

Titulación: GRADO INGENIERIA INFORMATICA
Año Académico: 2013/2014
Curso: 2º
Asignatura: Ficheros y Bases de Datos
Título: Memoria Práctica 2 – Manipulación y Relacional Avanzado



UNIVERSIDAD CARLOS III DE MADRID

Profesor:	Francisco Javier Calle Gómez		
Grupo Peq:	USER0341	Grupo	82
Alumno/a:	Carlos Contreras Sanz	NIA:	100303562
Alumno/a:	Álvaro Gómez Ramos	NIA:	100307009

1. Introducción

En este documento se van a explicar cada una de las tareas que se nos han pedido, además de las que hemos añadido sin que se nos pidan para hacer el modelo más próximo a lo que nos pide el cliente.

El documento se separará en consultas, vistas, disparadores y finalmente se añadirá una conclusión.

2. Consultas

Consulta 1: Vencedor de la Copa del Rey de Fútbol en 2013 (Salida: Nombre club).

En esta consulta lo que hemos hecho, ha sido simplemente intentar seleccionar de la tabla Club, aquellas tuplas que cumpliesen una serie de requisito.

En álgebra relacional sería:

$\pi_{\text{Nom_Club}} (\sigma_{\text{Deporte}='fútbol' \text{ AND } \text{Nom_Competicion}='Copa del Rey' \text{ AND } \text{Fecha_gana}=2013} (\text{Ganador}))$

```
SELECT Nom_Club
FROM Ganador
WHERE (Deporte='fútbol' AND Nom_Competicion='Copa del Rey' AND
Fecha_gana=2013);
```

Que se implementa en SQL como:

```
SQL>
SQL> SELECT Nom_Club FROM Ganador WHERE (Deporte='fútbol' AND Nom_Competicion='Copa del Rey' AND Fecha_gana=2013);
ninguna fila seleccionada
SQL> |
```

Al ir a ejecutar la consulta no nos devuelve ningún resultado.

Para probar la consulta entonces: seleccionamos una combinación de valores para Deporte, Nom_Competicion y Fecha_gana que sepamos que existen. Para ello imprimimos los que hay, y seleccionamos una cualquiera.

Comprobamos la consulta con esos valores:

```
SQL> SELECT Nom_Club FROM Ganador WHERE (Deporte='Surf' AND Nom_Competicion='Alta Competición benjamín de Surf' AND Fecha_gana=2012);
NOM CLUB
-----
Real Valchimeneas de la Alameda Club de Surf S.A.D.
SQL>
```

Luego se deduce que, en caso de existir tal competición, en tal año y de tal deporte, sí que nos devolverá el ganador, pero si no existe esa combinación de valores, no nos devolverá nada.

Consulta 2: Ciclistas que nunca se doparon (Salida: Nombre, apellido, apodo).

En este caso hemos buscado obtener todos los ciclistas, y descontar de esos los que se han dopado. Una vez obtenidos, sacar los nombres, apellidos y apodos de cada uno.

En álgebra relacional sería:

Π Nombre, Apellido1, Apodo $([\pi_{\text{Deportista} \equiv \text{Sujeto_Control}} (\sigma_{\text{Deporte}='Ciclismo'} (\text{Federado}))]$
 $- [\pi_{\text{Sujeto_Control}} (\sigma_{\text{Sustancia IS NOT NULL} (\text{Evidencia})) \theta_{\text{Sujeto_Control}=\text{CID}} (\text{Deportista}))])$

Que se implementa en SQL como:

```
SELECT Nombre, Apellido1, Apodo FROM
  (((SELECT Deportista AS Sujeto_Control FROM Federado WHERE
    Deporte='Ciclismo')
  MINUS
  (SELECT Sujeto_Control FROM Evidencia WHERE Sustancia IS
    NOT NULL)) JOIN Deportista ON Sujeto_Control=CID);
```

La ejecutamos en SQL:

```
SQL> select count(*) from(SELECT Nombre, Apellido1, Apodo FROM
  2  (((SELECT Deportista AS Sujeto_Control FROM Federado WHERE Deporte='Ciclismo')
  3  MINUS
  4  (SELECT Sujeto_Control FROM Evidencia WHERE Sustancia IS NOT NULL)) JOIN Deportista ON Sujeto_
  Control=CID));
  COUNT(*)
  -----
      2352
```

Esta consulta nos dice que hay 2352 ciclistas que no se han dopado. Para comprobar que efectivamente cuenta bien, vamos a introducir uno sin evidencia, comprobaremos, y luego le añadiremos evidencia, y volveremos a ejecutar la consulta.

```
INSERT INTO Deportista VALUES ('AAAAAAAAAAAA', 'paco', 'de lucia',
  'y almendralejo', 'el cigala', '3333-33-33');
INSERT INTO Federado VALUES ('AAAAAAAAAAAA', 'Ciclismo', 'Gambia',
  '11111', 'profesional');
INSERT INTO Control VALUES ('AAAAAAAAAAAA', '3333-33-3322:22',
  'Sotovacac');
```

Realizamos ahora la consulta de nuevo, y nos devuelve uno más. Todo correcto de momento.

```
SQL> INSERT INTO Federado VALUES ('AAAAAAAAAAAA', 'Ciclismo', 'Gambia', '11111', 'profesional');
1 fila creada.

SQL> select count(*) from (SELECT Nombre, Apellido1, Apodo FROM
2   (((SELECT Deportista AS Sujeto_Control FROM Federado WHERE Deporte='Ciclismo')
3  MINUS
4   (SELECT Sujeto_Control FROM Evidencia WHERE Sustancia IS NOT NULL)) JOIN Deportista ON Sujeto_
Control=CID));

COUNT(*)
-----
2353
```

Insertamos ahora una evidencia sin sustancia, y comprobamos que devuelve el mismo número. Tras eso, comprobamos que sale lo mismo e insertamos sustancia:

```
INSERT INTO Evidencia VALUES('2', '2222-22-22', 'AAAAAAAAAAAAA',
'3333-33-3322:22', 'mosto', 'dopaje leve');

INSERT INTO Evidencia VALUES('1', '2222-22-22', 'AAAAAAAAAAAAA',
'3333-33-3322:22', NULL, 'dopaje leve');
```

Y por último, vemos si nos devuelve uno menos (ya que habría una evidencia para ese deportista, y no se contaría)

```
SQL> select count(*) from (SELECT Nombre, Apellido1, Apodo FROM
2   (((SELECT Deportista AS Sujeto_Control FROM Federado WHERE Deporte='Ciclismo')
3  MINUS
4   (SELECT Sujeto_Control FROM Evidencia WHERE Sustancia IS NOT NULL)) JOIN Deportista ON Sujeto_
Control=CID));

COUNT(*)
-----
2352
```

Efectivamente nos devuelve el número original. Todo correcto.

