
# Aktuelle Technologien zur Entwicklung verteilter Java-Anwendungen

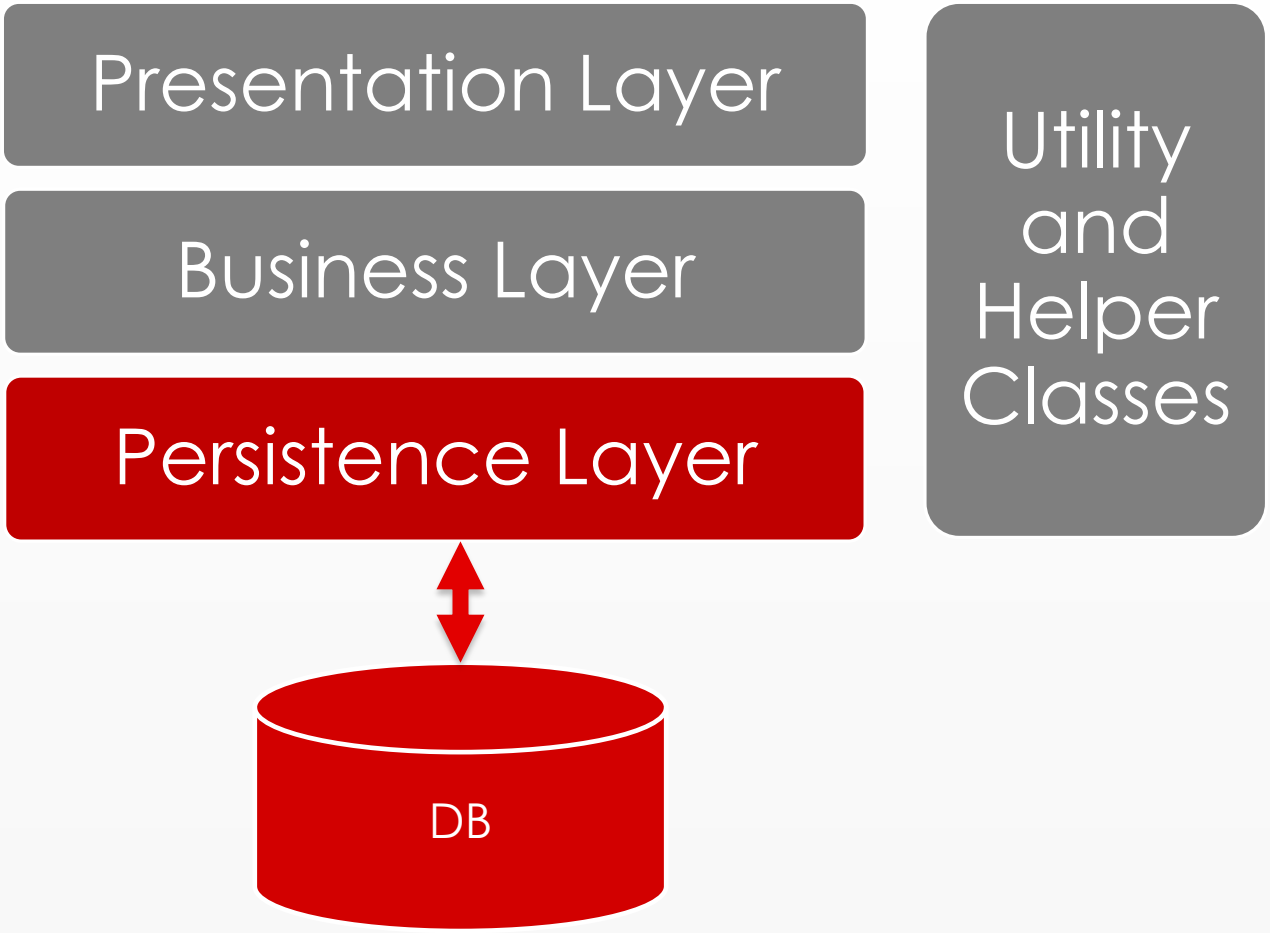
## Die Integration von Datenbanken mittels JPA 2

