Checking Transformation Model Properties with a UML and OCL Model Validator

Martin Gogolla, Lars Hamann, Frank Hilken
University of Bremen, Database Systems

Motivation and context

- consider model transformations in the form of transformation models connecting source and target metamodels

- analyze transformation models with a UML and OCL tool (model validator) on basis of relational logic on top of Kodkod

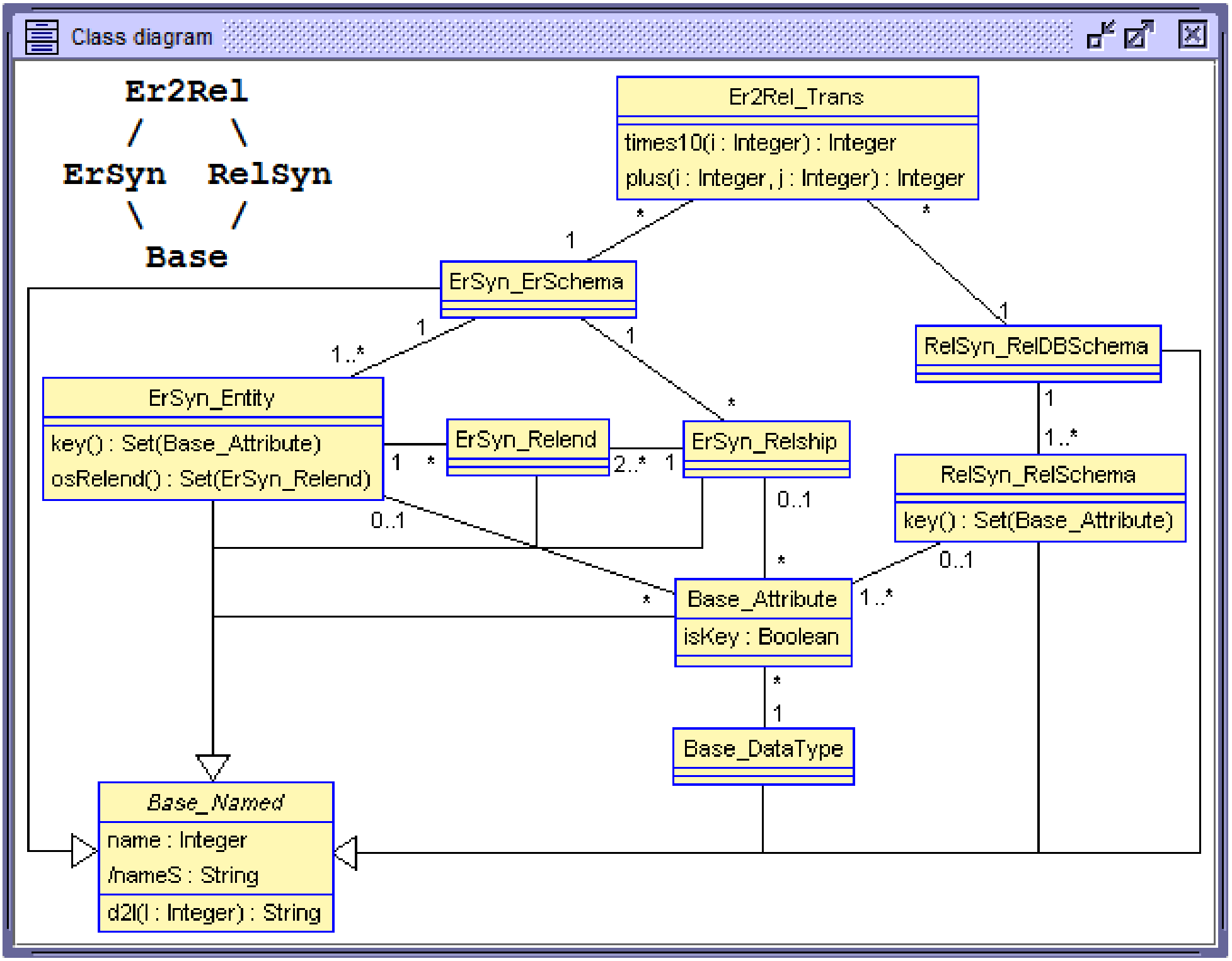
- central technique: automatically construct object diagrams for UML model with OCL invariants

