Memòria TFC Bases de Dades

Disseny i implementació d'un sistema de control de projectes

Autor: David Franco Sánchez

Consultor: Alexandre Cornet

Tardor 2004

1 Introducció

1.1 Resum del treball

L'àmbit d'aquest treball final de carrera és el camp del disseny de les bases de dades relacionals, per la qual cosa aquest treball s'ha enfocat des de diversos punts de vista:

- Acadèmic: Es descriurà la teoria del disseny de bases de dades relacionals
- Pràctic: Es posarà en pràctica tot el model teòric desenvolupant un cas concret (producte) que s'anirà intercalant amb el desenvolupament teòric del punt anterior.
- Tecnològic: El desplegament del producte es realitzarà al SGBD Oracle 9i (a la qual es realitza una introducció), on s'implementarà tota la interfície a la BD amb la tecnologia PL/SQL d'Oracle.

Per desenvolupar els punts acadèmics i pràctics he fet servir els materials docents i coneixements aportats a les assignatures de Bases de dades I i Bases de dades II; assignatures que he realitzat dins dels meus estudis d'Enginyeria Tècnica en Informàtica de Gestió a la Universitat Oberta de Catalunya. A més a més d'algunes altres fonts d'informació que detallo a l'apartat de Bibliografia.

El punt tecnològic ha estat diferent dels dos anteriors, ja que no tenia cap experiència prèvia ni amb el SGBD Oracle (les meves pràctiques s'havien desenvolupat sobre Informix), ni amb la tecnologia de procediments emmagatzemats (PL/SQL). Per la qual cosa aquest punt ha estat un treball de recerca d'informació i d'aprenentatge d'una nova tecnologia.

1.2 Especificació del producte

Dissenyaré i implementaré un sistema de control de projectes per a una organització tal que la seva estructura interna i funcionament obeeixin als següents paràmetres:

- La organització està implantada a diferents ciutats.
- En cada centre de treball de la organització hi ha n departaments.
- Cada departament té una sèrie de empleats que no poden estar en més d'un departament.
- Cada empleat te una funció tipificada, cap de departament, cap de projecte, dibuixant, enginyer, economista, ….
- Un empleat pot canviar de departament, de centre de treball i de funció, quan canviï alguna d'aquestes característiques, no s'ha de modificar les dades del treballador sinó que es donarà d'alta un altre cop i "el treballador antic" es "desactivarà", d'aquesta manera no barrejaran dades en l'hipotètic canvi de situació.
- Dins de cada departament hi ha caps de projecte els quals tindran assignats un o més projectes (o tasques).
- Cada projecte només pot tenir assignat un cap de projecte.
- Cada projecte tindrà n treballadors del departament al qual pertany el cap de projecte.
- Cada treballador podrà estar assignat a n projectes.
- Cada treballador pot entrar i sortir del centre de treball n vegades i aquestes quedaran enregistrades.
- A cada projecte s'hi poden assignar despeses de qualsevol tipus que es tingui, i cada assignació de despeses que es faci estarà relacionada amb una persona.
- Cada treballador tindrà assignat un cost per hora
- Cada projecte s'ha pressupostat i ha estat acceptat per el client, per tant abans de començar ja se sap que es cobrarà. Aquesta es una dada important i que el sistema ha de guardar al obrir el projecte

- Per controlar els tipus de despeses s'haurà de muntar un sistema de taules tals que serà fàcil tipificar les despeses.
- La direcció de la organització donarà permís a cada tipus de treballador sobre cada tipus de despesa que hi ha, de manera que si no te permís sobre un tipus no es podrà carregar despeses.
- Un cop un projecte s'ha finalitzat ja no es podran fer operacions sobre ell, és a dir assignar-hi un treballador o despeses o el que sigui.

El sistema ha de recollir informació sobre quant temps el treballador està treballant i en què, i quines han estat les seves despeses mentre ha estat assignat a cada projecte o tasca en el que ha treballat, d'aquesta manera, a part de tenir un control horari del treballador, la organització sabrà perfectament si la realització d'un projecte ha estat rendible i disposarà de dades en mig del desenvolupament de costos i hores per a fer actuacions correctives.

Dissenyaré la base de dades (Diagrama E/R), fent l'script de creació de taules, índexs, etc.;i implementaré els procediments emmagatzemats que es requereixin per tal de disposar d'una bona interfície, entre d'altres tindrà:

- Alta de projecte.
- Tancament del projecte.
- Alta, Baixa, Modificació Centre, Departament, Despesa, tipus de despesa ….
- Assignació / desassignació de treballadors i despeses al projecte.
- Treballador entra a treballar.
- Treballador surt de treballar.
- Alta d'un treballador.
- Baixa d'un treballador.
- Modificació de les dades del treballador.
- Canvi de centre/departament/tipus del treballador
- Consulta treballador, departament …

Implementaré un mecanisme d'inicialització de la base de dades tal que simuli un any de funcionament, tenint en compte que el numero de centres de la organització es de entre 5 i 10, que cada centre té un nombre de treballadors d'entre 25 i 150, un nombre de departaments d'entre 2 i 4, i cada departament té un nombre de membres d'entre 5 i 40. Aquest mecanisme funcionarà fent crides als procediments implementats, serà aleatori i s'ajustarà al que la realitat i el sentit comú marca, (és a dir no serà una inicialització cíclica de fàcil detecció).

1.3 Objectius

L'objectiu general que vull assolir amb aquest TFC és un coneixement profund de disseny de bases de dades. Vull aplicar els coneixements que he assolit durant el transcurs de la carrera, per realitzar un treball professional que em permeti estar preparat pel desenvolupament d'aplicacions empresarials al món laboral.

Concretament vull posar en pràctica els coneixements adquirits a les assignatures de Bases de Dades I i Bases de Dades II, i aprofundir ens els següents àmbits tècnics:

- SQL
- PL/SQL
- Disseny E/R
- Oracle 9i

2 Disseny de Bases de Dades

2.1 Introducció al disseny de bases de dades

El disseny d'una base de dades consisteix a definir l'estructura de les dades que ha de tenir la base de dades d'un sistema d'informació determinat. En el nostre cas (cas relacional), aquesta estructura serà un conjunt d'esquemes de relació amb els seus atributs, dominis d'atributs, claus primàries, claus foranes, etc.

El disseny d'una base de dades no és pas un procés senzill. Habitualment, la complexitat de la informació i la quantitat de requisits dels sistemes d'informació fan que sigui complicat. Per aquest motiu, quan es dissenyen bases de dades, és interessant aplicar la vella estratègia de dividir per a vèncer.

Per tant, convé descompondre el procés del disseny en diverses etapes, en cadascuna de les quals s'obté un resultat intermedi que serveix de punt de partida de l'etapa següent, i a la darrera etapa s'obté el resultat desitjat. D'aquesta manera no cal resoldre de cop tota la problemàtica que planteja el disseny, sinó que a cada etapa s'afronta un sol tipus de subproblema. D'aquesta manera es divideix el problema i alhora se simplifica el procés.

Descompondré el disseny de bases de dades en tres etapes:

1. **Etapa del disseny conceptual:** en aquesta etapa s'obté una estructura de la informació de la futura BD independent de la tecnologia que cal emprar. No es té en compte encara quin tipus de base de dades s'utilitzarà (relacional, orientada a objectes, jeràrquica, etc.)en conseqüència, tampoc no es té en compte amb quin Sistema Gestor de Bases de Dades (SGBD) ni amb quin llenguatge concret s'implementarà la base de dades. Així, doncs, l'etapa del disseny conceptual ens permet concentrar-nos únicament en la problemàtica de l'estructuració de la informació, sense haver-nos de preocupar alhora de resoldre qüestions tecnològiques. El resultat de l'etapa del disseny conceptual s'expressa mitjançant algun model de dades d'alt nivell. Un dels més emprats és el **model entitat-interrelació** (*entity-relantionship*), que abreviarem amb la sigla **ER**.

2. **Etapa del disseny lògic:** en aquesta etapa es parteix del resultat del disseny conceptual, el qual es transforma de manera que s'adapti a la tecnologia que s'ha d'emprar. Més concretament, cal que s'ajusti al model de l'SGBD amb el qual es desitja implementar la base de dades. Per exemple, si es tracta d'un SGBD relacional, aquesta etapa obtindrà un conjunt de relacions amb els seus atributs, claus primàries i claus foranes. Aquesta etapa parteix del fet que ja s'ha resolt la problemàtica de l'estructuració de la informació a un nivell conceptual i ens permet concentrar-nos en les qüestions tecnològiques relaciones amb el model de la base de dades. Mes endavant explicarem com es fa el disseny lògic d'una base de dades relacional prenent com a punt de partida un disseny conceptual expressat amb el model ER, és a dir, veurem com es pot transformar un model ER en un model relacional.

3. **Etapa del disseny físic:** en aquesta etapa es transforma l'estructura obtinguda a l'etapa del disseny lògic amb l'objectiu d'aconseguir una major eficiència i, a més, es completa amb aspectes d'implementació física que dependran de l'SGBD. Per exemple, si es tracta d'una base de dades relacional, la transformació de l'estructura pot consistir en els fets següents: tenir emmagatzemada alguna relació que sigui la combinació de diverses relacions que s'han obtingut a l'etapa del disseny lògic, partir una relació en diverses relacions, afegir algun atribut calculable a una relació, etc. Els aspectes d'implementació física que cal completar consisteixen normalment en l'elecció d'estructures físiques d'implementació de res relacions, la selecció de la mida de les memòries intermèdies (*buffers*) o de les pàgines, etc. A l'etapa del disseny físic (amb l'objectiu d'aconseguir un bon rendiment de la base de dades), s'han de tenir en compte les característiques dels processos que consulten i actualitzen la base de dades com ara els camins d'accés que utilitzen i les freqüències d'execució. També cal considerar els volums que s'espera tenir de les diferents dades que es volen emmagatzemar.

