

ETL 执行过程的优化研究^{*}

吴远红

(浙江海洋学院信息学院 舟山 316004)

摘要 提出了一个 ETL(Extraction-Transformation-Loading)优化框架并对 ETL 过程的逻辑优化进行了研究,把优化问题建模成状态空间搜索问题。每个 ETL workflow 看作一种状态,通过一系列正确的状态变换来构造状态空间,并且提出算法来获得最小执行时间的 ETL workflow。理论分析和实践表明其具有良好效果。

关键词 ETL, workflow, 优化

The Research of Optimizing ETL Execution Process

WU Yuan-Hong

(Information College of Zhejiang Ocean University, Zhoushan 316004)

Abstract An optimization framework is provided in the paper, and the logical optimization of ETL processes is researched. The optimization problem is modeled as a state-space search problem. Each ETL workflow is considered as a state and the state space is fabricated through a set of correct state transitions. Moreover, algorithms are provided towards the minimization of the execution cost of an ETL workflow. The theory and experiment result prove it to be efficient.

Keywords ETL, Workflow, Optimization

1 前言

数据仓库作为一种数据密集型应用,由两部分构成:静态部分和动态部分,静态部分是指数据仓库的体系架构和实例数据,动态部分是构建和维护数据仓库的各种进程,负责加载、刷新等,这主要由 ETL 工具完成。ETL 负责将分布的、异构的数据源数据(如关系数据、文本数据、XML、HTML 等)抽取到临时中间层(Data Staging Area),在中间层进行清洗、转换、集成,然后加载到数据仓库,成为决策支持(如 OLAP, 数据挖掘)的基础。ETL 作为一种数据转换和集成的工具,是构建数据仓库的基本工具。

算法,理论和实践表明这种方法对于海量数据的处理很有效。

ETL 过程不能单纯地作为一个大的查询交给数据库去处理、去优化。为此提出如图 1 所示的 ETL 执行优化框架,它由 ETL 过程设计器、优化器、调度、抽取、转换、加载等几种类型的活动组成,其中 ETL 执行优化主要是寻求一个和原 ETL 过程等价的、最小执行代价的 ETL 过程。其处理过程如下:由 ETL 过程设计器设计好 ETL 过程,提交给优化器,通过转换规则进行等价状态变换生成状态空间,再由算法根据代价模型进行状态空间搜索获得最小执行代价 ETL workflow。

2 问题建模

ETL workflow 的优化问题可建模成状态空间搜索问题:每一个状态是一个 ETL workflow 图,对每个状态采用状态变换产生所有可能的等价状态,从中找出代价最小的状态即为最优 ETL 执行过程。

2.1 ETL workflow 的形式化定义

活动是一个四元组 $A = (ID, I, O, S)$, ID 是活动标识符, I 是输入模式的集合, O 是输出模式的集合, S 是一个或多个扩展的关系代数表达式,表示每个输出模式的语义。

每个 ETL workflow 看作一个状态也就是一个有向无环图(DAG 图),图的节点可以是 ETL 活动和记录集,边代表数据供给关系。

假定有活动集 A ,记录集 RS ,供给关系集 Pr , ETL workflow 可以看作有向无环图(DAG 图), $G(V, E)$, $V = A \cup RS$, $E = Pr$. $G(V, E)$ 可以拓扑排序,因此可以给每一个活动赋值唯一的执行优先权作为活动标识符。

2.2 转换规则

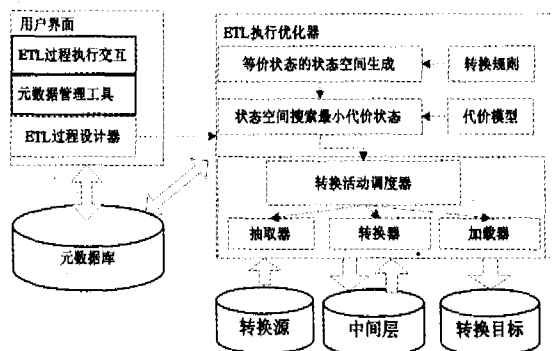


图 1 ETL 执行优化框架图

目前对它的研究主要在 ETL 过程的建模^[1,3],但对 ETL 过程优化的研究却不多,它不同于多查询优化^[7],因为多查询优化主要着眼于将各个不同的查询语句局部最优化,而在 ETL 过程中,各个活动相互关联,全局优化是必须考虑的。本文提出一种 ETL 优化执行框架,并给出具体的优化过程和

^{*} 基金项目:浙江省教育厅项目(20050113);浙江海洋学院项目(X05LQ07)。吴远红 讲师,硕士,研究方向:数据抽取、数据挖掘。

接下来引入状态的一系列逻辑转换。表达式 $S' = T(S)$ 表示从状态 S 到 S' 的变换, 这些逻辑转换包括:

