Semantic Web

9. Ontology-based Data Access

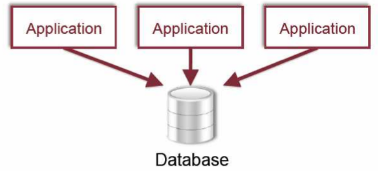
PD Dr. Matthias Thimm

thimm@uni-koblenz.de

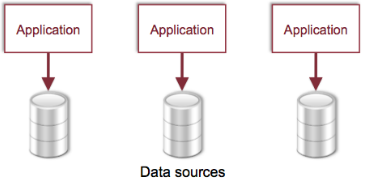
Institute for Web Science and Technologies (WeST)
University of Koblenz-Landau

Ontology-based Data Access: Overview 1/3

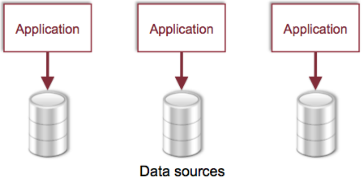
Ideal information architecture with database management systems (DBMS):



Actual information architecture:



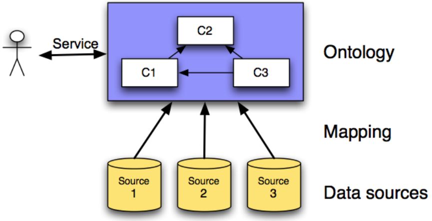
Ontology-based Data Access: Overview 2/3



- Distributed, redundant, application-dependent, and mutually incoherent data
- There is the need for a coherent, conceptual, unified view of data

$\rightarrow$ Data integration, in particular *ontology-based data integration and access*

Ontology-based Data Access: Overview 3/3



Based on three components:

- **Ontology**, a declarative, logic-based specification of the domain of interest, used as a unified, conceptual view for clients
- **Data sources**, representing external, independent, heterogeneous storage structures
- **Mappings**, used to semantical link data at the sources to the ontologies

Outline

Outline

Schemas and Instances

Schemas:

- A *database schema* $\mathcal{S}$ is a set of relations (=tables)

$$\mathcal{S} = \{R_1(X^1_1, \ldots, X^1_{n_1}), \ldots, R_m(X^m_1, \ldots, X^m_{n_m})\}$$

- $R_i(X^i_1, \ldots, X^i_{n_i})$ means that relation (=table) R_i has arity n_i and parameters (=columns, attributes) $X^i_1, \ldots, X^i_{n_i}$
- We also write $R_i.X^i_j$ to refer to the attribute directly.

Instances:

- An *instance* $\mathcal{I}$ of $\mathcal{S}$ is a set

$$\mathcal{I} = \{\mathcal{I}_{R_1}, \ldots, \mathcal{I}_{R_m}\}$$

with $\mathcal{I}_{R_i} \subseteq V^{n_i}$ where V is some set of values (here: only strings).

Example

Student	id	name
	4	Carl
	7	Anna

Course	id	name
	2	AlgoDat
	5	SemWeb

Grade	sid	cid	grade
	4	2	2.0
	4	5	1.3

$$\mathcal{S}_{Uni} = \{\text{Student}(\mathit{id}, \mathit{name}), \text{Course}(\mathit{id}, \mathit{name}), \text{Grade}(\mathit{sid}, \mathit{cid}, \mathit{grade})\}$$

$$\begin{aligned}
\mathcal{I}_{Uni} &= \{\mathcal{I}_{\text{Student}}, \mathcal{I}_{\text{Course}}, \mathcal{I}_{\text{Grade}}\} \\
\mathcal{I}_{\text{Student}} &= \{(4, \text{Carl}), (7, \text{Anna})\} \\
\mathcal{I}_{\text{Course}} &= \{(2, \text{AlgoDat}), (5, \text{SemWeb}\} \\
\mathcal{I}_{\text{Grade}} &= \{(4, 2, 2.0), (4, 5, 1.3)\}
\end{aligned}$$

Integrity Constraints

Functional dependencies;

- A *functional dependency* r is a rule $r : X_1, \ldots, X_s \to Y_1, \ldots, Y_t$ with attributes $X_1, \ldots, X_s, Y_1, \ldots, Y_t$
- Example: r_1 : Student.id $\to$ Student.name
- $\mathcal{I}$ satisfies the functional dependency r_1 if there are no two records in Student with the same id but different names

Primary Key:

- An attribute $R.X$ is a *primary key* of R (in instance $\mathcal{I}$) if $R.X \to R.Y_1, \ldots, R.Y_n$ (where $R.Y_1, \ldots, R.Y_n$ are all attributes of R without $R.X$)
- We then write $\mathit{primary}(R.X)$

