

Agenda

- Modeling: Relational vs OO
- Mapping OO-Relational
- JPA Annotations
- Basic mapping (Tables, Columns, Relations)
- Inheritance
- Queries
- References

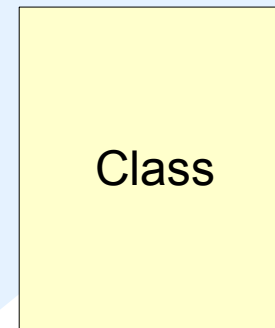
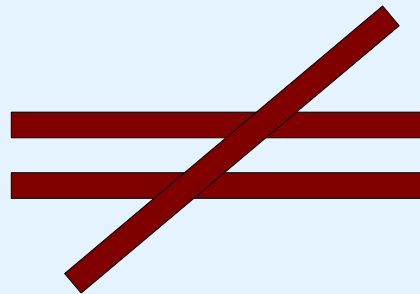
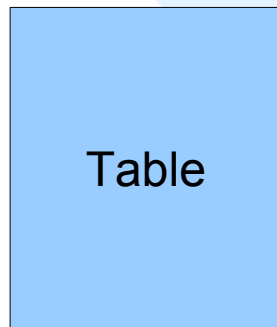
Modeling: Relational vs OO

Relational

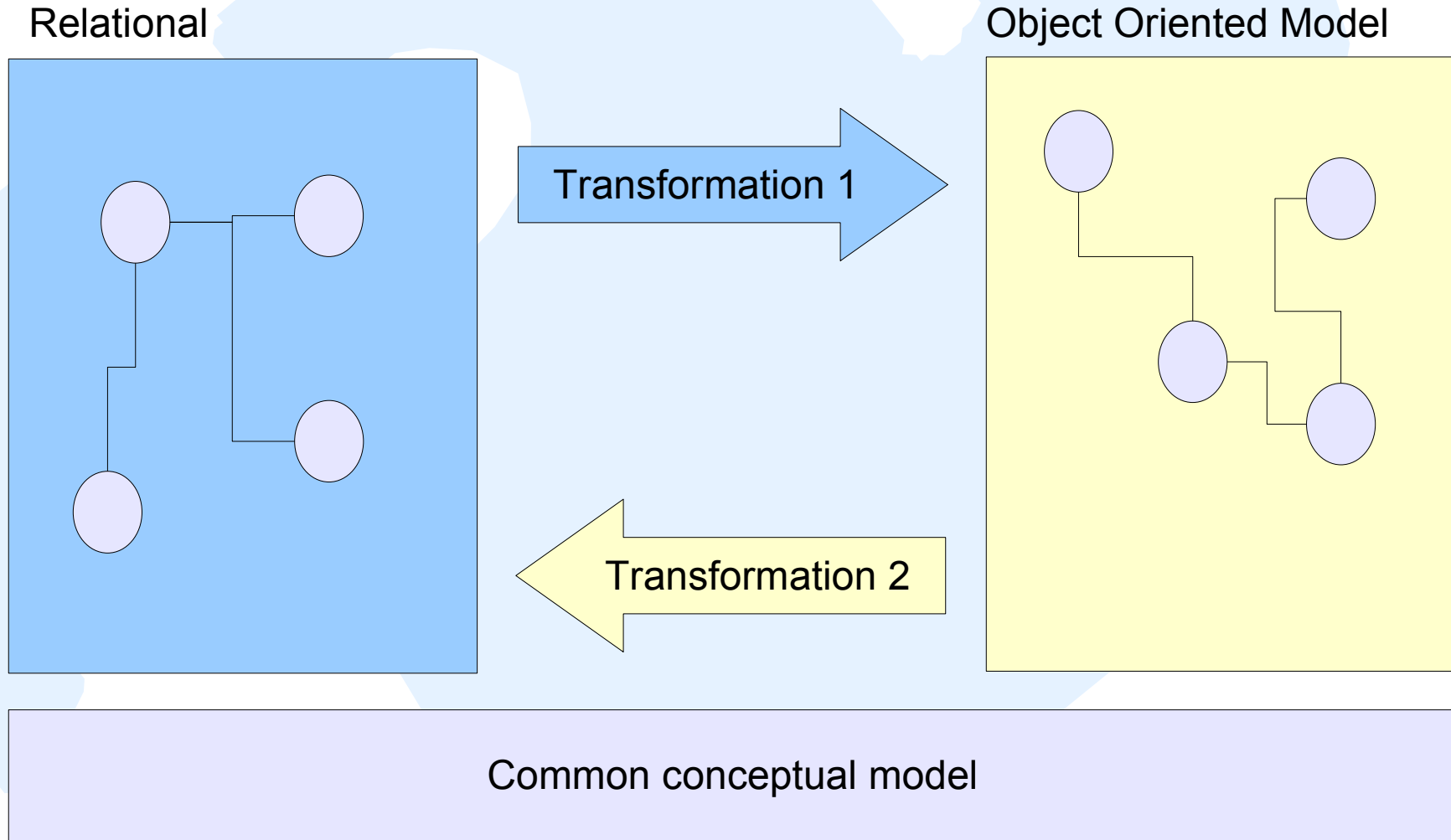
- ✓ Only data can be modeled.
- ✓ Rows, Columns and Tables.
- ✓ Simple relations.
- ✓ Query language: SQL
- ✓ Focused on persistence.
- ✓ Optimized for storage and lookup.

