数据库论战卷土重来

Christopher M. Stone和David Hentchel

摘　要

自"面向对象的数据库系统宣言"和"第三代数据库系统宣言"相继发表之后，当今数据库论战卷土重来。本文作者根据这两个宣言，详细讨论了这两大阵营在理论、观点和技术上的异同，澄清了一些错误看法，并分析了相应的应用领域和产品情况。作者认为论战将以OODB的胜利而告终。

当七十年代伟大的数据库论战偃旗息鼓时，Cood关于关系数据模型的十二条规则战胜了层次和网状模型。然而，在八十年代，一批专门研究面向对象技术的著名学者以著文"面向对象的数据库系统宣言"〔中译文见本刊1990年第三期〕又提出了新的挑战。关系数据库阵营很快响应，著文"第三代数据库系统宣言"〔中译文见本刊1991年第1期〕。从此，数据库设计者和用户的智力投资与购买预算都要担风险。

或许你领略不到专家们在理论上的唇枪舌剑，但你肯定会对论战的结果感兴趣。论战的焦点是，在未来十年中最适合用户需求的数据存贮类型问题。

关系的行动准则

几位数据库名流（Michael Stonebreaker、Larry Rows、Davied Beech、Bruce Lindsay等）所著的"第三代数据库系统宣

词汇表和一组有关问题。计算语言系统研究者建立他的系统来理解这个故事并且给出有关问题的回答，当他建立这个系统后，我就进一步提供这个故事与问题，他能用自然语言表达这些问题或翻译成该系统的合适输入。应事先描述该系统的一些局限，然后我们看一看该系统可以成功地回答什么问题，而系统建立者理解这些局限的程度如何。

第二个问题是如何建立这样一个系统，它能从数据库中获得信息并能用自然语言与用户进行交互。再事先给定词汇表，系统研究者调整系统去适应，然后测试系统能否与数据库联接并回答问题。何如，对一个人口普查数据库能否写一个程序给用户确定有关的人口分布信息。

我想这两个问题都是相当困难的，什么时候某个研究小组能很好解决它们尚不得知，或许，这是AAAI奖或其它什么奖的好题目。

然而，这样的挑战性问题并不能取代衡量自然语言理解研究的准则。

7. 图灵测验曾是一个AI的挑战性问题，但不是一个AI研究的科学准则。

适当修改的图灵测验只是确信AI达到人的智能程度的一个充分条件。然而，我们需要朴素点的主张，说明一个特殊的智能机制是解决某一类问题的充分条件或必要条件。

图灵测验即使作为一个充分条件也需要进行修正。模拟人的能力必须经受一个知道程序原理的人的挑战。此外，我们正处在一种让人们认为是魔术师的局面，我们不能忽视这种危险。何况通过对话就想出某种技巧即是机器（AI）是不合适的。

〔**林作铨**译自《The AI Magazine》5(1984) pp.282—285.〕

言”一文，试图再现不亚于信息革命的数据管理史。这些数据库界的前辈们划定：七十年代是第一代数据库（层次数据库）的时代；八十年代是第二代数据库（关系数据库）的时代；九十年代是第三代数据库的时代——虽然它还没有名称，但肯定是八十年代关系模型的一种扩充。

该宣言所描述的第三代数据库有三条基本原则：

1．它必须适合更为广泛的数据类型，例如：映象、多介质文挡、影象及其它“对象”。它还支持每秒处理100个事务的能力，以及支持用于数据完整性和商业处理的规则管理。

2．它必须支持第二代数据库所有“好的”特征，例如非过程化存取和数据独立性。宣言声明基于C++的对象数据库将不得不支持SQL，因而降低了性能（对关系级而言）。

3．它必须能和分布式DBMS、C语言、Unix命令、软件工程工具等软件通讯并能多操作并行（inter-operable）。它不能过份严格限于某种特定语言。

也许你已经注意到，这些关系迷尽量回避“OO”这个词。因此，阵线就一目了然了，即都自称是你的好伙伴的关系同盟对面向对象同盟，而他们都希望挣你的钱。

根据第三代数据库系统宣言，所有数据管理方面的进展都被纳入上述三条原则中，并且关系模型将在市场上占优势。但这并不能说那些正为其产品寻找可观市场的对象数据库人员就是未正名份的“精灵”。

