PostgreSQL
From a Java Enterprise Point of View

Manipulating complex data models using component or object oriented methods

Jesper Pedersen
Principal Software Engineer
jesper.pedersen@jboss.org

Agenda

- Introduction
- The Java Enterprise platform
- Enterprise JavaBeans Query Language (EJB/QL)
- SQL
- Performance
- Conclusion

Introduction

- **Goal of this presentation**
  - **Introduction to the Java Enterprise platform**
  - **Especially to its persistence mechanisms**
    - Enterprise JavaBeans 2.1 (Component)
    - Enterprise JavaBeans 3.0 (Object)
  - **Ideas for future enhancements**
    - The Java Enterprise implementation
      - JBoss Application Server
      - Hibernate
    - PostgreSQL
    - PostgreSQL / JDBC

The Java Enterprise platform

- Java Enterprise Edition is a widely used platform for server programming in the Java programming language
- Extends the functionality of Java Standard Edition
  - standard based components
  - fault-tolerant applications
  - distributed applications
  - multi-tier applications
- Write once – run anywhere
- Versions
  - J2EE 1.2 (December 12, 1999)
  - J2EE 1.3 (September 24, 2001)
  - J2EE 1.4 (November 11, 2003)
  - Java EE 5 (May 11, 2006)

The Java Enterprise platform

- **Specifications**
  - **Large number of specifications in different areas**
  - **Web**
    - Servlet, Java ServerPages (JSP), Java ServerFaces (JSF)
  - **Business layer**
    - Enterprise JavaBeans (Session, MessageDriven)
  - **Persistence layer**
    - Enterprise JavaBeans (Entity)
  - **Integration**
    - Web services
    - Java Connector Architecture
  - **Security**
  - **Transaction**
    - Container managed or user managed
    - Required, RequiresNew, Mandatory, Supports, NotSupported, Never

The Java Enterprise platform

- Specifications (cont.)
  - All specifications developed under the Java Community Process (JCP)
  - Reference implementation (RI) provided by each specification committee
  - Technology Compatibility Kit (TCK) must also be provided
- Implementations
  - JBoss
  - Oracle / BEA
  - IBM
  - Sun (Glassfish)
  - Apache (Geronimo)
  - and other vendors

The Java Enterprise platform

- Enterprise JavaBeans (Entity) provides an abstract schema representation of the data model
  - The EJB container provides the default mapping between Java types and the database
  - Local view: Only visible inside the container
  - Remote view: Visible outside the container
- Enterprise JavaBeans 2.1
  - Component based model
  - Glued together with meta-data
  - Lives in its own tier typically with a local view
  - Value Objects (VOs) provides a view of each entity for the tiers above
- Enterprise JavaBeans 3.0
  - Object based model
  - Detached – e.g. lives in all tiers
  - Business layer executes all interactive operations

The Java Enterprise platform

- **Java Enterprise Edition 1.4 (J2EE 1.4) specification**
- **Enterprise JavaBeans 2.1**
  - **If used at all – a lot projects use other persistence tools**
  - **Bean Managed Persistence (BMP)**
    - Do it yourself
    - Never used
  - **Container Managed Persistence (CMP)**
    - Let the container do the work for you
    - Container Managed Relationships (CMR)
      - Uni-directional or Bi-directional
      - One-to-One
      - One-to-Many
      - Many-to-One
      - Many-to-Many
    - Container can create all you need
      - CREATE, ALTER, DROP TABLE and INDEX

The Java Enterprise platform

- **Enterprise JavaBeans 2.1 CMP example**

```
public interface AccountLocalHome extends EJBLocalHome {
  AccountLocal create() throws CreateException;
  AccountLocal findByPrimaryKey(Long pk)
    throws FinderException;
}

public interface AccountLocal extends EJBLocalObject {
   Long getId();

   String getOwner();
   void setOwner(String owner);
   Double getBalance();
   void setBalance(Double balance);

   AccountCategoryLocal getAccountCategory();
   void setAccountCategory(AccountCategoryLocal a);

   AccountVO getVO();
}
```

The Java Enterprise platform

- **Enterprise JavaBeans 2.1 CMP example**

```
public abstract class AccountEJB implements EntityBean {
  public abstract Long getId();
  public abstract void setId(Long id);
  public abstract String getOwner();
  public abstract void setOwner(String owner);
  public abstract Double getBalance();
  public abstract void setBalance(Double balance);
  public abstract AccountCategoryLocal getAccountCategory();
  public abstract void setAccountCategory(AccountCategoryLocal a);

  public Long ejbCreate() throws CreateException {
    setId(Long.valueOf(System.nanoTime()));
    return null;
  }
  public void ejbPostCreate() throws CreateException {
  }
  ...
  public AccountVO getVO() {
    return new AccountVO(getId(), getOwner(), getBalance(),
                         getAccountCategory().getId());
  }
}
```

