第 4 部分　实时流 ETL 系统

实时 ETL 系统

建立实时 ETL 数据仓库的解决方案常常需要分辨出某些不太可靠的业务目标，要理解不同的整合技术，要知道其他人运用成功的一些实际的方法，还要拓展工程的灵活性及创新性。这个领域体现了具有新技术、新方法、新词汇的全新理念。显然，这可能是解决问题的关键，而实时数据仓库还将成为早期具有巨大潜力的适配器，使其获得竞争力，降低风险而盈利。本章提出了构建实时 ETL 的四个过程，通过选择合适的实时 ETL 技术、特征、及方法来指导专业实验数据仓库。（

1. 本章调研了实时数据仓库的技术状态的历史及业务情况。提供了一些背景知识，比如我们如何实现，及我们的目标是什么。
2. 其次，它描述了区分组织的实时需求的方式、方法，而这些方法在选择后来的设计结论时最有用。
3. 本章的中心是对一些实时报告以及整合服务机制的评估，并针对每一个方法提供了最合适的技术，以及针对这些技术的优势劣势的分析。
4. 最后，本书给出一个判定矩阵，它按需求分类，并使用了前边所描述的方法。他通过选择技术途径及方法论上指导 ETL 工作组。

必须指出，这份材料并不是建立实时 ETL 结果的诀窍，如上所述，这样的诀窍是不存在的。随着这项技术越来越普及，你必定要应对各种需求，而此时 ETL 还不十分完美，你就要应用你的创造性及积累的个人实践经验，来改革 ETL 结果，使其最适应于你所遇到的挑战。在完成此项工作的同时，你的工作也起到了帮助完善实时数据仓库技术的作用。

流程检查

规划与设计：本章触及到 ETL 系统的计划和设计的各个方面。我们希望本章能作为第 1 至第 10 章的补充。

数据流：本章所有的部分及技术能应用到现存的 ETL 系统框架中。但我们还要强调，从分段批次 ETL 到连续 ETL 是一个深刻的完全的转变。

为什么要有实时 ETL

不久前，设计师们都在强烈维护这个理念，即数据仓库需要给业务决策者提供一组不变的数据，提供一个可靠的信息平台。针对某一时刻数据库的最新报告 t，业务用户都是以事务的生产应用为目标，因此用户必须了解，到昨天为止这个事件所发生的历史过程，同时还要看到许多 OLTP 系统今天所发生的事情。业务用户完全不需要接受这种分割，他们为什么不能只去一个地方就可以得到他们所需的所有信息呢？

那好，许多都改变了，数据仓库已经成为他自己牺牲品 become a victim of its own

success。虽然在数据仓库中事物处理及事物再现之间的延迟一般小于 24 小时，但对于快速运行的垂直运行的系统，这个延迟就太长了。数据仓库要完成决策的任务，要给操作系统反馈回丰富的信息，使这个系统能提高处理执行过程、提供个性化服务并体现面向需求，推动信息的不断更新。

还有很多其他的重要因素，使数据仓库的实践者反思早就出现过情况：

- 客户关系管理（CRM）。现代 CRM 要求一个同步的、稳定的、完整的客户形象提供给操作系统。CRM 又在相当高的层面上直接或间接地服务于客户。尽管市场对于 CRM 卖主包装的需求而言，性能不能超出构架来购买，除非所有面对消费者的系统可由 CRM 单元来付清。尽管市场需要的都是已经打包好的 CRM 产品，但是这种需求是不可能实现现货供应；除非将所有的面向客户的系统都包装上 CRM 的外壳。在业务上，也需要集中消费者的实时信息来跨越传统的处理型 stovepipe 的应用。数据仓库当然完全需要从操作型应用上获取稳定不变的客户信息流来充实仓库中的客户信息，但事实上，越来越明显的是，操作型应用也依赖于数据仓库中丰富的客户信息。因此，可以预言，组织者已经开始开发结构化的交变器 architectural alternatives 使它能够支持没有显著特征但更为复杂的集成情况——操作型数据在操作系统和数据仓库同步进出——越发紧迫。
- 零滞后的企业业务处理模型。这里提倡的是理想化的速度效益和真实的单一模式。在一个实时、零滞后的企业中，信息在恰当的时间提交到一个恰当的地方就意味着获取最大的商业价值。某些人称这个为即时时间系统。正是企业对及时编目、实时供应链及客户化的指令和物理集成业务模式的需要，扩大了对绝对数据流和渗透信息的需求。现实中，真正的零滞后是不可能达到的理想状态，跨越各种生产系统和数据中心来统一信息是需要时间的。但是对于当前许多只能提供反映事物最佳状况的低滞后的数据仓库上的压力却是真实存在的。
- 全球化及互联网。最终，可能最有效的是全球化和网络化的共同作用，他使得数据仓库能在全球任一时间进行数据的存取与操作，能适应对数据仓库日益广泛的需求，这样就强烈地压缩了下载数据仓库需要的时间。

需要输入数据仓库的数据量不断扩大，而业务荡机时间（downtime）又要不断收缩，这对已超负荷工作的热爱数据仓库的 ETL 团队是一个挑战。如果你能设法在全天的时间中细水长流地不断输入数据仓库的数据，而不是猛地扩充数据载荷到因停机时间而收缩的时间期限，这可能会更容易些。所有这些因素促使数据仓库逐渐达到实时状态。

实时 ETL 的定义

实时 ETL 在许多情况下是对提供数据仓库式服务的一个误称。这个服务既不是真的实时也不是 ETL。而这个术语指的是可以在事物处理执行的几分钟内将数据马上但非同步地传输到数据仓库中的软件。多数情况下提交实时数据数据仓库化需要的方法，完全不同于使用定向批处理程序组数据仓库化的方法。整天单一的以更多频次地运行常规 ETL 程序组是不可能实现实时 ETL 的，无论是 OLTP 系统还是对数据仓库。反之， OLTP 系统的逻辑提交任务处理中的数据仓库也是不能这样工作的。让 OLTP 系统在着手它的下一个交易之前，等待数据仓库的数据下载到提交任务上，那简直是奢望。也没有任何同步的或两相的逻辑提交任务能实际跨越不同结构，不同粒度水平的系统。而你仅仅希望在事务可接受的限定时间内，将新的处理输入到数据仓库的特殊的实时分隔区（本章后边会有定义）内，并为每日的操作决策提供分析支持。就目前来说，这个过程就是我们对实时 ETL 的实际定义。

本章为了实现这些目标使用了数据仓库设计师熟悉的主流工具装置，研究了某些实际方法。然而，实时数据仓库是一个新的领域，充斥着各种各样的软件商的要求和较高的风险。本章所研究的实时 ETL 方法试图通过控制预期、重视成熟方法和实施战略工具选择来将风险减到最小。本章提供了在业务处理上和数据仓库可提供的范围内，为目标分配达到最短时间滞后的方法。

实时数据仓库化的挑战和机遇

实施实时数据仓库，给 ETL 的设计者提出了许多独特的挑战和机遇。从技术结构方面看，它要有大批量处理的能力，每天晚上批次处理 ETL 事务，使之形成整日的连续不断的 ETL 数据流。当事务开始依赖数据仓库中事务处理低滞后时间效应时，相对于实用性，系统就需要升级了。如果组织者选择本章所述的实时维管理方式，实用性就成为战略优势了。

从数据结构方面看，实时数据仓库挑战了以离散周期测定值建立的数据仓库的地位。后者只是提供了业务简单映射，而现在我们提倡的是更加综合，更加连续的瞬时信息系统。这种变化的发生是很细微的。例如，事实加载频率从每日一次增至每 15 分钟一次，但是如果事实加载和维度记录能连续进行那岂不是更加生动了吗。那么数据仓库就可以在所有的时间点上及时捕捉到事务处理的信息和他们前后的维度。维度的缓慢变化变为快速变化，数据仓库的支撑（bearing）变得更贴近实际的情况了。事实上实时数据仓库也支撑了实时维度的形成及同步化，它也涉及了操作系统本身的逻辑延伸。

实时数据仓库的回顾

数据仓库的实时方式能够很清晰的追溯到最原始的 ODS 系统。原始 ODS 的目的与现代实时数据仓库非常类似，但实时数据仓库的应用还影响到了新一代的硬件、软件技术的发展。下边将更详细论述这些思想。

第一代－－操作性数据存储

操作性数据存储或称 ODS，是第一代数据仓库。它构造的框架试图在数据仓库中分离出明显的框架结构及应用，来获得低滞后的报告。ODS 是半运行半支持决策的系统。试图在频繁更新和频繁访问的即时响应之间达到平衡。早期的 ODS 架构被描述成一个数据被整合并反馈到下层数据仓库的模式，因而起到扩展数据仓库 ETL 层的作用。后期的架构被描述成集成来自数据仓库 ETL 层数据的消费者，按照已存在的全面的架构和从操作型系统下载数据的及时性，将架构分为从类型 1 到类型 4，以及内在的或外在的 ODS。

实际上，ODS 已成为数据分级、数据清理、数据展现以及执行报表的架构元件汇集区。根据这些不同的作用，它是每一项任务的综合协调器，一个初级的、欠折衷的过渡体。

第二代－－实时分隔区

逻辑或物理的实时分隔区的使用，就像 Ralph Kimball 原来所描述的那样，是一个用于从数据仓库中传递实时分析的有效可用的解决方法。使用这种方法，创建了离散的实时事实

表，它的粒度和维数与每晚加载的静态数据仓库中相关的事实表相匹配。这种实时事实表只含有当天所发生的流量（这些尚未加载到静态数据仓库表中）。

图 11.1 显示了 2 组星型结构分别连接到实时和静态零售销售点事实表，他们的维度相同。

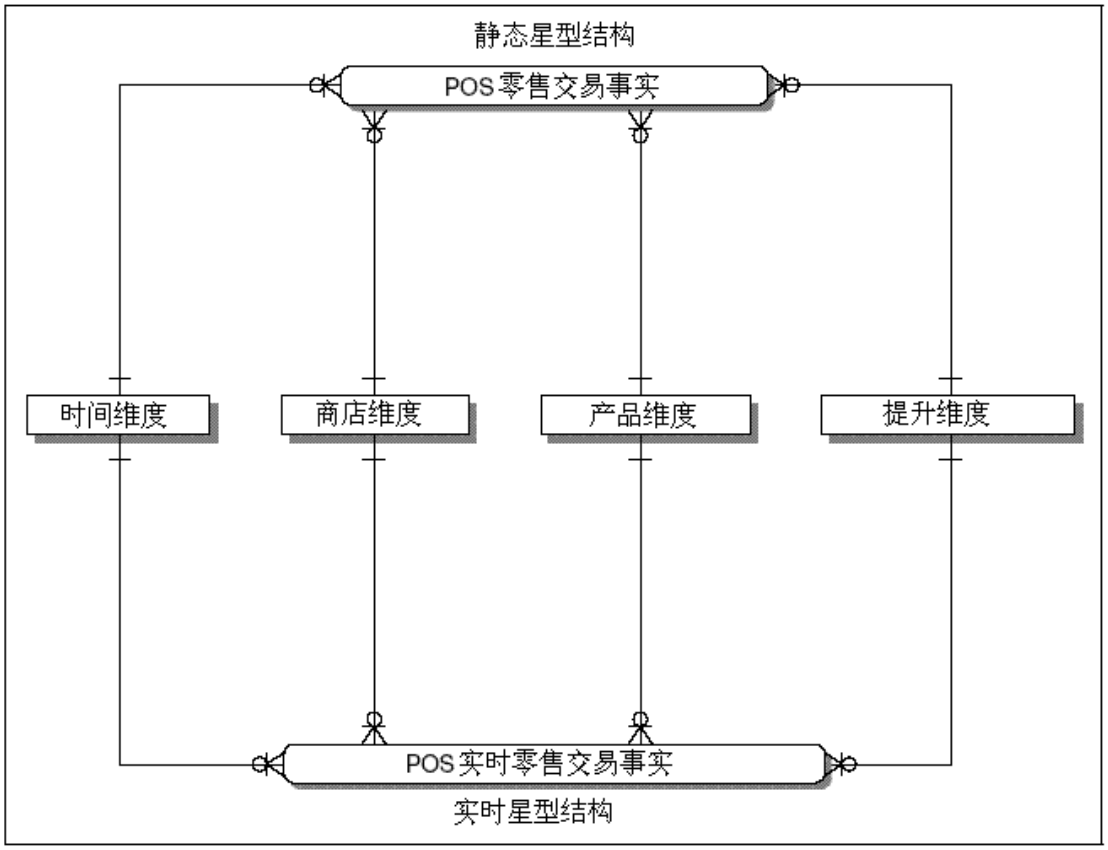

图 11.1　静态和实时星形结构之间的关系

每晚。实时分隔区的内容都被写进静态事实表中，然后实时分区即被清空，准备接收第二天的处理情况。图 11.2 列出了这个过程是如何工作的。实质上，这个方法将 ODS 实时报告的好处带给了数据仓库本身，使其在过程中消除了过多的 ODS 结构的开销。

