```xml
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
             http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">

    <persistence-unit name="tourismPU" transaction-type="RESOURCE_LOCAL">

        <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>

        <class>be.ucll.tourism.domain.Country</class>

        <properties>
            <property name="javax.persistence.jdbc.url" value="jdbc:derby://localhost:1527/tourism;create=true"/>
            <property name="javax.persistence.jdbc.user" value="app"/>
            <property name="javax.persistence.jdbc.driver" value="org.apache.derby.jdbc.ClientDriver"/>
            <property name="javax.persistence.jdbc.password" value="app"/>
            <property name="javax.persistence.schema-generation.database.action" value="drop-and-create"/>
        </properties>
    </persistence-unit>
</persistence>
```

WHY DROP-AND-CREATE?

PERSISTENCE.XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
             http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">

    <persistence-unit name="tourismPU" transaction-type="RESOURCE_LOCAL">

        <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>

        <class>be.ucll.tourism.domain.Country</class>

        <properties>
            <property name="javax.persistence.jdbc.url" value="jdbc:derby://localhost:1527/tourism;create=true"/>
            <property name="javax.persistence.jdbc.user" value="app"/>
            <property name="javax.persistence.jdbc.driver" value="org.apache.derby.jdbc.ClientDriver"/>
            <property name="javax.persistence.jdbc.password" value="app"/>
            <property name="javax.persistence.schema-generation.database.action" value="drop-and-create"/>
        </properties>
    </persistence-unit>
</persistence>
```

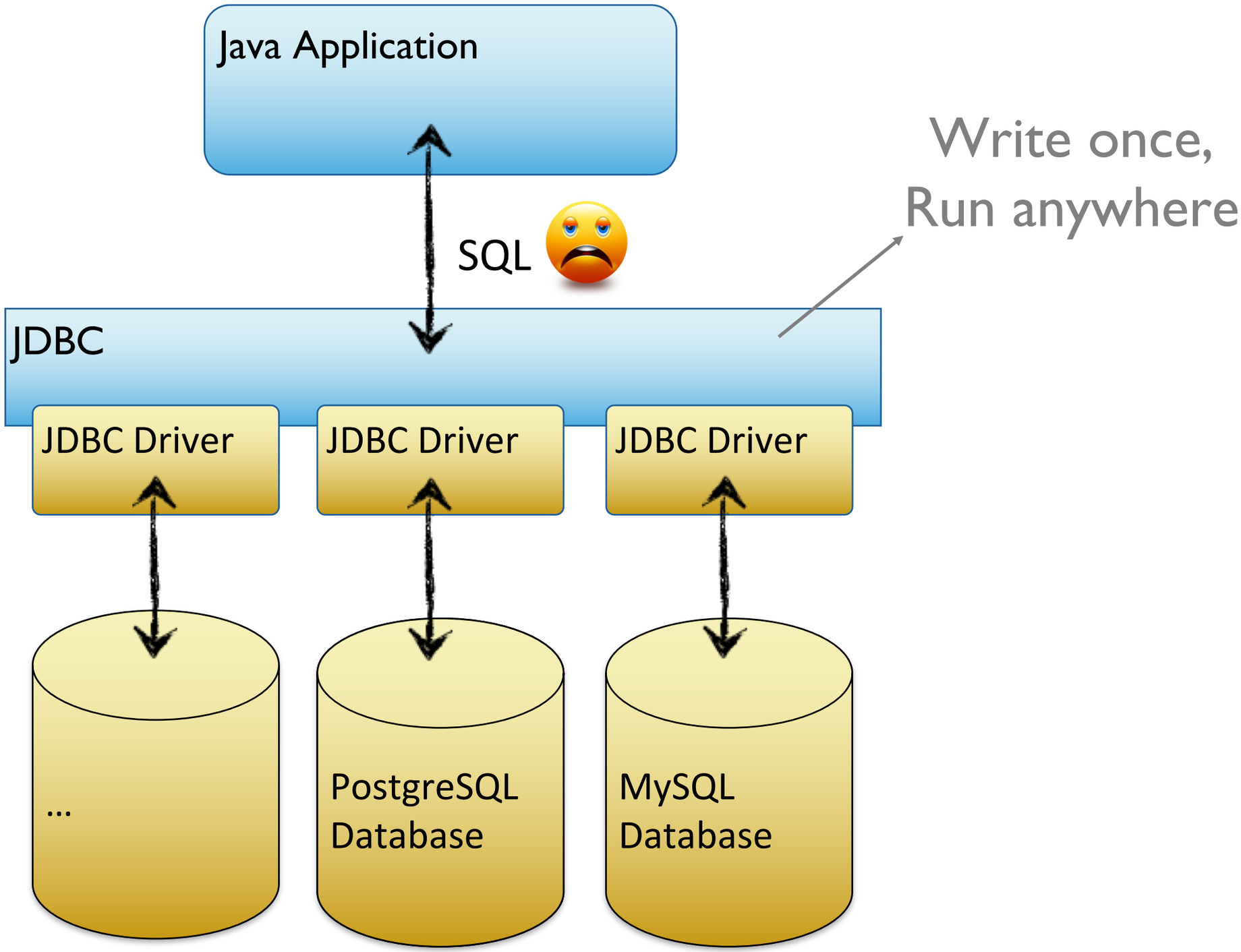
Other values possible:

- *none,*
- *create,*
- *drop-and-create,*
- *drop*

What if I want to use JPA In my application?

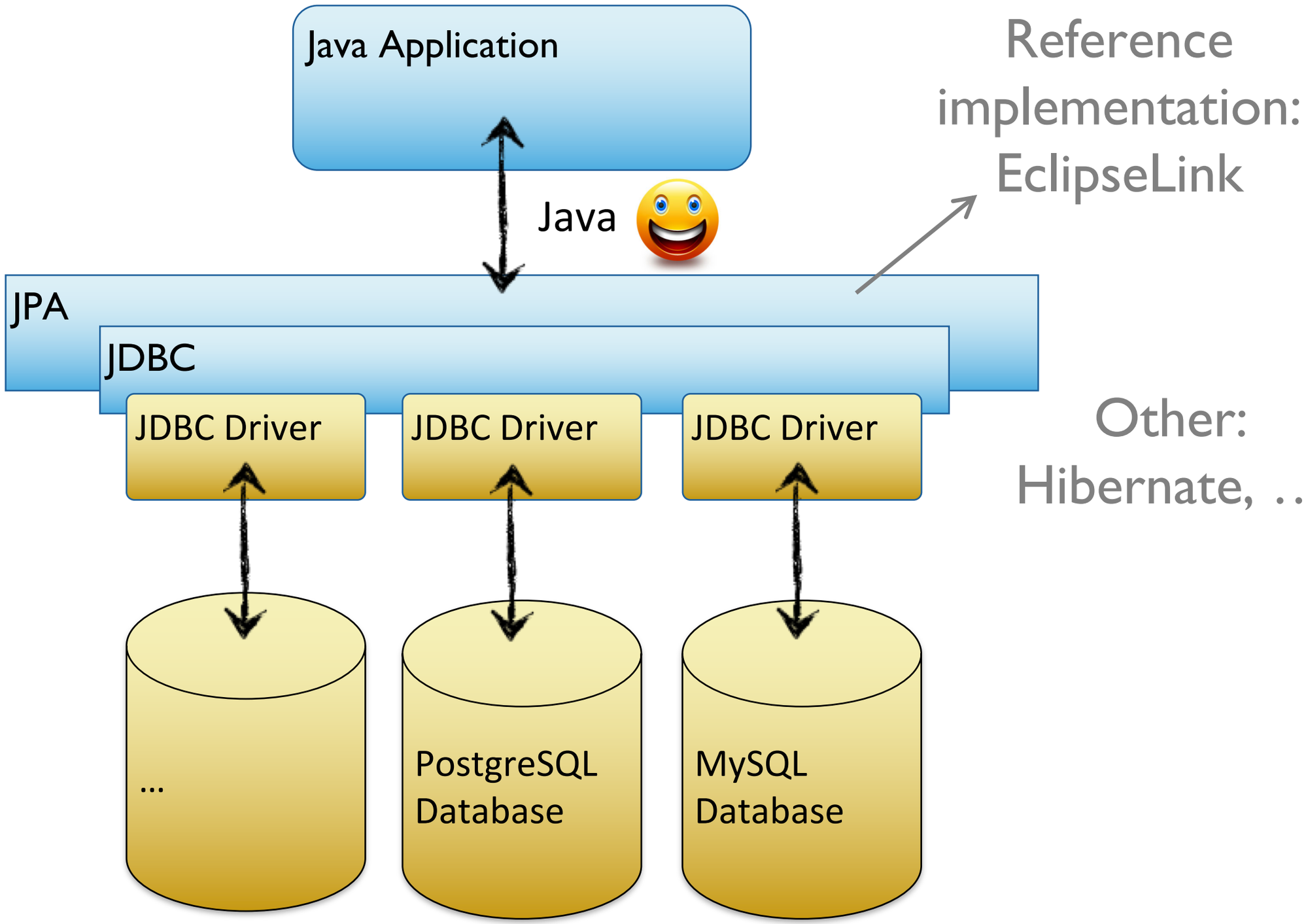
JDBC

Java DataBase Connectivity



JPA

Java Persistence API



Reference implementation: EclipseLink

Other: Hibernate, …

Model:

- ☑ Modify entity class
- ☑ Write persistence.xml
- ☑ Write database class
- ☑ Add libraries

When to open and close connections?

Connection pooling

JPA

Web 3

