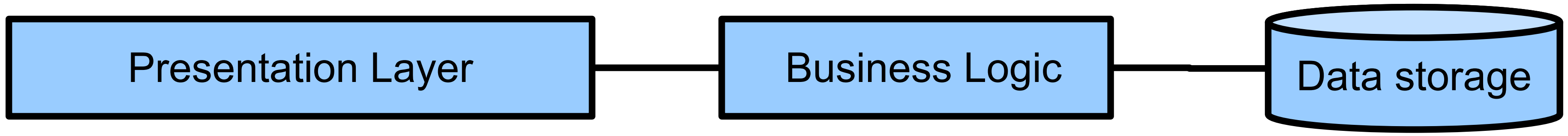
ORM and JPA 2.0

Zdeněk Kouba, Petr Křemen

What is Object-relational mapping ?

- a typical information system architecture:

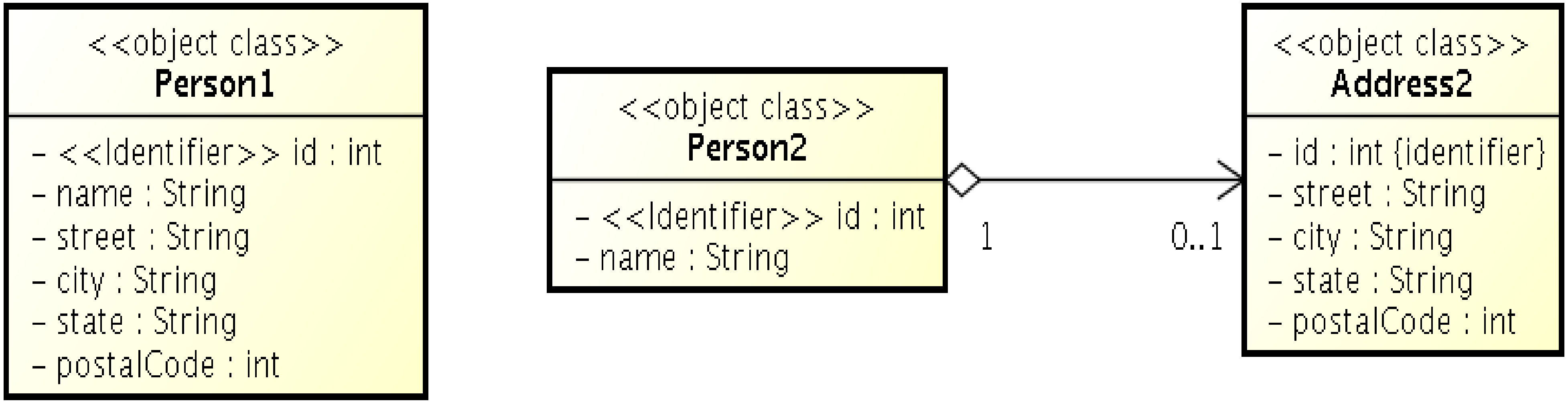


- How to avoid data format transformations when interchanging data from the (OO-based) presentation layer to the data storage (RDBMS) and back ?

- How to ensure persistence in the (OO-based) business logic ?

