ORACLE®

Oracle 技术系列丛书

Oracle 8

UML对象建模设计

ORACLE8 Design
Using UML Object Modeling

Oracle Press 授权出版

(美) Paul Dorsey
Joseph R. Hudicka 著

孟小峰 丁治明 刘 爽 等译

机械工业出版社
China Machine Press

OSBORNE

Oracle技术系列丛书

Oracle 8 UML对象建模设计

(美) Paul Dorsey
Joseph R. Hudicka 著

孟小峰 丁治明 刘 爽 等译

机械工业出版社
China Machine Press

对象技术的出现早已成为软件开发历程的一个重要里程碑。本书深入而全面地向广大用户介绍了如何利用面向对象方法进行数据库模型的设计，包括逻辑模型和物理数据模型。本书涵盖了为实际设计和构造数据库所需的所有必要论题，如域的设置、命名约定、非规范化操作及逻辑结构可被物理实施的不同方法等，并通过传统ERD和称为UML的新的面向对象标准展示了大量实例。本书以关系数据库为基础，不仅包括建模技术，而且通过所添加的丰富的UML语言进一步深化了面向对象和类的概念。本书当之无愧是Oracle数据库领域有关建模设计的主要参考指南。

本书不仅适用于在关系数据库领域有所研究，且对面向对象方法有所了解的数据库模型设计人员、开发人员，同时也是对Oracle数据库有兴趣的广大读者的重要参考读物。

本书版权登记号：图字：01-1999-3464

图书在版编目(CIP)数据

Oracle 8 UML对象建模设计 / (美) 道尔斯 (Dorsey, P.)，(美)哈迪卡(Hudicka, J. R.)著；孟小峰等译. -北京：机械工业出版社，2000.4
(Oracle技术系列丛书)
书名原文：Oracle8 Design Using UML Object Modeling
ISBN 7-111-07934-5

Ⅰ.O… Ⅱ.①道… ②哈… ③孟… Ⅲ.关系数据库-数据库管理系统，Oracle 8-建立模型-设计 Ⅳ.TP311.132.3

中国版本图书馆CIP数据核字（2000）第15345号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码100037）
责任编辑：马 珂
北京昌平第二印刷厂印刷 · 新华书店北京发行所发行
2000年4月第1版第1次印刷
787mm × 1092mm 1/16 · 17.25印张
印数：0 001~6 000册
定价：39.00元

贺　辞

祝贺中文版ORACLE 8系列丛书的出版。希望她能为中国的广大ORACLE用户和对数据库技术感兴趣的读者提供最先进的ORACLE技术知识。

ORACLE软件系统有限公司希望通过中文版ORACLE 8系列丛书的出版，更好地建立起与广大用户和数据库工作者之间技术交流的桥梁。

Oracle中国有限公司总经理

李文谦

1998.6

序

数据库技术的发展，使它已经成为现代信息技术的重要组成部分，成为现代计算机信息系统和计算机应用系统的基础和核心。可以说，如果没有数据库技术的发展，没有优秀的数据库产品的推出和应用，社会信息化的进程将是难以实现的。因此，在衡量一个国家信息化的程度时，其数据库的建设规模、数据库信息量的大小和使用程度也就成为重要的标志之一。

我国引进数据库技术始于70年代末，从微型计算机上运行的数据库到当前的大型数据库系统的引入和应用，已经有20多年的历史。20多年来，虽然在微型计算机数据库知识的普及和应用上取得了很大的进展和成绩，在大型数据库系统的开发和应用上也取得了进步，甚至还有了国产化的数据库软件，但如果从对数据库系统的应用效果和对数据库技术的掌握上来比较，则与发达国家之间仍然存在较大的差距，特别是在大型数据库系统的开发、建设和应用水平方面差距更大。这种差距主要表现在两个方面：第一是数据库的数量及所收集信息的数量和质量与发达国家相比存在很大的差距；第二是对数据库技术和产品的掌握和应用上更有待于提高和加强，应用人才急需培养，经验有待积累和总结。前者的改善应依靠于对数据库应用基础工作的加强，如重视基础数据的收集和整理，即重视数据工程的建设，并制定相应的数据政策；而后者的改善则会更多地依赖于人们对数据库技术和对数据库产品的掌握。为此，我们必须首先占有充足的资料并加以消化。如果数据库厂家和出版机构能提供较完整的、质量较高的技术资料和书籍，并为较多的数据库技术人员和应用人员所掌握，无疑将会促进国内数据库技术人才的成长并推动数据库应用水平的提高。

