Semantic Web
(Winter Term 2019/20)

(c) Prof Dr. Wolfgang May
Universität Göttingen, Germany

may@informatik.uni-goettingen.de

With contributions by Franz Schenk, Lars Runge, Sebastian Schrage
and indirectly several of our students.

Lecture: Advanced Course (Master) in Informatics; 3+1 hrs/week, 6 ECTS Credit Points
Lab Course: 2 hrs/week + exercises, 6 ECTS Credit Points

- participants should have knowledge of first-order logic syntax, semantics, model theory and tableau calculus.
- the slide set contains an introduction to first-order logic (e.g. taught in the modules "Formale Systeme" (in German, BSc) and "Deductive Databases") for recalling concepts and introducing the notation used in this lecture.

A comprehensive German-English dictionary can e.g. be found at

http://dict.leo.org/

AIMS OF THE COURSE

"The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation." – Tim Berners-Lee, James Hendler, Ora Lassila, "The Semantic Web", in *Scientific American*, May 2001.

- What is an ontology?
- The "Semantic Web Tower"
- RDF as a data model, query langages etc.
- RDFS: a restricted language and model theory for schema metadata
- underlying theoretical concepts: Description Logics
- OWL

Chapter 1
Introduction

THE BUZZWORD OF THE DAY: SEMANTIC WEB

“invented” by Tim Berners-Lee in an article in *Scientific American*, May 2001.

- computer-understandable semantics of data
- more “intelligent” applications use this semantics
- integration of data
- integration of behavior

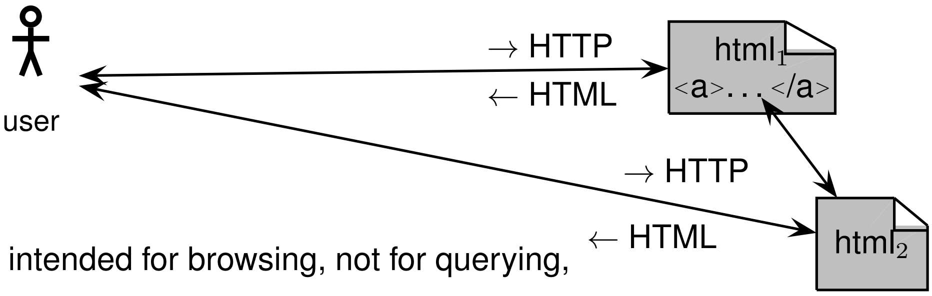
Different people mean quite different things when talking about the “Semantic Web” (e.g., focusing on “Semantic technologies”, “Web 2.0”, “Knowledge Representation”).

This lecture collects topics that are also useful outside the “Semantic Web”, and provide important foundations for the “Semantic Web”.

1.1 Data on the Web

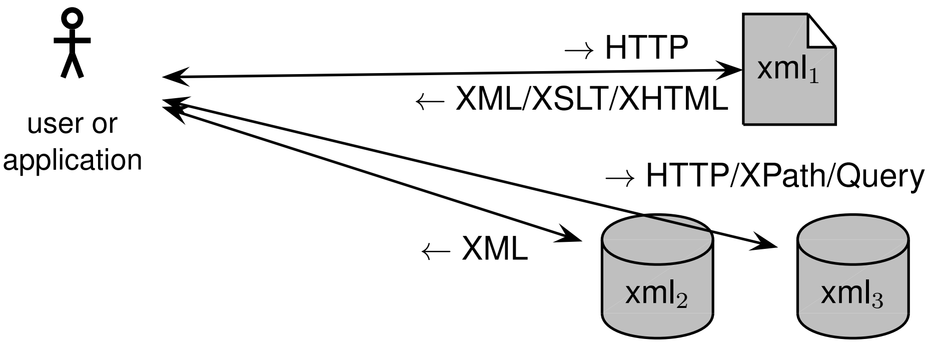
- World Wide Web:
  - Design of HTML started in 1989 (CERN), standard in 1991,
  - Browsing, Links between sources (“Web”)
- XML in the WWW:
  - XML as a data model + family of related languages.
  - XML data sources
  - Web-Services: successor of CORBA, based on XML data exchange
  - no “XML Web”, but mainly isolated sources, nearly not interlinked (XLink)
  - problem of *data integration* between sources
  - querying “the Web” as e.g. “give me a (train/flight/...) connection from Göttingen to St. Malo next weekend” not possible
  - users disappointed
- existing facts not appropriately accessible
- information = facts + semantics

