**Data Oriented Middleware**

The Internet meets Databases

*A presentation by*

Yannis Papakonstantinou

*on joint works with*

Alin Deutsch,

Yannis Katsis,

Michalis Petropoulos,

Vasilis Vassalos

---

**Conventional Architecture for Unified Access to Data & Services**

- (Web) Client Application
- (Web) Client Application
- (Web) Client Application

**Integrated (XML) Global View / Ontology + Services**

**Mediator**

- Local Common Model (XML) View + Services
- Local Common Model (XML) View + Services

**Wrapper**

- Information + Service Source
- Information + Service Source

**Cache & Replication**
Approaches towards View-Based Data Integration

- **Local As View (LAV)**
- **Global As View (GAV)**

\[ \text{GLAV} = \text{GAV} + \text{LAV} \]

**Integration Specification Method**

- **Info Model & Query Language**
  - **Relational (SQL)**
  - **Object-Oriented**
  - **XML (XQuery)**

**Storage Method**

- **Warehousing (materialized views)**
- **On-Demand (virtual views)**

### Industrial Involvement w/ Integration

- **Enosys Software in 2000**
  - Enosys = greek for “union, merge, fusion”
- **Enterprise Information Integration product**
  - XML, virtual, GAV view
- **Product also sold under BEA’s Liquid Data brand since 2002**
  - Acquired by BEA in 2003
Current Enterprise Information Integration Deployments

- Small Domain
- Mostly Vertical Partition of Sources
- Primarily Application-Driven View or Identity View
- Integration Administrator/Developer in charge

Integrated Global View $V(M, S, E)$

View Builder (design time)

GAV View $V$

Mediator Query Processor (run time)

Local View $M$

Local View $S$

Local View $E$

Schemas

Data

Opportunities and Needs Presented by “Motivated” Communities

Communities

- Emerging Myriads of Internet Communities of
  - Myriads of sources and clients
  - Source owners motivated to participate
- EII does not address needs
  - Expensive
  - Bottleneck of Single Integration Admin
- Make building corresponding portals as easy as starting and participating in newsgroups
- Appropriate tools needed to enable source owner and client participation
Visual Tools Matter!
(example from the Enosys Query Builder)

1. Open & View Source Schemas in XML
2. Drag & Drop to Create Target XML View
3. Run & Test XQuery

XML Result
XQuery Based on Design Specs
Architecture and Goal of Easy-To-Use Tools

Mediator

Allow clients to easily build portals that query the mediator
so that each other’s needs are accommodated

Allow source owners to easily and independently register their source and services

Applications
Queries
Source Owners
Sources
Global View (by mediator)

Application Developers

Large-Scale Data Integration Systems

How can the user query and Browse the integrated data?
QURSED

What queries can the mediator answer for me?
CLIDE

How do I export my database services functionality?
RIDE-Services

How do I export my data?
RIDE

Web Domain
Web Forms & Reports
Application Domain
Application
Application
Integration Domain
Mediator
Global View Schema
Source Domain
Source Schema
Data Source
Web Service
Source Schema
Data Source
Web Service
Web Service
...
Dual Interactive Registration Problems

1. Register Source Given Queries
   - Apps
   - Queries
   - New Source and Services
   - Global View
   - Guide the source owner in registering a new source and services

2. Register Client Given Sources
   - New App
   - New Query
   - Guide the client in query/form writing
   - Global View
   - Source Services

Information-Specific Source Owner’s Goal

Server Side

- Apps
- Queries
- Global View
- New Source
- Security
- Need for data cleaning
- Make source information visible to the apps
- Expose minimum amount of source information

Neglect registering services
How to achieve this Goal

Before

Apps → Queries

Global View

New Source

1. Look at all sources & queries
2. Decide how to register your source

Now

Apps → Queries

Global View

New Source

1. Follow the suggestions of the interface

Our Goal in Source Registration

Guide the source owner visually through the registration of the source

so as to contribute information to the answer of the queries

while exposing the minimum information possible and/or minimizing effort
The Problem

Client Queries

Mediator (Global DB)

Sources (Actual Local DBs)

What is the contribution of source S to the result of the query Q?
The Problem

- What is the contribution of source $S$ to the result of the query $Q$?

