

Data Oriented Middleware

The Internet meets Databases

A presentation by

Yannis Papakonstantinou

on joint works with

Alin Deutsch,

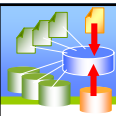
Yannis Katsis,

Michalis Petropoulos,

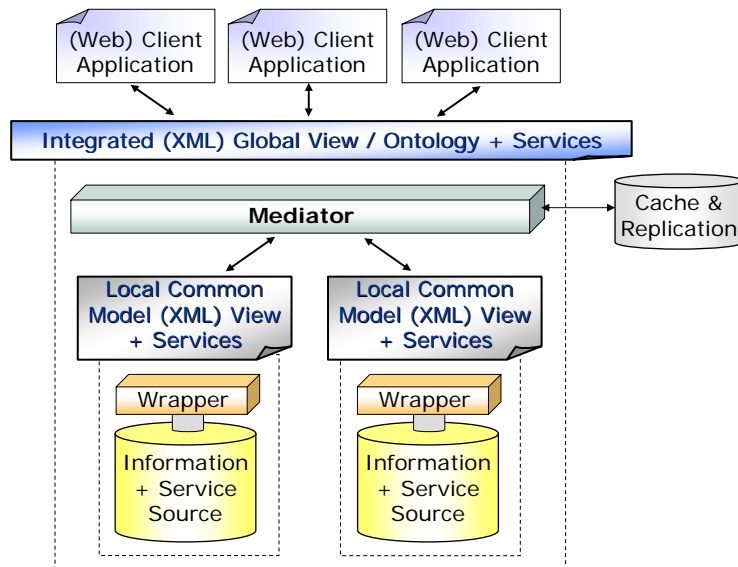
Vasilis Vassalos

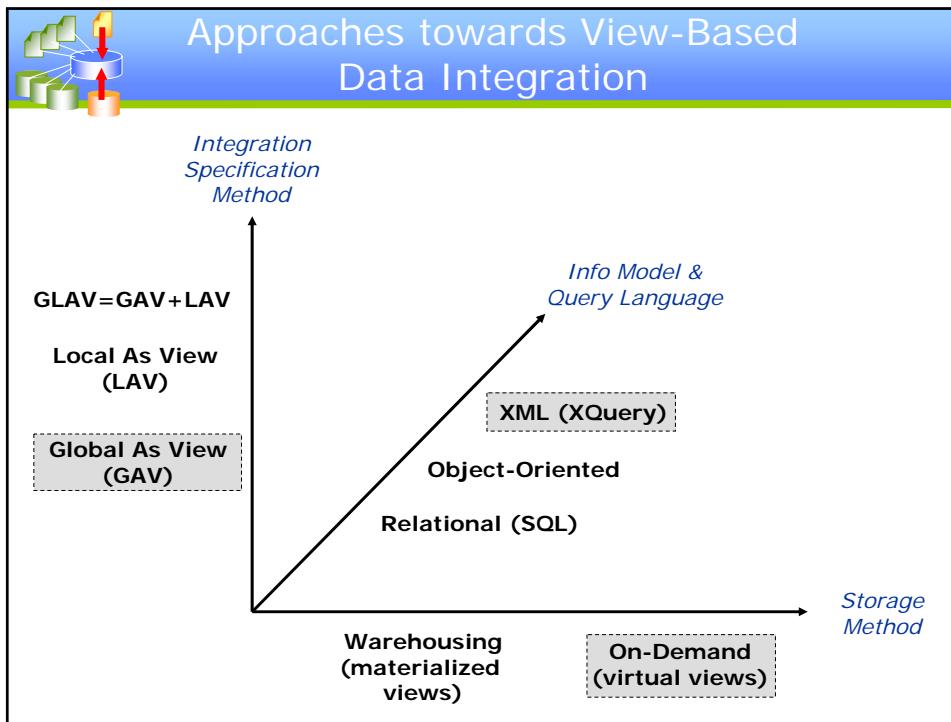


CSE Department

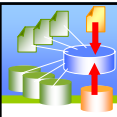


Conventional Architecture for Unified Access to Data & Services





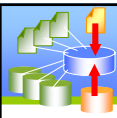
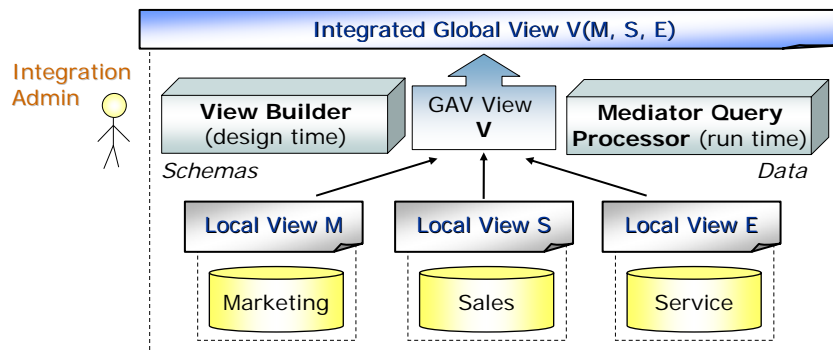
-
- Industrial Involvement w/ Integration**
- Enosys Software in 2000
 - Enosys = greek for “union, merge, fusion”
 - Enterprise Information Integration product
 - XML, virtual, GAV view
 - Product also sold under BEA's Liquid Data brand since 2002
 - Acquired by BEA in 2003



Current Enterprise Information Integration Deployments

ENTERPRISE

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



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

TARGET SCHEMA (XML VIEW)

AUTOMATICALLY GENERATED MAPS

Source	Target
WFAD0\wfado\ORDER_DETAIL\$TYPE	wfa_2a\PURCHASE_ORDERS\PURCHASE_ORDERLSR\$SERVIC
WFAD0\wfado\ORDER_DETAIL\$STATE	wfa_2a\PURCHASE_ORDERS\PURCHASE_ORDERLSR\$SERVIC
WFAD0\wfado\ORDER_DETAIL\$CIRCUIT_ID	wfa_2a\PURCHASE_ORDERS\PURCHASE_ORDERLSR\$SERVIC
WFAD0\wfado\ORDER_DETAIL\$IDO_STATUS	wfa_2a\PURCHASE_ORDERS\PURCHASE_ORDERLSR\$SERVIC
{RequestManager}\db\CONFIRMATION\ORD	wfa_2a\PURCHASE_ORDERS\PURCHASE_ORDERLSR\$SERVIC
{RequestManager}\db\LOC_WORKLIST\LAST_ACT	wfa_2a\PURCHASE_ORDERS\PURCHASE_ORDERLSR\$ACT_A

3 RUN & TEST XQUERY

XQUERY BASED ON DESIGN SPECS

XML RESULT

```

<!-- Generated by Data View Builder 2.0 -->
<PURCHASE_ORDERS>
{
for $RequestManager.PON_TABLE_1 in document("RequestManager")/db/PON_TABLE
for $RequestManager.LOC_WORKLIST_2 in document("RequestManager")/db/LOC_WORKLIST
where ($RequestManager.PON_TABLE_1/SER eq $RequestManager.LOC_WORKLIST_2/SRKEY)
return
<PURCHASE_ORDER>
<SER> { xfi:data($RequestManager.PON_TABLE_1/SER) }</SER>
<PON> { xfi:data($RequestManager.PON_TABLE_1/PON) }</PON>
<VERSION> { xfi:data($RequestManager.PON_TABLE_1/VERSION) }</VERSION>
<CLECID> { xfi:data($RequestManager.PON_TABLE_1/CLECID) }</CLECID>
<STATE> { xfi:data($RequestManager.PON_TABLE_1/STATE) }</STATE>
<SRKEY> { xfi:data($RequestManager.PON_TABLE_1/SRKEY) }</SRKEY>
<LSO>
<SRKEY> { xfi:data($RequestManager.LOC_WORKLIST_2/SRKEY) }</SRKEY>
<REPID> { xfi:data($RequestManager.LOC_WORKLIST_2/REPID) }</REPID>
<TERMD> { xfi:data($RequestManager.LOC_WORKLIST_2/TERMD) }</TERMD>
<DB_NAME> { xfi:data($RequestManager.LOC_WORKLIST_2/DB_NAME) }</DB_NAME>
<PON> { xfi:data($RequestManager.LOC_WORKLIST_2/PON) }</PON>
}

