Titulación: GRADO INGENIERIA INFORMATICA
Año Académico: 2015/2016
Curso: 2º
Asignatura: Ficheros y Bases de Datos
Título: Memoria Práctica 2 – Consultas, vistas y disparadores

UNIVERSIDAD CARLOS III DE MADRID

Profesor:	**María Dolores Cuadra Fernández**	**Grupo**	**50**
Alumno/a:	**Javier Bautista Rosell**	**NIA:**	**100315139**
Alumno/a:	**Lizaveta Mishkinitse**	**NIA:**	**100315144**

1. Introducción

En esta práctica se realiza la implementación de vistas y disparadores que completan la semántica no contemplada en la práctica anterior, así como el diseño en álgebra relacional de consultas a la base de datos proporcionada.

Con la realización de este ejercicio se busca familiarizarnos con el lenguaje SQL e implementar con la mayor precisión posible las consultas solicitadas.
Este documento tiene 5 partes: la introducción, las consultas implementadas, las vistas, los disparadores, las vistas y disparadores adicionales que también completan la semántica y finalmente la conclusión.

Se ha incluido el código de cada una de las consultas en este documento (además se incluirá el código en archivos de .sql aparte) ya que el enunciado no especificó con claridad si se debía de incluir dicho código en la memoria o no.

2. Consultas

a) Empleados en activo que han sido despedidos anteriormente.

a. Diseño en álgebra relacional

$$\Pi_{SSN}(\sigma_{END\_DATE>SYSDATE\ OR\ END\_DATE=NULL}(CONTRACTS)$$
$$\cap$$
$$\sigma_{CEASING\ !=\ NULL\ AND\ CEASING\ !='\ Contract\ termination\ due\ to\ new\ contract'}(CONTRACTS))$$

b. Su implementación en SQL

```
SELECT SSN
FROM CONTRACTS
WHERE END_DATE>SYSDATE OR END_DATE IS NULL
INTERSECT
SELECT SSN
FROM CONTRACTS
WHERE CEASING IS NOT NULL AND CEASING NOT LIKE 'Contract%';
```

- Resultado: 75 filas

c. Pruebas realizadas

```
-- Creación de otro empleado con contrato activo que ha
sido despedido anteriormente --

insert into employees values ('1234');
insert into contracts
values('1234','08/04/2015', '20/05/2015','6',‘End of
probation period');
insert into contracts
values('1234','08/04/2016', '20/05/2016','6','null');
```

- Resultado: 76 filas

b) Proyectos no finalizados cuyo gasto ya se ha excedido de su presupuesto.

a. Diseño en álgebra relacional

$$\text{GROUP BY}_{\text{IDPROJECT, BUDGET, HAVING TOTAL>BUDGET}}\Big($$
$$\Pi_{\text{PROJECTS.IDPROJECT, PROJECTS.BUDGET, SUM(COSTS.SALARY/160) AS TOTAL}}\Big($$
$$\sigma_{\text{PROJECTS.END\_DATE>SYSDATE AND CONTRACTS.START\_DATE<REP\_LINES.REP\_DATE AND (CONTRACTS.END\_DATE = NULL OR CONTRACTS.END\_DATE>REP\_LINES.REP\_DATE) AND COSTS.YEAR REP\_LINES.REP\_DATE(YEAR)}}\Big($$
$$(\text{REP\_LINES}) \theta_{\text{REP\_LINES.IDPROJECT = PROJECTS.IDPROJECT}}$$
$$(\text{PROJECTS}) \theta_{\text{REP\_LINES.SSN=CONTRACTS.SSN}}$$
$$(\text{CONTRACTS}) \theta_{\text{CONTRACTS.CAT\_NUM=COSTS.CAT\_NUM}}$$
$$(\text{COSTS})))$$

b. Su implementación en SQL

```
SELECT PROJECTS.IDPROJECT, PROJECTS.BUDGET,
SUM(COSTS.SALARY/160) AS BALANCE
FROM REP_LINES
INNER JOIN PROJECTS
ON REP_LINES.IDPROJECT = PROJECTS.IDPROJECT
INNER JOIN CONTRACTS
ON REP_LINES.SSN = CONTRACTS.SSN
INNER JOIN COSTS
ON CONTRACTS.CAT_NUM = COSTS.CAT_NUM
WHERE PROJECTS.END_DATE>SYSDATE AND
CONTRACTS.START_DATE<REP_LINES.REP_DATE AND
(CONTRACTS.END_DATE IS NULL OR
CONTRACTS.END_DATE>REP_LINES.REP_DATE) AND
 COSTS.YEAR = TO_NUMBER(TO_CHAR(REP_LINES.REP_DATE,
'YYYY'))
GROUP BY PROJECTS.IDPROJECT, PROJECTS.BUDGET
HAVING SUM(COSTS.SALARY/160)>BUDGET;
```

- Resultado: 1 fila

c. Pruebas realizadas

```
--Se crea un nuevo proyecto (anteriormente se crea el
manager)--
insert into employees
values ('E_PRUEBA1');
insert into contracts values('E_PRUEBA1','04/04/2016',
'20/05/2016','6','null');

insert into projects
values('EX/PRUEBA', 'PROYECTO PRUEBA','10', '5',
'04/04/2016','03/04/2022','E_PRUEBA1');
--Se crea otro empleado --
insert into employees values ('E_PRUEBA2');
insert into contracts values('E_PRUEBA2','08/04/2016',
'20/05/2016','6','null');
--Se crea jefe de equipo--
insert into employees values ('E_PRUEBA3');
insert into contracts values('E_PRUEBA3','08/04/2016',
'20/05/2016','6','null');
--Se crea equipo
insert into teams values('EX/PRUEBA','equipo
alfa','E_PRUEBA3');
insert into memberships
values('EX/PRUEBA','equipo alfa',
'E_PRUEBA2','11/04/2016','24/04/2016',6,'50');

--Se crean varias entradas correspondientes al nuevo
Proyecto en REP_LINES--
insert into reports values('E_PRUEBA2',
'11/04/2016','1234');
insert into rep_lines
values('E_PRUEBA2','11/04/2016','2',
'11','EX/PRUEBA','equipo alfa','6','Analysis');
insert into rep_lines
values('E_PRUEBA2','11/04/2016','2',
'12','EX/PRUEBA','equipo alfa','6','Validation of module
49 at subsystem 08');
```

