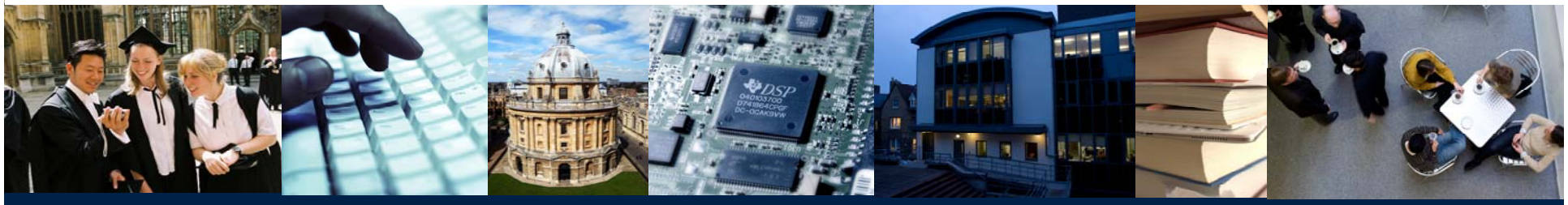
DEPARTMENT OF
COMPUTER
SCIENCE

UNIVERSITY OF
OXFORD



OBDA: Theory and Practice

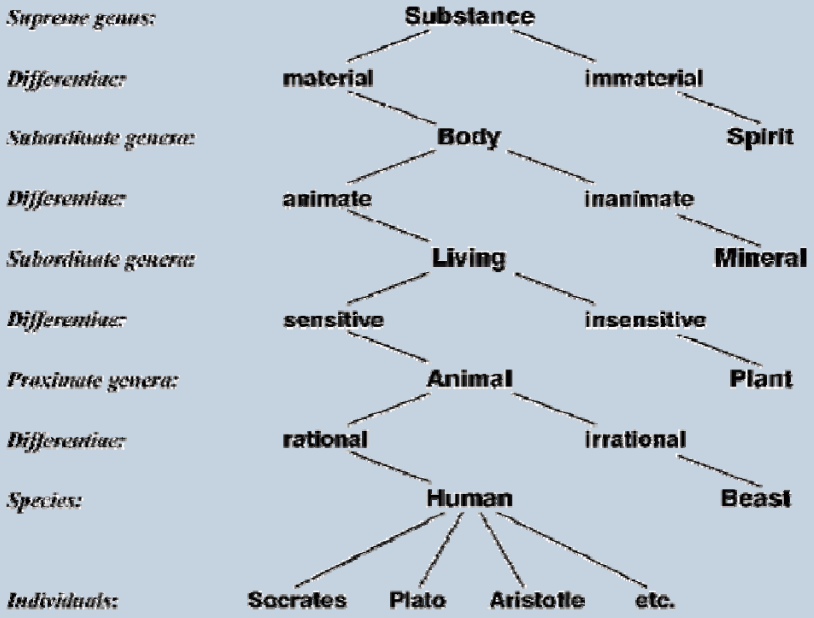
Ian Horrocks
Information Systems Group

What is an Ontology?

What is an Ontology?

A fundamental branch of **metaphysics**

- Studies “being” or “existence” and their **basic categories**
- Aims to find out what **entities** and **types of entities** exist



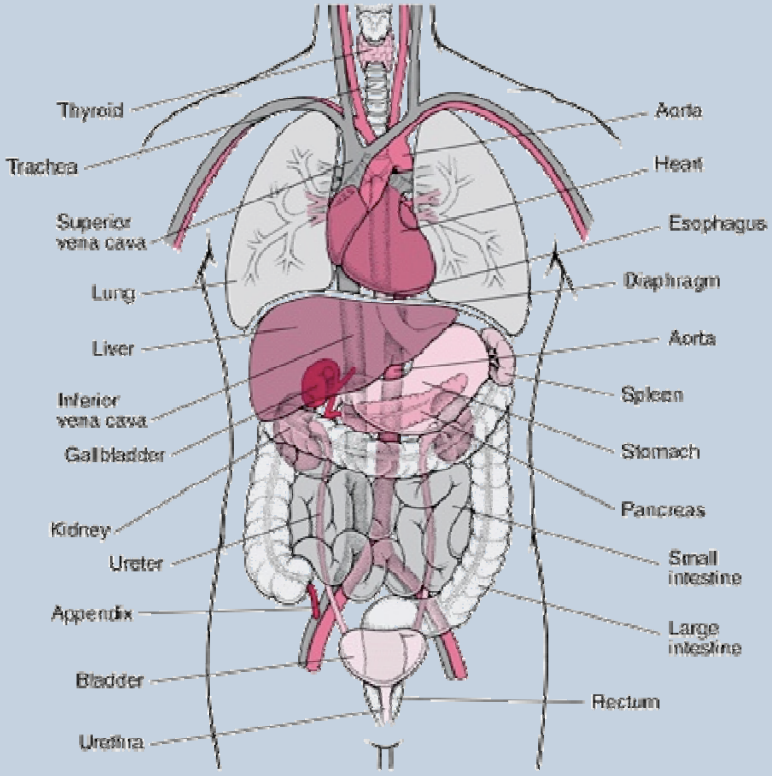
What is an Ontology?

A conceptual model of (some aspect of) the world

What is an Ontology?

A conceptual model of (some aspect of) the world

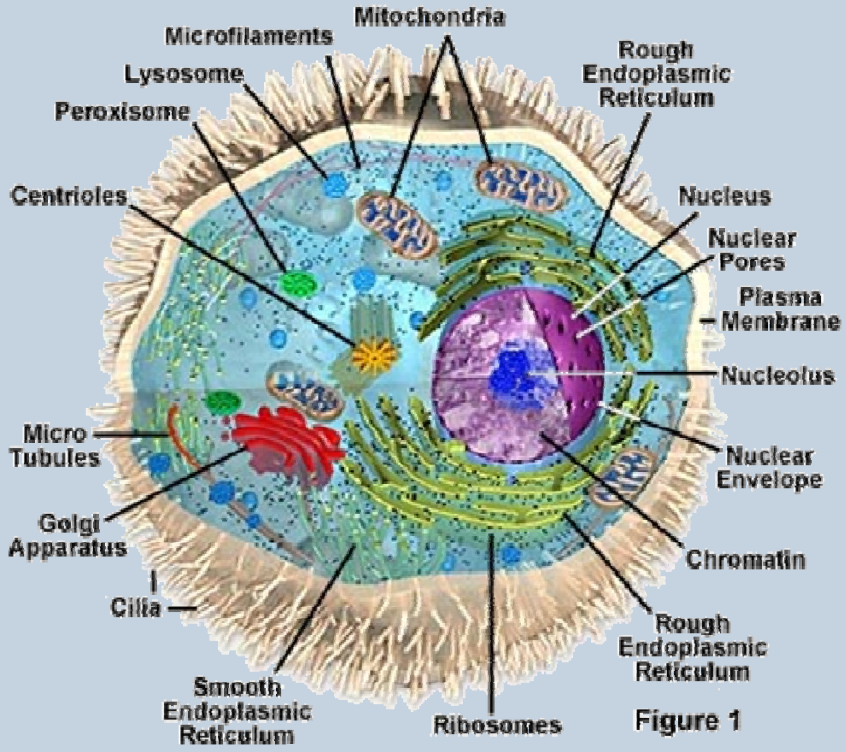
- Introduces **vocabulary** relevant to domain, e.g.:
  - Anatomy



What is an Ontology?

A conceptual model of (some aspect of) the world

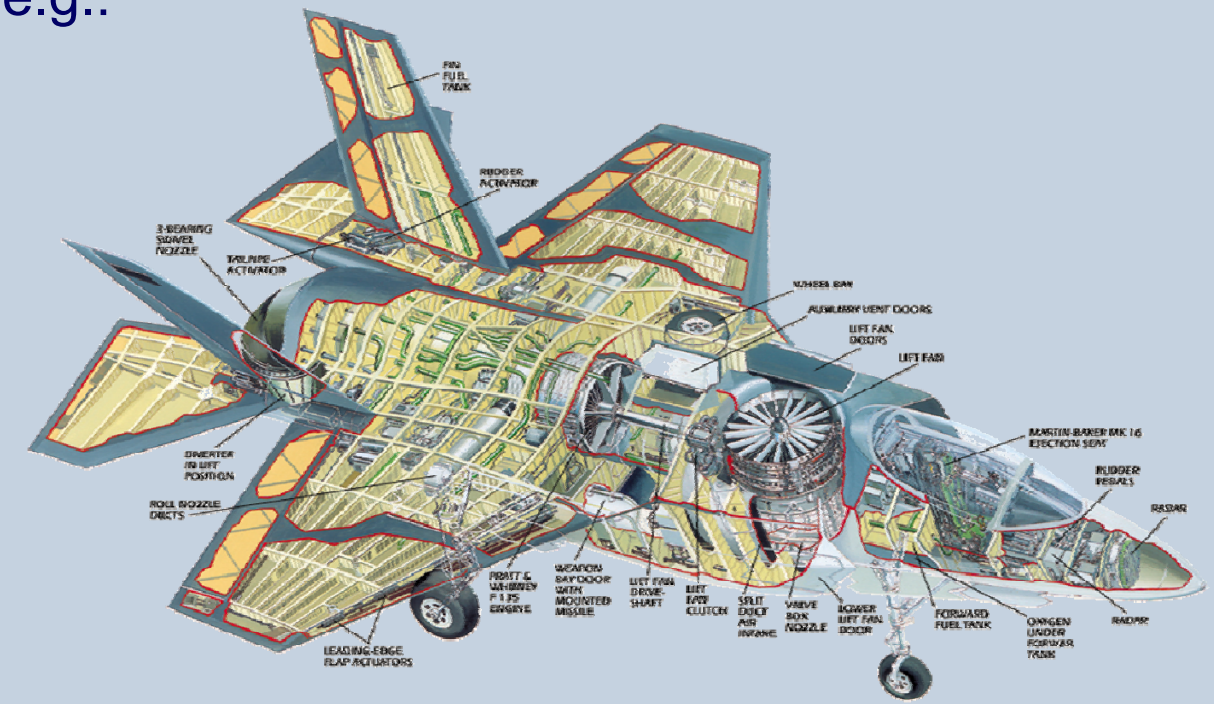
- Introduces **vocabulary** relevant to domain, e.g.:
  - Anatomy
  - Cellular biology



What is an Ontology?

A conceptual model of (some aspect of) the world

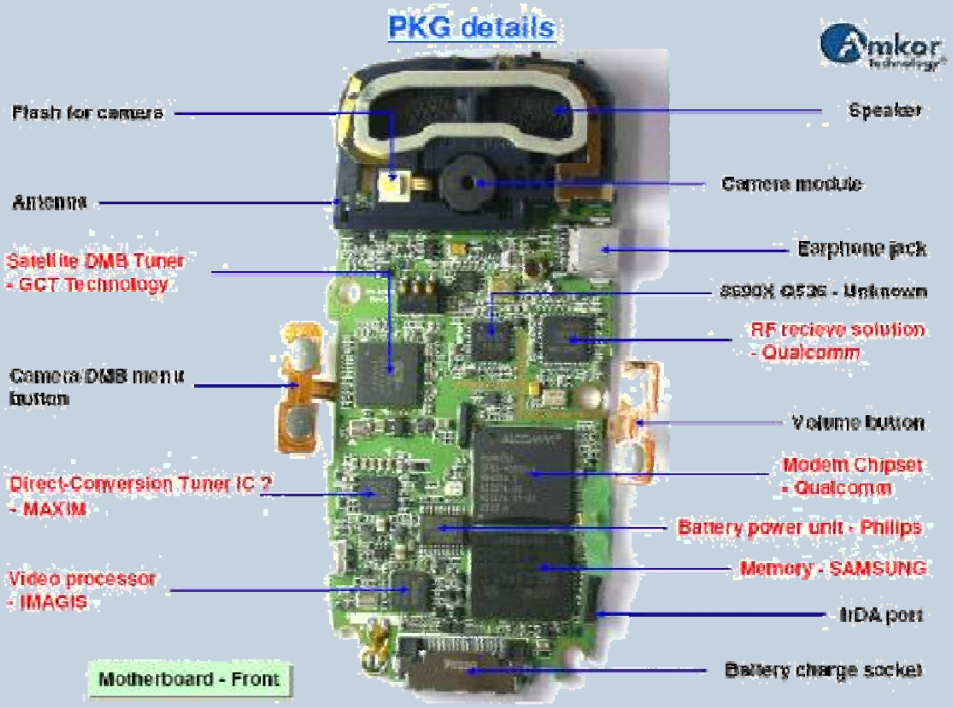
- Introduces **vocabulary** relevant to domain, e.g.:
  - Anatomy
  - Cellular biology
  - Aerospace



What is an Ontology?

A conceptual model of (some aspect of) the world

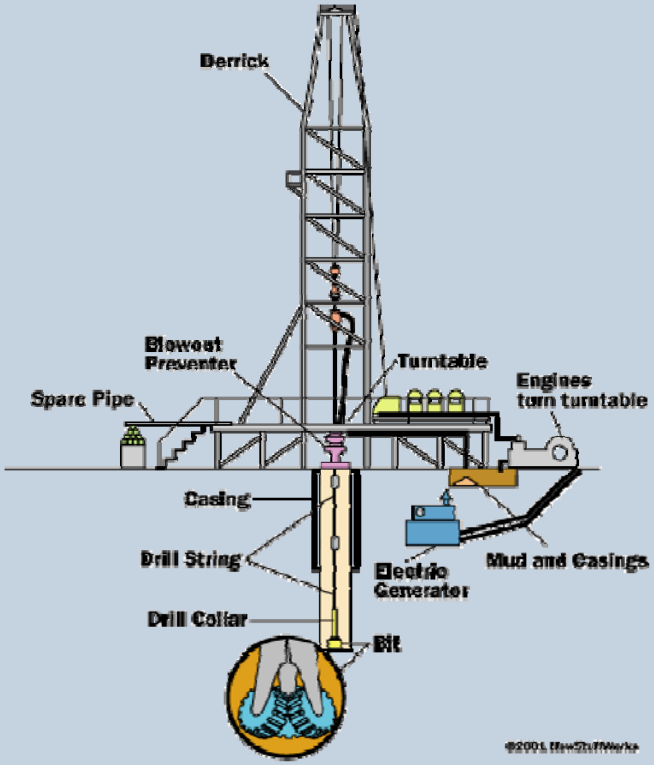
- Introduces **vocabulary** relevant to domain, e.g.:
  - Anatomy
  - Cellular biology
  - Aerospace
  - Cell Phones



What is an Ontology?