Maximilian Spelsberg

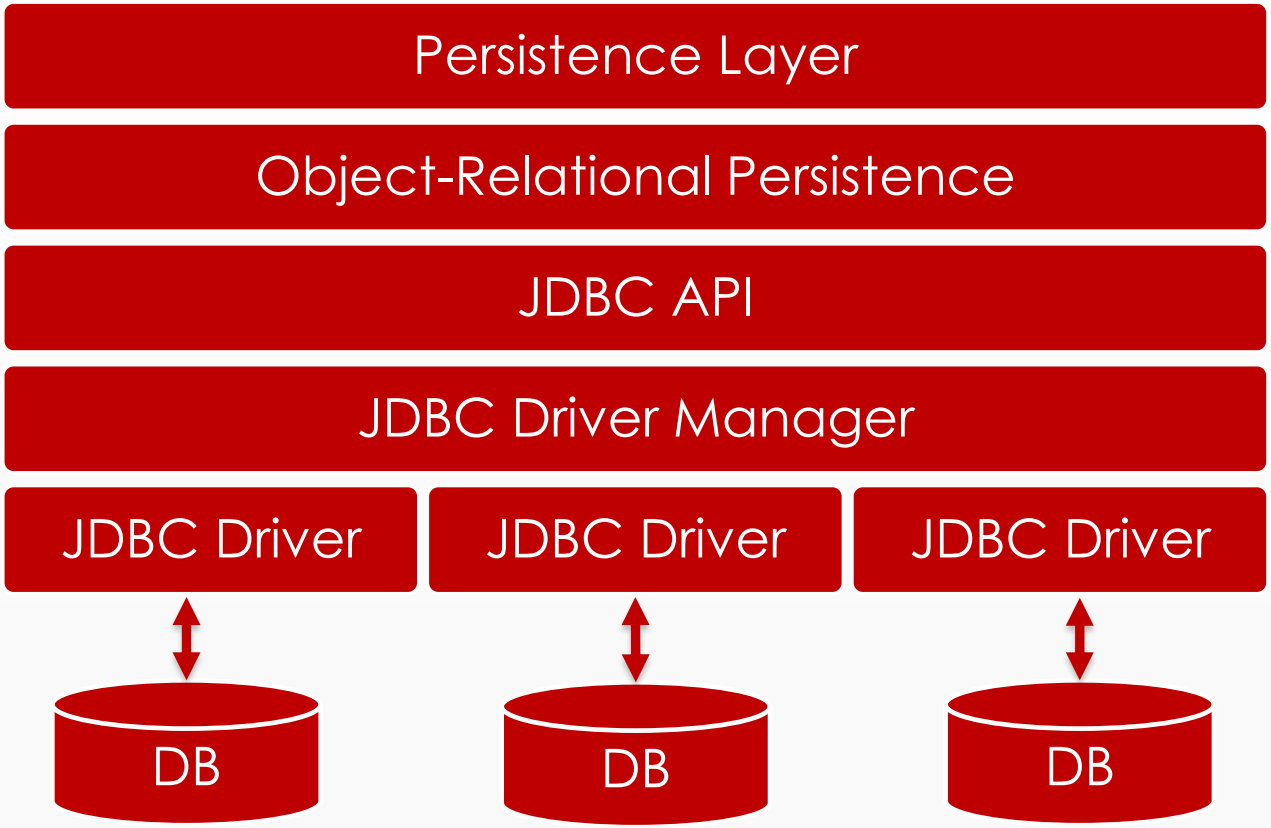
# Agenda

- Grundlagen: Objektrelationale Abbildung
- ORM mittels JPA
- JPA 2.1
- Code-Beispiel

