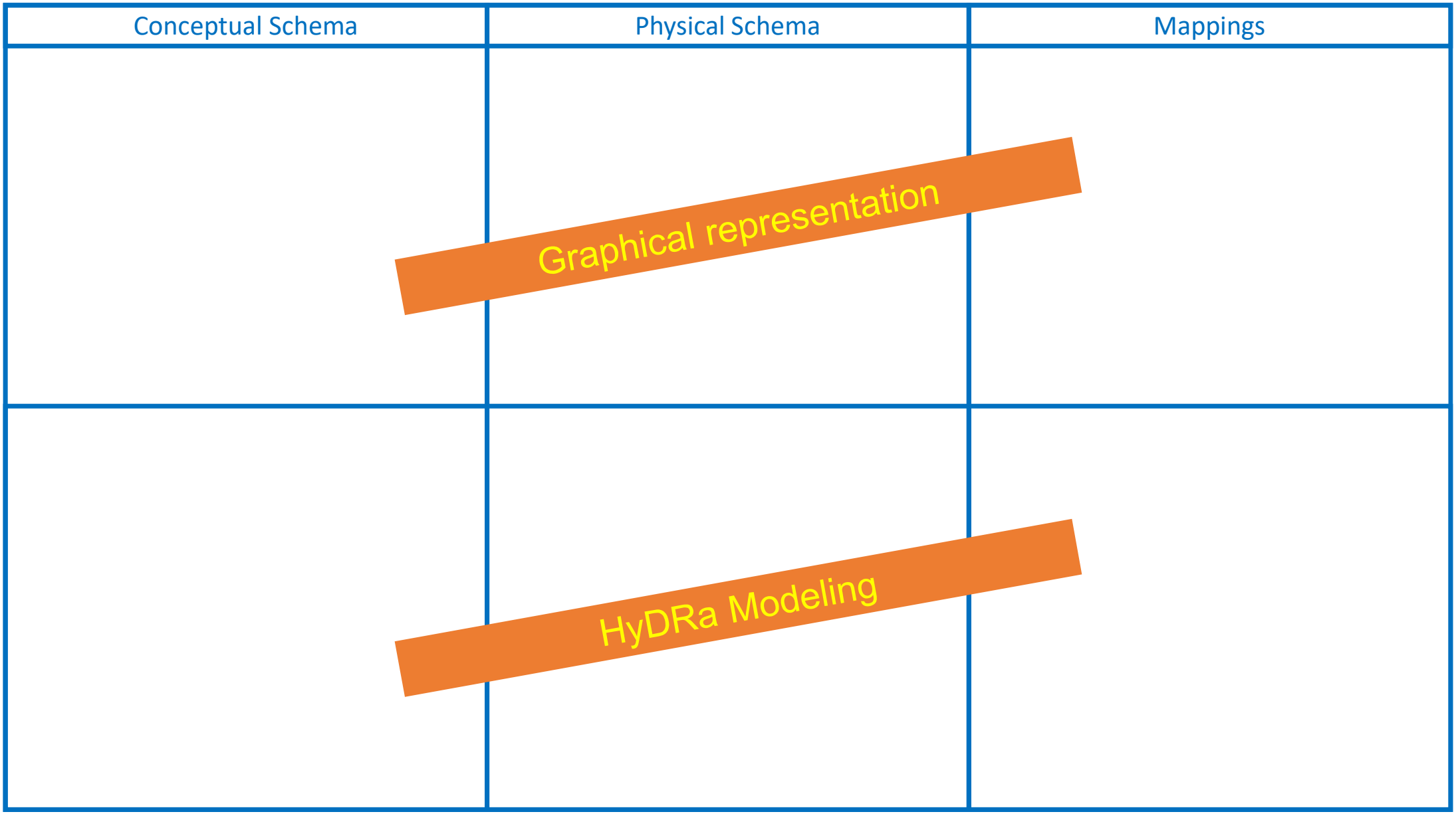
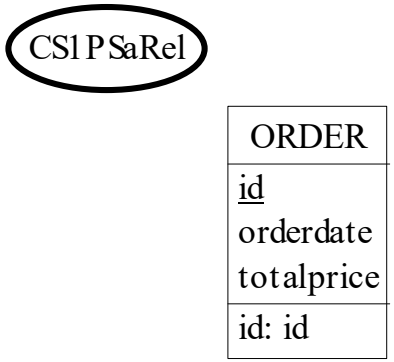
HyDRa – Mapping rules guide

Loup Meurice, Maxime Gobert and Anthony Cleve

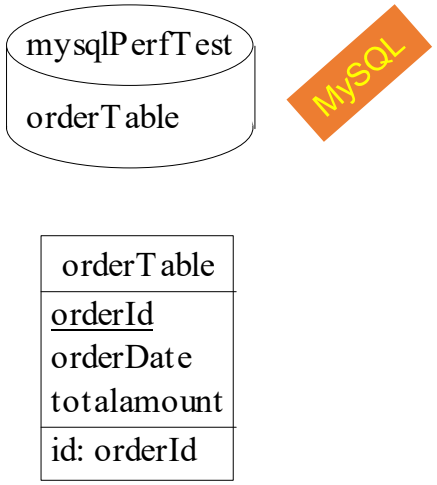

Conceptual Schema
Physical Schema
Mappings
Graphical representation
HyDRa Modeling

