Warehouses & Virtual Databases offer integrated views

Focus: Sources are relational DBs. Integration specified by distributed view definition(s). Clients issue queries on views.

Virtual View -> Mediator needs Distributed Query Processor
Materialized view (warehouse) -> Mediator also needs storage & Incremental View Maintenance

Mediators as providers of view-based virtual views over distributed data

- How will the join happen?
- What if the source offers multiple data services instead of JDBC access? (besides CSE232 material)
Join types

- Mediator-based Join
  - Ship results of queries at mediator
- Parameterized Join
  - Right subquery is enhanced with selection on join attribute
  - For each join value of left hand side, execute another right subquery
- Data Ship Join
  - Insert the result of left hand side (lhs) in the db of right hand side (rhs).
  - Execute join at db of right hand side
- Semijoin Reduction Join
  - Send rhs parameters to lhs
  - (Data ship-like variation) Lhs sends to rhs the semijoin of its subquery with the parameters set.
  - Execute join at db of rhs
  - Also, variation that looks like mediator-based join

IVM: Deferred version

Snapshot 0  Problem: Find efficient view updates
$$\Delta V^0 = f(\Delta R_1^0, ..., \Delta R_n^0, V_0^0)$$
$$\Delta R_1^0, ..., \Delta R_n^0, V_0^0$$
$$\Delta R_1^0, ..., \Delta R_n^0, \Delta R_1^0, ..., \Delta R_n^0$$

View
$$V^0 = V(R_0^0, ..., R_n^0)$$
$$\Delta V^0 = f(\Delta R_1^0, ..., \Delta R_n^0)$$
$$V^1 = V(R_1^1, ..., R_n^1)$$

Database tables
$$R_0^0, ..., R_n^0$$
$$R_1^1, ..., R_n^1$$

From logs or intercepted by triggers


Snapshot 0  Problem: Find efficient view updates
$$\Delta V^0 = f(\Delta R_1^0, ..., \Delta R_n^0)$$
$$\Delta R_1^0, ..., \Delta R_n^0, V_0^0$$
$$\Delta R_1^0, ..., \Delta R_n^0, \Delta R_1^0, ..., \Delta R_n^0$$

View
$$V^0 = V(R_1^0, ..., R_n^0)$$
$$\Delta V^0 = f(\Delta R_1^0, ..., \Delta R_n^0)$$
$$V^1 = V(R_1^1, ..., R_n^1)$$

Database tables
$$R_1^0, ..., R_n^0$$
$$R_1^1, ..., R_n^1$$

From logs or intercepted by triggers

IVM: Self-maintaining version (not always possible)

Snapshot 0  Problem: Find efficient view updates
$$\Delta V^0 = f(\Delta R_1^0, ..., \Delta R_n^0)$$
$$\Delta R_1^0, ..., \Delta R_n^0, V_0^0$$
$$\Delta R_1^0, ..., \Delta R_n^0, \Delta R_1^0, ..., \Delta R_n^0$$

View
$$V^0 = V(R_1^0, ..., R_n^0)$$
$$\Delta V^0 = f(\Delta R_1^0, ..., \Delta R_n^0)$$
$$V^1 = V(R_1^1, ..., R_n^1)$$

Database tables
$$R_1^0, ..., R_n^0$$
$$R_1^1, ..., R_n^1$$

From logs or intercepted by triggers

Generalizations

- Multiple views
  - self maintenance may involve a view utilizing the other views in its computation
- Genuine updates
  - Not simulated via insertions/deletions
  - Insertions, deletions, updates on tables and views expressed as DML statements

Comparisons

<table>
<thead>
<tr>
<th>Materialized View</th>
<th>Virtual View</th>
</tr>
</thead>
<tbody>
<tr>
<td>High query performance</td>
<td>No need for yet another database</td>
</tr>
<tr>
<td>Queries not visible outside warehouse</td>
<td>More up-to-date data</td>
</tr>
<tr>
<td>Local processing at sources unaffected</td>
<td>Depending on specifics of IVM</td>
</tr>
<tr>
<td>Can operate when sources unavailable</td>
<td>Query needs can be unknown</td>
</tr>
<tr>
<td>Extra information at warehouse</td>
<td>Only query interface needed at sources</td>
</tr>
<tr>
<td>Modify, summarize (store aggregates)</td>
<td>Lower Total Cost of Ownership</td>
</tr>
<tr>
<td>Add Historical information</td>
<td></td>
</tr>
</tbody>
</table>

Notes 11