Consulta 3: Deportistas que se han ‘saltado’ (no han pasado) por ciertas categorías (están en una superior sin haber pasado por las anteriores).

En esta consulta hemos averiguado los números que corresponden a cada categoría, y nos hemos quedado para cada deportista con su CID, deporte y número de categoría. Hemos almacenado todos los casos en una tabla, y hemos obtenido solo los que cumpliesen que el número de veces que aparecían en la tablas era fuese menor que la categoría mayor, lo que indicaría que ha pasado por todas.

Se tienen en cuenta tanto los de histórico como los de ficha actual vigente.



Consulta en álgebra relacional:

$\Pi_{\text{Deportista, Deporte}} \sigma_{\text{MAX(Numero) > COUNT(Numero)}} \text{ GROUP BY } \text{Deportista, Deporte}$
 $[\Pi_{\text{Deportista, Numero, Deporte}} ((\Pi_{\text{Deportista, Categoría}} \equiv \text{Nomb, Deporte Ficha}) \theta_{\text{(Categoría.Nombre=Nomb Categoría)})$
 $\cup \Pi_{\text{Deportista, Numero, Deporte}} (\text{Histórico } \theta_{\text{Categoría.Nombre=Historico_Ficha.Categoría Categoría)})]$

Que se implementa en SQL como:

```

select Deportista, Deporte from
  ((SELECT Deportista, Numero, Deporte FROM ((SELECT
    Deportista, Categoría AS Nomb, Deporte FROM Ficha) JOIN
    Categoría ON Categoría.Nombre=Nomb))
  UNION
  (SELECT Deportista, Numero, Deporte FROM Historico_Ficha
    JOIN
    Categoría
    ON
    Categoría.Nombre=Historico_Ficha.Categoría))
  GROUP BY Deportista, Deporte HAVING MAX(Numero)>COUNT(Numero);
  
```

```

SQL> select count(*) from(
  2 select Deportista, Deporte from
  3 ((SELECT Deportista, Numero, Deporte FROM ((SELECT Deportista, Categoría AS Nomb, Deporte FROM
  4 Ficha) JOIN Categoría ON Categoría.Nombre=Nomb))
  5 UNION
  6 (SELECT Deportista, Numero, Deporte AS Cont FROM Historico_Ficha JOIN Categoría ON Categoría.N
  7 ombre=Historico_Ficha.Categoría))
  8 GROUP BY Deportista, Deporte HAVING MAX(Numero)>COUNT(Numero)
  9 );
  
```

COUNT(*)
51832

Comprobaremos ahora para un caso concreto las distintas posibilidades y las variaciones que estas podrían tener sobre el conteo anterior (y ver si incluiría más o menos deportistas)

```
SQL> select * from ficha where Deportista='Q0I7772Q8125' AND Categoría='benjamín';
```

NOMBRE CLUB

PAIS

NOM_SECCION

CATEGORIA

DEPORTE

DEPORTISTA FECHA_INI

Real Club de Pedragales de las Aguas S.A.D.

Kyrgyzstan

Sección de Fútbol benjamín

benjamín

Fútbol

Q0I7772Q8125 2005-12-09

Tenemos que para ese CID, esa persona está en la categoría de benjamín para el fútbol.

Realizamos una inserción de la misma tupla con la categoría cambiada a alevín, que es la siguiente (para ello hacemos uso del disparador obligatorio) y además se copia en histórico_ficha esa tupla (lo hace el disparador obligatorio, la de categoría alevín). Esto no debería cambiar el número de casos detectados por la consulta:

```

SQL> UPDATE Ficha
  2 SET Categoria='alevín', Fecha_Ini=SYSDATE
  3 WHERE Deportista='Q0I7772Q8125' AND Categoria='benjamín' AND Deporte='Fútbol';

1 fila actualizada.

SQL> select count(*) from(
  2 select Deportista, Deporte from
  3 ((SELECT Deportista, Numero, Deporte FROM ((SELECT Deportista, Categoria AS Nomb, Deporte FROM
Ficha) JOIN Categoria ON Categoria.Nombre=Nomb))
  4 UNION
  5 (SELECT Deportista, Numero, Deporte AS Cont FROM Historico_Ficha JOIN Categoria ON Categoria.N
omb=Historico_Ficha.Categoria))
  6 GROUP BY Deportista, Deporte HAVING MAX(Numero)>COUNT(Numero)
  7 );

COUNT(*)
-----
51832
  
```

Ahora, borramos de histórico_ficha el caso de ese jugador en la categoría de benjamín de futbol, con lo que quedaría solo un registro de ese jugador en futbol, en la categoría de alevín.

Esto sí debería incrementar en uno el número de casos detectados por la consulta:

```

COUNT(*)
-----
51832

SQL> delete from Historico_Ficha where Deportista='Q0I7772Q8125';

1 fila suprimida.

SQL> select count(*) from(
  2 select Deportista, Deporte from
  3 ((SELECT Deportista, Numero, Deporte FROM ((SELECT Deportista, Categoria AS Nomb, Deporte FROM
Ficha) JOIN Categoria ON Categoria.Nombre=Nomb))
  4 UNION
  5 (SELECT Deportista, Numero, Deporte AS Cont FROM Historico_Ficha JOIN Categoria ON Categoria.N
omb=Historico_Ficha.Categoria))
  6 GROUP BY Deportista, Deporte HAVING MAX(Numero)>COUNT(Numero)
  7 );

COUNT(*)
-----
51833
  
```

Efectivamente, cuenta uno de más. En esas situaciones se ve reflejado el caso de que haya alguien en ficha o en histórico que no haya pasado por todas las categorías, y el caso de que alguien haya pasado correctamente por todas las categorías.

Además hemos comprobado de paso, el correcto funcionamiento del disparador obligatorio.



Consulta 4: Victorias irregulares: victorias de un club con algún caso de dopaje en el mes anterior a la fecha de la victoria.

En esta consulta, hemos buscado primero los dopados, obteniendo los clubes a los que pertenecen. De ahí, hemos obtenido para cada club, cuando gana la competición.

Después hemos comparado la fecha de la victoria y la del dopaje, quedándonos solo con los que tenían evidencia en el año anterior, o actual, de ganar la competición.