```

Query Parameter

Name	Value	Type

Choose size of result to retrieve:

Fetch entire result before printing

Begin printing result immediately

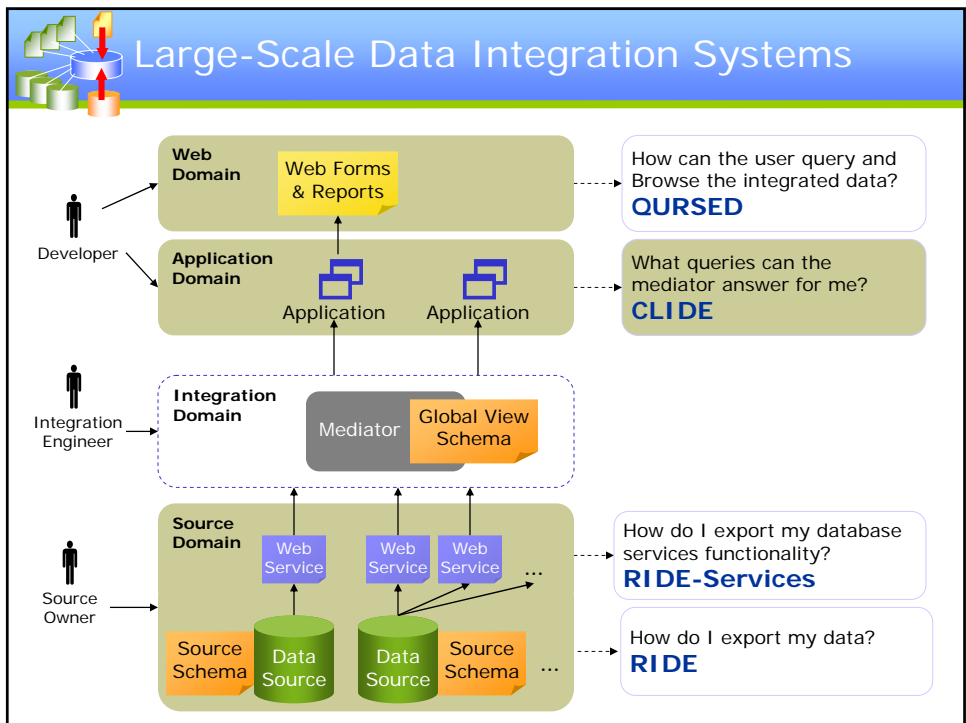
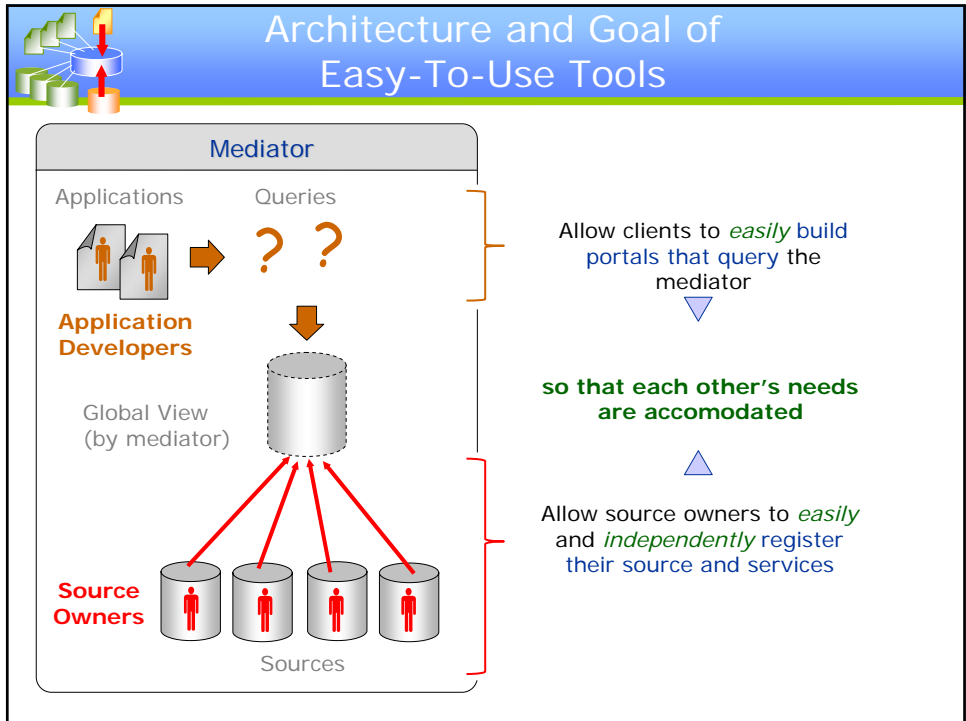
Number of results per request:

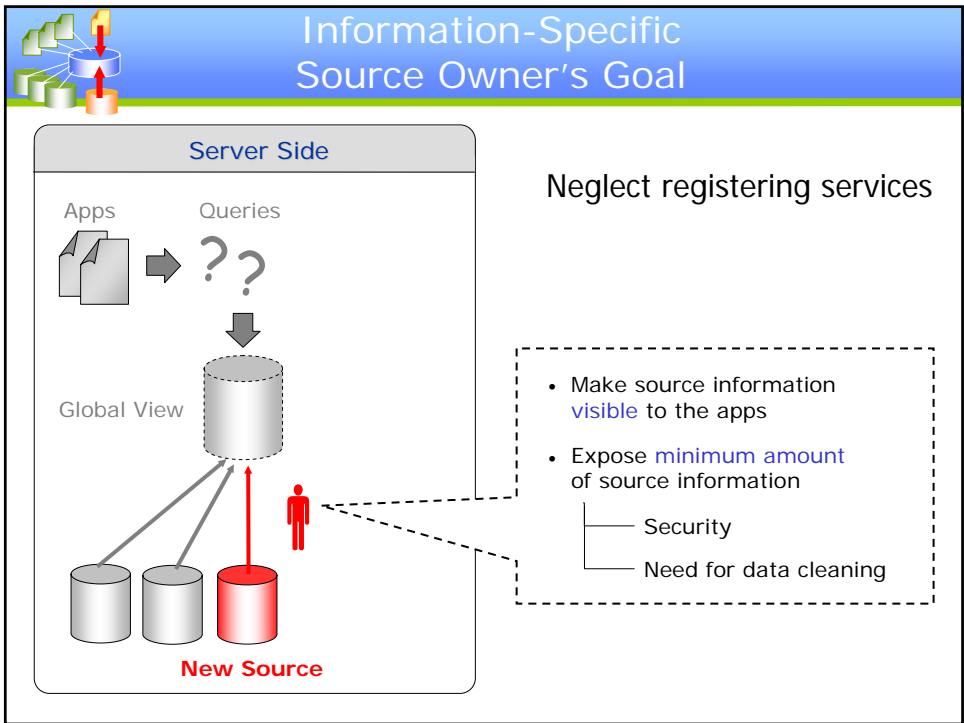
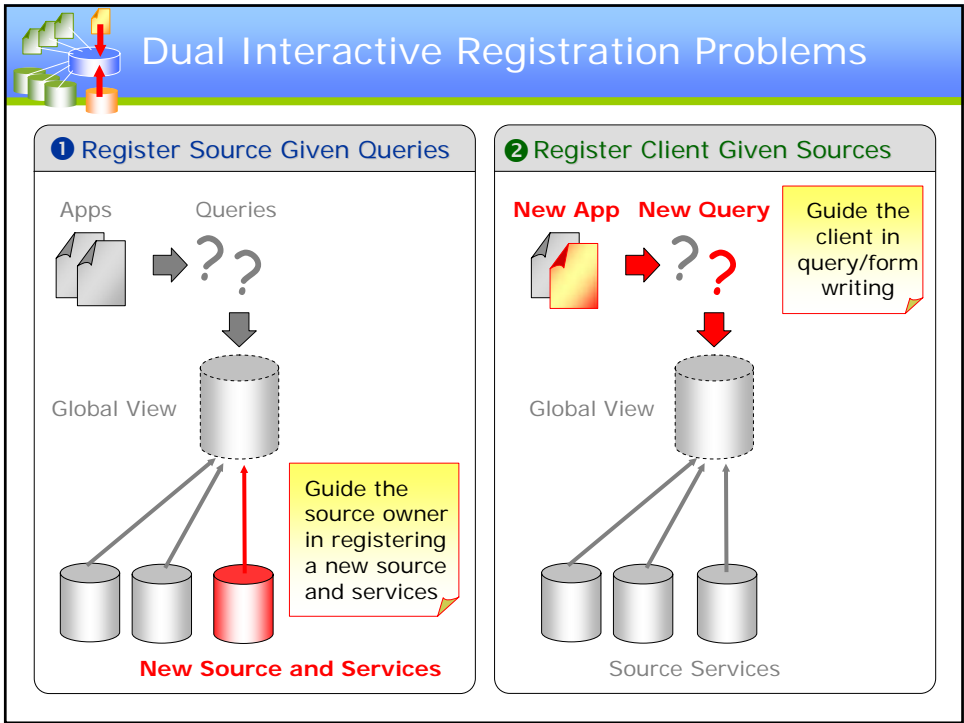
Result

