实时主动数据仓库中面向需求的实时数据集成方法研究

林子雨 杨冬青 宋国杰 王腾蛟
(北京大学信息科学技术学院，北京 100871)
(cainiu@263.net)

摘要 实时数据集成是实时主动数据仓库研究领域的一个重要问题。现有的研究成果都是从技术角度出发，而并没有考虑具体的商务应用需求。而在大型商务应用中，即使采用过滤规则只捕捉感兴趣的变化数据，也会产生大量的数据集成工作，从而导致不必要的沉重系统开销，同时还很有可能出现系统响应缓慢和用户需求无法得到满足等情况。本文从应用角度出发，提出了实时主动数据仓库中面向需求的实时数据集成方法，包括被频繁请求的数据的实时集成、满足突发请求的实时数据集成和由用户决定的实时数据集成。针对不同的商务需求，采用不同的数据集成策略，可以很好地满足不同类型的应用需求。

关键词 实时主动数据仓库；实时数据集成；ETL；

中图法分类号 TP301

Research on Requirement-based Real-time Data Integration in Real-time Active Data Warehouses

Lin Ziyu, Yang Dongqing, Song Guojie, Wang Tengjiao
(School of Electronics Engineering and Computer Science, Peking University, Beijing 100871, China)

Abstract Real-time data integration is a very important aspect in the field of real-time active data warehouse. Almost all the available research work now is from a technological point of view instead of an application angle. While in the real-world business application, a large amount of real-time data integration needs to be done even with the help of change data capture technology to integrate only the interesting part of the data from the data source, which will usually lead to the deteriorated system performance and fail to satisfy the business requirement in some cases. From an application angle, we here propose three requirement-based real-time data integration methods, including: real-time integration for the frequently requested data, real-time integration for the suddenly arising requirement and user-decided real-time integration. By adopting the appropriate method for the specific application occasion, we can better satisfy the various business requirements.

Keywords real-time active data warehouse; real-time data integration; ETL

1. 引言

传统的数据仓库通常不包含当前数据，因为他们采用 ETL 工具周期性地从数据源中抽取数据，经过处理后加载到数据仓库，而数据抽取的周期通常为一个月一次、一周一次、或者一天一次[1]。但是，当前的商务需求对数据的实时性提出了更高的要求，尤其是实时主动数据仓库[2],[3],[4]，要求实时捕获数据源中发生的变化，并且根据预先设置的规则做出战术决策。因此，实时数据集成逐渐成为数据仓库领域研究的热点。

目前，在实时主动数据仓库领域，针对实时数据集成，研究人员已经提出了许多可行的办法，比如最小化更新窗口[5]、数据/表交换[6]和变化数据捕捉技

收稿日期：2007-07-05
基金项目：国家自然科学基金项目(60473051)；国家高技术研究发展计划 863(2006AA12Z217)；HP 中国实验室联合项目

术[7]等等，但是，这些实时数据集成方法都是从技术的角度进行研究，都没有对具体的商务需求进行分析.。在大型的商务应用（比如移动通信领域）中，即使采用过滤规则只捕捉感兴趣的变化数据，也会包含大量的数据集成工作，从而导致不必要的沉重系统开销，同时还很有可能出现系统响应缓慢和用户需求无法得到满足等情况。

本文从应用角度出发，提出了实时主动数据仓库中面向需求的实时数据集成方法，针对不同的商务需求采用不同的数据集成策略，从而尽可能满足不同类型的用户需求。

本文结构安排为：第 2 部分介绍我们的研究所采用的实时主动数据仓库体系架构；第 3 部分详细阐述面向需求的实时数据集成方法；第 4 部分介绍相关工作；最后在第 5 部分给出结论。

2. 实时主动数据仓库体系架构

实时主动数据仓库可以采用不同的体系架构，比如实时数据可以保存在实时分区[6]，或者保存在外部实时数据缓存[8]中，我们的研究采用了基于外部实时数据缓存的架构。对于这种体系架构，需要解决数据建模、数据集成和数据合并等问题。

