Business Informatics Group

Vienna University of Technology

Common Pitfalls of Using QVT Relations – Graphical Debugging as Remedy

Angelika Kusel, Wieland Schwinger, Manuel Wimmer, and Werner Retschitzegger

4th IEEE International Workshop UML and AADL @ ICECCS 2009

Manuel Wimmer

Business Informatics Group

Institute of Software Technology and Interactive Systems
Vienna University of Technology

Favoritenstraße 9-11/188-3, 1040 Vienna, Austria
phone: +43 (1) 58801-18804 (secretary), fax: +43 (1) 58801-18896
office@big.tuwien.ac.at, www.big.tuwien.ac.at

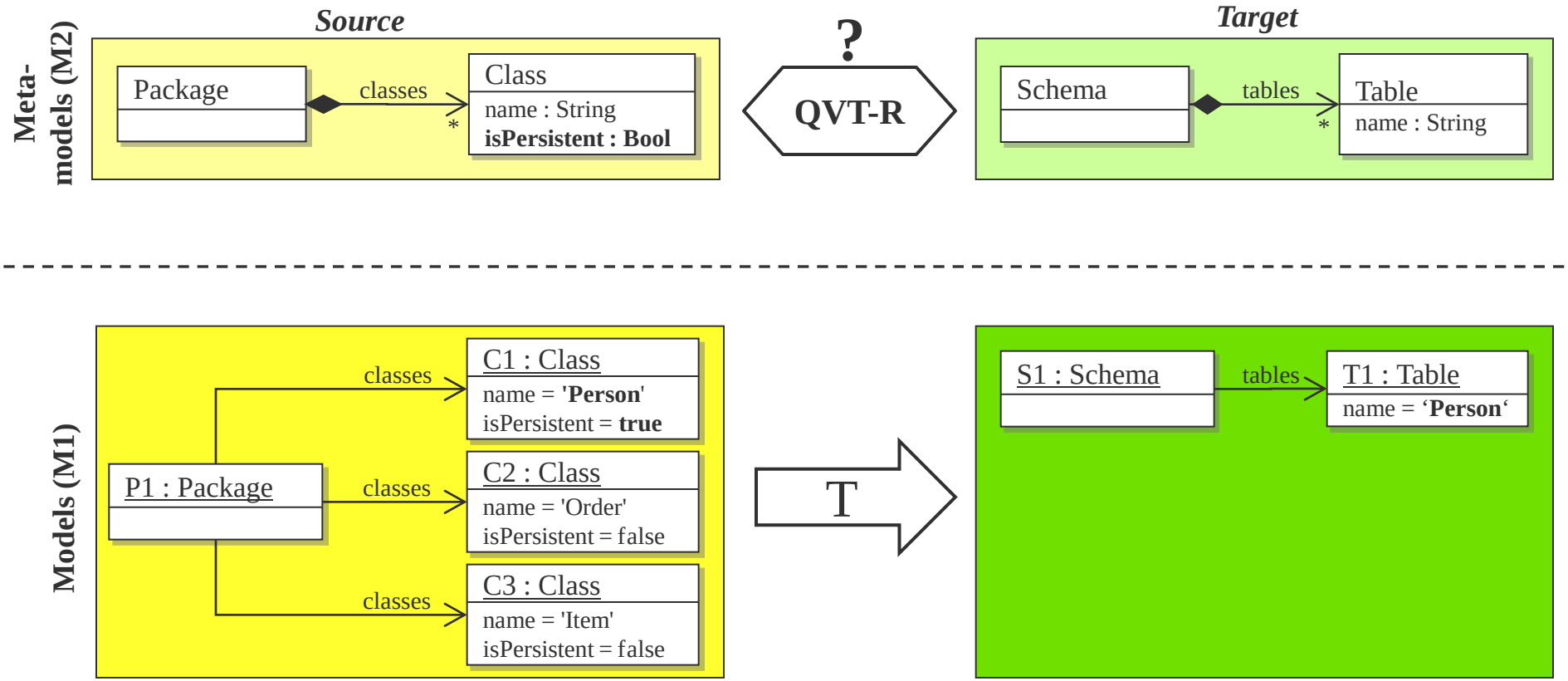
This work has been partly funded by the Austrian Science Fund (FWF) under grant P21374-N13. FWF

Introduction

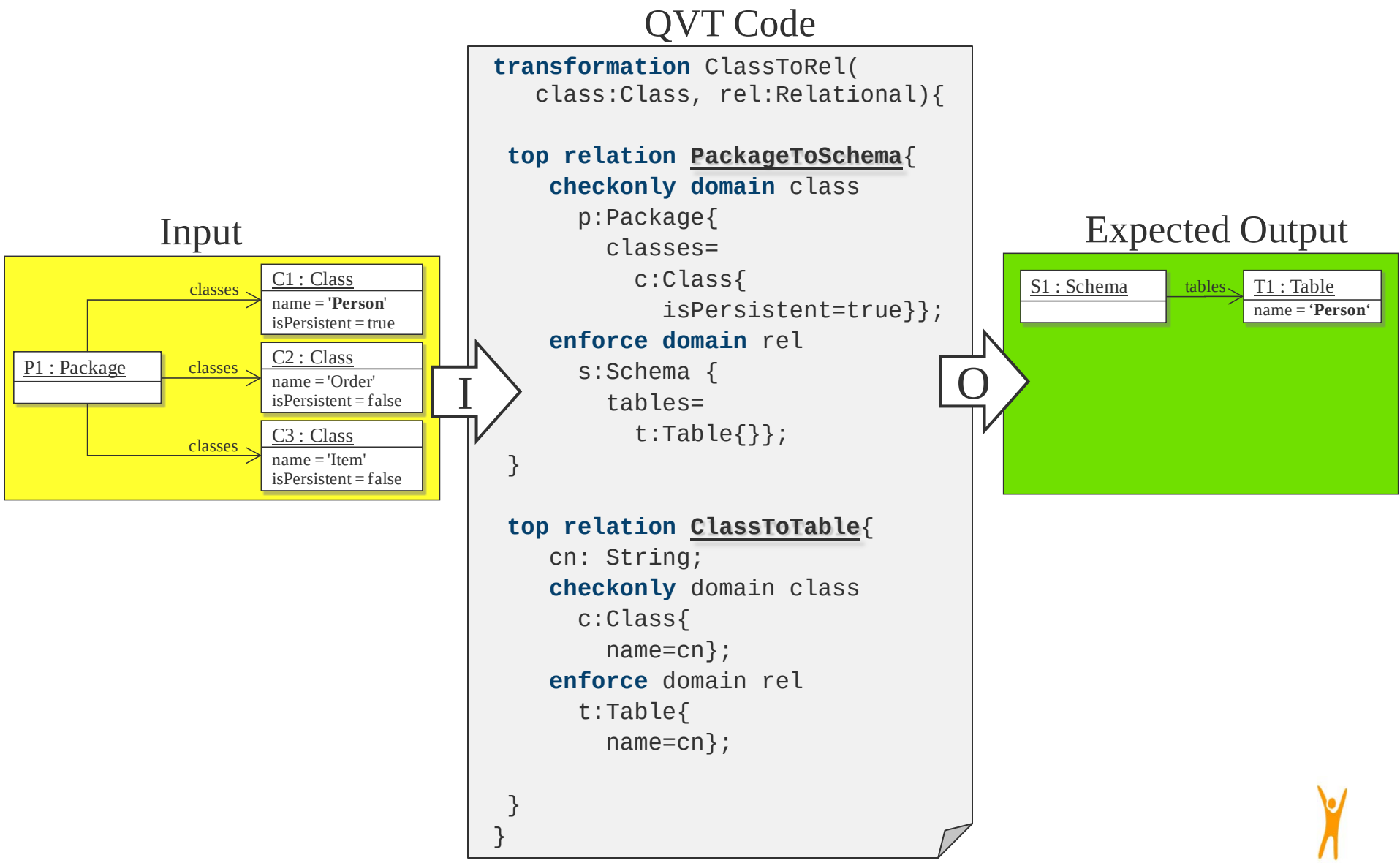
- Model Transformations are the key technology for model-driven engineering
- Various transformation approaches have been proposed
- OMG Standard: Query/View/Transformations (QVT)
  - High-level declarative: *Relations*
  - Low-level declarative: *Core*
  - Imperative: Operational *Mappings*
- QVT has not been adopted in practice
  - Especially the QVT Relations language
- Reasons
  - Tool support is still in it‘s infancy
  - Pragmatics is poorly understood

Running Example

- Class 2 Relational excerpt



UML 2 Relations: First Solution



UML 2 Relations: First Solution

QVT Code

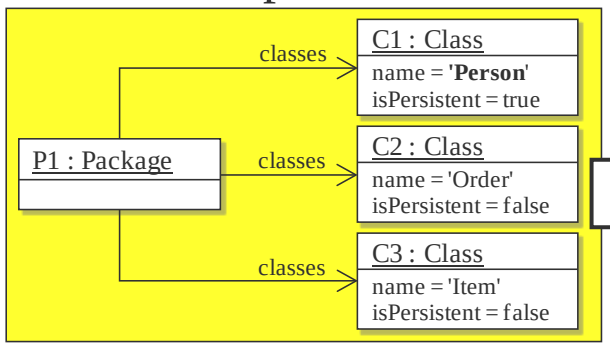
```
transformation ClassToRel(
   class:Class, rel:Relational){

 top relation PackageToSchema{
    checkonly domain class
      p:Package{
        classes=
          c:Class{
            isPersistent=true}};
    enforce domain rel
      s:Schema {
        tables=
          t:Table{}};
 }

 top relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn};
    enforce domain rel
      t:Table{
        name=cn};

 }
}
```

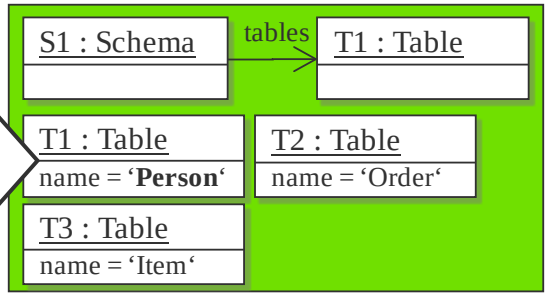
Input



I

O

Actual Output

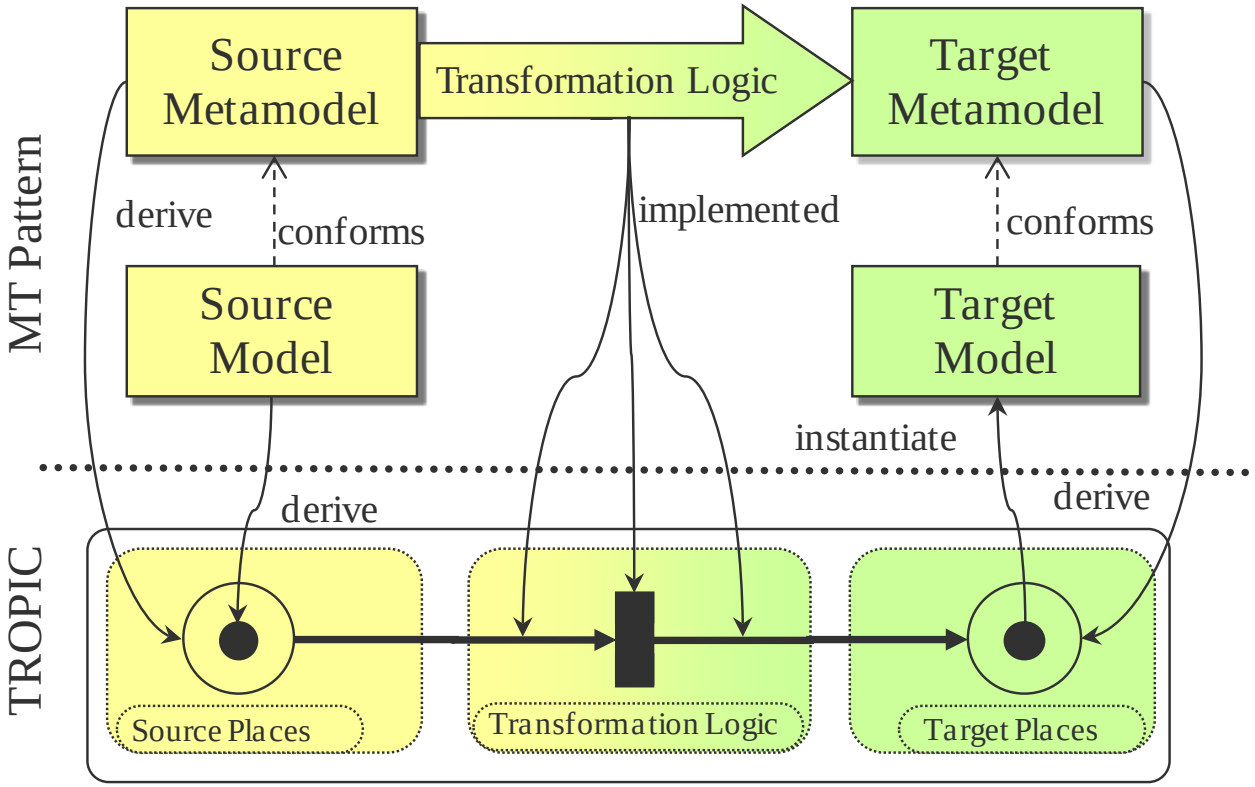


Dedicated Debugging View

TRansformations On Petri nets In Color

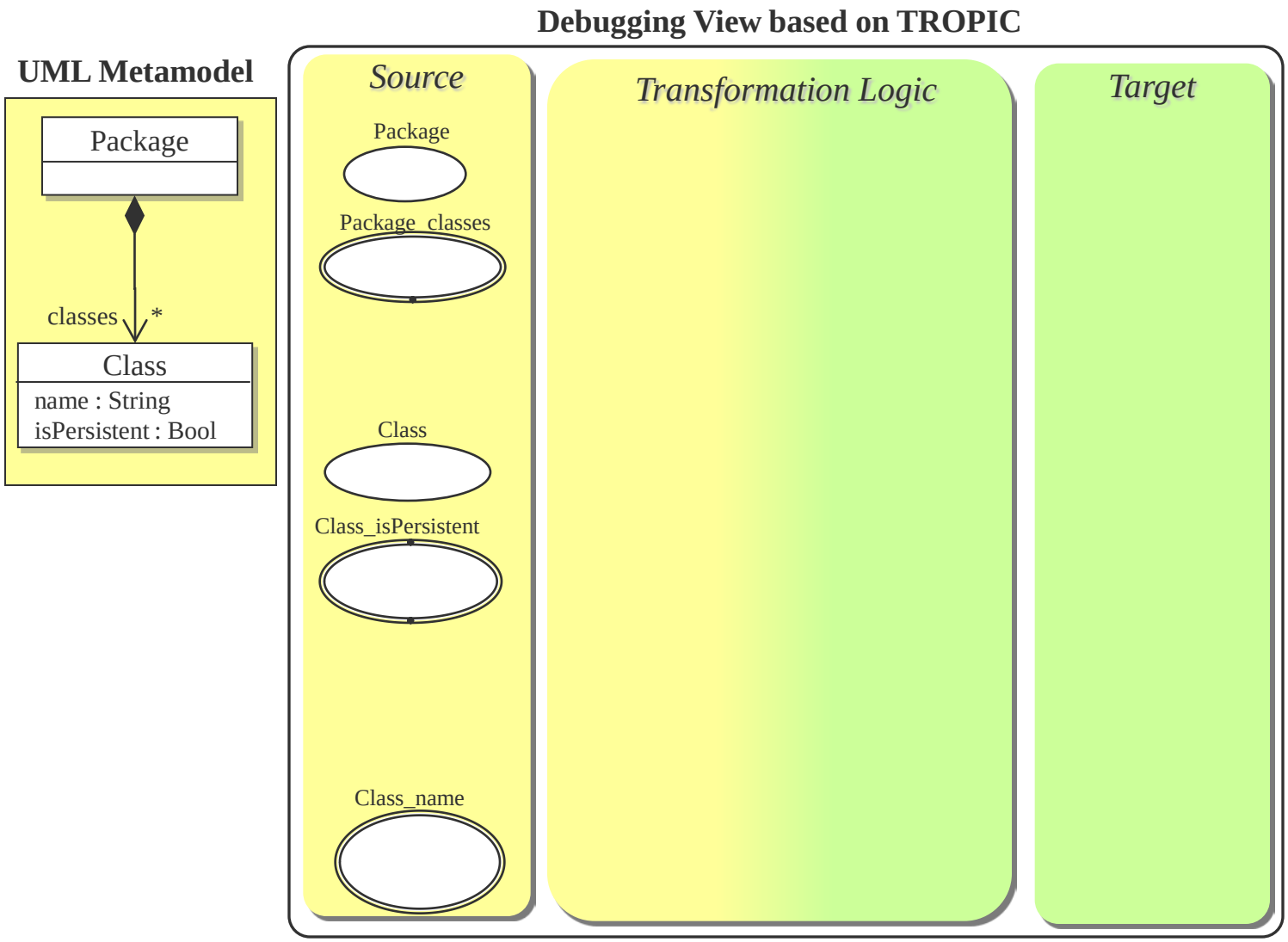
www.modeltransforamtion.net

- Debugging View based on TROPIC
  - TRansformations on Petri nets In Color
- Architecture