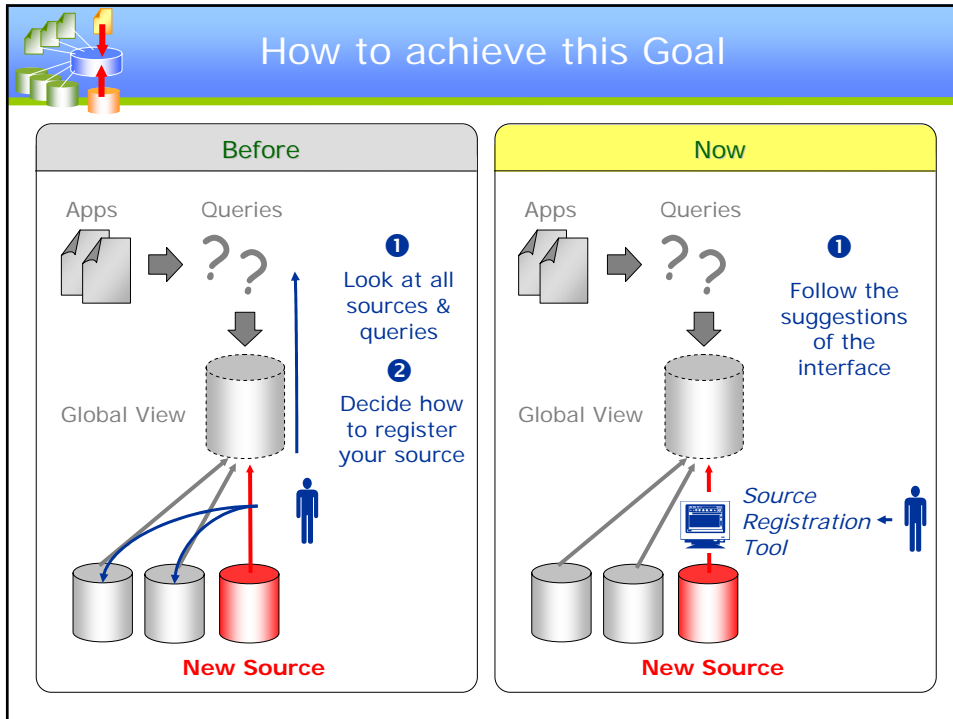
```

<PURCHASE_ORDER>
<PURCHASE_ORDER>
<SER>1</SER>
<PON>POM1234</PON>
<VERSION>ABC</VERSION>
<CLECID>/>
<STATE>5</STATE>
<SRKEY>1</SRKEY>
<LSO>
<SRKEY>1</SRKEY>
<REPID>/>
<TERMD>/>
<DB_NAME>/>
<PON>POM1234</PON>
<VERSION>ABC</VERSION>
<IDO>2002-05-06 12:00:00.0</IDO>
<ACT>/>
<STATUS>LSC</STATUS>
<LAST_ACT>LAST_1</LAST_ACT>
<CCHA>AAAA</CCHA>
<ACNA>AAAA</ACNA>
<LSR_NO>LSR00001</LSR_NO>
<SERVICE_ORDER>
<LSR_NO>LSR00001</LSR_NO>
<SO_NO>SPS21229</SO_NO>
<SO_DATE>2002-10-31</SO_DATE>
<SO_STATUS>PK</SO_STATUS>
<SO_TYPE>SO_TYPE</SO_TYPE>
<CONFIRMATION>

```







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)

15

The Problem

Client Queries
? Q ?

Mediator
(Global DB)

Sources (Actual Local DBs)

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

16

The Problem

Client Queries

?

Q

↓

Mediator
(Global DB)

Sources *(Actual Local DBs)*

- 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

17

Relational Schemas: Local and Global

?

↓

- Relational Schemas
- Visual Representation

S_1

- make
 - Carmake
 - Origin
 - Sales

Source 1
Business Magazine

S_2

- auto
 - Id
 - Model
 - Carmake
- detail
 - Id
 - Engine
 - Baseprice

Source 2
Car Magazine

G

- car
 - Model
 - Carmake
 - Doors
 - Baseprice
- brand
 - Carmake
 - Origin

Global
Car Portal

relations

attributes

18

Source Registration using GLAV Mappings

- **Source Registration:**
Correspondence between a source schema and the global schema
= Set of Mapping Constraints of the form

$$(U \subseteq V)$$

CQ=
over **source**
schema

CQ=
over **global**
schema

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

19

Target Constraints

- Constraints on the global schema
= Set of Constraints of the form

$$(U \subseteq V)$$

CQ=
over **global**
schema

CQ=
over **global**
schema

- Also Expresses Dependencies (PKs, Ref Integrity, ...)

20

Visual Representation of Mappings (1)

• Visual Representation (IBM Clio)

S_1

- make
- ├─ Carmake
- ├─ Origin
- └─ Sales

make
C
O
S

G

- car
- ├─ Model
- ├─ Carmake
- ├─ Doors
- └─ Baseprice

- brand
- ├─ Carmake
- └─ Origin

brand
C
O

Business Magazine: Provides Carmake and Origin

$(U_1 \subseteq V_1)$

$U_1(C, O) :- \text{make}(C, O, S)$
 $V_1(C, O) :- \text{brand}(C, O)$

21

Visual Representation of Mappings (2)

• Visual Representation (IBM Clio)

S_2

- auto
- ├─ Id
- ├─ Model
- └─ Carmake

auto
I
M
C

- detail
- ├─ Id
- ├─ Engine
- └─ Baseprice

detail
I
E
B

G

- car
- ├─ Model
- ├─ Carmake
- ├─ Doors
- └─ Baseprice

car
M
C
?
B

- brand
- ├─ Carmake
- └─ Origin

Car Magazine: Provides Model, Carmake and Baseprice

22

Example of Target Constraint

- (Model, Carmake) is a PK of car

\mathcal{G}

- car
 - Model ⌞
 - Carmake ⌞
 - Doors
 - Baseprice
- brand
 - Carmake
 - Origin

$(U_1 \subseteq V_1)$

$U_1(M, C, D_1, B_1, D_2, B_2) :-$	car(M, C, D ₁ , B ₁),
	car(M, C, D ₂ , B ₂)
$V_1(M, C, D, B, D, B) :-$	car(M, C, D, B)

23

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

24

Source Instance's Contribution

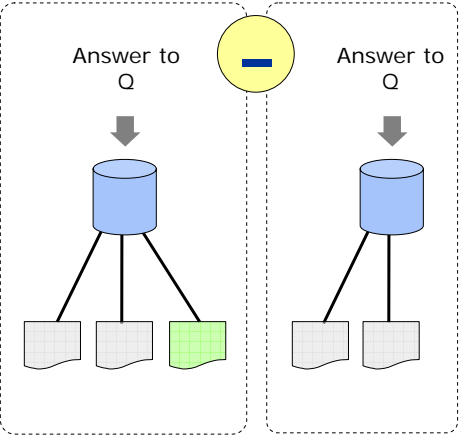
- For given instances of the sources

Contribution to Q of Source Instance



=

The tuples in answer of Q not provided by the other sources




25

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



More contribution

Less contribution

26

Self Sufficient Registration: Example

Example

Baseprices of Models

car
Model
Carmake
Doors
Baseprice

- car
- Model
- Carmake
- Doors
- Baseprice
- brand
- Carmake
- Origin

car
M3
BMW
?
45K

Green Registration is Self Sufficient

27

Self Sufficient Registration: Definition

① Self Sufficient

\exists Source instance

s.t.