Consulta en álgebra relacional:

π Nom_Club, País, Deporte, Categoría, Nom_Competicion, Edición, División σ Control_1 > (Fecha_gana-1) AND
 Control_1 < (Fecha_gana+1) (Ganador θ Ganador.Nom_Club=Nombre_Club AND Ganador.Pais=pais_compara
 [π Nombre_Club, País \equiv pais_compara, Fecha_Control \equiv Control_1 σ Sustancia IS NOT NULL (Ficha θ
 Historico_Ficha.Deportista=Evidencia.Sujeto_Control Evidencia) \cup
 (π Nom_Club, País, Fecha_Control σ Sustancia IS NOT NULL (Historico_Ficha θ
 Historico_Ficha.Deportista=Evidencia.Sujeto_Control Evidencia)))]

En SQL sería:

```

SELECT Nom_Club, Pais, Deporte, Categoria, Nom_Competicion,
Edicion, Division
FROM Ganador JOIN
    (SELECT DISTINCT Nombre_Club, Pais AS pais_compara,
    to_number(to_char(to_date(Fecha_Control, 'YYYY-MM-DD
    HH24:MI'), 'YYYY')) AS Control_1
    FROM Ficha JOIN Evidencia ON
    Ficha.Deportista=Evidencia.Sujeto_Control WHERE
    Sustancia IS NOT NULL
    UNION
    (SELECT Nom_Club, Pais,
    to_number(to_char(to_date(Fecha_Control, 'YYYY-MM-DD
    HH24:MI'), 'YYYY'))
    FROM Historico_Ficha JOIN Evidencia ON
    Historico_Ficha.Deportista=Evidencia.Sujeto_Control
    WHERE Sustancia IS NOT NULL))
ON Ganador.Nom_Club=Nombre_Club AND Ganador.Pais=pais_compara
WHERE Control_1 BETWEEN (Fecha_gana-1) AND (Fecha_gana+1);
  
```

```
SQL> select count(*) from (
  2 SELECT Nom_Club, Pais, Deporte, Categoria, Nom_Competicion, Edicion, Division
  3 FROM Ganador JOIN
  4 (SELECT DISTINCT Nombre_Club, Pais AS pais_compara, to_number(to_char(to_date(Fecha_Control,
'YYYY-MM-DD HH24:MI'),'YYYY'))AS Control_1
  5 FROM Ficha JOIN Evidencia ON Ficha.Deportista=Evidencia.Sujeto_Control WHERE Sustancia IS NOT
  6 NULL
  7 UNION
  8 (SELECT Nom_Club, Pais, to_number(to_char(to_date(Fecha_Control, 'YYYY-MM-DD HH24:MI'),'YYYY'
  9 ))
  10 FROM Historico_Ficha JOIN Evidencia ON Historico_Ficha.Deportista=Evidencia.Sujeto_Control WH
ERE Sustancia IS NOT NULL))
  11 ON Ganador.Nom_Club=Nombre_Club AND Ganador.Pais=pais_compara
  12 WHERE Control_1 BETWEEN (Fecha_gana-1) AND (Fecha_gana-1));

COUNT(*)
-----
      605
```

Para la prueba, borramos todo de evidencia, con lo que los clubes con casos de dopaje, no se dan ningunos:

```
SQL> UPDATE Ganador
  2 SET Fecha_gana=000000
  3 WHERE Nom_Competicion='LXXX Competición juvenil de Balonmano'
  4 AND Edicion='LXXX ed';

3 filas actualizadas.

SQL> select count(*) from (
  2 SELECT Nom_Club, Pais, Deporte, Categoria, Nom_Competicion, Edicion, Division
  3 FROM Ganador JOIN
  4 (SELECT DISTINCT Nombre_Club, Pais AS pais_compara, to_number(to_char(to_date(Fecha_Control,
'YYYY-MM-DD HH24:MI'),'YYYY'))AS Control_1
  5 FROM Ficha JOIN Evidencia ON Ficha.Deportista=Evidencia.Sujeto_Control WHERE Sustancia IS NOT
  6 NULL
  7 UNION
  8 (SELECT Nom_Club, Pais, to_number(to_char(to_date(Fecha_Control, 'YYYY-MM-DD HH24:MI'),'YYYY'
  9 ))
  10 FROM Historico_Ficha JOIN Evidencia ON Historico_Ficha.Deportista=Evidencia.Sujeto_Control WH
ERE Sustancia IS NOT NULL))
  11 ON Ganador.Nom_Club=Nombre_Club AND Ganador.Pais=pais_compara
  12 WHERE Control_1 BETWEEN (Fecha_gana-1) AND (Fecha_gana-1));

COUNT(*)
-----
      604
```

Además, volvemos a insertar las evidencias, y cambiamos la fecha de victoria de un caso concreto de los que nos devolvía la consulta en un primer momento:
Nos devuelve un resultado menos, luego es correcto.

Consulta 5: Media de edad de los clubes con más de tres deportistas envueltos en casos de dopaje.

En esta consulta hemos obtenido los deportistas dopados y obtenido sus datos de la ficha. Con eso conseguíamos su club, y de ahí la fecha de fundación del club.

Con una lista de clubes y fechas de fundación, se calcula la media de edad de los mismos, siempre que tuviesen más de tres deportistas dopados.



En álgebra relacional sería:

$A \equiv \Pi_{\text{Nom_Club, País}} \sigma_{\text{Sustancia IS NOT NULL (Historico_Ficha \theta \text{ Historico_Ficha.Deportista=Sujeto_Control Evidencia})}}$

$B \equiv \Pi_{\text{Nombre_Club, País}} (\text{Ficha } \theta \text{ Ficha.Deportista=Sujeto_Control } \Pi_{\text{Sujeto_Control}} \sigma_{\text{Sustancia IS NOT NULL (Evidencia))})$

$C \equiv \Pi_{\text{Nombre_Club, País}} \sigma_{\text{COUNT('x')>3}} \text{ GROUP BY Nombre_Club, País (A } \cup \text{ ALL B)}$

$\Pi_{\text{Hoy-Fecha_Fundacion}} \equiv \text{MEDIA } (\Pi_{\text{Nombre, Club.Pais, Fecha_Fundacion}} \text{ Club } \theta \text{ Nombre=Nombre_Club AND Club.pais=C.pais C})$

