数据库应用开发技术丛书

Java

数据库高级教程

张晓东　等编著

清华大学出版社

数据库应用开发技术丛书

Java 数据库高级教程

张晓东　等编著

清华大学出版社

北　京

内 容 简 介

本书由浅入深地介绍了 Java 与数据库应用开发的方方面面，并深入讨论了相关的高级技术专题。全书分 15 章，内容包括 Java 和数据库的基本知识，流行的数据库产品，数据库的基本操作和应用，数据库的高级应用技术，以及一些较新而又非常重要的技术专题。通过这些专题的学习，将会使读者受益非浅。

本书示例丰富，讲解深入透彻，使读者可以系统、全面地学习 Java 和数据库应用开发高级技术。本书适用于有一定 Java 和数据库开发经验的读者，对于从事在 Java 平台上开发 ERP 应用软件的专业人员来说，也是一本很好的参考书。

图书在版编目(CIP)数据

Java 数据库高级教程/张晓东等编著. —北京：清华大学出版社，2004
(数据库应用开发技术丛书)
ISBN 7-302-08369-X

Ⅰ.J… Ⅱ.张… Ⅲ. Java 语言—程序设计 Ⅳ.TP312
中国版本图书馆 CIP 数据核字(2004)第 026108 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
社 总 机：010-62770175　　**客户服务**：010-62776969
组稿编辑：孟毅新
文稿编辑：许书明
封面设计：久久度企划
版式设计：康　博
印 装 者：三河市印务有限公司
发 行 者：新华书店总店北京发行所
开　　本：185×260　**印张**：32.25　**字数**：759 千字
版　　次：2004 年 4 月第 1 版　2004 年 4 月第 1 次印刷
书　　号：ISBN 7-302-08369-X/TP·6023
印　　数：1～4000
定　　价：46.00 元

前　言

现在市面上关于Java、J2EE以及数据库方面的图书很多，但是很多图书都是大而全的初级参考书，对于一些有一定经验的读者来说，那些书的内容过于浅显而且重复。而关于Java与数据库应用的专业性书籍并不多，特别是系统的介绍Java与数据库高级应用的书籍更是很少。本书注重广泛的知识结构，内容广泛，又不失知识结构之间的相互关联性，正好可以弥补这方面书籍的空缺。

本书内容详实，是Java与数据库应用方面的一本精品书籍，特别适合想在这方面应用有较深入了解的读者。本书通过大量的实例讲解了Java与数据库应用的各个方面，并给出了一些有用的建议和资源供读者进一步学习。同时也分析了当今相关的最新技术发展情况，希望能让读者获得良好的技术基础，并提供一些思路和启发。

本书的结构主要是采用专题的方式，在最初的5个章节中，是关于Java和数据库的基本知识介绍，读者如果缺乏这方面的知识，那么按顺序学习这5章还是很有益的。如果对于这些比较基本的操作已经很熟悉，那么可以略过一些内容，但是并不建议完全略过这几章的内容不看，因为这几章中本书还补充了一些新的知识和应用技巧，还是会对读者有所帮助的。其后的各个章节，彼此间的联系不再像前面的5章那样密切，推荐读者在阅读的时候，根据自己的需要有选择地学习所需要的知识专题。如果读者有足够的时间全部学习当然会更有益处。本书采用这种结构是希望能更加方便读者的使用，所以可以根据具体情况来选择学习的内容和方式。

由于技术的进步日新月异，软件技术和产品的更新速度也越来越快，而本书的内容当然要受到一些时间因素的限制。为了尽量避免这种滞后所造成的影响，在本书中的一些相应部分提供了很多最新技术和知识的网络链接，这样读者在使用本书的时候，可以随时查阅最新的知识和更新的内容，保持与新技术的同步。

