


BASES DE DATOS SQL



ESCUELA DE INGENIERÍA INFORMÁTICA

Contenidos

- 
- ⌘ 1. Introducción a SQL – INSERT, UPDATE, DELETE y SELECT
 - ⌘ 2. Selección – WHERE
 - ⌘ 3. Ordenación de tuplas – ORDER BY, OFFSET y FETCH
 - ⌘ 4. Operaciones sobre conjuntos - UNION, INTERSECT y MINUS
 - ⌘ 5. Subconsultas anidadas - IN/NOT IN, SOME/ALL y EXISTS/NOT EXISTS
 - ⌘ 6. Producto – INNER JOIN ... ON y LEFT/RIGHT JOIN
 - ⌘ 7. Funciones de agregación – GROUP BY y HAVING
 - ⌘ 8 . Combinación de consultas e inserción, modificación o borrado de datos
 - ⌘ 9 . Vistas – CREATE VIEW

Introducción a SQL – INSERT, UPDATE, DELETE y SELECT

- 🌀 Lenguaje estructurado de consulta – (Structured Query Language)
- 🌀 Se compone de sentencias que permiten **crear y gestionar** (CRUD) los datos en bases de datos relacionales
- 🌀 Obtención de datos mediante ejecución de sentencias de consulta SQL. Por ejemplo:

Nombre	Apellido	Edad
Ana	Alonso	21
Beatriz	López	22
Beatriz	Gálvez	23
Daniel	Alonso	20

```
SELECT nombre, apellido  
FROM alumnos  
WHERE edad >22
```

Nombre	Apellido
Beatriz	López
Beatriz	Gálvez

Introducción a SQL – INSERT, UPDATE, DELETE y SELECT

∞ **INSERT:** inserta *una tupla o registro en la tabla indicada

- Sintaxis

```
INSERT INTO <NombreTabla> [Ai]  
VALUES <valores>
```

A_i – los atributos a insertar, entre paréntesis y separados por comas. Si se omite se entiende que en Values se indicarán los valores de todos los atributos de la tabla

Valores – los valores a insertar en las columnas, entre paréntesis y separados por comas. Deben seguir el mismo orden que el indicado en VALUES, en caso de utilizarlo

∞ **Ejemplo.** Suponiendo una tabla *Alumnos* con los atributos: *Id*, *Nombre* y *Edad*, las sentencias siguientes serían equivalentes:

```
INSERT INTO alumnos (nombre, edad, id) VALUES ('JAIME ROCA', 23, 1)  
INSERT INTO alumnos VALUES (1, 'JAIME ROCA', 23)
```

***Nota:** como veremos más adelante, INSERT permite inserción múltiple de registros, obtenidos como resultado de una consulta.

Introducción a SQL – INSERT, UPDATE, DELETE y SELECT

∞ **UPDATE:** actualiza los valores de los atributos indicados de una tabla

○ Sintaxis:

```
UPDATE <NombreTabla>
```

```
SET NombreAtributo1=<Expresión1> [, NombreAtributo2=<Expresión2>, ...]
```

```
[WHERE <Predicado>]
```

Expresión_i - el resultado de la misma será el valor que se asignará al atributo
NombreAtributo_i

predicado - predicado lógico, de forma que solo se actualizarán las tuplas que lo cumplan

∞ Ejemplo. Modificar el nombre del alumno con id=1 con el valor 'JAIME ROCAS'

```
UPDATE alumnos
```

```
SET nombre = 'JAIME ROCA'
```

```
WHERE ID=1
```

Introducción a SQL – INSERT, UPDATE, DELETE y SELECT

∞ **DELETE:** elimina las tuplas de una tabla que cumplan con el predicado indicado

○ Sintaxis:

```
DELETE FROM <NombreTabla>  
WHERE <Predicado>
```

predicado - predicado lógico, de forma que solo se actualizarán las tuplas que lo cumplan

∞ Ejemplo. Eliminar el alumno con id=1

```
DELETE FROM alumnos  
WHERE ID=1
```

Introducción a SQL – INSERT, UPDATE, DELETE y SELECT

∞ **SELECT:** obtiene las tuplas de la tabla indicada que cumplen con el predicado

- Sintaxis general de una consulta SQL

```
SELECT [ALL | DISTINCT] A1[, A2, ..., An]  
FROM r1[, r2, ..., rm] [WHERE <predicado>]
```

ALL - muestra todas las tuplas de la relación, aunque estén duplicadas (es el valor por defecto)

DISTINCT - muestra las tuplas de la relación, sin duplicados

A_i - los atributos a consultar. Admite operaciones con dichos atributos (suma, resta, ...) y funciones

r_i - las relaciones o tablas a consultar

predicado - predicado lógico, de forma que solo se mostrarán las tuplas que lo cumplan

```
SELECT ALL nombre  
FROM alumnos  
WHERE edad >22
```

Nombre
Beatriz
Beatriz

```
SELECT DISTINCT nombre  
FROM alumnos  
WHERE edad >22
```

Nombre
Beatriz

```
SELECT nombre AS nombre_alumno  
FROM alumnos al  
WHERE al.edad >22
```

