UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

Escola Politècnica Superior d'Enginyeria
de Vilanova i la Geltrú

Grau en Enginyeria Informàtica

Disseny i Administració de Bases de Dades

Problemes de Disseny resolts

Jordi Esteve

Dept. de Ciències de la Computació (CS). EPSEVG. UPC

Hivern 2020

---

1. Factoria

Una factoria immensa té una elevada quantitat de màquines. De cadascuna tenim el número amb què està inventariada i el seu fabricant.

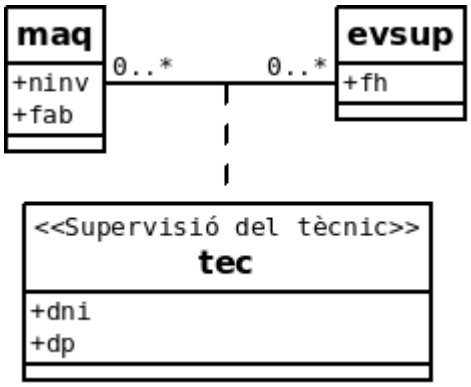
Periòdicament, un tècnic competent supervisa cada màquina.

Cada tècnic, de qui coneixem el DNI i les seves dades personals, coneix la distribució de la seva setmana laboral en franges horàries amb la indicació de quina màquina ha de supervisar en cadascuna d'aquestes franges.

1.1. 1ª solució usant diagrama de classes UML

Una primera aproximació al problema consisteix en diferenciar tres entitats: *Màquina*, *Event de supervisió* (franja horària) i *Supervisió del tècnic*. Podem raonar que hi ha una associació entre màquina i franja horària: Una màquina es supervisada en diferents franges horàries i a cada franja horària es supervisen diferents màquines. El concepte associatiu entre la màquina i la franja horària seria la supervisió d'un tècnic.

Això ens portaria a dissenyar un diagrama com aquest:



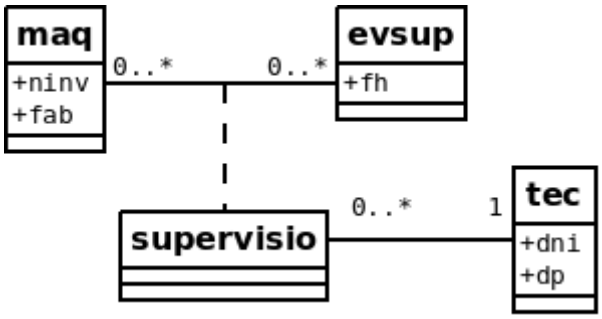
Que donaria el següent esquema relacional:

```
CREATE TABLE maq
        (ninv int PRIMARY KEY NOT NULL, fab text);
CREATE TABLE evsup
        (fh text PRIMARY KEY NOT NULL);
CREATE TABLE tec (
        dni int NOT NULL, dp text,
        fh text NOT NULL     REFERENCES evsup ON UPDATE CASCADE,
        ninv int NOT NULL   REFERENCES maq ON UPDATE CASCADE,
        PRIMARY KEY(fh,ninv), UNIQUE(fh,dni));
```

Aquesta solució no és gaire encertada perquè no separa dos entitats força evidents: El Tècnic i la Supervisió del tècnic.

1.2. 2ª solució usant diagrama de classes UML

Una altra possible solució més elegant és diferenciar el Tècnic i la Supervisió del tècnic en dues entitats diferents:



Que donaria el següent esquema relacional:

```
CREATE TABLE tec
        (dni int PRIMARY KEY NOT NULL, dp text);
CREATE TABLE maq
        (ninv int PRIMARY KEY NOT NULL, fab text);
CREATE TABLE evsup
        (fh text PRIMARY KEY NOT NULL);
CREATE TABLE supervisio (
        fh text NOT NULL    REFERENCES evsup ON UPDATE CASCADE,
        ninv int NOT NULL   REFERENCES maq ON UPDATE CASCADE,
        dni int NOT NULL    REFERENCES tec ON UPDATE CASCADE,
        PRIMARY KEY(fh,dni), UNIQUE(fh,ninv));
```

Aquesta solució evita la repetició de les dades personals dels tècnics dins de les supervisions de la 1ª solució. Dit d'una altra manera: Ara podem definir el dni dels tècnics com a clau primària.

1.3. 3ª solució usant dependències funcionals

De l'enunciat del problema podem deduir les següents dependències funcionals:

- DNI → D.P.
- N.Inv. → Fab
- DNI F.H. → N.Inv.
- N.Inv. F.H. → DNI;

Com podem veure, l'esquema relacional de la primera solució no estava ni en BCNF ni en 3NF doncs la dependència DNI → D.P. no corresponia a cap clau. En canvi la segona solució estava normalitzada en BCNF i, per tant, també en 3NF doncs totes les dependències corresponen a claus.

