



# Informàtica

## **ICB0 Desenvolupament d'aplicacions multiplataforma**

M06 Accés a dades

UF2 Persistència en BDR-BDOR-BDOO

**Enunciat examen 2017-2018**

Isidre Guixà



## Enunciat examen curs 2017-2018

Material dins el projecte NetBeans RRHH facilitat.

Considereu el disseny MR de la dreta, pensat per gestionar informació vinculada als recursos humans d'una empresa.

Creeu en el SGBD Oracle un esquema de nom RRHHV1 on executar el guió (facilitat dins el projecte) RRHH\_Oracle\_Schema&Data.sql, que conté la definició de taules per l'Oracle i joc de proves i un esquema RRHHV2 buit.

**Totalment prohibit efectuar cap modificació en aquest esquema!!!**

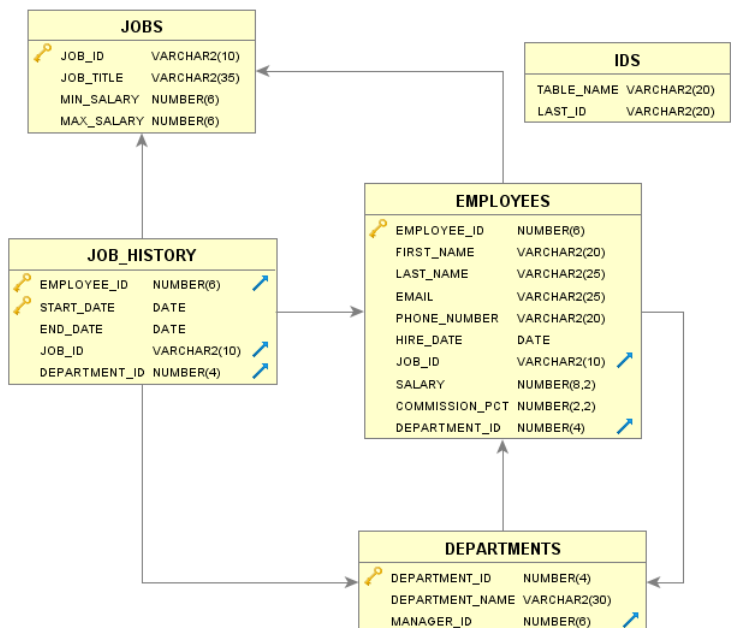
La taula IDS està pensada per contenir comptadors dels identificadors automàtics de diverses taules. En aquest moment, però, només conté el comptador per la taula EMPLOYEES com podeu comprovar si feu una consulta del seu contingut.

Els informàtics d'aquesta empresa han dissenyat unes classes Java (esquema UML de la dreta) per gestionar les dades de la BD, i volen aprofitar les facilitats que dona JPA (concretament EclipseLink i Hibernate) però no tenen experiència, ni en JPA ni en el disseny de classes amb relacions bidireccionals i necessiten de la vostra ajuda.

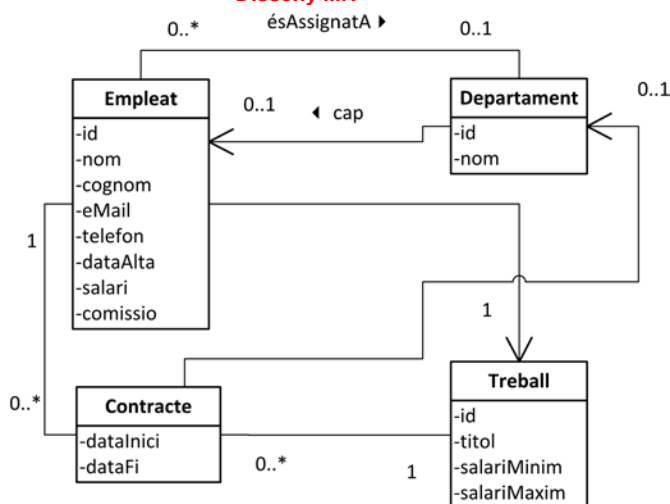
Es demana aplicar un seguit de millores i/o retocs al projecte RRHH facilitat i desenvolupar nous projectes.

Les classes estan "pelades"; només contenen la definició dels camps...

No perdeu temps creant mètodes i més mètodes. Només poseu els mètodes que siguin necessaris i/o imprescindibles pel què es demana.