在本书的写作过程中还要感谢同事和朋友对我的帮助，刘生平、王喆、刘壁松等同志都在写本书的过程中给予了很大的帮助和贡献，我在微软工作的同事同样给予我很大的帮助，他们给了我很多的意见和建议，使我写作本书能够更加顺手，本书的很多章节和观点都要感谢他们的无私奉献。另外，高海峰、王建胜、崔凌、魏勇、郝荣福、武思宇、牟博超、李彬、付鹏程、高翔、朱丽云、张巧玲、喻波、马天一、李辉、柏宇、郭强、金春范、程梅、李光龙、孙明、李大宇、黄霆、钟华、张浩、李湘和李欣等同志在整理材料方面给予了作者很大的帮助。在此，对他们表示感谢。

虽然笔者在编写本书的过程中竭尽全力，但是由于各方面的原因，书中难免会存在一些不让人满意之处，您的各种意见和批评请致信asnowstar@sina.com。

目　　录

第1章　Java面向对象的系统设计和数据库建模

Java 语言是 Internet 上的通用语言，同时也是最佳的网络应用开发语言。Java 的诞生从根本上解决了 Internet 的异质、代码交换以及网络程序的安全性等诸多问题。总的来说，Java 是一种简单、面向对象、分布式、解释、稳健、安全、结构中立、可移植、高效能、多线程、动态的语言。而正是这种语言在世界的 IT 产业掀起了一股巨大的浪潮。

Java、Internet 以及 WWW 的发展和广泛应用改变了应用软件开发和使用方式，如今软件开发和应用都要围绕着网络，围绕着“平台无关”这个主题。信息的价值在于使用和共享，Internet 和 Web 是信息使用和共享最快捷、最便宜的方式。另一种支持 Internet 发展的重要技术便是数据库技术，正是数据库技术的发展，促进了分布式网络的发展和应用，数据的共享和分布式存储成为现代数据存在的方式。数据库管理系统发展到了今天，在普通应用方面可以说已经接近极限。多年以来，人们一直在追求数据库系统与程序设计语言的完美结合。未来数据库系统发展的趋势将是：面向对象数据库和关系数据库不断融合。对象关系数据库由于继承了上述两者的优点，已经成为目前数据库发展的一个主要方向。

很多人认为，数据库编程是一个颇具诱惑力的工作，这份诱惑力来自于数据库应用的巨大市场需求。计算机应用有科学计算、数据处理与过程控制三大主要应用领域，而数据处理是其中所占比重最大的一个应用领域。现在最流行的客户机/服务器模式(C/S)、Internet 模式(B/S)应用，从广义上讲都可以归入此应用领域。Java 与数据库技术的结合也正是本书的切入点，在本书中分别通过各个不同的专题的方式来介绍 Java 和数据库的相关技术和知识。

1.1　面向对象技术简介

面向对象的软件开发方法第一次提出是在 20 世纪 60 年代，但 20 多年之后才开始变得流行和被广泛使用。面向对象技术首先是从面向对象编程(OOP)开始的，那时面向对象(OO)的思想只是被用于一系列的程序设计语言，以及基于其上的软件开发方法。接下来面向对象分析(OOA)、面向对象设计(OOD)、面向对象测试(OO 测试)等迅速发展起来，这样到了 20 世纪 90 年代，面向对象软件工程(OOSE)变成了很多软件产品的建造者，以及数量不断增长的信息系统和工程专业人士的首选范型。

1.1.1　面向对象软件工程(OOSE)

Jacobson 于 1994 年提出了 OOSE 方法，其最大特点是面向用例(Use Case)，并在用例的描述中引入了外部角色这个概念。用例的概念是精确描述需求的重要方法，用例贯穿于整个开发过程，包括对系统的测试和验证。OOSE 比较适合于应用在商业工程支持和需求分析方面。它涉及到整个软件生命周期，包括需求分析、设计、实现和测试等四个阶段。需求分析和设计密切相关。需求分析阶段的活动包括定义潜在的角色(角色指使用系统的人和与系统互相作用的软、硬件环境)、识别问题域中的对象和关系、基于需求规范说明和角色的需要来发现 Use Case、详细描述 Use Case。设计阶段包括两个主要活动：从需求分析模型中发现设计对象，以及针对实现环境调整设计模型。第一个活动包括从 Use Case 的描述发现设计对象，并描述对象的属性、行为和关联。在这里还要把 Use Case 的行为分派给对象。

