IUT Paris8– January 2012

Understanding JPA Session & Transaction

Proxy, Cache, Lazy Loading, Detached / DTO

Arnaud Nauwynck

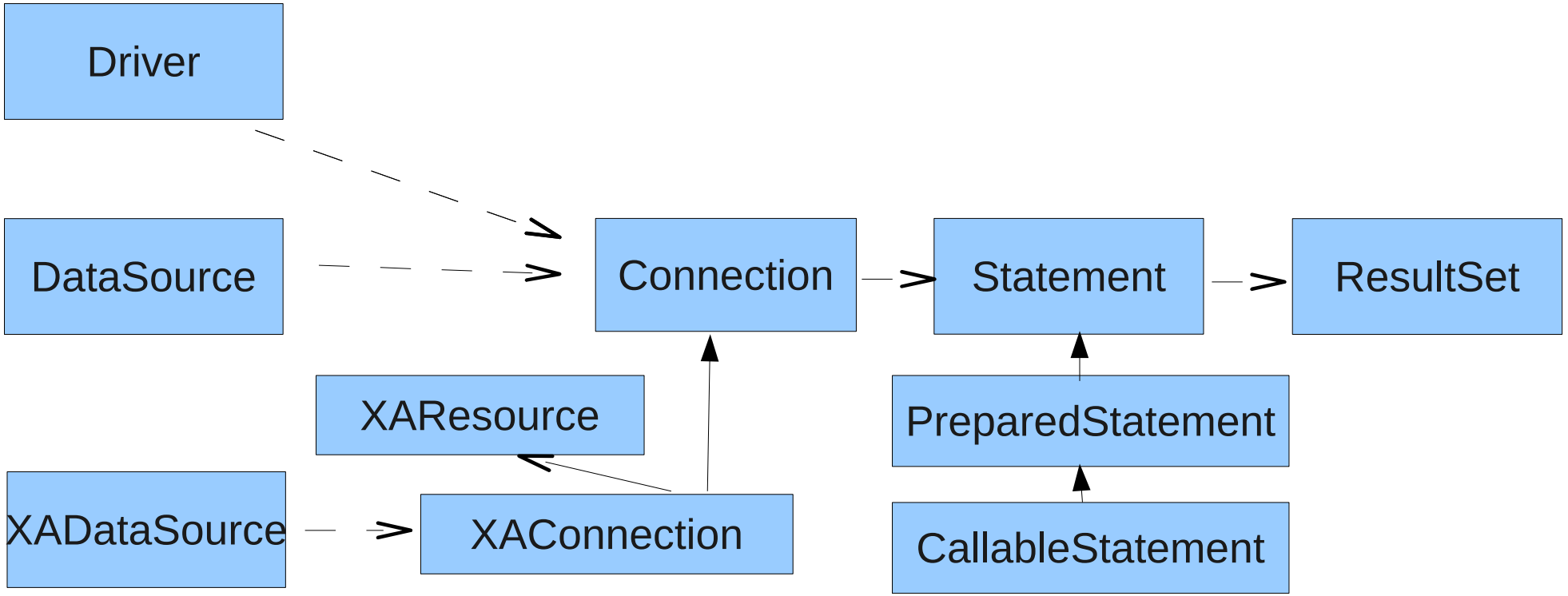
This document:
http://arnaud.nauwynck.chez-alice.fr/CoursIUT/JPA-SessionXA.pdf

