Software Ingeniaritza

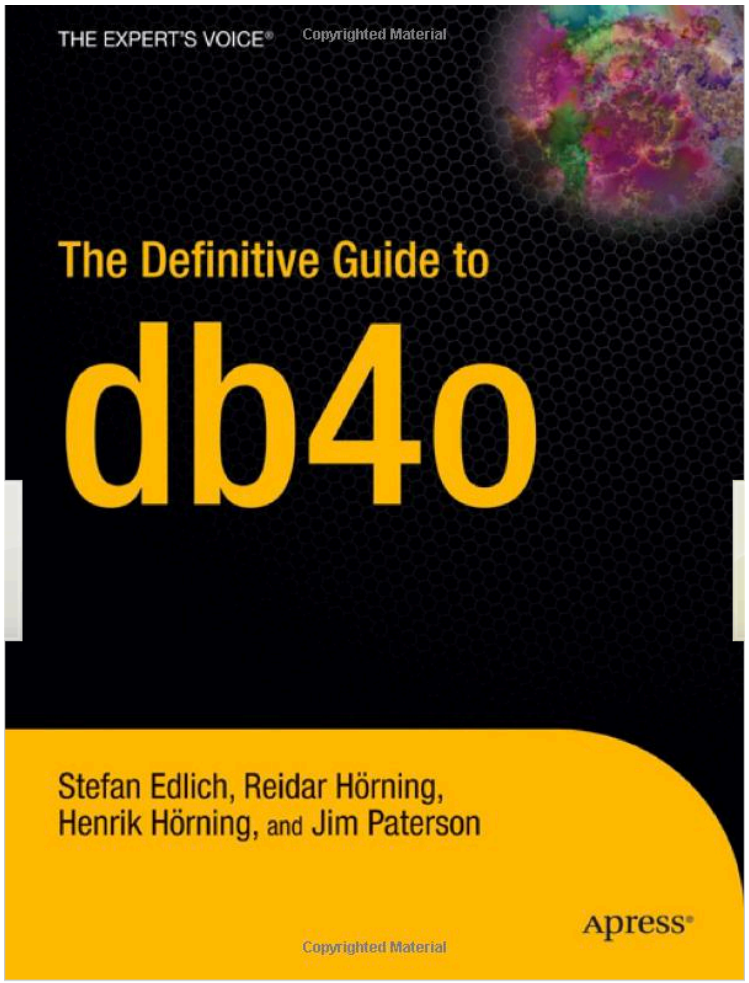
4. Gaia: Inplementazioa

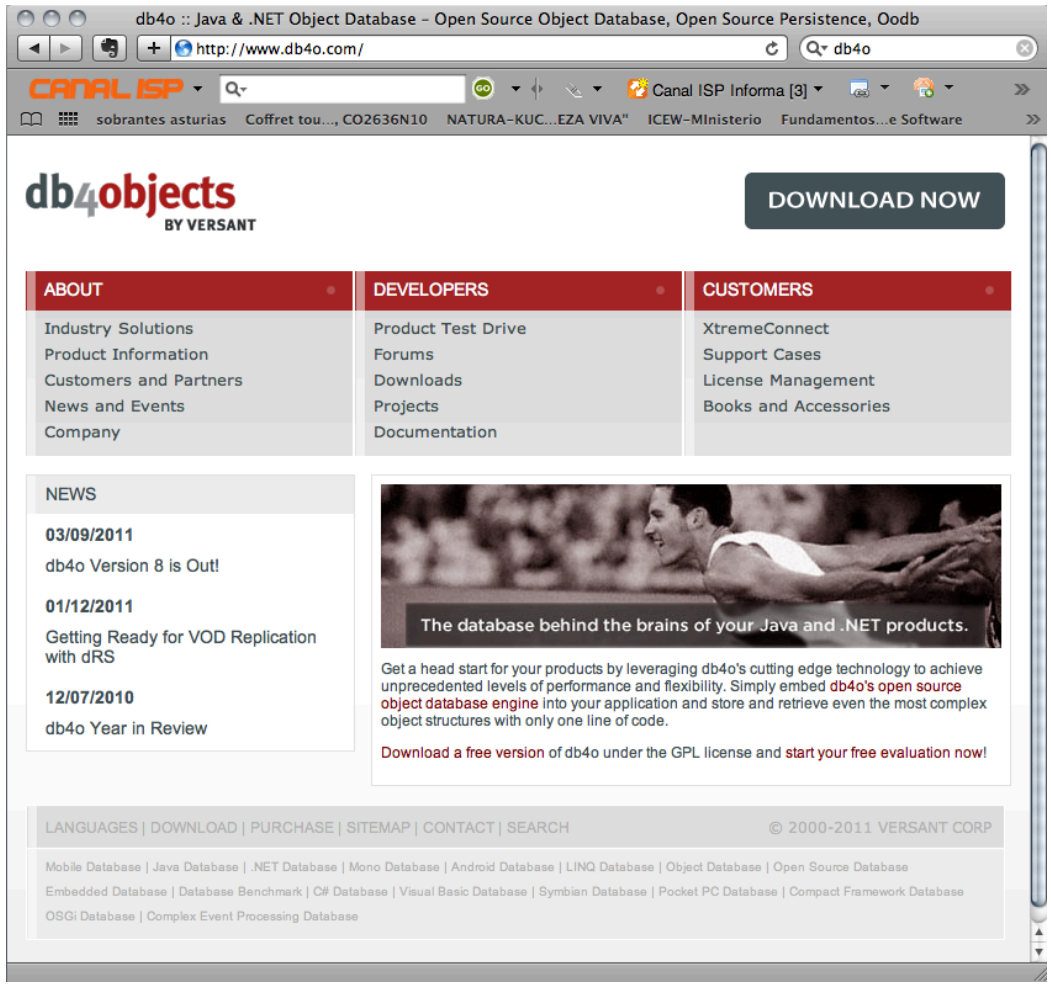
4.2 Objektuen pertsistentzia: db4o

A. Goñi, J. Ibáñez, J. Iturrioz, J.A. Vadillo

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Informazioa





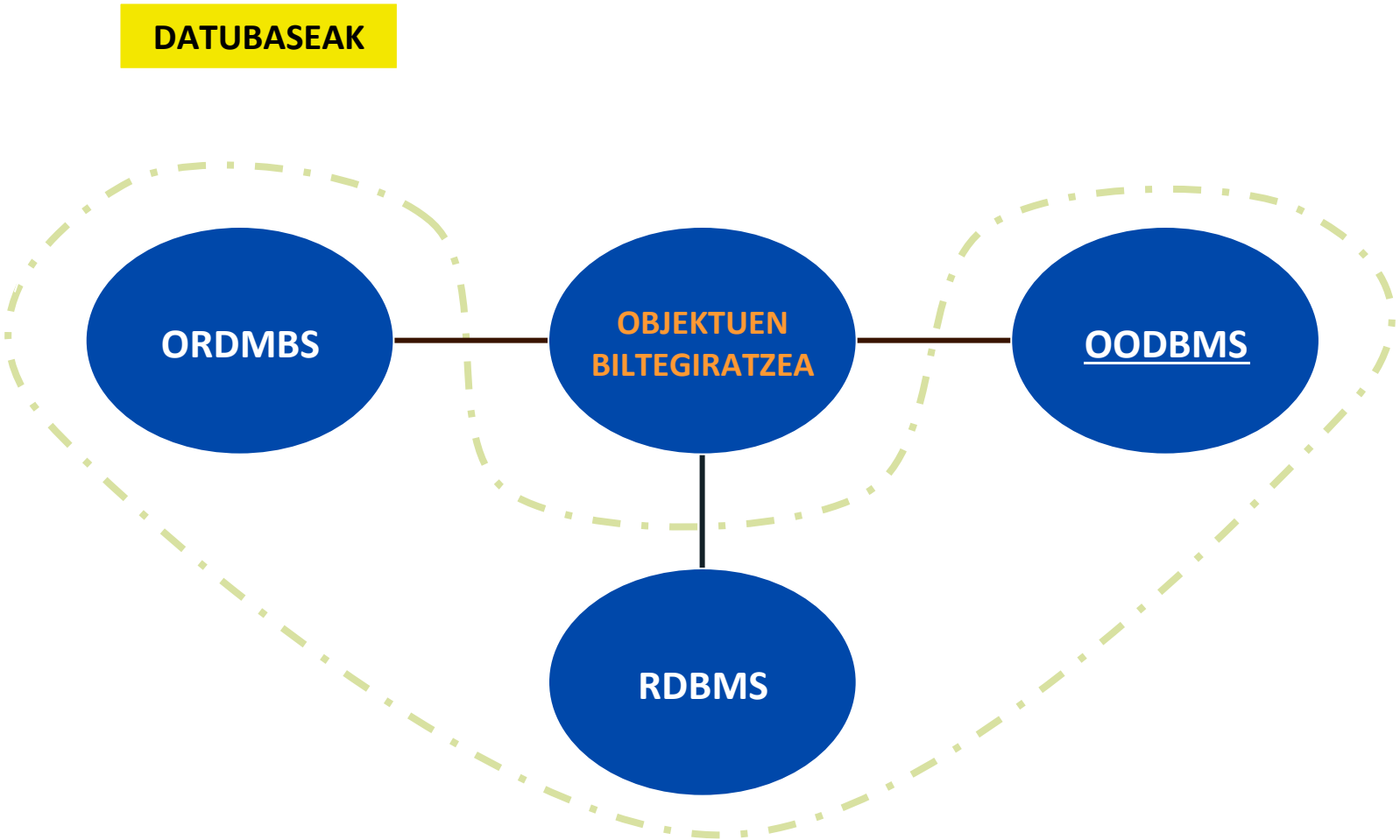
Aurkibidea

- Sarrera
- Db4o ezaugarriak
- Datu basea sortu
- Objektuak gorde
- Objektuak kontsultatu
- Objektuak eguneratu
- Objektuak ezabatu
- Herentzia
- Transakzioak

Pertsistentzia

- Objektuen biltegiratze eta errekuperazioa programazioaren lan garrantzitsuenetariko bat da.
- Pertsistentziak objektuak memoria egonkorrean biltegiratzen ditu, jarraian errekuperatzeko helburuarekin.
- Objektu zuzendutako sistemetan, aukera desberdinak daude objektuak gordetzeko.

