Indice de los ficheros de ejemplo incluídos:

Ej_Fechas.sql

```
/* @%ORACLE_HOME%:Ej_Fechas */
SET echo ON

column f_Com    heading "fecha Comienzo" format a22;
column f_Fin    heading "fecha Final"    format a22;
column f_Dif    heading "dif.(dias)"     format 9999.99999;
column dias     heading "dias"           format 9999;
column horas    heading "h."             format 99;
column minutos  heading "m."             format 99;
column segundos heading "s."             format 99;

DROP TABLE temp;
CREATE TABLE temp(fechaCom date,
                  fechaFin date,
                  numRef   number(4),
                  difTim   number);

INSERT INTO temp  VALUES ('12-jan-94', '13-jan-94', 1, NULL);

SELECT * FROM temp;

INSERT INTO temp
  VALUES (to_date('12-5-93','DD-MM-YY'),
          to_date('16-May-93','DD-Mon-YY'),
          2,
          NULL);
SELECT * FROM temp;

INSERT INTO temp
  VALUES (to_date('12-5-93 14:23:5','DD-MM-YY HH24:MI:SS'),
          to_date('14-5-93  4:25:32 PM','DD-MM-YY HH:MI:SS AM'),
          3,
          NULL);
SELECT * FROM temp;

DROP VIEW demo;
CREATE VIEW demo (f_Com, f_Fin, f_Dif, dias, horas, minutos, segundos) AS
 SELECT to_char(fechaCom,'HH24:MI:SS DD Mon YYYY'),
        to_char(fechaFin,'HH24:MI:SS DD Mon YYYY'),
        difTim,
        trunc(difTim,0),
        trunc((difTim-trunc(difTim,0))*24,0),
        trunc((difTim*24-trunc(difTim*24,0))*60,0),
        round((difTim*24*60-trunc(difTim*24*60,0))*60,0)
 FROM temp;

UPDATE temp SET difTim = fechaFin-fechaCom;

SELECT * FROM demo;
```

creaBD_Piezas.sql

```
/* @creaBD_Piezas.sql */
SET ECHO ON

/* Crear las tablas y alguna vista, a modo de ejemplo
*/
CREATE TABLE Pieza      (
  clvPieza  number(9) PRIMARY KEY,
  nombPieza char(32)  NOT NULL,
  color     char(32));

CREATE TABLE Proveedor  (
  clvProv  number(9) PRIMARY KEY,
  nombProv char(32)  NOT NULL);

CREATE TABLE suministrar (
  clvProv  number(9),
  clvPieza number(9),
  PRIMARY KEY (clvProv, clvPieza),
  FOREIGN KEY (clvProv)  REFERENCES Proveedor(clvProv),
  FOREIGN KEY (clvPieza) REFERENCES Pieza(clvPieza) ON DELETE cascade);

CREATE VIEW infoSuministros AS
  SELECT nombProv, nombPieza
  FROM   Proveedor V, suministrar S, Pieza P
  WHERE  V.clvProv = S.clvProv AND S.clvPieza = P.clvPieza;

/* mostrar la información básica de las tablas y vistas
*/
@infoBD_Piezas.sql
```

infoBD_Piezas.sql

```
/* @infoBD_Piezas.sql */
SET ECHO ON

/* mostrar la información básica de las tablas y vistas
*/
SELECT * FROM CAT;

DESCRIBE Proveedor;
DESCRIBE Pieza;
DESCRIBE suministrar;
DESCRIBE infoSuministros;
```

elimBD_Piezas.sql

```
/* @elimBD_Piezas.sql */
SET ECHO ON

/* eliminar las tablas y vistas creadas, en el orden
   adecuado, para no violarlas restricciones
*/

DROP VIEW infoSuministros;

DROP TABLE suministrar;

DROP TABLE Pieza;

DROP TABLE Proveedor;

/* mostrar la información básica de las tablas y vistas
*/
@infoBD_Piezas.sql
```

ponDatBD_Piezas.sql

```
/* @ponDatBD_Piezas.sql */
SET ECHO ON

/* eliminar los datos existentes en las tablas
*/
@elimDatBD_Piezas.sql

/* añadir algunos datos a las tablas, para probar
*/
SET ECHO OFF

INSERT INTO Proveedor VALUES (1,     'PEREZ');
INSERT INTO Proveedor VALUES (2,     'LOPEZ');
INSERT INTO Proveedor VALUES (3,     'MARTINEZ');

INSERT INTO Pieza VALUES (91,     'TUERCA',   'ROJO');
INSERT INTO Pieza VALUES (92,     'TUERCA',   'VERDE');
INSERT INTO Pieza VALUES (93,     'TORNILLO', 'AZUL');
INSERT INTO Pieza VALUES (94,     'TORNILLO', 'ROJO');
INSERT INTO Pieza VALUES (95,     'TUERCA',   'AZUL');
INSERT INTO Pieza VALUES (96,     'PALANCA',  '');
INSERT INTO Pieza VALUES (97,     'TUBO',     'GRIS');

INSERT INTO suministrar VALUES (1, 91);
INSERT INTO suministrar VALUES (1, 92);
INSERT INTO suministrar VALUES (2, 92);
INSERT INTO suministrar VALUES (2, 93);
INSERT INTO suministrar VALUES (2, 94);
INSERT INTO suministrar VALUES (1, 95);

SET ECHO ON

/* mostrar los datos de las tablas y vista
*/
@verDatBD_Piezas.sql
```

verDatBD_Piezas.sql

```
/* @verDatBD_Piezas.sql */
SET ECHO ON

/* mostrar los datos de las tablas y vista
*/
SELECT * FROM Pieza;

SELECT * FROM Proveedor;

SELECT * FROM suministrar;

SELECT * FROM infoSuministros;
```

elimDatBD_Piezas.sql

```
/* @elimDatBD_Piezas.sql */
SET ECHO ON

/* eliminar los datos de las tablas
*/

DELETE FROM suministrar;

DELETE FROM Pieza;

DELETE FROM Proveedor;

/* mostrar los datos de las tablas y vista
*/
@verDatBD_Piezas.sql
```

pregBD_Piezas.sql

```
/* @pregBD_Piezas.sql */
SET ECHO ON

/* Piezas de color 'verde'
*/
SELECT * FROM Pieza  WHERE color = 'VERDE';

/* Nombres de todas las piezas
*/
SELECT DISTINCT nombPieza FROM Pieza;

/* Nombres de todas las piezas que empiezan por 'T'
*/
SELECT DISTINCT nombPieza FROM Pieza WHERE nombPieza LIKE 'T%';


/* Piezas que son 'TUERCA' o 'TORNILLO'
*/
SELECT DISTINCT * FROM Pieza WHERE nombPieza = 'TUERCA' OR nombPieza = 'TORNILLO';

/* Piezas que son 'TUERCA' o 'TORNILLO'
*/
SELECT DISTINCT *
  FROM  Pieza
  WHERE nombPieza = 'TUERCA'
UNION
SELECT DISTINCT  *
   FROM  Pieza
   WHERE nombPieza = 'TORNILLO';

/* Piezas que son 'TUERCA' o 'TORNILLO'
*/
SELECT DISTINCT clvPieza "id Pieza", nombPieza nombre, color
  FROM  Pieza
  WHERE nombPieza IN ('TUERCA', 'TORNILLO');

/* Piezas que son 'TUERCA' o 'TORNILLO'
*/
SELECT DISTINCT *
  FROM  Pieza
  WHERE nombPieza = ANY ('TUERCA', 'TORNILLO');

/* Nombre de los proveedores que suministran al menos una pieza de color 'VERDE'
*/
SELECT DISTINCT nombProv "proveedor", 'suministra piezas de color verde' "función"
FROM  Proveedor V, Suministrar S, Pieza P
WHERE V.clvProv = S.clvProv AND S.clvPieza = P.clvPieza AND P.color = 'VERDE';

/* Nombre de los proveedores que suministran piezas de color 'VERDE' o 'ROJO'
*/
SELECT DISTINCT nombProv "proveedor", 'suministra piezas de color verde' "función"
FROM  Proveedor V, Suministrar S, Pieza P
WHERE V.clvProv = S.clvProv AND S.clvPieza = P.clvPieza AND P.color = 'VERDE'
UNION
SELECT DISTINCT nombProv "proveedor", 'suministra piezas de color rojo' "función"
FROM  Proveedor V, Suministrar S, Pieza P
WHERE V.clvProv = S.clvProv AND S.clvPieza = P.clvPieza AND P.color = 'ROJO';

/* Piezas de las que no se conoce el color
*/
SELECT * FROM Pieza WHERE  color IS NULL;

/* Piezas disponibles en varios colores
*/
SELECT DISTINCT  P1.clvPieza, P1.nombPieza, P1.color
  FROM  Pieza P1, Pieza P2
  WHERE   P1.nombPieza = P2.nombPieza AND P1.color <> P2. color;
```

```
/* Proveedores que suministran 'TUERCAS' y 'TORNILLOS'
*/
SELECT DISTINCT S1.clvProv
   FROM  Suministrar S1, Pieza P1, Suministrar S2, Pieza P2
   WHERE S1.clvPieza = P1.clvPieza AND P1.nombPieza = 'TUERCA'
     AND S2.clvPieza = P2.clvPieza AND P2.nombPieza = 'TORNILLO'
     AND S1.clvProv = S2. clvProv;

/* Proveedores que suministran 'TUERCAS' y 'TORNILLOS'
*/
SELECT DISTINCT clvProv
   FROM  Suministrar S, Pieza P
   WHERE S.clvPieza = P.clvPieza AND P.nombPieza = 'TUERCA'
INTERSECT
SELECT DISTINCT clvProv
   FROM  Suministrar S, Pieza P
   WHERE S.clvPieza = P.clvPieza AND P.nombPieza = 'TORNILLO';

/* Clave y nombre de los proveedores que no suministran piezas
*/
CREATE VIEW noSuministran AS
  SELECT clvProv FROM  Proveedor
MINUS
  SELECT clvProv FROM  Suministrar;

SELECT  *
FROM  Proveedor V, noSuministran N
WHERE V.clvProv = N.clvProv;

DROP VIEW noSuministran;

/* Clave y nombre de los proveedores que no suministran piezas
*/
SELECT *
FROM  Proveedor
WHERE clvProv NOT IN (SELECT clvProv FROM Suministrar);

/* Clave y nombre de los proveedores que no suministran piezas
*/
SELECT *
FROM  Proveedor V
WHERE NOT EXISTS (SELECT * FROM Suministrar WHERE clvProv = V.clvProv);

/* Clave y nombre de los proveedores que no suministran tuercas verdes
*/
SELECT *
FROM  Proveedor V
WHERE NOT EXISTS (SELECT * FROM Pieza P
                  WHERE (nombPieza,color) IN (('TUERCA', 'VERDE'))
                    AND EXISTS(SELECT * FROM Suministrar
                               WHERE clvProv = V.clvProv AND clvPieza = P.clvPieza)
                 );

/* Clave y nombre de los proveedores que no suministran tuercas verdes
*/
SELECT *
FROM  Proveedor V
WHERE clvProv IN (SELECT clvProv FROM Proveedor
                  MINUS
                  SELECT clvProv FROM Suministrar S, Pieza P
                  WHERE S.clvPieza = P.clvPieza
                    AND P.nombPieza = 'TUERCA' AND P.color = 'VERDE');

/* Clave y nombre de los proveedores que suministran tuercas de todos los tipos
*/
SELECT *
FROM  Proveedor V
WHERE NOT EXISTS (SELECT * FROM Pieza P
                  WHERE P.nombPieza = 'TUERCA'
                    AND NOT EXISTS (SELECT * FROM Suministrar
                                    WHERE clvProv = V.clvProv AND clvPieza = P.clvPieza) );
```

```
/* Clave y nombre de los proveedores que suministran tuercas de todos los tipos
*/
CREATE VIEW NoSumTuercas AS
SELECT clvProv, Pieza.clvPieza
 FROM  Proveedor, Pieza
 WHERE nombPieza = 'TUERCA'
MINUS
SELECT clvProv, P.clvPieza
 FROM  Suministrar S, Pieza P
 WHERE S.clvPieza = P.clvPieza AND P.nombPieza = 'TUERCA';

SELECT clvProv, nombProv FROM Proveedor
MINUS
SELECT P.clvProv, nombProv FROM Proveedor P, NoSumTuercas N WHERE P.clvProv = N.clvProv;

DROP VIEW NoSumTuercas;


/* Clave y nombre de los proveedores, ordenados por nombre
*/
SELECT clvProv, nombProv FROM Proveedor ORDER BY nombProv;

/* Clave y nombre de los proveedores, por orden inverso de nombre
*/
SELECT * FROM Proveedor ORDER BY 2 DESC;

/* total de piezas existentes
*/
SELECT count(DISTINCT clvPieza) "tot. Piezas" FROM Pieza;

/* total de piezas distintas existentes
*/
SELECT count(DISTINCT nombPieza) "tipos de Piezas" FROM Pieza;

/* total de proveedores
*/
SELECT count(*) "tot. proveedores" FROM   Proveedor;

/* mayor valor de clvPieza
*/
SELECT max(clvPieza) "max. clvPieza" FROM   Pieza;

/* clave y nombre de las piezas, y el último dígito de la clave
*/
SELECT clvPieza, mod(clvPieza, 10) "term.", nombPieza "nombre"
 FROM   Pieza;

/* clave y nombre de las piezas con clave impar, y el último dígito de la clave
*/
SELECT clvPieza, mod(clvPieza, 10) "term.", nombPieza "nombre"
 FROM   Pieza
 WHERE  mod(clvPieza,2) = 1;

/* nombres de las piezas, y cantidad de cada tipo
*/
SELECT nombPieza "tipo", count(clvPieza) "total"
 FROM   Pieza
 GROUP BY nombPieza;

/* mostrar los valores NULL como 'desconocido' (SQLPLUS)
*/
SET NULL desconocido

/* colores de las piezas y número de piezas de cada color
*/
SELECT color, count(*) "total", count(color) "colores"
 FROM   Pieza
 GROUP BY color;
```

```
/* número medio de piezas de cada color (incluyendo desconocidos)
*/
SELECT AVG(count(*)) "piezas/color (incluso nulo)"
 FROM   Pieza
 GROUP BY color;

/* número medio de piezas de cada color (sólo los conocidos)
*/
SELECT AVG(count(color)) "piezas/color (conocido)"
 FROM   Pieza
 GROUP BY color
 HAVING color IS NOT NULL;

/* nombre (y cantidad) de los tipos de piezas de las que hay mas de 1,
   ordenadas por nombre
*/
SELECT nombPieza "tipo", count(clvPieza) "total"
 FROM   Pieza
 GROUP BY nombPieza
 HAVING count(clvPieza) > 1
 ORDER BY nombPieza ASC;

/* colores de las piezas y número de piezas de cada color, pero sólo de
   los colores con menos piezas que la media, ordenados alfabeticamente
*/
SELECT color, count(*) "total"
 FROM   Pieza
 GROUP BY color
 HAVING count(*) < (SELECT AVG(count(*)) FROM Pieza GROUP BY color)
 ORDER BY color;

/* mostrar los valores NULL como '' (SQLPLUS)
*/
SET NULL ''

/* especificar los formatos de algunas columnas (SQLPLUS)
*/
column nombre                                   format a20;
column Pieza                                    format a16;
column color                                    format a12;

/* indicar que si no cabe en la columna, trunque el valor (SQLPLUS)
*/
SET WRAP OFF

/* clave y nombre de los proveedores, junto con las piezas que suministran
*/
SELECT V.clvProv "id.", nombProv "nombre", P.clvPieza "REF.", nombPieza "Pieza", color
FROM   Proveedor V, Suministrar S, Pieza P
WHERE  V.clvProv = S.clvProv (+) AND S.clvPieza = P.clvPieza (+);

/* especificar los formatos de las columnas (SQLPLUS)
*/
column nombPieza         heading "Pieza"         format a16;
column nombProv          heading "Vendedor"      format a16;
column clvProv           heading "ID"            format 999;
column clvPieza          heading "REF."          format 999;

/* clave y nombre de los proveedores, junto con las piezas que suministran
*/
SELECT V.clvProv, nombProv, P.clvPieza, nombPieza, color
FROM   Proveedor V, Suministrar S, Pieza P
WHERE  V.clvProv = S.clvProv AND S.clvPieza = P.clvPieza
UNION
SELECT clvProv, nombProv, TO_NUMBER(NULL), NULL, NULL
FROM   Proveedor
WHERE  clvProv NOT IN (SELECT clvProv FROM Suministrar);

/* añadir las piezas:  'tuerca amarilla', de clave 99, y 'martillo', de clave 98
*/
INSERT INTO Pieza VALUES (99, 'TUERCA',   'AMARILLO');
INSERT INTO Pieza VALUES (98, 'MARTILLO', '');

/* añadir como piezas la lista de proveedores (aunque no tenga sentido)
```

```
*/
INSERT INTO Pieza SELECT clvprov, nombprov, TO_CHAR(NULL) FROM Proveedor;
SELECT * FROM Pieza;


/* eliminar todas las tuercas de color amarillo. Como se ha especificado borrado
   en cascada, no es necesario eliminar previamente los suministros de las piezas
   a eliminar para no violar la integridad referencial.
DELETE  FROM Suministrar
        WHERE clvPieza IN (SELECT clvPieza FROM Pieza
                           WHERE nombPieza = 'TUERCA' AND color = 'AMARILLO');
*/

DELETE  FROM Pieza WHERE  nombPieza = 'TUERCA' AND color = 'AMARILLO';

/* eliminar todos los suministros de piezas
*/
DELETE  FROM Suministrar;

/* eliminar todas los tipos de piezas de las que hay 1 o menos
*/
DELETE FROM Pieza P
  WHERE 2 > (SELECT count(*) FROM Pieza  WHERE nombPieza = P.nombPieza);

/* cambiar el color rojo por rojizo
*/
UPDATE Pieza SET color = 'ROJIZO'  WHERE  color = 'ROJO';

/* poner a la pieza 91 el color de la pieza 95
*/
UPDATE Pieza SET color = (SELECT color FROM Pieza WHERE clvPieza = 95)
  WHERE clvPieza=91;

/* añadir las piezas:  'tuerca amarilla', de clave 99, y 'martillo', de clave 98
*/
INSERT INTO Pieza VALUES (99, 'TUERCA',   'AMARILLO');
INSERT INTO Pieza VALUES (98, 'MARTILLO', '');

/* asignar 'desconocido' al color de las piezas con color no existente
*/
UPDATE Pieza SET color = 'desconocido'  WHERE  color IS NULL;

SELECT * FROM Pieza;
```

