Enginyeria tècnica en Informàtica de Sistemes
Treball de fi de carrera

Disseny d'un sistema operacional que controli l'àrea de multes de tràfic d'un ajuntament.

M.Rosa Pérez Bielsa

Consultor: **Alexandre Cornet**

Data **Juny 2004** *(Segon semestre any acadèmic 2003-2004)*

ÍNDEX

RESUM DEL TREBALL

El treball que es presenta a continuació, consisteix en l'elaboració del disseny i implementació d'un sistema de bases de dades relacional per Oracle 9i.

Aquest sistema serà l'elaboració d'una eina de treball per un ajuntament, concretament per l'àrea de multes de trànsit del municipi d'aquest ajuntament.

Per tal de elaborar el treball de manera consistent, partirem d'una anàlisi de requeriments per, posteriorment, realitzar el disseny conceptual, basat en un esquema d'entitat-relació, que ens permetrà dissenyar lògicament la base de dades, amb les taules, camps, tipus de dades, restriccions, relacions....

Una vegada definides les entitats amb els seus atributs i les seves relacions, es realitzarà una mena d'interfície en forma de procediments que s'encarregaran, en darrera instància, de crear, eliminar i actualitzar les dades de la base de dades.

Per tal que tot funcioni a la perfecció es requereix un coneixement ampli de la base de dades Oracle 9i i, per tant, també s'haurà de cercar informació i aprofundir en el coneixement d'aquest eina.

Per últim, i per tal de donar credibilitat al treball, es generaran una sèrie de dades per una ciutat fictícia d'un determinat nombre d'habitants. Posteriorment, comprovarem que tot funciona correctament gràcies a l'elaboració d'un joc de proves.

1.- Descripció i objectius del projecte

1.1.- Descripció

Es tracta de dissenyar un sistema operacional complex. Concretament, s'ha de realitzar el disseny i implementació d'un sistema operacional que controli l'àrea de multes de trànsit d'un ajuntament amb una població de 250.000 habitants.
Per tal de realitzar la tasca, també s'haurà d'implementar un sistema de generació de dades simulades de 4 anys d'antiguitat, per tal de donar credibilitat al projecte

1.2.- Objectius

1.2.1.- Objectius generals

- Assolir coneixements de com funcionen els sistemes operacionals
- Posar en pràctica els coneixements de bases de dades adquirits en les assignatures BD1 i BD2 aplicant-los a un cas real i complex
- Demostrar la capacitat per realitzar un projecte des del seu inici fins la seva implementació i integració.

1.2.2.- Objectius específics

- Assolir coneixements sobre el gestor de bases de dades Oracle 9i
- Aprofundir en el coneixement del llenguatge SQL

1.3.- Temporització

Tasques a realitzar:

1. Instal·lació d'Oracle:
   - Es tracta d'instal·lar el programa gestor de base de dades Oracle 9i al PC de treball, resolent els possibles problemes d'instal·lació que puguin sorgir
2. Familiarització amb Oracle i PL/SQL
   - En el meu cas, jo sóc neòfita en l'utilització d'aquest programa, per tant hauré de buscar informació, algun tutorial per internet, la documentació del mateix Oracle... i estudiar totes les possibilitats que m'ofereix el programa i el llenguatge
3. Disseny del model E/R (entitat/relació)
   - Es tracta de definir els elements (entitats) que intervindran en el projecte i les relacions existents entre ells
4. Disseny lògic de la base de dades

- Es definiran les taules, els índexs, les claus (primàries i foranes) i altres elements com disparadors, vistes, procediments, funcions...que faran la nostra base de dades molt més robusta, eficient i estable

5. Implementació del disseny lògic en SQL
   - Es tracta de traduir el disseny lògic de la base de dades al llenguatge propi d'Oracle PL/SQL
6. implementació interfície d'usuari
   - Es dissenyarà i implementarà una interfície per tal que l'usuari pugui Interactuar amb la base de dades i realitzar diferents opcions amb les dades com: crear, modificar, esborrar, consultar
7. Implementació del programa generador de dades històriques
   - Per tal de donar veracitat al projecte, cal generar dades simulades, per demostrar que la base de dades és capaç de funcionar amb grans quantitats de dades sense cap error
8. Jocs de proves
   - Com a part fonamental del projecte, hi haurà uns jocs de proves per tal de confirmar que la base de dades funciona de manera idònia en totes les seves possibilitats
9. Memòria del projecte
   - Paral·lelament al desenvolupament del projecte, es durà a terme una recopilació de dades i informes que formaran una memòria de la realització de les diferents tasques

Es disposa de 14 setmanes per dur a terme la realització del projecte

Setmana		Tasca	Esdeveniment
1	**15/3**	Instal·lació i familiarització d'Oracle 9i Redacció del capítol Oracle de la memòria	
2	**22/3**	Model E/R Redacció del capítol E/R de la memòria	
3	**29/3**	Disseny lògic	
4	**5/4**	Redacció del capítol disseny lògic	
5	**12/4**	Implementació SQL de la BD	Lliurament PAC2 (13/4)
6	**19/4**	Redacció del capítol implementació SQL	
7	**26/4**		
8	**3/5**	Implementació interfície d'usuari	
9	**10/5**	Redacció del capítol implementació interfície	
10	**17/5**		Lliurament PAC2 (17/5)
11	**24/5**	implementació programa generador de dades simulades	
12	**31/5**	Redacció del capítol implementació programa generador	
13	**7/6**	Jocs de proves Redacció del capítol Jocs de proves	
14	**14/6**	Últims retocs	
15	**21/6**	Redacció del capítol retocs	Lliurament del projecte (18/6)
16	**28/6**		Debat final (30/6)

2.- DISSENY CONCEPTUAL

2.1.- Anàlisi de requeriments

Un pas previ al del disseny conceptual d'una base de dades és el de recol·lecció d'informació i anàlisi de requeriments, que consisteix en que els dissenyadors d'una base de dades s'entrevisten amb els futurs usuaris per a recollir i documentar les seves necessitats d'informació. Paral·lelament, es defineixen els requeriments funcionals que consisteixen en operacions (transaccions) que s'aplicaran a la base de dades, i inclouen l'obtenció de dades i l'actualització

2.2.- L'esquema conceptual d'una base de dades

L'esquema conceptual conté una descripció detallada dels requeriments d'informació dels usuaris, i conté descripcions dels tipus de dades, relacions entre ells i restriccions.

Pel model conceptual hem escollit una modelització basada en un disseny E/R (Entitat/relació) que descriu les dades com a entitats, vincles (relacions) i atributs.

2.3.- El model Entitat/Relació (E/R)

El model E/R va ser proposat per Peter P. Chen entre els anys 1976-1977. Després, altres autors han investigat i escrit sobre el model, proporcionant importants aportacions. És per aquesta raó, que no és pot considerar que existeixi un únic model E/R.

El modelo E/R descriu les dades com a entitats, relacions (vincles) i atributs i permet representar l'esquema conceptual d'una base de dades de forma gràfica mitjançant els diagrames E/R

2.3.1.- Entitats i atributs

L'objecte bàsic que es representa en el model E/R és l'entitat que és qualsevol objecte del món real amb existència pròpia, sobre el qual volem tenir informació en una base de dades. Una entitat pot ser un objecte amb existència física (una persona, una casa, un empleat, un cotxe,..) o un objecte amb existència conceptual (una empresa, un lloc de treball, un curs universitari,...).

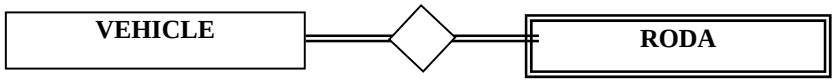
El conjunt d'entitats és la totalitat de les entitats del mateix tipus que comparteixen les mateixes propietats o atributs. En els diagrames E/R es representen mitjançant

un rectangle i dins es posa el nom. Per exemple: CIUTADÀ, PROPIETARI, AGENT, MULTA, etc. Hem d'escollir noms que comuniquin, fins on sigui possible, el significat de cada entitat. Normalment, s'utilitzen noms en singular i no en plural.

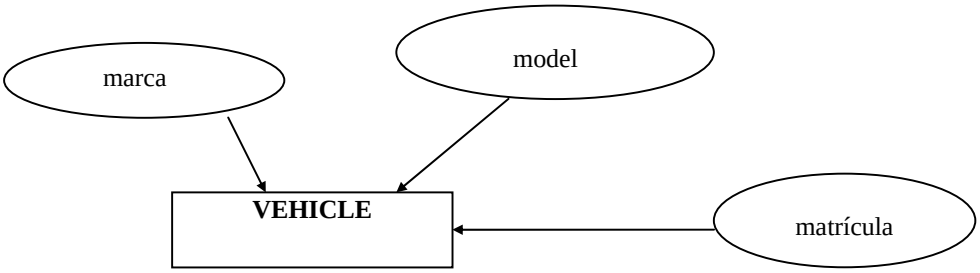
Tipus d'entitats

a) FORTES, que són les que tenen existència sense necessitat de cap altra entitat (Per exemple, MULTA). Les entitats fortes es representen així

b) FEBLES, l'existència de les quals depenen de l'existència d'una altra entitat (Per exemple, RODA depèn de VEHICLE). La desaparició de l'entitat VEHICLE provocaria la desaparició de l'entitat RODA). Aquests tipus d'entitats es representen normalment, amb un rectangle amb línies de doble traçada. És probable que aquestes entitats no tinguin suficient atributs per formar una clau primària.



