数据库设计

概述

数据库设计6阶段

Six stages of database design：Requirements analysis, conceptual design, logical design, physical design, database implementation, running and maintenance。（需求分析，概念设计，逻辑设计，物理设计，数据库实施，运行和维护）

需求分析中的数据字典

> Software requirements specification (SRS) : 软件的需求规格说明书，需求分析阶段最主要的产物。其中，Data dictionary 【包括哪些东西？（见下）】(DD,数据字典)，是SRS中重要的内容之一。
> Data dictionary usually includes data item, data structure, data stream, data storage and processing procedure。（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。）

软件需求规格说明书（Software Requirements Specification，简称SRS）是需求分析阶段的一个重要产出。它详细记录了软件的功能和非功能需求，为软件的设计、开发和测试提供了基准。在这个文档中，通常会包含一个数据字典（Data Dictionary，简称DD）部分，它是说明软件中用到的数据的详尽列表。

数据字典包括以下几个主要部分：

1. **数据项（Data Item）** 数据项是数据字典中的基本单元，它详细说明了软件中每个数据元素的属性。这些属性可能包括数据元素的名称、类型（如整数、字符串）、大小（如长度）、格式、可能的值和它们的含义，以及数据元素的使用说明。
2. **数据结构（Data Structure）** 数据结构描述的是数据项之间的组织关系。在数据字典中，你可以发现有哪些记录、文件、表等数据结构以及它们如何构建（如哪些数据项组成了一个特定的记录或表结构）。
3. **数据流（Data Stream）** 数据流是指在系统组件之间传递的数据集。数据字典中会描述这些数据流的内容、来源、目标、和数据流发生的触发事件。
4. **数据存储（Data Storage）** 数据存储部分说明了数据如何被保存和存取。这包括对数据库表、文件以及可能的访问路径等详细信息的描述。
5. **处理过程（Processing Procedure）** 处理过程说明数据是如何被处理的，例如，数据是如何被插入、查询、更新和删除的。这个部分通常会包含处理数据所需要各种算法和逻辑流的描述。

概念结构设计 -- E-R模型

实体间联系

- One-to-one : each entity in the relationship will have exactly one related entity（1:1，一对一）

- One -to-many : an entity on one side of the relationship can have many related entities, but an entity on the other side will have a maximum of one related entity（1：n，一对多）

- Many-to-many : entities on both sides of the relationship can have many related entities on the other side（n：m，多对多）

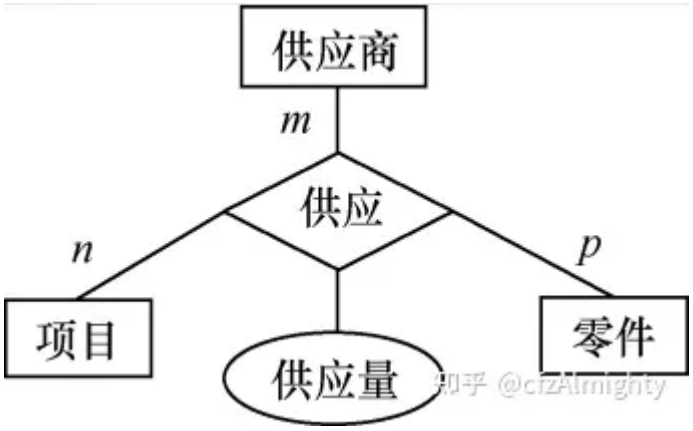
E-R图

E-R 图提供了表示实体型、属性和联系的方法。

（1）实体型：用矩形表示，矩形框内写明实体名。

（2）属性：用椭圆形表示，并用无向边将其与相应的实体型连接起来。

（3）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 或 m:n）。



一个例子❗

某个工厂物资管理的概念模型。物资管理涉及的实体有：

- 仓库：属性有仓库号、面积、电话号码
- 零件：属性有零件号、名称、规格、单价、描述
- 供应商：属性有供应商号、姓名、地址、电话号码、账号
- 项目：属性有项目号、预算、开工日期
- 职工：属性有职工号、姓名、年龄、职称

