On Suitability of Standard UML Notation for Relational Database Schema Representation

Drazen Brdjanin[(1)], Slavko Maric[(1)], Zvjezdan Spasic Pavkovic[(2)]

[(1)] **University of Banja Luka, Bosnia & Herzegovina**

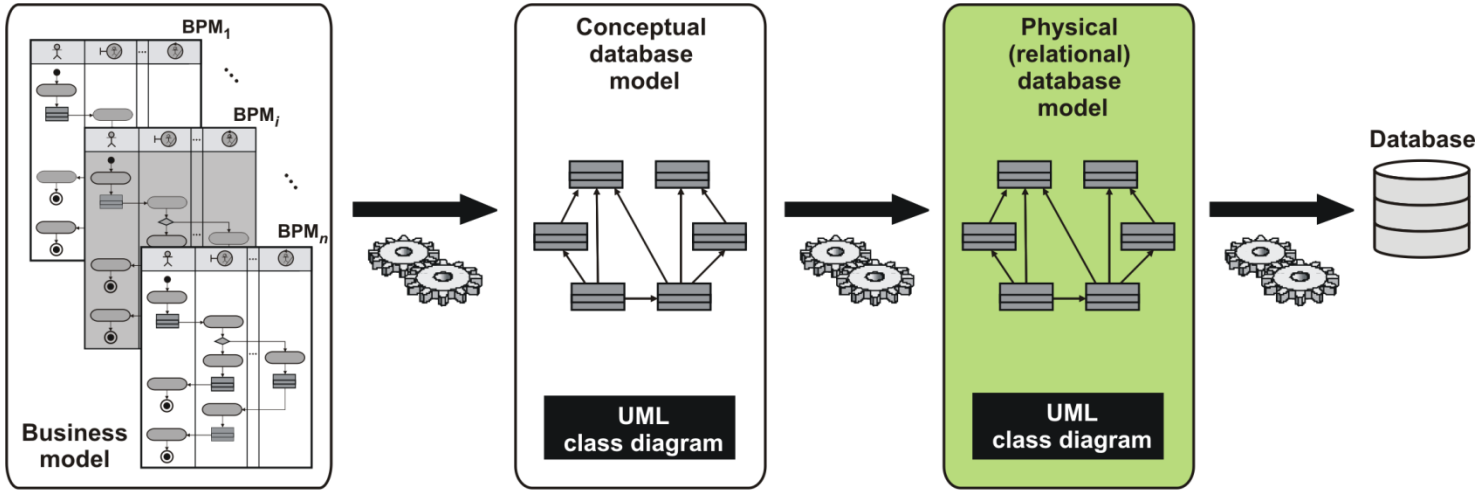
[(2)] RT-RK Institute, Banja Luka, Bosnia & Herzegovina

Presentation outline

- Motivation
- Related work
- Suitability of “isID” meta-attribute
- Our proposal for representation of (complex) keys
- Example of forward database engineering
- Conclusion and Future work

Motivation

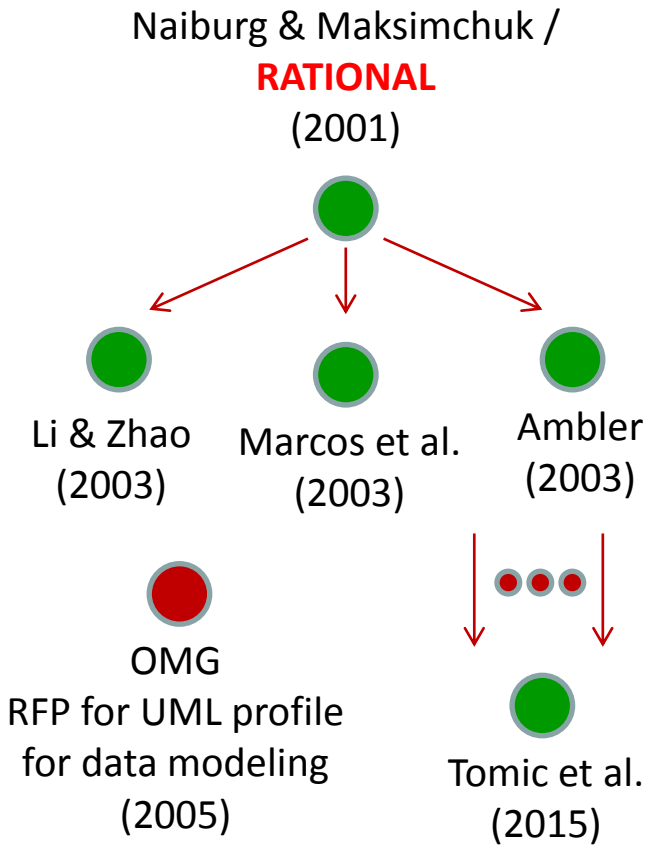
- A part of long-term research devoted to model-driven database design



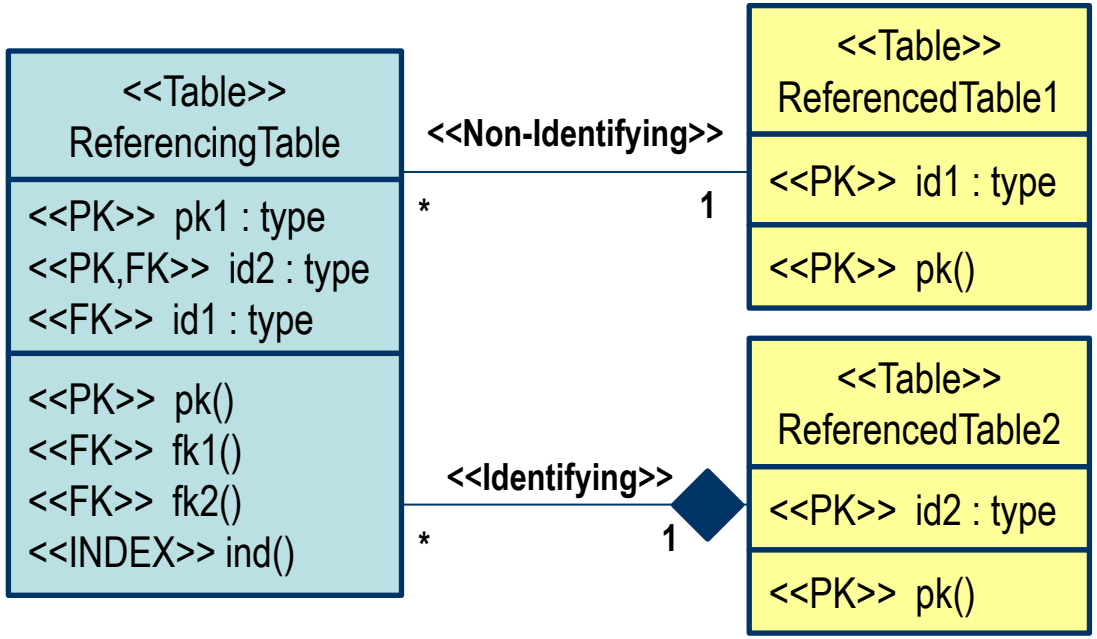
- There is no single and standardized approach to (UML-based) RDBS representation
- The standard UML notation is not fully adapted for RDBS representation – **RDBS represents a model containing some specific concepts which cannot be represented by the standard UML notation?! – UML profile is required?!**
- **Our research question: Is it possible to represent RDBS by standard UML?**