Object Oriented

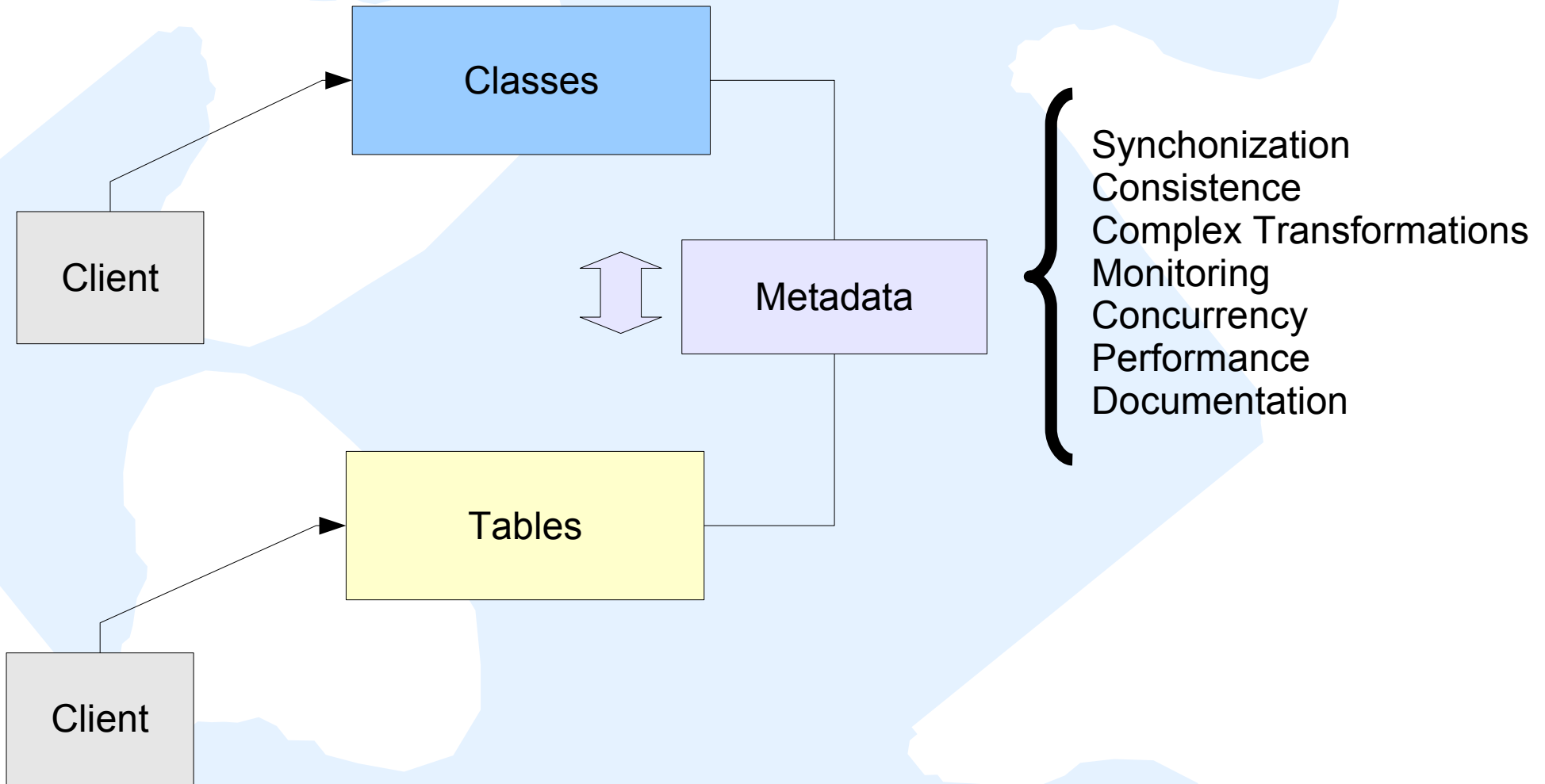
- ✓ Data and behavior can be modeled.
- ✓ Concepts, attributes, relations
- ✓ Complex relations (ej: Inheritance)
- ✓
- ✓ Focused on the problem's semantic.
- ✓ Optimized for complex model expression.