Single Entity Type – One Database

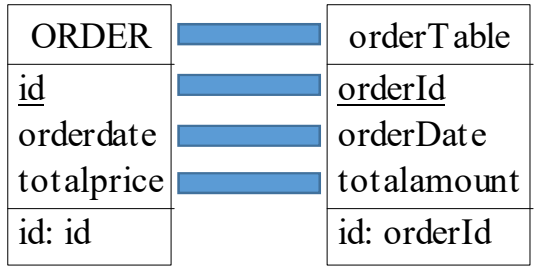
Conceptual Schema



Physical Schema



Mappings

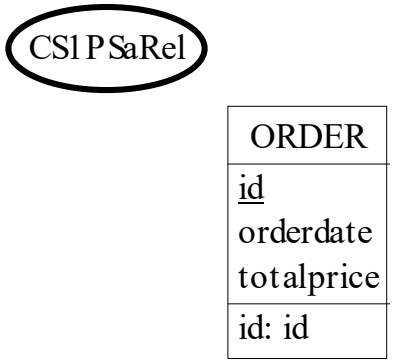



```
conceptual schema CS1PSaRel {

    entity type Order {
        id : string,
        orderdate : date,
        totalprice : float
        identifier {
            id
        }
    }

}
```

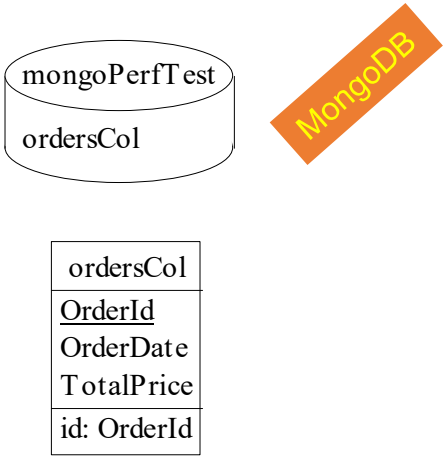
```
databases {
        mysql mysqlPerfTest{
        dbname : "mysqlPerfTest"
        host : "mysqlPerfTest"
        login : "root"
        password : "password"
        port : 3306
    }
}

physical schemas {
    relational schema relSchema : mysqlPerfTest {
        table orderTable {
            columns {
                orderId,
                orderDate,
                totalamount,
                customerId
            }
        }
    }
}
```

```
mapping rules {
    CS1PSaRel.Order(id, orderdate, totalprice)
    -> relSchema.orderTable( orderId, orderDate, totalamount)
}
```

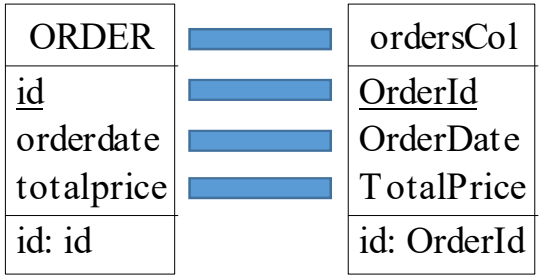
Conceptual Schema



Physical Schema



Mappings

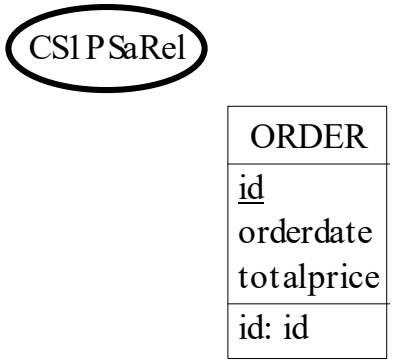



```
conceptual schema CS1PSaRel {

    entity type Order {
        id : string,
        orderdate : date,
        totalprice : float
        identifier {
            id
        }
    }

}
```

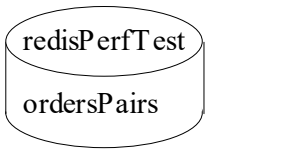
```
databases {
        mongodb mongoPerfTest{
        host:"mongoPerfTest"
        port : 27017
    }
}

physical schemas {
    document schema mongoSchema0: mongoPerfTest {

        collection ordersCol {
            fields {
                OrderId,
                OrderDate,
                TotalPrice
            }
        }

    }
}
```

```
mapping rules {
    CS1PSaDoc.Order(id,orderdate,totalprice)
    -> mongoSchema0.ordersCol(OrderId,OrderDate,TotalPrice)
}
```

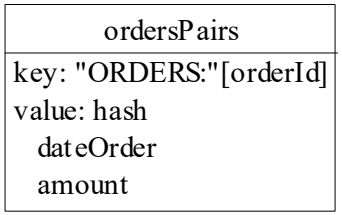
Conceptual Schema



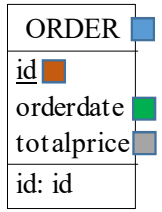
Physical Schema

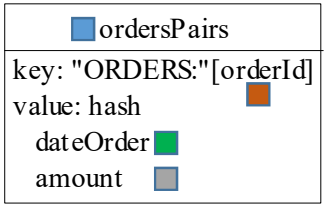


Redis



Mappings





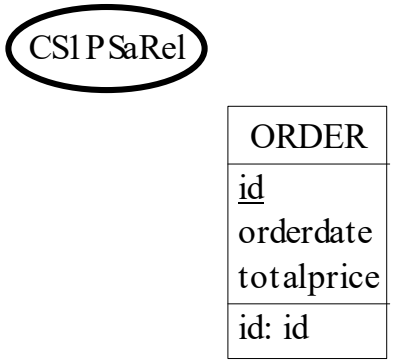

```
conceptual schema CS1PSaRel {

    entity type Order {
        id : string,
        orderdate : date,
        totalprice : float
        identifier {
            id
        }
    }

}
```

```
databases {
    redis redisPerfTest {
        host : "redisPerfTest"
        port : 6379
    }
}


physical schemas {
    key value schema kv : redisPerfTest {
        kvpairs ordersPairs {
            key : "ORDER:"[orderid],
            value : hash {
                dateOrder,
                amount
            }
        }
    }
}
```

