



decsai.ugr.es

Universidad de Granada

Fundamentos de Bases de Datos

Grado en Ingeniería Informática

Introducción al SQL: Consulta a tablas



**Departamento de Ciencias de la
Computación e Inteligencia Artificial**

La consulta a tablas

- **Introducción a la sentencia SELECT. Consultas simples de selección y proyección.**
- **Consulta con producto cartesiano: reunión**
- **Consulta con unión, diferencia e intersección.**
- **La sentencia SELECT anidada. Operadores booleanos especiales. Alias**
- **La división en SQL**
- **El uso de funciones de agregación. La cláusula GROUP BY**

Introducción al SELECT

- Forma general

```
SELECT [ALL|DISTINCT] { * | table.* | expre [c_alias]}
```

```
[, {table.*| expre [c_alias]}]... FROM [usuario].tabla [t_alias]
```

```
[,[usuario].tabla [t_alias]]... WHERE condicion
```

```
[CONNECT BY condicion [START BY condicion]]
```

```
GROUP BY expre, [expre]... [HAVING condicion]]
```

```
{UNION|INTERSEC|MINUS} SELECT ...
```

```
[ORDER BY {expre|posicion} [ASC|DESC] [,
```

```
{expre|posicion} [ASC|DESC]
```

```
FOR UPDATE OF columna [,columna]...[NOWAIT]
```

Opciones que no veremos

Opciones avanzadas

Introducción al SELECT

- [ALL|DISTINCT] Permite generar tuplas repetidos o no. Por defecto ALL
- { * | table.* | expre [c_alias] } [, {table.*| expre [c_alias]}]... Es el objetivo de la consulta.
 - * Significa obtener todos los campos de la(s) tabla(s)
 - expre es una expresión con campos la(s) tabla(s).
 - Puede ser un nombre de columna (**realiza la proyección**), o una combinación o función de estas.
 - c_alias permite dar un nombre a la columna de salida
- [ORDER BY {expre|posición} [ASC|DESC] [, {expre|posición} [ASC|DESC]} Permite ordenar la consulta

Introducción al SELECT

- FROM [usuario].tabla [t_alias] [, [usuario].tabla [t_alias]]... indica que tabla(s) se va(n) a utilizar.
 - Si aparece más de una tabla se establece el **producto cartesiano** de las tablas implicadas.
 - Los alias permiten nombrar una tabla de forma distinta, para realizar el producto de una tabla consigo misma o referenciar distintas tuplas de una misma tabla.
- WHERE condicion implica **seleccionar** las tuplas de la tabla resultante que verifican la condición. La condición puede ser tan compleja como se quiera lo que conducirá a “consultas anidadas”

Ejemplos simples

- Selecciona todos los elementos de la tabla alumnos y todos sus atributos
 - **select * from alumnos order by ape1,ape2,nombre ;**
- Selecciona el nombre y los apellidos de los alumnos menores de 25 años.
 - **select nombre,ape1,ape2 from alumnos where edad <=25 order by edad desc, ape1,ape2,nombre;**
- Selecciona el nombre y los apellidos de aquellos alumnos entre 20 y 30 años que son de Andalucia Oriental
 - **select dni,nombre,ape1,ape2 from alumnos where (edad between 20 and 30) and provincia in ('Jaen','Granada','Almeria')) order by ape1,ape2,nombre;**

Ejemplos simples

- Selecciona la lista de todos los curriculums existentes.
 - **select distinct curriculum from asigna order by curriculum ;**
- Selecciona el nombre y los apellidos de los alumnos menores de 25 años matriculados de la asignatura 'bd1s'.
 - **select nombre,ape1,ape2 from alumnos, matricula where (edad >25) and (alumnos.dni=matricula.dni and matricula.codasi#='bd1s') order by ape1,ape2,nombre;**
- Selecciona los nombre de asignaturas optativas de 4.5 o más créditos de las que está matriculado 'Jose Lopez Perez'
 - **select nombreas from alumnos,asigna,matricula where (carácter='op' and credt+credpt>=4.5 and nombre='Jose' and ape1='Lopez' and ape2='Perez' and alumnos.dni=matricula.dni and asi#=codasi#) order by nombreas**

