Cas d'estudi

Marc Gibert Ginestà

P06/M2009/02154

UOC
www.uoc.edu

Índex

Introducció

Aquest mòdul forma part del curs "Bases de dades" de l'itinerari "Administrador web i comerç electrònic" dins del màster internacional de Programari Lliure de la Universitat Oberta de Catalunya.

El mòdul està estructurat en apartats que corresponen a la resta dels mòduls de l'assignatura, de manera que l'estudiant pot anar seguint aquest cas d'estudi a mesura que va progressant en el curs.

Encara que alguns dels mòduls ja disposen d'exercicis d'autoavaluació, el cas d'estudi presenta una visió completa d'un projecte de bases de dades i proporciona una visió pràctica de cada un.

Objectius

Els objectius que hauríeu d'assolir en acabar el treball amb aquesta unitat són els següents:

- Comprendre des d'un punt de vista pràctic els conceptes explicats en les unitats didàctiques teòriques.

- Disposar d'un model de referència per a emprendre projectes de bases de dades.

- Adquirir prou criteri per a identificar les activitats clau i prendre decisions en un projecte que impliqui l'ús de bases de dades.

1. Presentació del cas d'estudi

Acabem d'entrar a treballar en una petita empresa –ElSeuOrdinadorAMida, SL– dedicada a la venda d'ordinadors a particulars i altres empreses. Quan ens van fer l'entrevista de treball, vam comentar la nostra passió pel programari lliure i, en concret, vam posar èmfasi en el nostre coneixement dels motors de bases de dades lliures i els avantatges que podien aportar respecte dels gestors de propietat. No sabem si això va ser el que va convèncer la persona de recursos humans o no, però, en tot cas, i havent vist el resultat de la reunió que hem tingut el nostre primer dia de feina, hem d'aplicar els nostres coneixements a fons.

Ens han explicat que, fins ara, la gestió de l'empresa es duia a terme amb programes propietaris de gestió i comptabilitat, però que a causa de problemes amb l'empresa que desenvolupava aquests programes, s'està considerant la migració de la gestió administrativa i d'operacions a entorns oberts. Per acabar de decidir-se, ens proposen que comencem per renovar el sistema de gestió de peticions i incidències per part dels clients, de manera que estigui basada en programari lliure.

Actualment, les peticions i incidències es reben telefònicament, per correu electrònic o en persona en algun dels locals que té l'empresa. La persona que atén el telèfon o llegeix els correus electrònics planteja una sèrie de preguntes al client i escriu en una plantilla de document les respostes. A continuació, s'imprimeix el document i es deixa en una safata que recullen els tècnics cada matí.

A mesura que els tècnics van avançant en la solució de la incidència (o han trucat al client per demanar més dades), van apuntant les accions i l'estat del problema al full que van recollir, fins que la incidència queda resolta. En aquell moment, la deixen en una safata que recull cada matí el personal d'administració, que es posa en contacte amb el client i factura l'import corresponent a les hores de feina i els components substituïts.

És obvi que aquest sistema presenta nombroses deficiències i que el rendiment, tant dels tècnics com del personal administratiu i d'atenció al client, podria augmentar enormement si molts d'aquests processos fossin automàtics, estiguessin centralitzats i, si fos possible, connectats amb la resta del sistema d'informació de l'empresa.

Aquest és, a grans trets, el problema que se'ns planteja i que utilitzarem com a cas d'estudi per a aplicar els coneixements adquirits durant el desenvolupament del curs.

2. El model relacional i l'àlgebra relacional

Havent vist el projecte plantejat, decidim fer les coses ben fetes per a, de passada, impressionar el nostre cap amb els nostres coneixements en bases de dades. El primer pas serà presentar-li un document que descrigui el model relacional que utilitzarem, en el qual inclourem algunes consultes de mostra perquè pugui comprovar què serà capaç de fer amb el nostre projecte quan estigui acabat.

2.1. Determinar les relacions

En primer lloc determinarem les relacions, els seus atributs i els dominis de cada un d'aquests:

```
PETICIO(referencia, client, resum, estat, datarecepcio, datainici, datafi,
tempsempleat)
NOTA_PETICIO(peticio, nota, data, empleat)
MATERIAL_PETICIO(nommaterial, peticio, quantitat, preu)
CLIENT(nom, nif, telefon, email)
EMPLEAT(nom, nif)
```

En la relació `PETICIO`, hem decidit que convindria tenir una referència interna de la petició, que ens ajudarà en parlar-ne amb el client (si en tingués diverses d'obertes) i evitarà confusions a l'hora de treballar. La resta d'atributs són bastant explícits.

Com que una petició pot evolucionar amb el temps, a mesura que es demanen més dades al client, la incidència va evolucionant, etc., hem creat les relacions `NOTA_PETICIO` i `MATERIAL_PETICIO` per reflectir-ho.

També hem hagut de definir les relacions `CLIENT` i `TREBALLADOR` per a poder relacionar-les amb les peticions i les notes que es vagin generant durant la seva resolució.

A continuació definirem els dominis dels atributs:

```
PETICIO:
domini(referencia)=números
domini(client)=NIF
```

```
domini(resum)=text
domini(estat)=estats
domini(datarecepcio)=dataihora
domini(datainici)=dataihora
domini(datafi)=dataihora
domini(tempsempleat)=horesiminuts

NOTA_PETICIO:
domini(peticio)=números
domini(nota)=text
domini(fecha)=dataihora
domini(empleat)=NIF

MATERIAL_PETICIO:
domini(nommaterial)=nomMaterial
domini(peticio)=números
domini(preu)=preu
domini(quantitat)=números

CLIENT:
domini(nom)=nomClient
domini(nif)=NIF
domini(telefon)=telèfons
domini(email)=emails

EMPLEAT:
domini(nom)=nomEmpleat
domini(nif)=NIF
```

En definir els dominis de cada atribut, ja ens hem avançat a la presa d'algunes decisions: en decidir, per exemple, que el domini de l'atribut `treballador` en la relació `NOTA_PETICIO` és `NIF`, implícitament estem determinant que la clau primària de la relació `TREBALLADOR` serà del domini NIF i que usarem un atribut d'aquest domini per a referir-nos-hi.

