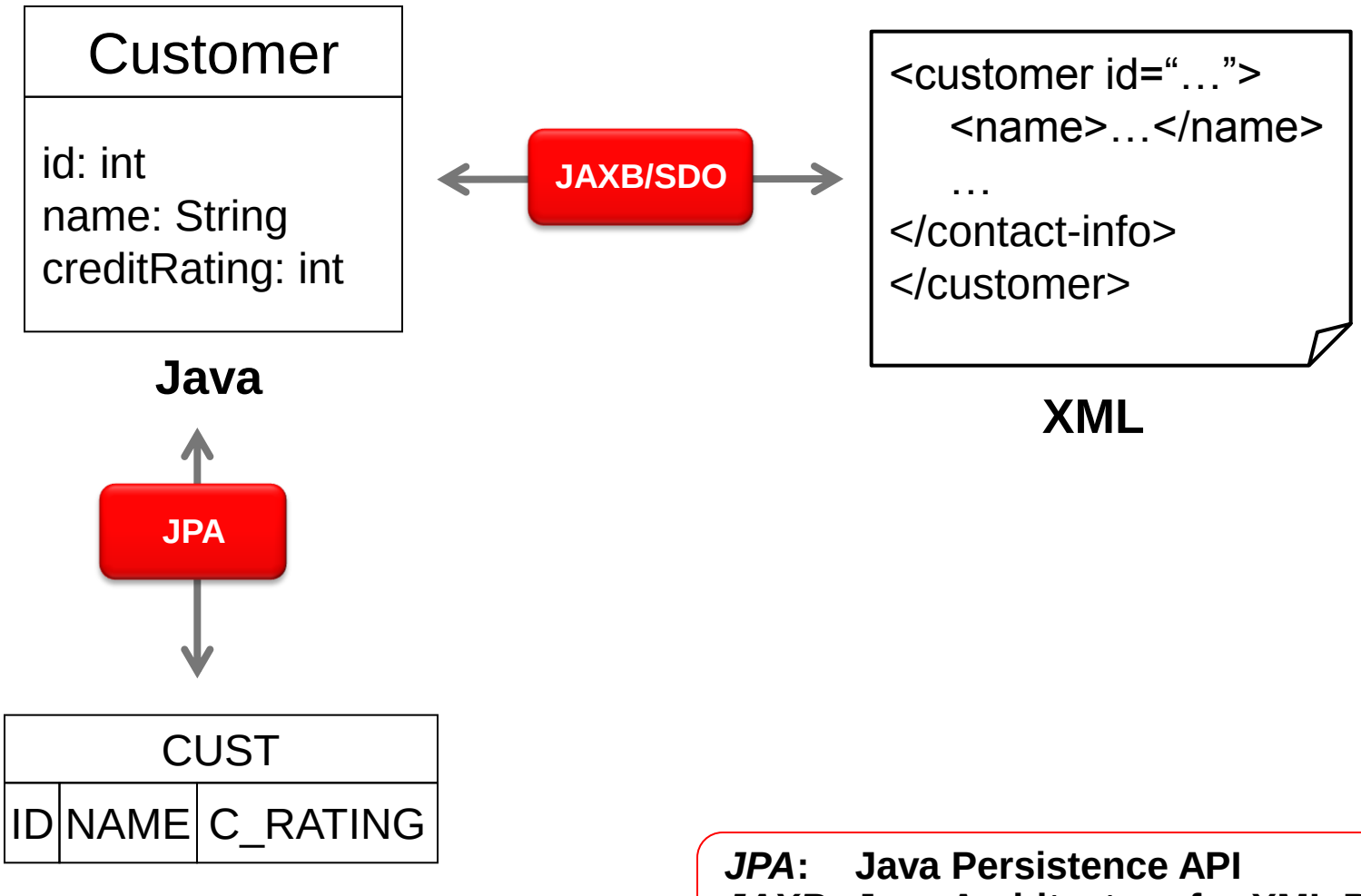
ORACLE® @ QCon

TopLink Grid: Scaling JPA Applications with Coherence

Shaun Smith, Principal Product Manager
Oracle Server Technologies, TopLink

Java Persistence: The Problem Space



JPA: **Java Persistence API**
JAXB: **Java Architecture for XML Binding**
SDO: **Service Data Objects**

EclipseLink Project

- Provides JPA, JAXB, SDO, DBWS, and EIS persistence services
- Open source Eclipse project
- Project Lead by Oracle
- Founded by Oracle with the contribution of full TopLink source code and tests
- Based upon product with 12+ years of commercial usage

EclipseLink JPA

- JPA 1.0 compliant with advanced persistence
- JPA 2.0 Reference Implementation (JSR 317)
- Supports Java EE, Java SE, Web, Spring, and OSGi
- Supports all leading RDMS with platform specific features
  - Best JPA for the Oracle Database—supporting advanced features
- Extensible and pluggable
- Key infrastructure:
  - Caching, Locking, Query Framework, Mapping, …
- … plus many valuable advanced features

