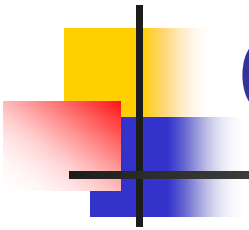
Data warehouse clustering on the web

European Journal of Operational Research

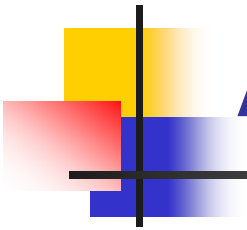
Volume 160, Issue 2 , 16 January 2005, Pages 353-364

- 指導教授: 楊東麟
- 報告人: 尹國正
- 94.12.12



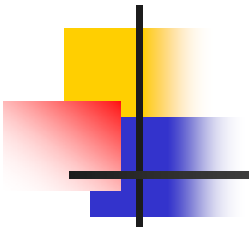
Outline

- Abstract
- Introduction
- Issue and challenge for refreshment
- System architecture
- Future work



Abstract

- Collaborative e-commerce environments
- We offer an approach that addresses refreshment in a federation of data warehouses.
- We propose an open multi-agent architecture for their incremental maintenance.



Introduction 2-1

- Ce-commerce poses new challenges to the MV maintenance problem, as extended enterprises have to integrate far more data originating outside the organization into a single repository.
- The streamlining of cross-company processes . . . is the next great frontier for reducing costs, enhancing quality, and speeding operations.

Introduction 2-2

- The problem we address in this paper can be stated as follows: ‘‘how to maintain MVs in environments where a data warehouse utilizes data from other data warehouses’’.

DWR in collaboratice e-commerce enviroment

- An integrated value system, which may include a number of data warehouses, requires a solution where the system of each participating business in the chain should be able to communicate with all others.
- The challenge is that the different maintenance policies of the MVs of each participating warehouse.

DWR in collaboratice e-commerce enviroment

- We define an MV as a hyper-view, because it provides a higher level of aggregation and consolidation of internal as well as external data sources.

