PowerBuilder 8.0 从基础到应用

崔杜武 姚全珠 范艳华 黑新宏

编著

人民邮电出版社

第二篇　应用篇

第 10 章　开发数据库应用软件综述

10.1　软件开发方法概述

在 20 世纪 60 年代中期爆发了众所周知的软件危机。为了克服这一危机，在 1968、1969 年连续召开的两次著名的 NATO 会议上提出了软件工程这一术语，并在以后不断发展、完善了这一技术。许多软件公司和机构都在研究软件开发方法这个概念性的东西，而且也提出了很多实际的开发方法，比如：生命周期法、原型化方法、面向对象方法等。就软件开发模型而言，目前有瀑布模型，螺旋模型，渐增式模型和喷泉模型。本节首先对比较常用的几种软件开发方法予以综述，以使读者对此有初步的了解。

10.1.1　结构化方法

1978 年，E．Yourdon 和 L．L．Constantine 提出了软件开发的结构化方法，即 SASD 方法，也称为面向功能的软件开发方法或面向数据流的软件开发方法。1979 年 TomDeMarco 对此方法作了进一步的完善。

Yourdon 方法是 80 年代使用最广泛的软件开发方法。它首先用结构化分析（SA）对软件进行需求分析，然后用结构化设计（SD）方法进行总体设计，最后是结构化编程（SP）。

结构化分析就是面向数据流自顶向下逐步求精进行分析。其步骤为：

(1) 按照可行性研究后画好的数据流图，根据输出要求沿数据流图回溯，看输出及运算所得到的信息是否能满足输出要求。

(2) 请用户复查数据流图，看是否能满足用户要求。

(3) 细化数据流图，把比较复杂的处理过程分解细化。

(4) 编写文档，并进行复查和复审。

结构化设计分为总体设计和详细设计。总体设计要确定系统的具体实现方案和软件的具体结构。其步骤为：

(1) 设想供选择的方案。
(2) 选取合理的方案。
(3) 推荐最佳方案。
(4) 功能分解，以确定系统由那些模块组成，以及这些模块之间的关系。
(5) 设计软件结构。根据数据流图的类型（处理型、事务型）采用相应的映射方法映射成相应的模块层次结构，并对其优化。
(6) 进行数据库设计。根据数据字典进行数据库的逻辑设计。

详细设计则是借助程序流程图，N-S 图或 PAD 图之类的详细设计工具来描述实现具体功能的算法。

结构化实现则是采用诸如 PASCAL、C、FORTRAN 等结构化语言对详细设计所得到的算法进行编码。

这一方法开发步骤明确，SA、SD、SP 相辅相成，一气呵成。使软件开发的成功率大大提高，从而深受软件开发人员的青睐。

10.1.2　面向数据结构的软件开发方法

1. Jackson 方法

1975 年，M．A．Jackson 提出了一类至今仍广泛使用的软件开发方法。这一方法从目标系统的输入、输出数据结构入手，导出程序框架结构，再补充其他细节，就可得到完整的程序结构图。这一方法对输入、输出数据结构明确的中小型系统特别有效，例如商业应用中的文件表格处理。该方法也可与其他方法结合，用于模块的详细设计。Jackson 方法有时也称为面向数据结构的软件设计方法。

2. Warnier 方法

1974 年，J．D．Warnier 提出的软件开发方法与 Jackson 方法类似。差别有三点：一是它们使用的图形工具不同，分别使用 Warnier 图和 Jackson 图；另一个差别是使用的伪码不同；最主要的差别是在构造程序框架时，Warnier 方法仅考虑输入数据结构，而 Jackson 方法不仅考虑输入数据结构，而且还考虑输出数据结构。

10.1.3　问题分析法

PAM（ProblemAnalysisMethod）是 80 年代末由日立公司提出的一种软件开发方法。PAM 方法希望能兼顾 Yourdon 方法、Jackson 方法和自底向上的软件开发方法的优点，而避免它们的缺陷。它的基本思想是：考虑到输入、输出数据结构，并指导系统的分解；在系统分析指导下逐步综合。这一方法的具体步骤是：

(1) 从输入、输出数据结构导出基本处理框；分析这些处理框之间的先后关系。

(2) 按先后关系逐步综合处理框，直到画出整个系统的 PAD 图。

从上述步骤中可以看出，这一方法本质上是综合自底向上的方法，但在逐步综合之前已进行了有目的的分解，其目的就是充分考虑系统的输入、输出数据结构。

PAM 方法的另一个优点是使用 PAD 图。这是一种二维树形结构图，是到目前为止最好的详细设计的表示方法之一，远远优于 NS 图和 PDL 语言。

这一方法在日本较为流行，软件开发的成功率也很高。但由于在输入、输出数据结构与整个系统之间同样存在着鸿沟，使这一方法仍只适用于中小型问题。

10.1.4 原型化方法

产生原型化方法的原因很多，主要是随着系统开发经验的增多，软件开发人员也发现并非所有的需求都能够预先定义，而反复修改是不可避免的。当然能够采用原型化方法还是因为开发工具的快速发展，比如用 VB，PowerBuilder，Delphi 等工具，可以迅速地开发出一个可以让用户看得见、摸得着的系统框架。这样，对计算机不是很熟悉的用户就可以根据这个样板提出自己的需求。历史上曾经形成了实现原型法的两种途径。