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

Answer to Q

—

Answer to Q

$\neq \emptyset$

28

New Complementary Registration: Example

Example

Baseprices of Models by German manufacturers

car
Model
Carmake
Doors
Baseprice

brand
Carmake
Origin = 'Germany'

\mathcal{Q}

- car
 - Model
 - Carmake
 - Doors
 - Baseprice
- brand
 - Carmake
 - Origin

car
M3
BMW
?
45K

brand
BMW
Germany

Green Registration is **Now Complementary**

29

New Complementary Registration: Definition

2 Now Complementary

Not Self Sufficient

&

\exists Source instances

s. t.

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

Answer to \mathcal{Q}

—

Answer to \mathcal{Q}

$\neq \emptyset$

30

Later Complementary Registration: Example

Example

Baseprices of Models by German manufacturers

car
Model
Carmake
Doors
Baseprice

brand
Carmake
Origin = 'Germany'

\mathcal{G}

- car
- Model
- Carmake
- Doors
- Baseprice
- brand
- Carmake
- Origin

car
M3
BMW
?
45K

brand
BMW
Germany

31

Green Registration is Later Complementary

Later Complementary Registration: Definition

3 Later Complementary

Not Self Sufficient & Not Now Complementary

&

∃ Potential future sources

& Source instances

s. t.

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

Answer to Q

−

Answer to Q

≠ ∅

32

Unusable Registration: Example

Example

Origin of Carmakes

brand
Carmake
Origin

Green Registration is
Unusable

33

Unusable Registration: Definition

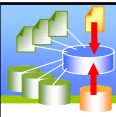
4 Unusable

Not Self Sufficient &
Not Now Complementary &
Not Later Complementary

↔

The source has a empty
contribution regardless of
what sources enter the
system

34

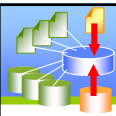


Decidability Results

Overview: What is decidable

		Target constraints		
		None	Primary keys	Primary keys + Referential Integrity Constraints
D e g r e e s	Self Sufficient	Yes	Yes	No
	Now complementary	Yes	Yes	No
	Later complementary	Yes	Yes	?
	Unusable	Yes	Yes	?

37



Issues

Unique client query Vs Multiple client queries 1

- Contribute to:
- all queries?
 - one query?
 - specific queries?
 - some queries based on some ranking?

Data independence Vs Data dependence 2

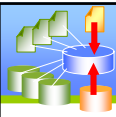
e.g. M_1 : cars, refPrices
 M_2 : reviews
 Q : cars JOIN reviews JOIN refPrices

DB_1 : cars, refPrices (Audis)

DB_2 : reviews (Hondas)

(M_2, Q) now-complementary

but Certain Answers for Instances $DB_1, DB_2 = \emptyset$



Putting it all together

Architecture

Query Answering / Mappings / Schemas

Contribution

4 categories:

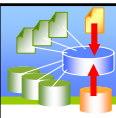
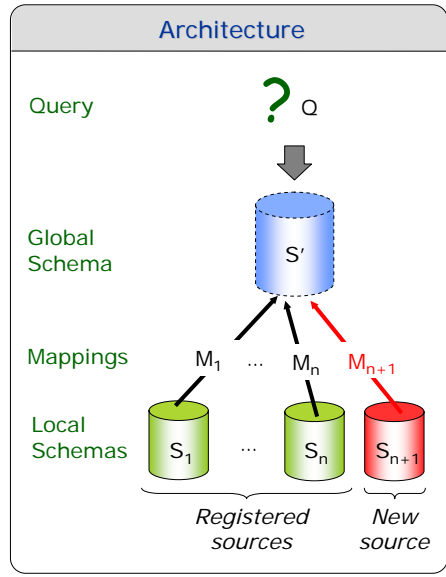
Self Sufficient / Now Complementary /
Later Complementary / Unusable

Goal

Guide the source owner visually through the *registration* of the *source*

so as to *raise contribution* to the *answer of the queries*

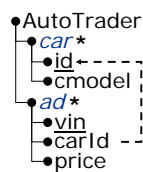
while *exposing the minimum info possible* and/or minimizing effort



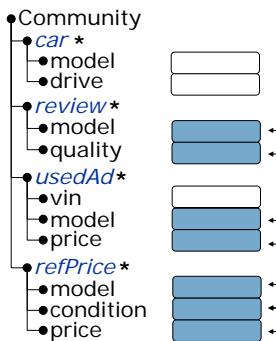
Example 1

Without primary keys in the target

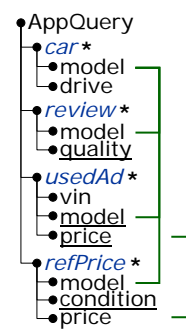
Local Schemas



Global Schema

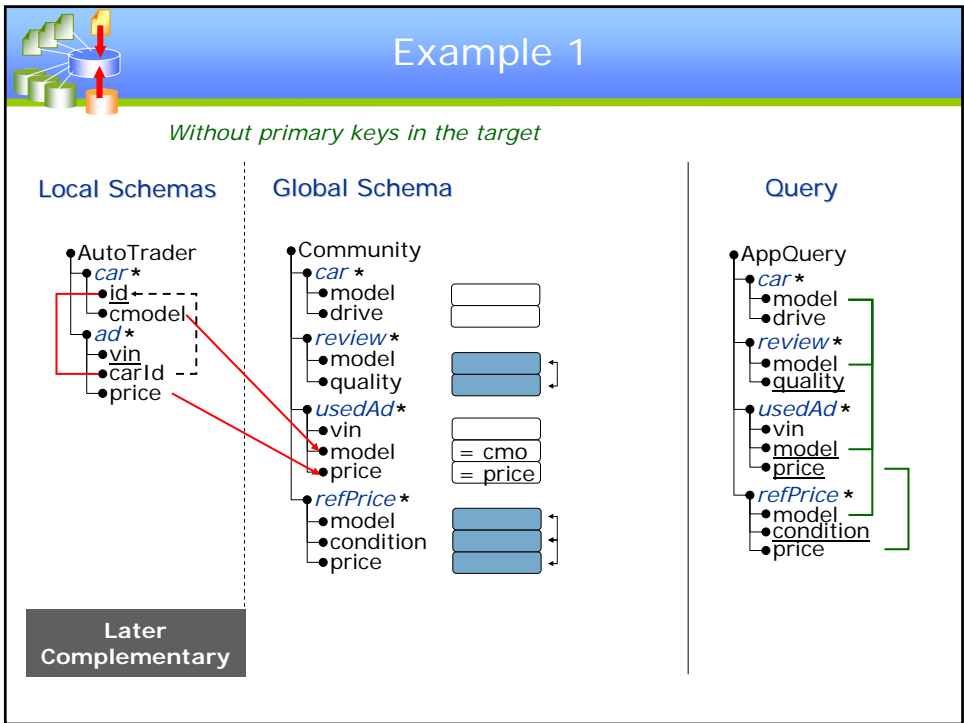
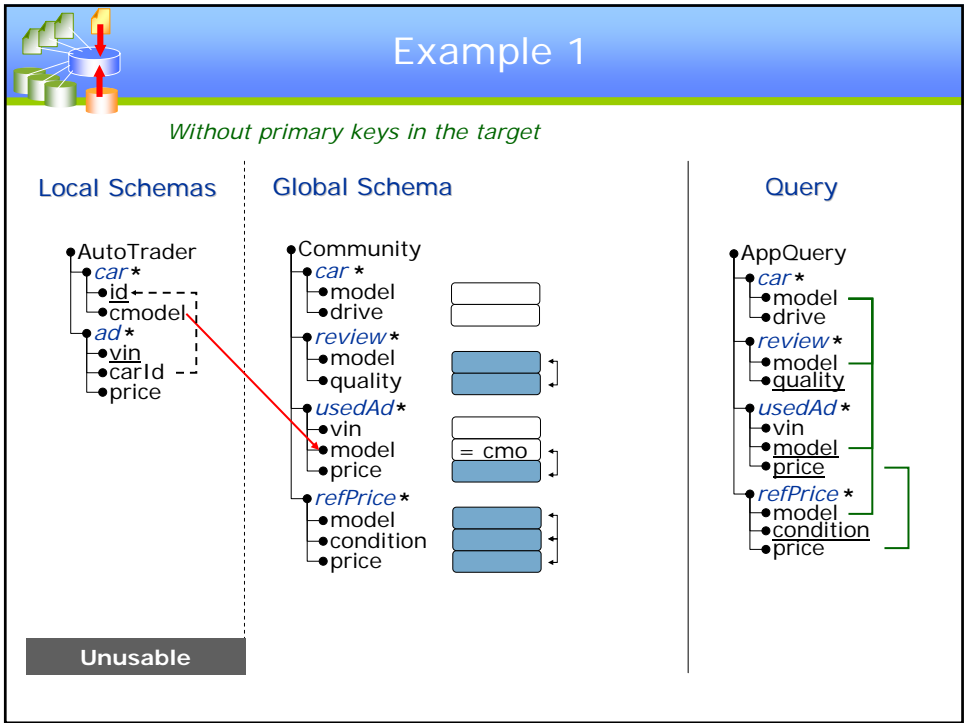


