Universitat Oberta de Catalunya

uoc.edu

Global Service Desk

Desenvolupament d'una base de dades de suport per a un sistema gestor d'incidències

Joaquim Galan i Gil
Grau d'Enginyeria Informàtica
Bases de dades

Jordi Ferrer i Duran
María Isabel Guitart i Hormigo

11/06/2018

FITXA DEL TREBALL FINAL

Títol del treball:	*Global Service Desk* *Desenvolupament d'una base de dades de suport per a un sistema gestor d'incidències*
Nom de l'autor:	*Joaquim Galan i Gil*
Nom del consultor/a:	*Jordi Ferrer i Duran*
Nom del PRA:	*María Isabel Guitart i Hormigo*
Data de lliurament (mm/aaaa):	*06/2018*
Titulació o programa:	*Grau d'Enginyeria Informàtica*
Àrea del Treball Final:	TFG - Bases de dades
Idioma del treball:	*Català*
Paraules clau	*Global Service Desk*

Resum del Treball (màxim 250 paraules): *Amb la finalitat, context d'aplicació, metodologia, resultats i conclusions del treball*

Global Service Desk –abreviat GSD- és un gestor d'incidències desenvolupat íntegrament sobre la plataforma Oracle 11g per a una multinacional amb seus a diferents països. Les principals característiques d'aquesta aplicació són la creació d'incidències convenientment tipificades, un mecanisme efectiu de control de canvis mitjançant una taula de *log*, la possibilitat d'interactuar amb els usuaris enviant missatges i una motor de consultes estadístiques, desenvolupat amb la idea de no penalitzar el rendiment del sistema.

La metodologia que s'ha seguit en el desenvolupament d'aquest projecte ha estat el cicle de vida clàssic, que diferencia clarament quatre fases: requeriments, anàlisi, programació i proves (la fase de manteniment queda fora de l'abast d'aquest TFG). En la pràctica, però, aquesta aproximació s'ha mostrat com excessivament rígida i inflexible al canvi, fins al punt que l'estudiant considera que hauria estat més encertat optar de bon principi per una metodologia més moderna, com pugui ser *agile* o *scrum*.

El resultat d'aquest treball són una sèrie de *scripts* en llenguatge PL/SQL que originen els objectes de la BD necessaris per satisfer tots els requeriments identificats. També es genera un joc de dades, mínim però suficient, que permet testejar de manera efectiva les principals característiques de l'aplicació, amb especialment èmfasi en l'apartat de consultes estadístiques.

Aquest treball ha permès estudiar a fons els requeriments necessaris per implementar amb èxit un gestor d'incidències, així com per entendre també la complexitat inherent a la manipulació de grans volums de dades, propi d' un entorn *Datawarehouse*.

Abstract (in English, 250 words or less):

Global Service Desk –GSD for short- is an issue manager developed entirely with Oracle 11g for an international company, with branches in different countries. Main features of this application are the creation of properly typed issues, an effective mechanism to control changes using a log table, the possibility of interacting with users by sending messages and a statistical queries engine, developed with system performance in mind.

The methodology applied to develop this project has been the classical life cycle, which clearly identifies four different phases: requirements, analysis, development and testing (maintenance is considered out of scope for this project). In practice, however, this classical approach has demonstrate to be excessively rigid and inflexible in front of changes, so the student believes now that it would be preferible to choose a more modern methodology, such as agile or scrum.

The resulting artifacts of this project are a serie of PL/SQL scripts that originate all the necessary BD objects, with the self-contained functionality to meet all the identified requirements. It also generates a minimal dataset, suffice to test the main characteristics of the application, with especially emphasis in the section of statistical queries.

This project has allowed us to study in depth the necessary requirements to successfully implement an issue manager and also to understand the inherent complexity of handling large volumes of data in a Data warehouse environment.

Índex

1. Introducció

1.1 Context i justificació del Treball

Una empresa multinacional, amb sèus a diferents països, ha optat per desenvolupar un sistema gestor d'incidències centralitzat. Les principals motivacions que hi han darrera d'aquesta decisió són:

- La possibilitat que qualsevol incidència o petició que s'origini arreu del món quedi enregistrada dins d'aquest sistema, de manera que es pugui derivar -en funció de la seva complexitat- a l'equip d'especialistes millor preparats per resoldre-la.
- També es desitja poder dur a terme l'oportú seguiment de tots els casos, fins el tancament dels mateixos.
- La interacció amb l'usuari que origina la incidència també serà un aspecte fonamental a tenir en compte: es vol mantenir a l'interessat informat en tot moment de quin és l'estat de la seva incidència.
- Finalment, un cop posat en marxa, el nou sistema permetrà obtenir informació estadística de valor, com pugui ser: temps promig de resolució d'incidències, problemes més freqüents, tècnics més eficients, punts més "conflictius", etc.

La quantitat de recursos, tan humans com econòmics, que una empresa d'aquesta envergadura destina a resoldre tota la casuística relacionada amb la gestió d'incidències pot arribar a ser considerable. És comprensible, doncs, que es vulgui racionalitzar i optimitzar tot el relacionat amb aquesta problemàtica i l'adopció d'un sistema gestor d'incidències sembla la millor manera d'aconseguir-ho.

No tenim informació sobre la forma actual com es resolen les incidències en aquesta companyia, però cal pressuposar que es tracta d'una solució ineficient, que no cobreix les necessitats més elementals esperades en un sistema d'aquestes característiques, com puguin ser:

- Identificació de problemes habituals.
- Estandardització de les solucions a aquests problemes habituals, mitjançant una base de dades de coneixements (*knowledge database*).
- Categorització de les incidències en funció de la seva tipologia i grau de complexitat.
- Control d'usuaris (i.e., qui pot fer peticions al sistema).
- Seguiment dels casos, estat actual dels mateixos, tancament,...
- Eficiència del personal tècnic.
- Punts més conflictius
- Etc

El nou sistema proposat pretén superar totes aquestes limitacions, així com satisfer també tots els requeriments identificats en aquest projecte. També s'espera del sistema que sigui prou flexible i obert per permetre incorporar noves especificacions en el futur.

1.2 Objectius del Treball

A banda de la finalitat pedagògica per l'estudiant que pugui oferir aquest treball en si mateix, el principal objectiu del mateix es desenvolupar una base de dades, que esdevingui l'element nuclear i fundacional d'un sistema gestor d'incidències. En aquest sentit, algunes de les característiques esperades del producte resultant haurien de ser:

- Traçabilitat i seguiment dels casos: el sistema ha de permetre en tot moment saber en quin estat es troba cada cas, si ja està resolt o qui s'està ocupant de resoldre'l i per quins estadis ha passat entre mig.

- *Feedback* amb l'usuari: el sistema ha d'oferir mecanismes àgils de comunicació amb l'usuari per informar-lo en tot moment de quin estat es troba el seu cas.

- Nivells de suport: el sistema disposarà de diferents nivells de suports, segons la complexitat de cada cas plantejat. El *workflow* d'assignació de nivell de suport per a cada cas seguirà un model jeràrquic, de menor a major complexitat. Els nivells de suports seran configurables, per poder donar resposta a necessitats futures.

- Equips de suport: el sistema disposarà de diferents equips de suport. Cada equip de suport donarà únicament suport a casos de la complexitat que tingui assignada i comptarà amb tècnics especialitzats per fer aquest tipus de tasques. Els equips de suports seran configurables, per poder donar resposta a necessitats futures.

- Identificació i tipificació de problemes habituals: el sistema ha de tenir identificats els problemes més habituals i oferir les solucions més apropiades en cada cas (*best practices*) per resoldre'ls en el menor temps possible. En aquest sentit, una base de dades de coneixements (*knowledge database*) resultaria de gran ajuda.

- Control d'accés: el sistema no ha de permetre l'accés públic, sinó que estarà reservat únicament als usuaris de la companyia, previ registre. Caldrà, doncs, definir els processos necessaris per enregistrar aquests usuaris i validar el seu posterior accés.

- Control de canvis: el sistema hauria d'incorporar algun mecanisme de *log* o control de canvis, que permetés saber QUI modifica QUE.

- Independència d'aplicacions de tercers: tota la funcionalitat ha d'estar auto continguda en la base de dades resultant, ja sigui mitjançant *triggers*, *stored procedures* o qualsevol altra característica que el producte ofereixi.

- Escalabilitat: a nivell intern, la base de dades resultant ha de ser escalable. Per escalabilitat s'entén la capacitat d'un sistema per créixer sense que es vegi afectat el seu rendiment. El terme "créixer" pot tenir diverses accepcions: major nombre d'usuaris, major volum de transaccions, major quantitat de dades emmagatzemades, etc. En el cas que ens ocupa, l'escalabilitat està garantida per la tria del propi SGBDR, ja que Oracle, per definició, està dissenyat amb una arquitectura escalable en ment. Es podria objectar que les versions *Express Edition* sí que tenen limitacions pel que fa a la mida màxima de la BD i nombre de CPU's però, arribat el cas, no seria problemàtic llicenciar una versió de pagament, superant aquests inconvenients. Cal recordar també que l'escalabilitat no s'assoleix únicament amb la tria del *software* adequat, sinó que el *hardware* sobre el que aquest s'executa també és determinant: un producte altament escalable que "corri" sobre un servidor mal dimensionat oferirà igualment un pobre rendiment. En qualsevol cas, no és un assumpte que ens hagi de preocupar ara, assumint que la tria d'Oracle ja garanteix l'escalabilitat futura.

- Característiques de *Data Warehouse*: sota el concepte de *Data Warehouse* s'engloben tota una sèrie de tècniques destinades a optimitzar el tractament de grans volums de dades. En concret per aquest projecte, aquesta filosofia s'ha aplicat en el desenvolupament del motor estadístic. En la fase de requeriments, s'han identificat una sèrie de consultes estadístiques que el sistema ha de saber respondre, les quals operen majoritàriament sobre la taula de *tickets*, una de les que, previsiblement, contindrà un major nombre de registres de tota l'aplicació. Per resoldre aquestes consultes és necessari dur a terme una sèrie d'operacions amb un elevat cost computacional (agrupacions, sumes, etc.) que podrien comprometre el rendiment del sistema. Per evitar-ho, s'ha desenvolupat un procés d'actualització massiva de dades estadístiques, amb la idea que s'executi en hores de baixa activitat del sistema (hores "vall"). Així, els usuaris que necessiten executar consultes estadístiques únicament accedeixen, en temps constant, a una taula de resultats prèviament calculats. Òbviament, amb aquest plantejament, és necessari que els resultats estadístics s'actualitzin cada cert temps per no quedar obsolets.

1.3 Enfocament i mètode seguit

Davant el problema exposat, hi ha la possibilitat d'adoptar alguna de les solucions comercials existents al mercat per a la gestió d'incidències, que sovint trobem sota denominacions tan diverses com *Help Desk, Ticketing* o *Issue Tracking.* Una altra possible alternativa seria optar per fer un desenvolupament a mida. Com de costum, ambdós plantejaments presenten avantatges i inconvenients:

Solucions *Help Desk* comercials	
Avantatges	**Inconvenients**
Relativament ràpides d'implantar. Funcionalitat estàndard completa, molt provada i lliure d'errors importants Altament configurables.. Actualitzacions i millores periòdiques. Solucions tecnològicament avançades. Bona documentació. Adequat suport tècnic i comercial.	Cost d'adquisició de llicències. Possible cost de manteniment anual (SaaS). Codi font no disponible. Poca flexibilitat (l'empresa s'ha d'adaptar a la funcionalitat oferta pel programa). Cicle de vida de l'aplicació decidit pel fabricant.

Desenvolupament fet a mida	
Avantatges	**Inconvenients**
Màxima flexibilitat (el programa s'adapta totalment a les necessitats de l'empresa). No hi ha cost d'adquisició de llicències. Cicle de vida de l'aplicació decidit pel propietari, que pot ser tan llarg com aquest vulgui (amortització de la inversió assegurada). Codi font disponible.	El cost de desenvolupament pot ser elevat. També hi ha cost de manteniment anual. Al tenir cicles de vida llargs, són aplicacions que, tard o d'hora, esdevenen tecnològicament obsoletes. Dependència dels programadors que l'han fet (encara que siguin empleats propis). Poden contenir errors no detectats. No sol haver-hi bona documentació.

En un estadi entremig, es troben les solucions *open source*, que si bé no representen cap cost en concepte d'adquisició de llicències i posen el codi font a disposició de la comunitat, en ocasions també poden presentar alguns inconvenients:

Solucions *help desk open source*	
Avantatges	**Inconvenients**
Codi font disponible,cosa que permet fer adaptacions a mida, si s'escau. No hi ha cost d'adquisició de llicències. Ampli suport de la comunitat,sovint amb línia directa amb els desenvolupadors. Actualitzacions i millores periòdiques, normalment més ràpides que en el cas dels productes comercials (especialment s'hi s'ha detectat un error greu).	Documentació pobre o dispersa. Menor base instal·lada. Són productes que requereixen de certs coneixements tecnològics per poder ser desplegats. Dificultat per trobar personal tècnic que coneguin bé aquests productes.

Cal advertir que existeix una certa confusió en referència als gestor d'incidències *open source*: molts productes que es presenten sota aquesta denominació són en realitat versions "retallades" (*scaled down*) de productes de pagament, amb la seva funcionalitat limitada (per exemple, en el nombre màxim d'usuaris als quals permeten donar servei). A més, aquestes aplicacions, tot i ser gratuïtes, no posen el codi font a disposició de la comunitat –cosa que òbviament permetria eliminar les seves restriccions- i, per tant, no haurien de ser considerades autèntiques *open source*. En qualsevol cas, a efectes de simplicitat, aquest és un matis que no es tindrà en compte de cara a la següent enumeració de productes.

Algunes destacades solucions comercials per a la gestió d'incidències son:

- Zendesk:
- HappyFox
- ZohoDesk
- OTRS Business Solution
- ManageEngine

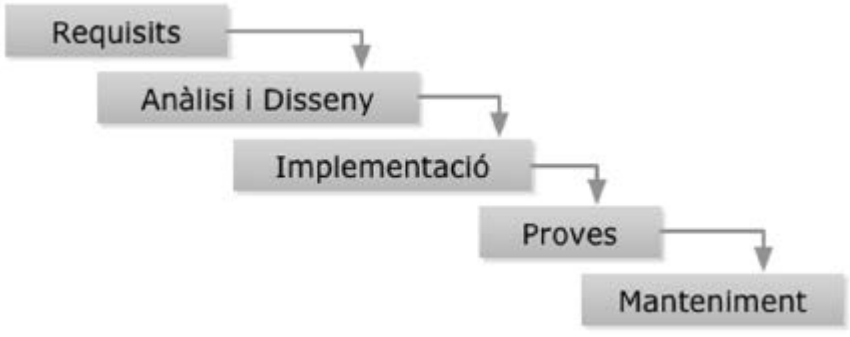
D'altra banda, algunes destacades solucions *open source* per a la gestió d'incidències son:

- osTicket
- OTRS Free
- C-DESK
- Freshdesk
- GLPI

Tots els productes acabats d'esmentar són, amb les seves fortaleses i debilitats, complets paquets de gestió d'incidències i inclouen totes o bona part de les funcionalitats esperades en aquest tipus d'aplicacions. Amb tot, al ser solucions tancades, obliguen a un major o menor esforç d'adaptació per part de la companyia, cosa que entra en conflicte amb la seva política interna d'aposta decidida pel programari lliure i el desenvolupament propi (sic). A més, poden resultar complicats d'integrar amb altres desenvolupaments interns que ja s'hagin dut a terme.

Així doncs, sembla més apropiat decantar-se per un desenvolupament a mida en el cas que ens ocupa. Un dels principals objectius d'aquesta aplicació, a més de resoldre tota la problemàtica actual associada a la gestió de incidències, és obtenir valuosa informació estadística *a posteriori*, que permeti prendre les millors decisions possibles, orientades a l'optimització d'aquest servei. En aquest sentit, sembla complicat que els paquets existents puguin donar resposta a tots els requeriments -tant presents com futurs-, que la companyia ha plantejat en aquest apartat, cosa que senyala el desenvolupament a mida com una bona solució. Per descomptat, tenir el control total sobre aquesta aplicació també simplificaria la seva possible integració amb altres desenvolupaments, argument que reforça encara més la decisió d'apostar per un **desenvolupament a mida**.

Un cop escollit el tipus de projecte que es vol abordar, el següent pas és escollir la metodologia que es vol aplicar per portar-lo a bon port i assolir els objectius definits. En aquest cas, s'opta per seguir un cicle de vida clàssic, també anomenat en cascada. Encara que aquest mètode és poc tolerant a possibles canvis en els requisits inicials -en comparació amb altres de tipus iteratiu o àgil-, és prou bo quan es tracta d'implementar un objectiu clar emprant solucions conegudes, com és el cas que ens ocupa.



Cicle de vida en cascada,
"Introducció a l'enginyeria del programari", pàg. 36, J. Pradel, J. Raya [2]
Publicacions de la UOC, PID_00171144

Aquest treball cobrirà les 4 primeres fases (Requisits, Anàlisi i Disseny, Implementació i Proves), deixant fora d'abast la fase de Manteniment, per entendre que aquesta ja correspondria a l'explotació de l'aplicació en un escenari real.

1.4 Planificació del Treball

Recursos que es faran servir

Les eines de programari que es faran servir per elaborar aquest projecte són:

- **Oracle 11g Express Edition**, com a contenidor de totes les dades i codi necessaris pel projecte d'aquest TFG.
- **Oracle SQL Developer ver. 4.0.2.15**, com a *front end* de la BD.
- **Microsoft Word 2007**, editor de text per documentar la memòria d'aquest TFG.
- **Microsoft Excel 2007**, full de càlcul emprat generar el cronograma amb la planificació temporal prevista de totes les tasques d'aquest TFG.
- **ArgoUML**: eina de modelat UML per confecciona el DER (disseny conceptual de la BD).
- **PDFCreator**, per generar la memòria del TFG en format PDF.
- **7-Zip**, compressor de fitxers per generar l'arxiu final a lliurar en format ZIP, que inclourà la memòria i el vídeo de defensa d'aquest TFG, així com els *scripts* SQL necessaris per regenerar tots els objecte de la BD.
- **Microsoft PowerPoint 365**, per l'elaboració del vídeo de presentació del projecte.

Treballs identificats

A l'hora d'identificar tots els treballs que s'han de dur a terme per assolir els objectius d'aquest TFG, s'ha trobat oportú diferenciar entre dues línies d'actuació:

- Treballs relacionats amb l'obtenció de l'artefacte final a lliurar, és a dir, la base de dades Oracle: aquí s'inclouran totes les tasques relacionades amb la gestió del projecte en llarg de les seves diferents fases:
  - Requisits: elaboració de la llista de requisits.
  - Anàlisi i Disseny: elaboració del DER, disseny lògic de la BD, etc.
  - Implementació: implementació física, generació de tots els objectes de la BD (taules, consultes, *stored procedures*, etc).
  - Proves: elaboració d'un joc de proves i verificació d'acompliment dels requisits.

- Treballs relacionats amb l'elaboració d'aquesta memòria: aquí s'inclouran tots les tasques relacionades amb la pròpia documentació d'aquesta memòria, incloent la preparació del vídeo per a la defensa final del TFG i altres tasques menors (correcció ortogràfica, revisió d'estils, etc).

