Ontology-Based Data Access

Subjects, Issues and Trends

Leopoldo Bertossi[⋆]

Carleton University
School of Computer Science
Ottawa, Canada

[⋆]: Faculty Fellow of the IBM Center for Advanced Studies

A Start: Metadata in Data Management

- Metadata (MD) is data about data

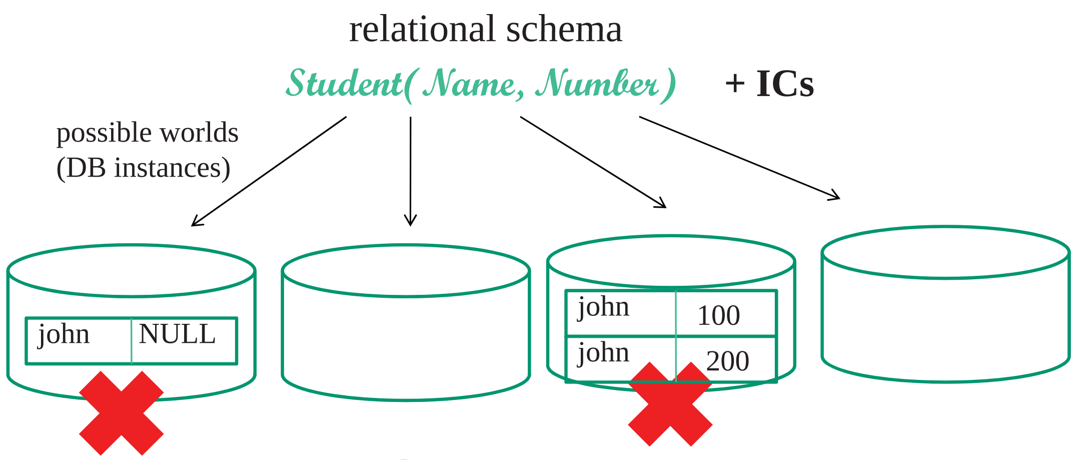
  An upper layer that gives information about a lower layer

  For example, about the data in relational tables

- We already know about MD in relational DBs: schemas, data types and domains, integrity constraints (ICs)

- If ICs are satisfied by the DB (as expected, but not always true), they provide synthetic, higher-level knowledge
  - ICs capture semantics (meaning) of data [2, 8]
  - By filtering out inadmissible (inconsistent) instances, the spectrum of possible instances is narrowed down

    By doing so, better targeting the intended meaning
  - Decreasing uncertainty



relational schema

Student(Name, Number) + **ICs**

possible worlds
(DB instances)

john	NULL

john	100
john	200

+ $\forall xy(Student(x,y) \rightarrow y \neq \text{NULL})$ + $\forall xyz(Student(x,y) \wedge Student(x,z) \rightarrow y = z)$

(capturing semantics via ICs, eliminating possible worlds)

- ICs tell us something about the stored data, still not much though
- ICs can be used, e.g. at query answering time

  For semantic query optimization
- ICs are also useful for interoperability purposes

  When data systems have to interact and possibly be integrated

  They tell us something about what’s stored in the data source

Why not going beyond in terms of MD?

What else do we know or have after having created a relational DB?

Recovering ER models as Metadata

- When creating a database, we usually start from en entity-relationship (ER) model
- An ER model represents an external, data-related reality

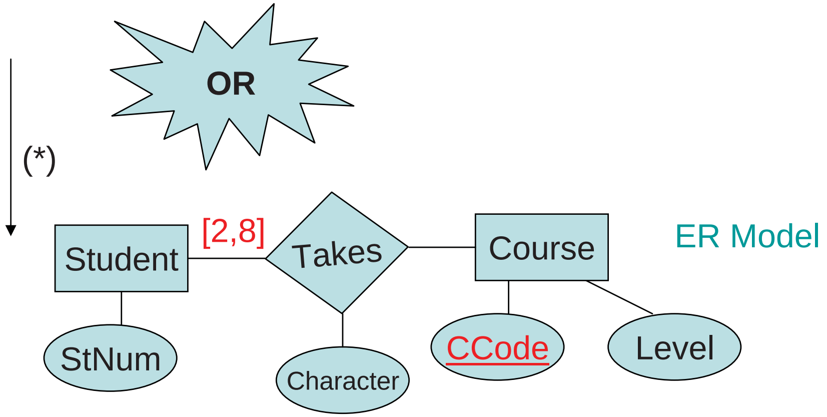
  For example, a model of a business environment

  The model is given as an ER diagram (UML diagram)
- The ER model is closer to the reality than the relational DB to be (which is also a model)
- The ER model is usually forgotten after the DB is created
- The ER model could be used as metadata!

