Chapter 1

数据库建模与设计

1.1 实体关系模型

在基本层面上，数据库存储有关不同对象或实体以及这些实体之间的关联或关系的信息。例如，大学数据库可能存储有关学生、课程和注册的信息。学生和课程是实体，而注册是学生和课程之间的关系。同样，库存和销售数据库可能存储有关产品、客户和销售的信息。产品和客户是实体，销售是客户和产品之间的关系。

1.1.1 表示实体

在 ER 图中，我们通过矩形表示实体集。属性描述所属的实体。

我们可以从更小的部分形成一个属性；例如，我们根据街道号码、城市、邮政编码和国家/地区组成邮政地址。如果属性以这种方式由更小的部分组成，我们将属性分类为复合属性，否则分类为简单属性。

对于给定实体，某些属性可以具有多个值 - 例如，客户可以提供多个电话号码，因此电话号码属性是多值的。

属性有助于将一个实体与同一类型的其他实体区分开来。我们可以使用名称属性来区分客户，但这可能不是一个适当的解决方案，因为多个客户可能具有相同的名称。为了区分它们，我们需要一个保证对每个客户来说都是唯一的属性（或属性的最小组合）。一个或多个标识属性形成唯一键，在这种特殊情况下，我们将其称为主键。

在 ER 图中，属性表示为与其实体相连的带标签的椭圆形。构成主键的属性用下划线显示。任何复合属性的部分都绘制为连接到复合属性的椭圆形，多值属性显示为双线椭圆形。

1.1.2 表示关系

实体可以与实体形成关系。与实体一样，关系也可以具有属性。关系的两端都可以由不同数量的实体。关系两边的实体数量（关系的基数）定义关系的键约束。在 ER 图中，我们用带有名称的菱形表示关系集。关系的基数通常在关系菱形旁边标注。

1.1.3 部分和全部参与

实体之间的关系可以是可选的或强制的。在我们的示例中，我们可以决定只有购买了产品的人才被视为客户。另一方面，我们可以说客户是我们认识的人，并且我们希望他们会购买某种产品，也就是说，我们可以将从未购买过产品的人列为我们数据库中的客户。在第一种情况下，客户实体完全参与购买关系（所有客户都购买了产品，我们不能有没有购买产品的客户），而在第二种情况下，它有部分参与（客户可以购买产品）。这些被称为关系的参与约束。在 ER 图中，我们用实体框和关系菱形之间的双线表示总体参与。

1.1.4 实体还是属性

有时，我们会遇到这样的情况：我们想知道一个项目应该是一个属性还是一个实体。例如，电子邮件地址本身可以被建模为一个实体。如有疑问，请考虑以下经验法则：

是数据库直接感兴趣的项目吗？ 直接感兴趣的对象应该是实体，描述它们的信息应该存储在属性中。我们的库存和销售数据库真正对客户感兴趣，而不是他们的电子邮件地址，因此电子邮件地址最好建模为客户实体的属性。

该项目是否有自己的组件？ 如果有，我们必须找到一种表示这些组件的方法；一个单独的实体可能是最好的解决方案。

对象可以有多个实例吗？ 如果是这样，我们必须找到一种方法来在每个实例上存储数据。最简洁的方法是将对象表示为单独的实体。在我们的销售示例中，我们必须询问是否允许客户拥有多个电子邮件地址；如果是，我们应该将电子邮件地址建模为一个单独的实体。

该对象通常不存在或未知吗？ 如果是这样，它实际上只是某些实体的属性，最好将其建模为单独的实体，而不是通常为空的属性。

1.1.5 实体还是关系？

决定一个对象应该是实体还是关系的一个简单方法是将需求中的名词映射到实体，并将动词映射到关系。

1.1.6 中间实体

通常可以通过用新的中间实体（有时称为关联实体）替换多对多关系并通过多对一和一对多关系连接原始实体来从概念上简化多对多关系。

1.1.7 强实体还是弱实体

在数据库设计中，对于依赖于其他实体的实体，我们可以省略一些关键信息。例如，如果我们想要存储客户孩子的姓名，我们可以创建一个子实体并仅存储足够的关键信息以在其父实体的上下文中识别它。我们可以简单地列出孩子的名字，假设客户永远不会有多个孩子具有相同的名字。这里，子实体是弱实体，它与客户实体的关系称为识别关系。弱实体完全参与识别关系，因为它们不能独立于其所属实体而存在于数据库中。

在 ER 图中，我们用双线显示弱实体和标识关系，并用下划线虚线显示弱实体的部分键弱实体在其拥有（或强）实体的上下文中被唯一标识，因此弱实体的完整密钥是其自己（部分）密钥与其拥有实