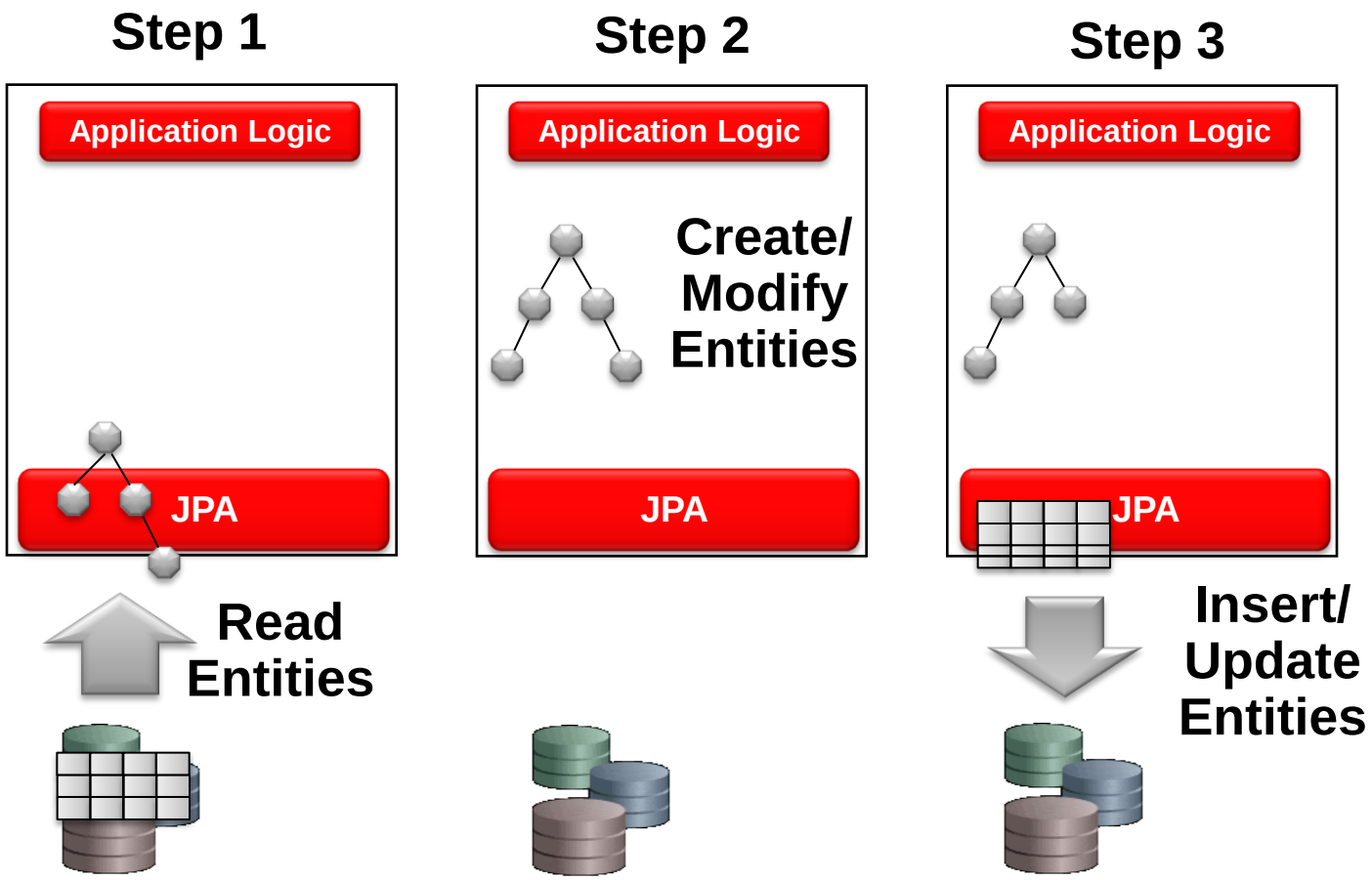
Oracle TopLink 11gR1

- Oracle's Enterprise Java Persistence Framework
  - Includes open source EclipseLink with Commercial Support
  - Certified on WebLogic and redistributed by Oracle as part of TopLink product
  - TopLink Grid: JPA integration with Coherence
  - Included in WebLogic Server
  - Tooling Support in JDeveloper and Eclipse

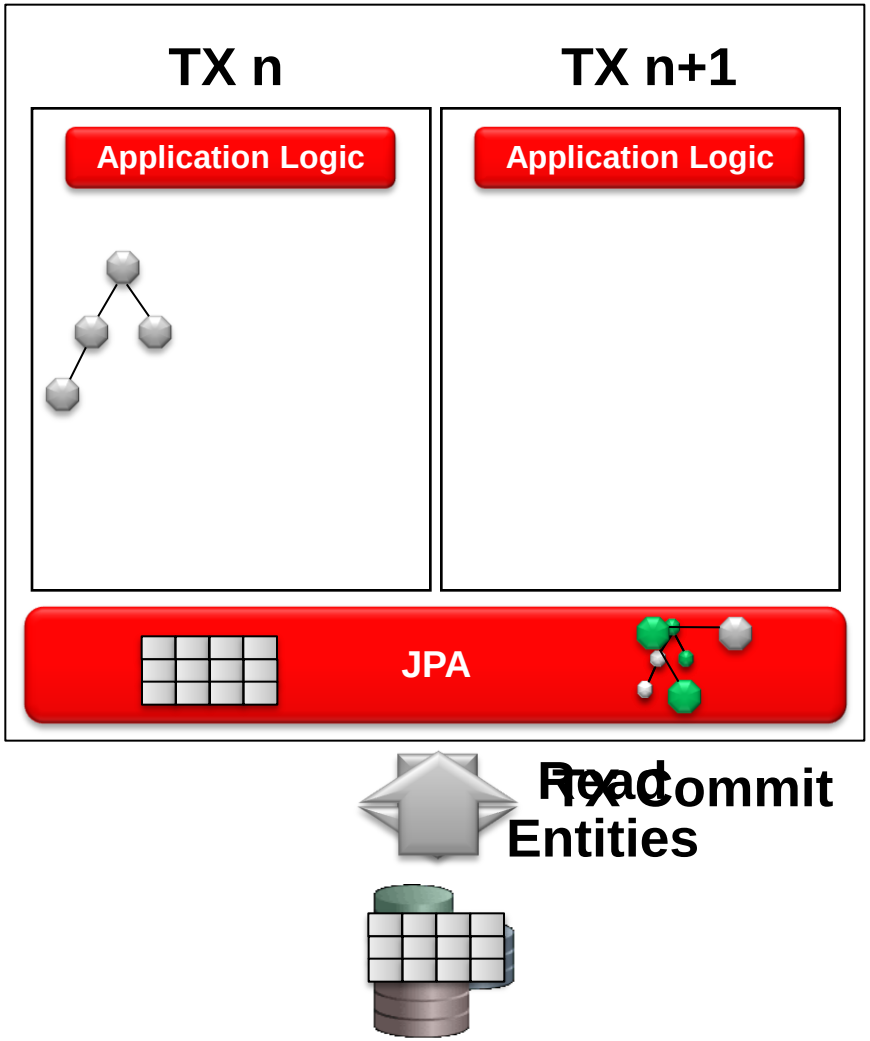
Example JPA Client Code

```
EntityManagerFactory emf = Persistence
        .createEntityManagerFactory("employee");
EntityManager em = emf.createEntityManager();

em.getTransaction().begin();
Employee employee = new Employee();
em.persist(employee);
em.getTransaction().commit();

em.close();
emf.close();
```

