



University Mentouri, Constantine



Department of Computer Science  
LIRE Laboratory

# Semantic Enrichment of Relational Databases

Directed By:

**Dr. F. Benchikha**

Presented By:

**Kamal Hamaz**

IWAISE 2012

# Plan

## 1. Introduction

### Context

- 2. Definition of the Reverse Engineering Process
- 3. Why to Apply Reverse Engineering Over Legacy Databases?
- 4. Reverse Engineering of Relational Databases to New Models
- 5. Related Work

### Contribution

- 6. The Proposed Approach
  - Quick Reminder About the Conception of Relational Databases
  - Transformation Process
  - Enrichment Process

## 7. Conclusion & Perspectives

# 1. Introduction

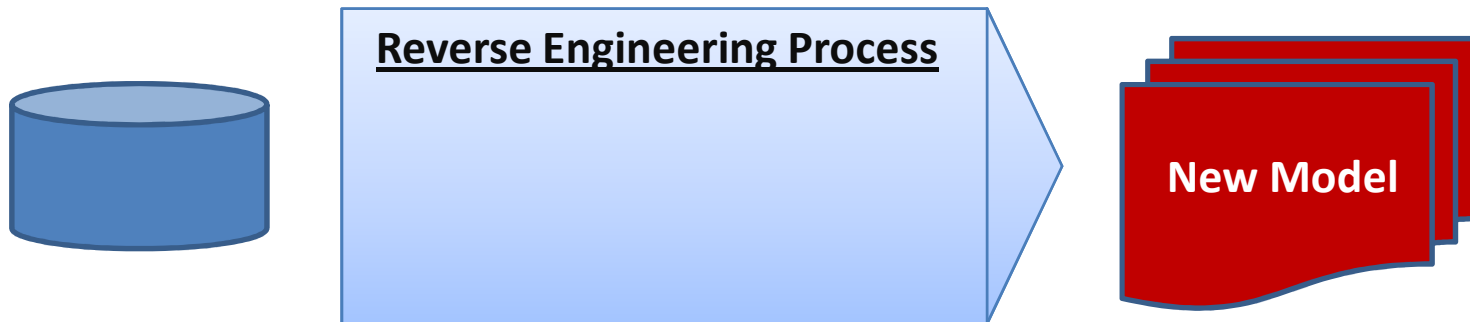
The development of **Information Systems** is one of the priorities of most enterprises and organizations. **Databases** play one of the principal roles in the development of every information system. And thus the development of databases will also lead to the development of any Information System.

The relational model that appeared in the 70' is still the most used model for storing data. However it has several weaknesses when dealing with complex and heterogeneous environments.

Therefore the conception of databases has been reviewed and other models were proposed. To this end, enterprises needed an efficient **Reverse Engineering** process to migrate their legacy databases to a new database model.

## 2. Definition of the Reverse Engineering Process

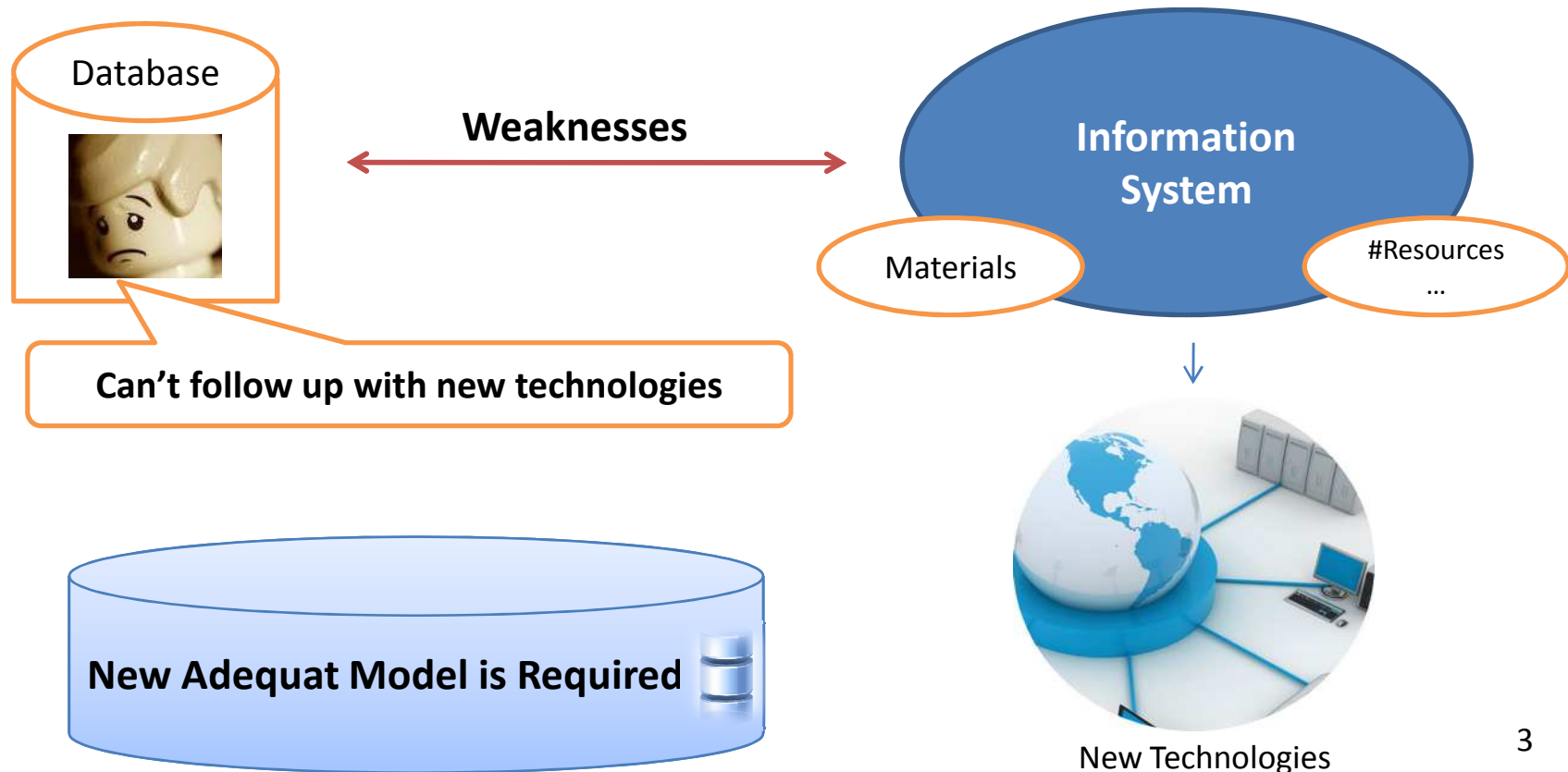
- The process of Reverse Engineering is generally defined by the process of **reconstruction or restructuration of a model into a higher level model**.
- For legacy **Databases**, the process of **Reverse Engineering** is mainly considered as the process of enrichment of a database source. This gives the ability to discover new information and semantics for the target **New Model**.



