

# Finding where to apply object- relational database schema refactorings: an ontology-guided approach



**Aline Baroni, Fernando Brito e Abreu**

QUASAR research group.

Universidade Nova de Lisboa

**Coral Calero**

ALARCOS research group.

Universidad de Castilla-La Mancha

# Indice

---

- Introducción
- La ontología del SQL:2003
- Formalización de métricas para BDORel
- Refactorización de BDORel
- Ejemplo
- Conclusiones

# Indice

---

- Introducción
- La ontología del SQL:2003
- Formalización de métricas para BDORel
- Refactorización de BDORel
- Ejemplo
- Conclusiones

# Introducción

---

- ❑ The maintainability of database schemas is a relevant research area.
- ❑ Our previous work on this area includes the formal definition of OR database metrics
- ❑ In this work, we are particularly interested in refactoring database schemas.
- ❑ A database refactoring is *“a simple change to a database schema that improves its design, while retaining both its behavioral and informational semantics”*

# Introducción

---

- Choosing where to apply a refactoring is often based on subjective criteria, rather than on objective ones.
- In this paper we propose a metrics-based approach for assisting database schema refactorings.
- It combines previous research results:
  - OR schema metrics
  - SQL:2003 ontology
  - guidelines for refactoring

# Introducción

---

- The process is:
  - (i) we instantiate the database schema ontology for a given OR schema;
  - (ii) we calculate automatically the values of the OR schema metrics;
  - (iii) with those metrics we identify candidate schema fragments for refactoring;
  - (iv) the refactorings are applied by following the corresponding guidelines;
  - (v) finally, we repeat the first two steps for the refactored schema, in order to ascertain that the transformation has produced the expected result (complexity reduction).

# Introducción

---

- ❑ For the first step (instantiation of the database schema ontology for a given OR schema), we use an ontology developed for the SQL:2003 standard
- ❑ The SQL:2003 ontology is represented as a UML class diagram enforced by the use of OCL invariants.
- ❑ The OR database schema metrics are expressed as OCL operations (selectors) upon the ontology classes

# Introducción

---

- The main benefits of this OCL-based metrics formalization are:
  - the metrics definitions are unambiguous, due to its definition upon the ontology (that provides the context) and OCL (that provides both formality and syntax understandability)
  - the formalization can be used for automating the metrics collection process.

