Capítulo 1

Vistas

Consideremos una base de datos que nos sirve para gestionar los equipos de hockey de varias ciudades, así como sus jugadores. El modelo E-R para esta base de datos se muestra en la Figura 1.1.

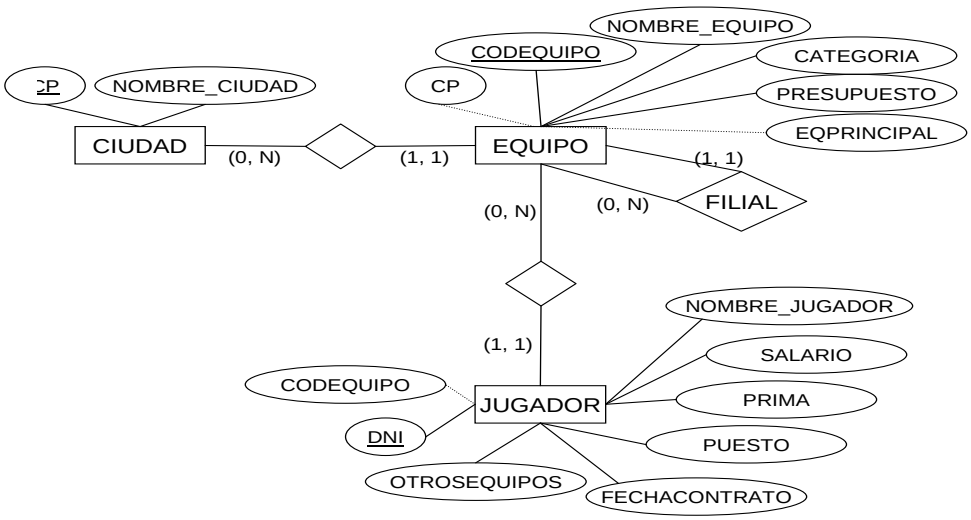

Figura 1.1: Modelo E-R de equipos de hockey

Una vez traducido al modelo relacional, e implementado en una base de datos, la sentencia de creación de las tablas sería la siguiente (además, se muestra, entre comentarios, la descripción de los campos):

```
CREATE TABLE CIUDAD(
   CP NUMBER(5) NOT NULL,        -- Código postal principal de la ciudad
   NOMBRE_CIUDAD VARCHAR2(15),   -- Nombre de la ciudad
   CONSTRAINT PK_CIUDAD PRIMARY KEY(CP)
   );

CREATE TABLE EQUIPO(
   CODEQUIPO CHAR(10) NOT NULL,  -- Código del equipo
   NOMBRE_EQUIPO VARCHAR2(20),   -- Nombre del equipo
   CATEGORIA VARCHAR2(20),       -- Categoría en la que juega
```

```
   PRESUPUESTO NUMBER(11,2),     -- Presupuesto del equipo, en euros
   EQPRINCIPAL CHAR(10),         -- Código del equipo principal, si es filial
   CP NUMBER(5),                 -- Código postal de la ciudad sede del equipo
   CONSTRAINT PK_EQUIPO PRIMARY KEY(CODEQUIPO),
   CONSTRAINT FK_CIUDAD FOREIGN KEY(CP) REFERENCES CIUDAD(CP),
   CONSTRAINT FK_EQPRIN FOREIGN KEY(EQPRINCIPAL) REFERENCES EQUIPO(CODEQUIPO)
   );

CREATE TABLE JUGADOR(
   DNI CHAR(11) NOT NULL,        -- DNI del jugador
   NOMBRE_JUGADOR VARCHAR2(15),  -- Nombre del jugador
   SALARIO NUMBER(7,2),          -- Salario anual del jugador, en euros
   PRIMA NUMBER(7,2),            -- Prima anual fija del jugador, en euros
   PUESTO VARCHAR2(10),          -- Puesto del jugador
   FECHACONTRATO DATE,           -- Fecha de contrato
   OTROSEQUIPOS NUMBER(2),       -- Número de (otros) equipos en que ha jugado
   CODEQUIPO CHAR(10),           -- Código del equipo en el que juega
   CONSTRAINT PK_JUGADOR PRIMARY KEY(DNI),
   CONSTRAINT FK_EQUIPO FOREIGN KEY(CODEQUIPO) REFERENCES EQUIPO(CODEQUIPO),
   CONSTRAINT C_PRIMAPOS CHECK (PRIMA >=0),
   CONSTRAINT C_SALPOS CHECK (SALARIO >=0)
   );
```

