实时数据仓库调研报告

北京大学计算机系数据库实验室

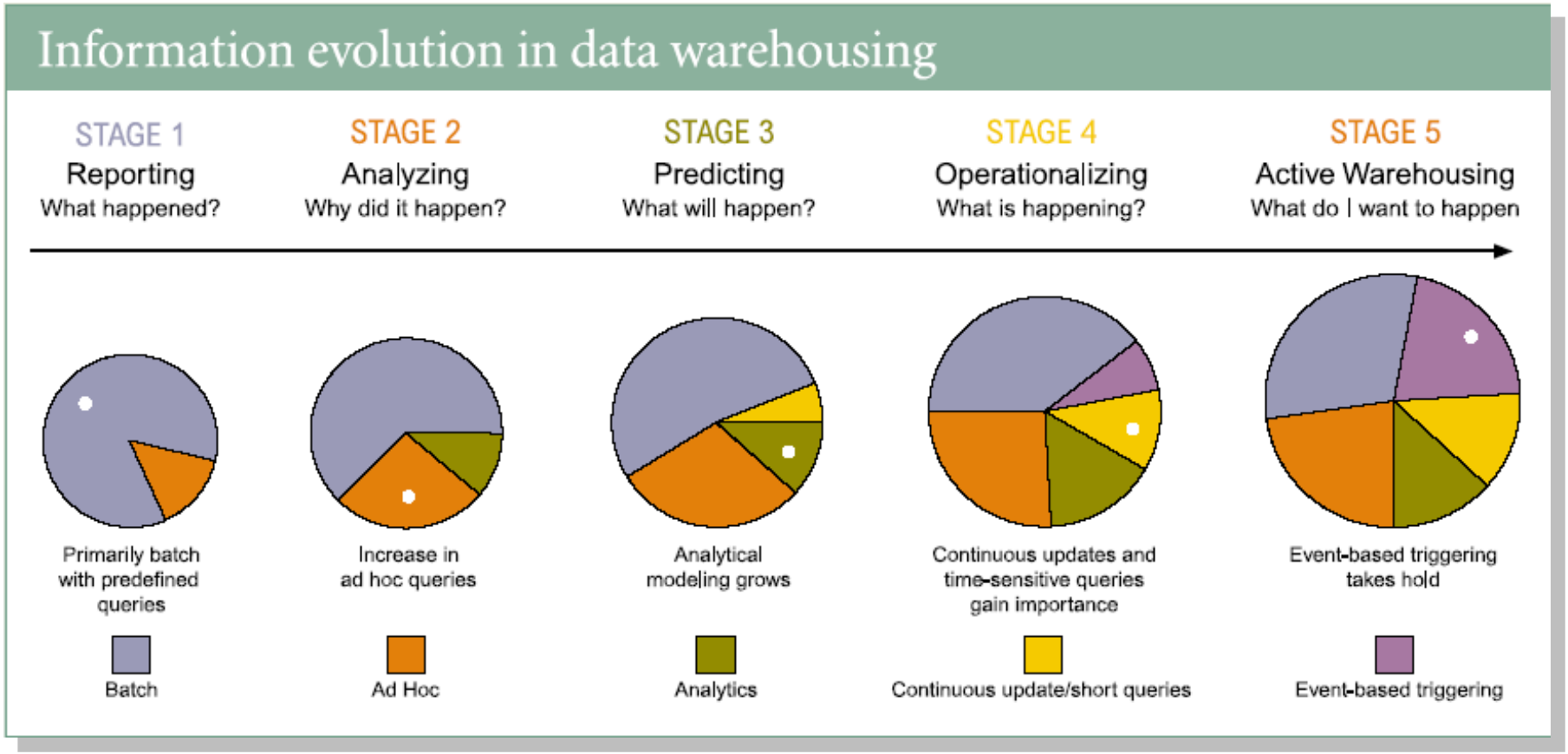
林子雨

提纲

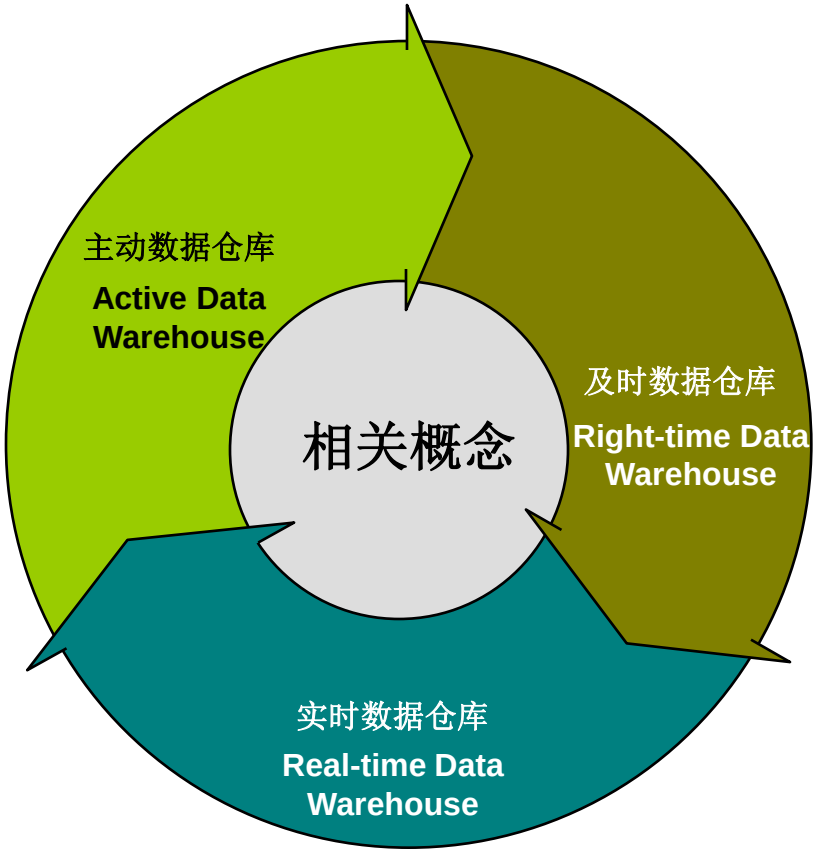
- 与实时数据仓库相关的概念
- 实时数据仓库面临的挑战
- 连续数据集成
- 总结
- 参考文献

数据仓库的5个发展阶段

1. 报表阶段 Reporting
2. 分析阶段 Analysis
3. 预测阶段 Prediction
4. 实时决策阶段 Operationalize
5. 主动决策阶段 Activate



与实时数据仓库相关的概念



与实时数据仓库相关的概念

实时数据仓库

- Michael Haisten首先提出实时数据仓库的概念
- 在数据仓库中保持两类数据，静态数据和动态数据
  - 静态数据：满足用户的查询分析要求
  - 动态数据：为了实时性，可以实时更新，并做相应转换，满足用户对“最后一分钟”数据的实时请求
- *其他定义……[11]*

总结：实时数据仓库是这样一个系统，只要行为发生，数据就变得可用，就能从中获得信息。

数据仓库相关概念区分

及时数据仓库

- 数据更新周期介于“实时和每天一次”之间[9]
- 在特定的商务问题提出时，就能马上给出答案
- 从及时数据仓库中得到的答案，能够帮助组织做出带来巨大收益的决策
- 为了回答这些事先设计的特定的商务问题，需要在数据仓库中预先存储该商务问题所需的集成的数据(比如一天一次或15分钟一次)
- *从技术角度讲，不存在实时数据仓库*

与实时数据仓库相关的概念

主动数据仓库

- 主动数据仓库是一个关系型数据仓库环境，支持：[8]
  - 数据的实时更新
  - 快速的响应时间
  - 基于钻取的聚集数据查询能力
  - 动态的交互能力

与实时数据仓库相关的概念

- 主动数据仓库(Active Data Warehouse)
- 及时数据仓库(Right-time Data Warehouse)
- 实时数据仓库(Real-time Data Warehouse)

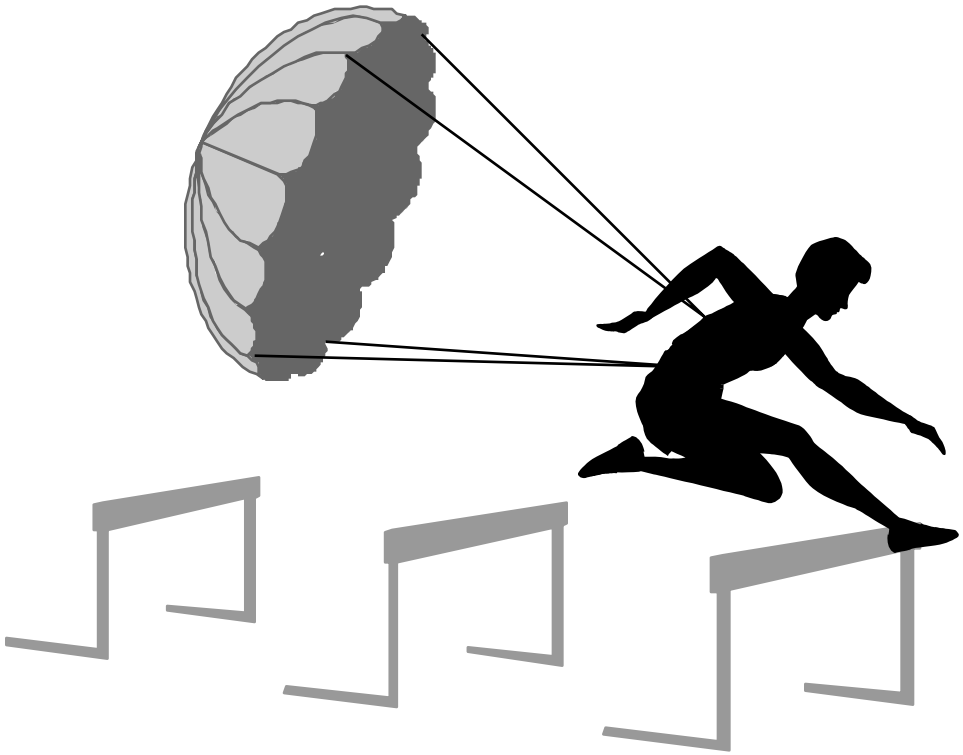
	主动数据仓库	及时数据仓库	实时数据仓库
更新方式	实时	及时	实时
自动规则触发	有	无	无[注]