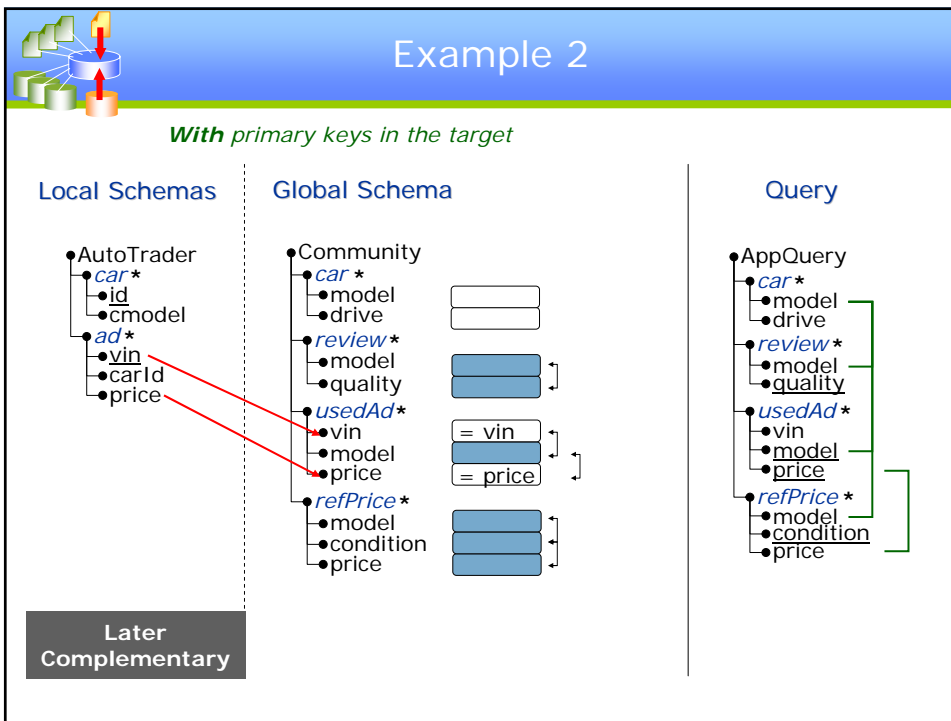
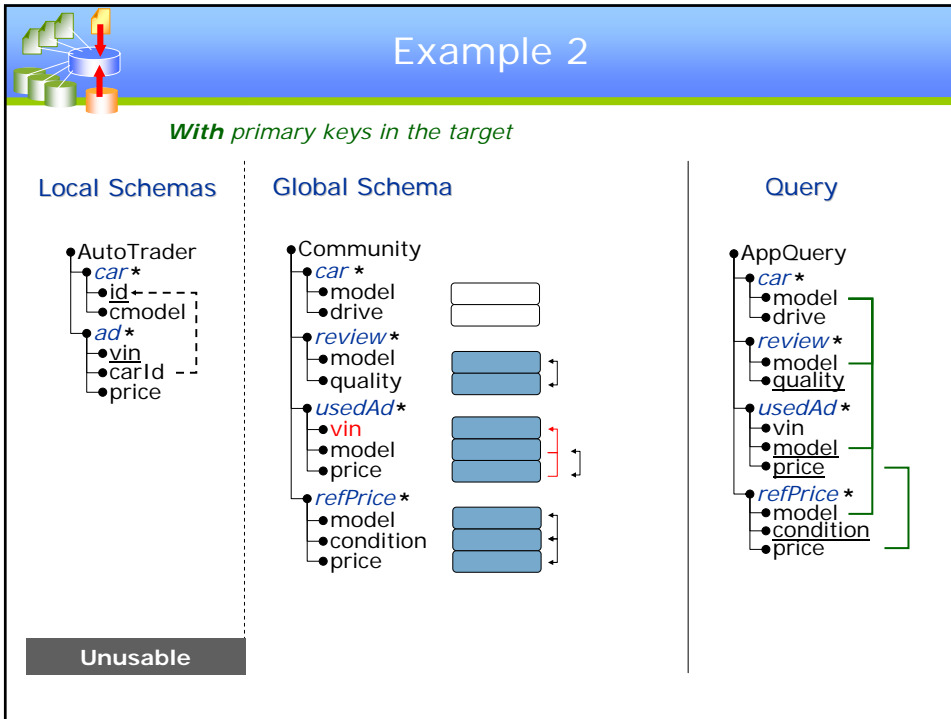
Query

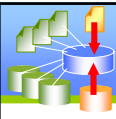


Unusable

BLUE: Map at least one of the groups







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

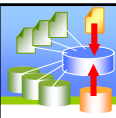
...all its attributes asked by the query

In presence of primary keys

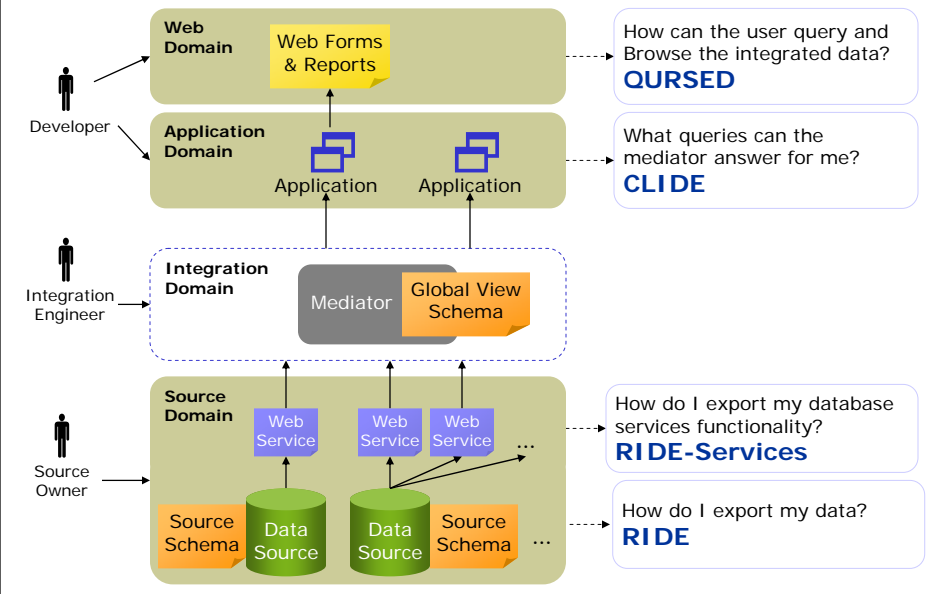
...its primary key and one of its attributes asked by the query

