Pràctica 3:

Llenguatge SQL

1ª Part: Manipulació de Bases de Dades

Objectius:

- Presentar la sintaxi del llenguatge SQL (només del Llenguatge de Manipulació).
- Vore alguns exemples senzills per a clarificar la semàntica del SQL.
- Presentar (o recordar) les bases de dades CICLISME i MÚSICA.
- Realitzar de menor a major complexitat consultes SQL sobre les dites bases de dades.
- Realitzar tot l'anterior utilitzant l'eina SQL Worksheet del sistema de gestió de bases de dades ORACLE.

1. Llenguatge de Manipulació del SQL

Es presenten les instruccions que es poden executar des d'un intèrpret de SQL, allò que es coneix com *SQL interactiu*.

SQL és un llenguatge molt expressiu i, en general, permet moltes formes d'expressar les mateixes ordres.

Les quatre instruccions que componen el llenguatge de manipulació de dades són les següents:

- select: permet la declaració de consultes per a la recuperació de informació d'una o més taules d'una base de dades.
- insert: realitza la inserció d'una o vàries files sobre una taula.
- delete: permet efectuar l'esborrament d'una o vàries files d'una taula.
- update: realitza una modificació dels valors d'una o més columnes d'una o vàries files d'una taula.

1.1. Consultes: instrucció select

select [**all** | **distinct**] *comallista_item_seleccionat* | *

from *comallista_referència_taula*

[**where** *expressió_condicional*]

[**group by** *comallista_referència_col*]

[**having** *expressió_condicional*]

[**order by** *comallista_referència_col*]

- *comallista_item_seleccionat*: informació a obtindre de la base de dades.
- **from** *comallista_referència_taula*: especifica de quines taules s'obté la informació buscada.
- **where** *expressió_condicional*: expressa una condició que han de complir les files de la consulta resultant.
- **group by** *comallista_referència_col*: permet formar consultes agrupades per a extraure informació global sobre els grups formats.
- **having** *expressió_condicional*: condició sobre els grups formats.
- **order by** *comallista_referència_col*: ordena per una o vàries columnes.

1.1.1. Condicions en consultes simples

3 **select** [**all** | **distinct**] *comallista_ítem_seleccionat* | *

1 **from** taula

2 [**where** *expressió_condicional*]

4 [**order by** *comallista_referència_col*]

- **all** : Permet l'aparició de files idèntiques (valor per defecte).
- **distinct**: No permet l'aparició de files idèntiques.
- L'*expressió_condicional* està formada per un conjunt de predicats combinats amb les connectives lògiques and, or i not. Els predicats utilitzats permeten comparar columnes:
- predicats de comparació: =, <>, >, <, >=, <=.
- predicat like: permet comparar una tira de caràcters amb un patró.
- predicat between: permet comprovar si un escalar està dins un rang.
- predicat in: permet comprovar si el valor està dins un conjunt.
- predicat is null: permet comprovar si el valor és nul.

EXEMPLE: Obtindre el nom i l'edat de tots els ciclistes.

```
SELECT nombre, edad FROM Ciclista;
```

EXEMPLE: Obtindre el nom i l'altura de tots els ports de 1ª categoria.

```
SELECT nompuerto, altura FROM Puerto
WHERE categoria = '1';
```

EXEMPLE: Obtindre el nom dels ciclistes tal que llur edat estiga entre 20 i 30 anys.

```
SELECT nombre FROM Ciclista
WHERE edad BETWEEN 20 AND 30;
```

(*) El predicat between és equivalent a una condició amb comparacions de la següent forma:

exp between exp_1 and exp_2 ≡ (*exp* >= exp_1) and (*exp* <= exp_2)

EXEMPLE: Obtindre el número de les etapes a on el nom de la ciutat de arribada tinga per segona lletra una “O” o a on el nom de la ciutat d’eixida tinga dues o més ‘A’s.

```
SELECT netapa FROM Etapa
WHERE llegada LIKE ‘_O%’ OR salida LIKE ‘%A%A%’;
```