1．抛弃原型法

其目的是要评价目标系统的某些特性，以便更准确地定义需求，使用之后就把这种原型抛弃掉。

2．演化原型法

演化原型法是一个多次迭代的过程，每次迭代都由以下几个阶段组成：

(1) 确定用户需求。

(2) 开发原始模型。

(3) 征求用户对初始原型的改进意见。

(4) 修改原型。

原型化开发比较适合于用户需求不清楚、业务不确定、需求经常变化的情况。当系统规模不是很大也不太复杂时采用该方法是比较好的。

10.1.5 面向对象的软件开发方法

面向对象技术的产生可称得上是软件技术的一次革命，在软件开

发史上具有里程碑的意义。随着 OOP（面向对象编程）向 OOD（面向对象设计）和 OOA（面向对象分析）的发展，最终形成面向对象的软件开发方法 OMT（ObjectModelingTechnique）。这是一种自底向上和自顶向下相结合的方法，而且它以对象建模为基础，不仅考虑了输入、输出数据结构，实际上也包含了所有对象的数据结构。所以 OMT 彻底实现了 PAM 没有完全实现的目标。不仅如此，面向对象技术在需求分析、可维护性和可靠性这三个软件开发的关键环节和质量指标上有了实质性的突破，解决了在这些方面长期存在的严重问题。

1．自底向上的归纳

OMT 的第一步是从问题的陈述入手，构造系统模型。从真实系统导出类的体系，即对象模型包括类的属性，与子类、父类的继承关系，以及类之间的关联。类是具有相似属性和行为的一组具体实例（客观对象）的抽象，父类是若干子类的归纳。因此这是一种自底向上的归纳过程。在自底向上的归纳过程中，为使子类能更合理地继承父类的属性和行为，可能需要自顶向下的修改，从而使整个类体系更加合理。由于这种类体系的构造是从具体到抽象，再从抽象到具体，符合人类的思维规律，因此能更快、更方便地完成任务。这与自顶向下的 Yourdon 方法构成鲜明的对照。在 Yourdon 方法中，构造系统模型是最困难的一步，因为自顶向下的“顶”是一个空中楼阁，缺乏坚实的基础，而且功能分解有相当大的任意性，因此需要开发人员有丰富的软件开发经验。而在 OTM 中这一工作可由一般开发人员较快地完成。在对象模型建立后，很容易在这一基础上再导出动态模型和功能模型。这三个模型一起构成要求解的系统模型。

2．自顶向下的分解

系统模型建立后的工作就是分解。与 Yourdon 方法按功能分解不同，在 OMT 中通常按服务（Service）来分解。服务是具有共同目标的相关功能的集合，如 I／O 处理、图形处理等。这一步的分解通常很明确，而这些子系统的进一步分解因有较具体的系统模型为依据，也相对容易。所以 OMT 也具有自顶向下方法的优点，即能有效地控制模块的复杂性，同时避免了 Yourdon 方法中功能分解的困难和不确定性。

3．OMT 的基础是对象模型

每个对象类由数据结构（属性）和操作（行为）组成，有关的所有数据结构（包括输入、输出数据结构）都是软件开发的依据。因此 Jackson 方法和 PAM 中输入、输出数据结构与整个系统之间的鸿沟在 OMT 中不再存在。OMT 不仅具有 Jackson 方法和 PAM 的优点，而且可以应用于大型系统。更重要的是，在 Jackson 方法和 PAM 方法中，当它们的出发点——输入、输出数据结构（即系统的边界）发生变化时，整个软件必须推倒重来。但在 OMT 中系统边界的改变只是增加或减少一些对象而已，整个系统改动极小。

4．需求分析彻底

需求分析不彻底常常是软件失败的主要原因之一。传统的软件开发方法不允许在开发过程中用户的需求发生变化，从而导致种种问题。正是由于这一原因，人们提出了原型化方法，推出探索原型、实验原型和进化原型，积极鼓励用户改进需求。在每次改进需求后又形成新的进化原型供用户试用，直到用户基本满意，从而大大提高了软件的成功率。但是它要求软件开发人员能迅速生成这些原型，这就要求有自动生成代码的工具支持。