salida.txt

```
SQL> @pregBD_Piezas.sql
SQL> /* @pregBD_Piezas.sql */
SQL> SET ECHO ON
SQL>
SQL> /* Piezas de color 'verde'
DOC>*/
SQL> SELECT * FROM Pieza  WHERE color = 'VERDE';

REF. Pieza            COLOR
---- ---------------- ------------
  92 TUERCA           VERDE

SQL>
SQL> /* Nombres de todas las piezas
DOC>*/
SQL> SELECT DISTINCT nombPieza FROM Pieza;

Pieza
----------------
PALANCA
TORNILLO
TUBO
TUERCA

SQL>
SQL> /* Nombres de todas las piezas que empiezan por 'T'
DOC>*/
SQL> SELECT DISTINCT nombPieza FROM Pieza WHERE nombPieza LIKE 'T%';

Pieza
----------------
TORNILLO
TUBO
TUERCA

SQL>
SQL>
SQL> /* Piezas que son 'TUERCA' o 'TORNILLO'
DOC>*/
SQL> SELECT DISTINCT * FROM Pieza WHERE nombPieza = 'TUERCA' OR nombPieza = 'TORNILLO';

REF. Pieza            COLOR
---- ---------------- ------------
  91 TUERCA           ROJO
  92 TUERCA           VERDE
  93 TORNILLO         AZUL
  94 TORNILLO         ROJO
  95 TUERCA           AZUL

SQL>
SQL> /* Piezas que son 'TUERCA' o 'TORNILLO'
DOC>*/
SQL> SELECT DISTINCT *
  2    FROM  Pieza
  3    WHERE nombPieza = 'TUERCA'
  4  UNION
  5  SELECT DISTINCT  *
  6    FROM  Pieza
  7    WHERE nombPieza = 'TORNILLO';

REF. Pieza            COLOR
---- ---------------- ------------
  91 TUERCA           ROJO
  92 TUERCA           VERDE
  93 TORNILLO         AZUL
  94 TORNILLO         ROJO
  95 TUERCA           AZUL
```

```
SQL>
SQL> /* Piezas que son 'TUERCA' o 'TORNILLO'
DOC>*/
SQL> SELECT DISTINCT clvPieza "id Pieza", nombPieza nombre, color
  2    FROM  Pieza
  3    WHERE nombPieza IN ('TUERCA', 'TORNILLO');

  id Pieza nombre               COLOR
---------- -------------------- ------------
        91 TUERCA               ROJO
        92 TUERCA               VERDE
        93 TORNILLO             AZUL
        94 TORNILLO             ROJO
        95 TUERCA               AZUL

SQL>
SQL> /* Piezas que son 'TUERCA' o 'TORNILLO'
DOC>*/
SQL> SELECT DISTINCT *
  2    FROM  Pieza
  3    WHERE nombPieza =ANY ('TUERCA', 'TORNILLO');

REF. Pieza            COLOR
---- ---------------- ------------
  91 TUERCA           ROJO
  92 TUERCA           VERDE
  93 TORNILLO         AZUL
  94 TORNILLO         ROJO
  95 TUERCA           AZUL

SQL>
SQL> /* Nombre de los proveedores que suministran al menos una pieza de color 'VERDE'
DOC>*/
SQL> SELECT DISTINCT nombProv "proveedor", 'suministra piezas de color verde' "función"
  2  FROM  Proveedor V, Suministrar S, Pieza P
  3  WHERE V.clvProv = S.clvProv AND S.clvPieza = P.clvPieza AND P.color = 'VERDE'
  4
SQL> /* Nombre de los proveedores que suministran piezas de color 'VERDE' o 'ROJO'
DOC>*/
SQL> SELECT DISTINCT nombProv "proveedor", 'suministra piezas de color verde' "función"
  2  FROM  Proveedor V, Suministrar S, Pieza P
  3  WHERE V.clvProv = S.clvProv AND S.clvPieza = P.clvPieza AND P.color = 'VERDE'
  4  UNION
  5  SELECT DISTINCT nombProv "proveedor", 'suministra piezas de color rojo' "función"
  6  FROM  Proveedor V, Suministrar S, Pieza P
  7  WHERE V.clvProv = S.clvProv AND S.clvPieza = P.clvPieza AND P.color = 'ROJO';

proveedor                      función
------------------------------ ------------------------------
LOPEZ                          suministra piezas de color rojo
LOPEZ                          suministra piezas de color verde
PEREZ                          suministra piezas de color rojo
PEREZ                          suministra piezas de color verde

SQL>
SQL> /* Piezas de las que no se conoce el color
DOC>*/
SQL> SELECT * FROM Pieza WHEREcolor IS NULL;

REF. Pieza            COLOR
---- ---------------- ------------
  96 PALANCA

SQL>
SQL> /* Piezas disponibles en varios colores
DOC>*/
SQL> SELECT DISTINCT  P1.clvPieza, P1.nombPieza, P1.color
  2    FROM  Pieza P1, Pieza P2
  3    WHERE P1.nombPieza = P2.nombPieza AND P1.color <> P2. color;
```

```
REF. Pieza           COLOR
---- --------------- ------------
  93 TORNILLO        AZUL
  94 TORNILLO        ROJO
  91 TUERCA          ROJO
  92 TUERCA          VERDE
  95 TUERCA          AZUL

SQL>
SQL> /* Proveedores que suministran 'TUERCAS' y 'TORNILLOS'
DOC>*/
SQL> SELECT DISTINCT S1.clvProv
  2    FROM  Suministrar S1, Pieza P1, Suministrar S2, Pieza P2
  3    WHERE S1.clvPieza = P1.clvPieza AND P1.nombPieza = 'TUERCA'
  4      AND S2.clvPieza = P2.clvPieza AND P2.nombPieza = 'TORNILLO'
  5      AND S1.clvProv = S2. clvProv;

  ID
----
   2

SQL>
SQL> /* Proveedores que suministran 'TUERCAS' y 'TORNILLOS'
DOC>*/
SQL> SELECT DISTINCT clvProv
  2    FROM  Suministrar S, Pieza P
  3    WHERE S.clvPieza = P.clvPieza AND P.nombPieza = 'TUERCA'
  4  INTERSECT
  5  SELECT DISTINCT clvProv
  6    FROM  Suministrar S, Pieza P
  7    WHERE S.clvPieza = P.clvPieza AND P.nombPieza = 'TORNILLO';

  ID
----
   2

SQL>
SQL> /* Clave y nombre de los proveedores que no suministran piezas
DOC>*/
SQL> CREATE VIEW noSuministran AS
  2    SELECT clvProv FROM  Proveedor
  3  MINUS
  4    SELECT clvProv FROM  Suministrar;

View created.

SQL>
SQL> SELECT  *
  2  FROM  Proveedor V, noSuministran N
  3  WHERE V.clvProv = N.clvProv;

  ID Vendedor           ID
---- ---------------- ----
   3 MARTINEZ            3

SQL>
SQL> DROP VIEW noSuministran;

View dropped.

SQL>
SQL> /* Clave y nombre de los proveedores que no suministran piezas
DOC>*/
SQL> SELECT *
  2  FROM  Proveedor
  3  WHERE clvProv NOT IN (SELECT clvProv FROM Suministrar);

  ID Vendedor
---- ----------------
   3 MARTINEZ
```

```
SQL>
SQL> /* Clave y nombre de los proveedores que no suministran piezas
DOC>*/
SQL> SELECT *
  2  FROM  Proveedor V
  3  WHERE NOT EXISTS (SELECT * FROM Suministrar WHERE clvProv = V.clvProv);

  ID Vendedor
---- ----------------
   3 MARTINEZ

SQL>
SQL> /* Clave y nombre de los proveedores que no suministran tuercas verdes
DOC>*/
SQL> SELECT *
  2  FROM  Proveedor V
  3  WHERE NOT EXISTS (SELECT * FROM Pieza P
  4              WHERE (nombPieza,color) IN (('TUERCA', 'VERDE'))
  5            AND EXISTS(SELECT * FROM Suministrar
  6                  WHERE clvProv = V.clvProv AND clvPieza = P.clvPieza)
  7             );

  ID Vendedor
---- ----------------
   3 MARTINEZ

SQL>
SQL> /* Clave y nombre de los proveedores que no suministran tuercas verdes
DOC>*/
SQL> SELECT *
  2  FROM  Proveedor V
  3  WHERE clvProv IN (SELECT clvProv FROM Proveedor
  4              MINUS
  5              SELECT clvProv FROM Suministrar S, Pieza P
  6              WHERE S.clvPieza = P.clvPieza
  7            AND P.nombPieza = 'TUERCA' AND P.color = 'VERDE');

  ID Vendedor
---- ----------------
   3 MARTINEZ

SQL>
SQL>
SQL> /* Clave y nombre de los proveedores que suministran tuercas de todos los tipos
DOC>*/
SQL> SELECT *
  2  FROM  Proveedor V
  3  WHERE NOT EXISTS (SELECT * FROM Pieza P
  4              WHERE P.nombPieza = 'TUERCA'
  5            AND NOT EXISTS (SELECT * FROM Suministrar
  6                  WHERE clvProv = V.clvProv AND clvPieza = P.clvPieza) );

  ID Vendedor
---- ----------------
   1 PEREZ

SQL>
SQL>
SQL> /* Clave y nombre de los proveedores que suministran tuercas de todos los tipos
DOC>*/
SQL> CREATE VIEW NoSumTuercas AS
  2  SELECT clvProv, Pieza.clvPieza
  3   FROM  Proveedor, Pieza
  4   WHERE nombPieza = 'TUERCA'
  5  MINUS
  6  SELECT clvProv, P.clvPieza
  7   FROM  Suministrar S, Pieza P
  8   WHERE S.clvPieza = P.clvPieza AND P.nombPieza = 'TUERCA';

View created.
```

```
SQL>
SQL> SELECT clvProv, nombProv FROM Proveedor
  2  MINUS
  3  SELECT P.clvProv, nombProv FROM Proveedor P, NoSumTuercas N WHERE P.clvProv = N.clvProv;

  ID Vendedor
---- ----------------
   1 PEREZ

SQL>
SQL> DROP VIEW NoSumTuercas;

View dropped.

SQL>
SQL>
SQL> /* Clave y nombre de los proveedores, ordenados por nombre
DOC>*/
SQL> SELECT clvProv, nombProv FROM Proveedor ORDER BY nombProv;

  ID Vendedor
---- ----------------
   2 LOPEZ
   3 MARTINEZ
   1 PEREZ

SQL>
SQL> /* Clave y nombre de los proveedores, por orden inverso de nombre
DOC>*/
SQL> SELECT * FROM Proveedor ORDER BY 2 DESC;

  ID Vendedor
---- ----------------
   1 PEREZ
   3 MARTINEZ
   2 LOPEZ

SQL>
SQL> /* total de piezas existentes
DOC>*/
SQL> SELECT count(DISTINCT clvPieza) "tot. Piezas" FROM Pieza;

tot. Piezas
-----------
          7

SQL>
SQL> /* total de piezas distintas existentes
DOC>*/
SQL> SELECT count(DISTINCT nombPieza) "tipos de Piezas" FROM Pieza;

tipos de Piezas
---------------
              4

SQL>
SQL> /* total de proveedores
DOC>*/
SQL> SELECT count(*) "tot. proveedores" FROM   Proveedor;

tot. proveedores
----------------
               3

SQL>
SQL> /* mayor valor de clvPieza
DOC>*/
SQL> SELECT max(clvPieza) "max. clvPieza" FROM  Pieza;

max. clvPieza
-------------
           97
SQL>
SQL> /* nombres de las piezas, y cantidad de cada tipo
DOC>*/
```

```
SQL> SELECT nombPieza "tipo", count(clvPieza) "total"
  2   FROM   Pieza
  3   GROUP BY nombPieza;

tipo                                total
------------------------------ ----------
PALANCA                                 1
TORNILLO                                2
TUBO                                    1
TUERCA                                  3

SQL>
SQL> /* mostrar los valores NULL como 'desconocido' (SQLPLUS)
DOC>*/
SQL> SET NULL desconocido
SQL>
SQL> /* colores de las piezas y número de piezas de cada color
DOC>*/
SQL> SELECT color, count(*) "total", count(color) "colores"
  2   FROM   Pieza
  3   GROUP BY color;

COLOR             total    colores
------------ ---------- ----------
AZUL                  2          2
GRIS                  1          1
ROJO                  2          2
VERDE                 1          1
desconocido           1          0

SQL>
SQL> /* número medio de piezas de cada color (incluyendo desconocidos)
DOC>*/
SQL> SELECT AVG(count(*)) "piezas/color (incluso nulo)"
  2   FROM   Pieza
  3   GROUP BY color;

piezas/color (incluso nulo)
---------------------------
                        1.4

SQL>
SQL> /* número medio de piezas de cada color (sólo los conocidos)
DOC>*/
SQL> SELECT AVG(count(color)) "piezas/color (conocido)"
  2   FROM   Pieza
  3   GROUP BY color
  4   HAVING color IS NOT NULL;

piezas/color (conocido)
-----------------------
                    1.5

SQL>
SQL> /* nombre (y cantidad) de los tipos de piezas de las que hay mas de 1,
DOC>   ordenadas por nombre
DOC>*/
SQL> SELECT nombPieza "tipo", count(clvPieza) "total"
  2   FROM   Pieza
  3   GROUP BY nombPieza
  4   HAVING count(clvPieza) > 1
  5   ORDER BY nombPieza ASC;

tipo                                total
------------------------------ ----------
TORNILLO                                2
TUERCA                                  3

SQL>
```

```
SQL> /* colores de las piezas y número de piezas de cada color, pero sólo de
DOC>   los colores con menos piezas que la media, ordenados alfabeticamente
DOC>*/
SQL> SELECT color, count(*) "total"
  2   FROM   Pieza
  3   GROUP BY color
  4   HAVING count(*) < (SELECT AVG(count(*)) FROM Pieza GROUP BY color)
  5   ORDER BY color;

COLOR            total
------------ ----------
GRIS                 1
VERDE                1
desconocido          1

SQL> /* mostrar los valores NULL como '' (SQLPLUS)
DOC>*/
SQL> SET NULL ''
SQL>
SQL> /* especificar los formatos de algunas columnas (SQLPLUS)
DOC>*/
SQL> column nombre      format a20;
SQL> column Pieza       format a16;
SQL> column color       format a12;
SQL>
SQL> /* indicar que si no cabe en la columna, trunque el valor (SQLPLUS)
DOC>*/
SQL> SET WRAP OFF
SQL>
SQL> /* clave y nombre de los proveedores, junto con las piezas que suministran
DOC>*/
SQL> SELECT V.clvProv "id.", nombProv "nombre", P.clvPieza "ref.", nombPieza "Pieza", color
  2  FROM   Proveedor V, Suministrar S, Pieza P
  3  WHERE  V.clvProv = S.clvProv (+) AND S.clvPieza = P.clvPieza (+);

       id. nombre                    REF. Pieza           COLOR
---------- -------------------- ---------- --------------- ------------
         1 PEREZ                        91 TUERCA          ROJO
         1 PEREZ                        92 TUERCA          VERDE
         1 PEREZ                        95 TUERCA          AZUL
         2 LOPEZ                        92 TUERCA          VERDE
         2 LOPEZ                        93 TORNILLO        AZUL
         2 LOPEZ                        94 TORNILLO        ROJO
         3 MARTINEZ

7 rows selected.

SQL>
SQL> /* especificar los formatos de las columnas (SQLPLUS)
DOC>*/
SQL> column nombPieza     heading "Pieza"         format a16;
SQL> column nombProv      heading "Vendedor"      format a16;
SQL> column clvProv       heading "ID"            format 999;
SQL> column clvPieza      heading "ref."          format 999;
SQL>
SQL> /* clave y nombre de los proveedores, junto con las piezas que suministran
DOC>*/
SQL> SELECT V.clvProv, nombProv, P.clvPieza, nombPieza, color
  2  FROM   Proveedor V, Suministrar S, Pieza P
  3  WHERE  V.clvProv = S.clvProv AND S.clvPieza = P.clvPieza
  4  UNION
  5  SELECT clvProv, nombProv, TO_NUMBER(NULL), NULL, NULL
  6  FROM   Proveedor
  7  WHERE  clvProv NOT IN (SELECT clvProv FROM Suministrar);

  ID Vendedor         REF. Pieza           COLOR
---- ---------------- ---- --------------- ------------
   1 PEREZ              91 TUERCA          ROJO
   1 PEREZ              92 TUERCA          VERDE
   1 PEREZ              95 TUERCA          AZUL
   2 LOPEZ              92 TUERCA          VERDE
   2 LOPEZ              93 TORNILLO        AZUL
   2 LOPEZ              94 TORNILLO        ROJO
   3 MARTINEZ

7 rows selected.

SQL> spool off
```