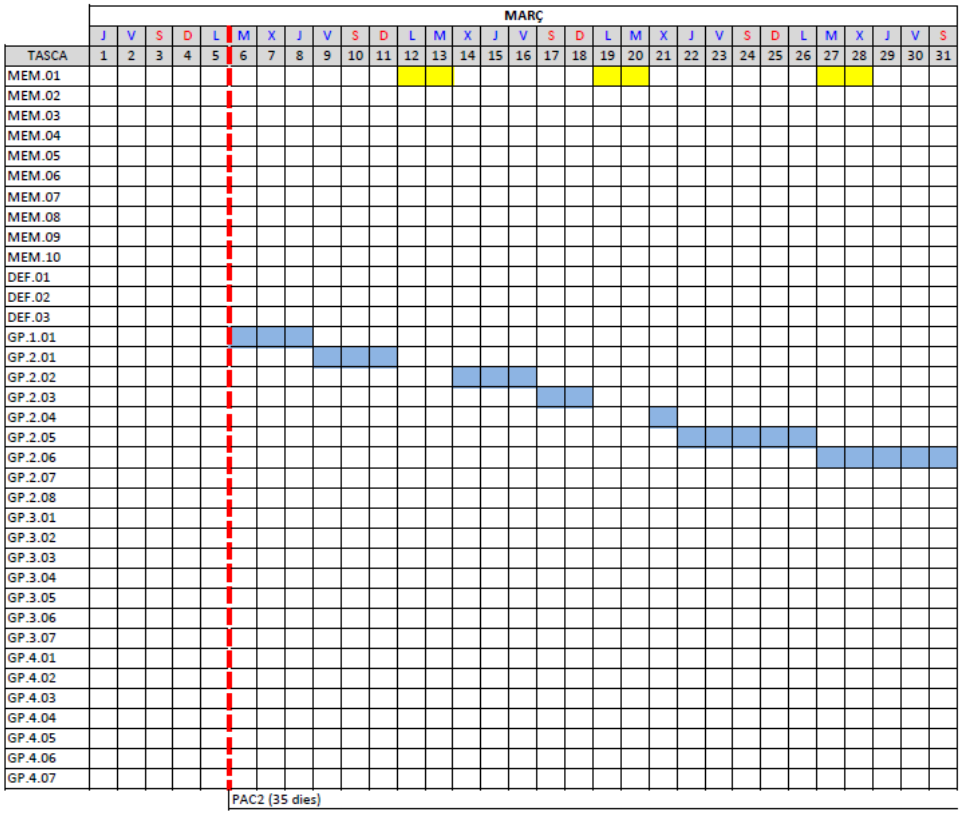
Descomposició de treballs en tasques

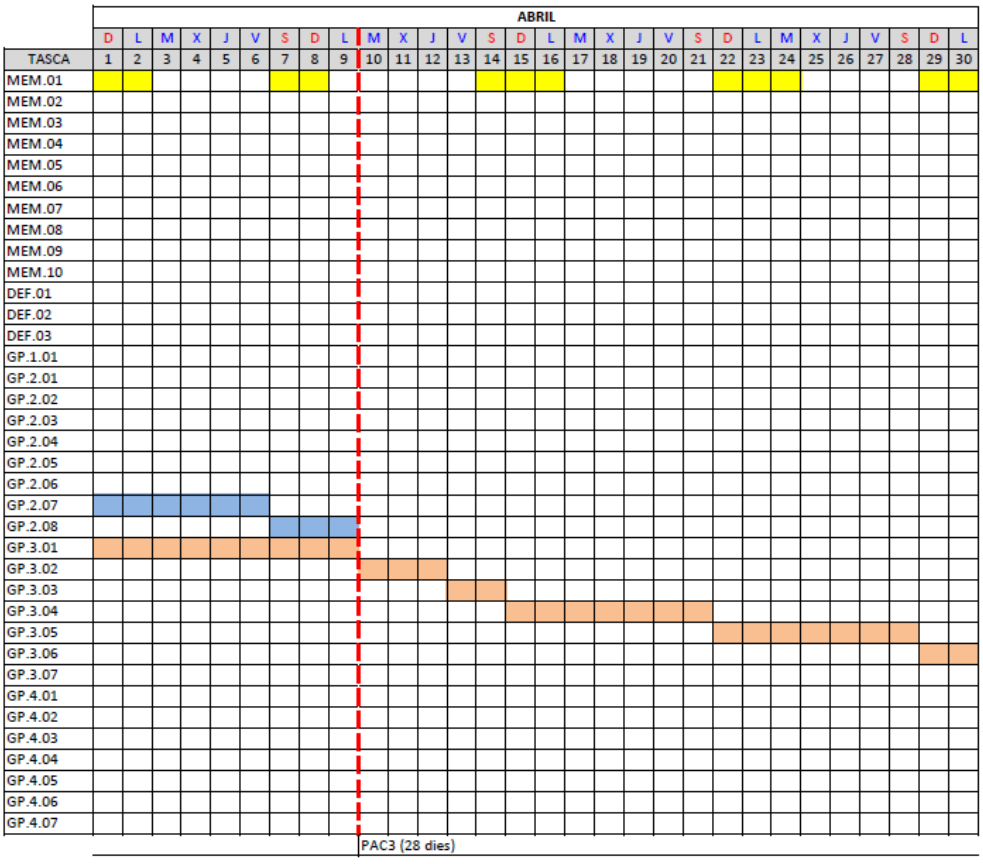
A continuació, s'indiquen les diferents tasques identificades, convenientment codificades i agrupades segons el tipus de treball a que pertanyen.

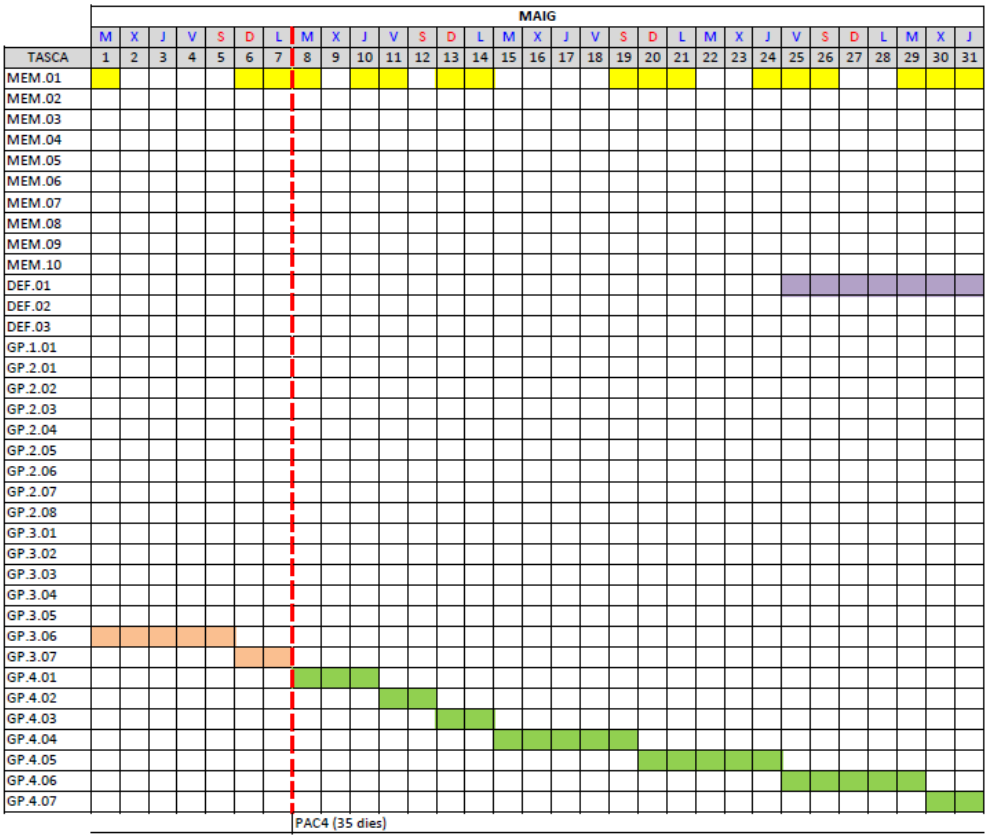
Tipus Treball	Id Tasca	Descripció Tasca
Memòria	MEM.01	Documentar cos principal Memòria
Memòria	MEM.02	Breu descripció resta capítols
Memòria	MEM.03	Resum (250 paraules)
Memòria	MEM.04	Abstract in english (250 words)
Memòria	MEM.05	Propostes de millora
Memòria	MEM.06	Conclusions
Memòria	MEM.07	Maquetació final (glossari, biblio, annexos, etc)
Memòria	MEM.08	Correcció ortogràfica
Memòria	MEM.09	Auto informe CT
Memòria	MEM.10	Preparar lliurament
Memòria	MEM.11	Debat virtual aula UOC
Defensa	DEF.01	Elaboració presentació defensa TFG
Defensa	DEF.02	Formació pròpia aplicació CAMTASIA
Defensa	DEF.03	Elaboració vídeo defensa TFG
Gestió Projecte	GP.1.01	Fase 1 – Identificar requisits
Gestió Projecte	GP.2.01	Fase 2 – Disseny conceptual BD (DER)
Gestió Projecte	GP.2.02	Fase 2 – Disseny lògic BD
Gestió Projecte	GP.2.03	Fase 2 – Normalització de taules
Gestió Projecte	GP.2.04	Fase 2 – Identificar CONSTRAINTS
Gestió Projecte	GP.2.05	Fase 2 – Identificar TRIGGERS (ABM)
Gestió Projecte	GP.2.06	Fase 2 – Identificar STORED PROCEDURES
Gestió Projecte	GP.2.07	Fase 2 – Identificar QUERYS (consultes)
Gestió Projecte	GP.2.08	Fase 2 – Checkpoint (revisió prèvia entrega PAC2)
Gestió Projecte	GP.3.01	Fase 3 – Formació pròpia PL/SQL (revisió)
Gestió Projecte	GP.3.02	Fase 3 – Implementar taules i relacions
Gestió Projecte	GP.3.03	Fase 3 – Implementar CONSTRAINTS
Gestió Projecte	GP.3.04	Fase 3 – Implementar TRIGGERS (ABM)
Gestió Projecte	GP.3.05	Fase 3 – Implementar STORED PROCEDURES
Gestió Projecte	GP.3.06	Fase 3 – Implementar QUERIES (consultes)
Gestió Projecte	GP.3.07	Fase 3 – Checkpoint (revisió prèvia entrega PAC3)
Gestió Projecte	GP.4.01	Fase 4 – Joc de dades de prova
Gestió Projecte	GP.4.02	Fase 4 – Proves: integritat PK i EK
Gestió Projecte	GP.4.03	Fase 4 – Proves: CONSTRAINTS
Gestió Projecte	GP.4.04	Fase 4 – Proves: TRIGGERS (ABM)
Gestió Projecte	GP.4.05	Fase 4 – Proves: STORED PROCEDURES
Gestió Projecte	GP.4.06	Fase 4 – Proves: QUERIES (consultes)
Gestió Projecte	GP.4.07	Fase 4 – Checkpoint (revisió prèvia entrega final)

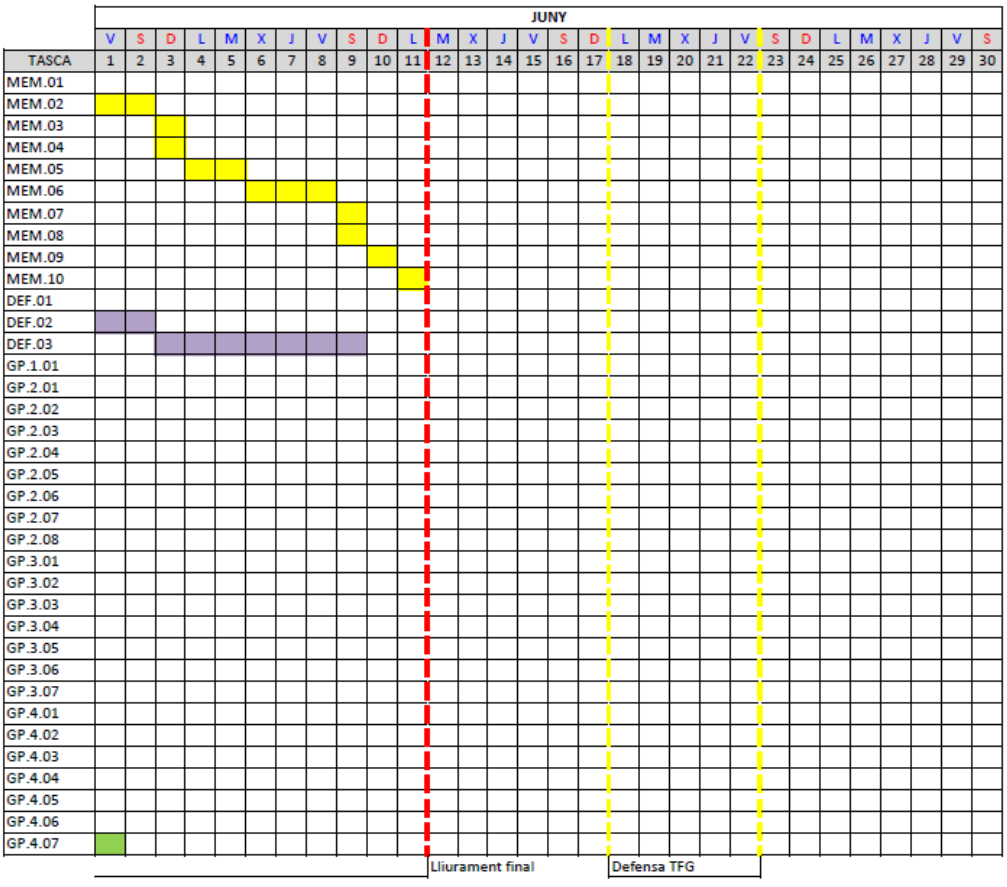
L'ordre d'execució d'aquestes tasques està indicat en el cronograma de la següent secció.

Cronograma


MARÇ
J V S D L M X J V S D L M X J V S D L M X J V S D L M X J V S
TASCA 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
MEM.01
MEM.02
MEM.03
MEM.04
MEM.05
MEM.06
MEM.07
MEM.08
MEM.09
MEM.10
DEF.01
DEF.02
DEF.03
GP.1.01
GP.2.01
GP.2.02
GP.2.03
GP.2.04
GP.2.05
GP.2.06
GP.2.07
GP.2.08
GP.3.01
GP.3.02
GP.3.03
GP.3.04
GP.3.05
GP.3.06
GP.3.07
GP.4.01
GP.4.02
GP.4.03
GP.4.04
GP.4.05
GP.4.06
GP.4.07
PAC2 (35 dies)


ABRIL
D L M X J V S D L M X J V S D L M X J V S D L M X J V S D L
TASCA 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
MEM.01
MEM.02
MEM.03
MEM.04
MEM.05
MEM.06
MEM.07
MEM.08
MEM.09
MEM.10
DEF.01
DEF.02
DEF.03
GP.1.01
GP.2.01
GP.2.02
GP.2.03
GP.2.04
GP.2.05
GP.2.06
GP.2.07
GP.2.08
GP.3.01
GP.3.02
GP.3.03
GP.3.04
GP.3.05
GP.3.06
GP.3.07
GP.4.01
GP.4.02
GP.4.03
GP.4.04
GP.4.05
GP.4.06
GP.4.07
PAC3 (28 dies)


MAIG
M X J V S D L M X J V S D L M X J V S D L M X J V S D L M X J
TASCA 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
MEM.01
MEM.02
MEM.03
MEM.04
MEM.05
MEM.06
MEM.07
MEM.08
MEM.09
MEM.10
DEF.01
DEF.02
DEF.03
GP.1.01
GP.2.01
GP.2.02
GP.2.03
GP.2.04
GP.2.05
GP.2.06
GP.2.07
GP.2.08
GP.3.01
GP.3.02
GP.3.03
GP.3.04
GP.3.05
GP.3.06
GP.3.07
GP.4.01
GP.4.02
GP.4.03
GP.4.04
GP.4.05
GP.4.06
GP.4.07
PAC4 (35 dies)


JUNY
V S D L M X J V S D L M X J V S D L M X J V S D L M X J V S
TASCA 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
MEM.01
MEM.02
MEM.03
MEM.04
MEM.05
MEM.06
MEM.07
MEM.08
MEM.09
MEM.10
DEF.01
DEF.02
DEF.03
GP.1.01
GP.2.01
GP.2.02
GP.2.03
GP.2.04
GP.2.05
GP.2.06
GP.2.07
GP.2.08
GP.3.01
GP.3.02
GP.3.03
GP.3.04
GP.3.05
GP.3.06
GP.3.07
GP.4.01
GP.4.02
GP.4.03
GP.4.04
GP.4.05
GP.4.06
GP.4.07
Lliurament final
Defensa TFG

1.5 Breu sumari de productes obtinguts

L'artefacte resultant d'aquest treball serà una base de dades desenvolupada sobre la plataforma *Oracle 11g Express Edition*, emprant el llenguatge propi d'aquest producte, PL-SQL, que és una variant del SQL estàndard. Els *scripts* lliurats generaran tots els objectes de la base de dades (taules, consultes SQL, *stored procedures*, etc), així com també el joc de dades de prova necessari per testejar els requeriments sol·licitats.

Tota la funcionalitat i validacions necessàries estaran auto contingudes dins la pròpia base de dades, ja sigui en forma de *constraints*, *triggers* o *stored procedures*; per tant, no resulta necessària cap altre aplicació de tercers per realitzar les proves d'usabilitat i verificació de requisits. En aquesta memòria del treball quedarà convenientment documentat com s'han dut a terme aquestes proves i quins resultats s'han obtingut.

1.6 Breu descripció dels altres capítols de la memòria

1.6.1 Capítol 2. Requisits

En aquest capítol es recullen tots els treballs relacionats amb la primera fase del cicle de vida en cascada que, bàsicament, consisteixen en identificar tots els requisits a partir de tota la documentació de que es disposa. Seguint la divisió proposada pels professors Pradel i Raya[1], aquests requisits es diferencien entre funcionals i no funcionals. Per a cada requisits identificat, es proposa també una possible solució d'implementació, que doni resposta a la necessitat plantejada.

L'artefacte resultant d'aquesta fase és una llista -o inventari, si es prefereix- de tots els requeriments identificats. Aquest inventari es pot fer servir més endavant com a *checklist* per validar que el producte obtingut satisfà tots els requeriments identificats.

1.6.2 Capítol 3. Anàlisi i disseny

Tenint presents els requeriments obtinguts en la fase anterior, aquest capítol recull tots els treballs relacionats amb la segona fase del cicle de vida en cascada. Probablement aquesta és la fase més crítica de tot el projecte, ja que les decisions adoptades en aquest punt tindran repercussió directa en el posterior desenvolupament del mateix i el resultat final.

En aquesta fase s'abordarà el disseny conceptual de la BD, que després originarà el disseny lògic de la mateixa. L'artefacte resultant del disseny conceptual serà el diagrama Entitat/Relació (DER), modelat d'acord amb l'estàndard UML.

[1] J.Pradell, J.Raya,"Requisits", pàg. 8 [1].

Tot i que encara no ens decantat per cap SGBD en concret, a nivell de disseny lògic sí que ja s'ha d'escollir aquí entre alguns dels models teòrics existents pel que fa a modelatge de BD. Com era d'esperar, l'opció triada serà el model relacional, atès que es tracta d'una de les teories més sòlides que existeixen; també la més estesa, amb un gran base instal·lada i un nombre elevat de productes que la suporten (SGBDR).

L'artefacte resultant del disseny lògic serà una proposta de relacions i atributs que són el pas preliminar a la creació física de les taules. Com ja s'ha dit, aquesta proposta està clarament orientada al model relacional escollit i ja emprarà alguns conceptes propis d'aquest model (claus primàries, claus externes, etc).

1.6.3 Capítol 4. Implementació

En el capítol 4 es correspon amb la fase d'implementació del projecte i aquí ja s'abandona l'àmbit purament teòric per abordar aspectes més pràctics. En concret, la tria d'un SGBDR determinat (Oracle) permet començar a definir el disseny físics dels objectes que han de contenir les dades (taules), així com tot el conjunt de procediments i disparadors (*triggers*) que han de contenir tota la lògica de negoci necessària per satisfer els requisits demanats.

Com no podia ser d'una altra manera, el llenguatge emprat per dur a terme aquesta implementació serà SQL, l'estàndard per antonomàsia del model relacional. L'artefacte resultant del disseny físic seran una sèrie de *scripts* SQL amb les instruccions necessàries per generar els objectes de la BD; bàsicament taules, seqüències, disparadors, procediments i funcions.

La primera "onada" d'implementació servirà per codificar el repertori de procediments i funcions estàndard que permetin el manteniment directe (altes, baixes, modificacions i consultes) de totes les taules de l'aplicació.

En una segona "onada" d'implementació ja s'abordaran problemes més concrets i específics, anomenats casos d'ús. Aquests casos d'ús es podran resoldre mitjançant funcionalitat estàndard o be potser necessiten de codificació "a mida" (i.e. noves funcions i procediments específics).

Finalment, la tercera onada d'implementació centrarà els seus esforços en resoldre l'apartat estadístic del TFG: resolució de les consultes proposades, actualització dels resultats estadístics, etc.

1.6.4 Capítol 5. Proves

Aquest capítol estarà dedicat a la prova i validació de tot el codi generat en la fase anterior d'implementació. Mitjançant proves unitàries, s'avaluaran exhaustivament tots i cada un dels procediments, funcions i disparadors originats en la primera "onada".