这些实体之间的联系如下：

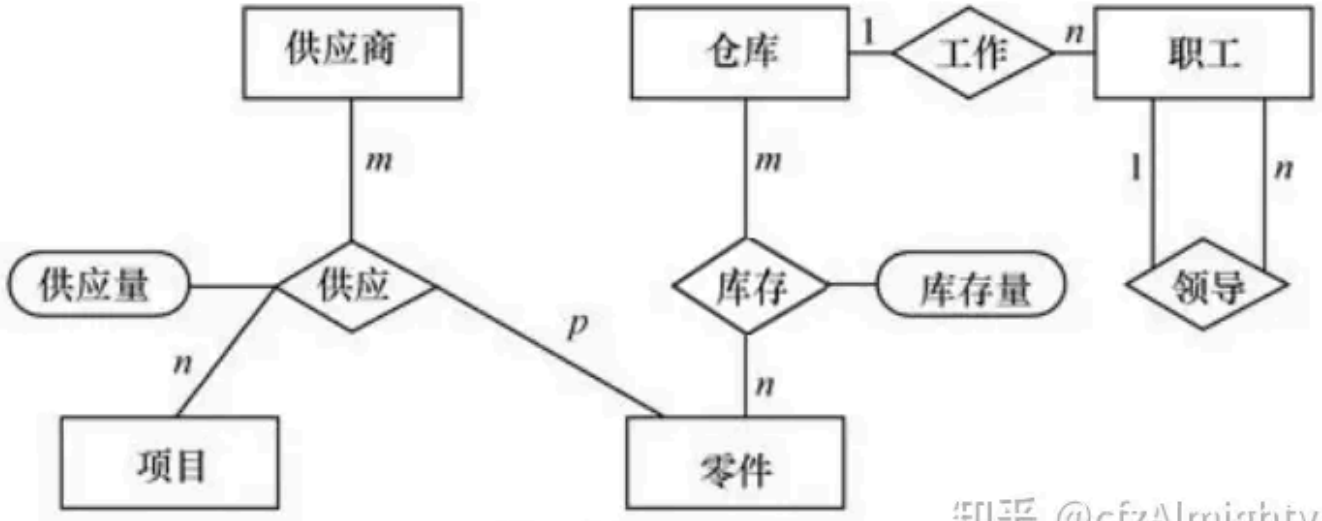
（1）一个仓库可以存放多种零件，一种零件可以存放在多个仓库中，因此仓库和零件具有多对多的联系。用库存量来表示某种零件在某个仓库中的数量。

（2）一个仓库有多个职工当仓库保管员，一个职工只能在一个仓库工作，因此仓库和职工之间是一对多的联系。

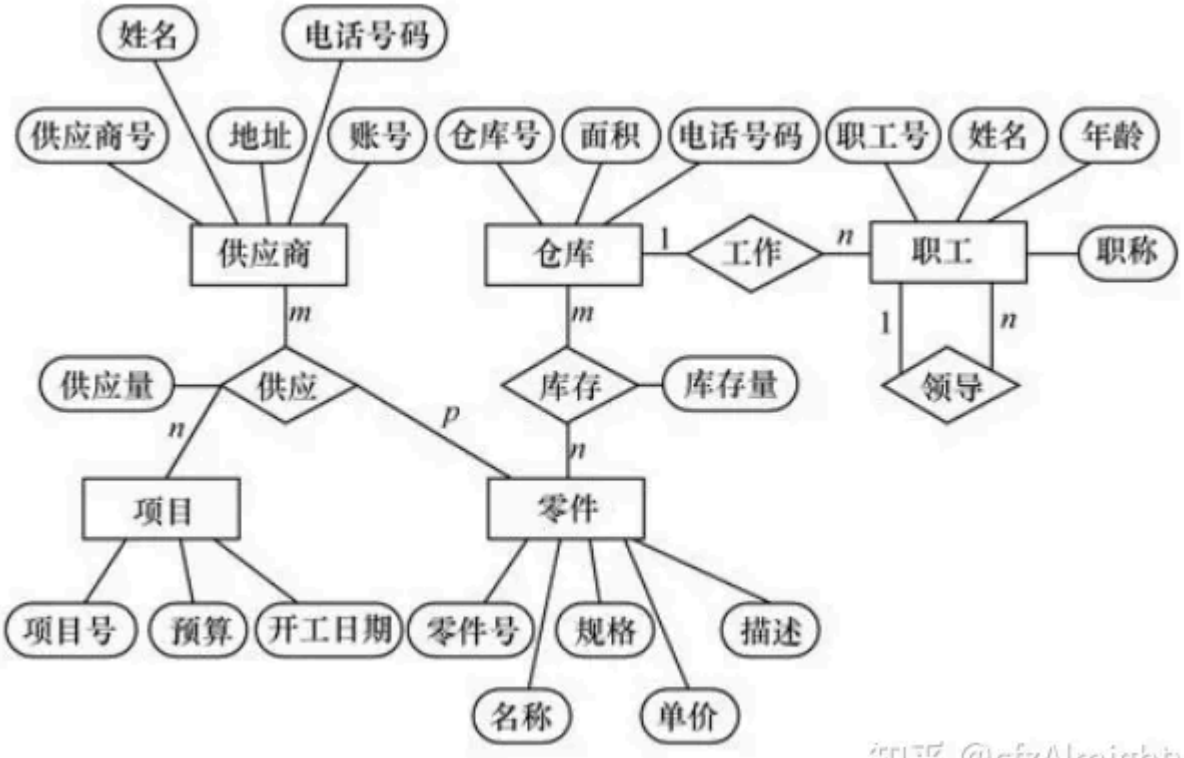
（3）职工之间具有领导与被领导关系，即仓库主任领导若干保管员，因此职工实体型中具有一对多的联系。