$Q$: cars

S is Self Sufficient w.r.t. $Q$

$Q$: cars JOIN reviews

S is Now Complementary w.r.t. $Q$

Relational Schemas: Local and Global

- Relational Schemas
- Visual Representation
Source Registration using GLAV Mappings

- **Source Registration:**
  Correspondence between a source schema and the global schema
  = Set of Mapping Constraints of the form $(U \subseteq V)$

  - **Open World**
  - **Global and Local As View (GLAV)**

Target Constraints

- **Constraints on the global schema**
  = Set of Constraints of the form $(U \subseteq V)$

  - **Also Expresses Dependencies (PKs, Ref Integrity, ...)**
Visual Representation of Mappings (1)

- Visual Representation (IBM Clio)

Business Magazine: Provides Carmake and Origin

$U_1 \subseteq V_1$  

\[
\begin{align*}
U_1(C, O) & :- \text{make}(C, O, S) \\
V_1(C, O) & :- \text{brand}(C, O)
\end{align*}
\]

21

Visual Representation of Mappings (2)

- Visual Representation (IBM Clio)

Car Magazine: Provides Model, Carmake and Baseprice

22
Example of Target Constraint

- (Model, Carmake) is a PK of car

```
G
  car
    Model
    Carmake
    Doors
    Baseprice
    brand
    Origin

(U1 \subseteq V1) : U1(M, C, D1, B1, D2, B2) :- car(M, C, D1, B1),
car(M, C, D2, B2)
V1(M, C, D, B, D, B) :- car(M, C, D, B)
```

Query Semantics

- Queries in UCQ=
- Set of Possible Global Instances
  Set of global instances that satisfy all constraints
- Query Answers = Set of Certain Answers
  The tuples appearing in the answer to Q for any possible global instance

```
Possible global instances
Answer to Q for any of the possible global instances
Certain Answers to Q
```
Source Instance's Contribution

For given instances of the sources

Contribution to Q of Source Instance

\[ \text{= The tuples in answer of Q not provided by the other sources} \]

Source Registration’s Contribution

- Source Registration: Source Mappings
- Degrees of Source Registration’s Contribution
  1. Self Sufficient
  2. Now Complementary
  3. Later Complementary
  4. Unusable
**Self Sufficient Registration: Example**

Example

Baseprices of Models

<table>
<thead>
<tr>
<th>car</th>
<th>Model</th>
<th>Carmake</th>
<th>Doors</th>
<th>Baseprice</th>
</tr>
</thead>
<tbody>
<tr>
<td>car</td>
<td>green</td>
<td>BMW</td>
<td>Green</td>
<td>45K</td>
</tr>
</tbody>
</table>

![Diagram](image)

Green Registration is Self Sufficient

---

**Self Sufficient Registration: Definition**

1. **Self Sufficient**

   ∃ Source instance

   s.t.

   The source has a non empty contribution in the absence of the other sources

   ![Diagram](image)
### Baseprices of Models by German manufacturers

<table>
<thead>
<tr>
<th>Car</th>
<th>Model</th>
<th>Carmake</th>
<th>Doors</th>
<th>Baseprice</th>
<th>Origin</th>
</tr>
</thead>
<tbody>
<tr>
<td>BMW</td>
<td>M3</td>
<td>BMW</td>
<td>?</td>
<td>45K</td>
<td>Germany</td>
</tr>
</tbody>
</table>

Green Registration is **Now Complementary**

### Now Complementary Registration: Definition

**Now Complementary**

\[
\text{Not Self Sufficient} \quad \& \quad \exists \text{Source instances s.t.}
\]

The source has a non empty contribution in combination with the other existing sources

\[ \neq \emptyset \]
Later Complementary Registration: Example

Example

Baseprices of Models by German manufacturers

Green Registration is Later Complementary

Later Complementary Registration: Definition

Later Complementary

Not Self Sufficient & Not Now Complementary & ∃ Potential future sources & Source instances

The source has a non empty contribution in combination with the future sources

Answer to Q ≠ ∅
Unusable Registration: Example

Unusable Registration: Definition

Unusable
Not Self Sufficient & Not Now Complementary & Not Later Complementary
⇔
The source has a empty contribution regardless of what sources enter the system
Subtleties for Unusable Registrations

Example

Baseprices and Doors of Models

Green Registration is Unusable

<table>
<thead>
<tr>
<th>car</th>
<th>Model</th>
<th>Carmake</th>
<th>Doors</th>
<th>Baseprice</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BMW</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>M3</td>
<td></td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Impresence of PK Unusable Example becomes Later Complementary

Example

Baseprices and Doors of Models

Green Registration is Later Complementary

<table>
<thead>
<tr>
<th>car</th>
<th>Model</th>
<th>Carmake</th>
<th>Doors</th>
<th>Baseprice</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BMW</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>M3</td>
<td></td>
<td>2</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>45K</td>
</tr>
</tbody>
</table>
Decidability Results

