第5章　数据仓库和技术

在许多方面，数据仓库比数据库需要一系列更简单的技术。数据仓库中没有联机的数据更新；只有非常少的一些锁定需要；而且对于远程处理接口的需要也只是最基本的；等等。然而，数据仓库有许多技术上的需求。这一章就讲述一下这方面的要求。

5.1　管理大量数据

对于数据仓库，第一个也是最重要的技术需求就是能够管理大量的数据，如图 5-1所示。

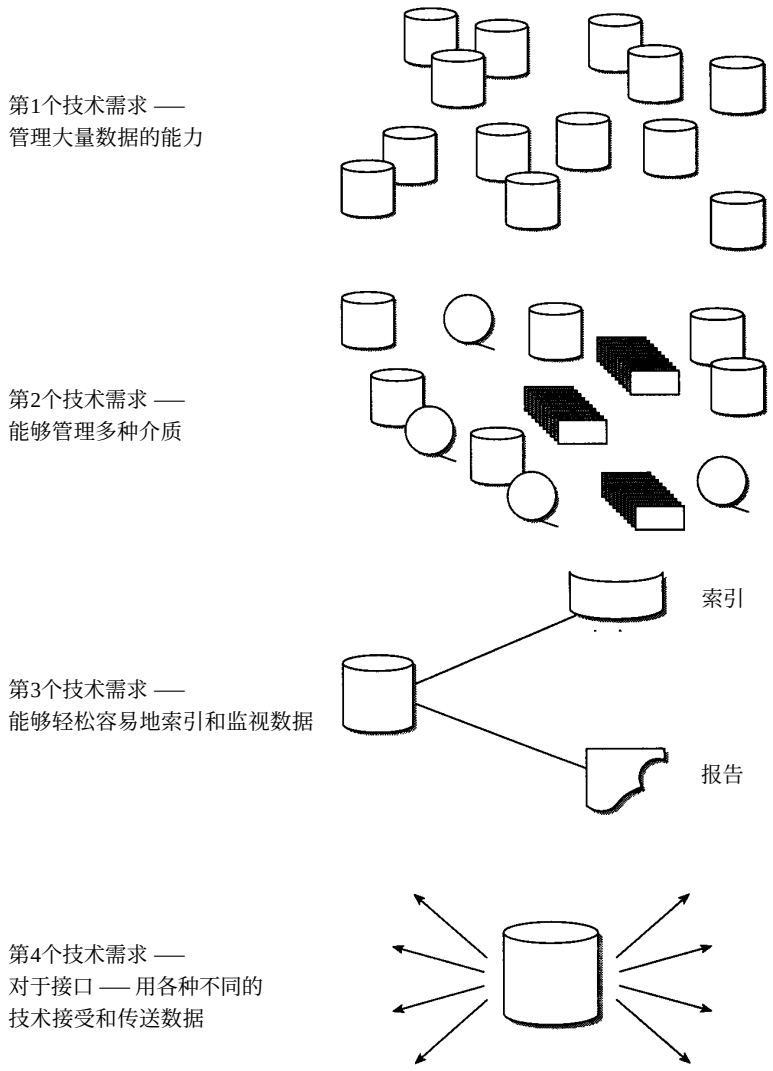


图5-1　对支持数据仓库的技术的一些基本需求

有好多种管理大量数据的方法——通过寻址，通过索引，通过数据的外延，通过有效的溢出管理，等等。管理大量的数据有两方面——能够管理大量数据的能力和能够管理好的能力。任何声称支持数据仓库的技术一定都要满足能力与效率的要求。

数据仓库开发者建造数据仓库时，在理想的情况下是假定其能够满足处理大量数据的需求的。在开发和实现数据仓库的时候，如果开发者不得不对技术进行扩展以适应数据仓库，那么所用的基本技术就存在一定的问题。

当谈到数据仓库时，问题不仅是基本的技术及其效率，还有存储和处理的费用也是要考虑的因素。

5.2 管理多介质

在处理大量数据时，为了满足高效率和合理的费用，应用在数据仓库中的基本技术应该能够解决多种存储介质的问题。仅仅在DASD上管理一个成熟的数据仓库是不够的。考虑到访问速度和存储费用，对数据的存储要分层次。层次的区分如下：

主存	——非常快	——非常贵
扩展内存	——非常快	——贵
高速缓存	——非常快	——贵
DASD	——快	——适中
光盘	——不慢	——不贵
缩微胶片	——慢	——便宜

由于数据仓库中的大量数量和被访问到的可能性这两方面因素存在，一个满载的数据仓库应该放在多种存储层次上。处理数据仓库技术应该能管理多种存储介质上的数据。

5.3 索引/监视数据

数据仓库的灵魂就在于灵活性和对数据的不可预测的访问。这一点也就是要求能够对数据进行快速和方便的访问。数据仓库中的数据如果不能方便和有效地检索，那么建立数据仓库这项工作就不是成功的。当然，设计者可以利用许多方法来使数据尽可能地灵活，例如利用双重粒度级和数据分割。但这些技术一定要支持方便的索引，一些索引技术常常是有用的，如二级索引、稀疏索引、动态索引、临时索引等等。而且，建立和应用索引的费用不能太高。

相同地，数据仓库中的数据也应能随意地被监视。监视数据的费用也不能太高，过程不能太复杂，监视程序在需要时应能随时运行。

有很多理由要监视数据仓库中的数据，包括：

- 决定是否应数据重组。
- 决定索引是否建立得不恰当。
- 决定是否有太多数据溢出。
- 决定数据的统计成份。
- 决定剩余的可用空间。

如果数据仓库技术不支持对数据的方便和高效地监视的话，那么它就不适用。

5.4 多种技术的接口

数据仓库另一个非常重要的问题是要能够用各种不同的技术获得和传送数据。如果在向

数据仓库传送数据和从数据仓库传递数据时有很大限制，那么这种支持数据仓库的技术实际上是没有用的。

接口不仅要高效还要便于使用，并能够在批模式下运行。在联机模式下运行是有趣的，但并不非常有用。

5.5 程序员/设计者对数据存放位置的控制

为了对数据进行高效地访问和更新，程序员 /设计者需要在物理的块/页的一级上对数据的存放进行特殊的控制，如图 5-2所示。

某项技术将数据放到它认为合适的地方是完全可以的，只要该项技术能在需要时被明确地管制。如果某项技术非要将数据存放在某一物理地址而不允许程序员管制，那么它就犯了严重的错误。

