Hibernate 3

和 Java Persistence API 程序开发

从入门到精通

葛 京 编著

附光盘

提供本书综合实例的
完整开源项目和配置文件

清华大学出版社

Hibernate 3

和 Java Persistence API 程序开发

从入门到精通

葛 京 编著

附光盘

提供本书综合实例的
完整开源项目和配置文件

清华大学出版社
北京

内 容 简 介

本书介绍 Hibernate 开发知识。全书主要介绍信息持久化、对象持久化、对象关系映射等概念，以及 Java SE5 中最重要的新特性，初始化 Hibernate 的核心类，使用 JPA 提供的标准注释将实体类的不同属性映射到关系型数据库的表格，对象关系映射所要面对的两种复杂关系，使用 Hibernate 的扩展注释映射各类复杂的集合接口和集合类，实体对象在 Hibernate 中存在的 4 种状态，Session 接口，Hibernate 建立的一套完善的查询框架(Criterion)，Hibernate 中封装 SQL 的策略，Hibernate EntityManager 模块的内部构架。

本书面向中高级程序开发人员，适合 Hibernate 程序员、系统构架师、项目经理以及负责协调管理项目开发的开发人员使用。

图书在版编目（CIP）数据

Hibernate 3 和 Java Persistence API 程序开发从入门到精通 / 葛京编著. —北京：清华大学出版社，2007.10
ISBN 978-7-302-15801-1

Ⅰ. H…　Ⅱ. 葛…　Ⅲ. JAVA 语言-程序设计　Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2007）第 113879 号

责任编辑：夏兆彦
责任校对：张　剑
责任印制：杨　艳
出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　**邮　　编**：100084
c-service@tup.tsinghua.edu.cn
社 总 机：010-62770175　**邮购热线**：010-62786544
投稿咨询：010-62772015　**客户服务**：010-62776969
印 刷 者：北京鑫丰华彩印有限公司
装 订 者：三河市李旗庄少明装订厂
经　　销：全国新华书店
开　　本：203×260　**印　张**：25.5　**字　数**：664 千字
附光盘 1 张
版　　次：2007 年 10 月第 1 版　　**印　　次**：2007 年 10 月第1 次印刷
印　　数：1～4000
定　　价：49.00 元

本书如存在文字不清、漏印、缺页、倒页、脱页等印装质量问题，请与清华大学出版社出版部联系调换。联系电话：(010)62770177 转 3103　　产品编号：024359-01

前言

FOREWORD

最近十几年来，信息科技可谓日新月异、飞速发展，几乎每隔几年就会发生一次大规模的技术升级。这些名目繁多的技术革新，看似变化多端，实际上完全围绕着信息在进行，可谓形散而神不散。从总体上来看，这些升级主要是为了让人们能够更好地加工、传输和存储信息。

面向对象语言的问世，如 Java 成功地将对象的概念引入软件世界，从此编程语言更加接近人类的思维方式，使人们加工信息的能力得到空前的提高。随着面向对象概念逐渐丰富、不断地成熟，一些更为高级的信息加工技术开始被大家接受。

首先出现的是基于模块编程。开发人员只需要选用适当的模块并把它们拼接在一起，然后再根据项目的实际需要进行必要的扩展即可，而不必像从前那样，即项目中的任何功能模块都必须亲自编写。这种情况可以与布置一个房屋进行类比。假如我们购买了一套房屋，需要添置家具，我们会去家具城选购沙发、书架、床、衣橱等，然后在自己的房屋中布置一番。这种“购买家具，自己组合”就是一种模块的概念，家具就是模块，需要什么模块时，只需拿来使用即可。那么不使用模块会如何呢？仍以家具为例。如果需要什么家具，我们必须自己动手把这个家具做出来。可以想象，布置一个房屋将需要我们自己动手做沙发、床、衣柜，等等，这样整个过程将会苦不堪言。

互联网彻底改变了以往信息传递的方式，从此，人们不再需要为传递信息而发愁。随着互联网的广泛应用，产生了一种更高级的信息加工技术：面向服务构架（SOA）。这种技术可以说是模块技术向着网络化的一种延伸，人们可以通过网络调用需要的模块，而不必将该模块安装在本地，从而省去了安装部署模块的烦琐。在理想情况下，SOA 技术甚至可以将整个地球上的所有服务器连接使之成为一个功能强大的超级软件。

