第七章 数据库设计

7.1 数据库设计概述
7.2 需求分析
7.3 概念结构设计
7.4 逻辑结构设计
7.5 物理结构设计
7.6 数据库的实施和维护
7.7 小结

7.1 数据库设计概述

- 数据库设计是建立数据库及其应用系统的技术，是信息系统开发和建设中的核心技术，具体说，数据库设计是指对于一个给定的应用环境，构造（设计）优化的数据库逻辑模式和物理结构，并据此建立数据库及其应用系统，使之能够有效地存储数据和管理数据，满足各种用户的应用需求（信息管理要求和数据操作要求）。这个问题是数据库在应用领域的主要研究课题。
- 在数据库领域内，常常把使用数据库的各类系统统称为数据库应用系统。

7.1 数据库设计概述

- 7.1.1 数据库设计的特点
- 7.1.2 数据库设计方法
- 7.1.3 数据库设计的基本步骤
- 7.1.4 数据库设计过程中的各级模式

7.1.1 数据库设计的特点

- 1、数据库建设的基本规律

  “三分技术，七分管理，十二分基础数据”是数据库设计的特点之一。

- 2、结构（数据）设计和行为（处理）设计相结合。

  数据库设计应该和应用系统设计相结合。也就是说，整个设计过程中要把数据库结构设计好对数据的处理设计密切结合起来。这是数据库设计的特点之二。

7.1.2 数据库设计方法

- 大型数据库设计，要求从事数据库设计的专业人员具备多方面的知识和技术：
  - 计算机的技术知识
  - 软件工程的原理和方法
  - 程序设计的方法和技巧
  - 数据库的基本知识
  - 数据库设计技术
  - 应用领域的知识

7.1.2 数据库设计方法

- 经过探索，提出了各种数据库设计的方法：
  - 新奥尔良方法；
  - 基于 E-R 模型的设计方法；
  - 3NF（第三范式）的设计方法；
  - 面向对象的数据库设计方法；
  - 统一建模语言方法等。
- 经过对数据库自动设计工具的研究和开发，数据库设计工具已经实用化和产品化，这些工具软件可以自动地或辅助设计人员完成数据库设计过程中的很多任务，已经普遍地用于大型数据库设计之中。

7.1.3 数据库设计的基本步骤

- 按照规范化设计方法，考虑数据库及其应用系统开发全过程，将数据库设计分为以下六个阶段：
  - 需求分析
  - 概念结构设计
  - 逻辑结构设计
  - 物理结构设计
  - 数据库实施
  - 数据库运行和维护

7.1.3 数据库设计的基本步骤

- 参加数据库设计的人员
  - 系统分析和数据库设计人员是数据库设计的核心人员，他们将自始至终参与数据库设计，他们的水平决定了数据库系统的质量。
  - 用户和数据库管理员在数据库设计中也是举足轻重的，他们主要参加需求分析和数据库的运行维护，他们的积极参与不但能加速数据库设计，而且也是决定数据库设计的质量的重要因素。
  - 程序员则在系统实施阶段参与进来，分别负责编制程序和准备硬件环境。
- 必要时使用数据库设计工具和CASE工具。

7.1.3 数据库设计的基本步骤

1. 需求分析阶段

- 准确了解与分析用户需求(包括数据与处理)
- 是整个设计过程的基础，是最困难、最耗费时间的一步

7.1.3 数据库设计的基本步骤

2. 概念结构设计

- 是整个数据库设计的关键
- 通过对用户需求进行综合、归纳与抽象，形成一个独立于具体DBMS的概念模型

7.1.3 数据库设计的基本步骤

3. 逻辑结构设计

- 将概念结构转换为某个**DBMS**所支持的数据模型
- 对其进行优化

7.1.3 数据库设计的基本步骤

4. 数据库物理设计阶段

- 为逻辑数据模型选取一个最适合应用环境的物理结构（包括存储结构和存取方法）

7.1.3 数据库设计的基本步骤

5. 数据库实施阶段

- 运用**DBMS**提供的数据语言、工具及宿主语言，根据逻辑设计和物理设计的结果
  - 建立数据库
  - 编制与调试应用程序
  - 组织数据入库
  - 并进行试运行

7.1.3 数据库设计的基本步骤

6. 数据库运行和维护阶段

- 数据库应用系统经过试运行后即可投入正式运行。
- 在数据库系统运行过程中必须不断地对其进行评价、调整与修改。

7.1.3 数据库设计的基本步骤

- 数据库设计与数据处理的设计相结合

在设计过程中，把数据库的设计和对数据库中数据处理的设计紧密结合起来，将这两个方面的需求分析、抽象、设计、实现在各个阶段同时进行，相互参照，相互补充，以完善两方面的设计。

7.1.4 数据库设计过程中的各级模式

- 数据库结构设计的不同阶段形成数据库的各级模式
  - 需求分析阶段，综合各个用户的应用需求；
  - 在概念设计阶段，形成独立于机器特点，独立于各个DBMS产品的概念模式（E－R图）；
  - 在逻辑设计阶段将E－R图转换成具体的数据库产品支持的数据模型（关系模式），形成数据库逻辑模式；
  - 根据用户处理的要求、安全性的考虑，在基本表的基础上再建立必要的视图（View），形成数据的外模式；
  - 在物理设计阶段，根据DBMS特点和处理的需求，进行物理存储安排，建立索引，形成数据库内模式。
- P210 图7.3 数据库的各级模式

7.2 需求分析

- 需求分析就是分析用户的要求。需求分析是设计数据库的起点，需求分析的结果是否准确地反映了用户的实际要求，将直接影响到后面各个阶段的设计，并影响到设计结果是否合理和实用。

7.2 需求分析

7.2.1 需求分析的任务

7.2.2 需求分析的方法

7.2.3 数据字典

7.2.1 需求分析的任务

- 需求分析的任务是通过详细调查现实世界要处理的对象，充分了解原系统工作概况，明确用户的各种需求，并确定新系统的功能。
- 新系统必须充分考虑今后可能的扩充和改变，不能仅仅按当前应用需求来设计数据库。

7.2.1 需求分析的任务

- 调查的重点是“数据”和“处理”，通过调查、收集与分析，获得用户对数据库的如下要求：
  - 信息要求
    - 在数据库中需要存储哪些数据
  - 处理要求
    - 完成什么处理、处理的时间要求、处理方式
  - 安全性与完整性要求

7.2.1 需求分析的任务

- 确定用户的最终需求是一件很困难的事
  - 用户缺少计算机知识，无法确定计算机究竟能作什么，不能作什么，往往不能准确地表达自己的需求，所提出的需求往往不断地变化。
  - 设计人员缺少用户的专业知识，不易理解用户的真正需求，甚至误解用户的需求。
- 设计人员必须不断深入地与用户交流，才能逐步确定用户的实际需求。

7.2.2 需求分析的方法

- 调查用户需求的具体步骤
  - 调查组织机构情况
  - 调查各部门的业务活动情况
  - 明确用户需求
  - 确定新系统的边界

7.2.2 需求分析的方法

- 调查方法可以根据不同的问题和条件而不同，常用的方法有：
  - 跟班作业
  - 开调查会
  - 请专人介绍
  - 询问
  - 设计调查表请用户填写
  - 查阅记录