Table Of Content

- Low Level
  - JDBC
  - JTA Transaction Manager, XAResource
-

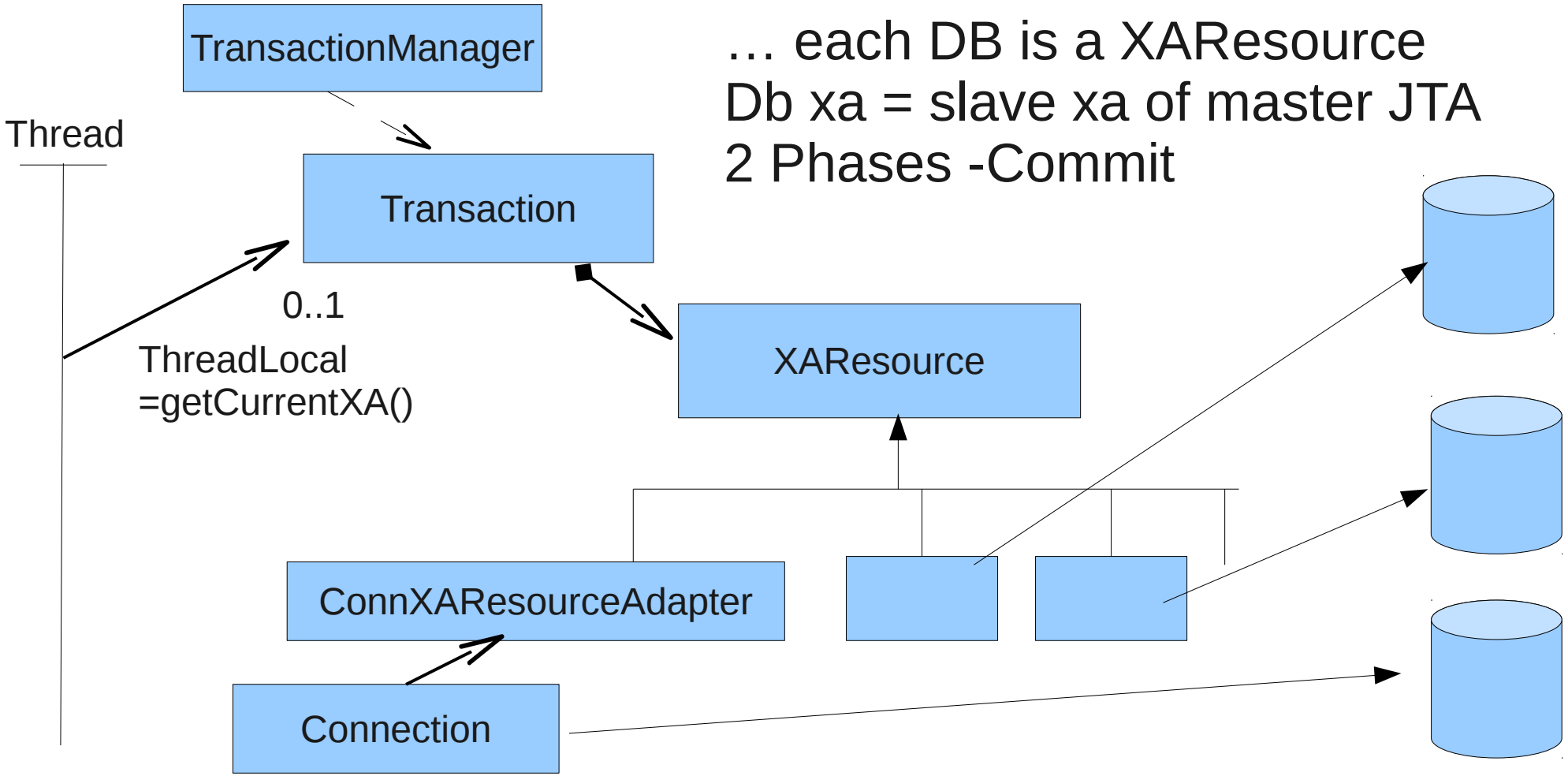
JDBC

- Standard, Stable and very well implemented
- API is vendor independent (but not SQL)



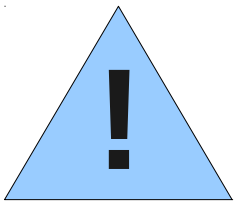
Jdbc … integrated with JTA

XA Multi-databases
… each DB is a XAResource
Db xa = slave xa of master JTA
2 Phases -Commit



Sample JDBC PreparedStatement

```
public List<Emp> findByLoginLike(String login) throws SQLException {
    List<Emp> res = new ArrayList<Emp>();
    Connection conn = null;
    try {
        conn = dataSource.getConnection();
        String sql = "select e.id, e.login, e.first_name, e.last_name"
                + " from EMP e"
                + " where e.login like ?";
        PreparedStatement pstmt = conn.prepareStatement(sql);
        pstmt.setString(1, login);
        ResultSet rs = pstmt.executeQuery();
        while(rs.next()) {
            Emp elt = new Emp();
            elt.setId(rs.getInt(1)); elt.setLogin(rs.getString(2));
            elt.setFirstName(rs.getString(3)); elt.setLastName(rs.getString(4));
            res.add(elt);
        }
    } finally {
        safeClose(conn);
    }
    return res;
}
```



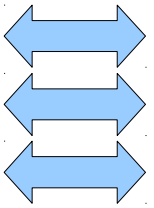
Rule of Thumb:
you open it => you close it !

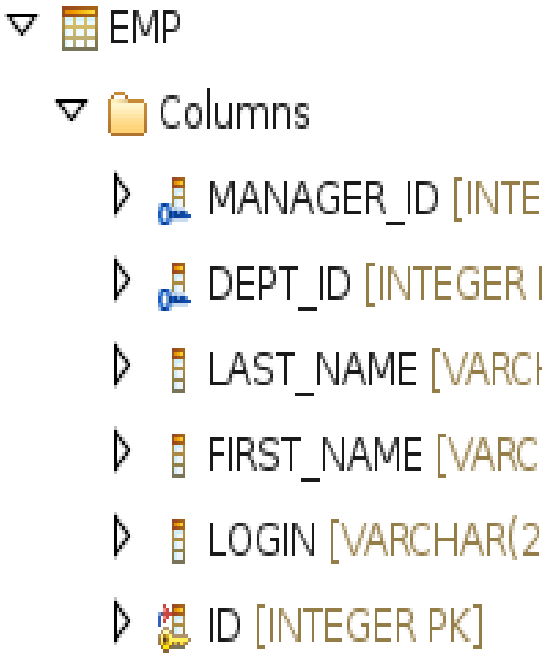
Close conn=>close pstmt => close rs

Same With JPA

```java
@Entity
public class Emp {

    @Id
    @GeneratedValue
    private int id;

    @Version
    private int version;

    private String login;

    private String firstName;

    private String lastName;
```