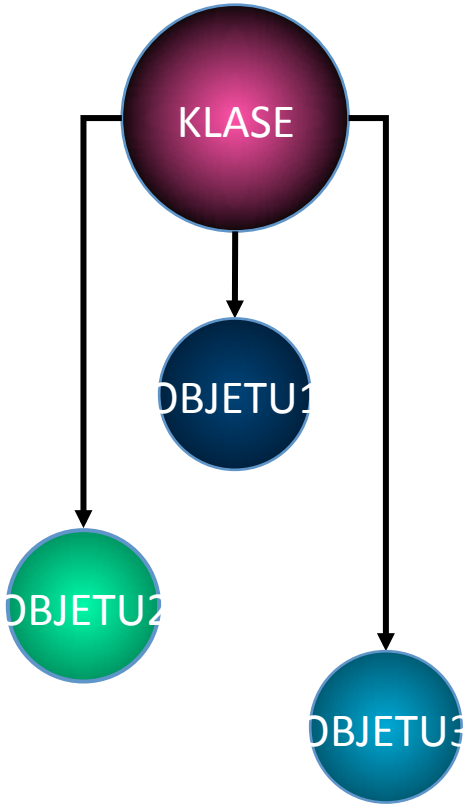
Objektuak datu basetan



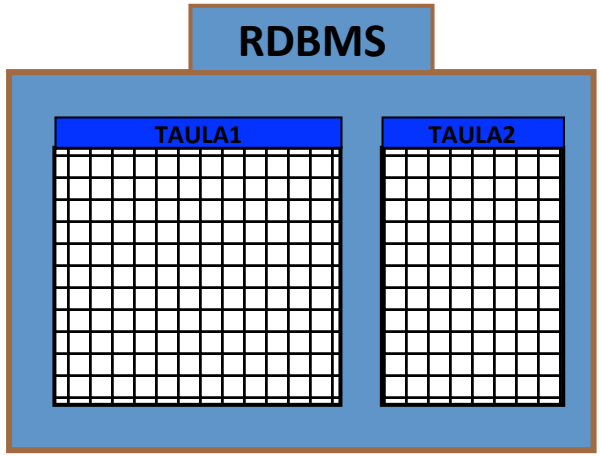
Objektuak -> RDBMS

- Objektuak Taulak, lerroak (erregistroak) eta zutabetan (atributuak) osatutako datu base erlazionaletan gordetzen dira.
- Programatzailearen lana da, objetuak tauletan biltegiratzea.