# Indice

---

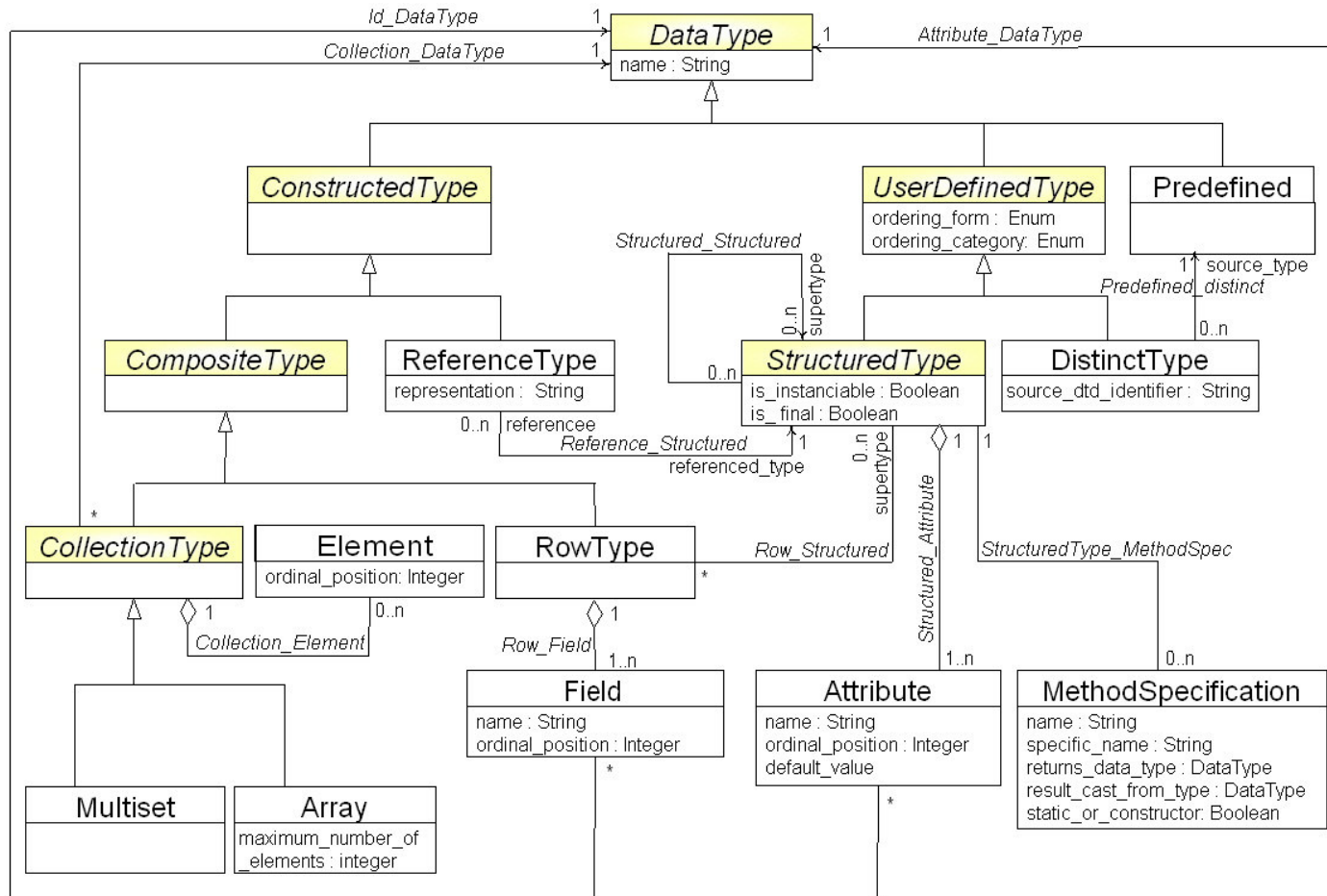
- Introducción
- La ontología del SQL:2003
- Formalización de métricas para BDORel
- Refactorización de BDORel
- Ejemplo
- Conclusiones

# La ontología del SQL:2003

---

- The ontology was constructed using parts 1 (Framework), 2 (Foundation) and 11 (Information and Definition Schema) of the SQL:2003 standard
- The ontology was divided into two parts:
  - One contains the aspects related to data types
  - The other, the information about the SQL schema objects

# La ontología del SQL:2003





# Indice

---

- Introducción
- La ontología del SQL:2003
- Formalización de métricas para BDORel
- Refactorización de BDORel
- Ejemplo
- Conclusiones

# Formalización de métricas para BDORel

---

## □ Table Size (TS)

$TS = TSSC + TSCC$

```
BaseTable:: TS(): Real
= if self.isKindOf(TypedTable)
  then
    self.structuredType.hierarchySize()
  else
    self.TSCC() + self.TSSC()
  endif
```

## □ Table Size Simple Columns (TSSC)

$TSSC = NSC$

```
BaseTable:: TSSC():
Integer
= self.allSimpleColumns()
  -> size()
```

# Formalización de métricas para BDORel

---

## □ Table Size Complex Column (TSCC)

$$TSCC = \sum_{i=1}^{NCC} CCS_i$$

```
BaseTable:: TSCC(): Real
self.allComplexColumns()
-> collect(elem:Column |
elem.CCS()) -> sum
```

## □ Number of Involved Classes (NIC)

```
BaseTable:: NIC(): Integer
= self.involvedClasses() -> size
```

## □ Number of Shared Classes (NSC)

```
BaseTable:: NSC(): Integer
= self.involvedClasses()
-> select(elem: StructuredType |
elem.isShared())
-> size
```