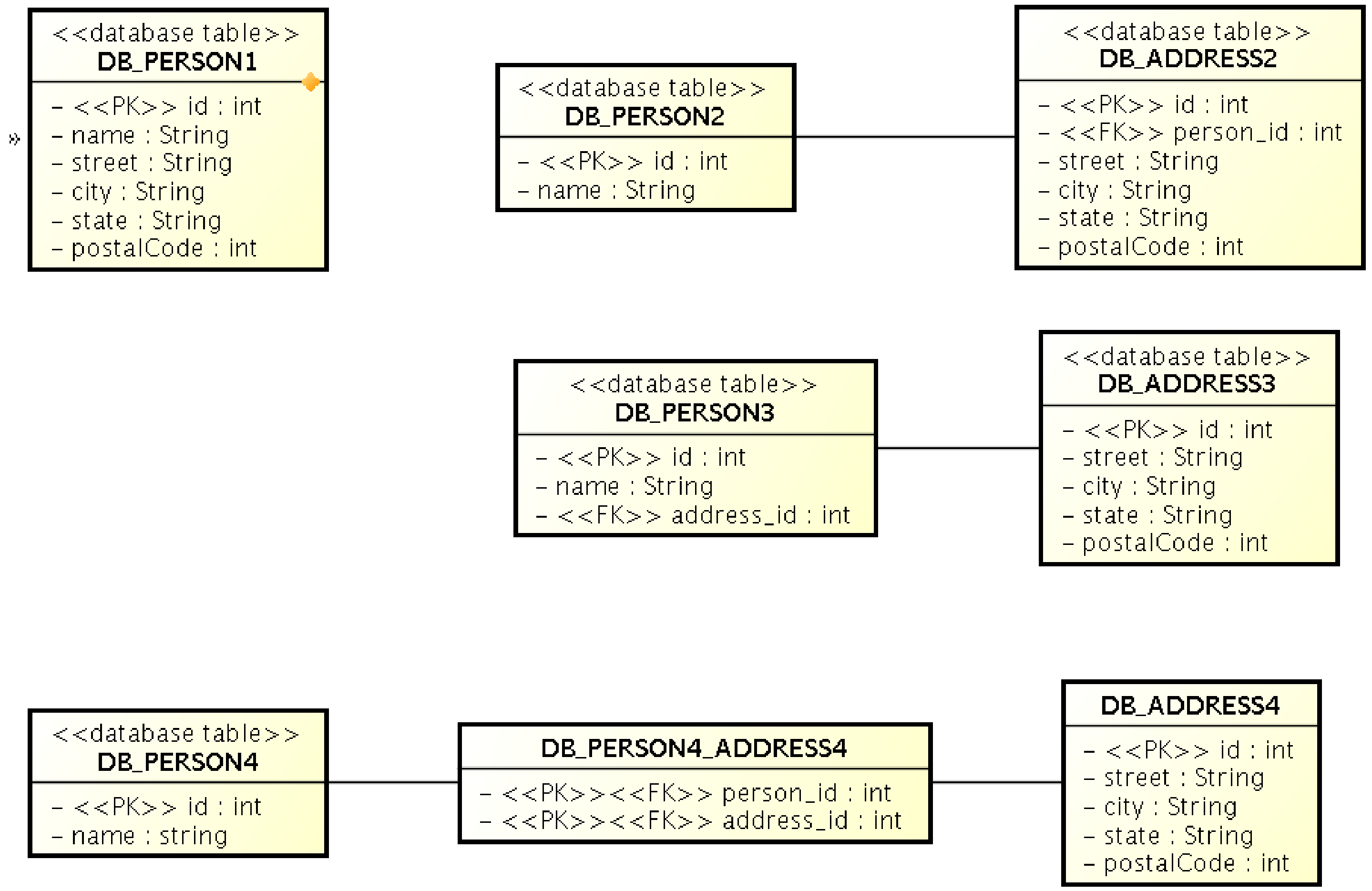
Example – object model

- When would You stick to one of these options ?



