计算机科学 1999 Vol. 26 №. 5

数据仓库技术和可复用构件库系统

Data Warehousing Technology and Reusable Component Library System

杨燕燕 梅 宏 陈海文 邵维忠
(北京大学计算机科学技术系 北京 100871)

Abstract Software reuse is considered as a realistic approach to solving the software crisis. Component Based Software Development (CBSD) is an effective way to support software reuse. The retrieval and management of components play a key role on CBSD. Decision support solutions are important to understanding, retrieving and managing components. Data warehouse technologies provides a feasible approach for effective decision support. In this paper, the application of data warehouse technologies in component library is discussed. Data warehouse solutions improved possibility and quality of reuse.

Keywords Software reuse, Data warehouse, Component library, Multidimensional schema, Decision support

1 引言

要支持基于构件复用的软件开发过程,就必须要有支持整个软件生存周期并包含有大量可复用构件的构件库系统,其中,构件的有效管理和查询选取是关键。但随着构件库中构件数目的增加,复用者在查询和选取构件时就会遇到困难,构件库的管理者在对构件进行管理和维护时也会力不从心。构件库系统如何提供好的检索机制使用户能快速地查询到所需构件,这直接影响到复用实践的成功。[9]国内外学术界对此也进行了深入研究,在构件的分类模式等方面取得了较大进展,但对用户检索到多个构件时,如何对多个候选构件进行评价,方便地从中选取最能满足需求、修改最少的构件,目前还没有较好的解决方法。软件评价的标准很多,已有一些比较有效的管理性方法,但在复用环境中这些传统的度量方法还需要修改,在这方面几乎没有什么经验,到目前为止,这个过程还依赖于复用者的经验。因而,如何为用户提供多样化的查询手段,帮助用户正确评估和选取所需的构件,减轻用户的负担,尽量缩短检索构件的时间,为构件库的管理者有效地管理构件提供决策支持就成为一个比较关键的问题。数据仓库技术为解决上述问题提供了一个比较好的解决方案。

数据仓库技术是90年代以来国外逐渐发展起来的新技术,它一经问世就得到了国外商业界、企业界和学术界的高度重视,特别是1995年的DB/Expro'95以后,世界范围内掀起了数据仓库研究和开发的高潮[10,11]。数据仓库技术首先在决策支持系统中得到了成功的应用。

本文探讨了如何把数据仓库技术引入到构件库系统中,为用户和构件库管理员提供切实有效的辅助决策支持,从而提高构件库系统的管理水平和构件复用的水平,为复用的成功提供技术保障。

2 构件库系统研究现状

构件库系统有效地组织和管理大量可复用构件,并提供相应的工具支持开发者在软件开发过程中方便地查询、理解和选取构件,使基于构件复用的软件开发成为现实。其中,构件的易理解性和可信任性是复用成功的一个重要前提。如何帮助用户快速理解、评价并选取所需构件直接关系到复用实践的成功。国内外学术界对此进行了深入研究。

随着构件库系统的发展,当构件数目逐渐增多时,用户在查询和选取构件时将会碰到以下一些问题:

1)在查询过程中,可能有多个满足用户查询条件的构件,如何快速有效地从众多的候选构件中准

确地判断并选取所需的构件，这是一个复杂的决策过程，一般都依赖于用户的复用经验以及对构件的理解和主观评判；

2）通常用户是通过刻面、属性、关键词、关系等表达所需构件的特征以进行构件的查询，然而，访问构件库的用户具有不同的层次，有可能并不熟悉构件的刻面分类模式等，对构件的理解比较困难；

3）用户只能使用"合法的"术语，亦即刻面的术语空间中提供的术语对构件进行检索。随着构件库系统的不断演化，刻面的术语空间相应也会增大，用户要迅速找到所需的术语可能会遇到困难；

4）用户在查询前也许并没有一个明确的目标，只是想通过查询构件库看是否有能够利用和复用的构件，因而如何通过构件的复用历史和其他用户的复用经验为用户提供一定程度的帮助是很必要的；

5）对需求规约、设计、模式、测试计划等文档型知识的构件复用属于间接复用，需要复用者首先进行分析和理解。而且，在大多数情况下，对构件的复用均是白盒复用，亦即要根据构件复用者的反馈，对构件进行适应性修改。如何跟踪软件复用的经验和构件的使用历史，辅助用户选取相应的分析、设计件以及改动最少的构件是很关键的。