Un cop validats tots aquests components, es procedirà a fer el mateix amb tot el codi generat en la segona "onada",en el benentès que,per dur-la a terme, serà necessari emprar els artefactes obtinguts en la primera.

Durant aquesta fase de proves es procedirà també a originar un joc de dades suficientment gran i divers per permetre la resolució de tot l'apartat estadístic del projecte, que bàsicament consisteix en implementar una estructura de dades (taula estadística) que permeti donar resposta a una sèrie de consultes que s'han plantejat. Els registres d'aquesta taula s'obtindran a partir d'un procés d'actualització de dades estadístiques.

1.6.5 Capítol 6. Conclusions

El darrer capítol de la memòria està dedicat a presentar les conclusions obtingudes, un cop finalitzat el projecte. Es comenten aspectes relacionats amb el seguiment de la metodologia escollida, alteracions sofertes sobre el calendari previst, així com possibles bondats i carències del producte final obtingut.

2. Requisits

D'acord amb la definició que els professors Pradel i Raya proposen, els requisits "*expressen les necessitats i restriccions que afecten un producte de programari que contribueix a la solució d'un problema del món real*".[2]

Així, la primera tasca que s'ha de dur a terme a l'hora d'abordar el desenvolupament de programari propi és identificar exactament quins són aquests problemes als quals es vol donar solució. Aquesta és una fase especialment crítica de la metodologia en cascada, doncs una errònia o incompleta identificació de requisits pot tenir conseqüències molt negatives en les fases posteriors del projecte. Això és degut a que –contràriament a altres metodologies de tipus iteratiu– el cicle de vida en cascada és poc flexible davant possibles alteracions en els requisits inicials. Cal, doncs, parar màxima atenció en aquesta fase primerenca del projecte.

En aquest capítol es documentaran tots els requisits identificats, directa o indirectament, a partir de la informació recopilada. La gran majoria sorgeixen de forma immediata, a partir d'una lectura atenta de l'enunciat, però altres no són tan evidents i requereixen d'un cert esforç d'anàlisi.

A efectes classificatoris, en aquest treball s'adoptarà la distinció entre requisits funcionals i no funcionals que els professors Pradel i Raya ens proposen també a la seva obra abans mencionada.

2.1 Requisits funcionals

"*Els requisits funcionals fan referència a la funcionalitat que ha de proporcionar el sistema.*"[3]

Els requisits funcionals defineixen el comportament general d'un sistema, és a dir, **QUE** ha de fer davant una certa situació o estímul. Un cop definits tots els requisits funcionals d'un projecte, s'espera d'ells que siguin complerts i consistents, en el sentit que han de satisfer tota la funcionalitat inicialment prevista i no presentar contradiccions entre ells.

En la següent taula s'indiquen tots els requisits funcionals identificats en aquest projecte, així com una possible solució proposada per satisfer la necessitat que cada un d'ells planteja (excepte per a les consultes plantejades, que ja s'implementaran en una fase posterior).

[2] J.Pradell, J.Raya,"Requisits", pàg. 7 [1]
[3] J.Pradell, J.Raya,"Requisits", pàg. 9 [1]

ID Requisit	Descripció
RF01	El sistema ha de permetre definir sèus internacionals. Solució: s'implementarà una taula de països (*Countries*) i una taula de sèus (*Offices*), relacionades 1aN.
RF02	Qualsevol petició o incidència s'ha d'enregistrar dins del sistema en forma de ticket. Solució: s'implementarà una taula anomenada *Tickets*, que contindrà tots els atributs necessaris per emmagatzemar la informació rellevant de cada cas.
RF03	El sistema ha de contemplar 2 tipus de tickets: • Incidències • Peticions Solució: definir dins la taula *Tickets* un atribut *Type* que permeti emmagatzemar tots aquests valors diferents.
RF04	Cada *ticket* tindrà una sèrie de possibles estats ben diferenciats: • Creat (valor inicial per defecte) • Assignat • En progrés • Resolt • Cancel·lat Solució: definir dins la taula *Tickets* un atribut *State* que permeti emmagatzemar tots aquests valors diferents.
RF05	El sistema ha de permetre el seguiment o traçabilitat d'un *ticket* en llarg de tot el seu cicle de vida, des de que s'origina fins que es tanca o cancel·la, incloent canvis d'estat, escalat de nivell de dificultat, etc. Solució: s'implementarà una taula en concepte de diari de moviments de *tickets* (*Ticket Journal*), relacionada 1aN amb la taula *Tickets*, que contindrà una entrada per a cada canvi d'estat del ticket. Cada registre d'aquest diari contindrà l'identificador del *ticket*, la data/hora en que es produeix el canvi, l'identificador de l'agent que el du a terme i els valors anteriors i nous implicats en cada transacció.
RF06	El sistema ha de permetre enviar missatges als usuaris per informar-los de qualsevol esdeveniment associat amb el seu cas (canvi d'estat, elevació de nivell de dificultat, resolució i tancament, etc). Tots els missatges enviats i rebuts entre agents i usuaris han de quedar convenientment emmagatzemats. Solució: s'implementarà una taula anomenada *Messages,* que emmagatzemarà tots els missatges rebuts i enviats pel sistema, incloent remitent, destinatari, data d'enviament, etc.

ID Requisit	Descripció
RF07	El sistema ha de contemplar diversos canals d'entrada per enregistrar incidències i peticions. En concret: • Telèfon • Xat • Web Solució: definir dins la taula *Tickets* un atribut *Channel (o Source)* que permeti emmagatzemar tots aquests valors diferents.
RF08	El sistema ha de permetre que un ticket sigui donat d'alta per un operador humà (agent), quan el seu origen sigui telèfon o chat, o bé de manera automàtica, quan provingui d'una web. Solució: es crearà un camp anomenat *CreatedBy* a la taula *Tickets*, que serà clau externa contra una taula d'operaris (*Agents)*. Aquest camp romandrà a NULL quan l'origen del ticket sigui des de la web.
RF09	El sistema ha de permetre diferents nivells de suport (L1, L2, L3 i L4), en funció de la complexitat del problema plantejat en cada cas. L'escalat de dificultat serà progressiu, passant sempre d'un nivell inferior a un de superior; el darrer nivell de suport (L4) és especial per a casos molt difícils, que seran tractats com a projectes.. Solució: es crearà un camp anomenat Severity a la taula de Tickets, que serà numèric. S'afegiran totes les validacions necessàries perquè el valor d'aquest camp només es pugui incrementar de 1 en 1, fins a un màxim establert.
RF10	El sistema diferenciarà entre agents usuaris i agents experts, però només a primer nivell de suport. Per a nivells més elevats, ja tot seran experts. Solució: a la taula d'agents (Agents) es crearà un camp IsUser (S/N), per diferenciar els uns dels altres. Un agent que no sigui expert, només podrà intervenir en casos de nivell 1 de dificultat.
RF11	Els agents i els experts s'organitzaran en grups (=equips) de treball. Solució: es crearà una taula d'equips de treball, *Teams*. Cal decidir si un mateix agent pot pertànyer a diferents equips de treball.
RF12	Inicialment, només hi haurà 1 equip de suport per a cada nivell de dificultat (però no s'hauria de limitar el sistema en aquest sentit). Solució: es definirà un camp *SupportLevel* a la taula de *Teams* que definirà el nivell de suport que aquest pot oferir. Inicialment, només es definirà un equip per a cada nivell de suport, però res impedeix més endavant afegir-ne més.

ID Requisit	Descripció
RF13	Els agents que no siguin experts, només podran ser assignats a equips de suport de nivell 1 de dificultat. Solució: en el moment d'assignar un agent a un equip, comprovar el nivell de dificultat de l'equip i si l'agent és expert o no.
RF14	El sistema ha de mantenir un catàleg de serveis homologats, als quals s'assignarà un nivell de prioritat (P1, P2, P3 i P4) i un temps mig de resolució. Solució: es crearà una taula de serveis homologats anomenada Services que, entre altres, contindrà els camps *Priority* i *Average Resolution Time*.
RF15	El sistema només permetrà l'accés als usuaris registrats de la companyia. En aquest sentit, la dada clau serà el correu electrònic de l'usuari, la resta de les seves dades estaran emmagatzemades en una base de dades global de la companyia. Solució: s'implementarà un procés per validar que cada petició correspongui a un usuari registrat del sistema. També s'implementarà un procés per registrar convenientment aquests usuaris.
RF15	El sistema ha de poder donar resposta a la següent consulta: • Donat un mes qualsevol, temps mig de resolució d'incidències per cada tipus de prioritat. Solució: s'implementarà un mòdul que actualitzarà les dades d'una taula estadística. També existirà una consulta específica sobre aquesta taula per retornar els valors demanats en aquest cas.
RF16	El sistema ha de poder donar resposta a la següent consulta: • Donats un país i un mes concrets, percentatge de peticions resoltes per sobre del temps definit (tenint en compte totes les prioritats i el seu diferent temps de resolució). Solució: s'implementarà un mòdul que actualitzarà les dades d'una taula estadística. També existirà una consulta específica sobre aquesta taula per retornar els valors demanats en aquest cas.
RF17	El sistema ha de poder donar resposta a la següent consulta: • Usuari que, en el moment de realitzar la consulta, ha creat més tickets de tipus incidències. Solució: s'implementarà un mòdul que actualitzarà les dades d'una taula estadística. També existirà una consulta específica sobre aquesta taula per retornar els valors demanats en aquest cas.

ID Requisit	Descripció
RF18	El sistema ha de poder donar resposta a la següent consulta: • Top10 per seu de les peticiones més sol·licitades (tenint en compte totes les dades de que es disposi). Solució: s'implementarà un mòdul que actualitzarà les dades d'una taula estadística. També existirà una consulta específica sobre aquesta taula per retornar els valors demanats en aquest cas.
RF19	El sistema ha de poder donar resposta a la següent consulta: • Percentatge d'increment (o decrement) del número de tickets resolts respecte al mes anterior. Solució: s'implementarà un mòdul que actualitzarà les dades d'una taula estadística. També existirà una consulta específica sobre aquesta taula per retornar els valors demanats en aquest cas.
RF20	El sistema ha de poder donar resposta a la següent consulta: • Entrada del catàleg de serveis que ha sigut més sol·licitada. S'haurà de tenir en compte només els tickets de tipus petició i es considerarà totes les dades recollides en l'aplicació fina al moment de la consulta. Solució: s'implementarà un mòdul que actualitzarà les dades d'una taula estadística. També existirà una consulta específica sobre aquesta taula per retornar els valors demanats en aquest cas.
RF21	El sistema ha de poder donar resposta a la següent consulta: • Seu en la que, en l'any en curs, s'han creat un número major d'incidències. Solució: s'implementarà un mòdul que actualitzarà les dades d'una taula estadística. També existirà una consulta específica sobre aquesta taula per retornar els valors demanats en aquest cas.
RF22	El sistema ha de poder donar resposta a la següent consulta: • Donat un any concret, hores totals invertides en resolució de tickets pel nivell L3. Solució: s'implementarà un mòdul que actualitzarà les dades d'una taula estadística. També existirà una consulta específica sobre aquesta taula per retornar els valors demanats en aquest cas.
RF23	El sistema ha de poder donar resposta a la següent consulta: • Número d'incidències no resoltes pels 3 primers nivells de suport (L1-L3) i assignades al grup de Problem Management en l'any en curs. Solució: s'implementarà un mòdul que actualitzarà les dades d'una taula estadística. També existirà una consulta específica sobre aquesta taula per retornar els valors demanats en aquest cas.

ID Requisit	Descripció
RF24	El sistema ha de poder donar resposta a la següent consulta: • En l'any en curs, percentatge de missatges que no han sigut respostos. Es considerarà que un missatge no ha sigut respost quan, por un determinat ticket, només hi ha missatges enviats pel sistema. Solució: s'implementarà un mòdul que actualitzarà les dades d'una taula estadística. També existirà una consulta específica sobre aquesta taula per retornar els valors demanats en aquest cas.
RF25	El sistema ha de poder donar resposta a la següent consulta: • En un moment concret, número total de tickets no oberts. Es considera obert quan el seu estat no es ni Resolt ni Cancel·lat. Solució: s'implementarà un mòdul que actualitzarà les dades d'una taula estadística. També existirà una consulta específica sobre aquesta taula per retornar els valors demanats en aquest cas.
RF26	El sistema ha de poder donar resposta a la següent consulta: • En un mes i un país concrets, percentatge de tickets creats por cada un dels diferents canals d'entrada (telèfon, xat o web). Solució: s'implementarà un mòdul que actualitzarà les dades d'una taula estadística. També existirà una consulta específica sobre aquesta taula per retornar els valors demanats en aquest cas.

2.2 Requisits no funcionals

"Els requisits no funcionals fan referència a restriccions sobre el conjunt possible de solucions."[4]

Els requisits no funcionals estan relacionats amb **COM** s'han de fer les coses i acostumen a ser restriccions que venen determinades per la plataforma escollida per implementar la solució. Els requisits no funcionals tenen a veure amb paràmetres relacionats amb el rendiment del sistema, consideracions d'arquitectura i, en general, tot allò que -si bé no es poden enquadrar en l'apartat anterior-, també resulta necessari per poder portar a bon port un projecte de desenvolupament.

Per exemple, cap *stakeholder* ens demanarà que la inserció d'un nou registre o la resolució d'una consulta es facin dins d'un temps acceptable. Aquestes són característiques que ja es donen per implícites en qualsevol sistema i queden fora del seu negociat, però que el bon analista també ha de contemplar a l'hora de dissenyar la millor solució possible.

[4] J.Pradell, J.Raya,"Requisits", pàg. 9 [1]

A continuació, es detallen els principals requisits no funcionals detectats i una possible solució per satisfer la necessitat que aquests plantegen:

ID Requisit	Descripció
RNF01	L'aplicació proposada ha de donar servei a diferents sèus multinacionals arreu del món. Solució: s'adopta l'anglès com a llengua estandarditzada per a tot l'aplicatiu, tant a nivell intern de codificació com a nivell extern de presentació de missatges. •
RNF02	Es demana que tota la funcionalitat sigui auto continguda, és a dir, independent de l'aplicació client amb que s'accedeixi a la BD. Solució: tota la funcionalitat es codificarà amb PL/SQL i quedarà incorporada dins de la pròpia BD, aprofitant totes les característiques que ofereixi el producte en aquest sentit (*triggers*, *stored procedures*, funcions d'usuari, etc).
RNF03	La BD ha de ser escalable. Solució: Oracle és, per definició, escalable; per tant, no cal adoptar cap mesura especial en aquest sentit.
RNF04	A efectes de millorar el rendiment, s'ha demanat que els resultats de certes consultes (veure requisits funcionals) siguin accessibles en temps constant 1. Solució: s'implementarà una taula que emmagatzemarà els diferents resultats estadístics requerits per l'aplicació. Els valors d'aquesta taula s'actualitzaran cada cert temps i a demanda, mitjançant un procés específic d'actualització d'estadístiques. Aquest procés inclourà totes les operacions "costoses" (agrupacions, sumes, etc), que es pretenen evitar en la recuperació dels resultats.

3. Anàlisi i Disseny

En aquest tercer capítol es recullen totes les activitats relacionades amb la segona fase del cicle de vida en cascada. En aquest treball es seguirà la metodologia proposada pels professors Jordi Casas Roma i Xavier Burgués Illa en les seves respectives publicacions, corresponents als mòduls 2on. i 3er. de l'assignatura "Disseny de Bases de Dades" del Grau d'Enginyeria Informàtica de la UOC.

A l'hora de dissenyar una base de dades, els professors Casas i Burgués distingeixen entre 3 nivell d'abstracció diferents:

- El disseny conceptual
- El disseny lògic
- El disseny físic

Els dos primers dissenys –conceptual i lògic- pertanyen al negociat de la fase d'anàlisi, mentre que el tercer, el disseny físic, pertany a la fase d'implementació i, per tant, serà tractat en el següent capítol.

Així doncs, el principal objectiu d'aquesta fase serà l'obtenció dels artefactes que defineixin els dissenys conceptual i lògic de la BD que es vol implementar. Recordem que, en aquest punt -tot i que ja s'ha optat pel model relacional com a base teòrica d'aquest desenvolupament-, encara hi ha certa independència del SGBD triat (encara que ja sabem que aquest serà Oracle).

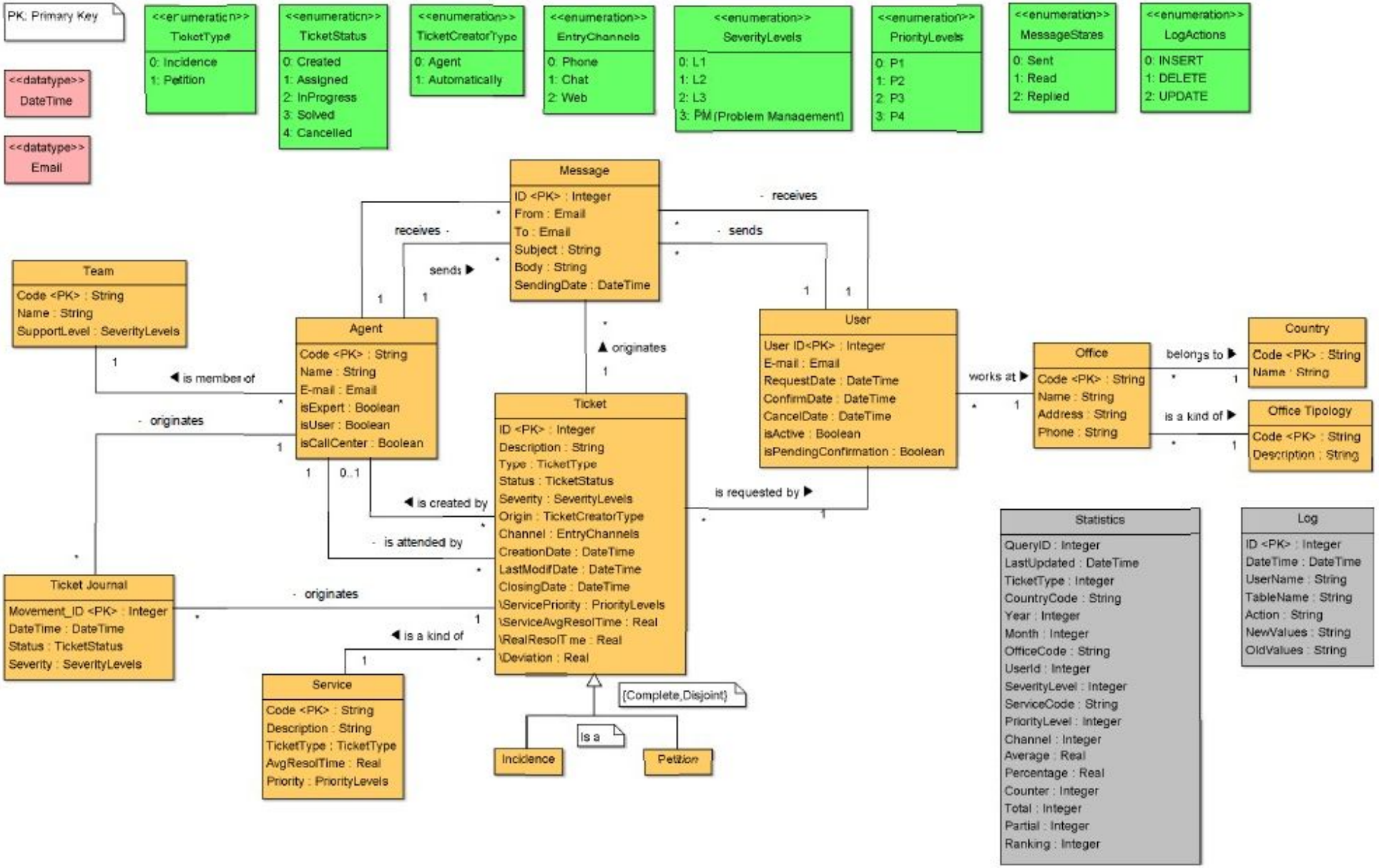
3.1 Disseny conceptual de la BD

"El disseny conceptual permet crear un esquema conceptual d'alt nivell i independent de la tecnologia d'implementació a partir de les especificacions i els requisits d'un problema del món real."[5]

L'artefacte, universalment acceptat, que millor defineix el disseny conceptual d'una BD és el Diagrama Entitat/Relació (DER), una eina modelada segons els estàndards UML que és capaç d'oferir molta informació en relativament molt poc espai. Gràcies al DER, amb un simple cop d'ull, el lector avesat podrà obtenir una magnífica visió de conjunt del problema que s'està tractant i la solució proposada.