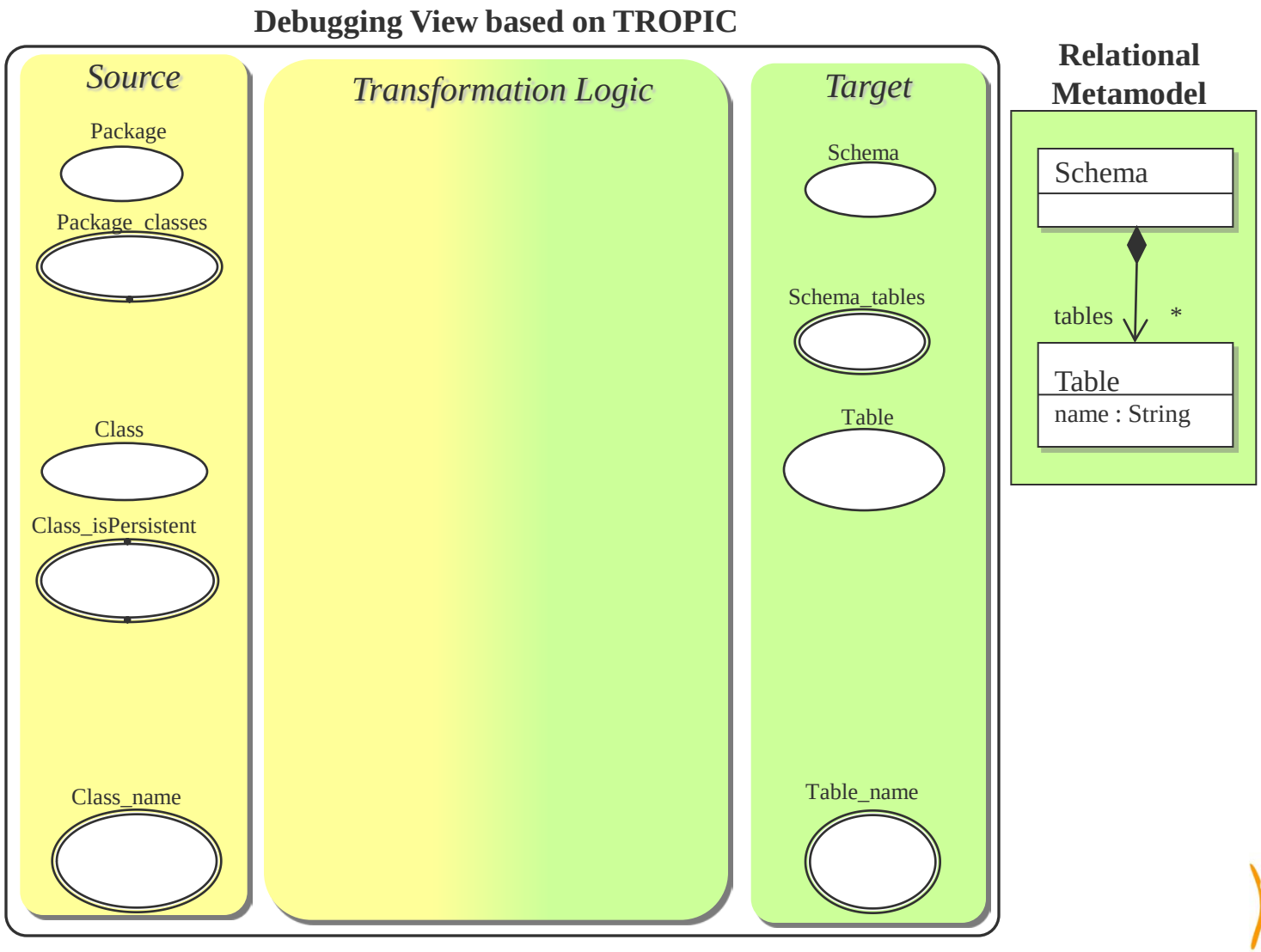
Starting the Debugger 1/3

Setting up Source and Target Places



Starting the Debugger 1/3

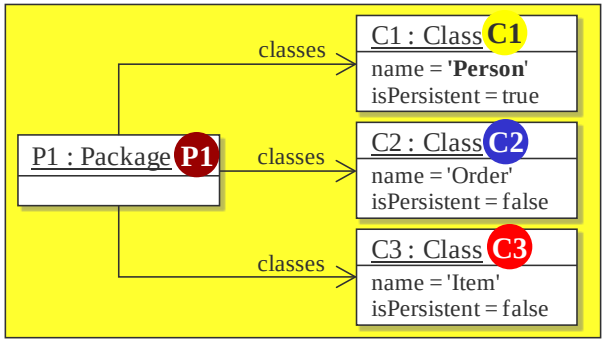
Setting up Source and Target Places



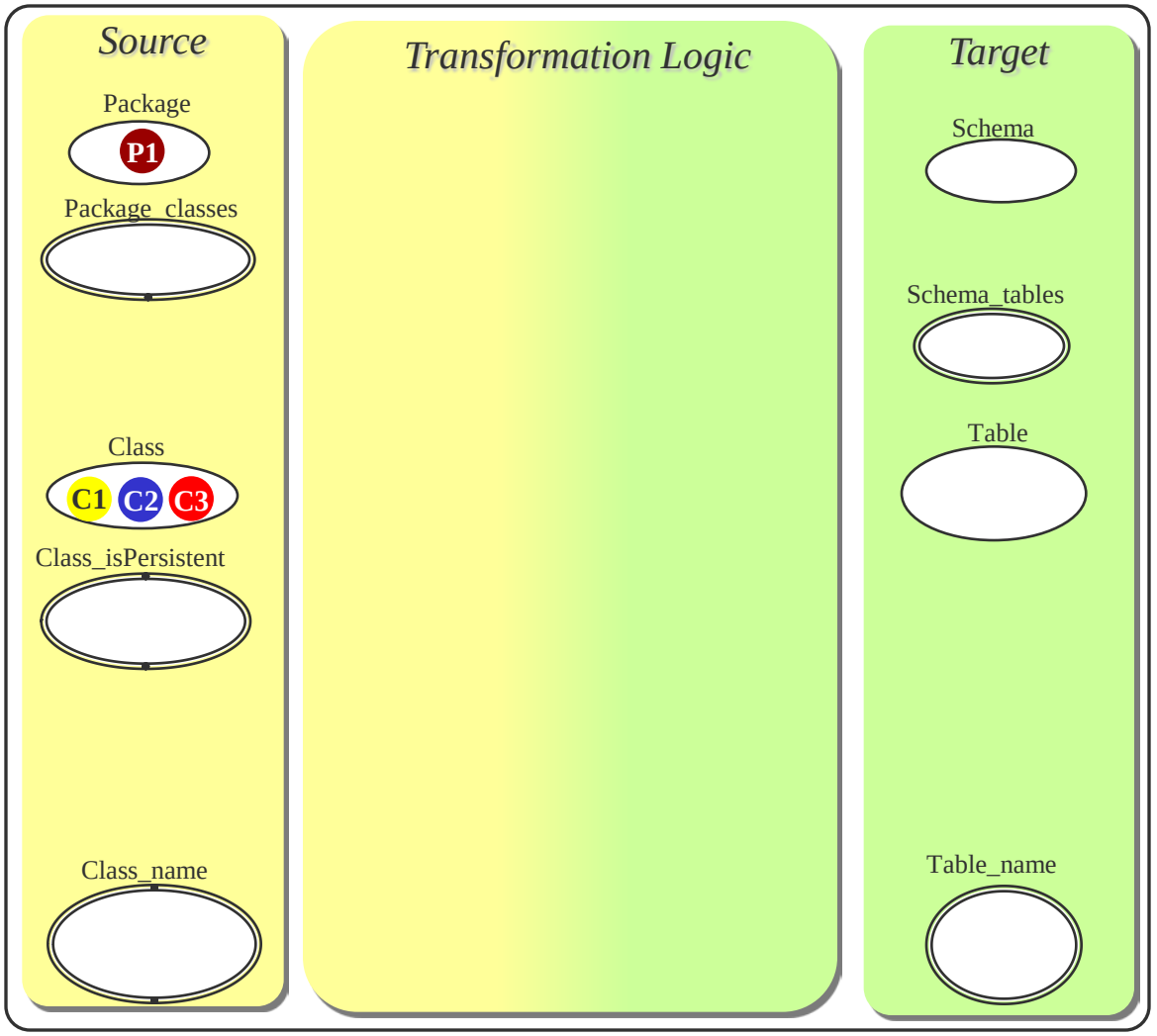
Starting the Debugger 2/3

Import Source Tokens

UML Model



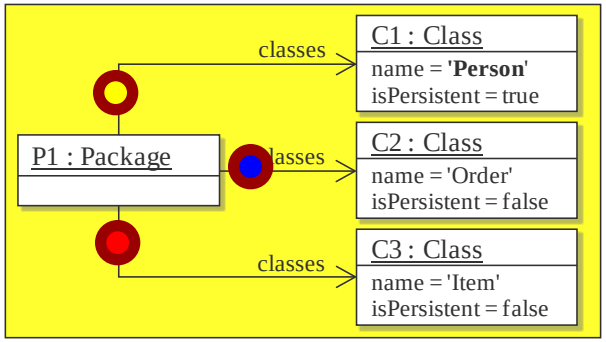
Debugging View based on TROPIC



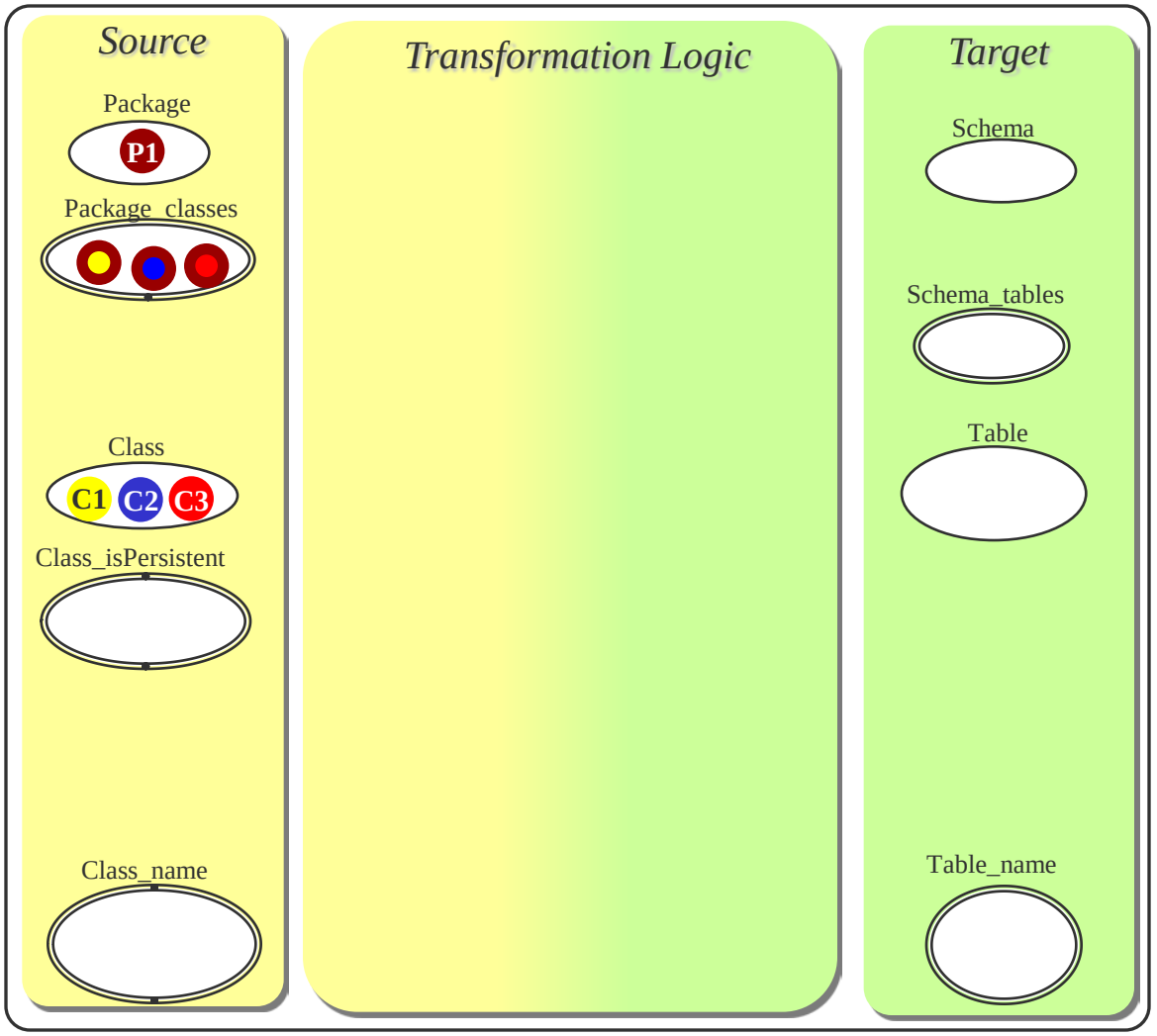
Starting the Debugger 2/3

Import Source Tokens

UML Model

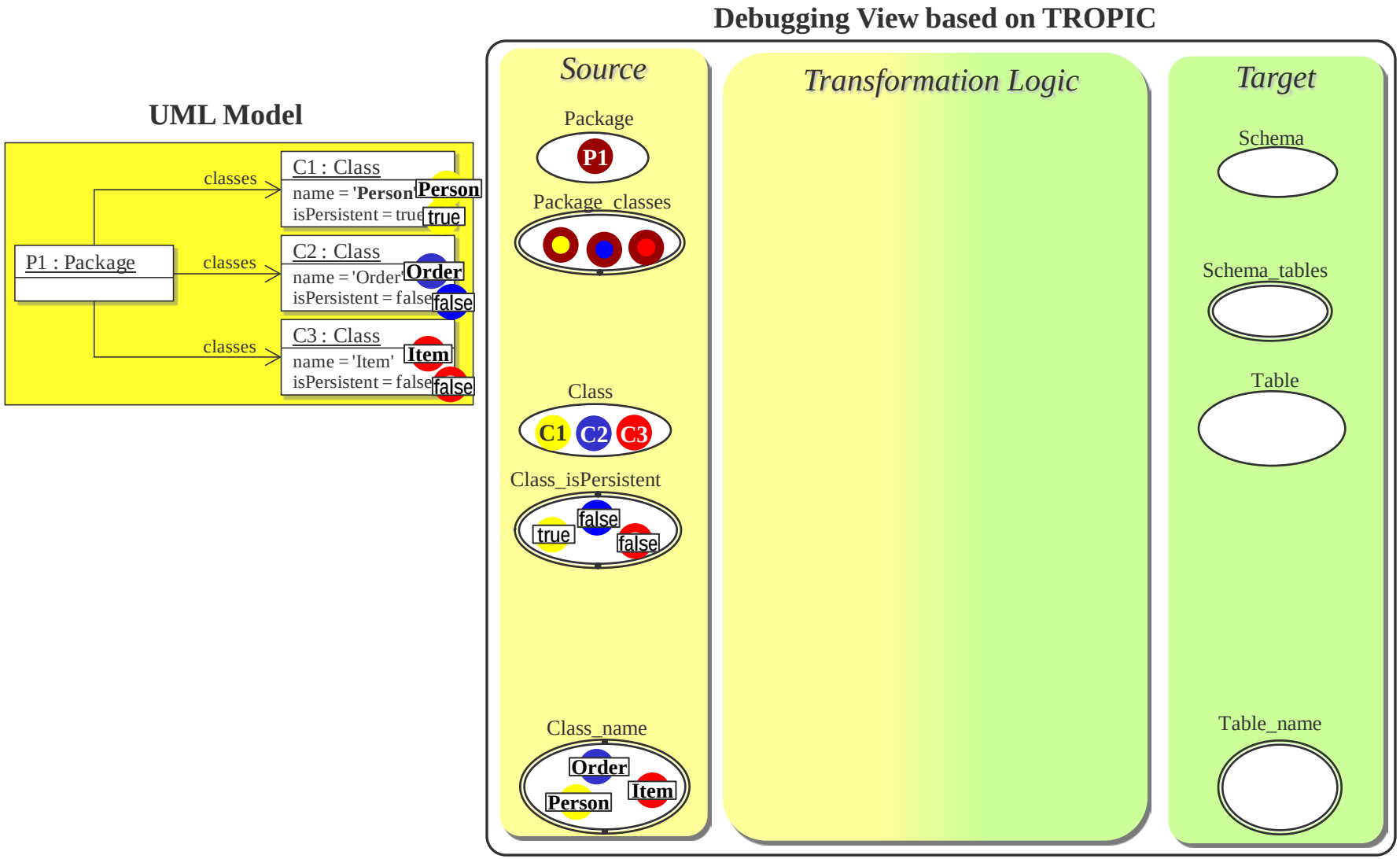


Debugging View based on TROPIC



Starting the Debugger 2/3

Import Source Tokens



Starting the Debugger 3/3

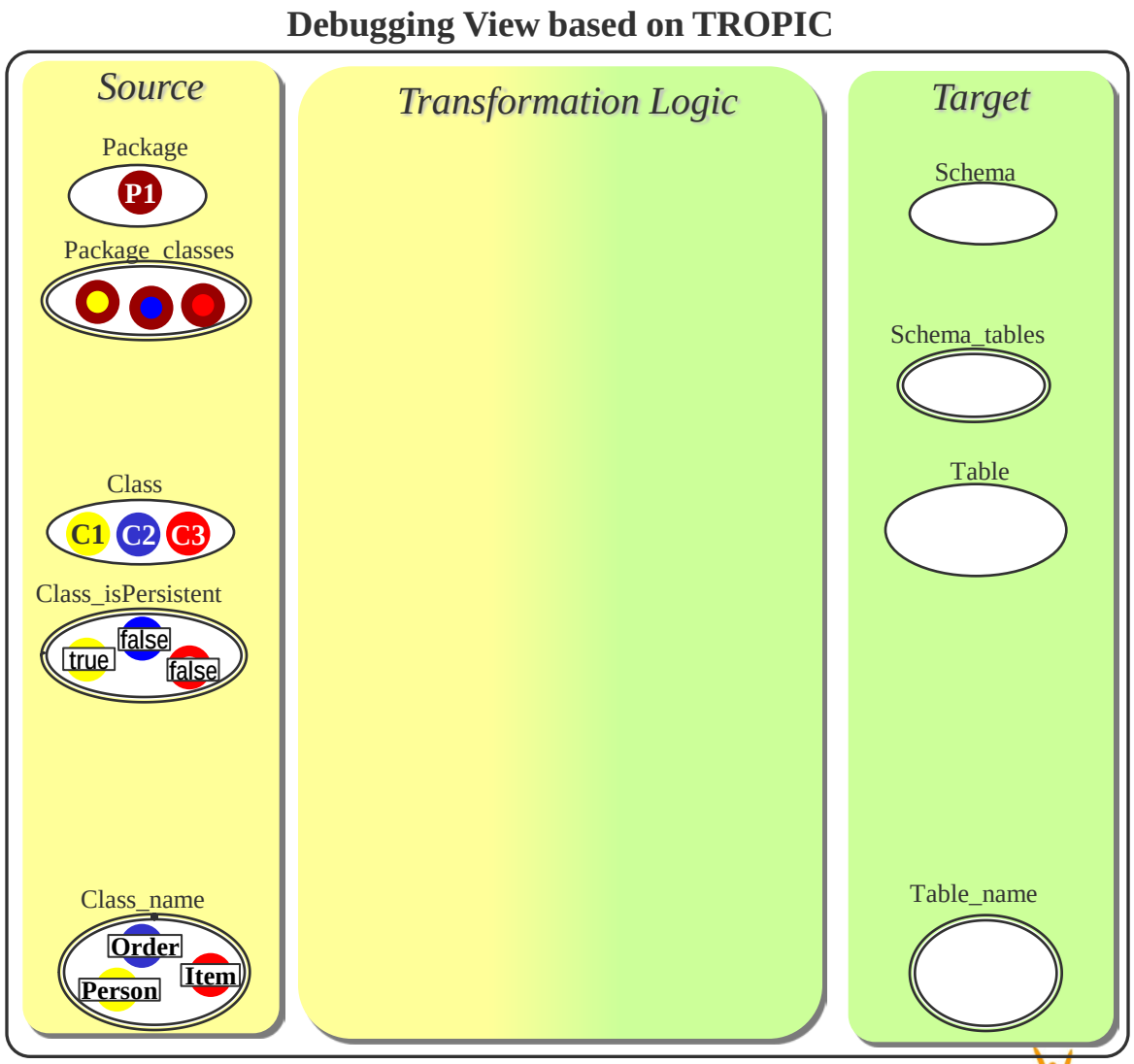
Represent Transformation Logic

```
transformation ClassToRel(
   class:Class, rel:Relational){

 top relation PackageToSchema{
    checkonly domain class
      p:Package{
        classes=
          c:Class{
            isPersistent=true}};
    enforce domain rel
      s:Schema {
        tables=
          t:Table{}};
 }

 top relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn};
    enforce domain rel
      t:Table{
        name=cn};

 }
}
```



QVT Code

```
top relation PackageToSchema{
    checkonly domain class
      p:Package{
        classes=
          c:Class{
            isPersistent=true}};
    enforce domain rel
      s:Schema {
        tables=
          t:Table{}};
}
```

```
top relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn};
    enforce domain rel
      t:Table{
        name=cn};

}
```

Debugging View based on TROPIC

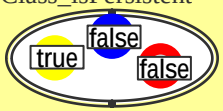
Source

Package

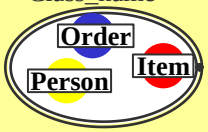
Package_classes

Class

Class_isPersistent



Class_name