Foreign Key:

- An attribute $R.X$ is a *foreign key* for $R'.Y$ (in instance $\mathcal{I}$): if Z appears in $R'.Y$ then Z also appears in $R.Z$
- We then write $\mathit{foreign}(R.X, R'.Y)$

Example

Student	id	name
	4	Carl
	7	Anna

Course	id	name
	2	AlgoDat
	5	SemWeb

Grade	sid	cid	grade
	4	2	2.0
	4	5	1.3

- Student.id is a primary key of Student
- Grade.sid is a foreign key for Student.id
- Grade.sid, Grade.cid $\rightarrow$ Grade.grade

Queries and SQL

Let $\mathcal{S}$ be a database schema, $\mathcal{C}$ some integrity constraints (dependency statements, key definitions), and $\mathcal{I}$ an instance of $\mathcal{S}$ satisfying $\mathcal{C}$.

- A (simple) *FOL*-query for $\mathcal{I}$ is an existentially quantified conjunctive FOL-formula with variables:

$$\exists x_1, \ldots, x_n : R(\ldots, x_i, \ldots) \wedge \ldots \wedge R(\ldots, x_j, \ldots)$$

- We usually use SQL for formalizing FOL-queries

Example

```
SELECT s.name AS x, c.name AS y
FROM Student s, Course c, Grade g
WHERE s.id = g.sid AND g.cid = c.id
```

Outline

- A description logic knowledge base $\mathcal{K}$ (=ontology) is a tuple $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ with
  - a TBox $\mathcal{T}$ with assertions $C_1 \sqcap C_2 \sqsubseteq C_3, C_1 \sqsubseteq \forall r_1.C_4, \ldots$
  - an ABox $\mathcal{A}$ with assertions $a : C_1, b : C_2, (a, b) : r_{1,\ldots}$

Example

Let $\mathcal{K}_{Uni} = (\mathcal{T}_{Uni}, \mathcal{A}_{Uni})$ with

$$\mathcal{T}_{Uni} = \{\text{Student} \sqsubseteq \text{Person} \sqcap \exists \mathit{enrolledIn}.\top\}$$

$$\mathcal{A}_{Uni} = \{\mathit{Carl} : \mathit{Student}, \mathit{Dave} : \mathit{Student}, \mathit{SemWeb} : \mathit{Course}, (\mathit{Dave}, \mathit{SemWeb}) : \mathit{enrolledIn}\}$$

Outline

The Impedance Mismatch 1/2

In the following, we assume the database schema not only contains the actual schema but also integrity constraints such as dependency statements and key declarations.

Observations:

- Database schema $\approx$ TBox
- Database instance $\approx$ ABox

The Impedance Mismatch 2/2

The impedance mismatch:

- Database paradigm: data is stored in records (=rows of tables), central notion is the relation and truth assignments
- DL paradigm (also valid for object-oriented programming): the central notion is the object (or instance or individual) that belongs to concepts and satisfies relations
- Mismatch: information on a object in a database can be distributed among several tables

Goal of OBDA: Bridge the impedance mismatch and use DL as a query language to access heterogeneous data sources that are accessible through RDMS

Architecture

A data integration system *dis* is a tuple *dis* $= (\mathcal{G}, \mathcal{S}, \mathcal{M})$ with

- $\mathcal{G}$ (global schema) is a set of terminological axioms (=TBox)
- $\mathcal{S}$ (source schema) is a database schema (or a set of database schemas)
- $\mathcal{M}$ is a set of mappings between $\mathcal{G}$ and $\mathcal{S}$ (basically, a mapping between SQL queries and DL queries)

How can a mapping between $\mathcal{G}$ and $\mathcal{S}$ be specified?

- Define the data of $\mathcal{S}$ in terms of the conceptualization in $\mathcal{G}$ (local-as-view approach, LAV)
- Define the data of $\mathcal{G}$ in terms of the schema information $\mathcal{S}$ (global-as-view approach, GAV)

Semantics

Let $dis = (\mathcal{G}, \mathcal{S}, \mathcal{M})$ be a data integration system and $\mathcal{I}$ a database instance wrt. $\mathcal{S}$.