可见，信息加工和信息传输技术的发展的确令人兴奋，已经可以满足绝大多数项目的需求。但是，信息存储技术领域的发展却差强人意。十多年过去了，存储领域的主流仍然是关系型数据库。如何将对象保存到关系型数据库中以及如何从数据库表格中提取对象等这类问题是每一个项目都必须正视和加以解决的问题。

虽然在 Java 领域里，开发人员可以使用 JDBC 接口类库完成所有对象和关系型数据库之间的存取工作，但是整个过程的编码工作十分繁琐，一个非常简单的存取过程往往需要十几行甚至几十行代码。更繁琐的是，每一个数据库应用项目都必须反复编写类似的 JDBC 代码。难道就没有人出来做沙发，让希望使用沙发的人直接拿来就用么？答案是：有，而且不只一个、不只一个种类。目前工业上使用比较广泛的有 Hibernate，iBatis，Spring JDBCTemplate，TopLink，EJB2 等。其中 Hibernate 是这个领域里面绝对的领导者，占据着最大的市场份额。

Hibernate 是一种对象关系映射（ORM）解决方案，它主要是为了挑战当时的工业标准 EJB2 中的 EntityBean 而出现的。EJB2 的设计人员由于没能正确把握当时的市场需求，导致 EJB2 规范存在着严重缺陷，比如基于 EJB2 的项目开发周期十分漫长、产品对 EJB2 容器过分依赖。Hibernate 则另辟蹊径，提供轻量级容器，轻松实现对象和关系型数据库之间的转换工作。作为当今市场上

无可争议的最佳 ORM 方案，Hibernate 具有如下优势。

- Hibernate 是开源项目。开发人员不但可以在项目中免费使用它，而且可以得到它的源代码。因此，在遇到问题时可以查看源代码了解 Hibernate 的运作原理。
- Hibernate 属于 JBoss 公司所有。因此采用 Hibernate 技术的公司可以得到商业级别的技术支持，而不用担心突然有一天有人宣布停止开发 Hibernate、停止提供技术支持。
- Hibernate 通过了认证，希望使用工业标准的公司可以放心大胆地使用 Hibernate。
- Hibernate 作为 JPA 的试验田将超前提供很多实用的功能。它是希望使用最新功能的公司的首选。
- JBoss 公司为 Hibernate 搭建了一个完善的社区。开发人员一旦遇到问题，则可以很快在社区内或者互联网上找到解决方案。
- Hibernate 采用轻量级容器，应用工业界几个比较著名的设计模式。它简单易用。学习周期相对较短。

本书结构

本书目的是希望读者在理解 Hibernate 内部基本工作原理的同时，由浅入深，一步一步地了解如何配置 Hibernate，如何使用 Hibernate 提供的接口开发应用程序，如何优化系统性能。考虑到实际项目之间存在着很大的区别，深入介绍一个完整的项目可能会对某些读者提供一定的帮助，但是对于绝大部分读者而言则效果不佳。因此本书始终围绕着一个简单的例子，逐渐将 Hibernate 的强大功能展现在读者面前，目的是为了给予读者点金之指，而非金山本身。

本书大体上包括三大部分，第一部分从第 1 章到第 3 章，主要介绍阅读本书需要的一些基本知识。第二部分为 Hibernate 入门（从第 4 章到第 6 章），主要介绍有关配置 Hibernate、基本实体类映射的知识。第三部分为 Hibernate 进阶（从第 7 章到第 13 章），详细介绍 Hibernate 的各方面的知识。另外，在本书的最后有相关的附录。

第 1 章首先从理论入手，讨论信息持久化、对象持久化、对象关系映射等概念以及它们所面对的问题，然后对市场上几种主流的对象——数据库持久化框架进行比较，最后着重介绍 Hibernate 之所以如此受欢迎的原因。

第 2 章介绍 Java SE5 中几个最重要的新特性，包括枚举类型、泛型、Auto-bosing 和 Unboxing、注释、静态导入以及 for/in 循环。主要目的是为了照顾那些还没有接触过 Java SE5 的读者，让他们对 Java SE5 有个基本的认识，从而不至于在碰到这些新代码时无所适从。