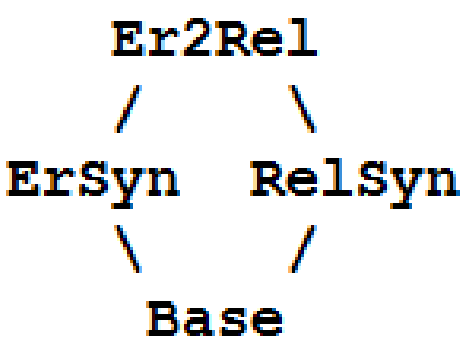
- prove transformation model properties

  - transformation consistency, i.e., to automatically construct a valid transformation metamodel instance

  - transformation properties implied by the transformation model are inspected as well, e.g., study whether a property is preserved by the transformation,

- example: well-known transformation between ER schemata and relational database schemata

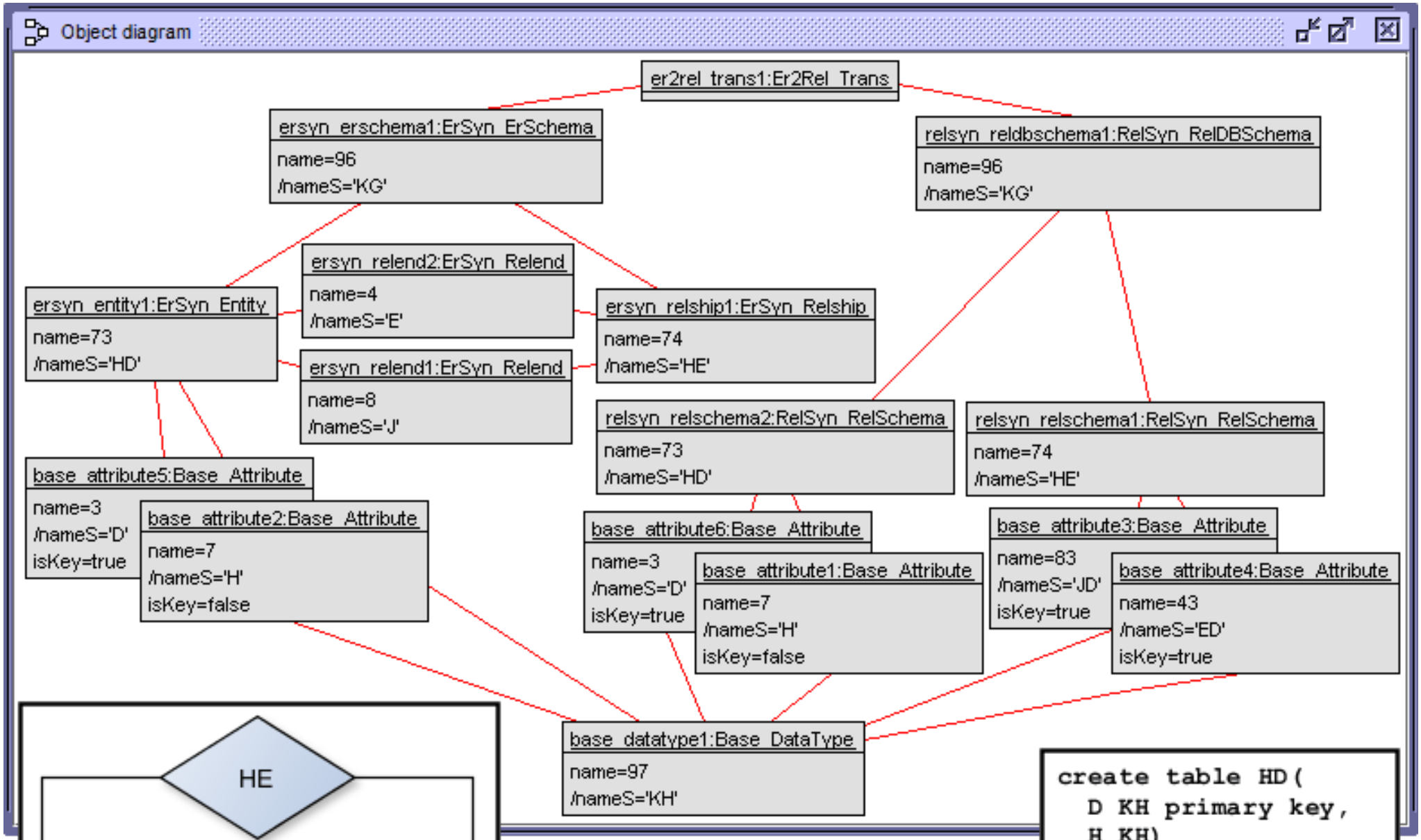

Class diagram
Er2Rel
ErSyn
RelSyn
Base
Er2Rel_Trans
times10(i : Integer) : Integer
plus(i : Integer, j : Integer) : Integer
ErSyn_ErSchema
RelSyn_RelDBSchema
ErSyn_Entity
key() : Set(Base_Attribute)
osRelend() : Set(ErSyn_Relend)
ErSyn_Relend
ErSyn_Relship
RelSyn_RelSchema
key() : Set(Base_Attribute)
Base_Attribute
isKey : Boolean
Base_DataType
Base_Named
name : Integer
/nameS : String
d2l(l : Integer) : String

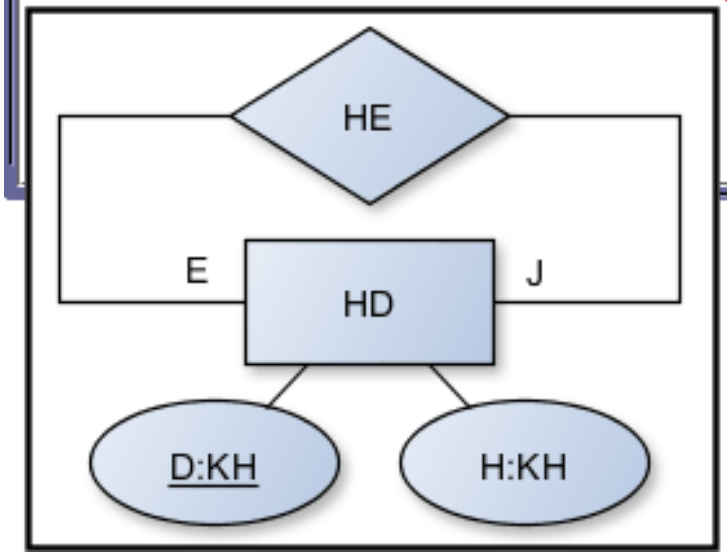


Class invariants

Invariant	Result
Base_Attribute::linkedToOneOfEntityRelshipRelSchema	true
Base_DataType::uniqueDataTypeNames	true
Er2Rel_Trans::forEntityExistsOneRelSchema	true