- The *semantics* of *dis* wrt. $\mathcal{I}$ are given as a set of description logic interpretations $I = (\Delta^I, \cdot^I)$
- An interpretation I is a model of *dis* wrt. $\mathcal{I}$ if it satisfies $\mathcal{G}$ (the TBox) and the actual data $\mathcal{I}$ by viewing it through the mapping $\mathcal{M}$
- Formally

$$sem^{\mathcal{I}}(dis) = \{I \mid I \models \mathcal{G} \wedge I \models_{\mathcal{M}} \mathcal{I}\}$$

- The definition of $\models_{\mathcal{M}}$ depends on the nature of $\mathcal{M}$

Example

Global schema (TBox):

$$\mathcal{G}: \quad Movie \sqsubseteq \exists hasTitle.\top \sqcap \exists createdInYear.\top \sqcap \exists hasDirector.Director$$
$$EuropeanDirector \sqsubseteq Director$$
$$Review \sqsubseteq \exists about.Movie \sqcap \exists hasComment.\top$$

Two data sources $\mathcal{S} = \mathcal{S}_1\mathcal{S}_2$:

$\mathcal{S}_1: \quad \{R1(title, year, director)\}$ since 1960 but only European directors

$\mathcal{S}_2: \quad \{R2(title, critique)\}$ since 1990

Query: Give me titles of all movies in 1998 which have at least one review

- All instances satisfying the concept $Movie \sqcap \exists createdInYear.\{1998\} \sqcap \exists about^{-}.\top$

GAV mappings

Elements in the global schema $\mathcal{G}$ can be considered as views over the source schema $\mathcal{S}$ (global as view).

Definition

A GAV mapping $\mathcal{M}$ is a set of assertions of the form

$$\phi_{\mathcal{S}}(\vec{x}) \rightarrow g(\vec{x})$$

one for every concept/role g appearing in $\mathcal{G}$ with $\phi_{\mathcal{S}}(\vec{x})$ being an SQL query over $\mathcal{S}$

The right-hand side may contain functional expressions that create new instances not explicitly defined on the left-hand side.

GAV mappings: Example

$$\mathcal{G}: \quad Movie \sqsubseteq \exists hasTitle.\top \sqcap \exists createdInYear.\top \sqcap \exists hasDirector.Director$$
$$EuropeanDirector \sqsubseteq Director$$
$$Review \sqsubseteq \exists about.Movie \sqcap \exists hasComment.\top$$

$\mathcal{S}_1: \quad \{R1(title, year, director)\}$ since 1960 but only European directors

$\mathcal{S}_2: \quad \{R2(title, critique)\}$ since 1990

GAV: for each concept in $\mathcal{G}$, $\mathcal{M}$ associates a view over $\mathcal{S}$:

```
SELECT R1.director AS x FROM R1                                 →  x: EuropeanDirector
SELECT R1.director AS x FROM R1                                 →  x: Director
SELECT R1.title AS x FROM R1 UNION SELECT R2.title AS x FROM R2 →  x: Movie
SELECT R1.title AS x, R1.year AS y FROM R1                      →  (x,y): createdInYear
SELECT R1.title AS x, R1.director as z FROM R1                  →  (x,z): hasDirector
SELECT R2.title AS x, R2.critique as z FROM R2                  →  f(x,z): Review
SELECT R2.title AS x, R2.critique as z FROM R2                  →  (f(x,z), x): about
SELECT R2.title AS x, R2.critique as z FROM R2                  →  (f(x,z), z): hasComment
```

GAV mappings: Semantics

Let $dis = (\mathcal{G}, \mathcal{S}, \mathcal{M})$ be a data integration system and $\mathcal{I}$ a data base instance wrt. $\mathcal{S}$.

Definition

Let $I = (\Delta^I, \cdot^I)$ be an interpretation of $\mathcal{G}$. Define $I \models_{\mathcal{M}} \mathcal{I}$ if there is an Abox $\mathcal{A}$ s.t. $\mathcal{M}(\mathcal{I}) \subseteq \mathcal{A}$ and $I \models \mathcal{A}$ where $\mathcal{M}(\mathcal{I})$ is the ABox resulting from applying $\mathcal{M}$ on $\mathcal{I}$.

$\rightarrow$ In this approach, query answering on a data integration system can be reduced to query answering on an ontology

Of course, this is not feasible as a query answering mechanism as it requires the *materialization* of the complete database.

LAV mappings

Elements in $\mathcal{S}$ can be considered as views over the global schema (local as views).