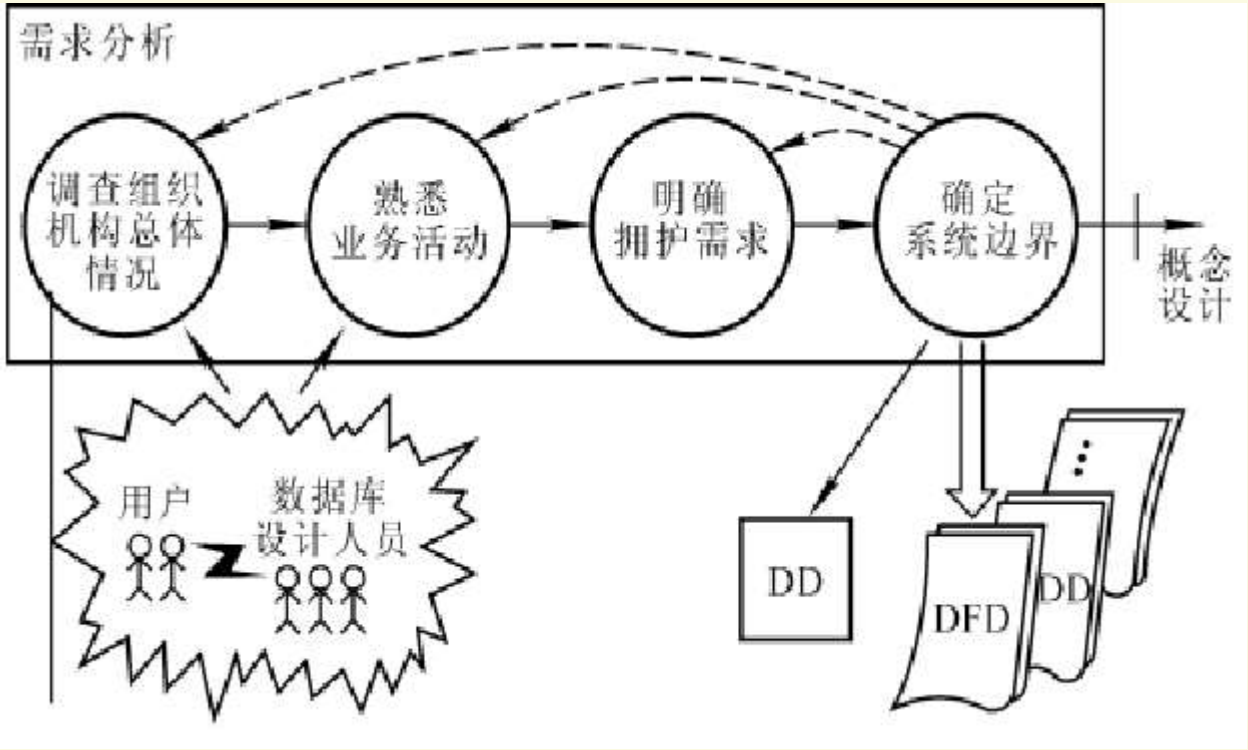
7.2.2 需求分析的方法

- 结构化分析方法

调查了解了用户的需求以后，还需要进一步分析和表达用户的需求。结构化分析方法（**Structured Analysis**，简称**SA**方法）是一种简单实用的方法，从最上层的系统组织机构入手，采用自顶向下、逐层分解的方式分析系统。

7.2.2 需求分析的方法

- 下图描述了需求分析的过程



7.2.3 数据字典

- 数据字典是系统中各类数据描述的集合，是进行详细的数据收集和数据分析所获得的主要成果。
- 数据字典在数据库设计中占有很重要的地位。
- 数据字典通常包括数据项、数据结构、数据流、数据存储和处理过程五个部分。

7.2.3 数据字典

1. 数据项

数据项是不可再分的数据单位。对数据项的描述通常包括以下内容：

数据项描述 = {数据项名，数据项含义说明，别名，数据类型，长度，取值范围，取值含义，与其他数据项的逻辑关系，数据项之间的联系}

2. 数据结构

数据结构反映了数据之间的组合关系。一个数据结构可以由若干个数据项及数据结构组成。对数据结构的描述通常包括以下内容：

数据机构描述 = {数据结构名，含义说明，组成：{数据项或数据结构}}

7.2.3 数据字典

3. 数据流

数据流是数据结构在系统内传输的路径。对数据流的描述通常包括以下内容：

数据流描述 = {数据流名，说明，别名，数据流来源，数据流去向，组成：{数据结构}，平均流量，高峰期流量}

4. 数据存储

数据存储是数据结构停留或保存的地方，也是数据流的来源和去向之一。对数据存储的描述通常包括以下内容：

数据存储描述 = {数据存储名，说明，编号，输入的数据流，输出的数据流，组成：{数据结构}，数据量，存取频度，存取方式}

7.2.3 数据字典

5. 处理过程

处理过程的具体处理逻辑一般用判定表或判定树来描述。数据字典中只需要描述处理过程的说明性信息，通常包括以下内容：

处理过程描述 = {处理过程名，说明，输入：{数据流}，输出：{数据流}，处理：{简要说明}}

- 数据字典是关于数据库中数据的描述，而不是数据本身。数据字典在需求分析阶段建立，在数据库设计过程中不断修改、充实、完善。

7.2.3 数据字典

- 需求分析阶段（收集、分析）收集到的基础数据（用数据字典来表达）和一组数据流图（**Data Flow Diagrame**，简称**DFD**）是下一步进行概念设计的基础。
- 强调两点：
  - 收集数据应充分考虑到可能的扩充和改变，使设计易于更改，易于扩充。
  - 强调用户的参与，与用户共同承担责任。

7.3 概念结构设计

- 将需求分析得到的用户需求抽象为信息结构（即概念模型）的过程就是概念结构设计。
- 概念结构设计是整个数据库设计的关键。

7.3 概念结构设计

- 7.3.1 概念模型
- 7.3.2 E-R模型
- 7.3.3 *扩展的E-R模型
- 7.3.4 *UML
- 7.3.5 概念结构设计

7.3.1 概念模型

- 在需求分析阶段所得到的应用需求应该首先抽象为信息世界的结构，才能更好地、更准确地用某一DBMS实现这些需求。
- 概念结构的主要特点：
  - 能真实、充分地反映现实世界，包括事物和事物之间的联系。
  - 易于理解。
  - 易于更改。
  - 易于向关系、网状、层次等各种数据模型转换。
- 描述概念模型的有力工具是E-R图。

7.3.2 E-R模型

- P.P.S.Chen提出的E-R模型是用E-R图来描述现实世界的概念模型
  - 1、实体之间的联系
  - 2、E-R图
  - 3、一个实例

1、实体之间的联系

- （1）两个实体型之间的联系
  - 一对一联系（1:1）
  - 一对多联系（1:n）
  - 多对多联系（m:n）

1、实体之间的联系

- 一对一联系

如果对于实体集 **A** 中的每一个实体，实体集 **B** 中至多有一个（也可以没有）实体与之联系，反之亦然，则实体集 **A** 与实体集 **B** 具有一对一联系，记为 **1:1**。

例如，学校里面，一个班级只有一个正班长，而一个班长只在一个班中任职，则班级与班长之间具有一对一联系。

1、实体之间的联系

- 一对多联系

如果对于实体集 A 中的每一个实体，实体集 B 中有 n 个实体（n>=0）与之联系，反之，对于实体集 B 中的每一个实体，实体集 A 中至多只有一个实体与之联系，则实体集 A 与实体集 B 有一对多联系，记为 1:n。

例如，一个班级中有若干名学生，而每个学生只在一个班级中学习，则班级与学生之间具有一对多联系。

1、实体之间的联系

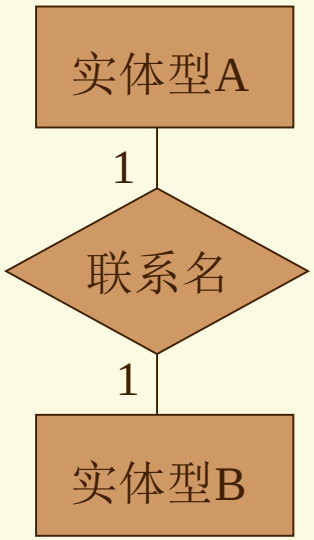
- 多对多联系

如果对于实体集 A 中的每一个实体，实体集 B 中有 n 个实体（n>=0）与之联系，反之，对于实体集 B 中的每一个实体，实体集 A 中也有 m 个实体（m>=0）与之联系，则实体集 A 与实体集 B 具有多对多联系，记为 m:n。