PackageToSchema

ClassToTable

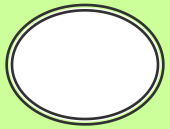
Target

Schema

Schema_tables

Table

Table_name



QVT Code

```
top relation PackageToSchema{
    checkonly domain class
      p:Package{
        classes=
          c:Class{
            isPersistent=true}};
    enforce domain rel
      s:Schema {
        tables=
          t:Table{}};
}
```

```
top relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn};
    enforce domain rel
      t:Table{
        name=cn};

}
```

Debugging View based on TROPIC

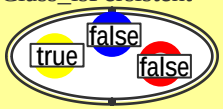
Source

Package

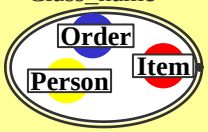
Package_classes

Class

Class_isPersistent



Class_name



domain object mapping

PackageToSchema

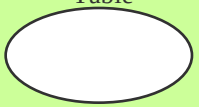
ClassToTable

Target

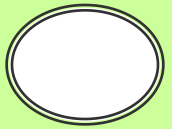
Schema

Schema_tables

Table



Table_name

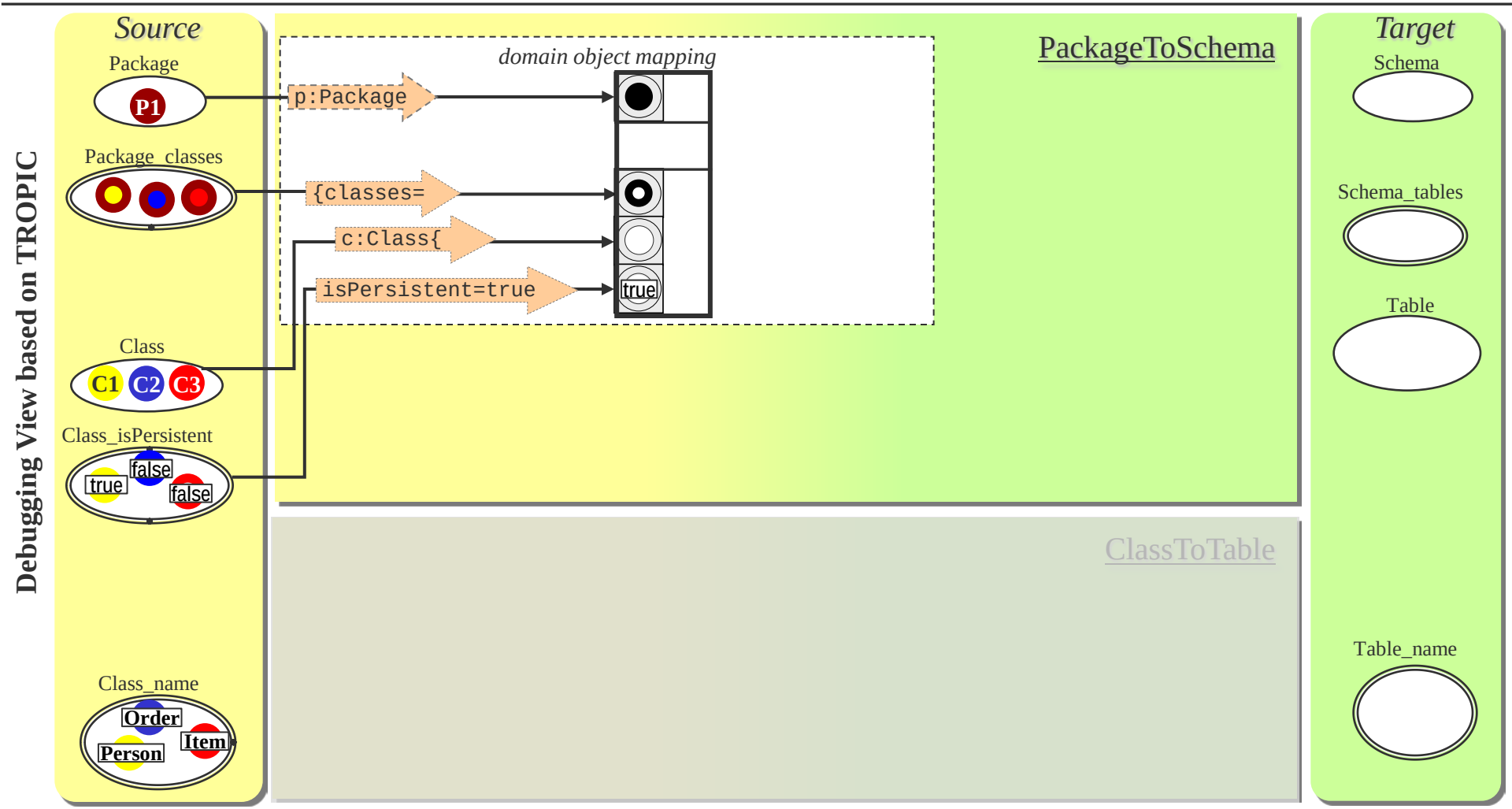


QVT Code

```
top relation PackageToSchema{
    checkonly domain class
      p:Package{
        classes=
          c:Class{
            isPersistent=true}};
    enforce domain rel
      s:Schema {
        tables=
          t:Table{}};
}
```

```
top relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn};
    enforce domain rel
      t:Table{
        name=cn};

}
```

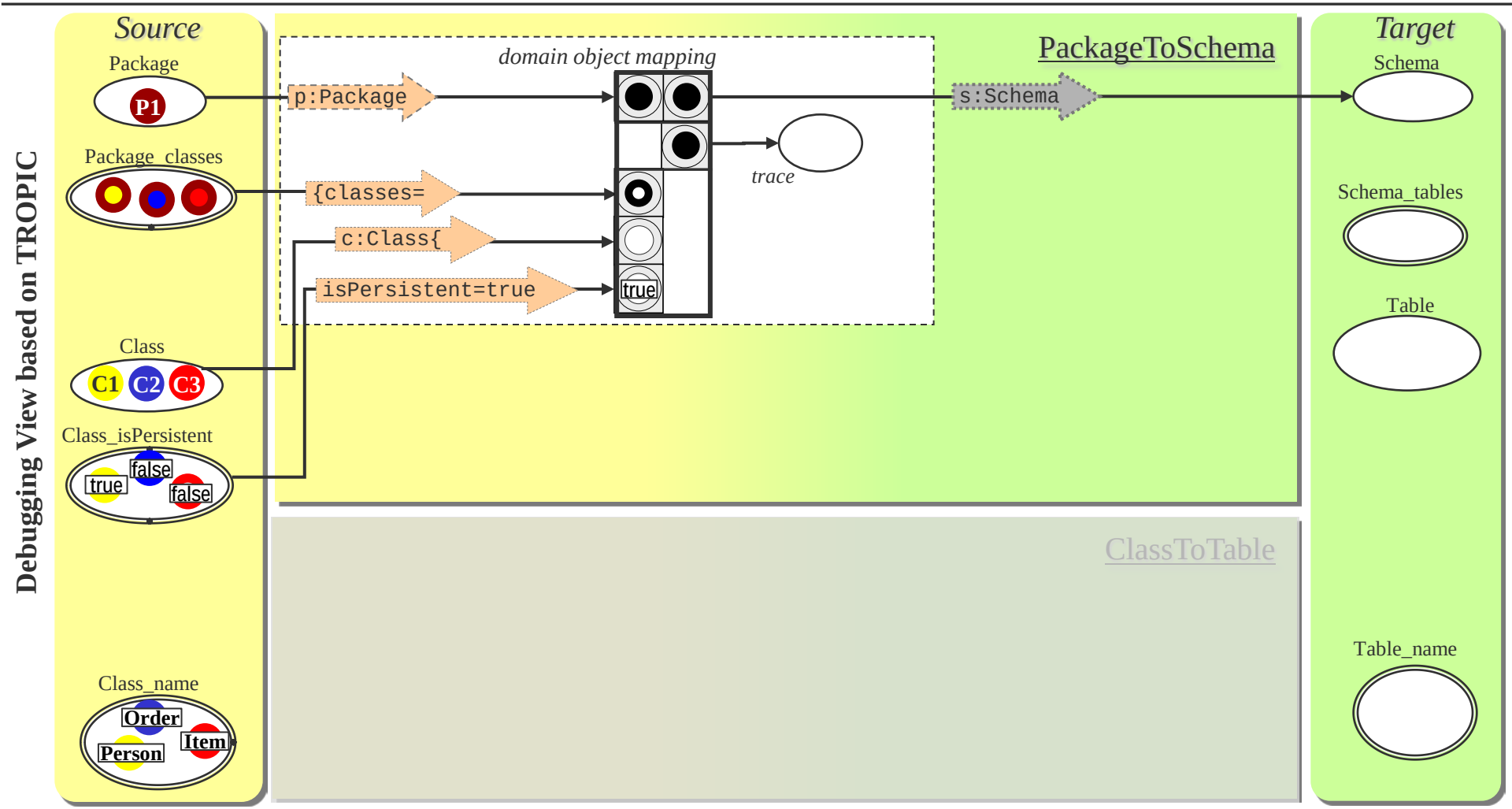


QVT Code

```
top relation PackageToSchema{
    checkonly domain class
      p:Package{
        classes=
          c:Class{
            isPersistent=true}};
    enforce domain rel
      s:Schema {
        tables=
          t:Table{}};
}
```

```
top relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn};
    enforce domain rel
      t:Table{
        name=cn};

}
```