The Java Enterprise platform

- **Enterprise JavaBeans 2.1 CMP example**

`ejb-jar.xml:`

```
<entity>
  <ejb-name>Account</ejb-name>
  <local-home>AccountLocalHome</local-home>
  <local>AccountLocal</local>
  <ejb-class>AccountEJB</ejb-class>
  <persistence-type>Container</persistence-type>
  <prim-key-class>java.lang.Long</prim-key-class>
  <reentrant>false</reentrant>
  <cmp-version>2.x</cmp-version>
  <abstract-schema-name>Account</abstract-schema-name>
  <cmp-field >
    <field-name>id</field-name>
  </cmp-field>
  <cmp-field >
    <field-name>owner</field-name>
  </cmp-field>
  ...
  <primkey-field>id</primkey-field>
  ...
```

The Java Enterprise platform

- **Enterprise JavaBeans 2.1 CMP example**

```
ejb-jar.xml:
 <ejb-relation>
   <ejb-relation-name>accountcategory-account</ejb-relation-name>
   <ejb-relationship-role>
     <ejb-relationship-role-name>account-in-accountcategory</ejb-
relationship-role-name>
     <multiplicity>Many</multiplicity>
     <relationship-role-source>
       <ejb-name>Account</ejb-name>
     </relationship-role-source>
     <cmr-field>
       <cmr-field-name>accountCategory</cmr-field-name>
     </cmr-field>
   </ejb-relationship-role>
   <ejb-relationship-role >
     <ejb-relationship-role-name>accountcategory-has-accounts</ejb-
relationship-role-name>
     <multiplicity>One</multiplicity>
     <relationship-role-source>
       <ejb-name>AccountCategory</ejb-name>
     </relationship-role-source>
     <cmr-field >
       <cmr-field-name>accounts</cmr-field-name>
       <cmr-field-type>java.util.Collection</cmr-field-type>
     </cmr-field>
   </ejb-relationship-role>
 </ejb-relation>
```

The Java Enterprise platform

- **Java Enterprise Edition 5 (JEE 5) specification**
- **Enterprise JavaBeans 3.0**
  - **Plain Old Java Object (POJO) with annotations**
  - **Inheritance (per hierarchy, per subclass, per entity)**
  - **EntityManager controls all communication with the database**
    - createQuery(), createNamedQuery(), createNativeQuery()
    - find(), merge(), persist(), remove()
  - **Object/relational persistence and query service**
    - Foundation for the EntityManager
    - Provides functionality to create all you need (TABLE, INDEX and so on)
    - The JBoss EJB3 implementation uses Hibernate
  - **Relationships with other beans handled through annotations**
    - @OneToOne, @OneToMany, @ManyToOne, @ManyToMany
    - @JoinTable, @JoinColoumns, @JoinColoumn

The Java Enterprise platform

- **Enterprise JavaBeans 3.0 example**

```
@Entity
public class Account implements Serializable {
  @Id
  private long id;
  private String owner;
  private double balance;

  public Account() {
    id = System.nanoTime();
  }

  public String getOwner() {
    return owner;
  }

  public void setOwner(String owner) {
    this.owner = owner;
  }
}
```

The Java Enterprise platform

- **Enterprise JavaBeans 3.0 example**

```
@Stateless
@Remote(Bank.class)
public class BankBean implements Bank {
  @PersistenceContext
  private EntityManager em;

  public List<Account> listAccounts() {
    List accounts =
      em.createQuery(“SELECT a FROM Account a”);
    return accounts;
  }

  public Account openAccount(String owner) {
    Account account = new Account();
    account.setOwner(owner);
    em.persist(account);
    return account;
  }
}
```

The Java Enterprise platform

- **Enterprise JavaBeans 3.0 example**
  - **persistence.xml specifies the configuration**
    - description – a description
    - provider – persistence provider
    - transaction-type – JTA or RESOURCE_LOCAL
    - jta-data-source – JNDI name of JTA datasource
    - non-jta-data-source – JNDI name of non-JTA datasource
    - mapping-file – OR mapping
    - jar-file / class – Search list for entities
    - exclude-unlisted-classes – Only include listed classes
    - properties – Vendor specific properties

```
<?xml version=”1.0” encoding=”UTF-8”?>
<persistence xmlns=”http://java.sun.com/xml/ns/persistence”>
  <persitence-unit name=”bank-intro”/>
</persistence>
```