Objektuetatik->Tauletara



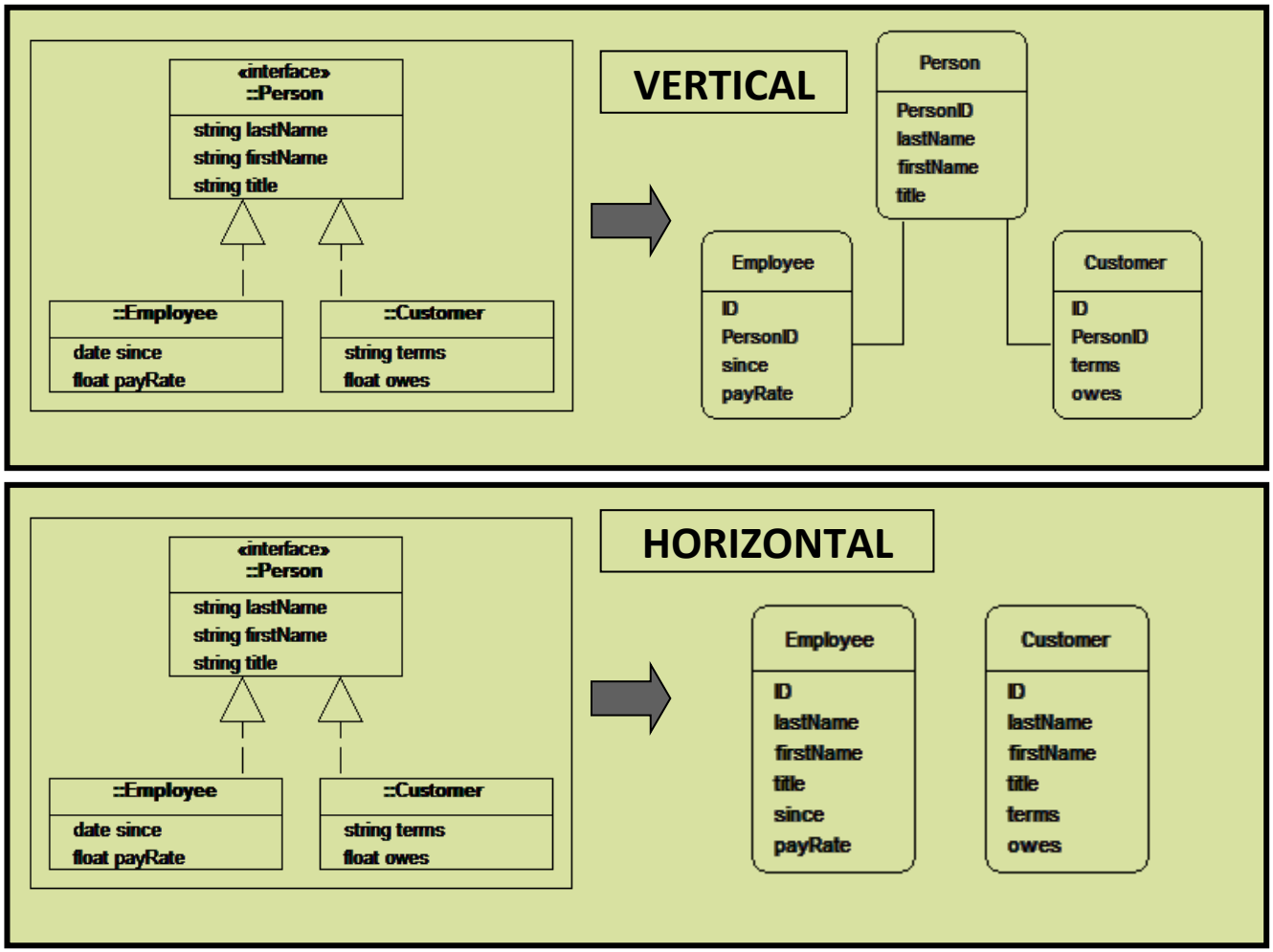
TRANSFORMAZIO



OBJEKTU INGURUNEA

DATU INGURUNEA

Aukera desberdinak

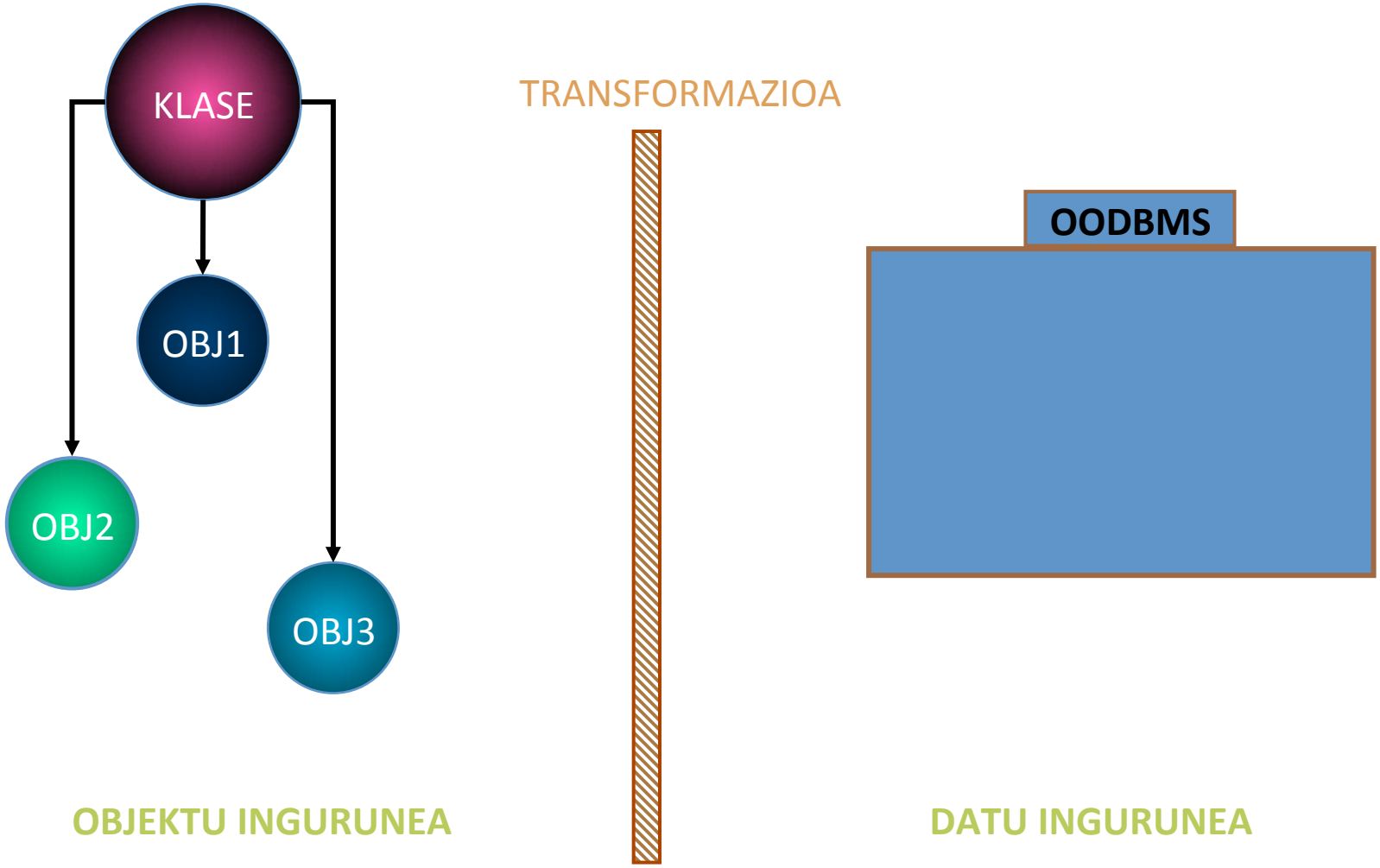


Objetuak -> objektu DB

- Objektuak zuzenean datu basean gordetzen dira inongo transformaziorik gabe (Ez daude taulak eta errregistroak).
- Transparentzia ematen digute datuak godetzerakoan.
- Flexibilitatea ematen digute etorkizuneko.
- Klaseen eremua datu basearen eskema da.

- Programatzailearen lana errezten du.

Objektuetatik -> Objetuetara



Objektuen identifikadoreak

- OID's : Objektuen identifikadoreak, objektu barnean gordeta, zein objektuekin erlazionatuta dagoen jakiteko.
- OID-a ez dago ikusgai erabiltzaile eta programatzailentzako.
- Objektu batek, objektu bera bezala jarraitzen du nahiz eta bere egoera balio desberdinak hartzen baditu.

