SQL Intermedio

Juan F. Pérez

Departamento MACC
Matemáticas Aplicadas y Ciencias de la Computación
Universidad del Rosario

juanferna.perez@urosario.edu.co

Primer Semestre de 2019

Contenidos

Strings

Strings

- Strings se definen con comillas sencillas (estándar) `'hola'`

Strings

- Strings se definen con comillas sencillas (estándar) `'hola'`
- Si se necesita una comilla en la cadena de caracteres, se escriben dos

Strings

- Strings se definen con comillas sencillas (estándar) `'hola'`
- Si se necesita una comilla en la cadena de caracteres, se escriben dos
- Distingue mayúsculas y minúsculas (estándar) - excepto MySQL y SQL Server

Operaciones sobre Strings

- Concatenación ||

Operaciones sobre Strings

- Concatenación ||
- `'Ho' || 'la'` resulta en `'Hola'`

Operaciones sobre Strings

- Concatenación ||
- `'Ho' || 'la'` resulta en `'Hola'`
- Convertir a minúsculas `lower`

Operaciones sobre Strings

- Concatenación ||
- `'Ho' || 'la'` resulta en `'Hola'`
- Convertir a minúsculas `lower`
- `lower('HolA')` resulta en `'hola'`

Operaciones sobre Strings

- Concatenación ||
- `'Ho' || 'la'` resulta en `'Hola'`
- Convertir a minúsculas `lower`
- `lower('HolA')` resulta en `'hola'`
- Convertir a mayúsculas `upper`

Operaciones sobre Strings

- Concatenación ||
- `'Ho' || 'la'` resulta en `'Hola'`
- Convertir a minúsculas `lower`
- `lower('HolA')` resulta en `'hola'`
- Convertir a mayúsculas `upper`
- `upper('HolA')` resulta en `'HOLA'`

Operaciones sobre Strings

- Reemplazar substring `overlay`

Operaciones sobre Strings

- Reemplazar substring `overlay`
- `overlay('Bogotá' placing 'aca' from 2)` resulta en `'Bacatá'`

Operaciones sobre Strings

- Reemplazar substring `overlay`
- `overlay('Bogotá' placing 'aca' from 2)` resulta en `'Bacatá'`
- `overlay('Bachata' placing 'ogo' from 2 for 4)` resulta en `'Bogota'`

Operaciones sobre Strings

- Reemplazar substring `overlay`
- `overlay('Bogotá' placing 'aca' from 2)` resulta en `'Bacatá'`
- `overlay('Bachata' placing 'ogo' from 2 for 4)` resulta en `'Bogota'`
- Determinar la posición de un substring `position`

Operaciones sobre Strings

- Reemplazar substring `overlay`
- `overlay('Bogotá' placing 'aca' from 2)` resulta en `'Bacatá'`
- `overlay('Bachata' placing 'ogo' from 2 for 4)` resulta en `'Bogota'`
- Determinar la posición de un substring `position`
- `position('dinamarca' in 'Cundinamarca')` resulta en `4`

Operaciones sobre Strings

- Extraer un substring usando posiciones `substring`

Operaciones sobre Strings

- Extraer un substring usando posiciones `substring`
- `substring('Bogotá' from 3 for 2)` resulta en `'go'`

Operaciones sobre Strings

- Extraer un substring usando posiciones `substring`
- `substring('Bogotá' from 3 for 2)` resulta en `'go'`
- Extraer un substring usando un patrón `substring`

Operaciones sobre Strings

- Extraer un substring usando posiciones `substring`
- `substring('Bogotá' from 3 for 2)` resulta en `'go'`
- Extraer un substring usando un patrón `substring`
- `substring(string from patrón for escape)` (patrón define una expresión regular)

Operaciones sobre Strings

- Eliminar los primeros caracteres de un string `trim`

Operaciones sobre Strings

- Eliminar los primeros caracteres de un string `trim`
- `trim(leading 'B' from 'BBBogotá')` resulta en `'ogotá'`

Operaciones sobre Strings

- Eliminar los primeros caracteres de un string `trim`
- `trim(leading 'B' from 'BBBogotá')` resulta en `'ogotá'`
- Eliminar los últimos caracteres de un string `trim`

