Tema 5 - Llenguatge de Manipulació de Dades (DML)

IES San Vicente

5. Llenguatge de manipulació de dades (DML)

Les sentencies de manipulació de dades o llenguatge de modificació de dades (DML) són sentencies SQL que recuperen, insereixen, actualitzen i eliminen les files d'una taula en una base de dades d'Oracle. Les quatre sentencies DML bàsiques són SELECT, INSERT, UPDATE i DELETE.

Per fer consultes a la base de dades utilitzarem la sentència SELECT de SQL. De la consulta se'n pot obtenir: qualsevol unitat de dades, totes les dades, qualsevol subconjunt de dades, qualsevol subconjunt de subconjunts de dades.

5.1. Base de dades d'alumnes i professors

Per al desenvolupament d'aquest tema utilitzarem una Base de Dades sobre Alumnes, Professors i les relacions d'Alumnes matriculats a Assignatures. Aquestes taules es detallen a continuació i també es poden descarregar des de la plataforma Aules.

```
CREATE TABLE alumnes (
    dni VARCHAR2(20) PRIMARY KEY,
    nom VARCHAR2(20),
    cognoms VARCHAR2(40),
    data_nac DATE
);

CREATE TABLE professors (
    dni VARCHAR2(20) PRIMARY KEY,
    nom VARCHAR2(20),
    cognoms VARCHAR2(40)
);


CREATE TABLE assignatures (
    cod VARCHAR2(5) PRIMARY KEY,
    nom VARCHAR2(20),
    dni_professor VARCHAR2(20),
      FOREIGN KEY (dni_professor)
            REFERENCES professors (dni) ON DELETE CASCADE
);

CREATE TABLE matriculat (
    dni_alumne VARCHAR2(20),
    cod_assignatura VARCHAR2(5),
    curs_academic NUMBER(4, 0) DEFAULT 2022,
    nota NUMBER(2, 0) DEFAULT 1,
      PRIMARY KEY (dni_alumne, cod_assignatura),
      FOREIGN KEY (dni_alumne)
            REFERENCES alumnes (dni) ON DELETE CASCADE
);
```

I com a dades d'exemple podríem emprar:

```
INSERT INTO alumnes (dni,nom,cognoms,data_nac)
VALUES ('11111111A','JORDI','SANCHEZ',TO_TIMESTAMP('10/10/80
00:00:00,000000000','DD/MM/RR HH24:MI:SS,FF'));
INSERT INTO alumnes (dni,nom,cognoms,data_nac)
VALUES ('11111112B','PACO','PEREZ',TO_TIMESTAMP('10/11/80
00:00:00,000000000','DD/MM/RR HH24:MI:SS,FF'));
INSERT INTO alumnes (dni,nom,cognoms,data_nac)
VALUES ('11111113C','RAFA','ROMERO',TO_TIMESTAMP('01/12/81
00:00:00,000000000','DD/MM/RR HH24:MI:SS,FF'));
INSERT INTO alumnes (dni,nom,cognoms,data_nac)
VALUES ('11111114D','JAVI','PONCE',TO_TIMESTAMP('12/01/82
00:00:00,000000000','DD/MM/RR HH24:MI:SS,FF'));
INSERT INTO alumnes (dni,nom,cognoms,data_nac)
VALUES ('11111115E','MANOLI','ALVAREZ',TO_TIMESTAMP('10/02/81
00:00:00,000000000','DD/MM/RR HH24:MI:SS,FF'));
INSERT INTO alumnes (dni,nom,cognoms,data_nac)
VALUES ('11111116F','MANUEL','PEREZ',TO_TIMESTAMP('13/03/80
00:00:00,000000000','DD/MM/RR HH24:MI:SS,FF'));
INSERT INTO alumnes (dni,nom,cognoms,data_nac)
VALUES ('11111117G','ANA','SANCHEZ',TO_TIMESTAMP('10/10/83
00:00:00,000000000','DD/MM/RR HH24:MI:SS,FF'));
INSERT INTO alumnes (dni,nom,cognoms,data_nac)
VALUES ('11111118H','ISABEL','PEREZ',TO_TIMESTAMP('15/04/86
00:00:00,000000000','DD/MM/RR HH24:MI:SS,FF'));
INSERT INTO alumnes (dni,nom,cognoms,data_nac)
VALUES ('11111119I','ESTEFANIA','SANCHEZ',TO_TIMESTAMP('10/10/85
00:00:00,000000000','DD/MM/RR HH24:MI:SS,FF'));

INSERT INTO professors (dni,nom,cognoms) values
('22222222A','ALVARO','LOPEZ');
INSERT INTO professors (dni,nom,cognoms) values
('22222223B','PEDRO','SANCHEZ');
INSERT INTO professors (dni,nom,cognoms) values
('22222224C','MAITE','PEREZ');
INSERT INTO professors (dni,nom,cognoms) values
('22222225D','PACO','ALMAGRO');
INSERT INTO professors (dni,nom,cognoms) values
('22222226E','FERMINA','AGUILAR');

INSERT INTO assignatures (cod,nom,dni_professor) VALUES ('BDD','Bases de
dades','22222222A');
INSERT INTO assignatures (cod,nom,dni_professor) VALUES ('DEC','Desenv. ent.
client','22222223B');
INSERT INTO assignatures (cod,nom,dni_professor) VALUES ('FOL','Form. orien.
laboral','22222226E');
INSERT INTO assignatures (cod,nom,dni_professor) VALUES ('ADA','Acces a
dades','22222224C');
INSERT INTO assignatures (cod,nom,dni_professor) VALUES
('PRO','Programacio','22222222A');
INSERT INTO assignatures (cod,nom,dni_professor) VALUES ('XAL','Xarxes area
local',NULL);

INSERT INTO matriculat (dni_alumne,cod_assignatura,curs_academic,nota) values
('11111111A','DEC',2020,7);
INSERT INTO matriculat (dni_alumne,cod_assignatura,curs_academic,nota) values
('11111111A','BDD',2020,5);
INSERT INTO matriculat (dni_alumne,cod_assignatura,curs_academic,nota) values
('11111111A','ADA',2020,7);
INSERT INTO matriculat (dni_alumne,cod_assignatura,curs_academic,nota) values
('11111112B','DEC',2019,8);
INSERT INTO matriculat (dni_alumne,cod_assignatura,curs_academic,nota) values
('11111113C','FOL',2020,4);
INSERT INTO matriculat (dni_alumne,cod_assignatura,curs_academic,nota) values
('11111113C','BDD',2020,3);
INSERT INTO matriculat (dni_alumne,cod_assignatura,curs_academic,nota) values
('11111114D','FOL',2020,9);
INSERT INTO matriculat (dni_alumne,cod_assignatura,curs_academic,nota) values
('11111114D','BDD',2017,6);
```

```
INSERT INTO matriculat (dni_alumne,cod_assignatura,curs_academic,nota) values
('11111115E','DEC',2020,9);
INSERT INTO matriculat (dni_alumne,cod_assignatura,curs_academic,nota) values
('11111115E','BDD',2020,3);
INSERT INTO matriculat (dni_alumne,cod_assignatura,curs_academic,nota) values
('11111116F','FOL',2020,4);
```

