Disseny i implementació de la base de dades d'un sistema de control energètic

Joan Gatius Lendinez

ETIG / ETIS

Jordi Ferrer Duran

Data Lliurament

14/01/2013

TFC. Arèa de base de dades.

El treball de fi de carrera és una assignatura que està pensada per a realitzar un treball de síntesi dels coneixements adquirits en altres assignatures de la carrera i que requereixi posar-los en pràctica conjuntament en un treball concret.

Els objectius d'aquest projecte han set dos: presentar una solució per als requeriments sol·licitats i documentar el procediment utilitzat per a elaborar-la.

El projecte s'engloba en la categoria de bases de dades, i la tasca consistirà en la elaboració d'un model relacional, així com els procediments necessaris per a gestionar-lo.

La solució que es proposa s'estructura en tres apartats:

- El model relacional elaborat a partir dels requeriments sol·licitats.
- L'estructura de les dades, així com les vistes, els procediments, disparadors i dades d'exemple en format SQL[1].
- Documentació relativa a la generació dels dos apartats anteriors.

Es tindrà especial cura en la descripció dels procediments, com les decisions preses per a donar a aquest document un valor afegit: a banda d'oferir el model generat s'ofereix una metodologia de treball per a poder generar aquest model, aplicable a altres models que es vulguin generar.

[1] Structured Query Language.

Motivacions.

Actualment, les TIC[2] guanyen importància dia a dia tant en l'àmbit domèstic com en l'àmbit empresarial. Gràcies a xarxes com Internet, és possible la comunicació i l'accés a molta informació d'una manera immediata, però sovint poc estructurada.

Des del punt de vista empresarial, aquestes noves tecnologies porten a les empreses a innovar i a obrir-se a aquests nous entorns, facilitant així l'accés als serveis i productes que ofereixen als clients.

Serà necessari organitzar i classificar tota aquesta informació d'una manera estructurada, coherent i normalitzada. Tot SI3 necessita un model que li permeti emmagatzemar, manipular i consultar les dades que gestiona. Les bases de dades relacionals, tot i que ja fa força temps que existeixen, són les eines més utilitzades a l'hora d'emmagatzemar i processar informació.

Com en tot sistema evolutiu, les bases de dades s'adeqüen als nous requeriments que l'evolució de les TIC sol·liciten. Des de les més específiques, com la incorporació de camps multimèdia a les taules de la base de dades o la possibilitat d'utilitzar elements propis de la POO4 com l'herència o el control d'excepcions, fins a les més obertes com les bases de dades distribuïdes.

Programari.

Per a l'elaboració d'aquest projecte s'han emprat diferents eines:

- Oracle SQLDeveloper.
- Oracle DataModeler.
- Oracle Database 11g Express Edition.
- Microsoft Office Word 2010.

[2] Tecnologies de la Informació i de la comunicació.
[3] Sistema d'Informació. Extret de l'assignatura: Gestió d'organitzacions i projectes informàtics.
[4] Programació Orientada a Objecte.

Taula de continguts.

El model entitat – relació.

Seguint els procediments normalitzats, es dóna forma al model Entitat – Relació que ens servirà de punt de partida per a donar una solució el més optimitzada possible, que satisfaci els requeriments sol·licitats per aquest control de sistema energètic i que analitzarem punt per punt, descrivint les decisions preses i les accions efectuades

Primer: Identificació de les entitats.

Farem un estudi inicial dels requeriments:

R1.

> *El model ha de permetre guardar totes les dades associades a un comptador, aquestes serien com a mínim:*

El nostre objectiu primer és detectar les entitats (taules) que es desprèn dels requeriments. Llavors, està clar que la taula associada a l'anterior cita és **comptador**, de la que haurem de detectar les propietats que calgui emmagatzemar (els camps) i els tipus de dades adients.

> Codi de *contracte*, per a la seva identificació ràpida per part dels clients (no hi ha un màxim establert en el nombre de contractes que pot tenir un client).

D'aquest segon extracte dels requeriments podem extreure diverses conclusions:

- El codi de contracte, associat al comptador, no pot formar part com a propietat de comptador ja que per deducció podem pensar que un comptador pot formar part de més d'un contracte (per la venda d'un habitatge, per exemple). Per aquesta raó no podrem tenir en un sol registre (fila de la taula) les dues o més referències als possibles contractes que puguin existir, ja que si el codi de contracte és una columna de la taula comptador només tindrem una sola casella per posar més d'un valor.

- Els **clients**, que també formen part o estan vinculats als comptadors, tenen el mateix problema que els contractes, més clarament ja que està estipulada la possibilitat de que un client pugui tenir molts contractes (aquest aclariment ens diu que no hi ha un màxim establert en el nombre de contractes. D'aquí només podem extreure el fet que en puguin haver més d'un però no haurem de posar

limitacions en el seu nombre, ja que això queda fora de l'àmbit d'actuació del model Boice-Codd). A més, la vinculació del client amb el comptador no és directa, ja que client està vinculat a **contracte**, que subsidiàriament està vinculat a comptador.

- **Client**, per simple deducció, ja és entitat pròpia. Donarà moltes avantatges al sistema el fet de tenir informació referent als clients, quanta més, millor i resol les limitacions esmentades.

- **Contracte** serà el vincle entre **clients** i **comptadors** i també el considerarem entitat, ja que té propietats específiques pròpies molt interessants de tenir en el sistema (el client, el comptador, la data, el tipus de pagament, etc.).

La potència contractada pels clients mesurada en Kilowats (KW).

Tot i que la potència està referenciada des del punt de vista del comptador, KW és una propietat de contracte, ja que per diferents contractes pot donar-se diferents potències contractades.

L'adreça complerta on es troba físicament el comptador: adreça, localitat, país.

Aquestes sí són propietats del comptador: Adreca (amb c i no ç. Això és un nom de camp que segurament s'utilitzarà en un motor de BD del mercat normalment americans, que no tenen aquest caràcter), número (separat de l'adreça. Utilitzarem el número 0 per quan no hi ha número), pis i porta, bloc, codi postal i país. També podem afegir les coordenades (latitud i longitud) per situar-les en un mapa, molt útil per a GEO localitzacions com ara Google Maps o GPS.

La lectura de consum del comptador, és a dir, el numero de kilowatts per hora (KHh) absoluts consumits des de la instal·lació del comptador.

Aquest és un valor únic per a cada comptador que s'anirà actualitzant per a cada nova lectura de consum. Anirà a la taula comptador com a propietat.

El consum elèctric mensual de cada comptador mesurat en KWh (per simplificar s'avaluaran les dades de consum mensualment tot mantenint l'històric de consum mensual de cada comptador).

Serà necessari guardar informació de cada més per a tenir disponible un històric.

Necessitarem l'any, el mes i el consum corresponent. Donat que necessitarem una relació de dades per a cada comptador, crearem una entitat nova **historicC**, amb les propietats tres propietats i que vincularem a la taula comptador com veurem més endavant.

> La o les centraletes de distribució a les que està connectat aquest comptador; tenint en compte que un comptador connectat a dues centraletes de distribució assegura que, davant la caiguda d'una de les centraletes, el servei es continua garantint. Les centraletes de distribució permeten un màxim d' energia elèctrica (KWh) que poden subministrar i s'identifiquen per l'adreça on es troben situades.

Les centraletes de distribució **(centraleta)** seran una nova entitat per al nostre model. De l'exposició anterior es desprèn que tindrà dues propietats: adreca i màxim d'energia, en KWh.

