



ORACLE®

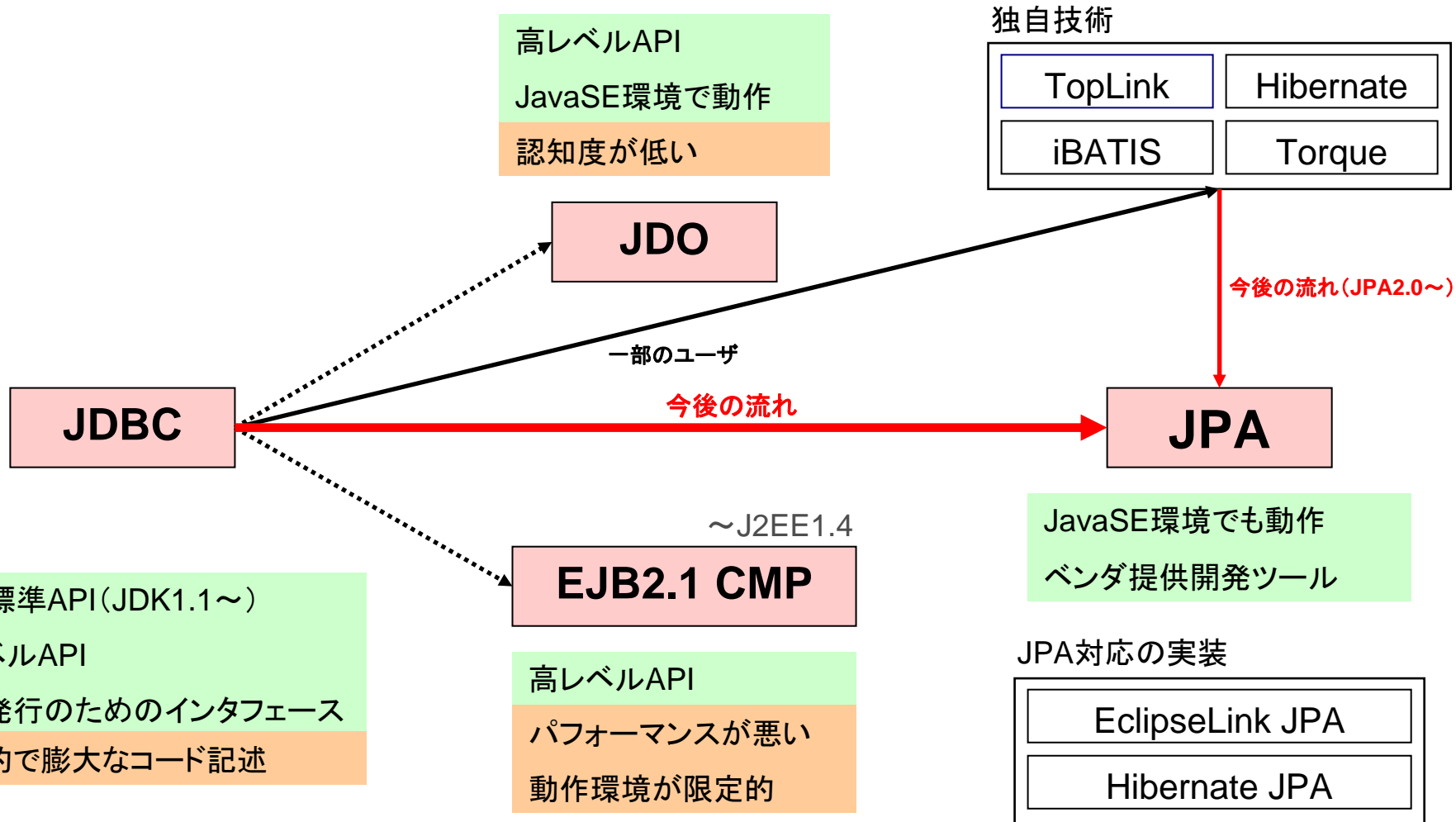
Java Persistence 2.0とEclipseLinkについて

日本オラクル株式会社

Agenda

- Java Persistence API (JPA)
 - DBアクセス技術の変遷
 - JPA概要
 - JPAアーキテクチャ
 - EclipseLink
 - EclipseLink JPA拡張機能
 - Java Persistence 2.0
- Oracle TopLink 11g
 - Oracle TopLink概要
 - TopLink Grid
- まとめ
- Appendix