Nombre_alumno
Beatriz
Beatriz

Introducción a SQL – INSERT, UPDATE, DELETE y SELECT

∞ En lo sucesivo utilizaremos las siguientes tablas de una base de datos para ilustrar con ejemplos el funcionamiento de las distintas cláusulas:

`alumno=(@nmat, nombre)`

`titulacion=(@codigo, creditos, escuela)`


`asignatura=(@nombre, creditos, codigoTitulacion)`

`matricula=(@nmatAlumno, @nombreAsignatura)`

`empresa=(@CIF, siglas, numempleados, numdirectivos)`

`practica=(@nmatAlumno, @CIFEmpresa)`

Contenidos

- ∞ 1. Introducción a SQL – INSERT, UPDATE, DELETE y SELECT
-  ∞ 2. Selección – WHERE
- ∞ 3. Ordenación de tuplas – ORDER BY
- ∞ 4. Operaciones sobre conjuntos - UNION, INTERSECT y MINUS
- ∞ 5. Subconsultas anidadas - IN/NOT IN, SOME/ALL y EXISTS/NOT EXISTS
- ∞ 6. Producto – INNER JOIN ... ON y LEFT/RIGHT JOIN
- ∞ 7. Funciones de agregación – GROUP BY y HAVING
- ∞ 8. Combinación de consultas e inserción, modificación o borrado de datos
- ∞ 9 . Vistas – CREATE VIEW

Selección - Where

∞ Filtra las tuplas mostrando aquellas que cumplen con el predicado indicado

- Sintaxis: **WHERE** <predicado>

donde <predicado> ::= <predicado> **OR** <predicado> | <predicado> **AND** <predicado> | **NOT** <predicado>

∞ Los **operadores** de relación que se pueden utilizar son =, <>, <, <=, >, >=


∞ Las **constantes cadena** se delimitan por comillas simples.

∞ Los **operadores** de relación aplicados a cadenas de texto las comparan alfabéticamente

∞ El operador lógico NOT tiene más prioridad que el AND y éste a su vez tiene más prioridad que el OR, utilizar paréntesis para deshacer ambigüedades

Ejemplo: Titulaciones impartidas en la Facultad de Ciencias y con más de 120 créditos

```
SELECT * FROM titulación WHERE escuela='FC' AND credits>120
```



Cod.	cred.	escuela
1	130	FC
2	100	FC
3	125	FC
4	130	FD

titulación

Cod.	cred.	escuela
1	130	FC
3	125	FC

Selección - Where

☞ Otros operadores:

☞ El operador **LIKE** retorna verdadero si la cadena hace match con el patrón

- Sintaxis: **WHERE** <atributo> **[NOT] LIKE** <patrón>


☞ Donde patrón es una cadena de texto que puede contener los caracteres siguientes:

- **%** match con **cualquier subcadena**. Ejemplo: 'F%' match con cadenas que comiencen por F como por ejemplo: 'FCO' o 'FC'
- **_** match con **cualquier carácter** (solo un carácter). Ejemplo: 'F_' match con cadenas de dos caracteres que comiencen por F como por ejemplo: 'FC' o 'FI'
- **\X** match con el carácter X, que puede ser cualquiera

☞ El operador **IS NULL** retorna verdadero si el valor del atributo especificado es nulo

- Sintaxis: **WHERE** <atributo> **IS [NOT] NULL**

Contenidos

- ☞ 1. Introducción a SQL – INSERT, UPDATE, DELETE y SELECT
- ☞ 2. Selección – WHERE
-  ☞ 3. Ordenación de tuplas – ORDER BY, OFFSET y FETCH
- ☞ 4. Operaciones sobre conjuntos - UNION, INTERSECT y MINUS
- ☞ 5. Subconsultas anidadas - IN/NOT IN, SOME/ALL y EXISTS/NOT EXISTS
- ☞ 6. Producto – INNER JOIN ... ON y LEFT/RIGHT JOIN
- ☞ 7. Funciones de agregación – GROUP BY y HAVING
- ☞ 8. Combinación de consultas e inserción, modificación o borrado de datos
- ☞ 9 . Vistas – CREATE VIEW

Ordenación y limitación — Order by, Offset y Fetch

🔗 Ordena las tuplas mostradas en función de las claves de ordenación indicadas

- Sintaxis:

```
ORDER BY {atributo|posiciónAtributo|expresion} [DESC|ASC] [NULLS FIRST| NULLS LAST]
        [ ,atributo|posicionAtributo|expresion [DESC|ASC] [NULLS FIRST| NULLS LAST]
        ]
```

- Siendo:

- **atributo**, la columna o campo por el que se quieren ordenar los registros
- **posicionAtributo**, valor entero que identifica el número de la columna por la que se desea ordenar. Debe ser un valor mayor que 0 y menor o igual que el número de columnas que figuran en la select.
- **expresión**, una expresión que puede devolver valores string, numéricos, datetime, etc.
- **[DESC|ASC]**, la forma de indicar que los resultados deben ser devueltos en orden descendente o ascendente.
- **[NULLS FIRST| NULLS LAST]**, la forma de indicar que los valores NULL deben ser devueltos antes (NULLS FIRST) o después (NULLS LAST) que los valores no-NULL.

