

DB2 OLAP 服务器

理论与实践

深入了解矩阵管理和存
储结构

高级 OLAP 设计实践

实际实施经验



Corinne Baragoin
Jorge Bercianos
Janez Komel
Gary Robinson
Richard Sawa
Erik Schuinder

ibm.com/redbooks



国际技术支持组织

DB2 OLAP 服务器理论与实践

2001 年 4 月

注意！

使用此信息及其支持的产品之前，请务必先阅读第 241 页的附录 G “注意事项”。

第一版（2001 年 4 月）

此版本适用于可在各种平台上使用的 IBM DB2 OLAP Server 7.1。

请将您的意见寄往：

IBM Corporation, International Technical Support Organization
Dept. QXXE Building 80-E2
650 Harry Road
San Jose, California 95120-6099

当您的信息发送给 IBM 后，即授予 IBM 非专有权，IBM 对于您所提供的任何信息，有权利以任何它认为适当的方式使用或散发，而不必对您负任何责任。

© 版权所有 国际商用机器公司 2001 年。保留所有权利。

美国政府用户注意 — 限定权利文档 — 使用，复制或公开文件应受到
IBM 公司签订的 GSA ADP 时效合同(Schedule Contract)所规定条款的限制。

前言

本 IBM 红皮书探究有用的设计过程以及分享在 DB2 OLAP 服务器和 Essbase 方面的重要实施体验。涵盖 OLAP 的设计要点到 OLAP，但并非是详尽无遗的。实施和部署的经验是通过与客户和合作伙伴的访谈描述的。

本红皮书的重点放在 IBM DB2 OLAP Server 版本 7（包含多维存储选项）和 Hyperion Essbase OLAP Server 版本 6.0。本白皮书所使用的术语 DB2 OLAP 专指这两种产品。

下表 1 列出 IBM DB2 OLAP Server 和 Hyperion Essbase OLAP Server 的版本。

Hyperion	IBM
Essbase Server 5.0	DB2 OLAP Server 1.0
Essbase Server 5.0.2	DB2 OLAP Server 1.1
Essbase Server 6.0	DB2 OLAP Server 7.1

表 1 Hyperion 和 IBM：产品版本及发布

1998 年 2 月，IBM 推出新型分析软件：基于 Arbor Essbase 5.0（运行于 Windows NT、OS/2、AIX 平台）的 DB2 OLAP Server V1.0。DB2 OLAP 的第一个版本仅包含位于 DB2 通用数据库上的关系存储区，而没有多维存储区。1998 年 10 月，1.0.1 版扩展到新的 UNIX 平台，包括 SUN SOLARIS 和 HP/UX。

1999 年 9 月，IBM 推出 DB2 OLAP Server 1.1，兼有关系性存储和多维存储，并以 Essbase 5.0.2 为基础。1.1 版从 2000 年 2 月就可运行于 OS/390 平台，而从 2000 年 6 月又可运行于 AS/400 平台。

2000 年 6 月，IBM 推出在 UNIX 和 Intel 平台上运行的基于 Essbase Server6.0 的 DB2 OLAP 7.1，2000 年 12 月推出运行于 AS/400 的 DB2 OLAP 7；而早在 2000 年 11 月 IBM 就推出了运行于 OS/390 平台的 DB2 OLAP 7。

2000 年 12 月，IBM 推出 DB2 OLAP Server Analyzer。它基于 Hyperion Analyzer 7.1（在本红皮书中简称为“分析器”），是一种适用于 Windows 和 Web 的易用 OLAP 客户机。

具有 DB2 OLAP 基础知识的设计人员和技术人员将会从阅读本红皮书中受益匪浅，从而有助于设计和实施未来的 DB2 OLAP 解决方案。

本红皮书的编排方式

本红皮书的最初构想是作为一本关于 DB2 OLAP 的“最佳做法”书。对任何学科最佳做法的透彻知识极少仅掌握在一两个人手里，因此，本红皮书也是许多个人的集体贡献。它实际上是最佳合作的结果。

有些贡献者没有留下姓名，但他们都十分重要。他们在 OLAP 原理和实践方面的工作具有创新精神。他们的实践活动是非常基础的工作，因而常常被人们忽视：这些实践活动最初曾经是“已开发过的”。但是，我们在这里无法一一介绍他们。然而，有四组主要贡献者（IBM 公司、Hyperion 解决方案公司、业务合作伙伴和 DB2 OLAP 的客户）值得一提。

本红皮书编写小组

IBM 的 DB2 OLAP 红皮书项目小组常驻国际技术支持组织“圣何塞中心”工作。该小组的任务是管理本红皮书项目、研究和试验许多建议的习惯做法，并最终产生像您现在所看到的文档。

Hyperion Solutions 的人员除没有参加文档的最终制作和交付而外，在本项目各个方面均提供了帮助。

IBM 公司和 Hyperion 解决方案公司双方的业务合作伙伴也做出很大贡献。这方面的人员编写并提交了本文相当大部分的内容。

最后，OLAP 的客户以接受 IBM 访谈的形式做出了贡献，介绍了目前能反映其自己的 DB2 OLAP 生产环境的实践及过程。

部分个人撰稿人：

- 下列人员编写了本书部分章节：**Corinne Baragoin、Gary Robinson、William Sterling、Richard Sawa、Cheryl A. McCormick、Paul Turner、Debra McRae、William Hodges、Bob Wallace**，以及 **Dave Nolby**。

- 下列人员进行了研究或接受了访谈，并提供了本书中的部分内容：Daniel DeKimpe、Christopher Dzekian、Jim Burnham、Steward Teed、Sujata Shah、Leah Wheelan、Rich Semetulsis、Alan Farkas、George Trudel、Joe Scovell、Jacques Chenot、Mark Rich、Steve Beier，以及 Aster Hupkes。
- 下列人员进行了研究、归档及测试工作：Jorge Bercianos、Janez Komel 和 Erik Schuinder。

因此，本书内容并非一家之言。尽管协作风格各不相同，但内容翔实、言之有物。

IBM 项目组成员：

Corinne Baragoin: 国际技术支持组织圣何塞中心的商业智能专家。加入 ITSO 之前，她曾在 IBM 法国公司从事售前技术支持工作。早在 1996 年，就已开始研究数据仓库环境。

Jorge Bercianos: IBM 乌拉圭公司数据挖掘专家。他涉足数据挖掘和 OLAP 集成已有两年时间。

Janez Komel: IBM 斯洛文尼亚公司软件技术销售经理。他在数据库管理和数据仓库实施方面拥有 12 年的工作经验。

Gary Robinson: IBM 公司（总部位于美国硅谷实验室）高级软件工程师。Gary 在商业智能和决策支持方面已拥有超过 12 年的工作经验。他是 DB2 OLAP Server 开发小组的一名成员，帮助客户和合作伙伴部署 DB2 OLAP Server。

Erik Schuinder: 设在荷兰的 IBM 全球服务部的商业智能 IT 专家。曾作为 OLAP 设计师/顾问参与在大型数据仓库上实施 OLAP 解决方案。

William Sterling: IBM 的一名在世界范围内的 DB2 OLAP 技术专家。

Daniel DeKimpe: IBM 硅谷实验室的一名软件咨询工程师。

Hyperion Solutions 公司项目组成员：

Richard Sawa: 担任 IBM 和 ShowCase 的 Hyperion Solutions 公司渠道销售技术经理。他在关系数据库和多维度数据库技术方面拥有 11 年的工作经验，专门从事 Essbase 数据库调优工作。

Cheryl A. McCormick: 数据集成服务组的一名高级首席顾问。

Paul Turner: Essbase 业务开发部的一名高级技术经理。

Debra McRae: eCRM 服务小组的一名区域实务经理。

Christopher Dzekian: Hyperion 分析工具部的业务开发和产品管理主管。

Jim Burnham: 信息技术服务组的首席软件工程师。

Steward Teed: 信息技术服务组的高级开发经理。

Sujata Shah: 客户产品保证实验室 Essbase 技术方面的高级产品保证工程师。

业务合作伙伴项目组成员:

William Hodges: 来自 Tech-OLAP 公司（一家在加拿大魁北克省蒙特利尔和美国康涅狄格州斯坦福德设有分支机构的 OLAP 咨询公司）。他曾经是蒙特利尔魁北克大学的 MIS 教授。

Bob Wallace 和 Dave Nolby: 在明尼苏达州圣保罗的 LumenSoft 公司工作。该公司致力于通过咨询服务、产品开发和训练，提供增值 OLAP 的解决方案。Bob 是一名 OLAP 认证的专业人员，在实施商业智能解决方案方面拥有 9 年工作经验，目前就职于产品开发部。Dave 有 10 年以上商业智能工作经验，是一名 OLAP 认证的专业人员和培训师。

Leah Wheelan: Beacon Analytics 公司总裁。该公司是一家专门从事 OLAP 实施的咨询公司。

Rich Semetulskis 和 Alan Farkas: 两人都在 ThinkFast 咨询公司工作。Alan Farkas 是高级 OLAP 顾问，拥有六年以上的 Essbase 经验。1988 年就开始向客户提供 OLAP 解决方案。

IBM 全球服务部的客户和客户代表:

George Trudel: Scrip Pharmacy Solutions 公司的信息服务总监, 拥有 25 年的商业和技术经验。1990 年开始从事多维数据库方面的工作, 是第一个试用 DB2 OLAP 的商业用户。Scrip Pharmacy Solutions 公司总结出了处方交易数据, 作为向其客户提供知识的组成部分。

Joe Scovell 和 Jacques Chenot: 两人都在堪萨斯城 DST Systems 公司工作, 该公司是共同基金行业的一家过户代理。Joe Scovell 担任客户服务经理, 拥有 13 年与众多不同的共同基金客户合作的经验。其专业知识集中在向客户提供使其能够智能挖掘数据的增值服务。

Mark Rich: 在 IBM 担任财务和业务分析师 16 年。1994 年, 他曾担任 IBM 的世界合并会计项目 — Hyperion 企业实施 — 的技术负责人。1998 年, 成为董事会成员, 负责领导 IBM 的“世界规划系统”, 该系统目前是 IBM DB2 OLAP 在 IBM RS6000 技术方面的基础。

Steve Beier: 是 IBM 全球服务部的高级 IT 专家, 目前在 IBM 存储部研究企业信息系统, 主要职责包括项目体系结构、技术指导和“复兴”支持。

Aster Hupkes; 现就职于设在荷兰的 IBM 全球服务部, 担任商业智能组的 IT 专家, 在荷兰多家客户设计和实施 OLAP 解决方案方面拥有两年以上的经验。

欢迎评论

您的评论对于我们十分重要!

我们希望 IBM 红皮书能为您提供尽可能的帮助。请将您对本书或其他红皮书的意见通过以下途径寄给我们:

- 将 263 页的“IBM 红皮书评论”中的评价表传真至表上写明的传真号。
- 使用位于以下地址的网上评价表: ibm.com/redbooks
- 将您的评论以 Internet 附注的形式发送至: redbook@us.ibm.com

第 1 章 OLAP简介

本章中,我们将阐述联机分析处理(OLAP)服务器在信息系统基础设施中所占显著地位的重要性。接下来,我们将描述 DB2 OLAP 数据存储结构,以及数据库设计者如何掌握稀疏矩阵管理的方法适应(稀疏)矩阵管理的。最后,我们将推荐一些有助于设计者改进其管理数据爆炸技能的策略。由于数据爆炸往往在实施 DB2 OLAP 时发生,我们认为这三个主题关系非常密切。

1.1 最佳实践始于先进的理论

本红皮书实际上是一本适用于 OLAP 实施者的务实的教材。对于书籍的任何推荐似乎都不应对解释或相应技术做过长的论证。相反,应着重向读者介绍相关原则,即如何充分利用该技术、如何避免错误地解释其功能、如何优先考虑具有较高成本/效益比的实施目标,等等。换言之,作者认为读者应有机会体验到这一有益结果,甚至能够充分理解其基本原理,以将该技术的原始力量展现出来。

IT 专家有充分证据表明 OLAP 对人们来说还只是一个 IT 解决方案。经常由于它是一个新型的解决方案,看似可行,某种形式应用相对容易,或者别人碰巧的成功而去实施,很少是因为对其深刻了解。实施此解决方案很少是因为深刻了解它确实是最适合的解决方案。

商界也有充分证据证明 OLAP 有众多定义。OLAP 已达到这样一种状态,即可作为强义词、术语或标志,正如其建议者所说的那样,用起来也令人满意而有效。这里有一个很有说服力的例子:OLAP 是进行财务数据合并的一种绝好技术,在此方面比 SQL 更好,更易于使用。但是,无论执行合并的工具或产品多么高效和精彩,这种工具或产品却并不是一种 OLAP 工具或产品。

为什么情况会这样呢？因为 OLAP 还有许多其他功能。事实上，合并只是 OLAP 工具应能执行的最基本任务，而且通常认为是一种理所当然的功能。让 OLAP 专家使用 OLAP 产品实施合并，除非还有其他更为重要的挑战，几乎能肯定，该 OLAP 专家不会对这项工作产生兴趣。

正是由于这种性质，我们才认为关于 OLAP 的务实书籍必须首先简洁明了地确定相关技术存在的理由。必须更深入地了解产生这种技术的原理，才有助于实施者更好地利用它。就部分人而言，OLAP 看起来极像一种“巧妙的技术”，一种最终用户可以选用的工具，而不太像是那种重要、可靠、稳定、经济、有效、广泛关注的组织信息系统基础设施的必需组件。DB2 OLAP 并非一种最终用户工具；*而是一个面向矩阵的应用程序服务器*。事实上，它可能也是市场上唯一一种名副其实的 OLAP 服务器。对于 DB2 OLAP 所能做的事情，其他类型的数据服务器均无法做到，客户需要了解这一点。对于每个公司都拥有 DB2 OLAP 的说法可能有些过分，但一点也不夸张。对于公司和个人没有拥有当今最好技术的原因，存在很多合理理由。

OLAP 在机构 IT 基础设施中扮演的角色

就像构成首字缩略词的单词(“联机”、“分析”和“处理”)所表示的意思一样，OLAP 在机构中的角色是使用户能以轻松的交互方式访问、分析资源，以达到支持管理或决策的目的。从历史角度看，决策支持理论认可两种类型的分析资源，即数据(静态信息)及模型(动态信息)(Steven Alter 著《决策支持系统：当前实践和连续挑战》，Addison-Wesley 公司 1999 年出版)。OLAP 完全符合该定义，即 OLAP 是一种响应查询有关原始数据和派生数据的信息服务器。原始数据是加载到其数据库中的数据，而派生数据则可能是该数据的摘要、从同一数据派生的或独立计算出来的预测、一种假设分析情形，等等。它们都由驻留在同一数据存储区内的公式或程序(而非外部程序)进行计算。总之，OLAP 是一种比其他技术更优秀，并结合了访问及计算便捷性的组合技术。

在大多数情况下，将数据组织为矩阵或矩阵集合可最好地实现这一角色。因此，本身并不表示任何特定数据结构的术语。OLAP 便成了“多维数据库”的同义词，并产生两个重要挑战：

1. 管理矩阵大小和稀疏性
2. 按照面向矩阵的公式和脚本建立商业环境(或其他)行为的模型。

这两个挑战又蕴涵另外一个挑战：

3. 培养通过大脑想象多维数据安排和数据计算的能力。

1.2 理论在OLAP实践中的作用

理论(现象中所包含相关事实及关系的解释)肯定有助于我们成功面对这些挑战。一方面,迄今为止尚未有人正式或完整地提出 OLAP 理论。然而,少数几个变动基本原理就足以证明 OLAP 作为任何组织信息基础设施中关键组件的潜力。这才是我们完全真正需要的。这些基本原理虽然简单,但却很有说服力,因而不容忽视。

首先,上段中已经提到,OLAP 由决策支持理论支持;其次,它是关联算法和数据结构组合以产生计算能力的计算基本原理应用(我们将在下面讲述);第三,它是最适合商业模型实施的环境,该模型可应用于系统动态原理(也将在后面讲述),有些专家(例如, Peter Senge, 其相关著作作为《第五门学科:学习性机构的艺术和实践》, Currency/Doubleday 公司 1991 年出版)认为,这实际上是可帮助我们处理复杂商业环境的唯一规则。

1.2.1 决策支持理论

利用建立决策支持系统的技巧,可以构建能解决模糊问题的基于计算机的解决方案。构建此类解决方案的秘密就是应用多层开发方法,将一般功能嵌入较低层,并同时获得该功能,这样,仅通过添加较高层,便可快速构建并修改最终解决方案。决策理论支持三种常用功能:数据管理、模型管理,以及接口管理(如 Ralph Sprague 和 Eric Carlson 在其《建立有效的决策支持系统》中所述。本书由 Prentice-Hall 公司于 1982 年出版)。在决策支持文献中,提供这三种功能的开发环境称为 DSS 生成器。DB2 OLAP 就是一种 DSS 生成器。

1.2.2 计算机科学基本原理

作为独特而更为出色的数据库技术，OLAP 的信誉因 Codd 最初公布其理论的方式不当而不幸受到玷污。Codd 创作了《向用户分析师提供 OLAP：一种 IT 需求》一书。本书可以在 www.essbase.com 上查阅。而且，该理论甚至还不完善，也不是真正的理论，相反，只是罗列了若干需求。此后，其他作者已尝试对其加以完善(例如，Erik Thomsen 著《OLAP 解决方案：建立多维信息系统》，John Wiley & Sons 公司 1997 年出版)。但基本问题在于我们还没有理论，而且可能永远也不会有，至少在 Erik Thomsen 提出的理论应该形成的抽象层次上不会有。

但是，如果我们认为 OLAP 不仅是一种数据服务器(就像 RDBMS 一样)，而且还是一种建模工具、应用程序开发环境、原型开发环境，那么，就不应按照与关系数据库模型相同的推理路线构造 OLAP 理论。例如，如果我们参考一下其他书籍，在经典计算机科学文献中，我们就会发现 DB2 OLAP 不需要依照集合论进行证明(就像 Codd 证明关系数据库)，也不需要依据缺少值的重要性进行证明(如 Erik Thomsen 那样)，而只是通过以下原则更为简单地进行证明：计算机性能是以数据结构和算法这两个组件的交互为基础的。

在这一点上，最有说服力的论述就是 Niklaus Wirth 所著的《算法 + 数据结构 = 程序》，(Prentice Hall 公司于 1976 年出版)。本书中，作者列出了三种基本数据结构，即记录、矩阵和位图。DB2 OLAP 是应用最佳可用数据结构解决手头问题的绝好实例。其既可将数据存储为记录，也可作为矩阵，它使用面向矩阵的算法进行简单计算和建立复杂模型，并使用位图加快对由这些模型所使用和/或生成的数据访问速度。

1.2.3 模型管理和系统动态学

如果我们计算世界去掉电子表格工具，就很难想象从 20 世纪 70 年代末一直到现在所发生的具有突破性的计算革命。许多人把个人电脑的成功归因于 Visi-Calcul 及后来的 Lotus 1-2-3，这使一般人能够使用计算机进行一系列计算，得出数字结果。现在，大多数电子表格工具用户知道我们不仅可以编写电子表格程序计算利润，也可让电子表格估算产生一个期望利润值所需的收入，完全利用公式计算，该公式并非专为计算收入而设计。我们称之为“寻找目标”。

这里要说明的一点是，我们可以寻找目标，并因此将前提转变为结论，因为公式集合是现实世界中某个方面的模型，通过操作该模型就可以更好地了解现实世界，就像使用飞机模型可以测试真实飞机的空气动力特征一样。为了有效使用，模型必需易于访问、操作、开发，以及易于存储。DB2 OLAP 具有所有这些特性。

1.3 位置依赖性和位置独立性

促成关系数据库模型理论的精确因素之一就是“关系”(表)中的位置是不相关的这一事实。某个记录出现在哪里以及某个字段出现在某个记录的什么地方与针对该表的查询产生什么结果没有任何关系。位置独立性是促使关系数据库技术取代旧技术(层次数据库，如 IMS 和 CODASYL 数据库)的特性之一。尽管旧技术具有明显的性能优势，但旧技术使用指针，从这个意义上讲，旧技术依赖于位置。位置独立性使数据库成为特定查询更灵活和更好的选择。

在早期编程历史中，程序员会偶尔通过提出以下问题来比较和评估编程语言：“这种语言是否包含数组？”以及“这些数组中允许的最大维数目为多少？”APL 是一种早期计算机语言，它包含根据线性代数定义的矩阵运算。例如，线性代数表达式 $A = B * C$ (其中 A、B、C 为矩阵)表示一种具有 $A_{ij} = \text{somme } B_i * C_j$ 类型的多项运算，换言之，数据结构和本机计算算法远远超出标准计算机语言所能“直接”进行的计算。

但是其问题不适合或不需要以矩阵形式数据表示的计算机用户，或不能将其计算难题概念化为矩阵的计算机用户，就无法利用 APL，不管 APL 作为一种编程语言是多么复杂，而且相对于早期的计算历史，还要更加复杂。

并不是所有的计算问题都需要、适合于或者可以受益于面向矩阵的处理。另一方面，那些属于这种情况的计算问题却仍然面临如何高效地管理数据存储的问题。就其本质而言，矩阵实际上是空间分配符。它们类似于蜂窝状邮箱集合，一旦给某个邮箱贴上标签，该邮箱就被保留，换句话说，也就取消了灵活使用某个物理空间的自由，而不管其所有人是否会收到邮件。在办公室有这种邮箱的人，只需直接走到他的邮箱就可以找到邮件。但是，将邮件放入邮箱之前，即使信封已按字母顺序分类并整齐地摆放在一个通用邮箱里，而且所有信封上的地址正确，这些相同的人也不大容易找到邮件。但是，这个通用邮箱占用的空间本质上要比蜂窝状邮箱小。

因此，在一种情形下是优势，而在另一种情形下则可能成为劣势。即使可以使用索引，在表中查找数据也会费时费力。在知道基本坐标值的情况下，从矩阵中查找数据则非常简单。您可以直接跳到相应的单元地址。如果将 RAM 比作磁带的话，那么矩阵则可比作桌子。在 RAM 中就像在矩阵中一样，当您知道要寻找什么东西时，您已经准确地知道这个东西的所在位置了。

1.4 处于基本级别的不同数据模型

矩阵和表之间有着根本性的差异，关系表(RDBMS)可将世界作为碰巧具有属性的相关实体集合来进行建模。属性不能独立于其实体而存在。有些属性可以成为实体标识符，但并非必须如此。最后，表(和 RDBMS)是面向实体的。

从另一方面讲，矩阵是面向数据的。在 OLAP 的特定实例中，矩阵是面向数字数据的。当我们使用某种编程语言将数字存储在内存中时，就会使用变量。例如，要将数字 546 放在内存中，我们可以确定 $VAR1=546$ 。要将四个数字存储在内存中，我们可以使用四个变量，或者使用一维矩阵(参见表 2)： $VAR(1)=546$ 、 $VAR(2)=765$ 、 $VAR(3)=762$ ，以及 $VAR(4)=951$ 。

表 2 一维矩阵

	A	B	C	D
1	546	765	762	951
2				
3				

我们可以将同样四个数字存储在二维矩阵中(参见表 3)：VAR(1,1)=546、VAR(1,2)=765、VAR(2,1)=762 和 VAR(2,2)=951。而且，我们能以同样的方式使用电子表格(一个电子表格就是一个矩阵)。

表 3 二维矩阵

	A	B	C	D
1	546	765		
2	762	951		

我们如何或为何选择一种或另一种方案呢？我们是根据各种情况下位置所赋予的优势来选择具体方案的。

例如，第二种方案可使我们轻松计算出各种小计，如表 4 所示。

表 4 计算小计

	A	B	C	D
1	546	765	1311	
2	762	951	1713	
3	1308	1716	3024	

因此，就为运算提供方便而言，位置非常有用。不仅如此，如果我们更改行和列标签(参见图 5)，就可以使用位置识别、确认和归档这些数字的实际意义。

表 5 命名行和列

	桌子	椅子	两种产品
纽约	546	765	1311
波士顿	762	951	1713
两个城市	1308	1716	3024

我们有意不写“赋予意义”。如果数据来源于现实生活，数据已经具有其意义，只是我们尚未认识它。这一点非常重要。OLAP 不是将维添加到数据，而是识别维，并使其立即可用。在此意义上，OLAP 是至少两种可用于表示维的不同方法中的一种，另一种方法是 Ralph Kimball 在其《数据仓库工具箱》(John Wiley&Sons 公司 1996 年出版)一书中提出的星型模型设计。

给列与行赋予了有意义的名称后，公式就可表示为，例如，椅子 = 桌子 * 4，表明每张桌子配有四把椅子(参见表 6)。

表 6 添加公式

	桌子	椅子	两种产品
纽约	546	2184	2730
波士顿	762	3048	3810
两个城市	1308	5232	6540

如果以上只是一种预测，而非实际的销售报表，并且波士顿的销售额通常只有纽约销售额的一半，那么，我们就可以建立一个预测模型，其中确定一个数字(我们期望纽约销售桌子的数量)就会得出纽约及波士顿的桌椅销售额(参见表 7)。

表 7 预测模型

	桌子	椅子	两种产品
纽约	546	2184	2730
波士顿	273	1092	1365
两个城市	819	3276	4095

总之，上面的微型模型中包含的公式是：

椅子 = 桌子 * 4；

波士顿 = 纽约 * 0.5 ;
 两个城市 = 纽约 + 波士顿 ;
 两种产品 = 桌子 + 椅子 ;

请注意，与电子表格中存储公式的方法相反，在 DB2 OLAP 中，公式只存储一次就可以在任何地方应用。同时请注意，上面的模型只有一个已知数。其余数字都是派生数据。(而且，如果我们添加产品及城市并建立类似的逻辑关系，则可建立一模型，该模型可根据一个数字计算出所有城市中所有产品的销售额！但是，本示例的目的是要说明应用和扩展著名编程范例(电子表格)的优势)。

通过使用相对及绝对地址，还可以进一步利用位置，如表 8 所示。

表 8 使用相对和绝对地址

	I	I	I	I	I	I
1		1	2	3	4	5
2	1	1	2	3	4	5
3	2	2	4	6	8	10
4	3	3	6	9	12	15
5	4	4	8	12	16	20

在表 8 中，粗体数字是通过将公式=\$A2*\$B1 输入单元 B2 中，再将此公式复制到单元 B2:F5 中所生成的公式计算而得来的。我们只需写一次公式，计算机就能够进行必要调整，正确地将同一个公式的不同版本应用到其他单元之中。无需深入了解计算过程，我们就能发现 DB2 OLAP 通过附加功能同样可利用位置。凭借这些附加功能，不需要为每个单元重新启动单元公式。计算公式时，会动态地对公式进行调整，而且实际上只存储公式的普通版本。同时，动态调整不局限于二维空间，且可扩展到 DB2 OLAP 数据库中存在的所有维。

1.5 数据冗余问题

计划实施 OLAP 解决方案时要克服的困难之一就是要证明数据存储机制，以及其中所包含数据的重复是正确的。数据仓库移动就是数据可用性改进的明证，而这些改进是通过不同方式重新确定数据而实现的。Inmon(其著作作为《建立数据仓库》，John Wiley&Sons 公司 1996 年出版)非常明确地指出，数据仓库及数据集市的内容中不存在冗余，因为其

中的数据经过重新确定，数据处于与其来源不同的级别。

1.6 DB2 OLAP多维数据库为矩阵

DB2 OLAP 存储结构作为矩阵(或数组)实施数据存储的概念也许是 DB2 OLAP 开发人员应该学习的最为重要的概念。

对矩阵管理的完全了解包括理解稀疏性的推论概念。也就是说，添加到矩阵中的维越多，矩阵中实际包含值的交点(或单元)所占比例就越小。

将二维数组的商业示例看作是含有“度量”维的三个成员，分别称为销售额、销售成本及利润(销售额减销售成本)，按照“时间”(第二个维)每天进行记录，持续一年。随着时间的推移(日复一日)，我们通常能够直观地测量销售额和销售成本，并生成利润。此模型会有 1,095 个交点(365 天乘以 3 个度量 - 参见图 1)。“日”和“度量”紧凑迁移的描述十分容易理解。对于给定日期，如果有一个销售额数字，您可能也有销售成本信息，并可因此计算出利润度量。除非有销售信息季节波动剧烈，否则多数时间里都会有销售信息，因此，可能在所有时间内矩阵被“紧凑”填充以进行“度量”。

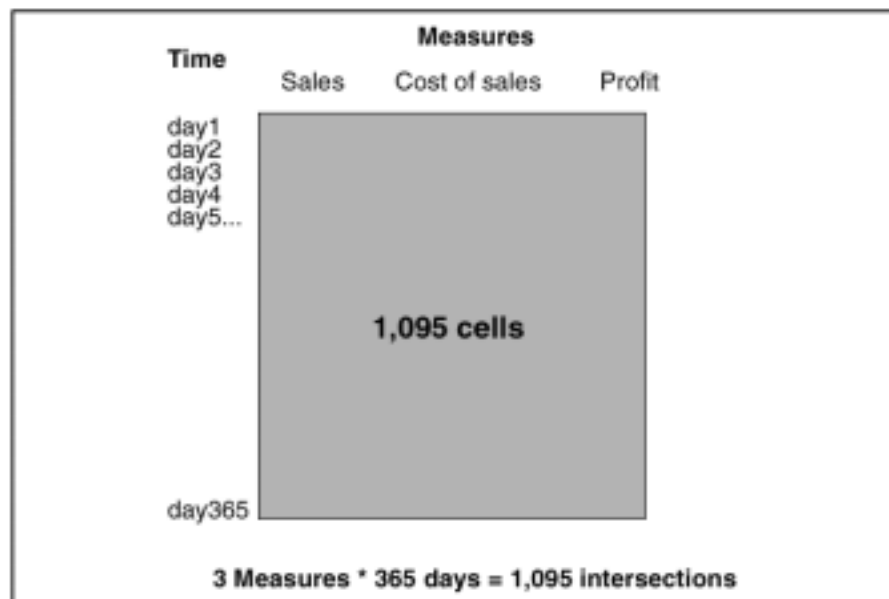


图1. 交点数量——示例

当我们添加更多维时,就会涉及到稀疏性这一概念。例如,试图将“客户”(数量为 100,000)和“产品”(数量为 50)的维添加到该模型。通常很容易想到,这些维中将存在许多不包含数据的交点。并非所有客户都会在一年中的每一天购买所有产品。

随着每个维所有成员的笛卡尔乘积般的增加,交点的数目也随之增加。我们的模型就从 1095 个单元增加到 5,475,000,000 ($365 \times 3 \times 50 \times 100,000$)个单元。

仅通过将数据存储为销售事件结果,交易关系数据库便可处理数据稀疏性。这些数据库将记录实际销售额。然后,可以使用结构化查询语言(SQL)询问和分析数据以回答类似的以下问题:哪些产品在规定时间内销售额最少?此答案是由程序派生的。关系数据库一般不存储(许多)客户没有购买(许多)产品这一事实的表。但数组则会。

通过声明，矩阵会为每个产品和每个客户保存所有时间内销售额和销售成本的各种可能组合。其中许多单元不包含数据。数据库中包含哪些产品在所有时间内均具有最少销售额的答案，并且只在检索相关交点以及检查其内容时，才显示此答案。

理解稀疏性的概念显然非常容易，但遗憾的是，不如推断 DB2 OLAP 存储结构是可用于使稀疏性影响最小化的中央设计机制那样简单。数据块和索引的两种存储结构实际上是由 Essbase 的设计人员开发而来的，目的是为了处理稀疏性这一现实问题。

1.7 数据块和索引说明

请看以下包含 21,370,660,375,680 个交点的数组说明：

DIM (172, 21, 27, 32, 209, 32765)

Arbor Essbase 的创建者将维标为紧凑或稀疏。当一个维标为紧凑时，就成为称作数据块的存储结构的组成部分(参见图 2)。

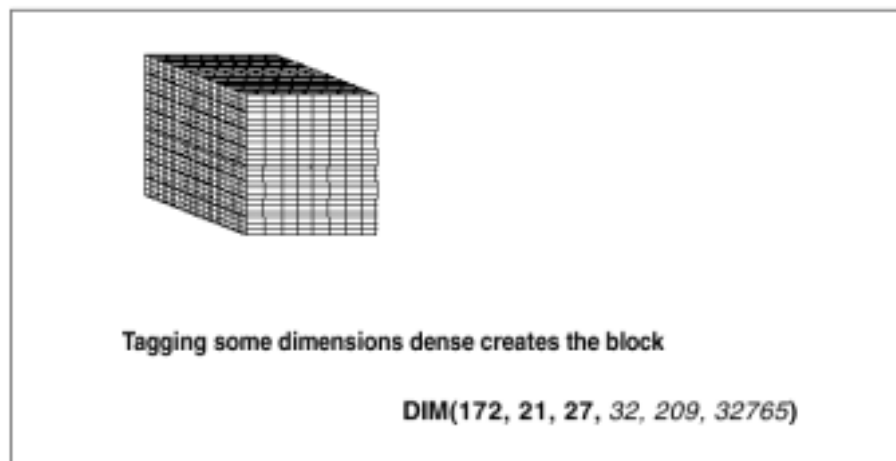


图 2. 紧凑维形成的块

数据库中创建的每个数据块都具有一个相同的结构。在本例中，精确地说，数据块包含 $172 * 21 * 27 = 97,524$ 个单元或交点。所有数据块都存储在磁盘上的 ESS*.PAG 文件内。

数据块寻址或定位是通过组合稀疏成员进行的。这些组合成为称作索引的存储结构的组成部分，并存储在含有 ESS*.IND 文件的磁盘上。通过实现数组维的这两个定义，Arbor Essbase 创建者实现了模块化矩阵。

数据块是一种固定格式的数据结构，其存在方式取决于索引中与数据相关的稀疏成员组合。“与数据相关”是指，只有在稀疏成员组合之间实际存在商业数据的地方才生成数据块。因此，如果我们一月份在北极没有销售任何 Sun 褐色石油，那么在数组中就不为那些交叉坐标保留任何空间。DB2 OLAP 存储结构与关系结构的区别之一为关系索引是否可选。在 DB2 OLAP 中，该索引不是可选的。删除关系表索引不会对表数据产生任何影响。删除 DB2 OLAP 数据库中的索引则会破坏该数据库。

数组的小型子组件(数据块及其索引地址)在磁盘和工作内存之间移动起来十分容易。这些结构非常符合一般用户需求：仅对任何时间点数组中的信息子集感兴趣。

1.7.1 块创建研究

上面的数据库中包含 6 个维，并具有以下 DB2 OLAP 配置：

- 紧凑维#1，包含 172 个成员
- 紧凑维#2，包含 21 个成员
- 紧凑维#3，包含 27 个成员
- 稀疏维#1，包含 32 个成员
- 稀疏维#2，包含 209 个成员
- 稀疏维#3，包含 32,765 个成员

实际创建哪些数据块取决于包含数据的独特稀疏组合(参见图 3)。

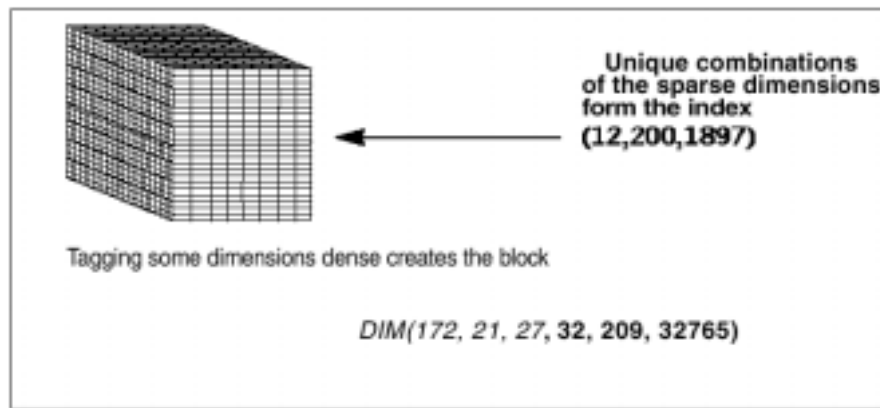


图 3. 稀疏维所形成的索引

在此情况下 ,只因为在该交点发生过商业事件 ,所以生成了带有地址(12, 200, 1897)的块。我们可以将上面的快照图 3 转换或解释如下 :

A&P(稀疏维#3 的客户 1897)在纽约(稀疏维#2 的成员 200)销售 colas(稀疏维#1 的成员 12)。

DB2 OLAP 以最简单的形式使机构将其商业元素标识为数组成员 ,并将数字值存储到该数组内的交点上。

1.7.2 矩阵爆炸

DB2 OLAP 数组的三个限定特征是 :

1. 维数目
2. 每个维内的成员数目
3. 每个维内成员的层次关系

数据爆炸可能会在每个特征之间发生 ,并会单独或同时产生一种组合(即笛卡尔乘积)影响。

例如 , 如果我们将稀疏维#1 增加到包含 5000 个成员 , 可能的交点数目会从 21,370,660,375,680 增加到 3,339,165,683,700,000 ! 以相似的方式 , 添加一个全新的维会使可能的交点数目激增 , 而且我们可以使两件事情同时进行 : 在添加一个全新维的同时 , 给一个维添加更多成员。下面的论述将模拟 DB2 OLAP 实施 , 并把稀疏维作为示例 , 同时假定读者对 DB2 OLAP 基本概念有一定了解。

试想，我们将 DIM(172, 21, 27, 32, 209, 32765)和 DIM(172, 21, 27, 5000, 209, 32765, 450.) 之间的差异作为例子。

OLAP 建模中第一条设计规则是使数据库设计中的维数目最小化。

尽管维数最小化是我们的首要指导原则，但是在该原则与企业的报表需要之间还是存在一定的弹性。问题是，需要将块爆炸的可能性限定为产生实际用于商业分析的交点。

矩阵爆炸的第三种方式不像前两种方式那样明显。它是增强内部维层次关系复杂性的结果。在数组 DIM(172, 21, 27, 5000, 209, 32765, 450)中，我们将对新的稀疏维 #4(包含 450 个成员)进行案例研究。

在牢记其他稀疏维的情况下，让我们假定该维是全平的，而且，除所有成员的上层之外，成员之间不包含任何层次关系。从图形上看，该维类似于图 4。

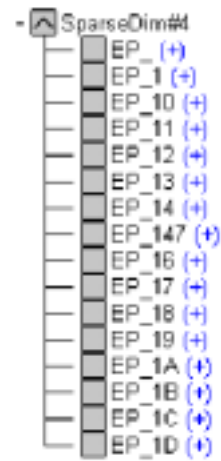


图 4. 无层次结构的维

该维总共有 449 个成员，全部聚集到一个上层，因而总共有 450 个成员。请参考图 4 中维与图 5 中维之间的数据点创建的含意。

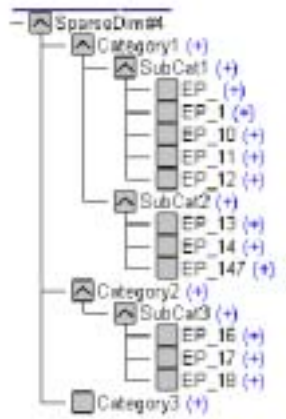


图 5. 具有嵌入式层次结构的维

在此声明中，我们拥有组织各组成员的层次结构，而且这对于我们的数据库有着重要的存储含意。

如果我们有成员 EP_11 的商业交易，但却没有平面稀疏维#4 中的 EP_12 的商业交易，那么就不会创建任何 EP_12 块。设计中，已有效地处理(消除)了数据稀疏性的影响。

但是，当我们将层次结构引入维，并且只存在一个子代的数据时，我们就是在声明存在上层数据点。在 SparseDim#4 的层次版本中，EP_11 处的事务将在 SparseDim#4 上层成员的 Category 1 的 SubCat 1 处生成数据，而且数据块创建会在有数据的所有其他稀疏维的每个成员之间产生一个笛卡尔乘积！这一结果具有极大的交叉维含意。

这里的问题不是平面维好，层次维不好的问题。相反，层次结构准确地反映出许多商业机构的真实情况，而且在建立这些关系的模型时，矩阵极为有效。真正的问题是层次结构给数组增添了复杂性，而且是导致多维数据库中发生数据爆炸的因素之一。我们必须充分理解其影响。

1.7.3 再谈数据块及索引

对于标为紧凑的每个维所有成员的所有交点(笛卡尔产品),数据块可为其保留一个单元。包含数据的稀疏成员组合可驱动数据块产生。因此,如果我们要实施将“时间”、“度量”和“产品”(假定无论如何我们每天都销售所有产品)的维标为紧凑的数据库设计,则只有实际购买产品的客户才会创建数据块。

更深入地思考一下上面的设计就会发现,由于数据块包含产品,而且并不是每个客户都购买每个产品(即使是在全年时间内),所以我们的数据块一般都包含空单元。从块(将“产品”标为稀疏维的块)中删除“产品”维可减小我们的块大小。这样,真正将实际购买特定产品的特定客户进行组合,就会产生数据块。

新的数据库设计已从存储结构中删除了不包含数据的交点。实质上,我们已设计一种可创建具有最高数据密度且最少数据块的模型。正是数据块密度与创建块数目之间有这一经验关系,因此它才可检测最佳数据库配置。

1.8 其他多维设计原则

从前面对于在 DB2 OLAP Server 中处理稀疏矩阵讨论起,我们就遵循了少数设计原则。这些原则最初在原来的“Hyperion 解决方案的 OLAP 基础知识”课程中作了简明描述,而且我们在此对这些原则进行了相当大篇幅的详细论述,但愿能够在理论方面起到一种承上启下的作用。

1.8.1 避免维间不相关问题

不相关维是其成员的多数交点与其他维多数成员组合时没有商业意义的维。请参考损益数据库包含员工薪金费用度量的示例。将员工主文件作为维包含在模型中的选项是以派生总薪金费用度量的方式出现的。实际上,在此模型内产生详细的员工分析能力具有诱惑力,但是,这样做的影响也是相当大的。因为,这样做会迫使创建绝大多数不包含相关商业数据的交点(例如,员工租金费用就是一个几乎没有意义的度量。)

人们可能会认为，如果将这些不相关维标为稀疏，则数据爆炸的影响就会被数据库配置有效地抵消。但事实并非如此，这些不相关维的出现仍然会导致在 OLAP 模型中创建交点。

我们再来看看第 16 页上图 5 中的维 SparseDim#4 的特征，看是否能将此维与另一个稀疏维进行组合。如图 6 的配置所示：

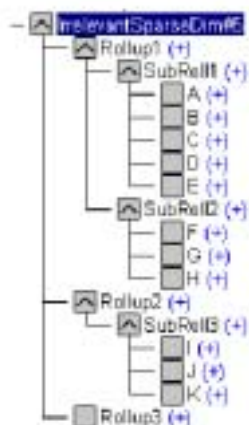


图 6. 不相关稀疏维

我们假定交点 EP_11 只在其他方面不相关维的成员之间才有商业意义。不相关维的每个成员(即成员 A、B、C 等)都会随 EP_11 一起加载。DB2 OLAP 存储结构内的影响将会为这两个维间的上层成员生成一个存储点。例如，此数组将在 SubCat 1->A 处生成一个交点。这个数据点绝对不包含任何业务值，但肯定有硬件和软件基础设施方面的额外开销。

由于维间的不相关性导致了不必要数据块的创建，因而生产效率低下。这些无用的块需要进行检索和计算，最终只会降低批处理和用户查询时的分析效率。维间不相关性的存在通常伴随着就模型中某些维(但不是所有维)而言有意义度量的存在。归根结底，维间不相关性意味着存在不止一个模型。

1.8.2 尽可能组合维

确定何时组合维的方法是调查维组合，以便查找一一对应关系存在于不同维成员间的位置。使用单独的维可以建立一对多的关系模型。维之间的一对一关系仅需求实施一个维。

请参看表 7 中的 2 个数据库大纲

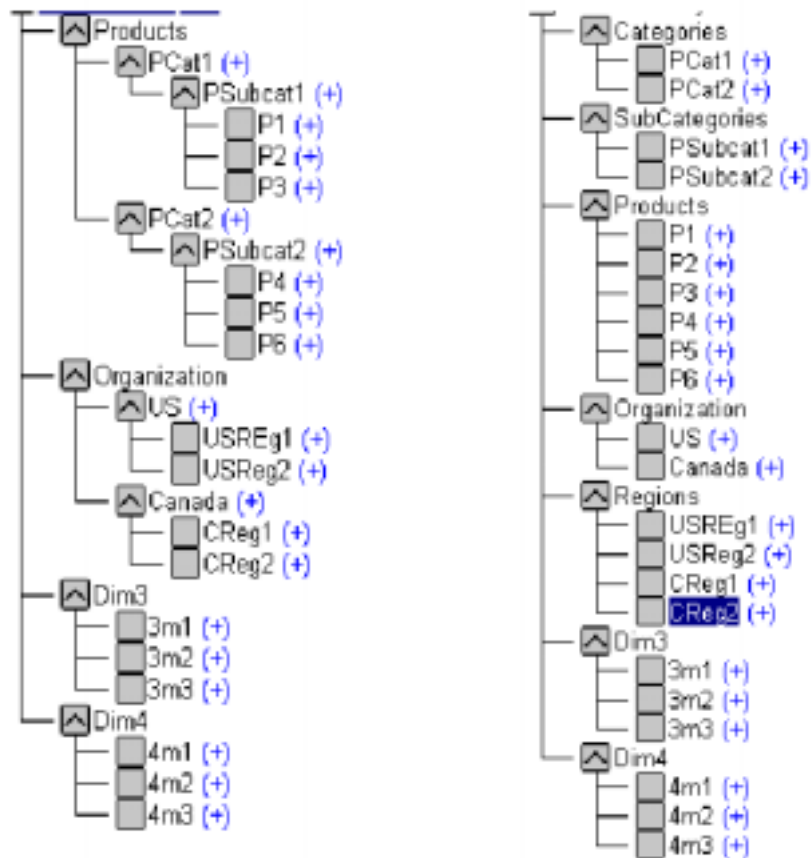


图 7. 结合的维

图 7 左边大纲有其按层次结构排列的一对一成员。右边大纲按单独的维排列层次结构。

左边大纲生成的交点总数为 1,232。右边大纲生成的交点总数为 15,120！

之所以发生数据点爆炸，是因为数组模型将生成绝对不包含任何商业值的交点(例如，交点 Products->Psubcat2。)这些不相关交点确实具有重要的存储结构额外开销。因此，仅需通过组合维(it2)，我们就可以设计一种更为有效的模型。

1.9 概要

本章概述了 DB2 OLAP 多维存储结构的主要组件。本章首先讨论了面向矩阵的应用程序服务器作为一种可行的信息系统数据库固有重要性。最后，提出一些非常一般的规则，这些规则可用于建立商业流程模型的实际任务，以便在多维数据库管理系统(MDDBMS)中实施这些模型。

使用 DB2 OLAP Server 实施商业模型并非一门科学，而只能称之为一种艺术，充其量是以某种巧妙方式进行的任何人类活动。设计成功与否与设计者理解 DB2 OLAP 存储结构的深度有着直接的关系，但不能说完全取决于此，而且不能说这一点对于 MDDBMS 比对于 RDBMS 更适用。

数据库实施各种组件的效率高低均取决于其对数据块和索引项的影响程度。甚至高效利用 DB2 OLAP 的 120 多个函数(本文不讨论这些函数)也取决于如何针对这些结构实施这些组件。

读者阅读概述实践和理论的各节时，需要将其牢记在心，以便确定这对于他们是否也是一条不证自明的真理。

第 2 章 OLAP模型开发一览表

本章介绍 OLAP 模型开发一览表。本章的最初意图是全面概述实施 DB2 OLAP 模型所需的步骤。

2.1 简介

下面我们将介绍 OLAP 设计方法的精华部分，其由经验丰富的 DB2 OLAP Server 和 Hyperion Essbase Server 开创者(以 William Sterling 为主)经过至少五年时间开发并不断地对其进行改进而成的。就数据库创建和设计的方面而言，此概述非常全面。就概述本身而言，它是对 DB2 OLAP 实施实践的一项重要贡献。

本章不包括对所介绍主题的深入论述，对于多数主题也不加任何评论。这我们这么做是有目的的。只要可能，我们都会列出本红皮书中包含的内容和实践参考资料。我们在表 9 中为读者提供了可用作 OLAP 一览表的概述，以便快速参考之用。如对该问题进行更深的论述，则只会使概述中简要表述的要点难以理解。

在参考有关内容和实践时，我们提供了类似于食谱的 OLAP 详细说明书。读者应查阅相关部分，以便了解要做的事情、具体步骤、具体方法，以及需经常练习的原因。作者认为，在理解了其他各章和附录包含的内容之后再阅读本节，效果会更好。

建议您复印如下一览表，并将其作为项目的一览表。最后一列为空白，以便您循序渐进地确认项目的进展情况。第 237 页附录 F，“OLAP 模型开发简明一览表”中还提供了此一览表的摘要。

2.2 OLAP一览表

表 9 提供的 OLAP 一览表可用于项目中，以帮助读者开发 OLAP 模型。

表 9 项目一览表

活动	参考内容？	检查？
1. 制定OLAP计划		
a. 识别OLAP机会。		
• OLAP是聚集分析	第1页的第一章，“OLAP简介”	
• 不应将数据仓库分割成块	第155页的附录A“OLAP数据集市设计方法”	
b. 评估OLAP的商机。	第29页的第三章，“OLAP项目管理”	
• 定义主题数据库		
• 技术人员必须了解OLAP应用程序将解决的业务事宜和业务问题。		
2. 进行高水平OLAP数据库结构建模		
a. 了解当前和所需报表的需求		
• 评估所需维		
• 针对每个报表，提出以下问题：“什么人、什么事情、什么时间、什么地点、什么方式”	第39页的4.1，“大纲原型设计”	

活动	参考内容？	检查？
<ul style="list-style-type: none"> 此时，开始设计维草图 <ul style="list-style-type: none"> 不要在纸上设计 将应用程序管理器作为快速应用程序设计工具(RAD) 在开发阶段创建OLAP大纲，以能够进行保存、向用户显示，以及快速修改 检查模型的范围是否恰当 <ul style="list-style-type: none"> 组合维2×2，并检查对维A \times 维B的分析是合理 	第39页的4.1，“大纲原型设计”	
b. 分组练习建模开发		
<ul style="list-style-type: none"> 把应用程序管理器作为一种快速应用程序设计工具 	第39页的4.1，“大纲原型设计”	
<ul style="list-style-type: none"> 将大纲投射到屏幕上，加快进展速度 	第39页的4.1，“大纲原型设计”	
c. 使用SQL评估数据，以便了解包含的数据量，以及检查数据的相关性	第42页的4.1.2.3，“尽可能使用SQL”	
<ul style="list-style-type: none"> 如果数据为关系数据，请使用 SQL <ul style="list-style-type: none"> 要确定维层次结构的范围，请对每个维执行“选择不同计数”操作 检查事实表的基数或将作为度量维来源的表的基数 		
<ul style="list-style-type: none"> 如果数据不是关系数据，使其成为关系数据 <ul style="list-style-type: none"> 也许需要使用桌面数据库软件 		
<ul style="list-style-type: none"> 检查用于建立维的数据空值 		
d. 预估所需开发软件硬件		

活动	参考内容？	检查？
<ul style="list-style-type: none"> 提出一种开发专用平台,(如果可能的话)使其与生产中使用的相似 	第32页的 3.3.2, “选择环境”以及第 98 页5.2节的, “访谈结果”	
<ul style="list-style-type: none"> 主要配置问题与内存和磁盘空间有主要关系 	第73页的 4.6, “性能调节: 缓冲器”以及第83页的 4.8.4.1, “估算数据库大小”	
3. 建立进行大小测试的原型		
a. 建立维		
<ul style="list-style-type: none"> 直接使用应用程序管理器, 手动键入维名称 		
<ul style="list-style-type: none"> 检查是否需要分区 	第86页的4.10.1, “分区技巧和策略”	
<ul style="list-style-type: none"> 初步设定紧凑/稀疏 <ul style="list-style-type: none"> 将应用程序管理器的“数据库/设置”对话框中的稀疏/紧凑建议作为第一切入点 	第45页的4.3.1, “稀疏概念: 维标记”	
<ul style="list-style-type: none"> 始终使用 esscmd.exe 脚本建立大型维 		
b. 让大纲发挥作用		
<ul style="list-style-type: none"> 使用成员标记 <ul style="list-style-type: none"> 仅使用标签 使用动态计算 将公式放在数据块中 	第52页的4.4.2, “成员标记使用注意事项”以及第 66 页的 4.5, “数据库计算注意事项”	
c. 加载测试数据		
<ul style="list-style-type: none"> 根据稀疏维对数据进行排序 	第91页的4.10.3, “数据加载优化”	

活动	参考内容？	检查？
<ul style="list-style-type: none"> 如果是SQL源，请用SQL进行转换。 <ul style="list-style-type: none"> 通过用 SQL 而非规则文件进行转换，向上准备数据 SQL 可以自成文档，而规则文件则不可以 	第 98 页的 5.2，“访谈结果”	
d. 计算和调节		
<ul style="list-style-type: none"> 计算 <ul style="list-style-type: none"> 把公式移动到大纲中，并执行 CALC ALL 命令。 对数据库进行配置，使其在一次中执行两遍计算。 通过 FIX 和 IF 集中计算。 使用智能计算。 	第 66 页的4.5，“数据库计算注意事项”、第 70 页的 4.5.3，“聚焦计算”以及第 84 页的 4.9，“智能计算”	
<ul style="list-style-type: none"> 调节 <ul style="list-style-type: none"> 使用压缩 安装高速缓存 	第 77 页的 4.7，“数据压缩”以及第 73 页的 4.6，“性能调节：缓冲器”。	
e. 随时测试大小、计算性能和查询		
<ul style="list-style-type: none"> 评估模型 <ul style="list-style-type: none"> 检查模型的大小 <ul style="list-style-type: none"> 使用SET MSGS ONLY方法 使用稀疏/紧凑方法 评估计算时间的模型 <ul style="list-style-type: none"> 使用SET MSGS ONLY方法 使用稀疏/紧凑方法 	第80页的4.8，“使用SET MSG ONLY”以及第60页的 4.4.4，“稀疏/紧凑方法”	

活动	参考内容？	检查？
<ul style="list-style-type: none"> 评估查询响应时间 检查电子表格检索因子应用程序日志 用户验证和用户验收 调整稀疏/紧凑维，以适应用户数据请求 	第89页的4.10.2，“应用程序日志”	
f. 改进紧凑/稀疏，优化大纲		
<ul style="list-style-type: none"> 使用稀疏/紧凑方法 使用配置向导 	第60页的4.4.4，“稀疏/紧凑方法”	
g. 根据原型大小测试，调整维和成员		
4. 建立和加载最终模型		
a. 用户验收测试：数据		
<ul style="list-style-type: none"> 使用电子表格工具 确保 OLAP 数据库能满足用户需求非常重要。计算是否提供用户所需的信息？用户对合并次数是否满意？数据库是否对其有用？ 	第36页的3.4，“验收测试”	
b. 用户验收测试：访问用户工具		
<ul style="list-style-type: none"> 编写报表 根据需要创建标准用户界面 如果打算向用户提供预定义报表，请使用最终用户的工具设计报表布局和运行报表 		
c. 建立和安装安全模型	第93页的4.10.4，“建立安全模型”	
<ul style="list-style-type: none"> 与用户一起检查安全模型 		
d. 训练用户		

活动	参考内容？	检查？
<ul style="list-style-type: none"> 在使用所选报表工具来访问数据时，用户应感觉轻松易用 		
5. 迁移到生产		
a. 确立系统管理		
<ul style="list-style-type: none"> 开发和编写生产过程： <ul style="list-style-type: none"> - 维护大纲 - 刷新/更新数据 - 验证数据 - 备份和恢复 OLAP 数据库 - 维护软件级别并安装修补程序 - 捕获错误 	第 98 页的“访谈结果”	

第 3 章 其他OLAP项目管理

本章重点关注 OLAP 解决方案的项目管理问题。本章内容基于 Bob Wallace 和 Dave Nolby 的经验。Bob Wallace 和 Dave Nolby 是 LumenSoft 公司的 OLAP 专家，均致力于通过咨询服务、产品开发和训练，提供增值的 OLAP 解决方案。LumenSoft 服务部门可提供优质的项目管理、OLAP 解决方案，以及全面的自定义 OLAP 课件。

3.1 OLAP项目管理的重要性

成功实施 OLAP 的关键是良好的项目管理规程。通过练习经过证明和考验的项目管理的方法，您的 OLAP 实施就可获得及时的并符合预算的结果，同时还可满足最具挑战性的业务需求。

严格的项目管理实践对于 IT 项目的成功极为重要，而 OLAP 的实施也同等重要。OLAP 的实施通常与业务端的联系比传统的系统开发项目更加密切，并且依赖性更强。

这往往意味着业务端工作人员和技术人员之间必须密切合作。开发 OLAP 项目所涉及的技术人员需要花费大量时间与抽取复杂业务规则和细微差别的业务端工作人员进行交互。此类交互完全不同于传统的系统开发，而且肯定需要一整套独特且来源于完善的业务及丰富技术经验的技能。

事实上，实施 OLAP 解决方案通常需要在业务和技术人员之间有深刻的内在联系，这种联系通常意味着：可靠的项目管理技术在确保项目成功中起着重要作用。也就是说，必须在传统项目管理实践中投入大量有意义的工作，例如，需求收集、系统设计、变更控制、实施以及最终项目验收。

如果借鉴传统技术项目，我们就会明白相同的公用项目阶段是存在的。这些基本阶段与下列阶段相似：

1. 启动阶段
2. 建设阶段

3. 实施阶段

这些阶段的每个阶段都有一个目的，而优秀的项目经理将对每个阶段的执行过程进行计划和安排，因为他们清楚，认真执行这些阶段会增加其成功的可能性，并有助于简化任务。

OLAP 的实施与上述传统阶段没有太大差别，经验丰富的项目经理在不太了解 OLAP 实施技术的情况下，也能侥幸成功。然而，要提高某位项目经理的成功能力，并将项目的不必要风险降到最低，则应将另外几个因素包含在传统项目计划中。但在讨论这些因素之前，让我们先概述一下开发一套完善的 OLAP 系统所需的组成部分。

在以下几节中，我们将讨论下列主题：

- OLAP 项目问题
- 实施 OLAP 解决方案的关键点
- 用户验收问题

3.2 OLAP项目问题

开展 OLAP 项目时，需要关注两个主要问题：OLAP 数据库的设计与开发和最终用户应用程序的设计与开发。

3.2.1 OLAP数据库

开发成功 OLAP 数据库的生命周期实际上是循环的。凭经验执行的许多任务通常会逐步发展为一种艺术，而非一门科学。

由于 OLAP 数据库是所有其他项目活动的基础，因而很容易被描述为项目中最重要且难度最大的活动。下面列出 OLAP 数据库开发过程的主要方面：

1. OLAP数据库结构建模
2. 大纲构造和属性应用
3. 公式和计算构造
4. 数据加载和验证
5. 数据库性能优化

3.2.2 OLAP最终用户应用

由于 OLAP 数据库被视为项目中最重要的一部分 ,最终用户应用程序在整个 OLAP 实施中的重要性可能会被轻视。遗憾的是,情况通常就是如此。因此,许多创意很好、设计精良的 OLAP 数据库惨遭失败,原因就在于它们不能满足最终用户群体的期望和需求。

整个 OLAP 的实施就像一辆豪华的汽车。OLAP 数据库是发动机,而最终用户应用程序就是车身及其所有附件。与汽车一样,数据库(发动机)是 OLAP(汽车)的最重要的组成部分,它有助于 OLAP 执行其设计任务。但是引擎不能单独完成任何任务 - 只有精心设计和实施的组件与发动机协同工作,才能完成规定任务。如果各个组件设计精良,最终用户(司机)就能够方便快捷地完成其任务。

考虑与实施成功的最终用户定义程序,事宜众多且多变。不可能在本章中有效论述。以下各节将会提供有关最终用户开发主题的更多信息。

3.2.3 OLAP项目小组结构

由于 OLAP 的实施在很大程度上取决于用户,因而确定角色十分重要。这些角色通常包括表 10 中所述的角色和责任。

表 10 项目小组角色

小组头衔	角色和责任
项目发起人	对项目交付事项、变更请求以及最终验收承担责任。确定目标和目的。
用户代表	负责提供系统功能和需求信息。
项目经理	负责支持项目小组和管理项目主导活动。
小组负责人	负责资源和进度管理。指导任务级项目活动。
签约人	帮助确定项目主导活动。帮助指导和开发这些活动。

3.3 实施OLAP项目

OLAP 项目计划包含项目评估和可行性研究、选择环境、计划和分析，以及主办。

3.3.1 项目评估和可行性

当然，每个项目经理都希望在启动项目之前确定可行性。项目能够成功还是注定失败？如果项目能够成功，是否可以在合理时间和费用范围内完成？

评估 OLAP 项目时应遵循标准的管理实践和原则，这将有助于评估实用性和潜在需要。其中，比较困难的部分是完成准确可信的评估。许多 OLAP 实施中的常见错误是过多地通过论坛、研讨会、调查和报表进行分析，因而导致结果不准确或不明确，小组成员意见不一、管理混乱而无效，从而造成时间和资金浪费。

进行有效评估的成功建议方案是为了获得 OLAP 和项目管理经验，应从概念证明(POC)项目或原型开始。

概念证明：一个重要成分是确定一小部分可管理的业务，其将提供适当的上下文，以便业务端从概念轻松推测出更大范围的情况。多数情况下，人们对多维概念知之甚少，因而最大的困难通常是对所达到的目标有基本了解。与传统的报表和分析系统相比，多维解决方案通常要强大灵活得多，因而让业务端完全掌握各种可能性相当困难。当完全了解解决方案的真正威力时，成功的概念证明通常会有“jaw on the table”的效果。

3.3.2 选择环境

具体因素不同，DB2 OLAP 环境会相差很大，这些因素包括与硬件和操作系统相关的公司标准、DB2 OLAP 项目的实际系统需求等。

理想的情况是, DB2 OLAP 环境包含至少两个也许三个服务器层次: 开发、测试和生产。对于开发层和生产层最好不要只用一个服务器。在单个服务器环境中操作不仅会牵制开发人员的精力, 而且会给该服务器上运行的生产应用程序带来风险。

环境决策还取决于是否已经存在 DB2 OLAP 服务器。无论是哪一种情况, 都应认真遵循满足磁盘和内存需求的建议调整大小技术(请参阅“*OLAP 数据库管理员指南*”)。在已有服务器的情况下, 必须进行全面的评估, 同时把目前服务器上运行的应用程序考虑在内。必须提出与以下示例相似的问题:

- 大概有什么磁盘空间需求?
- 大概有什么内存需求?
- 可能会在系统中添加多少用户?
- 是否会需要增加端口?