喜闻机械工业出版社华章公司与ORACLE出版公司合作，为配合ORACLE 8在中国的发行，由机械工业出版社买断了ORACLE出版公司出版的ORACLE 8系列丛书的中文简体字版的出版权，并组织国内从事ORACLE应用开发的科技人员和教学人员进行翻译出版，还邀请国内数据库专家对译稿进行了审定，以保证丛书在技术

上的权威性。无疑，这对大型数据库系统特别是ORACLE数据库系统的开发和应用将起到很好的推动作用。

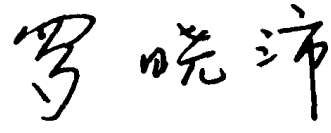
众所周知，ORACLE公司推出的ORACLE 8是一种面向网络计算的数据库(the database for Network Computing)并支持对象关系模型的数据库产品。该系列丛书全面地介绍了ORACLE 8的功能和技术，具体书目见封底。

该丛书内容丰富，涵盖了大型数据库应用开发中的全部技术内容，有的资料，如《Oracle 8数据仓库分析、构建实用指南》、《新版Oracle 8故障解决手册》等都是在国内首次面世，很值得数据库技术工作者参考和阅读。

无疑，这套丛书应该有广泛的读者，它可供大型数据库系统，特别是ORACLE系统应用的系统管理员、应用程序员、系统分析员和设计人员以及广大用户学习和参考，也可做为ORACLE系统培训的教材和高等学校本科和研究生的学习参考书，同时也是掌握大型数据库系统理论和实践的好材料。

应感谢机械工业出版社的领导和组织翻译丛书的先生们和女士们，大家的辛勤劳动将为我国信息化事业的发展增加动力。ORACLE中国有限公司对丛书的出版给予极大的关注与支持，李文谦总经理并为丛书的出版题词。

预祝丛书的出版和发行获得成功，并得到读者的欢迎。



中国科学院研究生院　教授
ORACLE大学(中国)　校长

译 者 序

当今面向对象技术在计算机领域得到了广泛应用，但在计算机最活跃的领域之一——数据库的研究与应用方面却并非如此：关系数据库产品走向成熟，小型数据库百花争艳，大型数据库分割天下。绝大部分数据库技术的新进展依然是在关系数据库方向的。但在同时，一个不容忽视的趋势就是几乎所有的关系数据库都扩展了它们的对象特征。从这一点上我们不难看出数据库的未来发展方向——面向对象的数据库。因此，我们也要寻找新的数据库设计技术。对象建模技术在过去10年里得到了良好发展，实际上也有了一个标准的描述：统一建模语言(UML)。利用面向对象方法进行数据库模型设计也必然会成为未来发展的趋势。为了尽快把这一技术介绍给我国的应用开发人员，我们翻译了这本著作。

本书由从事数据库研究的有经验的人士所撰写，他们在近几年里使用了UML，并把他们的数据库开发经验带入了对象世界，将对象建模的概念带入了数据库世界。可以这样说，本书是对象和数据建模之间的桥梁，是进入数据库系统全新世界的关键，也是进入对象系统全新世界的关键。

参加本书翻译的人员有：孟小峰、丁治明、刘爽、张文亮、王小凤、毕经平、杨琪、郭瑜、曹巍、谷云洪。本书的出版是集体劳动的结晶，由于时间仓促，且译者经验和水平有限，书中难免有不妥之处，恳请读者批评指正！

译 者

2000年1月20日

福 勒 序

在最近的10年里，对象技术的出现已成为软件开发历程的一个重要里程碑。对象技术原来只是一个时髦的词语，现在已成为许多新技术的基础，如：软件组件技术、GUI框架和语言设计。对象在数据库中的应用还不是很广泛。众多对象数据库公司原希望在10年前征服世界，可现在仍只在少数领域中使用对象技术，大部分新数据库仍然是以关系为基础的。