En SQL sería:

```

WITH A AS (SELECT Nom_Club, País FROM Historico_Ficha JOIN
Evidencia ON Historico_Ficha.Deportista=Sujeto_Control WHERE
Sustancia IS NOT NULL),
B AS (SELECT Nombre_Club, País FROM Ficha JOIN(SELECT DISTINCT
Sujeto_Control FROM Evidencia WHERE Sustancia IS NOT NULL) ON
Ficha.Deportista=Sujeto_Control),
C AS (SELECT Nombre_Club, País FROM (SELECT * FROM B UNION ALL
SELECT * FROM A) GROUP BY Nombre_Club, País HAVING Count('x')>3)

SELECT
AVG(to_number(to_char(SYSDATE, 'YYYY'))-
to_number(to_char(to_date(Fecha_Fundacion, 'YYYY-MM-DD
HH24:MI'), 'YYYY'))) AS MEDIA
FROM
(SELECT Nombre, Club.Pais, Fecha_Fundacion FROM (Club JOIN C ON
Nombre=Nombre_Club AND Club.pais=C.pais));
  
```

```

SQL> UPDATE Club
2 SET Fecha_Fundacion='2000-01-01';

3278 filas actualizadas.

SQL> WITH A AS (SELECT Nom_Club, País FROM Historico_Ficha JOIN Evidencia ON Historico_Ficha.Deporti
sta=Sujeto_Control WHERE Sustancia IS NOT NULL),
2 B AS (SELECT Nombre_Club, País FROM Ficha JOIN(SELECT DISTINCT Sujeto_Control FROM Evidencia WH
ERE Sustancia IS NOT NULL) ON Ficha.Deportista=Sujeto_Control),
3 C AS (SELECT Nombre_Club, País FROM (SELECT * FROM B UNION ALL SELECT * FROM A) GROUP BY Nombre
_Club, País HAVING Count('x')>3)
4 SELECT AVG(to_number(to_char(SYSDATE, 'YYYY'))-to_number(to_char(to_date(Fecha_Fundacion, 'YYYY-
MM-DD HH24:MI'), 'YYYY'))) AS MEDIA
5 FROM
6 (SELECT Nombre, Club.Pais, Fecha_Fundacion FROM (Club JOIN C ON Nombre=Nombre_Club AND Club.pai
s=C.pais));
  
```

MEDIA

14

Para comprobar esto, una primera comprobación sería que si no hay nada en histórico, la media de edad sería 0 (o nada en este caso al no tener datos):

```

SQL>
SQL> |
  
```

Otra comprobación es quitar un club de los que tienen más de 3 dopados, y ver si varía la media (poco, pero sí):

```

delete from Club
where Nombre='Aficionados de Atalaya del Nabar Fútbol Club S.D.'
AND Pais='Nauru' AND Fecha_Fundacion='1956-12-19';
  
```

```

SQL> WITH A AS (SELECT Nom_Club, Pais FROM Historico_Ficha JOIN Evidencia ON Historico_Ficha.Deportista=Sujeto_Control WHERE Sustancia IS NOT NULL),
2 B AS (SELECT Nombre_Club, Pais FROM Ficha JOIN (SELECT DISTINCT Sujeto_Control FROM Evidencia WHERE Sustancia IS NOT NULL) ON Ficha.Deportista=Sujeto_Control),
3 C AS (SELECT Nombre_Club, Pais FROM (SELECT * FROM B UNION ALL SELECT * FROM A) GROUP BY Nombre_Club, Pais HAVING Count('x')>3)
4 SELECT AVG(to_number(to_char(SYSDATE,'YYYY'))-to_number(to_char(to_date(Fecha_Fundacion,'YYYY-MM-DD HH24:MI'),'YYYY')) AS MEDIA
5 FROM
6 (SELECT Nombre, Club.Pais, Fecha_Fundacion FROM (Club JOIN C ON Nombre=Nombre_Club AND Club.pais=C.pais));

MEDIA
-----
59,983871
  
```

Otra comprobación que haremos (tras restaurar la tabla evidencia) es que pasaría si la fecha de fundación de todos los clubs fuese la misma (por ejemplo del 2000-01-01 debería salir la media de 14 años):

```

SQL> WITH A AS (SELECT Nom_Club, Pais FROM Historico_Ficha JOIN Evidencia ON Historico_Ficha.Deportista=Sujeto_Control WHERE Sustancia IS NOT NULL),
2 B AS (SELECT Nombre_Club, Pais FROM Ficha JOIN (SELECT DISTINCT Sujeto_Control FROM Evidencia WHERE Sustancia IS NOT NULL) ON Ficha.Deportista=Sujeto_Control),
3 C AS (SELECT Nombre_Club, Pais FROM (SELECT * FROM B UNION ALL SELECT * FROM A) GROUP BY Nombre_Club, Pais HAVING Count('x')>3)
4 SELECT AVG(to_number(to_char(SYSDATE,'YYYY'))-to_number(to_char(to_date(Fecha_Fundacion,'YYYY-MM-DD HH24:MI'),'YYYY')) AS MEDIA
5 FROM
6 (SELECT Nombre, Club.Pais, Fecha_Fundacion FROM (Club JOIN C ON Nombre=Nombre_Club AND Club.pais=C.pais));

MEDIA
-----
59,952381
  
```

Cabe destacar en esta consulta, que si seleccionamos todos los deportistas (sus CID) de la tabla evidencias, en lugar de filtrar por los distintos, tenemos que en lugar de tomar 63 valores (Clubes que cumplen la condición) para hacer la media, toma 208. No hemos sabido ver el porqué de esa diferencia de implementación, pero en cualquier caso la media varía en los decimales.



3. Vistas Requeridas Explícitamente

Vista 1: Campeones: deportistas que han ganado algo, y el qué en qué categoría, y con qué club

En esta consulta, hemos obtenido para cada ganador todos sus inscritos, y sacado los datos para identificar al deportista, al club y a la competición.

