TEMA 8

LENGUAJE DE MANIPULACIÓN DE DATOS (DML)

Sentencia SELECT AVANZADA

CURSO 23-24

ÍNDICE

0.- INTRODUCCIÓN

SENTENCIA SELECT (hoja de ruta)

◎ **CONSULTA DE DATOS SIMPLE**:

1. Introducción a la sentencia SELECT — **TEMA 4**
2. Obtención de datos de varias tablas:
   - ◆Subconsultas.
   - ◆Inclusión de varias tablas en clausula FROM.

   — **TEMA 7 (PARTE 2)**
3. Funciones agregadas.

◎ **CONSULTA DE DATOS AVANZADA**:

1. Comparación ALL/ANY.
2. Comparación EXISTS.
3. Consultas agrupadas.
4. Concatenación con operador JOIN.

— **TEMA 8**

0.- INTRODUCCIÓN

◎ La sentencia SELECT es la sentencia más completa de SQL, por ello la vemos dividida en varios temas.

◎ En este tema veremos la CONSULTA o SELECCIÓN AVANZADA de datos:

- Sin embargo como ya sabéis manejaros con esta sentencia este tema os resulta más sencillo.

ÍNDICE

1.- COMPARACION ALL_ANY

comparacion::= *comparacion_operador* | *comparacion_like* | *comparacion_null* | *comparacion_in* | *comparacion_between* | *comparacion_all_any* | *comparacion_exists*

comparacion_all_any::= *operando* {=|>|>=|<|<=|!=|<>} {ALL|ANY} *subconsulta*

- La comparación ALL se evalúa a cierto si la comparación con el correspondiente operador se evalúa a cierto **PARA TODOS** los valores que devuelve la subconsulta.
- La comparación ANY se evalúa a cierto si la comparación con el correspondiente operador se evalúa a cierto **PARA ALGUNO** de los valores que devuelve la subconsulta.

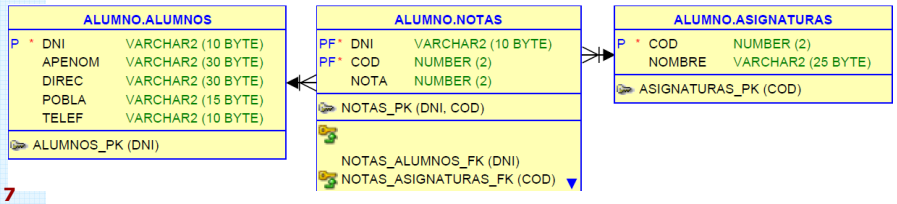
1.- COMPARACION ALL_ANY

- **EJEMPLOS:** Considerando el siguiente esquema

ASIGNATURAS (cod: varchar2 (2), nombre: varchar2 (25))
CP= {cod}

ALUMNOS (dni: varchar2 (10), apenom: varchar2 (30), direc: varchar2 (30), pobla: varchar2 (15), telef: varchar2 (10))
CP= {dni}

NOTAS (dni: varchar2 (10), cod: varchar2 (2), nota: number (2))
CP= {dni, cod}
CA= {dni} →ALUMNOS
CA= {dni} → ASIGNATURAS



1.- COMPARACION ALL_ANY

- **EJEMPLOS:** ¿Qué devuelven estas consultas?

```
SELECT NOMBRE
FROM   ASIGNATURAS
WHERE  5 = ALL  (SELECT NOTA
                 FROM   NOTAS
                 WHERE  NOTAS.COD=ASIGNATURAS.COD)
```

OBSERVAD!!!

```
SELECT NOMBRE
FROM   ASIGNATURAS
WHERE  5 = ANY  (SELECT NOTA
                 FROM   NOTAS
                 WHERE  NOTAS.COD=ASIGNATURAS.COD)
```

- **OBSERVAD:**
  - En este tipo de consultas es muy habitual ligar la consulta principal con la subconsulta

1.- COMPARACION ALL_ANY

◎ COMPARACIONES EQUIVALENTES:

operando = ANY *subconsulta*

Es equivalente a:

operando IN *subconsulta*

```
SELECT   NOMBRE
FROM     ASIGNATURAS
WHERE    5 = ANY  (SELECT   NOTA
                   FROM     NOTAS
                   WHERE    NOTAS.COD=ASIGNATURAS.COD)
```

```
SELECT   NOMBRE
FROM     ASIGNATURAS
WHERE    5 IN     (SELECT   NOTA
                   FROM     NOTAS
                   WHERE    NOTAS.COD=ASIGNATURAS.COD)
```


1.- COMPARACION ALL_ANY

◎ COMPARACIONES EQUIVALENTES:

¡CUIDADO!

- Esa equivalencia tan directa solo sirve para la igualdad:

```
SELECT   NOMBRE
FROM     ASIGNATURAS
WHERE    5 > ANY  (SELECT   NOTA
                   FROM     NOTAS
                   WHERE    NOTAS.COD=ASIGNATURAS.COD)
```

<>

```
SELECT   NOMBRE
FROM     ASIGNATURAS
WHERE    5 IN     (SELECT   NOTA
                   FROM     NOTAS
                   WHERE    NOTAS.COD=ASIGNATURAS.COD)
```


1.- COMPARACION ALL_ANY

◎ COMPARACIONES EQUIVALENTES:

- Puede hacerse más fácil y general de las formas siguientes

```
SELECT   NOMBRE
FROM     ASIGNATURAS
WHERE    5 = ANY  (SELECT   NOTA
                   FROM     NOTAS
                   WHERE    NOTAS.COD=ASIGNATURAS.COD)
```

```
SELECT   NOMBRE
FROM     ASIGNATURAS
WHERE    COD IN   (SELECT   COD
                   FROM     NOTAS
                   WHERE    NOTA=5)
```

OPERADOR IN
La subconsulta no necesita ligar con la principal…

```
SELECT   DISTINCT NOMBRE
FROM     ASIGNATURAS A, NOTAS N
WHERE    A.COD=N.COD AND
         NOTA=5
```

UN SIMPLE **FROM COMPUESTO** …
(DISTINCT es necesario)

1.- COMPARACION ALL_ANY

◎ COMPARACIONES EQUIVALENTES:

- Ahora si que vale para todos los operadores...

```
SELECT   NOMBRE
FROM     ASIGNATURAS
WHERE    5 > ANY  (SELECT   NOTA
                   FROM     NOTAS
                   WHERE    NOTAS.COD=ASIGNATURAS.COD)
```

```
SELECT   NOMBRE
FROM     ASIGNATURAS
WHERE    COD IN   (SELECT   COD
                   FROM     NOTAS
                   WHERE    NOTA>5)
```

OPERADOR IN
La subconsulta no necesita ligar con la principal…

```
SELECT   DISTINCT NOMBRE
FROM     ASIGNATURAS A, NOTAS N
WHERE    A.COD=N.COD AND
         NOTA>5
```

UN SIMPLE **FROM COMPUESTO** …
(DISTINCT es necesario)

- Yo personalmente **NO** utilizo nunca el operador ANY...