field=column





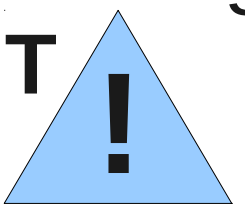

```java
public List<Emp> findByLoginLike(String loginPattern) {
    TypedQuery<Emp> qry = em.createQuery("select e from Emp e where e.login like ?", Emp.class);
    qry.setParameter(1, loginPattern);
    return qry.getResultList();
}
```

SQL Low Level CRUD

- CRUD = Create-Read-Update-Delete

- Create
```
INSERT Emp (id,name)
VALUES(?,?)
```
- Read
```
SELECT e.*
FROM emp e
WHERE e.id = ?
```
- Update
```
UPDATE Emp e
SET e.salary = ?
WHERE e.ID = ?
```
- Delete
```
DELETE Emp e
WHERE e.id = ?
```

… DON'T forget
COMMIT



Set autocommit false

CRUD with JPA

- **C**reate
- **R**ead
  - By Id
- **U**pdate
- **D**elete

```
@PersistenceContext
protected EntityManager em;

public void persist(T entity) {
    em.persist(entity);
}

public T findById(int id) {
    return em.find(entityClass, id);
}

// UPDATE ... nothing in DAO!
// simply cal setter

public void remove(T entity) {
    em.remove(entity);
}
```