第 3 章介绍开发 Hibernate 项目所需要的基本工具软件，它们分别是 Hibernate，ANT，JUnit，HSQLDB，Eclipse 和 Eclipse 上的 Quantum DB 插件。

第 4 章采用测试驱动开发的方式介绍如何使用工具软件开发一个简单的 Hibernate 项目。这一章按照开发实际项目的步骤，逐步介绍了如何建立项目，如何配置 Hibernate，如何编写 JUnit4 测试代码，如何编写实体类和 CRUD 代码以及如何使用 ANT 构建项目。

第 5 章从构架角度详细介绍几个负责初始化 Hibernate 的核心类，并由此引出两种配置 Hibernate 的方法，然后详细介绍配置文件中各个属性的含义，在这一章的末尾还介绍如何配置 Hibernate 的日志，以方便开发人员观察程序运行状况，了解 Hibernate 内部工作原理。

第 6 章首先简单地比较两种为实体类添加映射元数据的方式：影射文件方式和注释方式，然后开始着重介绍如何使用 JPA 提供的标准注释将实体类的不同属性映射到关系型数据库的表格中

去。另外还单独列出一节介绍 JPA 生成对象标识符的四种方法。

第 7 章以三维坐标的方式引出对象关系映射所要面对的 2 种复杂的关系：对象关联和对象继承。深入介绍如何使用 JPA 提供的标准注释完成这些复杂的映射工作。其中，映射关联对象需要面对方向和数量这两个关联属性，它们之间的排列组合一共产生七种关联情况；而映射继承类则需要面对类与表之间阻抗失衡的问题。另外这一章还介绍如何映射基本的集合接口。

第 8 章分为两部分。第一部分着重介绍如何使用 Hibernate 的扩展注释映射各类复杂的集合接口和集合类，以及如何映射以基元数据作为元素的集合对象。第二部分着重介绍如何扩展 Hibernate 的映射逻辑，以及如何映射用户自行定义的类。

第 9 章也分为两个部分。第一部分深入介绍实体对象在 Hibernate 中存在的四种状态，以及它们之间如何进行转换。第二部分着重介绍 Hibernate 的基本事务和缓存功能，以及与事务和缓存相关的编程接口。

第 10 章介绍 Hibernate 的心脏——Session 接口。为了让读者能够更清晰地了解 Hibernate 内部持久化操作的具体过程，这一章详细介绍 Hibernate 3 全新开发的事件监听机制和拦截器，并且举例说明如何在项目中使用它们。

第 11 章首先简单回顾 SQL 语言的一些基本使用方法，然后从构架角度深入介绍 Hibernate 建立的一套完善的查询框架（Criterion）。这套框架完美地将 SQL 语句封装在 API 之内，开发人员可以在毫无 SQL 经验的情况下开发出令人满意的查询模块。

第 12 章讲述 Hibernate 中另外一种封装 SQL 的策略，这种策略是将面向对象的特性融入 SQL 中，再将 SQL 升级成为面向对象的 HQL。此外，在这一章的结尾还通过实例介绍如何使用过滤器，过滤查询结果。

第 13 章深入分析 Hibernate EntityManager 模块的内部构架，介绍 Hibernate 如何巧妙地实现 JPA 规范，同时又最大限度地将自身与 JPA 接口进行解耦合。另外这一章还通过重构一个实例向读者介绍如何使用 Hibernate EntityManager 的相关接口进行实际开发。

本书适用对象

本书不要求读者具备数据库以及 ORM 的相关知识，但是，如果读者希望能够很顺利地学习本书内容，那么读者应该具备基本的面向对象知识，能够读懂基本的基于 Java SE 1.4 的代码。

下面提到的读者可以从本书中获得所需要的知识。

使用 Hibernate 开发数据库应用程序的程序员阅读本书，可以掌握所需的 API 使用方法。

希望了解 Java SE 5 的新特性的程序员阅读本书，可以在最短的时间里将自己的 Java 知识升级到 Java 5。

系统构架师可以通过阅读本书知道如何在系统中正确地使用 Hibernate，以及配置、启动、调整和封装 Hibernate。

希望了解 Ant 的读者阅读本书，可以了解如何使用 Ant 管理和构建 Java 应用程序。

希望了解 Junit 4 的新特性的读者阅读本书，可以掌握 Junit 4 的全新构架和使用方法。

项目经理或者是负责协调管理整个项目的开发人员阅读本书，可以掌握如何构建一个 Hibernate 项目的开发环境，如何使用开源项目来完成项目管理工作，从而提高项目开发效率。