Los operadores conjuntistas

- Forma general
(select
[UNION, INTERSECT, MINUS
(select....)
- Ejemplos:
 - **select dni from alumnos**
minus
select alumnos.dni from alumnos, alumnos al
where (al. edad < alumnos.edad)
Selecciona los alumnos más jóvenes
 - **select asi# from asigna where credit+credpr >6**
intersect
select codasi# from matricula where curso='1998-1999'
Selecciona aquellas asignaturas de más de seis créditos vigentes en el curso 1998-1999

Operad. booleanos adicionales

- Forma general:

[expresion] [not] operador (conjunto)

- La expresion puede ser una sucesion de expresiones o nombres de columnas.
- Los operadores pueden ser:
 - **IN** ya conocido
 - **{= | != | < | > | <= | >=}** **ANY** compara con **cualquier elemento** del conjunto citado y es cierta si se cumple la condicion.
 - **{= | != | < | > | <= | >=}** **ALL** compara con **todos los elementos** del conjunto citado y es cierta si se cumple la condicion.

Operad. booleanos adicionales

- **EXISTS** detecta si el conjunto está no vacío, no hay expresión asociada
 - Los conjuntos asociados pueden ser descritos mediante una sentencia **SELECT** con lo que se obtienen “selects anidados”.
- Ejemplos:
 - **select codal.ape1,ape2,nombre from matricula,alumnos where codasi# in (select asig# from asigna where caracter='op') and matricula.dni=alumnos.dni order by ape1,ape2,nombre;**

Selecciona los alumnos matriculados de alguna asignatura optativa.

Operad. booleanos adicionales

- **Ejemplo**
 - **Select dni from matricula where codasi#=‘bd1s’ and curso_academico=‘2013-2014’ and notas >= all (select notas from matricula);**

Selecciona aquellos alumnos que han obtenido la máxima calificación en bd1s en el curso 2014-2014

Operad. booleanos adicionales

- Ejemplos
 - **select asi#,nombres from asigna where curso>= all(select curso from asigna)**
Selecciona asignaturas de mayor curso
 - **select distinct dni from matricula where codasi# in (select asi# where curso <=all(select curso from asigna))**
Selecciona alumnos matriculados de asignaturas del curso más inferior

Operad. booleanos adicionales

- Ejemplos

- `select dni,ape1,ape2,nombre from alumnos where exists (select * from matricula where dni=codal and codas='db1s');`

Selecciona los alumnos matriculados de bds1, **es una forma adicional de conectar tablas propia del cálculo relacional**

- `select asi#,nombres from asigna where not exists (select * from matricula where asi#=codas);`

Selecciona asignaturas de las que no esta matriculado ningun alumno, **note se la equivalencia con la diferencia**

La division en SQL

- La división utilizando la filosofía del Cálculo Relacional
 - Idea básica

Que todos los elementos cumplan una propiedad es equivalente a que el conjunto de elementos que no la cumplan esté vacío.

- Ejemplos:
 - *Si un alumno está matriculado de todas las asignaturas optativas es lo mismo que si no existe ninguna asignatura optativa de la que no esté matriculado.*

**Select ape1,ape2,nombre from alumnos where
not exists (select asi# from asigna where carácter='op'
and not exists(select * from matricula where
matricula.dni=alumnos.dni and codasi#=asi#))**

La division en SQL

- La división utilizando la filosofía del Cálculo Relacional
 - Ejemplos:

Si en una asignatura están matriculados todos los alumnos de Almería entonces el conjunto de alumnos de Almería de que no están matriculados de esta asignatura está vacío.