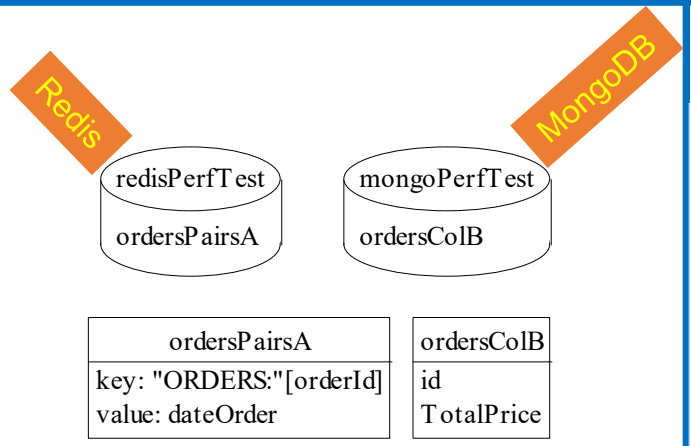
```
mapping rules {
    CS1PSaKey.Order(id,totalprice, orderdate)
    -> kv.ordersPairs(orderid,amount,dateOrder)
}
```

Single Entity Type – Multiple Databases

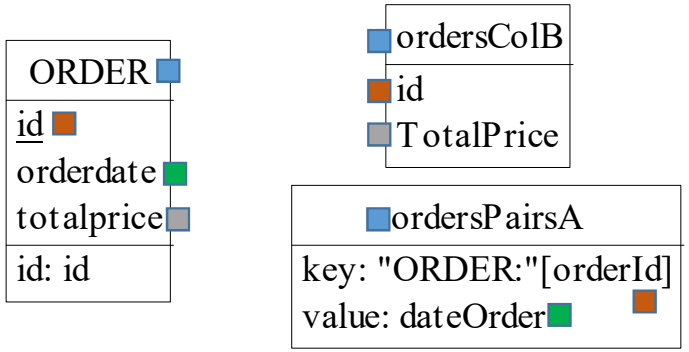
Conceptual Schema



Physical Schema



Mappings




```
conceptual schema CS1PSaRel {

    entity type Order {
        id : string,
        orderdate : date,
        totalprice : float
        identifier {
            id
        }
    }

}
```

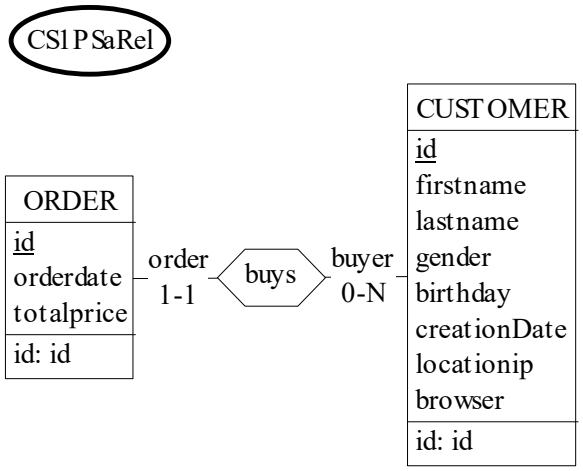
```
physical schemas {

    key value schema kv : redisPerfTest {
        kvpairs ordersPairsA {
            key : "ORDER:"[orderid],
            value : dateOrder
        }
    }

    document schema mongoSchema: mongoPerfTest {

        collection ordersColB {
            fields {
                id,
                TotalPrice
            }
        }

    }
}
```

```
databases {
        mongodb mongoPerfTest{
        host:"mongoPerfTest"
        port : 27017
    }
    redis redisPerfTest {
        host : "redisPerfTest"
        port : 6379
    }
```

```
mapping rules {
    CS1PSbKeyDoc.Order(id,totalprice)
    -> mongoSchema.ordersColB(id,TotalPrice),
    CS1PSbKeyDoc.Order(id, orderdate)
    -> kv.ordersPairsA(orderid,dateOrder)
}
```

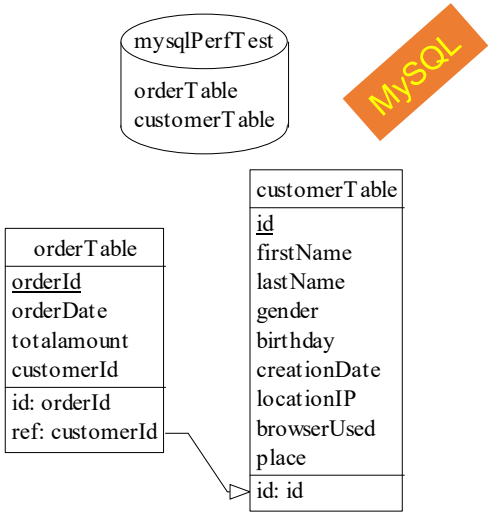
One To Many Relationship - Single Database

Conceptual Schema



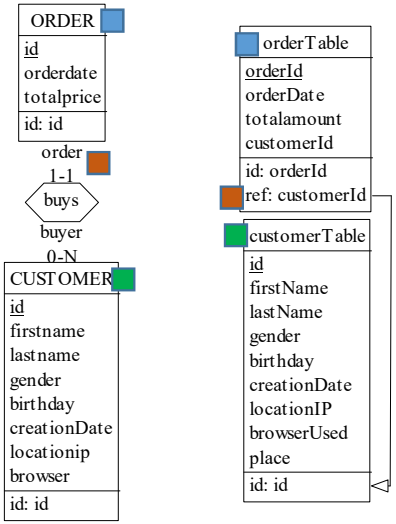

```
conceptual schema CS1PSaRel {
    entity type Customer {
        id : string,
        firstname : string,
        lastname : string,
        gender : string,
        birthday : date,
        creationDate : date,
        locationip : string,
        browser : string
        identifier{
            id
        }
    }

    entity type Order {
        id : string,
        orderdate : date,
        totalprice : float
        identifier {
            id
        }
    }


    relationship type buys {
        order[1]: Order,
        client[0-N]: Customer
    }
}
```

Physical Schema




```
physical schemas {
    relational schema relSchema : mysqlPerfTest {
        table orderTable {
            columns {
                orderId,
                orderDate,
                totalamount,
                customerId
            }
            references {
                clientRef: customerId->customerTable.id
            }
        }

        table customerTable {
            columns {
                id,
                firstName,
                lastName,
                gender,
                birthday,
                creationDate,
                locationIP,
                browserUsed,
                place
            }
        }
    }
}
```