2.2 Disseny conceptual

2.2.1 Introducció al model ER

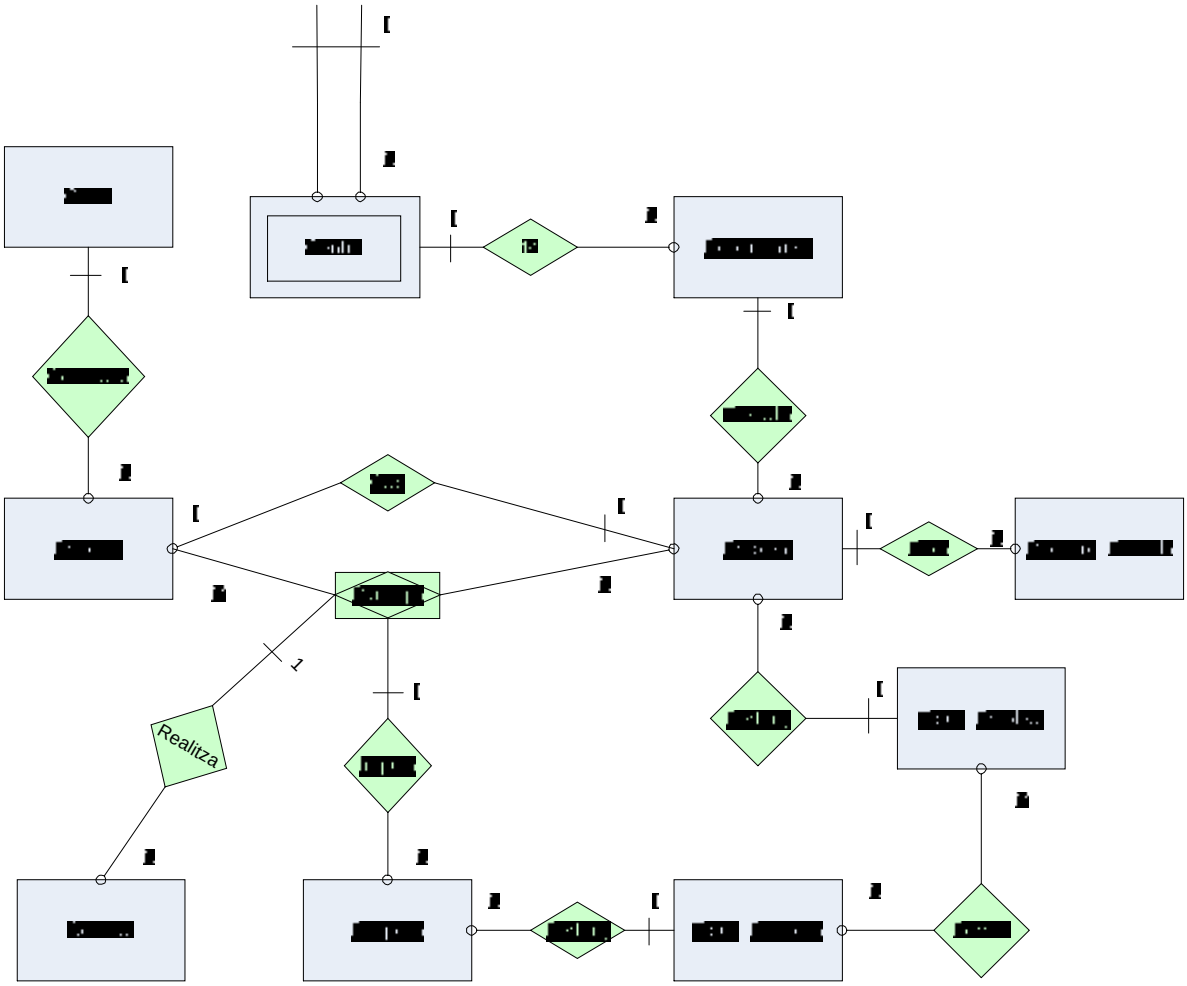
El **model ER** és un dels enfocaments de modelització de dades que més es fa servir actualment per la seva simplicitat i llegibilitat. La seva llegibilitat es veu afavorida perquè proporciona una notació diagramàtica molt entenedora. És una eina útil tant per a ajudar al dissenyador a reflectir en un model conceptual els requisits del món real d'interès, com per a comunicar-se amb l'usuari final sobre el model conceptual obtingut i, així, poder verificar si satisfà els seus requisits.

El nom complet del model ER és entity-relationship, i prové del fet que els principals elements que inclou són les entitats i les interrelacions (entities i relation-ships). Traduirem aquest nom per 'entitat-interrelació'. Alguns autors anomenen entitat-relació el model ER, però en el nostre cas hem preferit traduir relationship per 'interrelació' i no per 'relació', a fi d'evitar confusions entre aquest concepte i el de relació que s'empra en el model relacional.

L'origen del model ER es troba en treballs efectuats per Peter Chen el 1976. Posteriorment, molts altres autors han descrit variants i/o extensions d'aquest model. Així, a la literatura es troben moltes formes diferents del model ER que poden variar simplement en la notació diagramàtica o en alguns dels conceptes en què es basen per a modelitzar les dades.

Quan es vol fer servir el model ER per a comunicar-se amb l'usuari, és recomanable emprar una variant del model que inclogui només els seus elements més simples –entitats, atributs i interrelacions- i, potser, algunes construccions addicionals, com ara entitats dèbils i dependències d'existència. Aquests eren els elements inclosos en el model original proposat per Chen. En canvi, per a portar a terme la tasca de modelitzar pròpiament dita, sol ser útil fer servir un model ER més complert que inclogui construccions més avançades que estenen el model original.

Segons la noció general de *model de dades,* un model de dades té en compte tres aspectes de les dades: l'estructura, la manipulació i la integritat. El model ER, però, habitualment es fa servir per a reflectir aspectes de l'estructura de les dades i de la seva integritat, però no de la seva manipulació.



Atributs d'entitats:

- CIUTAT: codiCiutat, nomCiutat, paisCiutat, situacioCiutat
- CENTRE: codiCentre, nomCentre, adreçaCentre, situacioCentre
- DEPARTAMENT: codiDepartament, nomDepartament, situacioDepartament
- EMPLEAT: codiEmpleat, nomEmpleat, cognomsEmpleat, dniEmpleat, costEmpleat, situacioEmpleat
- TIPUS_EMPLEAT: codiTipusEmpleat, funcioEmpleat, situacioTipusEmpleat
- REGISTRE_ENTRADA: codiRegistre, dataRegistre, tipusRegistre, situacioRegistreEntrada
- PROJECTE: codiProjecte, nomProjecte, dataOberturaProjecte, pressupostProjecte, situacioProjecte
- CLIENT: codiClient, nomClient, adreçaClient, situacioClient
- DESPESA: codiDespesa, importDespesa, situacioDespesa
- TIPUS_DESPESA: codiTipusDespesa, nomCodiDespesa, situacioTipusDespesa
- ACTIVITAT: codiActivitat, dataIniciActivitat, dataFiActivitat, situacioActivitat

2.2.3 Justificació del disseny conceptual presentat

Com a regla general he intentat ser el més obert possible per no restringir les possibilitats del sistema d'informació, exceptuant les restriccions pròpies del model relacional de duplicitat d'informació i els principis d'economia de l'espai, claredat i reutilització.

S'ha decidit com a regla general assignar a cadascuna de les entitats una clau que no tingui valor propi a nivell conceptual. És a dir, s'assignarà un codi sense cap informació (únicament haurà de ser únic dins de cada entitat), d'aquesta manera tenim més flexibilitat a l'hora de permetre o no certs valors que poden o no poden estar repetits a nivell conceptual, però que no són una restricció física provocat pel disseny. D'aquesta manera es pot satisfer el requeriment de donar de baixa lògica un empleat modificant el valor del camp situació. A la fase de disseny físic incorporarem les restriccions de no duplicitat (exemple: dos empleats actius no poden tenir el mateix DNI) juntament amb les d'indexació per un bon accés.

Comentaris específics

- He separat l'entitat ciutat de l'entitat centre degut a que dins dels sistemes d'informació empresarials solen haver-hi bases de dades "comunes" i normalment un fitxer de poblacions o ciutats sol ja existir prèviament. A més a més, d'aquesta manera el manteniment d'aquest fitxer es pot fer des de fora del sistema utilitzant algun servei extern d'actualització periòdica. No he volgut afegir molts camps, únicament el nom i el país de la ciutat. No vaig considerar que calgués separar l'entitat país ja que a l'enunciat només es fa menció de les ciutats.
- Un centre l'he dissenyat com a entitat dèbil per poder fer èmfasi a l'estreta relació que hi ha entre una ciutat i els centres de treball que disposa. D'aquesta manera serà més fàcil identificar els diferents centres que hi ha a cadascuna de les ciutats. He permès que puguin haver-hi centres que no tenen departaments; per ser centres especials o fins i tot pel moment puntual en que s'està definint un centre de treball que es vol tenir inventariat però que encara no se li ha assignat personal. Obligatòriament un departament pertanyerà a un i només un centre de treball.
- Pot existir un departament al qual no treballi cap empleat, però un empleat ha de treballar en un departament i només en un departament. Aquest empleat pot fitxar moltes vegades (o cap) per la qual cosa normalment tindrà molts registres d'entrada (data i hora) amb tipus entrada o sortida. Aquest empleat tindrà una i només una funció tipificada dins del seu departament, com que aquestes funcions tenen relació amb quins tipus de despeses poden imputar, he decidit separar els tipus d'empleats en una entitat diferent que es relaciona amb cada empleat en comptes d'afegir un atribut "funció" dins de l'entitat empleat. D'aquesta manera el sistema controlarà a través de les relacions entre entitats tipus_empleat i tipus_despesa quins empleats poden imputar cada tipus de despesa i quins no.
- Un empleat pot participar en un projecte o no, a més a més pot participar en més d'un projecte. Aquesta relació s'ha de transformar en associativa ja que la "participació" en un projecte és el que li dona dret a un empleat a realitzar activitats (dedicar hores de feina al projecte) i a imputar despeses al projecte. Com que aquestes activitats i aquestes despeses poden ser moltes o cap s'han de relacionar amb la participació, per la qual cosa cal fer-la associativa per transformar-la específicament en una entitat.
- De la mateixa manera que les ciutats, normalment la taula de clients sol ser compartida per diverses aplicacions. Així que he donat la possibilitat de que hi hagi clients que no estan relacionats amb cap projecte però que existeixen dins de l'entitat client.

2.2.4 Restriccions textuals identificades que no estan expressades al model ER

Les següents restriccions no estan limitades expressament pel diagrama ER proposat:

1. Els empleats que participen en un projecte han de ser del mateix departament del cap de projecte
2. Un empleat només pot imputar tipus de despeses a les quals la seva funció estigui autoritzada
3. La data de finalització d'una activitat (si està informada) ha de ser posterior a la data d'inici d'activitat.
4. Un empleat només pot estar realitzant una activitat a la vegada dins d'un projecte al qual participa.
5. Un cap de projecte ha de ser un empleat de tipus_empleat cap de projecte

2.3 Disseny lògic

2.3.1 Introducció a la transformació d'entitats i interrelacions

En aquest apartat tractarem del disseny lògic d'una base de dades relacional. Partirem del resultat de l'etapa del disseny conceptual expressat mitjançant el model ER i el transformaré en una estructura de dades del model relacional.

Els elements bàsics del model ER són les entitats i les interrelacions:

- Les entitats, quan es tradueixen al model relacional, originen **relacions** (anomenades taules a la majoria dels SGBD del mercat).
- Les interrelacions, en canvi, quan es transformen, poden donar lloc a **claus foranes** d'alguna relació ja obtinguda o poden donar lloc a una nova **relació**.