Related work

- The UML-based RDBS representation has been the subject of research since the beginning of UML development



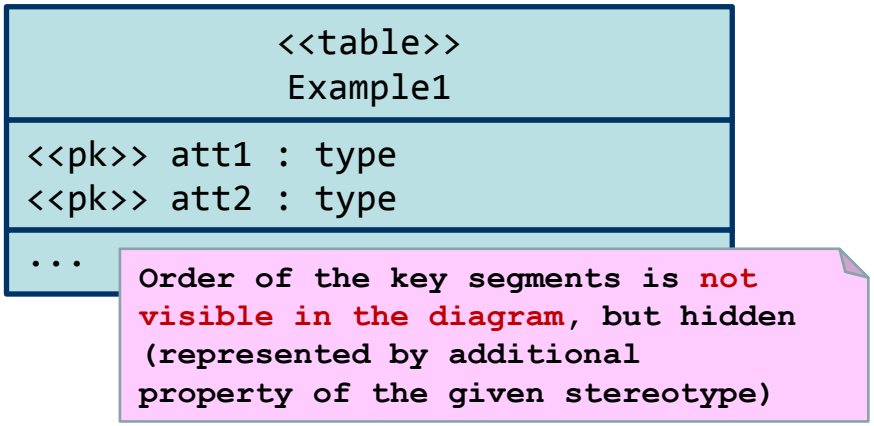
Typical UML stereotypes for RDBS representation
(Naiburg & Maksimchuk)

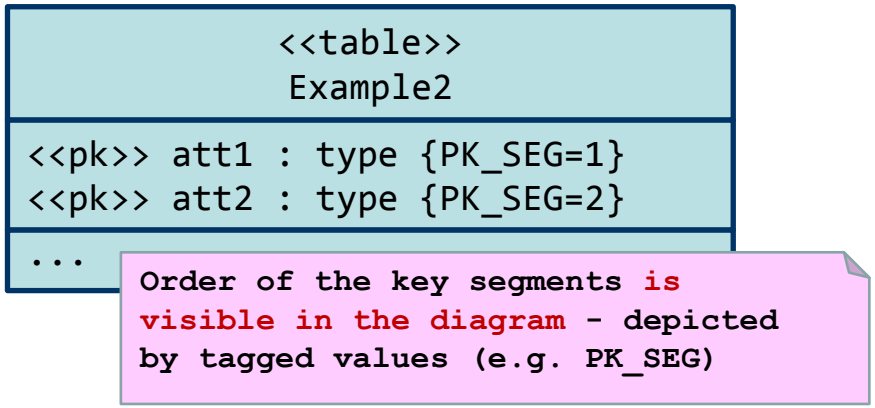


Related work

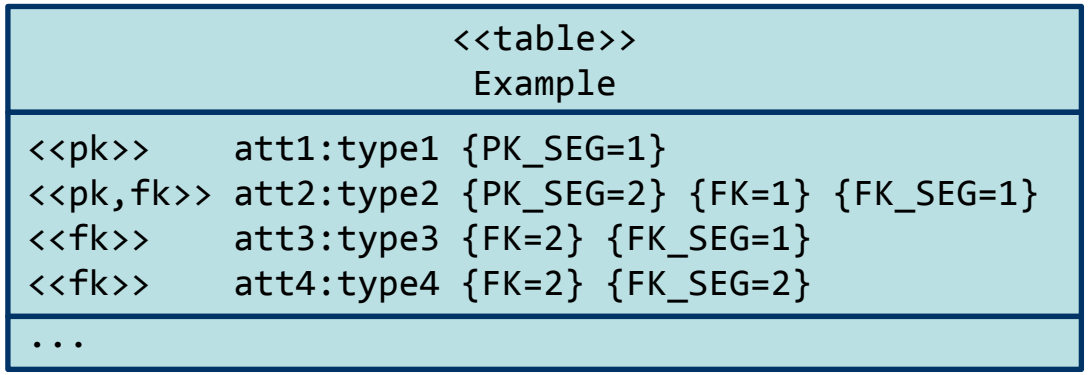
Difficulties with the existing approaches?

