第1章　决策支持系统的发展

信息系统领域是一个“不成熟”的领域。“不成熟”这个词通常具有消极的含义，因而公开使用这个词不得不多加小心。但是从历史的观点来看的确如此。如果我们将信息处理的历史与其他技术领域的历史进行比较的话，就没有争议了。我们知道古埃及的象形文字主要是当时的帐房先生用来表示所欠法老谷子的多少。当漫步在罗马市区，我们就置身于两千多年前土木工程师所设计的街道与建筑物之间。同样，许多其他的领域也可追溯到远古时代。

因为信息处理领域只是从60年代初期才出现的，所以，历史地来看，信息处理领域是不成熟的。

信息处理领域的年轻性表现之一就是其倾向于面面俱到。有这样一种说法，如果细节都正确了，那么我们就可以坐享其成。这就好象是说，若我们知道如何铺水泥、如何钻孔、如何安装螺母与螺栓，就不必操心桥梁的外型与用途了。如此态度会驱使一个成熟的土木工程师发疯的。

数据仓库的历史是伴随某种发展过程开始的，在此发展过程中，业界中人士所考虑的是投入更大的力量。更大规模的体系结构正在被勾勒出来 —— 在这种体系结构中数据仓库处于中心地位。最好从一种广阔的视角去观察这个体系结构，而不是从某种细节去认识。

1.1　演化

有趣的是，决策支持系统(DSS)处理是一个漫长而复杂的演化进程的结果，而且它仍在继续演化。DSS处理的起源可以追溯到计算机发展的初期。

图1-1表明了从20世纪60年代初期直到1980年的DSS处理的演化进程。在60年代初期，创建运行于主文件上的单个应用是计算领域的主要工作。这些应用的特点表现在报表和程序，常用的是COBOL语言。穿孔卡是当时常用的介质。主文件存放在磁带文件上。磁带适合于廉价地存放大容量数据，但缺点是需要顺序地访问。事实上，我们常说，在磁带文件的一次操作中，100%的记录都要被访问到，但是只有5%或更少的记录是真正需要的。此外，访问整条磁带的文件可能要花去20~30分钟时间，这取决于文件上是什么数据及当前正在做什么处理。

大约在60年代中期，主文件和磁带的使用量迅速膨胀。很快，处处都是主文件。随着主文件数量的增长，出现大量冗余数据。主文件的迅速增长和数据的巨大冗余引出了一些严重问题：

- 需要在更新数据时保持数据的一致性。
- 程序维护的复杂性。
- 开发新程序的复杂性。
- 支持所有主文件需要的硬件数量。

简言之，属于介质本身固有缺陷的主文件的问题成为发展的障碍。如果仍然只用磁带作为存储数据的唯一介质，那么难以想象现在的信息处理领域会是什么样子。

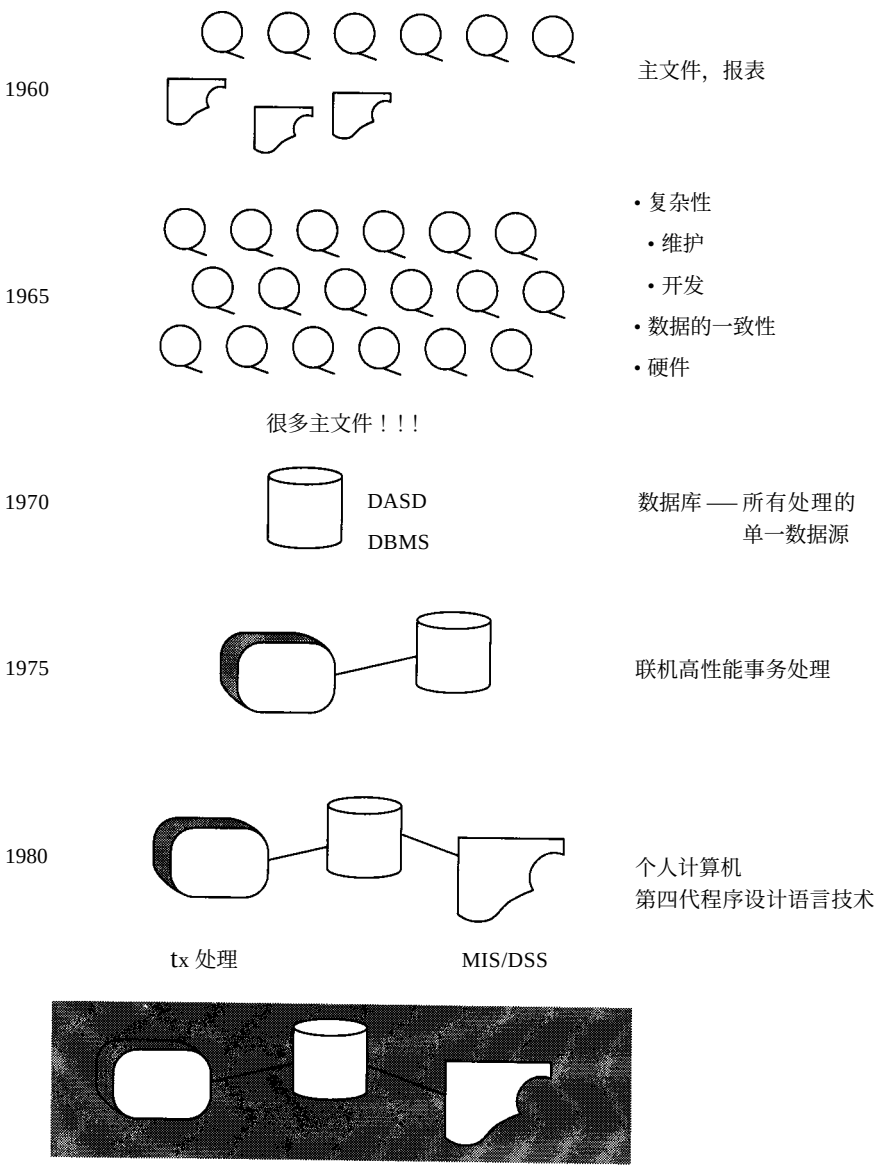


图1-1 体系化环境的早期演化阶段

如果除了磁带文件以外没有别的东西可以存储大量数据，那么世界上将永远不会有大型、快速的预定系统， ATM系统，以及其他系统。而事实上，在除磁带文件之外的种种介质上存储和管理数据的能力，为采用不同的处理方式和更强有力的处理类型开辟了道路，从而把技术人员和商务人员前所未有地聚集到一起。

1.2 直接存取存储设备的产生

到了1970年，一种存储和访问数据的新技术出现了。这就是 20世纪70年代见到的磁盘存

储，或者称之为直接存取存储设备 (DASD)。磁盘存储从根本上不同于磁带存储，因为 DASD上的数据能够直接存取。 DASD就不需要经过第1条记录，第2条记录……，第n条记录，才能得到第n+1条记录。一旦知道了第 n+1条记录的地址，就可以轻而易举地直接访问它。进而，找到第n+1条记录需要的时间比起扫描磁带的时间少得多。事实上，在 DASD上定位记录的时间是以毫秒(ms)来计量的。

随DASD而来的是称之为数据库管理系统 (DBMS)的一种新型系统软件。 DBMS的目的是使程序员在DASD上方便地存储和访问数据。另外， DBMS关心的是在DASD上存储、索引数据等任务。随着DASD和DBMS的出现，解决主文件系统问题的一种技术解决方案应运而生。“数据库”的思想就是DBMS的产物。纵观主文件系统所导致的混乱以及主文件系统累积的大量冗余数据，就不会奇怪为什么把数据库定义为——所有处理工作的单一数据源。

但这一领域的发展并未在 1970年停止。到70年代中期，联机事务处理开始取代数据库。通过终端和合适的软件，技术人员发现更快速地访问数据是可能的——这就开辟了一种全新的视野。采用高性能联机事务处理，计算机可用来完成以前无法完成的工作。当今，计算机可用于建立预定系统、银行柜员系统、工业控制系统，等等。如果仍然滞留在磁带文件系统时代，那么今天我们认为理所当然的大多数系统就不可能存在了。

