

***Actividad 1: En el cuaderno de prácticas realizar los ejercicios 3.71 a 3.75 referentes a la base de datos de jugadores de baloncesto:***

- SQL:

**3.71) Para cada equipo muestra el número de encuentros que ha disputado como local.**

```
select nombreE, count(*) from equipos, encuentros where code=elocal group by nombreE order by nombreE;
```

**3.72) Muestra los encuentros en los que se alcanzó mayor diferencia de puntos:**

```
select equipos.nombreE, eq.nombreE, abs(PLocal - PVisitante) from equipos, equipos eq, encuentros where equipos.codE=ELocal and eq.codE=EVisitante and abs(PLocal - PVisitante)=(select max(abs(encuentros.Plocal-encuentros.PVisitante)) from encuentros);
```

**3.73) Muestra los jugadores que no han superado 3 faltas acumuladas:**

```
select nombreJ, sum(faltas) from jugadores, alineaciones where alineaciones.codJ=jugadores.codJ group by nombreJ having sum(faltas)<=3;
```

**3.74) Muestra los equipos con mayores puntuaciones en los partidos jugados fuera de casa:**

```
select EVisitante, PVisitante from encuentros where PVisitante=(select max(encuentros.PVisitante) from encuentros);
```

**3.75) Muestra todas las victorias de cada equipo, jugando como local o como visitante:**

```
select nombreE, codE, ELocal, EVisitante from equipos, encuentros where (codE=ELocal and PLocal>PVisitante) or (codE=EVisitante and PVisitante>PLocal) order by nombreE;
```

**Actividad 2: En el cuaderno de prácticas realizar los ejercicios 4.1, 4.2 y 4.3:**

**4.1) Crear una vista con los proveedores de Londres. ¿Qué sucede si insertamos en dicha vista la tupla ('S7', 'Jose Suarez', 3, 'Granada')?**

```
create view vista1 as select * from proveedor where ciudad='Londres';

insert into vista1 values ('S7','Jose Suarez',3,'Granada');
```

Salida de Script x

Tarea terminada en 0,9 segundos

View VISTA1 creado.

1 fila insertadas.

Se inserta correctamente en la tabla proveedor a través de la vista1 dado que no existía ninguna tupla con la misma llave primaria en la tabla proveedor, pero como es lógico, no muestra la tupla en la tabla de la vista1 dado que no es de Londres.

OracleETSII x PROVEEDOR					OracleETSII x VISTA1				
Columnas Datos Model Restricciones Permisos Estadísticas					Columnas Datos Permisos Dependencias Detalles Dispa				
Ordenar... Filtrar:					Ordenar... Filtrar:				
	CODPRO	NOMPRO	STATUS	CIUDAD		CODPRO	NOMPRO	STATUS	CIUDAD
1	S7	Jose Suarez		3 Granada	1	S2	Manuel Vidal		1 Londres
2	S1	Jose Fernandez		2 Madrid					
3	S2	Manuel Vidal		1 Londres					
4	S3	Luisa Gomez		3 Lisboa					
5	S4	Pedro Sanchez		4 Paris					
6	S5	Maria Reyes		5 Roma					

**4.2) Crear una vista con los nombres de los proveedores y sus ciudades. Inserta sobre ella una fila y explica cuál es el problema que se plantea. ¿Habría problemas de actualización?**

```
create view vista2 as select nompro, ciudad from proveedor;

insert into vista2 values('Antonio Lopez', 'Madrid');
```

Salida de Script x

Tarea terminada en 1,09 segundos

View VISTA2 creado.

Error que empieza en la línea: 3 del comando :

```
insert into vista2 values('Antonio Lopez', 'Madrid')
```

Informe de error -

ORA-01400: no se puede realizar una inserción NULL en ("X0108849"."PROVEEDOR"."CODPRO")

No se puede hacer una inserción sin la clave primaria del proveedor.