如果需要购买新服务器, 则具体方法有所不同。显然, 必须考虑与操作系统和硬件相关的适当公司标准。除此之外, 像前一种情况一样, 必须进行潜在磁盘和内存需求的过程。而且, 还应认真预测系统的未来增长情况。是否存在加载到服务器上的应用程序数量或加载到现有应用程序中的数据量方面的预期增长? 如果发现存在增长因素, 则通常会需要购买比当时应用程序所需的更大容量的服务器, 最低限度, 要有一个可以合理支持未来硬件升级的服务器, 以适应预期增长。

3.3.3 计划和分析(需求 and 设计)

本节将考虑项目规划和分析阶段。

3.3.3.1 项目指导需求

需求阶段是系统开发周期的一个关键组成部分, 因为它将最终决定项目的成功以及业务端验收。收集需求的目标是周密定义最终系统用户必须具有哪些条件才能满足其业务需要。这意味着不仅要定义项目中有什么内容, 而且要定义项目中没有什么内容。一个成功的需求收集阶段的结果是, 所有当事方都应清楚系统最终带来的结果。下面是可能会有的一些常见的 OLAP 开发需求:

- 多少最终用户必须具有对系统的 24x7 访问能力？
- 用户是否能够直接更新系统上的数据？这会导致额外的安全性需求，并有可能导致客户机应用程序需求。
- 是否限制某些用户的查看权限？这也会造成安全性需求。
- 多维数据集的定义是相对不变的，还是经常变化？也就是说，业务需求是否在多维数据集定义中规定可在一个或更多维度中经常添加、删除或移动成员？这会导致必须加以考虑的日常维护需求。
- 业务需求是否表明将给多维数据集中的一个或多个维度定义多个成员以及多种级别？这可能需要研究分区选项。
- 项目的业务原因是否基本上是信息检索和分析？是不是也有一个正在实施的过程(例如预算编制)。如果正在实施一个过程，就应该研究使用 VBA 应用程序或其他客户机工具控制该过程(比如，自动运行计算脚本、控制输入屏，等等)。
- 来自项目的报表是否完全是特定的，还是有一些固定报表格式需求？如果是使用固定报表格式，就应研究使用 VBA 应用程序或其他客户机工具以控制其一致性。

成功的关键因素还包括制定正式的收集需求方法，而不是只有松散的收集观念，从而确保可代表所有主要的系统用户，并在设计和开发期间校验这些需求。

3.3.3.2 建立OLAP数据库模型(大纲)

这一步骤是制定可设计 OLAP 数据库的详细用户需求，以满足报表和分析需要。

建模工作涉及创建一个模拟用户需求维度的原型大纲，该大纲应包含尽可能少的维度和成员，但必须充分反映由用户传递的维度内的维数和深度。

其次，数据库中将填充测试数据。一旦填充好测试数据，就使用诸如电子表格插件这样的客户机访问工具，与用户一起确保模型提供所需报表和分析信息。

与用户一起反复研究这些步骤，直到建立一个令人满意的模型。

3.3.3.3 开发OLAP数据库(大纲)

这一步骤是用于开发数据库多维数据集的实际大纲结构。

大纲将以上面建模过程中定义的结构为基础，同时又包含具有必需属性和公式的实际成员。有些成员可以手动输入，而其他成员则可以加载来自其他系统的信息文件。

如果维度成员是从其他系统的文件中创建的，则应在此阶段创建维度加载规则。

3.3.3.4 计算策略

如果数据的完全合并计算已满足需求，就不再需要开发计算脚本，因为默认计算可执行此功能。但是，如果大纲中的公式不能满足计算需求或在不计算整个数据库的情况下就可充分满足合并需求，则必须开发计算脚本。

在这种情况下，建议脚本包含模块化概念，而不是建立大而复杂的单片脚本，即使这再会导致出现多个计算脚本的情况下也应该如此。项目实施中会有这样的情况出现：可编写 API 应用程序，而不是创建一系列条件计算脚本，并通过动态生成脚本，再将其发送到服务器以执行该脚本，从而简化整个过程。

3.3.3.5 数据加载

通过电子表格插件可将数据直接加载到数据库中，也可以将来自其他系统的数据文件加载到数据库。

如果是从其他系统的数据文件中加载数据，就必须定义数据接口的文件结构，并且在该步骤中开发数据加载规则。

如果通过电子表格插件直接输入数据，则不需任何加载规则。但为确保完整性，可能会需要更多安全性需求。

3.3.3.6 报表

现在，可以利用电子表格插件或其他客户机报表应用程序来完成特定分析和报表。一般情况下，用户应接受电子表格插件使用方面的训练，以便熟练使用此工具。

如果要定义和使用常用报表格式集，就应研究 VBA 应用程序或其他客户机工具以促进一致性和可用性。

也可以使用像分析器这样的工具从头创建表示级信息(请参见第 218 页的 D.2，“DB2 OLAP Server 分析器怎么样？”)。

3.3.3.7 客户机应用程序

如果目标包括一致性、控制性和易用性，则应考虑创建 VBA 应用程序、API 应用程序或其他客户机应用程序。如果开发的项目是为了管理某个过程，这一点尤为重要。

3.3.3.8 性能

性能调整是一个经验过程，最好在加载和使用代表性数据集之后进行。这个周期性过程涉及高速缓存设置调节、有选择性地计算数据库各节计算脚本的使用、动态计算成员存储设置的使用，等等。

3.4 验收测试

新开发的 OLAP 应用程序在部署之前，必须由用户群体进行验收。这包括数据验证、应用程序设计验证以及客户报表工具验收。

验收测试是项目质量保证循环的重要组成部分，并由包含数据质量保证、应用程序，以及客户工具的正式测试计划推动。测试计划应由 OLAP 开发小组代表成员以及商业用户制定和实施。

任何测试计划都应经过深思熟虑，并应覆盖应用程序所提供的广泛的数据和功能。这归结为开发一系列可在应用程序内执行的操作，同时注意预期结果并让测试者记录实际结果。例如，在分析器平台上建立的新报表系统可以向最终用户提供一个包含十份财务报表的公文包。测试计划项目可以如表 11 显示的示例一样简单。

表 11 测试计划项目示例

步骤	操作	预期结果	实际结果	通过 / 失败
14	执行财务汇总报表	系统生成和显示含有数字精确的财务汇总报表，其与已知控制数相平衡	成功生成报表，所有财务数据都与控制数据相匹配。	通过

应当特别重视数据验证，因为数据验证通常是正常多维数据集建立周期的组成部分。对输入数据和派生数据都应进行验证。如果 OLAP 应用程序取代或补充现有报表，那么验证就可能像从 OLAP 应用程序建立同样的报表以及比较二者一样简单。如果报表不可用，则需要一种工具或程序，这种工具或程序可从源系统抽取数据并独立计算聚集，以便进行比较。如果可以自动进行抽取、计算和比较，则可以容易地将这个步骤作为建立多维数据集的批过程组成部分来执行，结果会使质量保证作为 OLAP 过程中正在进行的部分。

第 4 章 调优技术、良好的设计方法以及有用的提示

示

本章是对调优技术、设计方法和有用提示的汇总。其目的不是向新开发人员按时间顺序详尽无遗地讲述 DB2 OLAP 应用程序开发实践。本章讲述的提示和技术依据许多数据库参数(如数据库隔离级别、聚集缺少值等)进行了扩充。有关这些参数的具体使用方法，请参阅 DB2 OLAP 文档和 IBM 公司及 Hyperion 解决方案公司提供的系统管理员课程。

尽管本章的结构不是按照从头到尾设计的，但事实上可以这样阅读。这样一来，新开发人员可逐渐熟悉本章内容，并且在开发中遇到问题时，他们能够更有效地返回来查阅其中的详细信息和方法。通过逐页阅读，能够使本章更好地发挥常规参考资料的作用。同时希望经验丰富的开发人员也能从以下内容的阅读中受益。

4.1 大纲原型设计

一旦决定实施 DB2 OLAP 解决方案，接下来要执行的合理步骤就是原型设计。其主要目的是尽早准确了解分析及基础设施需求。

设计大纲原型时，要执行的最重要步骤是：

- 训练和建立开发小组
- 高级建模

4.1.1 训练和组建开发小组

OLAP 小组应是技术信息系统人员和商业人员互补性的融合。将任何一方排除在此过程之外都会导致灾难。而且，实际上，选择小组成员之后，原型设计中的第一步是对双方人员进行 DB2 OLAP 训练。

技术人员将专门研究 DB2 OLAP 软件及硬件基础设施需求。他们将深入了解客户机/服务器和 Web 体系结构，以及稀疏矩阵管理等问题，因为它们归属 DB2 OLAP Server 存储结构。训练过程旨在使技术人员熟悉有关 DB2 OLAP 大纲中建立商业流程模型的问题。

商业人员的基本角色是定义最终用户分析和报表需求，并使所开发应用程序的特定商业逻辑在 DB2 OLAP 大纲中得到适当反映。也许后者是其最重要的职能。

商业人员花费时间研究大纲内嵌入商业逻辑的技术问题是极为重要的。训练过程将使他们能够熟悉 DB2 OLAP 的商业流程模型软件及硬件基础设施的创建。

最后，OLAP 开发小组双方将获得各种成套技能，这些技能彼此补充、相互完善。一流的 DB2 OLAP 训练班由信息系统专业人员及其所支持的商业专业人员组成。

4.1.2 高级建模

开始高级 OLAP 建模的方法是将上述开发小组成员集中在一起，共同设计出一个基本全面的报表及分析需求列表。通过系统地分解当前示例报表，开发小组能迅速开发出报表属性列表，每个报表属性最终可作为 OLAP 维及层次结构反映出来。通过寻找维之间的一对一关系、最大限度地减少维数目等形式，可遵守第一章“OLAP 简介”中提及的所有 OLAP 一般经验法则。

4.1.2.1 创建报表和分析网格

生成示例报表模板列表，详细列出所有维和商业度量。

表 12 中显示的示例网格说明了各种结果。

首先，请注意原来确定为重要维的“关闭理由”、“状态”和“分配到”不能出现在任何报表中。可将它们删除。其次，注意“过程分析”维不会出现在同一报表“产品”和“接收渠道”维中。从此图表上，我们可以推断出一个八维模型和九维模型将产生所有报表需求。通过去除未用维，并将已用维分群在另外两种高效配置中，我们避免了使用十三维模型提供分析方法的必要。

表 12. 分析网格

	维用户报表列表										
		1	2	3	4	5	6	7	8	9	10
维列表	案例类型	x			x			x	x	x	x
	呼叫中心	x		x	x	x	x	x	x		
	时间	x		x	x	x	x	x			
	当天时间				x						
	产品	x		x	x			x			
	客户	x	x	x	x		x	x			
	严重性	x		x	x		x	x	x	x	x
	关闭理由										
	接收渠道	x		x				x			
	状态										
	分配到										
	过程分析		x			x	x				x

通过检查维组合，您可以更容易地确定：要提供分析需求，需要开发多少个 DB2 OLAP 数据库模型。当确定是否必须需要一个数据库标度策略(如分区)时，这样的网格非常有用。

4.1.2.2 作为集体开展设计工作：将应用程序管理器用作一种RAD工具

设计会话中，将应用程序管理器大纲程序作为一种快速应用程序开发(RAD)工具。不要在纸上进行设计，因为这会产生不属于 OLAP 大纲的标题、类别和元大纲，并会延迟考虑实际数据。使用大纲程序的最终结果是，在此开发周期的最初阶段您就创建了 DB2 OLAP 原型大纲。

将正建立的大纲从台式机投射到观察屏上，供大家观看。当您构建该大纲时，可使用电子表格插件来演示其 OLAP 功能(旋转、向下钻取.....)。您不需要为此而将数据加载到模型。通过电子表格插件查看旋转和穿透钻取功能，可以简化设计决策制定过程。

4.1.2.3 尽可能使用SQL

能够访问包含数据(这些数据与在设计会话中正讨论的维相关)的关系数据库极为有用。关系数据库的数据元素成为多维数据库中元数据的组成部分，如图 8 所示。

星型模型是支持 OLAP 功能最有效的关系模型，这并非一种巧合。准确地说，星型模型是反映数据多维特征的一种关系尝试。

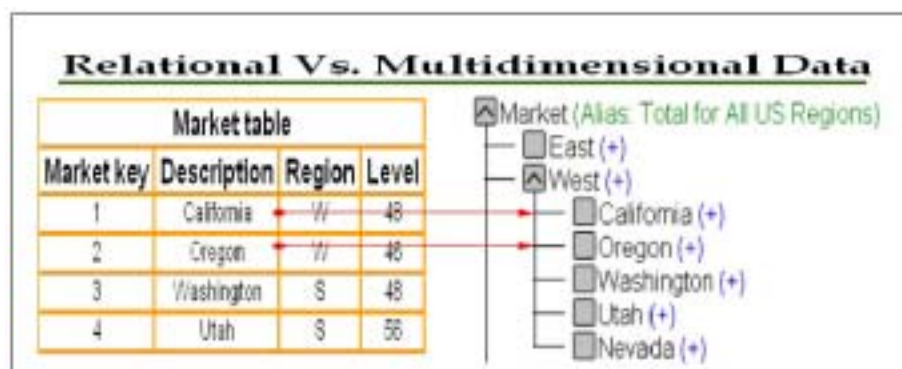


图8.关系与多维

如果可以访问一个关系数据库中的源数据，那就意味着，对各个维进行考虑时，开发小组可以快速生成表明维成员和层次结构的 SQL 查询。通过查找重复项、空值等，您可以了解源数据有多么纯粹。此外，通过检索成员数目，开发小组甚至可以找到有关 OLAP 模型潜在大小的普通但有用的信息。期望管理始于此过程。

如果资源允许，请考虑把将要用于执行最初数据库建立任务的源数据转换为关系数据库，具体地说，就是建立一个关系星型模型。

最重要的优点之一就是具有了研究详细级数据穿透钻取的实际能力。如果可以详细地访问源数据，当新的出乎意料的需求出现时，您同时也就具备了重建和改造源数据的能力。

归根结底，具备对源数据的完全控制能力可以极大地简化 OLAP 开发过程。一个重要好处在于开发小组可以直接看到将 OLAP 多维数据库与关系星型模型数据集市联系起来的体系结构的优点。

4.2 数据库调优简介

对 DB2 OLAP 数据库进行性能调优时，需要关注四个方面：

1. 如何处理维特征及嵌入商业逻辑
2. 如何实施“成员标记”
3. 如何处理大纲/数据库合并及商业公式
4. 如何确定最佳紧凑/稀疏设置

所有这些方面都围绕着一个最重要的 DB2 OLAP 构造：数据库大纲。

在 DB2 OLAP 环境中，数据库大纲就是数据库模型。大纲的存储特征对多维模型的重要性并不比模型的存储特征对关系模型的重要性小。适宜于 OLTP 或查询处理的关系模型关注模型设计及实现所需性能特征的关联存储结构的方式极为不同。只要讨论调节大纲的基础，就会讨论到 DB2 OLAP 存储结构的基础概念。

4.3 矩阵数据库基本概念

关系存储结构与多维存储结构之间的区别仅与其在多大程度上有助于证明 DB2 OLAP 存储结构是什么有关(参见图 9)：

1. 关系表中的值可以理解为多维数据库的元数据结构的组成部分。
2. 矩阵数据存储区消除了数据操作语言(如 SQL)进行数据检索的必要。对数据的访问是直接通过单元地址提供的。
3. DB2 OLAP 以最简单的形式让各个机构把其商业元素表示为某个数组的成员，并把数字值存储在该数组内的交点上。

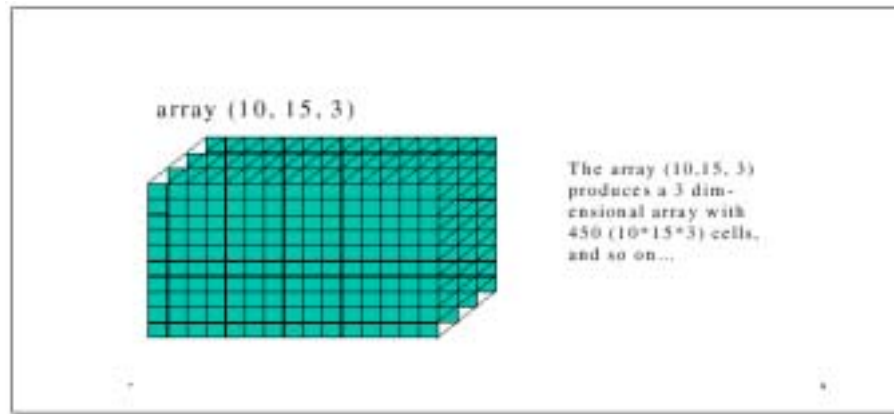


图9. 什么是数组?

程序员把数组的概念视为工作内存中用来临时存储信息的保留区域。以编程方式在数组中存储数据的部分精妙之处在于：如果出现检索数据需求，可以[直接](#)访问内容。

声明一个三(3)维数组(如图 9 中的数组(10,15,3))会导致创建 450 个交点，作为每个维的所有成员的交叉积或笛卡尔积，在这个案例中为 $10 \times 15 \times 3$ 。即使在这个普通示例的上下文范围内，我们也可以讨论稀疏性的概念。

让此数组存储 15 名销售人员 3 个时间段销售的 10 种产品。由于并非每个销售人员在每个时间段都销售每种产品，因而永远存在不含值的单元。一个标准化的关系数据存储区只存储实际发生的销售事务的值。矩阵为各个可能值保留空间，不管这样做是否有意义。

参考数组中的内容(参见图 10)通过提供相应的交点坐标 - 也就是提供单元地址 - 即可实现。DB2 OLAP 值数据检索只不过是提供单元地址给 DB2 OLAP Server。

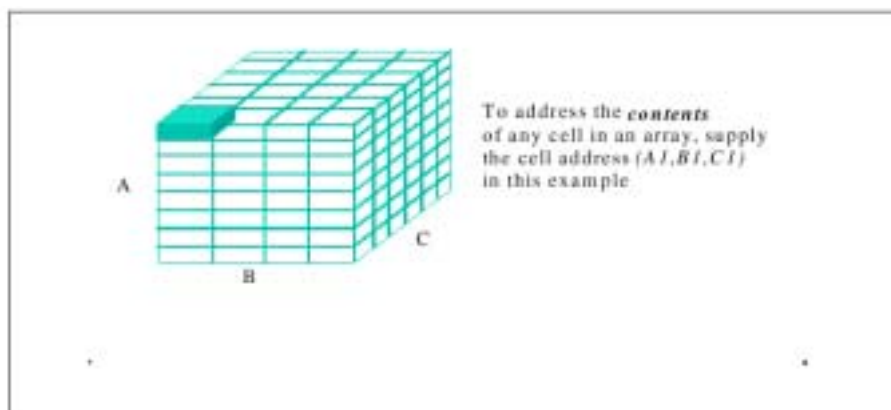


图10.相关内容

然而，在一个较深的层次，DB2 OLAP 使用户能够把复杂商业规则(计算脚本和公式)甚至可以向下反映到机构的给定多维视图内个别交点级别。这样，就可以通过计算机开发并测试真实和复杂的业务模型。

4.3.1 稀疏性的概念：维标记

任何数组或矩阵的总大小都是所有维间所有成员的笛卡尔积。下一图表说明即使是一个非常小的多维数据库，其可能的交点数目也多得惊人。使用多维数据库有丰富经验的开发人员都知道多数交点不包含数据。这就反映出所有商业数据集均具有固有稀疏性这一事实(参见图 11)。

Dimension	Members
Measures	172
Time	21
Mat	27
Structure	32
Product	209
Organizations	32765

Matrix Size = (172 * 21 * 27 * 32 * 209 * 32765) intersections

21,370,660,375,680 intersection points!

图 11. 多维数据库是固有的稀疏结构

但是，数组会为每个维的每个成员与其他每个维的每个成员之间生成一个交点，而不管商业现实中是否能够反映出该交点。

关系存储系统可用来专门存储商业生成数据。例如，把一月份 YoKon 地区没有售出野餐篮这一事实的数据的想法，对于关系数据库模型设计来说非常荒谬。在多维结构中，“野餐篮与 YoKon 与一月份”相交是按照声明存在的。归根结底，DB2 OLAP 存储结构专门用来使设计人员能有效地处理此固有稀疏性，而 DB2 OLAP 数组的固有稀疏性是通过将维标为“稀疏”或“紧凑”进行管理的。

<p>紧凑/稀疏设置</p> <ul style="list-style-type: none"> • 将维标为稀疏或紧凑 • 这样就可创建唯一的 DB2 OLAP 存储结构(BLOCK 和 INDEX) <p><i>反映块内紧凑性</i></p> <p><i>反映索引内稀疏性</i></p>
--

紧凑维包含数据块。一般可以说，实施紧凑维的目的是反映(数据块中的)数据集的紧凑性，而实施稀疏维的目的是消除数据集的稀疏性：

只有在稀疏成员组合包含数据的情况下，才创建数据块，如图 12 所示。

但是，必须明白，块存储结构很可能会保留一定程度的数据库稀疏性(由每个紧凑维的所有已存储成员的笛卡尔积所定义)。也就是说，数据块中存在空单元。实际上，不可能完全从 DB2 OLAP 数据库中消除稀疏性。因此，目的是最大限度地降低稀疏性的影响。

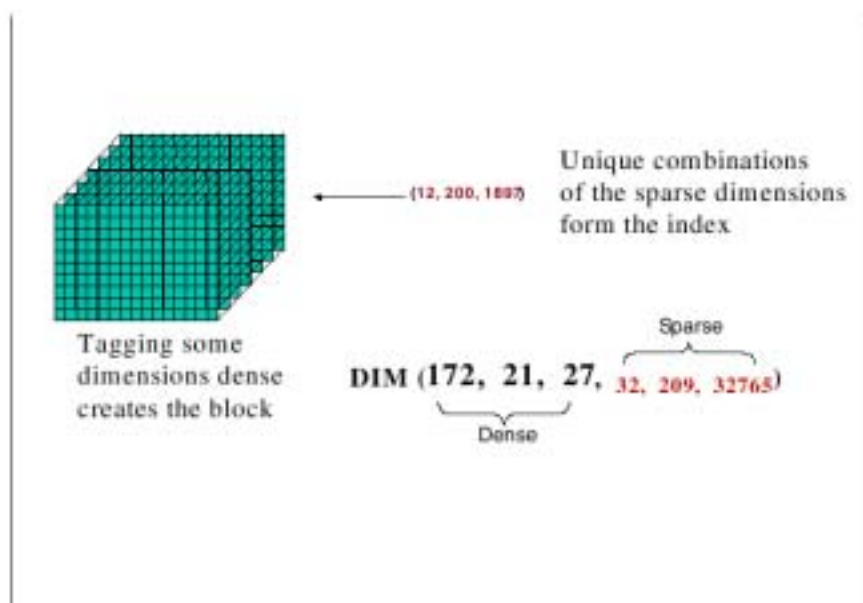


图 12.索引与块创建

重视 DB2 OLAP 环境中与商业相关的稀疏成员组合可驱动数据块创建这一事实，就可以实现稀疏性最小化。如果不存在稀疏组合，也就不存在商业数据，也就不会生成数据块。

矩阵的固有稀疏性是通过适当地标记维来处理的。也就是说，把彼此之间有最少交点的维标为稀疏。正确应用稀疏标记，就可以从数组中消除绝大多数交点。这就是使用 DB2 OLAP 实施大型业务模型的方式。

但是，已标为紧凑的稀疏维会导致数据库内出现大量空的也许是无意义的单元。有时，在商业活动并不是没有意义的商业事件时，分析师却需要了解它们。例如，我们五月份没有出售任何游泳衣这一事实，对于迈阿密的服装零售商来说，可能是非常重要的。

相反，当块结构内无法包含某个数据集的紧凑本质时，也就是说已将某个紧凑维标为稀疏时，这就会导致配置生成的数据块数量剧增。

多维数据库具有固定的维数目。因此，可以使用众多不同数据库配置(稀疏/紧凑设置)之一实施矩阵。形成数据块结构的紧凑维存储在磁盘上的 `ess*.pag` 文件内。标为稀疏的维形成索引结构，并存储在磁盘上的 `ess*.ind` 文件内(参见图 13)。索引项是指向数据块的指针，并包含单元(数组)地址的组成部分，而数据块被正确地视为实际存储数据的地方。

Index and Block Modularize the Matrix!				
Dim	Members	Type	Structure	Disk
Measures	172	DENSE	Block	} <code>ess*.pag</code>
Time	21	DENSE	Block	
Net	37	SPARSE	Index	} <code>ess*.ind</code>
Str	32	SPARSE	Index	
Prod	289	SPARSE	Index	
ORG	12765	SPARSE	Index	

- Block size: (21 * 172) 3612 cells * 8 bytes/cell = 28,896 bytes (~28.2 k)
 - Computer now able to move sections (block and index pages) of the matrix in and out of memory with efficiency

图 13. 如何创建矩阵

数据库设计员的目标：

数据库设计员的目标是双重的：

1. 尽可能少创建块。
2. 尽可能创建填充紧凑的块。

重复一下，实施**紧凑维**的目的是反映数据集的**紧凑性**，而实施**稀疏维**的目的是反映数据集的**稀疏性**或减少数据集稀疏性的影响。我们之所以说“减少”，是因为期望从 DB2 OLAP 矩阵中消除稀疏性根本是不现实的。同时还应牢记，如果块内不包含数据集的紧凑本质(紧凑维标为稀疏)，则会导致配置生成的块总数剧增。相反，多半空的数据块表明没有有效地从数据集中消除稀疏性。

4.4 大纲调节

调节大纲涉及以下各项：

- 维和商业逻辑
- 成员标记
- 合并类型
- 稀疏/紧凑方法

4.4.1 维数和商业逻辑

每个数据库模型的目的都是反映手头特定分析问题的商业逻辑。尽管针对商业算法解决方案不在本红皮书讨论范围之内，但是，出于设计大纲的需要，可以介绍某些一般概念：

- DB2 OLAP 值为永久值
 - a. 有多少维？
 - b. 维有多大(成员总数)？
 - c. 维有多深？
- 模型的行为方式如何反映上面三点之间的相互关系。

4.4.1.1 一般数据存储合并

DB2 OLAP 可通过(绝大多数)保持数据集值不变来优化其运行时性能。这有明显的批计算和磁盘存储含意。为有助于支付 I/O 的成本，*数据库设计员的目的是实施生成块数最少的数据库配置，而且这些块具有最高密度。*

4.4.1.2 一般大纲合并

主要注意事项是：

1. **有多少维？**实施者受可建模总数及开发模型硬件配置的限制。设法将常规维更改为属性。这会极大地减少大小和计算次数，因为属性计算速度，但通常对较小的数据库模型来说是有效的。
2. **维有多大？**成员总数最终确定数据集的稀疏性。
3. **维有多深？**数据库性能特征还因维层次结构深度的不同而不同。

设计员必须能够根据所开发的特定(实际)数据库的需要识别和调整数据库配置。必须更改稀疏/紧凑设置，以适应特定客户机需求，例如，支持成员计算或查询检索需求。*数据库配置的最佳化不一定取决于最佳稀疏/紧凑配置，而是视手头模型的具体需求而定。*几乎按照定义随时间增量更新的数据库排除了将时间维标为紧凑的可能性，即使数据库恰当地坚持数据集的密度。

4.4.1.3 何时添加维

初读起来，有意给数据库添加维的想法似乎有悖直觉。确实不需如此。在某种意义上，这只是基本设计技术的“变体”，当您发现重复标签时，这种技术会提示拆分维。

例如，建议在遇到“实际销售额”、“预算销售额”、“预测销售额”等度量时，最好添加一个包含以下三个成员的单独“方案”维：“实际”、“预算”和“预测”。现在，由于多维数据集的本质，这三个“方案”成员与度量维每个成员相交，而且，这会在所有其他维的所有成员之间发生。将“方案”添加为一个维，使用户能够执行对 OLAP 数据库的所有神奇操作：分片和划线、钻孔、旋转，等等。让我们把该示例提高一级。

应考虑所有当前数据库度量的需求，这些度量与前一时段的相同度量相关。这可能意味着会为每个现有度量创建新的度量，以便存储变化量。例如，“当前销售额”和“以前销售额”之间的差异。这会使“度量”维的大小实际增加一倍。

另一方面，图 14 显示“先前”维的内容。



图 14. 在大纲中添加维

注意此维的成员标记。维名称(“先前”)标有“只标签”。有一个称为“数据”的成员，它是为此维加载所有数据的目的点。在“数据”下面，有两个动态计算成员：“前月”和“前季”。每个动态成员都使用@PRIOR()函数的版本。因为这些成员已被添加到另一维，所以它们与“度量”维内的其他每个度量相交。这就是目前使用的多维范例。

请密切注意配置情况。我们已给此数据库添加了一个具有存储大小“1”的维。如果把此维标为紧凑维，我们就用当前块大小乘系数“1”。换句话说，块大小仍然相同！

我们确实增加了大纲的复杂性。同时增加了逻辑块的大小，而且这可能会对性能产生影响。但是，添加的一个维虽然未给存储结构增加什么，却相当于给模型功能注入了新的活力。实际上，可以毫不费劲、毫无代价地给“先前”维添加更多动态成员。

4.4.2 成员标记使用注意事项

成员注意事项

- 仅在必要时存储数据。
- 使用标记减少块大小：

只标签

动态计算成员

这里，使用成员标记成为良好实践原则：*仅在必要时存储数据*。例如，DB2 OLAP “动态计算”成员设计为仅在客户机请求时才返回数值。它们可以根据需要反映数据密度。也就是说，数值是在运行时动态计算的，因为用户期望返回一个值。

4.4.2.1 “只标签”和“动态计算”标记

如果矩阵中所有数据值都是预先计算的，唯一优化提示就是智能化使用“只标签”标记。“只标签”标记反映存储在商业范例内有意义值的业务需求。只标签标记反映了仅作为“标题”或“分组”成员工作的业务需求。其值是为组合成员提供占位符，而不是作为定量值。例如，“方案”就是一个在同一组中保存所有计划成员的父级。

图 15 中详细说明了块包含 $6 \times 4 \times 3 = 72$ 个单元。

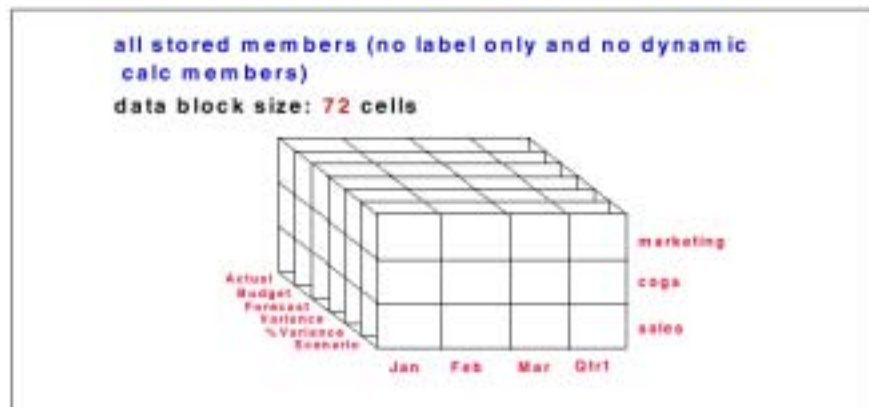


图 15. 合并成员：块

在上面的示例中，针对图 16 中的“方案”成员有效地利用“只标签”使块大小减小了 12/72 个单元。最终结果是有可能将总体数据库大小减小 16.7%。由于存在数据压缩，总体数据库大小的净减小很可能不等于 16.7%。

块大小 16.7%的减小量并不小，但是，只能针对大纲中的导航成员使用“只标签”。另一方面，使用动态计算成员可以极大地增强数据库设计人员减小数据库大小(存储值)的能力。

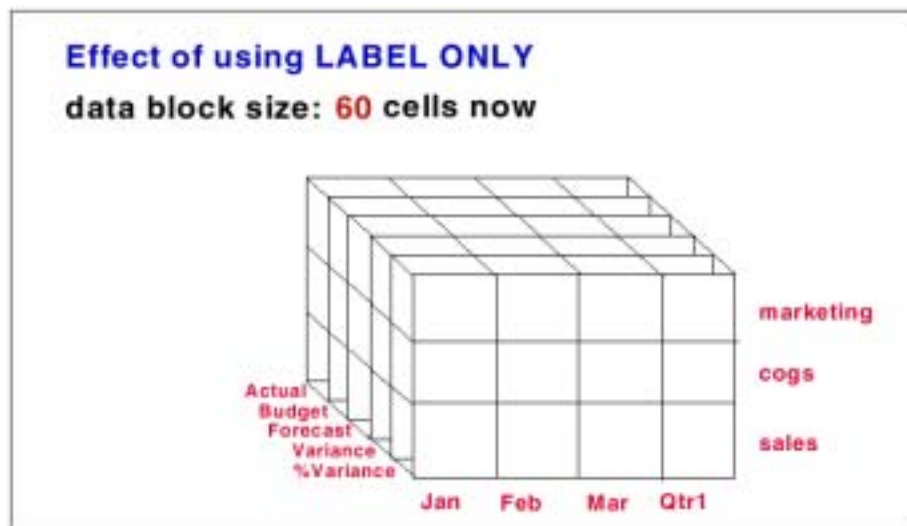


图16.合并成员：只标签

针对所有上层成员以及包含公式的成员实施示例数据块中的动态计算成员会从根本上改变矩阵的存储需求。图 17 中的块设计产生的数据块比原始块小 62.5%，比使用“只标签”实现的块大小小 56.5%！

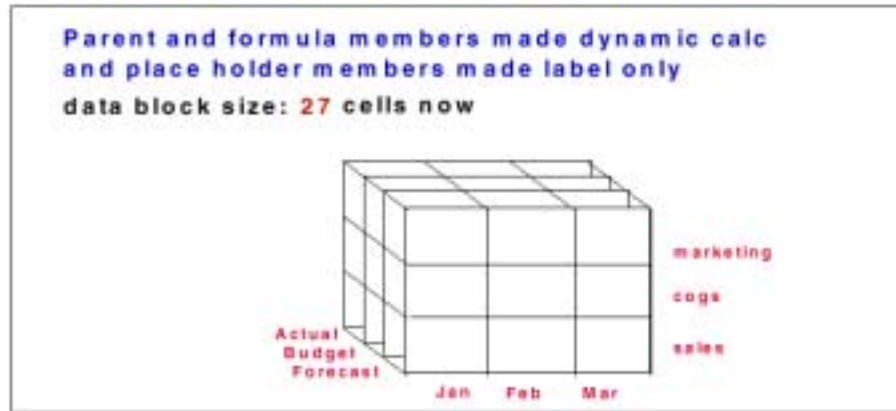


图 17.合并成员：动态计算

4.4.2.2 物理块和逻辑块

实施者必需学会在不同情况下以不同方式看待块的大小。图 18 中的物理块是为方便存储而优化的块。在应用程序管理器中，它用以字节计算的块大小统计方法表示，而在 GETDBSTATS 包含的单元中以实际块大小表示。如果数据库配置正确，物理块就是批计算中 DB2 OLAP 检索的块。

图 18 中的逻辑块是在工作内存(**不是** DB2 OLAP 高速缓存)中完全展开的块，并包括 RAM 中用于所有动态计算单元的存储空间。请记住，DB2 OLAP 只计算请求的动态值。但是，它必须在工作内存(而不是高速缓存)中其他某个地方为那些值保留空间。然而，应当清楚，在批计算过程中，DB2OLAP 计算器可能(无意地和不适当地)需要动态地占用数据高速缓存中的空间，以便完全展开数据块。

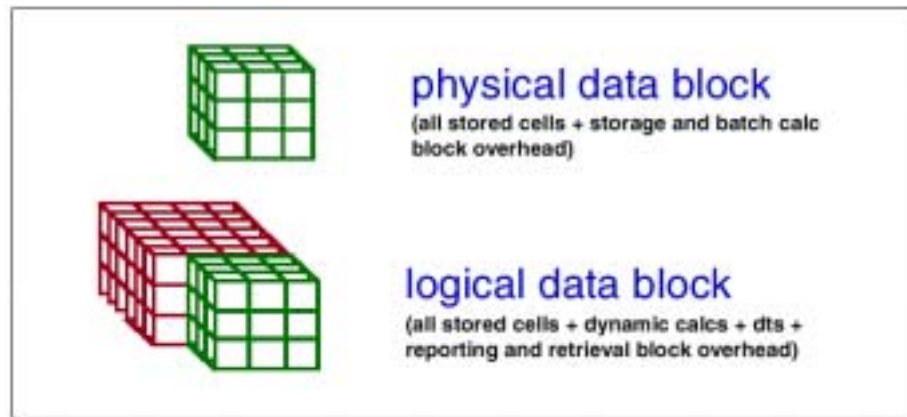


图 18. 两种块类型？何时结束？

批计算中涉及动态计算，比如，在存储成员值依赖于动态计算值的情况下。

请看图 19 中的大纲片段。注意成员“\$Gross Sales”为存储成员，而且其值取决于动态计算的“\$Sales”成员。此配置的大纲会使计算器在批计算中，在数据高速缓存内，把数据块完全展开到其逻辑大小。

这会导致减小数据高速缓存的大小。



图 19. 大纲片段

管理员可能已把数据高速缓存配置为保存 100 个物理数据块，进行批计算，但由于计算器在运行时把块在数据高速缓存中完全展开，以便执行这些动态计算，所以可能只剩下 2 到 3 个物理数据块空间！这会不必要地延长批计算过程。

4.4.2.3 注意

数据块是 DB2 OLAP 中的基本数据存储结构，而数据库配置最终会受到块大小的制约。与早期版本相比，7 版本与更新版本更是如此。即 5 版本以前的 Hyperion Essbase OLAP Server 版本。

4.4.2.4 什么样的块大小最佳？

有些经过精心调节的以及响应性良好的生产数据库具有从几百字节到一兆字节不等的数据库块大小。但是，归根结底，最佳块大小是可提供最佳性能的块大小。而且，数据库实施者需要充分了解所有标准，以此确定数据库性能是否在可接受的范围内。

已隔离最佳大小的标志可能是：已实现最快批计算时间。我们相信生成最少最紧凑数据库块的数据配置与最快计算时间有着很大关系。我们将在后面的内容中提供一种高效地确定此配置的方法。但是，数据库设计员必须明白在调优时，其他(过程)问题有时可能比稀疏/紧凑配置更重要。

有些数据库具有根据用户查询需求、特定复杂计算需求或二者兼有而优化的块配置。数据库调节及优化与其说是科学，道不如说是艺术，这话可能对，也可能不对。但是，我们坚信清楚地理解 DB2 OLAP 存储结构的本质是数据库调节的基础。

最佳块大小是针对模型的。因此，最佳做法不是向开发人员提供一个神话般块大小范围的目标，而是提供一种能够迅速隔离手头具体数据库最佳块大小的方法。

动态成员实践

- 使用动态成员将更大数据密度包含在数据块内
- 理由

动态成员可减小块大小

创造将其他维包含在块结构中的机会

增加了不同块配置的数目

块配置仍然受总体块大小限制！

4.4.2.5 用于优化的“动态计算”标记惯例

可以考虑实施多种多样的数据库(块)配置，因为可以实施动态计算的成员。使用动态成员终止批计算中某些成员计算的能力，可使设计人员有可能在数据块中包含更大数据集密度。

要清楚，由于 DB2 OLAP 必须动态产生数据值，所以仍会存在块大小的实际限制。归根结底，最佳数据库设计将高效地协调以下各项之间存在的关系：

1. 块大小
2. 动态检索的数目和性质
3. 硬件配置

4.4.3 合并类型注意事项

我们至少可以想出三种一般 DB2 OLAP 数据库合并类型。各种类型在 DB2 OLAP 存储结构上下文内都有自己的性能特征。

三种合并类型

- 自然合并：
 - 没有公式
 - 有紧凑成员公式
- 特案合并
 - 需要固定或稀疏成员公式的大纲

1. 最佳配置

- 一元运算符反映所有商业逻辑
- 根据默认计算算法以大纲顺序计算数据库
- 块创建及合并是常规且可预测的

2. 接近最佳的配置

- 关于紧凑成员的公式反映某些商业逻辑
- 交叉维引用是在块内解决的，执行方式与在大纲中相同
- 根据默认计算算法以大纲顺序计算数据库
- 块创建及合并是常规且可预测的

3. 次最佳配置

- 关于稀疏成员的公式
- 包含对稀疏成员的交叉块引用
- 块创建及合并不是常规的，也不可预测

最佳合并充分利用 DB2 OLAP 工程师实施的编程逻辑。所有商业逻辑都通过大纲中的(一元)层次关系反映。块创建是常规且可预测的，而且可轻松地与 DB2 OLAP 缓冲器协调，以实现最佳硬件利用率。CALC ALL 启动合并，而由内置到计算器引擎的内部算法高效地生成数据值。

图 20 显示一个与一元运算符自然合并的示例。

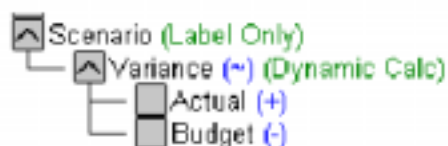


图 20. 与一元运算符自然合并的示例

近乎最佳的合并反映更为复杂的情形，其中，无法将商业逻辑嵌入一元层次结构，并且商业公式是必需的。所有公式都能在数据块内解答。因此，最佳合并和近乎最佳的合并之间的主要区别属于处理成员公式的额外开销。因为所有公式都是块内公式(不涉及多个块)，所以块创建依然是常规的、可预测的，以及高效的。

图 21 显示一个与公式自然合并的示例。

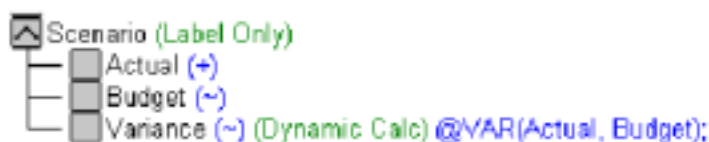


图 21. 与公式自然合并的示例

次最佳配置包含 FIX 语句、块间计算或二者兼有。启动稀疏交叉维引用的公式可以包含在大纲内，也可以包含在计算脚本内，但是性能与稀疏成员引用数目和程度直接相关。

图 22 显示用于稀疏维“分配到”公式的示例。



图 22. 具有公式的稀疏成员示例

尝试通过将所有公式都包含在大纲之内、把所有公式都放在数据块内，以及解决块内交叉维引用，以确保自然合并。设计人员应随时在必要时实施自然(默认计算)数据库合并，充分利用 DB2 OLAP 存储结构。

4.4.4 稀疏 / 紧凑方法

数据库设计员的双重目标(创建数量尽可能少、密度尽可能大的数据块)提示我们，数据块的密度与生成块的数量之间存在某种关系。事实上，在任何 DB2 OLAP 数据库内，数据块密度与创建的块数之间的关系可以用数学公式表示为：

(数据集密度)=(块密度) / (块数)或 $Dd = Bd/Nb$

开发人员的双重目的 *不是* 创建最少的块数，*也不是* 创建最紧凑的块，而是 *创建块数最少、密度最大的块*。利用反映此关系的数据集密度比率，我们可以简化确定最佳稀疏 / 紧凑设置的任务，以寻找最高比率。

很明显，比率本身是很普通的数字。可生成 3500 万个块的模型，其 1% 块密度实际上产生比例很小的数据集密度度量。这些比率不会因数据库的不同而不同。*它们仅与在不同稀疏 / 紧凑配置上使用相同源数据为同一数据库所派生的相同值有关！*

提示

- 反映块中的数据密度、索引中的数据稀疏性
创建密度尽可能高、数量尽可能少的块
- 块密度(Bd)与块数(Nb)之间存在某种关系(数据集密度 Dd)，这种关系可用以下比率表示：

$$Dd=Bd/Nb$$

任何给定的数据库实例都有一组包含数据的固定交点，而任何数据库实例的块密度与块数之间也存在一组固定比率(“数据库实例”是指在某个时间点导入的数据库)。

块密度度量升高时，总比率也会升高；块数度量降低时，总比率还会升高。最高比率反映同一数据库实例的所有数据库配置之间，块密度和块数之间的最佳关系。或者，如果您愿意的话，最高比率指向生成数量最少、密度最高的数据块的配置(参见图 23)。

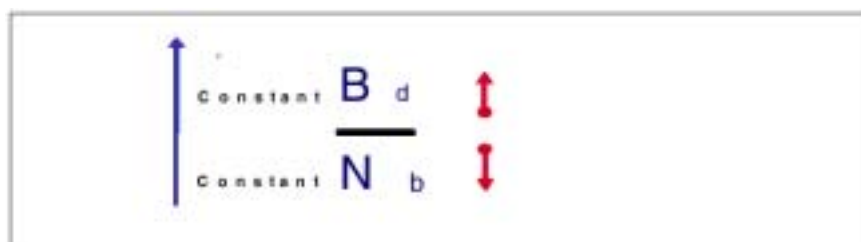


图 23：观察与方法

结合前面讨论的概念，我们可以对稀疏 / 紧凑方法进行详细描述。该方法首先假定已将数据库大纲进行设计，使其能够准确反映业务逻辑。

完成各种块配置：

1. 将所有公式都放在大纲内。
2. 将所有上层成员 / 公式放在数据块“动态计算非存储区”内。
3. 将数据加载到不同的稀疏 / 紧凑配置，以创建 Bd/Nb 值表。

紧凑 / 稀疏设置

大纲设计解决了业务逻辑和维问题之后

反映块中的数据密度、索引中的数据稀疏性

- *组合指导原则，以创建紧凑 / 稀疏方法*
 - 把公式放在大纲中
 - 把块中的成员标记为动态
 - 反映块中的密度、索引中的稀疏性
 - 查找高比率 Bd/Nb

4.4.4.1 方法详述

DB2OLAP 开发人员一直在实施此方法的一个版本。也就是说，它们反复测试了不同配置，并能够通过测量计算时间确定一种最佳配置。但也有可能无法确定。设计员通常寻找可满足批处理窗口的配置。事实上，可能有不只一种可以在批窗口内计算数据库的数据库配置。另一方面，即使设计员没有实施此配置，该方法仍将显示什么是最佳配置。

注意：已开发出一种配置实用程序，您可从以下网站免费下载：

www.essbase.com

www.ibm.com

www.olapunderground.com

以这种方式确定最佳配置的不适宜的先决条件是：预测(开发)服务器专用情况通常十分耗时。

希望测试最高效模型的开发人员会在一种受控环境中运行性能测试。他们希望尽可能消除资源争用现象，以便准确确定哪种配置占用计算时间。这通常不仅意味着在测试计算持续期间其他服务器活动受到了限制，而且还会导致发生十分耗时的重复测试。

测试的各种数据库配置都必须运行至少与最短(或迄今为止最佳)测试计算时间一样长的时间。数据库配置测试极度占用资源,而且非常耗时。这就说明了开发人员为何满足于采用不一定是最佳的第一种配置的原因,只要该配置能适应批处理窗口即可。

为了消除该过程,已开发出一种方法,可在最短时间内生成最佳数据库配置指标。

将包含子女或公式的紧凑成员设置为“动态计算”

将这些成员标记为“动态计算非存储区”。“动态计算和存储区”标记无助于降低数据块的存储需求,实际上是为稀疏成员标记保留的。

所有 DB2 OLAP 数据库都会受到数据块大小的限制。我们觉得没有理由排除这样一种可能性:存在某个数据集,其稀疏维能够定义仅仅由于太大而无法实施的数据块。计算的比率可把数据与其存储结构联系起来。块大小是一种硬件(和软件)限制,并且尽管对于数据库配置至关重要,但并不属于矩阵本身。

通过将数组分析为两个数组成分(稀疏和紧凑),Arbor Essbase 的创建者确实发明了一种聪明方法,以模块化矩阵并在相对较小的计算机硬件上实施巨大数组。事实上,可以把设计目的视为确定何种块大小最有利于最小化 I/O。

然而,如果根据最佳大小的假设寻找某种块大小,就有可能以丧失某种更为重要的东西而告终。此方法根据一组(子集)可能的数据库配置计算比率。最终将得出一个数值表,可以通过图形方式轻松地表示其中的数值。

该图(或数值表)准确地描述了模型(数组)的紧凑 / 稀疏特征,并隐喻性地描述了给定数据库的性能特征。在复杂模型中,最终选择哪种配置取决于对所有因素的综合考虑,而块大小、过程以及计算需求可能是其中最重要的三个因素。

通过更为自由地计算比率,我们希望数据库设计员能够学到更多有关数据知识。但愿此信息具有启发价值。因此,我们没有设定块大小限制,而是决定利用这种方法确定几乎任何配置的相对重要性。

此方法的当前版本实际上是各种数据库配置的例证，正因为此，它才能在可用 RAM 世界中存在下来。因此，不可能测试内存需求超出最大块大小的数据块配置 - DB2 OLAP 能够在任何给定机器上访问此块大小。无需实际创建数据块即可确定块大小的能力最终将使此方法摆脱这种限制，尽管不这样做的实际理由(大型维具有内在稀疏性)可能无法保证这种功能。

在块中设置“动态计算非存储区”成员的方法已成为块设计的*实际标准*。将所有可能成员设置为动态会减小块大小。这可能会对批计算时间产生极大影响。但是，从这种利用动态成员的方法中获得的最大好处可能是：它使设计人员能够将维包含在块内，而在其他情况下由于块大小的限制则无法做到这一点。归根结底，此方法可以在数据块内包含更大数据密度。

将真实和完全有代表性的数据加载到数据库

这一方法的准确性关键取决于是否能够在各种数据库维之间加载真正的生产数据。我们需要在紧凑维之间迁移所有单元，以确定 DB2 OLAP 报表的平均块密度是否具有代表性。同样地，我们需要在稀疏维之间导入所有点，以确保全部块创建同样具有代表性。由于我们重复了多种配置，因而确实需要有完全代表性的数据。

但是，请注意，我们可以在一定限度内假称如此。当且仅当已知某个维不是稀疏维或紧凑维，而且不进行相反测试时，数据才无需具有代表性。

应考虑包含 12 个月和 4 个季度的块。如果确信将把时间标记为*紧凑*，那么，随时间把数据只加载到某个成员子集(或 1 个成员)将会对所有已测试配置的块密度产生同样的影响。因此，如果时间总是该块的组成部分，仅加载一个月将会为所有其他测试配置的块密度产生可靠度量。另一方面，如果仅将数据加载到一个月，并且将时间标记为*稀疏*，则会极大地低估生成的数据块的数量，并且结果也会出现偏差。

从数据库统计报表中检索平均块密度度量

下面是数据库信息统计报表表的对于平均块密度的计算情况：块总数分为 100 个相等的部分。按百分之一的比例抽取密度样本，并从 100 个块中计算平均密度。

为了最大限度地减少计算比率的时间，我们假定数据加载时的平均块密度将代表完全计算的模型的块密度。我们假定这是针对某个具体数据库实例。也就是说，我们的假设基于这样一个需求：为该大纲将同样的数据集加载到可能的稀疏 / 紧凑配置间的同一大纲内。

块密度假设

数据加载时的平均块密度代表与其他稀疏 / 紧凑配置相关的计算进行完之后的总块大小。

从数据库统计报表中检索生成的块数

为最大限度地减少生成比率的时间，我们假定数据加载时创建的平均数据块数目代表完全计算的模型的总块数。假定这是针对某个具体数据库实例。也就是说，我们的假设基于这样一个需求：为该大纲将同样的数据集加载到可能的稀疏 / 紧凑配置间的同一大纲内。

块数假设

数据加载时生成的块数代表与其他稀疏 / 紧凑配置相关的计算进行完之后，生成的总块数。

我们不想说 DB2OLAP 中创建的每个数据库应用程序都将符合这些假设。到目前为止，尚未确定此方法的错误率。但是，我们不想提议大多数模型都符合这些假设。这就是迄今为止测试所揭示的信息。

4.4.4.2 非最佳实施？

上面描述的度量实际上表明，在给定模型数据内，数据是如何在数组间置换的。上述度量同时可用作数据库性能特征的一项指标，这一事实并非巧合。最大限度地从模型中消除稀疏性的设计可导致一种磁盘存储需求最少的设计。从这一观点来看，这是对 I/O 需求最少的配置。

毫无疑问，并不总是需要实施最佳数组配置。下面将详细讲述其中一些数组配置。

紧凑重构和增量数据加载

有些过程需要增量数据加载。通过考察按时间发生的增量数据加载，就可以最好地了解这方面的典型示例。例如，人们可能会想到，由于此方法认为应把时间维视为紧凑维的组成部分，开发人员就会实施一种把时间标记为稀疏的设计，以支持按时间增量加载数据。