- **Representation of complex keys**





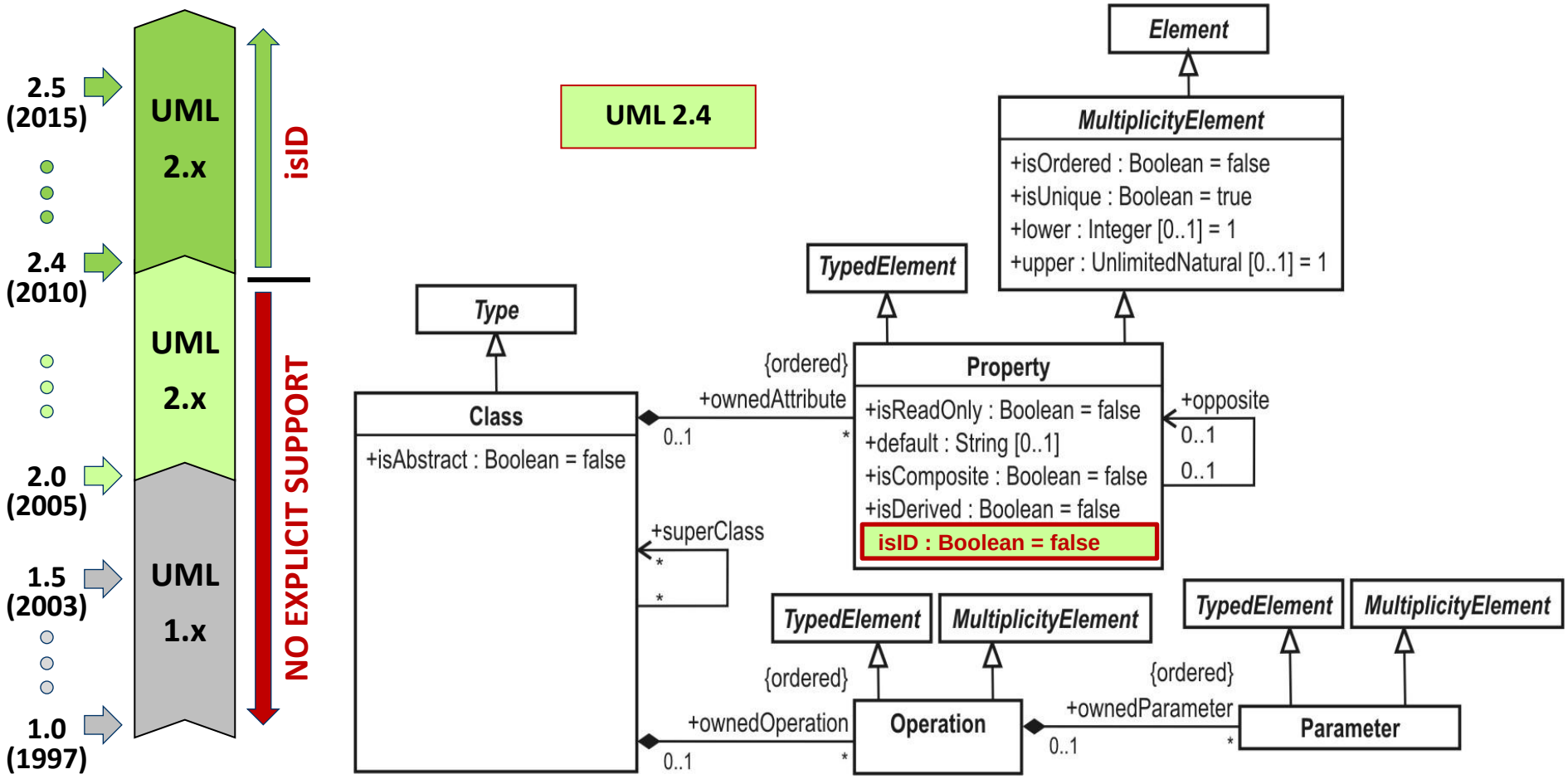
Even more difficulties with the existing approaches?



Our research on “RDBS by standard UML”

- Our research question: **Is it possible to represent RDBS by standard UML?**

  Focus in this paper: REPRESENTATION OF KEYS by standard UML notation

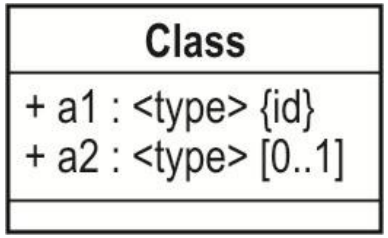


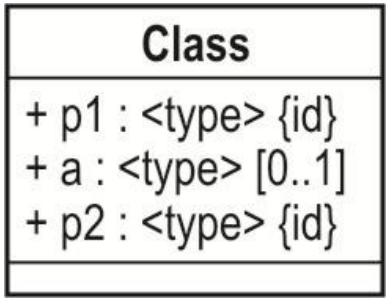
“isID” meta-attribute

- Its **default value is false** => class **attributes are not identifiers by default**.
- If **isID=true** for some class attribute => **the given attribute can be used to uniquely identify an instance** of the containing class.
- It is suggested (UML spec.) that **this could be mapped to** implementations such as **primary keys** for RDB tables.
- The fact that some attribute is marked as (a part of) an identifier (i.e. **isID=true**) is represented by the **{id}** modifier.

Is it possible to represent a PRIMARY KEY by setting isID=true for all attributes belonging to the primary key?

Simple PK



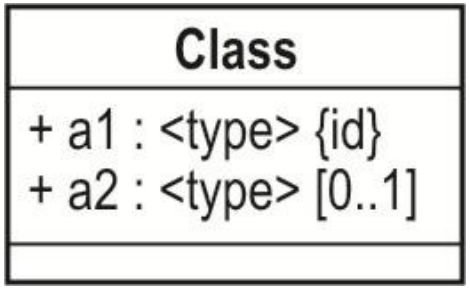


Composite PK

“isID” meta-attribute

Simple PK

- Trivial case! **PK=(a) => PK=a**
- The lower multiplicity for {id} attribute must be 1.



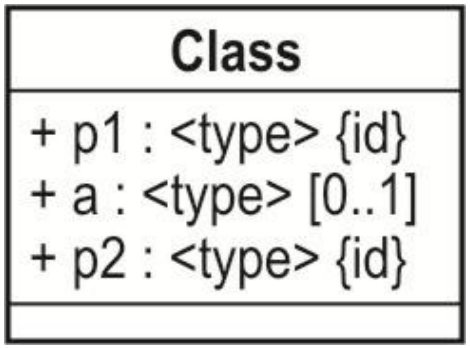

```
CREATE TABLE Class
(
   a1 <type> NOT NULL PRIMARY KEY,
   a2 <type>
);
```

Acceleo transformation program

```
...
[for (aClass:Class | aPack.ownedElement) after('\n')]
CREATE TABLE [aClass.name/]
(
   [for (aProp:Property | aClass.ownedAttribute)
       separator(',\n') after('\n')]
   [aProp.name/] [aProp.type.name/][if (aProp.lower>0)]
  NOT NULL[/if][if (aProp.isID)] PRIMARY KEY[/if][/for]
);
[/for]
...
```

“isID” meta-attribute

Composite PK



- **PK = (p_1, p_2, …)** - **PK is ordered sequence**
- According to the UML spec. the combination of corresponding `(property,value)` tuples logically provides the uniqueness for any instance. **There is no need for additional specification of order?!**

  ID={(p_1,v_1), … , (p_k, v_k)} - **ID is unordered set**

Whether the given set of attributes {p_1, … , p_k} may represent the primary key of the relation schema modeled by the given class?