Er2Rel_Trans::forRelSchemaExistsOneEntityXorRelship	true
Er2Rel_Trans::forRelshipExistsOneRelSchema	true
ErSyn_Entity::differentOsRelendAndAttributeNamesWithinEntity	true
ErSyn_Entity::entityKeyNotEmpty	true
ErSyn_Entity::uniqueAttributeNamesWithinEntity	true
ErSyn_Entity::uniqueOsRelendNamesWithinEntity	true
ErSyn_ErSchema::differentEntityAndRelshipNamesWithinErSchema	true
ErSyn_ErSchema::uniqueEntityNamesWithinErSchema	true
ErSyn_ErSchema::uniqueErSchemaNames	true
ErSyn_ErSchema::uniqueRelshipNamesWithinErSchema	true
ErSyn_Relend::c_Relend_Entity_Relship_ErSchema	true
ErSyn_Relship::differentRelendAndAttributeNamesWithinRelship	true
ErSyn_Relship::relshipKeyEmpty	true
ErSyn_Relship::uniqueAttributeNamesWithinRelship	true
ErSyn_Relship::uniqueRelendNamesWithinRelship	true
RelSyn_RelDBSchema::uniqueRelDBSchemaNames	true
RelSyn_RelDBSchema::uniqueRelSchemaNamesWithinRelDBSchema	true
RelSyn_RelSchema::relSchemaKeyNotEmpty	true
RelSyn_RelSchema::uniqueAttributeNamesWithinRelSchema	true

Constraints ok. (0ms) 100%






```
create table HD(
  D KH primary key,
  H KH)
create table HE(
  ED KH,
  JD KH,
  primary key(ED,JD))
```