我认为这种状况在今后的几年内将会有相当大的改观。

我确实认为对象数据库会有所进步，但它们在整个世界的数据中的比重仍然很小。关系数据库将会有很大变化，因为它们将扩充自己的数据库，使之可以支持更多面向对象的特征。这个过程已经开始。所有的关系数据库设计者都在夸口自己的数据库已具有一些对象的特点，虽然他们也只是像我一样处在对对象思想认识的初级阶段。我希望这种步伐能在今后的几年内明显加快，对象的能力也迅速扩大。

这就向数据库设计者们提出了一个挑战。当前数据库设计的目标是没有对象的数据库。虽然数据库的模型制作者们已经在努力构造可实施的独立模型，但事实是实施的能力只影响了事务模型。为了利用这些新特征，数据库设计者们需要从对象建模世界里汲取一些思想。最近10年里，对象建模技术已开发得卓有成效了，且确实已有了一个标准：即对象模型的统一建模语言（UML）。

希望这不会造成从数据模型到对象模型的盲从。正确地说，大多数对象模型制作者趋于忽略数据库开发者所需的建模类型。由于对象模型制作者忽略了数据模型制作者，我们已得到了不少教训。此外，大胆地创造新对象模型还是很不够的，我们需要知道如何将它们用到数据库中——是现在的数据库，可能和它们将来的内容会大不相同。

这就是本书的重点所在。它由从事数据库研究的有经验的人士所写，这些人在近几年里积极使用了UML。正因如此，Paul将他的数据库开发经验带入了对象世界，同时也将对象建模的概念带入了数据库世界。也正因为如此，Paul能够传达数据模型制作者应该接纳的重要思想，以便使用这种新的对象关系技术。数据库方面的经验使Paul认识到数据模型制作者不应忘记的原来是对象模型制作者从不知道的。他也能够解释在现行数据库中如何实施对象模型。本书是对象和数据建模之间的桥梁，是进入数据库系统全新世界的关键，也是进入对象系统全新世界的关键。

马丁·福勒

马丁·福勒是一个独立的软件顾问，你可在如下地址中找到他的网页：
http://ourworld.compuserve.com/homepages/Martin_Fowler/

前　言

这是一本关于如何设计和构造数据库的书，它覆盖了逻辑和高层物理设计的概念。本书并没有试图去解释整个系统的开发生命周期。系统开发生命周期方面的信息在《Oracle Designer信息系统开发》一书中找到（已由机械工业出版社出版）。本书只涉及到与逻辑数据模型的构造及逻辑设计的实施有关的系统设计过程的一小部分。

数据库设计分为三个阶段。第一阶段是逻辑模型，获取系统中与数据有关的业务规则，而与表的设计或性能考虑相对无关。第二阶段是物理数据模型，该模型指出了设计物理表的方法。它提供了开发者对于数据库的描述。第三阶段由实际的物理实施构成，此阶段我们可以定义索引、存储细节和数据库参数。本书将包括前两阶段（逻辑和物理设计），而不包括物理实施问题。

我们将利用面向对象方法进行数据模型设计，这对熟悉实体关系模型的人来说将会有点儿变化。即使我们仍以获得表和关系结束建模过程，但看过本书之后，对模型化处理方式的考虑也应有所变化。遵循本书描述的方法，你的数据模型相对于传统的数据模型来说，将会有很多优点：

- 它们将要求更少的实体（或面向对象术语中的“类”）。
- 由于它们不仅支持在分析阶段收集的特定的用户需求，而且支持更广泛的一类需求，所以它们会更健壮。
- 随着新需求的出现，模型将比传统模型要求有更少的变化，所以它们会更稳定。

我们将利用传统的ERD（实体关系图）和称为统一建模语言（UML）的新的面向对象标准来展示所有的例子。可以利用传统的ERD符号产生面向对象模型，但如果利用UML就会更容易一些。我们承认向UML和对象-关系数据库的转换可能不会马上发生。对于那些从事关系数据库、使用Oracle Designer的ERD数据模型工具的系统设计者来说，本书是有所帮助的，它有利于过渡到新标准。