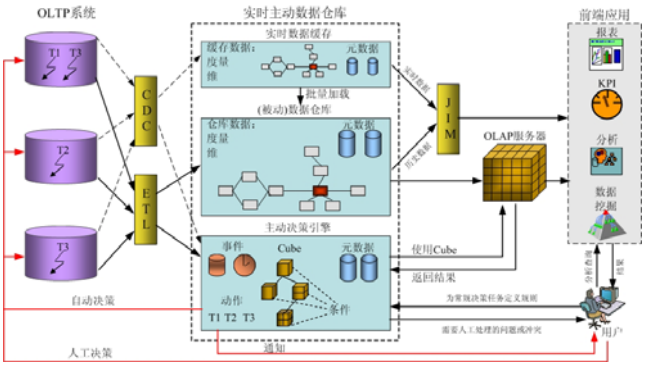

图 1 实时主动数据仓库体系架构

2.1 数据建模

当采用外部实时数据缓存来存储实时数据时，不需要特殊的数据建模。实时数据缓存的数据建模与数据仓库一样，都采用维度建模[9]，并且采用相同的事实表和维表结构。不同的是，实时数据缓存中只包含了那些需要实时更新的表。

2.2 数据集成

并不是所有的应用都需要实时数据，并且对于某些应用，采用实时数据引起的开销远远大于其获得的收益。因此，对于不同的应用，应当采用不同的数据集成方式，即批处理和实时数据集成。批处理通常以批量作业的方式周期性地执行数据加载任务，几乎所有的传统的 ETL 工具都被设计成以批处理的方式工作。但是，对于那些需要实时数据的应用，满足它们的最好的方式就是把变化数据源源不断给从数据源集成到数据仓库（或者外部实时数据缓存）中。CDC（变化数据捕捉技术）[7]就可以很好的满足这类实时数据集成需求。

2.3 数据合并

某些实时分析可能既需要历史数据也需要实时数据，这就意味着必须存在一种机制能够对历史数据和实时数据进行有效合并以提供给 OLAP 工具使用。这里，我们采用 JIM（即时信息合并）[8]机制来实现位于数据仓库中的历史数据和位于实时数据缓存中的实时数据的无缝集成。

3. 面向需求的实时数据集成方法

面向需求的实时数据集成方法充分考虑了具体商务应用环境的特定需求，对不同应用采用为其量身定制的实时数据集成方法，减小了需要集成的数据量，减轻了系统的负担，也提高了实时集成的效率。

从对移动通信领域的数据仓库应用的分析中，我们提炼出了三种典型的实时数据应用需求，包括：

- 频繁的实时数据请求
- 突发性实时数据请求
- 规则引擎需要获得用户输入的实时数据，从而做出主动决策并指导用户的行为

为了满足这三种应用需求，我们分别有针对性地设计了不同的实时数据集成框架。

3.1 被频繁请求的数据的实时集成

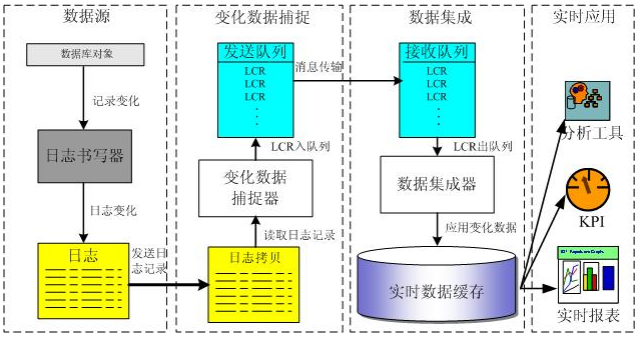

图 2 实时数据集成框架 A

在实际商务应用中，总是存在一些针对特定数据的频繁的实时请求，比如 KPI（关键性能指标）就需要不断地获得特定的实时数据，从而实时显示企业内一些重要指标的变化情况，为分析人员提供决策支持。对于这类应用，我们必须为其建立一条稳定有效

的实时数据获取渠道，从而保证在数据源中发生的变化能够以最快的速度反映到目标应用中。图 2 就是我们针对这类应用给出的实时数据集成框架 A。

3.1.1 实现技术

实时数据集成框架 A 的设计有效地结合了基于日志的变化数据捕捉技术、消息中间件和实时数据缓存技术。

基于日志的变化数据捕捉技术：对源系统影响小，代价低，效率高，可以很好地实现针对数据源中发生的变化的及时捕捉。

消息中间件：提供了基于队列的消息传递机制，可以保证消息传递的完整性和一致性。IBM MQ Series 就是一种具有代表性的消息中间件产品。

实时数据缓存：是独立于数据仓库的外部缓存，这样可以避免实时数据集成和查询操作给数据仓库带来的查询竞争和可扩展性问题。实时数据缓存采用实时更新的方式，数据仓库则使用 ETL 工具进行每天一次的周期性批量加载。利用 JIM（即时数据合并）技术，可以对实时数据缓存中的实时数据和数据仓库中的历史数据进行有效合并，以提供给查询工具使用。我们这里重点讨论如何满足实时应用的需求，不涉及到历史数据，因此，为了突出重点，图 2 中忽略了数据仓库部分。