- Resultado: 2 filas

IDPROJECT	BUDGET	BALANCE
EX/PRUEBA	10	34,7375
IN/GRALEXPENSES	100	27646,4

c) Empleados activos en el mes de Abril de 2014 que no cogieron vacaciones ese mes.

a. Diseño en álgebra relacional

$$\sigma_{\text{CONTRACTS.END\_DATE >= '2014\_04\_01' AND CONTRACTS.START\_DATE <= '2014\_04\_30'}} ($$
$$(\text{REP\_LINES})_{\text{REP\_LINES.SSN = CONTRACTS.SSN}}\ \theta$$
$$(\text{CONTRACTS})) \text{ AS A}$$
$$\sigma_{\text{REP\_LINES.TASK = 'I am on vacation! :-)' AND REP\_DATE+(DAY-1) >= '2014\_04\_01' AND REP\_DATE+(DAY-1) <= '2014\_04\_30'}} ($$
$$(\text{REP\_LINES})_{\text{REP\_LINES.SSN = CONTRACTS.SSN}}\ \theta$$
$$(\text{CONTRACTS})) \text{ AS B}$$
$$\Pi_{\text{SSN}}(\text{A-B})$$

b. Su implementación en SQL

```
SELECT DISTINCT REP_LINES.SSN
FROM REP_LINES
INNER JOIN CONTRACTS
ON REP_LINES.SSN = CONTRACTS.SSN
WHERE CONTRACTS.END_DATE >= TO_DATE('2014_04_01',
'YYYY_MM_DD') AND CONTRACTS.START_DATE <=
TO_DATE('2014_04_30', 'YYYY_MM_DD')
MINUS
SELECT DISTINCT REP_LINES.SSN
FROM REP_LINES
WHERE TASK LIKE 'I am on %' and REP_DATE+(DAY-1) >=
TO_DATE('2014_04_01', 'YYYY_MM_DD') AND REP_DATE+(DAY-1)
<= TO_DATE('2014_04_30', 'YYYY_MM_DD');
```

- Resultado: 9 filas

c. Pruebas realizadas

```
--Actualizar unos de los empleados que trabajaba en
Abril de 2014 para que este de vacaciones–

update rep_lines
set task='I am on vacation! :-) '
where ssn='93/48139083/08T' and task not like 'I am on
%' and extract(month from rep_date) = 4 and
extract(year from rep_date) = 2014;
```

- Resultado: 8 filas

d) Cuenta total de resultados (balance) por proyecto, ordenada por fecha de finalización.

a. Diseño en álgebra relacional

Π REP_LINES.SSN, COSTS.SALARY/160 AS SALARIO, REP_LINES.IDPROJECT, REP_LINES.JOB, REP_LINES.REP_DATE (

σ COSTS.YEAR REP_LINES.REP_DATE(YEAR) AND CONTRACTS.START_DATE<REP_LINES.REP_DATE AND (CONTRACTS.END_DATE = NULL OR CONTRACTS.END_DATE>REP_LINES.REP_DATE) (

(REP_LINES) REP_LINES.SSN = CONTRACTS.SSN θ

(CONTRACTS) COSTS.CAT_NUM = CONTRACTS.CAT_NUM θ

(COSTS))) AS SALARIO_EMPLEADO

Π IDPROJECT, END_DATE, BALANCE(

ORDER BY PROJECTS.END_DATE (

GROUP BY SALARIO_EMPLEADO.IDPROJECT, PROJECTS.END_DATE(

Π SALARIO_EMPLEADO.IDPROJECT, PROJECTS.END_DATE, SUM(COSTS.WAGE-SALARIO) AS BALANCE(

σ COSTS.YEAR REP_LINES.REP_DATE(YEAR)(

(SALARIO_EMPLEADO) SALARIO_EMPLEADO.JOB = COSTS.CAT_NUM θ

(COSTS) SALARIO_EMPLEADO.IDPROJECT = PROJECTS.IDPROJECT θ

(PROJECTS)))))

b. Su implementación en SQL

```
SELECT IDPROJECT, END_DATE, TO_CHAR(TOTAL, '9,999,999.99')
AS BALANCE
FROM(
WITH SALARIO_EMPLEADO AS(
SELECT REP_LINES.SSN, COSTS.SALARY/160 AS SALARIO,
REP_LINES.IDPROJECT, REP_LINES.JOB, REP_LINES.REP_DATE
FROM REP_LINES
INNER JOIN CONTRACTS
ON REP_LINES.SSN = CONTRACTS.SSN
INNER JOIN COSTS
ON COSTS.CAT_NUM = CONTRACTS.CAT_NUM
WHERE EXTRACT(YEAR FROM REP_LINES.REP_DATE)=COSTS.YEAR AND
CONTRACTS.START_DATE<REP_LINES.REP_DATE AND
(CONTRACTS.END_DATE IS NULL OR
CONTRACTS.END_DATE>REP_LINES.REP_DATE))
SELECT SALARIO_EMPLEADO.IDPROJECT, PROJECTS.END_DATE,
SUM(COSTS.WAGE-SALARIO) AS TOTAL
FROM SALARIO_EMPLEADO
INNER JOIN COSTS
ON SALARIO_EMPLEADO.JOB = COSTS.CAT_NUM
INNER JOIN PROJECTS
ON SALARIO_EMPLEADO.IDPROJECT = PROJECTS.IDPROJECT
WHERE EXTRACT(YEAR FROM
SALARIO_EMPLEADO.REP_DATE)=COSTS.YEAR
GROUP BY SALARIO_EMPLEADO.IDPROJECT, PROJECTS.END_DATE
ORDER BY PROJECTS.END_DATE);
```

- Resultado: 716 filas

c. Pruebas realizadas

```
--Se crea un nuevo proyecto (anteriormente se crea el
manager)--
insert into employees
values ('E_PRUEBA1');
insert into contracts values('E_PRUEBA1','04/04/2016',
'20/05/2016','6','null');

insert into projects
values('EX/PRUEBA', 'PROYECTO PRUEBA','10', '5',
'04/04/2016','03/04/2022','E_PRUEBA1');
--Se crea otro empleado --
insert into employees values ('E_PRUEBA2');
```

```
insert into contracts values('E_PRUEBA2','08/04/2016',
'20/05/2016','6','null');
--Se crea jefe de equipo--
insert into employees values ('E_PRUEBA3');
insert into contracts values('E_PRUEBA3','08/04/2016',
'20/05/2016','6','null');
--Se crea equipo
insert into teams values('EX/PRUEBA','equipo
alfa','E_PRUEBA3');
insert into memberships
values('EX/PRUEBA','equipo alfa',
'E_PRUEBA2','11/04/2016','24/04/2016',6,'50');

--Se crean varias entradas correspondientes al nuevo
Proyecto en REP_LINES--
insert into reports values('E_PRUEBA2',
'11/04/2016','1234');
insert into rep_lines
values('E_PRUEBA2','11/04/2016','2',
'11','EX/PRUEBA','equipo alfa','6','Analysis');
insert into rep_lines
values('E_PRUEBA2','11/04/2016','2',
'12','EX/PRUEBA','equipo alfa','6','Validation of module
49 at subsystem 08');
```