1.- COMPARACION ALL_ANY

◎ COMPARACIONES EQUIVALENTES:

- Conviene resaltar que la comparación ALL no tiene ninguna equivalencia directa con lo que conocemos hasta ahora

```
SELECT   NOMBRE
FROM     ASIGNATURAS
WHERE    5 > ALL  (SELECT   NOTA
                   FROM     NOTAS
                   WHERE    NOTAS.COD=ASIGNATURAS.COD)
```

- ¿Qué obtiene esta consulta?

  nombre de las asignaturas que solo tienen suspensos

  *(**TODAS** sus notas son menores de 5)*

- Es un tipo de comparación muy potente que si es necesario utilizar en determinadas consultas
- A este tipo de consultas se les suele denominar consultas **con cuantificación universal (∀)**

Copyright 2024 Marisa Escudero Sanchis

1.- COMPARACION ALL_ANY

◎ CONSIDERACION MUY IMPORTANTE:

- En Oracle la comparación ALL devuelve VERDADERO[1] cuando la subconsulta no devuelve registros:
  - ◆Esto se debe tener en cuenta para no obtener resultados indeseados.
- **Por ejemplo**:

```
SELECT COD
FROM   ASIGNATURAS
WHERE  5 = ALL  (SELECT NOTA
                 FROM   NOTAS
                 WHERE  NOTAS.COD=ASIGNATURAS.COD)
```

  - ◆Obtiene también el código de las asignaturas sin notas.
  - ◆Si no las deseamos debemos solucionarlo.

1: En lugar de INDEFINIDO que sería lo correcto.

Copyright 2024 Marisa Escudero Sanchis

1.- COMPARACION ALL_ANY

◎ CONSIDERACION MUY IMPORTANTE:

- Una posible solución sería:

```
SELECT DISTINCT COD
FROM   NOTAS N1
WHERE  5 = ALL  (SELECT NOTA
                 FROM   NOTAS N2
                 WHERE  N2.COD=N1.COD)
```

  - ◆Se busca sólo en las asignaturas con notas.
  - ◆No se necesita la tabla asignaturas.
- Y si se pide también el nombre de las asignaturas...

```
SELECT   A.COD, NOMBRE
FROM     ASIGNATURAS A, NOTAS N1
WHERE    A.COD=N1.COD AND
         5 = ALL  (SELECT          NOTA
                   FROM   NOTAS
                   WHERE NOTAS.COD=A.COD)
```


Copyright 2024 Marisa Escudero Sanchis

ÍNDICE

2.- COMPARACION EXISTS

```
comparacion::= comparacion_operador |
               comparacion_like |
               comparacion_null |
               comparacion_in |
               comparacion_between |
               comparacion_all_any |
               comparacion_exists
```

```
comparacion_exists::= [NOT] EXISTS subconsulta
```

- La comparación EXISTS se evalúa a cierto si la subconsulta devuelve alguna fila y a falso si la subconsulta no devuelve filas (tabla vacía)
- Es recomendable **leerlo/interpretarlo** de la forma:
  - EXISTS: existe algún@..
  - NO EXISTS: no existe ningún@...

2.- COMPARACION EXISTS

- **EJEMPLOS:** ¿Qué devuelven estas consultas?
  - Nombre de las asignaturas para las que existe alguna nota

```
SELECT  NOMBRE
FROM    ASIGNATURAS
WHERE   EXISTS      (SELECT *
                     FROM   NOTAS
                     WHERE  NOTAS.COD=ASIGNATURAS.COD)
```

OBSERVAD!!!

  - Nombre de las asignaturas para las que no existe ninguna nota

```
SELECT  NOMBRE
FROM    ASIGNATURAS
WHERE   NOT EXISTS  (SELECT *
                     FROM   NOTAS
                     WHERE  NOTAS.COD=ASIGNATURAS.COD)
```

OBSERVAD!!!

2.- COMPARACION EXISTS

- **EJEMPLOS:** ¿Qué devuelve esta consulta?

```
SELECT  NOMBRE
FROM    ASIGNATURAS
WHERE   NOT EXISTS  (SELECT *
                     FROM   NOTAS
                     WHERE  NOTAS.COD=ASIGNATURAS.COD AND
                            NOTA<5)
```


2.- COMPARACION EXISTS

- **COMPARACIONES EQUIVALENTES:**

```
EXISTS    (SELECT   *
           FROM     T
           WHERE    condicion AND X $ Y)
```

Es equivalente a:

```
X $ ANY   (SELECT   Y
           FROM     T
           WHERE    condicion)
```

```
SELECT  NOMBRE
FROM    ASIGNATURAS
WHERE   EXISTS    (SELECT   *
                   FROM     NOTAS
                   WHERE    NOTAS.COD=ASIGNATURAS.COD AND
                            NOTA=5)
```

```
SELECT  NOMBRE
FROM    ASIGNATURAS
WHERE   5 = ANY   (SELECT   NOTA
                   FROM     NOTAS
                   WHERE    NOTAS.COD=ASIGNATURAS.COD)
```

2.- COMPARACION EXISTS

◎ EQUIVALENCIAS ENTRE COMPARACIONES:

- Como sabemos, podemos hacerlo de otras formas:

```
SELECT  NOMBRE
FROM    ASIGNATURAS
WHERE   5 = ANY (SELECT  NOTA
                 FROM    NOTAS
                 WHERE   NOTAS.COD=ASIGNATURAS.COD)
```

```
SELECT  NOMBRE
FROM    ASIGNATURAS
WHERE   EXISTS (SELECT  *
                FROM    NOTAS
                WHERE   NOTAS.COD=ASIGNATURAS.COD AND
                        NOTA=5)
```

```
SELECT  NOMBRE
FROM    ASIGNATURAS
WHERE   COD IN (SELECT  COD
                FROM    NOTAS
                WHERE   NOTA=5)
```

```
SELECT  DISTINCT NOMBRE
FROM    ASIGNATURAS A, NOTAS N
WHERE   A.COD=N.COD AND
        NOTA=5
```


