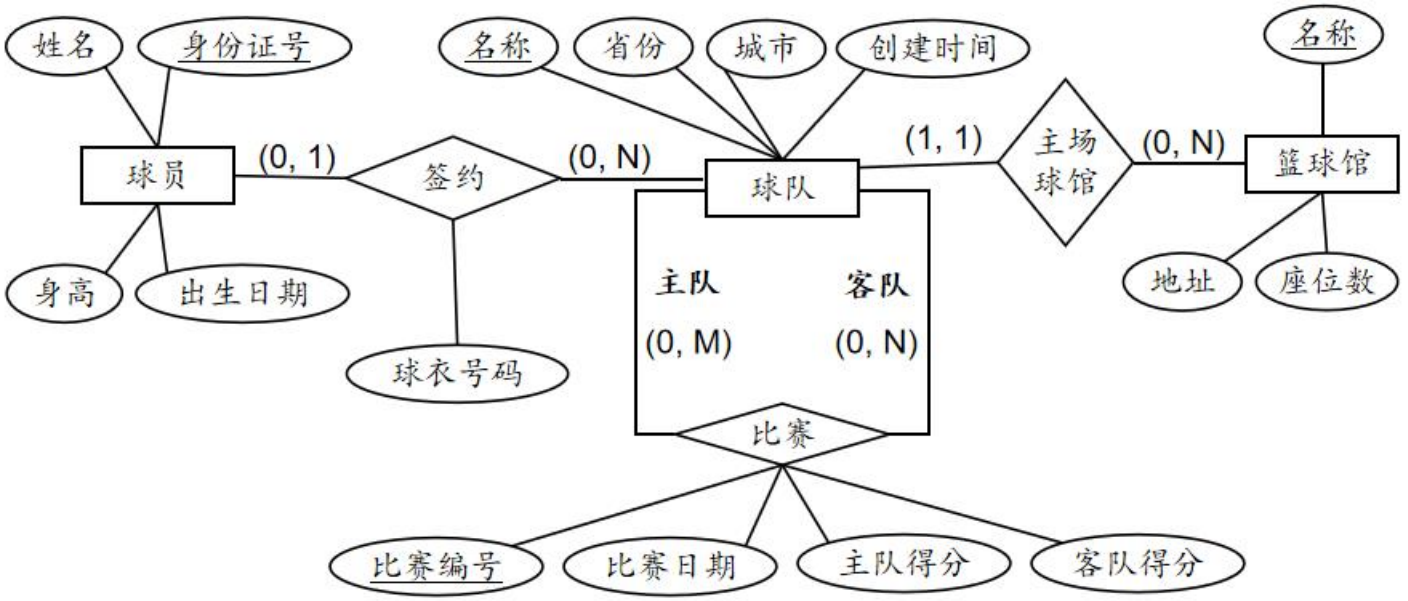
每个关系上的最小函数依赖集、所有候选码：

- 图书(书号, 书名, 出版社名称)
  - FDs：{书号 → (书名, 出版社名称)}
  - Keys：书号
- 出版社(出版社名称, 联系电话, 办公地址)
  - FDs：{出版社名称 → (联系电话, 办公地址)}
  - Keys：出版社名称
- 读者(借书证号, 身份证号, 姓名, 居住地址)
  - FDs：{借书证号 → (身份证号, 姓名, 居住地址)， 身份证号 → 借书证号}
  - Keys：借书证号 和 身份证号
- 读者电话(借书证号, 联系电话)
  - FDs：{ }
  - Keys：(借书证号, 联系电话)
- 借阅(借书证号, 书号, 借阅日期, 归还日期)
  - FDs：{(书号, 借阅日期) → (借书证号, 归还日期), (书号, 归还日期) → 借阅日期}
  - Keys：(书号, 借阅日期) 和 (书号, 归还日期)

所有关系都已满足3NF.

案例2：CBA篮球联赛信息管理

ER模型向关系模型的转换



（带下划线的是候选码）

- 球员(姓名, 身份证号, 身高, 出生日期, 球队名称, 球衣号码)
- 篮球馆(球馆名称, 地址, 座位数)
- 球队(球队名称, 省份, 城市, 创建时间, 主场球馆名称)
- 比赛(比赛编号, 比赛日期, 主队得分, 客队得分, 主队名称, 客队名称)