Si tens problemes per a instal·lar Oracle i vols fer proves bàsiques amb SQLite o un altre gestor de bases de dades (encara que en aqueix cas, no podràs provar els exemples més avançats), les dades de alumnes podrien ser:

```
INSERT INTO alumnes (dni,nom,cognoms,data_nac) VALUES
('11111111A','JORDI','SANCHEZ','1980/10/10');
INSERT INTO alumnes (dni,nom,cognoms,data_nac) VALUES
('11111112B','PACO','PEREZ','1980/11/10');
INSERT INTO alumnes (dni,nom,cognoms,data_nac) VALUES
('11111113C','RAFA','ROMERO','1981/12/01');
INSERT INTO alumnes (dni,nom,cognoms,data_nac) VALUES
('11111114D','JAVI','PONCE','1982/01/12');
INSERT INTO alumnes (dni,nom,cognoms,data_nac) VALUES
('11111115E','MANOLI','ALVAREZ','1981/02/10');
INSERT INTO alumnes (dni,nom,cognoms,data_nac) VALUES
('11111116F','MANUEL','PEREZ','1980/03/13');
INSERT INTO alumnes (dni,nom,cognoms,data_nac) VALUES
('11111117G','ANA','SANCHEZ','1983/10/10');
INSERT INTO alumnes (dni,nom,cognoms,data_nac) VALUES
('11111118H','ISABEL','PEREZ','1986/04/15');
INSERT INTO alumnes (dni,nom,cognoms,data_nac) VALUES
('11111119I','ESTEFANIA','SANCHEZ','1985/10/10');
```

5.2. La sentència SELECT

5.2.1. Sintaxi general de la senténcia SELECT:

```
SELECT [ALL/DISTINCT]
<expre_column1, expre_column2…, expre_column | * >
FROM <llista_de_taules>
[WHERE<condició>]
[GROUP BY<llista_d'atributs> [HAVING<condició_de_grup> ]]
[ORDER BY<llista_d'atributs> [ASC/DESC] ];
```

On expre_column pot ser una columna d'una taula, una constant, una expressió aritmètica, una funció o diverses funcions imbricades.

5.2.2. Clàusula FROM

L´única clàusula de la sentència SELECT que és obligatòria és la clàusula FROM, la resta són opcionals.

FROM: From [nom_taula1, nom_taula2,…..,nom_taula_n]

Especifica la taula o llista de taules de les quals es recuperaran les dades. Amb aquesta sentència obtenim la informació de les columnes requerides de tota la taula.

Exemple: consultem els noms dels ALUMNES.

```
SELECT nom FROM alumnes;
```

```
nom
---------
JORDI
PACO
RAFA
JAVI
MANOLI
MANUEL
ANA
ISABEL
ESTEFANIA
```

És possible associar un nom nou a les taules mitjançant **àlies**.

Exemple: si la taula professors li donem el nom P, les columnes de la taula aniran acompanyades de P

```
SELECT p.dni, p.nom FROM professors p;
```

```
dni         nom
---------   -------
22222223B   PEDRO
22222224C   MAITE
22222225D   PACO
22222226E   FERMINA
```

Així per exemple podrem diferenciar columnes que es diguen igual, per exemple entre Professors i Alumnes, on el Nom és el mateix.

Exemple (que mostrarà informació sense sentit, perquè relacionarà cada nom de professor amb tots els noms d'alumnes; més avant aprendrem a solucionar-ho):

```
SELECT p.nom, a.nom FROM professors p, alumnes a;
```

```
nom      nom
-------  ---------
PEDRO    JORDI
PEDRO    PACO
PEDRO    RAFA
PEDRO    JAVI
PEDRO    MANOLI
PEDRO    MANUEL
PEDRO    ANA
PEDRO    ISABEL
PEDRO    ESTEFANIA
MAITE    JORDI
MAITE    PACO
(...)
```

5.2.3. Clàusula WHERE

WHERE: [WHERE condició]

Obté les files que compleixen la condició expressada. La complexitat de la condició és il·limitada. El format de la condició és: *Expressió operador expressió*

```
SELECT nom FROM alumnes WHERE dni = '11111111A';
```

Els operadors de comparació i operadors lògics que es fan servir són:

Operadors de comparació			Operadors lògics	
OPERADOR	FUNCIÓ		OPERADOR	FUNCIÓ
< > < > <= >= =	Menor que Major que Diferent de Menor o Igual que Major o Igual que Igual que		AND	És el "i" lògic. Avalua dues condicions i torna un valor de veritat només si totes dues són certes.
BETWEEN	Especifica un interval de valors.		OR	És el "o" lògic. Avalua dues condicions i torna un valor de veritat si alguna de les dues és certa.
LIKE	Comparació d´un model (veure apartat 5.2.8)		NOT	Negació lògica. Retorna el valor contrari de l'expressió

Es poden construir condicions múltiples usant els operadors lògics booleans estàndards: **AND, OR, NOT**. Està permès emprar parèntesis per forçar l'ordre d'avaluació:

Exemples:

```
SELECT * FROM matriculat WHERE curs_academic > 2018;
```

```
dni_alumne  cod_assignatura  curs_academic  nota
----------  ---------------  -------------  ----
11111111A   DEC              2020           7
11111111A   BDD              2020           5
11111111A   ADA              2020           7
11111112B   DEC              2019           8
11111113C   FOL              2020           4
11111113C   BDD              2020           3
11111114D   FOL              2020           9
11111115E   DEC              2020           9
11111115E   BDD              2020           3
11111116F   FOL              2020           4
```

```
SELECT * FROM matriculat WHERE cod_assignatura = 'DEC';
```

```
dni_alumne  cod_assignatura  curs_academic  nota
----------  ---------------  -------------  ----
11111111A   DEC              2020           7
11111112B   DEC              2019           8
11111115E   DEC              2020           9
```

```
SELECT nom FROM professors WHERE nom LIKE 'A%';
```

```
nom
------
ALVARO
```

```
SELECT * FROM matriculat
WHERE curs_academic > 2018 AND curs_academic <=2020;
```

```
dni_alumne  cod_assignatura  curs_academic  nota
----------  ---------------  -------------  ----
11111111A   DAC              2020           7
11111111A   DEG              2020           5
11111111A   ADA              2020           7
11111112B   DAC              2019           8
11111113C   FOL              2020           4
11111113C   DEG              2020           3
11111114D   FOL              2020           9
11111115E   DAC              2020           9
11111115E   DEG              2020           3
11111116F   FOL              2020           4
```

5.2.4. Clàusula ORDER BY

ORDER BY:[Order By expre_columna [Desc | Asc] [,expre columna [Desc | Asc]]…]

Podem endreçar l'eixida produïda per la sentència Select per valors ascendents (Asc) o descendents (Desc) d'una columna en particular. Si no sindica res, l'ordenació serà ascendent.