The number of choices increases in presence of primary keys

2 Foreign keys on the target affect the suggestions



Large-Scale Data Integration Systems



Challenge: Heterogeneity

Sensors Form

General

Part Number:

Manufacturer: (Options: No preference, Balluff, Baumer, Turck)

Min Distance (mm):

Temperature (°C): to

Mechanical

Protection Rating 1: (Options: No preference, IP67, NEMA12, NEMA13, NEMA4)

Protection Rating 2: (Options: No preference, IP67, NEMA12, NEMA13, NEMA4)

Body Type: (Options: No preference, Cylindrical, Rectangular)

Max Diameter (mm):

Barrel Style: (Options: No preference, Smooth, Grooved)

Electrical

Min Frequency (Hz):

Switching Mode: (Options: No preference, NEMA4, NEMA4x, NEMA6)

Number Of Wires:

Supply Voltage (V): to

Output Type: (Options: No preference, IP67, NEMA12, NEMA13, NEMA4, NEMA4x)

Built-in Field Bus: (Options: No preference, IP67, NEMA12, NEMA13, NEMA4, NEMA4x)

Field Bus Protocol: (Options: No preference, IP67, NEMA12, NEMA13, NEMA4, NEMA4x)

Query Results

Order By: Sensing Distance (DESC) | Manufacturer (ASC)

Sensors Results

Product Image	Manufacturer	Part Number	Protection Rating	Sensing Distance mm	Body Style
	Balluff	S16-211-S21-EL-W	IP67 NEMA12 NEMA13 NEMA4	5.0	Rectangular Height mm: 14, Width mm: 9
	Balluff	S16-215-S5-EL-W	NEMA12 NEMA13 NEMA4 NEMA4x	5.0	Cylindrical Diameter mm: 15, Barrel Style: Smooth
	Balluff	S17-219-US-EL	NEMA4 NEMA4x	10.0	Rectangular Height mm: 10, Width mm: 35
	Balluff	S17-219-US-EL-S	IP67 NEMA12 NEMA13 NEMA4 NEMA4x	50.0	Rectangular Height mm: 15, Width mm: 10
	Balluff	S17-219-US-EL-S21	IP67 NEMA12 NEMA13	50.0	Cylindrical Diameter mm: 19, Barrel Style: Smooth
	Balluff	S17-219-US-EL-S21	IP67 NEMA12 NEMA13	50.0	Cylindrical Diameter mm: 27, Barrel Style: Smooth
	Balluff	S17-219-US-EL-S21	IP67 NEMA12 NEMA13	50.0	Cylindrical Diameter mm: 40, Barrel Style: Smooth

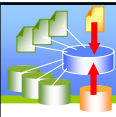
Cylindrical Sensors

- Diameter
- Barrel Style

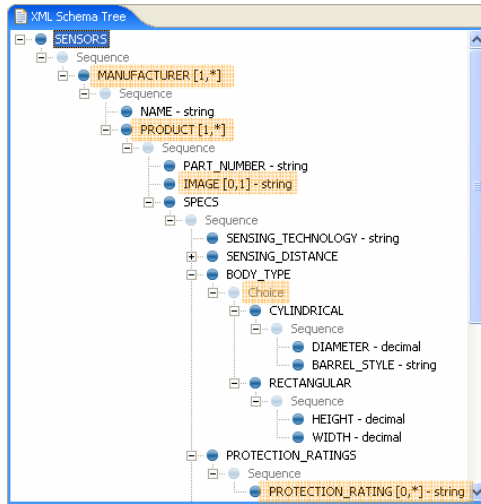
Query Forms and Reports

Current Practices

- Fixed Parameterized SQL Statements
- Custom Code Required for Combinations of Conditions
- Very Tricky for Semistructured Data



Capturing Heterogeneity



XML Schema

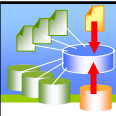
- Captures heterogeneity

XML

- Just a language
- Not a program

XQuery Standard (W3C)

- XML Query Language
- Behind the scenes



XQuery: Expressive But Verbose

```

<html>
<body>
<table>{
  FOR $doc IN accessSource('XMLDB'), $$ IN $doc/sensors,
  $SMAN IN $$/manufacturer, $SPROD IN $SMAN/product,
  $SPEC IN $SPROD/specs, $PROTS IN $$SPEC/protection_ratings,
  $PROT1 IN $PROTS/protection_rating, $PART IN
  $SPROD/part_number,
  $SDIST IN $$SPEC/sensing_distance, $BODY IN
  $$SPEC/body_type,
  $SN_BODY IN $BODY/name(), $PROT IN
  $PROTS/protection_rating,
  $SNAME IN $MAN/name
  WHERE
  $PROT1 = "NEMA3"
  AND ((SOME $CYL IN $BODY/cylindrical,
  $DIA IN $CYL/diameter,
  $SBAR IN $CYL/barrel_style
  SATISFIES
  $DIA <= 20 AND $DIA <= 40)
  OR
  (SOME $SREC IN $BODY/rectangular,
  $SHEI IN $SREC/height,
  $SWID IN $SREC/width
  SATISFIES
  $SHEI <= 20 AND $SWID <= 40))
  ORDER BY $SNAME DESCENDING, $SDIST
  RETURN
  GROUPBY $PROD, $SNAME, $SDIST AS
  <tr>{
    <td>{
      FOR $IMG IN $SPROD/image
      RETURN
      GROUPBY $IMG AS <img srv="{ $IMG }" />
    }</td>,
    <td>{
      IF ($SNAME = "Turck") THEN <img*="turck.gif"></img>
      <img*="balluff.gif"></img>
      IF ($SNAME = "Baumer") THEN
      <img*="baumer.gif"></img>
    }</td>,
  }</tr>
  <td>{
    <table>{
      <tr>{
        GROUPBY $SN_BODY AS <td>{$SN_BODY}</td>
      }</tr>,
      <tr>{
        <td>{
          FOR $CYL IN $BODY/cylindrical,
          $DIA IN $CYL/diameter,
          $SBAR IN $CYL/barrel_style
          WHERE
          $DIA <= 20 AND $DIA <= 40
          RETURN
          GROUPBY $CYL AS
          <table>{
            <tr>{
              GROUPBY $DIA AS
              <td>{$DIA}</td>,
              GROUPBY $SBAR AS
              <td>{$SBAR}</td>,
            }</tr>
          }</table>
        }</td>,
        <td>{
          FOR $SREC IN $BODY/rectangular,
          $SHEI IN $SREC/height,
          $SWID IN $SREC/width
          WHERE
          $SHEI <= 20 AND $SWID <= 40
          RETURN
          GROUPBY $SREC AS
          <table>{
            <tr>{
              GROUPBY $SHEI AS
              <td>{$SHEI}</td>,
              GROUPBY $SWID AS
              <td>{$SWID}</td>,
            }</tr>
          }</table>
        }</td>
      }</tr>
    }</table>
  }</td>
}
}
}

```


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

Meaningful Queries

Mechanical	
Body Type	Cylindrical
Max Diameter (mm)	
Barrel Style	Smooth

<f1, #BODY = 'Cylindrical'>

Mechanical	
Body Type	Rectangular
Max Height (mm)	
Max Width (mm)	