- Resultado: 717 filas

```
EX/PRUEBA       03/04/22          44.66

IDPROJECT       END_DATE BALANCE
--------------- -------- -------------
IN/GRALEXPENSES 22/11/11     35,532.96
IN/OWNITSYSTEMS 22/11/11    329,162.70

 717 filas seleccionadas
```

e) Empleados con varios periodos de contratación no contiguos.

a. Diseño en álgebra relacional

$$\Pi_{\text{DISTINCT SSN}} ($$
$$\sigma_{\text{COUNT(SSN)>1 AND COUNT(CEASING='Contract termination due to new contract') <COUNT(SSN)-1}}(\text{CONTRACTS})$$
$$\cap$$
$$\sigma_{\text{CEASING !=' Contract termination due to new contract'}}(\text{CONTRACTS}))$$

b. Su implementación en SQL

```
SELECT SSN
FROM(
SELECT DISTINCT SSN, COUNT(SSN), COUNT(CASE WHEN CEASING
LIKE 'Contract%' THEN 1 ELSE NULL END)
FROM CONTRACTS
GROUP BY SSN
HAVING COUNT(SSN)>1 AND COUNT(CASE WHEN CEASING LIKE
'Contract%' THEN 1 ELSE NULL END)<COUNT(SSN)-1)
INTERSECT
SELECT SSN
FROM CONTRACTS
WHERE CEASING NOT LIKE 'Contract%';
```

- Resultado: 94 filas

c. Pruebas realizadas

```
--Añadir un nuevo contrato no contiguo a un empleado que
únicamente tenia 1 contrato previo—

insert into contracts
values('00/13270684/05T','08/04/2016',  '20/05/2016', '6',
'null');
```

- Resultado: 95 filas

```
--Comprobacion de que los contratos sin motivo de ceasing '
Contract termination due to new contract' no están con
fechas contiguas--

select distinct B.ssn
from contracts A join contracts B
on A.ssn=B.ssn
where (A.END_DATE+1)=B.Start_date and A.ceasing not like
'Contract%';
```

- Resultado: 0 filas

3. Vistas

a) Vista plantilla: Balance (beneficio/pérdida) cada empleado.

a. Diseño en algebra relacional

$\Pi_{\text{REP\_LINES.SSN, REP\_LINES.JOB, REP\_LINES.REP\_DATE AS FECHA, COSTS.SALARY/160 AS SALARIO, REP\_LINES.TASK}}$ (
$\sigma_{\text{COSTS.YEAR = REP\_LINES.REP\_DATE(YEAR) AND CONTRACTS.START\_DATE<REP\_LINES.REP\_DATE AND (CONTRACTS.END\_DATE = NULL OR CONTRACTS.END\_DATE>REP\_LINES.REP\_DATE)}}$ (
(REP_LINES) $_{\text{REP\_LINES.SSN = CONTRACTS.SSN}}\theta$
(CONTRACTS) $_{\text{CONTRACTS.CAT\_NUM = COSTS.CAT\_NUM}}\theta$
(COSTS))) AS SALARIO_EMPLEADO

$\Pi_{\text{SSN, TOTAL}}$ (
GROUP BY SSN (
$\sigma_{\text{FECHA(YEAR)=COSTS.YEAR}}$ (
(SALARIO_EMPLEADO) $_{\text{SALARIO\_EMPLEADO.JOB = COSTS.CAT\_NUM}}\theta$
(COSTS))))

b. Implementación en SQL

```
CREATE OR REPLACE VIEW V_PLANTILLA(SSN,BENEFICIO) AS
SELECT SSN, TO_CHAR(TOTAL, '9,999,999.99')
FROM(
WITH SALARIO_EMPLEADO AS(
SELECT REP_LINES.SSN, REP_LINES.JOB, REP_LINES.REP_DATE AS
FECHA, COSTS.SALARY/160 AS SALARIO, REP_LINES.TASK
FROM REP_LINES
INNER JOIN CONTRACTS
ON REP_LINES.SSN = CONTRACTS.SSN
INNER JOIN COSTS
ON CONTRACTS.CAT_NUM = COSTS.CAT_NUM
WHERE EXTRACT (YEAR FROM REP_LINES.REP_DATE)=COSTS.YEAR AND
CONTRACTS.START_DATE<REP_LINES.REP_DATE AND
(CONTRACTS.END_DATE IS NULL OR
CONTRACTS.END_DATE>REP_LINES.REP_DATE) )
SELECT SALARIO_EMPLEADO.SSN, SUM(CASE WHEN TASK LIKE 'I am
on%' then (-SALARIO) ELSE(COSTS.WAGE-SALARIO) END) AS TOTAL

FROM SALARIO_EMPLEADO
INNER JOIN COSTS
ON SALARIO_EMPLEADO.JOB = COSTS.CAT_NUM
WHERE EXTRACT (YEAR FROM SALARIO_EMPLEADO.FECHA)=COSTS.YEAR
GROUP BY SALARIO_EMPLEADO.SSN);
```

- Resultado: 92 filas

c. Funcionalidad de la vista

No se pueden hacer operaciones de borrado, modificación o inserción ya que la vista referencia a más de una tabla.

```
--Se crea un nuevo proyecto (anteriormente se crea el
manager)--
insert into employees
values ('E_PRUEBA1');
insert into contracts values('E_PRUEBA1','04/04/2016',
'20/05/2016','6','null');

insert into projects
values('EX/PRUEBA', 'PROYECTO PRUEBA','10', '5',
'04/04/2016','03/04/2022','E_PRUEBA1');
--Se crea otro empleado --
insert into employees values ('E_PRUEBA2');
```

```
insert into contracts values('E_PRUEBA2','08/04/2016',
'20/05/2016','6','null');
--Se crea jefe de equipo--
insert into employees values ('E_PRUEBA3');
insert into contracts values('E_PRUEBA3','08/04/2016',
'20/05/2016','6','null');
--Se crea equipo
insert into teams values('EX/PRUEBA','equipo
alfa','E_PRUEBA3');
insert into memberships
values('EX/PRUEBA','equipo alfa',
'E_PRUEBA2','11/04/2016','24/04/2016',6,'50');

--Se crean varias entradas correspondientes al nuevo Proyecto
en REP_LINES--
insert into reports values('E_PRUEBA2',
'11/04/2016','E_PRUEBA1');
insert into rep_lines values('E_PRUEBA2','11/04/2016','2',
'11','EX/PRUEBA','equipo alfa','6','Analysis');
insert into rep_lines
values('E_PRUEBA2','11/04/2016','2', '12','EX/PRUEBA','equipo
alfa','6','Validation of module 49 at subsystem 08');
```

