

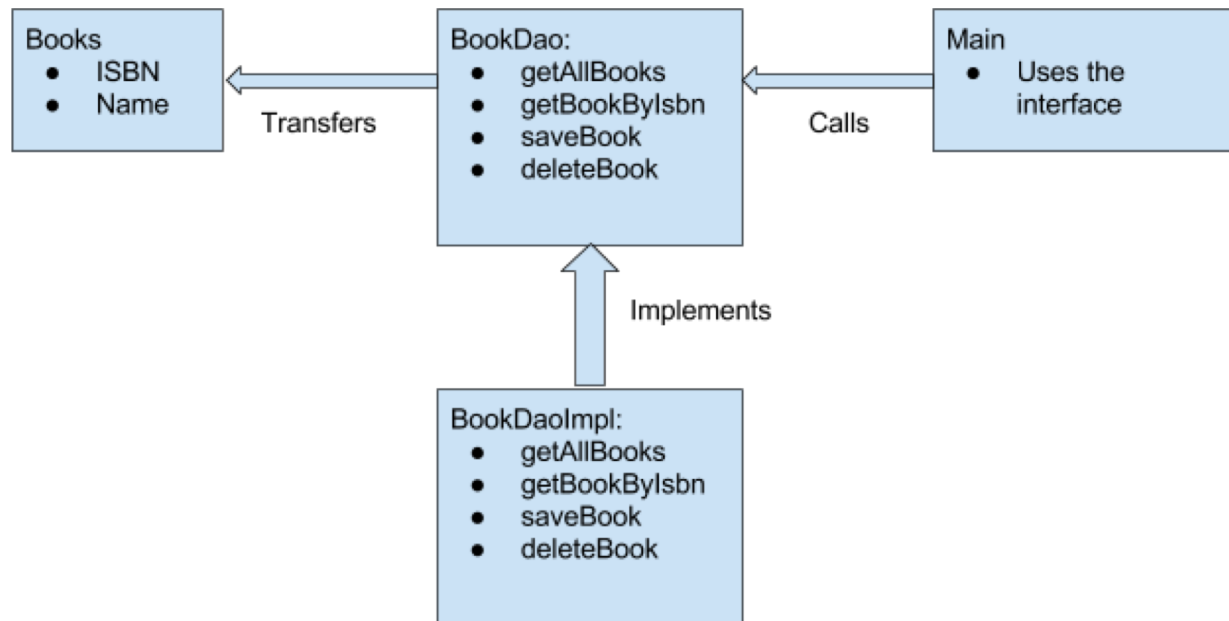
Consultas en Hibernate y JPA

DAOs y Repositorios

- El patrón de diseño DAO establece que por cada objeto de dominio persistente debe existir una clase que tenga la responsabilidad de persistir y recuperar dichas instancias.
- El problema de este patrón es que usualmente lleva a soluciones en las cuales el modelo está “anémico”.
- De las operaciones CRUD, la que conviene atacar en especial es la “R”, es decir la recuperación.
- De ahí surge el patrón de diseño Repository.