# Formalización de métricas para BDORel

---

- Some auxiliary functions were used in the formal definition of the metrics:
  - Column:: **CCS()**: Real = self.SHC() / self.NCU()
  - Column:: **SHC()**: Real = self.dataType.oclAsType+(StructuredType).SC()
  - Column:: **NCU()**: Integer =self.dataType.oclAsType (StructuredType).columnsNumberUsingThis().self.dataType.oclAsType
  - BaseTable:: **involvedClasses()**:Set(StructuredType)= self.complexColumnTypes() -> union(self.allComplexColumns() -> collect(c: Column| c.columnType().oclAsType(StructuredType). allDependencies()))-> flatten) -> asset

# Indice

---

- Introducción
- La ontología del SQL:2003
- Formalización de métricas para BDORel
- Refactorización de BDORel
- Ejemplo
- Conclusiones

# Refactorización de BDORel

---

- The refactoring process consists of a number of distinct steps, summarized below:
  1. identify where the refactoring(s) should take place;
  2. determine which refactoring(s) should be applied to the identified software fragment;
  3. guarantee that the identified refactoring(s) preserves behavior;
  4. apply the refactoring(s);
  5. assess the effect of the refactoring(s) on software quality characteristics (e.g. understandability, maintainability);
  6. maintain the consistency among the refactored software artifact and other artifacts (for example, if code is refactored, it should be consistent with the documentation, with the original requirements or with a test battery).

# Refactorización de BDORel

---

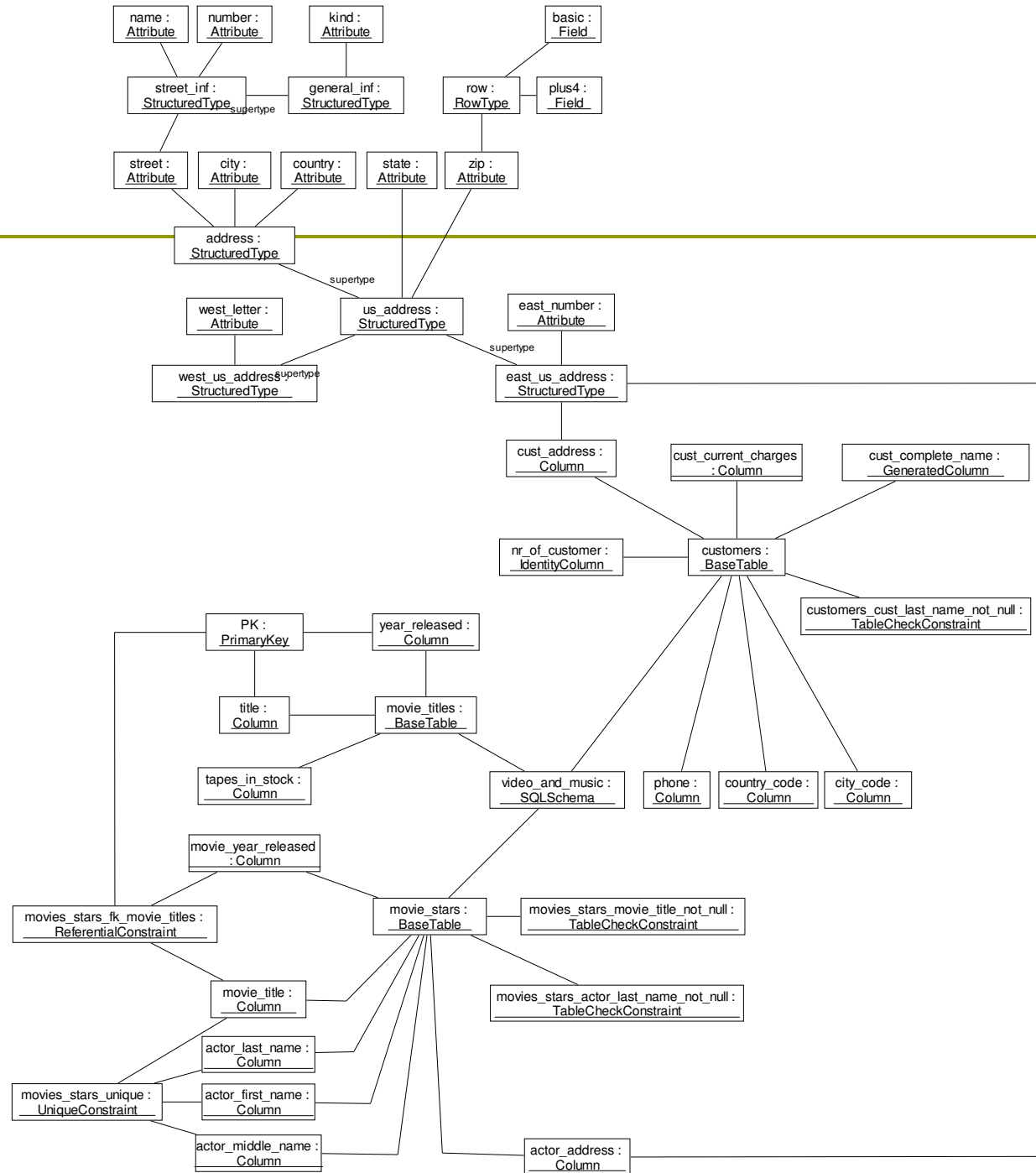
- The refactoring process consists of a number of distinct steps, summarized below:
  1. identify where the refactoring(s) should take place;
  2. determine which refactoring(s) should be applied to the identified software fragment;
  3. guarantee that the identified refactoring(s) preserves behavior;
  4. apply the refactoring(s);
  5. assess the effect of the refactoring(s) on software quality characteristics (e.g. understandability, maintainability);
  6. maintain the consistency among the refactored software artifact and other artifacts (for example, if code is refactored, it should be consistent with the documentation, with the original requirements or with a test battery).