### 3. Why to Apply Reverse Engineering Over Legacy Databases?

Applying a Reverse Engineering process over legacy databases can be motivated by 3 main reasons.

**Reason1:** Legacy Databases Show Weaknesses Dealing With New Technologies (Interoperability, Semantic Web ...etc..).



### 3. Why to Apply Reverse Engineering Over Legacy Databases?

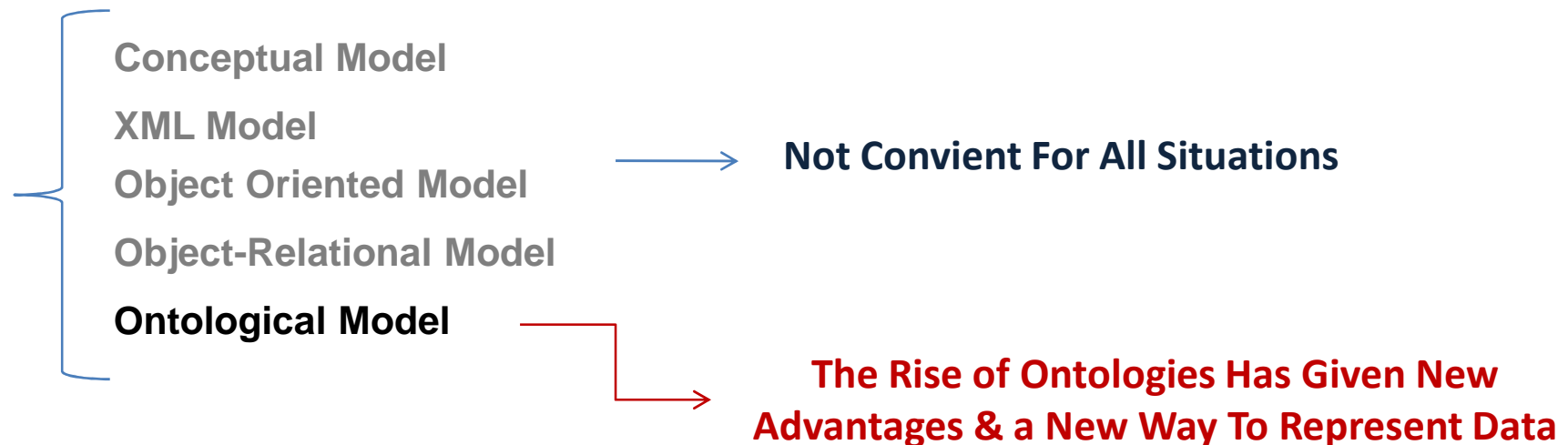
**Reason2:** Legacy Databases contain an important mass of data, as well as useful information to reuse.

**Reason3:** It is cost effective to build a new database model from scratch and is also complicated and will require a lot of time. However, by applying a Reverse Engineering process we reduce considerably costs as well as time.