Transformation ER 2 RelMod

Encoding of strings as integers:
20=2*10+0 == '2.concat(0)' == 'C'.concat('A')='CA'

```
Er2Rel_Trans : 1..1          -- (a)

Er2Rel_OwnershipTransErSchema    : *
Er2Rel_OwnershipTransRelDBSchema : *
```

```
ErSyn_ErSchema : 1..1
ErSyn_Entity   : 1..1
ErSyn_Relship  : 1..1
ErSyn_Relend   : 2..2

ErSyn_OwnershipErSchemaEntity   : *          -- (b)
ErSyn_OwnershipErSchemaRelship  : *
ErSyn_OwnershipEntityAttribute  : 2..2
ErSyn_OwnershipRelshipAttribute : 0..0       -- (b)
ErSyn_OwnershipRelshipRelend    : *
ErSyn_RelendTyping              : *
```

```
RelSyn_RelDBSchema : 1..1
RelSyn_RelSchema   : 2..2          -- (a)


RelSyn_OwnershipRelDBSchemaRelSchema : *
RelSyn_OwnershipRelSchemaAttribute   : 4..4
```

```
Base_Attribute       : 6..6
Base_DataType        : 1..1
Base_Named_name      : Set{0,1,2,3,4,5,6,7,8,9,10, ..., 89,90,91,92,93,94,95,96,97,98,99}
Base_Attribute_isKey : Set{false,true}          -- (c)

Base_AttributeTyping : *

Real      : 0..0
Real_step : 1
String    : 0..0
Integer   : 0..127                              -- (d)
```

```
Class       black-on-white
Association black-on-light-grey
```

Upper bounds for classes mandatory

	weak consistency	class instantiability	class and association instantiability
Class	0..UpperBound	1..Upperbound	1..UpperBound
Association	0..*	0..*	1..UpperBound

three configurations to check for

(1) weak consistency
(2) class instantiability
(3) class and association instantiability

(1) means that at least one valid object diagram can be found, even with no objects at all or empty populations for a single class, provided the model and the configuration allows this

(2) means all classes are instantiated with non-empty populations

(3) means that all classes and all associations are instantiated with non-empty populations

