METRICA

CONSULTING

SQL

2 Introducción

El lenguaje SQL 01

- Declarativo
- Estándar
- Basado en sentencias
- Interpretado
- Sencillo
- Natural

02 Tipos de instrucciones SQL

DQL

- Consulta

DML

- Manipulación

TPL

- Procesado de transacciones

DDL

- Definición datos

CCL

- Control y seguridad

Instrucciones de consulta DQL

SELECT

Instrucciones de manipulación DML

- INSERT
- UPDATE
- DELETE
- SELECT

Instrucciones de procesado de transacciones TPL

- COMMIT
- ROLLBACK

Instrucciones de definición de datos DDL

- CREATE
- DROP
- ALTER

Instrucciones de control CCL

GRANT

REVOKE

Tipos de datos

03

Tipos texto

CHAR(size)	• Longitud fija • 1 a 255 caracteres
VARCHAR(size)	• variable • 2000 caracteres
TEXT	• cadenas de hasta 2 GB

Tipos numéricos

INT(max_dig)	• enteros • entre -2^{31} y 2^{31} • se puede especificar número máximo de dígitos
FLOAT(max_dig, max_dec)	• decimal pequeño • parámetros: número máximo de dígitos y de decimales
DOUBLE(max_dig, max_dec)	• decimal grande • parámetros: número máximo de dígitos y de decimales
DECIMAL	• guardado como cadena

Tipos fecha

DATE

- Formato: AAAA-MM-DD

TIME

- Formato: HH: MI: SS

DATETIME

- Formato: AAAA-MM-DD HH: MI: SS

TIMESTAMP

- marca de tiempo
- segundos desde el tiempo UNIX (1970)

Tipos binarios

RAW

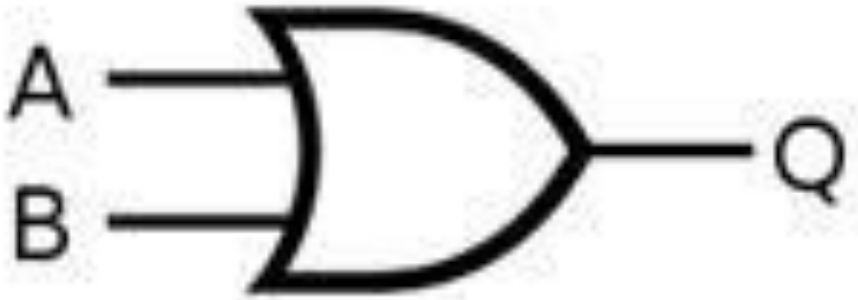
- cadenas binarias

LONGRAW

- igual, pero con mayor capacidad

Operadores

04



Operadores lógicos

=	!=	>
<	>=	<=

Operadores lógicos

AND	OR	NOT	ALL	ANY
BETWEEN	EXISTS	IN	LIKE	SOME

DLL

05

CREATE

DROP

ALTER

06 Creación de tablas

NOT NULL
PRIMARY KEY
UNIQUE
FOREING KEY
CHECK
DEFAULT

```
CREATE TABLE <nombre_tabla>
   ( <definición_columna>
   [, <definición_columna>...]
   [, <restricciones_tabla>]
   );
```

```
<nombre_columna> {<tipo_datos>|<dominio>} [<def_defecto>] [<restricciones_columna>]
```

```
CREATE TABLE personas (
            id INT(7) NOT NULL,
            nombre VARCHAR(15),
            CONSTRAINT PK PRIMARY KEY (id)
);
```

```
CREATE TABLE personas (
            id INT(7) NOT NULL,
            nombre VARCHAR(15),
            PRIMARY KEY (id)
);
```

Restricción FOREIGN KEY

```
FOREIGN KEY <clave_foránea> REFERENCES <nombre_tabla> [(<clave_primaria>)]
[ON DELETE {NO ACTION | CASCADE | SET DEFAULT | SET NULL}]
[ON UPDATE {NO ACTION | CASCADE | SET DEFAULT | SET NULL}]
```

```
CREATE TABLE personas (
        id INT(7) NOT NULL,
        id_dep INT(3),
        nombre VARCHAR(15),
        CONSTRAINT PK PRIMARY KEY (id),
        CONSTRAINT FK FOREIGN KEY (id_dep) REFERENCES departamentos(id)
);
```