EXEMPLE: Obtindre el nom dels ports de 1ª, 2ª o 3ª categoria.

```
SELECT nompuerto FROM Puerto
WHERE categoria IN ( ‘1’, ‘2’, ‘3’ ) ;
```

(*) També el predicat in és derivat i l’expressió equivalent és:

exp in (exp_1, exp_2, …, exp_n) ≡ (exp=exp_1) or (exp=exp_2) or…or (exp=exp_n)

EXEMPLE: Obtindre totes les dades d’aquells ciclistes dels que es desconeixia llur edat.

```
SELECT * FROM Ciclista
WHERE edad IS NULL;
```

COMPARACIÓ DE VALORS NULS

Les comparacions entre qualsevol valor i NULL resulten en *indefinit*. EXEMPLE:

```
select *
from T
where atrib1 > atrib2;
```

Si en una fila es donara el cas que atrib1 = 50 i atrib2 fóra nul, el resultat de la comparació seria indefinit i per tant la dita fila no s'inclouria en la selecció.

EXEMPLE de consulta incorrecta (error de sintaxi)

```
select nomeq
from Equipo
where  director = null;
```

MÉS EXEMPLES DE COMPARACIONS

Ús d'operadors aritmètics: + (suma), – (diferència), * (producte), / (divisió), etc.

EXEMPLE: Obtindre dels maillots el tipus i el premi en euros (suposem que encara està en pessetes) d'aquells maillots tal que llur premi supere els 100 euros

SELECT tipo, premio / 166.386 FROM Maillot

WHERE premio / 166.386 > 100;

Ús de LIKE

EXEMPLE: Obtindre el nom i l'edat dels ciclistes que pertanguen a equips tal que llur nom continga la cadena "100%".

SELECT nombre, edad FROM Ciclista

WHERE nomeq LIKE '%100\%%' ESCAPE '\';

CONSULTES DE VALORS AGREGATS

La sintaxi d'una referència a una funció agregada és la següent:

{ avg | max | min | sum | count } ([all | distinct] *expressió_escalar*) | count(*)

- Les funcions agregades no es poden niar.
- Per a les funcions sum i avg els arguments han de ser numèrics.
- distinct indica que els valors redundants siguen eliminats abans que es realitze el càlcul corresponent.
- La funció especial count(*), en la qual no està permés incloure distinct ni all, dóna com a resultat el cardinal del conjunt de files de la selecció.
- Els càlculs es realitzen després de la selecció i d'aplicar les condicions.
- Els valors nuls són eliminats abans de realitzar els càlculs (incl. count).
- Si el número de files de la selecció és 0, la funció count retorna el valor 0 i les altres funcions el valor nul.

FUNCIONS AGREGADES EN CONSULTES NO AGRUPADES

EXEMPLE:

```
SELECT 'Núm. de ciclistes =', COUNT(*), 'Mitjana Edat =', AVG(edad)
FROM Ciclista
WHERE nomeq = 'Banesto';
```

En consultes no agrupades, la selecció només podrà incloure referències a funcions agregades o literals ja que les funcions retornaran un únic valor.

EXEMPLE INCORRECTE:

```
SELECT nombre, AVG(edad)
FROM Ciclista
WHERE nomeq = 'ONCE';
```

CONSULTES SIMPLES SOBRE VÀRIES TAULES

Quan la informació que es desitja obtindre de la base de dades es troba emmagatzemada en més d'una taula es fa indispensable el declarar una consulta que manipule aquestes taules.

EXEMPLE: Obtindre parells de números d'etapes i noms de ports guanyats pel mateix ciclista.

```
select etapa.netapa, nompuerto
from Etapa, Puerto
where etapa.dorsal = puerto.dorsal;
```

En aquesta expressió és obligatori que la referència a la columna *dorsal* d'*Etapa* i *Puerto* siga qualificada amb el nom de la taula, si no és ambígua. En general,

[taula | *variable_recorregut*].*columna*

Les variables de recorregut permeten donar un nom alternatiu a la *mateixa* taula dins una consulta. La manera de declarar una variable de recorregut és:

from taula [as] *variable_recorregut*

ÚS DE CLAUS ALIENES EN CONSULTES DE VÀRIES TAULES

La consulta de vàries taules correspon al producte cartesià.