- Resultado: 93 filas

45	E_PRUEBA2	44.66
46	04/06517516/09T	20.569.53

b) Vista Nómina: listado de empleados en el mes corriente (sysdate), con su salario, número de trienios, y total a percibir (salario más 50€ por trienio).

a. Diseño en algebra relacional

$$\Pi_{\text{CONTRACTS.SSN, COSTS.SALARY, FLOOR(SYSDATE(YEAR)- CONTRACTS.START\_DATE(YEAR)/3), SALARY + FLOOR(SYSDATE(YEAR)-CONTRACTS.START\_DATE(YEAR)/3)*50}} \left(\sigma_{\text{COSTS.YEAR = SYSDATE(YEAR) AND (CONTRACTS.END\_DATE = NULL OR CONTRACTS.END\_DATE>REP\_LINES.REP\_DATE)}} \left((\text{CONTRACTS}) \theta_{\text{CONTRACTS.CAT\_NUM = COSTS.CAT\_NUM}} (\text{COSTS}) \right) \right)$$

b. Implementación en SQL

```
/*VISTA NOMINA*/
CREATE OR REPLACE VIEW V_NOMINA(SSN,SALARIO, NUM_TRIENIOS,
TOTAL) AS
SELECT CONTRACTS.SSN, COSTS.SALARY,
FLOOR(MONTHS_BETWEEN(SYSDATE, CONTRACTS.START_DATE)/36),
       SALARY + FLOOR(MONTHS_BETWEEN(SYSDATE,
CONTRACTS.START_DATE)/36)*50
FROM CONTRACTS
INNER JOIN COSTS
ON CONTRACTS.CAT_NUM = COSTS.CAT_NUM
WHERE COSTS.YEAR = EXTRACT (YEAR FROM SYSDATE) AND
(CONTRACTS.END_DATE>SYSDATE OR CONTRACTS.END_DATE IS
NULL);
```

- Resultado: 197 filas

d. Funcionalidad de la vista

No se pueden hacer operaciones de borrado, modificación o inserción ya que la vista referencia a más de una tabla.

```
--Se inserta un nuevo empleado que tiene un contrato vigente
en el mes corriente--

INSERT INTO EMPLOYEES VALUES('E_PRUEBA');
INSERT INTO CONTRACTS VALUES ('E_PRUEBA', '01-02-2004', NULL,
'7', NULL);
```

- Resultado: 198 filas

197	00/05689509/07T	2779	0	2779
198	E_PRUEBA	2215	4	2415

4. Disparadores

Categoría A): Disparador 'Categoría jefe proyecto'

Tabla(s) asociada(s)	PROJECTS
Evento(s) en los que se dispara	Inserción o modificación
Temporalidad	Antes de insertar (Before Each Row)
Granularidad	For Each Row
Condición	-

Se obtiene la categoría profesional del empleado que se inserta en 'MANAGER' consultando la tabla 'CONTRACTS', igualando el SSN del contrato con el SSN del manager insertado. En el caso de que el nuevo manager tiene una categoría de 0, 7, 8 o 9, entonces salta el error de categoría inválida; si el SSN insertado no se encuentre en los contratos (por no estar contratado o por tener una fecha de fin de contrato anterior a la inicialización del proyecto), salta el error "contrato invalido".

Acción:

```
CREATE OR REPLACE TRIGGER T_CATEGORIA_JEFE_PROYECTO
BEFORE
INSERT OR UPDATE ON PROJECTS
FOR EACH ROW
DECLARE CATEGORIA NUMBER;
BEGIN
     SELECT CAT_NUM INTO CATEGORIA
     FROM CONTRACTS
     WHERE :NEW.MANAGER = CONTRACTS.SSN AND
:NEW.END_DATE<=CONTRACTS.END_DATE AND CONTRACTS.START_DATE <=
:NEW.START_DATE ;

 if
     CATEGORIA IN (0,7,8,9)
     THEN RAISE_APPLICATION_ERROR (-20000, 'CATEGORIA JEFE DE
PROYECTO INVALIDA');

 END IF;
 exception
 when NO_DATA_FOUND then
 RAISE_APPLICATION_ERROR (-20000, 'CONTRATO INVÁLIDO');


 END;
```

Pruebas:

```
---Intentar insertar un jefe de proyecto con una categoría
inválida.---

update projects
set manager = '99/53659755/00T'  ----Empleado con cat_num=0---
where idproject='EX/56Q8086631S7';
```

- Resultado: RAISE_APPLICATION_ERROR (-20000, 'CATEGORIA JEFE DE PROYECTO INVALIDA');

Categoría B): Disparador 'Empleado sobrecargado'

Tabla(s) asociada(s)	REP_LINES
Evento(s) en los que se dispara	Inserción
Temporalidad	Antes de insertar (Before Each Row)
Granularidad	For Each Row
Condición	-

Antes de insertar una nueva fila en REP_LINES, se cuenta las horas que ha hecho el empleado en esa semana. Si las horas totales ya suman 40, salta el disparador. Si no se encuentran los datos (por estar mal el SSN o la fecha de REP_LINES), salta el error de 'Datos inválidos'

Acción:

```
CREATE OR REPLACE TRIGGER T_EMPLEADO_SOBRECARGADO
BEFORE
INSERT ON REP_LINES
FOR EACH ROW
DECLARE HORAS_ACTUALES NUMBER DEFAULT 0;
BEGIN
SELECT COUNT (HOUR) INTO HORAS_ACTUALES
FROM REP_LINES
WHERE SSN =:NEW.SSN AND REP_DATE = :NEW.REP_DATE;
IF (HORAS_ACTUALES)=40
THEN RAISE_APPLICATION_ERROR (-20000, "EMPLEADO SOBRECARGADO");
END IF;
EXCEPTION
WHEN NO_DATA_FOUND THEN
RAISE_APPLICATION_ERROR (-20000, 'DATOS INVÁLIDOS');
END;
```