A conceptual model of (some aspect of) the world

- Introduces **vocabulary** relevant to domain, e.g.:
  - Anatomy
  - Cellular biology
  - Aerospace
  - Cell Phones
  - Oil and gas

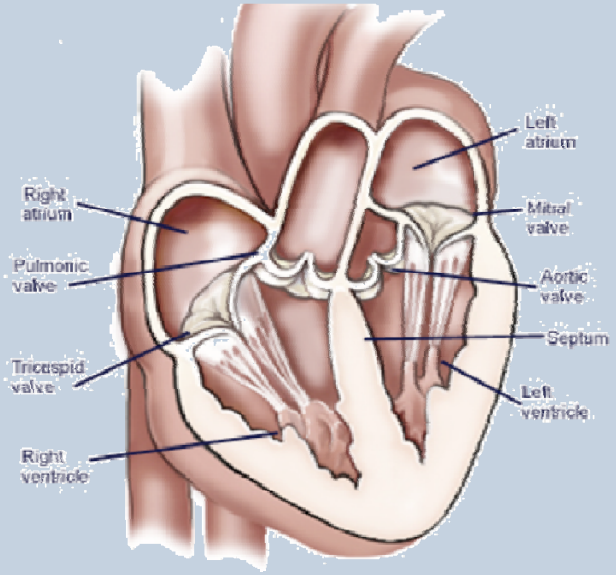


What is an Ontology?

A conceptual model of (some aspect of) the world

- Introduces **vocabulary** relevant to domain
- Specifies **meaning** (semantics) of terms

Heart is a muscular organ that is part of the circulatory system



What is an Ontology?

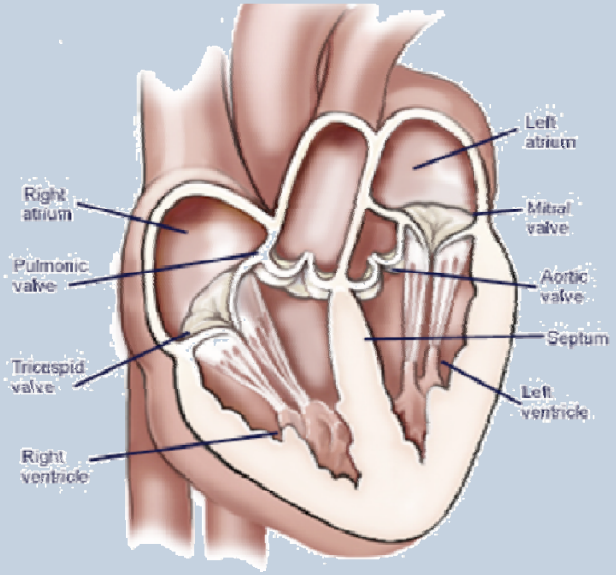
A conceptual model of (some aspect of) the world

- Introduces **vocabulary** relevant to domain
- Specifies **meaning** (semantics) of terms

  Heart is a muscular organ that is part of the circulatory system

- **Formalised** using suitable logic

$$\forall x.[\mathsf{Heart}(x) \rightarrow \mathsf{MuscularOrgan}(x) \wedge \exists y.[\mathsf{isPartOf}(x,y) \wedge \mathsf{CirculatorySystem}(y)]]$$



Applications: HCLS

- **OBO foundry** includes more than 100 biological and biomedical ontologies
- **Siemens** “actively building OWL based clinical solutions”
- **SNOMED-CT** (Clinical Terms) ontology
  - used in healthcare systems of more than 15 countries, including Australia, Canada, Denmark, Spain, Sweden and the UK
  - also used by major US providers, e.g., Kaiser Permanente
  - ontology provides common vocabulary for recording clinical data

Applications: Energy Supply Industry

- **EDF Energy** offer personalised energy saving advice to every customer

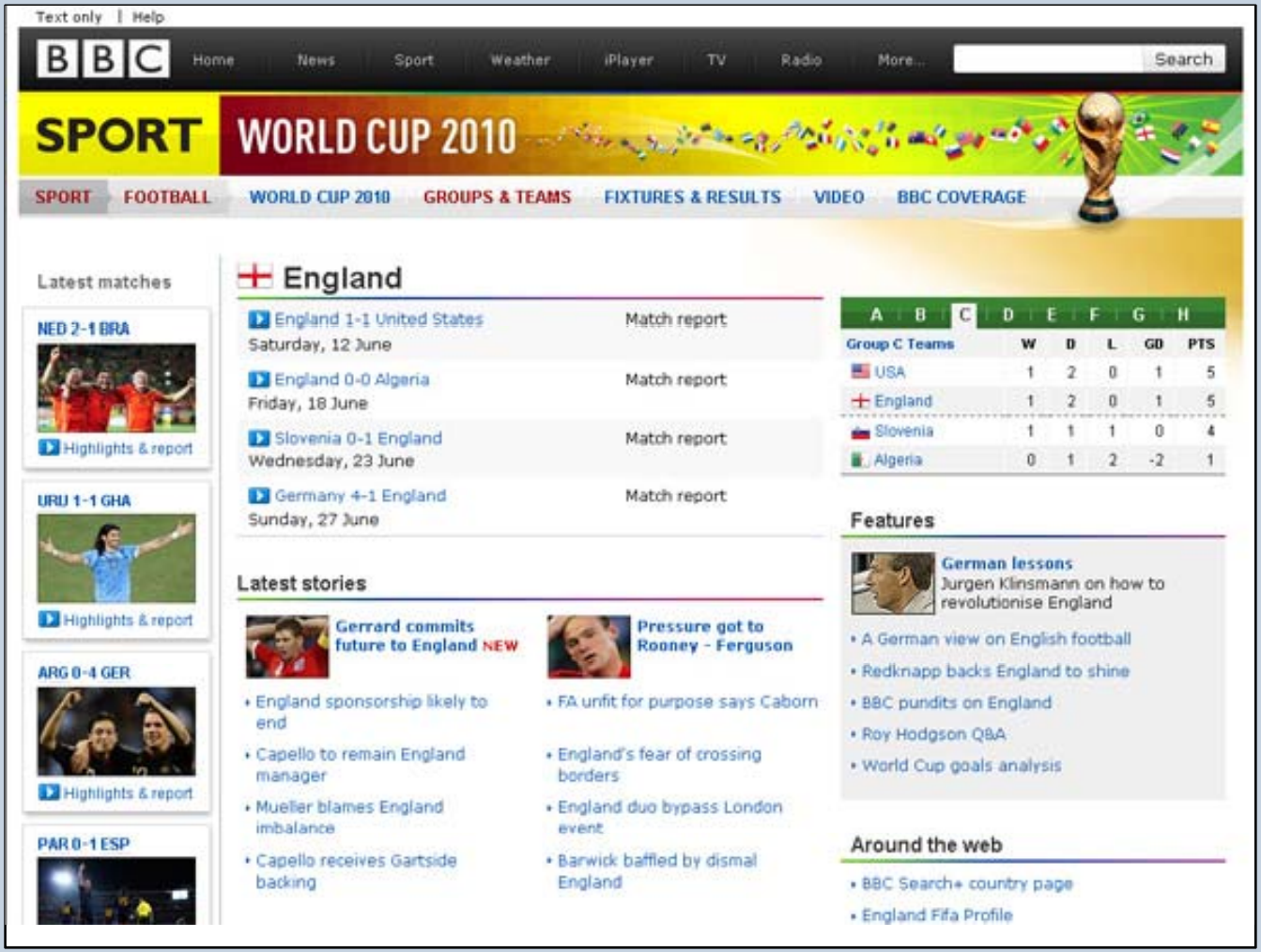
- **OWL ontology** used to model relevant environmental factors

- **Oxford's HermiT reasoner** used to match customer circumstances with relevant pieces of advice

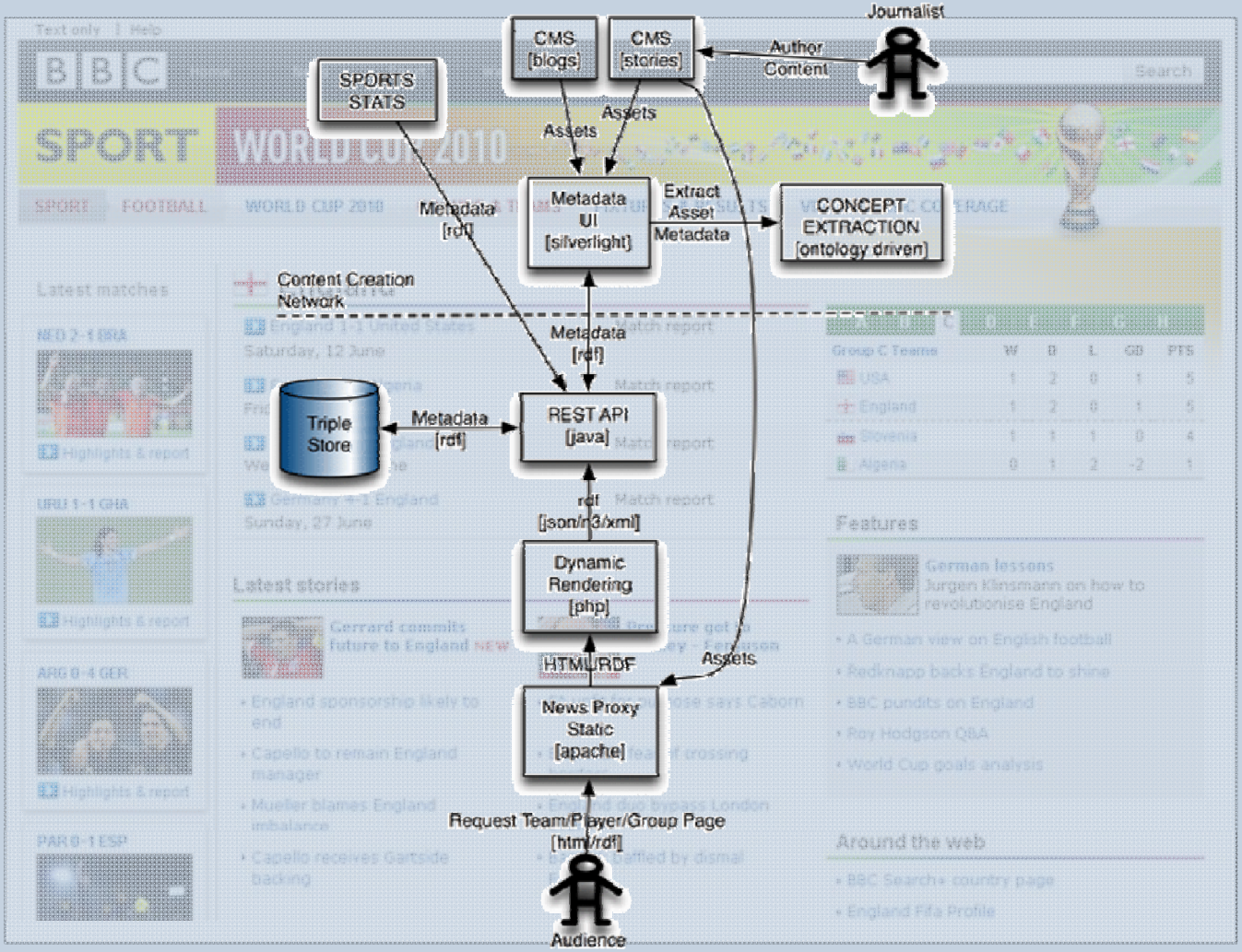
Applications: Intelligent Mobile Platform

- **Samsung** developing Intelligent Moblile Platform to support context-aware applications
- IMP monitors environment via **sensor data** (GPS, compass, accelerometer, ...)
- **OWL ontology** used to model environment and **infer context** (e.g., coffee with friends)
- Applications exploit context to enable more **intelligent behaviour**

Applications: Semantic Web



Applications: Semantic Web



DEPARTMENT OF COMPUTER SCIENCE

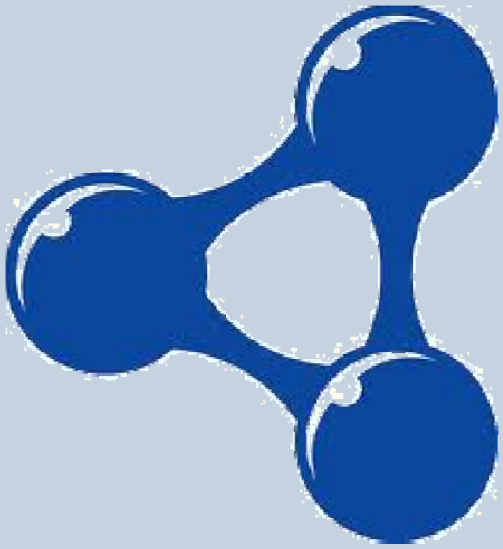
Optique

SEVENTH FRAMEWORK PROGRAMME

Applications: Semantic Web

Semantic Technology

Languages



Semantic Technology

Languages, Infrastructure


Hermit

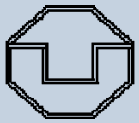
FaCT++

ORACLE

Racer

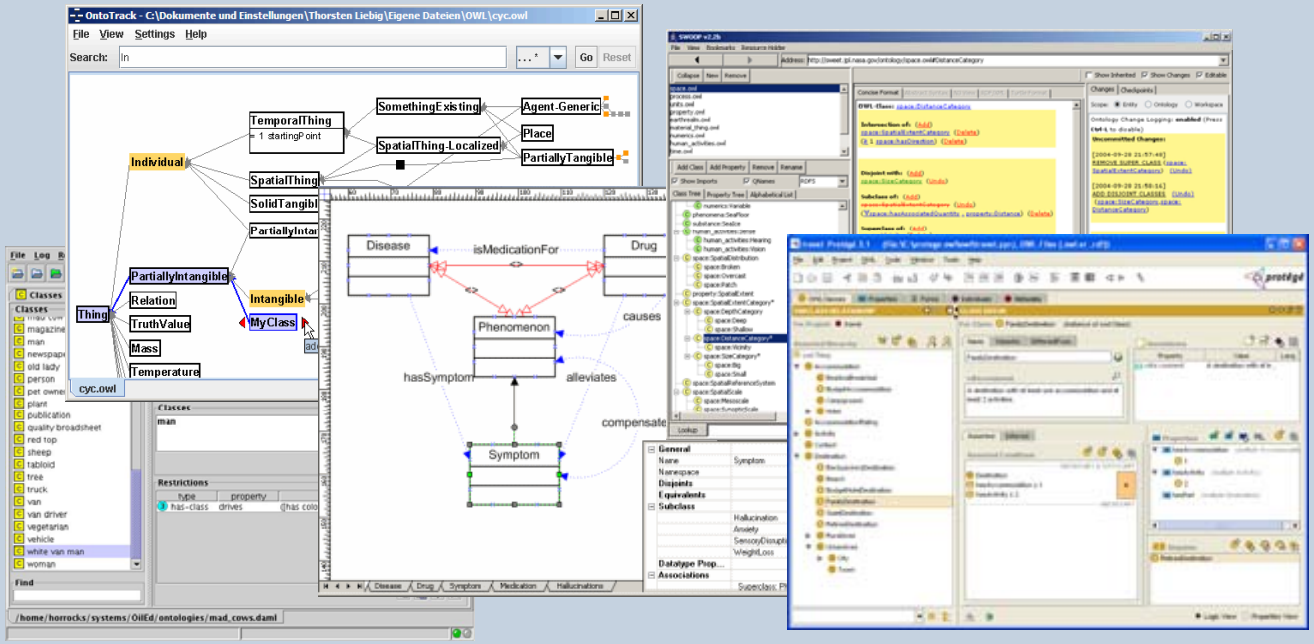
pellet


KAON2


CEL

Semantic Technology

Languages, Infrastructure & Tools



Applications: **OBDA**

Optique **objectives:**

- Provide semantic end-to-end connection between users and data sources
- Enable users to rapidly formulate intuitive queries using familiar vocabularies and conceptualisations