Mechanics of a JPA Application

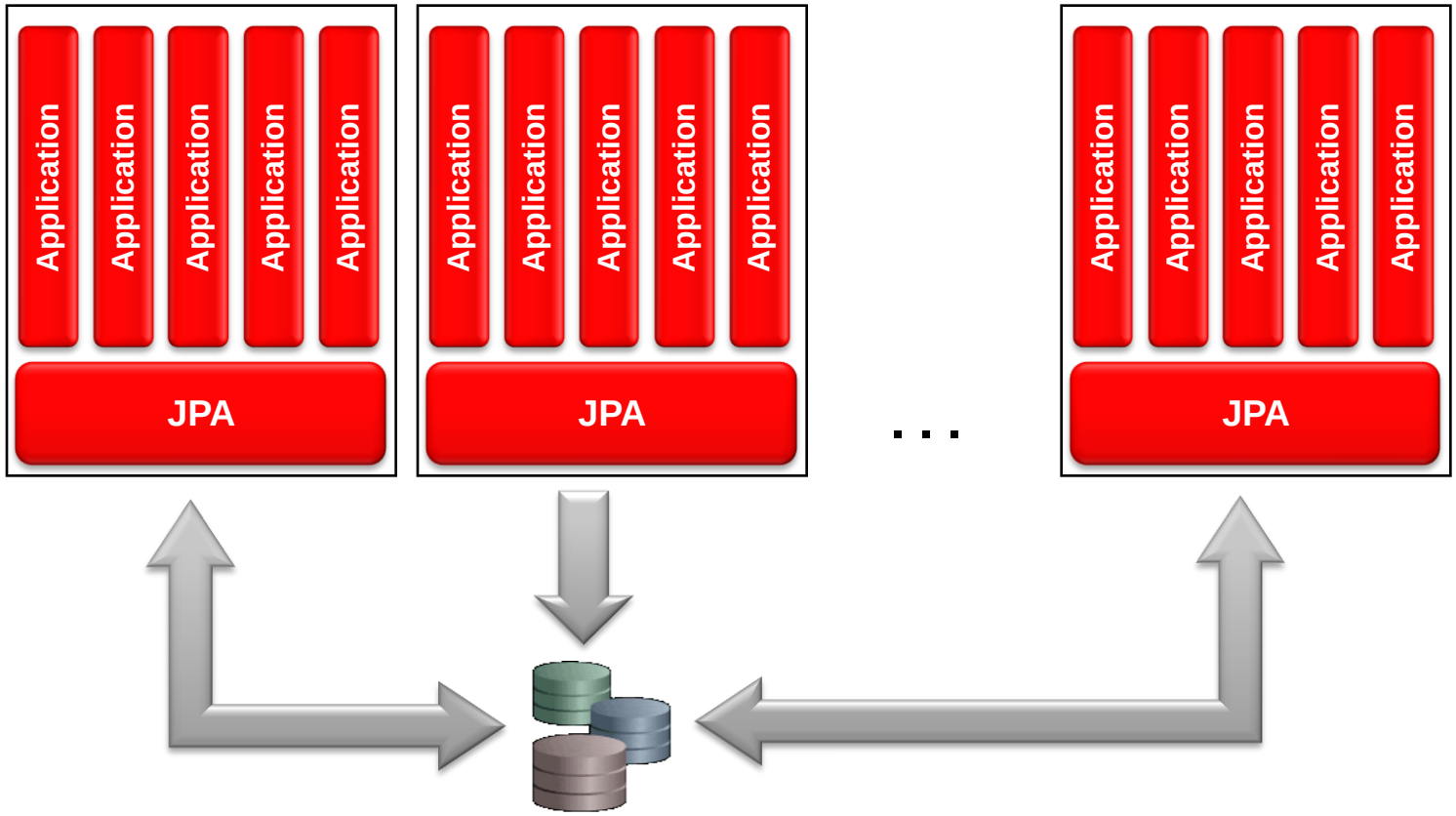


JPA with Cache

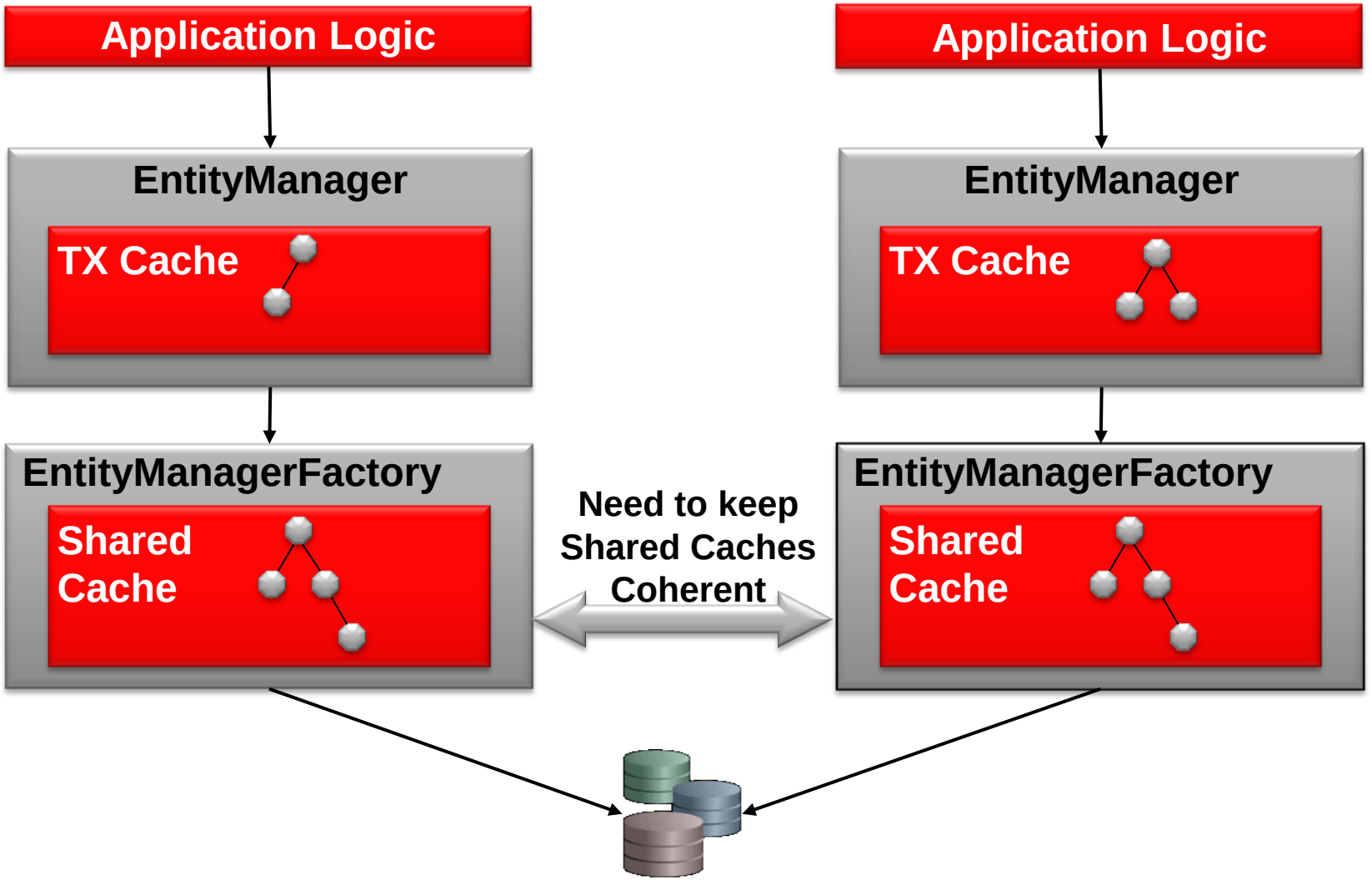


Cache hits avoid object build cost

Scaling Java Persistence



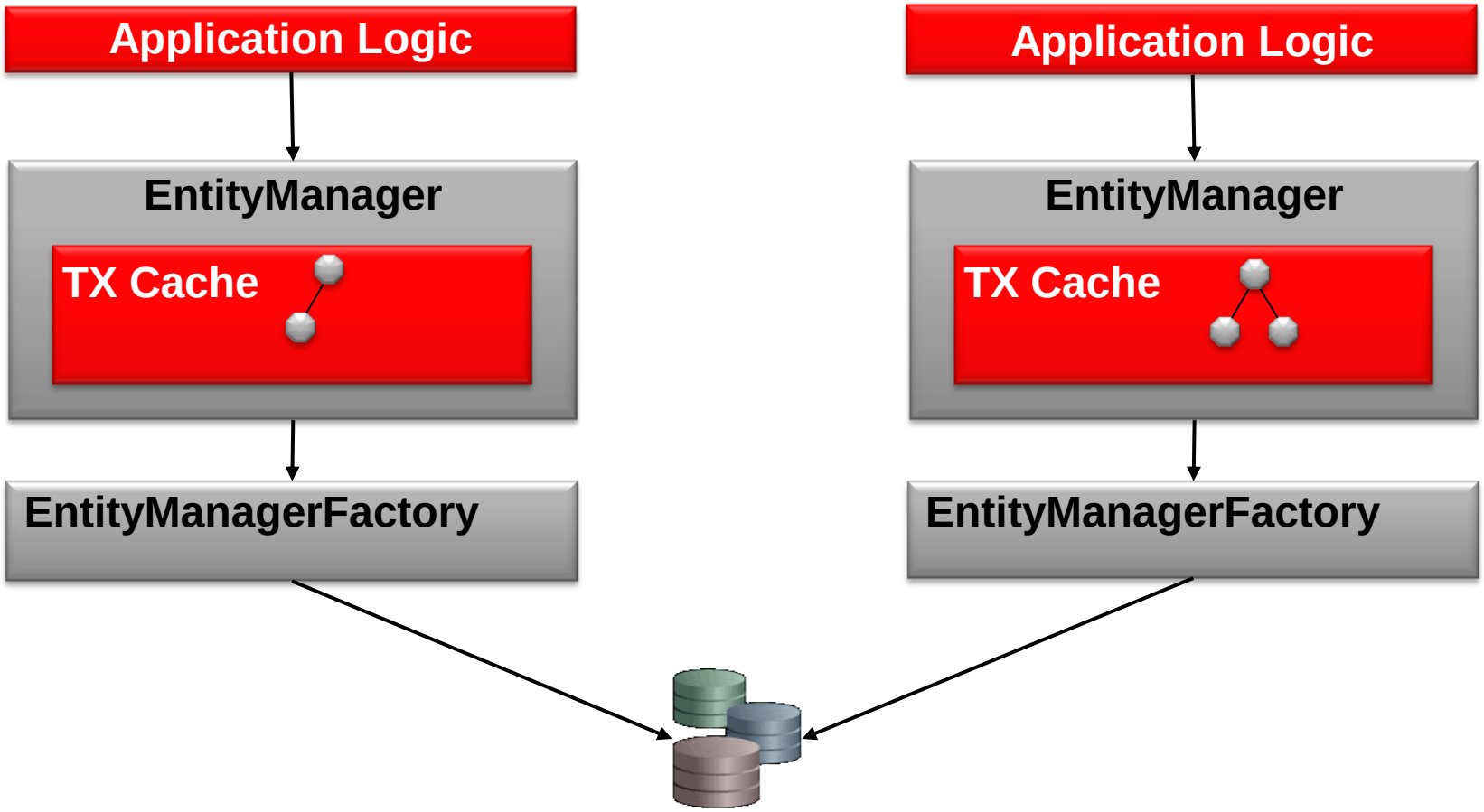
EclipseLink in a Cluster



Traditional Approaches to Scaling JPA

- Prior to TopLink Grid, there were two strategies for scaling EclipseLink JPA applications into a cluster:
  - Disable Shared Cache
  - Cache Coordination—communicate changes via messaging