OMT 彻底解决了这一问题。因为需求分析过程已与系统模型的形成过程一致，开发人员与用户的讨论是从用户熟悉的具体实例（实体）开始的。开发人员必须搞清现实系统才能导出系统模型，这就使用户与开发人员之间有了共同的语言，避免了传统需求分析中可能产生的种种问题。

5．可维护性大大改善

在 OMT 之前的软件开发方法都是基于功能分解的。尽管软件工程技术在可维护方面作出了极大的努力，使软件的可维护性有较大的改进，但从本质上讲，基于功能分解的软件是不易维护的。因为功能一旦有变化都会使开发的软件系统产生较大的变化，甚至推倒重来。更严重的是，在这种软件系统中，修改是困难的。由于种种原因，即使是微小的修改也可能引入新的错误。所以传统开发方法很可能会引起软件成本增长失控、软件质量得不到保证等一系列严重问题。正是 OMT 才使软件的可维护性有了质的改善。

OMT 的基础是目标系统的对象模型，而不是功能的分解。功能是对象的使用，它依赖于应用的细节，并在开发过程中不断变化。由于对象是客观存在的，因此当需求变化时对象的性质要比对象的使用更为稳定，从而使建立在对象结构上的软件系统也更为稳定。更重要的是 OMT 彻底解决了软件的可维护性。在面向对象语言中，子类不仅可以继承父类的属性和行为，而且也可以重载父类的某个行为（虚函数）。利用这一特点，就可以方便地进行功能修改：引入某类的一个子类，对要修改的一些行为（即虚函数或虚方法）进行重载，也就是对它们重新定义。由于不再在原来的程序模块中引入修改，所以彻底解决了软件的可修改性的问题，从而也彻底实现了软件的可维护性。面向对象技术还提高了软件的可靠性和健壮性。

10.1.6 可视化开发方法

可视化开发是 90 年代软件界最大的两个热点之一。其实可视化开发并不能单独地作为一种开发方法，更加贴切的说可以认为它是一种辅助工具。比如用过 SYBASE 的 S-Design 的人都知道，用这个工具可以进行显式图形化的数据库模式的建立，并可以导入到不同的数据库中

去。随着图形用户界面的兴起，用户界面在软件系统中所占的比例也越来越大，有的甚至高达 60～70%。产生这一问题的原因是图形界面元素的生成很不方便。为此 Windows 提供了应用程序设计接口 API（ApplicationProgrammingInterface），它包含了 600 多个函数，极大地方便了图形用户界面的开发。但是在这批函数中，大量的函数参数使用了数量更多的有关常量，使基于 WindowsAPI 的开发变得相当困难。为此 Borland C++推出了 ObjectWindows 编程。它将 API 的各部分用对象类进行封装，提供了大量预定义的类，并为这些类定义了许多成员函数。利用子类对父类的继承性，以及实例对类函数的引用，使应用程序的开发省却大量类的定义，以及成员函数的定义，或只需作少量修改以定义子类。ObjectWindows 还提供了许多标准的缺省处理，大大减少了应用程序开发的工作量。但要掌握它们，对非专业人员来说仍是一个沉重的负担。为此人们利用 WindowsAPI 或 Borland C++ 的 ObjectWindows 开发了一批可视开发工具。

可视化开发就是在可视开发工具提供的图形用户界面上，通过操作界面元素，诸如菜单、按钮、对话框、编辑框、单选框、复选框、列表框和滚动条等，由可视开发工具自动生成应用软件。这类应用软件的工作方式是事件驱动。对每一事件，由系统产生相应的消息，再传递给相应的消息响应函数。这些消息响应函数是由可视开发工具在生成软件时自动装入的。

可视开发工具提供了两大类服务。一类是生成图形用户界面及相关的消息响应函数。通常的方法是先生成基本窗口，并在它的外面以图标形式列出所有其他的界面元素，让开发人员挑选后放入窗口指定位置。在逐一安排界面元素的同时，还可以用鼠标拖动，以使窗口的布局更趋合理。另一类服务是为各种具体的子应用的各个常规执行步骤提供规范窗口，它包括对话框、菜单、列表框、组合框、按钮和编辑框等，以供用户挑选。开发工具还为所有的选择（事件）提供消息响应函数。

由于要生成与各种应用相关的消息响应函数，因此，可视化开发只能用于相当成熟的应用领域，如目前流行的可视化开发工具基本上用于关系数据库的开发。对一般的应用，目前的可视化开发工具只能提供用户界面的可视化开发。至于消息响应函数（或称脚本），则仍需用通常的高级语言（3GL）编写。只有在数据库领域才提供 4GL，使消息响应函数的开发大大简化。可视化开发使我们把注意力集中在业务逻辑和业务流程上，用户界面可以用可视化工具方便地构成。通过操作界面元素，诸如菜单、按钮、对话框、编辑框、单选框、复选框、列表框和滚动条等，由可视开发工具自动生成应用软件。