Example – database

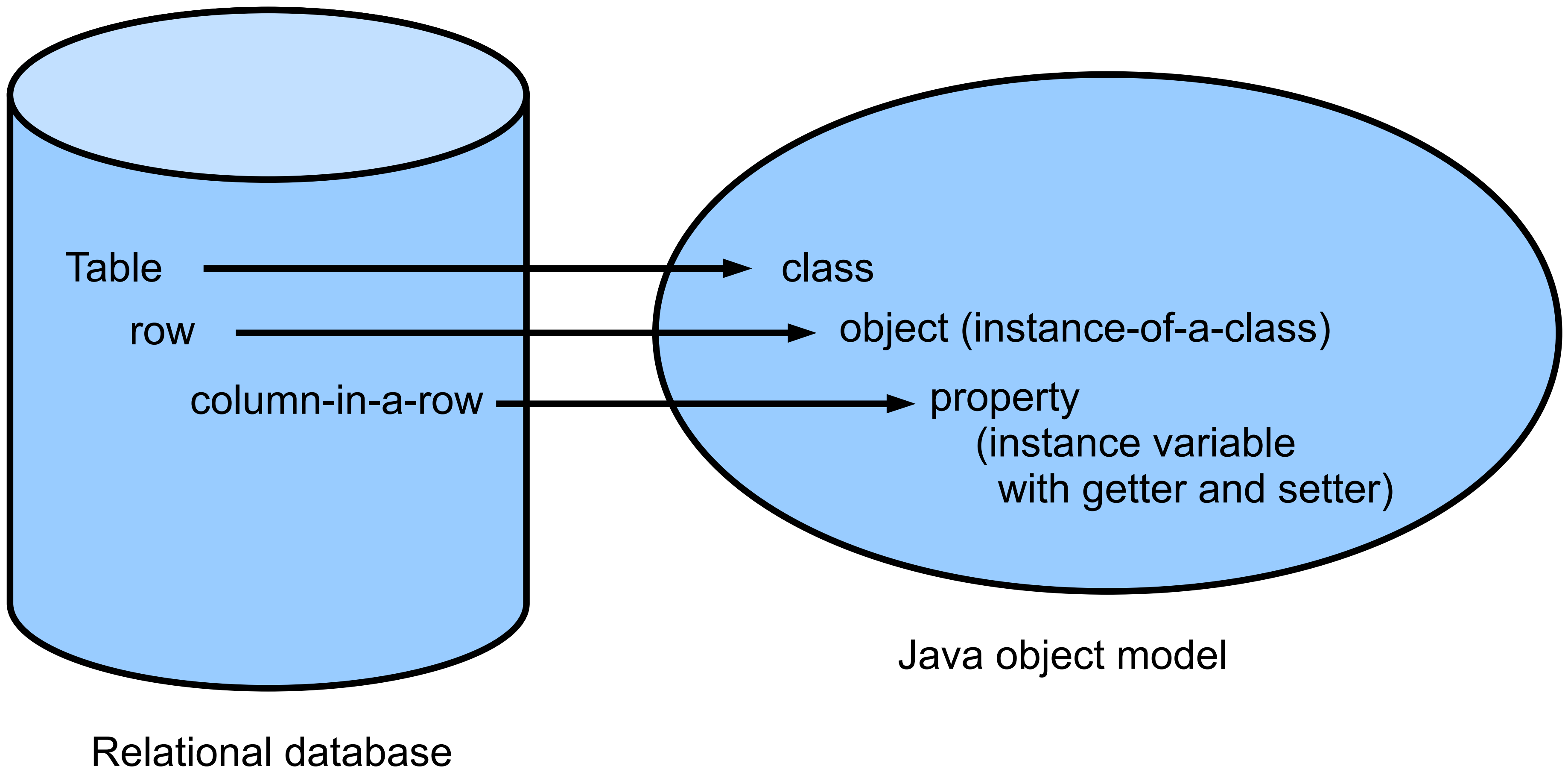
- … and how to model it in SQL ?



Object-relational mapping

- Mapping between the database (declarative) schema and the data structures in the object-oriented language.
- Let's take a look at JPA 2.0