Així, pel problema concret que ens ocupa, el DER proposat és el següent:

[5] J. Casas, "Disseny conceptual de bases de dades", pàg. 5 [3]


PK: Primary Key
<<datatype>>
DateTime
<<datatype>>
Email
<<enumeration>>
TicketType
0: Incidence
1: Petition
<<enumeration>>
TicketStatus
0: Created
1: Assigned
2: InProgress
3: Solved
4: Cancelled
<<enumeration>>
TicketCreatorType
0: Agent
1: Automatically
<<enumeration>>
EntryChannels
0: Phone
1: Chat
2: Web
<<enumeration>>
SeverityLevels
0: L1
1: L2
2: L3
3: PM (Problem Management)
<<enumeration>>
PriorityLevels
0: P1
1: P2
2: P3
3: P4
<<enumeration>>
MessageStates
0: Sent
1: Read
2: Replied
<<enumeration>>
LogActions
0: INSERT
1: DELETE
2: UPDATE
Message
ID <PK> : Integer
From : Email
To : Email
Subject : String
Body : String
SendingDate : DateTime
receives
sends
receives
sends
Team
Code <PK> : String
Name : String
SupportLevel : SeverityLevels
Agent
Code <PK> : String
Name : String
E-mail : Email
isExpert : Boolean
isUser : Boolean
isCallCenter : Boolean
is member of
originates
originates
User
User ID<PK> : Integer
E-mail : Email
RequestDate : DateTime
ConfirmDate : DateTime
CancelDate : DateTime
isActive : Boolean
isPendingConfirmation : Boolean
works at
Office
Code <PK> : String
Name : String
Address : String
Phone : String
belongs to
Country
Code <PK> : String
Name : String
is a kind of
Office Tipology
Code <PK> : String
Description : String
Ticket
ID <PK> : Integer
Description : String
Type : TicketType
Status : TicketStatus
Severity : SeverityLevels
Origin : TicketCreatorType
Channel : EntryChannels
CreationDate : DateTime
LastModifDate : DateTime
ClosingDate : DateTime
\ServicePriority : PriorityLevels
\ServiceAvgResolTime : Real
\RealResolTime : Real
\Deviation : Real
is created by
is attended by
is requested by
Ticket Journal
Movement_ID <PK> : Integer
DateTime : DateTime
Status : TicketStatus
Severity : SeverityLevels
originates
is a kind of
Service
Code <PK> : String
Description : String
TicketType : TicketType
AvgResolTime : Real
Priority : PriorityLevels
{Complete,Disjoint}
Is a
Incidence
Petition
Statistics
QueryID : Integer
LastUpdated : DateTime
TicketType : Integer
CountryCode : String
Year : Integer
Month : Integer
OfficeCode : String
UserId : Integer
SeverityLevel : Integer
ServiceCode : String
PriorityLevel : Integer
Channel : Integer
Average : Real
Percentage : Real
Counter : Integer
Total : Integer
Partial : Integer
Ranking : Integer
Log
ID <PK> : Integer
DateTime : DateTime
UserName : String
TableName : String
Action : String
NewValues : String
OldValues : String

En el DER podem distingir diferents elements:

- Tipus de dades (en color vermell), normalment es tracta de valors originats a partir d'una determinada composició de dades primàries (enters, caràcters, etc).

- Enumeracions (en color verd), també són un tipus especial de dades, formats per valors enumerables i discrets. Bàsicament, representen el domini de possibles valors que pot agafar un determinat atribut.

- Entitats (en color crema): normalment es corresponen amb les taules d'un SGBD i representen objectes o fets d'interès del nostre model, dels quals ens interessa emmagatzemar informació. Contenen atributs que defineixen les seves principals característiques. L'atribut que identifica de forma única cada ocurrència dins del seu conjunt s'identifica al diagrama com a clau primària (PK).

Atès que ens trobem en el nivell conceptual, encara no sabem com s'implementaran tots aquests elements a nivell físic. Per exemple, la majoria de SGBD suporten la representació de dates (*Date*), hores (*Time*) o fins i tot la combinació d'ambdós valors (DateTime), però no tots els SGBR permeten definir enumeracions, cosa que potser ens obligui a adoptar un plantejament diferent (per exemple, tractar-ho en forma de restricció, validant els possibles valors permesos en cada camp).

Un verb identifica cada una de les diferents relacions identificades entre entitats. La fletxa que acompanya a cada verb indica el sentit correcte en que s'ha de llegir, ajudant a entendre millor quina és la relació que es vol modelar exactament. Observis també que algunes entitats presenten més d'una relació entre elles, possiblement implicant atributs diferents en cada cas.

Per a cada relació s'indica també la seva cardinalitat, encara que pràcticament totes són del tipus 1aN (1..*). Només hi ha un cas en que no és així: la relació "...un *ticket* es creat per un agent..." pot tenir zero o una ocurrències en el cantó *Agent* de la relació, ja que els *tickets* creats automàticament des de la *web* mostraran un valor NULL com a representació de l'agent que els ha creat.

En el DER no s'han indicat les claus externes (*FK, Foreign Keys*), en el ben entès que aquestes ja apareixeran en el moment de realitzar el disseny lògic de la BD.

Hi ha un cas identificat d'especialització: les entitats *Incidence* i *Petition* són casos especialitzats de l'entitat *Ticket* i així s'ha reflectit al diagrama. En principi, es tracta d'una especialització completa (no hi ha cap altre tipus identificat de *ticket*) i disjunta (un *ticket* és una incidència o una petició, però no les dues coses a la vegada). Amb tot, el fet que no s'hagin identificat atributs específics per les peticions ni les incidències farà que segurament no originin cap taula en el nivell físic.

Les taules *Statistics* i *Log* són especials i per aquest motiu s'ha representat en un color diferent.

La taula *Statistics* es un artefacte creat a conveniència per poder implementar les 12 consultes estadístiques identificades en l'apartat de requeriments. Excepcionalment, aquesta taula no disposa de cap clau primària i tots els seus camps admeten valors nuls. Els registres d'aquesta taula no s'informen directament, sinó a partir d'uns procediments especials d'actualització estadística, que realitzen càlculs més o menys complexos. Existeix un procediment per actualitzar les dades de cada consulta. Un mateix camp pot tenir significat diferent segons la consulta de que formi part, però tampoc intervé necessàriament en totes les consultes (d'aquí la necessitat que admetin valors nuls).

Per la seva part, la taula *Log* conté un històric de totes les modificacions efectuades en forma d'inserció, esborrat o modificació de registres sobre les principals taules de l'aplicació. D'aquesta manera, és possible saber QUI ha modificat QUE i QUAN, ja que també es grava la data/hora i autor de cada moviment. Aquesta taula s'informa de manera automàtica a partir dels disparadors (triggers) d'inserció, esborrat i modificació que es desencadenes des de les taules monitoritzades. En cada cas, es grava una cadena de text amb els valors abans (*old values*) i després (*new values*) de cada operació.

Les úniques taules que no estan subjectes a *log* de monitorització son *Statistics* que, com ja s'ha dit, no s'informa amb valors entrats per cap usuari de l'aplicació, la pròpia taula de *Log* - Oracle 11g no permet recursivitat en els disparadors (*triggers*)- i *Ticket Journal*, que conceptualment també és una taula de *log*.

3.2 Disseny lògic de la BD

"El disseny lògic és una etapa intermèdia de les que componen el procés de desenvolupament d'una base de dades. Per tant, es parteix dels resultats d'una etapa prèvia (l'etapa del disseny conceptual) i se'n produeixen de nous que, al seu torn, serviran com a punt d'entrada d'una etapa posterior (el disseny físic)."[6]

Un cop finalitzat el disseny conceptual, el següent pas és procedir a elaborar el disseny lògic de la BD. En aquest estadi, tot i que encara no s'ha optat per cap SGBD concret, ja s'apliquen certs conceptes teòrics que limiten forçosament el ventall de productes disponibles per dur a terme la implementació física.

L'objectiu principal del disseny lògic és convertir els elements identificats durant la fase del disseny conceptual en estructures pròpies i identificables del model teòric escollit. En aquest sentit, hi han diverses opcions entre les quals triar:

- El model relacional
- El model jeràrquic
- El model de xarxa
- El model orientat a objectes
- El model documental
- El model transaccional
- Etc.

En aquest cas, s'opta per emprar el model relacional, per ser aquest un dels més estesos i universalment acceptats. Conceptes propis del model relacionals, com són les relacions, tuples i atributs, originaran posteriorment taules, files i columnes respectivament a nivell de disseny físic.

La notació emprada per construir el disseny lògic és la proposada pel professor Xavier Burgués[7]. Les principals característiques d'aquesta notació són:

- Les entitats del disseny conceptual esdevenen relacions en el model lògic relacional, que després originaran taules en el disseny físic. Les relacions s'indiquen amb una etiqueta identificativa, seguida dels seus atributs entre parèntesi.

relació(atribut-1, atribut-2, .., atribut-N)

[6] X. Burgués, "Disseny lògic de bases de dades", pàg. 7 [4]
[7] X. Burgués, "Disseny lògic de bases de dades" [4]

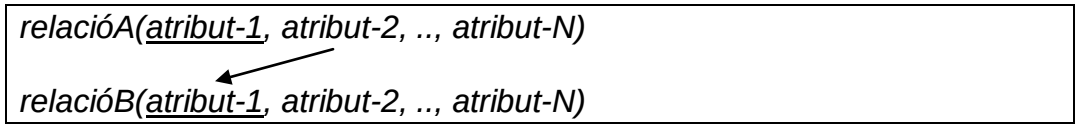
- Les claus primàries s'indiquen amb una línia contínua subratllant els atributs que la composen.

relació(atribut-1, atribut-2, .., atribut-N)

- Les claus alternatives -també anomenades candidates a clau primària- s'indiquen amb una línia discontínua subratllant els atributs que la composen.

relació(atribut-1, atribut-2, atribut3, .., atribut-1N)

- Les claus externes s'indiquen mitjançant una fletxa que apunta a la clau primària de la taula relacionada.



Tot i que, per a dissenys petits, aquesta notació és molt visual, pot resultar complicada de representar en cas que hi hagin masses relacions implicades. En aquests casos, una alternativa perfectament vàlida consisteix en indicar de manera textual quines són les claus externes i a quina taula fan referència.

relacióA(atribut-1, atribut-2, atribut-3, .., atribut-N)

atribut2 és clau externa de relacióB
atribut3 és clau externa de relacióC

- Els atributs que no poden agafar valor NULL s'indiquen en negreta.

*relació(atribut-1, **atribut-2**, .., atribut-N)*

Així doncs, es proposa el següent disseny lògic:

`Country(`**`Code`**`, `**`Name`**`)`

`OfficeTipology(`**`Code`**`, `**`Description`**`)`

`Office(`**`Code`**`, `**`Name`**`, `**`Address`**`, `**`Phone`**`, `**`CountryCode`**`, `**`TipologyCode`**`)`

`CountryCode is foreign key to Country`
`TipologyCode is foreign key to OfficeTipology`

Team(**Code**, **Name**, **SupportLevel**)

Service(**Code**, **Description**, **TicketType**, **AverageResolutionTime**, **Priority**)

User(**Id**, **E-mail**, **OfficeCode**,RequestDate, ConfirmDate, CancelDate, **isActive**, **PendingConfirm**)

OfficeCode is foreign key to Office

Agent(**Code**, **Name**, **E-mail**, **isUser**, UserId, **isExpert**, **TeamCode, isCallCenter**)

UserId is foreign key to User
TeamCode is foreign key to Team

Ticket(**Id**, **Description**, **Type**, **Status**, **Severity**, **Origin, Channel**,**CreationDate**, LastModifDate, ClosingDate, **UserId**, AgentCreatorCode, AgentAttendantCode,**ServiceCode**, ServicePriority, ServiceAvgResolTime,RealResolTime, Deviation)

UserId *is foreign key to User*
AgentCreatorCode is foreign key to Agent
AgentAttendantCode is foreign key to Agent
ServiceCode is foreign key to Service

TiketJournal(**Id**, **DateTime**, **TicketId**, Status, Severity, AgentCode, ServiceCode)

TicketId is foreign key to Ticket
AgentCode is foreign key to Agent
ServiceCode is foreign key to Service

Message(**Id**,**From**, **To**,**Subject**, **Body**, State, DateSent, ReplyTo, TicketId)

TicketId is foreign key to Ticket

Log(**Id**,**DateTime**, **UserName**, **TableName, Action,** NewValues, OldValues)

```
Statistics(QueryId, LastUpdated, TicketType, Country,
Year, Month, OfficeCode, UserId, Severity,
ServiceCode, Priority, Channel, Average, Percentage,
Counter, Total, Partial, Ranking)
```

Observis que en el disseny lògic proposat desapareixen les entitats *Incidence* i *Petition*. Tot i que conceptualment existeixen com a especialitzacions de l'entitat *Ticket*, en la pràctica no s'ha identificat cap atribut que els sigui propi i, per tant, no resulta necessari definir una taula a la BD per emmagatzemar les seves dades.

3.3 Normalització de taules

Un dels principals objectius del model relacional és que totes les taules resultants estiguin normalitzades, en major o menor grau. Cal recordar que existeixen diferents nivells de normalització, en funció del rigor teòric que es vulgui aplicar.

- Primera forma normal (1FN), una taula està en 1FN si
  - No presenta grups de repetició, tan a nivell d'atribut com de columna
  - Cada fila és única o, el que és el mateix, existeix una clau primària
  - Tots els seus atributs són atòmics.
- Segona forma normal (2FN), una taula està en 2FN si
  - Està en 1FN
  - Els atributs que no formen part de la clau primària depenen de la clau primària (i de tota la clau primària, si aquesta es composta).
- Tercera forma normal (3FN), una taula està en 2FN si
  - Està en 2FN
  - No existeixen dependències transitives entre atributs que no formen part de la clau primària

Tot i que existeixen formes normals més complexes, el més habitual és satisfer fins a la 2FN. En principi, aquest és el cas de totes les taules del disseny proposat. El fet d'haver escollit claus primàries relativament senzilles –en alguns casos, artificioses si es vol- ha ajudat notablement a assolir aquest objectiu.

Un cop completat el disseny lògic i normalitzades les taules, el proper pas és abordar el disseny físic de la BD. Aquesta tasca ja pertany a la fase d'implementació del projecte, que tractarem en el següent capítol.

4. Implementació

Aquesta nova fase del projecte es correspon amb el disseny físic de la BD i aquí ja abordarem la creació d'objectes dins d'un SGBDR determinat. En concret, Oracle 11g Express Edition, mitjançant el llenguatge PL/SQL que aquest incorpora. A nivell de llenguatge, cal diferenciar entre les instruccions que serveixen per a la definició de taules i altres objectes de la base de dades (DDL) de les instruccions per a la manipulació de les mateixes dades pròpiament (DML).

A l'hora de dur a terme el disseny físic de la BD i tenint en compte que aquesta aplicació està destinada a un àmbit internacional, sembla bona idea que, com a norma general, els camps de text descriptius suportin el joc de caràcters Unicode. Per aquest motiu, tots els camps de text seran definits de tipus NVARCHAR2, dada expressament dissenyada per suportar Unicode. D'aquesta manera, el sistema estarà preparat davant possibles eventualitats en que s'hagin d'introduir caràcter que pertanyen a altres idiomes (per exemple, noms d'empleats islandesos).

4.1 Nomenclatura

Algunes convencions de nomenclatura que s'han adoptat a l'hora d'abordar la implementació d'aquesta aplicació són:

- Oracle té una marcada preferència per convertir a majúscules tots els identificadors d'objectes (noms de taules, camps, *triggers*, *stored procedures*, etc). En aquest projecte, però, s'ha optat per treballar exclusivament amb identificadors en minúscules. Això obliga a un sobre esforç en la codificació, doncs cada referència a objecte s'ha d'emmarcar entre cometes dobles (i.e. "nom_objecte") però, a canvi, permet emprar paraules reservades del llenguatge com a identificadors i facilita enormement la identificació dels objectes propis del projecte, clarament diferenciats de la resta precisament per estar en minúscules.

- Addicionalment, amb l'objectiu de tenir tots els objectes del projecte alfabèticament agrupats, s'ha optat per identificar-los amb el prefix "gsd_". D'aquesta manera, apareixen de forma consecutiva a l'explorador d'objectes del *front-end*. Un altre avantatge d'aquest plantejament és que s'eviten possibles col·lisions de nom amb funcions i procediments d'altres paquets.

- Com a norma general, tots els paràmetres de funcions i procediments s'identifiquen amb el prefix "p_". La única excepció a aquesta regla és el paràmetre final RST, que emmagatzema el valor de retorn de cada funció o procediment. S'ha respectat aquest nom per coherència amb la documentació oficial del producte i també perquè així es recomana a l'enunciat del TFG.

- Les variable de tipus excepció (EXCEPTION), quan són necessàries per a tractament d'errors específics, es defineixen amb el prefix "e_".
- Les variables de tipus registre (%ROWTYPE) serveixen per fer referència a taules i, quan són necessàries, es defineixen amb el prefix "rec_". En alguns casos, per emfatitzar que són d'àmbit local, es fa servir també el prefix "lrec_".
- Per a la resta de variables, es fa servir el prefix "v_" i s'intenta que tinguin el mateix nom del camp al que fan referència.
- Com a norma general de nomenclatura per a tots els objectes, s'intenta seguir la notació del llenguatge C, que consisteix en emprar sempre minúscules i diferenciar amb un guió baix ("_") les diferents paraules d'un identificador (i.e. nom_de_variable). Aquesta sembla ser la notació més emprada per la comunitat de programadors PL/SQL, a jutjar de la documentació i els exemples consultats.

4.2 Enumeracions

En el disseny conceptual (DER) es van identificar una sèrie de camps que es podien resoldre mitjançant enumeracions. Bàsicament, una enumeració és una sèrie de nombres enters consecutius, identificats cada un d'ells per la seva corresponent etiqueta.

Nombre	Etiqueta
1	Literal 1
2	Literal 2
3	Literal 3
4	Literal 4

Les enumeracions són útils perquè permeten emprar el literal en lloc del valor numèric, contribuint a millorar enormement la llegibilitat del codi.

Malauradament, a diferència d'altres llenguatges, no existeix a PL/SQL cap tipus de data que emuli de forma nativa el comportament de les enumeracions. És possible, però, simular quelcom semblant a una enumeració mitjançant el tipus de dada INTEGER amb restriccions de valors permesos (CONSTRAINT) i codificant funcions de conversió de literal a nombre i a l'inrevés.

Exemple:

```
CREATE TABLE "test"(
...
"field" INTEGER NOT NULL,
...
CONSTRAINT "test_enum_01" CHECK ("field" BETWEEN 1 AND 5),
...
);
```

```
CREATE OR REPLACE FUNCTION “num_to_literal”
(p_num IN INTEGER)
RETURN VARCHAR2
AS
BEGIN
  IF    p_num=1 THEN RETURN('Literal 1');
  ELSIF p_num=2 THEN RETURN('Literal 2');
  ELSIF p_num=3 THEN RETURN('Literal 3');
  ELSIF p_num=4 THEN RETURN('Literal 4');
  ELSE               RETURN('????');
  END IF;
END;
```

```
CREATE OR REPLACE FUNCTION "literal_to_num"
(p_literal IN VARCHAR2)
RETURN INTEGER
AS
BEGIN
  IF    p_literal='Literal 1'  THEN RETURN(1);
  ELSIF p_literal=' Literal 2' THEN RETURN(2);
  ELSIF p_literal=' Literal 3' THEN RETURN(3);
  ELSIF p_literal=' Literal 4' THEN RETURN(4);
  ELSE                             RETURN(-1);
  END IF;
END;
```

Aquestes funcions sempre retornen algun valor - ja sigui numèric o literal-, fins i tot en el cas que no s’hagi subministrat un paràmetre correcte. La interpretació que es faci d’aquest valor es responsabilitat del nivell superior d’execució que fa la crida a la funció.

La instruccions necessàries per generar totes les funcions que manipulen enumeracions es troben dins del següent *script*:

- CREATE PROC AND FUNC

Per conveniència, es proveeix també el seu script antagonista, que esborra els objectes creats anteriorment:

- DROP PROC AND FUNC

En la següent taula s’enumeren tots els procediments (P) i funcions (F) relacionats amb el tractament d’enumeracions, incloent una breu explicació del que fan. La seva sintaxi completa, juntament amb exemples de com emprar-los, es poden consultar a l’Annex I d’aquesta memòria.

