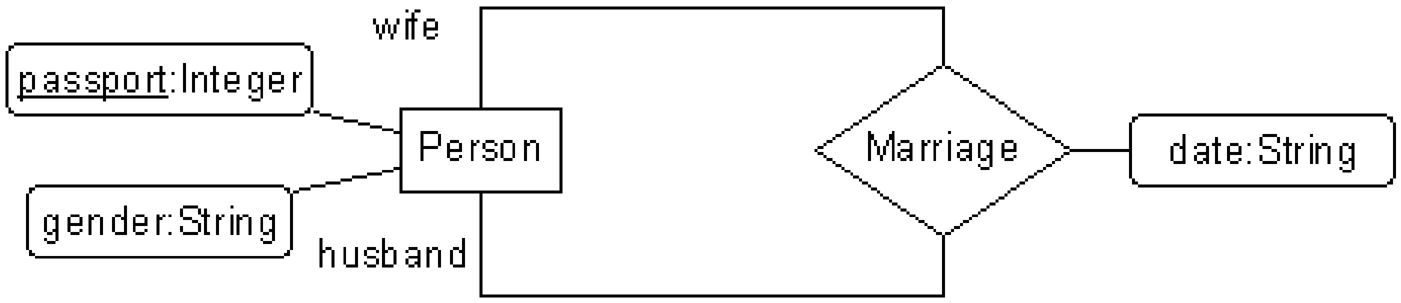
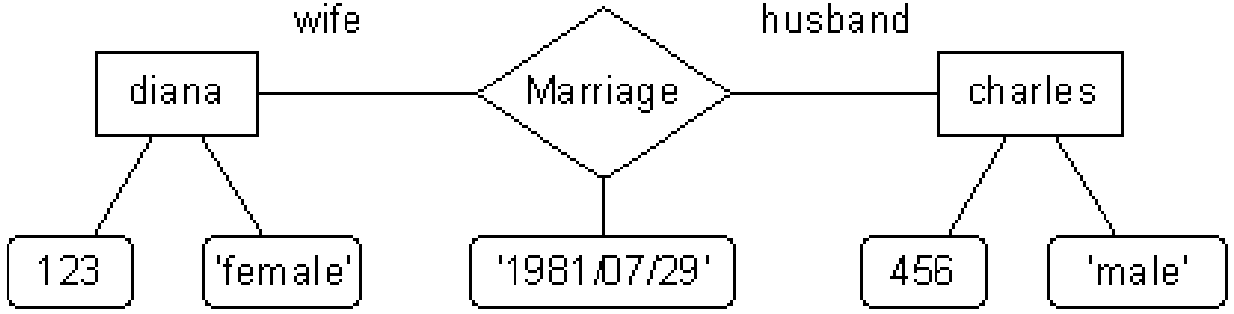
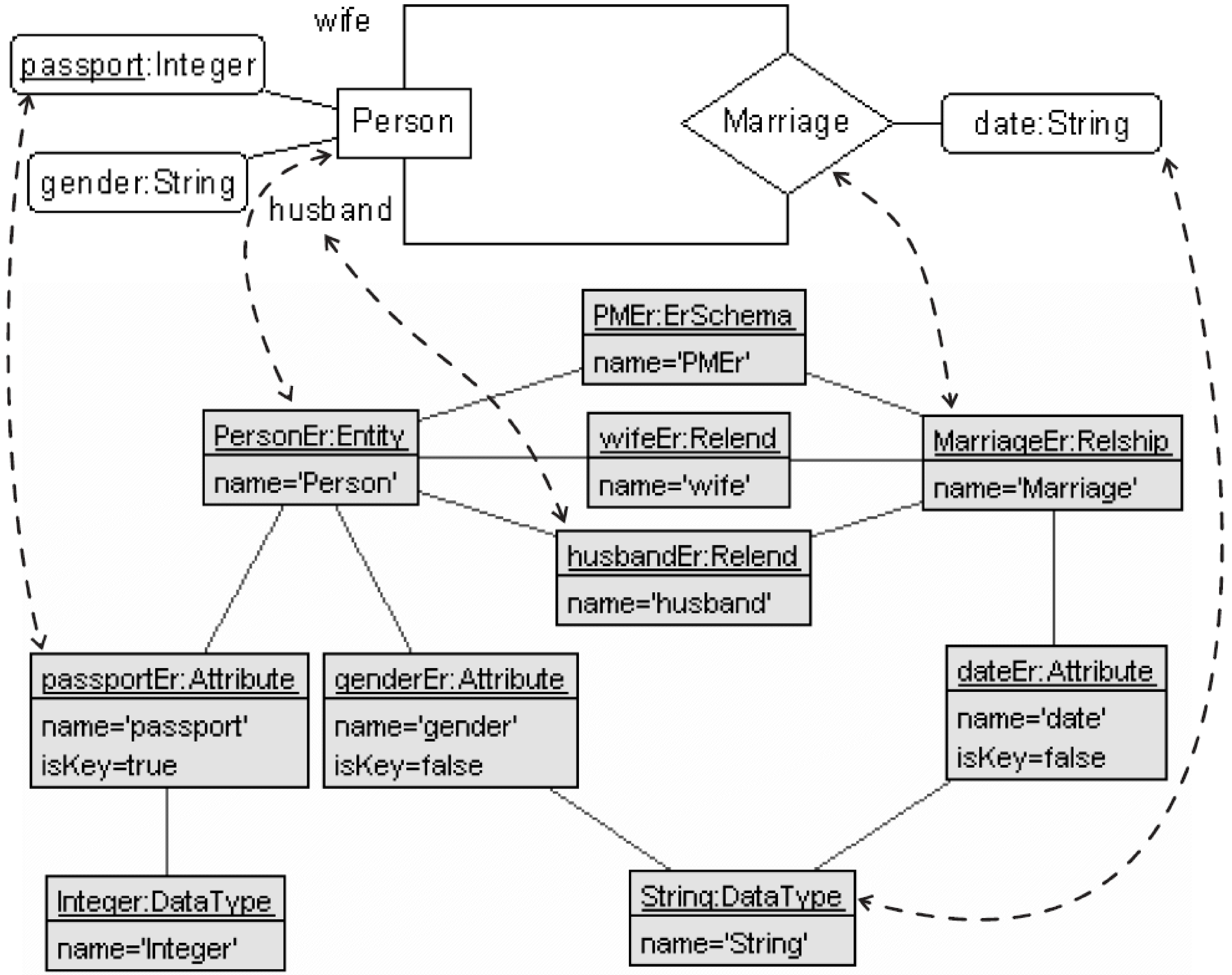
Metamodeling the Entity Relationship and Relational Data Model and their Transformation