A la taula següent es resumeix els aspectes més bàsics de les transformacions del model ER al model relacional:

Element del model ER	Transformació al model relacional
Entitat	Relació
Interrelació 1:1	Clau forana
Interrelació 1:N	Clau forana
Interrelació M:N	Relació
Interrelació n-ària	Relació
Interrelació recursiva	Com a les interrelacions no recursives: • Clau forana per a binàries 1:1 i 1:N • Relació per a binàries M:N i n-àries
Entitat dèbil	La clau forana de la interrelació identificadora forma part de la clau primària
Generalització/especialització	• Relació per a l'entitat superclasse • Relació per a cadascuna de les entitats subclasse
Entitat associativa	La transformació de la interrelació que l'origina és alhora la seva transformació

2.3.2 Transformació del model ER al model relacional

Seguint les regles del punt anterior obtenim la següent transformació al model relacional:

- **CIUTAT**(codiCiutat, nomCiutat, paisCiutat, situacioCiutat)
- **CENTRE**(codiCiutat, codiCentre, nomCentre, adreçaCentre, situacioCentre) on {codiCiutat} referencia CIUTAT
- **DEPARTAMENT**(codiDepartament, nomDepartament, codiCiutat, codiCentre, situacioDepartament) on {codiCiutat, CodiCentre} referencia CENTRE
- **TIPUS_EMPLEAT**(codiTipusEmpleat, funcioEmpleat, situacioTipusEmpleat)
- **EMPLEAT**(codiEmpleat, nomEmpleat, cognomsEmpleat, dniEmpleat, costEmpleat, codiDepartament, codiTipusEmpleat, situacioEmpleat) on {codiDepartament} referencia DEPARTAMENT i {codiTipusEmpleat} referencia TIPUS_EMPLEAT
- **REGISTRE_ENTRADA**(codiRegistre, dataRegistre, tipusRegistre, codiEmpleat, situacioRegistreEntrada) on {codiEmpleat} referencia EMPLEAT
- **CLIENT**(codiClient, nomClient, adreçaClient, situacioClient)
- **PROJECTE**(codiProjecte, nomProjecte, dataOberturaProjecte, pressupostProjecte, codiCapProjecte, codiClient, situacioProjecte) on {codiCapProjecte} referencia EMPLEAT(codiEmpleat) i {codiClient} referencia CLIENT
- **PARTICIPA**(codiProjecte, codiEmpleat, situacioParticipa) on {codiProjecte} referencia PROJECTE i {codiEmpleat} referencia EMPLEAT
- **TIPUS_DESPESA**(codiTipusDespesa, nomCodiDespesa, situacioTipusDespesa)
- **DESPESA**(codiDespesa, importDespesa, codiProjecte, codiEmpleat, codiTipusDespesa, situacioDespesa) on {codiProjecte, codiEmpleat} referencia PARTICIPA i {codiTipusDespesa} referencia TIPUS_DESPESA
- **PERMIS**(codiTipusEmpleat, codiTipusDespesa, situacioPermis) on {codiTipusEmpleat} referencia TIPUS_EMPLEAT i {codiTipusDespesa} referencia TIPUS_DESPESA
- **ACTIVITAT**(codiActivitat, dataIniciActivitat, dataFiActivitat, codiProjecte, CodiEmpleat, situacioActivitat) on {codiProjecte, codiEmpleat} referencia PARTICIPA

Per afavorir els accessos a les taules i també per aconseguir que no puguin haver repetits en alguns camps es crearan els següents índexs de clau única:

- **nomCiutatInd**: ciutat(nomCiutat, paisCiutat)
- **nomCentreInd**: centre(nomCentre, codiCiutat)
- **nomDepartamentInd**: departament(nomDepartament, codiCiutat, codiCentre)
- **funcioEmpleatInd**: tipus_empleat(funcioEmpleat)
- **nomCodiDespesaInd**: tipus_despesa(nomCodiDespesa)
- **nomClientInd**: client(nomClient)
- **dniEmpleatInd**: empleat(dniEmpleat, codiDepartament)
- **nomProjecteInd**: projecte(nomProjecte)

NOTA: A l'apartat *Implementació de les taules, índexs i seqüències* hi ha més informació

2.3.3 Especificació dels procediments emmagatzemats

S'han desenvolupat 82 procediments emmagatzemats que cobriran totes les necessitats de les aplicacions que els utilitzin, a més a més també serveixen per fer acomplir les restriccions textuals que el model relacional no és capaç de garantir.

Aquests procediments s'han dissenyat com a operacions sobre cadascuna de les entitats del sistema, tot i que dins de molts procediments es fan comprovacions sobre altres entitats. Totes aquestes operacions són sobre un únic registre de la base de dades, és responsabilitat de les aplicacions iterar sobre aquests procediments per la realització d'operacions d'àmbit massiu de dades.

Tots els procediments inclouen els següents mecanismes:

- Retornen un paràmetre de resultat (**res**) el qual retornarà:
  - En cas de que tot correcte 'OK: *Operació'*
  - En cas d'error 'ERROR: *Descripció de l'error*'
- Tractament d'excepcions
  - Hi ha un tractament específic de certes excepcions que es monitoritzen per donar l'error concret a la variable res. Tots els procediments incorporen també el tractament *"when others"* per no deixar cap excepció sense tractament.
- Tractament de traces
  - Després de cada acció important dins del procediment s'informarà una variable, que servirà per donar més informació en cas de que es doni algun error. Aquesta variable (**trace**) està composta per un comptador (10- ..., 20-...,) i un text que específica quina és l'acció que s'està donant.
  - Tant si el procediment s'executa bé o malament s'escriurà un registre a una taula especial (**traces**) que conté els següents camps:

Traces		
usuari	VARCHAR2(20)	Usuari d'execució
data	DATE	Data d'execució
programa	VARCHAR2(20)	Nom del procediment executat
traça	VARCHAR2(50)	Valor de la variable trace
miss	VARCHAR2(100)	Valor de la variable res
err_num	NUMBER	Valor SQLCODE. Codi d'error SQL
err_msg	VARCHAR2(100)	Valor SQLERRM. Missatge d'error SQL
param_in	VARCHAR2(300)	Paràmetres entrada separats per |
param_out	VARCHAR2(300)	Paràmetres sortida separats per |

- Tractament de transaccions
  - Es confirmem totes les accions un cop que s'han realitzat correctament amb un COMMIT; en cas d'error s'anul·laran les accions amb un ROLLBACK.

El paquet d'operacions estàndard és el següent:

- **altaEntitat: Dona d'alta l'entitat**
    - Es dona d'alta un nou registre amb els paràmetres d'entrada. A la majoria dels casos es fa servir una SEQUENCE per poder tenir un comptador que començarà des de 1 en endavant.
- **modEntitat: Modifica les dades de l'entitat**
    - Es modifiquen les dades pròpies de l'entitat. S'ha de tenir en compte que al modificar el nom d'una entitat pot entrar en conflicte amb el valor únic d'aquest camp que pugui tenir un altre registre de la base de dades.
- **baixaEntitat: Canvia la situació a baixa**
    - Es comprova si abans l'entitat estava en situació d'alta. També s'han de tenir en compte les dependències amb altres entitats, o sigui, aquesta entitat no podrà ser donada de baixa si cap altra entitat li fa referència.
- **activaEntitat: Canvia la situació a alta si aquesta entitat estava en baixa**
    - L'entitat ha d'estar en situació de baixa
- **elimEntitat: Elimina físicament l'entitat**
    - Només es permet eliminar una entitat que estigui en situació baixa
- **consEntitat: Consulta per la clau la entitat**
    - Es retornen tots els camps de l'entitat accedint pel paràmetre d'entrada codiEntitat.
- **consNomEntitat: Consulta pel nom "real" l'entitat**
    - Es retornen tots els camps de l'entitat accedint pel paràmetre d'entrada que identifica al món real l'entitat (nom de la ciutat, etc.).

	Ciutat	Centre	Departament	Tipus_Empleat	Empleat	Registre_entrada	Tipus_Despesa	Client	Projecte	Participa	Permis	Despesa	Activitat
alta	X	X	X	X	X		X	X	X	X	X	X	X
mod	X	X	X	X	X		X	X	X			X	X
baixa	X	X	X	X	X		X	X	X	X	X	X	X
activa	X	X	X	X	X		X	X	X	X	X	X	X
elim	X	X	X	X	X		X	X	X	X	X	X	X
cons	X	X	X	X	X		X	X	X			X	X
consNom	X	X	X	(1)	(2)		X	X	X				

Quadre d'operacions estàndard

Operacions amb un nom diferent però amb la mateixa funcionalitat:

(1) consFuncioTipusEmpleat : Consulta la funció del tipus d'empleat

(2) consDNIEmpleat: Consulta el DNI d'un empleat

	Ciutat	Centre	Departament	Tipus_Empleat	Empleat	Registre_entrada	Tipus_Despesa	Client	Projecte	Participa	Permis	Despesa	Activitat
entradaRegistreEntrada						X							
modCapProjecte									X				
modDepartamentEmpleat					X								
modFuncioEmpleat					X								
sortidaRegistreEntrada						X							
tancaProjecte									X				

Quadre d'operacions especials

Dels anteriors quadres de distribució d'operacions podem observar:

- Les taules Ciutat, Centre, Departament, Tipus_Empleat, Tipus_Despesa i Client tenen totes les operacions estàndard i només que aquestes. Aquestes són entitats de dades "fixes" és a dir, serveixen per fer de "registre" d'entitats molt clares i definides al model de negoci del sistema d'informació, per la qual cosa cal fer un manteniment d'operacions complet.
- Les taules Participa i Permis no tenen operacions de modificació ni de consulta. Això es degut a que aquestes taules representen relacions M:N i no incorporen cap atribut nou que pugui ser consultat o modificat.
- Les taules Despesa i Activitat només tenen una consulta per codi ja que aquestes taules no tenen cap altre camp identificador propi. Despesa fa referència a una tipologia de despeses (tipus_despesa) que serveix per lligar els tipus de despeses amb els tipus d'empleats que poden imputar-les.
- La taula Registre_Entrada és una taula molt simple que serveix únicament per registrar les entrades i les sortides dels empleats, per la qual cosa només te dues operacions (la de l'entrada i la de la sortida).
- La taula Empleat te dues operacions especials (modificar el Departament i la funció d'un empleat) seguint els requeriments del producte.
- La taula Projecte te dues operacions especials (modificar el Cap del Projecte i tancar un projecte) seguint els requeriments del producte.

Els procediments s'han agrupat dins de paquets per afavorir la seva gestió. Els paquets són els següents:

- **centre_actions. Procediments sobre les entitats:**
  - centre
  - ciutat
  - departament
- **empleat_actions. Procediments sobre les entitats:**
  - tipus_empleat
  - empleat
  - registre_entrada
- **projecte_actions. Procediments sobre les entitats:**
  - tipus_despesa
  - client
  - projecte
  - participa
  - permis
  - despesa
  - activitat

A continuació enumerarem tots els procediments desenvolupats amb una petita descripció i la seva signatura (capçalera de crida):

Paquet centre_actions: Accions sobre centre, ciutat i departament		
Procediment	**Descripció**	**Signatura**
altaCiutat	Es dona d'alta un nou registre a la taula de ciutats en situació activa	pnomCiutat IN Ciutat.nomCiutat%TYPE, ppaisCiutat IN Ciutat.paisCiutat%TYPE, pcodiCiutat OUT Ciutat.codiCiutat%TYPE, res OUT VARCHAR2
modCiutat	Es modifiquen els camps propis d'un registre existent a la taula de ciutats en situació activa	pcodiCiutat IN Ciutat.codiCiutat%TYPE, pnomCiutat IN Ciutat.nomCiutat%TYPE, ppaisCiutat IN Ciutat.paisCiutat%TYPE, res OUT VARCHAR2
baixaCiutat	Es modifica un registre existent en situació activa a la taula de ciutats a situació baixa	pcodiCiutat IN Ciutat.codiCiutat%TYPE, res OUT VARCHAR2
activaCiutat	Es modifica un registre existent en situació baixa a la taula de ciutats a situació activa	pcodiCiutat IN Ciutat.codiCiutat%TYPE, res OUT VARCHAR2
elimCiutat	S'elimina físicament el registre existent en situació baixa a la taula de ciutats	pcodiCiutat IN Ciutat.codiCiutat%TYPE, res OUT VARCHAR2
consCiutat	Es consulten tots els camps del registre existent a la taula de ciutats en situació activa	pcodiCiutat IN Ciutat.codiCiutat%TYPE, pnomCiutat OUT Ciutat.nomCiutat%TYPE, ppaisCiutat OUT Ciutat.paisCiutat%TYPE, res OUT VARCHAR2
consNomCiutat	Es consulten tots els camps del registre existent a la taula de ciutats en situació activa	pnomCiutat IN Ciutat.nomCiutat%TYPE, ppaisCiutat IN Ciutat.paisCiutat%TYPE, pcodiCiutat OUT Ciutat.codiCiutat%TYPE, res OUT VARCHAR2
altaCentre	Es dona d'alta un nou registre a la taula de centres en situació activa	pcodiCiutat IN Centre.codiCiutat%TYPE, pnomCentre IN Centre.nomCentre%TYPE, padreçaCentre IN Centre.adreçaCentre%TYPE, pcodiCentre OUT Centre.codiCentre%TYPE, res OUT VARCHAR2