注：某些厂商在其实时数据仓库解决方案中包含自动规则触发功能，但仍采用“实时数据仓库”的名称，而实际上已经等同于“主动数据仓库”。

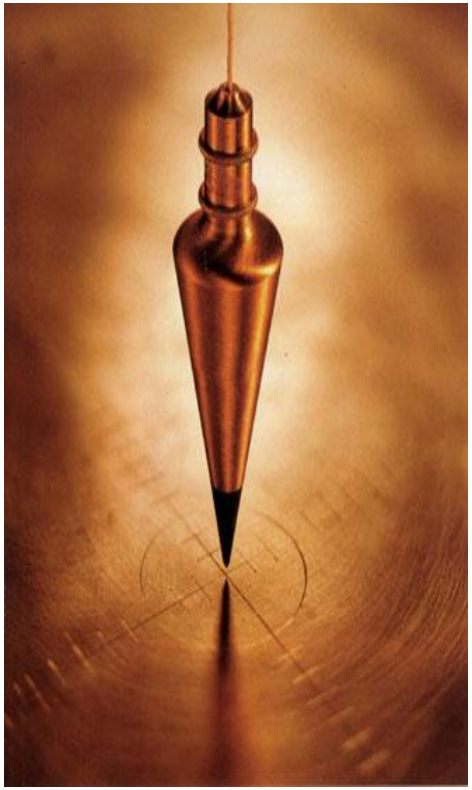
提纲

- 与实时数据仓库相关的概念
- 实时数据仓库面临的挑战
- 连续数据集成
- 总结
- 参考文献

实时数据仓库面临的挑战和解决方案



实时数据仓库面临的挑战和解决方案



- 挑战1 –支持实时ETL
- 挑战2 –实时数据的建模
- 挑战3 –OLAP查询和变化的数据
- 挑战4 –可扩展性和查询竞争
- 挑战5 –实时报警

挑战1：支持实时ETL

批处理ETL

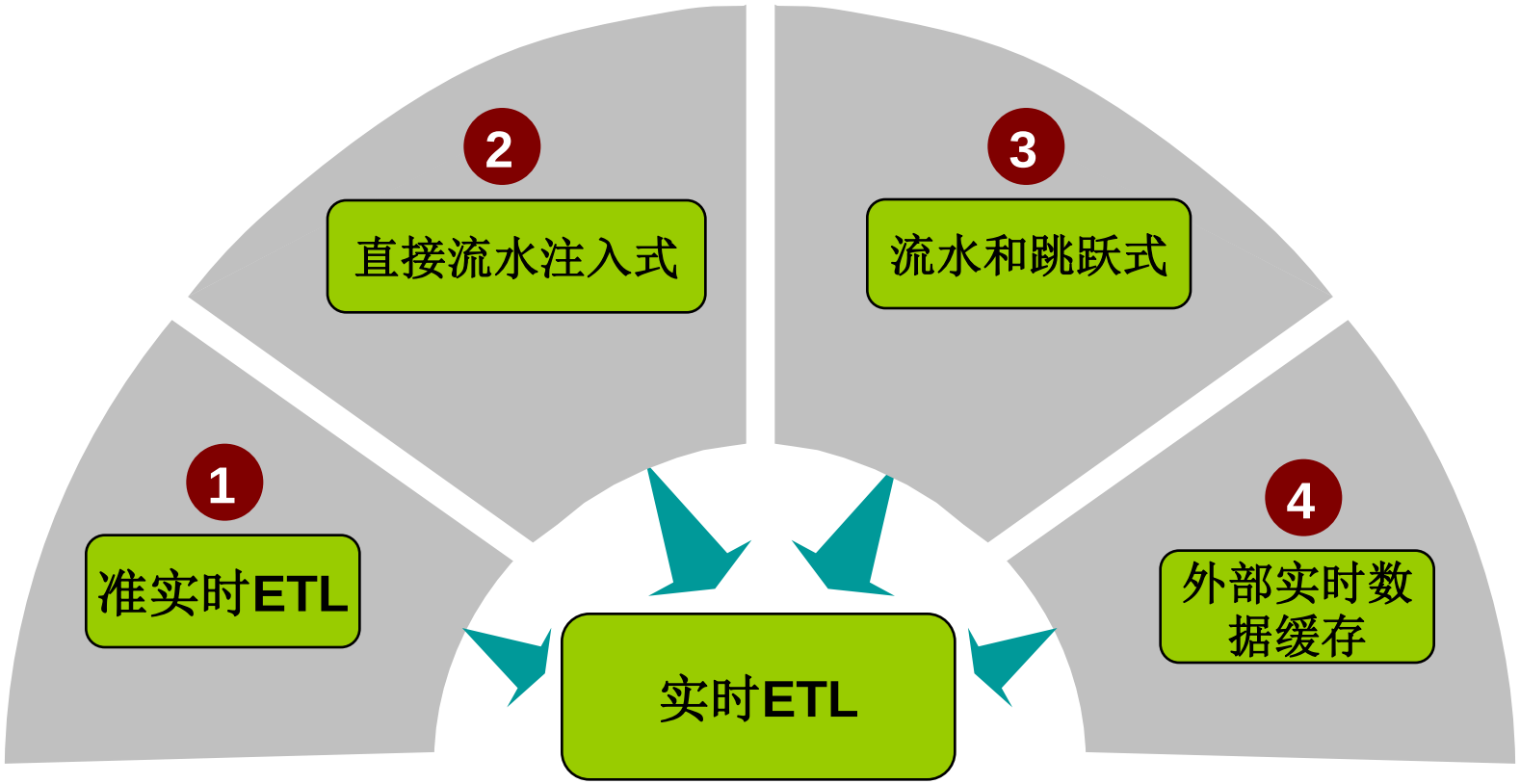
- 几乎所有的ETL工具和系统，不管是现成的产品还是定制编码的，都是以批处理方式工作
- ETL过程通常需要数据仓库暂时当机，停止对外服务

问题描述

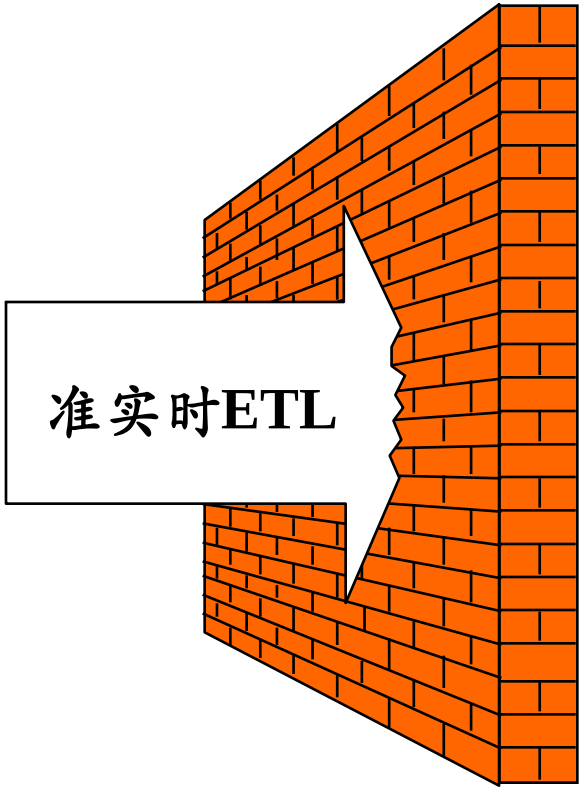
实时ETL

- 不可能允许存在系统的当机时间
- 没有当机的情况下对数据仓库进行连续更新，通常与传统的ETL工具和系统的设计理念是相互冲突的

挑战1：支持实时ETL



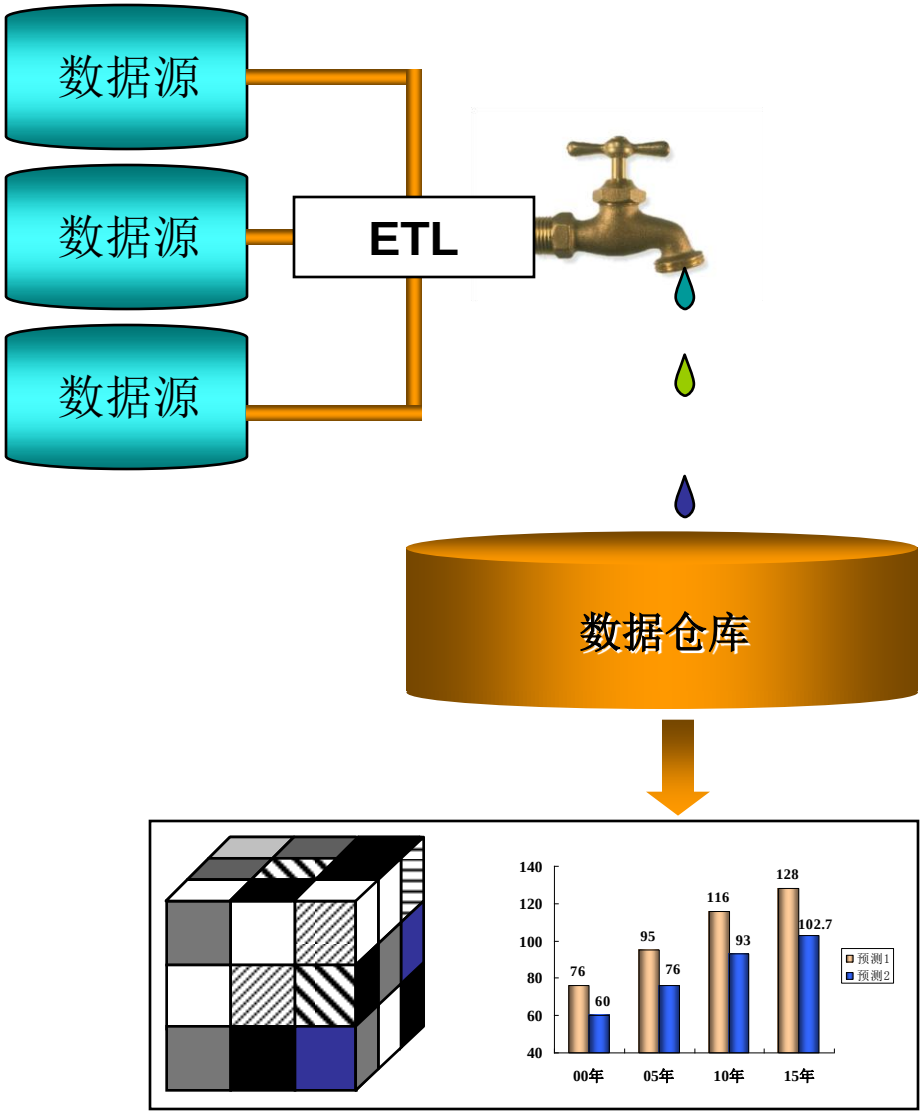
挑战1：支持实时ETL



解决方案1：准实时ETL

- 根本不考虑采用真正实时的ETL
- 并不是所有的问题都需要实时的答案
- 因实时而引起的开销可能超出由实时而带来的收益
- 对于某些应用，只要简单地提高现有的数据加载的频率即可
- 当不需要严格的实时时，准实时是一个比较可行的解决方案