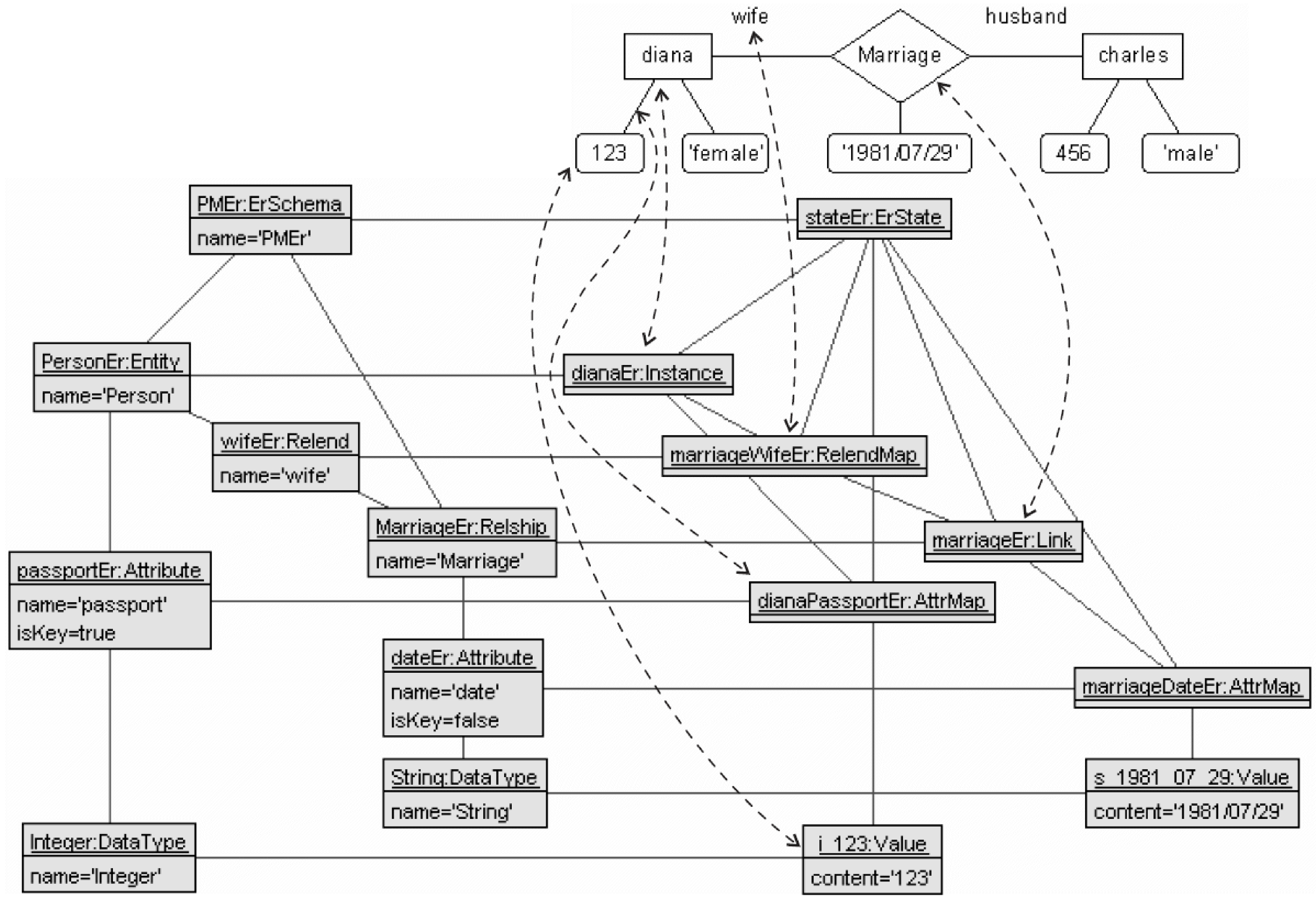


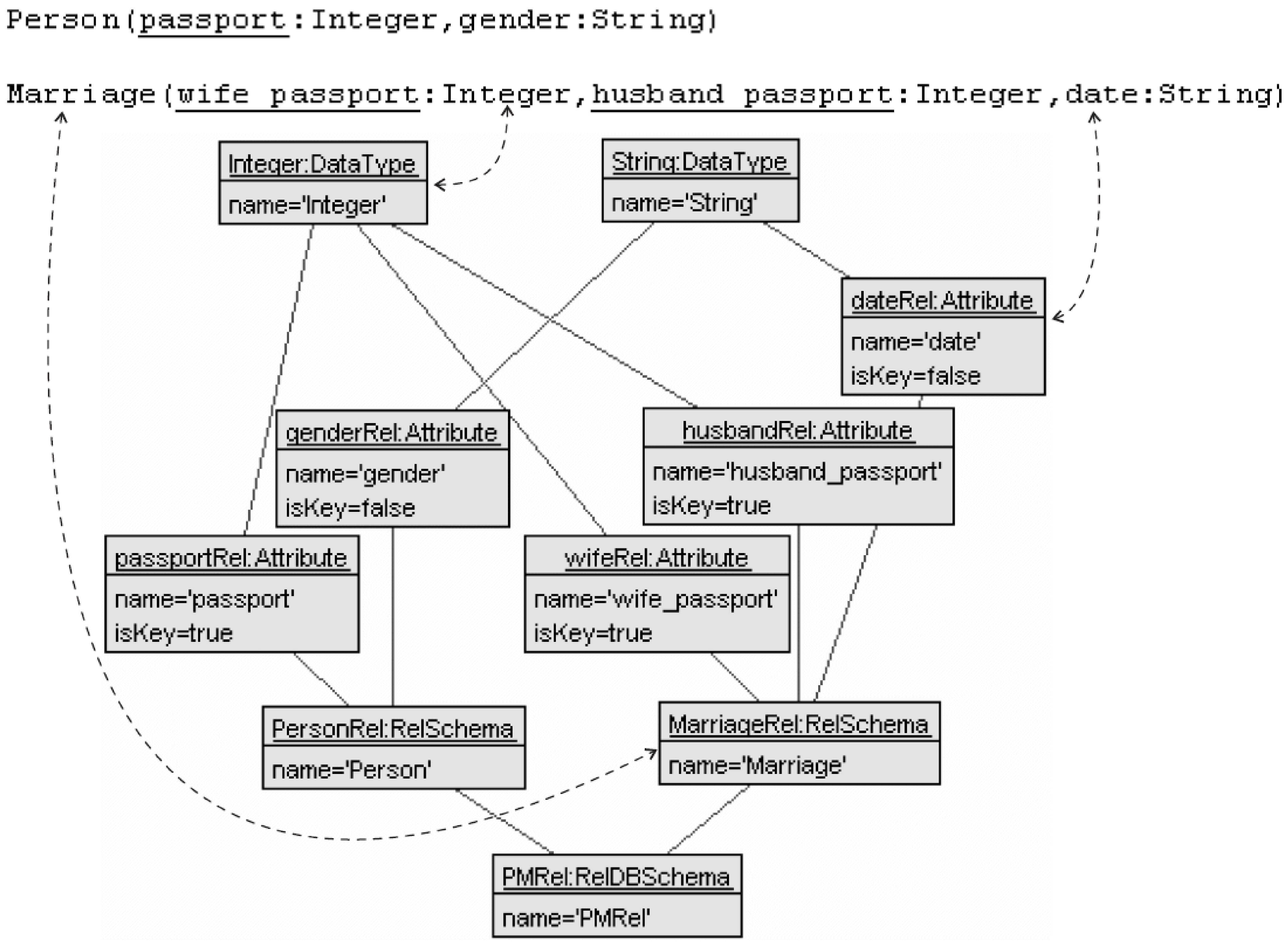


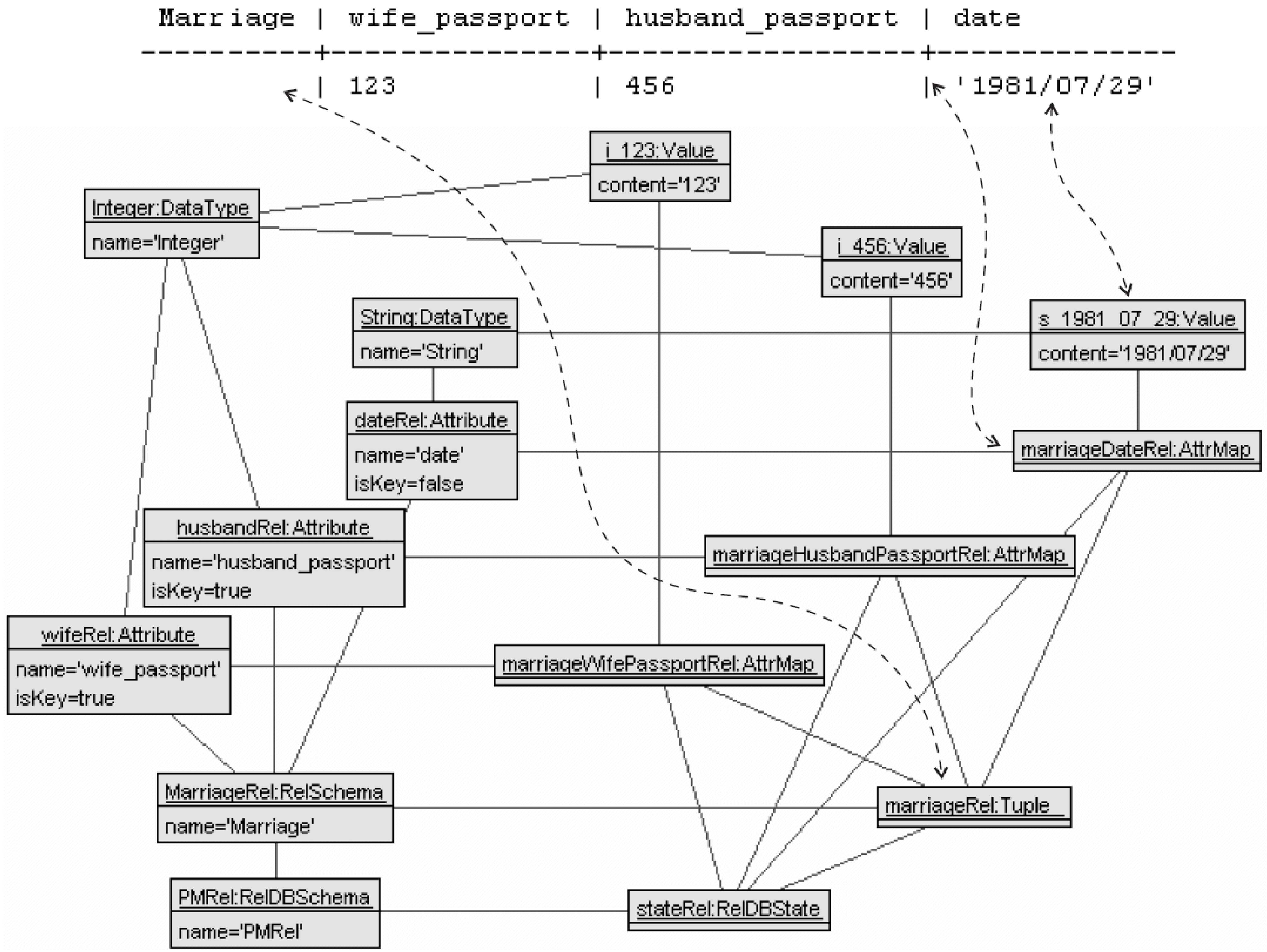
```
Person(passport:Integer,gender:String)

Marriage(wife_passport:Integer,husband_passport:Integer,date:String)
```

```
 Person | passport | gender
--------+----------+----------
        | 123      | 'female'
        | 456      | 'male'

 Marriage | wife_passport | husband_passport | date
----------+---------------+------------------+--------------
          | 123           | 456              | '1981/07/29'
```