细看来，这不是一场真正的战争，而更象是一场酒吧斗殴，分不清谁胜谁负。为了了解这场战斗，了解两种数据模型的优缺点，首先必须纵观这个领域的历史背景和事实。

数据库的差异

面向对象数据库和关系数据库有很多明显的差异。实质上，面向对象的数据库采用导航计算方式，众多关系界人士对此十分挑剔，他们认为这种方式源于层次数据库的“back-to-the-future”技术。关系数据库有数学理论基础，而大多数面向对象数据库却并非如此。

现有的关系数据库系统从未真正允许使用一个设计结构和视图的嵌套。因此，对面向对象数据库（OODBS）的批评意见集中在“嵌套”对象的导航方面。问题是怎样详细考察复杂结构（对象）？导航的优点是：通过对象（即现实世界的模型）而不是表、元组、记录进行设计显得更容易、也更自然，特别是对于大型复杂的应用（如航天、CAD/CAE等）尤其如此。

有关OODB的事实

事实上，OODB很象网状或层次数据库。它带有嵌套的对象结构，而层次数据库带有人工导航的记录结构，它们都有指针的概念。

在一个OODB中，对象标识符是决不能被复用的逻辑指针，通常对象标识符标识一个由系统生成的对象。它们检验对象和系统向其传递消息时所属的类的存在。这使得OODB将方法（method）加到实施任意完整性规则的对象中去。

层次数据库仅对系结构和索引保证完整性；关系数据库仅在索引中实施，但开发者已经公布了他们正在研究更一般的完整性规则。面向对象数据库和层次、关系数据库的最显而易见的差别是：OODBS引入了一套全新的概念，包括继承性、类、方法和消息。

OODB可在数据库自身中包含代码（方法）。这种关于应用的新认识提供了优化查询处理、控制事务并发执行的能力。因此，OODB是有活力的，而关系数据库已经过时。

完成工作为目的

对于任何系统的实现，性能总是一个主要目标。尤其是对于非常复杂和有众多交叉联系的信息管理，OODB要比关系数据库性

能好。例如，OODB能够根据预期要做的事情在存贮器中贮存某些对象。事实上，数据越复杂，使用OODB所带来的性能利益就越多。在OODB中，用于复杂嵌套结构的聚集和贮存技术，能够使系统性能高于关系数据库二个数量级。

OODB的两个主要商业应用是"文挡(document)"数据库和分布式数据库。前者允许丰富的数据模型构造、推理和存贮复杂对象（如声音、影象、向量、图象）。另外，它还支持也许是几天这样漫长的数据库交互作用以及数据的多版本。文挡数据库应该独立于某个特殊的程序设计语言，以允许不同应用在对象级共享数据。

OODB应用的第二个有作为的方面是委托人/服务员应用，其中，集中式对象服务员为许多委托人管理永久数据库。因此，这可以扩充到一个全分布式OODB，对象可以分布到不同的物理节点而对用户完全透明。

关系的差别

关系数据库和OODB是不会混淆的。规范化关系模型基于相当完备的数学理论。关系数据库在运行时用基于值的方法从存于表中的数据集导出一个虚拟结构。数据库采用从多个表选择数据并将其调到单个表中的方法构造数据视图。(OODB是从一个对象到另一个对象遍历数据的)。

关系数据库的内部数据类型既简单又有限（如整型、字符串等等），用于处理这些数据类型的内部操作也不多。在关系数据库中，也可以建立复杂的数据类型，但必须是基于线性的。例如，将多个域结合成一个记录，并且对这些新的复杂类型的操作再次限制在那些定义过的基本类型上（与此相反，在OODB中，有任意数据类型和带有继承性的子类)。