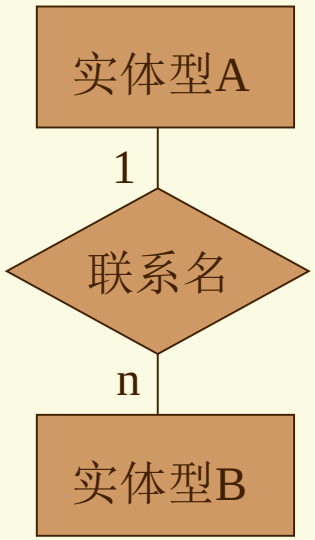
例如，一门课程同时有若干个学生选修，而一个学生可以同时选修多门课程，则课程与学生之间具有多对多联系。

1、实体之间的联系

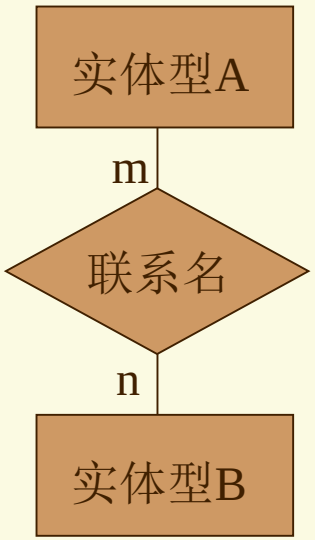
- 可以用图形来表示两个实体型之间的联系



1:1 联系



1:n 联系



m:n 联系

1、实体之间的联系

- （2）两个以上的实体型之间的联系

两个以上的实体型之间也存在也存在着一对一、一对多、多对多联系。

例如，对于课程、教师与参考书3个实体型，如果一门课程可以有若干个教师讲授，使用若干本参考书，而每一个教师只讲授一门课程，每一本参考书只供一门课程使用，则课程与教师、参考书之间是一对多的联系。

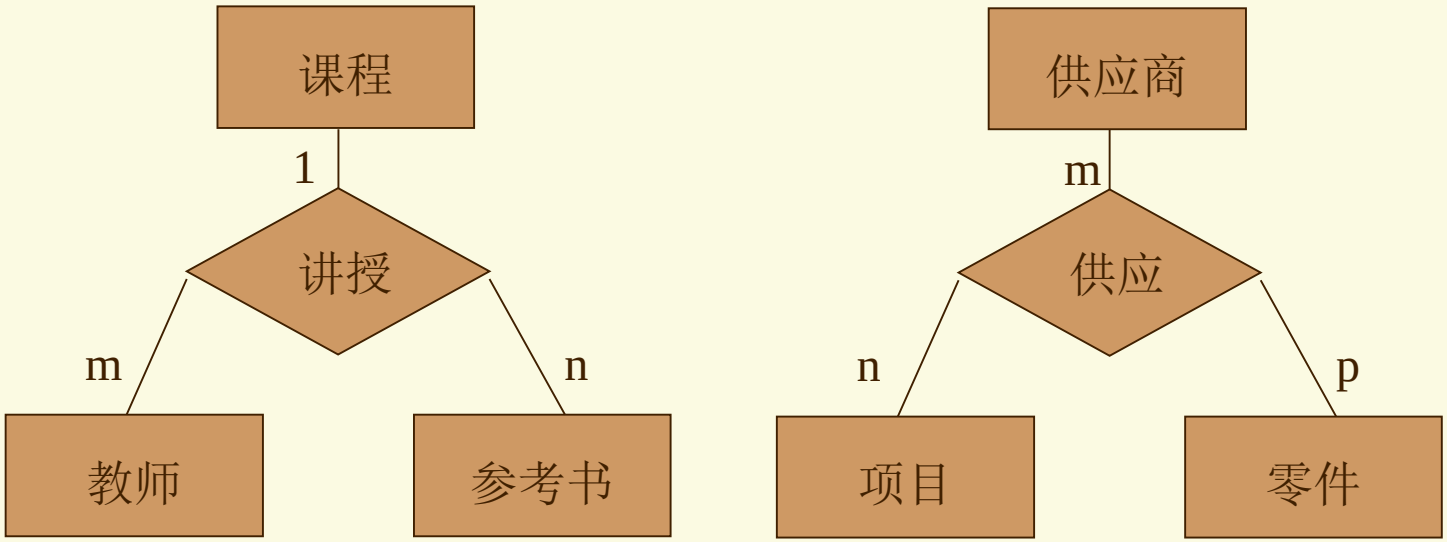
1、实体之间的联系

又如，有3个实体型：供应商、项目、零件，一个供应商可以供给多个项目多种零件，而每个项目可以使用多个供应商供应的零件，每种零件可由不同供应商提供。这样，供应商、项目、零件之间是多对多的联系

下面通过图示，给出这两个例子的实体型之间的联系

1、实体之间的联系

- 3个实体型之间的联系示例



一对多 联系

多对多 联系

1、实体之间的联系

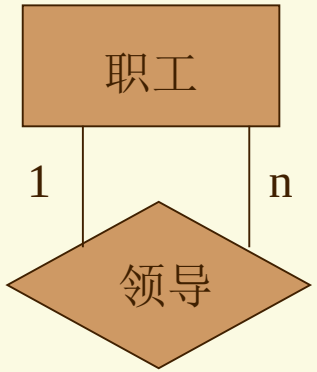
- 注意 3个实体型之间多对多的联系和3个实体型两两之间的（3个）多对多联系的语义是不同的。

1、实体之间的联系

- （3）单个实体型内的联系

同一实体集内的各实体之间也存在着一对一、一对多、多对多联系。

例如，职工实体型内部具有领导与被领导的联系，某一职工（干部）“领导”若干名职工，而一个职工仅被另外一个职工直接领导，因此这是一对多的联系。

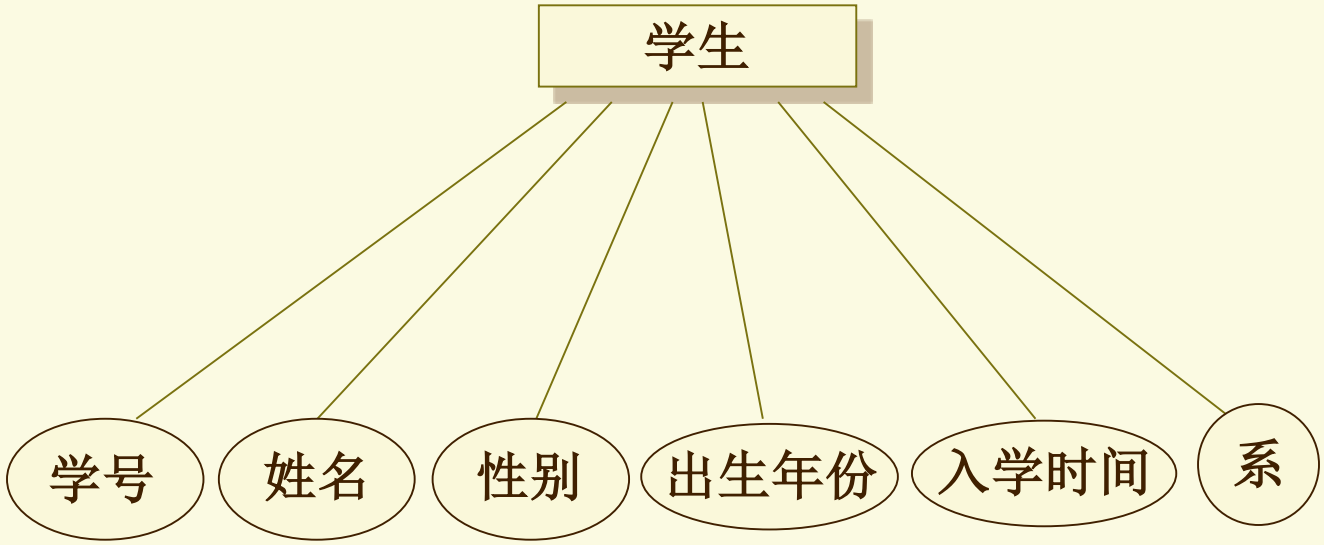


一个实体型之间一对多联系

2、E-R图

- E－R图提供了表示实体型、属性和联系的方法。
  - 实体型：用矩形表示，矩形框内写明实体名。
  - 属性：用椭圆形表示，椭圆中写入属性名，并用无向边将其与相应的实体连接起来。