Si volem normalitzar la relació R(DNI, D.P., N.Inv, Fab, F.H.) caldria trossejar-la en les següents relacions:

R1(DNI, D.P)

R2(N.Inv, Fab)

R3(DNI, F.H., N.Inv)

Ara aquest esquema està en BCNF i, per tant, també en 3NF doncs totes les dependències corresponen a claus (primàries/alternatives). L'esquema relacional que se'n deriva seria aquest:

```
CREATE TABLE tec
        (dni int PRIMARY KEY NOT NULL, dp text);
CREATE TABLE maq
        (ninv int PRIMARY KEY NOT NULL, fab text);
CREATE TABLE evsup (
        fh text NOT NULL,
        dni int NOT NULL     REFERENCES tec ON UPDATE CASCADE,
        ninv int NOT NULL   REFERENCES maq ON UPDATE CASCADE,
        PRIMARY KEY(fh,dni), UNIQUE(fh,ninv));
```

2. Magatzems

Volem una BD que contingui dades sobre la nostra xarxa de magatzems que tenim distribuïda per tot Catalunya, i sobre els nostres clients.

De cada magatzem volem tenir el seu nom (és un codi alfabètic de 6 lletres que, és clar, és diferent per a cada magatzem) el seu telèfon, l'adreça, la superfície, quins empleats hi treballen i quin d'ells consta com a director del magatzem.

De cadascun dels empleats dels nostres magatzems volem tenir el seu NIF, el seu nom, el seu telèfon, la data de naixement i la data de contractació. També volem saber qui és el seu supervisor, si en té (ser supervisor d'algú és independent de constar o no com a director del magatzem).

Cadascun dels nostres magatzems pot servir productes a un conjunt de poblacions. Volem mantenir en la BD la llista de totes les poblacions catalanes i, per a cadascuna, volem saber quants habitants té, la seva categoria (segons una codificació que les classifica en tres classes, A, B i C) i quin dels nostres magatzems s'hi troba situat (si n'hi ha algun, però no en tindrem mai més d'un). També ens interessa saber quin és el magatzem assignat a cada població (no sempre és el magatzem que s'hi troba situat). Cada població sol tenir un sol magatzem assignat per servir-la, però pot no estar assignada a cap magatzem o fins i tot pot passar, encara que rarament, que estigui assignada a més d'un magatzem. També ens interessa tenir la data en què el magatzem va ser assignat per servir la població.

Dels nostres clients (tots ells petits comerços familiars) ens interessa tenir, en aquesta BD, només el seu NIF, el seu nom i el seu telèfon. I òbviament volem saber a quina població està situat cada client.

Aquesta proposta de solució és deguda al professor -ja jubilat- de l'EPSEVG Rafael Camps amb algunes millores aportades pels professors José Luís Balcazar i Jordi Esteve.