对象模型支持浏览共用数据元素的对象类库，它允许复用而不能重建。OODB中，对象的存在负有多种使命，它们是持久性的。如果删除一个存在于数据库中的对象，余下的对象或许能导出它，但可能出错。因此，数据完整性就有问题，并且产生了不一致的版本。

在关系数据库中，复杂对象必须分解，并存放在不同的表中。完成这一顺序过程使用了基于前一个查询结果的查询。关系数据库不能理解全局请求，从而不能优化多重请求。OODB能够发送包括多重事务的单条消息（请求)。

在关系数据库领域，既成事实的查询语言SQL正迫使OODB与其相符（至少在接口级)，并且，关系语言接口SQL的操作是基于以未知数据的结构出发的谓词查找思想的。OODB是基于已知结构的指针式导航模型。许多OODB的拥护者呼吁扩充SQL语言，使其能够查询对象和表。这是对SQL的一个自然演变（假设它被正确实现)。

对象SQL或OSQL正被Data General, Hewlett Packard, Object Design所研究。改进SQL适应对象扩充可能要提供一个高级接口，这涉及对象的原始概念并且要将它与一个含单变量的函数集结合起来。这可能是强有力的接口，并且是从关系世界向面向对象的"真实世界"过渡的平坦之路。

揭露直相

关系与面向对象之争的结果之一是，双方都过度吹捧各自的领域和技术，应该相信谁？我们认为某种程度的"故弄玄虚"在所难免，但我们需要弄清以下几个误解。

·误解：关系数据库是非导航的

如果你曾经试图用COBOL实现SQL或为带有嵌套更新的复杂连接定义一个游标(cursor)，就会知道这是一种误解。对"游标"的一个良定义是对非导航语言中的导航问题进行编程。

·误解：对象系统将人们从强类型化中解救出来

尽管对象系统成功地隐藏了数据表示细节，但在查询或其它操作中仍然需要知道数据/对象的类型。在语言环境中，强类型化

和数据隐藏保护了不同库中的方法不因它们使用数据而被破坏。在数据库世界中，方法的存取意味着新方法和新域在老的应用中不需改变。强类型化是系统语言的一个特征。例如，由C++扩展的OODB之所以受到"青睐"，是因为C++的强类型化。

·误解：关系数据库是基于值而不是基于指针的

何种类型的值被放到SSN-TIE-BREAKER或PART-REC-SEGMENT的域中？在一个索引或甚至程序存取ROW-ID时，它来自何种有意义的应用域？事实上，关系数据库的卖主正暗地里将基于指针的算法渗入其产品中了。

·误解：C++是对象数据管理的标准

我们已经避免了造成这样的状况：尽管SQL是如此成功的标准，竟有几个可供选择的标准SQL，但是我们不能坐等C++变成OODB的既成事实标准。的确，它是既成事实的，并且我们还称赞AT&T公司推行语言的标准化，但它是数据库的标准吗？我们恐怕不能同意。作为模式定义语言，C++提供了很好的服务，但是它没有为应用程序员提供如何从OODB中存取数据的思路。所幸的是，C++是可扩充的。OODB阵营已经将其扩充到管理持久性方面。严峻的是OODB卖主极力赞同以这一界面为标准。

·误解：关系数据库支持联机事务处理要比OODB好

第三代数据库每秒100个事务（事务处理委员会定义的典型指标）的目标得到了称赞，也取得了成就，但是这对关系数据库并没有做许多实事。多年来这些系统的设计者在花力气达到这种性能和相当于层次数据库的生产能力，而且毫不怀疑，它也花了OODB设计者的一些时间。关键是存贮效率和锁机制与基础数据模型的关系不如特殊数据库产品设计的关系重大。不存在捷径，如果性能对你是至关重要的，就用自己的标准。

·误解：OODB具有无限的灵活性

从对象范例得到灵活性的巨大改进是众所周知的。但是，与此同时，这一般更需要依靠对象语言而不是一个OODB。对于良好的设计，面向对象意味着一帆风顺；而对于有漏洞的设计，有时会触礁沉没。"模型化现实世界"的一部分代价是：如果得出的现实有误，那将肯定是一场灾难。当需要重构时，对象范例给出控制指针以管理依赖。但它不太保证容易实现"模式迁移"。灵活性可能会害了你。