07 Aserciones

```
CREATE ASSERTION <nombre_aserción> CHECK (<condiciones>);
```

```
CREATE ASSERTION restriccion1 CHECK (
        NOT EXISTS (SELECT *
        FROM departamentos d, personas p
        WHERE d.id = p.id_dep
        AND p.salario > d.max_salario)
);
```

08 Modificación de tablas

```
{ADD [COLUMN] <nombre_columna> <def_columna> |
  ALTER [COLUMN] <nombre_columna> {SET <def_defecto>|DROP DEFAULT}|
  DROP [COLUMN] <nombre_columna> {RESTRICT|CASCADE}}
```

```
ALTER TABLE <nombre_tabla> {<acción_modificar_columna>|
                            <acción_modificar_restricción_tabla>};
```

```
{ADD <def_restricción>|
  DROP CONSTRAINT <nombre_restricción> {RESTRICT|CASCADE}}
```

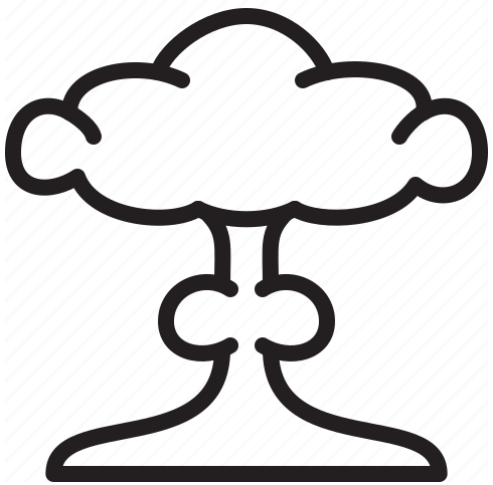
ALTER TABLE personas
ADD profesion VARCHAR(20);

ALTER TABLE personas
ADD FOREIGN KEY (id_dep) **REFERENCES** departamentos(id);

09 Borrar una tabla

```
DROP TABLE <nombre_tabla> {RESTRICT|CASCADE};
```

DROP TABLE personas **CASCADE**;



10 Creación y eliminación de vistas

```
CREATE VIEW <nombre_vista> [(lista_columnas)] AS (consulta)
            [WITH CHECK OPTION];
```

```
DROP VIEW <nombre_vista> (RESTRICT|CASCADE);
```

```
CREATE VIEW personas_dep (departamento, total) AS
            (SELECT d.nombre, COUNT(*)
            FROM departamentos d, personas p
            WHERE d.id = p.id_dep
            GROUP BY d.id
);
```

DML

SELECT

UPDATE

DELETE

INSERT

12 Inserción de filas

```
INSERT INTO <nombre_tabla> [(<columnas>)]
  {VALUES ({<v1>|DEFAULT|NULL}, ..., {<vn>|DEFAULT|NULL}) |<consulta>};
```

INSERT INTO personas (id, nombre)
VALUES (001, “Vanesa”);

INSERT INTO personas **VALUES** (001, “Vanesa”);

13 Borrado de filas

```
DELETE FROM <nombre_tabla>
[WHERE <condiciones>];
```

DELETE FROM personas
WHERE nombre = “Vanesa”;

14 Modificación de filas

```
UPDATE <nombre_tabla>
SET <nombre_columna> = {<expresión>|DEFAULT|NULL}
    [, <nombre_columna> = {<expresión>|DEFAULT|NULL} ...]
WHERE <condiciones>;
```

UPDATE personas
SET nombre = “María Vanesa”
WHERE id = 1;

15 Consulta de filas

```
SELECT <nombre_columnas_seleccionar>
FROM <tabla_consultar>
WHERE <condiciones>;
```

[DISTINCT]

```
SELECT nombre, id AS identificador
        FROM personas
        WHERE   profesion = 'sus labores';
```