**4.3) Crear una vista donde aparezcan el código del proveedor, el nombre de proveedor y el código del proyecto tales que la pieza suministrada sea gris. Sobre esta vista realiza alguna consulta y enumera todos los motivos por los que sería imposible realizar una inserción.**

```
create view vista3 as select proveedor.codpro, nompro, codpj from proveedor,ventas,pieza where
proveedor.codpro=ventas.codpro and pieza.codpie=ventas.codpie and pieza.color='Gris';

select * from vista3;
```

Salida de Script x | Tarea terminada en 1,038 segundos

View VISTA3 creado.

COD	NOMPRO	COD
S1	Jose Fernandez	J1
S1	Jose Fernandez	J3
S1	Jose Fernandez	J4
S1	Jose Fernandez	J1
S1	Jose Fernandez	J3
S1	Jose Fernandez	J2
S3	Luisa Gomez	J4
S3	Luisa Gomez	J1
S4	Pedro Sanchez	J3
S5	Maria Reyes	J4

10 filas seleccionadas.

Sería imposible realizar una inserción debido a que la vista no está construida sobre una única tabla y no contiene todas las llaves primarias de las tablas que la conforman.

### ***Actividad 3: Realizar los ejercicios correspondientes a índices BITMAP e IOT que aparecen descritos en el cuaderno de prácticas:***

#### **- Índices BITMAP:**

Cuando los valores de la clave tienen una baja cardinalidad, el uso de este tipo de índice puede ser apropiado. Se usa la cláusula BITMAP para indicar que son de este tipo de índice. Ejemplo de uso frente a índice normal:

1. Crear tabla ejemplo: CREATE TABLE Prueba\_Bit (color Varchar2(10));
2. Insertar 50000 tuplas con valores de color aleatorios. Ejecutar:
 

```
BEGIN
FOR i IN 1..50000 LOOP
INSERT INTO Prueba_bit (
select decode(round(dbms_random.value(1,4)),1,'Rojo',2,'Verde',
3,'Amarillo',4,'Azul') from dual);
END LOOP;
END;
```
3. Crear un índice normal: CREATE INDEX Prueba\_IDX ON Prueba\_Bit(color);
4. Ejecutar:
 

```
SELECT count(*) FROM Prueba_Bit
WHERE color='Amarillo' OR color='Azul';
```
5. Apuntar el tiempo empleado.
6. Pinchar el botón "Explicación del Plan" (F10) para ver el plan seleccionado para ejecutar esa consulta.
7. Borrar índice normal: DROP INDEX Prueba\_IDX;
8. Crear índice BITMAP:
 

```
CREATE BITMAP INDEX Prueba_BITMAP_IDX ON Prueba_Bit(color);
```
9. Volver a ejecutar la consulta anterior.
10. Apuntar el tiempo empleado y comparar con el anterior.
11. Pinchar el botón "Explicación del Plan" (F10) para ver el plan seleccionado para ejecutar esa consulta. Comparar con el plan usado por la consulta con el índice normal.
12. Borrar tabla e índice: DROP TABLE Prueba\_bit;

Hoja de Trabajo    Generador de Consultas

```

CREATE TABLE Prueba_Bit (color Varchar2(10));

BEGIN
  FOR i IN 1..50000 LOOP
    INSERT INTO Prueba_Bit(
      select decode(round(dbms_random.value(1,4)),1,'Rojo',2,'Verde',3,'Amarillo',4,'Azul') from dual);
    END LOOP;
  END;

CREATE INDEX Prueba_IDX ON Prueba_Bit(color);

SELECT count(*) FROM Prueba_Bit WHERE color='Amarillo' OR color='Azul';

```

Salida de Script x    Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 en 0,136 segundos