El álgebra de la consulta sería:

II Deportista, Nom_Club, País, Deporte, Nom_Competicion, Edición, Categoría, División (Inscrito * Ganador)

El SQL quedaría:

```
SQL> CREATE VIEW Campeones AS
  2 select Deportista, Nom_Club, Pais, Deporte, Nom_Competicion, Edicion, Categoria, Division
  3 FROM (Inscrito NATURAL JOIN Ganador)
  4 WITH CHECK OPTION;
```

Vista creada.

```
SQL> select count(*) from(campeones);
```

COUNT(*)
37324

Como primera comprobación vamos a borrar un ganador, y veremos que baja el número de deportistas que devuelve la consulta

```
SQL> delete from ganador where Nom_Club='Atletico de Valtorres de la Costa S.A.D.'
  2 AND Pais='Kiribati' AND Deporte='Voleibol' AND Nom_Competicion='Liga Galáctica de Voleibol seni
or'
  3 AND Categoria='senior';
```

1 fila suprimida.

```
SQL> select count(*) from campeones;
```

COUNT(*)
37316

Ya que no es muy buena esa consulta, vamos a hacer otra más completa. Vamos a insertar un nuevo deportista, que gane el solo una nueva competición, con lo que debería devolvernos una tupla más de la tabla.



```
SQL> insert into deportista values('AAAAAAAAAAAA','pepe', 'perez',NULL,'el rayas','3333-33-33');
1 fila creada.

SQL> insert into federado values('AAAAAAAAAAAA','Tenis','Colombia','RAYAS','cadete');
1 fila creada.

SQL> rollback;
Rollback terminado.

SQL> insert into deportista values('AAAAAAAAAAAA','pepe', 'perez',NULL,'el rayas','3333-33-33');
1 fila creada.

SQL> insert into federado values('AAAAAAAAAAAA','Tenis','Colombia','RAYAS','cadete');
1 fila creada.

SQL> insert into club values('real club de depiladores con cera','Colombia','0000-00-00',NULL);
1 fila creada.

SQL> insert into seccion values('real club de depiladores con cera','Colombia','Tenis','cadete','de
pilapiernas',NULL);
1 fila creada.

SQL> insert into ficha values('real club de depiladores con cera','Colombia','depilapiernas','cadet
e','Tenis','AAAAAAAAAAAA','1111-11-11');
1 fila creada.

SQL> insert into competicion values('Tenis','segadores de arbusto','I ed','cadete','única',2014,1);
1 fila creada.

SQL> insert into participante values('real club de depiladores con cera','Colombia','depilapiernas'
,'cadete','Tenis','segadores de arbusto','I ed','cadete','única');
1 fila creada.

SQL> insert into ganador values('real club de depiladores con cera','Colombia','Tenis','cadete','se
gadores de arbusto','I ed','única',2014);
1 fila creada.

SQL> insert into inscrito values('AAAAAAAAAAAA','real club de depiladores con cera','Colombia','Teni
s','cadete','segadores de arbusto','I ed','única');
1 fila creada.

SQL> select count(*) from campeones;

COUNT(*)
-----
37325
```

Efectivamente sale un ganador más.



Vista 2: deportistas no inscritos en ninguna competición pese a tener ficha (actual)

En esta vista hemos obtenido simplemente aquellos deportistas que no estén en inscrito, pero si en ficha.

El álgebra de la consulta sería:

$$\Pi_{\text{Deportista}} (\Pi_{\text{Deportista}} (\text{Ficha}) - \Pi_{\text{Deportista}} (\text{Inscrito}))$$

En SQL quedaría como:

```
SQL> CREATE VIEW Descartados AS  
2 SELECT Deportista FROM((SELECT DISTINCT Deportista FROM Ficha) MINUS (SELECT DISTINCT Deportista  
a FROM Inscrito))  
3 WITH CHECK OPTION;
```

Vista creada.

```
SQL> select count(*) from descartados;
```

COUNT(*)
0

De esto se deduce que actualmente todos los deportistas con ficha están inscritos en alguna competición, vamos a comprobarlo:

```
SQL> select count(*) from(SELECT DISTINCT Deportista FROM Ficha);
```

COUNT(*)
19060

```
SQL> select count(*) from(SELECT DISTINCT Deportista FROM Inscrito);
```

COUNT(*)
19060

4. Disparador Requerido Explícitamente

El disparador que se nos pide hacer obligatoriamente es un disparador que mueve las tuplas que se modifican o borran de la tabla Ficha a la tabla Historico_Ficha, cuando se “mueve” la tupla actualizamos el valor de la columna Fecha_fin_Ficha a la fecha de borrado o actualización (sysdate). Con “mover” nos referimos a insertar la tupla en Historico_Ficha y eliminar de Ficha.

Objetivo: Eliminar Fichas sin que estas desaparezcan de nuestra Base de Datos, para ello borramos una tupla de Ficha pero la insertamos en Historico_Ficha, por lo que no desaparece, sino que se “mueve” de una tabla a otra, y además actualizamos el valor de la columna Fecha_fin_Ficha.

Solución: Crear un disparador que después de borrar o modificar una tupla se dispare insertando la tupla que existía antes de modificarse o borrarse en la tabla Historico_Ficha.

Necesidad: Este disparador es necesario para cumplir la necesidad de almacenar las fichas cuando hay algún cambio de club, una inserción de nueva ficha para un mismo deportista, o un cambio de categoría. Además con este disparador vamos a poder llenar el Historico_Ficha.

Código Disparador en SQL:

```
CREATE OR REPLACE TRIGGER to_Historico
AFTER UPDATE OR DELETE ON Ficha
FOR EACH ROW
BEGIN
    INSERT INTO Historico_Ficha (Nom_Club, Pais, Deporte,
                                Categoria, Nom_Seccion, Deportista, Fecha_Ini, Fecha_Fin)
    VALUES (:OLD.Nombre_Club,      :OLD.Pais,      :OLD.Deporte,
            :OLD.Categoria,          :OLD.Nom_Seccion, :OLD.Deportista,
            :OLD.Fecha_Ini, to_char(sysdate, 'YYYY-MM-DD'));
END;
/
```

Pruebas:

Vemos el número de fichas que existen en la tabla Ficha:

```
SQL> select count(*) from ficha;
```

COUNT(*)
55206

Vemos el número de fichas que existen en la tabla Historico_Ficha:

```
SQL> select count(*) from historico_ficha;
```

COUNT(*)
0



Ahora vamos a modificar una tupla de la tabla Ficha:

```
SQL> SELECT * FROM Ficha WHERE Deportista='0016188Q4497' AND Categoria='juvenil';
```

NOMBRE CLUB

PAIS

NOM_SECCION

CATEGORIA

DEPORTE

DEPORTISTA FECHA_INI

Real Club de Bercial de los Viajeros S.A.D.

Puerto Rico

Sección de Hípica juvenil

juvenil

Hípica

0016188Q4497 2009-02-07

Ejecutamos la modificación, y vemos que insertamos en Historico_Ficha la tupla que anteriormente modificamos, además comprobamos que la tupla insertada existe en Ficha.

```
SQL> UPDATE Ficha
2 SET Categoria='cadete'
3 WHERE Deportista='0016188Q4497' AND Categoria='juvenil';
```

1 fila actualizada.

```
SQL> select * from Historico_ficha;
```

NOM CLUB

PAIS

DEPORTE

CATEGORIA

NOM_SECCION

DEPORTISTA

FECHA_INI

FECHA_FIN

Real Club de Bercial de los Viajeros S.A.D.