ARCHITECTURE: THE HTML WEB



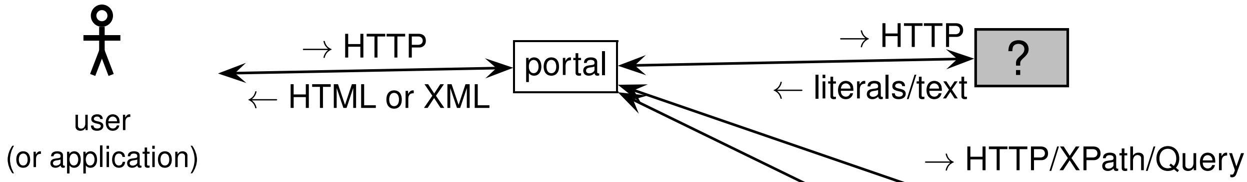
- HTML data intended for browsing, not for querying,
- access by absolute URL or by navigation through the Web,
- HTML (text, tables, list) is "data", but still far from information, is "machine readable", but only understandable for the browser, not for data-related tasks,
- "data integration" task mostly done offline (look up flight FRA-CDG and CDG-RNS at http://www.billigerfliegen.de, write down departure time, look up connection GOE-FRA at http://www.bahn.de),
- extracting information from HTML "data" requires detailed wrapper programming (into a common data model) based on the representational structure of the HTML pages. (example: generation of the Mondial database in 1997)

ARCHITECTURE: THE XML WEB

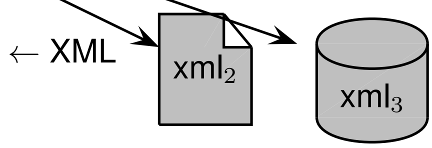


- XML is a data model
- can be wrapped by XSLT stylesheets to HTML for browsing,
- nearly no links between XML documents (XLink not used),
- XML data integration by high-level languages (XSLT, XQuery), requires knowledge of source DTDs or XML Schemas, manually written integration program (XQuery, XSLT).

XML Web: Data Integration with Portals



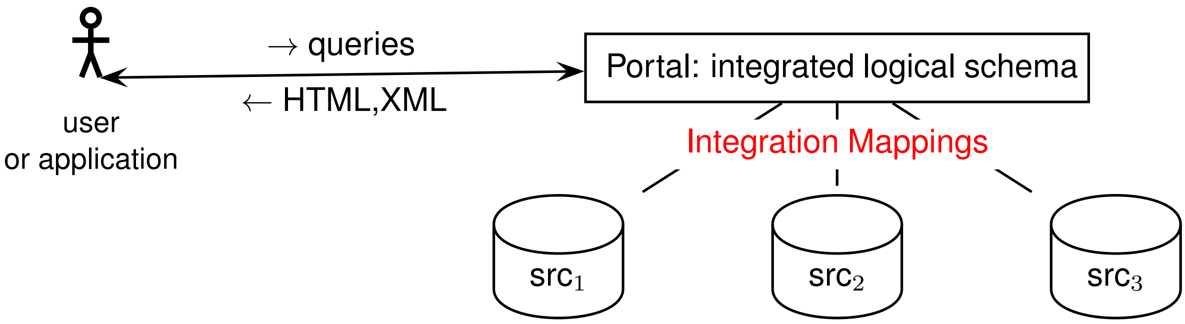
Portals provide services/access for a given *domain*, e.g.



