Lecture 5: 实体-关系模型

Database

Author: Forliage

Email: masterforliage@gmail.com

Date: June 10, 2025

College: 计算机科学与技术学院

Abstract

本讲笔记围绕实体–关系模型（E-R Model）的概念与应用展开，首先介绍了实体（Entity）与实体集的定义、属性（Attribute）与域（Domain）的分类（简单/复合、多值/派生）以及关系集（Relationship Set）的形式化表示和映射基数（cardinality）约束；随后，阐述了超键（Super Key）、候选键（Candidate Key）与主键（Primary Key）的概念及其在实体集和关系集中的运用；接着，通过 E-R 图示例说明了实体、关系与属性的图形化表示方式、参与约束与角色（Role）的标注；然后详细讲解了弱实体集（Weak Entity Set）的定义、标识关系及向关系模式的转换规则；在此基础上，介绍了扩展 E-R 特征，包括实体集的分层（特化/泛化）和聚集（Aggregation）；最后，概述了从 E-R 模型到关系模式的设计流程与关键决策，帮助读者构建规范、高效的数据库逻辑模型。
（该Abstract由ChatGPT-o4-mini-high生成）

Contents

1 实体集

在数据库设计中，Entity Set是从现实世界抽象出的一类"可区分对象"的集合，是构建ER模型的基本单元。

1.1 实体和实体集

1.Entity:

- Entity是现实世界中可被唯一识别的"物"或"事"，可以是具体的也可以是抽象的
- 每个实体由一组attributes来描述，例如student entity可能有id,name,age,gender,profession等属性

2.Entity Set:

- Entity Set是一类具有相同属性集合的entity的集合，即同一类型实体的全体。
- 例如"Student"实体集包含所有学生；"Customer"实体集包含所有客户;"Loan"实体集包含所有贷款
- 在关系模型中，一个实体集最终映射为一张关系(表)，实体映射为关系中的一行(元组)，属性映射为列(字段)

1.2 属性与域

1.属性:

- 属性是对实体某种特征或性质的描述。例如客户实体的属性包含customer-id,customer-name,customer-street,customer-city等
- 实体集customer对应关系的表头(列名)就是这些属性名称

2.域(Domain):

- 域也称"值域"或"取值集合"，指某个属性所允许的所有可能取值的集合
- 例如属性customer-id的域可以是所有符合特定编码规则的字符串；属性age的域可以是整数[0,150]

1.3 属性类型

1.简单属性 vs.复合属性

- 简单属性：原子不可分。如sex,age

- 复合属性：有多个分量属性组成，如name可以分解为first-name,middle-initial,last-name;address可分解为street,city,state,postal-code等

2.单值属性 vs. 多值属性

- 单值属性:对每个实体只能取一个值，例如某学生只有一个学号
- 多值属性:对某个实体可能取多个值，例如phone-numbers，一个客户有多个电话号码。在关系模型中多值属性要拆成单独的关联表来实现。

3.派生属性 vs. 基属性

- 基属性:在数据库中物理存储的属性
- 派生属性:可以由其它属性计算或推导出来，不必实际存储，例如age可以由birthday和当前日期计算得出。

1.4 实体集和关系模型

在从ER模型向关系模型转换时，每个实体集会成为一张关系表；

属性映射为表的列；

实体映射为表中的一条记录(元组);

对于多值属性或复合属性，需要额外设计关联表或将复合属性展开为多个列

2 关系集

2.1 关系和关系集

1.Relationship:

- 是两个或多个实体之间的一种关联
- 例如在银行模型中，实体集customer(客户)和实体集loan(贷款)之间存在"借款人"关系:

2.Relationship set:

- 是同一类关系的全体实例的集合
- 用于ER图中描绘实体集之间的联系
- 例如，定义了一个关系集borrower(customer-name, loan-number)，它包含所有"哪个客户借了哪笔贷款"的元组

2.2 形式化定义

假设有n个实体集$E_1, E_2, ..., E_n$，则一个关系集可以形式地看作：

$$R \subset E_1 \times E_2 \times ... \times E_n$$

其中$(e_1, e_2, ..., e_n) \in R$表示实体$e_i \in E_i (i = 1...n)$之间存在一次关联(一个关系实例)

2.3 关系集的属性

关系属性就像实体的属性一样，用来描述关系本身的特征。

例如`depositor(customer-name,account-number,access-date)`这个关系集:

- 它把`customer`与`account`关联起来，表示"哪个客户在什么时候访问了哪个账户"
- 其中`access-date`就是描述这次访问的一个属性

2.4 关系集的度

Degree表示一个关系集中参与的实体集数量(即上面定义里n的值)