16

Funciones agregación

COUNT Número total de filas seleccionadas

SUM Suma de los valores de una columna

MIN Valor mínimo de una columna

MAX Valor máximo de una columna

AVG Media aritmética de una columna

17 Funciones de ordenación

ORDER BY

- Una columna
- Puede ser DESC o ASC

GROUP BY

- Agrupa filas en función del valor de una columna

HAVING

- Condición para aparición
- Modifica a los dos anteriores

```
SELECT COUNT(*) AS desempleados
        FROM personas
        WHERE profesion = “sus labores” OR profesión = NULL;
```

```
SELECT COUNT(*) AS TotalFilas, COUNT(region_envio) AS FilasNoNulas,
        MIN(fecha_envio) AS FechaMin, MAX(fecha_envio) AS FechaMax,
        SUM(peso) AS PesoTotal, AVG(peso) PesoPromedio
        FROM pedidos;
```

```
SELECT COUNT(*) AS TotalFilas, COUNT(region_envio) AS FilasNoNulas,
        MIN(fecha_envio) AS FechaMin, MAX(fecha_envio) AS FechaMax,
        SUM(peso) AS PesoTotal, AVG(peso) PesoPromedio
        FROM pedidos
        GROUP BY id_empleado;
```

```
SELECT nombre
        FROM personas
        GROUP BY profesion;
```

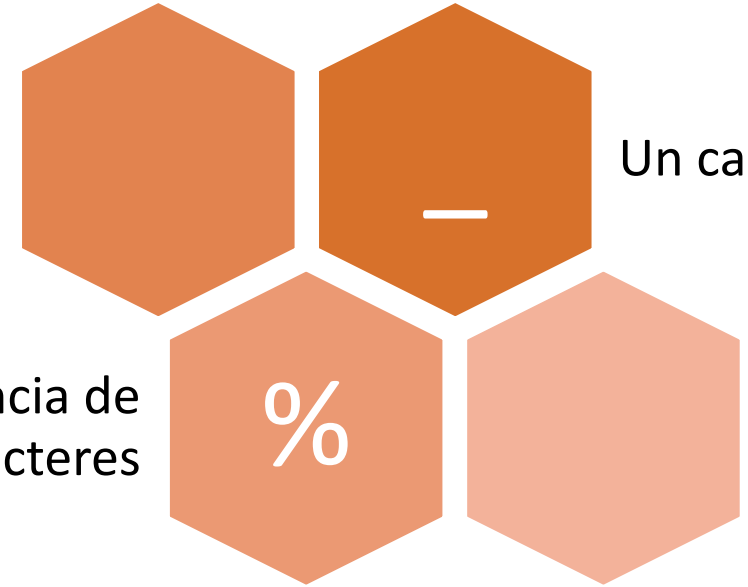
```
SELECT nombre
        FROM personas
        ORDER BY nombre;
```

```
SELECT COUNT (DISTINCT nombre), profesion
        FROM personas
        GROUP BY profesion
        HAVING edad > 40;
```

18 Subconsultas

```
SELECT nombre
        FROM personas
        WHERE edad =  (SELECT MAX(edad)
                       FROM personas);
```