Operaciones sobre Strings

- Eliminar los primeros caracteres de un string `trim`
- `trim(leading 'B' from 'BBBogotá')` resulta en `'ogotá'`
- Eliminar los últimos caracteres de un string `trim`
- `trim(trailing 'tá' from 'Bogotá')` resulta en `'Bogo'`

Operaciones sobre Strings

- Eliminar los primeros caracteres de un string `trim`
- `trim(leading 'B' from 'BBBogotá')` resulta en `'ogotá'`
- Eliminar los últimos caracteres de un string `trim`
- `trim(trailing 'tá' from 'Bogotá')` resulta en `'Bogo'`
- Eliminar los primeros y los últimos caracteres de un string `trim`

Operaciones sobre Strings

- Eliminar los primeros caracteres de un string `trim`
- `trim(leading 'B' from 'BBBogotá')` resulta en `'ogotá'`
- Eliminar los últimos caracteres de un string `trim`
- `trim(trailing 'tá' from 'Bogotá')` resulta en `'Bogo'`
- Eliminar los primeros y los últimos caracteres de un string `trim`
- `trim(both 'r' from 'rrrelerr')` resulta en `'ele'`

Expresiones regulares y el operador `LIKE`

- Porcentaje % representa cualquier string (incluso vacía)

Expresiones regulares y el operador `LIKE`

- Porcentaje % representa cualquier string (incluso vacía)
- Guión bajo _ representa un caracter

Expresiones regulares y el operador `LIKE`

- Porcentaje % representa cualquier string (incluso vacía)
- Guión bajo _ representa un caracter
- Compara con el operador `LIKE`

Expresiones regulares y el operador `LIKE`

- Porcentaje % representa cualquier string (incluso vacía)
- Guión bajo _ representa un caracter
- Compara con el operador `LIKE`
- `'Hola' LIKE '%la'` verdadero

Expresiones regulares y el operador `LIKE`

- Porcentaje % representa cualquier string (incluso vacía)
- Guión bajo _ representa un caracter
- Compara con el operador `LIKE`
- `'Hola' LIKE '%la'` verdadero
- `'Hola' LIKE '_la'` falso

Expresiones regulares y el operador `LIKE`

- Porcentaje % representa cualquier string (incluso vacía)
- Guión bajo _ representa un caracter
- Compara con el operador `LIKE`
- `'Hola' LIKE '%la'` verdadero
- `'Hola' LIKE '_la'` falso
- `'Hola' LIKE '__la'` verdadero

Expresiones regulares y el operador `LIKE`

- Porcentaje % representa cualquier string (incluso vacía)
- Guión bajo _ representa un caracter
- Compara con el operador `LIKE`
- `'Hola' LIKE '%la'` verdadero
- `'Hola' LIKE '_la'` falso
- `'Hola' LIKE '__la'` verdadero
- `'Camino' LIKE '%mi%'` verdadero

Expresiones regulares y el operador `LIKE`

- Porcentaje % representa cualquier string (incluso vacía)
- Guión bajo _ representa un caracter
- Compara con el operador `LIKE`
- `'Hola' LIKE '%la'` verdadero
- `'Hola' LIKE '_la'` falso
- `'Hola' LIKE '__la'` verdadero
- `'Camino' LIKE '%mi%'` verdadero
- `'Siderúrgica' LIKE '%de___gi%'` verdadero

Expresiones regulares y el operador `LIKE`

estudiantes(Nombre, Apellidos, <u>ID</u>)

Nombre	Apellidos	ID
Juan	Pérez	8888
Ana	Gutiérrez	5555
Ernesto	Huertas	2222

Expresiones regulares y el operador `LIKE`

estudiantes(Nombre, Apellidos, ID)

Nombre	Apellidos	ID
Juan	Pérez	8888
Ana	Gutiérrez	5555
Ernesto	Huertas	2222

```
SELECT Apellidos FROM estudiantes WHERE Apellidos LIKE '%rez
    %;
```

Expresiones regulares y el operador `LIKE`

estudiantes(Nombre, Apellidos, ID)

Nombre	Apellidos	ID
Juan	Pérez	8888
Ana	Gutiérrez	5555
Ernesto	Huertas	2222

```
SELECT Apellidos FROM estudiantes WHERE Apellidos LIKE '%rez
    %;
```

```
SELECT Apellidos FROM estudiantes WHERE Apellidos LIKE '%t%;
```

Expresiones regulares y el operador `LIKE`

- Si la expresión incluye un porcentaje % o un guión bajo _ se usa operador `ESCAPE`

Existe también el operador `SIMILAR TO` que permite escribir expresiones regulares más generales.