La relació que es descriu en aquest apartat entre les centraletes i els comptadors la definirem més endavant.

> El model del comptador, detallant-ne model concret , empresa de fabricació i any de fabricació, com a mínim.

Necessitarem emmagatzemar informació sobre el fabricant, el model, i l'any entre d'altres. Totes aquestes propietats es podrien afegir a la taula comptador però es pot donar que dos o més comptadors tinguin el mateix fabricant i siguin del mateix model. Això ens porta a separar les dades referents a la fabricació i crear una entitat nova **modelC**, que inclourà aquestes dades i que vincularem al comptador, com veurem més endavant. Tenim la possibilitat de detectar més informació útil respecte del comptador que es podria afegir a la taula, ja que les propietats comentades són un mínim necessari que pot ampliar-se.

> Data de la darrera lectura efectuada.

La data de la darrera lectura és bàsicament un valor que anirà actualitzant-se per a cada lectura realitzada. Sent doncs un sol valor que fa referència a un comptador podem posar-la com una propietat de **comptador**.

R2.

> El model haurà de permetre guardar les dades sobre les centrals de producció, concretament hauria de contemplar:

Obtenim una nova entitat **centralPr**, les propietats i les entitats relacionades les podem extreure de les següents declaracions.

> El codi de la central, que actuarà com a identificador únic de l'equipament.

El codi de la central que serà l'identificador (clau principal) de la taula **centralPr**. Cap altra propietat actuarà de clau principal combinada, tal i com se'ns demana.

> L'adreça complerta de la central: adreça, localitat, país.

Una central de producció tindrà una adreça més simple que un comptador o un client, ja que no caldrà pis, porta ni bloc. Entenem que una central estarà situada en un terreny més o menys gran i ubicat en un polígon industrial o espai semblant. Posarem adreça (recordem de substituir la 'ç' i els accents entre d'altres en els camps), número (que podria ser número de parcel·la o finca, si és un polígon industrial), codi postal, localitat, país i per a GEO localització latitud i longitud.

> Les centraletes de distribució alimentades des d'aquesta central.

Amb aquesta informació obtenim la relació existent entre les taules **centraleta** i **centralPr**.

> Històric de l'energia elèctrica produïda mensualment mesurat en Kilowatts hora (KWh) entès com l'energia absoluta produïda per la central.

Si ha d'haver un històric està clar que els diferents valors que compondran aquest històric no 'cabran' a la taula **centralPr**, ja que només hi hauria una casella per columna per guardar un valor per a cada central. Crearem una entitat nova **produccioMes**, on guardarem l'any, el mes i l'energia produïda, a part de la relació que s'haurà d'establir amb la taula **centralPr**, que comentarem més endavant.

> Energia màxima produïda (KWh).

Aquest valor, en canvi, sí es pot afegir a la taula **centralPr**, ja que només n'hi haurà un per cada central, amb el que el podrem posar a la vora del seu nom i d'altres dades referents a la central.

> Data de la darrera inspecció tècnica.

De la mateixa manera que ens ha passat amb *energia màxima produïda*, la data de la darrera inspecció també es una dada que es pot afegir a la taula **centralPr**, ja que només n'hi haurà una per a cada central. Utilitzarem format de data per a que el motor de base de dades pugui utilitzar-la com a tal.

> Segons el tipus de central també necessitarem les dades següents:

Aquí podem entendre que es demana que especifiquem diferents dades per diferents tipus de centrals. En aquest apartat podem utilitzar el principi *generalització – especificació (especialització)*, i separar per una banda les propietats que són comunes a totes les centrals (com ara l'adreça, la data de l'última inspecció, entre d'altres) de les específiques de cada tipus de central.

Per a que això sigui possible es crearan entitats noves (taules noves) per a cada un dels subtipus de central i s'afegiran en cada una d'elles les propietats específiques que necessiti. Posteriorment es crearà la relació entre la taula comuna (generalització) amb els subtipus (especificació) utilitzant el tipus de relació 1:1, que ampliem més endavant.

Concretament, els subtipus són aquests:

- Nuclear: Energia mínima necessària per funcionar i els kilograms de rebuig radioactiu generat.
- Tèrmica: Kilograms d'emissions de CO2.
- Carbó: Kilograms d'emissions de CO2.
- Eòlica: Nombre de molins de vent instal·lats.
- Solar: Nombre de panells necessaris pel seu funcionament.

Amb les corresponents propietats específiques.

R3.

> El sistema ha de contemplar la gestió de les línies de comunicació, enteses com a tals les línies que connecten les centrals producció a les de distribució, una línia pot alimentar a varies centrals de distribució, però només connecten una central de producció. Aquestes línies permeten el transport d'una capacitat màxima i tenen un codi numèric que les identifica.

Definirem una taula (entitat) nova **liniaCom**, amb un codi numèric i una capacitat màxima com a propietats. Definirem les relacions més endavant.

R4.

> El model ha de permetre emmagatzemar la informació següent dels clients: DNI o NIF que identificarà al client sigui aquest una persona física o una empresa, l'adreça i la resta de dades que es considerin necessàries.

L'entitat **client**, que ja havíem detectat anteriorment, inclou especificacions. Com ha passat amb les centrals de producció en quant als diferents tipus de centrals que inclouen, els clients tenen una part comuna i una d'específica per a cada tipus de **client**, persona física i persona jurídica.

Cal identificar les propietat que són comunes, com ara el NIF (CIF per a les empreses que té el mateix format). Tot i que algunes dades, com ara l'adreça, tenen parts comuns per tots dos clients, optem per compartir només el NIF (CIF), i personalitzem la resta de les dades per a cada tipus de client. Una persona física tindrà nom i cognoms, data de naixement i d'altres dades que els clients jurídics no tindran (tenint en compte que dades com la data de naixement podria ser una propietat compartida utilitzant-la com a data d'apertura de l'empresa). És important utilitzar aquest tipus de model (generalització – especialització) per a que els dos grups, tot i tenir força propietats diferents, estiguin englobats en l'àmbit de **client**.

R5.

> El model ha de permetre gestionar un control de les lectures de comptadors efectuades. Caldrà controlar les dates i els valors de les diferents lectures fetes a cada comptador així com la modalitat en què s'han fet. Per aquest darrer aspecte, cal considerar que hi ha comptadors on la lectura es fa telemàticament i d'altres on cal passar-hi de forma presencial.