这是一本技术性建模书，它很难与我们在市场上看到的任何其他书作比较。本书描述了很多逻辑模型。同时我们也尝试论述一个其他书很少谈到的论题，即物理数据库从头到尾的设计。为了建立一个数据库，我们需要知道如下基础内容：

- 域的设置。
- 命名约定。
- 如何建立表。
- 何时和在多大程度上进行非规范化操作。
- 逻辑结构可被物理实施的不同方法。

总而言之，我们将讨论所有必要的论题来实际设计和构造数据库。

读者对象

本书是为Oracle数据库界所写的。本书并没有明确指明读者需要理解关系理论，但是，具备关系理论的知识是会有所帮助的。我们将在第2章中对基本的关系理论作简单介绍。

作者试图把每一个新概念在UML和传统的ERD间作比较。我们希望这会使那些对实体关系模型技术已经熟悉的人读这本书时感觉更容易一些。

本书是针对逻辑和物理数据库模型制作者而写的（从某种观点来看，他们应该是同一类人）。但是，为了成为数据模型更灵活的用户，即使开发者们自己不设计和构造数据模型，他们也可以从中获取有用的信息。

面向对象建模涉及的其中一个方向是对象结构库的建立。最近，已经有一些关系数据库界的书遵循同样的模型。本书试图提供有关这类模型如何构造的概念。我们将演示一些用来支持这些原型的不同模型技术和结构。但是，这些模型的作用也是有限的。大概永远不会出现这样一种情形：所有的系统仅通过将不同的类属模型组件结合在一起即可构造成功。好的模型技术总处于一个成功实施的系统的中心位置。

合并分析和设计——一种新的方法

传统的数据库开发要求设计者构造两个数据模型。一个是用来支持用户需求的逻辑数据模型，它并不涉及物理实施；另一个是物理数据模型，此时逻辑数据模型被转换成物理数据模型并被实施。从传统上说，逻辑和物理数据模型是大不相同的。

按照传统的方法，设计者执行分析并产生一个逻辑关系模型。在分析末尾，这个模型被定型。逻辑数据模型符合大多数与数据相关的业务规则。在逻辑设计的构造过程中并没有过多地考虑物理设计。

设计组然后进行实际的数据库设计。他们构造了一个数据模型，即逻辑数据模型的物理实施。物理数据模型与逻辑数据模型大不相同是有可能的（这样大概更有利于性能），之后物理数据模型被用于创建实际的数据库。

这种方法是具有一定吸引力的。分析可以集中在收集逻辑需求上，而不会因性能考虑搅乱我们的思考。在分析末尾，我们确信已经有了一个可以支持已收集的大多数与数据相关的业务规则的模型。然后这个模型被定型，一个数据库就形成了。这时，就可以进行对这个系统运行的任意修改了。

然而，如此好的理论上的方法并不完全与众多的系统设计工作方法相符合。即使我们说分析阶段已完成，也并不意味着发现用户需求的结束，新的需求总会出现。

界面和报表设计经常不断出现新的需求。显示用户界面和报表有助于加深设计者对整个过程的理解。新的字段会在大多数应用程序中出现，并要求以经常性的、结构性的变化支持新需求。

数据迁移不能使一个系统设计保持不变。从历史系统迁移数据的过程会产生一些新的字段。我们也许会发现遗留系统的一些已有字段实际上拥有多于一个的属性。有时，一个字段在一个环境中拥有一个属性值，而在另一个环境中拥有另一个不同的属性值。一些字段将会把许多属性值连接在一起，例如连接零件号和合同号的这类属性。

在开发过程中，当开发者对系统的理解成熟后模型就会有所变化。分析过程中设计的数据模型与系统结束时的数据模型绝对不会完全相同。实体被组合或分离，新的参考表或新的广义化也在形成。