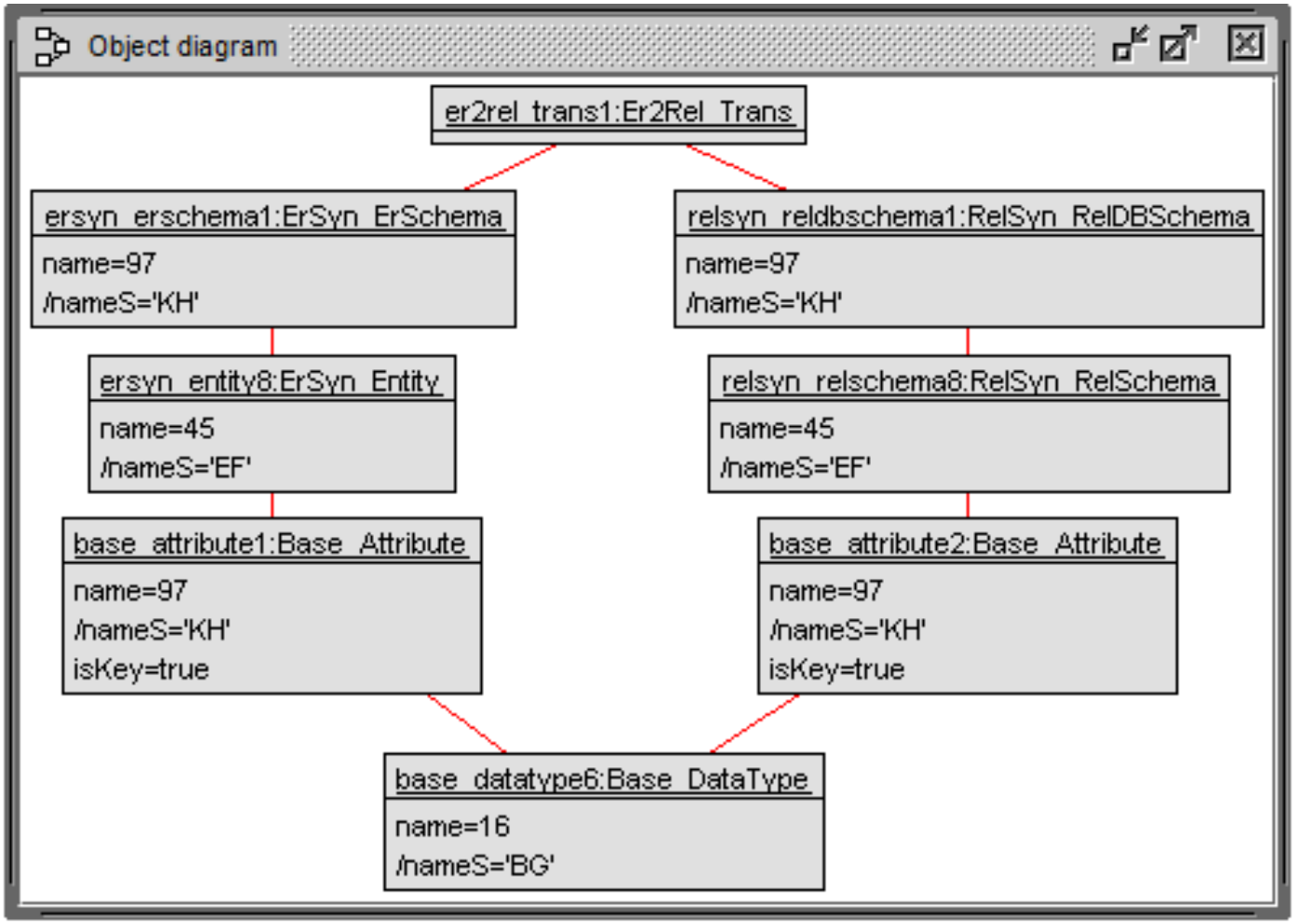
```
Er2Rel_Trans : 1..1

Er2Rel_OwnershipTransErSchema    : 1..1
Er2Rel_OwnershipTransRelDBSchema : 1..1
```

```
ErSyn_ErSchema : 1..1
ErSyn_Entity   : 0..9 | 1..9 | 1..9
ErSyn_Relship  : 0..9 | 1..9 | 1..9
ErSyn_Relend   : 0..9 | 1..9 | 1..9

ErSyn_OwnershipErSchemaEntity   : *    | *    | 1..9
ErSyn_OwnershipErSchemaRelship  : *    | *    | 1..9
ErSyn_OwnershipEntityAttribute  : *    | *    | 1..9
ErSyn_OwnershipRelshipAttribute : *    | *    | 1..9
ErSyn_OwnershipRelshipRelend    : *    | *    | 1..9
ErSyn_RelendTyping              : *    | *    | 1..9
```

```
RelSyn_RelDBSchema : 1..1
RelSyn_RelSchema   : 0..9 | 1..9 | 1..9

RelSyn_OwnershipRelDBSchemaRelSchema : *    | *    | 1..9
RelSyn_OwnershipRelSchemaAttribute   : *    | *    | 1..9
```

```
Base_Attribute       : 0..9 | 1..9 | 1..9
Base_DataType        : 0..9 | 1..9 | 1..9
Base_Named_name      : Set{0,1,2,3,4,5,6,7,8,9,10, ..., 89,90,91,92,93,94,95,96,97,98,99}
Base_Attribute_isKey : Set{false,true}

Base_AttributeTyping : *    | *    | 1..9

Real      : 0..0
Real_step : 1
String    : 0..0
Integer   : 0..127
```

```
Class       black-on-white          single interval black-on-white
Association black-on-light-grey     triple interval white-on-dark-grey
```