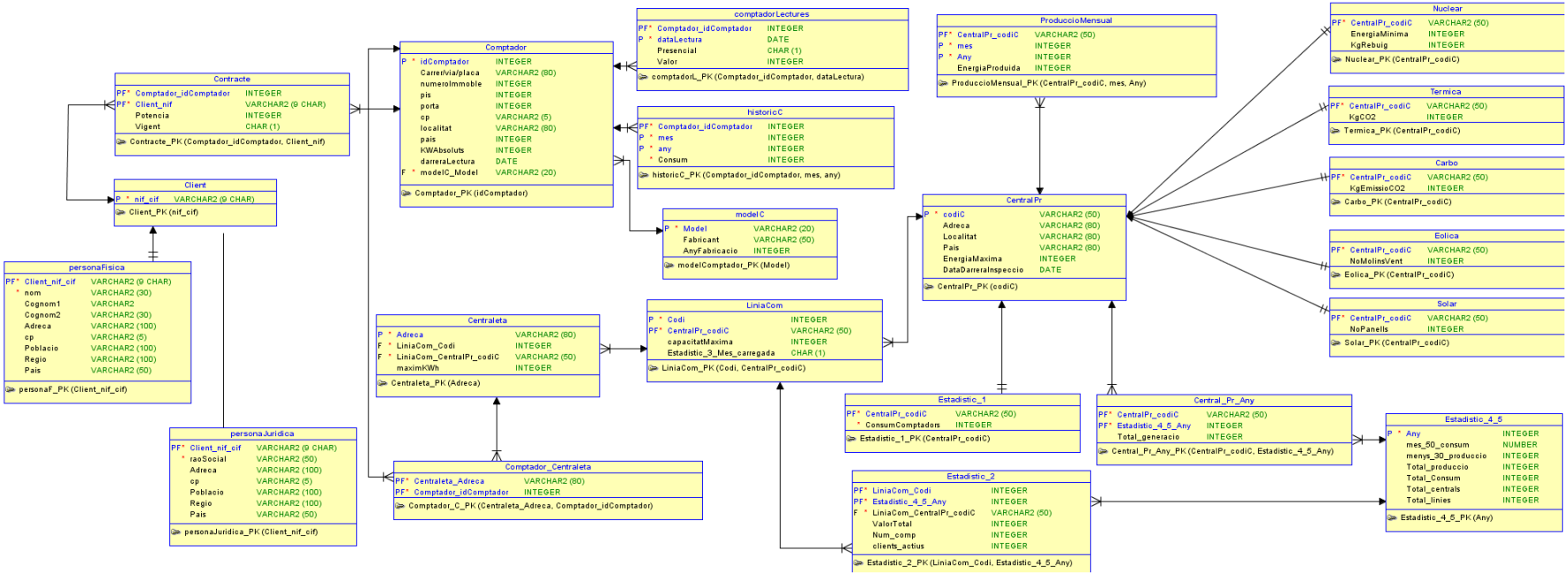
Finalment (en quant a la definició d'entitats i propietats) necessitarem una entitat més **lecturaC** on guardarem la informació que se'ns demana. Això és així donat que per a un comptador hi hauran diverses lectures com podria no haver-ne cap (per al comptadors nous, per exemple) i si ho afegíssim com propietat en la taula **Comptador**, només es podria incloure una lectura i el valor llegit, perdent-se l'anterior dada.

Aquesta entitat nova inclou la data de la lectura, valor i el tipus de lectura (presencial o no), i la vinculació amb la taula **Comptador**.

Arribats aquí ja podem extreure la llista de entitat (taules) i la majoria de les propietats que inclouen necessàries per al nostre model.

Entitats:

Comptador	**nuclear**
Client	**termica**
Contracte	**carbo**
historicC	**eolica**
centraleta	**solar**
modelC	**liniaCom**
centralPr	**lecturaC**
produccioMes	



Segon: Definició dels atributs. Model relacional.

En aquest apartat definirem en detall els atributs que siguin necessaris per a cada entitat així com aquells que tot i no ser imprescindibles puguin millorar el nostre model afegint-hi funcionalitats interessants.

En aquesta secció definirem també el sistema de relació que haurem d'implementar en el nostre model.

Realitzarem totes dues tasques a l'hora ja que estan estretament relacionades. La definició dels atributs donaran a l'entitat les propietats necessàries per la pròpia estructura d'entitat, emmagatzemant tota la informació necessària per a cada element que s'hi afegeixi. La definició del model relacional implicarà afegir propietats de relació a les entitats seguint el model formal que aquí detallarem.[5]

Model_C

Com comentàvem en la secció anterior, la taula model_C inclou dades referents als diferents models que ens podem trobar de comptadors i els seus fabricants. Caldria esmentar al client la possibilitat d'explotar les possibilitats d'aquesta taula ampliant les dades sobre el fabricant, el preu de compra, els costos d'instal·lació entre d'altres. La connexió existent amb els clients, les zones o els consums fa que aquestes dades més detallades puguin ser molt útils.

Nom de camp	Tipus de dades	Caràcters
Model	VARCHAR2	(20)
Fabricant	VARCHAR2	(50)
AnyFabricacio	INTEGER	

Comptador.

Comptador és una entitat clau en el nostre model i una de les que portarà més informació. La localització geogràfica ve definida per l'adreça física, carrer, número, pis i porta. Cal comentar la possibilitat, ja que actualment la tecnologia ho permet, d'emmagatzemar les coordenades geogràfiques de la seva situació, informació molt útil per al gestors de mapes.

L'adreça i el número de porta seran dues propietats independents, dos camps. Podríem pensar que tots dos camps poden funcionar junts però això ens porta a un seguit de

[5] http://www.barrywise.com/2008/01/database-normalization-and-design-techniques/

problemes amb la consistència de les dades. Com a exemple, podríem demanar al sistema quants comptadors trobem a l'avinguda Diagonal de Barcelona. Si les dades estan separades serà molt fàcil agrupar la columna 'carrer' ja que totes les dades d'aquest camp seran iguals per als comptadors de l'avinguda Diagonal. No passarà el mateix si va juntament amb el número perquè llavors no tenim opcions d'agrupament, el número farà que siguin diferents.

Una de les normatives de normalització en un sistema de bases de dades relacionals és l'atomització, el fet de separar les dades en diferents columnes (camps) fins que ja no siguin divisibles (que siguin atòmiques), com comentàvem en l'exemple. Aquest fet és indispensable (però no suficient) per a que la taula estigui en primera forma normal (1FN)[6].

La taula Model_C, que conté dades referents als comptadors, és independent de la taula comptadors per a que el nostre model relacional compleixi amb la segona forma normal (2FN)[7]. Podríem pensar que les propietats de Model_C podrien afegir-se a la taula Comptadors, així quan afegim un comptador juntament amb ell afegim les propietats del model. El model de comptador té dues propietats (fabricant i anyFabricacio), així, afegides a comptadors, ens trobem alguns problemes: tots els comptadors del mateix model tindran aquestes propietats repetides. Si tenim cinc-cents comptadors iguals, aquestes dades es repetiran 500 cops, amb la conseqüent despesa d'espai de disc i de temps per introduir-les.

Per altra banda, és necessari crear una vinculació entre les dues taules. Per a fer aquesta relació serà indispensable definir el tipus de relació entre les tres possibles: de un a un (un model, un comptador), de un a molts (un model, molts comptadors o cap), de molts a molts (molts models, molts comptadors).

Observant els requeriments del client i fins i tot aplicant la lògica, és clar que un model tindrà molts comptadors i no al revés.

Detectada la relació, procedim a crear-la. Els tipus 1 – N implica incloure en la taula N la clau primària de 1, que passarà (en N) a ser clau forana de la taula Model_C. El camp modelC_Model farà aquesta funció.

[6] http://databases.about.com/od/specificproducts/a/firstnormalform.htm
[7] http://databases.about.com/od/specificproducts/a/2nf.htm

Dit d'un altra manera, cada comptador tindrà una referència cap al model al que pertany i no al revés.

Nom de camp	Tipus de dades	Caràcters
idComptador	INTEGER	
Carrer/via/placa	VARCHAR2	(80)
numeroImmoble	INTEGER	
Pis	INTEGER	
Porta	INTEGER	
Cp	VARCHAR2	(5)
Localitat	VARCHAR2	(80)
País	INTEGER	
KWAbsoluts	INTEGER	
darreraLectura	DATE	
modelC_Model	VARCHAR2	(20)

LecturaC

Es requereix, com s'especifica en la documentació facilitada pel client, guardar un històric de les lectures dels comptadors.

Per emmagatzemar aquestes lectures tenim la data en que es realitza, si és presencial i quin és el valor del comptador.