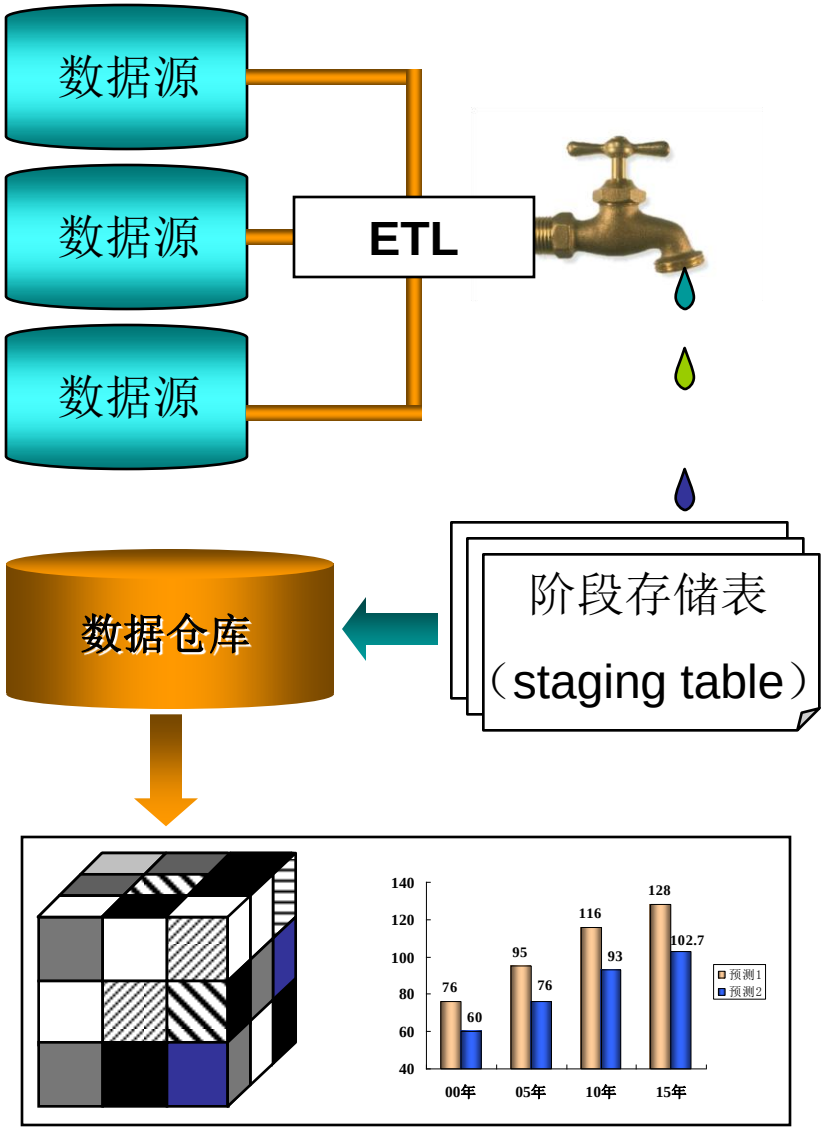
挑战1：支持实时ETL



解决方案2：直接流水注入式[15]

- 把从源系统产生的新数据象水流一样直接注入到数据仓库
- 可以直接在数据库仓库事实表中插入或更新数据
- 也可以把数据插入到实时分区当中的单独的事实表中
- --------缺点---------
- 可扩展性不好， 复杂查询和连续插入及更新混在一起进行会严重影响数据库的性能

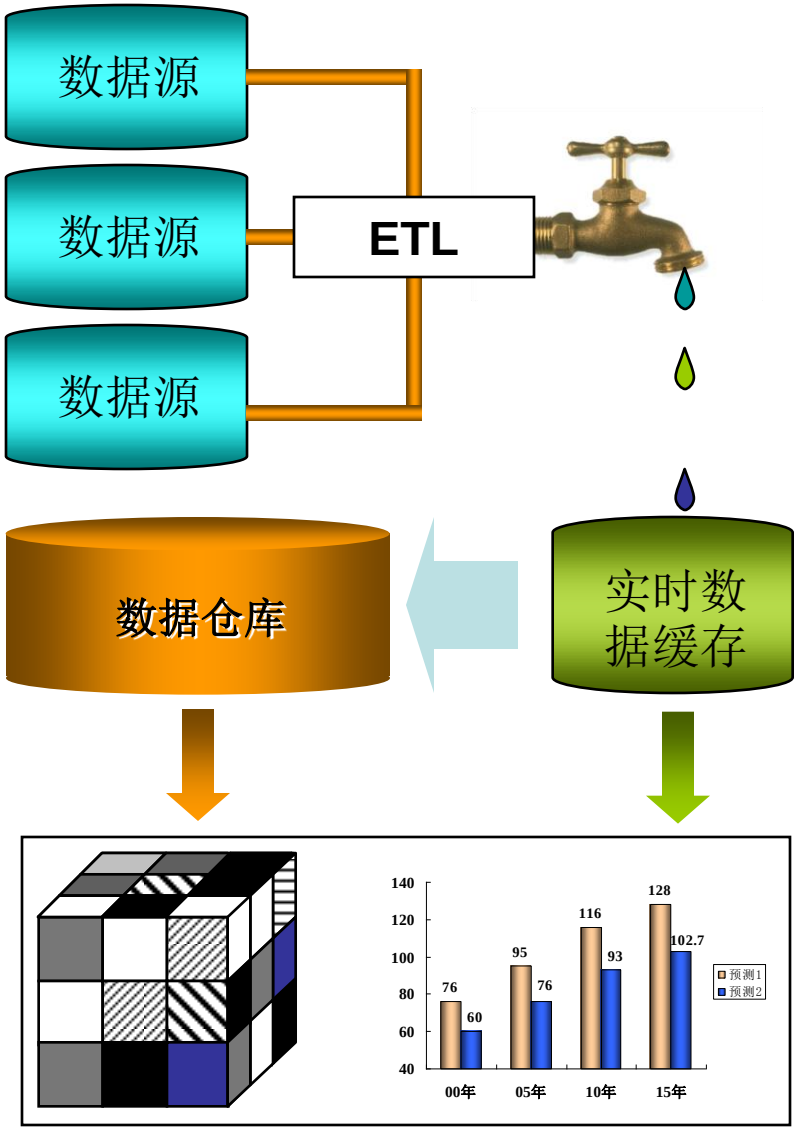
挑战1：支持实时ETL



解决方案3：流水和跳跃式

- 把数据连续地注入到阶段存储表
- 阶段存储表的结构和数据仓库表的结构相同
- 阶段存储表中的内容会和事实表周期性地进行交换
- 采用“以视图实现集成的实时分区”这种方法，只需要修改视图的定义
- 在处理数据交换的时候，最好暂时停止OLAP服务
- 当更新周期为每分钟一次到每小时一次之间时，可以采用该方法