的确，经历常规成员插入的维是稀疏标记的强有力候选者，即使这些维可能是数据集密度的组成部分。如果刷新过程不能实现数据库的完全重新加载和重新计算，那么，向紧凑维中添加成员就会需要一种完全紧凑的重新构造。设计员会认真地考虑把这样的维设置为稀疏，以避免通常需要经历的紧凑重新构造。

总块大小

可以想象，此方法将指向这样一种配置：需求必须实施不受所用硬件基础设施支持的块大小。事实上，这准确地反映了 Hyperion Essbase OLAP Server 版本(5 版本中动态计算成员版本之前的版本)的情况。

4.5 数据库计算注意事项

本节讨论计算 DB2 OLAP 数据库时的主要注意事项：

1. 使用 Set NOTICE 命令
2. 使用动态计算
3. 聚焦计算

4.5.1 使用SET NOTICE命令

运行开发或测试环境计算时，请使用 SET NOTICE 命令。此命令提供一种方法，即在对脚本或数据库设置进行更改时就可以比较性能，而无需等待运行整个计算。通过跟踪计算到达同一通知点所花费的时间，您就能推断出更改是否已成功地缩短或延长了计算过程。有关使用 SET NOTICE 命令的详细信息，请参阅 DB2 OLAP 文档。

4.5.2 经过检查的动态计算

本节将讨论与动态计算相关的各种注意事项。

“动态计算”概览

- 动态计算
 - 上层紧凑成员
 - 包含公式的紧凑成员
- “动态计算”(和“存储区”)与稀疏成员
 - 包含小型扇出(<7)的稀疏成员
 - 记住把多个(稀疏)维的多个成员标记为动态的交叉积(笛卡尔积)结果
- 练习时一定要谨慎！

4.5.2.1 针对紧凑维的动态计算

将数据块内的每个成员都标记为“动态计算非存储区”已成为一种标准做法。这包括所有包含公式的成员以及任何包含下层成员的成员。这种做法可确保仅存储必须存储的值。

但是，有时由于“动态计算”成员交叉引用的复杂性，而无法正确地解出最终的“动态计算”值。在这样的情况下，如果不能得出正确结果，请从公式中删除“动态计算”成员名称，并对成员公式本身中的引用进行实际硬编码。

忽略错误逻辑，请看一下图 24 中的示例，其中包含突出显示的示例成员。



图 24. 具有动态依赖性的大纲示例

DynamicDependenciesExample2 上的公式中已完全解 *DynamicDependenciesExample1* 上公式中的每个动态成员。请将简明公式($\text{Profit} * (\text{Margin} \% \text{TotalExpenses}) / \text{Margin} \%$)与显式公式

$((\text{Sales} + \text{COGS}) - \text{Marketing} + \text{Payroll} + \text{Misc}) * (\text{Sales} + \text{COGS}) / (\text{Marketing} + \text{Payroll} + \text{Misc}) / (\text{Margin} \% \text{Sales})$ 进行比较。尽管显式引用不太简洁，但它们肯定能正确解出值。

CALC DIM 加 AGG 比 CALC ALL

针对数据块实施“动态计算非存储区”策略的数据库还可能受益于以下情况。为了概括此方法，所有上层成员以及包含公式的成员都标记为动态。

发出与 **CALC ALL** 命令相反的 **CALC DIM** (**AllSparseDimensionsList**) 或 **AGG** (**AllSparseDimensionsList**) 命令，这应该是生成上层数据块的一种较为高效的方法。**CALC DIM**(**AllSparseDimensionsList**) 命令将忽略紧凑维，并直接用所有数据块导入数据库。**AGG**(**AllSparseDimensionsList**) 也会产生同样的结果，但会忽略检查稀疏成员公式是必需的。理论上，**AGG** 是产生上层块的最高效的方法。

应考虑到成员计算仅在级别 0 数据上执行，性能提高的程度将与级别 0 与上层数据块之比数字有着直接的关系(参见图 25)。

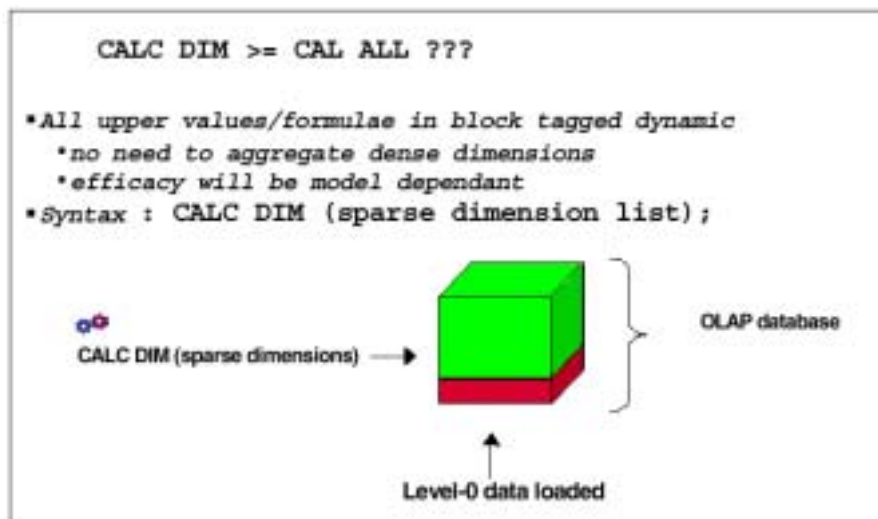


图25.CALC DIM 比 CAL ALL

检查一下计算同一个数据库的两种不同方法的应用程序日志输入。CALC ALL 命令会导致以下输入：

Total Block Created: [0.0000e+000] Blocks
 Sparse Calculations: [1.9700e+002] Writes and [8.2900e+002] Reads
Dense Calculations: [1.7700e+002] Writes and [1.7700e+002] Reads
 Sparse Calculations: [3.3096e+004] Cells
 Dense Calculations: [0.0000e+000] Cells

现在，把这些输入与同一数据库的某个 AGG 的应用程序日志输入进行比较：

Total Block Created: [1.9700e+002] Blocks
 Sparse Calculations: [1.9700e+002] Writes and [6.3200e+002] Reads
Dense Calculations: [0.0000e+000] Writes and [0.0000e+000] Reads
 Sparse Calculations: [3.3096e+004] Cells
 Dense Calculations: [0.0000e+000] Cells

依靠实施“动态计算”标记，AGG 的数字结果与 CALCALL 完全相同。但是，(而且仍然依靠“动态计算”标记实施的)计算时间比 AGG 的计算时间快~17%！上面突出显示的应用程序日志摘录似乎反映出 AGG 的额外开销比 CALC ALL 小。

4.5.2.2 针对稀疏维的动态计算

您可以考虑在只有少数子成员的上层稀疏维成员上使用“动态计算”标记。如果您在上层使用这些标记，不要使用太多“动态计算”标记。这只能充分减小计算时间和数据库大小。例如，如果您将三个稀疏维的顶部标记上“动态计算”，而且每个维有 10 个子维，那么，要检索数据库的总数，服务器就必须检索 $10 \times 10 \times 10 = 1000$ 个块，并把它们相加。这会极大地减缓检索时间。您可以把这样的成员标记为“动态计算和存储区”，因此而需要支付随后检索同一成员的费用。

总之，在一个数组中，维就是维，成员就是成员。而在 DB2 OLAP 中，维并非都是相同的。维要么标记为稀疏，要么标记为紧凑。当动态计算反映此 DB2 OLAP 事实时，动态计算就获得最佳配置。

4.5.3 聚焦计算

每次仅计算数据库的一个子集，这是一项数据库需求。聚焦计算是通过三种方法实现的：FIX 命令、IF 逻辑构造或使用交叉维运算符。

主要建议应该是*隔离数据库的特定节，以提高计算效率*：

- FIX...ENDFIX：聚焦块子集
- IF...ELSE...ELSEIF...ENDIF：微调逻辑
- CROSS DIM OPERATOR：单元级别的粒度

4.5.3.1 FIX和IF

在计算中使用 FIX 会使计算脚本只影响数据库的一个子集(参见图 26)。

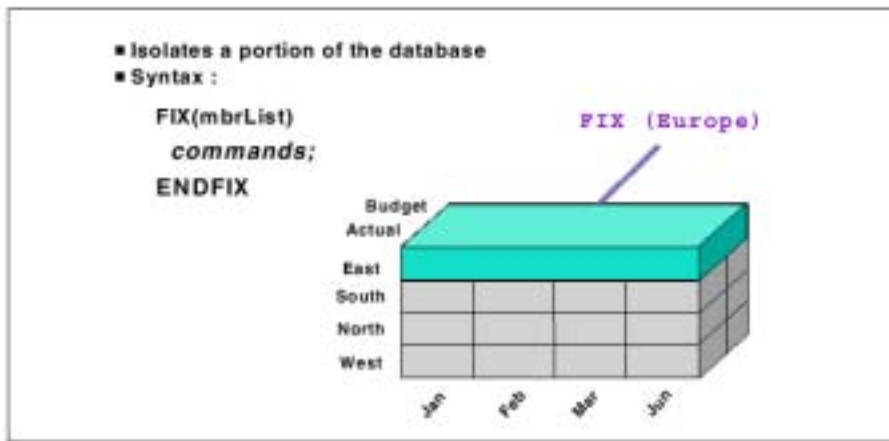


图26.使用 FIX 和 ENDFIX 进行聚焦计算

将数据块与索引隔离开来的最佳方法是使用 FIX。“关注”稀疏成员(如图 27 所示)可使计算器能够通过单遍索引分清楚哪些块是处理所必需的。

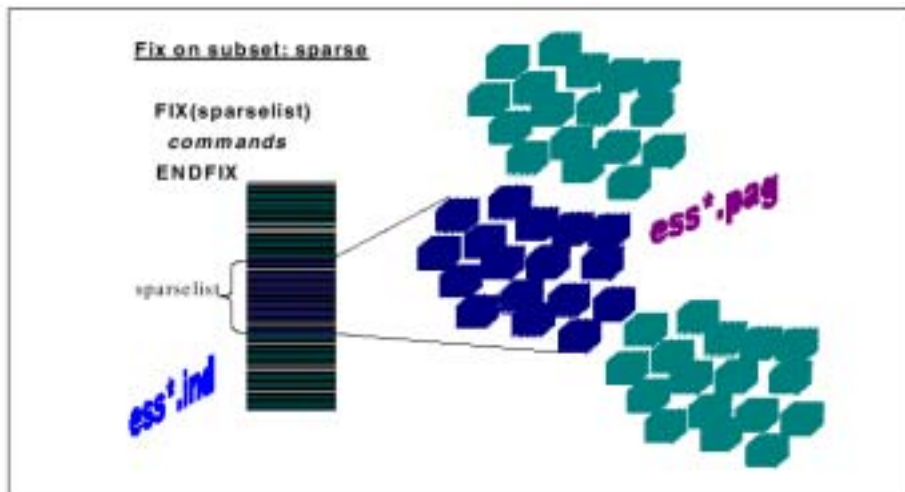


图27.关注稀疏维

只关注一个紧凑维(如图 28 所示)不会取数据库的子集,因为数据库中的每个块都包含必需成员。有时,仅关注一个紧凑维的活动是一个必需的计算需求。

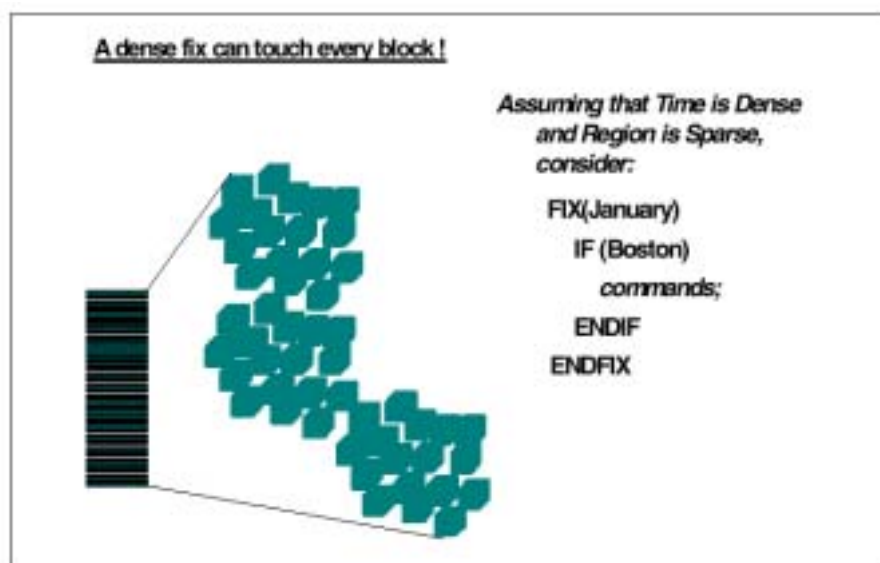


图 28.关注密度维

也可以使用逻辑“IF...ELSE...”构造隔离要计算的成员。只有无法在数据上使用 FIX 的情况下,才应使用此构造。也就是说,只有在必须以其他方式使用多个 FIX 语句(这会导致在索引中进行多遍计算)的情况下,才使用 IF。

数组数据库中的计算目的与关系数据库中的计算目的没有什么区别,我们对存储结构进行计算的遍数尽可能少些。归根结底,哪种方法在计算过程中证明是最有效的方法,哪种方法就是应该使用的方法。

4.5.3.2 聚焦计算概要

尽可能使用 FIX:

- 使用 FIX 关注数据库内的子集
- 使用 IF 解析不能使用 FIX 地方的逻辑
 - 您可以在块中使用 FIX
 - 您可以在块的子集上使用 IF

- 组合使用 FIX 和 IF，可最高效地隔离要计算的数据

4.5.3.3 特案计算需求

有些数据库方案排除了在上述数据块内实施“动态计算非存储区”。在数据计算(度量之间存在依赖性)的复杂性使所有公式无法成为“动态计算非存储区”的模型中，应遵循与默认计算顺序相关的规则，以确保进行最佳数据库计算。《OLAP 数据库管理员指南》的“定义计算顺序”一节中详细介绍这些规则。

一般计算脚本概念是：

- 紧凑计算先于稀疏计算
 - 普遍过时 - 所有上层值都是动态的，但是，必要时，请用聚集的值迁移单元
- 两遍计算
 - 动态计算
 - 不是动态时，确保进行单遍计算

4.6 性能调节：缓冲器

DB2 OLAP 服务器的总体性能与许多不同因素有关。要最佳地配置 DB2 OLAP 缓冲器，DB2 OLAP 数据库管理员不必是操作系统专家。但是，要牢记操作系统有其自己的内存需求，必须在这些限制范围内设置 DB2 OLAP 缓冲器。

缓冲器管理：基础

- 与内存相关的总体性能
 - *DB2 OLAP 缓冲器需求*
 - 多个应用程序
 - 多个数据库
 - 逻辑块
 - *操作系统内存需求*
 - 虚拟内存

在总资源需求超过总资源可用性的多应用环境中，如何分配资源并非易事。也许资源不足的 DB2 OLAP 环境无法进行调节，但至少证明管理期望值的需要与管理内存的需要一样大。

4.6.1 DB2 OLAP高速缓存配置原则

系统而科学地确定 DB2 OLAP 缓冲器的大小是设有替代方法的。下面将介绍一些有助于为 DB2 OLAP 缓冲器的配置制定总体策略的原则。

4.6.1.1 索引、数据和物理(文件系统)高速缓存

数据高速缓存是 RAM 中的一个扩展的物理块数据页存储库，而物理(文件系统)高速缓存则是 RAM 中的一个压缩物理数据块页存储库。索引高速缓存是 RAM 中的一个索引地址页存储库。DB2OLAP 存储管理器(或内核)通过为诸如计算机和报表模块这样的服务器功能层提供数据高速缓存寻址，管理索引和数据高速缓存(参见图 29)。

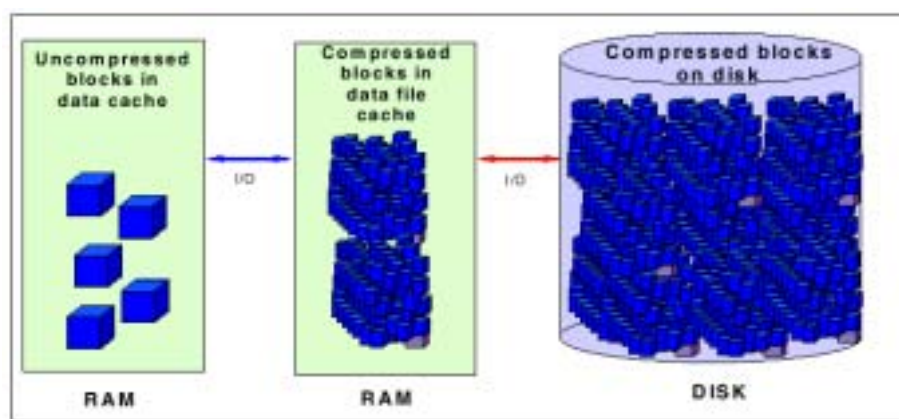


图 29.数据、OS 高速缓存和磁盘上数据

DB2 OLAP 数据高速缓存中的数据块没有压缩，而物理数据高速缓存(或 OS 文件系统高速缓存)中的数据块则是经过压缩的。由于数据高速缓存中没有进行数据块压缩，增加数据高速缓存存储器最终会增加解析从磁盘抽取数据的可能性，因为这会减少内存中可以保存的块总数。

换句话说，数据高速缓存以相对较低的值实现优化。要反映该事实，请使物理数据高速缓存(或 OS 文件系统高速缓存)优先于数据高速缓存。

每次块抽取都需要访问相应索引页。索引项(在 7.1 版本中，每个索引项 112 字节)通常比数据块小得多。每个索引页包含多个块地址。对于索引高速缓存，可以采取与我们上面对数据高速缓存采取相似的方法，我们建议使索引高速缓存优先于物理数据高速缓存(或 OS 文件系统高速缓存)，因为可以在存储页上按比例存放的块地址要比块多。

协调这三种高速缓存之间的内存分配是一种平衡方法。这不是火箭科学。但系统地确定这些设置所涉及的科学更像我们在高中一年级曾经学过的科学：除一个常量之外，保留所有变量，并观察其结果。

4.6.1.2 成批与生产设置比较

批计算时的缓冲器内存需求可以不同于用户查询时的缓冲器需求。例如，当 500 个用户通过特定分析随机请求数据块时，“自然”或最佳批计算(定期或可预测地迁移具有数据块的数据库)在所有三种缓冲器中需要的内存要比同一数据库在计算之后需要的内存少得多。前一种情况下，索引高速缓存上的命中率有望非常接近 100%，而在后一种情况下，根本不可能接近这个数字。

4.6.2 计算器高速缓存

计算器高速缓存不是 RAM 中的数据存储库，而是一个路线图，计算器引擎使用它来跟踪块创建和合并(请参阅《OLAP 数据库管理员指南》中有关计算器高速缓存配置的文档)。

DB2OLAP 文档指出，实现多个位图是一种可取的做法，而且与减少计算时间有着密切关系。

但是，通过计算器高速缓存的搜索机制是有顺序的，并且没有索引。尽管一般情况下需求在此高速缓存内实现多位图表示，但有时由于锚定维的稀疏性非常大，以至于通过计算器高速缓存进行搜索比调用内核对系统产生更高的开销。

4.6.2.1 多个位图上的单个位图

在这样的情况下，建议最好在大纲中更改稀疏维的声明(即更改其顺序)，以作为检查是否减少批计算时间的测试在多个稀疏维之间实现一种单一位图表示。

请查看图 30 中的大纲片段。Measures、Page_ID 和 Date 都是紧凑维。

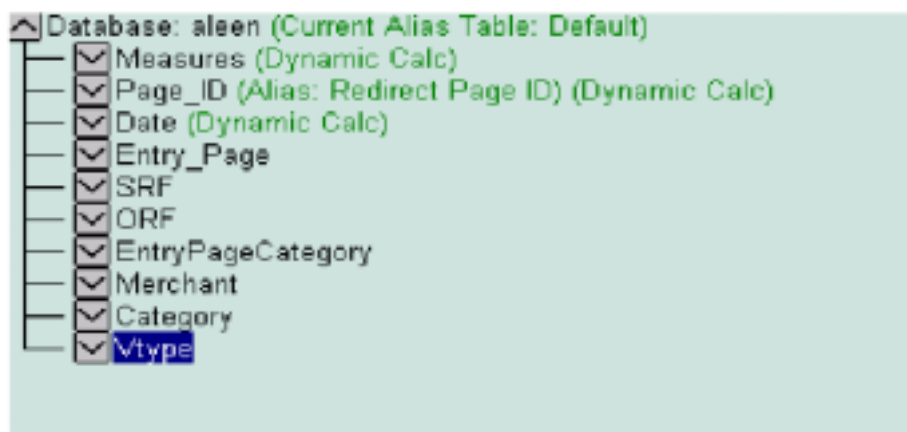


图 30.大纲片段

下面的日志片段表明 DB2 OLAP 在最后四个稀疏维(EntryPageCategory、Merchant、Category 和 Vtype)上建立单个位图锚点：

Maximum Number of Lock Blocks: [100] Blocks

Completion Notice Messages: [Disabled]

Calculations On Updated Blocks Only: [Enabled]

Clear Update Status After Full Calculations: [Enabled]

Calculator Cache With Single Bitmap For: [EntryPageCategory]

意思是，访问 OLAP Server 内核(在四维锚定维之间)比通过一个大而非常稀疏导入的多位图计算高速缓存图像连续搜索的效率更高。

4.7 数据压缩

数据压缩类型可提高系统的总体性能。压缩数据块越小，在磁盘和内存之间的移动速度越快。选择使用哪种压缩算法与块密度有关。DB2 OLAP 文档建议，当平均块密度大约或低于 2-3%时，行程长度编码(RLE)压缩比默认位图压缩更有效。磁盘上的块越小，每次磁盘所读取的块就越多，并且可以在 RAM 中保留的压缩块也就越多。

可以为每个数据库设置数据压缩类型。一般说来，位图压缩可为数据块和重复块值的 RLE 压缩图中的每个单元保留一个位。图 31 和图 32 提供一些有关各种算法如何压缩同一数据块的详细信息。但问题是，最有效的数据与提高 I/O 性能有关。

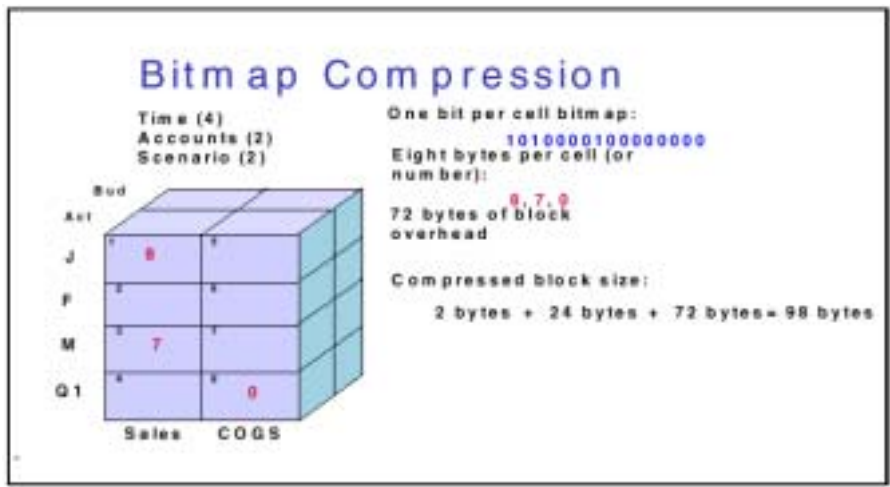


图 31. 位图压缩

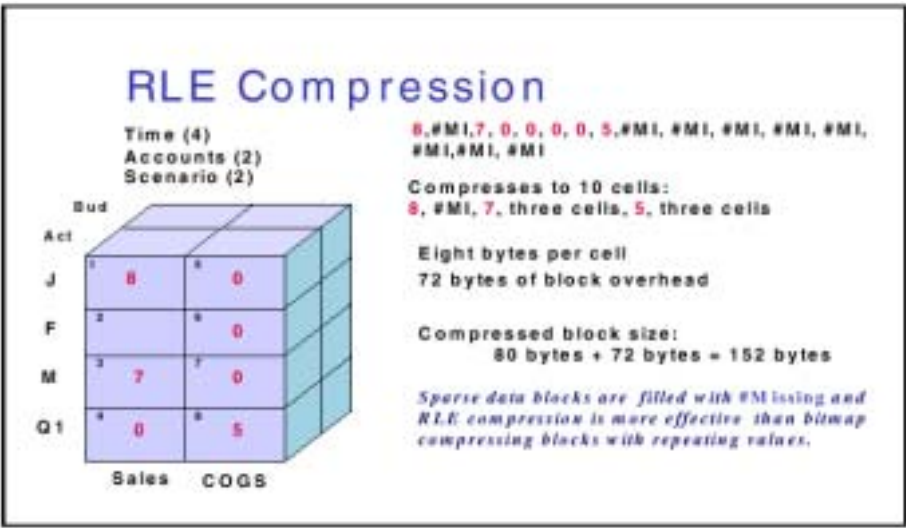


图32.RLE 压缩

4.7.1 RLE压缩和数组说明

图 33 和图 34 以图形方式说明数组中紧凑维的顺序可能如何影响 RLE 压缩。在以下图例中，您可以形象化地看出为什么必须通过算法映射各行。在后续月份中，每个行都是按值(X)和#Missing 以相同方式映射的。

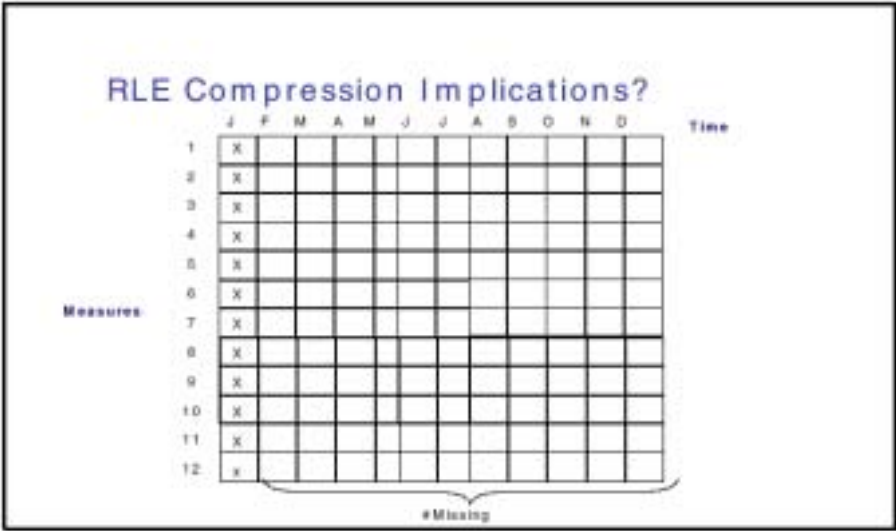


图 33.RLE 压缩含义(1)

注意：变更紧凑密度的顺序可创造利用 RLE 压缩性质的机会。在下一个声明中，只需映射第一行单元，并且所有后续行都可以压缩成 RLE 单语句：开始地址、结束地址和 #Missing。

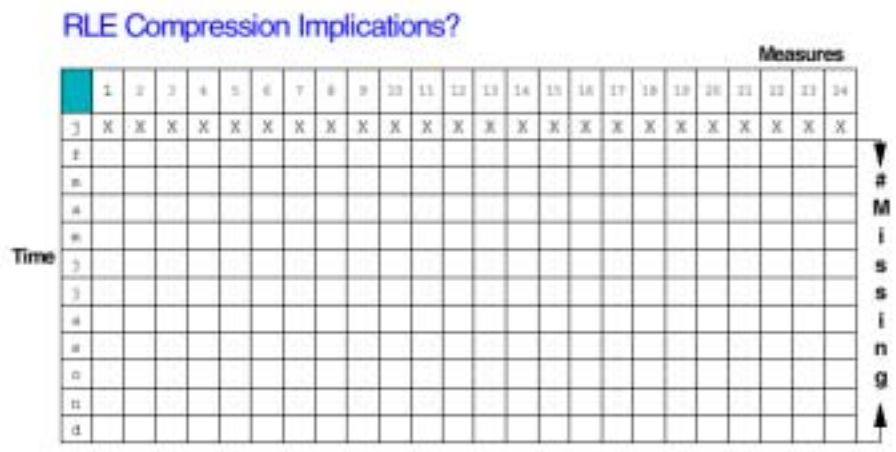


图 34.RLE 压缩含义(2)

归根结底，应利用实现最多压缩的压缩算法。如何在大纲中声明紧凑维可能会对 RLE 压缩产生影响。

4.8 使用SET MSG ONLY

SET MSG ONLY 是一个完全*无证明文件及无支持的*命令，现已发展为在咨询界有许多非常重要的应用，因此，很有必要在此讨论此命令。

4.8.1 如何使用SET MSG ONLY？

把级别 0 数据加载到一个全空的数据库。然后，使用以下脚本启动对该数据库的计算：

```
SET MSG ONLY;  
CALC ALL;
```

请注意，如果把此命令嵌入计算脚本内，就会生成一个语法错误。这是正常现象，可以完全忽略。

另外，请注意这样一个事实：使用此命令需要在级别 0 加载具有代表性的数据。*进行 SET MSG ONLY 计算后报表的可靠统计信息，是以使用数据加载所生成的数据块数方面与生产数据完全相同的数据为基础的。*请阅读以下内容了解其中原因。

4.8.2 SET MSG ONLY具有什么功能？

如果在计算脚本中指定了此命令，或把它作为数据库默认计算组成部分，此命令就会启动对数据库的假计算。“假计算”的意思是：DB2 OLAP Server 引擎将执行通过数据库运行的计算，并通过 I/O 跟踪计算过程(块创建及更新)。应用程序日志中将会报表此信息，其方式与 SET MSG SUMMARY 设置的相同。通过此信息，您可以推断出数据库已创建数据块的大概数量，就好像您实际计算过该数据库一样。

SET MSG ONLY *根据加载数据的性质*报表上层块创建。因此，如果仅加载一些数据，或人造数据，则只能生成一些上层块或人造上层块，或二者兼有！为确保可靠性及准确性，您需要在模型中所有维间加载真实数据。

这样，SET MSG ONLY 就使您能够非常精确地以比实际计算短得多的时间估算某个数据库(配置)生成的块总数。我们使用“估算”一词，是因为算法根本不会考虑成员公式(或通过计算脚本)。如果模型中包含创建数据块的公式，SET MSG ONLY 就不会报表这些信息，因此是否能够在这样的模型上有效利用 SET MSG ONLY 受此因素的限制。

4.8.3 使用 “Sample::Basic” 的SET MSG ONLY应用程序日志示例

下面的数据库统计信息是使用 GETDBSTATS 命令在 ESSCMD 中注册的：

Number of dimensions	: 11
Declared Block Size	: 12000
Actual Block Size	: 168
Declared Maximum Blocks	: 594
Actual Maximum Blocks	: 513
Number of Non Missing Leaf Blocks	: 177
Number of Non Missing Non Leaf Blocks	: 0
Number of Total Blocks	: 177
Index Type	: B+ TREE
Average Block Density	: 92.57143
Average Sparse Density	: 34.50292
Block Compression Ratio	: 0.9464407

请注意，数据库已经历级别 0 数据加载，而且只有 177 个块。

CALC ALL 命令是为此数据库发出的，前面是 SET MSG SUMMARY 语句。下面是进行完整的数据库计算后报表的统计信息：

Number of dimensions	: 11
Declared Block Size	: 12000
Actual Block Size	: 168
Declared Maximum Blocks	: 594
Actual Maximum Blocks	: 513
Number of Non Missing Leaf Blocks	: 177
Number of Non Missing Non Leaf Blocks	: 197
Number of Total Blocks	: 374
Index Type	: B+ TREE
Average Block Density	: 92.85714
Average Sparse Density	: 72.90448
Block Compression Ratio	: 0.9491525

产生的应用程序 SET MSG SUMMARY 日志项显示以下计算结果：

Total Block Created: [1.9700e+002] Blocks

Sparse Calculations: [1.9700e+002] Writes and [6.3200e+002] Reads

Dense Calculations: [1.7700e+002] Writes and [1.7700e+002] Reads

Sparse Calculations: [3.3096e+004] Cells

Dense Calculations: [0.0000e+000] Cells

请注意“Total Block Created: [1.9700e+002] Blocks”是数据库统计信息中报表的精确块数，“非缺失非叶块的数目”为197。把此日志摘录与加载同样数据的同一数据库的假计算(SET MSG ONLY)生成的日志摘录进行比较。运行计算脚本之前，用级别0数据重新设置和加载数据库：

Total Potential blocks: [3.7400e+002] Blocks

Sparse Calculations: [1.9700e+002] Writes and [0.0000e+000] Reads

Dense Calculations: [1.7700e+002] Writes and [0.0000e+000] Reads

Sparse Calculations: [0.0000e+000] Cells

Dense Calculations: [0.0000e+000] Cells

值得注意的是，应用程序日志中的“Total Potential blocks”(总潜在块数)统计数字与GETDBSTATS命令返回的“Declared Maximum Blocks”(声明的最大块数)统计数字不相同。“Declared Maximum Blocks”返回潜在块总数，是因为取了所有稀疏维的所有稀疏成员的交叉积。

注意输出方面的差异。假计算日志项的第一行为“‘Total Potential Blocks’(374)”，而实计算日志项的第一行却是“‘Total Blocks Created’(197)”。下面这一事实非常重要：此假计算中的“*Sparse Calculations: Writes + Dense Calculation: Writes* (374)”(稀疏计算：写数 + 紧凑计算：写数 (374))的数目与实计算生成的块总数(374)完全相等。同样地，“Total Potential blocks”(潜在总块数)度量也与数据库计算(374)生成的数据块实际数目相同。

4.8.4 您可以使用SET MSG ONLY执行什么操作？

如果可以生成与模型将创建的(无需您亲自创建)块总数相关的可靠统计数据，您就可以做两件非常奇妙的事情！事实上，在最初开发稀疏/紧凑方法和实用程序时，就曾使用SET MSG ONLY 计算生成总块数度量。

4.8.4.1 估算数据库大小

一旦把级别 0 数据加载到数据库，则立即记录以下统计信息：总块数和数据库大小 (ess*.pag 文件和 ess*.ind 文件的字节总数)。然后，用已加载块数除字节总数，就得出磁盘上每个压缩块字节的平均数。以块总数乘该数字，就得到压缩数据库的总大小近似值(以字节计)！

4.8.4.2 用有限数据估算数据库大小

通过取数据库的一个稀疏切片(把生产数据加载到所有稀疏组合之间的数据上)，您可以运行假计算，以便为总块数派生一个近似值。当对紧凑信息的访问受到限制时，这是十分有用的(因为数据与时间相关，未来信息不可用)。

如果可以有限度地访问紧凑数据，那么，通过选择一小组稀疏组合(例如，一个客户的所有数据)并用该稀疏组合的所有紧凑数据填充数据库，就可以生成一个数据紧凑切片。这样做的目的是为了确保更精确地估算实际块密度。这样，就可生成精确得多的平均块密度，而且每个块的平均字节数更能反映一个生产环境。现在，可以创建另一个关于数据库总大小的准确估算。

必须指出这样一个事实：在 7.1 版中，.pag 文件展开时增加了 8MB。另外，如果您选用稀疏/紧凑切片方法，则有必要考虑数据块标题和索引项额外开销。

4.8.4.3 估算批计算时间

假定您有一个数据库总大小的非常精确的近似值(来自上面的“数据库大小”)，您就可以用系统的 DB2 OLAP 吞吐量度量除这个数字，以便生成计算模型所需总时间的估算值。注意：在这里，我们假设批计算吞吐量为线性。经验观察一再表明，当计算接近完成时，DB2 OLAP Server 的吞吐量就会减小。在计算包含数千万个块的特大型数据库时，这一点尤其明显。

4.8.5 您需要对SET MSG ONLY有何了解？

SET MSG ONLY 算法与计算器高速缓存配合得很好。因此，如果您的数据库配置取得了多个位图，RAM 中就会出现整个过程，而且速度非常快。但是，如果在假计算过程中没有取得多个位图，此算法就会通过将全部空数据块写到磁盘来跟踪块创建。

此过程比取得多个位图时慢得多，可是仍比实际计算快得多。但是这意味着在(已描述过的)假计算中，您将看到 ess*.pag 文件和 ess*.ind 文件均会变大！请不要绝望。务必把数据库压缩设置为 RLE(记住，这些为空块 - 与原先一样，填有#Missing)，那么占用的总磁盘空间会快速减小，就像完成假计算所需要的时间会减少一样。

4.9 智能计算

在定期进行数据库更新的情况下，使用计算器引擎的智能计算功能是一个非常好的做法。

智能计算利用这样一个事实：作为索引结构的组成部分，对脏数据块(已经更新并需要重新计算的数据块)进行跟踪。而对数据库的定期更新可依赖这样一个事实：计算器引擎只能隔离那些受更新影响而需重新计算的数据块。因此，计算器可以*智能地*选择要刷新的脏数据块。

但是，请注意，智能计算的使用效果与模型相关。从长远来看，定期更新对大多数已有上层块产生影响的数据库将无法从智能计算的使用方法中受益。要牢记现有上层数据块需要更新，而且，就 I/O 而言，块更新的成本比数据块创建高出一倍。

智能计算需要管理。(有关智能计算的详细信息，请参阅《OLAP 数据库管理员指南》)。哪些块由于更新活动而被弄脏，并不总是直观或明显的。另外，与 CALC ALL 和 CALC DIM 或 AGG 一起使用时，智能计算的行为是不同的。

您可以在进行实际计算之前，使用无证明文件的 SET MSG ONLY 命令，确定大概有多少数据块会受到向数据库添加数据块的影响。

下面是“SAMPLE::BASIC”数据库略加修改后版本的日志摘录。在电子表格更新数据库时，创建了新的数据块。此更新会创建 2 个新数据块，并更新 10 个上层块。将包含 SET CLEARUPDATESTATUS ONLY 命令的计算脚本与 CALC ALL 命令组合。使用时，锁定和发送前的数据库被设置为清洁。使用以下脚本启动一个假计算：

```
SET MSG ONLY;  
CALC ALL;
```

下面是 OLAP Server 在应用程序日志中记录的数据：

```
Total Potential blocks: [1.2000e+001] Blocks  
Sparse Calculations: [1.0000e+001] Writes and [0.0000e+000] Reads  
Dense Calculations: [2.0000e+000] Writes and  
[0.0000e+000] Reads  
Sparse Calculations: [0.0000e+000] Cells  
Dense Calculations: [0.0000e+000] Cells
```

注意：“Total Potential Blocks” (潜在总块数) 数字正好是“Sparse Calculations...Writes” (稀疏计算...写数) 与“Dense Calculations...Writes” (紧凑计算...写数) 之和。对此数据库的完全计算(CALC ALL)会导致更新或创建总共 12 个块。这是所创建/更新的 10 个上层块与在计算过程中更新锁定和发送所创建的 2 个块相加之和。

从此示例中，我们可以推断出，此数据库稀疏维的 CALC DIM 或 AGG 将更新/创建 10 个数据块(即，Sparse Calculations:[1.0000e+001] Writes)。这样，您就可以把定期更新的重要性(受影响的数据块总数)与数据库联系起来。但您无法从此信息推断出：将要创建的新块数目与将要更新的现有上层块数目之比。

实际上，执行此计算会产生以下日志项：

```
Total Block Created: [4.0000e+000] Blocks  
Sparse Calculations: [1.0000e+001] Writes and [3.7000e+001] Reads  
Dense Calculations: [2.0000e+000] Writes and [2.0000e+000] Reads  
Sparse Calculations: [1.6800e+003] Cells  
Dense Calculations: [0.0000e+000] Cells
```

第一行与所创建的块数 (=4) 相关。从 SET MSG ONLY 的“Total Potential Blocks” 数字 (=12-4) 中减去此数字，您就可以(大概)确定，有多少数据块(=8)会因数据库增量更新及

计算而得到更新。当已更新块的数目接近现有数据块的 50% 时，您可以预计，智能计算至少需要对(空)数据库的完整级别 0 数据加载和重新计算一样长的时间。之所以这样，是因为就 I/O 而言，数据块更新与数据块创建的成本一样昂贵。而更新 50 个块所需的时间可能会与创建 100 个块所需的时间一样长。

4.10 其他问题

本节将讲述以前没有讨论过的一些实施设计问题：

- 分区技巧及策略
- 应用程序日志
- 数据加载优化

4.10.1 分区技巧及策略

数据库分区可以成为有助于衡量特大 OLAP 模型的一种极为有效的策略。一般来说，分区是把具有不同计算/处理需求的大型模型的那些部分分成若干个单独的数据库。这些不同部分可通过计算发生的频繁程度、或计算的复杂程度、或将模型的某个部分设置为读写还是只读进行区分。因此，分区可防止整个数据库以最小公分母进行计算。

分区不仅把数据库大小分成若干份，而且使数据库计算需求同时跨越多个 CPU。归根结底，分区策略是否成功，主要取决于分区定义的体系结构。也就是说，实施分区时必须认真考虑数据库配置。

4.10.1.1 按非相加性维进行分区

使用分区可以解决的主要问题是数据库总体大小或数据库缩放问题。当选择按其进行分区的维时，始终建议您选择稀疏维。但是，如果有存在于分区定义之间的值，解决方案会再次回避这个问题。下面就这个问题进行详细探讨。



图 35. 根据市场及方案进行分区

假定有两个作为分区策略的维：“市场”和“方案”(参见图 35)。

如果选择了“市场”，就会创建四个源数据库，每个地区(东、西、南、北)一个，并在目标数据库中对“市场”实施透明分区。将会有 4 个要更新和计算的小型数据库，而且现在就可以在多个 CPU 间同时实施分区。问题在于如何生成整个“市场”的数据，哪些值仅作为“东”、“西”等的和在分区定义间存在？

必须生成这些值。记住，检索过程中可能涉及到的块数可能是源数据库中所有稀疏维间数据的笛卡尔积。因此，根据服务器上可能会生成的 I/O，简单地使“市场”成员在目标分区中成为“动态”，并不总是可行的。这一点前面已经有过提示。大型数据库通常就是这样(请读者把“Sample::Basic”视为一个大型数据库！)

如果我们考虑按照“方案”维进行分区，情况就好得多。“总方案”的概念没有任何商业意义。因此，不会根据此分区定义生成任何值。现在，可以配置源数据库基础设施，从而轻易地把该数据输送到目标处。

4.10.1.2 按照紧凑维进行分区

如果我们更仔细地考察前一个假设示例，就会发现“方案”维完全符合数据集的紧凑性质。因此，分区策略会忽略我们的数据库稀疏/紧凑优化配置，而且我们把“方案”维标为稀疏以进行适应。但是，如果我们(明显地)完全根据某个紧凑维进行分区，那会出现什么情况？

来看这样一种情形：随时间周期性更新某个模型且把时间标为稀疏，会使数据块数量剧增到无法接受的程度。这样就无法周期性更新该数据库，也无法为用户交互而及时计算数据库。这一两难困境可通过以下策略进行解决。

请参看图 36 中的大纲。

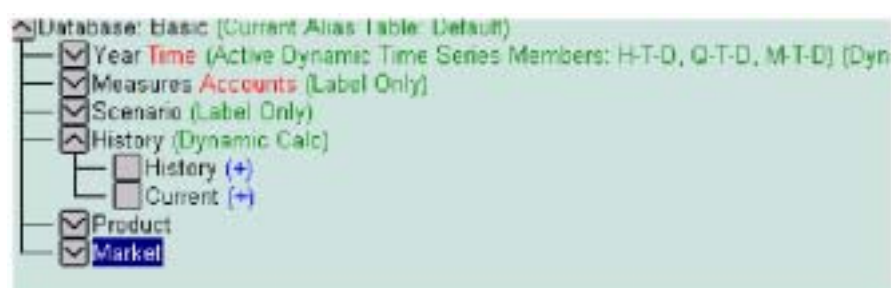


图 36. 大纲示例: 历史维

已把“历史”维作为稀疏维插入，其父级成员被标为动态(或“动态计算和存储区”)。现在，我们就可以创建两个源数据库，每个数据库拥有相同的维。添加维来增强其他维的设想非常聪明，而且在第 50 页 4.4.1.3 节“何时添加维”中已经讲述了这种技术的另外一个示例。

分区定义是按照“历史”维创建的。周期性数据加载到数据库的“当前”成员中。查询时动态派生整个“历史”的值。在下一周期更新之前，需将“当前”数据库数据最终按程序移动到较大的“历史”数据库中。

此策略有效地消除了把时间维标为稀疏的必要，并使用户能够及时获得周期性更新信息。下面这一事实同样重要：此策略有效地增大了批窗口的大小，其有利于尽快把数据从“当前”移动到“历史”多维数据集，从而缩短了周期性更新之间的时间。如果这是一个为期一周的刷新模型，管理员会花费几乎七天时间将“当前”数据库的最新更新移动到“历史”数据库。

我们来看此策略的一种变化情况，其中对“当前”数据库的周期性刷新每小时都在发生。那么，就会创建一个“当前”模型，而且所有其上层值都是动态的！在该模型与“历史”数据库之间进行透明分区，就可以使这些值可用。这样用户就能够进行几乎实时的数据分析。为了使分区策略的性能特征保持最佳状态，差不多需要每天晚上都要把“当前”模型中的数据移动到“历史”模型。

稀疏成员是否具有动态特征？

上面的假设示例违反了前面作为最佳做法确立的设置动态成员规则。该示例在整个分区之间设置动态成员！

这使我们想到前面提到的另一陈述：“数据库实施者需要了解所有标准，以此确定数据库性能是否在可接受的范围内。”上述策略是否会成为一种实际分区方法，取决于整个数据库(其中时间标为稀疏)内批处理周期性更新的额外开销与运行时某数据库分区内进行动态计算的额外开销之间的差异程度。

调优和优化规则并不是一成不变，实际上，这些规则必须适应具体环境以及相关用户的需求。

4.10.2 应用程序日志

确实值得重申定期检查应用程序日志的重要性。例如，事件日志记录 DB2OLAP 服务器如何(通过抽取或电子表格检索因素项)响应用户请求。将电子表格插件的检索性能特征与由不同查询工具生成的检索性能特征进行比较，则有助于快速查明故障点。

应用程序事件日志中会记录许多通常不被注意的事件。查阅该日志中记录的活动有助于即时查明数据库性能问题。参看以下示例：

[Fri Jan 05 14:54:28 2001]Local/Sample///Info(1012710)

Essbase needs to retrieve [1066] Essbase Kernel blocks in order to calculate the top dynamically-calculated block.

此日志项说明，由于稀疏成员上有“动态计算”(Dynamic Calc)标记，服务器需要创建 1066 个块，才能计算多维数据集的顶层块。由此可推断出用户检索时可能会生成块创建 I/O 的数量，因而可能需要对这些标签进行相应的调整。

不同的日志项表明，已经在每个维上设置了动态计算，并给紧凑成员标上“动态计算和存储区”标记：

[Fri Jan 05 14:59:31 2001]Local/Sample/Basicx/me/Info(1007125)

The number of Dynamic Calc Non-Store Members = [8 5 2 19 25]

[Fri Jan 05 14:59:31 2001]Local/Sample/Basicx/me/Info(1007126)

The number of Dynamic Calc Store Members = [0 1 0 0 0]

尽管前面一段信息可能作为设计决策，但后面的信息却非常糟糕，需要立即修正。数据块成员上的“动态计算和存储区”标记不会减少块大小，但是，确实会迫使 DB2 OLAPServer **更新**数据库中请求某个成员时而检索的每个块。

除非这是误标的标记，使用“动态计算和存储区”标记通常会反映有缺陷的逻辑。其推断出应以该方式标记成员，以便在第一次检索后使该值保持不变，分析师非常喜欢这种度量。这会产生急剧增加数据块更新次数的非预期结果，并会产生非常零碎的数据库。

4.10.2.1 记录应用程序日志事件

目前性能良好的多维数据集，将来如果修改了大纲且数据量增加，则有可能出现性能问题。如果多维数据集一开始性能就很差，则需要了解性能变差是在什么时候，以及在此之前进行过什么更改。记录对大纲所作的更改及执行关键任务所需的时间将有助于诊断性能问题。应用程序日志列出执行每项操作所需的实际时间。由于应用程序日志变大了，并且会被定期修整，所以您应把执行任务所用时间保存在另外一个文件中。日志文件可能包含以下信息：

[Tue Jun 27 18:39:04 2000]Local/Sales/Main/admin/Info(1003024)

Data Load Elapsed Time : [335.141] seconds

[Wed Jun 28 01:14:22 2000]Local/Sales/Main/admin/Info(1012550)

Total Calc Elapsed Time : [3718.276] seconds

表 13 是一个可能会记录的信息类型示例。此表表明是否以及何时会出现性能降低情况。执行某项操作所需的时间大量增加，并不一定意味着出现问题。也许是进行了某项更改，而这项更改需要进行大量处理工作。例如，将成员添加到某一紧凑维需要重新建立所有块。如果性能有所降低，可能是由于增加了多维数据集的大小。此时，可能有必要检查高速缓存命中率，以确保多维数据集没有突破高速缓存的设置。

表 13. 记录应用程序日志信息

日期	时间	操作	消耗时间
2000-06-27	18:32	建立维	45.469
2000-06-27	18:39	加载	335.141
2000-06-27	18:45	计算	3718.276
2000-06-30	14:00	给产品维添加成员	
2000-07-02	18:12	建立维	48.123
2000-07-02	18:29	加载	337.532
2000-07-02	18:35	计算	3723.533

4.10.3 数据加载优化

只有一种提高数据加载性能的方法，即按稀疏维对输入数据进行排序。这种方法之所以有效，是因为通过在稀疏维之间对数据进行排序，可确保由于加载数据而创建的每个数据块仅被访问一次。

让我们看看以下具有随机次序特征的数据加载文件的含意，如图 37 所示。

纽约	古典鸡尾酒	实际	销售额	61	61	63	66	69	72	77	78	68	69	61	66
佛罗里达	减肥奶油	预算	销售额	80	80	80	80	80	80	80	50	50	20	30	40
佛罗里达	葡萄	实际	销售额	80	80	81	85	89	105	110	114	87	94	70	75
佛罗里达	葡萄	预算	销售额	80	80	80	90	90	110	110	120	90	100	70	70
佛罗里达	草莓	预算	销售额	80	120	130	130	140	150	150	160	130	120	80	80
马萨诸塞	葡萄	实际	销售额	80	80	79	75	71	60	57	55	72	66	88	82
马萨诸塞	葡萄	预算	销售额	80	80	80	80	70	60	60	50	70	70	90	80
马萨诸塞	草莓	实际	销售额	80	56	53	53	49	44	44	41	50	53	80	73
马萨诸塞	草莓	预算	销售额	80	60	50	50	50	40	40	40	50	50	80	70
佛罗里达	草莓	实际	销售额	81	115	121	121	130	144	144	154	126	118	78	85
佛罗里达	深色奶油	预算	销售额	90	90	90	90	100	100	100	100	110	60	80	100
马萨诸塞	深色奶油	预算	销售额	100	100	100	90	90	80	80	70	60	60	90	70
佛罗里达	减肥奶油	实际	销售额	110	110	112	103	110	110	115	69	76	51	52	56
佛罗里达	香草奶油	预算	销售额	110	120	120	110	130	130	140	170	140	90	110	120
佛罗里达	深色奶油	实际	销售额	120	118	120	127	130	133	133	140	149	126	121	145
马萨诸塞	古典鸡尾酒	预算	销售额	120	120	120	110	110	100	100	100	90	100	110	90
马萨诸塞	古典鸡尾酒	实际	销售额	126	128	125	117	114	111	111	105	98	116	120	99
马萨诸塞	深色奶油	实际	销售额	130	132	129	121	118	115	115	109	102	120	124	103
纽约	古典鸡尾酒	实际	销售额	134	189	198	198	210	230	230	245	199	187	123	133

图 37. 前两列标为稀疏的加载文件

图 37 中的加载文件用于“Sample::Basic”数据库，该文件的前两列(“市场”和“产品”)被标为稀疏。由于把此数据加载到空数据库而创建的第一个数据块是“纽约->古典鸡尾酒”。注意：这也是最后一个(可见)记录的地址。因为加载过程中已创建“纽约->古典鸡尾酒”数据块，OLAPServer 将从磁盘上获取现有块，并对其进行更新。这会导致两个非预期事件。首先，除了创建块之外，数据加载中还会出现一个更新块。由于出现了这一数据块，I/O 增加了两倍，又因为 DSB2 OLAP Server 从不对数据块进行“恰当更新”，于是数据库分成若干碎片。

现在，请看加载同一个数据文件的含意，该文件已按稀疏维进行排序。如图 38 所示。

佛罗里达	可乐	实际	销售额	210	200	210	222	235	278	286	286	249	205	197	202
佛罗里达	可乐	预算	销售额	190	190	190	210	220	260	270	270	230	170	170	190
佛罗里达	深色奶油	实际	销售额	120	118	120	127	130	133	133	140	149	126	121	145
佛罗里达	深色奶油	预算	销售额	900	90	90	90	100	100	100	100	110	60	80	100
佛罗里达	减肥可乐	实际	销售额	200	206	214	267	273	282	336	277	230	218	245	262
佛罗里达	减肥可乐	预算	销售额	190	190	200	250	250	260	310	260	210	180	220	230
佛罗里达	减肥奶油	实际	销售额	110	110	112	103	110	110	115	69	76	51	52	56
佛罗里达	减肥奶油	预算	销售额	80	80	80	80	80	80	80	50	50	20	30	40
佛罗里达	葡萄	实际	销售额	80	80	81	85	89	105	110	114	87	94	70	75
佛罗里达	葡萄	预算	销售额	80	80	80	90	90	110	110	120	90	100	70	70
佛罗里达	草莓	实际	销售额	81	115	121	121	130	144	144	154	126	118	78	85
佛罗里达	草莓	预算	销售额	80	120	130	130	140	150	150	160	130	120	80	80
佛罗里达	香草奶油	实际	销售额	150	154	155	151	170	177	189	226	182	174	164	177
佛罗里达	香草奶油	预算	销售额	110	120	120	110	130	130	140	170	140	90	110	120
马萨诸塞	深色奶油	实际	销售额	130	132	129	121	118	115	115	109	102	120	124	103
马萨诸塞	深色奶油	预算	销售额	100	100	100	90	90	80	80	80	70	60	90	70
马萨诸塞	葡萄	实际	销售额	80	80	79	75	71	60	57	55	72	66	88	82
马萨诸塞	葡萄	预算	销售额	80	80	80	80	70	60	60	50	70	70	90	80
马萨诸塞	草莓	实际	销售额	80	56	53	53	49	44	44	41	50	53	80	73
马萨诸塞	草莓	预算	销售额	80	60	50	50	50	40	40	40	50	50	80	70
马萨诸塞	古典鸡尾酒	实际	销售额	126	128	125	117	114	111	111	105	98	116	120	99
马萨诸塞	古典鸡尾酒	预算	销售额	120	120	120	110	110	100	100	100	90	100	110	90
纽约	古典鸡尾酒	实际	销售额	134	189	198	198	210	230	230	245	199	187	123	133
纽约	古典鸡尾酒	实际	销售额	61	61	63	66	69	72	77	78	68	69	61	66

图 38. 按稀疏维排序的加载文件

注意：现在，相继把每个稀疏组合加载到数据库中，这意味着每个数据块仅接触一次。这样就完全消除数据块更新和数据库碎片！

4.10.4 建立安全模型

IBM DB2 OLAP 可提供全面的多层安全系统，管理员可以利用它控制对应用程序、数据库以及相关主题的访问。

可以通过许多方法衡量实施是否成功，其中包括：

- 系统是否满足企业基本需求？
- 系统是否赋予适当个人适当权限？
- 系统对用户是否是无缝的且不唐突？
- 系统是否易于维护？
- 此安全系统是否可以导入到一个多服务器环境？

初步实现安全系统或模型仅仅是一个起点。在系统的整个生命过程中，与维护、自动化以及连续完整性相关的事项具有更大重要性。而且，如果使用的安全系统设计不适当，就无法达到基本目的：安全和无缝的方式，帮助推动客户机的应用程序开发工作。

安全模型定义阶段不应只是试图定义目标访问需求，而可考虑未来自定义应用程序开发。可以使用 DB2 OLAP API 在用 Visual Basic、C++ 或 Microsoft Excel 开发的客户机应用程序内，访问所有安全组件。这就意味着可开发能解释和遵循定义的安全组件自定义应用程序，从而使开发人员能根据每个用户的具体情况定制客户机应用程序内容。这取决于自定义应用程序的重点，但可以包含其他，例如，根据在用户安全过滤器配置文件指定的成员，生成自定义报表。

4.10.4.1 维护安全系统

安全模型定义和实施只是管理安全系统过程的部分工作。安全系统维护也是一项必须认真考虑的工作，只有这样，才能确保达到安全模型的本来目的，并确保个别安全事项的维护不会成为繁重和耗时的任务。

可以通过手动方式透明地管理所有安全事项。这通常是典型行动步骤，而且在整个安全系统相当小的情况下，这可能是一种切实可行的选择。但是，如果某个安装中包含数百个用户、几十个应用程序和数据库，并需要许多复杂过滤器的话，维护工作就可能非常困难。此外，手动维护安全事项可能会把人为误差引入等式之中。可能会无意中授予用户错误的访问权限，也可能会把离职的员工保留在系统中。

API 是开发维护应用程序的一种切实可行方法，其可减少总体维护工作的负担，并可提高维护过程的准确性和完整性。

以下两个示例描述了使用 API 实现安全维护自动化的理想情况。

不断更新 DB2 OLAP 用户列表

在此示例中，系统管理员实施一种自动化系统以不断更新 DB2 OLAP 系统用户。我们假设管理员有权从某个局域网访问用户 ID 主列表。

访问公司中央资源(如局域网用户列表)有助于管理员在发生以下事件的情况下及时更新用户列表：员工离职、新员工任职或调离其他部门。可以根据公司局域网系统上的有效用户配置文件列表，自动从 DB2 OLAP 安全系统中删除离职员工。采用哪种公司中央系统或资源验证 DB2 OLAP 系统上的用户配置文件列表并不重要。重要的是可使用 API 准确地将 DB2 OLAP 用户配置文件与包含用户配置文件的其他系统合并起来。

一个重要的前提条件是：开发 DB2 OLAP 安全模型时使用的命名约定应与用于比较的主系统或开发两个列表之间的映射机制的主系统所用的命名约定相同。理想的情况是，管理员在 DB2 OLAP 中利用与在主系统中相同的用户名创建配置文件。这有助于促进及时更新 DB2 OLAP 用户列表，以及简化最终用户登录过程(最终用户不必记住另外用于 DB2 OLAP 的用户名)方面的自动化工作。为了提高安全系统完整性，会生成一个验证日志文件，以便通知管理员已删除的用户。

自动过滤器生成示例

可以利用公司内的其他系统自动生成和维护过滤器。例如，需要指定过滤器读或写访问特定市场的安全模型，既需要初始设置，也需要在员工变更职位、离职或任职时保持连续维护。我们假设这样一个需求：相关员工只能查看其所在市场范围内的数据。

这样就会在另一个系统(也许是 DB2 表或其他部门的某个表)中维护此信息。可以开发 API 应用程序来自动读取员工和市场之间关系的过程，并使用 DB2 OLAP 内的安全过滤器实施此关系。该进程可包括创建指定到用户 ID 的原始过滤器及维护过滤器，以响应机构内发生的安全事件。这可能包括某个员工不再有权查看某个市场时删除相应过滤器，或更改某个过滤器的内容以反映某个员工已调配到另一个市场。添加和删除过滤器以及更改现有过滤器内容的功能均可通过 API 提供。

4.10.4.2 在多服务器环境中管理安全性

多服务器环境是 DB2 OLAP 领域中常见的事情。目前,应用程序管理器不在服务器间管理或迁移信息。这对于现有的庞大安全系统是一个非常重要的问题。例如,有这么一个多服务器安装情形,其中包含测试服务器和生产服务器,问题是:如何从测试服务器复制生产服务器安全系统?