1.3 个人计算机/第四代编程语言技术

到了80年代，一些更新颖的技术开始涌现出来，比如个人计算机 (PC)和第四代编程语言(4GL)。最终用户开始扮演一种以前无法想象的角色——直接控制数据和系统，这超出了对传统数据处理人员的界定。随着 PC与4GL技术的发展，诞生了一种新思想，即除了高性能联机事务处理之外，对数据可以做更多的处理。管理信息系统 (MIS)——(早期被如此称呼)也可能实现了。 MIS如今称为DSS，是用来产生管理决策的处理过程。以前，数据和技术不能一并用来导出详细的操作型决策。一种新的思想体系开始出现，即一个单一的数据库既能用作操作型的高性能事务处理，同时又用作 DSS分析处理。图 1-1表明了这种单一数据库的范例。

1.4 进入抽取程序

大型联机高性能事务处理问世后不久，就开始出现一种称为“抽取”处理的程序(见图 1-2)，这种程序并不损害已有系统。

抽取程序是所有程序中最简单的程序。它搜索整个文件或数据库，使用某些标准选择合乎限制的数据，并把数据传到其他文件或数据库中。

抽取程序很快就流行起来，并渗透到信息处理环境中。至少有两个理由可以用来解释它为什么受到欢迎：

- 因为用抽取程序能将数据从高性能联机事务处理方式中转移出来，所以在需要总体分析数据时就与联机事务处理性能不发生冲突。
- 当用抽取程序将数据从操作型事务处理范围内移出时，数据的控制方式就发生了转变。最终用户一旦开始控制数据，他(她)就最终“拥有”了这些数据。

由于这些原因(以及其他众多原因)，抽取处理很快就无处不在。到了 90年代已有了很多抽取程序，如图1-3所示。

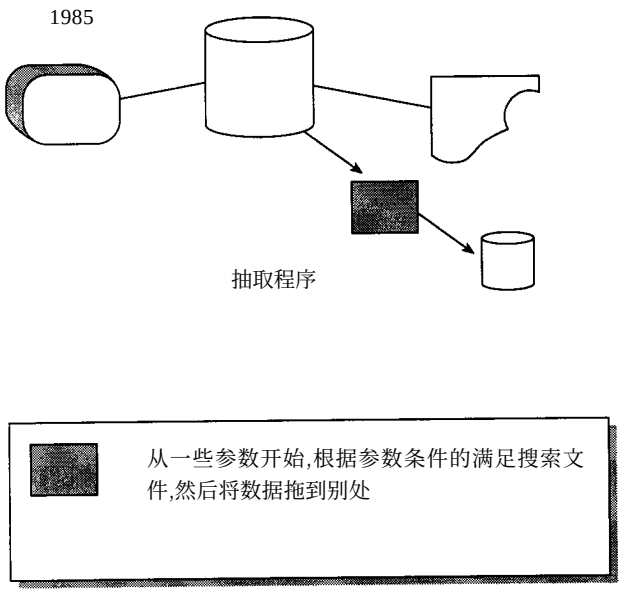


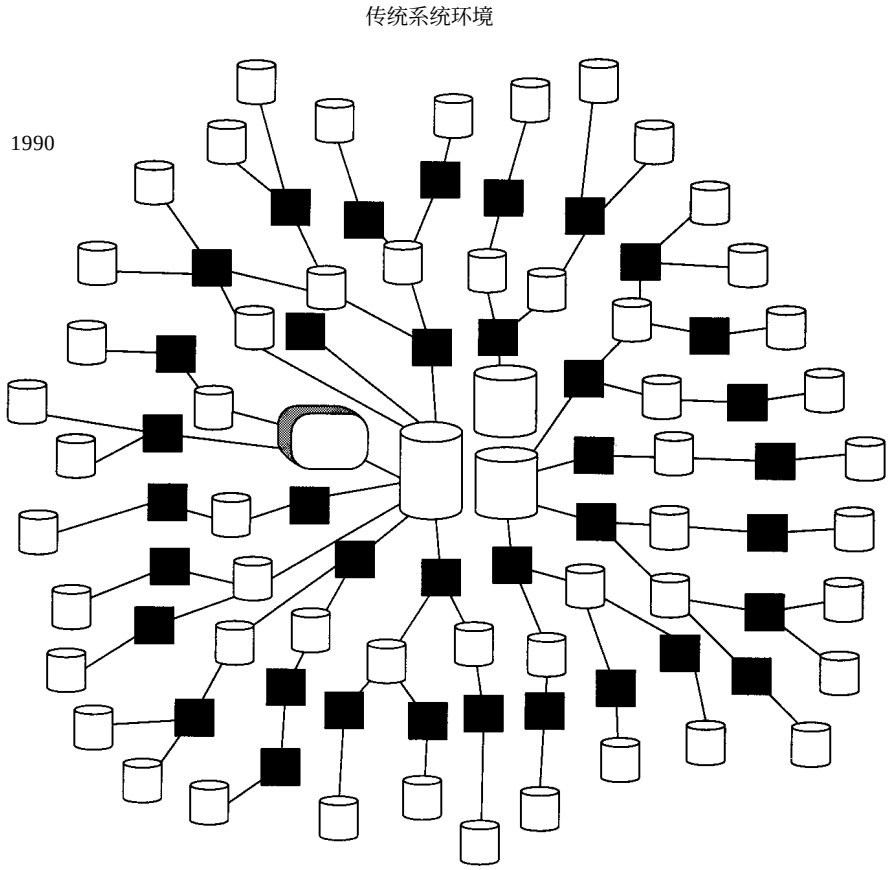
图1-2 抽取处理的特性

1.5 蜘蛛网

图1-3显示抽取处理的蜘蛛网开始形成。起初只是抽取，随后是抽取之上的抽取，接着是在此基础上的再次抽取，如此等等。对于一个大公司，每天进行多达 45 000次的抽取不是没有听说过的。

贯穿于公司或组织的这种抽取处理模式很常见，以致得到一个专有名称。这种由失控的抽取过程产生的结构被称为“ 自然演化体系结构 ”—— 当一个组织以放任自流的态度处理整个硬、软件体系结构时，就会发生这种情况。组织越庞大，越成熟，自然演化体系结构问题就变得越严重。

从总体上看，抽取程序形成了蜘蛛网，这正是自然演化 (或“ 传统系统 ”)体系结构的另一



自然演化的体系结构(或称为“蜘蛛网”)

图1-3 抽取处理广泛采用必然是件好事情

个名称。

1.6 自然演化体系结构的问题

与自然演化体系结构相关联的困难到底是什么呢？问题很多，主要有：

- 数据可信性。
- 生产率。
- 数据转化为信息的不可行性。

1.6.1 数据缺乏可信性

以上问题之首是数据缺乏可信性，如图 1-4所示。两个部门向管理者呈送报表，一个部门说业绩下降了15%，另一个部门说业绩上升了 10%。两个部门的结论不但不吻合，而且相去甚远。另外，两个部门的工作也很难协调。除非十分细致地编制了文档，否则对任何应用目的而言，协调是不可能的。