## 4. Reverse Engineering of Relational Databases to New Models

**The Relational Database Model** was and widely still, the most used model for storing data by enterprises. It can also be seen as a legacy form for storing data.

Many Models were proposed as a source-to-target for Reverse Engineering of Relational Databases:



## 4. Reverse Engineering of Relational Databases to New Models

**Ontologies** are defined as an hierarchy based on structured vocabularies, grouping together concepts/classes with their relationships and instances.

**Advantages:** Ontologies provide many advantages in comparison with the previous proposed models.

- ❖ Rigorous formulation of the schemas with high abilities to well represent semantically any domain.
- ❖ Possibility to use online dictionaries and vocabularies as WordNet.
- ❖ Possibility to infer new results and new facts.

**Based on these advantages and others, many works on integrating ontologies within databases have appeared.**



## 5. Related Work

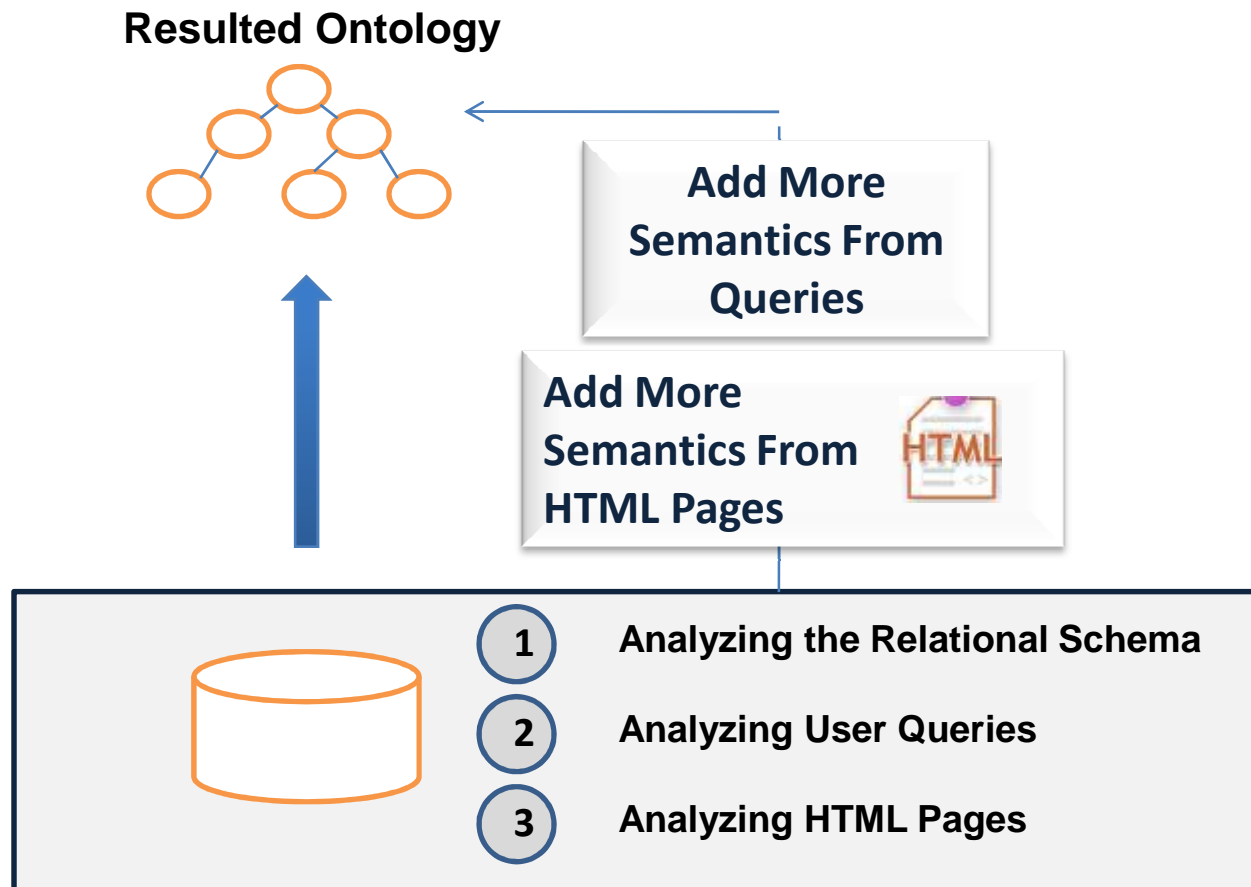
**Related Work** on Reverse Engineering of Relational Databases to the 'Ontological Model' can be divided into 3 categories.