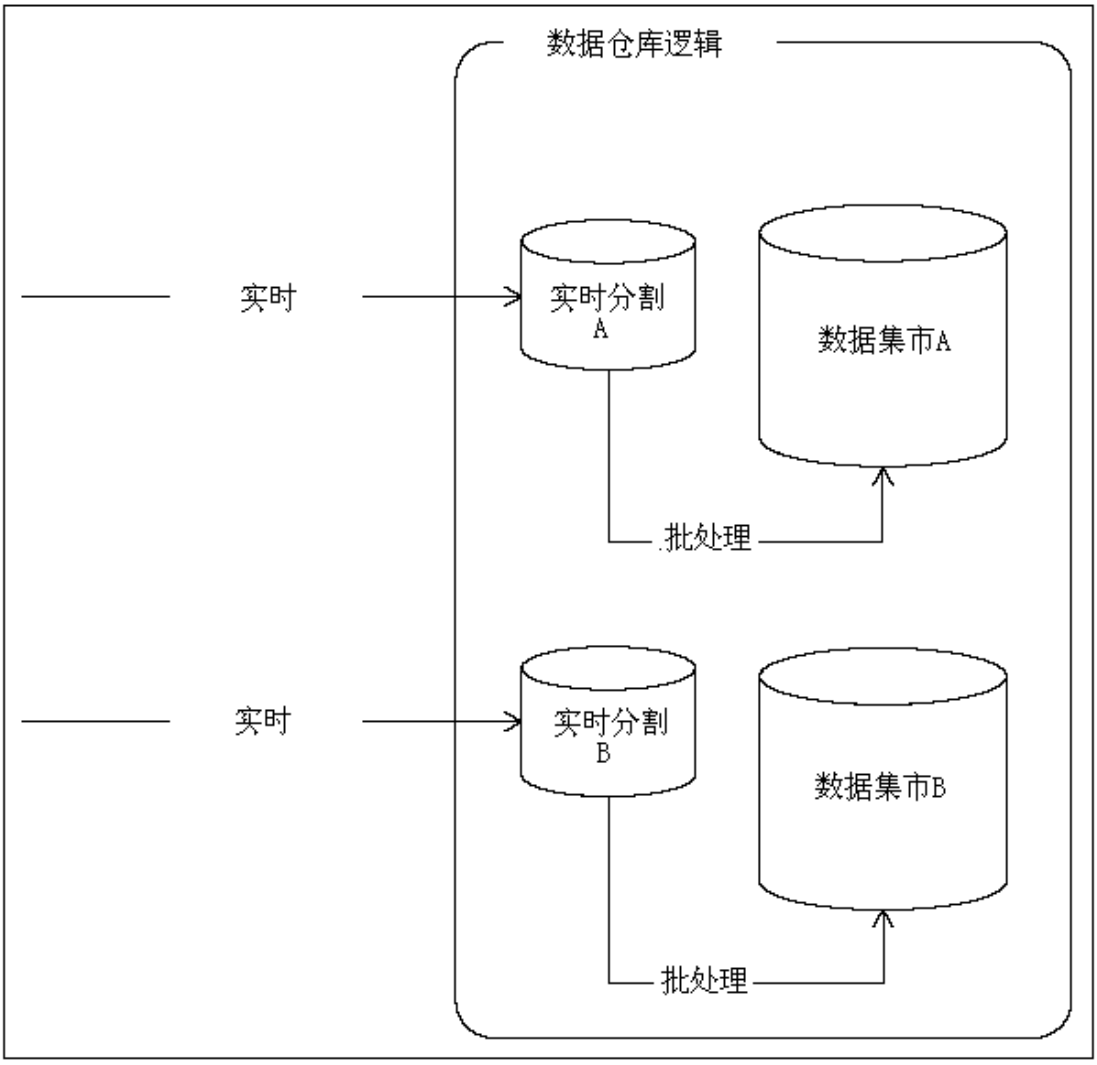

图 11.2　数据集市实时部分的逻辑关系

剩下整个一天的时间，事件逐步插入到实时事实列表中，客户对实时列表的查询,不会不完整,也不会被加载过程所中止。为了减小数据加载作用和查询回应次数所带的影响，实时事件列表的索引是最小的或不存在的。靠限制（一天内）列表中的数据量和将完整的实时事件列表存入内存中可以提高性能。不经意间，在实时和静态列表中的关联事件的观点就产生了。

如果单独的事件记录被逐步地插入到实时间隔区，这时需要某些策略来处理发生在夜间的大容量载荷产生的维度变化。例如，在白天出现新的客户记录，你可能会延迟将这些新变动的客户的一个批次加载到静态客户维数中。因为一般新的客户记录都是在晚上更新到进一步描述客户记录的客户维数中。而实时数据仓库就能选择保持更多的频次断点来改变维度简单映射或可以放弃时间点（point-in-time）的概念而捕捉到所有刚发生的维度变化。

最近，本章所描述的问题与选择合适的方法来处理维度的变化有关，一些有效的方法是将数据在整个事务日内逐步地插入到实时间隔区，及对这种方法的利弊。

最近 CRM 的趋势

CRM 要求对每一位客户系统的历史包括对所有客户的每一个接触点都有一个全面的了

解，并要洞察客户在市场中所面临的挑战及前瞻预期。在过去几年中完整地 CRM 系统已被广泛的应用于商务活动，以支持前期目标，使组织系统中最简单最普通的客户接触点一致化。然而，在分割客户来支持分析系统上，这些系统体现了对组织的重要优势，面对这种优势他们就不理解了。常常是那些比较老的，比较特殊的系统都还存在着，而这些支持客户相互作用的系统不再是 CRM 系统体系了。这些处理过程再没有找到他们返回 CRM 系统体系的方法。CRM 系统体系也是典型的用户智能装备不合格的组织机构，它需要被它的客户认识到作为智能合作者和伙伴，他们还缺乏收集、获得成功的机构，协调一致的客户、和跨企业的智能市场。进一步打破 CRM 体系市场，建立相对应的分析型 CRM 类型的运作型 CRM 扩大了这种分化。业务没有运作或分析型的客户，同样操作型和决策支持系统也要共同运作以服务于客户。

现在需要一种方法，这种方法可以将过去和现在的客户相互作用的组织数据一起快速整合，联合外部市场信息，用某些机制将数据转换为客户智能，这意味着组织中的每个人都可以共享资源。将这些东西集中在一起代表了混合数据仓库的技术和集成技术的应用。

CRM 的供应商敏锐的察觉到组织结构所面临的挑战，所以某些结构就将商务智能能力与他们的操作型 CRM 系统结合起来。这个结果只是基本的，过分简单化的，对结构的保护是困难的，最终难于形成不同于他人的有竞争力的产品。

第二代 CRM 入我们在本章定义的，不是可以购买和安装的应用程序。它要求是一个综合数据仓库，含有所有客户接触点、智能筛选和市场数据的利用；它要求是一个连续获取客户智能的数据仓库，要求是一个在跨企业间分享和同步传递客户信息的机制。能提供这种能力的任务，似乎更应放进同期 ETL 结构后备库中。

维度管理者的战略角色

在数据仓库中维度总线结构中，逻辑和物理独立主题区域的联合维持了维度和事实的规范化，可以通过利用本章描述的维度管理系统完成。。通常维度管理者被认为是定义、维护、给所有数据中心发布特殊维度的一种角色。这些数据中心是经数据仓库中的传输总线相互作用的。

最后，实时数据仓库将把大型目标的准备通道提供给最当前的最重要的数据，并提供给企业的所有客户另外为了给数据仓库快速提交事实记录，在提供实时同步关键维度方面，诸如跨越所有组织的操作系统的客户和产品上，要建立巨大的竞争优势。这种同步信息作用要考虑维度管理作用的逻辑扩展，要是一个有效的、可靠的机构。这个机构提供数据仓库分布式分区的方法和给操作领域提供其他丰富的信息，在操作领域和数据仓库间形成闭环。

在战略实时数据仓库中客户维度的管理者不仅可以向数据仓库逐步插入新的合适的客户信息，而且可以与某些机构合作跨越相关操作系统提供同步客户信息。这个实时客户信息应包括数据仓库本身客户智能。

显然这些是很有雄心的目标。如本文所写，非整套的决策方案或 end to end 端对端衔接的工具装置，显著地简化了建造一个双向 EAI 或实时数据仓库解决方案的过程。但是，像这样的系统已经创建出来了，这些系统的基本结构几乎已经存在并变得更加完善。由这样的系统提供的商务差异的可能性被取消了。它就像当今早期的接收器，具有市场的有利条件，可推动将来更广泛的使用这样的系统。今天建造这样的系统应考虑将来能使这种组织能力发展为实时 EAI 或数据仓库华的解决方法。组织机构处于竞争的压力下，或现在需要通过友好的客户寻求存在差异的市场来越过这一关.