希望了解 ORM 原理以及 Hibernate 实现 ORM 功能的内部机制的读者阅读本书，可以从构架的角度理解 Hibernate 的工作原理。

使用 EJB2，Hibernate2 进行项目开发的公司，如果希望将系统升级到 Hibernate3；或者希望直接使用 Hibernate3 进行项目开发的公司阅读本书，可以了解 Hibernate3 所能够提供的各种功能，知道如何正确地使用 Hibernate，从而避免在项目开始之前做出错误的抉择。

本书代码

本书中的代码以两种形式出现。一种是完整的源代码，例如本书创建的实体类；或者是相对完整的类源代码，例如节选 Hibernate 类的源代码，这类源代码被放置在一个背景为灰色的方框中并且设置了编号。另一种代码用于测试上述代码或者介绍如何使用 Hibernate API，这类代码将被混入文字中并且没有编号。

随书所附的光盘里包含所有在书中涉及到的代码。本书发行之后，作者还会不断地改正代码中可能存在的错误，最新的源代码可以从作者的个人主页下载，地址是 www.jingge.de。

感谢

一本书的面世往往意味者很多人的劳动，本书也不例外。首先我要感谢清华大学出版社，感谢编辑们在我写作过程中给予的种种帮助和宝贵的意见，感谢他们给予我这样一个机会，使本书得以出版。再者我要感谢姚亦敏女士，由于本人在汉堡工作、生活，对于国内的出版事宜实在是鞭长莫及，感谢她为本书的出版劳心劳力，可以说，没有她的努力，本书根本无法出版。最后我要感谢我的父母，感谢我的家人，感谢他们对我写书工作的全力支持。

关于作者

本书作者葛京现居住德国汉堡，毕业于汉堡科技大学，信息工程硕士，SCJP，SCWCD。毕业后一直从事 Java EE 和 Java 桌面应用程序的开发工作，在软件设计开发领域有 6 年以上的实战经验。从 2005 年开始，作者逐渐将精力集中到软件构架，并开始在一家欧洲顶级物流软件公司负责开发核心框架。

目前，本书作者就职于一家跨国物流集团公司，负责该公司物流系统的构架设计和核心框架开发等工作。在业余时间，作者喜欢运动、和朋友一起玩 paintball、看电影和打游戏、并且喜欢阅读大量最新的技术文章。目前作者关注 Java SE 7，JBoss Seam，以及动态语言如 Ruby 和 Groovy。读者可以通过作者的个人主页 www.jingge.de 与他联系。

由于本人水平有限，尽管笔者已经尽了最大的努力，但是书中难免还是存在错漏之处。如果读者朋友发现了什么错误，请来信告诉作者，邮箱地址是 gejing@gmail.com。作者本人也非常欢迎大家来论坛 www.jingge.de/bbs 讨论相关问题。

目 录

CONTENTS

第1章 对象映射

IT（Information Technology，信息技术）这个词在最近十几年来被广泛使用，简单来说，它就是存储、加工、传输信息等管理信息的方法。从某种角度来说，这和其他工农业技术没有本质的区别。例如，从生产果汁饮料到生产汽车、飞机、大炮，根本上就是生产加工、运输、存放到仓库，最终为消费者所使用。在IT世界里，只不过就是把果汁饮料，汽车、飞机、大炮换成了信息，工作的核心就是不断提供更好的方法来更好地管理信息。

管理信息主要涉及到信息的加工、传输和存储，整个管理工作通过使用不同的软硬件来完成。大多数应用软件提供加工信息的功能，从人们天天接触的办公室软件到处理复杂商务逻辑的中间件产品。不断发展完善的程序设计语言，尤其是面向对象语言，为开发这些软件提供了便利。生产和加工电子信息已不再神秘，理论上讲，任何一个普通人都可以做得到，比如写一封email、使用Word编写一份计划书、等等，都属于这个范畴。另外，Internet以及P2P等其他一些网络传输协议的发明和应用，彻底解决了传递信息的困难，从此信息好似孙悟空架上了筋斗云，一个筋斗，就能从纽约飞到北京。再者，随着信息加工和传输技术的不断发展，信息的存储就愈发显得重要。没有被存储的信息不能持续地提供给人们使用，无论是费用高昂的商业信息还是你我他在个人博客上写的三两句话，如果不存储这些信息则如同一条没有源头的小溪无法持续提供流动的水。一个在系统崩溃甚至系统关闭后就失去所有信息的应用软件是毫无实用价值的。