Puerto Rico

Sección de Hípica juvenil

Hípica

juvenil

0016188Q4497 2009-02-07 2014-04-03

```
SQL> SELECT * FROM Ficha WHERE Deportista='0016188Q4497' AND Categoria='cadete';
```

NOMBRE CLUB

PAIS

NOM_SECCION

CATEGORIA

DEPORTE

DEPORTISTA FECHA_INI

Real Club de Bercial de los Viajeros S.A.D.

Puerto Rico

Sección de Hípica juvenil

cadete

Hípica

0016188Q4497 2009-02-07



Comprobamos que también funciona el borrado de tuplas de la tabla Ficha, insertándose dicha tupla en Historico_Ficha.

```
SQL> DELETE FROM Ficha WHERE Deportista='IQ03164I4085' AND Categoria='senior';
```

1 fila suprimida.

```
SQL> select * from Historico_ficha;
```

NOM_CLUB

PAIS	DEPORTE	CATEGORIA
NOM_SECCION	DEPORTISTA	FECHA_INI FECHA_FIN
Real Club de Bercial de los Viajeros S.A.D.	Hípica	juvenil
Puerto Rico	00I6188Q4497	2009-02-07 2014-04-03
Sección de Hípica juvenil		

PAIS	DEPORTE	CATEGORIA
NOM_SECCION	DEPORTISTA	FECHA_INI FECHA_FIN
Practicantes de Remo de Nava de las Golondrinas S.D.	Remo	senior
Barbados	IQ03164I4085	2008-08-15 2014-04-03
Sección de Remo senior II		

NOM_CLUB

PAIS	DEPORTE	CATEGORIA
NOM_SECCION	DEPORTISTA	FECHA_INI FECHA_FIN

```
SQL> SELECT * FROM Ficha WHERE Deportista='IQ03164I4085' AND Categoria='senior';
```

ninguna fila seleccionada

5. Completitud Semántica

Vistas extra 1

En primer lugar hemos creado vistas, que serían las usadas por los usuarios para consultar de las tablas federación, club y sección. Estas serían las “tablas” con las que trabajaría el usuario, no viendo las tuplas borradas.

Se usarían como se ha dicho para implementar el no dejar borrar tuplas de federaciones, clubes o secciones.

```
CREATE VIEW User_Federacion AS
SELECT Deporte, País, Nombre FROM Federación WHERE
Fecha_disolucion IS NULL
WITH CHECK OPTION;
```

```
CREATE VIEW User_Club AS
SELECT Nombre, País, Fecha_Fundacion FROM Club WHERE
Fecha_Disolucion IS NULL
WITH CHECK OPTION;
```

```
CREATE VIEW User_Seccion AS
SELECT Nom_Club, País, Deporte, Categoria, Nombre FROM Seccion
WHERE Fecha_Disolucion IS NULL
WITH CHECK OPTION;
```




```
SQL> CREATE VIEW User_Federacion AS
  2 SELECT Deporte, País, Nombre FROM Federacion WHERE Fecha_disolucion IS NULL
  3 WITH CHECK OPTION;
```

Vista creada.

```
SQL> select count(*) from(User_Federacion);
```

```
  COUNT(*)
-----
      4822
```

```
SQL> select count(*) from federacion;
```

```
  COUNT(*)
-----
      4822
```

Comprobamos que, sin tener nada borrado, nos da las mismas tuplas.

Vista extra 2

En esta vista, se hace lo mismo que en la vista obligatoria, pero además se tienen en cuenta los de histórico.

```
CREATE VIEW Campeones_v2 AS
SELECT Deportista, Nom_Club, País, Deporte, Nom_Competicion,
Edicion,                      Categoria_Jug, Division
FROM (
  (Select * FROM (Inscrito NATURAL JOIN Ganador)) NATURAL JOIN
    ((SELECT Nombre_Club AS Nom_Club, País, Nom_Seccion,
      Categoria_Jug, Deporte, Deportista FROM
      Ficha)
    UNION
    (SELECT Nom_Club, País, Nom_Seccion, Categoria AS Categoria_Jug,
      Deporte, Deportista FROM Historico_Ficha)))
WITH CHECK OPTION;
```

DISPARADOR NO OBLIGATORIO 1 (Categorías Únicas)

Se dispara cuando insertamos una competición con categoría 'única' y existe otra competición idéntica con una categoría distinta de única. Al crear el disparador da error de tabla mutante, por lo que tenemos que crear una tabla auxiliar para poder guardar las tuplas con las que tenemos el conflicto. Una vez que insertemos las tuplas con las que tenemos el conflicto en la tabla auxiliar se disparará un segundo disparador que borre todas las tuplas de la tabla auxiliar que aparezca en competición.

Objetivo: Eliminar Competiciones de la tabla Competición cuando se inserte una tupla idéntica a una o varias existentes y que solo varían en la categoría, estas tuplas se eliminarán cuando el

valor de categoría de la tupla a insertar sea única, quedando solo insertada en la tabla Competición la tupla con la categoría única.

Solución: Crear dos disparadores y una tabla auxiliar para evitar el problema de tabla mutante. Se soluciona con un disparador before insert que nos insertara las tuplas con las que tenemos el conflicto en la tabla auxiliar. Una vez que se termina este disparador se inserta la tupla y se dispara el segundo disparador, que es after insert, que recorre la tabla auxiliar borrando todas las tuplas de la tabla Competiciones que sean iguales a las tuplas de la tabla auxiliar.

Necesidad: Este disparador es necesario para cumplir la necesidad que nos piden de que cuando insertamos una competición que ya existe pero si tiene como categoría única, las competiciones anteriores se anulan, por lo que se borran.