19 Uso de patrones con LIKE

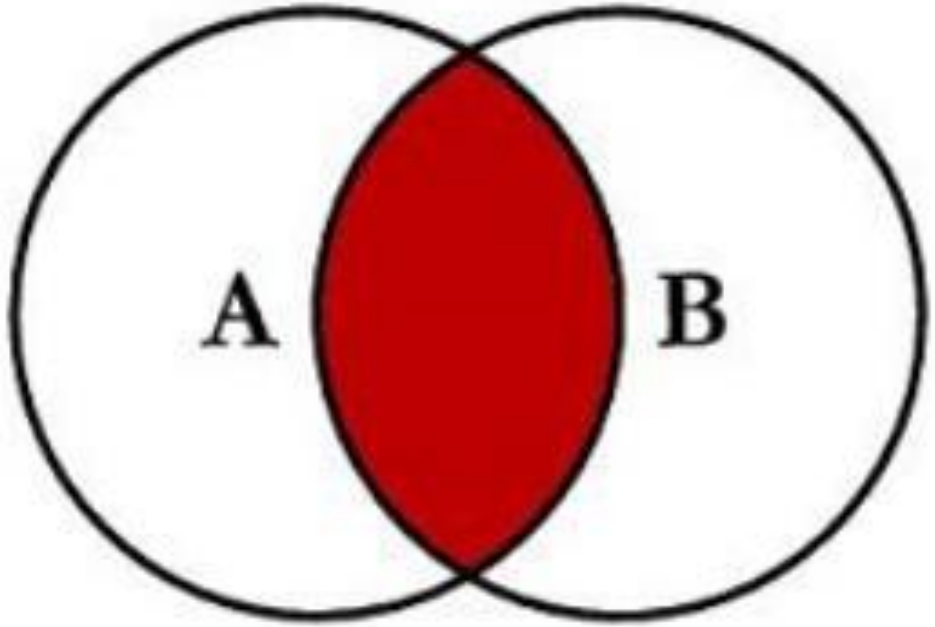


Un carácter

Secuencia de caracteres

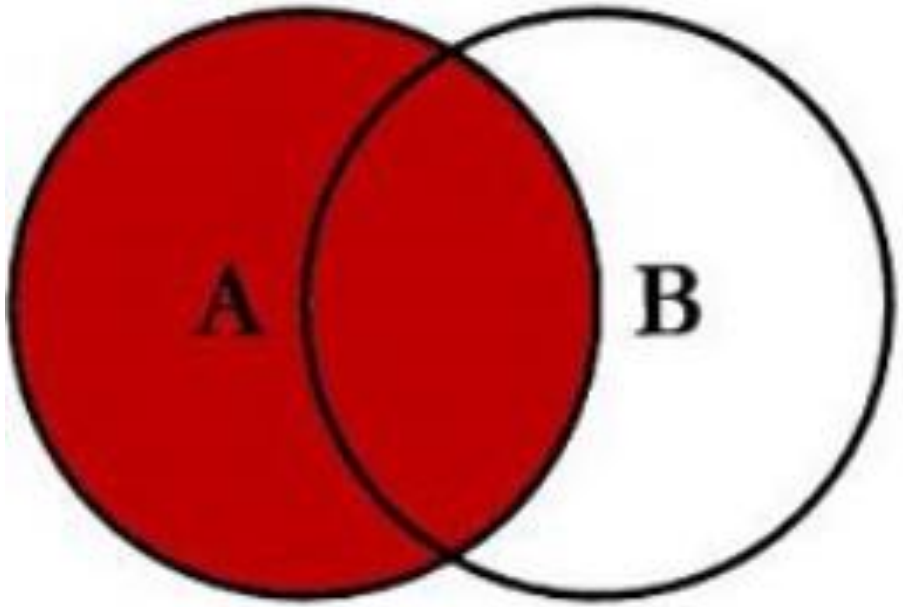
```
SELECT nombre, id
        FROM personas
        WHERE nombre LIKE “V%a”;
```

A ∩ B



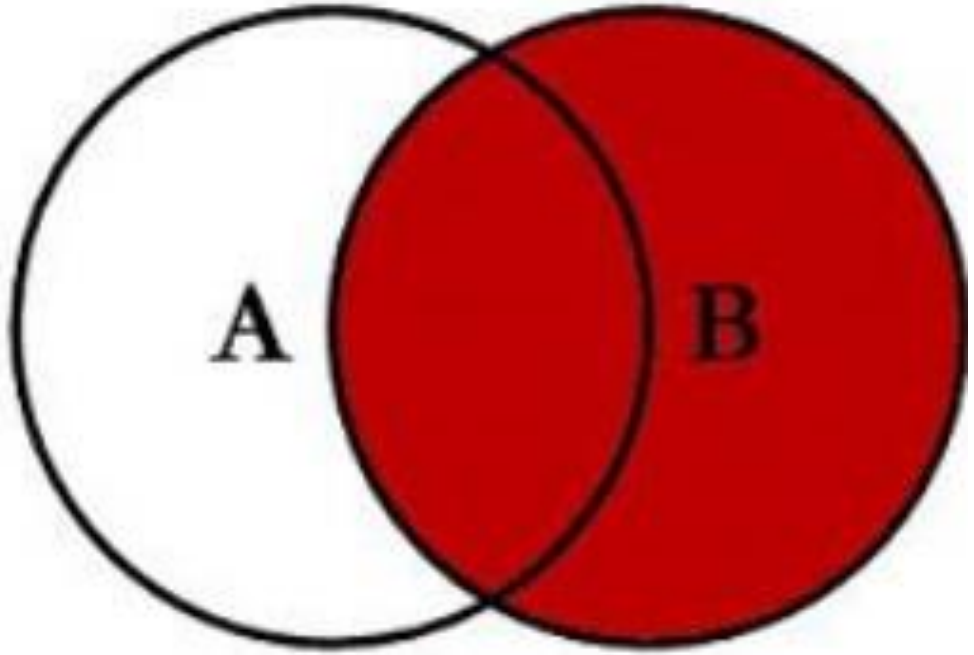

```
SELECT <columnas>
FROM TablaA A
INNER JOIN TablaB B
        ON A.id = B.id;
```