本章将从信息持久化入手，逐步过渡到对象持久化，介绍各种可能的对象持久化方式。然后根据目前工业界的实际情况，将问题讨论的范围缩小到使用关系型数据库作为存储媒介，同时逐一介绍几种主流的对象——数据库持久化框架并比较它们的优缺点。最终，焦点将集中到本书重点——对象关系影射（Object/Relational Mapping，简称ORM），如图1.1所示。

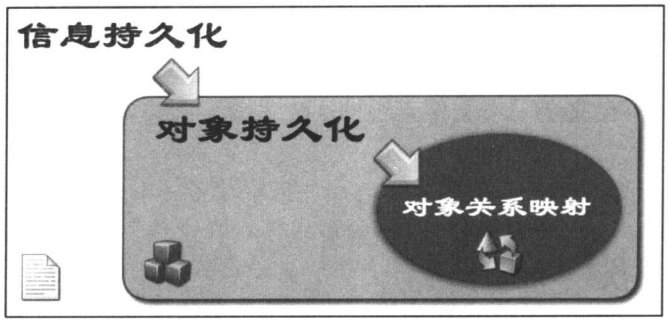

图1.1 关注焦点逐渐集中到对象关系映射

1.1 信息持久化

信息持久化也就是信息的存储，Wekipedia给出了持久化的定义如下。

Persistence

In computer science, persistence refers to the characteristic of data that outlives the execution of the program that created it. Without this capability, data only exists in memory, and will be lost when the memory loses power, such as on computer shutdown.

这段定义的中文翻译如下：在计算机科学领域，持久化指的是数据的一种特性，这种特性使得数据在创建它的程序运行完之后仍然存在。如果没有持久化能力，则数据只能保存在内存中，并且随着内存失去电力支持（例如关闭电脑）而丢失。

从定义中可以看出，持久化就是物理上持久地保存信息，即使是整个软件系统崩溃了，全球电力系统瘫痪了，这些信息也能够完好地保存下来。只要保存信息的硬件没有损坏，人们就可以随时通过软件重新获取这些信息。

从宏观的意义来说，持久化范围很广，将一个故事写在纸上，就是对“故事”这个信息的持久化，这样人们可以相互传阅，随时从这张纸上重新获取这个故事的信息。在计算机科技里，持久化主要指的是将电子信息存放在能持久保存数据的物理媒介上，例如磁带和硬盘上。在硬件方面，在摩尔定律的指引下，硬盘的容量在不断地攀升，其稳定性和容错性也在不断地提高；在软件方面，随着技术的进步，人们可以选择很多不同的技术来使需要的信息持久化。

几种常见的信息持久化方案有以几种。

1.1.1 使用文件系统

使用文件系统来使信息持久化就是将信息以电子文件的形式存储在能持久保存数据的物理媒介上。使用这种方案时，应用程序直接面对底层文件系统，采用指定的编码方式将信息保存在文件中。这可以说是早期的电子持久化技术（比它更早的持久化技术可以追朔到打孔带时代），它甚至在硬盘还没有被发明之前就得以应用，那个时候所有的文件都存储在 5 寸软盘上或者磁带上。

目前，文件系统的持久化方案通常适用于个人需求，在个人电脑上保存的文章、照片、音乐、等等都属于这种持久化方案。小型商业应用程序或者大学和研究单位的研发软件由于信息量很小或者应用不强调信息持久化，可能会采用这种持久化方案，因为该方案实施起来比较简单。而中型和大型的商业应用则不会完全采用文件系统持久化方案，因为这一方案无法满足商业应用的基本要求。原因如下。

- **数据移植困难** 由某一个软件持久化的文件很难被另一个软件所使用。
- **信息查找困难** 由于数据以某种编码的形式保存在文件中，因此通过文件系统在众多文件中查找某个信息十分困难。
- **数据缺乏关联** 信息通常不是独立的，它们之间都或多或少地有关联，这些关联信息很难通过文件系统保存下来。
- **保存修改的数据时缺乏保护措施** 如果在保存的过程中出现软硬件故障，很难将信息还原成保存前的状态。