Código Disparador en SQL:

```
CREATE GLOBAL TEMPORARY TABLE competicionaux (  
    Deporte                VARCHAR(15),  
    Nombre                 VARCHAR(100),  
    Edicion                VARCHAR(14),  
    Categoria              VARCHAR(11),  
    Division               VARCHAR(5)  
);  
  
CREATE OR REPLACE TRIGGER Compe_cat_Unica_aux  
BEFORE INSERT ON Competicion  
FOR EACH ROW  
WHEN (NEW.Categoria = 'única')  
BEGIN  
    INSERT INTO competicionaux SELECT Deporte, Nombre, Edicion,  
    Categoria, Division FROM Competicion  
    WHERE Deporte=:NEW.Deporte AND Nombre=:NEW.Nombre AND  
    Edicion=:NEW.Edicion AND Categoria!:=:NEW.Categoria AND  
    Division=:NEW.Division;  
END;  
/  
  
CREATE OR REPLACE TRIGGER Compe_cat_Unica  
AFTER INSERT ON Competicion  
BEGIN  
    FOR fila IN (SELECT * FROM competicionaux)  
    LOOP  
        DELETE FROM Competicion  
        WHERE (Deporte=fila.Deporte AND Nombre=fila.Nombre AND  
        Edicion=fila.Edicion AND Categoria=fila.Categoria AND  
        Division=fila.Division);  
    END LOOP;  
    DELETE FROM competicionaux;
```



END;

/

Pruebas:

Comprobamos que tenemos una competición con una categoría distinta de única, que vamos a utilizar para realizar las pruebas.

Una vez seleccionada, insertamos una competición que sea idéntica excepto en la categoría, que será única, para comprobar que se borran las otras competiciones idénticas con otras categorías. Comprobamos que al volver a hacer la misma selección del principio ya no existe en la tabla al haber insertado una competición idéntica con la categoría única.

```
SQL> SELECT * FROM Competicion WHERE Deporte='Triatlón' AND Nombre='CCXXII Copa de Triatlón en Villatiesa del Secaral' AND Edicion='CCXXII ed' AND Categoria='alevín' AND Division='única';
```

DEPORTE

NOMBRE

EDICION	CATEGORIA	DIVIS	AÑO	PREMIO
---------	-----------	-------	-----	--------

Triatlón				
CCXXII Copa de Triatlón en Villatiesa del Secaral				
CCXXII ed	alevín	única	2012	16050000

```
SQL> INSERT INTO Competicion VALUES ('Triatlón', 'CCXXII Copa de Triatlón en Villatiesa del Secaral', 'CCXXII ed', 'única', 'única', 2013, 10);
```

1 fila creada.

SQL>

```
SQL> SELECT * FROM Competicion WHERE Deporte='Triatlón' AND Nombre='CCXXII Copa de Triatlón en Villatiesa del Secaral' AND Edicion='CCXXII ed' AND Categoria='alevín' AND Division='única';
```

ninguna fila seleccionada

DISPARADOR NO OBLIGATORIO 2 (Borrado Deportistas)

Este disparador se dispara cuando intentamos borrar una tupla de la tabla Deportista. Según el enunciado, debemos de almacenar a los deportistas de los que existan casos de dopaje, por lo que no podremos borrarlos, en cambio con los deportistas que no tengan casos de dopaje si podremos borrarlos. En el disparador comprobamos si existen evidencias de dopaje, y en el caso de que exista la evidencia, se lanzara una excepción impidiendo el borrado, cuando no exista la evidencia se borrara.

Objetivo: Eliminar o no Deportistas de la tabla Deportista, estas tuplas se borran cuando se intente eliminar una tupla de la tabla (Deportista) que no tenga evidencias de dopaje, en el caso de que las tenga se evitara el borrado.

Solución: Crear un disparador before delete que compruebe si el deportista a borrar tiene evidencias de dopaje o no.

Necesidad: Este disparador es necesario para cumplir la necesidad que nos piden diciendo que los deportistas pueden solicitar su baja, pero si existen evidencias de dopaje sus datos no pueden desaparecer, por lo que no hay que borrarlos.



Código Disparador en SQL:

```

CREATE OR REPLACE TRIGGER borra_deportista
BEFORE DELETE ON Deportista
FOR EACH ROW
DECLARE X NUMBER:=0;
BEGIN
  SELECT COUNT(*) INTO X FROM (SELECT DISTINCT Sujeto_Control FROM
  Evidencia WHERE Sujeto_Control=:OLD.CID);
  IF X>0 THEN
    RAISE_APPLICATION_ERROR (-20101,'Existen evidencias de
    dopaje del deportista que se desea de borrar, no puede ser
    borrado');
  END IF;
END;
/
  
```

Pruebas:

Vemos un deportista que si tenga evidencias

Disparador creado.

```
SQL> SELECT * FROM Evidencia WHERE Sujeto_Control='00Q3809I3722';
```

COD_JUZGADO	FECHA_JUZG	SUJETO_CONTR	FECHA_CONTROL	SUSTANCIA
1	2012-12-17	00Q3809I3722	2012-12-16 13:02	manzanilla

dopaje grave

Intentamos borrarlo de deportistas, además comprobamos que no se ha borrado.

```

SQL> delete from deportista where CID='00Q3809I3722';
delete from deportista where CID='00Q3809I3722'
*
ERROR en línea 1:
ORA-20101: Ese deportista se ha dopado, no puede ser borrado
ORA-06512: en "USER0341.BORRA_DEPORTISTA", línea 5
ORA-04088: error durante la ejecución del disparador
'USER0341.BORRA_DEPORTISTA'
  
```

```
SQL> SELECT * from deportista where CID='00Q3809I3722';
```

CID	NOMBRE
00Q3809I3722	Fernando

APODO	FECHA_NAC
Vivanco	
Pelaez	
Vivanco	1979-12-08



Ahora vamos a intentar borrar un deportista sin evidencia

```
SQL> SELECT * FROM Evidencia WHERE Sujeto_Control='I005868I4321';  
ninguna fila seleccionada  
SQL> delete from deportista where CID='I005868I4321';  
1 fila suprimida.  
SQL> SELECT * from deportista where CID='I005868I4321';  
ninguna fila seleccionada
```

DISPARADOR NO OBLIGATORIO 3 ("Borrado" de Federaciones)

Este disparador se hace para dispararse con la vista Federaciones_Activas, ya que no podemos hacerlo directamente desde la tabla Federación, porque no queremos borrar la Federación realmente, sino cambiar la fecha de disolución de NULL a la fecha (sysdate) del momento en el que se "Borra" de la vista creada.

Objetivo: Actualizar el valor de la columna Fecha_Disolucion de la tabla Federación, eligiendo la fecha de cuando el usuario borre piense que ha borrado la federación, ya que no la borra, sino que deja de verla al ser una vista.

Solución: Crear una vista y un disparador que actúe sobre la vista que se disparara cuando el usuario quiera borrar una competición de la vista, haciendo que el disparador actualice la fecha de la columna Fecha_Disolucion de la tupla borrada.

Necesidad: Este disparador es necesario para poder darle valor a la columna Fecha_Disolucion.