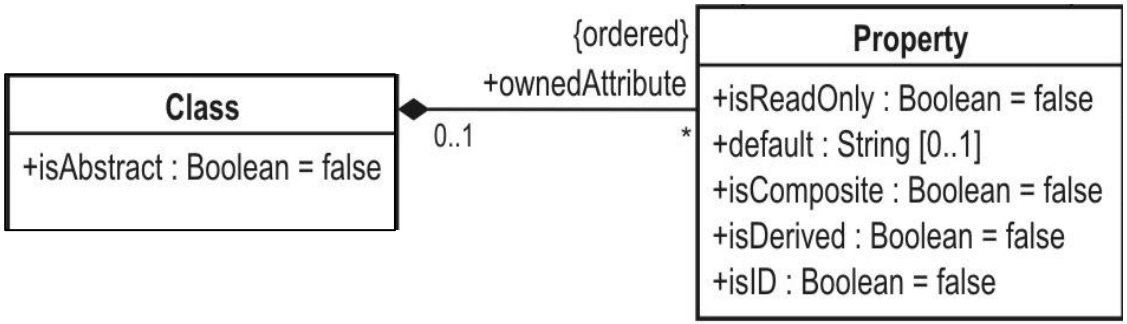
The answer lies in the serialization of UML models.

Namely, **if the physical order of class attributes (serialized model) is equal to the logical order (visualized on the diagram), then the set of {id} attributes can be used as a primary key**.

“isID” meta-attribute

Composite PK

- **Serialization of UML models is standardized by the XMI**.
- XMI specifications define serialization of composite elements in the form of XML elements, where the order of corresponding XML elements in the model corresponds to the order of attributes in the diagram.

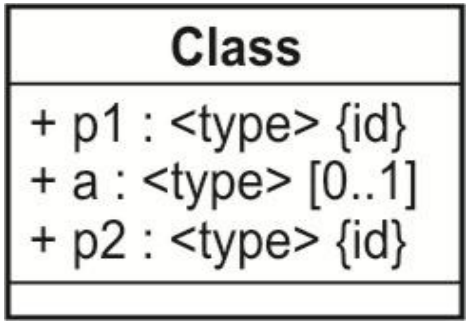


- **XMI-based serialization makes it possible that set of {id} class attributes can be used as a primary key of the relation schema represented by given class, since the set of {id} attributes is ordered**, i.e.

$$\{p_1, \ldots, p_k\} = (p_1, \ldots, p_k).$$

“isID” meta-attribute

Composite PK




```
CREATE TABLE Class
(
    p1 <type> NOT NULL,
    a <type>,
    p2 <type> NOT NULL,
    PRIMARY KEY (p1, p2)
);
```

Acceleo transformation program

```
...
[for (aClass:Class | aPackage.ownedElement) separator('\n')]
CREATE TABLE [aClass.name/]
(
  [for (aProp:Property | aClass.ownedAttribute)]
    [aProp.name/] [aProp.type.name/][if (aProp.lower>0)]
 NOT NULL[/if],
[/for]
[if (aClass.ownedAttribute->select(o | o.isID)->notEmpty())]
PRIMARY KEY [for
(aProp:Property | aClass.ownedAttribute->select(o | o.isID))
before('(') separator(', ')after(')\n')][aProp.name/][/for][/if]
);
[/for]
...
```

“isID” meta-attribute

Is it possible to represent a PRIMARY KEY by setting isID=true for all attributes belonging to the primary key?

YES! It is possible to represent a PRIMARY KEY by setting isID=true for all attributes belonging to the primary key.

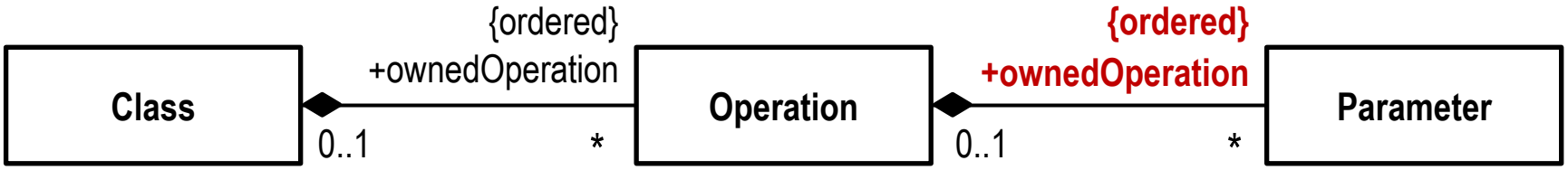
BUT:

- **Some modeling tools**, including some open source development platforms, **still** do not support the recent UML specifications and **do not allow representation of the primary key by applying the isID (meta)attribute**.
- **How to represent foreign keys?**
- How to represent other RDBS specific concepts?

Are there alternative options for representation of RDBS specific concepts by applying standard UML notation, without extending the UML metamodel?

Our proposal for representation of keys

- **UML possesses another inherent (but not explicit) mechanism that also provides the possibility for very simple and efficient representation of keys.**



KEY is ordered sequence of colums. **Operation parameters constitute the sequence.**

$key(p_1, \ldots, p_k) \Leftrightarrow operation(op_1{:}type, \ldots, op_k{:}type)$

KEYS can be modeled by OPERATIONS!
Key segments $p_1,\ldots,p_k$ are represented by corresponding operation parameters $op_1,\ldots,op_k$.

Primary key:

$PK(p_1, \ldots, p_k)$

Foreign key:

$FK(f_1, \ldots, f_k)$

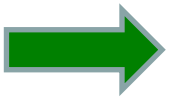
Table
a1 : type1 a2 : type2
PK(a1:type1, a2:type2)

Our proposal for representation of keys

Example

Existing approaches

<<table>> Table
<<pk>> a1:t1 {PK_SEG=1} <<pk,fk>> a2:t2 {PK_SEG=2} {FK=1} {FK_SEG=1} <<fk>> a3:t3 {FK=2} {FK_SEG=1} <<fk>> a4:t4 {FK=2} {FK_SEG=2}
...