需求分类

很明显，这个话题给我们提供了很多架构方面的思考。提供更为复杂的一系列与主流实时数据仓库相关的优劣分析，，列出实时需求的范围是非常重要的。

在接下来的部分向您展示的是一些 litmus 测试问题，这些问题的答案可以帮助您将您的企业组织需要的实时系统的能力进行分类,并且为您以后的任务选择适当的方法和工具。在本章临近结束的时候我们给您列出了一个矩阵，这个矩阵概括了这些讨论内容并且能够在方法和架构的选择上引导 ETL 团队。

数据的饱满度和历史的需求

降低 OLTP 与数据仓库间反应时间的发展成本和复杂性是遵循收益递减法则的，降低反应时间会非线性的增加复杂性和成本。因此，您需要为数据仓库必需的数据饱满度设置现实的目标和期望值。

同时您还需要彻底明确一个概念，即困难的商业需求设置既不能满足传统数据仓库日常的发布也不能满足 OLTP 系统的操作报告。当您考虑期望的实时数据仓库的需求时请参考下列红色标记的内容。

- 低于 5 分钟的响应：以这样低的响应速度生成报告将不能满足主流实时数据仓库的可靠性。这个时间是持续降低的，但是它同样需要进行超大量的处理和时间将信息从 OLTP 系统中移动、转换并装载到实时分区中。那些完全拥有超过 5 分钟刷新时间的信息量的企业组织应当考虑直接通过操作系统生成他们的报告。

企业信息集成程序不涉及到这种响应时间的限制并能够直接通过操作系统传输几乎最新的报告。然而他们也有其他必须考虑的特点和限制。企业信息集成系统以及这些限制将会在本章稍后讨论。

- 只需要很少或无需历史数据支持的单一数据源需求：这种报告不要求由数据仓库提供集成的历史数据，并且最好由操作系统本身寻址（addressed）。恰当的来说，他们应当呈现非常细致的 OLTP 系统报告（footprint）并且不能明显降低操作性能。如果网络可用的话，这些报告可以通过商业智能入口（BI portal）呈现，这个时候用户群则会认为这些是基于数据仓库的报告。
- 通过已存在的数据仓库生成面向不同用户的报告：这些报告的分发也许需要新的报告词汇和机制，以及会使已经很复杂的实时数据仓库发展成果更加复杂的因素。实时系统不是一个自动的项目终结者，举一个例子来说，实施系统架构师应当意识到在海运中和市场管理中所应用到的商业词汇和矩阵很可能是有很大差别的。
- 特定分析的非实际需求：如果能够稍微对低响应数据部分进行特定分析的话，你就可能避免对一个完整的 ETL 系统流进行重新设计。也许你只需要简单的创建一个基于最新操作系统数据的快速报告，并将这个报告附加在传统数据仓库中根据昨天的数据所生成的报告中。
- 还没有成功实现数据仓库的企业组织：至少从目前来看，将尝试建立一个实时数据仓库作为最初的商业智能发展成果是不被推荐的，仅仅是因为它需要同时精通大量的学科。值得庆幸的是空间数据仓储架构和方法允许企业组织今后可以很好的添加实时报告功能。

以上这些红色标志的共同的象征应该是一个报告，这个报告要求的数据是在昨晚之前更新的而且必须能够满足最少 5 分钟的响应时间。这样的报告也许同样依赖于数据历史、报告词汇、已存在的非实时数据仓库的演示等方面的连续性。这些红色的标志是符合实时数据仓储 ETL 方法所描述的合适考虑对象。

接下来的部分我们将讨论一些关于实时报告的基本要求。

仅仅是报告还是一个综合过程？

企业组织需要一个单一的解决方法仅仅为了报告的目的而把操作数据移动到数据仓库中吗？或者说对于通过在操作系统自身和/或数据仓库间闭环转移的规范化的维度数据也同样有需求？举一个例子来说，将基于数据仓库的客户部分返回操作系统是否需要一个机制？这也许是最为影响选择实时数据仓储方法的一个问题。

当然，任何客户关系管理策略都可能需要共享时间列表和最为完整的客户信息，包括可操作的客户数据（例：最近的销售或投诉信息）和源于数据仓储或数据挖掘的客户市场信息，比如客户分割、资料以及寿命值。在企业组织可操作系统和决策支持系统间闭环循环的过程中，实时报告是否为第一步骤？

仅仅是事实还是包括维度变化？

商业人士和空间维度数据仓库架构师将世界定义为事实和维度，但是 OLTP 系统不会做出如此明确的区别。但是，作为工程师来说，你必须明白并将 OLTP 系统的商业操作按照终端用户的喜好做好分类并进行适当的设计。

实时报告需求是否专一的关注在诸如最近提交的订单、最近发送的电话、近期完成的股票交易以及近期拨打的销售电话等最新的事实吗？还是说它同样要考虑到一些诸如客户或产品记录的更新维度一类的最新操作？如果实时空间维度变化是报告所必需的，那么他是否是连续或缓慢变化的？换句话说，用户群在面临做出操作的时候是否需要一个准确的维度表，或者当更新发生时，是否所有或部分维度更新会被破坏性的进行过度重写？报告是否需要有可重复性？第一类型慢慢地改变维度，对维数属性的变动破坏性地重写期望值，导致连续重建历史的数据仓库，报告事件不是看他们操作的时间而是要看今天的维度状况。在这种情况下，通过不同的时间点相同的维度生成的报告可能会有轻微的或者显著的不同。如果第一类型转变用于聚集演算依据的话会造成历史聚集的无效，因此这种变动是危险的。

类型 2 和类型 3 在某些点上会缓慢的改变维度从而使得粒状维度图更加清晰，也许是每天都在改变，但是他们仍然不能捕获到发生在抽取时的维度变化。实时维度刷新可以使粒度提升到每隔几分钟或者捕获所有维度变化。

建筑学的含义不是微观的。通过采取加快捕获维度变化图像频率的方法，数据仓库远离了他的早期状态即周期性测量（快照）系统，继而转变为 0 响应时间的决策支持理想模型。当数据仓库和应用综合化技术开始相结合的时候，数据仓库实际上将成为管理企业操作的系统的一个真实的逻辑引伸。暂时作为当务之急的，ETL 系统大概需要被设计成为能为被测量的事实提供尽可能接近 0 的响应速度，但是同时还要允许部分或所有维度属性可以进行批量更新。

告警，持续处理或者无事件?

虽然通常情况下 ETL 系统对于维度化数据递交前台的界限有非常明确的定义，但是在许多情况下一个实时系统无法有这个界限。The architecture of front-end tools is affected at the same time。现有三种数据交付范例，都要求通过端对端的观点来达到所有从初始源到终端用户的方法。

- 告警：通过源数据条件强制用户界面进行更新。
- 持续处理：终端用户的应用程序为了实时更新屏幕而连续探查源数据。
- 无事件通知：如果某个特定事件在某一间隔时间内或者由于某些特定条件制约而没有发生，终端用户将被通知。

在以上每一个事件中，实时 ETL 系统将通过发送通报或者接受直接请求的方式与终端用户的应用程序相连接。

数据综合化或应用程序综合化?

假设实时数据仓库的要求同样适用于某些横跨可操作系统的综合的测量，您需要将您的要求分类为数据综合化或应用程序综合化。

- 总的来说，能够通过简单的在数据库间移动数据来满足综合化叫做数据综合化。通常，这些解决方案都是点对点的形式，对于异构数据库通过 ASCII 码文件抽取、触发、数据库链接和网关执行，或者通过同类数据库的复制服务或者表格快照方式执行的。精炼的说，在所有参与的应用程序间数据是共享的，完全绕过应用程序的逻辑。一些高结束数据综合化(higher-end data-integration)工具为数据的预定和移动提供集中化管理支持，支持更多企业控制和管理点对点数据综合化事务。
- 应用程序综合化（有时也叫做功能综合化）可以被描述为通过使用一些常见的中间设备把应用程序连通化从而建立新的商业解决方案。中间设备是一组可独立应用的软件组，可以提供捕捉、寻址以及在应用程序间执行可操作信息的手段。总之，连接器或适配器被用来连接所有参与的应用程序到综合化网络上，而 broker 则是用来根据公告和协议规则发送消息的。

点对点对比集线器广播

如果您的实时数据仓库同样支持某种程度的应用程序（或者功能）综合化，那么在选择构造方式上的一个重要因素是发布的数量和您预定的在未来（比如 24 个月）所期望的可支持系统。这个数字能够帮助您决定一个相对简单的点对点解决方案是否可以满足或者是否需要一个更加强壮的构造法。

图 11.3 告诉我们，即便只有相对少量的应用程序交换数据，点对点解决方案都可能要求有极大量的数据交换界面，每一个界面在来源或目标程序改变时都需要进行维护。向综合化网络添加应用程序同样需要为所有公告和预定应用程序添加数据交换界面。尽管如此，拥有一份短小的应用程序列表的企业依然需要规范化的维度综合化，并且他们期待这份列表依然稳定，并在未来能够找到一种相当有吸引力的点对点的综合方法。它避免了创建 EAI 中间设备组件的复杂性，并且能够在一些数据综合化技术比如捕获、变换、流程（CTF）工具的应用过程中被部分支持。