现实性检验

我们都熟悉那些关系数据库用得很出色的商业应用（例如：记帐和报表）。那么，哪些方面OODB会更好呢？

对象数据库主要不是商业产品，而是由像Mentor Graghics这样的公司嵌入CAD产品中的专有系统。这些卖主在R&D上花费多年时间和成百万美元，最后得出关系数据库不能控制他们所需要的复杂性和吞吐量。由于他们没有其它选择，就只好建立他们自己的数据库。事实上，这种内部市场是OODB领域的成员（公司）轻率起步的主要因素。在过去两年，国际标准化组织（ISO）下属的设计数据的数据结构的主要标准化机构——产品数据交换说明（PDES）委员会已经坚定地改变了面向对象的说明方向（无论是倾向OODB还是基于关系数据库的对象都将留在以后再见分晓）。

因为CAD更接近于一个模拟问题而与数据管理距离较大，所以我们能够指望数据库支持强有力的语义模型。还因为数据项间的联系是非常复杂和经常改变的，所以一个CAD系统中的数据库必须支持向量、表、链表（并且，它们是在设计数据中经常以有形的方式出现）。

CAD中数据存取的模式也和商业数据库不同。典型地，设计一个对象（如一个IC）是由数千个设计单元组成。设计单元或许和外部项（如物理布局、部件说明、分析数据）有关。当设计者查看这个IC时，所有嵌

套的单元和许多有关单元要以簇的形式快速地从数据库中调出。经验表明，这种成堆存取和复杂的内外结构组合不能在关系数据库中有效地实现。

另外一个需要复杂结构的应用是地理信息系统（GIS），但此处的数据库需求略有不同。至少有一个卖主Wild Leitz (Prime Computer的子公司)，已经用基于关系的对象实现GIS。然而，这一实现涉及了传统关系能力的有意义扩充，包括内部数据过程和可扩充的数据类型。在此，关系库是有吸引力的选择，理由是GIS数据库相当强调复杂的查询而不是结构操作。关系数据库在管理即席查询和内外部数据库的链接等方面的历史功绩能增加其这方面的价值。

范例的局限性

对一个拿到一把新锤子的人来说，每样东西看上去都像钉子，而人们在选择关系数据库或面向对象数据库时，必然会根据自己的工作需要而定。

例如，工程数据如何做为关系来存放？一个典型的造船项目管理三百万不同的部分，每部分有512KB到10MB的设计单元（如几何形状和拓扑结构），加上10到200个属性及10MB的分析数据。每个部分都要保持大约七个不同的版本，这造成了大量交互作用的潜在复杂性。

用于工程数据管理的PDES模型能够表达大约16个对象类、30个对象链类型。作为比较，这种工程数据管理的类型可能需要数百个关系记录类型和数千个连接链。在应用中，每个类型可能不得不分别处理。这对关系数据库的任何现实利益都是不平衡的，因为即席查询和数据结构相对来说用得比较少。

另一方面，设想银行帐户作为纯对象来管理：一个出纳事务会"在付帐后离开"贷方帐户对象。一个寻找加锁帐户的并发ATM事务将生成一个借方对象的新版本。然而，企业的部门经理会用一个子类来为某些帐户特别设定利息算法，那么凭什么假设查帐员会理解这些呢？

在上述的这些极端例子中，只要提供时间和好的开发者，你都能够从"有毛病的"技术得到有效的应用。对于那些不需要最好工具的问题，正是这一点使得他们犹豫不决。你应该相当明智地避开针对未经历过的问题的任何热门技术。

下表展示了关系和面向对象数据库的异同。你的工作不是决定哪个更好，而是决定哪个更适合你的需要。

关系数据库和面向对象数据库的比较*)