- HTML: e.g. traveling
  graphical user interfaces: browsing/navigation, forms
  e.g. `http://www.billiger-fliegen.de` is a portal to airlines
- maps different schemas to the common one
  - wrappers for non-XML data sources (legacy),
  - (mostly hand-written) integration program in high-level language,
- obvious advantages if sources use a commonly agreed DTD/XML Schema,
- must be adapted if schema of a source changes or a source is removed or added.
- much more difficult: portal for access by computers
- computers lack the background knowledge that a user has wrt. a graphical interface

Data Integration: Centralised Architecture

- federated databases of the 80s and 90s
- WWW portals (since late 90s)

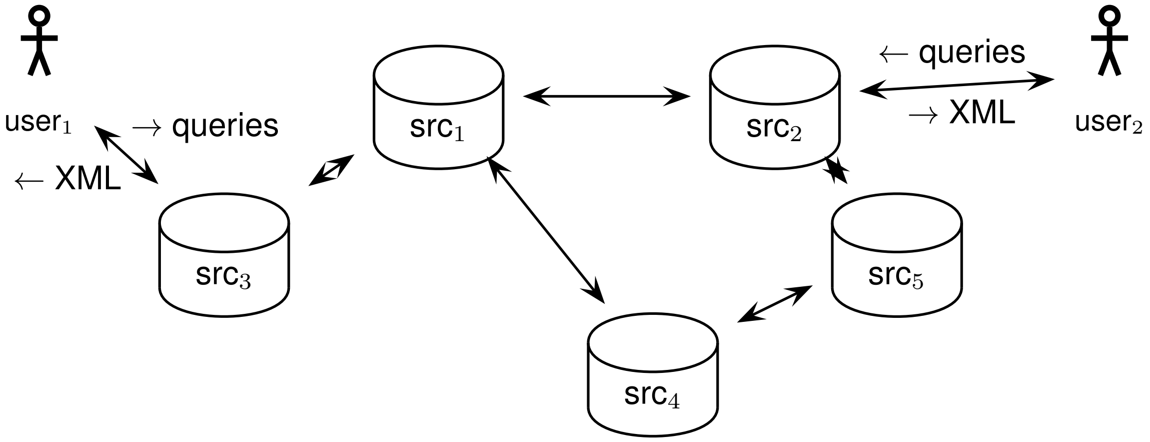


Integration Mapping

- usually "Global as View" (GAV), defining the integrated database as a view over the original sources,
- "Warehouse/materialized approach": evaluate queries regularly and maintain an integrated database,
- "Virtual Approach": queries are stated against the integrated schema and *rewritten* (in GAV: view expansion) and stated against the original databases.

DATA INTEGRATION: PEER-TO-PEER ARCHITECTURE

- P2P: since mid-90s (Tsimmis/OEM, Piazza, Napster, Gnutella, ...)



- communication between nodes
- Peer-to-Peer mappings between sources (bidirectional)
  worst case: exponential number of mappings needed
- located in the data nodes (e.g., using XLinks and some embedding)
- main issues in P2P: routing + integration mappings

1.2 Outlook: Semantic Web

Make data acquisition and integration easier *and* more powerful:

- underlying technology: Internet + XML, Web Services,
- agreed terminology ("ontology") throughout an application domain:
  - syntax and semantics of names,
  - agreed schema for identifiers of entities
    (⇒ matching of entities between different sources),
- unique mapping from the ontology to the data model,
- knowledge base instead of simple database:
  - expressive ontology language,
  - logical foundation, model theory,
  - reasoning mechanisms.
- semantics-based applications.

Data Model and Representation: RDF

Goal: data model *independent* from representational structure and syntax.