Si no es trien bé les condicions el número de files resultants pot ser molt gran.

Si existeixen claus alienes, el més normal és una igualtat entre la clau aliena i els atributs corresponents de la taula a la que es fa referència.

EXEMPLE: Obtindre els noms dels ciclistes pertanyents a l'equip dirigit per 'Álvaro Pino'.

```
SELECT C.nombre FROM Ciclista C, Equipo E
WHERE C.nomeq = E.nomeq AND E.director = 'Alvaro Pino';
```

EXEMPLE: Obtindre parells nom de ciclista, número d'etapa, de tal forma que el dit ciclista haja guanyat la dita etapa. A més l'etapa ha de superar els 150 km. de recorregut.

```
SELECT C.nombre, E.netapa FROM Ciclista C, Etapa E
WHERE C.dorsal = E.dorsal AND E.km > 150;
```

CONSULTES COMPLEXES: SUBCONSULTES

Si la informació que s'està buscant està inclosa en una taula i la condició de cerca d'aquesta informació requereix accedir a altres taules, llavors (EN ALGUNS CASOS) es poden utilitzar les subconsultes per a expressar aquest tipus de condicions.

EXEMPLE: El mateix EXEMPLE anterior: Obtindre els noms dels ciclistes pertanyents a l'equip dirigit per 'Álvaro Pino'. Teníem utilitzant igualtats:

```
SELECT C.nombre FROM Ciclista C, Equipo E
WHERE C.nomeq = E.nomeq AND E.director = 'Álvaro Pino';
```

Utilitzant subconsultes:

```
SELECT C.nombre FROM Ciclista C
WHERE C.nomeq = (SELECT E.nomeq FROM Equipo E
                 WHERE E.director = 'Alvaro Pino');
```

Açò és possible perquè la informació que es requereix, nom del ciclista, no està en la taula de la subconsulta (Equipo) i perquè la subconsulta retorna un únic valor.

PREDICATS QUE ACCEPTEN SUBCONSULTES:

Les subconsultes poden aparéixer en les condicions de cerca, tant de la clàusula where com del having, així com arguments d'alguns predicats.

Els predicats que poden dur com a arguments subconsultes son els següents:

- predicats de comparació (=, <>, >, <, >=, <=).
- in: comprova que un valor *pertany* a una col·lecció donada mitjançant una subconsulta.
- match: comprova si un valor és idèntic a algun valor d'una col·lecció.
- predicats de comparació quantificats (any i all): permeten comparar un valor amb un conjunt de valors.
- exists: equivalent al quantificador existencial, comprova si una subconsulta retorna alguna fila.
- unique: retorna cert si la consulta no té files repetides.

PREDICATS DE COMPARACIÓ (=, <>, >, <, >=, <=)

Cadascun dels dos costats d'un predicat de comparació ha de ser una única tupla formada pel mateix número de columnes. És a dir:

(A1, A2, …, An) predicat_comparació (B1, B2, …, Bn)

Les subconsultes poden ser arguments sempre i quan retornen una única fila i el número de columnes coincidisca en número i tipus amb l'altre costat del predicat de comparació. Direm *constructor_fila* a una llista d'atributs entre parèntesis o una subconsulta.

constructor_fila predicat_comparació *constructor_fila*

- En el cas que la subconsulta estiga buida, es converteix a una fila amb valors nuls en totes les columnes.
- Per a poder comparar dos *constructor_fila* de més d'una columna, existeix una forma definida de realitzar aquesta comparació per a cadascun de (=, <>, >, <, >=, <=).
- En general, no obstant això, es voran subconsultes d'una única columna, com l'exemple anterior.