Aquest procés descrit indicant directament el seu resultat normalment és fruit d'una revisió de les entitats a mesura que es van definint i se n'analitzen les necessitats.

Nota

La regla d'integritat del model corresponent a la clau primària comportarà que no hi hagi dues notes sobre la mateixa petició fetes en la mateixa data i hora per part del mateix treballador, la qual cosa és perfectament lícita i coherent.

2.2. Definició de claus

Encara que algunes claus ja s'intueixen a partir dels atributs de les relacions, les determinarem per completar el cas.

```
PETICIO:
Claus candidates: {referencia}
Clau primaria: {referencia}

NOTA_PETICIO:
Claus candidates: {peticio,data,empleat}
Clau primaria: {peticio,data,empleat}

MATERIAL_PETICIO:
Claus candidates: {nommaterial,peticio}
Clau primaria: {nommaterial,peticio}

CLIENT:
Claus candidates: {nif}
Clau primaria: {nif}

EMPLEAT:
Claus candidates: {nif}
Clau primaria: {nif}
```

En tots els casos només tenim una clau candidata i, per tant, no hi ha dubtes a l'hora d'escollir la clau primària. Això no ha de ser així necessàriament: en la relació TREBALLADOR, podríem haver inclòs més atributs (número de la Seguretat Social, un número de treballador intern, etc.) que serien claus candidates susceptibles de ser clau primària.

Ara podem reescriure les relacions:

```
PETICIO(referencia, client, resum, estat, datarecepcio, datainici, datafi, tempsem-
pleat)
NOTA_PETICIO(peticio, nota, data, empleat)
MATERIAL_PETICIO(nommaterial, peticio, quantitat, preu)
CLIENT(nom, nif, telefon, email)
EMPLEAT(nom, nif)
```

Les claus foranes ja s'intueixen a partir de les relacions, tot i que les comentarem per completar el cas:

NOTA_PETICIO:

Té de clau forana l'atribut `{peticio}`, que estableix la relació (i pertany al mateix domini) amb l'atribut `{referencia}` de la relació `PETICIO`.

També té la clau forana `{treballador}`, la qual estableix la relació amb `TREBALLADOR` a partir de la seva clau primària `{nif}`.

MATERIAL_PETICIO:

Té de clau forana l'atribut `{peticio}`, que estableix la relació (i pertany al mateix domini) amb l'atribut `{referencia}` de la relació `PETICIO`.

2.3. Regles de integritat

En aquest punt, no cal preocupar-se per les regles d'integritat del model que tracten sobre la clau primària, ja que ens seran imposades en el moment de crear les taules en l'SGBD.

És convenient, no obstant això, fixar les decisions sobre la integritat referencial; en concret, què farem en cas de restricció. Així doncs, per a cada relació que té una clau primària referenciada des d'una altra, haurem de decidir quina política es pot aplicar en cas de modificació o esborrament:

PETICIO

- Modificació de l'atribut `{referencia}` referenciat des de `NOTA_PETICIO` i `MATERIAL_PETICIO`: aquí podem optar per la restricció o per l'actualització en cascada (més còmoda, encara que no tots els SGBD la implementen, com veurem més endavant).

- Esborrament de l'atribut `{referencia}`. Aquí optarem per una política de restricció. Si la petició té notes associades o materials, significa que hi ha hagut alguna activitat i, per tant, no hauríem de poder esborrar-la. Si es vol anul·lar-la, ja establirem un estat d'aquesta que ho indiqui.

CLIENT

- Modificació de l'atribut `{nif}` referenciat des de `PETICIO`. És probable que si un client canvia de NIF (per un canvi del tipus de societat, etc.) desitgem mantenir les seves peticions. Aquí la política ha de ser d'actualització en cascada.

- Esborrament de l'atribut `{nif}`. És possible que si volem esborrar un client sigui perquè hem acabat tota relació amb ell i, per tant, és coherent utilitzar aquí la política d'anul·lació.

TREBALLADOR

- Modificació de l'atribut `{nif}` referenciat des de `NOTA_PETICIO`. No és probable que un treballador canviï el seu NIF, llevat de cas d'error. Tot i així, en cas que es produeixi, és preferible l'actualització en cascada.

- Esborrament de l'atribut `{nif}`. Encara que eliminem un treballador si acaba la seva relació amb l'empresa, no hauríem d'eliminar les seves notes. La millor opció és la restricció.

2.4. Àlgebra relacional

Per a provar el model, una bona opció és intentar realitzar algunes consultes sobre ell i veure si obtenim els resultats desitjats. En el nostre cas, realitzarem les consultes següents:

- Obtenció d'una petició juntament amb les dades del client:

```
R:= PETICIO [client=nif] CLIENT
```

- Obtenció d'una petició amb totes les seves notes:

```
NP(peticionota, nota, datanota, empleat):=NOTA_PETICIO(peticio, nota, data, empleat)
R:=PETICIO[peticio=peticionota]NOTA_PETICIO
```

- Obtenció de les dades de tots els treballadors que han participat en la petició 5:

```
NP:=NOTA_PETICIO[peticio=5]
RA:=EMPLEAT[nif=empleat]NP
R:=RA[nom,nif]
```

Exercici

Us suggerim que intenteu més operacions sobre el model per a familiaritzar-vos-hi.

3. El llenguatge SQL

Una vegada acabat el model relacional, decidim completar la documentació que havíem fet amb les sentències SQL corresponents. Així, veurem en què es concretarà el model relacional.

Com que encara no sabem en quin sistema gestor de base de dades implantarem la solució, decidim simplement anotar les sentències segons l'estàndard SQL92, i, posteriorment, ja examinarem les particularitats del sistema gestor escollit per a adaptar-les.

3.1. Sentències de definició

Atenció

En alguns casos és convenient la definició de dominis per a facilitar la feina posterior de manteniment de la coherència de la base de dades. No és aconsellable definir dominis per a cada domini relacional, però sí en els casos en els quals una columna pot prendre una sèrie de valors determinats.