DAOs

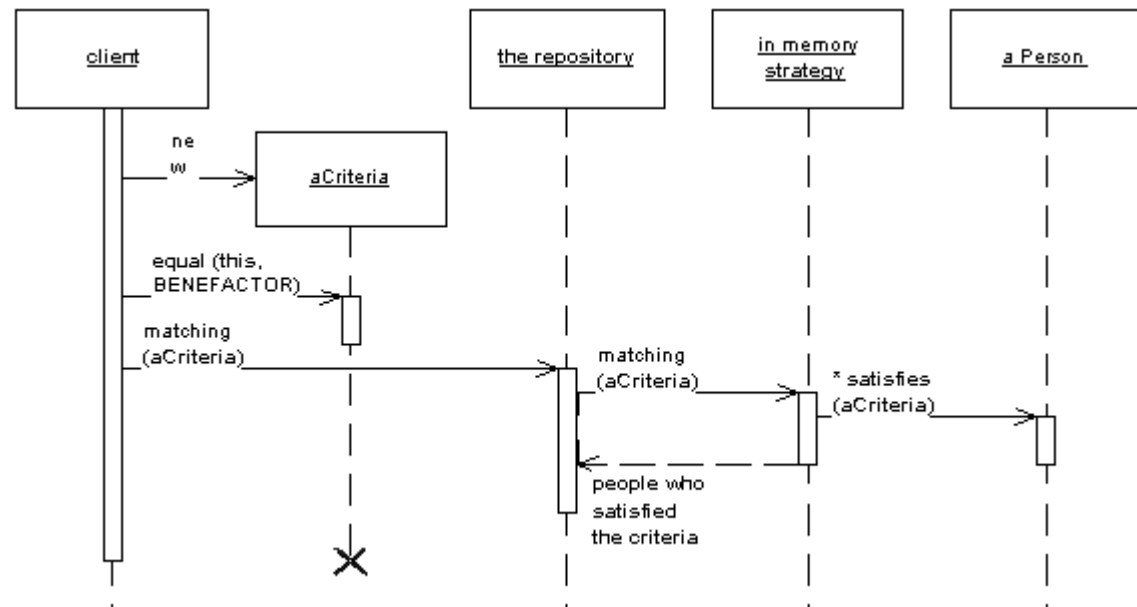
- Esquema típico



Patrones

- Repository



- Mediates between the domain and data mapping layers using a collection-like interface for accessing domain objects.





Patrones

- Repository



- La implementación en Java puede ser considerable

▼   HibernateIndividualsRepository

- ▲ findIndividualByEmail(String) : Individual
- ▲ findIndividuals(int, long, String, String, boolean, String) : Collection<Individual>
- ▲ getIndividualsCount(boolean, String) : long

▼   HibernateUsersRepository

- ▲ findUserByEmail(String) : User
- ▲ findUserByName(String) : User
- ▲ findUsers(int, long, String, String, boolean) : Collection<User>
- ▲ getUsersCount(boolean, String) : long

▼   HibernateZoigenRepository

- ▲ findExtractionCenter(String) : ExtractionCenter
- ▲ findServiceDefinitionByName(String) : ServiceDefinition
- ▲ findServiceDefinitions(int, long, String, String) : Collection<ServiceDefinition>
- ▲ findServiceRequestById(String) : ServiceRequest
- ▲ findServiceRequests(int, long, String, String, String) : Collection<ServiceRequest>
- ▲ findServiceRequestsByIndividual(String) : Collection<ServiceRequest>
- ▲ findZoigen() : Zoigen
- ▲ getNextId() : long
- ▲ getServiceDefinitionsCount() : long
- ▲ getServiceRequestsCount(String) : long

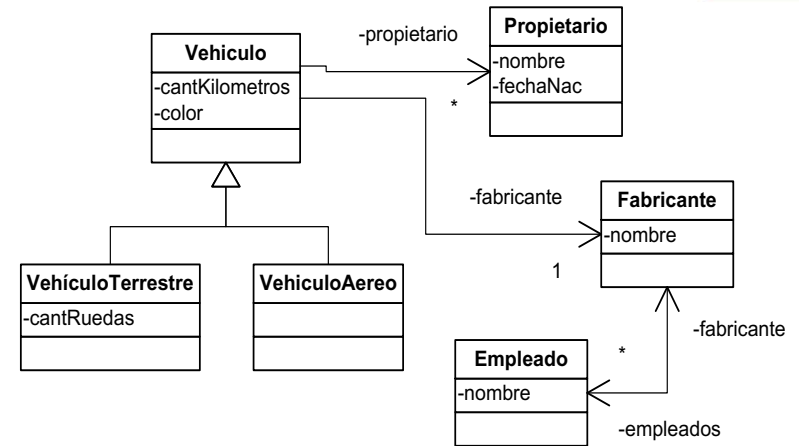
OQL

- Alrededor del año 2000 la organización O.M.G. (Object Management Group) decide publicar un estándar para el acceso a las bases de datos orientadas a objetos.
- Dicho lenguaje de consulta recibió el nombre de O.Q.L. (Object Query Language).
- Con el fin de asegurarse una rápida adopción, tomaron como base el SQL