Expresiones regulares y el operador `LIKE`

- Si la expresión incluye un porcentaje % o un guión bajo _ se usa operador `ESCAPE`
- `LIKE '%50-100\%' ESCAPE '\'` es equivalente a cualquier string que termine en '50-100 %'

Existe también el operador `SIMILAR TO` que permite escribir expresiones regulares más generales.

Ordenar y agrupar

Ordenar

- La cláusula `ORDER BY` permite ordenar los resultados de acuerdo con el valor de algunos atributos.

Ordenar

- La cláusula `ORDER BY` permite ordenar los resultados de acuerdo con el valor de algunos atributos.
- Orden ascendente por defecto. Operadores `DESC` y `ASC` determinan el tipo de ordenamiento.

Ordenar

- La cláusula `ORDER BY` permite ordenar los resultados de acuerdo con el valor de algunos atributos.
- Orden ascendente por defecto. Operadores `DESC` y `ASC` determinan el tipo de ordenamiento.

```
SELECT Apellidos FROM estudiantes
WHERE Apellidos LIKE '%rez%
ORDER␣BY␣Nombre;
```

Ordenar

```
SELECT Apellidos FROM estudiantes
WHERE Apellidos LIKE '%rez%
ORDER BY Nombre;
```

Ordenar

```
SELECT Apellidos FROM estudiantes
WHERE Apellidos LIKE '%rez%
ORDER BY Nombre;
```

```
SELECT Apellidos FROM estudiantes
WHERE Apellidos LIKE '%rez%
ORDER BY Nombre DESC;
```

Ordenar

```
SELECT Apellidos FROM estudiantes
WHERE Apellidos LIKE '%rez%
ORDER BY Nombre;
```

```
SELECT Apellidos FROM estudiantes
WHERE Apellidos LIKE '%rez%
ORDER BY Nombre DESC;
```

```
SELECT Apellidos FROM estudiantes
WHERE Apellidos LIKE '%rez%
ORDER BY Apellidos DESC, Nombre ASC;
```

Agregados

- Agregado es una función que recibe una colección de valores y retorna un valor

Agregados

- Agregado es una función que recibe una colección de valores y retorna un valor
- Funciones de agregación estándar en SQL:

Agregados

- Agregado es una función que recibe una colección de valores y retorna un valor

- Funciones de agregación estándar en SQL:
  - **avg**: promedio (numérico)

Agregados

- Agregado es una función que recibe una colección de valores y retorna un valor
- Funciones de agregación estándar en SQL:
  - **avg**: promedio (numérico)
  - **min**: mínimo

Agregados

- Agregado es una función que recibe una colección de valores y retorna un valor
- Funciones de agregación estándar en SQL:
  - **avg**: promedio (numérico)
  - **min**: mínimo
  - **max**: máximo

Agregados

- Agregado es una función que recibe una colección de valores y retorna un valor
- Funciones de agregación estándar en SQL:
  - **avg**: promedio (numérico)
  - **min**: mínimo
  - **max**: máximo
  - **sum**: suma/total (numérico)

Agregados

- Agregado es una función que recibe una colección de valores y retorna un valor
- Funciones de agregación estándar en SQL:
  - **avg**: promedio (numérico)
  - **min**: mínimo
  - **max**: máximo
  - **sum**: suma/total (numérico)
  - **count**: cuenta

Agregados

estCursos(estID, cursoID, nota)

estID	cursoID	nota
8888	000	4.2
5555	000	3.6
5555	001	4.8
3333	001	3.3
3333	002	2.5