# Drei-Schichten-Architektur

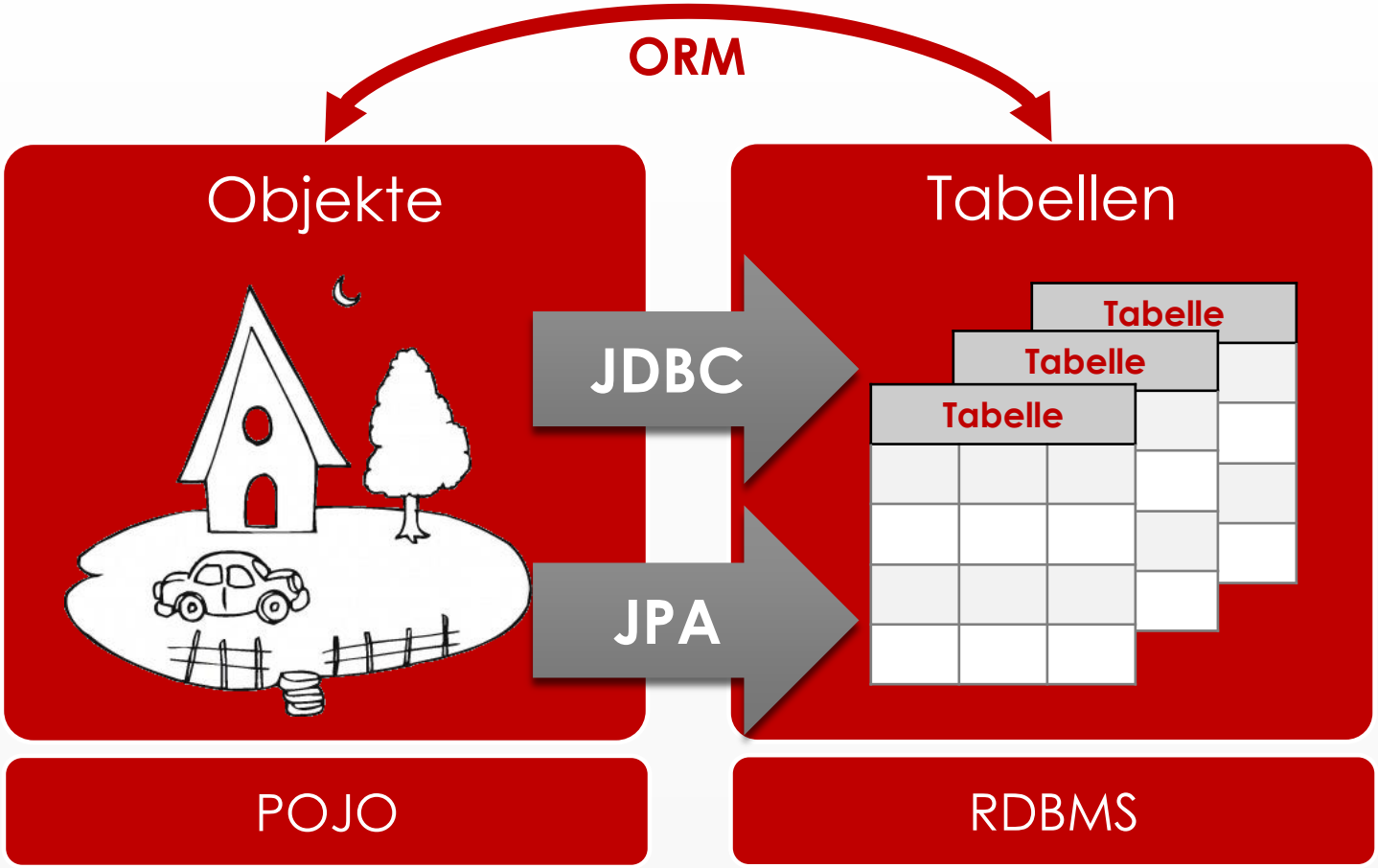


# Der Persistence Layer




Quelle: http://www.developersbook.com/jdbc/interview-questions/jdbc-interview-questions-faqs.php

# Objektrelationales Abbilden



# Agenda

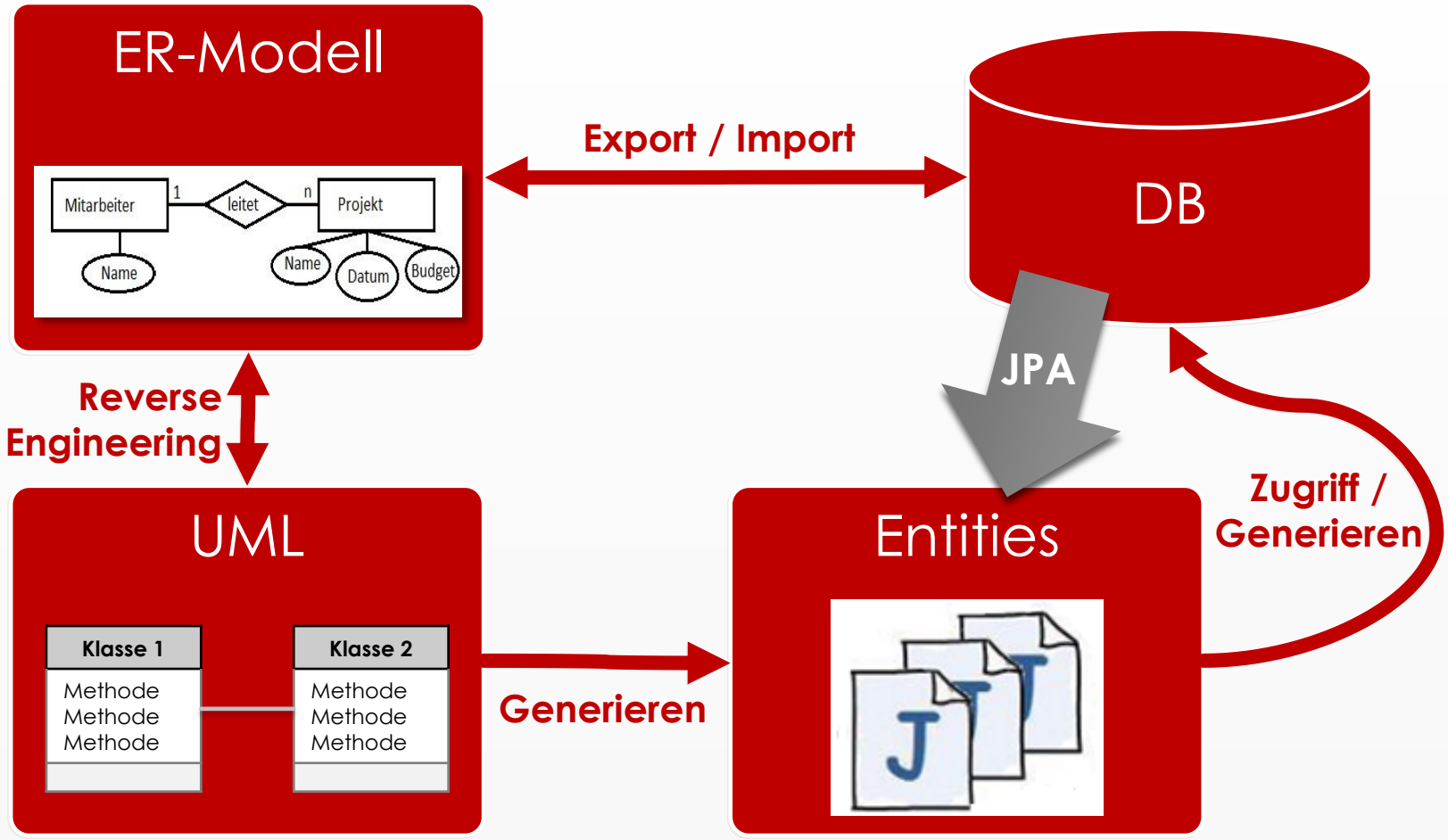
- Grundlagen: Objektrelationale Abbildung
- ORM mittels JPA
- JPA 2.1
- Code-Beispiel