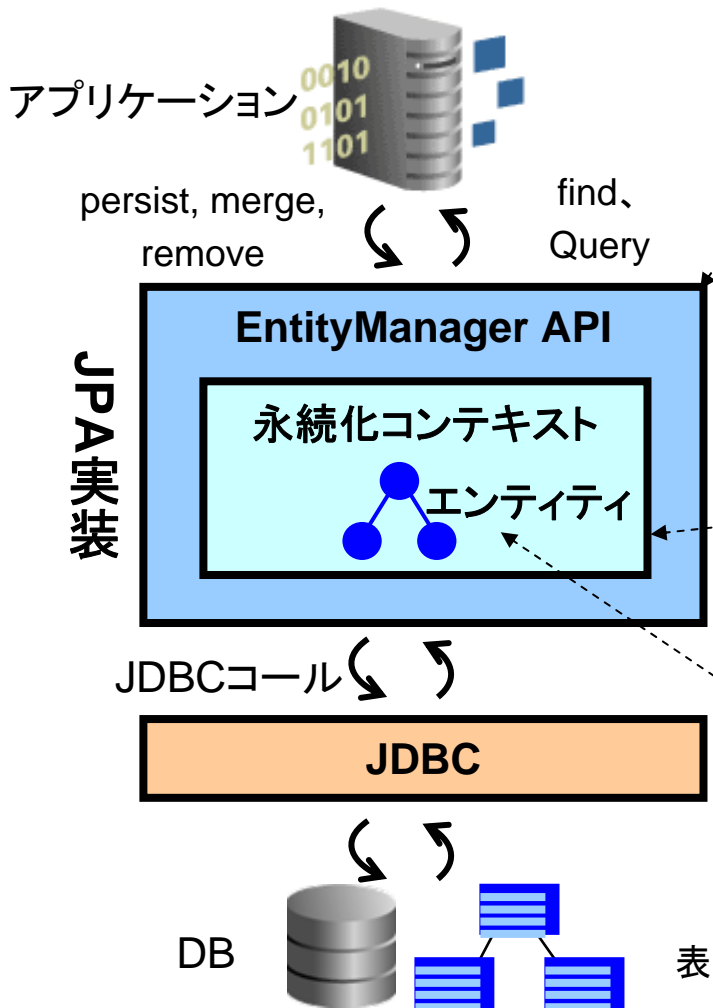
DBアクセス技術の変遷



JPA概要

- Java-RDB間の永続化管理のための標準API
- EJB3.0仕様の一部として策定されたが、現在はJPA仕様書として別ドキュメントとなっている
- TopLink EssentialsがJPAの参照実装
 - Oracleが提供しているO/Rマッピングツール「Oracle TopLink」のソースコードの一部を提供
 - JPA1.0仕様策定にはOracle TopLink開発チーム「Mike Keith」がスペックリードとして参加
- 機能的な特徴
 - POJOベース
 - JavaEEコンテナ外で使用可能
 - アノテーションによるO/Rマッピング定義
 - クエリ機能の大幅な拡張
 - EJB QLの拡張: 明示的なJoin、bulk update/delete、副問合せ、group by
 - Native SQL: SQLをそのまま使用可能

JPAアーキテクチャ



EntityManager API

- ・エンティティおよび永続化コンテキストを操作するためのAPI

永続化ユニット

- ・persistence.xmlに記述する
- ・DB接続情報や、JPAプロバイダの設定など構成情報を記述する

永続化コンテキスト

- ・永続化ユニットの実行時オブジェクト
- ・エンティティの管理を行う

エンティティ

- ・DBの表にマッピング可能な永続化Javaオブジェクト
- ・エンティティ・クラスとして実装される

EclipseLink

- **Eclipse Persistence Services Project (EclipseLink)**

- JPA、JAXB、SDO等の実装製品を提供するプロジェクト
- オラクルがプロジェクトリーダー
- EclipseLink JPAがJava Persistence 2.0の参照実装