Mappings



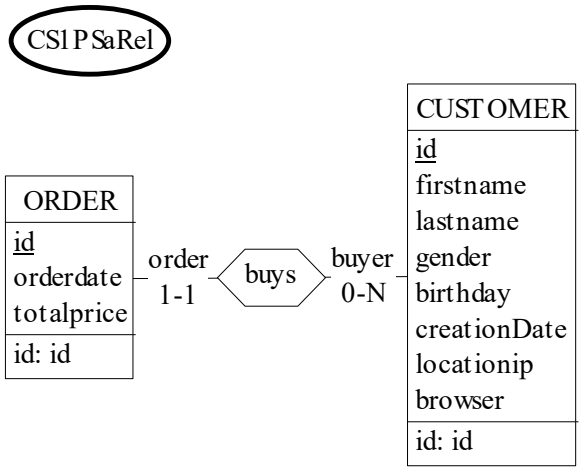

```
mapping rules {
    CS2PSaRel.Order(id, orderdate, totalprice)
        -> relSchema.orderTable( orderId, orderDate, totalamount),

    CS2PSaRel.Customer(id,firstname,lastname, gender,
                       birthday, creationDate, locationip, browser)
        -> relSchema.customerTable(id, firstName, lastName, gender,
                                   birthday, creationDate, locationIP, browserUsed),

    CS2PSaRel.buys.order -> relSchema.orderTable.clientRef
}
```

Nom du rôle

Nom de la référence

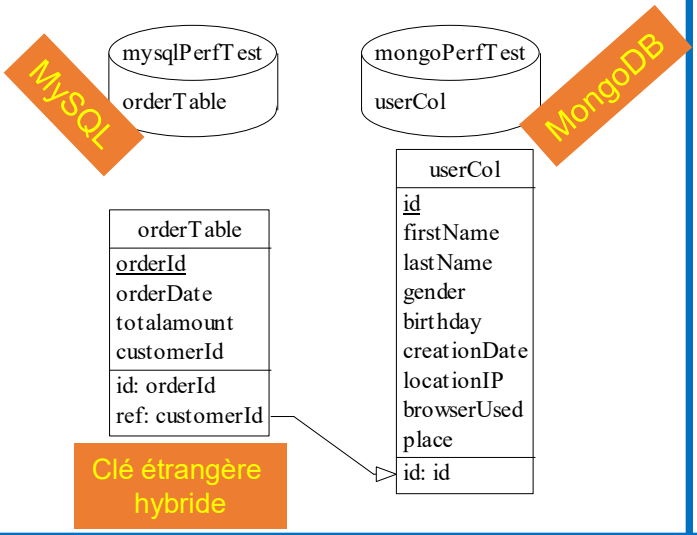
One To Many Relationship - Multiple Databases

Conceptual Schema



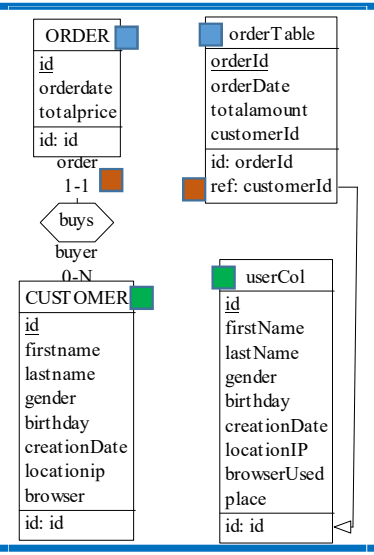

```
conceptual schema CS1PSaRel {
    entity type Customer {
        id : string,
        firstname : string,
        lastname : string,
        gender : string,
        birthday : date,
        creationDate : date,
        locationip : string,
        browser : string
        identifier{
            id
        }
    }

    entity type Order {
        id : string,
        orderdate : date,
        totalprice : float
        identifier {
            id
        }
    }

    relationship type buys {
        order[1]: Order,
        client[0-N]: Customer
    }
}
```

Physical Schema




```
physical schemas {
    relational schema relSchema : mysqlPerfTest {
        table orderTable {
            columns {
                orderId,
                orderDate,
                totalamount,
                customerId
            }
            references {
                clientRef: customerId->mongoSchema.userCol.id
            }
        }
    }

    document schema mongoSchema: mongoPerfTest {
        collection userCol{
            fields {
                id,
                firstName,
                lastName,
                gender,
                birthday,
                creationDate,
                locationIP,
                browserUsed,
                place,
                orders[0-N]{
                    id,
                    buydate,
                    totalamount
                }
            }
        }
    }
}
```

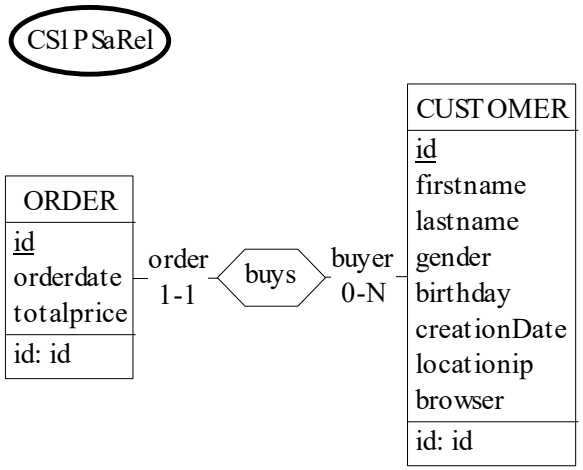
Mappings




```
mapping rules {
    CS2PSbRelDoc.Order(id, orderdate, totalprice)
        -> relSchema.orderTable( orderId, orderDate, totalamount),

    CS2PSbRelDoc.Customer(id,firstname,lastname, gender,
                          birthday, creationDate, locationip, browser)
        -> mongoSchema.userCol(id, firstName, lastName, gender, birthday,
                               creationDate, locationIP, browserUsed),

    CS2PSbRelDoc.buys.order -> relSchema.orderTable.clientRef
}
```