2.- COMPARACION EXISTS

◎ EQUIVALENCIAS ENTRE COMPARACIONES:

```
NOT EXISTS  (SELECT  *
             FROM    T
             WHERE   condicion AND NOT (X $ Y))
```

Es equivalente a:

```
X $ ALL  (SELECT  Y
          FROM    T
          WHERE   condicion)
```

```
SELECT  NOMBRE
FROM    ASIGNATURAS
WHERE   NOT EXISTS (SELECT  *
                    FROM    NOTAS
                    WHERE   NOTAS.COD=ASIGNATURAS.COD AND
                            NOTA<>5)
```

```
SELECT  NOMBRE
FROM    ASIGNATURAS
WHERE   5 = ALL (SELECT  NOTA
                 FROM    NOTAS
                 WHERE   NOTAS.COD=ASIGNATURAS.COD)
```


2.- COMPARACION EXISTS

◎ CONSIDERACIONES FINALES:

- Cuando se utiliza NOT EXISTS se debe tener en cuenta que esta comparación se evalúa a verdadero cuando la subconsulta no devuelve registros
- **Por ejemplo**:

```
SELECT  COD
FROM    ASIGNATURAS
WHERE   NOT EXISTS (SELECT  *
                    FROM    NOTAS
                    WHERE   NOTAS.COD=ASIGNATURAS.COD AND
                            NOTA<>5)
```

  - ◆Obtiene también el código de las asignaturas sin notas.
    - ❖ES CORRECTO (no tienen ninguna nota distinta de 5)
  - ◆Si no las deseamos debemos solucionarlo.

2.- COMPARACION EXISTS

◎ CONSIDERACION MUY IMPORTANTE:

- Una posible solución sería:

```
SELECT  DISTINCT COD
FROM    NOTAS N1
WHERE   NOT EXISTS (SELECT  *
                    FROM    NOTAS N2
                    WHERE   N2.COD=N1.COD AND
                            N2.NOTA<>5)
```

  - ◆Se busca sólo en las asignaturas con notas.
  - ◆No se necesita la tabla asignaturas.
- ¿Y si se pide también el nombre de las asignaturas?

ÍNDICE

3.- ALL VS EXISTS

- El predicado ALL puede resolver consultas del tipo:
  - "Obtener el numero de aquellos departamentos en los que **todos sus empleados son analistas"**

```
SELECT    DISTINCT DEPT_NO
FROM      EMPLE E1
WHERE     'ANALISTA'= ALL   (SELECT  OFICIO
                            FROM    EMPLE E2
                            WHERE   E2.DEPT_NO=E1.DEPT_NO);
```

  - "Obtener el numero de aquellos departamentos en los que **todos sus empleados tienen el mismo oficio"**

```
SELECT    DISTINCT DEPT_NO
FROM      EMPLE E1
WHERE     OFICIO = ALL  (SELECT  OFICIO
                         FROM    EMPLE E2
                         WHERE   E2.DEPT_NO=E1.DEPT_NO);
```


3.- ALL VS EXISTS

- Este tipo de consultas se conocen como **consultas con cuantificación universal (∀)**
  - En ocasiones no somos capaces de resolver una consulta de este tipo con ALL (incluso a veces no se puede)
    - En estos casos es bueno acudir al predicado EXISTS
    - Buscamos aquellos datos que NO cumplan lo contrario.
- **Por ejemplo**:
  - Obtener el número de aquellos departamentos en los que **todos sus empleados tienen el mismo oficio**

VS

  - Obtener el número de aquellos departamentos en los que **no existen empleados con distinto oficio**

3.- ALL VS EXISTS

- **Por ejemplo**:
  - Obtener el número de aquellos departamentos en los que **todos sus empleados tienen el mismo oficio**

```
SELECT    DISTINCT DEPT_NO
FROM      EMPLE E1
WHERE     OFICIO = ALL  (SELECT  OFICIO
                         FROM    EMPLE E2
                         WHERE   E2.DEPT_NO=E1.DEPT_NO);
```

  - Obtener el número de aquellos departamentos en los que **no existen empleados con distinto oficio**

```
SELECT    DISTINCT DEPT_NO
FROM      EMPLE E1
WHERE     NOT EXISTS  (SELECT   *
                       FROM     EMPLE E2
                       WHERE    E2.DEPT_NO=E1.DEPT_NO AND
                                E2.OFICIO!=E1.OFICIO);
```

3.- ALL VS EXISTS

- En realidad se ha aplicado la equivalencia entre ALL y EXISTS vista en la **diapositiva 22**:

```
X $ ALL   (SELECT   Y
           FROM     T
           WHERE    condicion)
```

```
NOT EXISTS  (SELECT   *
             FROM     T
             WHERE    condicion AND NOT (X $ Y))
```

```
SELECT   DISTINCT DEPT_NO
FROM     EMPLE E1
WHERE    OFICIO = ALL   (SELECT   OFICIO
                         FROM     EMPLE E2
                         WHERE    E2.DEPT_NO=E1.DEPT_NO);
```

```
SELECT   DISTINCT DEPT_NO
FROM     EMPLE
WHERE    NOT EXISTS    (SELECT   *
                        FROM     EMPLE E2
                        WHERE    E2.DEPT_NO=E1.DEPT_NO AND
                                 E2.OFICIO!=E1.OFICIO);
```


ÍNDICE

4.- CONSULTAS AGRUPADAS

- **SENTENCIA SELECT**: Sintaxis para consultas agrupadas

```
SELECT [ALL|DISTINCT]
       [comalista_ref_columna | *]
FROM   comalista_ref_tabla
[WHERE condicion]
[GROUP BY comalista_ref_columna
[HAVING condicion]]
[ORDER BY comalista_ref_columna [DESC|ASC]]
```

- GROUP BY: define en la tabla generada por las cláusulas FROM y WHERE grupos de filas de forma que cada grupo tiene el mismo valor en comalista_*ref_columna*
  - Un GRUPO es pues un conjunto de filas con el mismo valor para el conjunto de referencias de columna incluidas en la cláusula GROUP BY

4.- CONSULTAS AGRUPADAS

- La claúsula GROUP BY en una consulta SELECT varía el comportamiento de la consulta de la forma:
  - Las funciones agregadas no actúan sobre todas las filas seleccionadas, sino sobre las filas de cada grupo
    - Devuelven un valor por cada grupo.
  - Una consulta agrupada sólo puede devolver como resultado un único valor por grupo:

    ref_columna en la claúsula SELECT sólo puede ser
    - La misma *ref_columna* utilizada en el GROUP BY
    - Una expresión aritmética con una de esas *ref_columna*
    - Una referencia a una función AGREGADA (*ref_funcion*) que se aplicará al grupo y devolverá un único valor por grupo.