- Creació de la base de dades

```
CREATE SCHEMA GESTIO_PETICIONS;
```

- Definició de dominis

```
CREATE DOMAIN dom_estats AS CHAR (20)
   CONSTRAINT estats_valids
   CHECK (VALUE IN ('Nova','Es necessiten més dades','Acceptada','Confirmada',
'Resolta',
'Tancada'))
   DEFAULT 'Nova';
```

Atenció

Aquí haurem de tenir en compte les regles d'integritat, ja que caldrà explicitar la política escollida com a restricció.

- Creació de les taules

```
CREATE TABLE PETICION ( referencia INTEGER NOT NULL,
   client INTEGER NOT NULL,
   resum CHARACTER VARYING (2048),
   estat dom_estats NOT NULL,
   datarecepcio TIMESTAMP NOT NULL,
   datainici TIMESTAMP, datafi TIMESTAMP,
   tempsempleat TIME, PRIMARY KEY (referencia),
   FOREIGN KEY client REFERENCES CLIENT(nif)
      ON DELETE CASCADE
      ON UPDATE CASCADE,
   CHECK (datarecepcio < datainici),
   CHECK (datainici < datafi) );
```

```
CREATE TABLE NOTA_PETICIO ( peticio INTEGER NOT NULL,
   nota CHARACTER VARYING (64000),
   data TIMESTAMP NOT NULL,
   empleat CHARACTER (9),
   FOREIGN KEY (peticio) REFERENCES PETICION(referencia)
      ON DELETE NO ACTION
      ON UPDATE CASCADE,
   FOREIGN KEY (empleat) REFERENCES EMPLEAT(nif)
      ON DELETE NO ACTION
      ON UPDATE CASCADE );

CREATE TABLE MATERIAL_PETICIO ( nommaterial CHARACTER
VARYING (100) NOT NULL,
   peticio INTEGER NOT NULL,
   preu DECIMAL(8,2),
   quantitat INTEGER,
   FOREIGN KEY (peticio) REFERENCES PETICIO(referencia)
      ON DELETE NO ACTION
      ON UPDATE CASCADE );

CREATE TABLE CLIENT ( nom CHARACTER VARYING (100) NOT
NULL,
   nif CHARACTER (9) NOT NULL,
   telefon CHARACTER (15),
   email CHARACTER (50),
   PRIMARY KEY (nif) );

CREATE TABLE EMPLEAT (nom CHARACTER VARYING (100) NOT
NULL,
   nif CHARACTER (9) NOT NULL,
   PRIMARY KEY (nif) );
```

- Creació de vistes

Funció de vistes

Les vistes agilitaran les consultes que preveiem que seran més freqüents.

– Peticions pendents:

```
CREATE VIEW peticions_pendents (referencia, nom_client, resum, estat, durada,
datarecepcio) AS (
SELECT P.referencia, C.nom, P.resum, P.estat, (P.datainici P.datarecepcio),
P.datarecepcio
FROM PETICIO P JOIN CLIENT C ON P.client = C.nif
WHERE estat NOT IN ('Resolta','Tancada') ORDER BY datarecepcio )
```

– Temps i preu dels materials emprats per a les peticions acabades el mes en curs:

```
CREATE VIEW peticions_terminades (referencia, nom_client, resum, temps_empleat,
import_materials) AS (
     SELECT P.referencia, C.nom, P.resum, P.tempsempleat, SUM(M.preu)
     FROM PETICIO P, CLIENT C, MATERIAL_PETICIO M
     WHERE P.client=C.nif AND M.peticio=P.referencia AND estat='Resolta'
     GROUP BY P.referencia)
```

3.2. Sentències de manipulació

A continuació, decidim indicar algunes sentències de manipulació corrents per a completar la documentació. D'aquesta manera, quan comencem el desenvolupament, tindrem molt més clares aquestes operacions sobre la base de dades:

- Nou client:

```
INSERT INTO CLIENT VALUES ('Joan Pérez','42389338A','912223354','joanperez@gmail.com');
```

- Nova petició:

```
INSERT INTO PETICIO VALUES (5,'42389338A', 'No se li engega l\'ordinador',
'Nova',CURRENT_TIMESTAMP,NULL,NULL,NULL);
```

- Canvi d'estat de la petició, afegim una nota i un material:

```
UPDATE PETICIO SET estat='Acceptada' WHERE referencia=5;
INSERT INTO NOTA_PETICIO VALUES (5,'Sembla un problema del disc dur. L\'examinarem més
a fons.',CURRENT_TIMESTAMP,'35485411G');

INSERT INTO MATERIAL_PETICIO VALUES ('Disc dur 20Gb',5,250.00,1);
```

- Materials sol·licitats en la petició 5:

```
SELECT nommaterial, quantitat, preu FROM MATERIAL_PETICIO WHERE peticio=5
```

- Nombre de peticions obertes del client '42389338 A':

```
SELECT COUNT(*) FROM PETICIO WHERE client='42389338A' AND estat NOT IN ('Resolta',
'Tancada');
```

La creació de vistes de l'apartat anterior ens ha mostrat també algunes consultes complexes que repetim a continuació:

- Peticions obertes:

```
SELECT P.referencia, C.nom, P.resum, P.estat, (P.datainici P.datarecepcio),
P.datarecepcio FROM PETICIO P JOIN CLIENT C ON P.client = C.nif WHERE estat NOT IN
('Resolta','Tancada') ORDER BY datarecepcio;
```

- Temps i preu dels materials emprats per a les peticions acabades el mes en curs:

```
SELECT P.referencia, C.nom, P.resum, P.tempsempleat, SUM(M.preu) FROM PETICIO P, CLIENT
C, MATERIAL_PETICIO M WHERE P.client=C.nif AND M.peticio=P.referencia AND estat='Resolta'
GROUP BY P.referencia;
```

Finalment, practicarem amb les consultes que hem fet en àlgebra relacional en l'apartat anterior:

- Obtenció d'una petició juntament amb les dades del client:

```
R:= PETICIO [client=nif] CLIENT
SELECT * FROM PETICIO JOIN CLIENT ON PETICIO.client=CLIENT.nif;
```