- When creating a relational database, we usually start from an outside reality (OR), e.g. a company, a university, etc.

  We want to model that OR, i.e. produce an abstract, simplified description or representation of OR (leaving aside non-relevant, contingent aspects and details)

- A model can be an ER model, in terms of entities and relationships between them



- For the model to be a good model of OR, it must have a semantics or meaning that corresponds to OR

  ... and keeps the correspondence (*) in place (semantically correct)

- That is why we impose in the model some semantic constraints, like those in red in it

  - A student must take between 2 and 8 courses

  - The course code is a key for the entity: If two objects in *Course* coincide in their values for *CCode*, then their other attribute values must coincide too

- Without those constraints, there could be too many possible ORs that conform to the ER model

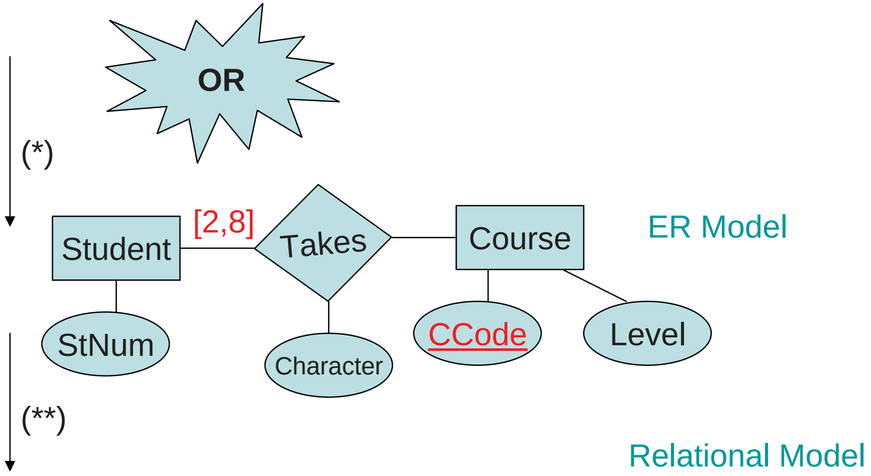
  The model becomes too ambiguous or uncertain

- Imposing semantic constraints eliminates unintended ORs

  ... by narrowing down meaning and filtering out undesirable ORs (other than the intended one)

  We want the ER model to be as close as possible to the initial OR

- The usual next step is producing a relational model from the ER model



Relational Schema: Student(StNum,Address), Course(<u>CCode</u>,Level,MaxReg), Takes(StNum,CCode,Character) + ICs

- The relational model is also a model of OR

- Now a logical model that uses the languages of predicate logic and set theory

- The relational ICs become part of the model, and are also semantic constraints

  Some of them come from the original ER model with its semantics constraints

- As mentioned above, the ER model may be discarded (or not used) after the relational DB is created and populated

  But the ER model contains much semantic information

  It could be put to good use: It could become metadata

  A semantic layer -that can be used with the DB- and is closer to OR and what the user understands

How to combine a diagrammatic model with a logical model?

How to realize the integration?

So that a computer system can take advantage of the combination ...

Interlude on the Semantic Web

This idea of a semantic layer is at the very basis of the semantic web effort [1, 11]

The idea is to wrap web sites with descriptions of their contents (resources)

So that systems that access them will know:

(a) What to find in them

What resources, and how they are presented and related

(b) Conditions satisfied by those resources

Useful for querying, integrating and making web sites interoperate

All this has to be automatized ...

Logical languages have been created to produce those semantic layers

Those descriptions become ontologies, which are knowledge-bases expressed in standardized logical languages

Since all has to be automatized, the ontology languages are expected (not always successfully) to keep a balance between expressive power and difficulty of reasoning

Languages have been proposed: RDF, RDF(S), OWL (in several versions, light- and heavy-weight), etc.

E.g. RDF(S) has found many applications in data management, and there are multiple RDF DBs (check out DBpedia!)