PL_SQL_1.sql

```
/* @PL_SQL_1.sql */
SET ECHO ON
SET LINESIZE 132
SET PAGESIZE 80
SET WRAP OFF

/* seleccionar los proveedores que suministran mas de una pieza */

/* creación de una tabla temporal para resultado (refProv, totPiezas) */
CREATE TABLE temp (refProv number, totPiezas number);

DECLARE
  laRefProv number;
  numPiezas number;

  CURSOR selProv IS SELECT distinct clvProv FROM Suministrar;

BEGIN
  OPEN selProv;
  LOOP
    FETCH selProv INTO laRefProv;

    EXIT WHEN selProv%NOTFOUND;

    SELECT count(distinct clvPieza) INTO numPiezas
    FROM  Suministrar
    WHERE clvProv = laRefProv;

    IF numPiezas > 1 THEN
      INSERT INTO temp VALUES (laRefProv, numPiezas);
    END IF;
  END LOOP;
  COMMIT;
END;
.
/
SELECT * FROM temp;

DROP TABLE temp;
```

PL_SQL_2.sql

```
/* @PL_SQL_2.sql */
SET ECHO ON
SET LINESIZE 132
SET PAGESIZE 80
SET WRAP OFF

/* contar los proveedores que suministran mas de una pieza */

VARIABLE contador number;

/* especificación del bloque PL/SQL
*/
DECLARE
  laRefProv number;
  numPiezas number;

  CURSOR selProv IS SELECT distinct clvProv FROM Suministrar;

BEGIN
  :contador := 0;
  OPEN selProv;
  LOOP
    FETCH selProv INTO laRefProv;

    EXIT WHEN selProv%NOTFOUND;

    SELECT count(distinct clvPieza) INTO numPiezas
    FROM  Suministrar
    WHERE clvProv = laRefProv;

    IF numPiezas > 1 THEN
      :contador := :contador + 1;
    END IF;
  END LOOP;
  COMMIT;
END;
/

PRINT contador
```

creaBD_UZ.sql

```
/* @creaBD_UZ.sql */
SET ECHO ON

/* Crear las tablas y añadir algunas restricciones, de varias maneras */

CREATE TABLE Departamento (
  clvDpto    number(3)   CONSTRAINT PK_clvDpto   PRIMARY KEY,
  codDpto    char(10)    CONSTRAINT UN_codDpto   UNIQUE,
  nombDpto   char(20)    CONSTRAINT NN_nombDpto  NOT NULL);

CREATE TABLE AreaConoc (
  clvArea    number(3)   CONSTRAINT PK_clvArea   PRIMARY KEY,
  codArea    char(10)    CONSTRAINT UN_codArea   UNIQUE,
  nombArea   char(30)    CONSTRAINT NN_nombArea  NOT NULL,
  clvDpto    number(3),
  CONSTRAINT FK_clvDptoArea FOREIGN KEY (clvDpto) REFERENCES Departamento(clvDpto) );

CREATE TABLE Profesor (
  clvProf    number(3)   CONSTRAINT PK_clvProf   PRIMARY KEY,
  codProf    char(10)    CONSTRAINT UN_codProf   UNIQUE,
  nombProf   char(30)    CONSTRAINT NN_nombProf  NOT NULL,
  clvArea    number(3),
  CONSTRAINT FK_clvAreaProf FOREIGN KEY (clvArea) REFERENCES AreaConoc(clvArea) );

CREATE TABLE Asignatura (
  clvAsign   number(3)   CONSTRAINT PK_clvAsign  PRIMARY KEY,
  codAsign   char(10)    CONSTRAINT UN_codAsign  UNIQUE,
  nombAsign  char(30)    CONSTRAINT NN_nombAsign NOT NULL,
  tt_HT      number(3),
  tt_HP      number(3),
  clvArea    number(3),
  CONSTRAINT FK_clvAreaAsig FOREIGN KEY (clvArea) REFERENCES AreaConoc(clvArea) );

CREATE TABLE Titulacion (
  clvTitulo  number(3)   CONSTRAINT PK_clvTitulo  PRIMARY KEY,
  codTitulo  char(10)    CONSTRAINT UN_codTitulo  UNIQUE,
  nombTitulo char(20)    CONSTRAINT NN_nombTitulo NOT NULL);

CREATE TABLE ImparteAsign (
  clvProf    number(3),
  clvAsign   number(3),
  HT         number(4),
  HP         number(4),
  CONSTRAINT FK_clvProfImpAsg  FOREIGN KEY (clvProf)  REFERENCES Profesor(clvProf),
  CONSTRAINT FK_clvAsigImpAsg  FOREIGN KEY (clvAsign) REFERENCES Asignatura(clvAsign) );

CREATE TABLE AsignTitulo (
  clvAsign   number(3),
  clvTitulo  number(3),
  CONSTRAINT FK_clvAsgAsgTit   FOREIGN KEY (clvAsign)  REFERENCES Asignatura(clvAsign),
  CONSTRAINT FK_clvTitAsgTit   FOREIGN KEY (clvTitulo) REFERENCES Titulacion(clvTitulo) );

SELECT * FROM cat;

DESCRIBE Departamento;
DESCRIBE AreaConoc;
DESCRIBE Profesor;
DESCRIBE Asignatura;
DESCRIBE Titulacion;
DESCRIBE ImparteAsign;
DESCRIBE AsignTitulo;
```

infoBD_UZ.sql

```
/* @infoBD_UZ.sql */
SET ECHO ON

DESCRIBE Departamento;
DESCRIBE AreaConoc;
DESCRIBE Profesor;
DESCRIBE Asignatura;
DESCRIBE Titulacion;
DESCRIBE ImparteAsign;
DESCRIBE AsignTitulo;

SET LINESIZE 132

column OWNER             heading "dueño"         format a10;
column TABLE_NAME        heading "tabla"         format a14;
column CONSTRAINT_NAME   heading "namRestric"    format a16;
column SEARCH_CONDITION  heading "cond.busq."    format a24;
column R_OWNER           heading "dueñoRef"      format a10;
column R_CONSTRAINT_NAME heading "namRestricRef" format a16;

SELECT OWNER, TABLE_NAME, CONSTRAINT_NAME, SEARCH_CONDITION, R_OWNER, R_CONSTRAINT_NAME
 FROM user_constraints
 WHERE INITCAP(table_name) IN ('Departamento', 'Areaconoc', 'Profesor', 'Asignatura',
                               'Titulacion', 'Imparteasign', 'Asigntitulo');
```

elimBD_UZ.sql

```
/* @elimBD_UZ.sql */
SET ECHO ON

/* Eliminar las tablas y todas las restricciones ligadas, de tal modo que
   solo se borre una tabla cuando sus atributos no son referenciados
 */
ALTER TABLE AreaConoc    DROP CONSTRAINT FK_clvDptoArea;
ALTER TABLE Profesor     DROP CONSTRAINT FK_clvAreaProf;
ALTER TABLE Asignatura   DROP CONSTRAINT FK_clvAreaAsig;
ALTER TABLE ImparteAsign DROP CONSTRAINT FK_clvProfImpAsg;
ALTER TABLE ImparteAsign DROP CONSTRAINT FK_clvAsigImpAsg;
ALTER TABLE AsignTitulo  DROP CONSTRAINT FK_clvAsgAsgTit;
ALTER TABLE AsignTitulo  DROP CONSTRAINT FK_clvTitAsgTit;

DROP TABLE ImparteAsign;
DROP TABLE AsignTitulo;
DROP TABLE Asignatura;
DROP TABLE Profesor;
DROP TABLE AreaConoc;
DROP TABLE Departamento;
DROP TABLE Titulacion;

SELECT * FROM cat;
```

ponDat_UZ.sql

```
/* @ponDat_UZ.sql */
SET ECHO ON

DELETE FROM ImparteAsign;
DELETE FROM AsignTitulo;
DELETE FROM Asignatura;
DELETE FROM Profesor;
DELETE FROM AreaConoc;
DELETE FROM Titulacion;
DELETE FROM Departamento;

INSERT INTO Departamento VALUES (  1,     '284D',     'DIEI');
INSERT INTO Departamento VALUES (  2,     '285D', 'ING.MEC.');
INSERT INTO Departamento VALUES (  3,     '286D', 'MAT.APL.');

INSERT INTO AreaConoc    VALUES (  1,    'AC001',  'LENG. Y SIST. INFORM.',   1);
INSERT INTO AreaConoc    VALUES (  2,    'AC002',      'TECN. ELECTRONICA',   1);
INSERT INTO AreaConoc    VALUES (  3,    'AC003',         'ING. ELECTRICA',   1);
INSERT INTO AreaConoc    VALUES (  4,    'AC004',    'ING. DE TRANSPORTES',   2);
INSERT INTO AreaConoc    VALUES (  5,    'AC005',          'EXPR. GRAFICA',   2);
INSERT INTO AreaConoc    VALUES (  6,    'AC006',        'MATEM. APLICADA',   3);
INSERT INTO AreaConoc    VALUES (  7,    'AC007',       'TECN. MATEMATICA',   3);

INSERT INTO Profesor     VALUES (  1,   'PRF001',     'S. VELILLA',   1);
INSERT INTO Profesor     VALUES (  2,   'PRF002',      'J. CAMPOS',   1);
INSERT INTO Profesor     VALUES (  3,   'PRF003',     'J. NAVARRO',   2);
INSERT INTO Profesor     VALUES (  4,   'PRF004',      'A. MTNEZ.',   2);
INSERT INTO Profesor     VALUES (  5,   'PRF005',     'I. RAMIREZ',   3);
INSERT INTO Profesor     VALUES (  6,   'PRF006',       'E. CAROD',   3);
INSERT INTO Profesor     VALUES (  7,   'PRF007',    'A. MIRAVETE',   4);
INSERT INTO Profesor     VALUES (  8,   'PRF008',       'J. ORTAS',   4);
INSERT INTO Profesor     VALUES (  9,   'PRF009',   'E. ZUBIAURRE',   5);
INSERT INTO Profesor     VALUES ( 10,   'PRF010',      'G. TRAVER',   5);
INSERT INTO Profesor     VALUES ( 11,   'PRF011',    'V. CAMARENA',   6);
INSERT INTO Profesor     VALUES ( 12,   'PRF012',      'C. BUDRIA',   6);
INSERT INTO Profesor     VALUES ( 13,   'PRF013',   'J.M. CORREAS',   7);
INSERT INTO Profesor     VALUES ( 14,   'PRF014',      'M. AGUADO',   7);

INSERT INTO Asignatura   VALUES (  1,   'ASG001',         'INFORMATICA',  90,  30, 1);
INSERT INTO Asignatura   VALUES (  2,   'ASG002',     'FICH. y B.DATOS',  45,  20, 1);
INSERT INTO Asignatura   VALUES (  3,   'ASG003', 'ELECTRONICA DIGITAL',  60,  35, 2);
INSERT INTO Asignatura   VALUES (  4,   'ASG004',    'MICROELECTRONICA',  20,  10, 2);
INSERT INTO Asignatura   VALUES (  5,   'ASG005',       'ELECTROTECNIA',  30,  30, 3);
INSERT INTO Asignatura   VALUES (  6,   'ASG006',      'LINEAS Y REDES',  25,  15, 3);
INSERT INTO Asignatura   VALUES (  7,   'ASG007',       'FERROCARRILES',  10,  10, 4);
INSERT INTO Asignatura   VALUES (  8,   'ASG008',    'TRACCION ELECTR.',  20,  20, 4);
INSERT INTO Asignatura   VALUES (  9,   'ASG009',            'DIBUJO I',  40,  20, 5);
INSERT INTO Asignatura   VALUES ( 10,   'ASG010',   'GEOM. DESCRIPTIVA',  40,  10, 5);
INSERT INTO Asignatura   VALUES ( 11,   'ASG011',             'ALGEBRA',  60,   0, 6);
INSERT INTO Asignatura   VALUES ( 12,   'ASG012',   'GEOM. DIFERENCIAL',  40,   0, 6);
INSERT INTO Asignatura   VALUES ( 13,   'ASG013',             'CALCULO',  20,  10, 7);
INSERT INTO Asignatura   VALUES ( 14,   'ASG014',    'CALCULO NUMERICO',  40,  20, 7);

INSERT INTO Titulacion   VALUES (  1,   'TIT001',       'ING. INFORMATICA');
INSERT INTO Titulacion   VALUES (  2,   'TIT002',    'ING. TELECOMUNICAC.');
INSERT INTO Titulacion   VALUES (  3,   'TIT003',        'ING. INDUSTRIAL');
INSERT INTO Titulacion   VALUES (  4,   'TIT004',   'ING. TECN. ELECTRON.');

INSERT INTO ImparteAsign VALUES (  1,  1,   3,  4);
INSERT INTO ImparteAsign VALUES (  1,  2,  23, 44);
INSERT INTO ImparteAsign VALUES ( 14, 14,  13, 0);

INSERT INTO AsignTitulo  VALUES (  1,  1);
INSERT INTO AsignTitulo  VALUES (  1,  2);
INSERT INTO AsignTitulo  VALUES (  1,  3);
INSERT INTO AsignTitulo  VALUES (  1,  4);
INSERT INTO AsignTitulo  VALUES (  2,  1);
INSERT INTO AsignTitulo  VALUES (  2,  2);
INSERT INTO AsignTitulo  VALUES (  2,  3);
INSERT INTO AsignTitulo  VALUES (  2,  4);
```

lstDat_UZ.sql

```
/* @lstDat_UZ.sql */
SET ECHO ON

SELECT COUNT(*) "nº Dptos" FROM Departamento;
SELECT * FROM Departamento;

SELECT COUNT(*) "nº aCon" FROM AreaConoc;
SELECT * FROM AreaConoc;

SELECT COUNT(*) "nº Prof" FROM Profesor;
SELECT * FROM Profesor;

SELECT COUNT(*) "nº Asign" FROM Asignatura;
SELECT * FROM Asignatura;

SELECT COUNT(*) "nº Titul" FROM Titulacion;
SELECT * FROM Titulacion;

SELECT COUNT(*) "nº ImpAsg" FROM ImparteAsign;
SELECT * FROM ImparteAsign;

SELECT COUNT(*) "nº AsgTit" FROM AsignTitulo;
SELECT * FROM AsignTitulo;
```

elimDatUZ.sql

```
/* @elimDat_UZ.sql */
SET ECHO ON

DELETE FROM ImparteAsign;
DELETE FROM AsignTitulo;
DELETE FROM Titulacion;
DELETE FROM Profesor;
DELETE FROM Asignatura;
DELETE FROM AreaConoc;
DELETE FROM Departamento;
```

pregSimple_UZ.sql

```
/* @pregSimple_UZ.sql */
SET ECHO ON

/* creación de una vista con total horas impartidas por profesor
 */
CREATE VIEW totHorasProf (clvProf, ttHT, ttHP) AS
SELECT clvProf, sum(HT), sum(HP)
FROM   ImparteAsign
GROUP BY clvProf;

/* listado de los profesores (clave y nombre), y el total horas impartidas
 */
SELECT Profesor.clvProf, nombProf, ttHT "h. Teoría", ttHP  "h. Prácticas"
FROM   Profesor, totHorasProf
WHERE  Profesor.clvProf = totHorasProf.clvProf;

/* eliminación de la vista creada
 */
DROP VIEW totHorasProf;
```