Exemples

```
SELECT * FROM alumnes ORDER BY cognoms;
```

```
dni         nom         cognoms  data_nac
---------   ---------   -------  ----------
11111115E   MANOLI      ALVAREZ  1981/02/10
11111112B   PACO        PEREZ    1980/11/10
11111116F   MANUEL      PEREZ    1980/03/13
11111118H   ISABEL      PEREZ    1986/04/15
11111114D   JAVI        PONCE    1982/01/12
11111113C   RAFA        ROMERO   1981/12/01
11111111A   JORDI       SANCHEZ  1980/10/10
11111117G   ANA         SANCHEZ  1983/10/10
11111119I   ESTEFANIA   SANCHEZ  1985/10/10
```

5.2.5. Clàusula ALL/DISTINCT

- **ALL**: Amb la clàusula ALL, recuperem totes les files encara que estiguen repetides. És l'opció per defecte
- **DISTINCT**: Elimina les files repetides.

Exemples

```
SELECT ALL cognoms FROM alumnes ORDER BY cognoms;
```

```
cognoms
-------
ALVAREZ
PEREZ
PEREZ
PEREZ
PONCE
ROMERO
SANCHEZ
SANCHEZ
SANCHEZ
```

```
SELECT DISTINCT cognoms FROM alumnes ORDER BY cognoms;
```

```
cognoms
-------
ALVAREZ
PEREZ
PONCE
ROMERO
SANCHEZ
```

5.2.6. Crear i utilitzar Àlies de columnes

Quan es consulta la base de dades, els noms de les columnes es fan servir com a capçaleres de presentació. Si el nom resulta massa llarg, curt o complicat, hi ha la possibilitat de canviar-lo utilitzant un Àlies a la mateixa sentència Select.

Exemple:

```
SELECT dni_alumne AS alumne FROM matriculat;
```

```
alumne
---------
11111111A
11111111A
11111111A
11111112B
11111113C
11111113C
11111114D
11111114D
11111115E
11111115E
11111116F
```

5.2.7. Operadors aritmètics

Els operadors Aritmètics serveixen per formar expressions amb constants, valors de columnes i funcions de valors de columna

Operador aritmètic	Operació
+	Suma
-	Resta
*	Multiplicació
/	divisió

Exemples:

```
SELECT 5+3 AS suma, curs_academic, curs_academic+10 FROM matriculat;
```

```
suma  curs_academic  curs_academic+10
----  -------------  ----------------
8     2020           2030
8     2020           2030
8     2020           2030
```

```
8       2019            2029
8       2020            2030
8       2020            2030
8       2020            2030
8       2017            2027
8       2020            2030
8       2020            2030
8       2020            2030
```

5.2.8. Operadors de comparació de cadenes de caràcters (LIKE)

Per comparar cadenes senceres de caràcters es pot fer amb l'operador de comparació =. No obstant això, s'utilitza l'operador LIKE quan volem comparar alguns caràcters d'aquesta cadena.

L'operador **LIKE** es fa servir amb dos caràcters especials:

- **% (comodí)**: substitueix qualsevol cadena de 0 o més caràcters.
- **'_' (Marcador de posició)**: substitueix un caràcter qualsevol.

També es pot utilitzar NOT LIKE

Exemples:

A partir de la taula professors obté els noms que comencen per una M

```
SELECT nom FROM professors WHERE nom LIKE 'M%';
```

```
nom
-----
MAITE
```

Obtingues aquells noms que tinguen una L en segona posició:

```
SELECT nom FROM professors WHERE nom LIKE '_L%';
```

```
nom
------
ALVARO
```

Obtingues aquells noms que comencen per P i tinguen una A al seu interior

```
SELECT nom FROM professors WHERE nom LIKE 'P%A%';
```

```
nom
----
PACO
```

5.2.9. Null i Not Null

Una columna d'una fila és Null si està completament buida. Per comprovar si el valor d'una columna és nul, utilitzem l'expressió: columna IS NULL. Per saber si el valor de la columna no és nul, utilitzem

l'expressió: columna IS NOT NULL. Quan comparem amb valors nuls o no nuls no podem utilitzar els operadors d'igualtat, major o menor.

Exemples:

```
SELECT * FROM assignatures WHERE dni_professor IS NULL;
```

```
cod  nom                dni_professor
---  -----------------  -------------
XAL  Xarxes area local  NULL
```

```
SELECT * FROM assignatures WHERE dni_professor IS NOT NULL;
```

```
cod  nom                    dni_professor
---  --------------------   -------------
BDD  Bases de dades         22222222A
DEC  Desenv. ent. client    22222223B
FOL  Form. orien. laboral   22222226E
ADA  Acces a dades          22222224C
PRO  Programacio            22222222A
```

5.2.10. Comprovacions amb conjunts de valors

Podem comparar una columna o una expressió amb una llista de valors fent servir els operadors IN i BETWEEN.

Operador IN: Ens permet comprovar si una expressió pertany o no a un conjunt de valors, podent fer comparacions múltiples.

Format: <EXPRESSIÓ>NOT IN (LLISTA DE VALORS SEPARATS PER COMES)

(La llista de valors pot estar formada per números o per cadenes de caràcters)

Exemple: consulta el nom a la taula MATRICULAT on l'alumne estiga matriculat a DEC o FOL.

```
SELECT * FROM matriculat
WHERE cod_assignatura IN ('DEC','FOL') ORDER BY cod_assignatura;
```

```
dni_alumne  cod_assignatura  curs_academic  nota
----------  ---------------  -------------  ----
11111111A   DEC              2020           7
11111115E   DEC              2020           9
11111112B   DEC              2019           8
11111113C   FOL              2020           4
11111114D   FOL              2020           9
11111116F   FOL              2020           4
```

Exemple: consulta el nom a la taula MATRICULADO on l'alumne no estiga matriculat a DEC o FOL.

```
SELECT * FROM matriculat
WHERE cod_assignatura NOT IN ('DEC','FOL')
ORDER BY cod_assignatura;
```

```
dni_alumne  cod_assignatura  curs_academic  nota
----------  ---------------  -------------  ----
11111111A   ADA              2020           7
11111111A   BDD              2020           5
11111113C   BDD              2020           3
11111114D   BDD              2017           6
11111115E   BDD              2020           3
```

Operador BETWEEN... AND: estableix una comparació dins un interval. També es pot fer servir NOT BETWEEN.