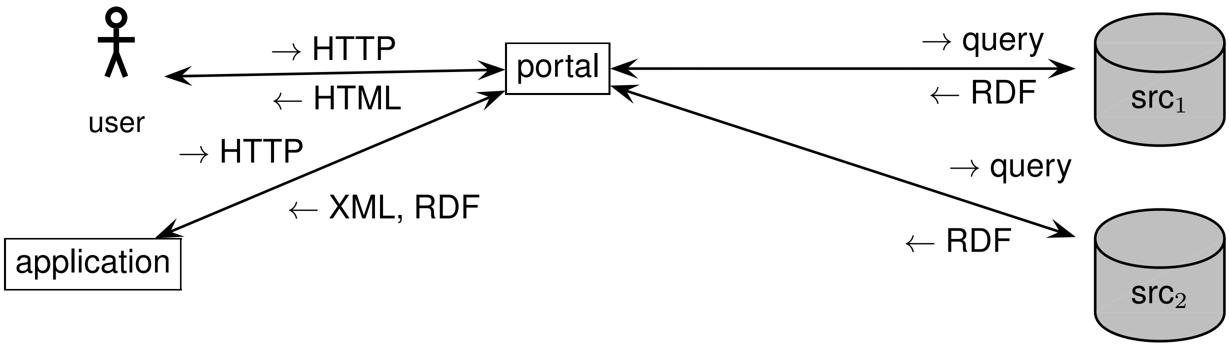
- *graph-based* data model
  - vertices: entities (identified by URIs – Unique Resource Identifiers) and values
  - edges: relationships and properties
  - all facts can be expressed by triples: Subject-Predicate-Object,
  - unified inherent semantics of the triples.
- metadata included within the data (RDFS, OWL) by predefined relationship names ("ontology vocabulary"),
- an ontology has a *unique* mapping to this data model,
- RDF data can be represented in XML
  - note: also relational data can be represented in XML – cf. SQLX
  - use XML *not* as a data model, but as a *data exchange format*.

Adding "Intelligence" to the Web

- Conceptual model of an ontology includes *knowledge*.
- reasoning not application-dependent, but based on generic concepts and formalisms that include schema, data, and further information:
  - class hierarchy
  - transitivity, inverse of properties, domains and ranges, cardinalities
  - derivation rules
  - algorithms/calculi
- logics, model-theory, model-based semantics of "the Web":
  "from (the information in) the Web one can infer . . . "
- reasoning (= application of Artificial Intelligence methods)

⇒ Description Logics, OWL (Web Ontology Language)

SEMANTIC WEB



- sources provide data in the RDF datamodel,
- transferred in a generic XML format,
- agreed names (ontology) and identifiers (URIs),
- data integration ≡ union of facts + reasoning.

- In addition to this portal-based architecture, peer-2-peer communication is supported.

SEMANTIC WEB ARCHITECTURE

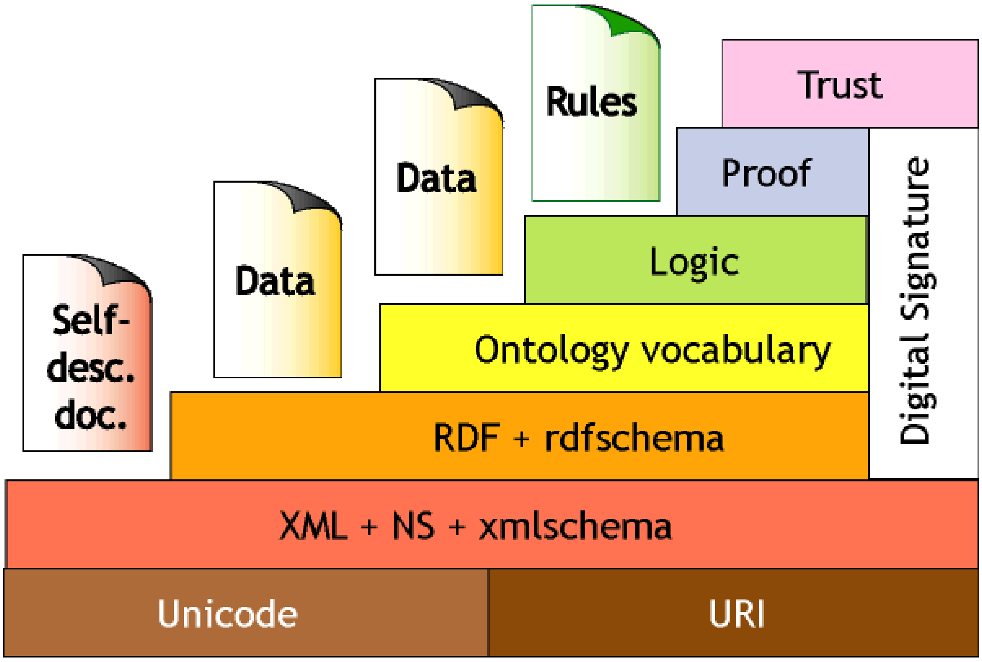
- intelligence located in application nodes and in the infrastructure
- Portals (integration + reasoning):
  serve e.g. as centralized entry points to a domain
- Peer-to-peer (P2P) communication:
  communication between "related" nodes

Additional Issues

- "find" information in the Semantic Web
  brokers & traders, yellow pages, registries, information propagation
  ⇒ "Linked Open Data (LOD)"
- "social" aspects: communities of nodes, trust, recommender services.