EXEMPLE: Obtindre els noms dels ports tal que llur altura és major que la mitjana d'altura dels ports de 2ª categoria.

```
SELECT nompuerto FROM Puerto
WHERE altura > (SELECT AVG(altura) FROM Puerto
        WHERE categoría = 2 );
```

INCORRECTE: (error d'execució):

```
SELECT nompuerto FROM Puerto
WHERE altura > (SELECT altura FROM Puerto
        WHERE categoría = 2 );
```

Predicat in

constructor_fila [not] in (expressió_taula)

A la dreta d'IN poden aparéixer més d'una fila i per això s'anomena *expressió_taula*.

EXEMPLE: Obtindre el nº de les etapas guanyades per ciclistes amb edat superior als 30 anys.

```
SELECT netapa FROM Etapa
WHERE dorsal IN (SELECT dorsal FROM Ciclista
                 WHERE edad > 30);
```

SUBCONSULTES ENCADENADES

EXEMPLE: Obtindre el número de les etapes guanyades per ciclistes que pertanguen a equips tal que llur director tinga un nom que comence per ‘A’.

```
SELECT netapa FROM Etapa
WHERE dorsal IN (SELECT dorsal FROM Ciclista
WHERE nomeq IN (SELECT nomeq FROM Equipo
    WHERE director LIKE ‘A%’));
```

Predicats de comparació quantificats (any i all)

constructor_fila predicat_comparació {all | any | some} (*expressió_taula*)

- El predicat de comparació quantificat amb ALL s'avalua a cert si així ho és per a totes les files de l'expressió de taula (si la taula està buida també s'avalua a cert).
- El predicat de comparació quantificat amb ANY o SOME s'avalua a cert si així ho és per a alguna fila de l'expressió de taula (si la taula està buida s'avalua a fals).

(*) el predicat *in* és idèntic al predicat de comparació quantificat =*any*.

EXEMPLE 8) Obtindre el nom dels ports i dels ciclistes que els hagen guanyat que tinguen la major pendent.

```
SELECT P.nompuerto, C.nombre FROM Puerto P, Ciclista C
WHERE P.dorsal = C.dorsal  AND
P.pendiente >= ALL (SELECT P1.pendiente FROM Puerto P1 );
```

EXEMPLE 9) Obtindre el nom dels ports i dels ciclistes que els hagen guanyat, complint que el port no siga el que tinga la menor pendent.

```
SELECT P.nompuerto, C.nombre FROM Puerto P, Ciclista C
WHERE P.dorsal = C.dorsal  AND
P.pendiente > ANY (SELECT P1.pendiente FROM Puerto P1 );
```

(*) Qualsevol ANY es pot convertir en un ALL canviant la condició a la seua condició negada i afegint un NOT.

Predicat exists

exists (*expresión_tabla*)

El predicat *exists* s'avalua a cert si l'expressió *select* retorna almenys una fila.

EXEMPLE: Obtindre el nom d'aquells ciclistes que han dut un maillot d'un premi menor de 50.000 ptes.

```
SELECT C.nombre FROM Ciclista C, Llevar L
WHERE C.dorsal = L.dorsal AND
EXISTS (SELECT *
        FROM Maillot M
        WHERE M.premio < 50000 AND M.codigo = L.codigo);
```

(*) En l'exemple es veu a més una referència externa des de la subconsulta a la taula C externa a ella, cosa que sol ser molt habitual (encara que no exclusiu) a les subconsultes amb EXISTS.

(*) En general, IN i EXISTS (en afirmatiu) són intercanviables i es poden eliminar fent consultes a múltiples taules i igualant per claus alienes (o per altres atributs).