SQL	OQL
Select <atributos>	Select mensajes
From <tabla,tabs>	From Clase
Where <condiciones>	Where mensajes

OQL

- Un ejemplo simple



- Listado de todos los Vehículos

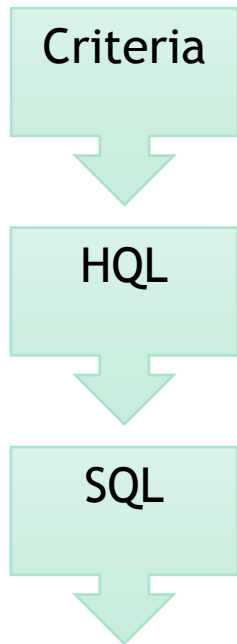
*Select * from Vehiculo*

- Listado de todos los propietarios

Select v.getPropietario() from Vehiculo v

Consultas en Hibernate

- Hibernate tiene varias formas de consultar a la base, una bastante parecida a OQL.



- Existen tres formas de escribir una consulta:
 - Con SQL
 - `session.createQuery(
 "select {p.*} from PERSONA {p} where
 nombre like "%Pablo%", p, Persona.class)`
 - En HQL (Hibernate Query Language)
 - `session.createQuery(
 "from Persona p where p.nombre like
 "%Pablo%");`
 - Con un Criteria (consultas dinámicas)
 - `session.createCriteria(Persona.class)
 .add(Expression.like("nombre", "%Pablo%");`

Instanciando Query

- Para crear una nueva instancia de `Query`, se debe invocar a:

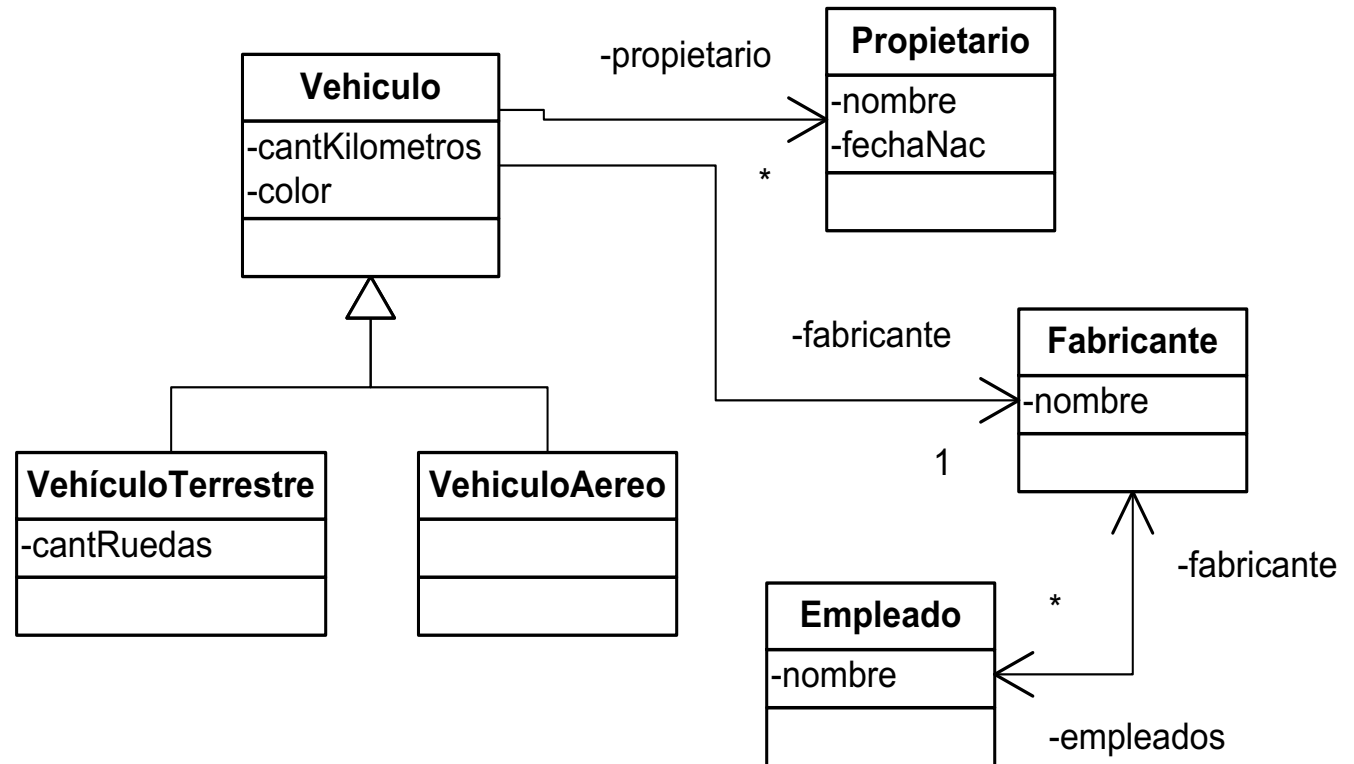
- `createQuery()` : prepara una consulta con HQL