Some of those languages are being used to express ontologies as metadata for data sources

ER Models as Ontologies and OBDA

Logical languages to express metadata can interact with the logical data model (database)

Being the ER model a diagrammatic model, it can be reconstructed as a symbolic and logic-based ontology

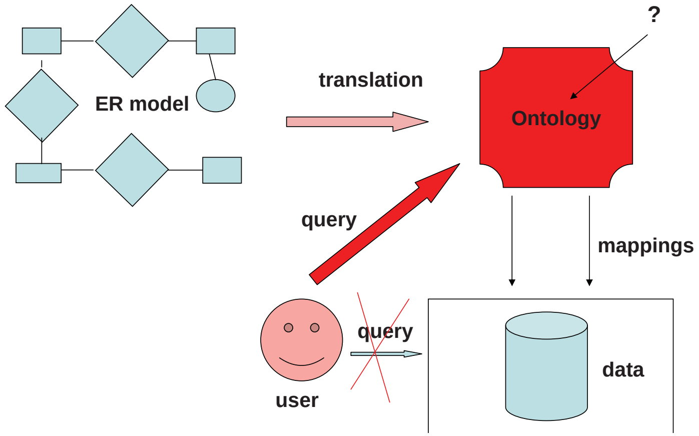
In general, an ontology is a (logical) description of a set of concepts and their relationships [7]

The ontology becomes metadata, now an explicit and formal ER model

The ontology (ex ER model) -being closer to the user or business reality- can be used to query the DB

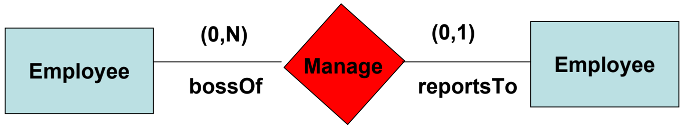
Querying data sources through ontologies is an active research area

OBDA: Ontology-based data access [9]



ER model is replaced by (reconstructed as) a symbolic, logical *ontology*

For example, for the following entities/relationship



Introduce basic predicates for the ontology:

- Unary predicates for concepts: $Employee(\cdot)$
- Binary predicates for roles: $BossOf(\cdot,\cdot)$, $ReportsTo(\cdot,\cdot)$

Symbolic statements go into the ontology

E.g. to capture the $(0,1)$ constraint on the ER's **reportTo**: *"Every employee reports to at most one employee"* :

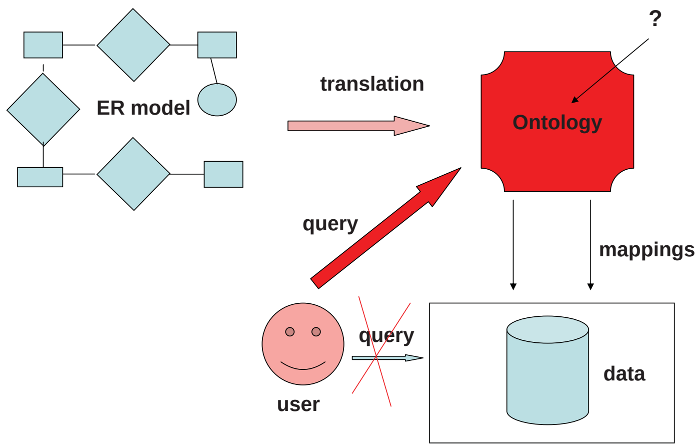
$\forall x(Employee(x) \rightarrow \exists^{\leq 1} y(Employee(y) \wedge ReportsTo(x,y))$[1]

A symbolic, machine-processable sentence ...

Back to OBDA ...

Query language is the language of the ontology

Data stay underneath



Ontology queries are internally “translated” into DB queries

For that, use the mappings between the ontology and the underlying database (data source)

[1] I.e., $\forall x(Employee(x) \rightarrow \forall x \forall y_1 y_2((Employee(y_1) \wedge ReportsTo(x,y_1) \wedge Employee(y_2) \wedge ReportsTo(x,y_2)) \rightarrow y_1 = y_2)$. If the ER constraint were $(1,1)$, it would be: $\forall x(Employee(x) \rightarrow \exists y(Employee(y) \wedge ReportsTo(x,y)) \wedge \forall x \forall y_1 y_2((Employee(y_1) \wedge ReportsTo(x,y_1) \wedge Employee(y_2) \wedge ReportsTo(x,y_2)) \rightarrow y_1 = y_2)$

Just for the gist:

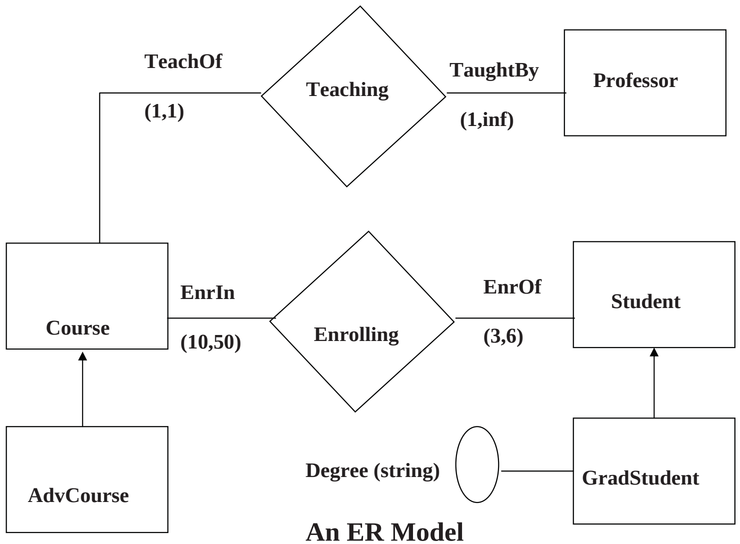
The link between $\mathbf{AdvCourse}$ and $\mathbf{Course}$ is an IS-A link