当管理者收到这两张报表时，他们不知如何是好。管理者面临着根据政策和个人意志做决定的状况。这是在自然演化体系结构中可信性危机的一个实例。

这种危机很广泛存在，而且是可以预想得到的，为什么？有五个理由可以解释危机的可预测性(见图1-4)，它们是：

- 数据无时基。
- 数据算法上的差异。
- 抽取的多层次。
- 外部数据问题。
- 无起始公共数据源。

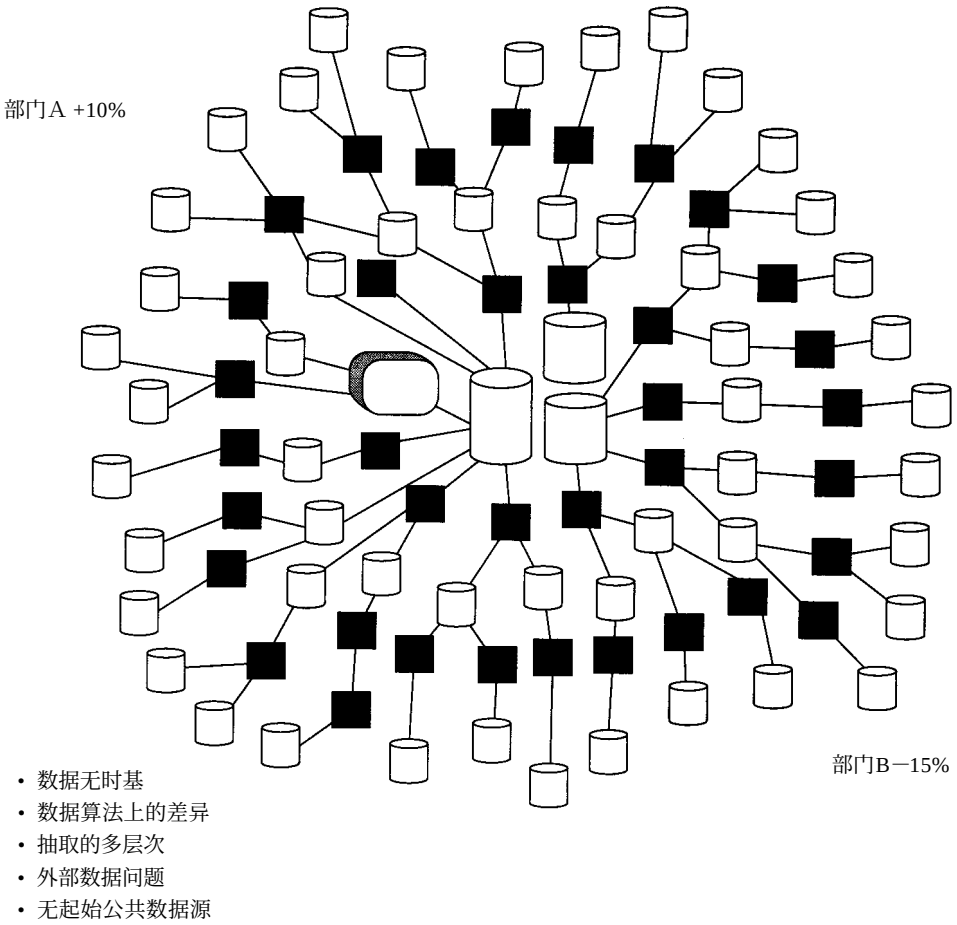


图1-4 在自然演化体系结构中缺乏数据可信性

图1-5显示一个部门在星期日晚上提取分析所需的数据，而另一个进行分析的部门在星期三下午就抽取了数据。有任何理由相信对某一天抽取的数据样本进行的分析与对另一天抽取的数据样本进行的分析可能相同吗？当然不能！公司内的数据总是在变的。任何在不同时刻抽取出来用于分析的数据集之间只是大致相同。

在自然演化体系结构中，数据可信性危机具有可预见性的第二个理由是算法上的差异。

比如，一个部门选择所有的老帐号作分析。而另一个部门选择所有大帐号作分析。在有老帐号的顾客和有大帐号的顾客之间存在必要的相关性吗？可能没有。那么分析结果大相径庭就没有什么可大惊小怪的了。

可信性危机可预见性的第三个理由是前两个理由的扩展。每次新的抽取结束，因为时间和算法上的差异，抽取结果就可能出现差异。对一个公司而言，从数据进入公司系统到决策者准备好分析所采用的数据，经过八层或九层抽取不是罕见的。

缺乏可信性的第四个理由是由外部数据引起的问题。利用当今在 PC层次上的技术很容易从外部数据源取得数据。在图 1-5所示的例子中，一个分析人员从《华尔街日报》取得数据放入分析流中，而另一个分析人员从《商业周刊》中取得数据。分析人员在取得数据之时所做的第一件事就是从大量外部数据中抽出所需要的部分。数据一旦进入 PC，就不再属于《华尔街日报》了，而简单地变成了可能出自于任何数据源的普通数据。