10.1.7　基于组件的软件开发

1．组件的概念

有关组件，目前还没有严格的定义。但是我们可以大致描述如下：组件是一种能够提供某种服务的自包含的软件模块，它封装了一定的数据（属性）和方法，并提供特定的接口，开发人员利用这一特定的接口来使用组件，并使其与其他组件交互通讯，以此来构造应用程序。组件和对象的区别在于：对象封装了一组相关的函数，而组件则封装了一组相关的对象。

2．组件的特点

从广义上来说，构件有如下几个基本属性：

(1) 组件是可独立配置的单元，组件的概念是独立于编程语言的，这也就是说，用不同语言编写的组件应能在一起协同工作，或者说用一种语言编写的组件能在用另一种语言编写的应用程序中很好地工作。因此构件必须自包容。

(2) 构件强调与环境和其他构件的分离，因此构件的实现是严格封装的，外界没机会或没必要知道构件内部的实现细节，只提供接口供开发人员使用，这使得开发人员不必了解组件的内部细节，就能利用组件方便地构筑应用程序。这种特点还使得组件开发人员可以对组件单独进行升级，改进原来的功能，却不影响整个应用系统的运行，只要保证组件对外界的接口保持不变即可。

(3) 可以在适当的环境中被复合使用，因此构件需要提供清楚的接口规范，与环境交互。组件还能跨网络而运行。也就是说，组件能被部署到联网的各个计算机上，它们之间可以通过某种通讯机制相交互。从而构筑基于网络环境的分布式应用程序，实现分布式计算。

(4) 构件不应当是持续的，即构件没有个体特有的属性。这可理解为构件不应当与自身副本区别。

从以上四个属性可以看出，构件沿袭了对象的封装特性，但同时并不局限在一个对象，其内部可以封装一个或多个类、原型对象，甚至过程，结构是灵活的。构件突出了自包容和被包容的特性，这是在软件工厂的软件开发生产线上作为零件的必要特征。

通过以上分析，可以看到组件可以被重复地使用，而且不受到操作平台、开发环境，甚至是物理计算机的限制。

3．用组件开发应用程序的意义

组件的出现改变了传统的软件开发方法，我们可以把应用程序的需求分解成明确定义的服务，然后进一步创建具体的物理组件来实现它们。事实上，目前已有许多专门的组件开发商开发出了大量的能完成特定任务的组件。例如 Microsoft 公司的 COM/DCOM，由 OMG 集团定义的 CORBA 等。在 COM 库中就可提供如下 4 类可执行代码：

(1) 提供 COM 的应用程序接口函数（API），例如在客户方提

供生成基本对象的函数，在服务器方提供对外公布其接口的函数等。

(2) 从类标识符寻找生成该类对象的服务器，在服务器的系统注册表中存有它所支持的类标识符及生成该类对象所需的有关数据，可支持此类服务。

(3) 提供远程过程调用服务。

(4) 提示进程内的内存分配管理。

用户可以首先考虑购买合适的组件，其次再考虑自行开发，然后将这些组件按照一定的要求组装起来就能构建自己的应用系统，从而大大提高了工作效率，又便于日后的修改，从整体上来说降低了开发成本。

10.2　数据库系统开发的几个关键技术

10.2.1　数据库系统的组成

在图 10-1 中显示了数据库系统的主要组件。数据库包含 4 个要素，它们分别是：

- 用户数据：用户希望此系统保存和使用的数据。
- 元数据：数据库自身结构描述的数据。
- 索引：为了改变数据库的性能和可访问性所增加的一组辅助性数据。
- 应用元数据：用来存放用户表格、报表、查询和其他形式界面的数据。