Prueba:

```
---Insertar una nueva fila REP_LINES para un empleado que ya
tiene 40 h esa semana---

insert into rep_lines values ('77/11752769/08T', '11/01/16', 1,
'7', 'IN/Y4981M874187', null, 6, 'Validation');
```

- Resultado: RAISE_APPLICATION_ERROR (-20000, "EMPLEADO SOBRECARGADO")

Categoría C): Disparador 'Fechas en contratos'

Tabla(s) asociada(s)	CONTRACTS
Evento(s) en los que se dispara	Inserción
Temporalidad	Antes de insertar (Before Each Row)
Granularidad	For Each Row
Condición	-

Se compara la fecha inicio del contrato insertado con la fecha fin del contrato anterior más reciente, saltando el disparador si las fechas se solapan.

Acción:

```
CREATE OR REPLACE TRIGGER T_FECHAS_CONTRATOS
BEFORE
INSERT ON CONTRACTS
FOR EACH ROW
DECLARE FECHA_FIN DATE DEFAULT TO_DATE('01-01-1900', 'DD-MM-
YYYY');

BEGIN
select max(nvl(end_date,to_date('31-12-9999', 'dd-mm-yyyy')))
into fecha_fin
from contracts
WHERE SSN=:NEW.SSN;

IF FECHA_FIN > :NEW.START_DATE
THEN RAISE_APPLICATION_ERROR (-20000, 'FECHA CONTRATO SE
SOLAPA');
END IF;
END;
```

Pruebas:

```
---Se inserta un nuevo contrato para un empleado que tiene un
contrato indefinido---

insert into contracts values ('65/93099365/09T',
to_date('01/01/2017', 'dd-mm-yyyy'), null, 7, null);
```

- Resultado: RAISE_APPLICATION_ERROR (-20000, 'FECHA CONTRATO SE SOLAPA')

Categoría D): Disparador 'Reemplazo Jefe Equipo'

Tabla(s) asociada(s)	TEAMS
Evento(s) en los que se dispara	Modificación
Temporalidad	Antes de modificar (Before Each Row)
Granularidad	For Each Row
Condición	-

El enunciado especifica que un jefe de equipo sólo puede ser sustituido por un jefe de proyecto. Este disparador comprueba que el nuevo jefe de equipo cumple esta condición.

Acción:

```
CREATE OR REPLACE TRIGGER T_REEMPLAZO_JEFE_EQUIPO
BEFORE
UPDATE OF LEADER ON TEAMS
FOR EACH ROW
DECLARE JEFE_PROYECTO VARCHAR(15);
BEGIN
SELECT MANAGER INTO JEFE_PROYECTO
FROM PROJECTS
WHERE MANAGER = :NEW.LEADER AND END_DATE>SYSDATE;
EXCEPTION
WHEN NO_DATA_FOUND THEN RAISE_APPLICATION_ERROR (-20000, 'JEFE
EQUIPO SOLO PUEDE SER SUSTITUIDO POR UN JEFE PROYECTO');
END;
```

Pruebas:

```
--- Introducir un jefe de equipo que no es jefe de proyecto---

select idproject
from projects
where manager = '87/95210788/05T'
--No se selecciona ninguna fila--

update teams
set leader = '87/95210788/05T'
where idproject = 'EX/3UZ0BRM34S20';
```

- Resultado: RAISE_APPLICATION_ERROR (-20000, 'JEFE EQUIPO SOLO PUEDE SER SUSTITUIDO POR UN JEFE PROYECTO')

5. Completitud Semántica (Práctica Opcional)

1. **Vista Leaders Top Five: los cinco jefes de proyecto cuyos proyectos dirigidos más beneficio han aportado a la empresa.**

   a. *Diseño en algebra relacional*

$$\Pi_{\text{REP\_LINES.SSN, REP\_LINES.JOB, REP\_LINES.REP\_DATE AS FECHA, COSTS.SALARY/160 AS SALARIO, REP\_LINES.TASK}} ($$
$$\sigma_{\text{COSTS.YEAR = REP\_LINES.REP\_DATE(YEAR) AND CONTRACTS.START\_DATE<REP\_LINES.REP\_DATE AND (CONTRACTS.END\_DATE = NULL OR CONTRACTS.END\_DATE>REP\_LINES.REP\_DATE)}} ($$
$$(\text{REP\_LINES}) \theta_{\text{REP\_LINES.SSN = CONTRACTS.SSN}}$$
$$(\text{CONTRACTS}) \theta_{\text{CONTRACTS.CAT\_NUM = COSTS.CAT\_NUM}}$$
$$(\text{COSTS}))) \text{ AS SALARIO\_EMPLEADO}$$

$$\Pi_{\text{MANAGER, TOTAL}} ($$
$$\sigma_{\text{ROWNUM<=5}} ($$
$$\text{ORDER BY}_{\text{SUM(COSTS.WAGE - SALARY\_EMPLOYEE.SALARY/160) DESC}} ($$
$$\text{GROUP BY}_{\text{PROJECTS.IDPROJECT}} ($$
$$\Pi_{\text{PROJECTS.IDPROJECT, SUM(IF TASK = 'I am on a vacation' then (-SALARY\_EMPLOYEE.SALARY/160) ELSE(COSTS.WAGE-SALARY\_EMPLOYEE.SALARY/160)) AS TOTAL, MAX(PROJECTS.MANAGER) AS MANAGER}} ($$
$$\sigma_{\text{COSTS.YEAR = SALARY\_EMPLOYEE.YEAR}} ($$
$$(\text{SALARY EMPLOYEE}) \theta_{\text{SALARY\_EMPLOYEE.JOB = COSTS.CAT\_NUM}}$$
$$(\text{COSTS}) \theta_{\text{SALARY\_EMPLOYEE.IDPROJECT = PROJECTS.IDPROJECT}}$$
$$(\text{PROJECTS})$$
$$))))))$$