  例如：学生实体具有学号、姓名、性别、出生年份、入学时间、所在院系等属性，用E-R图表示如下所示。



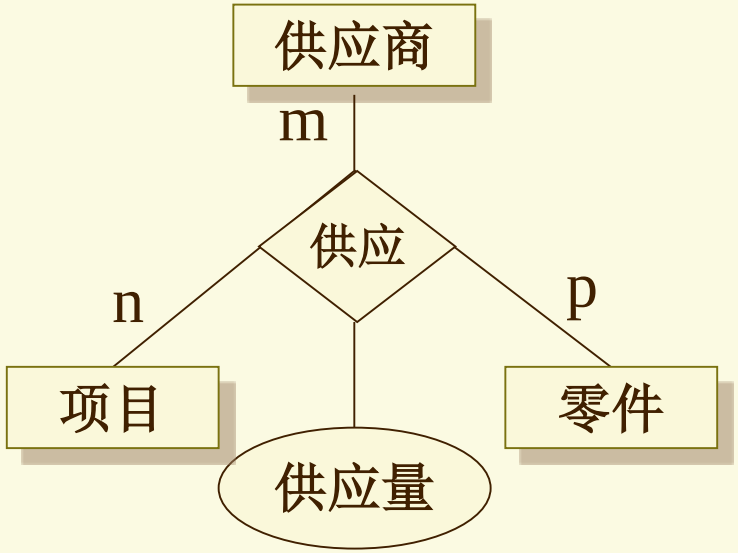
学生实体及属性

2、E-R图

- 联系：用菱形表示，菱形框内写明联系名，并用无向边分别与有关实体型连接起来，同时在无向边旁标上联系的类型（1:1，1:n，m:n）。

注意，联系也可以具有属性，这些属性也要用无向边与该联系连接起来。

例如，前面“供应商、项目、零件”的例子，在“供应”联系中具有“供应量”属性，表示某供应商供应了多少数量的零件给某个项目，其E-R图表示如下。



实体之间联系属性的表示

3、一个例子

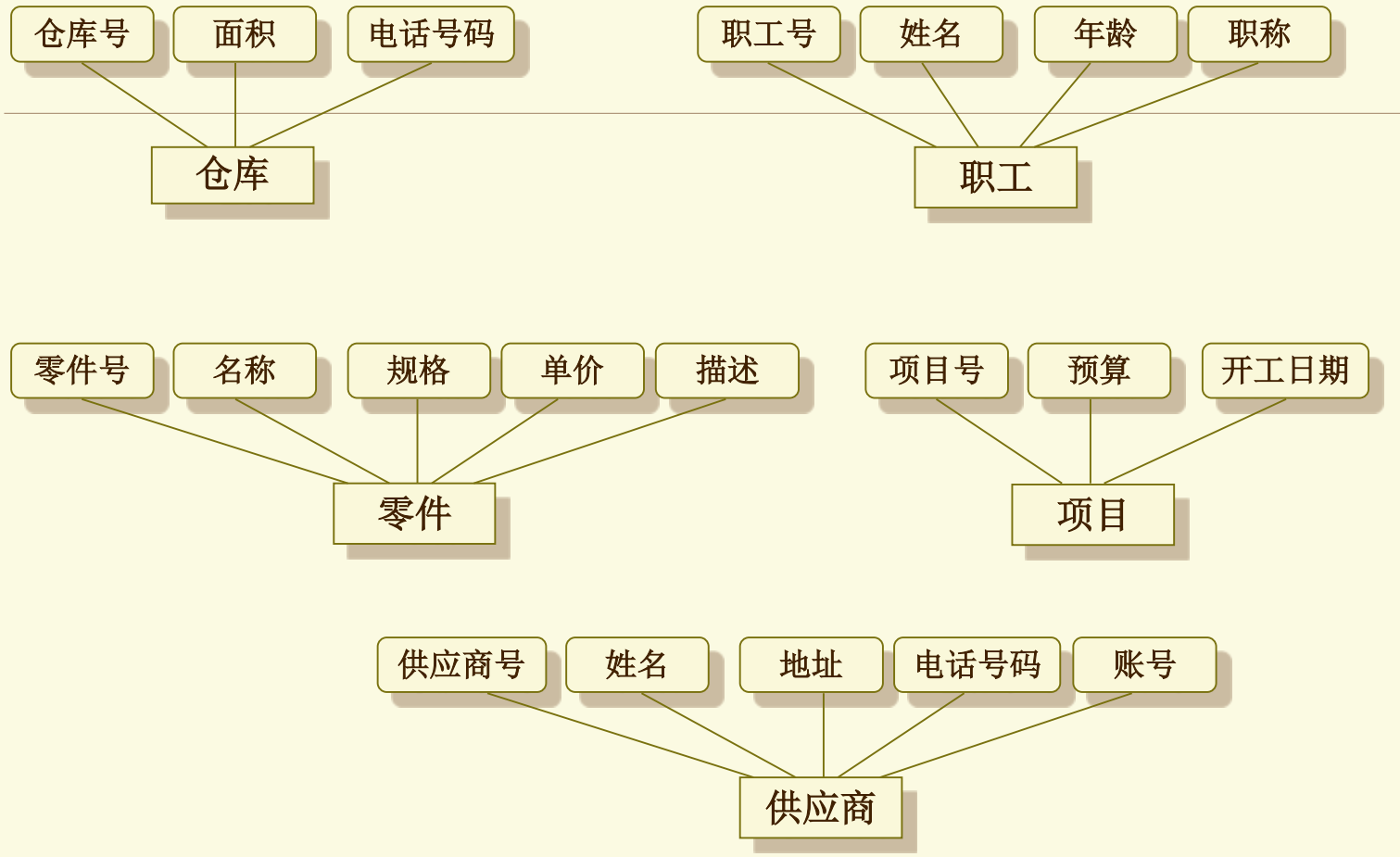
- 用E－R图来表示某工厂物资管理的概念模型。
- 物资管理涉及的实体：
  - 仓库：仓库号、面积、电话号码。
  - 零件：零件号、名称、规格、单价、描述。
  - 供应商：供应商号、姓名、地址、电话号码、帐号。
  - 项目：项目号、预算、开工日期。
  - 职工：职工号、姓名、年龄、职称。