Our solution based on **the operation signature semantics!**

Table
a1 : t1 a2 : t2 a3 : t3 a4 : t4
PK(a1:t1, a2:t2) **FK_1(a2:t2)** **FK_2(a3:t3, a4:t4)**

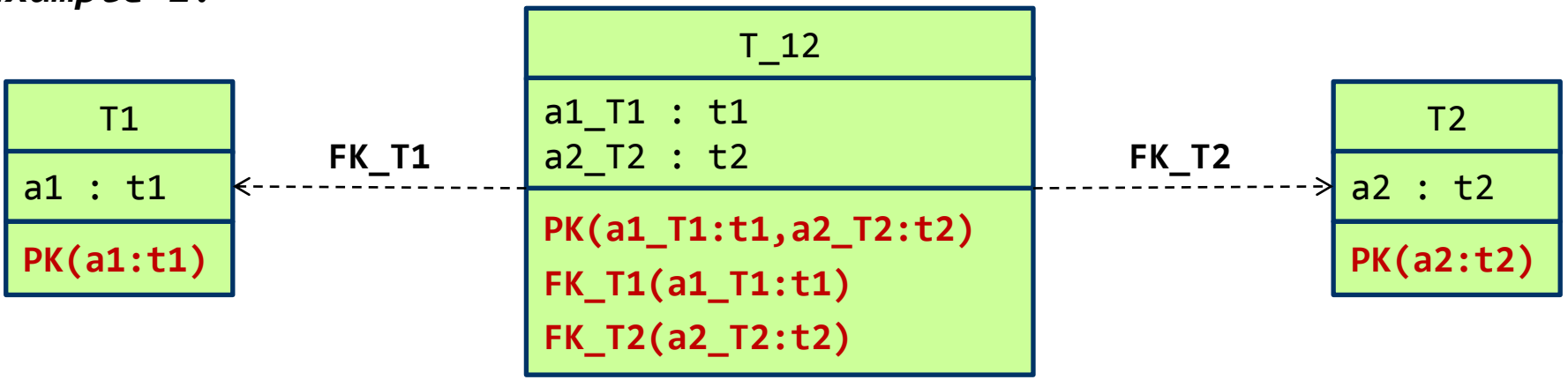
Direct advantages of our approach:

- RDBS is represented by **standard UML notation;**
- There is **no need to define and apply the specific profile**;
- **Modeling is easier and faster** than using the specialized notation;
- **Visualization is better** (without specific stereotypes, order of operation parameters represents the key's segments, order of the key segments is visible in the diagram).

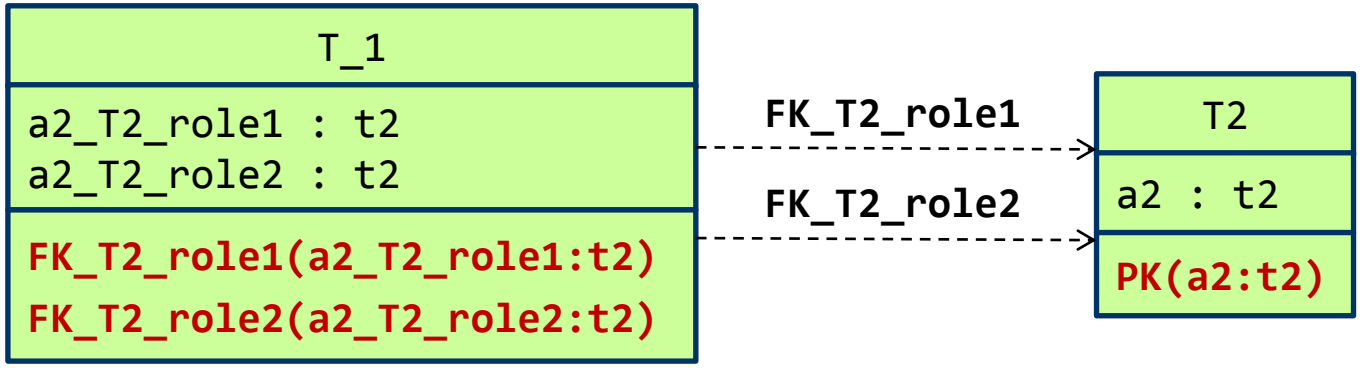
Our proposal for representation of keys

How to deal with multiple foreign keys?

Example 1:



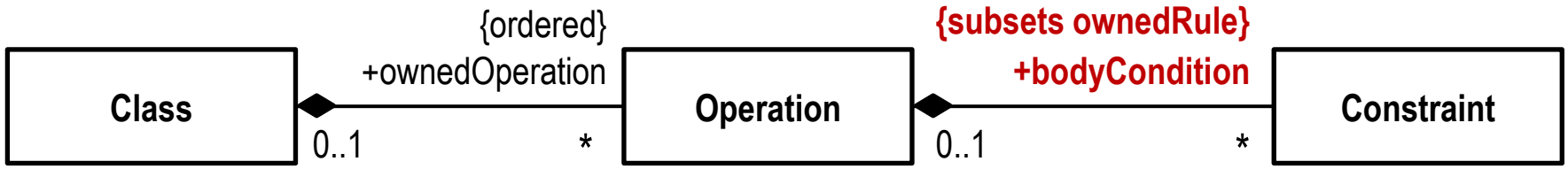
Example 2:



Our proposal for representation of keys

How to specify referential actions for foreign keys?