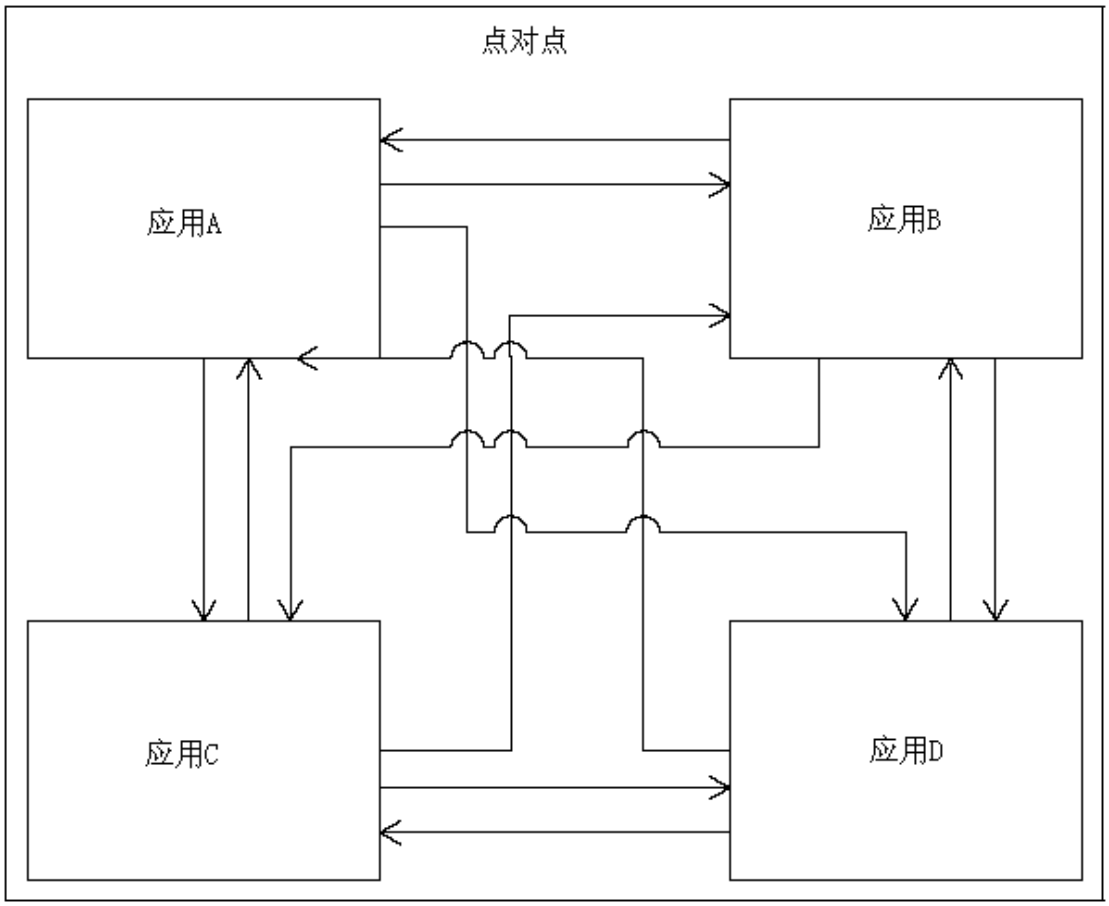

图 11.3　点对点应用集成

与点对点的架构对比，通过方法可以使用户界面和跨系统的附属物降到最低。（见图 11.4）

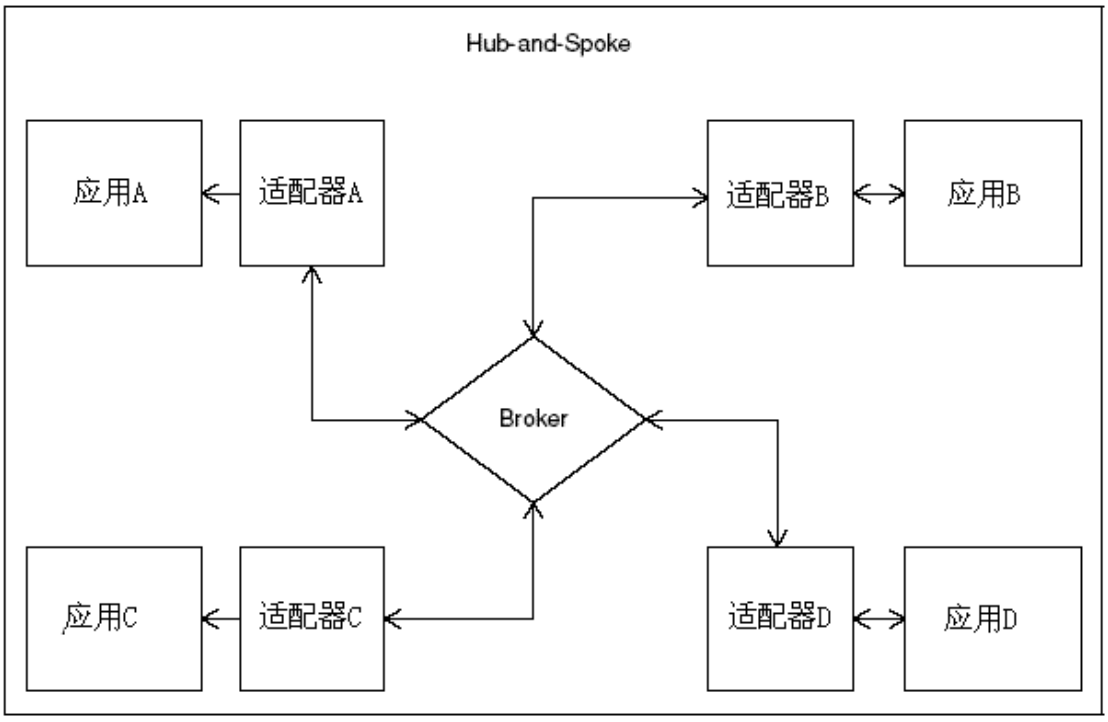

图 11.4　集线器广播应用集成

然而，建立 EAI 中间设备适配器和 broker 组件所带来的额外负担也是很大的。每一个综合化网络中的应用程序都需要一个可以将特定的操作转变为通用的消息并解释和执行的适配器。任何一个相关的主应用程序或者通用消息的改变都需要对适配器进行维护。

决定选用点对点还是的构造法并没有一个明确和快速的规则，但是那些期望综合化横跨 3 种或更多应用程序以及那些期望在未来增加综合化网络参加者的企业组织可以考虑集线器广播 EAI 构造法。这个公认的有弹性的界限同样可以有企业组织的适应程度以及对 EAI 传讯技术的承诺度来调整。

客户化数据清洗考虑因素

如果企业需要实时清洗并保持客户化数据的规范化，在选择方法上你需要额外考虑。在产生新的客户化关键上系统是否有一些适当的集中方法，以保证没有多余的客户化记录产生。这样的系统相当少见，而且它时常作用于管理数据仓库中的客户化维度以提供此服务给企业。

假设此系统是不适当地，在适合实时的客户化维度管理上，它可能产生假定匹配 (双倍的) 客户记录。目前，市面上有一些决定性和可能性的匹配工具可以满足这些需求，但是很不幸的是，这些工具大部分只能在批处理模式下运行。客户化清洗的有效性支持确认目标地址，倾向于分割虚假记录，验证有效记录，此外，一类（householding）需要批处理。无论如何，这样已经很接近实时程度了，构造运行频率发生的微批处理 microbatches 架构，稍后在本章节中描述，首先我们分析有效的输入和输出。

实时 ETL 过程

通过创造性的回收利用一些已有的 ETL 技术和工具，一项成熟和前景广阔技术的可用于满足实时数据仓库需求。

小批处理 ETL

传统的 ETL 过程，在这一本书各处被描述为以文件为基础的方式，在每日,每周 和每月的批处理报告需求中被非常有效的选择。新的或更改的操作（事实记录）按照 en masse 运行，在每一次的下载中迅速捕获维度。因此，数据仓库中在批处理过程中更改维度是不可能的。因此，ETL 不适合于对于系统需要低延迟报告的数据和整合申请，对于需要获取更详细的维度变化的系统也不适合。但是，传统的 ETL 过程对于需要更多偶然潜在需求和复杂整合挑战的系统而言是简单的、直接的并且被验证为有效的方式。图 11.5 说明了传统的 ETL 过程。

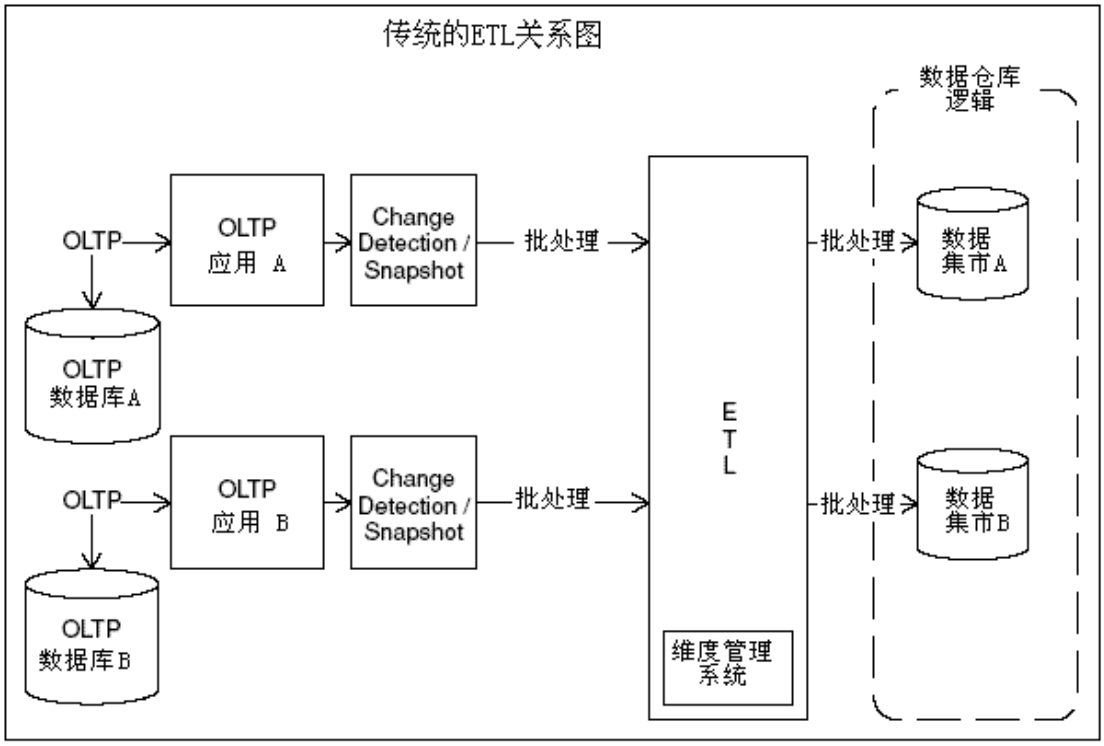

图 11.5　传统 ETL 方式

小批处理 ETL 与传统的 ETL 除了批处理频率被增加外（很有可能是每小时一次），在其他方面是非常类似的。频繁的微处理模式是以其它传统的 ETL 方式和直接将实时分类数据输入到数据集市来处理的。实时分类的数据每天一次备份到静态的数据集市并被清空。图 11.6 说明了微处理 ETL 过程。

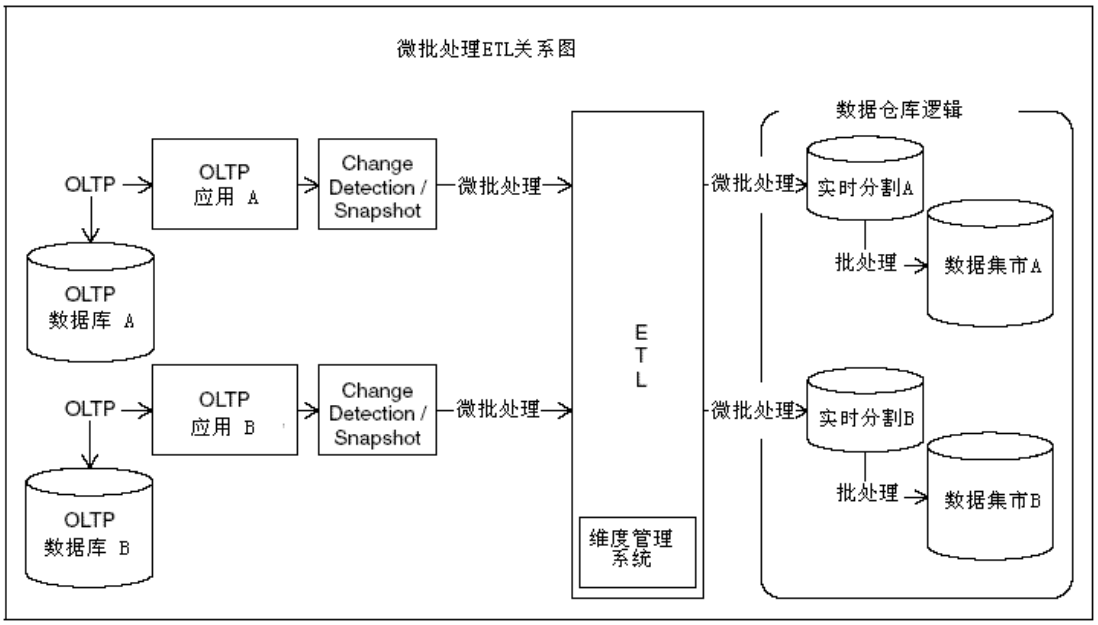

图 11.6　小批处理 ETL 方式

维度管理系统在类型 2 或 3 中慢慢地产生新的维度标准（images）更新维度，但是由于运行频率的增加，一整天更新的维度可能很快变更而且深入增长。在所有的字段中经常使

用缺省值替代只为新实例产生的维度记录，忽略对维度的变更是很无奈的选择。这种折中的选择能满足于在一个给定的日子产生少量新记录的系统，也能涵盖从前一天起维度前后变化的潜在关系，但是它明显的地弱化了实时报告的一些优势。如果不可避免，处理快速变更维度的唯一实际的解决方法是正确使用细化维度，对于最频繁变更的大维度属性，使之产生单独的维度，而且藉此减少在 ETL 过程产生的新维度实例需要的数量。

有趣的综合选择是，在当天内，维度的专有备份与实时分类关联，此类变更的维度被视为类型 1。在当天内触发的简单重写产生变更。在当天工作结束时，直到关注在数据仓库的静态部分中的维度备份，任何的此类变更将被视为类型 2。举例来说，实时数据仓库的一个用户可以立即设置一个信用-价值的指标。

微处理 ETL 要求一个全面的工作管理，计划安排，从属关系，和减少错误的方式，在大部份的时间是无人看管下，对于运行给于足够强健的支持，在处理最常见的载入数据事务方面能够执行数据仓库发布的策略。一些可控事务的实效性支持这一个功能 ，但是传统发展型的任务很可能与微处理 ETL 数据仓库自动操作的任务相背离。