3、一个例子

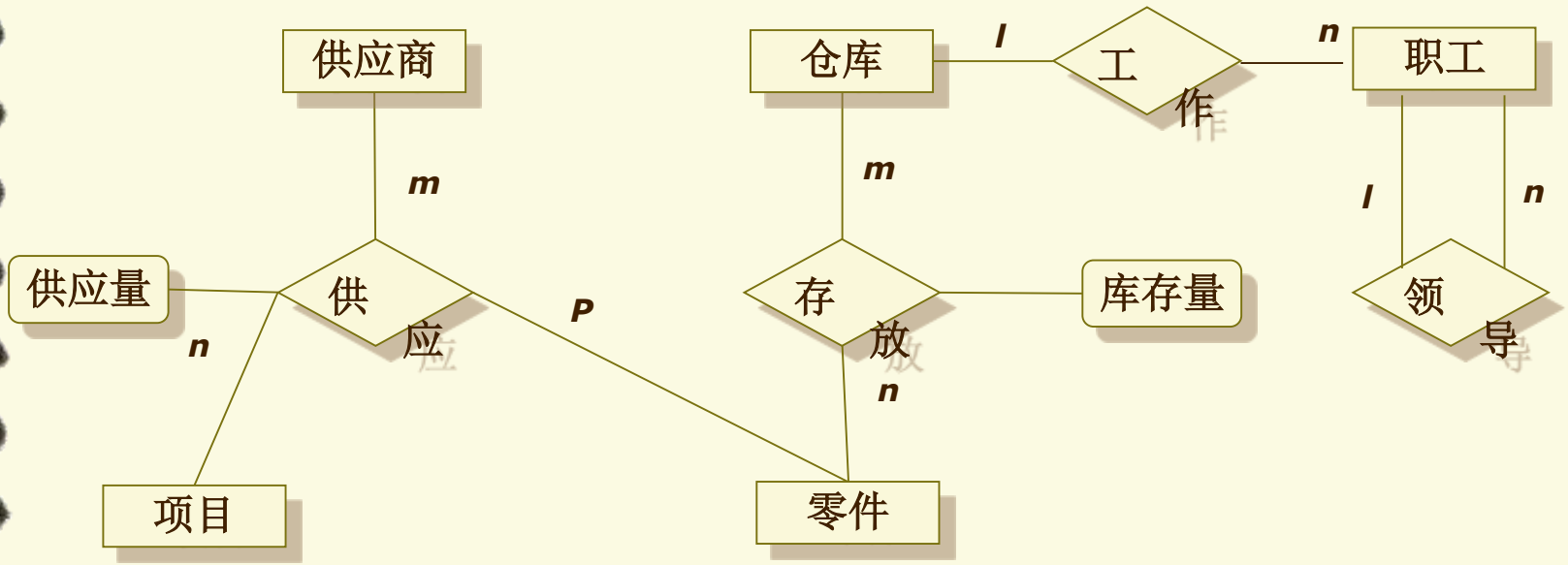
- 实体之间了联系
  - 仓库和零件：多对多；
  - 仓库和职工：一对多；
  - 职工实体集：自身具有一对多的关系；
  - 供应商、项目和零件：多对多。

3、一个例子

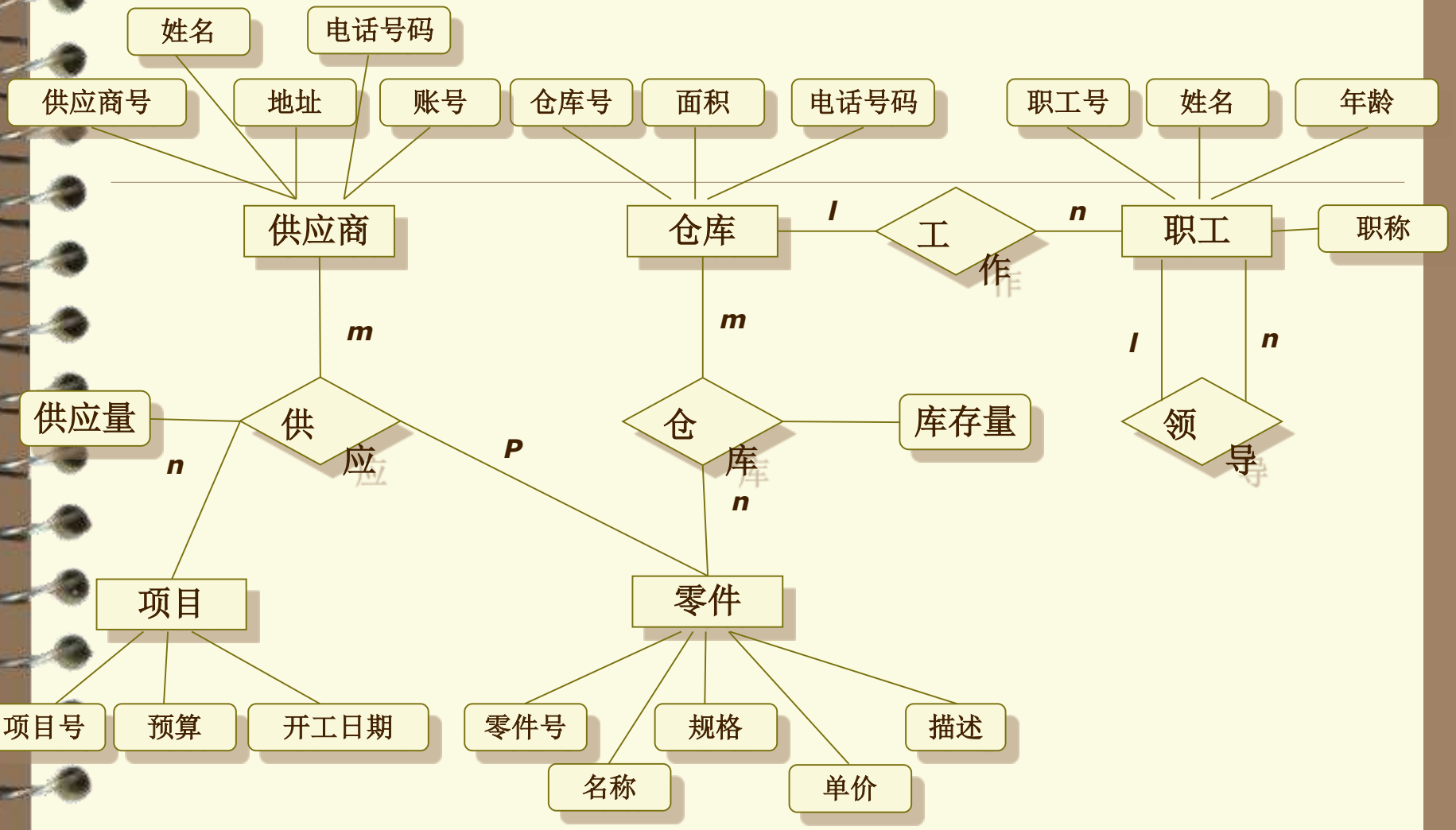
- 为了更清晰地表示实体及实体之间的联系，常常把实体及其属性用一幅图表示。实体及实体之间的联系图用另一幅图表示。
- 如下图。


仓库号
面积
电话号码
仓库
职工号
姓名
年龄
职称
职工
零件号
名称
规格
单价
描述
零件
项目号
预算
开工日期
项目
供应商号
姓名
地址
电话号码
账号
供应商

(a) 实体及其属性图



(b) 实体及其联系图



(C)工厂物质管理的完整实体联系图

7.3.3 扩展的E-R模型

- 实体－联系方法是抽象和描述现实世界的有力工具。用E－R图表示的概念模型独立于DBMS所支持的数据模型，它是各种数据模型的共同基础，因而比数据模型更一般、更抽象、更接近现实世界。

7.3.3 扩展的E-R模型

- ISA联系
  - 分类属性
  - 不相交约束与可重叠约束
  - 完备性约束
- 基数约束
- Part-of 联系

7.3.4 UML

- 表示E-R图的方法有若干种，使用统一建模语言UML是其中之一。

7.3.5 概念结构设计

- 1、实体与属性的划分原则
- 根据数据字典设计E-R图后，再进行必要的调整，一条原则是：能作为属性对待的，尽量作为属性对待。
- 实体与属性并没有形式上可以截然划分的界限，但可以给出两条准则：
  - 作为属性，不能再具有需要描述的性质，必须是不可分的数据项，不能包含其他属性。
  - 属性不能与其他实体具有联系，即E-R图中所表示的联系是实体之间的联系。
  - 例：P225