此外，构件的后期度量是基于复用者的反馈信息，如何收集用户反馈数据，进行合理的组织；如何对数据进行适当的处理得到度量值，并和前期度量的结果相比较；如何把整个度量结果以易理解的形式显示给用户，让用户有一个直观的映象；如何使用户能动态调整评估模型及度量各个层次的权值等也是亟需解决的问题。

REBOOT（Reuse Based on Object-Oriented Techniques）[2]是国际上比较著名的构件库系统，包括一个存储可复用构件的构件库和一组产生、认证、插入、提取、评价和适配可复用构件的工具。

REBOOT 环境中，针对上述问题，把构件的复用历史和用户复用经验的反馈作为构件评价、构件查询机制改进的重要依据，以及用户理解构件的一个重要方面。在该环境中，对每个构件进行详细描述，包括测试报告、用户满意程度、批评意见、改善构件的修改需求等，并提供工具使用户方便地访问构件的描述信息以帮助其理解构件。此外，对每一个构件设置一个开放论坛，用户必须报告他们的复用经验和复用中碰到的问题，作为构件度量的依据之一，以及辅助其他用户理解构件。

REBOOT 环境中，定义了两个质量与可复用性模型以评估一个构件的质量和可复用性，但评估的因素是通过静态计算构件代码得到。REBOOT 中指出，这些统计仅仅是一种信息来源，另一个重要的来源是其他用户的经验和他们对构件可复用性的主观评估。在进一步的研究中，应使复用者能够从不同的视角提供观测的值，并与静态的统计结合起来评估构件。

目前，REBOOT 中对构件的描述仅针对单个构件，且都是文本类型，当用户检索到一组候选构件时，如何提供量化的比较信息，使用户快速选取最能满足需求的构件，REBOOT 中尚未提供有效的技术支持。如何有效地组织用户反馈信息以改进构件度量正在进一步研究中。

NATO（北大西洋公约组织）[3]制定了一整套软件复用的指导性标准，其中《可复用软件构件库管理指南》指出：构件库系统应对每个推荐入库的可复用构件进行定性和定量的评价；在对构件进行配置管理时，应记录下复用者的复用历史，以此来改进构件库的查询机制，使用户更好地选取构件，并作为对构件进行配置管理和维护的依据，同时这对构件库性能的改进也很有帮助。构件库小组还应追踪以各种方式从构件库中提取过构件的复用者，并坚持用构件库工具追踪对构件的复用。复用者需要提供一个"反馈日期"的协议，表明自己将何时反馈回该构件的使用意见以及复用的经验。NATO 标准指出复用者应能访问候选构件的摘要、质量评价、软件度量等相关信息，从而找出与需求最为匹配的构件。

美国军方发起的 STARS（Software Technology for Adaptable, Reliable Systems）项目于 92 年提出了开放体系结构的可复用资产框架 ALOAF[1]。ALOAF 认为并非所有的软件资产库都使用同一数据模型，即使同一软件资产库的数据模型也会随时间而变；软件资产库用户查找各种软件资产来支持软件系统某个生存周期的活动，可能要访问多个软件资产库以使复用的数量和有效性尽可能大。多个软件资产库构成了多个信息源，理想情况是用户能用一个最喜爱的界面访问所有软件资产库，不必关心它们的位置和特性。STARS 定义软件资产包括软件工作产品、软件子系统、软件构件、专家联系名单、构架、领域分析、设计、文档、个案分析、经验教训、研究成果、以及有创见的软件工程思想与表述等，用户在复用过程中的一些好的经验、教训等均可作为软件资产。

RIG（Reuse library Interoperability Group）[2,4]

认为，软件复用中理解问题被认为是影响复用成功的一个关键问题，而现存的软件资产库使用各自不同的数据模型、分类模式和术语，这种多样性即多个信息源的存在对于支持不同的领域、客户和技术研究是十分有用的，但复用者浏览不熟悉的库或从不熟悉的库获得软件资产时，将花费更多的时间理解不同库中的软件资产，同时也妨碍了库之间软件资产的共享。RIG 的一个技术委员会 TC 开发了一个统一数据模型 UDM(Uniform Data Model)，其中定义了支持可互操作性的库之间交换软件资产所需的信息。UDM 提供了一个可能的解决方案。但 UDM 并不是定义可互操作数据模型的最终结果，还需要 UDM 的一个超集提供比 UDM 更加强大的软件资产交换能力。