Para los ejemplos que se utilizarán en este tema, las tablas tienen la siguiente extensión:

```
  -- CIUDAD --
       CP NOMBRE_CIUDAD
---------- ---------------
     15000 A CORUÑA
     28000 MADRID
     27000 LUGO

3 filas seleccionadas.

  -- EQUIPO --
CODEQUIPO NOMBRE_EQUIPO CATEGORIA           PRESUP EQPRINCIPA CP
--------- ------------- ------------------ -------- -------- ----------
HCL         H.C. LICEO  DIVISION DE HONOR   1500000 <NULO>  15000
LB          LICEO B     PRIMERA DIVISION    900000  HCL     15000
ALC         ALCOBENDAS  DIVISION DE HONOR   2000000 <NULO>  28000
CHM         C.H. MADRID DIVISION DE HONOR   3100000 <NULO>  28000

4 filas seleccionadas.

  -- JUGADOR --
DNI     NOMBRE_JUGADOR  SALARIO PRIMA   PUESTO    FECHA  OTROS COD
------  --------------- ------- ------ ---------- ------- ---- -----
12345678  PEREZ, PEDRO  20000   <NULO>  PORTERO   01/06/01 1 HCL
12335678  PEREZ, PABLO  23000   3000    DELANTERO 01/09/01 2 HCL
22552278  ALVAREZ, JUAN 18000   0       MEDIO     14/02/02 0 HCL
33333334  ZAS, ANTONIO  17500   0       MEDIO     05/11/01 0 HCL
28342228  PARRA, JOSE   10000   <NULO>  PORTERO   21/01/02   LB
28321321  CASTOR, LUIS  9000    2000    DELANTERO 01/06/02 2 LB
24555555  CASA,FERNANDO 95000   <NULO>  MEDIO     23/04/02 0 LB
```

```
22178678  ARAUJO, JUAN  40000   0       PORTERO   11/02/02 1 ALC
22375989  ARAS, MIGUEL  23000   0       DELANTERO 21/09/01 2 ALC
22511111  ALVAREZ,ANGEL 19000   0       MEDIO     14/02/02 0 ALC

10 filas seleccionadas.
```

1.1 Definición de vista

La definición más simple de una vista es la que nos dice que *una vista es una consulta a la que se da un nombre*. Una vista es una *tabla virtual* que nos permite alcanzar tres objetivos:

- la confidencialidad de la información
- la transparencia de la estructura interna de la BD
- el control de las restricciones de integridad

Se denomina virtual porque no tiene existencia propia; ella parece y se comporta como una tabla para el usuario, pero consiste a nivel interno en una sentencia SQL, que está catalogada en el diccionario de datos. Así, cada vez que se invoca a una vista, Oracle accede al diccionario de datos para lanzar la ejecución de la sentencia que le corresponde.

Una vez que la vista se ha definido, se puede acceder a ella exactamente igual que si fuese una tabla para ver su estructura o para seleccionar los datos que contiene. Es decir, las operaciones:

```
DESCRIBE <nombre_vista>
```

```
SELECT ... FROM <nombre_vista>...
```

son válidas.

El mayor problema asociado a las vistas es determinar cuándo los datos de una vista se pueden actualizar (insertar, borrar o modificar filas).

Las tablas indicadas en la consulta que define una vista se denominan *tablas base*, y pueden a su vez ser vistas.