Paquet centre_actions: Accions sobre centre, ciutat i departament		
Procediment	**Descripció**	**Signatura**

modCentre	Es modifiquen els camps propis d'un registre existent a la taula de centres en situació activa	`pcodiCiutat IN Centre.codiCiutat%TYPE,`
		`pcodiCentre IN Centre.codiCentre%TYPE,`
		`pnomCentre IN Centre.nomCentre%TYPE,`
		`padreçaCentre IN`
		`Centre.adreçaCentre%TYPE,`
		`res OUT VARCHAR2`

baixaCentre	Es modifica un registre existent en situació activa a la taula de centres a situació baixa	`pcodiCiutat IN Centre.codiCiutat%TYPE,`
		`pcodiCentre IN Centre.codiCentre%TYPE,`
		`res OUT VARCHAR2`

activaCentre	Es modifica un registre existent en situació baixa a la taula de centres a situació activa	`pcodiCiutat IN Centre.codiCiutat%TYPE,`
		`pcodiCentre IN Centre.codiCentre%TYPE,`
		`res OUT VARCHAR2`

elimCentre	S'elimina físicament el registre existent en situació baixa a la taula de centres	`pcodiCiutat IN Centre.codiCiutat%TYPE,`
		`pcodiCentre IN Centre.codiCentre%TYPE,`
		`res OUT VARCHAR2`

consCentre	Es consulten tots els camps del registre existent a la taula de centres en situació activa	`pcodiCiutat IN Centre.codiCiutat%TYPE,`
		`pcodiCentre IN Centre.codiCentre%TYPE,`
		`pnomCentre OUT Centre.nomCentre%TYPE,`
		`padreçaCentre OUT`
		`Centre.adreçaCentre%TYPE,`
		`res OUT VARCHAR2`

consNomCentre	Es consulten tots els camps del registre existent a la taula de centres en situació activa	`pnomCentre IN Centre.nomCentre%TYPE,`
		`pcodiCiutat IN Centre.codiCiutat%TYPE,`
		`pcodiCentre OUT Centre.codiCentre%TYPE,`
		`padreçaCentre OUT`
		`Centre.adreçaCentre%TYPE,`
		`res OUT VARCHAR2`

Paquet centre_actions: Accions sobre centre, ciutat i departament		
Procediment	**Descripció**	**Signatura**
altaDepartament	Es dona d'alta un nou registre a la taula de departaments en situació activa	pnomDepartament IN Departament.nomDepartament%TYPE, pcodiCiutat IN Departament.codiCiutat%TYPE, pcodiCentre IN Departament.codiCentre%TYPE, pcodiDepartament OUT Departament.codiDepartament%TYPE, res OUT VARCHAR2
modDepartament	Es modifiquen els camps propis d'un registre existent a la taula de departaments en situació activa	PcodiDepartament IN Departament.codiDepartament%TYPE, pcodiCiutat IN Departament.codiCiutat%TYPE, pcodiCentre IN Departament.codiCentre%TYPE, pnomDepartament IN Departament.nomDepartament%TYPE, res OUT VARCHAR2
baixaDepartament	Es modifica un registre existent en situació activa a la taula de departaments a situació baixa	PcodiDepartament IN Departament.codiDepartament%TYPE, res OUT VARCHAR2
activaDepartament	Es modifica un registre existent en situació baixa a la taula de departaments a situació activa	PcodiDepartament IN Departament.codiDepartament%TYPE, res OUT VARCHAR2
elimDepartament	S'elimina físicament el registre existent en situació baixa a la taula de departaments	PcodiDepartament IN Departament.codiDepartament%TYPE, res OUT VARCHAR2

Paquet centre_actions: Accions sobre centre, ciutat i departament		
Procediment	**Descripció**	**Signatura**
consDepartament	Es consulten tots els camps del registre existent a la taula de departaments en situació activa	pcodiDepartament IN Departament.codiDepartament%TYPE, pnomDepartament OUT Departament.nomDepartament%TYPE, pcodiCiutat OUT Departament.codiCiutat%TYPE, pcodiCentre OUT Departament.codiCentre%TYPE, res OUT VARCHAR2
consNomDepartament	Es consulten tots els camps del registre existent a la taula de departaments en situació activa	pnomDepartament IN Departament.nomDepartament%TYPE, pcodiCiutat IN Departament.codiCiutat%TYPE, pcodiCentre IN Departament.codiCentre%TYPE, pcodiDepartament OUT Departament.codiDepartament%TYPE, res OUT VARCHAR2

Paquet empleat_actions: Accions sobre tipus_empleat, empleat i registre_entrada		
Procediment	**Descripció**	**Signatura**

altaTipusEmpleat	Es dona d'alta un nou registre a la taula de ciutats en situació activa	`pfuncioEmpleat IN Tipus_Empleat.funcioEmpleat%TYPE,`
		`pcodiTipusEmpleat OUT`
		`Tipus_Empleat.codiTipusEmpleat%TYPE,`
		`res OUT VARCHAR2`

modTipusEmpleat	Es modifiquen els camps propis d'un registre existent a la taula de ciutats en situació activa	`pcodiTipusEmpleat IN`
		`Tipus_Empleat.codiTipusEmpleat%TYPE,`
		`pfuncioEmpleat IN Tipus_Empleat.funcioEmpleat%TYPE,`
		`res OUT VARCHAR2`

baixaTipusEmpleat	Es modifica un registre existent en situació activa a la taula de ciutats a situació baixa	`pcodiTipusEmpleat IN`
		`Tipus_Empleat.codiTipusEmpleat%TYPE,`
		`res OUT VARCHAR2`

activaTipusEmpleat	Es modifica un registre existent en situació baixa a la taula de ciutats a situació activa	`pcodiTipusEmpleat IN`
		`Tipus_Empleat.codiTipusEmpleat%TYPE,`
		`res OUT VARCHAR2`

elimTipusEmpleat	S'elimina físicament el registre existent en situació baixa a la taula de ciutats	`pcodiTipusEmpleat IN`
		`Tipus_Empleat.codiTipusEmpleat%TYPE,`
		`res OUT VARCHAR2`

consTipusEmpleat	Es consulten tots els camps del registre existent a la taula de ciutats en situació activa	`pcodiTipusEmpleat IN`
		`Tipus_Empleat.codiTipusEmpleat%TYPE,`
		`pfuncioEmpleat OUT`
		`Tipus_Empleat.funcioEmpleat%TYPE,`
		`res OUT VARCHAR2`

consFuncioTipusEmpleat	Es consulten tots els camps del registre existent a la taula de ciutats en situació activa	`pfuncioEmpleat IN Tipus_Empleat.funcioEmpleat%TYPE,`
		`pcodiTipusEmpleat OUT`
		`Tipus_Empleat.codiTipusEmpleat%TYPE,`
		`res OUT VARCHAR2`

Paquet empleat_actions: Accions sobre tipus_empleat, empleat i registre_entrada		
Procediment	**Descripció**	**Signatura**

altaEmpleat	Es dona d'alta un nou registre a la taula d'empleats en situació activa	`pnomEmpleat IN Empleat.nomEmpleat%TYPE,`
		`pcognomsEmpleat IN Empleat.cognomsEmpleat%TYPE,`
		`pdniEmpleat IN Empleat.dniEmpleat%TYPE,`
		`pcostEmpleat IN Empleat.costEmpleat%TYPE,`
		`pcodiDepartament IN Empleat.codiDepartament%TYPE,`
		`pcodiTipusEmpleat IN Empleat.codiTipusEmpleat%TYPE,`
		`pcodiEmpleat OUT Empleat.codiEmpleat%TYPE,`
		`res OUT VARCHAR2`

modEmpleat	Es modifiquen els camps propis d'un registre existent a la taula d'empleats en situació activa	`pcodiEmpleat IN Empleat.codiEmpleat%TYPE,`
		`pnomEmpleat IN Empleat.nomEmpleat%TYPE,`
		`pcognomsEmpleat IN Empleat.cognomsEmpleat%TYPE,`
		`pdniEmpleat IN Empleat.dniEmpleat%TYPE,`
		`pcostEmpleat IN Empleat.costEmpleat%TYPE,`
		`res OUT VARCHAR2`

modDepartamentEmpleat	Es dona d'alta un nou registre a la taula d'empleats en situació activa amb les noves dades de Departament i es modifica el registre anterior a situació baixa.	`pcodiEmpleat IN Empleat.codiEmpleat%TYPE,`
		`pcodiCentre IN Centre.codiCentre%TYPE,`
		`pcodiDepartament IN Empleat.codiDepartament%TYPE,`
		`res OUT VARCHAR2`

modFuncioEmpleat	Es dona d'alta un nou registre a la taula d'empleats en situació activa amb les noves dades de funció de l'empleat i es modifica el registre anterior a situació baixa.	`pcodiEmpleat IN Empleat.codiEmpleat%TYPE,`
		`pcodiTipusEmpleat IN Empleat.codiTipusEmpleat%TYPE,`
		`res OUT VARCHAR2`

baixaEmpleat	Es modifica un registre existent en situació activa a la taula d'empleats a situació baixa	`pcodiEmpleat IN Empleat.codiEmpleat%TYPE,` `res OUT VARCHAR2`
activaEmpleat	Es modifica un registre existent en situació baixa a la taula d'empleats a situació activa	`pcodiEmpleat IN Empleat.codiEmpleat%TYPE,` `res OUT VARCHAR2`
elimEmpleat	S'elimina físicament el registre existent en situació baixa a la taula d'empleats	`pcodiEmpleat IN Empleat.codiEmpleat%TYPE,` `res OUT VARCHAR2`

Paquet empleat_actions: Accions sobre tipus_empleat, empleat i registre_entrada		
Procediment	**Descripció**	**Signatura**
consEmpleat	Es consulten tots els camps del registre existent a la taula d'empleats en situació activa	pcodiEmpleat IN Empleat.codiEmpleat%TYPE, pnomEmpleat OUT Empleat.nomEmpleat%TYPE, pcognomsEmpleat OUT Empleat.cognomsEmpleat%TYPE, pdniEmpleat OUT Empleat.dniEmpleat%TYPE, pcostEmpleat OUT Empleat.costEmpleat%TYPE, pcodiDepartament OUT Empleat.codiDepartament%TYPE, pcodiTipusEmpleat OUT Empleat.codiTipusEmpleat%TYPE, res OUT VARCHAR2
consDNIEmpleat	Es consulten tots els camps del registre existent a la taula d'empleats en situació activa	pdniEmpleat IN Empleat.dniEmpleat%TYPE, pcodiDepartament IN Empleat.codiDepartament%TYPE, pcodiEmpleat OUT Empleat.codiEmpleat%TYPE, pnomEmpleat OUT Empleat.nomEmpleat%TYPE, pcognomsEmpleat OUT Empleat.cognomsEmpleat%TYPE, pcostEmpleat OUT Empleat.costEmpleat%TYPE, pcodiTipusEmpleat OUT Empleat.codiTipusEmpleat%TYPE, res OUT VARCHAR2
entradaRegistreEntrada	Es dona d'alta un nou registre a la taula de registre d'entrada en situació activa i tipus de registre entrada	pcodiEmpleat IN Registre_Entrada.codiEmpleat%TYPE, pdataRegistre IN Registre_Entrada.dataRegistre%TYPE, pcodiRegistre OUT Registre_Entrada.codiRegistre%TYPE, res OUT VARCHAR2
sortidaRegistreEntrada	Es dona d'alta un nou registre a la taula de registre d'entrada en situació activa i tipus de registre sortida	pcodiEmpleat IN Registre_Entrada.codiEmpleat%TYPE, pdataRegistre IN Registre_Entrada.dataRegistre%TYPE, pcodiRegistre OUT Registre_Entrada.codiRegistre%TYPE, res OUT VARCHAR2