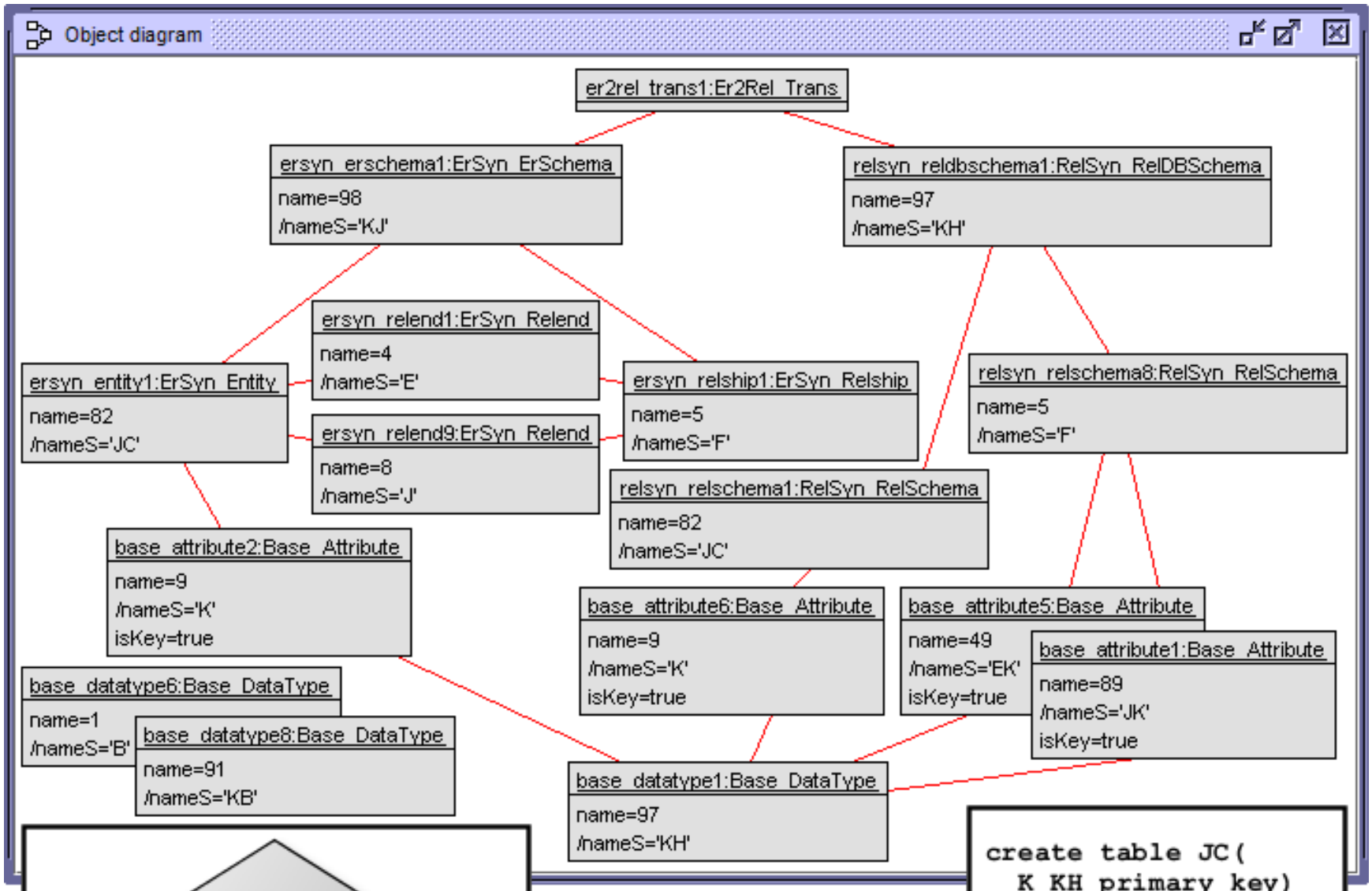


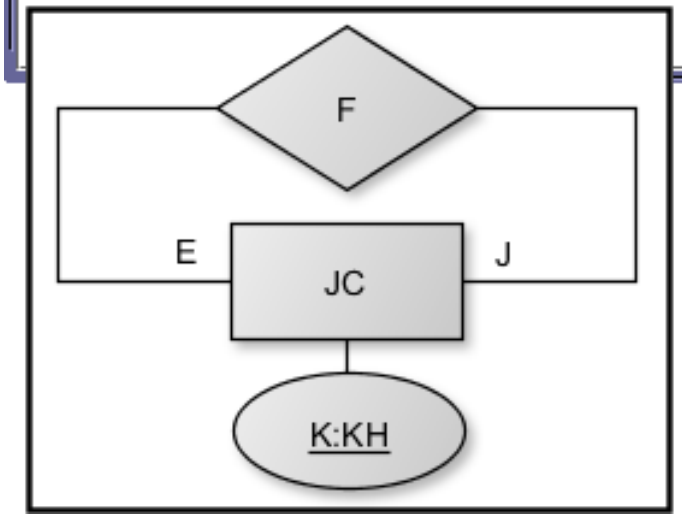
Object count

Class	# Objects
Base_Attribute	2
Base_DataType	1
Base_Named	0
Er2Rel_Trans	1
ErSyn_Entity	1
ErSyn_ErSchema	1
ErSyn_Relend	0
ErSyn_Relship	0
RelSyn_RelDBSchema	1
RelSyn_RelSchema	1