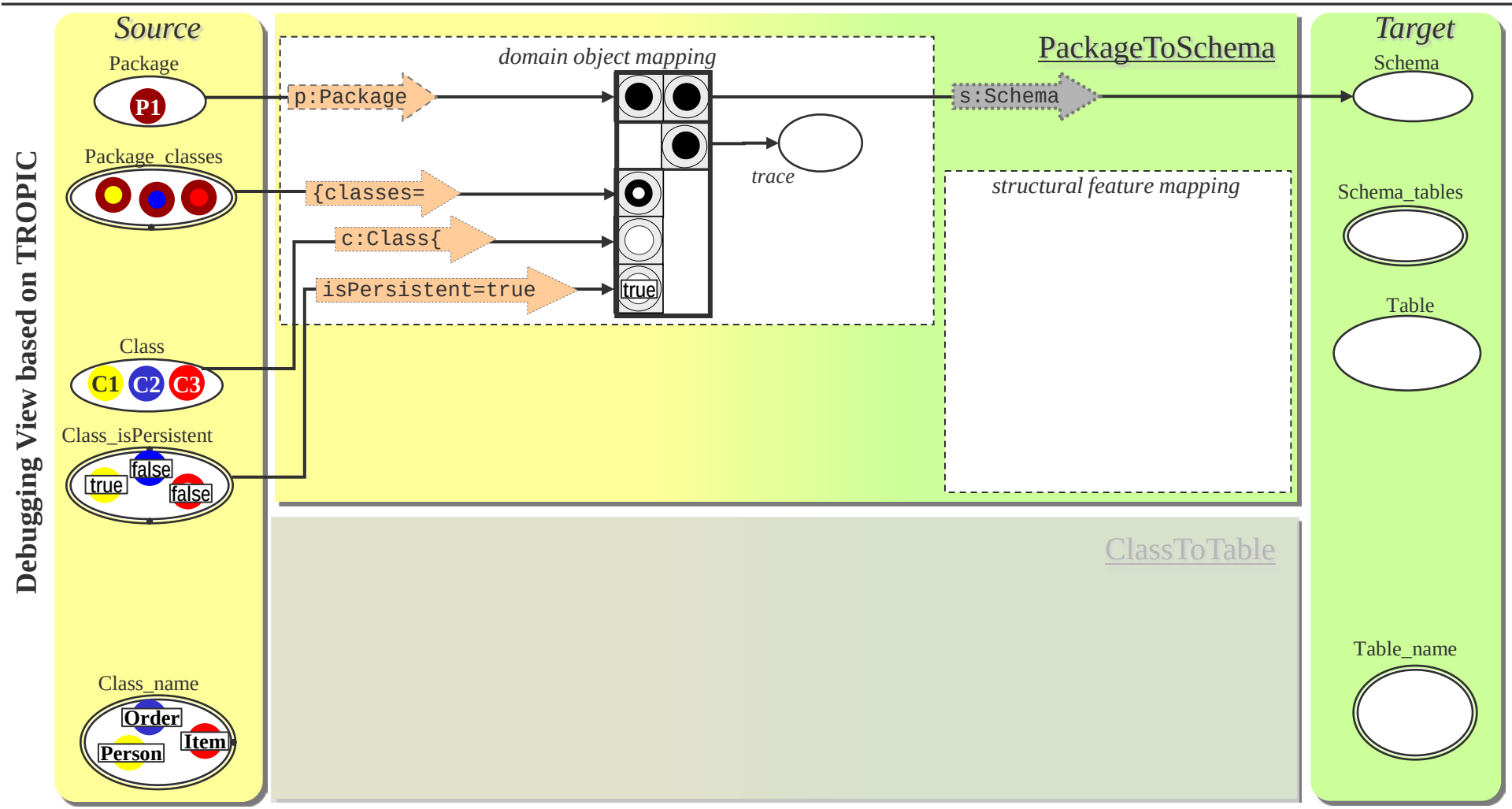


QVT Code

```
top relation PackageToSchema{
    checkonly domain class
      p:Package{
        classes=
          c:Class{
            isPersistent=true}};
    enforce domain rel
      s:Schema {
        tables=
          t:Table{}};
}
```

```
top relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn};
    enforce domain rel
      t:Table{
        name=cn};

}
```

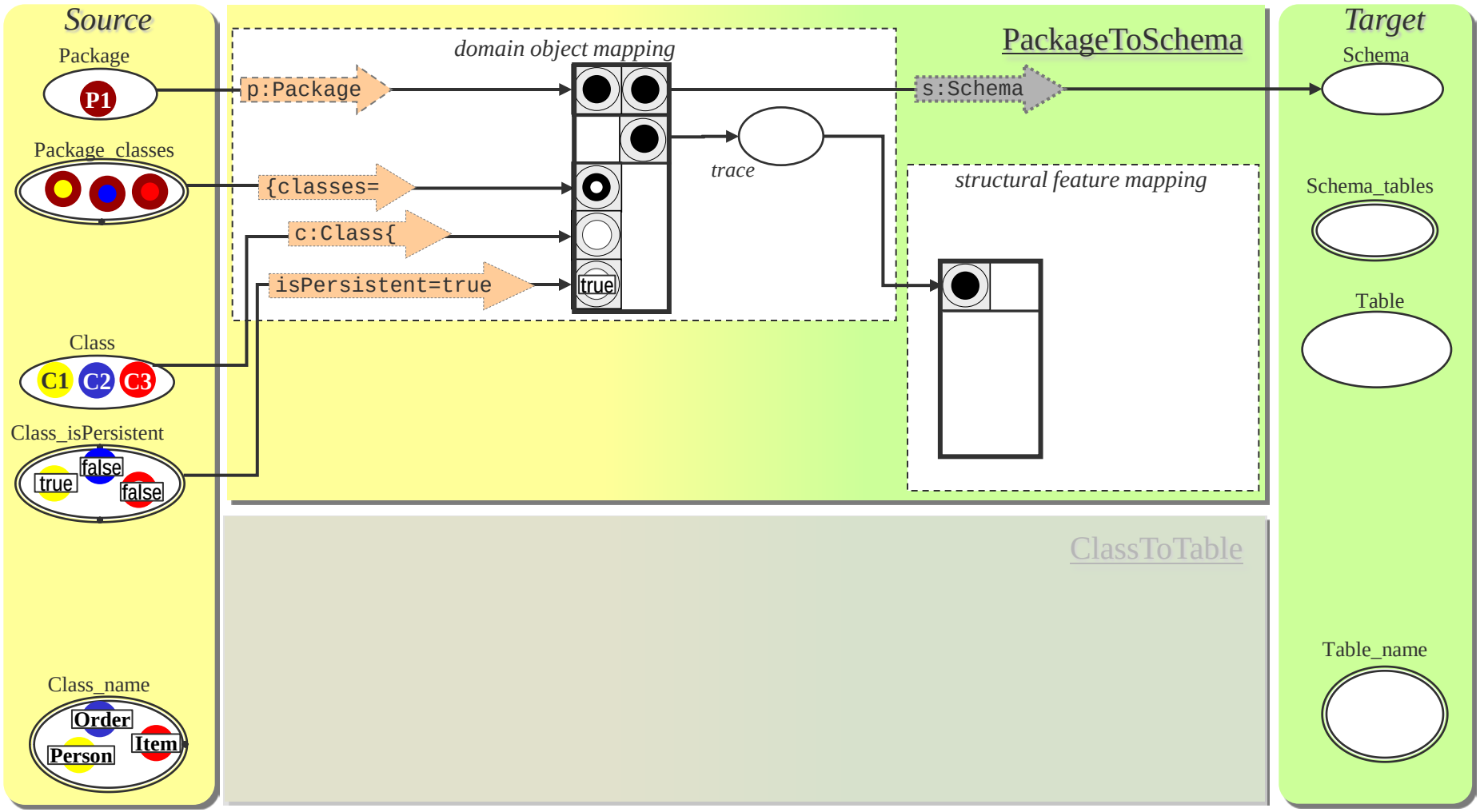


QVT Code

```
top relation PackageToSchema{
    checkonly domain class
      p:Package{
        classes=
          c:Class{
            isPersistent=true}};
    enforce domain rel
      s:Schema {
        tables=
          t:Table{}};
}
```

```
top relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn};
    enforce domain rel
      t:Table{
        name=cn};

}
```

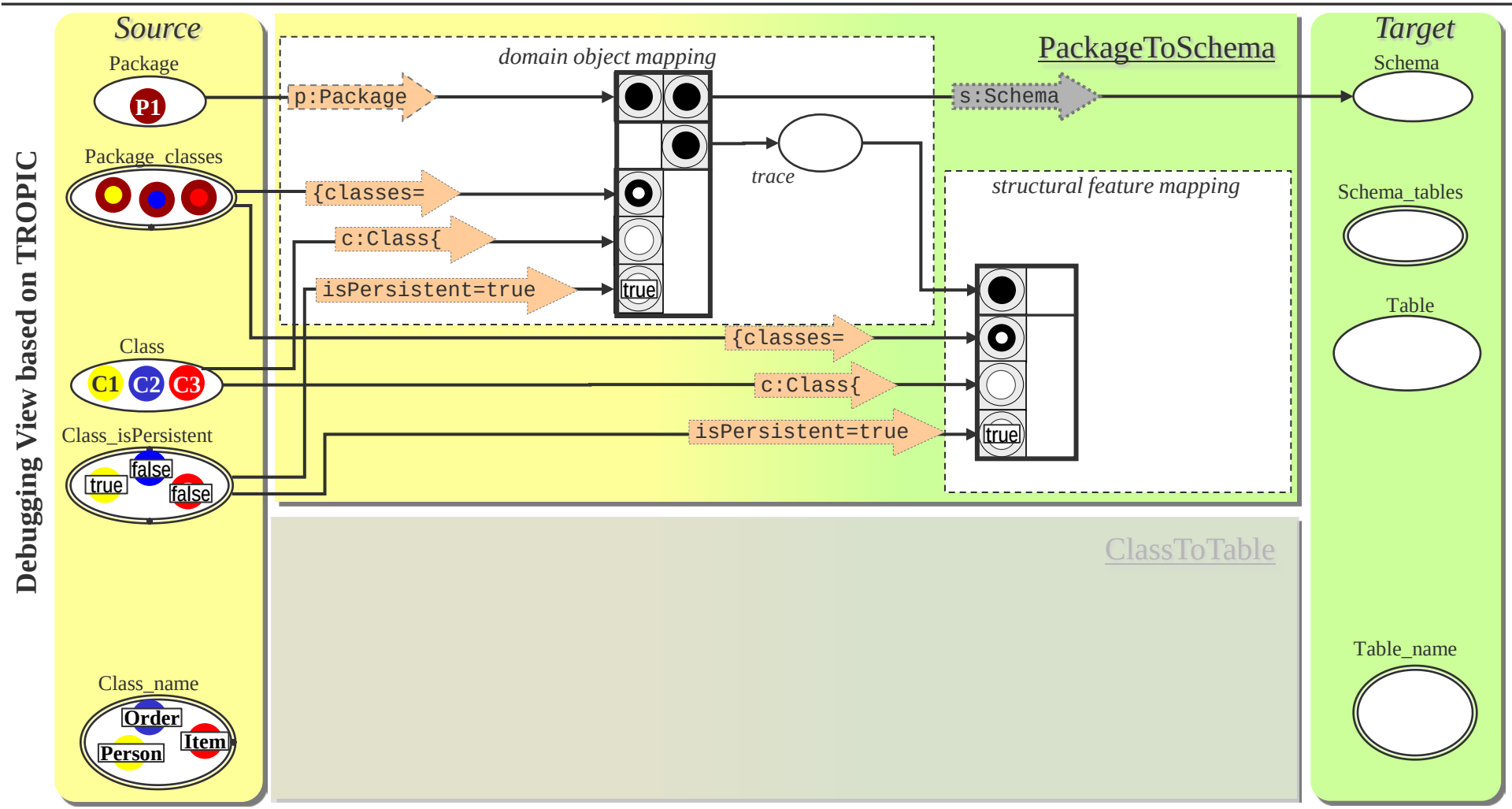


QVT Code

```
top relation PackageToSchema{
    checkonly domain class
      p:Package{
        classes=
          c:Class{
            isPersistent=true}};
    enforce domain rel
      s:Schema {
        tables=
          t:Table{}};
}
```

```
top relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn};
    enforce domain rel
      t:Table{
        name=cn};

}
```

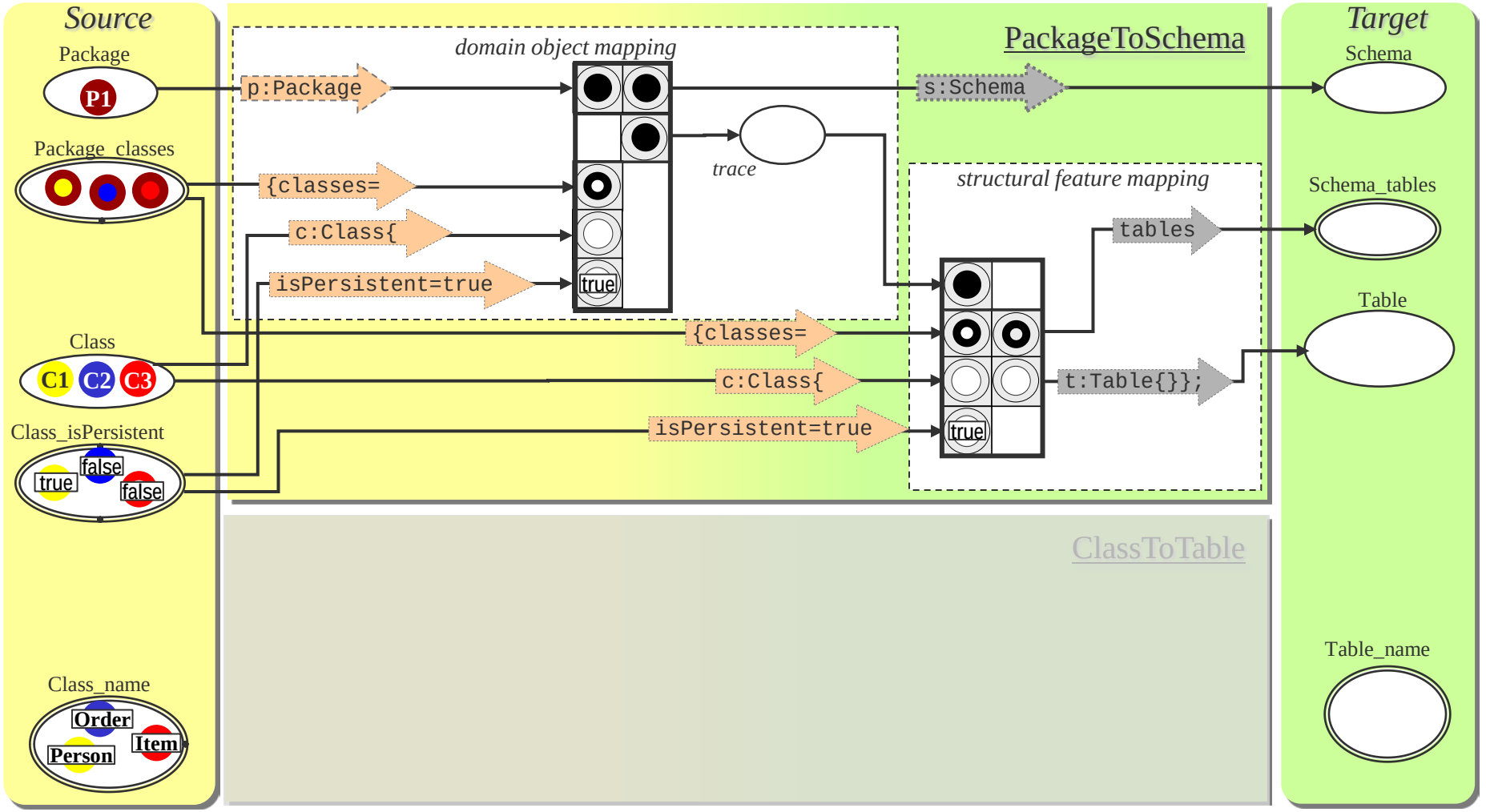


QVT Code

```
top relation PackageToSchema{
    checkonly domain class
      p:Package{
        classes=
          c:Class{
            isPersistent=true}};
    enforce domain rel
      s:Schema {
        tables=
          t:Table{}};
}
```

```
top relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn};
    enforce domain rel
      t:Table{
        name=cn};

}
```