程序员/设计者时常对数据的物理位置进行整理来使之适合其用途。这样做可以使数据的访问更加经济。

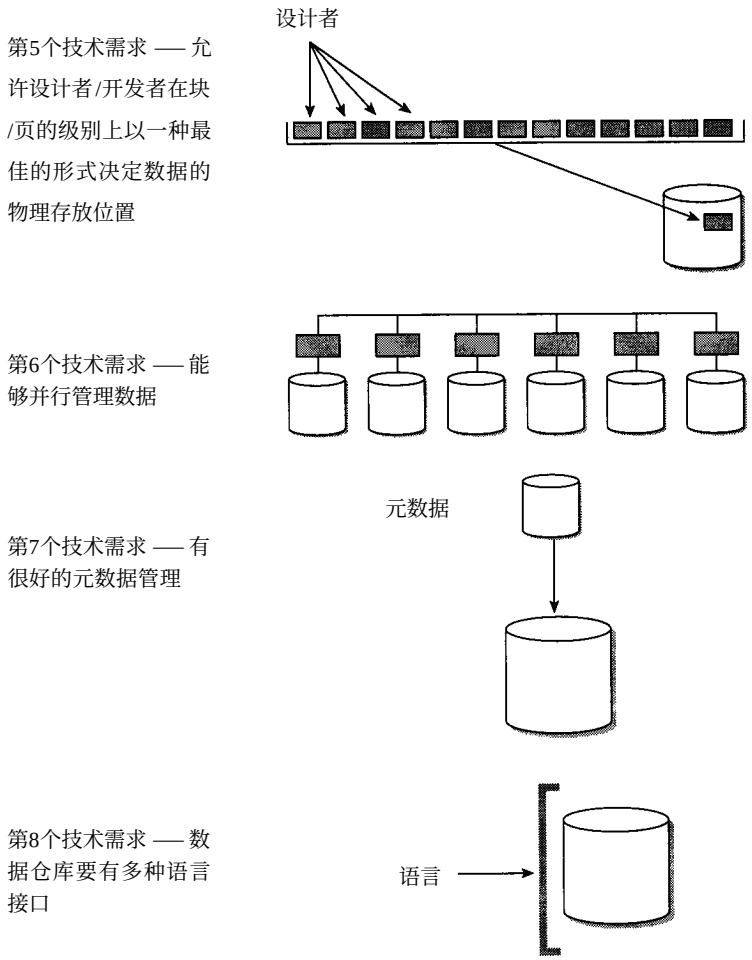


图5-2 数据仓库的另外一些技术需求

5.6 数据的并行存储/管理

数据仓库中数据管理的最重要的特征之一是数据的并行存储 /管理。当数据被并行存储和管理时，性能上会提高很多。通常，假定对数据的访问都是等概率的话，性能的提高与数据所分布的物理设备的多少成反比。

整个数据的并行存储/管理是非常复杂和重要的，在这里的论述是不够的，但应先在此提及一下。

5.7 元数据管理

由于各种各样的原因，数据仓库中元数据比在传统操作型的数据库中更重要。元数据之所以重要是由于与数据仓库相关的开发生命周期是完全不同的，数据仓库是在一种启发式的、反复的开发生命周期上运作的。为了更加有效，数据仓库的用户应该能够对准确和实时的元数据进行访问。没有一个好的元数据来源来运作的话， DSS分析员的工作就非常困难。典型的元数据包括：

- 数据仓库表的结构。
- 数据仓库表的属性。
- 数据仓库的源数据（记录系统)。
- 从记录系统到数据仓库的映射。
- 数据模型的规格说明。
- 抽取日志。
- 访问数据的公用例行程序。

5.8 语言接口

数据仓库需要有非常丰富的语言规定。没有一种健壮的语言，数据仓库中进入接口和访问数据就非常困难。而且，访问数据仓库的语言一定要是高效的。

典型的数据仓库语言接口需要：

- 能够一次访问一组数据。
- 能够一次访问一条记录。
- 特别要保证，为了满足某个访问要求能够支持一个或多个索引。
- 有SQL接口。
- 能够插入、删除、更新数据。

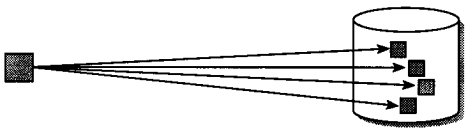
5.9 数据的高效装入

数据仓库的一个重要的技术能力就是要能够高效地装入数据，如图 5-3所示。

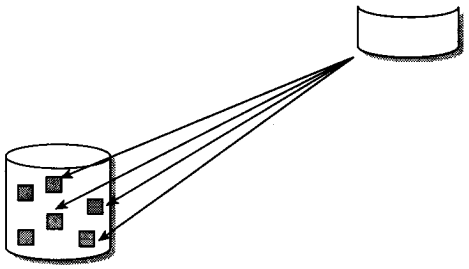
有好多种装入数据的方法：通过一个语言接口一次一条记录或者一起使用一个程序一次全都装入。另外，在装入数据的同时，索引也要高效地装入。在有些时候，为了平衡工作负载，数据索引的装入可以推迟。

如果数据仓库中数据的装入有不可克服的困难，那么这个数据仓库就没有用处了。

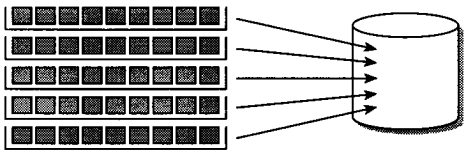
第9个技术需求 —— 能够高效地装载数据仓库



第10个技术需求 —— 有效地使用索引



第11个技术需求 —— 能够以压缩的方式存放数据



第12个技术需求 —— 支持复合键码

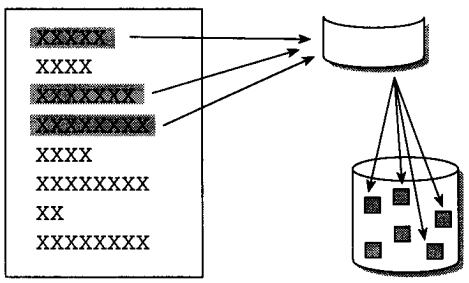


图5-3 进一步的技术需求

5.10 高效索引的利用

数据仓库技术不仅必须能够方便地支持新索引的创建和装入，而且要能够高效地访问这些索引。

有多种方法能够高效地访问索引：

- 用位映像的方法。
- 用多级索引。
- 将部分或全部索引装入内存。
- 当被索引的数据的次序允许压缩时对索引项进行压缩。
- 创建选择索引或范围索引。

除了索引的高效存储和浏览外，在主存储器层次对数据后续的存取也是很重要的。然而不幸的是对主存数据访问的优化并不像对索引数据的访问一样有那么多选择。

5.11 数据压缩

数据仓库的成功之处就在于能够管理大量的数据。达到这一目的的中心是数据的压缩。当数据能够被压缩时，它便能存储在很小的空间中。这尤其与数据仓库的环境有关，因为数据在插入到数据仓库中后，是很少被更新的。数据仓库中数据的稳定性减少了空间管理问题，这些问题是在更新紧密压缩的数据时发生的。

压缩的另一个好处是程序员可以完全脱离给定的输入 /输出操作。当然，对数据的访问就会有相应的解压缩的问题。虽然解压缩需要一定的开销，但这个开销不是 I/O资源的开销，而是CPU的开销。通常，在数据仓库环境中 I/O资源比CPU资源少得多，因此数据的解压缩并不是一个主要的问题。