b. Implementación en SQL

```
CREATE OR REPLACE VIEW V_LEADERS_TOP5(MANAGER,BENEFICIO) AS

SELECT MANAGER, TO_CHAR(TOTAL, '9,999,999.99')
FROM(
WITH SALARY_EMPLOYEE AS(
SELECT REP_LINES.SSN, COSTS.SALARY, REP_LINES.IDPROJECT,
REP_LINES.JOB, EXTRACT (YEAR FROM REP_LINES.REP_DATE) AS YEAR,
REP_LINES.TASK
FROM REP_LINES
INNER JOIN CONTRACTS
ON REP_LINES.SSN = CONTRACTS.SSN
INNER JOIN COSTS
ON CONTRACTS.CAT_NUM = COSTS.CAT_NUM
WHERE EXTRACT (YEAR FROM REP_LINES.REP_DATE)=COSTS.YEAR AND
CONTRACTS.START_DATE<REP_LINES.REP_DATE AND
(CONTRACTS.END_DATE IS NULL OR
CONTRACTS.END_DATE>REP_LINES.REP_DATE)
)

SELECT PROJECTS.IDPROJECT, SUM(CASE WHEN TASK LIKE 'I am on%'
then (-SALARY_EMPLOYEE.SALARY/160) ELSE(COSTS.WAGE-
SALARY_EMPLOYEE.SALARY/160) END) AS TOTAL,
                              MAX(PROJECTS.MANAGER) AS MANAGER
FROM SALARY_EMPLOYEE
INNER JOIN COSTS
ON SALARY_EMPLOYEE.JOB = COSTS.CAT_NUM
INNER JOIN PROJECTS
ON SALARY_EMPLOYEE.IDPROJECT = PROJECTS.IDPROJECT
WHERE COSTS.YEAR = SALARY_EMPLOYEE.YEAR
GROUP BY PROJECTS.IDPROJECT
ORDER BY SUM(COSTS.WAGE - SALARY_EMPLOYEE.SALARY/160) DESC )
WHERE ROWNUM<=5;
```

c. Funcionalidad de la vista

No se pueden hacer operaciones de borrado, modificación o inserción ya que la vista referencia a más de una tabla.

2. Vista Employee of the Month: jefe de proyecto o empleado que más beneficio aporta a la empresa en cada mes y año, ordenado cronológicamente.

a. Diseño en algebra relacional

Π REP_LINES.SSN, REP_LINES.JOB, REP_LINES.REP_DATE AS FECHA, COSTS.SALARY/160 AS SALARIO(

σ COSTS.YEAR = REP_LINES.REP_DATE(YEAR) AND CONTRACTS.START_DATE<REP_LINES.REP_DATE AND (CONTRACTS.END_DATE = NULL OR CONTRACTS.END_DATE>REP_LINES.REP_DATE) and TASK NOT LIKE ('I am on%') (

(REP_LINES) REP_LINES.SSN = CONTRACTS.SSN θ

(CONTRACTS) CONTRACTS.CAT_NUM = COSTS.CAT_NUMθ

(COSTS))) AS SALARY_EMPLOYEE

GROUP BY SSN, MONTH, YEAR(

Π SALARY_EMPLOYEE.SSN, SALARY_EMPLOYEE.REP_DATE(MONTH) AS MONTH, SALARY_EMPLOYEE.REP_DATE(YEAR) AS YEAR, SUM(COSTS.WAGE-SALARY_EMPLOYEE.SALARY) AS TOTAL (

σ COSTS.YEAR = REP_LINES.REP_DATE(YEAR) (

(SALARY_EMPLOYEE) SALARY_EMPLOYEE.JOB = COSTS.CAT_NUMθ

(COSTS)))) AS SSN_EMPLEADO

GROUP BY MONTH, YEAR

Π MONTH, YEAR, MAX(TOTAL) AS TOTAL (

σ (SSN_EMPLOYEE)

) AS BALANCE

ORDER BY YEAR, MONTH(

Π DISTINCT SSN_EMPLOYEE.SSN, BALANCE.MONTH, BALANCE.YEAR, BALANCE.TOTAL(

σ BALANCE.MONTH = SSN_EMPLOYEE.MONTH AND BALANCE.YEAR = SSN_EMPLOYEE.YEAR(

(BALANCE) BALANCE.TOTAL = SSN_EMPLOYEE.TOTAL θ

(SSN_EMPLOYEE)

)))

b. Implementación en SQL

```
CREATE OR REPLACE VIEW V_EMPLOYEE_OF_THE_MONTH(EMPLOYEE, MONTH,
YEAR, BALANCE) AS

WITH SALARY_EMPLOYEE AS(
SELECT REP_LINES.SSN, REP_LINES.REP_DATE, COSTS.SALARY/160 AS
SALARY, REP_LINES.JOB
FROM REP_LINES
INNER JOIN CONTRACTS
ON REP_LINES.SSN = CONTRACTS.SSN
INNER JOIN COSTS
ON CONTRACTS.CAT_NUM = COSTS.CAT_NUM
WHERE EXTRACT (YEAR FROM REP_LINES.REP_DATE)=COSTS.YEAR AND
       CONTRACTS.START_DATE<=REP_LINES.REP_DATE AND
(CONTRACTS.END_DATE IS NULL OR
CONTRACTS.END_DATE>=REP_LINES.REP_DATE) AND TASK NOT LIKE ('I am
on%')

),

SSN_EMPLOYEE AS(
SELECT SALARY_EMPLOYEE.SSN, EXTRACT (MONTH FROM
SALARY_EMPLOYEE.REP_DATE) AS MONTH,EXTRACT (YEAR FROM
SALARY_EMPLOYEE.REP_DATE) AS YEAR,
       SUM(COSTS.WAGE-SALARY_EMPLOYEE.SALARY) AS TOTAL
FROM SALARY_EMPLOYEE
INNER JOIN COSTS
ON SALARY_EMPLOYEE.JOB = COSTS.CAT_NUM
WHERE EXTRACT(YEAR FROM SALARY_EMPLOYEE.REP_DATE)=COSTS.YEAR
GROUP BY SALARY_EMPLOYEE.SSN, EXTRACT (MONTH FROM
SALARY_EMPLOYEE.REP_DATE), EXTRACT (YEAR FROM
SALARY_EMPLOYEE.REP_DATE)
),
BALANCE AS(
SELECT MONTH, YEAR, MAX(TOTAL) AS TOTAL
FROM SSN_EMPLOYEE
GROUP BY MONTH, YEAR
)
SELECT DISTINCT SSN_EMPLOYEE.SSN, BALANCE.MONTH, BALANCE.YEAR,
TO_CHAR(BALANCE.TOTAL, '9,999,999.99')
FROM BALANCE
INNER JOIN SSN_EMPLOYEE
ON BALANCE.TOTAL = SSN_EMPLOYEE.TOTAL
```

```
WHERE BALANCE.MONTH = SSN_EMPLOYEE.MONTH AND BALANCE.YEAR =
SSN_EMPLOYEE.YEAR
ORDER BY YEAR, MONTH;
```