INFORMALMENTE:
"En la selección de una consulta agrupada, sólo pueden aparecer referencias a columnas por las cuales se agrupa o referencias a funciones agregadas"

4.- CONSULTAS AGRUPADAS

EJEMPLOS: Recordando esquema lógico CICLISMO...

EQUIPO (nomeq: varchar2(25), director: varchar2(100))
CP= {nomeq}

CICLISTA (dorsal: number(3), nombre: varchar2(30), edad: number(2), nomeq: varchar2(25))
CP= {dorsal}
CA= {nomeq} → EQUIPO
VNN= {nomeq}
VNN= {nombre}

ETAPA (netapa: number(2), km: number(3), salida: varchar2(35), llegada: varchar2(35), dorsal: number(3))
CP= {netapa}
CA= {dorsal} → CICLISTA

PUERTO (nompuerto: varchar2(35), altura: number(3), categoria: char(1), pendiente: number(3,2), netapa: number(2), dorsal: number(3))
CP= {nompuerto}
CA= {netapa} → ETAPA
CA= {dorsal} → CICLISTA
VNN= {netapa}

MAILLOT (codigo: char(3), tipo: varchar2(30), premio: number(7), color: varchar2(20))
CP= {codigo}

LLEVAR (netapa: number(2), codigo: char(3), dorsal: number(3))
CP= {netapa,codigo}
CA= {netapa}→ ETAPA
CA= {codigo}→ MAILLOT
CA= {dorsal}→ CICLISTA
VNN= {dorsal}

4.- CONSULTAS AGRUPADAS

Ejemplo: (esquema ciclismo)

- Obtener el nombre de cada equipo y la edad media de los ciclistas de dicho equipo

```
SELECT    NOMEQ, AVG (EDAD)
FROM      CICLISTA
GROUP BY  NOMEQ
```

CICLISTA

...	nomeq	edad
...	Banesto	22
...	Once	25
...	PDM	32
...	Banesto	25
...	Kelme	28
...	Once	30
...	Kelme	29
...	Banesto	28

4.- CONSULTAS AGRUPADAS

Ejemplo:

- Resultado

```
SELECT    NOMEQ, AVG (EDAD)
FROM      CICLISTA
GROUP BY  NOMEQ
```

...	Nomeq	edad
...	Banesto	22
...	Banesto	25
...	Banesto	28
...	Once	25
...	Once	30
...	PDM	32
...	Kelme	28
...	Kelme	29

nomeq	AVG(edad)
Banesto	25
Once	27,5
PDM	32
Kelme	28,5

Existe un único valor por grupo!

4.- CONSULTAS AGRUPADAS

Ejemplo:

```
SELECT    NOMEQ, ~~NOMBRE~~, AVG (EDAD)
FROM      CICLISTA
GROUP BY  NOMEQ
```

→ ERROR!

...	nombre	nomeq	edad
...	Gorospe	Banesto	22
...	Delgado	Banesto	25
...	Indurain	Banesto	28
...	Zulle	Once	25
...	Jalabert	Once	30
...	Gonzalez	PDM	32
...	Corredor	Kelme	28
...	Mora	Kelme	29

NO existe un único valor por grupo!

4.- CONSULTAS AGRUPADAS

◎ GROUP BY Y WHERE:

- Si una consulta agrupada incluye la cláusula **WHERE,** esta cláusula se aplica antes de la agrupación.

```
4 SELECT    NOMEQ, AVG (EDAD)
1 FROM      CICLISTA
2 WHERE     EDAD>25
3 GROUP BY  NOMEQ
```

...	nomeq	edad
...	Banesto	22
...	Banesto	25
...	Banesto	28
...	Once	25
...	Once	30
...	PDM	32
...	Kelme	28
...	Kelme	29

...	nomeq	edad
...	Banesto	28
...	Once	30
...	PDM	32
...	Kelme	28
...	Kelme	29

nomeq	AVG(edad)
Banesto	28
Once	30
PDM	32
Kelme	28,5

ÍNDICE

4.- CONSULTAS AGRUPADAS
4.1.- Claúsula HAVING

```
SELECT [ALL|DISTINCT]
       [comalista_ref_columna | *]
FROM   comalista_ref_tabla
[WHERE condicion]
[GROUP BY comalista_ref_columna
[HAVING condicion]]
[ORDER BY comalista_ref_columna [DESC|ASC]]
```

- HAVING: solo puede aparecer en consultas agrupadas
  - Similar a la claúsula WHERE pero para grupos en lugar de para filas individuales:
    - Permite seleccionar los grupos que cumplen *condicion*
  - Las *ref_columna* permitidas en HAVING son las mismas que se permiten en la claúsula SELECT (diapositiva 32)
  - Se evalúa después de realizar el agrupamiento.
  - Informalmente se conoce como el WHERE del GROUP BY

4.1.- Claúsula HAVING

◎ Ejemplo: (esquema ciclismo)

- Obtener de los equipos con más de 2 corredores, el nombre del equipo y la edad media de sus ciclistas.

```
4 SELECT    NOMEQ, AVG (EDAD)
1 FROM      CICLISTA
2 GROUP BY  NOMEQ
3 HAVING    COUNT (*) >2
```

...	nomeq	edad
...	Banesto	22
...	Banesto	25
...	Banesto	28
...	Once	25
...	Once	30
...	PDM	32
...	Kelme	28
...	Kelme	29

nomeq	AVG(edad)
Banesto	25

4.1.- Claúsula HAVING

- **GROUP BY, WHERE Y HAVING**: ejemplo

```
5 SELECT    NOMEQ, AVG (EDAD)
1 FROM      CICLISTA
2 WHERE     EDAD>25
3 GROUP BY  NOMEQ
4 HAVING    COUNT (*)>=2
```