Format: EXPRESSIÓ [NOT] BETWEEN VALOR_INICIAL AND VALOR_FINAL

Exemple: Mostra els alumnes matriculats entre el 2017 i el 2021

```
SELECT * FROM matriculat WHERE curs_academic BETWEEN 2017 AND 2021;
```

```
dni_alumne  cod_assignatura  curs_academic  nota
----------  ---------------  -------------  ----
11111111A   ADA              2020           7
11111113C   FOL              2020           4
11111114D   FOL              2020           9
11111116F   FOL              2020           4
11111111A   DEC              2020           7
11111115E   DEC              2020           9
11111111A   BDD              2020           5
11111113C   BDD              2020           3
11111114D   BDD              2017           6
11111115E   BDD              2020           3
11111112B   DEC              2019           8
```

5.3. Relacions entre taules

Normalment ens sorgirà la necessitat d'associar una taula amb una altra. En aquest punt veurem quins tipus de relacions podem fer.

5.3.1. Producte cartesià

Per exemple, per a saber el nom dels alumnes que estan al curs 2020. Si veiem la taula de MATRICULAT, allà no hi ha el nom l'alumne sinó el seu DNI, el nom de l'Alumne apareix a la taula Alumnes, per això hem d'associar la taula MATRICULAT amb ALUMNES i relacionar-la. Per això caldria fer el següent:

Si realitzem la següent consulta:

```
SELECT * FROM matriculat, alumnes;
```

```
11111111A   ADA   2020   7    11111111A  JORDI      SANCHEZ   1980/10/10
11111111A   ADA   2020   7    11111112B  PACO       PEREZ     1980/11/10
11111111A   ADA   2020   7    11111113C  RAFA       ROMERO    1981/12/01
11111111A   ADA   2020   7    11111114D  JAVI       PONCE     1982/01/12
11111111A   ADA   2020   7    11111115E  MANOLI     ALVAREZ   1981/02/10
11111111A   ADA   2020   7    11111116F  MANUEL     PEREZ     1980/03/13
11111111A   ADA   2020   7    11111117G  ANA        SANCHEZ   1983/10/10
11111111A   ADA   2020   7    11111118H  ISABEL     PEREZ     1986/04/15
11111111A   ADA   2020   7    11111119I  ESTEFANIA  SANCHEZ   1985/10/10
11111113C   FOL   2020   4    11111111A  JORDI      SANCHEZ   1980/10/10
```

```
11111113C   FOL   2020   4    11111112B  PACO       PEREZ     1980/11/10
11111113C   FOL   2020   4    11111113C  RAFA       ROMERO    1981/12/01
(...)
```

Hi ha diverses pàgines a la consulta.

Com s'aprecia, ha fet una operació X (producte cartesià) de l'àlgebra relacional. És a dir, ha fet totes les combinacions de cada tupla (filera) de MATRICULAT amb cada tupla (filera) d'ALUMNES.

Per fer la consulta de forma correcta hauríem de fer el següent:

```
SELECT dni_alumne, nom, cognoms, curs_academic
FROM matriculat, alumnes
WHERE dni_alumne = dni
AND curs_academic = 2020;
```

```
dni_alumne  nom     cognoms  curs_academic
----------  ------  -------  -------------
11111111A   JORDI   SANCHEZ  2020
11111113C   RAFA    ROMERO   2020
11111114D   JAVI    PONCE    2020
11111116F   MANUEL  PEREZ    2020
11111111A   JORDI   SANCHEZ  2020
11111115E   MANOLI  ALVAREZ  2020
11111111A   JORDI   SANCHEZ  2020
11111113C   RAFA    ROMERO   2020
11111115E   MANOLI  ALVAREZ  2020
```

Com es pot apreciar apareixen repetits, això és perquè PACO, per exemple, està matriculat a les assignatures de DAC, DEG, ADA.

Per solucionar-ho i que només ens apareguessen una tupla per cada alumne, hauríem de fer el següent:

```
SELECT DISTINCT dni_alumne, nom, cognoms, curs_academic
FROM matriculat, alumnes
WHERE dni_alumne = dni
AND curs_academic = 2020;
```

```
dni_alumne  nom     cognoms  curs_academic
----------  ------  -------  -------------
11111111A   JORDI   SANCHEZ  2020
11111113C   RAFA    ROMERO   2020
11111114D   JAVI    PONCE    2020
11111115E   MANOLI  ALVAREZ  2020
11111116F   MANUEL  PEREZ    2020
```

5.3.2. Reunió de relacions

Inner join	Amb aquesta operació es calcula el producte creuat de tots els registres; així cada registre a la taula A és combinat amb cada registre de la taula B. Cal tenir especial cura quan es combinen columnes amb valors NULL ja que el valor nul no es combina amb un altre valor o amb un altre nul.

Natural inner join	En aquest cas es comparen totes les columnes que tinguen el mateix nom a les dues taules.
Left outer join	El resultat d'aquesta operació sempre conté tots els registres de la taula de l'esquerra (la primera taula que s'esmenta a la consulta), encara que no existeixi un registre corresponent a la taula de la dreta, per a un de l'esquerra.
Right outer join	Aquesta operació inversa a l'anterior; el resultat d'aquesta operació sempre conté tots els registres de la taula de la dreta (la segona taula que s'esmenta a la consulta), encara que no hi hagi un registre corresponent a la taula de l'esquerra, per a un de la dreta.

Exemples:

Tabla Empleado

Apellido	IDDepartamento
Rafferty	31
Jordán	33
Steinberg	33
Róbinson	34
Smith	34
Gaspar	36

Tabla Departamento

NombreDepartamento	IDDepartamento
Ventas	31
Ingeniería	33
Producción	34
Marketing	35

```
SELECT *
FROM empleado
INNER JOIN departamento
ON empleado.IDdepartamento = departamento.IDdepartamento
```

Donaria el mateix resultat que:

```
SELECT *
FROM empleado, departamento
WHERE empleado.IDdepartamento = departamento.IDDepartamento
```

Empleado.Apellido	Empleado.IDdepartamento	departamento.NombreDepartamento	departamento.IDDepartamento
Smith	34	Producción	34
Jordán	33	Ingeniería	33
Róbinson	34	Producción	34
Steinberg	33	Ingeniería	33
Rafferty	31	Ventas	31

```
SELECT *
FROM empleado NATURAL JOIN departament
```

Empleado.Apellido	IDDepartamento	Departamento.NombreDepartamento
Smith	34	Producción
Jordán	33	Ingeniería
Róbinson	34	Producción
Steinberg	33	Ingeniería
Rafferty	31	Ventas