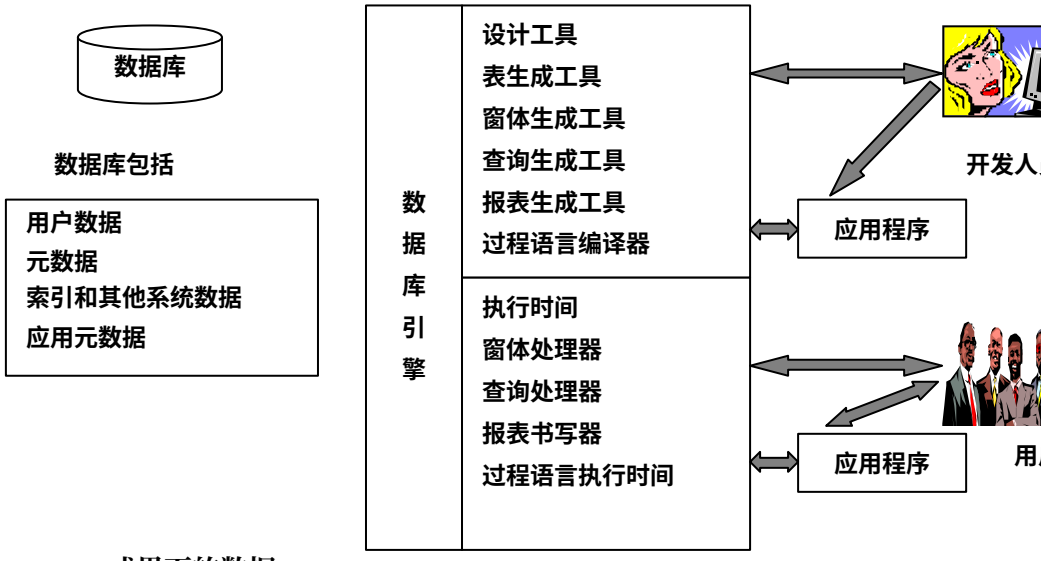


图 10-1　数据库系统的组件

数据库管理系统（DBMS）按其特点和功能可以划分为三个子系统：

- 设计工具子系统：它有一个方便数据库及其应用产生的工具箱集，典型的工具箱集包含产生表、窗体、查询和报表的工具，DBMS 还提供编程语言和对编程语言的接口。此部分功能通常由数据定义语言（DDL）实现，用来完成数据库定义功能。
- 运行子系统：它处理用设计工具开发的应用组件，此外还有一种运行组件，由它处理应用程序读写数据库数据的请求。此部分功能通常由数据操纵语言（DML）实现它完成数据库存取功能。
- DBMS 引擎：它介于设计工具子系统及运行子系统和数据库本身之间。DBMS 引擎从其他两个组件接受请求，并把它们翻译成对操作系统的命令，从而读写物理介质上的数据。DBMS 引擎还涉及事务管理、锁定、备份和恢复。此部分功能通常由数据控制语言（DCL）实现，它提供数据库例行程序。

10.2.2 开发策略

数据库是用户关于它们的业务活动的模型的模型。因此为了建一个有效的数据库及其相关的应用，开发人员首先应该完全熟悉用户模型。为此，要建立数据模型，以此来标识需要在数据库中存储的内容，并定义它们的结构及关系。这些工作应该在需求调查中完成。

开发数据库的策落有两种：自顶向下和自底向上。自顶向下是从一般到特殊，它开始于对组织战略目标、完成这些目标的方法、达到这些目标必须完成的工作和需要提供这些信息的系统进行研究，从这些研究可以构造抽象数据模型。使用这个高层模型，开发人员逐步构造越来越详细的描述以获得中层模型。对中层模型不断地扩展细化，直到构造出能识别特定的数据库及其应用的低层模型为止。接下来就可按项目的重要性，选择这些应用中比较重要的一个或一些来开发。随着时间的推移，整个高层数据模型都转换成了低层模型，所有指定的系统、数据库和应用就产生了。

自底向上方法采用与上相反的方式进行。始于开发特定系统，逐步构成较大系统乃至最终构成整个系统。

自顶向下的优点是，数据模型是用全局观点建立的，这种系统相互之间的接口好，一致性强。数据共享及完整性约束性能都比较好。

自底向上方式风险小，避免了自顶向下方式可能导致的许多难以实现的结果。自底向上方式尽管不一定能产生最好的系统，但却能很快地产生一个有用的系统。

在实际工作中常常把这两种方法结合起来使用，全局上采用自顶向下的方式，而在局部中则采用自底向上的方法。

10.2.3　数据建模

数据建模是通过一个从用户的陈述出发进行推理的过程而建立的。开发人员收集表格、报表和查询需求，反复来推断用户设想的结构。这一过程是必须的，因为大多数用户不能直接描述他们的数据摸型。根据我们的经验，向用户进行调查的部分表格可参考表 10-1 至 10-4。