As an ontology written in Description Logic (DL)

Entities become DL-concepts; ER links become DL-roles (binary predicates)

DL is at the basis of SW languages, such as OWL

($\sqsubseteq$ is $\subseteq$ or $\rightarrow$; $\sqcap$ is $\cap$ or $\wedge$; $^-$ denotes the inverse role (predicate); original constraints in red)



An ER Model

$$\begin{aligned}
\mathit{Teaching} &\sqsubseteq \forall \mathit{TeachOf}.\mathit{Course} \ \sqcap\ \exists^{=1}\mathit{TeachOf} \ \sqcap \\
&\quad \forall \mathit{TaughtBy}.\mathit{Professor} \ \sqcap\ \exists^{=1}\mathit{TaughtBy} \\
\mathit{Enrolling} &\sqsubseteq \forall \mathit{EnrIn}.\mathit{Course} \ \sqcap\ \exists^{=1}\mathit{EnrIn} \ \sqcap \\
&\quad \forall \mathit{EnrOf}.\mathit{Student} \ \sqcap\ \exists^{=1}\mathit{EnrOf} \\
\mathit{Course} &\sqsubseteq \forall \mathit{TeachOf}^{-}.\mathit{Teaching} \ \sqcap\ \exists^{=1}\mathit{TeachOf}^{-} \ \sqcap \\
&\quad \forall \mathit{EnrIn}^{-}.\mathit{Enrolling} \ \sqcap\ \exists^{\geq 10}\mathit{EnrIn}^{-} \ \sqcap\ \exists^{\leq 50}\mathit{EnrIn}^{-} \\
\mathit{AdvCourse} &\sqsubseteq \mathit{Course} \\
\mathit{Professor} &\sqsubseteq \forall \mathit{TaughtBy}^{-}.\mathit{Teaching} \\
\mathit{Student} &\sqsubseteq \forall \mathit{EnrOf}^{-}.\mathit{Enrolling} \ \sqcap\ \exists^{\geq 3}\mathit{EnrOf}^{-} \ \sqcap\ \exists^{\leq 6}\mathit{EnrOf}^{-} \\
\mathit{GradStudent} &\sqsubseteq \mathit{Student} \ \sqcap\ \forall \mathit{Degree}.\mathit{String} \ \sqcap\ \exists^{=1}\mathit{Degree}
\end{aligned}$$

The mappings are between unary and binary predicates in the ontology and database predicates (tables), which can be of any arity

The restricted syntax of DL makes automated reasoning feasible, and sometimes, also efficient

Notice that full classical predicate logic of which (most of the variants of) DL is a (are) fragment(s) is provably undecidable

The DL ontology above could be written in OWL

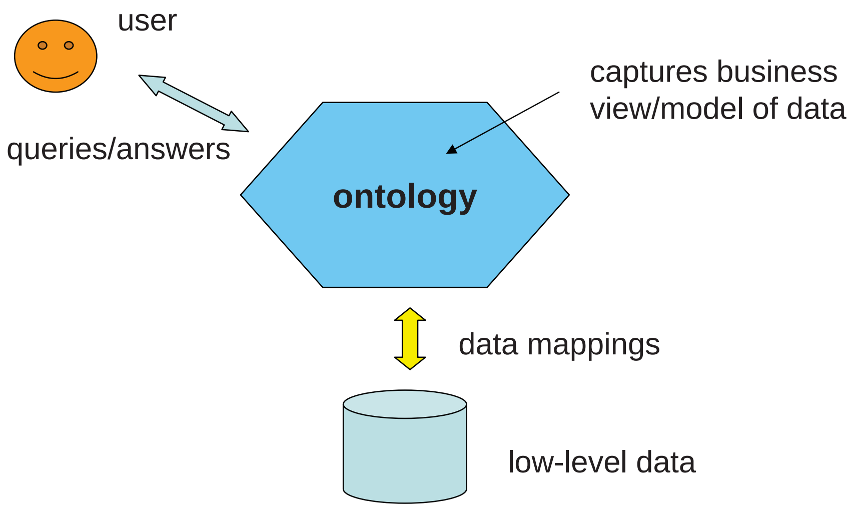
(Above, ER constraints captured in red in the ontology)

