



Programación y Laboratorio II

Clase 16

Introducción a bases de datos y SQL

Prof. Mauricio Cerizza

Bases de datos

- ¿Qué es una base de datos?
- Tablas
- Tipos de datos
- ¿Qué es una base de datos relacional?
- Modelo entidad-relación
- Claves primarias y foráneas
- ¿Qué es la cardinalidad?
- ¿Qué es un DBMS?
- SSMS: SQL Server Management Studio
- ¿Qué es una columna de identidad?
- Exportar e importar una base de datos

SQL

- ¿Qué es SQL?
- Clasificación de SQL

Data Manipulation Language (DML)

- Sentencias DML
- INSERT: Crear registros
- SELECT: Realizar consultas
- INSERT usando SELECT
- Operadores de comparación
- Operadores matemáticos
- Cláusula LIKE
- Operadores lógicos
- Otros operadores
- Operador ORDER BY
- Operador DISTINCT
- Consultas con relaciones entre tablas
- Funciones de agregación
- GROUP BY y HAVING
- UPDATE: Modificar registros
- DELETE: Eliminar registros

Antes de empezar...

Descripción de la materia

Docentes y comisiones

Regularización de la materia

Exámenes parciales

Material, sitios y herramientas
de la cursada

Slack

Preparar el entorno de trabajo

Guía de estilos

Recomendaciones

Resúmenes (cheatsheets)

Clases de consulta



Normas académicas y
administrativas



las características de edición múltiple de editores como Sublime Text o Visual Studio Code.

Pueden encontrarlos todos en [Visual Studio Marketplace](#).

Entorno de base de datos

Instalación del entorno de base de datos

Sobre el final de materia necesitaremos un motor y un IDE especializados para trabajo con bases de datos.

- **SQL Server (Desarrollador):** <https://www.microsoft.com/es-es/sql-server/sql-server-downloads>
- **SQL Management Studio:** <https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15>





01.

Bases de datos



¿Qué es una **BASE** **DE DATOS**?

Una **base de datos** es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.

- Persisten una gran cantidad de información.
- Segurizan el acceso a los datos.
- Organizan y estructuran los datos por tipo y relación.

Tablas

Son una colección de datos relacionados organizados en filas y columnas.

Las **columnas** (*atributos*) definen un conjunto de datos de un tipo particular. Las filas tomarán un valor concreto para cada columna de la tabla.

Una **fila** (*tupla / registro*) contiene un valor específico para cada columna. Representan un conjunto de datos relacionados con la misma estructura, la cual está definida por las columnas.

Un **campo** (*celda*) es donde una columna y una fila se intersectan. Contiene un valor concreto definido por la columna y perteneciente al conjunto de datos de la fila.

Emp_No	Apellido	Oficio	Dir	Fecha_Alt	Salario	Comision	Dept_No
7119	SERRA	DIRECTOR	7839	1983-11-19 00:00:00	225000.00	39000.00	20
7322	GARCIA	EMPLEADO	7119	1982-10-12 00:00:00	129000.00	0.00	20
7369	SANCHEZ	EMPLEADO	7902	1980-12-17 00:00:00	10400.00	0.00	20
7499	ARROYO	VENDEDOR	7698	1981-02-22 00:00:00	208000.00	39000.00	30
7521	SALA	VENDEDOR	689	1981-02-22 00:00:00	162500.00	65000.00	30
7566	JIMENEZ	DIRECTOR	7839	1981-04-02 00:00:00	386750.00	0.00	20
7654	MARTIN	VENDEDOR	7698	1981-09-28 00:00:00	182000.00	182000.00	30
7698	NEGRO	DIRECTOR	7839	1981-05-01 00:00:00	370500.00	0.00	30
7782	CEREZO	DIRECTOR	7839	1981-06-09 00:00:00	318500.00	0.00	10
7788	NINO	ANALISTA	7566	1987-03-30 00:00:00	390000.00	0.00	20
7839	REY	PRESIDENTE	0	1981-11-17 00:00:00	650000.00	0.00	10
7844	TOVAR	VENDEDOR	7698	1981-09-08 00:00:00	195000.00	0.00	30
7876	ALONSO	EMPLEADO	7788	1987-05-03 00:00:00	143000.00	0.00	20
7900	JIMENO	EMPLEADO	7698	1981-12-03 00:00:00	123500.00	0.00	30
7902	FERNA...	ANALISTA	7566	1981-12-03 00:00:00	390000.00	0.00	20
7934	MUÑOZ	EMPLEADO	7782	1982-06-23 00:00:00	169000.00	0.00	10