<f2, #BODY = 'Rectangular'>

OURSED Editor: No Coding

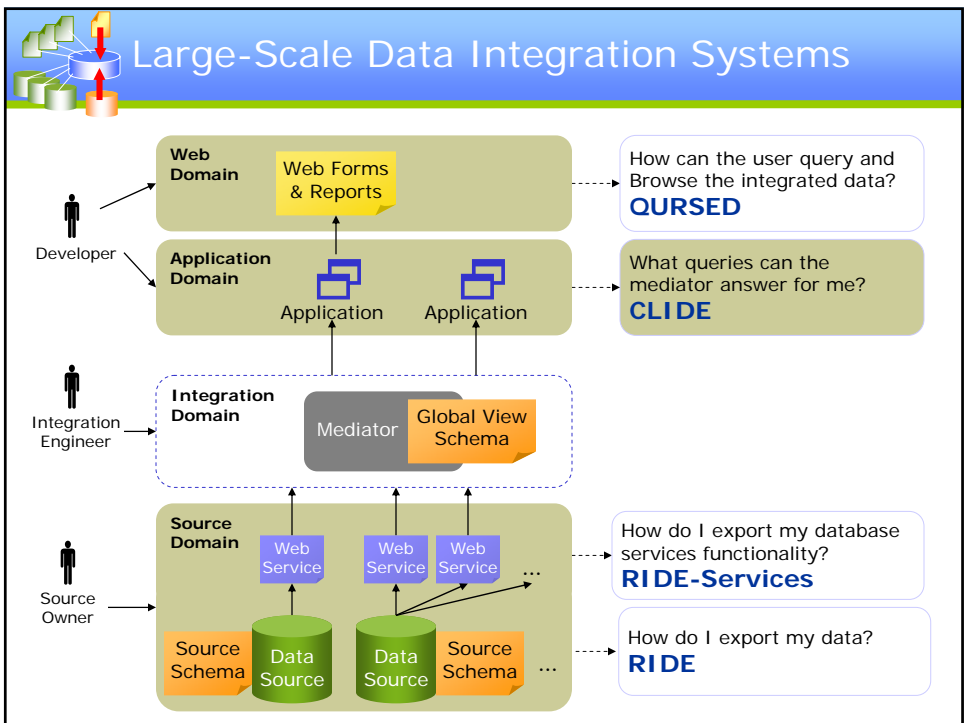
CURSED Editor: Automatic Reports

Building Reports

Content Mappings

Manufacturer	Part Number	Product Image	Body Style	Protection Rating
Balluff	S16-211-S21-EL-W		Rectangular	IP67
			Height mm	Width mm
	S16-211-SS-EL-W		Cylindrical	NEMA12
			Diameter mm	Barrel Style
	S16-215-S21-EL-W		Rectangular	NEMA13
			Height mm	Width mm
S16-215-SS-EL-W		Rectangular	NEMA4	
		Height mm	Width mm	

Report follows schema nesting



Running Example

Parameterized Views

Dell

Schema
Computers(cid, cpu, ram, price)
NetCards(cid, rate, standard, interface)

Views

V1 ComByCpu(cpu) → (Computer)*
SELECT DISTINCT Com1.*
FROM Computers Com1
WHERE Com1.cpu=*cpu*

V2 ComNetByCpuRate(cpu, rate) → (Computer, NetCard)*
SELECT DISTINCT Com1.*, Net1.*
FROM Computers Com1, Network Net1
WHERE Com1.cid=Net1.cid
AND Com1.cpu=*cpu*
AND Net1.rate=*rate*

Cisco

Schema
Routers(rate, standard, price, type)

Views

V3 RouByTypeW() → (Router)*
SELECT DISTINCT Rou1.*
FROM Routers Rou1
WHERE Rou1.type='Wired'

V4 RouByTypeWLO → (Router)*
SELECT DISTINCT Rou1.*
FROM Routers Rou1
WHERE Rou1.type='Wireless'

Computers for a given *cpu*

Wired Routers

Wireless Routers

Computers & NetCards for a given *cpu* & *rate*

Running Example

Integrated Schema

The diagram illustrates the Mediator pattern. An **Application** and a **Developer** interact with a **Mediator**. The **Mediator** is associated with an **Integrated Schema**. Below the mediator are four **Views** (V1, V2, V3, V4). V1 and V2 are associated with the **Dell** data source, while V3 and V4 are associated with the **Cisco** data source.

Computers
cid
cpu
ram
price

NetCards
cid
rate
standard
interface

Routers
rate
standard
price
type

- 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

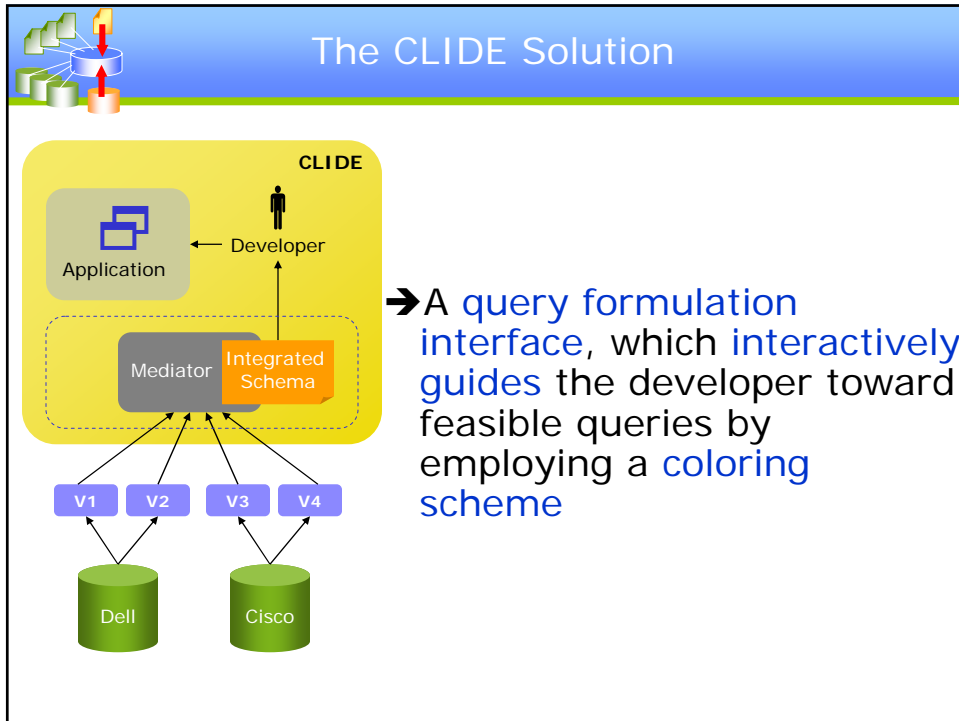
Developer formulates an application query

→ 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)



QBE-Like Interfaces

Microsoft SQL-Server

Add Table

Tables | Views

- Computers
- NetCards
- Routers

Add

Data in Table 'Computers' in 'Computers' o...

Column	Table	Output	Criteria
cpu	Computers		= 'P4'
ram	Computers	✓	
price	Computers	✓	
rate	NetCards		= '54Mbps'
interface	NetCards	✓	

```

SELECT Computers.ram, Computers.price, NetCards.interface
FROM   Computers INNER JOIN
       NetCards ON Computers.cid = NetCards.cid
WHERE  (Computers.cpu = 'P4') AND (NetCards.rate = '54Mbps')

```

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

CLIDE Interface

Table List	
Computers	
NetCards	
Routers	

Com1	
cid	
cpu	=P4'
ram	
price	

Feasible

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

Blue → required choice of action

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

Example Interaction

Snapshot 3

CLIDE Interface

Table List	
Computers	
NetCards	
Routers	

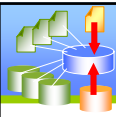
Com1		Net1	
cid		cid	
cpu	=P4'	rate	
ram		standard	
price		interface	

Infeasible

```
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

CLIDE Interface

Table List

Computers

NetCards

Routers

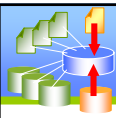
Com1

cid		cid
cpu	=P4*	rate
ram		standard
price		interface

Net1

Infeasible

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



Example Interaction

Snapshot 5

CLIDE Interface

Table List

Computers

NetCards

Routers

Com1

cid		cid
cpu	=P4*	rate
ram		standard
price		price

Net1

rate
standard
price
type

Rou1

Infeasible