Paquet projecte_actions: Accions tipus_despesa, client, projecte, participa, permis, despesa i activitat		
Procediment	**Descripció**	**Signatura**
altaTipusDespesa	Es dona d'alta un nou registre a la taula de tipus_despesa en situació activa	pnomCodiDespesa IN Tipus_Despesa.nomCodiDespesa%TYPE, pcodiTipusDespesa OUT Tipus_Despesa.codiTipusDespesa%TYPE, res OUT VARCHAR2
modTipusDespesa	Es modifiquen els camps propis d'un registre existent a la taula de tipus_despesa en situació activa	pcodiTipusDespesa IN Tipus_Despesa.codiTipusDespesa%TYPE, pnomCodiDespesa IN Tipus_Despesa.nomCodiDespesa%TYPE, res OUT VARCHAR2
baixaTipusDespesa	Es modifica un registre existent en situació activa a la taula de tipus_despesa a situació baixa	pcodiTipusDespesa IN Tipus_Despesa.codiTipusDespesa%TYPE, res OUT VARCHAR2
activaTipusDespesa	Es modifica un registre existent en situació baixa a la taula de tipus_despesa a situació activa	pcodiTipusDespesa IN Tipus_Despesa.codiTipusDespesa%TYPE, res OUT VARCHAR2
elimTipusDespesa	S'elimina físicament el registre existent en situació baixa a la taula de tipus_despesa	pcodiTipusDespesa IN Tipus_Despesa.codiTipusDespesa%TYPE, res OUT VARCHAR2
consTipusDespesa	Es consulten tots els camps del registre existent a la taula de tipus_despesa en situació activa	pcodiTipusDespesa IN Tipus_Despesa.codiTipusDespesa%TYPE, pnomCodiDespesa OUT Tipus_Despesa.nomCodiDespesa%TYPE, res OUT VARCHAR2
consNomTipusDespesa	Es consulten tots els camps del registre existent a la taula de tipus_despesa en situació activa	pnomCodiDespesa IN Tipus_Despesa.nomCodiDespesa%TYPE, pcodiTipusDespesa OUT Tipus_Despesa.codiTipusDespesa%TYPE, res OUT VARCHAR2
altaClient	Es dona d'alta un nou registre a la taula de clients en situació activa	pnomClient IN Client.nomClient%TYPE, padreçaClient IN Client.adreçaClient%TYPE, pcodiClient OUT Client.codiClient%TYPE, res OUT VARCHAR2
modClient	Es modifiquen els camps propis d'un registre existent a la taula de clients en situació activa	pcodiClient IN Client.codiClient%TYPE, pnomClient IN Client.nomClient%TYPE, padreçaClient IN Client.adreçaClient%TYPE, res OUT VARCHAR2

Paquet projecte_actions: Accions tipus_despesa, client, projecte, participa, permis, despesa i activitat		
Procediment	**Descripció**	**Signatura**
baixaClient	Es modifica un registre existent en situació activa a la taula de clients a situació baixa	`pcodiClient IN Client.codiClient%TYPE,` `res OUT VARCHAR2`
activaClient	Es modifica un registre existent en situació baixa a la taula de clients a situació activa	`pcodiClient IN Client.codiClient%TYPE,` `res OUT VARCHAR2`
elimClient	S'elimina físicament el registre existent en situació baixa a la taula de clients	`pcodiClient IN Client.codiClient%TYPE,` `res OUT VARCHAR2`

consClient	Es consulten tots els camps del registre existent a la taula de clients en situació activa	`pcodiClient IN Client.codiClient%TYPE,`
		`pnomClient OUT Client.nomClient%TYPE,`
		`padreçaClient OUT Client.adreçaClient%TYPE,`
		`res OUT VARCHAR2`

consNomClient	Es consulten tots els camps del registre existent a la taula de clients en situació activa	`pnomClient IN Client.nomClient%TYPE,`
		`pcodiClient OUT Client.codiClient%TYPE,`
		`padreçaClient OUT Client.adreçaClient%TYPE,`
		`res OUT VARCHAR2`

altaProjecte	Es dona d'alta un nou registre a la taula de projectes en situació activa	`pnomProjecte IN Projecte.nomProjecte%TYPE,`
		`pdataOberturaProjecte IN`
		`Projecte.dataOberturaProjecte%TYPE,`
		`ppressupostProjecte IN`
		`Projecte.pressupostProjecte%TYPE,`
		`pcodiCapProjecte IN Projecte.codiCapProjecte%TYPE,`
		`pcodiClient IN Projecte.codiClient%TYPE,`
		`pcodiProjecte OUT Projecte.codiProjecte%TYPE,`
		`res OUT VARCHAR2`

modProjecte	Es modifiquen els camps propis d'un registre existent a la taula de projectes en situació activa	`pcodiProjecte IN Projecte.codiProjecte%TYPE,`
		`pnomProjecte IN Projecte.nomProjecte%TYPE,`
		`pdataOberturaProjecte IN`
		`Projecte.dataOberturaProjecte%TYPE,`
		`ppressupostProjecte IN`
		`Projecte.pressupostProjecte%TYPE,`
		`res OUT VARCHAR2`

modCapProjecte	Es modifica el cap de projecte d'un registre existent a la taula de projectes en situació activa	`pcodiProjecte IN Projecte.codiProjecte%TYPE,`
		`pcodiCapProjecte IN Projecte.codiCapProjecte%TYPE,`
		`res OUT VARCHAR2`

Paquet projecte_actions: Accions tipus_despesa, client, projecte, participa, permis, despesa i activitat		
Procediment	**Descripció**	**Signatura**
baixaProjecte	Es modifica un registre existent en situació activa a la taula de projectes a situació baixa	pcodiProjecte IN Projecte.codiProjecte%TYPE, res OUT VARCHAR2
activaProjecte	Es modifica un registre existent en situació baixa a la taula de projectes a situació activa	pcodiProjecte IN Projecte.codiProjecte%TYPE, res OUT VARCHAR2
tancaProjecte		pcodiProjecte IN Projecte.codiProjecte%TYPE, res OUT VARCHAR2
elimProjecte	S'elimina físicament el registre existent en situació baixa a la taula de projectes	pcodiProjecte IN Projecte.codiProjecte%TYPE, res OUT VARCHAR2
consProjecte	Es consulten tots els camps del registre existent a la taula de projectes en situació activa	pcodiProjecte IN Projecte.codiProjecte%TYPE, pnomProjecte OUT Projecte.nomProjecte%TYPE, pdataOberturaProjecte OUT Projecte.dataOberturaProjecte%TYPE, ppressupostProjecte OUT Projecte.pressupostProjecte%TYPE, pcodiCapProjecte OUT Projecte.codiCapProjecte%TYPE, pcodiClient OUT Projecte.codiClient%TYPE, res OUT VARCHAR2
consNomProjecte	Es consulten tots els camps del registre existent a la taula de projectes en situació activa	pnomProjecte IN Projecte.nomProjecte%TYPE, pcodiProjecte OUT Projecte.codiProjecte%TYPE, pdataOberturaProjecte OUT Projecte.dataOberturaProjecte%TYPE, ppressupostProjecte OUT Projecte.pressupostProjecte%TYPE, pcodiCapProjecte OUT Projecte.codiCapProjecte%TYPE, pcodiClient OUT Projecte.codiClient%TYPE, res OUT VARCHAR2
altaParticipa	Es dona d'alta un nou registre a la taula de participació en situació activa	pcodiProjecte IN Participa.codiProjecte%TYPE, pcodiEmpleat IN Participa.codiEmpleat%TYPE, res OUT VARCHAR2
baixaParticipa	Es modifica un registre existent en situació activa a la taula de participació a situació baixa	pcodiProjecte IN Participa.codiProjecte%TYPE, pcodiEmpleat IN Participa.codiEmpleat%TYPE, res OUT VARCHAR2

Paquet projecte_actions: Accions tipus_despesa, client, projecte, participa, permis, despesa i activitat		
Procediment	**Descripció**	**Signatura**

activaParticipa	Es modifica un registre existent en situació baixa a la taula de participació a situació activa	`pcodiProjecte IN Participa.codiProjecte%TYPE,`
		`pcodiEmpleat IN Participa.codiEmpleat%TYPE,`
		`res OUT VARCHAR2`

elimParticipa	S'elimina físicament el registre existent en situació baixa a la taula de participació	`pcodiProjecte IN Participa.codiProjecte%TYPE,`
		`pcodiEmpleat IN Participa.codiEmpleat%TYPE,`
		`res OUT VARCHAR2`

altaPermis	Es dona d'alta un nou registre a la taula de permís en situació activa	`pcodiTipusEmpleat IN Permis.codiTipusEmpleat%TYPE,`
		`pcodiTipusDespesa IN Permis.codiTipusDespesa%TYPE,`
		`res OUT VARCHAR2`

baixaPermis	Es modifica un registre existent en situació activa a la taula de permís a situació baixa	`pcodiTipusEmpleat IN Permis.codiTipusEmpleat%TYPE,`
		`pcodiTipusDespesa IN Permis.codiTipusDespesa%TYPE,`
		`res OUT VARCHAR2`

activaPermis	Es modifica un registre existent en situació baixa a la taula de permís a situació activa	`pcodiTipusEmpleat IN Permis.codiTipusEmpleat%TYPE,`
		`pcodiTipusDespesa IN Permis.codiTipusDespesa%TYPE,`
		`res OUT VARCHAR2`

elimPermis	S'elimina físicament el registre existent en situació baixa a la taula de permís	`pcodiTipusEmpleat IN Permis.codiTipusEmpleat%TYPE,`
		`pcodiTipusDespesa IN Permis.codiTipusDespesa%TYPE,`
		`res OUT VARCHAR2`

altaDespesa	Es dona d'alta un nou registre a la taula de despeses en situació activa	`pimportDespesa IN Despesa.importDespesa%TYPE,`
		`pcodiProjecte IN Despesa.codiProjecte%TYPE,`
		`pcodiEmpleat IN Despesa.codiEmpleat%TYPE,`
		`pcodiTipusDespesa IN Despesa.codiTipusDespesa%TYPE,`
		`pcodiDespesa OUT Despesa.codiDespesa%TYPE,`
		`res OUT VARCHAR2`

modDespesa	Es modifiquen els camps propis d'un registre existent a la taula de despeses en situació activa	`pcodiDespesa IN Despesa.codiDespesa%TYPE,`
		`pimportDespesa IN Despesa.importDespesa%TYPE,`
		`res OUT VARCHAR2`

baixaDespesa	Es modifica un registre existent en situació activa a la taula de despeses a situació baixa	`pcodiDespesa IN Despesa.codiDespesa%TYPE,` `res OUT VARCHAR2`
activaDespesa	Es modifica un registre existent en situació baixa a la taula de despeses a situació activa	`pcodiDespesa IN Despesa.codiDespesa%TYPE,` `res OUT VARCHAR2`
elimDespesa	S'elimina físicament el registre existent en situació baixa a la taula de despeses	`pcodiDespesa IN Despesa.codiDespesa%TYPE,` `res OUT VARCHAR2`