- Por defecto ORDER BY ordena de manera ascendente

- Cuando hay varios campos se ordena por el primero en la cláusula ORDER BY y si hay empate se ordenan por la siguiente, así hasta el final

Ordenación y limitación — Order by, Offset y Fetch

1. Ejemplo : Titulaciones impartidas en la Facultad de Ciencias con más de 120 créditos y ordenadas por número de créditos:

```
SELECT * FROM titulacion
```

```
WHERE escuela='FC' AND credits>120 ORDER BY credits // ordena asc por créditos
```

```
SELECT * FROM titulacion
```

```
WHERE escuela='FC' AND credits>120 ORDER BY 2 // ordena asc por créditos. Utiliza la posición de la columna dentro
```




cód.	créd.	escuela
1	130	FC
2	100	FC
3	125	FC
4	130	FD
5	null	FD
6	null	FI
7	150	FI

cód.	créd.	escuela
3	125	FC
1	130	FC

2. Ejemplo: Mostrar el promedio de créditos impartido en cada Escuela ordenados de mayor a menor

```
SELECT escuela, AVG(credits)promedio FROM titulacion GROUP BY escuela
```

```
ORDER BY promedio DESC // uso de una expresión para ordenar
```

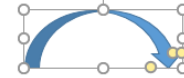


cód.	créd.	escuela
1	130	FC
2	100	FC
3	125	FC
4	130	FD
5	null	FD
6	null	FI
7	150	FI

escuela	promedio
FI	150
FD	130
FC	118,33

Ordenación y limitación — Order by, Offset y Fetch

3. Ejemplo: Mostrar todos los datos de las titulaciones ordenadas por el número de créditos. Las titulaciones cuyos créditos sean nulos deben aparecer al principio del listado.



```
SELECT *  
FROM titulación  
ORDER BY créditos NULLS FIRST
```

cód.	créd.	escuela
1	130	FC
2	100	FC
3	125	FC
4	130	FD
5	null	FD
6	null	FI
7	150	FI

cód.	créd.	escuela
5	null	FD
6	null	FI
2	100	FC
3	125	FC
1	130	FC
4	130	FD
7	150	FI

4. Ejemplo: Mostrar un listado con todas las titulaciones ordenadas por el nombre de la escuela de forma ascendente y por el número de créditos de forma descendente. Los valores nulos de los créditos deben aparecer al final

```
SELECT * FROM titulación  
ORDER BY escuela, créditos DESC NULLS LAST // ordena en primer lugar por la escuela y si se repite el  
nombre de la escuela ordena por los créditos de forma descendente.
```



cód.	créd.	escuela
1	130	FC
2	100	FC
3	125	FC
4	130	FD
5	null	FD
6	null	FI
7	150	FI

cód.	créd.	escuela
1	130	FC
3	125	FC
2	100	FC
4	130	FD
5	null	FD
7	150	FI
6	null	FI

Ordenación y limitación — Order by, Offset y Fetch

☞ La cláusula **OFFSET** indica el número de tuplas a omitir en el resultado de una consulta:

- Sintaxis: **OFFSET** [n] **<ROW | ROWS>**

☞ La cláusula **FETCH** limita el número de tuplas que devuelve una consulta:

- Sintaxis: **FETCH** **<FIRST | NEXT>** [n | n PERCENT] **<ROW | ROWS>** **<ONLY | WITH TIES>**


☞ Donde:

- n número de tuplas o el porcentaje de tuplas si se usa con PERCENT
- **FIRST** y **NEXT** son sinónimos y hay que utilizar uno de ellos obligatoriamente
- **ROW** y **ROWS** son sinónimos y hay que utilizar uno de ellos obligatoriamente
- **ONLY** recupera exactamente el número de tuplas n o el n% de ellas si se usa PERCENT
- **WITH TIES** con ORDER BY retorna tuplas adicionales si tienen el mismo valor que la clave de ordenación de la última fila obtenida.

☞ Ejemplos:

- `SELECT * FROM ALUMNOS ORDER BY edad FETCH FIRST 10 ROWS ONLY`
- `SELECT * FROM ALUMNOS ORDER BY edad OFFSET 100 FETCH FIRST 10 ROWS WITH TIES`

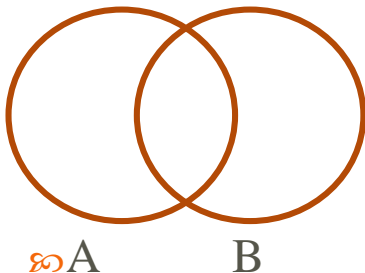
Contenidos

- ⌘ 1. Introducción a SQL – INSERT, UPDATE, DELETE y SELECT
- ⌘ 2. Selección – WHERE
- ⌘ 3. Ordenación de tuplas – ORDER BY, OFFSET y FETCH
-  ⌘ 4. Operaciones sobre conjuntos - UNION, INTERSECT y MINUS
- ⌘ 5. Subconsultas anidadas - IN/NOT IN, SOME/ALL y EXISTS/NOT EXISTS
- ⌘ 6. Producto – INNER JOIN ... ON y LEFT/RIGHT JOIN
- ⌘ 7. Funciones de agregación – GROUP BY y HAVING
- ⌘ 8. Combinación de consultas e inserción, modificación o borrado de datos
- ⌘ 9 . Vistas – CREATE VIEW