最常见的binary relationship：degree=2，两个实体集之间的联系。

也可以有三元(ternary)或更高元(n-ary)的关系，但在实际建模中较少见。

2.5 映射基数

对于二元关系集，通常用映射基数来描述"一个实体最多能和多少个对方实体关联"：

1. 一对一:一个A中的实体最多关联一个B中的实体，反之亦然。例如:`country-president`
2. 一对多:一个A中的实体可以关联多个B中的实体，但每个B中的实体仅对应一个A。例如:`department-employee`
3. 多对一:对应上面一对多的反向视图：多个A对应同一个B，且每个B最多对应一个A。例如:`patient-doctor`
4. 多对多:一个A可以与多个B关联，同时一个B也可与多个A关联。例如:`student-course`

注意:在E-R图中，通常用在连线上标注“1”，“N” 或“M” 来表示各端的映射基数。

2.6 关系集到关系模式的映射

在将 ER 模型转换为关系模式时，每个二元关系集可以根据其映射基数不同，选择不同的实现方式：

1. 一对一：可将二者合并为一张表；或在任意一侧添加对方主键作为外键。

2. 一对多：在“多”端的关系表中，添加“1”端的主键作为外键。例如`employee`表中加`department_id`列，引用`department(dept_id)`
3. 多对多：需要拆出一个关联表（交叉表），其主键是两端实体的主键组合。

3 键

在E-R Diagram和关系模型中，Key是保证Entity或Relation中每个实例唯一可区分的属性或属性组合。

3.1 实体集的键

1.Super Key:

- 定义:实体集E中的属性集$S \subset$所有属性，若对同一个实体集中的任何两个不同实体e_1, e_2，它们在S上的取值总不相同，则称S是E的一个super key.
- Super key可以包含荣誉属性；只要能够唯一标识实体，就算超键。

2.Candidate Key:

- 定义：在所有超键中，如果一个超键K去掉任一属性后就不再能唯一标识实体（即再不是超键），则称K为一个候选键，也叫极小超键（minimal superkey）。
- 一个实体集可有多个候选键，例如学生实体集可能有 {学号} 和 {身份证号} 都是候选键。
- 在候选键中选择一个最具业务意义的，作为该实体集的主键（Primary Key）。

3.Primaey Key:

- 从所有候选键中选出的、用来唯一标识实体的那个，在关系模型里常被声明为 PRIMARY KEY。
- 主键有以下特点：唯一性,非空性

3.2 关系集的键

对于一个二元关系集$R \subset E_1 \times E_2$，参与实体集的主键组合本身就构成它的超键：

1.Super Key:将参与关系集的各实体集中对应的主键组合在一起，就能唯一标识一次关联。

2.Candidate Key:在大多数多对多（M:N）关系中，这个组合就是唯一的候选键；对于一对多（1:N）或一对一（1:1）的关系，有时可以只选组合中的一部分作为候选键（但要符合最小性和非空性）。选择哪个组合或子集作为主键，需要根据映射基数（Mapping Cardinality）及业务语义来决定，例如“作为外键的属性不应常变”、“关系属性不能为 NULL” 等。

3.3 映射基数对键的影响

- 一对一:关系集本身可以合并到任意一侧实体表，也可用一侧实体的主键做关系表的主键/外键。
- 一对多:多端实体集的主键与对应一端主键组合的超键可简化为“多端的主键 + 一端外键”形式；若一端的外键属性在多端表中一定非空，则这个外键本身也能唯一标识关联，可能成为候选键。
- 多对多:通常需在中间关系表中用“双方主键组合”做主键；这是最小且唯一的候选键。