在需求分析阶段的识别领域对象和关系的活动中，开发人员识别类、属性和关系。关系包括继承、属性(关联)、组成(聚集)和通信关联。定义 Use Case 的活动和识别设计对象的活动共同完成了行为的描述。Jacobson 方法还将对象区分为语义对象(领域对象)、界面对象(如用户界面对象)和控制对象(处理界面对象和领域对象之间的控制)。

Jacobson 的 OOSE 方法中的关键概念就是 Use Case。Use Case 是指行为相关的事务(transaction)序列，该序列将由用户在与系统对话中执行。因此，每一个 Use Case 就是一个使用系统的方式，当用户给定一个输入，就执行一个 Use Case 的实例并引发执行属于该 Use Case 的一个事务。基于这种系统视图，Jacobson 将 Use Case 模型与其他五种系统模型关联，这五种模型是：

(1) 领域对象模型　Use Case 模型根据领域来表示
(2) 分析模型　Use Case 模型通过分析来构造
(3) 设计模型　Use Case 模型通过设计来具体化
(4) 实现模型　该模型依据具体化的设计来实现 Use Case 模型
(5) 测试模型　用来测试具体化的 Use Case 模型

事实上 Jacobson 所说的 OOSE 是一种狭义上的面向对象软件工程，而广义上的面向对象软件工程则可以从 Edward Bernard 的如下论述中略知一二：如果在软件工程过程的早期就全程强调面向对象技术，则该技术的收益将更多。那些考虑面向对象技术的人们必须要评估它对这个软件工程过程的影响。仅仅是用面向对象程序设计(OOP)将不会产生最好的结果，软件工程师及其管理者必须考虑面向对象需求分析(OORA)、面向对象设计(OOD)、面向对象领域分析(OODA)、面向对象数据库系统(OODBMS)和面向对象计算机辅助软件工程(OOCASE)。

Edward Bernard 推荐在整个软件过程中都采用某种正规的软件工程方法。广义上讲，OOSE 是指利用面向对象技术进行整个软件开发过程管理和控制的软件工程方法。现今对软件工程的研究也非常热门，CMM 和 Agile 可以说是当今流行的两大软件工程方法，它们

都是基于 OOSE 的。

面向对象的分析与设计(OOA&D)方法在 20 世纪 80 年代末至 20 世纪 90 年代中期出现了一个发展高潮，UML 是这个高潮的产物。它不仅统一了 Booch、Rumbaugh 和 Jacobson 的表示方法，而且对其作了进一步的发展，并最终统一为大众所接受的标准建模语言。下面简单的介绍一下这些软件工程的各个不同方面。

1.1.2　面向对象分析(OOA)

面向对象分析是软件开发过程中的问题定义阶段。这一阶段最后得到的结果是对问题领域的清晰、精确的定义。传统的系统分析产生一组面向过程的文档和定义好的目标系统的功能。面向对象分析则产生一种描述系统功能和问题领域基本特征的综合文档。它关注的部分不再局限于与问题直接相关的部分，而是在更大领域范围里考虑问题。在分析过程中识别的概念是高层的抽象，这些抽象成为一个灵活的可扩充的软件基本构件块。面向对象需求分析可以说是 OOA 的一种具体细化。面向对象的方法进行需求分析集中在对象和类以及对象之间的动态交互上。

Coad 和 Yourdon 认为 OOA 主要关注于与一个特定应用有关的对象，以及对象与对象之间在结构和相互作用上的关系。OOA 有两个任务：一是形式地说明所面对的应用问题，最终成为软件系统基本构成的对象，还有系统所必须遵从的，由应用环境所决定的规则和约束；二是明确地规定构成系统的对象如何协同合作，完成指定的功能。