5.12 复合键码

数据仓库环境中一种简单而又重要的技术需求是能够支持复合键码。这种键码在数据仓库环境中随处可见，主要是因为数据仓库中数据的随时间变化特性。

5.13 变长数据

数据仓库环境的另一个简单而又重要的技术需求是有效管理变长数据的能力，如图5-4所示。

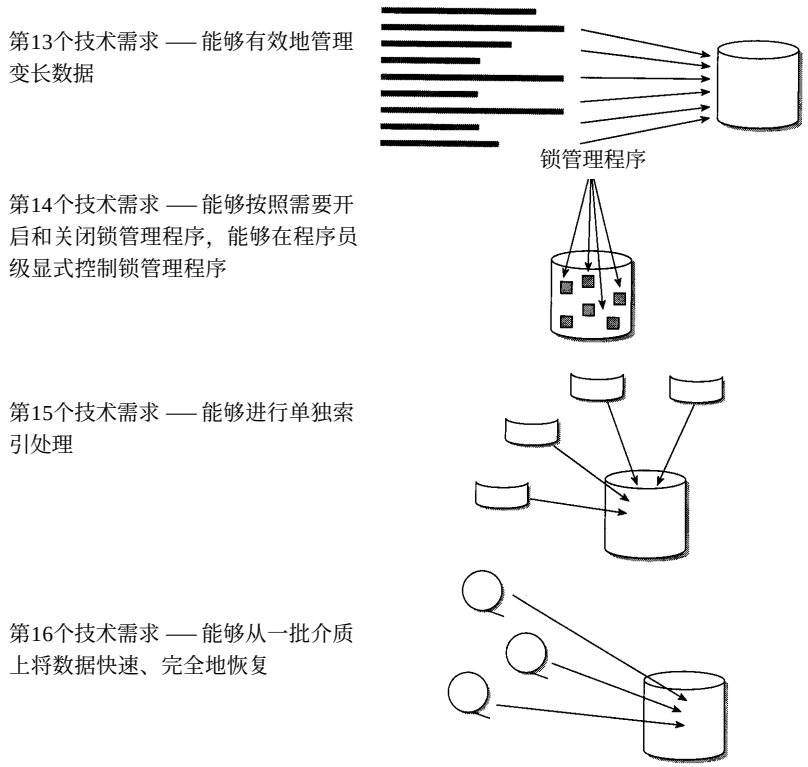


图5-4 数据仓库还需要的另外一些技术

变长数据如果被经常更新和改变，就会产生性能上的严重问题。但当变长数据很稳定，如在数据仓库中时，就没有固有的性能问题。

另一方面，由于数据仓库中数据的多样性，对数据的变长结构的支持是强制性的。

5.14 加锁管理

数据库技术的一个基本部分是加锁管理。加锁管理程序确保没有两个或两个以上的用户在同一时间对同一数据进行更新。但数据仓库中是没有更新的。

应用加锁管理程序的后果之一是它消耗了相当的资源，即使数据不被更新也是一样。简单地一直把加锁管理程序打开是会消耗很多资源的。为了使数据仓库环境更加合理，需要有选择地将加锁管理程序打开或关闭。

5.15 单独索引处理

数据库管理系统的一个基本特征是能够进行单独索引处理。在许多情况下，只查看一个索引(或一些索引)就可以提供某些服务。当只通过查看一下索引就可以满足某些请求时，由于用不着查看数据的最初数据源而会更加有效。但并不是所有的 DBMS都有辨别索引是否能满足请求的智能。

5.16 快速恢复

数据仓库环境的一个简单而重要的技术特性，是能够从非直接存取存储设备快速地恢复数据仓库表。当可以从二级存储设备上恢复时，就可能节约大量开支。如果没有能从二级存储设备上快速恢复的能力，通常的做法是将 DASD的数目增加一倍，然后将增加出的数目作为恢复/复原的存储区。

5.17 其他的技术特征

这里所讨论的特征只是最重要的一些。有好多其他的特征需要我们考虑，但由于数量太多而不便详述。

值得注意的是传统技术中的很多其他特征只起很小的作用（如果它们还起作用的话)。这样的一些特征包括：

- 事务集成性。
- 高速缓存。
- 行/页级的锁定。
- 参照完整性。
- 数据视图。

5.18 DBMS类型和数据仓库

随着数据仓库的到来，以及人们认识到 DSS是现代信息系统基本结构的必不可缺少的一部分，一类新的 DBMS产生了。这种新的数据库管理系统可以称为“ 数据仓库专用数据库管理系统 ”。数据仓库专用数据库管理系统是特别为数据仓库和决策支持而优化设计的。

数据仓库环境中的处理类型可以概括为装入和访问过程。数据从操作型数据环境中集成、转换和装入到数据仓库中去。数据到了数据仓库后，在那里被访问和分析。在数据仓库中，数据一旦被装入，通常是不被更新的。如果需要对数据仓库进行数据的更正和调整的话，也是在对数据仓库数据没有分析操作的空闲时间进行。

传统数据库环境和数据仓库环境的另一个重要的区别在于，数据仓库中有更多的数据量，比一般的操作型环境中要多得多。数据仓库中的数据量以 10GB或100GB计，而一个通用的DBMS通常管理的数据要比它少得多。数据仓库要管理大量的数据，是因为它们：

- 包括粒状的、原子的细节。
- 包括历史信息。
- 包括细节和汇总数据。

谈到基本的数据管理功能，数据仓库用与标准的操作型 DBMS非常不同的一组参数来进行优化。

传统的通用数据库管理系统和数据仓库专用数据库管理系统的第一个也是最重要的区别在于数据的更新。前者必须适合于记录级的更新，并将其作为操作的一个标准部分。由于记录级的数据更新是一通用DBMS的常规特征，所以它必须提供以下功能：

- 锁定
- 提交
- 检测点
- 日志处理
- 死锁处理
- 回退

不仅这些是DBMS的常规特征，它们还要花费巨大的开销。有趣的是，当 DBMS 不被使用时也要耗费这笔开销。换句话说，至少对某些 DBMS当仅执行只读操作时， DBMS也要提供更新和锁定的开销。依赖于通用 DBMS，更新所需的开销能够被最小化，但不能完全没有。而对于一个数据仓库专用的DBMS来说，不会付出任何更新的开销。

通用DBMS和数据仓库专用DBMS之间的第二个主要区别是基本的数据管理。对于通用的DBMS来说，对数据在块级上的管理要包括一些附加的空间，这些空间用于以后更新和插入数据时块的扩张。典型情况下，这些空间是些自由空间。对于通用 DBMS，自由空间可能要占到50%。对于数据仓库专用的DBMS，根本就不需要自由空间，因为数据一旦装入到数据仓库后是不需要更新的，也就没有扩展物理块的需要。事实上，给定了数据仓库中要管理的数据量后，留下以后不会用到的大量空间是没有意义的。

数据仓库和通用环境之间的另一个有关的区别反映在不同类型的 DBMS上，是索引的区别。通用DBMS环境限制在有限数量的索引，有这个限制是因为当有数据的更新和插入时，索引本身需要数据管理。然而，在数据仓库环境中，没有数据的更新，却有必要对数据的访问进行优化，也有多种索引的必要 (和机会)。事实上，数据仓库相对于操作型的、面向更新的数据库来说，能够应用更健壮和更完善的索引结构。

除了索引、更新和物理块级上的基本数据管理以外，在数据管理能力和策略上，通用DBMS和数据仓库专用DBMS之间还存在其他一些基本区别。其中，这两种类型的 DBMS的最基本的区别可能是在物理上以优化方式组织数据以适应不同类型访问的能力。通用 DBMS在