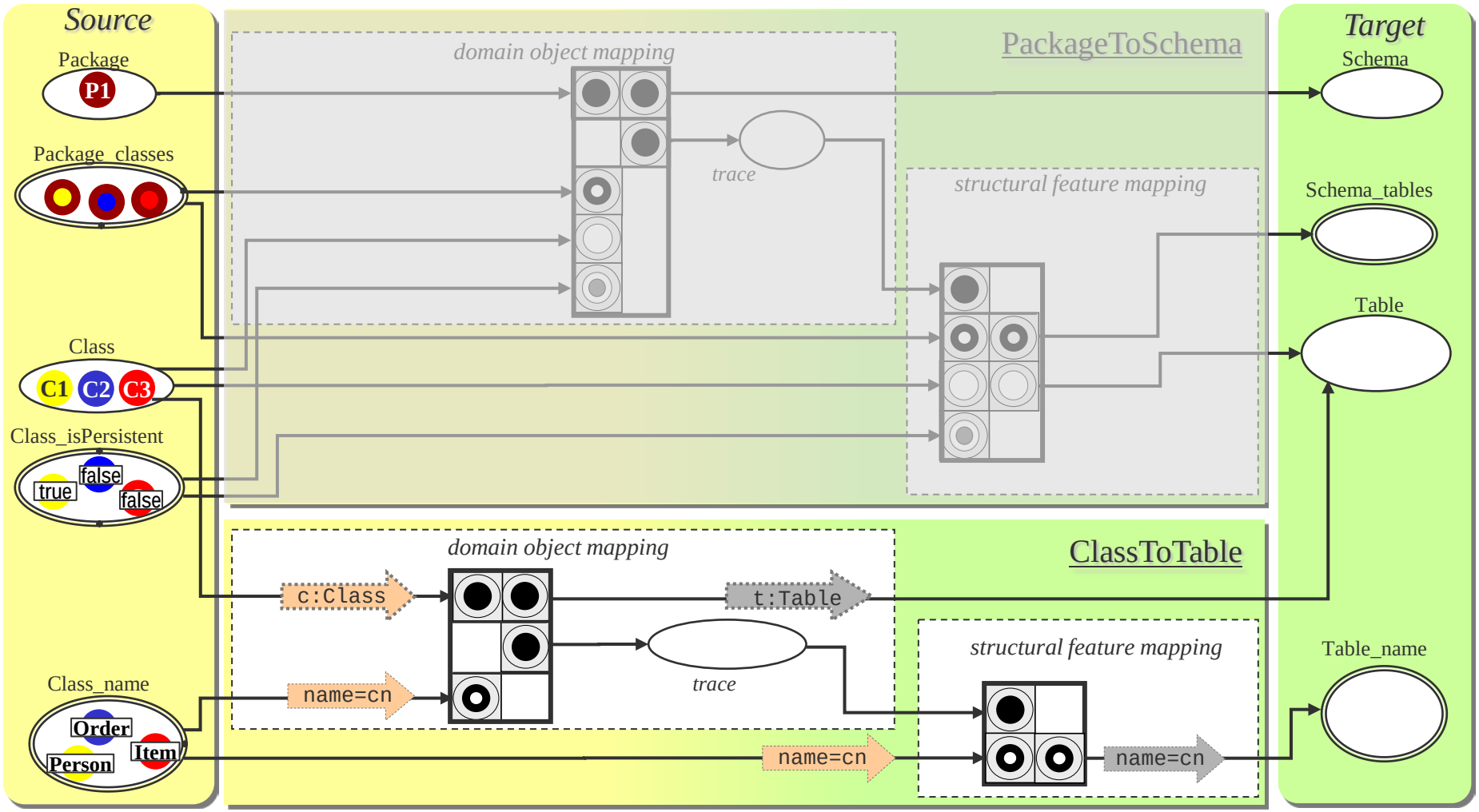
QVT Code

```
top relation PackageToSchema{
    checkonly domain class
      p:Package{
        classes=
          c:Class{
           isPersistent=true}};
    enforce domain rel
      s:Schema {
        tables=
          t:Table{}};
}
```

```
top relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn};
    enforce domain rel
      t:Table{
        name=cn};

}
```

Debugging View based on TROPIC



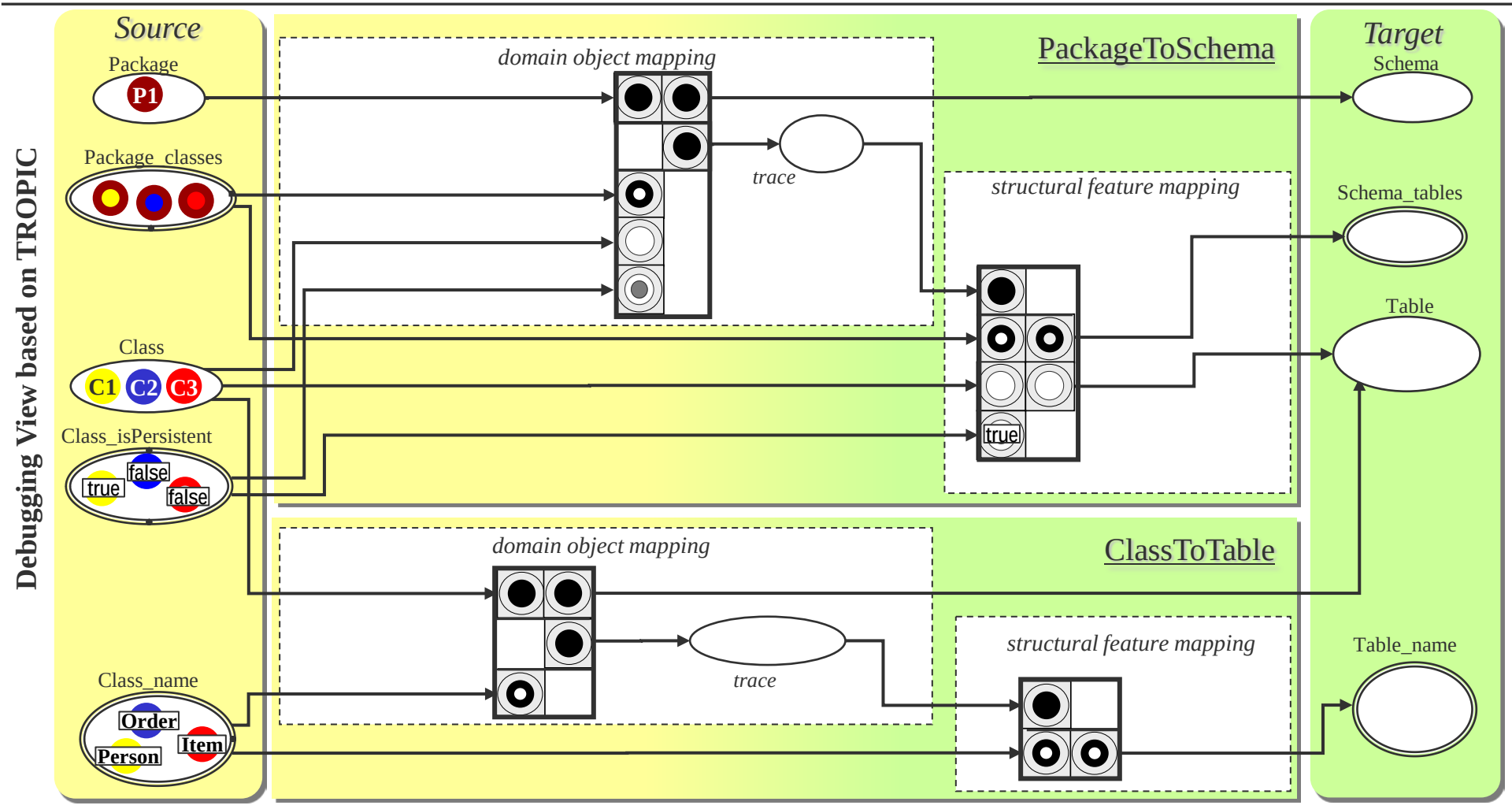
Running the Debugger

QVT Code

```
top relation PackageToSchema{
    checkonly domain class
      p:Package{
        classes=
          c:Class{
            isPersistent=true}};
    enforce domain rel
      s:Schema {
        tables=
          t:Table{}};
}
```

```
top relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn};
    enforce domain rel
      t:Table{
        name=cn};

}
```



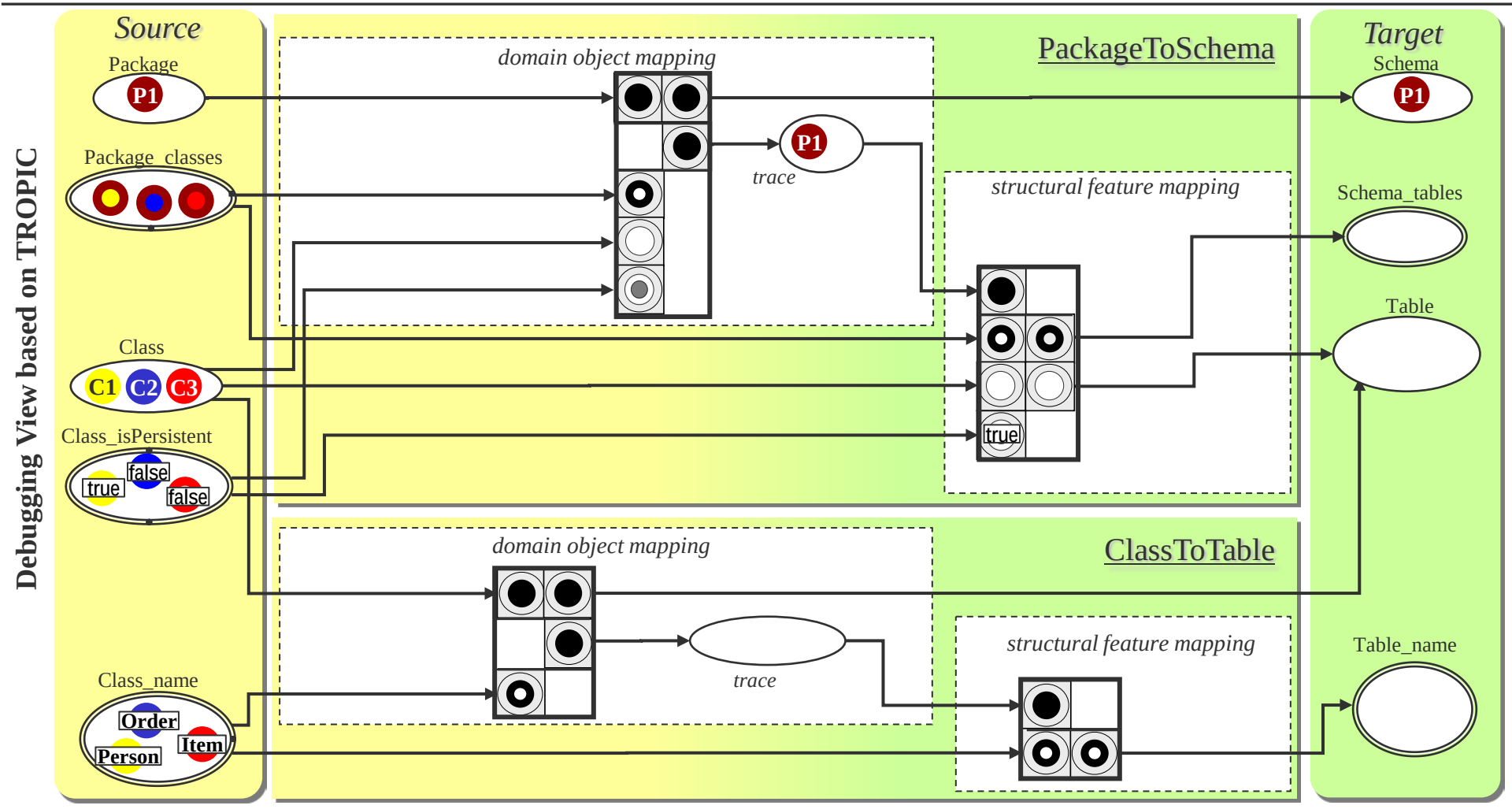
Running the Debugger

QVT Code

```
top relation PackageToSchema{
    checkonly domain class
      p:Package{
        classes=
          c:Class{
            isPersistent=true}};
    enforce domain rel
      s:Schema {
        tables=
          t:Table{}};
}
```

```
top relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn};
    enforce domain rel
      t:Table{
        name=cn};

}
```



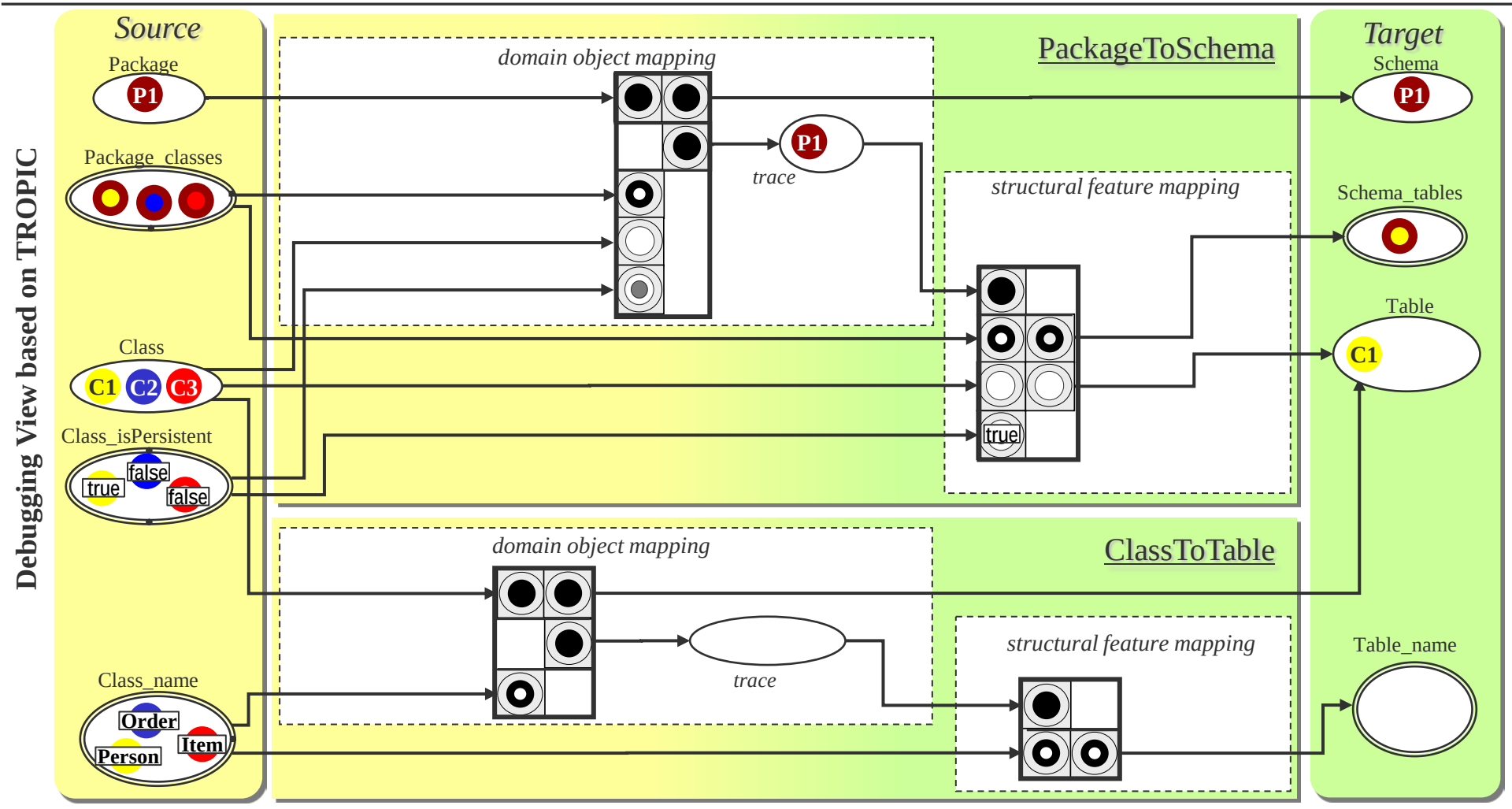
Running the Debugger

QVT Code

```
top relation PackageToSchema{
    checkonly domain class
      p:Package{
        classes=
          c:Class{
            isPersistent=true}};
    enforce domain rel
      s:Schema {
        tables=
          t:Table{}};
}
```

```
top relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn};
    enforce domain rel
      t:Table{
        name=cn};

}
```