1.2 Creación de Vistas

La sentencia utilizada para crear una vista es `CREATE VIEW`, cuya sintaxis general se muestra a continuación.

```
CREATE [OR REPLACE] VIEW nombre_vista [( esquema_vista )]
AS sentencia select
```

Si indicamos `CREATE VIEW...` y ya existe una vista con ese nombre, Oracle dará un error. Si indicamos `CREATE OR REPLACE VIEW...` y la vista existe, se sustituye la declaración de la vista existente por la nueva.

La `sentencia select` es una consulta realizada utilizando la sentencia `SELECT`. El nombre `nombre_vista` especifica el nombre que se da a la vista que se crea. Así, por ejemplo, la siguiente sentencia define una vista sobre la tabla `JUGADOR`, extrayendo solamente el DNI y nombre.

```
CREATE VIEW JUGS
     AS SELECT DNI, NOMBRE_JUGADOR
        FROM JUGADOR;

Vista creada.
```

Una vez definida, la vista se puede consultar como si fuese una tabla. Los nombres de los atributos de la vista serán los mismos atributos que se seleccionan de la(s) tabla(s). Así, veamos la vista anterior:

```
DESCRIBE JUGS
 Nombre                         >Nulo?   Tipo
 ------------------------------ -------- --------------
 DNI                            NOT NULL CHAR(11)
 NOMBRE_JUGADOR                          VARCHAR2(15)
```

La vista anterior tiene el esquema `JUGS(DNI,NOMBRE_JUGADOR)`. Esto es posible cuando la consulta selecciona atributos o expresiones que toman un nombre de atributo válido. Sin embargo, la siguiente definición de vista no sería válida en Oracle:

```
CREATE VIEW V1
     AS SELECT NOMBRE_JUGADOR, SALARIO+PRIMA -- Erróneo!!
         FROM JUGADOR;

AS SELECT NOMBRE_JUGADOR, SALARIO+PRIMA -- Erróneo!!
                                    *
ERROR en línea 1:
ORA-00998: debe proporcionar un nombre a esta expresión con un alias de columna
```

El motivo de ser una definición de vista errónea, como sugiere el error de Oracle, es que `SALARIO+PRIMA` no es un nombre de atributo válido. Lo mismo ocurriría si se selecciona cualquier expresión que no produzca un identificador válido para el atributo[1].

Hay dos formas de solucionar este problema:

1. Especificar un alias para las expresiones que no corresponden a nombres de atributos:

   ```
   CREATE VIEW V1
       AS SELECT NOMBRE_JUGADOR, SALARIO+PRIMA AS SALARIO_TOTAL
          FROM JUGADOR;

   Vista creada.
   ```

   En este caso, la vista tendrá un atributo `NOMBRE_JUGADOR` y otro `SALARIO_TOTAL`.

2. Utilizar la parte opcional `esquema_vista`, indicando (en una lista entre paréntesis) los nombres que se darán a los atributos de la vista que representan las expresiones seleccionadas:

   ```
   CREATE OR REPLACE VIEW V1 (NOMBRE_JUGADOR, SALARIO_TOTAL)
       AS SELECT NOMBRE_JUGADOR, SALARIO+PRIMA
          FROM JUGADOR;

   Vista creada.
   ```

   El esquema de la vista sería el mismo que en el caso anterior.

[1] Un identificador válido en Oracle está compuesto de letras del alfabeto inglés, números y el guión bajo, empezando por una letra, o bien ir entre comillas dobles.