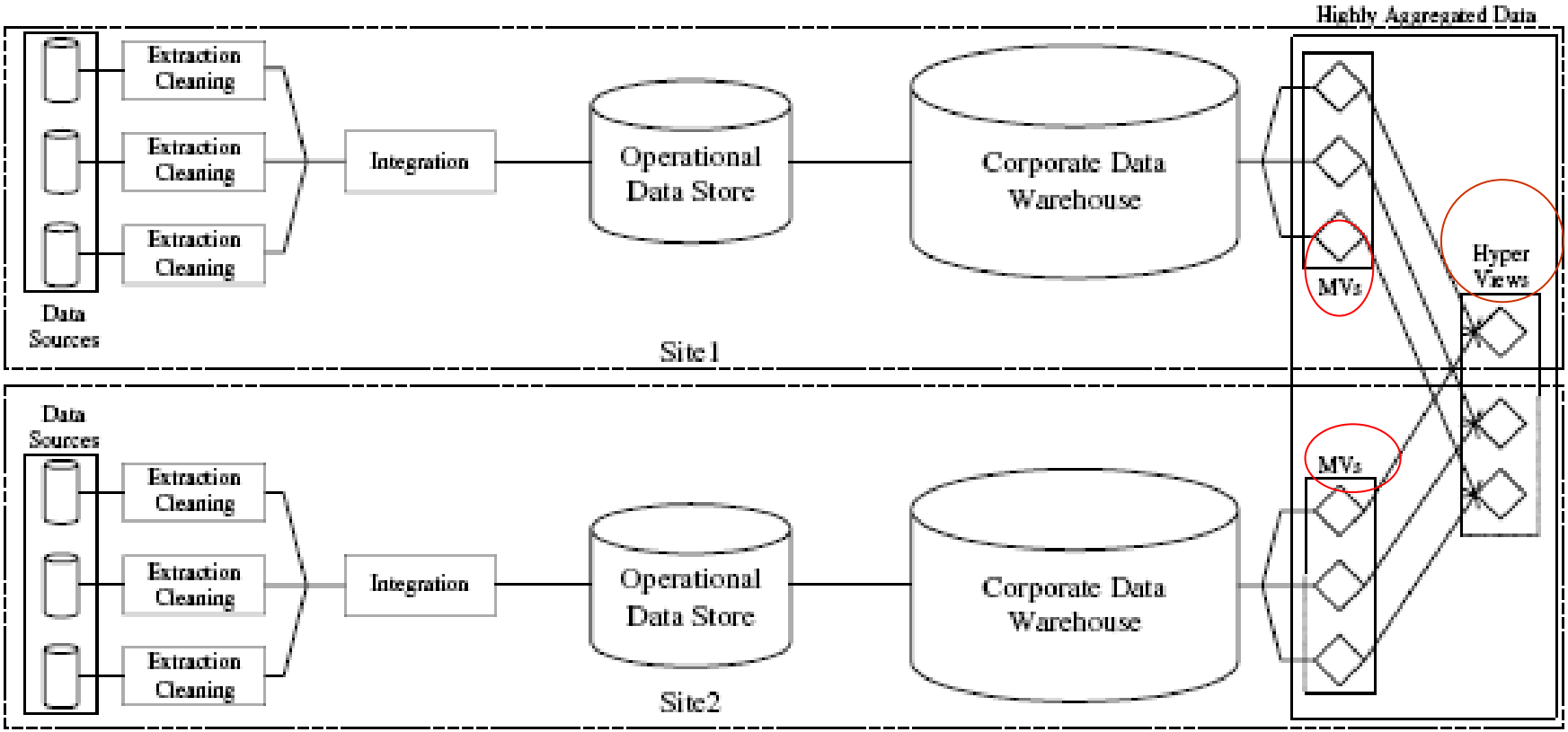


Fig. 2. Data warehouse architecture in Collaborative E-Commerce.

DWR in collaboratice e-commerce enviroment

- Forms of hyper-views
  - 1.Union over MVs

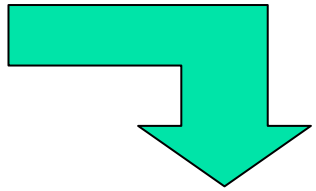
$$\mathrm{HV} = \bigcup_{i=1}^{n} \mathrm{MV}_i,$$

  - 2.Aggregate query

$$\mathrm{HV} = \mathrm{MV}_1 \bowtie \mathrm{MV}_2 \bowtie \cdots \bowtie \mathrm{MV}_n$$

全家超商 MV

七月	飲料	1000
七月	麵包	500
七月	日用品	3000
八月	飲料	1100
八月	麵包	450
八月	日用品	3100

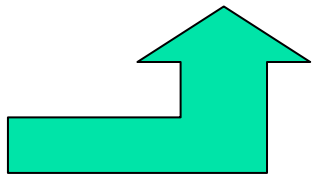


Hyper view(union)

		全家	萊爾富
七月	飲料	1000	900
七月	麵包	500	400
七月	日用品	3000	2000
八月	飲料	1100	950
八月	麵包	450	350
八月	日用品	3100	2100

萊爾富 MV

七月	飲料	900
七月	麵包	400
七月	日用品	2000
八月	飲料	950
八月	麵包	350
八月	日用品	2100

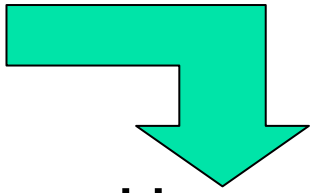


全家超商 MV

七月	飲料	1000
七月	麵包	500
七月	日用品	3000
八月	飲料	1100
八月	麵包	450
八月	日用品	3100

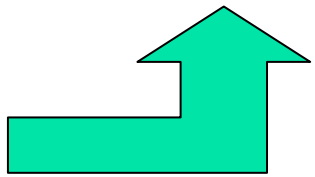
萊爾富 MV

七月	飲料	900
七月	麵包	400
七月	日用品	2000
八月	飲料	950
八月	麵包	350
八月	日用品	2100

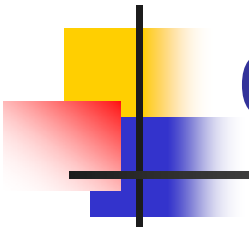


Hyper view (aggregate)