物理上组织数据是为了优化事务的访问和处理。这种模式下的组织允许许多不同类型的数据通过一个公共键码在物理上聚集起来，并能有效地通过 1次或2次I/O访问。信息访问的最优化的数据通常有很不同的物理描述。在物理上组织它们而使对同一类型数据的多次不同访问能够通过1次或2次I/O高效地进行。

数据能够在物理上进行优化以便于事务访问和 DSS访问。通用的DBMS优化数据是为了事务的访问，而数据仓库专用的 DBMS物理上优化数据是为了便于决策支持系统的访问和分析。

5.19 改变DBMS技术

对于信息仓库的一个有趣的考虑是，在数据仓库数量增长后， DBMS技术的变化。有几个理由来说明为什么进行这样的改变：

- 当今可用的DBMS技术，当数据仓库首次载入数据时并不一定适合。
- 数据仓库已经变得非常之大，以至于应该提出新的技术方法。
- 数据仓库的利用已经提高许多，也改变了许多，使得现在的数据仓库的 DBMS技术已经不适用了。

总之，我们有可能要一次又一次地重新审视基本的 DBMS技术。

是否应考虑找一种新的DBMS技术？我们要考虑的是什么？以下的几点非常重要：

- 新的DBMS技术是否满足可预知的需求？
- 从旧的DBMS向新的DBMS的转换应该怎样去做？
- 转换的程序应该怎样改变？

所有的这些考虑中，最后一个是最令人头痛的。即使在最好的环境中，试图去改变转换程序也是一项很复杂的工作。

5.20 多维DBMS和数据仓库

一项在数据仓库中经常讨论的技术是多维数据库管理系统。多维数据库管理系统（有时也叫“ 数据集市 ”)提供了一种信息系统结构使得对数据的访问非常灵活，可以用多种方法对数据进行切片、分割，动态地考察汇总数据和细节数据的关系。多维 DBMS不仅提供了灵活性，还可以对终端用户进行管理，这些非常适合 DSS环境。如图5-5所示，多维DBMS和数据仓库之间有非常有趣和互补的关系。

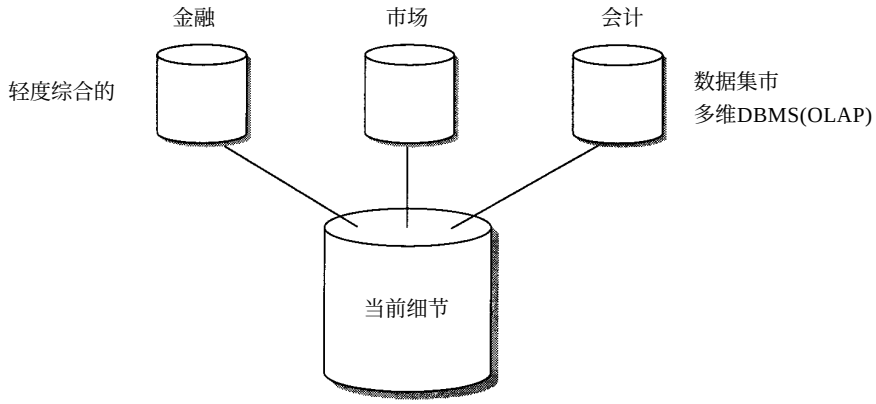


图5-5 数据仓库的传统结构以及当前细节数据如何同部门数据(或多维DBMS，数据集市)配合

数据仓库中的细节数据为多维DBMS提供了非常健全和方便的数据源，多维DBMS需要定期地刷新。为此，数据要定期从数据仓库中导入到多维 DBMS中去。由于传统应用程序的数据在进入数据仓库时被集成，多维 DBMS就用不着从操作型环境中抽取与集成数据，并因些而受到推崇。另外，数据仓库在最低级别保存数据为那些利用多维 DBMS的用户进行低级别的分析提供了基础。

可能有人会认为多维 DBMS应该作为数据仓库的数据库技术。除了一些非常特殊的情况外，这种想法是不正确的。使得多维 DBMS技术最佳的那些性质并不是数据仓库的基本的重要特性。数据仓库中最重要的特性也不是多维 DBMS技术的特性。

看一下数据仓库和多维DBMS的区别：

- 数据仓库有大量的数据；多维DBMS中的数据至少要少一个数量级。
- 数据仓库只适于少量的灵活访问；而多维 DBMS适合大量的非预知的数据的访问和分析。
- 数据仓库内存储了很长时间范围内的数据 —— 从5年到10年；而多维DBMS 中存储着较短时间范围内的数据。
- 数据仓库允许分析人员以受限的形式访问数据，而多维 DBMS 允许自由的访问。

多维DBMS和数据仓库有着互补的关系，而不是数据仓库建立在多维 DBMS之上的关系。

数据仓库和多维 DBMS关系中有趣的一点是，数据仓库为非常细节的数据提供了基础，而这在多维DBMS中通常是不能看到的。数据仓库能容纳非常详细的数据，这些数据在导入多维DBMS时被轻度综合了。而导入到多维DBMS之后，数据会被进一步地汇总。在这种模式下，多维DBMS可以包含除了非常细节以外的所有数据。使用多维 DBMS的分析者可以一种灵活和高效的方式来对多维 DBMS中所有不同层次的数据进行钻取。如果需要的话，分析者还可以向下钻取到数据仓库。通过这种方式将数据仓库和多维 DBMS相结合，DSS分析者可以得到这两者的好处。DSS分析者大部分时间里可以在多维DBMS中享受其操作高效的优点，同时如果需要的话，还可以向下钻取最低层次的细节数据。

数据仓库和多维DBMS的另一个互补的方面是汇总的信息在多维 DBMS中计算和收集后被存储在数据仓库中。通过以这种方式将数据进行融合，汇总数据在数据仓库中比在多维DBMS中能够存储更长的时间。

数据仓库和多维DBMS还有一个方面是互补的。多维DBMS存放中等时间长度的数据，依应用的不同从12个月到15个月。而数据仓库存放数据的时间跨度要大得多 —— 从5年到10年。考虑到这一点，数据仓库就成为多维 DBMS分析者进行研究的源泉。多维DBMS分析者乐于知道，如果需要的话，有大量的数据是可用的，但在不需要时却用不着在他们的环境中花费存储所有这些数据的代价。

多维DBMS有不同的特色。一些多维DBMS建立在关系模型上，而另一些多维 DBMS建立在能优化“ 切片和分块 ”数据的基础上，在这里数据可以被认为存储在多维立方体内。后者的技术基础可以称为“ 立方体基础 ”。

两种技术基础都支持多维DBMS数据集市，但这两种技术基础之间存在着一些差异：

多维DBMS数据集市的关系型基础

- 优点：
  - 能支持大量数据。

- 能支持数据的动态连接。
- 已被证实是有效的技术。
- 能够支持通用的数据更新处理。
- 如果对数据的使用模型不清楚的话，关系型结构与其他任何结构一样好。

■ 弱点：

- 性能上不是最佳的。
- 不能够单纯对访问处理进行优化。

多维DBMS数据集市的“立方体”基础

■ 优点：

- 对于DSS处理性能上是优化的。
- 能够对数据的非常快访问进行优化。
- 如果已知数据访问的模式，则数据的结构可以优化。
- 能够很轻松地进行“切片和分块”。
- 可以用多种方法检测。