1) SWA 变换: 交换一元活动 a_1, a_2 在图中的顺序, 记为 $SWA(a_1, a_2)$ 如图 2(a), 这样可以把选择频率高的活动推向工作流的开端, 类似于传统的代数优化。

2) FAC 与 DIS 变换: FAC 把汇聚前在汇聚的两个分支里各做了一次即两次操作, 放在汇聚后做一次。DIS 把一个

活动分配到两个并行分支里来提高效率。这两种变换分别记为 $FAC(a_k, a_1, a_2)$ 和 $DIS(a_k, a)$ 如图 2(b)。FAC 与 DIS 本质上是对一元和二元活动进行交换。

3) MER 与 SPL 变换: 用这两个变换来组合活动和取消组合而不改变它们的语义。这两种变换分别记为 $MER(a_{1+2}, a_1, a_2)$ 和 $SPL(a_{1+2}, a_1, a_2)$ 如图 2(c)。这样搜索空间可以大大减少。

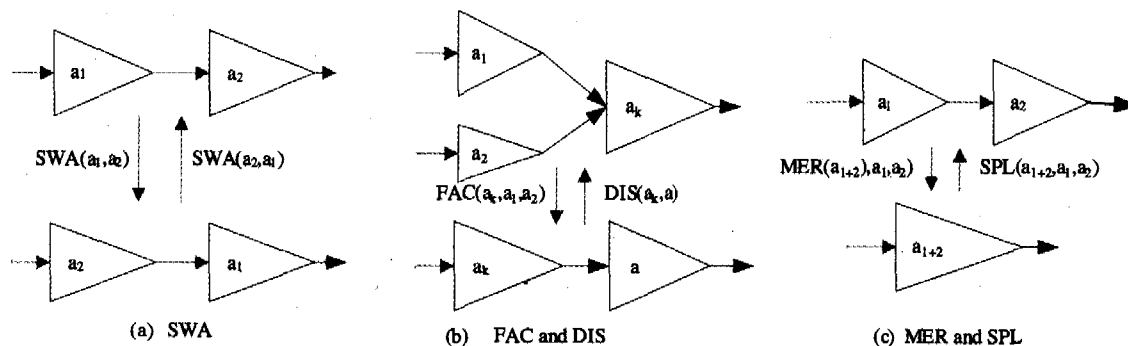


图 2 状态的逻辑变换

2.3 代价模型

给定活动 a , $C(a)$ 代表 a 的代价 (不仅和代价模型有关而且和活动在 workflow 中所处的位置有关), $C(a)$ 代价评估可以采用查询优化的各种代价模型。整个状态的代价是它所有活动的代价和。

$$C(S) = \sum_{i=1}^n c(a_i)$$

首先对搜索空间的每一个状态利用元数据库的统计信息进行代价评估。最优化 ETL 工作流问题就是找到一个状态 S_{MIN} , $C(S_{MIN})$ 最小。

2.4 元数据库

主要保存 ETL 过程的元数据和临时中间层中数据库概貌的统计描述, 包括模型信息、表定义、视图、用户自定义类型和函数约束等等。ETL 执行优化器在生成执行计划时将其作为定量分析的参考, 通常包括元组的数目, 属性的大小, 和对于不同属性的不同值的数目。为了保证基本统计信息的正确性, 需要不断地修改元数据库中的相关内容。

3 基于算法的状态空间搜索

3.1 穷举法

在穷举搜索法中, 对每个状态采用状态变换产生所有可能的状态, 并把状态空间抽象成图, 节点代表状态, 边代表状态间的转换。穷举搜索算法设置已访问节点集合保存已经访问节点和未访问节点集合保存未访问节点, 算法从未访问节点集合中取出一个未访问状态, 产生它的经过状态变化后的状态进行进一步处理。算法产生所有可能的状态, 然后从所有已访问状态中找出代价最小的状态, 即为问题的解。算法如下:

算法 Exhaustive Search (ES)

输入: 初始状态 S_0 , 即图 $G = \{V, E\}$ 。(表达式 $S' = T(S)$ 表示从状态 S 到 S' 的变换; SGen 函数用来产生某一个状态的所有模式)

输出: 最小代价状态 S_{MIN}

```
1. Begin
2.  $S_{MIN} = S_0$ ;
3.  $unvisited = S_0$ ;
4.  $visited = \emptyset$ ;
5. for each  $s'$  in  $unvisited$ 
6.   for each  $s' = T(S)$ 
7.     if ( $s' \notin unvisited$ ) & ( $s' \notin visited$ )
8.        $S_{MIN} = \min(Cost(s'), cost(S_{MIN}))$ ;
```

```
 $s' = SGen(s')$ ;
 $unvisited \leftarrow s'$ 
}
 $visited \leftarrow S$ 
}
5. return  $S_{MIN}$ 
6. End.
```