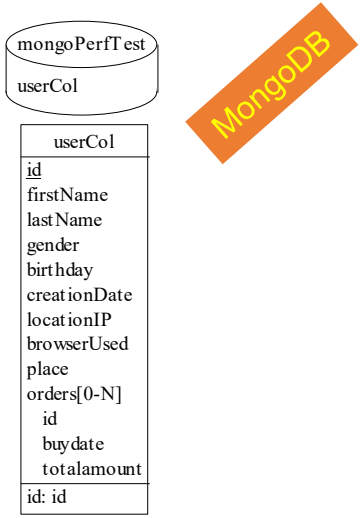
One To Many Relationship
- Nested Entities

Conceptual Schema



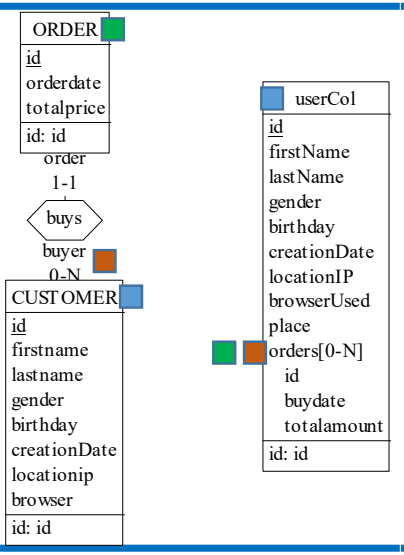

```
conceptual schema CS1PSaRel {
    entity type Customer {
        id : string,
        firstname : string,
        lastname : string,
        gender : string,
        birthday : date,
        creationDate : date,
        locationip : string,
        browser : string
        identifier{
            id
        }
    }

    entity type Order {
        id : string,
        orderdate : date,
        totalprice : float
        identifier {
            id
        }
    }

    relationship type buys {
        order[1]: Order,
        client[0-N]: Customer
    }
}
```

Physical Schema




```
physical schemas {
    document schema mongoSchema: mongoPerfTest {

        collection userCol{
            fields {
                id,
                firstName,
                lastName,
                gender,
                birthday,
                creationDate,
                locationIP,
                browserUsed,
                place,
                orders[0-N]{
                    id,
                    buydate,
                    totalamount
                }
            }
        }

    }
}
```

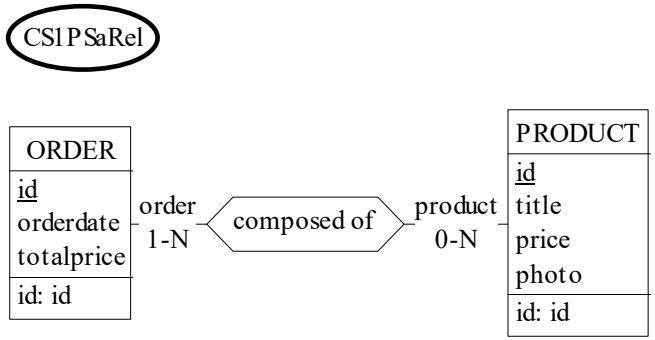
Mappings




```
mapping rules {
    CS2PSdDoc.Order(id,orderdate,totalprice)
        -> mongoSchema.userCol.orders(id,buydate,totalamount),

    CS2PSdDoc.Customer(id,firstname,lastname, gender, birthday,
                       creationDate, locationip, browser)
        -> mongoSchema.userCol(id, firstName, lastName, gender, birthday,
                               creationDate, locationIP, browserUsed),

    CS2PSdDoc.buys.client -> mongoSchema.userCol.orders()
}
```

Many To Many Relationship

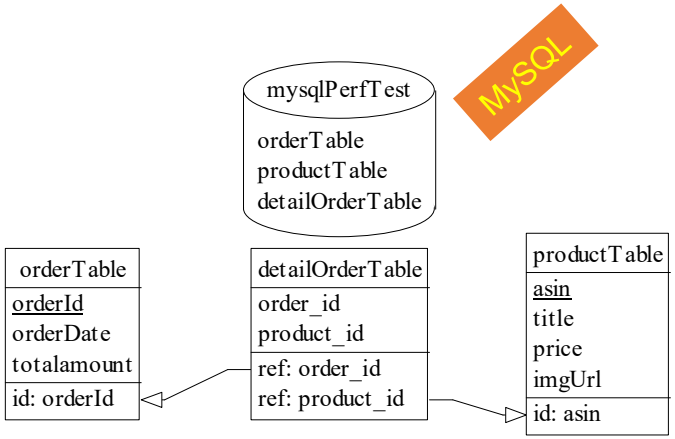
Conceptual Schema




```
conceptual schema CS1PSaRel {

    entity type Product {
        id : string,
        title : string,
        price : float,
        photo : string
        identifier{
            id
        }
    }
```

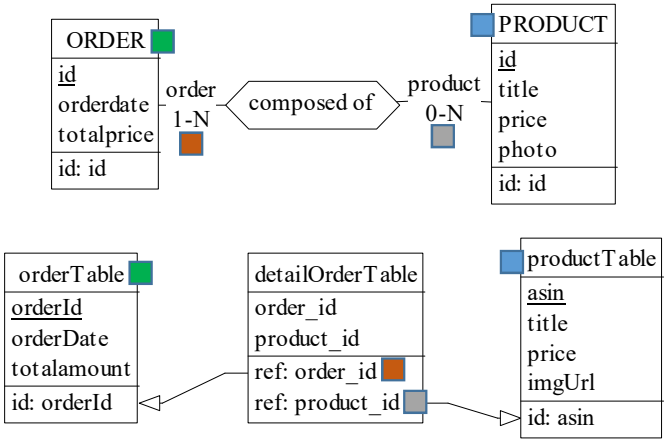
```
entity type Order {
    id : string,
    orderdate : date,
    totalprice : float
    identifier {
        id
    }
}

relationship type composed_of {
    orderP[1-N] : Order,
    orderedProducts[0-N] : Product
}
```