...	nomeq	edad
...	Banesto	22
...	Banesto	25
...	Banesto	28
...	Once	25
...	Once	30
...	PDM	32
...	Kelme	28
...	Kelme	29

...	nomeq	edad
...	Banesto	28
...	Once	30
...	PDM	32
...	Kelme	28
...	Kelme	29

→

nomeq	AVG(edad)
Kelme	28,5

ÍNDICE

4.- CONSULTAS AGRUPADAS
4.2.- Funciones agregadas

- Tal y como ya hemos comentado el comportamiento de las funciones agregadas es distinto en consultas agrupadas:
  - La función agregada se aplica al grupo y no a todas las filas de la tabla.
  - Esto permite los 3 casos prohibidos sin el GROUP BY:
    - Combinar campo individual y función agregada
    - Utilizar funciones agregadas en el WHERE
    - Anidar funciones agregadas
  - Es como si tuviéramos una vida más…
  - No obstante hay que tener cuidado porque: **SOLO TENEMOS UNA VIDA MÁS**…

4.- CONSULTAS AGRUPADAS
4.2.- Funciones agregadas

- **Ejemplo 1**: SELECT agregada+campo agrupado

SELECT FROM GROUP BY	NOMEQ, AVG (EDAD) CICLISTA NOMEQ	**vs**	SELECT FROM	NOMEQ, AVG (EDAD) CICLISTA
CORRECTA			**INCORRECTA**	

  - ¿Qué obtiene la consulta correcta?
- **Ejemplo 2**: agregada en HAVING

SELECT FROM GROUP BY HAVING	NOMEQ CICLISTA NOMEQ AVG (EDAD)<25	**vs**	SELECT FROM WHERE	NOMEQ CICLISTA AVG (EDAD)<25
CORRECTA			**INCORRECTA**	

  - ¿Qué obtiene la consulta correcta?

4.- CONSULTAS AGRUPADAS
4.2.- Funciones agregadas

- **Ejemplo 3**: ANIDAMIENTO agregadas en SELECT

SELECT FROM GROUP BY	MAX (AVG (EDAD)) CICLISTA NOMEQ	**VS**	SELECT FROM	MAX (AVG (EDAD)) CICLISTA
CORRECTA			**INCORRECTA**	

  - ¿Qué obtiene la consulta correcta?
- Esto es lo que **ESTÁ PERMITIDO** en las consultas agrupadas que **NO ESTABA PERMITIDO** en las no agrupadas
- Pero, ya hemos agotado nuestra vida…
  - Volvemos a encontrarnos con lo mismo

    Cuidado con lo mostrado a continuación…

4.- CONSULTAS AGRUPADAS
4.2.- Funciones agregadas

- Estas consultas son **INCORRECTAS**:

```
SELECT    NOMEQ, MAX (AVG (EDAD))
FROM      CICLISTA
GROUP BY  NOMEQ;
```

```
SELECT    NOMEQ
FROM      CICLISTA
GROUP BY  NOMEQ
HAVING    AVG (EDAD)= MAX (AVG (EDAD));
```

```
SELECT    COUNT (MAX (AVG (EDAD)))
FROM      CICLISTA
GROUP BY  NOMEQ;
```

4.- CONSULTAS AGRUPADAS
4.2.- Funciones agregadas

- Entonces ¿como resolverías esta consulta?
  - *Obtener el nombre del equipo con mayor media de edad en sus ciclistas. Mostrar también la media de edad de sus ciclistas.*

PISTA: LAS SUBCONSULTAS EXISTEN COMO TERUEL

ÍNDICE

4.- CONSULTAS AGRUPADAS
4.3.- Agrupación no significativa

- Supongamos (de nuevo) la siguiente consulta:
  - Obtener el nombre de cada equipo y la edad media de los ciclistas de dicho equipo

```
SELECT    NOMEQ, AVG (EDAD)
FROM      CICLISTA
GROUP BY  NOMEQ
```

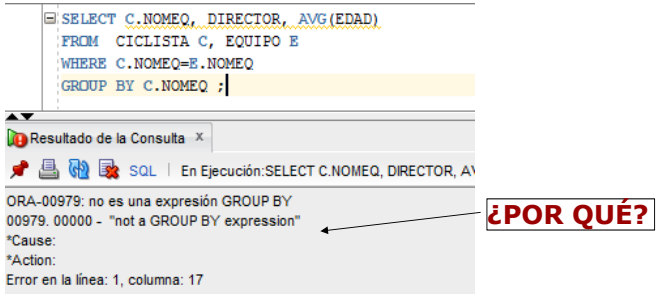
- Supongamos ahora que modificamos la consulta de la siguiente forma:
  - Obtener el nombre **y el director** de cada equipo y la edad media de los ciclistas de dicho equipo.
  - LO PRIMERO QUE DEBEMOS PLANTEARNOS ES:
    - ¿Tiene sentido querer obtener esta información en una consulta agrupada?
    - ¿Existe un solo director por cada nombre de equipo?

4.- CONSULTAS AGRUPADAS
4.3.- Agrupación no significativa

- Formulamos la consulta de la siguiente forma:

```
SELECT    C.NOMEQ, DIRECTOR, AVG (EDAD)
FROM      CICLISTA C, EQUIPO E
WHERE     C.NOMEQ=E.NOMEQ
GROUP BY  C.NOMEQ
```

- Pero esto nos dará un error…



4.- CONSULTAS AGRUPADAS
4.3.- Agrupación no significativa

- RECORDANDO…

> **INFORMALMENTE**:
> "En la selección de una consulta agrupada, sólo pueden aparecer referencias a columnas por las cuales se agrupa o referencias a funciones agregadas"

- La SOLUCIÓN consiste en añadir en la agrupación el campo que queremos mostrar **y que no cambia la agrupación** (los grupos son los mismos)
  - A esto se le denomina **AGRUPACIÓN NO SIGNIFICATIVA**
  - Podemos añadir al GROUP BY todos los campos que queramos mostrar que no cambien la agrupación

```
SELECT    C.NOMEQ, DIRECTOR, AVG (EDAD)
FROM      CICLISTA C, EQUIPO E
WHERE     C.NOMEQ=E.NOMEQ
GROUP BY  C.NOMEQ, DIRECTOR
```


ÍNDICE

5.- CONCATENACIÓN DE TABLAS: operador JOIN

- Como ya vimos en el tema anterior existen distintas formas de obtener datos de varias tablas en una consulta SQL:
  - Incluir varias tablas en la cláusula FROM.
    - Y ligarlas adecuadamente en el WHERE.
  - Uso de subconsultas en las cláusulas WHERE y HAVING.
- Existen además otras formas que no hemos visto:
  - **Concatenacion de tablas**: uso de las distintas variantes del operador JOIN o concatenación.
    - INNER, LEFT, RIGHT.
  - **Combinaciones conjuntistas de tablas**: a través de operadores conjuntistas.
    - UNIÓN, INTERSECCIÓN, DIFERENCIA.