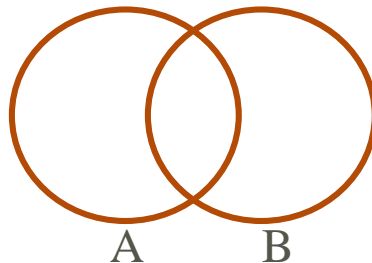
Operaciones sobre conjuntos – UNION, INTERSECT y MINUS

- Los resultados de las consultas se pueden tratar como conjuntos de elementos
- Para poder aplicar estos operadores de conjuntos todas las relaciones tienen que ser compatibles, mismo número de atributos y del mismo tipo
- Estas operaciones siempre eliminan duplicados, pero se puede añadir a la instrucción la palabra reservada ALL y conservaría los duplicados

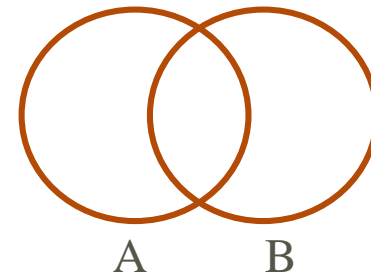
A UNION B



A INTERSECT B



A MINUS B



Operaciones sobre conjuntos – UNION, INTERSECT y MINUS

∞ **UNION**: obtiene la unión de tuplas de dos consultas

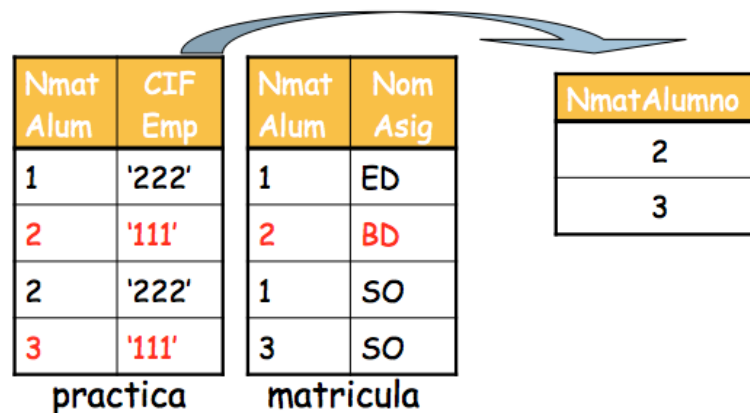
○ Sintaxis: <consulta> **UNION** <consulta>

∞ Ejemplo: obtener el número de matrícula de los alumnos que hagan prácticas en la empresa cuyo CIF sea '111' o estén matriculados en BD

```
(SELECT nmatAlumno FROM practica WHERE CIFEmpresa='111')
```

UNION

```
(SELECT nmatAlumno FROM matricula WHERE nomAsig='BD')
```



Operaciones sobre conjuntos – UNION, INTERSECT y MINUS

∞ **INTERSECT**: obtiene la intersección de tuplas de dos consultas

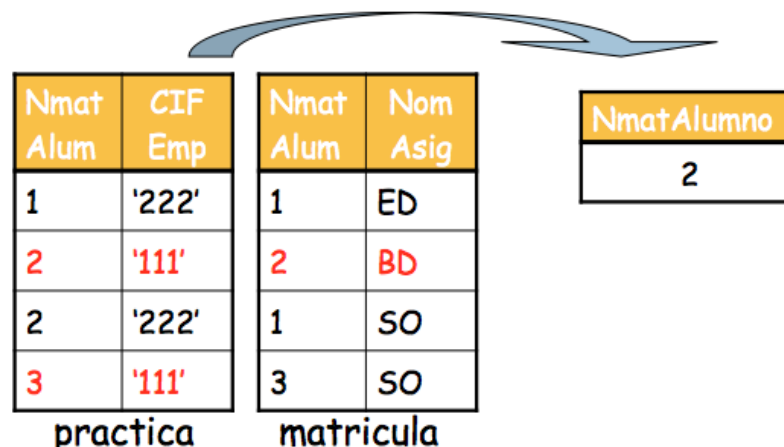
○ Sintaxis: <consulta> **INTERSECT** <consulta>

∞ Ejemplo: obtener el número de matrícula de los alumnos que hagan prácticas en la empresa cuyo CIF sea '111' y estén matriculados en BD

```
(SELECT nmatAlumno FROM practica WHERE CIFEmpresa='111')
```

INTERSECT

```
(SELECT nmatAlumno FROM matricula WHERE nomAsig='BD')
```



Operaciones sobre conjuntos – UNION, INTERSECT y MINUS

∞ **MINUS**: obtiene las tuplas de la primera consulta que no están en la segunda