EXEMPLE: Obtindre el nom dels ciclistes que no han guanyat etapes.

```
SELECT nombre FROM Ciclista
WHERE NOT EXISTS (SELECT * FROM Etapa
                  WHERE Etapa.dorsal = Ciclista.dorsal);

SELECT nombre FROM Ciclista
WHERE dorsal NOT IN (SELECT dorsal FROM Etapa);
```

Equivalències:

WHERE EXISTS (SELECT * FROM …)

equival a: WHERE 0 < (SELECT COUNT(*) FROM …)

WHERE NOT EXISTS (SELECT * FROM …)

equival a: WHERE 0 = (SELECT COUNT(*) FROM …)

Ús d'EXISTS per a quantificacions universals (NO HI HAN EN SQL):

EXEMPLE: Obtindre el nom del ciclista (si n'hi ha) que ha guanyat **totes** les etapes de més de 200 km.

```
SELECT nombre FROM Ciclista C
WHERE NOT EXISTS (SELECT * FROM Etapa E
                  WHERE km > 200 AND
                  C.dorsal <> E.dorsal );
```

S'ha hagut de convertir en: "Obtindre el nom del ciclista tal que ***no*** existeix una etapa de més de 200 km. que ell ***no*** haja guanyat"

Equivalències fórmules lògiques generals (CRT) i SQL:

Les expressions lògiques generals (p.ex. CRT) es poden convertir a SQL de la següent manera:

Exp. Lògica		**Pas intermedi**	**SQL**
$p \rightarrow q$	$\Rightarrow$	$\neg p \vee q$	(NOT p) OR q
$\forall x\ p$	$\Rightarrow$	$\neg\exists x \neg p$	NOT EXISTS (…negar….)
$\forall x\ (p \rightarrow q)$	$\Rightarrow$	$\neg\exists x \neg(p \rightarrow q) \equiv \neg\exists x \neg(\neg p \vee q)$ $\equiv \neg\exists x\ (p \wedge \neg q)$	

“Afegits” en consultes amb quantificació universal:

EXEMPLE: Obtindre el nom del ciclista (si n’hi ha) que ha guanyat totes les etapes de més de 200 km.

```
SELECT C.nombre FROM Ciclista C
WHERE NOT EXISTS (SELECT * FROM Etapa E
                  WHERE E.km > 200 AND C.dorsal <> E.dorsal );
```

¿Què passa si no hi han etapes de més de 200 km?

¡¡¡EIXIRIEN TOTS ELS CICLISTES!!!

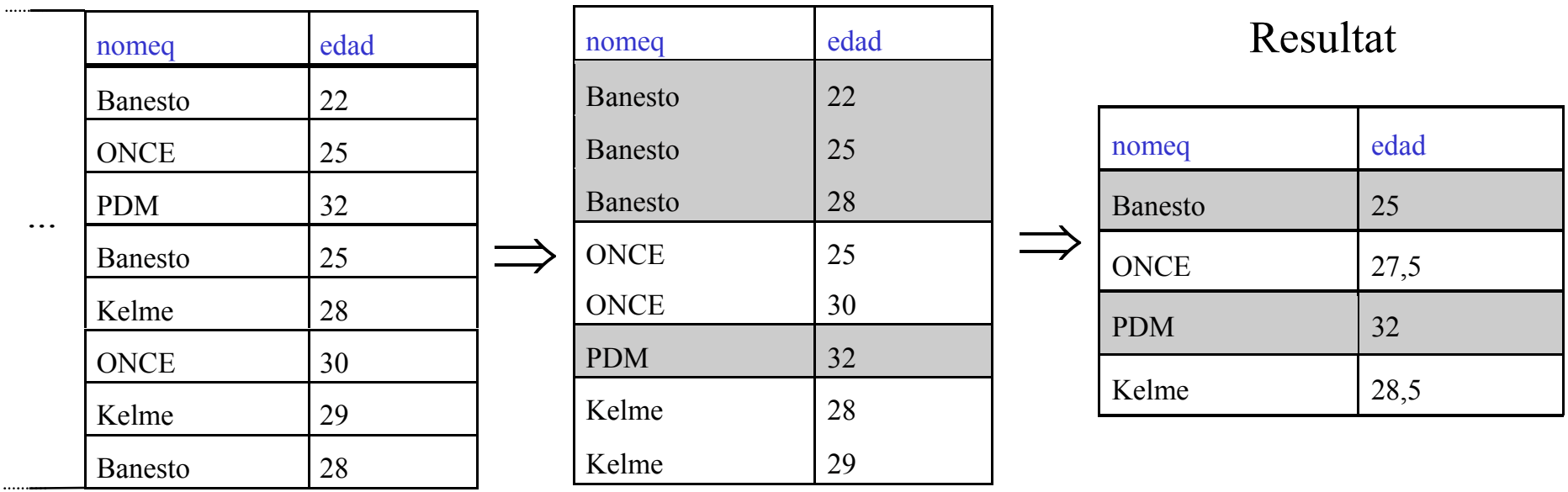
Solució:

```
SELECT C.nombre FROM Ciclista C
  WHERE NOT EXISTS (SELECT * FROM Etapa E
                    WHERE E.km > 200 AND C.dorsal <> E.dorsal )
        AND EXISTS (SELECT * FROM ETAPA E2 WHERE E2.km > 200);
```