Agenda

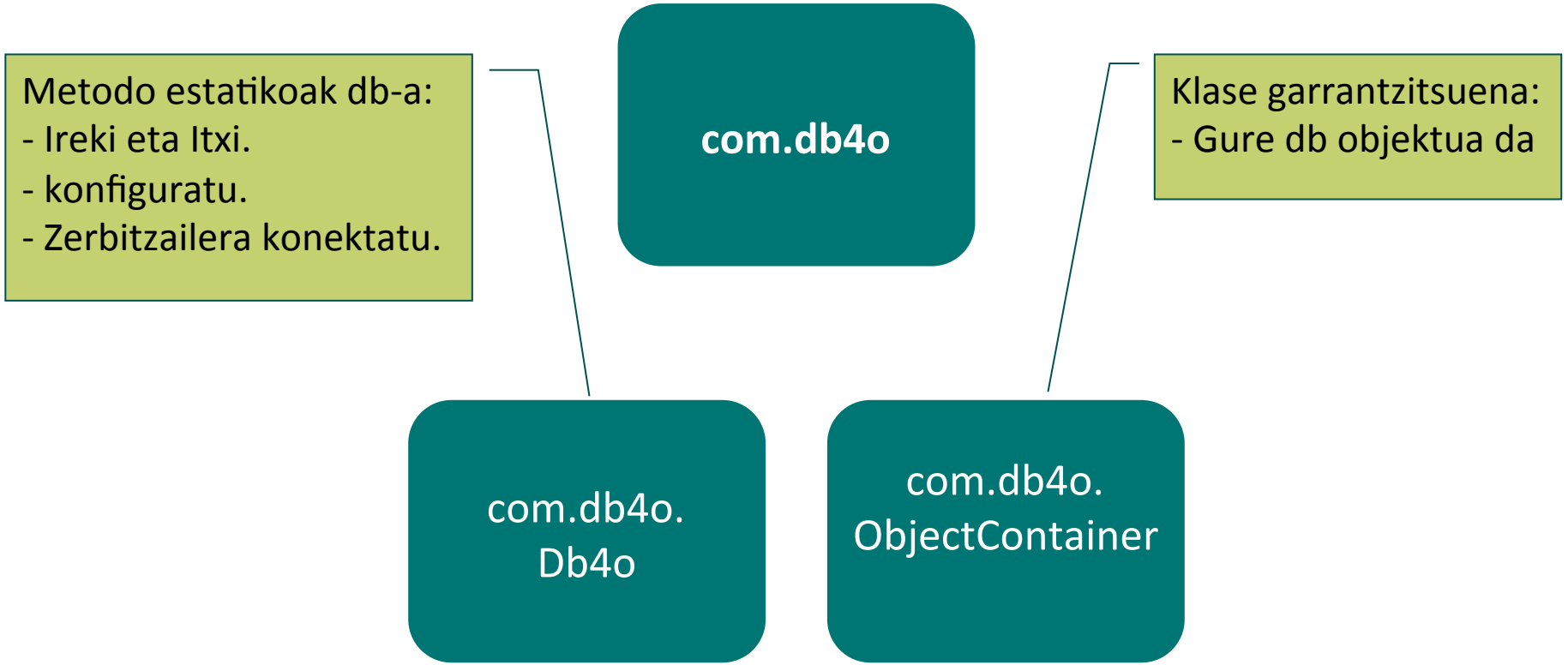
- Sarrera
- Db4o ezaugarriak
- Datu basea sortu
- Objektuak gorde
- Objektuak kontsultatu
- Objektuak eguneratu
- Objetuak ezabatu
- Herentzia
- Transakzioak

db4o: ezaugarriak

- Objektu zuzendutako datu base natiboa.
- Sillicon Valley-n eginda.
- Aplikazio baten barnean erraz murgildu daiteke.
- Aplikazio sinple(Standalone) edo Bezero/ Zerbitzaile (Aplikazio banatua) aplikazioentzako.
- Java eta .net ingunentzako.

Db4o: Datu base sortu

- com.db4o.Db4o eta com.db4o.ObjectContainer klaseekin aplikazio bat garatu dezakegu.



Datu base ireki eta itxi

.openFile(String file)
.close()

```
ObjectContainer db = Db4o.openFile("Archivo.yap");
try {
        // Zerbait egin datubasearekin
finally {
   db.close();   // datu basea itxi atera baino lehen
```

Datu basea errepresentatzen du

Pilot klasea

```
public class Pilot {
  private String name;
  private int points;
  public Pilot(String name,int points) {
    this.name=name;
    this.points=points;
  }
  public int getPoints() {
    return points;
  }
  public void addPoints(int points) {
    this.points+=points;
  }
  public String getName() {
    return name;
  }
  public String toString() {
    return name+"/"+points;
  }
}
```

Db4o: Objektuak gorde

.store(Object o)

```
Pilot pilot1 = new Pilot("Michael Schumacher",100);

db.store(pilot1);

System.out.println("Stored "+pilot1);
```

OUTPUT:
Stored Michael Schumacher/100

Beste Pilot bat datubasean

```
Pilot pilot2 = new Pilot(”Rubens Barrichelo",99);

db.store(pilot2);

System.out.println("Stored "+pilot2);
```