此需求的自动化是通过 DB2 OLAP API 实现的。API 对所有 DB2 OLAP 安全系统组件具有完全访问权限。可以读取用户和组配置文件以及过滤器详细信息,并可随后将其写出。可将 API 驱动进程编写为能从生产服务器读取任何安全组件,并可写到测试服务器。

4.11 本章评语

本章以一定篇幅讲述了数据块和索引存储结构的概念。在让读者正确理解这些结构的情况下,本章精心设计了其他节和小节。只有在充分理解了数据块和索引结构的概念之后,才能有效地进行数据库调节工作。

第 5 章 访谈和经验

有关 DB2 OLAP 服务器的绝大部分专门技术和宝贵知识属于那些实施 DB2 OLAP 服务器的人士。

我们的最佳实施者是客户和合作伙伴，他们在设计满足用户需求的 OLAP 数据库及改进面临开发限制的生产性 OLAP 数据库方面起着重要作用。

在实施 OLAP 应用程序时，通常会存在多种方法。为了反映知识的实用性和多样性，我们通过与客户及合作伙伴进行的各种访谈来解释如何实施部署 OLAP 应用程序。

5.1 简介

我们请各种不同客户及合作机构突出其设计、实施和部署 OLAP 应用程序方面的最佳做法。本章将详细介绍访谈结果。

下面的访谈问题与作为整体介绍的不同个人访谈可相互参照。

这些访谈以同样的结构为基础，并且都包含以下问题：

请描述您的环境。

- 您采用了什么设计方法和设计选项？
- 您使用什么客户机工具？
- 您使用什么管理过程？
- 您如何管理操作？

每次访谈主要关注某个具体领域，如表 14 所示。

表 14. 访谈

	环境	设计	客户机工具	管理	操作
Steve Beier 访谈	X	X		X	X
George Trudel 访谈	X	X	X		X
Mark Rich 访谈	X	X		X	
Joe Scovell 和 Jacques Chenot 访谈	X	X	X	X	X
匿名人士访谈	X	X			X
Rich Semetulskis 和 Alan Farkas 访谈	X				X
Aster Hupkes 访谈		X			

5.2 访谈结果

在本节中，我们将介绍访谈结果。

5.2.1 Steve Beier访谈

Steve Beier(SB)是一名高级 IT 专家，他已在 IBM 全球服务部工作了五年时间。目前在 IBM 存储部开发企业信息系统。他们利用 DB2 UDB、Essbase、DB2 OLAP、Visual Warehouse、AIX 和一些“自制”工具建立和维护项目。他的主要职责包括项目体系结构、技术指导和“恢复”支持。

5.2.1.1 环境

1. 描述您的 OLAP 环境。

SB：“我们的 OLAP 系统都是在 RS/6000 上运行的 AIX。之所以选择 AIX，是因为它具有很好的灵活性、可扩展性、可用性及易管理性。所需的全部应用程序(包括用于数据仓库的 DB2)均能在 AIX 上运行。我们对生产的选择始终是 UNIX。

我们的生产系统在 2 台 RS/6000 上运行。并在具有 16GRAM 及 12 个 262MHz 处理器(具有 8MB 二级高速缓存)的 S7A 上运行 DB2 和 DB2 OLAP。额外二级高速缓存是一重要考虑事项。DB2 OLAP 运行在另一个生产系统上，即具有 32G RAM 和 6 个 450MHz 处理器的 S80 上。这是一台功能十分强大的机器，计划将来升级到 12 个处理器。该系统中的额外内存使我们可以调节 OLAP Server 高速缓存，从而获得最佳计算性能。我们设法给索引高速缓存分配足够内存，以便获得数值为 1 的索引高速缓存命中率。这就意味着整个高速缓存都在内存中，因而可减少 I/O。同时还利用 DBA 指南中描述的算法调节计算器高速缓存。我们分配足够内存让计算器能够使用多个位图，而且通过 SET MSG SUMMARY 运行计算来对此进行验证。用该方法调节的最新模型的计算时间从 4 小时降至 40 分钟。”

2. 开发和测试情况如何？

SB：“除生产系统外，我们还运行着开发系统及测试系统。理想的情况是，开发系统与生产服务器相配，但是为了降低成本，我们把旧的生产服务器级联到开发系统，而把旧的开发服务器级联到测试系统。为 OLAP 配备合适的硬件至关重要。在一段时间内，由于生产服务器功率不足，致使工作非常困难。很明显，您并不希望开发和生产互相妨碍。为正常工作，客户需要不间断地访问生产服务器，而开发人员却需要一个他们可以控制的环境。于是，我们决定把测试服务器包括在内，这样还可以测试产品新版本、运行代码以及试验新想法，同时不会妨碍开发工作或用户工作。”

3. 您的源数据来自哪里？

SB：“现在，我们的许多数据来自 SAP，但也从其他 DB2 系统及平面文件中获取数据。直到最近，还不得不从 VM 中抽取。所有这些数据都经过一个复杂且高度自动化的‘抽取转换和加载’(ETL)过程，从而加载到 S7A 上的 DB2 数据仓库中。这些过程是通过使用 Visual Warehouse¹ 及自行开发的自定义系统运行的。然后，使用 SQL 接口把数据直接从仓库加载到我们的 OLAP 多维数据集之中。加载之前，DB2 进行所有数据转换工作。我们使用非常简单的加载规则，每个规则文件均从数据仓库的‘加载视图’中加载。这极大地改进了维护和重复利用工作，因为我们精确地知道数据来自哪里以及其精确程度。创建‘加载视图’的数据定义语言(DDL)被存储和管理起来，我们就可以非常迅速地重建视图。从‘加载视图’中选择的 SQL 语句总是包含一个基于模型稀疏维的 ORDERBY 子句。这样可按块加载数据块，从而减少 I/O 并提高性能。”

4. 您使用什么存储设备，如何使用？

SB：“我们针对操作系统使用 SCSI 设备，针对 OLAP 存储则使用 SSA 设备。RAID-1 镜像极为有效，并且其运行情况也非常适合。我们把各组页面及索引文件分配到自己的 I/O 通道。”

5. 目前您支持多少个 OLAP 多维数据集？

SB：“目前，在生产方面我们有 120 个多维数据集。它们的大小从几个 MB 到 8GB 不等，具体取决于用户的需求。”

6. 多维数据集支持什么应用程序？

SB：“我们有种类繁多的应用程序，其中涉及供应和需求管理、财务、规划和预测。许多用户使用锁定和发送来更新模型及运行方案。”

7. 您支持多少用户？

SB：“我们总共有 250 个用户。平均一个服务器负载大约 20 个并行用户。在峰值时间，服务器需要处理 50 个并行用户。”

8. 您工作小组的构成如何？

SB：“我们总共有 20 个人管理整个数据仓库及 OLAP 应用程序。其中一些具备关

¹ Visual Warehouse 是一种 IBM ETML 工具，现已包含在 DB2 UDB V7 的数据仓库中心功能及 DB2 仓库管理特性中。

键技能的人员是数据库管理、Essbase 和 DB2 OLAP Server 专家、支持和帮助台人员。日常工作的主要内容是管理夜间抽取、仓库加载和 OLAP 编译过程。每晚都有两个人负责管理和监视此过程。我们拥有许多精通 Java、‘C’及 SQL 的程序员，他们负责开发包括仓库管理过程、OLAP 实用程序在内的各种自定义程序。我们还有多个高级用户帮助进行设计工作。”

5.2.1.2 设计

1. 您的 OLAP 需求来自何处？

SB：“ 我们的商业用户推动着需求的发展。我们的工作就是满足其需求，并与他们通力合作在需求与可用资源之间进行平衡。”

2. 开发新的 OLAP 模型时，您采用什么技术？

SB：“ 计划和项目管理非常重要。在理解需求并合理管理项目之后，就可以非常迅速地实施 OLAP。实施工作进展得如此之快，以至于您可能会认为没有必要进行规划和项目管理工作，但是，我们很清楚不能忽略应用程序开发这些方面。我们的高级用户处于设计过程的中心。他们对技术的理解非常深刻，因此，通常，首先由他们使用应用程序管理器对需求进行整理，以便组合成示例大纲。我们与他们共同工作，改进该大纲并确定输入数据需求。当开发多维数据集时，我们构造一个原型，并经过多次反复与用户密切合作。这些深刻了解业务且对技术有一定了解的高级用户的工作非常重要。您的 OLAP 专家详细了解技术并对业务有一定了解同样非常重要。使您的 OLAP 专家与高级用户密切合作意义非常重大。”

3. 您如何验证多维数据集？

SB：“ 当一个新的多维数据集进入生产阶段后，签发之前，验证工作是用户的部分职责。生产多维数据集时，我们使用 SQL 从资源系统进行抽取，并使用 API 从多维数据集进行抽取，以及自动进行尽可能多的验证工作。同时使用 SQL 程序和电子表格完成一些手动、自动化工作。作为批过程组成部分，我们每天晚上都进行验证。”

4. 开发应用程序时，您经过什么调节步骤？

SB：“ 当我们改进原型并迈向生产阶段时，进行加载及计算性能测试。当多维数据集进入生产阶段时，则完成最终调节，但是，会存在始终使用的设计默认值和标准。它们包括：密切注意紧凑及稀疏维的数据存储设置、使用“沙漏”大纲布局、获得

介于 20K 和 60K 之间的块大小、对紧凑成员进行完全动态计算以及使用将紧凑维排除在计算之外的计算脚本。虽然没有正式的调节一览表，但我觉得应该这样做。我们经常根据用户需求调整模型。例如，可能会删除多维数据集中使用量过大的一些动态计算。有时，我们从大纲中移出公式，并将其移到一个计算脚本之中，这样就可以更好地控制计算顺序。我们有许多曾经必须把“时间”标为稀疏维的多维数据集，因为必须通过多维数据集才能清除和加载时间切片。”

5.2.1.3 客户机工具

1. 您使用什么客户机工具？

SB：“我们在交互式分析中使用电子表格插件，但是我们还运行作为位图捕获并通过 Lotus Notes 发布的报表和图表。报表和图表是使用我们自己的程序(推动电子表格程序)而自动生成的。在不久的将来，我们很可能会使用 Alphablocks 查看 Web 内容。”

2. 谁来设计您提供到客户机的报表？

SB：“用户通过我们的控制系统请求报表及对现有报表进行更改。我们的工作小组负责生成报表并进行更改。”

5.2.1.4 管理

1. 您如何管理大纲更新？

SB：“我们使用针对加载视图的加载规则从数据仓库中建立维。新成员添加到该维组成部分的存储桶中。然后，用户负责将新成员移到层次结构中的恰当位置。”

2. 您如何管理加载更新？

SB：“我们实行全面的多维数据集重建和增量数据加载。对于额外开销很低的小型多维数据集，我们清除数据，并彻底重新加载。对于较大的多维数据集，以及那些用户使用锁定和发送输入数据的多维数据集，我们使用增量加载。有些多维数据集提供滚动时间窗口，要对其进行管理，我们使用参数化的自定义程序，这样就可以驱动这些程序清除适当切片，并将新数据加载到多维数据集的那个部分。”

3. 您如何管理数据库碎片？

SB：“ 我们觉得最好运行一个级别 0 导出，然后重新加载并进行完整计算。曾尝试过紧凑重构，但我们发现导出、运行简单默认数据加载和计算效率更高。 ”

4. 您是否使用应用程序分区？

SB：“ 是的，我们已有许多分区的应用程序，并发现分区的最佳应用可用于高可用性系统。我们可以在后台加载及计算分区，然后使用复制刷新主要的多维数据集，还使用了透明分区，但目前我们尚未发现关于链接分区的需求。我们不使用大纲同步。 ”

5. 您如何管理安全性？

SB：“ 我们有专人负责安全性的各个方面。可以想象，IBM 对安全性非常重视，所以这是一项重要的工作，而且您需要恰当的人来做这项工作。我们使用 Lotus Notes 编写了自定义支持系统，用户通过访问该系统来请求 ID、数据库访问或安全过滤器更改。该系统将请求路由到用户经理处进行签发，然后路由到安全协调员处进行实施。该系统还帮助进行审核以及使用户 ID 和权限处于受控状态。我们使用 OLAP 安全系统的所有可用功能，而且殷切希望该产品能够提供更多这样的功能，例如，提高管理和控制角色的粒度、密码终止规则、审核及统计。我们还使用 LumenSoft ServerManager 协助进行大量与安全性相关的活动，否则，这些活动将会是非常耗时且容易导致错误的手动过程。 ”

5.2.1.5 操作

1. 您采用什么步骤将新型应用程序投入生产？

SB：“ 我们将多维数据集从开发系统移至生产系统，并在该环境中对其进行测试。然后，将新应用程序集成到自动化过程中并进行测试。当我们如期集成、调优及构造新应用程序时，则与用户共同完成了完全的端到端测试。其中包括验证输入数字和计算。一切就绪后，我们从用户那里签发该应用程序。 ”

2. 您如何管理备份？

SB：“可以采用多种策略。

首先是使用该产品的‘开始存档’和‘结束存档’功能进行‘热’备份。这些命令将应用程序置于只读模式，这样就无法更改数据库。用户可以继续查询多维数据集，而备份操作非常安全。这是理想的解决方案，即使它不在使用，我们仍然发现‘开始存档’命令启动了该应用程序。‘结束存档’不终止以前闲置的应用程序，则使用这些命令的自动过程就会导致每个应用程序的运行，而这会用尽所有服务器资源。我们可以停止该自动过程，但如果它是用户正在使用的应用程序，将其关闭，就会完全丧失此热备份便利可能带来的优势。

第二项策略是通过终止服务器、进行备份及在备份完成后启动服务器来进行‘冷’备份。我们不能使用此方法，由于用户遍布世界各地，因此，我们没有时间使服务器停留足够长的时间可靠地完成一次备份。

最后一项策略是，我们使用卸载应用程序并进行备份的自动化过程。计划在得知应用程序不在使用中时卸载和备份。在特殊情况下，会有风险。某个用户可能正在使用我们使其脱机的应用程序，但迄今为止，还没有出现过问题。我们可以尽可能高效地备份每个应用程序，并尽量减少对用户的影响。

我们使用 IBM 的 ADSM 备份应用程序来备份整个安装目录及应用程序数据。根据更改情况进行增量备份，这样我们就可以把所需时间降至最小程度，而且始终可以恢复任何内容，从单一文件到完整的 OLAP 系统。作为批过程的组成部分，我们每天都运行备份。通常，我们可以用 4 小时的时间从问题中恢复。理想的情况是，简单备份大纲和级别 0 数据，但是还有许多其他需要管理的文件，而且应用程序中的文件与系统中的其他文件(如 CFG 和 SEC 文件)之间通常存在某种关系。因而不可能通过这样一种简单策略恢复完整的应用程序。”

3. 您采用什么策略应用修复包或修补程序？

SB：“理想的情况是，当应用工作最适合日程安排时，我们能每季度选择一次应用服务，但迄今为止，还没有办法做到这一点。我们发现不得不频繁应用服务，以便获得所需的修复。如果可能的话，我们将不应用最新的服务级别，而宁愿等待两个月时间，来了解其是否稳定。但是，我们经常需要修复，因而选择的余地非常小。由于非常频繁地应用服务而没有时间完成测试循环，因此，我们把服务直接安装到

生产服务器上。我们正在寻找采用 Scapa 技术的某种产品，这种技术将使我们自动运行针对服务器的复杂的查询工作负载仿真，这样就可以快速检查服务器，并确保没有引入功能或性能问题。”

a. 您应用新版本的策略是什么？

SB：“只要有时间，我们就参与尽可能多的计划，而且某个版本一经发布，我们就开始观察其新的特性和功能。但是并不急于投入生产，而是先观察 6 个月。

我们将把新版本放在测试服务器上，并从构造最重要的多维数据集开始。然后，对一些其他多维数据集进行抽样，即那些在某方面有趣的多维数据集。之所以有趣，可能是因为它们所使用的功能，也可能是因为它们的大小。我们观察新版本的许多特征，其中包括加载和计算性能、存储需求及自动化。我们还计划使用 Scapa 来推动查询工作。用户帮助进行验证工作。同时还使用新版本测试整个端到端自动化过程。测试和验证循环完成后，通常，我们设法在周末升级生产服务器，并刷新所有多维数据集。

b. 您如何管理升级客户机工具？

SB：“我们把新工具放到 IBM 的公司安装站点 ISSI 上，并向用户发送电子邮件通知，告诉他们有新的升级。当我们应用服务时，升级为可选，当我们升级到一个新版本时，所有用户必须升级其客户机代码。用户使用 ISSI 下载和安装这些代码。”

4. 您的批过程是由什么构成的？

SB：“我们每周的目标都是顺利度过一周，而没有问题发生。当系统稳定并运行一段时间后，确实实现了该目标。显然，进行的更改越多，整个过程就越不稳定。最近，我们对数据源进行了一些重要更新，目前仍保持每周 100% 正常运转。我们定期召开会议，检查问题、讨论解决方案以及跟踪进度。这些会议由一些用户进行管理，所以他们参与整个过程。他们非常重视我们所面临的问题。我们对问题的影响有较深刻的了解，并将共同设法解决问题。”

5. 您如何监控批过程？

SB：“ 驱动批过程的自动化程序是我们管理整个系统方法的关键。这些程序发送有关成功或失败的电子邮件通知。所幸的是，工作小组的部分成员在阿根廷工作，那里的时间比我们这里早 5 小时，这样他们可以提前着手解决任何问题。 ”

6. 批过程中，您何时验证多维数据集？

SB：“ 验证操作已内置到批过程中。如果没有完成验证，我们不会认为批过程已完成。 ”

7. 您现在使用什么批窗口？

SB：“ 我们每天晚上大约花费 8 小时完成数据仓库加载并建立所有多维数据集。批过程大约 20%到 30%为 ETML(抽取、转换、修改、加载)和仓库加载，而 70%到 80%为 OLAP 加载和计算。我们努力同时进行尽可能多的操作，但是，相关性和资源使其仅限于此。 ”

5.2.1.6 结束

1. 您认为您正在做的世界一流的事情是什么？

SB：“ 我们所做过的最为出色的事情包括：先进的端到端自动操作、基于 Lotus Notes 的用户和安全性管理基础设施，以及在数据结构和大纲中的实施标准。 ”

2. 为了改进 OLAP 安装，您将会进行的最重要更改是什么？

SB：“ 提高性能和可靠性是我们的首要任务。 ”

5.2.2 George Trudel访谈

George Trudel (GT)是 Scrip Pharmacy Solutions 公司的信息服务总监。他早在 1990 年就开始从事多维数据库方面的工作,是第一个试用 DB2 OLAP 的商业用户。Scrip Pharmacy Solutions 公司总结了处方事务数据,作为向客户提供知识的组成部分。

5.2.2.1 环境

1. 描述 OLAP 环境。

GT:“我们目前的服务器端生产环境是具有 2G 内存和 1.3TBEMC 对称存储阵列的 DEC Alpha 4100。我相信,3 年以前 DEC Alpha 被视为顶端服务器之一。出于对多维数据集总体预期大小的考虑,性能是选择平台的关键因素之一。目前,在该环境中运行的 DEC Alpha 已经没有任何特殊优势可言,所以我们正在积极地用运行于 AIX 的 RS/6000 服务器取代 DEC Alpha。进行此改革的重要原因之一是:DEC Alpha 不再支持 Windows NT,因此,Hyperion 将不再在 DEC Alpha 上支持其产品。由于已知道必须采取措施,所以我们到处去寻找同一性能类型的平台。RS/6000 由于其卓越的性能而成为我们的首选。我们只需为所有 OLAP 多维数据集运行一个生产服务器。目前,这个单一生产服务器可运行四个 500MHz 处理器。无论任何时候,我们可以使用一个包含 40 个模型的库中的 10 到 15 个模型。”

2. 开发和测试情况如何?

GT:“我们有一个运行于 Windows NT 服务器(450MHz、25G 硬盘、256M RAM)的开发环境。在此服务器上遇到的问题是,它不能复制大型生产模型。因此,我们可能会在测试服务器上进行一些模型开发工作,但是,通常把数据加载到生产机的非生产数据库中。我们在开发机上进行一些测试,同时也在生产机器上进行测试。”

3. 您的源数据来自哪里?

GT:“我们的源数据实际上来自两个不同地方。首先,我们运行 Pharmacy 处方处理系统。作为一种 Pharmacy 索赔处理器,我们在 AS400 上运行专用程序包。我们通常每天收集当天的活动,并将其下载到一个关系数据库之中。然后,运行编写数据摘要的 SQL 程序并为多维数据集提供输入文件。最后,将关系输出文本文件加载到多维数据集中。此过程是 OLAP 仓库演进的结果。起初,我们是从 AS400 获得源文件。后来用来自数据仓库的源文件取代了该过程,而且,我们没有超越那种做法的商业案例,使其成为某种类似于集成服务器的东西。”

4. 您使用什么存储设备，如何使用？

GT：“我们有一个 EMC Symmetrics 1.3TB 存储阵列。实际上，此阵列包含 96 个 18GB 驱动器。我们把这些驱动器从逻辑上分为许多卷。有三个卷由 OLAP 支持，其中一个大约 500GB，另外两个为 300GB。我们不在特定机器上使用 RAID 或镜像。

5. 您如何利用服务器上的多个处理器？

GT：“我们同时运行多项计算和数据库加载。我们有深入生产数据库内部的商业应用，而且可随时使用 10 到 15 个活动模型。”

6. 您目前支持多少个多维数据集？

GT：“最近，我们经历了一个再生过程。CIO 的原始构思过程是建立一个大型数据库，该数据库包含他原来希望见到的所有维和一切事物。我们的生产环境就是从该理念演化而来，它基于导致细分信息数据库的完全大小。目前，我们的模型是按区域划分的。首先把模型分成三个区域，然后再按时间进行划分。当前基础设施在模型中为三个不同区域，包含六个月的数据。平均数据库大小约 18GB。以前，数据库超过 30GB。

我们有多达七个维，由于模型的完全大小，并没有充分使用它们。经过商业分析，我们为制造商删除了一个维，因为实际上并没有用到它。必要时，可以建立特定制造商模型来满足该信息的独特需要，因为这不同于查看主要 OLAP 核心数据的方式。通过从生产模型中抽出该维，可使总体大小需求减少 60% 的空间利用。我们实现了极大节约，这使我们可以做一些其他事情，即职责中的正常报表和分析部分。

我们还对潜在客户进行分级。我们发出确定构造分级模型维和度量的格式。潜在客户可填充这些模板的任何部分。通过关系后端进程运行这些文件，该过程生成所需的全部多维数据集源信息。我们使用自动建立这些多维数据集的编程例程。可使它们在一天内回转，以便支持分级人员进行分析工作。通常，潜在客户会提供一些药物成本方面的信息。将取得的信息在我们的模型中运行，并与已知情况进行一些比较。这就为分级过程建立了基础。”

7. 多维数据集支持什么应用程序？

GT：“ 尽管我们实际上完全混合地运行规划、预算、查询和预测，但主要是自己提供的有关载体账户组处方过程的摘要信息。客户维在全国大概有 40,000 组。

我们通过近 200 种不同可用度量编写药物摘要，这些度量包括计数、美元及有关处方成员的百分比。我们不跟踪多维数据集中的成员级详细信息，而在医生级编写摘要。针对 40,000 组、15,000 多种药物，跟踪 100,000 名医生。 ”

8. 您支持多少用户？

GT：“ 总数为 50 名，但只有 30 名是有效的。原因是我们为客户进行了大量报表工作。有 10 个预订联机服务的客户。但是，每个季度还为客户生成大约 1000 份执行摘要。另外，我们给许多客户提供来自 AS400 和关系数据库的详细文件。

执行摘要经过最近刚由 Applied OLAP 实施的自动批过程，为 Essbase 提供一个前端。此自动化批过程可望在第一年为我们节约大约 30,000 美元。这将是一笔相当大的经营费用。 ”

9. 您的工作小组的构成如何？

GT：“ 在关系端，我们有一个开发人员和一个 DBA。该 DBA 实际上分担 Oracle 和 AS400 的工作。我们有二人 OLAP 小组。我负责总体管理和 OLAP 设计，而 Essbase 开发员则管理所有维护和支持过程以及一些像分级这样的开发工作。 ”

5.2.2.2 设计

1. 您的 OLAP 需求来自何处？

GT：“ 我们首先考虑商业情况。 ”

2. 开发新的 OLAP 模型时，您使用什么技术？

GT：“我们在设计新模型时，所做的第一件事情就是与商业用户坐下来讨论。我们会进行一个简单会话，该会话将建立依次定义维的商业规则。

通常，我们会先看一看度量，再确定度量的价值。然后按照其商业视图和维定义此度量。接下来讨论维之间如何进行交互，看是否存在像我在前面讲述制造维时提及的示例那样的机会。在分析中，还发现制造商与低级别药品编号之间有着明确的关系，这样药品编号就可以成为属性，这是利用此技术特殊功能的另一种方法。我们在经过分析过程时设法开发商业所有权。一旦完成这些分析任务，生成样本文件和使用应用程序管理器建立维、加载模型以及在极短时间内提供第一个概念证明就变成一个非常简单的过程。由于可以把所有源数据放入某种形式的关系数据或某种形式的文本数据之中，我们获得了有关专门知识，并了解到如何利用 Essbase 规则的用途来获取这些源文件以及非常迅速地从中创建模型。

由于了解到商业需求和维，所以可以非常快速地生成原型。我们把这些原型交给用户，让他们先行使用。

3. 您如何从原型发展到生产模型？

GT：“发布新模型之前，我们会考虑生产中的两个方面。首先，验证信息的准确性。我们用大量例程确保 OLAP 数据符合其他索赔和金融数据，这样提供给客户的数据就能够保持一致和准确。第二个方面是管理。我们估算文件大小并支持资源需求。由于具有丰富的经验，所以估算非常准确。”

4. 您如何验证多维数据集？

GT：“我们进行一些自动化验证。我们有若干均衡例程，它们从 AS/400 开始，一直到底，最终以 VBA 例程结束，而 VBA 例程进入并执行每天和每月的验证任务。发布多维数据集之前，我们会尽最大努力在所有维之间进行平衡。此过程非常稳定。我们没有遇到太多问题。”

5. 开发应用程序时，您经过什么调节步骤？

GT：“我们确实仔细地检查了稀疏/紧凑和块大小，为索引和数据高速缓存做准备。我们始终确保为服务器正确配置高速缓存，这样就能在计算方面获得最大优势。我通常遵循久负盛名的 DB2 OLAP 块大小构造方法，这似乎非常可行。处理响应时间方面的注意事项是动态计算的使用。例如，有一种动态计算非常耗时，因为它在各个维的若干小节之间进行。一方面节约时间和空间，另一方面检索时却会出现响应时间问题。我们根据应用程序的不同进行权衡。

我们所使用的普遍认可的规则之一是使紧凑块内所有计算成为动态计算，而且不使任何稀疏维成员成为动态成员。

我们还在大纲中嵌入公式，这比把公式嵌入计算脚本更有效率。”

6. 您已形式化了这一知识，是以标准和一览表为依据，还是基于经验？

GT：“我在企业内部互联网上开发了一个知识库。在其中写入有关我所知道以及所学到的所有知识的说明。我们正在用相关文档和信息充实此知识库。”

7. 请问如何使用全部维操作新模型？

GT：“我们有单独的维构造和数据加载摘录。通过两个不同过程进行该项工作，因为我们每月更新输入源文件总共有大约 700MB。试图将全部 700MB 都运行一遍来选出并更新维是一项非常耗时的任务。相反，我们建立若干 SQL 例程，这些例程查询关系数据库会发现放入格式化输入文件的新月度输入项。把此文件用作每月过程的组成部分，以便在加载数据之前更新层次结构。我们将处理一个效率高但却小得多的层次结构文件。

我们使关系数据库执行尽可能多的数据转换工作，从而使加载规则非常简单。”

5.2.2.3 客户机工具

1. 您使用什么客户机工具？

GT：“我们利用 Applied OLAP 进行了一些 VB/Essbase 开发工作。这是产生于这些应用程序的另一项演进过程。我们针对小型组执行摘要和医生报表卡生成报表。它们可能是客户摘要，也可能是按某一时间段医生配发的药品摘要以及关于医生报表卡的摘要。医生报表卡是该医生与本专业领域其他医生的比较。这些报表是通过

FoxPro 生成的。

仓库小组的目标之一是迁出该结构。因此，大约一年半以前，我们开始寻找快速开发内部过程的方法。我们使用电子表格工具箱和 VBA 宏编制应用程序，并用了几周的时间才完成这项工作。然而，该应用程序的维护和支持成本非常高，因为需要开发人员进行手动操作，因此，我们发起对该应用程序进行改进的另一个项目。与此同时，寻找如何实现在 Web 上得到类似于 Excel 的功能，其中包括访问多维数据集。

我们发现 Applied OLAP 可提供所需的一切功能。去年三月，我们利用 Applied OLAP 启动一个项目。九月，开始投产。整个过程得以简化。现在，我们采用前端工具，它使我们的用户可以维护 MS Access 数据库中的报表规范。在那里，生成 XML 页，这些 XML 页按照 Applied OLAP 自定义报表应用程序运行，产生我们分发的数千份报表。账户管理员输入新客户的账户信息，该信息自动流过此过程。‘生产控制’重新生成 XML 以便自动更新检索列表，然后就可自动地成批运行所有请求。为了赋予客户良好的灵活性，我们可以打印报表、用电子邮件将其发送出去或将其存储在特定客户公告栏上。这种电子交付包含 Excel 工作簿。客户可以进入该工作簿使用更多功能，而无需重新键入打印结果信息。

事实上，客户还可以进行直接联机访问。我们 1998 年曾经与 Painted Word 有联系。他们构造了基于 Web 的 Java 应用程序，使外部客户能够进入并同样使用 Excel。该应用程序具有有限的 Excel 功能，备有一个电子表格网格。一旦将客户导航到 OLAP 多维数据集内足够低的层级，就可以选取度量，并可深入获取生成该多维数据集的关系细节。凭借此应用程序，我们在三年前获得了巨大优势。”

2. 谁设计您提供给客户的报表？

GT：“内部药剂师和账户管理员设计原始报表。附加价值的方法之一就是让门诊人员吸收进来。他们以药剂学或临床的观点从不同角度进行分析。另外，我们提供财务和索赔处理报表。有些报表是索赔处理系统的标准输出，但是，一般来说，商业用户根据客户需求设计报表。”

5.2.2.4 管理

1. 您如何管理安全性？

GT：“我认为管理安全性不是一件小事。我们的问题是向客户提供通过前端应用程序的基于 Web 的访问。我们必须将其访问限制在 OLAP 多维数据集的范围之内。

例如，有些客户属于第三方管理员。他们可能有多个账户，其中有可能是竞争者。我们必须通过 Web 接口创建安全方案，该方案可限制这些账户查看其他数据的能力。OLAP Server 内的问题是，如果您访问某个账户并缩小，然后返回载体并放大，您就会看到所有账户。我们在 Web 接口范围内的解决方案是引入一个可处理应用程序安全性的关系表，从而限制客户查看数据库。

我们的 OLAP 安全系统无法与其他系统进行集成，可是，我们期望实现单点登录。”

5.2.2.5 操作

1. 您采用什么步骤将一个新应用程序投入生产？

GT：“在应用程序开发过程中，我们可以在两个星期内(经常在一个星期内)应用完新应用程序。但是，项目生命周期中还存在许多其他问题。所以开发周期可能会持续两到三个星期。

绝大部分开发工作是由 OLAP 工作小组处理的。我们主要负责仓库的生产端，坐下来讨论并进行总结。首先，对应用程序进行检查。然后，让几个商业用户对其进行观察。我们还更新自动平衡例程验证数字，并检查美元数额是否有误。通常情况下，用户知道应用程序从原型阶段进入生产阶段的唯一可辨认方法是看所有数据是否已都处于生产阶段。我们根据模型和生产环境的状况发出规则信息。”

2. 您如何管理备份？

GT：“用基于 Veritas 产品进行磁带备份。每个周末均进行备份过程。有时，备份过程与生产重叠进行，因为有些计算需要的时间较长。如果某个多维数据集正在进行计算操作，并因此而处于活动状态，就不会备份该多维数据集。我们针对该多维数据集另外运行备份。当每月第一周执行前一个月的更新时，通常会发生这种情况。由于目前的服务器存在 I/O 问题，所以我们对批计算持谨慎态度。

我们在 AIX 平台的系统测试成功运行了大型数据库的多个计算。在目前的生产环境中，运行大型计算，然后同时运行较小的计算。如果某个模型驻留在单个逻辑驱动器上，我们就可以配合跨越驱动器的模型计算该模型。在跨区驱动器上运行多个计算或多个模型，到了 Alpha 服务器上则不太顺利。”

3. 您是否有灾难恢复计划？

GT：“我们有非现场存储区。我们定期拍摄非现场快照。这样做可能无法进行完全灾难恢复，但即使最坏的情况发生，我们都可以在几个星期内恢复生产。”

4. 您采用什么策略应用修复包或修补程序？

GT：“如果确实需要更新，我们会进行更新；但是，如果所处的级别非常稳定，则保持不动。当某种级别的软件在生产中运行时，我们运行的修补程序级别可能相当落后。对我们而言，必须拥有相当显著的增强功能，以便于顺利升级。我们在当前级别进行转换，但是，一旦 AIX 生产处于稳定状态，就会升级到最新版本及最新修补程序。

除非对于我们没有的功能有紧急需求，否则，我们不会进行更新，直到发现确实有必要时才会更新。”

5. 您应用新版本的策略是什么？

GT：“我们始终留意新版本，而且通常会发现一些值得进行升级的重要的新功能。但是，我们并不急于升级，而是等待需要某种新功能的现实需求出现。”

6. 您如何管理客户机工具升级？

GT：“升级意味着交出应用程序管理器、交出电子表格客户机以及交出我们集成到前端应用程序中的其他一切，而且还得确保升级后一切都正常工作，因此，对于一个小型机构来说，升级是一项重大决策。我们有支持此过程的用户和应用程序列表。”

7. 您的批过程是由什么构成的？

GT：“ 我们有一个二层系统，每月和每日。

我们按区域细分详细的每月模型。多数内部处理不需要相同级别的详细信息，因此，我们建立另一模型，它包含所有区域、所有客户机，但却没有像其他维中那样多的详细信息，这样就可以进行每月报表。此数据库通常在次月 2 日准备好。许多情况下，当月的第一天就可以使用。另外，我们还有合并到每月多维数据集中的每日模型。

每日过程在当月最后一天运行。然后，关系数据库运行一系列例程，这样，到次月第一天半上午，我们交出所有层次结构文件和数据文件。然后，我们就可以开始更新过程。通常情况下，到半下午，我们已经交出内部模型，以便审阅每月数据。

我们不会因更加频繁地进行更新而感受到许多压力；在处理每月模型和每日模型之间，我们满足了需求。无论何时有人想进来检查与前月相比当前月趋势是否良好，均可通过每日模型进行检查。每天，直到午夜才通过系统处理的索赔在夜间都被移至关系数据库中，然后对多维数据集源文件进行格式化，并在早上提交给我们。我们在白天加载这些文件，并在第二天晚上运行计算。”

8. 您如何实现 OLAP 操作自动化？

GT：“ 关系数据处理、文本文件抽取、维构造、数据加载都使用自动化软件例程，这些例程从 AS400 获得数据、将其加载到数据仓库，然后为 OLAP 多维数据集提供平面文件。我们有若干数据仓库以不同方式编写的摘要文件。我们有成员，以及存在于成员导入中拥有处方的人员。根据使用某些药品的人数、利用某种服务场所(例如，某一流动服务、某种长期设施或邮购设施)的人数，上卷那些人员。然后，还按医生上卷，这样就可以了解哪些医生开的哪些药品。这是另一组信息，都是非常小的文件，但这是能够把另一级别数据合并到 OLAP 模型中的另一种方法，即另一种摘要级别。我们针对大纲变动的更新过程，其绝大部分是手动进行的。我们有 Painted Word 提供的名为 Clockwork 的产品，这是一个运行数据加载和计算的 OLAP 批加载程序。在 Clockwork 内，还有用于夜间计算的自动开始时间。”

5.2.2.6 访谈结束

1. 您认为您正在做的世界一流的事情是什么？

GT：“首先是，我们能够导入许多不同的数据源、对其进行换向，然后生产出内部人员可以查看的 OLAP 多维数据集。我们的 Web 接口使客户能够使用一个基于 Java 的用户界面，通过深入关系数据库或使用基于 Excel 的 Web 应用程序，进入多维数据集和分析数据。也就是说，客户只需一个浏览器就可以找到所需数据。

在操作端，我们每个月都在运行。这是包含大量数据的复杂生产环境，但是该生产环境工作状况非常好，很少发生停机事件。我们的操作人员明白可用性非常重要。我们使可用时间保持在 90% 以上。”

2. 为了改进 OLAP 安装，您将会进行的最重要的变革是什么？

GT：“我们将向客户提供他们在 Web 世界中所需的性能水平。”

5.2.3 Mark Rich 访谈

Mark Rich 在 IBM 从事财务和商业分析工作已有 16 年。1998 年，成为董事会成员，负责领导 IBM 的“世界规划系统”，该系统目前在 IBM RS6000 技术(APEX 项目)上以 IBM DB2 OLAP 为基础。

5.2.3.1 环境

1. 描述 OLAP 环境。

MR：“我们在 SP2 上运行 AIX。

AIX 赋予我们可扩展性。在 RS/6000 平台上，我们可以不断添加处理器、内存和磁盘，而且操作系统完全支持高端系统。当然，会受到所拥有的特定系统的限制，这就是 SP2 群集所要承担的任务。

由于是 IBM 系统，我们必须遵守严格的安全原则，而 NT 几乎不可能给予所需的安全性，AIX 具有我们所需的支持。

我们把 RS/6000 节点群集到 SP2 之中。利用 SP2，我们能够使用多个节点、在世界各地维护的机器，并使其成为一个综合体。这使我们可以通过安排专人控制和管理整个 SP2 综合体，来提供 24X7 支持。

我们有通过 SP2 内的一个入口点即可操作所有软件、所有安全更新、所有操作系统的工作站。”

2. 您如何利用多个处理器？

MR：“SP2 本身被分割成若干 RS/6000 的群集，以便于管理。利用此配置，我们能够运行 24 X 7 X 365，从而获得读取能力。而且，除非出现问题，每个模型均可用于每天 24 小时更新。如果不支持更新，就会出现一个非常小的窗口，持续 10 或 15 分钟。这取决于多维数据集的大小。我们必须脱离三个镜像之一，以便进行备份。一旦完全脱离一个镜像，就只有两个镜像，但可再次变为可写。”

3. 现在节点的情况如何，是不是高端节点？

MR：“该系统已使用约三年的时间。主要群集使用 604E 个节点(2-4G 内存、215G 存储区用于 OLAP 多维数据集)，采用三倍镜像。我们目前正在利用具有 375 或 400MHz 处理器的最新技术构建一个全新的群集，该群集高达 24 个 CPU、32G 内存，以及每个物理节点 400G 可写 DASD。”

4. 您怎么决定在每个节点上运行什么？

MR：“我们总共有 28 个节点。其中，4 个节点使用 HACMP 功能进行备份。两个节点用于开发，这包含我们和开发小组进行的系统集成测试以及获取新代码时的操作。其他节点分散在 4 个不同生产群集，具体由 IBM 商业小组进行分配：一个用于软件组、一个用于服务器组，一个用于 PC 公司。然后，还有按地理位置分配的节点，例如，EMEA 有 2 个节点。我们为公司 CHQ 模型分配了 3 个节点。

我们进行基准测试以决定如何根据以下几点进一步分配节点：特定模型使用多少 RAM、使用多大 CPU 或使用多少个 CPU 以及在多维数据集本身内运行多少个不同函数。

因此，我们按行业平衡，然后根据资源需求进一步平衡，但不会随意平衡。那会导致需要 2 台或 3 台 SP2。

5. 开发和测试情况如何？

MR：“现在，SP2 综合体中有 2 个节点，我们使用它们进行各种开发和测试活动。理想的情况是，不应该在生产综合体中进行开发工作，因此，我们也从旧系统级联机器。这些机器不在 SP2 综合体内管理，但开发小组使用它们，而且我们在其中做多数开发工作，比如建立新模型、测试新功能 程序。”

6. 您的源数据来自哪里？

MR：“输入数据为量度，与会计数据一致，来自数据仓库，但我们的模型主要用于规划和预算，因此，多数数据是分析师在模型中创建的。”

7. 您目前支持多少个多维数据集？

MR：“我们有 490 个多维数据集。

最小的多维数据集在 200 M 到 300 M 之间。这些多维数据集很小，极易操作。迄今为止，最大的多维数据集有大约 70 GB。我们把它从原来超过 100 GB 减小到现在的大小。

- a. 是否减少多维数据集的数量可能会使整个事情更容易管理？

MR：“是的。尽管我们在 SP2 中安装了许多 CPU，但最大的问题仍然是性能问题。如果我们减少多维数据集的数目，就可以更好地利用可用资源。”

8. 多维数据集支持什么应用程序？

MR：“我们的多维数据集都是财务方面的。这些多维数据集用于测量报表，多数用于 IBM 财务规划。

下一年的主要任务之一就是了解为何有这么多的多维数据集。我们认为很多模型非常相似，可以跨部门合并。我们希望能够利用应用程序分区获得具有四五个维的源多维数据集，并获得通过完全相同的四五个维连接在一起的四五个多维数据集，然后再有一个额外的维，而且是唯一的。

我们想调整普通多维数据集，希望能把数量减少到 200 个左右。

9. 您支持多少用户？

MR：“我们就多少个人可以连接，对每个节点设定了限制。针对其中 10 个节点，设定 100 个并行用户的限度。在另外 10 个节点上设定了从 100 到 175 不等的并行用户的限度。我们有一个具有多达 250 个并行用户的节点。我们不按模型限制用户数量，而是按节点限制用户数量。

我们在全世界的用户数量已达 6000 个。在群集内，我们可以支持 1,675 个并行用户。”

10. 您的工作小组的构成如何？

MR：“开发人员目前大约有 10 人。随着高级用户基数的增长，核心开发小组正在逐渐变小。有些卫星区域(如软件小组和全球服务小组)各有 2 到 3 名开发人员。他们做自己模型的开发工作，只有在有问题的时候才来找我们。

在 10 个开发人员之中，有 2 人几乎没有 OLAP 方面的知识。他们负责开发产品周围工具。他们熟悉 API 和编程，但不建立模型。

我们有 4 个人负责系统维护和支持工作，前面提到的两人负责 OLAP API 开发，另一个人负责相关的开发工作，这 3 个人确实是优秀的 OLAP 建模人员。

因此，我们有技术力量雄厚的核心小组，还有分布在各个商业单元中的模型设计员和优秀的模型开发人员。”

a. 您是否还负责数据仓库？

MR：“数据仓库由另外一个小组负责，但是一旦进入我们的模型，那就是我们的责任了。”

b. 您与仓库小组的关系如何，关系好吗？

MR：“关系很好，但实际情况是，任何工作小组之间都有完全不同的重点。仓库小组与我们合作并帮助我们高效地处理数据，但彼此的重点是不同的。”

5.2.3.2 设计

1. 您的 OLAP 需求来自何处？

MR：“许多 OLAP 需求来自商业用户和业内。我们工作小组可能经常与业内不熟悉 OLAP 的人有来往，这些人想要 27 个维和第 n 级详细信息。最难处理的事情是让人以多维方式思考，即使是高级用户和开发人员，当他们开始思考模型时，与最终用户思考方式一样，让人理解什么是维需要很高技巧。一旦找到一组恰当的维，事情就相当容易了。”

a. 您是否依靠高级用户、开发人员和设计人员了解维？

MR “良好的起点是利用现有模型。然后，他们提出需要在模型内某个点添加信息的附加请求。这就是请求通常开始的情况。”

2. 开发新的 OLAP 模型时，您使用什么技术？

MR：“良好的起点是利用现有模型。然后，他们提出需要在模型内某个点添加信息的附加请求。这就是请求通常开始的情况。”

我们希望他们理解我们想要向他们表达什么意思，反之亦然。我们花费大量时间做的事情是，模型在 Lotus 电子表格中是什么样子以及他们如何获取其信息。”

- a. 因此，您非常迅速地建立一个电子表格原型，并让他们在 Lotus 1-2-3 中观察？

MR：“ 是的。从那里我们快速建立一个原型大纲，并让他们开始切片和划线。”

- b. 如果我想要新模型，我们共同努力并提出认为合理的设计，那么您是进行评估过程，还是直接建立多维数据集？

MR：“ 很遗憾，我们尚未找到一种进行有效估算的方法。我们随时使用 Hyperion 提供的标准以及在开发中积累的知识，只是随手拿来而已。”

- c. 在构造第一个模型时，技术设置方面您优先考虑哪些事情？您是否最担心块大小，其次是稀疏/紧凑，具体有多详细？

MR：“ 我们考虑识别维的问题。然后，观察这会发生什么变化以及变化有多么频繁，并将此作为选择紧凑和稀疏的因素。我们通常没有太大灵活性。有些模型的块大小超过 300K。”

- d. 您是否使用动态计算？

MR：“ 我们使用动态计算，但只在块内进行。我们知道也可以在稀疏维上使用动态计算，但那会影响检索性能。高级用户与最终用户一起进行平衡，并使检索时间为各个最终用户所接受。”

- e. 您是否用一览表将这类事情形式化？

MR：“ 很遗憾，我们很少就此进行形式化。建立一个模型后，我们不会进行复查，确定该模型是否对这个组有效率。谁操作这个模型，谁就负责这项工作。除了供应商传来的资料以及技术会议上发的资料之外，我们实际上没有什么书面的东西。”

- f. 那么您是依靠员工所具有的知识？

MR：“ 是的。我们仍然认为建立多维数据集与其说是一门科学，道不如说是一门艺术，我们还不能把什么事情形式化”。

- g. 当新模型经过设计过程时，您是否设法测试查询响应时间？还是把这件事情交由用户自己去做？

MR：“ 如果让我们做这件事，我们会接受这项任务，但是我们的工作主要是

管理生产端。只是确保可以交付模型，其余工作由高级用户去做。要知道，我们生产 490 个模型，而且这些模型会发生变化。高级用户具有数据库管理权限，而且给每个节点安排一个管理员。他们负责随时修改多维数据集，我们不会返回去定期复查。一旦开发好模型并将其投入生产，我们就不再返回去检查它们的情况。”

3. 您如何从原型发展到生产模型？一般来说，您需要多长时间开发和应用新模型并将其投入生产？

MR：“如果是基于现有大纲的新模型，大约 75% 可重复利用，所以说，我们最快需要一个星期。我们愿意多花些时间，但是 4 个星期很可能是最大限度。”

4. 您如何验证多维数据集？

MR：“对于加载数据的模型，我们验证输入数据。对于纯粹的规划模型，数据由分析师输入，高级用户负责验证这些模型。在结束规划方案或规划周期之前，必须对其进行验证。这由规划师、分析师和高级用户负责。然后，IBM 财务人员验证模型内的数据。我们没有验证程序，因为这项工作由 IBM 的内部分析师和财务人员负责管理。”

5.2.3.3 客户机工具

1. 您使用什么客户机工具？

MR：“所有用户均运行具有电子表格插件的 Lotus 1-2-3。我们也利用 Alphablocks 及其新的“电子表格框”(Spreadsheet Box) 进行一些工作。迁移到 Web 显然是一件非常有趣的事情，而且汇编网页的功能可显示多个多维数据集以及一页内的其他数据源。”

2. 您是否发布有关数据的报表？

MR：“我们没有连接系统的 EIS 型接口。用户群体是由财务分析师和规划师构成，而不是由高级管理人员构成。”

5.2.3.4 管理

1. 您如何管理加载更新？

MR：“我们操作的某些多维数据集已滚动 12 个月的周期，我们可能会在年底进行一些大的更改，到时必须重新设置一切并重新开始。好的消息是，除了约 20 个

模型外，很少有开发端的参与，由管理员负责管理更改过程。通常，为新周期做准备的最好方法是制作当前模型的副本。这会自动复制所有最新更新。比如，分析师的最新输入、最新的安全过滤器，等等。然后，他们可以更新模型，并使其为新的财务周期做好准备。有这么多大型模型，确实使我们的资源承受巨大压力。我们核查模型使用情况以及存储设备使用情况，并尽量使资源保持高效率。在转换过程中，我们经受许多压力，因为用户暂时需要两套模型。”

2. 您如何管理安全性？

MR：“ 由于有 6000 多个用户，安全性对于我们来说是一个真正的问题。例如，当开始新的财政年度时，我们必须创建许多新模型，而且每个模型都应具有与上年同样复杂的安全模型，包括全年开发的所有过滤器和规则。我们必须把整个事情从一个模型移动到另一个模型。高级用户和管理员在安全和管理方面花费大量时间。

我们正在研究一个应该会有帮助的内部系统。每个用户在 IBM 内具有唯一标识符，而且我们还将多维数据集之间、Web 的报表库上针对应用程序管理器功能，以及电子表格插件维护这些标识符的安全。他们将只用一个密码来管理所有访问。”

a. 也就是说，您正向单点登录转变？

MR：“ 是的，那是我们想要达到的目标。”

b. 创建过滤器、将人员添加到组以及人员离开时将其删除，这方面所有的安全管理工作都是由控制每个节点的高级用户进行的，是这样吗？

MR：“ 是”。

c. 您是否将此安全系统与其他安全系统进行集成？

MR：“ 是。CHQ 系统与用于财务会计数据的数据仓库进行集成，这样，人们就可以通过中心请求访问，该中心管理全球所有对 IBM 会计数据进行的访问。这是一个高度安全的系统。我们自己就可以做到访问其他模型，但需要通过适当的渠道。”

d. 对 OLAP 安全系统的实际更新是否是自动化的？

MR：“ 不是，遗憾的是，我们采用手动方法。”

5.2.3.5 操作

1. 您如何管理备份？

MR：“ 我们使用三倍镜像，而且将一个镜像脱机，以便进行备份。我们使用 ADSM，它运行在一个节点上，因此，每个群集有其自己的 ADSM 设置。备份作为批过程的组成部分在夜间运行，但是，由于我们有支持全球不同地理的群集，所以可根据地理位置进行备份。亚太地区的备份在午餐时间运行，欧洲的备份在下午 7 点运行。

我们使用“开始存档”功能将模型置于只读模式，然后使镜像脱机，运行档案，再使镜像返回到联机状态，并发出“结束存档”命令。”

a. 备份的粒度如何，您是不是只备份数据？

MR：“ 我们备份写到磁盘的而不是写到操作系统的一切信息。我们也可以撤销，例如，我们有一些不改变的历史模型。我们可以针对这些模型进行一次备份。”

b. 您是否进行增量备份？

MR：“ 不进行增量备份。即使丢失整个节点，我们需要能够迅速恢复，因此，我们进行完整备份，这样，甚至可以把完整备份重新加载到 SP2 综合体内的另一个节点。如果丢失一个磁盘，我们有三倍镜像。如果同时在多个地方丢失多个磁盘，则必须转到磁带备份，但迄今为止尚未遇到这样的问题。因为有 HACMP，如果由于硬件故障而失去节点，HACMP 就会接管工作。”

c. 那么，如果您失去整个节点，您恢复模型的速度有多快？

MR：“ 如果我们丢失整个节点，却没有失去 DASD，仍然可以备份模型，而且是在毫不知晓的情况下进行。HACMP 可以直接接管。但是，如果我们失去 DASD，或者数据库被毁坏，那就得从磁带重新加载该多维数据集。对于特大模型来说，可能会花费 3 到 4 小时；而小型模型，则需要 20 到 30 分钟，重新加载时间受磁带速度限制。”

2. 您是否有针对 OLAP 的灾难恢复计划？

MR：“ 我们确实研究过这样的计划，但目前那样做不划算。”

3. 您采用什么策略应用修复包或修补程序？

MR：“ 原来的想法是每六个月升级一次，仅此而已。现在，已经过去六个多月了，还没有应用任何服务，我们很可能在下一个版本出来后才进行升级。六个月以前，升级了每一个修补程序，因为当时确实需要修补。我们需要了解当前水平的局限性以及围绕什么进行工作。如果可能的话，宁愿不作改动。由于这个原因，我们没有使用一些重要功能，例如，尝试过使用应用程序分区，却发现使用该功能会导致问题。问题用后来的修补程序得以修复，但在此之前，系统一直非常稳定，所以我们不想冒险。很早就留意主要版本，包括运行 版。我们必须决定是否要有充足的新功能，证明有必要进行包括系统集成测试周期在内(这会需要长达两个月的时间)的升级，然后再对所有生产节点和 6000 个用户进行更新。如果断定升级有意义，就会过一遍测试过程，直到找到可保持稳定的级别。理想的情况是，在生产中使用新版本三个月，但我们还没有使用过。找到功能性与稳定性的恰当组合花了我们将近八个月时间。 ”

4. 您使用什么工具运行系统？

MR：“ 确切地说，我们依靠代码。这些代码是依据计划和管理进程的 AIX 设备开发的。我们使用 Tivoli 和 ADSM，以及将其捆绑在一起的客户代码。 ”

5. 您如何监控批过程？

MR：“ 我们有 24x7 操作支持，因此，一旦有异常情况，总会有人收到讯息。如果公司模型的备份失败或中止，我会得到通知。 我每天都收到每日状态报表，因此可以了解各种情况如何注意以及哪些方面需要加以注意。 ”

a. 您说过，您的目标是 24x7x365，您认为您实际做到了吗？

MR：“ 就读取功能而言，迄今为止我们做到了。除一些问题外，实际上已达到目标了。 ”

b. 您多长时间会因为出现问题而使工作变慢？

MR：“ 大概一个月会发生一次。有时，由于发生某种软件问题或设计问题，会频繁一些，例如，当试图把应用程序分区用于一组模型时，我们必须每天回收一个节点，持续八天，直到解决问题。 ”

5.2.3.6 访谈结束

1. 您认为您正在做的世界一流的事情是什么？

MR：“我们所做的一切工作都是通过 SP2 完成的。我认为，整个过程是最先进的，但需要花费时间才能达到这种水平。我们必须逐步培养各种技能。我们也是将大规模企业 OLAP 用于规划、预测和预算的一个很好的范例。我们的模型不是用于报表，而是用于世界上最大的公司之一的分析、规划和预测工作。我们有 6000 个用户，他们不是查看报表的 6000 个人，而是不断更新数据和运行方案的 6000 个用户。”

2. 为了改进 OLAP 安装，您将会进行的最重要的变革是什么？

MR：“我们将应用 OLAP Server 的新版本，而且还要更新硬件。同时还希望应用集成服务器。这将极大地帮助我们数据仓库中建立模型。”

5.2.4 Joe Scovell 和 Jacques Chenot 访谈

Joe Scovell 和 Jacques Chenot (JS&JC) 两人都在堪萨斯城 DST Systems 公司工作,该公司是共同基金行业的一家过户代理。Joe Scovell 担任客户服务经理,拥有 13 年与许多的共同基金客户合作的经验。其专业知识集中在向客户提供使其能够智能挖掘数据的增值服务。

5.2.4.1 环境

1. 描述 OLAP 环境。

JS&J: “我们在两个服务器上运行 Windows NT。它们都是 4 路 400 MHz Pentium。每个服务器拥有 1 GB 以上 RAM。

一个服务器用于生产,而另一个服务器则是一个测试环境。

选择 NT,是因为这是我们绝大部分专业技术和知识之所在,因此,在该环境中工作非常得心应手。随着商业的发展,我们所使用的所有软件都可以扩展到 UNIX 和 AIX,所以有可能迁移到大型平台。”

2. 开发和测试情况如何?