- `Query hqlQuery = session.createQuery("from Persona");`

- `createSQLQuery()` : crea una consulta SQL usando la sintaxis de la base subyacente

- `Query sqlQuery = session.createSQLQuery("select {p.*} from PERSONA {p}", "p", Persona.class)`

Ejemplos



Consultas

- Listar todos los vehículos

```
from Vehiculo v
```

- Listar todos los vehículos rojos

```
from Vehiculo v
```

```
where v.color like "rojo"
```

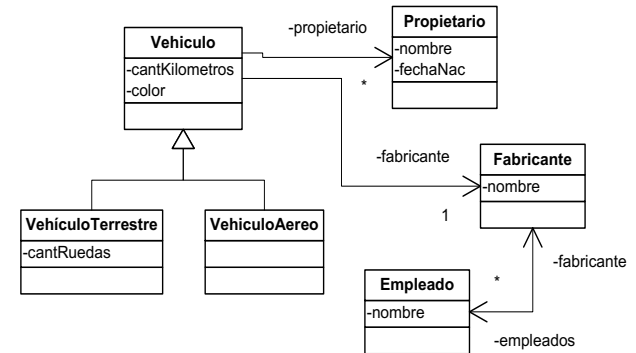
- Listar todos los vehículos del fabricante con nombre honda

```
select v
```

```
from Vehiculo v
```

```
      join v.fabricante as f
```

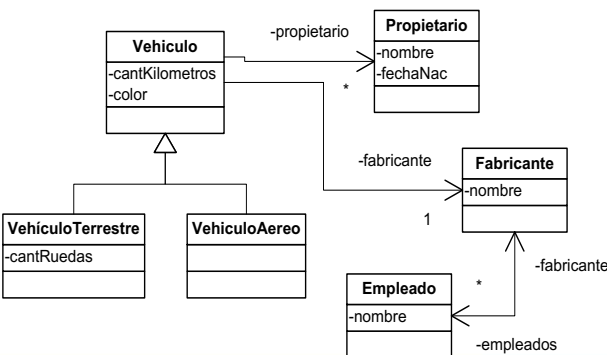
```
where f.nombre like "honda"
```



Consultas

- Listar todos los vehículos rojos de más de 20000 kilómetros que pertenecen al fabricante honda y cuyo propietario se llama Juan.

```
select v
from Vehiculo as v
      join v.propietario as p
      join v.fabricante as f
where v.color like "rojo" and
      v.cantKilometros > 20000 and
      f.nombre like "honda" and
      p.nombre like "juan"
```