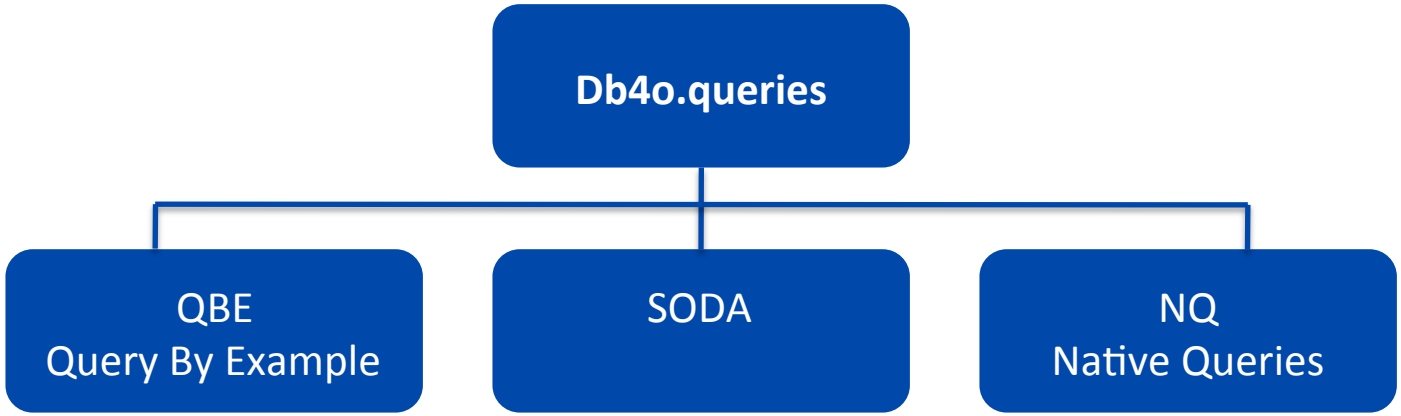
OUTPUT:
Stored Rubens Barrichelo/99

Aurkibidea

- Sarrera
- Db4o ezaugarriak
- Datu basea sortu
- Objektuak gorde
- Objektuak kontsultatu
- Objektuak eguneratu
- Objektuak ezabatu
- Herentzia
- Transakzioak

Db4o: Queries

- 3 galdera mota
  - Query by Example (QBE): Prototipo bat erabiliz
  - Native Queries (NQ): Jatorrizko lengoaian egindako galderak
  - Simple Object Database Access (SODA): Nodo bitartez definitutako galdera dinamikoak



Query by Example (QBE)

1. Galdera oso errazak eta azkarrak.
2. Prototipo objektu bat sortzen da lortu nahi ditugun ezaugarriekin (defektuzko null eta 0 atributuak).
3. .get metodoa erabiltzen da prototipo objektua parametro bezala pasatuz.
4. ObjectSet objektu bat lortzen dugu emaitzarekin.
5. Murrizpenak:
   - Espresio konposatuak dauzkaten galderak ezin dira egin (AND, OR, NOT, etc.)
   - Ezin dira 0 edo null balioak ezarri galdereei.
   - Objektuen eraikitzaile bat behar da.

Pilot guztiak errekuperatu

.get(Object o)

```
Pilot proto = new Pilot(null,0);

ObjectSet result=db.get(proto);

listResult(result);
```

```
ObjectSet result=db.get(Pilot.class);
```

```
public static void listResult(ObjectSet result) {
  System.out.println(result.size());
    while(result.hasNext()) {
      System.out.println(result.next());
    }
}
```

```
OUTPUT:
2
Michael Schumacher/100
Rubens Barrichello/99
```

Pilot konkretu batzuk errekuperatu

100 puntu dauzkaten Pilotoak.

```
Pilot proto = new Pilot(null,100);

ObjectSet result=db.get(proto);

listResult(result);
```

```
OUTPUT:
1
Michael Schumacher/100
```

Native Queries

- Erabiltzen dugun **programazio lengoai berberarekin** idazten dira.
- Konpilazio garaian frogatzen dira.
- Kontsulta barruan metodoei deitu daitezke.

Native Queries

1. Predicate interfazea inplementatzen duen klase anonimo baten objektu bat sortzen dugu, nahi dugun galdera deskribatzeko.

   [new Predicate()]

2. Deskribatu nahi den galdera

   boolean match(Object o)

   metodoan definitzen da.

3. **match** metodoaren inplementazioan galderaren baldintzak definitzen dira, true itzuliz objektuak betetzen baditu.

4. **ObjectSet query(Predicate p)** metodoak Predicate objektua betetzen dituzten objektuak itzultzen ditu ObjectSet egituran.

Native Query

100 puntu dauzkaten Pilotoak

```
Predicate<Pilot> galdera=new Predicate<Pilot>(){
      public boolean match(Pilot pilot) {
       return pilot.getPoints() == 100;
     }
}

ObjectSet pilots = db.query(galdera);
```

Native Queries: ariketak

- Landetxeak dauzkaten jabeak itzuli
- Eskaeraren bat dauzkaten landetxeen jabeak itzuli
- Erreserbaren bat dauzkaten landetxeen jabeak itzuli

Native Queries. Sort

Pilotoak puntu handienetik txikienera ordenatuta

```
Comparator<Pilot> pilotCmp = new Comparator<Pilot>() {
    public int compare(Pilot p1, Pilot p2) {
        return (p1.getPoints()-p2.getPoints());
    }
};
```

compare metodoaren emaitza:

- if (a>b) return >0 (adibidez 1)
- if (a==b) return 0
- if (a<b) return <0 (adibidez -1)

Native Queries. Sort

Orain galdera osatzen da comparator objektuarekin

```
Predicate<Pilot> galdera=new Predicate<Pilot>(){
       public boolean match(Pilot pilot) {
        return true;
      }
}
Comparator<Pilot> pilotCmp = new Comparator<Pilot>() {
    public int compare(Pilot p1, Pilot p2) {
       return (p1.getPoints()-p2.getPoints());
    }
}

ObjectSet pilots = db.query(galdera,pilotCmp);
```

db4o: objektuak eguneratu

```
ObjectSet result=db.get(new Pilot("Michael Schumacher",0));

Pilot found=(Pilot)result.next();

found.addPoints(11);
db.store(found);
System.out.println(“Added 11 points for “+found);


retrieveAllPilots(db);
```

```
OUTPUT:
Added 11 points to Michael Schumacher/111
2
Michael Schumacher/111
Rubens Barrichello/99
```