- Resultado: 391 filas

c. *Funcionalidad de la vista*
No se pueden hacer operaciones de borrado, modificación o inserción ya que la vista referencia a más de una tabla.

```
--Se crea un nuevo proyecto para insertar al final un
nuevo empleado en abril de 2016(anteriormente se crea el
manager)--
insert into employees
values ('E_PRUEBA1');
insert into contracts values('E_PRUEBA1','04/04/2016',
'20/05/2016','6','null');

insert into projects
values('EX/PRUEBA', 'PROYECTO PRUEBA','10', '5',
'04/04/2016','03/04/2022','E_PRUEBA1');
--Se crea otro empleado --
insert into employees values ('E_PRUEBA2');
insert into contracts values('E_PRUEBA2','08/04/2016',
'20/05/2016','6','null');
--Se crea jefe de equipo--
insert into employees values ('E_PRUEBA3');
insert into contracts values('E_PRUEBA3','08/04/2016',
'20/05/2016','6','null');
--Se crea equipo
insert into teams values('EX/PRUEBA','equipo
alfa','E_PRUEBA3');
insert into memberships
values('EX/PRUEBA','equipo alfa',
'E_PRUEBA2','11/04/2016','24/04/2016',6,'50');

--Se crean varias entradas correspondientes al nuevo
Proyecto en REP_LINES--
insert into reports values('E_PRUEBA2',
'11/04/2016','1234');
insert into rep_lines
values('E_PRUEBA2','11/04/2016','2',
'11','EX/PRUEBA','equipo alfa','6','Analysis');
insert into rep_lines
```

```
values('E_PRUEBA2','11/04/2016','2',
'12','EX/PRUEBA','equipo alfa','6','Validation of module
49 at subsystem 08');
```

- Resultado: 392 filas

391	65/93099365/09T	1	2016	1,953.84
392	E_PRUEBA2	4	2016	44.66

3. Vista Edad empleados

a. Diseño en algebra relacional

$$\Pi_{\text{SSN, FLOOR(TRUNC(MONTHS\_BETWEEN(SYSDATE, BIRTHDATE))/12) AS EDAD}} (\sigma\,(\text{PROFILES}))$$

b. Implementación en SQL

```
CREATE OR REPLACE VIEW V_EDAD_EMPLEADOS (SSN, EDAD) AS
SELECT SSN, FLOOR(TRUNC(MONTHS_BETWEEN(SYSDATE, BIRTHDATE))/12)
AS EDAD
FROM PROFILES;
```

- Resultado: 999 filas

c. Funcionalidades de la vista

No se pueden hacer operaciones de modificación o inserción ya que la edad es un atributo derivado. Aunque en teoría se podría modificar el SSN, no tendría sentido hacerlo para esta vista en concreto, que solo recoge las edades de todos los empleados.

```
---Se inserta un nuevo empleado para comprobar que se actualiza
la vista---

insert into employees values('E_PRUEBA');
insert into profiles values ('E_PRUEBA5','123', 'Ana', 'Martin',
'Lopez', '10-10-1985', '1235', 'ana@gmail.com', 'calle mayor');
```

- Resultado: 1000 filas

999	73/94234864/00T	27
1000	E_PRUEBA5	30

4. Disparador Categoría jefe equipo

Tabla(s) asociada(s)	TEAMS
Evento(s) en los que se dispara	Inserción
Temporalidad	Antes de insertar (Before Each Row)
Granularidad	For Each Row
Condición	-

Se obtiene la categoría profesional del empleado consultando la tabla 'CONTRACTS', igualando el SSN del contrato con el SSN del leader insertado. En el caso de que el leader nuevo tiene una categoría de 0, 1, 2, 3, 4 o 5, entonces salta el error de categoría inválida; si el SSN insertado no se encuentre en los contratos (por no estar contratado o por tener una fecha de fin de contrato anterior a la inicialización del proyecto), salta el error de "Contrato inválido".

Acción:

```
CREATE OR REPLACE TRIGGER T_CATEGORIA_JEFE_EQUIPO
BEFORE
INSERT ON TEAMS
FOR EACH ROW
DECLARE CATEGORIA_INVALIDA EXCEPTION;
CATEGORIA NUMBER;
BEGIN
     SELECT CAT_NUM INTO CATEGORIA
     FROM CONTRACTS
      WHERE :NEW.LEADER = CONTRACTS.SSN AND
CONTRACTS.END_DATE>SYSDATE;

 if
     CATEGORIA IN (0,1,2,3,4,5)
     then RAISE_APPLICATION_ERROR(-20000, 'CATEGORIA JEFE DE EQUIPO
INVALIDA');

 END IF;
 exception
 when NO_DATA_FOUND then
 RAISE_APPLICATION_ERROR (-20000, 'CONTRATO INVÁLIDO');
 END;
```

Pruebas:

```
update teams
set leader = '99/53659755/00T' ---cat_num = 0---
where idproject='EX/3UZ0BRM34S20';
```

- Resultado: `RAISE_APPLICATION_ERROR(-20000, 'CATEGORIA JEFE DE EQUIPO INVALIDA')`

5. Disparador Categoría empleado

Tabla(s) asociada(s)	MEMBERSHIPS
Evento(s) en los que se dispara	Inserción y modificación
Temporalidad	Antes de insertar (Before Each Row)
Granularidad	For Each Row
Condición	-

Se obtiene la categoría profesional del empleado consultando la tabla 'CONTRACTS', igualando el SSN del contrato con el SSN del employee insertado. En el caso de que la categoría del puesto es mayor que (1+"categoría por la que está contratado") o menor que ("categoría por la que está contratado -1), entonces salta el error de categoría invalida; si el SSN insertado no se encuentre en los contratos (por no estar contratado o por tener una fecha de fin de contrato anterior a la inicialización del proyecto), salta el error "contrato invalido".