JS&JC: “我们在测试服务器上进行大部分开发工作。如果只是粗略了解,那么有另一个可以作为 OLAP 服务器工作的机器。”

3. 您的源数据来自哪里?

JS&JC: “我们有一个非常强大的数据仓库环境。我们可以从基于主机的系统获得源数据,也可以从 NT 上 DB2 通用数据库的数据仓库获得源数据,用户也可以使用该数据仓库进行限席查询和报表。”

a. 您是直接从 SQL 接口抽取数据,还是拍文件快照?

JS&JC: “我们可以根据需要采用这两种方式。

我们比较喜欢使用 SQL 接口。我没有注意两种方法各有什么优缺点,SQL 接口只不过是一种寻找数据来源的更加集中的方式。”

b. 您用数据仓库做多少工作来使 OLAP 获取数据?您是否建立用于加载数据的特殊表和特殊视图?

JS&JC: “不,我们用 SQL 进行绝大部分转换工作。”

4. 您使用什么存储设备，如何使用？

JS&JC：“主驱动器为 RAID 1，数据和索引驱动器为 RAID 5。它们都有 100 GB 以上的 DASD。”

5. 您如何优化服务器资源分配？

JS&JC：“用户具有良好的响应时间，因此，我们在加载和计算过程中集中精力分配服务器资源。我们已经学会如何通过需要在运行的所有作业之间平衡加载，从而最好地利用可用 CPU 和内存。

我们编写了所有加载和计算的脚本，并作为脚本的组成部分，根据每个多维数据集的需求分配可用内存。这是在各组加载和计算开始之前动态完成的。脚本把高速缓存设置为获得最佳计算性能，然后，完成整组计算时，把高速缓存重新设置为支持查询所需的级别。通过在同一组内同时运行恰当的多维数据集，以确保没有过度调配内存。如果内存够用，每组可运行多达 4 项并行计算。”

- a. 您有时运行四项并行计算，有时运行一项或两项，具体取决于它们需要多少内存？

JS&JC：“没错”。

- b. 最大并行数目取决于可用的 CPU 数目，而且您的脚本根据正在进行的操作动态地向 OLAP Server 高速缓存分配内存？

JS&JC：“的确如此。”

6. 您支持多少个多维数据集？

JS&JC：“生产服务器上有 27 个多维数据集，测试服务器上有 21 个。这些多维数据集目前从几 MB 到 4 GB 不等。”

7. 多维数据集支持什么应用程序？

JS&JC：“所有的多维数据集都用于财务分析。”

8. 您支持多少用户？

JS&JC：“我们总共约有 85 个用户，并且可能有 10 到 15 个用户同时进行登录。而且这一数字还在迅速增加。”

- a. 您支持更多客户和更多用户的策略是什么？您是否还会添加更多 NT 服务器？

JS&JC：“我们正在评估 OLAP 体系结构以获得最佳性能。无论是继续使用 NT 服务器还是使用大型 UNIX 服务器，都将在分析中确定。”

- b. 促使您添加更多服务器的关键因素是什么，是计算时间、用户数目，还是您获得的物理分区？因为您可以由一个服务器服务于一个客户，而由另一个服务器服务于另一个客户。决策中哪一项会被视为决定因素？

JS&JC：“上述各项的组合。”

5.2.4.2 设计

1. 您的 OLAP 需求来自何处？当引入新的 OLAP 多维数据集时，您经过什么规划过程？

JS&JC：“这基本上由我们的客户决定。根据他们所提出的需求、根据其商业智能化需要。”

- a. 商业用户了解 OLAP 吗？

JS&JC：“是的，他们了解。”

- b. 也就是说，您有一个经过训练的商业用户群体，他们给您提供需求？

JS&JC：“是的。”

- c. 您是主动去寻找新机会，还是商业用户群体已提供了足够多的机会？

JS&JC：“两种情况都有。我们找商业用户商谈，告诉其我们拥有信息的可用性。我们倾听其所需，并根据这些需要决定新机会是什么。”

- d. 不懂技术的商业用户有时会认为他们需要每个可用维，所有维均是详细级别最低的维，并都在多维数据集之中。您是否接到过这类请求，如果是，您如何处理？

JS&JC：“我们知道数据仓库的功能是什么。我们详细向其解释功能以及用户收到较大多维数据集时的响应时间。我们有用来确定多维数据集可以有多大以及可提供多大粒度的原则。”

- e. 也就是说，有一个教育过程，并且您提到原则。您可否谈谈这些情况？

JS&JC：“多数客户采用我们已开发好的多维数据集，来满足可能有的市场和财务需要。他们可进行修改满足其具体需要。我们根据模型的具体情况把维

数目限制在 7 到 10 个之间。我们不经常变化，因此这些原则是已创建的基本多维数据集。”

- f. 实际上，您创建了客户可以进行一定程度修改的模板。

JS&JC：“是的”。

2. 开发新的 OLAP 模型，即没有模板模型时，您使用什么技术？

JS&JC：“我们有已创建好的调整大小的电子表格，可输入存储成员的数目，并根据公式，得出估算的 DASD 空间。”

- a. 如何找到恰当维？

JS&JC：“我们非常熟悉数据以及商业领域，因此，我们将创建一个初始大纲，然后以此为基础，致力于得到一个合适的块大小和合适的密度”。

- b. 您是使用纸上设计，还是使用应用程序管理器？

JS&JC：“我们快速进入应用程序管理器，并把它用作重复设计工具。”

- c. 开发初始大纲通常需要多长时间？

JS&JC：“初始大纲开发很可能是整个开发过程中耗时最短的部分。它满足了客户在时间方面的实际需求。我们需要经过一段快速调整大纲的时期。与此同时，客户逐渐熟悉了 OLAP 环境以及具体将要做什么。一旦客户看到多维数据集集中的数据，并学会如何操作，他们就真正开始思考。他们可能会判定目前的设计完全不是他们所需要的，而且将从头设计一遍，直到满意为止。他们可能会需要添加几个现有维、新的事情、新的度量，等等。根据应用程序不同，开发和投产时间会需要二至四个月。最后，客户通常非常满意最终结果。我们没有从生产中抽取过要修改的多维数据集。”

- d. 您尽可能快地使用技术，而不是进行抽象的设计，利用技术解决设计。

JS&JC：“是的，这似乎是一种最可行的方法。给他们可以看到并可以使用的东西，而不是研究抽象的设计。”

- e. 建立维时，尤其是建立新模型维时，您是否直接从仓库环境建立？

JS&JC：“是的，我们从仓库环境建立维，然后根据每个客户的需要进行更改和添加层次结构。例如，每个客户很可能有一种不同的进行公文包分类方法。仓库或内部系统中可能没有这些分类，但只要我们以较低级别加载数据，就不会有问题。”

3. 开发应用程序时，您经过什么调节步骤？

JS&JC：“除稀疏/紧凑和块大小之外，使用动态计算非常重要。将紧凑维中的父级标为动态并把公式标为动态，有很大帮助。可以把精力集中在计算时间上，因为我们希望批过程尽可能小些。我们尚未发现采用动态计算查询响应时间会增加的现象。”

- a. 您是使用书面一览表，还是根据自己的丰富经验确定块大小、大纲顺序、动态计算这样的事情？

JS&JC：“我们没有一览表，因为我们遵循预先创建好的大纲，尽管设计得不一定完全精确，但非常相似。所有维都按照最终执行的方式进行标记，因此，我们确实有某种模板。”

- b. 您是否使模板经过审核过程以确保其适合？

JS&JC：“是的，我们把发现的所有设计技术包含在内。”

- c. 您是如何获得这些设计技术的？

JS&JC：“我们进行许多实验和测试。”

- d. 您总是在加载数据时实施一些特殊技巧或技术吗？

JS&JC：“我们遵循建议的沙漏大纲设计，然后使用 SQL 中的 ORDER BY 以恰当顺序获取数据”。

- e. 在处理公式的最佳方法上 - 是使用计算脚本还是使用大纲公式 - 存在一些争议。有人说把一切都放在大纲中，也有人说把一切都放在计算脚本中，您有什么发现？

JS&JC：“我们确实使用一些公式，但会受到限制，因为我们使用替换变量，而且由于无法把它们包含在公式中，所以无法使用。”

- f. 替换变量有什么用途？

JS&JC：“主要用于月份，因为我们要进行许多时间序列基础分析。”

4. 您如何从原型发展到生产模型？一般来说，您需要多长时间开发和应用新模型并将其投入生产？

JS&JC：“我们确实在多维数据集进入生产阶段之前进行多次测试。我们观察高速缓存命中率，并检查是否给高速缓存分配了足够内存。我们趋向于将索引高速缓存命中率调整在大约 100%。

在开发周期的前一个月，客户就可以访问多维数据集，运行查询并与我们一起测试。这样，一旦他们感到满意，我们就会把它投入生产。”

5. 您如何验证多维数据集？

JS&JC：“我们经过一个正式的过程把多维数据集投入生产，该过程也包含数据验证。我们根据现有报表进行反向检查，而现有报表则根据主机上的源系统运行。我们与客户一起确认如何计算字段以及从哪里获取数据，而且共同开发了验证过程，该过程使用一组从主机系统生成的报表。每当建立多维数据集时，都要根据这些报表进行验证。”

5.2.4.3 客户机工具

1. 您使用什么客户机工具？

JS&JC：“有些人使用 Excel 电子表格插件，其他人使用 Cognos Powerplay 或商业对象。因此，我们有许多客户机工具。”

- a. 那么用户如何选择使用哪种工具，还是趋向于使用两种结合？

JS&JC：“我们发现客户倾向于使用已有工具。DB2 OLAP 的优势之一是其开放式体系结构，该体系结构有多种客户机工具，客户可以据其喜好选用。”

2. 谁设计客户运行的报表？

JS&JC：“客户机设计其自己的报表，许多电子表格的使用是临时的。”

3. 您是否发布来自数据的报表？

JS&JC：“不，我们的客户需要 OLAP Server 具有的特定功能。”

5.2.4.4 管理

1. 您如何管理大纲更新？

JS&JC：“只是时而不时地给每个维添加新成员，新成员到来时，维构造自动处理其位置。”

2. 您如何管理加载更新？

JS&JC：“我们通过完全重建多维数据集或使用增量加载和计算来更新多维数据集。两种方法都与多维数据集的大小相关。如果多维数据集很小，可以快速重创；如果很大，就进行增量创建。”

a. 您认为两种方法各有什么优缺点？

JS&JC：“与增量相比，较小型多维数据集一般均可完全重建，这样好像能够减少碎片。”

b. 您多久刷新一次多维数据集？

JS&JC：“我们每月更新一次。”

c. 多维数据集的历史有多久？

JS&JC：“从一个月到两年时间不等。”

3. 您如何管理安全性？

JS&JC：“我们不花费太多时间对安全性进行更改和更新。当客户机需要添加用户时，他们会通过电子邮件通知我们，而我们会设置新的 ID。安全性非常简单，工作人员根据用户群体的请求负责管理安全性。”

a. 您是否将 OLAP 安全系统与其他系统集成？

JS&JC：“没有。”

5.2.4.5 操作

1. 您如何管理备份？

JS&JC：“我们使用 ADSM 备份页面、索引文件以及应用程序目录。我们还使用导出。把级别 0 数据导出到文本文件，并对这些文本文件进行备份。”

a. 多长时间运行一次备份？

JS&JC：“每星期一次，在每个星期日。如果某个数据库上有许多更改，我们就会根据需要更加频繁地进行更新。”

- b. 每周备份是不是创建整个多维数据集的批过程的组成部分？

JS&JC：“不是，它是一个独立的过程。”

- c. 您提到要么备份页面和索引文件，要么执行级别 0 导出，然后备份加载规则之类的应用程序目录。那么系统的其他部分怎么处理，如配置文件和安全文件，它们也包括在备份过程内吗？

JS&JC：“是的。”

- d. 您是否进行增量备份？

JS&JC：“我们通常进行完全备份。”

- e. 您进行完全备份。如果需要恢复什么数据，您就仅恢复相应应用程序？

JS&JC：“没错。”

2. 您是否有灾难恢复计划？

JS&JC：“有。我们进行完全备份的原因之一，就是为了应付发生无法访问服务器等灾难性情况。若出现这样的情况，我们可以使用完全备份恢复到另一个服务器上。”

- a. 您是把磁带存储在两个地方，还是有专用于磁带的非现场存储区？

JS&JC：“非现场存储区。”

- b. 如果发生绝对糟糕的情况，如地震或类似灾难，而且整个数据中心瘫痪，您有相应的灾难恢复计划吗？

JS&JC：“有，DST 针对企业各个方面(包括您所说的情况)的全面应急计划。”

3. 您采用什么策略应用修复包或修补程序？

JS&JC：“我们把修复包加载到前面提到过的一个独立的机器上。如果测试结果令人满意，就更新测试服务器，并通过更多测试来查明可能干扰我们的应用程序故障。一旦确信修复包不会产生任何干扰，就将其加载到我们的生产服务器上。”

- a. 您针对服务频率的策略是什么？您是否留意每个修复包？

JS&JC：“稳定性是决定因素。如果 readme 文件中有针对我们所存在的某个问题的修复包，我们肯定会下载并安装。但是，如果修复包是有关我们不使用的某种功能，那就不去管它。”

b. 您所希望的更新频率是什么？

JS&JC：“我们希望看到经常发布修复包，这样就可以决定哪些修复包适应需要。”

c. 您多长时间把一个修复包放入生产服务器？

JS&JC：“测试服务器运行最新版本，其中有最新的修复包。我们发现必须快速应用此修复包，因为 SQL 接口会出现问题。生产服务器仍然运行以前的版本，我们的修复包往往落后于最新可用级别三个修复包。我们从不应用所有修复包。”

4. 您目前利用什么批窗口？

JS&JC：“每个月，批过程在 24 小时内完成。有了相应文件之后，我们才能启动批过程。但迄今为止，我们还没有在达到最后期限方面遇到问题。”

a. 您的 OLAP 批窗口在一端是由数据仓库数据的可用性确定，那么，另一端由什么确定呢？

JS&JC：“我们与客户签订了一个服务级别协议，其中规定应在月末处理完成三天之后，提供包含多维数据集的仓库。”

5. 您如何监控批过程？

JS&JC：“我们始终有开发人员在监控该过程。”

a. 您是否在用自动化功能检测故障？

JS&JC：“目前是由工作小组负责监控该过程。”

b. 您如何了解所有的事情顺利完成？

JS&JC：“每个计算结束时就会触发触发器。此时，我们就会针对多维数据集运行报表脚本，并将报表文件发送到另一平台，在那里，将这些报表文件与针对主机上的源系统直接生成的报表进行对比。如果值不相等，我们就知道肯定是哪儿出了问题，并研究其中缘由。”

c. 也就是说，您不只是检查已完成的模型，还检查所创建模型的准确性？

JS&JC：“是的，我们的客户需求这样。我们与用户共同确定他们将要运行的重要报表类型，创建相应的验证报表。”

d. 您是否丢失过批窗口？

JS&JC：“没有。”

e. 是否有一些特别原因能够说明您为何通常能够满足需求？

JS&JC：“我们的商业分析师花费大量时间来了解客户的需求，因此，生产应用程序非常稳定。我们进行大量测试和验证，而且开发人员经常试验可以改进工作和缩短处理时间的不同技术。”

5.2.4.6 访谈结束

1. 您认为您正在做的世界一流的事情是什么？

JS&JC：“首先是精确性。我们在验证过程上花费大量时间，而且已能够针对主机源系统精确验证数据。因此，客户对我们的产品及其精确性感到非常满意。其次，是我们在与客户合作过程中培养起来的了解其商业需求以及提供有价值信息的技能。”

a. 是什么使您能够具备上述能力？

JS&JC：“我们在了解企业、数据和技术方面具有丰富的经验。DST 在这方面有许多经验。”

b. 是否还有其他最佳做法？

JS&JC：“在技术方面，我们基于模板的开发过程非常强大，而且可以极其高效地管理服务器资源。”

2. 为了改进 OLAP 安装，您将会进行的最重要的变革是什么？

JS&JC：“从技术观点看，我们希望把最新版本提供的某些功能包含在内，例如，“属性”维。我们还希望使用 OLAP 集成服务器赋予客户一些能够改进本产品的附加功能。我们也在研究通过 Web 提供信息——实际上是想通过 Web 更加顺畅地访问关系数据和多维数据。”

5.2.5 匿名人士访谈

该匿名人士 (AA) 就职于美国一家大型制造企业。

5.2.5.1 环境

1. 请描述您的环境。

AA：“我们使用 Windows NT 服务器、Windows 客户机、Excel/VBA 接口、用于格式化工作簿的 FTP、InTouch 自动化服务器、Essbase 502 修补程序 11。我们选择 Windows NT 服务器和 Windows 客户机以降低成本及获得内部支持。

每个 OLAP 服务器均拥有 2 GB 内存、4 个处理器，以及 60 GB 以上磁盘。我们目前使用 Compaq Proliant 6500 6/200 处理器：购置时，该处理器属最佳 NT 服务器配置。”

2. 开发和测试情况如何？

AA：“我们有一套生产系统、一套测试系统及一套备份系统。各个系统均有 2 GB 内存。

由于生产系统和测试系统是独立的，因而可以测试新的版本/修补程序、数据库更改、大纲更改。备份服务器可为生产服务器提供冗余。”

3. 请问您采用哪些存储设备，如何使用？

AA：“镜像 (EMC) 为生产服务器提供冗余磁盘存储区。这样就可以快速从生产服务器切换到备份服务器。”

4. 您如何优化您服务器资源？

AA：“就内存使用而言，我们确定数据库块大小、高速缓存设置、Essbase 配置线程设置。对于磁盘，我们根据预测的数据库大小进行分配。索引和页面文件的估算值/大小确定分配量。至于处理器，我们在加载和计算过程中测定资源使用情况。数据库设计确定资源使用情况。我们还对设计选项进行审核，以确定其对资源使用情况的影响。”

5. 您如何充分利用多个处理器？

AA：“它们的利用取决于 Essbase 对多处理器的使用情况，Essbase 利用这些处理器来进行计算、检索，等等。”

6. 您目前支持多少个 OLAP 多维数据集？

AA：“我们管理着 7 个生产多维数据集，并且所有多维数据集都包含在一个生产服务器上。页面文件大小从 200 M 到 2 G 不等。”

7. 多维数据集支持哪些应用程序？

AA：“‘实际总账’产生报表和滚动运算预测。每个业务单元仅查询自己的数据。”

8. 您支持多少个用户？

AA：“我们一般支持 70 个用户，但支持的最大用户数量为 150 个。”

9. 您的 OLAP 系统管理工作小组由哪些人员构成？

AA：“有 4 个人负责 OLAP 系统管理，即负责客户安全管理、大纲更改、试验/测试以及生产的迁移。”

5.2.5.2 设计

1. 您的 OLAP 需求来自何处？

AA：“首先确定客户报表需求，然后选择适当工具/方法(如 OLAP)。”

2. 当开发新型 OLAP 模型时，您采用哪种技术？

AA：“我们使用标准客户需求分析。数据分析非常重要。我们进行了广泛的原型/设计选择。”

a. 您如何识别 OLAP 机会？

AA：“客户需求标识 OLAP 报表特征(透视/明细/总成)。我们使用数据分析控制维数目以及粒度。我们确定所需的维/成员，然后进行可行性/优化/成本分析。我们还确定了已分区多维数据集/SQL 穿透钻取是否将提供充分的报表细节。”

b. 进行高级建模时，您使用什么样的工具及技术？

AA：“我们使用数据/相似性分析，也使用案例。我们确定数据需求和视图。我们还考虑现有报表。”

c. 您在纸上进行了多大量的设计工作？利用 OLAP Server 又进行了大量的设计工作？

AA：“我们使用纸张进行初始需求/高级建模，我们使用 OLAP 进行更为详细的原型设计/可行性分析。”

d. 开发一个初始大纲有多快？

AA：“在确定高级数据需求之后，我们很快就能开发出初始大纲。”

e. 您通过什么渠道收集源数据？

AA：“我们从现有文件/电子表格手动收集源数据，我们还从操作源中抽取源数据。”

f. 如何估计多维数据集大小以及资源需求？

AA：“我们需完成如下工作：

- 确定块大小
- 确定/测试高速缓存大小
- 确定 OS 线程数
- 确定索引和页面大小/估算增加情况
- 预测加载/计算使用情况。”

g. 在该点上，最重要的配置步骤是什么？块大小、紧凑/稀疏……

AA：“最重要的步骤有：

- 检查块大小以确定是否是最佳块大小
- 测试数据加载时间并检查数据输入顺序
- 检查整个多维数据集和子多维数据集的计算时间
- 检查动态计算和只标签成员的使用情况
- 检查存储成员、动态成员以及父级成员的检索时间。

h. 在此阶段，您采用了哪些技巧/技术？动态计算、只标签、大纲顺序……

AA：“我们遵循以下原则：

- 仅存储所需要的(动态计算/只标签/ UDA)
- 测试动态计算的检索时间
- 概述动态计算的顺序。”

i. 如何创建全部维？

AA：“我们手动或通过从 SQL 或序列文件中抽取来创建全部维。”

j. 如何获得源数据？

AA：“我们使用工作表或抽取程序。尽早确定数据源和验证数据源内容/精确性非常重要。”

k. 加载数据时，您采用了哪些技巧/技术？—— 排序、使用 SQL 转换？

AA：“我们进行抽样导出以确定最佳数据加载顺序，减小数据文件大小以缩短加载时间，我们还使用大纲顺序。”

l. 您是使用计算脚本，还是使用大纲公式？

AA：“动态大纲公式用于广泛减少批量计算时间和存储需求。公式通过检索时间得到平衡。除了清除/移动数据之外，没有计算脚本。”

m. 您如何测试查询响应时间？

AA：“我们加载代表性数据文件，测试动态计算，而且还用 Excel 客户机和报表脚本测试所有维级别组合/缩放/透视。我们使用一系列 Excel 工作表和报表脚本来比较不同的实施选择。”

3. 您如何从原型发展到生产模型？

AA：“我们在专门用于测试的 Essbase 服务器上进行测试。我们进行初始生产测试，并在下一个会计月结算周期之前应用到客户处。”

4. 您经过哪些调节步骤？采用了哪些调节技巧/技术？

AA：“我们进行了如下工作：

- 检查高速缓存设置，确定哪些设置是最佳的目前尚无固定方法，而且我们使用的是不同设置进行测试。
- 检查检索缓冲器
- 检查块大小
- 测试动态计算的使用情况，并检查对于检索和数据加载/计算的影响。检查大纲对于计算顺序的影响。
- 测试动态计算检索
- 检查默认计算时间
- 检查所有检索
- 执行完整数据加载测试
- 减少不必要的上卷，使用只标签
- 只存储所需数据。使用动态计算
- 使用分区缩短多维数据集计算时间。”

5. 您如何从原型演进到生产模型？

AA：“我们进行了以下工作：

- 加载所有维成员以验证大纲
- 创建完整的生产数据加载文件以验证数据源和内容
- 检查所有安全过滤器
- 对原型进行广泛的客户测试
- 创建新的应用程序/数据库，并把大纲和数据文件复制到生产服务器。我们还创建了安全组/过滤器。”

5.2.5.3 客户机工具

1. 您使用什么客户机工具？

AA：“我们使用 Excel 插件、Visual Basic 实用程序，用于安全性考虑和 LRO 管理。我们之所以选择这些工具，是因为它们是标准 Essbase 程序包的组成部分，并提供 Visual Basic API。我们所使用客户机工具的最重要特性是易于使用，而且不会增加成本。客户使用的是 Excel 插件。IT 专业人员使用的是 Visual Basic。”

2. 谁设计并创建报表？

AA：“公司客户创建格式化报表。全球客户可创建特定报表。”

3. 您如何发布报表？

AA：“报表存储在 Windows NT 服务器上，并可用于应用程序中的 FTP 访问。”

5.2.5.4 管理

1. 您如何管理大纲更新？

AA：“大纲每月至少变化一次。我们进行手动和自动维创建。使用 SQL 进行自动创建，并保留抽取创建文件。我们测试更改对于数据库性能的影响，并验证报表的准确性。”

2. 请问如何管理加载更新？

AA：“如果多维数据集中有碎片，便可重新创建多维数据集。重新构造可能会丢失 LRO；然后，在大纲更改后，如果有要保留的 LRO，我们就重新构造所有数据。如果没有，我们就清除上层块，并重新构造级别 0 数据。每个功能关闭其书本之后，加载增量数据。”

3. 您是否管理滚动时间周期？如何使用？您面临的问题有哪些？

AA：“存在可确定属于哪个预算提交周期的维。”

4. 您是否使用应用程序分区，如何使用？

AA：“以前，我们曾使用过分区。Essbase 中的错误有时会导致复制时间过长。我们用报表脚本予以取代，以便于快速复制。必须使所有分区定义保持同步。”

5. 您如何管理安全性？

AA：“在识别新型业务单元或新的客户时，可添加 Essbase 组、用户 ID 以及过滤器。过滤器每月更改一次，以便把多维数据集更新到当前月/提交数据。DB2 表提供备份并记录安全更改。”

5.2.5.5 操作

1. 请问如何管理 OLAP 的备份？

AA：“EMC 用于为所有生产 Essbase 目录(系统和应用程序)提供镜像。在备份服务器上拆分 EMC 镜像，以及验证/导出多维数据集，并进行复制以用于备份。镜像是在备份之后重新建立的。我们使用 EMC 进行镜像，并使用 InTouch 进行作业调度。备份每天都在运行。我们通过恢复备份目录和重新为 ssaudit 日志文件加载源数据进行恢复。”

2. 请问是否有灾难恢复计划？

AA：“备份存储于安全站点上，并将作为整个 LAN 恢复的组成部分进行恢复。”

3. 您采用什么策略应用修复包或修补程序？

AA：“LAN 支持组每年应用一两次 OLAP 修复包。升级也是按照同样的频率计划的。所有版本/修补程序均在测试服务器上进行测试。所有 Essbase 软件(实用程序、应用程序管理器、API)都要经过测试。升级过程如下：

- a. 把新的修补程序应用到测试服务器
- b. 更新测试脚本
- c. 进行必要的转换
- d. 测试所有 Essbase 软件
- e. 升级到与客户计划相符的生产阶段
- f. 把升级版本安装在生产服务器上
- g. 运行安装验证脚本。”

4. 您目前利用的批窗口有那些？

AA：“有一个必须加载和计算所有功能数据的 6 天窗口。加载完最后的功能数据之后，有一个 2 小时窗口，必须完成合并进行分析/报表。生产 Essbase 必须 7/24 可用，并使备份停机时的停机机率达到最小化。”

a. 哪些可定义批窗口？

AA：“公司会计报表需求。”

b. 在一般的批过程中，您运行的过程有哪些？它们需要多长时间？

AA：“我们在一般批过程中启动的过程有：

1. 能数据清除、加载和计算。需要 10 分钟。
2. 计算所有用于总帐数据多维数据集的所有数据。需要 20 分钟

5. 您如何管理批过程中的错误？

AA：“我们使用 InTouch 自动化软件电子邮件发出通知(还具有分页功能)。更正、重新加载和计算错误文件。”

6. 如何自动操作批过程？

AA：“我们仍然使用 InTouch 自动化软件。主机作业创建 InTouch 作业和数据加载文件。如果有错误，InTouch 便会发出通知。”

7. 您的批过程由哪些部分构成？

AA：“除了备份过程中的几分钟之外，应用程序始终可用。我们丢失的批窗口不到 1%。我们实施附加的数据加载监控/记录功能，以便在使用 InTouch 或 Essbase 时发出通知。”

5.2.6 Rich Semetulskis 和 Alan Farkas 访谈

Rich Semetulskis 和 Alan Farkas (RS&AF)两人都就职于 ThinkFast 咨询公司(商业合作伙伴)。 Rich Semetulskis 是一名高级项目经理，并在使用多维数据集和关系技术实施报表以及分析解决方案方面拥有超过 15 年的丰富工作经验。ThinkFast 咨询公司是一家企业范围商业智能服务的全面服务提供商，这些服务可用来为客户提供对于战略和战术决策信息的访问。

5.2.6.1 环境

1. 请您描述一下 OLAP 环境。

RS&AF：“我们使用 OLAP 系统，其运行于 SUN SOLARIS(包含 4 个处理器、1 GB 内存和 30 GB 磁盘)和 NT(包含两个 500 MHz 处理器、2 GB 内存和 30 GB 磁盘)。我们将建议在生产中使用 UNIX 平台。”

2. 开发和测试情况如何？

RS&AF：“我们建议使用专用平台进行测试和生产，并使用相同操作系统，以便于把过程从开发服务器迁移到生产服务器。”

3. 从何处可获得您的源数据？

RS&AF：“我们的源数据事实上来自 SQL。我们使用 ORACLE V7 和 DB2 V5.2 数据源，并通常将 SQL 接口与加载规则文件一起使用。对于新的 OLAP 数据库开发，当在输入多个不同来源时，我们会设法把数据存储于关系存储库中，并创建星型模型，从而使用 SQL 脚本清除和传输数据。

如果我们比较一下平面文件和 SQL 接口之间的加载时间，我们会发现从平面文件加载要快一些，但是，我们宁愿在关系数据库中寻找源数据，以消除创建平面文件以及进行文件传输所需的时间。”

4. 请问目前可支持多少个多维数据集？

RS&AF：“我们的生产服务器上有三个 3 应用程序，总共有 15 个多维数据集，大小介于 10 到 12 GB 之间。”

5. 多维数据集支持哪些应用程序？

RS&AF：“我们有多种应用程序，从市场营销和零部件存货分析，到财务、预算、制造应用程序。”

“我们认为管理属性的能力将会开放，并将促进销售及客户分析。”

6. 请问您现在支持多少个用户？

RS&AF：“我们定义了 1000 个潜在用户和 300 个并行用户。同时连接的用户平均数为 25。”

5.2.6.2 设计

1. 请谈谈您验证多维数据集的方法？

RS&AF：“我们使用日志检查是否已加载所有数据，并将结果与详细级星型模型输入进行比较。我们还自动生成了一些总计及详细级别的验证报表。”

2. 在开发应用程序时，您采用了哪些调节步骤？

RS&AF：“我们小心对待紧凑和稀疏维，获得不到 64K 的块大小，并对维进行排序。”

3. 您如何在新模型中使用完全维进行操作？

RS&AF：“当使用多个数据源时，我们总是设法定义一个关系中转区，并从此处设计适合于多维数据集的星型模型。”

5.2.6.3 操作

1. 请问如何管理备份？

RS&AF：“我们在备份时关闭 DB2 OLAP。我们习惯于创建每日备份，每周有五个晚上进行。我们小心备份安全文件，以确保数据没有遭到破坏。”

2. 您目前使用什么样的批窗口？

RS&AF：“我们每天晚上有大约 6 个小时的批窗口工作量。”

5.2.6.4 访谈结束

1. 您认为哪些需要做的事情是世界一流的事情？

RS&AF：“OLAP 成功的关键是在于深暗有哪些调和需求，以便能确保向最终用户提供数据的完整性，并能制定强有力的项目管理方法。”

5.2.7 Aster Hupkes 访谈

Aster Hupkes (AH) 现就职于设在荷兰的 IBM 全球服务部，担任商业智能组的 IT 专家，在荷兰多家客户设计和实施 OLAP 解决方案方面拥有超过两年的经验，而且主要使用 DB2 OLAP Server / Hyperion Essbase。

5.2.7.1 环境

1. 请您描述一下 OLAP 环境。

AH：“我们大部分的实施工作均在 AIX 平台上进行的，因为该平台可为大型专业数据仓库环境提供极大的灵活性和可扩展性。我们也在 AS/400 上进行实施工作，迄今为止，除原型设计之外我们尚未在 Windows NT 上进行任何实施工作。通常情况下，不管是 DB2 还是 Oracle 数据仓库均运行于同一 AIX 服务器上。但是，我们已经完成了数据仓库位于 OS/390，而 OLAP 多维数据集位于 AIX 上的项目。

我们经常使用 SQL 接口填充多维数据集。我们已将 DB2 或 Oracle 用作 RDBMS。在不同的客户那里，我们看到了不同的环境：

- 数据库和 DB2 OLAP 都在同一台 UNIX 机器或 AS/400 上
- 数据仓库位于 OS/390 MVS 系统，而 DB2 OLAP 在 RS/6000 上。”

2. 能否谈谈有关开发及测试的情况？

AH：“在不同的客户那里，我们看到了不同的分隔开发、测试及生产情形。当然，最理想的解决方案是安装独立的服务器。但是，由于许可证成本较高，这一方案并不总是可行的。下面列出了几种不同的情形：

- 四种不同的服务器和 DB2 OLAP 安装，分别用于开发、测试、验收和生产。
- 采用逻辑分隔的服务器，分别以应用程序名称命名：首字母 D-、T-、A- 和 P- 分别代表开发、测试、验收和生产应用程序。
- 上面两种情形的混合：两个服务器：一个用于生产，另一个用于开发、测试，以及验收(如果适用的话)。通过应用程序名称的首字母对开发/测试服务器进行分隔。例如，这会在生产服务器上产生一个应用程序“实例”，在开发/测试服务器上产生应用程序“D 实例”、“T 实例”和“A 实例”。

在理想的情形下，测试服务器应与生产服务器具有相同的配置，但在实践中，它们通常都比较小，均具有处理器的功率，磁盘的空间。”

3. 请问您从何处获得源数据？

AH：“由于始终将 DB2 OLAP 作为大型三层商业智能参考体系结构的组成部分，因而我们用于 DB2 OLAP 多维数据集的输入源要么是中央数据仓库，要么是关系数据集市。因而，我们的维及数据加载输入总是 DB2 或 Oracle 关系数据库，并通过 SQL 接口加载。只有在我们把多维数据集的原型设计作为设计阶段组成部分时，才使用平面文件。我们直接从源系统创建 DB2 OLAP 多维数据集。”

数据仓库的输入来自公司内的不同操作系统，但是，客户购买的外部数据也可成为数据仓库的输入。”

4. 请问采用了哪些存储设备？

AH：“我们使用 SSA 存储设备。整个数据仓库(包括 OLAP 多维数据集)的存储空间介于 50 GB 到 10 TB 之间。OLAP 数据和索引文件分布在多个磁盘上。通常情况下可激活镜像。”

5. 您如何充分利用服务器上的多个处理器？

AH：“我们始终针对每个应用程序定义数据库，因为这使我们可以同时计算多个数据库。”

6. 请问目前可支持多少个 OLAP 多维数据集？

AH：“这取决于具体的客户，但每个客户通常介于 5 个和 10 个多维数据集之间，其大小从几 MB 到 40 GB 不等。”

7. 多维数据集都支持哪些应用程序？

AH：“我们已创建的多维数据集支持像‘销售’、‘财务’、‘库存’、‘采购’和‘规划’等多种应用程序。大多数多维数据集用于分析和报表，而且用户对这些多维数据集仅拥有读取访问的权限。只有少数多维数据集用于规划，并允许用户把数据写回多维数据集。”

8. 您可支持多少用户？

AH：“用户数量随客户的不同而异，目前介于 5 到 30 个之间，他们以交互方式分析多维数据集。更多用户使用在 OLAP 多维数据集上创建的报表。”

9. 工作小组由哪些人员构成？

AH：“ DB2 OLAP 开发人员是大型数据仓库工作小组的一部分，包括人数介于 3 到 20 之间。其他角色包括 DBA、ETL 开发人员、项目经理等。 ”

5.2.7.2 设计

1. 您的 OLAP 需求来自何处？

AH：“ 需求绝大部分来自商业分析师，有时来自管理层。我们遇到的商业分析师通常希望把一切都放在多维数据集中，其中也包括所有的可用详细信息。我们的工作是将这些需求转换为小型的高质量多维数据集模型。 ”

2. 在开发新型 OLAP 模型时，您采用了哪些技术？

AH：“ 我们采用多维建模技术收集用户需求，因为其可容易地向用户显示 OLAP 模型。我们发现该技术极为有用，因为其很容易为商业分析师所理解，而且还可极轻松地由 IT 人员转换为技术设计。实际上，我们通过一个或一系列设计研讨会拉近了初始多维数据集模型与用户间的距离。

在设计研讨会期间，您有时可确定制作多个多维数据集，因为有些度量在某些维之间是不相关的。我们也举办许多有关多维数据集中详细级别的讨论会。例如，如果有人说他们需要日详细级别的数据，您可以提出以下质疑：他们是否确实需要知道一年以前某一天的销售额，或他们是否仅仅想对每月数字进行日常更新以便跟踪当月 ‘ Month-To-Date ’ 数字。

首次设计研讨会之后，我们设计了一个原型，并把一些假数据加载到其中。接下来的一次设计研讨会使用此原型验证需求。我们发现原型设计和让用户浏览原型是一种验证和改进设计的非常有效的方法。

我们已创建了若干具有 6 到 12 个维的多维数据集。利用属性维功能，我们可以改造一些旧的多维数据集，并将实维转换为属性维。遗憾的是，我们发现在大型多维数据集(1.5 GB，100,000 个基本成员)中使用属性维会严重影响多维数据集顶层的性能。但是，在详细级，性能却很好。尽管还有待于进行改进，我们仍然非常喜欢属性维提供的额外机会。

我们已开发出了大小电子表格，以根据管理员指南中的计算规则估算多维数据集的大小。然而，这只是粗略数字，因为大小的最大决定因素 - 密度和稀疏性 此时还是未知数。”

3. 您如何从原型发展到生产模型？

AH：“原型是我们创建过程的基础，但是，我们基本上会从头开始。原型用于验证需求，并且很大程度上是在大纲中进行手动定义的。生产模型中的维将从数据仓库内的维表中生成。

如下是一些创建原则：

- 如有可能，我们应避免在一个应用程序中包含多个数据库。因为每个应用程序只有一个 ESSSVR 过程，某个挂起的数据库就会对整个应用程序导致问题。
- 您可以在规则文件中的许多地方执行像添加前缀译码/替换变量这样的操作：在 SQL 窗口中、在使用前缀选项的字段属性中，以及通过创建一个新的字段(字段 - 使用文本创建)并把该字段与另一个字段连接在一起。为了使这些规则文件易于理解，应始终在一个地方执行此操作，最好在 SQL 窗口中，因为这是个最灵活的地方，例如：

```
Select 'P_' || PRO_COD, DECODE(PRO_GRP,null,'Other',PRO_GRP)
```
- 如果计算时间太长，请考虑使用增量更新。不可能总是可以进行增量加载，例如，在历史更改的情况下。同样，当计算时间非常短(少于一小时)和/或每个月只在周末运行一次时，花费大量时间开发增量加载方案就很不划算。
- 使用增量更新时，把时间维设置为稀疏。
- 如果有多个公式，就把度量维设置为紧凑。把度量维中的公式设置为动态计算。
- 设法在一个地方定义所有公式(如比率)，要么在大纲中，要么在必要时在计算脚本中。只有在计算需要特定计算顺序时，才使用计算脚本，这在大纲中不可能实现。例如，根据部门的收入百分比，在各部门之间摊派总成本。”

4. 请问如何验证多维数据集？

AH：“我们发现，验证一种设计的最佳方法是使用原型设计。在第一次建模研讨会之后，我们通常创建一个非常简单的原型。此原型设计得非常简单，因为所有维和度量都是在大纲中手动定义的，数据也是如此。但是，关键在于，该原型中包含模型的所有元素：所有维以及每个维中的每个聚集级别上至少有一个成员(包括备用层次结构和属性维)。这通常会花费大约半天时间，而且，在下一个研讨会开始

之前，我们让用户浏览此模型，以便验证多维数据集设计。

创建多维数据集之后，开发人员进行一次技术测试。这通常是通过将多维数据集结果与 SQL 查询或现有报表的输出进行比较而完成的。开发人员还负责性能测试和调节。在此之后，测试多维数据集就是最终用户的责任了。”

5. 在开发应用程序时，您都经历了哪些调节步骤？

AH：“我们所做的调节量很大程度上取决于多维数据集的大小，以及是否存在性能问题。如果一个数据库只有几 MB，在几分钟内就可以计算完毕，那么花费许多时间进行调节就没有意义。调节总是权衡较短计算时间和性能之间，以及较小多维数据集和用户功能之间的利弊。因而，这不仅仅是一个技术问题，但还需要用户参与。

我们所进行的调试步骤包括：

- 在加载数据的代表性部分，并根据块大小、密度、稀疏性以及加载和计算时间选择最佳配置时，检验紧凑和稀疏设置。在版本 5 中，我们总是倾向于使块大小尽可能小些(块大小应在 8 K 到 64 K 之间，我们通常选择接近 8K)。
- 优化大纲顺序(首先是紧凑维，从大到小，然后是稀疏维，从小到大)，并根据此大纲的顺序对数据加载进行排序。
- 如有可能，请使用“动态计算”或“只标签”。我们通常在紧凑维上使用动态计算，但有时也在稀疏维上使用动态计算，不会对检索性能造成太大影响。
- 计算索引、数据和计算器高速缓存所需的高速缓存大小。
- 开发增量加载方案。我们发现，当时间维为稀疏时，使用增量加载方案效果最好。
- 有时，我们使用计算脚本，以便能够影响计算顺序，并在可能的地方使用 FIX 语句取代 IF 语句。

我们已把这些知识体现在工作小组内的一个 DB2 OLAP 原则文档中了。”

5.2.7.3 客户机工具

1. 您使用哪些客户机工具？

AH：“在大多数客户那里，我们把分析工具和报表工具区分开来。利用分析工具，用户能够以交互方式快速浏览多维数据集。报表工具可用在 OLAP 多维数据集上创建标准、更为复杂的报表，并将其分配给用户。作此区分的原因是，在报表工具中比在分析工具(如 Executive Viewer 或电子表格插件)中，计划、分配和打印的可能性以及所有布局选项将更加复杂。

我们优先选择的分析工具是来自 Temtec 公司的 Executive Viewer，因为这是我们所知道的最强大、最易于使用，以及最直观的 OLAP Viewer。有些客户确实使用电子表格插件，但我们不太喜欢该插件的用户友好特性。”

2. 向客户提供的报表由谁来设计？

AH：“在多数项目中，提供报表是最终用户的责任。我们向最终用户提供一个或多个 OLAP 多维数据集，并且用户负责分析和报表多维数据集。我们发现，多数用户可轻松地使用多维数据集并创建简单报表。如果他们提出更为复杂的报表需求，如基于根据脚本生成报表、计划等，我们可以帮助用户或为他们创建报表。”

5.2.7.4 管理

1. 请问您如何管理大纲更新？

AH：“维在关系表中进行维护，并且使用维创建规则文件加载到大纲中。有时在大纲中手动定义和维护的维只是度量维(包括所有公式)和方案维。

每次数据加载之前，大纲都会发生变化，因为加载数据之前会执行维创建规则文件。维是从数据仓库中的维表加载的。如果有些层次结构在任一源系统中均未得到维护，用户就负责在数据仓库中的关系维表中维护层次结构。”

2. 您如何管理数据加载更新？

AH：“对于小型多维数据集，我们总是执行完全重新加载，而对于大型多维数据集，是否有可能进行增量加载就值得调查了。但是，并不总是可行的，例如，在允许更改历史值的情况下，如果把时间维设置为稀疏，增量加载的速度就会非常快。

有些多维数据集有一个滚动时间范围，该滚动时间范围是通过在数据加载和时间维规则中，添加一个 Where 子句实施的，这些规则文件指定根据系统日期加载成员和数据。如果旧的时间成员随维构造一起被删除，旧的时间段就会缩短。”

3. 您如何管理数据库碎片？

AH：“我们设法在加载数据时防止出现碎片，但是从不进行导出和重新加载。”

4. 您是否使用应用程序分区？

AH：“我们不常使用分区，但我们确实看到一些积极的方面。在测试和生产应用程序之间，可轻松使用复制分区。我们曾经计划在小型、快速，以及高级的多维数据集与大型、慢速，以及详细的多维数据集之间使用链接分区。原因是在这些详细多维数据集中，我们使用属性维，它们在最高级别的多维数据集上的响应时间极低。遗憾的是，我们发现根据属性维进行分区是不可能的。”

5. 您如何管理安全性？

AH：“一般说来，即使那个组中只有一个人，那我们也总是在组级定义安全性。从不在用户级定义安全性。命名约定也很重要，我们尽量遵守客户的标准，也就是说，使用 LAN 或 MAIL 用户名。定义安全性是单个组的责任。

我们有时使用过滤器定义，为某些用户隐藏部分多维数据集或允许用户在数据库的特定部分写入。这是一个非常强大的特性，但我们却发现把过滤器定义从一个服务器复制到另一个服务器上并不容易。有些前端工具可以使用 DB2 OLAP 安全机制，而其他前端工具则需要使用单独的安全机制。”

5.2.7.5 操作

1. 您使用哪些步骤将新型应用程序转化为生产？

AH：“如果有不同的服务器，我们就通过 FTP 把大纲、规则文件和计算脚本传送到生产服务器。如果只有一个服务器，我们就复制数据库。同时复制批作业，并完成批作业计划。”

2. 您如何管理备份？

AH：“管理备份的方法有两种：

- 使用 BEGINARCHIVE 和 ENDARCHIVE 命令
- 关闭 Essbase 服务器，完全备份文件，以及重新启动服务器。

备份通常在夜间进行，因此，可行的方法是关闭服务器、进行备份，然后再重新启动服务器。这是最不复杂的方案。在备份策略中，可以区分两件事情：

- OLAP 定义，如大纲、规则文件和计算脚本。应定期备份这些多维数据集，而且通常是每日备份不可或缺的组成部分。
- OLAP 数据，索引文件和页面文件。在众多案例中，通常无需备份 OLAP 数据，因为在其加载多维数据集的地方关系会备份数据仓库。如果发生灾难，可以在一天之内从数据仓库加载所有多维数据集，而且这种方案通常可以满足需要。但是，多数管理员发现，仅采用备份方式备份一切似乎更为简单。

通常情况下，每天都要进行备份。有人曾经主张按应用程序确定不同的备份需求，但系统管理员通常每天备份所有应用程序，因为这种策略最容易实施。”

3. 请问您有灾难恢复计划吗？

AH：“在有些客户那里，灾难恢复程序非常严格，而且还要经过测试。但是，多数客户只进行备份，而不测试灾难恢复程序。”

4. 就应用修复包或修补程序而言，您采用的策略是什么？

AH：“实际上，我们是根据需要应用修复包，即修复包可以修复现有问题或包含我们绝对需要的新功能。如果没有直接理由，我们宁愿让环境保持稳定，而不去应用修复包。

如果有单独的测试服务器和生产服务器，我们就可先在测试服务器上安装修补程序，并经过几个星期证明有效后，再安装到生产服务器上。”

5. 您为新版本的应用采用了哪些策略？

AH: “如果在新版本中有我们确实需要的新功能, 我们就会安装新版本。版本 7 中的属性维就是一个非常新的功能, 所以我们很快在多数客户中应用了该新功能。我们在迁移现有多维数据集时确实遇到过一些问题; 有些多维数据集的计算时间在新版本中变长了, 而且 SQL 接口工作不正常。而最后一个问题就是通过第一个修复包得以解决的。”

6. 您怎样管理客户机工具的升级？

AH: “这要取决于客户。有时会根据需要只将软件安装在每个用户的 PC 上。只有在最大用户数量介于 10 到 20 名之间时, 这种方法才可行, 而且会使客户机工具更新非常费力。在其他情况下, 前端工具是转出到整个公司标准客户机平台定义的一部分。

我们越来越转向基于 Web 的前端工具, 其中一个理由是容易管理。发布 Web 前端的新版本时, 只需要更新服务器软件。当用户连接时, 服务器自动检测用户是否需要新的插件。”

7. 您如何实现 OLAP 操作自动化？

AH: “我们使用 ESSCMD 批脚本自动执行所有 OLAP 操作。这些 OLAP 过程的计划安排通常依赖于其他过程是否成功完成: 一般为从数据仓库中的源系统进行的加载和/或加载关系数据集市。我们发现, 使用一种计划机制来计划所有数据仓库操作最容易。其可以是一种现有的计划工具, 可以是 UNIX 脚本, 可以是 Powermart 中的 PostSession 命令, 也可以是 OS/390 上的 OPC。”

8. 您如何监控批过程？

AH: “管理员每天早上都要检查数据仓库过程是否成功完成。”

9. 您利用什么样的批窗口？

AH: “通常从 12:00 AM 到 7:00 AM 使用批窗口, 并且是在周末, 但是, 在这个时间里, 还必须从源系统中填充中央数据仓库。”

附录 A OLAP数据集市设计方法

本附录由 Hyperion 的 Paul Turner 编写，介绍许多种设计方法。利用这些方法，可以快速解释多种要求和数据源，从而对创建功能数据集市进行快速跟踪。本附录假设读者基本熟悉关系数据库和多维数据库。同时，了解数据仓库的概念也会有所帮助。

A.1 什么是数据集市？

数据集市通常是针对应用程序的数据库，用来适应具体问题，而不是作为数据存储库。数据集市针对的是可视化工具的特定业务规则的数据提供者，因此数据集市设计是否有效，很大程度上取决于用户需要如何呈现数据。这不同于多数关系架构，因为它们一般用作源系统，并且是为冗余性和性能目的而设计。

即使星型/雪片/星座架构也不会自动成为数据集市。这些结构更多地用于创建数据仓库，而数据仓库是非特定应用程序的(即不是专门为适应定义问题空间而设计的)，而是为大量企业数据提供规格化的存储库。

这是个重要区别。数据仓库包含具有极高粒度的关系数据，这些关系数据是从许多系统抽取而来的，并集中化作为企业数据的标准表现形式。而数据集市只针对某个用户群体，他们需要回答或研究适用于其商业功能的一组具体问题。

经常从数据仓库创建数据集市。数据集市提供一个单独的、非特定于应用程序的世界粒状视图，并通过一系列数据集市把数据呈现给用户，这些数据集市添加功能特定的规则、合并、时间颗粒和附加功能(如预算编制)，它们都对要支持的仓库没有意义。

为了进一步阐明这个定义，可以把数据集市视为目标数据库，该数据库允许：

- *快速对数据及其聚集级别进行专门分析。*

数据集市中的数据一般不如数据仓库那样有粒度。在这个意义上，可把数据集市视为复杂的聚集表。例如，数据集市可按城市、或按商店以最高粒度处理销售数据。这与数据仓库形成对比，因为数据仓库不仅与商店有关，而且与每个销售人员和/或每个销售点终端的销售数据有关。

数据集中数据的粗粒度是查询保持一致和快速的原因之一。例如，如果 2 月 1 日发生 10,000 项事务，而 2 月 2 日只发生 5 项事务，2 月 1 日从关系事务处理系统创建销售汇总表要花费较长时间，因为查询时间随数据量增加而增加。这与时间颗粒为按日的数据集市形成对比，因为数据集市每日仅包含一个数字，而且不管各项事务的数量如何，查询时间都保持一致。

增加特定功能分析意味着可以从数据仓库创建多个数据集市。一个集市可能针对“销售”，而另一个数据集市可能针对“营销”，再一个数据集市则可能针对“财务”。每个数据集市实际上都利用同一数据仓库中同样的数据，但从不同方面查看数据。这使用户可以：

- 查看具有不同维的数据。例如，“销售”数据集市可能包含按销售人员细分的数据，而“营销”人员则可能有兴趣查看按促销活动细分的数据。
- 查看从数据派生的不同度量标准。可以根据源数据进行复杂分析，如果源数据是从数据仓库派生的，分析起来就非常耗时。例如，数据集中总共计算的百分之一在数据仓库中都是极其具有挑战性的，因为在数据仓库中，计算时需要聚集整个维(这可能相当于数百万个记录，因为数据仓库具有极高的粒度)。
- 写回到数据，并执行假设分析。例如，“财务”部可能希望按数据集测试各种预算方案。此分析类型通常不需要数据仓库中的细粒度数据。事实上，数量如此庞大的详细信息会阻碍假设分析计算。其他要求(如存储重复的一致预算结果和计算实际变化)也不足以对细粒度数据仓库进行假设分析计算。

- *将历史数据存储和数据仓库中数据的范围之外*

有时，数据仓库具有不适合于存储大量历史数据的容量。例如，具有细粒度和高活动量的 Web 日志就禁止进行漫长的历史存储。在这样的情况下，数据集市可能更适合于以一种粒度不太细的、易于管理的摘要形式卸载以前年度的数据。

与通过为事务处理而设计的关系数据库相比，通过优化的多维数据库，可更容易地提供所有这些数据集市要求：聚集数据瞬时特别分析、特定功能分析，以及历史数据访问。

A.2 设计数据集市

数据集市设计取决于许多因素：

- 报表要求(用户希望如何对信息进行切片和切块)
- 计算/派生数据要求
- 多维数据集大小
- 维是否有意义
- 多维数据集粒度(每个维的最粗颗粒)
- 数据的维粘合性(所有引入维之间有多少测度相关)

所有这些因素都是相互关联的。例如，尽管某个给定计算可能必需某个维，但该维有可能会使多维数据集的大小剧增，并会在过程中引入无意义的维。因此，数据集市设计的困难在于平衡这些要求。

A.2.1 确定粒度

数据集市的粒度与产生的多维数据集的大小紧密相关。一般情况下，多数系统的事实表(除时间之外)中都有一系列可识别的最细颗粒。例如，呼叫中心系统的案例号、销售系统的事务号以及 Web 日志的 Cookie 或用户标识符。

数据集市设计员必须考虑两种类型的粒度：多维数据集内的粒度和多维数据集外的粒度。

A.2.1.1 多维数据集内的粒度

多维数据集内的粒度确定维明细数据的详细级别。颗粒越细，应用到叶级维成员的汇总就越少。

表 15 显示许多示例。

表 15：粒度示例

	细粒度	中级粒度	粗粒度
时间	秒，作为特殊时间间隔	小时	天或月
产品	产品 SKU	产品名称	产品类别
地理	国会地址：35 Laurel Drive	邮政编码	城市

用户经常易于使其数据集市粒度太细。需要牢记的是，多维数据集中的数据就是用户将来浏览的数据，而且通常使用非常特殊的用户界面。在许多情况下，数据粒度越简单，创建的分析就越笨重，而且不会增加任何实际价值。

例如，看看 Web 流量分析应用程序内时间的粒度。鉴于源数据的粒度非常精细，很可能把数据集市内的时间频率设置为每五分钟一次。但是请看在数据集市整个生命周期内此数据的值。用户是否会真正关心 1998 年 2 月 5 日 11:05:15 发生了什么事？一般来说，回答是否定的。

多数情况下，某个维的粒度取决于报表或计算要求。请看下面的基本时间报表要求：日、星期、月、季度和年。问题是星期不能自然演进到月，例如，“第五星期”可能跨越 1 月和 2 月。解决此问题的自然方法就是加载日粒度，并让其演进到月(然后，又演进到季度和年)，并且定义备用层次结构，其中日颗粒演进到星期。

另一种方法是使用日期加载过程以不同粒度编写数据汇总。这可以用两种方法来完成：

- 如果用两个维表示时间，那么数据集市可以有一个“年”维(包含年、季度和月)以及一个一年各个星期维(包含从第 1 到第 52 星期的统一列表。加载数据时，只按星期创建一个二维月引用。这会产生许多有趣的查询可能性，例如：“显示每年前 5 个星期如何比较”。在只有存在某个查询(即“Suppress Missing”)的数据情况下才显示成员时，这种方法最适用，例如在查看 1 月过程中深入到星期时，就仅显示第 1 到第 4 星期。

- 如果必须让月和星期处于同一维，就按月编排时间键的格式(利用 SQL 时做起来很容易)，并把数据加载到月成员。然后使用第二次数据加载按星期编排时间键格式，并将数据加载到此时隙。缺点是，这会使从数据仓库加载数据集市所需的时间增加一倍，因而可能不可行。

A.2.1.2 多维数据集之外的粒度

多维数据集之外的粒度由忽略的数据确定。例如，“财务”人员可能有兴趣查看表 16 中的销售数据，但按销售人员单独显示。同时，“销售”人员明显地希望查看每个销售人员的数字。

表 16：多维数据集外的粒度示例

	面粉公司	燕麦公司	小麦国际公司
Paul Turner	423	542	245
Carole Turner	123		145
Jim Turner			124
按制造商的总销售额	546	542	514

“财务”多维数据集：时间 X 账户 X 制造商

“销售”多维数据集：时间 X 账户 X 制造商 X 销售人员

包含的维越多，数据集市的粒度就越高。更为重要的是，此新颗粒(本案例中为“销售人员”)允许进行一系列新的复杂计算，但如果没有这个新颗粒，就无法派生这些计算。看看计算“一个销售人员占用的平均数”。由于 Paul 占了三个，Carole 占了两个，而 Jim 占了一个，因而答案就是 $((3+2+1)/3 = 2)$ 。查看“财务”多维数据集时，看不到此信息，并只显示汇总的“按制造商的总销售额”。

但是，如果“财务”人员想在其数据集中查看度量标准“一个销售人员涉及的制造商平均数”，那该怎么办呢？如果暂时忘记诸如分区和 @XREF 这样的特性，“财务”人员如何在不包括“销售人员”维的情况下计算此度量？