挑战1：支持实时ETL



解决方案4：外部实时缓存

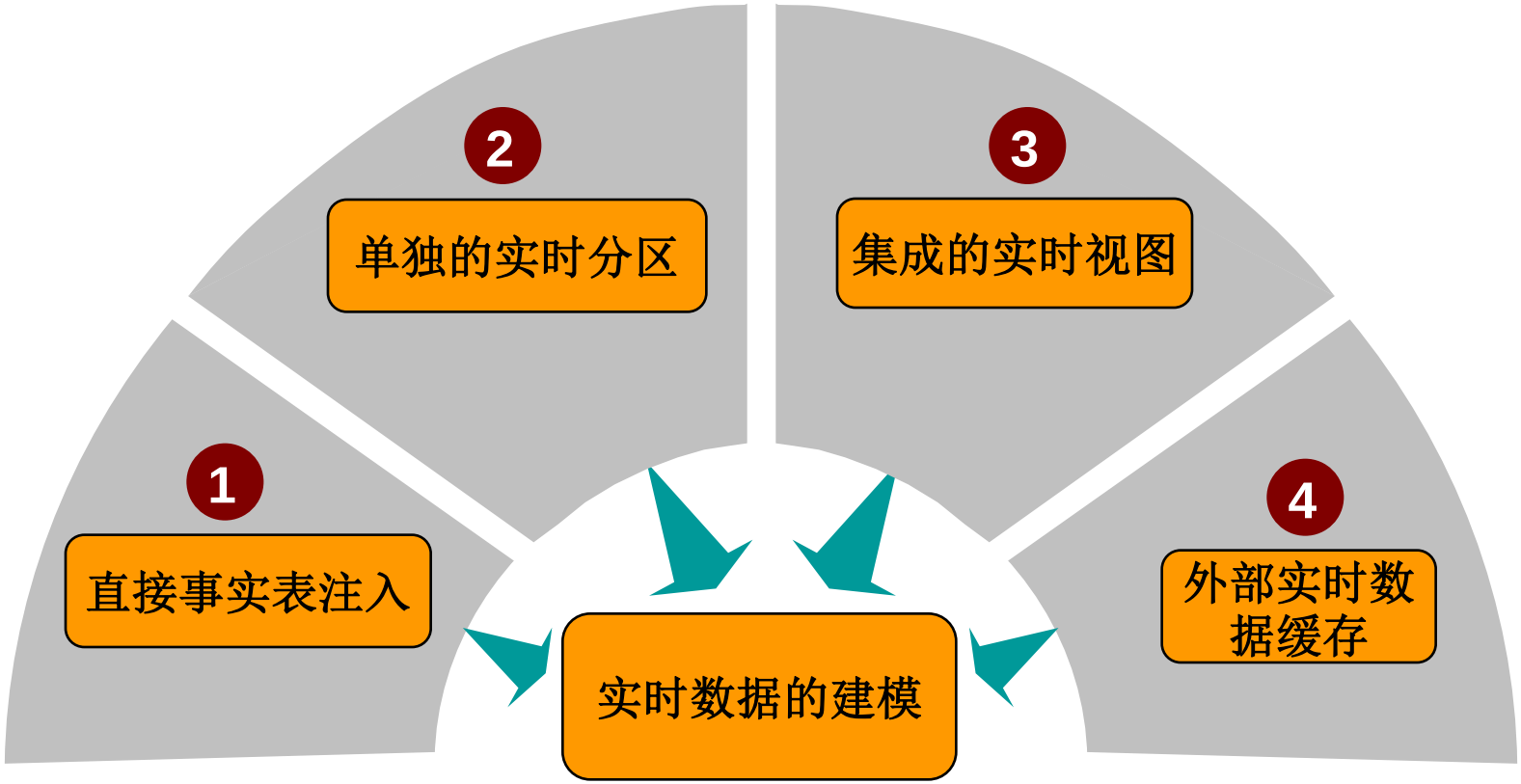
- 可以避免对数据仓库性能的影响
- 不用对现有的数据仓库做出修改
- 可以是另一个专用的数据库服务器
- 也可以是一个大的数据库系统的单独的实例
- 把所有那些需要实时数据的查询定向到实时数据缓存
- 或者把某个查询所需要的实时数据临时地无缝隙地整合到传统的数据仓库中
- ----------缺点----------
- 要安装和维护一个额外的单独的数据库

挑战2：实时数据的建模

- 把实时数据存储在哪里
- 如何把实时数据和数据模型的其余部分连接起来

问题描述

挑战2：实时数据的建模



挑战2：实时数据的建模

直接事实表注入

- 用“直接流水注入式”或者“流水和跳跃式”方式实现实时ETL，不需特殊的数据建模方法
- 从查询工具的角度而言，以这种方式建模得到的实时数据仓库和非实时数据仓库是没有什么区别的
- 需要考虑的最主要的事情就是缓存

单独的实时分区

- 把实时数据存储在一个单独的数据仓库事实表中
- 许多支持实时分区的查询工具，就可以自动从实时分区取得其所需的实时数据
- 从查询工具配置和管理的角度而言，这种数据建模方法实现起来是最复杂的

集成的实时视图

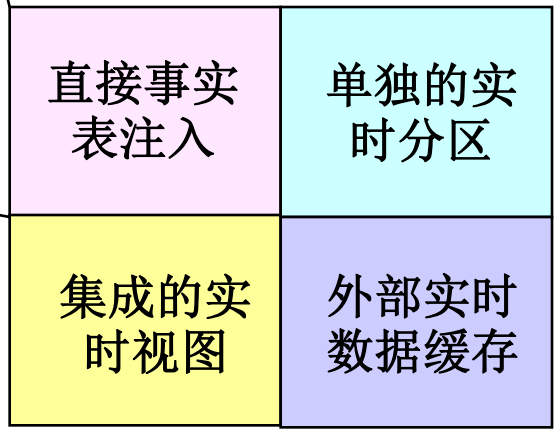
- 把实时数据和历史数据存储在不同的表中，但是采用相同的表结构
- 通过使用数据库视图，把历史数据和实时数据表结合在一起
- 查询工具和终端用户不需要进行两个表的连接操作

外部实时数据缓存

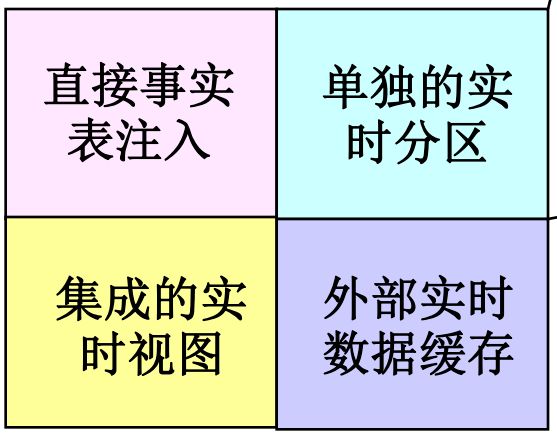
- 不需要特殊的数据建模
- RTDC数据库使用和数据仓库相同的数据模型，但只包含需要实时数据的表
- 当查询和分析需要历史数据和实时数据实现无缝集成时，外部实时缓存是很有用的
- 消除性能问题

挑战2：实时数据的建模

- 用“直接流水注入式”或者“流水和跳跃式”方式实现实时ETL，不需特殊的数据建模方法
- 从查询工具的角度而言，以这种方式建模得到的实时数据仓库和非实时数据仓库是没有什么区别的
- 需要考虑的最主要的事情就是缓存

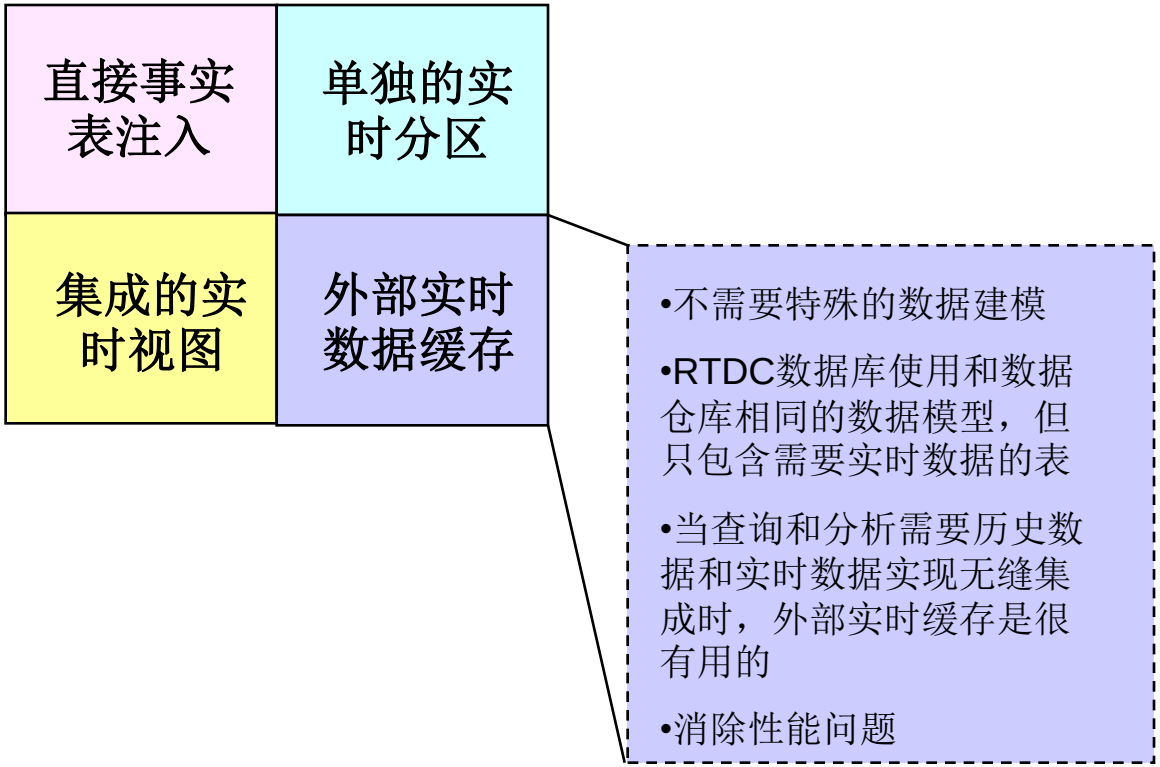


挑战2：实时数据的建模

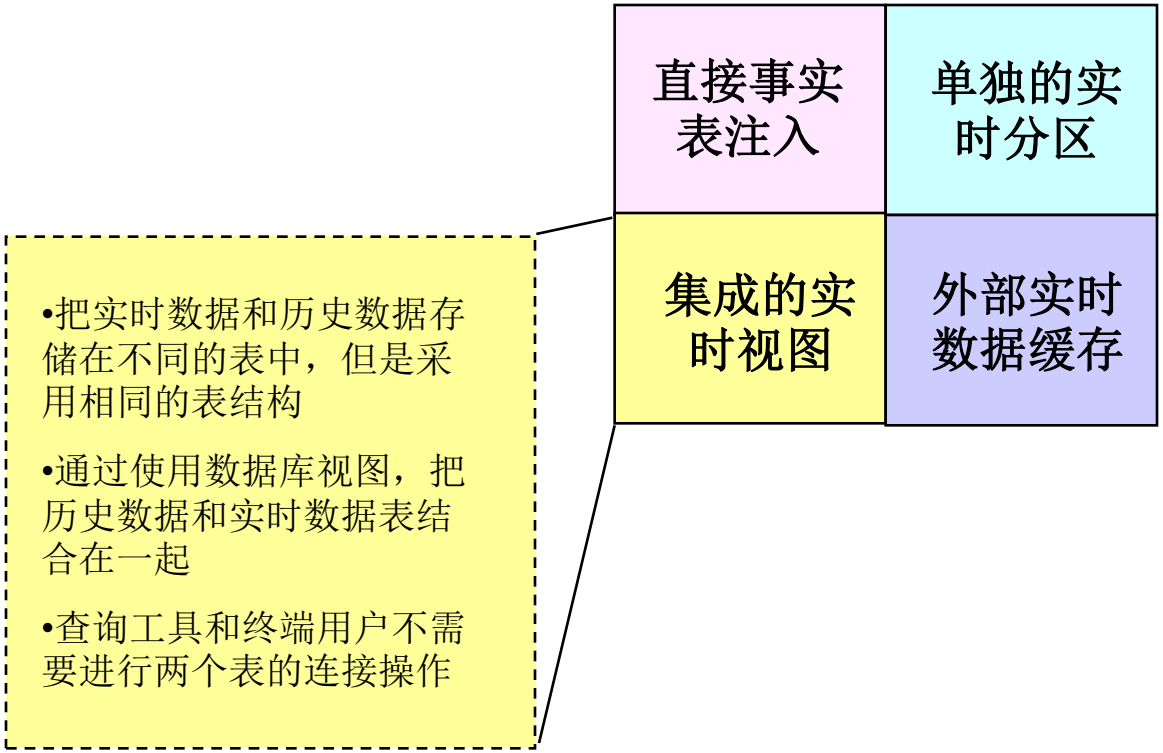


- 把实时数据存储在一个单独的数据仓库事实表中[13]
- 许多支持实时分区的查询工具，就可以自动从实时分区取得其所需的实时数据[12]
- 从查询工具配置和管理的角度而言，这种数据建模方法实现起来是最复杂的