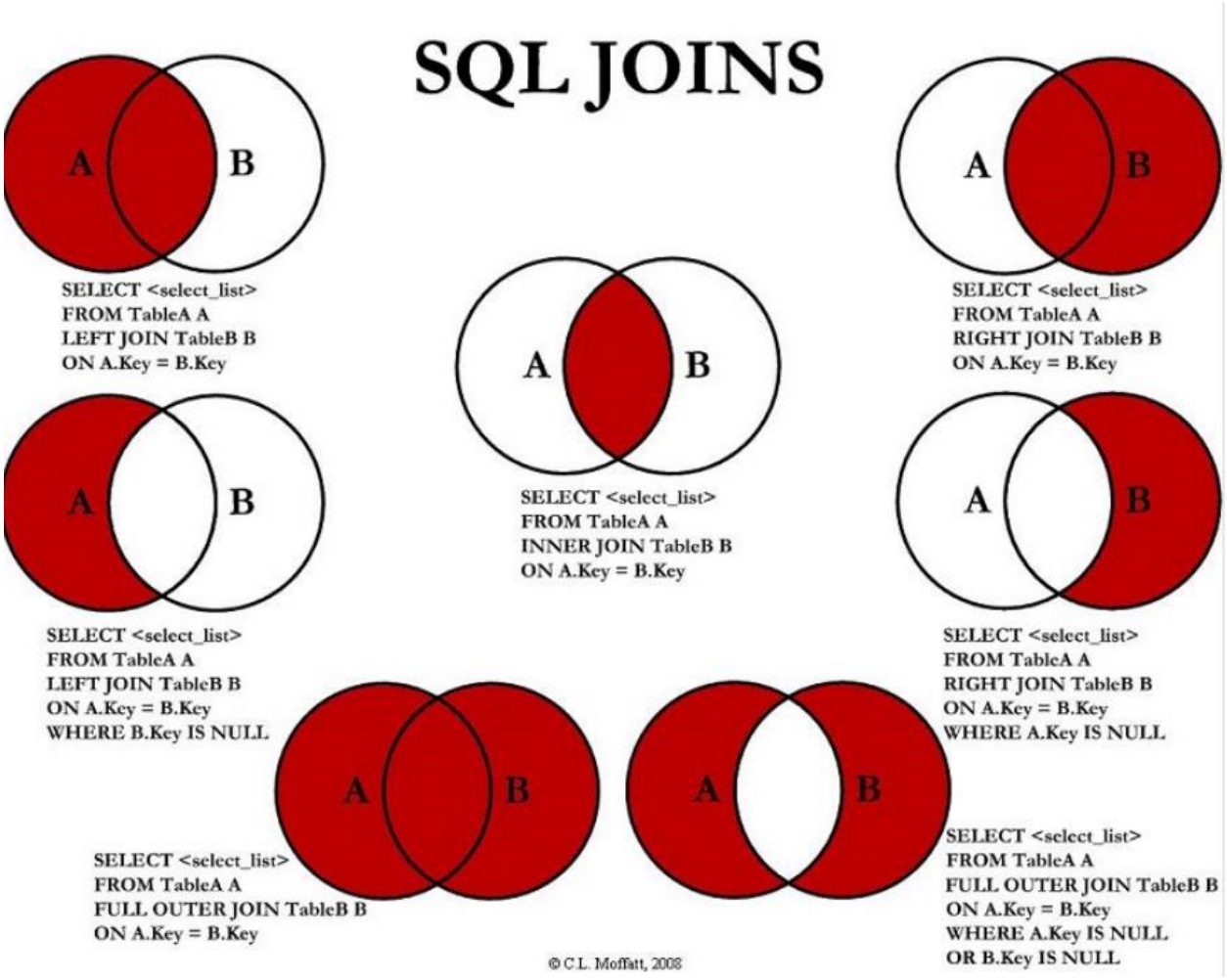
```
SELECT distinct * FROM empleado
LEFT OUTER JOIN departamento
ON empleat.IDDepartamento = departament.IDDepartamento
```

Empleado.Apellido	Empleado.IDDepartamento	Departamento.NombreDepartamento	Departamento.IDDepartamento
Jordán	33	Ingeniería	33
Rafferty	31	Ventas	31
Róbinson	34	Producción	34
Smith	34	Producción	34
Gaspar	36	NULL	NULL
Steinberg	33	Ingeniería	33

NOTA: La tupla Gaspar s'ha omplert amb valors nuls

```
SELECT *
FROM empleado
RIGHT OUTER JOIN departament
ON empleat.IDDepartament = departament.IDDepartamento
```

Empleado.Apellido	Empleado.IDDepartamento	Departamento.NombreDepartamento	Departamento.IDDepartamento
Smith	34	Producción	34
Jordán	33	Ingeniería	33
Róbinson	34	Producción	34
Steinberg	33	Ingeniería	33
Rafferty	31	Ventas	31
NULL	NULL	Marketing	35



A continuació teniu una llista de sinònims:

A LEFT JOIN B = A LEFT OUTER JOIN B

A RIGHT JOIN B = A RIGHT OUTER JOIN B

A FULL JOIN B = A FULL OUTER JOIN B

A INNER JOIN B = A JOIN B

5.4. Subconsultes

De vegades, per realitzar alguna operació de consulta, necessitem les dades tornades per una altra consulta. Per exemple, saber el nom de l'Alumne que està matriculat a DAC, com veiem estan en diferents taules. Una possible solució és fer servir el producte cartesià de dues taules. O també dir que ens mostren els alumnes que pertanyen al subconjunt d'alumnes que tenen DAC.

Aquest problema es pot resoldre usant una subconsulta, que no és més que una sentència SELECT dins una altra sentència SELECT Les subconsultes són aquelles sentències SELECT que formen part d'una clàusula WHERE d'una sentència SELECT anterior. Una subconsulta consistirà a incloure una declaració SELECT com a part duna clàusula WHERE.

Format de la subconsulta:

```
SELECT ...
FROM. ..
WHERE columna operador_comparatiu (SELECT...........
```

Primer es resoldrà el SELECT del parèntesi i després el SELECT de la sentència principal.

Exemples

```
SELECT * FROM alumnes
WHERE dni IN
  (SELECT dni_alumne FROM matriculat WHERE cod_assignatura='DEC');
```

```
dni         nom     cognoms   data_nac
---------   ------  -------   ----------
11111111A   JORDI   SANCHEZ   1980/10/10
11111112B   PACO    PEREZ     1980/11/10
11111115E   MANOLI  ALVAREZ   1981/02/10
```

Com es pot apreciar, és una consulta que dins té una altra subconsulta.

5.4.1. Condicions de cerca a subconsultes

Comparació de subconsulta (>, < ,<>, <=, >= , =) . Compareu el valor d'una expressió amb un valor únic produït per una subconsulta.

Exemple: obtenir aquells alumnes (DNI) que estiguen matriculats a DEC

```
SELECT dni_alumne FROM matriculat WHERE cod_assignatura =
(
  SELECT cod FROM assignatures WHERE
  cod = 'DEC'
```

```
);
```

```
dni_alumne
----------
11111111A
11111112B
11111115E
```

Pertinença a un conjunt retornat per una subconsulta (IN): Comproveu si el valor d'una expressió coincideix amb un del conjunt de valors produït per una subconsulta.