Applications: **OBDA**

Optique **objectives:**

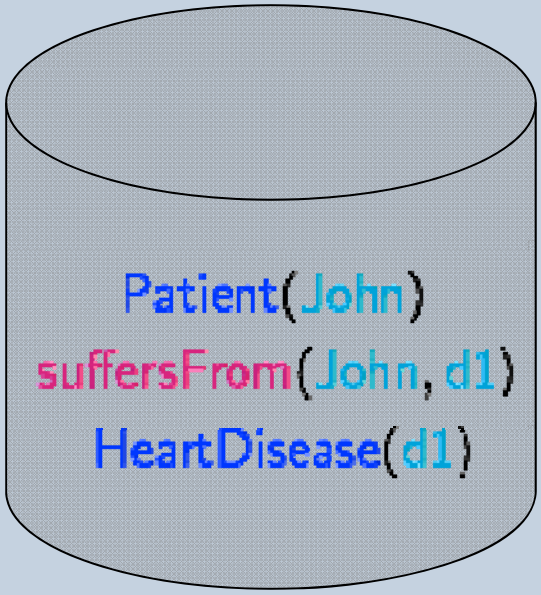
- Provide semantic end-to-end connection between users and data sources
- Enable users to rapidly formulate intuitive queries using familiar vocabularies and conceptualisations

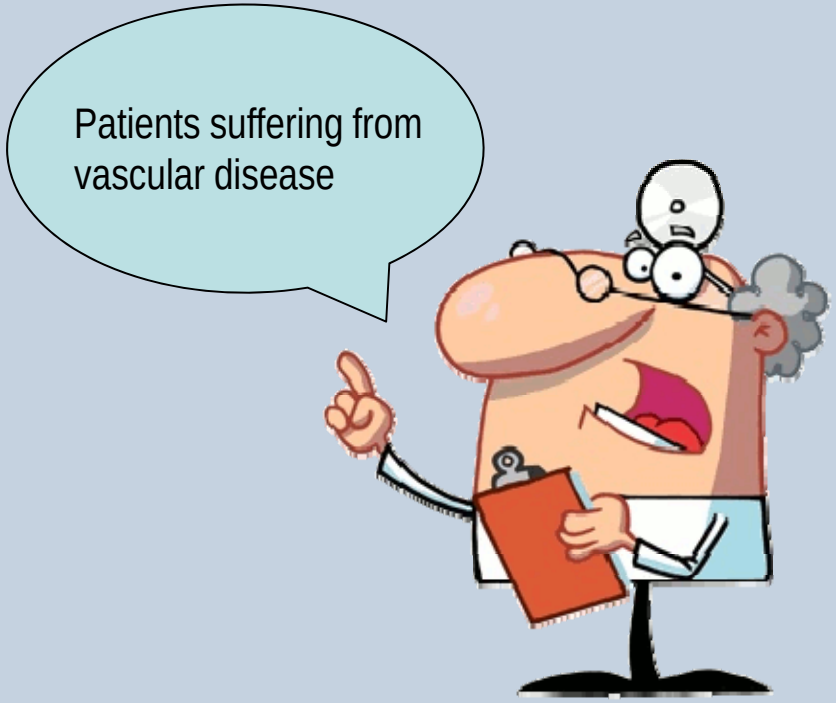
Applications: **OBDA**

Optique **objectives:**

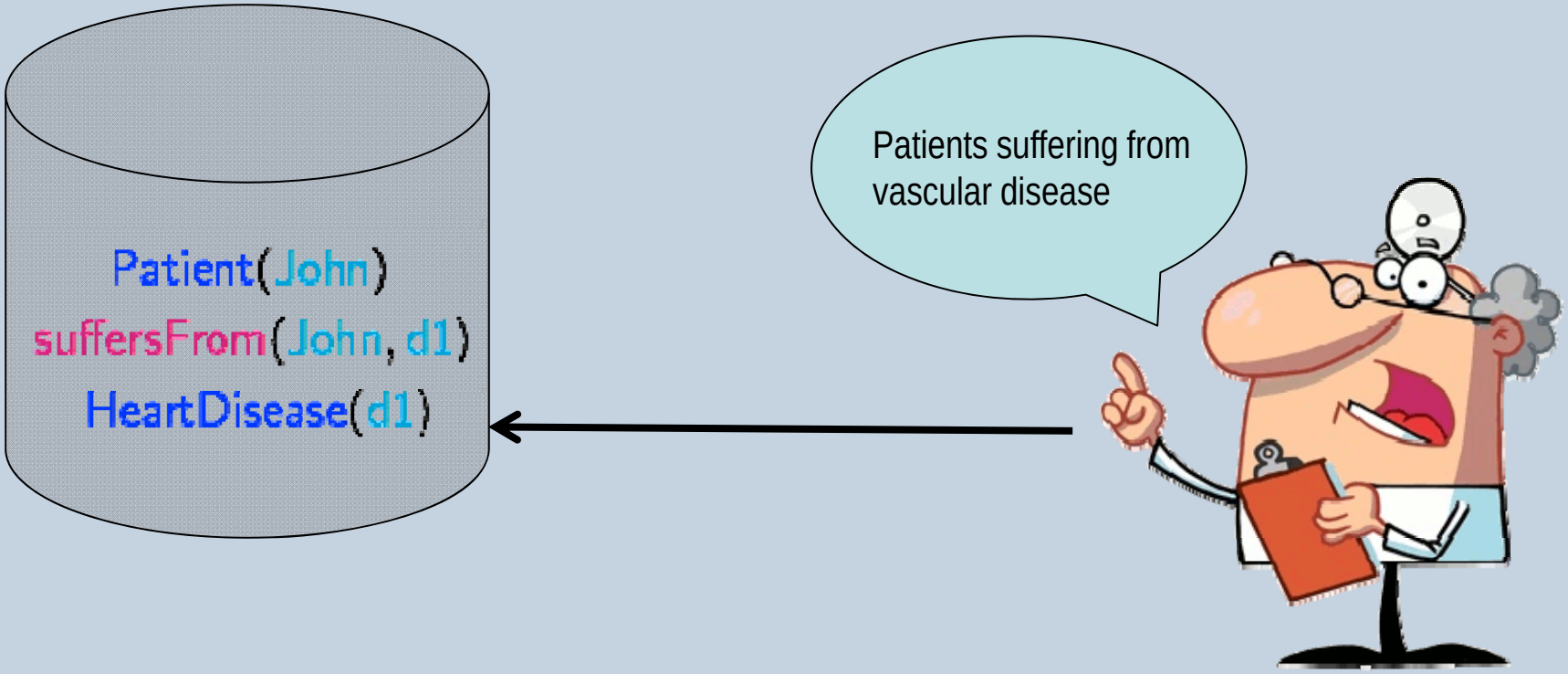
- Provide semantic end-to-end connection between users and data sources
- Enable users to rapidly formulate intuitive queries using familiar vocabularies and conceptualisations