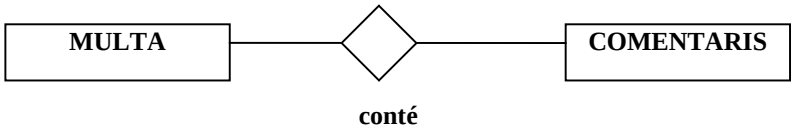
Cada entitat té propietats específiques, anomenades atributs que la descriuen. Per exemple, l'entitat VEHICLE es pot descriure per matrícula, propietari, marca, model, etc. Els atributs es representen per el·lipses connectades a l'entitat mitjançant una línia recta. En algunes representacions, com serà el nostre cas, i per economia d'espai i legibilitat, aquests atributs s'enumeren posteriorment.



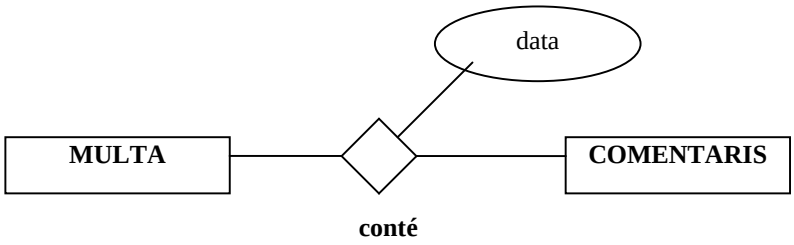
El conjunt de valors que pot prendre un atribut s'anomena domini de l'atribut
Tota entitat ha de tenir, almenys un atribut, que permeti diferenciar unes entitats d'altres, no prenen mai el mateix valor per dues entitats. Aquest atributs són les claus. En el diagrama E/R els atributs clau han de ser destacats, per exemple subratllant-los.

2.3.2.- Vincles o relacions

Es poden definir com una correspondència, associació o connexió entre dues o més entitats. En els diagrames E/R es representen gràficament mitjançant un rombe i els seus noms són verbs. Per exemple conté, sanciona...

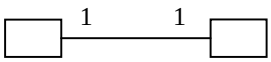
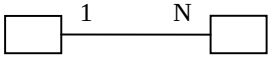
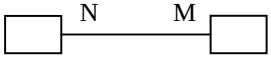


Una relació pot tenir atributs descriptius. Per exemple, a la relació anterior, podria haver-hi un atribut *data* (la data en qual es fa el comentari).



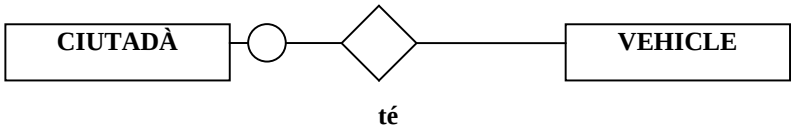
El grau d'una relació és el nombre d'entitats que hi participen. Es pot restringir el model E/R per incloure, només, conjunts de relacions binàries, es a dir, de grau 2.

La correspondència de cardinalitat, expressa el nombre màxim d'entitats que estan relacionades amb una única entitat de l'altre conjunt d'entitats que intervé a la relació. Segons la cardinalitat, podem classificar les relacions en els següents tipus:

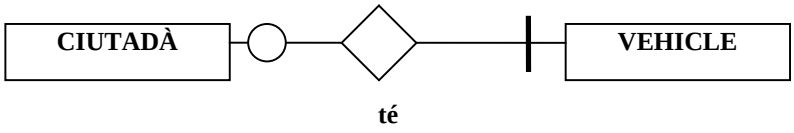
TIPUS	RELACIÓ	REPRESENTACIÓ
1:1	Una a una : La cardinalitat màxima en ambdues direccions és 1	
1:N	Una a moltes: La cardinalitat màxima en una direcció és 1 i a l'altra moltes.	
N:M	Moltes a Moltes : La cardinalitat màxima en ambdues direccions és moltes	

Tipus de participació de les entitats en una relació.

Opcional (parcial): No totes les ocurrències d'una entitat han d'estar relacionades amb alguna altra entitat. Es representa mitjançant un cercle. (Per exemple, no tot CIUTADÀ té un VEHICLE)



Obligatòria (total): Totes les ocurrències d'una entitat han d'estar vinculades amb alguna de la entitat relacionada. Per tant, existeix una participació total d'aquest conjunt d'entitats en el conjunt de relacions i es representa mitjançant una línia senzilla. (Per exemple, tot VEHICLE ha de tenir un CIUTADÀ com a propietari).

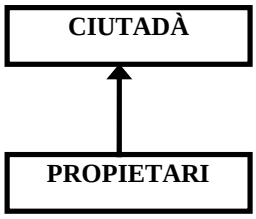


2.3.3.- superclasse i subclasse:

Una superclasse és tot tipus d'entitat sobre la qual es defineixen subclasses. (Per exemple, CIUTADÀ). Com es tracta d'entitats es representen de la mateixa manera

Una subclasse és un subconjunt del tipus entitat que té sentit en el submón, ja que té atributs particulars. (Per exemple, PROPIETARI).Com es tracta d'entitats es representen de la mateixa manera

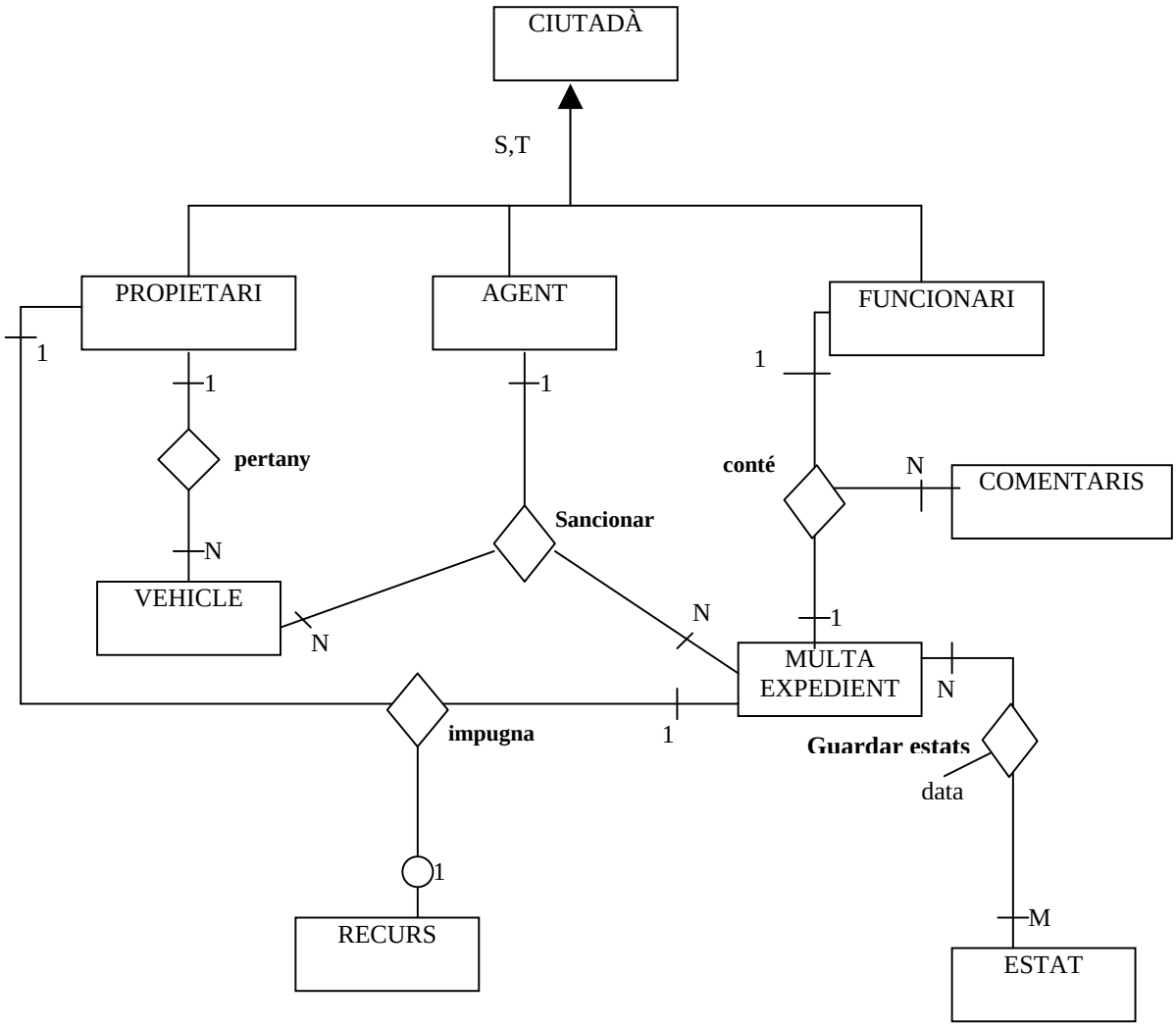
El tipus de relació entre una superclasse i les seves subclasses, és del tipus ES_UN (IS_A). Aquest tipus de relació es representa de la següent manera



2.4.- Disseny conceptual del projecte

Pel model conceptual hem escollit una modelització basada en un disseny E/R

2.4.1.- Disseny E/R del projecte



2.4.2.-Decisions del disseny

En l'herència referent a CIUTADÀ, hem considerat que la generalització/especialització ha de ser encavalcada i total.