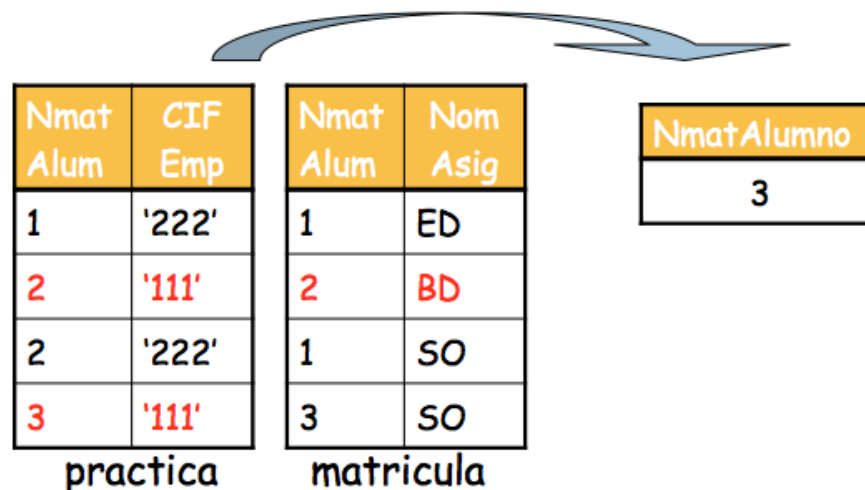
○ Sintaxis: <consulta> **MINUS** <consulta>

∞ Ejemplo: obtener el número de matrícula de los alumnos que hagan prácticas en la empresa cuyo CIF sea '111' pero que no estén matriculados en BD


```
(SELECT nmatAlumno FROM practica WHERE CIFEmpresa='111')
```

MINUS

```
(SELECT nmatAlumno FROM matricula WHERE nomAsig='BD')
```



Contenidos

- ☞ 1. Introducción a SQL – INSERT, UPDATE, DELETE y SELECT
- ☞ 2. Selección – WHERE
- ☞ 3. Ordenación de tuplas – ORDER BY, OFFSET y FETCH
- ☞ 4. Operaciones sobre conjuntos - UNION, INTERSECT y MINUS
-  ☞ 5. Subconsultas anidadas - IN/NOT IN, SOME/ALL y EXISTS/NOT EXISTS
- ☞ 6. Producto – INNER JOIN ... ON y LEFT/RIGHT JOIN
- ☞ 7. Funciones de agregación – GROUP BY y HAVING
- ☞ 8. Combinación de consultas e inserción, modificación o borrado de datos
- ☞ 9 . Vistas – CREATE VIEW

Subconsultas anidadas – IN/NOT IN, SOME/ALL y EXISTS/NOT EXISTS

∞ Se trata de consultas que en la cláusula WHERE indican una condición que conlleva comparar con un conjunto de valores resultado de una consulta.


∞ **IN / NOT IN**: La cláusula IN y NOT IN se puede usar para escribir consultas equivalentes a la intersección y a la diferencia, respectivamente.

○ Sintaxis: **WHERE** <atributo> **IN** | **NOT IN** <subconsulta>

Donde <atributo> y el atributo de la subconsulta tienen que ser de un tipo compatible

∞ Ejemplo **IN**: obtener el número de matrícula de los alumnos que hagan prácticas en la empresa cuyo CIF sea '111' y matriculados en BD

```
SELECT nmatAlumno FROM practica WHERE CIFEmpresa='111' AND nmatAlumno  
IN (SELECT nmatAlumno FROM matricula WHERE nomAsig='BD')
```



Nmat Alum	CIF Emp
1	'222'
2	'111'
2	'222'
3	'111'

practica

Nmat Alum	Nom Asig
1	ED
2	BD
1	SO
3	SO


matricula

NmatAlumno
2

Subconsultas anidadas – IN/NOT IN, SOME/ALL y EXISTS/NOT EXISTS

∞ Ejemplo **NOT IN**: obtener el número de matrícula de los alumnos que hagan prácticas en la empresa cuyo CIF sea '111' pero que no estén matriculados en BD

```
SELECT nmatAlumno FROM practica WHERE CIFEmpresa='111' AND nmatAlumno
NOT IN
(SELECT nmatAlumno FROM matricula WHERE nomAsig='BD')
```



Nmat Alum	CIF Emp
1	'222'
2	'111'
2	'222'
3	'111'

practica

Nmat Alum	Nom Asig
1	ED
2	BD
1	SO
3	SO

matricula

NmatAlumno
3

Subconsultas anidadas – IN/NOT IN, SOME/ALL y EXISTS/NOT EXISTS

∞ **SOME / ALL**: Se utiliza para determinar si un valor de un atributo es mayor que alguno del conjunto (SOME) de tuplas o que todos los del conjunto (ALL) de tuplas.