Exemple: obtenir aquells alumnes que tinguen classe amb ALVARO.

```
SELECT nom, cognoms FROM alumnes WHERE dni IN
(
      SELECT dni_alumne FROM matriculat WHERE cod_assignatura IN
      (
            SELECT cod FROM assignatures WHERE dni_professor IN
            (
                  SELECT dni FROM professors WHERE nom='ALVARO'
            )
      )
);
```

```
nom     cognoms
------  -------
JORDI   SANCHEZ
RAFA    ROMERO
JAVI    PONCE
MANOLI  ALVAREZ
```

Test de existència (Exists, not exists). Examina si una subconsulta produeix alguna fila de resultats. La consulta és True si torna files, si no és False.

Exemple: Llistar les assignatures que no tinguen professors

```
SELECT * FROM assignatures a WHERE NOT EXISTS (
  SELECT dni FROM professors p WHERE a.dni_professor = p.dni
);
```

```
cod  nom               dni_professor
---  ----------------  -------------
XAL  Xarxes area local  NULL
```

Exemple: Llistar els professors que no facen classe

```
SELECT * FROM professors p WHERE NOT EXISTS (
  SELECT * FROM assignatures a WHERE a.dni_professor = p.dni );
```

```
dni        nom   cognoms
---------  ----  -------
22222225D  PACO  ALMAGRO
```

Test de comparació quantificada (any ,all). Es fan servir a unió amb els operadors de comparació (>, < ,<>, <=, >= , =).

Ex. Obtenir les notes dels alumnes i assignatures, que tinguen més nota que alguna de les notes de DEC

```
SELECT * FROM matriculat
WHERE nota > ANY (
```

```
  SELECT nota FROM matriculat WHERE cod_assignatura='DEC');
```

```
dni_alumne  cod_assignatura  curs_academic  nota
----------  ---------------  -------------  ----
11111114D   FOL              2020           9
11111115E   DEC              2020           9
11111112B   DEC              2019           8
```

(Aquesta consulta es pot realitzar en Oracle, però potser no en altres SGBD més senzills com SQLite)

Ex. Obté un llistat de la nota més alta de cadascuna de les assignatures.

```
SELECT * FROM matriculat m1
WHERE nota >= ALL (
  SELECT nota FROM matriculat m2
  WHERE m1.cod_assignatura=m2.cod_assignatura);
```

```
dni_alumne  cod_assignatura  curs_academic  nota
----------  ---------------  -------------  ----
11111111A   ADA              2020           7
11111114D   BDD              2017           6
11111115E   DEC              2020           9
11111114D   FOL              2020           9
```

(Aquesta consulta es pot realitzar en Oracle, però potser no en altres SGBD més senzills com SQLite)

Exemple: Mostrar, de cada assignatura, l'alumne que té la màxima nota.

```
SELECT nom, cognoms, cod_assignatura, nota
FROM matriculat m, alumnes a
WHERE m.dni_alumne=a.dni AND nota >= ALL (
  SELECT nota FROM matriculat m2
    WHERE m.cod_assignatura=m2.cod_assignatura
);
```

```
nom    cognoms cod_assignatura nota
------ ------- --------------- ----
JORDI  SANCHEZ ADA             7
MANOLI ALVAREZ DAC             9
JAVI   PONCE   FOL             9
JAVI   PONCE   DEG             6
```

5.5. Funcions

Les funcions es fan servir dins d'expressions i actuen amb els valors de les columnes variables o constants. S'utilitzen a: clàusules select, clàusules where i clàusules order by.

És possible el nidament de funcions. Hi ha cinc tipus de funcions: aritmètiques, de cadena de caràcters, de maneig de dates de conversió i altres funcions.

5.5.1. Funcions aritmètiques

Treballen amb dades de tipus numèric NUMBER. Inclou els dígits de 0 al 9, el punt decimal el signe menys, si cal. Els literals numèrics no es tanquen entre cometes. Ex :-123.32

Aquestes funcions treballen amb tres classes de números: valors simples, grups de valors i llista de valors. Algunes modifiquen els valors sobre les quals actuen; altres informen d'alguna cosa sobre els valors. Podem dividir les funcions aritmètiques en tres grups:

***Funcions de valors simples**:* són funcions senzilles que treballen amb valors simples. Un valor simple és: un número, una variable o una columna d'una taula. Les funcions de valors simples són:

Funció	Propòsit
Abs(n)	Retorna el valor absolut de "n". El valor absolut és sempre positiu
Mod(m,n)	Retorna la resta resultant de dividir "m" entre "n"
Power(n,exponent)	Calculeu la potència d'un nombre. torna el valor de m elevat a un exponent
Round(nombre[,m])	Retorna el valor de nombre arrodonit a "m" decimals. Si "m" és negatiu, l'arrodoniment de dígits es fa a l'esquerra del punt decimal. Si s'omet "m", torna número amb 0 decimals i arrodonit
Sqrt(n)	Retorna l'arrel quadrada de n. El valor de n no pot ser negatiu.
Trunc(número,[n])	Trunca els números perquè tinguen un cert nombre de dígits de precisió .torna "número" truncat a "m" decimals;"m" pot ser negatiu; si ho és, trunca per l'esquerra del punt decimal. Si s'omet "m" torna "número" amb 0 decimals
NVL(valor, expressió)	Substituïu un valor NUL per un altre valor

Funcions de grups de valors: Actuen sobre un grup de files per obtenir un valor. Els valors nuls són ignorats per aquest tipus de funcions i els càlculs es realitzen sense comptar-hi. Les clàusules DISTINCT i ALL es poden utilitzar amb totes, tot i que generalment es fa servir amb COUNT.

Aquestes funcions són:

Funció	Propòsit
AVG(n)	Calcula el valor mitjà de "n" ignorant els valors nuls
COUNT(* | [DISTINCT|ALL] expressió)	Compte el nombre de vegades que l'expressió avalua alguna dada amb valor no nul. L'opció "*" compta totes les files seleccionades
MAX(expressió)	Calcula el màxim valor de la "expressió"
MIN(expressió)	Calcula el mínim valor de la "expressió"
SUM(expressió)	Obté la suma de valors de la "expressió" diferents de nuls

Funcions de llistes: Treballen sobre un grup de columnes dins una mateixa fila. Compareu els valors de cadascuna de les columnes a l'interior d'una fila per obtenir el major o menor valor de la llista.

Aquestes funcions són:

Funció	Propòsit

Greatest (valor1,valor2,...)	Obté el valor més gran de la llista.
Least(valor1,valor2,.)	Obté el menor valor de la llista.

5.5.2. Funcions de cadenes de caràcters

Treballen amb dades de tipus char o varchar2. Permeten manipular cadenes de lletres o altres caràcters. Aquestes funcions poden calcular el nombre de caràcters d'una cadena, convertir una cadena a majúscules o minúscules o afegir caràcters a l'esquerra o a la dreta, etc.