Overview: What is decidable

<table>
<thead>
<tr>
<th>Degree</th>
<th>Target constraints</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>None</td>
<td>Primary keys</td>
<td>Primary keys + Referential Integrity Constraints</td>
</tr>
<tr>
<td>Self Sufficient</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Now complementary</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Later complementary</td>
<td>Yes</td>
<td>Yes</td>
<td>?</td>
</tr>
<tr>
<td>Unusable</td>
<td>Yes</td>
<td>Yes</td>
<td>?</td>
</tr>
</tbody>
</table>

Issues

<table>
<thead>
<tr>
<th>Unique client query</th>
<th>Vs</th>
<th>Multiple client queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contribute to: - all queries?</td>
<td>- one query?</td>
<td>- specific queries?</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data independence</th>
<th>Vs</th>
<th>Data dependence</th>
</tr>
</thead>
<tbody>
<tr>
<td>e.g. M₁: cars, refPrices, M₂: reviews, Q: cars JOIN reviews JOIN refPrices, DB₁: cars, refPrices (Audis), DB₂: reviews (Hondas), (M₂, Q) now-complementary, but Certain Answers for Instances DB₁, DB₂ = ∅</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Putting it all together

Architecture

Query Answering / Mappings / Schemas

Contribution

4 categories:
Self Sufficient / Now Complementary / Later Complementary / Unusable

Goal

Guide the source owner visually through the registration of the source so as to raise contribution to the answer of the queries while exposing the minimum info possible and/or minimizing effort

Example 1

Without primary keys in the target

Local Schemas

Global Schema

Query

Unusable

BLUE: Map at least one of the groups
Example 1

**Without primary keys in the target**

**Local Schemas**
- AutoTrader
  - `car` (id, cmodel, ad, vin, carId, price)

**Global Schema**
- Community
  - `car` (model, drive, review, model, quality, usedAd, vin, model, price, refPrice, model, condition, price)

**Query**
- AppQuery
  - `car` (model, drive, review, model, quality, usedAd, vin, model, price, refPrice, model, condition, price)

**Example 1**

**Without primary keys in the target**

**Local Schemas**
- AutoTrader
  - `car` (id, cmodel, ad, vin, carId, price)

**Global Schema**
- Community
  - `car` (model, drive, review, model, quality, usedAd, vin, model, price, refPrice, model, condition, price)

**Query**
- AppQuery
  - `car` (model, drive, review, model, quality, usedAd, vin, model, price, refPrice, model, condition, price)
Example 2

**With primary keys in the target**

**Local Schemas**
- AutoTrader
  - `car`*
    - `id`
    - `model`
    - `ad`
      - `vin`
      - `carId`
      - `price`

**Global Schema**
- Community
  - `car`*
    - `model`
    - `drive`
    - `review`*
      - `model`
      - `quality`
    - `usedAd`*
      - `vin`
      - `model`
      - `price`
    - `refPrice`*
      - `model`
      - `condition`
      - `price`

**Query**
- `AppQuery`
  - `car`*
    - `model`
    - `drive`
    - `review`*
      - `model`
      - `quality`
    - `usedAd`*
      - `vin`
      - `model`
      - `price`
    - `refPrice`*
      - `model`
      - `condition`
      - `price`

---

Later

Complementary

---

Unusable
Lessons learned

Target constraints make a difference

1. To merge data with that of other sources (become complementary):
   - Pick a relation and provide...
     - In absence of primary keys
       - ...all its attributes asked by the query
     - In presence of primary keys
       - ...its primary key and one of its attributes asked by the query
   - The number of choices increases in presence of primary keys

2. Foreign keys on the target affect the suggestions

Large-Scale Data Integration Systems

- How can the user query and Browse the integrated data?
  - QURSED
- What queries can the mediator answer for me?
  - CLIDE
- How do I export my database services functionality?
  - RIDE-Services
- How do I export my data?
  - RIDE
Query Forms and Reports

Challenge: Heterogeneity

- Many combinations must be accommodated
- Optional
- Nesting
- Heterogeneity: Condition Both Cylindrical & Rectangular Sensors
  - Cylindrical Sensors: Diameter, Barrel Style
  - Rectangular Sensors: Height, Width

Current Practices

- Macromedia Dreamweaver: Fixed Parameterized SQL Statements, Custom Code Required for Combinations of Conditions, Very Tricky for Semistructured Data
Capturing Heterogeneity