That's All … That works

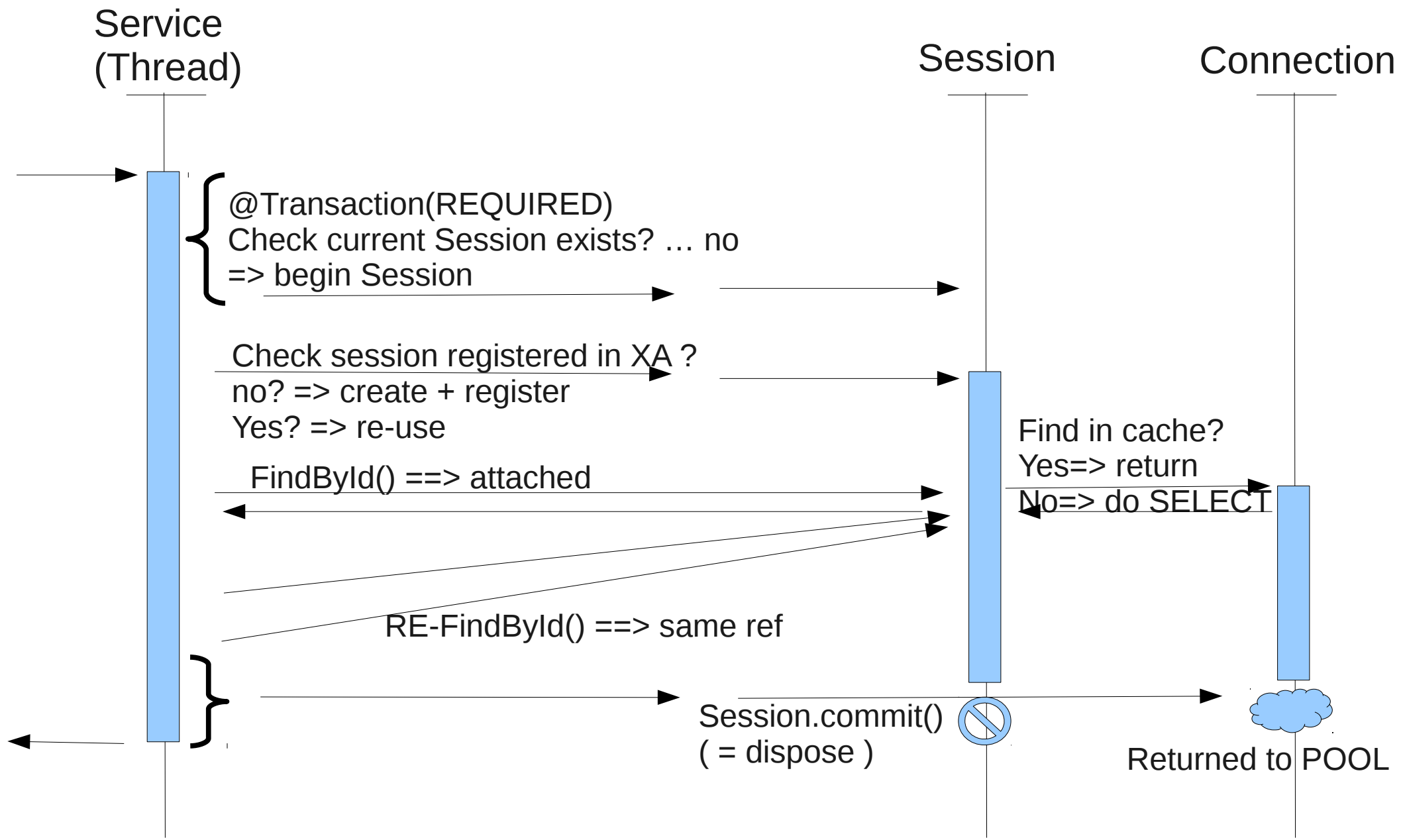
The API is simple

BUT you need to understand underneath...

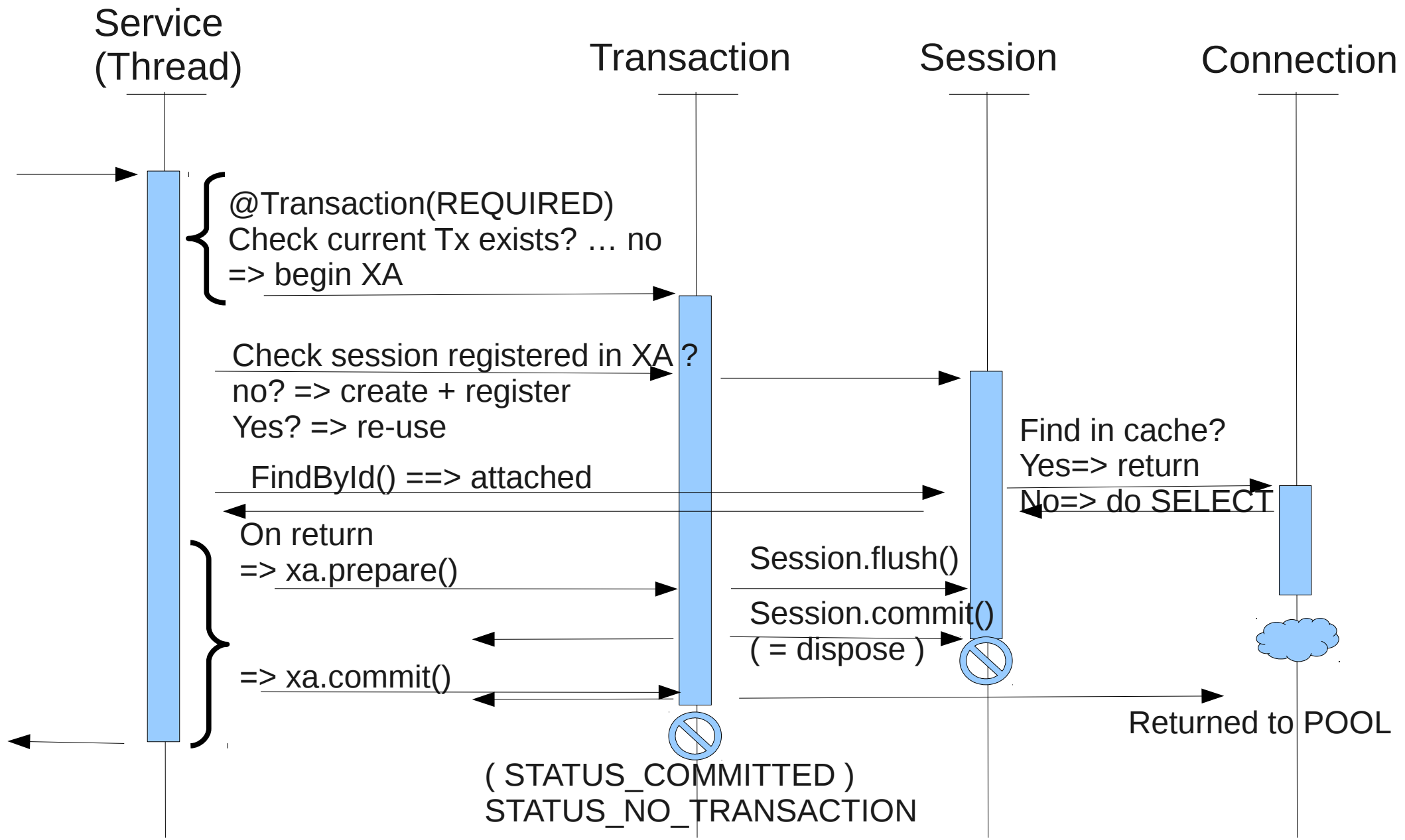
- General concepts / Behaviors
  - Transaction – XAResource - Session – Attached/Detached Entity – LazyLoading – DeferredUpdate - Lock …
- And also Vendor Specific...
  - Cache Optimizations, Cache sync, Proxy, ReadOnly
  - SQL Joins, Batch Read, etc...