Paquet projecte_actions: Accions tipus_despesa, client, projecte, participa, permis, despesa i activitat		
Procediment	**Descripció**	**Signatura**
consDespesa	Es consulten tots els camps del registre existent a la taula de despeses en situació activa	pcodiDespesa IN Despesa.codiDespesa%TYPE, pimportDespesa OUT Despesa.importDespesa%TYPE, pcodiProjecte OUT Despesa.codiProjecte%TYPE, pcodiEmpleat OUT Despesa.codiEmpleat%TYPE, pcodiTipusDespesa OUT Despesa.codiTipusDespesa%TYPE, res OUT VARCHAR2
altaActivitat	Es dona d'alta un nou registre a la taula d'activitats en situació activa	pdataIniciActivitat IN Activitat.dataIniciActivitat%TYPE, pdataFiActivitat IN Activitat.dataFiActivitat%TYPE, pcodiProjecte IN Activitat.codiProjecte%TYPE, pcodiEmpleat IN Activitat.codiEmpleat%TYPE, pcodiActivitat OUT Activitat.codiActivitat%TYPE, res OUT VARCHAR2
modActivitat	Es modifiquen els camps propis d'un registre existent a la taula d'activitats en situació activa	pcodiActivitat IN Activitat.codiActivitat%TYPE, pdataIniciActivitat IN Activitat.dataIniciActivitat%TYPE, pdataFiActivitat IN Activitat.dataFiActivitat%TYPE, res OUT VARCHAR2
baixaActivitat	Es modifica un registre existent en situació activa a la taula d'activitats a situació baixa	pcodiActivitat IN Activitat.codiActivitat%TYPE, res OUT VARCHAR2
activaActivitat	Es modifica un registre existent en situació baixa a la taula d'activitats a situació activa	pcodiActivitat IN Activitat.codiActivitat%TYPE, res OUT VARCHAR2
elimActivitat	S'elimina físicament el registre existent en situació baixa a la taula d'activitats	pcodiActivitat IN Activitat.codiActivitat%TYPE, res OUT VARCHAR2
consActivitat	Es consulten tots els camps del registre existent a la taula d'activitats en situació activa	pcodiActivitat IN Activitat.codiActivitat%TYPE, pdataIniciActivitat OUT Activitat.dataIniciActivitat%TYPE, pdataFiActivitat OUT Activitat.dataFiActivitat%TYPE, pcodiProjecte OUT Activitat.codiProjecte%TYPE, pcodiEmpleat OUT Activitat.codiEmpleat%TYPE, res OUT VARCHAR2

2.4 Disseny físic

2.4.1 Funcionalitat i objectius dels SGBD

Una **base de dades (BD)** és un conjunt estructurat de dades que representa entitats i les seves interrelacions, Aquesta representació informàtica integrada ha de poder ser consultada i actualitzada de forma compartida per molts usuaris de tipus diversos. El **sistema de gestió de bases de dades (SGBD)** és el programari especialitzat que en facilita la utilització, un programari complex que actualment constitueix una part essencial de tot sistema d'informació (SI).

Objectius i funcionalitat dels SGBD :

- Consultes no predefinides i complexes
    - Els usuaris poden fer consultes de qualsevol tipus i complexitat directament a l'SGBD. Aquest ha de respondre immediatament sense que estiguin preestablertes, és a dir, sense que s'hagi d'escriure, compilar i executar un programa específic per a cada consulta. L'usuari ha de poder formular la consulta en un llenguatge i el sistema l'ha d'interpretar directament. Però també s'ha de poder escriure programes amb consultes incorporades. El llenguatge que es fa servir universalment per a assolir aquest doble objectiu (consultes no predefinides i complexes) és el **llenguatge SQL**.
- Flexibilitat i independència
    - Interessa obtenir la màxima independència entre les dades i els processos d'usuari. Cal que es pugui fer tot tipus de canvis tecnològics i canvis en la descripció de la base de dades (BD) sense que s'hagi de modificar els programes d'aplicació ja escrits ni variar la manera de fer les consultes directes.
- Integritat de les dades
    - Quan l'SGBD detecta que un programa, o un usuari, intenta fer una operació que va contra les regles declarades en definir la BD, no li ho ha de permetre, i li ha de tornar un estat d'error. En dissenyar una BD per a un SI concret i escriure'n l'esquema no definirem solament les dades, sinó també les regles d'integritat.

      Els processos de restauració de què tot SGBD disposa poden reconstruir la BD fins a un estat consistent, correcte, anterior a l'incident. Això s'acostuma a fer gràcies a l'obtenció de còpies periòdiques de els dades, que se solen anomenar **còpies de seguretat**, i al manteniment continu d'un diari en el qual l'SGBD anota tots els canvis que es fan a la BD.
- Concurrència d'usuaris
    - Un objectiu fonamental dels SGBD és permetre que un nombre elevat d'usuaris puguin accedir concurrentment a la mateixa BD

      Una **transacció** consisteix en un conjunt d'operacions simples que s'executen com una unitat indivisible. Els SGBD han d'aconseguir fer que el conjunt d'operacions d'una transacció mai no s'executi parcialment: o s'executen totes o no se n'executa cap.
- Seguretat
    - En el camp dels SGBD, el terme *seguretat* s'acostuma a utilitzar per a fer referència als temes relatius a la confidencialitat, la privacitat, les autoritzacions, els drets d'accés, etc.
- Altres objectius
    - Servir eficientment els *data warehouse*
    - Adaptar-se al desenvolupament orientat a l'objecte
    - Incorporar el temps com un element de caracterització de la informació
    - Adaptar-se al món d'Internet

2.4.2 Introducció a SQL

SQL, és el llenguatge estàndard ANSI/ISO de definició, manipulació i control de bases de dades relacionals. És un llenguatge declaratiu, només s'ha de dir què és vol fer. En canvi, en els llenguatges procedimentals cal especificar com s'ha de fer qualsevol cosa sobre la base de dades. L'SQL és un llenguatge molt semblant al llenguatge natural, concretament s'assembla a l'anglès, i és molt expressiu. Per aquestes raons, i com a llenguatge estàndard, l'SQL és un llenguatge am què es pot accedir a tots els sistemes relacionals comercials.

Comencem amb una breu explicació de la manera com l'SQL ha arribat a ser el llenguatge estàndard de les bases de dades relacionals.

Al principi dels anys setanta els laboratoris d'investigació Santa Teresa d'IBM van començar a treballar en el projecte System R. L'objectiu d'aquest projecte era implementar un prototipus s'SGBD relacional i, per tant, necessitaven també investigar en el camp dels llenguatges de bases de dades relacionals. A mitjan anys setanta, el projecte d'IBM va donar un primer llenguatge anomenat SEQUEL (*Structured English Query Language*), que per raons legals més endavant es va denominar SQL (*Structured Query Language*). Al final de la dècada dels setanta i al principi de la dels vuitanta, un cop finalitzat el projecte System R, IBM i altres empreses van començar a utilitzar l'SQL en els seus SGBD relacionals, i així aquest llenguatge va adquirir una gran popularitat.

El 1982, ANSI (*American National Standards Institute*) va encarregar a un dels seus comitès (X3H2) la definició d'un llenguatge de bases de dades relacionals. Aquest comitè, després d'avaluar diferents llenguatges, i davant l'acceptació comercial de l'SQL, va escollir com a llenguatge estàndard un llenguatge basat en l'SQL gairebé en la seva totalitat. L'SQL va esdevenir oficialment el llengautge estàndard d'ANSI l'any 1986, i d'ISO (*International Standards Organization*) el 1987. També ha estat adoptat com a llenguatge estàndard per FIPS (*Federal Information Processing Standard*), Unix X/Open i SAA (*Systems Application Architecture*) d'IBM.

L'any 1989, l'estàndard fou objecte d'una revisió i una ampliació que van donar lloc al llenguatge que es coneix amb el nom d'SQL1 o SQL89. I, l'any 1992, l'estàndard va tornar a ser revisat i ampliat considerablement per a cobrir mancances de la versió anterior. Aquesta nova versió de l'SQL, que es coneix amb el nom d'SQL2 o SQL92- , és la que actualment és troba en ús. Amb la sigla SQL sempre ens referim a l'SQL92, ja que aquest té com a subconjunt l'SQL89 i , per tant, tot el que era vàlid a l'SQL89 ho continuarà essent a l'SQL92.

2.4.3 Introducció al disseny físic

Després del disseny lògic de la base de dades s'arriba al disseny físic tenint en compte les característiques de cada sistema gestor de base de dades (SGBD).

Els components físics de cada SGBD són específics, i el constructor els ha dissenyat pensant en el sistema en què ha de funcionar, per tal de treure'n el màxim profit. El disseny físic de la base de dades parteix del disseny lògic, i ha de tenir present les peculiaritats de cada gestor i adaptar-s'hi convenientment.

Cada SGBD ha desenvolupat un llenguatge propi, fet a mida pel mateix constructor, per a implementar el disseny físic de la base de dades, d'acord amb les característiques de l'entorn, i per a obtenir el màxim de rendiment del maquinari, del sistema operatiu i del mateix gestor. De fet, es podria considerar com una ampliació del llenguatge SQL estàndard, amb aquelles clàusules pròpies, que cada gestor necessita per a definir els components del disseny físic. No obstant això, hi ha una gran similitud o equivalència entre gran part dels components, dels diferents gestors.

També formen part del disseny físic els aspectes relacionats amb el rendiment de les aplicacions, i la manera d'ajustar el disseny de la base de dades per a millorar-lo.

NOTA: L'etapa del disseny físic no es tracta en profunditat a les assignatures BD I ni a BD II, per la qual cosa no entraré en profunditat en alguns temes d'aquesta etapa de disseny (formes normals, rendiment, disseny virtual, etc.) per quedar fora de l'abast d'aquest projecte.

2.4.4 Presentació plataforma Oracle

Fa vint-i-cinc anys, Larry Ellison va veure una oportunitat de negoci quan es va trobar amb la descripció d'un prototip de treball per a una base de dades relacional i va descobrir que cap empresa s'havia compromès a comercialitzar la tecnologia. Ellison i els seus socis, Bod Miner i Ed Oates, se'n van adonar de que existia un gran potencial de negocis en el model de la base de dades relacional, però no eren conscients que aquesta tecnologia canviaria la indústria de la informàtica comercial per sempre.

A l'actualitat, Oracle es troba pràcticament a totes les indústries del món i a les oficines de 98 de les 100 empreses Fortune 100. Oracle és la primera companyia de programari que desenvolupa i implementa programari per a empreses 100 % Internet a través de tota la seva línea de productes: base de dades, aplicacions comercials i eines de desenvolupament d'aplicacions i suport de decisions. Oracle és el proveïdor mundial líder de programari per a l'administració d'informació, i la segona empresa de programari independent més gran del món.

L'SGBD Oracle va ser dissenyat per suportar grans quantitats d'informació, a més a més per admetre connexions concurrents de multitud d'usuaris (entorns multi-usuari) cap a les mateixes dades.

Oracle aporta un SGBD que estarà ubicat en un maquinari específic i sota un sistema operatiu determinat. L'elecció de l'entorn de treball (maquinari, SO, i tipologia de l'estructura client/servidor) serà una decisió que haurà d'estar d'acord amb les necessitats del sistema d'informació. Oracle pot ser executar en ordinadors personals (PC), mainframes i computadors amb processament paral·lel massiu. Disposa de 17 idiomes, i està suportat per més de 80 arquitectures de maquinari i programari diferent sense tenir la necessitat de canviar cap línia de codi. Això és degut a que més del 80% dels codis interns d'Oracle son iguals a totes les plataformes.