Modeling: Relational vs OO



Mapping: Object-Relational (ORM)

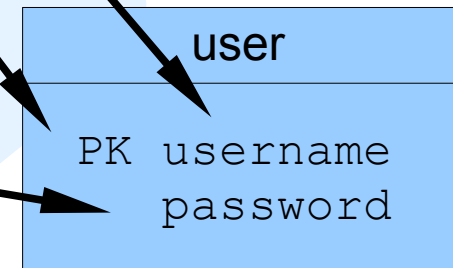


JPA Annotations

- Provide information needed to do all transformations between both models.
- Centralizes the information of both models.
- Based on a standard. Provides model portability.
- Simplifies the configuration management.
- Take advantage of the implicit information. Reduces the need to provide obvious information.

Table and Column mappings

```
@Entity  
@Table(name="user")  
public class User implements Serializable {  
  
    @Id  
    @Column(length=20)  
    private String username;  
  
    @Column(length=30)  
    private String password;  
  
    // Getters y Setters .....  
}
```



Relations mapping

One to many relations

```
@OneToMany  
private Set<Address> addresses;
```

Many to one relations

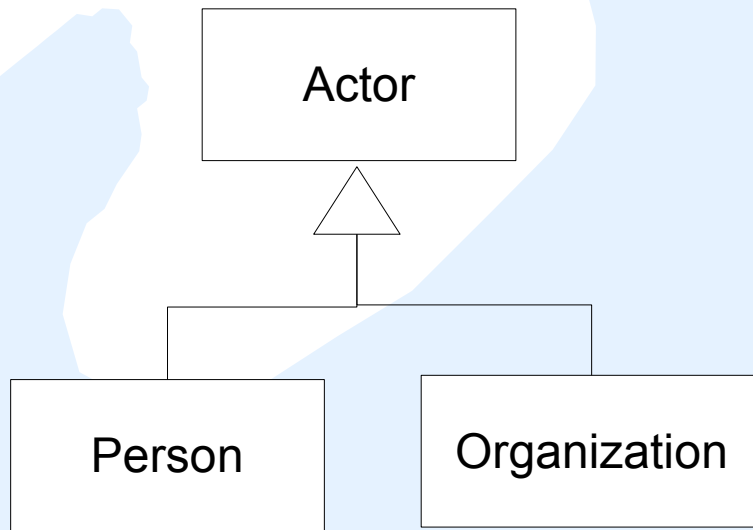
```
@ManyToOne  
private Person person;
```

Many to many relations

```
@ManyToMany()  
private Set<Role> roles;
```

Java Persistence API (Very basic intro)