CONSULTES COMPLEXES: AGRUPACIÓ

EXEMPLE: Obtindre el nom de **cada** equip i l'edat mitjana de els ciclistes del dit equip:

```
SELECT nomeq, AVG(edad) FROM Ciclista
    GROUP BY nomeq;
```



nomeq	edad
Banesto	22
ONCE	25
PDM	32
Banesto	25
Kelme	28
ONCE	30
Kelme	29
Banesto	28

…

⇒

nomeq	edad
Banesto	22
Banesto	25
Banesto	28
ONCE	25
ONCE	30
PDM	32
Kelme	28
Kelme	29

⇒

Resultat

nomeq	edad
Banesto	25
ONCE	27,5
PDM	32
Kelme	28,5

Relació Selecció-Agrupament

Un grup es pot entendre com un conjunt de files amb el mateix valor per al conjunt de columnes por les quals s'agrupa (les incloses en la clàusula group by).

Les funcions agregades en les consultes agrupades funcionen de forma diferent que en les consultes normals, retornant un valor per cada grup format.

D'una consulta agrupada només es pot realitzar una selecció que retorna un valor, simple o compost, per cada grup format en la consulta.

EXEMPLE INCORRECTE:

```
SELECT nomeq, nombre, AVG(edad) FROM Ciclista
GROUP BY nomeq;
```

La regla sintàctica que apliquen els sistemes relacionals per a assegurar el bon funcionament de les consultes agrupades és la següent:

"en la selecció d'una consulta agrupada, només poden aparéixer referències a columnes per les quals s'agrupa, referències a funcions agregades o literals".

GROUP i WHERE

Si s'inclou la clàusula where l'aplicació d'aquesta clàusula es produeix prèviament a l'agrupació.

```
SELECT nomeq, AVG(edad) FROM Ciclista
WHERE edad > 25
GROUP BY nomeq;
```

GROUP, WHERE i HAVING

La clàusula HAVING, que només pot anar en consultes agrupades, és similar a WHERE però l'ordre és el següent:

1º) Condició WHERE (s'utilitza per a les files)

2º) Agrupament i càlcul de valors agregats

3º) Condició HAVING (s'utilitza per als grups)

En la clàusula HAVING, només podran aparéixer directament referències a columnes per les quals s'agrupen o a funcions agregades.

EXEMPLE: Obtindre el nom de **cada** equip i l'edat mitjana dels seus ciclistes amb més de 25 anys, d'aquells equips amb més de 3 corredors majors de 25 anys.

```
SELECT nomeq, AVG(edad) FROM Ciclista
WHERE edad > 25
GROUP BY nomeq
HAVING COUNT(dorsal) > 3;
```

EXEMPLE: Obtindre el nom del ciclista i el número de ports que ha guanyat, sent la mitjana de la pendent d'aquests superior a 10.

```
SELECT C.nombre, COUNT(P.nompuerto)
FROM Ciclista C, Puerto P
WHERE C.dorsal = P.dorsal
GROUP BY C.dorsal, C.nombre        /* Agrupar sempre per CP */
HAVING AVG (P.pendiente) >10;
```

COMBINACIONS DE TAULES

Existeixen altres formes de combinar vàries taules en consultes i totes elles, juntament a les ja vistes, donen lloc a una "expressió de taula".