利用传统方法，必须维持两种模型——逻辑和物理模型。若利用当前可用工具，这两种模型不容易保持同步。而且，在产生一个数据模型时，有关性能的业务规则是否应该在逻辑

模型中体现通常不是很清楚。

逻辑和物理模型的分裂包括以下两个原因：

- 过去，Oracle数据库引擎和硬件非常缓慢，因而出于性能方面的考虑，物理模型需要与逻辑模型大不相同。
- 分析和设计任务频繁地被不同的人执行。从分析者和逻辑数据模型那里收集到的需求被主要分析者汇合。然后，物理数据库的设计就被具有实际经验的人创建出来了。

今天，我们已有了更为有效的数据库引擎。计算机速度越来越快，磁盘容量的成本继续下降。Oracle 8的分区功能使它易于保持逻辑表的大规模而不影响性能。开发工具的高效性可以使一个小的、高技能的小组从头到尾设计一个完整的系统。最后，建模过程自身也有了发展。利用本书所讨论的面向对象技术，创建一个逻辑模型而在物理实施过程中不做大的变动是可能的。

我们正在考虑建模的一个不同方法。我们不打算把逻辑模型的构造看作是物理模型构造单独的一步。在分析过程中，我们将创建一个与物理实施问题无关的分析模型。在分析完成和开发开始时，这个模型会进一步发展，其所代表的不仅是成熟的理解，而且也包括支持实施需求所需要的修改。所以，与其有一个逻辑数据模型和一个物理数据模型，不如有一个始于纯粹逻辑数据模型的数据模型。该数据模型最终能表达所有与系统需求相关的数据，能用来产生物理数据库。

这种方法产生了构造一个系统时既支持逻辑的又支持物理特性的单一数据模型。这种统一方法的便利之处在于只需维护一个模型，这就需要考虑从逻辑到物理数据模型的无缝转换。数据模型仍然应该被定期地更新版本，这样当我们准备创建一个真正的数据库时，我们就会有一个准确代表与数据相关的业务规则的模型，并能产生一个工作数据库。

擅长逻辑设计的数据模型制作者通常认为逻辑数据模型只用于获取业务规则，而与最终数据库并无明显的连接。这种方法既有缺陷，且效率也很低。在数据模型中通常有多种方法描述一组数据需求，既然我们认为这个模型是可被实际实施的较为合适的方法，那就没有理由不去选择它。如果我们能够创造出一个逻辑模型，它只需较少的精力就能转化为好的物理模型，那我们就应该尽全力构造这个模型以便减少总的项目开支。

为了构造一个逻辑数据模型，使之只做小小的改动就可成为一个好的物理数据模型，以下条件是很有必要的：

- 数据模型制作者理解如何实际构造系统，在整个SDLC中担任项目领导或起类似的作用。
- 数据模型制作者同时理解数据建模的逻辑和物理两方面。

我们假定数据模型不需要严格的非规范化，并且逻辑和物理数据模型是相似的。但是，当实际实施时，也可能会发生一些小小的变化。如果你需要执行严格的非规范化，那么，想要拥有一个同时考虑逻辑和物理两方面的统一的数据模型就会变得很困难，因为物理数据模型可能会使逻辑业务规则变得很模糊。但是，这种非规范化并非是必需的，好的物理模型应该十分接近好的逻辑数据模型。

我们提倡一个统一的逻辑和物理模型并不意味着不能存在两者有必要分离的例子。但需要记住的是，如果试图维持两个数据独立的数据模型，那么保持两个版本的数据模型同步将要付出相当大的费用和精力。

本书讨论数据模型以及如何实际实施那些数据模型。我们并不会讨论如何保持逻辑和物理数据模型同步，因为（当我们构造了系统时）只存在一个数据模型。

表面上，这种方法或许会使大多数数据模型制作者望而却步。通过进一步考察，我们能确认至少存在一个物理数据模型与逻辑模型有明显不同的工程，且两者再不会同步。一个统一的数据模型应避免这种分离的发生。

在关系表中产生这种分离模型的主要原因是分析和设计的目标不同。在分析时，我们试图获取业务规则；而在设计时，我们则尽力用最好的方法满足分析时收集到的需求。我们不应该因为物理数据模型的限制阻碍分析过程。