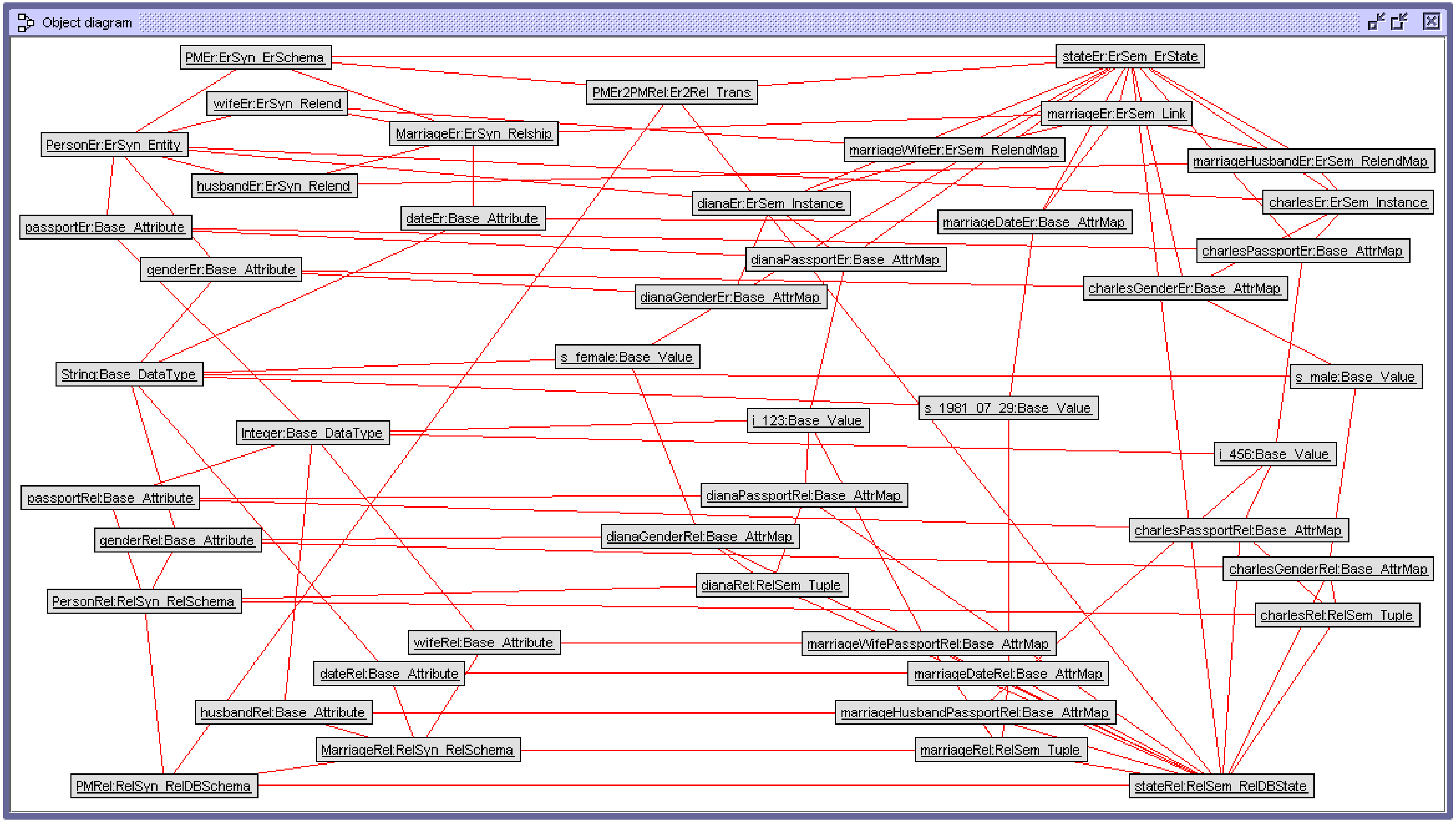

wife
passport:Integer
Person
Marriage
date:String
gender:String
husband
PMEr:ErSchema
name='PMEr'
PersonEr:Entity
name='Person'
wifeEr:Relend
name='wife'
MarriageEr:Relship
name='Marriage'
husbandEr:Relend
name='husband'
passportEr:Attribute
name='passport'
isKey=true
genderEr:Attribute
name='gender'
isKey=false
dateEr:Attribute
name='date'
isKey=false
Integer:DataType
name='Integer'
String:DataType
name='String'

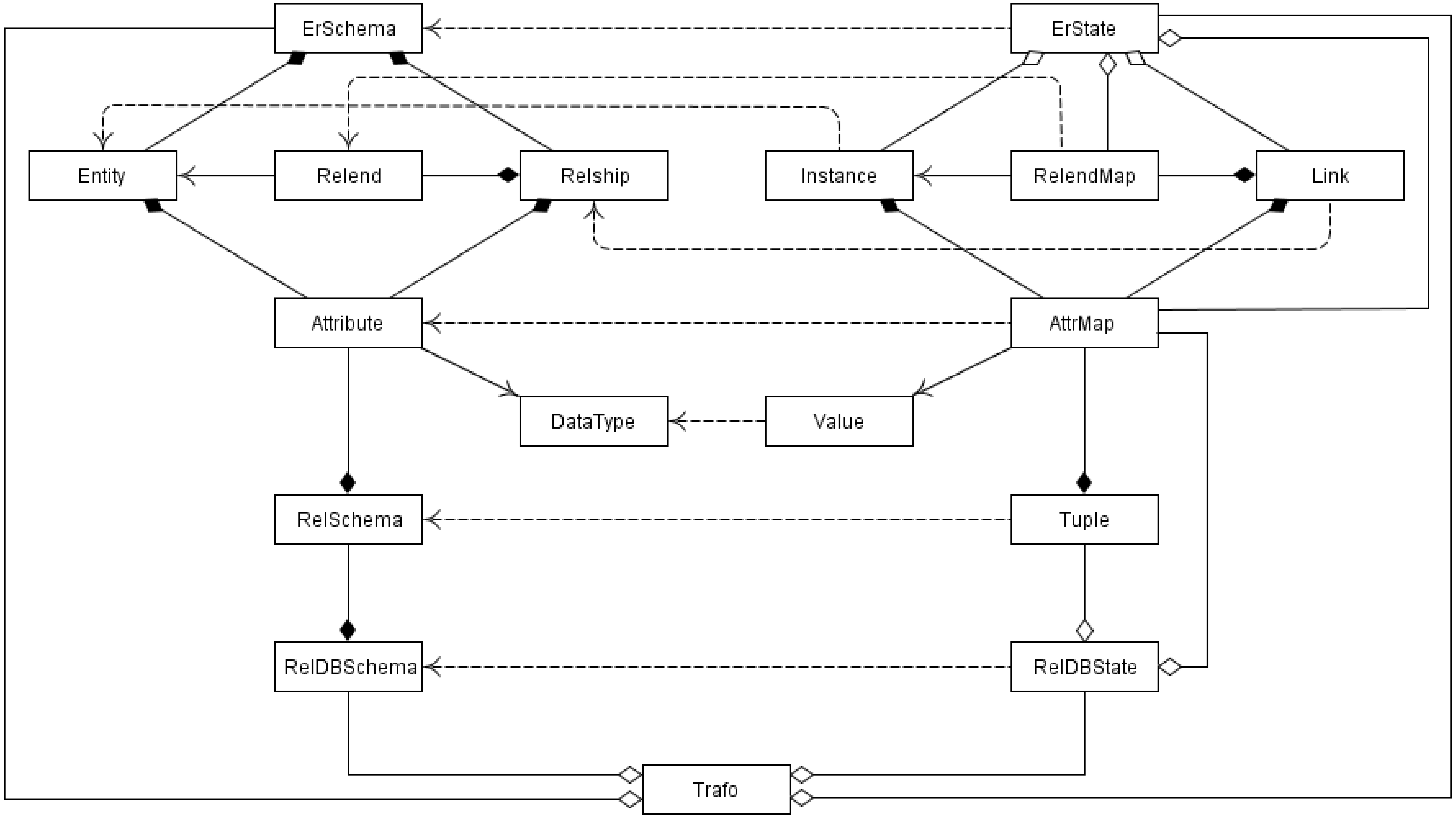

wife
husband
diana
Marriage
charles
123
'female'
'1981/07/29'
456
'male'
PMEr:ErSchema
name='PMEr'
stateEr:ErState
PersonEr:Entity
name='Person'
dianaEr:Instance
wifeEr:Relend
name='wife'
marriageWifeEr:RelendMap
MarriageEr:Relship
name='Marriage'
marriageEr:Link
passportEr:Attribute
name='passport'
isKey=true
dianaPassportEr:AttrMap
dateEr:Attribute
name='date'
isKey=false
marriageDateEr:AttrMap
String:DataType
name='String'
s_1981_07_29:Value
content='1981/07/29'
Integer:DataType
name='Integer'
i_123:Value
content='123'