综上所述，构件的复用历史和用户复用经验的反馈是非常重要的，如何有效地组织这些信息为用户查询、理解和选取最能满足需求的构件以及管理员有效地管理构件提供辅助决策支持直接关系到复用的成功。此外，构件库系统为了保持各个库中的软件构件与领域相关的特性，它们之间的差异将会继续存在，如何针对多个构件库提供一致的用户界面使用户理解构件并为整个构件库系统的高层管理员有效地管理多个构件库是非常重要的研究课题。目前很多构件库系统采用传统的数据库技术来解决上述问题。然而，为用户和管理员提供辅助决策支持属于分析型处理。在数据仓库技术出现以前，传统的数据库技术进行着从事务处理、批处理，到决策分析等各种类型的数据处理工作，没有划清数据处理的分析型环境与操作型环境之间的界限。随着数据仓库技术的出现，数据处理由原来的以单一数据库为中心的数据环境发展为新的体系化环境，即由操作型环境和分析型环境(数据仓库级，部门级，个人级)构成[12]。

传统数据库技术对分析处理的支持一直不能令人满意，尤其是当 OLTP(联机事务处理)应用与 OLAP(联机分析处理)应用共存于同一个数据库系统中时，这两种类型的处理发生了明显的冲突。人们逐渐认识到事务处理和分析处理具有极不相同的性质，直接使用事务处理环境来支持分析处理是行不通的。事务处理环境不适宜分析处理应用的原因概括起来主要有以下几方面：

(1)**数据的综合问题** 在事务处理系统中积累了大量的细节数据，一般而言，分析型处理并不对这些细节数据进行分析，这主要有两个原因：一是细节数据数量太大，会严重影响分析的效率；二是太多的细节数据不利于分析人员将注意力集中于有用的信息上。因此，在分析前，往往需要对细节数据进行不同程度的综合。而事务处理系统不具备这种综合能力，根据规范化理论，这种综合还往往因为是一种数据冗余而加以限制。

(2)**数据集成问题** 事务处理的目的在于使业务处理自动化，一般只需要与应用有关的当前数据。而决策中经常用到外部数据，这部分数据不是由事务处理系统产生的，而是来自其它外部数据源，例如，权威性刊物发布的统计数据，业界的技术报告，市场比较和分析报告等，这些数据通常都是非结构化数据。尽管每个单独的事务处理应用可能是高效的，能产生丰富的细节数据，但这些数据却不能成为一个统一的整体。数据集成是一项十分繁杂的工作，都交给应用程序完成会大大增加程序员的负担。并且，如果每做一次分析，都要进行一次这样的集成，将会使处理效率极低。

(3)**历史数据问题** 事务处理一般只需要当前数据，在数据库中一般也只存储短期数据，且不同数据的保存期限也不一样，即使有一些历史数据保存下来了，也被束之高阁，未得到充分利用。但对于决策分析而言，历史数据是相当重要的，许多分析方法必须以大量的历史数据为依托。没有对历史数据的详细分析，是难以把握整个发展趋势的。

由此可以看出，分析型处理对数据在空间和时间的广度上都有了更高的要求。而事务处理环境难以满足这些要求。

虽然也可以在传统的数据库技术之上构筑分析型应用，但以上这些问题表明，在事务型环境中并不适于直接构建分析型应用。数据仓库技术本质上是对这些存在问题的回答。建立在事务处理环境上的分析系统无法达到这一要求。要提高分析和决策的效率和有效性，分析型处理及其数据必须与操作型处理及其数据相分离。必须把分析数据从事务处理环境中提取出来，按照分析处理的需要重新组织，建立单独的分析处理环境，数据仓库技术正是为了构建这种新的分析处理环境而出现的一种数据存储和组织技术。

青鸟构件库系统 JBCL(Jade Bird Component Library System)是"基于构件-构架模式的软件复用支持系统"的核心子系统[10,11]，包括构件库、构架库以及相应的库管理工具。JBCL 用于对可复用构件进行描述、管理、存储和检索，以满足基于"构件—构

架"复用的软件开发过程的需要。在JBCL系统中，针对上述问题，尝试采用数据仓库技术，从构件的复用历史和用户复用构件的反馈信息出发，建立了基于多维模式的用户反馈库，为复用者选取和复用构件以及构件库各个层次的管理人员管理和改进构件库系统提供了一定程度的辅助决策支持，通过对反馈信息进行量化处理和合理组织，对构件的后期度量提供了一定的帮助。