		加總
七月	飲料	1900
七月	麵包	900
七月	日用品	5000
八月	飲料	2050
八月	麵包	800
八月	日用品	5200

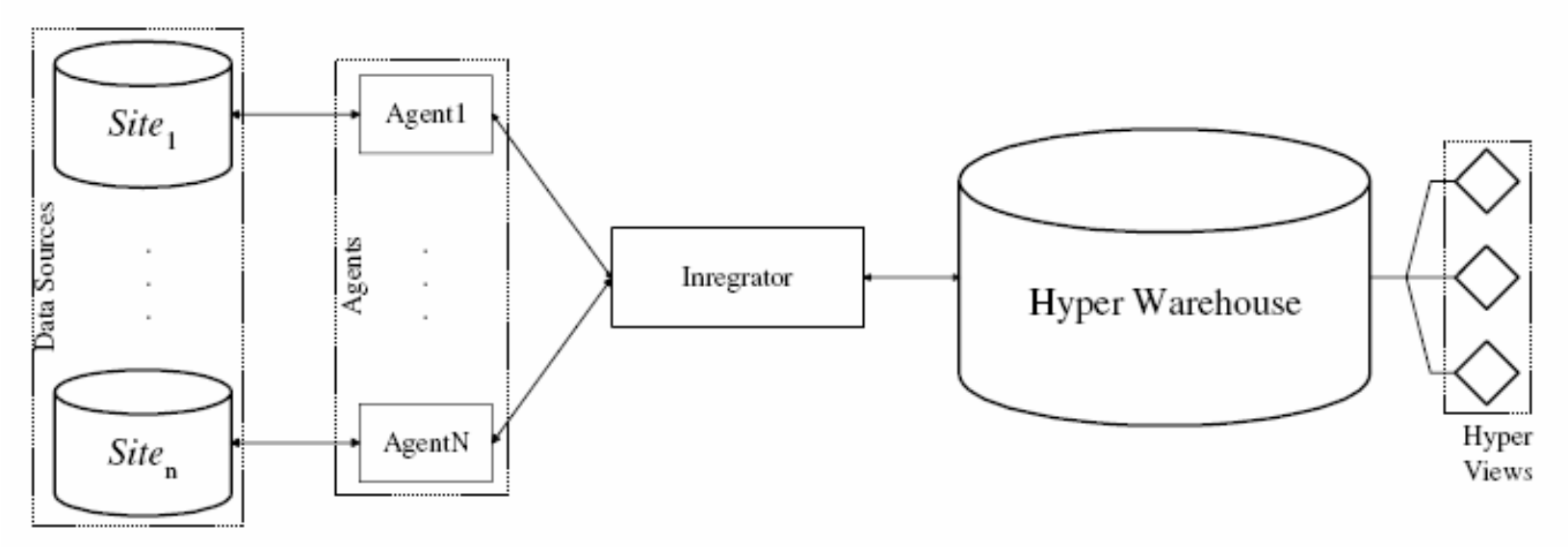


DWR in collaboratice e-commerce enviroment



- The union-based view is not materialized but recomputed every time.
- A hyper-view that stores union-based result sets has greater performance.

Over all system architecture



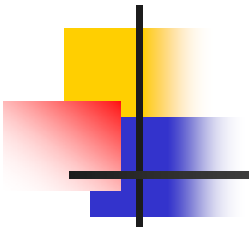
Main component

- Data sources: technologies such as RDBMSs or Data Warehouses (e.g. MVs).
- Integrator: information broker that is responsible for receiving the update notifications from the agents and installing the updates to the hyper-view in a unit of work. The messages that are exchanged between the agents and the integrator are flat files marked up with XML.

Main component

- Agents: server processes (pre-spawned) that control the interaction with a data source, and wait for requests from the integrator to monitor specific MVs or perform specific actions. They also transmit the updates to the integrator markedup with XML.

Integrator



- Sends a request to every agent to monitor the corresponding MV
- When an agent detects a change , marks up with XML the recordsd and transmits this XML file to the integrator.
- Then the integrator parses the XML file and installs the changes to the hyper-view on a FCFS (First-Come-First-Serve) basis.

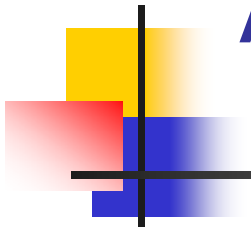
Agents

- An agent is a server process waiting for requests from the integrator.
- When a request to monitor a specific MV is present then the child agent operates under the following algorithm:

Agents algorithm

- 1. Parse the incoming request and extract the name of the view that is to be monitored (e.g. C_MV).
- 2. Analyse the definition of the view and extract the table names and fields that participate in the view.
- 3.Create a temporary table, named after the MV that keeps the base table deltas.

Agents algorithm



- 4. Install triggers on the C_MV that replicate the changes to the C_MV_TEMP.
- 5. Run periodically a query Q that selects all tuples from the temporary table,

Example for algorithm 3-1

CREATE VIEW C_MV as

(

SELECT c.p_name, sum(b.ps_availqty) as availqty

FROM partsupp b, part c

WHERE c.p_partkey = b.ps_partkey

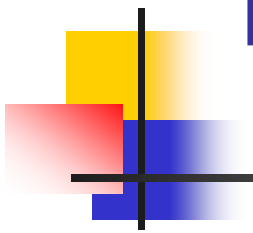
GROUP BY p_name

)