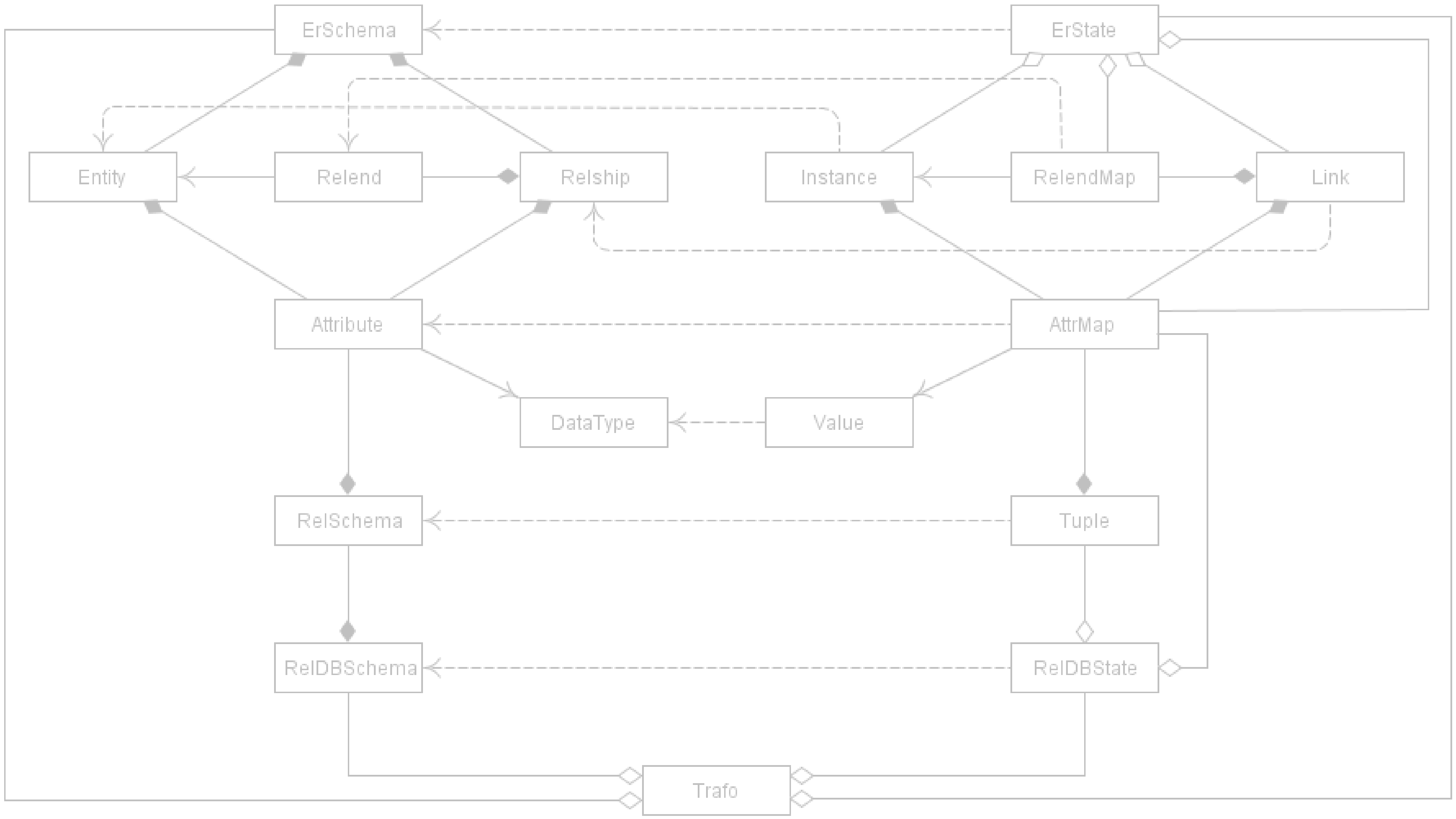

Person(passport:Integer,gender:String)
Marriage(wife_passport:Integer,husband_passport:Integer,date:String)
Integer:DataType
name='Integer'
String:DataType
name='String'
dateRel:Attribute
name='date'
isKey=false
genderRel:Attribute
name='gender'
isKey=false
husbandRel:Attribute
name='husband_passport'
isKey=true
passportRel:Attribute
name='passport'
isKey=true
wifeRel:Attribute
name='wife_passport'
isKey=true
PersonRel:RelSchema
name='Person'
MarriageRel:RelSchema
name='Marriage'
PMRel:RelDBSchema
name='PMRel'