- Sintaxis: **WHERE** <atributo> <operador de comparación> **SOME** | **ALL** <subconsulta>

∞ Ejemplo **SOME**: obtener el nombre de las asignaturas de la titulación GIITIN que tienen un número de créditos igual o mayor que algunas las de la titulación GITELE

```
SELECT Nombre FROM asignatura
```

```
WHERE titulacion='GIITIN' AND credits >= SOME (SELECT credits FROM  
asignatura WHERE titulacion='GITELE')
```



nombre	créditos	codigoTit
TC	6	GITELE
BD	12	GIITIN
FI	6	GIITIN
CD	9	GITELE
CPM	3	GIITIN

asignatura

nombre
BD
FI

Subconsultas anidadas – IN/NOT IN, SOME/ALL y EXISTS/NOT EXISTS

∞ Ejemplo **ALL**: obtener el nombre de las asignaturas de la titulación GIITIN que tienen más o igual número de créditos que todas las de la titulación GITELE

```
SELECT Nombre FROM asignatura
WHERE titulacion='GIITIN' AND credits >= ALL (SELECT credits FROM
asignatura WHERE titulacion='GITELE')
```



nombre	créditos	codigoTit
TC	6	GITELE
BD	12	GIITIN
FI	6	GIITIN
CD	9	GITELE
CPM	3	GIITIN

asignatura

nombre
BD

Subconsultas anidadas – IN/NOT IN, SOME/ALL y EXISTS/NOT EXISTS

∞ **EXISTS / NOT EXISTS**: Comprueba si la subconsulta (o el conjunto) es vacía. La subconsulta hará referencia habitualmente al valor de un campo de la tupla que trata el SELECT externo. En cierta manera es equivalente al IN y NOT IN.

- Sintaxis: **WHERE EXISTS | NOT EXISTS** <subconsulta>

∞ Ejemplo **EXISTS**: obtener el número de matrícula de los alumnos que hagan prácticas en la empresa cuyo CIF sea '111' y matriculados en BD

```
SELECT nmatAlumno FROM practica p WHERE CIFEmpresa='111' AND EXISTS  
(SELECT * FROM matricula m WHERE nomAsig='BD'  
AND m.nmatAlumno = p.nmatAlumno)
```

Nmat Alum	CIF Emp
1	'222'
2	'111'
2	'222'
3	'111'

practica

Nmat Alum	Nom Asig
1	ED
2	BD
1	SO
3	SO


matricula

NmatAlumno
2

Subconsultas anidadas – IN/NOT IN, SOME/ALL y EXISTS/NOT EXISTS

∞ Ejemplo **NOT EXISTS**: obtener el número de matrícula de los alumnos que hagan prácticas en la empresa cuyo CIF sea '111' pero que no estén matriculados en BD

```
SELECT nmatAlumno FROM practica p
WHERE CIFEmpresa='111' AND NOT EXISTS
(SELECT * FROM matricula m
WHERE nomAsig='BD' AND m.nmatAlumno = p.nmatAlumno)
```



Nmat Alum	CIF Emp
1	'222'
2	'111'
2	'222'
3	'111'


practica

Nmat Alum	Nom Asig
1	ED
2	BD
1	SO
3	SO

matricula

NmatAlumno
3

Contenidos

- ⌘ 1. Introducción a SQL – INSERT, UPDATE, DELETE y SELECT
- ⌘ 2. Selección – WHERE
- ⌘ 3. Ordenación de tuplas – ORDER BY, OFFSET y FETCH
- ⌘ 4. Operaciones sobre conjuntos - UNION, INTERSECT y MINUS
- ⌘ 5. Subconsultas anidadas - IN/NOT IN, SOME/ALL y EXISTS/NOT EXISTS
-  ⌘ 6. Producto – INNER JOIN ... ON y LEFT/RIGHT JOIN
- ⌘ 7. Funciones de agregación – GROUP BY y HAVING
- ⌘ 8. Combinación de consultas e inserción, modificación o borrado de datos
- ⌘ 9. Vistas – CREATE VIEW

Producto – INNER JOIN ... ON y LEFT/RIGHT JOIN

∞ **Producto interno:** se trata de obtener las tuplas resultado de combinar las tuplas de dos tablas mediante el producto cartesiano y realizar una selección forzando la igualdad de los atributos comunes en ambas tablas.

∞ Dichos atributos comunes han de ser del mismo tipo.

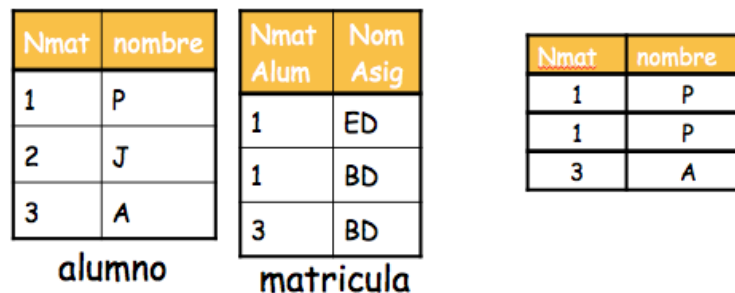
○ Sintaxis: FROM <tabla1> **INNER JOIN** <tabla2>

ON <AtributoComúnTabla1> = <AtributoComúnTabla2>

∞ Ejemplo: obtener el número de matricula y el nombre de los alumnos matriculados en alguna asignatura

SELECT nmat, nombre **FROM** alumno

INNER JOIN matricula **ON** nmat=nmatalumno



Producto – INNER JOIN ... ON y LEFT/RIGHT JOIN

∞ **Producto externo:** se trata de obtener las mismas tuplas que el producto (reunión) natural pero añadiendo aquellas tuplas que no están relacionadas con ninguna otra.