Funció	Propòsit
Chr(n)	Retorna el caràcter el valor del qual en binari és equivalent a "n"
Concat(cad1,cad2..)	Retorna "cad1" concatenada amb "cad2" és equivalent a l'operador ||
Lower(cad)	Retorna la cadena "cad" amb totes les lletres convertides a minúscules
Upper(cad)	Retorna la cadena cad en majúscules.
Lpad(text, ampladaMàxima, [caràcter_farcit])	Omple el text a l'esquerra amb el caràcter indicat fins a obtenir l'amplada indicada
Rpad(text, ampladaMàxima, [caràcter_farcit])	Omple el text a la dreta amb el caràcter indicat fins a obtenir l'amplada indicada
Ltrim(cad,[set])	Suprimeix un conjunt de caràcters a la esquerra de la cadena
Rtrim(cad,[set])	Suprimiu un conjunt de caràcters a la dreta de la cadena
REPLACE(cad, cadena_cerca [,cadena_sustitució])	Substituïu un caràcter o caràcters d'una cadena amb 0 o més caràcters.
SUBSTR (cad, m [,n])	Obté part duna cadena. Des de la posició m fins a n (si no s'indica, fins al final)

5.5.3. Funcions que tornen valors numèrics

Funció	Propòsit
ASCII(cad)	Retorna el valor ASCII de la primera lletra de la cadena "cad"
Instr(cad1,cad2[,començament[,m]])	Cerca un conjunt de caràcters en una cadena. Retorna la posició de la "m_èsima" ocurrència de cad2 en cad1, començant la cerca en la posició començament. Per omissió comença buscant a la posició 1
Length(cad)	Retorna el nombre de caràcters de "cad"

5.5.4. Funcions per al maneig de dates

Oracle pot emmagatzemar dades de tipus data (date), té un format per defecte: "DD/MM/YY", però amb la funció TO_CHAR és possible mostrar les dates de qualsevol manera. Els literals de data han de tancar-se entre cometes simples. EXEMPLE: '18/11/05'

Funció	Propòsit
SYSDATE	Retorna la data del sistema
ADD_MONTHS(data,n)	Retorna la data "data" incrementada en 'n' mesos.
LAST_DAY(data)	Retorna la data del darrer dia del mes que conté "data"
MONTHS_BEETWEN(data1,data2)	Retorna la diferència en mesos entre les dates "data1" i "data2"
NEXT_DAY(data, cad)	Retorna la data del primer dia de la setmana indicat per cad, després de la data indicada per data .El dia de la setmana a 'cad' s'indica amb el seu nom, és a dir, dilluns (Monday)…

5.5.5. Funcions de conversió

Funció	Propòsit
TO_CHAR(camp, format)	Transforma un tipus DATE o NUMBER a una cadena de caràcters. Obté un text a partir d'un número o data.

TO_DATE(camp,format_DATA)	Transforma un tipus NUMBER o CHAR a DATE. Converteix textos en dates.
TO_NUMBER(camp, format)	Transforma una cadena de caràcters a NUMBER. Converteix textos a números.

5.5.6. Funció especial DECODE

Funció	Propòsit
DECODE(expressió,valor1,resultat1,[,valor2,resultat 2,…,[valorPerDefecte]	Funció que permet avaluar condicions en una consulta
Exemple: DECODE(nota,10,'Excel·lent',9,'Excel·lent',8,'Notable'…5,'Aprovat','Suspès')	

I.E.S. «SANT VICENT»
Codi centre: 03008423
www.iessanvicente.com
03008423@edu.gva.es

C/ Lillo Juan, 128
03690 Sant Vicent del Raspeig
Telf.: 965 93 65 05
Fax: 965 93 65 06

5.6. Clàusules avançades de selecció

Estudiarem les sentències que ens permeten agrupar files d'una taula segons alguna condició, o sense cap condició, per obtenir algun resultat referent a aquest grup de files agrupades: GROUP BY.

També utilitzarem les sentencies que ens permeten seleccionar algunes files d'una taula, encara que aquestes no tinguen la correspondència amb una altra taula: HAVING

5.6.1. Group by

La sentència SELECT permet utilitzar un o més conjunts de files. L'agrupament es duu a terme mitjançant la clàusula GROUP BY per les columnes especificades i en l'ordre especificat. La sentència SELECT tindrà el format:

```
SELECT ... FROM ...
GROUP BY columna1, columna2, columna3,...
HAVING condició
ORDER BY ...
```

Les dades seleccionades a la sentència SELECT que porta el GROUP BY han de ser: una constant, una funció de grup (SUM, COUNT, AVG,..), una columna expressada al GROUP BY.

La clàusula GROUP BY serveix per calcular propietats d'un o més conjunts de files. A més, si se selecciona més d'un conjunt de files, GROUP BY controla que les files de la taula original siguen agrupades en una temporal.

Exemples

Exemple: Mostrar la nota mitjana de cada assignatura.

```
SELECT cod_assignatura, AVG(nota) AS mitjana
FROM matriculat
GROUP BY cod_assignatura
ORDER BY cod_assignatura;
```

```
cod_assignatura  mitjana
---------------  ----------------
ADA              7.0
BDD              4.25
DEC              8.0
FOL              5.66666666666667
```

Exemple: Mostra la nota mitjana dels alumnes que han aprovat.

```
SELECT cod_assignatura, AVG(nota) AS mitjana
FROM matriculat
GROUP BY cod_assignatura
HAVING AVG(nota)>=5 ORDER BY cod_assignatura;
```

I.E.S. «SANT VICENT»
Codi centre: 03008423
www.iessanvicente.com
03008423@edu.gva.es
C/ Lillo Juan, 128
03690 Sant Vicent del Raspeig
Telf.: 965 93 65 05
Fax: 965 93 65 06

```
cod_assignatura  mitjana
---------------  ----------------
ADA              7.0
DEC              8.0
FOL              5.66666666666667
```

Exemple: Mostrar l'assignatura i quants alumnes hi ha matriculats per al curs 2020.

```
SELECT cod_assignatura, COUNT(*) AS num_alumnes
FROM matriculat
WHERE curs_academic=2020 GROUP BY cod_assignatura
ORDER BY num_alumnes DESC;
```

```
cod_assignatura  num_alumnes
---------------  -----------
FOL              3
BDD              3
DEC              2
ADA              1
```

5.6.2. Having

HAVING és semblant a WHERE, determina quins registres se seleccionen una vegada agrupats. Quan els registres s'han agrupat utilitzant GROUP BY, HAVING determina quins es mostraran.

S'avalua sobre la taula que torna el GROUP BY. No hi pot haver sense GROUP BY.