The screenshot shows the EclipseLink website homepage. At the top, there is a navigation menu with links for Home, Downloads, Users, Members, Committers, Resources, Projects, and About Us. A Google Custom Search box is located on the right side of the header. The main content area features the EclipseLink logo, a brief description of the project, and a 'Getting Started' section with links to various persistence services. On the right side, there are four vertical buttons for JPA, MOXy (JAXB), SDO, and DBWS, each with a corresponding icon. The left sidebar contains a 'Project Summary' section with links to JPA, MOXy, SDO, and DBWS, followed by a 'Documentation' section with links to FAQ, User Guide, Search User Guide, API, API (latest), EclipseLink Wiki, and Search Wiki. At the bottom of the sidebar, there is a 'Downloads' section with links to 1.1.2, Previous Releases, and Milestones.

Home Downloads Users Members Committers Resources Projects About Us

Google™ Custom Search Search

eclipse link

The Eclipse Persistence Services Project (EclipseLink) delivers a comprehensive open-source Java persistence solution. EclipseLink focuses on standards with advanced features, performance and scalability for enterprise software developers across data sources, formats, and containers.

Getting Started

Select a persistence service and learn more about how to get started developing with EclipseLink.

Object-Relational	EclipseLink JPA
Object-XML Binding	EclipseLink MOXy
Service Data Objects	EclipseLink SDO
Web Services for RDBMS	EclipseLink DBWS

JPA

MOXy (JAXB)

SDO

DBWS

EclipseLink Project >>

Project Summary

- JPA
- MOXy
- SDO
- DBWS

Documentation >>

- FAQ
- User Guide
- Search User Guide
- API
- API (latest)
- EclipseLink Wiki
- Search Wiki

Downloads >>

- 1.1.2
- Previous Releases
- Milestones

EclipseLink JPA拡張機能

- 拡張機能
 - 追加アノテーション、persistence.xmlへ追加プロパティを指定することで使用可能。
 - 拡張機能例
 - マッピングのための追加アノテーション
 - キャッシュ
 - ロギング
 - データロック(悲観排他制御)
 - ストアド・プロシージャ・クエリ
 - スキーマ生成
 - クエリ・ヒント

TopLink JPA(EclipseLink JPA) 拡張プロパティ

[http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_\(ELUG\)](http://wiki.eclipse.org/Using_EclipseLink_JPA_Extensions_(ELUG))

Java Persistence 2.0

- 主な新機能
 - Locking and Concurrency
 - Bean Validation
 - Criteria Query

Java Persistence 2.0

- Locking and Concurrency
 - JPA1.0まではOptimistic Lockingのみのサポート
 - @Versionを付けたフィールドとバージョン用フィールドに対応するRDBの列を使った楽観排他制御を行う。更新に失敗したトランザクション側ではOptimisticLockExceptionがthrowされる。
 - lockメソッドで指定するLockModeTypeは下記の2種類
 - READ
 - WRITE



- Java Persistence 2.0で追加されるLockMode
 - OPTIMISTIC(上記READと同じ)
 - OPTIMISTIC_FORCE_INCREMENT(上記WRITEと同じ)
 - PESSIMISTIC_READ
 - PESSIMISTIC_WRITE
 - PESSIMISTIC_FORCE_INCREMENT

Java Persistence 2.0

- Bean Validation (JSR-303)
 - Beanのメンバ変数の値検証を宣言的に行う仕組み
 - Hibernate Validatorが参照実装
 - Java Persistence 2.0ではLifecycleイベント (Prepersist, PreUpdate, Preremove) と連動したValidationをサポート
 - 動作モードはvalidation-mode elementをpersistence.xmlで指定
 - AUTO (デフォルト)
 - CALLBACK
 - NONE

Bean Validationの例

```
@Entity public class Employee {  
    @Id Integer empld;  
    @Length(max=50) String name;  
    @Size(max=100) String address;  
    @Email String email;  
}
```