与传统的数据库技术相比，采用数据仓库技术有以下几方面的好处：

- 把分析型处理及其数据与操作型处理及其数据相分离，提高了分析和决策的效率和有效性；
- 根据最终用户的观点来组织和提供数据；
- 提供用于决策支持的历史信息；
- 采用实物化视图技术，提供增强的概括和聚集能力；
- 采用多维分析和切片切块(slice and dice)方法，支持drill-up和roll-down分析；
- 提供可视化分析工具；
- 星型模式易于用户定制评估模型；
- 通过提供辅助决策支持，为构件的选取从主观向客观迈进了一步，从定性向定量迈进了一步。

3 数据仓库基本概念及特点

W. H. Inmon给数据仓库作出的定义是："数据仓库是面向主题的、集成的、时变的、非违约的系列用于管理和决策制定的数据集。"它与传统的数据库有较大的不同[14]。

面向主题是数据仓库的重要特征，这是与传统数据库面向应用相对应的。传统的E—R型数据模式能较好地执行联机事务处理(OLTP)，但不太适应决策支持分析，而数据仓库则是为决策管理提供支持信息，根据业务需求从用户的角度基于主题来组织数据并生成相应的数据视图、汇总表等。主题是一个在较高层次将数据归类的标准。基于主题组织的数据被划分为各自独立的领域，每个领域有自己的逻辑内涵，互不交叉。而基于应用的数据只是为处理具体应用而组织在一起的。

数据仓库的第二个特点是其数据都是集成的，数据在从面向应用的操作环境中提取到数据仓库中时，都要经过集成化，首先要统一原始数据中的所有矛盾之处，如字段的同名异义、异名同义等；其次还要将原始数据结构做一个从面向应用到面向主题的大转变，集成性以多种形式表现出来，如一致的命名转换、一致的变量度量、一致的编码结构、一致的数据物理属性等。

数据仓库是随时间变化的，这一点与传统的系统不同，传统操作型环境中包含的是当前数据，即在存取时刻是正确、有效的数据，而数据仓库中的数据都是历史数据；其次，数据仓库内的数据时限要远远长于操作型环境中的数据时限，一般是5～10年，而前者只有60～90天；数据仓库数据的码键都包含时间项，从而标明了该数据的历史。

数据仓库的另一个特点是其非违约性。因为数据仓库只有两种基本操作：装载数据和访问数据。由于这一原因，带来很多有利的结果，就是数据仓库在物理层上可做很多优化工作。在以往的操作型系统中，数据库管理系统和应用要用很多复杂技术以保证后援和恢复、事务和数据集成、探测和修复死锁等。数据仓库反映的是历史数据的内容，而不是处理联机数据，因而，数据经集成进入数据仓库后一般不需要改变。

正如E. F. Codd所指出的[15]，在其提出的E—R理论基础上的关系型数据库是为联机事务处理的需求环境设计的，而要实现联机分析处理(OLAP)则需要有与之相适应的、集成的、快速的、多维的信息构架和查询机制。数据仓库技术正是为满足这一需求而提出的。

4 基于多维数据模式的用户反馈库

JBCL系统中，用户反馈库的数据模式采用数据仓库中通常使用的多维数据模式—星形模式[16,17](Star Schema)。

与ER模式相比，星形模式是一个可预测的、标准的框架，可适应用户行为难以预料的变化。每个维是平等的，都是对等的指向事实表，逻辑设计能独立于预期查询模式进行，用户界面、查询策略以及在星形模式基础上生成的SQL都是一致的[12]。星形模式数据组织方式直观，符合用户的思维方式，结构简单，易于理解和使用，有良好的连接路径，易于修改和增补。

用户反馈库中，对反馈信息进行一定程度的量化处理，主要包括七个维度：时间维Time、制作者维Provider、用户维UserInfo、组装维CompositeInfo、测试维TestInfo、描述维DescripInfo、评价维ValuationInfo，两类反馈事实：细节事实FeedbackFact、聚集事实AggregateFact，其多维模式如图1所示：