■ 弱点：

- 几乎不能处理像标准的关系模型那么多的数据。
- 不支持通用的更新处理。
- 装入的时间很长。
- 如果对路径的访问不被数据设计所支持的话，这种结构就显得不灵活。
- 对数据的动态连接的支持是有问题的。

多维DBMS(OLAP)是一种技术，而数据仓库是一种体系结构的基础。这两者之间存在着互补的和共生的关系。最一般的情况下，数据仓库作为多维 DBMS的基础 —— 从中选出细节数据的一个子集传到多维 DBMS中，在那里，数据要么被汇总，要么被聚积。但在某些团体中有一种观点认为多维DBMS并不需要一个数据仓库作为它的数据的基础。

如果没有数据仓库作为多维DBMS的基础，那么装入多维DBMS中的数据就是直接从老的、传统的应用环境中得到。图 5-6展示了数据直接从传统环境装入多维 DBMS中去的情形。由于它是直截了当的，并很容易实现，所以这种方法很吸引人。一个程序员能够立刻开始工作来建造它。总之，这一设计的简单、快捷对于终端用户来说是一种诱惑，使他们认为构造多维DBMS的这种方法是合适的。

然而不幸的是，图 5-6所示的结构中有一些并不显而易见的缺陷。由于各种各样的原因，使得用数据仓库的当前详细级的数据装入多维 DBMS比直接从传统的操作型应用环境中装入数据更有意义。

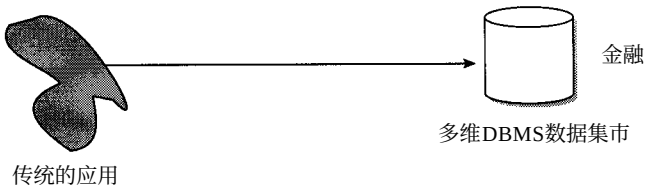


图5-6 从还没有当前细节数据的应用建立多维DBMS数据集市

图5-7展示了用数据仓库的当前详细级的数据装入多维 DBMS环境的方法。老的传统的

操作型数据被集成、转换，再传入到数据仓库中。一旦到了数据仓库，被集成的数据就存储在细节数据的当前级中。多维 DBMS就是从数据仓库中这一级别的数据之中得到其数据。

第一眼看上去，图5-6和图5-7所示的两种结构似乎并没有本质上的区别。事实上，将数据首先装入到数据仓库中似乎是浪费精力。但是，有一个非常好的理由说明为什么在创建多维DBMS的第一步要先将数据集成到数据仓库中。

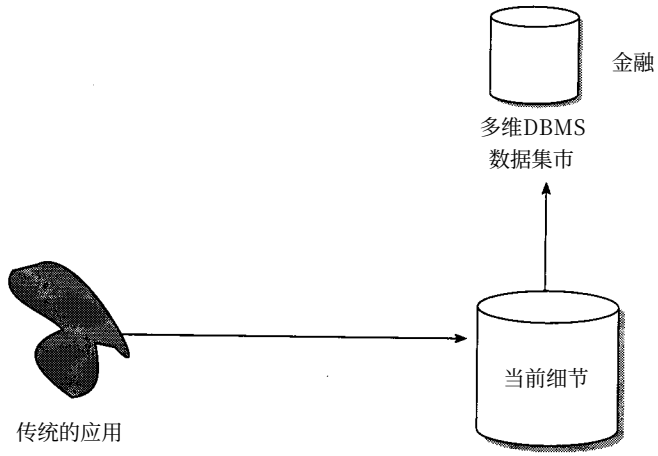


图5-7 数据从应用环境传入当前细节级然后再传入多维DBMS数据集市

考虑一下这样一个事实，在通常情况下，一个公司需要建立多个多维 DBMS。金融需要它们自己的多维DBMS，财务部门也需要它们自己的。市场部、销售部和其他一些部门都需要他们自己的多维DBMS。因为在这个公司里将会有众多的多维 DBMS，所以图5-6所示的方案就会变得非常复杂。这样图 5-6 就扩展为一个现实的方案 —— 有众多的多维DBMS都要直接地和独立地从传统系统环境中获得数据，如图 5-8所示。

图5-8展示了众多的多维DBMS直接从相同的传统应用中获得数据的情况，这种结构有什么不对呢？

问题如下：

- 抽取数据所需进行的开发量是巨大的。每一个不同部门的多维 DBMS都需要开发一套适合它的抽取程序。抽取处理过程有大量的重叠。浪费的开发工作量很大。当多维DBMS是从数据仓库中抽取数据时，它只要一套集成和转换的程序就够了。
- 当多维DBMS是从传统系统环境中直接提取数据时，是没有集成的基础的。每一个部门的多维DBMS对于怎样从不同的应用中集成数据都有它们自己的解释。不幸的是，一个部门集成数据的方法通常和其他部门集成相同数据的方法是不同的。结果导致最终没有单一集成的确定的数据源。在数据仓库建造时，本来有一个能够作为基础的单一的、集成的、确定的数据源。
- 为了维护所进行的开发工作量是巨大的。旧的传统应用的一种单一改变就会影响许多抽取程序。在有抽取程序的地方都要做一些改动，而且这种改动会很多。当有了数据仓库后，由于只需要写很少的程序来处理传统环境和数据仓库的接口，所以应用中的改变产生的影响也是很少的。