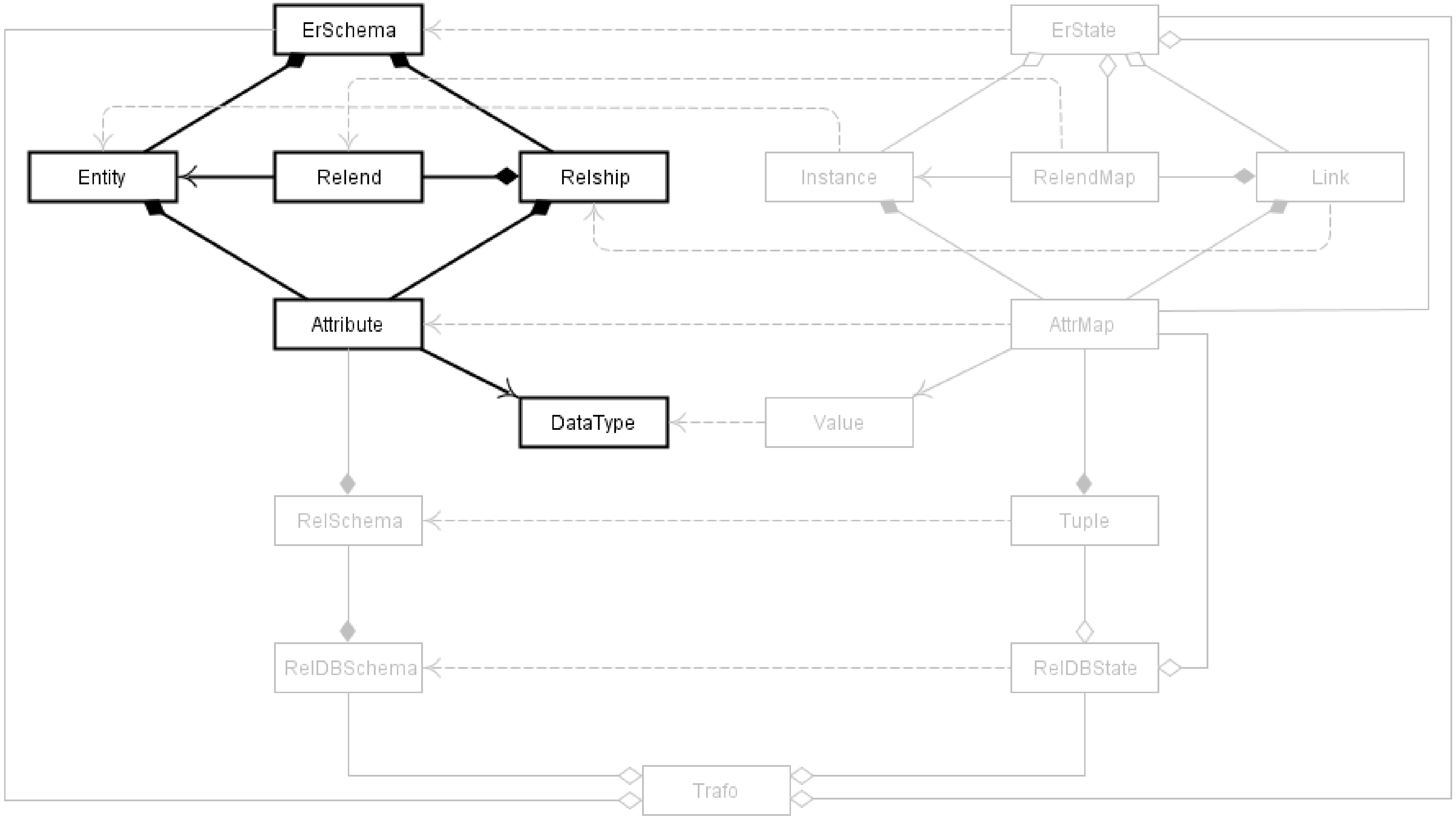


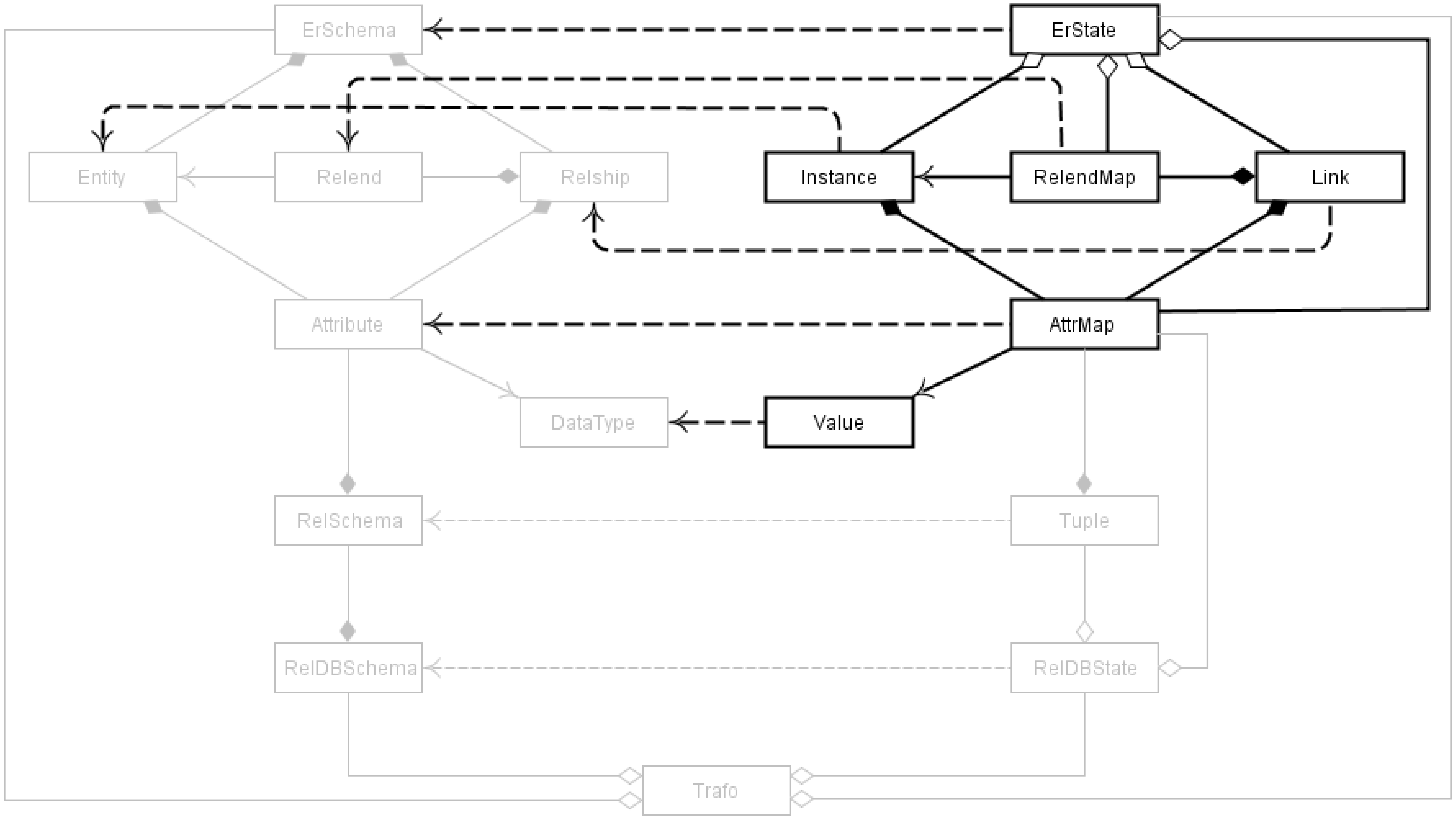
```
Marriage | wife_passport | husband_passport | date
---------+---------------+------------------+---------------
         | 123           | 456              | '1981/07/29'
```