Applications: **OBDA**

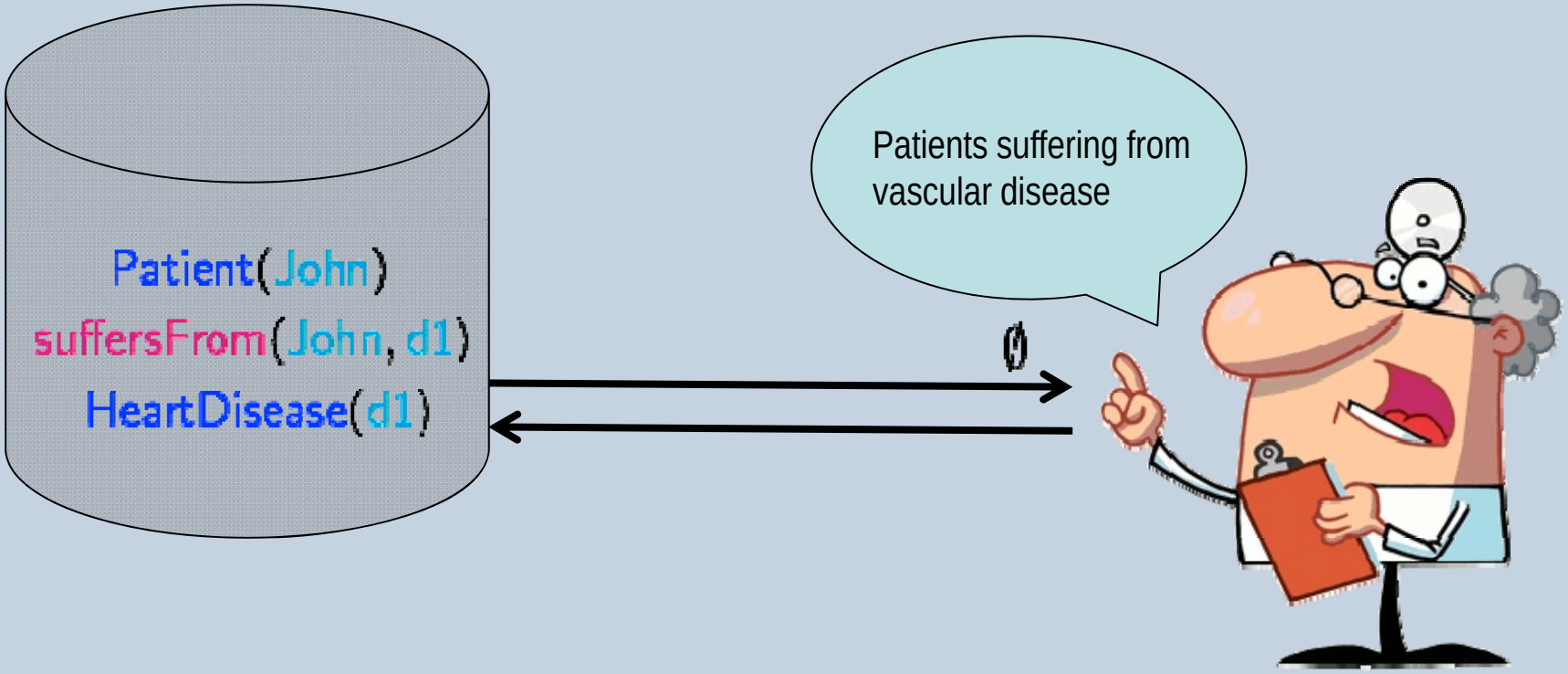




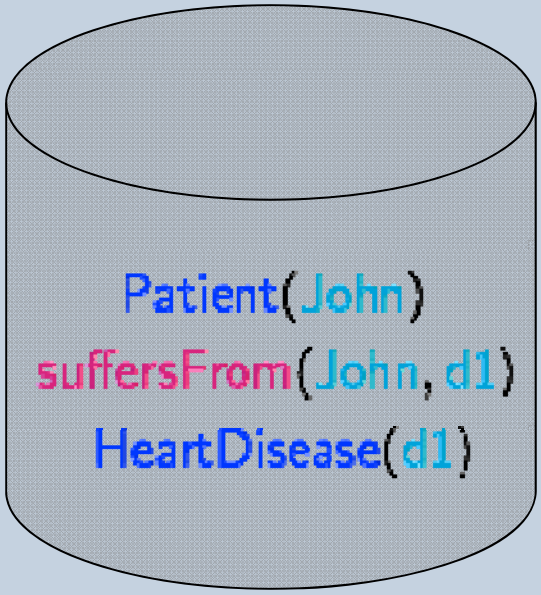
Applications: **OBDA**



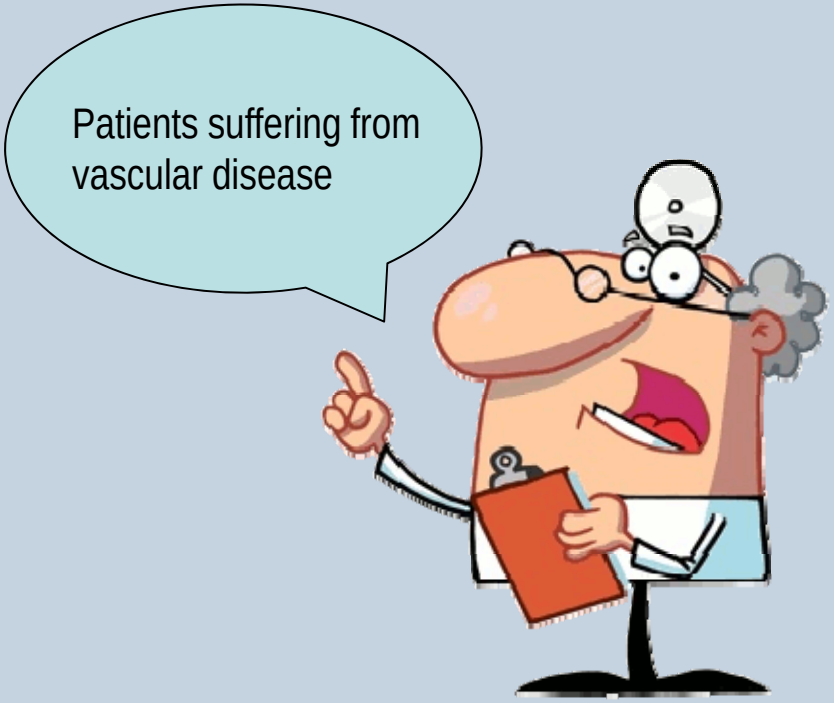
Applications: **OBDA**



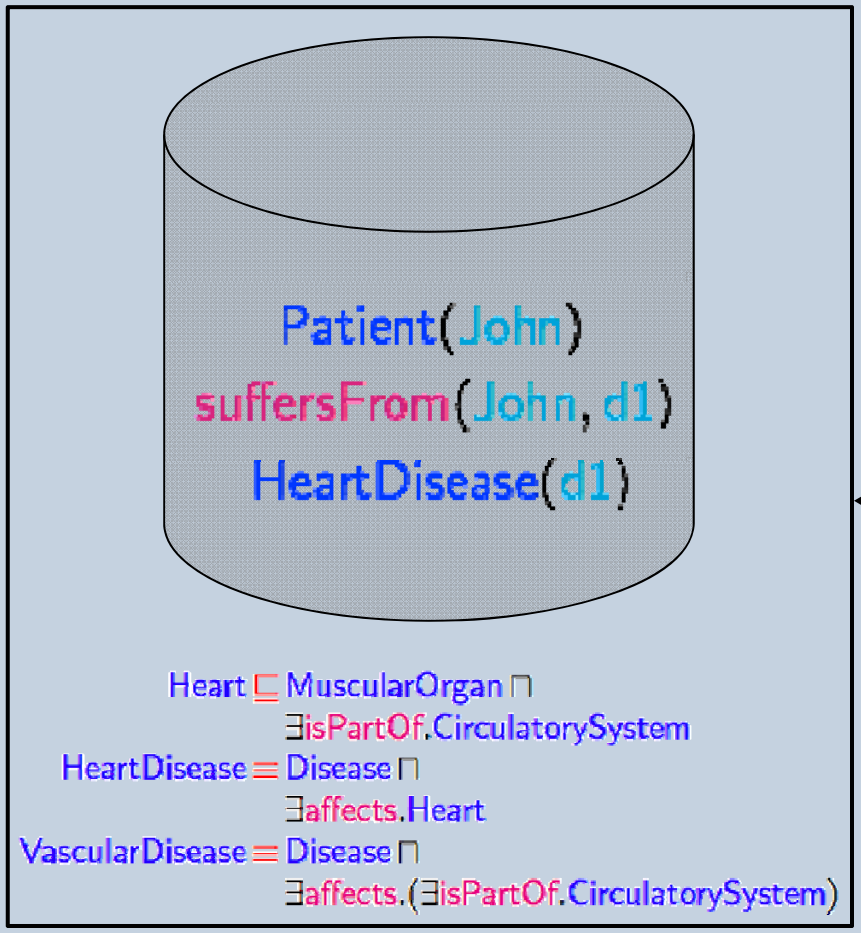
Applications: **OBDA**



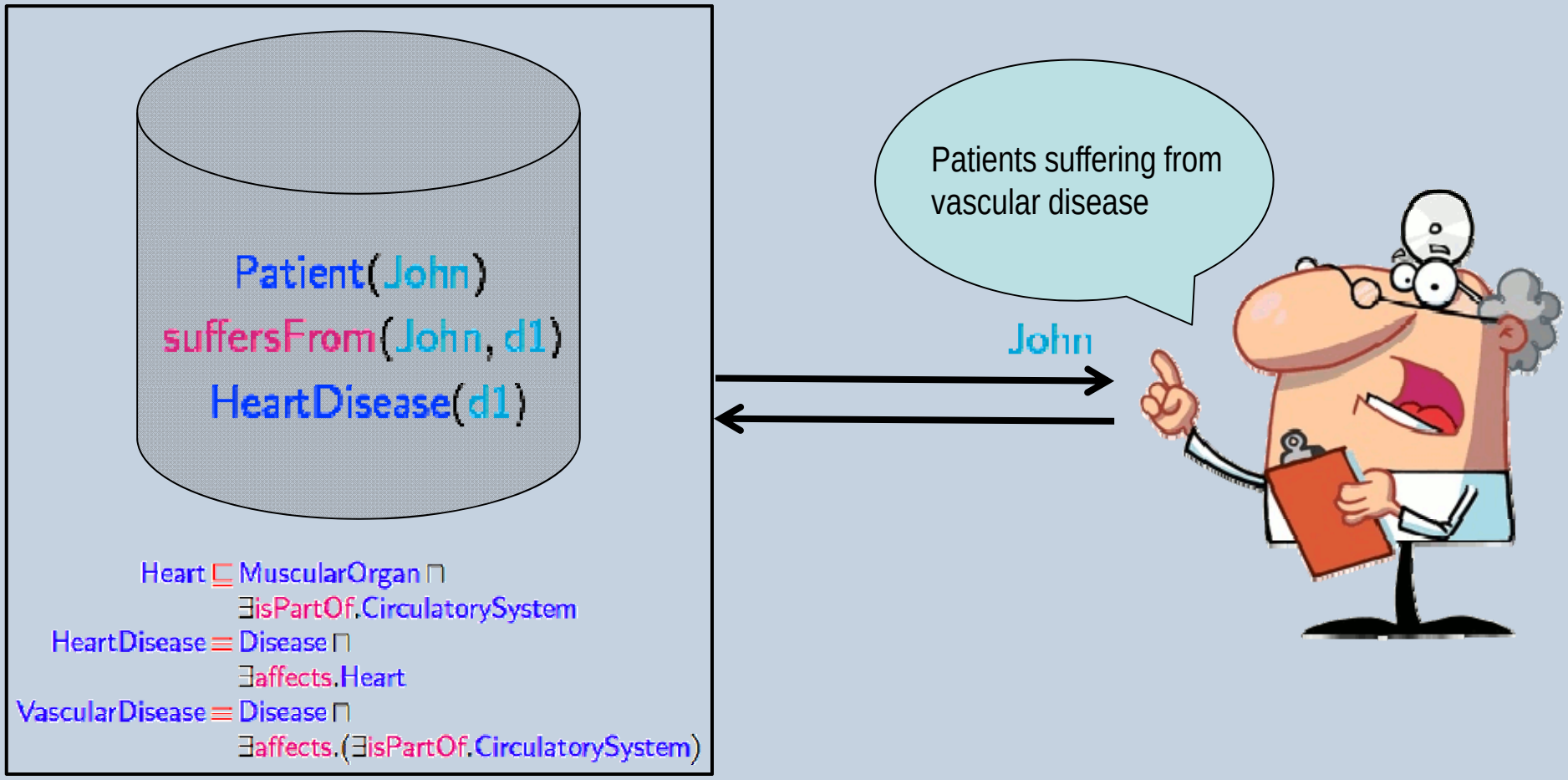
$$\begin{aligned} \text{Heart} &\sqsubseteq \text{MuscularOrgan} \sqcap \\ &\quad \exists \text{isPartOf}.\text{CirculatorySystem} \\ \text{HeartDisease} &\equiv \text{Disease} \sqcap \\ &\quad \exists \text{affects}.\text{Heart} \\ \text{VascularDisease} &\equiv \text{Disease} \sqcap \\ &\quad \exists \text{affects}.(\exists \text{isPartOf}.\text{CirculatorySystem}) \end{aligned}$$