Les principals funcionalitats aportades pel SGBD Oracle són:

- Suport i tractament d'una gran quantitat de dades (GBytes)
- Suport d'una gran quantitat d'usuaris concurrents
- Seguretat d'accés a les dades, restringint dit accés segons les necessitats de cada usuari
- Integritat referencial a la seva estructura de base de dades
- Connectivitat entre les aplicacions dels clients i el servidor de dades Oracle (client/servidor)
- Connectivitat entre bases de dades remotes
- Portabilitat
- Compatibilitat

2.4.5 Implementació de les taules, índexs i seqüències

Els scripts de creació de taules, índexs i seqüències han estat provats en una instància del SGBD Oracle 9i Enterprise Edition i es poden executar en una sessió de Oracle SQL*Plus i s'executen sense errors.

Tots els objectes de la base de dades s'han creat sense especificar un esquema concret, es responsabilitat del DBA (Administrador de la BD) decidir en quin esquema de la seva instal·lació ho vol implementar.

El DBA tindrà les següents responsabilitats, les quals no han estat implementades per aquest producte per quedar fora del seu abast:

- Instal·lació de nous components del programari
- Seguretat del sistema
- Monitorització del rendiment del sistema
- Còpies de seguretat
- Prevenció de riscos

A continuació podem veure els scripts complerts de creació de les taules de la base de dades:

```
CREATE TABLE ciutat(
      codiCiutat VARCHAR2(10),
      nomCiutat VARCHAR2(20) NOT NULL,
      paisCiutat VARCHAR2(20) NOT NULL,
      situacioCiutat CHAR(1) NOT NULL,
PRIMARY KEY(codiCiutat)
);

CREATE TABLE centre(
      codiCiutat VARCHAR2(10),
      codiCentre VARCHAR2(10),
      nomCentre VARCHAR2(20) NOT NULL,
      adreçaCentre VARCHAR2(20) NOT NULL,
      situacioCentre CHAR(1) NOT NULL,
PRIMARY KEY(codiCiutat, codiCentre),
FOREIGN KEY (codiCiutat) REFERENCES ciutat
);

CREATE TABLE departament(
      codiDepartament VARCHAR2(10),
      nomDepartament VARCHAR2(20) NOT NULL,
      codiCiutat VARCHAR2(10),
      codiCentre VARCHAR2(10),
      situacioDepartament CHAR(1) NOT NULL,
PRIMARY KEY(codiDepartament),
FOREIGN KEY (codiCiutat, codiCentre) REFERENCES centre
);

CREATE TABLE tipus_empleat(
      codiTipusEmpleat VARCHAR2(10),
      funcioEmpleat VARCHAR2(20) NOT NULL,
      situacioTipusEmpleat CHAR(1) NOT NULL,
PRIMARY KEY(codiTipusEmpleat)
);
```

```
CREATE TABLE empleat(
      codiEmpleat VARCHAR2(10),
      nomEmpleat VARCHAR2(20) NOT NULL,
      cognomsEmpleat VARCHAR2(20) NOT NULL,
      dniEmpleat NUMBER(8) NOT NULL,
      costEmpleat NUMBER(5,2) NOT NULL,
      codiDepartament VARCHAR2(10),
      codiTipusEmpleat VARCHAR2(10),
      situacioEmpleat CHAR(1) NOT NULL,
PRIMARY KEY(codiEmpleat),
FOREIGN KEY (codiDepartament) REFERENCES departament,
FOREIGN KEY (codiTipusEmpleat)REFERENCES tipus_empleat
);

CREATE TABLE registre_entrada(
      codiRegistre VARCHAR2(10),
      dataRegistre DATE NOT NULL,
      tipusRegistre CHAR(1) NOT NULL,
      codiEmpleat VARCHAR2(10),
      situacioRegistreEntrada CHAR(1) NOT NULL,
PRIMARY KEY(codiRegistre),
FOREIGN KEY (codiEmpleat) REFERENCES empleat
);

CREATE TABLE client(
      codiClient VARCHAR2(10),
      nomClient VARCHAR2(20) NOT NULL,
      adreçaClient VARCHAR2(20) NOT NULL,
      situacioClient CHAR(1) NOT NULL,
PRIMARY KEY(codiClient)
);

CREATE TABLE projecte(
      codiProjecte VARCHAR2(10),
      nomProjecte VARCHAR2(20) NOT NULL,
      dataOberturaProjecte DATE NOT NULL,
      pressupostProjecte NUMBER(12,2) NOT NULL,
      codiCapProjecte VARCHAR2(10),
      codiClient VARCHAR2(10),
      situacioProjecte CHAR(1) NOT NULL,
PRIMARY KEY(codiProjecte),
FOREIGN KEY (codiCapProjecte) REFERENCES empleat(codiEmpleat),
FOREIGN KEY (codiClient) REFERENCES client
);

CREATE TABLE participa(
      codiProjecte VARCHAR2(10),
      codiEmpleat VARCHAR2(10),
      situacioParticipa CHAR(1) NOT NULL,
PRIMARY KEY(codiProjecte, codiEmpleat),
FOREIGN KEY (codiProjecte) REFERENCES projecte,
FOREIGN KEY (codiEmpleat) REFERENCES empleat
);

CREATE TABLE tipus_despesa(
      codiTipusDespesa VARCHAR2(10),
      nomCodiDespesa VARCHAR2(20) NOT NULL,
      situacioTipusDespesa CHAR(1) NOT NULL,
PRIMARY KEY(codiTipusDespesa)
);
```

```
CREATE TABLE despesa(
      codiDespesa VARCHAR2(10),
      importDespesa NUMBER(8,2) NOT NULL,
      codiProjecte VARCHAR2(10),
      codiEmpleat VARCHAR2(10),
      codiTipusDespesa VARCHAR2(10),
      situacioDespesa CHAR(1) NOT NULL,
PRIMARY KEY(codiDespesa),
FOREIGN KEY (codiProjecte, codiEmpleat) REFERENCES participa,
FOREIGN KEY (codiTipusDespesa) REFERENCES tipus_despesa
);

CREATE TABLE permis(
      codiTipusEmpleat VARCHAR2(10),
      codiTipusDespesa VARCHAR2(10),
      situacioPermis CHAR(1) NOT NULL,
PRIMARY KEY(codiTipusEmpleat, codiTipusDespesa),
FOREIGN KEY (codiTipusEmpleat) REFERENCES tipus_empleat,
FOREIGN KEY (codiTipusDespesa) REFERENCES tipus_despesa
);

CREATE TABLE activitat(
      codiActivitat VARCHAR2(10),
      dataIniciActivitat DATE NOT NULL,
      dataFiActivitat DATE,
      codiProjecte VARCHAR2(10),
      codiEmpleat VARCHAR2(10),
      situacioActivitat CHAR(1) NOT NULL,
PRIMARY KEY(codiActivitat),
FOREIGN KEY (codiProjecte, codiEmpleat) REFERENCES participa
);

CREATE TABLE traces(
      usuari VARCHAR2(20),
      data DATE,
      programa VARCHAR2(30),
      traça VARCHAR2(50),
      miss VARCHAR2(100),
      err_num NUMBER,
      err_msg VARCHAR2(100),
      param_in VARCHAR2 (300),
      param_out VARCHAR2 (300)
);
```

Per a la implementació d'aquesta base de dades s'han pres els següents criteris:

- Tots els camps són obligatoris exceptuant la data de finalització d'una activitat.
- Totes les entitats tenen com a clau un codi de fins a 10 caràcters de longitud.
- Tots els noms tenen fins a 20 caràcters de longitud.
- Els camps de situació i de tipologia són d'una longitud fixa d'1 caràcter.
- El cost per hora d'un empleat pot ser fins a 999,99 unitats monetàries
- El pressupost d'un projecte pot ser fins a 9999999999,99 unitats monetàries
- Es poden imputar despeses de fins a 999999,99 unitats monetàries

Els índexs s'han creat amb clau única per afavorir els accessos a les taules i també per aconseguir que no puguin haver repetits en alguns camps (els camps de "nom"). Aquesta restricció UNIQUE també es podria haver ficat als scripts de les taules, però com que aquest criteri no està específicament declarat a les especificacions, crec que és més flexible fer-ho d'aquesta manera ja que en cas de que es vulgui inactivar sempre es pot tornar a crear l'índex de manera més fàcil sense la restricció UNIQUE.

A continuació podem veure els scripts complerts de creació dels índexs de la base de dades:

```
-- Pot haver el mateix nom de ciutat en un país diferent
CREATE UNIQUE INDEX nomCiutatInd
      ON ciutat(nomCiutat, paisCiutat);

-- Pot haver el mateix nom de centre a una ciutat diferent
CREATE UNIQUE INDEX nomCentreInd
      ON centre(nomCentre, codiCiutat);

-- Pot haver el mateix nom de departament a un centre diferent
CREATE UNIQUE INDEX nomDepartamentInd
      ON departament(nomDepartament, codiCiutat, codiCentre);

-- No pot haver el mateix nom de funció de tipus empleat
CREATE UNIQUE INDEX funcioEmpleatInd
      ON tipus_empleat(funcioEmpleat);

-- No pot haver el mateix nom de codi de tipus despesa
CREATE UNIQUE INDEX nomCodiDespesaInd
      ON tipus_despesa(nomCodiDespesa);
-- No pot haver el mateix nom de client
CREATE UNIQUE INDEX nomClientInd
      ON client(nomClient);

-- No pot haver el mateix dni d'un empleat al mateix departament
-- Com que al realitzar un canvi de departament d'un empleat, el
-- treballador es "desactiva" pot ser que estigui la mateixa persona
-- en situació inactiva en un altre departament(on treballava abans)
CREATE UNIQUE INDEX dniEmpleatInd
      ON empleat(dniEmpleat, codiDepartament);

-- No pot haver el mateix nom de projecte
CREATE UNIQUE INDEX nomProjecteInd
      ON projecte(nomProjecte);
```

Els procediments informaran automàticament el codi dels nous registres fent servir les següents seqüències:

```
CREATE SEQUENCE ciutat_seq;
CREATE SEQUENCE centre_seq;
CREATE SEQUENCE departament_seq;
CREATE SEQUENCE tipus_empleat_seq;
CREATE SEQUENCE empleat_seq;
CREATE SEQUENCE registre_entrada_seq;
CREATE SEQUENCE client_seq;
CREATE SEQUENCE projecte_seq;
CREATE SEQUENCE tipus_despesa_seq;
CREATE SEQUENCE despesa_seq;
CREATE SEQUENCE activitat_seq;
```

2.4.6 Implementació dels procediments emmagatzemats

La tecnologia PL/SQL és una extensió de SQL amb capacitat procedimental. Es un llenguatge de programació de Oracle que combina el poder de manipulació de dades de SQL amb el poder de procés de dades dels llenguatges procedurals.

- Característiques:
    - Capacitat procedural
    - Conté instruccions de :
        - Control de flux
        - Repetició
        - Assignació
    - Maneig robust d'errors
    - Al contrari que SQL, PL/SQL permet agrupar un conjunt d'instruccions i enviar-les al motor PL/SQL com un únic bloc
    - Els blocs PL/SQL poden dividir-se en dos tipus:
        - Blocs anònims
        - Subprogrames emmagatzemats: PROCEDURES, PACKAGES i TRIGGERS.
- PL/SQL suporta:
    - Totes les comandes SQL de manipulació de dades
    - Comandes de control de transaccions
    - Funcions
    - Pseudocolumnes
    - Operadors
- PL/SQL no suporta:
    - Comandes de definició de dades (DDL, ex: CREATE, ALTER, ...)
    - Comandes de control de sessió
    - Comandes de control de sistema
- Avantatges:
    - Suporta SQL, donant la capacitat procedural que aquest no te
    - Augment de la productivitat
    - Millora el rendiment
    - Portabilitat
    - Integració amb Oracle

Un PACKAGE és un grup de procediments i funcions, juntament amb els cursors i variables que usen, emmagatzemats junts a la base de dades per al us continuat com a una uniat.