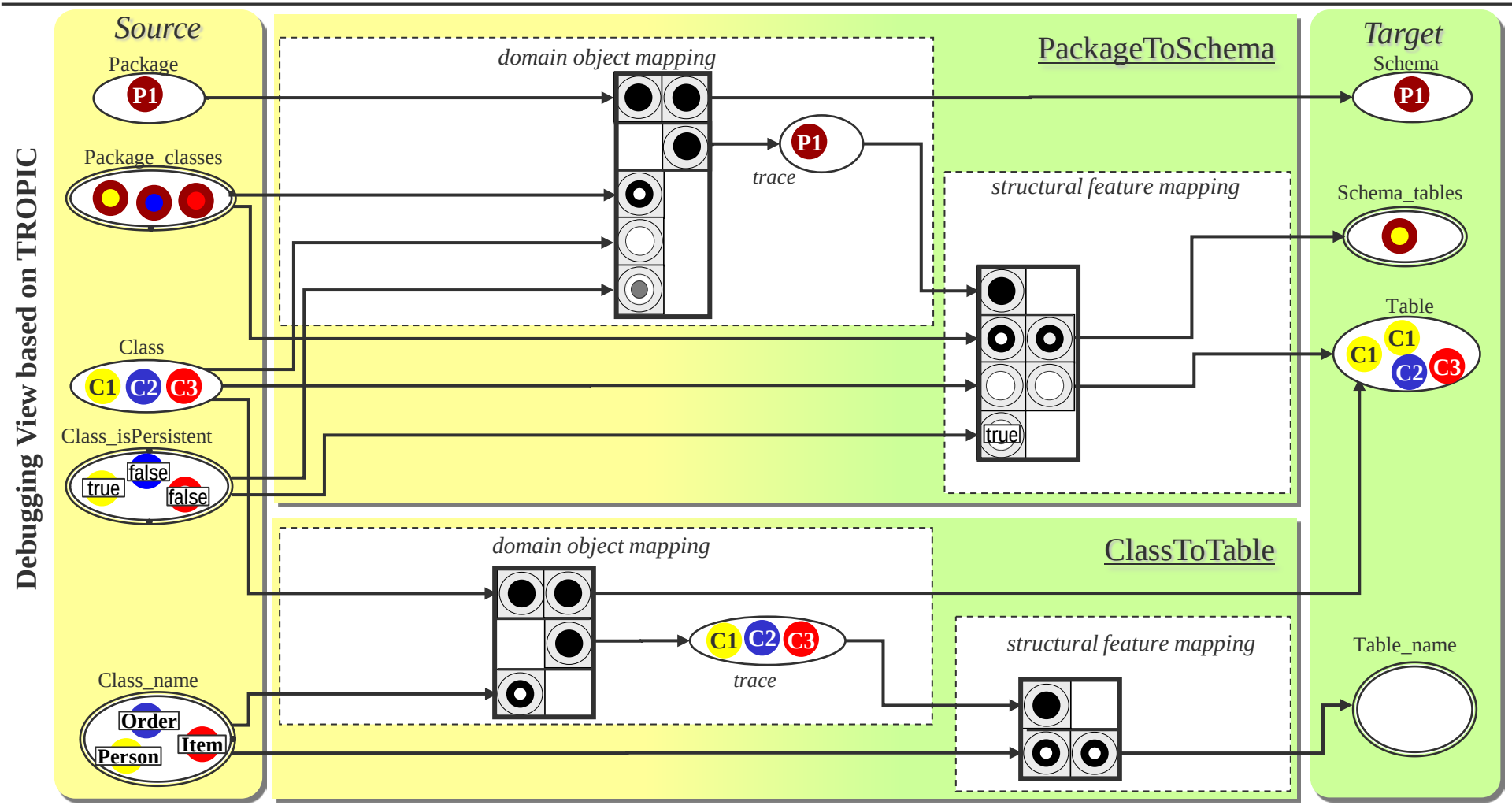
Running the Debugger

QVT Code

```
top relation PackageToSchema{
    checkonly domain class
      p:Package{
        classes=
          c:Class{
            isPersistent=true}};
    enforce domain rel
      s:Schema {
        tables=
          t:Table{}};
}
```

```
top relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn};
    enforce domain rel
      t:Table{
        name=cn};

}
```



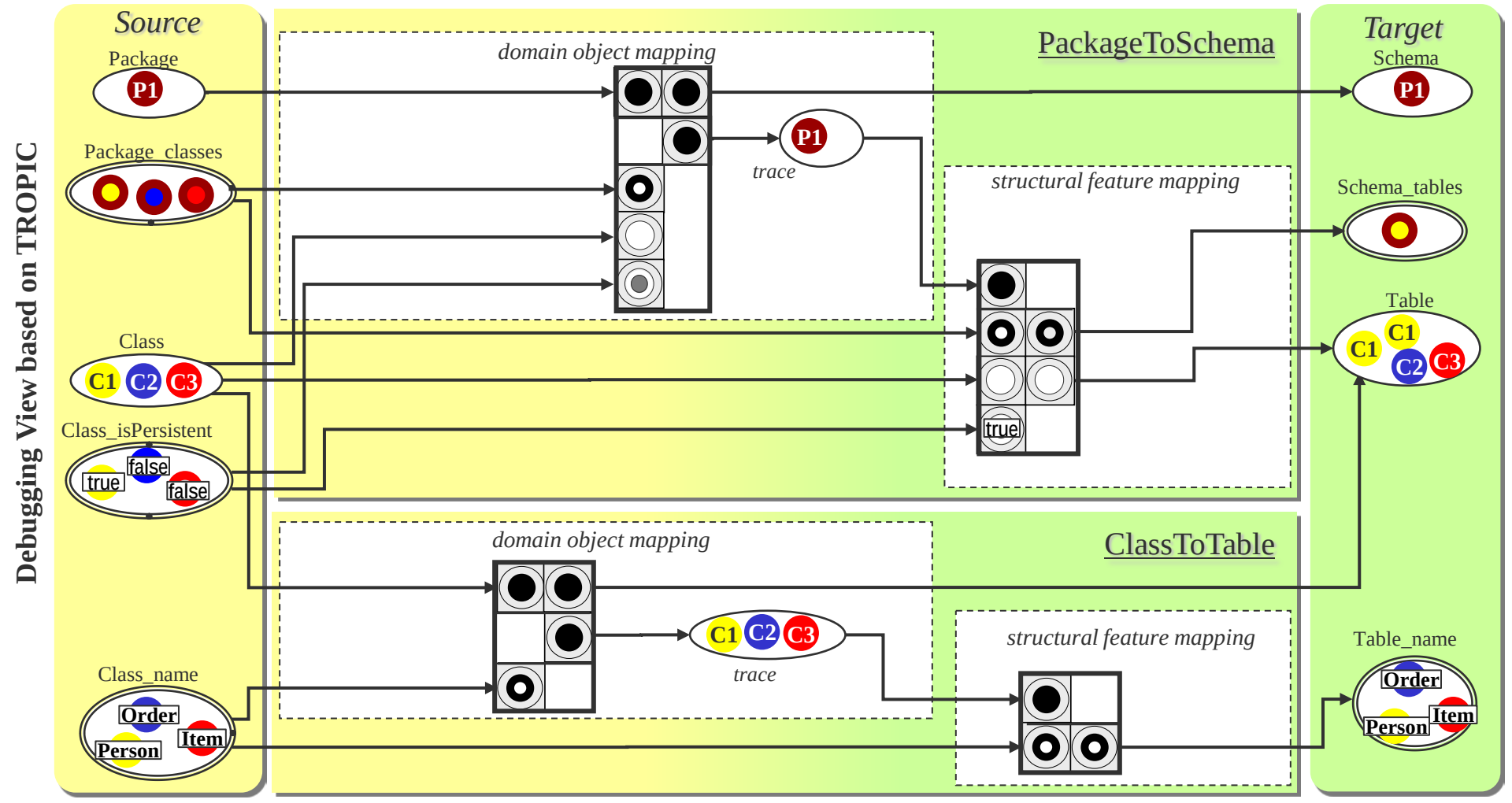
Running the Debugger

QVT Code

```
top relation PackageToSchema{
    checkonly domain class
      p:Package{
        classes=
          c:Class{
            isPersistent=true}};
    enforce domain rel
      s:Schema {
        tables=
          t:Table{}};
}
```

```
top relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn};
    enforce domain rel
      t:Table{
        name=cn};

}
```



Running the Debugger

Problems:

(1) ClassToTable is to weak

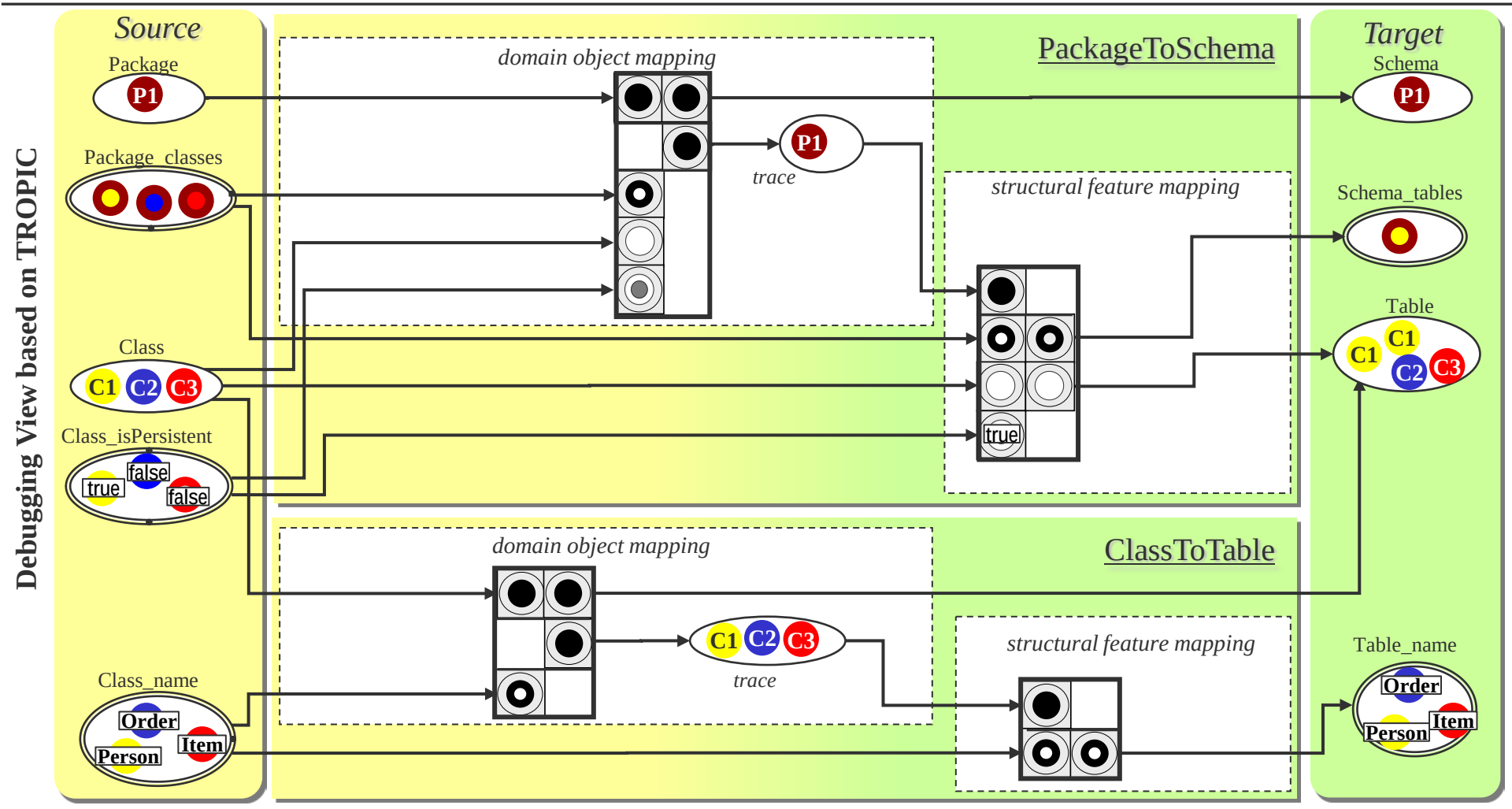
(2) Relations are working independently

QVT Code

```
top relation PackageToSchema{
    checkonly domain class
      p:Package{
        classes=
          c:Class{
            isPersistent=true}};
    enforce domain rel
      s:Schema {
        tables=
          t:Table{}};
}
```

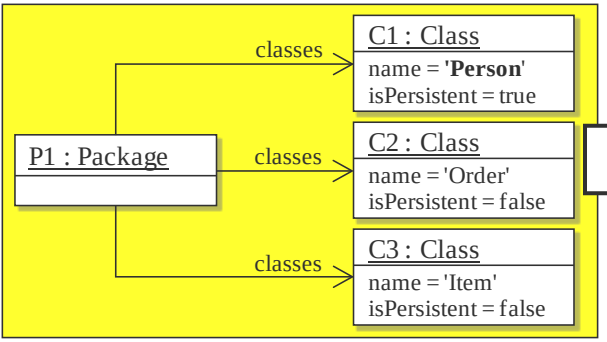
```
top relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn};
    enforce domain rel
      t:Table{
        name=cn};

}
```



UML 2 Relations: Second Solution

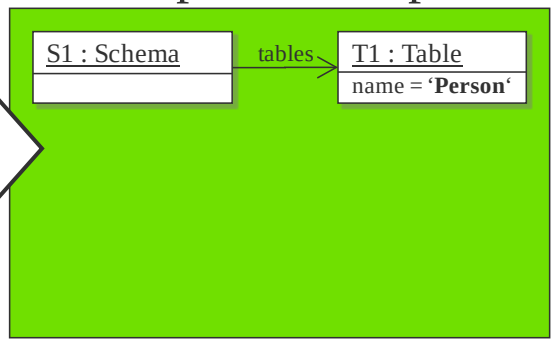
Input



QVT Code

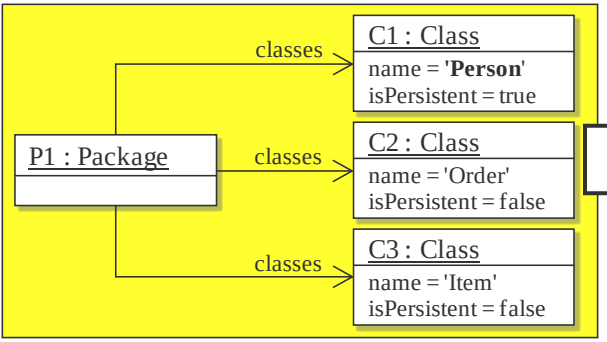
```
transformation ClassToRel(
   class:Class1, rel:Relational1){
 top relation PackageToSchema{
    checkonly domain class
      p:Package {
        classes=
          c:Class{}};
    enforce domain rel
      s:Schema {
        tables=
          t:Table{}};
    where{
        ClassToTable(c,t);
    }
 }
 relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn,
        isPersistent=true};
    enforce domain rel
      t:Table{
        name=cn};
 }
}
```

Expected Output



UML 2 Relations: Second Solution

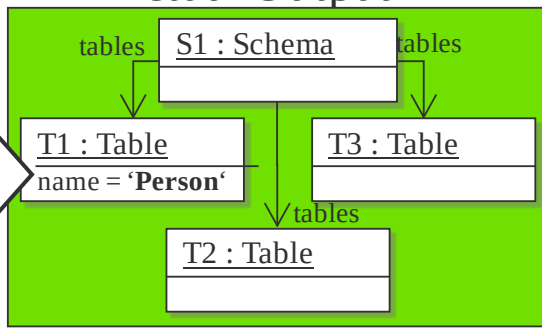
Input



QVT Code

```
transformation ClassToRel(
   class:Class1, rel:Relational1){
 top relation PackageToSchema{
    checkonly domain class
      p:Package {
        classes=
          c:Class{}};
    enforce domain rel
      s:Schema {
        tables=
          t:Table{}};
    where{
        ClassToTable(c,t);
    }
 }
 relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn,
        isPersistent=true};
    enforce domain rel
      t:Table{
        name=cn};
 }
}
```