并且，从《华尔街日报》取得数据的分析人员对从《商业周刊》中取得的数据是一无所知的，反之亦然。这就不足为怪，外部数据导致自然演化体系结构中的数据缺乏可信性。

导致数据缺乏可信性的最后一个因素是通常没有一个公共的起始数据源。部门 A的分析工

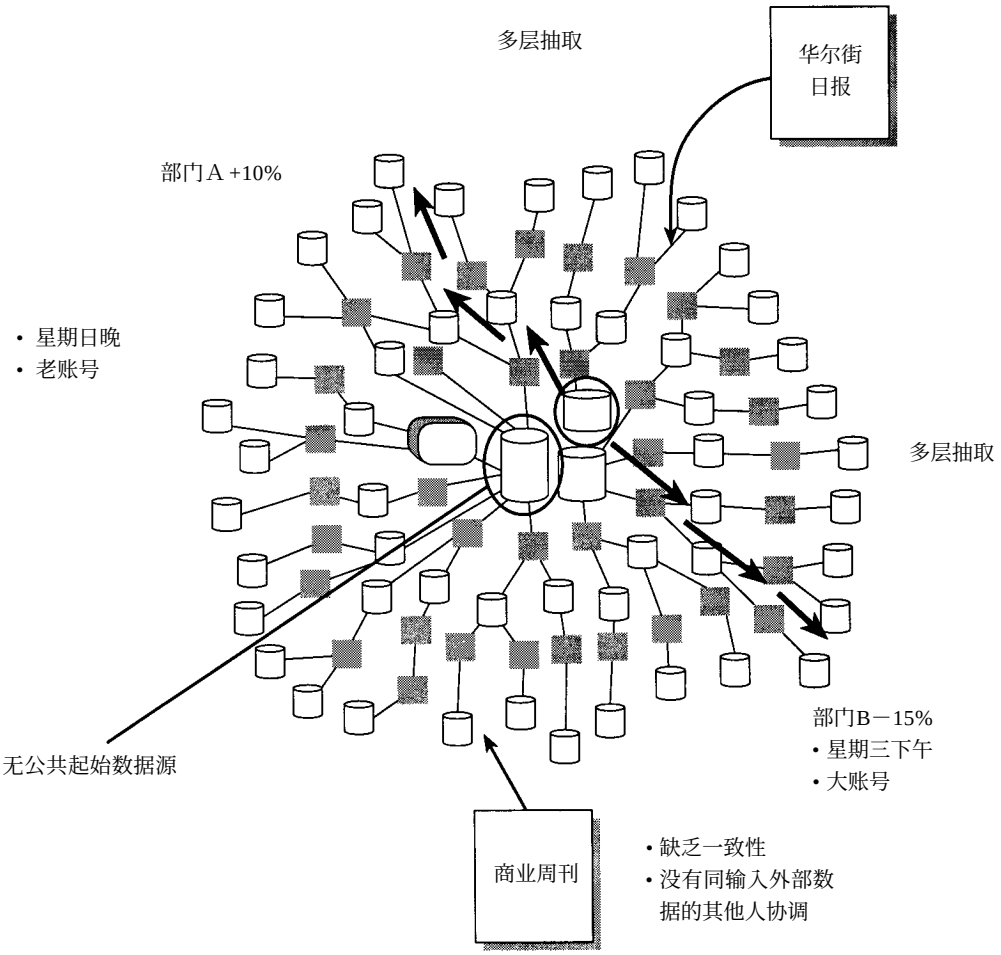


图1-5 自然演化体系结构中可信性危机可预见性的原因

作源于文件XYZ，部门B的分析工作源于数据库ABC。不论文件XYZ与数据库ABC之间关系怎样，都不存在数据同步或数据共享。

有了这些理由，在每一个企业或机构中，如果允许软件、硬件和数据的体系结构自然地演化为蜘蛛网，那么这种企业或机构中正酝酿着可信性危机就不足为奇了。

1.6.2 生产率问题

但是数据可信性还不是自然演化体系结构中的唯一的主要问题。在自然演化体系结构中，当需要查询机构范围内的数据时，生产率(或者说生产率低)是不可预测的。

设想一个机构在商业上已运营了一段时间,并且已经建立起了大型数据集合,如图1-6顶部所示。

管理者期望用数年来积累的数据集合和众多文件生成一张企业报表，接受了该任务的设计者为产生企业报表决定做三件事：

- 定位报表需要的数据并分析数据。
- 为报表编辑数据。
- 为完成以上工作，召集程序员/分析员。

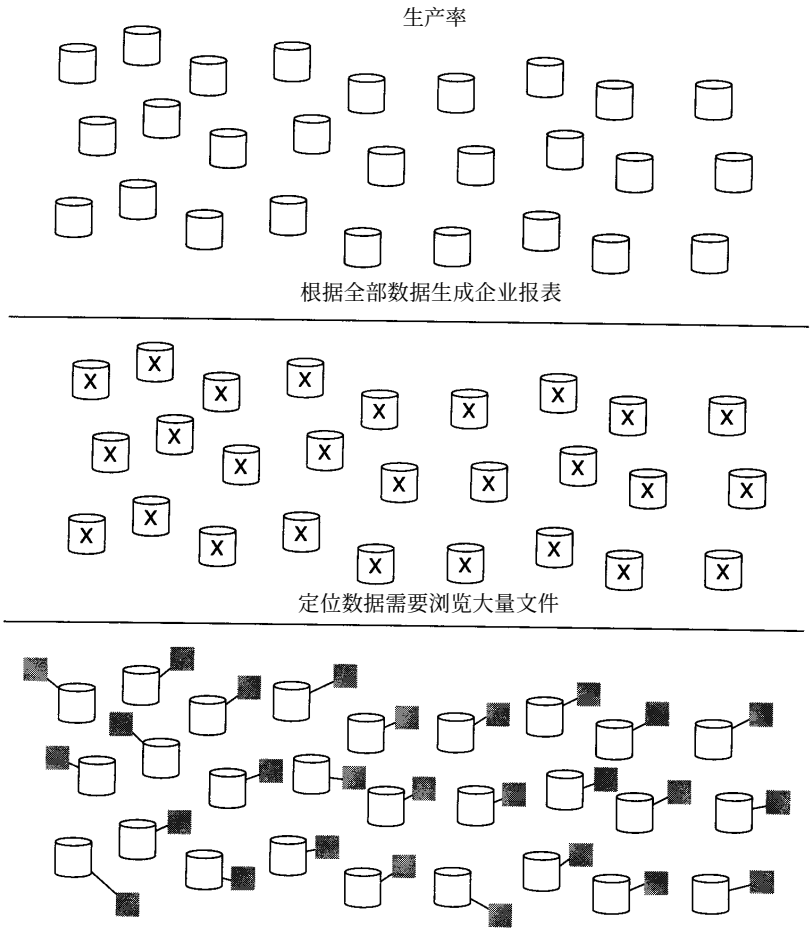


图1-6 自然演化体系结构不利于生产率的提高

要进行数据定位，必须分析很多文件和数据布局。而且，存在一些复杂因素：一个文件有一个被称为BALANCE的元素；而另一个文件有一个同名的元素。但是两个元素的意义相去甚远。另一种情况是，一个数据库有一个被称为 CURRBAL的文件，而在另一个数据集中存在一个称为INVLEVEL的文件，此文件恰好与 CURRBAL相同。这就不得不遍历每一个数据，不只按名遍历，而且按数据的定义和计算要求遍历，这是一个十分乏味的过程。但是如果要生成企业报表，这个过程就必不可少。除非对数据进行分析和“合理化”处理，否则报表最终将产生更大的混乱，就像苹果和桔子混在一起一样。

一旦数据定位完成，下一个任务就是编辑数据。当然，为从众多的数据源中取得数据而必须编制的程序相当简单。但是以下事实使这种工作复杂化：

- 要写的程序很多。
- 每个程序必须是定制的。
- 程序涵盖了公司拥有的所有技术。

简言之，即使必须写报表生成程序，并且看起来并不难，但是由于上面的因素，为生成企业报表检索数据仍是个十分乏味的工作。

在一个面临以上这些问题的公司里，分析人员估算过要完成这项工作需要很长时间，如图1-7所示。

如果设计者只要求了两三个人月资源的工作量，那么生成这样一个报表可能不需要管理者过

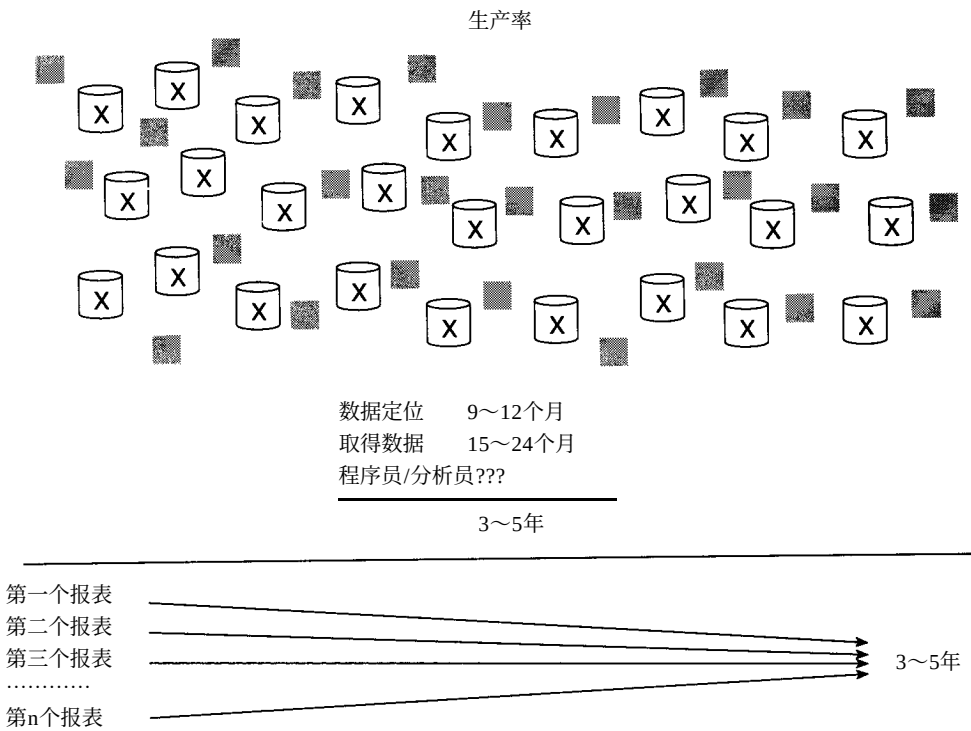


图1-7 在编写第一张报表时,对后继报表的需求还不清楚

多的关注。但如果一个分析员需要很多资源，管理者就必须将这个请求与其他对资源的请求一并考虑，并且必须为这些请求制定优先级。因此，分析员估算了漫长的时间以生成期望的报表。

如果时间代价是一次性的，那么为生成报表花费大量的资源也是可取的。换句话说，如果生成第一份企业报表需要大量资源，生成所有后继报表可以建立在第一份报表基础之上，那么不妨为生成第一份报表付出一些代价。但是事实并非如此。

除非事先知道未来的企业报表需求，并且除非这些需求影响到第一张企业报表的建造，每个新的企业报表总要花费同前面差不多大的代价。换句话说，第一张企业报表非常不可能为将来别的企业报表需求做出什么贡献。

因此在公司环境中，生产率是自然演化体系结构和传统系统所面临的一个主要问题。

1.6.3 从数据到信息

看来生产率和可信性还不是问题的全部，自然演化体系结构还存在着另一个主要缺陷——从数据到信息转化的不可行性。乍一看来，从数据转化成信息的思想是一个缺少实际意义的虚无概念，但是事实完全不是这样。

考虑下面对信息的需求，这种需求在银行环境中很典型：“今年的帐号活动同过去五年中各个年份有什么不同？”

