基于 SOA 的实时 ETL 的研究与实现

Research and Implementation of Real – time ETL based on SOA

许 力 （东软股份大连分公司基础软件事业部 116023）
马瑞新 （大连理工大学软件学院 大连 116200）

摘要：本文对基于 SOA 技术实现实时 ETL 过程进行了研究，并提出了一种实时 ETL 过程解决方案，为政府部门、企业的决策者对实时数据的需求提供正确的保障决策，给出一套完整系统架构。

关键词：ETL SOA Web Services 数据仓库 数据集成

1 前言

数据仓库主要为管理者决策过程提供支持，它是基于主题的、集成的、时间相关的，非易失数据的集合（Inmon，1993）。ETL 是构建数据仓库的重要过程，一般情况下，ETL 过程要占整个数据仓库构建总工作量的 60% 到 80%。传统数据仓库中数据一般是每天，每周，或是每月定期更新，难以保证数据的实时、有效。目前，通用的实时数据仓库数据加载解决方案还不存在，各种可行的解决方案都存在一定的局限性，大多是针对特定使用场景下实现实时 ETL 过程，产品不易复用很难满足所有企业、政府部门的需求。本文主要对基于 SOA 技术实现实时 ETL 过程进行了研究，并提出了一种实时 ETL 过程解决方案。

2 相关技术

ETL（Extract Transform Load）是将数据从数据源抽取、转换，并加载到数据仓库中的过程。ETL 划分为全量 ETL 过程和增量 ETL 过程。全量 ETL 负责处理全量数据，主要提供大批量数据抽取导入，系统易于设计。增量 ETL 负责增量数据处理，过程包括增量数据捕获，增量数据转换传输过程，系统相对复杂。

面向服务的体系结构 SOA（Service – oriented architecture）是一个组件模型，它提供了一种通过接口将各种实现独立功能的组件以 Web 服务的方式集成在一起的解决方案。采用面向服务体系结构增加了软件的可复用性，和系统集成的灵活性，降低了系统维护的复杂度。

实时 ETL 过程指将业务数据源中产生的数据实时地抽取加载到数据仓库，为数据挖掘系统、OLAP 软件、商务智能软件等决策支持系统产生更加及时有效的结果提供数据支持。

3 实时 ETL 过程研究

四种常用的实时 ETL 过程解决方案：

（1）准实时 ETL 方式（"Near Real – time" ETL）

（2）直接持续少量数据更新（Direct trickle feed）

（3）持续少量数据更新和翻页（Trickle & Flip）

（4）外部实时数据缓存方式（External Real – time Data Cache）

第一种方式是最简单，也是最廉价的实现方式，它通过增加当前 ETL 过程频率来提高数据仓库中数据实时性，这种调整避免了对原有系统较大改动，而且很有可能满足用户现实需求，对数据实时性要求也不是很高，如果当前数据仓库中数据更新速度不能满足需要的情况下可以考虑采用这种方式。

第二种数据更新方式是最常用，也是简单实现真正意义上实时 ETL 过程的方法。它通过持续不断从业务数据源中抽取数据，直接向数据仓库表中插入、删除数据或向数据仓库实时更新备份表中插入数据来实现实时数据更新。一些商业化 ETL 工具采用了这种方式提供实时 ETL 功能。但是这种方式需要持续不断地监控数据源数据更新，向数据仓库中加载数据，增加了业务数据源与数据仓库的负担。而且如果向某些被报表系统、OLAP 系统查询频率较高的表中持续更新数据将

会严重影响数据仓库性能。

第三种数据更新主要是为了解决同时更新和查询相同的数据仓库表所带来的问题。这种方式通过在数据仓库中创建相同结构的分段传输表，将查询和更新数据仓库的操作分离。分段传输表可以只包含当天的数据，如果事实表足够小，也可以包含所有数据。在特定周期内，分段传输表被复制，并与历史数据合并，重新构造一张事实表与当前事实表交换，保持数据仓库的实时性。如果使用视图，这个过程只需要修改视图定义就能完成。由于各种关系数据库管理系统处理表替换的策略不同，在替换过程中停止查询过程是比较安全的。表的替换周期可以根据数据仓库中数据的规模和硬件处理能力制订。这个周期有可能是几小时或是几分钟。

第四种方式采用在数据仓库外部创建实时数据缓存(RTDC)来加载、存储、处理实时数据更新的任务。RTDC 可以是一个指定的数据库服务器，如果在需要处理大量实时数据的情况下，使用内存数据库(IMDB)作为 RTDC 会是一个更好的选择。这种方式实现成本比较高，需要额外维护实时数据缓存数据库，而且原有的数据仓库应用程序也需要增加对实时数据缓存查询的机制。