**Select asi#,nombres from asigna where
not exists(select dni from alumnos where
provincia='Almeria' and not exists (select * from
matricula where codasi#=asi# and
matricula.dni=alumnos.dni));**

La division en SQL

- La división utilizando la filosofía del Cálculo Relacional
 - Ejemplos:

Encontrar los alumnos que han aprobado todas las asignaturas de primero de Ingenieria Superior.

No existe ninguna asignatura de primero de Ingeniería Superior tal que no exista una matricula del alumno donde esté dicha asignatura y la calificación sea aprobado o mayor

**Select ape1,ape2,nombre from alumnos where
not exists (select asi# from asigna where curso=1 and
curriculum ='Informatica Superior' and not exists (select *
from matricula where matricula.dni=alumnos.dni and
codasi#=asi# and calificacion in('np','no','sb','mh')));**

La division en SQL

- La división utilizando la filosofía del Cálculo Relacional
 - Ejemplos:

Encontrar los alumnos becarios matriculados de todas las asignaturas de más de seis créditos

```
select ape1,ape2,nombre from alumnos where beca='si'
and not exists (select asi# from asigna where credit
+credpr>6 and not exists (select * from matricula where
matricula.dni=alumnos.dni and asi#=codasi#))
```

Encontrar aquellas asignaturas que aparecen en todos los cursos academicos

```
Select asi#,nombreas from asigna where not exists
(select matricula.curso_academico from matricula where
not exists (select * from matricula ma where
asi#=ma.codasi# and
matricula.curso_academico=ma.curso_academico));
```

La division en SQL

La división utilizando la filosofía del Algebra Relacional

- Idea Básica:

Sean $D=R \div S$ y r,s,d instancias de $D\ R\ y\ S$

$$\forall a \in D ; s \subseteq d(a) = \{b \in S / (a,b) \in R\}$$

para detectar la inclusion:

$$s \subseteq d(a) \Rightarrow s - d(a) = \emptyset \Rightarrow \text{not exists}(s - d(a))$$

Alumnos matriculados de todas las asignaturas optativas

**select ape1,ape2,nombre from alumnos where not exists
((select asi# from asigna where caracter='op') minus
(select codasi# from matricula where
matricula.dni=alumnos.dni))**

La division en SQL

La división utilizando la filosofía del Algebra Relacional

– Ejemplos:

- *Asignaturas en que estan matriculados todos los alumnos de Almeria*

```
select asi#,nombreas from asigna where not exists
((select dni from alumnos where provincia='Almeria')
minus (select codal from matricula where asi#=codas))
```

- *Alumnos que han aprobado todas las asignaturas de primero de Ingenieria Superior*

```
select ape1,ape2,nombre from alumnos where not exists
((select asi# from asigna where curso=2 and
curriculum ='Informatica Superior') minus
(select codas from matricula where dni=codal
and calificacion in ('ap','no','sb','mh')))
```

La division en SQL

La división utilizando la filosofía del Algebra Relacional

- *Alumnos becarios matriculados de todas las asignaturas de más de seis creditos*

```
select ape1,ape2,nombre from alumnos
where beca='si' and
not exists((select asi# from asigna where credit+credpr>6)
minus (select codasi# from matricula where
matricula.dni=alumnos.dni))
```

- *Asignaturas que aparecen en todos los cursos*

```
Select asi#,nombreas from asigna where not exists (
(select matricula.curso_academico from matricula)
minus
(select ma.curso_academico from matricula ma where
asi#=ma.codasi#))
```

Funciones de Agregación

- **Idea básica:**

Utilizar funciones cuyo resultado sea un “resumen” de los datos de una columna de una tabla.

- Forma general: `funcion(expresión)`

- **Funciones existentes:**

- `AVG(.)` calcula la media de la expresión dada,

- `STDDEV(.)` calcula la desviación típica,

- `VARIANCE(.)` calcula la varianza. Ignoran valores nulos:

- ```
select avg(nota), stddev(nota) from alumnos where
sexo='v'
```

- `MIN(.)` calcula el mínimo de la expresión dada,

- `MAX` calcula el máximo

- ```
select min(credit+credpr),max(credit+credpr) from asigna
```

Funciones de Agregación

- **Funciones existentes:**
 - COUNT(*expresion*) calcula el numero de filas donde la expresión es no nula.