Actual Output

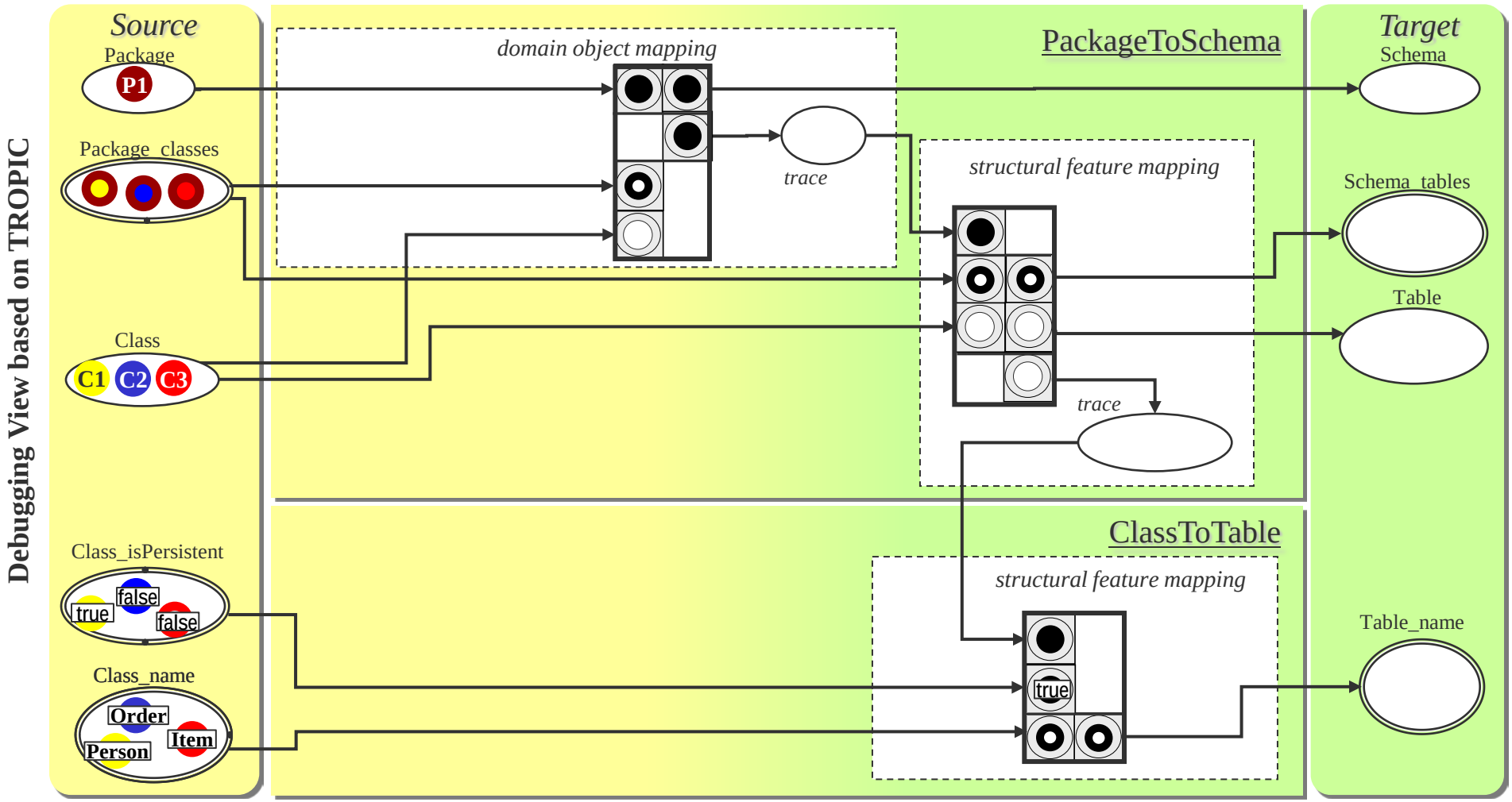


Running the Debugger

QVT Code

```
top relation PackageToSchema{
    checkonly domain class
      p:Package {
        classes= c:Class{}};
    enforce domain rel
      s:Schema {
        tables= t:Table{}};
    where{
        ClassToTable(c,t);
    }
}
```

```
relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn,
        isPersistent=true};
    enforce domain rel
      t:Table{
        name=cn};
}
```

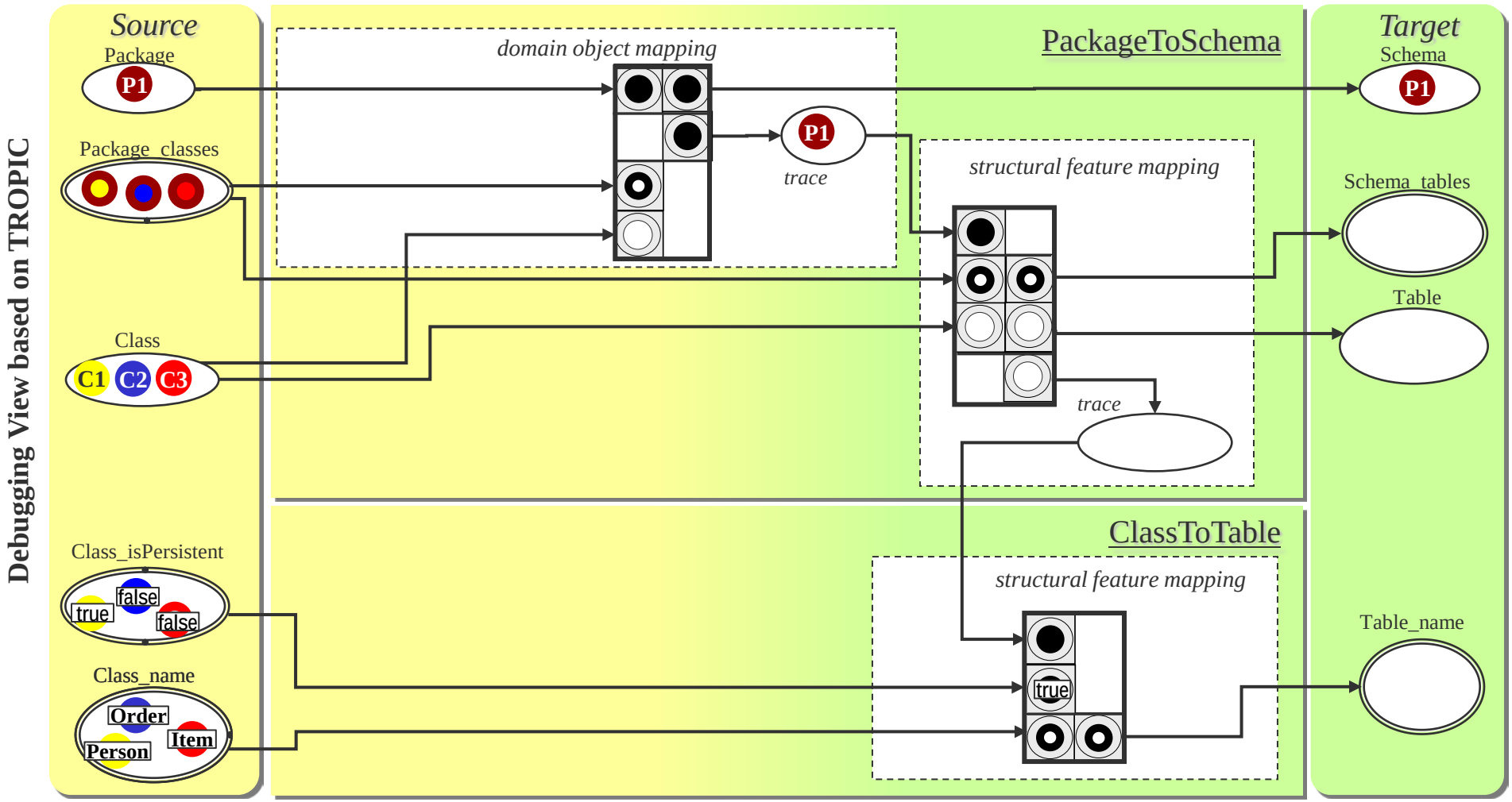


Running the Debugger

QVT Code

```
top relation PackageToSchema{
    checkonly domain class
      p:Package {
        classes= c:Class{}};
    enforce domain rel
      s:Schema {
        tables= t:Table{}};
    where{
        ClassToTable(c,t);
    }
}
```

```
relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn,
        isPersistent=true};
    enforce domain rel
      t:Table{
        name=cn};
}
```

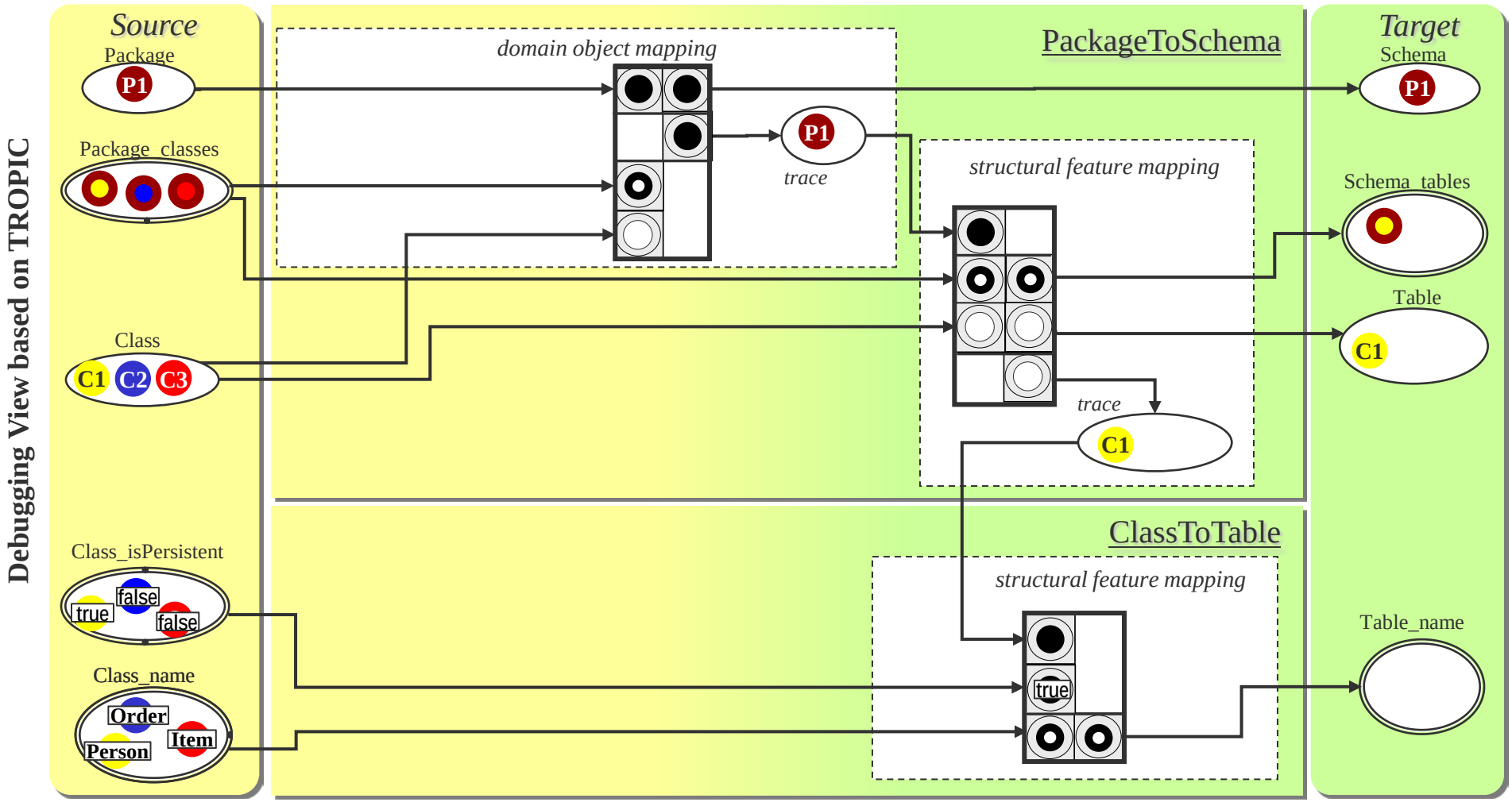


Running the Debugger

QVT Code

```
top relation PackageToSchema{
    checkonly domain class
      p:Package {
        classes= c:Class{}};
    enforce domain rel
      s:Schema {
        tables= t:Table{}};
    where{
        ClassToTable(c,t);
    }
}
```

```
relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn,
        isPersistent=true};
    enforce domain rel
      t:Table{
        name=cn};
}
```

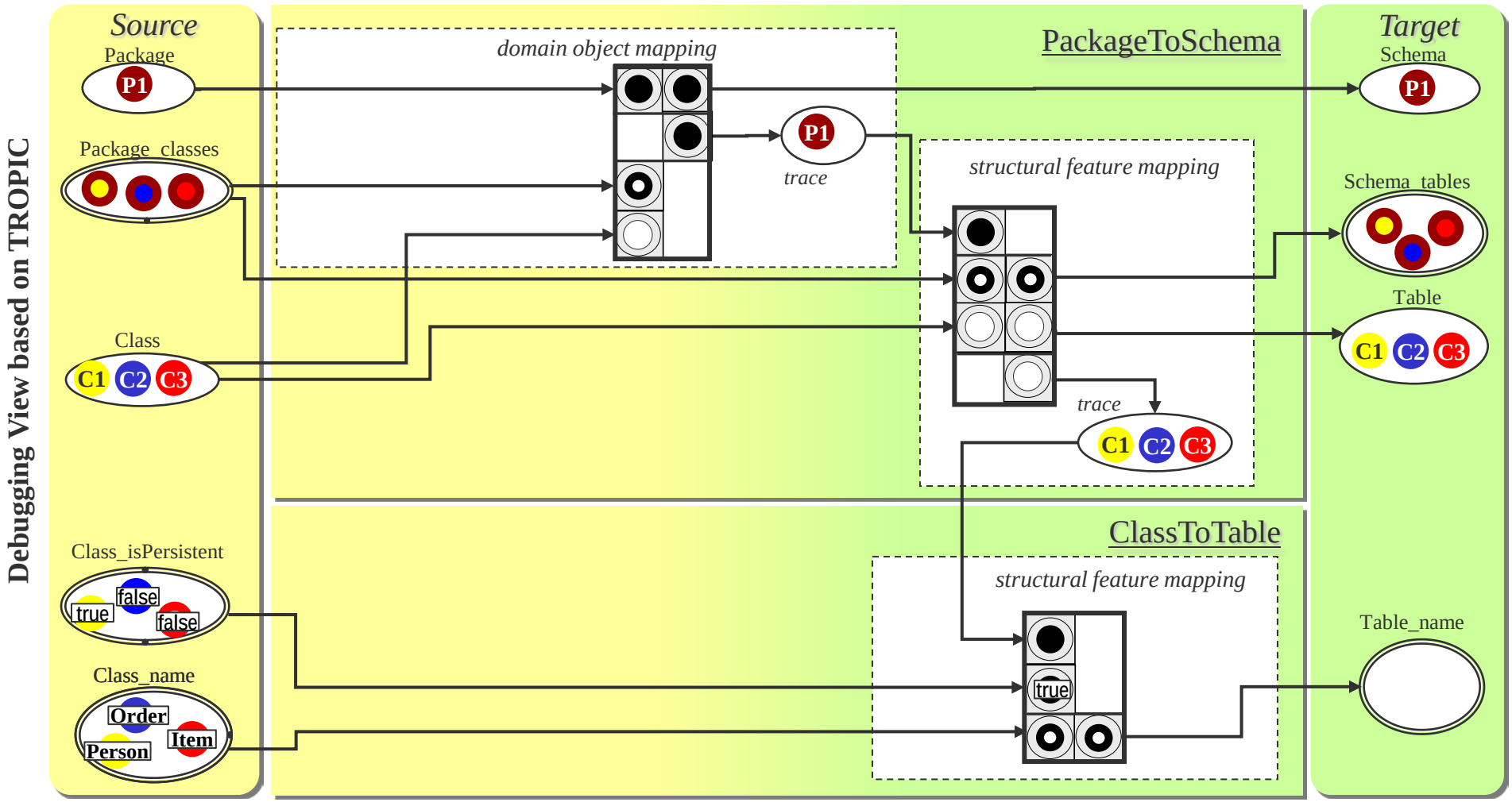


Running the Debugger

QVT Code

```
top relation PackageToSchema{
    checkonly domain class
      p:Package {
        classes= c:Class{}};
    enforce domain rel
      s:Schema {
        tables= t:Table{}};
    where{
        ClassToTable(c,t);
    }
}
```

```
relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn,
        isPersistent=true};
    enforce domain rel
      t:Table{
        name=cn};
}
```