Exemple: Mostrar la nota mitjana de cada assignatura que siga mes gran o igual que 5.

```
SELECT cod_assignatura, AVG(NOTA) AS mitjana FROM matriculat
GROUP BY cod_assignatura
HAVING AVG(nota) >= 5
ORDER BY cod_assignatura;
```

Exemple: Mostra l'assignatura amb la nota més baixa.

```
SELECT cod_assignatura, MIN(nota)
FROM matriculat
GROUP BY cod_assignatura
HAVING MIN(nota) <= ALL(
  SELECT MIN(nota) FROM matriculat GROUP BY cod_assignatura);
```

5.7. Operacions sobre conjunts

5.7.1. Unió

Uneix diversos conjunts, però ambdues tuples a unir han de tenir el mateix nombre i tipus. UNION elimina duplicats, per conservar els duplicats usaríem UNION ALL.

I.E.S. «SANT VICENT»
Codi centre: 03008423
www.iessanvicente.com
03008423@edu.gva.es
C/ Lillo Juan, 128
03690 Sant Vicent del Raspeig
Telf.: 965 93 65 05
Fax: 965 93 65 06

Exemple: Mostra un llistat amb alumnes i professors, mostrant el seu DNI, Nom i Cognoms

```
SELECT dni,nom,cognoms FROM alumnes
UNION
SELECT dni,nom,cognoms FROM professors;
```

```
dni         nom         cognoms
---------   ---------   -------
11111111A   JORDI       SANCHEZ
11111112B   PACO        PEREZ
11111113C   RAFA        ROMERO
11111114D   JAVI        PONCE
11111115E   MANOLI      ALVAREZ
11111116F   MANUEL      PEREZ
11111117G   ANA         SANCHEZ
11111118H   ISABEL      PEREZ
11111119I   ESTEFANIA   SANCHEZ
22222222A   ALVARO      LOPEZ
22222223B   PEDRO       SANCHEZ
22222224C   MAITE       PEREZ
22222225D   PACO        ALMAGRO
22222226E   FERMINA     AGUILAR
```

5.7.2. Intersect

Dóna el resultat de la intersecció de diversos conjunts, però ambdues tuples a unir han de tenir el mateix nombre i tipus. INTERSECT elimina duplicats, per conservar els duplicats "INTERSECT ALL"

Exemple: Mostra un llistat de noms on es mostre quins professors s'anomenen igual que els alumnes

```
SELECT nom FROM alumnes
INTERSECT
SELECT nom FROM professors;
```

```
nom
----
PACO
```

5.7.3. Minus / Except

Es relaciona amb l'operació '-' sobre conjunts. L'estàndard SQL proposa la paraula EXCEPT, però Oracle permet també usar la paraula MINUS:

Exemple: Mostra quins alumnes estan matriculats a BDD i, d'ells, treu els que estan matriculats també a DEC.

```
SELECT dni, nom, cognoms FROM alumnes
WHERE dni IN(
  (SELECT dni_alumne FROM matriculat WHERE cod_assignatura='BDD')
  EXCEPT
  (SELECT dni_alumne FROM matriculat WHERE cod_assignatura='DEC')
);
```

```
dni         nom    cognoms
```

I.E.S. «SANT VICENT»
Codi centre: 03008423
www.iessanvicente.com
03008423@edu.gva.es

C/ Lillo Juan, 128
03690 Sant Vicent del Raspeig
Telf.: 965 93 65 05
Fax: 965 93 65 06

```
---------  ----  -------
11111113C  RAFA  ROMERO
11111114D  JAVI  PONCE
```

Que també es podria escriure (només en Oracle) així:

```
SELECT dni, nom, cognoms FROM alumnes
WHERE dni IN(
  (SELECT dni_alumne FROM matriculat WHERE cod_assignatura='BDD')
  MINUS
  (SELECT dni_alumne FROM matriculat WHERE cod_assignatura='DEC')
);
```

5.8. Inserció de dades

A partir d'una taula com esta, que ja havíem proposat anteriorment com a exemple:

```
CREATE TABLE professors (
    dni VARCHAR2(20) PRIMARY KEY,
    nom VARCHAR2(20),
    cognoms VARCHAR2(40)
);
```

És possible introduir dades en format abreujat si s'indiquen valors per a tots els camps, en el mateix orde en què es van definir, així:

```
INSERT INTO professors VALUES ('22222222A','ALVARO','LOPEZ');
```

Com a alternativa, es poden introduir només alguns camps, o bé les dades en un orde que no siga el de definició, si els camps es detallen abans de "VALUES", així:

```
INSERT INTO professors (dni,nom) values ('22222222A','ALVARO');
```

```
INSERT INTO professors (nom,cognoms,dni) values ('ALVARO','LOPEZ',
'22222222A');
```

Com a ús més avançat, és possible inserir dades obtinguts d'una altra taula, sempre que les dades obtingudes siguen compatibles amb el format de la taula de destí:

```
INSERT INTO professors
SELECT dni,nom,cognoms FROM backupProfessors
WHERE grupDeTutoria ='1ESOA';
```

I.E.S. «SANT VICENT»
Codi centre: 03008423
www.iessanvicente.com
03008423@edu.gva.es
C/ Lillo Juan, 128
03690 Sant Vicent del Raspeig
Telf.: 965 93 65 05
Fax: 965 93 65 06

5.9. Modificació de dades

És possible modificar dades que s'havien introduït prèviament, emprant la sentència UPDATE, amb el següent format:

```
UPDATE taula SET camp = nouValor WHERE condició;
```

Per exemple:

```
UPDATE professors SET nom = 'ARTURO' WHERE dni = '22222222A';
```

També es pot modificar el valor de més d'un camp alhora:

```
UPDATE professors SET nom = 'ARTURO', cognoms = 'MARTINEZ'
WHERE dni = '22222222A';
```

És possible ometre el WHERE, si es desitja fer una modificació en tots els registres de la taula

```
UPDATE professors SET salari = salari * 1.05;
```

5.10. Esborrament de dades

Per a esborrar dades s'haurà d'emprar l'orde DELETE. La clàusula WHERE és opcional, però ometre-la suposaria que s'esborraren totes les dades de la taula:

```
DELETE FROM taula WHERE condició;
```

Per exemple:

```
DELETE FROM professors WHERE dni = '22222222A';
```

5.11. Creació d'una taula a partir de dades existents

És possible crear una taula (només l'estructura bàsica, sense clau primari ni cap altre tipus de "constraints") a partir de dades existents, amb una variant del comando CREATE TABLE que pren com a base una sentència SELECT:

```
CREATE TABLE professors1eso AS
SELECT dni,nom,cognoms FROM backupProfessors
WHERE grupDeTutoria LIKE '1ESO%';
```

I.E.S. «SANT VICENT»
Codi centre: 03008423
www.iessanvicente.com
03008423@edu.gva.es

C/ Lillo Juan, 128
03690 Sant Vicent del Raspeig
Telf.: 965 93 65 05
Fax: 965 93 65 06

Alguns gestors de bases de dades permeten també aconseguir el mateix emprant l'estructura SELECT - INTO:

```
SELECT dni,nom,cognoms
INTO professors1eso
FROM backupProfessors
WHERE grupDeTutoria LIKE '1ESO%';
```