Cal comentar que Comptador_idComptador és clau forana de la taula comptador, ja que aquesta taula és la part N de la relació amb la taula comptador, que és del tipus 1 – N. Aquest camp també actua com a clau principal d'aquesta taula, juntament amb dataLectura. Aquesta clau amb més d'un camp permet repetir alguns dels elements de la combinació de clau però mai totes dues alhora. Ens assegurem així que en una data determinada només es fa una lectura d'un comptador i és única.

Nom de camp	Tipus de dades	Caràcters
Comptador_idComptador	INTEGER	
dataLectura	DATE	
Presencial	CHAR	(1)
Valor	INTEGER	

HistòricC

Per guardar un històric de consum mensual ens assegurem que per a un comptador en un més i un any específic només s'introdueixi un valor, fent que tots tres camps siguin clau principal.

Nom de camp	Tipus de dades	Caràcters
Comptador_idComptador	INTEGER	

mes	INTEGER
any	INTEGER
Consum	INTEGER

Contracte

Un contracte inclourà la referencia cap al client i cap al comptador (cap a les seves respectives taules) mitjançant les claus foranes adients. No estan inclosos en la clau principal per què podria donar-se que un client contracti el mateix comptador dos cops (per haver-se donat de baixa, per exemple). Així, afegim un identificador de contracte que farà de clau principal.

Nom de camp	Tipus de dades	Caràcters
idContracte	INTEGER	
Comptador_idComptador	INTEGER	
Client_nif	VARCHAR2	(9)
Potencia	INTEGER	
Vigent	CHAR	(1)
Data_contracte	DATE	

Client

De clients n'hi han de dos tipus, clients físics i clients jurídics. Tots dos són clients però cada un tindrà peculiaritats (propietats) específiques que l'altre no té.

La relació existent serà del tipus generalització – especialització[8], entenent la generalització com el 'client' i la especialització com 'físics' o 'juridics'. Per interpretar això en taules relacionals crearem relacions 1 – 1 entre la taula client i les taules especialitzades. Els clients mantindran la vinculació entre ells tot i tenir peculiaritats. Les taules 'filles' tindran la mateixa clau principal que la taula mare, que farà tant de clau principal com de clau forana.

Nom de camp	Tipus de dades	Caràcters
Client_nif_cif	VARCHAR2	(9)
Adreca	VARCHAR2	(100)
cp	VARCHAR2	(5)
Poblacio	VARCHAR2	(100)
Regio	VARCHAR2	(100)
Pais	VARCHAR2	(50)

personaFisica

Nom de camp	Tipus de dades	Caràcters
Client_nif_cif	VARCHAR2	(9)
nom	VARCHAR2	(30)

[8] http://ca.wikipedia.org/wiki/Generalització

Cognom1	VARCHAR2	(30)
Cognom2	VARCHAR2	(30)

personaJuridica

Nom de camp	Tipus de dades	Caràcters
Client_nif_cif	VARCHAR2	(9)
raoSocial	VARCHAR2	(50)

CentralPr

Les centrals de producció, al igual que fèiem amb els clients, inclòuen un tipus de relació generalització / especialització. Es tracta de definir propietats específiques per a cada tipus de central. El model tindrà una taula per a cada tipus de central específica, amb la mateixa clau principal que la CentralPr, les dades necessàries pròpies del tipus de central. En la taula general, CentralPr, totes les propietats que afecten a totes les centrals.

Nom de camp	Tipus de dades	Caràcters
codiC	VARCHAR2	(50)
Adreca	VARCHAR2	(80)
Localitat	VARCHAR2	(80)
Pais	VARCHAR2	(80)
EnergiaMaxima	INTEGER	
DataDarreraInspeccio	DATE	

produccioM

Un dels requeriments del client és la capacitat del sistema de recollir l'energia produïda per cada central cada mes. Donat que existiran moltes dades referents a una sola central podem dir que la relació entre aquestes dues taules es 1 – N on 1 és la CentralPr. Tant central com mes i any són clau principal, d'aquesta manera evitem que s'introdueixi dos valors per a la mateixa central el mateix mes del mateix any.

Nom de camp	Tipus de dades	Caràcters
CentralPr_codiC	VARCHAR2	(50)
mes	INTEGER	
Any	INTEGER	
EnergiaProduida	INTEGER	

LiniaCom

Les línies de comunicació tenen una relació 1 – N amb CentralPr, ja que una central pot tenir moltes línies de comunicació.

Nom de camp	Tipus de dades	Caràcters
Codi	INTEGER	

CentralPr_codiC	VARCHAR2	(50)

Centraleta

La relació amb línies de comunicació és de 1 – N, relació que també s'estableix directament amb CentralPr. Emmagatzemarà el màxim de KWh de consum.

Nom de camp	Tipus de dades	Caràcters
Adreca	VARCHAR2	(80)
LiniaCom_Codi	INTEGER	
LiniaCom_CentralPr_codiC	VARCHAR2	(50)
maximKWh	INTEGER	

comtador_centraleta

El tercer tipus de relació N – M[9] es dona entre aquestes dues entitats. Per representar-ho en el model es crea una entitat nova comtador_centraleta que contindrà les claus principals de totes dues taules que, en la taula nova, actuaran de clau principal conjunta, a part de ser claus foranes.

Nom de camp	Tipus de dades	Caràcters
Centraleta_Adreca	VARCHAR2	(80)
Comptador_idComptador	INTEGER	

Log

Es guardaran tots els moviments fets en la base de dades en aquesta taula on es posarà una breu descripció de l'event, la data i la hora.

Estadística

En el nostre model s'inclouen unes entitats estadístiques destinades a realitzar càlculs específics. És un requeriment del client evitar les funcions d'agregació, per estalvi de procés (aquesta base de dades podria emmagatzemar una quantitat d'informació considerable). Aquestes taules s'hauran d'actualitzar en quan s'hi afegeixin nous registres a les taules afectades, per a repartir el procés de càlcul en petits processos que es realitzaran automàticament. Aquestes entitats es comentaran detalladament juntament amb els disparadors que les gestionen.

[9] http://databases.about.com/od/specificproducts/a/3nf.htm

Gestió de la base de dades. SQL i PL-SQL.

Tercer: Consulta de dades.

Un dels objectius de tot model relacional és la capacitat que ens dona el SGBD (Sistema Gestor de Base de Dades)[10] de filtrar, ordenar i / o calcular les dades emmagatzemades per extreure un resultat concret.

Per aconseguir aquesta fita ens centrarem en interpretar i construir consultes exclusivament amb el llenguatge associat al SGBD, l'SQL (Structured Query Language)[11].

Seguidament, utilitzarem els procediments per automatitzar la crida a la instrucció SQL i gestionar els paràmetres d'entrada necessaris per la consulta i també per a controlar possibles errors (excepcions).

Finalment afegirem l'entrada de log per a registrar la petició del procediment i retornarem informació sobre l'èxit de la crida.

En aquesta secció ens centrarem en la tasca de generació del codi SQL, concretament en l'extracció de dades del model. Els procediments i les excepcions, tot i que les afegirem, només les argumentarem de passada deixant l'aprofundiment en el tema per més endavant.

Donat que totes les consultes es generaran utilitzant el mateix procediment a grans trets, ens centrarem a detallar la creació de la primera de les sol·licitades, fent extensible el procediment a la resta.

> Donada una ciutat i una data com a paràmetres, el llistat de tots els comptadors on el consum mensual de la data indicada ha superat el 80% del consum mitjà de tots els comptadors de la ciutat en aquell mateix període de temps.
>
> En aquest llistat caldrà retornar la informació bàsica següent:
>
> Codi de contracte.
> La potència màxima contractada.
> El tant per cent de consum elèctric consumit en relació al consum mitjà.