挑战2：实时数据的建模


直接事实表注入
单独的实时分区
集成的实时视图
外部实时数据缓存
•不需要特殊的数据建模
•RTDC数据库使用和数据仓库相同的数据模型，但只包含需要实时数据的表
•当查询和分析需要历史数据和实时数据实现无缝集成时，外部实时缓存是很有用的
•消除性能问题

挑战2：实时数据的建模


直接事实表注入
单独的实时分区
集成的实时视图
外部实时数据缓存
•把实时数据和历史数据存储在不同的表中，但是采用相同的表结构
•通过使用数据库视图，把历史数据和实时数据表结合在一起
•查询工具和终端用户不需要进行两个表的连接操作

挑战3： OLAP查询和变化的数据

问题描述

– OLAP和查询工具被设计成在静态的、不发生变化的历史数据上进行操作

– OLAP和查询工具在设计时不会考虑到如何保证他们的结果不会被同时发生的数据更新所影响

– 关系型OLAP工具对这种问题就特别敏感，因为他们使用多段SQL（multi-pass SQL）来执行最简单的数据分析操作

挑战3： OLAP查询和变化的数据

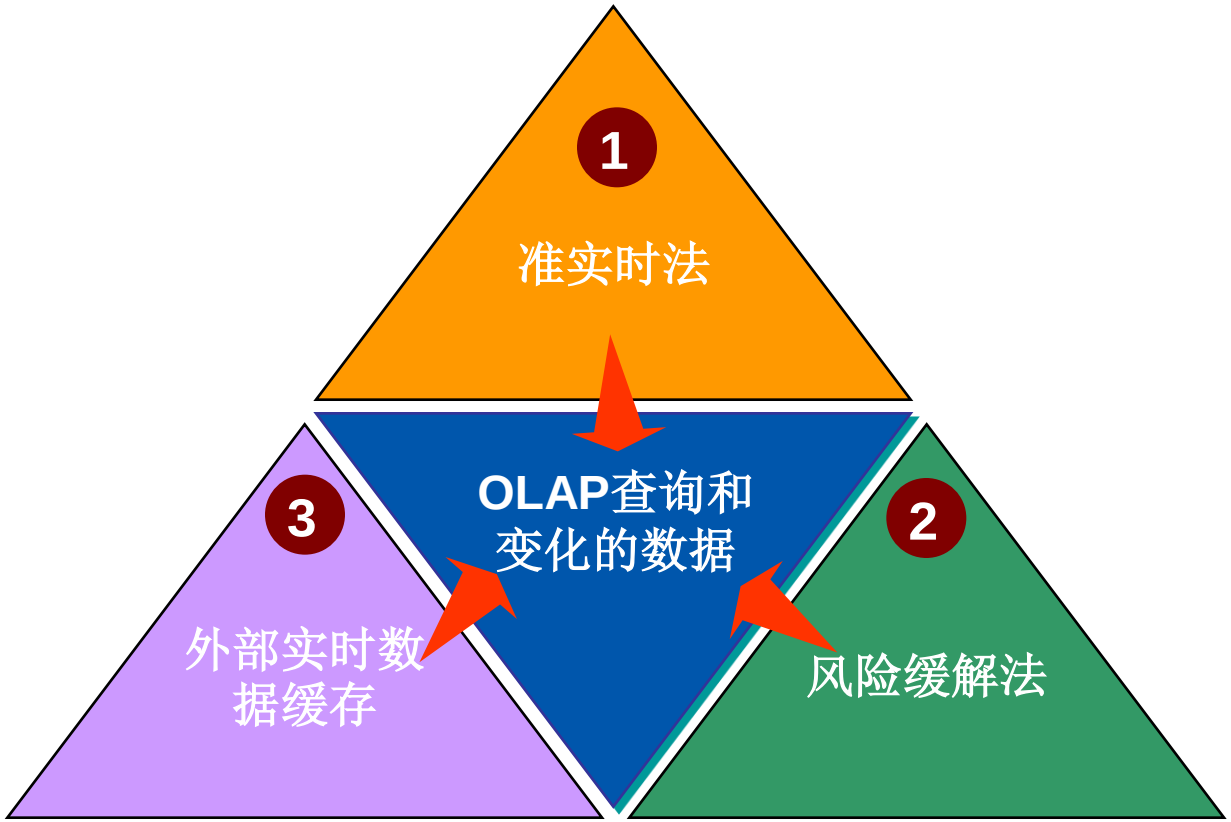
```
0:00  create table TEMP1(
        Category_Id LONG, DOLLARSALES DOUBLE)
0:01  insert into TEMP1
        select all.[Category_Id] AS Category_Id,
          sum (all.[Tot_Dollar_Sales]) AS DOLLARSALES
         from [YR_CATEGORY_SLS] all
         group by  all.[Category_Id]
0:05  create table TEMP2 (ALLPRODUCTSD DOUBLE)
0:06  insert into TEMP2
        select sum((all.[Tot_Dollar_Sales]) AS
        ALLPRODUCTSD
        from [YR_CATEGORY_SLS] all
```

```
0:08  select distinct pa1.[Category_Id] AS Category_Id,
          all.[Category_Desc] AS Category_Desc,
          all.[DOLLARSALES] AS DOLLARSALES,

(pa1.[DOLLARSALES]/pa2.[ALLPRODUCTSD])
        AS DOLLARSALESC
        from   [TEMP1] pa1,
               [TEMP2] pa2,
               [LU_CATEGORY] all
        where
pa1.[Category_Id]=all.[Category_Id]
0:09  drop table TEMP1
0:10  drop table TEMP2
```