∞ Las uniones externas por la izquierda o derecha se realizan sustituyendo el operador INNER por LEFT o RIGHT respectivamente.

○ Sintaxis:

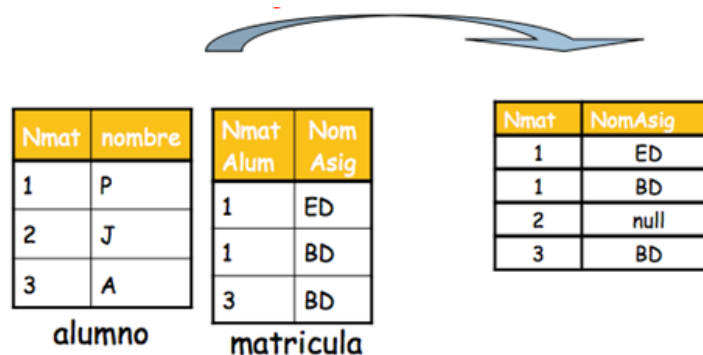
<Tabla1> **RIGHT JOIN** <Tabla2> **ON** <AtributoComúnTabla1> = <AtributoComúnTabla2>

<Tabla1> **LEFT JOIN** <Tabla2> **ON** <AtributoComúnTabla1> = <AtributoComúnTabla2>

∞ **Ejemplo:** obtener el número de matrícula de todos los alumnos y para aquellos que estén matriculados el nombre de las asignaturas de las que están matriculados

SELECT nmat, nomAsig **FROM** alumno **LEFT JOIN** matricula **ON** nmat=nmatalumno

SELECT nmat, nomAsig **FROM** matricula **RIGHT JOIN** alumno **ON** nmat=nmatalumno



Producto – INNER JOIN ... ON y LEFT/RIGHT JOIN

∞ **Variantes naturales:** con los productos internos y externos indicados anteriormente se puede utilizar la variante “NATURAL” de modo que no es necesario indicar con el ON los atributos que se utilizan para unir las tablas, se considerarán automáticamente los atributos con el mismo nombre. En el resultado no hay columnas duplicadas.

- Sintaxis:

```
<tabla1> NATURAL INNER JOIN <tabla2>
```

```
<Tabla1> NATURAL RIGHT JOIN <Tabla2>
```


```
<Tabla1> NATURAL LEFT JOIN <Tabla2>
```

∞ **Ejemplo:** obtener el número de matrícula de todos los alumnos y para aquellos que estén matriculados el nombre de las asignaturas de las que están matriculados, suponiendo que el atributo del número de matrícula tiene el mismo nombre en ambas tablas:

```
SELECT nmat, nomAsig FROM alumno NATURAL LEFT JOIN matricula
```

```
SELECT nmat, nomAsig FROM matricula NATURAL RIGHT JOIN alumno
```

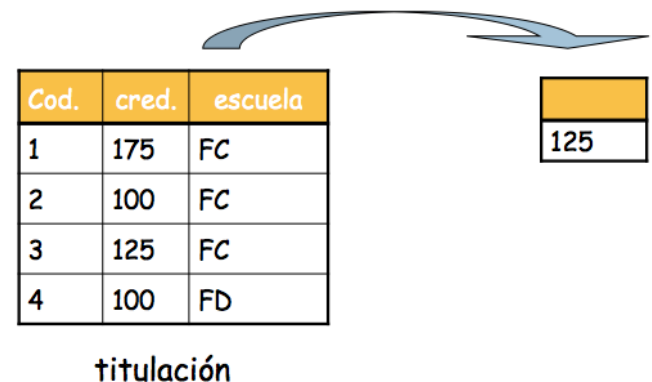

Contenidos

- ∞ 1. Introducción a SQL – INSERT, UPDATE, DELETE y SELECT
- ∞ 2. Selección – WHERE
- ∞ 3. Ordenación de tuplas – ORDER BY, OFFSET y FETCH
- ∞ 4. Operaciones sobre conjuntos - UNION, INTERSECT y MINUS
- ∞ 5. Subconsultas anidadas - IN/NOT IN, SOME/ALL y EXISTS/NOT EXISTS
- ∞ 6. Producto – INNER JOIN ... ON y LEFT/RIGHT JOIN
-  ∞ 7. Funciones de agregación – GROUP BY y HAVING
- ∞ 8. Combinación de consultas e inserción, modificación o borrado de datos
- ∞ 9 . Vistas – CREATE VIEW

Funciones de agregación – GROUP BY y HAVING

- ⌘ Las funciones de agregación toman una colección de datos y calculan una operación
- ⌘ Es posible aplicar un filtro sobre el resultado de la operación
- ⌘ Sintaxis, aplicable a:
 - Una colección de cualquier dominio, sirve para contar tuplas: **SELECT COUNT**(<atributo>)
 - Números, cadenas, fechas y calculan el mínimo o el máximo: **SELECT MIN**(<atributo>), **SELECT MAX**(<atributo>)
 - Números y calculan la suma y el promedio: **SELECT SUM**(<atributo>), **SELECT AVG**(<atributo>)
- ⌘ Ejemplo: obtener el promedio de créditos de las titulaciones