THE VISION: SEMANTIC WEB TOWER

- Tim Berners-Lee, talk at a W3C meeting, 2000.



Chapter 2
Metadata and Ontologies

- metadata: data about data
  - classical in databases: schema
  - classical in documents: author and last-changed-date, keywords in header section of Web pages
    problem: which keywords to use?
- Ontology: describe and relate all kinds of facts and relationships that are relevant when talking about a *domain*
  - geography, bioinformatics, medicine, ...
  - biology: carnivores eat animals
- more than schema: schema with annotations *what* notions mean
- No consensus whether an ontology is only the metadata (schema level), or if it contains the metadata + the instances

Ontologies

Description of concepts

- what concepts
- properties
- relationships (hierarchy, uses-relationship, any others)

Simple kind of Ontology: Taxonomy

- taxonomy: hierarchy
- sometimes also: typical properties (base for taxonomy)

Example Taxonomy: Biology

- animals, plants, mammals, fishes, insects, carnivores, felidae, canidae, bovinae, ...
- with their identifying properties
- tasks:
  - given an instance, determine its most specific class
  - show/prove that certain things hold for all instances of a class

Ontologies: Examples

Academic Ontology

- concepts like lecturers, professors, PhD candidates, research groups, lectures, exams, publications, etc.,
- relationships such as “professors lead research groups”, “professors supervise PhD candidates” etc.
- Description Logics: “T-Box” (Terminology/Taxonomy, intensional knowledge)

The IFI Ontology

- also describes the instances, e.g., that DH, JG, WM and SW are the professors, SW is dean of studies, tmg and dbis are research groups, and WM leads the dbis group etc.
- Description Logics: “T-Box + A-Box” (assertional knowledge, extensional knowledge)

Ontology Formalisms in Informatics

- (a first-order logic language/signature is an ontology)
- an ER Model is an ontology
- an UML class diagram is an ontology

⇒ an agreed ontology allows for interoperation and data interchange

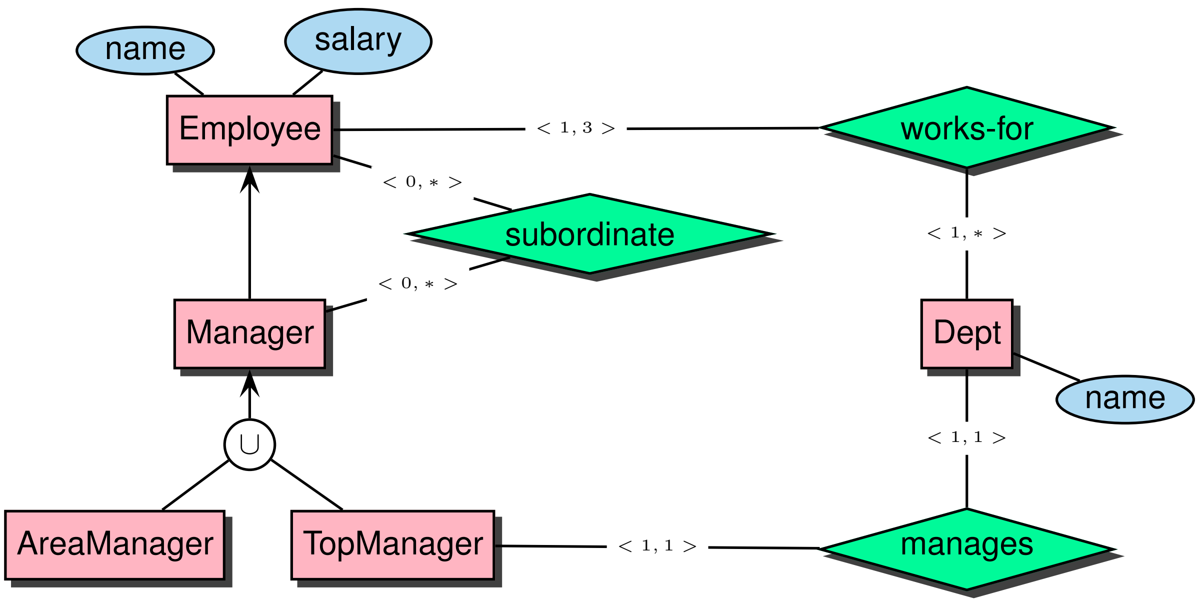
- an XML DTD or XML Schema is an ontology.
  It defines "names" *and* representational structure as an XML tree
  ⇒ it is more restrictive as an ontology should be.