Agregados

```
SELECT avg(nota)
FROM estCursos
WHERE cursoID = '000';
```

estCursos(<u>estID</u>, <u>cursoID</u>, nota)

estID	cursoID	nota
8888	000	4.2
5555	000	3.6
5555	001	4.8
3333	001	3.3
3333	002	2.5

Agregados - Count

estCursos(estID, cursoID, nota)

estID	cursoID	nota
8888	000	4.2
5555	000	3.6
5555	001	4.8
3333	001	3.3
3333	002	2.5

Agregados - Count

```
SELECT count(estID)
FROM estCursos;
```

estCursos(<u>estID</u>, <u>cursoID</u>, nota)

estID	cursoID	nota
8888	000	4.2
5555	000	3.6
5555	001	4.8
3333	001	3.3
3333	002	2.5

Agregados - Count distinct

estCursos(estID, cursoID, nota)

estID	cursoID	nota
8888	000	4.2
5555	000	3.6
5555	001	4.8
3333	001	3.3
3333	002	2.5

Agregados - Count distinct

```
SELECT count(distinct estID)
FROM estCursos;
```

estCursos(<u>estID</u>, <u>cursoID</u>, nota)

estID	cursoID	nota
8888	000	4.2
5555	000	3.6
5555	001	4.8
3333	001	3.3
3333	002	2.5

Agregados - Count

estCursos(estID, cursoID, nota)

estID	cursoID	nota
8888	000	4.2
5555	000	3.6
5555	001	4.8
3333	001	3.3
3333	002	2.5

Agregados - Count

```
SELECT count(*)
FROM estCursos
WHERE cursoID = '000';
```

estCursos(<u>estID</u>, <u>cursoID</u>, nota)

estID	cursoID	nota
8888	000	4.2
5555	000	3.6
5555	001	4.8
3333	001	3.3
3333	002	2.5

Agregados - Count

estCursos(estID, cursoID, nota)

estID	cursoID	nota
8888	000	4.2
5555	000	3.6
5555	001	4.8
3333	001	3.3
3333	002	2.5

Agregados - Count

```
SELECT count(1)
FROM estCursos
WHERE cursoID = '000';
```

estCursos(estID, cursoID, nota)

estID	cursoID	nota
8888	000	4.2
5555	000	3.6
5555	001	4.8
3333	001	3.3
3333	002	2.5

Agregados Agrupando

estCursos(estID, cursoID, nota)

estID	cursoID	nota
8888	000	4.2
5555	000	3.6
5555	001	4.8
3333	001	3.3
3333	002	2.5

Agregados Agrupando

```
SELECT cursoID, avg(nota) as nota_prom
FROM estCursos
GROUP BY cursoID;
```

estCursos(estID, cursoID, nota)

estID	cursoID	nota
8888	000	4.2
5555	000	3.6
5555	001	4.8
3333	001	3.3
3333	002	2.5

Agregados Agrupando

estCursos(estID, cursoID, nota)

estID	cursoID	nota
8888	000	4.2
5555	000	3.6
5555	001	4.8
3333	001	3.3
3333	002	2.5

Agregados Agrupando

```
SELECT estID, avg(nota) as nota_prom
FROM estCursos
GROUP BY estID;
```

estCursos(<u>estID</u>, <u>cursoID</u>, nota)

estID	cursoID	nota
8888	000	4.2
5555	000	3.6
5555	001	4.8
3333	001	3.3
3333	002	2.5

Agregados Agrupando

```
SELECT estID, avg(nota) as nota_prom
FROM estCursos
GROUP BY estID;
```

Agregados Agrupando

```
SELECT estID, avg(nota) as nota_prom
FROM estCursos
GROUP BY estID;
```

- En la claúsula `SELECT` pueden aparecer:
  - Atributos agregados

Agregados Agrupando

```
SELECT estID, avg(nota) as nota_prom
FROM estCursos
GROUP BY estID;
```

- En la claúsula `SELECT` pueden aparecer:
  - Atributos agregados
  - Atributos agrupados con `GROUP BY`

Agregados: `Having`

estCursos(estID, cursoID, nota)

estID	cursoID	nota
8888	000	4.2
5555	000	3.6
5555	001	4.8
3333	001	3.3
3333	002	2.5