Physical Schema



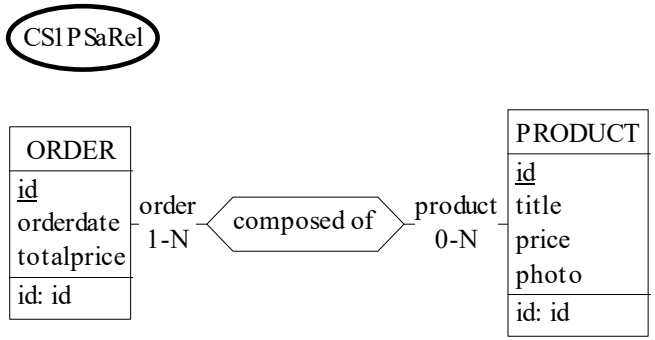

```
physical schemas {
    relational schema relSchema : mysqlPerfTest {
        table orderTable {
            columns {
                orderId,
                orderDate,
                totalamount,
                customerId
            }
        }

        table productTable{
            columns{
                asin,
                title,
                price,
                imgUrl
            }
        }
        table detailOrderTable{
                columns{
                    order_id,
                    product_id
                }

                references {
                    orderRef : order_id -> relSchema.orderTable.orderId
                    productRef : product_id -> productTable.asin
                }
            }
        }
    }
```

Mappings




```
mapping rules
{
    CS3PSaRel.Product(id, title, price,photo)
        -> relSchema.productTable(asin, title, price, imgUrl),

    CS3PSaRel.Order(id, orderdate, totalprice)
        -> relSchema.orderTable(orderId, orderDate, totalamount),

    CS3PSaRel.composed_of.orderP
        -> relSchema.detailOrderTable.orderRef,

    CS3PSaRel.composed_of.orderedProducts
        -> relSchema.detailOrderTable.productRef
}
```

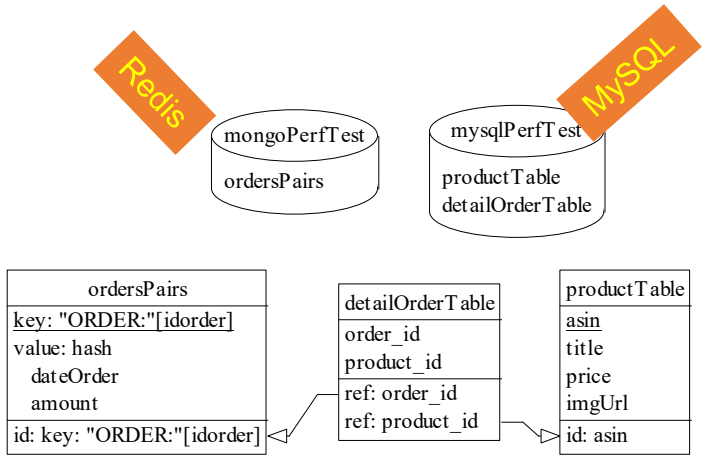
Conceptual Schema




```
conceptual schema CS1PSaRel {

    entity type Product {
        id : string,
        title : string,
        price : float,
        photo : string
        identifier{
            id
        }
    }
```

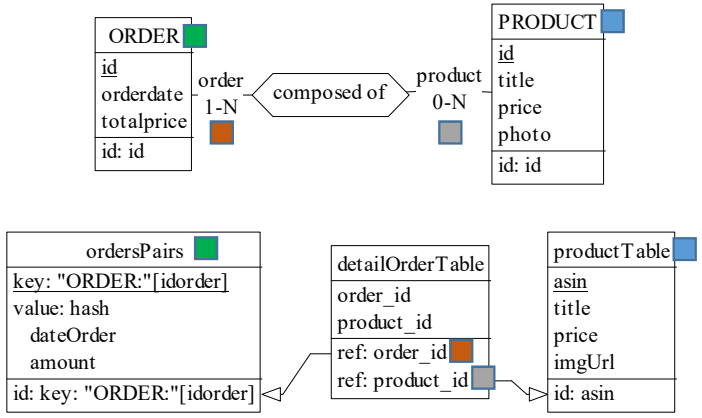
```
    entity type Order {
        id : string,
        orderdate : date,
        totalprice : float
        identifier {
            id
        }
    }

    relationship type composed_of {
        orderP[1-N] : Order,
        orderedProducts[0-N] : Product
    }
```

Physical Schema



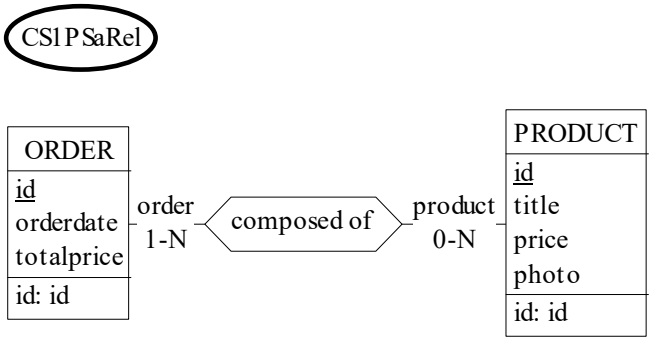

```
physical schemas {

    key value schema kv: redisPerfTest {

        kvpairs ordersPairs {
            key : "ORDER:"[idorder],
            value : hash {
                dateOrder,
                amount
            }
        }

    }
    relational schema relSchema : mysqlPerfTest {

        table productTable{
            columns{
                asin,
                title,
                price,
                imgUrl
            }
        }

    table detailOrderTable{
            columns{
                order_id,
                product_id
            }

            references {
                orderRef : order_id -> kv.ordersPairs.idorder
                productRef : product_id -> productTable.asin
            }
        }
    }
```