3.2 启发式算法

为了避免搜索整个状态空间, 采用启发式算法进行改进, 转换前对工作流可以约束的活动进行 MER 变换; 接着 HS 找到初始状态中所有的同类活动 (H) 和可分解活动 (D), 然后把初始状态 S_0 分组 (L); 仅在线性路径中应用 SWAP 变换; 对处于两个汇聚流的同类活动应用 FAC 变换; 在转换适用性允许的情况下应用 DIS 变换; 仅在前面用 FAC 变换和 DIS 变换产生的新状态的线性路径中再次应用 SWAP 变换; 最后返回最小代价状态 S_{MIN} 。

下面是其实现算法。

算法 Heuristic Search (HS)

输入: 初始状态 S_0 , 即图 $G = \{V, E\}$ 和在预处理中用到的一系列合并约束

输出: 最小代价状态 S_{MIN}

```
1. Begin
2. 首先对工作流可以约束的活动进行合并 MER 变换
3.  $Unvisited = \emptyset$ 
4.  $visited = \emptyset$ 
5.  $S_{MIN} = S_0$ 
6.  $H \leftarrow \text{Find-Homologous-Activities}(S_0)$ ;
7.  $D \leftarrow \text{Find-Distributable-Activities}(S_0)$ ;
8.  $L \leftarrow \text{Find-Local-Groups}(S_0)$ ;
9. For each  $g_i$  in  $L$ 
10.  For each pair  $(a_i, a_j)$  in  $g_i$ 
11.     $S_{new} \leftarrow SWA(a_i, a_j)$ ;
12.    If ( $c(S_{new}) < c(S_{MIN})$ )  $S_{MIN} = S_{new}$ ;
13.  }
14.  $visited \leftarrow S_{new}$ ;
15. For each pair  $(a_i, a_j)$  in  $H$ 
16.  If ( $(\text{shiftFrw}(a_i, a_b)) \text{ and } (\text{shiftFrw}(a_j, a_b)))$ 
17.     $S_{new} \leftarrow FAC(a_b, a_i, a_j)$ 
18.    If ( $c(S_{new}) < c(S_{MIN})$ )  $S_{MIN} = S_{new}$ ;
19.     $visited \leftarrow S_{MIN}$ ;
20.  }
21. }
22.  $unvisited \leftarrow visited$ 
23. For each  $S_i$  in  $unvisited$ 
24.  For each  $a_u$  in  $D$ 
25.    If ( $\text{shiftFrw}(a_u, a_b)$ )
26.       $S_{new} \leftarrow DIS(a_b, a_u)$ ;
27.      If ( $c(S_{new}) < c(S_{MIN})$ )  $S_{MIN} = S_{new}$ ;
28.       $visited \leftarrow S_{new}$ ;
```

```

}}}
13. For each Si in visited{
    L ← Find_Local_Groups(Si)
    For each gi in L{
        For each pair(ai, aj) in gi{
            Snew ← SWA(ai, aj);
            If(c(Snew) < c(SMIN)) SMIN = Snew;
        }
    }
14. Apply all SPLs according to spl_cons;
15. Return SMIN
16. End.

```

结束语 本文提出了一个 ETL 优化框架,并对 ETL 过程的逻辑优化进行了研究,把优化问题建模成状态空间搜索问题,每个状态代表一个 ETL 工作流的设计图,同时也定义了状态之间的转换,并且讨论了状态的产生和正确性问题,最后提出了状态空间搜索算法。首先提出穷举算法来构建所有的状态空间,以找到最优 ETL 工作流;然后提出一种启发式算法来减少搜索的状态空间。下一步的工作包括算法的实验比较分析,算法的进一步改进。

(上接第 20 页)

务。相比其它没有使用 OSGi 框架的项目,比如基于多 Agent 结构的 CoBrA 项目^[15],FollowMeLite 使用面向服务计算的思想去构造上下文感知应用,提供 Context 服务、Workflow Engine 服务以及 Unit Registry 服务去创建一个可靠、安全和独立的系统平台。通过配置应用和领域相关的上下文,FollowMeLite 可以方便地适用于多种应用环境。

和同样使用语义 Web 技术构造上下文模型的 SOCAM 相比,我们的上下文模型根据人思考的方式使用属性去表示活动。通过增加生存时间和时间戳特性,扩充三元组到五元组上下文表示,使得应用可以准确和及时的得到上下文信息。同时,我们也设计了一种基于 Ontology 的上下文冲突检测方法。

FollowMeLite 中间件的一个重要的贡献是把工作流模型引入了上下文感知应用的开发中。使用工作流描述应用逻辑可以很大程度上的减少开发的时间和降低开发的复杂度,并且增强上下文感知应用的复用性。我们通过裁减和修改工作流标准,得到 CPDL 语言。CPDL 使用事件触发机制和 RDQL 作为上下文信息的描述,使得工作流能够方便地应用于普适计算领域。