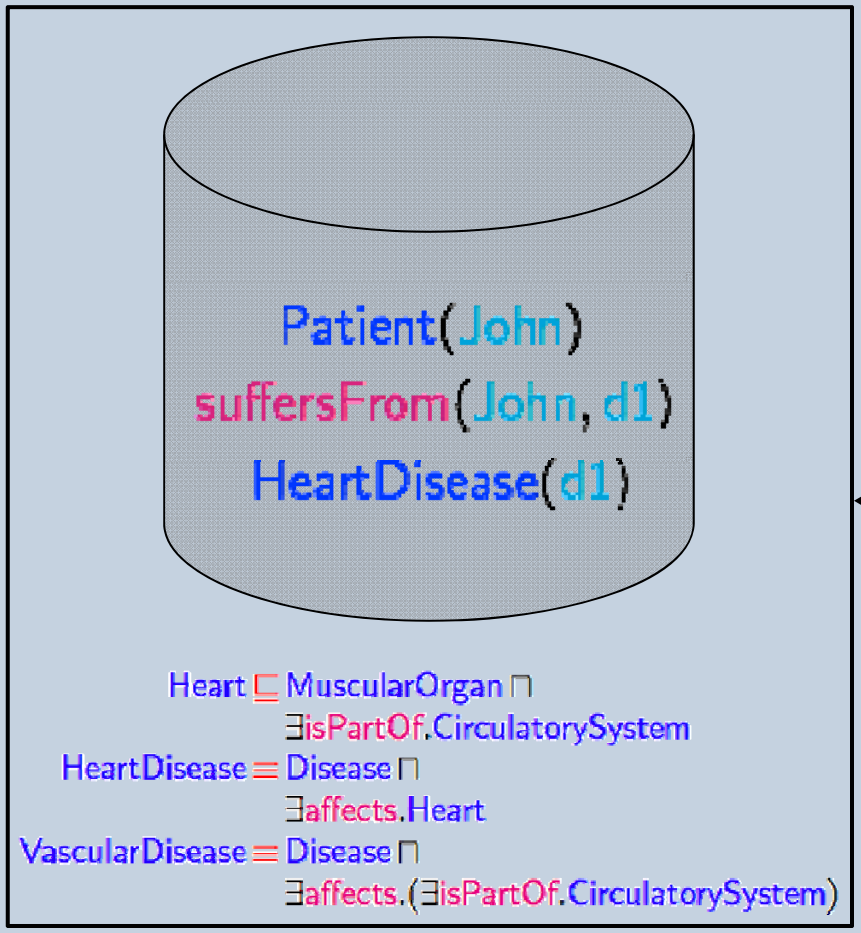
Applications: **OBDA**



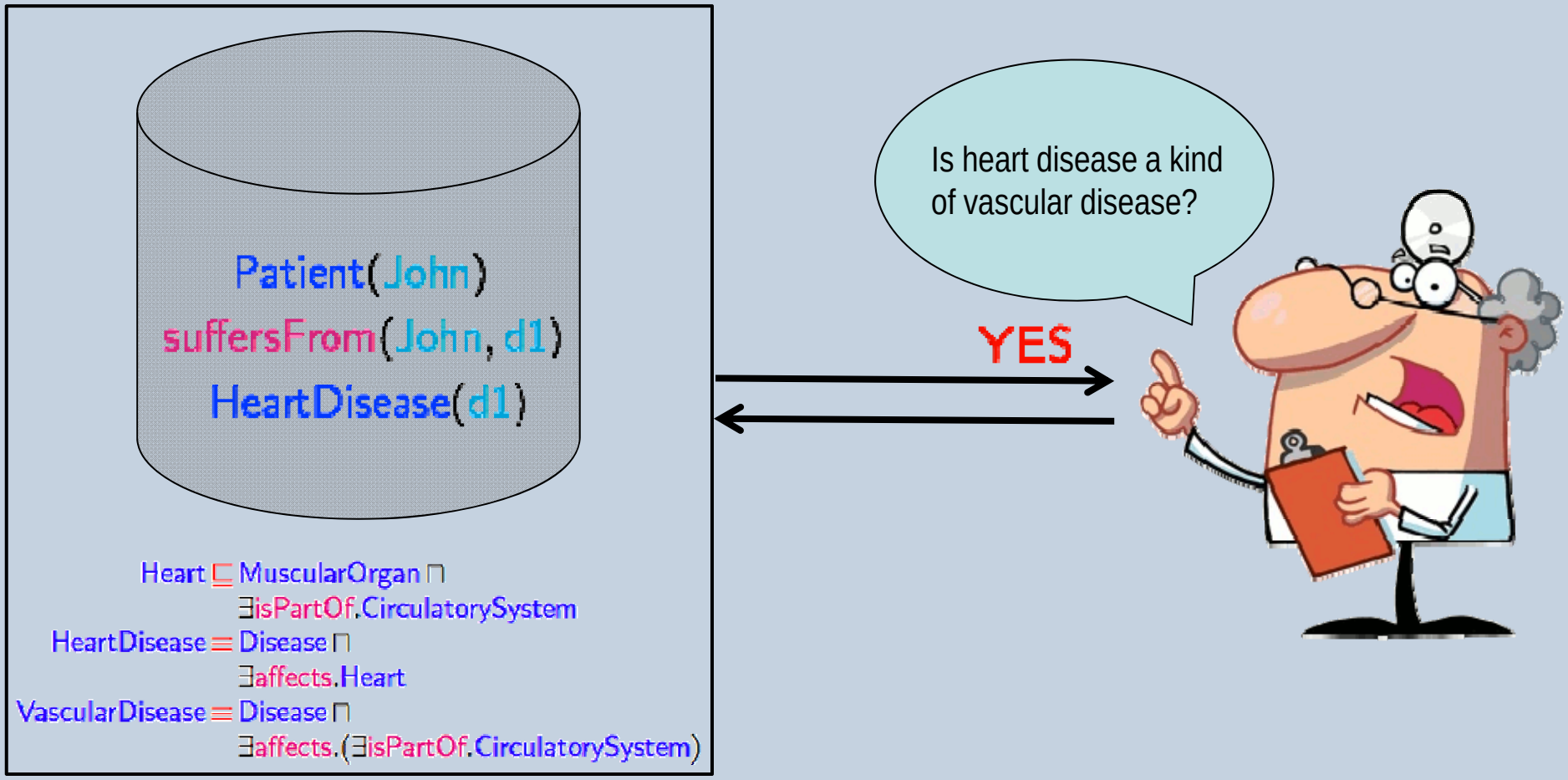
Applications: **OBDA**



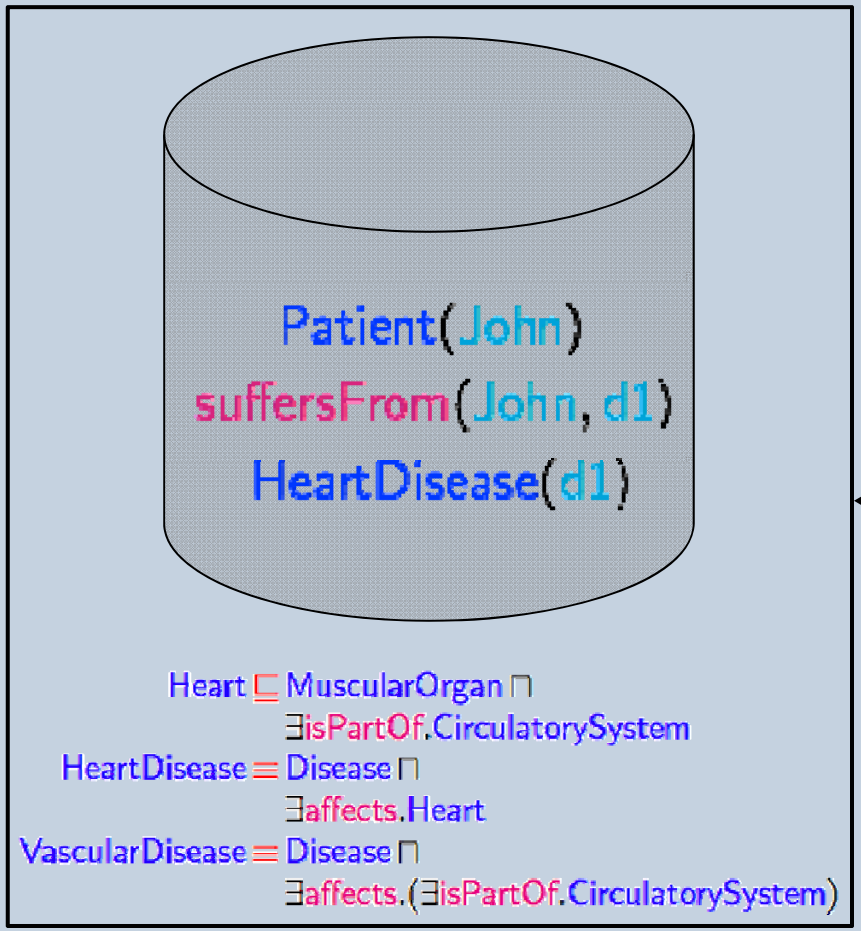
Applications: **OBDA**



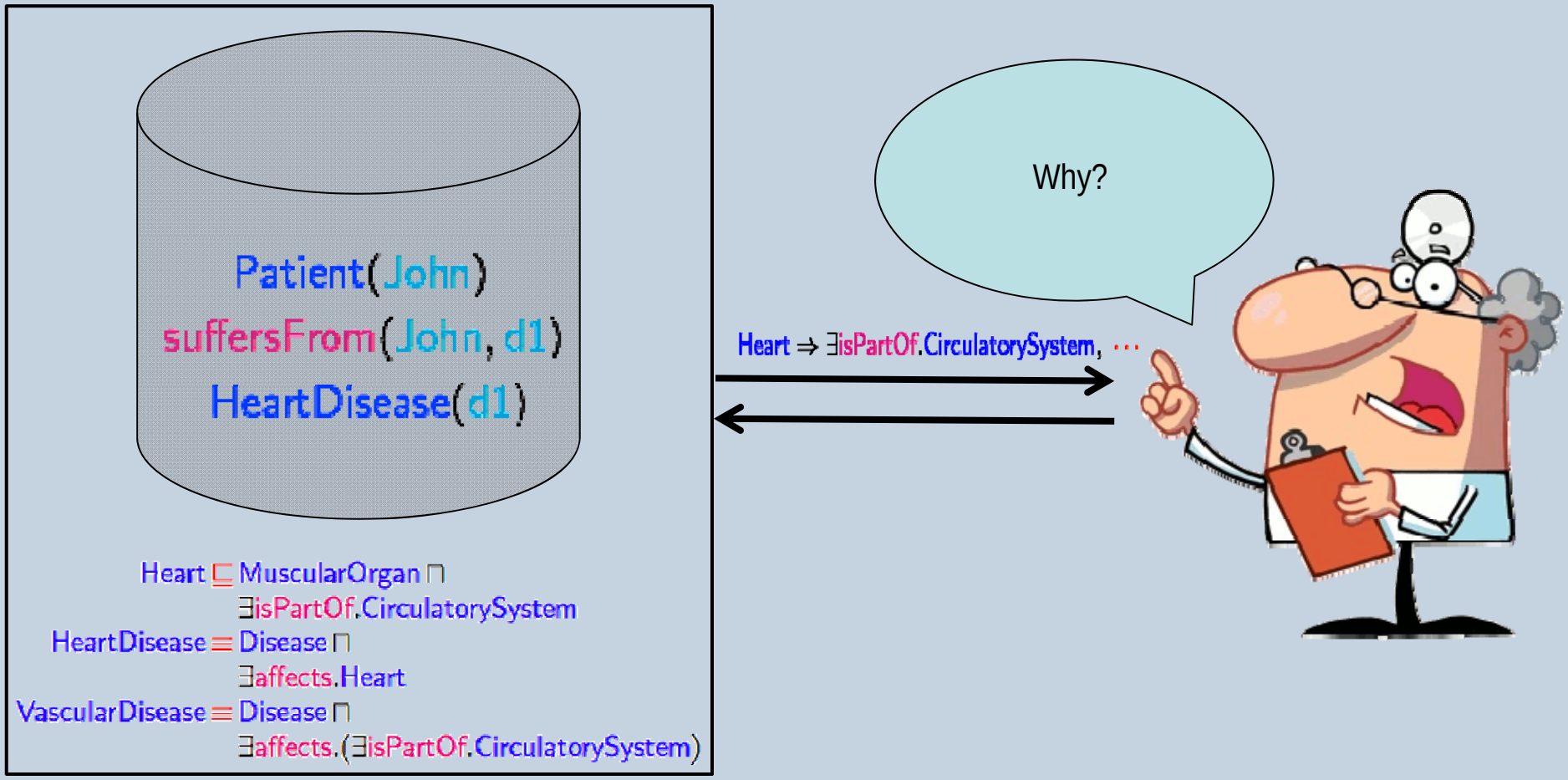
Applications: **OBDA**



Applications: **OBDA**



Applications: **OBDA**



OBDA: Practical Issues

Large datasets → conflicting requirements

- Modeling complex application domains requires:
  Rich ontology languages
- Fine-grained information access requires:
  Powerful query languages
- Applications (often) require:
  Predictable performance (correct and timely answers)

Various Approaches — Different Tradeoffs

Use full power of OWL and complete reasoner

input: any OWL ontology, dataset and query

output: sound and complete query answer

Use full power of OWL and incomplete reasoner

input: any OWL ontology, dataset and query

output: sound but (possibly) incomplete query answer

❸ Use a suitable "profile" and specialised reasoner:

input: restricted OWL ontology; any dataset and query

output: sound and complete query answer

Various Approaches — Different Tradeoffs

Use full power of OWL and complete reasoner:

- ✓ Well-suited for modeling complex domains
- ✓ Reliable answers
- ✗ High worst-case complexity
- ✗ Scalability problems for large datasets

Complete ontology reasoners:

- E.g., FaCT++, **HermiT**, Pellet, ...
- Based on (hyper)tableau (model construction) theorem provers
- Proof needed for each answer tuple
- Unlikely to scale to very large datasets

Various Approaches — Different Tradeoffs

Use full power of OWL and incomplete reasoner:

- ✓ Well-suited for modeling complex domains
- ✓ Favourable scalability properties (for query answering)
- ✗ Incomplete answers (and degree of incompleteness not known)
- ✗ Materialisation based techniques assume “static” RDF data

Incomplete ontology reasoners:

- E.g., Oracle’s Semantic Datastore, Sesame, Jena, OWLim, ...
- Based on RDF triple stores and deductive DB technologies
- Widely used in practice to reason with large datasets

Materialisation

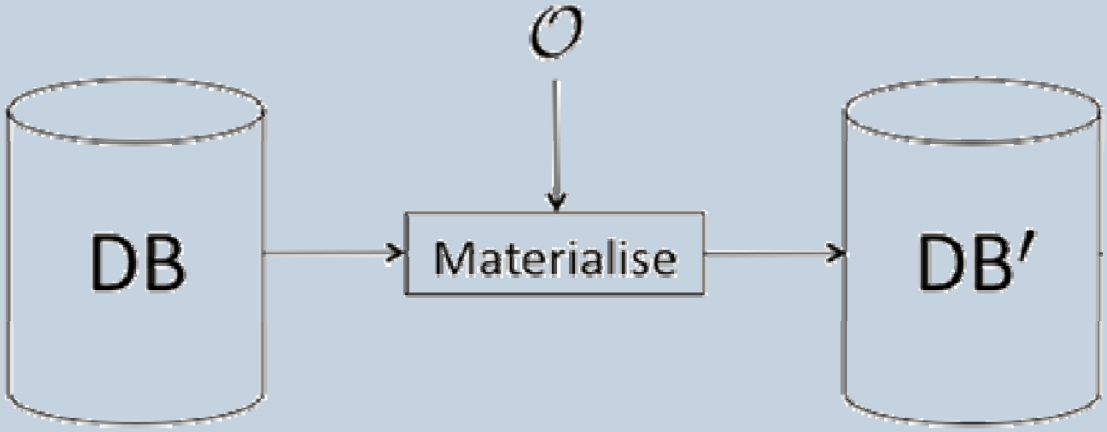
Given (RDF) data DB, ontology $\mathcal{O}$ and query $\mathcal{Q}$:

Materialisation

Given (RDF) data DB, ontology $\mathcal{O}$ and query $\mathcal{Q}$:

- **Materialise** (RDF) data DB $\rightarrow$ DB′ s.t. evaluating $\mathcal{Q}$ w.r.t. DB′ equivalent to answering $\mathcal{Q}$ w.r.t. DB and $\mathcal{O}$

  nb: Closely related to **chase** procedure used with DB dependencies



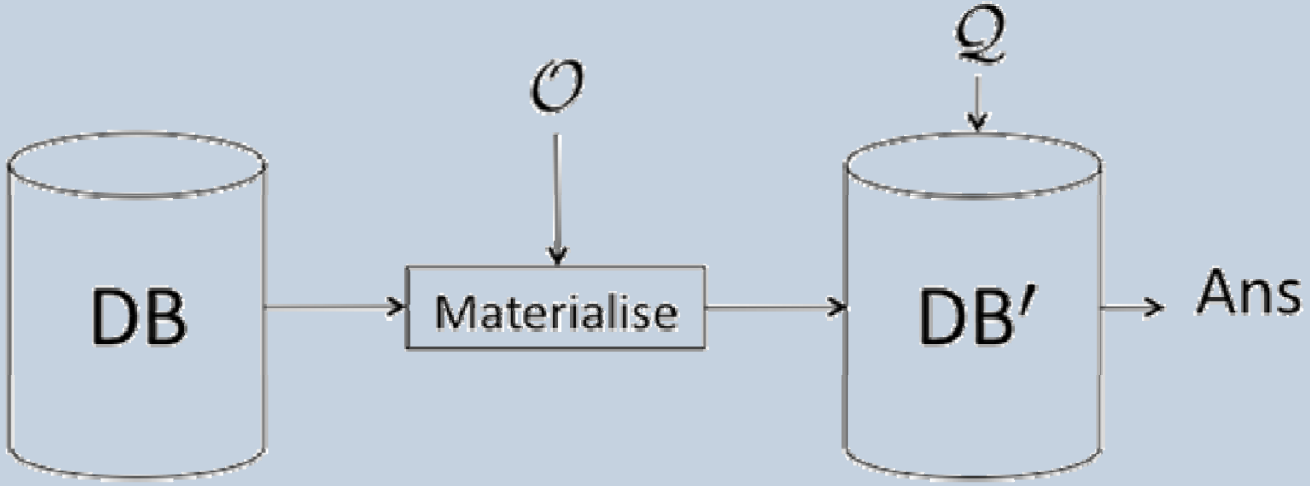
Materialisation

Given (RDF) data DB, ontology $\mathcal{O}$ and query $\mathcal{Q}$:

- **Materialise** (RDF) data DB $\rightarrow$ DB′ s.t. evaluating $\mathcal{Q}$ w.r.t. DB′ equivalent to answering $\mathcal{Q}$ w.r.t. DB and $\mathcal{O}$

  nb: Closely related to **chase** procedure used with DB dependencies

- **Evaluate** $\mathcal{Q}$ against DB′



Materialisation — Example

$$\mathcal{O} \left\{ \begin{array}{rl} \mathsf{Doctor} & \equiv \exists \mathsf{treats}.\mathsf{Patient} \\ \mathsf{Consulatant} & \sqsubseteq \mathsf{Doctor} \end{array} \right.$$

$$\mathsf{DB} \left\{ \begin{array}{l} \mathsf{treats}(d_1, p_1) \\ \mathsf{Patient}(p_1) \\ \mathsf{Doctor}(d_2) \\ \mathsf{Consultant}(c_1) \end{array} \right.$$

Materialisation — Example

$$\mathcal{O} \left\{ \begin{array}{rl} \mathsf{Doctor} & \equiv \exists \mathsf{treats}.\mathsf{Patient} \\ \mathsf{Consulatant} & \sqsubseteq \mathsf{Doctor} \end{array} \right.$$

$$\mathsf{DB} \left\{ \begin{array}{l} \mathsf{treats}(d_1, p_1) \\ \mathsf{Patient}(p_1) \\ \mathsf{Doctor}(d_2) \\ \mathsf{Consultant}(c_1) \end{array} \right.$$

$$\mathsf{DB}' \left\{ \begin{array}{l} \mathsf{treats}(d_1, p_1) \\ \mathsf{Patient}(p_1) \\ \mathsf{Doctor}(d_2) \\ \mathsf{Consultant}(c_1) \\ \mathsf{Doctor}(d_1) \\ \mathsf{Doctor}(c_1) \end{array} \right.$$

Materialisation — Example

$$\mathcal{O} \begin{cases} \text{Doctor} \equiv \exists \text{treats}.\text{Patient} \\ \text{Consulatant} \sqsubseteq \text{Doctor} \end{cases}$$

$$\text{DB} \begin{cases} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \end{cases}$$

$$\text{DB}' \begin{cases} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \\ \text{Doctor}(d_1) \\ \text{Doctor}(c_1) \end{cases}$$

$$\mathcal{Q}_1 \quad Q(x) \leftarrow \text{Doctor}(y)$$

Materialisation — Example

$$\mathcal{O} \left\{ \begin{array}{rl} \text{Doctor} & \equiv \exists \text{treats}.\text{Patient} \\ \text{Consulatant} & \sqsubseteq \text{Doctor} \end{array} \right.$$

$$\text{DB} \left\{ \begin{array}{l} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \end{array} \right.$$

$$\text{DB}' \left\{ \begin{array}{l} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \\ \text{Doctor}(d_1) \\ \text{Doctor}(c_1) \end{array} \right.$$

$$\mathcal{Q}_1 \quad Q(x) \leftarrow \text{Doctor}(y) \qquad \rightsquigarrow \qquad \{d_2, d_1, c_1\}$$