Agregados: Having

```
SELECT estID, avg(nota) as nota_prom
FROM estCursos
GROUP BY estID
HAVING avg(nota) > 4.0;
```

estCursos(estID, cursoID, nota)

estID	cursoID	nota
8888	000	4.2
5555	000	3.6
5555	001	4.8
3333	001	3.3
3333	002	2.5

Agregados: Having

```
SELECT estID, avg(nota) as nota_prom
FROM estCursos
GROUP BY estID
HAVING avg(nota) > 4.0;
```

Agregados: Having

```
SELECT estID, avg(nota) as nota_prom
FROM estCursos
GROUP BY estID
HAVING avg(nota) > 4.0;
```

Orden de operaciones (comprensión, no ejecución):

1. FROM: obtener una o más relaciones
2. WHERE: filtrar los registros de las relaciones

Agregados: Having

```
SELECT estID, avg(nota) as nota_prom
FROM estCursos
GROUP BY estID
HAVING avg(nota) > 4.0;
```

Orden de operaciones (comprensión, no ejecución):

1. FROM: obtener una o más relaciones
2. WHERE: filtrar los registros de las relaciones
3. GROUP BY: asignar las tuplas/registros a grupos

Agregados: Having

```
SELECT estID, avg(nota) as nota_prom
FROM estCursos
GROUP BY estID
HAVING avg(nota) > 4.0;
```

Orden de operaciones (comprensión, no ejecución):

1. FROM: obtener una o más relaciones
2. WHERE: filtrar los registros de las relaciones
3. GROUP BY: asignar las tuplas/registros a grupos
4. HAVING: filtrar grupos, eliminando los que no cumplan con esta condición

Agregados: Having

```
SELECT estID, avg(nota) as nota_prom
FROM estCursos
GROUP BY estID
HAVING avg(nota) > 4.0;
```

Orden de operaciones (comprensión, no ejecución):

1. FROM: obtener una o más relaciones
2. WHERE: filtrar los registros de las relaciones
3. GROUP BY: asignar las tuplas/registros a grupos
4. HAVING: filtrar grupos, eliminando los que no cumplan con esta condición
5. SELECT: aplicada funciones de agregación para calcular el valor de los atributos asociados a los grupos

Agregados: `Having`

estCursos(estID, cursoID, nota)

estID	cursoID	nota
8888	000	4.2
5555	000	3.6
5555	001	4.8
3333	001	3.3
3333	002	2.5

Agregados: Having

```
SELECT estID, avg(nota) as nota_prom
FROM estCursos
WHERE cursoID < '100'
GROUP BY estID
HAVING avg(nota) > 4.0;
```

estCursos(estID, cursoID, nota)

estID	cursoID	nota
8888	000	4.2
5555	000	3.6
5555	001	4.8
3333	001	3.3
3333	002	2.5

Valores nulos NULL

Valores nulos NULL

- Resultado de ooperaciones artiméticas con NULL resulta en NULL

Valores nulos NULL

- Resultado de ooperaciones artiméticas con NULL resulta en NULL
- 1 <NULL resulta en UNKNOWN

Valores nulos NULL

- Resultado de ooperaciones artiméticas con NULL resulta en NULL
- 1 <NULL resulta en UNKNOWN
- Valores lógicos: TRUE, FALSE y UNKNOWN

Valores nulos NULL

- Resultado de ooperaciones artiméticas con NULL resulta en NULL
- 1 <NULL resulta en UNKNOWN
- Valores lógicos: TRUE, FALSE y UNKNOWN
- TRUE AND UNKNOWN resulta en UNKNOWN

Valores nulos NULL

- Resultado de ooperaciones artiméticas con NULL resulta en NULL
- 1 <NULL resulta en UNKNOWN
- Valores lógicos: TRUE, FALSE y UNKNOWN
- TRUE AND UNKNOWN resulta en UNKNOWN
- FALSE AND UNKNOWN resulta en FALSE

Valores nulos NULL

- Resultado de ooperaciones artiméticas con NULL resulta en NULL
- 1 <NULL resulta en UNKNOWN
- Valores lógicos: TRUE, FALSE y UNKNOWN
- TRUE AND UNKNOWN resulta en UNKNOWN
- FALSE AND UNKNOWN resulta en FALSE
- UNKNOWN AND UNKNOWN resulta en UNKNOWN