1. (I. Astrova et al, 2008)

2. (V. Kashyap, 1999)

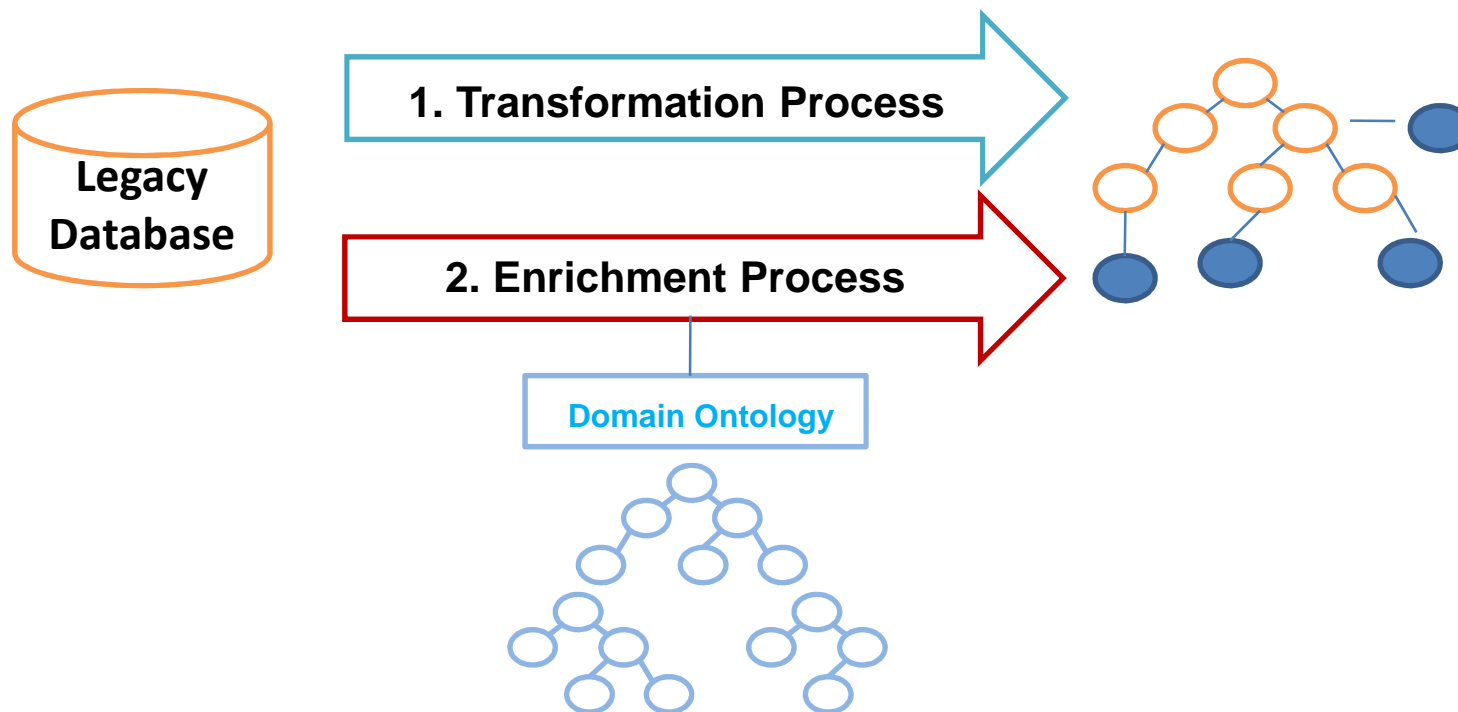
3. (S. Benslimane et al, 2008)

- This process creates a first ontology from the schema
- Analyse HTML forms and tables to extract new entities
- Add the discovered entities to the ontology



## 6. The Proposed Approach

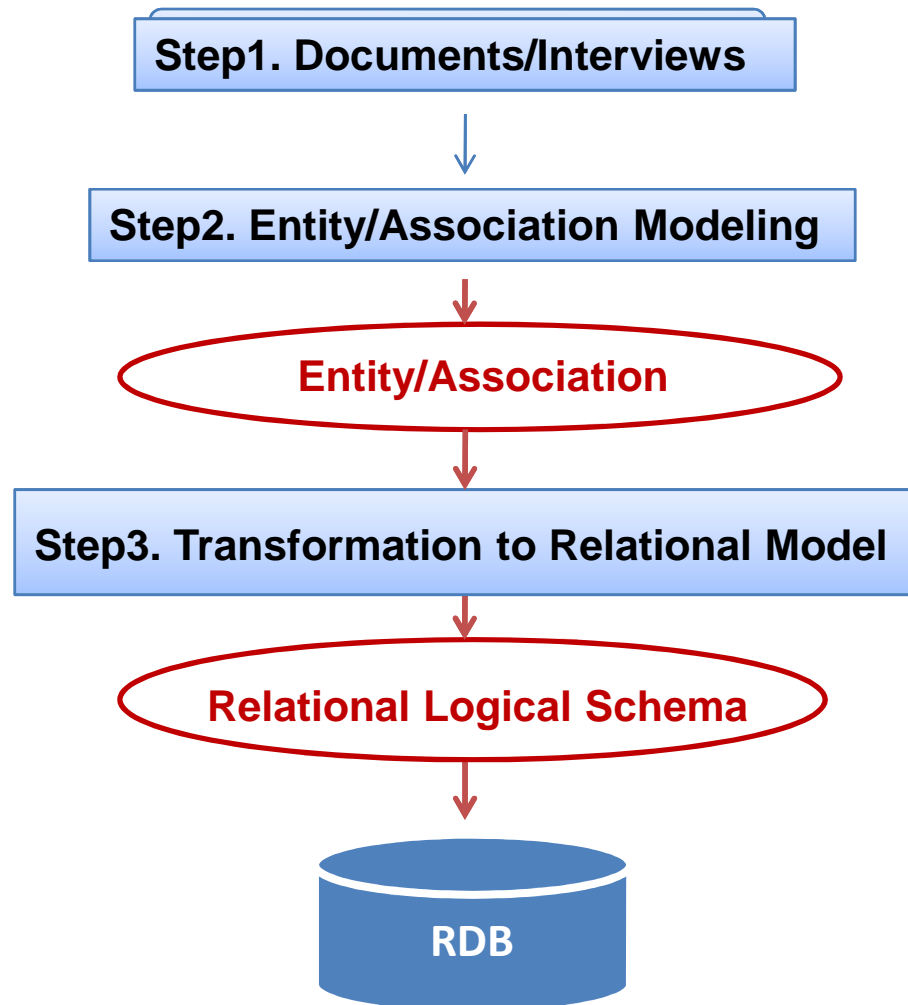
### Our Proposed Approach:



