M2. BASES DE DADESlínea horizontal  
Jordi Quesada

jordi.quesada@iesjoandaustria.org

dam1m2

<https://meet.google.com/rmp-ngnh-cgk>

**Estructuració del curs**

| UF 1. Introducció a les bases de dades |
| --- |
| UF 2. Llenguatge SQL |
| UF 3. Assegurament |
| **UF4. BDD orientada a Objectes** |

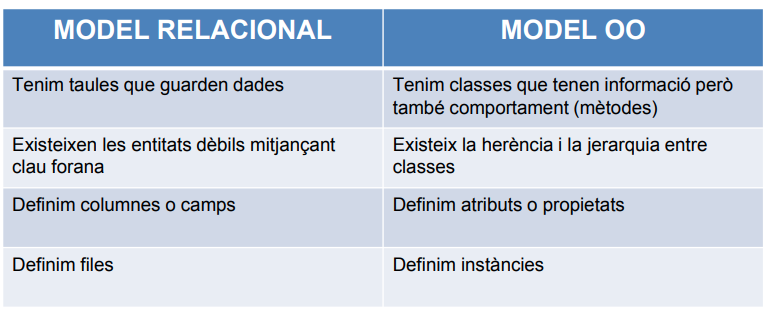
## Introducció

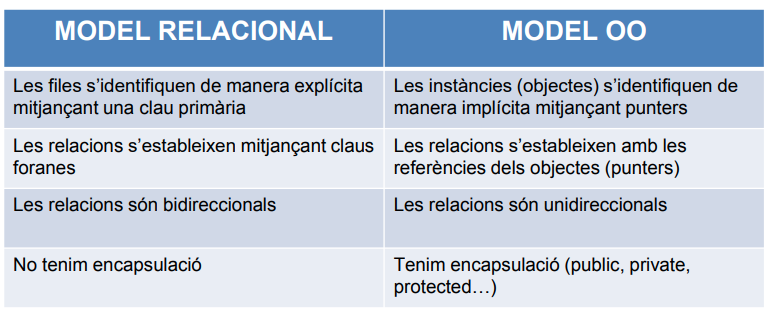
Fins ara hem vist el que s’anomenen bases de dades relacionals i els gestors que ens permeten treballar amb elles (Mysql o Oracle)

Per a la majoria de sistemes, aquestes bases de dades són suficients

Però… i el que hem estudiat a Orientació a Objectes, com es guarda en una base de dades relacional?

Entre la teoria OO i la relacional existeixen algunes diferències importants:





A la industria han aparegut diferents sistemes per treballar amb bases de dades orientades a objectes. Aquests sistemes s’anomenen SGBDOO (sistemes gestors de bases de dades orientat a objectes).

Pràcticament cap d’aquests sistemes ha quallat a la indústria del software, segurament degut al fet que els SGBDR són models matemàtics molt forts i consistents .

Així ha aparegut el model SGBDOR (sistemes gestors de bases de dades Objecte relacional) com un intermedi.

Entre els sistemes SGBDOR més populars estan Oracle i PostgreSQL .

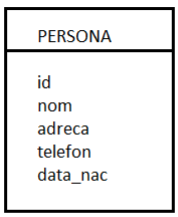
En realitat són SGBDR amb un complement per poder treballar amb OO.

## Tipus definits per l’usuari

* A banda dels tipus de dades stàndar, a Oracle podem definir els nostres tipus de dades.
* A més es reconeix el concepte d’objecte, i per tant, podrem crear objectes que tindran un identificador únic (OID: Object Identifier)
* Aquests OID’s es podran fer servir per representar relacions d’associació i agregació
* També podrem crear mètodes que s’associaran als tipus creats
* Per definir els nostres tipus de dades disposem de
  + **Tipus objecte:** per representar objectes. Tenen
    - Un nom
    - Un conjunt d’atributs
    - Un conjunt de mètodes escrits en PL/SQL
  + **Tipus col·lecció:** Ens serviran per representar relacions de tipus 1:N.

Els tipus objecte són les plantilles per als objectes.