微处理 ETL 在 OLTP 系统上，也要求更加频繁检测新产生的和更新的交易记录，因此应考虑和谨慎处理加载对整个系统运行的影响。

以下是识别微处理 ETL 载入实时数据仓库的更新候选记录的也一些方法：

- 时间戳：验证性和有效性，时间戳主要作用于实时微处理 ETL,用于抽取区分候选数据产生和更新的记录的操作系统。简单任务中,当 ETL 过程运行时，这一个方法确实作用于在操作系统的所有更新发生时的时间戳频繁写入和读出。索引时戳提高了读出而且减少读出的开支，但是有时会惊人的增加的插入和更新操作的开支。ETL 工程师必须衡量这些关系。
- ETL 日志表：另外的方法是在 OLTP 环境下创建触发器，将新的独特的传统标识符和被改变的记录插入到一系列的特定的 ETL 日志表格中。这些专门的日志独立存在以加速 ETL 处理，并且通过微处理 ETL 进程来决定哪一列与早先的微处理相比发生了改变。ETL 日志表格包含新的或是被改变的空间记录的独特标识符，还可能有状态值、时间戳以及一个对被改变的记录进行了最后操作的微处理 ETL 进程的运行标识符。微处理 ETL 进程将 ETL 日志表格加入到 ETL 运行标识符为空的可操作表格中，提取总值列然后删除(或者移入运行标识符)被提取的 ETL 日志表格。通过使用这种方法，由触发器驱动的插入行为不会过度使用 OLTP 系统，降低了企业花费在可操作系统上的费用。
- 数据库管理系统(DBMS)日志筛选：被作为有备份和补救作用的副产品而创建的 DBMS 审计日志文件，有时可以通过一种叫做日志筛选的特定功能来辨别新的和被改变的操作。有些日志-筛选功能可以及时地在某些特定点，有选择的提取和重建应用于数据库表格的 SQL 声明，不仅允许 ETL 知道哪些记录在最后一次提取后发生了改变，而且还知道这些记录中的什么元素发生了改变，信息可以被 ETL 进程用来直接在集结地修改目标表格。
- 网络探测器：这些功效可以监控某些我们感兴趣的网络流量，过滤并记录看到的这些流量。网络探测器经常被用来捕捉网络点击流量，因为它消除了从多台服务器中将网页日志联合在一起的需要，提供网络浏览者的 sessionizing 并且可以增加由动态网页反映出的实际网络内容的可见性。网络探测器是一种能够对在任何地方出现的要求数据仓储支持的流量进行选择的 ETL 过程，包括电信呼叫路由、制造业基础工作流程或者 EAI 通讯流量。

对于要求满足在每小时反应周期内都不出现在一个小时内维度更新的数据仓库需求来说，微处理 ETL 是个很不错的选择，而且它不要求在数据仓库和可操作系统间的维度数据的双向同步。显然这是提交近似实时数据仓储报告的最简单的方法。

企业应用集成

企业应用集成（EAI）是出现于复杂频谱的高端，有时也叫做功能集成。EAI 是一套支持真实应用程序综合化的技术，它允许每个独立的操作系统通过潜在的不同新方法进行互联操作。EAI 需要建立一套典型的能够通过消息的形式，在综合化网络中跨多种系统推动商业操作，并能够使该网络中所有系统从技术和独立性上完全独立与其他系统的适配器和 broker 组件。应用程序特定适配器负责处理所有创建和执行消息所必需的逻辑，brokers 负责以适当的方式并根据公告和条款规则对消息进行寻址。适配器和 brokers 之间通过独立程序消息进行通信，通常以 XML 来实施。当一次重大应用事件发生譬如顾客纪录的更新，触发器被触发，并且应用程序的适配器创造一则新消息。当适配器收到一个包含他所选择接受信息的消息时，比如从客户维度管理系统中得到的新近被确认的客户记录，适配器还负责在各个应用程序中启动相应操作。Broker 在适配器之间发送消息，消息的发送是基于标准协议。消息队列经常被部署在各应用和适配器之间，适配器与 broker 之间，形成异步消息的分段传递部署，保证了集成网络中传输和交易完整。

在图 11.7 中，应用 A 和应用 B 是独立运作的两个系统，但是它们之间可以通过 EAI 交互数据，相互操作。

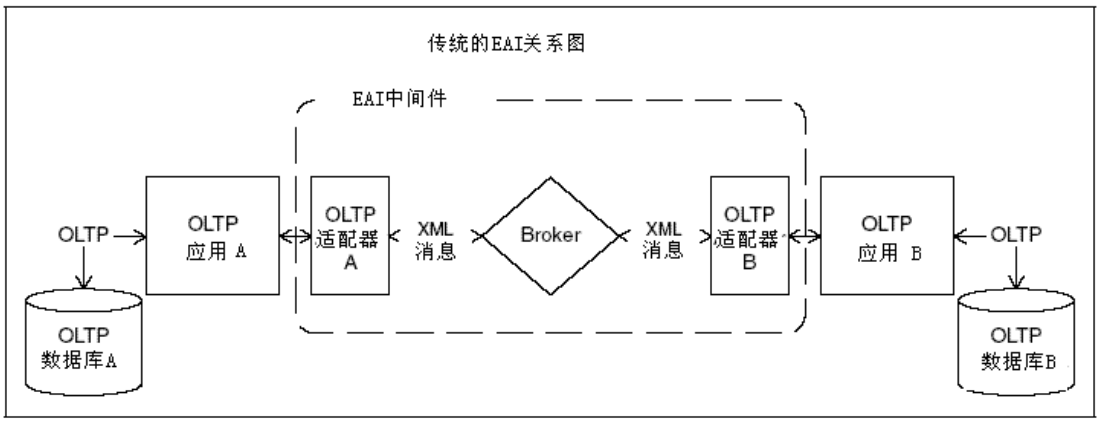

图 11.7　传统 EAI 架构

例如：修改系统 A 中一条客户信息，会触发到 A 的适配器，适配器生成一条描述该改变的 xml 格式消息，并把它发送到 broker，如果系统 B 从系统 A 处订阅了该种信息，那么 broker 就会向系统 B 的适配器发送该消息，适配器做出指令会影响全部或一部分有关该客户的信息。系统 AB 之间没有直接的交互，他们各自适配器负责捕获信息，翻译，做出指令。这是一个强有力的概念，他扩展了 EAI 网络。引入一个新的应用到 EAI 网络中，仅仅只需要一个新的适配器和规则。我们这里强调的“仅仅”，并不意味着适配器的作用是微不足道的。恰恰相反，每个适配器必须拥有在其主机系统上处理潜在复杂事务的能力，以及能够处理大量并行事务的能力，这些并行事务是由那些基于同一逻辑数据的应用产生的。假如不考虑集成方法，事务一定是在结构中的某个地方被处理，由 EAI 的适配器来做这件事是最佳的选择。

EAI 技术与实时数据仓库相结合，能发挥更大的功效。他们能够支持重要数据的同步处

理，比如那些在各个应用上的客户信息。提供高效的方法，意味着源自信息资源的分布式数据仓库将横跨整个企业，譬如新客户的分割值。

实时-EAI-数据仓库通过维度管理将“单片”ETL 处理模块化。将维度管理划分为独立结构单元，每一个单元有自己的适配器，（数据集市适配器）负责数据集市实时分区的上的数据转换和装载事务。图 11.8 是关于实时 EAI 数据仓库的图示。

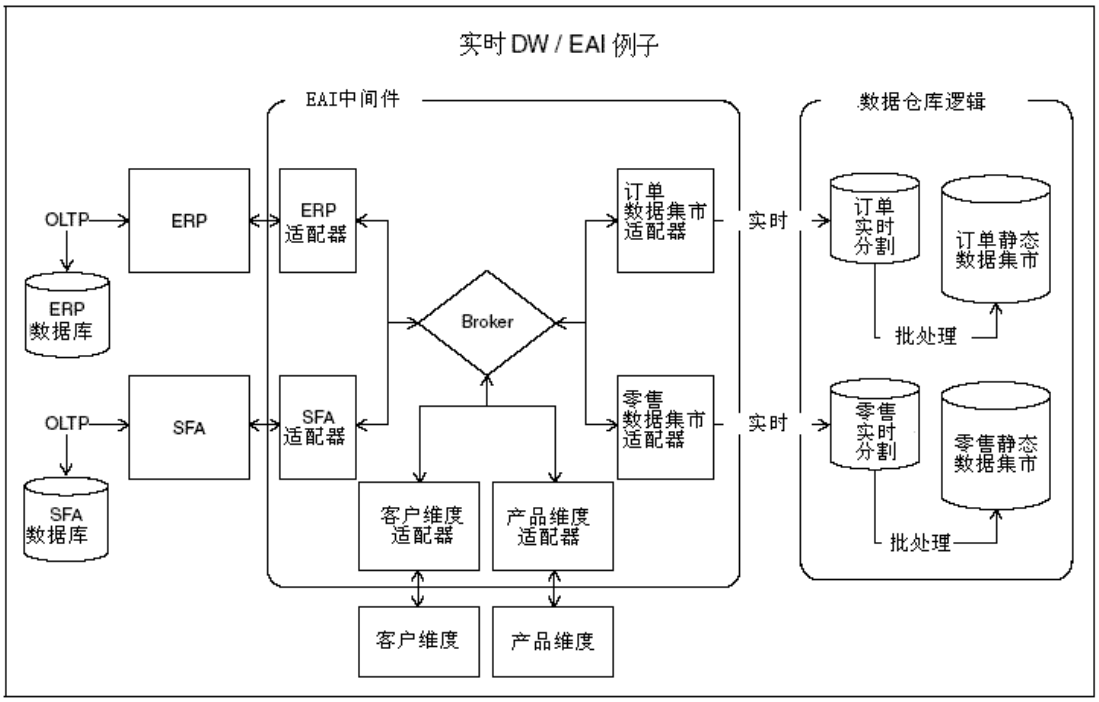

图 11.8　实时数据仓库 EAI 架构

一个典型的实际案例，可能涉及到一系列的 OLTP 系统，比如企业资源计划 ERP，销售自动化，客户产品维度管理系统（进行实时清洗和备份），订单销售主题的数据集市。

任何一个 OLTP 应用的客户或产品变化事务都会被适配器所捕获，发送不规则的维度消息给 broker,然后 broker 将维度消息发送给那些“订阅”了维度消息的系统，特别是维度管理系统。 系统将维度纪录以某种规则一致化，然后适配器将规则的维度消息返回给 broker，broker 再把规则的消息发送给所有订阅消息的系统，特别是数据集市和 OLPT 系统。

根据这个例子，ERP 系统更新一条客户纪录，ERP 的适配器检测到这个变化，生成标签为来源于 ERP 的非统一客户交易的 XML 格式消息，并把它发送给 broker. Broker 转发这条消息到客户维度管理，特别是从 ERP 订阅了不规则消息的系统。客户维度管理系统收到消息后将其放入工作队列（或堆栈），客户维度管理者处理这条事务，如果它导致一条或者多条规则客户纪录发生改变，适配器将捕捉到这些发生的变化，将修改后的客户纪录打包到源自客户维度管理的统一客户交易消息中，把他们发送到 broker。假定那些订单数据集市，销售数据集市，ERP，SFA 系统都订阅了源自客户维度管理的统一客户交易消息，那么 broker 将复制消息，并分发到所有的系统中。每个的系统的适配器对客户记录的变化作出反应，改变对应应用的纪录。