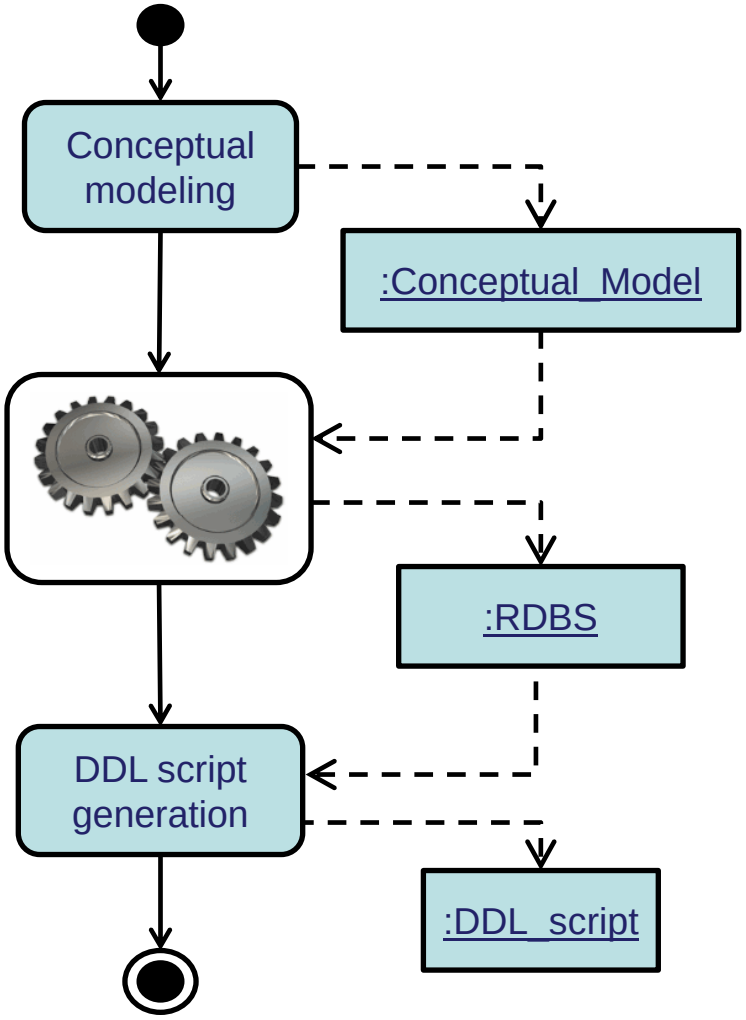
- UML allows specification of operation constraints.
- A set of constraints (`ownedRule`) can be specified for each operation.



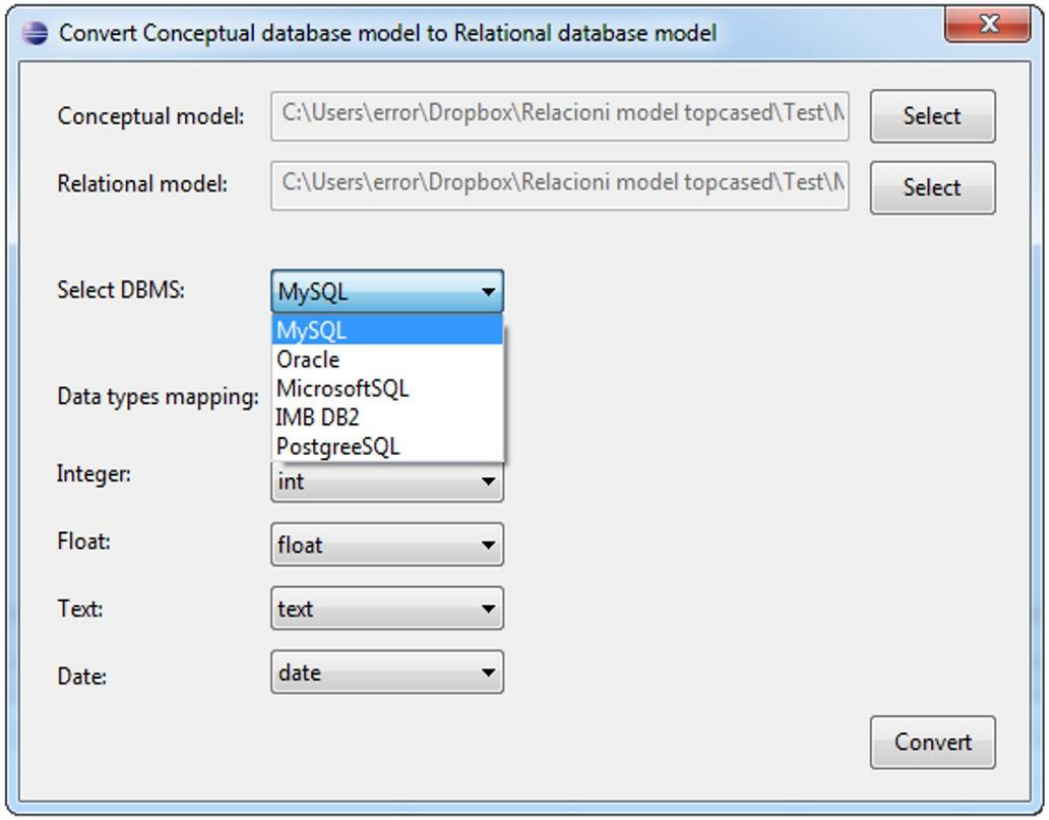
- **In our case, the appropriate constraint specifiying the referential integrity actions, is to be defined for each operation representing the foreign key.**
- Although constraints can be specified by using the OCL, they can be specified in another way, as well.
- **The easiest way is to directly specify the DDL statement part**, e.g.

```
ON DELETE RESTRICT ON UPDATE CASCADE
```