表 10-1　　业务项目描述表

科室或单位名称：　　调研人：

业务人或岗位名称：　　调研日期：

业务名称	业务简述	业务来源	业务去向	业务流量(次/ 天)	备　注

表 10-2　　数据录入调查表

单位名称：　　调研人：　　业务人或岗位名称：

调研日期：

序号	录入具体内容	数据格式	录入频率	数据有效期限	数据保密等级	录入方式	备注

填表人：　　填表日期：　　审查人：

审查日期：

表 10-3　　数据查询调查表

单位名称：　　调研人：　　业务人或岗位名称：

调研日期：

序号	查询内容	查询格式	查询的频率	查询权限	查询建议	备注

填表人：　　填表日期：　　审查人：

审查日期：

表 10-4　　需求调查表

单位名称：　　调研人：　　业务人或岗位名称：

调研日期：

交叉重复 的业务	
无法理顺的业务	
旧系统中不满意的地方	
旧系统中满意的地方	
对不满意的改进建议	
希望新系统补充的功能	
最需要计算机取代的手工劳动	

填表人：　　填表日期：　　审查人：　　审查日

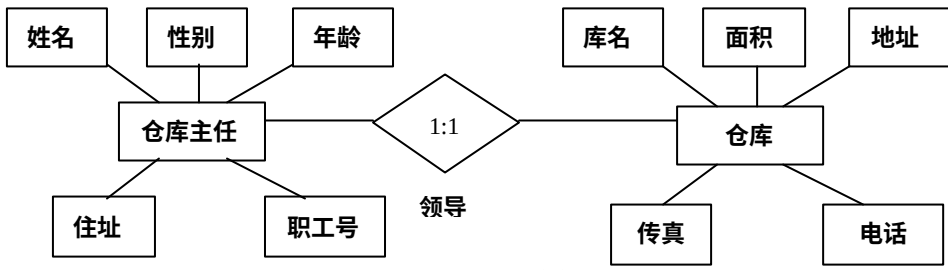
期：

传统的数据库模型有层次模型、网状模型和关系模型。其中关系模型应用最广，目前绝大部分数据库均采用关系型数据库。在概念级描述数据模型的方法有：实体—联系模型和语义对象模型以及类模型。

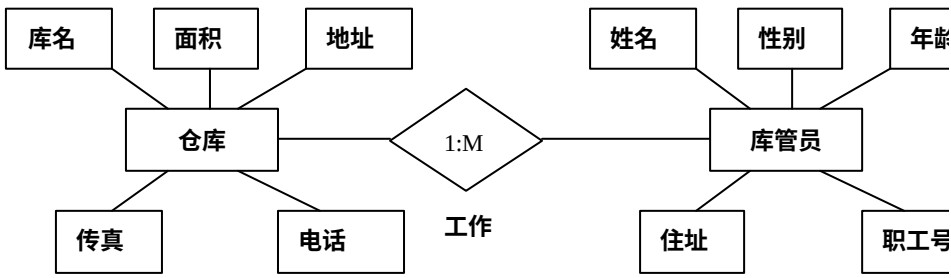
1．实体—联系模型

它又被称为 E-R（Entity-RelationshipApproach）模型，是由 PeterChen 在 1976 年提出的。实体是可以从用户工作环境中标识的事物。实体可归结为实体类或同一类型的实体集合。实体类是一个事物的一般形式或描述。而实体类的一个实例则表示一个特定实体。实体与实体类的关系类似于面向对象概念中的对象与类的关系。实体具有属性，有时也称作性质，是用来描述实体的特征的。实体可以通过联系相互关联。E-R 模型包含联系和联系类，联系类是实体类之间的联系，联系是实例之间的联系，联系也可拥有属性。

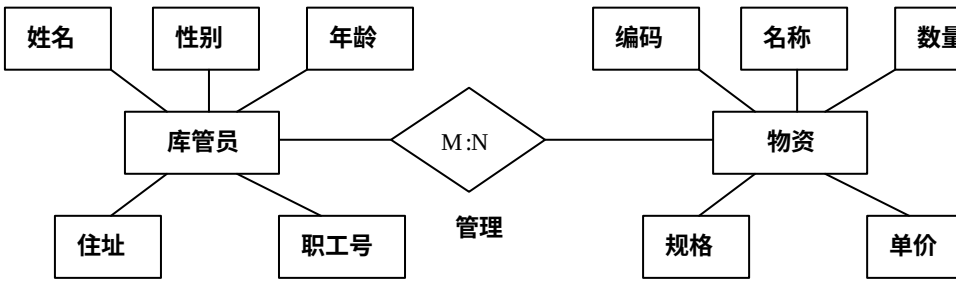
两个实体之间的联系有 3 种类型：1 对 1 联系、1 对多联系和多对多联系。例如一般而言，仓库正主任和仓库之间就是 1 对 1 联系（一个仓库只有一个正主任，一个仓库正主任也一定属于某个仓库），仓库和库管员之间就是 1 对多联系（库管员一定隶属于某个仓库，一个仓库可能有多个库管员），而库管员和货物之间就是多对多联系（一个库管员管多种货物，同一种货物可能由多个库管员保管）。在图 10-2 中描述了上述情况。



(a)



(b)



（c）

图 10-2　两个实体之间的联系

(a)1:1 联系　(b)1:M 联系　(c)M:N 联系

2．语义对象模型

语义对象的概念是 DavidM.Kroenke1988 年在《数据库处理》一书的第三版中提出的。在 E-R 模型中称为实体的事物在语义对象模型中被称为语义对象，语义对象是用来部分地对用户数据的含义建模的。语义对象模型比 E-R 模型更接近用户的感觉。语义对象（有时简称为对象）代表明确的本体，是足以描述一个确切本体属性的命名集合。他们被认为是独立的和分离的事物，是用户需要跟踪和报告的事物。对象也有一个属性集合，每个属性代表所表示对象的一个特征。属性有三种类型，简单属性是单值的，例如姓名、性别等。属性组是其他属性的组合。例如地址，它包含属性{省份，区/县，街道}，另一个例子是电话号码，它包含属性{区号，市内号码}。第三种属性是语义对象属性，它是在语义对象和另一个语义对象之间建立关系的属性。在图 10-3 中给出了仓库系统语义对象的描述。