Valores nulos NULL

- TRUE OR UNKNOWN resulta en TRUE

Valores nulos NULL

- `TRUE OR UNKNOWN` resulta en `TRUE`
- `FALSE OR UNKNOWN` resulta en `UNKNOWN`

Valores nulos NULL

- `TRUE OR UNKNOWN` resulta en `TRUE`
- `FALSE OR UNKNOWN` resulta en `UNKNOWN`
- `UNKNOWN OR UNKNOWN` resulta en `UNKNOWN`

Valores nulos NULL

- `TRUE OR UNKNOWN` resulta en `TRUE`
- `FALSE OR UNKNOWN` resulta en `UNKNOWN`
- `UNKNOWN OR UNKNOWN` resulta en `UNKNOWN`
- `NOT UNKNOWN` resulta en `UNKNOWN`

Valores nulos NULL en filtros

- Si la claúsula WHERE resulta en FALSE o UNKNOWN, la tupla se descarta

Valores nulos NULL en filtros

- Si la claúsula WHERE resulta en FALSE o UNKNOWN, la tupla se descarta
- IS NULL e IS NOT NULL se pueden usar para evaluar si los registros son nulos o no

Valores nulos NULL en filtros

- Si la claúsula WHERE resulta en FALSE o UNKNOWN, la tupla se descarta
- IS NULL e IS NOT NULL se pueden usar para evaluar si los registros son nulos o no

```
SELECT count(1)
FROM estCursos
WHERE cursoID IS NOT NULL;
```

Valores nulos NULL en agregados

- La función COUNT cuenta todos los registros, incluso si son nulos

Valores nulos NULL en agregados

- La función COUNT cuenta todos los registros, incluso si son nulos
- Las demás funciónes de agregación ignoran los registros nulos

Valores nulos NULL en agregados

- La función COUNT cuenta todos los registros, incluso si son nulos
- Las demás funciónes de agregación ignoran los registros nulos

```
SELECT avg(nota)
FROM estCursos
WHERE cursoID = '000';
```

Subconsultas embebidas

Subconsultas embebidas

- Expresiones select-from-where dentro de otra consulta

Subconsultas embebidas

- Expresiones select-from-where dentro de otra consulta
- Pueden ir en la cláusula `WHERE`

Subconsultas embebidas

- Expresiones select-from-where dentro de otra consulta
- Pueden ir en la cláusula `WHERE`
  - Membresía de conjunto

Subconsultas embebidas

- Expresiones select-from-where dentro de otra consulta
- Pueden ir en la cláusula `WHERE`
  - Membresía de conjunto
  - Comparación de conjuntos

Subconsultas embebidas

- Expresiones select-from-where dentro de otra consulta
- Pueden ir en la cláusula `WHERE`
  - Membresía de conjunto
  - Comparación de conjuntos
- Pueden ir en la cláusula `FROM`

Subconsultas embebidas `WHERE` - Membresía de conjunto `IN`

estCursos(estID, cursoID, nota)

estID	cursoID	nota	año	periodo
8888	000	4.2	2018	1
5555	000	3.6	2018	2
5555	001	4.8	2017	1
3333	001	3.3	2018	1
3333	002	2.5	2018	1

Encontrar todos los cursos en los que los estudiantes '3333' y '5555' sacaron menos 4.
Primero los cursos en los que el estudiante '3333' sacó menos de 4:

Subconsultas embebidas WHERE - Membresía de conjunto IN

estCursos(estID, cursoID, nota)

estID	cursoID	nota	año	periodo
8888	000	4.2	2018	1
5555	000	3.6	2018	2
5555	001	4.8	2017	1
3333	001	3.3	2018	1
3333	002	2.5	2018	1

Encontrar todos los cursos en los que los estudiantes '3333' y '5555' sacaron menos 4.