Mappings

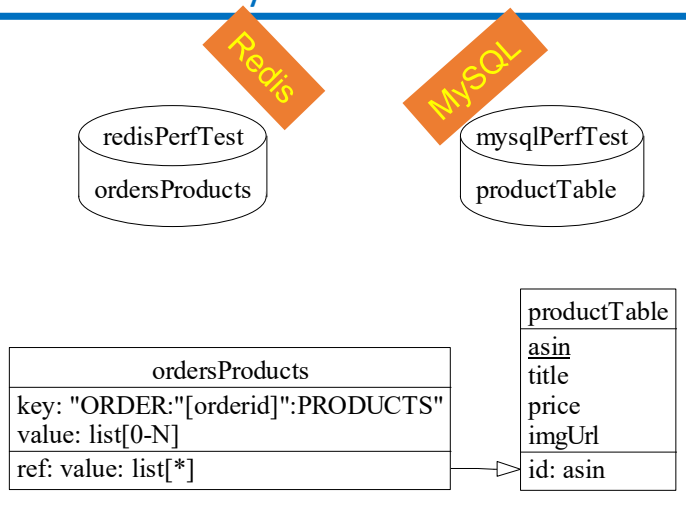



```
mapping rules
{
    CS3PSbRelKey.Order(id,totalprice, orderdate)
      -> kv.ordersPairs(idorder,amount,dateOrder),

    CS3PSbRelKey.Product(id, title, price,photo)
      -> relSchema.productTable(asin, title, price, imgUrl),

    CS3PSbRelKey.composed_of.orderP
      -> relSchema.detailOrderTable.orderRef,

    CS3PSbRelKey.composed_of.orderedProducts
      -> relSchema.detailOrderTable.productRef
}
```

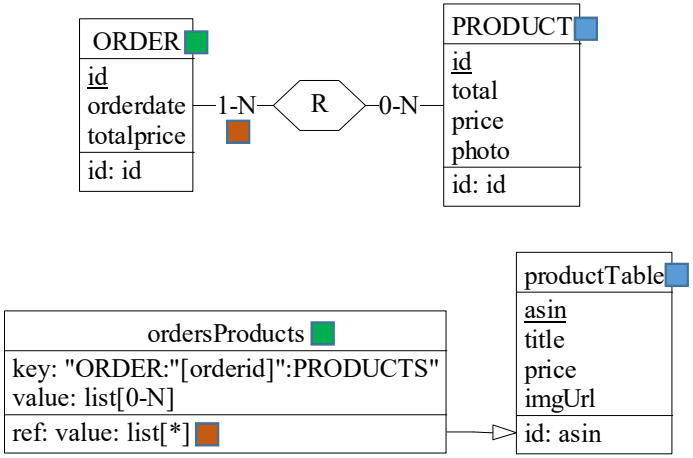
Conceptual Schema



Physical Schema



Mappings



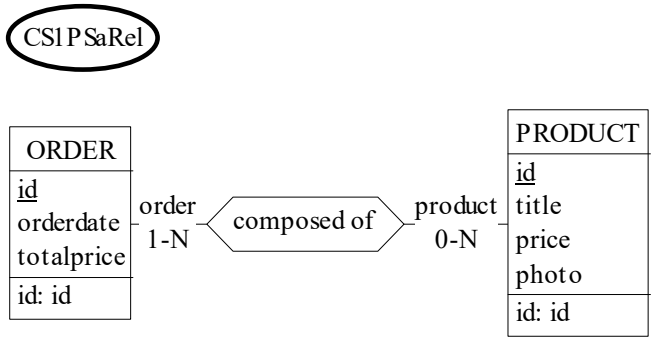

```
conceptual schema CS1PSaRel {

    entity type Product {
        id : string,
        title : string,
        price : float,
        photo : string
        identifier{
            id
        }
    }
```

```
    entity type Order {
        id : string,
        orderdate : date,
        totalprice : float
        identifier {
            id
        }
    }

    relationship type composed_of {
        orderP[1-N] : Order,
        orderedProducts[0-N] : Product
    }
```

```
key value schema kv: redisPerfTest {

    kvpairs ordersProducts {
        key : "ORDER:"[orderid]":PRODUCTS",
        value : list {
            productid
        }
        references {
            bought : productid -> relSchema.productTable.asin
        }
    }


relational schema relSchema : mysqlPerfTest {

    table productTable{
        columns{
            asin,
            title,
            price,
            imgUrl
        }
    }
```

```
mapping rules
{

  CS3PSbRelKey.Product(id, title, price,photo)
    -> relSchema.productTable(asin, title, price, imgUrl),
CS3PSaKey.Order(id) -> kv.ordersProducts(orderid),
CS3PSaKey.composed_of.orderP -> kv.ordersProducts.bought
```

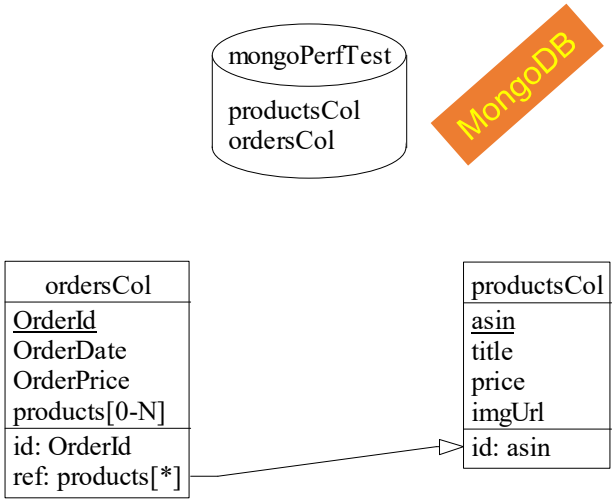
Conceptual Schema




```
conceptual schema CS1PSaRel {

    entity type Product {
        id : string,
        title : string,
        price : float,
        photo : string
        identifier{
            id
        }
    }
```