Type	Name	Action
F	gsd_enum_bool_literal( )	0 ⇨ N 1 ⇨ Y
F	gsd_enum_bool_value( )	N ⇨ 0 Y ⇨ 1
F	gsd_enum_severity_literal( )	1 ⇨ L1 2 ⇨ L2 3 ⇨ L3 4 ⇨ PM
F	gsd_enum_severity_value( )	L1 ⇨ 1 L2 ⇨ 2 L3 ⇨ 3 PM ⇨ 4
F	gsd_enum_priority_literal( )	1 ⇨ P1 2 ⇨ P2 3 ⇨ P3 4 ⇨ P4
F	gsd_enum_priority_value( )	P1 ⇨ 1 P2 ⇨ 2 P3 ⇨ 3 P4 ⇨ 4
F	gsd_enum_message_state_literal( )	1 ⇨ SENT 2 ⇨ REPL
F	gsd_enum_message_state_value( )	SENT ⇨ 1 REPL ⇨ 2
F	gsd_enum_ticket_type_literal( )	1 ⇨ INC 2 ⇨ PET
F	gsd_enum_ticket_type_value( )	INC ⇨ 1 PET ⇨ 2
F	gsd_enum_ticket_status_literal( )	1 ⇨ CREATED 2 ⇨ ASSIGNED 3 ⇨ IN PROGRESS 4 ⇨ SOLVED 5 ⇨ CANCELLED
F	gsd_enum_ticket_status_value( )	CREATED ⇨ 1 ASSIGNED ⇨ 2 IN PROGRESS ⇨ 3 SOLVED ⇨4 CANCELLED ⇨5
F	gsd_enum_ticket_origin_literal( )	1 ⇨ AGENT 2 ⇨ AUTOM
F	gsd_enum_ticket_origin_value( )	AGENT ⇨ 1 AUTOM ⇨ 2
F	gsd_enum_ticket_chanel_literal( )	1 ⇨ PHONE 2 ⇨ CHAT 3 ⇨ WEB
F	gsd_enum_ticket_chanel_value( )	PHONE ⇨ 1 CHAT ⇨ 2 WEB ⇨ 3

4.3 Seqüències auto numèriques

Els camps de tipus auto numèric són valors enters que s'incrementen automàticament amb cada nova inserció d'un registre dins d'una taula. Aquesta és una facilitat que ofereixen la majoria de SGBDR com a manera ràpida i senzilla d'obtenir una clau primària d'una taula –artificiosa, si es vol-, quan no hi ha cap altre camp que sigui bon candidat a ser-ho i tampoc es vol optar per definir claus primàries compostes.

Alguns SGBDR incorporen tipus de dades natius per implementar aquesta funcionalitat. No és el cas d'Oracle, que prefereix emprar el concepte de seqüències, un tipus d'objecte propi i diferenciat de la resta, que s'ha de vincular explícitament amb el camp que es vol incrementar.

Les seqüències incorporen dues funcions que resulten de gran utilitat a l'hora de simular el comportament d'un comptador seqüencial auto incrementat:

- *NEXTVAL*, retorna incrementat el següent valor de la seqüència; aquesta és la funció que s'ha d'emprar quan es vol assignar un nou valor per la clau primària.
- CURRVAL, retorna el valor actual de la seqüència, sense incrementar-lo; aquesta és la funció que s'ha d'emprar quan es vol saber quin ha estat el darrer valor assignat com a clau primària.

D'acord amb les recomanacions de la documentació oficial d'Oracle, el valor auto incremental s'ha d'assignar abans de fer la inserció.

Exemple:

```
--Define new sequence
CREATE SEQUENCE "sequence_name";
```

```
--Increment sequence value
CREATE OR REPLACE TRIGGER "table_name_before_trigger"
BEFORE INSERT ON " table_name"
FOR EACH ROW
BEGIN
  :new."id" := "sequence_name".nextval;
END;
```

```
--Get current sequence value
...
last_id := "sequence_name".currval;
...
```

La clàusula FOR EACH ROW del *trigger* és molt important, ja que volem que el valor de la seqüència s'incrementi per a cada nou registre inserit dins la taula.

En aquest projecte hi ha 5 taules que fan servir un camp auto numèric –que, per conveni, anomenarem "id"- com a clau primària:

- "gsd_log"
- "gsd_user"
- "gsd_message"
- "gsd_ticket"
- "gsd_ticket_journal"

4.4 Tipus referenciats

Una altra característica generalista d'Oracle que es farà servir intensament en llarg d'aquest projecte són els tipus referenciats.

Oracle suporta una sèrie de tips de dades bàsiques (nombre, cadenes de text, dates, etc) que es fan servir per definir els valors que han d'emmagatzemar els camps de les taules, així com també els paràmetres i variables que es fan servir dins del codi PL/SQL. Sovint, aquestes variables fan referència a valors de camps i passa que -si es modifica el tipus o mida del camp-, s'han de revisar totes aquestes referències.

Per exemple, si un camp passa de VARCHAR2(50) a VARCHAR2(100), s'haurien de revisar totes les variables que fan referència a aquest camp per actualitzar-les. Gràcies als tipus referenciats, això ja no serà necessari, doncs se li indica a la variable que adopti el mateix tipus de dada definit a nivell de camp, sigui el que sigui.

Exemple:

```
DECLARE
  variable_name  "table_name"."field_name"%TYPE;
```

Òbviament, aquesta és una característica molt interessant que protegeix el nostre codi davant possibles modificacions a nivell de DDL.

4.5 Disseny físic de la BD

La implementació física de la BD es correspon amb la definició i posterior compilació de tots els objectes necessaris per obtenir la funcionalitat desitjada. Recordem que els principals objectes amb que treballa Oracle 11g són:

- *Sequences* (seqüències): serveixen per emular el comportament d'una clau primària auto numèrica.
- *Tables* (taules), per emmagatzemar dades. Són l'element nuclear de qualsevol BD.
- *Constraints* (restriccions), per definir regles de negoci associades a possibles valors permesos o denegats dins d'un camp. Tot i que es poden definir per separat, estan tan estretament relacionades amb les taules que la seva definició s'acostuma a incloure dins d'aquestes.
- *Procedures* (procediment emmagatzemats): són blocs de codi PL/SQL que implementen regles de negoci no associades a cap esdeveniment d'inserció, esborrat o modificació de registres en particular. Poden rebre paràmetres d'entrada i sortida, però no retornen cap valor.
- *Functions* (funcions): ídem procediment emmagatzemats, però sí que poden retornar un valor.
- *Triggers* (disparadors), per emmagatzemar tota la lògica de negoci associada al fet d'inserir, esborrar o modificar les dades d'una taula, ja sigui a nivell general de registre o específic de camp.

L'ordre de creació dels objectes es significatiu: no es pot compilar un objecte que faci referència a un altre que encara no existeix dins de la BD. Per exemple, no es pot definir un trigger per a una taula que encara no s'hagi definit prèviament o compilar un trigger que invoqui a un procediment inexistent. En aquest sentit, un possible ordre lògic de creació d' objectes seria:

- Creació de seqüències.
- Creació de taules, incloent les seves restriccions (*constraints*).
- Creació de procediments i funcions.
- Creació de disparadors (*triggers).*

4.5.1 Seqüències

Com ja s'ha dit, les seqüències són necessàries per implementar la funcionalitat de les claus primàries auto numèriques. Atès que aquest objectes poden ser referenciats des de les taules o disparadors, haurien de ser els primers objectes a definir dins d'una BD.

En aquest projecte es fan servir les següents seqüències:

- *gsd_autonum_log*, per assignar valors a la PK de *gsd_log*.
- *gsd_autonum_user*, per assignar valors a la PK de *gsd_user*.
- *gsd_autonum_message*, per assignar valors a la PK de *gsd_message.*
- *gsd_autonum_ticket*, per assignar valors a la PK de *gsd_ticket.*
- *gsd_autonum_ticket_journal*, per assignar valors a la PK de *gsd_ticket_journal.*

En ocasions, pot resultar interessant restablir el comptador inicial d'una seqüència, ja sigui perquè es vol començar una nova bateria de proves o bé per posar la BD en explotació. En aquest cas, la manera més senzilla de fer-ho és esborrant i tornant a crear la seqüència.

Els scripts SQL que contenen les instruccions per crear i esborrar seqüències respectivament són:

- CREATE SEQUENCES
- DROP SEQUENCES

4.5.2 Taules

D'acord amb la teoria del model relacional, totes les taules incorporen una clau primària (*primary key*) i, allí on sigui necessari, també una clau externa (*foreign key*) per satisfer les regles d'integritat referencial identificades. En alguns casos, les claus primàries sorgeixen de forma natural del propi anàlisi, mentre que, en altres, s'han de forçar explícitament mitjançant seqüències auto numèriques. La única excepció a aquesta norma és la taula d'estadístiques ("gsd_stats") que, com es veurà més endavant, és especial i no incorpora cap clau primària.

Tot i que es poden definir per separat, la implementació de restriccions (CONSTRAINT) s'inclou dins de la pròpia definició de taules, atesa l'estreta relació que hi ha entre ambdós objectes. Com a norma habitual, es donarà un nom entenedor a cada restricció, evitant els identificadors per defecte assignats per Oracle, ja que aquests noms són els que apareixen en els missatges d'error quan es viola la regla d'integritat definida.

Es fan servir CONSTRAINT en 5 escenaris concrets:

- Per definir claus primàries (PK):

```
CREATE TABLE “test”(
...
“field_name” VARCHAR2(10),
...
CONSTRAINT “test_pk”
      PRIMARY KEY (“field_name”)
);
```

- Per definir claus externes (FK):

```
CREATE TABLE “test”(
...
“field_name” VARCHAR2(10),
...
CONSTRAINT “test_pk”
      FOREIGN KEY (“field_name”)
      REFERENCES “external_table”(“external_pk”)
);
```

- Per definir claus alternatives (UNIQUE):

```
CREATE TABLE “test”(
...
“field_name” VARCHAR2(10) UNIQUE,
...
);
```

- Per definir valors per defecte (DEFAULT):

```
CREATE TABLE “test”(
...
“field_name” INTEGER DEFAULT 1,
...
);
```

- Per simular el comportament d’una enumeració:

```
CREATE TABLE “test”(
...
“field_name” INTEGER,
...
CONSTRAINT “test_field_name_enum”
      CHECK(“field_name” BETWEEN 1 AND 4 )
);
```

Pel que fa a disseny de taules, s'han establert els següent convenis:

- Tots els camps que siguin clau primària es definiran com a VARCHAR2(10), excepte en aquelles taules on es faci servir un identificador auto numèric.
- Tots els camps que incloguin una nom o una descripció es normalitzen a VARCHAR2(100).
- Tots els camps de data es normalitzen a DATE; no es considera necessari emprar la precisió (superior) del tipus TIMESTAMP.
- Els camps que hagin de guardar el valor d'alguna "enumeració" es definiran de tipus INTEGER.

Els scripts SQL que contenen les instruccions necessàries per crear i esborrar taules respectivament són:

- CREATE TABLES
- DROP TABLES
- DELETE ALL DATA

En principi, només és necessari esborrar taules en la fase de desenvolupament, quan aquestes encara s'estan definint i convé regenerar-les per incloure nous camps i restriccions. En aquest cas, però, es recomanable buidar prèviament les dades que contenen o ens podem trobar amb problemes d'integritat referencial a l'hora d'esborrar alguna taula.

A continuació, s'indiquen totes les taules que s'han implementat en aquesta aplicació, incloent una breu explicació de les decisions de disseny significatives que s'hagin pres en cada cas.

4.5.2.1 Països (*gsd_country*)

Taula que conté els diferents països on hi han sèus de la empresa. Amb únicament dos camps, resulta quasi auto explicativa:

- *Code*: codi de país.
- *Name*:nom de país.

4.5.2.2 Tipologies d'oficines (*gsd_office_tipology*)

Taula que conté les diferents tipologies d'oficines aplicables a les sèus de la empresa. També amb dos únics camps:

- *Code*: codi de tipologia d'oficina
- *Description*: descripció de tipologia d'oficina

4.5.2.3 Oficines (*gsd_office*)

Taula que conté les diferents sèus de la empresa.

- *Code*: codi de l'oficina.
- *Name*: nom de l'oficina.
- Address: adreça de l'oficina.
- *Phone*: telèfon de contacte de l'oficina.
- *CountryCode*: codi del país on es troba ubicada l'oficina.
- *OfficeTipologyCode*: codi de tipologia aplicable a l'oficina.

4.5.2.4 Equips de support (*gsd_team*)

Taula que conté els diferents equips de suport de la empresa. Podem considerar que un equip és una agrupació d'agents de suport que treballen plegats. En principi, cada equip només dona suport a tickets d'un determinat nivell i, si no es capaç de resoldre un cas, el deriva cap a l'equip de suport del nivell immediatament superior. En principi, només hi haurà definit un equip de suport per a cada un dels 4 nivells de dificultat definits.

- *Code*: codi de l'equip de suport.
- *Name*:nom de l'equip de suport.
- SupportLevel: nivell de dificultat dels tickets que pot atendre un determinat equip de suport.

4.5.2.5 Serveis homologats (*gsd_service*)

Taula que conté els diferents serveis homologats del gestor d'incidències. Es definiran una sèrie de serveis homologats, tant per *tickets* de tipus petició com incidència. Cada un d'aquests serveis tindrà associada una prioritat i un temps promig de resolució.

- *Code*: codi del servei homologat.
- *Description*: descripció del servei homologat.
- *TicketType*: tipus de ticket al qual es aplicable el servei (incidència o petició).
- *AverageResolutionTime*: temps promig de resolució dels tickets d'aquest servei.
- *Priority*: prioritat del servei.

4.5.2.6 Usuaris registrats (*gsd_user*)

Taula que conté els diferents usuaris registrats en el gestor d'incidències. En lloc de la clau candidata *E-mail*, s'ha optat per definir una clau primària auto numèrica, que s'assignarà a cada usuari en el moment de completar el seu procés de registre (veure casos d'ús).

- *Id*: identificador numèric únic de cada usuari, assignat a partir d'una seqüència
- *Email*: e-mail únic de cada usuari registrat. Es validarà que sigui un e-mail corporatiu, que pertanyi al domini registrat de la companyia (anyname@**anycompany.com**).
- *OfficeCode*: codi de la sèu on treballa l'usuari registrat.
- RequestDate: data en que l'usuari realitza una petició d'alta al gestor d'incidències.
- ConfirmDate: data en que l'usuari confirma una petició d'alta al gestor d'incidències.
- *CancelDate*: data en que un usuari registrat es dona de baixa del gestor d'incidències..
- *isActive*: indicador booleà (Y/N) que indica si un usuari està actiu dins del sistema o s'ha donat de baixa.
- *PendingConfirm*: indicador booleà (Y/N) que indica si un usuari té pendent una confirmació per donar-se d'alta al gestor d'incidències.

4.5.2.7 Agents de suport (*gsd_agent*)

Taula que conté els diferents agents que donen suport a la resolució de tickets. Cada agent només pot pertànyer a un equip de suport i, per tant, donarà únicament suport als ticket de la dificultat assignada al seu equip.

- *Code*: codi de l'agent de suport.
- *Name*: nom de l'agent de suport.
- *Email*: e-mail de l'agent de suport. Es validarà que sigui un e-mail corporatiu, que pertanyi al domini registrat de la companyia (anyname@**anycompany.com**)
- *isExpert*: valor booleà (Y/N) que indica si l'agent és un expert.
- *isUser*: valor booleà (Y/N) que indica si l'agent és un usuari registrat (cas especial).
- *UserId*: en cas que l'agent sigui un usuari registrat, identificador únic d'aquest usuari.
- *TeamCode*: codi de l'equip de suport al que pertany l'agent. Condiciona el nivell de dificultat dels tickets que pot atendre l'agent.

4.5.2.8 Tickets (*gsd_ticket*)

Taula que conté tots els tickets donats d'alta en el gestor d'incidències. És la taula central de tot l'aplicatiu, sobre la que pivota pràcticament tota la funcionalitat i consultes estadístiques.

- *Id*: identificador únic de cada ticket, assignat a partir d'una seqüència
- *Description*: text descriptiu del ticket.
- *Type*: tipus de ticket; pot set "incidència" o "petició".
- *Status*: estat del ticket; pot ser "Creat", "Assignat", "En progrés", "Solucionat" o "Cancel·lat".
- *Severity*: nivell de dificultat del ticket; pot ser "L1", "L2", "L3" o "PM" (Problem Management).
- *Origin*: com s'ha creat el ticket; pot ser "Automàticament" o mitjançant la intervenció d'un "Agent".
- *Channel*: canal d'entrada del ticket; pot ser "Telèfon", "Chat" o "Web" i està condicionat per l'origen del ticket. Un ticket amb origen automàtic només pot ser creat des de la web, mentre que si el seu origen és un "agent", aleshores pot tenir com origen "telèfon" o "chat".
- *CreationDate*: data de creació del ticket.
- *LastModifDate*: data de la darrera modificació del ticket; inicialment tindrà valor nul i s'actualitzarà cada cop que es modifiqui l'estat d'un ticket.
- *ClosingDate*: data de tancament del ticket, ja sigui perquè s'ha solucionat o bé perquè s'ha cancel·lat. Inicialment tindrà valor nul.
- *UserId*: id de l'usuari que ha originat el ticket.
- AgentCreatorCode: codi de l'agent que ha creat el ticket; ha de ser un valor nul si el ticket s'ha creat automàticament mitjançant la web.
- *AgentAttendantCode*: codi de l'agent que assumeix la resolució del ticket. El agent assignat ha de pertànyer al equip de support de la dificultat indicada pel propi ticket.
- ServiceCode: codi de servei associat al ticket; està condicionat pel tipus de ticket, doncs hi ha servei diferenciats en funció
- ServicePriority: camp calculat, que hereta el seu valor de del camp "Priority" del servei seleccionat pel ticket.
- *ServiceAvgResolTime*: camp calculat, que hereta el seu valor de del camp "AverageResolutionTime" del servei seleccionat pel ticket.
- *RealResolTime*: temps real de resolució del ticket; el seu valor surt de calcular la diferència entre la data de creació i la data de resolució del ticket.
- *Deviation*:
- *Comments*: camp de text lliure per introduir comentaris.

4.5.2.9 Missatges (*gsd_message*)

Taula que conté totes les interaccions –en forma de missatge- entre els usuaris que registren *tickets* i els agents que atenen la seva resolució. L'abast d'aquest projecte no recull la implementació de l'enviament físic dels missatges a través de cap canal, només el seu emmagatzemament dins d'aquesta taula, a manera d'històric.

- *Id*: identificador únic de cada missatge, assignat a partir d'una seqüència.
- *From:* e-mail de la persona que envia el missatge. Es valida que sigui un agent o un usuari registrat.
- *To*: e-mail de la persona a qui va destinat el missatge. Es valida que sigui un agent o un usuari registrat.
- *Subject*: assumpte (breu descripció) del missatge.
- *Body*: cos (text ampliat) del missatge.
- *State*: estat del missatge; pot ser "Enviat" o "Contestat"
- *DateSent*: data d'enviament del missatge.
- *ReplyTo*: identificador únic del missatge al qual es dona resposta.
- *TicketId*: identificador únic del *ticket* associat a cada missatge.

4.5.2.10 Moviments de tickets (*gsd_ticket_journal*)

La taula de moviments de tickets és especial. La seva filosofia és similar a la taula de *log*, comentada més endavant. Bàsicament, es tracta d'un històric que emmagatzema totes les modificacions (canvis d'estat, escalat de nivell de dificultat, etc) que ha sofert un *ticket* en llarg de la seva vida útil. Certament, aquesta mateixa informació es podria obtenir, de forma més genèrica, a partir de les entrades de la taula *log*, però en la fase de disseny s'ha considerat que aquestes dades eren prou rellevants com per tenir una taula pròpia.

- *Id*: identificador únic de cada moviment de ticket, assignat a partir d'una seqüència.
- *DateTime:* data i hora en que es genera el moviment.
- *TicketId*: identificador únic del *ticket* al que pertany el moviment generat.
- *Status*: nou estat assignat al ticket, reflectit en el moviment generat.
- *Severity*: nou nivell de dificultat assignat al ticket, reflectit en el moviment generat.
- *AgentCode*: nou agent assignat per resoldre el ticket, reflectit en el moviment generat.