（4）供应商、项目和零件三者之间具有多对多的联系，即一个供应商可以供给若干项目多种零件，每个项目可以使用不同供应商供应的零件，每种零件可由不同供应商供给。

(a) 实体及其属性图



(b) 实体及其联系图



(c) 完整的实体-联系图

冲突

两个准则

（1）作为属性，不能再具有需要描述的性质，即属性必须是不可分的数据项，不能包含其他属性。

（2）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。

冲突和解决

> There are three main types of conflicts between subsystem ER diagrams: attribute conflict, naming conflict and structure conflict. (子系统ER图之间的冲突主要由三类：属性冲突，命名冲突，结构冲突。)

各子系统的 E-R 图之间的冲突主要有三类：属性冲突、命名冲突和结构冲突。

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。例如零件号，有的部门把它定义为整数，有的部门把它定义为字符型。例如年龄，某些部门以出生日期形式表示职工的年龄，而另一些部门用整数表示职工的年龄。
- 属性取值单位冲突。例如零件的重量有的以公斤为单位，有的以斤为单位，有的以克为单位。

属性冲突理论上好解决，但实际上需要各部门讨论协商，解决起来并非易事。

② 命名冲突

- 同名异义，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（一义多名），即同一意义的对象在不同的局部应用中具有不同的名字。

如对科研项目，财务科称为项目，科研处称为课题，生产管理处称为工程。

命名冲突可能发生在实体、联系一级上，也可能发生在属性一级上，通过讨论、协商等行政手段加以解决。

③ 结构冲突

主要包含以下三类冲突：

- 同一对象在不同应用中具有不同的抽象。

例如，职工在某一局部应用中被当作实体，而在另一局部应用中则被当作属性。

解决方法：把属性变换为实体或把实体变换为属性，使同一对象具有相同的抽象。

- 同一实体在不同子系统的 E-R 图中所包含的属性个数和属性排列次序不完全相同。

解决方法：使该实体的属性取各子系统的 E-R 图中属性的并集，再适当调整属性的次序。

- 实体间的联系在不同的 E-R 图中为不同的类型。

实体 E1 与 E2 在一个 E-R 图中是多对多联系，在另一个 E-R 图中是一对多联系。

解决方法：根据应用的语义对实体联系的类型进行综合或调整。

> 结构冲突:
>
> 例子1：实体与属性的抽象差异
> 在图书馆子系统中，可能将“作者”作为一个书籍（Book）实体的属性；而在出版社子系统中，“作者”可能是一个独立的实体，与书籍通过“编写”关系相连接。实体和属性在不同系统中的抽象层次不同，构成了结构冲突。
> 例子2：实体的属性差异
> 假设在销售子系统中，客户（Customer）实体只有“姓名”和“电话”两个属性；而在市场营销的子系统中，客户实体不但包含“姓名”和“电话”，还包含了“地址”、“购买历史”等多个属性。当合并这些子系统时，就要处理实体属性的差异性冲突。
> 例子3：实体间联系的类型差异
> 考虑一个学校的教务系统和教师个人管理系统。在教务系统中，课程（Course）和教师（Teacher）之间是多对多关系，一个教师可以教授多门课程，一门课程也可以由多个教师共同授课。而在教师个人管理系统中，可能将教师和课程的关系设计为一对多关系，即假定一门课程只由一个教师授课。这就造成了实体间联系的类型冲突。

逻辑结构设计

E-R图向关系模型的转换❗

ER图转换关系模型 - vlj - 博客园 (cnblogs.com)

一个实体型转换为一个关系模式，关系的属性就是实体的属性，关系的码就是实体的码

实体型间的联系有以下不同情况：

（1）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

① 转换为一个独立的关系模式

- 关系的属性：与该联系相连的各实体的码以及联系本身的属性
- 关系的候选码：每个实体的码

② 与某一端实体对应的关系模式合并

- 合并后关系的属性：加入对应关系的码和联系本身的属性
- 合并后关系的码：不变

> 例子：把一个人(Person)和他的驾照(Driving License)之间的关系转化。
> 转化为独立的关系模式：创建一个关系，比如叫做 Person_Driving_License，属性是 Person_ID、Driving_License_ID 和 其它属性（例如发证日期）。
> 合并：我们可以把 Driving License 实体的属性（比如驾照号、发证日期）添加到 Person 实体中，这样驾照实体就不再需要了。