Enterprise JavaBeans Query Language

- The Enterprise JavaBeans Query Language (EJB/QL) defines a language to retrieve entities from the database
- EJB/QL uses the abstract persistence schemas of entity beans, including their relationships, for its data model
- The EJB/QL language is similar to SQL
  - SELECT DISTINCT OBJECT(*)
  - FROM Orders AS o, IN(o.lineItems) AS l
  - WHERE l.shipped = FALSE
- There are currently two versions being used
  - Enterprise JavaBeans Query Language 2.1
  - Enterprise JavaBeans Query Language 3.0

Enterprise JavaBeans Query Language

- Enterprise JavaBeans Query Language 2.1
- Basic selection support
  - Object – SELECT OBJECT(a) FROM ...
  - Single value – SELECT a.balance FROM ...
  - Aggregate function – SELECT AVG(a.balance) FROM ...
- Navigation on relationships
  - SELECT OBJECT(a) FROM Account a
  - WHERE a.accountCategory.name = 'Checking'
- Mapped to methods in the home interface(s)
  - javax.ejb.EJBLocalHome or javax.ejb.EJBHome
- Reserved identifiers
  - SELECT, FROM, WHERE, DISTINCT, OBJECT, NULL, TRUE, FALSE, NOT, AND, OR, BETWEEN, LIKE, IN, AS, UNKNOWN, EMPTY, MEMBER, OF, IS, AVG, MAX, MIN, SUM, COUNT, ORDER, BY, ASC, DESC, MOD

Enterprise JavaBeans Query Language

- **EJB/QL 2.1 examples**

```
SELECT DISTINCT OBJECT(o)
FROM Order o, IN(o.lineItems) l
WHERE l.product.productType = ‘office_supplies’

SELECT DISTINCT OBJECT(o1)
FROM Order o1, Order o2
WHERE o1.quantity > o2.quantity AND
      o2.customer.lastname = ?2 AND
      o2.customer.firstname= ?1

SELECT OBJECT(o)
FROM Order o, IN(o.lineItems) l
WHERE o.lineItems IS EMPTY

SELECT SUM(l.price)
FROM Order o, IN(o.lineItems) l
WHERE o.customer.lastname = ?2 AND
      o.customer.firstname = ?1
```

Enterprise JavaBeans Query Language

- **EJB/QL 2.1 examples**

```
ejb-jar.xml:
 <query>
   <query-method>
     <method-name>findQuantity</method-name>
     <method-params>
       <method-param>java.lang.String</method-param>
       <method-param>java.lang.String</method-param>
     </method-params>
   </query-method>
   <result-type-mapping>Local</result-type-mapping>
   <ejb-ql><![CDATA[
      SELECT DISTINCT OBJECT(o1)
      FROM Order o1, Order o2
      WHERE o1.quantity > o2.quantity AND
            o2.customer.lastname = ?2 AND
            o2.customer.firstname= ?1
            ]]></ejb-ql>
 </query>
```

Enterprise JavaBeans Query Language

- Enterprise JavaBeans Query Language 3.0
- Extension of the EJB/QL 2.1 standard
- New features
  - Bulk updates and deletes
  - JOIN operations
  - GROUP BY clause
  - HAVING clause
  - Projection
  - Subqueries
  - Dynamic queries
  - Named queries
  - Native Queries
  - Constructing new objects in SELECTs

Enterprise JavaBeans Query Language

- New reserved identifiers
  - UPDATE, DELETE, JOIN, OUTER, INNER, LEFT, GROUP, HAVING, FETCH, SET, UPPER, LOWER, TRIM, POSITION, CHARACTER_LENGTH, CHAR_LENGTH, BIT_LENGTH, CURRENT_TIME, CURRENT_DATE, CURRENT_TIMESTAMP, NEW, EXISTS, ALL, ANY, SOME

Enterprise JavaBeans Query Language

- **EJB/QL 3.0 example**

```
@NamedQuery(name=”findByBirthday”,
            queryString=”SELECT u FROM User u
                         WHERE u.birthday BETWEEN
                           :from AND :to
                         ORDER BY u.birthday”)

public List<User> findByBirthday(Date from, Date to)
{
   List<User> result =
     em.createNamedQuery(“findByBirthday”).
       setParameter(“from”, from).
       setParameter(“to”, to).
       getResultList();

   return result;
}
```

SQL

- All SQL statements are generated by the persistence framework in the EJB container
- We must provide as much information about the functionality of the database as possible
  - Which features are supported ? – row locking, subquery...
- The distribution of EJB/QL and therefore SQL is of course very specific to the application
- As a general rule
  - Enterprise JavaBeans applications are very SELECT heavy
  - Joins can occur between multiple tables
  - Performance can suffer – we'll get to this later