Cualquiera de las dos opciones indicadas son necesarias cuando se selecciona alguna expresión que produce un nombre de atributo inválido. Además, también cualquiera de ellas se puede utilizar para renombrar los atributos, dándoles un nombre diferente a los que tienen en la tabla, como en el ejemplo siguiente:

```
 CREATE OR REPLACE VIEW JUGS
     AS SELECT DNI AS DNIJUGADOR, NOMBRE_JUGADOR AS NOMBRE
        FROM JUGADOR;

Vista creada.

 CREATE OR REPLACE VIEW JUGS(DNIJUGADOR,NOMBRE)
     AS SELECT DNI, NOMBRE_JUGADOR
         FROM JUGADOR;

Vista creada.
```

Usando la segunda opción, definir el esquema de la vista, implica que hay que hacer la definición *completa* del esquema. Así, la primera de las siguientes vistas es correcta, pero la segunda incorrecta ya que el número de atributos definido en el esquema no es el mismo que el número de expresiones seleccionadas:

```
 CREATE OR REPLACE VIEW SUMAEQ (CODIGO, SUMASALARIO)
      AS SELECT CODEQUIPO, SUM(SALARIO)
          FROM JUGADOR
          GROUP BY CODEQUIPO;

Vista creada.

  CREATE OR REPLACE VIEW SUMAEQ (SUMASALARIO)
      AS SELECT CODEQUIPO, SUM(SALARIO)
          FROM JUGADOR
          GROUP BY CODEQUIPO;
 CREATE OR REPLACE VIEW SUMAEQ (SUMASALARIO)
                                *
ERROR en línea 1:
ORA-01730: número de nombres de columna especificado no válido
```

Como se ve en los ejemplos anteriores, las consultas pueden ser todo lo complejas que se quieran, incluyendo cláusulas como `WHERE`, `GROUP BY`, `HAVING`, `UNION` e incluso `ORDER BY`. Así, la siguiente consulta nos ofrece una relación en la que se ven los nombres de los jugadores, el número de otros equipos, y el nombre del equipo al que pertenecen, para aquellos jugadores con prima:

```
 CREATE OR REPLACE VIEW UNA_VISTA (JUGADOR, NUM_EQUIPOS, EQUIPO)
     AS SELECT NOMBRE_JUGADOR, OTROSEQUIPOS, NOMBRE_EQUIPO
        FROM JUGADOR J, EQUIPO E
        WHERE J.CODEQUIPO = E.CODEQUIPO
             AND PRIMA > 0;

Vista creada.
```

Una consulta sobre esta vista nos daría el resultado siguiente:

```
 SELECT * FROM UNA_VISTA;
```

```
JUGADOR          NUM_EQUIPOS EQUIPO
---------------- ----------- --------------------
PEREZ, PABLO               2 H.C. LICEO
CASTOR, LUIS               2 LICEO B

2 filas seleccionadas.
```

1.3 Borrado de vistas

Para borrar una vista (sólo la definición, no los datos) se utiliza la sentencia siguiente:

```
DROP VIEW nombre_vista
```

Por ejemplo,

```
DROP VIEW UNA_VISTA;

Vista borrada.
```

1.4 Ventajas del uso de vistas

Existen múltiples ventajas derivadas del uso de vistas en la definición de un esquema de bases de datos. Entre ellas destacan las siguientes, de las que se da algún ejemplo:

- Ocultación de información:

  Ejemplo: Un empleado puede gestionar datos de los jugadores, pero quizás no debería ver cierta información, como por ejemplo el salario. Para esto se puede definir una vista que obtenga los datos de los jugadores, pero no el atributo salario ni prima. Así, la persona que gestiona los empleados no accede a información a la que no debe.

- Independencia con respecto al diseño de tablas: Si se cambia la estructura de alguna tabla, como por ejemplo de algún atributo de la misma (que no está incluido en la definición de la vista), eso no afecta a los programas que acceden a la vista.

  Ejemplo: Si añadimos un atributo "fecha de nacimiento" para los jugadores, los programas que acceden a la vista `JUGADORES` no se verán afectados.

- Facilitar la realización de consultas: Una vista puede definir una consulta relativamente compleja, y se pueden hacer a su vez consultas sobre esta vista. Así, si en algún caso un usuario debe realizar consultas complejas y no es experto en SQL, el uso de una vista que haga parte de la consulta de de gran ayuda.