直接从应用到多维DBMS方法不能行之有效的一个主要原因

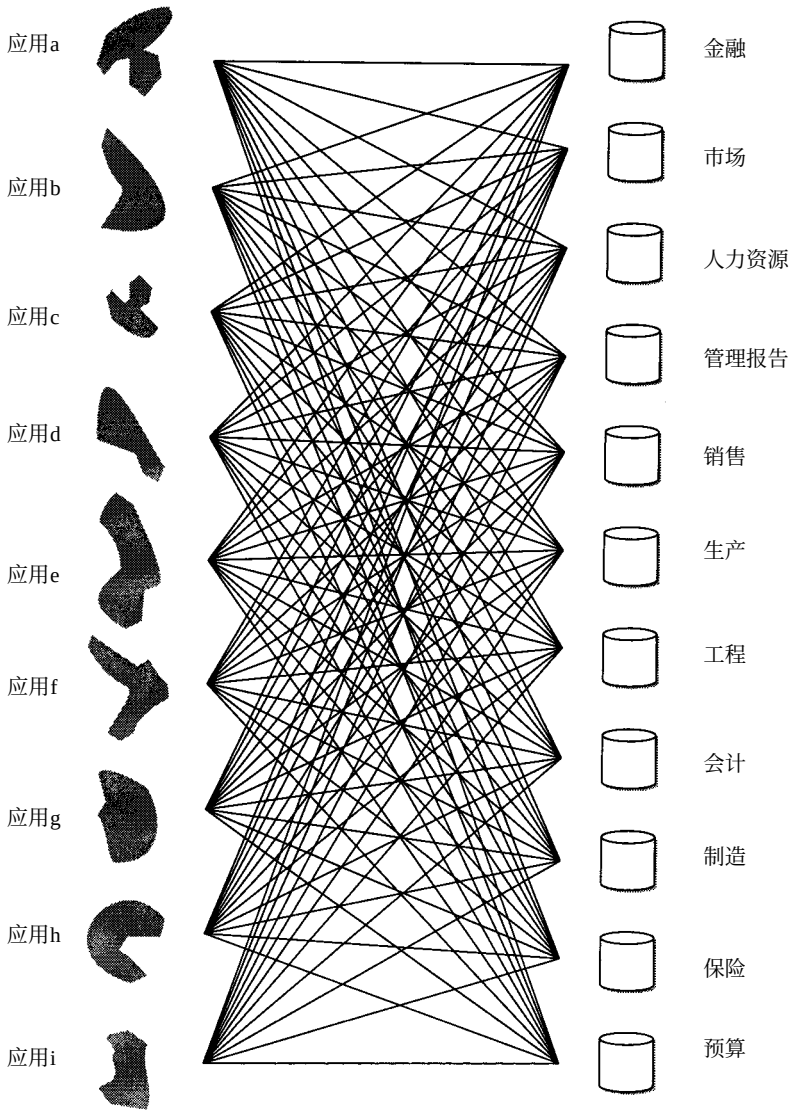


图5-8 有许多的应用和许多的数据集市，每对之间都需要一个接口。

回避细节数据的当前级的后果是产生无法管理的“蜘蛛网”

- 需要消耗的硬件资源的数量。对于每一个部门的每一个抽取处理，同样的传统的数据都要顺序地重复地传送。而在数据仓库中，为了刷新数据仓库中的数据，传统数据只需要传送一次。
- 从传统环境中将数据直接导入多维 DBMS环境中的复杂度使得对元数据的高效管理和控制受到损失。而在数据仓库可以对元数据进行直接的捕捉和管理。
- 缺乏数据的一致性。当不同的部门存在意见上的分歧时，由于它们各自都有自己的多维DBMS，这就很难解决。但用数据仓库后，冲突的解决是很自然和简单的。

5.21 双重粒度级

数据仓库的一个有趣的方面是通常会创建双重环境。一个环境是能够进行联机的、交互式处理的DASD环境，另一个环境通常是磁带处理或其他具有不同特征的大容量存储环境。在许多情况下，支持DASD环境的底层技术和支持大容量存储环境的底层技术是不同的。当在数据仓库中有双重环境时，采用混合技术是正常和自然的。

但是，还有另一种方式把技术区分开，这种方式不是正常的或自然的。可以想象，数据仓库环境 —— DASD部分 —— 可以用多种技术将之分开，但除非这种区分是分布式数据仓库的一部分，否则，是不提倡这种方法的。

5.22 数据仓库环境中的元数据

在数据仓库环境中的元数据所扮演的角色和在操作型环境中数据所扮演的角色是不同的。在操作型环境中，元数据几乎被当成文档来处理并且降低到同样的重要性级别。然而，在数据仓库环境中，元数据的重要性提高了。数据仓库环境中元数据的重要性如图 5-9所示。操作型数据和数据仓库中的数据服务于两类不同的群体，操作型数据由 IT专业人员使用，许多年来IT人员都是偶然地使用元数据。 IT专业人员不仅懂计算机，而且由于学历背景和所受的培训，他们会在系统中找到他们自己的方法。然而，数据仓库数据是给 DSS分析者用的。DSS分析人员通常首先是专业人员，他们通常没有很高的计算机水平。为了能够有效地使用数据仓库环境，DSS分析人员需要尽量多的帮助，而元数据恰能很好地帮助他们。另外，在 DSS分析者计划该怎样去做信息型/分析型处理时，他们要首先去看元数据。由于所服务的人员的种类不同，以及元数据在每天的工作中所起的作用不同，元数据在数据仓库环境中比在操作型环境中重要得多。然而还有其他的一些原因使得数据仓库的元数据很重要。