表 一个多段查询的例子

挑战3： OLAP查询和变化的数据



挑战3： OLAP查询和变化的数据

只有当数据变化得太快，以至于在多段查询开始执行时的数据和执行结束时的数据不一致，才会出现报表的不一致性问题

如果OLAP服务器被设计成不会在数据仓库加载或跳越时发起新的查询，那么使用准实时的ETL方法，或者采用具备相对较长周期的流水跳跃式，可以使该问题得到缓解

解决方案1：准实时方法

– 这种方法的缺点是，数据并不是真正实时的，而是有一定的延迟

– 还必须采取措施保证在数据仓库加载或跳跃期间，OLAP服务器可以被正确地停止

挑战3： OLAP查询和变化的数据

解决方案2：
风险缓解法

- 最简单的方法是禁止用户对实时数据执行非常复杂的查询

- 另一个替代方法是，在单独的分区维护一份实时数据的快照，为复杂查询提供服务，快照的更新不必太频繁。但是，这又增加了安装、维护和用户培训的复杂性

挑战3： OLAP查询和变化的数据

把实时数据和历史数据分开存储，报表就不会出现内部不一致性问题

采用即时数据合并处理，仍然可以实现历史数据和实时数据的无缝集成，为报表提供服务

解决方案3：
外部实时数据缓存

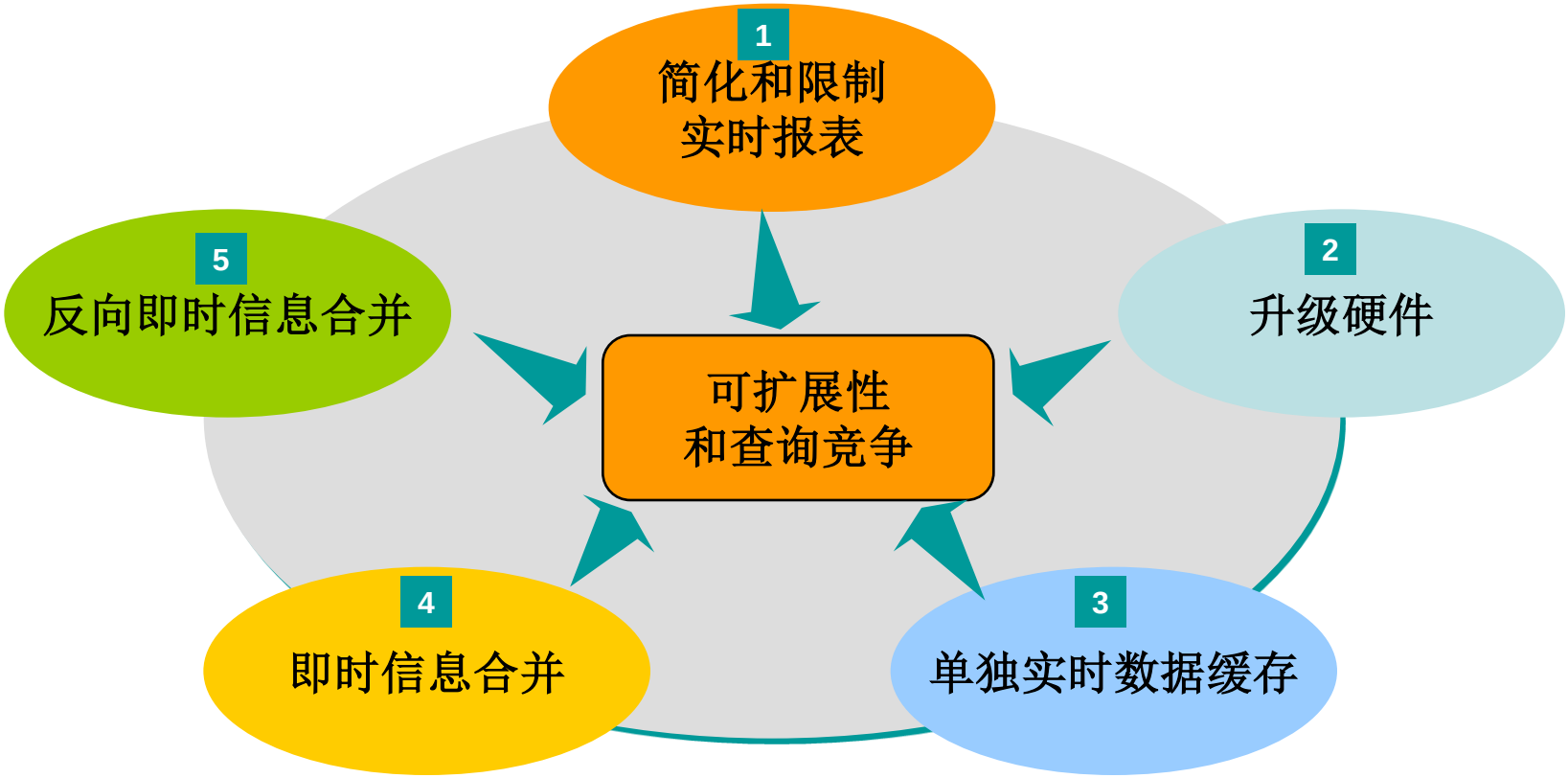
采用反向的即时数据合并（把历史数据合并到基于内存的实时数据缓存中），就可以解决报表的延迟问题

挑战4：可扩展性和查询竞争

问题描述

1. 实施实时数据仓库解决方案所面临的重要的问题就是查询竞争和可扩展性问题
2. 数据仓库和交易系统是分离的
3. 通常数据仓库可扩展性是“查询涉及的数据量和并发用户数”的直接函数
4. 在实时系统中，连续数据更新和加载也会进一步增加系统资源的消耗
5. 连续数据插入和复杂SELECT操作的冲突将会严重限制系统的可扩展性

挑战4：可扩展性和查询竞争



挑战4：可扩展性和查询竞争

解决方案

1. 简化和限制实时报表
2. 升级硬件
3. 单独实时数据缓存
4. 即时信息合并
5. 反向即时信息合并

- 需要实时报表的用户可能只发出很简单的查询要求，如果可以把报表限制在很简单的且快速的单段SQL查询，那么，许多关系数据库本身就可以处理这种冲突
- 数据仓库中最复杂的查询都会在较长的时间跨度内访问数据。如果使这些查询只基于不发生变化的静态的历史数据，那么就可以消除和实时数据的冲突
- 到底有哪些用户需要访问实时数据
- 可以通过通知机制来很好地满足用户对实时数据要求

挑战4：可扩展性和查询竞争

解决方案

1. 简化和限制实时报表
2. 升级硬件
3. 单独实时数据缓存
4. 即时信息合并
5. 反向即时信息合并

- 可以为高端的SMP数据库系统增加更多的节点
- 或者可以为数据仓库配备更快的处理器和更大的内存

缺点

- 可以解决短期的可扩展性问题，但并非解决问题的良策

挑战4：可扩展性和查询竞争

解决方案

1. 简化和限制实时报表
2. 升级硬件
3. 单独实时数据缓存
4. 即时信息合并
5. 反向即时信息合并

- 把所有实时数据加载和查询活动重定向到一个独立的专门设计用来存储实时数据的数据库
- 把所有的实时活动都放在一个单独的实时数据缓存上执行，就不会给已有的数据仓库增加任何开销

缺点

- 把实时数据的存储和数据仓库分离开来，报表工具就无法把实时数据和历史数据有效结合起来
- 如果在实时数据缓存上运行很复杂的报表，那么，就可能出现前面所论述的报表的内部不一致性、数据冲突和可扩展性等问题

挑战4：可扩展性和查询竞争

解决方案

1. 简化和限制实时报表
2. 升级硬件
3. 单独实时数据缓存
4. 即时信息合并
5. 反向即时信息合并

特殊应用场景

- 需要真正的实时数据（而不是准实时数据）
- 包含快速变化的数据（每秒10-1000个交易）
- 需要满足10个、100个甚至1000个并发用户的访问
- 结合历史数据和实时数据进行分析
- 复杂的分析型OLAP查询

即时信息合并

- 实时数据被保存在外部的实时数据缓存中，历史数据保存在数据仓库中，这两种数据会根据需要被整合到一起呈现给应用。可通过JIM(just-in-time information merging)机制实现

挑战4：可扩展性和查询竞争

解决方案

1. 简化和限制实时报表
2. 升级硬件
3. 单独实时数据缓存
4. 即时信息合并
5. 反向即时信息合并

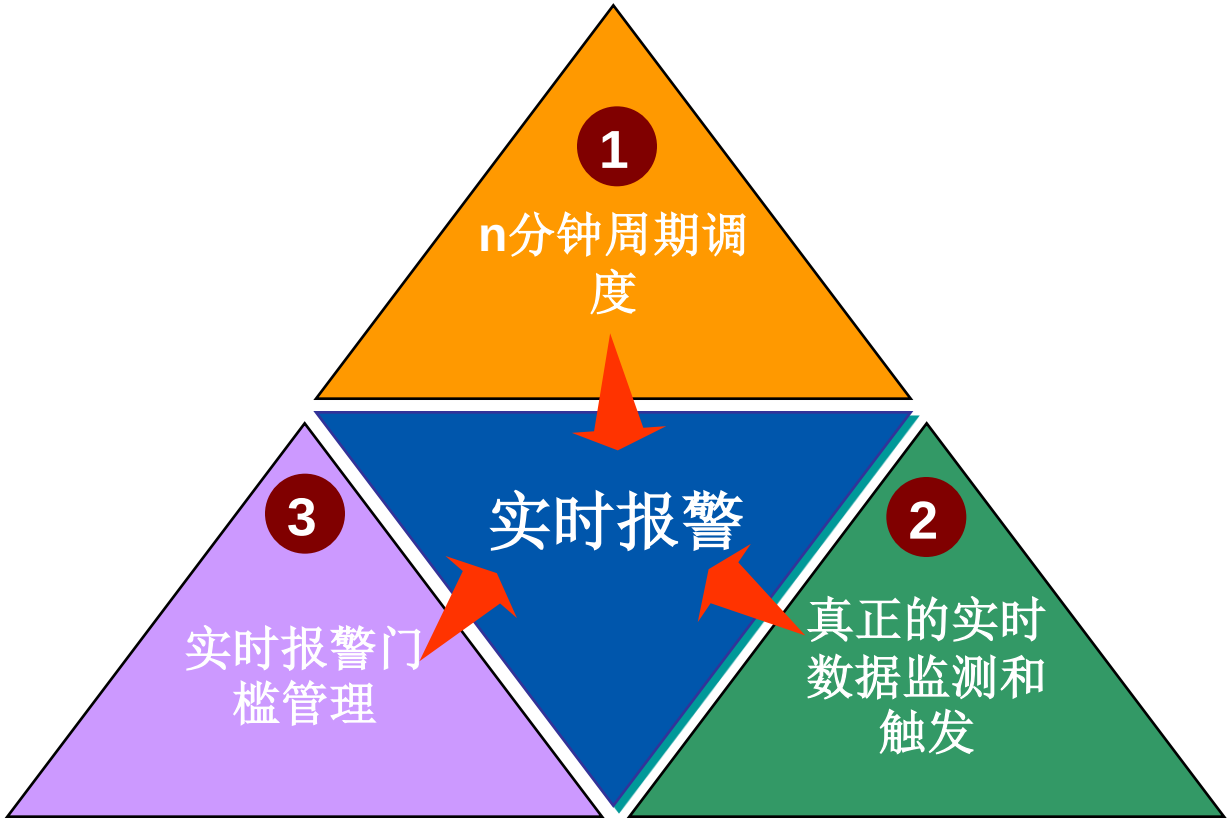
- 对于那些主要使用实时数据而很少使用历史数据的查询而言，RJIM非常有用
- 所需要的数据需要从数据仓库临时地反向加载到实时数据缓存，查询就在实时数据缓存上运行
- 这种机制只有当实时数据缓存所在的RDBMS完全支持SQL时才可以实现，对于那些没有支持许多SQL函数的内存数据库来说，无法实现这种机制
- 对于一个智能的RJIM来说，它会在数据仓库上做尽量多的分析，从而得到所需要的最合适的具备很高粒度的综合数据，然后再加载到实时数据缓存中，这样可以减少数据传递量

挑战5：实时报警

问题描述

- 许多和数据仓库相关的报警应用，主要是在晚上完成数据仓库的数据加载以后，采用以电子邮件的形式通知用户
- 实时数据的可用性，使得数据仓库报警应用更加吸引用户
- 现有产品以周期方式或以事件触发方式进行运作
- 必须有机制保证不会重复触发报警