  Ejemplo: En una BD de contabilidad existe una tabla en la que se almacenan los apuntes de los asientos contables. Podríamos definir una vista para calcular los saldos del debe y el haber de las cuentas, y un usuario normal tendría que hacer una consulta realmente simple para obtener el saldo de una determinada cuenta (del tipo `SELECT * FROM VISTA WHERE CUENTA=<cuenta>`) en vez de tener que definir la consulta completa, con sumas, agrupamientos, etc.

1.5 Consultas sobre vistas

Como ya se ha indicado, realizar una consulta sobre una vista es exactamente igual (sintácticamente) que realizar una consulta sobre una tabla real. Considérese la siguiente definición de vista, que nos da el nombre de los equipos y el número de jugadores que tiene.

```
 CREATE VIEW JUGS_POR_EQUIPO
     AS SELECT NOMBRE_EQUIPO, COUNT(*) AS NUMJUGS
         FROM EQUIPO E, JUGADOR J
         WHERE E.CODEQUIPO=J.CODEQUIPO
         GROUP BY NOMBRE_EQUIPO;

Vista creada.
```

Ahora obtengamos, por ejemplo, los nombres de equipos que tengan exactamente 4 jugadores en la base de datos. La siguiente consulta los obtiene:

```
 SELECT NOMBRE_EQUIPO
     FROM JUGS_POR_EQUIPO
     WHERE NUMJUGS=4;

NOMBRE_EQUIPO           NUMJUGS
-------------------- ----------
H.C. LICEO                   4

1 fila seleccionada.
```

Como se ve, es bastante más sencilla que la consulta que deberíamos usar si no existiese la vista:

```
  SELECT NOMBRE_EQUIPO
      FROM EQUIPO E, JUGADOR J
      WHERE E.CODEQUIPO=J.CODEQUIPO
      GROUP BY NOMBRE_EQUIPO
      HAVING COUNT(*)=4;

NOMBRE_EQUIPO
--------------------
H.C. LICEO

1 fila seleccionada.
```

La ejecución (teórica) de la consulta que usa la vista seguiría 2 pasos:

1. Ejecutar la consulta que define la vista, obteniendo una relación temporal que contiene los datos de la vista.

2. Ejecutar la consulta pedida sobre esa relación temporal.

Lógicamente, la realización de una consulta sobre una vista se traslada a una consulta a las tablas base sobre las que se define la vista. Sin embargo, normalmente, los gestores de bases de datos implementan algoritmos de optimización sobre las consultas, por lo que las condiciones establecidas en la consulta se suelen añadir a las establecidas en la definición de la vista. Así, la consulta anterior quedaría probablemente definida (automáticamente y de forma transparente para el usuario que tecleó la consulta) como la que se ha escrito sin usar la vista, añadiendo la condición en la cláusula `HAVING`.

1.6 Actualización de vistas

Al igual que cuando se hace una consulta, cuando se lanza una sentencia de actualización de datos (`INSERT`, `UPDATE`, `DELETE`) sobre una vista, el gestor de bases de datos tratará de trasladar esa actualización a la tabla o tablas base sobre las que está definida la vista. Al intentar una actualización, en algunos casos es posible realizarla, pero en otros no es posible y el gestor dará un error.

De forma genérica, podemos dividir las vistas en las siguientes categorías:

Todas las vistas: Todas las vistas que pueden ser definidas.

Vistas teóricamente actualizables: Es un subconjunto del anterior, formado por las vistas que en teoría se podrían actualizar. Es decir, la traducción de actualizar los datos en la vista a actualizar los datos en las tablas base es teóricamente posible.

Vistas realmente actualizables: Es un subconjunto de las vistas teóricamente actualizables, formado por aquellas vistas que en realidad son consideradas actualizables por los gestores de bases de datos. Probablemente cada gestor de bases de datos permitirá la actualización de datos en un subconjunto (reducido) de este conjunto de vistas. Veremos concretamente la reacción de Oracle ante el intento de actualización de varios tipos de vistas.