Acción:

```
CREATE OR REPLACE TRIGGER T_CATEGORIA_EMPLEADO
BEFORE
INSERT OR UPDATE OF EMPLOYEE ON MEMBERSHIPS
FOR EACH ROW
DECLARE CATEGORIA NUMBER;
BEGIN

      SELECT CAT_NUM INTO CATEGORIA
      FROM CONTRACTS
      WHERE :NEW.EMPLOYEE = CONTRACTS.SSN;

 if
     CATEGORIA != :NEW.CAT_PROJECT OR CATEGORIA !=
:NEW.CAT_PROJECT+1 OR CATEGORIA != :NEW.CAT_PROJECT-1
     then RAISE_APPLICATION_ERROR(-20000, 'CATEGORIA EMPLEADO
INVALIDA');

 END IF;
 exception
 when NO_DATA_FOUND then
 RAISE_APPLICATION_ERROR (-20000, 'CONTRATO INVÁLIDO');
 END;
```

Pruebas:

```
select cat_project
from memberships
where idproject ='EX/3UZ0BRM34S20';
---la categoría necesaria para este Proyecto es de 8 o 9---

update MEMBERSHIPS
set EMPLOYEE = '99/53659755/00T'
where idproject='EX/3UZ0BRM34S20';
---la categoria de este empleado = 0---
```

- Resultado: RAISE_APPLICATION_ERROR(-20000, 'CATEGORIA EMPLEADO INVALIDA')

6. Disparador jefe proyecto sobrecargado

Tabla(s) asociada(s)	PROJECTS
Evento(s) en los que se dispara	Inserción
Temporalidad	Antes de insertar (Before Each Row)
Granularidad	For Each Row
Condición	-

El disparador jefe Proyecto sobrecargado está asociado a la tabla 'PROJECTS'. Se dispara ante el evento de insertar una nueva fila en la tabla 'PROJECTS' (pero no para la modificación, ya que no se puede despedir o cambiar un jefe proyecto según el enunciado).
Se suman el número de proyectos asignados a ese jefe de proyecto, si los proyectos asignados suman 3 o más, salta error de jefe de proyecto sobrecargado.

Acción:

```
CREATE OR REPLACE TRIGGER T_JEFE_PROYECTO_SOBRECARGADO
BEFORE
INSERT ON PROJECTS
FOR EACH ROW
DECLARE PROYECTOS_ACTUALES NUMBER;
BEGIN
SELECT COUNT(IDPROJECT)INTO PROYECTOS_ACTUALES
FROM PROJECTS
WHERE MANAGER=:NEW.MANAGER AND END_DATE>SYSDATE;
IF (PROYECTOS_ACTUALES)>=3
THEN RAISE_APPLICATION_ERROR (-20000, 'JEFE DE PROYECTO
SOBRECARGADO');
END IF;
END;
```

Pruebas:

```
insert into projects
valueS ('EX/PRUEBA', 'PROYECTO PRUEBA,'0', '5', '04/04/2016',
'03/04/2022', '39/31612749/00T');
```

- Resultado: `RAISE_APPLICATION_ERROR (-20000, 'JEFE DE PROYECTO SOBRECARGADO')`

7. Disparador Fechas contiguas

Tabla(s) asociada(s)	CONTRACTS
Evento(s) en los que se dispara	Inserción
Temporalidad	Antes de insertar (Before Each Row)
Granularidad	For Each Row
Condición	-

Se compara la fecha inicio del contrato insertado con la fecha fin del contrato anterior más reciente. En el caso de que se solapen las fechas, se modifica el contrato anterior para que termine un dia antes del inicio del contrato nuevo, poniendo además como motivo de cese 'Contract termination due to new contract'.

Acción:

```
CREATE OR REPLACE TRIGGER T_FECHAS_CONTRATOS
BEFORE
INSERT ON CONTRACTS
FOR EACH ROW
DECLARE FECHA_FIN DATE DEFAULT TO_DATE('01-01-1900', 'DD-MM-YYYY') ;

BEGIN
select max(nvl(end_date,'31-12-9999')) into fecha_fin
from contracts
WHERE SSN=:NEW.SSN;

IF FECHA_FIN >= :NEW.START_DATE
THEN  DBMS_OUTPUT.PUT_LINE('FECHA CONTRATO SE SOLAPA');
UPDATE CONTRACTS
SET END_DATE = :NEW.START_DATE-1, CEASING = 'Contract termination due
to new contract'
WHERE SSN=:NEW.SSN AND FECHA_FIN = nvl(end_date,'31-12-9999')  ;
END IF;
END;
```

Pruebas:

```
insert into contracts values ('70/66789976/02T', to_date('02/03/2019',
'dd-mm-yyyy'), null, 7, null);

--Se inserta un Nuevo contrato para un empleado con contrato
indefinido---
```

Resultado:

	SSN	START_...	END_DATE	CAT_NUM	CEASING
1	70/66789976/02T	02/03/19	(null)	7	(null)
2	70/66789976/02T	01/03/14	01/03/19	9	Contract termination due to new contract

6. Conclusiones

En esta práctica se ha implementado algunos de los supuestos semánticos no contemplados en la primera práctica. En concreto, se implementaron:

- La edad de los empleados
- Comprobación de la categoría del jefe de proyecto
- Comprobación de la categoría del jefe de equipo
- Comprobación de la categoría del empleado
- Comprobación de que un jefe de proyecto solo puede liderar 3 proyectos
- Comprobación de que un jefe de equipo solo puede ser sustituido por un jefe de proyecto

Además de estas consultas se implementaron otras que no se especificaron en el enunciado de la primera práctica se podían obtener a partir de los datos proporcionados, como el gasto por proyecto o beneficio por empleado. Se ha intentado ser lo más riguroso posible a la hora de realizar las consultas para seleccionar los datos más apropiados, atendiendo siempre a lo especificado en el enunciado. Cabe señalar que la base de datos proporcionada no siempre era consistente. Por ejemplo, en había empleados realizando tareas antes de la fecha de inicio de su contrato. Estos casos se han filtrado a la hora de hacer la implementación.

En cuanto al esfuerzo dedicado a la realización de las prácticas, cabe señalar que la segunda práctica nos ha llevado considerablemente más tiempo que la primera, ya que para nosotros ha resultado más complejo realizar correctamente las consultas que el volcado de datos. Así mismo era más difícil comprobar que se ha realizado correctamente las tareas de la segunda práctica. Aunque las dos prácticas han sido muy laboriosas, gracias a su realización hemos aprendido mucho del lenguaje SQL.

Finalmente, respecto a posibles mejoras para otros años, sería mejor si el enunciado fuese más preciso. Muchos de los problemas que hemos tenido a la hora de implementar las consultas o de diseñar el modelo relacional han sido debido a que no entendíamos claramente cómo se organizaban los datos, lo que nos ha llevado a dedicarle horas extra a la práctica.