（2）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

① 转换为一个独立的关系模式

- 关系的属性：与该联系相连的各实体的码以及联系本身的属性
- 关系的码：n 端实体的码

② 与 n 端对应的关系模式合并

- 合并后关系的属性：在 n 端关系中加入 1 端关系的码和联系本身的属性
- 合并后关系的码：不变

可以减少系统中的关系个数，一般情况下更倾向于采用这种方法

> 例子：一个部门(Department)有多个员工(Employee)。
> 转化为独立关系模式：创建一个关系，比如 Department_Employee，属性包括 Department_ID、Employee_ID 和其它属性（例如雇佣日期）。
> 合并：我们可以合并 n 端对应的实体至一端，也就是将部门编号（Department_ID）增加到员工实体中作为一个新的属性。

（3）一个 **m:n 联系**转换为一个关系模式

- 关系的属性：与该联系相连的各实体的码以及联系本身的属性
- 关系的码：各实体码的组合

例子：一个学生(Student)可以选修多门课程(Course)，一门课程也可以被多个学生选修。
转化为新的关系模式：可以创建一个学生-课程关系(Student_Course)，属性是 Student_ID、Course_ID 和其它可能的属性，如选课日期、成绩等。

（4）三个或三个以上实体间的一个多元联系转换为一个关系模式

- 关系的属性：与该多元联系相连的各实体的码以及联系本身的属性
- 关系的码：各实体码的组合

例子：一个学生(Student)可以在特定的时间(Time)、特定的地点(Location)参加一次考试(Exam)。
转化为新的关系模式：可以创建一个关系，比如叫做Student_Exam_Location_Time，属性是Student_ID、Exam_ID、Location_ID、Time_ID及其它可能的属性例如考试成绩。

（5）具有相同码的关系模式可合并

- 目的：减少系统中的关系个数
- 合并方法：将其中一个关系模式的全部属性加入到另一个关系模式中，然后去掉其中的同义属性（可能同名也可能不同名），适当调整属性的次序

例子：在学院(College)和部门(Department)两个实体间，我们发现它们都有一个相同的属性：学院编号(College_ID) [有着相同的候选键]
合并：我们可以将 Department 的所有属性（例如 Department_ID, Department_Name 等）添加到 College 中，然后删除 Department。这样我们就把两个实体合并成了一个新的实体。

其他

```
Relation
•       A relation is a named, two - dimensional table of data.
•       A table consists of rows (records) and columns (attribute or field).
•       Requirements for a table to qualify as a relation:
o       It must have a unique name.
o       Every attribute value must be atomic(not multivalve, not composite)
o       Every row must be unique (can't have two rows with exactly the same values
for all their fields).
o       Attributes (columns) in tables must have unique names.
o       The order of the columns must be irrelevant.
o       The order of the rows must be irrelevant.

Correspondence with ER model
•       Relations (tables) correspond with entity types and with many - to - many
relationship types.
```

```
•       Rows correspond with entity instances and with many - to - many
relationship instances.
•       Columns correspond with attributes.
NOTE: the word relation (in relational database) is NOT the same as the word
relationship (in ER model).
-------------------------------------------------------------------------------
-----------------------------------------
Key fields
Primary keys are unique identifiers of the relation.
Foreign keys are identifiers that enable a dependent relation (on the many side of
a relationship) to refer to its parent relation(on the one side of the
relationship)

Keys can be simple (a single field) or composite (more than one field).
Keys usually are used as indexers to speed up the response to user queries.
All primary key fields must have data.
```

关系型数据库与ER模型的对应关系

在关系型数据库中的"关系"与ER模型中的元素有以下对应：

- 关系（**Relations/Tables**）对应**ER模型中的实体类型和多对多关系类型。**
  - 例如，ER模型中的"学生"实体类型在数据库中对应"学生"表，ER模型中的"选课"多对多关系在数据库中对应"选课"表。
- 行（**Rows**）对应**ER模型中的实体实例和多对多关系的实例。**
  - 表中的每一行记录都代表一个实体的实例（如一个具体的学生）或者多对多关系的一个实例（如某个学生与某门课程的选课关系）。
- 列（**Columns**）对应**ER模型中的属性。**
  - 表的每一列对应着ER模型中定义的某个实体或关系的属性（如学生的名字或课程的编号）。

注意：在这里需要强调的是，关系型数据库的"关系"（表）与ER模型中的"关系"（Relationship）是不同的。前者指的是存储数据的结构，而后者指的是实体之间的联系方式。