# Java Persistence API

- JPA ist die Spezifikation


- Implementierung erfolgt über Frameworks

# ORM mittels der Java Persistence API



# Entitäten

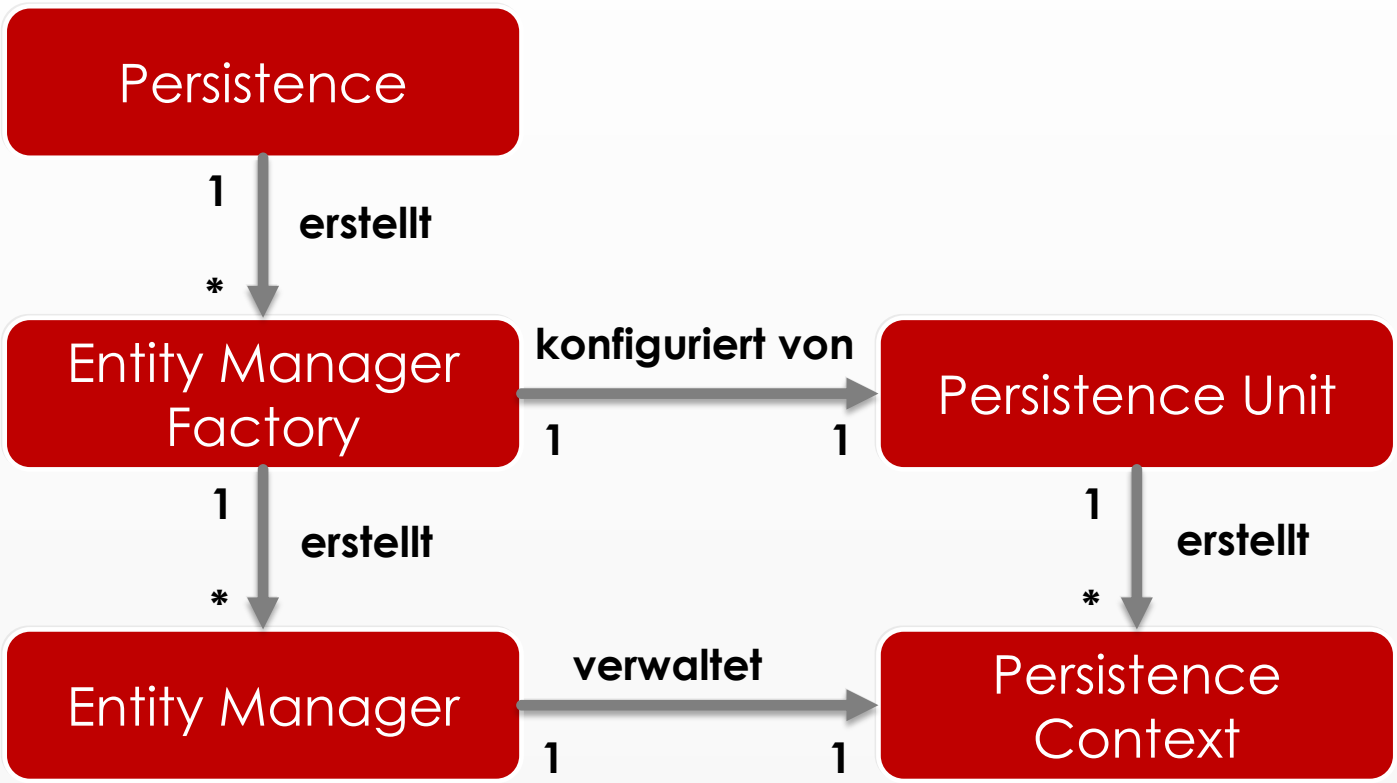
## POJO

```
public class Klasse {

        public Klasse() {
        }

        private String variable;

        public String getVariable() {
                return variable;
        }

        public void setVariable(String v) {
                this.variable = v;
        }

}
```