本文研究的实时 ETL 解决方案结合了后三种实现方式，基于面向服务的体系结构，实现了通过利用实时数据缓存，由控制中心统一控制数据流程的较简单灵活的实时 ETL 过程。

4 问题分析

实时 ETL 设计过程中所遇到的问题主要有：

(1) 分部异构的业务数据库增量数据的捕获；

(2) 表/对象/关系映射；

(3) Web 服务方式数据传输性能瓶颈；

(4) 数据更新策略控制设计。

分部异构的业务数据库增量数据的捕获有六种常用的增量数据捕获方式：触发器方式，时间戳方式，数据复制，日志方式，快照方式，修改应用程序。这些方式各有优缺点，受使用场景的限制，其实现细节也有不同。本文介绍的实时 ETL 过程主要针对已有数据仓库的改造，数据增量捕获，主要考虑到针对已有业务数据库，增量数据抽取的通用性，采用触发器方式是比较可行的。触发器方式通过在数据库中建立触发器和辅助表，将更新数据抽取到辅助表中，ETL 过程将辅助表中数据导出，转换，加载到目标数据后，删除辅助表中相应数据，完成增量更新。这种方式可以在保持原数据库结构的基础上，相对简单快速地捕获增量数据，相比日志，快照方式效率更高。

在表/对象/关系映射过程中，可以把 XML 文件结构看作是对象树，这些对象可以映射到数据库中。下面的 XML 文档清楚地描述了数据库表，对象，XML 之间的映射关系。

表 1　XML document 和 DB table map 表

XML	Objects	Tables

<employee>
<Id>10011</Id>
<name> Tom </name>
<age>30</age>
</employee>
	object employee {
	id = "10011"
	name = " Tom "
	age = "30"
	}
		Table employee
		id = 10011
		name = Tom
		age = 30

同样，在元素类型定义、类、数据库表之间的映射关系也可以相互映射，其关系数据库，数据对象和 DTD 之间的对应关系如表 2 所示。

表 2　XML DTD 和 DB table map 表

DTD	Classes	Table Schema

<! ELEMENT employee (id, name, age) >	Class employee {
	String id;
	String name;
	String age;
	}
		CREATE TABLE employee (id VARCHAR (10) NOT NULL, name VARCHAR(10) NOT NULL, age VARCHAR(10) NOT NULL)

Web Services 是一种通过 URI 标识的软件应用，其接口及绑定形式可以通过 XML 标准定义、描述和检索，Web Services 能够通过 XML 消息及互联网协议完成与其他软件应用的直接交互，采用简单对象访问协议(soap)以 XML 的方式传递消息。通常它以 HTTP 协议来发送 XML 消息，也可采用专有的应用程序消息传递或传输协议(如 JMS)发送 XML 消息。本文介绍的应用场景需要利用 XML 消息来传递实时更新数据，这就需要高效的 Web Services 消息传送。过程中涉及影响 Web Services 数据传输性能的瓶颈主要在数据对象与 XML 消息之间的转换，XML 数字签名和 XML 数据加密。

如果 ETL 过程不需要经过 Internet，可以考虑使用专有的应用程序消息传递或传输协议提高数据传输效率。在 XML 消息传输过程中，包括 XML 数字签名（XML Digital Signatures）和 XML 加密（XML Encryption）的 Web 服务安全（WS－Security）功能的处理有可能严重影响服务性能，Web 服务通过 XML 数字签名（XML Digital Signature）提供验证和消息的完整性，通过 XML 加密提供消息的机密性，然而，通过 Web 服务技术来支持安全性所消耗的时间要高于使用传统的用 HTTP 的 SSL 提供相似功能。所以在确定使用 HTTP 协议传输消息的情况下，可以使用 SSL 来加密，然后使用 XML 数字签名来验证应用程序端点并确保消息的完整性。