注意：如果包含有“销售人员”维，利用 DB2 OLAP/Essbase 完成此计算的方法就是引入存储公式 -SALESCOUNT。如果“销售人员”的销售额为“<#Missing”，那么 SALESCOUNT 就设置为“1”；否则，销售额就停留在“#Missing”。这会给出必需的 3,2,1。要计算平均值，请用销售人员数目除总计。（这可以预先加载，也可使用 DB2 OLAP 7.1 或 Hyperion Essbase 6.0 或更新版本中的计数函数进行计算。）

这为数据集市设计员提出重要挑战。尽管报表不需要某个维，而且包含该维可能会极大地增加数据集市的大小，但某些计算可能必需该维。考虑另一个与 Web 流量相关的示例。一个公用度量为特殊用户，如“有多少特殊用户买了我的 SuperWidget”或“有多少特殊用户看过我的主页？”

在该情况下，使用唯一 Cookie 标识符（例如，“web302ca_941227897_157097”）确定用户。一个电子商务站点可能有数百万特殊用户，每个用户有不同的 Cookie。深入 Cookie 维会产生数百万个像上面一样的系统生成键，它们都是对报表无用的数据。但只有包含 Cookie 维的情况下，才可能进行有用的计算，如计算查看某个 Web 页用户的特殊 Cookie 数目。

解决这个两难问题的一个解决方案（尽管并不是特别好的方案）是在加载时预处理数据，因为关系数据库了解 Cookie 的一切情况。但是从数据加载的观点看，这会非常昂贵。例如，考虑一下计算各种产品和类别的特殊用户。如果 SuperWidgetA 的特殊用户计数为 1，而且 SuperWidgetB 的特殊用户计数也是 1，父级 Widget 的特殊用户计数就可能是 1 或 2，具体计数取决于 SuperWidgetA 的计数和 SuperWidgetB 的计数是否代表同一个用户。换句话说，不会在任何维之间添加特殊维计数；必须预先用 SQL 计算每个级别，因而使加载工作极为昂贵。

另一种方法是采用一种混合近似法，而且只通过关系查询支持特殊用户分析，并提供数据集市和数据仓库之间的图形链接。

像分析器这样的工具允许通过其关系连接进行这种链接。

A.3 确定维

面对星型或星座结构，数据集市设计员如何才能快速确定创建有意义的、易管理的 OLAP 数据集市维？所有维表都可能是该数据集市中的维，而且每个维表内的列可能是维、层次结构、属性或数据。第一步通常是确定用户要求。但这些要求必须始终与数据仓库的要求和结构相关。

此外，还有许多快速定义数据集市基本维的其他原则。请看下表 17 中的信息：

表 17：产品表

产品 ID	产品代码	产品名称	产品类型	大小	产品价格	产品输入日期
1234	ABX-1294	SuperWidgetX	Widget	8lbs	5	1/1/99
5678	ABY-1349	SuperWidgetY	Widget	10lbs	7	5/1/99
9012	XYZ-4567	DongleA	Dongle	10lbs	5	8/1/99
3456	WXW-6543	DongleB	Dongle	8lbs	7	10/1/99

建立此数据的模型方法有多种，具体取决于用户的要求。所有着色列可能是维，或有些着色维可能是层次结构，有些是测度，而其他则是属性(后面讲述)。

例如，可能会把产品名称和产品类型作为单独的维进行建模。查看所有产品的 Widget 会显示 Widget 总销售额，深入到产品名称维则只显示有数据的产品(如果行中有产品名称的话)，在此案例中，显示的产品为 SuperWidgetX 和 SuperWidgetY。但这不会创建难以置信的稀疏关系，因为一种产品只能有一个类型。显然，在此案例中，作为单一维和层次结构建立两种产品的模型，并且产品名称演进成产品类型，这样更有意义。

下面看看产品价格。它很容易成为加载到准备计算的 OLAP 数据库的测度。(因为它是一个仅与产品名称相关的一维测度，因而最有可能在上级为其他维加载。)但产品价格也可能是维，并具有加载时用 SQL 重新编排格式的列，这样 5 就变成“0-5”，7 变成“6-10”，等等。这使用户可以打印定价柱状图(如图 39 所示)以及回答诸如“告诉我价格在 1 到 5 美元之间的产品对产品总销售额的百分比贡献率”这样的问题。

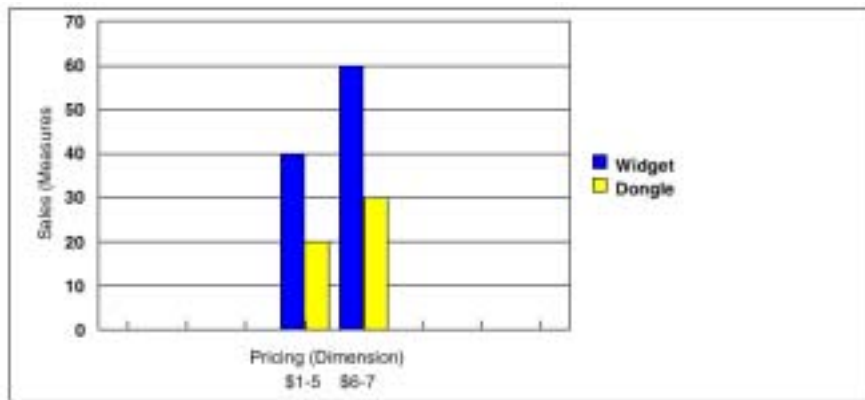


图 39 定价柱状图

因为在产品表中，数据以最低粒度(产品名称)表示，所以另一选项就会使价格成为产品的数字属性。这样就可以进行诸如“告诉我价格在 1 到 12 美元之间的产品总销售额、平均销售额、最低销售额以及最高销售额”这样的查询。

产品大小列提供了另外一种选择。它可以作为产品维中的备用层次结构，因而使用户可以按产品类型和产品大小查看总销售额。但是这不会允许用户创建交叉表 - 例如报表所有 8 磅 Dongle 的总销售额。如果这是个问题，则大小就可能是产品属性。换句话说，如果属性查询太慢(因为是动态计算)，并且许多产品重 8 磅，大小可能是存储案例维。如果用户对个别产品根本没有兴趣，而只是对产品大小和类型有兴趣，则按大小和类型维编写数据汇总更为高效。

如上面的可能性所说明的那样，有许多选项可用于从数据仓库表创建数据集市。维表可概括维、层次结构、测度以及 - 通过与其他维表联接 - 更多层次结构的任意组合。一切都取决于如何设计架构以及如何使用数据集市。如下面的具有不同维表种类的示例所示，确定包含什么通常比较容易。

A.3.1 产品和地理类型维

产品维一般都有许多属性，他们很容易转换为层次结构。对这些层次结构容易可通过类似于以下示例的问题进行定义：“给定产品类型是否始终属于单个产品类别？”如果回答是，则假定产品类型可演化为产品类别。如果回答否，则创建备用层次结构或维。

产品维可以包含大量信息。数据集市设计员必须明智地选择所需的维和层次结构，并应考虑使用 DB2 OLAP 集成服务器/Hyperion 集成服务器或 Hyperion 分析器虚拟多维数据集，或者启用“移动多维数据集”方法(其中可以按需选择不同层次结构和列)，来穿透钻取细节。

产品维还可以包含未对齐的层次结构，如图 18 所示。

表 18：产品维和未对齐层次结构

父级产品 ID	产品 ID	产品名称
空	1	塑料制品
1	2	Dongles
1	3	Widgets
3	4	SuperWidgetX
2	5	DongleA

达到此目的的一种方法是，简单地把成员名称作 ID，而在维准备编辑器中使用父子构造。产品名称成为别名。这样可以快速创建，无需自联接操作。

如果出现影响唯一性的问题(例如，塑料制品出现在不同的层次结构，但确实是指不同意义上的塑料制品)，只需把产品 ID 用作产品名称的后缀，从而在大纲中创建像“Widgets (4)”这样的成员。有许多常用方法可以解决唯一性问题，但有些方法必须使用与上表不同的格式才能实现(例如，以世代格式，而不是以父子格式)。下面介绍其中一些替代方法：

- 对付维之间的唯一性问题时，请把维名称本身用作区分符。请考虑一下存在一个销售人员维和用户名维的应用程序。两个维都可能会包含成员 Paul Turner。假定 Paul Turner 在任何维内都不会出现一次以上，则只需要把维名称用作后缀，即 Paul Turner(用户名)，Paul Turner(销售人员)。这种区分方法同时有助于用户进行导航，因为它明确了用户正在查看哪个维，并可减少混乱。
- 在维内处理唯一性问题时，方法是否最好取决于唯一性问题发生的位置。如果在不同层级发生重复，请考虑使用包含父级名称的前缀，即 SuperWidgetX (Widgets)，DongleA (Dongles)。如果在同一层发生重复，即兄弟使用同一名称，则使用维表中成员的主键来确保唯一性，例如，SuperWidgetX(5)。

未对齐层次结构的另一个潜在问题是成员的语义选择不一致。如果某个类别不包含产品——例如，一个名为“Gadgets”的全新类别，该类别就会在 OLAP 大纲中造成选择不一致问题。Gadgets 和 SuperWidgetX(例如)都将处于叶级，或处于 OLAP 大纲层次结构中的级别 0，即使一个是类别，而另一个是产品也是如此。解决此问题的一个方法是通过用户定义的属性 (UDA)，因为 UDA 允许用户通过在维表中添加列，把属性“ProductType=Category”和“ProductType=Product”分配给成员。

这样就可以进行针对这些属性的 UDA 查询，而不是进行级别或世代查询 *generational queries*，后者不适合于这种类型的树。

A.3.2 浏览器类型维

尽管浏览器类型维是 Web 分析环境所特有的，但这种维显示出一些建立数据模型时面临的问题。Web 浏览器类型实质上是包含诸如浏览器名称、版本号以及操作系统这样的信息字符串，此字符串是用一种简略的、分隔的格式进行编码的。

有些数据集市设计员只是简单地把此字符串直接插入维表，并把对其进行解码的工作留给用户界面。这基本上会创建一个两行维表，一行包含主键(链接到事实表)，另一行包含浏览器类型字符串。请看一下通过一种更为有效的数据集市设计，此信息会变得有多么丰富，如图 40 所示。

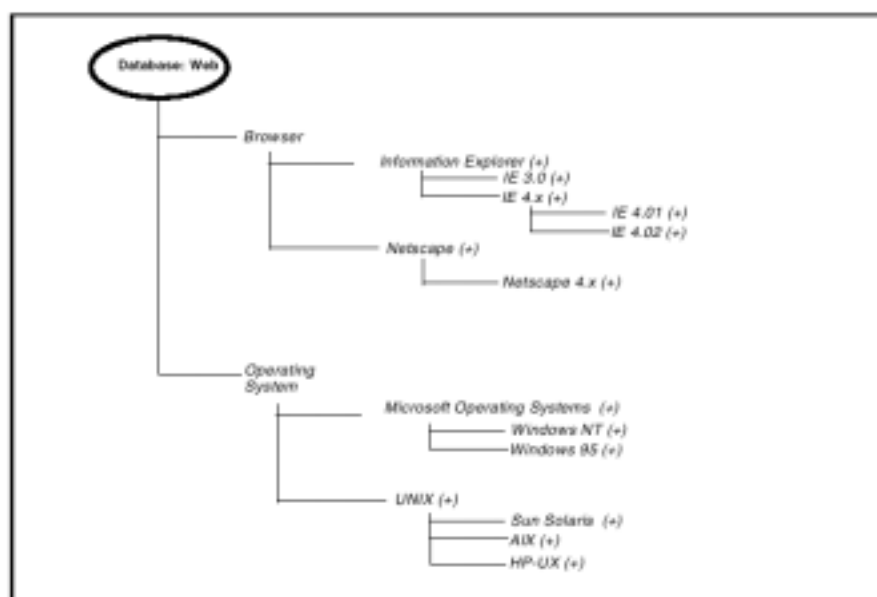


图 40 浏览器类型维

这显示出许多数据仓库的重大局限性之一：人们不知道他们可以进行、可能进行以及可能需要进行哪种分析，因而无法把所进行的分析设计到其数据仓库之中。他们进行数据建模，并以一种多维架构表示数据，但他们不从他们所记录的数据中抽取信息，而且不用属性丰富数据。一般来说，数据模型的属性越丰富(就关系意义上来说，维表中描述成员的个别列越多)，OLAP 多维数据集就越有用和越有价值。

在图 40 的示例中，可能已在日志中找到“IE 4.02”。在日志中对其进行预处理，会添加属性“IE 4.x”和“Internet Explorer”。如果同时还记录操作系统(即 Sun Solaris 2.7)，预处理就可以包括属性“Sun OS”和“UNIX”。这可以在“浏览器版本”和“浏览器平台”上实现两个附加的维表：表 19 和表 20。

表 19：浏览器版本表

浏览器版本键	浏览器版本	浏览器发行版	浏览器类别
1	IE 4.02	IE 4.X	Internet Explorer
2	IE 4.01	IE 4.X	Internet Explorer
3	Netscape 4.1	Netscape 4.x	Netscape
4	Netscape 4.5	Netscape 4.x	Netscape

表 20：浏览器平台

浏览器平台键	操作系统版本	操作系统类别
1	Windows NT	Microsoft
2	Windows 95	Microsoft
3	Solaris	UNIX

这说明了一些仓库分析如何可以极大地丰富分析。在许多情况下，解决数据模型的基本不足之处要比强迫 DB2 OLAP 利用不太好的数据更容易。

A.3.3 客户银行帐号维

分析本身具有极高粒度的数据(如各个客户银行帐号)是一项关系任务。对于这种要求，请考虑采用关系混合解决方案。像分析器这样的工具很适合于这种情况，因为这些工具可同时针对多维数据库和关系数据库进行报表，从而可以映射到一个与另一个多维数据源显示的相同逻辑星型模型。将视图链接在一起，使用户可以深入细节，同时还可以不考虑基本数据源提供者。

A.4 理解属性和基本维

在某些情况下，在数据集中包含粒度最高的维使其他一切事情(除度量之外)都成为该维的属性。请看呼叫中心应用程序，其中数据库最细的颗粒为案例号。数据库说明可能是以下内容：

案例号(基本维，由数百万个案例号构成)

分析师(案例号的属性)

位置(案例号的属性)

产品(案例号的属性)

类别(案例号的属性)

时间(案例号的属性)

当然，这会创建一个巨大的且毫无意义的报表维(案例号)，并是检索极为缓慢，因为像“显示在英国开始的所有案例”这样的计算是动态的。但是这说明一点：只有属性是维可以理解的颗粒属性时，才可以引入该属性。如果相关，一切事情都可以是多维数据集外的颗粒或多维数据集内的颗粒属性。如果不把该颗粒包含在内，属性必须是基本维。

如果不包含维“案例号”(因为它是一系列大而无意义的系统生成键)，则其他颗粒必须成为基本维。当然，这些基本维(如“产品”)可以有其自己的属性。但是它们是产品的属性，而“产品”本质上是排除的“案例号”属性。

应根据计算所需的检索时间进行衡量，这是因为可以使用一个属性维建立模型的某种东西并不意味着就应该那样建模。例如，如果有 10,000 个产品，而且其中 5,000 个具有属性“昂贵”，创建名为“成本属性”的属性维就可能不是个好主意，因为查询时编写该属性的摘要会要求 DB2 OLAP 将 5,000 个块都过一遍。如果每个块为 20K，则仅仅其他稀疏维的组合就有 100,000K 数据。在该情况下，预先计算的基本维很可能会更有意义。

A.5 处理数据加载

从数据仓库加载数据可能是非常耗时的任务。主要原因是规格化(通过把一个表分成多个表并使用键表示这些记录,从该表中删除冗余数据的过程),而这是多数数据仓库的内在组成部分。一般说来,数据仓库越规格化,数据加载越慢,因为要抽取维和数据之间的映射,必须使用更多联接。

由于事实表占据了绝大部分数据仓库空间,因而没有必要规格化维表。规格化只能节约总空间的极小百分比,但却极大地增加了数据抽取时间。

A.5.1 优化正规维表

优化从关系数据库到 OLAP 多维数据集加载的方法通常是减少数据加载时间,而且除非必需,否则不要简单地制作连接。确保 OLAP 大纲中的维在事实表中具有代表性,并确保 OLAP 大纲中有每个外键的对应成员。换句话说,每个 OLAP 多维数据集维的颗粒应与事实表中的维颗粒相同。

请看第 161 页的表 17:

产品代码(及其别名,产品名称)是此维表的最细颗粒,因为每个产品 ID 都与单个产品相关。事实表中的每个事实行都与产品维表中定义的单个产品相关。

一般情况下,加载数据需要使用产品 ID 把产品和事实表连接起来,然后使产品代码或产品名称(它们将是 DB2 OLAP 中的级别 0 成员名称)与事实相关联。然而,这种方法不适合于具有冗长事实表的特大型数据仓库。它会导致 RDBMS 运转时数据加载时间长的令人难以忍受。若试图连接表,RDBMS 可能会在达到查询限度或用完临时空间之后崩溃。

反过来，考虑主键从维表嵌入 OLAP 数据库内的别名表。无需任何连接，即可创建扫描维表，如图 41 所示。

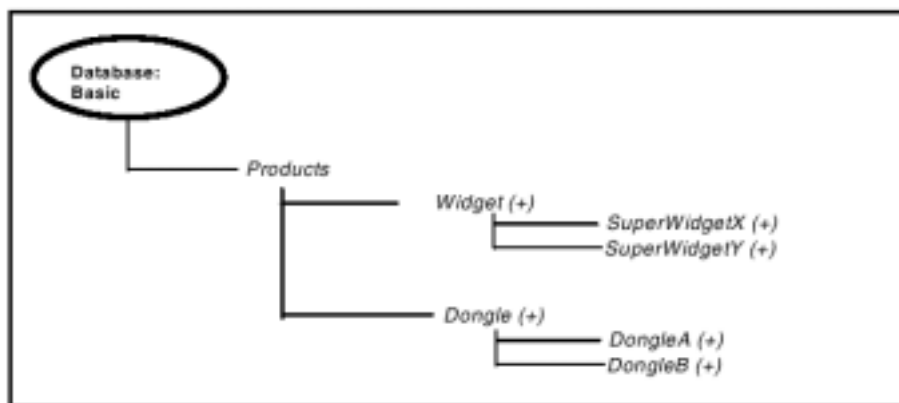


图 41 产品维大纲

请注意每个产品的别名。可以把这些别名存储在其中一个别名表中，并且不让用户看到，因为它们实际上对于报表没有用处。但是，由于 DB2 OLAP 现在知道 1234 确实是 SuperWidgetX(以及所有其他产品 ID/产品名称关系)，数据集市不再一定要将事实表与产品表联接起来，并且执行速度会快得多。

A.5.2 优化时间维

如果事实表中用于代表时间的键是日期/时间格式字段，就没有必要将该键与架构中的某个时间表连接起来，即使事实表与 OLAP 数据库中的时间维之间存在粒度差异。只需在加载时把时间键重新格式化为匹配粒度。例如，如果 OLAP 数据库维降低到月级，就在加载过程中把键转换为月，尽管该键还包含日、月和分钟信息。

A.6 结束语

数据集市设计和实施成功的关键之一就是，在设计数据仓库时，要同时考虑该数据仓库将要馈给的数据集市。如果设计数据仓库时心中只有规格化和代表性目标，而不考虑数据和元数据是否丰富和易于抽取，数据集市设计中就会遇到本来可以避免的困难。因此，设计数据仓库时，必须考虑满足最终用户的报表和分析要求。否则，您的数据仓库就会成为数据“监狱”，使用户无法使用需要使用的信息。

附录 B 集成服务器实施原则

本附录介绍实施 Hyperion 的 Hyperion 集成服务器或 IBM 的 IBM DB2 OLAP 集成服务器，以便从关系源创建 DB2 OLAP 多维数据集时需要遵循的原则和建议。我们将使用术语“集成服务器”(简称为“IS”)专指这两种产品。

本附录由 Cheryl A. McCormick 编写，他是 Hyperion Solutions 公司数据集成服务组的一名高级首席顾问。Cheryl 致力于混合使用关系数据库和 OLAP 多维数据库设计和实施数据集市/数据仓库。

B.1 概述

集成服务器提供关系星型模型和强大的 DB2 OLAP 服务器之间的关键链接。

集成服务器产品家族提供一种将关系星型模型中相关数据快速传输到多维数据库的方法(参见图 42)。集成服务器产品家族提供有助于您执行以下任务的图形工具：

- 在关系数据库中的表、视图和列中创建逻辑 OLAP 模型。您所创建的 OLAP 模型是一种由事实表构成的逻辑星型模型，周围是相关维表。
- 使用 OLAP 模型创建元大纲。这是一种大纲模板，它包含生成 DB2 OLAP 大纲所需的结构和规则。
- 使用元大纲创建和填充 DB2 OLAP 多维数据库。

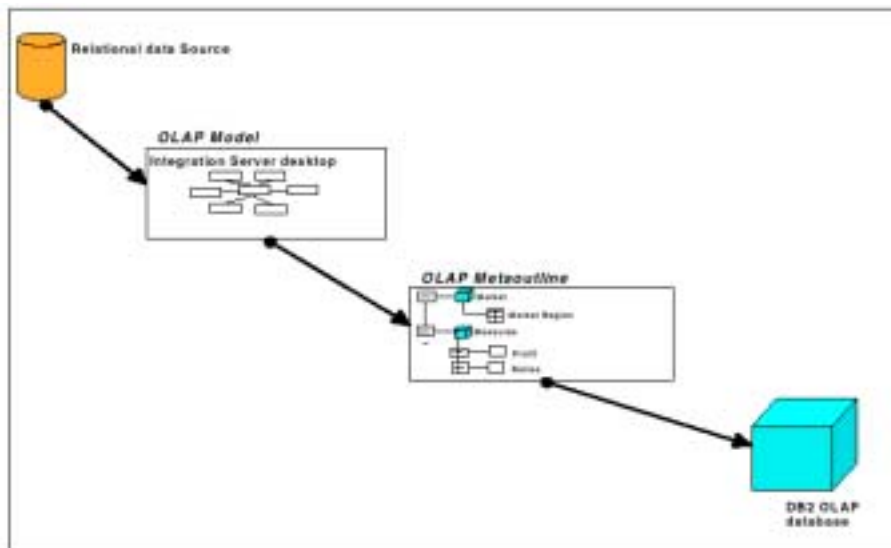


图 42. DB2 OLAP 集成服务器流程说明

集成服务器由两大组件构成：桌面和服务端(参见图 43)。

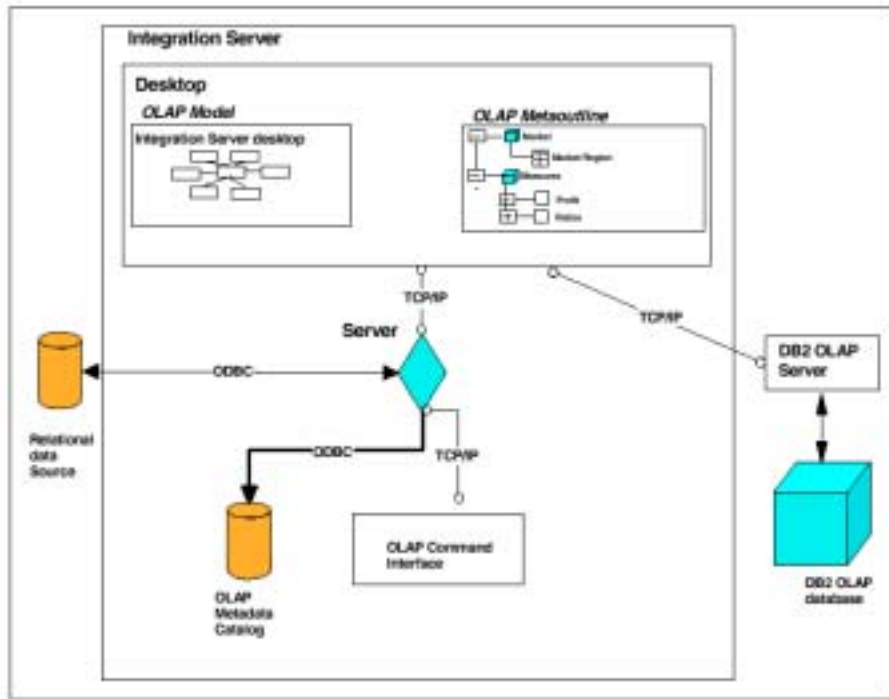


图43. IS 组件

使用 IS 的工作流程

从关系数据源创建 DB2 OLAP 数据库：

1. 建立一个基于关系数据源中表的 OLAP 模型。IS 可存储 OLAP 模型以及检索 OLAP 元数据编录中的相关表所必需的信息。
2. 从 OLAP 模型中创建一个元大纲。IS 把元大纲存储在 OLAP 元数据编录中。
3. 把成员和数据加载到 DB2 OLAP 数据库中。
4. 用新成员和数据更新 DB2 OLAP 数据库。

B.1.1 OLAP 模型

每个 OLAP 模型均包含一个星型模型。OLAP 模型以此概念为基础，可以把关系数据库按事实或事实的维进行分类。事实是数据库中的数字、可变值，例如，销售额以及销售件数。与事实关联的是相关数据值，这些数据值提供诸如存储位置和售出产品的产品 ID 这样的附加信息。每个 OLAP 模型包含一个事实表、一个或多个维表以及一个或多个维分支。OLAP 模型可能包含时间维和账户维。

与其他集成产品不同的是，IS 创建一种属于逻辑模型但不是物理星型模型的 OLAP 模型。OLAP 模型是数据值的逻辑表示。您从关系数据库的表中选择这些数据值，并希望 DB2 OLAP 中予以报表。

B.1.2 OLAP 模型和元大纲

使用 OLAP 模型创建一个或多个元大纲。元大纲包含建立 DB2 OLAP 大纲和把数据加载到 DB2 OLAP 中所需的基本结构。通过把原始 OLAP 模型保存在不同名称下，并在需要时对其进行编辑以满足报表要求，您可以把一个 OLAP 模型用作另一个 OLAP 模型的基础。您可以为建立元大纲而创建任意数量的 OLAP 模型。但是，每个元大纲以一个特定 OLAP 模型为基础。

OLAP 模型具有以下功能：

- 它们可以重复利用。您可以把同一 OLAP 模型作为多个元大纲的基础。
- 它们提供一个抽象层，该抽象层把 DB2 OLAP 数据库大纲与关系数据库中的更改隔离开来。
- 它们使您能够创建用以组织并汇总关系数据库中数据的层次结构。您可以在多个元大纲中使用这些层次结构。

B.2 决定：初始IS实施

首次使用 IS 时，无论是自己实施，还是为客户实施，您都应遵循一条非常特别的路径：从 DB2 OLAP 多维数据集向后运作！具体说来，一般任务应包括以下内容：

- 定义旨在定义维、层次结构、属性维、别名、UDA 等的 DB2 OLAP 要求。
- 定义和测试公式及计算脚本。
- 定义最终用户穿透钻取要求。
- 检查星型模型/ DB2 OLAP / IS 环境，并确定最有效的配置。
- 安装 IS 桌面和 IS 服务器。
- 利用已定义的 DB2 OLAP 大纲，开发支持 DB2 OLAP 维和穿透钻取报表要求的星型模型设计。
- 开发星型模型的索引策略。
- 如果要求实现过程自动化，就开发/测试/应用 OLAP 命令行界面命令。

应连同 IS 手册中包含的步骤，仔细检查上述每个步骤。

B.2.1 没有数据仓库或数据集市或星型模型

在多数 IS 实施中，DB2 OLAP 顾问处理 DB2 OLAP 实施工作：顾问定义要求、编写和测试计算脚本和公式以及定义所需的任何 ESSCMDs。一旦定义了大纲，或定义并测试了至少 75% 的大纲，集成服务器顾问就可以开始 IS 实施工作。在尚未准备好数据仓库、数据集市或星型模型的情况下，这种在实施 IS 之前定义 DB2 OLAP 多维数据集的方法非常成功。

B.2.2 已有数据仓库或数据集市或星型模型

如果数据仓库、数据集市或星型模型已经到位，则初始步骤应包括检查现有星型模型，以及把星型模型与 DB2 OLAP 大纲(维)进行比较。如果现有星型模型包含 DB2 OLAP 所需的维，而且 FACT 表的粒度也与 DB2 OLAP 要求相一致，那么，初始开发工作就应集中在使用现有星型模型上。除了维和粒度要求之外，您还必须确定星型模型的索引策略将是否支持要求的性能。

概要：

1. 确定星型模型是否包含 DB2 OLAP 所要求的维。
2. 确定 FACT 表数据中的粒度是否满足 DB2 OLAP 要求。
3. 确定星型模型的索引策略将是否满足 IS 性能要求。

B.3 从头开发星型模型

在没有数据仓库、数据集市或星型模型的环境中，需要开发星型模型来支持 IS 和 DB2 OLAP。星型模型驻留在其中一个 IS 支持关系数据库中，并通过本机 ODBC 驱动程序或通过 IS 提供的 MERANT ODBC 驱动程序建立连接。

B.3.1 维模型

星型模型的另一个名称是维模型。实体关系模型非常对称，而维模型则不对称。架构中心有一个大的主表，若干较小的附带表环绕在其周围。此中心表称为 FACT 表，包含数字数据；周围的小表为维表，包含元数据。

B.3.1.1 设计FACT表

事实表包含 DB2 OLAP 随后聚集和报表的值。您想在 DB2 OLAP 中报表的所有度量值都通过事实表以及账户维进行引用，后者可以是事实表的一个精确副本，并且通常采用以下两种格式之一：

1. 每个维都由列表示，而每个 DB2 OLAP MEASURE 也由列表示。每个行包含每个维的一个有效成员以及每个 MEASURE 下的数字值。
2. 每个维(包括 ACCOUNTS 维)都由列表示。标有“DATA”的列包含每个 DB2 OLAP 维交点的数字数据值。

当 MEASURES 的数目有限时，就使用 Method #1。当采用财务应用程序创建星型模型时，ACCOUNTS 维通常包含大量成员。为此，通常把 Method #2 用于财务应用程序。

标准 FACT 表 [Method #1]

如前所述,标准 FACT 表将包含每个维的列,加上每个 ACCOUNT 成员或 MEASURE 的列。每个 MEASURE 的数据类型必须是数字数据类型。如果使用其他数据类型,IS 就不会把该列识别为 MEASURE。

在 TBC 示例数据库中,FACT 表是使用 Method #1 创建的。

图 44 是维模型中显示的 FACT 表的示例。



图 44. 事实表示例

STATEID、PRODUCTID、SCENARIOID 和 SUPPLIERID 代表 DB2 OLAP 大纲中的一个维。

SALES、COGS、MARKETING、PAYROLL、MISC、OPENNINGINVENTORY 和 ADDITIONS 代表 ACCOUNT 维中的 MEASURES。

在此示例中,STATEID、PRODUCTID、SCENARIOID 和 SUPPLIERID 为所有外键。在 FACT 表中使用外键是标准的数据仓库做法。

每个维表都包含“主键”。主键的特征之一是唯一性。每个 FACT 表维代码都应是该维表主键的一个外键。例如,维表中的 PRODUCTID 就是 PRODUCT 维的主键 [它是唯一值,也是 DB2 OLAP 要求]。FACT 表中的 PRODUCTID 是外键。这就是说,把 PRODUCTID 加载到 FACT 表之前,PRODUCTID 必须存在于 PRODUCT 维。此主/外键关系还使 DB2 OLAP 元数据和数据保持同步。有关如何使用外键的更多信息,请参阅数据仓库文档。

备用 FACT 表 [Method #2]

为财务应用程序建立维模型时，Method #2 通常是首选方法。财务应用程序通常把账户表用作 ACCOUNT 维。账户表可以包含数千个账户。即使是在一个较高的详细级 [即不在账户表的最低账户级] 建立 DB2 OLAP，也可以创建两到三千个基本账户。把每个基准级账户都放在同一列中会创建一个极为庞大的表，因而维护起来十分困难。Method #2 使用 ACCOUNT 维单独的列和实际数字数据单独的列。如图 45 所示。

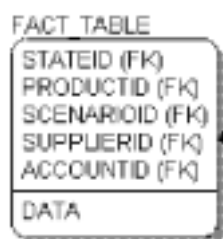


图 45. 备用事实表示例

ACCOUNT 表包含 ACCOUNT 元数据，其中包括 ACCOUNT 层次结构。DATA 列包含与每个维的列相关的实际数字数据。

FACT 表的粒度

IS 根据默认值或 IS 元大纲中定义的“Build to here”准则，直接从 FACT 表加载数据。FACT 表通常及其大：包含数万条记录的 FACT 表很平常。确定 FACT 表的粒度时，应主要考虑是否有一个必须在其中加载和计算 DB2 OLAP 的有限时间窗口。如果有一个有限时间窗口，就可以使用聚集 FACT 表策略。

使用此策略，检查每个 DB2 OLAP 维，并确定最有效的聚集布局。例如，原始 FACT 表粒度处于发票行详细级，并按“产品”和“月份”编写 DB2 OLAP 多维数据集的摘要。通过创建聚集 FACT 表 TIME = Month 和 PRODUCT = By Product [并非发票行项目] 编写摘要会极大地减少聚集 FACT 表中的行数。使用原始 FACT 表，IS 会实际地 SUM 到 DB2 OLAP 数据要求级。使用聚集 FACT 表，IS 则不必汇总数据，这样可以缩短 SQL 检索时间。最终结果是：数据加载更加快速。

FACT 表数据类型

FACT 表中的所有 MEASURES，无论是使用 Method#1 还是 Method#2，必须是数字数据类型。对于 Method #1，每个标为 MEASURE 的列都必须是数字数据类型。标识维的列的数据类型必须与其关联“主键”的数据类型相同。

对于 Method #2，“DATA”列必须是数字数据类型。如上所述，标识维的列的数据类型必须与其关联“主键”的数据类型相同。

B.3.1.2 设计维表

维表包含元数据。高效设计的维表会在 IS 元数据加载和数据加载方面产生极大的差别。如前所述，每个维表必须包含一个主键。此唯一键还代表 DB2 OLAP 叶级成员。这并不是说，不能使用 IS 元大纲中的“Build to here”功能以较高级别创建 DB2 OLAP 多维数据集。考虑到 DB2 OLAP 的“唯一性”要求，使用主键是一个理想的选择。

除了 DB2 OLAP 叶级成员之外，还应为关联的“ALIAS”和“用户定义的属性 (UDA)”定义列。因此，前三列定义 DB2 OLAP 叶级成员。制定维表时，请牢记 DB2 OLAP 的要求。例如，DB2 OLAP 把成员名称和别名的字段大小限制在最长 79 个字符。这一限制应包含在维表设计中。

维表：层次结构

IS 支持建立 DB2 OLAP 层次结构的各种表布局。应优先选择一种表布局。此格式是一种混合世代构造，基本上主键仍然作为第一列(如果适用，再加上关联别名和 UDA)。下一组列代表一个“世代构造”，GEN01 为最高父级，GEN02 次之，依此类推。我们必须继续在右边的列中建立世代，并且不在任何世代列中包含叶级成员 [主键]。

表 21 显示此维表布局的示例。

表 21. 具有层次结构的维表示例

PRODUCT_CODE	PRODUCT_ALIAS	GEN_01	GEN_02
100_10	COLA	COLA VS NONCOLA	COLAS
100_20	DIET COLA	COLA VS NONCOLA	COLAS
100_30	CAFFEINE FREE COLA	COLA VS NONCOLA	COLAS
200_10	ROOT BEER	COLA VS NONCOLA	NON-COLAS
200_20	DIET ROOT BEER	COLA VS NONCOLA	NON-COLAS
200_30	VANILLA CREAM	COLA VS NONCOLA	NON-COLAS
200_40	DIET VANILLA CREAM	COLA VS NONCOLA	NON-COLAS
400_10	GRAPE	COLA VS NONCOLA	FRUIT SODA
400_20	DIET GRAPE	COLA VS NONCOLA	FRUIT SODA
400_30	ORANGE	COLA VS NONCOLA	FRUIT SODA
400_40	DIET ORANGE	COLA VS NONCOLA	FRUIT SODA

PRODUCT_CODE 为主键。GEN_01 为主层次结构的最高父级。GEN_02 为该层次结构的次高父级。如果父级需要别名和 UDA，则把必需的列添加到关联世代列的右边。

维表：备用层次结构

备用层次结构是非常强大的 DB2 OLAP 功能。处理备用层次结构的首选方法是在主层次结构右边的列中建立备用层次结构。并不是所有叶级成员或维表的主键都是备用层次结构中的成员。对于那些成员，应把备用层次结构的列保留为“NULL”。

表 22 显示一个维示例，该维包含主层次结构和备用层次结构。

表 22. 具有备用层次结构的维表

PRODUCT_CODE	PRODUCT_ALIAS	GEN_01	GEN_02	ALT_GEN_01	ALT_GEN_02
100_10	COLA	COLA VS NONCOLA	COLAS	REGULAR VS DIET	REGULAR
100_20	DIET COLA	COLA VS NONCOLA	COLAS	REGULAR VS DIET	DIET
100_30	CAFFEINE FREE COLA	COLA VS NONCOLA	COLAS	REGULAR VS DIET	REGULAR
200_10	ROOT BEER	COLA VS NONCOLA	NON-COLAS	REGULAR VS DIET	REGULAR
200_20	DIET ROOT BEER	COLA VS NONCOLA	NON-COLAS	REGULAR VS DIET	DIET
200_30	VANILLA CREAM	COLA VS NONCOLA	NON-COLAS	REGULAR VS DIET	REGULAR
200_40	DIET VANILLA CREAM	COLA VS NONCOLA	NON-COLAS	REGULAR VS DIET	DIET
400_10	GRAPE	COLA VS NONCOLA	FRUIT SODA	REGULAR VS DIET	REGULAR
400_20	DIET GRAPE	COLA VS NONCOLA	FRUIT SODA	REGULAR VS DIET	DIET
400_30	ORANGE	COLA VS NONCOLA	FRUIT SODA	REGULAR VS DIET	REGULAR
400_40	DIET ORANGE	COLA VS NONCOLA	FRUIT SODA	REGULAR VS DIET	DIET

维表：未对齐层次结构

一种最难建立的层次结构是未对齐层次结构。未对齐层次结构是一种包含不等级别数量的层次结构。一个父级可能有两（2）个级别在叶级成员前面，而另一个父级则可能有五（5）个级别在叶级成员前面。

要为具有未对齐层次结构的维建立维表，请使用上面概述的方法。把 GEN_01 作为顶级父级，右边的每个列都包含下一个父级。当叶级成员是该层次结构中的“下一个”成员时，请输入“NULL”值。因此，在表 22 的示例中，第一个父级的 Gen_01 和 GEN_02 中包含有值，而 GEN_03 到 GEN_05 中则包含有 NULL 值。第二个父级在 GEN_01 到 GEN_05 中包含有值。任何 GEN_XX 列下都不会出现叶级值。

表 23 中显示具有未对齐主层次结构的示例维。

表 23. 具有未对齐层次结构的维表

PRODUCT_CODE	PRODUCT_ALIAS	GEN_01	GEN_02	GEN_03	GEN_04	GEN_05
100_10	COLA	COLA VS NONCOLA	COLAS	NULL	NULL	NULL
100_20	DIET COLA	COLA VS NONCOLA	COLAS	NULL	NULL	NULL
100_30	CAFFEINE FREE COLA	COLA VS NONCOLA	COLAS	NULL	NULL	NULL
200_10	ROOT BEER	COLA VS NONCOLA	NON-COLAS	NULL	NULL	NULL
200_20	DIET ROOT BEER	COLA VS NONCOLA	NON-COLAS	NULL	NULL	NULL
200_30	VANILLA CREAM	COLA VS NONCOLA	NON-COLAS	NULL	NULL	NULL
200_40	DIET VANILLA CREAM	COLA VS NONCOLA	NON-COLAS	NULL	NULL	NULL
400_10	GRAPE	COLA VS NONCOLA	FRUIT SODA	GRAPE FAMILY	GRAPE REGULAR	SAMPLE 1
400_20	DIET GRAPE	COLA VS NONCOLA	FRUIT SODA	GRAPE FAMILY	GRAPE DIET	SAMPLE 2
400_30	ORANGE	COLA VS NONCOLA	FRUIT SODA	ORANGE FAMILY	ORANGE REGULAR	SAMPLE 3
400_40	DIET ORANGE	COLA VS NONCOLA	FRUIT SODA	ORANGE FAMILY	ORANGE DIET	SAMPLE 4

雪片问题

讨论维层次结构时，总是要谈及有关使用“雪片”结构的问题，而不是与使用标准星型模型相关的问题。雪片结构包含一个或多个维表的分支。例如，PRODUCT 维可能会有一个“FAMILY”表的备用层次结构。备用层次结构的层次信息将存储在单独的表中。FAMILY 表就会是 PRODUCT 维表的一个分支。

图 46 显示雪片维示例。

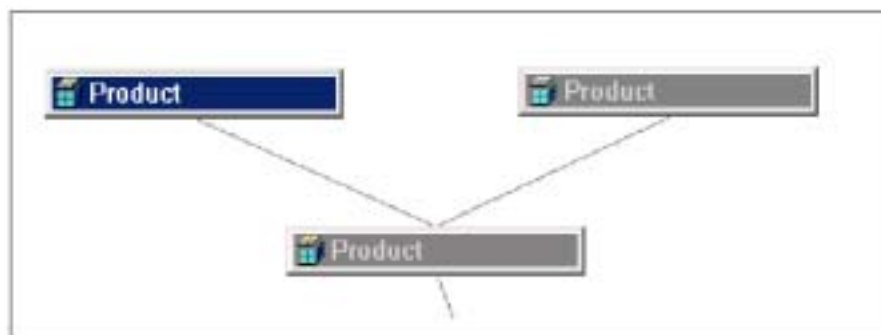


图 46. 雪片示例

非雪片星型模型仅包含一个 PRODUCT 维表，其中在单个表中定义了主层次结构和备用层次结构。

我们不提倡使用“雪片”方式，因为这样在运行查询时会需要额外的 SQL 时间。雪片架构需要额外的 SQL 时间来考虑维表与雪片表之间的多个联接。性能时间是 IS 和 DB2 OLAP 约定中的关键考虑事项之一。

B.3.1.3 星型模型索引策略

尽管每个星型模型各不相同，但在 IS 实施的初步阶段仍然可以利用一个基本的索引策略。请牢记，就索引所需的空间而言，索引比较昂贵。一般情况下，索引会占用尽可能多的数据。

索引策略可以进行或中断 IS 实施。

主键和外键的确定把这些键标识为唯一索引。

对于初步索引策略，您应编写 SELECT 语句中包含的每个列的索引，该语句是由 IS 生成的。您应确保检查元数据 SQL 语句和数据加载 SQL 语句，以识别可能的候选索引。另外，如果同时使用穿透钻取报表，请检查穿透钻取报表和索引相应生成的 SQL 语句。

B.3.1.4 代理键

代理键不是 IS 必需的，它们只是简单的“良好数据仓库”，并应在所有星型模型中实施。代理键的定义是什么？Per Ralph Kimball：

“在数据仓库中，代理键是自然生产键的必需归纳，并且是数据仓库设计的基本元素之一。数据仓库环境中的维表和事实表之间的任何联接都应代理键为基础，而不是以自然键为基础。”

其基本意思是，用唯一的按顺序分配的整数替换您的自然键。使用代理键的原因有许多，但对于 IS 最重要的原因是改进了查询性能时间。

有关使用代理键方面的更多信息，请参阅 Ralph Kimball 的任何一本数据仓库论著，或从 DBMSMag.com 网站搜索与使用和部署代理键相关的文章。

B.4 IS安装和环境配置

DB2 OLAP / IS /星型模型环境很大程度上取决于客户环境。回答以下问题有助于您为 DB2 OLAP 和 IS 确定最有利的环境。

1. 获得以下服务器规范信息：
 - a. 服务器 RAM
 - b. 服务器上的处理器数目
 - c. 硬盘驱动器空间数量
 - d. 这是一个专用服务器，还是其他应用程序也使用此服务器？
 - 确定共享该服务器的应用程序以及使用百分比。
2. 此服务器上运行了多少个 DB2 OLAP 应用程序？
3. 此服务器上的 DB2 OLAP 多维数据集多长时间更新一次？多长时间运行一次计算脚本？完成计算需要多长时间？
4. 目前是否使用 IS 生成和加载 IS 多维数据集？
5. 目前对星型模型多长时间更新一次？填充或更新星型模型需要多长时间？

6. 是否已测试客户机网络性能？网络中是否存在已知问题？

针对可用于此项目的每个服务器回答这些问题，有助于确定最佳 DB2 OLAP / IS / 星型模型配置。同时，请牢记以下信息：

1. 当 DB2 OLAP 运行计算时，使用一个处理器。如果有另一项计算，就需要另一个处理器。在双处理器服务器上，两项 DB2 OLAP 计算可能已经创建了一个队列(操作系统也需要一些处理器时间)。
2. 当 IS 运行元数据加载或数据加载时，一个处理器在用。
3. 当 RDBMS 运行 IS 生成的 SQL 时，一个处理器在用。
4. 当 IS 生成 DB2 OLAP 多维数据集、元数据构造时，运行 SQL，而且在把整个结果集加载到 DB2 OLAP 之前，会将其存储起来。将元数据加载到 DB2 OLAP 之前，IS 服务器处理 DB2 OLAP 验证规则。
5. 当 IS 把数据加载到 DB2 OLAP 之中时，生成 SQL，而且以 100 条记录增量把数据加载到 DB2 OLAP 之中。这 100 条记录递增量是可以修改的。

B.4.1 IS 环境配置建议

在上面信息的基础上，应考虑以下建议：

1. 如果以下条件适用，DB2 OLAP / IS / 星型模型可全部驻留在同一个服务器上：
 - a. 服务器规范包括四路处理器，2 GB RAM，以及最大的硬盘空间。
 - b. 服务器上最多可以存在两个 DB2 OLAP 多维数据集。每月更新和计算 DB2 OLAP 多维数据集。
 - c. 按照一个合理的时间范围增量更新星型模型。

在此方案中，应对排队情况加以限制，因为可用处理器数目和 RAM 本身是有限的。

过程可能是下面的样子：

- 更新星型模型(星型模型使用一个处理器 + 必需的 RAM)。
- IS 把元数据和数据加载到 DB2 OLAP 之中(更新星型模型之后，星型模型使用一个处理器 + 必需的 RAM)。
- DB2 OLAP 计算多维数据集(星型模型使用一个处理器 + 必需的 RAM)。

加载两个 DB2 OLAP 多维数据集会明显地占用两倍资源。在此情况下，我们预料仍然不会发生排队或性能问题。

2. 如果以下条件适用，IS / 星型模型可驻留在一个服务器上，而 DB2 OLAP 驻留在一个单独的服务器上：
 - a. 服务器规范包括：双处理器，2 GB RAM 以及最大的硬盘空间。
 - b. 有两个以上 DB2 OLAP 多维数据集处于生产状态，或两个 DB2 OLAP 多维数据集每月更新一次以上。DB2 OLAP 多维数据集应驻留在一个单独的服务器上。
 - c. 星型模型每月更新一次以上，例如，每周更新一次。

在此方案中，应对排队情况加以限制，因为我们把需要处理器时间和 RAM 的过程独立开来。过程可能是下面的样子：

- 每周更新星型模型(星型模型使用一个处理器 + 必需的 RAM)。
- IS 每周把元数据和数据加载到 DB2 OLAP 之中(更新星型模型之后，星型模型使用一个处理器 + 必需的 RAM)。
- DB2 OLAP 计算多维数据集(星型模型使用(一个处理器 + 必需的 RAM)* DB2 OLAP 多维数据集的数目)。由于 DB2 OLAP 运行在另一个服务器上，运行多个计算所需的处理器数目和 RAM 不会影响 IS 或更新星型模型。

3. 如果以下条件适用，请考虑使 IS、星型模型和 DB2 OLAP 全部驻留在单独的服务器上：
 - a. 服务器不是专用服务器。果真如此，则对您的应用程序进行“负载平衡”极为重要。您必需了解每个服务器上运行的其他应用程序，并将其 RAM 和处理器要求与 IS / DB2 OLAP 和星型模型处理器以及 RAM 要求进行平衡。
 - b. IS 每天把元数据和数据加载到多个 DB2 OLAP 多维数据集。
 - c. 最终用户穿透钻取活动量很大。
 - d. 目前每天计算多个 DB2 OLAP 多维数据集。

B.5 IS 模型

从本节起,本文将指出好的做法,并把补充信息提供到 IS 文档。循序渐进地建立 IS 模型和 IS 元大纲时,应查阅 IS 文档。

B.5.1 建立 IS 模型

建立 IS 模型时,把星型模型中定义的所有维合并到 IS 模型之中。通常情况下,这可能有 15 或 20(或更多)维!通过把所有可能的维包含在 IS 模型中,保证 DB2 OLAP 大纲的参照完整性(当然,这里假定星型模型定义了必需的主键和外键的关系!)。

B.5.1.1 TIME 维

建立 IS 模型的第一步是把 FACT 表拖动到右面板的中心。当系统询问是否要添加 TIME 维时,应回答“否”。

这是为什么呢?FACT 表一般包含数百万条记录。通过使用 FACT 表建立 TIME 维,您基本上会在 FACT 表的 TIME 列上执行“Select distinct”,以便填充 TIME 维。相反,则要创建一个单独的 TIME 维表,并使用代理键连接到 FACT 表。此步骤对性能的影响很大。

动态 TIME 系列

创建层次结构和利用“动态 TIME 系列”功能是通过以下步骤完成的:

1. 创建包含 FACT 表中代理键的一列的“TIME 维”表,并使用 DATETIME 数据类型,即包含月终(或任何特定)日期的列(参见表 24)。

表 24. 创建时间维表

TIME_KEY	TIME
1	1/31/2001
2	2/28/2001
3	3/31/2001
4	4/30/2001
5	5/31/2001
6	6/30/2001
7	7/31/2001
8	8/31/2001
9	9/30/2001
10	10/31/2001
11	11/30/2001
12	12/31/2001

2. 将“TIME 维”表拖动到 IS 桌面上,并添加 TIME 维。把 TIME 维联接到 FACT 表。“创建新维”时,确保把 TIME 维标识为“‘TIME’维类型”。
3. 突出显示“TIME 维”时,请依次选择“视图”/“属性”/“表”。
4. 在“表属性”对话框中,选择“COLUMNS”选项卡。选择“DATETIME”列,然后选择右边的“日期-层次结构”框。
5. 检索“日期-层次结构”对话框,如图 47 所示。

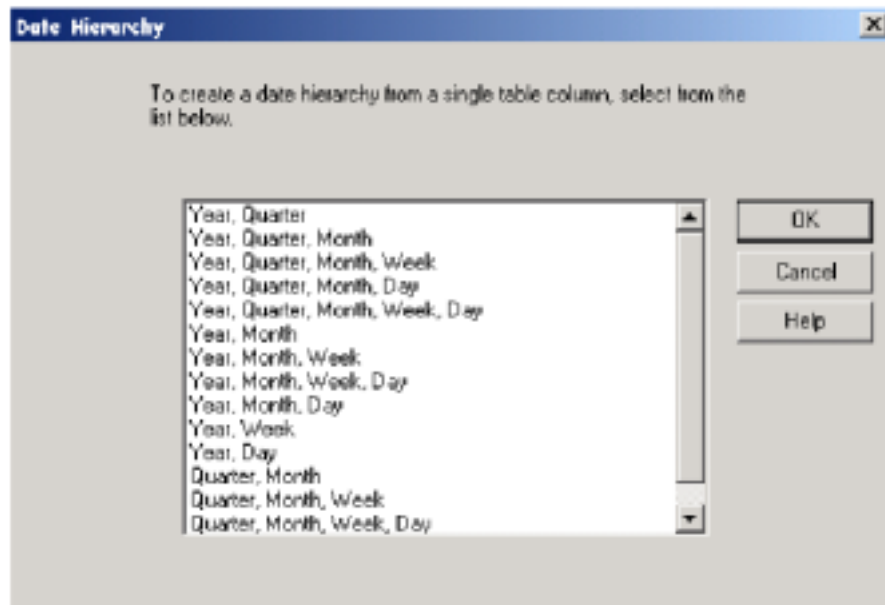


图 47. 创建日期层次结构

6. 选择适当的 TIME 层次结构。
7. 进行日期-层次结构选择之后,把“TIME 维”更改为图 48 所示的样子,给 TIME 维表添加另外三个列:“月份”、“季度”和“年份”。

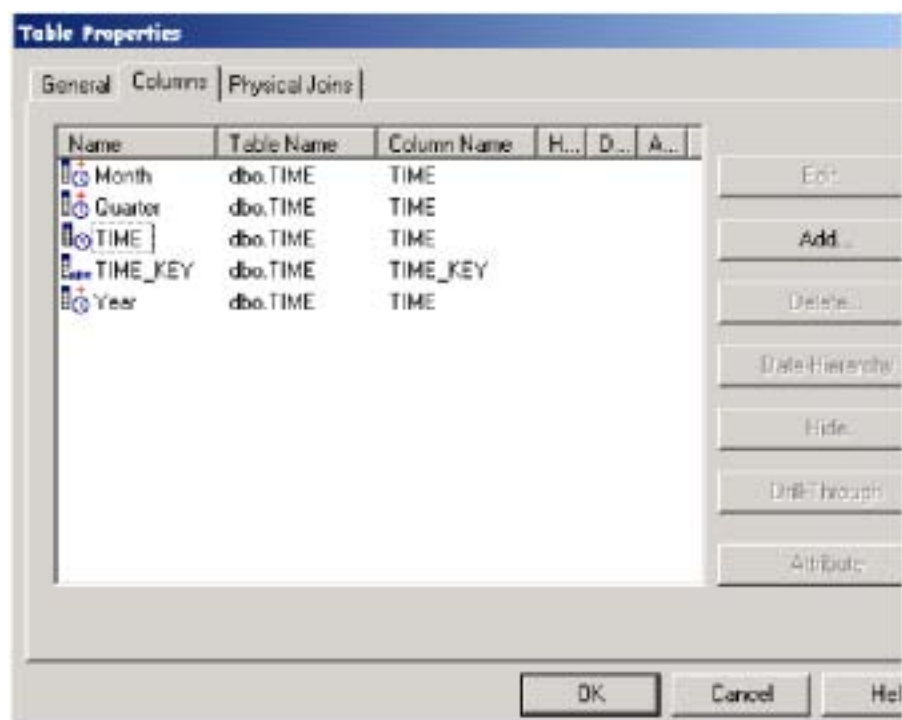


图 48. 更改的时间维

8. 下面的步骤是在“IS 元大纲”中执行的：
 - a. 突出显示“TIME 层次结构”中的 YEAR 级，转到“成员属性”，并选择 DYNAMIC TIME 框(参见图 49)。
 - b. 在 TIME 层次结构中，完成 QTR 级的以前步骤。

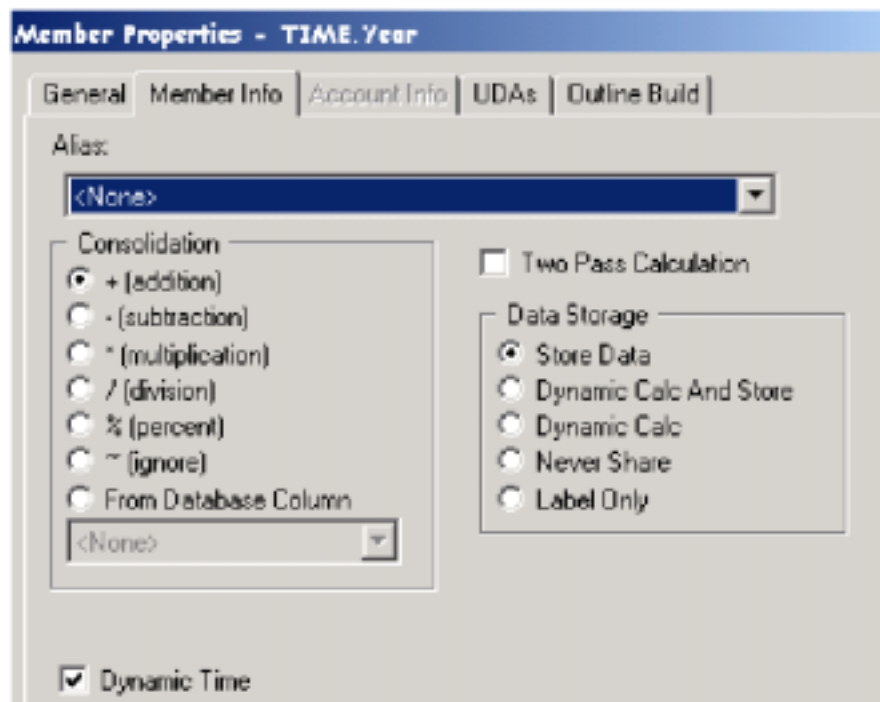


图 49. 选择动态时间

B.5.1.2 ACCOUNT 维

如果目前在用 Method #1 建立 FACT 表(MEASURES 作为列反映在“FACT 表”中)，当系统询问是否在添加 FACT 表后建立“ACCOUNT 维”时，必须回答“YES”。

如果目前在用 Method #2 建立 FACT 表(MEASURES 反映在“FACT 表”的单个列中)，当系统询问是否在添加 FACT 表后建立“ACCOUNT 维”时，必须回答“NO”。我们需要为 IS 在 FACT 表中标识特定 ACCOUNT 列。然后，可以把它拖动到 ACCOUNT 维表上，并把该维标识为通用维：