Consultas

- Listar todos los vehículos terrestres rojos cuyo kilometraje es mayor al promedio.

```
select v
from VehiculoTerrestre as v
where
    v.color like "rojo"
    v.cantKilometros > (
        select avg(v1.cantKilometros)
        from Vehiculo as v1 )
```

Externalización de consultas

- Hibernate permite guardar consultas en un archivo xml y recuperarlas por un nombre.
- Esto se conoce como consultas nombradas
 - Ejemplo:

```
session.getNamedQuery("buscarPersonasPorNombre")  
    .setString("nombre", nombre)  
    .list();
```

- Donde “buscarPersonasPorNombre” está en Persona.hbm.xml usando el elemento <query>

```
<query name="buscarPersonasPorNombre">  
    <![CDATA[from Persona persona  
    where persona.nombre like :nombre  
    ]]>  
</query>
```

- No necesariamente tiene que ser HQL, puede ser SQL.

Tipos de retorno posibles

- Array de Object[]

```
select mother, offspr, mate.name  
from DomesticCat as mother  
    inner join mother.mate as mate  
    left outer join mother.kittens as offspr
```

- Array de listas

```
select new list(mother, offspr, mate.name)  
from DomesticCat as mother  
    inner join mother.mate as mate  
    left outer join mother.kittens as offspr
```

- Objeto nuevo

```
select new Family(mother, mate, offspr)  
from DomesticCat as mother  
    join mother.mate as mate  
    left join mother.kittens as offspr
```

Funciones de agregación

- Ejemplos

```
select avg(cat.weight), sum(cat.weight), max(cat.weight), count(cat)
from Cat cat
```

```
select cat.weight + sum(kitten.weight)
from Cat cat
      join cat.kittens kitten
group by cat.id, cat.weight
```

```
select count(distinct cat.name), count(cat) from Cat cat
```

```
select firstName||' '||initial||' '||upper(lastName) from Person
```


Campos especiales

- Id

```
from Cat as cat where cat.id = 123
```

- Class

```
from Cat cat where cat.class = DomesticCat
```

Otras funciones (any, some, all, exists)

```
select mother from Cat as mother, Cat as kit  
where kit in elements(foo.kittens)
```

```
select p from NameList list, Person p  
where p.name = some elements(list.names)
```

```
from Cat cat where exists elements(cat.kittens)
```

```
from Player p where 3  
> all elements(p.scores)
```

```
from Show show where 'fizard' in indices(show.acts)
```

Subconsultas

```
from Cat as fatcat
where fatcat.weight
> (
    select avg(cat.weight) from DomesticCat cat
)
```

```
from DomesticCat as cat
where cat.name = some (
    select name.nickName from Name as name
)
```

```
from Cat as cat
where not exists (
    from Cat as mate where mate.mate = cat
)
```

```
from DomesticCat as cat
where cat.name not in (
    select name.nickName from Name as name
)
```

```
select cat.id, (select max(kit.weight) from cat.kitten kit)
from Cat as cat
```

JPA

- Relación entre JPA y Hibernate



Una consulta JPQL es siempre una consulta HQL válida.
La inversa no siempre es verdadera

JPA

- Consultas

- JPQL

```
1 entityManager.createQuery("from Project p where p.projectPeriod.startDate = :startDate",  
Project.class).setParameter("startDate", createDate(1, 1, 2015));
```

- Criteria API

```
1 CriteriaBuilder builder = entityManager.getCriteriaBuilder();  
2 CriteriaQuery<Person> query = builder.createQuery(Person.class);  
3 Root<Person> personRoot = query.from(Person.class);  
4 query.where(builder.equal(personRoot.get("firstName"), "Homer"));  
5 List<Person> resultList = entityManager.createQuery(query).getResultList();
```