Java Persistence 2.0

- Criteria API

- クエリーを組み立てるためのObject-based API

New

- 永続化ユニットの管理対象クラスのMetamodelベース

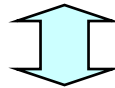
- Metamodelは型の安全性のために導入されたインタフェース

- 下記例で“_”が付いたクラスを自動生成するための仕組みを提供予定

- 従来から利用されている文字列ベースのCriteria APIもサポート

JPQLの例

```
SELECT c.name  
FROM Customer c JOIN c.orders o  
WHERE o.product.productType = 'printer'
```



Criteria APIの例

```
QueryBuilder qb = em.getQueryBuilder();  
CriteriaQuery cq = qb.create();  
Root<Customer> c = cq.from(Customer.class);  
Join<Customer, Order> o = c.join(Customer_.orders);  
cq.where(qb.equal(o.get(Order_.product).get(Product_.productType), "printer"))  
.select(c.get(Customer_.name));
```

Oracle TopLink 11g

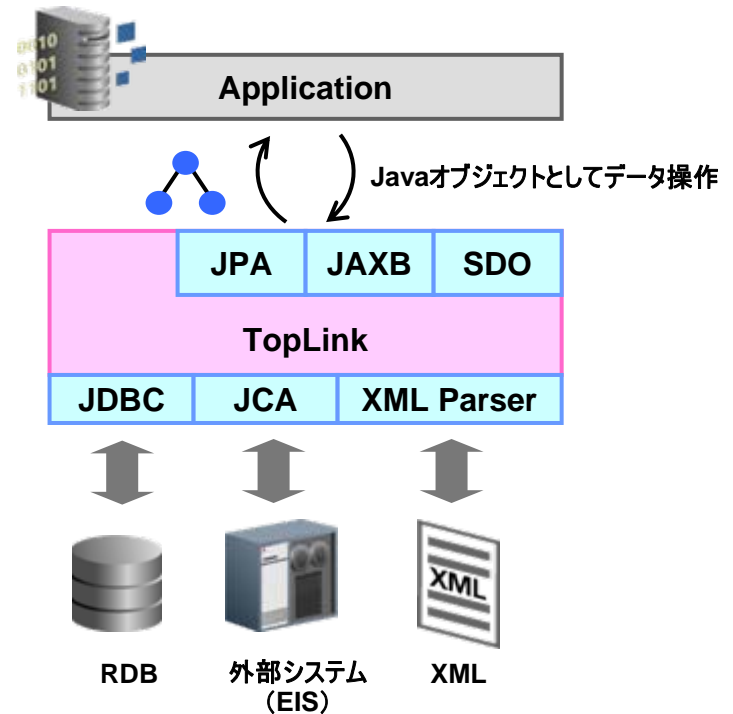


Oracle TopLink

- Javaオブジェクトの永続化フレームワーク
 - マッピング機能
 - オブジェクト/リレーショナルマッピング
 - オブジェクト/XMLマッピング
 - EISマッピング
 - オブジェクトキャッシュ
 - Oracle DB固有機能の利用
 - 商用製品として10年以上の実績
- Oracle TopLink 11g (11.1.1.0.1)
 - オープンソースのEclipseLink 1.0.1がベース
 - 既存アプリのためのTopLinkライブラリ
 - JPA拡張機能
 - TopLink Grid (Coherence連携機能)