SQL

- **Tables / Indexes**

```
pgcon=> \d account
                                  Table "public.account"
     Column      |       Type       |                     Modifiers
-----------------+------------------+----------------------------------------------------
 id              | bigint           | not null default nextval('account_id_seq'::regclass)
 owner           | text             |
 balance         | double precision |
 accountcategory | bigint           |
Indexes:
    "pk_account" PRIMARY KEY, btree (id)
    "account_idx1" btree (owner)
    "account_idx2" btree (accountcategory)

pgcon=> \d accountcategory
                         Table "public.accountcategory"
 Column |  Type  |                         Modifiers
--------+--------+------------------------------------------------------------
 id     | bigint | not null default nextval('accountcategory_id_seq'::regclass)
 name   | text   |
Indexes:
    "pk_accountcategory" PRIMARY KEY, btree (id)
    "accountcategory_idx1" btree (name)
```

SQL

- **EJB 2.1 example**
  - **EJB/QL**
    - 1) SELECT DISTINCT OBJECT(*) FROM Account a WHERE a.owner = ?1
    - 2) SELECT DISTINCT OBJECT(*) FROM Account a WHERE a.accountCategory.name = ?1
  - **SQL**

```
1)
SELECT DISTINCT t0_a.id FROM account t0_a WHERE (t0_a.owner = ?)
SELECT owner, balance, accountcategory FROM account WHERE (id=?)
SELECT name FROM accountcategory WHERE (id=?)

2)
SELECT DISTINCT t0_a.id FROM account t0_a, accountcategory
t1_a_accountCategory WHERE (t1_a_accountCategory.name = ? AND
t0_a.accountcategory=t1_a_accountCategory.id)

SELECT id, owner, balance, accountcategory FROM account WHERE (id=?) OR
(id=?) OR ... OR (id=?)

SELECT name FROM accountcategory WHERE (id=?)
```

SQL

- **EJB 2.1 example**
  - **EXPLAIN ANALYZE**

```
1)
Unique  (cost=8.28..8.29 rows=1 width=8) (actual time=0.187..0.191 rows=1 loops=1)
   ->  Sort  (cost=8.28..8.28 rows=1 width=8) (actual time=0.184..0.186 rows=1 loops=1)
         Sort Key: id
         Sort Method:  quicksort  Memory: 25kB
         ->  Index Scan using account_idx1 on account t0_a  (cost=0.00..8.27 rows=1
   width=8) (actual time=0.165..0.168 rows=1 loops=1)
               Index Cond: (owner = 'Account #1'::text)
Total runtime: 0.249 ms

Index Scan using pk_account on account  (cost=0.00..8.27 rows=1 width=28) (actual
   time=0.062..0.065 rows=1 loops=1)
   Index Cond: (id = 1)
Total runtime: 0.116 ms

Seq Scan on accountcategory  (cost=0.00..1.02 rows=1 width=8) (actual time=0.023..0.026
   rows=1 loops=1)
   Filter: (id = 2)
Total runtime: 0.070 ms
```

SQL

- **EJB 2.1 example**
  - **EXPLAIN ANALYZE**

```
        2)

 Unique  (cost=51.20..53.70 rows=500 width=8) (actual time=5.169..7.447 rows=513 loops=1)
   ->  Sort  (cost=51.20..52.45 rows=500 width=8) (actual time=5.166..5.894 rows=513 loops=1)
         Sort Key: t0_a.id
         Sort Method:  quicksort  Memory: 49kB
         ->  Hash Join  (cost=1.04..28.79 rows=500 width=8) (actual time=0.102..4.244 rows=513 loops=1)
               Hash Cond: (t0_a.accountcategory = t1_a_accountcategory.id)
               ->  Seq Scan on account t0_a  (cost=0.00..19.00 rows=1000 width=16) (actual time=0.017..1.574
rows=1000 loops=1)
               ->  Hash  (cost=1.02..1.02 rows=1 width=8) (actual time=0.027..0.027 rows=1 loops=1)
                     ->  Seq Scan on accountcategory t1_a_accountcategory  (cost=0.00..1.02 rows=1 width=8)
(actual time=0.014..0.018 rows=1 loops=1)
                           Filter: (name = 'Checking'::text)
 Total runtime: 8.218 ms


 Bitmap Heap Scan on account  (cost=12.78..19.63 rows=3 width=36) (actual time=0.062..0.067 rows=3 loops=1)
   Recheck Cond: ((id = 1) OR (id = 3) OR (id = 5) OR AND SO ON...)
   ->  BitmapOr  (cost=12.78..12.78 rows=3 width=0) (actual time=0.052..0.052 rows=0 loops=1)
         ->  Bitmap Index Scan on pk_account  (cost=0.00..4.26 rows=1 width=0) (actual time=0.031..0.031 rows=1
loops=1)
               Index Cond: (id = 1)
         ->  Bitmap Index Scan on pk_account  (cost=0.00..4.26 rows=1 width=0) (actual time=0.008..0.008 rows=2
loops=1)
               Index Cond: (id = 3)
         ->  Bitmap Index Scan on pk_account  (cost=0.00..4.26 rows=1 width=0) (actual time=0.006..0.006 rows=2
loops=1)
               Index Cond: (id = 5)

...
```