您必须在把 ACCOUNT 维标识为“账户”类型维。

B.5.1.3 通用维表

建立通用维表时，您应考虑转换，并建立未对齐层次结构。

转换

请考虑使用 TRANSFORMATIONS 下可用的 FIND 和 REPLACE 功能的 SQL 性能成本。如果所有可能的 DB2 OLAP 多维数据集均必需 FIND 和 REPLACE 功能(例如,替换嵌入的“ with a single ”),您应考虑在抽取、转换和加载阶段或在加载星型模型之前,对此进行处理。一旦开始进入星型模型,清洗数据就比每次使用 IS 将数据加载到 DB2 OLAP 中更有效率。

建立未对齐层次结构

利用推荐的星型模型格式,通过 IS 建立未对齐层次结构非常简单!从“编辑层次结构”对话框中,您可以拖动到每个 GEN_XX 列上,并采用从上到下的顺序(参见图 50)。

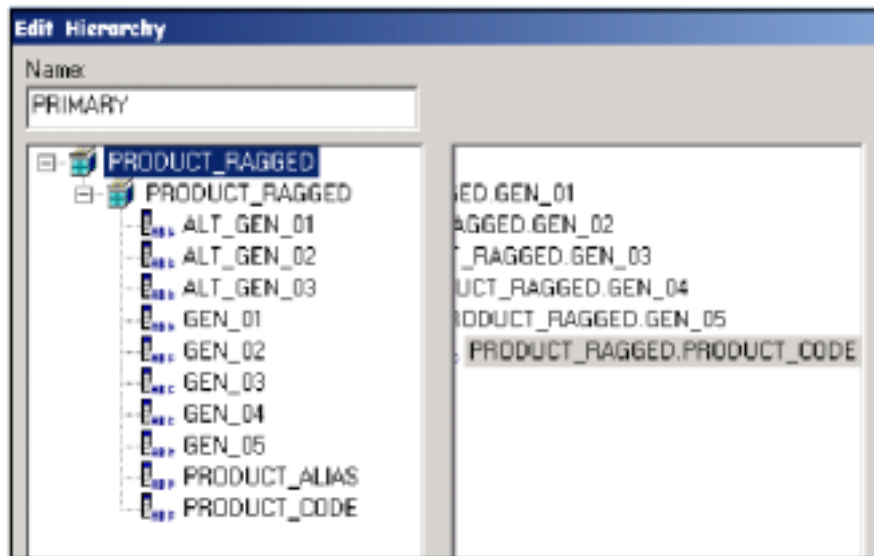


图 50. 建立未对齐层次结构

最后一代应该是叶级成员。

任何 GEN_XX 列中包含的 <NULL> 值都会被忽略。图 51 显示 DB2 OLAP 中生成的 RAGGED_PRODUCT 维的结果。

同样地,高效设计的星型模型维表使 IS 实施困难的未对齐层次结构成为非常简单的过程。

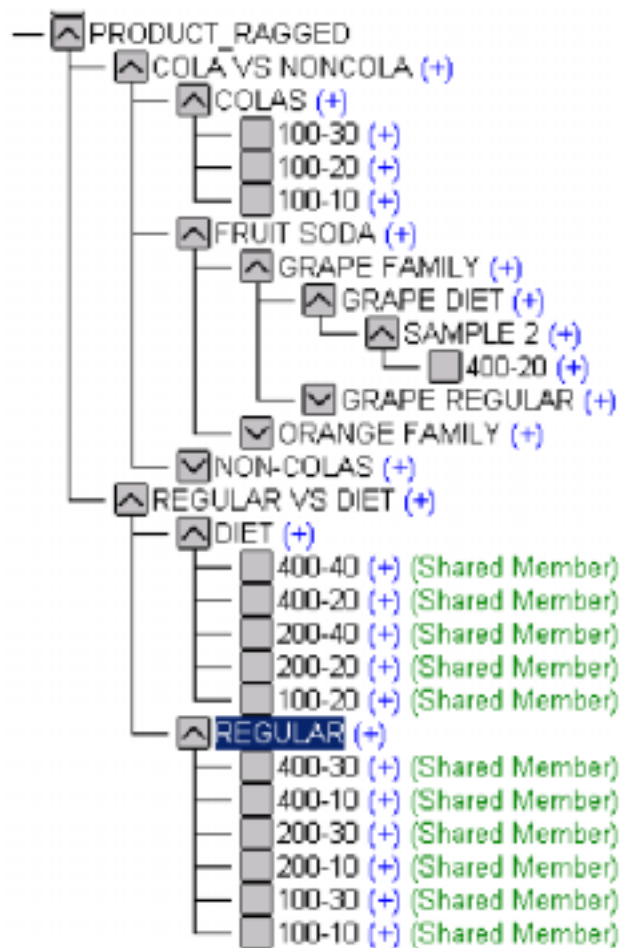


图 51. 产生的未对齐维

B.6 IS 元大纲

现在就可以把星型模型中的所有维以及可用的 IS 模型包含在 DB2 OLAP 大纲之中。

B.6.1 建立 IS 元大纲

创建新的元大纲时，所有可用维都出现在左边的面板上。除了在 IS 模型中建立的层次结构外，IS 开发人员还可以在 IS 元大纲中从头建立层次结构。在 IS 元大纲中建立的层次结构只适用于该特定元大纲。

B.6.1.1 MEASURES 维

如果您使用 Method #2 定义了 MEASURES 维，就需要对多元大纲进行一次修改，以便使 IS 识别 MEASURES 列。

要定义 MEASURES 列，请突出显示元大纲名称，并单击鼠标右键。从下拉菜单中选择“属性”。选择“**数据库度量**”选项卡，然后单击“ADD”按钮(参见图 52)。

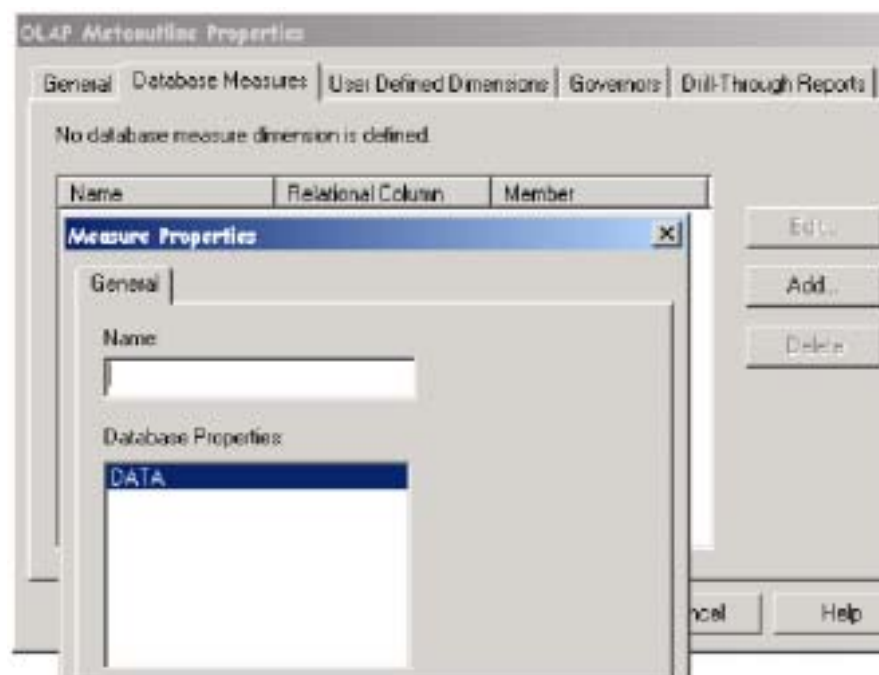


图 52. 定义 MEASURES 维

任何数字数据类型都会出现在“数据库属性”列表下。选择包含 MEASURES 维的数据的列。在 ACCOUNT 维上拖动时，确保把此维标识为“ACCOUNTS”维类型。

B.6.1.2 通用维

与第 187 页 B.5.1 节“建立 IS 模型”功能提及的与转换相关的规则(尤其是 FIND 和 REPLACE 功能)也适用于 IS 元大纲。每当您使用 TRANSFORMATION 功能时，您会把时间添加到元数据，并把数据加载到 DB2 OLAP。

并置和子串

在 IS 模型和 IS 元大纲中，查看表时，可以使用并置和子串功能。可以在“转换”按钮下使用前缀和后缀功能。

如果发现某个列本身没有意义，并因此而需要并置和子串功能使其具有意义，请考虑进行全局更改，并在抽取、转换和加载 (ETL) 过程中进行处理。在加载星型模型过程中处理此过程一次，然后每当建立和加载 DB2 OLAP 多维数据集时处理多次，会更加高效。

过滤和建立层次结构

建立层次结构时，针对特定列成员进行过滤使得 IS 特别灵活。过滤器在生成的 SQL 中创建一个“where”子句。您可以针对与特定维关联的任何列进行过滤。如果使用过滤器参数使您可以只建立层次结构，但可过滤的数据却不在您的维表中，请考虑把列添加到星型模型的维中。

B.6.2 加载数据

按照文档中详述的方法建立星型模型和层次结构，会改进数据加载性能。原因是：把主键和外键关系(星型模型的参照完整性)定义为 IS 模型联接，可在数据加载生成的 SQL 中创建一个非常简单的“WHERE”子句。其他方法，尤其是在没有星型模型参照完整性的情况下，会为数据加载创建非常低效的 SQL 脚本，因而会降低性能。

作为针对星型模型索引策略的一项建议，您应：

- 检查数据加载过程中生成的 SQL。您可以在 IS/ BIN 子目录中包含的 olapisvr.log 中找到该 SQL。
- 验证是否至少给每个“SELECT”列编写了索引。

- 进行基准测试，以确定附加索引的成本(必需的附加表空间)是否可带来适当好处(采用新索引，数据加载有多快？)。

B.6.2.1 FACT 表的粒度与数据加载性能

在数据加载过程中，IS 将聚集 MEASURES 数据。对于维字符串的每个唯一事件，把 MEASURES 加在一起。例如，如果以发票行详细级填充 FACT 表，并在月份级加载 DB2 OLAP，IS 就会把 FACT 表合计到月份。如果数据加载性能时间不可接受，请考虑创建“聚集 FACT 表”。在 DB2 中，这称为“自动汇总表”(AST)。通过在建立星型模型过程中聚集 FACT 表，您可以有效地减少 FACT 表中的列总数，以及 SUM 功能所需的 SQL 性能时间。出于穿透钻取目的，必须把详细 FACT 表联接到星型模型。

B.6.3 IS 穿透钻取功能

IS 穿透钻取功能是 IS 的主要功能。要记住一个简单的事实：

IS 穿透钻取报表使最终用户能够穿透钻取到星型模型中包含的细节中。

穿透钻取报表不会返回到源关系数据库：穿透钻取报表返回到星型模型。作为星型模型和 IS 模型组成部分的任何和所有表都可用于 IS 穿透钻取。

B.6.3.1 穿透钻取报表详细信息

在 IS 元大纲中创建穿透钻取报表时，定义两件事情：

1. 作为“WHERE”子句元素的 DB2 OLAP 交点。
2. 要返回的列或数据，即在 SELECT 语句中标识的列。

有关创建穿透钻取报表的最重要的事情是：

1. 定义 DB2 OLAP 交点时，从 FACT 表中选择叶级成员，而不是从每个维表中选择叶级成员。这会降低 SQL 脚本的复杂性，并因此而提高检索性能。

2. 定义要检索的数据列时，从最少数量的表中选择列。同样地，通过限制表的数目和减少连接数目，您将会改进检索时的性能时间。
3. 对穿透钻取报表所作的任何更改都需要您重新建立 DB2 OLAP 大纲。DB2 OLAP 大纲文件 (.otl) 包含穿透钻取所必需的信息。更改穿透钻取报表，而不重建 DB2 OLAP 大纲，就会导致最终不能更改穿透钻取报表。
4. 考虑修改星型模型索引策略，以便体现穿透钻取要求。

附录 C 案例研究：端到端方法实例

为了说明通过 OLAP 端到端方法进行工具集成，我们实施了一种基于系统缺陷应用程序的案例研究。

C.1 上下文和业务情况

全球性软件公司 SoftCompany 有许多客户交互中心 (CIC)，其可接收有关其客户提交的产品缺陷案例。CIC 遍布世界各地，共享可记录有关客户请求详细信息的公用事务系统。呼叫接收器通过 Web 或通过自动电话应用程序把客户请求注册到事务处理系统。

该公司一般在“回叫”模型上操作，其中，与客户最近的联系点通常负责收集信息，并在事务系统中创建案例记录。然后把电话信息分配给具有相应技能的技术人员以查看所提供信息，并启动与客户的对话，以便答复其请求。一旦客户请求得以满足，就把案例记录标为关闭。

该公司需要进行分类和分析：

- 公司接听电话技术支持人员的表现：
 - 谁解决的案例数量最多？
 - 一般而言，谁解决案例时需要的天数最少？
 - 如何解决案例？
- 客户呼叫：
 - 谁呼叫得最频繁？
 - 什么时候客户呼叫最频繁？
 - 请求的种类有哪些？
- 影响的产品：
 - 什么产品需要的技术支持最多？
 - 什么产品需要的技术支持最少？

OLAP 解决方案必须为该公司提供所有这些问题的答案，并能使用更复杂的分析预测其他因素。

C.2 建议方法

我们把 SoftCompany 提供的输入数据加载到 IBM DB2 通用数据库 (DB2 UDB) 上的关系中转区。我们使用 DB2 通用数据库可轻松操作、整理和转换数据,以便加载关系数据集市(请参阅 Ralph Kimball 的《数据集市工具箱》一书。星型模型中的表结构特别适用于多维数据库,而且是准备要加载到多维数据库中的结构关系数据的最好方法。

星型模型包含 FACT 表,该表中保存有业务专业领域和数字度量的关键数字。FACT 表将这些度量与不同维表相关,因为其包含了所记录事实的上下文信息,并保存有实现深入操作的聚合层次结构。

为了灵活处理要实施的 OLAP 数据库数目,并能够处理不同维,我们使用了 IBM DB2 OLAP 集成服务器 (OIS) 以便星型模型数据集市上定义逻辑 OLAP 模型,这是 OIS 的先决条件。

该 OLAP 模型包括我们使用 OIS 元大纲结合在一起的所有可能的维,以便于针对不同测试目的创建不同的 OLAP 数据集市。

我们使用 Intelligent Miner for Data 进一步了解输入数据,并查找我们的 OLAP 设计的有用维。

我们把分析器用作报表和回写的工具。针对 OIS 穿透钻取报表,我们选择了分析器虚拟多维数据集功能,因为它可在关系数据库内实现多步骤操练。

图 53 详细说明了整个体系结构,而图 54 则详细说明了硬件和软件配置。

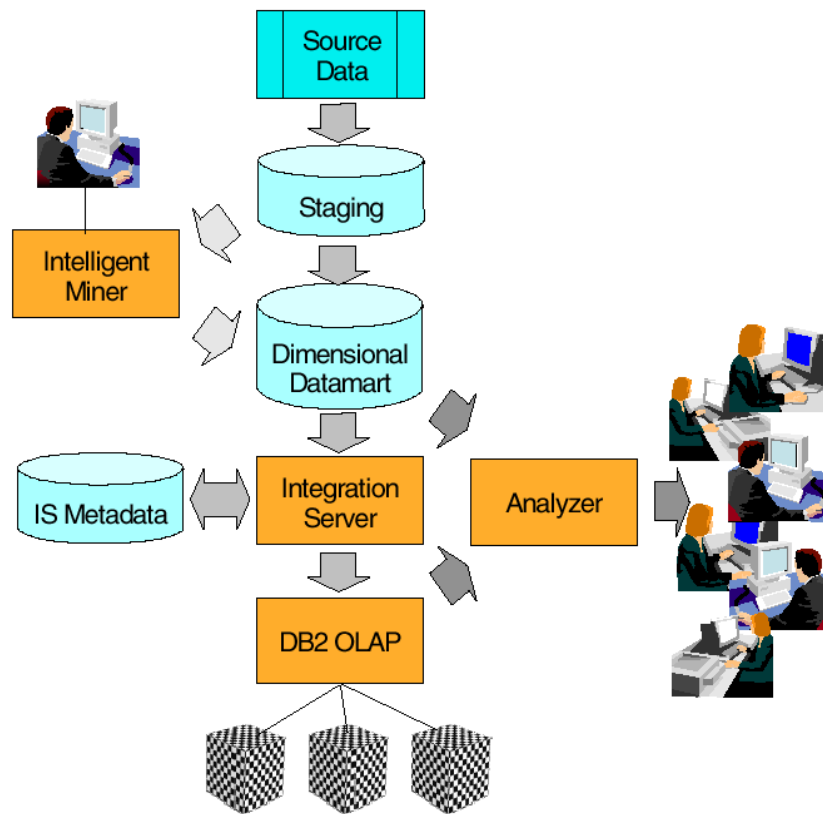


图 53. 案例研究中涉及的产品

考虑到硬件和软件体系结构，我们使用了 RS/6000 产品系列的 J50 服务器(1 GB 内存和 4 PowerPC_604 200MHz 处理器)。我们安装了 DB2 V7.1、DB2 OLAP Server V7.1、DB2 OLAP Integration Server V7.1，以及 Intelligent Miner for Data V6.1。源数据也在 RS/6000 服务器上。作为一种最终用户工具，我们使用运行在 IBM PC 365(带有 Pentium Pro 200 MHz 处理器和 256 MB 内存)上的 Hyperion Analyzer Server V5.0.1。图 54 显示了硬件配置。

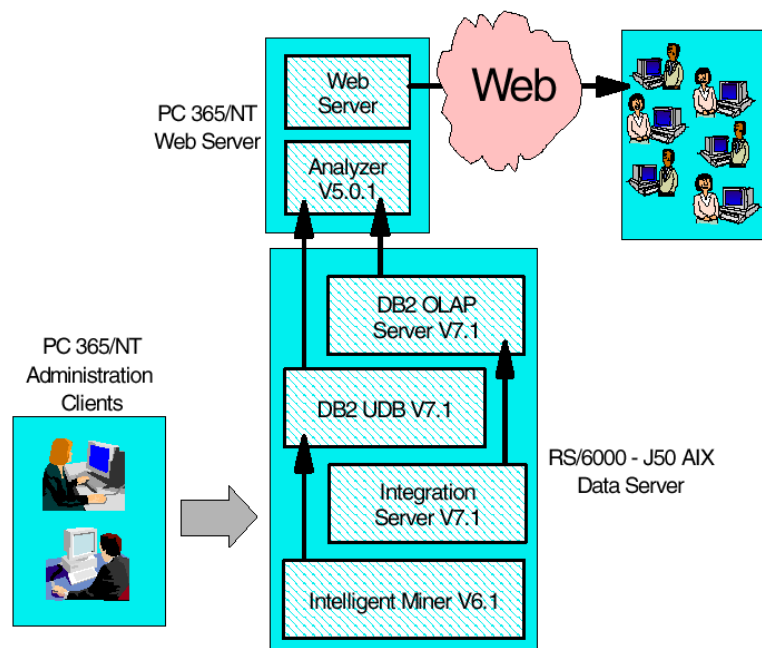


图 54. 案例研究硬软件配置

C.2.1 构建关系数据集市

为了能清洁并轻松转换输入数据，我们把源数据导入到关系中转区 AIX 上的 DB2UDB。

使用 RDBMS 功能和 SQL 实力准备用于 OLAP 数据库的输入数据，以便最好地适应 OLAP 应用程序，我们建立了星型模型模型。

在此星型模型模型上创建的关系数据集市包含 1 个 FACT 表和 14 个维表(如图 55 所示)。FACT 表包含 48,420 行。只有 0.07% 的案例有状态“新建”，4.3% 的案例有状态“打开”。其他案例都处于关闭状态。

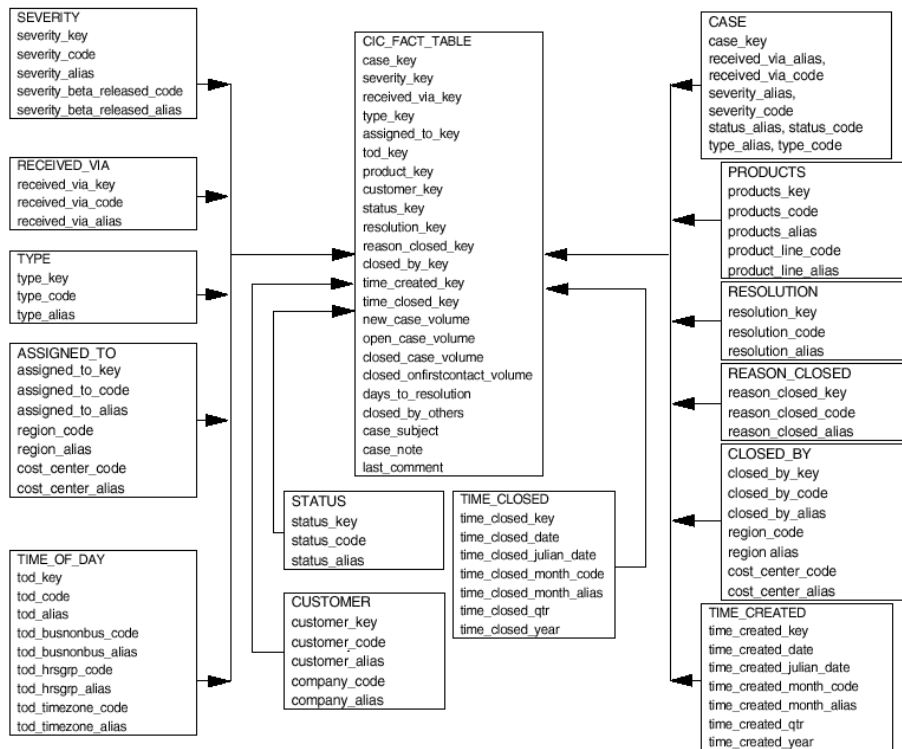


图 55. 数据集市中的星型模型

在清楚我们的 OLAP 维中什么应是维，以及什么应是度量之后，我们建立了此数据集市。

C.2.2 使用Intelligent Miner for Data计 OLAP 维

数据挖掘是从大型数据存储区发现以前未知的和最终可理解的信息的过程。

建立和开发新的 OLAP 模型时，数据挖掘是 DB2 OLAP 的补充工具：

1. 便于发现最有意义的维
2. 便于包含新的维
3. 便于定义最有用的成员、层次结构和属性

C.2.2.1 发现最有意义的维

我们使用 Intelligent Miner for Data 发现最有意义的维。OLAP 模型设计中最重要的一方面之一是要包含在特定多维数据集中的有限维集的定义。包括太多维会增加分析的难度，并会影响性能和存储。如果所选维是最相关的，并且具有高预测值，那么最终分析就会更高效。

在呼叫中心案例研究中，我们使用了双变量统计法、神经网络，以及因子分析，以便为 OLAP 报表确定最有意义的属性，并通过补充方法验证一种计数的结果。

通过组合这些方法，我们发现应把八个最相关的属性作为“days to resolution”的维包含在内：

- Type
- ClosedBy
- Time_Created
- AssignedTo
- Severity
- Closed_by
- ReceivedVia
- Days_to_resolution

在任何情况下，我们都应确保这些分析中不包含冗余属性。有时，如果我们怀疑可能存在冗余，就值得在候选字段之间执行特定相关测试，以便消除多余的信息。

C.2.2.2 包括新维

要把新维融入我们的 OLAP 模型中，可以考虑以下数据挖掘任务：

- 分群
- 分类
- 预测

在案例研究中，我们使用了分群，以便在我们的人口中找到有意义的案例数据细分。我怎么发现了案例集的八个不同数据细分，并通过智能挖掘器可视功能观察其不同侧面。例如，发现的特定数据细分与临界严重性远远低于平均值的案例相对应。

我们把结果作为一个新的维包含在我们的 OLAP 大纲中，以便在分析中考虑这些新的数据细分。

就我们的案例研究而言，属于同样数据细分的案例代表在某个方面相似的案例。把这一点考虑在内，OLAP 用户现在可以按这些数据细分过滤信息、观察其特征，以及查明其中一个或多个数据细分是否对未来分析有意义。在图 56 中，我们可以看到八个数据细分的列表，神经元分群挖掘的全局结果。

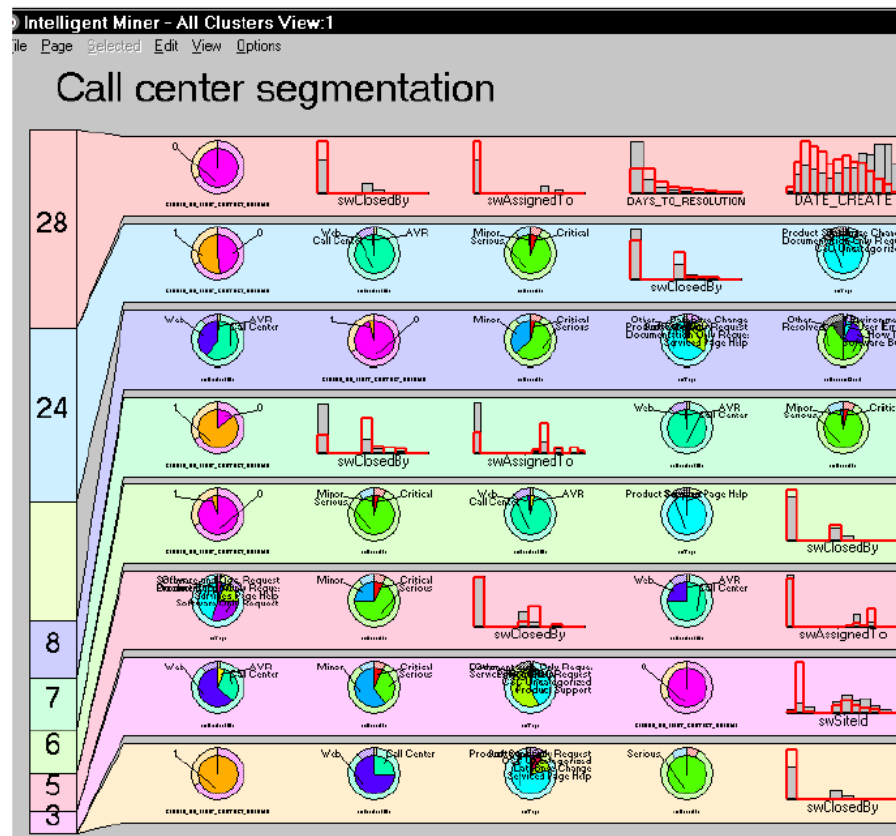


图 56. 用于案例研究的分群可视化

在图 57 中，我们可以看到有关发现的群集统计信息。群集 5 对应 7.3% 的案例人口。我们注意到在此群集中，解决案例的平均时间少于在整个人口中的时间。

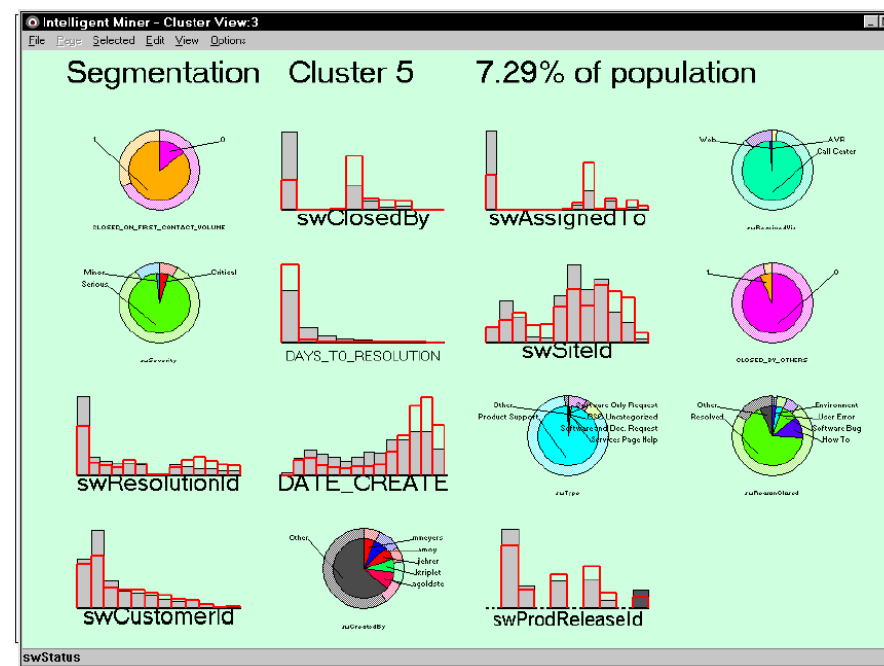


图 57. 数据细分示例

C.2.3 使用 OIS 构建 OLAP 数据库

本节中，我们将描述构建 OLAP 数据库时如何使用 OIS。

C.2.3.1 定义模型

在所有维建立方面的可用技术中，使用 OIS 是最复杂和灵活的。前提条件是星型模型数据集相连接。

图 58 显示我们根据案例研究建立的 CIC OLAP 模型。

我们尽量使事情变得简单一些。我们使用单表，并避免连接表，以及操作和转换数据(就像处理 SQL 接口一样)，因为这会减缓维建立过程。

我们确保用于维建立数据源的关系表尽量便于使用。注意：如果没有适当的表，您可以请数据库管理员为您创建一个。如果不可能的话，那么您的第二个最佳选项是使用 OIS 转换选项。

与使用 SQL 接口访问关系表中存储的源数据一样，与 RDBMS 的连接不是本机方式，而是通过 ODBC。

通过 SQL 接口建立维时使用 OIS 的优势在于其能使您重复利用现有 OLAP 模型和元大纲，从而消除了编写和维护多个 SQL 语句的必要。

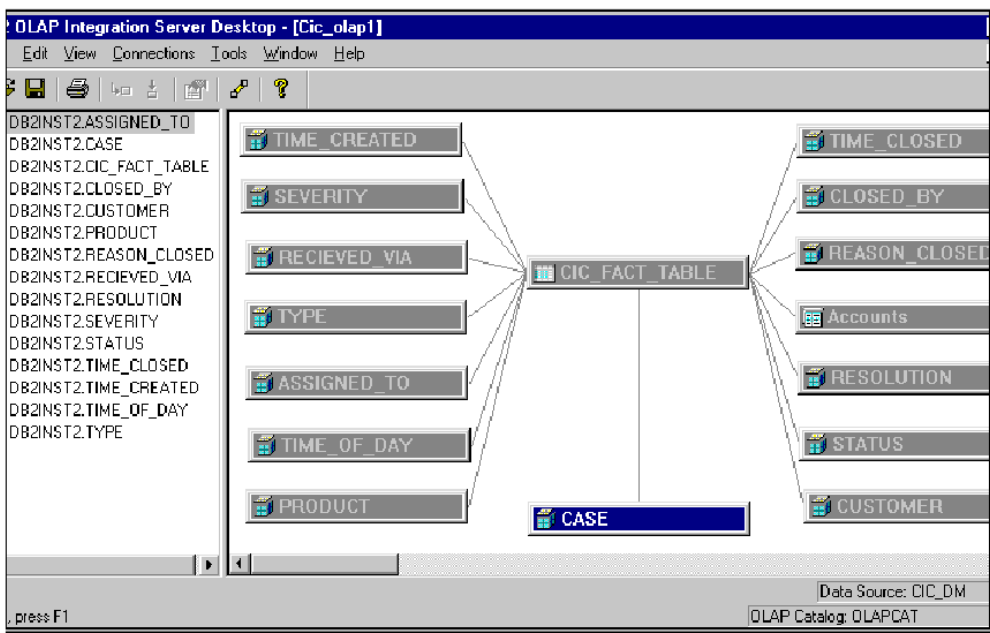


图 58. 案例研究中的 OLAP 模型

C.2.3.2 定义多个元大纲

在我们的案例研究中，我们创建了多个元大纲，以便根据同一 OLAP 模型生成 OLAP 多维数据集。

下面是我们所执行的操作：

- 对于技术支持管理而言，为了衡量其技术人员的业绩，我们使用了智能挖掘器提供的大纲(注重 `days_to_resolution`)，或使用图 59 中显示的“CIC 工作量细节元大纲”。
- 对于销售人员，为了在有客户访问之前检查用于客户、客户电话，以及其请求的所有打开和关闭案例，我们使用了图 60 中显示的“CIC 客户元大纲”。

我们使用了把维 `RECEIVED_VIA`、`SEVERITY` 和 `STATUS` 用作属性维。属性维需要有其基本维，为此，我们包含了 `CASE` 维。状态、严重性和类型均是每个具体案例的属性，因此我们必须把 `CASE` 作为一个维包含在内。

在稀疏维之后，我们根据与基本维“案例”的关联性，对“元大纲中的属性维“接收方式”、“严重性”和状态进行了排序。

为了能够获得有关客户的详细信息，并找到解决案例的方法，我们通过所链接的分析器创建了两个虚拟多维数据集。(参见第 211 页上的 C.2.4: “将分析器用作报表工具”)。

CIC workload detail			Type/Consolidat...	Alias/Member Properties
CIC workload detail				
Measures		Accounts		(Label Only)
New case volume		(*)		
Open case volume		(*)		
Closed case volume		(*)		
Closed on first contact volume		(*)		
Days to resolution		(*)		
Time created		Time		(Dynamic Calc)
TIME_CREATED.TIME_CREATED_YEAR		(+)		(Dynamic Calc)
TIME_CREATED.TIME_CREATED_QTR		(+)		(Dynamic Calc)
TIME_CREATED.TIME_CREATED_MONTH_CODE		(+)		(Dynamic Calc)
TIME_CREATED.TIME_CREATED_DATE		(+)		(Dynamic Calc)
Status				(Sparse)
STATUS.STATUS_CODE		(+)		(Alias: STATUS_ALIAS)
Severity				(Sparse)
SEVERITY.SEVERITY_BETA_RELEASED_CODE		(+)		(Alias: SEVERITY_BETA_RELEASED_ALIAS)
SEVERITY.SEVERITY_CODE		(+)		(Alias: SEVERITY_ALIAS)
Product				(Sparse)
PRODUCT.PRODUCT_LINE_CODE		(+)		(Alias: PRODUCT_LINE_ALIAS)
PRODUCT.PRODUCT_CODE		(+)		(Alias: PRODUCT_ALIAS)
Assigned to				(Sparse)
ASSIGNED_TO.REGION_CODE		(+)		(Alias: REGION_ALIAS)
ASSIGNED_TO.COST_CENTER_CODE		(+)		(Alias: COST_CENTER_ALIAS)
ASSIGNED_TO.ASSIGNED_TO_CODE		(+)		(Alias: ASSIGNED_TO_ALIAS)
Data Source: CIC_DM				
OLAP Catalog: OLAPCAT				

图 59. OLAP 元大纲 — 工作量细节

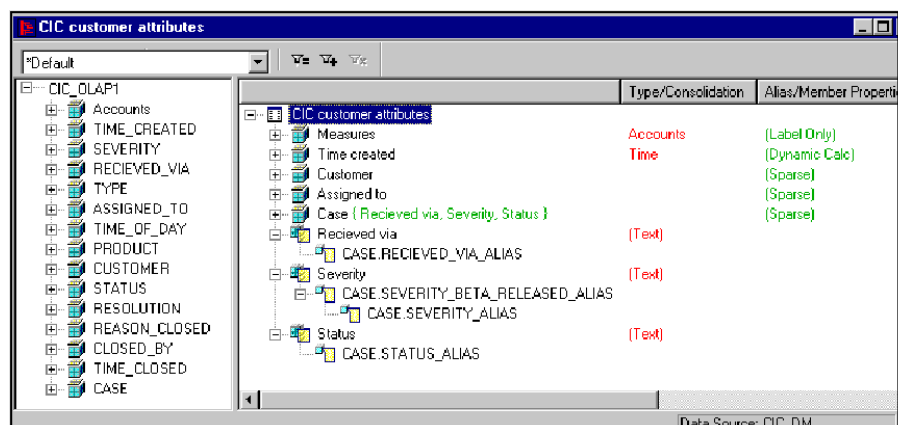


图 60. OLAP 元大纲 — 具有属性维的 CIC 客户

C.2.4 将分析器用作报表工具

要从 DB2 OLAP Server 创建志案报表和图表，我们使用了分析器，并测试了穿透钻取功能以访问关系数据和多维数据。

利用分析器，我们可以使用两种穿透钻取功能以便浏览关系数据：

- 我们可以建立可访问 CIC 客户 OLAP 数据库中数据的视图，以便了解解决天数以及根据产品关闭的案例数目(参见图 61)。

根据这些结果，我们可以访问关系列 CASE_SUBJECT、CASE_NOTE、LAST_COMMENT，使用 OIS 穿透钻取功能获得有关案例主题的详细信息。为此，在 OIS 中建立模型时，我们需要定义穿透钻取报表。

OIS 自动生成 SQL，并通过分析器和电子表格中的 LRO(链接报表对象)使用报表。我们不能链接多重穿透钻取。为此，我们在 CIC_fact_table 中包含了另外 3 个列，以便详细说明关闭和解决案例的原因(CASE_SUBJECT、CASE_NOTE、LAST_COMMENT)。

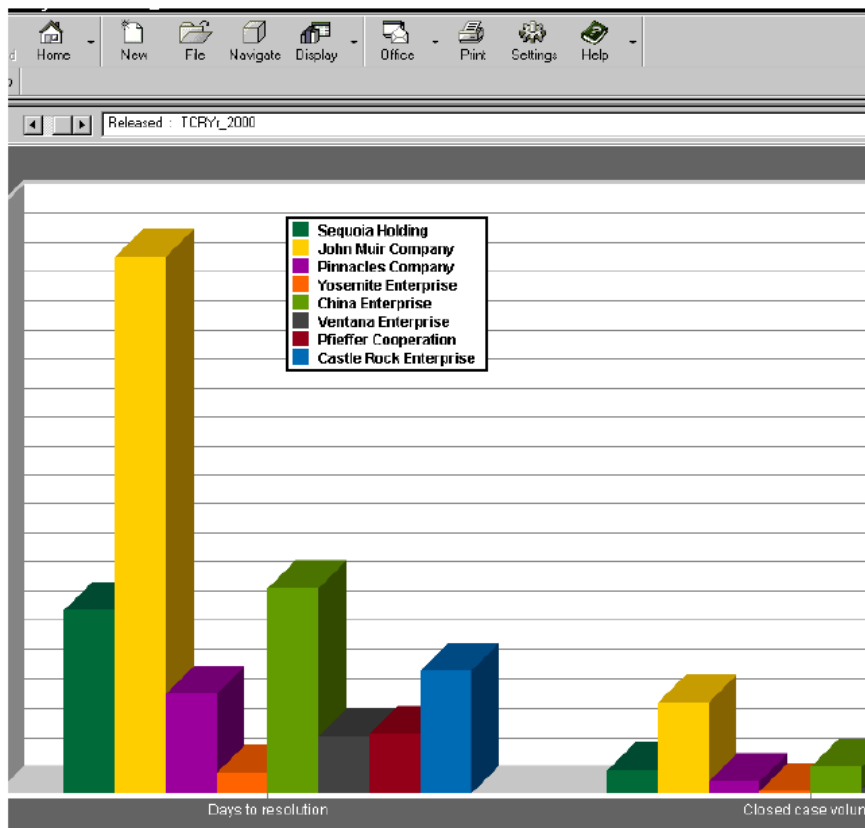


图 61. 2000 年根据客户以及根据每个严重性所定的性能

我们可以使用分析器直接提供的虚拟多维数据集。虚拟多维数据集是一个 SQL 选择语句，其向分析器返回数据，从而使用户能够以多维格式显示关系数据。定义虚拟多维数据集之前，需要预定义一个视图，以便转换、联接、选择，以及聚集所需数据，使定义虚拟多维数据集的 SQL 成为一个简单的选择。

每个虚拟多维数据集都应直接按照单个视图操作。通过将视图作为“自动一览表”进行举例说明，可以改进性能。通过链接视图，用户可以透明地在各种关系和多维数据源之间切换。从另一个视图浏览到一个虚拟多维数据集视图(钻孔链接)时，可以作为动态生成 SQL “where” 子句内的限制回避过滤器和页面成员选择。仅检索相关记录。

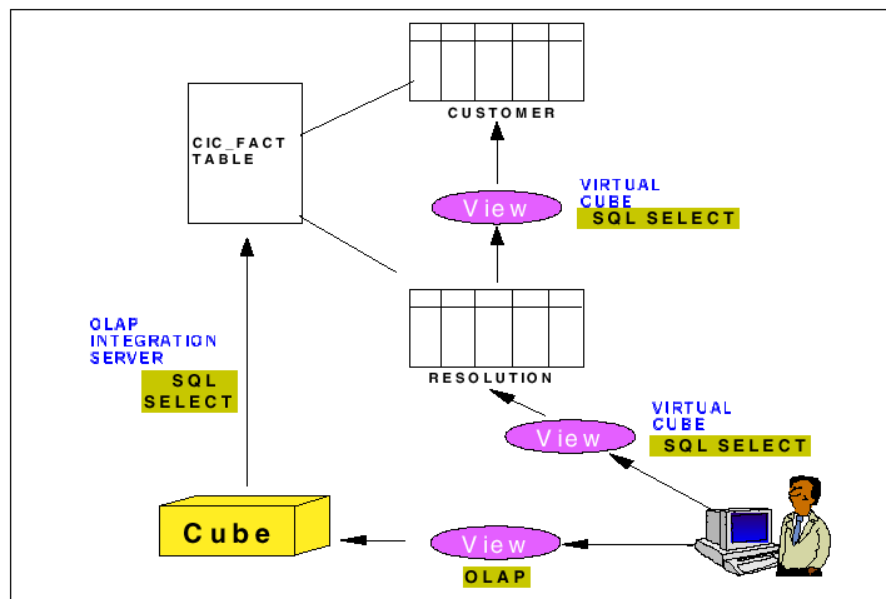


图 62. 穿透钻取功能

1. 请先取得连接。
 - 把源关系数据集市作为系统数据源注册到 ODBC。
2. 使用分析器管理员为第一个虚拟多维数据集创建一个模型 (RESOLUTION)

- 从工具栏上单击“关系”，并插入新模型。“模型名称”识别在分析器客户机工具中使用的虚拟多维数据集创建视图。BDE“别名”识别关系数据库。
 - 键入 SQL 查询。
 - 如果 SQL 查询需要找到查询表中键的说明，首先定义查询表及其 SQL 查询，并测试它们。(ex: select type_key,type_alias from type)
 - 如果该列将要作为多维数据集中的维或度量，则可为在 SQL 请求中选定的每个列进行定义。
 - 如果我们想要在分析器视图中看到作为行或列标题的列项目，那么我们就把列定义为维说明。另外，如果该列是一个键列，而且需要其说明，我们就定义一个查询表(以及从查询表中选择代码和说明的 SQL 选择语句)，而且我们选的是维代码，而非维说明。
 - 如果我们希望看到分析器视图的数据单元中的列项目，那么我们把该列定义为一个度量(例如，把 RESOLUTION_ALIAS 定义为度量)。我们需要至少把一个列定义为维，一个列定义为度量。
3. 测试模型。
- 单击“测试”。
4. 使模型同步。
- 此步骤在“分析器管理员”中的数据库列表中添加虚拟多维数据集。
5. 管理用户对多维数据集的访问。
- 在“分析器管理员”中单击工具栏上的“管理”，并就像处理正常 OLAP 数据库一样建立用户访问。
6. 为该虚拟多维数据集创建视图(参见图 63)。

3 of 3		Released Critical
		resolution detail
John Muir Company	LERENE	Explained upgrade procedure/discussed upgrade concerns/walked client through upgrading file(s).
	LERETI	Client will rebuild the application.
	LONOFI	Error when accessing Retrieve Lotus- app.ini file must specify app_data_42 for this release.
	LETEJA	Manually entering External accounts out of sequence in Ranges or Exceptions causes accounts to be lo
	LUNEDA	Resolved
	LUZEGD	Client will restore the application prior to problem.
	LASESO	General Resolution
	LASIFE	Client resolved issue themselves
	LUATI	Client will rebuild data category.
	LUNUJE	Software Bug submitted with acceptable workaround
	LUSELA	WFO Installation - General Questions or Problems
	LAVAKE	Consultant helped resolving the issue.
	LASODE	this is only occuring on one PC client all other do not exhibit this behavior. The suggestion at th
	LUKIZA	Consolidating - Error reading header for PSF file in DF_PSFRELOAD
	LUKUWI	Run chart logic
	LUOGA	restore from backup
	LUSUWI	Issue Resolved
	LETERA	Problem resolved by hardware upgrade
	LETUSO	Shipping Request
	LUMACR	Set user limit
	LUKJRA	Shipping Request
	LONETE	Referred to consulting
	LUSEKU	User Error
	LUZYI	Client wanted to know how to make a copy of their application and test it in 4.3.1.
all_resol		
Virtcube		

图 63. 查看 RESOLUTION 虚拟多维数据集

- 我们也可以使用分析器管理员为第二个虚拟多维数据集 (CUSTOMER) 及其关联视图创建另一个模型，以获得有关客户 LERENE、LERETI、LONOFI 等的详细信息。
- 创建从 OLAP 多维数据集到虚拟多维数据集的链接(参见图 64)，以及虚拟多维数据集之间的链接。
- 当我们单击 OLAP 多维数据集上的单元时，分析器自动跳到本身可以链接到另一个视图等的链接视图。

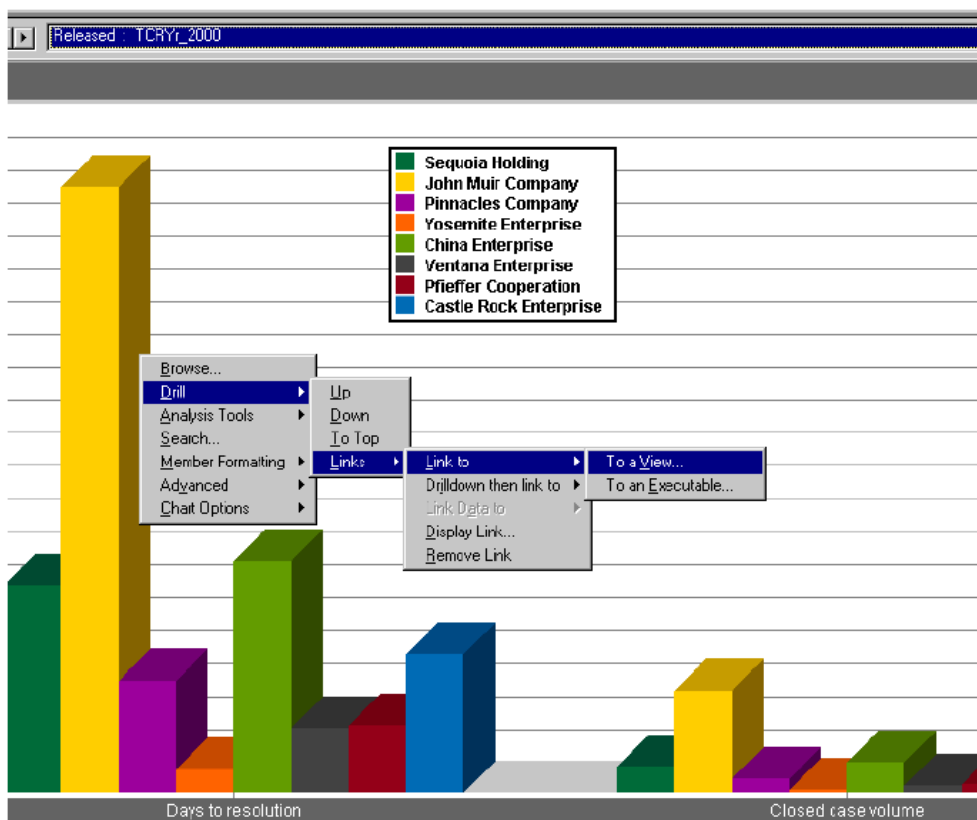


图 64. 创建视图间的链接

注意：为了限制 SQL 请求的数目并改进性能，我们必须使用“自动一览表”。

C.3 摘要

通过此案例研究，我们尝试了演示建立 OLAP 数据库的实用方法 — 集成多种工具，例如，DB2 OLAP、Intelligent Miner for Data、OIS，以及分析器。

附录 D 获得最佳OLAP交付方式的注意事项

与过去的静态预定义报表相比，现在的报表要求已有了很大的发展。现在用户需要：

- 更大的灵活性，以便能根据自身业务需要而非出于报表目的进行交互、导航，以及分析数据。
- 访问信息更轻松，从而能够专注于业务问题。
- 具有更快的决策能力。
- 更广泛的部署，从而迎合企业内外广大用户群体的需要。

D.1 OLAP 报表工具中需要哪些属性？

定义属性一览表时，我们需要考虑计划最佳 OLAP 交付方式。报表工具应满足以下要求：

1. 高级报表功能：

交互/分析	不仅仅具有一组有限的可用视图，用户还应能够操作视图和数据，以进行更多分析。他们应能够根据检索的数据定义公式，以及根据数据准则格式化报表。
可用性	对于多数机构极为重要的问题是：创建一个视图要花费多少精力？通常，创建视图的用户并不是程序员，而是具有极高技术天资的分析师。如果他们必须学习编写 Java 或一些其他脚本语言，那么产品可能并非最佳选择。
回写功能	一种可以修改服务器上数据的工具具有回写功能。模拟需要回写功能。
构建与建立	我想要一种“现成的”解决方案(易于实施，绝大部分是点击操作)，还是想让我的一名程序员专门负责该工具，能够使用脚本和 Java 语言，以获得更大灵活性呢？

多源

是否可以把工具绑定到多个数据源以生成视图？需要什么水平的多源功能？工具是否可以从 OLAP 系统穿透钻取到事务处理系统？首先，OLAP 系统必须支持此功能，然后前端必须把此功能传递给用户。

2. 应用功能：

零安装

零安装客户机是完全驻留在服务器上的客户机。这种类型的工具通常是支持 Web 的工具，其不需要在客户机计算机上设置。它不需要进行太多开发工作，因此对许多企业来说是很好的解决方案。用户是否可以执行通过桌面浏览器所需的所有功能？如果可以，那么这对他们来说便是一个很好的工具。

客户机类型

Windows/Java/HTML/Dynamic HTML。这些是表现模式和客户机类型。表现模式是显示数据的方式。有些工具允许您创建可通过多种此类方式表现的视图。例如，一种工具允许您根据需要的交互性级别创建 Java、HTML 和 Dynamic HTML 视图。另一种工具具有 Windows 客户机，您可以用它来建立所有视图并将其用作客户机，同时还能够把视图发布到 Web 上。

3. 成本问题：

没有成本

最后一个需要注意的重要因素是成本，以及如果该工具满足所有上述要求，但不符合预算，需要做出哪些牺牲。

D.2 DB2 OLAP Server 分析器怎么样？

DB2 OLAP Server 分析器是基于 Hyperion Analyzer 5.0 的 DB2 OLAP Server 7.1 的一个新增功能。

分析器是完全按照多维概念构建的，与根据早期关系体系架构设计的传统报表工具完全不同。因此，像层次结构、[嵌套维](#)等概念，对于产品界面都是自然而直观的。

分析器在报表工具领域的主要价值是：

- 交互和分析功能
- 可应用到多种用户的方法
- 使用像 API 工具箱这样的开放式主流开发工具的可扩展性
- 分析器利用和影响 DB2 OLAP 能力的方式

D.2.1 交互和分析功能

分析器为最终用户(根据其安全配置文件)提供了动态生成的报表，其中包含一组或多组视图。每个视图均为一个交互式、易于创建且支持强大分析功能的报表，其分析功能使最终用户可以执行数据分析并可充分利用 DB2 OLAP Server 数据库的优势。

分析器支持多种视图类型：

- 电子表格：数据表格或网格表示方式。
- 图表：条形、线条、立方形、面积图，以及其他可提供数据图形表示法的图表。
- 窗体：窗体是一种复合视图，可使用用户在屏幕上同时看到多种分析器显示，以及其他第三方内容(网页、OLE 对象等)。另外，像(内部应用程序或外部应用程序的)列表框、单选控件，以及按钮这样的图形组件使用户可以创建易于使用的应用程序，这些应用程序指导用户并使其能够研究其数据或启动备用报表。窗体尽管非常复杂，但创建时不需要编码。可以创建具有直观的 EIS(高级管理人员信息系统)界面的窗体，此界面非常有助于用户使用应用程序。
- 插接板：插接板可以是任何有助于您显示数据的图形对象。例如，计算机芯片制造商使用分析器跟踪库存。芯片图片被用作“插接板”，当库存很少时芯片的某些区域会显示为红色。

用户可以直接根据报表布局进行交互，也可以使用“多维数据集导航器”界面创建报表。

用户交互包括分析所需的各种活动：组合和添加维的能力；在图表、电子表格和插接板上(向上、向下、向底部，以及向下一个)搜索的能力，以及在电子表格上旋转数据的能力。此外，用户还可通过显示计算(可以使用多种计算功能，其中包括基本数学运算、最小值/最大值、百分比、分类、特定计算)以及分析器的交通灯(以便比较它们之间的成员)，把值添加到数据。

交互性和现场数据检索使用户可以查明问题，或进一步确定问题的细节。

分析器可用不到一秒钟的响应时间从 DB2 OLAP Server 中检索联机数据。通过深入到低级数据、以多种方式交换到显示数据，用户可以查明趋势，并获得较深的商业智能。联机数据可确保所有用户均可使用同一数据真值。

创建视图或窗体可由最终用户来完成，无需编程或 IT 部门的参与。通过向用户授权，可以更快地进行决策，因为将“报表要求”传达到另一个部门非常耗时，而且经常是交互式延时过程。用户仍然可以使用易于使用的图形环境。

同样地，用户创建的报表可以让所选择的用户组进行共享，也可以专供自己使用，还可以让整个企业共享。

D.2.2 应用功能

由于是专门为广泛企业应用而精心设计，分析器可满足不同用户类型的需要。其适用于企业的各种层次——从需要高级管理人员信息系统 (EIS) 和关键性能指标报表的高级管理人员，到希望在其电子表格中继续进行其分析的高级用户。

包装的 Windows、Java 和 HTML 客户机的直接体验使得安装分析器非常容易。

D.2.2.1 Web 策略

企业可根据其 Web 策略进行部分应用分析器。除了 Windows 客户机之外，还可以使用强健的 Java 客户机和 HTML 瘦客户机。

此外，将报表数据发送到 HTML 以进行静态分析，以及格式定制的功能也非常吸引人。

导出到 HTML 时，产品中包含多个示例模板，因而可轻松地应用到 HTML，而无需深入了解其语法。但是，如果 Web 主管希望定制输出，分析器模板就会使用开放的 HTML 策略。因此，Web 主管就可以自由地完全定制格式、外观感受，以及第三方扩展(Java 小程序、ActiveX 控件、JDBC 对 IBM DB2 UDB 的调用或其他关系数据存储器)。

在您的网页上可以非常灵活地创建分析器内容，其中包括“单元收集”，因而使用户可以创建格式丰富的输出(句子、自动通知删除交通灯问题的脚本等)。

Java 客户机

除了 Windows 客户机之外，还可以在企业应用中使用一种 Java 客户机。该客户机无需安装软件，并且将您的 Web 浏览器用作其界面。与 Windows 客户机一样，可利用便于方便快捷地进行报表共享和个性化的公用存储库。Java 客户机是交互式的，它不仅可提供报表交互性，而且拥有标准报表创建和数据编辑功能。只有 Windows 客户机可进行高级报表创建(例如，复杂选择)和窗体创建。

Java 客户机可支持视图阅读和标准报表创建功能。支持电子表格、图表、插接板以及窗体。此外，还可轻松从界面上启动 Excel 电子表格对象和网页。同样，如果操作系统可查找程序，就可支持从窗体到应用程序的任何链接。

就像 Windows 客户机一样，显示是交互式的，而且易于使用。视图也是成组的，并可自动更新为主页。

由于视图存储在公用存储库(如 DB2 通用数据库)中，您从 Java 客户机创建或修改的任何新报表都可立即为其他 Web 或 Window 用户所用。保存报表功能可以把报表发布到其他 Window 和 Web 用户。

动态 HTML 客户机

除了 Windows 客户机和 Java 客户机之外，企业应用中还可以使用瘦的动态 HTML 客户机。该客户机不需要安装软件，且只使用 Web 浏览器作为界面。与 Windows 客户机和 Java 客户机一样，可利用方便快捷地进行报表共享和个性化的公用存储库。Java 客户机是交互式的，它不仅具有报表交互性，而且具有数据便捷功能。

此版本仅支持电子表格和图表。由于速度需要方面的原因，格式编排稍有不同。凭借 HTML 客户机，我们可以保持“超瘦”，并可减少网络/Web 流量。如果用户没有更高速的调制解调器，他们可以通过选择“切换样式”按钮栏来选择使用其他格式编排方式。

显示是交互式的，并可支持钻孔和显示更改、分页，以及通过“信息面板”进行额外的维旋转和移动。

通过保存视图，使视图可用于 Windows 和 Java 客户机。

HTML 客户机还允许输入数据。由于 HTML 客户机以 HTML 模板为基础，因而管理员可以更改应用程序的外观及感受，以便使其符合公司标准，并根据需要删除功能(例如，编辑)。

D.2.2.2 发布和分发报表

当创建视图后，Windows 客户机、Java 客户机，以及 HTML 客户机界面就立即可以使用该视图，而无需任何管理工作，可注册到集中的分析器存储库。

把当前视图或信息从该视图发送到 Excel、123、剪贴板、文本文件或电子邮件或 Lotus Notes 中附加视图参考的能力，使用户可以把信息分发给其他应用程序。