Existeixen, en definitiva, vàries formes de combinar dues taules en el llenguatge SQL:

- Incloure vàries taules en la clàusula from.
- Ús de subconsultes en les condicions de les clàusules where o having (o fins i tot en el from).
- **Combinacions conjuntistes de taules**: utilitzen per a combinar les taules operadors de la teoria de conjunts.
- **Concatenacions de taules**: combinen dues taules utilitzant diferents formes variants de l'operador concatenació de l'Àlgebra Relacional.

COMBINACIONS CONJUNTISTES DE TAULES

Corresponen als operadors *unió, diferència* i *intersecció* de l'Àlgebra Relacional.

- UNION
- EXCEPT
- INTERSECT

Permeten combinar taules que tinguen esquemes compatibles.

UNION

expressió_taula union [all] *expressió_taula*

Realitza la unió de les files de les taules provinents de les dues expressions.

Es permetran o no duplicats segons s'incloga o no l'opció all.

EXEMPLE: Obtindre el nom dels ciclistes de 'Banesto' i de la 'ONCE'.

(SELECT nombre FROM Ciclista WHERE nomeq = 'Banesto')
UNION
(SELECT nombre FROM Ciclista WHERE nomeq = 'ONCE')

CONCATENACIONS DE TAULES

Corresponen a variants de l'operador concatenació de l'Àlgebra Relacional.

- Producte cartesià (cross join)
- Concatenació interna

referència_taula [natural] [inner] join *referència_taula*

[on *expressió_condicional* | using (*comallista_columna*)]

- Concatenació externa

referència_taula [natural]

{left [outer] |

right [outer] |

full [outer] } JOIN *referència_taula*

[on *expressió_condicional* | using (*comallista_columna*)]

- Concatenació unió

... FROM T1 UNION JOIN T2 ≡ ... FROM

```
select t1.*, null, null, ..., null from t1
union all
select null, null, ..., null, t2.* from t2
```

ÀLGEBRA RELACIONAL -- SQL

A.R.	SQL ESTÀNDARD	SQL de ORACLE'8
• ∪	• UNION	• UNION
• –	• EXCEPT	• MINUS
• ∩	• INTERSECT	• INTERSECT
• ×	• CROSS JOIN	• (no fa falta, es posa una coma)
• |><|	• NATURAL JOIN	• (no està, es fa =’s en el WHERE)

Altres

• ∪ (dups.)	• UNION ALL	• UNION ALL
• ><	• Left/Right/Full JOIN	• WHERE TX.a1(+) = TY.a2 (eq. right join)
		• WHERE TX.a1 = TY.a2 (+) (eq. left join)
	• UNION JOIN	

INTRODUCCIÓ D'INFORMACIÓ: INSTRUCCIÓ INSERT

insert into taula [(*comallista_columna*)]

{ default values | values (*comallista_àtoms*) | *expressió_taula*}

- Si no s'inclou la llista de columnes s'hauran d'inserir files completes de *taula*.
- Si s'inclou l'opció default values s'inserirà una única fila en la taula amb els valors per defecte apropiats en cada columna (segons la definició de *taula*).
- En l'opció values(*comallista_átomos*) els àtoms vénen donats per expressions escalars.
- En l'opció *expressió_taula*, s'inseriran les files resultants de l'execució de l'expressió (SELECT).

EXEMPLE: Afegir un ciclista de dorsal 101, nom 'Joan Peris', edat 20 anys i de l'equip 'Kelme'.

```
insert into Ciclista
values (101, 'Joan Peris', 20, 'Kelme');
```

MODIFICACIÓ DE LA INFORMACIÓ: INSTRUCCIÓ UPDATE

update taula

set *comallista_assignacions*

[where *expressió_condicional*]

a on una *assignació* és de la forma:

columna = {default | null | *expressió_escalar*}

Si s'inclou la clàusula where només s'aplicarà a les files que facen certa la condició.