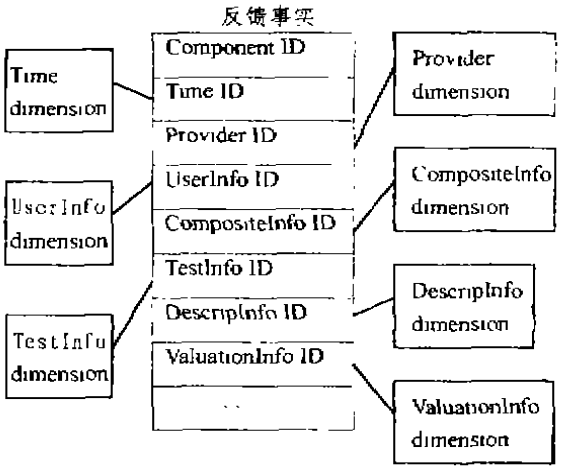

图 1　用户反馈库数据模式

用户反馈库中，通过收集工具捕获用户提交的反馈信息并存入临时库中，这个过程主要支持OLTP，因为用户随时可能提交反馈信息，尤其当构件库连入 INTERNET 以后，世界各地的用户全天 24 小时内都有可能提交反馈信息；通过提取工具、清理工具定时地（如每隔 12 小时一次）对临时库中的数据进行提取和数据清理，合格的反馈信息通过转换、集成工具装载到用户反馈库中；通过访问、分析工具可以查询、分析用户反馈库中的数据。

随着构件库系统的不断发展和网上构件库的不断完善，可能在不同的地理位置存在多个构件库，这些构件库可以是同构或异构的，相应地有各自对应的反馈信息，通过提取工具和转换、集成工具，可以把各个信息源中的数据方便地集成到中央的统一信息库中，用户只需访问该中央信息库即可。而且，由于用户反馈库采用多维数据模式，当需要查询和分析的主题不断增加时，例如构件库的管理者需要分析构件的生产、销售情况，则可通过为新增的主题建立相应的事实表和维表（可以共享原来的很多维表），方便地集成到原来的信息库中，并不改变原来的数据结构，用户反馈库可以逐步演化到满足整个软件企业业务需求的软件企业信息仓库，这也符合数据仓库的开发原则[11 15]，即先从一个主题入手不断地加以完善。

数据仓库实施中的一个难点就是对已有的数据源的集成问题。随着业务需求的增加，在软件企业内部将会有多个构件库系统，形成多个信息源，这是一个亟待研究和解决的问题。在青鸟系统中，由于用户反馈库已建立了一套完整的数据仓库体系，因而可在统一的仓库数据模型的指导下建立企业内部的各类数据库系统、采用自顶向下的设计方法、减少由各个信息源向数据仓库集成的难度。

结束语：软件复用是目前解决软件危机的一条现实可行的途径，对软件复用技术的研究已成为软件工程领域的重要课题之一，基于构件的软件开发方法以其对软件复用的有效支持，逐渐被越来越多的开发组织所采用，为了达到有效的复用，构件必须具有可识别性、易理解性、可信任性，构件库系统作为核心系统，其构件的查询选取和有效管理是关键，国内外学术界对此展开了深入的研究，取得了一定的进展。本文对如何把数据仓库技术引入到构件库系统中进行了探讨，在 JBCL 系统中，尝试采用数据仓库技术，建立了基于多维数据模式的面向分析的用户反馈库。通过对数据仓库技术的应用，为复用者理解和选取构件以及构件库各个层次的管理人员管理和改进构件库系统提供了一定程度的辅助决策支持，提高了复用的可能性和复用的质量，为复用的成功提供了有效的技术保障。

参 考 文 献

1　Mcllroy. Mass produced software components. Software Engineering Concepts and Techniques. In J M Buxton, et al. eds. 1968 NATO Conference on Software Engineering. New York, 1969, 88～98

2　Morel J-M, Faget J. The REBOOT Environment. BULL S. A. Rue Jean JAURES, F-78340 LES-CLAYES-SOUS-BOIS, France

3　Reuse Library Interoperability Group. RIG Uniform Data Model for Reuse Libraries (UDM). [RPS-0002] January 1994

4　Reuse Library Interoperability Group. RIG Basic Interoperability Data Model (BIDM). [RPS-0001]. 1993

5　NATO Communications and Information Systems Agency. NATO Standard for Development of Reusable Software Components, 1991

6　NATO Communications and Information Systems Agency, NATO Standard for Management of a Reusable Software Component Library, 1991

7　NATO Communications and Information Systems Agency, NATO Standard for Software Reuse Procedures, 1991

8　STARS Technical Committee. Asset Library Open Architecture Framework: Version 1.2: [Informal Technology Report, STARS-TC-04041/001/02] 1992