Inheritance



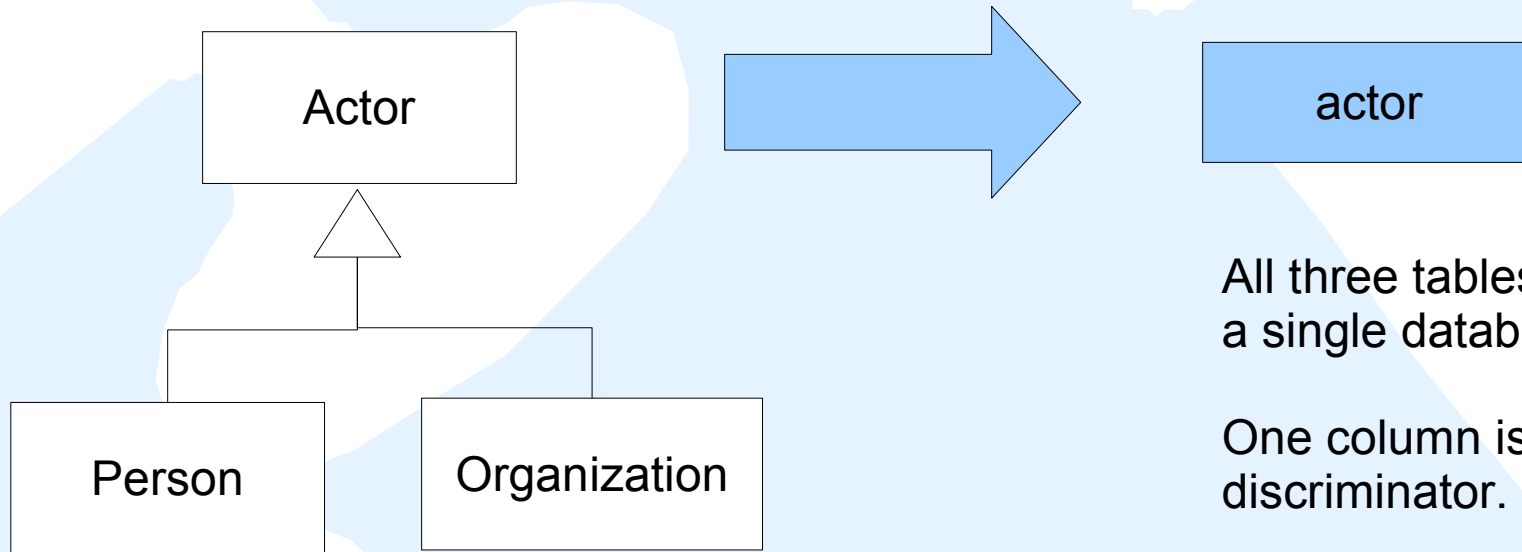
3 Mapping Strategies

Single Table

Joined

Table per class

Inheritance (Single table based)

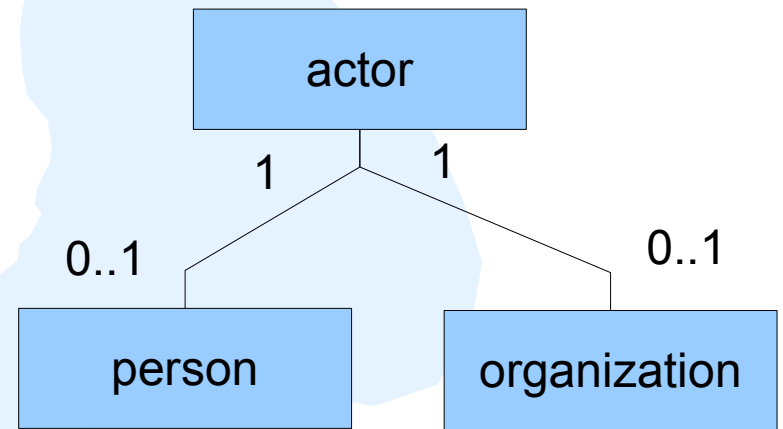
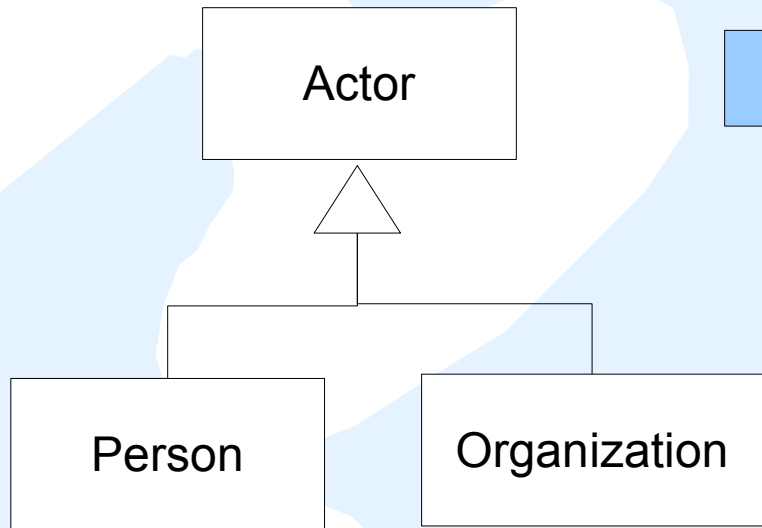


All three tables are mapped to a single database table.

One column is defined as a class discriminator.

The table contains fields for all attributes in all three classes.