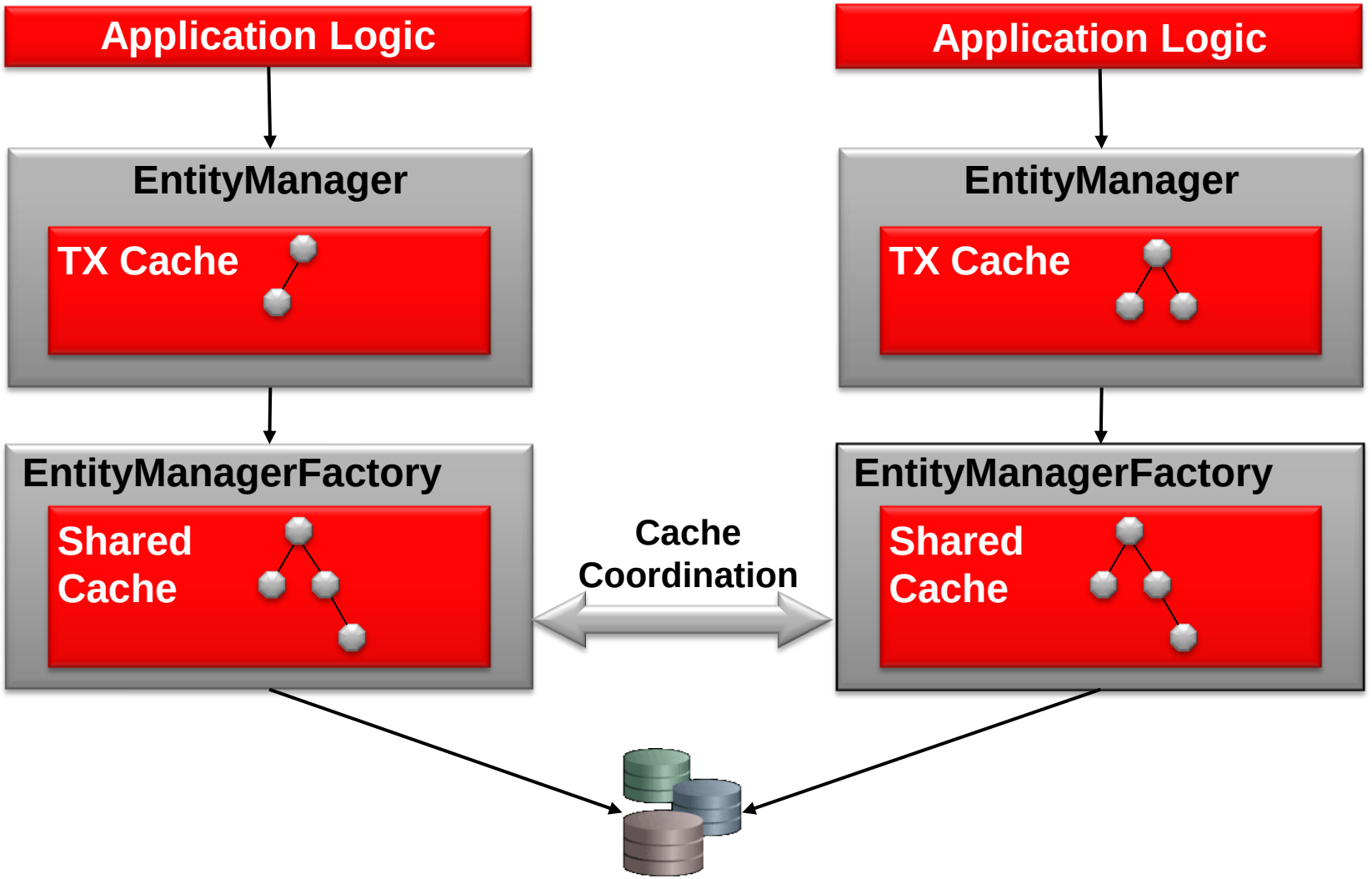
Strategy 1: Disable Shared Cache



Disable Shared Cache

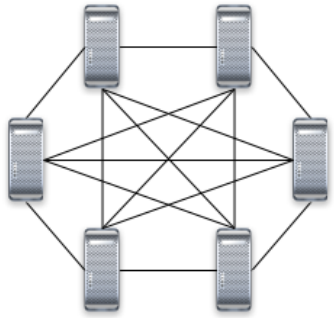
- Ensures all nodes have coherent view of data.
  - Database is always right
  - Each transaction queries all required data from database and constructs Entities
- No inter-node messaging
- Memory footprint of application increases as each transaction has a copy of each required Entity
- Every transaction pays object construction cost for queried Entities.
- Database becomes bottleneck

Strategy 2: Cache Coordination



Cache Coordination

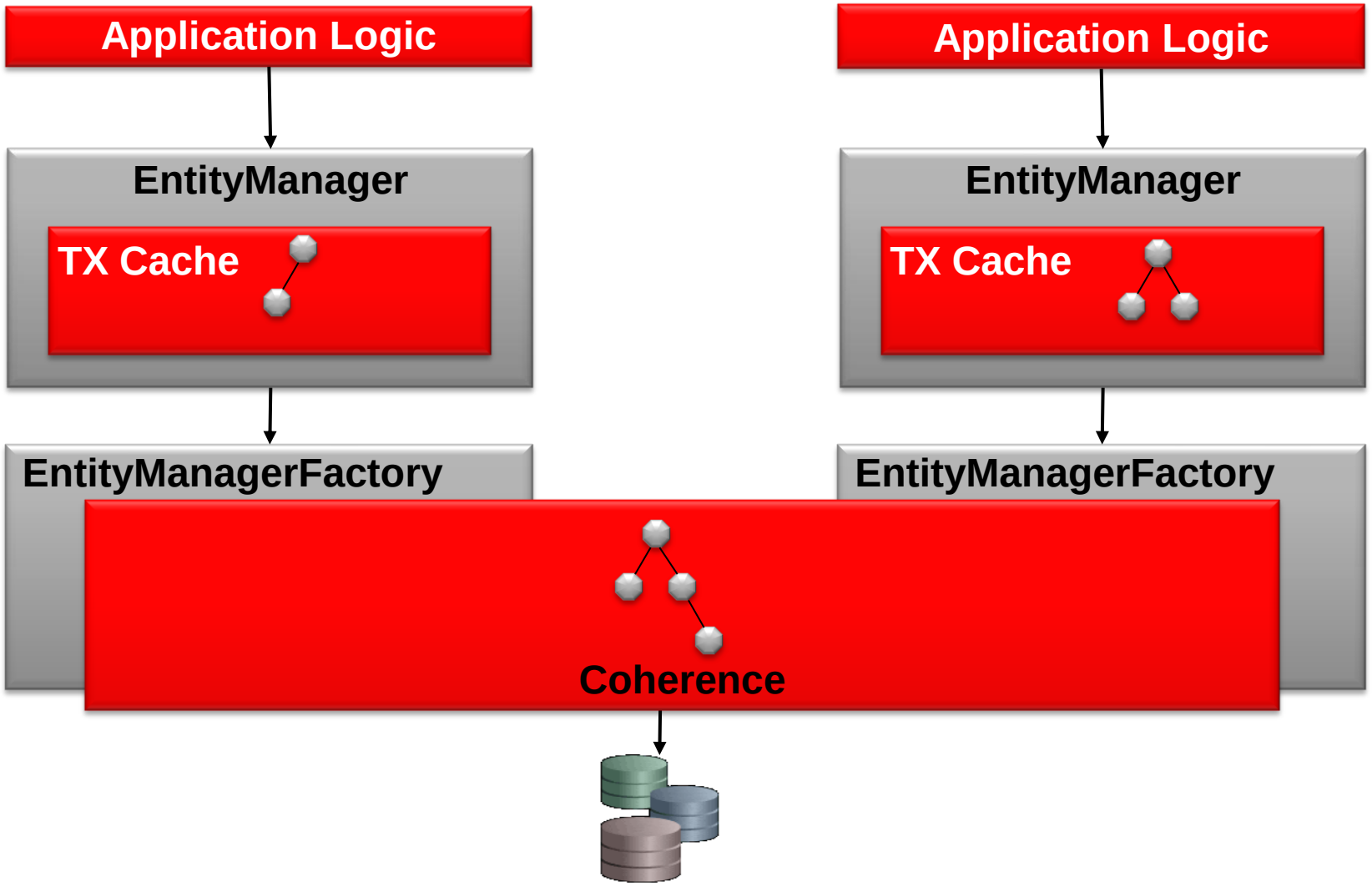
- Ensures all nodes have coherent view of data.
  - Database is always right
  - Fresh Entities retrieved from shared cache
  - Stale Entities refreshed from database on access
- Creation and/or modification of Entity results in message to all other nodes
- Cost of coordinating 1 concurrent update per node is $O(n^2)$ as all nodes must be informed—cost of communication and processing may eventually exceed value of caching
- Shared cache size limited by heap of each node



Introducing TopLink Grid

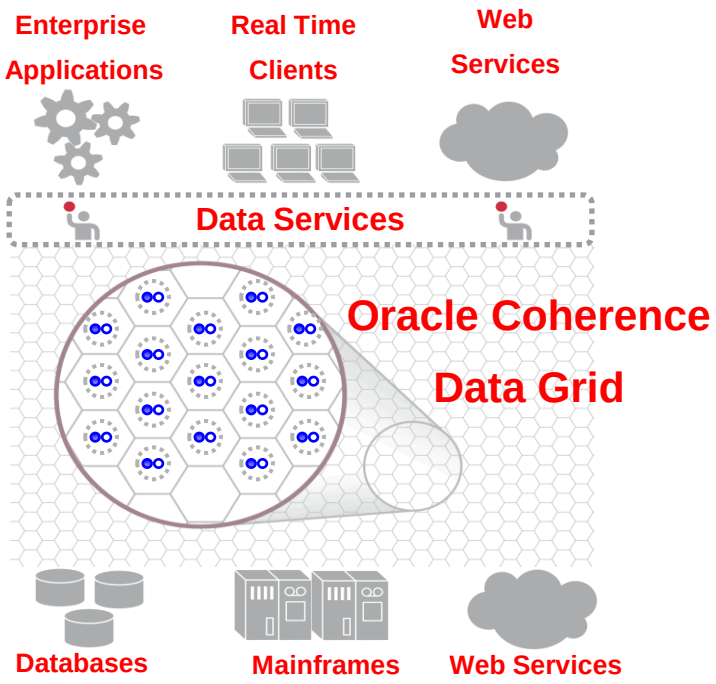
- TopLink Grid allows Java developers to transparently leverage the power of the Coherence data grid
- TopLink Grid combines:
  - the simplicity of application development using the Java standard Java Persistence API (JPA) with
  - the scalability and distributed processing power of Oracle's Coherence Data Grid.
- Supports 'JPA on the Grid' Architecture
  - EclipseLink JPA applications using Coherence as a shared (L2) cache replacement along with configuration for more advanced usage