```
SELECT DISTINCT Com1.ram, Com1.price
FROM Computers Com1, NetCards Net1, Routers Rou1
WHERE Com1.cpu='P4' AND Com1.cid=Net1.cid
```

- * → any other constant
- Red → prohibited action
 - Does not appear in any feasible query
 - Lead to "Dead End" state

Example Interaction

Snapshot 6

CLIDE Interface

Table List

Table	Attributes	Value
Computers	cid	
NetCards	cpu	=P4'
Routers	rate	
	standard	
	price	
	type	=Wireless'

SELECT DISTINCT Com1.ram, Com1.price, Net1.rate, Net1.interface, Rou1.price
 FROM Computers Com1, NetCards Net1, Routers Rou1
 WHERE Com1.cpu='P4' AND Com1.cid=Net1.cid
 AND Net1.rate=Rou1.rate AND Rou1.type='Wireless'

ram	price	rate	interface	price
512	400	10	USB	50
1024	550	54	USB	120

A

RouByTypeW
LO

V4

B

Routers.*			
10	.11b	512	Wireless
54	.11g	1024	Wireless

D

ComNetByCpuRate('P4', rate)

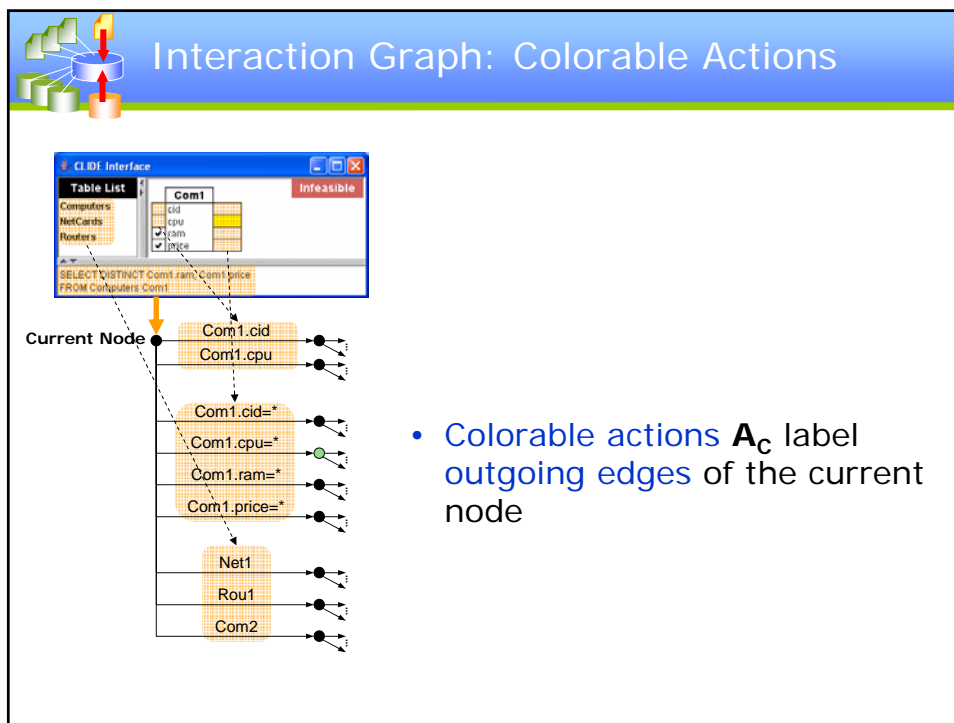
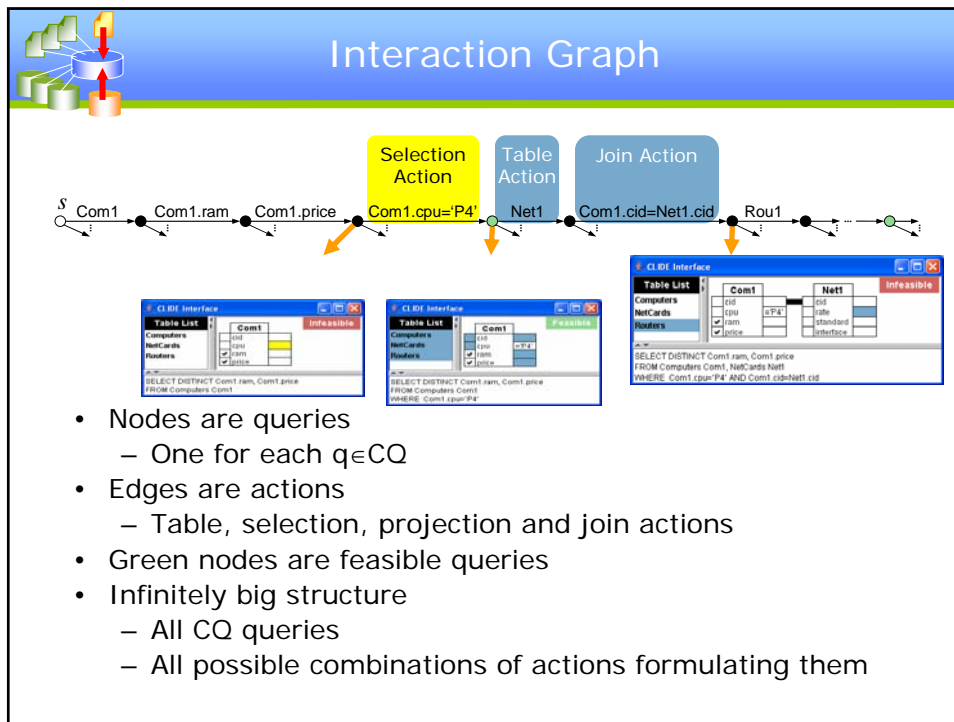
E

Computers.*		NetCards.*	
A123	P4	512	400
B123	P4	1024	550
A123	.11b	50	
B123	.11g	120	

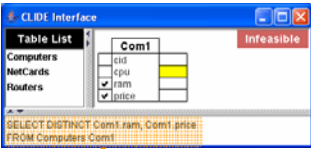
Mediator

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



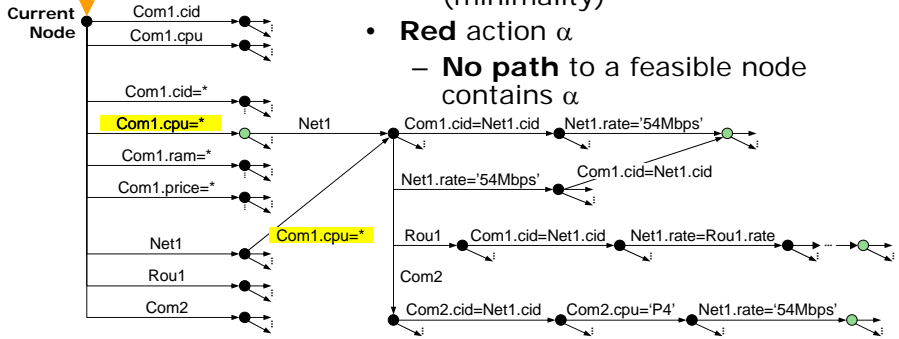
Interaction Graph: Colors



Current Node

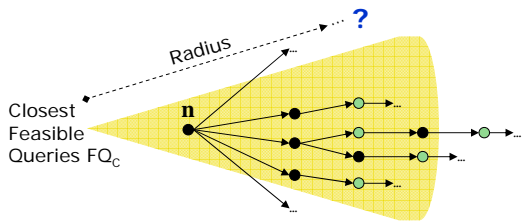
- Com1.cid
- Com1.cpu
- Com1.cid=*
- Com1.cpu=***
- Com1.ram=*
- Com1.price=*
- Net1
- Rou1
- Com2

- **Yellow** action α
 - **Every path** from current node n to a feasible node contains α
- **Blue** action α
 - At least one feasible query cannot be formulated unless this action is performed (minimality)
- **Red** action α
 - **No path** to a feasible node contains α



Color Determined By a Finite Set of Feasible Queries

Challenge: Infinitely Many Feasible Queries



Solution: Closest Feasible Queries FQ_C

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

Challenge: How far can the Closest Feasible Queries FQ_C be?

Solution: Based on Maximally Contained Queries FQ_{MC}