ERP 系统和 SFA 系统对统一后客户纪录的包容性，可以通过适配器来改变自己系统内部的客户数据改变，因此而触发新的一系列 EAI 事务。

在集成网络上，OLTP 系统或者维度管理系统上设计一个可选择的发布策略时，一定要避免无限循环，或 race 情形。

实际交易也会被 OLTP 适配器所捕获，作为订单和销售事实消息发送给 broker，然后 broker 再将消息发送给所有消息订阅者，特别是数据集市，数据集市适配器作所有需要的转变并将新的处理直接插入到数据集市的实时分割区。EAI 对于同步处理关键业务信息来说是一个强大的方法，逐步输入数据集市和发布，以及面向客户的 OLTP 系统的源自数据仓库的分布。但是，这种方法复杂而且实现的成本较高。对于那些有低延迟报告需求的组织来说，EAI 是一个非常好地解决办法。对于这种组织来说，当天维度更新数据的损失是不被允许的，或者要求数据仓库和操作型系统的维度数据的双向同步。

使用 EAI 构架来解决海量交易数据载入数据仓库，如果每个交易都是独立封装成为一条 EAI 消息，每个消息占用很大的系统消耗，那么 EAI 方法显然会效率低下。在提交给这种设计方法之前，要明确是否会出现大量的消息通信甬堵，同样的也要明确你的 EAI broker 接口允许事务数据的契约交涉。

获取，转换，和流动

获取，转换，和流动(CTF) 是数据集成工具一种新的相关类别，这种数据集成工具被用来设计简化实时数据在不同数据库中转移。事务性应用的应用层是旁路的bypassed. 相反地，直接库对库的交换被执行。事务，包括新事实和维度变化，可以直接快速从操作型系统向数据仓库进行数据移植，只需几秒钟时间。CTF 工具的转换功能基本上是基于与现今成熟 ETL 工具的对比之上的。所以实时数据仓库的 CTF 方案经常是操作环境中转移数据，使用 CTF 工具作数据转换，然后转移数据。数据转换工作包括数据格式的标准化，数据类型的重新定义，缩短或增长字段长度。一旦数据被转移，超出 CTF 工具能力的附加转换将会出现。后续的转换可以被小量批处理 ETL 调用或者通过触发器通过向传输区插入数据来触发。

在每个转换中，记录被直接写入到数据集市中的实时分区数据表中，这些并发的转换可能包含数据验证，维度数据清洗和匹配，维度数据的代理键查找，以及建立新的维记录需要慢慢改变，见图 11.9 所示。

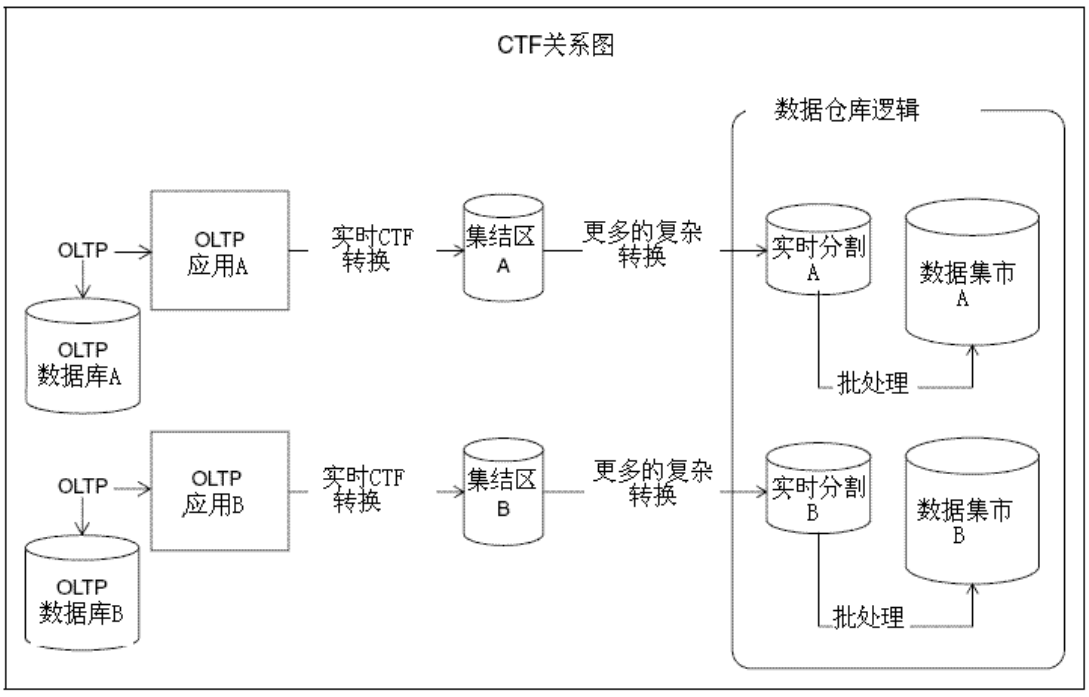

图 11.9　CTF 架构

一些 CTF 工具能够简化信息从数据仓库返回到操作性系统的批处理，例如客户数据的周期性更新。这是应为他们旁路 bypass 处理应用逻辑层。利用这些特征把批处理的负担转移到 CTF 管理员，来确保结果性更新数据不会混淆操作性系统的处理或交易事务的遗漏。

CTF 对于那些有实时上报，少量数据集成需求的组织来说是一个很好的方法，CTF 可以很好处理核心应用软件较低的共同活动时期,可以让数据同步发生混乱最低。CTF 融合了 EAI 的好处部分，同时避免了复杂。

企业信息集成

企业信息集成（EII）是另一种新方法来帮助企业较快提高商务智能系统的实时上报能力。在某种程度上讲，EII 是一种实质上的实时数据仓库，OLPT 系统中最近数据的逻辑视图，向业务用户展现适当的架构分析，通过 ETL 转换分发。

在传统 ETL 中，在 OLTP 系统中定义一系列的源结构，在数据仓库中定义目标结构。根据时间计划，一般是在晚上，触发 ETL 进程，ETL 工具抽取数据，转换格式，然后将其装载到数据仓库表中。EII 也是大致依照相同的原理来工作的，不同的是，EII 的目标文件可以是报表，电子表格，或者对象链接或嵌入数据库，或者 XML 对象。EII 触发器是当业务分析者需要更新第二个操作性信息时被触发。无论是否需要达到第二层操作型数据，EII 触发器都是通过业务分析员推动。EII 系统实际上产生了一系列的数据库查询，基本上是通过 SQL，在请求的那个时刻，申请所有被指定转换的结果数据，然后将数据释放给业务用户。

它的作用表现的相当有意思：一个零延迟报告引擎需要联合成熟的 ETL 工具数据集成的充足作用来增强。

还没有这样的先例：在 OLPT 系统中的数据同时存在于 EII 系统中。所以，描述趋势的报表还是要在数据仓库中产生。数据仓库本身也可以被定义为一个 EII 系统的源信息，所以，操作型系统的实时数据与数据仓库中历史数据的集成至少在理论上是可行的。EII 的转换能力，虽然很强大，但不是没有界限。它并不能在线支持所有的现代 ETL 和数据清洗功能（例如，模糊匹配），所以相应的降低了对他的预期。同样的，由于抽取的数据直接依靠 OLPT 系统，因此在抽取的频率和复杂度的管理上要参照在 OLPT 技术框架上的脚本大小的管理。

EII 的另一个优势是他的 ETL 继承性。EII 可以被看作是数据仓库的一个有效的原型设置。组织从同一个供应商那里选择 ETL 和 EII 工具，可能会发现成功的 EII 与数据仓库技术联系在一起的，数据仓库中的数据转换规则复用

如果将成功的 EII 主题转变成数据仓库主题范围可以重复利用在 EII 工具中开发的数据转换规则。组织选择从同一供应商提供的 ETL 和 EII 工具时，会发现成功的 EII 主题区域需要发展为数据仓库主题区域，后者可以通过复用已经成熟的 EII 工具的数据转换规则来推动。.EII 在全面数据仓库商务智能系统实时数据上报部分上也扮演着支持者的角色。在数据仓库和 EII 之间使用规则的维度和事实,作为维度化数据仓库总线结构的一部分，允许这些系统相互协作。On the fly，依照维度和事实，然而，说来容易做来难。在这种情况下，在最低程度上，你必须在 EII 查询上设置事实约束，拒绝已经装载数据仓库的事实来避免重复计算，联合新维度记录关注展现还没有加载进数据仓库的实时实际数据。

EII 对于那些要求零延迟实时上报的组织可能是一个非常关注的方法。对组织来说也许会有价值。那些组织相信他们需要发展实时数据仓库，但是并不确定它们的实时统计业务需求或业务需求改变的十分迅速。

最后，EII 对于改组中的企业也是引人注目的选择，它可以尽可能快的满足实时集成操作性上报。

实时维度管理

实时元（维度）管理，多次在本书中被提及，主要在客户信息中使用，转换新引入的客户信息，这些客户信息有可能是不完全，错误的或者冗余的。将这些客户插入到标准的客户记录中去。标准并不意味着完美，但是他们的维度纪录已经达到了组织能够达到的最好状态。在实际中，这就意味着所有的合理方法被用来消除冗余，除去脏数据，尽可能的将数据编译完整，同时指派代理数据键值。图 11.10 是实时维度管理的一个普通示意图。

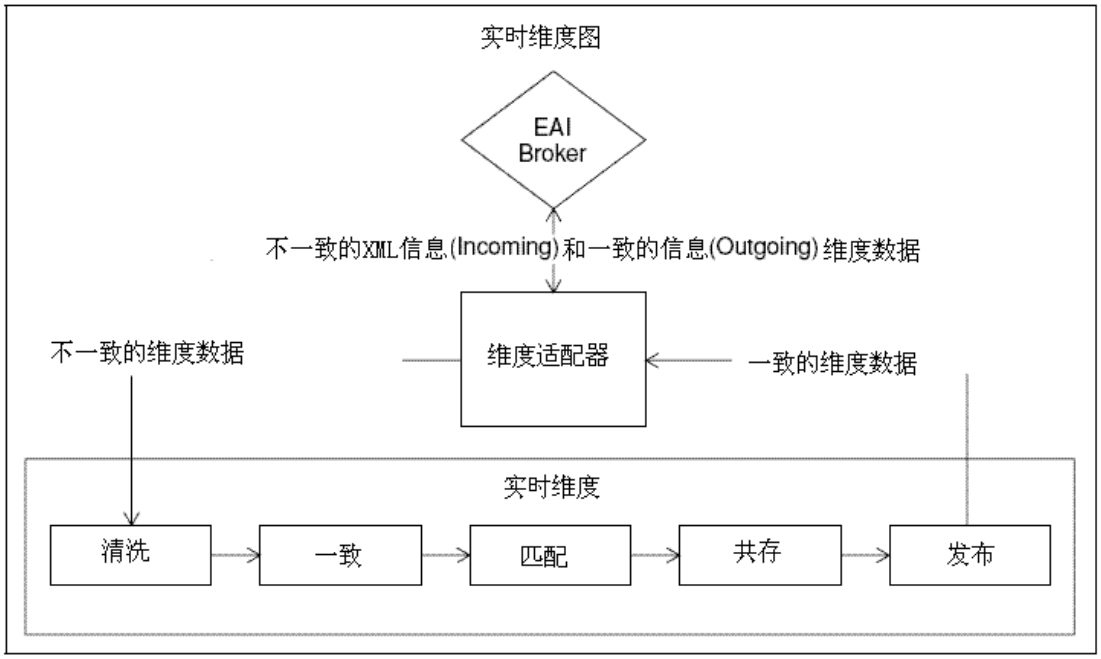

图 11.10　实时维度管理架构

EAI broker 是相同的 EAI 中间件组成部分，而中间件是在另一个 EAI 图表中被描述。.EAI broker 负责在适配器之间根据发出和接受规则“路由”消息。在维度管理中，EAI 是根据操作性系统的不规则维度零散信息改变到维度管理者进行路由处理，并将规则的维度改变返回需要接受信息的操作性系统上，或者是数据集市上。从实时维度管理得到的规则的客户信息一定也被包含在一系列 OLPT 系统的历史键值中，这些 OLTP 系统是被规则化进程连接在一起的。接着，这些键值被 OLTP 系统的适配器用来指认出如何应用规则化后的客户信息到各自的应用上。这些 OLTP 的结果性改变可以导致新纪录的产生，对原有数据的更新，或者两条或更多数据的合并，这些数据被维度数据管理认为是冗余的。在实时数据集市装载软件中也可以使用出现在引入到数据仓库替代键的事实记录的原有键值。