TopLink Grid with Coherence Cache



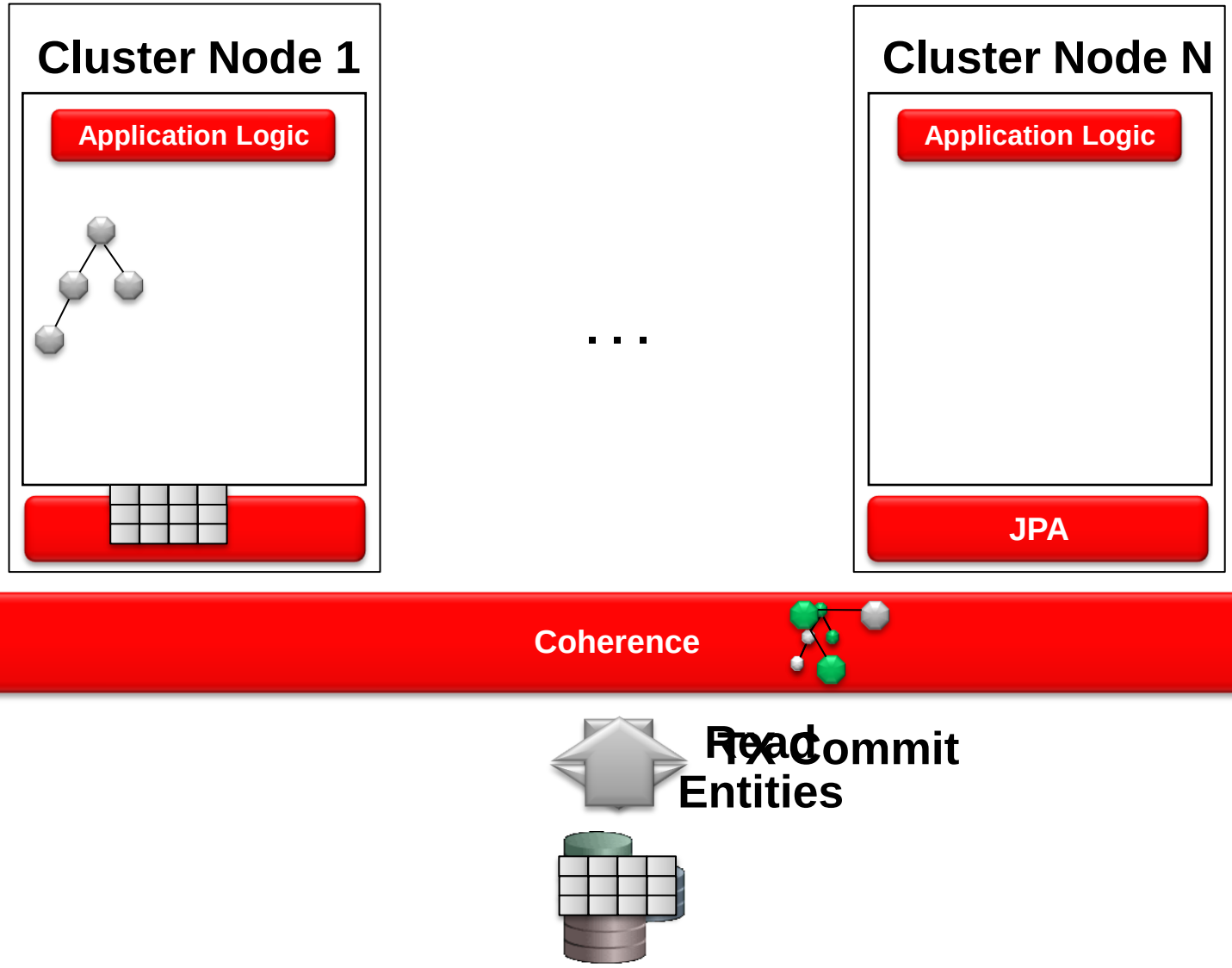
Oracle Coherence Data Grid

Distributed in Memory Data Management



- Provides a **reliable data tier** with a single, consistent view of data
- Enables dynamic data capacity including **fault tolerance** and load balancing
- Ensures that **data capacity scales** with processing capacity

JPA with Coherence

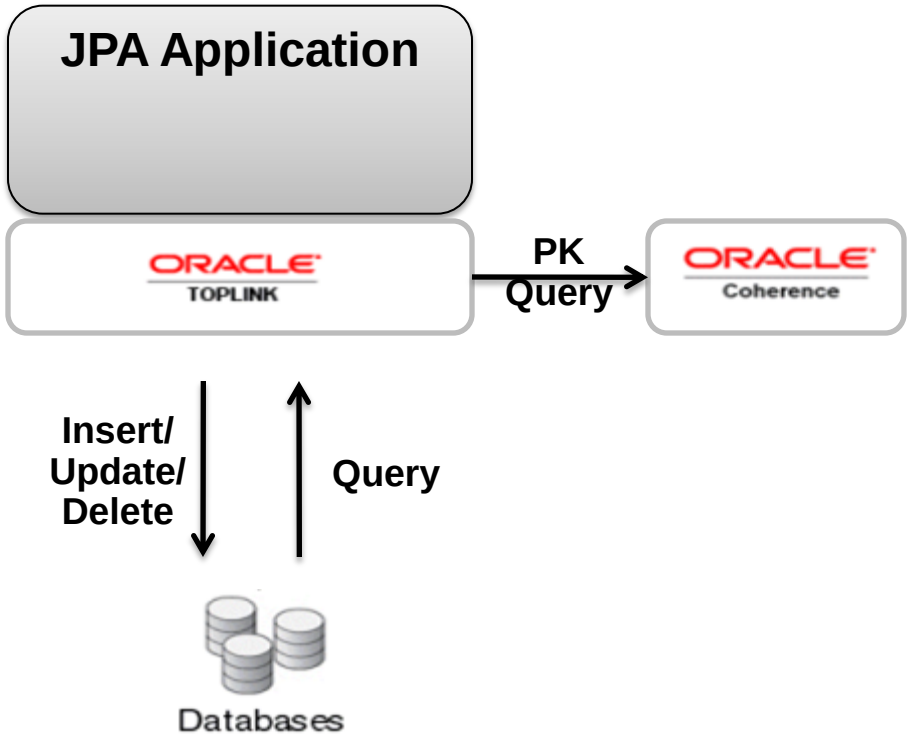


TopLink Grid—Configurations

- Grid Cache—Coherence as Shared (L2) Cache
  - Configurable per Entity type
  - Entities read by one grid member are put into Coherence and are immediately available across the entire grid
- Grid Read
  - All supported read queries executed in the Coherence data grid
  - All writes performed directly on the database by TopLink (synchronously) and Coherence updated
- Grid Entity
  - All supported read queries and all writes are executed in the Coherence data grid