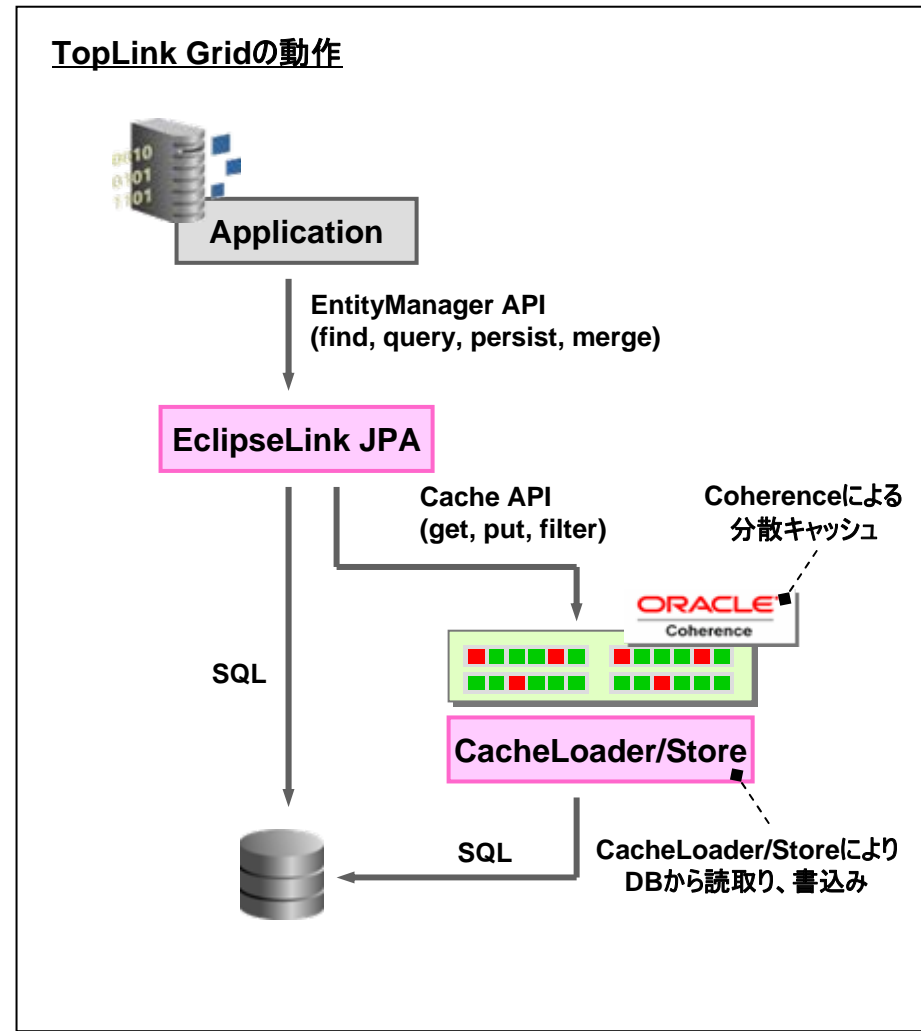
New

Oracle TopLinkの概要



TopLink Grid概要

- TopLink Grid
 - JPAのデータストアとしてOracle Coherenceと連携する機能
- 利用メリット
 - アプリケーションサーバー上のメモリにキャッシュされているデータへの高速アクセス
 - DBの負荷軽減
 - Coherence分散キャッシュの機能（データ容量の拡張性、クエリの並列処理）
- 動作パターン
 - Coherence Read
 - JPAの読み取り (find, Query) をCoherenceにリダイレクト
 - 書き込みはJPAで実行
 - Coherence Read/Write
 - JPAの読み取り、書き込みをCoherenceにリダイレクト
 - Coherence L2 Cache



TopLink Grid構成

- Entityのアノテーションで設定

- Coherence Read

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

- Coherence Read/Write

```
@Entity
@Customizer(CoherenceReadWriteCustomizer.class)
public class Person implements Serializable {...}
```

- Coherence L2 Cache

```
@Entity
@CacheInterceptor(value=CoherenceInterceptor.class)
public class Person implements Serializable {...}
```

- Persistence.xmlで設定

```
<property name="eclipselink.descriptor.customizer.エンティティ"
value="oracle.eclipselink.coherence.integrated.config.カスタマイザ"/>
```

- ◇ Coherenceが提供する最適化Serializationフレームワークを利用することが可能です。
- ◇ 必要なクラスはcoherence-eclipselink.jar内にあります。coherence-eclipselink.jarをクラスパスに追加します。
- ◇ coherence-eclipselink.jarはTopLink用のzipを解凍した中のjlibフォルダ内にあります。

TopLink Grid Coherence Read

- Entityオブジェクトの読み取り動作イメージ (find)
 - CoherenceReadCustomizer指定
 - CacheLoader構成
 - 実装クラス「oracle.eclipselink.coherence.integrated.EclipseLinkJPACacheLoader」

```
Person person = em.find(Person.class, personPK); // personPKはPersonクラスの@IdClassのインスタンス
```

変換

```
NamedCache cache = CacheFactory.getCache("Person");  
Person person = (Person)cache.get(personPK);
```