creaBanco.sql

```
/* @creaBanco.sql */

/* hacer eco de las sentencias */
SET ECHO ON

/* especificar líneas de hasta 132 caracteres */
SET LINESIZE 132

/* especificar tamaño de página 80 líneas */
SET PAGESIZE 80

/* truncar los valores de los atributos si "no caben" en la columna */
SET WRAP OFF

/* la Base de Datos propuesta, además de corresponder a un diseño no demasiado adecuado, es
   algo incompleta (faltan restricciones).  La razón básica de este hecho es que, además de
   servir de ejemplo de  implementación de algunas operaciones en  SQL, pueda ser utilizado
   como ejercicio de diseño.    Así pues, se recomienda  estudiar detenidamente el problema
   para mejorar el esquema relacional propuesto (hacer la transformación con sentencias SQL)
*/

CREATE TABLE empleado(
  numsegsocial number(16),
  nombre       char(25),
  categoria    char(20));

CREATE TABLE destino(
  numsegsocial number(16),
  ciudad       char(20),
  agencia      number(2));

CREATE TABLE cliente(
  nombre char(25) NOT NULL,
  DNI    number(8),
  calle  char(25),
  ciudad char(20));

CREATE TABLE cuenta(
  numero number(10),
  saldo  number(15));

CREATE TABLE sucursal(
  ciudad  char(20)  NOT NULL,
  agencia number(2) NOT NULL,
  pasivo  number (18));

CREATE TABLE apercuenta(
  DNI     number(8),
  numero  number(10),
  ciudad  char(20),
  agencia number(2));

CREATE TABLE prestamo(
  numprestamo number(6),
  importe     number(15),
  interes     number(6,2));

CREATE TABLE prestFormalizado(
  numero      number(10),
  numprestamo number(6));

CREATE TABLE inversion(
  numinversion number(6),
  importe      number(15),
  interes      number(6,2));

CREATE TABLE invFormalizada(
  numero       number(10),
  numinversion number(6));

@infoBanco.sql
```

infoBanco.sql

```
/* @infoBanco.sql */
/* hacer eco de las sentencias */
SET ECHO ON
/* especificar líneas de hasta 132 caracteres */
SET LINESIZE 132
/* especificar tamaño de página 80 líneas */
SET PAGESIZE 80
/* truncar los valores de los atributos si "no caben" en la columna */
SET WRAP OFF

SELECT * FROM cat;

DESCRIBE empleado;
DESCRIBE destino;
DESCRIBE prestamo;
DESCRIBE prestFormalizado;
DESCRIBE inversion;
DESCRIBE invFormalizada;
DESCRIBE cliente;
DESCRIBE cuenta;
DESCRIBE sucursal;
DESCRIBE apercuenta;
```

elimBanco.sql

```
/* @elimBanco.sql */
/* hacer eco de las sentencias */
SET ECHO ON
/* especificar líneas de hasta 132 caracteres */
SET LINESIZE 132
/* especificar tamaño de página 80 líneas */
SET PAGESIZE 80
/* truncar los valores de los atributos si "no caben" en la columna */
SET WRAP OFF

DROP VIEW  numAgenCliente;
DROP TABLE destino;
DROP TABLE empleado;
DROP TABLE prestFormalizado;
DROP TABLE prestamo;
DROP TABLE invFormalizada;
DROP TABLE inversion;
DROP TABLE apercuenta;
DROP TABLE cliente;
DROP TABLE cuenta;
DROP TABLE sucursal;

@infoBanco.sql
```

ponDatBanco.sql

```
/* @ponDatBanco.sql */

/* hacer eco de las sentencias */
SET ECHO ON

/* especificar líneas de hasta 132 caracteres */
SET LINESIZE 132

/* especificar tamaño de página 80 líneas */
SET PAGESIZE 80

/* truncar los valores de los atributos si "no caben" en la columna */
SET WRAP OFF

/* indicar que no muestre el resultado de la operación */
SET FEEDBACK OFF

INSERT INTO empleado VALUES (50000123,'Antonio Segura', 'Auxiliar');
INSERT INTO empleado VALUES (50004502,'Juan Gutierrez', 'Auxiliar');
INSERT INTO empleado VALUES (50000567,'Anselmo Rincon', 'Auxiliar');
INSERT INTO empleado VALUES (50001234,'Benjamin Cacho', 'Auxiliar');
INSERT INTO empleado VALUES (50002222,'Ines Castan', 'Oficial');
INSERT INTO empleado VALUES (50003222,'Maria Otin', 'Oficial');

INSERT INTO destino  VALUES (50000123,'Zaragoza', 4);
INSERT INTO destino  VALUES (50004502,'Zaragoza', 4);
INSERT INTO destino  VALUES (50000567,'Huesca', 1);
INSERT INTO destino  VALUES (50001234,'Teruel', 2);
INSERT INTO destino  VALUES (50002222,'Zaragoza', 2);
INSERT INTO destino  VALUES (50003222,'Zaragoza', 3);

INSERT INTO cuenta   VALUES (14187, 217456);
INSERT INTO cuenta   VALUES (16543, 4567);
INSERT INTO cuenta   VALUES (13087, 50000);
INSERT INTO cuenta   VALUES (18564, 107870);
INSERT INTO cuenta   VALUES (11456, 56890);
INSERT INTO cuenta   VALUES (10980, 47);
INSERT INTO cuenta   VALUES (15434, 987665);
INSERT INTO cuenta   VALUES (13345, 107456);
INSERT INTO cuenta   VALUES (16657, 21765);

INSERT INTO prestamo VALUES (82,50000, 17.12);
INSERT INTO prestamo VALUES (17,200000, 17.12);
INSERT INTO prestamo VALUES (45,100000, 17.12);
INSERT INTO prestamo VALUES (99,340000, 16.75);
INSERT INTO prestamo VALUES (40,100000, 16.75);
INSERT INTO prestamo VALUES (55,50000, 16.75);

INSERT INTO prestFormalizado VALUES (13087, 40);
INSERT INTO prestFormalizado VALUES (14187, 17);
INSERT INTO prestFormalizado VALUES (16543, 45);
INSERT INTO prestFormalizado VALUES (10980, 82);
INSERT INTO prestFormalizado VALUES (13087, 55);
INSERT INTO prestFormalizado VALUES (15434, 99);

INSERT INTO inversion VALUES (13, 1000000, 15.0);
INSERT INTO inversion VALUES (17, 250000, 14.0);
INSERT INTO inversion VALUES (25, 175000, 12.75);
INSERT INTO inversion VALUES (27, 975000, 13.75);

INSERT INTO invFormalizada VALUES (14187, 17);
INSERT INTO invFormalizada VALUES (13087, 25);
INSERT INTO invFormalizada VALUES (13345, 13);
INSERT INTO invFormalizada VALUES (10980, 27);

INSERT INTO sucursal VALUES ('Zaragoza', 1, 1347456789);
INSERT INTO sucursal VALUES ('Zaragoza', 2, 947456300);
INSERT INTO sucursal VALUES ('Zaragoza', 3, 1600567455);
INSERT INTO sucursal VALUES ('Zaragoza', 4, 1090890076);
INSERT INTO sucursal VALUES ('Zaragoza', 5, 890765444);
INSERT INTO sucursal VALUES ('Zaragoza', 6, 334567800);
INSERT INTO sucursal VALUES ('Huesca', 1, 567865444);
INSERT INTO sucursal VALUES ('Huesca', 2, 666400987);
INSERT INTO sucursal VALUES ('Teruel', 1, 432456789);
```

```
INSERT INTO sucursal VALUES ('Teruel', 2, 890876777);

INSERT INTO cliente  VALUES ('Susana Ramon', 12543122, 'La Paz 121', 'Zaragoza');
INSERT INTO cliente  VALUES ('Herminio Salgado', 11321009, 'Nueva 2', 'Teruel');
INSERT INTO cliente  VALUES ('Maria Abos', 18678999, 'Lorca 34', 'Zaragoza');
INSERT INTO cliente  VALUES ('Miguel Iglesias', 12123322, 'Constitucion 65', 'Huesca');
INSERT INTO cliente  VALUES ('Jose Pons', 17656777, 'Mayor 35', 'Zaragoza');
INSERT INTO cliente  VALUES ('Ana Ferrer', 11765789, 'Libertad 22', 'Teruel');
INSERT INTO cliente  VALUES ('Jose Lopez', 15444345, 'Mayor 17', 'Zaragoza');

INSERT INTO apercuenta VALUES (17656777, 10980, 'Zaragoza', 4);
INSERT INTO apercuenta VALUES (11765789, 11456, 'Zaragoza', 2);
INSERT INTO apercuenta VALUES (15444345, 14187, 'Zaragoza', 4);
INSERT INTO apercuenta VALUES (12123322, 14187, 'Zaragoza', 4);
INSERT INTO apercuenta VALUES (17656777, 13087, 'Huesca', 2);
INSERT INTO apercuenta VALUES (18678999, 16543, 'Teruel', 1);
INSERT INTO apercuenta VALUES (12543122, 16657, 'Zaragoza', 5);
INSERT INTO apercuenta VALUES (12543122, 16543, 'Zaragoza', 4);

COMMIT;

/* indicar que muestre el resultado de la operación */
SET FEEDBACK ON

@lstDatBanco
```

lstDatBanco.sql

```
/* @lstDatBanco.sql */

/* hacer eco de las sentencias */
SET ECHO ON

/* especificar líneas de hasta 132 caracteres */
SET LINESIZE 132

/* especificar tamaño de página 80 líneas */
SET PAGESIZE 80

/* truncar los valores de los atributos si "no caben" en la columna */
SET WRAP OFF

/* especifica formatos para las columnas */
column ciudad  format a10;
column calle   format a16;
column numero  format 999999 heading 'nº CC';
column dni     format 99999999;
column agencia format 999    heading 'Ag.';

/* especificación de formato de listado sin repetir valores iguales que en fila anterior */
break ON nombre nodup ON dni nodup ON calle nodup ON ciudad nodup;

SELECT * FROM empleado;
SELECT * FROM destino;
SELECT * FROM prestamo;
SELECT * FROM prestFormalizado;
SELECT * FROM inversion;
SELECT * FROM invFormalizada;
SELECT * FROM cliente;
SELECT * FROM cuenta;
SELECT * FROM sucursal;
SELECT * FROM apercuenta;

/* Eliminar las "rupturas" especificadas */
clear break;
```

elimDatBanco.sql

```
/* @elimDatBanco.sql */
/* hacer eco de las sentencias */
SET ECHO ON
/* especificar líneas de hasta 132 caracteres */
SET LINESIZE 132
/* especificar tamaño de página 80 líneas */
SET PAGESIZE 80
/* truncar los valores de los atributos si "no caben" en la columna */
SET WRAP OFF

DELETE FROM apercuenta;
DELETE FROM empleado;
DELETE FROM destino;
DELETE FROM prestamo;
DELETE FROM prestFormalizado;
DELETE FROM inversion;
DELETE FROM invFormalizada;
DELETE FROM cliente;
DELETE FROM cuenta;
DELETE FROM sucursal;
```

ej0_Banco.sql

```
/* @preg1Banco.sql */
/* hacer eco de las sentencias */
SET ECHO ON
/* especificar líneas de hasta 132 caracteres */
SET LINESIZE 132
/* especificar tamaño de página 80 líneas */
SET PAGESIZE 80
/* truncar los valores de los atributos si "no caben" en la columna */
SET WRAP OFF

column ciudad  format a10;
column calle   format a16;
column numero  format 999999 heading 'nº CC';
column dni     format 99999999;
column agencia format 999 heading 'Ag.';

break ON nombre nodup ON dni nodup ON calle nodup ON ciudad nodup;

SELECT * FROM empleado;
SELECT * FROM destino;
SELECT * FROM prestamo;
SELECT * FROM prestFormalizado;
SELECT * FROM inversion;
SELECT * FROM invFormalizada;
SELECT * FROM cliente;
SELECT * FROM cuenta;
SELECT * FROM sucursal;
SELECT * FROM apercuenta;

/* ejemplo de consulta sencilla: mostrar informacion de los clientes del banco (DNI, nombre,
   calle y ciudad) junto con las cuentas (numero, ciudad, agencia) de las que son titulares.
   Observese que sólo aparecerán los clientes que tienen abierta alguna cuenta
*/
SELECT cliente.dni, nombre, calle, cliente.ciudad, numero, agencia, apercuenta.ciudad
FROM   apercuenta, cliente
WHERE  apercuenta.dni = cliente.dni
ORDER BY cliente.dni;
```

ej1_Banco.sql

```
/* @ej1_OpBanco */
SET echo ON
```

/* listar las cuentas abiertas */

```
SELECT * FROM apercuenta;
```

/* nº de cuentas de cada sucursal */

```
SELECT ciudad,agencia,count(distinct numero)
FROM apercuenta
GROUP BY ciudad,agencia;
```

/* nº máximo de cuentas de una sucursal */

```
SELECT max(count(distinct numero)) "nº max. suc."
FROM apercuenta
GROUP BY ciudad,agencia;
```

/* sucursales que tienen 3 cuentas */

```
SELECT ciudad,agencia
FROM apercuenta
WHERE 3 = (SELECT count(distinct numero)
           FROM apercuenta ap
                  WHERE apercuenta.ciudad  = ap.ciudad and
                        apercuenta.agencia = ap.agencia)
GROUP BY ciudad,agencia;
```

/* agencias cuyo nº es una unidad menor que el mayor nº de cuentas de una agencia */

```
SELECT ciudad,agencia
FROM apercuenta
WHERE agencia+1 = (SELECT max(count(distinct numero))
                   FROM apercuenta
                   GROUP BY ciudad,agencia);

SET pagesize 64;
SET linesize 100;
column  TABLE_NAME format a30;
column comments format a68;
```

/* visualización del diccionario del SGBD */

```
SELECT table_name, comments FROM DICT;
```

/* visualización de las tablas, vistas, etc. accesibles al usuario */

```
SELECT * FROM USER_CATALOG;
```

/* adición de un cliente con un campo nulo (la calle) */

```
INSERT INTO cliente VALUES ('Pedro',179,'','Zaragoza');
SELECT * FROM cliente;
```

/* obtención del número de clientes con calle conocida */

```
SELECT count (calle) FROM cliente;
```

/* obtención del número de clientes */

```
SELECT count(*) FROM cliente;
```

/* visualizar los clientes con calle desconocida */

```
SELECT * FROM cliente WHERE calle is NULL;
```

/* añadir un cliente que vive donde 'José López' */

```
INSERT INTO cliente
VALUES ('Angel Sanchez', 17171718, 'Mayor 17', 'Zaragoza');
```

/* listado de clientes que viven en la misma ciudad y calle que el cliente con dni = 15444345 */

```
SELECT nombre,dni,ciudad,calle
FROM cliente
WHERE (ciudad,calle) = (SELECT ciudad,calle
                        FROM cliente
                        WHERE dni=15444345);
```

/* otro modo alternativo sería */

```
SELECT X.nombre,X.dni,X.ciudad,X.calle
FROM cliente X, cliente Y
WHERE X.calle=Y.calle and X.ciudad=Y.ciudad and Y.dni=15444345;
```

/* listado del número de clientes que hay en cada ciudad */

```
SELECT ciudad,count(*)
FROM cliente
GROUP BY ciudad;
```

/* listado de las ciudades con más de 2 clientes */

```
SELECT ciudad,count(*)
FROM cliente
GROUP BY ciudad
HAVING count(*) > 2;
```

/* cambio del importe del prestamo número 17 */

```
UPDATE prestamo
SET importe=110000
WHERE numprestamo=17;
```

/* listado de préstamos por importe mayor que la media de los importes */
/* de los prestamos que tienen igual interés (agrupados por interés) */

```
SELECT numprestamo,importe,interes
FROM prestamo X
WHERE importe > (SELECT avg(importe)
                 FROM prestamo
                 WHERE X.interes=interes)
ORDER BY interes;
```

/* informe de dichos préstamos para ilustrar el uso del group by */

```
SELECT interes,avg(importe),max(importe),count(*)
FROM prestamo X
WHERE importe > (SELECT avg(importe) FROM prestamo WHERE X.interes=interes)
GROUP BY interes;
```

```
/*************************************************************************/
/* Simplificación del esquema eliminando prestformalizado */
/*************************************************************************/

/* adición de la cuenta ligada al prestamo a la tabla prestamo */
/* paso 1: creación de la columna numCuenta */
ALTER TABLE prestamo add (numCuenta number(10));

/* listado de la información de los prestamos (para comprobar) */
SELECT X.numprestamo,importe,interes,numero
FROM prestamo X, prestformalizado Y
WHERE X.numprestamo = Y.numprestamo;

/* paso 2: asignación de valores a la columna numCuenta */
UPDATE prestamo X
SET numCuenta=(SELECT numero
               FROM prestformalizado Y
               WHERE X.numprestamo=Y.numprestamo);
DROP TABLE prestformalizado;
```

```
/*****************************************************************************/
/* Simplificación del esquema eliminando invformalizada   */
/*****************************************************************************/

/* adición de la cuenta ligada a la inversion a la tabla inversion */
/* paso 1: creación de la columna numCuenta */
ALTER TABLE inversion add (numCuenta number(10));

/* listado de la información de las inversiones (para comprobar) */
SELECT X.numinversion,importe,interes,numero
FROM inversion X, invformalizada Y
WHERE X.numinversion = Y.numinversion;

/* paso 2: asignación de valores a la columna numCuenta */
UPDATE inversion X
SET numCuenta=(SELECT numero
               FROM invformalizada Y
               WHERE X.numinversion=Y.numinversion);
DROP TABLE invformalizada;


/* comprobación del resultado de la operacion */
/* visualización de las tablas, vistas, etc. accesibles al usuario */
SELECT * FROM CAT;
SELECT * FROM prestamo;
SELECT * FROM inversion;
```

ej2_Banco.sql

```
/* @ej2_OpBanco */
/*************************************************************/
/*  Eliminar los cambios introducidos por ej1    */
/*************************************************************/
DELETE FROM cliente WHERE dni=179;

DELETE FROM cliente WHERE dni=17171718;

UPDATE prestamo
SET    importe=200000
WHERE  numprestamo=17;

CREATE TABLE prestFormalizado (numero, numprestamo)
AS (SELECT numCuenta,numprestamo
    FROM prestamo);

/* eliminar la columna añadida a préstamo (nueva tabla) */
RENAME prestamo TO oldPrestamo;

CREATE TABLE prestamo (numprestamo,importe,interes)
AS (SELECT numprestamo,importe,interes
    FROM oldPrestamo);

DROP TABLE oldPrestamo;

CREATE TABLE invFormalizada (numero, numInversion)
AS (SELECT numCuenta,numInversion
    FROM inversion);

/* eliminar la columna añadida a inversión (nueva tabla) */
RENAME inversion TO oldInversion;

CREATE TABLE inversion (numInversion,importe,interes)
    AS (SELECT numinversion,importe,interes
        FROM oldInversion);

DROP TABLE oldInversion;

SELECT * FROM USER_CATALOG;

SELECT * FROM prestFormalizado;
SELECT * FROM invFormalizada;
SELECT * FROM prestamo;
SELECT * FROM inversion;
```