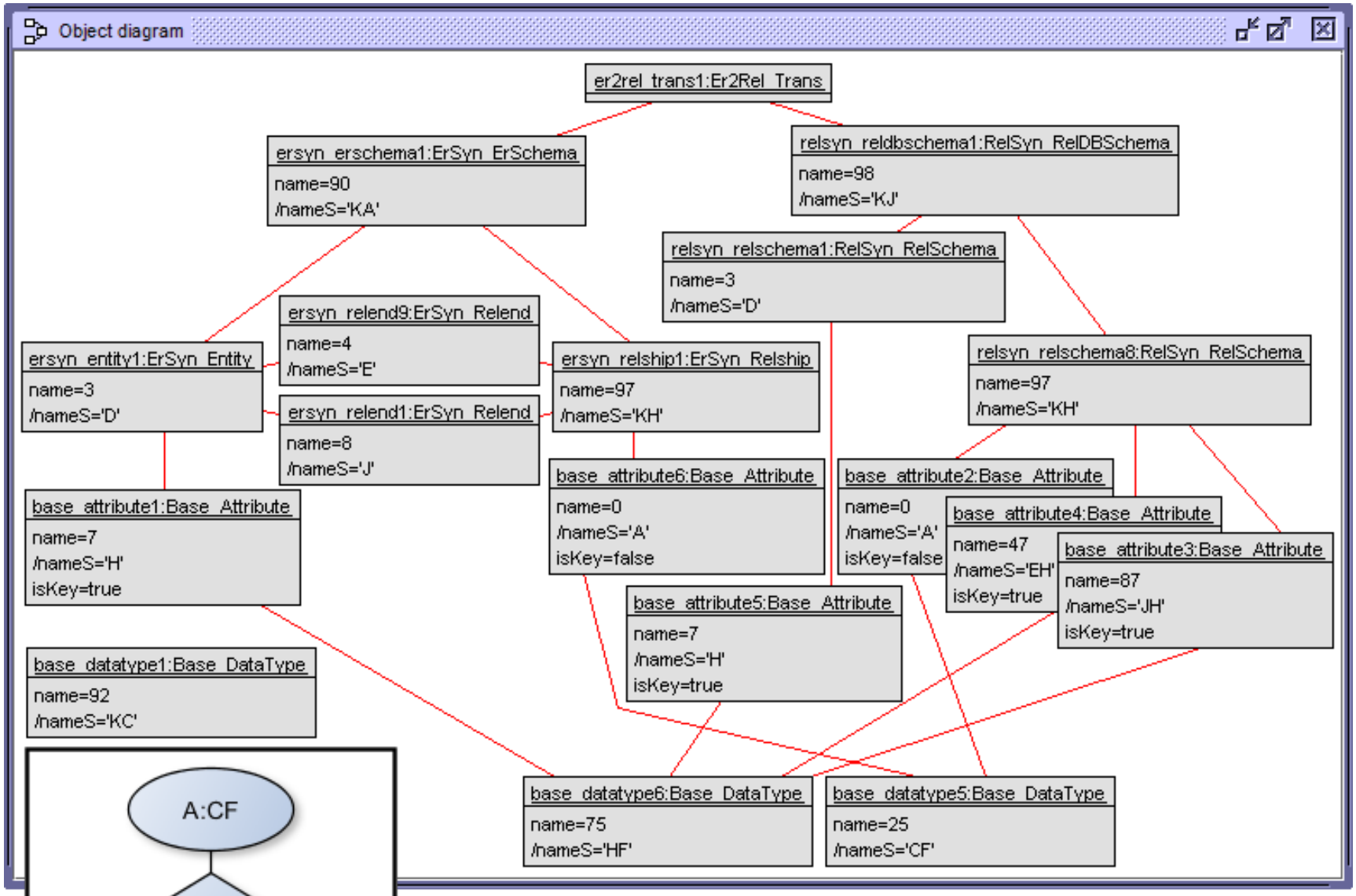
Link count

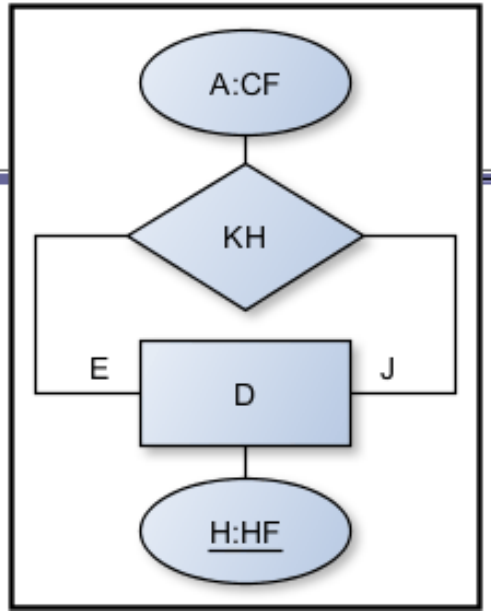
Association	# Links
Base_AttributeTyping	2
Er2Rel_OwnershipTransErSchema	1
Er2Rel_OwnershipTransRelDBSchema	1
ErSyn_OwnershipEntityAttribute	1
ErSyn_OwnershipErSchemaEntity	1
ErSyn_OwnershipErSchemaRelship	0
ErSyn_OwnershipRelshipAttribute	0
ErSyn_OwnershipRelshipRelend	0
ErSyn_RelendTyping	0
RelSyn_OwnershipRelDBSchemaRelSchema	1
RelSyn_OwnershipRelSchemaAttribute	1






```
create table JC(
  K KH primary key)
create table F(
  EK KH,
  JK KH,
  primary key(EK,JK))
```






```
create table D(
  H HF primary key)
create table KH(
  EH HF,
  JH HF,
  A CF,
  primary key(EH,JH))
```

Transformation preserves property:
  source has only key attributes THEN target has only key attributes

add invariant:

```
context self:Er2Rel_Trans inv erHasOnlyKeyAttrs_relHasSomeNonKeyAttrs:
  self.erSchema.entity.attribute->
    union(self.erSchema.relship.attribute)->
      select(a|a.isKey=false)->isEmpty() and
  self.relDBSchema.relSchema.attribute->
    select(a|a.isKey=false)->notEmpty()
```

using the 'weak consistency' configuration, i.e., the least restrictive configuration, the model validator reports 'UNSATISFIABLE'

```
unsatisfiable: erSchema.hasOnlyKeyAttrs() and
               not relDBSchema.hasOnlyKeyAttrs()

valid:         not ( erSchema.hasOnlyKeyAttrs() and
                     not relDBSchema.hasOnlyKeyAttrs() )

valid:         ( not erSchema.hasOnlyKeyAttrs() ) or
                     relDBSchema.hasOnlyKeyAttrs()

valid:         erSchema.hasOnlyKeyAttrs() implies
                     relDBSchema.hasOnlyKeyAttrs()
```

Translation and Solving Times

	Translation [ms]	Solving [ms]
ER schema with binary association	2.152	2.481
Weak consistency	187.002	2.355
Class instanciability	208.258	177.388
Class and association instanciability	184.345	320.705
Implied property (‘ER keys’ vs. ‘rel. keys’)	182.957	58.305