S'implementen freqüentment per a proveir d'avantatges a les següents àrees:

- Encapsulació de variables i procediments
- Declaració pública i privada de procediments, variables, constants i cursors
- Separació de l'especificació i del cos del PACKAGE
- Millor rendiment

Com ja s'ha comentat al punt *Especificació dels procediments emmagatzemats* s'han creat tres PACKAGE. Per motius d'espai no els puc afegir tots a aquest document així que comentaré els blocs de declaració global de cada paquet (veure últim punt *Annex: Scripts*).

```
CREATE OR REPLACE PACKAGE BODY centre_actions AS
-- Declaració de variables d'ús general per tots els subprogrames
10 trace VARCHAR2(50);
20 err_num NUMBER;
30 err_msg VARCHAR2(100);
40 codi_dummy VARCHAR2(10);
50 num_dummy NUMBER;
60 NO_TROBAT EXCEPTION;
70 DEPENDENCIA EXCEPTION;
80 ACTIU CONSTANT CHAR(1) := 'A';
90 BAIXA CONSTANT CHAR(1) := 'B';
(…)
```

10: Variable per controlar el mecanisme de traces
20: Variable per emmagatzemar el SQLCODE
30: Variable per emmagatzemar el SQLERRM
40: Variable "dummy" no serveix per res, únicament per fer saltar excepcions de no trobat o similars
50: Variable "dummy" no serveix per res, per fer sumatoris que no facin saltar excepcions
60: Excepció d'usuari per llençar quan no es trobi una referència necessària
70: Excepció d'usuari per llençar quan hi hagi dependències entre entitats que es volen modificar
80: Constat general de situació activa
90: Constat general de situació baixa

```
CREATE OR REPLACE PACKAGE BODY empleat_actions AS
-- Declaració de variables d'ús general per tots els subprogrames
10  trace VARCHAR2(50);
20  err_num NUMBER;
30  err_msg VARCHAR2(100);
40  codi_dummy VARCHAR2(10);
50  num_dummy NUMBER;
60  NO_TROBAT EXCEPTION;
70  DEPENDENCIA EXCEPTION;
80  ACTIU CONSTANT CHAR(1) := 'A';
90  BAIXA CONSTANT CHAR(1) := 'B';
100 ENTRADA CONSTANT CHAR(1) := 'E';
110 SORTIDA CONSTANT CHAR(1) := 'S';
(…)
```

100: Constat general de tipus entrada
110: Constat general de tipus sortida

```
CREATE OR REPLACE PACKAGE BODY projecte_actions AS
-- Declaració de variables d'ús general per tots els subprogrames
10  trace VARCHAR2(50);
20  err_num NUMBER;
30  err_msg VARCHAR2(100);
40  codi_dummy VARCHAR2(10);
50  num_dummy NUMBER;
60  NO_TROBAT EXCEPTION;
70  DEPENDENCIA EXCEPTION;
80  ACTIU CONSTANT CHAR(1) := 'A';
90  BAIXA CONSTANT CHAR(1) := 'B';
100 TANCAT CONSTANT CHAR(1) := 'B';
110 CAP CONSTANT Tipus_Empleat.funcioEmpleat%TYPE := 'CapProjecte';
(…)
```

100: Constat general de situació tancat
110: Constant general de tipus d'empleat cap de projecte

3 Jocs de proves

3.1 Simulació d'un any de funcionament del sistema

S'ha desenvolupat un bloc PL/SQL anònim per intentar simular un any de funcionament de la base de dades. Aquesta simulació s'ha desenvolupat fent servir números aleatoris allà on ha estat possible per intentar simular al màxim possible a la realitat.

Aquest procés consta dels següents passos:

- Alta de 5 ciutats
- Alta de 10 centres
    - 2 centres assignats a cadascuna de les 5 ciutats
- Alta de 20 departaments
    - 2 departaments assignats a cadascun dels 10 centres
- Alta de 5 tipus d'empleats
- Alta de 100 empleats
    - 5 empleats per cada departament
    - 1 empleat de cada tipus per cada departament
- Alta de 44000 registres d'entrada
    - 220 registres d'entrada de tipus entrada per cada empleat
    - 220 registres d'entrada de tipus sortida per cada empleat
- Alta de 10 clients
- Alta de 200 projectes
    - 20 projectes per cada client
    - 10 projectes per cada cap de projecte
- Alta de 1000 participacions
    - 5 participacions per cada projecte
- Alta de 5 tipus de despeses
- Alta de 5 permisos
    - 1 permís a 1 tipus de despesa per cada tipus d'empleat
- Alta de 1000 despeses
    - 1 despeses per cada participació
    - 1000 despeses per cada tipus de despesa
- Alta de 2000 activitats
    - 2 activitats per cada participació

3.2 Proves unitàries

S'han realitzat proves amb tots els procediments. Per cada procediment es faran dos tipus de comprovacions:

- Procediment retorna OK
    - S'introduiran tots els paràmetres correctament i s'acompliran les dependències entre entitats convenientment
    - Es comprovarà que l'operació demanada (alta, baixa, etc.) s'ha realitzat correctament revisant la base de dades
- Procediment retorna ERROR
    - Es forçaran els errors monitoritzats per excepcions més importants de cada procediment
    - Es comprovarà que a la taula traces s'ha donat d'alta un registre amb l'error que es volia provocar

S'annexa script de realització de proves de l'entitat Ciutat .

4 Glossari

- **Administrador de BD**
    - Tipus d'usuari especial que realitza funcions centralitzades d'administració i control de la BD que asseguren que l'explotació de la BD és correcta
- **Atribut**
    - Propietat d'una entitat
- **Base de dades**
    - Conjunt estructurat de dades que representa entitats i les seves interrelacions. La representació serà única, integrada, malgrat que ha de permetre utilitzacions diverses i simultànies
- **Camp**
    - Representació del valor d'un atribut
- **Clau**
    - Atribut o conjunt d'atributs que permet identificar els objectes (distingir-los els uns dels altres)
- **Dada**
    - Nom que rep la informació en el món de les representacions informàtiques
- **Disseny conceptual**
    - Etapa del disseny d'una base de dades que obté una estructura de la informació de la futura BD independent de la tecnologia que es vol emprar
- **Disseny físic**
    - Etapa del disseny d'una base de dades que transforma l'estructura obtinguda a l'etapa del disseny lògic amb l'objectiu d'aconseguir una major eficiència i que, a més, la completa amb aspectes d'implementació física que dependran de l'SGBD que s'ha d'utilitzar
- **Disseny lògic**
    - Etapa del disseny d'una base de dades que parteix del resultat del disseny conceptual i el transforma de manera que s'adapti al model de l 'SGBD amb el qual es desitja implementar la base de dades
- **Entitat**
    - Conceptualització d'un objecte del món real. El concepte del qual una entitat és instància s'anomena també tipus d'entitat
- **Fitxer**
    - Conjunt de registres relatius a un mateix tipus d'entitat
- **Procediment emmagatzemat**
    - Acció o funció definida per un usuari que proporciona un determinat servei. Un cop ha estat creat, es guarda en la BD i passa a ser tractat com un objecte més d'aquesta. L'execució d'un procediment pot retornar cap, un o més valors.
- **Registre**
    - Conjunt de dades relatives a un objecte

- **Sistema de gestió de BD (SGBD)**
    - Programari que gestiona i controla la BD. Les seves principals funcions són facilitar-ne la utilització a molts usuaris simultanis i de tipus diferents, independitzar l'usuari del món físic i mantenir la integritat de les dades.
- **Structured Query Language (SQL)**
    - Llenguatge especialitzat en la descripció (DDL) i la utilització (DML) de BD relacionals. Creat per IBM al final dels anys setanta i estandarditzat per ANSI-ISO l'any 1985 (l'últim estàndard de SQL és de 1992). Actualment és utilitzat pràcticament per tots els SGBD del mercat.
- **Transacció**
    - Conjunt d'operacions (de BD) que volem que s'executin com un tot (totes o cap) i aïlladament (sense interferències) d'altres conjunts d'operacions que s'executin concurrentment

5 Programari utilitzat

5.1 Suport Oracle

- Oracle Enterprise Manager Console
  - Administració SGBD Oracle
- Oracle SQL*Plus
  - Execució dels scripts, consultes, etc.

5.2 Suport ofimàtica

- Microsoft Word 2003
  - Edició de la memòria
- Microsoft Power Point 2003
  - Edició de la presentació
- Microsoft Access 2003
  - Suport per comprovar resultats proves (fent servir ODBC Oracle)
- Microsoft Visio 2003
  - Model ER
- UltraEdit -32 v10.20c
  - Edició scripts

6 Bibliografia

6.1 Llibres de text UOC

- Jaume Sistac Planas (coordinador); Rafael Camps; Dolors Costal; Franch Gutiérrez; Xavier Franch; Carme Martín. ***"Base de dades I"*** Barcelona UOC, 2002.
- Jaume Sistac Planas (coordinador); Rafael Camps; Dolors Costal; Pablo Costa; M.Elena Rodríguez; Ramon Segret; Josep Maria Marco; Toni Urpí. ***"Base de dades II"*** Barcelona UOC, 2004.
- Xavier Burgués Illa; Fatos Xhafa; Xavier Franch Gutiérrez. ***"Estructura de la informació"*** Barcelona. UOC, 2001.
- Jaume Sistac Planas (coordinador); Albert Abelló Gamazo; Blai Cabré i Segarra; Marta Oliva Solé; Santi Ortego Carazo; Elena Rodríguez González; Ramon Segret i Sala. **"Sistemes de gestió de bases de dades"** Barcelona UOC, 2001

6.2 Altra documentació

- Documentació del curs de SB Consultors: **Curs de programació ORACLE**
- Documentació del curs de SB Consultors: **Curs de PL/SQL**
- Documentació del curs de SB Consultors: **Curs d'Administració ORACLE**
- **Oracle9i Online Documentation Library**

6.3 Portals d'Internet

- http://www.dbasupport.com
  - Portal amb recursos per DBA
- http://www.orape.net
  - Portal amb recursos Oracle
- http://www.programatium.com
  - Portal amb recursos per programadors
- http://www.programacion.net
  - Portal amb recursos per programadors
- http://www.oracle.com
  - Portal oficial Oracle

7 Annex: Scripts

- Script de creació de les taules, índexs i seqüències:
  - Versió executable text pla: CREA.SQL
  - Versió llegible: CREA.PDF
- Script amb les especificacions dels tres PACKAGES:
  - Versió executable text pla: especificacions.SQL
  - Versió llegible: especificacions.PDF
- Script amb el PACKAGE centre_actions:
  - Versió executable text pla: centre_actions.SQL
  - Versió llegible: centre_actions.PDF
- Script amb el PACKAGE empleat_actions:
  - Versió executable text pla: empleat_actions.SQL
  - Versió llegible: empleat_actions.PDF
- Script amb el PACKAGE projecte_actions:
  - Versió executable text pla: projecte_actions.SQL
  - Versió llegible: projecte_actions.PDF
- Script amb la càrrega de simulació d'un any d'ús:
  - Versió executable text pla: CARREGA.SQL
  - Versió llegible: CARREGA.PDF
- Script amb el joc de proves dels procediments:
  - Versió executable text pla: PROVES.SQL
  - Versió llegible: PROVES.PDF
- Script amb l'esborrat de totes les taules, índexs i seqüències:
  - Versió executable text pla: ESBORRA.SQL
  - Versió llegible: ESBORRA.PDF
- Script amb l'esborrat previ, i creació de tots els objectes:
  - Versió executable text pla: COMPILA.SQL