A continuación veremos varios tipos de consultas que se podrían usar para definir una vista, y qué es lo que pasaría, utilizando Oracle, si tratamos de actualizar datos en la vista. Como regla general, una vista es actualizable si las tuplas de las tablas base son *"back-traceable"*, esto es, cuando es posible identificar la tupla de la tabla base que corresponde a cada tupla de la vista. Sin embargo, veremos por separado si se pueden insertar, borrar o modificar datos de las vistas creadas.

1.6.1 Sólo selección de filas o proyección de atributos de una tabla

Proyecciones de atributos: Veamos una vista definida proyectando atributos de una sola tabla, y las posibles actualizaciones sobre ella.

Consideremos el siguiente ejemplo de vista, que obtiene el nombre y salario de todos los jugadores (es decir, se hace solamente una proyección de datos):

```
CREATE VIEW VISTA1
    AS SELECT NOMBRE_JUGADOR, SALARIO
          FROM JUGADOR;
```

Consideremos las posibles actualizaciones sobre esta vista. Evidentemente, se tratará de traducir todas las actualizaciones sobre la vista a actualizaciones sobre la tabla `JUGADOR`.

- Inserción: Se pueden dar varios casos. Si alguno de los atributos de la tabla base que no están en la vista no admite valores nulos ni tiene valores por defecto, la inserción no es posible (ya que la traducción intentaría insertar valores nulos). Este es el caso de `VISTA1`, ya que el atributo `DNI`, que no está en la definición de la vista, no admite valores nulos, por ser la clave primaria.

  ```
  INSERT INTO VISTA1
      VALUES('NOMBRE',0);
  INSERT INTO VISTA1
  *
  ERROR en línea 1:
      ORA-01400: no se puede realizar una inserción NULL en ("SCOTT"."JUGADOR"."DNI")
  ```

  Como regla general, en el resto de los casos sí es posible la inserción; es decir, cuando los atributos de la tabla base no incluidos en la definición de la vista o bien admiten valores nulos o tienen definidos valores por defecto. Sin embargo, hay algunos casos en los que no es posible.

- Borrado: Siempre es posible, ya que las tuplas de la vista en este caso se corresponden una a una con las de la tabla base. Por lo tanto, las tuplas que se borren de la vista se borran realmente de la tabla base.

  En este sentido, debemos matizar algo. Veamos la siguiente vista y una consulta sobre ella:

```
 CREATE VIEW VSALARIO
     AS SELECT SALARIO
          FROM JUGADOR;

Vista creada.

 SELECT * FROM VSALARIO
     ORDER BY SALARIO;

 SALARIO
--------
    9000
   10000
   17500
   18000
   19000
   20000
   23000
   23000
   40000
   95000

10 filas seleccionadas.
```

  Como se ve, hay dos tuplas con salario 23000. Si intentamos borrar de la vista `VSALARIO`, no podemos distinguir entre esas dos tuplas, por lo que si ejecutamos la sentencia

```
 DELETE FROM VSALARIO
     WHERE SALARIO=23000;

2 filas suprimidas.

 SELECT COUNT(*) FROM JUGADOR;

  COUNT(*)
----------
         8

1 fila seleccionada.
```

  Vemos que se borran las dos tuplas correspondientes en la tabla base. Que *nosotros* no podamos diferenciar las tuplas no indica que Oracle no pueda, internamente, hacerlo. Por ello, si la consulta es **sólo** una proyección de atributos, siempre se podrán borrar los datos de ella.

- Modificación: Al igual que el borrado, siempre es posible modificar los datos de una vista de este tipo. Evidentemente, sólo es posible modificar aquellos atributos que aparecen en la definición de la vista.