4.5.2.11 Log (*gsd_log*)

La taula de *log* és especial, en el sentit que no emmagatzema dades pertanyents a cap entitat significativa del sistema, sinó que conté un històric dels canvis realitzats sobre la resta d'entitats. En aquest sentit, a diferència de les altres, la taula de log no s'informa directament sinó a partir de disparadors (*triggers*), que s'executen automàticament en resposta a alguna de les possibles 3 situacions que impliquen alteració de les dades:

- Alta o inserció d'un nou registre.
- Modificació d'un registre existent.
- Esborrat d'un registre existent.

En principi, a la taula de *log* només s'han d'inserir registres. Com ja s'ha dit, això succeeix de forma automàtica gràcies als disparadors, tot i que res impedeix modificar o esborrar registres de forma manual (sempre que es tinguin els permisos adequats per fer-ho).

Cada entrada de la taula de *log* documenta un canvi (alta, baixa o modificació) dut a terme sobre alguna de les taules monitoritzades. Concretament, el canvi és pot esbrinar per comparació de les diferències entre els valors vells i els valors nous.

Els camps de la taula *log* són:

- *Id*: identificador únic de cada moviment de *log*, assignat a partir d'una seqüència.
- *DateTime:* data i hora en que es produeix el canvi.
- *UserName*: usuaris que provoca el canvi.
- *TableName*: nom de la taula que sofreix el canvi.
- *Action*: acció concreta que provoca el canvi; pot ser "Alta" (INSERT), "Baixa" (DELETE) o "Modificació" (UPDATE). En principi, la selecció de registres (SELECT) no es considera que impliqui cap modificació de dades.
- *NewValues*: relació dels nous valors de cada camp del registre després d'una modificació, concatenats mitjançant un caràcter separador (punt i coma). En el cas d'esborrat (DELETE), el valor d'aquest camp serà NUL, atès que no existeixen nous valors (simplement s'esborren els vells). En principi, no s'inclou el nom del camp, tot i que aquesta podria ser una possible millor per ajudar a millorar la llegibilitat.

```
value-1;value-2;...;value-N
```

- *OldValues*: relació dels valors anterior de cada camp del registre després d'una modificació, concatenats mitjançant un caràcter separador (punt i coma). En el cas d'inserció d'un nou registre (INSERT), el valor d'aquest camp serà NUL, atès que no existeixen valors anteriors (simplement s'insereixen els nous). Mateixes consideracions que en el cas anterior pel que fa als noms dels camps.

4.5.2.12 Estadístiques (*gsd_stats*)

Aquesta taula també és especial i està dissenyada específicament per emmagatzemar una sèrie de resultats que es calculen mitjançant uns processo d'actualització estadística. Tots els seus camps són de propòsit general, en el sentit que poden intervenir en més d'una consulta, però també han d'admetre el valor NULL, per quan no formen part d'alguna d'elles.

Excepcionalment, aquesta taula no disposa de clau primària ni de procediments ABM (alta, baixa, modificació). Com ja s'ha dit, els registres d'aquesta taula s'insereixen a partir d'uns processos especials d'actualització estadística, que també són els encarregats d'esborrar els seus corresponents valors previs. El funcionament d'aquesta taula s'explicarà amb més detall en l'apartat de consultes estadístiques.

Els camps de la taula *log* són:

- *QueryId*: identificador numèric de la consulta a la que pertanyen cada registre d'aquesta taula; pot haver-hi varis registres dins la taula que pertanyin a la mateixa consulta (no hi ha restricció de clau primària). De moment, hi ha 12 consultes identificades, tot i que podrien haver-hi més en el futur.
- LastUpdate: data de la darrera modificació dels valors de cada registre o, el que és el mateix, darrera vegada que es va executar el procediment per actualitzar els valors estadístics d'aquella consulta.
- *TicketType*: camp que conté els diferents tipus de tickets que existeixen (Incidència o Petició). S'utilitza com a concepte d'agrupació en algunes consultes.
- *Country*: camp que conté codis de país. S'utilitza com a concepte d'agrupació en algunes consultes.
- *Year*: número de l'any, que es pot extreure de la data de creació o bé de la data de tancament de cada ticket, segons la problemàtica que s'estigui tractant. S'utilitza com a concepte d'agrupació en algunes consultes.
- *Month*: número de mes de l'any, que es pot extreure de la data de creació o bé de la data de tancament de cada ticket, segons la problemàtica que s'estigui tractant. S'utilitza com a concepte d'agrupació en algunes consultes.

- *OfficeCode*: camp que conté codis de sèus. S'utilitza com a concepte d'agrupació en algunes consultes.
- *UserId*: camp que conté identificadors d'usuaris registrat. S'utilitza com a concepte d'agrupació en algunes consultes.
- *Severity*: camp que conté nivells de dificultat (L1, L2, L3 o PM) dels *tickets*. S'utilitza com a concepte d'agrupació en algunes consultes.
- *ServiceCode*: camp que conté codis de serveis homologats. S'utilitza com a concepte d'agrupació en algunes consultes.
- *Priority*: camp que conté prioritats (P1, P2, P3 o P4) de *tickets*. S'utilitza com a concepte d'agrupació en algunes consultes.
- *Channel*: camp que conté canals d'entrada (Telèfon, Chat o Web) de *tickets*. S'utilitza com a concepte d'agrupació en algunes consultes.
- *Average*: camp genèric, reutilitzable en totes aquelles consultes que necessitin presentar un valor promig com a resultat d'algun dels seus càlculs.
- *Percentage*: camp genèric, reutilitzable en totes aquelles consultes que necessitin presentar un valor de tipus percentatge (%) com a resultat d'algun dels seus càlculs.
- *Counter*: camp genèric, reutilitzable en totes aquelles consultes que necessitin presentar el recompte d'algun concepte com a resultat d'algun dels seus càlculs.
- *Total*: camp genèric, reutilitzable en totes aquelles consultes que necessitin presentar un sumatori total com a resultat d'algun dels seus càlculs.
- *Partial*: camp genèric, reutilitzable en totes aquelles consultes que necessitin presentar un sumatori parcial com a resultat d'algun dels seus càlculs.
- *Ranking*: camp genèric, reutilitzable en totes aquelles consultes que necessitin presentar un valor de tipus *ranking* (i.e., posició dins d'una classificació) com a resultat d'algun dels seus càlculs.

4.5.3 Procediments i funcions

Els procediments i funcions són blocs de codi PL/SQL que no estan associats a cap esdeveniment en concret, és a dir, no s'executen automàticament sinó a demanda, invocant-los pel seu nom. La diferència entre un procediment i una funció és que la segona retorna explícitament un valor.

Sota la denominació ABM (alta, baixa, modificació) s'agrupen una sèrie de procediments i funcions que aglutinen tota la funcionalitat necessària per dur a terme un manteniment bàsic de dades. En principi, amb les 3 funcionalitats bàsiques citades (ABM) ha de ser suficient, però l'autor d'aquest TFG ha cregut convenient implementar-ne dues addicionals:

- *get()*, per accedir a les dades d'un registre, normalment mitjançant la seva clau primària.
- *show()*, per imprimir per pantalla les dades d'un registre.

Així, exceptuant certs casos especials -que més endavant s'explicaran- per a cada taula de l'aplicació es defineix un repertori de 5 procediments i funcions bàsics:

- "gsd_*tablename*_insert":
  - Procediment per inserir un nou registre.
  - Paràmetres d'entrada: valors dels camps a inserir.
  - Paràmetres de sortida: resultat de l'operació (RST).
- "gsd_*tablename*_delete":
  - Procediment per esborrar un registre.
  - Paràmetres d'entrada: clau primària del registre a esborrar.
  - Paràmetres de sortida: resultat de l'operació (RST).
- "gsd_*tablename*_update":
  - Procediment per modificar un registre.
  - Paràmetres d'entrada: clau primària del registre a modificar i nous valors de la resta de camps.
  - Paràmetres de sortida: resultat de l'operació (RST).
- "gsd_*tablename*_get":
  - Funció per accedir a les dades d'un registre.
  - Paràmetres d'entrada: clau primària del registre a esborrar.
  - Paràmetres de sortida: resultat de l'operació (RST).
  - Valor de retorn: registre trobat (o NULL, si no existeix).
- "gsd_*tablename*_show":
  - Funció per mostrar en pantalla les dades d'un registre.
  - Paràmetres d'entrada: clau primària del registre a mostrar.
  - Paràmetres de sortida: resultat de l'operació (RST).

Tal i com diu l'enunciat del TFG, tots els procediments/funcions han d'incorporar control d'excepcions i retornar sempre algun codi indicatiu del resultat de les seves operacions. La manera d'implementar aquest requeriment és la següent:

- Tots els procediments/funcions rebran com a últim paràmetre de sortida (OUT) una variable anomenada RST. Aquesta variable s'informarà dins del cos principal del procediment/funció amb el resultat de l'operació que, si no es produeix cap anomalia, ha de culminar amb èxit.
- S'inclourà dins del cos principal del procediment/funció un apartat específic de tractament d'excepcions, on s'informarà la variable RST amb un codi d'error en cas d'excepció.

Exemple:

```
CREATE OR REPLACE PROCEDURE "procedure_name"
(
  ...
  RST             OUT VARCHAR2
)
AS
BEGIN
  ...
  RST := 'OK';
EXCEPTION
  WHEN OTHERS THEN
    RST := SQLERRM;
END;
```

La instruccions necessàries per generar tots els procediments i funcions d'aquest projecte –excepte els relacionats amb el motor estadístic- es troben dins del següent *script*:

- CREATE PROC AND FUNC

Per conveniència, es proveeix també el seu script antagonista, que esborra tots objectes creats anteriorment:

- DROP PROC AND FUNC

En la següent taula es presenten ordenats alfabèticament tots els procediments (P) i funcions (F) d'aquest projecte (excepte els relacionats amb les enumeracions, que ja s'han vist en l'apartat corresponent). En cada cas, s'indica una breu explicació del que fa cada mòdul.

Type	Name	Action
P	*gsd_agent_delete( )*	Esborra un registre de la taula *gsd_ agent*.
F	*gsd_agent_exists( )*	Indica si un agent existeix a la taula *gsd_agent.*
F	*gsd_agent_get( )*	Accedeix a un registre de la taula *gsd_ agent*.
F	*gsd_agent_get_email( )*	Retorna el e-mail d'un agent registrat a la taula *gsd_agent*
P	*gsd_agent_insert( )*	Insereix un nou registre a la taula *gsd_agent*.
P	*gsd_agent_show( )*	Mostra les dades d'un registre de la taula *gsd_ agent*.
P	*gsd_agent_update( )*	Actualitza un registre de la taula *gsd_ agent*.
P	*gsd_country_delete( )*	Esborra un registre de la taula *gsd_country*.
F	*gsd_country_get( )*	Accedeix a un registre de la taula *gsd_country*.
P	*gsd_country_insert( )*	Insereix un nou registre a la taula *gsd_country*.
P	*gsd_country_show( )*	Mostra les dades d'un registre de la taula g*sd_country*.
P	*gsd_country_update( )*	Actualitza un registre de la taula *gsd_country*.
P	*gsd_log_insert( )*	Insereix un nou registre a la taula *gsd_log*.
P	*gsd_message_delete( )*	Esborra un registre de la taula *gsd_ message*.
F	*gsd_message_exists( )*	Indica si un missatge existeix a la taula *gsd_message*.
F	*gsd_message_get( )*	Accedeix a un registre de la taula *gsd_ message*.
P	*gsd_message_insert( )*	Insereix un nou registre a la taula *gsd_message*.
P	*gsd_message_mark_as_replied( )*	Marca un missatge de la taula *gsd_message* com a contestat.
P	*gsd_message_reply( )*	Contesta a un missatge de la taula *gsd_message*.
P	*gsd_message_send( )*	"Envia" un missatge (és a dir, l'insereix a la taula *gsd_message*).
P	*gsd_message_show( )*	Mostra les dades d'un registre de la taula *gsd_ message*.

Type	Name	Action
P	*gsd_message_update( )*	Actualitza un registre de la taula *gsd_ message*.
P	*gsd_office_delete( )*	Esborra un registre de la taula *gsd_ office*.
F	*gsd_office_exists( )*	Indica si un registre de la taula gsd_office existeix o no.
F	*gsd_office_get( )*	Accedeix a un registre de la taula *gsd_ office*.
P	*gsd_office_insert( )*	Insereix un nou registre a la taula *gsd_office*.
P	*gsd_office_show( )*	Mostra les dades d'un registre de la taula *gsd_ office*.
P	*gsd_office_tipology_delete( )*	Esborra un registre de la taula *gsd_ office_tipology*.
F	*gsd_office_tipology_get( )*	Accedeix a un registre de la taula *gsd_ office_tipology*.
P	*gsd_office_tipology_insert( )*	Insereix un nou registre a la taula *gsd_office_tipology*.
P	*gsd_office_tipology_show( )*	Mostra les dades d'un registre de la taula *gsd_ office_tipology*.
P	*gsd_office_tipology_update( )*	Actualitza un registre de la taula *gsd_ office_tipology*.
P	*gsd_office_update( )*	Actualitza un registre de la taula *gsd_ office*.
P	*gsd_service_delete( )*	Esborra un registre de la taula *gsd_ service*.
F	*gsd_service_get( )*	Accedeix a un registre de la taula *gsd_ service*.
P	*gsd_service_insert( )*	Insereix un nou registre a la taula *gsd_service*.
P	*gsd_service_show( )*	Mostra les dades d'un registre de la taula *gsd_ service*.
P	*gsd_service_update( )*	Actualitza un registre de la taula *gsd_ service*.
P	*gsd_team_delete( )*	Esborra un registre de la taula *gsd_ team*.
F	*gsd_team_get( )*	Accedeix a un registre de la taula *gsd_ team*.
P	*gsd_team_insert( )*	Insereix un nou registre a la taula *gsd_team*.
P	*gsd_team_show( )*	Mostra les dades d'un registre de la taula *gsd_ team*.
P	*gsd_team_update( )*	Actualitza un registre de la taula *gsd_ team*.

Type	Name	Action
P	*gsd_ticket_assign_agent( )*	Assigna un agent de suport a un ticket
P	*gsd_ticket_cancel( )*	Cancel·la un ticket
P	*gsd_ticket_close( )*	Marca un ticket com a resolt.
F	*gsd_ticket_create( )*	Crea un nou ticket (és a dir, l'insereix a la taula *gsd_ticket*).
P	*gsd_ticket_delete( )*	Esborra un registre de la taula *gsd_ticket*.
F	*gsd_ticket_get( )*	Accedeix a un registre de la taula *gsd_ticket*.
P	*gsd_ticket_in_progress( )*	Marca un ticket com
P	*gsd_ticket_insert( )*	Insereix un nou registre a la taula *gsd_ticket*.
P	*gsd_ticket_journal_insert( )*	Insereix un nou registre a la taula *gsd_ticket_journal*.
P	*gsd_ticket_scale_up( )*	Eleva un ticket al següent nivell de suport
P	*gsd_ticket_show( )*	Mostra les dades d'un registre de la taula *gsd_ticket*.
P	*gsd_ticket_update( )*	Actualitza un registre de la taula *gsd_ticket*.
P	*gsd_user_cancel_registration( )*	Cancel·la a una petició d'alta al servei GSD per part d'un usuari.
P	*gsd_user_confirm_registration( )*	Confirma una petició d'alta al servei GSD per part d'un usuari.
P	*gsd_user_delete( )*	Esborra un registre de la taula *gsd_ user*.
F	*gsd_user_exists( )*	Indica si un usuari existeix a la taula *gsd_user.*
F	*gsd_user_get_by_email( )*	Retorna les dades d'un registre de la taula *gsd_user* a partir del seu e-mail.
F	*gsd_user_get_by_id( )*	Retorna les dades d'un registre de la taula *gsd_user* a partir del seu identificador.
F	*gsd_user_get_email( )*	Retorna el e-mail d'un usuari registrat a la taula *gsd_user*
P	*gsd_user_insert( )*	Insereix un nou registre a la taula *gsd_user*.
P	*gsd_user_mark_as_active( )*	Marca un usuari de la taula *gsd_user* com actiu.
P	*gsd_user_request_registration( )*	Formalitza una petició d'alta al servei GSD per part d'un usuari.

Type	Name	Action
P	*gsd_user_show( )*	Mostra les dades d'un registre de la taula *gsd_ service*.
P	*gsd_user_update( )*	Actualitza un registre de la taula *gsd_ user*.

La sintaxi completa de cada procediment/funció, juntament amb exemples de com invocar-los, es poden consultar a l'Annex I d'aquesta memòria.

4.5.4 Disparadors (*triggers*)

Els disparadors o *triggers* són procediments de codi PL/SQL especials, que s'executen de manera automàtica en resposta a un determinat esdeveniment o succés. A nivell de taula, els principals esdeveniments reconeguts per Oracle són:

- BEFORE INSERT: abans d'inserir un nou registre.
- BEFORE DELETE: abans d'esborrar un registre existent.
- BEFORE UPDATE: abans de modificar un registre existent
- AFTER INSERT: després d'inserir un nou registre.
- AFTER DELETE: després d'esborrar un registre existent
- AFTER UPDATE: després de modificar un registre existent

Els *triggers* resulten de gran utilitat en diversos escenaris; concretament per aquest projecte podem identificar els següents:

- Per inserir una entrada en el fitxer de log ("gsd_log"), cada cop que es produeix una inserció, esborrat o modificació de registre en alguna de les taules que volem monitoritzar dins del control de canvis.
- Per incrementar una clau primària auto numèrica abans d'inserir un nou registre en alguna de les taules que fan servir aquesta característica ("gsd_user", "gsd_message", "gsd_ticket" i "gsd_ ticket_journal").
- Per inserir una entrada en el fitxer de moviments de ticket ("gsd_ticket_journal"), cada cop que es produeix un canvi d'estat en algun ticket, de forma similar a com es fa amb la taula gsd_log.
- Per enviar automàticament un missatge a l'interessat cada cop que es produeix un canvi d'estat en algun ticket. Recordem que, dins del context que ens ocupa, "enviar" un missatge vol dir inserir un registre dins la taula de missatges.

Tots i que en Oracle es possible codificar un *trigger* per a cada situació, resulta molt més compacte implementar un únic disparador que gestioni

els 3 esdeveniments (INSERT, DELETE, UPDATE) i discriminar dins del cos del mateix en funció de l'operació que l'ha desencadenat.

Exemple:

```
CREATE OR REPLACE TRIGGER "trigger_name"
AFTER INSERT OR UPDATE OR DELETE
ON "table_name"
FOR EACH ROW
BEGIN
  IF INSERTING THEN
    //specific actions on INSERT
  ELSIF DELETING THEN
    //specific actions on DELETE
  ELSIF UPDATING THEN
    //specific actions on UPDATE
  END IF;
END;
```

La instruccions necessàries per generar tots els *triggers* d'aquest projecte es troben dins del següent *script*:

- CREATE TRIGGERS

Per conveniència, es proveeix també el seu script antagonista, que esborra tots els *triggers* creats anteriorment:

- DROP TRIGGERS

En la següent taula es presenten ordenats alfabèticament tots els *trigger* que ha calgut implementar en aquest projecte. En cada cas, s'indica una breu explicació del que fa cada *trigger* i a quina taula afecta.

Trigger	**Event**	**Action**
gsd_agent_after_trigger()	After Insert	Insereix un registre a la taula *gsd_log*.
	After Update	Insereix un registre a la taula *gsd_log*.
	After Delete	Insereix un registre a la taula *gsd_log*.
gsd_agent_before_trigger()	Before Insert	Regles de negoci
	Before Update	Regles de negoci
gsd_country_after_trigger()	After Insert	Insereix un registre a la taula *gsd_log*.
	After Update	Insereix un registre a la taula *gsd_log*.
	After Delete	Insereix un registre a la taula *gsd_log*.
gsd_log_before_trigger()	Before Insert	Incrementa la clau primària auto numèrica.