首先，经验表明，将分析和设计数据模型过程分离会产生弱点，其主要问题是分析不会真正结束，在设计阶段经常会出现新的需求。如果我们强迫自己同时维持一个分析模型和一个设计模型，那我们无疑正在做额外的工作。

其次，我们已从经验中得知，如果我们过于非规范化逻辑模型，这些模型相对于将来的修改就会变得很不灵活。最好的物理模型是最接近逻辑模型的那一个。

最后，我们承认完全获取业务规则，并设计出一个实际可实施的数据模型是可能的。当然，会有额外的“属性”因物理实施的原因被加入到模型中，但我们发现这种非规范化的字段（只要它们能容易地被识别）并不减损模型的可读性。有时，其效果恰恰相反。一些非规范化的字段用来存储那些用户认为应该放在第一位的信息。例如，在购货单表中，我们经常加入冗余属性存储购买总额。当然，这些信息可通过汇总详细记录获得，但用户通常希望在主记录中看到总额。

UML在数据模型定义中提供了更多的灵活性。Oracle的Object Database Designer（对象数据库设计器）允许开发可在整个系统开发生命周期服务于我们的详细数据模型，并使该数据模型具有产生DDL的能力。这部分将考察用来为样本类产生数据结构的方法。

本书的写作方法

每一章将提供UML语法和能够用这个语法构造的数据模型，并利用传统的关系语法和Oracle 8对象显示这些模型是如何被实际实施的。我们将针对创建模型和实施它们的最好途径给出我们的观点。虽然对象建模已存在了一段时间，但将其用到关系数据库领域并没有被广泛讨论。大部分数据库系统利用关系模型构造，对象建模是数据库设计的一种相对较新的方法。

本书以关系数据库为基础，不仅包括最新的建模技术，而且添加了丰富的UML语言以便进一步深化面向对象和类的概念。其目的是不必牺牲模型的清晰度或性能，而最终能构造更为有效和灵活的系统。

目　录

第三部分 基本建模

第一部分 基 础

第1章 概 述

本章我们将简单讨论对象-关系模型，并概述数据库行业从60和70年代的展开文件结构到数据库理论最近趋势的发展。我们也将针对统一建模语言（UML）、Oracle环境的概观以及Oracle 8i中可利用的新特征作简单介绍。

1.1 对象-关系数据库的发展

当从事于信息系统技术的人们开始利用数据库工作时，数据库由一系列文件构成。我们主要用CODASYL语言的COBOL扩展来访问这些结构。在处理文件时，CODASYL给了我们利用FIRST_RECORD、NEXT_RECORD和LAST_RECORD这类命令的能力，这是我们第一次利用数据库游标。在处理数据文件方面，CODASYL使我们能够比通过阅读text文件更有效地工作。我们对这些结构使用索引、ISAM和VSAM文件，这类结构建立在链接表基础上且与索引结构分离。

第一次我们有了一个数据库管理系统（DBMS），在数据和程序之间具有了一定程度上的独立性。不幸的是，其中仍旧存在一些问题：程序仍然很大，甚至系统中微小的变化可能也需要数百（如果不是数千）小时的人工。你只需看看现在扰人的2000年问题，就可认识到这种系统的局限性。

关系系统以其巨大的理论优势胜过传统的以CODASYL为基础的系统。开发者能够把文件看作是简单的逻辑展开文件，所有的索引和SQL查询语句分析都可由关系DBMS（RDBMS)来进行。

不幸的是，仍然存在许多问题。查询运行得如此缓慢以致于开发者不得不取代缺省的SQL语法分析算法。80年代，要想获得好的性能是如此困难，以致于一些数据库设计者感觉到很有必要回到非规范化数据模型。

1.1.1 关系数据库

关系模型是一种非常精巧、清楚的模型，它已经为数据库行业提供了近20年的基础。关系理论中较少的概念已使关系数据库成为行业标准。关系数据库厂家已经能够主要从系统逻辑设计上孤立数据库物理实施的复杂性，由此提供了一个简单的应用开发者接口。