By reasoning we can infer that constraints that apply to $Course$ also apply to $AdvCourse$

And less direct logical consequences from the ontology

Ontologies can be more expressive than ER models

We could start directly with/from an ontology (not necessary coming from an ER model)



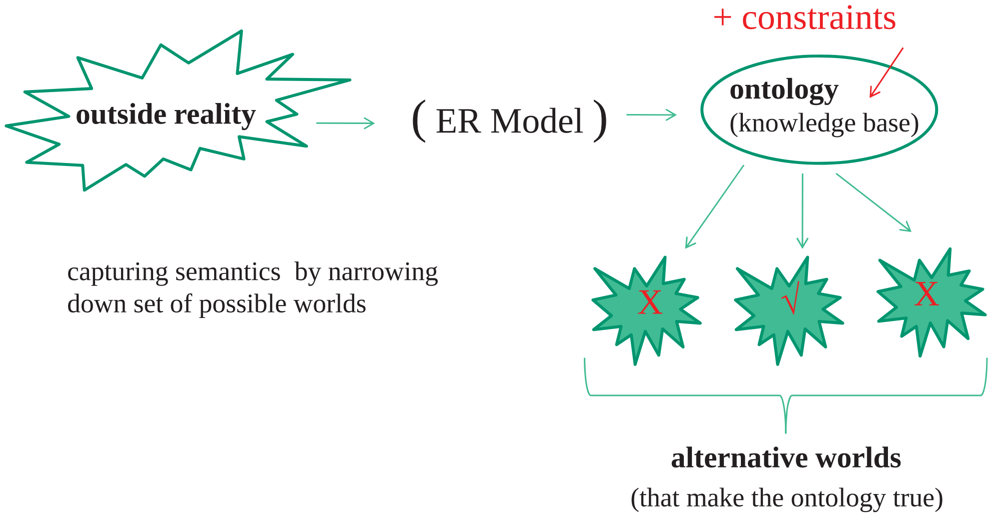
Their logic-based languages have precise syntax and semantics

The ontology can be used to capture more semantics

... in declarative, precise, and executable terms ...

It is possible to do automated reasoning from those ontologies

Via extra logical conditions (constraints) unintended possible worlds that make the ontology true (satisfy the ontology) can be filtered out (cf. page 3)



+ constraints

outside reality

(ER Model)

ontology

(knowledge base)

capturing semantics by narrowing down set of possible worlds

alternative worlds

(that make the ontology true)

This ontology-based approach enables conceptually simpler and more flexible integration of data management with higher-level reasoning systems

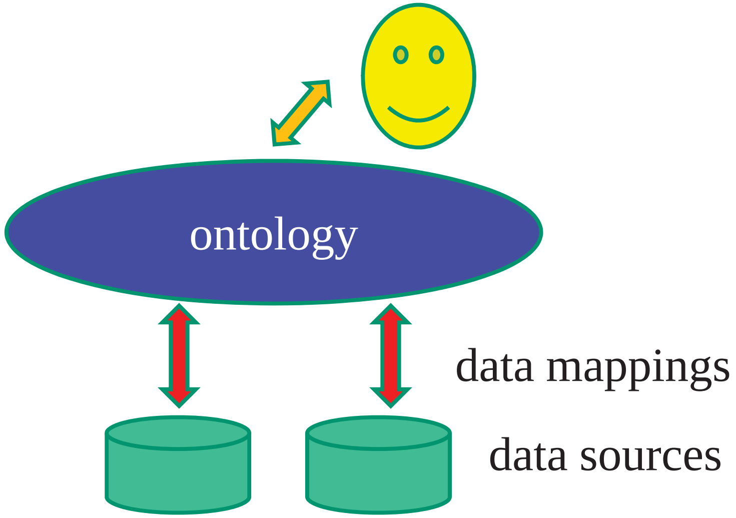
... intelligent information systems, knowledge bases, ontologies, semantic web repositories, etc.

Those ontologies can be useful for interoperability and integration purposes

They convey semantics [10], then sources can be compared in terms of “semantic compatibility”

Other data sources (at the bottom of the picture above) could be added

Integrating data sources through/under the same ontology



The Data Integration Connection

- Integrating data sources is a crucial problem in business applications

  Not only there, also for example, in bioinformatics

- Sources can be databases, but also data repositories of all possible kinds

  From structured data (e.g. in relational databases) to documents and WWW pages

- Crucial issue is variety   Data come in all forms, formats, ...