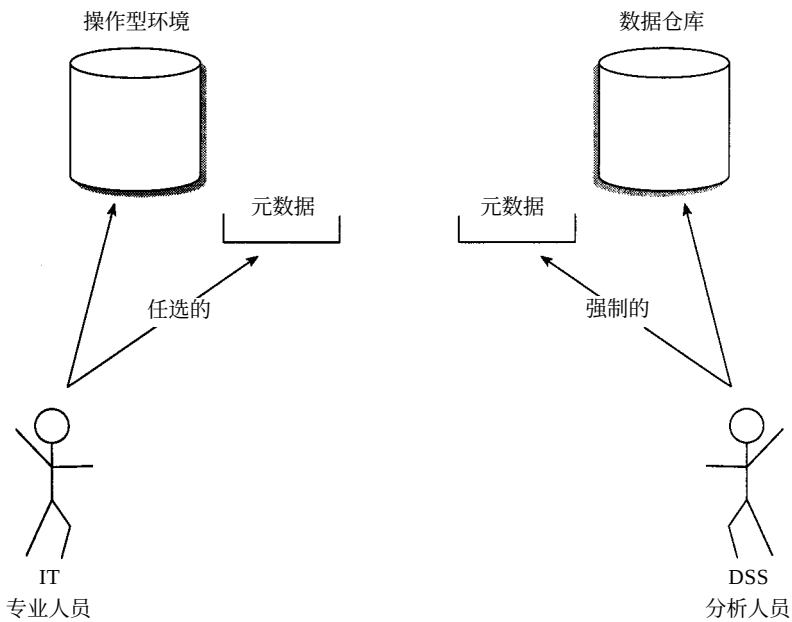


图5-9 IT专业人员偶尔使用元数据，DSS分析人员经常使用元数据并作为分析的第一步

数据仓库的元数据所以重要的另一个原因，涉及到从操作型环境到数据仓库环境的映射，图5-10展示了这一观点。

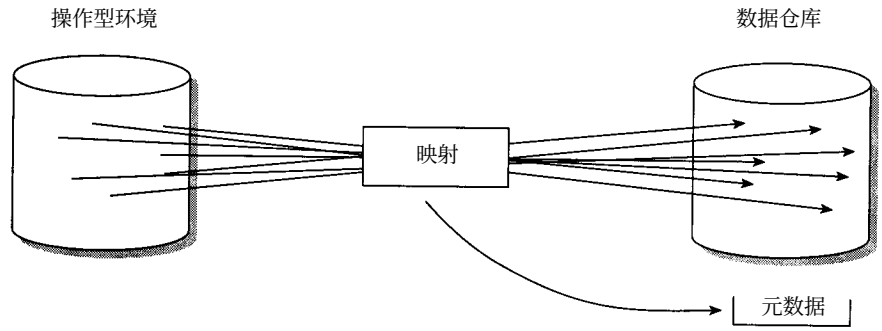


图5-10 操作型环境和数据仓库环境之间的映射是需要元数据的另一个主要原因；没有这种映射，对接口进行控制是非常困难的

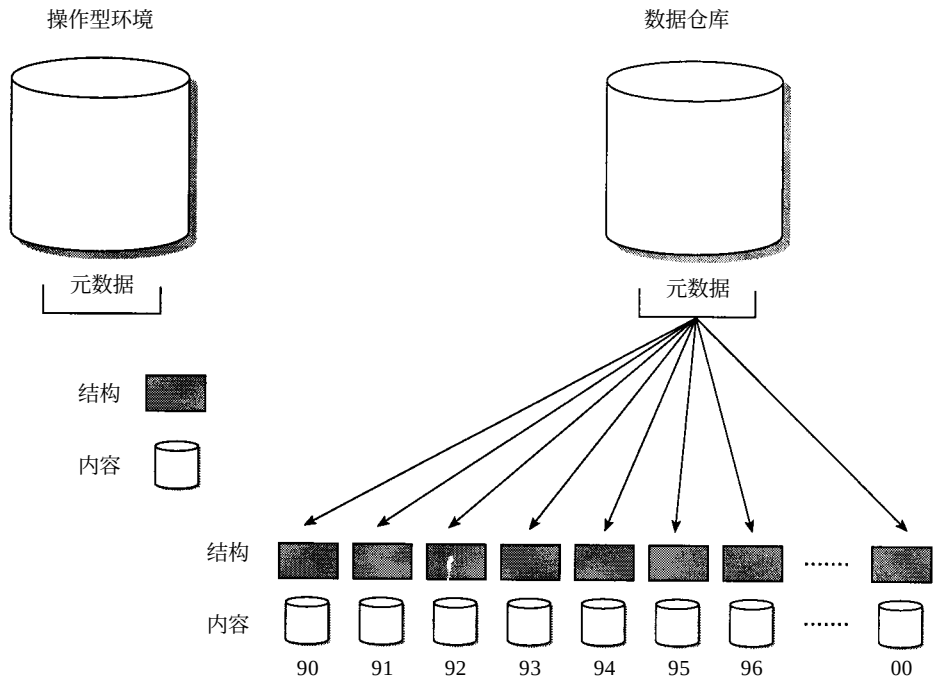


图5-11 数据仓库中包含很长一段时间的数据，因此必须管理多种数据结构/定义。操作型环境假设在任一时刻只有唯一的一个正确的数据定义

当数据从操作型环境传入数据仓库环境时，数据要经历一个重大的转变。转化、过滤、汇总、结构改变等等都会发生。有必要去跟踪一下这些转变，而数据仓库中的元数据正是能做到这一点的理想所在。当一个管理者需要将数据从数据仓库追溯到操作型环境中时 (最终的向下钻取)，对这种转变保持一个细致的记录的重要性就显而易见了。在这种情况下，对数据转变的记录恰恰是描绘了怎样从数据仓库钻取到操作型环境的源数据。

对于数据仓库环境中的元数据需要细致管理有另外一个重要原因，如图 5-11所示。数据

仓库中数据会存在一段很长的时间 —— 从5年到10年。而在5年到10年这么长的时间段内，数据仓库改变它的结构是很正常的。换句话说，一个数据结构能在 5到10年内保持不变是很不平常的。那么，随着时间的流逝来跟踪数据结构的变化，则是数据仓库中元数据很自然的一项任务。

将数据仓库环境中随时间变化的多种数据结构的概念与操作型环境中的元数据比较一下。在操作型环境中，我们假定在任何时候，对数据结构有且仅有一个正确的定义。

5.23 上下文和内容

数据仓库引起人们兴趣的一个方面就是可以对数据在很长一段时间内进行存储和管理，数据仓库要求对数据进行5年到10年或更长时间的管理。

在过去，经典的操作型信息系统集中注意于公司的当前数据。在操作型世界中，强调的重点是此刻帐目的余额是多少，或此刻的存货有多少，或此刻货物的运送情况如何。当然任何一个组织都有必要知道当前的信息。但对过去一段时间的信息进行考察也有真正的价值。当一个组织能够考察很长一段时间的信息时，发展趋势就很明显了。这是仅查看当前信息所观察不到的。数据仓库定义中的一个重要特征就是能够对一段时间内的数据进行存储和访问。

考察作为数据仓库一部分的时间谱，使我们认识到了数据新的一个维 —— 上下文维。为了阐明上下文信息的重要性，下面给出了一个例子。

假定一个管理者想从数据仓库中要一份 1995年的报表。报表生成后，管理者很满意。事实上，由于管理者很满意，所以他便想要一份 1990年的报表。由于数据仓库载有历史信息，这样的一个要求并不难实现， 1990年的报表生成了。现在管理者将这两份报表 —— 1990年的一份和1995年的一份 —— 拿在手中，并且宣布这些报表是一场灾难。

数据仓库工程师检查了报表，发现 1995年的财政报告显示收入为 $50,000,000，而1990年的报告对同一种类显示为 $10,000。管理者宣称从来没有任何方法使任何种类的帐目在 5年内增长这么多。

在就要放弃之前，数据仓库工程师向管理者指出报表中还有一些相关的因素没有显示出来。数据仓库中 1990年和1995年的数据是从不同来源得到的； 1990年的产品定义不同于 1995年的； 1990年和1995年有不同的市场领域； 1990年和1995年也有如对贬值的不同的计算。另外，还有许多不同的外部因素需要考虑，如在通货膨胀、税款、经济预测方面的差别等等。一旦报表的上下文向管理者解释了之后，内容就显得非常可接受和可解释了。

在这个简单而又常见的例子中，一段时间内数据的内容如果毫无附加信息地显示，那么内容本身也就是非常难于理解、难以置信的。然而，当上下文加到一段时间内数据的内容上去时，内容和上下文都变得非常令人明白。

为了理解和解释一段时期内的信息，需要一个全新的上下文维。当信息的内容仍十分重要时，一段时期内信息的比较和理解使得上下文和内容同等重要。而在过去的几年中，上下文是一直未被发现、未被开发的信息的一个维。

5.24 上下文信息的三种类型

我们需要管理三个级别的上下文信息：

■ 简单上下文信息。

■ 复杂上下文信息。
■ 外部上下文信息。

简单上下文信息与数据本身的基本结构有关，包括如下一些内容：

■ 数据的结构。
■ 数据的编码。
■ 数据的命名约定。
■ 描述数据的度量，如：
  • 数据的多少。
  • 数据增长速度。
  • 数据的哪一部分增长。
  • 数据是怎样被使用的。

简单上下文信息在以往是用字典、目录、系统监视器等管理的。复杂上下文信息描述的是和简单上下文信息相同的数据，但是从一个不同的侧面描述。复杂上下文信息强调了数据的下面几点：

■ 产品定义。
■ 市场领域。
■ 定价。
■ 包装。
■ 组织结构。
■ 分发。

复杂上下文信息是一些非常有用，同时又非常难以捉摸的信息。它令人难以捉摸是因为它是想当然的，并存在于背景环境中。它非常之基本，以致于没有人会想到要定义它是什么，或怎样随时间变化。然而，在一段长的时期内，复杂上下文信息在理解和解释跨时间段信息方面有着非常重要的作用。

外部上下文信息是那些公司以外的而在理解随时间变化的信息方面起重要作用的信息。外部上下文信息的实例包括：

■ 经济预测：
  • 通货膨胀。
  • 金融。
  • 税务。
  • 经济增长。
■ 政治信息。
■ 竞争信息。
■ 技术进展。
■ 用户人数的统计变动。

外部上下文信息并没有直接指出关于一个公司的任何事情，但指出了公司运转和竞争中所面对的任何事情。考虑到外部上下文信息的及时显示和它随时间的变化，外部上下文信息是很令人感兴趣的。同复杂上下文信息一样，很少会有组织尝试去采集和量度这些信息。外部上下文信息非常之多，也非常之显然，以致于人们总会想当然。它会很快被遗忘，而在需