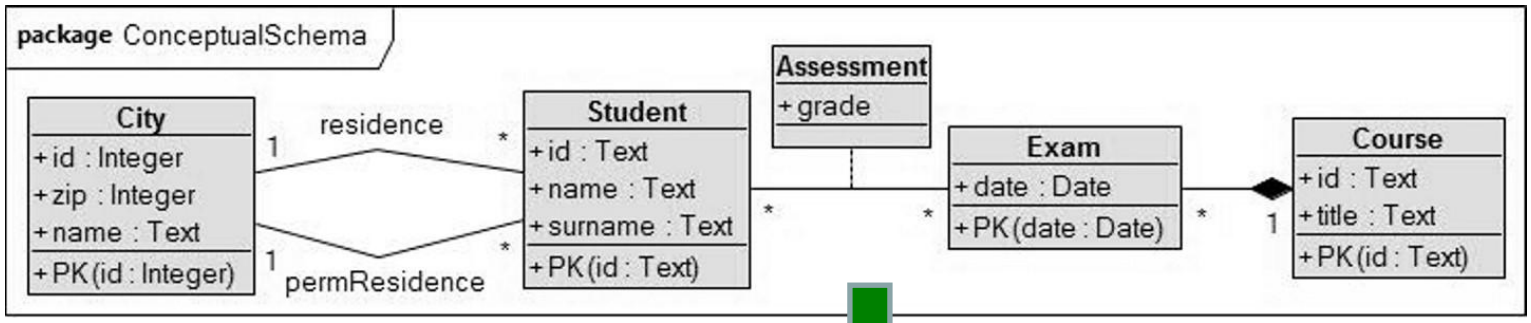
Example of forward database engineering

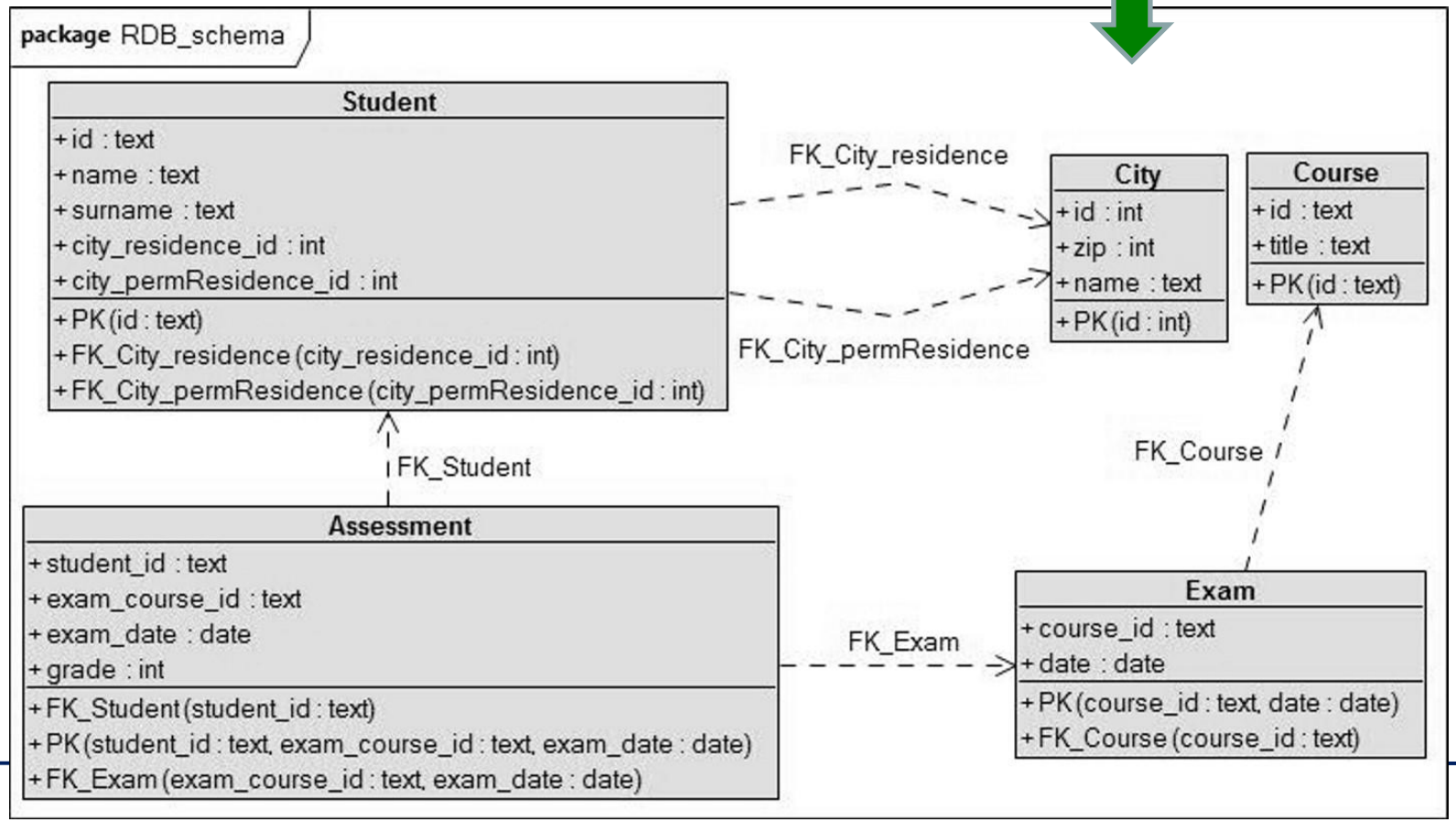


ECLIPSE-TOPCASED
plug-in

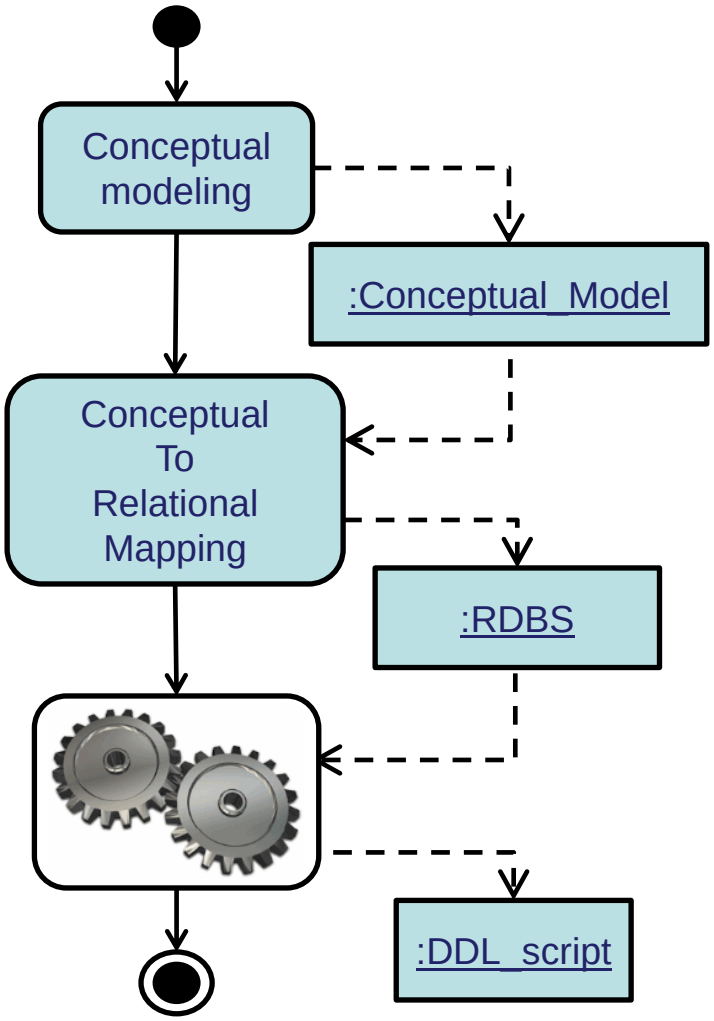


Example of forward database engineering





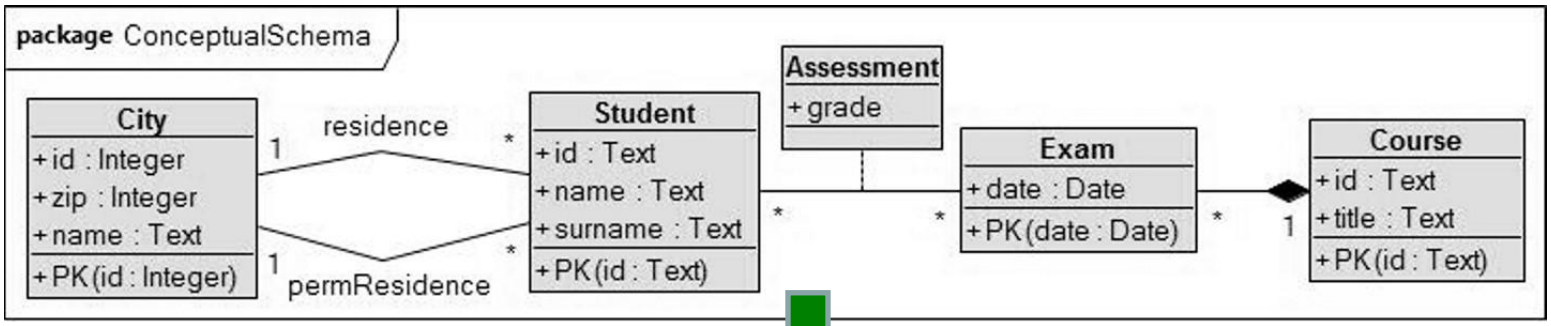
Example of forward database engineering




```
[comment encoding = UTF-8 /]
[module generate('http://www.eclipse.org/uml2/3.0.0/UML')]
[template public generateElement(aPackage : Package)]
[comment @main/]
[file (aPackage.name.concat('.ddl'), false, 'UTF-8')]
CREATE SCHEMA  [aPackage.name/];
 [for (aClass:Class | aPackage.ownedElement) after('\n')]
CREATE TABLE [aPackage.name.concat('.').concat(aClass.name)/]
(
 [for (aProperty:Property | aClass.ownedAttribute) separator('\n') after('\n')]
 [aProperty.name/] [aProperty.type.name/][if (aProperty.lower>0)] NOT NULL[/if],[/for]
 [for (aOperation:Operation | aClass.ownedOperation->
        select(o | o.name.startsWith('FK') or o.name.equalsIgnoreCase('PK')) )
  separator(',\n') after('\n')]
   [if (aOperation.name.equalsIgnoreCase('PK'))]
 PRIMARY KEY ([for (op:Parameter | aOperation.ownedParameter)
   separator(', ')][op.name/][/for])[/if]
   [if (aOperation.name.startsWith('FK'))]
 CONSTRAINT [aOperation.name/]
  FOREIGN KEY ([for (op:Parameter | aOperation.ownedParameter)
   separator(', ')][op.name/][/for])
  REFERENCES [for (aDependency:Dependency | aPackage.ownedElement)]
    [if (aDependency.name.equalsIgnoreCase(aOperation.name))]
[aPackage.name.concat('.')/][aDependency.supplier.name/][/if][/for]
[for (r:Constraint | aOperation.ownedRule)] [r.name/][/for][/if]
 [/for]
);
[/for] [/file]
[/template]
```

Acceleo transformacioni program (30 LOC!!!)

Example of forward database engineering




```
CREATE SCHEMA RDB_schema;
CREATE TABLE RDB_schema.Assessment
(
   student_id text NOT NULL,
   exam_course_id text NOT NULL,
   exam_date date NOT NULL,
   grade int NOT NULL,
   PRIMARY KEY (student_id, exam_course_id, exam_date),
   CONSTRAINT FK_Student
      FOREIGN KEY (student_id)
      REFERENCES RDB_schema.Student ON DELETE RESTRICT ON UPDATE CASCADE,