OtraPreg.SQL

```
/* @OtraPreg.sql */
/* hacer eco de las sentencias */
SET ECHO ON
/* especificar líneas de hasta 132 caracteres */
SET LINESIZE 132
/* especificar tamaño de página 80 líneas */
SET PAGESIZE 80
/* truncar los valores de los atributos si "no caben" en la columna */
SET WRAP OFF
```

/* obtener el número de clientes que tienen abiertas cuentas en más de una sucursal */

```
/*****************************************************************/
/* Soluc.1: creando una vista auxiliar con el número de agencias */
/*          en que tiene cuentas cada cliente                    */
/*****************************************************************/
DROP   VIEW numAgenCliente;
CREATE VIEW numAgenCliente
  AS SELECT count(distinct agencia) numAg,dni
     FROM apercuenta
     GROUP BY dni;
SELECT * FROM numAgenCliente;            -- comprobar la vista creada

/* ahora basta con contar las filas que tienen en la columna numAg un valor > 1
*/
SELECT count(distinct dni) masUnaAg
FROM numAgenCliente
WHERE numAg>1;

DROP VIEW numAgenCliente;                -- eliminar la vista creada

/*****************************************************************/
/* Soluc.2: utilizando la cláusula "having"                      */
/*****************************************************************/
/* a) selección de los clientes con cuentas en más de una sucursal */
SELECT count(distinct agencia),dni
FROM apercuenta
GROUP BY dni
HAVING count(distinct agencia) > 1 ;

/* b) contar los clientes con cuentas en más de una sucursal */
SELECT count(count(distinct agencia))
FROM apercuenta
GROUP BY dni
HAVING count(distinct agencia) > 1 ;

/*****************************************************************/
/* Soluc.3: construyendo join para buscar                        */
/*****************************************************************/
/* a) selección de los clientes con cuentas en más de una sucursal */
SELECT *
FROM apercuenta ap1, apercuenta ap2
WHERE ap1.dni=ap2.dni AND ap1.agencia<>ap2.agencia;

/*    puesto que así salen muchos repetidos, hay que ...     */
/* b) contar los clientes con cuentas en más de una sucursal */
SELECT count(distinct ap1.dni)
FROM apercuenta ap1, apercuenta ap2
WHERE ap1.dni=ap2.dni AND ap1.agencia<>ap2.agencia;
```

```
/*********************************************************************/
/* Soluc.4: con selecciones encadenadas                              */
/*********************************************************************/

/* a) selección de los clientes con cuentas en más de una sucursal */

SELECT *
FROM apercuenta ap1
WHERE 0 < (SELECT count(*)
           FROM apercuenta ap2
           WHERE  ap2.dni = ap1.dni  AND
                  ap2.agencia <> ap1.agencia);

/* b) contar los clientes con cuentas en más de una sucursal */

SELECT count(distinct dni)
FROM apercuenta ap1
WHERE 0 < (SELECT count(*)
           FROM apercuenta ap2
           WHERE  ap2.dni = ap1.dni  AND
                  ap2.agencia <> ap1.agencia);


/*********************************************************************/
/* Soluc.5: con el lenguaje PL/SQL                                   */
/*********************************************************************/

/* a) Selección de los clientes con cuentas en más de 1 sucursal */

CREATE TABLE temp (nAg number(3), dni number);

DECLARE
    elDni    number;
    numAg    number;
    cont     number(3) := 0;

    CURSOR c1 IS  SELECT distinct dni FROM apercuenta;

BEGIN
 OPEN c1;
 LOOP
  FETCH c1 INTO elDni;

  EXIT WHEN c1%NOTFOUND;

  SELECT count(distinct agencia) INTO numAg
  FROM apercuenta
  WHERE dni=elDni;
  IF numAg > 1 THEN
    INSERT INTO temp VALUES (numAg, elDni);
    cont := cont +1;
  END IF;
 END LOOP;
 COMMIT;
END;
/
SELECT * FROM temp;

DROP TABLE temp;
```

```
/* b) Contar los clientes con cuentas en más de 1 sucursal */

CREATE TABLE temp (nAg number(3));

DECLARE
    elDni    number;
    numAg    number;
    cont     number(3) := 0;

    CURSOR c1 IS  SELECT distinct dni FROM apercuenta;
BEGIN
 OPEN c1;
 LOOP
  FETCH c1 INTO elDni;

  EXIT WHEN c1%NOTFOUND;

  SELECT count(distinct agencia) INTO numAg
  FROM apercuenta
  WHERE dni=elDni;
  IF numAg > 1 THEN
    cont := cont +1;
  END IF;
 END LOOP;
 INSERT INTO temp VALUES (cont);
 COMMIT;
END;
/
SELECT * FROM temp;

DROP TABLE temp;
```

elimRestric.SQL

```
/* @elimRestric */
/* este script ilustra con un muy sencillo ejemplo la capacidad de SQLPLUS para  realizar tareas complejas de forma
   automatizada y simple.    Elimina todas las restricciones definidas sobre las tablas indicadas  interactivamente por el
   usuario.   Se seleccionan todas las tablas cuyo nombre es  LIKE <nombreIntroducido>
   La idea es crear un fichero con las sentencias de eliminación que, posteriormente es ejecutado. Las sentencias se cons-
   truyen a partir de la consulta adecuada.
*/
SET LINESIZE 350
SET PAGESIZE 350
/* preguntar el nombre de la tabla y mostrar las restricciones definidas en ella */
SELECT * FROM USER_CONSTRAINTS WHERE TABLE_NAME LIKE UPPER('&&laTabla');

SET HEADING  OFF
SET HEADSEP  OFF
SET ECHO     OFF
SET FEEDBACK OFF

UNDEFINE laTabla

SPOOL tarea.sql
/* generar las sentencias de eliminación con la informacion del diccionario */
SELECT 'ALTER TABLE ' || TABLE_NAME, ' DROP CONSTRAINT '||CONSTRAINT_NAME ||';'
FROM  USER_CONSTRAINTS
WHERE TABLE_NAME LIKE UPPER('&&laTabla');

SPOOL OFF
SET HEADSEP  ON
SET FEEDBACK ON
SET HEADING  ON
SET ECHO     ON
UNDEFINE laTabla

/*  eliminar las restricciones de la tabla
*/
@tarea
```

borraTodo.SQL

```
/* @borraTodo.sql */
/* este script ilustra con un muy sencillo ejemplo la capacidad de SQLPLUS para realizar tareas complejas de forma
   automatizada y simple.  Sirve para eliminar todas las tablas, vistas y sinonimos definidos por el usuario.
  La idea es crear un fichero con las sentencias de eliminación que, posteriormente es ejecutado.
  Las sentencias se construyen a partir de la consulta adecuada.

  Aunque se intenta eliminar algunas restricciones, no se hace ninguna comprobación especial, por lo que puede
  producirse algun error al eliminar dterminados elementos
*/
SET LINESIZE 350
SET PAGESIZE 350

/* visualizar los elementos del catálogo del usuario */
SELECT * FROM cat;

SET HEADING  OFF
SET HEADSEP  OFF
SET ECHO     OFF
SET FEEDBACK OFF

SPOOL tarea.sql

/* generar las sentencias de eliminación con la informacion del diccionario */
SELECT 'ALTER TABLE ' || TABLE_NAME, ' DROP CONSTRAINT '||CONSTRAINT_NAME ||';'
FROM  USER_CONSTRAINTS;

SELECT 'DROP TABLE '||TABLE_NAME ||';'
FROM  CAT
WHERE TABLE_TYPE = 'TABLE';

SELECT 'DROP VIEW '||TABLE_NAME ||';'
FROM  CAT
WHERE TABLE_TYPE = 'VIEW';

SELECT 'DROP SYNONYM '||TABLE_NAME ||';'
FROM  CAT
WHERE TABLE_TYPE = 'SYNONYM';

SPOOL OFF
SET HEADSEP  ON
SET FEEDBACK ON
SET HEADING  ON
SET ECHO     ON

/*  eliminar las tablas, vistas y sinonimos
*/
@tarea
```

EmpDep.sql

```
/*  Ejemplos SQL para entorno SQLPlus Oracle 8  */

/* Eliminación esquema anterior */

DROP PROCEDURE Currar;
DROP TRIGGER   ContabHorasProy;

DROP  TABLE Participar;
ALTER TABLE Departamento  DROP CONSTRAINT FK_Departamento;
DROP  TABLE HistorialSalario;
DROP  TABLE Empleado;
DROP  TABLE Departamento;
DROP  TABLE Proyecto;

DROP  CLUSTER clusterEmpDep;

/* Creación del nuevo esquema */

CREATE CLUSTER clusterEmpDep(numDep char(2));

CREATE INDEX ind_clusterEmpDep ON CLUSTER clusterEmpDep;

CREATE TABLE Departamento (
  codDep        char(2)     PRIMARY KEY,
  nombre        varchar(15) UNIQUE NOT NULL,
  director      char(4))
 CLUSTER clusterEmpDep(codDep);

CREATE TABLE Empleado (
  codEmp        char(4),
  nombre        varchar(20) NOT NULL,
  puesto        varchar(15),
  salario       numeric(8),
  codDepto      char(2),
  FinContrato   date,
  CONSTRAINT PK_Empleado PRIMARY KEY (codEmp),
  CONSTRAINT FK_Empleado FOREIGN KEY (codDepto) REFERENCES Departamento, /*ON UPDATE CASCADE*/
  CHECK (salario > 0))
 CLUSTER clusterEmpDep(codDepto);

CREATE UNIQUE INDEX ind_nombre ON Empleado(nombre);

CREATE INDEX ind_dep ON Empleado(codDepto);

CREATE TABLE Proyecto (
  numProy       numeric(5)  PRIMARY KEY,
  titulo        varchar(20) NOT NULL,
  presupuesto   numeric(9),
  horas         number(4));

CREATE TABLE Participar (
  codEmp        char(4),
  numProy       numeric(5),
  CONSTRAINT PK_Participar PRIMARY KEY (codEmp, numProy),
  FOREIGN KEY (codEmp)  REFERENCES Empleado(codEmp),
  FOREIGN KEY (numProy) REFERENCES Proyecto(numProy) ON DELETE CASCADE);
```

```
/* Inserción de datos */

INSERT INTO Departamento VALUES ('01', 'Software',       'A369');
INSERT INTO Departamento VALUES ('02', 'Hardware',       'A325');
INSERT INTO Departamento VALUES ('03', 'Comunicaciones', 'A369');
INSERT INTO Departamento VALUES ('04', 'Mantenimiento',  'A777');

INSERT INTO Empleado
 VALUES ('A369', 'Fernández', 'jefeProyecto', 5500000, '01', to_date('28-02-98', 'DD-MM-YY'));
INSERT INTO Empleado
 VALUES ('A269', 'Martinez',  'analista',     4500000, '01', to_date('18-12-97', 'DD-MM-YY'));
INSERT INTO Empleado
 VALUES ('A359', 'López',     'analista',        NULL, '02', to_date('12-10-97', 'DD-MM-YY'));
INSERT INTO Empleado
 VALUES ('A367', 'González',  'programador',  3000000, '01', to_date('08-02-99', 'DD-MM-YY'));
INSERT INTO Empleado
 VALUES ('A569', 'Rodriguez', 'programador',  2800000, NULL, to_date('02-09-99', 'DD-MM-YY'));
INSERT INTO Empleado
 VALUES ('A389', 'Sánchez',   'programador',  2600000, '03', to_date('22-06-98', 'DD-MM-YY'));
INSERT INTO Empleado
 VALUES ('A361', 'Juarez',    'programador',  2300000, '02', to_date('04-10-99', 'DD-MM-YY'));
INSERT INTO Empleado
 VALUES ('A777', 'Ruperez',    NULL,          3500000, '04', to_date('15-02-99', 'DD-MM-YY'));
INSERT INTO Empleado
 VALUES ('A325', 'Vázquez',   'jefeProyecto', 6000000, '04', to_date('15-07-97', 'DD-MM-YY'));
INSERT INTO Empleado
 VALUES ('A364', 'Márquez',   'analista',     3800000, '04', to_date('30-03-98', 'DD-MM-YY'));
INSERT INTO Empleado
 VALUES ('A969', 'Jiménez',   'programador',  2000000, NULL, to_date('31-05-97', 'DD-MM-YY'));
INSERT INTO Empleado
 VALUES ('A312', 'Pérez',     'programador',  1500000, '04', to_date('23-02-99', 'DD-MM-YY'));

INSERT INTO Proyecto VALUES (1, 'SoftIntegral', 15000000, 0);
INSERT INTO Proyecto VALUES (2, 'Web Magic',    25000000, 0);
INSERT INTO Proyecto VALUES (3, 'Copia y Pega', 17000000, 0);
INSERT INTO Proyecto VALUES (4, 'Word Maker',    5000000, 0);
INSERT INTO Proyecto VALUES (5, 'ContaWell',     2000000, 0);
INSERT INTO Proyecto VALUES (6, 'Saucer Attack', 9000000, 0);

INSERT INTO Participar VALUES ('A369', 1);
INSERT INTO Participar VALUES ('A359', 1);
INSERT INTO Participar VALUES ('A389', 1);
INSERT INTO Participar VALUES ('A777', 2);
INSERT INTO Participar VALUES ('A969', 2);
INSERT INTO Participar VALUES ('A325', 3);
INSERT INTO Participar VALUES ('A777', 4);
INSERT INTO Participar VALUES ('A312', 4);
```

```
/* Modificación del esquema inicial */

ALTER TABLE empleado MODIFY (nombre varchar(40));

ALTER TABLE Participar ADD (NumHoras number(4), CONSTRAINT CH_NumHoras CHECK (NumHoras>0));
UPDATE Participar SET NumHoras = 15 * numProy;

UPDATE Empleado SET salario = salario * 1.1;

COMMIT;

ALTER TABLE Departamento
  ADD CONSTRAINT FK_Departamento FOREIGN KEY (director) REFERENCES Empleado(codEmp);


/* Definición de procedimientos y disparadores */
```

```
CREATE OR REPLACE PROCEDURE Currar (
 emp   IN Participar.codemp%type,
 proy  IN Participar.numproy%type,
 horas IN Participar.numhoras%type)
 AS
  begin
    UPDATE Participar
    SET    numHoras = numHoras + horas
    WHERE  emp = codemp AND proy = numproy;
  end;
.
RUN;
```

```
CREATE OR REPLACE TRIGGER ContabHorasProy
AFTER UPDATE ON Participar
FOR EACH ROW
WHEN (NEW.numHoras > 0)
 begin
   UPDATE Proyecto
   SET    horas = horas + :NEW.numHoras - :OLD.numHoras
   WHERE  numProy = :NEW.numProy;
 end ContabHorasProy;
.
RUN;

CREATE TABLE HistorialSalario (
  codEmp             char(4),
  salarioAnt         numeric(8),
  fechaCambioSalario  date,
  CONSTRAINT PK_HistSalario    PRIMARY KEY (codEmp, fechaCambioSalario),
  CONSTRAINT FK_HistSalarioEmp FOREIGN KEY (codEmp) REFERENCES Empleado);
```

```
CREATE OR REPLACE TRIGGER GuardarHistorialSalario
BEFORE UPDATE ON Empleado
FOR EACH ROW
 begin
  if (:OLD.Salario <> :NEW.salario)
   then INSERT INTO HistorialSalario VALUES (:OLD.codEmp, :OLD.salario, sysdate);
  end if;
 end GuardarHistorialSalario;
.
RUN;
```