3.1.2 关键组件

实时数据集成框架 A（见图 2）的两个关键组件是变化数据捕捉器和数据集成器。

（1） 变化数据捕捉器

变化数据捕捉器包含了监督模块、规则模块和数据转换模块。

监督模块：负责对日志进行实时监督，一旦有新的日志记录到达，就读取该记录并提交给规则模块。

规则模块：具备对变化数据的过滤和捕捉的功能，它采用预先设定的规则集对读取的日志记录进行判断，如果不满足任何条件，就丢弃掉，如果满足某一条件，就提交给数据转换模块。

数据转换模块：负责把捕捉到的日志记录中的数据进行转换，生成以 LCR（逻辑变化记录）[10]形式表示的记录，然后将该 LCR 放到由消息中间件提供的消息发送队列中，由消息中间件负责把 LCR 发送到消息接收队列。

（2） 数据集成器

数据集成器则包含了监督模块、数据转换和清洗模块和数据加载模块。

监督模块：负责对消息接收队列进行监督，一旦有新的消息（LCR）到达，就把 LCR 取出队列提交给转换和清洗模块。

数据转换和清洗模块：提供对原始数据进行清洗和转换的功能，从而满足建立一致性数据视图的需求。

数据加载模块：数据经过清洗和转换以后，由数据加载模块负责加载到实时数据缓存中。实时应用（如 KPI 和实时报表）就可以从实时数据缓存中读取需要的新鲜数据。

3.2 满足突发请求的实时数据集成

对于实时数据的需求并非可以预先完全确定，在某些特殊情形下，可能会有一些针对某些数据的突发性实时需求。由于实时数据缓存只包含了那些事先确定的需要实时更新的表，所以，很难满足这类突发请求。针对这类请求，我们给出了如图 3 所示的实时数据集成框架 B。

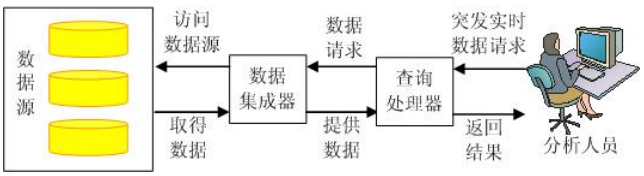

图 3 实时数据集成框架 B

3.2.1 关键组件

实时数据集成框架 B 包含两个关键组件，即查询处理器和数据集成器（如图 3 所示）。

查询处理器：在接收到分析人员的数据请求以后，查询处理器首先对其进行判断，如果该请求所涉及的数据能够从实时数据缓存中获得，就判定该请求为常规请求，从而直接从缓存中读取所需数据返回给分析人员；如果该请求所涉及的数据不在实时数据缓存中，就判定该请求为突发请求，于是查询处理器把该请求提交给数据集成器。

数据集成器：在接收到来自查询处理器的数据集成请求以后，数据集成器直接到数据源读取所需的数据，进行转换清洗以后，把集成的数据提交给查询处理器，再由查询处理器把查询结果返回给分析人员。

3.2.2 与框架 A 的区别

实时数据集成框架 B 与框架 A 有很大的不同，主要表现在以下两个方面：

（1）在框架 A 中，实时数据先被集成到实时数据缓存，实时应用从实时数据缓存中获得所需数据。而在框架 B 中，数据源中的数据不需要被集成到实时数据缓存中，而是直接被集成以后提供给应用，来满足一些突发请求。

（2）从数据集成方式来看，框架 A 采用的是“推(push)”的方式，也就是说，数据源中的变化数据源源不断地推送到实时数据缓存中，供实时应用读取。而框架 B 则采用“拉（pull）”的方式，在突发请求发生时，才到数据源把所需的数据“拉”过来使用。

3.2.3 必要性和可行性

按照数据仓库的设计理念，把操作型系统和数据仓库分离开来的一个很重要的原因，就是为了把查询负载从前者分离出来，使其专门负责提供高效率的事务型操作，而让后者负责处理大量的复杂查询。