# Indice

---

- Introducción
- La ontología del SQL:2003
- Formalización de métricas para BDORel
- Refactorización de BDORel
- **Ejemplo**
- Conclusiones

# Ejemplo



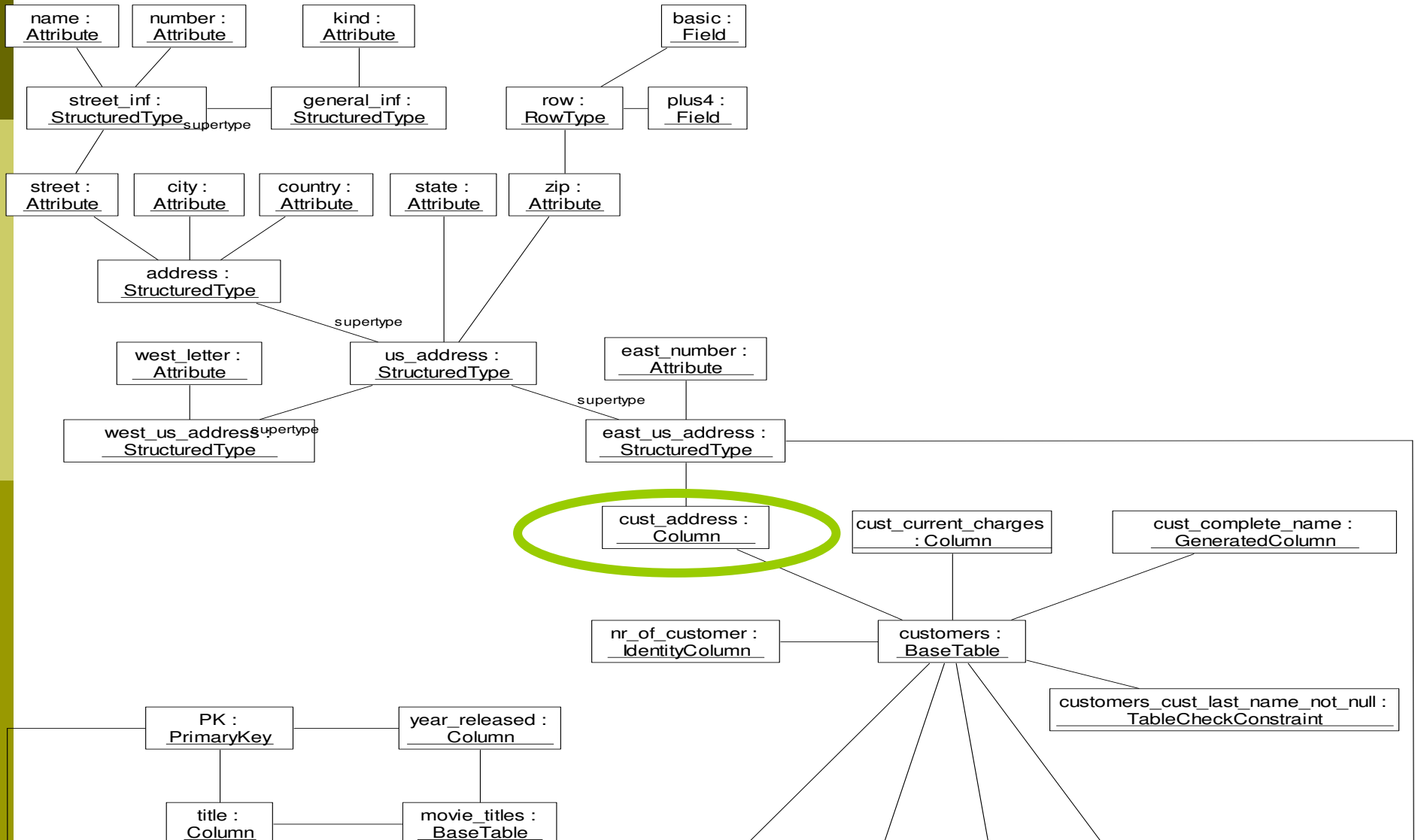
# Ejemplo

---

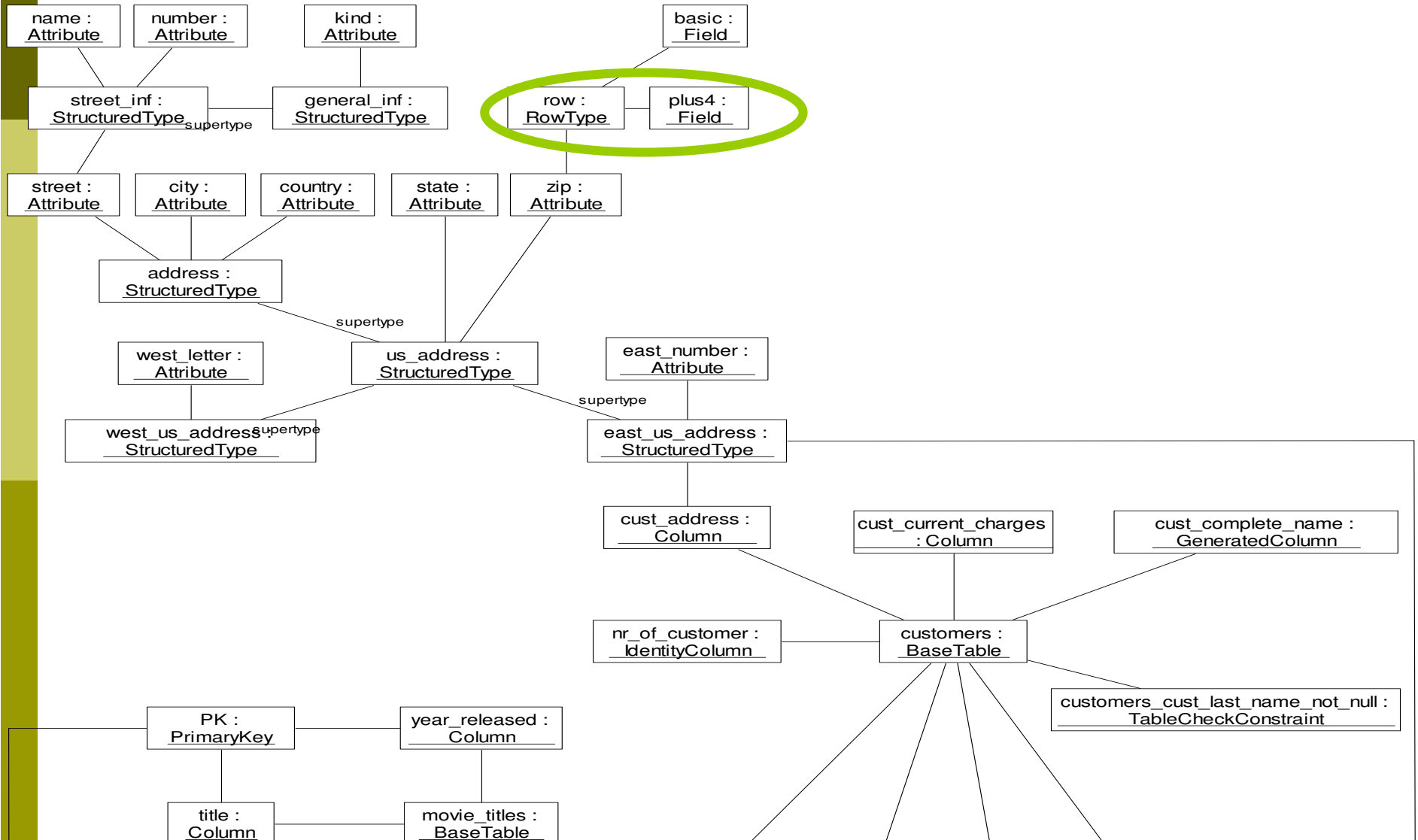
	customers	movie_stars	movie_titles
NIC	5	5	0
NSC	3	3	0
TSSC	6	5	0
TSCC	3	3	3
TS	9	8	3

- ❑ The table size metric (TS) for *customers* and *movie\_stars* has a much higher value compared with *movie\_titles*.
- ❑ Besides, those tables present high coupling values, as shown by NIC and NSC metrics.
- ❑ Therefore, tables *customers* and *movie\_stars* are potential candidates for performing a refactoring.