## Entity

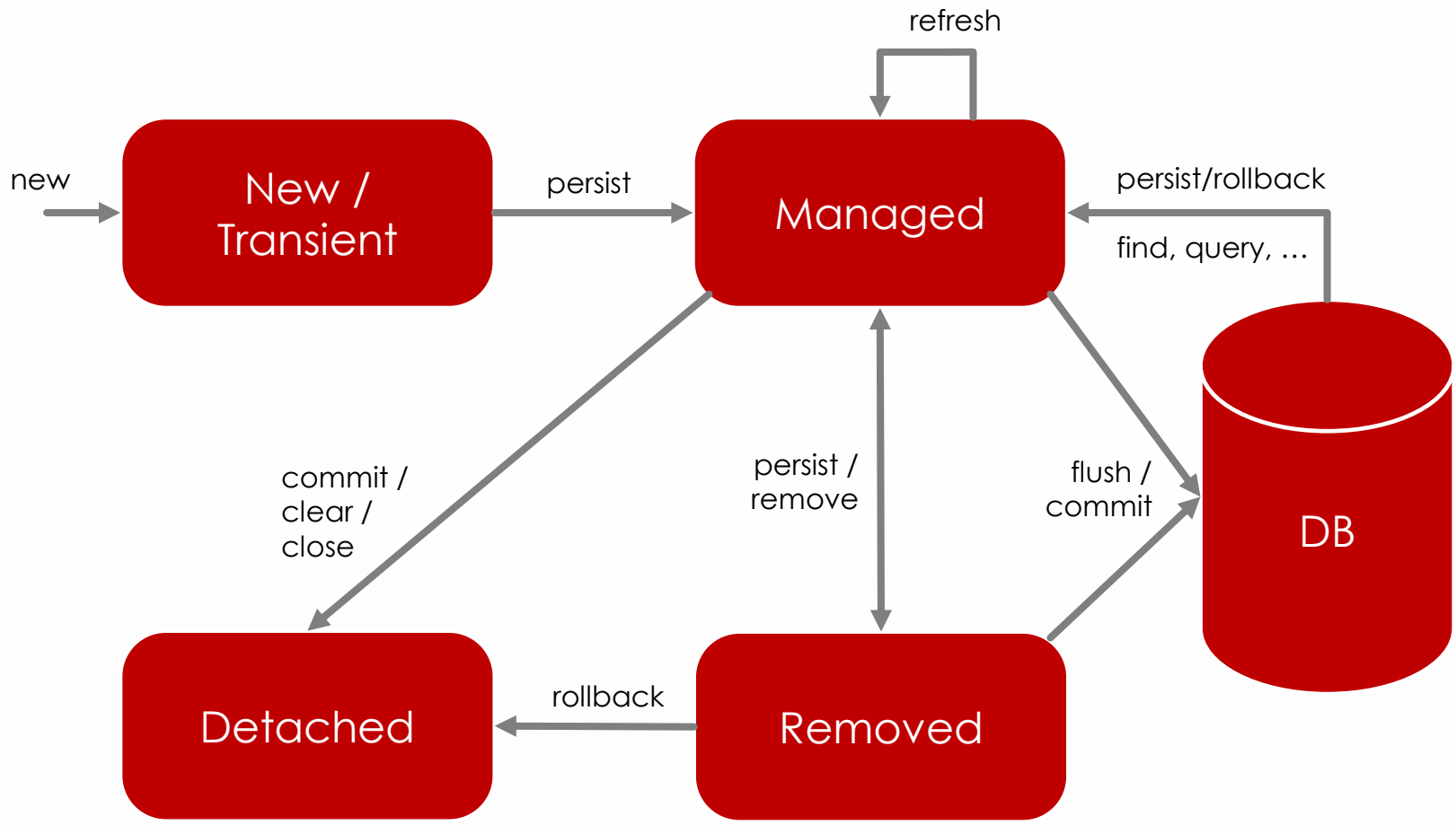
```
@Entity
public class Klasse implements Serializable {

        public Klasse() {
        }

        @Id
        private String variable;

        public String getVariable() {
                return variable;
        }

        public void setVariable(String v) {
                this.variable = v;
        }
}
```