Inheritance (Joined)



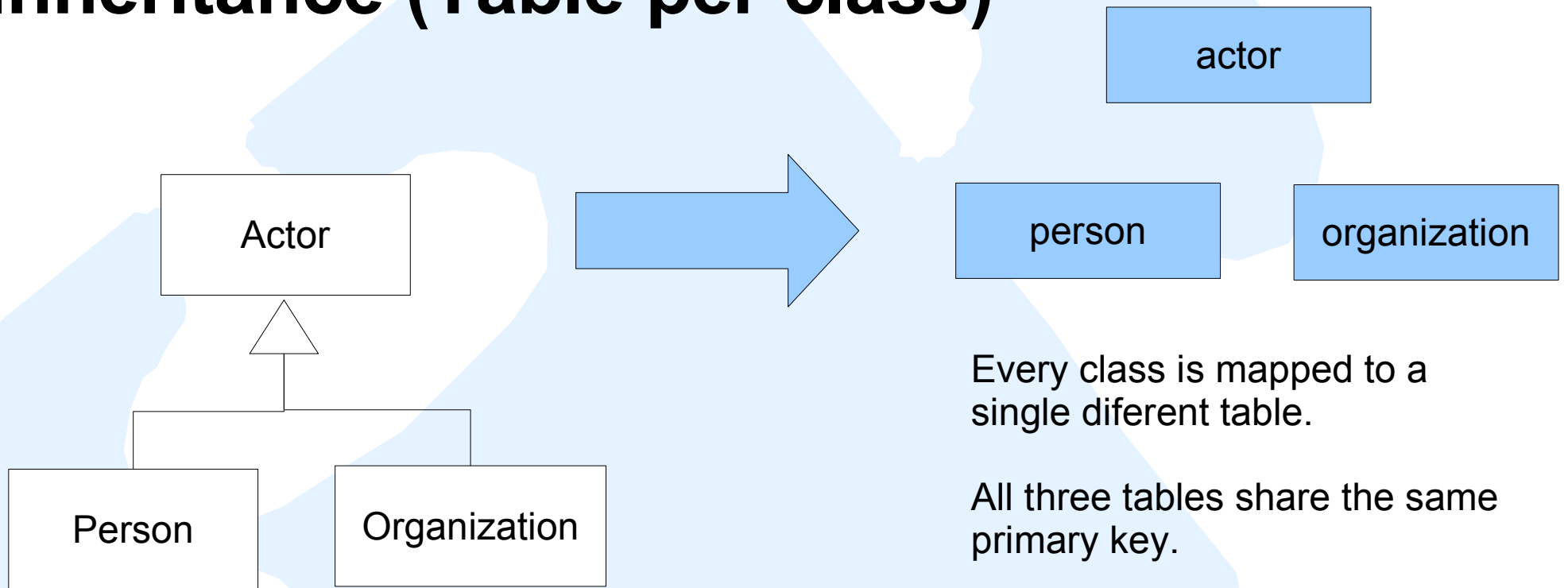
The Actor superclass is mapped to a single table with all common attributes of all tree classes.

Person and Organization are mapped to separated tables with not inherited attributes as fields.

All tree tables share the same primary key.

All tree tables are associated as shown above.

Inheritance (Table per class)



Every class is mapped to a single different table.

All three tables share the same primary key.

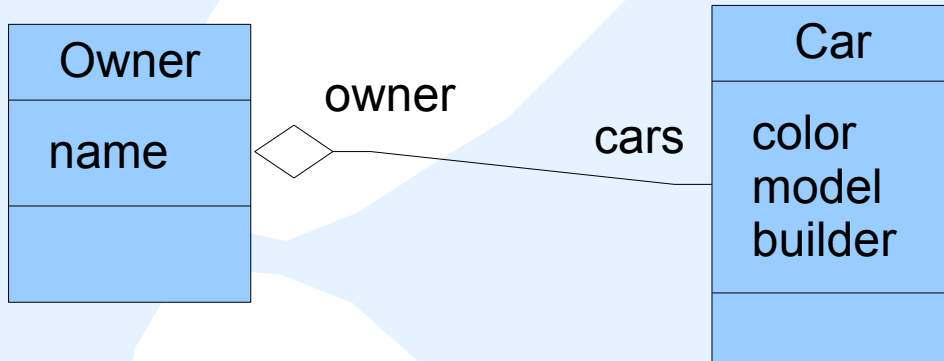
Common attributes are not shared. Each table contains all attributes of its class.

Java Persistence API (Very basic intro)

Queries

JPQL : Java Persistence Query Language

With this language you can query the model using OO Model based propositions, then these will be transformed to SQL.



```
SELECT c.owner
FROM Car c
WHERE c.color = 'Red'
```

```
SELECT c
FROM Car c
WHERE c.model > 1976
```

These queries returns lists of the OO Model objects.

References

- <http://java.sun.com/javaee/overview/faq/persistence.jsp>
- <http://java.sun.com/developer/technicalArticles/J2EE/jpa/>
- <http://www.hibernate.org/>
- <http://openjpa.apache.org/>
- <http://www.jpox.org/>
- <http://www.oracle.com/technology/products/ias/toplink/jpa/index.html>
- <http://jcp.org/en/jsr/detail?id=220>