5.- CONCATENACIÓN DE TABLAS: operador JOIN

```
SELECT [ALL|DISTINCT]
       [comalista_ref_columna | *]
FROM   comalista_ref_tabla ←
[WHERE condicion]
[GROUP BY comalista_ref_columna
[HAVING condicion]]
[ORDER BY comalista_ref_columna [DESC|ASC]]
```

```
ref_tabla::= tabla [alias]|
             ref_tabla [NATURAL] [INNER| LEFT| RIGHT] JOIN ref_tabla
             [ON condicion| USING (comalista_atributo)] ←
```

- **TIPOS DE CONCATENACIÓN**:
  - Concatenación interna: INNER JOIN
  - Concatenaciones externas: OUTER JOINS
    - LEFT, RIGHT

ÍNDICE

5.1.- CONCATENACIÓN INTERNA

- **CONCATENACION INTERNA (INNER JOIN)**
  - **INNER JOIN ON (Ejemplo):**

```
SELECT   *
FROM     ciclista INNER JOIN equipo ON ciclista.nomeq=equipo.nomeq
```

≡

```
SELECT   *
FROM     ciclista, equipo
WHERE    ciclista.nomeq=equipo.nomeq
```

CONCATENA (liga) ambas tablas por nomeq

EQUIPO

nomeq	director
Banesto	Echevarria
Once	Pino
PDM	Kruis
Kelme	Sainz

CICLISTA

dorsal	nombre	nomeq	edad
1	Indurain	Banesto	32
2	delgado	Banesto	35
3	Zulle	Once	27

RESULTADO

dorsal	nombre	nomeq	edad	nomeq	director
1	Indurain	Banesto	32	Banesto	Echevarria
2	Delgado	Banesto	35	Banesto	Echevarria
3	Zulle	Once	27	Once	Pino

5.1.- CONCATENACIÓN INTERNA

CONCATENACION INTERNA (INNER JOIN)

- **INNER JOIN ON (En general):**

```
SELECT   *
FROM     tabla1 INNER JOIN tabla2 ON  tabla1.a1=tabla2.b1 AND
                                      tabla1.a2=tabla2.b2 AND
                                      ...
                                      tabla1.an=tabla2.bn
```

≡

```
SELECT   *
FROM     tabla1, tabla2
WHERE    tabla1.a1=tabla2.b1 AND
         tabla1.a2=tabla2.b2 AND
         ...
         tabla1.an=tabla2.bn
```

CONCATENA (liga) ambas tablas por los atributos especificados

- No se debe olvidar que una CP puede ser múltiple y se puede ligar por varios campos

5.1.- CONCATENACIÓN INTERNA

CONCATENACION INTERNA (INNER JOIN):

- **INNER JOIN USING (Ejemplo):**

```
SELECT   *
FROM     ciclista INNER JOIN equipo ON ciclista.nomeq=equipo.nomeq
```

≡

```
SELECT   *
FROM     ciclista INNER JOIN equipo USING (nomeq)
```

mismo nombre!

- **En general**: si los atributos por los que se quiere ligar o concatenar ambas tablas tienen el mismo nombre

```
SELECT   *
FROM     tabla1 INNER JOIN tabla2 ON  tabla1.a1=tabla2.a1 AND
                                      tabla1.a2=tabla2.a2 AND
                                      ...
                                      tabla1.an=tabla2.an
```

≡

```
SELECT   *
FROM     tabla1 INNER JOIN tabla2 USING (a1, a2, ...an)
```


5.1.- CONCATENACIÓN INTERNA

CONCATENACION INTERNA (INNER JOIN):

- **NATURAL JOIN (Ejemplo):**

```
SELECT   *
FROM     ciclista NATURAL INNER JOIN equipo
```

Concatena por los atributos con el mismo nombre en ambas tablas!

≡

```
SELECT   *
FROM     ciclista INNER JOIN equipo ON ciclista.nomeq=equipo.nomeq
```

- **En general**: si las tablas a concatenar tienen atributos con el mismo nombre (a1, a2, ...an)

```
SELECT   *
FROM     tabla1 NATURAL INNER JOIN tabla2
```

≡

```
SELECT   *
FROM     tabla1 INNER JOIN tabla2 ON  tabla1.a1=tabla2.a1 AND
                                      tabla1.a2=tabla2.a2 AND
                                      ...
                                      tabla1.an=tabla2.an
```


5.1.- CONCATENACIÓN INTERNA

CONCATENACION INTERNA: RESUMEN

- **Ejemplo:**

```
SELECT   *
FROM     ciclista INNER JOIN equipo ON ciclista.nomeq=equipo.nomeq
```

≡

```
SELECT   *
FROM     ciclista INNER JOIN equipo USING (nomeq)
```

≡

```
SELECT   *
FROM     ciclista NATURAL INNER JOIN equipo
```

≡

```
SELECT   *
FROM     ciclista, equipo
WHERE    ciclista.nomeq=equipo.nomeq
```

5.1.- CONCATENACIÓN INTERNA

CONCATENACION INTERNA: RESUMEN

- **En general:**

```
SELECT   *
FROM     tabla1 INNER JOIN tabla2 ON  tabla1.a1=tabla2.a1 AND
                                      tabla1.a2=tabla2.a2 AND
                                      ...
                                      tabla1.an=tabla2.an
```

≡

```
SELECT   *
FROM     tabla1 INNER JOIN tabla2 USING (a1, a2, ...an)
```

≡

Sólo si los atributos por los que se desea concatenar tabla1 y tabla2 tienen el mismo nombre!

```
SELECT   *
FROM     tabla1 NATURAL INNER JOIN tabla2
```