**XML Schema**
- Captures heterogeneity
- XML
  - Just a language
  - Not a program

**XQuery Standard (W3C)**
- XML Query Language
- Behind the scenes

---

### XQuery: Expressive But Verbose

```xml
<html>
<body>
<table>{
  FOR $doc IN accessSource('XMLDB'), $S IN $doc/sensors, 
  $MAN IN $S/manufacturer, $PROD IN $MAN/product, 
  $SPEC IN $PROD/specs, $PROTS IN $SPEC/protection_ratings, 
  $PROT1 IN $PROTS/protection_rating, $PART IN 
  $PROD/part_number, 
  $DIST IN $SPEC/sensing_distance, $BODY IN 
  $SPEC/body_type, 
  WHERE $PROT1 = "NEMA3"
  AND (SOME $CYL IN $BODY/cylindrical, 
    SATISFIES $DIA <= 20 AND $DIA <= 40)
  OR (SOME $REC IN $BODY/rectangular, 
      SATISFIES $HEI <= 20 AND $WID <= 40)
  ORDER BY $NAME DESCENDING, $DIST 
RETURN GROUPBY $PROD, $NAME, $DIST AS 
<tr>{
  FOR $IMG IN $PROD/image 
RETURN GROUPBY $IMG AS <img src="{$IMG}" />
</tr>
}</table>
</body>
</html>
```
The QURSED Solution

- XML Schema-Driven
- Declarative!
  - Separation of querying & presentation
  - Model-View-Controller pattern
- No programming required!
  - Editor provides visual actions
  - Automatic report construction
- Query Set Specification (QSS)

---

Tree Query Language (TQL)

**Condition Tree**
- AND $DIST >= 5
  - sensors manufacturer product
  - part_number $PART speed
  - sensing_distance $DIST body_type
  - OR
  - AND $DIA <= 30 AND $DIA <= 40
    - cylindrical $CYL diameter $DIA barrel_style $BAR
  - AND $WID <= 30 AND $WID <= 40
    - rectangular $REC height $WID width $WID

**Result Tree**
- HTML
- table
- group by ($PART, $DIST)
- order by ($PART asc)
- $PART $DIST $DIA $HEI $WID $CYL $REC $BAR

**XML Data Tree**

**XML Result Tree**

<table>
<thead>
<tr>
<th>$PART</th>
<th>$DIST</th>
<th>$CYL</th>
<th>$DIA</th>
<th>$HEI</th>
<th>$WID</th>
</tr>
</thead>
<tbody>
<tr>
<td>A123</td>
<td>7</td>
<td>Cylindrical</td>
<td>17</td>
<td>Smooth</td>
<td></td>
</tr>
<tr>
<td>B123</td>
<td>50</td>
<td>Rectangular</td>
<td>10</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>C123</td>
<td>90</td>
<td>Rectangular</td>
<td>20</td>
<td>40</td>
<td></td>
</tr>
</tbody>
</table>
Query Set Specification (QSS): Addresses Condition Combinations

- Parameterized boolean expressions
- Multiple boolean expressions per AND node

Run-Time: QSS to TQL Queries

- Instantiate parameters
- Activate fragments
- TQL query
- XQuery Expression
**Fragment Dependencies**

**Address Heterogeneity**

\[
DIA \leq \#DIA \text{ AND } BAR \leq \#BAR
\]

\[
HEI \leq \#HEI \text{ AND } WID \leq \#WID
\]

\[
f_1 < f_2, \quad \#BODY = 'Cylindrical'
\]

\[
f_2 < f_1, \quad \#BODY = 'Rectangular'
\]

**Meaningful Queries**

**QURSED Editor: No Coding**

- XML Schema
- Condition Fragments
- Query Form
Building Reports

Report follows schema nesting

Large-Scale Data Integration Systems

How can the user query and Browse the integrated data?

QURSED

What queries can the mediator answer for me?

CLIDE

How do I export my database services functionality?

RIDE-Services

How do I export my data?

RIDE
### Running Example

#### Parameterized Views

<table>
<thead>
<tr>
<th>Dell</th>
<th>Cisco</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Schema</strong></td>
<td><strong>Schema</strong></td>
</tr>
<tr>
<td><strong>Computers</strong> (cid, cpu, ram, price)</td>
<td><strong>Routers</strong> (rate, standard, price, type)</td>
</tr>
<tr>
<td><strong>NetCards</strong> (cid, rate, standard, interface)</td>
<td></td>
</tr>
<tr>
<td><strong>Views</strong></td>
<td><strong>Views</strong></td>
</tr>
<tr>
<td><strong>V1 ComByCpu(cpu) → (Computer)</strong>*</td>
<td><strong>V3 RouByTypeW() → (Router)</strong>*</td>
</tr>
<tr>
<td>SELECT DISTINCT Com1.*</td>
<td>SELECT DISTINCT Rou1.*</td>
</tr>
<tr>
<td>FROM Computers Com1</td>
<td>FROM Routers Rou1</td>
</tr>
<tr>
<td>WHERE Com1.cpu = cpu</td>
<td>WHERE Rou1.type = 'Wired'</td>
</tr>
<tr>
<td><strong>V2 ComNetByCpuRate(cpu, rate) → (Computer, NetCard)</strong>*</td>
<td><strong>V4 RouByTypeWL() → (Router)</strong>*</td>
</tr>
<tr>
<td>SELECT DISTINCT Com1.<em>, Net1.</em></td>
<td>SELECT DISTINCT Rou1.*</td>
</tr>
<tr>
<td>FROM Computers Com1, Network Net1</td>
<td>FROM Routers Rou1</td>
</tr>
<tr>
<td>WHERE Com1.cid = Net1.cid</td>
<td>WHERE Rou1.type = 'Wireless'</td>
</tr>
<tr>
<td>AND Com1.cpu = cpu</td>
<td>AND Net1.rate = rate</td>
</tr>
</tbody>
</table>