Rule 1 : Query Multiple Times the Same Attached Entity …

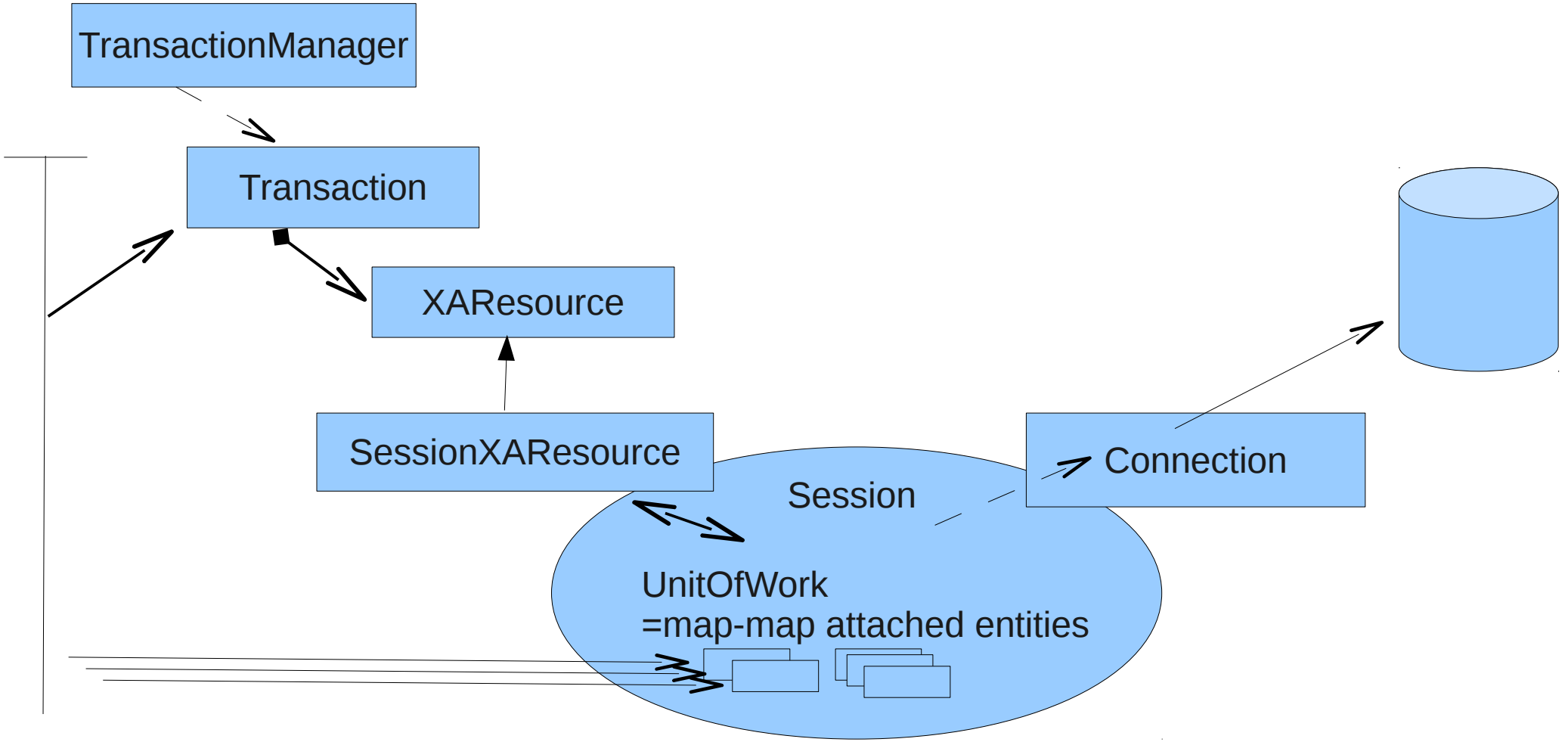
Looking in the JPA Find (cRud)



Zooming Session … => Transaction



Session = Glue between Transaction (XAResource) and Database (Connection) + hold Attached Entities