- *Encavalcada* ja que res impedeix que un ciutadà pugui ser agent i propietari a l'hora.
- Total, perquè en el nostre cas considerem que només existeixen aquest subtipus de ciutadans, encara que en el món real existeixen d'altres tipus.

2.4.3.- Descripció del Model E/R

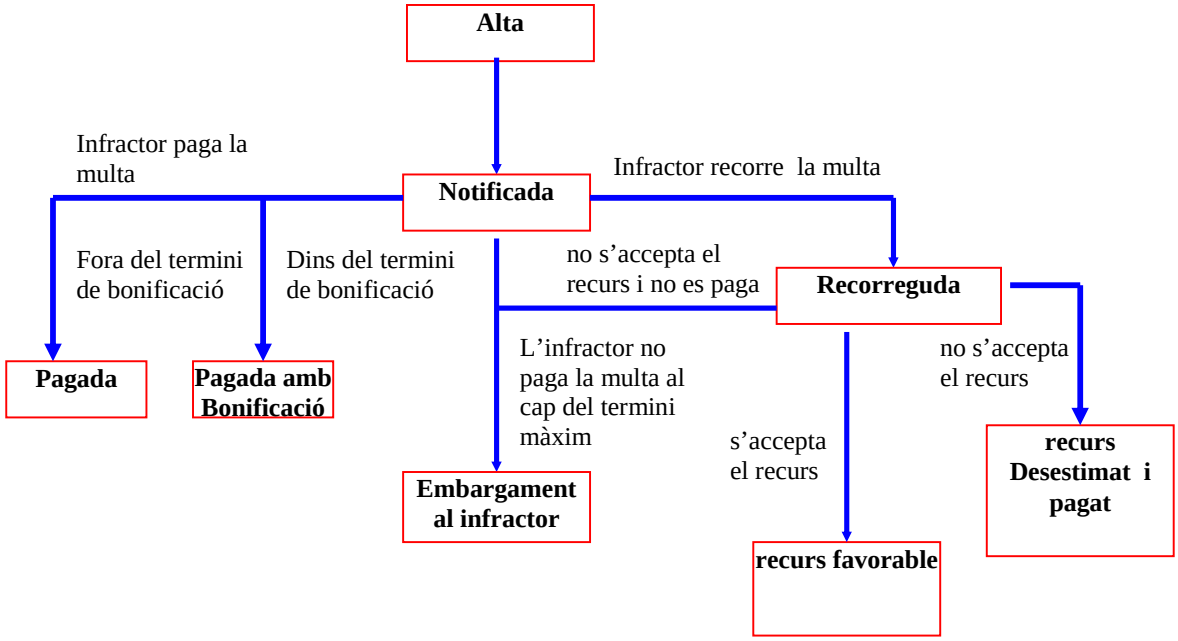
Per la realització del projecte s'han escollit aquestes entitats i el seu estudi ens ha donat les relacions que apareixen al model.

Descripció de les entitats:

MULTA - EXPEDIENT: En aquest cas es considera que el fet de posar una multa inicia un expedient, on constaran totes les dades requerides. Per tant, és indiferent parlar de multa o expedient.
Els atributs d'aquesta entitat seran:

- **IdMulta**: nombre únic i irrepetible que identificarà la multa. Com a tal, formarà la clau primària de l'entitat.
- **Data**: És la data en la que es va posar la multa. Consta de dia, mes, any i hora.
- **Agent** Identificador de l'agent que ha posat la multa. En aquest cas s'utilitzarà com a clau forana el DNI.
- **Vehicle** És el vehicle infractor. Com a identificador utilitzarem la matrícula, que és clau forana de l'entitat vehicle.
- **Lloc:** on es va posar la multa
- **Infracció**: Tipus d'infracció per la qual es va posar la multa
- **DataNotificacio:** Data en la que l'ajuntament notifica el propietari que se li ha imposat una multa
- **Import:** Import de la multa
- **DataBonificacio:** Data, abans de la qual, hi ha bonificació en el pagament
- **Bonificació:** Ens indica si hi ha bonificació en el pagament depenen de la data de notificació
- **TerminiPagament:** Data de termini que té el propietari per pagar la multa
- **TerminiRecurs:** Data de termini que té el propietari per interposar recurs
- **DataPagament**: Data en la qual s'ha pagat la multa

ESTAT. S'ha considerat que l'expedient tindrà uns estats determinats i per la millor eficàcia només es tindran en compte els registres mostrats en el següent gràfic:



Els atributs d'aquesta entitat seran:

- **IdEstat:** nombre únic i irrepetible que identifica l'estat. Formarà la clau primària de l'entitat.
- **NomEstat:** Nom de l'estat

VEHICLE. Els atributs d'aquesta entitat seran:

- **Matrícula**: nombre únic i irrepetible que identificarà el vehicle. Formarà la clau primària de l'entitat.
- **Propietari**: És el responsable del vehicle. En aquest cas s'utilitzarà com a clau forana el DNI.
- **Marca:** Marca del vehicle
- **Model:** Model del vehicle

CIUTADÀ. Els atributs d'aquesta entitat seran:

- **DNI**: nombre únic i irrepetible que identificarà el ciutadà. Formarà la clau primària de l'entitat.
- **Nom**: Nom del ciutadà
- **Cognom1**: Primer cognom del ciutadà
- **Cognom2**: Segon cognom del ciutadà
- **Adreça**: Adreça on està domiciliat el ciutadà

AGENT. És una subclasse de CIUTADÀ.
Els atributs d'aquesta entitat seran:

- **IdAgent**: nombre únic i irrepetible que identificarà l'agent de policia. Formarà la clau primària de l'entitat.
- **DNI**: nombre únic i irrepetible que l'identificarà com a ciutadà. Formarà la clau primària de l'entitat.
- **Nom**: Nom de l'agent
- **Cognom1**: Primer cognom de l'agent
- **Cognom2**: Segon cognom de l'agent
- **Adreça**: Adreça on està domiciliat l'agent

FUNCIONARI. És una subclasse de CIUTADÀ.
Els atributs d'aquesta entitat seran:

- **IdFuncionari**: nombre únic i irrepetible que identificarà al funcionari de l'ajuntament. Formarà la clau primària de l'entitat.
- **DNI**: nombre únic i irrepetible que l'identificarà com a ciutadà. Formarà la clau primària de l'entitat.
- **Nom**: Nom de l'agent
- **Cognom1**: Primer cognom de l'agent
- **Cognom2**: Segon cognom de l'agent
- **Adreça**: Adreça on està domiciliat l'agent

PROPIETARI. És una subclasse de CIUTADÀ.
Els atributs d'aquesta entitat seran:

- **DNI**: nombre únic i irrepetible que l'identificarà com a ciutadà. Formarà la clau primària de l'entitat.
- **Nom**: Nom del propietari
- **Cognom1**: Primer cognom del propietari
- **Cognom2**: Segon cognom del propietari
- **Adreça**: Adreça on està domiciliat el propietari

RECURS. Els atributs d'aquesta entitat seran:

- **IdRecurs**: nombre únic i irrepetible que identificarà el ciutadà. Formarà la clau primària de l'entitat.
- **Data**: És la data en la que es va iniciar el procediment de recurs
- **Propietari**: és el ciutadà que inicia el recurs. En aquest cas s'utilitzarà com a clau forana el DNI.