- Heterogeneity is the norm

- There are also semantic issues

- The semantics may be conflicting: think of two mutually contradictory logical theories (ontologies) for two DBs

Example: Two databases with same schema $\mathcal{S} = \{R[A,B], S[B,C]\}$

One DB has the referential IC: $R[B] \subseteq S[B]$
(each value for B in relation R must appear in relation S)

The other has the denial constraint: $not\ (R(A,B), S(B,C))$
(no joins allowed between the two tables)

The two ICs together are inconsistent (in a limited form though: only DBs with empty tables for predicate R can make both true)

- The semantics may be mutually consistent, i.e. their union as logical theories is consistent, but not the combined data

Example: Two student databases, with same schema and same key constraint for the student number

Even if the two DBs separately are consistent, the combination of the two DBs may violate the key constraint in common

Different forms of integration:

- Different basic approaches and paradigms for data integration (DI)

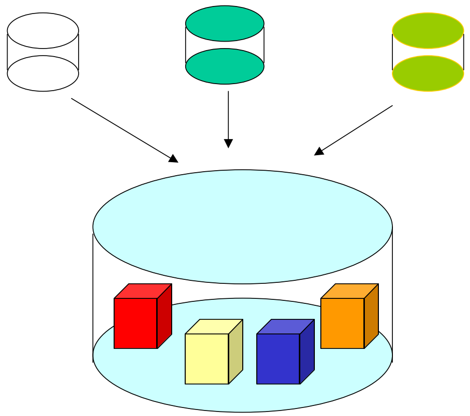
  And hybrid approaches, combinations of the former, that can be combined in complex solutions and systems

  - Materialized: a new physical, material data repository is created
  - Mediated: data stay at the sources, a virtual integration system is created

- In all cases, mappings are needed, to correlate and exchange data between data sources and data targets

Materialized approaches:

A new physical database is created, importing data from other data sources



Data sources may be independent and autonomous

Data warehouses (DWHs): prominent example of materialized DI

Data at the DWH structured differently than those at the sources

Multidimensional business-oriented representation at the DWH

Data cubes in the DWHs, suggesting different dimensions of data

They give context to (usually) numerical data

DWH can be conceived as a collection of materialized views, defined on the combination of data sources

Sources and DWHs are meant to be used for different purposes, e.g. transactional/operational vs. business-oriented analysis

Mappings from sources are kept, for refreshment (usually one-directional mappings)

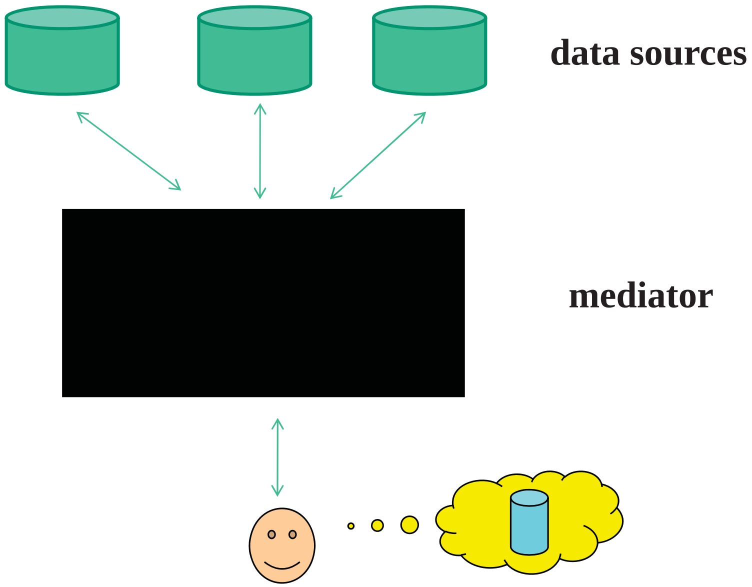
Virtual approaches:

Through a mediator [12]

SW system offering DB-like schema interface

User interacts with mediator

Data stay at the sources



data sources

mediator

Mappings allow to send specific queries to sources and retrieve data

Notice the similarity with ontology-based data integration (page 20)

Example: OCICS wants to virtually integrate their CU and OU DBs

Sources: Carleton U. Ottawa U.

CUstudents	Number	Name
	101	john
	102	mary

OUstudents	Number	Name
	103	claire
	101	peter

SpecialCU	Number	Field
	101	alg
	102	ai

SpecialOU	Number	Field
	101	db

Single global relation schema, at mediator level

$$Students(Number, Name, Univ, Field)$$

Mapping between the source schemas and the mediated schema?