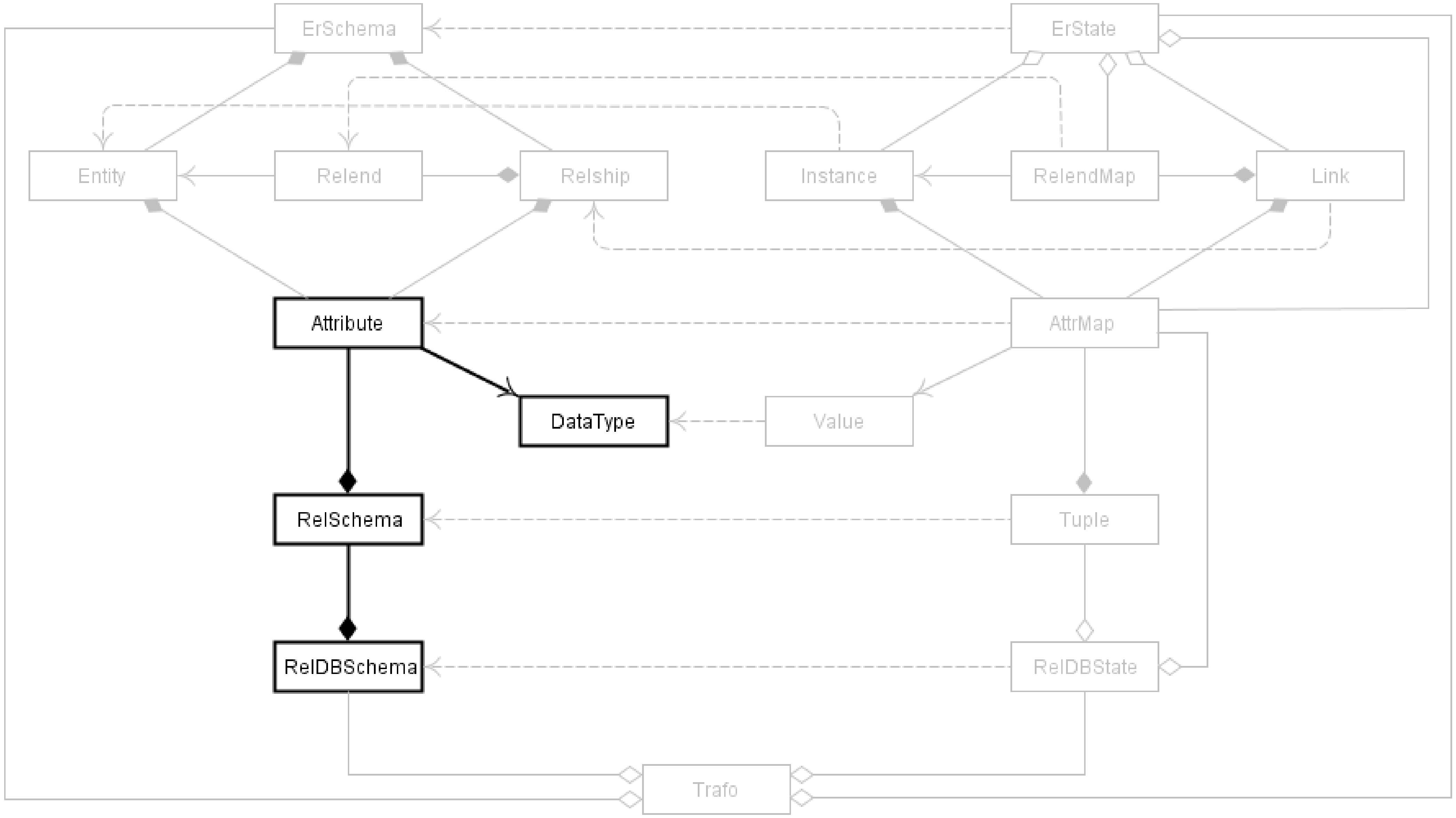

Object diagram
PMEr:ErSyn_ErSchema
stateEr:ErSem_ErState
PMEr2PMRel:Er2Rel_Trans
wifeEr:ErSyn_Relend
marriageEr:ErSem_Link
MarriageEr:ErSyn_Relship
PersonEr:ErSyn_Entity
marriageWifeEr:ErSem_RelendMap
marriageHusbandEr:ErSem_RelendMap
husbandEr:ErSyn_Relend
dianaEr:ErSem_Instance
charlesEr:ErSem_Instance
passportEr:Base_Attribute
dateEr:Base_Attribute
marriageDateEr:Base_AttrMap
charlesPassportEr:Base_AttrMap
dianaPassportEr:Base_AttrMap
genderEr:Base_Attribute
charlesGenderEr:Base_AttrMap
dianaGenderEr:Base_AttrMap
s_female:Base_Value
String:Base_DataType
s_male:Base_Value
s_1981_07_29:Base_Value
i_123:Base_Value
Integer:Base_DataType
i_456:Base_Value
passportRel:Base_Attribute
dianaPassportRel:Base_AttrMap
charlesPassportRel:Base_AttrMap
genderRel:Base_Attribute
dianaGenderRel:Base_AttrMap
charlesGenderRel:Base_AttrMap
dianaRel:RelSem_Tuple
PersonRel:RelSyn_RelSchema
charlesRel:RelSem_Tuple
wifeRel:Base_Attribute
marriageWifePassportRel:Base_AttrMap
dateRel:Base_Attribute
marriageDateRel:Base_AttrMap
husbandRel:Base_Attribute
marriageHusbandPassportRel:Base_AttrMap
MarriageRel:RelSyn_RelSchema
marriageRel:RelSem_Tuple
PMRel:RelSyn_RelDBSchema
stateRel:RelSem_RelDBState

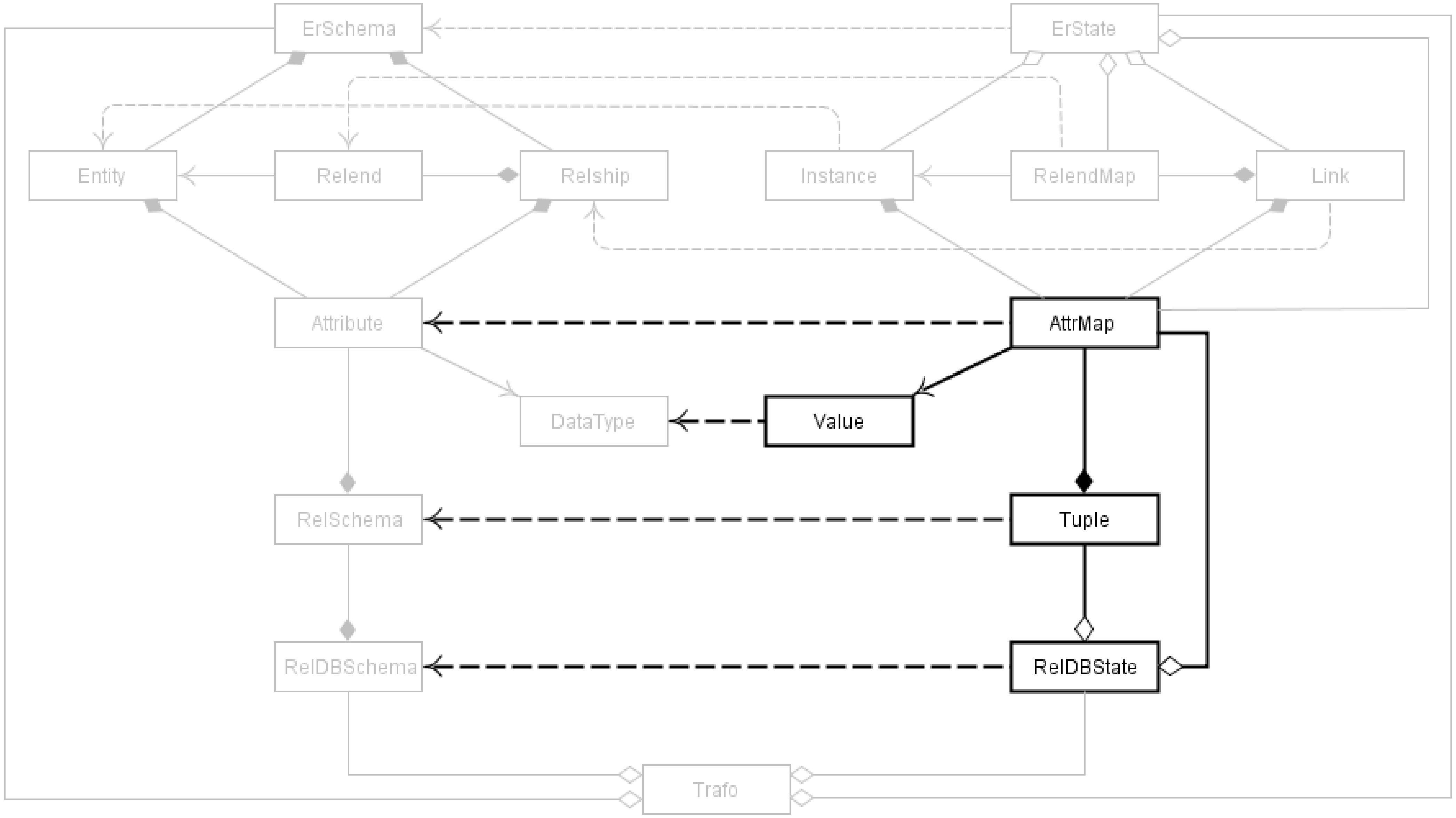

ErSchema
ErState
Entity
Relend
Relship
Instance
RelendMap
Link
Attribute
AttrMap
DataType
Value
RelSchema
Tuple
RelDBSchema
RelDBState
Trafo