- Obtenció d'una petició amb totes les seves notes:

```
NP(peticionota,nota,datanota,empleat):=NOTA_PETICIO (peticio,nota,data,empleat)
R:=PETICIO[referencia=peticionota]NP
SELECT PETICIO.*, peticio AS peticionota, nota, data as datanota, empleat FROM
PETICIO JOIN NOTA_PETICIO ON referencia=peticionota;
```

- Obtenció de les dades de tots els empleats que han participat en la petició 5:

```
NP:=NOTA_PETICIO[peticio=5]
RA:=EMPLEAT[nif=empleat]NP
R:=RA[nom,nif]
SELECT E.nom, E.nif FROM EMPLEAT E, NOTA_PETICIO N WHERE E.nif=NOTA_PETICIO.empleat AND
NOTA_PETICIO.peticio=5;
```

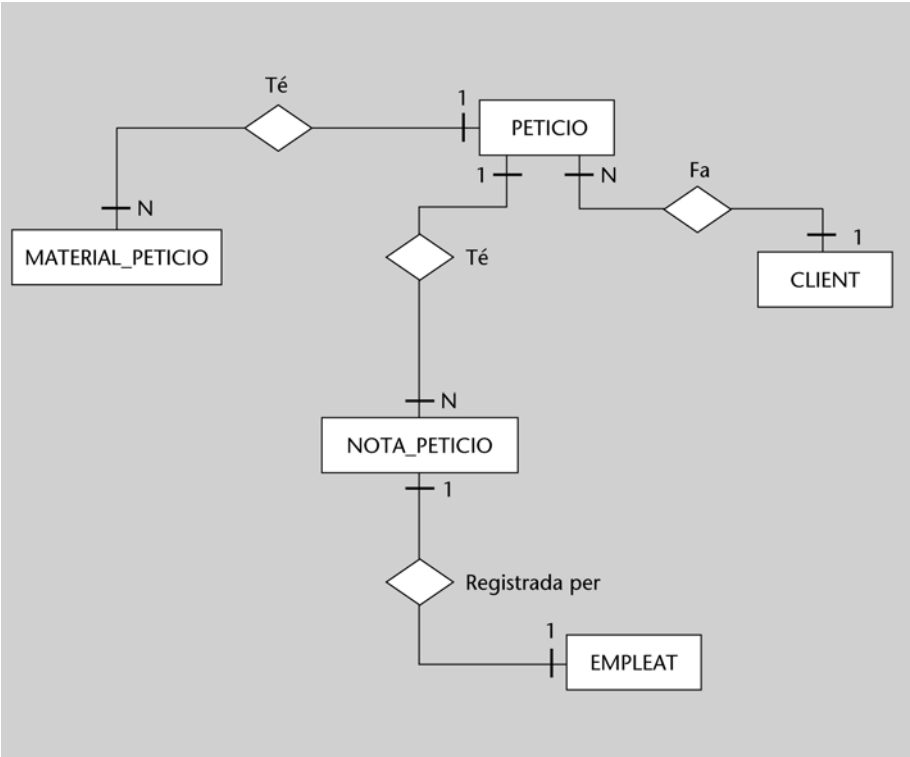
4. Introducció al disseny de bases de dades

Encara que les sentències SQL de creació de taules són bastant clares per a un usuari tècnic, amb vista a la reunió prèvia a la presa de decisió sobre l'SGBD concret en el qual implantarem la solució, necessitarem alguna cosa més.

Tenint en compte que entre els assistents a la reunió no hi ha més tècnics especialitzats en bases de dades que nosaltres, hem pensat que disposar un model entitat-relació del sistema ens ajudarà a comunicar millor l'estructura que estem plantejant i, de passada, a demostrar (o, si és necessari, corregir) que el model relacional que hem plantejat al començament és el correcte.

4.1. Disseny conceptual: el model ER

En primer lloc plantejarem el model obtingut i, després, comentarem els aspectes més interessants:



El model, expressat d'aquesta manera, és molt més comprensible per part de personal no tècnic o no especialitzat en tecnologies de bases de dades.

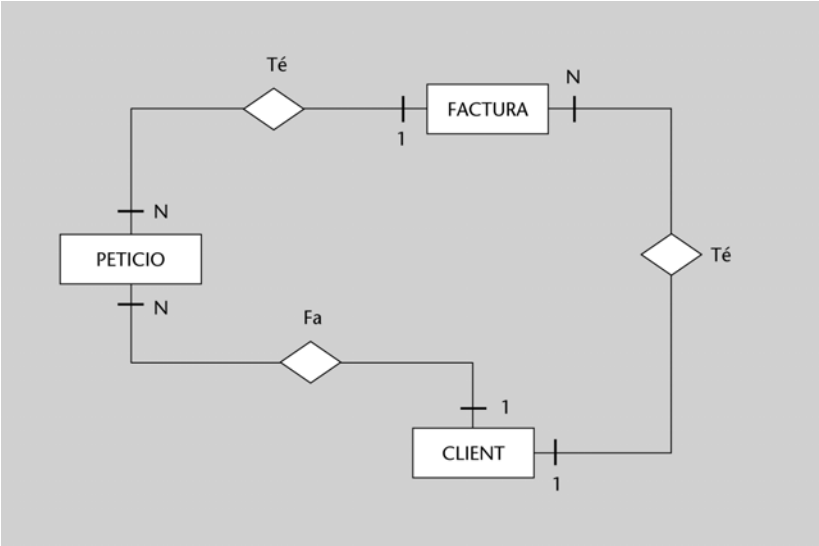
A partir de l'expressió gràfica del model, identifiquem limitacions o punts de millora, que anotem a continuació:

- Probablement, hem d'incloure informació de facturació a clients per les peticions realitzades.

- Probablement, hi haurà peticions que es puguin agrupar en una entitat superior (un projecte o treball), o bé peticions relacionades les unes amb les altres (peticions que s'hagin de resoldre abans que unes altres).

