数据仓库构建之行为模式分析*

蒋 彬 余肖生 王东娟 姜艳静 赵美林
(三峡大学计算机与信息学院,湖北 宜昌 443002)

摘 要 数据仓库ETL程序的构建与维护在数据仓库的建设与使用中占有举足轻重的地位。传统上的ETL之构建方法分为水平式和垂直式。实践证明,这两种传统方式都没能有效地解决ETL构建过程中生产效率、软件质量、文档质量等方面的问题。通过数据仓库构建的行为模式分析,发现数据仓库构建中可以区分出域通用知识和特殊对象知识。其中,域通用知识是可以重复使用的,在数据仓库构建过程中只需处理一次。据此,ETL程序的生产效率、软件质量、文档质量等可以得到实质性提高。

关键词 数据仓库构建,ETL,行为模式
中图分类号 TP311

1 引言

数据仓库ETL(提取、转换、加载)机制是一组从源数据集提取数据的程序,它将这些数据转换并加载到数据仓库的目标表中,以用于随后的分析。对满足数据仓库的三个主要功能而言,即数据的整合、积累及准备,它是数据仓库的关键器官。此外,对于数据仓库之运行,其重要性可与我们的心血管系统相比。设计、开发、测试、维护及扩展ETL是数据仓库构建的主要挑战,这一点在当今快速变化的世界中尤为突出。在这个意义上我们可以说,如能有效地构建ETL机制,原则上我们也就同时解决了数据仓库构建的问题。换言之,此二挑战本质上是同一的。

在过去的30年里,人们尝试过几乎所有能够想得到的办法,这包括不同的方法、体系结构、模型、工具等。然而,效果仍不尽如人意:费用昂贵,构建运作耗时,政治上高风险。对于为改善绩效而采用数据仓库技术的企业组织来说,该情形还有一个心理抑制的效应。他们不能也不愿意想得足够远大,因为他们所知道的和所经历的使他们认为,他们怎么也无法有效地做得那么大。

然而,作为一个程序集,ETL机制通常可以用两种方式来给予观察:

(1) 水平方式。程序集中每个程序都只负责单个目标表的相应提取、转换和加载。

(2) 垂直方式。整个程序集包括三个子集。第一个负责所有表的提取;第二个负责所有表的转换;第三个负责所有表的加载。

换言之,对所有目标表而言,所需构建的是非常相似但又不完全相同的程序。这是ETL机制的特点,也是数据仓库构建的特别之处。

本文首先分析现有的、一般的机制构建方法,指出其对数据仓库构建的不足。接着再分析、观察实验室环境下数据仓库构建之行为模式,由此引入域通用知识和特殊对象知识的概念。以此为据,再对现实生活中数据仓库构建之行为模式进行分析。这些观察、分析及概念将为后面新构建方法的引入打下基础。

* 通信作者:余肖生,三峡大学计算机与信息学院,副教授。E-mail:yuxiaosheng_2005@163.com。

2 阿基米德及亨利·福特的ETL机制构建方法

我们先看看古代最伟大的工程师阿基米德（约前287年—前212年）及汽车工程师的先驱和装配线生产技术的发起者亨利·福特（1863—1947年）将如何构建对付1000个目标表的ETL机制。假设，该任务必须在极短的时间内完成，譬如在一天之内。另外，可调用的程序开发人员数量不限。

2.1 阿基米德的构建方法

在锡拉丘兹的围攻战中（约前214—前212年），阿基米德曾试用“阿基米德热射”进行火攻以摧毁敌舰，即用数百个高度抛光的青铜或黄铜盾牌作为反射镜，将阳光聚焦到驶近的敌船，使其着火，如图1所示[1]。

依此原理，阿基米德有可能把对应每个目标表的ETL程序看作一副盾牌镜。由此将程序开发人员划分为小型的开发小组。根据提取、转换、加载等功能要求，每一小组将对一小数量的目标表的ETL程序进行全功能开发。