Consulta.sql

```
/*  Ejemplos SQL para entorno SQLPlus Oracle 8  */

/* Formateado de la salida de datos */
column nombre   format a20;
column codEmp   format a6;
column codDepto format a6;

SELECT * FROM Empleado;
SELECT * FROM Departamento;
SELECT * FROM Proyecto;
SELECT * FROM Participar;
SELECT codEmp, nombre FROM Empleado WHERE codDepto= '03';
SELECT * FROM Empleado WHERE codDepto IS NULL;
SELECT codEmp FROM Empleado WHERE codDepto= '02' AND salario<2000000;
SELECT codEmp FROM Empleado WHERE salario between 2000000 AND 4000000;
SELECT * FROM Empleado WHERE codDepto in ('01', '02', '04');
SELECT codEmp, nombre FROM Empleado WHERE NOT (codDepto = '01');
SELECT * FROM Empleado WHERE nombre like 'P%';
SELECT * FROM Empleado WHERE nombre like 'R_____';
SELECT avg(salario) SalarioMedio FROM Empleado WHERE puesto = 'analista';
SELECT distinct puesto FROM Empleado;
SELECT count(puesto) NumeroPuestos FROM Empleado;
SELECT * FROM Empleado ORDER BY codDepto, salario desc;
SELECT * FROM Empleado WHERE FinContrato < add_months(sysdate, 6);
SELECT codDepto, avg(salario) SalarioMedio FROM Empleado GROUP BY codDepto;
SELECT codDepto, avg(salario) SalarioMedio FROM Empleado
 GROUP BY codDepto HAVING min(salario) < 2000000;
SELECT nombre FROM Departamento WHERE EXISTS (SELECT * FROM Empleado
 WHERE Empleado.codDepto = Departamento.codDep AND salario > 5000000);
SELECT nombre FROM Empleado
 WHERE codDepto in (SELECT codDepto FROM Empleado WHERE nombre='Martinez');
SELECT Empleado.nombre, Proyecto.Titulo FROM Empleado, Participar, Proyecto
 WHERE Empleado.codEmp = Participar.codEmp AND Participar.numProy = Proyecto.numProy;
SELECT E2.nombre FROM Empleado E1, Empleado E2
 WHERE E1.codDepto = E2.codDepto AND E1.codEmp <> E2.codEmp AND E1.nombre = 'Martinez';
SELECT * FROM Empleado WHERE salario>5000000
 UNION SELECT * FROM Empleado WHERE salario<2000000;

/* Creación y eliminación de una vista */
CREATE VIEW Cargos AS
 SELECT codEmp, nombre, puesto, salario FROM Empleado WHERE puesto = 'analista'
 WITH CHECK OPTION;
SELECT * FROM Cargos WHERE salario > 4000000;
INSERT INTO Cargos VALUES ('B666', 'Dominguez', 'analista', NULL);
DROP VIEW Cargos;

/* Número medio de horas dedicadas por empleado en proyectos que participan */
SELECT codEmp, sum(NumHoras)/count(*) FROM Participar GROUP BY codEmp;
SELECT codEmp, avg(NumHoras) FROM Participar GROUP BY codEmp;

/* Título de los proyectos que se desarrollan en el departamento Software */
SELECT distinct titulo
FROM Proyecto P, Empleado E, Participar Pa, Departamento D
WHERE (P.numProy = Pa.numProy) AND (D.codDep = E.codDepto) AND
      (Pa.codEmp = E.codEmp)   AND (D.nombre = 'Software');
```

```
SELECT distinct titulo
FROM Proyecto
WHERE EXISTS
 (SELECT *
  FROM Empleado
  WHERE (Empleado.codDepto=(SELECT codDep FROM Departamento WHERE nombre = 'Software'))
    AND EXISTS (SELECT * FROM Participar
                WHERE (codEmp = Empleado.codEmp) AND (Participar.numProy =
Proyecto.numProy)));

/* Empleados que participan en todos los proyectos */
SELECT codEmp, nombre
FROM Empleado
WHERE NOT EXISTS
  (SELECT * FROM Proyecto
   WHERE NOT EXISTS
      (SELECT * FROM Participar
       WHERE (Participar.numProy = Proyecto.numProy) AND
             (Participar.codEmp = Empleado.codEmp)));

/* Listado de proyectos ordenados por el número de empleados asignados */
CREATE VIEW dedicacion AS
 SELECT numProy, count(*) NumEmpleados
 FROM Participar
 GROUP BY numProy;
SELECT P.titulo
FROM Dedicacion D, Proyecto P
WHERE (P.numProy=D.numProy)
ORDER BY NumEmpleados;
DROP VIEW Dedicacion;

/* Otras operaciones */
/* Introducción de datos para una consulta */
ACCEPT horas PROMPT 'Horas trabajadas: ';
SELECT * FROM Participar WHERE numHoras = &horas;
SELECT * FROM Proyecto WHERE numProy = 1;

/* Llamada a un procedimiento almacenado */
BEGIN
 Currar('A369', 1, 5);
END;
.
RUN;
SELECT * FROM Proyecto WHERE numProy = 1;
SELECT * FROM HistorialSalario;
UPDATE empleado SET salario = salario + 100000 WHERE codemp = 'A369';
SELECT * FROM HistorialSalario;
DROP TRIGGER GuardarHistorialSalario;

CREATE OR REPLACE TRIGGER GuardarHistorialSalario
BEFORE UPDATE ON Empleado
FOR EACH ROW
DECLARE
   aux integer;
 begin
  if (:OLD.Salario <> :NEW.salario)
   then
    SELECT count(*) INTO aux
    FROM   HistorialSalario HS
    WHERE   HS.codEmp = :NEW.codEmp AND
            trunc(fechaCambioSalario) = trunc(sysdate);
  if aux > 0
    then raise_application_error( -20501, 'No puede haber 2 cambios el mismo día');
    else INSERT INTO HistorialSalario VALUES (:OLD.codEmp, :OLD.salario, sysdate);
   end if;
  end if;
 end GuardarHistorialSalario;
.
RUN;
```

creatblProd_DATE.sql

```
/* @creatblProd_DATE.sql */
SET ECHO ON

/* Crear las tablas y añadir algunas restricciones, de varias maneras
*/
CREATE TABLE Suministradores (
  rfSum    char(3),
  nombSum  char(20) CONSTRAINT NN_nSumin NOT NULL,
  NIF      char(10) CONSTRAINT UN_NIF_S  UNIQUE,
  ciudad   char(12));
ALTER  TABLE Suministradores add    (CONSTRAINT PK_Sumin PRIMARY KEY (rfSum));
ALTER  TABLE Suministradores modify (NIF CONSTRAINT NN_NIF_S  NOT NULL);

CREATE TABLE Piezas (
  rfPza    char(4)   CONSTRAINT PK_Piezas   PRIMARY KEY,
  nombPza  char(14)  CONSTRAINT NN_nPieza   NOT NULL,
  color    char(10),
  peso     number(2),
  ciudad   char(12));

CREATE TABLE Proyectos (
  rfProy   char(3),
  nombProy char(20) CONSTRAINT NN_nProy NOT NULL,
  ciudad   char(12),
  CONSTRAINT PK_Proy PRIMARY KEY (rfProy));

CREATE TABLE SuminPiezasProy (
  rfSum  char(3)  CONSTRAINT FK_SP_Sum REFERENCES Suministradores(rfSum),
  rfPza  char(4),
  rfProy char(3),
  cant   number (3),
  CONSTRAINT FK_SP_Pza  FOREIGN KEY (rfPza)  REFERENCES Piezas(rfPza),
  CONSTRAINT FK_SP_Proy FOREIGN KEY (rfProy) REFERENCES Proyectos(rfProy));

DESCRIBE Suministradores;
DESCRIBE Piezas;
DESCRIBE Proyectos;
DESCRIBE SuminPiezasProy;
```

elimTblProd_DATE.sql

```
/* @elimTblProd_DATE.sql */
SET ECHO ON

/* Eliminar las tablas y todas las restricciones ligadas. SuminPiezasProy hay que eliminarla primero, porque
   referencia a las tablas Suministradores y Piezas
 */
DROP TABLE SuminPiezasProy;
DROP TABLE Suministradores;
DROP TABLE Piezas;
DROP TABLE Proyectos;

SELECT * FROM CAT;
```

ponDatProd_DATE.sql

```
/* @ponDatProd_DATE.sql */

INSERT INTO Suministradores VALUES ('S1', 'Salazar', 20, 'Londres');
INSERT INTO Suministradores VALUES ('S2', 'Jaimes',  10, 'Paris');
INSERT INTO Suministradores VALUES ('S3', 'Bernal',  30, 'Paris');
INSERT INTO Suministradores VALUES ('S4', 'Corona',  25, 'Londres');
INSERT INTO Suministradores VALUES ('S5', 'Aldana',  35, 'Atenas');

INSERT INTO Piezas VALUES ('P1', 'Tuerca',    'Rojo',  12, 'Londres');
INSERT INTO Piezas VALUES ('P2', 'Perno',     'Verde', 17, 'Paris');
INSERT INTO Piezas VALUES ('P3', 'Birlo',     'Azul',  17, 'Roma');
INSERT INTO Piezas VALUES ('P4', 'Birlo',     'Rojo',  14, 'Londres');
INSERT INTO Piezas VALUES ('P5', 'Leva',      'Azul',  12, 'Paris');
INSERT INTO Piezas VALUES ('P6', 'Engranaje', 'Rojo',  19, 'Londres');

INSERT INTO Proyectos VALUES ('J1', 'Clasificador', 'Paris');
INSERT INTO Proyectos VALUES ('J2', 'Perforadora',  'Roma');
INSERT INTO Proyectos VALUES ('J3', 'Lectora',      'Atenas');
INSERT INTO Proyectos VALUES ('J4', 'Consola',      'Atenas');
INSERT INTO Proyectos VALUES ('J5', 'Componedora',  'Londres');
INSERT INTO Proyectos VALUES ('J6', 'Terminal',     'Oslo');
INSERT INTO Proyectos VALUES ('J7', 'Cinta',        'Londres');

INSERT INTO SuminPiezasProy VALUES ('S1', 'P1', 'J1', 200);
INSERT INTO SuminPiezasProy VALUES ('S1', 'P1', 'J4', 700);
INSERT INTO SuminPiezasProy VALUES ('S2', 'P3', 'J1', 400);
INSERT INTO SuminPiezasProy VALUES ('S2', 'P3', 'J2', 200);
INSERT INTO SuminPiezasProy VALUES ('S2', 'P3', 'J3', 200);
INSERT INTO SuminPiezasProy VALUES ('S2', 'P3', 'J4', 500);
INSERT INTO SuminPiezasProy VALUES ('S2', 'P3', 'J5', 600);
INSERT INTO SuminPiezasProy VALUES ('S2', 'P3', 'J6', 400);
INSERT INTO SuminPiezasProy VALUES ('S2', 'P3', 'J7', 800);
INSERT INTO SuminPiezasProy VALUES ('S2', 'P5', 'J2', 100);
INSERT INTO SuminPiezasProy VALUES ('S3', 'P3', 'J1', 200);
INSERT INTO SuminPiezasProy VALUES ('S3', 'P4', 'J2', 500);
INSERT INTO SuminPiezasProy VALUES ('S4', 'P6', 'J3', 300);
INSERT INTO SuminPiezasProy VALUES ('S4', 'P6', 'J7', 300);
INSERT INTO SuminPiezasProy VALUES ('S5', 'P2', 'J2', 200);
INSERT INTO SuminPiezasProy VALUES ('S5', 'P2', 'J4', 100);
INSERT INTO SuminPiezasProy VALUES ('S5', 'P5', 'J5', 500);
INSERT INTO SuminPiezasProy VALUES ('S5', 'P5', 'J7', 100);
INSERT INTO SuminPiezasProy VALUES ('S5', 'P5', 'J7', 100);
INSERT INTO SuminPiezasProy VALUES ('S5', 'P1', 'J4', 100);
INSERT INTO SuminPiezasProy VALUES ('S5', 'P3', 'J4', 200);
INSERT INTO SuminPiezasProy VALUES ('S5', 'P4', 'J4', 800);
INSERT INTO SuminPiezasProy VALUES ('S5', 'P5', 'J4', 400);
INSERT INTO SuminPiezasProy VALUES ('S5', 'P6', 'J4', 500);

SELECT * FROM Suministradores;
SELECT * FROM Piezas;
SELECT * FROM Proyectos;
SELECT * FROM SuminPiezasProy;
```

elimDatProd_DATE.sql

```
/* @elimDatProd_DATE.sql */

DELETE FROM SuminPiezasProy;
DELETE FROM Suministradores;
DELETE FROM Piezas;
DELETE FROM Proyectos;

SELECT * FROM Suministradores;
SELECT * FROM Piezas;
SELECT * FROM Proyectos;
SELECT * FROM SuminPiezasProy;
```

PregProd1_DATE.sql

```
/* @PregProd1_DATE.sql */
SET LINESIZE 132
SET PAGE 60
SET ECHO ON

DESCRIBE Suministradores;
DESCRIBE Piezas;
DESCRIBE Proyectos;
DESCRIBE SuminPiezasProy;

/*   Obtener la ref. y el NIF  de todos los proveedores de París
*/
SELECT rfSum, NIF FROM Suministradores WHERE ciudad = 'Paris';

/*   Obtener los Suministradores y las Piezas de la misma ciudad (equijoin)
*/
SELECT S.* , P.*
 FROM Suministradores S, Piezas P
 WHERE S.ciudad = P.ciudad;

/*  Obtener la ref. y ciudad de los Suministradores y las Piezas relacionadas  (es decir, de las piezas y sus suministradores)
 */
SELECT distinct S.rfSum, S.ciudad , P.rfPza, P.ciudad
 FROM Suministradores S, Piezas P, SuminPiezasProy SPP
 WHERE S.rfSum = SPP.rfSum and P.rfPza = SPP.rfPza;

SELECT distinct S.rfSum, S.ciudad , P.rfPza, P.ciudad
 FROM Suministradores S, Piezas P
 WHERE EXISTS
       (SELECT *
         FROM SuminPiezasProy
         WHERE rfSum = S.rfSum and rfPza = P.rfPza);

SELECT distinct S.rfSum, S.ciudad , P.rfPza, P.ciudad
 FROM Suministradores S, Piezas P
 WHERE (S.rfSum, P.rfPza) in (SELECT rfSum, rfPza FROM SuminPiezasProy);

/* Obtener el nombre de los Suministradores que no suministran la pieza  'P2'
*/
SELECT nombSum
 FROM Suministradores S
 WHERE NOT EXISTS
      (SELECT * FROM SuminPiezasProy
       WHERE rfSum = S.rfSum and rfPza = 'P2');

SELECT nombSum
 FROM Suministradores S
 WHERE rfSum NOT IN
      (SELECT rfSum FROM SuminPiezasProy
       WHERE  rfPza = 'P2');

SELECT nombSum FROM Suministradores
MINUS
SELECT nombSum
 FROM Suministradores S, SuminPiezasProy SPP
 WHERE S.rfSum = SPP.rfSum and rfPza = 'P2';

SELECT nombSum FROM Suministradores
MINUS
SELECT nombSum
 FROM Suministradores S
 WHERE S.rfSum in
       (SELECT rfSum FROM SuminPiezasProy WHERE rfPza = 'P2');
```

```
/* Obtener el nombre de los Suministradores que suministran todas las piezas (es decir, de aquellos tales que no hay
   ninguna pieza que no suministren
 */
SELECT nombSum
 FROM Suministradores S
 WHERE NOT EXISTS
      (SELECT * FROM Piezas P
        WHERE NOT EXISTS
             (SELECT * FROM SuminPiezasProy
              WHERE rfSum = S.rfSum and rfPza = P.rfPza));

column "Gramos"  FORMAT 99999

/* Obtener la ref. de las piezas, su peso, color y cantidad máxima suministrada, de las piezas de color 'rojo' o 'azul'
   tales que la cantidad total suministrada sea mayor que 350, excluyendo en este computo los envíos de cantidades menores
   o iguales que 200.      El listado deberá estar ordenado por cantidades máximas en sentido creciente, y para valores
   iguales, por la referencia de la pieza, en sentido descendiente.
*/
SELECT P.rfPza "rf.", p.Peso * 1000 "Gramos",
       P.Color "Color", 'Cant. max. enviada' " ", max(SPP.cant)
FROM Piezas P, SuminPiezasProy SPP
WHERE P.rfPza = SPP.rfPza         and
      P.color IN ('Rojo', 'Azul') and
      SPP.Cant > 200
GROUP BY P.rfPza, P.Peso, P.Color
HAVING sum(Cant) > 350
ORDER BY 5, P.rfPza desc;

COLUMN gramos  HEADING "peso" FORMAT 99999
COLUMN maxCant HEADING "max." FORMAT 9999

/* también se prodrá poner el nombre de la columna en vez de su número
*/
SELECT P.rfPza " ref.", p.Peso * 1000 gramos, 'gramos, color' " ",
       P.Color " ", 'Cant. max. enviada' " ", max(SPP.cant) maxCant
FROM Piezas P, SuminPiezasProy SPP
WHERE P.rfPza = SPP.rfPza         and
      P.color in ('Rojo', 'Azul') and
      SPP.Cant > 200
GROUP BY P.rfPza, P.Peso, P.Color
HAVING sum(Cant) > 350
ORDER BY maxCant, P.rfPza desc;
```