Sólo selecciones de datos: Consideremos el siguiente ejemplo de vista, que obtiene todos los atributos de aquellos jugadores que han estado en más de 1 equipo externo. Es importante hacer notar que no se eliminan los duplicados de las filas obtenidas, es decir, no se usa la palabra clave `DISTINCT` en la definición de la vista.

```
CREATE VIEW VISTA2
   AS SELECT *
      FROM JUGADOR
      WHERE OTROSEQUIPOS > 1;
```

Veamos las posibles actualizaciones sobre este tipo de vista:

- Inserción: Siempre es posible insertar tuplas, ya que cada una de las tuplas de la vista se corresponde directamente a una tupla de la tabla base, y se conocen todos los atributos. Sin embargo, las inserciones pueden dar lugar a situaciones extrañas. Por ejemplo, si insertamos una fila en la anterior vista, con un número de equipos externos igual a 0, la fila se inserta en la tabla base `JUGADOR`, pero no aparece en la vista ya que no satisface la condición.

- Borrado: También es siempre posible, por el mismo motivo. Las tuplas que se borren de la vista se borran realmente de la tabla base. En este caso las situaciones extrañas del caso de la inserción no suceden, ya que a la condición del borrado, si existe, se añade la condición de la vista antes de realizar el borrado.

```
 DELETE FROM VISTA2;

3 filas suprimidas.

 SELECT COUNT(*) FROM JUGADOR;

  COUNT(*)
----------
         7

1 fila seleccionada.
```

  Como se ve, la sentencia sólo borra 3 tuplas, las que aparecen en la vista, pero no las demás tuplas de la tabla `JUGADOR`. De todas formas, se sugiere actuar con cuidado en estos casos y comprobar el comportamiento de cada versión de Oracle. Esta prueba, en la que no borra los datos que no están en la vista, se ha realizado con Oracle 9*i* (9.2.0.1.0).

- Modificación: También es posible siempre modificar los datos de una vista de este tipo, actuando con las mismas consideraciones que con el borrado de filas. En este caso, Oracle 9*i* sólo actualiza las filas de la tabla que están en la vista.

  Además, hay otro tipo de consideraciones. Veamos el siguiente ejemplo:

```
 UPDATE VISTA2 SET OTROSEQUIPOS=0
     WHERE DNI='12.335.678';

1 fila actualizada.

 SELECT * FROM VISTA2;
```

```
DNI         NOMBRE_JUGADOR   SALARIO  PRIMA PUESTO     FECHACON OTROS CODEQUIPO
----------- --------------- -------- ------ ---------- -------- ----- ----------
28.321.321  CASTOR, LUIS        9000   2000 DELANTERO  01/06/02     2 LB
22.375.989  ARAS, MIGUEL       23000      0 DELANTERO  21/09/01     2 ALC

2 filas seleccionadas.
```

El resultado es que la fila indicada se actualiza en la tabla base JUGADOR, pero como resultado de ello, esa fila desaparece de la vista (como si fuese borrada) porque ya no satisface la condición de haber estado en más de un equipo externo.

Combinación de proyecciones y selecciones de datos:

Combinando selecciones y proyecciones sobre una tabla se pueden obtener vistas con una gran potencia. Los problemas de este tipo de vistas combinarían los aspectos indicados individualmente para las selecciones o las proyecciones.

1.6.2 Consultas con DISTINCT o agrupamiento sobre una tabla

Consideremos la siguiente definición de vista, que utiliza DISTINCT para eliminar tuplas duplicadas:

```
CREATE VIEW VISTA3
   AS SELECT DISTINCT CODEQUIPO
     FROM JUGADOR;
```

A diferencia de las vistas definidas anteriormente, para esta no es posible saber la tupla original de la tabla base sobre la que se ha definido la vista, ya que se eliminan los duplicados.