SELECT AVG(créditos) **FROM** titulación



Funciones de agregación – GROUP BY y HAVING

- ∞ **GROUP BY**: agrupa los resultados de la consulta en función de los diferentes valores que puedan tener los campos de agrupación indicados
- ∞ Los campos que se indiquen en esta cláusula son los únicos que pueden indicarse en la cláusula SELECT sin funciones de agregación
- ∞ Sintaxis:
 - **GROUP BY** <Atributo1> [, <Atributo2>, ...]
- ∞ Ejemplo: obtener el número medio de créditos de las titulaciones para cada escuela

SELECT escuela, **AVG**(créditos) **FROM** titulación **GROUP BY** escuela



Cod.	cred.	escuela
1	175	FC
2	100	FC
3	125	FC
4	100	FD

titulación

escuela	
FC	133,4
FD	100

Funciones de agregación – GROUP BY y HAVING

∞ **HAVING**: permite filtrar los resultados de la agrupación indicando una condición a satisfacer

∞ Sintaxis:

- **HAVING** <condición>

∞ Ejemplo 1: obtener el número medio de créditos de las titulaciones para cada escuela siempre y cuando sea dicho promedio superior a 110 créditos

```
SELECT escuela, AVG(créditos) FROM titulación  
GROUP BY escuela HAVING AVG(créditos)>110
```




Cod.	cred.	escuela
1	175	FC
2	100	FC
3	125	FC
4	100	FD

escuela	
FC	133,4

Funciones de agregación – GROUP BY y HAVING

∞ Ejemplo 2: obtener el número medio de créditos de las titulaciones de menos de 150 créditos para cada escuela siempre y cuando sea dicho promedio superior a 110 créditos


```
SELECT escuela, AVG(créditos) FROM titulación
WHERE credits<150
GROUP BY escuela
HAVING AVG(créditos)>110
```



Cod.	cred.	escuela
1	175	FC
2	100	FC
3	125	FC
4	100	FD

escuela	
FC	112,5

Contenidos

- ∞ 1. Introducción a SQL – INSERT, UPDATE, DELETE y SELECT
- ∞ 2. Selección – WHERE
- ∞ 3. Ordenación de tuplas – ORDER BY, OFFSET y FETCH
- ∞ 4. Operaciones sobre conjuntos - UNION, INTERSECT y MINUS
- ∞ 5. Subconsultas anidadas - IN/NOT IN, SOME/ALL y EXISTS/NOT EXISTS
- ∞ 6. Producto – INNER JOIN ... ON y LEFT/RIGHT JOIN
- ∞ 7. Funciones de agregación – GROUP BY y HAVING
-  ∞ 8. Combinación de consultas e inserción, modificación o borrado de datos
- ∞ 9 . Vistas – CREATE VIEW

Combinación de consultas e inserción, modificación o borrado de datos

∞ **Ejemplo 1:** Insertar en prácticas los alumnos que no estén matriculados en ninguna asignatura.

```
INSERT INTO practicas(nmatAlumno, cifEmpresa)
(SELECT nmat, '111' FROM alumno
WHERE nmat NOT IN (SELECT nmatAlumno FROM matricula))
```

∞ **Ejemplo 2:** Incrementar en 3 créditos las asignaturas que no tengan alumnos matriculados

```
INSERT INTO practicas(nmatAlumno, cifEmpresa)
(SELECT nmat, '111' FROM alumno
WHERE nmat NOT IN (SELECT nmatAlumno FROM matricula))
```

∞ **Ejemplo 3:** Borrar las asignaturas que no tengan alumnos matriculados

```
DELETE FROM asignaturas WHERE nombre
NOT IN (SELECT nombreAsig FROM matricula)
```

Contenidos

- ☞ 1. Introducción a SQL – INSERT, UPDATE, DELETE y SELECT
- ☞ 2. Selección – WHERE
- ☞ 3. Ordenación de tuplas – ORDER BY, , OFFSET y FETCH
- ☞ 4. Operaciones sobre conjuntos - UNION, INTERSECT y MINUS
- ☞ 5. Subconsultas anidadas - IN/NOT IN, SOME/ALL y EXISTS/NOT EXISTS
- ☞ 6. Producto – INNER JOIN ... ON y LEFT/RIGHT JOIN
- ☞ 7. Funciones de agregación – GROUP BY y HAVING
- ☞ 8. Combinación de consultas e inserción, modificación o borrado de datos
- ☞ 9 . Vistas – CREATE VIEW



Vistas – CREATE VIEW

∞ **Vistas:** se podrían definir como tablas virtuales que se construyen a partir de una consulta

- Sintaxis:

```
CREATE VIEW <nombre> AS <consulta>
```

∞ A tener en cuenta:

- En las vistas se guarda el código de la consulta (a no ser que se trate de una vista materializada) mientras que en las tablas convencionales se guardan todos los datos insertados en ellas
- Las vistas normalmente no son actualizables, solo de consulta

∞ Ejemplo: Crear una vista con las titulaciones de la EPI

```
CREATE VIEW titulacionesEPI AS SELECT nombre FROM titulación WHERE escuela='EPI'
```