(A ∩ B) ∪ A



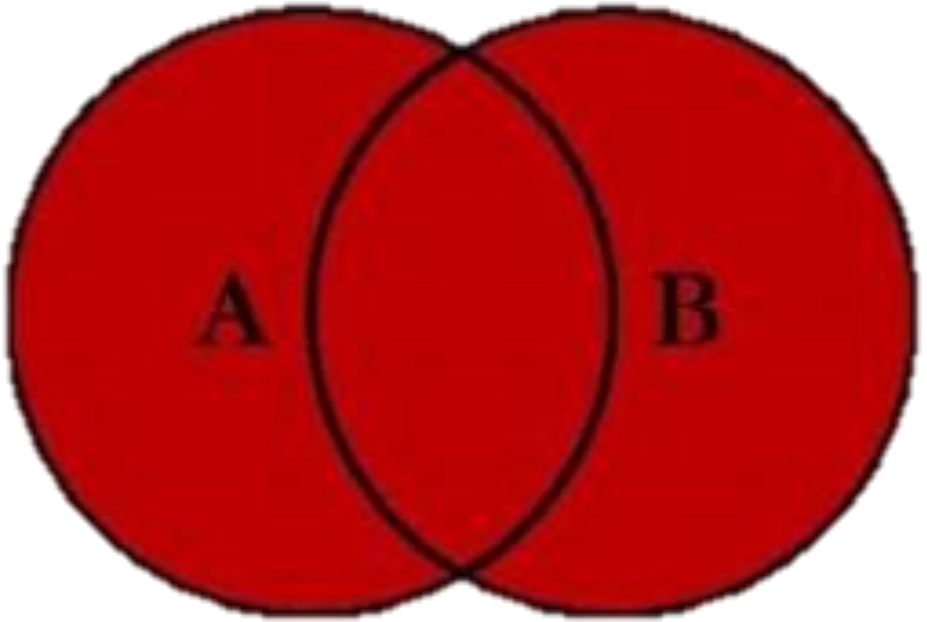

```
SELECT <columnas>
FROM TablaA A
LEFT JOIN TablaB B
        ON A.id = B.id;
```

(A ∩ B) ∪ B



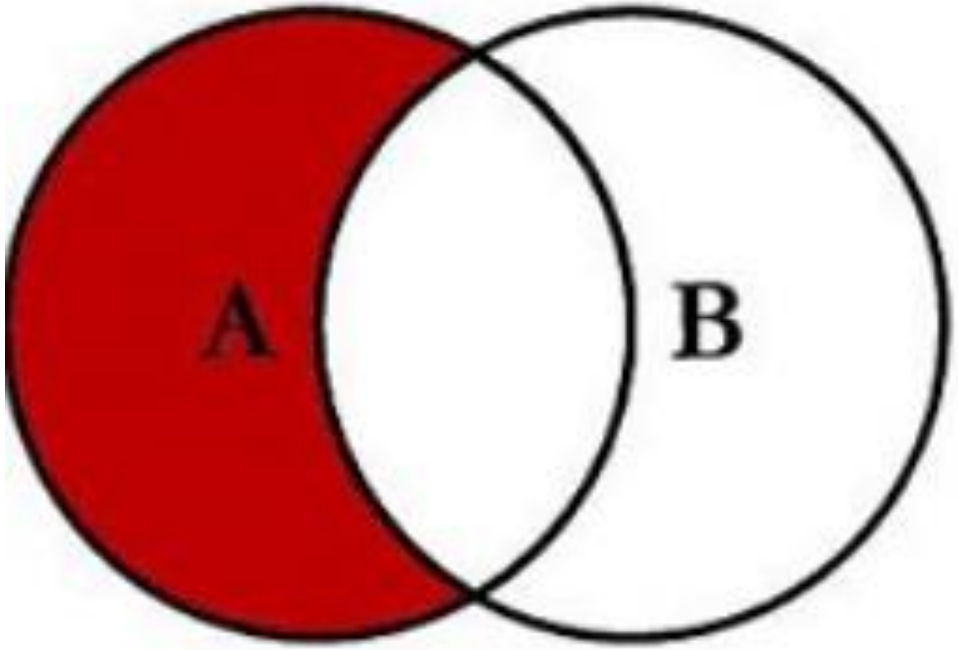

```
SELECT <columnas>
FROM TablaA A
RIGHT JOIN TablaB B
        ON A.id = B.id;
```