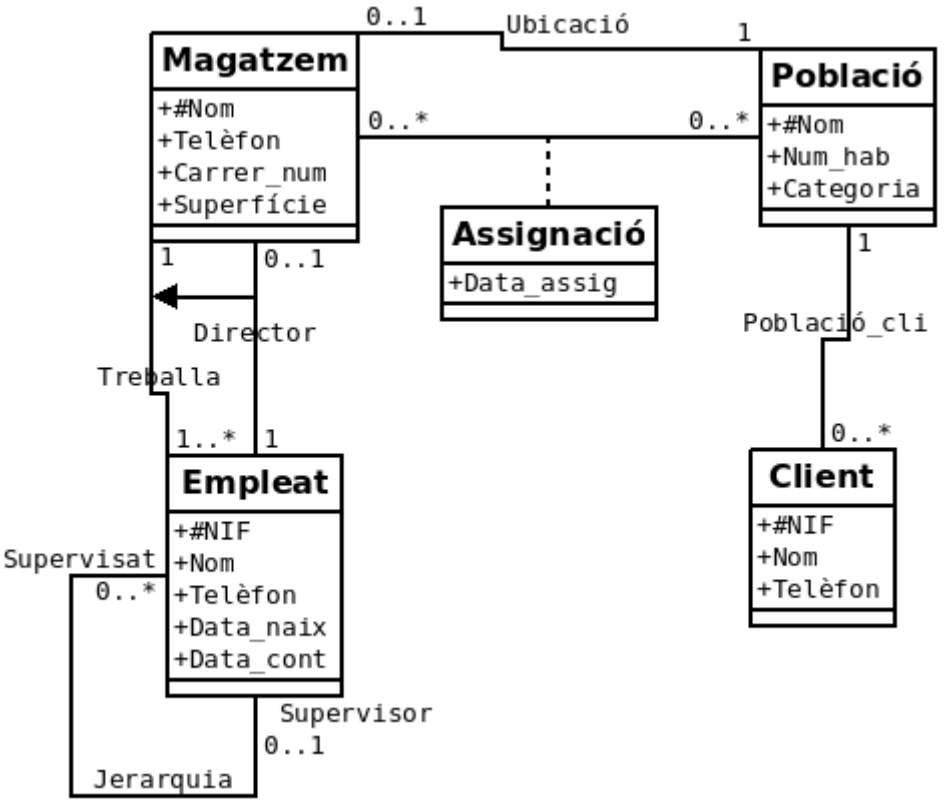
2.1. Primera part

Hem de crear un model de classes UML a partir de l'enunciat del problema.

No sembla que hi hagi molts dubtes sobre les classes (o entitats) del nostre model. Hem de tenir dades dels següents tipus d'objectes, que a més són identificables per atributs propis:

Magatzem, *Població*, *Empleat*, *Client*

Per tant tindrem aquestes quatre classes. Podem tenir dubtes sobre si hem de modelar o no les assignacions dels magatzems a les poblacions com classes. Però com les assignacions ens parlen de parelles magatzem/població i a més no tenen identificador propi, sembla més raonable modelar-les com una associació (interrelació) entre magatzems i poblacions. Aquesta associació tindrà un atribut, la data d'assignació, de manera que serà una classe associativa.



Les associacions *Treballa* i *Població_cli* no necessiten justificació doncs són evidents. L'associació *Director* és una especialització de l'associació *Treballa* doncs el director és un dels empleats del magatzem. El 0..1 al costat *Magatzem* vol dir que un empleat és, o no, director d'un (i només un) magatzem.

La dependència jeràrquica entre empleats es representa per mitjà de l'associació recursiva *Jerarquia*. El 0..1 expressa que cada empleat pot tenir com a màxim 1 supervisor i pot no tenir-ne cap. Per a més claredat hem expressat els dos rols: *Supervisor* i *Supervisat.*

Una cosa és la població en la qual està ubicat un magatzem i una altra, com diu l'enunciat, les poblacions a les que serveix (o sigui les que té assignades). Per això hem disposat dues associacions diferents, independents: *Ubicació* i *Assignació*. El 0..1 a *Ubicació* expressa el fet que hi ha poblacions sense magatzem, però que si en té un no pot tenir més. El fet que ens diguin que hi ha poques poblacions que estan assignades a més d'un magatzem, o que hi ha pocs magatzems que no serveixen a la població en la qual estan ubicats, no és raó suficient per ignorar-ho.

Observar que a *Magatzem*, per expressar la direcció només hem posat el *Carrer_num*, però no la població. Aquesta ja queda expressada per l'associació *Ubicació*. Si poséssim un atribut *Població*, aquest podria ser diferent a la població a la que el magatzem està associat per mitjà d'*Ubicació*. Per tant el valor d'aquest atribut podria ser incoherent amb l'associació (com sempre pot passar quan tenim qualsevol redundància).

Nota 1: L'enunciat no aclareix totalment si tot magatzem necessita algun empleat (per exemple, com a mínim un director) i si tot magatzem està assignat a alguna població. Mentre no es tingui més informació, hi ha diverses opcions defensables.

2.2. Segona part

Ara hem de convertir el model de classes UML de la primera part a un model relacional.

És més senzill pensar sobre un esquema relacional informal i simple, que sobre un esquema detallat escrit en SQL. Per això deixarem per al final el pas a SQL.

Primer escriurem l'esquema de les cinc taules, quatre per a les classes i una per a la classe associativa. Destacarem en negreta les claus primàries (PK), en cursiva les claus alternatives (UK) i subratllades les foranes (FK). A la taula *Assignació* li hem de posar com a clau primària el parell de columnes clau primària de les dues taules a les que interrelaciona, però com totes dues tenen el mateix nom (precisament Nom!) els hem canviat.

Magatzem (**Nom**, Telèfon, Carrer_num, Superfície)

Població (**Nom**, Num_hab, Categoria)

Assignació (**Població_assig**, **Magatzem_assig**, Data_assig)
FK de *Població* FK de *Magatzem*

Empleat (**NIF**, Nom, Telèfon, Data_naix, Data_cont)

Client (**NIF**, Nom, Telèfon)

Ens falta representar les altres cinc associacions.

Les associacions *Treballa* i *Població_cli* són 1 * i per això és suficient posar en el costat del * una columna FK que faci referència a la clau primària de la taula del costat 1. Aquestes columnes hauran de tenir prohibits els valors nuls perquè així es compleixi la cardinalitat mínima = 1.