## 6. The Proposed Approach (2)

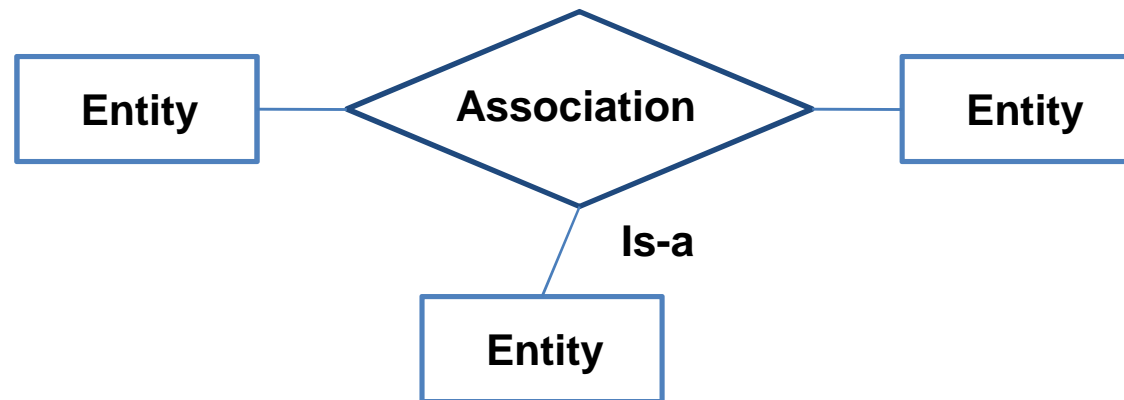
### A Quick Reminder About the Conception of Relational Databases:

A Relational Database (RDB) is the fruit of Many steps:



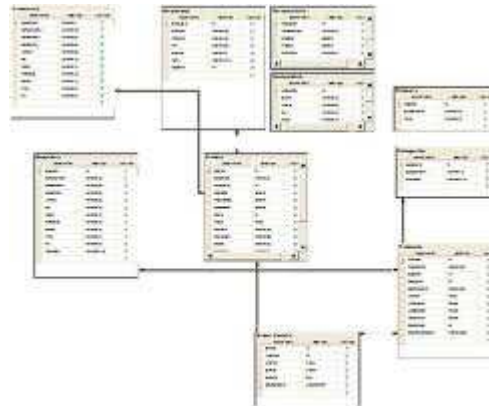
## 6. The Proposed Approach (2)

### Step2. Entity/Association Modeling



Part of the  
Semantics  
Still Exist

### Step3. Transformation to Relational Model



Loss of  
Semantics

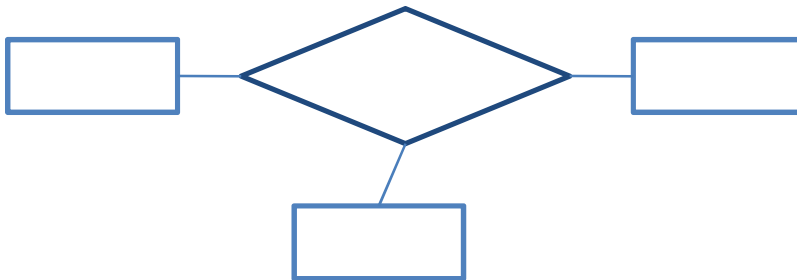
## 6. The Proposed Approach (2)