Adibidea: *Car* klasea

```
public class Car {
  private String model;
  private Pilot pilot;

  public Car(String model) {
    this.model=model;
    this.pilot=null;
}
  public Pilot getPilot() {
    return pilot;
  }
  public void setPilot(Pilot pilot) {
    this.pilot=pilot;
  }
  public String getModel() {
    return model;
  }
  public String toString() {
    return model+"["+pilot+"]";
  }
}
```

Objektu konposatuak eguneratu

```
// SESIO 1
ObjectSet result=db.query(new Predicate() {
public boolean match(Car car){
return car.getModel().equals("Ferrari"); }
});
Car found=(Car)result.next();
found.getPilot().addPoints(1);
db.store(found);
listResult(result);
```

```
OUTPUT:
1
Ferrari[Michael Schumacher/101]
```

```
// SESIO 2
ObjectSet result=db.query(new Predicate() {
public boolean match(Car car){
return car.getModel().equals("Ferrari");}
});
listResult(result);
```

```
OUTPUT:
1
Ferrari[Michael Schumacher/100]
```

ERROREA, EZ DU EGUNERATU PUNTU KOPURUA!

Objektu konposatuak eguneratu

```
// DATUBASEA KONFIGURATU OBJEKTU BAT EGUNERATZEKOAN, BERE OBJEKTU ERLAZIONATUAK EGUNERATU DADIN

Db4o.configure().objectClass(Car.class).cascadeOnUpdate(true);
```

```
// SESIO 1

ObjectSet result=db.query(new Predicate() {

public boolean match(Car car){

return car.getModel().equals("Ferrari"); }

});

Car found=(Car)result.next();

found.getPilot().addPoints(1);

db.store(found);

listResult(result);
```

```
OUTPUT:
1
Ferrari[Michael Schumacher/101]
```

```
// SESIO 2

ObjectSet result=db.query(new Predicate() {

public boolean match(Car car){

return car.getModel().equals("Ferrari");}

});

listResult(result);
```

```
OUTPUT:
1
Ferrari[Michael Schumacher/101]
```

ORAIN ONDO EGITEN DU!

Objektuak ezabatu

.delete(Object o)

```
ObjectSet result=db.get(new Pilot("Michael Schumacher",0));
Pilot found=(Pilot)result.next();
db.delete(found);
System.out.println("Deleted "+found);

retrieveAllPilots(db);
```

Ezagutzen dugun objektu bat ezabatu

OUTPUT:
Deleted Michael Schumacher/101
1
Rubens Barrichello/99

Objektu konposatuak ezabatu

```
// SESIO 1
ObjectSet result=db.query(new Predicate() {
public boolean match(Car car){
return car.getModel().equals("Ferrari"); }
});
Car found=(Car)result.next();
db.delete(found);
result=db.get(new Car(null));
listResult(result);
```

```
// SESIO 2
// retrieveAllPilotsQBE
Pilot proto=new Pilot(null,0);
ObjectSet result=db.get(proto);
listResult(result);
```

OUTPUT:
2
Rubens Barrichello/99
Michael Schumacher/101

OUTPUT:
1
BMW[Rubens Barrichello/99]

“Michael Schumacher” Ferrari pilotoa ez da ezabatu. Logiko dirudi.

```
// JAUZIAN EZABATU NAHI BADIRA (KONTUZ!!!!!)
Db4o.configure().objectClass(Car.class).cascadeOnDelete(true);
```

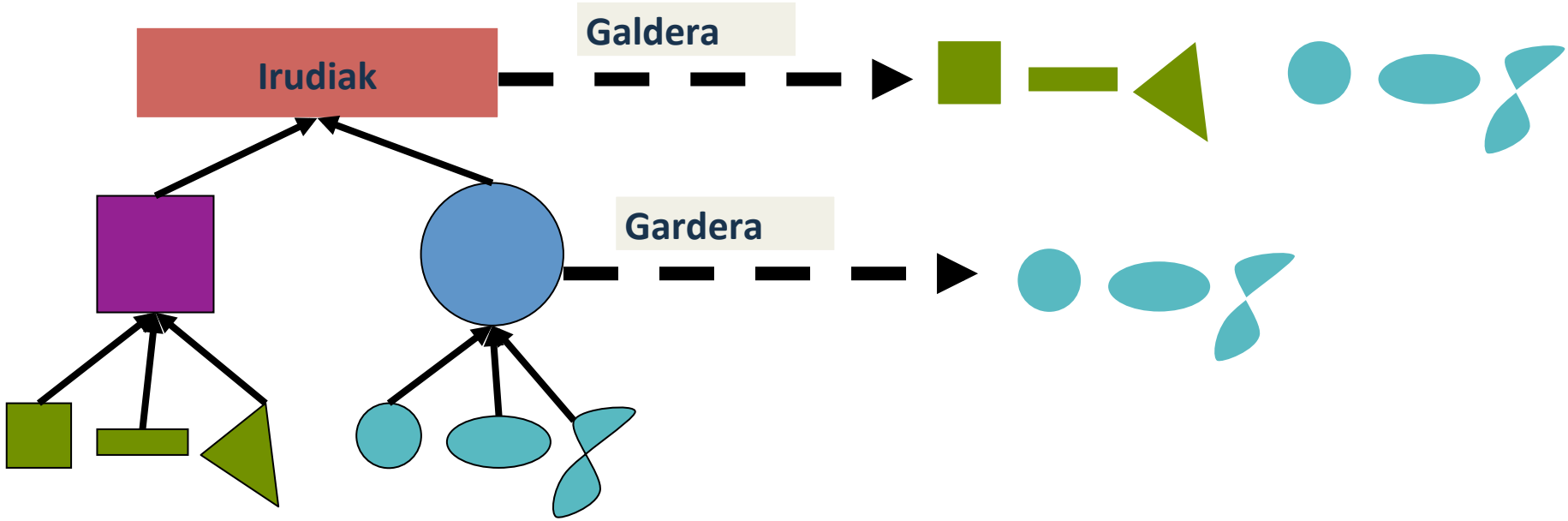
Aurkibidea

- Sarrera
- Db4o ezaugarriak
- Datu basea sortu
- Objektuak gorde
- Objektuak kontsultatu
- Objektuak eguneratu
- Objektuak ezabatu
- Herentzia
- Transakzioak