（下转第 30 页）

得到了良好的控制，该公司现已把精力投入到了降低硬件费用上。

图 6 说明了 SS2000 产品生产线的组织结构。

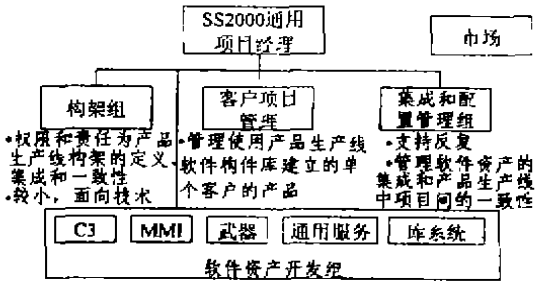

图 6　SS2000 产品生产线的组织结构

图中的构架组负责定义和演化产品生产线构架，并要保证构架的一致性，它通常由几名(10 名以下)具备丰富的软件和系统经验的高级技术工程师组成。客户项目管理组的职责与 ESC 产品生产线中的 SPO 类似，主要作为产品生产线组织与用户之间的中介。软件资产开发组主要负责开发、标识和维护软件资产，在它内部有一个软件资产库系统。集成和配置管理组负责将产品生产线构架和软件资产集成为产品生产线系统，保证各个系统之间的一致性。

小结　使用产品生产线方法的好处是显而易见的，它将以更小的代价和资源，更快地开发出系统。由于使用的是高度可靠、性能经过验证的可复用软件资产，产品生产线系统的质量将大大提高，而且产品生产线方法还开拓了一个广阔的软件资产市场，这将极大地促进软件资产业的发展。

但是，产品生产线方法仍然面临着很大的挑战和障碍，这主要体现在：

• 文化上：产品生产线策略意味着软件组织和管理者对他们产品开发的直接控制减少了，对其它组织的依赖增加了，需要一种思想观念上的转变。

• 战略计划上：产品生产线的规划不仅是对一组相关系统的管理过程，还需要考虑用户的长期需要和现有产品生产线的能力，对未来的发展作出长远规划。

• 需要折衷：产品生产线方法需要用户作出折衷，是"独自开发一个恰好是我所需要的系统"，还是"利用产品生产线开发一个与我的需要非常接近的系统，但可以节省开发的代价和时间"。

• 资源的所有权：软件资产归谁"所有"？在目前的体制下，这的确是一个容易引起纠纷的问题。这就需要整个软件开发和管理组织的重心从当前的程序获取转移到商业化的软件资产开发上来。

• 提拔和奖励：当前的开发方法主要是提拔和奖励那些提交了最终系统的开发人员，采用产品生产线方法后还应该提拔和奖励那些开发软件资产和促进了产品生产线开发中软件资产复用的人员。

参 考 文 献

1　Concept of Operations for the ESC Product Line Approach: [Technical Report, CMU/SEI-96-TR-018, ESC-TR-96-018]

2　Brownsword Lisa.et al. Successful Product Line Engineering: A Case Study. Software Technology Conference, April 1996

3　Brownsword Lisa, Clements Paul. A Case Study in Successful Product Line Development: [Technical Report, CMU/SEI-96-TR-016, ESC-TR-96-016]

(上接第 60 页)

9　Sorumgard Lars Sivert, et al. Experiences from Application of a Faceted Classification Scheme. in Advances in Software Reuse. In: R. Prieto-Diaz, et al. eds. the Second International Workshop on Software Reusability. Lucca, Italy, 1993. 116～124

10　Widom J. Research Issues in Data Warehousing. Prepared talk on DB/Expro 95(with slides)

11　杨冬青，唐世渭．应用的深化推动数据库技术发展．计算机科学，1997，24(2)

12　Li Keqin, et al. An Overview of JB(JadeBird) Component Library System JBCL. In: Jian Chen, et al. eds. Proceedings of the twenty-fourth International Conference TOOLS. ASIA, Beijing, 1997. 261～267

13　青鸟Ⅱ型项目组．青鸟构件模型：[内部技术报告]．北京大学：计算机科学与技术系，1997

14　Inmon W H. Building the Data Warehouse. John Wiley & Sons Inc. 1993

15　Codd F E, et al. Providing OLAP(On-Line Analytical Processing) to User-Analysts, PC WORLD, 1993 9

16　Kimball Ralph. A Dimensional Modeling Manifesto, DBMS Online, July 1997

17　杨燕燕，等．数据仓库星形模式及其建模工具研究．软件学报，1998-6

18　Gill Harjinder S, Rao Prakash C. The Official Guide to Data Warehousing. Que Corporation, 1996

19　Kathy Bohn Converting Data for Warehouses DBMS Magazine, 1997(6)