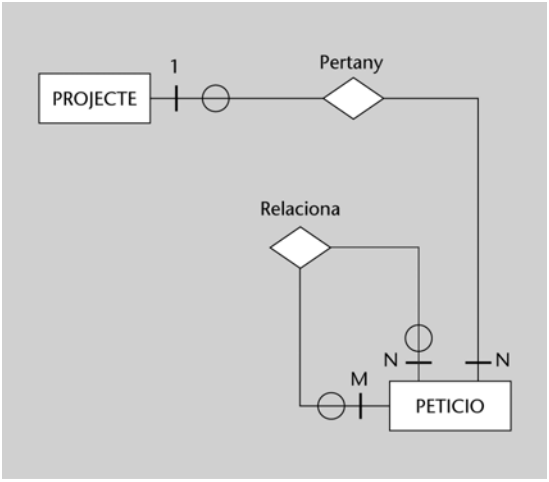
Una vegada identificades aquestes limitacions, ampliarem el model per corregir-les i impressionar els nostres superiors a la reunió.

- Informació de facturació a clients:



L'única cosa que hem introduït ha estat l'entitat `FACTURA`, que es relaciona amb *N* peticions i amb un únic client. Un `CLIENT` pot tenir diverses factures associades, però una petició només pot pertànyer a una única factura.

- Grups de peticions i relacions entre elles.



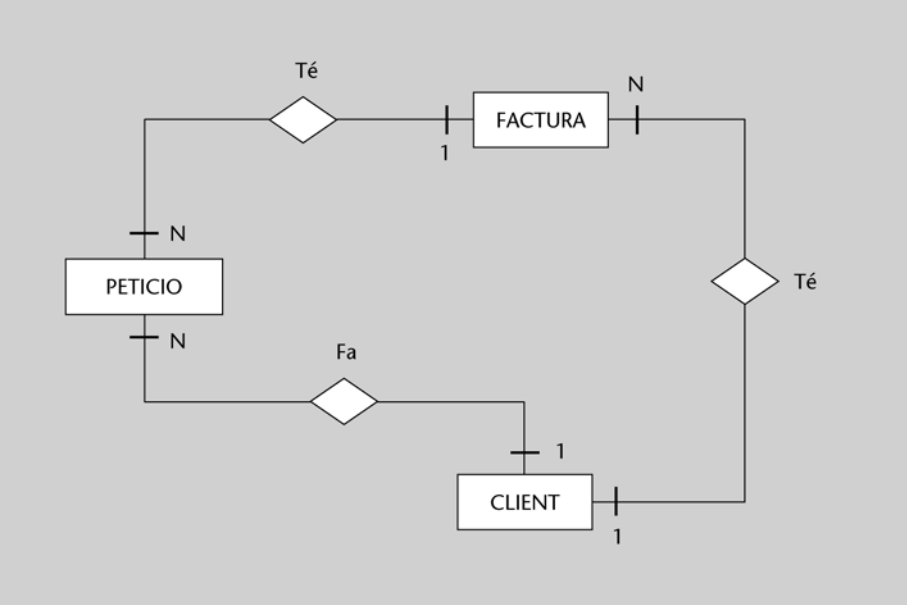
La pertinença a un projecte serà opcional, i així ho indiquem en el diagrama. Pel que fa a les relacions de peticions entre elles, es tracta d'una interrelació recursiva. Si volem tenir en compte casos com els següents, hem d'expressar la relació com a M:N recursiva i opcional. En la interrelació `RELACIONA`, hem de considerar algun atribut que indiqui de quin tipus de relació es tracta en cada cas:

- Una petició depèn d'una o més peticions.
- Una petició bloqueja una o més peticions.
- Una petició és la duplicada d'una o més peticions.
- Una petició està relacionada amb una o més peticions.

4.2. Disseny lògic: la transformació del model ER al model relacional

En l'apartat anterior hem suggerit unes ampliacions sobre el model ER que proporcionaven més prestacions al projecte. A continuació, realitzarem la transformació al model relacional d'aquestes ampliacions:

- Informació de facturació a clients.



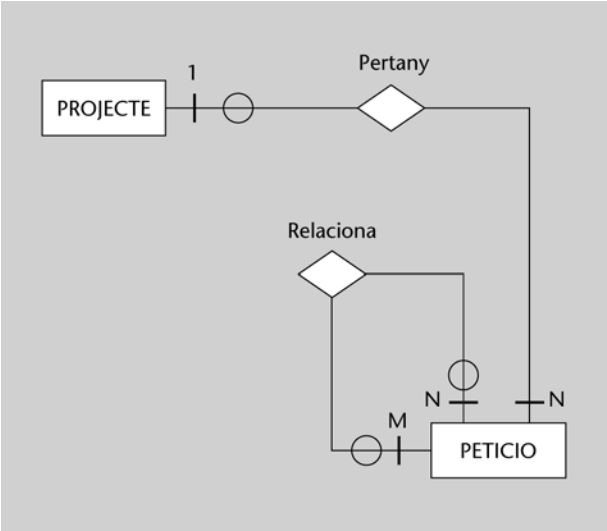
Segons les transformacions vistes en el mòdul "El llenguatge SQL", l'entitat `FACTURA` es transforma en la relació `FACTURA`, amb els atributs següents:

```
FACTURA(numfactura, data, client)
```

On `client` és una clau forana que correspon a la interrelació `TÉ` entre `CLIENT` i `FACTURA`. Un client pot tenir *N* factures, però una factura pertany només a un únic client.

La interrelació entre FACTURA i PETICIO del tipus 1:N es transforma també en una nova clau forana, que apareix sempre en el costat *N* de la interrelació; és a dir, en la relació PETICIO. Si hi ha peticions que no s'hagin de facturar (perquè s'han tancat sense resoldre, o eren duplicades de d'altres, etc.), la seva clau forana prendria el valor NUL.

- Grups de peticions i relacions entre elles.