- **Multa**: Expedient al que va lligat el recurs. S'utilitzarà com a clau forana l'identificador de la multa
- **resolt:**: Ens dirà si el recurs està, o no, resolt
- **termini:** Data en la què expira la possibilitat de pagar la multa

COMENTARI. Els atributs d'aquesta entitat seran:

- **IdComentari**: nombre únic i irrepetible que identificarà el comentari. Formarà la clau primària de l'entitat.
- **Data**: És la data en la que es va fer el comentari
- **Text**: és el comentari que fa la persona de l'Administració.
- **Multa**: Expedient al que va lligat el comentari. S'utilitzarà com a clau forana l'identificador de la multa

GUARDAR_ESTAT. Els atributs d'aquesta Interrelació/Entitat seran:

- **IdMulta**: nombre únic i irrepetible que identificarà la multa.
- **Estat**: situació administrativa en la qual es troba l'expedient.
- **Data**: És la data en la que passa a un nou estat

La clau primària d'aquesta interrelació serà composta i estarà formada pels tres camps.

3.- Model lògic del projecte

3.1.- Entitats del projecte

CIUTADÀ (DNI, nom, Cognom, Cognom2, Adreça)

AGENT (IdAgent, DNI, Nom, Cognom1, Cognom2, Adreça)
Subclasse de CIUTADÀ.
{DNI} clau forana cap a CIUTADÀ (DNI)
{DNI} clau alternativa

FUNCIONARI (IdFuncionari, DNI, Nom, Cognom1, Cognom2, Adreça)
Subclasse de CIUTADÀ.
{DNI} clau forana cap a CIUTADÀ (DNI)
{DNI} clau alternativa

PROPIETARI (DNI, Nom, Cognom1, Cognom2, Adreça)
Subclasse de CIUTADÀ.
{DNI} clau forana cap a CIUTADÀ (DNI)

RECURS (IdRecurs, data, propietari, multa, resolt, termini)
{propietari} clau forana cap a PROPIETARI (DNI)
{multa} clau forana cap a MULTA (IdMulta)

MULTA-EXPEDIENT (IdMulta, data, agent, vehicle, lloc, infracció, estat, import, bonificació, dataBonificacio, dataNotificacio, terminiPagament, terminiRecurs, DataPagament)
{agent} clau forana cap a AGENT (IdAgent)
{vehicle} clau forana cap a VEHICLE (Matricula)

VEHICLE (Matricula, propietari, marca, model)
{propietari} clau forana cap a PROPIETARI (DNI)

ESTAT (IdEstat, NomEstat)

COMENTARI (IdComentari, text, data, multa)
{multa} clau forana cap a MULTA (IdMulta)

3.2.- Interrelacions:

GUARDAR_ESTAT (IdMulta, estat, data)
clau principal composta → (IdMulta, estat, data)
{idMulta} clau forana cap a MULTA (IdMulta)
{estat} clau forana cap a ESTAT (IdEstat)

4.- Model físic del projecte

A la fase de disseny físic s'especifiquen les estructures on s'han d'emmagatzemar les dades i l'organització de l'arxiu de la base de dades. També s'implementen els programes d'aplicació per les transaccions i els procediments propis del sistema.

4.1.- Sentències SQL de definició de taules

4.1.1.- taula ciutadà

```
CREATE TABLE CIUTADA (
  DNI VARCHAR2 (10) NOT NULL,
  nom VARCHAR2 (15) NOT NULL,
  Cognom VARCHAR2 (15) NOT NULL,
  Cognom2 VARCHAR2(15),
  Adreça VARCHAR2(30),
  PRIMARY KEY (DNI)
);
```

Definitivament, s'utilitzen variables de tipus VARCHAR2 que donen més flexibilitat per tractar-les, i s'especifica que els camps DNI, nom i cognom no poden contenir valors nuls.

4.1.2.- taula Agent

```
CREATE TABLE AGENT (
  IdAgent INT NOT NULL,
  DNI VARCHAR2(10) NOT NULL,
  nom VARCHAR2 (15) NOT NULL,
  Cognom VARCHAR2 (15) NOT NULL,
  Cognom2 VARCHAR2(15),
  Adreça VARCHAR2(30),
  PRIMARY KEY (IdAgent),
  FOREIGN KEY (DNI) REFERENCES CIUTADA (DNI) ON DELETE CASCADE,
  UNIQUE (DNI)
);
```

En aquest cas s'utilitzen variables de tipus VARCHAR2 excepte el camp idAgent que ens serà de tipus INTEGER per poder tractar-lo amb més facilitat en els

procediments alhora per tal de que siguin generats de manera automàtica. Els camps idAgent, DNI, nom i cognom no poden contenir valors nuls.
A més, indiquem que si s'eliminés el ciutadà referit per DNI també s'eliminaria el registre corresponent

4.1.3.- taula Funcionari

```
CREATE TABLE FUNCIONARI (
  IdFuncionari INT NOT NULL,
  DNI VARCHAR2(10) NOT NULL,
  nom VARCHAR2 (15) NOT NULL,
  Cognom VARCHAR2 (15) NOT NULL,
  Cognom2 VARCHAR2(15),
  Adreça VARCHAR2(30),
  PRIMARY KEY (IdFuncionari),
  FOREIGN KEY (DNI)  REFERENCES CIUTADA (DNI) ON DELETE CASCADE,
  UNIQUE (DNI)
);
```

Es tracta d'un cas idèntic al de la taula Agent

4.1.4.- taula Propietari

```
CREATE TABLE PROPIETARI (
  DNI VARCHAR2(10) NOT NULL,
  nom VARCHAR2 (15) NOT NULL,
  Cognom VARCHAR2 (15) NOT NULL,
  Cognom2 VARCHAR2(15),
  Adreça VARCHAR2(30),
  PRIMARY KEY (DNI),
  FOREIGN KEY (DNI)  REFERENCES CIUTADA (DNI) ON DELETE CASCADE
);
```

En aquest cas, tenim una taula pràcticament idèntica, en propietats a Agent i Funcionari, però en aquest cas no existeix un identificador exclusiu, si no que el mateix camp DNI actua com a clau primària

4.1.5.- taula Vehicle

```
CREATE TABLE VEHICLE (
  matricula VARCHAR2(10) NOT NULL,
  propietari VARCHAR2 (10),
  marca VARCHAR2 (15) NOT NULL,
  model VARCHAR2(15) NOT NULL,
  PRIMARY KEY (matricula),
  FOREIGN KEY (propietari)  REFERENCES PROPIETARI (DNI) ON DELETE SET NULL
);
```

La taula Vehicle també té tots els seus camps definits com a VARCHAR2 i en cas d'eliminació del propietari referit per PROPIETARI el camp es posaria a valor nul

4.1.6.- taula Multa

```
CREATE TABLE MULTA (
  idMulta INT NOT NULL,
  dataMulta DATE NOT NULL,
  agent INT,
  vehicle VARCHAR2 (10) NOT NULL,
  lloc VARCHAR2 (30) NOT NULL,
  infraccio VARCHAR2(30) NOT NULL,
  import INT NOT NULL,
  bonificacio VARCHAR2(1),
  dataBonificacio DATE,
  dataNotificacio DATE,
  dataPagament DATE,
  terminiPagament DATE,
  terminiRecurs DATE,
  PRIMARY KEY (idMulta),
  FOREIGN KEY (agent)  REFERENCES AGENT (idAgent) ON DELETE SET NULL,
  FOREIGN KEY (vehicle)  REFERENCES VEHICLE (matricula),
  CHECK (bonificacio = 'S' or bonificacio = 'N')
);
```

En aquesta taula tenim vàries coses a tenir en compte.
Els camps idMulta, agent i import estan definits com a INTEGER.
Els camps vehicle, lloc, infracció i bonificació com a VARCHAR2.
I, per últim, data, dataNotificacio, terminiPagament i terminiRecurs es defineixen del tipus DATE.
Els camps idMulta, data, agent, vehicle, lloc, infracció i bonificació no poden contenir valors nuls.
A més, hi ha una restricció al camp BONIFICACIO que ha de ser, o bé el caràcter "S" o bé el caràcter "N"

4.1.7.- taula Recurs

```
CREATE TABLE RECURS (
  idRecurs INT NOT NULL,
  dataRecurs DATE NOT NULL,
  propietari VARCHAR2 (10) NOT NULL,
  multa INT NOT NULL,
  resolt VARCHAR2(1) NOT NULL,
  termini DATE NOT NULL,
  PRIMARY KEY (idRecurs),
  FOREIGN KEY (propietari)  REFERENCES CIUTADA (DNI) ON DELETE CASCADE,
  FOREIGN KEY (multa)  REFERENCES MULTA (idMulta) ON DELETE CASCADE,
  CHECK (resolt = 'F' or resolt = 'D' or resolt = 'N')
);
```

Aquí tenim un cas semblant a l'anterior. Varis camps estan definits com a INTEGER (idRecurs, multa), altres com a VARCHAR2(propietari, resolt) i d'altres com a DATE(dataRecurs, termini).
També existeix una restricció al camp RESOLT, que ha de contenir el caràcter "F" que vol dir que s'ha resolt favorablement o bé el caràcter "D" que vol dir que s'ha resolt de desfavorablement o bé el caracter 'N' que vol dir que encara no s'ha resolt.