现在，我们采用实时数据集成框架 B 来满足一些突发性的实时请求，似乎和传统数据仓库设计理念“背道而驰”，但我们认为，这种做法不仅具有必要性，也具有可行性。

必要性：传统数据仓库采用 ETL 工具进行周期性加载，而且不允许对数据仓库进行更新。为了满足实时数据需求，实时数据仓库引入了外部实时数据缓存[8]等概念，作为实时数据的集中存储地。但是，实时数据缓存中只是包含那些需要实时更新的表，并不是所有表，因此，确定“哪些表是需要事实更新的”就是一个基于实际应用需求调查分析基础上的人为决定。这就意味着，在设计实时数据缓存时，只有一些具有频繁实时需求的表才会被放到缓存中，而那些很少有或根本没有实时需求的表，以及那些暂时无法判别将来是否有实时需求的表，都不会放到缓存中。这就导致在实际应用过程中，很可能会出现一些实时需求无法从缓存中得到数据，这时就有必要从数据源中直接读取数据来满足这些突发实时应用的需求。

可行性：虽然到数据源直接读取数据来满足一些突发实时应用会给源系统增加负担，但是，突发性需求毕竟是非常少量的，总体来说，其给源系统增加的负担是有限的，不会造成源系统的性能下降而无法完成事务处理。

3.3 由用户决定的实时数据集成

实时主动数据仓库的一个很重要的方面就是主动决策功能。源系统中的变化数据经过集成（过滤、清洗、转换或者再增加一些额外数据）以后作为规则引擎的输入，一旦满足预先设置的某个条件，规则引擎就触发特定的动作，完成相应的功能，比如说实时报警。这个过程就可能包含由用户决定的实时数据集成。

3.3.1 典型应用场景

在移动通信领域，营业厅的服务人员负责和顾客达成交易。当顾客购买某个移动产品（如充值卡）时，服务员负责交易数据的录入。交易记录存储到运营系统数据库以后，会被系统捕捉到，并在被集成以后提交给规则引擎。如果规则引擎判断出该顾客购买该产品的次数已经达到优惠的标准次数，就会自动做出决策，给该顾客赠送一定价值的礼品，并把该决策返回到运营系统，显示给该服务人员。

3.3.2 问题描述

为了实现上面描述的主动决策，通常需要把数据源中的变化数据源源不断地捕捉并集成以后提供给规则引擎。这种数据集成方式的一个显著弊端就是大量的系统开销，因为运营系统每天都会产生大量的交易记录。其实，很多交易记录并不需要实时集成到规则引擎。比如以下情形：

（1）服务员自己就能够确定是否需要给顾客提供优惠，比如顾客是第一次购买该产品，就不需要系统来做决定是否需要赠送礼品。

（2）服务员看到营业大厅有大量顾客等待交易，有些顾客在等待过程中开始表现出焦急的状态，这时候，最重要的是提高办事效率，一些由系统根据顾客的相关交易信息而做出的决定（比如向符合特定条件的顾客推销新的产品和服务），可能就显得多余。

虽然，可以在变化数据捕捉时设置一些规则来捕捉所需的变化，但规则的设定通常是比较“粗放”的，不可能细致地照顾到实际应用中的每种细微的具体需求，规则设置过细，也会大幅度增加系统开销，而且经常对规则进行变更也是不现实的。

3.3.3 解决方案

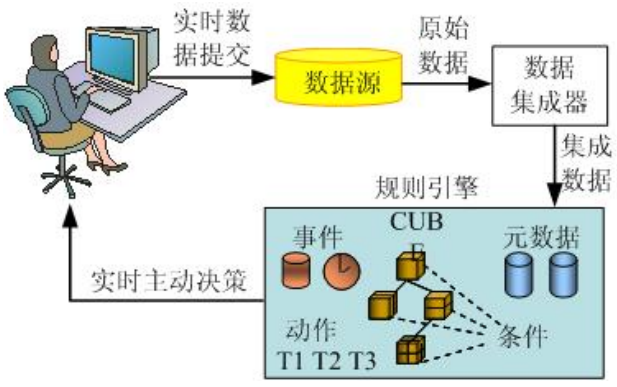

图 4 实时数据集成框架 C

针对上面的应用场景及存在的数据集成问题，我们给出了如图 4 所示的实时数据集成框架 C。如图 4 所示，只有当服务员认为有必要把当前交易记录实时集成到规则引擎，让规则引擎做出主动决策时，服务员才决定让系统把实时数据提交到规则引擎。一旦做出决定，实时数据（交易记录）就会经过数据集成器的处理以后到达规则引擎，规则引擎再根据预先设定