### What Our Reverse Engineering Process Actually Do:

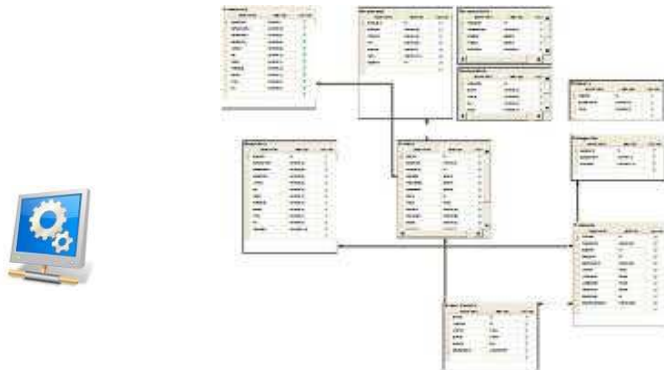
OWL Document



Step2. Entity/Association Modeling



Step3. Transformation to Relational Model



1. Recovers the semantics lost in the relational model and tries to discover more additional information

2. Instead of representing the semantics/information discovered in a conceptual schema, our process represents it as an ontology (OWL)

## 6. The Proposed Approach (3)

### Our Transformation Process is Composed of A Set of Rules:

#### Rule for the Creation of Classes

**Rule1:** Create a class for each relation in the database where the primary key isn't composed exclusively of foreign keys

**Person**(Id, Name, PhoneNum, Address, Country) -> **New Class PERSON**  
Created

#### Rules for the Creation of the Hierarchy

**Rule2:** Create a generalization relationship between two classes if the relations related to the classes have the same primary key, and/or the values of one of the primary key are a subset of the values in the second primary key.

**Person**(Id, Name, PhoneNum, Address, Country)  
**Student**(StudentID, Deprtn)

+ Values of **StudentID** are a subset of the values of **Id**

-> **Class STUDENT** Will be Represented as a **Subclass** of **PERSON**

## 6. The Proposed Approach (4)

### Rules for the Creation of the Hierarchy (2)

**Rule3:** Create a generalization relationship between two classes if: one of the two relations related to the classes has its values of the primary key included in the candidate key of the second relation.

Employee(emp, **Name, address (CandidateKey)**, .....)

Librarians(name,adresse, phone)

+ Values of **name,address** (Librarians) are a subset of the values of **name, address** (Employee)

-> Class **LIBRARIAN** Will be Represented as a **Subclass** of **EMPLOYEE**

## 6. The Proposed Approach (5)

### Rules for the Creation of Properties

**Rule4:** Create an 'Object Property' if the primary key of a relation is used as a foreign key in a second relation and not participant to its primary key. The domain and range are the classes related to the relations. For this case the cardinality will be considered as  $<1:N>$  and in other cases  $<0:N>$ .

Student(StudentID, IdDep)

Department(IdDep, Name, Organisation)

+ **IdDep** the primary key of Department is used as a foreign key in Student

-> '**Object Property**' created between **STUDENT** and **DEPARTMENT**



## 6. The Proposed Approach (6)

### Rules for the Creation of Properties (2)

**Rule5:** Create an 'Object Property' if the primary key of a relation is used as a foreign key in a second relation and participating to its primary key. The domain and range are the classes related to the relations. In this case the cardinality will be <1:1>.

Student(StudentID, IdDep)

Sanctions(SanctionNum, StudentID, Reason, Year...)

+ **StudentID** which is part of primary key in Sanctions refers to **StudentsID** of Student.

-> '**Object Property**' created between **STUDENT** and **Sanctions**

## 6. The Proposed Approach (7)

### Rules for the Creation of Properties (3)

**Rule6:** Create an 'Object Property' if a relation has a **primary key** which is **composed of two foreign keys**. The domain and range are the classes related to the relations that references those foreign keys. In this case the **cardinality** will be **<M:N>**.

**Student**(**StudentID**, IdDep)

**Article** (**Idart**, Author, Title, Domain, Event)

**Prepares**(**StudentID, Idart**)

+ **StudentID, Idart** which is the primary key of Prepares is composed of two attributes that are used as foreign keys in two other relations.

-> '**Object Property**' created between **STUDENT** and **ARTICLE**

## 6. The Proposed Approach (8)

### Rules for the Creation of Properties (3)

**Rule7:** Create a 'Functional Property' if a relation has a primary key composed of more than two foreign keys.

**Student** (StudentID, IdDep)

**Book** ( ISBN, Title, Author, Code)

**Date** ( IDATe, day, month, year)

**Borrow** (ISBN, StudentID, IDATe)

+ ISBN, StudentID, IDATe which is the primary key of Borrow is composed of 3 attributes that are used as foreign keys in 3 other relations.

-> **BORROW** created as Class & '**Functional Property**' are created between **STUDENT BOOK** and **DATE**

## 6. The Proposed Approach (9)

### Rules for the Creation of Properties (4)

**Rule8:** Create a disjoint property between two classes if the union of the primary keys of the relations related to these classes constitutes the values of the primary key of another relation.