	COUNT(*)
1	74936

Salida de Script x    Resultado de la Consulta x    Explicación del Plan x

SQL | 0,608 segundos

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT			1	3
SORT		AGGREGATE	1	
INLIST ITERATOR				
INDEX	PRUEBA_IDX	RANGE SCAN	80043	3
Access Predicates				
OR				
COLOR='Amarillo'				
COLOR='Azul'				
Other XML				
{info}				
info type="db_version"				
12.1.0.2				
info type="parse_schema"				
"X0108849"				
info type="dynamic_sampling" note="y"				
2				
info type="plan_hash_full"				
3912808686				
info type="plan_hash"				
4138648906				
info type="plan_hash_2"				
3912808686				
{hint}				
INDEX(@"SEL\$1" "PRUEBA_BIT"@"SEL\$1" ("PRUEBA_BIT"."COLOR"))				
OUTLINE_LEAF(@"SEL\$1")				
ALL_ROWS				
DB_VERSION("12.1.0.2")				
OPTIMIZER_FEATURES_ENABLE("12.1.0.2")				
IGNORE_OPTIM_EMBEDDED_HINTS				

Hoja de Trabajo    Generador de Consultas

```

DROP INDEX Prueba_IDX;

CREATE BITMAP INDEX Prueba_BITMAP_IDX ON Prueba_Bit(color);

SELECT count(*) FROM Prueba Bit WHERE color='Amarillo' OR color='Azul';

```

Salida de Script x    Resultado de la Consulta x

SQL | Todas las Filas Recuperadas: 1 en 0,168 segundos

	COUNT(*)
1	74936

The screenshot shows the Oracle SQL Developer interface. The top pane displays the execution plan for a query. The bottom pane shows the command window with the command `DROP TABLE Prueba_Bit;` and the status bar indicating 'Tarea terminada'.

OPERATION	OBJECT_NAME	OPTIONS	CARDINALITY	COST
SELECT STATEMENT				6
SORT				1
INLIST ITERATOR		AGGREGATE		1
BITMAP CONVERSION				
BITMAP INDEX	PRUEBA_BITMAP_IDX	COUNT	80043	6
Access Predicates		SINGLE VALUE		
OR				
COLOR='Amarillo'				
COLOR='Azul'				

Other XML:

```

{info}
  info type="db_version"
  12.1.0.2
  info type="parse_schema"
  "X0108849"
  info type="dynamic_sampling" note="y"
  2
  info type="plan_hash_full"
  2371678419
  info type="plan_hash"
  457884405
  info type="plan_hash_2"
  2371678419
  {hint}
    INDEX_COMBINE(@"SEL$1" "PRUEBA_BIT"@"SEL$1" ("PRUEBA_BIT"."COLOR"))
    OUTLINE_LEAF(@"SEL$1")
    ALL_ROWS
    DB_VERSION("12.1.0.2")
    OPTIMIZER_FEATURES_ENABLE("12.1.0.2")
    IGNORE_OPTIM_EMBEDDED_HINTS
  
```

Hoja de Trabajo    Generador de

`DROP TABLE Prueba_Bit;`

Salida de Script x

Tarea terminada

Table PRUEBA\_BIT borrado.

## - Índices IOT:

Este tipo de tablas organizan sus tuplas según el valor de la clave utilizando una estructura de árbol B\*. La diferencia respecto a un índice normal estriba en que en las hojas están las tuplas, no los RID que apuntan a las tuplas. Para crear estas tablas se usa la cláusula `ORGANIZATION INDEX` en la sentencia `CREATE TABLE`, normalmente utilizan los campos de la clave primaria para ordenar la tabla. Una recuperación total devuelve los resultados ordenados por la clave. El siguiente ejemplo ilustra su uso:

1. Crear la tabla:  
`CREATE TABLE Prueba_IOT (id NUMBER PRIMARY KEY) ORGANIZATION INDEX;`
2. Carga de 2000 tuplas con id entre 1 y 2000, *ordenadas de forma aleatoria*. Abrir y ejecutar el script SQL: `Carga_prueba_iot.sql`.
3. Ejecutar la consulta: `SELECT id FROM Prueba_IOT`. Observar el orden en que se obtienen las tuplas en relación al campo id.
4. Pinchar el botón "Explicación del Plan" (F10) para ver el plan seleccionado para ejecutar esa consulta.
5. Borrar tabla: `DROP TABLE Prueba_IOT;`

```
CREATE TABLE Prueba_IOT (id NUMBER PRIMARY KEY) ORGANIZATION INDEX;

@C:\Users\Josele\Desktop\Carga_prueba_iot.sql

SELECT id FROM Prueba_IOT;
```