D'una banda, l'entitat projecte s'ha de transformar en la relació PROJECTE, amb atributs com els següents:

```
PROJECTE(codi, nom, datainici, datafi)
```

La relació 1:N entre PROJECTE i PETICIÓ es transformarà en la inserció d'una nova clau forana en la relació PETICIO, que podrà tenir valor NUL si la petició no pertany a cap projecte; és a dir, si es tracta d'una petició aïllada. La relació PETICIO quedaria així:

```
PETICIO(referencia, client, resum, estat, datarecepcio, datainici, datafi, tempsempleat,
factura, projecte)
```

Pel que fa a les relacions entre peticions, es tracta d'una interrelació recursiva N:M i, per tant, es transformarà en una nova relació, PETICIO_RELACIO:

```
PETICIO_RELACIO(referencia_peticio1, referencia_peticio2, tipusrelacio)
```

En aquest cas, i segons el valor que pugui prendre l'atribut `{tipusrelacio}`, tindrà importància o no quina referència de petició apareix en cada atribut de la relació.

Exemple

Si la relació ha d'indicar simplement que dues peticions estan relacionades, llavors no importa quina referència sigui, l'1 o la 2.

En canvi, si l'atribut `{tipusrelacio}` indica un bloqueig o una dependència entre relacions (perquè una s'ha de resoldre abans que l'altra, per exemple), llavors sí que té sentit quina referència de petició s'emmagatzema en l'atribut 1 i quina en el 2. En tot cas, aquesta tasca correspondrà a la solució que s'adapti i treballi amb la base de dades en darrer terme, no al mateix model.

5. Bases de dades en MySQL

Una vegada hem acabat el procés de disseny de la nostra solució, quant al seu sistema d'informació, és hora d'implantar-lo sobre un sistema gestor de bases de dades.

Atès que disposem de dues alternatives (MySQL i PostgreSQL), i no ens correspon prendre la decisió final (només fer la recomanació), elaborarem una llista amb els aspectes clau en la presa de decisions i puntuarem, o comentarem, cada SGBD segons els ítems següents:

- Model de llicència, preu.
- Suport per part del fabricant.
- Connexió des de PHP.
- Prestacions en creació de les estructures (taules, índexs, etc.).
- Prestacions en tipus de dades.
- Prestacions en consultes simples.
- Prestacions en consultes complexes.
- Prestacions en manipulació de dades.
- Facilitat en l'administració d'usuaris.
- Facilitat en la gestió de còpies de seguretat.

Nota

Mostrarem aquesta llista en forma de taula, i al final elaborarem unes conclusions. Encara que no fem una ponderació de cada aspecte de la llista anterior, la simple comparació ens servirà per a arribar a una conclusió ràpida.

Concepte	Valoració	Comentaris
Model de llicència, preu	2	Encara que no ens plantegem vendre la nostra solució, ni comercialitzar-la amb una llicència de propietat, la llicència dual de MySQL sempre serà un aspecte que haurem de tenir en compte si algú s'interessa per la nostra aplicació.
Suport per part del fabricant	3	Tenim tant l'opció de contractar suport en diverses modalitats, com la d'optar per consultar l'amplíssima gamma d'usuaris del producte. En tot cas, en ambdues situacions obtindrem un suport excel·lent.
Connexió des de PHP	3	PHP sempre ha inclòs suport per a aquest SGBD bé amb funcions especials dedicades que aprofiten al màxim les seves característiques, o bé amb llibreries PEAR com DB que ens abstreuen de l'SGBD i que suporten MySQL a la perfecció. Encara que recentment hi ha hagut algun problema amb la llicència i semblava que PHP no inclouria suport per a MySQL en les seves últimes versions, MySQL ha fet una excepció amb PHP (que sens dubte ha contribuït molt a la popularització de MySQL) pel bé de la comunitat i dels seus usuaris.
Prestacions en creació de les estructures (taules, índexs, etc.)	2	MySQL és francament fàcil de manejar en aquest aspecte, i encara que no ofereix totes les prestacions previstes en l'estàndard, és "satisfactori" per a l'aplicació que estem planejant.
Prestacions en tipus de dades	2	Els tipus de dades suportades per MySQL i també els operadors inclosos en l'SGBD són més que suficients per a la nostra aplicació.
Prestacions en consultes simples	3	Aquesta és precisament la característica que fa que MySQL sigui un dels SGBD més ben posicionats.

Clau de valoració

- 1: no satisfactori
- 2: satisfactori
- 3: molt satisfactori

Concepto	Valoración	Comentarios
Prestacions en consultes complexes	2	Fins fa poc, MySQL no suportava subconsultes i això implicava un esforç més gran per part dels programadors. Ara ja les suporta i amb el mateix nivell que el dels seus competidors.
Prestacions en manipulació de dades	3	MySQL inclou multitud d'opcions no estàndard per a carregar dades externes, inserir o actualitzar sobre la base de consultes complexes i la utilització d'operadors com a condicions per a la manipulació.
Facilitat en l'administració d'usuaris	3	Suporta molt bé l'estàndard quant a la creació d'usuaris i la gestió dels seus privilegis amb GRANT i REVOKE. A més, totes aquestes dades són accessibles en taules de sistema, la qual cosa fa molt senzilla la verificació de permisos.
Facilitat en la gestió de còpies de seguretat	3	Disposem tant d'eines de bolcatge, com la possibilitat de còpia binària de la base de dades. A més, donada la seva popularitat, diversos fabricants de solucions de còpies de seguretat proporcionen connectors per realitzar *backups* de la base de dades de calent en calent.
Conclusió	2,6	Som davant d'un "més que satisfactori" SGBD per a la solució que ens plantegem. No hi ha cap carència insalvable.