4.1.8.- taula Estat

```
CREATE TABLE ESTAT (
  idEstat INT NOT NULL,
  nomEstat VARCHAR2 (30) NOT NULL,
  PRIMARY KEY (idEstat)
);
```

Aquesta taula guarda els 8 diferents estats que pot trobar-se una multa. Ja que no hi ha més possibles estats aprofitem per omplir-la amb els diferents valors que contindrà i una vegada plena hauria de quedar bloquejada.

4.1.8.1. Sentències per omplir la taula Estat

Les sentències que ompliran la taula Estat són les següents:

```
INSERT INTO ESTAT VALUES (1, 'ALTA');
INSERT INTO ESTAT VALUES (2, 'NOTIFICADA');
INSERT INTO ESTAT VALUES (3, 'PAGADA');
INSERT INTO ESTAT VALUES (4, 'PAGADA AMB BONIFICACIO');
INSERT INTO ESTAT VALUES (5, 'RECORREGUDA');
INSERT INTO ESTAT VALUES (6, 'EMBARGAMENT A L`INFRACTOR');
INSERT INTO ESTAT VALUES (7, 'RECURS FAVORABLE');
INSERT INTO ESTAT VALUES (8, 'RECURS DESTIMAT I PAGAT');
```

4.1.9.- taula Comentari

```
CREATE TABLE COMENTARI (
  idComentari INT NOT NULL,
  text VARCHAR2 (100) NOT NULL,
  data DATE NOT NULL,
  funcionari INT NOT NULL,
  multa INT NOT NULL,
  PRIMARY KEY (idComentari),
  FOREIGN KEY (multa)  REFERENCES MULTA (idMulta) ON DELETE CASCADE,
  FOREIGN KEY (funcionari)  REFERENCES FUNCIONARI (IdFuncionari) ON
DELETE CASCADE
);
```

De nou ens trobem amb una dependència, en la qual, si s'eborren els registres referits per MULTA o per FUNCIONARI, també s'eliminaria l'actual

4.1.10.- taula Guardar-Estat

```
CREATE TABLE GUARDAR_ESTAT (
  multa INT NOT NULL,
  estat INT NOT NULL,
  data DATE NOT NULL,
  PRIMARY KEY (multa,estat,data),
  FOREIGN KEY (multa)  REFERENCES MULTA (idMulta) ON DELETE
CASCADE,
  FOREIGN KEY (estat)  REFERENCES ESTAT (idEstat) ON DELETE CASCADE
);
```

4.2.- Sentències SQL de definició de procediments

4.2.1.- procediment crear-ciutadà

```
CREATE OR REPLACE PROCEDURE CREARCIUTADA (PDni VARCHAR2,
                                          pNom VARCHAR2,
                                          pCognom VARCHAR2,
                                          pCognom2 VARCHAR2,
                                          pAdreça VARCHAR2) IS
BEGIN

      insert into CIUTADA values (pdni, pNom, pCognom, pCognom2, pAdreça);

END;
/
```

Aquest procediment crea un nou ciutadà simplement agafant els paràmetres i inserint-los directament amb una sentència INSERT a la taula ciutadà:

4.2.2.- procediment eliminar-ciutadà

```
CREATE OR REPLACE PROCEDURE ELIMINARCIUTADA (PDni VARCHAR2) IS

BEGIN

      DELETE FROM CIUTADA
WHERE pdni = dni;

END;
/
```

Aquest procediment elimina un registre ciutadà a partir d'un paràmetre que és el *dni* del ciutadà.

4.2.3.- procediment actualitzar-ciutadà

```
CREATE OR REPLACE PROCEDURE ACTUALITZARCIUTADA (PDni VARCHAR2,
                                                pNom VARCHAR2,
                                                pCognom VARCHAR2,
                                                pCognom2 VARCHAR2,
                                                pAdreça VARCHAR2) IS

BEGIN
UPDATE CIUTADA
            SET nom = pNom,
                cognom = pCognom,
                cognom2 = pCognom2,
                adreça = pAdreça
            WHERE pDni = dni;

END;
/
```

Aquest procediment actualitza totes les dades d'un determinat ciutadà especificat pel seu *dni*.

4.2.4.- procediment crear-agent

```
CREATE OR REPLACE PROCEDURE CREARAGENT (PDNI CHAR) IS

   pIdAgent  INT;
       pNom CHAR(15);
       pCognom CHAR(15);
       pCognom2 CHAR(15);
       pAdreça CHAR(30);

BEGIN

      SELECT nom, cognom, cognom2, adreça
         INTO  pNom, pCognom, pCognom2, pAdreça
         FROM CIUTADA
         WHERE PDNI = DNI;

      SELECT MAX(idAgent)
         INTO pIdAgent
         FROM AGENT;

      if (pIdAgent is null) then pIdAgent := 1;
      else pIdAgent := pIdAgent + 1;
      end if;

      insert into agent values (pIdAgent, pdni, pNom, pCognom, pCognom2,
pAdreça);

END;
/
```

Aquest procediment té, bàsicament, dues feines:

1.- A partir d'un valor (pDNI), cerca a la taula CIUTADA, el registre corresponent al valor-paràmetre pDNI i crea un nou registre a la taula AGENT.

2.- crea un nou *idAgent* per aquest nou registre cercant en la mateixa taula AGENT quin és el valor màxim del camp *idAgent* i afegint-hi una unitat.

Nota: Tal i com està dissenyat el nostre sistema no hi ha la possibilitat de crear agents si no existeix com a ciutadà. És a dir, abans ha d'existir el ciutadà per, posteriorment, convertir, aquest ciutadà en Agent.

4.2.5.- procediment eliminar-agent

```
CREATE OR REPLACE PROCEDURE ELIMINARAGENT (pIdAgent INT) IS

BEGIN
      DELETE FROM AGENT
WHERE pIdAgent = idAgent;
END;
/
```

Aquest procediment elimina un registre agent a partir d'un paràmetre que és el *idAgent* de l'agent.

4.2.6.- procediment actualitzar-agent

```
CREATE OR REPLACE PROCEDURE ACTUALITZARAGENT (PDni VARCHAR2) IS

      pNom VARCHAR2(15);
      pCognom VARCHAR2(15);
      pCognom2 VARCHAR2(15);
      pAdreça VARCHAR2(30);
BEGIN
      SELECT nom, cognom, cognom2, adreça
         INTO  pNom, pCognom, pCognom2, pAdreça
         FROM CIUTADA
         WHERE PDNI = DNI;

UPDATE AGENT
            SET nom = pNom,
                cognom = pCognom,
                cognom2 = pCognom2,
                adreça = pAdreça
            WHERE pDni = dni;
END;
/
```

Aquest procediment actualitza les dades d'un determinat agent especificat pel seu *dni.*
Però no totes les dades, ja que per mantenir la integritat referencial respecte al ciutadà del qual prové l'agent, les dades s'han d'actualitzar a partir de les del ciutadà i es mantenen el *idAgent* i el *dni*.

4.2.7.- procediment crear-funcionari

```
CREATE OR REPLACE PROCEDURE CREARFUNCIONARI (PDNI CHAR) IS

       pIdFuncionari  INT;
       pNom VARCHAR2(15);
       pCognom VARCHAR2(15);
       pCognom2 VARCHAR2(15);
       pAdreça VARCHAR2(30);

BEGIN
      SELECT nom, cognom, cognom2, adreça
         INTO  pNom, pCognom, pCognom2, pAdreça
         FROM CIUTADA
         WHERE DNI = pDNI;

      SELECT MAX(idFuncionari)
         INTO pIdFuncionari
         FROM FUNCIONARI;

  if (pIdFuncionari is null) then pIdFuncionari :=1;
      else pIdFuncionari := pIdFuncionari + 1;
      end if;

insert into FUNCIONARI values (pIdFuncionari, pdni, pNom, pCognom,
pCognom2, pAdreça);

END;
/
```

Aquest procediment és, bàsicament, el mateix que crea registres a la taula AGENT:

1.- A partir d'un valor (pDNI), cerca a la taula CIUTADA, el registre corresponent al valor-paràmetre pDNI i crea un nou registre a la taula FUNCIONARI.

2.- crea un nou *idFuncionari* per aquest nou registre cercant en la mateixa taula FUNCIONARI quin és el valor màxim del camp *idFuncionari* i afegint-hi una unitat.

4.2.8.- procediment eliminar-funcionari

```
CREATE OR REPLACE PROCEDURE ELIMINARFUNCIONARI (pIdFuncionari INT) IS

BEGIN
      DELETE FROM FUNCIONARI
            WHERE pIdFuncionari = idFuncionari;
END;
/
```

Aquest procediment elimina un registre funcionari a partir d'un paràmetre que és el *idFuncionari* de l'agent.