Trigger	Event	Action
gsd_message_after_trigger()	After Insert	Insereix un registre a la taula *gsd_log*.
	After Update	Insereix un registre a la taula *gsd_log*.
	After Delete	Insereix un registre a la taula *gsd_log*.
gsd_message_before_trigger()	Before Insert	Incrementa la clau primària auto numèrica.
gsd_office_after_trigger()	After Insert	Insereix un registre a la taula *gsd_log*.
	After Update	Insereix un registre a la taula *gsd_log*.
	After Delete	Insereix un registre a la taula *gsd_log*.
gsd_office_tipo_after_trigger()	After Insert	Insereix un registre a la taula *gsd_log*.
	After Update	Insereix un registre a la taula *gsd_log*.
	After Delete	Insereix un registre a la taula *gsd_log*.
gsd_service_after_trigger()	After Insert	Insereix un registre a la taula *gsd_log*.
	After Update	Insereix un registre a la taula *gsd_log*.
	After Delete	Insereix un registre a la taula *gsd_log*.
gsd_team_after_trigger()	After Insert	Insereix un registre a la taula *gsd_log*.
	After Update	Insereix un registre a la taula *gsd_log*.
	After Delete	Insereix un registre a la taula *gsd_log*.
gsd_ticket_after_trigger()	After Insert	Insereix un registre a la taula *gsd_log*. Insereix un registre a la taula *gsd_ticket_journal*. Envia un missatge de notificació a l'usuari propietari del ticket.
	After Update	Insereix un registre a la taula *gsd_log*. *gsd_ticket_journal*. Envia un missatge de notificació a l'usuari propietari del ticket.
	After Delete	Insereix un registre a la taula *gsd_log*.

Trigger	Event	Action
gsd_ticket_after_trigger()	Before Insert	Incrementa la clau primària auto numèrica.
gsd_ticket_journal_after_trigger()	Before Insert	Incrementa la clau primària auto numèrica.
gsd_user_after_trigger()	After Insert	Insereix un registre a la taula *gsd_log*.
	After Update	Insereix un registre a la taula *gsd_log*.
	After Delete	Insereix un registre a la taula *gsd_log*.
gsd_user_before_trigger()	After Insert	Incrementa la clau primària auto numèrica. Regles de negoci
	After Update	Regles de negoci.

A l'Annex II d'aquesta memòria es pot trobar informació més detallada del cada *trigger*, incloent el *pseudo* codi de les accions que du a terme.

4.5.5 Motor estadístic

Sota la denominació de "motor estadístic" s'inclou tota la funcionalitat destinada a donar resposta, en temps i forma eficients, a una sèrie de consultes que s'han plantejat en la fase de requeriments. Sense dubte, un dels aspectes més interessants d'aquesta aplicació serà aquesta capacitat per explotar les dades ja introduïdes i poder obtenir informació útil de cara a la presa de decisions.

La funcionalitat del motor estadístic s'ha dissenyat tenint en consideració aspectes de rendiment. S'assumeix que aquesta BD, un cop estigui en producció, acabarà tenint un volum important de dades. Per definició, les consultes estadístiques duen a terme càlculs computacionalment costosos (agrupacions, sumes, recomptes, etc) que -quan involucren una gran quantitat de registres, com és el cas-, poden penalitzar seriosament el rendiment del sistema.

Davant aquest escenari, s'ha decidit que -en lloc de calcular aquests resultats *ad hoc*-, es preferible tenir-los emmagatzemats d'antuvi en una taula estadística ("gsd_stats"). D'aquesta manera, tots els registres són accessibles en temps constant 1, mitjançant una senzilla operació de SELECT. La contrapartida d'aquest plantejament és que pot existir un petit decalatge entre les dades reals i les calculades. Amb tot, això no hauria de representar cap problema quan es consulten períodes històrics tancats, on teòricament ja no s'ha de produir cap alteració de les dades originals.

La taula de resultats estadístics (gsd_stats) contindrà un nombre indeterminat de registres, però com aquests són el resultat de consultes d'agrupació, sumes, etc... és fàcilment demostrable que el seu nombre serà relativament baix, en comparació amb el volum total de dades tractades. A més, estarà "segmentada" per cada consulta, de manera que resulti senzill recuperar la part d'informació que interessi en cada cas.

La següent taula pot ajudar a entendre el criteri amb que s'ha dissenyat la taula de resultats estadístics (*gsd_stats*). La marca verda indica quins camps es fan servir en cada consulta.

	Ticket Type	Country Code	Year	Month	Office Code	Serv. Code	Priority	User Id	Pct (%)	Avg	Count
Query01	✓		✓	✓			✓			✓	
Query02	✓	✓	✓	✓					✓		
Query03	✓							✓			✓
Query04	✓				✓	✓					✓
Query05			✓	✓					✓		✓
Query06	✓					✓					✓
Query07	✓		✓		✓						✓
Query08			✓								
Query09	✓		✓								✓
Query10			✓						✓		
Query11											✓
Query12		✓	✓	✓					✓		

Com es pot observar, cada consulta utilitza un repertori diferents de camps, d'acord amb les seves necessitats específiques. Tots els camps de la taula "gsd_stats" s'han definit de forma genèrica, de manera que puguin ser reutilitzats en el major nombre de consultes possible.

Un cop definida l'estratègia a seguir, només queda posar-la en pràctica. En aquest sentit, existiran dos grups de procediments conceptualment diferenciats:

- Processos d'actualització estadística
  - Actualitzen els resultats de la taula *gsd_stats*.
  - Efectuen càlculs computacionalment costosos.
- Processos de consulta estadística
  - Només consulten les dades de *gsd_stats.*
  - Són relativament "lleugeres" (temps constant 1).
  - Donen resposta concreta al plantejament de cada enunciat mitjançant l'aplicació d'un filtre determinat, però intenten ser el més genèriques possible.

Així, per a cada una de les 12 consultes plantejades a l'enunciat, existiran dos procediments relacionats: un per actualitzar les dades necessàries en cada cas i un segon per efectuar la consulta pròpiament.

La instruccions necessàries per generar tots els objectes del motor estadístic es troben dins del següent *script*:

- CREATE STATISTIC OBJECTS

Com és habitual, es proveeix també el seu script antagonista, que esborra tots els objectes creats anteriorment:

- DROP STATISTIC OBJECTS

4.5.5.1 Procediments d'actualització estadística

Com a norma general, els procediments d'actualització estadística tenen les següents característiques en comú:

- S'executen a demanda, mai de forma automatitzada.
- Efectuen càlculs computacionalment costosos.
- Són procediments que no reben cap paràmetres de filtre (excepte RST). Intenten calcular el major nombre de resultats possible.
- Fan servir SQL estàtic.
- Insereixen registres dins la taula *gsd_stats*, esborrant prèviament . els valors generats en la seva anterior execució.
- Cada procés d'actualització "marca" els registres que insereix amb el número de consulta a que pertanyen (de 1 a 12) i la data/hora de la darrera execució, cosa que permet saber sempre en quin moment s'han actualitzat els resultats.
- Depenent de la complexitat dels càlculs, alguns procediments s'ajuden de funcions auxiliars, creades específicament per resoldre situacions on emprar únicament SQL podia arribar a ser complicat.

Com ja s'ha dit, existeix un procediment d'actualització estadística per a cada una de les 12 consultes platejades a l'enunciat:

Procediment	**Acció**
gsd_stats_q01_update( )	Actualitza els resultats de la consulta 1.
gsd_stats_q02_update( )	Actualitza els resultats de la consulta 2.
gsd_stats_q03_update( )	Actualitza els resultats de la consulta 3.
gsd_stats_q04_update( )	Actualitza els resultats de la consulta 4.
gsd_stats_q05_update( )	Actualitza els resultats de la consulta 5.
gsd_stats_q06_update( )	Actualitza els resultats de la consulta 6.
gsd_stats_q07_update( )	Actualitza els resultats de la consulta 7.
gsd_stats_q08_update( )	Actualitza els resultats de la consulta 8.
gsd_stats_q09_update( )	Actualitza amb els resultats de la consulta 9.
gsd_stats_q10_update( )	Actualitza els resultats de la consulta 10.
gsd_stats_q11_update( )	Actualitza els resultats de la consulta 11.
gsd_stats_q12_update( )	Actualitza els resultats de la consulta 12.

A l'Annex III d'aquesta memòria es pot trobar informació detallada sobre com funcionen aquests procediments.

4.5.5.2 Procediments de consulta estadística

Com a norma general, els procediments de consulta estadística tenen les següents característiques en comú:

- S'executen a demanda.
- No modifiquen en cap cas les dades de la taula *gsd_stats*.
- No efectuen càlculs computacionalment costosos; es limiten a recuperar registres en temps constant 1 (SELECT).
- Es plantegen de la forma més genèrica possible, per resoldre no únicament el cas particular que planteja l'enunciat. Per exemple, quan es parla "de mes actual", s'entén que s'ha d'aplicar un filtre genèric de tipus mes/any amb uns valors determinats.
- Són procediments que reben diversos paràmetres, concretament els valors dels filtres que es volen aplicar.
- Fan servir SQL dinàmic, .
- Presenten informació en pantalla dels registres que satisfan els criteris de selecció indicats.

Al igual que amb els procediments d'actualització, existeix un procediment de consulta estadística per a cada una de les 12 consultes platejades a l'enunciat:

Procediment	**Acció**
gsd_stats_q01_run( )	Executa la consulta 1 amb els filtres indicats.
gsd_stats_q02_run( )	Executa la consulta 2 amb els filtres indicats.
gsd_stats_q03_run( )	Executa la consulta 3 amb els filtres indicats.
gsd_stats_q04_run( )	Executa la consulta 4 amb els filtres indicats.
gsd_stats_q05_run( )	Executa la consulta 5 amb els filtres indicats.
gsd_stats_q06_run( )	Executa la consulta 6 amb els filtres indicats.
gsd_stats_q07_run( )	Executa la consulta 7 amb els filtres indicats.
gsd_stats_q08_run( )	Executa la consulta 8 amb els filtres indicats.
gsd_stats_q09_run( )	Executa la consulta 9 amb els filtres indicats.
gsd_stats_q10_run( )	Executa la consulta 10 amb els filtres indicats.
gsd_stats_q11_run( )	Executa la consulta 11 amb els filtres indicats.
gsd_stats_q12_run( )	Executa la consulta 12 amb els filtres indicats.

A l'Annex IV d'aquesta memòria es pot trobar informació detallada sobre com funcionen aquests procediments.

5. Proves

En el cicle de vida en cascada, la fase de proves és una de les darreres abans de la posada en marxa del producte i, per aquesta causa, sovint és la gran damnificada dins del calendari del projecte. Les possibles desviacions sofertes en fases anteriors acaben repercutint negativament en aquest estadi, normalment en forma d'una major pressió per assolir les fites d'entrega, que inevitablement es tradueix en menys temps per efectuar les proves previstes.

És un error. Sovint es subestima la importància de provar acuradament tota la funcionalitat desenvolupada abans de lliurar un projecte, però pot suposar la diferència entre l'èxit i el fracàs del mateix. De fet, hi ha *epic fails* dins el món de la informàtica –alguns d'ells, bastant coneguts-, que han succeït per no prendre's seriosament aquesta fase.

En aquest projecte, s'han identificat i dut a terme 3 tipus diferents de proves:

- Proves unitàries sobre procediments i funcions.
- Verificació de *triggers*.
- Proves sobre el Motor Estadístic:
    - Generació d'un joc de dades de prova
    - Provar els procediments d'actualització estadística
    - Provar els procediments de consulta estadística

A continuació,s'expliquen amb més detall quines han estat les accions dutes a terme en cada un d'aquests apartats.

5.1 Proves unitàries sobre procediments i funcions

És considera que hi ha un error de codificació quan algun procediment o funció avorta de manera inesperada la seva normal execució. Aquest és un escenari que s'ha d'intentar evitar a tota costa, doncs ofereix una pèssima imatge del producte i genera desconfiança entre els seus usuaris. Per evitar que això passi, s'ha afegit de manera sistemàtica un senzill però efectiu tractaments d' excepcions genèriques en tots els procediments i funcions,mitjançant la següent clàusula:

```
EXCEPTION
  WHEN OTHERS THEN
    RST := SQLERRM;
```

En principi, amb aquesta senzilla implementació, s'aconsegueix una més que acceptable estabilitat pel que fa a execució de codi. Però amb això no és suficient: a més d'un producte que no finalitzi la seva execució de forma abrupta, també es desitjable que els procediments i funcions facin correctament allò que s'espera d'ells.

En aquest sentit, per garantir el bon funcionament de tots els procediments i funcions d'aquest projecte, s'han dut a terme una sèrie de proves unitàries. La manera d'efectuar aquestes proves ha consistit en invocar vàries vegades cada procediment/funció, amb diverses combinacions de paràmetres, i observar el seu resultat; en concret, inspeccionant el contingut de la variable RST, que tots els procediment/funció incorporen de forma estàndard i que recull un missatge de text informant sobre el resultat de cada execució.

Val a dir que no es considerarà error que aparegui un missatge d'error –valgui la redundància- dins d'aquesta variable RST. Aquest fet simplement confirma que tant la lògica de negoci com el tractament d'excepcions implantats han funcionat correctament.

Les proves unitàries dutes a terme sobre cada procediment/funció del projecte estan recollides i documentades dins del següent *script*:

- TEST PROC AND FUNC

Abans de repetir aquesta bateria de proves, és molt recomanable buidar de registres totes les taules de l'aplicació, per evitar possibles errors de violació de clau primària (i.e., intentar inserir un registre que ja existeix dins d'una taula). En aquest sentit, es subministra el següent *script*:

- DELETE ALL DATA

De la mateixa manera, també és molt recomanable regenerar les seqüències, per garantir que totes comencen a numerar des de 1 (volem tenir la certesa que el primer ticket inserit tindrà Id=1). Per aconseguir-ho, es poden executar en ordre aquests dos *scripts*:

- DROP SEQUENCES
- CREATE SEQUENCES

Cada bloc de proves unitàries es troba comentat, per evitar així que interfereixi amb la resta. Per executar de nou les instruccions de cada bloc, es recomana descomentar-lo, executar a continuació el *sript* per observar els resultats i, finalment, tornar-lo a deixar comentat abans de passar a examinar el següent bloc.

Un manual imprescindible a l'hora de reproduir aquesta bateria de proves és l'Annex I, on es troben documentats tots els procediments i funcions de l'aplicació. Cada entrada d'aquest document inclou una explicació del que fa cada procediment/funció, la seva sintaxi i un exemple de com fer la corresponent crida.

El primer repertori de funcions que es proven són totes les relacionades amb les enumeracions. Són funcions de conversió relativament senzilles, que no accedeixen a la BD. Amb tot, per coherència, també s'han inclòs en aquesta bateria de proves.

A continuació, s'executen les proves unitàries dels procediments relacionats amb les taules de l'aplicació. L'estratègia seguida ha estat començar per les taules més senzilles de l'aplicació, deixant pel final les més complicades.

Cal recordar que cada taula té associat, com a mínim, un repertori de 5 procediments elementals:

- <*table_name*>_insert( )
- <*table_name*>_delete( )
- <*table_name*>_update( )
- <*table_name*>_get( )
- <*table_name*>_show( )

Aquest repertori és suficient per implementar un manteniment bàsic sobre les taules més senzilles de l'aplicació. En canvi, altres taules més complexes, presenten un major nombre de funcionalitats; especialment remarcables són els casos de les taules d'usuaris i tickets, que disposen d'un nombre sensiblement superior al de la resta. De nou, l'Annex I és el document imprescindible per saber que fa cada un dels procediments i funcions de l'aplicació.

Les proves unitàries han finalitzat amb èxit. Tots els procediments i funcions han superat aquesta prova.

5.2 Verificació de *triggers*

Els triggers, a diferència dels procediment i funcions, no es poden invocar directament. Per tant, la única manera de saber si funcionen correctament és observant si es produeixen els seus efectes.

En aquesta aplicació, s'ha fet servir *triggers* en els següents escenaris:

- Entrades a la taula de Log (gsd_log): cada cop que s'insereix, esborra o modifica un registre en alguna de les taules monitoritzades.
- Com a cas particular, cada cop que es crea o modifica un ticket, addicionalment passa que:
    - Es crea una entrada a la taula de Ticket Journal, per guardar un històrics de canvis d'estat del ticket.
    - Es crea una entrada a la taula de missatges, simulant l'enviament d'un missatge a l'usuari propietari del ticket.
- Increment de la clau auto numèrica en 5 taules:
    - gsd_log
    - gsd_ticket

    - gsd_ticket_journal
    - gsd_message
    - gsd_user
- En alguns casos concrets, per validar certes regles de negoci i avortar la transacció si aquestes no es compleixen. En concret;
    - Abans d'inserir o modificar un registre a la taula *gsd_agent* es duen a terme unes comprovacions, en funció de si és un usuari o no.
    - Abans d'inserir o modificar un registre a la taula *gsd_user,* es valida que el seu e-mail sigui corporatiu.

Aprofitant la bateria de proves unitàries del punt anterior, és pot verificar també si els *triggers* s'estan executant correctament. Així, cada cop que s'invoqui amb èxit algun procediment per inserir, modificar o esborrar un registre, a continuació s'ha d'anar a la taula de Log per verificar que s'ha generat l'entrada corresponent a aquella transacció. En cas que es tracti d'un ticket, addicionalment s'ha de fer la mateixa comprovació a les taules *gsd_ticket_journal* i *gsd_message*.

Pel que fa al correcte increment de les claus auto numèriques, només cal obrir cada una de les 5 taules indicades i comprovar que els registres estan convenientment numerats. La pròpia taula de Log, que s'haurà de visitar sovint, és un bon exemple.

Pel que fa a regles de negoci, la manera de provar que els *triggers* funcionen correctament és provocant l'error que fa avortar la transacció. Això es pot fer passant els paràmetres adequats des de la pròpia crida als procediments. Una de les proves unitàries ha consistit precisament en provocar aquest error.

En principi, totes aquestes comprovacions s'han dut a terme i es pot afirmar que el comportament dels *triggers* és el correcte i esperat en cada cas. Per tant, es donen per superades aquestes validacions.

5.3 Proves sobre el motor estadístic

Atesa la seva importància dins aquest projecte, el motor estadístic mereix un apartat propi dins el capítol de proves. Seguint la dicotomia habitual, es provaran primer els procediments d'actualització estadística, per continuar després amb les proves sobre els procediment de consultes.

Per poder provar el motor estadístic és necessari disposar d'un joc de proves suficientment gran i variat que permeti contemplar tota la casuística plantejada. Per tant, la primera tasca -que no la menys important- serà generar un conjunt de registres prou bo que ens permeti executar les proves sobre el motor estadístic.

5.3.1 Generació d'un joc de dades de prova

Amb la idea de poder generar un joc de dades suficient, s'ha desenvolupat un *script* que insereix un nombre determinant de registres en cada una de les taules de l'aplicació. El nom d'aquest *script* és:

- GENERATE STATISTIC DATASET

Abans d'executar aquest *script*, però, s'haurien de dur a terme un parell d'accions preliminars per assegurar la correcta inserció de les dades:

- Buidar de registres totes les taules de l'aplicació (recordem que existeix un *script* específic per dur a terme aquesta tasca, anomenat DELETE ALL DATA)
- Regenerar les seqüències, mitjançant l'execució en ordre dels dos *scripts* DROP SEQUENCES i CREATE SEQUENCES. Amb això, es restableixen a 1 els comptadors auto numèrics.

Un cop efectuades aquestes preparacions, ja es pot procedir amb l'execució del script GENERATE STATISTIC DATASET. En concret, genera:

- 20 països.
- 10 tipologies d'oficina.
- 1.000 usuaris registrats.
- 4 equips de suport (aquest és un requisit de l'enunciat).
- 50 agents de suports.
- 25 serveis homologats per a incidències.
- 25 serveis homologats per a peticions.
- 10.000 tickets.

El nombre de registres a generar en cada taula es pot ampliar o reduir simplement modificant el límit superior dels bucles FOR.

Per no violar les regles d'integritat referencial, s'ha de garantir que totes les claus externes existeixin com a clau primària en la seva respectiva taula d'origen. Per aconseguir-ho, les claus primàries de totes les taules –excepte les auto numèriques- s'han originat emprant uns determinats patrons, de manera que siguin fàcilment referenciables, fins i tot en el cas que s'hagin generat aleatòriament.

Existeix un component aleatori en la generació d'aquest joc de dades:

- Al inserir una nova oficina, se li assigna un codi de país i un codi de tipologia generats aleatòriament.
- Al inserir un nou usuari, se li assigna un codi d'oficina generat aleatòriament.
- Al inserir un nou agent, se li assigna un codi d'equip de suport generat aleatòriament.
- Al inserir un servei homologat, se li assigna una prioritat i un temps promig de resolució generats aleatòriament (tot i que, en aquest darrer cas, hi ha una mica de "cuina").