Grid Cache ('Cache Aside')

- Reading:
  - Primary Key queries check Coherence first.
  - If found in Coherence, Entity is returned.
  - If not found the database is queried.
  - Entities queried from database are put() into Coherence and returned to the application.
- Writing:
  - All inserts, updates, and deletes are directed to the database
  - On successful commit, Coherence is updated



Grid Cache—Leveraging Cache

- Cache is used when processing database results
- EclipseLink extracts primary keys from results and checks cache to avoid object construction.
- Even if a SQL query is executed, an object cache can still improve application throughput by eliminating object construction costs for cached Entities

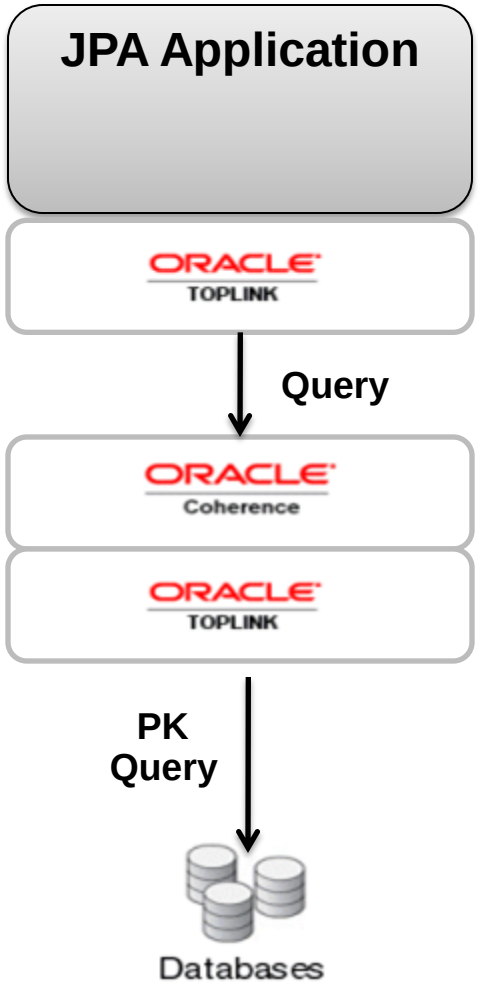
Grid Entity—Reading

- Primary key queries result in get() on Coherence
- JPQL queries, e.g.,

  ```
  Select e from Employee E
  ```

  are translated to Filters and executed in Coherence
- The database is not queried by EclipseLink.

CacheLoaders *may* be configured to query database with PK query on cache miss



Limitations in TopLink 11gR1

- TopLink Grid 11gR1 Supports single Entity queries with constraints on attributes, e.g.:

```
select e from Employee e where e.name = 'Joe'
```

- Complex queries must be executed on database:
  - Multi-Entity queries or queries that traverse relationships ('joins'), e.g.:

```
select e from Employee e
where e.address.city = 'Bonn'
```

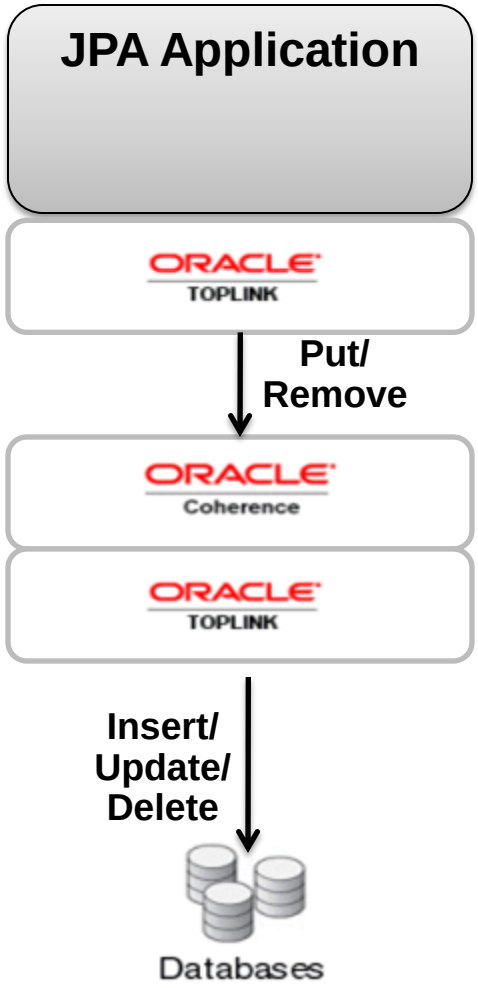
  - Projection (Report) queries, e.g.:

```
select e.name, e.city from Employee e
```

Grid Entity—Writing

- Applications commit JPA transactions with new, deleted, or modified Entities
- EclipseLink put()s all new and updated Entities into Coherence and remove()s deleted Entities.

  CacheStores *may* be configured to write cache changes to the database using TopLink Grid

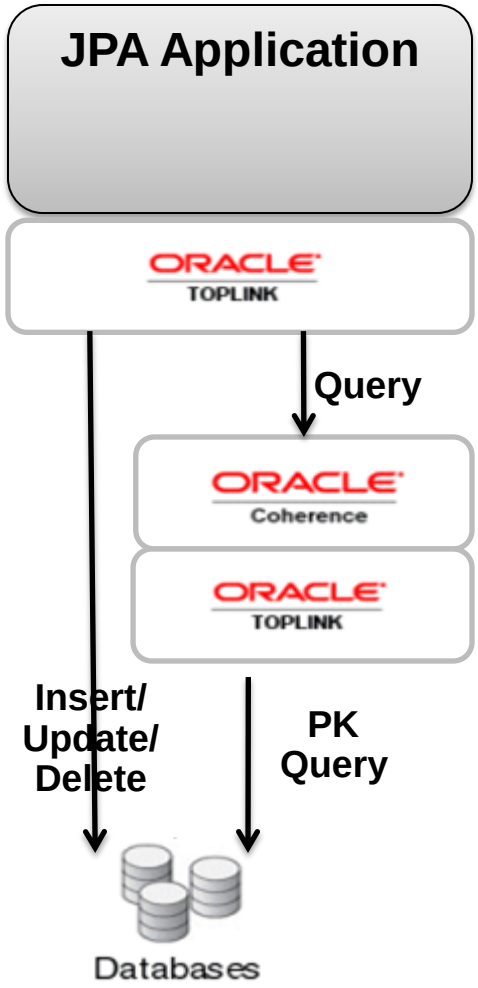


Grid Read

- All writes performed directly on database.
- Primary key queries result in get() on Coherence
- JPQL queries, e.g.,