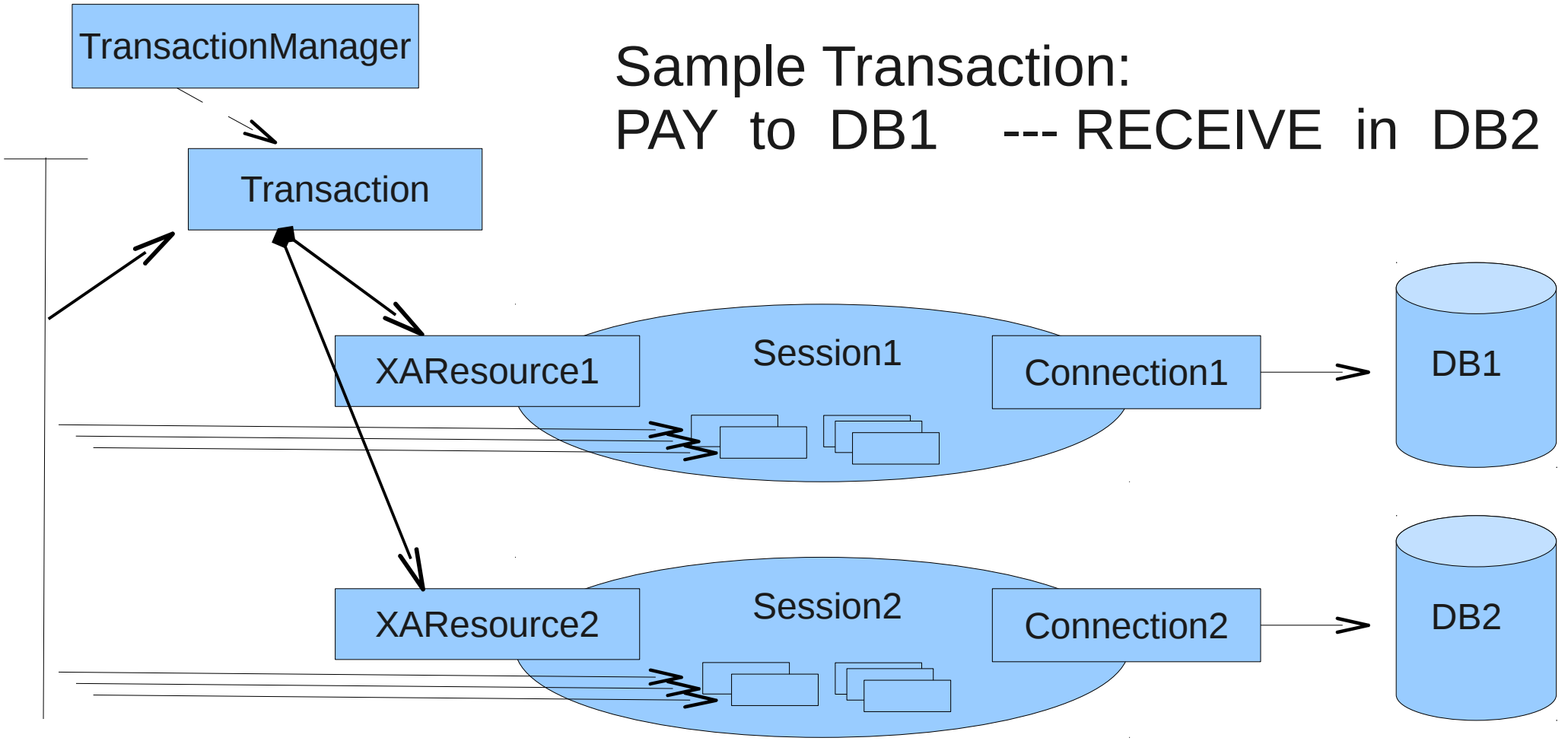
Pseudo-Code for Transaction-XAResource enlist

```
interface XAResource {
    public Object getXAResourceKey();
    public void prepareXA();
    public void commitXA();
    public void rollbackXA();
}

class Transaction {
    Map<Object,XAResource> enlistedXAResources = new HashMap<Object,XAResource>();

    public XAResource getEnlistedXAResource(Object xaResourceKey) {
        return enlistedXAResources.get(xaResourceKey);
    }
    public void enlistXAResource(XAResource xaResource) {
        enlistedXAResources.put(xaResource.getXAResourceKey(), xaResource);
    }
    public Collection<XAResource> getEnlistedXAResources() {
        return enlistedXAResources.values();
    }

    public void prepareXA() { for (XAResource r : enlistedXAResources.values()) { r.prepareXA(); } }
    public void commitXA() { for (XAResource r : enlistedXAResources.values()) { r.commitXA(); } }
    public void rollbackXA() { for (XAResource r : enlistedXAResources.values()) { r.rollbackXA(); } }
}
```

Pseudo-code for Transaction get or register Session

```
protected Session getOrRegisterSessionForCurrentTransaction() {
    Session session;
    Transaction xa = em.getTransactionManager().getCurrentTransaction();
    XAResource sessionXAResource = xa.getEnlistedXAResource(em.sessionFactory.getXAResourceKey());
    if (sessionXAResource == null) {
        // first time Session is used from transaction => create it + register in XA
        session = em.sessionFactory.createSession();
        sessionXAResource = session.getXaResourceAdapter();
        xa.enlistXAResource(sessionXAResource);
    } else {
        // re-use Session already registered in XA
        SessionXAResourceAdapter sessionXAAdapter = (SessionXAResourceAdapter) sessionXAResource;
        session = sessionXAAdapter.getSession();
    }
    return session;
}
```