- Al inserir un nou ticket, diversos són els valors generats aleatòriament que se li assignen:
    - Tipus de ticket.
    - Canal d'entrada.
    - Agent que el crea.
    - Data de creació.
    - Estat del ticket.
    - Etc

El problema de treballar amb dades generades aleatòriament és que són irrepetibles: si tornem a executar el mateix *script*, obtindrem un nou joc de dades, completament diferent de l'anterior. Per tant, és molt important conservar les dades de prova que s'han emprat per validar el correcte funcionament de l'aplicació. Una bona opció pot ser exportar-les a fitxers externs. Oracle 11g disposa d'una utilitat per exportar a diversos formats (Excel,CSV, XML, etc..). En cas que sigui necessari regenerar de nou el conjunt de dades originals, s'hauran d'importar aquests fitxers sobre les seves respectives taules.

En el lliurament del producte final, juntament amb els script PL/SQL, s'adjunta un full de càlcul Excel anomenat "Complete Dataset.xlsx" que conté el joc sencer de dades que s'han fet servir en la fase de proves.

5.3.2 Provar els processos d'actualització estadística

Disposar d'un joc de dades permet començar la fase de proves sobre el motor estadístic. El primer pas consistirà en provar el correcte funcionament dels processos d'actualització estadística, com a pas previ per poder validar les posterior consultes.

Recordem que existeixen 12 processos d'actualització estadística, un per cada consulta requerida. S'ha desenvolupat un *script* que efectua la corresponent crida a aquests procediments:

- TEST STATISTIC OBJECTS

La forma de provar aquests procediments és invocant-los, cosa relativament senzilla, atès que tots ells reben únicament un paràmetre (RST). Cal recordar una vegada més que aquests procediments duen a terme càlculs intensius, motiu pel qual poden trigar una mica del normal en finalitzar la seva execució. A més, no emeten cap signe extern de la seva activitat, simplement es limiten a gravar els seus resultats dins de la taula estadística (gsd_stats).

L'execució es pot dur a terme de forma individual o tots a la vegada, com es prefereixi.

```
/*************************************************************/
/* Update statistics                                         */
/*************************************************************/
/*
  "gsd_stats_q01_update"(RST); dbms_output.put_line(RST);
  "gsd_stats_q02_update"(RST); dbms_output.put_line(RST);
  "gsd_stats_q03_update"(RST); dbms_output.put_line(RST);
  "gsd_stats_q04_update"(RST); dbms_output.put_line(RST);
  "gsd_stats_q05_update"(RST); dbms_output.put_line(RST);
  "gsd_stats_q06_update"(RST); dbms_output.put_line(RST);
  "gsd_stats_q07_update"(RST); dbms_output.put_line(RST);
  "gsd_stats_q08_update"(RST); dbms_output.put_line(RST);
  "gsd_stats_q09_update"(RST); dbms_output.put_line(RST);
  "gsd_stats_q10_update"(RST); dbms_output.put_line(RST);
  "gsd_stats_q11_update"(RST); dbms_output.put_line(RST);
  "gsd_stats_q12_update"(RST); dbms_output.put_line(RST);
*/
```

La manera de comprovar si cada procediments ha funcionat correctament serà inspeccionant la taula *gsd_stats*, verificant la data de darrera actualització dels registres propi de cada consulta.

query_id	last_updated	country	year	month	priority	average	ticket_type	percentage
12	09/06/18	COUNTR.10	2017	9	(null)	(null)	(null)	22,2222...
12	09/06/18	COUNTR.10	2017	9	(null)	(null)	(null)	56,25
12	09/06/18	COUNTR.10	2017	10	(null)	(null)	(null)	25
12	09/06/18	COUNTR.10	2017	10	(null)	(null)	(null)	25,6578...
12	09/06/18	COUNTR.10	2017	10	(null)	(null)	(null)	49,3421...
12	09/06/18	COUNTR.10	2017	11	(null)	(null)	(null)	26,1538...
12	09/06/18	COUNTR.10	2017	11	(null)	(null)	(null)	30
12	09/06/18	COUNTR.10	2017	11	(null)	(null)	(null)	43,8461...
12	09/06/18	COUNTR.10	2017	12	(null)	(null)	(null)	22,9885...
12	09/06/18	COUNTR.10	2017	12	(null)	(null)	(null)	28,7356...
12	09/06/18	COUNTR.10	2017	12	(null)	(null)	(null)	48,2758...
12	09/06/18	COUNTR.10	2018	1	(null)	(null)	(null)	25
12	09/06/18	COUNTR.10	2018	1	(null)	(null)	(null)	25
12	09/06/18	COUNTR.10	2018	1	(null)	(null)	(null)	50

Contràriament al que es podia pensar, no es dedicarà ara cap esforç a comprovar la bondat dels resultats obtinguts, perquè aquesta validació ja es farà en la segona part, quan es verifiquin els resultats dels processos de consulta. Simplement es donarà per bo cada procés d'actualització estadística només pel simple fet d'haver actualitzat el seu segment de registres dins la taula *gsd_stats*.

Dins de l'Annex III d'aquesta memòria es pot trobar informació detallada de cada procés d'actualització estadística. En concret, la comanda de SQL estàtic -que origina els resultats gravats a la taula *gsd_stats*- és directament executable des de la finestra de treball del SQL Developer. Per a cada una de les 12 consultes d'actualització, els resultats de la seva respectiva comanda SQL haurien de coincidir amb el que s'ha gravat a la taula.

Després d'haver executat els 12 processos d'actualització estadística i comprovar que, efectivament, han actualitzat les seves dades a la taula estadística, es considera que aquesta etapa de la fase de proves s'ha superat satisfactòriament i es procedeix a iniciar la següent.

5.3.3 Provar els processos de consulta estadística

La darrera part de la fase de proves consisteix en executar els processos de consulta i validar els seus resultats. Davant aquest escenari, hi han dos plantejaments possibles:

- Introduir un conjunt de dades conegudes d'antuvi, sabent *a priori* els resultats que ha de retornar cada consulta. "Pros" i contres d'aquest plantejament:
  - Es més laboriós preparar el conjunt de dades de prova, que ha de ser "a mida" dels resultats que es volen obtenir.
  - A canvi, la validació de resultats és immediata, doncs aquests ja són coneguts.

- Introduir un conjunt de dades generades aleatòriament, sense saber *a priori* quins resultats ha de retornar cada consulta. "Pros" i contres d'aquest plantejament:
  - Preparar el joc de proves és més senzill.
  - Es poden fer proves de *stress* del sistema, augmentat la quantitat de registres a inserir.
  - Malauradament, cada joc de prova és irrepetible.
  - S'han de calcular els resultats amb una altra eina, per poder-los comparar amb els obtinguts.

En aquest projecte, s'ha adoptat el segon plantejament, atès que les dades han estat generades aleatòriament, com s'ha vist anteriorment.

Es fa necessari, doncs, disposar d'una eina externa a Oracle 11g que permeti calcular els resultats, per comparar-los després amb els de les consultes estadístiques. En aquest sentit, s'ha optat per Microsoft Excel, que és ràpida i relativament senzilla d'utilitzar. La idea és aplicar des del propi Excel els mateixos filtres que aplica cada consulta, verificant que s'obtenen els mateixos resultats.

Es planteja, però, un problema addicional: la gran majoria de consultes plantejades no treballen sobre una única taula, sinó sobre diverses (mitjançant l'ús de la clàusula JOIN). Addicionalment, apareixen alguns camps calculats –per exemple, l'any i mes extrets a partir d'una data- que no existeixen en les taules físiques. Cal, doncs, generar una vista que contingui tots els camps sobre els que ens interessi treballar i exportar-la a Excel. Aquest serà el document mestre per validar el resultat de les consultes.

El darrer objecte que genera el script CREATE STATISTIC OBJECTS precisament és aquesta vista (*gsd_stats_view_tickets*), que ja conté tots els camps -físic i calculats- així com les JOIN's necessàries per obtenir el joc de dades pretès. El document Excel resultant de la seva exportació s'anomena "Testing Dataset.xlsx" i també s'inclou en el lliurament del producte.

A efectes de millorar la llegibilitat d'aquest document, és molt recomanable substituir els valors numèrics de les enumeracions pels seus literals equivalents.

	A	B	C	D	E	F	G	H	I	J	
1	id	description	type	status	severity	origin	channel	creation_date	creation_year	creation_month	last_
2	1	Ticket 00001	INC	IN PROGRESS	L3	AGENT	PHONE	15/02/18	2018	2	
3	2	Ticket 00002	PET	CREATED	PM	AUTOM	WEB	14/12/17	2017	12	
4	3	Ticket 00003	PET	IN PROGRESS	L3	AGENT	CHAT	07/03/18	2018	3	
5	4	Ticket 00004	PET	ASSIGNED	L3	AGENT	PHONE	03/10/17	2017	10	
6	5	Ticket 00005	INC	SOLVED	PM	AUTOM	WEB	30/04/18	2018	4	
7	6	Ticket 00006	PET	IN PROGRESS	PM	AUTOM	WEB	24/02/18	2018	2	
8	7	Ticket 00007	INC	ASSIGNED	L1	AUTOM	WEB	15/03/18	2018	3	
9	8	Ticket 00008	PET	ASSIGNED	L3	AGENT	PHONE	02/01/18	2018	1	
10	9	Ticket 00009	INC	ASSIGNED	L3	AUTOM	WEB	18/10/17	2017	10	
11	10	Ticket 00010	PET	ASSIGNED	L2	AUTOM	WEB	17/04/18	2018	4	
12	11	Ticket 00011	PET	SOLVED	L2	AUTOM	WEB	15/02/18	2018	2	
13	12	Ticket 00012	PET	ASSIGNED	PM	AUTOM	WEB	12/05/18	2018	5	
14	13	Ticket 00013	INC	SOLVED	L2	AUTOM	WEB	10/10/17	2017	10	
15	14	Ticket 00014	PET	ASSIGNED	PM	AGENT	CHAT	24/04/18	2018	4	
16	15	Ticket 00015	INC	IN PROGRESS	PM	AGENT	CHAT	22/10/17	2017	10	
17	16	Ticket 00016	INC	IN PROGRESS	L3	AUTOM	WEB	22/05/18	2018	5	
18	17	Ticket 00017	PET	ASSIGNED	L3	AGENT	CHAT	16/04/18	2018	4	
19	18	Ticket 00018	INC	CREATED	L2	AUTOM	WEB	04/09/17	2017	9	
20	19	Ticket 00019	PET	IN PROGRESS	L1	AUTOM	WEB	16/03/18	2018	3	
21	20	Ticket 00020	INC	CANCELLED	L3	AUTOM	WEB	09/11/17	2017	11	
22	21	Ticket 00021	PET	CREATED	L3	AGENT	PHONE	24/11/17	2017	11	

Pràcticament totes les consultes de l'enunciat són fàcilment reproduïbles dins d'Excel mitjançant la funcionalitat de taules dinàmiques que el full de càlcul ofereix. S'assumeix que la persona que efectua aquestes validacions coneix l'eina i enten en tot moment el que està fent per reproduïr els resultats de cada consulta. Per descomptat, també hauria de tenir un bon coneixement del model de dades subjacent per entendre el significat de cada camp i quins són els càlculs involucrats en cada cas.

Comentar que la consulta número 10 ("En l'any en curs, percentatge de missatges...") no agafa les seves dades de la taula de tickets, sinó de la de missatges. En aquest cas, excepcionalment, s'ha obert el fitxer "Complete Dataset.xlsx" i s'ha treballat directament sobre el full "gsd_message", que conté les dades de tots els missatges del joc de prova. Per poder resoldre aquesta consulta, ha calgut afegir al full de càlcul una columna amb l'any de la data d'enviament (*date_sent*).

Pel que fa a la resta de consultes, s'han pogut obtenir els seus resultants en Excel i comparar-los amb els obtinguts en Oracle, donant-les per bones.

Amb això, doncs, és dona per acabada la fase de proves i es consideren assolits els objectius de l'aplicació.

6. Conclusions

Aquest capítol és el darrer de la memòria i presenta les conclusions que hom extreu després d'haver dut a terme aquest projecte.

La primera cosa que crida l'atenció és que no ha resultat excessivament complicat implementar un gestor d'incidències bàsic amb Oracle 11g. Pràcticament amb la definició de 12 taules i unes 6.000 línies de codi ha estat suficient per generar l'embrió d'un aplicatiu més o menys operatiu.

En llarg del projecte, s'han tingut també l'oportunitat de treballar amb alguns aspectes interessant d'Oracle 11g, que no s'havien vist en assignatures anteriors del Grau, com per exemple el tractament d'excepcions, les seqüències per generar valors auto numèrics, la utilitat d'importació/exportació de dades o la manera de simular enumeracions. Referent a aquestes últimes, comentar que seria molt interessant que futures versions dels producte les suportessin de forma nativa (si és que no ho fan ja).

Especialment remarcable ha estat la inclusió d'un sistema de control de canvis mitjançant la taula de *log*: aquesta és una funcionalitat que pot resultar de molta utilitat pels administradors del sistema i és fàcilment exportable a altres projectes. Sense ser indispensable, una millora del mecanisme actual -que no s'ha explorat per manca de temps-, podria ser la possibilitat d'obtenir el nom de la taula automàticament consultant el catàleg de la BD, en lloc d'haver-la d'informar manualment, abans d'afegir una nova entrada al *log*.

Una de les lliçons més importants apreses en el desenvolupament d'aquest projecte ha estat la importància de triar bé la metodologia. Adoptant un enfocament conservador, en el seu moment, és va considerar que el cicle de vida clàssic podia ser un bon guió a seguir. Certament, aquesta és una metodologia amb anys de recorregut i un munt de desenvolupaments completats amb èxit a les seves esquenes, però no ha acabat de funcionar bé per aquest projecte, esdevenint sovint un inconvenient més que no pas una ajuda. El seu caràcter estrictament seqüencial entre fases la converteix en una eina massa rígida, inflexible als canvis. No poques han estat les vegades en que ha fet falta revisar un anàlisi incomplert o erroni, havent de retocar codi i documentació que ja s'havia donat per bona.

En honor a la veritat, s'ha de dir que aquestes disfuncions no han estat per culpa de la metodologia pròpiament, sinó pel mal ús que l'autor d'aquest TFG ha fet d'ella, fruit de la seva manca de perícia amb el llenguatge de programació PL/SQL i l'entorn de desenvolupament d'Oracle. En qualsevol cas, queda la sensació que una altra metodologia més moderna i millor adaptada als canvis -com puguin ser, per exemple, *Agile* o *Scrum*- hauria estat més útil pel desenvolupament d'aquest projecte. De fet, en les darreres setmanes del projecte, la forma de treballar s'assemblava més als *sprints* de *Scrum* que no pas al cicle de vida en cascada.

Pel que fa al seguiment de la planificació del projecte, s'han produït algunes desviacions importants, especialment en la fase d'anàlisi i implementació de tot el relacionat al motor estadístic. Si bé conceptualment estava bastant clar tot el que s'havia de fer, ha costat una mica més trobar la manera adequada d'implementar-ho en Oracle 11g. De nou, la desconeixença del producte i falta de pràctica amb l'eina de desenvolupament han estat factors que han penalitzat severament en aquest apartat.

Afortunadament, en previsió de possibles desviacions, en el moment de confeccionar el calendari del projecte, s'havia deixat vacant de tasques la setmana prèvia a l'entrega final. Aquests dies han resultat fonamentals per minimitzar l'impacte de la desviació soferta.

El producte obtingut, a l'espera de la seva futura integració amb una interfície gràfica -aspecte que quedava fora d'abast en aquest projecte- és mínimament operatiu i s'han assolit, amb millor o menor fortuna, tots els objectius establerts inicialment. Amb tot, encara hi ha un important marge de millora en diversos apartats com perquè pugui ser considerat una solució professional.

Algunes possibles millores podrien ser:

- Desenvolupar una base de dades de coneixement d'ús intern, d'accés accés restringit només per a tècnics.
- Al mateix temps, desenvolupar també en paral·lel una base de dades de coneixement d'ús públic
    - FAQ's
    - Problemes més habituals i com resoldre'ls (Howto's).
- Establir nous i més àgils canals de comunicació amb els usuaris, com per exemple l'enviament de SMS.
- Possibilitat d'efectuar enquestes de satisfacció en el moment de tancar un *ticket*.
- Possibilitat de poder adjuntar arxius (captures de pantalla) a les incidències/peticions, així com als missatges que es produeixen en les interaccions per resoldre un cas.

Comentari apart mereix el motor estadístic. Certament, les consultes plantejades a l'enunciat s'han resolt, però sembla poc pràctic dependre d'un sistema que necessita recompilar-se cada cop que és necessari afegir una nova consulta. De fet, aquest sistema pot esdevenir ràpidament insostenible si el nombre de consultes a mantenir és molt elevat. Per tant, és un clar candidat a ser substituït en el futur per alguna eina de *Business Analytics* o similar, que permeti la generació de noves consultes *ad hoc*, sense necessitat de tenir-les que programar una per una, com passa ara. En aquest sentit, Oracle disposa de productes empresarial per fer-ho, com per exemple *Oracle Business Intelligence Enterprise Edition 11g*.

Finalment, constatar que la documentació existent sobre Oracle 11g és extraordinàriament extensa, però sovint confusa i mal estructurada, especialment quan no es té massa clar el que s'està cercant.

7. Glossari

agile, metodologia de desenvolupament de software moderna. Veure Scrum

BD, base de dades

constraint, restricció que s'aplica sobre un tipus de data, definida a nivell de DDL en el context d'un SGBDR.

Data Warehouse**,** magatzem de dades, repositori de totes les dades històriques d'una organització.

DDL, *Data Definition Language*, llenguatge de definició de dades. És un subconjunt d'instruccions SQL destinades a la creació, modificació o eliminació d'objectes dins d'una base de dades (taules, camps, etc).

DER, Diagrama Entitat-Relació, tipus de diagrama que modela les entitats existents en el disseny d'una BD relacional, així com les relacions entre elles.

Help Desk**,** escriptori d'ajuda, denominació genèrica que engloba una sèrie d'aplicacions i serveis orientats a donar suport als usuaris davant possibles incidències o dubtes que aquests puguin tenir.

IRM**,** Information Resources Management, gestió de recursos de la informació. Programari específic per realitzar inventari del hardware i software existent dins d'una organització.

Issue Manager**,** Gestor d'Incidències. Tipus de software especialitat.

Issue Tracking**,** Seguiment de Problemes, veure Help Desk.

Open Source, codi obert, programari lliure, normalment gratuït, amb el seu codi font accessible pel gran públic.

PL/SQL**,** Procedural Language/Structured Query Language. llenguatge de programació propi dels productes de bases de dades d'Oracle.

SaaS**,** *Software as a Service*, Programari com a Servei, nou paradigma de contractació de serveis de programari, similar a una espècie de lloguer, en que només es paga per allò que es fa servir.

Scrum**,** metodologia de desenvolupament de software moderna que es caracteritza per escurçar els cicles d'iteració entre fases.

SGBDR**,** Sistema Gestor de Bases de Dades Relacionals, programari de base de dades que aplica els principis del model relacional.

SQL**,** Structured Query Language, llenguatge de consulta estructurat i estandarditzat per a consultar bases de dades relacionals.

Stored Procedure, procediment emmagatzemant, tipus especial de codi que s'executa a demanda del programador.

Ticketing, veure *Help Desk*.

Trigger, disparador, tipus especial de codi que s'executa de forma automàtica en resposta a un esdeveniment.

8. Bibliografia

[1] **Jordi Pradel, Jose Raya**, "*Requisits*".
PID_00171146. Publicacions de la UOC. Barcelona.

[2] **Jordi Pradel, Jose Raya**, "*Introducció a l'enginyeria del programari*".
PID_00171144. Publicacions de la UOC. Barcelona.

[3] **Jordi Casas**, "*Disseny conceptual de bases de dades*".
PID_00220512. Publicacions de la UOC. Barcelona.

[4] **Xavier Burgués**, "*Disseny lògic de bases de dades*".
PID_00220510. Publicacions de la UOC. Barcelona.

[5] **José Ramón Rodríguez**, "*La gestió de projectes. Conceptes bàsics*".
PID_00153527. Publicacions de la UOC. Barcelona.

[6] **Oracle Database 11g Documentation**
https://docs.oracle.com/cd/E11882_01/index.htm
Consulta del 04/03/2018 i posteriors.

[7] **Tech On The Net. Oracle PL/SQL**
https://www.techonthenet.com/oracle/
Consulta del 04/03/2018 i posteriors.

[8] **ORACLE-BASE Articles 11g**
https://oracle-base.com/articles/11g/articles-11g
Consulta del 04/03/2018 i posteriors.