6. Bases de dades en PostgreSQL

Concepte	Valoració	Comentaris
Model de llicència, preu	3	La llicència BSD no ens limita en cap aspecte. Simplement haurem d'incloure la nota sobre aquesta en el nostre programari, tant si el volem comercialitzar com si no.
Suport per part del fabricant	2	PostgreSQL no ofereix suport directament, encara que sí que proporciona els mecanismes perquè la comunitat ho ofereixi (llistes de correu, IRC, enllaços, etc.). També té una llista (curta) d'empreses que ofereixen suport professional de PostgreSQL. A Espanya només n'hi ha una, i no és una empresa de desenvolupament de programari.
Connexió des de PHP	3	PHP sempre ha inclòs suport per a aquest SGBD, bé amb funcions especials dedicades que aprofiten al màxim les seves característiques o bé amb llibreries PEAR com DB, que ens abstreuen de l'SGBD i que suporten PostgreSQL a la perfecció.
Prestacions en creació de les estructures (taules, índexs, etc.)	3	PostgreSQL és molt potent en aquest aspecte, ofereix pràcticament totes les prestacions previstes en l'estàndard, i té un fantàstic sistema d'extensió.
Prestacions en tipus de dades	3	Els tipus de dades suportades per PostgreSQL, i també els operadors inclosos en l'SGBD són més que suficients per a la nostra aplicació. A més, el seu sistema d'extensions i definició de tipus i dominis inclou gairebé tot el que puguem necessitar.
Prestacions en consultes simples	3	Sens dubte, PostgreSQL no decep en aquest punt.
Prestacions en consultes complexes	3	PostgreSQL ha suportat subconsultes, vistes i tot el que puguem necessitar en la nostra aplicació des de fa diversos anys. La seva implementació d'aquestes és ja molt estable.
Prestacions en manipulació de dades	3	PostgreSQL inclou multitud d'opcions no estàndard per a carregar dades externes, inserir-les o actualitzar-les sobre la base de consultes complexes i la utilització d'operadors com a condicions per a la manipulació.
Facilitat en l'administració d'usuaris	2	Suporta bastant bé l'estàndard quant a la creació d'usuaris i la gestió dels seus privilegis amb GRANT i REVOKE. El seu sistema múltiple d'autenticació el fa massa complex en aquest aspecte.
Facilitat en la gestió de còpies de seguretat	3	Disposem tant d'eines de bolcatge com de la possibilitat de còpia binària de la base de dades.
Conclusió	2,8	Som davant de l'SGBD gairebé ideal. Només li falta facilitar la gestió d'usuaris i millorar el seu suport.

Clau de valoració

- 1: no satisfactori
- 2: satisfactori
- 3: molt satisfactori

7. Desenvolupament d'aplicacions en connexió amb bases de dades

En la reunió mantinguda amb la direcció es van examinar amb molta cura les anàlisis dels SGBD seleccionats. En ser la diferència de valoració tan lleu, no va ser fàcil prendre una decisió, però al final es va decidir la implementació de la solució sobre l'SGBD PostgreSQL.

Es va decidir també fer la implementació en PHP, abstraient-nos de l'SGBD amb el qual treballéssim. Així, en cas que la major dificultat en l'administració de PostgreSQL ens fes rectificar la decisió en el futur, el temps de posada en marxa del canvi seria mínim.

Abans d'iniciar la implantació, realitzarem unes proves conceptuals de la mateixa implementació que després passarem a un equip de desenvolupament intern perquè faci la resta. En concret, prendrem algunes de les consultes vistes en el capítol 3 i programarem els *scripts* PHP de les pàgines corresponents, documentant-les al màxim per facilitar la feina a l'equip de desenvolupament.

En primer lloc, crearem un fitxer .php amb la connexió a la base de dades per a incloure'l en tots els PHP que el necessitin i evitar, així, haver de repetir codi cada vegada. Aquesta acció també ajudarà a mantenir centralitzades les dades de la connexió i, en cas que hàgim de canviar l'usuari o la contrasenya o qualsevol altra dada de la connexió, només hauríem d'actualitzar aquest fitxer.

dadesconnexio.php

```
<?php
// Incloem la llibreria una vegada instal·lada mitjançant PEAR
require_once 'DB.php';


// Creem la connexió a la base de dades, en aquest cas PostgreSQL
$db =& DB::connect('pgsql://usuari:password@servidor/basededades');


// Comprovem error en la connexió
if (DB::isError($db)) {
   die($db->getMessage());
}
?>
```

a. Nou client. Pàgina de resultat de la inserció d'un nou client

```
<?php
// Incloem el fitxer amb les dades de la connexió.
include_one 'dadesconnexio.php';

// Utilitzem el mètode quoteSmart() per a evitar que determinats caràcters
// (intencionats o no) puguin trencar la sintaxi de la sentència SQL.
// El mètode inserirà automàticament cometes al voltant de les cadenes
// de text, o tractarà els valors NULL correctament segons l'SGBD, etc.
$db->query("INSERT INTO CLIENT VALUES ("
   . $db->quoteSmart($_REQUEST['nom']) . ","
   . $db->quoteSmart($_REQUEST['dni']). ","
   . $db->quoteSmart($_REQUEST['telefon']) . ","
   . $db->quoteSmart($_REQUEST['email']) . ")");

if (DB::isError($db)) {
   echo "<h2>Error en inserir el client</h2>";
   die($db->getMessage());
}

$db->disconnect();
```