2.2 亨利·福特的构建方法

如图2所示[2]，福特可能会采取汽车生产装配线的方法。他可能会将整个构建ETL的任务分成更小的子任务，如提取、转换和加载，把它们看作工作步骤类型，并把每一步骤类型分配给一专门的、独立的开发团队。换句话说，每个开发团队将只处理整个工作步骤链中的一个固定的小部分，对所有的目标表的处理均如此。

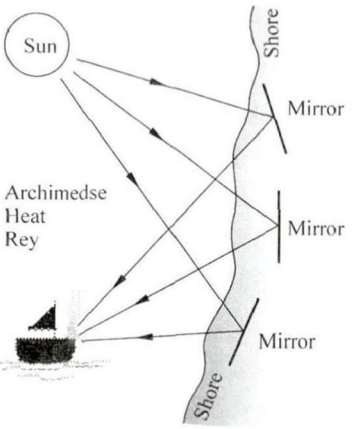

图1 阿基米德用“阿基米德热射”进行火攻以摧毁敌舰

图2 福特生产汽车的装配线

2.3 两种构建方法对比及讨论

总之，阿基米德会水平地划分任务，而福特却是垂直划分。每个阿基米德的程序开发员都能在整体上了解自己的ETL程序，由此他们在此都是全能手。而且，他们会以其工作为乐，因为产生的程序并不要求完全相同，而且简明的设计任务书允许得到单独的、创造性的、因而不同的、富有个性的诠

释。然而，程序的生产会因此变得昂贵、费时、低效。因为程序的差别相当大，团队之间的成员也不便交换[3]。

由于同一团队生产的所有程序都几乎相同，福特的开发人员对于既定的小任务都是训练有素的专家。因此，他们能高效地生产高质量的产品。然而，他们不会真正地关注全局。因此，他们的工作会显得单调无聊。

当然，现实生活中的数据仓库 ETL 机制并不像我们上述的那么简单。我们已经确认了二十多个 ETL 机制的功能任务类型。其中有些相当复杂，具有挑战性。此外，其适用性与数据源的状况及企业组织对其数据仓库的具体要求密切相关。后者本身往往也会很复杂和具有挑战性。此外，某些任务类型可能还会有多种变体。

在过去的 30 年里，我们通常都是用阿基米德法来构建数据仓库的。由于有诠释的自由，这工作给我们带来许多乐趣。然而，这乐趣常常变成了玛丽·雪莱(Mary Shelley)之噩梦[4]。换言之，对我们而言，阿基米德法已被证明是不当的。福特没有给予其员工任何诠释的自由。他的方法基于一简单的观察，即分配给每个团队的工作可以绝对相同地完成。然而，ETL 程序虽然非常的相似，但它们还是不完全相同，即对我们而言，亨利·福特的方法也不合适。存在一个有效适用的方法吗？下面，我们将分析构建 ETL 机制时会出现的问题，借以给此问题一个明确的答复。

3 实验室环境下数据仓库构建之行为模式分析[5]

为找到高效的数据仓库构建方法，下面，我们首先对 ETL 机制构建中的典型编程过程进行分析。为了便于对讨论的理解，我们要考虑每位数据仓库构建者几乎每天都会面对的基本任务，即为接收新提取的源数据做准备。尽管这项任务本身微不足道，但它却包含了所有下面将给予详尽考察的基本概念元素。

假设我们的数据库“my_db”含 1000 个表，如“tab_adfhkfha”、“tab_lkeas”、…、“tab_dajpgh”。我们想清空这些表。我们知道，符合 SQL(结构化查询语言)语法“DELETE FROM 〈数据库名〉.〈表名〉;”的语句适合于这一任务。我们如何完成这一任务呢？一种可能是写下下面这 1000 个非常相似，但又不尽相同的语句，然后执行它们：

```
DELETE FROM my_db. tab_adfhkfha;
DELETE FROM my_db. tab_lkeasr;
...
DELETE FROM my_db. tab_dajpgh;
```

为此，你可能会复制用于删除第一个表的语句，再将其粘贴 999 次。然后，一个个调整这每一粘贴语句的某些部分，如表名。当然，调整之后你还须验证这些调整是否正确。现在的问题是，你需要多少时间才能完成这些表的清空呢？如果你不太走运，它将使你度过好几个乏味的时辰。