* Imagina que volem crear una classe persona en java



A Oracle, començarem definint un tipus. Això es fa amb la instrucció:

| **CREATE** **OR** **REPLACE** **TYPE** Nom **as** **OBJECT**( ...definició ); / |
| --- |

| CREATE or replace TYPE persona\_t AS OBJECT (  id NUMBER,  nom VARCHAR2(100),  adreca varchar2(200),  telefon VARCHAR2(10),  data\_nac DATE ); / |
| --- |

Crear objecte-composició:

CREATE OR REPLACE TYPE t\_dimensions IS OBJECT(

llarg integer,

amplada integer,

alt integer

);

/

Crear l’objecte principal:

CREATE OR REPLACE TYPE t\_vehicle IS OBJECT(

matricula char(7),

marca varchar2(30),

model varchar2(30),

color varchar2(20),

dimensions t\_dimensions

);

/

Crear una tabla amb l’objecte:

CREATE TABLE vehicle of t\_vehicle;

Insertar dades a la tabla:

INSERT INTO vehicle values(

t\_vehicle('1111AAA', 'Seat', 'Ibiza', 'Blanco',

t\_dimensions(300, 200, 160)

)

);

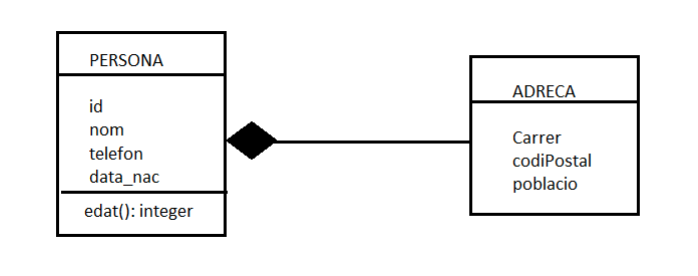
Per seleccionar, necesitarem un alies:

| SELECT v.marca, v.dimensions.largo from vehicle v; |
| --- |

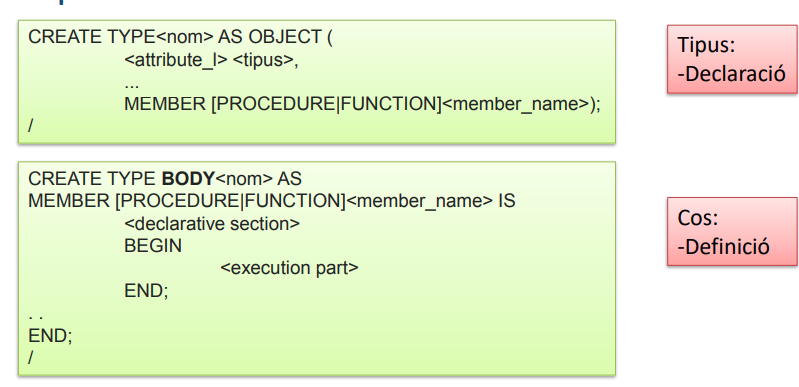
(...FALTA TEORIA…)

## 3. Métodes

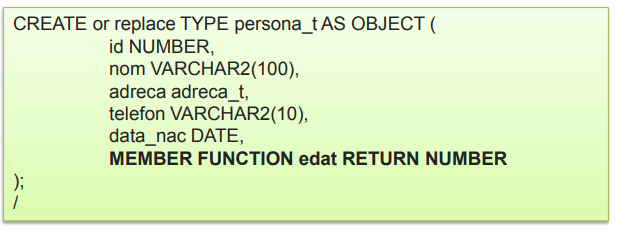
* Quan has definit classes a Java, hauràs definit també els mètodes que incorpora



* Els mètodes els declararem dins de la declaració del tipus
* Si definim un mètode anomenat edat en el tipus persona\_t i x és un objecte del mateix tipus, aleshores x.edat() calcularà la edat de la persona guardada a la variable x.
* Si un tipus té mètodes, aquests es declaren en el tipus, però es defineixen en una secció separada anomenada cos



* Elimina la taula persona per poder refer el tipus:



create or replace type persona\_t as object(

id integer,

nom varchar2(100),

adreca adreca\_t,

telefon telefon\_t,

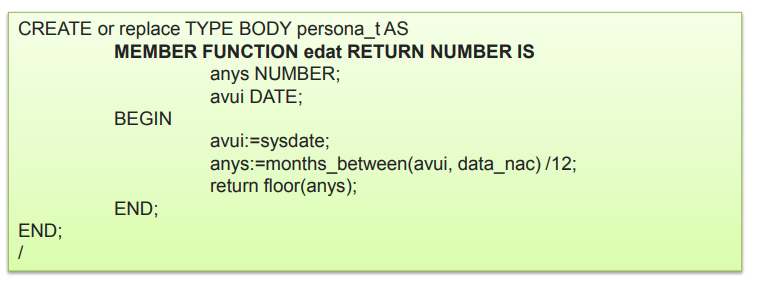
data\_nac date,

member fuction edat return integer

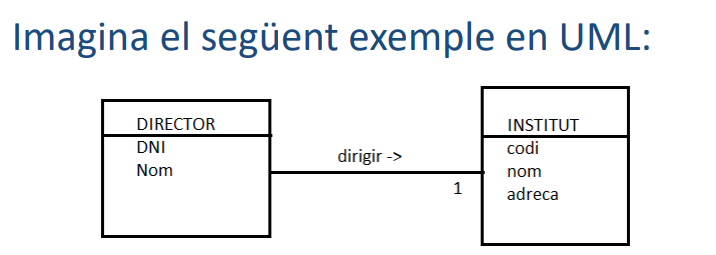
);