4.2.9.- procediment actualitzar-funcionari

```
CREATE OR REPLACE PROCEDURE ACTUALITZARFUNCIONARI (PDni VARCHAR2) IS

      pNom VARCHAR2(15);
      pCognom VARCHAR2(15);
      pCognom2 VARCHAR2(15);
      pAdreça VARCHAR2(30);
BEGIN
      SELECT nom, cognom, cognom2, adreça
         INTO  pNom, pCognom, pCognom2, pAdreça
         FROM FUNCIONARI
         WHERE PDNI = DNI;

UPDATE FUNCIONARI
            SET nom = pNom,
                cognom = pCognom,
                cognom2 = pCognom2,
                adreça = pAdreça
            WHERE pDni = dni;
END;
/
```

Aquest procediment actualitza les dades d'un determinat funcionari especificat pel seu *dni.*
Però no totes les dades, ja que per mantenir la integritat referencial respecte al ciutadà del qual prové el funcionari, les dades s'han d'actualitzar a partir de les del ciutadà i es mantenen el *idFuncionari* i el *dni*.

4.2.10.- procediment crear-propietari

```
CREATE OR REPLACE PROCEDURE CREARPROPIETARI (PDNI VARCHAR2) IS

       pNom VARCHAR2(15);
       pCognom VARCHAR2(15);
       pCognom2 VARCHAR2(15);
       pAdreça VARCHAR2(30);

BEGIN

      SELECT nom, cognom, cognom2, adreça
         INTO  pNom, pCognom, pCognom2, pAdreça
         FROM CIUTADA
         WHERE PDNI = DNI;

      insert into PROPIETARI
       values (pdni, pNom, pCognom, pCognom2, pAdreça);

END;
/
```

Aquest procediment és, bàsicament, el mateix que crea registres a la taula AGENT però no crea cap nou identificador i manté com a identificador del registre el mateix DNI que el de ciutadà

4.2.11.- procediment eliminar-propietari

```
CREATE OR REPLACE PROCEDURE ELIMINARPROPIETARI (pDni VARCHAR2) IS

BEGIN
      DELETE FROM PROPIETARI
            WHERE pDni = dni;
END;
/
```

Aquest procediment elimina un registre propietari a partir d'un paràmetre que és el *dni* del propietari.

4.2.12.- procediment actualitzar-propietari

```
CREATE OR REPLACE PROCEDURE ACTUALITZARPROPIETARI (PDni VARCHAR2) IS

      pNom VARCHAR2(15);
      pCognom VARCHAR2(15);
      pCognom2 VARCHAR2(15);
      pAdreça VARCHAR2(30);
BEGIN
      SELECT nom, cognom, cognom2, adreça
         INTO  pNom, pCognom, pCognom2, pAdreça
         FROM PROPIETARI
         WHERE PDNI = DNI;

UPDATE PROPIETARI
            SET nom = pNom,
                cognom = pCognom,
                cognom2 = pCognom2,
                adreça = pAdreça
            WHERE pDni = dni;
END;
/
```

Aquest procediment actualitza les dades d'un determinat propietari especificat pel seu *dni.*
Però no totes les dades, ja que per mantenir la integritat referencial respecte al ciutadà del qual prové el propietari, les dades s'han d'actualitzar a partir de les del ciutadà i es manté sense canvis el *dni*.

4.2.13.- procediment crear-vehicle

```
CREATE OR REPLACE PROCEDURE CREARVEHICLE (pMatricula VARCHAR2,
                                          pPropietari VARCHAR2,
                                          pMarca VARCHAR2,
                                          pModel VARCHAR2) IS
BEGIN

      insert into VEHICLE values (pMatricula, pPropietari, pMarca, pModel);

END;
/
```

Aquest procediment crea un nou vehicle simplement agafant els paràmetres i inserint-los directament amb una sentència INSERT a la taula vehicle:

4.2.14.- procediment eliminar-vehicle

```
CREATE OR REPLACE PROCEDURE ELIMINARVEHICLE (pMatricula VARCHAR2) IS

BEGIN
      DELETE FROM VEHICLE
            WHERE pMatricula = matricula;
END;
/
```

Aquest procediment elimina un registre vehicle a partir d'un paràmetre que és la *matrícula* del vehicle.

4.2.15.- procediment crear-multa

```
CREATE OR REPLACE PROCEDURE CREARMULTA (  pDataMulta DATE,
                                          pAgent INT,
                                          pVehicle VARCHAR2,
                                          pLloc VARCHAR2,
                                          pInfraccio VARCHAR2,
                                          pImport INT) IS
pIdMulta INT;

BEGIN
         SELECT MAX(idMulta)
         INTO pIdMulta
         FROM MULTA;

      if (pIdMulta is null) then pIdMulta :=1;
      else pIdMulta := pIdMulta + 1;
      end if;

insert into MULTA values (pIdMulta, pDataMulta, pAgent, pVehicle, pLloc,
pInfraccio, pImport, null, null, null, null, null, null);

CanviEstat(pIdMulta,1);

END;
/
```

Aquest procediment crea una nova multa però deixa alguns camps buits per què, fins que no estigui notificada (cosa que farem en un altre procediment), no s'hi considerarà que està activa amb la possibilitat de recurs, pagament....

El procediment realitza dues funcions:

1.- Crea la multa introduint els camps idMulta, dataMulta, agent, vehicle, lloc, infraccio i import, que són els camps coneguts en el moment de la denúncia.

2.- crea un nou *idMulta* per aquest nou registre cercant en la mateixa taula MULTA quin és el valor màxim del camp *idMulta* i afegint-hi una unitat.

3.- Crida al procediment CANVI_ESTAT per fer constar el nou estat de la multa (en aquest cas la multa passa a l'estat **ALTA**)

4.2.16.- procediment notifica-multa

```
CREATE OR REPLACE PROCEDURE NOTIFICAMULTA  (pIdMulta VARCHAR2) IS

   pBonificacio  VARCHAR2(1);
   pDataBonificacio DATE;
   pDataNotificacio DATE;
   pTerminiPagament DATE;
   pTerminiRecurs DATE;

BEGIN

  pBonificacio :='S';
  pDataNotificacio := SYSDATE();
  pDataBonificacio := pDataNotificacio + 15;
  pTerminiPagament := pDataNotificacio + 30;
  pTerminiRecurs := pDataNotificacio + 15;

      Update MULTA
         SET bonificacio = pBonificacio,
             dataNotificacio = pDataNotificacio,
             dataBonificacio = pDataBonificacio,
             terminiPagament = pTerminiPagament,
             terminiRecurs = pTerminiRecurs
         WHERE idMulta = pIdMulta;

      CanviEstat(pIdMulta,2);

END;
/
```

Aquest procediment s'executa al produir-se la notificació d'una multa i omple els camps que havíem deixat amb valor NUL al crear la multa.

El procediment realitza les següents funcions:

1.- A partir de la data de notificació calculem les dates de pagament amb bonificació, termini per impugnar la multa i termini per pagar-la.

2.- Crida al procediment CANVI_ESTAT per fer constar el nou estat de la multa (en aquest cas la multa passa a l'estat **NOTIFICADA**)

4.2.17.- procediment crear-recurs

```
CREATE OR REPLACE PROCEDURE CREARECURS(pIdMulta INT) IS

   pDataRecurs  DATE;
   pTerminiRecurs DATE;
   pPropietari VARCHAR2(10);
   pIdRecurs INT;

BEGIN

      SELECT DNI
         INTO  pPropietari
         FROM PROPIETARI,MULTA, VEHICLE
         WHERE pIdMulta = idMulta AND
               multa.vehicle = vehicle.matricula AND
               vehicle.propietari = propietari.dni;
   SELECT MAX(idRecurs)
         INTO pIdRecurs
         FROM RECURS;

      pDataRecurs := SYSDATE();
      pTerminiRecurs := pDataRecurs + 10;

      if (pIdRecurs is null) then pIdRecurs :=1;
      else pIdRecurs := pIdRecurs + 1;
      end if;

      canviestat(pIdMulta, 5);

      INSERT INTO RECURS VALUES (pIdRecurs, pDataRecurs,
                        pPropietari, pIdMulta, 'N',
                        pTerminiRecurs);

END;
/
```

Aquest procediment crea un recurs sobre una multa determinada.

El procediment realitza dues funcions:

1.- Crea el recurs introduint els camps tots els camps d'un recurs. El camp *resolt* quedarà inicialment amb el valor 'N'

2.- crea un nou *idRecurs* per aquest nou registre cercant en la mateixa taula RECURS quin és el valor màxim del camp *idRecurts* i afegint-hi una unitat.