让我们来仔细看看上面的过程。在这里，最突出的应是重复。这里有两种类型的重复。

(1) 内容重复。如上述完整的 SQL 语句中的短语“DELETE FROM my_db”。

(2) 操作重复。即为产生后续的 999 个删除(DELETE)语句的编辑操作：“复制(第 1 个语句)—粘贴(999 次)—搜索(表名 999 次)—替换(用新的表名替换旧的表名 999 次)—调整(如有必要的话，999 次)—验证(验证结果语句是否正确 999 次)”。

3.1 内容重复

为什么会有第一种类型的重复呢？这是因为对于以上所有删除语句而言，上述的SQL-语法要求是通用的。由于SQL-语法，或由于编程风格的约定，它必须看起来是这个样子，比如，所有SQL保留字必须大写。对于相关的开发团队而言，其他任何形式的语句可能不合适，或很难看。我们将这样重复的内容看做“域通用知识”(domain-generic knowledge)的载体。这里所谓的“域”是指一个予已考虑的、有限的活动区域。

与此相应的问题是，哪些内容不重复呢？在清空的例子里，表名是完全不重复的。每个语句含有其唯一的、特殊的表名。我们称这样的事物为“对象特殊知识”(object-specific knowledge)的载体。在我们的例子中，一个表就是一个对象。

基于以上概念，对于内容重复可简言为：域通用知识的载体重复，而“对象特殊知识”的载体则不重复。

事实上，福特的流水线法是基于这样的观察：许多汽车组件是完全相同的——用我们的术语来说，它们是某种通用知识的载体，从而可以简单地“复制和粘贴”，即重复相同的生产或制造。通过他的流水线法，福特使汽车构建达到了革命性的高效率。另外，他不能恰当处理对象特殊知识。而这正是ETL机制的特点。因此，他的方法还不能直接移植到我们的数据仓库构建上来。

3.2 操作重复

第二类重复见于用得极多的编辑操作链，或最常见的现代行为模式：“复制—粘贴—搜索—取代—调整—验证”。

对于提高生产率而言，使用计算机编辑工具对所有相关事物进行“复制 & 粘贴”可能是自几千年前发明印刷术以来最为伟大和最具影响力的发明之一。基于数据仓库的特点，我们这些数据仓库构建者充分地利用它来快速地生产ETL程序。(若没能意识到这一点，那我们可能还没开始就已经失败了)另一方面，将被开发的ETL程序通常虽极其相似，但并非完全相同。因此，我们必须“查找”——同“替换”一道组成另一伟大且极具影响力的发明——应该含有其他内容的地方。然后用新的内容自动地“替换”那些旧的内容。如果所涉及的程序并非十分相似，那么通常有必要在一定的程度上手工“调整(修改/改变/优化/等)”新程序。无论如何，“验证”结果的正确性是不可或缺的。

为什么需要“复制 & 粘贴”呢？这是因为基于其显著的相似性，旧的和新的程序应该有一个相当的部分可以共享。换句话说，应存在相当分量双方共有的部分。如果共有的部分分量不够，则没必要采用“复制 & 粘贴”。为什么能自动“搜索 & 替换”呢？这是因为我们准确地知道什么能够区分它们。换言之，我们知道这些单个程序有一些有规律的、特殊的东西。为什么必须手工“调整”生成的程序呢？这是因为存在其他的、特殊的、不是那么容易简单地通过“搜索 & 替换”予以处理的东西。如果我们不能感知或确定两个程序之间存在明显的、可利用的相似性，无论是“复制 & 粘贴”，还是“搜索 & 替换”，都将不便于用来构建新程序。在这种情形下，长操作链退化成一个短链，即“调整 & 验证”。换言之，我们将会像福特一百多年前所做的那样，用古典的打字机，或像阿基米德两千多年前那样，用羽毛笔构建我们的ETL程序。为什么需要单个地“验证”这些程序呢？这是因为“搜索 & 替换”可能做一些超越期望的事。此外，任何手工操作都可能出错。我们估计，亨利·福特和他的设计团队在设计他们的生产线之前也对其员工的生产行为进行了类似的分析。