≡

```
SELECT   *
FROM     tabla1, tabla2
WHERE    tabla1.a1=tabla2.a1 AND
         tabla1.a2=tabla2.a2 AND
         ...
         tabla1.an=tabla2.an
```


5.1.- CONCATENACIÓN INTERNA

CONSIDERACIONES FINALES:

Supongamos que queremos realizar la siguiente consulta:

"Obtener para cada equipo su nombre, director y el nº de ciclistas del equipo".

```
SELECT   C.NOMEQ, DIRECTOR, COUNT(*) NUMCICLISTAS
FROM     CICLISTA C, EQUIPO E
WHERE    C.NOMEQ=E.NOMEQ
GROUP BY C.NOMEQ, DIRECTOR
```

- **Analicemos el resultado...**

5.1.- CONCATENACIÓN INTERNA

CONSIDERACIONES FINALES:

Suponiendo que las tablas tienen los siguientes datos:

EQUIPO

nomeq	director
Banesto	Echevarria
Once	Pino
PDM	Kruis
Kelme	Sainz

CICLISTA

dorsal	nombre	nomeq	edad
1	Indurain	Banesto	32
2	delgado	Banesto	35
3	Zulle	Once	27

RESULTADO CONCATENACIÓN INTERNA (FROM/WHERE)

dorsal	nombre	nomeq	edad	nomeq	director
1	Indurain	Banesto	32	Banesto	Echevarria
2	Delgado	Banesto	35	Banesto	Echevarria
3	Zulle	Once	27	Once	Pino

RESULTADO FINAL

nomeq	director	NUMCICLISTAS
Banesto	Echevarria	2
Once	Pino	1

- Obviamente no salen los equipos sin ciclistas...

5.1.- CONCATENACIÓN INTERNA

CONSIDERACIONES FINALES

Supongamos ahora que queremos realizar la siguiente consulta:

"Obtener para cada equipo su nombre, director y el nº de ciclistas del equipo. Se deben visualizar todos los equipos"

- No queremos que salgan solo los equipos que tienen ciclistas, sino todos los equipos.
- Para los equipos que no tienen ciclistas el nº de ciclistas del equipo será cero.
- Con lo que sabemos hasta ahora es imposible...
- Necesitamos una concatenación EXTERNA

ÍNDICE

5.2.- CONCATENACIONES EXTERNAS:
5.2.1.- LEFT JOIN

◎ LEFT JOIN ON (Ejemplo):

```
SELECT  *
FROM    equipo LEFT JOIN ciclista ON equipo.nomeq=ciclista.nomeq
```

- Es como un INNER JOIN extendido.
- Selecciona todas las filas de la tabla equipo aunque no tengan correspondencia con ninguna fila de ciclista.
  - Los campos de ciclista de las filas sin correspondencia serán NULL

EQUIPO

nomeq	director
Banesto	Echevarria
Once	Pino
PDM	Kruis
Kelme	Sainz

CICLISTA

dorsal	nombre	nomeq	edad
1	Indurain	Banesto	32
2	delgado	Banesto	35
3	Zulle	Once	27

RESULTADO

nomeq	director	dorsal	nombre	nomeq	edad
Banesto	Echevarria	1	Indurain	Banesto	32
Banesto	Echevarria	2	Delgado	Banesto	35
Once	Pino	3	Zulle	Once	27
PDM	Kruis	NULL	NULL	NULL	NULL
Kelme	Sainz	NULL	NULL	NULL	NULL

5.2.- CONCATENACIONES EXTERNAS:
5.2.1.- LEFT JOIN

◎ LEFT JOIN ON (EN GENERAL):

```
SELECT  *
FROM    tabla1 LEFT JOIN tabla2 ON  tabla1.a1=tabla2.b1 AND
                                    tabla1.a2=tabla2.b2 AND
                                    ...
                                    tabla1.an=tabla2.bn
```

- Concatena las filas de tabla1 con las filas de tabla2 por los atributos indicados (INNER JOIN).
- Selecciona también las filas de tabla1 que no pueden ser concatenadas (no tienen correspondencia en tabla2)
  - Los campos de tabla2 para esas filas serán NULL.

5.2.- CONCATENACIONES EXTERNAS:
5.2.1.- LEFT JOIN

◎ ¿Pero qué sentido tiene obtener este resultado?

- ¿Para que me pueden valer los registros de equipos sin ciclistas? (con valores NULL)

◎ Volvamos ahora a la siguiente consulta:

"Obtener para cada equipo su nombre, director y el nº de ciclistas del equipo. Se deben visualizar todos los equipos"

- Ya podemos intentar resolverla...

```
SELECT    C.NOMEQ, DIRECTOR, COUNT(*) NUMCICLISTAS
FROM      EQUIPO E LEFT JOIN CICLISTA C ON C.NOMEQ=E.NOMEQ
GROUP BY  C.NOMEQ, DIRECTOR
```

- **Analicemos el resultado...**

5.2.- CONCATENACIONES EXTERNAS:
5.2.1.- LEFT JOIN

- *Suponiendo que las tablas tienen los siguientes datos:*

EQUIPO

nomeq	director
Banesto	Echevarria
Once	Pino
PDM	Kruis
Kelme	Sainz

CICLISTA

dorsal	nombre	nomeq	edad
1	Indurain	Banesto	32
2	delgado	Banesto	35
3	Zulle	Once	27

RESULTADO CONCATENACIÓN EXTERNA (FROM/WHERE)

nomeq	director	dorsal	nombre	nomeq	edad
Banesto	Echevarria	1	Indurain	Banesto	32
Banesto	Echevarria	2	Delgado	Banesto	35
Once	Pino	3	Zulle	Once	27
PDM	Kruis	NULL	NULL	NULL	NULL
Kelme	Sainz	NULL	NULL	NULL	NULL

No me da ¿XQ?