Materialisation — Example

$$\mathcal{O} \left\{ \begin{array}{rcl} \mathsf{Doctor} & \equiv & \exists \mathsf{treats}.\mathsf{Patient} \\ \mathsf{Consulatant} & \sqsubseteq & \mathsf{Doctor} \end{array} \right.$$

$$\mathsf{DB} \left\{ \begin{array}{l} \mathsf{treats}(d_1, p_1) \\ \mathsf{Patient}(p_1) \\ \mathsf{Doctor}(d_2) \\ \mathsf{Consultant}(c_1) \end{array} \right.$$

$$\mathsf{DB}' \left\{ \begin{array}{l} \mathsf{treats}(d_1, p_1) \\ \mathsf{Patient}(p_1) \\ \mathsf{Doctor}(d_2) \\ \mathsf{Consultant}(c_1) \\ \mathsf{Doctor}(d_1) \\ \mathsf{Doctor}(c_1) \end{array} \right.$$

$$\mathcal{Q}_1 \quad Q(x) \leftarrow \mathsf{Doctor}(y) \qquad \rightsquigarrow \qquad \{d_2, d_1, c_1\}$$

$$\mathcal{Q}_2 \quad Q(x) \leftarrow \mathsf{treats}(x, y) \wedge \mathsf{Patient}(y)$$

Materialisation — Example

$$\mathcal{O} \left\{ \begin{array}{rl} \text{Doctor} & \equiv \exists \text{treats}.\text{Patient} \\ \text{Consulatant} & \sqsubseteq \text{Doctor} \end{array} \right.$$

$$\text{DB} \left\{ \begin{array}{l} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \end{array} \right.$$

$$\text{DB}' \left\{ \begin{array}{l} \text{treats}(d_1, p_1) \\ \text{Patient}(p_1) \\ \text{Doctor}(d_2) \\ \text{Consultant}(c_1) \\ \text{Doctor}(d_1) \\ \text{Doctor}(c_1) \end{array} \right.$$

$$\mathcal{Q}_1 \quad Q(x) \leftarrow \text{Doctor}(y) \quad \rightsquigarrow \quad \{d_2, d_1, c_1\}$$

$$\mathcal{Q}_2 \quad Q(x) \leftarrow \text{treats}(x, y) \wedge \text{Patient}(y) \quad \rightsquigarrow \quad \{d_1\}$$

Materialisation — Issues

Incompleteness

- Difficult to predict and/or quantify
- Query negation, aggregation, etc., can can cause unsoundness

Frequently changing data

- Materialisation is time consuming
- Need to re-materialise whenever ontology or data changes

❸ Requires data migration

- Data assumed to be in an RDF DB
- Migrating data from existing RDBMSs may be impractica

Various Approaches — Different Tradeoffs

❸ Use a suitable "profile" and specialised reasoner:

✓ Tractable query answering in size of data

✓ Reliable answers (for inputs in the profile)

✗ Restricted expressivity of the ontology language

OWL 2 QL ontology reasoners:

- E.g., QuOnto, Mastro, Requiem, ...
- Based on query rewriting technique — ontology used to rewrite (expand) query
- Data in scalable (relational) data stores

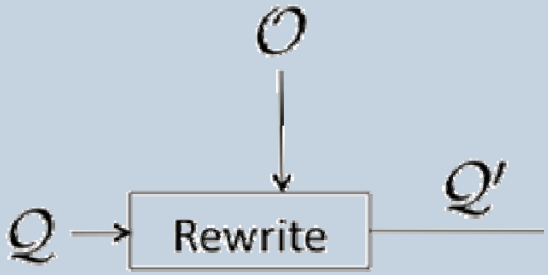
Query Rewriting

Given ontology $\mathcal{O}$ query $\mathcal{Q}$ **and mappings** $\mathcal{M}$**:**

Query Rewriting

Given ontology $\mathcal{O}$ query $\mathcal{Q}$ **and mappings** $\mathcal{M}$**:**

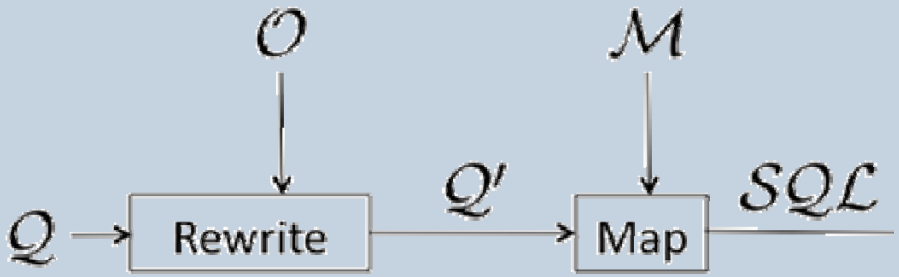
- **Rewrite** $\mathcal{Q} \rightarrow \mathcal{Q}'$ s.t. answering $\mathcal{Q}'$ without $\mathcal{O}$ equivalent to answering $\mathcal{Q}$ w.r.t. $\mathcal{O}$ *for any dataset*



Query Rewriting

Given ontology $\mathcal{O}$ query $\mathcal{Q}$ **and mappings** $\mathcal{M}$**:**

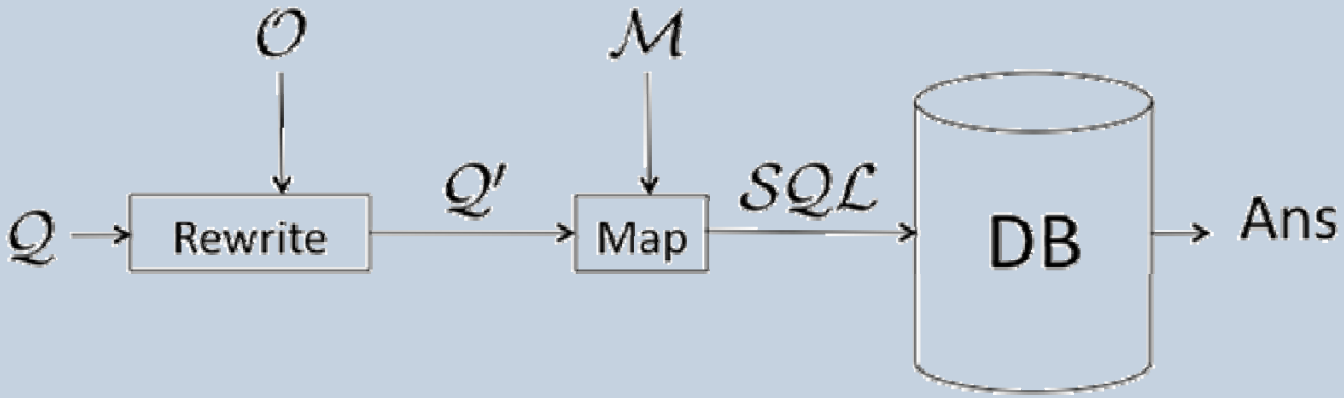
- **Rewrite** $\mathcal{Q} \rightarrow \mathcal{Q}'$ s.t. answering $\mathcal{Q}'$ without $\mathcal{O}$ equivalent to answering $\mathcal{Q}$ w.r.t. $\mathcal{O}$ *for any dataset*
- **Map** ontology queries → DB queries (typically SQL) using mappings $\mathcal{M}$ to rewrite $\mathcal{Q}$' into a DB query



Query Rewriting

Given ontology $\mathcal{O}$ query $\mathcal{Q}$ **and mappings** $\mathcal{M}$**:**

- **Rewrite** $\mathcal{Q} \rightarrow \mathcal{Q}'$ s.t. answering $\mathcal{Q}'$ without $\mathcal{O}$ equivalent to answering $\mathcal{Q}$ w.r.t. $\mathcal{O}$ *for any dataset*
- **Map** ontology queries → DB queries (typically SQL) using mappings $\mathcal{M}$ to rewrite $\mathcal{Q}$' into a DB query
- **Evaluate** (SQL) query against DB



Query Rewriting — Example

$$\mathcal{O} \left\{ \begin{array}{r} \text{Doctor} \sqsubseteq \exists \text{treats}.\text{Patient} \\ \text{Consultant} \sqsubseteq \text{Doctor} \end{array} \right.$$

$$\mathcal{Q} \quad Q(x) \leftarrow \text{treats}(x, y) \wedge \text{Patient}(y)$$

$$\mathcal{M} \left\{ \begin{array}{rcl} \text{Doctor} & \mapsto & \text{SELECT Name FROM Doctor} \\ \text{Patient} & \mapsto & \text{SELECT Name FROM Patient} \\ \text{treats} & \mapsto & \text{SELECT DName, PName FROM Treats} \end{array} \right.$$

Query Rewriting — Example

$$\mathcal{O} \begin{cases} \text{Doctor} \sqsubseteq \exists \text{treats}.\text{Patient} \\ \text{Consultant} \sqsubseteq \text{Doctor} \end{cases}$$

$$\mathcal{Q} \quad Q(x) \leftarrow \text{treats}(x, y) \wedge \text{Patient}(y)$$

$$\mathcal{Q}' \begin{cases} Q(x) \leftarrow \text{treats}(x, y) \wedge \text{Patient}(y) \\ Q(x) \leftarrow \text{Doctor}(x) \wedge \text{Patient}(f(x)) \\ Q(x) \leftarrow \text{treats}(x, f(x)) \wedge \text{Doctor}(x) \\ Q(x) \leftarrow \text{Doctor}(x) \\ Q(x) \leftarrow \text{Consultant}(x) \end{cases}$$

$$\mathcal{M} \begin{cases} \text{Doctor} \mapsto \text{SELECT Name FROM Doctor} \\ \text{Patient} \mapsto \text{SELECT Name FROM Patient} \\ \text{treats} \mapsto \text{SELECT DName, PName FROM Treats} \end{cases}$$

Query Rewriting — Example

$$\mathcal{O} \begin{cases} \text{Doctor} \sqsubseteq \exists \text{treats}.\text{Patient} \\ \text{Consultant} \sqsubseteq \text{Doctor} \end{cases}$$

$$\mathcal{Q} \quad Q(x) \leftarrow \text{treats}(x, y) \wedge \text{Patient}(y)$$

$$\mathcal{Q}' \begin{cases} Q(x) \leftarrow \text{treats}(x, y) \wedge \text{Patient}(y) \\ \sout{Q(x) \leftarrow \text{Doctor}(x) \wedge \text{Patient}(f(x))} \\ \sout{Q(x) \leftarrow \text{treats}(x, f(x)) \wedge \text{Doctor}(x)} \\ Q(x) \leftarrow \text{Doctor}(x) \\ \sout{Q(x) \leftarrow \text{Consultant}(x)} \end{cases}$$

$$\mathcal{M} \begin{cases} \text{Doctor} \mapsto \text{SELECT Name FROM Doctor} \\ \text{Patient} \mapsto \text{SELECT Name FROM Patient} \\ \text{treats} \mapsto \text{SELECT DName, PName FROM Treats} \end{cases}$$

Query Rewriting — Example

$$\mathcal{O} \begin{cases} \text{Doctor} \sqsubseteq \exists \text{treats}.\text{Patient} \\ \text{Consultant} \sqsubseteq \text{Doctor} \end{cases}$$

$$\mathcal{Q} \quad Q(x) \leftarrow \text{treats}(x, y) \wedge \text{Patient}(y)$$

$$\mathcal{Q}' \begin{cases} Q(x) \leftarrow \text{treats}(x, y) \wedge \text{Patient}(y) \\ \sout{Q(x) \leftarrow \text{Doctor}(x) \wedge \text{Patient}(f(x))} \\ \sout{Q(x) \leftarrow \text{treats}(x, f(x)) \wedge \text{Doctor}(x)} \\ Q(x) \leftarrow \text{Doctor}(x) \\ \sout{Q(x) \leftarrow \text{Consultant}(x)} \end{cases}$$

$$\mathcal{M} \begin{cases} \text{Doctor} \mapsto \text{SELECT Name FROM Doctor} \\ \text{Patient} \mapsto \text{SELECT Name FROM Patient} \\ \text{treats} \mapsto \text{SELECT DName, PName FROM Treats} \end{cases}$$

$$\mathcal{SQL} \begin{cases} \text{SELECT Name FROM Doctor UNION} \\ \text{SELECT DName FROM Treats, Patient WHERE PName=Name} \end{cases}$$

Query Rewriting — Issues

Scalability

- Rewritings can be large (exponential) in the worst case
- Rewritten queries may be challenging for existing RDBMSs

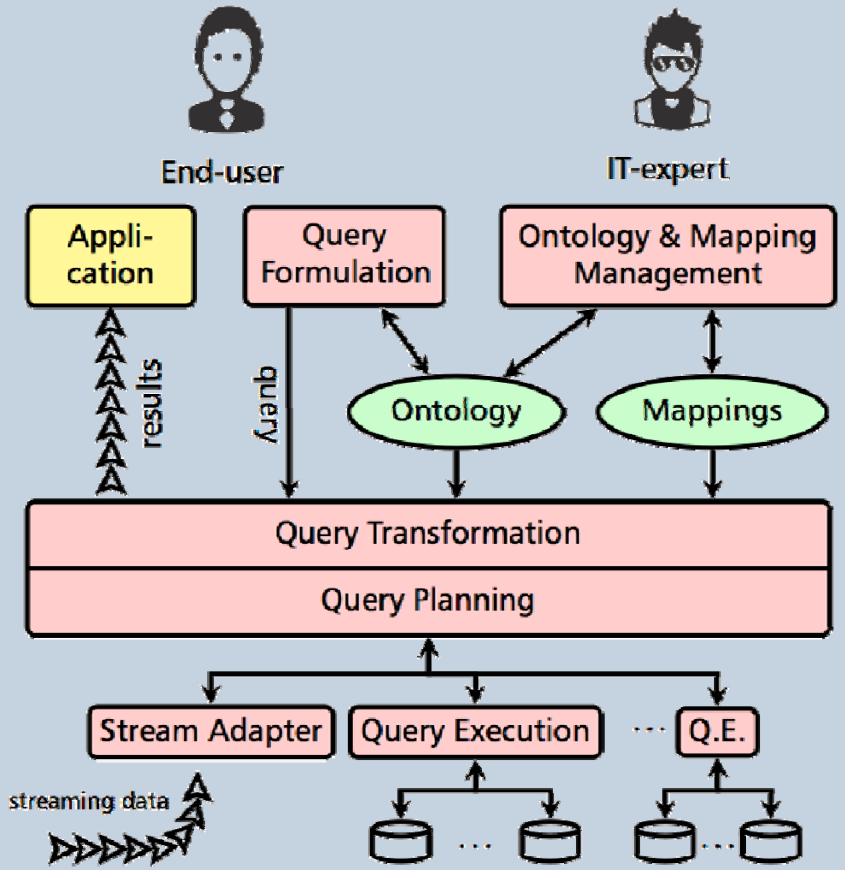
Data integration

- Data may be stored in multiple/distributed RDBMSs
- Mappings help, but still need query planning etc.