3.- Crida al procediment CANVI_ESTAT per fer constar el nou estat de la multa (en aquest cas la multa passa a l'estat **RECORREGUDA**)

4.2.18.- procediment resoldre-recurs

```
CREATE OR REPLACE PROCEDURE RESOLDRERECURS (pIdRecurs INT,
                                            pResolucio VARCHAR2) IS

pTermini DATE;
pIdMulta INT;
pRes VARCHAR2(1);

BEGIN

  pRes := pResolucio;

  SELECT termini, multa
  INTO pTermini, pIdMulta
  FROM RECURS
  WHERE pIdRecurs = idRecurs;

  if ( ((pRes = 'F') and (pTermini < sysdate()) )  or (pRes = 'N') ) then
    pRes := 'D';
    CanviEstat (pIdMulta,8);
  end if;


  UPDATE RECURS
    SET resolt = pResolucio
    WHERE pIdMulta = multa;

  if (pRes = 'F') then CanviEstat(pIdMulta,7); end if;
  if (pRes = 'D') then CanviEstat(pIdMulta,8); end if;

END;
/
```

Aquest procediment crea un recurs sobre una multa determinada.

El procediment realitza dues funcions:

1.- Crea el recurs introduint els camps tots els camps d'un recurs. El camp *resolt* quedarà inicialment amb el valor 'N'

2.- crea un nou *idRecurs* per aquest nou registre cercant en la mateixa taula RECURS quin és el valor màxim del camp *idRecurts* i afegint-hi una unitat.

3.- Crida al procediment CANVI_ESTAT per fer constar el nou estat de la multa (en aquest cas la multa passa a l'estat **RECORREGUDA**)

4.2.19.- procediment canvi-estat

```
CREATE OR REPLACE PROCEDURE CANVIESTAT (pIdMulta INT,
                                        pNouEstat INT) IS

data DATE;

BEGIN

      data := SYSDATE();

INSERT into GUARDAR_ESTAT values(pIdMulta, pNouEstat, data);

END;
/
```

Aquest procediment crea un nou registre a la taula GUARDAR_ESTAT amb el idMulta i el nou estat que entren com a paràmetres i, per altra banda, la data actual.

4.2.20.- procediment estat-de-multa

```
CREATE OR REPLACE PROCEDURE ESTAT_DE_MULTA (pIdMulta INT) IS

pData DATE;
pEstat INT;

BEGIN

  SELECT  MAX(data)
    INTO pData
    FROM GUARDAR_ESTAT
    WHERE  multa = pIdMulta;

    SELECT  estat
    INTO pEstat
    FROM GUARDAR_ESTAT
    WHERE  multa = pIdMulta and data=pData;

    DBMS_OUTPUT.PUT_LINE('estat');
    DBMS_OUTPUT.PUT_LINE(pEstat);
    DBMS_OUTPUT.PUT_LINE('data');
    DBMS_OUTPUT.PUT_LINE(pData);
    DBMS_OUTPUT.PUT_LINE('----------------------');

END;
/
```

Aquest procediment ens diu en quin estat es troba, actualment, una determinada multa que és la dada que entra com a paràmetre al procediment

4.2.21.- procediment historial-de-multa

```
CREATE OR REPLACE PROCEDURE HISTORIAL_DE_MULTA (pIdMulta INT) IS

pMulta INT;
pEstat INT;
pData DATE;

Cursor historial IS
      SELECT multa, estat, data
      FROM GUARDAR_ESTAT
      WHERE multa = pIdMulta
      ORDER BY data;

BEGIN

    OPEN historial ;

    LOOP

    FETCH historial INTO  pMulta , pEstat , pData;
    EXIT WHEN historial%NOTFOUND OR historial%NOTFOUND IS NULL;

    DBMS_OUTPUT.PUT_LINE('idmulta');
    DBMS_OUTPUT.PUT_LINE(pMulta);
    DBMS_OUTPUT.PUT_LINE('estat');
    DBMS_OUTPUT.PUT_LINE(pEstat);
    DBMS_OUTPUT.PUT_LINE('data');
    DBMS_OUTPUT.PUT_LINE(pData);
    DBMS_OUTPUT.PUT_LINE('-----------------------');
    end LOOP;

END;
/
```

Aquest procediment ens mostra tots els estats pel quals ha passat una determinada multa que és la dada que entra com a paràmetre al procediment

4.2.22.- procediment pagar-multa

```
CREATE OR REPLACE PROCEDURE PAGARMULTA (pIdMulta INT) IS

pDataPagament date;
pTerminiPagament date;
pImport INT;
pBonificacio VARCHAR2(1);

BEGIN

SELECT import, bonificacio,terminiPagament
INTO pImport, pBonificacio, pTerminiPagament
FROM MULTA
WHERE idMulta = pIdMulta;

pDataPagament:= SysDate();

if pTerminiPagament > pDataPagament then
  pBonificacio:= 'S';
  CanviEstat(pIdMulta,4);
else
  pBonificacio:= 'N';
  CanviEstat(pIdMulta,3);
end if;

if pBonificacio = 'S' then
  pImport:= pImport/2;
end if;

UPDATE MULTA
  SET Bonificacio=pBonificacio,
      import=pImport,
      dataPagament= pDataPagament
  WHERE idMulta=pIdMulta;

END;
/
```

Aquest procediment executa el pagament d'una multa. A partir de l'identificador de la multa, que entra per paràmetre comprovem el termini de pagament amb bonificació (recordem que era que si es pagava la multa abans de 15 dies hi havia una bonificació del 50%) si entra dins dels paràmetres establerts el valor del camp *bonificació* s'actualitza a "S" o "N" depenent de les dates.

Per altra banda el camp *dataPagament* s'actualitza segons la data del sistema i el camp *import* canvia a la meitat del seu valor en cas que hi hagi bonificació.

Tant en un cas com en l'altre actualitzem l'estat de la multa mitjançant el procediment canviEstat().

4.2.23.- procediment actualitzar-termini-pagament

```
CREATE OR REPLACE PROCEDURE ACTUALITZARTERMINIPAGAMENT IS


 pIdMulta INT;
 pDataPagament DATE;

    Cursor termini IS
      SELECT idMulta, dataPagament
      FROM MULTA
      WHERE terminiPagament < SYSDATE();

BEGIN

    OPEN termini ;
    LOOP

    FETCH termini INTO pIdMulta, pDataPagament;
    EXIT WHEN termini%NOTFOUND OR termini%NOTFOUND IS NULL;

    canviEstat(pIdMulta,6);

    UPDATE MULTA SET
      dataPagament = SYSDATE()
      WHERE idMulta = pIdMUlta;

    end LOOP;

END;
/
```

Aquest procediment realitza una tasca important.

Crea un CURSOR on hi desa una selecció de multes, que compleixen dos requisits:
1.- Que el termini per pagar la multa hagi vençut
2.- Que el camp dataPagament estigui buit

Després fa canviar l'estat de cada una d'aquestes multes a l'estat número 6 (Embargament a l'infractor) i omple el camp dataPagament amb la data actual.

4.2.24.- procediment Crear-Comentari

```
CREATE OR REPLACE PROCEDURE CREARCOMENTARI (pComentari CHAR,
pFuncionari INT, pIdMulta INT) IS

      pIdComentari  INT;
      pData date;

BEGIN

      SELECT MAX(idComentari)
         INTO pIdComentari
         FROM Comentari;
      pData := SYSDATE();

      if (pIdComentari is null) then pIdComentari := 1;
      else pIdComentari := pIdComentari + 1;
      end if;

      insert into comentari values
      (pIdComentari, pComentari, pdata, pFuncionari, pIdMulta);

END;
/
```

Aquest procediment genera un nou comentari corresponent a una multa determinada, fet per un funcionari en una data concreta

4.2.25.- procediment Comentaris-de-multa

```
CREATE OR REPLACE PROCEDURE COMENTARISDEMULTA (pIdMulta INT) IS

pComentari VARCHAR2(100);
pdata date;

Cursor comentaris IS
      SELECT text, data
      FROM comentari
      WHERE multa = pIdMulta
      ORDER BY data;

BEGIN

    OPEN comentaris ;

    LOOP

    FETCH comentaris INTO  pComentari , pData;
    EXIT WHEN comentaris%NOTFOUND OR comentaris%NOTFOUND IS
NULL;


    DBMS_OUTPUT.PUT_LINE('comentari');
    DBMS_OUTPUT.PUT_LINE(pComentari);
    DBMS_OUTPUT.PUT_LINE('data');
    DBMS_OUTPUT.PUT_LINE(pData);
    DBMS_OUTPUT.PUT_LINE('-----------------------');
    end LOOP;

END;
/
```

Aquest procediment ens mostra tots els comentaris que s'han realitzat referent a una determinada multa i la data en que es va realitzar

4.2.26.- procediment comentaris-de-funcionari

```
CREATE OR REPLACE PROCEDURE COMENTARISDEFUNCIONARI  (pIdFuncionari INT) IS

pComentari VARCHAR2(100);
pdata date;
pIdMulta INT;

Cursor comentaris IS
      SELECT text, data, multa
      FROM comentari
      WHERE funcionari = pIdFuncionari
      ORDER BY data;

BEGIN

    OPEN comentaris ;
    LOOP
    FETCH comentaris INTO  pComentari , pData, pIdMulta;
    EXIT WHEN comentaris%NOTFOUND OR comentaris%NOTFOUND IS NULL;

    DBMS_OUTPUT.PUT_LINE('Multa');
    DBMS_OUTPUT.PUT_LINE(pIdMulta);
    DBMS_OUTPUT.PUT_LINE('comentari');
    DBMS_OUTPUT.PUT_LINE(pComentari);
    DBMS_OUTPUT.PUT_LINE('data');
    DBMS_OUTPUT.PUT_LINE(pData);
    DBMS_OUTPUT.PUT_LINE('----------------------');
    end LOOP;

END;
/
```

Aquest procediment ens mostra tots els comentaris que ha realitzat un determinat funcionari, indicant la multa corresponent i la data en que es va realitzar

5.- Jocs de proves

Per comprovar el funcionament de la base de dades s'ha creat un conjunt de proves (insercions, esborrats, actualitzacions...) per tal de validar i comprovar l'efectivitat del disseny.