要时又很难重建。

5.25 捕获和管理上下文信息

复杂上下文信息和外部上下文信息之所以如此难捕获和确定的一个原因就是它们是非结构化的。与简单上下文信息相比较，外部和复杂上下文信息非常杂乱无章。另外的一个因素是上下文信息变化很快。这一刻我们感兴趣的和相关的信息，在下一时刻就可能会变得无关和陈旧。正是外部和复杂上下文信息的这些不断变化和无组织的特点使得这种类型的信息很难系统化。

下面看一看以前的情况。一个人可以认为信息系统行业在过去已经有了上下文信息。过去是通过字典、仓库、目录和图书馆来试图捕获这些信息的，所有这些尝试都是针对简单上下文信息管理的。尽管有这些好想法，但这里存在的一些明显的局限大大降低了其有效性。以往管理简单上下文信息的方法存在以下的缺点：

- 信息的管理是针对信息系统的开发者，而不是最终用户。这样，对于最终用户就会有很少的可视性。因此，最终用户对并不明显的事情几乎没有热情，也不支持。
- 对上下文信息管理的意图是被动的。开发者可以选择用或不用这些上下文信息管理工具，很多人倾向于回避这些工具。
- 对上下文信息管理的意图在很多情况下会从开发计划中删除掉。一个接一个的事例，如应用是在1965年开发的，数据字典是1985年做的，而到了1985年就再没有更多的开发经费了。甚至，那些非常有助于组织和定义简单上下文信息的人早已改行或到了其他公司。
- 对上下文信息的管理意图仅局限于简单上下文信息，没有尝试去捕获和管理外部和复杂上下文信息。

5.26 刷新数据仓库

一旦数据仓库建好以后，注意力就从数据仓库的构造转向每天的操作上。人们发现操作和维护数据仓库的费用很高。数据仓库中，数据量的增长速度比任何人预计的都要快，数据仓库的最终用户——DSS分析人员对数据仓库的大量的、不可预测的应用在管理数据仓库的服务器端引起了竞争，而与数据仓库有关的最大最不可预知的开销是数据从传统数据环境到数据仓库的定期刷新。起先是非常偶然的开销很快就变为一项巨大的开销。

大多数的组织在考虑刷新数据仓库时的第一步一般是直接读取老的传统的数据库。在某些环境下进行某些处理时，刷新只能通过直接读取旧的传统文件。当数据必须从多个不同的传统数据源收集，组成一个整体放入数据仓库中时，直接读取传统数据可能是进行数据仓库刷新的唯一选择。或者当一个事务处理同时使多个传统文件更新时，直接读取传统数据也是刷新唯一的方法。但是，作为一个通用的策略，商场发现重复地直接读取传统数据开销非常大。直接读取传统数据库的开销体现在两个方面。第一个开销是当直接读取传统数据时，传统的DBMS在读取过程中必须是联机的和活动的。对于传统数据库的长时间连续处理的机会是很有限的。为了刷新数据仓库而分配很多的时间是不可取的。第二个方面就是相同的传统数据并无必要地被传送了好几次。当只要 1%或2%的传统数据需要扫描时，刷新活动却得100%地扫描整个文件。这种对资源的浪费在每一次刷新时都会发生。由于这些低效性的存在，

直接地重复读取传统数据来进行刷新是应用非常有限的一种策略。

刷新数据仓库的一个更吸引人的方法是在传统环境中捕捉正在被修改的数据。当传统环境中数据发生变化时，通过捕捉它，就不需要当刷新数据仓库时对传统环境中的表全部扫描。另外，因为数据是在改变时被捕捉的，所以也就不需要传统 DBMS联机来进行长时间的顺序扫描。相反，可以在脱机时处理捕捉到的数据。

在传统操作型环境中，当数据改变时有两种基本的技术来捕获这种数据。一种称为“数据复制”。另一种称为“变化数据捕获”，指将发生了的变化从在联机更新时生成的日志或日志磁带中提取出来。

数据复制要求将要捕获的数据在修改之前标识出来。那么，改变发生时数据就被捕获。设置一个“触发器”来捕获数据的更新活动。数据复制的一个好处是可以有选择地控制捕获处理。事实上只有需要捕获的数据被捕获到。数据复制的另外一个好处是数据的形式“简洁”，定义完善。所被捕获的数据的内容和结构会很好地归档，易于程序员理解。数据复制的缺点是由于要捕获数据，所以产生了许多额外的 I/O操作。同时由于数据仓库不稳定的、总在变化的特性，系统也要不断地注意参数和触发器的定义，使得在数据更新时捕获之。 I/O需求的数量通常不是很小的。另外， I/O操作是在系统高性能运行时进行的，这个时间系统是很难提供I/O的。

第二种对数据仓库环境的高效的刷新方法是通过所谓“变化数据捕获”(CDC)。CDC使用了日志磁带来捕获和确定联机处理时的变化。 CDC需要读取日志或日志磁带。读取一个日志磁带并不是一件小事，这其中有很多阻碍，诸如：

- 日志磁带包含了许多无关数据。
- 日志磁带格式难理解。
- 日志磁带包括跨区记录。
- 日志磁带通常包含的是数据的地址而并非它的值。
- 日志磁带反映了DBMS的特征，并随DBMS的不同而有很大的不同。

CDC的主要障碍就是读取和理解日志磁带。但是，一旦解决了这个问题后，我们就会发现用日志来处理数据仓库刷新的一些吸引人的好处。第一个优点就是高效率。日志磁带处理不像复制处理一样需要附加的 I/O操作。日志磁带不管它是否用于数据仓库的刷新，它都是要写的。因此，日志磁带的 CDC处理不需要增加I/O操作。CDC的第二个好处是，日志磁带会捕获所有的数据更新操作。当对数据仓库或对传统系统环境做一些改变时，用不着再回头重新定义参数。而且日志磁带是你所能得到的最稳定和基本的设备。

这里所描述的刷新技术的发展进程是模仿企业在他们对数据仓库的理解和操作过程中所产生的想法。首先，是从传统数据库中直接读取数据来刷新数据仓库，然后他们尝试数据复制。最后，操作的经济和效率使他们把 CDC当作数据仓库刷新的主要方法。在这个过程中，他们发现一些文件是需要直接读取的。另外还有一些文件适合于用复制的方法。但对于通用目的的数据仓库刷新来说， CDC是一种长期的最终的数据仓库刷新方法。

5.27 小结

为了满足数据仓库处理的需要，应该具有一些技术特征。这些技术特征包括健壮的语言接口、支持复合键码和变长数据，以及如下的一些能力：

- 管理大量数据。
- 管理各种各样介质上的数据。
- 方便的索引和监视数据。
- 大量接口技术。
- 允许程序员将数据直接存放在物理存储设备上。
- 数据的并行存储和访问。
- 有数据仓库的元数据控制。
- 高效地装入数据仓库。
- 有效地使用索引。
- 以压缩方式存储数据。
- 有选择地关闭锁管理。
- 单独索引处理。
- 从大容量存储器迅速恢复。