这里有一些关系需要说一下：

(1) 因为存在着一些明显的、可利用的相似性或共性，内容重复引发操作重复。

(2) 由于不完全相同的或特殊的成分的存在,操作重复中的所有四个后续操作是由其前两个操作,即"复制 & 粘贴",所引发的。

简言之,存在一些有相当分量的共有成分对这种处理行为模式是有决定性意义的。而 ETL 程序中确确实实存在许多这种成分。

3.3 更论操作

操作对"复制 & 粘贴"威力无比。只要其应用范围选择恰当,"粘贴"则是最悍猛的操作。福特非常漂亮地利用了它来构建汽车,最终导致工业革命。另一方面,实际中一些具体要求限制了其总体效果。首先是其副本不可随后单独更改。如果做不到这点,一个方便便宜的工业化生产问题将变成一个烦琐昂贵的工匠生产问题。这也就是为什么只有在交付给用户的相机或笔记本电脑没被客户打开或自行修改的前提下,供应商才提供保修的原因。另一问题是,原件或设计必须是完美无瑕的。如果它包含任何错误、缺陷,或诸如此类,超猛的"粘贴"将无情地散发它们。被"复制"的部分"粘贴"得越频繁,上述纰漏则散发得越广泛。由于"原型"/"设计"上的一些缺陷,我们时常见到要召回某型号的被"粘贴"的汽车的告示。当一些分散了的缺陷由客户自己单独更改或以无控的方式得以"纠正"时,灾难也就发生了。

如果一个复制品的共同的部分(即某通用知识的载体)通过所谓的"调整"被单独地改变了,那它就不再是原件的副本了。它成了,或者至少似乎是,一些新的特殊知识的载体。"调整"运用得越频繁,存在于你的 ETL 程序集的(新的)特殊知识就越多;存在于你的 ETL 程序集中的特殊知识越多,你的 ETL 程序集也就越像一个工艺品的集合。通常,工艺品是昂贵的,并且处理它们是烦琐的。在这个意义上我们说,"调整"是有害的。虽然在一切都刚开始时你的 ETL 程序集可通过"复制 & 粘贴"(像福特生产他的汽车那样)及"搜索 & 替换"(这是福特做不到的,因为他没有相应的操作对)等方式进行工业化生产。但最终由于"调整",你不再能够区分什么是通用的,什么是特殊的,因为一切都混在了一起。白天,它是一套精美的工艺品,但晚上它可能是玛丽·雪莱噩梦的主题。

4 现实生活中数据仓库构建之行为模式分析[6,7]

为了对本质性的东西有清晰深刻的理解,上面,我们在一个分离的、极其简化了的条件下,对 ETL 机制构建中的典型编程过程进行了详尽的分析。下面,我们将用由此获得的基本认识来考察现实生活中数据仓库构建之行为。

如果我们只用操心于少量的、单个的、手工制造的 ETL 程序的话,如同对待通常的应用程序那样,我们的生活应是相当高枕无忧的。然而,在任一中等规模的企业组织里,其企业数据仓库就可能有几十个源应用系统,且每个源应用系统可能提供几十上百个源表。因此,整个 ETL 机制可能含有数以千计的 ETL 程序,且其中某些部分还可能相当复杂。这是 ETL 机制的另一个主要特征,即量性特征。与上面谈到的质性特征一样,我们可以说,企业数据仓库的 ETL 机制通常是由大量的、极其相似,但又不完全相同的程序所组成。

存在于这些程序中的大量的相似性可以用来提高构建效率。与此同时,这大量的程序使得数据仓库建设成为真正的挑战。这是因为除了烦琐的编程外,所有这些程序都必须经历设计、指定、测试、记录以及维护的过程。这庞大的工作量不可能由一个人在短期内完成。相反,这通常需要大型的设计及开发团队在一段很长的时间里完成。