Goals

- ontology formalism wrt. an *abstract* data model
- integration not of (XML) data structures, but of (abstract) models, representable in ... XML
- for reasoning (proving consistency or deriving knowledge), an ontology formalism must be based on some kind of logic

First-order logic as an ontology formalism?
first-order logic is undecidable

2.1 Example: ER Diagram as an Ontology



(TopManager: leads a department;
AreaManager: intermediate group leaders in a department)

What can be "models" of this ontology? How do you represent them? Give an example.

Semantics: Set theory

- a class is a set of instances,
  Employee={alice, bob, john, mary, tom, larry} and Manager={alice, bob, john, mary},
  AreaManager={mary} and TopManager={alice, bob, john},
  Dept={sales, production, management}

  Constraints from subclasses:
  Manager = *AreaManager* $\cup$ *TopManager*
  Manager $\subseteq$ *Employee*
  AreaManager $\subseteq$ *Manager* and
  TopManager $\subseteq$ *Manager* (both redundant)

- an attribute is a set of pairs of (i) an instance and (ii) an element of a literal domain (constraint!)
  name =
  {(alice, "Alice"), (bob, "Bob"), (john, "John"), (mary, "Mary"), (tom, "Tom"), (larry, "Larry")}
  salary =
  {(alice, 70000), (bob, 60000), (john, 100000), (mary, 40000), (tom, 25000), (larry, 20000)}
  analogously for department names.

Semantics: Set theory (Cont'd)

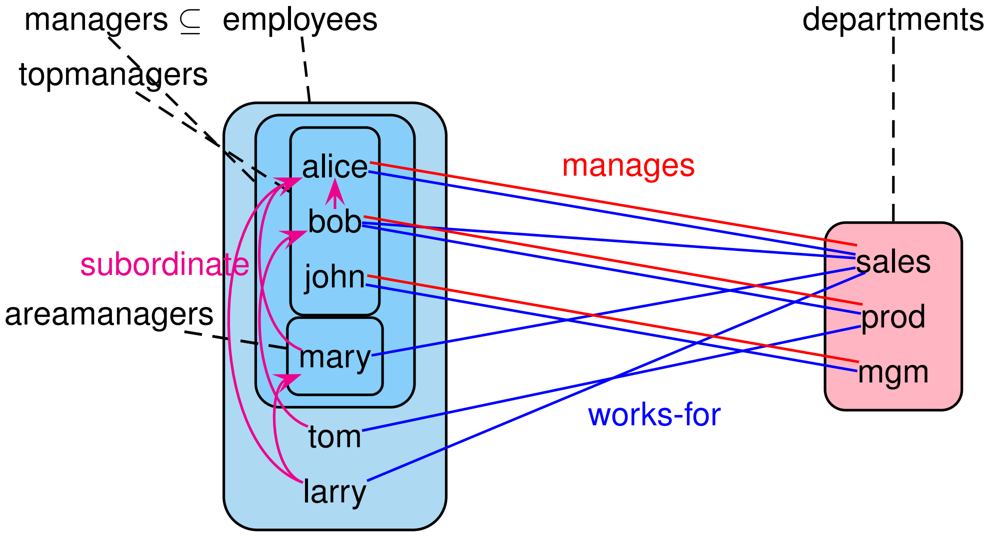
- a relationship is a set of pairs of instances,
  (or a set of n-tuples, in case of n-ary relationships)
  works-for $\subseteq$ *Employee* $\times$ *Dept*