Disseny MR



Disseny UML

### Exercici 1:

Reanomenar projecte (i carpeta) RRHH facilitat per RRHHV1 on cal retocar i marcar **MÍNIMAMENT** les classes Empleat i Treball via XML i les classes Contracte i Departament via anotacions per a que JPA pugui gestionar adequadament el mapatge entre les classes i l'esquema RRHHV1 emplenat amb el guió facilitat. Aquest marcatge ha de sincronitzar (validar) amb l'estructura existent dins l'esquema RRHHV1. Cal tenir en compte que:

- Cal implementar la correcta sincronització de les relacions bidireccionals.
- L'eliminació d'un empleat ha de suposar l'eliminació automàtica dels seus contractes.
- L'eliminació d'un departament no ha de suposar l'eliminació dels seus empleats, els quals quedaran sense departament assignat.
- La persistència d'un departament ha de provocar la persistència dels empleats encara no persistents que pugui tenir assignats.
- La classe Empleat ha d'incorporar les consultes JPQL (no natives) amb nom següents:
  - trobaTots, que recuperi tots els empleats
  - trobaEmpleatPerId, que recuperi l'empleat amb identificador passat per paràmetre
  - trobaContractesDeEmpleatPerId, que recuperi els contractes d'un empleat amb identificador passat per paràmetre

El marcatge XML incorporeu-lo en carpeta META-INF dins el projecte RRHHV1.

### **Exercici 2:**

Incorporeu en el projecte RRHHV1 els següents programes en paquet `org.milaifontanals.progs`:

- `P01_ComprovarEsquema`, que validi la correctesa del marcatge amb l'estructura de l'esquema RRHHV1.
- `P02_ConsultaEmpleats`, que invoqui la consulta `trobaTots` i mostri les dades de cada empleat, invocant mètode `toString` (generat per NetBeans) de classe `Empleat`.
- `P03_ConsultaContractesEmpleat`, que demani la introducció d'un codi d'empleat per consola i:
  - Comprovi la seva existència (via `trobaEmpleatPerId`), mostrant-lo (`toString`) o informant de la no existència
  - Si hi és, mostri els seus contractes, via `trobaContractesDeEmpleatPerId`, mostrant-los un a un, invocant mètode `toString` (generat per NetBeans) de classe `Contracte`.
- `P04_GestióDades1` que:
  - Comprovi existència de treball (`ED_TE`, `Teacher`, `6000,12000`) creant-lo si no existeix.
  - Comprovi existència de departament (`300`, `Education`) creant-lo si no existeix.
  - Comprovi existència de treballador amb el vostre correu (que és únic dins la taula `EMPLOYEES`) de `milaifontanals.org` creant-lo amb el vostre nom i cognom en cas que no existeixi i, tant si existeix com si no existeix, assignant-lo al departament `300` fent de professor.
  - Assigneu-vos al departament `300` com a cap.
- `P05_GestióDades2` que demani per consola un codi de treball i un percentatge d'augment estrictament positiu i procedeixi a aplicar el percentatge als salaris mínim i màxim del treball indicat així com a tots els empleats que en aquest moment tenen assignat aquest treball. Cal controlar l'existència del treball.

### **Exercici 3:**

- Feu còpia del projecte RRHHV1 amb nom RRHHV2, que caldrà completar.
- Retoqueu RRHHV2 per a que treballi contra l'esquema RRHHV2 buit.
- Completeu el marcatge existent al projecte RRHHV2 amb el màxim de detall possible per a que l'eina JPA creï l'estructura de taules, el més similar possible a la creació que n'efectua el guió facilitat, sense incorporar les restriccions de tipus `CHECK` ni `columnDefinition` (que poden variar segons SGBDR).
- Dins el paquet `org.milaifontanals.progs` de RRHHV2 deixeu només el programa `P01`, de manera que en executar-lo es procedeixi a recrear tota l'estructura de dades dins l'esquema RRHHV2 d'Oracle.