(A ∩ B) ∪ (A-B) ∪ (B-A)



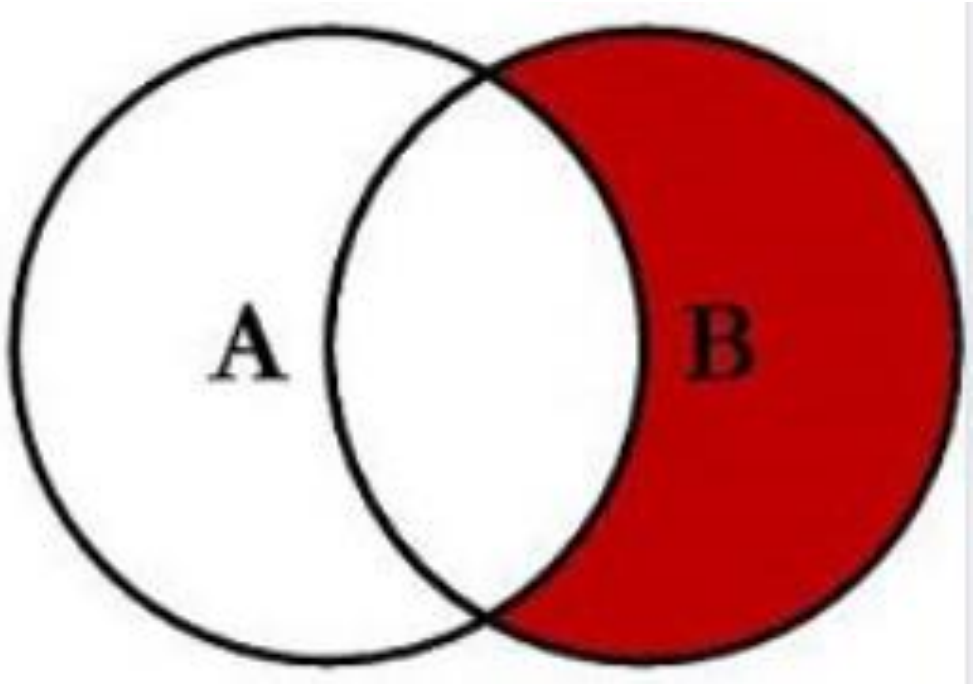

```
SELECT <columnas>
FROM TablaA A
FULL OUTER JOIN TablaB B
        ON A.id = B.id;
```

A - (A ∩ B)