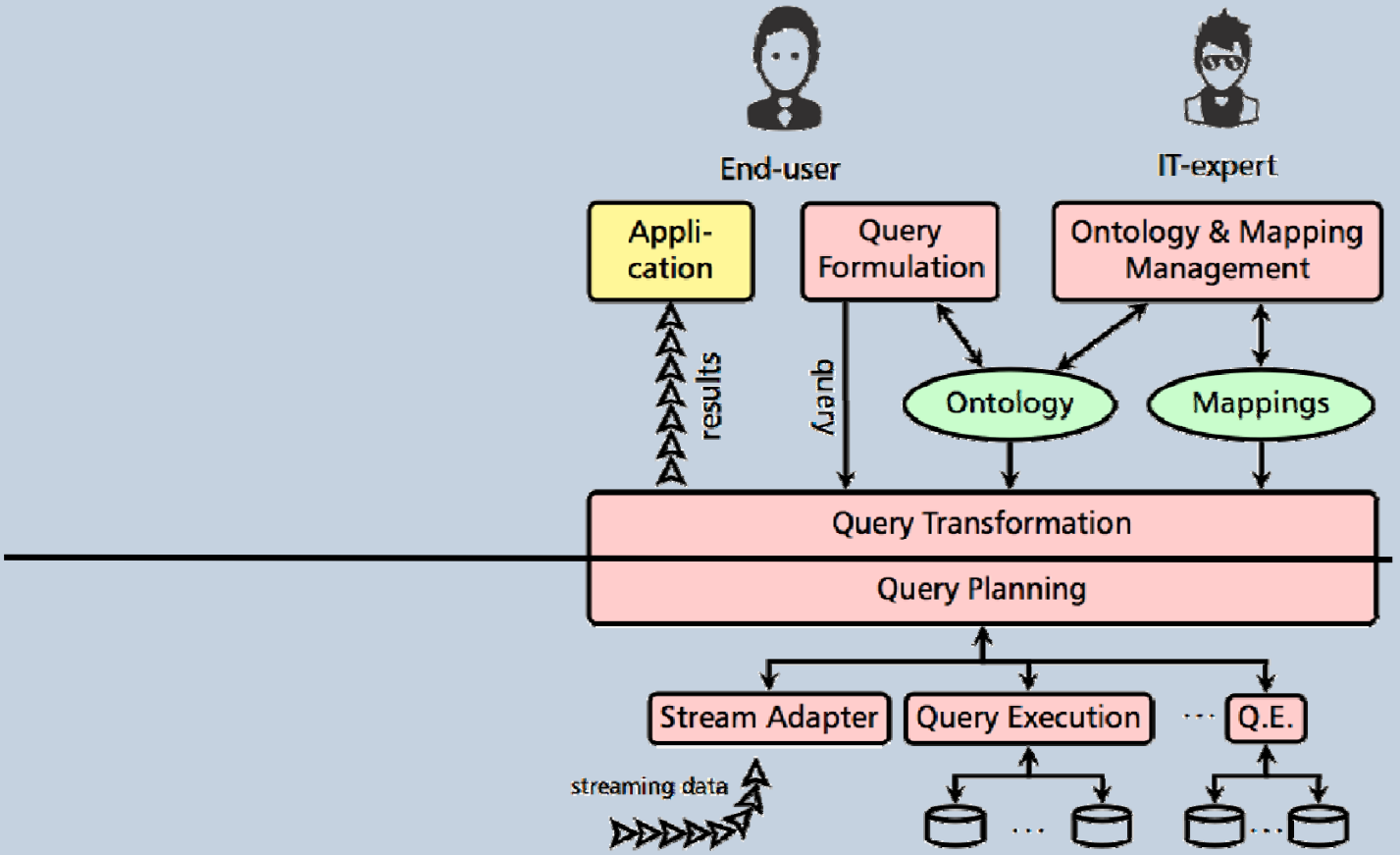
❸ Ontology and mappings

- Ontology development and maintenance known to be costly
- Added overhead of mapping D&M

Optique Platform Architecture



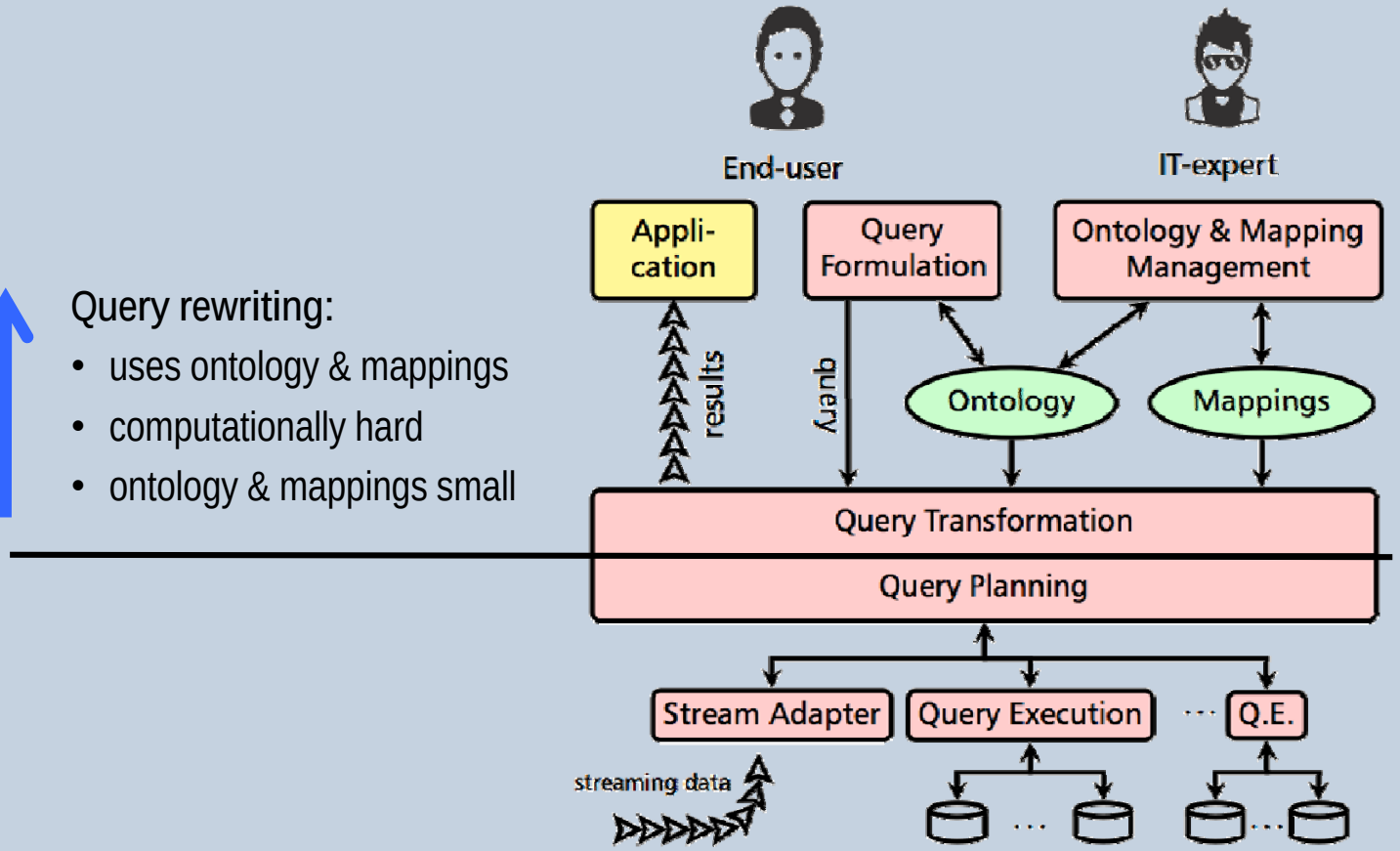
Optique Platform Architecture



Optique Platform Architecture

Query rewriting:

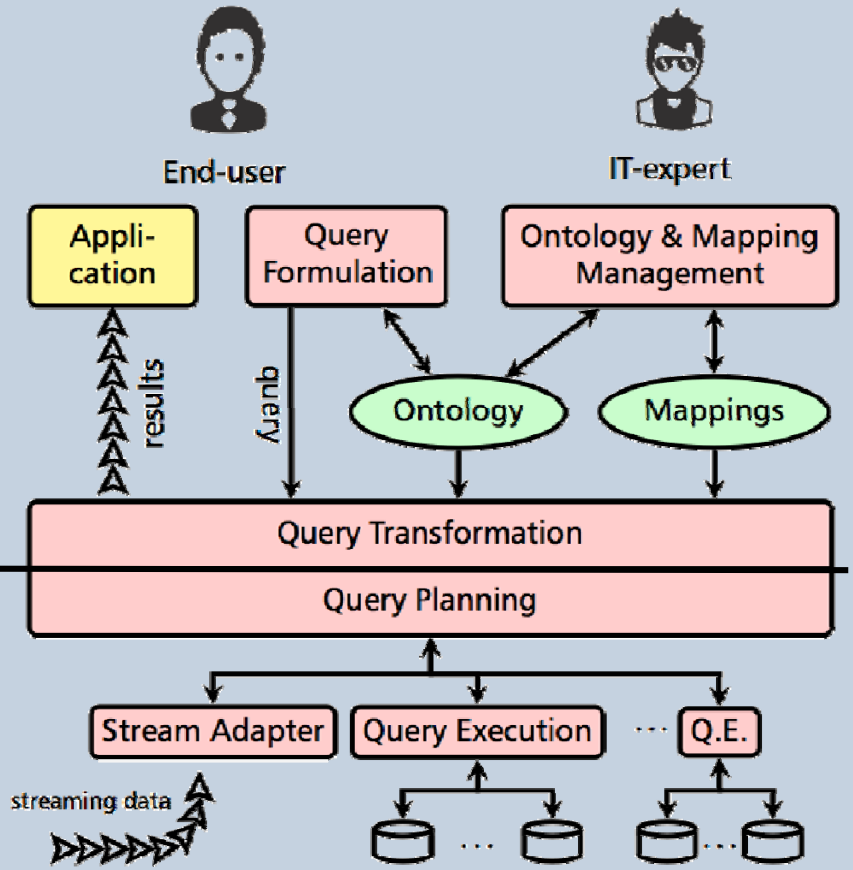
- uses ontology & mappings
- computationally hard
- ontology & mappings small



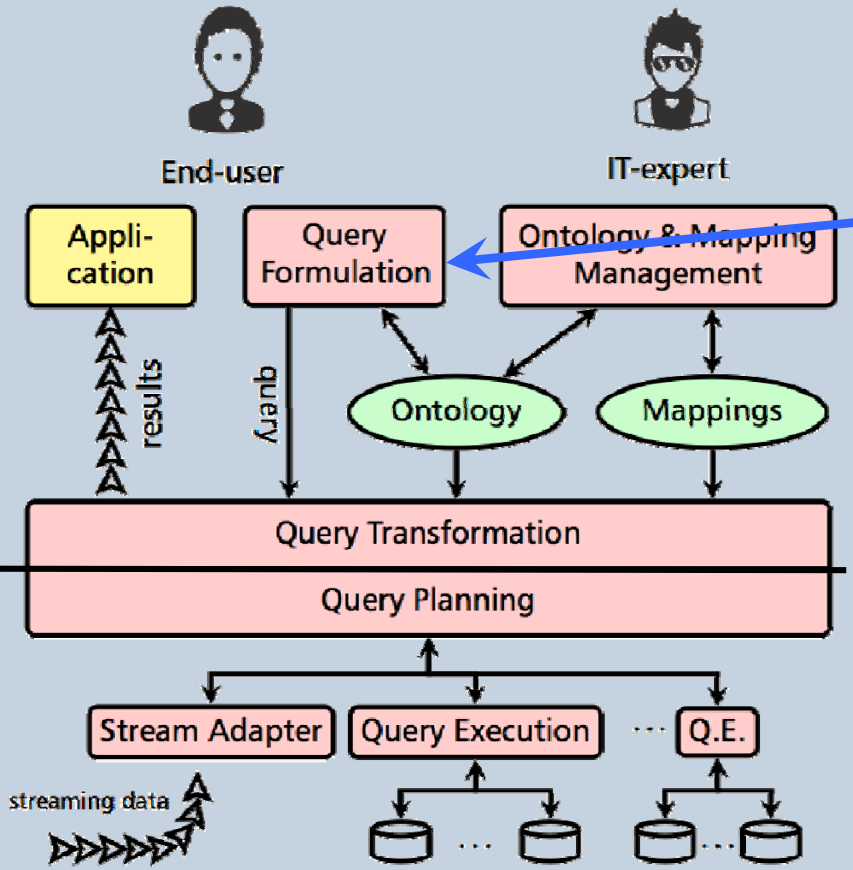
Optique Platform Architecture

Query rewriting:
- uses ontology & mappings
- computationally hard
- ontology & mappings small

Query evaluation:
- ind. of ontology & mappings
- computationally tractable
- data sets very large