Salida de Script x Resultado de la Consulta x	
SQL   Todas las Filas Recuperadas: 2000 en 4,713 segundos	
ID	
1960	420
1967	429
1968	430
1969	431
1970	432
1971	433
1972	434
1973	435
1974	436
1975	437
1976	438
1977	439
1978	440
1979	441
1980	442
1981	443
1982	444
1983	445
1984	446
1985	447
1986	448
1987	449
1988	450
1989	451
1990	452
1991	453
1992	454
1993	455
1994	456
1995	457
1996	458
1997	459
1998	460
1999	461
2000	462

Hoja de Trabajo    Generador de

DROP TABLE Prueba\_IOT;

Salida de Script x

Tarea terminada

Table PRUEBA\_IOT borrado.

**Actividad 4: En la relación de problemas Ejercicios de Álgebra Relacional resolver en Álgebra relacional y SQL el ejercicio 5:**

⑤ Representante ( ANI, nombre, direc, provincia )  
 Zona-Rep ( ANI, cod-zona, población, provincia )  
 Pedidos ( ANI, cod-art, cantidad, población )  
 Articulo ( cod-art, nombre, color, prov-fab )

a) Listar las provincias que son visitadas por todos los representantes.

\* AR:

$$\pi_{ANI, provincia} (Zona-Rep) \div \pi_{ANI} (Representante)$$

\* SQL:

```
select provincia from Zona-Rep where not exists (
  select ANI from Representante
  minus
  select ANI from Zona-Rep Zona where Zona-Rep.provincia = Zona.provincia);
```

o

```
select provincia from Zona-Rep where not exists (
  select ANI from Representante where not exists (
    select * from Zona-Rep Zona where Zona.ANI = Representante.ANI and Zona-Rep.provincia = Zona.provincia))
```

b) Encontrar los representantes que vendan fuera de su provincia artículos fabricados en su provincia.

\* AR:

$$\pi_{Representante} * \left( \pi_{(Representante.ANI = Zona-Rep.ANI) \wedge (Zona-Rep.ANI = Pedidos.ANI) \wedge (Representante.provincia \neq Zona-Rep.provincia) \wedge (Representante.provincia = Articulo.prov-fab) \wedge (Pedidos.cod-art = Articulo.cod-art)} (Representante \times Zona-Rep \times Pedidos \times Articulo) \right)$$

\* SQL:

```
select Representante.* from Representante, Zona-Rep, Pedidos, Articulo where Representante.ANI = Zona-Rep.ANI and
Representante.ANI = Pedido.ANI and Pedido.cod-art = Articulo.cod-art and Representante.provincia <> Zona-Rep.provincia and
Representante.provincia = Articulo.prov-fab;
```



c) Obtener las pólizas de Granada que hayan superado los 50000 euros de facturación y quién realizó el pedido.

\* AR:

$\Pi_{ANI, Pedido, poliza} (\sigma_{(provincia = 'Granada' \wedge cantidad \geq 50000)} (Zona-Rep \bowtie Pedidos))$

\* SQL:

select ANI, Pedido, poliza from Zona-Rep, Pedidos where Zona-Rep.ANI = Pedido.ANI and Zona-Rep.poliza = Pedidos.poliza and Zona-Rep.provincia = 'Granada' and Pedido.cantidad >= 50000;

d) Mostrar las zonas que incluyen a una sola poliza.

\* AR:

$\rho(Zona-Rep) = Zon$

$\rho(Zona-Rep) = Zon$

$\Pi_{*} (Zona-Rep) - \Pi_{Zona-*} (\sigma_{(Zona.ANI = Zon.ANI) \wedge (Zona.cod-zona = Zon.cod-zona)} \wedge (Zona.poliza \neq Zon.poliza) (Zon \times Zona))$

\* SQL:

select \* from Zona-Rep where not exists (

select \* from Zona-Rep Zon where Zona-Rep.ANI = Zon.ANI and Zona-Rep.cod-zona = Zon.cod-zona and Zona-Rep.poliza <> Zon.poliza);