- Inserción: No es posible, ya que al intentar trasladar la inserción a la tabla base, no se sabría cómo realizar la inserción. Por ejemplo, no se sabría cuantas copias de la fila se podrían insertar, ya que al usar DISTINCT en la definición de la vista se eliminarían los duplicados.
- Borrado: Aunque sería teóricamente posible, borrando todas las copias de la tabla base que dan lugar a cada fila de la vista que se desea borrar, no se permite el borrado, ya que eso no sería el resultado deseado en muchos casos.
- Modificación: Por el mismo motivo, aún siendo teóricamente posible la actualización, no se permite en este tipo de vistas.

Consideremos ahora una vista que incluye agrupamiento y funciones de agregación:

```
CREATE VIEW JUGS_POR_EQ
    AS SELECT CODEQUIPO, COUNT(*) AS NUMJUGS
        FROM JUGADOR
        GROUP BY CODEQUIPO;
```

De nuevo, en este caso las tuplas de la vista no se corresponden una a una con las tuplas de la tabla base. Veamos caso por caso por qué las actualizaciones no se podrán realizar:

- Inserción: No es posible, ya que de nuevo al intentar trasladar la inserción a la tabla base no se sabría cómo realizar la inserción. Por ejemplo, al intentar ejecutar

  ```
  INSERT INTO JUGS_POR_EQ VALUES('ABC',2)
  ```

  habría que insertar 3 filas en la tabla `JUGADOR`, pero no se sabe qué poner en ninguno de sus atributos excepto el código del equipo.

- Borrado: Sería teóricamente posible borrar las filas de la vista, pero por ejemplo intentar ejecutar

  ```
  DELETE FROM JUGS_POR_EQ  WHERE CODEQUIPO='HCL'
  ```

  dado que la vista sólo nos dice cuántos jugadores tiene el equipo, no parece lógico borrar todas las filas del la tabla `JUGADOR` de ese equipo. Por lo tanto, el borrado no se permite en vistas definidas con agrupamientos y/o funciones de agregación.

- Modificación: Tampoco se permiten las modificaciones en este tipo de vistas. En algún caso sería teóricamente posible, por ejemplo,

  ```
  UPDATE JUGS_POR_EQ SET CODEQUIPO='HCC' WHERE CODEQUIPO='HCL',
  ```

  pero el resultado, que sería modificar el código de equipo de los jugadores correspondientes, no sería lógico. Otro tipo de modificaciones, como

  ```
  UPDATE JUGS_POR_EQ SET NUMJUGS=2 WHERE CODEQUIPO='HCL'
  ```

  se podría llevar a cabo borrando una fila de la tabla `JUGADOR` de ese equipo (ya que actualmente tiene 3), pero tampoco sería una actuación correcta, por lo que no se permite.

Por lo tanto, como se ha visto, ninguna vista definida utilizando `DISTINCT` ni agrupamientos y/o funciones de agregación permite la actualización de los datos directamente a través de la vista.

1.6.3 Vistas definidas sobre más de una tabla

Es común definir vistas sobre más de una tabla, haciendo joins entre varias. Por ejemplo, la siguiente vista nos daría los nombres de jugadores y los nombres de los equipos a los que pertenecen:

```
CREATE VIEW JUGS_EQS
    AS SELECT NOMBRE_JUGADOR, NOMBRE_EQUIPO
        FROM JUGADOR J, EQUIPO E
        WHERE J.CODEQUIPO=E.CODEQUIPO;
```

Según el estándar SQL-92 (el que *supuestamente* sigue Oracle), cuando hay más de una tabla en la cláusula `FROM` de la sentencia `SELECT` que define la vista, no es posible realizar ningún tipo de actualización directamente sobre la vista. La razón es que no hay forma genérica fiable de trasladar los cambios deseados de la vista a las tablas base. Oracle permite, bajo algunas circunstancias, hacer algunas actualizaciones sobre este tipo de vistas, pero ya que no es estándar y es algo que puede cambiar, no entraremos en detalles sobre eso y consideraremos que **no se pueden realizar actualizaciones sobre vistas definidas sobre más de una tabla**.