# Entity Manager und -Factory



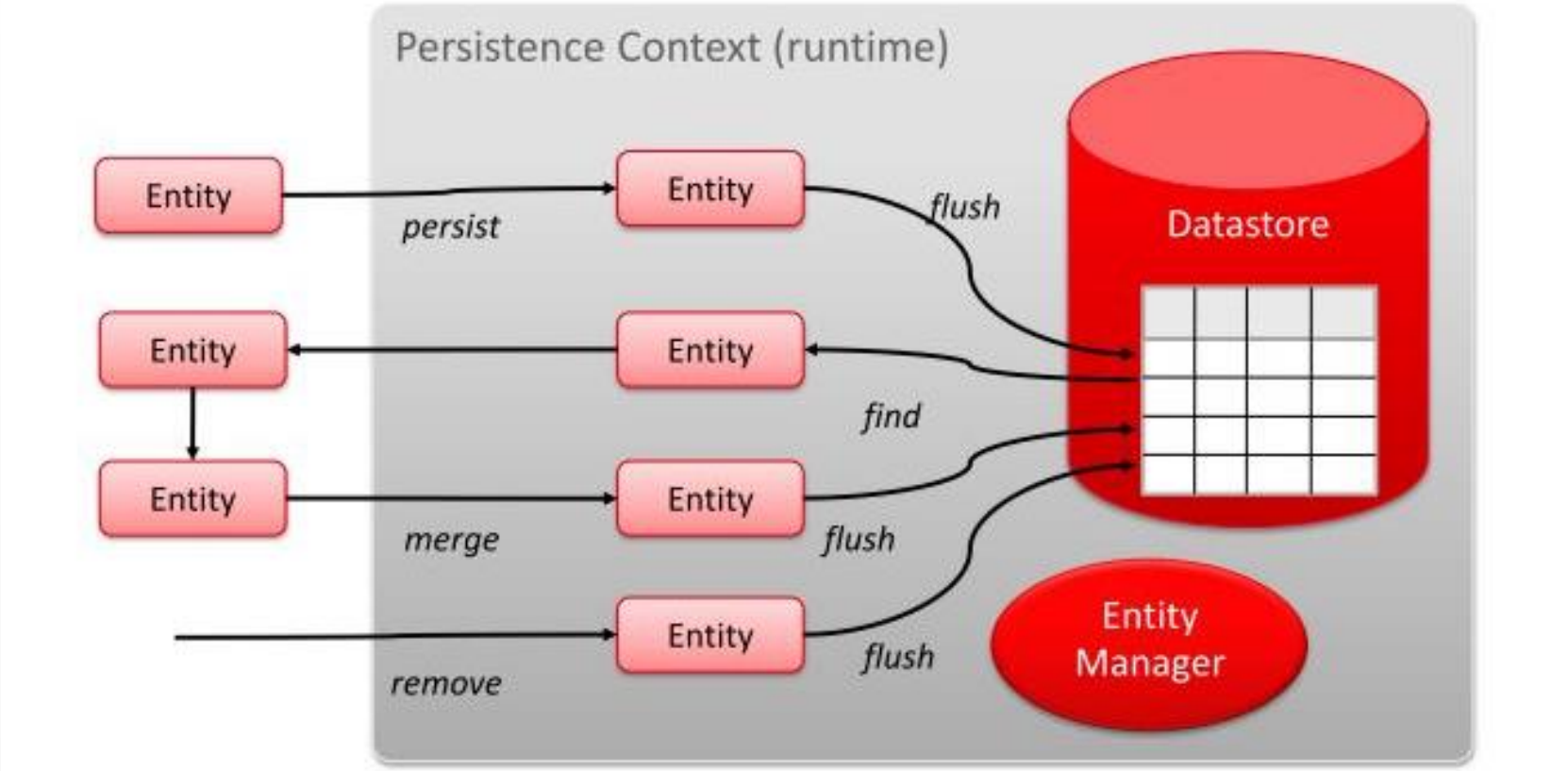

Quelle: http://www.eclipse.org/eclipselink/documentation/2.4/concepts/blocks001.htm

# Entity-Lifecycle




Angelehnt an Quelle: http://www.objectdb.com/java/jpa/persistence/managed

# Entity-Lifecycle




Angelehnt an Quelle: http://tschutschu.de/resources/SS2012_05_Persistence_JPA.pdf

# Beziehungen und Vererbung in JPA

| Beziehungstypen | Beziehungsrichtungen | Default Ladetypen |
|---|---|---|
| One To One | Uni-/Bidirektional | Fetch-Typ EAGER |
| Many To One | Unidirektional | Fetch-Typ EAGER |
| One To Many | Uni-/Bidirektional | Fetch-Typ LAZY |
| Many to Many | Uni-/Bidirektional | Fetch-Typ LAZY |