使用 Excel 时，分析器具有一种使用户可将 Excel 工作簿保存到分析器存储库的功能，从而可以进行集中的安全共享和分发。

通常情况下，用户希望阅读静态信息或喜好格式丰富的显示。通过把分析器视图导出到 HTML，两种希望都可以实现。另外，用户还可以原路返回(超级链接)到 Java 客户机或 HTML 客户机，以便在需要时进行实时数据和连续分析。通过这种往返功能，用户可以获得静态分发(即，客户请求报表时，不需要花费时间进行处理)的优势，同时需要在需要额外分析时，仍然可以方便地跳转到一种有效交互模式上。

D.2.2.3 维护 Web 上的报表

维中的成员选择可静态进行(直接选择“东”或“西”)，也可动态进行(“市场”维的子级)。通过利用动态选择，在您把成员添加到 DB2 OLAP Server(例如，引入新的产品线或市场)时，维护分析器视图的成本极低，因为无需进行任何维护工作——报表自动进行更新！这使得企业可应用的策略更加强大。

D.2.3 API 工具箱

除了 Web 策略之外，开发人员还可使用分析器 API 工具箱扩展分析器功能，该工具箱允许使用开放式主流开发工具快速汇编基于 Web 的定制商业分析应用程序。

这使开发人员可定制前端界面，并可分析器视图和技术集成到现有的公司 Web 策略和门户网站上。

有些用户可能只需要定制一两种显示方式以满足其个别要求。例如，也许他们需要一种特定的数据输入屏幕，而非已应用好的分析器“编辑数据”对话框。也许某个图表类型需要某种特种定制(例如，双曲线树)。

其他用户可能希望把分析器视图、基础设施，以及对话框无缝集成到现有网页或门户应用程序中。

让我们来演练一些定制操作，它们与您对 Web 及相关技术(例如，Java 脚本、插件和 JDBC)的了解相关。此种可能性数不胜数，无法一一说明。

D.2.3.1 基本网格组件

分析器中包含了一系列组件(网格、图表、公用对话框，以及函数调用)，其可使开发人员汇编基于 Web 的定制分析应用程序。

开发人员可以把分析器网格包含在内，并可参考分析器存储库中的现有视图。

可以轻松将此页面集成到包含其他门户网站或网页内容的网页中。

由于开发人员把分析器网格组件包含在内，就可以继承许多网格控件(右键单击鼠标菜单、交换、向下搜索检测，等等)，从而可节约开发人员的许多宝贵时间。如果开发人员希望定制网格，则可以使用第三方网格控件。

D.2.3.2 图表组件

至于图表，开发人员可能已经汇编了分析器图表组件，并通过若干函数调用将其捆绑到一系列 HTML 控件。

这说明可以将您自己的菜单项添加到分析器“单击鼠标右键菜单”系统。

开发人员可以插入自己的第三方控件，以满足定制图表要求。

D.2.3.3 多个小程序

可同时在同一个网页上嵌入多个小程序，每个小程序指向一个不同视图，并可组合第三方内容。

分析器可提供一个现成的“数据输入”对话框。例如，假定用户想要此对话框的一个变体，并想要把其他业务逻辑或某个外观及感受界面包含在内。通过分析器 API 工具箱，就可以达到这一目的。

一旦创建某个 Web 应用程序(或任何其他网页)，就可以把内容布置到分析器窗体上，并可无缝地在 Windows 客户机界面内重新显示。

D.2.3.4 通过 API 的关系穿透钻取

可以把分析器扩展为引用关系数据。

分析器 Windows、Java 和 HTML 客户机也包含一个强大的关系集成策略。分析器虚拟多维数据集具有一种把关系数据直接显示到分析器显示屏上的机制。分析器还可充分利用 DB2 OLAP 集成服务器及其关系穿透钻取功能。

可以添加一个鼠标右键单击菜单以调用 Java 脚本，而 Java 脚本运行一个 SQL 查询以便从关系数据库中获得某个客户的详细信息(电话号码、地址等)。

D.2.3.5 实例摘要

这些实例展示了开发人员是如何利用分析器工具箱扩展分析器功能的。

我们已经看到一些定制的简单实例，例如，访问关系数据或把数据与门户型应用程序中的第三方内容合并的附加定制操作。

利用分析器 API 工具箱，可以汇编组件，或进行定制创建，并始终使用用于以下多个方面的现成基础结构：视图存储、安全性、防火墙、已经部署到其他已开发的前端，或所提供的 Windows、Java 和 HTML 客户机。

D.2.4 利用 DB2 OLAP

分析器是完全利用多维概念构建的，其中包括基本父/子层次结构关系，以及有关级别、世代、替换变量、动态时间和属性的高级 DB2 OLAP Server 知识。

因此，像层次结构、嵌套维等概念，对于产品界面来说非常自然，从而使得产品界面极易使用、直观，并拥有高性能。

使用服务器可提供更好的性能，并可减少任何组织机构的网络流量。例如，当分析器需要创建“高和低产品”视图时，服务器便会执行分析，以读取大量数据，并只返回每季度的前三条记录及后三条记录。分析器以不到一秒钟的响应时间从 DB2 OLAP Server 进行数据检索。

分析器可充分利用 Hyperion Essbase OLAP Server 和 DB2 OLAP Server 的强大功能，同时还可提供对关系数据的穿透钻取功能。

分析器支持两种穿透钻取关系细节数据的现成方法：

- 分析器虚拟多维数据集(在 Windows 客户机和 JAVA 客户机中)。
- DB2 OLAP 集成服务器穿透钻取报表。(此外，利用 Hyperion 分析器 API 工具箱还可实现其他关系技术集成和定制。请参见 224 页第 D.2.3.4 节上的“通过 API 的关系穿透钻取”。)

此外，可应用 DB2 OLAP Server 安全机制来确保用户是否只能更改可安全编辑的数据。从我们的界面上，用户可选择运行某个计算脚本以更新整个模型。

附录 E Web日志合并

在传统的商业界中，商家只能跟踪可获得的信息。对于有关潜在客户的信息，通常采取客户调查、市场研究、焦点调查和产品测试的形式。要收集关于客户的信息，商家必须跟踪和分析销售记录、客户评论、咨询台电话、客户反馈、产品退货等渠道产生的数据。

现在，电子商务使这一切发生了很大的变化。因特网展示了商业历史上前所未有的环境，在此环境中商家可以跟踪、分析活动，以及决策现有客户和潜在客户行动。

本附录将解释如何将 Web 日志作为 DB2 OLAP 分析来源的组成部分，以便提供更有价值的信息。

E.1 网站分析套件

网站分析使用一套商业分析应用程序，它们可用来帮助各种机构了解、管理和优化访问者在其网站的体验。通过分析和了解访问者行为，机构不仅可以更有效地响应客户请求，而且可以在从现有客户那里获得更多业务以及从新客户那里获得新业务的工作中更具主动性。因此网站分析是组织机构客户关系管理 (CRM) 和数据仓库策略不可缺少的组成部分。

Web 日志提供可用来识别网站访问者(或客户)、访问者在访问(会话)期间的活动，以及访问者如何转到这个网站的信息。分析客户行为以及公司网站的有效性时，这样的信息非常重要。

E.2 日志文件概述

使用网站访问者信息进行报表和分析之前，必须把日志文件中的原始数据转换为可用的相关信息。这是一项非常艰巨的任务，因为日志文件可能非常庞大，而且阅读起来非常困难。

- Web 日志文件捕获每个单独的访问者交互和服务器响应。繁忙的网站的 Web 日志可能会在极短的时间内迅速扩大。必须以可以识别和跟踪访问者活动的方法定义过程。识别唯一访问者听起来容易做起来难。Web 日志文件本身并不识别唯一访问者。要检索此信息，必须对日志文件进行解释。实施将访问者活动汇总成各个会话来识别唯一访问者的方法是网站分析的关键。
- Web 日志不维护访问者会话。唯一 IP 地址(访问者操作)和网站之间的信息交换在点击流内发生时被输入 Web 日志。此外，Web 服务器必须能够同时服务于多个网页，因为从要求上讲，Web 服务器必须能够响应数千个并行访问者。由于日志文件只能记录发生时的点击流，所以每个唯一访问者的操作被分散地记录在整个 Web 日志中。因此必须设计一种将操作和行为与每个访问者关联起来的策略。

E.3 作为源数据的 Web 日志

本节讲述如何把 Web 日志用作 DB2 OLAP 的源数据。

E.3.1 Web 日志字段

重复一下，Web 日志提供原始数据项目，这些原始数据项目必须经过解释之后才能创建各个访问会话。为了提供有效的分析数据，需要对 Web 日志中捕获的信息进行规划，并由组织机构自由地进行定义。此功能当然是由 Web Server 软件提供的。一旦捕获到此点击流数据，就可以将其从这些原始数据元素转换为可汇总的信息，然后把汇总信息迁移到 OLAP 模型中，这样就可以提供一套完整的报表和分析。

可以将多数 Web 服务器日志配置为包含(但不局限于)数据项目，例如，DATE、TIME、CLIENT IP ADDRESS、BYTES RECEIVED、BYTES SENT、COOKIE、HTTP STATUS、METHOD、REFERRER、SERVER NAME、TIME TAKEN、URI QUERY、URI STEM、VISITOR AGENT 以及 VISITOR NAME。所有这些数据在确定网站有效性中具有潜在的价值。一旦将 Web 日志数据转换为可操作的数据，就可使用这些数据确定用户访问是否成功(是否发生购买交易)，或用来提高网站的效率。

E.3.2 访问者和会话识别

由于 Web 日志中包含的访问者操作不是有结合力的格式，因此使用一种称为会话化的过程来从该日志中派生有意义的信息。会话化算法实际上是通过过滤数据来收集有关访问者会话的准确信息。会话化算法的目标是将用于会话化和汇总数据的独特属性汇集到一起。

E.3.3 会话化方法

有多种识别访问者的方法。有些方法比其他方法更准确一些，而且所有方法都与 Web 日志中捕获的信息的类型相关。

E.3.3.1 只 IP 会话化

只 IP 地址会话化是计算唯一 IP 地址数目的方法。此方法不太准确，因为许多用户可通过单个 IP 地址进入同一站点。例如，多数公司用户通过单个代理服务器或一组代理服务器进行访问，它们都会被记录为同一 IP 地址。例如，AOL 用户都出示单个 IP 地址。

动态 IP 地址分配 (DHCP) 也可以在不同时间将不同的 IP 地址分配给同一工作站。利用仅 IP 地址方法，会将此新的 IP 地址解释为新的访问者。目前，基于仅 IP 地址的会话化不被视为计算页面视图或访问的可靠方法，因为它不符合任何审计规则。

E.3.3.2 具有外部参照的 IP

外部参照是一个网站，用户在该网站上单击链接或广告会被带到另一个网站。具有外部参照的 IP 地址的方法将外部参照作为唯一访问的关键指标：如果某个 IP 地址有外部参照，就会将其记录为一次新的访问。此方法也不可靠，因为它要求记录网址的所有别名。例如，站点 *www.ibm.com* 就有以下多个别名：*ibm.com*、*IBM.com*、*IBM.COM*、*9.19.138.191*，等等。此外，使用具有外部参照的 IP 地址的方法，每个网页访问都会被解释为*唯一访问者*。因而，此方法也不符合审计规则。

E.3.3.3 具有用户代理和外部参照的 IP

用户代理识别用户使用的操作系统和 Web 浏览器。与 IP 地址和外部参照相结合，用户代理可较好地地区分来自同一 IP 地址的访问者。通过将代理包含在识别过程中，可根据不同的代理，把来自同一 IP 地址的访问者标识为唯一。

此方法比前两种方法更准确，而且假定不同访问者会使用不同浏览器或不同版本浏览器以及不同计算机。例如，如果两个访问者来自 9.19.138.242，但其中一个访问者具有 MSIE 3.0 用户代理，而另一个访问者具有 MSIE 4.0 用户代理，然后就可有把握地解释为两个不同的访问者。

E.3.3.4 具有用户代理和 Cookie 的 IP

访问者 Cookie 是确定访问者在网站上身份的另一种方法。Cookie 是与某个用户关联的永久客户机状态 HTTP 标识符。将 IP 地址、用户代理，以及 Cookie 结合在一起，可比前面的方法更为准确地识别唯一访问者。但是，使用 Cookie 会产生一些问题，下面将对此详细讲述。

E.4 Cookie

服务器把 Cookie 存储在 Web 服务器和访问者之间的交换的客户机。服务器然后使用该信息记录和确定用户首选项、记住密码和登录 ID，并收集有关用户的附加统计信息。可以发行两种类型的 Cookie：会话 Cookie 和访问者 Cookie。

E.4.1 会话 Cookie

会话 Cookie 用于访问者识别。它们由站点为每次访问发出，并在不活动 30 分钟后过期。它们被公认为计算访问者数目的最准确方法。

网站利用会话 Cookie 分发方式的不同会影响最终结果。例如，服务器可以仅在视图前几页之后分配会话 Cookie。使用此 Cookie 分发方法，Cookie 前页面视图和 Cookie 后页面视图都会被算作单独的会话。这会使访问和访问者计数大量增加。而且越来越多的访问者目前在其浏览器范围内禁用(不接受)Cookie 的事实使得利用会话 Cookie 对访问者识别失效。

增加使用会话 Cookie 复杂性的策略是使用一种 Cookie 不可用时的退守方法。选择一种或多种不同方法作为退守方法意味着(很可能)无法比较两种方法的会话和访问者统计信息和汇总信息。例如, 如果在 Cookie 不可用时回复到基于 IP 的会话化, 所产生的 Web 日志信息将不同于会话 Cookie 可用时收集的信息。

E.4.1.1 访问者 Cookie

访问者 Cookie 是分发给访问者并且在会话结束后也不会从客户机删除的永久 Cookie。使用访问者 Cookie 是计算访问次数的另一种方法。

许多站点只在某个网页给访问者分配 Cookie。必须丢弃该网页之前的所有 Web 日志数据, 或者把它单独包含在分析中, 以确保统计信息的准确性。例如, 请看这样一个网站, 它有注册机制, 但只在访问者到达某一点时才要求访问者进行注册。这类网站的部分网页是允许自由访问的。而其余部分是私有的。在此情况下, 无法进行有关基于站点参照的注册用户行为的详细分析。

可以使用某种算法智能帮助解决此问题, 但多数这类站点没有进行这样的设计, 因此可以把新注册用户的信息与其未注册的信息联系起来。另外, 也可以禁用访问者 Cookie, 这会对会话准确性与禁用会话 Cookie 产生同样的影响。

E.5 其他问题

一旦选定如何识别和集合唯一访问者信息的方法, 就会有更多要做的工作。需要根据要进行的类型分析做出以下决定: 应把哪些网页定义为支付页(确定是否可以把访问者在网站上花费时间视为成功访问)、哪些网页可以查看或不可以查看、哪些类型的文件可以下载, 等等。必须分析、过滤和汇总 Web 日志, 以便生成适合定义和加载要分析的 OLAP 模型的元数据和数字数据。

E.6 实际示例

要进行分析，例如，哪个网页是最常请求的网页、访问者从哪里转来、或哪些是最常请求的可下载文件，您可能需要建立多个不同的 OLAP 模型。当然，需要建立的模型数量取决于要执行的分析类型以及属于 DB2 OLAP 实施的设计和规模问题。

第一步是确定在 Web 日志中进行分析所必需的数据元素。然后，用一种适用于您分析要求的格式分析、过滤、汇总和存储数据。

E.6.1 Web 日志项示例

请看以下日志项示例：

```
206.24.101.81 - John 30/Sep/1999:17:53:41 -0400] "GET
/javaclasses/headline.txt HTTP/1.0" 200 1175
"http://www.hyperion.com/index/" "Mozilla/4.6 [en] (Win95; U)" GET
/javaclasses/headline.txt - "HTTP/1.0"
```

表 25 描述日志项的某些不同部分(字段)。

表 25. 日志项

日志字段	示例
客户机的主机名或IP地址	206.24.101.81.(在此案例中，显示IP地址，因为没有启用Web服务器的DNS查询设置。如果启用DNS查询，就会出现客户机的主机名。)
RFC 931信息	未实施RFC 931身份， 身份验证服务器协议
用户名	John(客户机输入的用户名，用于身份验证)
请求日期/时间	30/Sep/1999:17:53:41 -0400 (-0400是GMT偏移值)
完整请求	GET /javaclasses/headline.txt HTTP/1.0
状态代码	200
传输的字节	1175
HTTP引用站点	Http://www.hyperion.com/index/

日志字段	示例
用户代理	Mozilla/4.6 [en] (Win95; U)
方法	GET
URI	/javaclasses/headline.txt
URI 的 查询字符串	- (在此案例中，什么也没有)
协议	HTTP/1.0

我们可以解释这些日志字段，以反映 Web 日志数据的不同维度。例如，“HTTP 引用站点”字段用于建立“引用站点”维度，而“用户代理”字段用于填充我们的“代理”和“平台”维度等。另外，会话化过程必须可以计算属于访问者行为的有用度量，例如，访问过的网页的数目、在每个网页上花费的时间、会话总长度，等等。

将维度信息与数字统计数据相结合和关联，我们就可以创建 Web 日志数据的逻辑和物理星型结构表示(从不同观点看)。然后，可以使用星型结构填充 OLAP 多维数据集，如图 65 所示。

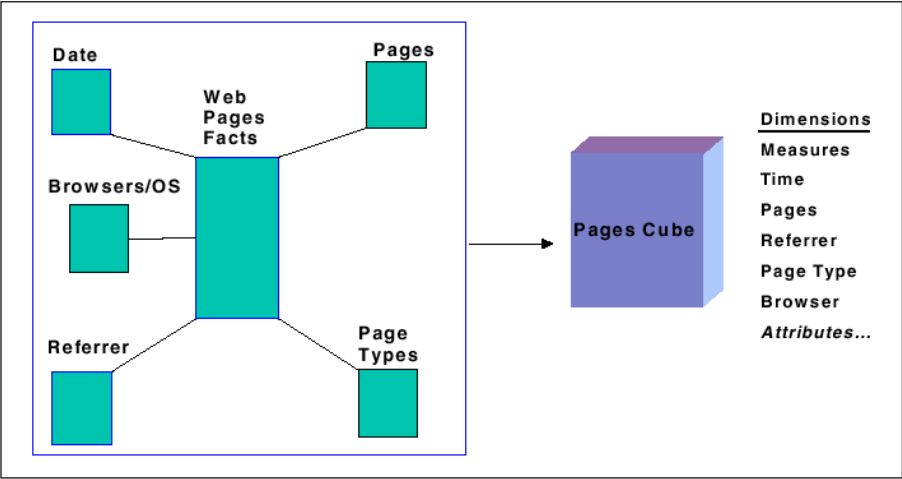


图 65. 填充 Web 多维数据集

E.6.2 两个 OLAP 模型示例

用以下特征建立的 OLAP 模型提供类似于以下问题的答案：哪些网页是最常访问的网页、访问者从哪里转来(以及转到哪些网页)、哪些是最常请求的可下载文件，等等(参见表 26)。

表 26: OLAP 模型示例

维度	建立方式	说明
日期	手动建立	日常时间维度
测度	手动建立	捕获查看次数的度量
引用	从 Web 日志动态建立	引用网页
网页	从 Web 日志动态建立	捕获所有网站页面。需要考虑一个分层结构，以便满足报表和分析需要。
网页类型	手动建立	用于类似于以下示例的属性标记网页：下载、搜索类型、非内容、支付。

此模型的 DB2 OLAP 大纲如图 66 所示。

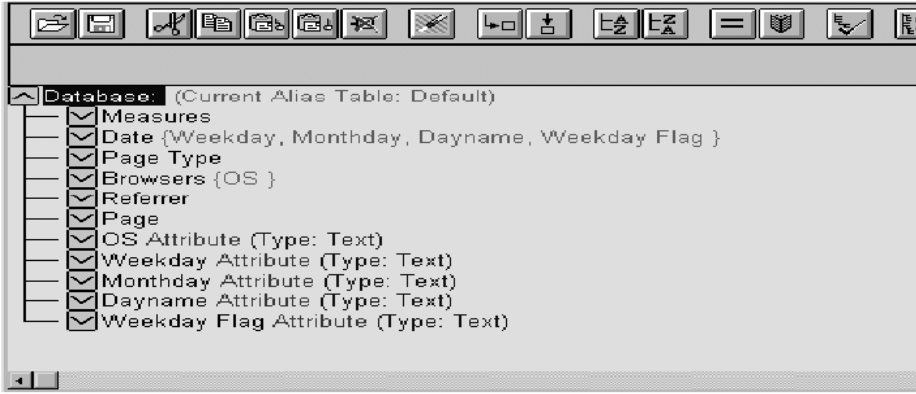


图 66. 示例模型的大纲

除了这第一个模型之外，可以利用原始模型中的一些维度并根据需要添加新的维度来生成第二个 OLAP 模型(参见表 27)。这还需要适当更改星型结构结构。添加几个新维度可以使模型分析诸如“网页未找到”错误或“内部服务器”错误这样的信息(参见表 27)。

表 27. 附加 OLAP 模型示例

维度	建立方式	说明
日期	网页重复利用	日常时间维度和属性
测度	手动建立	捕获请求数目的度量
状态代码	手动建立	标准状态代码
代理/平台	从 Web 日志动态建立	捕获访问者使用的所有用户代理和平台。需要考虑一个分层结构，以便满足报表和分析需要。
网页	网页重复利用	捕获所网站网页。需要考虑一个分层结构，以便满足报表和分析需要。

这第二个模型的 DB2 OLAP 大纲如图 67 所示。

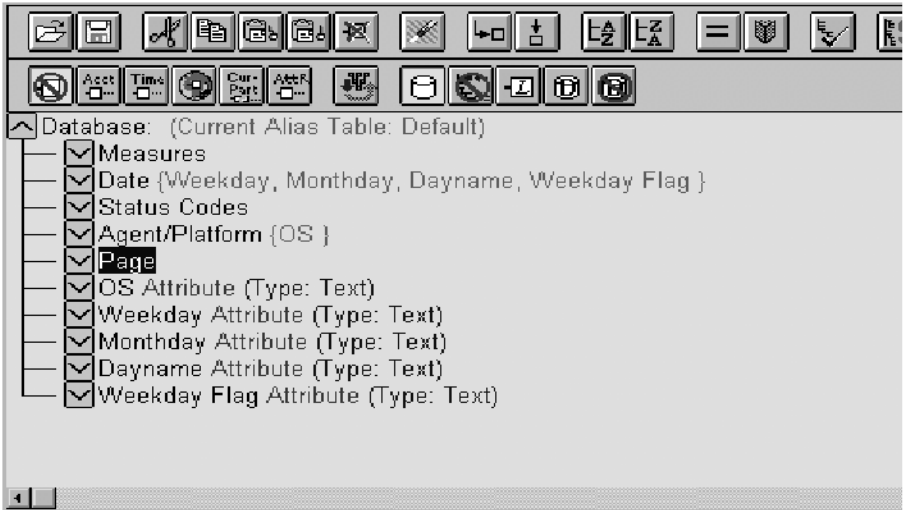


图 67. 附加模型的大纲

E.7 结束语：集成

分析上述 Web 日志信息会有明显收获，但焦点很大程度上在此电子领域内。最有希望和有益的回报是能够增强 Web 访问者信息与各种客户信息的汇总能力。

要使 Web 数据对于公司的价值最大化，就应把我们迄今为止讨论的数据视为一个数据集市，该数据集市对于运行网站的小组具有分析价值，而且，如果与公司商业智能系统以及数据仓库集成，该数据集市还对整个公司具有价值。

此数据应视为来自附加渠道的数据，并且此渠道与现有的各种渠道一样重要。集成可通过从 Web 分析多维数据集或数据集市提取相关数据并将其与来自更多传统渠道的类似数据进行合并来实现。这样，我们从一个包含所有渠道的客户视图获得客户信息，而且从此视图获得的信息要比从任何单个渠道获得的信息更为全面。

尽管公司数据仓库是最终集成点，但不一定是集成工作的必需起点。我们必须避免隔离 Web 数据，或者为了丰富 Web 数据，而从其他来源复制公司数据。

附录 F OLAP模型开发简明一览表

该附录以表 28 的形式显示第二章描述的(“OLAP 模型开发一览表”，第 21 页)OLAP 模型开发一览表摘录。

表 28. 项目简要目录

活动	备注?	检查?
1. 制定OLAP计划		
a. 识别OLAP机会。		
b. 评估OLAP机会。		
2. 开始进行高水平 OLAP 数据库结构建模		
a. 了解当前和所需报表要求		
• 评估所需维		
• 针对每个报表进行评估		
• 此时，开始设计维草图		
b. 分组练习建模开发		
• 把应用程序管理器作为一种快速应用程序设计工具		
• 将大纲透射到屏幕上，加快进展速度		
c. 使用SQL评估数据，以便了解包含的数据量，及检查数据的相关性		
• 如果数据为关系数据，请使用 SQL		
• 如果数据为非关系数据，使其成为关系数据		
• 检查用于建立维的数据中的空值		
d. 预估所需开发软件硬件		

活动	备注?	检查?
3. 建立进行大小测试的原型		
a. 建立维		
<ul style="list-style-type: none"> 直接使用应用程序管理器，手动键入维名称 		
<ul style="list-style-type: none"> 检查是否需要分区 		
<ul style="list-style-type: none"> 初步设定紧凑/稀疏 		
<ul style="list-style-type: none"> 始终使用esscmd.exe脚本建立大型维 		
b. 让大纲发挥作用		
<ul style="list-style-type: none"> 使用成员标记 <ul style="list-style-type: none"> 仅使用标签 使用动态计算 将公式放在数据块中 		
c. 加载测试数据		
<ul style="list-style-type: none"> 按稀疏维对数据进行排序 		
<ul style="list-style-type: none"> 如果是SQL源，请用SQL进行转换 		
d. 计算和调节		
e. 随时测试大小、计算性能和查询		
<ul style="list-style-type: none"> 评估模型 <ul style="list-style-type: none"> 检查模型的大小 评估计算时间的模型 		
<ul style="list-style-type: none"> 评估查询响应时间 <ul style="list-style-type: none"> 检查电子表格检索因子应用程序日志 用户验证和用户验收 调整稀疏/紧凑维，以适应用户数据请求 		

活动	备注?	检查?
f. 改进紧凑/稀疏, 优化大纲		
<ul style="list-style-type: none"> 使用稀疏/紧凑方法 使用配置向导 		
g. 根据原型大小测试, 调整维和成员		
4. 建立和加载最终模型		
a. 用户验收测试: 数据		
<ul style="list-style-type: none"> 使用电子表格工具 		
b. 用户验收测试: 访问用户工具		
c. 建立和安装安全模型		
<ul style="list-style-type: none"> 与用户一起检查安全模型 		
d. 训练用户		
5. 迁移到生产		
<ul style="list-style-type: none"> 开发和编写生产过程: <ul style="list-style-type: none"> - 维护大纲 - 刷新/更新数据 - 验证数据 - 备份和恢复 OLAP 数据库 - 维护软件级别并安装修补程序 - 捕获错误 		

附录 G 特别说明

本出版物旨在帮助设计人员和 DB2 OLAP 基础知识有一定了解的技术人员设计和实施未来的 DB2 OLAP 解决方案而设的。本出版物无意作为 IBM DB2 OLAP Server 提供的任何编程接口的规范。如欲了解可视为产品文档出版物种类的详细信息，请参阅“IBM DB2 OLAP Server 编程公告”中的“出版物”一节。

在本出版物中所提到的 IBM 产品、程序或服务并不意味着 IBM 将为所有 IBM 运作的国家提供。任何对 IBM 产品、程序或服务的引用并不说明或暗示只能使用 IBM 的产品、程序或服务。凡是同等功能的产品、程序或服务，只要不侵犯 IBM 的知识产权，都可以用来代替 IBM 产品、程序或服务。

本文档中的信息与特定设备的使用介绍结合在一起，但仅限于使用那些特定硬件和软件产品和标准的应用程序。

IBM 可能已经申请或正在申请与本文档有关的各项专利权。提供本文档并不表示允许您使用这些专利。您可以用书面方式将特许查询寄往：IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785, USA 。



为了以下目的：(i) 允许在独立创建的程序和其他的程序(包括本程序)之间进行信息交换和 (ii) 允许对已经交换的信息进行相互使用，而希望获取本程序有关信息的合法用户请与下列地址联系：IBM Corporation, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA 。

在一些案例或支付一定的费用。您可以通过适当的条款和条件来得到这些信息。

本文档包含的信息并没有提交给任何正式的 IBM 测试并发布。本信息的使用或任何这些技术的执行是客户的责任，并依靠客户的能力来评估它们，将其整合到客户的操作环境中。如果其中的条款由 IBM 在特定情形下被正确地校订，并不能保证在别处也可以得到相同或相似的结果。如果客户企图修改在他们自己环境下的这些技术，将自己承担风险。

本资料中对非 IBM Web 站点的任何引用只是为方便您而提供的，IBM 不以任何方式对这些 Web 站点作保证。

下列术语是国际商用机器公司在美国和/或其他国家的商标：

e (logo)® 	Redbooks
IBM®	Redbooks Logo 
AT	AIX
DB2 OLAP Server	DB2
Intelligent Miner	DataJoiner
DB2 OLAP Integration Server	QMF
DB2 Warehouse Manager	

以下术语是其他公司的商标：**Tivoli、Manage. Anything. Anywhere.、The Power To Manage.、Anything. Anywhere.、TME、NetView、Cross-Site、Tivoli Ready、Tivoli Certified、Planet Tivoli 和 Tivoli Enterprise** 是 **Tivoli Systems Inc.(IBM 子公司)**在美国和/或其他国家(或地区)的商标或注册商标。在丹麦，**Tivoli** 是经 **Kjøbenhavns Sommer – Tivoli A/S** 许可的商标。

C-bus 是 Corollary 公司在美国和/或其他国家的商标。

Java 及所有基于 Java 的商标及徽标均为 Sun 微系统公司在美国和/或其他国家的商标或注册商标。

Microsoft、Windows、Windows NT、Windows 商标均为微软公司在美国和/或其他国家的商标。

PC Direct 是 Ziff 通信公司在美国和/或其他国家的商标，并且许可 IBM 公司使用。

ActionMedia、LANDesk、MMX、Pentium 和 ProShare 是 Intel 在美国和/或其他国家的商标。

UNIX 是在美国及其他通过 Open Group 唯一特许的国家的注册商标。

SET、SET Secure Electronic Transaction 和 SET 标志是 SET Secure 电子交易公司的商标。

其他的公司、产品和服务名称可能是其自己的商标或服务标志。

附录 H 相关出版物

本节所列出出版物有助于了解本书包含的主题以外的更多详细信息。

H.1 IBM红皮书

如欲了解有关订阅这些出版物的详细信息，请参阅第 247 页的“如何获得 IBM 红皮书”。

- 适用于数据仓库的数据建模技术, SG24-2238
- 商业智能认证指南, SG24-5747
- S/390 商业智能体系架构表示指南, SG24-5641
- 商业智能应用程序容量规划: 途径和方法, SG24-5689
- 运行于 OS/390 的 DB2OLAP Server 入门, SG24-5665
- 智能数据挖掘器: 增强您的商业智能, SG24-5522
- S/390 在商业智能方面的作用, SG24-5625
- 建立适用于 BI OS/390 上运行的应用程序的 VLDB: 案例研究经验, SG24-5609
- 利用可视仓库和 DB2 管理多维数据集 OLAP Server, SG24-5270

H.2 IBM红皮书集

红皮书还有以下光盘版本。请单击 ibm.com/redbooks 上的 CD-ROM 按钮，即可了解有关所有光盘、更新和格式的信息。

光盘标题	红皮书全集工具箱编号
IBM系统/390红皮书集	SK2T-2177
IBM网络连接红皮书集	SK2T-6022
IBM事务处理和数据管理红皮书集	SK2T-8038
IBMLotus红皮书集	SK2T-8039
Tivoli红皮书集	SK2T-8044
IBM AS/400红皮书集	SK2T-2849
IBM Netfinity硬件和软件红皮书集	SK2T-8046
IBM RS/6000红皮书集	SK2T-8043
IBM 应用程序开发红皮书集	SK2T-8037

H.3 其它资源

以下关于 DB2 OLAP Server 出版物也可作为相关信息资源：

- *OLAP 数据库管理员指南，第 I 卷，SC27-0788*
- *OLAP 数据库管理员指南，第 II 卷，SC27-0789*
- *OLAP 快速技术参考，SC27-0790*
- *OLAPSQL 接口指南，SC27-0791*
- *OLAP 集成服务器模型用户指南，SC27-0783*
- *OLAP 集成服务器元大纲用户指南，SC27-0784*
- *OLAP 集成服务器管理指南，SC27-0787*
- *OLAP 电子表格插件用户指南(适用于 Excel)，SC27-0786*
- *OLAP 电子表格插件用户指南(适用于 1-2-3)，SC27-0785*

H.4 引用的网站

以下网站也可作为相关技术来源：

- <http://www-4.ibm.com/software/data/> IBM 数据库和数据管理主页
- <http://www.ibm.com/software/download/> IBM 免费软件下载网站
- <http://www.ibm.com/software/IBM> 软件站点
- <http://www.essbase.com/Hyperion> Essbase 网站
- <http://www.olapunderground.com/> 站点，专用于分发免费件 Essbase 信息、应用程序和代码。

如何获得IBM红皮书

本节说明客户和 IBM 员工如何了解 IBM 红皮书、红皮资料和光盘。提供通过传真或电子邮件订购红皮书和光盘的表格。

- **红皮书网站:** ibm.com/redbooks

请从红皮书网站搜索、查看、下载或订购硬拷贝/光盘红皮书。

同时请阅读本红皮书站点提供的红皮资料和下载补充资料(代码示例或磁盘/光盘图像)。

红皮资料是正在编撰中的红皮书;并不是所有红皮书都会成为红皮资料,有时只以这种方式发布几个章节。目的是以比正式发布过程更快的速度发布信息。

- **电子邮件订单**

通过电子邮件将包含 IBM 红皮书传真订单表提供的信息在内的订单发送到:

电子邮件地址	
美国或加拿大	pubscan@us.ibm.com
北美以外地区	联系信息位于以下站点的“如何订购”部分: http://www.elink.ibm.com/pbl/pbl
● 电话订购	1-800-879-2755
美国(免费电话)	1-800-IBM-4YOU
加拿大(免费电话)	国家协调处电话号码位于以下站点的“如何订购”部分: http://www.elink.ibm.com/pbl/pbl
北美以外地区	1-800-445-9269
● 传真订购	1-403-267-4455
美国(免费传真)	传真电话号码位于以下站点的“如何订购”部分: http://www.elink.ibm.com/pbl/pbl
加拿大	
北美以外地区	

此信息在出版时是最新的,但会随时的更改。读者可从红皮书网站查找到最新的信息。

IBM 员工用企业网

IBM 员工可以通过访问 IBM 企业网 <http://w3.itso.ibm.com/>并单击“ITSO 邮件列表”按钮,获得有关研讨会、高级培训和红皮书方面的信息。

从“资料”存储库查阅 ITSO 技术专业人员开发和编写的研讨会、演示文稿、论文和网页;单击“附加资料”按钮。员工可以访问位于 <http://w3.ibm.com/>的 MyNews, 获得红皮书、高级培训和研讨会公告。

IBM红皮书传真订单表

请填好以下表格发送给我:

[illegible]

我们接受 American Express、Diners、Eurocard、Master Card 和 Visa。并不是在所有国家都可以通过信用卡进行支付。信用卡支付必须签名。

词汇表

<p>A</p> <p>API 应用程序编程接口。例如, Essbase API 是一种函数库。您可以在自定义 C 或 Visual Basic 程序中使用这些函数访问 DB2 OLAP 服务器。</p> <p>API. Application Programming Interface. For example, the Essbase API is a library of functions that you can use in a custom C or Visual Basic program to access the DB2 OLAP Server.</p> <p>应用程序管理器 IBM DB2 OLAP 服务器和 Essbase 中包含的标准化管理工作, 用于建立 OLAP 数据库。</p> <p>Application Manager. Administration tool provided in standard with IBM DB2 OLAP Server and Essbase to build OLAP databases.</p> <p>参数 传递到某个 Web 服务器的名称/值对, 而且该服务器与某种动态资源的 URL 相关联。可以根据一个参数内不同值提供不同内容。通常以一个问号将参数字符串与页面资源隔开, 而用一个“&”(和)号将各个参数彼此隔开。</p> <p>argument. A name/value pair passed to a web server in association with the URL of a dynamic resource. Different content may be served based on different values within an individual argument. Usually argument strings are separated from the page resource by a question mark and separated from each other with an ampersand (&).A</p> <p>数组 一种与多维度数据库相关的矩阵结构。</p> <p>array. A matrix structure related to a</p>	<p>计算脚本 一种文本文件, 包含在 Essbase 数据库内执行计算的指令。</p> <p>calc script or calculation script. A text file that contains instructions to perform calculation within an Essbase database.</p> <p>单元 一个数据点, 指通过从多维度组中每个紧凑维中选择一个成员定义的交叉部分。</p> <p>cell. A single datapoint that occurs at the intersection defined by selecting one member from each dense dimension in a multidimensional array.</p> <p>点击率 此术语通常用于广告目的。它指在某个广告或链接出现次数对点击次数的比率。例如, 如果一个牙膏广告显示了 100 次, 但却只点击了一次, 可以说该广告的点击率为 1%。</p> <p>click-thru. This term is typically used for advertising purposes. It refers to the ratio of impressions to clicks on a particular advertisement or link. For example, a toothpaste ad that is shown 100 times but only clicked once has a 1% click-through.</p> <p>Cookie 与某个用户关联的永久客户机状态 HTTP 标识符。</p> <p>cookie. Persistent client-state HTTP identifier associated with a user.</p> <p>多维据集 请参见 OLAP 数据库。在关系数据库中, 多维度据集用一个事实表来表示。</p> <p>cube. See OLAP database. In relational, the cube is represented by a fact table.</p> <p>D</p>
---	---

<p>multidimensional database.</p> <p>B</p> <p>浏览器 一种在计算机上运行的软件程序,可以请求、加载和显示万维网上可用的文档。在我们的定义中,假定由“人”手动操作浏览器和查看网页。</p> <p>browser. A software program running on a computer that can request, load and display documents available on the World Wide Web. In our definition, it is assumed that a “human being” is operating the browser manually and viewing the page</p> <p>C</p> <p>高速缓存 一种内存元件。每个 OLAP 数据库都包含有一个数据高速缓存和一个索引高速缓存,而且您可以配置计算器高速缓存。</p> <p>cache. A component of memory. Each OLAP database contains a data cache, an index and you can configure a calculator cache.</p> <p>计算 (calc) 数据库大纲、计算脚本或报表脚本内的一种等式,计算报表中某个成员或点的值。</p> <p>calc or calculation. An equation within a data-base outline, a calculation script, or a report script that calculates a value for a particular member or point in a report.</p>	<p>数据加载 用数据填充 Essbase 数据库的过程。加载数据会为该数据库的数据库大纲中定义的单元格值建立实际值。</p> <p>data load. The process of populating an Essbase database with data. Loading data establishes actual values for the values of the cells defined in the database outline for the database.</p> <p>数据加载规则 在一个规则文件中注册的一组运算,当从一个外部源文件加载数据时,DB2 OLAP 服务器就会在这些数据上执行这组运算。</p> <p>data load rules. A set of operations registered in a rules file that the DB2 OLAP Server performs on data as it is loaded from an external source file.</p> <p>数据集市 一种面向主题的、集成的、随时间变化的数据集合,可用于针对某个特定用户组进行决策。</p> <p>datamart. A subject-oriented, integrated, time-variant collection of data to enable decision making for a specific group of users.</p> <p>数据挖掘 使用某个计算机应用程序搜寻和发现对商业以及数据间重要关系的新的认识。</p> <p>data mining. The use of a computer application to search for and discover new insights about the business and significant relationships among the data.</p>
--	--

<p>数据仓库 一种面向主题的、集成的、随时间变化的数据集，可用来针对一个完全不同的用户组进行决策。</p> <p>data warehouse. A subject-oriented, integrated, time-variant collection of data to enable decision making across a disparate group of users.</p> <p>数据库管理员 (DBA) 负责管理关系数据库的人员。</p> <p>DBA or database administrator. . A person responsible for administering a relational database.</p> <p>紧凑 如果某个多维度数据库的维度成员的相当大比例可能组合包含数据值，该数据库就是紧凑的。</p> <p>dense. A multidimensional database is dense if a relatively high percentage of the possible combinations of its dimension members contain data values.</p> <p>维 一种结构数据类别，例如，时间、账户、产品等。在大纲中，维度代表最高合并级别。</p> <p>dimension. A dimension is a structural data category such as time, accounts, products. In an outline, the dimensions represent the highest consolidation level.</p> <p>域 用户网络位置中包括地理位置的最高级别的部分 — .edu、.com、.org、.de、.jpe。</p> <p>domain. The affiliation of a user's network location at the highest level including geographic location - .edu, .com, .org, .de, .jpe.</p> <p>域名系统 (DNS) 因特网上使用的将 IP 地址映射到用户好用的名称的协议和系统。</p>	<p>退出页 一次访问中访问的最后一页。</p> <p>Exit page. The last page within a visit.</p> <p>F</p> <p>事实表 由测度和竞争数据构成的事实集合。</p> <p>fact table. A collection of facts consisting of mea-sures and contest data.</p> <p>FTP File Transfer Program (文件传输程序) 的缩写。</p> <p>FTP File Transfer Program.</p> <p>H</p> <p>命中 浏览器向 Web 服务器发出的对于任何项目 (包括图形、页面、空行或其它资源) 的请求。调出一个与 Web 浏览器中显示的相同网页可能会需要许多次命中。命中还指日志文件中的任意行。命中用于计算总命中次数以及传输的总字节。</p> <p>hit. A hit is a browser request to the web server for any item including graphics, pages, blank lines or other resources. It may take many hits to bring up a single web page as displayed in a web browser.</p> <p>A hit is also any line in a log file. Hits are used to calculate summary hit counts and total bytes transferred.</p> <p>主机名 某个 IP 地址解析成的名称和相关部分，例如，Hyperion.com 或 ibm.com。</p> <p>hostname. The name and affiliation that a particular IP address resolves to, such as Hyperion.com or ibm.com.</p>
--	--

<p>Domain Name System (DNS). A protocol and system used on the Internet to map IP addresses to user-friendly names.</p> <p>钻通 一种特定的分析技术，通过它，用户可以从一个关系数据库内数据中的 OLAP 多维度数据集进行导航，并检索数据仓库中存储的详细数据。</p> <p>drill through. Or drill thru is a specific analytical technique whereby the user can navigate from an OLAP cube among data ranging in a relational database and retrieve detailed data stored in a data warehouse.</p> <p>E</p> <p>进入页 一次访问中看到的第一个页面。</p> <p>Entry page. The first page within a visit.</p> <p>Essecmd 一种命令行界面，用于以交互方式或通过一个批文件执行服务器操作。</p> <p>Essecmd. A command-line interface used to perform server operations interactively or through a batch file.</p> <p>误码率是从您的ISP（互联网服务供应商）维护的地址池中在首次连接时分派的。每次连接的时候您都可以得到一个不同的IP地址。</p> <p>ber is assigned from a pool of addresses maintained by your ISP (Internet Service Provider) on a first-come basis. You can get a different IP address each time you connect.</p>	<p>HTTP 状态码 与请求关联的状态码，表明请求成功、不成功或其它情况。例如，如果某项资源找不到，浏览器就会返回一个 404 错误，该错误就是 HTTP 状态码的一个示例。</p> <p>HTTP status code. The status code associated with a request indicating whether it was successful, unsuccessful, or otherwise. For example if a resource is missing, the browser will return a 404 error, which is one example of an HTTP status code.</p> <p>Hyperion 会话程序 一种可扩展、可配置的进程，管理 Hyperion 网站分析应用程序的日志文件和其它 Web 数据的收集、处理和汇聚。</p> <p>Hyperion sessionizer. An extensible, config-urable process that manages the collection, processing and aggregation of log files and other web data for the Hyperion Web Site Analysis application.</p> <p>I</p> <p>IP 地址 一种识别 Internet 上所有计算机的唯一编号，。每当将某个计算机连接到 Internet 时，就会给该计算机分配一个 IP 地址。如果您有一个静态 IP 地址，每当您进行连接时，就会收到同一个编号。如果您有一个动态 IP 地址，就会按照首次访问原则从您的 ISP（Internet 服务提供商）保存的一个地址库中分配一个编号。您可以在每次连接时获得一个不同的编号。</p> <p>IP address. A unique number called an IP address identifies every computer on the Internet.</p> <p>Each time you connect to the Internet your</p>
---	---

	<p>machine</p> <p>is assigned an IP address. If you have a static IP address you get the same number each time you connect. If you have a dynamic IP address a num-cube. An OLAP database includes a database out-line,data, associated optional calculation scripts, optional report scripts, and data load rules.</p>
--	---

<p>IP 解析 将 IP 地址转换为用户可识别的域名的过程。此过程也称为反向 DNS。例如，198.105.232.4 变为 Hyperion.com。这种识别用户的方法可能会非常耗时，并可能会导致有关用户位置的不正确假设。</p> <p>IP resolution. The process of transforming IP addresses into user recognizable domain names. This process is also known as reverse DNS. For example 198.105.232.4 becomes Hyperion.com. This method of identifying users can be time consuming and may result in incorrect assumptions about user location.</p> <p>信息技术 (IT) 信息技术代表在企业中从事信息系统工作的技术人员。</p> <p>IT. Information Technology represents the technical staff working on Information System in an enterprise.</p> <p>M</p> <p>度量 事实的数字属性，表示事实的维度的性能或行为。</p> <p>measure. A numeric attribute of a fact representing the performance or behavior of its dimensions.</p>	<p>OLAP 集成服务器 从关系源建立多个大纲的模型和开发这些大纲的功能部件。</p> <p>OLAP Integration Server. Feature to model and develop multiple outlines from a relational source.</p> <p>大纲 定义 OLAP 数据库所有元素的多维结构。大纲包含维度和成员、紧凑或稀疏维度标记和属性、计算、共享成员以及数据库基本上卷结构的替代物。</p> <p>outline. The multidimensional structure that defines all elements of an OLAP database. It contains definitions of dimensions and members, dense or sparse dimension tags and attributes, calculations, shared members, and alternations to the basic roll-up structure of the database.</p> <p>P</p> <p>页分辨率 一种测度，以用户在一个页面上花费的秒数计。</p> <p>page resolution. A measure in seconds of the amount of time a user spends on a single page.</p> <p>页命中次数 未由 Hyperion 会话程序预先过滤的所有资源的测度。此数字包括可查看和不</p>
--	---

<p>成员 一个维度内的分立元件。嵌入各个成员时，这些成员就组成某个维度的层次结构。</p> <p>member. A discrete component within a dimension. When embedded, members compose the hierarchy in a dimension.</p> <p>元大纲 元大纲是使用 Hyperion 集成服务器或 OLAP 集成服务器建立的，它包含建立 DB2 OLAP 大纲以及将数据加载到 DB2 OLAP 数据库的基本结构。</p> <p>metaoutline. A metaoutline built using Hyperion Integration Server or OLAP Integraton Server contains the basic structure to build a DB2 OLAP outline and to load data into a DB2 OLAP database.</p> <p>模型 模型是使用 Hyperion 集成服务器或 OLAP 集成服务器建立的，包含一种星形架构，并以关系数据集市为基础。</p> <p>model A model built using Hyperion Integration Server or OLAP Integraton Server contains a star schema and is based on relational datamarts.</p> <p>O</p> <p>OLAP On-Line Analytical Processing（联机分析处理）的缩写。这是一种软件技术，它使分析师或经理人员能够通过快速、一致和交互地访问多种可能的信息视图深入了解数据，这些信息视图反映用户所了解的企业的实际维度。</p> <p>OLAP. On-Line Analytical Processing is a software technology that enables analysts and executives to gain insight into data through fast consistent, interactive access to a wide variety of possible views of information that reflect the real dimensionality of the enterprise as understood by the user.</p>	<p>可查看的页数。</p> <p>page hits. A measure of all resources not prefiltered by the Hyperion Sessionizer. This number includes both viewable and non-viewable page counts.</p> <p>页视图 通过 Internet 浏览器查看的单个网页。网页在 Hyperion 应用程序中预定义为可查看或不可查看。只有可查看的页面才会被算作页视图。预定义页视图包括所有以 HTML、ASP、JSP 等结尾的资源。页视图也称为页印象。</p> <p>page view. Page view is a single web page as viewed through an Internet browser. Pages are predefined in the Hyperion application as either viewable or nonviewable. Only viewable pages are counted as page views. Pre-defined page views could include all resources ending in HTML, ASP, JSP, to name a few. Page views are also known as page impressions.</p> <p>支付 支付页是由某个站点标为重要的网页，因为如果用户点击这些网页，网页就具有收入含意。例如：Edmunds.com 可以将 Jeep Grand Cherokee 页标为支付页，并附上一个每次查看该页的美元数（.60）的标志。然后，Hyperion 网页分析功能就可以产生表明该节或该页的美元数的报表。同样地，假定无论什么时间有人点击包含 Danoff 博士的文章的网页，AHN 网站都必须向 Danoff 博士支付一定款项。Hyperion 网站分析功能可以在支付页上标上付出款项，并表明是此页所值的美元数。然后，当我们在查看这组文章的新访问量与重复访问量时，就可以判断出就吸引和/或留住客户而言，这些文章是否是一项很好的投资。</p> <p>payoff. Payoff pages are pages tagged by a site as important because they have revenue</p>
--	--

<p>OLAP 数据库 一种多维度数据库或多维度数据集。OLAP 数据库包括数据库大纲、数据、关联的可选计算脚本、可选报表脚本以及数据加载规则。</p> <p>OLAP database. A multidimensional database or anytime that somebody hits the page containing Dr. Danoff's article. Hyperion Web Site Analysis can tag the payoff page as a payout, and indicate the dollars of cost for this page. Then when we start looking at new versus repeat traffic for this set of articles we could start figuring out whether the articles are a good investment in terms of attracting and/or retaining customers.-</p>	<p>implications if a user hits them. For example: Edmunds.com can tag the Jeep Grand Cherokee page as a payout page and attach an attribute of dollars (.60) per page view. Hyperion Web Site Analysis can then produce a report indicating dollars for that section or that page. Similarly, suppose the AHN Web site must pay Dr. Danoff. mode, using the ESSCMD command-line interface, or through the Application Manager. The script is a text file that contains data retrieval, formatting, and output instructions.</p>
--	---

<p>Q</p> <p>查询字符串 附加在页请求末尾的字符串。例如： recipe.jsp?cui-sine=Mexican&timeofday=morning。</p> <p>query string. String attached to the end of a page request. For example: recipe.jsp?cui-sine=Mexican&timeofday=morning.</p> <p>R</p> <p>RDBMS Relational DataBase Management System（关系数据库管理系统）的缩写，用于将关系对象作为数据库、表、索引进行管理和控制。</p> <p>RDBMS. A Relational DataBase Management System to manage and control relational objects as databases, tables, indexes.</p> <p>参考页 参考某个用户查看过的某个网页的前一页。</p> <p>referring page. References the previous page</p>	<p>RLE Run Length Encoding（行程长度编码）的缩写。 Essbase 引擎使用的一种压缩方法。</p> <p>RLE or Run Length Encoding. A compression method used by Essbase engine.</p> <p>S</p> <p>搜索 页面属性，表明页面状态，与在搜索进程中使用的某种状态相同。</p> <p>search. An attribute of a page indicating its status as something that is used in the search process.</p> <p>搜索查询 用户作为查询提交的搜索词。</p> <p>search. query Search terms that the user has submitted as a query.</p> <p>稀疏 如果某个多维度数据集的维度的相当大比例成员的可能组合（交集）中包含遗漏数据，该数据集就是稀疏的。</p> <p>sparse. A multidimensional dataset is sparse if a relatively high percentage of the possible</p>
---	---

<p>to a particular page that a user is looking at.</p> <p>参考站点 参考用户在其中点击导向您的站点的链接或广告的任何站点。也称为访问起点。例如：用户从 yahoo.com 上点击一个将其导向 Hyperion.com 的链接。</p> <p>referring site. Refers to any site where a user clicked on a link or an advertisement to get to your site. Also known as visitor origin. Example: A user clicks on a link from yahoo.com that sends them to Hyperion.com.</p> <p>关系数据库 一种数据库，看上去是一组表和索引，可以通过 SQL 语言并根据数据的关系模型进行操作。</p> <p>relational database. A database that can be perceived as a set of tables and indexes and manipulated in accordance with the relational model of data, through the SQL language.</p> <p>关系表 关系数据库中的二维结构，由列和行组成，可通过 SQL 进行访问。</p> <p>relational table. A 2 dimensions structure in a relational database composed of columns and rows and accessed through SQL.</p> <p>报表脚本 一种 ASCII 文件，包含生成一个或多个生产报表的“报表生成程序”命令。可以使用 ESSCMD 命令行界面或通过应用程序管理器以批处理方式运行报表脚本。脚本是一种包含数据检索、格式化和输出指令的文本文件。</p> <p>report script. An ASCII file that contains Report Writer commands that generate one or more production reports. Report scripts can be run in</p>	<p>combinations(intersections) of the members from the dataset's dimensions contains missing data.</p> <p>电子表_插件 可与 Microsoft Excel 和 Lotus 1-2-3 无缝合并的软件。软件库看上去像附加到电子表上的一个菜单，具有诸如连接、检索、放大和计算这样的功能。</p> <p>spreadsheet_add-in. Software that merges seamlessly with Microsoft Excel and Lotus 1-2-3.</p> <p>The software library appears as a menu add-In to the spreadsheet and provides such features as connect, retrieve, zoom-in, and calculate.</p> <p>星型架构 关系数据库架构类型，由一个主要事实表和一组维度表组成。事实表中包含实际数据数字值，而维度表中则包含成员及其相互关系的数据。</p> <p>star schema. The type of relational database schema composed of a main fact table and a set of dimension tables. The fact table holds the actual data numeric values and the dimension tables hold data about members and their relationships.</p> <p>状态码 用户作为查询提交的搜索词。状态码是由 Web 服务器返回到 Web 日志的，Web 日志定义请求属于已成功、已重新定向，还是导致错误。</p> <p>status code. Search terms that the user has submitted as a query. The code returned by the web server to the web log that defines whether the request was successful, redirected, or resulted in an error.</p>
---	--

batch	
-------	--

<p>结构化查询语言 (SQL) 一组确定的语句，用来管理关系数据库中存储的信息以及添加、删除和更新表中的信息。</p> <p>Structured Query Language. (SQL) An established set of statements used to manage information stored in a relational database and add, delete, update information in a table.</p> <p>子域 提供商级用户网络位置的相关部分，例如，aol.com、ibm.com、Microsoft.com 等。使用反向 IP 查找，可以将一个用户的 IP 地址变成一个子域。</p> <p>sub-domain. The affiliation of a user's network location at the provider level such as aol.com, ibm.com, Microsoft.com. Using the process of reverse IP lookup, a user's IP address is resolved into a subdomain.</p> <p>U</p> <p>统一资源标识符 (URI). 访问 Web 上对象的所有名称和地址类型的通用术语。URL 就是一种 URI。</p> <p>Uniform Resource Identifier (URI). The generic term for all types of names and addresses that refer to objects on the web. A URL is one kind of URI.</p> <p>统一资源定位器 (URL) 一个导向 Internet 上某个对象或其它目的地的地址。例如： www.hyperion.com/solutions。</p> <p>Uniform Resource Locator (URL). An address to an object or other destination on the Internet. Example: www.hyperion.com/solutions.</p>	<p>访问 指在某个网站进行的一系列请求（查看的页面）。当指定时间段（默认值为 30 分钟）过去后没有对该网站的新的请求时，访问就会结束。访问通常通过 IP 地址、用户代理和/或 Cookie 的一些唯一组合进行识别。一次访问也称为一次会话。</p> <p>visit. A visit is a series of requests (pages viewed) by a user while at a single web site. A visit ends when a specified period of time (default is 30 minutes) has passed without any additional requests to the site. A visit is usually identified by some unique combination of IP address, user agent and/or cookies. A visit is also known as a session.</p> <p>访问持续时间 一种测度，以用户在单次访问内所花费的秒数计。</p> <p>visit duration. A measure in seconds of the amount of time a user spends within a single visit.</p> <p>访问长度 页视图测度，以用户在一次访问内查看的页数计。</p> <p>visit length. A measure in page views of the amount of pages a user views within a single visit.</p> <p>访问超时 非活动时间长度，确定访问必须在这个时间长度内结束。通常将 30 分钟作为默认值。</p> <p>visit timeout. The length of time of inactivity during which a visit is determined to have ended. A default value of 30 minutes is normally used.</p>
---	---

<p>唯一访问者 一种表示某个时间段内唯一标识符数目的测度。通常按天、按周和按月进行测量。一天内访问某个站点两次的访问者将仅算作一次。</p> <p>unique visitor. A measure that represents the count of unique identifiers within a time period. Usually measured per day, week and month. A visitor that visits a site twice in one day will only be counted once. The</p> <p>用户代理 Web 日志的组成部分，识别用户使用的操作系统和浏览器。</p> <p>user agent. The part of a web log, which identifies the Operating System and browser employed by the user.</p> <p>用户名 某个用户在某个站点的注册用户名。</p> <p>user name. Registered user name of a user at a particular site.</p> <p>V</p> <p>VBA Visual Basic 的缩写，一种开发程序。</p> <p>可查看 一种页面属性，表明页面是页视图还是页命中。通常把可查看页定义为可包含广告标题的任何资源。帧或其它浏览元素不会被识别为可查看。</p> <p>VBA or Visual Basic. Development program.</p> <p>viewable. An attribute of a page indicating if it is a page view or a page hit. Usually a viewable page is defined as any resource that can contain an ad banner. A frame or other navigation element would not be identified as viewable.</p>	
--	--