7.3.5 概念结构设计

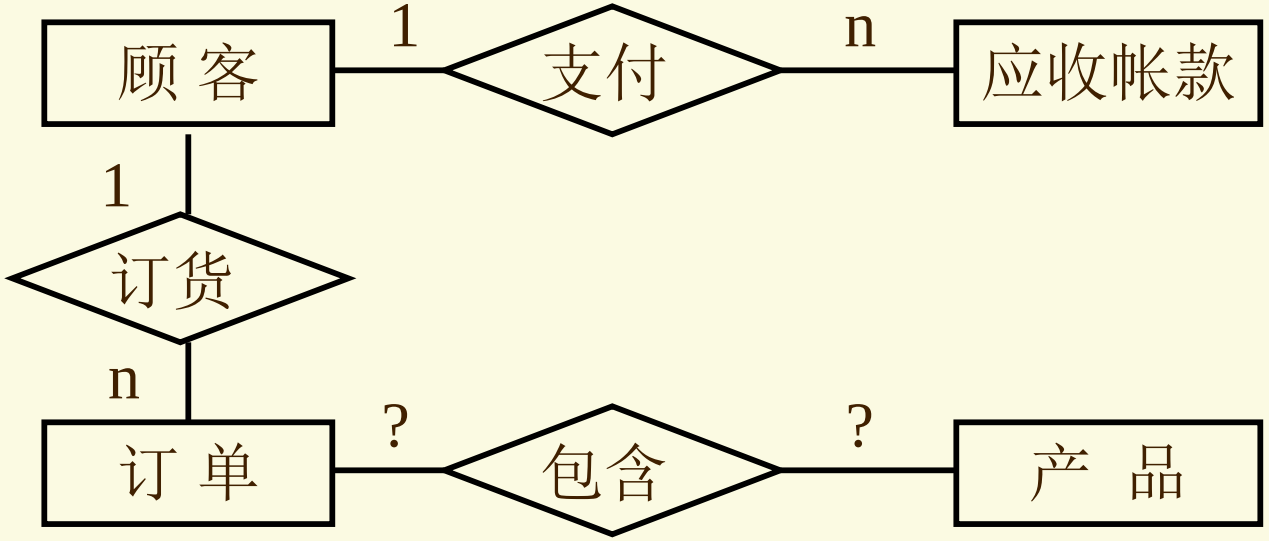
- 实例 销售管理子系统E-R图的设计

某工厂开发管理信息系统，包括物资管理、销售管理、劳动人事管理等子系统。该子系统的主要功能是：

- 处理顾客和销售员送来的订单；
- 工厂是根据订货安排生产的；
- 交出货物同时开出发票；
- 收到顾客付款后，根据发票存根和信贷情况进行应收款处理

7.3.5 概念结构设计

- 首先依据“订单”与“应收帐款”的数据流图，设计E-R图的草图

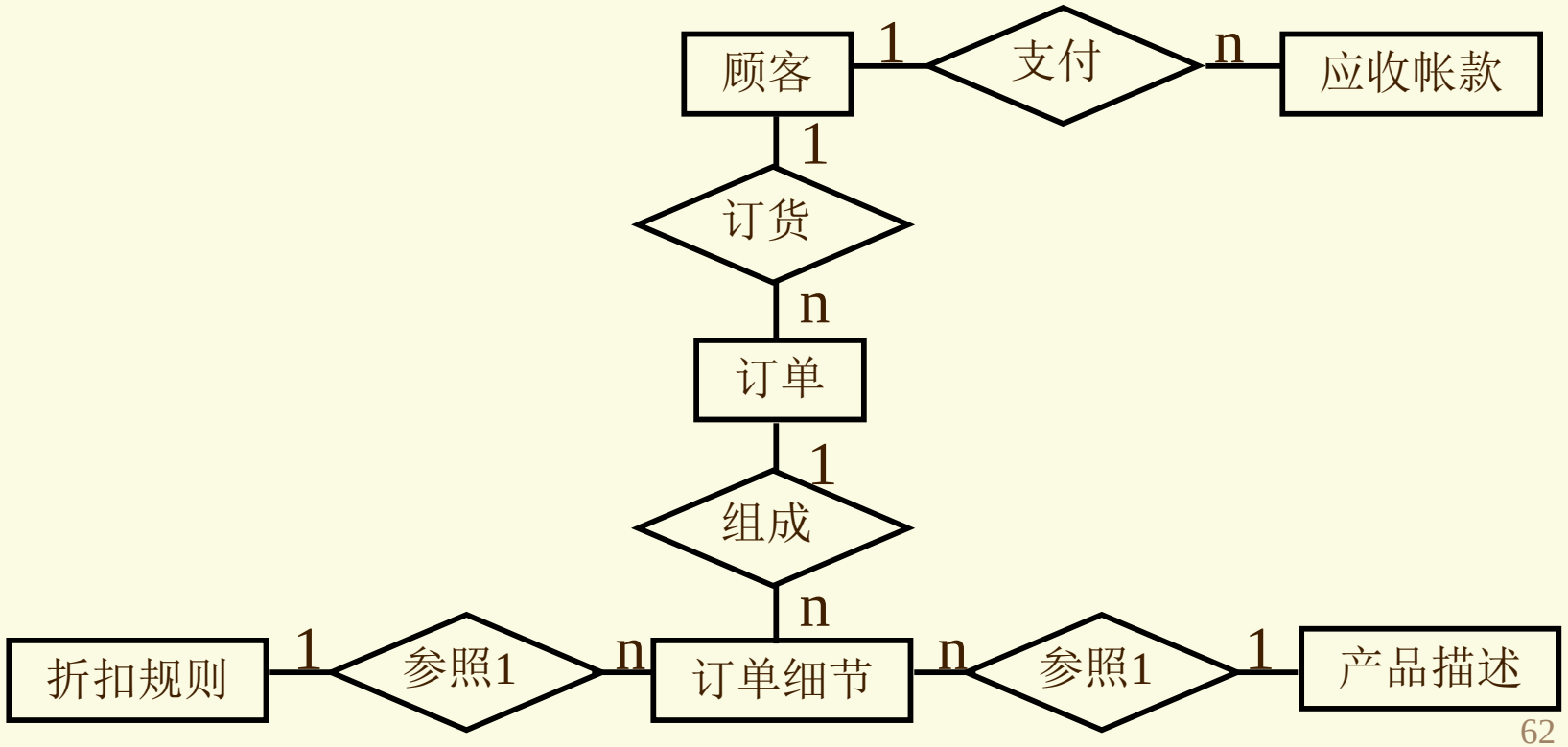


7.3.5 概念结构设计

- 参照数据流图和数据字典的详尽描述，遵循“属性”的两个准则，进行相应调整。得到实体及属性如下（含下划线的为标识码）：
  - 顾客：{顾客号，顾客名，地址，电话，信贷状况，帐目余额}
  - 订单：{订单号，顾客号，订货项数，订货日期，交货日期，工种号，生产地点}
  - 订单细则：{订单号，细则号，零件号，订货数，金额}
  - 应收帐款：{顾客号，订单号，发票号，应收金额，支付日期，支付金额，当前余额，货款限额}
  - 产品：{产品号，产品名，单价，重量}
  - 折扣规则：{产品号，订货量，折扣}

7.3.5 概念结构设计

- 最后得到销售管理子系统E-R图如下



7.3.5 概念结构设计

- 2、在开发一个大型信息系统时，最终采用的策略是自顶向下地进行需求分析，然后再自底向上的设计概念结构。即首先设计各子系统的分E-R图，然后将它们集成起来，得到全局E-R图。