Performance

- **Enterprise JavaBeans usage must be tuned**
  - **Out-of-the-box experience will not be optimal**
  - **Very application specific**
- **Use local interface where possible**
  - **Remote calls will need to serialize data**
- **Load strategies**
  - **EJB 3.0**
    - Controlled through “fetch” attribute
    - Lazy – only columns that are needed
    - Eager – load the data you will need at once
  - **EJB 2.1**
    - Vendor specific extensions to control data fetching
      - read-ahead (jbosscmp-jdbc.xml)
    - Multiple Value Objects per entity
      - Using different eager-load-group and lazy-load-groups

Performance

- **Enterprise JavaBeans 2.1**
  - **Instance pool**
    - Controlled through conf/standardjboss.xml
    - Or by making special container configurations
  - **Different commit options can be used in order to control the load/reload of data**
    - Option A
      - Caches a “ready” instance between transactions
      - Exclusive access
    - Option B
      - Caches a “ready” instance between transactions
      - Non-exclusive access
    - Option C
      - Doesn't cache a “ready” instance between transactions
    - Option D (Vendor specific)
      - Like Option A, but with a refresh rate – f.ex. 60 seconds

Performance

- **Enterprise JavaBeans 3.0**
  - **You need to go through the container configuration**
    - @org.jboss.ejb3.annotation.Pool
    - EJB 3: deploy/ejb3-interceptors-aop.xml
  - **Hibernate dialect**
    - org.hibernate.dialect.PostgreSQLDialect
    - Additional dialects will appear based on work by Simon Riggs, Diego Plentz and the pgsql-jdbc mailinglist
    - http://opensource.atlassian.com/projects/hibernate/browse/HHH-2942
  - **Hibernate options**
    - Go through the different options when outside container
      - hibernate.cache.provider_class
      - hibernate.cache.use_query_cache
      - and so on

Performance

- **PostgreSQL**
  - **Java Enterprise applications are just like any other applications**
    - Tables and indexes are just controlled from outside the database – this can be disabled if you really want to
  - **Standard tuning rules applies**
    - Run EXPLAIN ANALYZE of the generated queries to find hot-spots
  - **Check your postgresql.conf file**
    - Tune to your system setup
    - shared_buffers, temp_buffers, work_mem and so on
  - **Run benchmark that simulates your application**
    - Simulate your work-load (distribution of SQL statements)
    - Otherwise use standard benchmark suites
      - SPECjAppServer 2004
      - EAStress2004
      - Others (TPC-x)

Conclusion

- The Java Enterprise platform provides the developers with a way to work with databases without knowing the precise mapping between types
  - Easy to change database in the different environments
  - You need to know how a database work in order to gain maximum performance for your application
- The Enterprise JavaBeans 3.0 specification lets the developer work with annotations and with more control over the generated SQL
  - More simple development model than EJB 2.1
- PostgreSQL provides a strong foundation for Java Enterprise applications
  - Full Open Source stack making it possible to do research in all layers of the applications
  - High-performance as shown by SPECjAppServer2004
  - Strong community and commercial support

Questions ?

- What ... ?
- How ... ?

Links

- Java Enterprise
  - http://java.sun.com/javaee
- JBoss Application Server
  - http://www.jboss.org/jbossas
- JBoss Enterprise Application Platform
  - http://www.redhat.com/jboss/platforms/application/
- Enterprise JavaBeans 3.0 (JSR-220)
  - http://jcp.org/en/jsr/detail?id=220
- Hibernate
  - http://www.hibernate.org
- PostgreSQL
  - http://www.postgresql.org
- PostgreSQL / JDBC
  - http://jdbc.postgresql.org

Thank you !

- Feel free to contact me with questions
  - Jesper Pedersen
  - jesper.pedersen@jboss.org
  - http://www.jboss.org