  `Select e from Employee E`

  are translated to Filters and executed in Coherence
- The database is not queried by EclipseLink.
- CacheLoaders *should* be configured to query database with PK query on cache miss



Grid Enabling JPA Entities

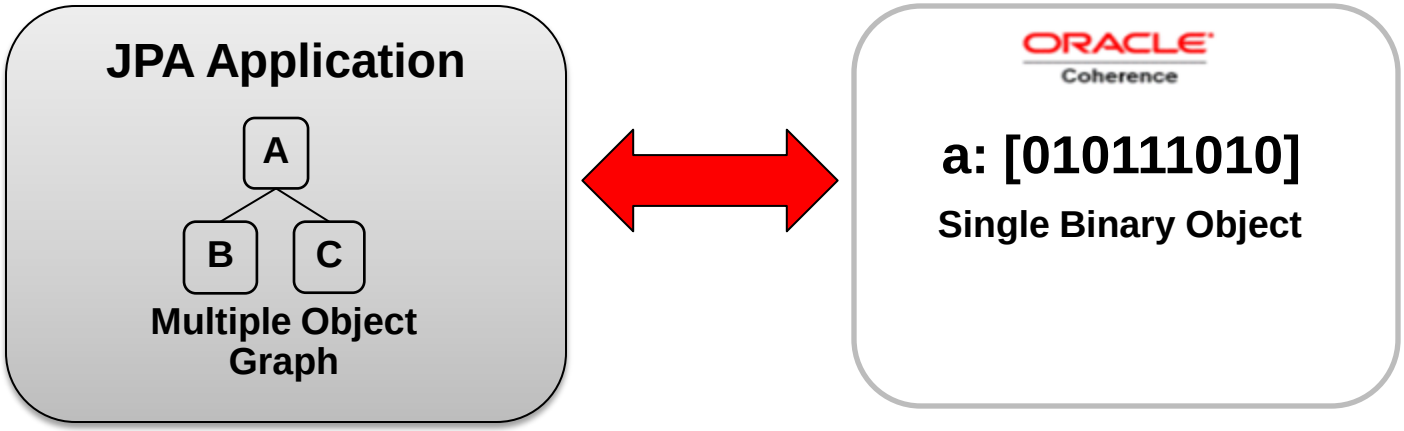
- A single annotation is added to an Entity to enable Coherence usage, e.g.,

```
@Entity
@Customizer(CoherenceReadCustomizer.class)
public class Employee implements Serializable {
...
}
```

- Standard Coherence cache configuration applies
  - POF, ExternalizableLite, and Serializable Entities supported

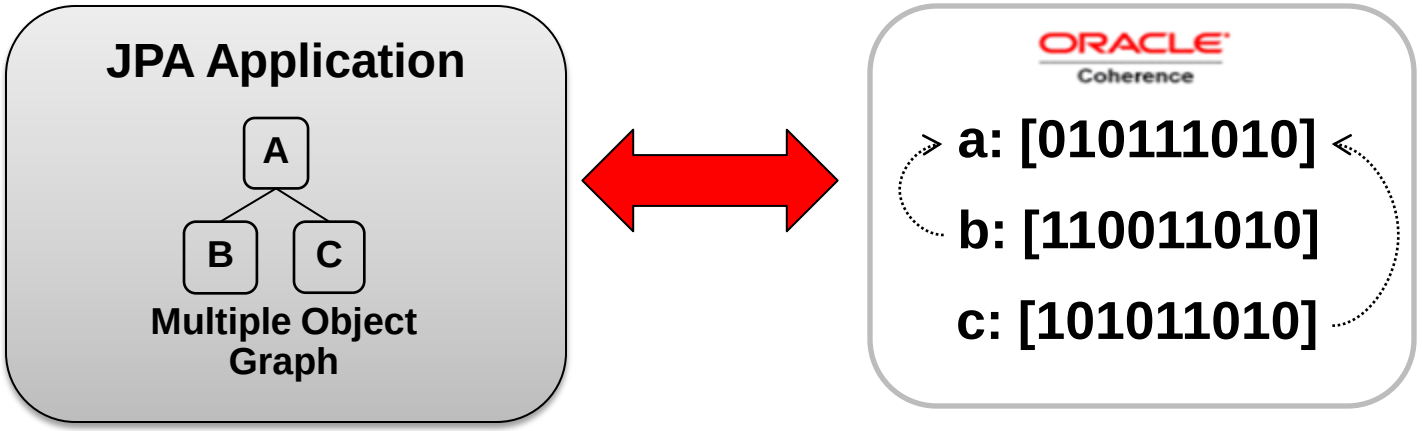
TopLink Grid Relationship Support

- Coherence does not provide support for the serialization of complex graphs across caches.
  - Coherence serializes objects/object graphs and places the results in to a single cache under a key.
  - Can't query or lazy load individual objects from the graph—all or nothing



TopLink Grid Relationship Support

- TopLink Grid 11gR1 does support storage of complex graphs of Entities with each Entity type stored in a corresponding Coherence cache.
  - *Relationship* information *is stored* into Coherence *and reconstituted* upon retrieval
  - Can query for objects of class A, B, or C
  - Lazy and eager relationships are supported—even to db data!



TopLink and Coherence: Objects, not Data

- TopLink's shared cache is an *object cache*
  - Cache hits do not incur object construction costs—typically an expensive part of object/relational mapping
- Coherence caches serialized objects
- Using Coherence as TopLink's shared object cache
  - Only incurs serialization cost, not object construction
  - Can leverage POF serialization for maximum performance
- TopLink Grid pays the object construction cost only once and eliminates it for each cluster member

How is TopLink Grid different from Hibernate with Coherence?

- Hibernate does not cache objects, it caches *data rows* in Coherence
- Using Coherence as a cache for Hibernate
  - Every cache hit incurs both object construction *and* serialization costs
  - Worse, object construction cost is paid by every cluster member for every cache hit
- Hibernate only uses Coherence as a cache—TopLink Grid is *unique* in supporting execution of queries against Coherence which can significantly offload the database and increase throughput

Summary

- TopLink supports a unique range of strategies for scaling JPA applications
- **TopLink Grid provides**:
  - **An easy way for JPA developers to scale out their Java EE applications**
  - 'JPA on the Grid' functionality to support scaling JPA applications with Coherence
  - Support for caching Entities with relationships in Coherence


Q&A