7.3.5 概念结构设计

- **E-R图的集成一般需要分两步走：**
  - **合并。解决各分E-R图之间的冲突，将各分E-R图合并起来生成初步E-R图。**
  - **修改和重构。消除不必要的冗余，生成基本E-R图。**
- **图7.24，P227.**

7.3.5 概念结构设计

一.合并分E-R图，生成初步E-R图

- 由不同的设计人员进行设计的局部视图，在E-R图之间必定存在许多不一致的地方，称之为冲突。在合并E-R图时必须着力消除各个E-R图中的不一致。
- 各分E-R图之间的冲突主要有三类：
  - 属性冲突
  - 命名冲突
  - 结构冲突

一.合并分E-R图，生成初步E-R图

1. 属性冲突
   - 属性域冲突，即属性值的类型、取值范围或取值集合不同。
   - 属性取值单位冲突。
2. 命名冲突
   - 同名异议，即不同意义的对象在不同的局部应用中具有相同的名字。
   - 异名同义，即同一意义的对象在不同的局部应用中具有不同的名字。

一.合并分E-R图，生成初步E-R图

3. 结构冲突

- 同一对象在不同应用中具有不同的抽象。
- 同一实体在不同分E-R图中所包含的属性个数和属性排列次序不完全相同。
- 实体间的联系在不同的分E-R图中为不同的类型。
  - 例如 P229 图7.25

7.3.5 概念结构设计

二. 消除不必要的冗余，设计基本E-R图

- 在初步E-R图中，可能存在一些冗余的数据和实体间冗余的联系。应当予以消除，消除了冗余后的初步E-R图称为基本E-R图。
- 消除冗余主要采用分析方法，即以数据字典和数据流图为依据，根据数据字典中关于数据项之间逻辑关系的说明消除冗余。
- 分析 P230 图7.26 消除冗余 示例

二. 消除不必要的冗余，设计基本E-R图

- 并不是所有的冗余数据与冗余联系都必须加以消除，有时为了提高效率，不得不以冗余信息作为代价。如果人为地保留了一些冗余数据，则应把数据字典中数据关联的说明作为完整性约束条件。
- 除分析方法外，还可以用规范化理论来消除冗余。（了解）

7.3.5 概念结构设计

- [实例] 某工厂管理信息系统的视图集成
  - 分E-R图构成
    - P219 图7.11(c) 为物资管理子系统
    - P227 图7.23 为销售管理子系统
    - P230 图7.27 为劳动人事管理子系统
  - 基本E-R图
    - 图示（图7.28）
    - 集成过程解决的问题
      - 异名同义
      - 消除冗余

7.4 逻辑结构设计

- 概念结构是独立于任何一种数据模型的信息结构。逻辑结构设计的任务就是把概念结构阶段设计好的基本**E-R**图转换为与选用**DBMS**产品所支持的数据模型相符合的逻辑结构。
- 目前的数据库应用系统都采用支持关系数据模型的关系数据库管理系统，这里只介绍**E-R**图向关系数据模型的转换原则与方法。

7.4 逻辑结构设计

7.4.1 E-R图向关系模型的转换

7.4.2 数据模型的优化

7.4.3 设计用户子模式

7.4.1 E-R图向关系模型的转换

- E-R图向关系模型的转换要解决的问题是如何将实体和实体间的联系转换为关系模式，如何确定这些关系模式的属性和码。
- 关系模型的逻辑结构是一组关系模式的集合。
- E-R图则是由实体、实体的属性和实体之间的联系三个要素组成的。
- 转换就是将实体、实体的属性和实体之间的联系转换为关系模式。

7.4.1 E-R图向关系模型的转换

- 转换的一般原则：
  - 一个实体型转换为一个关系模式。实体的属性就是关系的属性，实体的码就是关系的码。
  - 一个 1:1 联系可以转换为一个独立的关系模式，也可以与任意一端对应的关系模式合并。
  - 一个 1:n 联系可以转换为一个独立的关系模式，也可以与 n 端对应的关系模式合并。
  - 一个 m:n 联系转换为一个关系模式。
  - 三个或三个以上实体间的一个多元联系可以转换为一个关系模式。
  - 具有相同码的关系模式可合并。

7.4.1 E-R图向关系模型的转换

- 分析P231 图7.28 中上部的E-R图转换的关系模型。
- 形成了一般的数据模型后，下一步就是向特定的RDBMS的模型转换。

7.4.2 数据模型的优化

- 数据库逻辑设计的结果不是唯一的。为了进一步提高数据库应用系统的性能，还应该根据应用需要适当地修改、调整数据模型的结构，这就是数据模型的优化。
- 关系数据模型的优化通常以规范化理论为指导，方法如下：

7.4.2 数据模型的优化

1. 确定数据依赖。
2. 对于各个关系模式之间的数据依赖进行极小化处理，消除冗余的联系。
3. 按照数据依赖的理论对关系模式逐一进行分析，考察是否存在部分函数依赖、传递函数依赖、多值依赖，确定各关系模式分别属于第几范式。

7.4.2 数据模型的优化

4. 按照需求分析阶段得到的处理要求，分析这些模式对于这样的应用环境是否合适，确定是否要对某些模式进行合并或分解。
5. 对关系模式进行必要的分解，提高数据操作的效率和存储空间的利用率。常用的两种分解方法是水平分解和垂直分解。

7.4.2 数据模型的优化

- 水平分解是把（基本）关系的元组分解为若干子集合，定义每一个子集合为一个关系，以提高系统的效率。
- 垂直分解是把关系模式 **R** 的属性分解为若干子集合，形成若干子关系模式。垂直分解可以提高某些事务的效率，但也可能使另外一些事务不得不执行连接操作，从而降低了效率。因此是否进行垂直分解取决于分解后 **R** 上的所有事务的总效率是否得到了提高。

7.4.3 设计用户子模式

- 将概念模型转换为全局逻辑模型后，还应该根据局部应用需求，结合具体DBMS的特点，设计用户的外模式。
- 利用关系数据库系统提供的视图（View），可以设计更符合局部用户需要的用户外模式。
- 定义数据库全局模式主要是从系统的时间效率、空间效率、易维护等角度出发。由于用户外模式与模式是相对独立的，因此在定义用户外模式时可以注重考虑用户的习惯与方便。

7.4.3 设计用户子模式

- 定义外模式应该：
  - 使用更符合用户习惯的别名

    用View机制可以在设计用户View时重新定义某些属性名，使其与用户习惯一致，以方便使用。
  - 可以对不同级别的用户定义不同的View，以保证系统的安全性。

    主要是防止非法用户访问本来不允许他们查询的数据，保证了系统的安全性。
  - 简化用户对系统的使用

    为了方便用户，可以将复杂查询定义为视图，简化用户的使用。

7.5 物理结构设计

- 数据库在物理设备上的存储结构与存取方法称为数据库的物理设计，它依赖于给定的计算机系统。为一个给定的逻辑数据模型选取一个最合适应用要求的物理结构的过程，就是数据库的物理设计。
- 数据库的物理设计通常分为两步：
  - 确定数据库的物理结构（存取方法和存储结构）
  - 对物理结构进行评价，评价的重点是时间和空间效率。满足原设计要求，则可进入到物理实施阶段，否则，就需要重新设计或修改物理结构。

7.5 物理结构设计

7.5.1 数据库的物理设计的内容和方法

7.5.2 关系模式存取方法选择

7.5.3 确定数据库的存储结构

7.5.4 评价物理结构

7.5.1 数据库的物理设计的内容和方法

- 设计优化的物理数据库结构，使得在数据库上运行的各种事务响应时间小、存储空间利用率高、事务吞吐率大，需要注意两点：
  - 首先要对运行的事务进行详细分析，获得选择物理数据库设计所需要的参数；
  - 其次要充分了解所用的**RDBMS**的内部特征，特别是系统提供的存取方法和存储结构。