creaTblProd.sql

```
/* @creaTblProd.sql */
SET ECHO ON

/*  Crear las tablas y añadir algunas restricciones, de varias maneras
*/
CREATE TABLE Suministradores (
  rfSum    char(3),
  nombSum  char(20) CONSTRAINT NN_nSumin NOT NULL,
  NIF      char(10) CONSTRAINT UN_NIF_S  UNIQUE,
  ciudad   char(12));
ALTER  TABLE Suministradores ADD    (CONSTRAINT PK_Sumin PRIMARY KEY (rfSum));
ALTER  TABLE Suministradores MODIFY (NIF CONSTRAINT NN_NIF_S  NOT NULL);

CREATE TABLE Piezas (
  rfPza    char(4)   CONSTRAINT PK_Piezas   PRIMARY KEY,
  nombPza  char(14)  CONSTRAINT NN_nPieza   NOT NULL,
  color    char(10),
  peso     number(2),
  ciudad   char(12));

CREATE TABLE SuminPiezas (
  rfSum char(3)   CONSTRAINT FK_SP_Sum REFERENCES Suministradores(rfSum),
  rfPza char(4),
  cant  number (3),
  PRIMARY KEY (rfSum,rfPza),
  CONSTRAINT FK_SP_Pza  FOREIGN KEY (rfPza) REFERENCES Piezas(rfPza));

DESCRIBE Suministradores;
DESCRIBE SuminPiezas;
DESCRIBE Piezas;
```

ponDatProd.sql

```
/* @ponDatProd.sql */
SET ECHO ON

INSERT INTO Suministradores VALUES ('S01',     'PEREZ',     'X01',    'ZARAGOZA');
INSERT INTO Suministradores VALUES ('S02',  'MARTINEZ',     'A12',      'HUESCA');
INSERT INTO Suministradores VALUES ('S03',     'SOLER',     'B13',      'TERUEL');
INSERT INTO Suministradores VALUES ('S04',      'PLAS',     'A01',    'ZARAGOZA');
INSERT INTO Suministradores VALUES ('S05',   'SANCHEZ',     'A02',      'HUESCA');
INSERT INTO Suministradores VALUES ('S06',     'IDEAL',     'W24',    'ZARAGOZA');
INSERT INTO Suministradores VALUES ('S07',      'MALO',     'W25',    'ZARAGOZA');

INSERT INTO Piezas (rfPza, nombPza, color, peso, ciudad)
  VALUES ('P01R',  'TAPON',   'ROJO', 23,   'MADRID');
INSERT INTO Piezas (rfPza, nombPza, color, peso, ciudad)
  VALUES ('P01V',  'TAPON',  'VERDE', 23, 'ZARAGOZA');
INSERT INTO Piezas (rfPza, nombPza, color, peso, ciudad)
  VALUES ('P02R',  'BOTON',   'ROJO',  8,    'HUESCA');
INSERT INTO Piezas (rfPza, nombPza, color, peso, ciudad)
  VALUES ('P03V',  'CLIP',   'VERDE', 12,   'SEVILLA');

INSERT INTO SuminPiezas VALUES ('S01', 'P01R', 300);
INSERT INTO SuminPiezas VALUES ('S01', 'P01V', 100);
INSERT INTO SuminPiezas VALUES ('S02', 'P01R',  50);
INSERT INTO SuminPiezas VALUES ('S03', 'P02R',  80);
INSERT INTO SuminPiezas VALUES ('S04', 'P03V', 860);
INSERT INTO SuminPiezas VALUES ('S05', 'P03V',  60);
INSERT INTO SuminPiezas VALUES ('S06', 'P03V',  30);
INSERT INTO SuminPiezas VALUES ('S06', 'P01R',  12);
INSERT INTO SuminPiezas VALUES ('S06', 'P01V',  50);

SELECT * FROM Suministradores;
SELECT * FROM SuminPiezas;
SELECT * FROM Piezas;
```

elimTblProd.sql

```
/* @elimTblProd.sql */
SET ECHO ON

/* Eliminar las tablas y todas las restricciones ligadas. SuminPiezas hay que eliminarla
   en primer lugar porque referencia a las tablas Suministradores y Piezas
 */
DROP TABLE SuminPiezas;
DROP TABLE Suministradores;
DROP TABLE Piezas;

SELECT * FROM CAT;
```

elimDatProd.sql

```
/* @elimDatProd.sql */
SET ECHO ON

DELETE FROM SuminPiezas;
DELETE FROM Suministradores;
DELETE FROM Piezas;

SELECT * FROM Suministradores;
SELECT * FROM SuminPiezas;
SELECT * FROM Piezas;
```

preg1Prod.sql

```
/* @preg1Prod.sql */
SET ECHO ON

SELECT          rfSum  FROM SuminPiezas;
SELECT distinct rfSum  FROM SuminPiezas;
SELECT distinct rfSum  FROM SuminPiezas WHERE rfSum NOT in ('S01','S03');
SELECT distinct rfSum  FROM SuminPiezas WHERE rfSum  = any ('S01','S03');
SELECT distinct rfSum  FROM SuminPiezas WHERE rfSum <> all ('S01','S03');

/* Obtención de las ciudades donde sólo hay suministradores */
SELECT distinct ciudad FROM Suministradores
 MINUS
SELECT distinct ciudad FROM Piezas;

/* Creación de una tabla con las ref. de las piezas llamadas 'TAPON' */
CREATE TABLE tapones (rfPza char(4));
INSERT INTO  tapones SELECT distinct rfPza FROM Piezas WHERE nombPza = 'TAPON';
SELECT * FROM tapones;

/* Obtención de los suministradores de tapones de todos los tipos. Estos se pueden obtener como (SuminPiezas : tapones).
   Para implementar la división, se opta por crear una vista auxiliar on los pares (rfSum,rfPza) que no existen.
   La solución se obtiene restando de los suministradores existentes los de la vista obtenida.
 */
CREATE VIEW aux AS
  SELECT distinct rfSum,t.rfPza FROM SuminPiezas,tapones t
   MINUS
  SELECT rfSum,rfPza FROM SuminPiezas;
SELECT * FROM aux;
SELECT distinct rfSum FROM SuminPiezas
  MINUS
SELECT distinct rfSum FROM aux;
DROP VIEW aux;
```

```
/*  información de las piezas de cada suministrador  */
SELECT rfSum,P.* FROM SuminPiezas SP, piezas P WHERE sp.rfPza=p.rfPza;

/*  información de los suministradores de piezas en su misma ciudad  */
SELECT * FROM Suministradores
WHERE (rfSum,ciudad) IN
      (SELECT rfSum,ciudad
       FROM SuminPiezas SP, piezas P
       WHERE sp.rfPza=p.rfPza);

/*  información completa de los suministradores y piezas de la misma ciudad  */
SELECT S.*, P.*
FROM   Suministradores S, SuminPiezas SP, Piezas P
WHERE  S.rfSum=SP.rfSum and SP.rfPza=P.rfPza and S.ciudad=P.ciudad;

/*   total de piezas, piezas distintas y nº medio de piezas distintas de cada suministrador  que suministra algún tipo de pieza
*/
SELECT rfSum, count(*) "tipos", sum(cant) "total", avg(cant) "media"
 FROM SuminPiezas
 GROUP BY rfSum;

/*  total de piezas y piezas distintas de cada suministrador (todos)
*/
SELECT rfSum, count(*) "tipos", sum(cant) "total"
 FROM SuminPiezas
 GROUP BY rfSum
UNION
  SELECT rfSum, 0, 0
  FROM   Suministradores
 MINUS
  SELECT rfSum, 0, 0
  FROM SuminPiezas;

SELECT S.rfSum, count(distinct SP.rfPza) "tipos", sum(cant) "total"
 FROM Suministradores S, SuminPiezas SP
 WHERE S.rfSum = SP.rfSum (+)
 GROUP BY S.rfSum;

/*  suministradores de más de 1 piezas distintas  */
SELECT rfSum, count(*) dif FROM SuminPiezas
GROUP BY rfSum
HAVING count(*) > 1;

/*  Obtención del total de proveedores que han suministrado piezas  */
SELECT count(distinct rfSum) "tot. Sumin." FROM SuminPiezas;

/*  suministradores de más de 1 piezas distintas y cantidad > 60  */
SELECT rfSum, count(*) dif FROM SuminPiezas
WHERE cant > 60
GROUP BY rfSum
HAVING count(*) > 1;

/*  proveedores distintos de la pieza 'P01R', y total de piezas recibidas  */
SELECT count(*) "nº Sumin.", sum(cant) "tot. Pzas." FROM SuminPiezas
WHERE rfPza = 'P01R';

DROP   TABLE tapones;
```

Preg2Prod.sql

```
/* @Preg2Prod.sql */

/* eliminar temporalmente algunas restricciones
*/
ALTER TABLE suminpiezas DROP CONSTRAINT FK_SP_Sum;
ALTER TABLE suministradores DROP CONSTRAINT PK_Sumin;
ALTER TABLE suministradores DROP CONSTRAINT UN_NIF_S;

/* añadir un suministrador duplicado y otros dos con sólo el NIF distinto
*/
DELETE FROM Suministradores WHERE rfSum='S01' or rfSum='S02';
INSERT INTO Suministradores VALUES ('S01',     'PEREZ',     'X01',    'ZARAGOZA');
INSERT INTO Suministradores VALUES ('S01',     'PEREZ',     'X01',    'ZARAGOZA');
INSERT INTO Suministradores VALUES ('S02',  'MARTINEZ',     'A12',      'HUESCA');
INSERT INTO Suministradores VALUES ('S02',  'MARTINEZ',     'A11',      'HUESCA');

/* listar los Suministradores con nº de referencia distinto
*/
SELECT distinct rfSum FROM Suministradores;

/* listar los Suministradores distintos
*/
SELECT distinct rfSum, nombSum, NIF, ciudad FROM Suministradores;

/* Obsérvese que la diferencia de ambas tablas es nula
*/
SELECT * FROM Suministradores
 MINUS
SELECT distinct rfSum, nombSum, NIF, ciudad FROM Suministradores;

/* contar los Suministradores que hay por cada nº de referencia
*/
SELECT rfSum, count(rfSum) FROM Suministradores GROUP BY rfSum;

/* contar los números de referencia de Suministradores que hay repetidos
*/
SELECT count(rfSum)-count(distinct rfSum) "rfSum Rep." FROM Suministradores;

/* listar los números de referencia de Suministradores que hay repetidos
*/
SELECT rfSum FROM Suministradores GROUP BY rfSum HAVING count(rfSum) > 1;

/* También se puede obtener con SELECT anidados (observese la necesidad de DISTINCT)
*/
SELECT distinct rfSum, nombSum, NIF, ciudad FROM Suministradores X
 WHERE 1 < (SELECT count(rfSum) FROM suministradores WHERE rfSum=X.rfSum);

/* listar los Suministradores que hay repetidos
*/
SELECT * FROM Suministradores
 GROUP BY rfSum, nombSum, NIF, ciudad
 HAVING count(rfSum) > 1;

/* También se puede obtener con SELECT anidados (observese la necesidad de DISTINCT)
*/
SELECT distinct rfSum, nombSum, NIF, ciudad FROM Suministradores X
 WHERE 1 < ( SELECT count(rfSum)
             FROM suministradores
             WHERE rfSum=X.rfSum and nombSum=X.nombSum and NIF=X.NIF and ciudad=X.ciudad);
DELETE FROM Suministradores WHERE rfSum='S01' or rfSum='S02';
INSERT INTO Suministradores VALUES ('S01',     'PEREZ',     'X01',    'ZARAGOZA');
INSERT INTO Suministradores VALUES ('S02',  'MARTINEZ',     'A12',      'HUESCA');

/* volver a dejar las restricciones como estaban
*/
ALTER TABLE Suministradores add (CONSTRAINT PK_Sumin  PRIMARY KEY (rfSum));
ALTER TABLE Suministradores add (CONSTRAINT UN_NIF_S  UNIQUE(NIF));
ALTER TABLE suminpiezas     add (CONSTRAINT FK_SP_Sum FOREIGN KEY (rfSum)
                                                      REFERENCES Suministradores);
```

ejemplo_01.pc

```
/* Este programa pregunta interactivamente un numero de pieza (clvPieza), y después consulta
   en la tabla Pieza para obtener la información de la pieza.  Utiliza una variable indicador,
   asociada al resultado (registro) de la consulta, para determinar si el color es NULO.
                Santiago Velilla, 20-May-2001, Informática e Ingeniería de Sistemas, Universidad Zaragoza.
 */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <sqlda.h>
#include <sqlcpr.h>

/* Definir constantes para las longitudes de los VARCHAR utilizados */
#define     UNAME_LEN      20
#define     PWD_LEN        40
#define     NOMBRE_LEN     32

/* Declaracion de variables.  No es necesaria la sección declare si MODE=ORACLE. */
VARCHAR     username[UNAME_LEN];  /* VARCHAR es un estructura suministrada por Oracle */
varchar     password[PWD_LEN];    /* varchar es válido también en minúsculas */

/* Definir un registro para los valores devueltos (piezas) por la sentencia SELECT */
struct {
    VARCHAR   nombrePieza[NOMBRE_LEN];
    int       clavePieza;
    VARCHAR   colorPieza[NOMBRE_LEN];
} regPieza;

/* Definir un registro de indicadores, correspondiente al registro anterior, para detectar la
   presencia de valores nulos.  Es necesario si pueden aparecer NULOS. */
struct {
    short     nombre_ind;
    short     clave_ind;
    short     color_ind;
} regPieza_ind;

/*  variables locales del programa */
int         clavePieza;
int         total_consultas;

/* Incluir el área de comunicacion de SQL. Se puede utilizar #include ò EXEC SQL INCLUDE */
#include <sqlca.h>

/* Declaración del procedimiento definido para el tratamiento de errores */
void sql_error(char *msg)
{
    char    err_msg[128];
    size_t  buf_len, msg_len;

    EXEC SQL WHENEVER SQLERROR CONTINUE;

    printf("\n%s\n", msg);
    buf_len = sizeof (err_msg);
    sqlglm(err_msg, &buf_len, &msg_len);
    printf("%.*s\n", msg_len, err_msg);

    EXEC SQL ROLLBACK RELEASE;
    exit(EXIT_FAILURE);
}
```

```
void main()  {    /*  PROGRAMA  PRINCIPAL  */

    char  temp_char[32];

/* Preparar la conexión a ORACLE: Copiar el username en el VARCHAR, y añadir su longitud
 */
    strncpy((char *) username.arr, "SCOTT", UNAME_LEN);
    username.len = (unsigned short) strlen((char *) username.arr);

/* Copiar el password y añadir su longitud */
    strncpy((char *) password.arr, "TIGER", PWD_LEN);
    password.len = (unsigned short) strlen((char *) password.arr);

/* Especificar el procedimiento sql_error() como gestor de errores */
    EXEC SQL WHENEVER SQLERROR do sql_error("ORACLE error--\n");

/* Conexion a ORACLE.  El programa llamara a sql_error() si se produce un error en la
   conexion a la Base de Datos con los valores especificados
 */
    EXEC SQL CONNECT :username IDENTIFIED BY :password;

    printf("\nConectado a ORACLE como usuario: %s\n", username.arr);

/* Bucle para selección individual de la informacion de las Piezas */
    total_consultas = 0;
    for (;;) {
        clavePieza = 0;
        printf("\nIntroduzca la clave de la Pieza (0 --> FIN): ");
        gets(temp_char);
        clavePieza = atoi(temp_char);
        if (clavePieza == 0)  break;
/* Ir a noEncontrada cuando se produzca la condición de error ("No data found" = 1403)
 */
        EXEC SQL WHENEVER NOT FOUND GOTO noEncontrada;
        EXEC SQL SELECT nombPieza, clvPieza, color
            INTO :regPieza INDICATOR :regPieza_ind
            FROM Pieza
            WHERE clvPieza = :clavePieza;

/* Escribir los datos de la Pieza */
        printf("\nnombre                              clave   color\n");
        printf("--------------------------------   -----   ---------------\n");
/* añadir el terminador(nulo) a las cadenas (nombrePieza y colorPieza) */
        regPieza.nombrePieza.arr[regPieza.nombrePieza.len] = '\0';
        regPieza.colorPieza.arr[regPieza.colorPieza.len] = '\0';
        printf("%s    %3d    ", regPieza.nombrePieza.arr, regPieza.clavePieza);

        if (regPieza_ind.color_ind == -1)  printf("  ??\n");
        else printf("%s\n", regPieza.colorPieza.arr);
        total_consultas++;
        continue;

noEncontrada:
        printf("\nNo es una clave valida - inténtelo otra vez.\n");
    } /* end for(;;) */
    printf("\n\nTotal filas consultadas %d.\n", total_consultas);
    printf("\nAdios.\n\n\n");

/* Desconexión de ORACLE deshaciendo las transacciones pendientes (si las hubiera) */
    EXEC SQL ROLLBACK WORK RELEASE;
    exit(EXIT_SUCCESS);
}
```