EXEMPLE: Incrementar un 10% la pendent del port 'Aitana' en haver-se tancat la carretera que havia en bon estat i ser necessari pujar per una altra pitjor.

UPDATE Puerto SET pendiente = pendiente * 1.10

WHERE nompuerto = 'Aitana' ;

ELIMINACIÓ D'INFORMACIÓN: INSTRUCCIÓ DELETE

delete from taula [where *expressió_condicional*]

Si s'inclou la clàusula where s'eliminaran aquelles que facen certa la condició.

EXEMPLE: Eliminar la informació del ciclista 'M. Induráin' ja que s'ha jubilat.

DELETE FROM Ciclista WHERE nombre = 'M. Induráin';

MÉS EXEMPLES DE L'ESQUEMA 'CICLISME'

1) Obtindre el número de les etapes i la ciutat d'eixida d'aquelles etapes que no tinguen ports de muntanya.

```
SELECT netapa, salida
FROM etapa
WHERE  not exists ( SELECT * FROM puerto
                    WHERE puerto.netapa=etapa.netapa );
```

MÉS EXEMPLES DE L'ESQUEMA 'CICLISME'

2) Obtindre el nom de la ciutat d'eixida i d'arribada de l'etapa a on estiga el port amb major pendent.

```
SELECT e.salida, e.llegada
FROM etapa e, puerto p
WHERE e.netapa=p.netapa AND
    p.pendiente=(select MAX(pendiente) from puerto );
```

MÉS EXEMPLES DE L'ESQUEMA 'CICLISME'

3) ¿Qui és el ciclista més jove? .

```
SELECT nombre
FROM ciclista c
WHERE edad = ( SELECT MIN(edad) FROM ciclista );
```

4) Obtindre el nom dels ciclistes que han guanyat tots els ports d'una etapa i a més han guanyat eixa mateixa etapa. (Ex. 17 butlletí)

```
SELECT c.nombre FROM ciclista c, etapa e
WHERE e.dorsal=c.dorsal AND
NOT EXISTS( SELECT * FROM puerto p
                    WHERE p.netapa=e.netapa
                    AND c.dorsal <> p.dorsal )
AND EXISTS ( SELECT * FROM puerto p
                    WHERE p.dorsal=c.dorsal
                    AND p.netapa=e.netapa );
```

5) Obtindre el color d'aquells maillots que només han estat duts per ciclistes d'un mateix equip.

```
SELECT DISTINCT color FROM maillot m, llevar l, ciclista c
WHERE c.dorsal=l.dorsal AND m.codigo=l.codigo
 AND NOT EXISTS( SELECT * FROM llevar l2, ciclista c2
                        WHERE c2.dorsal=l2.dorsal AND
                         c2.nomeq<>c.nomeq AND l2.codigo=l.codigo);
```

6) Obtindre el nom dels ciclistes que pertanguen a un equip que tinga més de cinc corredors indicant el número d'etapes guanyades per cadascú.

```
SELECT c.nombre, COUNT(*) FROM ciclista c, etapa e
WHERE c.dorsal=e.dorsal AND
                    5<( SELECT COUNT(*) FROM ciclista c2
                        WHERE c2.nomeq=c.nomeq )
GROUP BY c.nombre, c.dorsal;
```

7) Obtindre el nom dels equips que tinguen la mitjana d'edat màxima de tots els equips.

```
SELECT C.nomeq, AVG(C.edad)
FROM ciclista C
GROUP BY C.nomeq
HAVING AVG(C.edad) >= ALL
                        (SELECT AVG(D.edad)
                         FROM ciclista D
                        GROUP BY D.nomeq);
```

8) Nom dels ciclistes que no han dut tots els maillots que ha dut el ciclista de dorsal 1.

```
SELECT c.nombre FROM ciclista c
WHERE EXISTS( SELECT * FROM llevar l
                    WHERE l.dorsal=1 AND
          NOT EXISTS(SELECT * FROM llevar l2
                    WHERE     l2.dorsal=c.dorsal     AND
                              l2.codigo=l.codigo) );
```