### Running Example

#### Integrated Schema

- Integrated schema puts together the Dell and Cisco schemas

**Attribute Associations**
- (Computers.cid, NetCards.cid)
- (NetCards.rate, Routers.rate)
- (NetCards.standard, Routers.standard)
Sophisticated Mediators Make Feasible Queries Hard to Predict

**Feasible Queries FQ**
- Equivalent CQ query rewritings using the views
- Might involve more than one views
- Order might matter

*Query: Feasible*
Get all ‘P4’ Computers, together with their NetCards and their compatible ‘Wireless’ Routers

*Query: Infeasible*
Get all Computers

---

**Problem**

1. Large number of sources
2. Large number of views (web-services)
3. Mediator capabilities

Developer formulates an application query

⇒ Is an application query feasible?
⇒ If not, how do I know which ones are feasible?

Previous options:
- The developer had to browse the view definitions and somehow formulate a feasible query
- Or formulate queries until a feasible one is found (trial-and-error)
A query formulation interface, which interactively guides the developer toward feasible queries by employing a coloring scheme.
CLIDE Interface

- Table, selection, projection and join actions
- Feasibility Flag
- Color-based suggestions

Example Interaction

Snapshot 1

Yellow ➜ required action
- All feasible queries require this action

White ➜ optional action
- Feasible queries can be formulated w/ or w/o these actions
Example Interaction

**Snapshot 2**

**Blue** → required choice of action

- At least one feasible query cannot be formulated unless this action is performed

```
SELECT DISTINCT Com1, ram, Com1 price
FROM Computers Com1, NetCards Net1
WHERE Com1 cpu = 'P4'
```

Join Lines:
- Only yellow and blue are displayed
- Must appear in Attribute Associations
Example Interaction

**Snapshot 4**

- * ➔ any other constant
- **Red** ➔ prohibited action
  - Does not appear in any feasible query
  - Lead to “Dead End” state
Example Interaction

Snapshot 6

CLIDE Properties

- **Rapid Convergence**
  - At every step, yellow and blue actions lead to a feasible query in a minimum number of steps

- **Completeness of Suggestions**
  - Every feasible query can be formulated by performing yellow and blue actions at every step

- **Minimality of Suggestions**
  - At every step, only a minimal number of actions are suggested, i.e., the ones that are needed to preserve completeness
Interaction Graph

- Nodes are queries
  - One for each $q \in CQ$
- Edges are actions
  - Table, selection, projection and join actions
- Green nodes are feasible queries
- Infinitely big structure
  - All CQ queries
  - All possible combinations of actions formulating them

Interaction Graph: Colorable Actions

- Colorable actions $A_c$ label outgoing edges of the current node
Interaction Graph: Colors

- **Yellow** action $\alpha$
  - **Every path** from current node $n$ to a feasible node contains $\alpha$

- **Blue** action $\alpha$
  - At least one feasible query cannot be formulated unless this action is performed (minimality)

- **Red** action $\alpha$
  - **No path** to a feasible node contains $\alpha$

Color Determined By a Finite Set of Feasible Queries

Challenge: **Infinitely Many Feasible Queries**

Solution: **Closest Feasible Queries $FQC$**

- $FQC$ is sufficient to color actions in $A_C$
- **Theorem:** Set of Closest Feasible Queries is Finite

Challenge: **How far can the Closest Feasible Queries $FQC$ be?**

Solution: **Based on Maximally Contained Queries $FQ_{MC}$**
Large-Scale Data Integration Systems

How can the user query and Browse the integrated data? **QURSED**

What queries can the mediator answer for me? **CLIDE**

How do I export my database services functionality? **RIDE-Services**

How do I export my data? **RIDE**