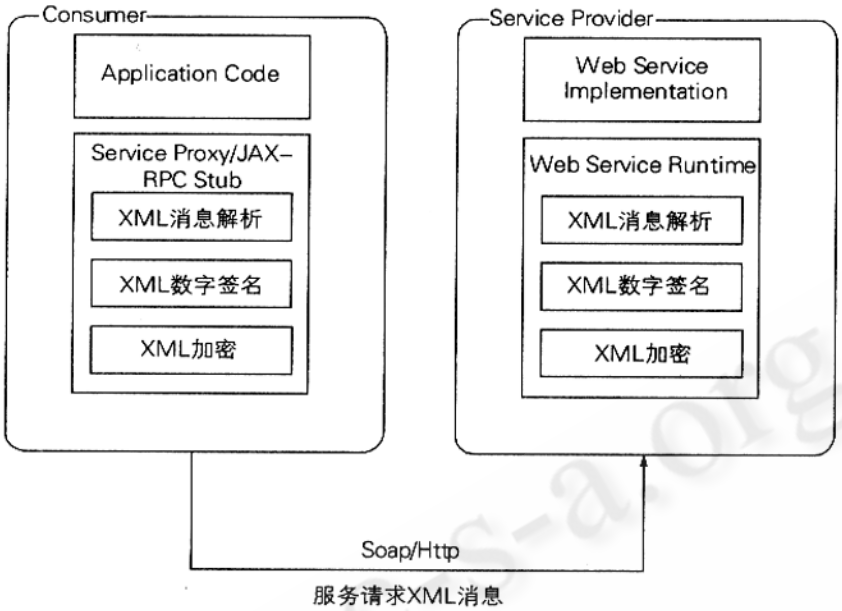

图 1　Web 服务请求示意图

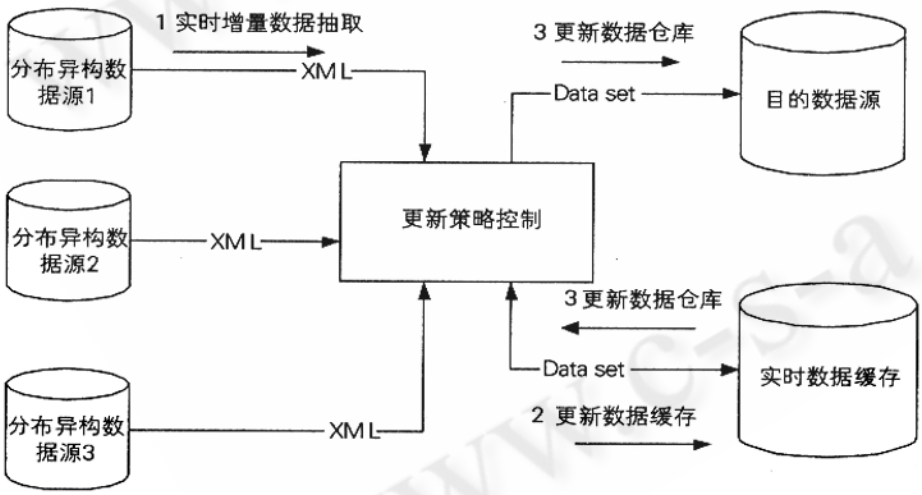

图 2　数据更新流程图

数据更新策略的设计需要根据具体数据仓库实时更新策略而定。一般情况下，增量数据捕获与实时数据缓存之间的 ETL 是持续进行的，这个过程不影响数据仓库提供查询服务的性能。但是在数据仓库接收到包含实时数据的查询时，如何将实时数据缓存中的数据快速地提供给数据仓库，并与历史数据融合生成查询结果就成了该过程中首先要解决的问题。目前共有两种解决方案，第一种修改原数据仓库数据模型，数据仓库本身不需增加数据模型。外部数据缓存中的数据建模应该与数据仓库相同，但只包含需要实时更新数据的表。外部数据缓存采用独立的数据访问方式，有可能需要一些额外的表，如查询表等。对于数据的查询，一般来说，把外部数据缓存与数据仓库中历史数据无缝地整合起来将更有利。第二种则是通过将实时数据缓存中数据通过定时触发或设定阀值监控缓存中实时数据量触发直接将数据导入数据仓库的方式来完成数据更新。

5　系统体系结构

本文介绍的实时 ETL 过程基于 SOA 框架，分为业务数据库端 Wrapper，控制中心和元数据管理三个部分，以 Web 服务方式连接各部分功能。这种结构屏蔽了数据库和操作系统的异构性，增强了系统部署的灵活性和可扩展性。

包装器 Wrapper 部署在业务数据库端，通过 JDBC 连接业务数据库。其中包括的元数据抽取模块和增量数据抽取模块分别负责实现元数据抽取和增量数据抽取 Web 服务。抽取规则引擎则通过控制中心调用 Web 服务传入 XML 形式的抽取规则定义来创建触发