キャッシュにデータが無い場合

```
EclipseLinkJPACacheLoader.load(personPK)
```

loadの内部実装

```
EntityManager.find(Person.class, personPK)
```

CoherenceのCacheをupdate

```
Person person = (Person)cache.put(personPK, person);
```

- ◇ JDBCのSQL実行がキャンセルされ、Coherenceのキャッシュからデータを取得する。
- ◇ Coherenceのキャッシュにデータが無い場合にCacheStoreのloadがコールされ、EntityManagerのfindでデータを取得する。
- ◇ 取得したデータはCoherenceのキャッシュに格納する。

TopLink Grid Coherence Read

- Entityオブジェクトの読み取り動作イメージ(JPQL)
 - CoherenceReadCustomizer指定
 - CacheLoader構成
 - 実装クラス「oracle.eclipselink.coherence.integrated.EclipseLinkJPACacheLoader」

```
Query query = em.createQuery("select p from Person p where p.surname=:param");  
query.setParameter("param", "hashimoto");  
List<Person> results = query.getResultList();
```



変換

```
NamedCache cache = CacheFactory.getCache("Person");  
Set result = cache.entrySet(new EqualsFilter("getSurname", "hashimoto"));
```

- ◇ JPAのクエリーに対応したCoherenceのFilterが生成され、Coherenceのキャッシュからデータを取得する。
- ◇ Coherenceのキャッシュにデータが無い場合に何もしない。
- ◇ あらかじめCoherenceのキャッシュにデータをロードしておく必要がある。

TopLink Grid Coherence Read/Write

- Entityオブジェクトの書き込み動作イメージ
 - CoherenceReadWriteCustomizer指定時
 - CacheStore構成
 - 実装クラス「oracle.eclipselink.coherence.integrated.EclipseLinkJPACacheStore」

```
em.getTransaction().begin();  
em.persist(person); // 又はem.merge(person)  
em.getTransaction().commit();
```



CoherenceのCacheをupdate (カスタマイザの挙動)

```
NamedCache cache = CacheFactory.getCache("Person");  
cache.put(person.getKey(), person);
```



CacheStoreのstoreメソッド呼び出し (非同期連携可能)

```
EclipseLinkJPACacheStore.store(personPK)
```



storeの内部実装

```
EntityManager.merge(Person.class, personPK)
```



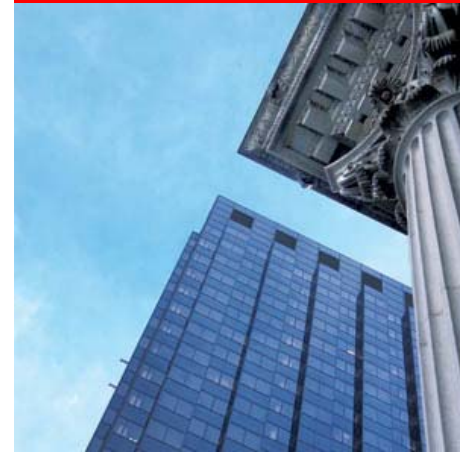
通常のJPA実装の動き

```
INSERT INTO PERSON ... 又は UPDATE PERSON ...
```

EclipseLink JPAとTopLink Gridの使い分け

- 使い分け例
 - 複雑なクエリー
 - EntityManager.createNativeQuery()でNativeなSQLを実行
 - Criteria APIを利用 (Java Persistence 2.0以降)
 - JDBC APIでNativeなSQLを実行
 - 通常クエリー
 - 事前にキャッシュデータをロード可能
 - 更新処理が同一レコードに短期集中する事が予想できるテーブル
 - Coherence Read/Write
 - データ量の多いバッチ処理
 - Coherence Read/Write
 - 上記以外
 - Coherence Read
 - 事前にキャッシュデータをロード不可能
 - L2 Cache

まとめ



まとめ

- Java Persistence API (JPA)
 - JPAが今後のDBアクセス技術の主流
 - EclipseLink JPAが参照実装
- Oracle TopLink 11g
 - EclipseLinkベース
 - 多数の拡張機能
 - TopLink Grid
 - JPAをインタフェースとしたOracle Coherenceとのシームレスな連携