[10] En el nostre cas Oracle Database 11g Express Edition.
[11] http://es.wikipedia.org/wiki/**SQL**

Tot això ordenat de forma ascendent pel tant per cent de consum elèctric consumit.

Generació de l'SQL. Consulta de selecció.

El primer pas serà centrar-nos en els camps (calculats o no) que haurà de retornar la consulta: Codi, potència, Mitjana de consum, històric de consum i localitat. Tot seguit, detectarem els camps que, tot i no tenir que sortir en el resultat, són indispensables per a calcular la consulta com ara la localitat i el % sobre la mitjana.

Una consulta de selecció es composa per unes instruccions molt concretes referents a tot el que necessita l'SGBD per extreure les dades sol·licitades.

SELECT (els paràmetres d'aquesta instrucció són els camps que s'han de mostrar)
DISTINCT (per a que els valors siguin diferents)
FROM (i aquí posarem les taules d'on provenen els camps utilitzats).
WHERE (les condicions que s'han de complir per ser dades vàlides, opcional)
GROUP BY (si volem afegir algun nivell d'agrupament, opcional)
HAVING (si volem condicionar els grups, opcional)
ORDER BY (si volem afegir ordenacions d'algun tipus, també opcional)

Aquesta instrucció de selecció permet afegir sub consultes en qualsevol dels seus apartats i nosaltres utilitzarem aquesta qualitat per extreure el resultat desitjat.

D'entrada crearem la consulta inicial amb el codi i la potencia de la taula contracte.

```
SELECT codi, potencia
    FROM contracte;
```

Seguidament, anirem a cercar un camp que no existeix a les taules, sinó que s'ha de crear. Necessitem la *mitjana de consum*.

Utilitzarem la funció agregada *AVG (mitjana)* per a calcular-la i com que el camp no existeix no te nom, li afegirem amb la instrucció *AS*:

```
SELECT AVG(historicc.consum) AS "mitjana consum"
    FROM historicc;
```

El resultat de la funció AVG[12] pot retornar-nos un valor amb més decimals dels desitjats. Aquestes funcions poden aniuar-se (incloure'n una dins d'un altra). Arrodonim així el valor resultant a dos decimals amb la funció *ROUND(valor,num decimals)*:

```
SELECT ROUND(AVG(historicc.consum), 2) AS "mitjana consum"
        FROM historicc;
```

És interessant centrar-nos en el retorn de la petició a la instrucció *SELECT*. Els possibles retorns podran ser des d'un simple valor, a un vector de valors i fins a una taula de valors distribuïts en files i columnes (registres i camps).

Aquesta informació és molt important de cara a reutilitzar aquest resultat per a aconseguir-ne d'altres.

Si el resultat de la consulta és un sol valor, el podrem utilitzar com a camp de sortida de la instrucció *SELECT* juntament amb els altres camps, afegint-hi un alies (*AS*) per identificar-lo:

```
SELECT codi, potencia,
                (SELECT ROUND(AVG(historicc.consum), 2) AS "mitjana consum"
                    FROM historicc) AS "Mitjana consum"
        FROM contracte;
```

El valor *mitjana consum* apareixerà per a cada element que es derivi de contracte i serà sempre el mateix.

Cal tenir en compte que el valor *mitjana consum* no està vinculat a cap valor o restricció de la taula *contracte*, aquesta mitjana no depèn dels valors seleccionats en la clàusula *FROM* de la consulta principal. Es pot deduir fàcilment que aquesta columna afegida a la consulta com a sub consulta tindrà el mateix resultat afegida a qualsevol altra consulta sobre qualsevol altra taula.

També necessitarem l'històric de consum afegit a la consulta principal:

```
SELECT      contracte .codi, contracte .potencia,
            (SELECT ROUND(AVG(historicc.consum), 2) AS "mitjana consum"
                FROM historicc) AS "Mitjana consum",
```

[12] **AVeraGe**. Mitjana en anglès.

```
        historicc.consum
    FROM contracte, historicc
    WHERE (la condició de relació no és possible);
```

Al afegir el *consum* de la taula *historicc*, automàticament hem d'afegir la taula d'on prové el camp en l'apartat *FROM*.

Aquesta consulta tal i com l'hem deixat no és funcional ja que falta definir la condició de relació que hi ha entre les dues taules *contracte* i *historicc*.

Si mirem el model veurem que les dues taules no tenen camps en comú però sí una vinculació amb la taula *comptador*. Aquesta vinculació ens permetrà definir les condicions de relació entre les tres taules i utilitzar els camps relacionals lliurement, en el nostre cas, per afegir *localitat*:

```
SELECT      contracte .codi, contracte .potencia,
            (SELECT ROUND(AVG(historicc.consum), 2) AS "mitjana consum"
                FROM historicc) AS "Mitjana consum",
            historicc.consum,
            comptador.localitat
    FROM contracte, historicc, comptador
    WHERE comptador.idcomptador = historicc.comptador_idcomptador
      AND comptador.idcomptador = contracte.comptador_idcomptador;
```

Aquesta consulta de selecció és correcta però hi falten les restriccions sol·licitades. La primera restricció ens diu que es mostraran els valors resultants de la consulta sempre que el consum superi el 80% del consum mitjà.

Per fer aquest càlcul dividirem el valor del consum per la mitjana i ens dirà el % que en representa. Llavors només caldrà condicionar aquest resultat per que sigui superior al 80%.

```
        ROUND ((historicc.consum /
        (SELECT ROUND(AVG(historicc.consum), 2) FROM historicc) * 100), 2) > 80
```

I seguidament afegirem la resta de restriccions:

```
        AND historicC.mes = EXTRACT(MONTH FROM parametre1)
```

```
    AND historicC."any" = EXTRACT(YEAR FROM parametre2)
    AND comptador.localitat = parametre3;
```

Cal comentar les funcions *EXTRACT*, per extreure un valor d'un element compost (com ara la data). Amb les funcions *MONTH* i *YEAR* extraiem els elements corresponents. Aquestes funcions ens permetran filtrar les dades de la taula *historicC* que emmagatzemen informació sobre l'any i el mes de l'històric.

Seguidament farem una reorganització dels elements. No per que no sigui funcional sinó per que l'estructura serà més correcta.

Per declarar la relació entre les taules que s'inclouen en la consulta s'ha afegit les restriccions corresponents dins la clàusula *WHERE*. Substituirem aquestes restriccions de posició i les afegirem dins l'apartat *FROM* amb la clàusula de vinculació *JOIN*.

```
    WHERE comptador.idcomptador = historicc.comptador_idcomptador
      AND comptador.idcomptador = contracte.comptador_idcomptador;
```

Per

```
    FROM comptador
     INNER JOIN historicc
     ON comptador.idcomptador = historicc.comptador_idcomptador
     INNER JOIN contracte
     ON comptador.idcomptador = contracte.comptador_idcomptador
```

I finalment:

```
SELECT      contracte .codi,
            contracte .potencia,
            (SELECT ROUND(AVG(historicc.consum), 2) AS “mitjana consum”
                  FROM historicc) AS “Mitjana consum”,
            historicc.consum,
            comptador.localitat
      FROM comptador
       INNER JOIN historicc
       ON comptador.idcomptador = historicc.comptador_idcomptador
       INNER JOIN contracte
       ON comptador.idcomptador = contracte.comptador_idcomptador
      WHERE ROUND ((historicc.consum /
         (SELECT ROUND(AVG(historicc.consum), 2) FROM historicc) * 100), 2) > 80
      AND historicC.mes = EXTRACT(MONTH FROM parametre1)
      AND historicC."any" = EXTRACT(YEAR FROM parametre2)
      AND comptador.localitat = parametre3;
```

On *parametreN* són valors que ens facilitarà l'usuari.