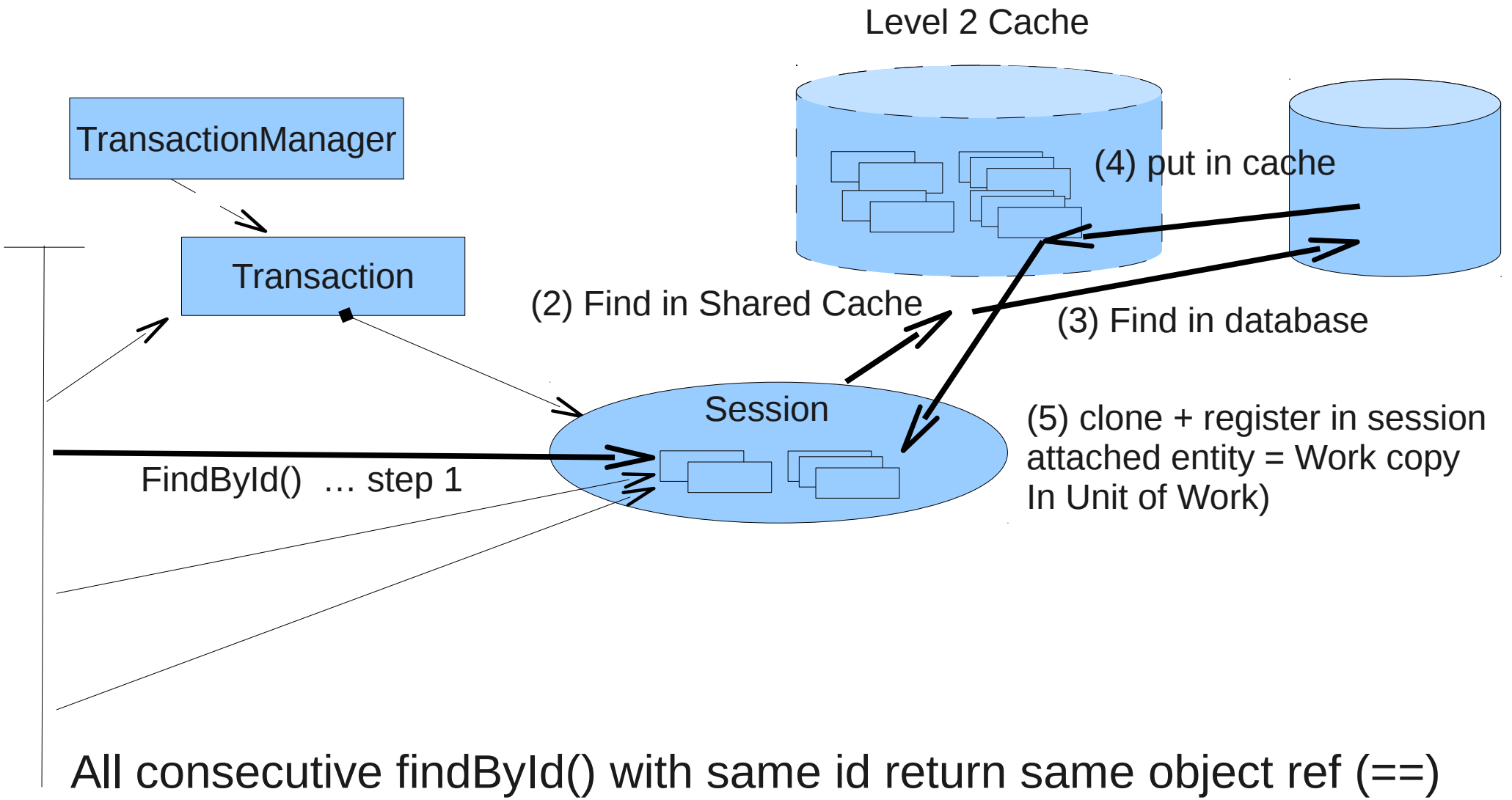
Generalisation to understand...
1 Transaction – N Sessions (N DBs)