1.1.2 使用对象序列化

Java 语言的一个重点是强化网络应用、简化网络开发工作。对象序列化是 Java 内建的功能，

其作用是将内存中的一些相互关联的对象实例转化为字节流，这些字节流可以通过网络传输给另外一个 Java 虚拟机，并还原为原来那些相互关联的对象实例。Java RMI 正是利用这个功能来实现远程方法调用，远程方法需要的对象参数和远程方法返回的对象实例都依赖于对象序列化得以传输。除此之外，对象序列化还可以用来临时保存一些对象实例的状态。例如，在 EJB2.x 中，当状态会话 Bean 钝化的时候，EJB 容器将 Bean 的对话状态序列化为字节流并安全地保存在磁盘上，进而释放宝贵的内存空间；当被钝化的 Bean 的触发方法重新被调用时，该会话 Bean 会被激活，这时写入硬盘的字节流被重新读回内存并转化为内存中的 Bean 状态数据。

虽然对象持久化技术可以将一系列对象实例完整地保存在文件中，但是它只适用于保存对象的临时状态，而并不适用于商业级的信息持久化工作。原因如下。

- 序列化后的数据局限于在 Java 应用程序中使用，这样使用其他编程语言来读取这些数据则需要很多额外的编码工作。
- 序列化的数据通常作为一个整体来处理，因此单独修改其中某个对象非常困难。
- 无法直接使用序列化的字节流，也就是说字节流必须经过反序列化还原为对象后，其包含的信息才能得以使用。
- 信息以字节码的形式保存在本件中，无法执行高效率的查询工作。

从上面可以看出，虽然从理论上讲对象序列化技术可以作为持久化技术加以使用，但是由于它的使用效率低下、实施困难、平台依赖性强，所以在商业应用程序里很少会采用对象序列化来保存整个系统的信息。

1.1.3 使用数据库

数据库是目前软件开发中应用最为广泛的信息持久化方案，它可以通过定义不同的数据库模式来将一系列记录系统地保存在表格中。在 Java 世界里，记录就代表着对象实例，保存记录也就意味着保存了 Java 对象的信息。

第一个数据库管理系统诞生于 1960 年，其后，人们不断地开发数据模型（Data model），依次出现了表模型（Table/flat model）、层次模型（Hierarchical model）、网络模型（Network model）、关系模型（Relational model）和对象数据库模型（Object database model）。

1. 使用关系型数据库

采用关系模型的数据库简称为关系型数据库，关系型数据库在近十几年里被整个 IT 界广泛使用，占据着绝大多数的市场。1990 年开始出现的采用对象模型的面向对象数据库将面向对象的概念融入数据库中，力图直接处理和保存对象，面向对象数据库也逐渐开始引起人们的注意。

相信每一个软件开发人员都或多或少地接触过关系型数据库，目前市面上流行的数据库软件，从小到不足 700KB 的开源 HSQL 到价值不菲 Oracle，都是关系型数据库管理系统（RDBMS）。即使是更加贴近面向对象概念的面向对象数据库花了长达 10 年的时间也无法撼动它的地位。

（1）关系型数据库有如下三个优势

① 简洁就是美，关系型数据库的成功就表现为其简洁的特点：在关系型数据库中只有一种类型的数据结构——表格，所有的数据都保存在表格中。例如表 1.1 展示了保存 Book 信息的数据库表格 BOOK。

表 1.1 数据库表 BOOK

OID	NAME	PUBLISH	DESCRIPTION
1	Hibernate 3	2007.04.08	Hibernate3 开发
2	JPA	2007.08.08	JPA 开发
3	EJB3	2007.09.09	EJB3 开发

表格中的每一列只能接受一种类型的数据，例如 OID 列只能保存类型为 Integer 的数据，PUBLISH 列只能接受类型为 Date 的数据。表格中的每一行记录都拥有唯一的一个标识符（主键），这个标识符可以由表格的某一列或某几列组成，例如 BOOK 表格中 OID 列含有的值是各个记录的键值，它对每一个记录而言都是唯一的。记录之间的关联并不是采用数据引用，而是通过标识相关联记录的主键来实现的，如表 1.2 所示。