4 实体-关系图

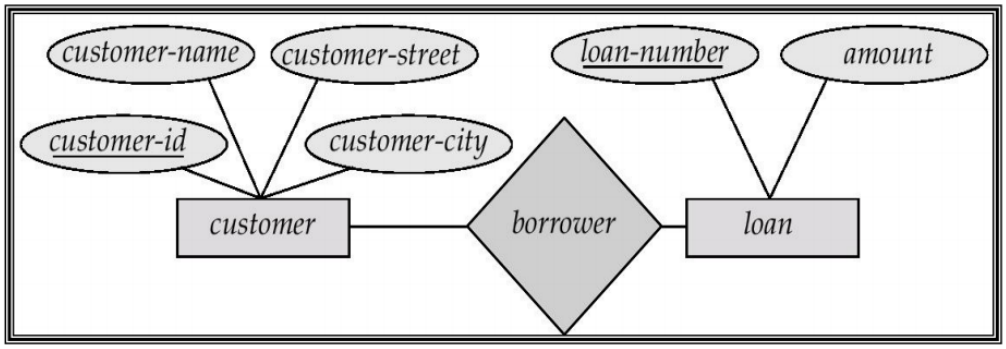

Figure 1

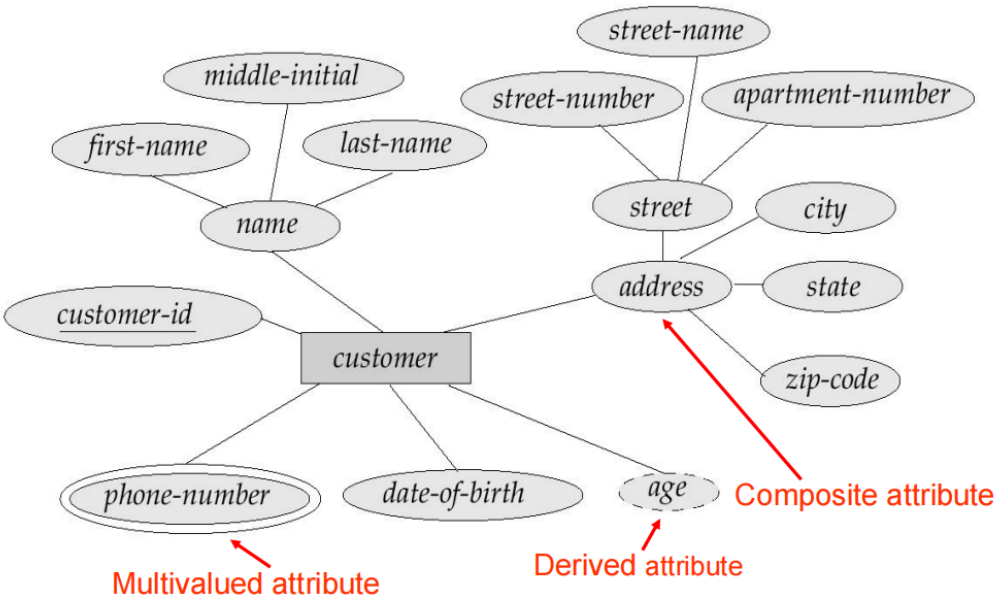

Figure 2

- 矩形表示Entity Set

- 菱形表示Relationship Set
- 线条将属性与实体集相连，将实体集与关系集相连。
- 椭圆表示属性：双椭圆表示多值属性；虚线椭圆表示派生属性。
- 下划线表示主键属性。

一个关系的实体集不必是不同的，例如，自环联系集。

Role:实体在关系中所起的作用，例如，label "manager" and "employee"被称为role；它们指定了员工实体如何通过"为...工作"关系集进行交互。

角色标签是可选的，用于阐明关系的语义。

我们通过在关系集和实体集之间绘制有向线(→)(表示"一个")或无向线(-)(表示"多个")来表达基数约束。

实体集在关系集中的参与情况：全参与(用双线表示)：实体集中的每个实体至少参与关系集中的一个关系。

部分参与：某些实体可能不参与关系集中的任何关系。

映射基数约束限定了一个实体与发生关联的另一端实体可能关联的数目上限。

一般来说，任何非二元关系都可以通过创建一个人工实体集，用二元关系来表示。用一个实体集E和三个新的关系集来替换实体集A, B和C之间的非二元关系R。

- 为E创建一个特殊的标识属性
- 将R的任何属性添加到E中
- 对于R中的每个关系(a_i, b_i, c_i)，创建R_A, R_B, R_C

5 弱实体集

在E-R模型中，Weak Enitity Set是一类自身没有完全主键、其存在依赖于另一个"强实体集"的实体集合。

5.1 什么是弱实体集

弱实体集是指"没有自己固有的主键"的实体集：它的每个实体不能单凭自身属性唯一标识，必须借助与某个强实体的关联才能区分。

弱实体集通常用双实线的矩形表示，关联它的标识性联系用双菱形表示。

5.2 弱实体集的要素

1.标识实体集：弱实体集依赖的强实体集，后者必须有一个完整的主键。

2.标识关系：把弱实体集和标识实体集连接起来的全参与、一对多联系:

- 全参与：弱实体必须对应某个强实体才存在
- 一对多：一个强实体可关联多个弱实体，但每个弱实体仅对应一个强实体

3.部分码/辨别符：

- 在弱实体集中，用来区分同一标识实体下不同实体的"局部唯一属性"