Optimització de la consulta. Procediments.

Per a que el motor de la base de dades pugui gestionar les instruccions SQL, gestioni els paràmetres d'entrada i faciliti la utilització de la comanda s'utilitzaran procediments.

En el següent apartat (el quart) es parlarà amplament dels procediments, aquí fem una breu introducció per a donar funcionalitat a la consulta.

```
create or replace
PROCEDURE          "Consulta_A"
(
      Localitat    IN VARCHAR2,
      dataP        IN DATE,
      RSP          OUT VARCHAR2
)
```

```
IS
BEGIN
        EXECUTE IMMEDIATE
        << Instrucció SQL generada>>
        RSP:= 'Ok';
EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Error'||SQLCODE||SQLERRM);
            RSP := 'Error'||SQLCODE||SQLERRM;
END "Consulta_A";
```

Es declara el procediment amb *create o replace* (*replace* per si ja existeix) *PROCEDURE* i seguidament es dona nom al procediment.

Entre parèntesis declararem els paràmetres del procediment posant el seu nom, si és un paràmetre d'entrada o de sortida (*IN* o *OUT*) i el tipus de dada.

Comença el cos de procediment a partir de *IS BEGIN* fins a *END "nom_procediment"* seguit d'un punt i coma.

Dins la clàusula *EXECUTE IMMEDIATE* nosaltres posarem la nostra instrucció *SQL*, substituint els paràmetres (*parametreN*) de la consulta pels noms dels paràmetres del procediment *localitat* i *dataP* dos cops, un per a l'any i un per al mes.

La variable RSP (resposta) contindrà 'Ok.' Si el procediment ha funcionat correctament.

L'apartat *EXCEPTION* gestionarà les accions o events inesperats. S'utilitzarà per a gestionar els errors que es puguin donar durant l'execució del procediment. Aquest apartat, com els anteriors es descriuran en l'apartat següent.

Ja podem fusionar la instrucció SQL amb el procediment per obtenir la funcionalitat sol·licitada pel client:

Procediment de consulta.

```
create or replace
PROCEDURE            "Consulta_A"
(
        Localitat    IN VARCHAR2,
        dataP        IN DATE,
        RSP          OUT VARCHAR2
)
IS
BEGIN
        EXECUTE IMMEDIATE
        'SELECT      contracte .codi,
                     contracte .potencia,
                     (SELECT ROUND(AVG(historicc.consum), 2) AS "mitjana consum"
                        FROM historicc) AS "Mitjana consum",
                     historicc.consum,
                     comptador.localitat
        FROM comptador
         INNER JOIN historicc
         ON comptador.idcomptador = historicc.comptador_idcomptador
         INNER JOIN contracte
         ON comptador.idcomptador = contracte.comptador_idcomptador
        WHERE ROUND ((historicc.consum /
           (SELECT ROUND(AVG(historicc.consum), 2) FROM historicc) * 100), 2) > 80
        AND historicC.mes = EXTRACT(MONTH FROM dataP)
        AND historicC."any" = EXTRACT(YEAR FROM dataP)
        AND comptador.localitat = Localitat;';
        RSP:= 'Ok';
EXCEPTION
        WHEN OTHERS THEN
            DBMS_OUTPUT.PUT_LINE('Error'||SQLCODE||SQLERRM);
            RSP := 'Error'||SQLCODE||SQLERRM;
END "Consulta_A";
```

S'inclou el procediment adjunt amb la resta de SQL generat.

Quart: Disparadors (Triggers).

L'objectiu principal de qualsevol base de dades és l'organització de les dades utilitzant un model formal, que ens permeti emmagatzemar i processar les dades per extreure informació d'una manera ràpida i eficient.

Com podem comprovar en l'apartat anterior, mitjançant *SQL* podem obtenir qualsevol tipus d'informació per complexa que aquesta sigui de generar. Per contrapartida, quanta més complexitat afegim a les nostres peticions més recursos de sistema s'han d'utilitzar per a resoldre-les.

L'SQL no és més que una normalització de llenguatge. Per tant, cada motor de base de dades s'encarrega de traduir les instruccions externes en instruccions internes per a poder processar-les. Aquest fet implica que cada SGBD aplicarà el seu sistema de gestió propi que pot ser més o menys feixuc i que estarà limitat també pel maquinari on estigui implantat.

És un fet força comú trobar-nos un sistema 'col·lapsat' quan per exemple visitem pàgines web. Pot donar-se que el maquinari més el programari estigui dissenyat per a suportar un nombre X d'execucions concurrents i que per algun fet més o menys puntual es sobrepassi aquest límit.

Es pot pal·liar els efectes d'una sobrecàrrega distribuint les feines de procés que requereixin molts recursos puntualment. D'aquesta manera reduïm considerablement el temps de resposta per aquestes peticions i descarreguem el sistema de puntes de feina.

Si fem una ullada a l'apartat anterior, on hem generat una consulta força complexa amb *SQL*, veurem que sovint hem utilitzat unes funcions que incorpora el llenguatge com ara *AVG*. Aquestes funcions agregades són molt útils a nivell de procés de dades però requereixen de molts recursos del sistema per a dur-se a terme.

Per a fer una mitjana d'un camp, l'SGBD necessitarà repassar seqüencialment tots els registres de les taules afectades per realitzar la suma i després dividir-la pel número d'elements. Si la base de dades ha de tenir un volum considerable, com ara la nostra, hauríem d'evitar aquests pics de recursos sol·licitats.

Per a obtenir la mitjana (o la suma) de tots els registres afectats d'una taula (o d'un

conjunt de taules) crearem una taula auxiliar on anirem sumant el valor en el moment en que l'usuari l'introdueixi, així com el nombre de valors que s'han sumat, si és que cerquem la mitjana. Aquesta acció, pràcticament imperceptible per l'usuari, ens estalviarà de realitzar el seguiment seqüencial de la taula en quan necessitem el total o la mitjana (que trobarem dividint la suma total entre el nombre de valors introduïts), amb l'estalvi de procés que implica.

Les especificacions del nostre model requereix el disseny d'un mòdul estadístic que ens donarà resultats de càlculs complexos amb l'afegit que no s'hauran d'utilitzar funcions agregades, per aprofitar els avantatges que representa.

Explicarem els passos seguits en la generació d'un d'aquests mòduls estadístics, tenint en compte que es segueix el mateix procediment per a la resta de mòduls estadístics.

1. Donada una central de producció, el consum dels comptadors que depenen de la central.

Per a conservar el consum dels comptadors que depenen de la central es crea una taula nova on la seva clau principal serà la mateixa clau principal que *centralPr*, sent clau principal i forana a l'hora. Es podria haver afegit el camp ConsumComptadors directament a la taula *centralPr* però, per a separar conceptes, s'ha procedit d'aquesta manera.

Estadistic_1

Nom de camp	Tipus de dades	Caràcters
CentralPr_codiC	VARCHAR2	(50)
ConsumComptadors	INTEGER	

Els triggers s'implementen inicialment definint el seu nom i quan es 'dispararà'.

```
create or replace
TRIGGER ESTADISTICA1
AFTER INSERT ON HISTORICC
```

Amb la instrucció *AFTER INSERT ON* estem advertint al disparador que haurà d'executar-se just després d'haver inserta dades en la taula *HistoricC.*

De fet, quan es faci una entrada a la taula *historicC* s'entén que volem afegir la despesa feta en un mes determinat d'un comptador.