这种维度连续不断的变化是参与集成网络的各系统保持同步的一种机制。这种集成网络所发挥的作用能够超过简单的同步化。从 CEO 的观点来看，这些应用要协同工作。

除非是计划好的，一种反馈器能够在参与集成网络的各系统之间发挥作用。在一条非结构化维度纪录被维度管理者处理之后，应用生成一条维度信息记录自身的改变。你一定要小心处理这种反馈作用以及各种不利的处境，包括维度规范化信息的无限循环。一个稳固的 EAI 结构的重要性在于，能够解决这种类型的事物。你一定要建立一套良好的策略来管理消息。以下是一个策略的例子：集成网络的各参与的应用，包括 OLTP 和维度管理者，在要更

新现有的维度记录或生成新纪录时，这些应用选择发布消息。策略暗示了维度管理者，例如，不向所有的引入非结构化信息发布结构消息，只是那些可以引起一条或多条结构维度数据的改变的信息发布消息。同样的，一个 OLTP 系统接受了一个结构维度信息的话，只能是那种可以改变其维度表的那种信息。

维度管理者的适配器是 EAI 中间件的组件，适配器在实时维度管理系统中捕获各种非标准化数据的消息，并把他们放入维度管理者的队列中，监听发布事件，并被维度管理者所出发。随后，这些被转换成为 XML 格式，并发布到 broker。维度管理适配器隔绝了其他的 EAI 结构和任何 awareness 或者实时维度管理系统本身。

根据图 11.10，实际的实时规则维度，是下列子部分的典型模块。

- 清洗：清洗部分读取引入的不规则数据，从工作流中抛弃那些残缺的或非法的维度实例。他保证了查询可以被展示出来，这种查询有可能是跨越不同数据源的。还可以保证包含在属性里的数据值的合法性。
- 规范化：规范化部分从工作流中获取清洗后的数据，执行域对域转换。这种转换是原始系统经过验征后的值转换到一个规范化致域，是一系列的企业级数据根据各自维度属性所作的变换。在某种情况下，规范化的值可以到达目的地，通过查找表映射所有的源系统值到一致值。另一方面，规范化值的到达是概然说，使用模糊统计计算获得起始关系。符合概然说标准的特殊工具通常是被用来清洗整理街道地址的。把地址整理成为已知的格式；更正街道的拼写，城市和州的名称；更正邮政编码。其它一些模糊工具还可以更正名字的错误拼写；标准化常用名称。
- 匹配：匹配部分从工作流中获取清洗一致化后的数据，识别并排除冗余数据。专门的软件工具被用来帮助执行匹配功能，使用明确论和概然论的匹配技术。关于这些工具工作的详细说明超出了本章的描述标准，本文在这里只能说，这些工具是开发者定义很多匹配规则和匹配运算。每个通过和新产生的一个联合记录的匹配，匹配引擎记录这些匹配的相似性，这是全面模糊性匹配的一种平衡。最后的记录是比较由维度管理者定义的高（明确匹配）低（明确不匹配）两种极端，还包括定义的匹配组键值。在高低端值之间的纪录用特殊的方法来标记，特别是回顾指南。匹配的定义和高低端值的设定是一个半科学半艺术的工作。他很大程度上依赖于企业对于错误的容忍性，低延迟的需要，政策和调整需要考虑的事项，和回顾指南人员构成。

  不足匹配（保持一致但是客户记录与分离的维度管理不恒等）大体上被认为比过渡匹配更保守而且较低插入，并且被认为是规范的。在实时 EAI 环境中消除一个错误匹配不是那么容易完成的事情，而且常常意味着客户处理已经被 OLTP 系统错误整理过了，是在 OLTP 系统对维度管理者手工分离和中继转发的融合请求做出反应时作出的错误处理。

  在实时环境中，需要手工维护的纪录对于不匹配状态是一种典型的缺省状态，而且手工维护可以延迟执行。由于性能的原因，当处理大维度时例如零售业的客户维度，一定要限定候选键,使其达到合理的实时性能。抽取候选键可以大幅度加快匹配处理，但是有时会在传递用来匹配的候选纪录时出错。因此，实时在线匹配有时会危及安全，同时，整个维度的周期性重新匹配也许是必需的。这种单片集成电路的重新匹配处理能够创造大量的一致维度信息和操作性系统的更新，这些都是你需要小心处理的。特殊元数据是匹配部分中用来描述匹配变化逻辑，匹配极限，和匹配过载信息。过载匹配是指那些永远不需要匹配或永远要匹配的纪录。我们经常需要开发一个专门的用户接口来支持手工匹配处理和手工维护的元数据，也包括变化逻辑，匹配极端和候选键处理。
- 共存：一旦纪录被认为是做匹配，维度的最好反映一定是从匹配纪录蒸馏而来的用

来建立完整和正确的组合映像。一组记录一旦被定义成另一匹配，维度的最优值必须从创建完整和精确合成值的匹配记录中萃取出来。这种萃取处理经常被作为共存被提及，这是因为他保证了最好的维度属性源在全部维度规范化处理中的存在。存在处理利用业务逻辑，在属性对属性基础上，识别源系统优先权被应用在存在规范化维度映像。这种共存处理利用确定的业务规则，基于属性对属性，源系统在已存在提供的统一的维度值中排列优先顺序。非空源属性值存在于最终的完整的维度纪录中，这些记录是理想元数据基于层级源系统规则所捕获。存在部分也应该支持属性的定义组，这些定义组是当同源纪录作为避免摘取奇怪数据的阻碍时存在的，例如，地址行 1 和地址行 2 是来自于不同源纪录的。存在模块也能够处理明显的时间点代理键的产生。

所以当一个客户纪录变化了 10 次之后一定会有 10 个不同的代理键值。然而每一个都应含有相同的全面客户代理键。这两种键值的处理方法都是必须的，。因为，维度管理是为两大部分服务的：数据集市，数据集市必须使用时间点代理键；OLTP 系统，它只需要维度纪录的同代映射。

- 发布：一旦维度映像充分集成（清洗，一致化，匹配和存在），你必须要决定作为结果存在的数据是否为全新或与先前的维度映像是否不同，从而授权发布给集成网络。大部分维度管理者会选择性发布来防止无限发布循环。如果发布是经过授权的，发布部分会唤起维度管理者的适配器，该适配器一直处于连续监听发布请求的状态，所以他可以收集所有的或一部分的维度纪录，并把它转换成一致的 XML 维度消息。将消息发送到 EAI broker 中在企业内部分配。唤起维度管理者适配器通常会占用表空间，表空间是被应用来在特殊存储中一致化数据或触法监听适配器时更新一条活多条记录的。

  设计发布规则要保证足够的维度同步化，它有可能贯穿在整个企业中。同时还要避免无穷回应或空转条件，避免钻取到详细发布应用和规则，这些可以被 EAI broker 轻易处理。

当 ETL 构架师设计实时维度管理者时一样要小心处理分析业务需求而且有时要勇于处理关于组织政策的阻力。很多寻求真实设想版本的经理人， 而不是其他人。技术上，架构师必须判定什么时候什么地点完成全部或部分维度纪录，这些记录已经在应用间一消息的方式被传递。这种情况应该会引起一致化数据的发布。如何处理可能的争论和空驶条件，如何最好自动平衡应用需求，是否使用较少磁盘的直接贯穿方式还是微批处理方式？这些我们将在下一部分讨论。

听起来很复杂？是的，是这么复杂。但是实时维度经理对于企业需要在整个企业范围同步丰富客户信息或其他关键维度信息来说，是一个真实的强有力的办法。给那些早期采用者提供了竞争优势。

小批处理过程

当设计实时数据集市分区或维度系统时，你经常会面对一个普遍困难选择：解决方案应该包含直接处理或利用频繁的微批处理？目前的工具都是支持低延迟数据环境的，比如说 CTF，常常缺乏已经存在的 ETL 工具的转换能力。一批 ETL 工具提供商已经开始提供他们工具的实时版本了，以更加事务性的方式处理信息，但有时功能会受到限制。实时维度管理系统的设计者，当选择明确或概然论匹配工具时，也常常会面对类似的困境：一些工具专门已批处理方式操作。你如何使批处理方式工具在实时环境中性能最佳？

对于那些面向批处理工具的接近实时传递性能，一个合理的折衷方法是设计一套方案来

处理频度发生的微批处理，使用标准转变工作来控制元数据结构。

考虑图 11.11。在工作表中的每一件工作代表了任意一个或非标准维度处理纪录，对任意一个实时数据集市或维度管理者。这些工作被几个处理所贯穿，每个都在发布之前定义了处理表，例如清洗和一致化。在表 11.11 中的数据模型里，微批处理表代表了小批量的既定处理，每个微批处理都完成一定的工作。每个微批处理所完成的工作会被工作处理事件表所捕获，也会捕获工作事件状态属性成功或失败标志。

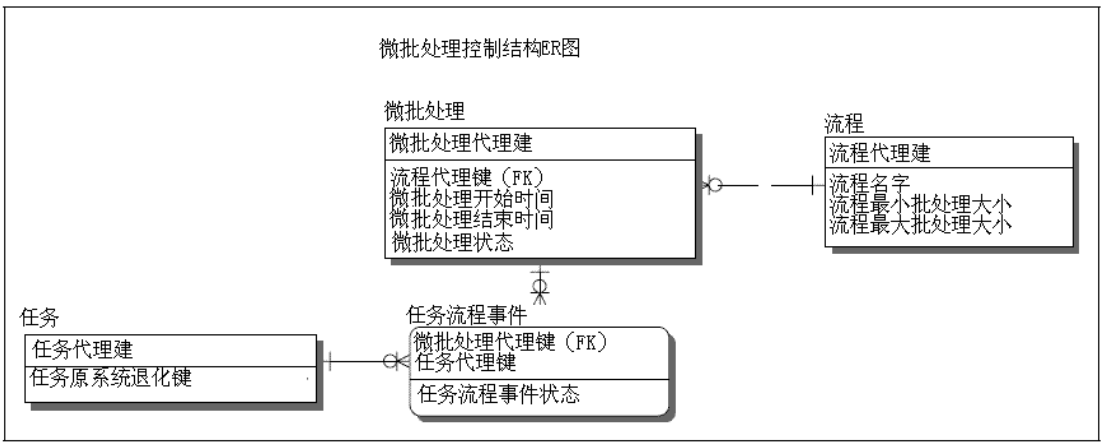

图 11.11　小批处理数据模型

处理连续执行，就像邮件收发的后台程序，寻找那些适当处理状态的工具。当他们发现可接受的最低数量工作就会调用微批处理来处理工作，生成工作处理事件纪录及其适当状态值；然后他们继续寻找更多的工作来处理