L'associació *Ubicació* es pot representar afegint a la taula *Magatzem* una columna *Pobl_ubi* que ens digui quina és la població (només una, com ens exigeix la cardinalitat màxima en el costat Població) on està ubicat. Perquè la cardinalitat mínima sigui 1 haurem declarar aquesta columna com NOT NULL. Però, com aconseguir l'1 del costat *Magatzem*? O sigui, com exigir que en una població hagi un magatzem com a màxim? Una possible solució és declarant la columna Pobl_ubi com a clau alternativa (UNIQUE) o sigui que no pot tenir valors que apareguin en més d'un magatzem.

Per a l'associació *Director* podrem emprar una solució idèntica a la que acabem d'explicar per a la *Ubicació*.

La *Jerarquia* es pot representar fàcilment afegint a la taula *Empleat* una columna *Supervisor* que sigui clau forana de la pròpia taula *Empleat*. Aquesta columna haurà d'acceptar nuls per acceptar empleats sense supervisor.

Com a resultat de tot això ens queda l'esquema següent:

Magatzem (**Nom**, Telèfon, Carrer_num, Superfície, *Pobl_ubi*, *Empl_dir*)
Unique, Not null Unique, Not null
FK de Població FK d'Empleat

Població (**Nom**, Num_hab, Categoria)

Assignació (**Població_assig**, **Magatzem_assig**, Data_assig)
FK de Població FK de Magatzem

Empleat (**NIF**, nom, Data_naix, Data_cont, Treballa, Supervisor)
FK de Magatzem FK d'Empleat
Not null

Client (**NIF**, Nom, Telèfon, Població)
FK de Població, Not null

Ara només ens queda transcriure aquest esquema al llenguatge SQL. A més afegirem els tipus de les dades i restriccions de domini complementàries.

```
CREATE TABLE Magatzem (
    Nom             CHAR(6)         NOT NULL  PRIMARY KEY,
    Telefon         INTEGER,
    Carrer_num      VARCHAR(25)     NOT NULL,
    Superficie      INTEGER,
    Pobl_ubi        CHAR(15)        UNIQUE NOT NULL REFERENCES Poblacio,
    Empl_dir        INTEGER         UNIQUE NOT NULL REFERENCES Empleat
);

CREATE TABLE Poblacio (
    Nom             CHAR(15)        NOT NULL  PRIMARY KEY,
    Num_hab         INTEGER,
    Categoria       CHAR            CHECK (Categoria IN ( "A", "B", "C" ) )
);

CREATE TABLE Assignacio (
    Poblacio_assig  CHAR(15)        REFERENCES Poblacio,
    Magatzem_assig  CHAR(6)         REFERENCES Magatzem,
    Data_assig      DATE,
    PRIMARY KEY (Poblacio_assig, Magatzem_assig)
);

CREATE TABLE Empleat (
    NIF             INTEGER         NOT NULL PRIMARY KEY,
    Nom             VARCHAR(35)     NOT NULL,
    Telefon         INTEGER,
    Data_naix       DATE,
    Data_cont       DATE,
    Treballa        CHAR(6)         NOT NULL REFERENCES Magatzem,
    Supervisor      INTEGER         REFERENCES Empleat,
    CHECK (Data_naix < Data_cont)
);
```

```
CREATE TABLE Client (
    NIF             INTEGER         NOT NULL PRIMARY KEY,
    Nom             VARCHAR(25)     NOT NULL,
    Telefon         INTEGER,
    Poblacio_cli    CHAR(15)        NOT NULL REFERENCES Poblacio
);
```

Nota 2: En aquest model relacional no ha quedat reflectida l'especialització de l'associació *Director* a partir de l'associació *Treballa*; la lògica de l'aplicació que usi aquest model relacional s'haurà d'encarregar de fer efectiva aquesta especialització. Una opció seria substituir la columna *Empl_dir* de la taula *Magatzem* per un booleà dins de la taula *Empleat* que indiqués si l'empleat és o no és director del magatzem on treballa, però llavors el model relacional deixaria de representar que cada magatzem té sempre un director.

Nota 3: Aquest codi SQL no especifica clàusules "ON DELETE" per a les claus foranes. Convé assegurar-se l'opció que per defecte apliqui el SGDB que s'estigui utilitzant per esborrats de tuples apuntades per claus foranes; la més habitual, que ja ens convé, és impedir l'esborrat. No obstant això, algunes claus foranes (per exemple, "Supervisor") admeten NULL: en aquests casos, podríem especificar "ON DELETE SET NULL "si el nostre SGBD ho admet.