Select count(calificacion) from matricula

- COUNT(*) calcula el numero total de filas de una tabla. No tiene en cuenta las filas distintas

select count(*) from matricula

- **Otros usos:** Se pueden combinar las funciones de agregacion con los operadores ALL y ANY.

- Ejemplos

**select ape1,ape2,nombre,edad from alumnos where
edad=any(select min(edad) from alumnos);**

Calcula los alumnos de edad minima

Funciones de Agregación

- **Otros usos:**

```
select ape1,ape2,nombrenota from alumnos where  
nota<=any(select avg(nota) from alumnos);
```

Calcula los alumnos con nota menor o igual a la media

```
select asi#,nombresas,cred+credpr from asigna where  
(cred+credpr)=any(select max(cred+credpr) from  
asigna);
```

Calcula las asignaturas con numero maximo de creditos

```
select ape1,ape2,nombre from alumnos where  
15<=any(select count(*) from matricula where  
matricula.dni=alumnos.dni);
```

Calcula los alumnos matriculados de mas de 15 asignaturas

```
select asi#,nombresas from asigna where 25<=any(select  
count(*) from matricula where asi#=codasi#);
```

Calcula las asignaturas con más de 15 alumnos

La Cláusula GROUP BY

- **Idea básica:**

Obtener “tablas resumen” donde cada fila corresponda al valor de uno o varios atributos y las columnas sean funciones de agregación que resuman dicho atributos.

- **Ejemplos de dichas tablas:**

sexo	avg(edad)	carácter	curso	count(así#)
v	----	tr	1	----
m	----	tr	2	----
	----	tr	3	
.

ob	1	1	1	----
ob	2	2	2	----

- Nótese que en las columnas sólo aparecen los atributos que “resumen” (agrupan) y funciones de agregación

La Cláusula GROUP BY

- **Forma general:**

```
SELECT expre,[expre]...from tabla,tabla...
WHERE condicion .....
GROUP BY expre, [expre]... [HAVING condicion]]
```

- Las expresiones detrás de “select” describen el esquema de la tabla resumen (*solo atributos que agrupan y funciones de agregación*)
- La condición detrás de “where” restringe las tablas “de entrada” (*involucra atributos de las tablas originales*)
- Las expresiones detrás de “group by” definen los atributos que agrupan (*deben coincidir con los del “select”*)
- La condición detrás de “having” restringe la tabla “de salida” (*involucra columnas que aparecen detrás del select*)

La Clausula GROUP BY

- **Ejemplos:**

- **select sexo, avg(edad) from alumnos group by sexo order by sexo**
- **select caracter,curso,count(asi#) from asigna group by carácter, curso order by carácter,curso**

Obtienen las dos tablas que aparecen en los ejemplos iniciales

- **select curso_academico,codasi#,count(*) from matricula group by curso_academico,codasi# order by curso_academico, codasi#**

Obtiene el numero de alumnos matriculados en cada curso en cada asignatura

- **Select dni,count(*) from matricula where calificacion in ('ap','no','sb','mh') group by dni order by dni**

Obtiene el numero de asignaturas que tiene aprobadas cada alumnos

La Clausula GROUP BY

- **select sexo, avg(edad) from alumnos group by sexo order by sexo having sexo =‘v’;**

Obtiene la edad media de los alumnos varones

- **select curso_academico,codasi#,count(*) from matricula group by curso_academico,codasi# order by curso_academico,codasi# having count(*)>=10**

Obtiene el numero de alumnos matriculados en cada curso en cada asignatura siempre que haya más de 10 alumnos.

- **Consideraciones adicionales:**

El calculo y mantenimiento de resúmenes es una de las actividades a las que se han dedicado más atención en los últimos tiempos y es la base de del concepto de “OLAP” o “data cube” y de los “almacenes de datos” o “data warehouse”