**Student** (StudentID, IdDep)

**Professor**(ProfessorID, Salary)

||

**Person**(Id, Name, PhoneNum, Address, Country)

+ The union of the values of **StudentID** and **ProfessorID** is equal to the values of the primary key **id** of Person.

-> '**Disjoint Property**' is created where **STUDENT** is disjoint with **PROFESSOR** for the class **PERSON**.

## 6. The Proposed Approach (10)

### Rules for the Creation of the Attributes

**Rule9:** For each class, create the same attributes as those that exist in the related relation, in the exception of foreign keys

**Person**(Id, Name, PhoneNum, Address, Country)

**Student**(StudentID, ~~IdDep~~(Used as Foreign Key))

- > PERSON will have all same attributes for its class.
- > STUDENT will have the its same attributes in exception of IdDep.

## 6. The Proposed Approach (Results of Applying Transformation Rules)

Rule for classes

Rules for hierarchy

Rules for properties

Rule for attributes

Part of the OWL code

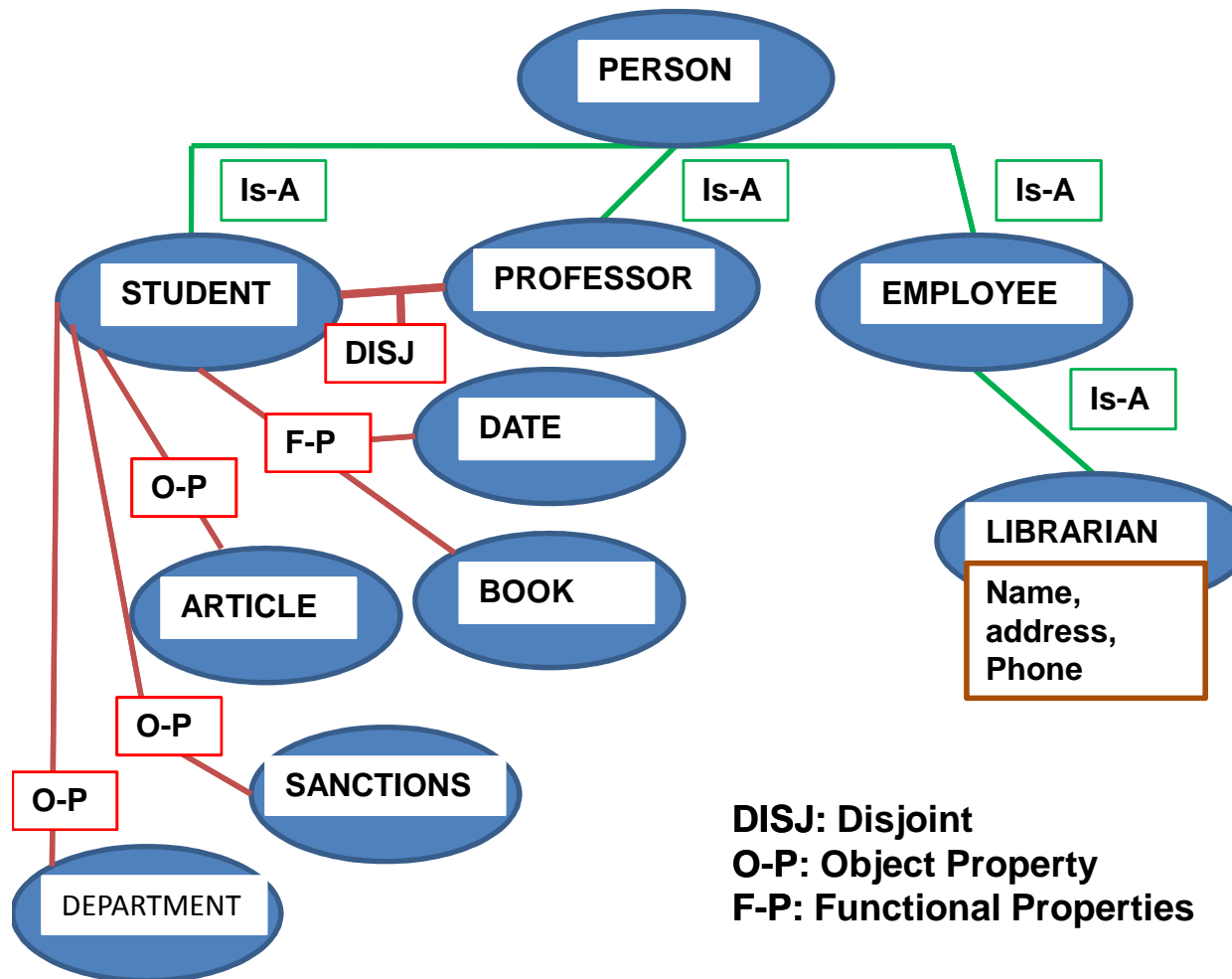
```
<owl:Class rdf:about="Person"/>
```

```
<owl:Class rdf:about="Person">
  <rdfs:subClassOf
    rdf:resource="Student"/>
</owl:Class>
```

```
<owl:ObjectProperty rdf:ID="IdDep">
  <rdfs:domain
    rdf:resource="Department"/>
  <rdfs:range rdf:resource="Student"/>
</owl:ObjectProperty>
```

```
<owl:Class rdf:ID="Person">
  <rdfs:subClassOf
    rdf:resource="Professor"/>
  <owl:disjointWith
    rdf:resource="Student"/>
</owl:Class>
```

```
<owl:DatatypeProperty
  rdf:ID="Phone">
  <rdfs:domain
    rdf:resource="Librarian"/>
  <rdfs:range
    rdf:resource="xsd:Integer"/>
</owl:DatatypeProperty>
```