表 1.2 数据库表 AUTHOR

OID	FIRSTNAME	LASTNAME	ISMALE	BOOK_OID
1	Jing	Ge	true	1
2	Anderson	Neo	true	1

表 AUTHOR 保存了编写图书的人，BOOK_OID 列（外键）定义了该表格与 BOOK 表格的关联。从中可以看出，AUTHOR 表中的两位作者共同编写主键值为 1 的图书。

② 关系型数据库除了提供简洁的数据存储方式以外，它还提供一种简洁而且功能强大的数据管理语言：Structured Query Language（SQL）。使用 SQL 提供的一些基本命令，我们可以随心所欲地管理整个数据库。表 1.3 展示了几个主要的命令，其中 DDL 表示数据定义语言（Data Definition Language），DML 表示数据操纵语言（Data Manipulation Language）。

表 1.3 SQL 语言基本命令

命令	说明	语言类型
Create	创建数据库表格	DDL
Alter	修改数据库表格，包括表格名、列名、列数据类型	DDL
Drop	删除数据库表格	DML
Insert	插入数据库表格记录	DML
Update	更新数据库表格记录	DML
Delete	删除数据库表格记录	DML
select	查询数据库表格记录	DML

③ 另外，关系型数据库还提供完善的事务处理机制以确保应用程序正确地处理数据，提供统一的接口如 ODBC，JDBC 以方便应用程序与数据库进行信息交换。

（2）关系型数据库也有美中不足

虽然关系型数据库是现在使用最广泛的信息持久化技术，但是针对目前主流的面向对象编程技术如 Java 而言，它也有自身的不足之处。如图 1.2 所示。

图中 Petri Net 描述一个业务逻辑，经过分析和设计，我们可以通过定义一系列相互关联的对象来表现业务逻辑内部信息的关系。但是，当我们需要保存这些对象的时候，问题就出现了：对象和数据库表格是两种完全不匹配的类型，无论是存储还是读取，二者之间的转换工作都是无法

避免的。这种类型的不匹配叫做对象——关系阻抗失配（Object-ralational impedance mismatch），任何一个采用关系型数据库作为信息持久化解决方案并使用面向对象技术开发的项目都必须面对并且解决这个问题。

图中的业务逻辑使用的是 Petri Net[PERTRINET]。Pentri Net 是一种建模语言，由 Carl Adam Petri 于 1962 年在他的博士论文中首次提出并经后人不断完善而成。它可以实现精确的数学定义，是一种适合于系统描述和分析的数学模型，主要用来描述分布式系统和商务流程。Petri Net 在工业上应用比较广泛，特别是在非计算机领域内使用的广泛程度要高于 UML。

2. 使用面向对象数据库

使用关系型数据库开发 Java 应用程序时，我们始终必须面对的一个问题就是将对象保存在表格里，换句话说，对象与表格之间始终存在着转换工作。对此 Esther Dyson 作了一个很形象的比喻："利用表格存储对象，就像将汽车开回家，然后拆成零件放进车库里，早晨必须再把汽车装配起来。但是人们不禁要问，这是不是泊车的最有效的方法呢？"

随着面向对象概念逐渐被人们接受、逐渐普及，数据库技术也开始朝这个方向发展了。1990 年出现的第一个面向对象的商业级数据库曾经一度引起轰动，当时人们纷纷预测，在不久的将来，就像面向对象语言全面代替结构化语言一样，面向对象数据库也将会全面代替关系型数据库。

面向对象数据库主要就是把数据库与面向对象语言这两者的功能结合起来。对于应用程序，面向对象数据库直接提供对象实例，这些实例就如同程序开发语言自己创建的一样。程序员在开发软件时，几乎感觉不到数据库的存在，只需要调用几个简单的方法，就可以得到需要的对象实例，如图 1.3 所示。