Per a que aquesta dada quedi guardada a la taula *estadístic1*, caldrà comprovar si el comptador on s'afegeix la dada pertany a la *centralPr* on s'ha de sumar aquest valor.

Per afegir això, dins la clàusula *BEGIN* trobem l'actualització (*UPDATE*) de la taula *Estadistic_1* per sumar la despesa generada pels comptadors.

```
UPDATE Estadistic_1
   SET consumcomptadors = (consumcomptadors + :new.consum)
   WHERE centralpr_codic = (codi de la central vinculada al comptador)
```

Donat que els comptadors operen amb energia d'una central específica caldrà condicionar aquesta actualització a la central vinculada al comptador. D'aquí la clàusula *WHERE* amb una subconsulta que retorna un sol valor(en la subconsulta només cerquem el codi de la central que s'agrupa amb *GROUP BY* per evitar repeticions).

Procediment disparador.

```
create or replace
TRIGGER ESTADISTICA1
AFTER INSERT ON HISTORICC
for each row
BEGIN
 UPDATE Estadistic_1
  SET consumcomptadors = (consumcomptadors + :new.consum)
  WHERE centralpr_codic =
  (
   SELECT LINIACOM.CENTRALPR_CODIC
    FROM HISTORICC
    INNER JOIN COMPTADOR
    ON COMPTADOR.IDCOMPTADOR = HISTORICC.COMPTADOR_IDCOMPTADOR
    INNER JOIN COMPTADOR_CENTRALETA
    ON COMPTADOR.IDCOMPTADOR =
COMPTADOR_CENTRALETA.COMPTADOR_IDCOMPTADOR
    INNER JOIN CENTRALETA
    ON COMPTADOR_CENTRALETA.CENTRALETA_ADRECA = CENTRALETA.ADRECA
    INNER JOIN LINIACOM
    ON LINIACOM.CODI          = CENTRALETA.LINIACOM_CODI
    AND LINIACOM.CENTRALPR_CODIC = CENTRALETA.LINIACOM_CENTRALPR_CODIC
    WHERE historicc.comptador_idcomptador = :new.COMPTADOR_IDCOMPTADOR
    GROUP BY LINIACOM.CENTRALPR_CODIC
  );
EXCEPTION
 WHEN OTHERS THEN
  DBMS_OUTPUT.PUT_LINE('Error'||SQLCODE||SQLERRM);
END;
```

En vermell estan marcats uns elements propis dels disparadors: els valors anteriors o posteriors a l'actualització. :**old**.nom_de_camp o :**new**.nom_de_camp.

Cal recordar que el control d'excepcions actuarà si es dona una situació inesperada, com un error.

Cinquè: Procediments.

Mitjançant l'SQL es pot realitzar qualsevol acció sobre la base de dades. Tot programador té un coneixement més o menys ampli sobre el llenguatge que els permet accedir als motors de base de dades per gestionar les dades que necessiten les seves aplicacions.

De tota manera, actualment les aplicacions acostumen a generar-se amb la participació de diferents col·laboradors, així, les tasques a desenvolupar redueixen considerablement el temps que s'ha de dedicar per a portar-les a terme.

Aquesta col·laboració porta als participants a especialitzar-se en tasques específiques del conjunt i a optimitzar al màxim l'acoblament dels diferents components que formen el producte.

Una de les tasques de l'encarregat de generar el model relacional és facilitar al màxim la tasca de gestionar aquesta base de dades.

Així, tot i que el llenguatge SQL és força senzill, la majoria de les transaccions poden automatitzar-se des del motor de la base de dades i descarregar considerablement la gestió des de l'aplicació.

Per a generar aquestes transaccions l'Oracle els ofereix una ampliació de les capacitats que ofereix l'SQL, el PL-SQL[13], que dóna més capacitat als gestors.

Els procediments tenen la capacitat de realitzar processos complexos sobre les dades i oferir-les ja preparades al client (ja sigui un programa, un administrador ...) descarregant les tasques i fent les aplicacions més lleugeres i portables.

El nostre model ha d'incorporar procediments per facilitar la introducció de les dades a les taules i també per a controlar els possibles errors que es derivin de la gestió de les dades. De la mateixa manera que es gestionarà l'entrada de dades, també es gestionarà la modificació i la baixa de les mateixes.

[13] http://ca.wikipedia.org/wiki/PL/SQL

Procediment de gestió de dades.

Per a declarar un procediment utilitzem *create or replace*, ja que si ja existeix el procediment es substitueix per aquest. Després de la clàusula *PROCEDURE* introduïm el nom del procediment:

```
create or replace
PROCEDURE "ALTA_COMPTADOR"
definició de paràmetres
IS
BEGIN
Cos del procediment.
EXCEPTION
END "ALTA_COMPTADOR";
```

El procediment es cridarà pel seu nom i entre parèntesi els paràmetres que s'utilitzaran dins del cos. Aquests paràmetres podran ser paràmetres d'entrada (*IN*) o de sortida(*OUT*).

Els paràmetres que nosaltres necessitarem seran bàsicament els valors que s'hauran d'introduir a la taula de la base de dades més un paràmetre que rebrà informació sobre el bon funcionament de la transacció.

```
(
  IDCOMPTADOR                IN    NUMBER
, CARRER_VIA_PLACA           IN    VARCHAR2
, NUMEROIMMOBLE              IN    NUMBER
, PIS                        IN    NUMBER
, PORTA                      IN    NUMBER
, CP                         IN    VARCHAR2
, LOCALITAT                  IN    VARCHAR2
, PAIS                       IN    NUMBER
, KWABSOLUTS                 IN    NUMBER
, MODELCOMPTADOR_MODEL       IN    VARCHAR2
, DARRERALECTURA             IN    DATE
, RSP                        OUT   VARCHAR2
)
```

Els tipus de dades hauran de ser compatibles amb el format de les dades de la taula o bé el procediment ha de processar les dades per a ferles compatibles.

En el cos del procediment utilitzarem la instrucció *SQL* per introduir dades *INSERT INTO*, juntament amb els valors que rebem per paràmetres.

```
INSERT INTO comptador
        VALUES (IDCOMPTADOR, CARRER_VIA_PLACA, NUMEROIMMOBLE, PIS, PORTA,
        CP, LOCALITAT, PAIS, KWABSOLUTS, MODELCOMPTADOR_MODEL,
        TO_DATE (DARRERALECTURA,'dd/mm/yyyy'));
 RSP := 'Ok.';
```

El control d'excepcions l'afegirem per a gestionar els possibles mal funcionaments o imprevistos que es puguin produir durant l'execució.

```
  WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error'||SQLCODE||SQLERRM);
        RSP := 'Error'||SQLCODE||SQLERRM;
```

Procediment alta comptador.

```
create or replace
PROCEDURE              "ALTA_COMPTADOR"
(
  IDCOMPTADOR                 IN      NUMBER
, CARRER_VIA_PLACA            IN      VARCHAR2
, NUMEROIMMOBLE               IN      NUMBER
, PIS                         IN      NUMBER
, PORTA                       IN      NUMBER
, CP                          IN      VARCHAR2
, LOCALITAT                   IN      VARCHAR2
, PAIS                        IN      NUMBER
, KWABSOLUTS                  IN      NUMBER
, MODELCOMPTADOR_MODEL        IN      VARCHAR2
, DARRERALECTURA              IN      DATE
, RSP                         OUT     VARCHAR2
)
IS

BEGIN

INSERT INTO comptador
        VALUES (IDCOMPTADOR, CARRER_VIA_PLACA, NUMEROIMMOBLE, PIS, PORTA,
        CP, LOCALITAT, PAIS, KWABSOLUTS, MODELCOMPTADOR_MODEL,
        TO_DATE (DARRERALECTURA,'dd/mm/yyyy'));
 RSP := 'Ok.';
EXCEPTION

  WHEN OTHERS THEN
        DBMS_OUTPUT.PUT_LINE('Error'||SQLCODE||SQLERRM);
```

```
        RSP := 'Error'||SQLCODE||SQLERRM;
END "ALTA_COMPTADOR";
```

Sisè. Excepcions.