CUstudents	Number	Name
	101	john
	102	mary

OUstudents	Number	Name
	103	claire
	101	peter

SpecialCU	Number	Field
	101	alg
	102	ai

SpecialOU	Number	Field
	101	db

Mediated schema: $Students(Number, Name, Univ, Field)$

A logical schema mapping: (uses two Datalog rules for view definitions)

$$CUstudents(x, y), SpecialCU(x, z) \rightarrow Students(x, y, \text{‘cu’}, z)$$

$$OUstudents(x, y), SpecialOU(x, z) \rightarrow Students(x, y, \text{‘ou’}, z)$$

Students becomes a view defined as a disjunction of two conjunctive queries

Global relation as a view of source relations (not the only possibility)

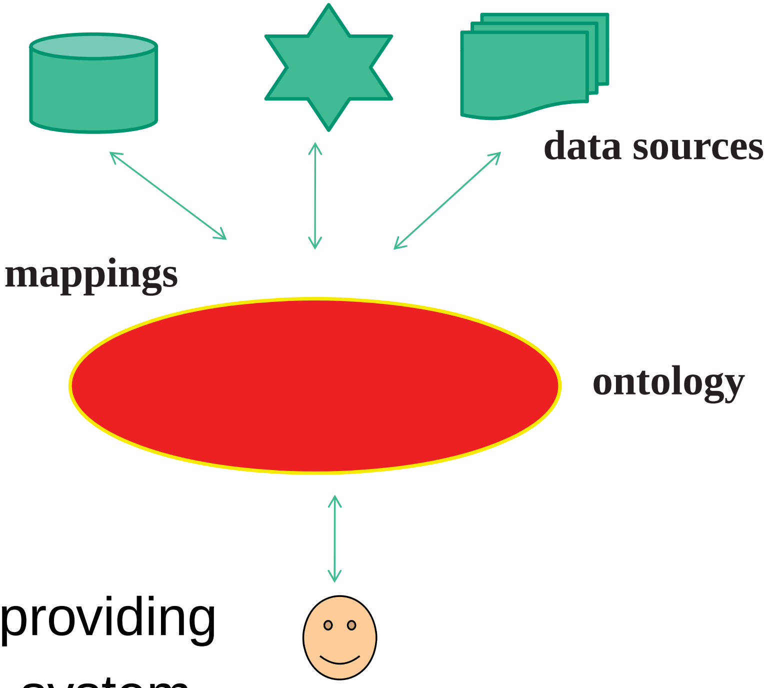
(Can be put as a view defined in relational calculus:

$$\forall xyuz[(CUstudents(x, y) \wedge SpecialCU(x, z) \wedge u = \text{‘cu’}) \vee$$
$$(OUstudents(x, y) \wedge SpecialOU(x, z) \wedge u = \text{‘ou’}) \rightarrow Students(x, y, u, z)]$$

The mappings above are stored at- and managed by the mediator

The logical part (the non-procedural components) of the mediator could be conceived as an ontology

More generally:



Different kinds of sources

Sometimes with wrappers, providing right presentation for the DI system

What Languages for ODBA?

- We saw that (dialects of) DL can could be such an ontological language
- Something closer to database practice?
- Datalog has been around for some years in the DB community

  As a query and view definition language for relational DBs

  As opposed to relational algebra/calculus and older versions of SQL, Datalog provides recursion

Parent	A1	A2
	juan	pablo
	adam	cain
	adam	abel
	eve	cain
	pablo	luis
	.	.

$$
\begin{aligned}
Ancestor(x,y) \quad &\leftarrow \quad Parent(x,y)\\
Ancestor(x,z) \quad &\leftarrow \quad Ancestor(x,y), Parent(y,z)
\end{aligned}
$$

- Datalog has many nice properties and implementations, but also limited expressive power

- Can we extend Datalog to make it more expressive while keeping most of its nice properties?

Datalog± as an Ontological Framework

- Datalog± is a family of extensions of classic Datalog, with new kinds of rules and constraints [4, 6]
- Its languages allow to represent ontological axioms and integrity constraints that cannot be expressed in Datalog
- The idea is to extend Datalog with new constructs to gain expressive power
- While trying to keep the good properties of Datalog:

$\longrightarrow$ declarativity, clear logical semantics,
effectiveness & efficiency

(as extensions of whatever available for Datalog)

Several applications:

- Express/represent ontologies that interact with data sources
- Represent conceptual data models, and semantic layers on top of databases
- Ontology-Based Data Access (OBDA)
  - Query a database through the ontology
  - In the language of the ontology (better understood by- and closer to the user)
  - Automatically access the underlying data sources
  - Get answers through Datalog evaluation
- Datalog± ontologies can represent: ER, Semantic Web languages/ontologies, UML with object classes, ... [5, 3]
  (but not classic Datalog!)