ejemplo_02.pc

```
/*  Este programa visualiza los datos (clvPieza, nombPieza y color) de las piezas existentes.
    Para ello, se conecta al SGBD y mediante un CURSOR recorre la tabla Pieza.
    Ilustra el modo de utilizar un CURSOR: 1) Declaración, 2) Iniciar (open), 3) Iteración con
    sentencia FETCH .. INTO y, finalmente 4) Cerrar el cursor. Además deberá especificarse una
    condición de terminación cuando no se encuentren más tuplas, normalmente con WHENEVER.
    Obsérvese que, puesto que el color puede tomar el valor nulo, es necesario definir una
    variable indicador para obtener información del atributo correspondiente (si no, la apari-
    ción de un valor nulo provoca un error en la consulta)
                Santiago Velilla, 20-May-2001, Informática e Ingeniería de Sistemas, Universidad Zaragoza.
*/

#include <stdio.h>
#include <string.h>
#include <sqlca.h>
#include <stdlib.h>
#include <sqlda.h>
#include <sqlcpr.h>

#define     UNAME_LEN      30      /* longitud máxima del nombre de usuario */
#define     PWD_LEN        20      /* longitud máxima de la clave de acceso */

/* Se utiliza la capacidad de definición de tipos del precompilador para crear string con
   caracter terminador nulo. Aunque no sería necesario (bastaría con char *), se hace así
   para ilustrar dicha capacidad (en otros ejemplos se hace de manera diferente).
*/
typedef char asciiz[PWD_LEN];

EXEC SQL type asciiz is charz(UNAME_LEN) reference;
asciiz     username;
asciiz     password;

/* Definir un registro para los valores devueltos por la consulta a la Base de Datos */
struct regPieza {
    asciiz    nombrePieza;
    int       clavePieza;
    asciiz    colorPieza;
};

/* Definir un registro de indicadores correspondiente al registro anterior */
struct {
    short     nombre_ind;
    short     clave_ind;
    short     color_ind;
} regPieza_ind;

/* Declaración del procedimiento definido para el tratamiento de errores */
void sql_error(char *msg)
{
    char    err_msg[512];
    size_t  buf_len, msg_len;

    EXEC SQL WHENEVER SQLERROR CONTINUE;

    printf("\n%s\n", msg);

/*  Ejecutar sqlglm() para obtener el texto completo del mensaje de error
*/
    buf_len = sizeof (err_msg);
    sqlglm(err_msg, &buf_len, &msg_len);
    printf("%.*s\n", msg_len, err_msg);

    EXEC SQL ROLLBACK release;
    exit(EXIT_FAILURE);
}
```

```
void main() {     /* PROGRAMA PRINCIPAL */

/*  Variables locales del programa */
    int    total_Piezas;
    struct regPieza *regPieza_ptr;

/* Reservar memoria para el dato de tipo regPieza (es dinámico) */
    if ((regPieza_ptr = (struct regPieza *) malloc(sizeof(struct regPieza))) == 0) {
      fprintf(stderr, "error en la reserva de memoria.\n");
      exit(EXIT_FAILURE);
    }

/* Preparar el nombre y password para acceder al gestor
 */
    strcpy(username, "SCOTT");
    strcpy(password, "TIGER");

/* Especificar el procedimiento sql_error() como gestor de errores */
    EXEC SQL WHENEVER SQLERROR do sql_error("ORACLE error--");

/* Conectar con el SGBD ORACLE
 */
    EXEC SQL CONNECT :username IDENTIFIED BY :password;
    printf("\nconectado a ORACLE como usuario: %s\n", username);

/* Declaración del cursor. Todos los cursores estáticos y explícitos de SQL contienen
 * sentencias SELECT. 'listaPiezas' es un identificador SQL, no una variable C.
 */
    EXEC SQL declare listaPiezas CURSOR FOR
           SELECT nombPieza, clvPieza, color
           FROM Pieza;

/* Iniciar (abrir) el cursor */
    EXEC SQL OPEN listaPiezas;

/* Mostrar la cabecera del listado */
    printf("\n\n  listado de Piezas disponibles\n\n");
    printf(" tipo Pieza    ref.   color Pieza\n");
    printf("------------   ----   -----------\n");

/* Construir un bucle con una sentencia FETCH con el CURSOR listaPiezas para obtener
 * la información de las piezas. Especificar que el bucle termina (break) cuando se
 * han recorrido todas las tuplas de la tabla Pieza (es decir, se produce el error de
 * "tupla no encontrada" al ejecutar la sentencia FETCH)
 */
    EXEC SQL WHENEVER not found DO break;

    total_Piezas = 0;
    for (;;) {
       EXEC SQL FETCH listaPiezas INTO :regPieza_ptr INDICATOR :regPieza_ind;

       printf(" %s %4d     ", regPieza_ptr->nombrePieza, regPieza_ptr->clavePieza);

       if (regPieza_ind.color_ind == -1)  printf("  ??\n");
       else  printf("%s\n", regPieza_ptr->colorPieza);

       total_Piezas++;
    }

/* Cerrar el cursor */
    EXEC SQL CLOSE listaPiezas;

    printf("\n\ntotal de piezas %d.\n", total_Piezas);
    printf("\nAdios.\n\n\n");

    EXEC SQL COMMIT WORK RELEASE;
    exit(EXIT_SUCCESS);
}
```

SQLemb_UZ.pc

```
/* ------------------------------------------------------
    EJEMPLO DE APLICACION CON ACCESO AL S.G.B.D.R. ORACLE
   ------------------------------------------------------ */

#INCLUDE <stdio.h>                    /* Declaración de bibliotecas */
#INCLUDE <string.h>
#INCLUDE <ctype.h>

#define LeerULong(num)   scanf("%lu", num);
#define LeerCadena(cad,max)   scanf("%"#max"[^\n]", cad);
#define LeerLinea while(getchar() != '\n');
#define LeerLinCadena(cad,max) LeerCadena(cad,max); LeerLinea;
#define LeerLinULong(num)   LeerULong(num);  LeerLinea;

EXEC SQL BEGIN DECLARE SECTION;

VARCHAR uid[30];                      /* nombre de usuario       */
VARCHAR pwd[20];                      /* password                */

int     NumProfesores;                /* Variables de trabajo    */

int     ClaveProfesor;
VARCHAR CodigoProfesor[10];
VARCHAR NombreProfesor[30];
int     AreaProfesor;

short   ind_clvProf;
short   ind_codProf;
short   ind_nombProf;
short   ind_areaProf;

int l;                                /* Variables auxiliares    */

EXEC SQL END DECLARE SECTION;
EXEC SQL INCLUDE SQLCA;

int SelecOpcionMenu (void)
{
  int eleccion;
  char tecla;

  do {
    printf ("\n\n MENU PRINCIPAL:\n\n");
    printf ("\t 1. Dar de alta a un nuevo profesor \n");
    printf ("\t 2. Listado de profesores \n");
    printf ("\t 3. Opción 3 \n");
    printf ("\n\t 0. Salir\n");
    printf ("\n\t Introduce opción: ");
    tecla = getchar();
    printf ("\n\n");
  }while ((tecla < '0') || (tecla > '3'));
  LeerLinea;
  return (tecla - '0');
}
```

```
void main()
{
  int opcion;

/* --------------------
     Acceso a Oracle.
   -------------------- */
  strcpy((char *) uid.arr, "SCOTT");
  uid.len = (unsigned short) strlen((char *) uid.arr);
  strcpy((char *) pwd.arr, "TIGER");
  pwd.len = (unsigned short) strlen((char *) pwd.arr);

  EXEC SQL WHENEVER SQLERROR goto SalidaPorError;
  EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;

  printf ("\n conectado como : %s", uid.arr);

  /* PROGRAMA PRINCIPAL */

  while ((opcion=SelecOpcionMenu()) != 0)         /* Menú principal */
   switch (opcion) {
    case 1:  /*   Inserción de un profesor  */

     l= asks(" Introduce nombre del profesor: ", NombreProfesor.arr);
     if (l <= 0) break;
     printf ("profesor: %s", NombreProfesor.arr);
     NombreProfesor.len = l;
     askn(" Introduce clave del profesor: ", &ClaveProfesor);

     EXEC SQL WHENEVER NOT FOUND STOP;
     EXEC SQL INSERT INTO PROFESOR(clvProf,nombProf)
                VALUES (:ClaveProfesor, :NombreProfesor);

     printf("  \n%s Añadido a la tabla PROFESOR \n", NombreProfesor.arr);
     continue;

    case 2:  /*   Listado de  profesores  */

      EXEC SQL DECLARE ListaProfesores CURSOR for
                SELECT clvProf, codProf, nombProf, clvArea
                FROM PROFESOR
                ORDER BY clvProf;

      EXEC SQL OPEN ListaProfesores;
      EXEC SQL WHENEVER NOT FOUND goto FinListado;
      printf("\n\nCLAVE  CODIGO     NOMBRE                          AREA\n");
      printf("------ ---------- ------------------------------ ------\n");
      for (;;) {
        EXEC SQL FETCH ListaProfesores INTO :ClaveProfesor:ind_clvProf,
                                            :CodigoProfesor:ind_codProf,
                                            :NombreProfesor:ind_nombProf,
                                            :AreaProfesor:ind_areaProf;

        CodigoProfesor.arr[CodigoProfesor.len]= '\0';
        NombreProfesor.arr[NombreProfesor.len]= '\0';
        printf("%-6d %-10s ", ClaveProfesor, CodigoProfesor.arr);
        printf("%-30s %-6d\n", NombreProfesor.arr, AreaProfesor);
      }

    FinListado:
      EXEC SQL CLOSE ListaProfesores;
      continue;

    case 3: /*   Opciones por implementar  */
      printf ("\t Opción no implementada \n");
      continue;

    case 0: break;
  }
```

```
/* -------------------
     Salir de Oracle
   ------------------- */

   EXEC SQL COMMIT WORK RELEASE;
   return;

SalidaPorError:
   errrpt();
   EXEC SQL WHENEVER SQLERROR continue;
   EXEC SQL ROLLBACK WORK RELEASE;

  return;
}

                    /* Funciones auxiliares */
int askn (char *text, int *variable)
{
   char s[20];
   printf(text);
   fflush(stdout);
   if ( gets(s) == (char *)0 ) return(EOF);

   *variable = atoi(s);
   return(1);
}

int asks (char *text, char *variable)
{
   printf(text);
   fflush(stdout);
   return( gets(variable) == (char *)0 ? EOF : strlen(variable) );
}

/* ---------------------------------------------------------
   VOID errrpt()  Muestra el mensaje y código de error Oracle
   --------------------------------------------------------- */

errrpt()
{
   printf("%.70s (%d)\n", SQLCA.SQLERRM.sqlerrmc, -SQLCA.SQLCODE);
   return(0);
}
```

SQLemb_UZ_1.pc

```
/* ------------------------------------------------------
    EJEMPLO DE APLICACION CON ACCESO AL S.G.B.D.R. ORACLE
   ------------------------------------------------------ */

#INCLUDE <stdio.h>                    /* Declaración de bibliotecas */
#INCLUDE <string.h>
#INCLUDE <ctype.h>

#define LeerULong(num)   scanf("%lu", num);
#define LeerCadena(cad,max)   scanf("%"#max"[^\n]", cad);
#define LeerLinea while(getchar() != '\n');
#define LeerLinCadena(cad,max) LeerCadena(cad,max); LeerLinea;
#define LeerLinULong(num)   LeerULong(num);  LeerLinea;
#define preguntarVARCHAR(msg,dest,max) _preguntarVARCHAR(msg,(VARCHAR *)dest, max)

EXEC SQL BEGIN DECLARE SECTION;

VARCHAR uid[30];                      /* nombre de usuario       */
VARCHAR pwd[20];                      /* password                */

int     ClaveProfesor;
VARCHAR CodigoProfesor[10];
VARCHAR NombreProfesor[30];
int     AreaProfesor;

short   ind_clvProf;
short   ind_codProf;
short   ind_nombProf;
short   ind_areaProf;
char    nombre[30];

EXEC SQL END DECLARE SECTION;

int     totChar;                      /* variable auxiliar       */

EXEC SQL INCLUDE SQLCA;

/* PROTOTIPOS de las funciones utilizadas  */
void strToVARCHAR (char *orig, VARCHAR *dest);
int  _preguntarVARCHAR (char * msg, VARCHAR *dest, short max);
int   preguntarSTRING  (char *msg,  char    *dest, short max);

int SelecOpcionMenu (void)
{
  int eleccion;
  char tecla;

  do {
    printf ("\n\n MENU PRINCIPAL:\n\n");
    printf ("\t 1. Dar de alta a un nuevo profesor \n");
    printf ("\t 2. Listado de profesores \n");
    printf ("\t 3. Dar de baja a un profesor \n");
    printf ("\t 4. Opción 4 \n");
    printf ("\n\t 0. Salir\n");
    printf ("\n\t Introduce opción: ");
    tecla = getchar();
    printf ("\n\n");
  }while ((tecla < '0') || (tecla > '4'));
  LeerLinea;
  return (tecla - '0');
}

void main()
{
  int opcion;

  preguntarVARCHAR("nombre de Usuario: ", &uid, 30);       /*  Acceso a Oracle  */
  preguntarVARCHAR("clave de acceso  : ", &pwd, 20);

  EXEC SQL WHENEVER SQLERROR goto SalidaPorError;
  EXEC SQL CONNECT :uid IDENTIFIED BY :pwd;

  printf ("\n conectado como : %s", uid.arr);
```

```
  /* PROGRAMA PRINCIPAL */

  while ((opcion=SelecOpcionMenu()) != 0)         /* Menú principal */
   switch (opcion) {
    case 1:  /*   Inserción de un profesor  */

     totChar = asks(" Introduce nombre del profesor: ", NombreProfesor.arr);
     if (totChar <= 0) break;
      NombreProfesor.len = totChar;
      askn(" Introduce clave del profesor : ", &ClaveProfesor);

     EXEC SQL WHENEVER NOT FOUND STOP;
     EXEC SQL INSERT INTO PROFESOR(clvProf,nombProf)
                VALUES (:ClaveProfesor, :NombreProfesor);

     printf("  \n%s Añadido a la tabla PROFESOR \n", NombreProfesor.arr);

     continue;

    case 2:  /*   Listado de  profesores  */

      EXEC SQL DECLARE ListaProfesores CURSOR for
                SELECT clvProf, codProf, nombProf, clvArea
                FROM PROFESOR
                ORDER BY clvProf;

      EXEC SQL OPEN ListaProfesores;
      EXEC SQL WHENEVER NOT FOUND goto FinListado;

      printf("\n\nCLAVE  CODIGO     NOMBRE                          AREA\n");
      printf("------ ---------- ------------------------------ ------\n");
      for (;;) {
        EXEC SQL FETCH ListaProfesores INTO :ClaveProfesor:ind_clvProf,
                                            :CodigoProfesor:ind_codProf,
                                            :NombreProfesor:ind_nombProf,
                                            :AreaProfesor:ind_areaProf;

        CodigoProfesor.arr[CodigoProfesor.len]= '\0';
        NombreProfesor.arr[NombreProfesor.len]= '\0';
        printf("%-6d %-10s ", ClaveProfesor, CodigoProfesor.arr);
        printf("%-30s %-6d\n", NombreProfesor.arr, AreaProfesor);
      }

    FinListado:
      EXEC SQL CLOSE ListaProfesores;
      continue;

    case 3:  /*   Eliminación de un profesor por su nombre */

     preguntarSTRING("Nombre del profesor: ", nombre, 30);
     if (strlen(nombre) == 0) break;

     EXEC SQL WHENEVER NOT FOUND goto FinEliminar;
     EXEC SQL DELETE FROM PROFESOR
                WHERE UPPER(nombProf)=UPPER(:nombre);

     printf("\n %s ELIMINADO de la tabla PROFESOR \n", nombre);
     continue;

    FinEliminar:
     printf("\n %s NO ENCONTRADO en la tabla PROFESOR \n", nombre);
     continue;

    case 4: /*   Opciones por implementar  */
      printf ("\t Opción no implementada \n");
      continue;

    case 0: break;
  }


/*   Salida normal de la aplicación: desconexión de Oracle
```

```
*/
   EXEC SQL COMMIT WORK RELEASE;
   return;

/*   Salida anormal: deshacer transacciones pendientes y desconexión de Oracle
*/
SalidaPorError:
   errrpt();
   EXEC SQL WHENEVER SQLERROR continue;
   EXEC SQL ROLLBACK WORK RELEASE;
  return;
}
```

/* Funciones auxiliares */

```
int askn (char *text, int *variable) {
   char s[20];
   printf(text);
   fflush(stdout);
   if ( gets(s) == (char *)0 ) return(EOF);
   *variable = atoi(s);
   return(1);
}
```

```
int asks (char *text, char *variable) {
   printf(text);
   fflush(stdout);
   return( gets(variable) == (char *)0 ? EOF : strlen(variable) );
}
```

```
void strToVARCHAR (char *orig, VARCHAR *dest) {
  strcpy((char *) dest->arr, orig);
  dest->len = (unsigned short) strlen(orig);
}
```

```
int _preguntarVARCHAR (char *msg, VARCHAR *dest, short max) {
  int i;
  char *pt;
  printf(msg);   fflush(stdout);
  pt = (char *) &dest->arr;
  for(i=0; i<max; i++) {
    if ('\n' == (*pt = getchar())) break;
    pt++;
  }
  *pt = '\0';
  dest->len = (unsigned short) strlen((char *) dest->arr);
  return(dest->len);
}
```

```
int  preguntarSTRING  (char *msg,  char    *dest, short max) {
  int i;
  printf(msg);   fflush(stdout);
  for(i=0; i<max; i++) {
    if ('\n' == (*dest = getchar())) break;
    dest++;
  }
  *dest = '\0';
  return(i);
}
```

```
/* Mostrar el mensaje y código de error Oracle  */
errrpt()
{
   printf("%.70s (%d)\n", SQLCA.SQLERRM.sqlerrmc, -SQLCA.SQLCODE);
   return(0);
}
```