Definition

An LAV mapping $\mathcal{M}$ is a set of assertions of the form

$$s(\vec{x}) \rightarrow \phi_{\mathcal{G}}(\vec{x})$$

one for every relation s appearing in $\mathcal{S}$ with $\phi_{\mathcal{S}}(\vec{x})$ being a "DL query" over $\mathcal{G}$

LAV mappings: Example

$$\begin{aligned}\mathcal{G}: \quad & Movie \sqsubseteq \exists hasTitle.\top \sqcap \exists createdInYear.\top \sqcap \exists hasDirector.Director \\ & EuropeanDirector \sqsubseteq Director \\ & Review \sqsubseteq \exists about.Movie \sqcap \exists hasComment.\top\end{aligned}$$

$\mathcal{S}_1: \quad \{R1(title, year, director)\}$ since 1960 but only European directors

$\mathcal{S}_2: \quad \{R2(title, critique)\}$ since 1990

LAV: for each table in $\mathcal{S}$, $\mathcal{M}$ associates a view over $\mathcal{G}$:

```
SELECT R1.title AS x, R1. year AS y, R1.director AS z FROM R1
    → x: Movie ∧ (x,y): createdInYear ∧ x≥1960 ∧ z: EuropeanDirector
SELECT R2.title AS x, R2.critique as y FROM R2
    → ∃ z1, z2: x: Movie ∧ (x,z1): createdInYear ∧ z1 ≥1990 ∧ (z2,x): about ∧
          (z2,y): hasComment
```

LAV mappings: Semantics

Let $dis = (\mathcal{G}, \mathcal{S}, \mathcal{M})$ be a data integration system and $\mathcal{I}$ a data base instance wrt. $\mathcal{S}$.

Definition

Let $I = (\Delta^I, \cdot^I)$ be an interpretation of $\mathcal{G}$. Define $I \models_{\mathcal{M}} \mathcal{I}$ via $\mathcal{M}(\mathcal{I}) \subseteq eval_I(\mathcal{M})$ where $\mathcal{M}(\mathcal{I})$ is the ABox resulting from applying $\mathcal{M}$ on $\mathcal{I}$ and $eval_I(\mathcal{M})$ is the Abox resulting from evaluating the DL queries in $\mathcal{M}$ in I.

Comparison

GAV:

- Whenever a source changes or a new one is added, the global schema needs to be reconsidered
- Query processing can be implemented by materialization (or simple rewriting if $\mathcal{S}$ contains no complex integrity constraints)

LAV:

- High modularity and extensibility (if the global schema is well designed, when a source changes, only its definition is affected)
- Query processing needs reasoning (query rewriting)

Outline

The Paradigm of Ontology-based Data Access

- The idea of OBDA is that the user only needs to know the global schema (=TBox) $\mathcal{G}$ and asks queries wrt. to this schema
- The actual structure of the information system is hidden
- The user asks queries in the form of DL queries (e.g. SPARQL)

$\rightarrow$ how to handle the processing of the query internally?

- Simple approach: Canonical database
  - use $\mathcal{M}$ to create the ABox explicitly
  - only sound for GAV
  - big effort
- Query rewriting:
  - Incorporate the information from $\mathcal{G}$ into the query and translate it to SQL
  - Evaluate the resulting query directly on the database(s)
  - Depending on the DL this approach can be implemented efficiently

Queries

Let $dis = (\mathcal{G}, \mathcal{S}, \mathcal{M})$ be a data integration system, $\mathcal{I}$ a database instance wrt. $\mathcal{S}$.

- We only consider *unions of conjunctive* queries, i. e. queries of the form $q = q_1 \vee \ldots \vee q_l$ with

  $$q_i = x_1 : C_1 \wedge \ldots \wedge x_n : C_n \wedge (y_1, y_1') : R_1 \wedge \ldots \wedge (y_m, y_m') : R_m$$

  where
  - $C_1, \ldots, C_n, R_1, \ldots, R_m$ are concepts/roles from $\mathcal{G}$
  - $x_1 \ldots, x_n, y_1, \ldots, y_m, y_1', \ldots, y_m'$ are individuals or variables
- The evaluation $eval_{dis,\mathcal{I}}(q)$ of q on *dis* is the set of all mappings $eval_{dis,\mathcal{I}}(q) = \{\sigma_1, \ldots, \sigma_k\}$ where
  - $\sigma_i(q)$ replaces all variables in q with some indivuals
  - for all I with $I \models dis$ we have $I \models \sigma_i(q)$

Query Rewriting: Idea

Given a query q the general process of rewriting is as follows:

1. Pre-Processing:
   - Incorporate the information from $\mathcal{G}$ into q s.t. $\mathcal{G}$ is no longer needed
   - Example: If $q = x : C$ and $B \sqsubseteq C \in \mathcal{G}$ then rewrite $q' = x : C \vee x : B$
   - this step may be skipped for GAV
2. Unfolding
   - Use $\mathcal{M}$ to create SQL queries from the resulting queries (for LAV e. g. by using logic programming techniques)
   - Complex step
3. Evaluation
   - Evaluate the resulting queries on the database instances
4. Translation
   - Translate the result back into the DL formalism