器和辅助表。

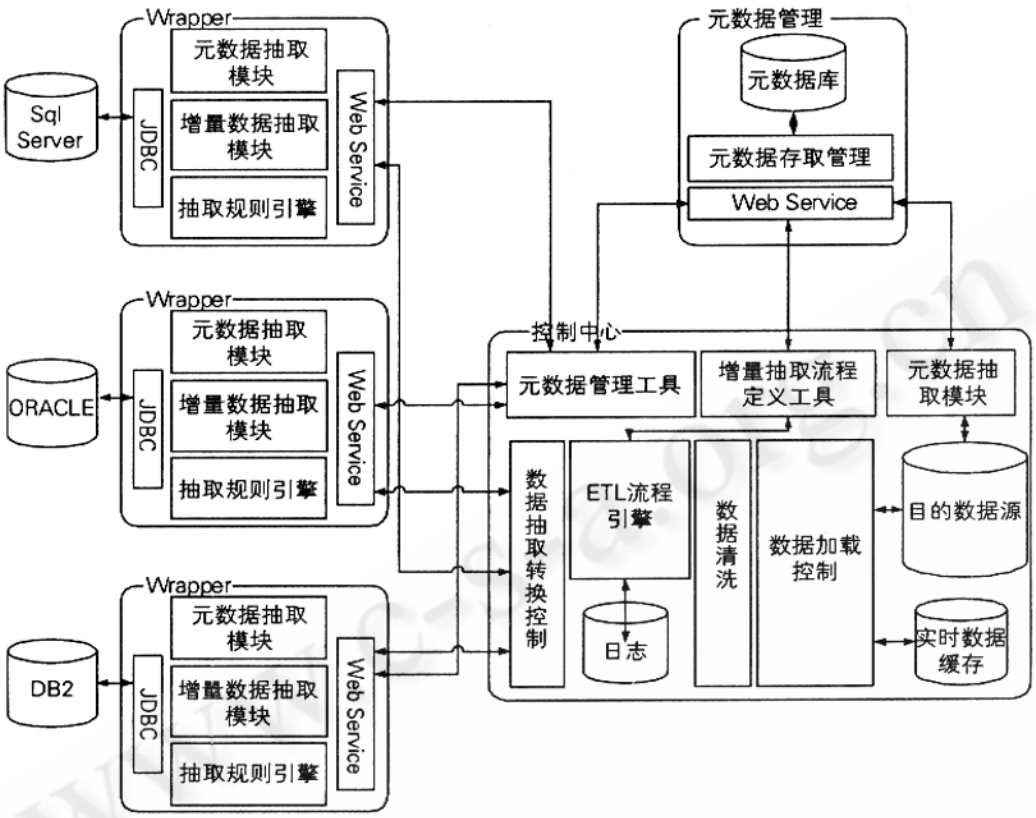

图 3　系统体系结构

XML 数据转换中间层为上层企业应用提供了一个统一的数据访问格式，屏蔽了存储于不同 RDB 中数据格式的差异性。目前企业级应用中涉及的数据类型主要有存放在 RDB 中的数据，按各种 XML Application 标注的数据，EDI 格式的数据和存放在 LDAP 及其它信息组织形式中的数据。前三类是当前信息存储的主要部分。由于已经有 EDI 数据的 XML 组织方式 XEDI 的存在，所以从逻辑上主要面对关系模式下的结构化数据和 XML 模式下的半结构化数据两类数据。关系模式和 XML 模式数据间转换基于对象关系映射模型（object – relationcl model）理论。

元数据管理提供元数据库读取 Web 服务，管理元数据库。控制中心则是该 ETL 过程的核心，它部署在目的数据库端，管理实时数据缓存，提供 ETL 过程设计人员可视化的元数据管理工具和增量抽取流程定义工具。ETL 流程引擎负责解析，执行流程定义模板，通过调用数据抽取转换控制和数据加载控制模块接口控制实时 ETL 数据流程。目的数据源实时数据更新策略的定义包含在流程模板中。

6　结束语

实时 ETL 的实现是一个比较复杂的过程，需要考虑的因素很多。目前很多 ETL 工具生产商都在研究支持实时 ETL 过程。如何增强实时 ETL 过程的通用性，降低改造现有数据仓库 ETL 过程成本则成了该技术竞争的焦点。采用可跨平台，具有灵活部署和结构松耦合特性的 SOA 框架和 XML 统一数据格式可以较好地提高通用性，降低成本。

参考文献

1 W. H. Inmon, Building the Data Warehouse. Wiley & Sons, New York, 1993.

2 张旭峰 孙未未 汪卫 冯雅慧 施伯乐，增量 ETL 过程自动化产生方法的研究，计算机研究与发展，2005.

3 章水鑫、徐宏炳、于 立，增量式 ETL 工具的研究与实现，研究与开发，2004.

4 IBM，Web 服务安全性，http：//www – 128. ibm. com/ developerworks/ cn/ webservices / ws – secure/.

5 Dan Linstedt, Additional thoughts: ETL and ELT, RDBMS, http: //www. teradata. com/t/page/127099/ index. html.