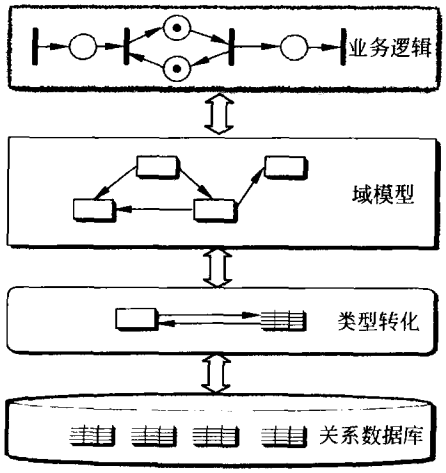

图 1.2 保存对象到关系型数据库

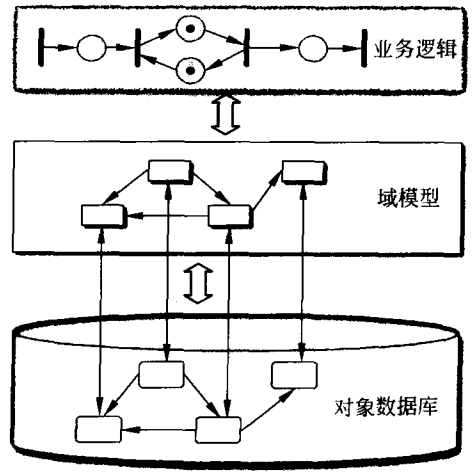

图 1.3 面向对象数据库

从功能上而言，面向对象数据库提供透明的数据持久层、对象关联查询、与面向对象语言无缝连接、对象缓存、等等非常诱人的技术。当然，使用面向对象数据库，我们无需再面对烦人的

阻抗失配问题，车子开回家，不需要拆开，直接开进车库就行了。

可是经过十多年的发展，面向对象数据库并没有在市场上站稳脚跟，更不要说撼动关系型数据库的统治地位了。强如 IBM，Oracle 这样的大厂商也迟迟不进入这一领域。这里的原因有很多，对此 Wikipedia 给出了如下解释[1]。

- 只有针对特殊的查询路线，面向对象数据库才能显示出优势。对于普通的数据查询，在面向对象数据库内要比在关系型数据库内难于实现而且速度慢。
- 面向对象数据库缺乏大量的工具支持，例如报表工具，OLAP 工具等。
- 面向对象数据库模型不是一个数学（mathematical）模型，这也导致了它在查询功能上的不足。

虽然面向对象数据库目前仍存在很多技术上的缺陷，但是它的大方向还是正确的，希望在不久的将来研究人员能够不断攻克技术难关，使面向对象数据库不断发展并最终全面代替关系型数据库。

1.2 对象持久化

程序开发语言和数据库模型可以说是目前软件开发领域中最为重要的两个技术，程序开发语言负责信息的加工和传输，而数据库模型则负责信息的存储和查询。在过去的十几年里，程序设计语言逐渐变得更加高级、更加人性化，其方向主要是将机器代码逐渐抽象化，尽量接近人类的思维方式。面向对象语言如 Java，C#，更是形象地把人类社会的“对象”概念引入枯燥的代码世界，从此开发人员眼里见到的是对象，嘴里谈论的也是对象，头脑里思考的仍旧是对象。人们开始像管理现实生活中的普通物品那样来管理信息，这在程序设计层面上无疑是一次质的飞跃。

本书讨论问题的范围限制在使用 Java 语言进行软件开发工作。在 Java 应用程序中，信息只是以一种形态存在，即对象。如同河里的水流由无数大小不等的水滴汇聚而成一样，信息流是由无数大小不等的对象实例相互交流而成。信息的持久化将会以对象的持久化这一形式来实现。从这里开始，我们讨论问题的焦点将从信息持久化转移到对象持久化。结合上一节持久化的定义，我们可以为本书中所提及的对象持久化下如下定义。

对象持久化

所谓对象持久化，就是将封装在对象中的信息保存在硬盘、磁带等这类物理媒介上。信息能够在程序中断甚至断电的情况下长期存在，并能够通过适当的方式重新还原为信息等同、结构相似的对象。

1.1.2 节中涉及到的对象序列化就是一种对象持久化技术，但它是基于文件系统的。由于数据库已成为工业上数据存储的标准，所以本节讨论重点将是如何将对象保存在数据库中。

相对于程序开发语言从结构化到面向对象的顺利进化而言，数据库技术却没能如愿地从关系型进化到面向对象。程序开发语言与数据库模型的发展如图 1.4 所示。

坐标的 Y 轴表示程序设计语言的发展方向，X 轴表示数据库模型的发展方向。理想的情况当

1 这里是笔者的总结翻译，英文原文见[ODBMS]。