关系数据库	面向对象数据库
结构化查询	导航查询
最小化数据依赖	最小化过程依赖
"人思考的实际方法"	"人思考的实际方法"
比层次慢	或许比关系更慢
但无人注意到	但仍无人注意到
短事务，	长事务
优化并发	正获得优化并发
符合第四代语言	符合对象语言
隐式联系	显式联系
没有唯一标识符	唯一对象标识符
能够表示"对象"	能够表示"关系"

*)关系和面向对象卖主如何刻画各自的产品，这为你提供了区别它们的思想。

争论正在结束

OODB与早期的语义数据模型有部分重迭并且提供了一个比关系模型更为丰富的环境。面向对象模型固有的概括和聚集关系已经促使了重新研究那些原本为关系数据库系统开发的结构概念。这些概念包括模式演化、查询、并发控制、存贮结构和索引。由于关系模型的局限性（特别是它不能处理复杂数据类型）OODB正在演化。

许多权威、学者和改革者提出关系模型的扩充，这势必为关系阵营指定了方向。事实是他们中的大部分人（包括"第三代数据库系统宣言"的支持者）正在其产品的前端

函数式、逻辑式和面向对象式程序设计及其多范例合成语言

梅　宏　孙永强（上海交通大学计算机系）

摘　要

本文综述了函数式、逻辑式和面向对象式三种新型程序设计语言的基本特点，简介了这三种程序设计风格相互合成的研究现状，并给出了一些有关的代表性语言。

计算机科学发展的一个重要方向是更完美、更接近地模拟人类智能，使新一代计算机成为智能计算机。当今计算机科学所研究的热点：新的体系结构、新型语言、人工智能、人工神经元网络等都有一个共同的目标——计算机的高度智能化。然而，在计算机体系结构尚未取得突破性进展的今天，人们对智能系统的研究只能是在传统的von Neumann机器上用软件来实现智能模拟。但是，传统的命令型语言及程序设计方法有着较大的缺陷和不足，难以很好地胜任这种模拟工作。因此，新的程序设计方法及其支撑环境——新型智能语言的研究显得尤其重要，吸引了众多研究人员为之努力。

一、函数式程序设计语言

当今研究的新型主流语言主要有三大类：函数式语言、逻辑式语言和面向对象式语言。函数语言以λ-演算为其语义基础，它的基本机制是函数对参数的作用，函数是程序的基本项，程序的编制便是函数的递归构造过程。从数学观点看，函数是从一个域（定义域）到另一个域（值域）的映射，即函数描述了两个域上元素的对应关系。因此，函数语言是一种描述性语言，它只给出需求解问题的定义而不需给出具体的求解过程和细节。求解过程是语言本身经过应用一系列重写规则实现的。λ-演算作为一个重写系统满足合流性（Chucch-Rosser性质），即一个项若有范式，则不同的重写策略将导致相同范式，从而保证程序求解的唯一性。λ-演算作为重写系统是相当简单的，它只有三条重写规则：（1）α**规则**：若y不在X中自由出现，则有$\lambda_x.X \xrightarrow{\alpha} \lambda_y.[y/x]X$。（2）β**规则**：$(\lambda_x.M)N \xrightarrow{\beta} [N/x]M$。（3）η**规则**：若x不在M中自由出现，则$\lambda_x.M_x \xrightarrow{\eta} M$。

加入面向对象的扩充。随着时间的推移，将没有什么能阻止他们转向完全的OODB。技术上领先的公司将发现，在转向OODB方面要比市场领先的公司更困难。这使得目标变得严峻起来。问题是计算机界的人喜欢把事情弄得比实际情况更难。界线不是黑白分明的，而是含混不清的。

一个OSQL和OODB可能使你改变方向，因此你可以开始做真正的数据管理问题：劝人们去管理数据。注视着即将来临的数据库技术变革，就像一个人坐在开始涨潮的海边，你可以跳入水中、或者向岸上跑、或者等待潮水到来。

〔**朱扬勇**　译自《BYTE》OCTOBER，1990，pp233—242　**吴军耀**校〕