```
public void add(User user) {
    if (user == null) {
        throw new DbException("Nothing to add.");
    }
    String sql = "INSERT INTO app.person (userid, password) VALUES (?,?)";
    try {
        statement = connection.prepareStatement(sql);
        statement.setString(1, user.getUserid());
        statement.setString(2, user.getPassword());
        statement.execute();
    } catch (SQLException e) {
        throw new DbException(e);
    }
}

public List<User> getAll() {
    List<User> users = new ArrayList<User>();
    try {
        String sql = "SELECT * FROM app.person";
        statement = connection.prepareStatement(sql);
        statement.executeQuery();
        ResultSet result = statement.executeQuery();
        while (result.next()) {
            String userid = result.getString("userid");
            String password = result.getString("password");
            User user = new User(userid);
            user.setPassword(password);
            users.add(user);
        }
    } catch (SQLException e) {
        throw new DbException(e.getMessage(), e);
    }
    return users;
}
```

Do not forget

to close the statement

and the connection!

When?

When application is down?

After each request?

…?

1. Application Scope

- 100 simultaneous users:
  - how many connections ? ⟶ 1
  - time lost creating connections ? ⟶ Not much
  - how many statements per connection? ⟶ 1
  - scalable ? ⟶ No, threads will wait for each other !

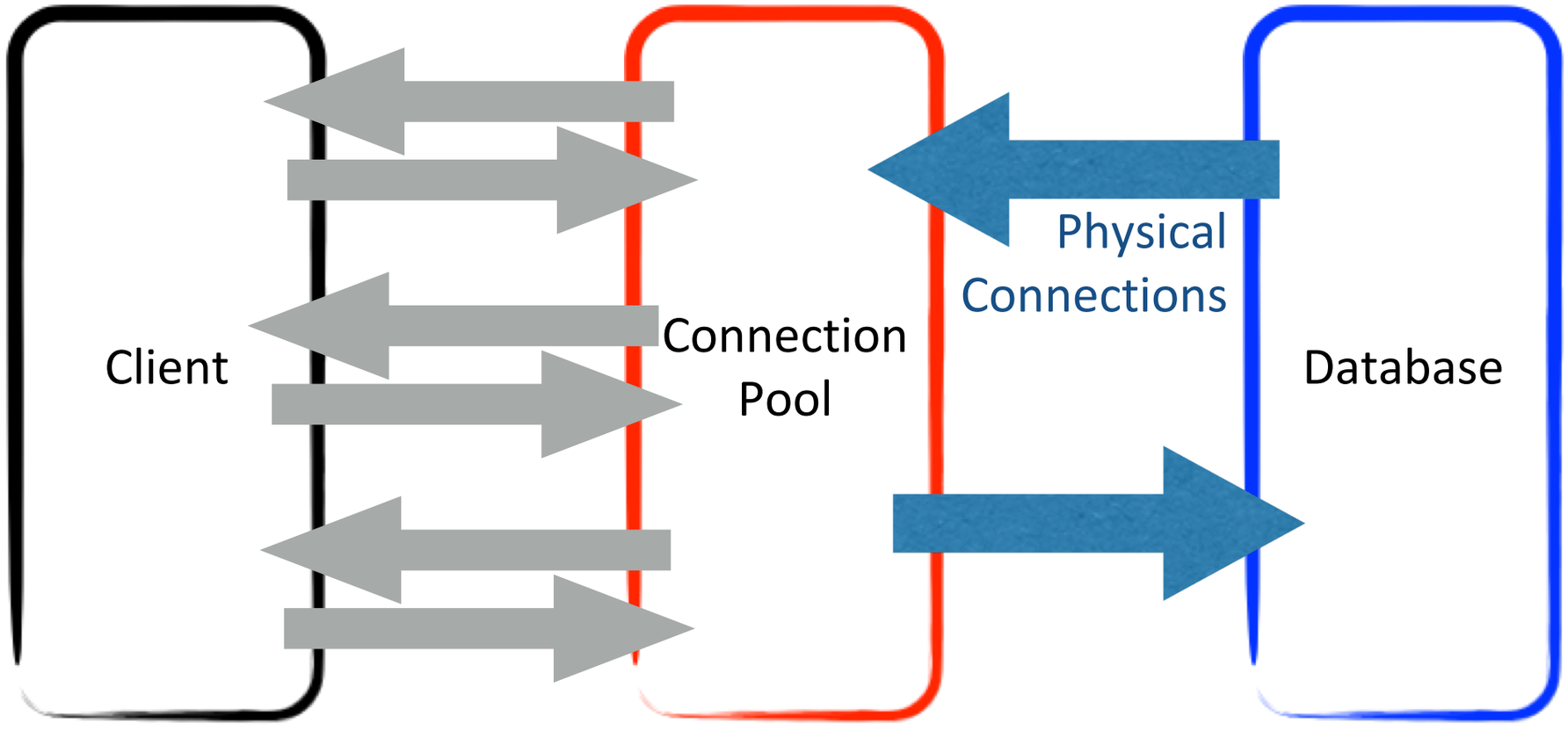
- 1 connection per application:
  - OK for desktop application
  - NOK for web application

2. Request Scope

- 100 simultaneous users:
  - how many connections ? ⟶ 100
  - time lost creating connections ? ⟶ A lot !
  - how many statements per connection? ⟶ 1
  - scalable ? ⟶ Yes

- 1 connection per request:
  - Better for web application
  - Performance issue !

3. Connection Pooling



‘Cache’of database connections:

- connection is created once
- added to the pool
- used over and over again

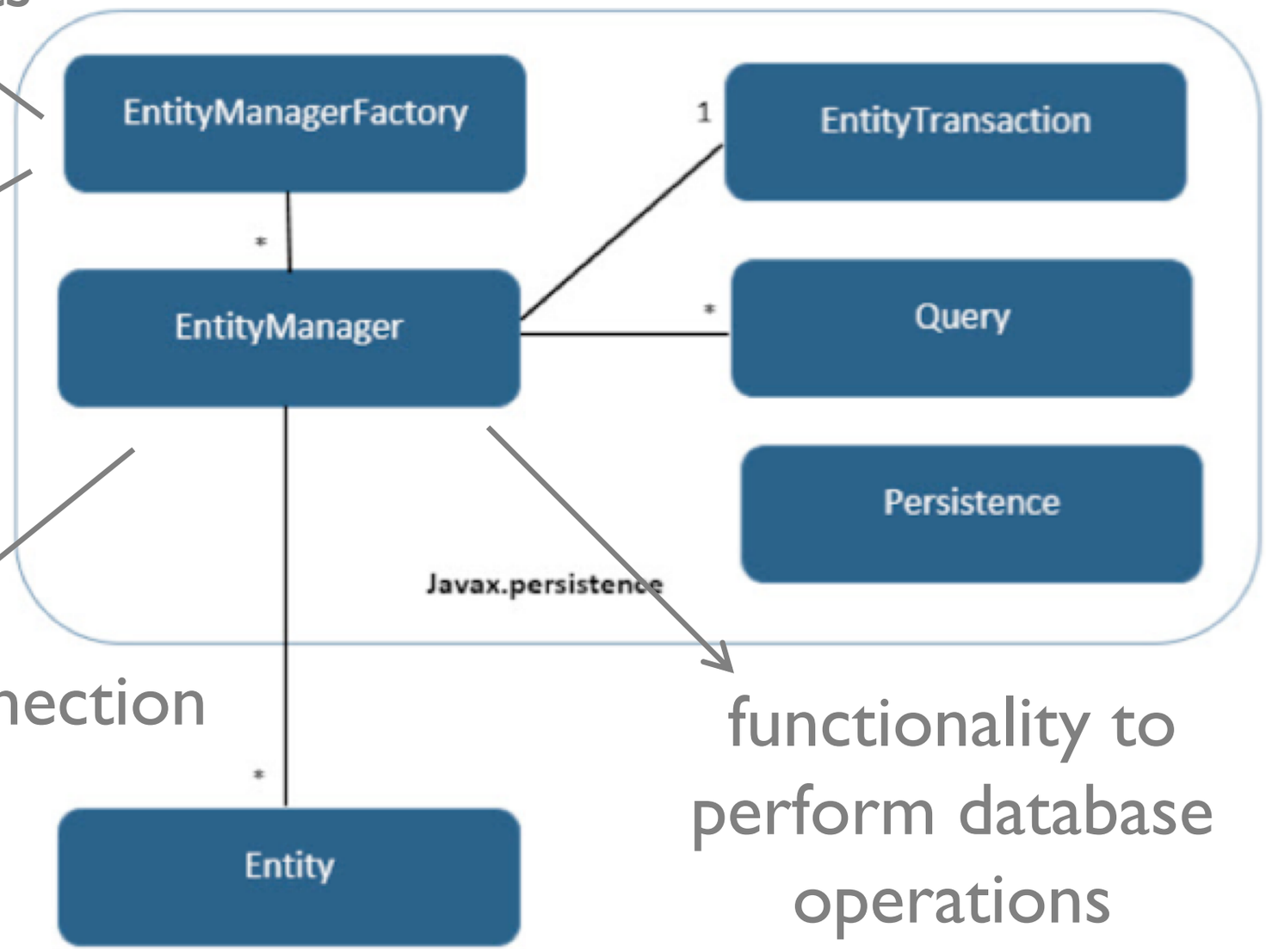
HOW

- Write it yourself
- JDBC
- JPA

JPA Overview

instantiates
EntityManager objects

maintains
connection pool



represents
database connection

functionality to
perform database
operations

http://www.objectdb.com/java/jpa/persistence/overview

```
...
private void openConnection() {
    emf = Persistence.createEntityManagerFactory(getPuName());
    em = emf.createEntityManager();
}
private void closeConnection() {
    try {
        em.close();
        emf.close();
    } catch (Exception e) {
        throw new DatabaseException(e.getMessage(), e);
    }
}
```

OK?

Not for every request!

```
public long add(Country object) {
   try {
        openConnection();
        em.getTransaction().begin();
        em.persist(object);
        em.flush();
        em.getTransaction().commit();
        closeConnection();
        return object.getId();
    } catch (Exception e) {
        throw new DatabaseException(e.getMessage(), e);
    }
}
...
```

OK?

```
...
private void openConnection() {
    em = emf.createEntityManager();
}
private void closeConnection() {
     try {
         em.close();
    } catch (Exception e) {
         throw new DatabaseException(e.getMessage(), e);
     }
}

public long add(Country object) {
    try {
         openConnection();
         em.getTransaction().begin();
         em.persist(object);
         em.flush();
         em.getTransaction().commit();
         closeConnection();
         return object.getId();
     } catch (Exception e) {
         throw new DatabaseException(e.getMessage(), e);
     }
}
...
```

Better!

What if anything goes wrong here?

rolback transaction !


Any
Question?