  works-for = *{(alice, sales), (mary, sales), (larry, sales), (bob, production),*
  (bob, sales), (tom, production), (john, management)}

  manages $\subseteq$ *TopManager* $\times$ *Dept*
  manages = *{(alice, sales), (bob, production), (john, management)}*
  subordinate $\subseteq$ *Employee* $\times$ *Manager*
  subordinate = *{(mary, alice), (bob, alice), (larry, mary), (larry, alice), (tom, bob)}*

- not so obvious: constraints coming from the cardinality specifications, e.g.,
  the set of top managers is a subset of the things that manage exactly one department,
  the set of employees is a subset of the things that work for at least one and at most three departments
  (see later after the discussion of first-order logic)

Semantics: Set theory – Graphical Illustration



- does manages ⊆ works-for hold in general?
- subordinate ⊇ works-for ∘ manages^{-1} ?
- subordinate ⊇ (works-for \ manages) ∘ manages^{-1} !! ("\" denotes set difference)
- the subordinates relationship wrt. area managers (larry→mary) cannot be derived but must be stated explicitly.

Adequateness of (extended) ER Diagrams

An ontology should give a concise characterization of a domain and its constraints.

- classes, key constraints
- subclasses (specialization/generalization, disjointness)
- ranges and domains of properties (e.g., that the domain of "manages" is not all employees but only the managers)
- cardinalities
- is manages ⊂ works-for ?
- is manages ∩ works-for = ∅ ?
- subproperty constraints cannot be expressed
- further constraints (e.g., employees that work for a department are subordinate to the department's manager) cannot be expressed.

Exercise

- Discuss alternatives for the cardinalities for "subordinate".

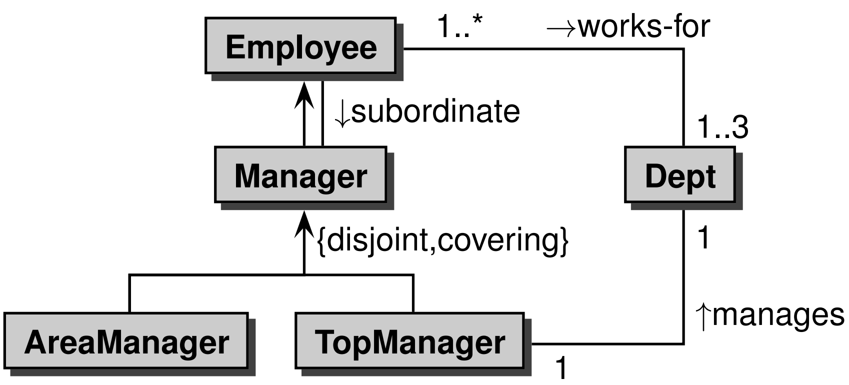
Alternative Semantics: Relational Model

Exercise: give a relational schema and the corresponding database state to the above ER diagram.

Relational Schema as an Ontology

- basically non-graphical, can be supported e.g. by dependency diagrams (cf. the Mondial documentation)
- no distinction between "classes" and "relationships"
- key constraints, foreign key/referential constraints
- keys/foreign key allow to guess classes vs. relationships
- cardinality "1": functional dependencies (adding n:1-relationships into a table like department/manages country/capital)
- no general cardinality constraints
- sometimes: domain constraints (by foreign keys)
- no further (inter- and intra-relation) constraints

2.2 Example: UML Class Diagram as an Ontology



common: UML class diagrams + OCL (object constraint language) constraints

OCL: first-order logic formulas associated one or more with UML elements (e.g.,

$$\text{subordinate} \supseteq \{(x,y) \mid \exists d : \text{works-for}(x,d) \wedge \text{works-for}(y,d) \wedge \text{manages}(y,d) \wedge x \neq y\}$$

associated with associations subordinate, works-for and manages).