S'ha intentat que siguin el més complets possible i s'ha intentat cercar els possibles errors que hi puguin haver per tal d'esmenar-los.

Bàsicament es tracta de crear, eliminar, fer *Selects* de les taules per tal de comprovar que els procediments funcionin tal i com han estat dissenyats.

5.1.- Scripts SQL (Jocs de proves)

```
SET SERVEROUTPUT ON;

--proves referents a ciutadans
EXECUTE CREARCIUTADA('11', 'Enric','Fernàndez','Guerrero','blocs 4');
EXECUTE CREARCIUTADA('22', 'Miquel','Pérez','Carpio','Gran via 42');
EXECUTE CREARCIUTADA('33', 'Antoni','Gómez','Gonzàlez','Diagonal 42');
EXECUTE CREARCIUTADA('44', 'Tomàs','Andújar','Pol','Vertical 3');
EXECUTE CREARCIUTADA('55', 'Jordi','Moliner','Gil','Horitzontal 11');
EXECUTE CREARCIUTADA('77', 'Albert','Fernàndez','Guerrero','blocs 4');
EXECUTE CREARCIUTADA('88', 'Norbert','Pérez','Carpio','Gran via 42');
EXECUTE CREARCIUTADA('99', 'Eloi','Gómez','Gonzàlez','Diagonal 42');
EXECUTE CREARCIUTADA('66', 'Jaume','Andújar','Pol','Vertical 3');
EXECUTE CREARCIUTADA('1000', 'Andrea','Moliner','Gil','Horitzontal 11');

SELECT * FROM CIUTADA;

EXECUTE ELIMINARCIUTADA('22');
EXECUTE ELIMINARCIUTADA('33');

SELECT * FROM CIUTADA;

-- proves referents a agents

EXECUTE CREARAGENT('44');
EXECUTE CREARAGENT('55');

SELECT * FROM AGENT;

EXECUTE ELIMINARAGENT(1);

SELECT * FROM AGENT;

-- proves referents a funcionaris

EXECUTE CREARFUNCIONARI('44');
EXECUTE CREARFUNCIONARI('55');

SELECT * FROM FUNCIONARI;

EXECUTE ELIMINARFUNCIONARI(2);
```

```
SELECT * FROM FUNCIONARI;

-- proves referents a propietaris

EXECUTE CREARPROPIETARI('44');
EXECUTE CREARPROPIETARI('55');
EXECUTE CREARPROPIETARI('66');

SELECT * FROM PROPIETARI;

EXECUTE ELIMINARPROPIETARI('44');

SELECT * FROM PROPIETARI;


--proves referents a ciutadans

EXECUTE CREARVEHICLE('B-44444','55','Renault', 'Space');
EXECUTE CREARVEHICLE('B-33333','66','Seat', 'Panda');
EXECUTE CREARVEHICLE('B-22222','66','Peugeot', '306');

SELECT * FROM VEHICLE;

EXECUTE ELIMINARVEHICLE('B-22222');

SELECT * FROM VEHICLE;

-- proves referents a Multes

EXECUTE CREARMULTA('25/05/2004',2,'B-44444','Balmes', 'estacionament prohibit',90);
EXECUTE CREARMULTA('18/05/2004',2,'B-33333','Gran de Gracia','estacionament prohibit',90);
EXECUTE CREARMULTA('30/05/2004',2,'B-33333','Rbla. Catalunya', 'estacionament prohibit',90);

SELECT * FROM MULTA;

EXECUTE NOTIFICAMULTA(1);
EXECUTE NOTIFICAMULTA(2);
EXECUTE NOTIFICAMULTA(3);

SELECT * FROM MULTA;

-- proves referents a comentaris

EXECUTE CREARCOMENTARI('COMENTARI1',1,1);
EXECUTE CREARCOMENTARI('COMENTARI2',1,2);
EXECUTE CREARCOMENTARI('COMENTARI3',1,3);

SELECT * FROM COMENTARI;

EXECUTE COMENTARISDEMULTA(1);
EXECUTE COMENTARISDEMULTA(2);
EXECUTE COMENTARISDEFUNCIONARI(1);

-- proves referents a Recursos

EXECUTE ESTAT_DE_MULTA(1);
EXECUTE CREARECURS(1);
SELECT * FROM RECURS;
EXECUTE ESTAT_DE_MULTA(1);

EXECUTE RESOLDRERECURS(1,'F');
SELECT * FROM RECURS;
```

```
SELECT * FROM GUARDAR_ESTAT;

EXECUTE HISTORIAL_DE_MULTA(1);
```

Una vegada executats es joc de proves, s'ha pogut confirmar el funcionament correcte del sistema.

6.- Generació de dades històriques simulades

La generació de dades simulades històriques s'ha efectuat mitjançant una aplicació programada en un llenguatge d'alt nivell (Java).

El codi font del programa, així com l'executable corresponent estan accessibles des dels vincles següents.


Tfc.java


Tfc.class

El programa està pensat per fer-lo servir des d'una finestra MS-DOS redireccionant la sortida cap a un fitxer de text que, posteriorment, executarem en una sessió del SQL Plus Worksheet d'Oracle.

Un exemple d'execució del programa seria:

Java Tfc paràmetre >arxiuSortida.txt

El paràmetre indica el nombre de ciutadans generats per un cas concret. La resta de entitats del sistema es generen en relació al nombre de ciutadans.
Per tal que el programa funcioni de manera correcta és necessari que el paràmetre sigui un enter múltiple de 1000.

L'arxiu de sortida es genera amb les característiques adients per ser tractats com uns Scripts SQL que només cal executar-los per tal de aconseguir la generació de dades històriques simulades del sistema.

7.- Conclusions

Com a conclusió general, es pot dir que s'han assolit els objectius marcats a l'inici del projecte, tant els objectius generals com els específics

La part més complicada del projecte ha estat l'utilització del gestor de bases de dades Oracle 9i, ja que l'experiència prèvia amb aquest programari, abans de començar el projecte, era nul·la, i la documentació d'aquest programa és tant extensa que trobar una solució a un problema determinat es convertia en una tasca dificultosa.

El disseny conceptual ha requerit un estudi profund i exhaustiu de la situació del projecte, una àmplia recerca mitjançant la lectura d'una bibliografia especialitzada i cerques en Internet

El disseny lògic ha resultat la part més dificultosa del projecte en especial la implementació de procediments i la seva depuració a través de jocs de proves.

Els jocs de proves i la generació de dades simulades han resultat especialment satisfactòries al comprovar el seu funcionament

8.- Bibliografia

- ORACLE 9I: ADMINISTRACION Y ANALISIS DE BASES DE DATOS, César Pérez , ed. RA-MA

- ORACLE 9I: MANUAL DE REFERENCIA, Kevin Loney i George Koch, ed. MCGRAW-HILL

- FUNDAMENTOS DE SQL, Forrest Houlette, ed. MCGRAW-HILL

- PROCESAMIENTO DE BASES DE DATOS (8ª ED.): FUNDAMENTOS, DISEÑO E IMPLEMENTACION, David Kroenke, ed. Prentice Hall

- MATERIAL DIDÀCTIC DE BASES DE DADES 1 I 2 DE LA UOC

Enllaços webs

- http://www3.uji.es/~mmarques/f47/apun/node31.html
- http://rinconprog.metropoliglobal.com/CursosProg/BDatos/IntroBD/index.php?cap=2
- http://milugar.homeip.net:8080/archivos/actweb/Documentacion/ActWeb_Documentacion_Extendida/out-htmls/der.html
- http://www.itlp.edu.mx/publica/tutoriales/basedat1/