OOA 是对现实世界中的“问题空间”建模，其分析侧重于总体，是较为粗糙(粒度较大)的研究，它是完全独立于编程语言的。而 OOD 则要求对特定的实现空间建模。所以说，OOA 的各层次将“问题空间” 模型化，而 OOD 各层扩充了 OOA，模型化一个特定的“理论空间”。

公认的面向对象建模语言出现于 20 世纪 70 年代中期，并在随后的实践中衍生出多种建模语言。20 世纪 90 年代中期，一批新方法出现了，其中最引人注目的是 Booch 1993、OOSE 和 OMT-2 等。在以后的十几年中，Booch、Rumbaugh 和 Jacobson 一起协作，将它们各自的面向对象分析和设计方法的最好的特征组合成为一种统一方法，其结果成为统一建模语言(Unified Modeling Language，UML)，已经在整个业界广为使用。如图 1-1 所示显示了 UML 语言的发展历程。

统一建模语言(Unified Modeling Language)是一种用于描述、构造软件系统以及商业建模的语言，综合了在大型、复杂系统的建模领域得到认可的优秀的软件工程方法。UML 是大多数公司采用的标准，也是 ANSI 和 OMG 等组织采用的标准。

UML 的产生有三方面的原因：首先，不同的面向对象方法有着许多相似之处，通过这项工作，能够消除可能会给使用者造成混淆的不必要的差异；其次，语义和表示法的统一，可以稳定面向对象技术的市场，使工程开发可以采用一门成熟的建模语言，CASE 工具的设计者也可以集中精力设计出更优秀的系统；第三，这种统一能使现有的方法继续向前发展，积累已有的经验，解决以前没有解决好的问题。

UML 为软件系统建模提供了以下 4 个方面的支持。

- 使用事件模型(Use Case)：定义系统的使用事件(Use Case)、角色(Actor)及角色与事件之间的交互行为(Association)。
- 类和对象模型：定义类、对象及相互之间的关系。
- 组件模型：组件是组成应用程序的可执行单元，类被分配到组件中，以提供可重复使用的应用程序结构部件。UML 对可重用性的支持，在设计的前期体现在支持可重复使用的类和结构，后期则体现在组件装配。
- 分布处理模型：将软件系统映射到分布处理结构中。UML 能够描述网络拓扑结构的节点，这些节点相互的连接方式以及软件系统在网络中的分布情况。

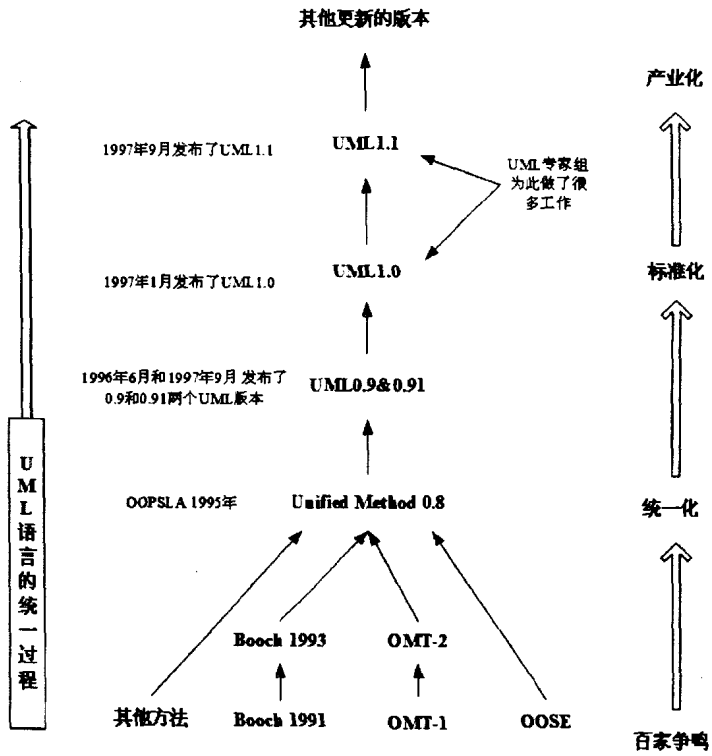

图 1-1　UML 发展历程

作为一种建模语言，UML 的定义包括 UML 语义和 UML 表示法两个部分：

UML 语义描述基于 UML 的精确元模型定义。元模型为 UML 的所有元素在语法和语义上提供了简单、一致、通用的定义性说明，使开发者能在语义上取得一致，消除了因人而异的表达方法所造成的影响。此外 UML 还支持对元模型的扩展定义。