Tipos de datos

Tipo	Descripción
bit	Valor binario 1 o 0.
int	Número entero con o sin signo.
float	Número en coma flotante de precisión simple.
char(n)	Almacena una cadena alfanumérica de longitud fija. Rellena con espacios vacíos.
varchar(n)	Almacena una cadena alfanumérica de longitud variable.
date	Fecha con formato <i>año-mes-día</i> .
datetime	Fecha y hora con formato <i>año-mes-día horas:minutos:segundos</i> .

Tipos de datos

CUANDO DESCUBRES QUE HAY
OTROS TIPOS DE DATOS EN SQL





¿Qué es una **BASE DE DATOS RELACIONAL**?

Llamamos así a toda base de datos que cumple con el **modelo relacional**.

Dicho paradigma consiste en generar **relaciones entre los datos guardados en diferentes tablas**, y a través de estas relaciones, conectar dichas tablas.

Las relaciones se llevarán a cabo a través de campos especiales, conocidos como **claves primarias (*primary key*)** y **claves foráneas (*foreign key*)**.

Modelo entidad-relación

Llamamos **modelo de datos** a una colección de conceptos empleados para describir la estructura de una base de datos.

Entidad

Una representación de un objeto, ser o concepto del mundo real que se describe en una base de datos.

DB: *tablas.*

Atributo

Las entidades tienen atributos que representan sus características o propiedades.

DB: *columnas de las tablas.*

Relaciones

La descripción de ciertas interdependencias entre una o más entidades.

DB: *claves primarias y foráneas.*



Claves primarias y foráneas

Clave primaria (PK - Primary Key)

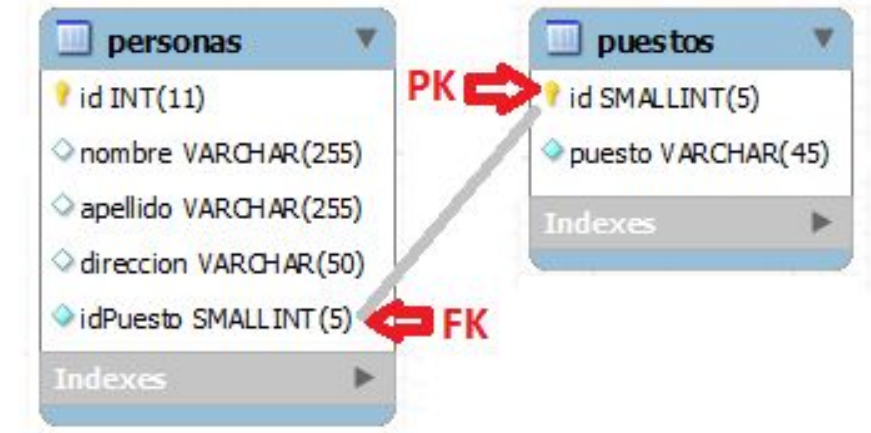
Una columna (o combinación de varias) que identifica de manera única a cada fila.

- *No pueden ser null y no pueden repetirse.*
- *Pueden ser cualquier tipo de dato.*
- *Están indexadas por defecto.*

Clave foránea (FK - Foreign Key)

Una columna (o combinación de varias) de una tabla que refiere a una columna (o combinación de varias) de otra tabla.