图1-8显示了对信息的这种需求。

DSS分析员试图满足对信息的需求，而在此过程中发现的第一件事情，就是到现存的系统

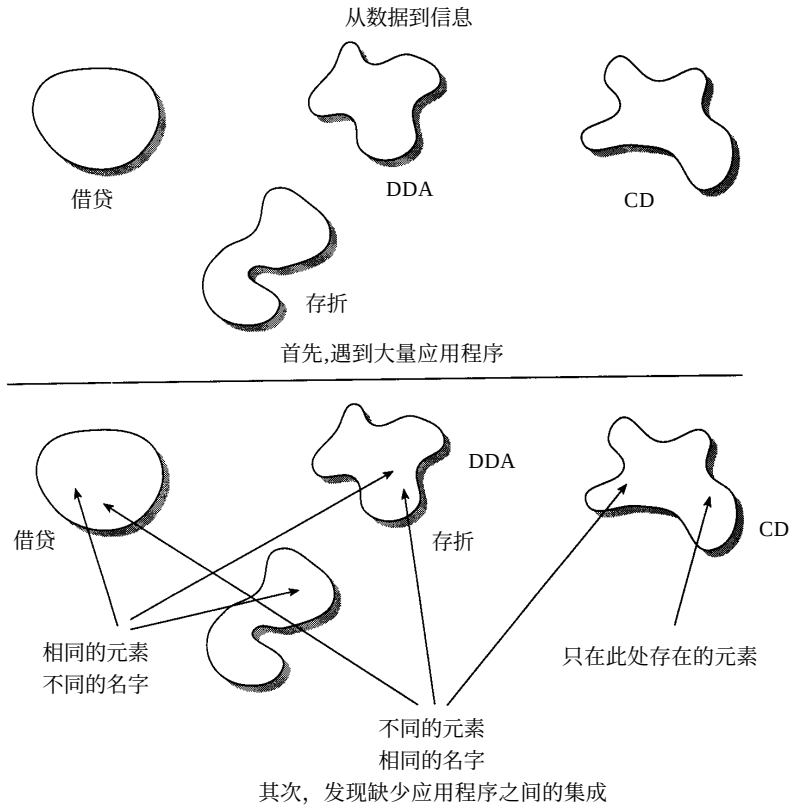


图1-8 “这个金融机构今年的账号活动同过去五年中各个年份有什么不同？”

中寻求必要的数据大概是最糟的举动。在传统环境中存在着 DSS分析员将要遇到的很多应用程序。

试图发现一个帐号存在哪些相关数据是相当困难的。系统中有储蓄应用程序、借贷应用程序、DDA(活期存款记帐)应用程序和信用应用程序。从它们当中抽取公共信息几乎不可能。设计这些应用程序时从未考虑数据集成，对 DSS分析员来说对它们进行解释并不比任何其他人更容易。

但是，集成化并非分析人员在试图满足信息需求过程中遇到的唯一困难。第二个主要障碍是在应用程序中没有存储足够的历史数据以满足 DSS分析员的需求。

图1-9显示借贷部门拥有长达两年的有用数据，存折处理程序有长达一年的数据， DDA应用程序有60天的数据，CD处理程序有18个月的数据。建造这些应用程序是用来满足当时收支处理的需要(自然是足够的！)。设计时从未考虑过保存这些历史数据以满足 DSS分析的需求。那么不用说，对 DSS分析来说，求助于现存系统不是明智的选择。但是除了这些又能求助于什么呢？

在自然演化体系结构中建立起来的系统对信息需求的支持确实是不充分的，因为它们缺乏集成性，以及在分析性处理需要的时间期限上和在蜘蛛网环境中应用程序的可用时间期限上存在差异。

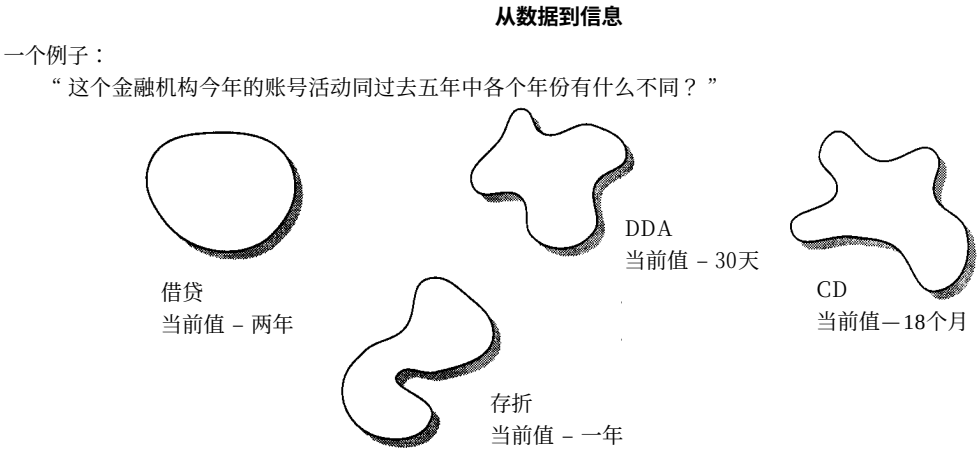


图1-9 现有的应用程序的确没有将数据转化成信息所需的历史数据

1.6.4 方法的变迁

自然演化体系结构的存在方式(今天大多数商场采取这种模式)确实不足以满足明天的需要。体系结构需要转变，体系化的数据仓库环境应该在变化了的体系结构上建造。

体系结构设计环境的核心是意识到存在着两种基本数据：原始数据和导出数据。图 1-10显示了原始数据与导出数据之间的一些主要区别。

原始数据是公司每天操作运行所用的细节性数据，导出数据是统计出来的或计算出来的满足公司管理者需要的数据。原始数据可以更新，导出数据不可以更新。原始数据主要是当前值数据，导出数据通常为历史数据。原始数据由以重复方式运行的过程操作，导出数据由非重复地启发式地运行的程序操作。操作型数据是原始的， DSS数据是导出的。原始数据支

持日常工作，导出数据则支持管理工作。

因此，在原始数据与导出数据之间存在着本质区别。奇怪的是，在信息处理界曾经考虑过将原始数据和导出数据都放入一个单一数据库中。

方法的变迁

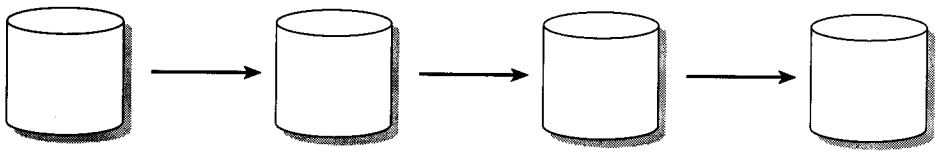
原始数据/操作型数据	导出数据/DSS数据
• 面向应用	• 面向主题
• 详细的	• 综合的,或提炼的
• 在存取瞬间是准确的	• 代表过去的数据
• 为日常工作服务	• 为管理者服务
• 可更新	• 不更新
• 重复运行	• 启发式运行
• 处理需求事先可知	• 处理需求事先不知道
• 生命周期符合SDLC	• 完全不同的生命周期
• 对性能要求高	• 对性能要求宽松
• 一个时刻存取一个单元	• 一个时刻存取一个集合
• 事务处理驱动	• 分析处理驱动
• 更新控制主要涉及所有权	• 无更新控制问题
• 高可用性	• 松弛的可用性
• 整体管理	• 以子集管理
• 非冗余性	• 时常有冗余
• 静态结构；可变的内容	• 结构灵活
• 一次处理数据量小	• 一次处理数据量大
• 支持日常操作	• 支持管理需求
• 访问的高可能性	• 访问的低可能性或适度可能性