工作展望 我们已经实现了一种适用于普适计算领域的轻量级中间件平台,并且以这个平台为基础开发了一个智能地图的应用。未来的研究会集中在以下几个方面:设计针对上下文信息处理的通讯协议,使上下文信息可以分布式的处理;使用 Web Service 技术进行上下文信息的发布和获取;引入数据挖掘技术来增强上下文的准确性和可理解性;提供一些图形化工具,以 Eclipse 的 Plug-in 形式去支持上下文感知应用的开发。分析工作流模型的性能,探索使用模型驱动的方法开发上下文感知应用。

参考文献

- Weiser M. Computer for the 21st century. Scientific American, 1991, 265(3): 94~104
- Gu T, Pung H K, Zhang D Q. Towards an OSGi-Based Infrastructure for Context-Aware Applications in Smart Homes. IEEE Pervasive Computing, 2004, 3(4)
- Want R, Hopper A, Falcao V, et al. The Active Badge Location

参考文献

- Vassiliadis P. Conceptual Modeling for ETL Process [C]. In: Proc. 5th International Workshop on Data Warehousing and OLAP (DOLAP 2002), McLean, VA, USA, 2002
- Milo T, Zohar. Using schema matching to simplify heterogeneous data translation [C]. In: Proceedings of the 24th International conference on Very Large databases. New York: Morgan Kaufmann, 2004
- Vassiliadis P. Modeling ETL Activities as Graphs [C]. CAiSE'02, Toronto, Canada, 2002. 52~61
- Yan Ling Ling, Miller R J. Data-Driven Understanding and Refinement of Schema Mappings [C]. ACM SIGMOD, Santa Barbara, California, USA, 2001
- Cui YingWei, Widom J. Lineage Tracing for General Data Warehouse Transformations [C]. In: Proceedings of the 27th VLDB conference, Roma, Italy, 2001
- 张超. ETL 过程改进的研究[D]. 江苏南京: 东南大学, 2006
- 王木林. 基于 SQL 优化提高数据仓库的 ETL 效率的方案[J]. 中国科技信息, 2005, 18, 98~99
- 韩京宇, 徐立臻, 董逸生. ETL 执行的流水线优化[J]. 小型微型计算机系统, 2005, 26, 134~138
- System. ACM Transactions on Information Systems, 1992, 10 (1): 91~102
- Kindberg T, Barton J. A Web-based Nomadic Computing System. Computer Networks (Amsterdam, Netherlands: 1999), 2001, 35(4): 443~456
- Dey A K, Salber D, Abowd G D. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Anchor article of a special issue on Context-Aware Computing. Human-Computer Interaction (HCI) Journal, 2001, 16(2-4): 97~166
- 马凯, 张小蕾, 陶先平, 等. 一种基于 Agent 的移动应用框架设计与应用研究. 计算机科学, 2004(9)
- Chen Guanlin. Solar: Building a context fusion network for pervasive computing. [Ph D Thesis]. Dartmouth University, 2004
- The Open Services Gateway Initiative (OSGi), www.osgi.org
- Grimm R, Davis J, Lemar E, et al. Programming for pervasive computing environments. Submitted for publication, Jan. 2002
- Berners-Lee T, Hendler J, Lassila O. The SemanticWeb. Scientific American, 2001, 284 (May 2001): 34~43
- Owl. Web Ontology Language Reference (OWL). Available at: http://www.w3.org/TR/owl-ref/, 2002
- Resource Description Framework (RDF). Available at: http://www.w3.org/TR/rdf-ref/
- Workflow Process Definition Interface-XML Process Definition Language. Available at: www.wfmc.org/standards/docs/TC-1025_10_xpdl_102502.pdf
- RDF Data Query Language (RDQL) Available at: http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/
- Chen H, Perich F, Chakraborty D, et al. Intelligent Agents Meet Semantic Web in a Smart Meeting Room. Proceedings of the Third International Joint Conference on Autonomous Agents & Multi Agent Systems (AAMAS 2004), New York City, NY, July, 2004. 19~23
- http://www.xbow.com/Products/Wireless_Sensor_Networks.htm
- Li Jun, Bu Yingyi, Chen Shaxun, et al. FollowMe: A Pluggable Infrastructure for Context-Awareness. Workshop on Situating Ubiquitous Computing in Everyday Life: Bridging the Social and Technical Divide, in the 7th International Conference on Ubiquitous Computing. Tokyo, Japan. September 2005
- Li Jun, Bu Yingyi, Chen Shaxun, et al. On Research of Pluggable Infrastructure for Context-Awareness. In: The IEEE 20th International Conference on Advanced Information Networking and Applications. Vienna, Austria, April 2006