的规则来决定是否需要触发具体动作来指导服务员的业务开展。

针对这种类型的实时数据集成，我们之所以采用框架 C 的方法，是因为这类应用具备一个很明显的特点，那就是在整个主动决策过程中，某个特定的用户（比如使用运营系统的服务人员）即是实时数据的输入者，也是实时主动决策的执行者。这就使得用户可以在实时数据集成中发挥能动的作用，可以根据具体情况做出自己的判断，从而在源头上就对数据进行了过滤，避免很多不必要的实时数据集成工作，大大减轻了系统的负担，也提高了决策的效率。

4. 相关工作

传统的数据仓库采用一次加载并周期更新的方法，在进行数据更新时，不允许进行分析操作。实时主动数据仓库则必须支持实时的查询处理，也就必须具备实时数据集成的能力。

实时数据集成是实时主动数据仓库[2],[3],[4]领域研究的重点内容。近年来，有很多研究人员投入了这一领域的研究，有好几种方法都在朝着这个方向努力。为了最小化更新窗口，Labio[5]等人尝试使用批量（bulk）加载工具来实现高性能的数据集成。04 年左右，学术界发表的一些论文中的方法则以最小化整体更新工作量为出发点，来确定更新数据仓库中的单个物化视图的最优策略。其他方法则重点放在数据/表交换[6]，在这些方法中，ETL 工具获得很大的权限，可以删除表、重加载表和操纵其他主要的数据库系统，同时不影响终端用户的查询。但是，这些方法都没能实现真正实时的数据集成。

变化数据捕捉技术[7]就是一种被广泛采用的可以实现高效的数据集成的技术，并且已在多种产品中得到应用，比如 Informatica 公司的 PowerCenter 8 Real Time Option、DataMirror 公司的 Transformation Server5.3 和 Attunity 公司的 Attunity Integration Suite 等等。

数据变化捕捉技术使得 ETL 过程不再有当机时间，并保持数据的新颖性；它是由变化捕捉代理、变化数据服务和变化分发机制三部分构成。变化捕捉代理负责确定和捕捉发生在操作型系统中的数据变化；变化数据服务则提供过滤、排序、附加数据、生命周期管理和审记等服务。变化分发机制负责把变化分发到消费者。

另外还有一些相关的研究，比如文[1]对实时数据集成进行了研究，并提出了针对数据流的实时数据集成方法。

5. 结束语

实时数据集成在今后一段时间内将仍然是实时主动数据仓库领域的一个研究热点。本文从基于应用的角度出发，提炼出了三种典型的实时数据应用需求，并分别有针对性的提出了三种面向需求的实时数据集成方法，既满足了不同应用需求，也取得了更好的效果。

同时，我们也认为，在不断变化发展的商务运营环境中，比如快速发展的移动通信领域，针对具体的商务应用需求设计特定的实时数据集成方式，会使实时主动数据仓库在实施过程中获得更高的成功率和满意度。今后，我们也将继续在这方面开展更多的研究工作。

参 考 文 献

[5] LABIO,WJ.;YERNENI,R.;GARCIA-MOLINA,H.;Shrinking the Warehouse Update Window; in:ACM SIGMOD Record,Vol.28(2),PP:383-394,June 1999.

[7] I. Ankorion. Change Data Capture–Efficient ETL for Real-Time BI. Article published in DM Review Magazine, Vol 16(1), 2005, pp:23-27.

[8] J. Langseth. Real-Time Data Warehousing: Challenges and Solutions. Article published in DSSResources.COM, 2004. http://dssresources.com/papers/features/langseth/langseth02082004.html.

[10] R. Gadodia. Right in Time. Inteligent Enterprise. Vol 7 (2). pages 26-44. 2004.

[9] R. KIMBALL and M. Ross. The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling, 2nd edition, Wiley, NewYork, 2002.

[6] R. Kimball. Real-time Partitions. Intelligent Enterprise. Vol 2.(1) pages 16-20, 2002.

[1] M. N. Tho and A. M. Tjoa.“Zero-Latency Data Warehousing for Heterogeneous Data Sources and Continuous Data Streams”;Proceedings of iiWAS 2003, Jakarta, Indonesia, pp:55－64.

[2] Michael Schrefl, Thomas Thalhammer: On Making Data Warehouses Active. In: Proceedings of DaWaK 2000: London, UK, pages 34-46, 2000.

[3]. T. ThalhammerM. Schrefl and M. Mohania Active Data Warehouses Complementing OLAP with Analysis Rules Data&Knowledge Engineering Vol 39 (3), pages 241-269, 2001.

[4] S. Brobst and J. Rarey. The Five Stages of an Active Data Warehouse Evolution. Teradata Magazine. Vol 3(1), 2001, pp:38-44.