Example for algorithm 3-1

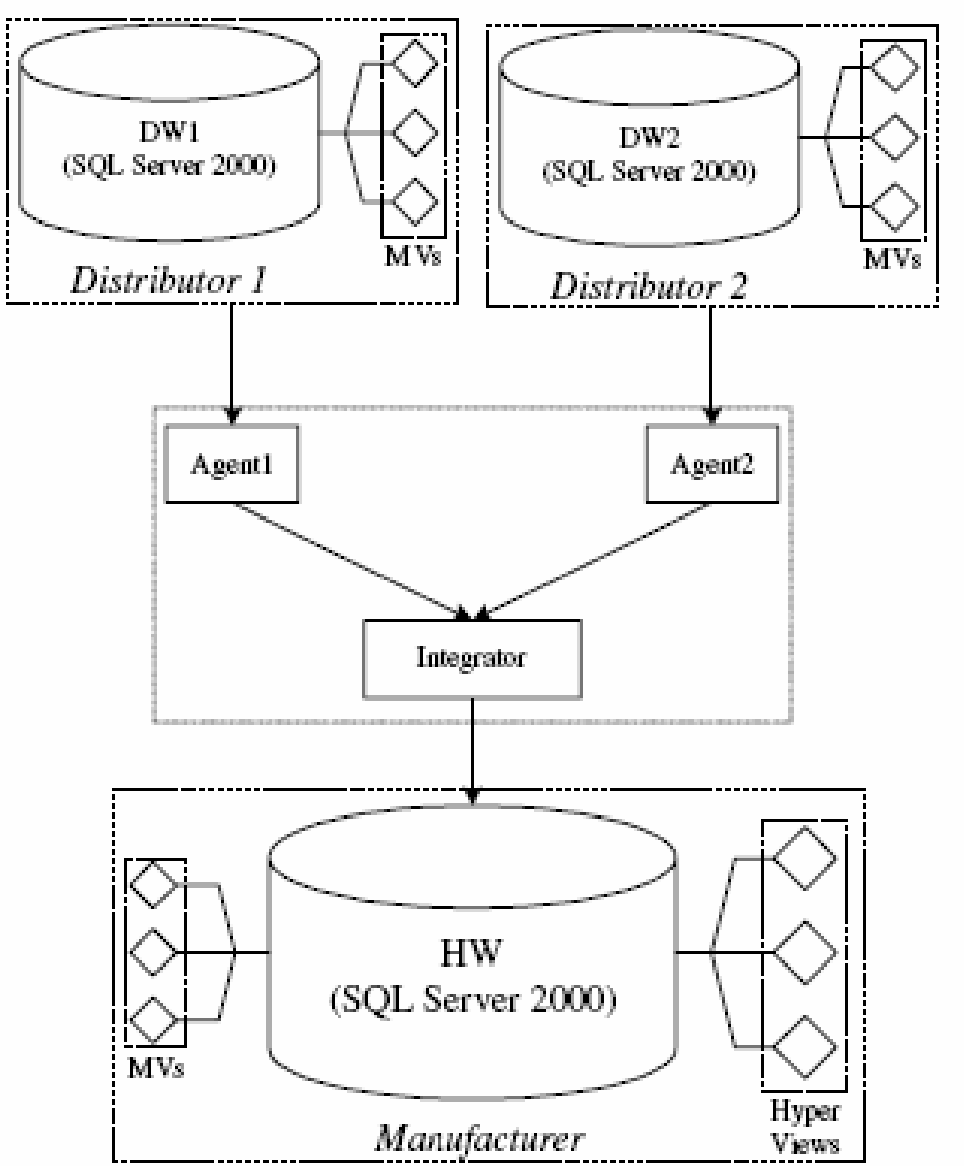
- 1.C_MV(extract name of view)
- 2. P_NAME, PS_AVAILQTY (extract table name and field)
- 3.C_MV_TEMP {P_NAME,PS_AVAILQTY, ACTION, STATUS, ROWID} (create temporary table)
  - ACTION: U:update D:delete I:insert

Example for algorithm 3-1



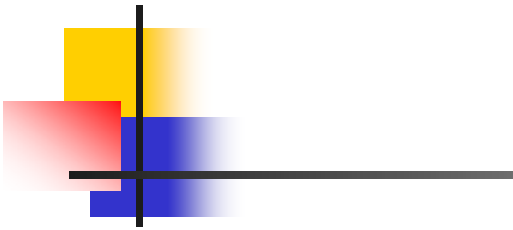
- 4. Install triggers on the C_MV
- 5. Run periodically a query

High level view of prototype's architecture



Conclusions and future work

- It focuses mainly on update propagation through MVs for single warehouses in the past. This paper draws attention to new set of challenges.
- Further research should concentrate on installing triggers directly on the MVs instead of the base tables.
- further research should deal with the consistency of data elements from different data sources having different semantics.



- Discussion