建立模块处理后台程序的好处，与使用直接贯穿处理相反，好处在于他们可以独立发展，有独立可调的批处理规格，可以作为具有新特性的新工具被替换和独立加固升级。同样的，新处理可以轻易的被加入到工作流中，而且需要选择性处理的工作可以轻易的实现调节。例如专门的地址确认或信用价值评分。但是这种弹性消耗成本。微批处理流要求每个处理详细定义和表述接口，特别是数据库表的来源和去向。这种附加的 I/O 和强加的复杂性的影响是很明显的。在实际当中，可以通过设计个别的工作表来降低复杂度，工作表被用来作为所有处理的共同源头和目的。可以数据库内存中缓存来减少 I/O 压力。不过，微批量的方法和直接处理的执行不一样。控制性纪录和全部工作将被频繁分开以保证他们尽可能的小。一个典型微批处理的每个处理分支都是非常简单的。当每个批处理被调用时，将生成一条微批处理纪录，同时生成一定数量的工作事件处理纪录（处理表中最低数量和最高数量的详细批处理数量）。这种基本技巧给管理大量并发批处理提供足够信息，也保证了所有批量处理和工作处理有足够的审计信息。表 11.12 说明了如何工作。

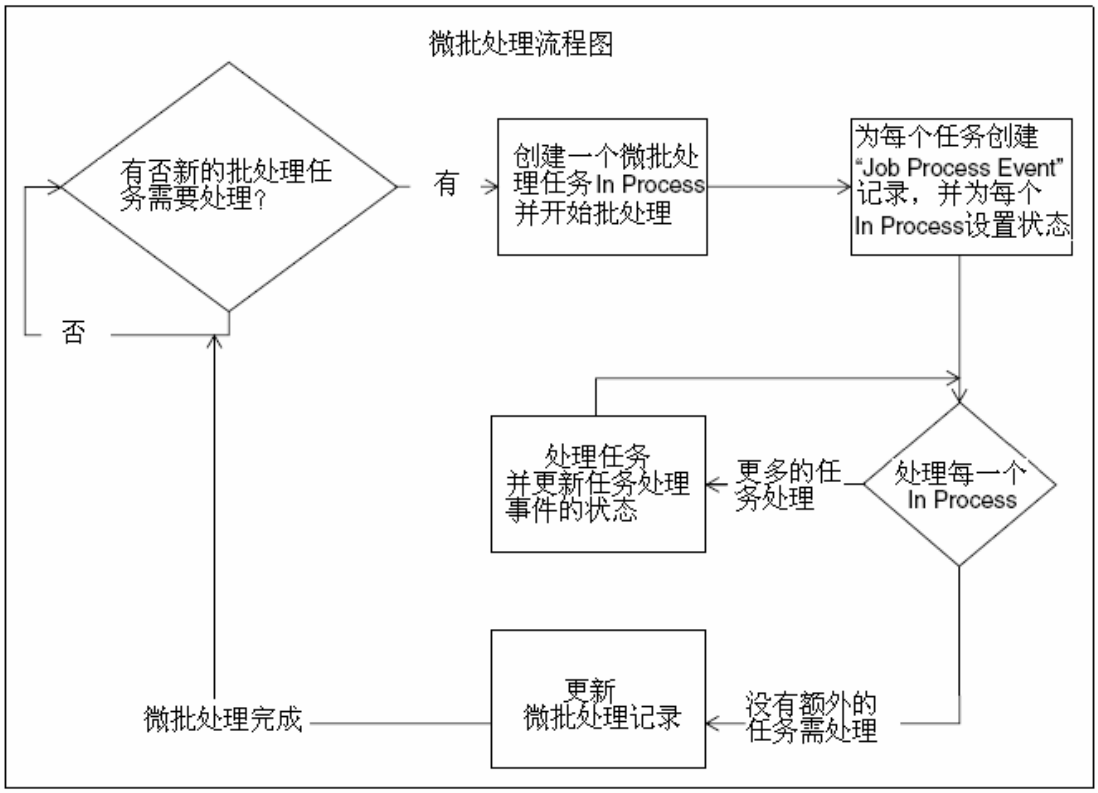

图 11.12　小批处理流图

表 11.12 描述了一个单一处理。各个处理连续运行，而且像邮件收发后台程序一样和其他处理共同完成同一工作，因此设置工作处理事件和微批处理状态值，然后继续。所以，一个维度管理者的数据清洗，一致化，匹配，存在和发布处理后台处理有时会在不同的工作任务上同时发生。一个数据集市的实时 CTF 系统可能会存在转换过程和代理键查找后台程序，等等。每个后台程序连续的查找任务进行处理。我们这篇文章中所讲的任务工作是指在给定处理中以最优方式处理的阶段，已经被处理过的任务纪录。

正如在图 11.12 中所看到的，任务处理事件的状态在处理中被设置，而且一个关系数据库处理的起始点是确定的。假如每个任务都处理过了（清洗，一致化等等），任务处理事件状态被更新为成功或失败。作为更替降低处理成本，这种更新发生在批处理结束时。一旦所有的任务都执行过，批处理完成，批处理控制表被更新。假如成功执行没有致命错误，在处理中加固所有的处理事件纪录，然后执行提交，转换结果被写入到数据库中。如果有失败发生或一个无法接受的大数量任务处理事件状态为失败，将会执行回滚处理，数据库将会回复到微批处理执行之前的状态。

回滚事件并不一定回滚错误信息或控制表中的状态值。很多关系型数据库提供了处理控制选项来支持这种约束。

应用于实时维度管理者的微批处理 ETL 在图 11.13 中表述。作为系列后台处理，从数据库表中读取，更新控制，分段运输，以及一致化的资料库。

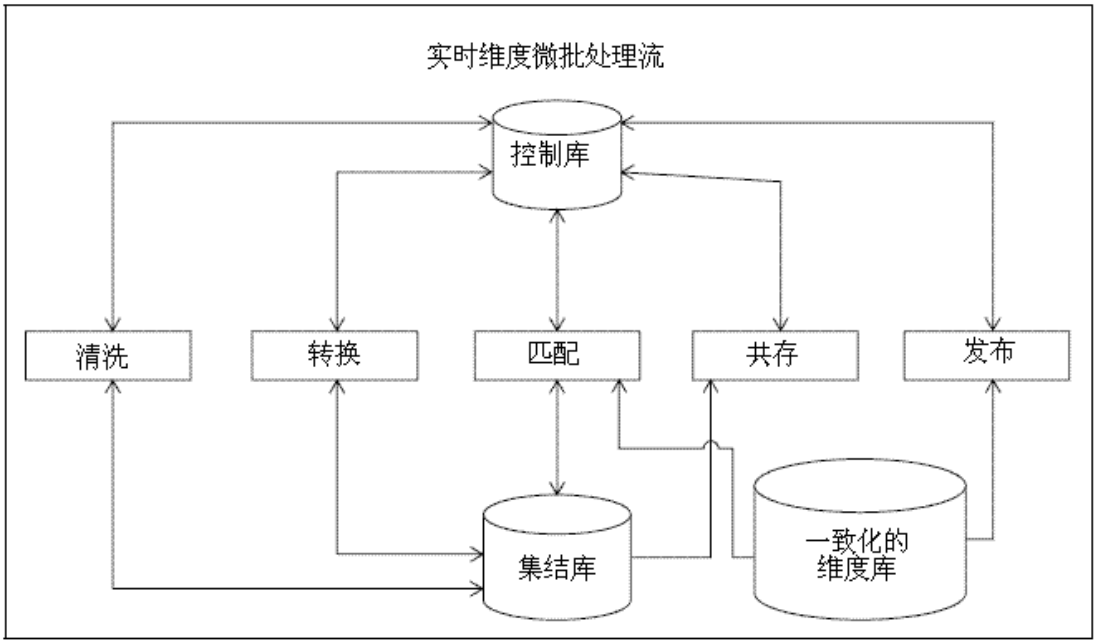

图 11.13　实时维度管理器的小批处理流

每个处理在任务事件表中更新状态值，在分段传输阶段修改和新建数据，或一致化维度。

- 清洗过程从分段传输库中读取未经验证的数据，并只向控制库中写入状态值。
- 在规范化过程中，从分段传输读取清洗后的没有一致化的数据，并向分段传输写入一致值来保证一致属性。
- 匹配阶段从分段传输读取一致化后的未经匹配的数据，向分段传输写入匹配键值。
- 存在阶段从分段传输读取经过匹配的未存在的纪录，并向一致化维度库中插入或更新纪录。
- 发布阶段从一致化维度库中读取经过一致化的纪录，唤起维度管理者适配器，然后适配器将纪录发布到集成网络中。

一个微批处理 ETL 系统也可以被用来转换 CTF 到实时数据仓库，这时的数据转换处理会比那些只被 CTF 工具支持的数据转换要复杂得多。CTF 可以被用来实现操作系统的近实时数据抽取和轻度转换服务，在分段传输阶段删除数据。从分段传输阶段，微批处理开始处理复杂转换和逐步插入到到实时数据集市分区表中。同时支持实时数据上报和正规的夜间批处理使数据装载到统计数据集市表中，在处理过程中清空实时分区表，为第二天的转换作准备。

选择一个方法－－－一个决定向导

目前，实时数据仓库的整个区域还是十分混乱的。有非常多的方法和技术可以选择，被各种厂商和评论家所包围，但是在实际应用中没有多少成功的案例可供分析学习。所以选择一个适当的构架和方法是一个非常令人畏缩的任务。

下列的表格试图分析我们在这一章节讨论过的方法，形成帮助我们缩小选择范围的指导方针。表 11.1 是实时报告方法的对比矩阵。

表 11.1　实时报告决策向导矩阵

对比矩阵		EII ONLY	EII+ STATIC DW	ETL	CTF	CTF-MB-ETL	MB-ETL	EAI
		实时数据仓库里的企业综合信息	和传统的非实时数据仓库一致的企业综合信息	标准 ETL 过程	捕获，转换，传输到实时数据仓库	捕获，转换，ETL 微批处理传输到实时数据仓库	ETL 微批处理传输到实时数据仓库	企业应用传输到综合实时数据仓库
支持历史数据		√	√	√	√	√	√	
复杂报告综合数据	弱	√	√	√	√	√	√	√
	适中	√	√	√	√	√	√	√
	强	√	√	√		√	√	√
数据更新最大间隔	1 分钟	√	√					√
	15 分钟	√	√		√			√
	1 小时	√	√		√	√	√	√
	1 天	√	√	√	√	√	√	√

表 11.2 提供了实时维度管理系统的对比矩阵，他们都需要实时应用集成，可以通过批数据集成得到。

表 11.2　维度验证决策向导矩阵

对比矩阵		ETL	CTF	CTF-MB-ETL	MB-ETL	EAI	EAI-MB-ETL
		标准 ETL 过程	捕获，转换，传输到维度管理系统	捕获，转换，ETL 微批处理传输到维度管理系统	ETL 微批处理传输到维度管理系统	企业综合应用传输到维度管理系统	企业综合应用和 ETL 微批处理传输到维度管理系统
复杂报告综合数据	弱	√	√	√	√	√	√
	适中	√	√	√	√	√	√
	强	√		√	√	√	√
数据更新最大间隔	1 分钟	√				√	
	15 分钟		√			√	√
	1 小时		√	√	√	√	√
	1 天	√	√	√	√	√	√
企业综合	仅数据集市	√	√	√	√	√	√
	“虚”数据综合		√	√		√	√
	“实”数据综合		√			√	√
	应用综合					√	√

小结

实时 ETL 并不仅仅是一种潮流或是新特征。向实时数据传输迁移对于 ETL 管道的各个方面来说都是挑战，这种挑战是关于物理上的和逻辑上的。也许实时系统的最好设备是用流动 ETL 替代面向批操作的 ETL。在这一章中，我们介绍了实时 ETL 实践方法的艺术，并且

指出了我们可能会面临的风险。