2.3 Summary and Outlook

ONTOLOGY MODELING ASPECTS

- an ontology describes the notions of a domain.
  (Note: sometimes it also describes the individuals)
- descriptions in database context (ER, UML) interpreted as *constraints*, but can also be read as *definitions* (Logical Rules, e.g., definition of "subordinate").
- ER has a restricted expressiveness
- UML diagrams without OCL have a restricted expressiveness,
- UML with using OCL is as expressive as first-order logic.
- first-order logic: is too expressive (undecidable)

⇒ subsets of First-Order Logic as ontology formalisms
  - Derivation Rules: Horn subset of FOL
  - Description Logics and OWL (Web Ontology Language)

DATA MANAGEMENT: LOGICAL AND PHYSICAL DATA MODELS

- ER Model, UML Class Diagrams: conceptual models
  There are no existing "ER databases".
  UML specifications can be implemented in object-oriented databases.
- Relational model:
  - logical model, query languages (relational algebra, SQL)
  - unambiguous mapping from ER to relational model (incl. normalization of tables)
- XML:
  - assumed to be a logical model, but is between a logical and a physical model,
  - many ways how to map an ER model to an XML representation
    (there are many reasonable DTDs for representing the Mondial DB)
    ⇒ even though XML data exchange is easy, integration of XML data wrt. different DTDs to an agreed ER model can be troublesome.
  - XML Unicode serialization as a data exchange format
  - the "algebra" underlying XQuery is on a lower level than the relational algebra.
    (structural + content-oriented querying)

Comparison

- XML (mapping a domain to a tree structure) is less "semantic" than the relational model (relations for entity types and for relationship types, keys/foreign keys)
- [aside: even the historical network database model (see history part of the SSD/XML lecture) is more semantic than XML]

"Dirty" Points

- XML: subelement relationship has no "name":
  country/city: subelement means "located in", while
  city/population: subelement means: has property
- relational model: when n:1 relationships are mapped into a relation (cf. country/capital), the foreign key means "attribute is not a property but a relationship"
  Less dirty, since explicit in the schema.

Network-like models with nodes (objects) and edges (relationships) are more "semantical"

[Aside: "semantic networks" in early approaches to knowledge representation models like KL-ONE etc.]

Outlook: RDF Data Model

- RDF "Resource Description Framework" as a graph-based *logical* data model,
- close to the idea of an "ER database"
  - "things" (graph: nodes) belong to classes/entity types and have an identification (URI - Uniform Resource Identifier)
  - they have attributes (with literal values)
  - there are (binary) relationships (graph: edges). Relationships have names.
- a *node-labeled* and *edge-labeled* data model,
- graphical representation,
- syntactical representations (in plain Unicode, or in an XML(-Unicode) representation of the graph),
- and an SQL-style relational-flavor query language,
- and some other query languages (e.g., again path-based)
  (obvious dialects of OEM/MSL/Lorel and F-Logic apply)

Development of Languages etc.

Shown in the "history" part of the SSD/XML lecture

XML/XPath/XQuery was influenced by many earlier concepts:

- XML: Network database model, object-oriented database model, earlier self-describing semistructured models (OEM, F-Logic), SGML
- ASCII/Unicode representation/data exchange: ODMG's OIF (object interchange format)
- XPath: C++ and OQL path expressions (with UNIX notation), F-Logic (conditions embedded into path expressions)
- XQuery: SQL/OQL (clause-based language, but not on a "simple" algebraic foundation)
- not very much theory (except things like tree automata that deal mainly with structural things)
  (recall that there was no theory part in the SSD/XML lecture)

RDF: even more influenced by earlier concepts

- semantic networks: resources (objects and classes), properties and relationships
- relational model/representation: algebra, algebraic query formalisms
- graph model: navigation-based expressions (formally: based on semijoins)
- several Unicode representations
- but: unordered – not suitable for documents.
- in line with *ontology formalisms* for metadata: RDF-RDFS-OWL
- metadata: means logic-based *model theory* with *reasoning*
  (note: constraints etc. in the relational model are also a simple form of model theory)
  - RDF: no reasoning. Data is just data.
    The world is expressed in a single table with only three columns!
  - RDFS: some restricted reasoning
  - OWL Lite/OWL DL/OWL Full: up to undecidability.