Query Rewriting: Example for GAV 1/2

$$\mathcal{G}: \quad \mathit{Movie} \sqsubseteq \exists \mathit{hasTitle}.\top \sqcap \exists \mathit{createdInYear}.\top \sqcap \exists \mathit{hasDirector}.\mathit{Director}$$
$$\mathit{EuropeanDirector} \sqsubseteq \mathit{Director}$$
$$\mathit{Review} \sqsubseteq \exists \mathit{about}.\mathit{Movie} \sqcap \exists \mathit{hasComment}.\top$$

$\mathcal{S}_1: \quad \{R1(\mathit{title}, \mathit{year}, \mathit{director})\}$ since 1960 but only European directors

$\mathcal{S}_2: \quad \{R2(\mathit{title}, \mathit{critique})\}$ since 1990

```
SELECT R1.director AS x FROM R1                              →   x:  EuropeanDirector
SELECT R1.director AS x FROM R1                              →   x:  Director
SELECT R1.title AS x FROM R1 UNION SELECT R2.title AS x FROM R2 → x:  Movie
SELECT R1.title AS x, R1.year AS y FROM R1                   →   (x,y):  createdInYear
SELECT R1.title AS x, R1.director as z FROM R1               →   (x,z):  hasDirector
SELECT R2.title AS x, R2.critique as z FROM R2               →   f(x,z):  Review
SELECT R2.title AS x, R2.critique as z FROM R2               →   (f(x,z), x):  about
SELECT R2.title AS x, R2.critique as z FROM R2               →   (f(x,z), z):  hasComment
```

Query: Give me all movies and their reviews from the year 1998

$q = x : \mathit{Movie} \wedge (x, 1998) : \mathit{createdInYear} \wedge (z, x) : \mathit{about} \wedge (z, u) : \mathit{hasComment}$

Query Rewriting: Example for GAV 2/2

```
SELECT R1.director AS x FROM R1                                →  x:  EuropeanDirector
SELECT R1.director AS x FROM R1                                →  x:  Director
SELECT R1.title AS x FROM R1 UNION SELECT R2.title AS x FROM R2 →  x:  Movie
SELECT R1.title AS x, R1.year AS y FROM R1                     →  (x,y):  createdInYear
SELECT R1.title AS x, R1.director as z FROM R1                 →  (x,z):  hasDirector
SELECT R2.title AS x, R2.critique as z FROM R2                 →  f(x,z):  Review
SELECT R2.title AS x, R2.critique as z FROM R2                 →  (f(x,z), x):  about
SELECT R2.title AS x, R2.critique as z FROM R2                 →  (f(x,z), z):  hasComment
```

$q = x : \mathit{Movie} \wedge (x, 1998) : \mathit{createdInYear} \wedge (z, x) : \mathit{about} \wedge (z, u) : \mathit{hasComment}$

1. Pre-processing: can be skipped here
2. Unfolding:

```
x:  Movie                  → SELECT R1.title AS x FROM R1 UNION SELECT R2.title AS x FROM R2
(x,1998):  createdInYear   → SELECT R1.title AS x, R1.year AS y' FROM R1 WHERE R1.year = 1998
(z,x):  about              → SELECT R2.title AS x, R2.critique as u FROM R2
(z,u):  hasComment         → SELECT R2.title AS x, R2.critique as u FROM R2
```

($f^{-1}(z) = (x, u)$) for the last two lines.

Combine $(f^{-1}(z) = (x, u))$:

```
SELECT R1.title AS x, R2.critique AS u FROM R1, R2 WHERE
    R1.year = 1998 AND R1.title = R2.title
```

Outline

Summary

- Ontology-based Data Access as a data integration approach
- Heterogeneity of database schemas need unifying global schema to be processed in an application-independent manner
- The impedance mismatch
- OBDA Architecture: $(\mathcal{G}, \mathcal{S}, \mathcal{M})$
- Mappings: GAV, LAV
- Canonical databases and query rewriting

Pointers to further reading

- OBDA resources: `http://obda.inf.unibz.it`
- H. Wache, T. Voegele, T. Visser, H. Stuckenschmidt, H. Schuster, G. Neumann, and S. Huebner. Ontology-based integration of information: a survey of existing approaches. IJCAI-01 Workshop: Ontologies and Information, page 108-117. (2001)
- Maurizio Lenzerini. A Tutorial on Data Integration: `http://www.tks.informatik.uni-frankfurt.de/data/events/deis10/downloads/10452.LenzeriniMaurizio.Slides.pdf`
- Maurizio Lenzerini. Data integration: a theoretical perspective. Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. 2002.