UML 表示法是指定义 UML 符号的表示法，这些图形符号和文本语法为系统建模提供了标准。这些图形符号和文字所表达的是应用级的模型，在语义上它是 UML 元模型的实例。

标准建模语言 UML 的重要内容可以由下列五类图(共 9 种图：用例图、类图、对象图、

状态图、顺序图、活动图、协作图、组件图、部署图)来定义：用例图(Use Case diagram)、静态图(Static diagram)、行为图(Behavior diagram)、交互图(Interactive diagram)和实现图(Implementation diagram)。

1.1.3　面向对象设计(OOD)

面向对象的设计(Object-Oriented Design，OOD)主要是利用面向对象的技术建立能够集成产品设计和制造信息的产品定义模型，即面向对象的产品定义模型，在该模型的基础上实现系统的设计。由这个定义，我们可以知道 OOD 的本质就是“根据需求决定所需的类、类的操作以及类之间关联的过程”。

OOD 的目标是管理程序内部各部分的相互依赖。为了达到这个目标，OOD 要求将程序分成模块(Module)，每个模块的规模应该小到可以管理的程度，然后分别将各个模块隐藏在接口(Interface)的后面，让它们只通过接口相互交流。例如，用 OOD 的方法来设计一个服务器-客户端(Server-Client)应用，那么服务器和客户端之间不应该有直接的依赖，而是应该让服务器的接口和客户端的接口相互依赖。

这种依赖关系的转换使得系统的各部分具有了可复用性，客户端就不必依赖于特定的服务器，可以复用到其他的环境下。如果要复用某一个程序块，只要实现必需的接口即可。

OOD 是一种解决软件问题的设计范式(Paradigm)，是一种抽象的范式。使用 OOD 这种设计范式，可以用对象(Object)来表现问题领域(Problem Domain)的实体，每个对象都有相应的状态和行为。刚才说到 OOD 是一种抽象的范式，抽象可以分成很多层次，从非常概括的到非常特殊的都有，而对象可能处于任何一个抽象层次上。另外，彼此不同但又互有关联的对象可以共同构成抽象，只要这些对象之间有相似性，就可以把它们当成同一类的对象来处理。

OOD 模型设计由四个部分构成，它们是：问题域部分、人机交互部分、任务管理部分和数据管理部分。每一部分的设计都在五个层次上进行，即对象层、结构层、主题层、属性层、服务层。

分析是问题抽象(做什么)，设计是问题求解(怎么做)，实现是问题的解(结果)。任何方法学对客观世界的抽象和求解过程都是如此。

在软件工程基本原则中有一条“形式化原则”，即对问题世界的抽象结论应该以形式化语言(图形语言、伪码语言等)表述出来。结构化方法可以用数据流图、系统结构图、数据字典、状态转移图、实体关系图来进行系统逻辑模型的描述；而面向对象方法可以使用对象模型图、数据字典、动态模型图、功能模型图。其中对象模型图类似于系统结构图与实体关系图的结合，动态模型图类似于状态迁移图，功能模型图类似于数据流图。

和 OOA 方法一样，UML 也是适用于 OOD 的，因为 UML 本身就是由对象分析和设计方法的最好特征而形成的统一方法。UML 的目标是以面向对象图的方式来描述任何类型的系统，具有很宽的应用领域，其中最常用的是建立软件系统的模型，但它同样可以用于描述非软件领域的系统，如机械系统、企业机构或业务过程，以及处理复杂数据的信息系