Código Disparador en SQL:

```
CREATE VIEW Federaciones_Activas AS  
SELECT * FROM Federacion WHERE Fecha_disolucion IS NULL  
WITH CHECK OPTION;
```

```
CREATE OR REPLACE TRIGGER Federaciones_Borrado  
INSTEAD OF DELETE ON Federaciones_Activas  
FOR EACH ROW  
BEGIN  
    UPDATE Federacion SET Fecha_disolucion=to_char(sysdate, 'YYYY-MM-DD') WHERE (Deporte=:OLD.Deporte AND Pais=:OLD.Pais);  
END;  
/
```



Pruebas:

Para este tipo de consulta, se presupone que la tabla a la que “tiene acceso” el usuario es en realidad una vista, que tan solo muestra aquellas tuplas que no tengan fecha de fin, y que el usuario no tiene acceso a la tabla federaciones.

Por tanto lo que vamos a hacer es, cuando intenten borrar una tupla de federación (o las otras dos tablas antes mencionadas), lo que se hará es actualizar esa tupla de la tabla original, y dado que el usuario solo tiene acceso a la vista, y esta no muestra las que tiene fecha de disolución, para el usuario será como si se hubiese borrado, cuando en realidad no se ha hecho.

Tenemos la vista Federaciones_Activas creada, con la característica antes mencionada.

Probamos a borrar de ella una tupla, y comprobamos si se ha “borrado” de la vista, y si en realidad sigue en la tabla Federaciones o no.

```
SQL> SELECT * FROM Federaciones_Activas WHERE (Deporte='Fútbol Sala' AND Pais='Nicaragua');
```

DEPORTE	PAIS

NOMBRE	

FECHA_DISO	

Fútbol Sala	Nicaragua
Gran Federación de Fútbol Sala de Nicaragua	

```
SQL> DELETE FROM Federaciones_Activas WHERE (Deporte='Fútbol Sala' AND Pais='Nicaragua');
```

1 fila suprimida.

```
SQL> SELECT * FROM Federaciones_Activas WHERE (Deporte='Fútbol Sala' AND Pais='Nicaragua');
```

ninguna fila seleccionada

```
SQL> SELECT * FROM Federacion WHERE (Deporte='Fútbol Sala' AND Pais='Nicaragua');
```

DEPORTE	PAIS

NOMBRE	

FECHA_DISO	

Fútbol Sala	Nicaragua
Gran Federación de Fútbol Sala de Nicaragua	
2014-04-03	

DISPARADOR NO OBLIGATORIO 4 (No insertar competición cuando existe con categoría única)

Este disparador se hace para evitar que se inserten competiciones que sean idénticas a las existentes en la tabla Competición si estas tuplas de competición poseen una categoría única y la tupla que queremos insertar no tiene como categoría única. Cuando se da esta situación, no insertaremos la tupla, ya que tenemos en la tabla Competición la misma tupla pero con categoría única. Cuando no se inserte la tupla nos lanzara una excepción que muestra que no se puede insertar.

Objetivo: Evitar insertar una competición en la tabla Competición cuando en esta tabla exista una tupla con categoría única y la tupla que se desea insertar es idéntica pero con otra categoría diferente.

Solución: Crear un disparador before insert que compruebe si existe o no una o varias tuplas idénticas a la competición que se desea insertar, y si hay alguna tupla existente que tenga la categoría única se impedirá la inserción, y si no existe se inserta.

Necesidad: Este disparador es necesario para poder completar aun mas la misma necesidad que solucionamos con el disparador no obligatorio 1.

Código Disparador en SQL:

```

CREATE OR REPLACE TRIGGER Compe_no_insertar
BEFORE INSERT ON Competicion
FOR EACH ROW
WHEN (NEW.Categoria!='única')
DECLARE X NUMBER:=0;
BEGIN
    SELECT COUNT(*) INTO X FROM (SELECT * FROM Competicion
    WHERE Deporte=:NEW.Deporte AND Nombre=:NEW.Nombre AND
    Edicion=:NEW.Edicion AND Categoria='única' AND
    Division=:NEW.Division);
    IF(X>0) THEN
        RAISE_APPLICATION_ERROR (-20101,'Esta competicion no puede
        ser insertada, debido a la existencia de la misma
        Competicion con Categoria única');
    END IF;
END;
/
  
```

Pruebas:

Comprobamos que no podemos insertar una competición con una categoría que no sea única si ya existe una competición idéntica con la categoría única.

```

SQL> SELECT * FROM Competicion WHERE Deporte='Triatlón' AND Nombre='CCXXII Copa de Triatlón en Villatiesa del Secaral' AND Edicion='CCXXII ed' AND Categoria='única' AND Division='única';
  
```

DEPORTE

NOMBRE

EDICION	CATEGORIA	DIVIS	AÑO	PREMIO
Triatlón				
CCXXII Copa de Triatlón en Villatiesa del Secaral				
CCXXII ed	única	única	2013	10

```

SQL> INSERT INTO Competicion VALUES ('Triatlón', 'CCXXII Copa de Triatlón en Villatiesa del Secaral', 'CCXXII ed', 'cadete', 'única', 2013, 10);
  
```

```

INSERT INTO Competicion VALUES ('Triatlón', 'CCXXII Copa de Triatlón en Villatiesa del Secaral',
*
  
```

ERROR en línea 1:

```

ORA-20101: Esta competicion no puede ser insertada
ORA-06512: en "USER0341.COMPE_NO_INSERTAR", línea 5
ORA-04088: error durante la ejecución del disparador
'USER0341.COMPE_NO_INSERTAR'
  
```

```

SQL> SELECT * FROM Competicion WHERE Deporte='Triatlón' AND Nombre='CCXXII Copa de Triatlón en Villatiesa del Secaral' AND Edicion='CCXXII ed' AND Categoria='cadete' AND Division='única';
  
```

ninguna fila seleccionada



Vemos como nos da el error de que no puede insertar la competición, y si insertamos otra competición de la que no hay única, vemos como se crea la tupla.

```
SQL> INSERT INTO Competicion VALUES ('Triatlón', 'Copa de Triatlón en Villatiesa del Secaral Mota', 'CCXXII ed', 'cadete', 'única', 2013, 10);
```

1 fila creada.

6. Conclusiones

Somos conscientes de que la solución planteada no cubre todos los casos, a pesar de haberle dedicado gran número de horas y esfuerzo. Aun así creemos que hemos mejorado en cierta medida el diseño del que partíamos, con los disparadores y vistas creadas, permitiendo por ejemplo implementar las características de borrado que nos pedía el cliente.