Primero los cursos en los que el estudiante '3333' sacó menos de 4:

```
(SELECT cursoID
FROM estCursos
WHERE estID = '3333' and nota < 4.0;
```

Subconsultas embebidas WHERE - Membresía de conjunto IN

Encontrar todos los cursos en los que los estudiantes '3333' y '5555' sacaron menos 4.
Primero los cursos en los que el estudiante '3333' sacó menos de 4:

```
SELECT cursoID
FROM estCursos
WHERE estID = '3333' AND nota < 4.0;
```

Ahora los cursos en los que el estudiante '5555' sacó menos de 4:

Subconsultas embebidas WHERE - Membresía de conjunto IN

Encontrar todos los cursos en los que los estudiantes '3333' y '5555' sacaron menos 4.
Primero los cursos en los que el estudiante '3333' sacó menos de 4:

```
SELECT cursoID
FROM estCursos
WHERE estID = '3333' AND nota < 4.0;
```

Ahora los cursos en los que el estudiante '5555' sacó menos de 4:

```
SELECT cursoID
FROM estCursos
WHERE estID = '5555' AND nota < 4.0;
```

Subconsultas embebidas `WHERE` - Membresía de conjunto `IN`

Encontrar todos los cursos en los que los estudiantes '3333' y '5555' sacaron menos 4.
Incluimos la primera como una subconsulta en la segunda con el operador `IN`:

Subconsultas embebidas WHERE - Membresía de conjunto IN

Encontrar todos los cursos en los que los estudiantes '3333' y '5555' sacaron menos 4.
Incluimos la primera como una subconsulta en la segunda con el operador IN:

```
SELECT cursoID
FROM estCursos
WHERE estID = '5555' AND nota < 4.0 AND
cursoID IN (
SELECT cursoID
FROM estCursos
WHERE estID = '3333' and nota < 4.0
);
```

Subconsultas embebidas `WHERE` - Membresía de conjunto

`NOT IN`

Encontrar todos los cursos en los que el estudiante '3333' sacó menos de 4 pero el estudiante '5555' no sacó menos de 4.

Subconsultas embebidas `WHERE` - Membresía de conjunto

`NOT IN`

Encontrar todos los cursos en los que el estudiante '3333' sacó menos de 4 pero el estudiante '5555' no sacó menos de 4.

```
SELECT cursoID
FROM estCursos
WHERE estID = '5555' AND nota < 4.0 AND
cursoID NOT IN (
SELECT cursoID
FROM estCursos
WHERE estID = '3333' and nota < 4.0
);
```

Subconsultas embebidas `WHERE` - Membresía de conjunto

`IN` y `NOT IN` pueden ser usados en conjuntos enumerados:

Subconsultas embebidas WHERE - Membresía de conjunto

IN y NOT IN pueden ser usados en conjuntos enumerados:

```
SELECT cursoID
FROM estCursos
WHERE estID in ('3333','5555');
```

Subconsultas embebidas `WHERE` - Membresía de conjunto

`NOT IN`

Encontrar todos los cursos en los que el estudiante '3333' sacó menos de 4 pero el estudiante '5555' no sacó menos de 4.

Subconsultas embebidas WHERE - Membresía de conjunto

NOT IN

Encontrar todos los cursos en los que el estudiante '3333' sacó menos de 4 pero el estudiante '5555' no sacó menos de 4.

```
SELECT cursoID
FROM estCursos
WHERE estID = '5555' AND nota < 4.0 AND
cursoID NOT IN (
SELECT cursoID
FROM estCursos
WHERE estID = '3333' and nota < 4.0
);
```

Subconsultas embebidas `WHERE` - Comparación de conjuntos

Determinar los estudiantes que tienen mayores notas que al menos uno de los estudiantes que tomó el curso '000'

estCursos(<u>estID</u>, <u>cursoID</u>, nota)

estID	cursoID	nota	año	periodo
8888	000	4.2	2018	1
5555	000	3.6	2018	2
5555	001	4.8	2017	1
3333	001	3.3	2018	1
3333	002	2.5	2018	1

Subconsultas embebidas `WHERE` - Comparación de conjuntos

Determinar los estudiantes que tienen mayores notas que al menos uno de los estudiantes que tomó el curso '000'

estCursos(<u>estID</u>, <u>cursoID</u>, nota)

estID	cursoID	nota	año	periodo
8888	000	4.2	2018	1
5555	000	3.6	2018	2
5555	001	4.8	2017	1
3333	001	3.3	2018	1
3333	002	2.5	2018	1

```
SELECT A.estID
FROM estCursos as A, estCursos as B
WHERE A.nota > B.nota AND A.cursoID = '000';
```