db4o: Herentzia

Galdetutako objektu mota itzultzen digu :

- Superklasea galdetuz, bere subklase guztien objektuak itzultzen ditu.
- Subklasea galdetuz, bere subklaseko objektuak itzultzen ditu soilik.



db4o: herentzia

Zer gertatzen da QBE klasea egiterakoan klasea abstraktua edo Interfazea bada?

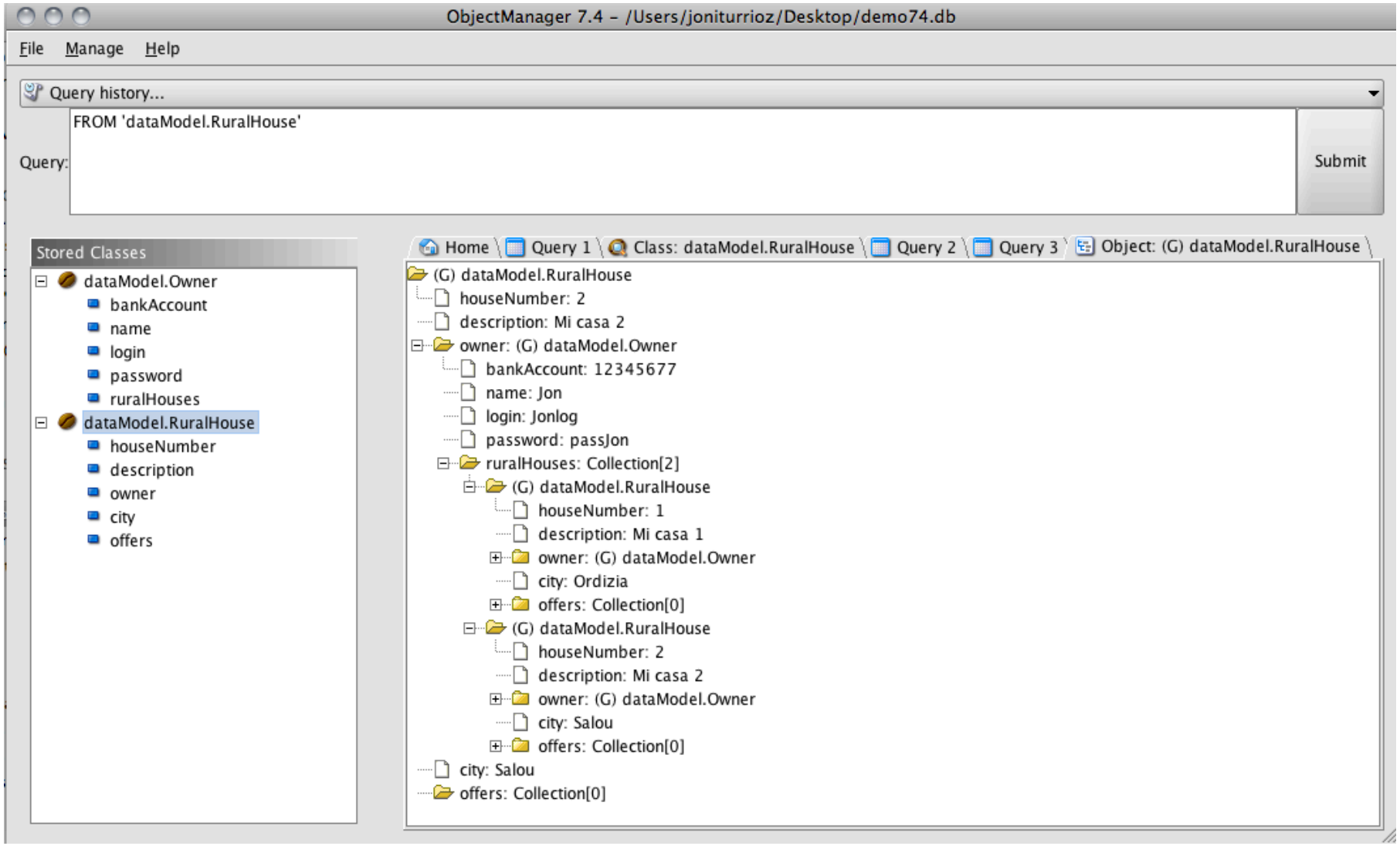
- Ezin dugu eraikitzailea erabili prototipoa sortzeko.
- Ebazpena: 'NireClase'**.class** erabiltzen dugu

```
ObjectSet result=db.get(Pilot.class);
```

db4o: transakzio sinpleak

- db4o bi metodo eskaintzen ditu:
  - *commit()* transakzio bat amaitzen du.
  - *rollback()* transakzio bat desegiten du.
- Transakzio bat inplizituki ixten da db-a ixten denean.

Objektuak bistaratzeko tresna



http://code.google.com/p/db4o-om/