图1-10 在体系结构设计环境中数据整体思想的变化

1.7 体系结构设计环境

由于原始数据和导出数据的不同而导致的数据分离的自然扩展过程如图 1-11所示。

体系结构层次



操作型	原子/数据仓库	部门	个体
• 细节的	• 大部分粒度	• 领域狭隘	• 暂时的
• 逐日	• 随时间变化的	• 一些导出数据；一些原始数据	• 为特定目的的
• 当前值的	• 集成的	• 典型的部门	• 启发式的
• 访问的高可能性	• 面向主题	• 财务	• 非重复的
• 面向应用	• 一些汇总	• 市场	• 基于PC和工作站的
		• 工程	
		• 保险	
		• 制造	

图1-11 尽管看起来不太明显,但在体系结构设计环境中存在的数据冗余很少

1.7.1 体系结构设计环境的层次⊖

在体系结构设计环境中有四个层次——操作层、原子或数据仓库层、部门层、个体层。数据操作层只保存原始数据并且服务于高性能事务处理领域。数据仓库层存储不更新的原始数据，此外一些导出数据也在此存放。数据的部门层几乎只存放导出数据。在数据个体层中完成大多数启发式分析。

对这种体系结构的直接反应是在体系结构设计环境中存在大量冗余数据。事实上完全不是这样，尽管数据冗余很少这一点乍看起来不明显。相反，在蜘蛛网中倒是存在着大量的数据冗余。

考察贯穿这种体系结构的数据的简单实例，如图 1-12所示。在操作层中存在一个顾客记录J.Jones。在操作层的记录是包含当前值的数据记录。要了解顾客的当前的情况，就访问操作层的记录。当然，如果关于 J.Jones的信息变化了，那么操作层的记录将随之变化成正确的新数据。

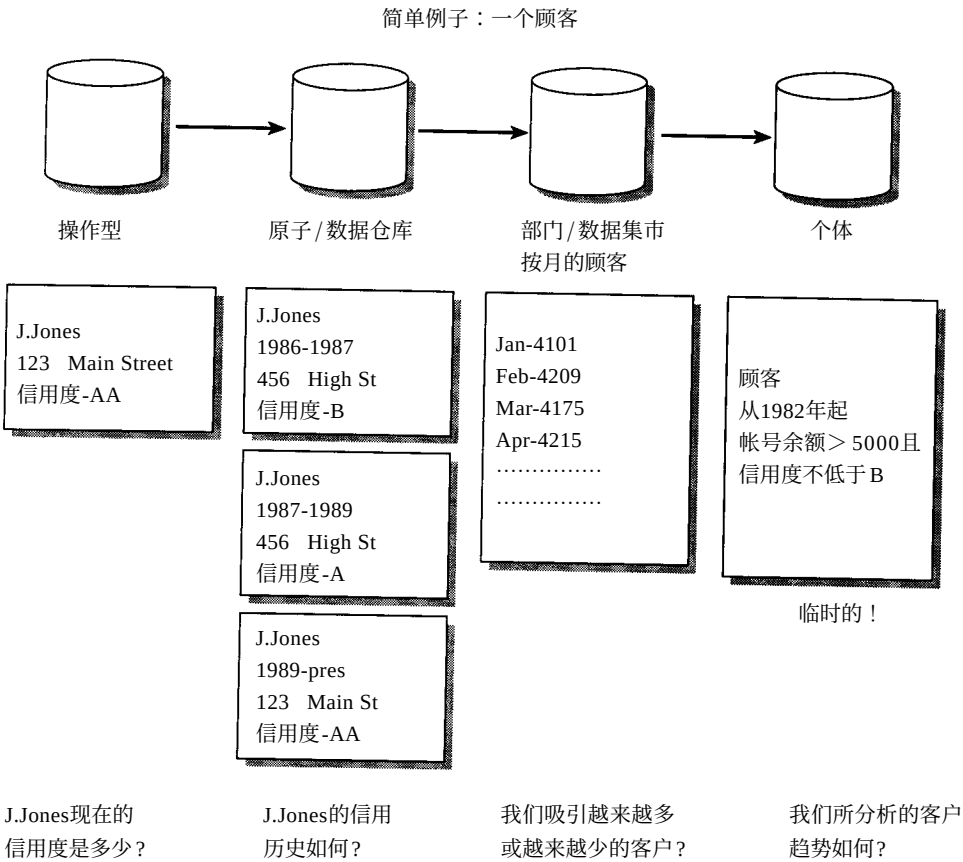


图1-12 可用不同数据层次进行查询的不同类型

在数据仓库环境中可以找到几条有关 J.Jones的记录，这些记录反映了 J.Jones的历史信息。比如，要发现J.Jones去年住在什么地方，就搜索数据仓库中的记录。在数据仓库环境中的数据与在操作型环境中的数据之间无重叠。如果 J.Jones的地址发生了变化，那么在数据仓库中

⊖ 本节标题是译者添加的。

将产生一个记录，这个记录反映了从什么时间到什么时间 J.Jones住在哪里。注意数据仓库中的记录无重叠，并且在数据仓库中存在与每个记录相关联的时间元素。

部门环境包括对一个公司中不同地区的部门有用的信息。部门环境包括市场部门数据库，财务部门数据库，保险部门数据库，等等。所有部门的数据源都是数据仓库。部门层常被称为“ 数据集市层 ”、OLAP层或“ 多维DBMS ”层。

部门层典型数据是月度顾客文件。在此文件中是一张所有顾客的分类列表。 J.Jones每月都出现在这个汇总当中。可以进一步考虑将记帐信息作为冗余的一种形式。

最后的数据层是个体层。个体层数据常常是暂时的、小规模的。在个体层要做很多启发式分析。通常，个体层数据被认为是由 PC机支持的数据。高级管理人员信息系统 (EIS)处理主要运行在个体层上。

1.7.2 集成

在图1-12中没有显示出来的体系结构设计环境的一个重要方面是发生于整个体系结构中的数据集成。当数据从操作型环境传向数据仓库环境时，数据就被集成，如图 1-13所示。

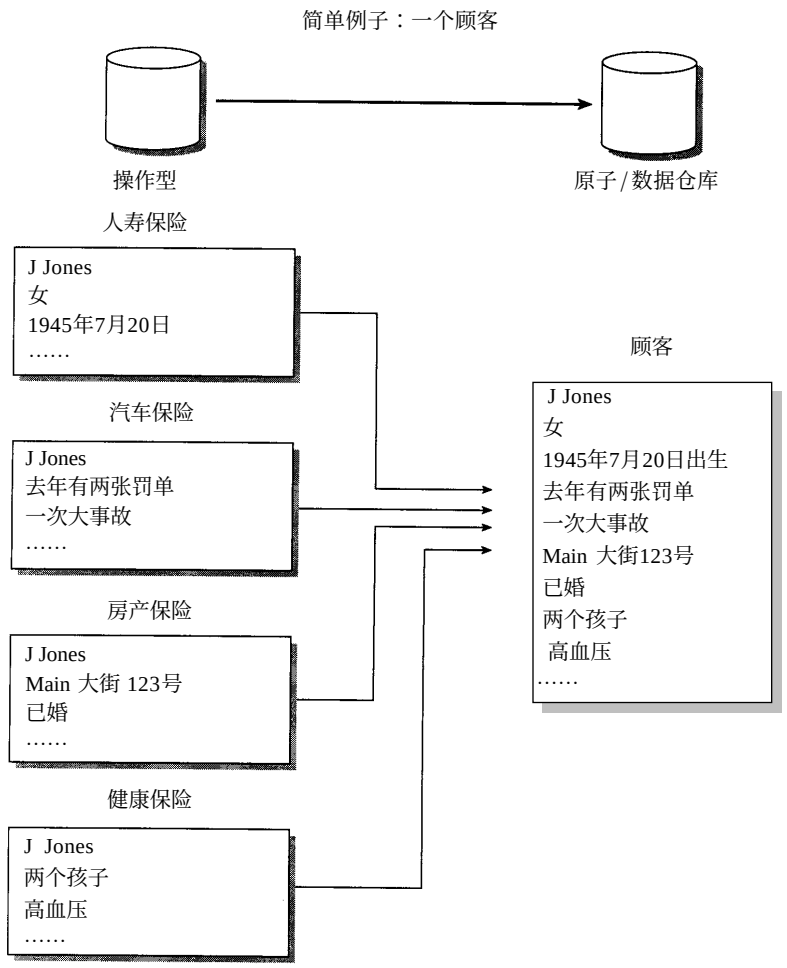


图1-13 数据在从操作型环境转移到数据仓库环境的同时进行集成

把数据从操作型环境载入到数据仓库环境时，如果不进行集成就没有意义。如果数据以一种非集成状态到达数据仓库，它就不能被用来支持数据的企业视图。数据的企业视图是体系结构设计环境的本质之一。