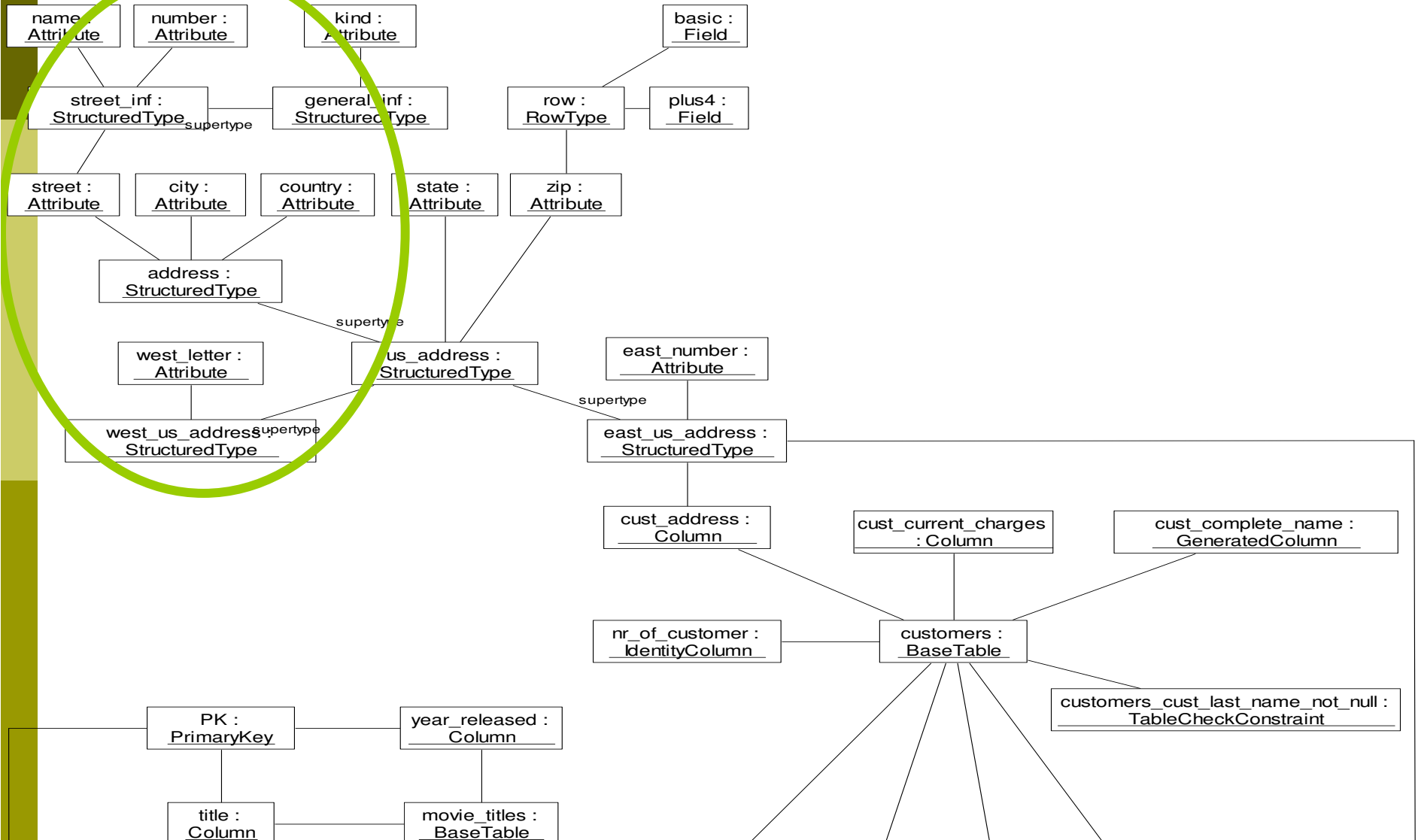
# Ejemplo



# Ejemplo

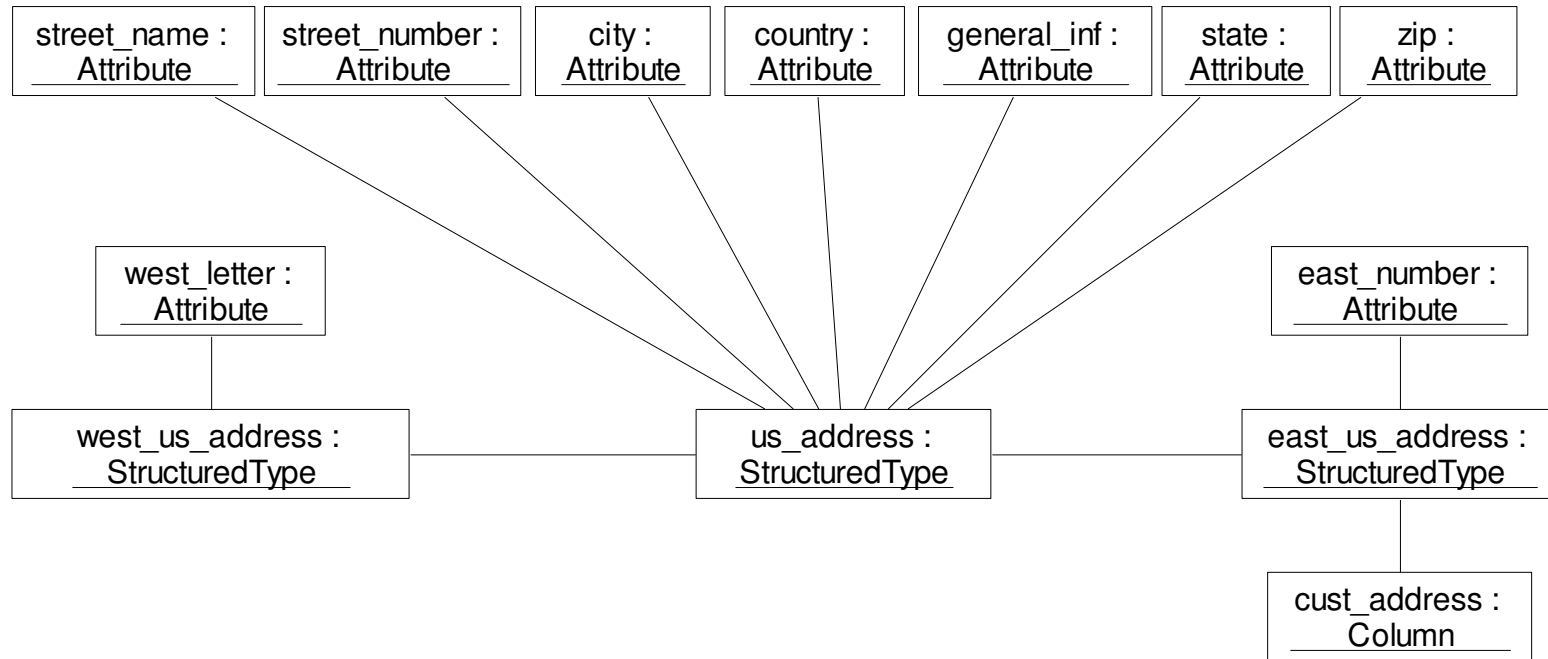


# Ejemplo



# Ejemplo

---



# Ejemplo

- The resulting values, contained in Table 2, compared to those in Table 1, show a reduction both in coupling (NIC and NSC values) and table size, for *customers* and *movie\_stars*.

	customers	movie_stars	movie_titles
<b>NIC</b>	2	2	0
<b>NSC</b>	2	2	0
<b>TSSC</b>	6	5	0
<b>TSCC</b>	2.25	2.25	3
<b>TS</b>	8.25	7.25	3

Metrics values AFTER refactoring

	customers	movie_stars	movie_titles
<b>NIC</b>	5	5	0
<b>NSC</b>	3	3	0
<b>TSSC</b>	6	5	0
<b>TSCC</b>	3	3	3
<b>TS</b>	9	8	3

Metrics values BEFORE refactoring

# Ejemplo

---

- ❑ The selected refactorings depend upon the experience of the database developers, which are responsible for verifying that the schema semantics is not violated.
- ❑ For instance, the elimination of the structured type in step 2 can only be performed if the user requirements allow having only one country.
- ❑ After the refactorings, the metrics are again collected.

# Indice

---

- Introducción
- La ontología del SQL:2003
- Formalización de métricas para BDORel
- Refactorización de BDORel
- Ejemplo
- Conclusiones

# Conclusiones

---

- ❑ Using OCL upon an ontology for OR database schemas, we formalized a set of complexity metrics.
- ❑ This formalization solves the problem of ill definitions and eases the metrics collection process.
- ❑ In this paper we addressed the problem of pinpointing where to apply refactorings and assess if their application was successful, regarding database schema complexity reduction.
- ❑ The preliminary results presented herein are encouraging on the potential of our approach to assist the database schemas refactoring process.

# Conclusiones

---

- ❑ As future work, we plan to research the use of other OR database metrics for the same aim.
- ❑ We also plan to build more elaborated examples for empirically validating the OR schema metrics as indicators of refactoring prone tables.
- ❑ The validation work will use experts' opinion in controlled experiments, by comparing how their proposed refactoring prone tables match with those identified through the metrics.

Finding where to apply object-  
relational database schema  
refactorings: an ontology-guided  
approach

---