3．类模型

在采用面向对象方法开发软件时则可采用类模型。面向对象中类概念与语义对象既相似又不同。它们的相似之处是，都是用来为数据及其关系建模，都包含封装结构的构件，都有对象继承。它们的区别在于，面向对象中的类提供方法定义、方法继承以及多态性，而语义对象则没有方法及有关的结构。

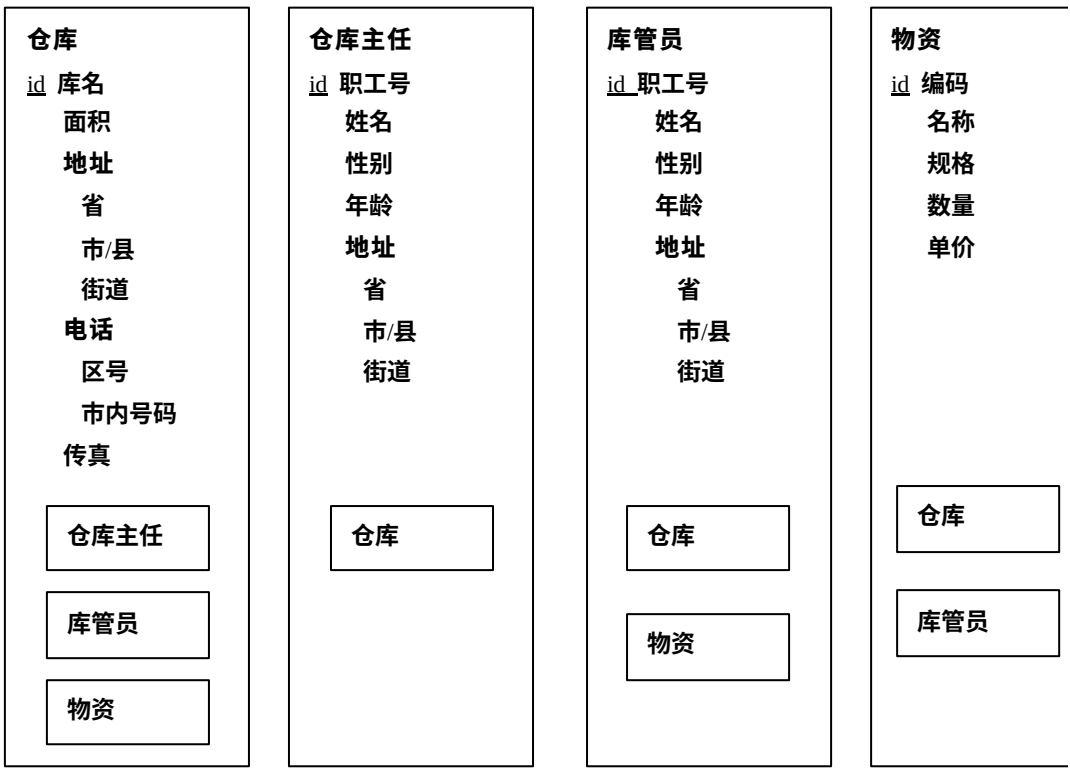


图 10-3　仓储系统语义对象模型

10.2.4　数据规范化

在分析问题的过程中，通常把问题分解，按子系统建模，然后再把子模型集成起来形成完整系统模型。在集成的过程中会产生数据冗余，有时会发生冲突，冲突体现在以下三个方面：

- 属性冲突：包括属性值的类型、取值范围和取值集合不同。单位冲突（如：某些货物重量单位用吨而另一些用公斤，某些货物的单位用米等）。
- 结构冲突：同一属性组的外延不同。不同 E-R 图中实体和属性存在冲突，例如一个事物在此处被看作属性，而在其他地方又被看作实体。
- 命名冲突：同名异义，异名同义。

冲突可通过复查并制定统一规范来消除。

冗余数据指可由基本数据导出的数据，冗余的存在容易破坏数据库的完整性，可用分析法消除冗余，但最常用的却是规范化理论。

规范化理论的一个重要的目标就是将数据分割存储在特定的地方来消除冗余、简化数据库的更新，提高数据库完整性，并且减少存储的要求。常用范式来衡量数据规范化的程度。现在已定义出了第 1、第 2