1.8 用户是谁

数据仓库的用户称为DSS分析员。他首先是个商务人员，其次才是技术人员。 DSS分析员的主要工作是定义和发现在企业决策中使用的信息。

了解DSS分析员的想法及他们对使用数据仓库的理解是很重要的。 DSS分析员有一种想法，即“给我看一下我说我想要的东西，然后我告诉你我真正想要什么。”换句话说，DSS分析员在发现模式下工作。直到看到报表或屏幕上的数据时，他们才开始探讨是否有必要进行DSS分析。

DSS分析员的态度之所以重要的理由如下：

- 它是合理的。
- 它是广泛的。
- 它对数据仓库的开发方式和系统怎样使用被开发的数据仓库有深远的影响。

传统的系统开发生命周期(SDLC)不适用于DSS分析领域。SDLC假设在设计之初需求是已知的(或至少是可以被发现的)。但是，在DSS分析员眼中，在DSS开发生命周期的最后才发现真正的需求。与数据仓库相关联的是一种完全不同的开发生命周期。

1.9 开发生命周期

操作型数据通常是非集成的，而数据仓库数据必须是集成的。在数据和处理的操作层和数据和处理的数据仓库层之间，存在其他几个重要区别。操作层和数据仓库层之间内在的区别关系到系统开发生命周期，如图1-14所示。

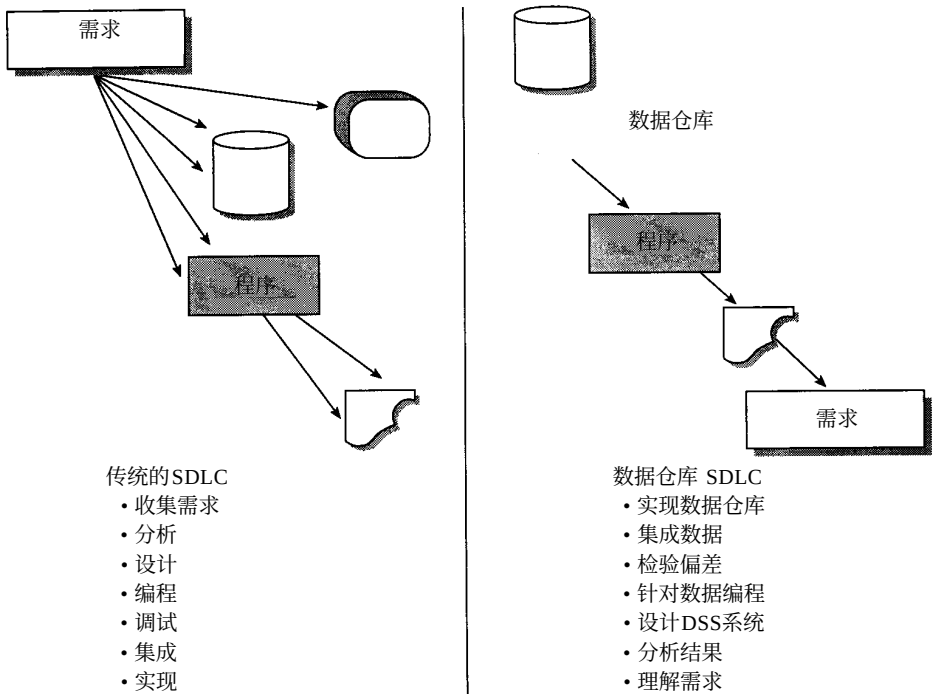


图1-14 数据仓库环境下的系统开发生命周期与传统的SDLC几乎完全相反

图1-14显示传统的系统开发生命周期支持操作型环境。数据仓库运行于一个与之完全不同的生命周期下，有时称为CLDS(与SDLC顺序相反)。传统的SDLC是需求驱动的。为建立系统，你必须首先理解需求，然后进入到设计和开发阶段。 CLDS几乎刚好相反。CLDS由数据开始，一旦数据到手就集成数据。然后，如果数据有偏差，就检验看看数据存在什么偏差。再针对数据写程序，分析程序执行结果。最后，系统需求才得到了理解。

CLDS是典型的数据驱动开发生命周期，而 SDLC是典型的需求驱动开发生命周期。试图采用不适当的开发工具和技术只会导致浪费和混乱。比如， CASE领域是由需求驱动分析所支配的。试图将CASE工具和技术用于数据仓库领域是不明智的，反之亦然。

1.10 硬件利用模式

操作型环境和数据仓库环境之间的还有另一个主要差别，即在各自环境中硬件利用模式也不同，如图1-15所示。

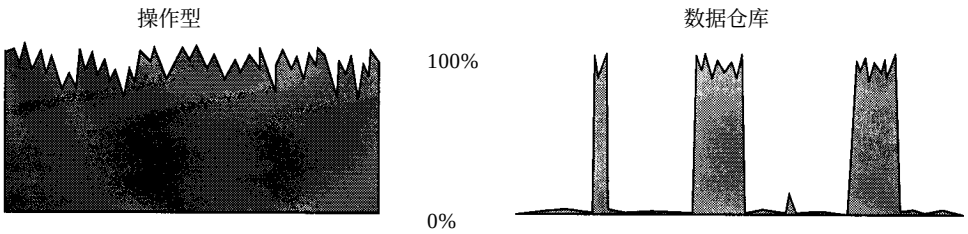


图1-15 不同环境下不同的硬件利用模式

图1-15左面显示操作型处理的典型的硬件利用模式。在操作型处理中有波峰和波谷，但总归存在相当稳定的利用模式。

数据仓库环境中具有根本不同的硬件利用模式 (如图的右部所示)，即利用的二元模式。要么利用全部硬件，要么根本不用硬件。估算数据仓库环境中的硬件平均利用率是没有意义的。

这种根本区别也表明同时在同一台机器上把两种环境混在一起为什么不可行。要么针对操作型处理优化机器，要么针对数据仓库处理优化机器。但是你不可能同时在同一台设备上两者都作到。

1.11 建立重建工程的舞台

从生产环境转变到体系结构设计的数据仓库环境过程中有一个非常有用的副作用，尽管它不是直接的。图1-16显示了这种过程。

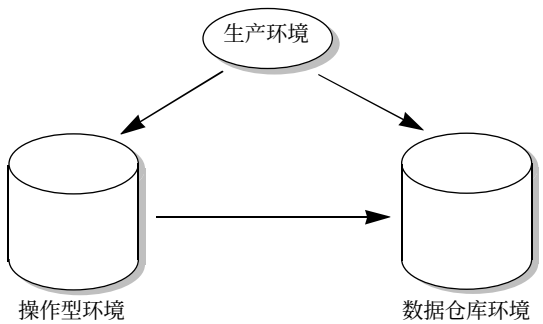


图1-16 从传统系统环境向体系结构设计的以数据仓库为中心的环境转变

在图1-16中，在生产环境中发生一种转变。第一个作用是从生产环境中移走大量数据——大部分是档案数据。移走大量数据在许多方面具有好的效果，包括如下几条：

- 生产环境更易于纠错。
- 生产环境更易于重构。
- 生产环境更易于监控。
- 生产环境更易于索引。

简言之，仅仅是移走可观数目的数据就可使生产环境更具有可塑性。

另一个作用是从生产环境中移走信息性处理。信息性处理采取报表、屏幕显示、抽取等形式。信息处理的特点是不停地变化。商业形势变化、机构变化、管理变化、财务状况变化，等等。这些变化中的任何一个都对综合与信息性处理产生影响。当信息性处理处在生产传统环境中时，维护起来无休无止。事实上，在生产环境中，大多数所谓的维护就是贯穿于正常的信息变化周期中的信息性处理。通过把大多数信息性处理移到数据仓库中，生产环境中的维护负担将大大减轻。图1-17显示从生产环境中移走大量数据和信息性处理的效果。