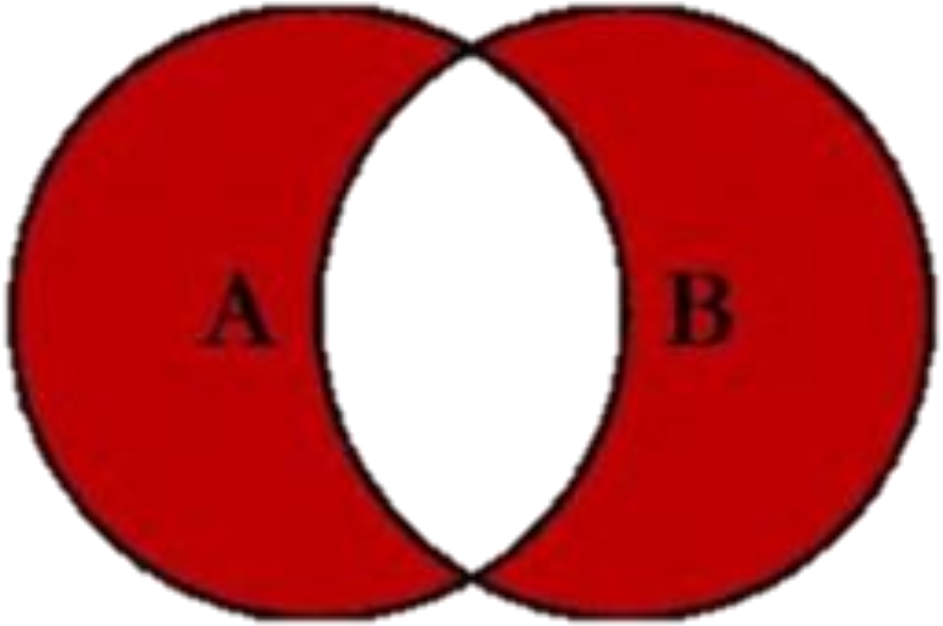

```
SELECT <columnas>
FROM TablaA A
LEFT JOIN TablaB B
        ON A.id = B.id
        WHERE B.id IS NULL
```

B - (A ∩ B)



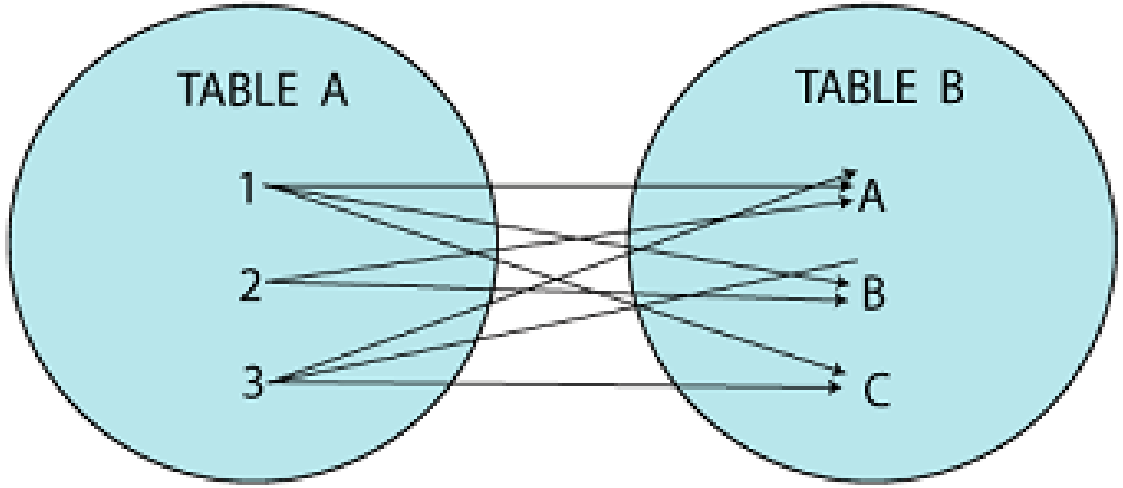

```
SELECT <columnas>
FROM TablaA A
RIGHT JOIN TablaB B
        ON A.id = B.id
        WHERE A.id IS NULL
```

A + B - 2(A ∩ B)




```
SELECT <columnas>
FROM TablaA A
FULL OUTER JOIN TablaB B
        ON A.id = B.id
        WHERE A.id IS NULL
        OR B.id IS NULL
```

A x B




```
SELECT <columnas>
FROM TablaA A
CROSS JOIN TablaB B
```

RICA TEIII CONSULTING

05

GROUP BY - SQL Basics: Simple GROUP BY
https://www.codewars.com/kata/58111f4ee10b5301a7000175

EXIST - SQL Basics: Simple EXISTS
https://www.codewars.com/kata/58113a64e10b53ec36000293/

Identifying Codewars' Bad Kata Authors
https://www.codewars.com/kata/650d7aa9fc2cd80018e3c210

Count IP Addresses
https://www.codewars.com/kata/526989a41034285187000de4/sql

First Normal Form
https://www.codewars.com/kata/62b0da0e58e471000f28ce99

Successful Film Stars Analysis
https://www.codewars.com/kata/649a8ed2c6ba0600314b258d