### **Passos a seguir per aconseguir la solució**

- Completar les classes (els mètodes `setter` estan incomplets, caldria afegir les comprovacions)
  - `Serializable`
  - Donat que a l'exercici 2-P04 hem de procedir a crear objectes de diverses classes (`Departament`, `Empleat` i `Treball`), necessitem dotar com a mínim a aquestes classes d'un constructor. Cap problema si es proporciona un constructor en totes les classes.
  - Classe `Empleat`, donat que la clau (camp `id`) és autonumèrica:
    - El constructor no pot contenir paràmetre `id`.
    - No pot tenir mètode `setId`.
  - Constructor sense paràmetres `protected` per a totes les classes que tenen altre constructor.
  - Següents mètodes `setter` privats per què el(s) camp(s) identificadors han de ser immutables:
    - `Departament.setId`
    - `Treball.setId`
    - `Contracte.setEmpleat`
    - `Contracte.setDataInici`
- Sincronització de relacions bidireccionals:
  - Relació `ésAssignatA` entre `Empleat` i `Departament`
    - `Empleat.setDepartament`, que ha de permetre departament nul.
    - `Departament.iteEmpleats`, `Departament.addEmpleat` i `Departament.removeEmpleat`.
  - Relació entre `Contracte` i `Empleat`.
    - `Contracte.setEmpleat`, que NO ha de permetre empleat nul.
    - `Empleat.iteContractes` i `Empleat.addContracte`.



- Relació entre `Contracte` i `Treball`.
  - `Contracte.setTreball`, que NO ha de permetre treball nul.
  - `Treball.iteContractes` i `Treball.addContracte`.
- Crear el fitxer `persistence.xml` adequat.
  - Incorporarem les dades de connexió que corresponguin.
  - Incorporarem en elements `class` totes les classes gestionades per JPA
  - Incorporarem en elements `mapping` els fitxers XML que continguin marcatge XML d'algunes classes. Recordeu que un fitxer XML pot contenir marcatge de vàries classes
- Marcatge via anotacions/XML segons requeriments:
  - En RRHHV1, només introduim el marcatge per a que JPA pugui sincronitzar les classes amb les taules i es garanteixin els requeriments de funcionament indicats.  
Per tant, introduïm:
    - Marca per indicar nom de taula si no coincideix amb el nom de la classe
    - Marca per indicar nom de columna quan no coincideixi amb el nom del camp dins la classe
    - Marca per indicar la clau
    - Marques per la clau automàtica a `Empleat`.
    - Marques i muntatge per aconseguir la clau automàtica composta a `Contracte` on una part de la clau és camp `many2one`. Es pot usar qualsevol dels 2 mecanismes aplicats en classe `AgendaTallers` via `@IdClass` o via `@EmbeddedId`
    - Marca `@ManyToOne/many-to-one` per les relacions molts a un.  
Introduir `Fetch.Lazy` (no s'explicitava però millor fer-ho)  
Introduir `Cascade.persist` (no s'explicitava però millor fer-ho)
    - Informar de tots els camps que no són obligatoris.
    - Marca `@OneToMany/one-to-many` per les relacions un a molts que han de tenir inversa
    - Les consultes amb nom indicades a la classe `Empleat`.
    - No afegim cap clau forana ni restricció d'unicitat doncs ja tenim la BD creada.
    - Respecte: *L'eliminació d'un empleat ha de suposar l'eliminació automàtica dels seus contractes.*  
Cal afegir `cascade-remove` a la relació `one-to-many` d'`Empleat` cap a `Contracte`.
    - Respecte: *La persistència d'un departament ha de provocar la persistència dels empleats encara no persistents que pugui tenir assignats.*  
Cal afegir `CascadeType.PERSIST` a la relació `OneToMany` de `Departament` cap `Empleat`
    - Respecte: *L'eliminació d'un departament no ha de suposar l'eliminació dels seus empleats, els quals quedaran sense departament assignat.*  
La clàusula `cascade` no permet definir un comportament similar a `on delete set null` de la definició de les claus foranes i en aquesta BD no tenim la clau forana `EMP_DEPT_FK` amb `on delete set null`. La solució és usar la funcionalitat de triggers que ens facilita JPA, vista [aquí](#).  
Per tant, incorporem a `Departament` el següent contingut:
 

```
@PreRemove
private void posarNullEnEmpleats() {
    empleats.forEach(e -> e.setDepartament(null));
}
```

El codi anterior no funciona quan la llista `empleats` conté més d'un element degut a que s'està intentant modificar una col·lecció en un procés iteratiu i això Java no ho permeti genera excepció `java.util.ConcurrentModificationException`. Una possible solució consisteix en traspassar els empleats de la llista en una taula i iterar sobre la taula. És a dir:

```
@PreRemove
private void posarNullEnEmpleats() {
    Object t[] = empleats.toArray();
    for (Object o: t) {
        ((Empleat)o).setDepartament(null);
    }
}
```
  - En RRHHV2, a partir de la còpia de RRHHV1, afegim marcatge necessari per intentar crear la BD el més similar possible a la creació que en fa el guió facilitat:
    - Afegim la definició de les claus foranes.
    - Afegim restricció d'unicitat de correu electrònic a la taula `EMPLOYEES`.