Most prominent new ingredients: (the “$+$” in Datalog$\pm$)

- Rules in Datalog$\pm$ admit existentially quantified variables:

  $$\exists x P(x,y) \ \leftarrow \ R(y,z)$$

  Can be seen as tuple-generating dependencies (TGDs)

- Negative Constraints (NCs): (in particular, denial constraints)

  $$\bot \ \leftarrow \ P(x,y), R(y,z)$$

- Equality generating dependencies (EGDs):

  $$y = z \ \leftarrow \ P(x,y), P(x,z)$$

  In this case, a key constraint (KC)

Example: An incomplete EDB D of employers and employees

- Impose on D the TGD (usually as an inclusion dependency):

  "every manager is an employee"

  Expressed by a Datalog rule: $employee(x) \leftarrow manager(x)$

- Another TGD: *"every manager supervises someone"*

  As a rule in Datalog±: $\exists y \; supervises(x, y) \leftarrow manager(x)$

- Impose IC: *"employees are not employers"*

  As negative constraint (NC): $\perp \leftarrow employee(x), employer(x)$

- An EGD: *"every employee is supervised by at most one manager"*

$$x = x' \leftarrow supervises(x, y), supervises(x', y)$$

Properties & issues:

- The "$-$" in Datalog$\pm$ refers on syntactic restrictions we impose on the rules and their (syntactic) interactions
- This limits the gained expressive power
- We can still use Datalog$\pm$ to express ER models and much more
- It can be used as an ontological language
- It can be used as a language to extend incomplete DBs
- The syntactic restrictions ensure that query evaluation (QE) becomes feasible and sometimes efficient

  (Without them, QE under Datalog$\pm$ can be undecidable/non-computable)
- Datalog$\pm$ is still declarative and has a precise and clean semantics
- QE can be implemented

Conclusions

- Ontologies have been used for some time in AI (KR) and the Semantic Web

- Now they are being increasingly used in data management

  In particular, in interaction with relational DBs

- Ontologies can be used to access DBs through a model that is close to the user or application environment, e.g. business data

- They can also be used for data integration

- The ontological "schema" can be different from the DB schema

  Connection established via logical mappings

- DL and Datalog$\pm$ have been used for OBDA

- Datalog± is a family of extensions of Datalog

  The latter has been around for more than two decades in the DB community

- DL and Datalog± have been used to symbolically/logically represent ER, UML, ..., models

- Many applications are still to be unveiled

- There are many interesting open research problems

References

[1] Tim Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001, pp. 3443.

[2] Alexander Borgida and John Mylopoulos: Data Semantics Revisited. Proc. SWDB, Springer LNCS 3372, 2004, pp. 9-26.

[3] Andrea Cali, Georg Gottlob and Thomas Lukasiewicz: A General Datalog-Based Framework for Tractable Query Answering over Ontologies. *Journal of Web Semantics*, 2012, 14:57-83.

[4] Andrea Cali, Georg Gottlob and Andreas Pieris. Towards More Expressive Ontology Languages: The Query Answering Problem. *Artificial Intelligence*, 2012, 193:87-128.

[5] Andrea Cali, Georg Gottlob and Andreas Pieris. Ontological Query Answering under Expressive Entity-Relationship Schemata. *Information Systems*, 2012, 37(4):320-335.

[6] Andrea Cali, Georg Gottlob, Thomas Lukasiewicz and Andreas Pieris. A Logical Toolbox for Ontological Reasoning. *SIGMOD Record*, 2011, 40(3):5-14.

[7] B. Chadrasekaran, J. Josephson and V. Richard Benjamins. What are Ontologies, and Why Do We Need Them?. IEEE Intelligent Systems, Jan/Feb. 1999, pp. 20-26.

[8] David Harel and Bernhard Rumpe. Meaningful Modeling: What's the Semantics of "Semantics"?. *IEEE Computer*, 2004, 37(10): 64-72.

[9] Maurizio Lenzerini. Ontology-Based Data Management. Proc. AMW 2012, CEUR Proceedings, Vol. 866, pp. 12-15.

[10] Alexander Maedche, Boris Motik, Ljiljana Stojanovic, Rudi Studer and Raphael Volz. Ontologies for Enterprise Knowledge Management. *IEEE Intelligent Systems*, 2003, 18(2):26-33.

[11] Nigel Shadbolt, Tim Berners-Lee and Wendy Hall. The Semantic Web Revisited. *IEEE Intelligent Systems*, 2006, 21(3):96-101.

[12] Gio Wiederhold. Mediators in the Architecture of Future Information Systems. *IEEE Computer*, 1992, 25(3):38-49.