## 6. The Proposed Approach (Enrichment Process)

Our Enrichment Process has 2 steps:

1st

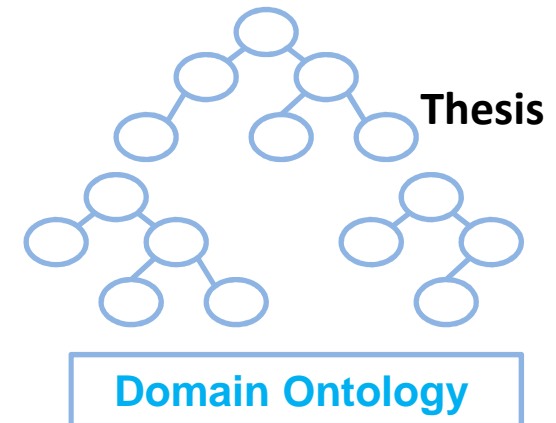
Explore the possibility to represent some candidate keys(CK) as classes.  
(CK's can be seen as relevant hidden object-types (Johannsson 1994))

For each CK

Apply Stemming

Get Synonyms  
(WordNet)

Compare Synonyms with Classes of the Domain Ontology



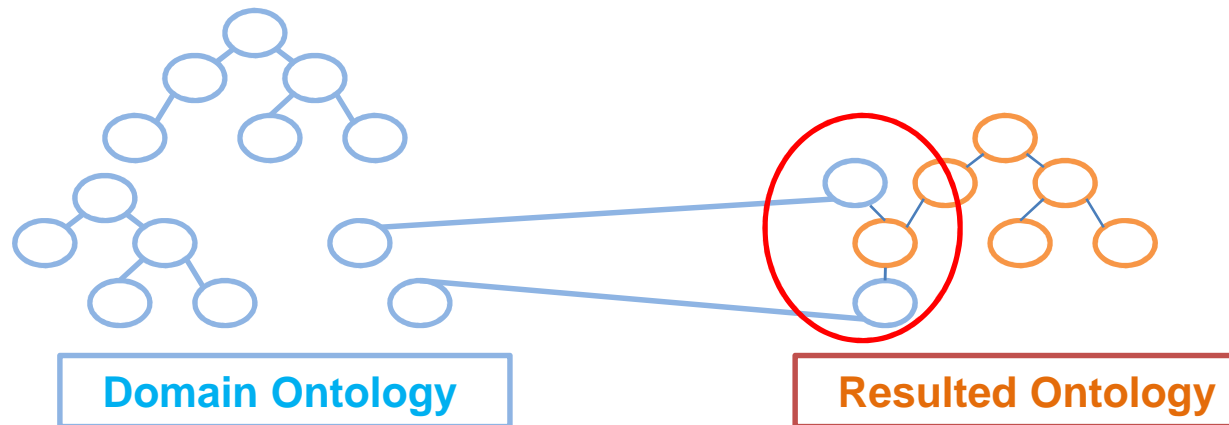
Publication (Idpub, Publisher, Title, ThesisNum (CK))

-> **THESIS** is created as a **new class** and also as a **subclass of** the class **PUBLICATION**.  
Cardinality for this case will be always <1:1>.

## 6. The Proposed Approach (Enrichment Process 2)

2nd

### Algorithm



- 1) Comparison is based on comparing properties of both classes and calculates a degree of similarities
- 2) The Algorithm adds either subclasses or superclasses aswell as their properties once two classes are highly similar.
- 3) The Algorithm prevents a non-consistent (on further use) ontology by dealing with redundancy and makes sure that classes added are well organized.



## 7. Conclusion & Perspectives

The main purpose behind our work is the creation of an **Ontology-Based Database** that satisfies enterprises dealing with the new technologies.

- **We proposed a set of rules** for the creation of a local ontology, and **we enriched** the resulted ontology in order to make the final ontology more complete, more adequate and more interesting for the database developers.

**Future work consists on:**

- **Populating the ontology** with instances and **storing it** in a repository following one of the specific representation approaches.
- **Finishing the implementation of our approach** and run tests
- Providing mechanisms to represent more **constraints of the database in the ontology**.
- Proposing rules to **infer new facts on requesting-time**.

---

***THANK YOU***

**Questions?**