Larger example from [6]	12.714	374.650

Conclusion

- presented an approach for automatically checking transformation model properties
  - consistency
  - class instantiability
  - class and association instantiability
  - property preservation by the transformation model
- implementation of model validator based on relational model finder Kodkod

Future work

- study invariant independence, i.e., minimality of transformation models
- deliver to the model validator partial solution and let validator automatically complete the transformation
- feasible: checking for unique results (constructing further results beyond first found solutions)
- handling of strings to be improved
- larger case studies must check the practicability

Thanks for your attention!

```
-- For every RelSchema there is either an Entity or a Relship with the
-- same properties and name; if the RelSchema corresponds to an
-- Entity, both have Attributes with the same names, DataTypes, and
-- key properties; if the RelSchema corresponds to a Relship, the
-- RelSchema has Attributes corresponding to the arms of the Relship
-- and both have Attributes with the same properties

context self:Er2Rel_Trans inv forRelSchemaExistsOneEntityXorRelship:
  self.relDBSchema.relSchema->forAll(rl |
    self.erSchema.entity->one(e |
      rl.name=e.name and
      rl.attribute->forAll(ra |
        e.attribute->one(ea |
          ra.name=ea.name and ea.dataType=ra.dataType and
          ra.isKey=ea.isKey)))
    xor
    self.erSchema.relship->one(rs |
      rl.name=rs.name and
      rl.attribute->forAll(ra |
        rs.relend->one(re |
          re.entity.key()->one(rek |
            ra.name=re.name.concat('_').concat(rek.name) and
            ra.dataType=rek.dataType and ra.isKey))
        xor
        rs.attribute->one(rsa |
          ra.name=rsa.name and ra.dataType=rsa.dataType and
          ra.isKey=false))))
```