En la programació estructurada era necessari preveure els errors, ja que el programa ha de tenir les pautes ben marcades i no es poden donar imprevistos. És més complicat del que sembla preveure imprevistos o possibles errors que es donaran en temps d'execució i és pràcticament impossible controlar-los tots.

En la programació orientada a objectes s'enfoca el problema dels errors d'un altra manera: no són errors sinó situacions inesperades, excepcionals.

Llavors, en comptes de perseguir els possibles errors, deixem que esdevinguin els errors per a poder recollir-los i gestionar-los. Esdevé el control d'excepcions.

El seu format és el següent:

```
DECLARE
        Declaracions
BEGIN
        Execució
EXCEPTION
        Excepció
END;
```

Llavors, en l'apartat *Exception*, podem afegir les excepcions que creguem oportunes o necessàries.

```
DECLARE
        Declaracions
BEGIN
        Execució
EXCEPTION
WHEN NO_DATA_FOUND THEN
        S'executarà quan es doni una excepció de tipus NO_DATA_FOUND
WHEN ZERO_DIVIDE THEN
        S'executarà quan es doni una excepció de tipus ZERO_DIVIDE
WHEN OTHERS THEN
        S'executarà quan es doni una excepció de tipus no tractat en els apartat anteriors.
END;
```

En qualsevol cas, si es dóna alguna d'aquestes excepcions s'interromp l'execució del procediment.

En Oracle trobem dos tipus d'excepcions, les predefinides i les definides per l'usuari.

Excepcions predefinides.

El llistat de les excepcions predefinides és el següent:

Excepció	S'executarà quan ...	SQLCODE
ACCESS_INTO_NULL	El programa intenti assignar valor a una variable no iniciada.	-6530
COLLECTION_IS_NULL	El programa intenti afegir elements a una col·lecció no iniciada	-6531
CURSOR_ALREADY_OPEN	El programa provi d'obrir un cursor ja obert	-6511
DUP_VAL_ON_INDEX	El programa intenti afegir un valor duplicat a una columna amb restricció referencial.	-1
INVALID_CURSOR	El programa intenti utilitzar un cursor invàlid.	-1001
INVALID_NUMBER	Si convertim una cadena a número i no conté un número vàlid	-1722
LOGIN_DENIED	El nom d'usuari o la contrasenya d'accés és invàlid.	-1017
NO_DATA_FOUND	Una sentència SELECT INTO no retorna valors.	100
NOT_LOGGED_ON	No esteu connectat al motor d'oracle.	-1012
PROGRAM_ERROR	Problema intern PL/SQL	-6501
ROWTYPE_MISMATCH	Error de tipus en assignació de valors a variables o a objectes.	-6504
SELF_IS_NULL	El paràmetre SELF és null.	-30625
STORAGE_ERROR	Error de memòria o memòria plena.	-6500
SUBSCRIPT_BEYOND_COUNT	Recerca d'un element indexat fora de límits.	-6533
SUBSCRIPT_OUTSIDE_LIMIT	Accés a un element d'una col·lecció fora de límits.	-6532
SYS_INVALID_ROWID	La conversió de String a RowID ha fallat ja que no conté un número.	-1410
TIMEOUT_ON_RESOURCE	Exhaurit el temps d'espera per a una petició a Oracle.	-51
TOO_MANY_ROWS	La sentència SELECT INTO retorna més d'una fila.	-1422
VALUE_ERROR	Error aritmètic, de truncament o conversió.	-6502
ZERO_DIVIDE	Divisió per zero.	-1476

Excepcions definides per l'usuari.

Les excepcions, com a objectes que són s'hauran de declarar a la secció *DECLARE*, i posteriorment caldrà llençar-les amb la instrucció *RAISE*

```
DECLARE
        Fallida EXCEPTION;
BEGIN
        Execució
        …
        IF (…)
                RAISE Fallida;
EXCEPTION
        WHEN Fallida THEN
        dbms_output.put_line('Descripció de l'error.');

END;
```

Valoració econòmica

És difícil sempre fer una valoració econòmica per a un projecte i encara més si ets una persona amb poca experiència en el camp de la comercialització de projectes de programari.

De tota manera, una aproximació econòmica sempre és possible realitzar, tenint en compte que el preu hora que podria cobrar jo no seria un preu hora tant eficaç com la de qualsevol professional amb experiència.

Llavors, la dedicació directa al desenvolupament del projecte arrodonim a 90 hores. La dedicació exclusiva a la generació de la documentació que acompanya el projecte ronda les 70.

Llavors un càlcul senzill (sense coses exhaustives com el cocomo) seria un preu hora de 20€ multiplicat per 90 hores fan un total de **3.200€**. Tenint en compte que el software no representa cost econòmic i el desgast dels aparells informàtics (un portàtil i un ordinador fix) és pràcticament inapreciable.

Conclusions.

Les aplicacions clàssiques estaven dissenyades per a un sistema concret amb una gestió de dades concreta, pròpia i difícilment manipulable.

Amb l'aparició dels models relacionals i dels formats estandarditzats, les aplicacions varen començar a separar el codi 'dur' dels programes de les dades que utilitzaven, facilitant així la gestió de les dades externament.

Actualment, amb aparells com els smartphones14 o les tablets, és necessari que el programari que es crea sigui independent de la plataforma on hagi de ser executat. Es popularitza el Java15.

Aquesta disparitat d'aparells, també implica una disparitat de capacitats (recursos) d'execució, amb el que els dissenyadors de programari tendeixen a 'alleugerir' el software de mòduls o funcionalitats que carreguin massa el sistema.

Amb solucions com PL-SQL16, es pot traspassar funcionalitats pròpies de l'aplicació directament al motor de la base de dades alleugerint així els programes finals i deixant els aplicatius pràcticament com una interfície entre l'usuari i la BBDD, que fa la resta de la feina.

Amb el temps s'ha fet un trasbals de procés del programa final fins al servidor remot, arribant a crear-se eines totalment executades al servidor, com Google Docs (ara Drive17).

Com a creador d'aplicacions tinc molt present aquest moviment de funcionalitats i la meva opinió és que tots els programadors estem obligats a actualitzar els nostres coneixements i ampliar les nostres capacitats amb les eines que ens ofereixen.

El futur passa, entre altres elements, per les Bases de dades.

[14] Telèfon intel·ligent. http://ca.wikipedia.org/wiki/Telèfon_intel·ligent
[15] Llenguatge de programació. http://ca.wikipedia.org/wiki/Java_(llenguatge_de_programació)
[16] Procedural Language SQL. http://en.wikipedia.org/wiki/PL/SQL
[17] Disc virtual. https://drive.google.com