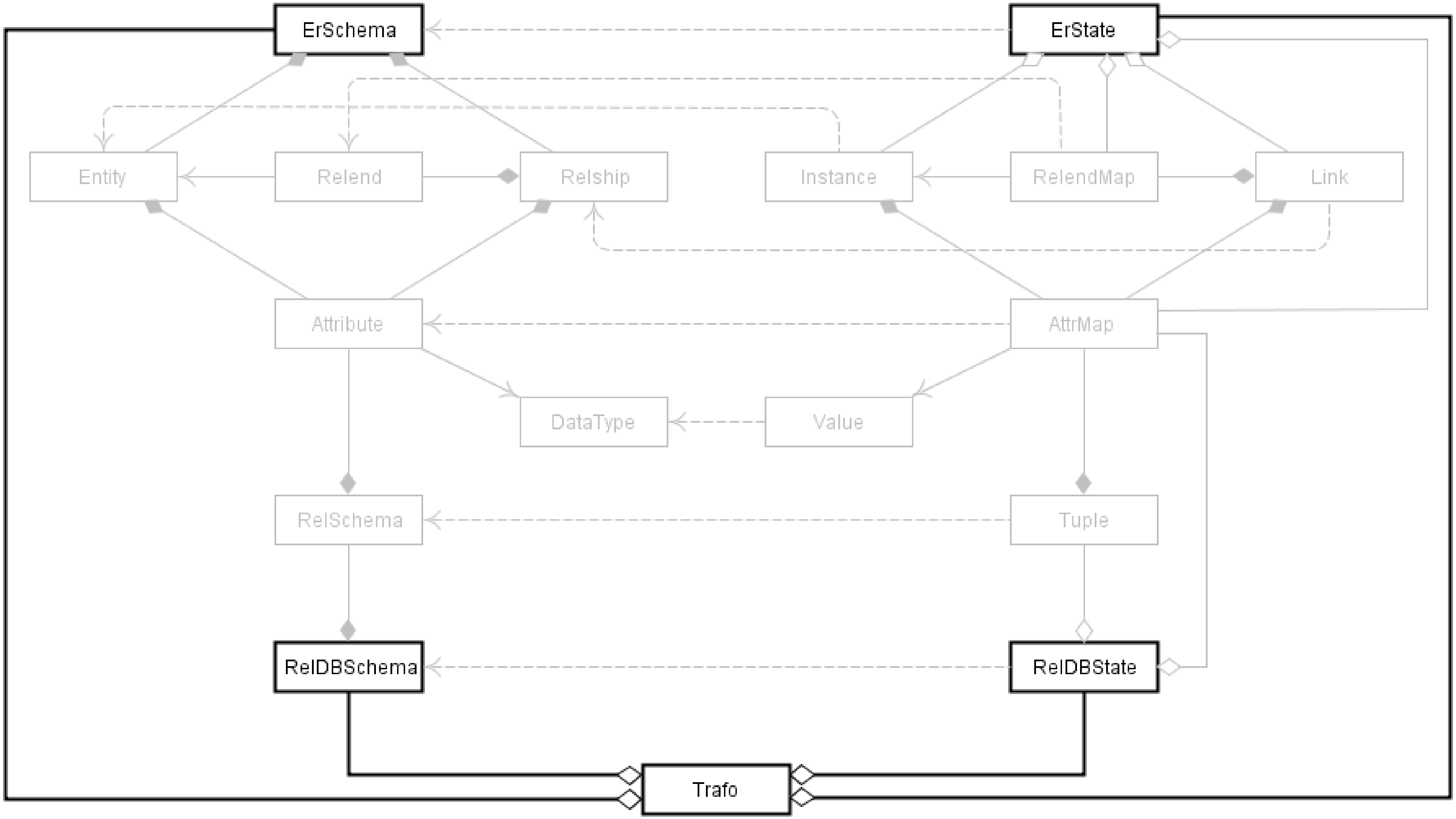

ErSchema
ErState
Entity
Relend
Relship
Instance
RelendMap
Link
Attribute
AttrMap
DataType
Value
RelSchema
Tuple
RelDBSchema
RelDBState
Trafo


ErSchema
ErState
Entity
Relend
Relship
Instance
RelendMap
Link
Attribute
AttrMap
DataType
Value
RelSchema
Tuple
RelDBSchema
RelDBState
Trafo


ErSchema
ErState
Entity
Relend
Relship
Instance
RelendMap
Link
Attribute
AttrMap
DataType
Value
RelSchema
Tuple
RelDBSchema
RelDBState
Trafo


ErSchema
ErState
Entity
Relend
Relship
Instance
RelendMap
Link
Attribute
AttrMap
DataType
Value
RelSchema
Tuple
RelDBSchema
RelDBState
Trafo


ErSchema
ErState
Entity
Relend
Relship
Instance
RelendMap
Link
Attribute
AttrMap
DataType
Value
RelSchema
Tuple
RelDBSchema
RelDBState
Trafo


ErSchema
ErState
Entity
Relend
Relship
Instance
RelendMap
Link
Attribute
AttrMap
DataType
Value
RelSchema
Tuple
RelDBSchema
RelDBState
Trafo

```
BASE

-- Naming restriction: Different DataTypes have different names

context self:Base_DataType inv uniqueDataTypeNames:
  Base_DataType.allInstances->
    forAll(self2 | self.name=self2.name implies self=self2)
```

ER SYNTAX

```
-- Different ErSchemas have different names

context self:ErSyn_ErSchema inv uniqueErSchemaNames:
  ErSyn_ErSchema.allInstances->
    forAll(self2 | self.name=self2.name implies self=self2)

-- Within one ErSchema, different Entities have different names

context self:ErSyn_ErSchema inv uniqueEntityNamesWithinErSchema:
  self.entity->forAll(e1,e2 | e1.name=e2.name implies e1=e2)
```

ER SEMANTICS

```
-- Two different Instances of one Entity can be distinguished in every
-- ErState where both Instances occur by a key Attribute of the Entity

context self:ErSem_Instance inv keyMapUnique:
  ErSem_Instance.allInstances->forAll(self2 |
    self<>self2 and self.entity=self2.entity
    implies
    self.erState->intersection(self2.erState)->forAll(s |
      self.entity.key()->exists(ka |
        self.applyAttr(s,ka)<>self2.applyAttr(s,ka))))
```

```
REL SYNTAX

-- The set of key Attributes of a RelSchema is not empty

context self:RelSyn_RelSchema inv relSchemaKeyNotEmpty:
  self.key()->notEmpty
```

REL SEMANTICS

```
-- Two different Tuples of one RelSchema can be distinguished in every
-- RelDBState where both Tuples occur by a key Attribute of the
-- RelSchema

context self:RelSem_Tuple inv keyMapUnique:
  RelSem_Tuple.allInstances->forAll(self2 |
    self<>self2 and self.relSchema=self2.relSchema
    implies
    self.relDBState->intersection(self2.relDBState)->forAll(s |
      self.relSchema.key()->exists(ka |
        self.applyAttr(s,ka)<>self2.applyAttr(s,ka))))
```

```
TRANSFORMATION

-- For every Relship in the ErSchema there is a RelSchema having the
-- same name, Relends representing the arms of the relationship, and
-- Attributes with the same properties, i.e., name, DataType, and key
-- property

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