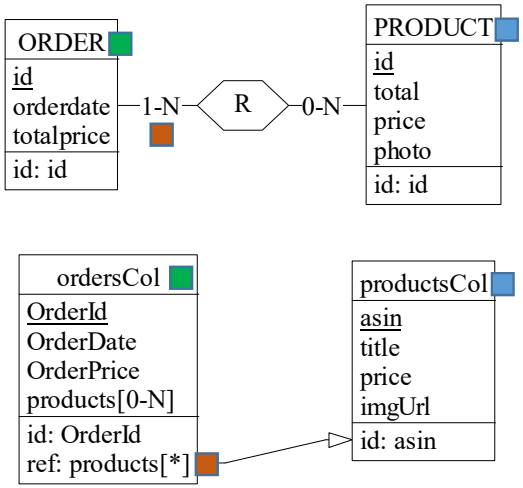
```
entity type Order {
    id : string,
    orderdate : date,
    totalprice : float
    identifier {
        id
    }
}

relationship type composed_of {
    orderP[1-N] : Order,
    orderedProducts[0-N] : Product
}
```

Physical Schema




```
physical schemas {
    document schema mongoSchema: mongoPerfTest {

        collection ordersCol {
            fields {
                OrderId,
                OrderDate,
                TotalPrice,
                products[] : ProductRef
            }

            references {
                prodOrder: ProductRef -> productsCol.asin
            }
        }

        collection productsCol {
            fields {
                asin,
                title,
                price,
                imgUrl
            }
        }
```

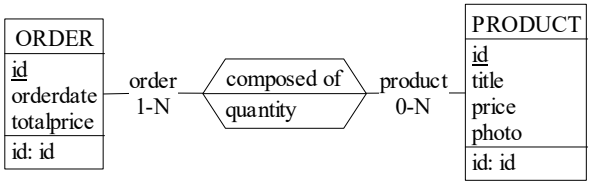
Mappings




```
mapping rules
{
    CS3PSaDoc.Order(id,orderdate,totalprice)
      -> mongoSchema.ordersCol(
            OrderId,OrderDate,TotalPrice),

    CS3PSaDoc.Product(id,title,price,photo)
      -> mongoSchema.productsCol(
            asin,title,price,imgUrl),

    CS3PSaDoc.composed_of.orderP
      -> mongoSchema.ordersCol.prodOrder
}
```

Relationship with attributes

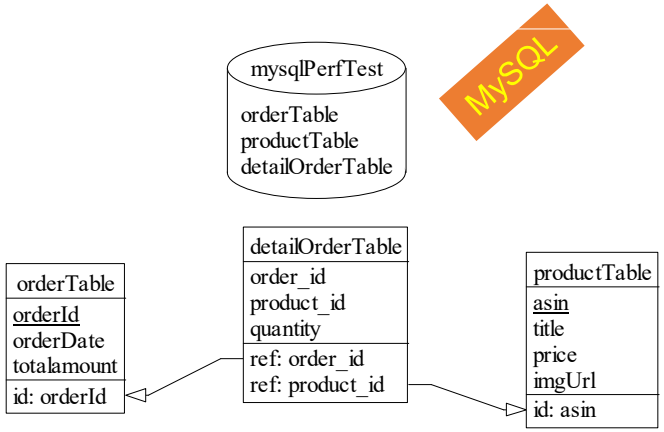
Conceptual Schema




```
conceptual schema CS1PSaRel {

    entity type Product {
        id : string,
        title : string,
        price : float,
        photo : string
        identifier{
            id
        }
    }
```

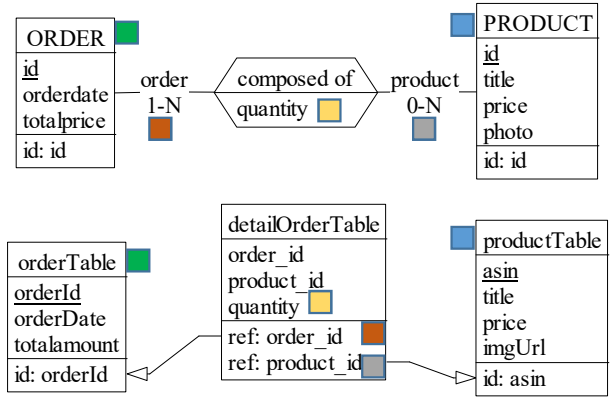
```
entity type Order {
    id : string,
    orderdate : date,
    totalprice : float
    identifier {
        id
    }
}

  relationship type composed_of {
      orderP[1-N] : Order,
      orderedProducts[0-N] : Product,
      quantity : int
  }
```

Physical Schema




```
physical schemas {
    relational schema relSchema : mysqlPerfTest {
        table orderTable {
            columns {
                orderId,
                orderDate,
                totalamount
            }
        }

        table productTable{
            columns{
                asin,
                title,
                price,
                imgUrl
            }
        }

    table detailOrderTable{
            columns{
                order_id,
                product_id,
                quantity
            }

            references {
                orderRef : order_id -> relSchema.orderTable.orderId
                productRef : product_id -> productTable.asin
            }
        }
    }
}
```

Mappings




```
mapping rules
{
    CS3PSaRel.Product(id, title, price,photo)
        -> relSchema.productTable(asin, title, price, imgUrl),
    CS3PSaRel.Order(id, orderdate, totalprice)
        -> relSchema.orderTable(orderId, orderDate, totalamount),
    CS3PSaRel.composed_of.orderP
        -> relSchema.detailOrderTable.orderRef,
    CS3PSaRel.composed_of.orderedProducts
        -> relSchema.detailOrderTable.productRef,
    rel : CS3PSaRel.composed_of(quantity)
        -> relSchema.detailOrderTable(quantity)
}
```