Optique Platform Architecture



Other features:

support for query formulation

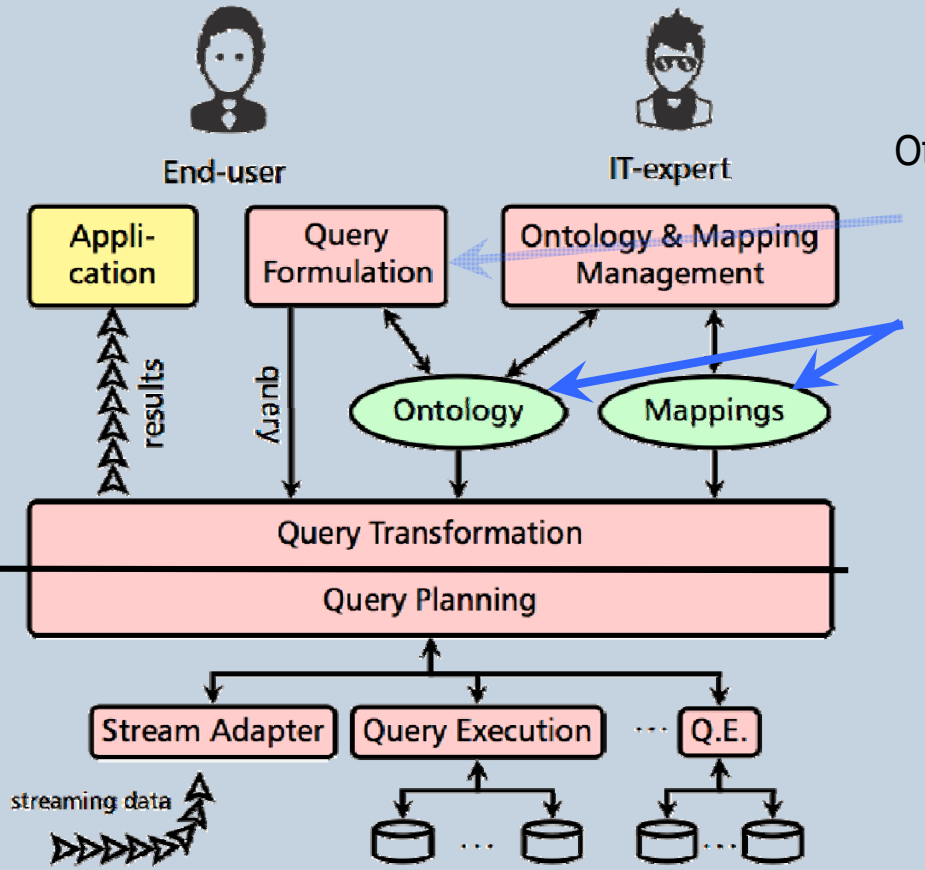
Query rewriting:

- uses ontology & mappings
- computationally hard
- ontology & mappings small

Query evaluation:

- ind. of ontology & mappings
- computationally tractable
- data sets very large

Optique Platform Architecture



Query rewriting:

- uses ontology & mappings
- computationally hard
- ontology & mappings small

Query evaluation:

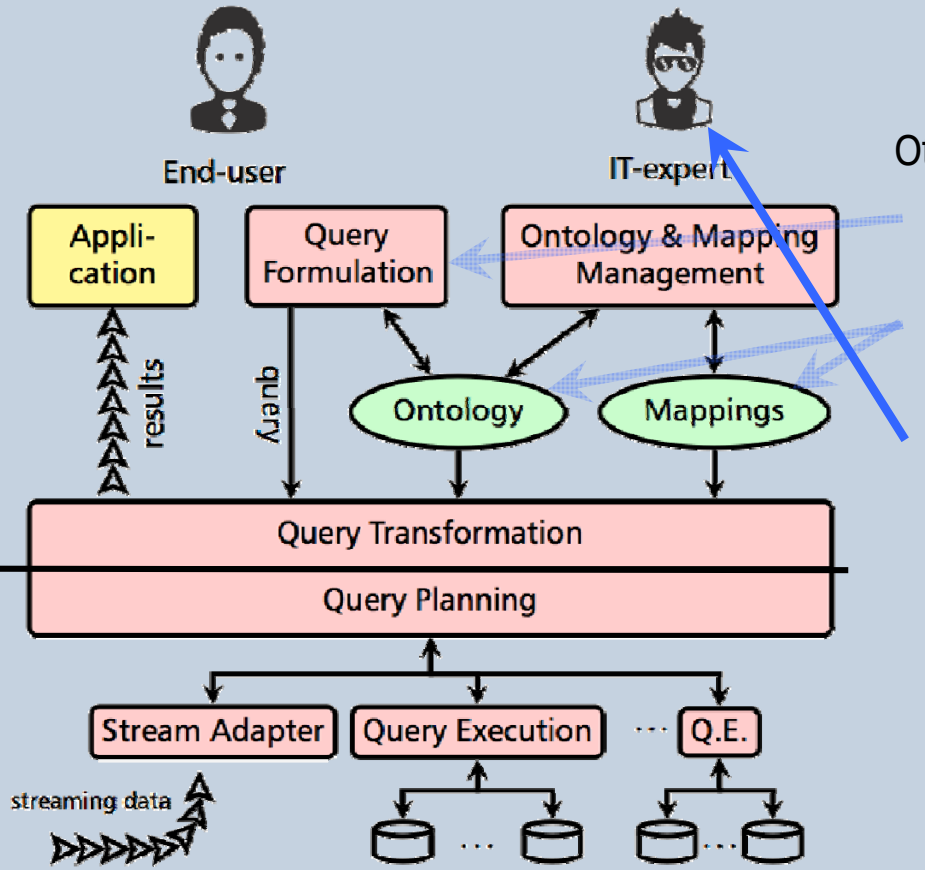
- ind. of ontology & mappings
- computationally tractable
- data sets very large

Other features:

support for query formulation

Ontology & mappings extended “on the fly”

Optique Platform Architecture



Query rewriting:

- uses ontology & mappings
- computationally hard
- ontology & mappings small

Query evaluation:

- ind. of ontology & mappings
- computationally tractable
- data sets very large

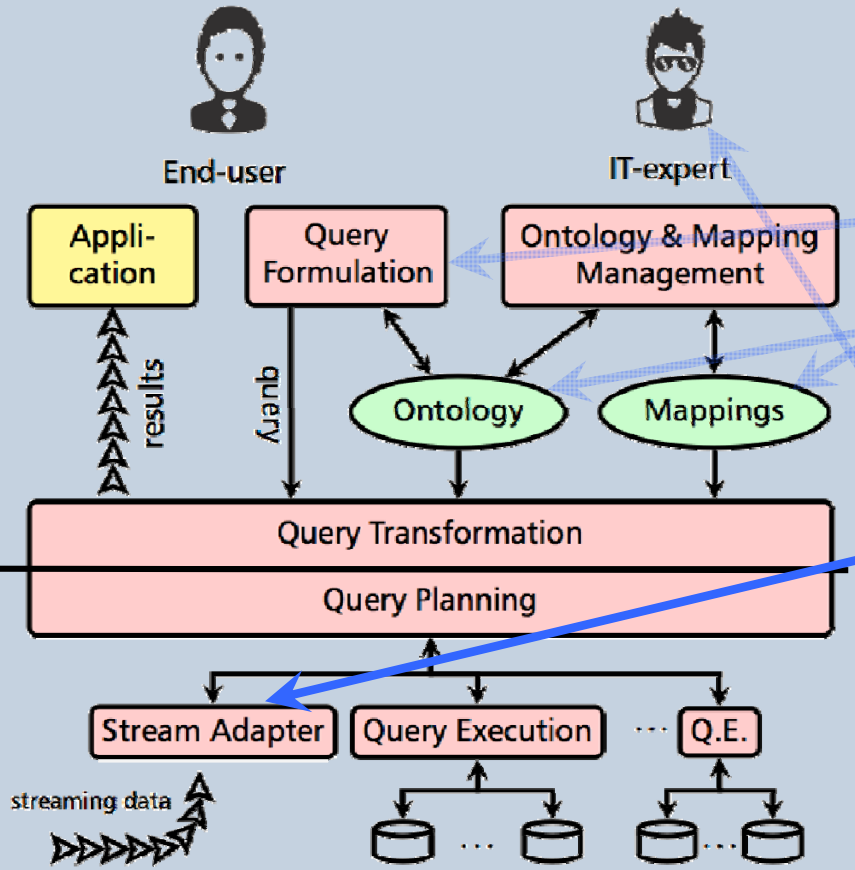
Other features:

support for query formulation

Ontology & mappings extended "on the fly"

IT-expert oversees O&M management

Optique Platform Architecture



Query rewriting:

- uses ontology & mappings
- computationally hard
- ontology & mappings small

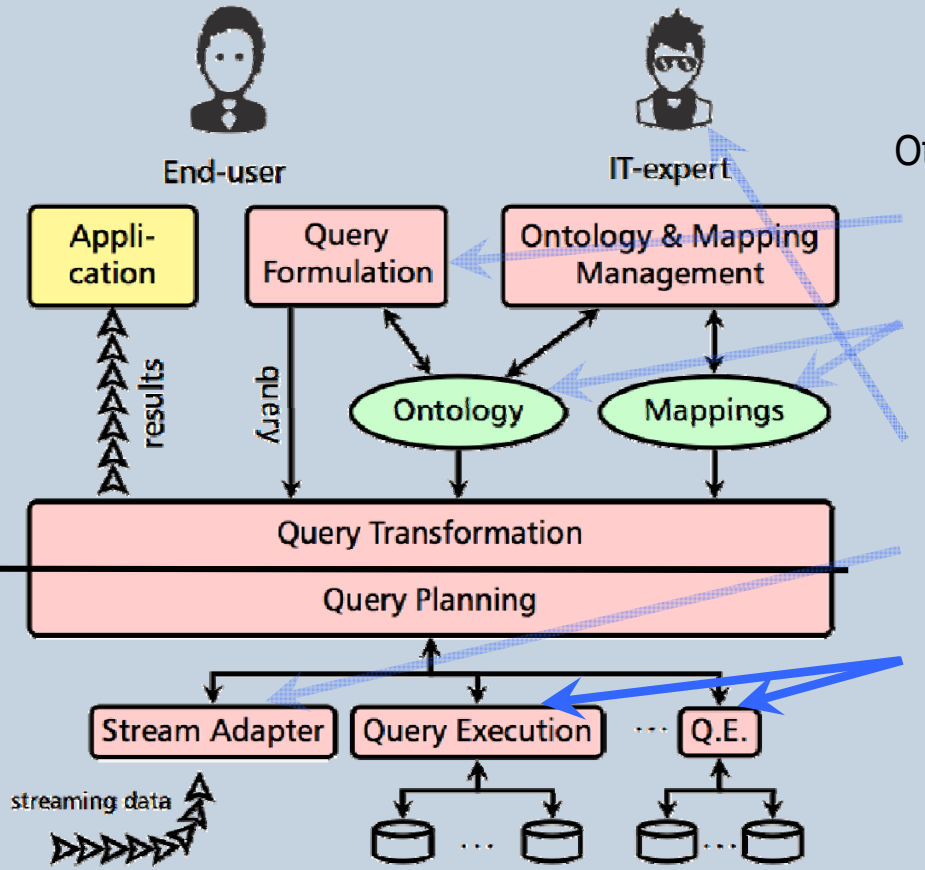
Query evaluation:

- ind. of ontology & mappings
- computationally tractable
- data sets very large

Other features:

- support for query formulation
- Ontology & mappings extended "on the fly"
- IT-expert oversees O&M management
- Adapter to support streaming data

Optique Platform Architecture



Query rewriting:

- uses ontology & mappings
- computationally hard
- ontology & mappings small

Query evaluation:

- ind. of ontology & mappings
- computationally tractable
- data sets very large

Other features:

- support for query formulation
- Ontology & mappings extended "on the fly"
- IT-expert oversees O&M management
- Adapter to support streaming data
- Distributed query execution

Acknowledgements

EPSRC Engineering and Physical Sciences Research Council

SIXTH FRAMEWORK PROGRAMME

SEVENTH FRAMEWORK PROGRAMME

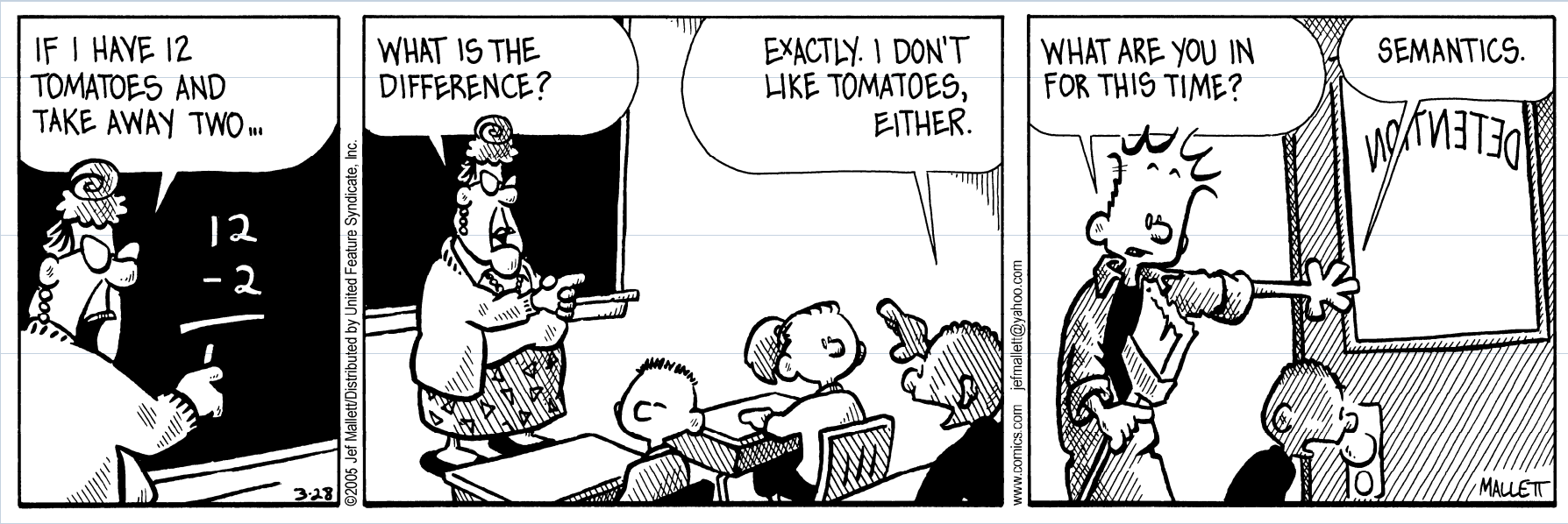
DARPA

UNIVERSITY OF OXFORD DEPARTMENT OF COMPUTER SCIENCE

Optique

SEVENTH FRAMEWORK PROGRAMME

Thank you for listening



FRAZZ: © Jeff Mallett/Dist. by United Feature Syndicate, Inc.

Any questions?

UNIVERSITY OF OXFORD
DEPARTMENT OF COMPUTER SCIENCE

Optique

SEVENTH FRAMEWORK PROGRAMME