HyperQueries: Dynamic Distributed Query Processing on the Internet

Alfons Kemper, Christian Wiesner
VLDB Conference, 2001

Presented by: Yasemin Ugur
02/23/05
Introduction

• Context: Building an open market place on the internet (for data providers and clients)

• Aim: Providing a data integration and distributed query processing capabilities

• Requirements:
  – Keeping local sites autonomous and heterogeneous
  – Controlled access to site’s sensitive data
  – Scalable to large number of sites

• Approaches:
  – Mediator-based: Integrated schema of the data on the mediator, and sub-query executer and wrappers on each data source
  – ..
QueryFlow

- Market place host act as a mediator; contains integrated schema tables (e.g. Product catalog table), and provides registration services
- Each provider register their data to a virtual table, and defines hyperlinks for virtual attributes (and provide associated HyperQuery)

<table>
<thead>
<tr>
<th>ProductDescription</th>
<th>Supplier</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Battery, 12V 32 A</td>
<td>Supplier 1</td>
<td>hq://supplier1.com/Electrical/Price?ProdID=Cb1232</td>
</tr>
<tr>
<td>Battery, 12V 55 A</td>
<td>Supplier 1</td>
<td>hq://supplier1.com/Electrical/Price?ProdID=Cb1255</td>
</tr>
<tr>
<td>Tire: 175/65TR14</td>
<td>Supplier 2</td>
<td>hq://supplier2.com/Price?ProdKey=175/65TR14</td>
</tr>
<tr>
<td>Spark Plug VX</td>
<td>Supplier 3</td>
<td>hq://supplier3.com/PriceForUSA!Currency=USD?ID=12</td>
</tr>
</tbody>
</table>

```sql
SELECT h.*, p.Price as Price 
FROM HyperQueryInputStream h, Products p 
WHERE h.ProdID = p.ProdID 
```
HyperQuery Processing

- Simplified algorithm:
  - For each tuple of the virtual table being processed (Dispatch operator)
    - Process the hyperlink in the virtual attribute (s), and push the object-specific parameters, and input objects to the remote site that the hyperlink points to
    - Input object and object-specific parameter is evaluated by the query sub-plan (HyperQuery) on remote site, and output object is sent back to the market place (or surrounding sub-plan)
    - The sub-plan is instantiated once for the same URI (with different parms)
  - Merge all the result objects and return back to the user (Union operator)
- One-level (simple) or multi-level queries
HyperQuery Processing

- On remote site (SQL query example):

The operators such as join, selection are built-in operator implementation of the ObjectGlobe.
More details on QueryFlow

• Implementation specific optimization in order to reduce the amount of data transferred on network
  – Push selection predicates into the sub-plans on remote host
  – Avoid passing attributes that are not involved in the subplan to remote host
  – Use cache to avoid processing of the same virtual attribute (at least in intra-query) due to duplicates

• Additional attributes (that are not requested by the user) can be returned from sub-plans using a container attribute (initially specified by the schema on the mediator)
Registration

- A new service should be easily integrated into the system, and clients and providers should be able to access them without effort
  - E.g. Registration of new suppliers, products and HyperQueries
Other Systems (ObjectGlobe, Mariposa)

Running example: searching for the specific car part’s details (material, price)

- **ObjectGlobe:**
  - Each supplier associate their corresponding data source with the theme of car parts (with a set of predefined attributes), and register the attributes
  
  - Each data source has to be (as long as quality parameters are not violated) queried to see if a requested part is provided by this supplier
  
  The query processing enforces the user’s quality constraints (time, charge, cardinality data) and adapts accordingly

  Employ different query capabilities of different sites (not only data) for better query execution plan (e.g. performing a join of two themes on cycle provider that is physically close to the data providers, rather than shipping each scanned data back to client to perform this join operation)

Note: Query processing capabilities of the QueryFlow system are based on the *ObjectGlobe*
Mariposa query processing

- Data layout: horizontally partitioned table fragments, and replicas
- 3-modules: clients, middleware, local site manager.
- Each participant site can join the system by advertising its services and bidding on queries
- Client submits query with a required budget (how much the user is willing to pay for the query execution within \( t \) timeframe, whether a user is willing to sacrifice performance for a lower charge)
- Parser finds the metadata (e.g. location of each fragment) from the name server for each table requested in the FROM clause of query
- Query processing/optimization:
  - (1st phase): Generate a single-site query plan (ignores data distribution, assumes all fragments are located on a single site)
  - (2nd phase): Decompose the single query plan into fragmented query plan: decompose each referenced table node into subqueries (one per fragment), and joins are into join subquery for each pair of fragment join
  - Sends out bid request to sites that might be interested
  - Each bidder (local site) returns (Cost, Delay, Expiration)
  - Middleware accepts bids, constructs a final plan, and informs local sites of their jobs
- Sites (storage manager) can buy and sell fragments based on the estimated revenue from the fragments
Comparison

• Mariposa
  • Each supplier only register that their horizontal fragmentation for the car part table
  • Depending fragmentation criteria and user query, some of the fragment providers may not be required to be involved in the query execution plan

• Objective is to construct a query plan within user’s cost/$ and time budget
• De-centralized / point-to-point decision-making to find a optimized global query plan

• QueryFlow:
  • Each car parts supplier register its product (e.g. product name) & hyperlinks for attributes (virtual attribute) that should be obtained from the supplier itself each time the query is issued
  • Only the suppliers that have product name matching with the user query would be contacted to find the actual value for the other attributes

• The cost (time, price, ..) of the query is not considered
• Does not deal with limited/different query capabilities of sites
• Seem to be applicable to specific type of application (e.g. supply-chain)
References

• The URL for ObjectFlow project:
  http://www-db.in.tum.de/research/projects/OG/OnlineDemo/queryflow.shtml

• HyperQueries: Dynamic Distributed Query Processing on the Internet,


• An overview of Mariposa system:
  http://redbook.cs.berkeley.edu/redbook3/mariposa.html
Comments