4.弱实体集的主键：由标识实体集的主键加上自身的部分码共同组成。

5.3 转换到关系模式

在关系模式中，弱实体集section和payment会映射为表，但不再单独存储”标识实体的主键“，因为它隐含在外键列中:

```
CREATE TABLE course (
  course_id    INT       PRIMARY KEY,
  title        VARCHAR(50),
  credits      INT
);

CREATE TABLE section (
  sec_id       INT          NOT NULL,
  semester     CHAR(6)      NOT NULL,
  year         INT          NOT NULL,
  course_id    INT          NOT NULL,
  PRIMARY KEY (course_id, sec_id, semester, year),
  FOREIGN KEY (course_id) REFERENCES course(course_id)
);

CREATE TABLE loan (
  loan_number VARCHAR(20) PRIMARY KEY,
  amount       DECIMAL(10,2)
);

CREATE TABLE payment (
  payment_number INT       NOT NULL,
  payment_date   DATE      NOT NULL,
  payment_amount DECIMAL(10,2),
  loan_number    VARCHAR(20) NOT NULL,
  PRIMARY KEY (loan_number, payment_number),
  FOREIGN KEY (loan_number) REFERENCES loan(loan_number)
);
```

section 表中没有再声明单独的 course_id 属性为主键，因为 course_id 已包含在复合主键里。

外键 (course_id) 与标识关系相对应，确保“无主课程的节次不能存在”。

6 扩展实体-关系特征

6.1 概述

扩展E-R特征是在传统E-R模型基础上，增强了对复杂数据结构的建模能力，主要包括以下几个特征：

- 分层实体集：特化、泛化、设计约束
- 聚集

6.2 实体集的分层

6.2.1 特化、具体化

含义：一种自顶向下的设计过程。

目的：将一个实体集中区分出子类，这些子类具有区别于其它实体的特性。

过程：

- 从一个高层次的实体集出发，将具有特殊属性或关系的实体划分到子集
- 子集可以拥有额外的属性或参与额外的关系，而这些在父实体集中不存在

属性继承：低层次的实体集会继承高层次实体集的所有属性和关系参与情况。

6.2.2 泛化、普遍化

含义：一种自底向上的设计过程

目的：将多个具有共同特征的实体集合并成一个高层次的实体集

过程：

- 从多个低层次的实体集出发，找出它们的共性，构成一个高层次的实体集
- 泛化和特化是逆过程，在E-R图中表示方法相同

注意：Specialization和Generalization可以互换使用，视设计场景而定。

6.2.3 特化/泛化设计约束

设计约束可以细化建模的规则，主要包括:

(1)基于哪些实体可以成为子集成员的约束

条件定义的：通过明确条件来决定实体是否属于某个子集。

用户定义的：根据业务需求人工定义，不依赖属性值。

(2)子集之间的互斥性约束

不相交：一个实体只能属于一个低层次实体集。E-R图中标记为disjoin或Disj

可重叠：一个实体可以同时属于多个低层次实体集。

(3)完全性约束：指定高层实体集中是否必须出现在至少一个低层实体集中。

完全泛化：高层实体集中的实体必须属于至少一个低层实体集。E-R 图中 ISA 使用双线表示。

部分泛化：高层实体集中的实体可以不属于任何低层实体集。E-R 图中 ISA 使用单线表示。

6.3 聚集

问题：如何表达关系之间的关系？有些场景中，关系本身还需要参与到另一个关系中。

解决方案→ 聚集（Aggregation）：

- 将关系（如 works-on）视为一个抽象实体，可以再与其他实体形成关系
- 这样可以避免冗余信息，并清晰表达“关系之间的关系”。

优势：

- 支持关系之间的关系建模。
- 允许将关系抽象为新实体进行管理。

7 实体-关系模式数据库设计

7.1 E-R 数据库模式设计流程

(1)需求分析（Requirement analysis）

- 目的：明确系统需要处理哪些数据、支持哪些应用程序、完成哪些操作。
- 产出：系统需求文档，作为后续设计的依据。

(2)概念数据库设计（Conceptual database design）

- 使用 E-R 模型（或类似的高层数据模型）描述数据和约束。

- 目的是提供一个高层次的抽象描述，让业务人员、开发人员都能理解数据库结构。

(3)逻辑数据库设计（Logical database design）

- 将概念设计转化为逻辑数据库模式，即表（tables）。
- 包括模式优化：关系规范化（Normalization），消除冗余和异常，确保数据一致性和高效性。

(4)物理数据库设计（Physical database design）

- 涉及索引（Indexing）、聚簇（clustering）、数据库调优（database tuning）。
- 目标是优化数据库的性能，提升查询和事务处理速度。