@Inheritance (strategy = InheritanceType…

| | |
|---|---|
| SINGLE_TABLE (default) | Eine Tabelle |
| @DiscriminatorColumn | |
| @DiscriminatorValue | |
| JOINED | Tabelle je Unterklasse |
| TABLE_PER_CLASS | Tabelle je konkreter Unterklasse |

# Agenda

- Grundlagen: Objektrelationale Abbildung
- ORM mittels JPA
- JPA 2.1
- Code-Beispiel

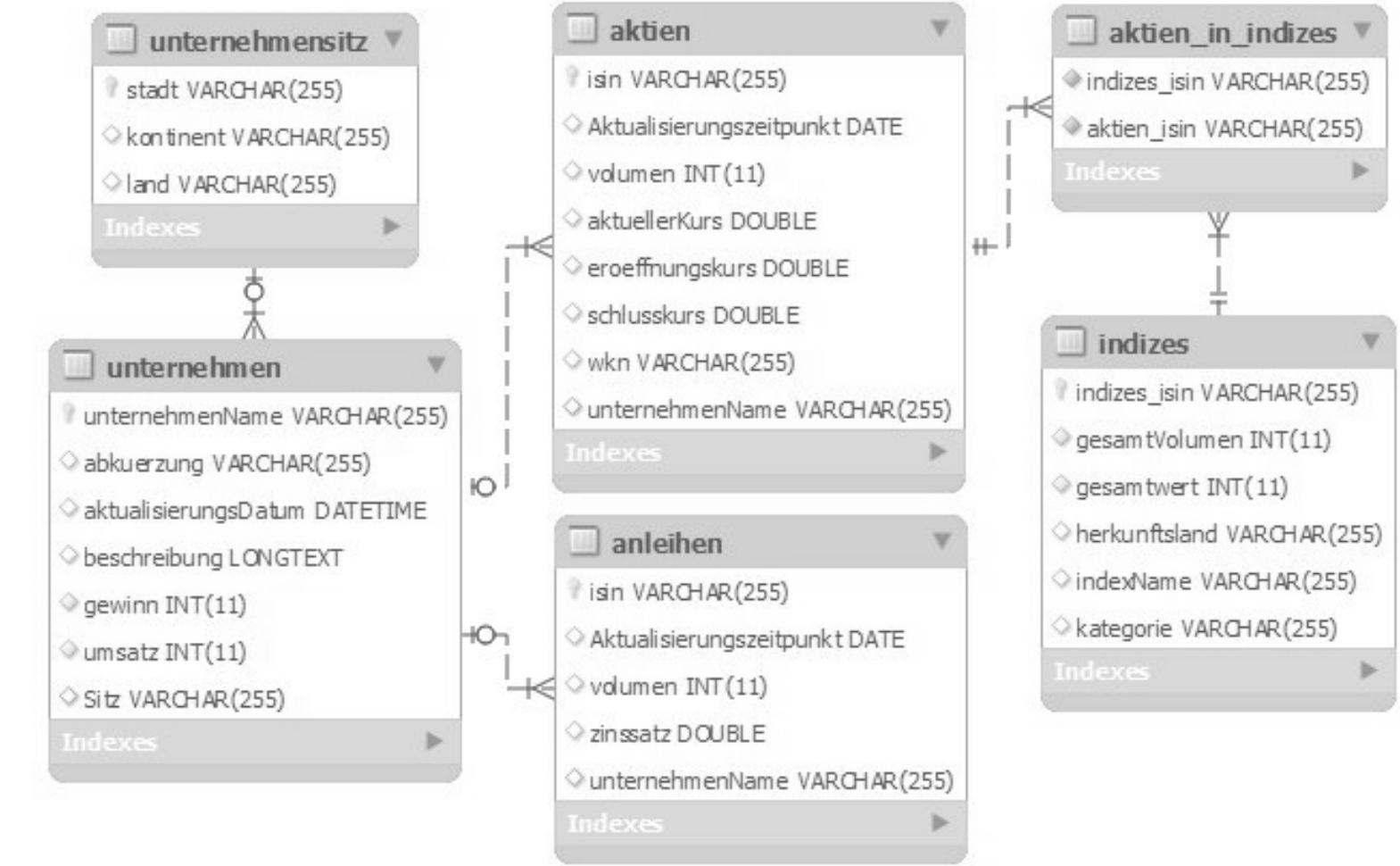
# JPA 2.1

- Update für Version JPA 2.0
- Teil der Neuerungen in der Spezifikation 2.1:
  - Converters
  - Schema generation
  - Unsynchronized persistence contexts
    - SynchronizationType.UNSYNCHRONIZED
    - Manuell über „joinTransaction“ synchronisieren
  - Query Language New Features