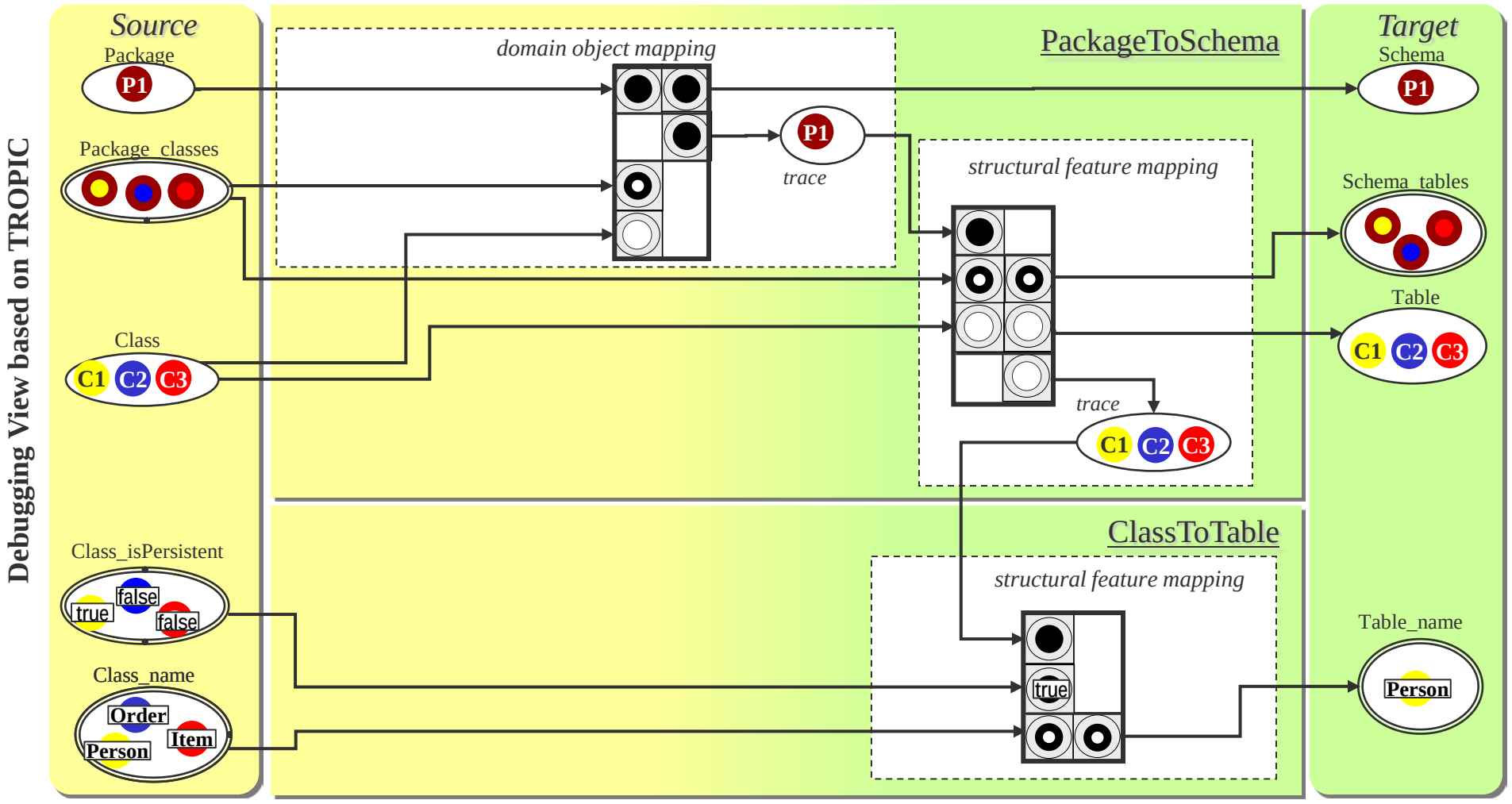


Running the Debugger

QVT Code

```
top relation PackageToSchema{
    checkonly domain class
      p:Package {
        classes= c:Class{}};
    enforce domain rel
      s:Schema {
        tables= t:Table{}};
    where{
        ClassToTable(c,t);
    }
}
```

```
relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn,
        isPersistent=true};
    enforce domain rel
      t:Table{
        name=cn};
}
```



Running the Debugger

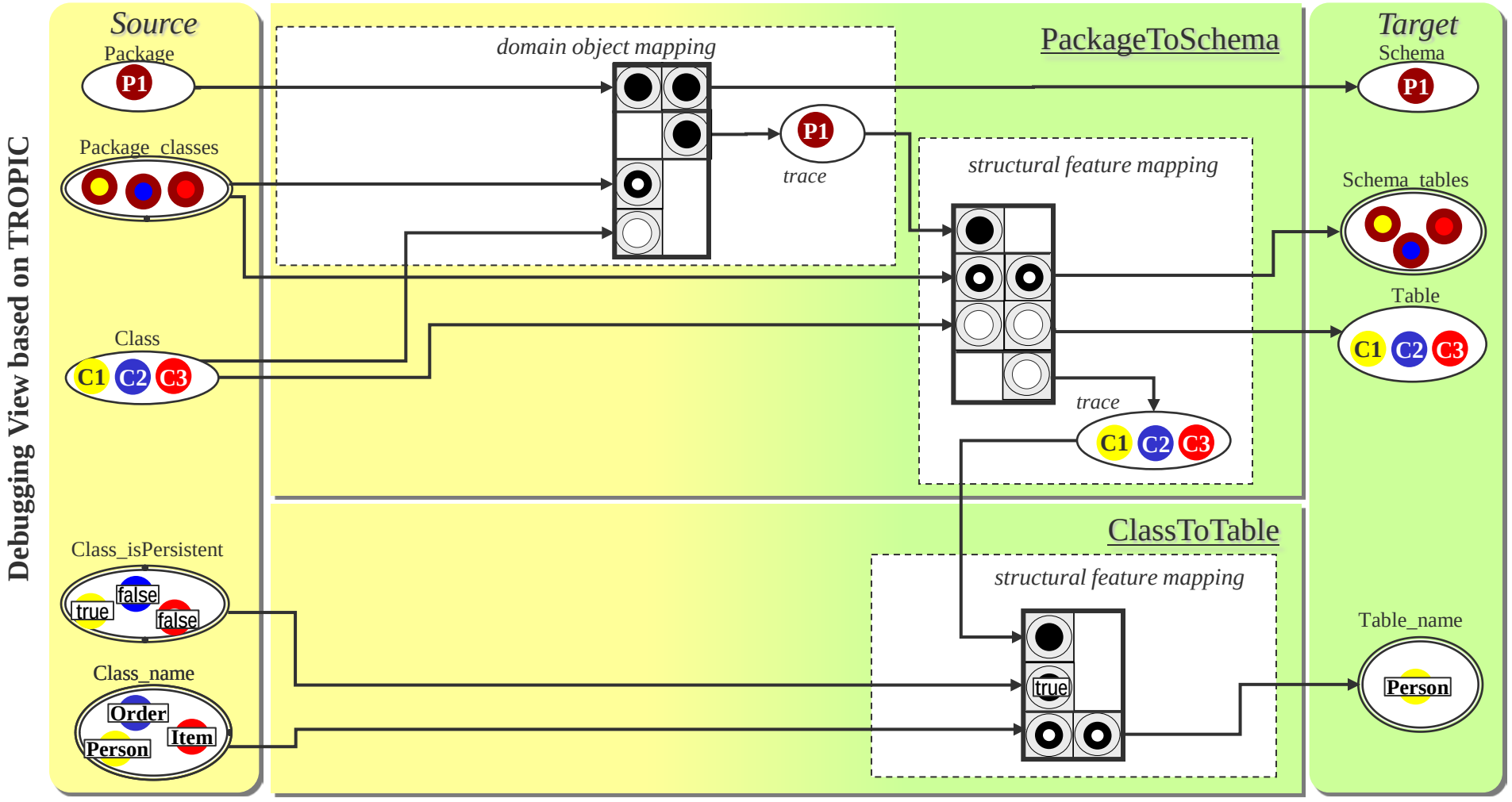
Problems:

(1) isPersistent has to be checked in PackageToSchema

QVT Code

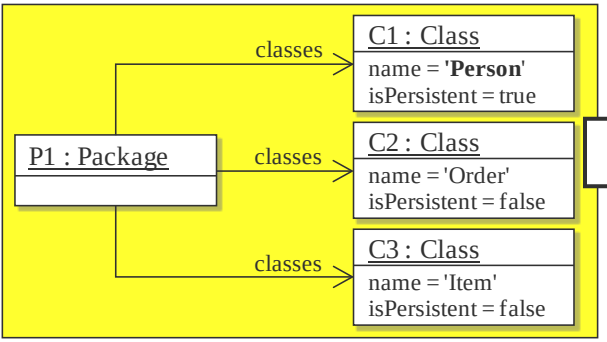
```
top relation PackageToSchema{
    checkonly domain class
      p:Package {
        classes= c:Class{}};
    enforce domain rel
      s:Schema {
        tables= t:Table{}};
    where{
        ClassToTable(c,t);
    }
}
```

```
relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn,
        isPersistent=true};
    enforce domain rel
      t:Table{
        name=cn};
}
```



UML 2 Relations: Third and correct Solution

Input



QVT Code

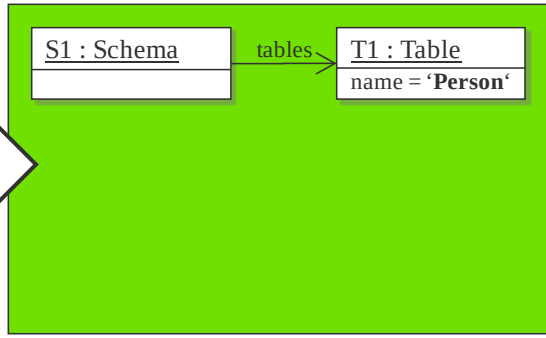
```
transformation ClassToRel(
   class:Class1, rel:Relational1){

 top relation PackageToSchema{
    checkonly domain class
      p:Package {
        classes=
          c:Class{
            isPersistent=true}};
    enforce domain rel
      s:Schema {
        tables=
          t:Table{}};
    where{
        ClassToTable(c,t);
    }
 }
 relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn};
    enforce domain rel
      t:Table{
        name=cn};
 }
}
```

I

O

Expected Output



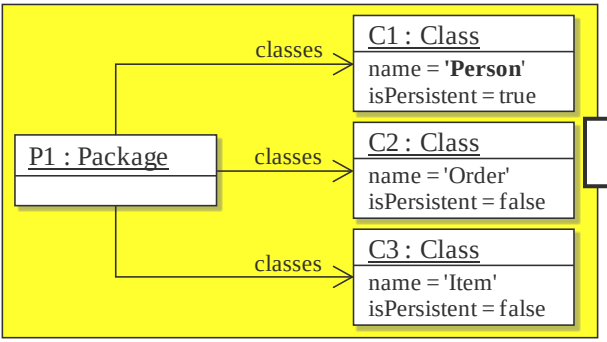
UML 2 Relations: Third and correct Solution

QVT Code

```
transformation ClassToRel(
   class:Class1, rel:Relational1){

 top relation PackageToSchema{
    checkonly domain class
      p:Package {
        classes=
          c:Class{
            isPersistent=true}};
    enforce domain rel
      s:Schema {
        tables=
          t:Table{}};
    where{
        ClassToTable(c,t);
    }
 }
 relation ClassToTable{
    cn: String;
    checkonly domain class
      c:Class{
        name=cn};
    enforce domain rel
      t:Table{
        name=cn};
 }
}
```

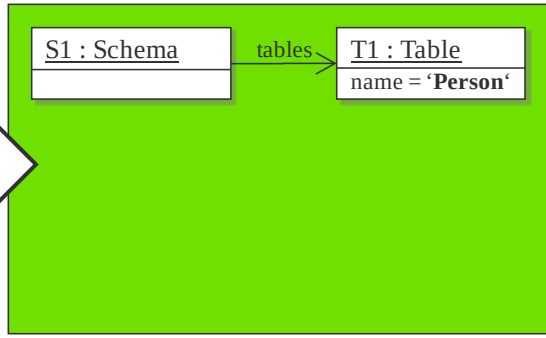
Input



I

O

Actual Output



Conclusion & Ongoing Work

- Graphical debugging of QVT Relations based on TROPIC provides
  - explicit representation of operational semantics
  - homogenous representation of all artifacts

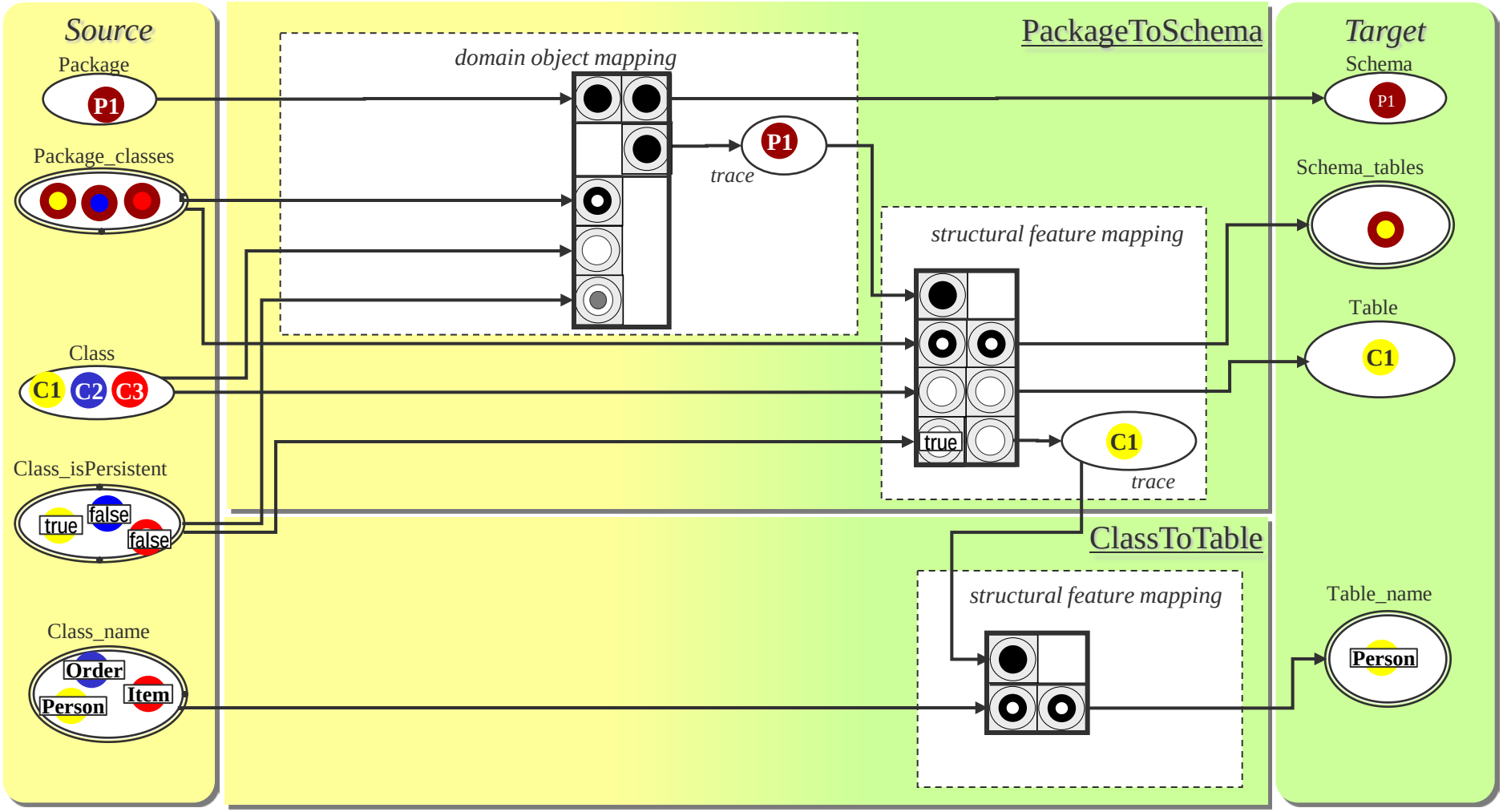
- No common understanding how QVT Relations should be used
  - How much transformation logic should be located in one relation?
  - How should be relations linked appropriately?
  - First attempt: Taxonomy of common pitfalls
  - What is missing: QVT Guide

- Ongoing work
  - OCL query debugger
  - Propagating bug fixes from TROPIC back to QVT Relations

Backup Slides

Backup Slides

Transformation Net for correct Solution



Backup Slides

Taxonomy of Common Pitfalls of Using QVT Relations