4.1 大型的开发团队

动用大批的开发人员将给数据仓库构建带来挑战。这些开发人员每个人都有自己的把握和理解力,习惯于不同的思维和工作方式,他们掌握开发工具和辅助设备的水平不同并且驾驭数据库管理系统、SQL的程度也不一样。因此,虽然是根据相同或相似的设计说明书加以开发,但是,开发人员甲开发的程序并非总能被开发人员乙直接理解和维护,即使该程序运行正常。简言之,这些程序里包含有太多个性的东西。换言之,设计说明书中隐含的通用知识可以由不同的开发人员通过编程以不同的方式,但又完全正确地给予诠释。在这种情况下,实际存在的通用性难以识别,会被认作为某种特异性,我们称之为伪特异性。更具体地说,某通用知识的载体,譬如由某一开发人员开发的某程序的一部分,对其他开发人员而言显得像是某些特殊知识的载体。由此,这些开发人员也只能这样特殊地处理这些载体了。

4.2 很长的开发时间段

长时期开发意味着,即使程序是由相同的开发人员但在不同的时间点开发的,该开发人员应用的布局、隐含的风格、采用的技巧以及融入的个人习惯都可能会发生变化,哪怕这变化来得缓慢。某一开发人员难于理解他两年前开发的程序的现象很常见。由于时间在记忆上的作用,常常发生标准、协议或指南被遗忘或被误解的现象。如哲学家们早在几千年前就已经注意到的,今天的开发人员和两年前的他已是大不一样了。同理,对于相同的设计说明书,他今天的"诠释"也和他两年前的诠释不太一样了。在这个意义上,假设我们的一个团队事实上有10个开发人员,他们在两年的年初和年尾面对同样的设计说明书。那么我们在实际效果上将会有20～30个不同的、提交诠释的开发人员。这是一个应该值得注意的维的值的转换:从一时间维的值,转换到一物质维(即人员)的值。

事实上,时间因素的影响不仅如此。随着时间的流逝

(1) 业务要求和技术环境在变化;

(2) 利益相关者(赞助者、管理人员、架构师、设计师、开发人员、测试人员等)在流动;

(3) 老的数据源被废弃,新的数据源被开发;

(4) 副本程序的原本,即"通用知识"之源,可能被修改、调整、改进、优化,甚至重新设计及重新制作。

如果副本保持不变,情况还不至于太糟糕。因为原件的任何变化都可采用受控的"搜索—替换—调整—验证"重新复制并重新粘贴到现有的副本上。然而实践一再表明,这些副本本身也有可能出于某种原因被单独地修改、调整、改进、优化及扩展。

在一切刚开始的时候,什么是共有的、通用的,什么是个别的、特殊的,可能很清楚。因此,我们能够通过神力无比的操作链以产业化方式高效地工作。这时,原件和副本之间的关系是明确无误的。然而,在最后结束时,所有东西都混淆在一起了。没有人知道它们之间曾有过一个"原件-副本"的密切关系。现在,它们一个个都像孤儿一样孤苦伶仃。因此,所有其后而来的变动不得不单独地、直接地在副本上进行。

4.3 不限于编程

事实上,上述观察对一个庞大的设计团队的不同设计师在很长一段时间内的设计过程及由他们交付的设计说明书来说同样有效。实际上,前面细述的编辑行为模式不仅局限于程序与编程、说明书与说明,它们同样也广泛地存在于文档编制与文档之中。此外,只要程序在开发或维修期间发生变

化，那些原来通过运用我们驾轻就熟的编辑操作链产生的文档则应马上得到相应的更新。就像上面描述和讨论的那样，随着时间的推移，这将变得越来越困难。此外还有一个原因，程序文档的及时更新虽然对于维护和推广的成效至关重要，但对于程序本身的正常运行并非绝对必要。显而易见，若无相当的努力，所有这些程序最终至少在心理上被认为是无文档程序。

突然间，我们发现我们身处阿姆斯特丹跳蚤市场：一大堆单一手工制作的，无文档记录说明的，其维护和扩展需要不成比例的时间和金钱的工艺品。这就是当今数据仓库构建人的生活及工作现实的真实写照。最后，但同样重要且必须明白无误说明的是，所有这一切观察都与是否采用某种 ETL 工具或辅助设施无关。