/

Hará falta definir la función:



## 4. RELACIONS



CREATE OR REPLACE TYPE t\_director IS OBJECT(

DNI char(9),

Nom varchar2(40)

);

CREATE OR REPLACE TYPE t\_instituto IS OBJECT(

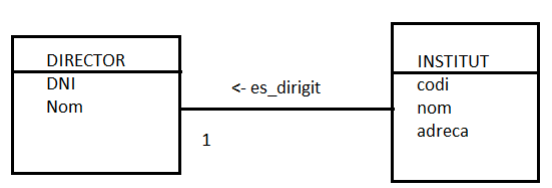
codi char(8),

nom varchar2(30),

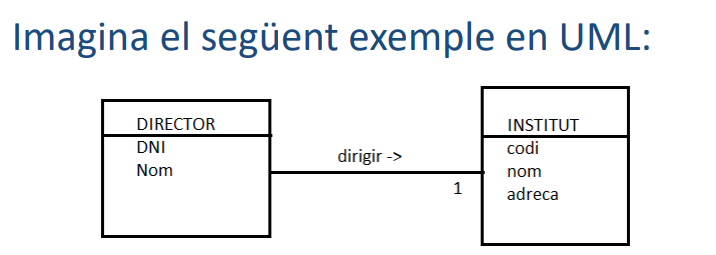
adreca varchar(50)

);

/



Per implementar la relació dirigeix o és dirigit: (depèn de com sigui el UML, ficarem la ref a Director o Institut)



| CREATE OR REPLACE TYPE t\_instituto IS OBJECT(  codi char(8),  nom varchar2(30),  adreca varchar(50) ); / CREATE OR REPLACE TYPE t\_director IS OBJECT(  DNI char(9),  nom varchar2(40),  institut REF t\_insti *--si tinc aquí la ref. s'haurà de crear primer l'altre* ); / |
| --- |

Crear les taules i inserir dades a Coles

| *--Crear les taules* CREATE TABLE dires of t\_director; CREATE TABLE coles of t\_instituto;  *--Insertar dades a coles* INSERT INTO coles VALUES ('1', 'Joan D''Austria', 'Carrer Dos' ); INSERT INTO coles VALUES ('2', 'Massallera', 'Carrer Tres'); |
| --- |

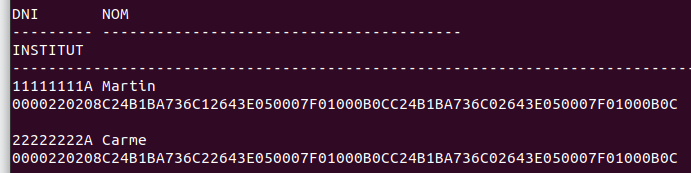
Per inserir dades a Director necessitem la referencia del institut:

| *--Inserir dades a director (necessitem referencia de l’institut)* SELECT ref(c) from coles c; *--Retorna el punter*  *-- REF(C)* *-- --------------------------------------------------------------------------------* *-- 0000280209C24B1BA736C12643E050007F01000B0CC24B1BA736C02643E050007F01000B0C0040B1* *-- 990000*  *-- 0000280209C24B1BA736C22643E050007F01000B0CC24B1BA736C02643E050007F01000B0C0040B1* *-- 990001* |
| --- |

Per inserir llavors, podem fer un select:

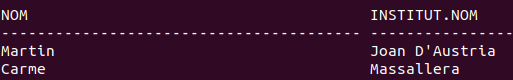
| INSERT INTO dires VALUES ('11111111A', 'Martin', (SELECT ref(c) from coles c where codi=1)); INSERT INTO dires VALUES ('22222222A', 'Carme', (SELECT ref(c) from coles c where codi=2)); |
| --- |

El problema és:



Per accedir:

| select d.nom, d.institut.nom from dires d; |
| --- |



### Crear relació persona - vehicle (1)

create or replace type persona\_t as object(

id integer,

nom varchar2(100),

adreca adreca\_t,

telefon telefon\_t,

data\_nac date,

vehicle ref t\_vehicle,

member function edat return integer

);

/