至第 5 范式，这些范式是嵌套的，也就是说某关系当满足第 2 范式时一定满足第 1 范式。

在规范化理论中所定义的关系是一个二维表，它满足下列约束：首先，表中的每一格必须是单值的，每一列的所有条目都必须是同一类型的；其次每一列都有唯一的名字，列在表中的顺序并不重要；最后，表中任意两行（元组）不能完全相同，行在表中的顺序也不重要。

所谓第一范式（1NF）是指，任何符合关系定义的关系满足第 1 范式。

所谓第二范式（2NF）是指，如果一个关系的所有非关键字属性都依赖于整个关键字，那么该关系就属于第 2 范式。所谓关键字是指唯一能标识一个元组的字段（属性）或字段的集合，在职工工资档案表中职工号就可作为关键字，但职工有可能同名，因此，姓名不能作为关键字。

第三范式：一个关系如果在第 2 范式，且没有传递依赖，则该关系在第 3 范式中。

随着关系的一步步规范化，对其进行插入、删除、更新的异常操作会逐步消除。对于其他范式的定义可参阅有关文献。

规范化的关系避免了更新异常，似乎更可取，但从其他方面来考虑，有时却不值得。因为当数据必须从两个单独的表中组合起来时，DBMS 就要做额外的关系运算。因此非规范化的表可能更易于处理。在有些情况下，数据冗余的缺点并不是很重要，往往在规范化的基础上故意保留少量非规范化的内容，以改进性能，并简化实现的复杂性。一般说来，实际操作中，第三范式已能满足商业化数据库的一致性和完整性约束。

10.2.5　数据模型到关系数据库的映射

1．E-R 模型到关系数据库的映射

一般地讲，使用关系模型表示实体是很直接的。首先，为每个实体定义一个关系，关系的名字就是实体的名字，关系的属性就是实体的属性。然后根据规范化准则检查每个关系，初始的设计可能需要改变，也可能不需要改变。如果从数据模型能知道哪个属性标识该实体，则该属性就是关系的关键字。E-R 模型到关系数据库的映射分下述几种情况：

(1) 在 1∶1 关系中，首先将一个实体用一个关系表示，然后将一个关系的关键字置于另一个实体中，形成 1 对 1 关联。一个关系的关键字存贮在另一个关系中时，称作另一个关系的外键。

(2) 在 1∶N 关系中，首先将一个实体用一个关系表示，然后将代表父实体的关系（1 的一方）的关键字置于代表子实

体（多的一方）的关系中，形成 1 对多关联。

(3) 在 M：N 关系中，首先将一个实体用一个关系表示，然后建第三个关系，以表示多对多关系本身并在该关系中存放原来两个关系的主键。这样就把多对多关系转换为一对多关系了。

2．语义对象模型到关系数据库的映射

分下述几种情况：

(1) 简单对象到关系数据库的映射。简单对象是没有多值属性和语义对象属性的，因此简单对象在数据库中可用单个关系表示。

(2) 组合对象到关系数据库的映射。组合对象是指包含一个或多个多值简单属性或属性组但不包含对象属性的对象。通常是把组合对象本身定义成一个关系，并为每个多值属性也定义一个关系，并把组合对象的关键字放入多值属性所对应的表中。

(3) 复合对象模型到关系数据库的映射。复合对象至少包含一个对象属性。这时可按复合对象与包含对象的关系是属于 1：1，1：M 或 M：N 关系参照 E-R 模型的方法处理。

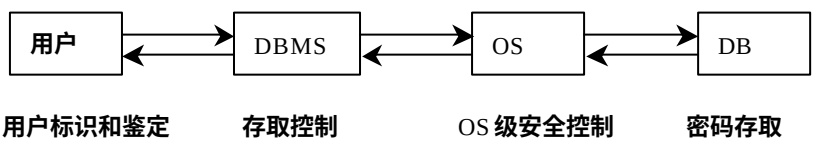
还有混合对象、关联对象、父子类型对象及原型/版本对象，它们的转换方法可参考有关文献。

3．类到关系数据库的映射

此映射方法参见 12 章第 7 节。

10.2.6 数据库的安全设计

随着 Intranet/Internet 的普及及电子商务活动的开展，网络安全已日益成为人们所关注的话题。在一般的计算机系统中安全措施是逐级设



置的，如图 10-4 所示。

图 10-4　网络安全级别

例如在目前的 Windows NT 中，用户上网需先输入用户名及口令，在连接到数据库服务器时也要输入相应于该数据库的用户名和口令。由于企业内部信息多，各部门不同人员之间要共享的数据权限有很大差异，因而，在 MIS 设计一开始就应充分重视这一问题，并结合动态菜单授权技术才能设计出一个对用户来说是安全、可靠、使用方便的 Intranet 系统。如果忽略了这一问题，或者这一问题解决得不好，有可