7.5.1 数据库的物理设计的内容和方法

- 对于数据库查询事务，需要了解：
  - 查询的关系
  - 查询条件所涉及的属性
  - 连接条件所涉及的属性
  - 查询的投影属性
- 对于数据更新事务，需要了解：
  - 被更新的关系
  - 每个关系上的更新操作条件所涉及的属性
  - 修改操作要改变的属性值

7.5.1 数据库的物理设计的内容和方法

- 另外，需要了解每个事务在各关系上运行的频率和性能要求
- 以上信息是确定关系的存取方法的依据
- 通常对于关系数据库物理设计的内容主要包括：
  - 为关系模式选择存取方法
  - 设计关系、索引等数据库文件的物理存储结构

7.5.2 关系模式存取方法选择

- 存取方法是快速存取数据库中数据的技术。物理设计的任务之一就是要确定选择哪些存取方法，即建立哪些存取路径。
- 常用的存取方法有：
  - 索引方法
  - 聚簇方法

7.5.2 关系模式存取方法选择

1. B+树索引存取方法的选择

- 根据应用要求确定对关系的哪些属性列建立索引、哪些属性列建立组合索引、哪些索引要设计为唯一索引等。一般：

（1）如果一个（或一组）属性经常在查询条件中出现，则考虑在这个（或这组）属性上建立索引（或组合索引）；

7.5.2 关系模式存取方法选择

（2）如果一个属性经常作为最大值或最小值等聚集函数的参数，则考虑在这个属性上建立索引；

（3）如果一个（或一组）属性经常在连接操作的连接条件中出现，则考虑在这个（或这组）属性上建立索引；

- 索引可以加快查询的速度，但也不是索引越多越好，因为维护索引也要付出代价。

7.5.2 关系模式存取方法选择

2. hash索引存取方法的选择

- 如果一个关系的属性主要出现在等值连接条件中或主要出现在等值比较选择条件中，而且满足下列两个条件之一，则此关系可以选择hash存取方法。
  - （1）一个关系的大小预知，而且不变。
  - （2）关系的大小动态改变，但数据库管理系统提供了动态hash存取方法。

7.5.2 关系模式存取方法选择

3. 聚簇存取方法的选择

- 为了提高某个属性（或属性组）的查询速度，把这个或这些属性（称为聚簇码）上具有相同值的元组集中存放在连续的物理块中称为聚簇。
- 聚簇的查询效率高于索引。
- 聚簇功能适用于单个关系，也适用于经常进行连接操作的多个关系，提高连接操作的效率。

7.5.2 关系模式存取方法选择

- 一个数据库可以建立多个聚簇，一个关系只能加入一个聚簇。
- 选择聚簇存取方法，即确定需要建立多少个聚簇，每个聚簇中包含哪些关系。

7.5.2 关系模式存取方法选择

- 先设计候选聚簇

1. 对经常在一起进行连接操作的关系可以建立聚簇；
2. 如果一个关系的一组属性经常出现在相等的比较条件中，则该单个关系可建立聚簇。
3. 如果一个关系的一个（或一组）属性上的值重复率很高，则此单个关系可建立聚簇。

7.5.2 关系模式存取方法选择

- 然后检查候选聚簇中的关系，取消其中不必要的关系：

1. 从聚簇中删除经常进行全表扫描的关系；
2. 从聚簇中删除更新操作远多于连接操作的关系；
3. 对于出现在不同聚簇中的关系，应选择一个较优的聚簇方案，使在这个聚簇上运行的各种事务的总代价最小。

7.5.2 关系模式存取方法选择

- 使用聚簇的优缺点
  - 聚簇可以提高某些应用的性能。当通过聚簇码进行访问或连接是该关系的主要应用时，可以使用聚簇。如在SQL语句中包含ORDER BY,GROUP BY, UNION, DISTINCT 等子句或短语时，使用聚簇特别有利。
  - 对已有的关系建立聚簇，将导致关系中元组移动其物理存储位置，并使此关系上原有的索引无效。聚簇码值要相对稳定，以减少修改聚簇码值所引起的维护开销。

7.5.3 确定数据库的存储结构

- 确定数据库物理结构主要指确定数据的存放位置和存贮结构，包括确定关系、索引、聚簇、日志、备份等的存储安排和存储结构；确定系统配置等。
- 确定数据的存放位置和存储结构要综合考虑存取时间、存储空间利用率和维护代价三方面的因素。

7.5.3 确定数据库的存储结构

1. 确定数据的存放位置

为了提高系统性能，应该根据应用情况将数据易变部分与稳定部分、经常存取部分和存取频率较低部分分开存放。

2. 确定系统配置

DBMS产品一般都提供了一些系统配置变量、存储分配参数，供设计人员和DBA对数据库进行物理优化。在缺省值不适合应用环境时，可以重新配置这些参数，以改善系统的性能。

7.5.4 评价物理结构

- 数据库物理设计过程中需要对时间效率、空间效率、维护代价和各种用户要求进行权衡，可以产生多种方案。对这些方案要进行细致的评价，从中选择一个较优的方案作为数据库的物理结构。
- 在物理结构不符合用户需求时，要修改设计。

7.6 数据库的实施和维护

- 完成数据库的物理设计之后，设计人员就要用**RDBMS**提供的数据定义语言和其他实用程序将数据库逻辑设计和物理设计的结果严格描述出来，成为**DBMS**可以接受的源代码，再经过调试产生目标模式。然后就可以组织数据入库了，这就是数据库实施阶段。

7.6 数据库的实施和维护

7.6.1 数据的载入和应用程序的调试

7.6.2 数据库的试运行

7.6.3 数据库的运行和维护

7.6.1 数据的载入和应用程序的调试

- 数据库实施阶段包括两项重要的工作：
  - 数据的载入
  - 应用程序的编码和调试

7.6.2 数据库的试运行

- 在原有系统的数据有一小部分已输入数据库后，就可以开始对数据库系统进行联合调试，这又称为数据库的试运行。
- 这一阶段要实际运行数据库应用程序，执行对数据库的各种操作，测试应用程序的功能是否满足设计要求。如果不满足，对应用程序部分则要修改、调整，直到达到设计要求为止。

7.6.2 数据库的试运行

- 在数据库试运行时，还要测试系统的性能指标，分析其是否达到设计目标。如果测试的结果与设计目标不符，则要返回物理设计阶段，重新调整物理结构，修改系统参数，某些情况下甚至要返回逻辑设计阶段，修改逻辑结构。

7.6.2 数据库的试运行

- 注意两点：
  - 试运行阶段，先输入小批量数据进行调试。待试运行基本合格后，再大批量输入数据，逐步增加数据量，逐步完成运行评价。
  - 在试运行阶段，应首先调试运行**DBMS**的恢复功能，做好数据库的转储和恢复工作。一旦故障发生，能使数据库尽快恢复，尽量减少对数据库的破坏。

7.6.3 数据库的运行和维护

- 数据库试运行合格后，数据库开发工作就基本完成，即可投入正式运行了。
- 在数据库运行阶段，对数据库经常性的维护工作主要由**DBA**完成的，包括：
  - 数据库的转储和恢复
  - 数据库的安全性、完整性控制
  - 数据库性能的监督、分析和改造
  - 数据库的重组织与重构造
    - 重组织不修改原设计的逻辑和物理结构，而重构造将部分修改数据库的模式与内模式。
    - 重构无效，数据库应用系统的生命周期结束。

7.7 小结

- 本章详细介绍了数据库设计各个阶段的目标、方法、应注意事项。
- 重点是概念结构的设计和逻辑结构的设计，这也是数据库设计过程中最重要的两个环节。