c. Canvi d'estat de la petició:

```
<?php
include_once 'dadesconnexio.php';

// Treballarem amb totes les operacions d'aquesta pàgina en forma de
// transacció, ja que, si es produeix un error en inserir una nota o un
// material, la petició pot quedar en un estat erroni.
$db->autoCommit(false);

// Diem a PHP que emmagatzemi en una memòria intermèdia la sortida, per a poder així
// rectificar en cas que es produeixi un error.
ob_start();

// Suposem que els estats ens arriben directament amb els valors
// suportats pel domini, per exemple, a partir dels valors
// fixos d'un desplegable.

// Suposem que la data d'inici i data de final ens arriben en format
// espanyol dd/mm/yyyy i els convertim a YYYY-mm-dd segons l'ISO 8601.

// Suposem que el temps emprat ens arriba en dos camps, hores i minuts, de
// manera que concatenant-los i inserint un ":" al mig, obtenim una
// hora en format ISO 8601.
```

```
if (isset($_REQUEST['datainici']) && !empty($_REQUEST['datainici'])) {
   $datainici_array=split("/",$_REQUEST['datainici']);
   $datainiciDB=date("Y-M-d", mktime(0, 0, 0, $datainici[1], $datainici[0],
$datainici[2]));
} else {
   $datainiciDB=NULL;
}

if (isset($_REQUEST['datafi']) && !empty($_REQUEST['datafi'])) {
   $datafi_array=split("/",$_REQUEST['datafi']);
   $datafiDB=date("Y-M-d", mktime(0, 0, 0, $datafi[1], $datafi[0], $datafi[2]));
} else {
   $datafiDB=NULL;
}

$db->query("UPDATE PETICIO SET "
   . "client=" . $db->quoteSmart($_REQUEST['client']) . ","
   . "resum=" . $db->quoteSmart($_REQUEST['resum']) . ","
   . "estat=" . $db->quoteSmart($_REQUEST['estat']) . ","
   . "datainici=" . $db->quoteSmart($datainiciDB) . ","
   . "datafi" . $db->quoteSmart($datafiDB) . ","
   . "tempsempleat=". $db->quoteSmart($_REQUEST['hora'] . ":" . $_REQUEST['minuts']));

if (DB::isError($db)) {
   echo "<h2>Error en inserir el client</h2>";
   ob_flush();
   die($db->getMessage());
} else {
   echo "<h2>Petició actualitzada correctament</h2>";
}

// Comprovem si han afegit alguna nota
if (isset($_REQUEST['text_nota']) && !empty($_REQUEST['text_nota'])) {
   // Tenim l'identificador de petició en $_REQUEST['referencia']
   $db->query("INSERT INTO NOTA_PETICIO VALUES ("
   . $db->quoteSmart($_REQUEST['referencia']) . ","
   . $db->quoteSmart($_REQUEST['text_nota']) . ","
   . $db->quoteSmart(date("Y-M-d",mktime())) . ","
   . $db->quoteSmart($_REQUEST['nifEmpleat']) . ")");
   if (DB::isError($db)) {
      ob_clean();
      echo "<h2>Error en inserir la nota. Dades de la petició no actualitzades</h2>";
      ob_flush();
      $db->rollback();
      die($db->getMessage());
   } else {
      echo "<h3>Nota actualitzada correctament</h2>";
   }
}
```

```
// Comprovem si han afegit cap material+
if (isset($_REQUEST['nommaterial']) && !empty($_REQUEST['nommaterial'])) {
   // Tenim l'identificador de la petició en $_REQUEST['referencia']
   $db->query("INSERT INTO MATERIAL_PETICIO VALUES "
   . $db->quoteSmart($_REQUEST['nommaterial']) . ","
   . $db->quoteSmart($_REQUEST['referencia']) . ","
   . $db->quoteSmart($_REQUEST['preu']) . ","
   . $db->quoteSmart($_REQUEST['quantitat']) . ")");
   if (DB::isError($db)) {
      ob_clean();
      echo "<h2>Error en inserir el material. Dades de la petició no actualitzades</h2>";
      ob_flush();
      $db->rollback();
      die($db->getMessage());
   } else {
      echo "<h3>Material actualitzat correctament</h2>";
   }
}

$db->commit();
$ob_flush();

$db->disconnect();
?>
```

f. y g. Peticions obertes d'un client i resum dels materials usats en cada una

```
<?php
include_once 'dadesconnexio.php';

// Busquem les peticions obertes d'un client
$res=$db->query("SELECT P.referencia, P.resum, P.estat, P.datarecepcio FROM
PETICIO P JOIN CLIENT C ON P.client=C.nif WHERE ESTAT NOT IN ('Resolta',
'Tancada') ORDER BY datarecepcio");

if (DB::isError($db)) {
   die($db->getMessage());
}

// Abans de començar la iteració per les peticions, prepararem la consulta
// corresponent als materials emprats en cada una.
$queryMaterial=$db->prepare("SELECT SUM(M.preu) as preuMaterials,
COUNT(M.nommaterial) numMaterials FROM MATERIAL_PETICIO M WHERE M.peticio=? ");

if (DB::isError($db)) {
   die($db->getMessage());
}
```

```
echo "<table>";
echo "<tr><th>Referencia</th><th>Resum</th><th>Estat</th><th>Durada</th><th>
Datarecepció</th><th>Preumaterials</th><th>Númmaterials</th></tr>";
while($res->fetchInto($row,DB_FETCHMODE_ASSOC)) {
   echo "<tr>";
   echo "<td>" . $row['referencia'] . "</td>";
   echo "<td>" . $row['resum'] . "</td>";
   echo "<td>" . $row['estat'] . "</td>";
   echo "<td>" . $row['datarecepcio'] . "</td>";
   $resMaterial=$db->execute($queryMaterial,$row['referencia']);
   if (DB::isError($db)) {
      die($db->getMessage());
   } else {
      $res->fetchInto($rowMaterial,DB_FETCHMODE_ASSOC);
      echo "<td>" . $rowMaterial['preuMaterials'] . "</td>";
      echo "<td>" . $rowMaterial['numMaterials'] . "</td>";
   }
   echo "</tr>";
}

echo "</table>";
?>
```

Mitjançant aquests tres exemples, disposem ja d'una base tant de codi, com d'estil i mecanismes de comprovació d'error, per a desenvolupar la resta de l'aplicació, sense tenir en compte el seu disseny.

Hem intentat escollir consultes i operacions representatives del funcionament de l'aplicació i, alhora, que es corresponguessin amb les que hem vist en apartats anteriors. A més, hem introduït algunes funcions PHP que se solen utilitzar en combinació amb el treball en bases de dades per a tipus concrets, i per al tractament d'errors, per a evitar que l'usuari rebi informació confusa en la pàgina de resultats.

Resum

En aquesta unitat hem vist les activitats més destacades de les fases inicials d'un projecte de desenvolupament amb connexió a bases de dades.

Més que resoldre el cas, es tractava d'identificar els aspectes clau dels casos reals i enllaçar-los amb el contingut de la resta d'unitats del curs.

Haureu pogut identificar quines activitats són més rellevants per al resultat final del projecte, en quines convé invertir més temps i quines no són tan crítiques per als objectius d'aquest cas o d'un altre de similar.

Si heu seguit la planificació suggerida i repassat cada unitat a la qual es feia referència, haureu pogut comprendre l'aplicació pràctica del material i s'hauran assolit els objectius que ens proposàvem en redactar aquesta unitat didàctica.

També és possible llegir aquest cas d'estudi com un capítol final del curs, on es desenvolupa un exemple complet. S'ha intentat redactar amb aquesta doble comesa.