Object-relational mapping



JPA 2.0

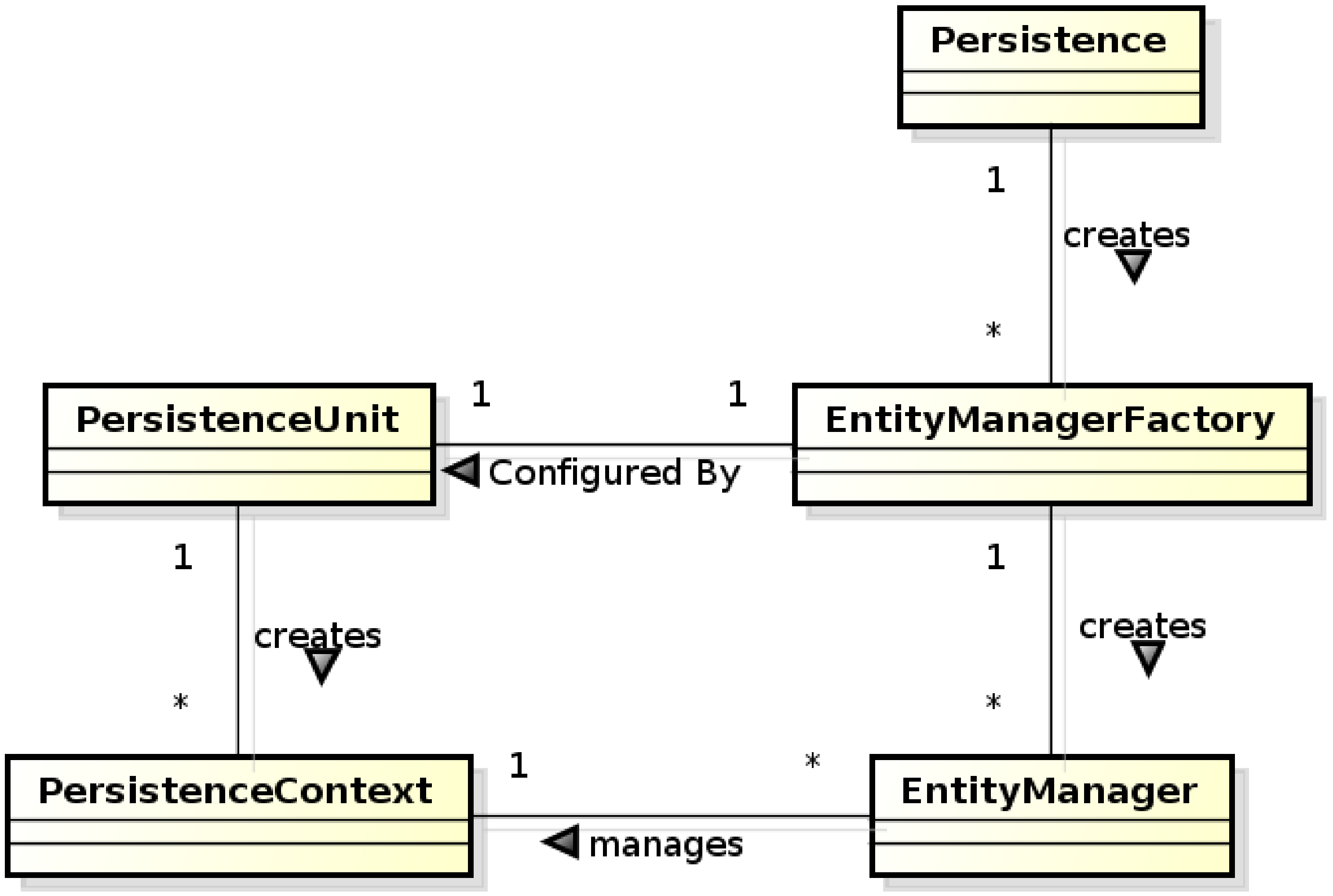
- Java Persistence API 2.0 (JSR-317)
- Although part of Java EE 6 specifications, JPA 2.0 can be used both in EE and SE applications.
- Main topics covered:
  - Basic scenarios
  - Controller logic – EntityManager interface
  - ORM strategies
  - JPQL + Criteria API

JPA 2.0 – Entity Example

- Minimal example (configuration by exception):

```
@Entity
public class Person {
   @Id
   @GeneratedValue
   private Integer id;
   private String name;
      // setters + getters
}
```