在过去的20年里，数据库已经作为一个行业发展成熟起来。许多人已经感觉到使建模环境更加丰富的必要性，这种环境更能适应向类属建模的发展。而且，我们认识到面向对象理论的一些概念能够被引入数据库行业，甚至可以带来更高的效率。

关系数据库的词汇量相对有限。我们通过精心设计展开文件来实现结构的设计，并将一些不同类型的索引放在那些文件的不同字段中。就访问而言，表之间的连接只被逻辑指明，

所以参照指针通过外码执行，表之间确实无显式连接。参照完整性约束仅仅是一些阻止特殊种类的无效数据放在数据模型中的代码段。当一个记录被插入、更新或删除时，可以将触发器加到表中，以便明确地执行一些操作，并且可将程序单元存储到数据库中。最后，可用簇表来改善性能。这些策略仍旧会限制我们以一个相当有限的方式思考。在关系数据库中，每个表事实上是一个独立的结构。

在逻辑实体关系（ER）模型中，有作为其他实体子集的实体。例如，领薪金者雇员是所有雇员的子集。同样，有基于其他实体的实体。例如PO明细，它是基于购货单（Purchase Order）的。但是，在关系数据库范例中表达这种结构的能力是有限的。

看似矛盾的是，面向对象数据库保持了早期数据库理论的一些概念。正如在CODASYL时期，当把对象定位带入关系数据库世界时，人们发现遇到了“老朋友”，如链接表和指针。

重要的是不要忘记最初放弃链接表和指针的理由，要记住在80年代早期关系数据库出现之前数据库是怎样的。目前仍有一些高性能的数据库（至少要到2000年问题将它们消灭为止），它们的运行主要使用COBOL编写的CODASYL、ISAM和VSAM文件。CODASYL数据库的修改通常需要花费几个月的精力。我们早期遇到过一个COBOL项目，其中有一个将一个字段的长度从10个字符改为12个字符的简单需求。这个转换花费了几百个小时的编程。现在，在关系数据库环境中，这样一个转换最多仅需几天时间。如果应用程序是用Oracle Designer（或其他一些集成CASE工具）产生的，这样一个转换即使在一个大系统中也只需执行1~2天就能实现。

在关系数据库之前的时代，支持报表需求是很困难的。如果一个特殊的报表并非为了系统的最初设计规范而计划，那就能够容易地花费几周时间去编写一个新的报表（假定编写这个报表是可能的）。对基础性数据结构的修改是很麻烦的，似乎微小的变化就得需要整个大系统重新设计。虽不能断言所有这些问题已随着实体关系图（ERD）和关系数据库的出现而消失，但状况显然已得到改善。

早期的关系数据库看起来很像展开文件的前身，规范化被认为是一件纯粹满足好奇心的事。直到最后，至少通过第三范式，才发现规范化并不是一个坏想法。由于80年代的严重不规范化，在处理关系结构时遇到了一些处理展开文件时遇到的同样问题。数据结构的修改仍然很难且很昂贵，就像应用程序的修改一样。截止到1990年，大多数数据模型制作者已经认识到，当80年代的那些系统需要修改时，其严重的非规范化会引起许许多多的问题。规范化最终成为“流行时尚”。

90年代中期，一些早期的面向对象的思考开始转向关系数据库。在Oracle世界里，这涉及到通过创建更抽象的结构构造模型，但仍然在关系数据环境中操作。在最近几年里，一些关系数据库已经包含了面向对象的思想。

现在，在大部分系统中看到某种级别的类属建模是很普通的。例如，把组织单位表示为一个简单的递归结构是行业标准，而不是用单独的表表示区域、部门和科室（或代表特殊组织的任一种结构）。

更抽象的模型正变得很平常。在最近的一次会议中，设计数据库调查表的人问如何支持一个有几百列的表，且每一份调查表都有几百个问题。被调查者中的一些人回答说应通过将答案放在一个单独的表中和将调查表的结构作为数据存入数据库来为调查表建立模型。一个无争议的解决办法因此而产生，面向对象的思想进入了数据模型。

最近,面向对象的发展更进了两步。首先，数据库本身已包括新的面向对象的特点。其次，