挑战5：实时报警



挑战5：实时报警

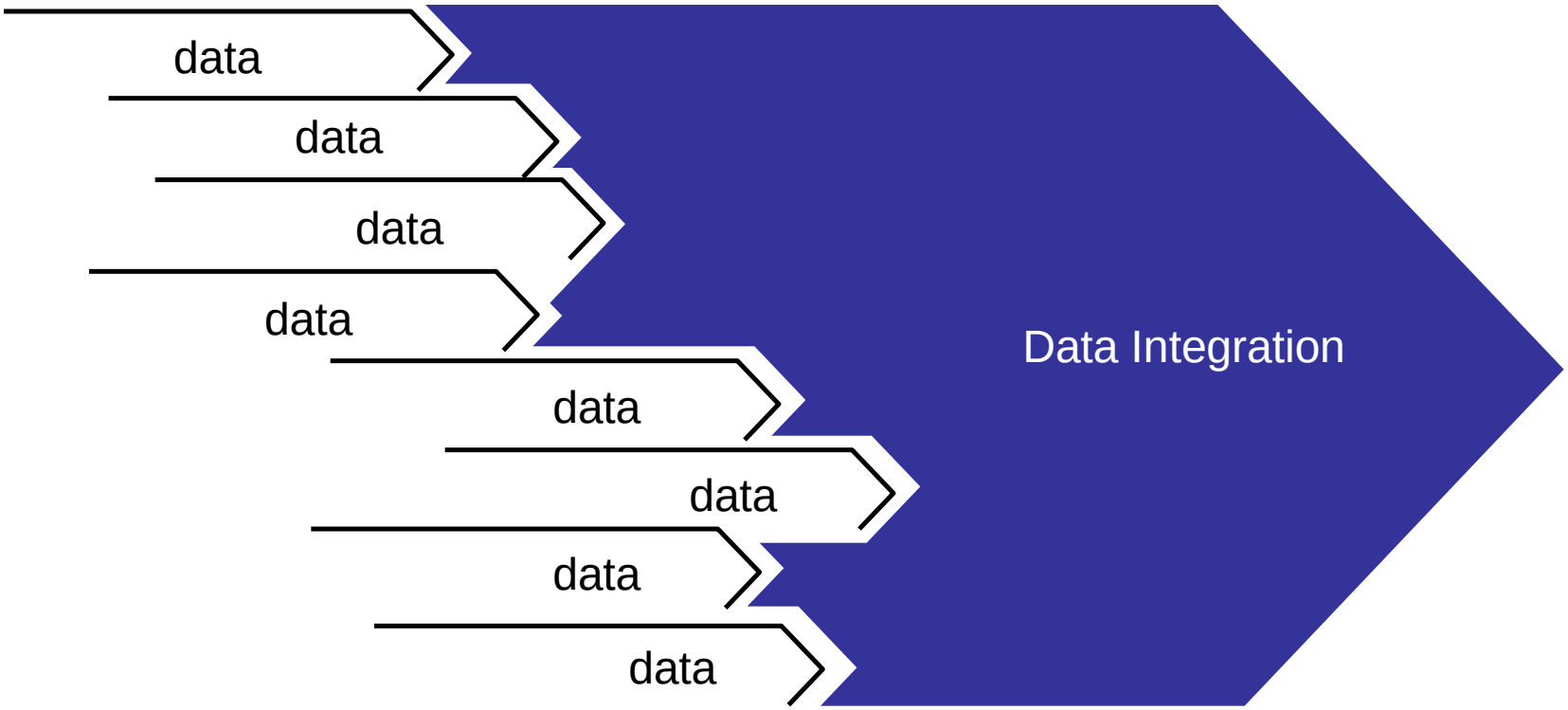
1. n分钟周期调度
   - 周期性地使用数据仓库报警工具包，实现近似实时报警
   - 以准实时方式加载数据的数据仓库，只需在数据更新结束以后触发警报
2. 真正的实时数据监测和触发
   - 需要引入触发系统
   - 系统相当复杂，复杂度取决于用户的数量、报警的条件和到达的数据量，需要大量的硬件资源，尤其是内存
3. 实时报警门槛管理
   - 采用静态的门槛值定义，对于那些每晚更新一次或每月更新一次的系统来说，不会发生频繁地重复报警问题，对于更新更频繁的系统则会发生

提纲

- 与实时数据仓库相关的概念
- 实时数据仓库面临的挑战
- 连续数据集成
- 总结
- 参考文献

数据集成方式

- 数据整合（Data Consolidation）
- 数据传播（Data Propagation）
- 数据联邦（Data Federation）
- 混合方式（A Hybrid Approach）



数据仓库中的数据集成

- 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**。
- 主动数据仓库则必须支持**实时**的查询处理，也就必须要求具备实时数据集成的能力，有好几种方法都是朝着这个方向努力：
  - 为了最小化更新窗口，Labio[2]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[3]，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。

不足：上述方法的一个不尽人意的方面是，数据仓库并不是真正的实时的。在需要真正实时的应用中，最好的方法是象水灌溉一样，数据从源系统源源不断地输入到数据仓库中。

数据集成技术

有许多种技术可以为数据仓库提供数据获取服务，但是，只有部分技术能提供实时(连续)的数据集成。选择技术的标准应该着重参考以下几个方面的因素：数据质量、频率、可接受的延迟、数据集成、转换需求和处理开销。

属性	脚本	ETL	EAI	CDC
数据量*	中等	很高	低	高
频率	间歇性	间歇性	连续性	连续性
延迟	中等到高	中等到高	低	低
数据一致性	无	无	保证	保证
转换	中度	高级	基本	基本
处理开销	间歇性/高	间歇性/高	连续性/中等	连续性/低
*数据量和技术之外的很多其他因素相关，比如数据源、网络带宽和数据库变化捕捉机制				

数据集成技术

脚 本

- 使用灵活且比较经济
- 很容易着手开发和进行修改
- 几乎任何操作系统和绝大部分DBMSs都可以使用脚本
- 耗费开发者的时间和精力
- 不好管理和操作以及不能满足服务水平协议（SLA：Service-Level Agreements）

数据集成技术

ETL

- 实现大规模数据初步加载的理想解决方案
- 提供了高级的转换能力
- 通常都是在“维护时间窗口”进行
- 在ETL任务执行期间，数据源默认不会发生变化

数据集成技术

EAI

- 和ETL解决方案并存，并增强了ETL的功能
- 在源系统和目标系统之间进行连续的数据分发
- 提供高级的工作流支持和基本的数据转换
- 受到数据量的限制

数据集成技术

CDC

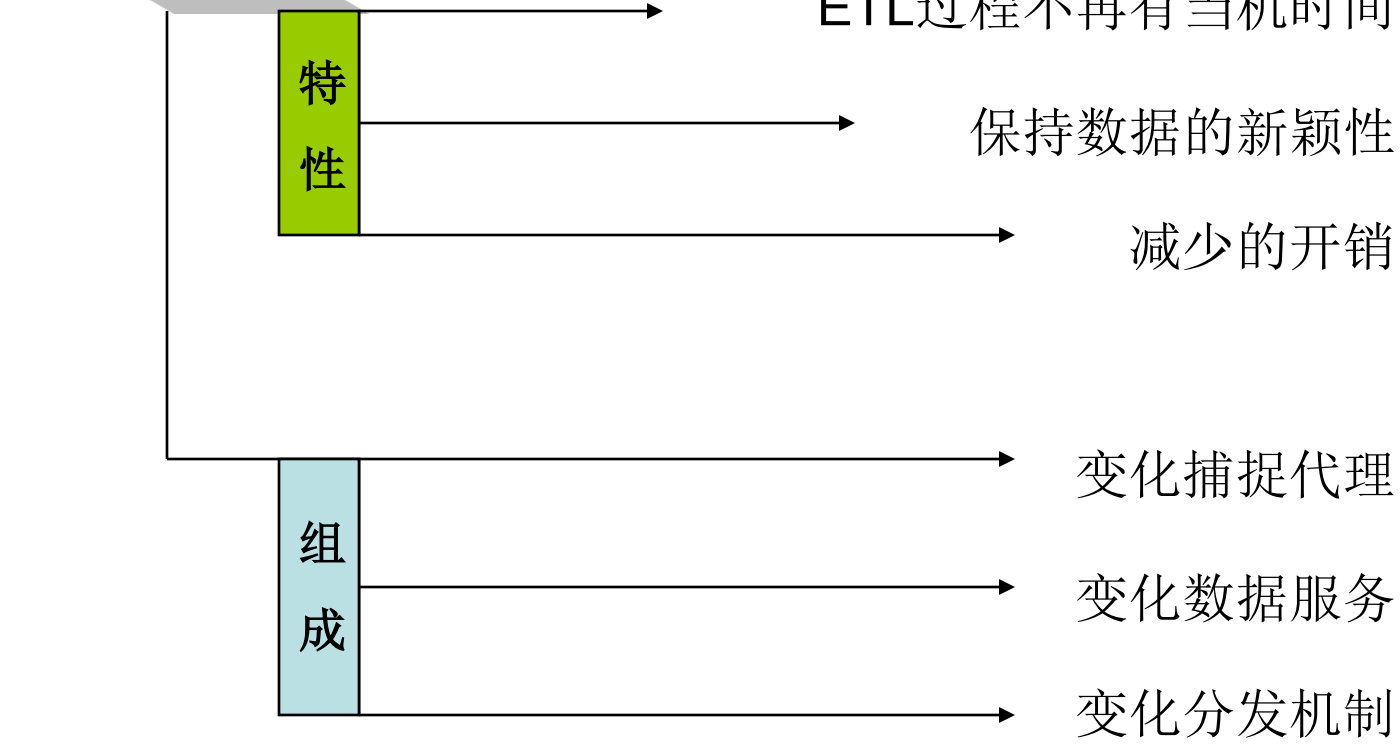
- 提供了连续变化数据的捕捉和分发能力
- 从OLTP系统中捕获变化的数据，进行基本的转换后把数据发送到数据仓库中
- 在体系结构上，CDC属于异步的，但它表现出类似同步的行为
- 维护数据事务的一致性

总　结

- EAI和CDC都只移动变化的数据和更新，而不是整个数据集，从而极大地减少了数据移动量
- EAI和CDC都不需要假设数据源的状态不发生改变，因为他们自己可以维护数据的一致性
- ETL适合作为数据仓库数据初步加载时的解决方案，而EAI和CDC则更适合作为此后的连续数据加载解决方案