JPA2.0 – Basic concepts

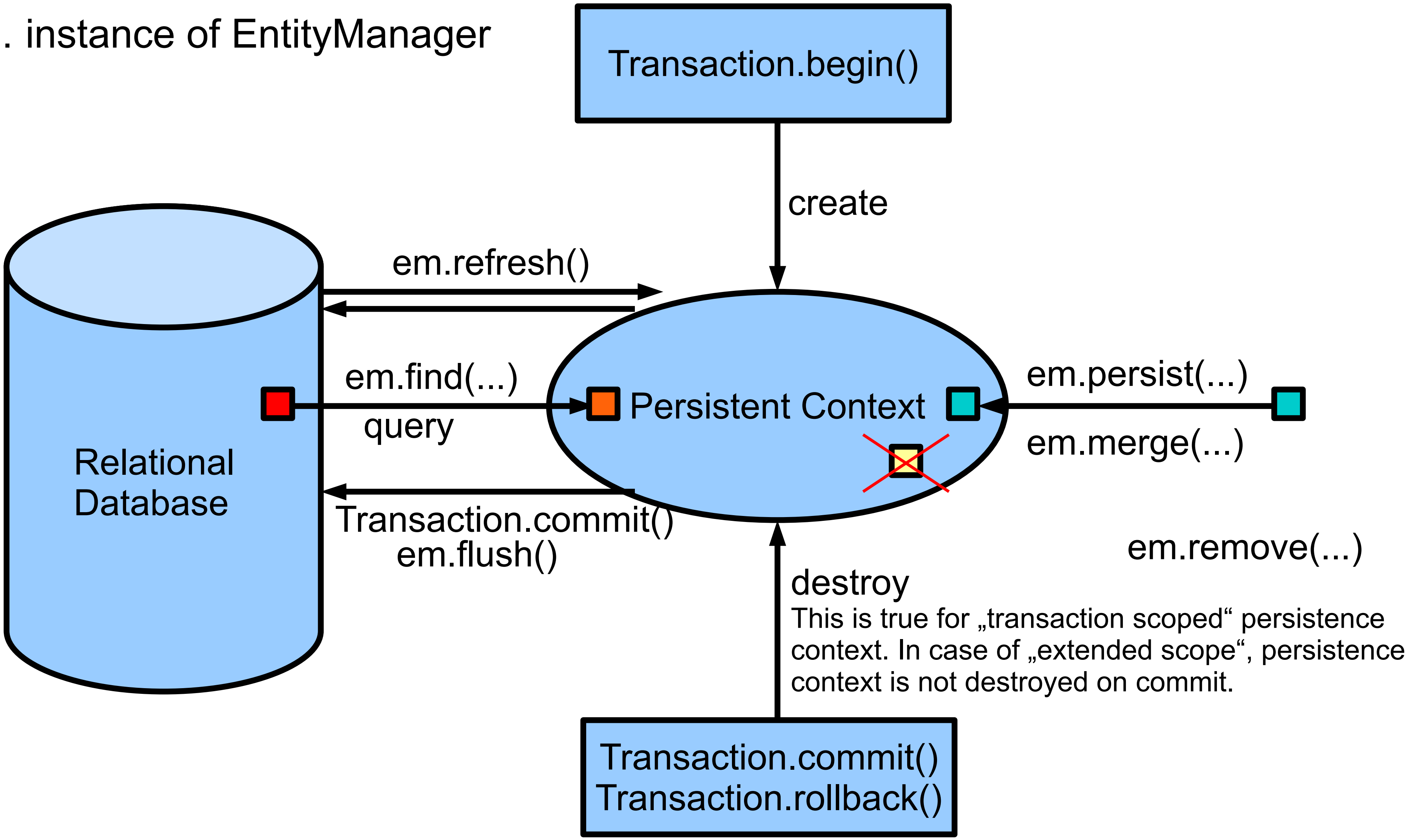


JPA 2.0 - Basics

- Let's have a set of „suitably annotated“ POJOs, called ***entities***, describing your domain model.
- A set of entities is logically grouped into a ***persistence unit***.
- JPA 2.0 providers :
  - generate persistence unit from existing database,
  - generate database schema from existing persistence unit.
  - TopLink (Oracle) … JPA
  - EclipseLink (Eclipse) … JPA 2.0
- What is the benefit of the keeping Your domain model in the persistence unit entities (OO) instead of the database schema (SQL)