事实上，这里的主要问题是对存在于 ETL 机制中的通用知识之载体的处理方式不恰当。由于上面讨论的种种原因，它们的形式发生了变化以至于它们所携带的通用知识不能够被感知并得到利用。这样，通用知识同特殊知识混杂在一起，以至于通用知识也被视为特殊知识。对个别的、单一的、特殊的事物的处理几乎总是更昂贵，更麻烦。这就是我们的数据仓库构建悲剧的真正原因所在。

5 小结

- 通常的企业数据仓库的 ETL 机制包含大量相当类似，但又不完全相同的程序。
- 由于这种显著的相似性，ETL 机制中存在大量的域通用知识。
- 无论是否使用工具或辅助设施，采用当今流行的构建方法，这类通用知识不知不觉地、不受控制地被分散在每一程序之中。
- 由于 ETL 程序数量大，加之数据仓库的长寿性及动态性，这些程序不可避免地会经受大量变化。
- 这些变化最终导致通用知识与特殊知识的载体成为一种不可分离的混合物。而这又使一切显得都是某种特殊知识的载体。
- 特殊知识必须特殊、单独地加以处理。过去几十年的数据仓库构建经验告诉我们，这种处理是极其昂贵、耗时、高风险的。
- 为避免伪特殊知识载体的产生，通用知识不应被分散。用管理学的话说，对通用知识只允许唯一的诠释。这就是一个全新的、高效的数据仓库构建方法的核心思想所在。

参考文献

[1] Archimedes[EB/OL]. [2012-12-18]. http://en. wikipedia. org/wiki/Archimedes.

[2] Henry Ford[EB/OL]. [2012-12-18]. http://en. wikipedia. org/wiki/Henry_Ford.

[3] Jiang B. Data Warehouse Construction: How Would Great Engineers Have Done It? [EB/OL]. [2012-12-25]. http://www. b-eye-network. com/view/16285.

[4] Inmon B. Whatever happened to Code Maintenance? [EB/OL]. [2012-12-25]. http://www. b-eye-network. com/view/15863.

[5] Jiang B. Data Warehouse Construction: Behavior Pattern Analysis[EB/OL]. [2012-12-25]. http://www. b-eye-network. com/view/16390.

[6] Jiang B. Data Warehouse Construction: The Real Life[EB/OL]. [2012-12-25]. http://www. b-eye-network. com/view/16473.

[7] Jiang B. Constructing Data Warehouses with Metadata-driven Generic Operators and More[M]. Switzerland: DBJ Publishing, 2011.

Behavior Pattern Analysisin Data Warehouse Construction

JIANG Bin, YU Xiaosheng, WANG Dongjuan, JIANG Yanjing, ZHAO Meilin
(College of Computer and Information, Three Gorges University, Yichang 443002, Hubei, China)

Abstract Construction and maintenance of ETL programs are of decisive importance for data warehouse construction and application. Traditionally, ETL programs are treated in horizontal or vertical ways. It is showed that both traditional approaches are not effective regarding development productivity, software quality and documentation quality for constructing and maintaining ETL programs. Through analyzing behavior patterns in constructing ETL programs, it is observed that the knowledge involved can be divided into domain-generic knowledge and object-specific knowledge. Domain-generic knowledge repeats and can thus be treated only once. Based on this observation, the development productivity, software and documentation quality of ETL programs can be improved significantly.

Key words Data Warehouse Construction, ETL, Behavior Pattern

作者简介

蒋彬(1955—),男,三峡大学计算机与信息学院,博士,教授。研究方向:数据仓库构建。E-mail:bin.jiang@bluewin.ch。

余肖生(1973—),男,三峡大学计算机与信息学院,博士,副教授。研究方向:商务智能与信息管理。E-mail:yuxiaosheng_2005@163.com。

王东娟(1977—),女,三峡大学计算机与信息学院,讲师。研究方向:商务智能与信息管理。E-mail:wdj@ctgu.edu.cn。

姜艳静(1978—),女,三峡大学计算机与信息学院,讲师。研究方向:商务智能与信息管理。E-mail:49699671@qq.com。

赵美林(1979—),女,三峡大学计算机与信息学院,讲师。研究方向:商务智能与信息管理。E-mail:1530870472@qq.com。