RESULTADO FINAL

nomeq	director	NUMCICLISTAS
Banesto	Echevarria	2
Once	Pino	1
PDM	Kruis	1
Kelme	Sainz	1

5.2.- CONCATENACIONES EXTERNAS:
5.2.1.- LEFT JOIN

- Vamos a afinar un poco...
  - ¿Cuantos registros voy a tener siempre para cada equipo sin ciclistas?
  - Observa bien esos registros ¿Por qué se caracterizan?
- Volvamos de nuevo a la consulta:

"Obtener para cada equipo su nombre, director y el nº de ciclistas del equipo. Se deben visualizar todos los equipos"

```
SELECT   C.NOMEQ, DIRECTOR, COUNT(DORSAL) NUMCICLISTAS
FROM     EQUIPO E LEFT JOIN CICLISTA C ON C.NOMEQ=E.NOMEQ
GROUP BY C.NOMEQ, DIRECTOR
```


5.2.- CONCATENACIONES EXTERNAS:
5.2.1.- LEFT JOIN

- *Suponiendo que las tablas tienen los siguientes datos:*

EQUIPO

nomeq	director
Banesto	Echevarria
Once	Pino
PDM	Kruis
Kelme	Sainz

CICLISTA

dorsal	nombre	nomeq	edad
1	Indurain	Banesto	32
2	delgado	Banesto	35
3	Zulle	Once	27

RESULTADO CONCATENACIÓN EXTERNA (FROM/WHERE)

nomeq	director	dorsal	nombre	nomeq	edad
Banesto	Echevarria	1	Indurain	Banesto	32
Banesto	Echevarria	2	Delgado	Banesto	35
Once	Pino	3	Zulle	Once	27
PDM	Kruis	NULL	NULL	NULL	NULL
Kelme	Sainz	NULL	NULL	NULL	NULL

No me da ¿XQ?

RESULTADO FINAL

nomeq	director	NUMCICLISTAS
Banesto	Echevarria	2
Once	Pino	1
PDM	Kruis	1
Kelme	Sainz	1

5.2.- CONCATENACIONES EXTERNAS:
5.2.2.- RIGHT JOIN

- **CONCATENACIONES EXTERNAS:**
  - **RIGHT JOIN ON (EN GENERAL):**

```
SELECT  *
FROM    tabla1 RIGHT JOIN tabla2 ON  tabla1.a1=tabla2.b1 AND
                                     tabla1.a2=tabla2.b2 AND
                                     ...
                                     tabla1.an=tabla2.bn
```

  - **Idéntico a LEFT JOIN pero cambiando el sentido**
    - No se suele usar (se usa LEFT JOIN)
  - Concatena las filas de tabla2 con las filas de tabla1 por los atributos indicados (INNER JOIN).
  - Selecciona también las filas de tabla 2 que no pueden ser concatenadas (no tienen correspondencia en tabla1)
    - Los campos de tabla1 para esas filas serán NULL.

5.2.- CONCATENACIONES EXTERNAS:
5.2.2.- RIGHT JOIN

◎ CONCATENACIONES EXTERNAS:

- **RIGHT JOIN ON (Ejemplo):**

```
SELECT  *
FROM    ciclista RIGHT JOIN equipo ON equipo.nomeq=ciclista.nomeq
```

  - ◆El resultado es idéntico

EQUIPO

nomeq	director
Banesto	Echevarria
Once	Pino
PDM	Kruis
Kelme	Sainz

CICLISTA

dorsal	nombre	nomeq	edad
1	Indurain	Banesto	32
2	delgado	Banesto	35
3	Zulle	Once	27

RESULTADO

nomeq	director	dorsal	nombre	nomeq	edad
Banesto	Echevarria	1	Indurain	Banesto	32
Banesto	Echevarria	2	Delgado	Banesto	35
Once	Pino	3	Zulle	Once	27
PDM	Kruis	NULL	NULL	NULL	NULL
Kelme	Sainz	NULL	NULL	NULL	NULL

5.- CONCATENACIÓN DE TABLAS:
CONSIDERACIONES FINALES

◎ CONCATENACIONES CON USING/NATURAL:

- En la práctica si realizamos un JOIN (de cualquier tipo) utilizando las cláusulas USING o NATURAL, la relación resultado sólo dispondrá 1 vez de la columna(s) por la que se concatena

```
SELECT  *
FROM    equipo INNER|LEFT|RIGTH JOIN ciclista USING (nomeq)
```

```
SELECT  *
FROM    ciclista NATURAL INNER|LEFT|RIGTH JOIN equipo
```

  - ◆No tendrá sentido cualificar la columna(s) por la que se concatena con su nombre de tabla
  - ◆Esto no ocurre si utilizamos la cláusula ON

```
SELECT  *
FROM    ciclista INNER|LEFT|RIGTH JOIN equipo ON ciclista.nomeq=equipo.nomeq
```


5.- CONCATENACIÓN DE TABLAS:
CONSIDERACIONES FINALES

◎ CONCATENACIONES CON USING/NATURAL:

- Ejemplos:

EQUIPO

nomeq	director
Banesto	Echevarria
Once	Pino
PDM	Kruis
Kelme	Sainz

CICLISTA

dorsal	nombre	nomeq	edad
1	Indurain	Banesto	32
2	delgado	Banesto	35
3	Zulle	Once	27

```
SELECT  *
FROM    equipo INNER JOIN ciclista USING (nomeq)
```

```
SELECT  *
FROM    equipo NATURAL INNER JOIN ciclista
```

nomeq	director	dorsal	nombre	edad
Banesto	Echevarria	1	Indurain	32
Banesto	Echevarria	2	Delgado	35
Once	Pino	3	Zulle	27

- No existen ciclista.nomeq y equipo.nomeq (sólo nomeq)

5.- CONCATENACIÓN DE TABLAS:
CONSIDERACIONES FINALES

◎ OPERADOR (+) EN ORACLE:

- Hasta hace poco Oracle no disponía del operador JOIN:
  - ◆Las concatenaciones internas se implementan mediante el uso de varias tablas en la cláusula FROM y ligando apropiadamente ambas tablas en la cláusula WHERE.
  - ◆Las concatenaciones externas eran de la siguiente forma:

```
SELECT  *
FROM    tabla1, tabla2
WHERE   tabla1.a1=tabla2.b1 (+) AND
        tabla1.a2=tabla2.b2 (+) AND
        ...
        tabla1.an=tabla2.bn (+)
```

≡

```
SELECT  *
FROM    tabla1 LEFT JOIN tabla2 ON  tabla1.a1=tabla2.b1 AND
                                    tabla1.a2=tabla2.b2 AND
                                    ...
                                    tabla1.an=tabla2.bn
```