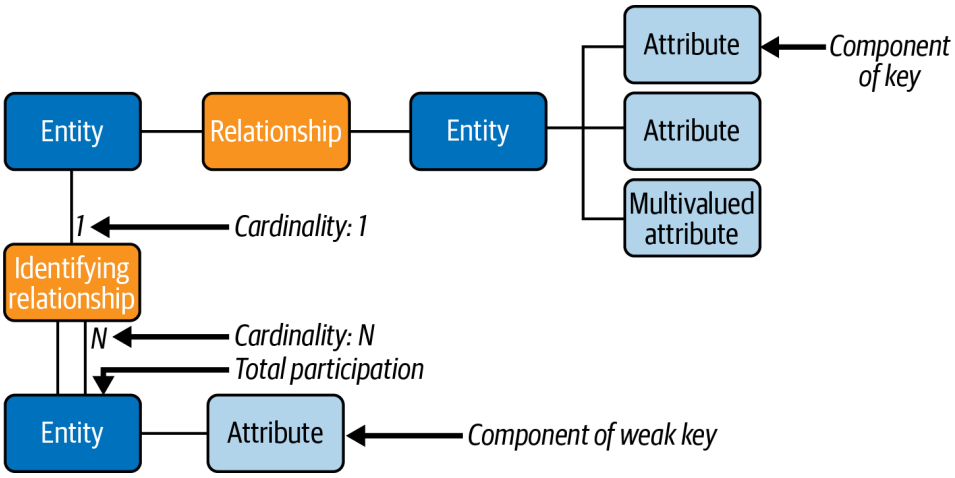

图 1.1: ER 图符号总结

体的密钥的组合。为了在我们的示例中唯一地标识孩子，我们需要孩子的名字和孩子父母的电子邮件地址。

Figure 1.1 显示了我们为 ER 图解释的符号的摘要。

1.2 数据库范式

范式的主要目标是减少数据冗余并提高数据完整性。规范化还促进了重新设计和扩展数据库结构的过程。

第一范式（1NF）有以下目标

- 消除各个表中的重复组。
- 为每组相关数据创建一个单独的表。
- 使用主键标识每组相关数据。

如果关系包含复合或多值属性，则它违反了第一范式。 相反，如果关系不包含任何复合或多值属性，则该关系处于第一范式。因此，如果该关系中的每个属性都具有适当类型的单个值，则该关系处于第一范式。

第二范式（2NF）的目标是

- 为适用于多个记录的值集创建单独的表。
- 将这些表与外键相关联。

记录不应依赖于表的主键（复合键，如果需要）以外的任何东西。

第三范式（3NF）又增加了一个目标

- 消除不依赖于键的字段。

记录中不属于该记录键的值不属于该表。 一般来说，只要一组字段的内容可能适用于表中的多个记录，您就应该考虑将这些字段放在单独的字段中。

1.3 规范化一个示例表

规范化之前的表如下所示：

Student#	Advisor	Adv-Room	Class1	Class2	Class3
1022	Jones	412	101-07	143-01	159-02
4123	Smith	216	201-01	211-02	214-01

1.3.1 第一范式：无重复组

表的每个属性应该只有一个字段。由于一个学生有多个班级，因此这些班级应在单独的表中列出。我们的非标准化表中的字段 Class1、Class2 和 Class3 表明存在设计问题。

电子表格通常具有同一属性的多个字段（例如，address1、address2、address3），但表格不应如此。这是看待这个问题的另一种方式：**对于一对多关系，不要将一侧和多侧放在同一个表中**。相反，通过消除重复组以第一范式创建另一个表 - 例如，使用 Class：

Student#	Advisor	Adv-Room	Class#
1022	Jones	412	101-07
1022	Jones	412	143-01
1022	Jones	412	159-02
4123	Smith	216	201-01
4123	Smith	216	211-02
4123	Smith	216	214-01

1.3.2 第二范式：消除冗余数据

请注意上表中每个 Student# 值的多个 Class# 值。 Class# 在功能上不依赖于 Student#（主键），因此这种关系不符合第二范式。

下面两个表演示了到第二范式的转换。我们现在有一个学生表：

Student#	Advisor	Adv-Room
1022	Jones	412
4123	Smith	216

和一个 Registration 表:

1.3.3 第三范式：消除不依赖于键的数据

在前面的示例中，Adv-Room（顾问的办公室号码）在功能上取决于顾问属性。解决方案是将该属性从“学生”表移至“教师”表，如下所示。

学生表像这样：

教师表像这样：

Student#	Class#
1022	101-07
1022	143-01
1022	159-02

Student#	Advisor
1022	Jones
4123	Smith

Name	Room	Dept
Jones	412	42
Smith	216	42