Subconsultas embebidas `WHERE` - Comparación de conjuntos

Determinar los estudiantes que tienen mayores notas que al menos uno de los estudiantes que tomó el curso '000'

estCursos(<u>estID</u>, <u>cursoID</u>, nota)

estID	cursoID	nota	año	periodo
8888	000	4.2	2018	1
5555	000	3.6	2018	2
5555	001	4.8	2017	1
3333	001	3.3	2018	1
3333	002	2.5	2018	1

Subconsultas embebidas `WHERE` - Comparación de conjuntos

Determinar los estudiantes que tienen mayores notas que al menos uno de los estudiantes que tomó el curso '000'

estCursos(estID, cursoID, nota)

estID	cursoID	nota	año	periodo
8888	000	4.2	2018	1
5555	000	3.6	2018	2
5555	001	4.8	2017	1
3333	001	3.3	2018	1
3333	002	2.5	2018	1

```
SELECT estID
FROM estCursos
WHERE nota > SOME (
SELECT estID FROM estCursos WHERE cursoID = '000');
```

Joins

Natural join

Producto cartesiano de las dos relaciones

Natural join

Producto cartesiano de las dos relaciones

```
select *
from estudiante natural join estCursos;
```

Natural join - on

Unión por solo algunos atributos

Natural join - on

Unión por solo algunos atributos

```
select *
from estudiante join estCursos
on estudiante.id = estCursos.estID;
```

Natural join - on

Unión por solo algunos atributos

```
select *
from estudiante join estCursos
on estudiante.id = estCursos.estID;
```

```
select *
from estudiante, estCursos
where estudiante.id = estCursos.estID;
```

Natural join - on

Unión por solo algunos atributos

Natural join - on

Unión por solo algunos atributos

```
select estudiante.id as id, nombres, apellidos, nombre,
    unid_acad, creditos
from estudiante join estCursos
on estudiante.id = estCursos.estID;
```

Left outer join

- Mantiene todas las tuplas en la relación de la izquierda
- Agrega campos de la relación de la derecha
- Si los campos no existen los deja en `NULL`

Left outer join

- Mantiene todas las tuplas en la relación de la izquierda
- Agrega campos de la relación de la derecha
- Si los campos no existen los deja en `NULL`

```
select *
from estudiante left outer join estCursos
on estudiante.id = estCursos.estID;
```

Right outer join

- Mantiene todas las tuplas en la relación de la derecha
- Agrega campos de la relación de la izquierda
- Si los campos no existen los deja en `NULL`

Right outer join

- Mantiene todas las tuplas en la relación de la derecha
- Agrega campos de la relación de la izquierda
- Si los campos no existen los deja en `NULL`

```
select *
from estudiante right outer join estCursos
on estudiante.id = estCursos.estID;
```

Right outer join

- Mantiene todas las tuplas en la relación de la derecha
- Agrega campos de la relación de la izquierda
- Si los campos no existen los deja en `NULL`

```
select *
from estudiante right outer join estCursos
on estudiante.id = estCursos.estID;
```

```
select *
from estCursos right outer join estudiante
on estudiante.id = estCursos.estID;
```

Full outer join

- Mantiene todas las tuplas en las dos relaciones
- Si los campos no existen los deja en `NULL`

Full outer join

- Mantiene todas las tuplas en las dos relaciones
- Si los campos no existen los deja en `NULL`

```
select *
from estCursos full outer join estudiante
on estudiante.id = estCursos.estID;
```

Full outer join

```
from (
        select *
        from estudiante
        where nombres='Juan'
        ) as A full outer join (
        select *
        from estCursos
        where nota >= 3
        ) as B on A.id = B.estID;
```

Inner join (default)

```
select *
from (
        select *
        from estudiante
        where nombres='Juan'
        ) as A inner join (
        select *
        from estCursos
        where nota >= 3
        ) as B on A.id = B.estID;
```