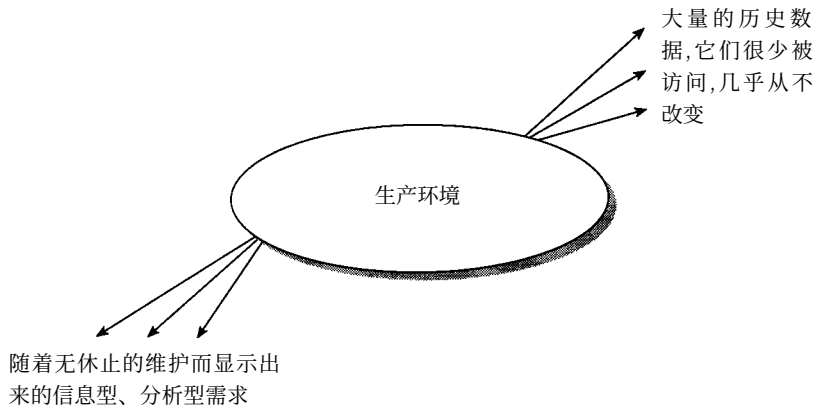


图1-17 从生产环境中移走不需要的数据和信息型需求——建造数据仓库的效果

一旦生产环境经历转变到以数据仓库为中心的体系结构设计环境的变化，生产环境就正好适合于重建工程。因为此时生产环境：

- 更小。
- 更简单。
- 更集中。

总之，一个公司要想成功地重建生产系统和修整遗留系统，最重要的步骤是首先建立数据仓库环境。

1.12 监控数据仓库环境

通常，数据仓库环境中两种受监控的操作成分是存储于数据仓库中的数据和数据的使用。监控数据仓库环境中的数据是管理数据仓库环境的基本能力。通过监控数据仓库环境中的数据能取得一些重要信息，包括：

- 识别发生了什么增长，增长发生在什么地方，增长以什么速率发生。

- 识别正在使用什么数据。
- 估算最终用户得到的响应时间。
- 确定谁在实际使用数据仓库。
- 说明正在使用数据仓库中的多少数据。
- 精确指出数据仓库何时被使用。
- 识别数据仓库的多少数据被使用。
- 检查使用数据仓库的层次。

当数据体系结构设计者不知道这些问题的答案时，有效的管理运行中的数据仓库环境是不可能的。

监控数据仓库真的有用吗？只要考虑一下知道“在数据仓库中什么数据正在被使用”有多么重要就明白了。数据仓库的特性是不停地增长。历史数据不停地加入数据仓库，汇总数据也不停地加入，新的抽取流在创建。同时数据仓库驻留的存储和处理技术并不昂贵。有时会问这样的问题：“为什么所有这些数据要积累起来？真有人用这些数据吗？”显然，不论是否有数据仓库的合法用户，在数据仓库正常运行期间，一旦数据放入数据仓库，数据仓库的开销就会增长。

只要数据体系结构设计者没有办法确定如何使用数据仓库中的数据，那么除了不断购买新的计算机资源之外就别无选择了——购买更多的存储设备、更多的处理器，等等。但是通过监控数据仓库中数据的使用，就有机会把不用的数据移到其他介质上。当数据体系结构设计者发现当前一些数据没有使用，就把这种数据移到不昂贵的介质上，这是合适的做法。通过监控数据仓库中数据的使用和活动情况，数据体系结构设计者能确定现在什么数据不在使用，就能进行转移。监控数据仓库环境中的数据及活动会得到非常实在的和迅速的回报。

在数据监控处理期间，可以建立数据的各种概要文件包括：

- 数据仓库中所有表的目录。
- 这些表的内容。
- 数据仓库中表的增长。
- 用于访问表的可用的索引目录。
- 汇总表和汇总源的目录。

监控数据仓库活动的需求通过下列问题来说明：

- 什么数据正在被访问？
  - 什么时候访问？
  - 由谁访问？
  - 访问频率怎样？
  - 在什么细节层次？
- 对请求的响应时间是什么？
- 在一天的什么时间提出请求？
- 请求多大的数据量？
- 请求是被终止的还是正常结束的？

DSS环境中响应时间的概念与联机事务处理 (OLTP)环境中响应时间的概念大不相同。在OLTP环境中，响应时间总是十分重要的。在 OLTP中当响应时间太长时，业务情况很快就开始变糟。在DSS环境中不存在这种关系。在 DSS数据仓库环境中，响应时间总是宽松的。在

DSS中响应时间不是决定性的，相应地，在DSS数据仓库环境中响应时间以分钟和小时计，在某些情况下以天计。

但是，在DSS数据仓库环境中响应时间很宽松并不意味着响应时间不重要。在DSS数据仓库环境中，最终用户重复地进行开发工作。这意味着下一个层次的开发依赖于当前分析中所得到的结果。如果最终用户进行重复分析，并且周转时间只有10分钟，那么他(她)将比周转时间多达24小时的情况具有更高的生产率。因此，在DSS环境中，响应时间与生产率之间存在十分密切的关系。DSS环境中响应时间只是非关键性的，并不意味着它无关紧要。

测量DSS环境中的响应时间是管理DSS的第一步。仅此一点，监控DSS活动就是必须进行的非常重要的步骤。

在DSS环境中响应时间度量的问题之一是“要度量什么？”在OLTP环境中，要度量什么的答案是显而易见的。发出请求、接受服务，然后返回给最终用户。在OLTP环境中响应时间的度量是从请求被提交的时刻算起到结果被返回的时间。但是DSS数据仓库环境不同于OLTP环境，因为没有明确的度量数据返回的时间。在DSS数据仓库环境中，经常有作为查询结果返回的大量数据。其中一些数据在某一时间返回，另一些数据在晚些时候返回。定义数据仓库环境中数据返回时间不是件容易的事。一种解释是数据第一个返回的时间；另一种解释是数据最后一个返回的时间。对度量响应时间还有很多其他可能的解释。 DSS数据仓库活动监控程序必须能提供多种不同的解释。

在数据仓库环境中使用监控程序的一个根本问题是在哪儿进行监控。能进行监控工作的一个地方是最终用户终端。这是做监控工作的一个方便位置，因为这里有很多空闲的机器周期，并且在这里进行监控工作对系统性能只有很小的影响。但是，在最终用户终端监控系统意味着每个被监控的终端需要自己的管理员。在一个单独的DSS网络中，可能有多达10 000台终端，试图管理每个终端的监控工作几乎是不可能的。

另一个途径是在服务器层次对DSS系统进行监控。在查询已形式化并且已经传给管理数据仓库的服务器后，才开始进行监控。毫无疑问，在此处管理监控程序要容易得多。但是存在系统范围内性能下降的很大可能性。因为监控程序使用服务器资源，监控程序影响整个DSS数据仓库环境的工作性能。监控程序的位置是必须仔细考虑的重要问题，要在管理的方便性和降低性能之间进行权衡。

监控程序最有效的用途之一是能够将今天的结果与每天平均的结果进行比较。发现异常时，能够问一句“今天与每天平均的结果有什么不同？”这通常是有好处的。在大多数情况下会发现性能变化不象想象中那么坏。但为了做这样的比较，需要一个“每天平均概况”。“每天平均概况”包括了DSS环境中描述一天情况的各种标准的重要度量指标。一旦对当天的情况进行了度量，就可以与每天平均概况进行比较。

当然，每天平均值总是随时在变化的。定期地追踪这些变化，使得对长期系统趋势能够进行度量将是有意义的。

1.13 小结

本章讨论了体系结构问题，数据仓库适合于采用这种体系结构。这种体系结构的演化贯穿于信息处理不同阶段的整个历史。在这种体系结构中有四个数据及处理层次——操作层，数据仓库层，部门层和个体层。