Pseudo-code for “em.findById()”

```
public Object em_find_PseudoCode(Class<?> entityClass, int id) {
    // step 1: current Transaction (ThreadLocal ) -> get already enlisted or register new Session
    Session session = getOrRegisterSessionForCurrentTransaction();

    // step 2: session.find()
    // first try find in UnitOfWork  (= Cache Level 1)
    Object foundEntity = session.findEntityInUOW(entityClass, id);
    if (foundEntity != null) {
        return foundEntity; // case found attached entity (already registered in "unit of work")
    } else {
        // not found in unit of work => do find in database
        // step 3-1: execute jdbc SELECT in database + convert jdbc->entity)
        Connection conn = null;
        try {
            conn = getOrRegisterDataSourceConnectionForCurrentTransaction(session);
            PreparedStatement pstmt = buildFindByIdPstmt(conn, id); // build query for entity "select
            ResultSet rs = pstmt.executeQuery();
            if (!rs.next()) { throw new EntityNotFoundException(); }
            try { foundEntity = entityClass.newInstance(); } catch (Exception e) { throw new RuntimeEx
            // step 3-b: ...  transform jdbc ResultSet to Object  (="hydrateObject") by introspection
        } catch(SQLException ex) {
            throw new RuntimeException();
        } finally {
            // close connection (=> still locked as XAResource to XA)
            if (conn != null) try { conn.close(); } catch(Exception ex) {}
        }
        // step 3-c: register attached entity to Unit Of Work
        session.registerEntityInUOW(entityClass, id, foundEntity);
    }
    return foundEntity;
}
```

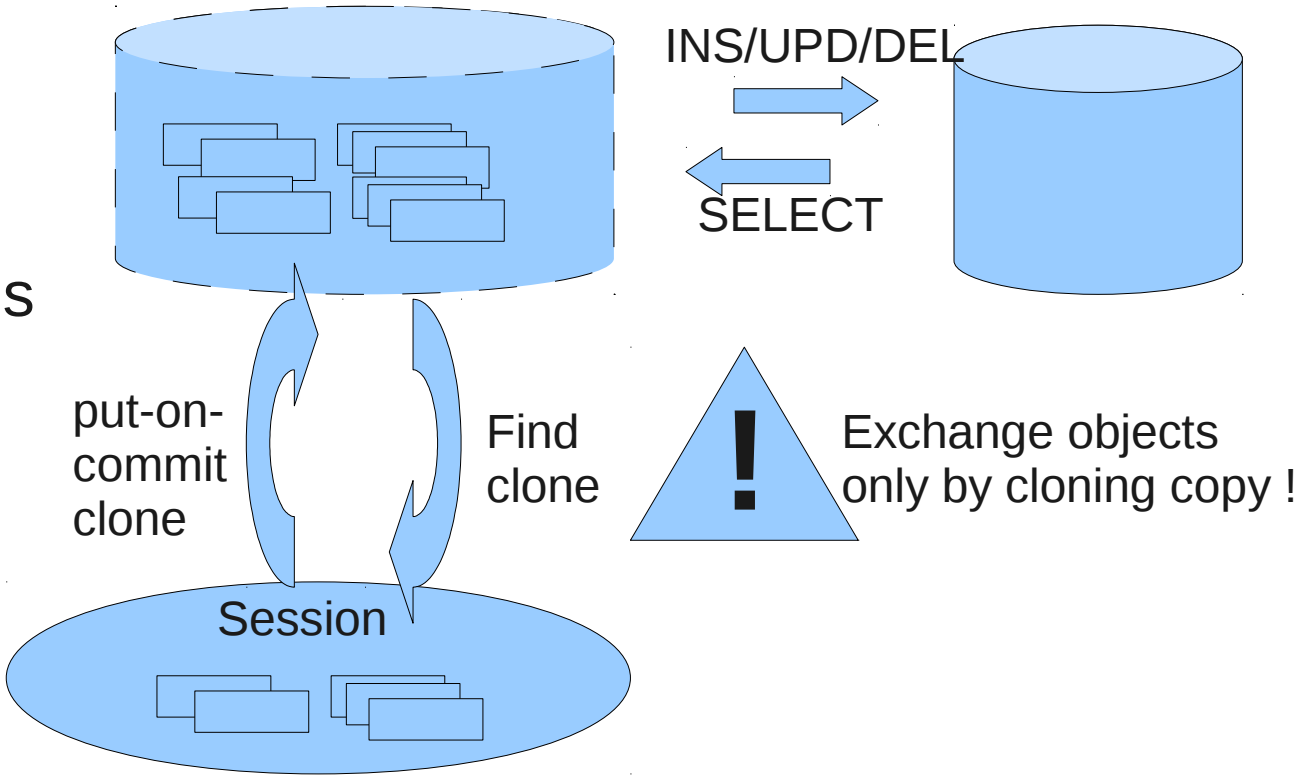
Find By ID
Attached Entity (=UnitOfWork / Level1)
+ Cache Level 2



All consecutive findById() with same id return same object ref (==)

Session = RW - Single Threaded
Level 2 Cache = RO Shared

Shared Cache
= Level 2 Cache
= Partial Mirror of DB
= Read-Only objects
… Shared by All threads



Session
= Unit Of Work
= Map Entity by Ids
= R-W
= Level 1
… Owned by Single Thread