JPA 2.0 – Persistence Context

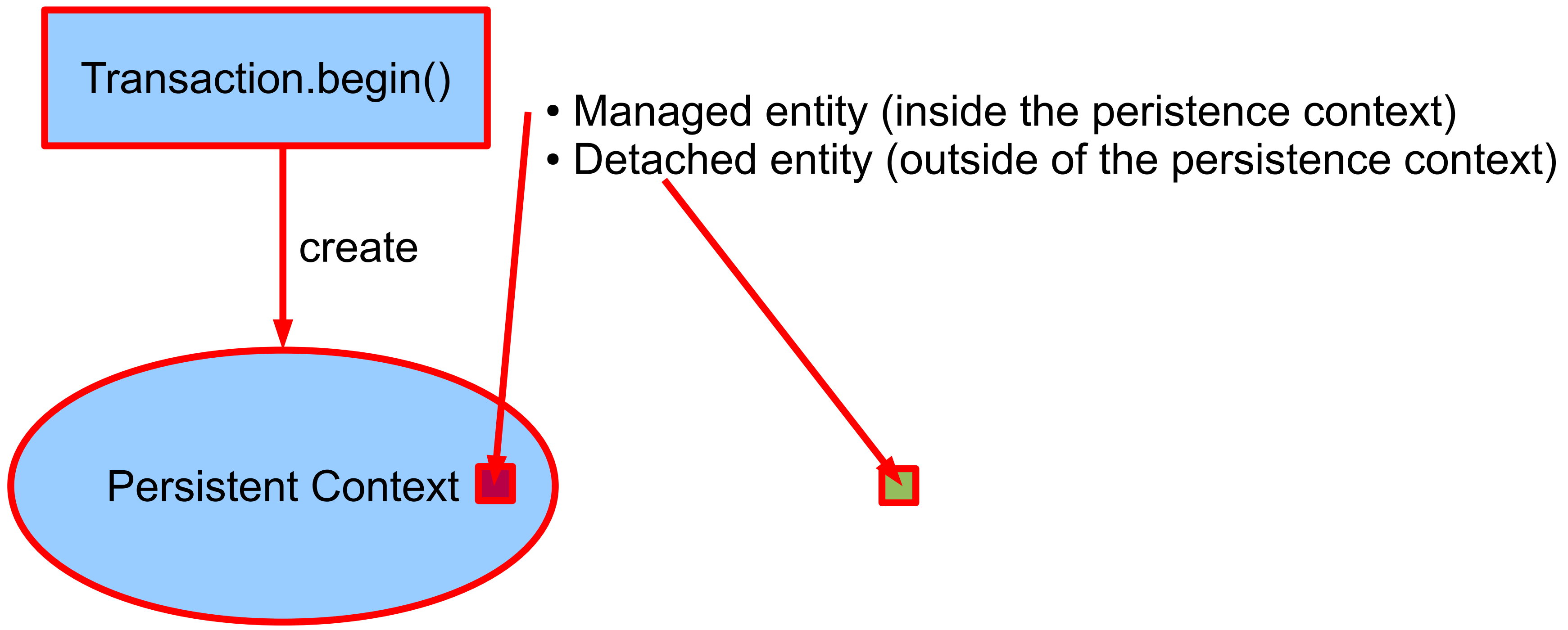
em … instance of EntityManager



This is true for „transaction scoped“ persistence context. In case of „extended scope“, persistence context is not destroyed on commit.

JPA 2.0 – Persistence Context

em … instance of EntityManager



• Managed entity (inside the peristence context)
• Detached entity (outside of the persistence context)

• em.persist(entity) … persistence context must not contain an entity with the same id
• em.merge(entity) … merging the state of an entity existing inside the persistence context and its other incarnation outside

JPA 2.0 – Persistence Context

- In runtime, the application accesses the object counterpart (represented by entity instances) of the database data. These (*managed*) entities comprise a ***persistence context (PC)***.
  - PC is synchronized with the database on demand (refresh, flush) or at transaction commit.
  - PC is accessed by an EntityManager instance and can be shared by several EntityManager instances.

JPA 2.0 – EntityManager

- **`EntityManager` (EM)** instance is in fact a generic DAO, while entities can be understand as DPO (managed) or DTO (detached).
- Selected operations on EM (CRUD) :
  - ***C***reate : em.persist(Object o)
  - ***R***ead : em.find(Object id), em.refresh(Object o)
  - ***U***pdate : em.merge(Object o)
  - ***D***elete : em.remove(Object o)
  - native/JPQL queries: createNativeQuery, createQuery, etc.
  - Resource-local transactions: getTransaction().[begin(),commit(),rollback()]