7.2 E-R 设计决策

(1)用属性（attribute）还是实体集（entity set）表示对象？

决策依据：

- 如果对象只有简单的名字或单值特性 ,可用属性。
- 如果对象本身有多个属性需要描述 , 应定义为实体集。

电话（phone）可以作为属性存储在 employee 中，但如果需要记录多个电话和电话的其他信息（位置、类型、颜色），则应定义 phone 为独立实体集。

设计原则：

- 一个对象不能同时作为实体和属性。
- 实体集之间才能建立联系，属性不能单独与实体集建立联系。

(2)用实体集（entity set）还是关系集（relationship set）表示？

决策依据：

- 如果要描述两个实体之间发生的“动作” , 用关系集。
- 如果描述的是对象本身 , 用实体集。

“借款人（borrower）” 是客户和贷款之间的关系，建模为关系集。“贷款（loan）” 是一个独立实体，建模为实体集。

设计影响因素：

- 关系的映射基数（mapping cardinality）会影响建模方式。
- 关系集的设计有助于更清晰表达实体之间的交互行为。

(3)用实体的属性（attribute of an entity）还是关系（relationship）表示？

决策依据:需要从对象的独立性和减少数据冗余角度权衡。

在 student 表中直接存 supervisor-id、supervisor-name 等 supervisor 信息，会导致冗余。更合理的设计是将 supervisor 定义为单独实体集，student 和 supervisor 之间通过关系（如 stu-sup）联系。

对于语义上独立的对象，应建模为实体集。关系集能更灵活表达动态变化的关联关系。

7.3 其它 E-R 设计决策

(4)使用三元或 n 元关系（ternary or n-ary relationship）还是一对二元关系（binary relationships）

如果三个或多个实体之间存在统一的语义关系,n元关系;若可拆解成多个二元关系且含义明确 ,二元关系

(5)使用强实体集（strong entity set）还是弱实体集（weak entity set）

如果实体有自己独立的主键,强实体集。如果实体必须依赖于另一个实体，且没有完全主键,弱实体集。

(6)使用特化/泛化（specialization/generalization）

有助于模块化设计，提高模型的灵活性和扩展性;通过特化/泛化机制，可以清晰表达实体之间的继承关系

(7)使用聚集（aggregation）

将 E-R 图的一部分组合成一个单独实体集，作为一个单元处理;便于处理关系之间的关系，提高设计的抽象层次

8 将实体-关系模式转换为表

8.1 概述

基本原则：

- 将 E-R 图转换为表，是实现关系数据库设计的关键步骤。
- 一个符合 E-R 图结构的数据库→ 可以用一组表（collection of tables）来表示。
- 每一个实体集（entity set）和关系集（relationship set）都对应一个唯一的表，表名通常与实体集或关系集同名。

8.2 表示实体集为表

强实体集（Strong Entity Set）：

- 直接转换为表，表的属性 = 实体的属性。
- 主键直接作为表的主键。

8.3 复合属性

转换规则：

- 将复合属性拆分成多个简单属性。
- 每个组成部分作为单独的属性出现在表中。

原则：提取所有“叶子节点”属性放入表中。

8.4 多值属性

转换规则：

- 为每一个多值属性创建单独的表。
- 新表的主键 = 原实体的主键 + 多值属性本身。

8.5 弱实体集

转换规则：

- 弱实体集自身不能独立存在，依赖于强实体集的主键。
- 转换成表时需要包含：自身的属性;强实体集的主键（作为外键）

8.6 关系集的表示

转换规则：关系集转为表时，表中应包含：参与关系的实体集的主键（作为外键）。关系集自身的属性（如果有）.

多对一 or 一对多:可以选择在“多”端表中添加一个额外属性，存储“1”端实体的主键。

8.7 表的冗余

Total Participation（完全参与）：直接将 "1" 端的主键加到 "多" 端表中作为属性。

Partial Participation（部分参与）：若将主键加到 "多" 端表中，可能导致 null 值出现。此时是否拆出单独表需根据场景权衡。

关系集与弱实体的冗余：连接弱实体与其标识强实体的关系集→ 通常是冗余的，因为弱实体表中已经包含了强实体的主键。

8.8 特化/泛化的表示

方法1：为高层实体集建一个表；为每个低层实体集单独建表，包含高层主键 + 低层特有属性。访问完整信息需联表查询。

方法2：为每个实体集建立表，包含本地属性 + 继承的属性。若特化是 total，可省略高层表（用 view 替代）。可能出现冗余，尤其是 street、city 出现在多个表中，数据一致性需维护。