变化数据捕捉技术

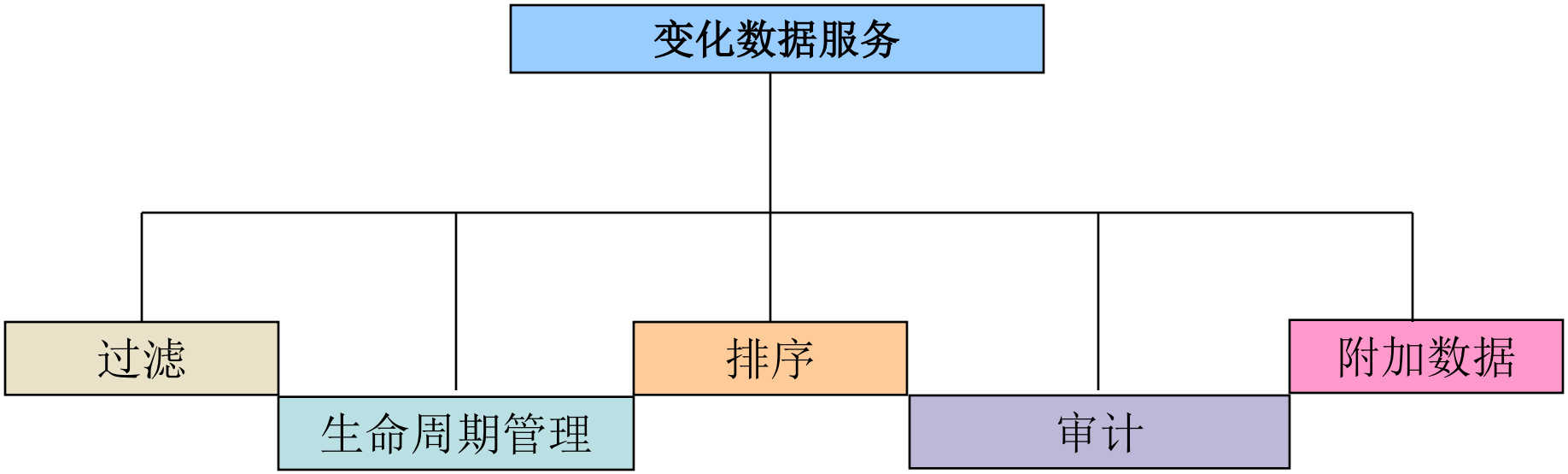
CDC 是一种变化数据捕捉技术，可以实现实时高效的数据集成[18]



变化捕捉代理

- 变化捕捉代理是一个软件组件
- 负责确定和捕捉发生在运营系统中的数据变化
- 可以对变化捕捉代理进行专门优化，使它适用于特定的源系统

变化数据服务



变化分发机制

- 负责把变化分发到消费者（通常是ETL程序）
- 可以支持一个或多个消费者，并且提供了灵活的数据分发方式
- Pull方式需要消费者周期性地发送请求，通常采用标准接口实现
- Push方式需要一直监听和等待变化的发生，一旦捕捉到变化，就立刻转移变化的数据，通常采用消息中间件来实现
- 提供动态返回的能力，从而满足重复处理和恢复处理等任务

数据分发方式

数据分发方式

推vs.拉	周期vs.非周期	一对一vs.一对多	数据分发选择
拉	非周期	一对一	request/response
		一对多	request/response with snooping
	周期	一对一	polling
		一对多	polling with snooping
推	非周期	一对一	Message queue
		一对多	Publish/subscribe
	周期	一对一	email sending
		一对多	email list digest

在传统的数据仓库实施方案中，大都采用pull机制。不管是周期性还是非周期性的pull，都会对运营系统造成额外负担，当我们需要近实时性地数据集成时，这种负担更加严重。对于运营系统来说，push机制是比较理想的，系统自己可以控制什么时候把数据推向什么地方。

应用场景1：面向批处理的pull CDC

场景描述

- ETL工具周期性地请求变化，每次都接收批量数据
- 变化分发请求可以采取不同的频度
- 提供变化数据的一种比较好的方式是以数据表的记录的形式表示
- CDC需要维护上次变化分发的位置和分发新的变化

总结：这种应用场景和传统的ETL很相似，不同的是，pull CDC只需要转移变化的数据，并不需要转移所有的数据，这就极大地减少了资源消耗，也消除了传统ETL过程的当机时间。

应用场景2：实时CDC（push CDC）

场景描述

- 满足零延迟的要求
- 变化分发机制一旦探测到变化，就把变化push给ETL程序
- 通常是通过可靠的传输机制来实现

注释：虽然面向消息和面向事件的集成方法在EAI产品中更为常见，但现在，已经有很多ETL工具厂商在他们的解决方案中提供这种功能，以满足高端、实时的商务应用需求。

关于CDC技术需要思考的问题

1 对运营系统的入侵（intrusion）程度

- 所有的CDC解决方案都会对系统造成一定程度的影响
- 最高级别的入侵是源代码入侵
- 程度稍低的入侵是“进程内”或“地址空间”入侵
- 入侵程度最低的解决方案不会影响应用的运营数据源

关于CDC技术需要思考的问题

2 捕捉延迟

- CDC解决方案的一个最主要的考虑因素
- 延迟会受到诸多因素的影响，比如：变化捕捉方法、对变化的处理和变化分发机制的选择
- 变化可以周期性地、高频率地甚至实时地进行分发
- 不同的BI应用对数据延迟的要求也不同， CDC解决方案应能进行灵活配置

关于CDC技术需要思考的问题

3 过滤和排序服务

- CDC解决方案应提供不同的服务实现对分发数据的过滤和排序
- 过滤可以保证只有需要的变化才被分发
- 排序则定义了变化被分发的顺序

关于CDC技术需要思考的问题

4 支持多个消费者

- 捕捉到的变化可能需要被分发到一个以上的消费者那里，比如多个ETL进程、数据同步应用和商务活动监测等等

- CDC解决方案需要支持多个消费者，每个消费者可能具有不同的延迟要求

关于CDC技术需要思考的问题

5 失败和恢复

- CDC解决方案必须保证变化能够被正确地分发，即使系统和网络发生异常

- 在进行恢复的时候，必须保证变化分发数据流从最近一次位置开始，而且必须保证在整个分发周期内满足变化的事务一致性

关于CDC技术需要思考的问题

6 主机和遗产数据源

- 专家估计[4]，主机系统仍然存储了大约70%的公司商业信息，主机仍然处理世界上大量的商业事务
- 主机数据源通常存储大量的数据，这就更需要有高效的方法来转移数据
- ETL和DW工具一般都要求关系型数据源，这就需要把非关系型数据源映射成关系型

关于CDC技术需要思考的问题

7 和ETL工具的无缝集成

- CDC解决方案与其他ETL工具之间互操作的难易程度
- 采用标准接口和插件的形式可以降低风险，并加快数据集成进度

提纲

- 与实时数据仓库相关的概念
- 实时数据仓库面临的挑战
- 连续数据集成
- 总结
- 参考文献

总结

- 与实时数据仓库相关的概念
  - 数据仓库5个发展阶段
  - 实时、及时和主动数据仓库概念
- 实时数据仓库面临的挑战和解决方案
  - 实时ETL
  - 实时数据的建模
  - OLAP查询和变化的数据
  - 可扩展性和查询竞争
  - 实时报警
- 连续数据集成
  - 脚本、ETL、EAI和CDC
  - CDC：变化捕捉代理、变化数据服务和变化分发机制

提纲

- 与实时数据仓库相关的概念
- 实时数据仓库面临的挑战
- 连续数据集成
- 总结
- 参考文献

参考文献

[1] “Zero-Latency Data Warehousing for Heterogeneous Data Sources and Continuous Data Streams”;Proceedings of iiWAS 2003, Fifth International Conference on Information and Web-based Applications Services, Jakarta, Indonesia; Austrian Computer Society (OCG) (2003), 3-902134-72-0; 55 – 64.

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

[3]KIMBALL, R. Real-time Partitions, in: Intelligent Enterprise Magazine. Vol.5(10),June 2002.

[4] Itamar Ankorion ,Change Data Capture – Efficient ETL for Real-Time BI Article published in DM Review Magazine,January 2005 Issue.

[5] Brobst, S. “Active Data Warehousing and Enterprise Application Integration,” Proceedings of Data Warehousing 2002:From Data Warehousing to the Corporate Knowledge Center, Physica-Verlag Heidelberg, November 12-13, 2002. pp. 15-23.

[6] Brobst, S. and J. Rarey. *The Five Stages of an Active Data Warehouse Evolution*. Teradata Magazine. Winter, 2001. 4.

[7] Stephen Brobst，Carrie Ballinger，*Active Data Warehousing: Why Teradata Warehouse is the Only Proven Platform*，October 2003

参考文献

[8] OLAP and the Active Data Warehouse. Wingspan Technology, Inc. WhitePaper.2003.
[9] Dan E. Linstedt.Active and Right-Time Data Warehousing Defined. http://www.b-eye-network.com/blogs/mt/mt-tb.cgi/346. January 30, 2006.
[10] Colin White. Data Integration: Using ETL, EAI, and EII Tools to Create an Integrated Enterprise. TDWI Report,November,2005.
[11] Simon Terr. Real-Time Data Warehousing 101. Published March 29, 2004
[12] Neil Raden.Real Time: Get Real, Part II. June 30, 2003 . http://www.intelligententerprise.com/030630/611warehouse1_1.jhtml?_requestid=1132505
[13] Ralph Kimball. Real time Partitions Feb, 2002. http://www.intelligententerprise.com/020201
[14] Neil Raden. Raden.Real Time: Get Real. June 17, 2003 . http://www.intelligententerprise.com/030617/610warehouse1_1.jhtml;jsessionid=RQKS3PEOAEEXYQSNDLPSKH0CJUNN2JVN
[15] 4]Langseth, J., "Real-Time Data Warehousing: Challenges and Solutions", DSSResources.COM, 02/08/2004.

参考文献

[16] Michael Haisten . The Real-Time Data Warehouse: The Next Stage in Data Warehouse Evolution . http://www.damanconsulting.com/company/articles/dwrealtime.htm
[17] Robert M. Bruckner and A.M. Tjoa. Capturing Delays and Valid Times in Data Warehouses—Towards Timely Consistent Analyses. Journal of Intelligent Information Systems, 19:2, 169–190, 2002
[18] Itamar Ankorion. Change Data Capture – Efficient ETL for Real-Time BI. Article published in DM Review Magazine, January 2005 Issue.

北京大学计算机系数据库实验室　　2006年8月