# Agenda

- Grundlagen: Objektrelationale Abbildung
- ORM mittels JPA
- JPA 2.1
- Code-Beispiel

# Code Beispiel



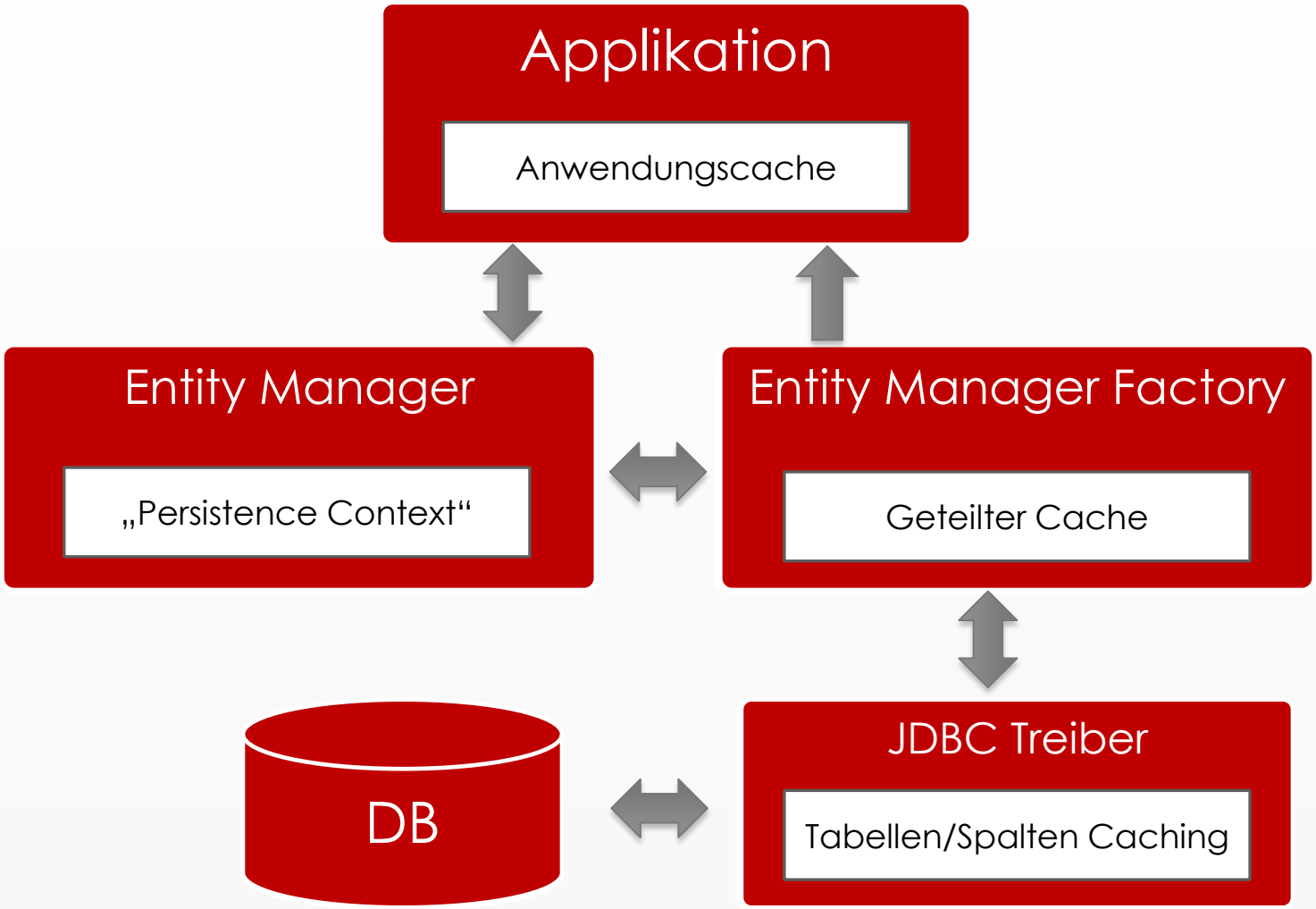
# Vielen Dank für Ihre Aufmerksamkeit!

# Fragen ?

Maximilian Spelsberg

# Backup

# Entity Manager und -Factory



# Grundlagen ORM


- Was ist ORM?
  - Das Speichern von Objekten in Datenbanken
- Warum Datenbank?
  - Effiziente Datenspeicherung, und Datenabfragen
  - Konsistente und dauerhafte Datenhaltung
- Was welche DBMS-Typen gibt es?
  - RDBMS
  - OODBMS
  - ORDBMS
- Was ist Persistenz?