   CONSTRAINT FK_Exam
      FOREIGN KEY (exam_course_id, exam_date)
      REFERENCES RDB_schema.Exam ON UPDATE CASCADE ON DELETE RESTRICT
);
...
```

Conclusion

- The suitability of the standard UML notation for RDBS representation has been considered. Firstly, the suitability of the built-in mechanism (isID meta-attribute) for representation of primary keys has been analyzed.
- **It has been shown that the isID meta-attribute can be used to represent the primary key, regardless of whether it is simple or composite.**
- UML still has no built-in mechanism dedicated to representation of other important RDBS concepts (foreign keys, etc).
- We proposed an **alternative representation of composite keys by using class operations.** The main idea of the proposed approach is based on the fact that the **standardized order of operation parameters can be used to represent the order of key segments** (primary, foreign, alternate).

Conclusion

- Our approach has several direct advantages compared to existing approaches:
  - RDBS is represented by standard UML notation and there is no need to define and apply the specific profile;
  - modeling of RDBS is easier and faster than using the specialized notation;
  - visualization of RDBS is better;
  - forward database engineering is easier and more efficient.
- The preliminary results of applying the proposed approach show that the fundamental RDBS concepts can be represented by standard UML notation in an easy and intuitive way. Furthermore, the implemented tool provides good visualization of the automatically generated and/or manually modeled RDBS, as well as simple generation of DDL script and forward database engineering for several contemporary DBMSs.
- In the future work we will focus on:
  - further analysis of the suitability of standard UML notation to represent other important RDBS-related concepts (indices, views, queries, triggers, etc.),
  - integrating the tool in the ADBdesign tool for automated business MDD of RDBS.

On Suitability of Standard UML Notation for Relational Database Schema Representation

Thank You!