- *Normalmente apuntan a la PK de otra tabla.*

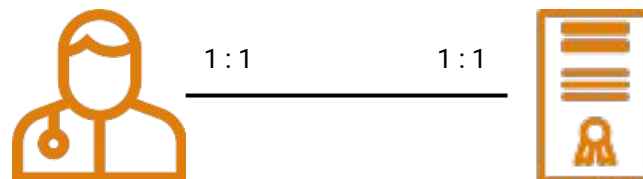




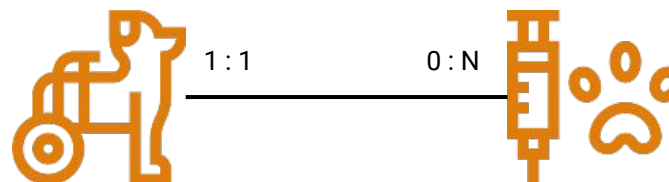
¿Qué es la CARDINALIDAD?

En **modelado de datos** se llama **cardinalidad** al número de instancias o elementos de una entidad que pueden asociarse a un elemento de la otra entidad relacionada.

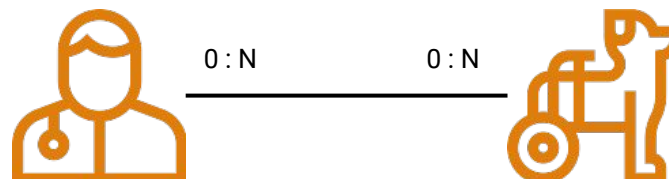
Un veterinario tiene una única matrícula y la matrícula está asociada a un único veterinario.



Una mascota puede tener ninguna o muchas vacunas, pero cada vacuna sólo fue asignada a una mascota.



Un veterinario puede atender a más de una mascota, y cada mascota pudo haber sido atendida por más de un veterinario.



¿Qué es la cardinalidad?

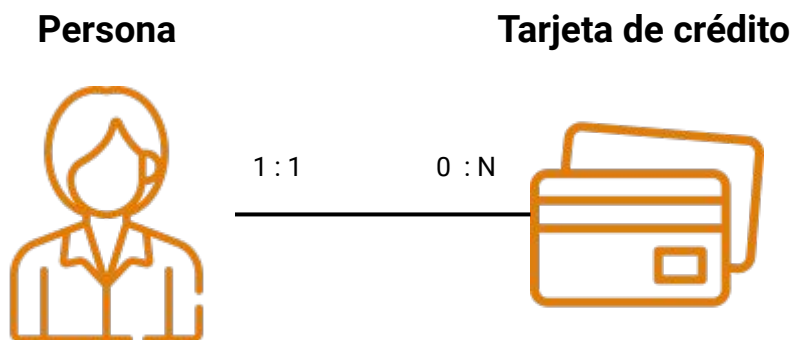
Uno a uno

Una instancia de la *Tabla A* se puede relacionar con una y sólo una instancia de la *Tabla B*.



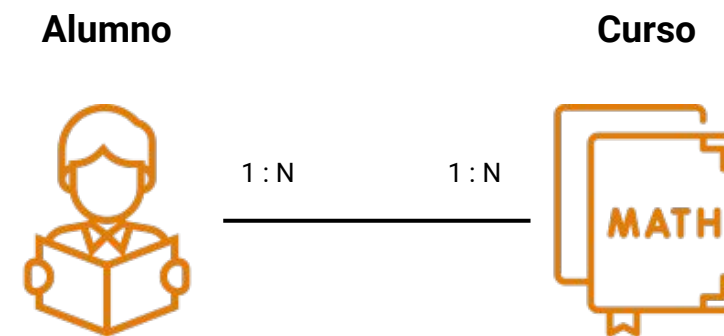
Uno a muchos

Una instancia de la *Tabla A* se puede relacionar con varias instancias de la *Tabla B*, pero una instancia de la *Tabla B* sólo puede estar relacionada a una instancia de la *Tabla A*.



Muchos a muchos

Varias instancias de la *Tabla A* se podrán relacionar con varias instancias de la *Tabla B* y viceversa.





¿Qué es un DBMS?

Un **sistema de administración de bases de datos (DBMS)** es un conjunto de programas que permiten el almacenamiento, modificación y extracción de la información en una base de datos.

Permite:

- Controlar el acceso a los datos.
- Asegurar la integridad de los datos.
- Gestionar el acceso concurrente a ellos.
- Recuperar los datos tras un fallo del sistema.
- Hacer copias de seguridad.

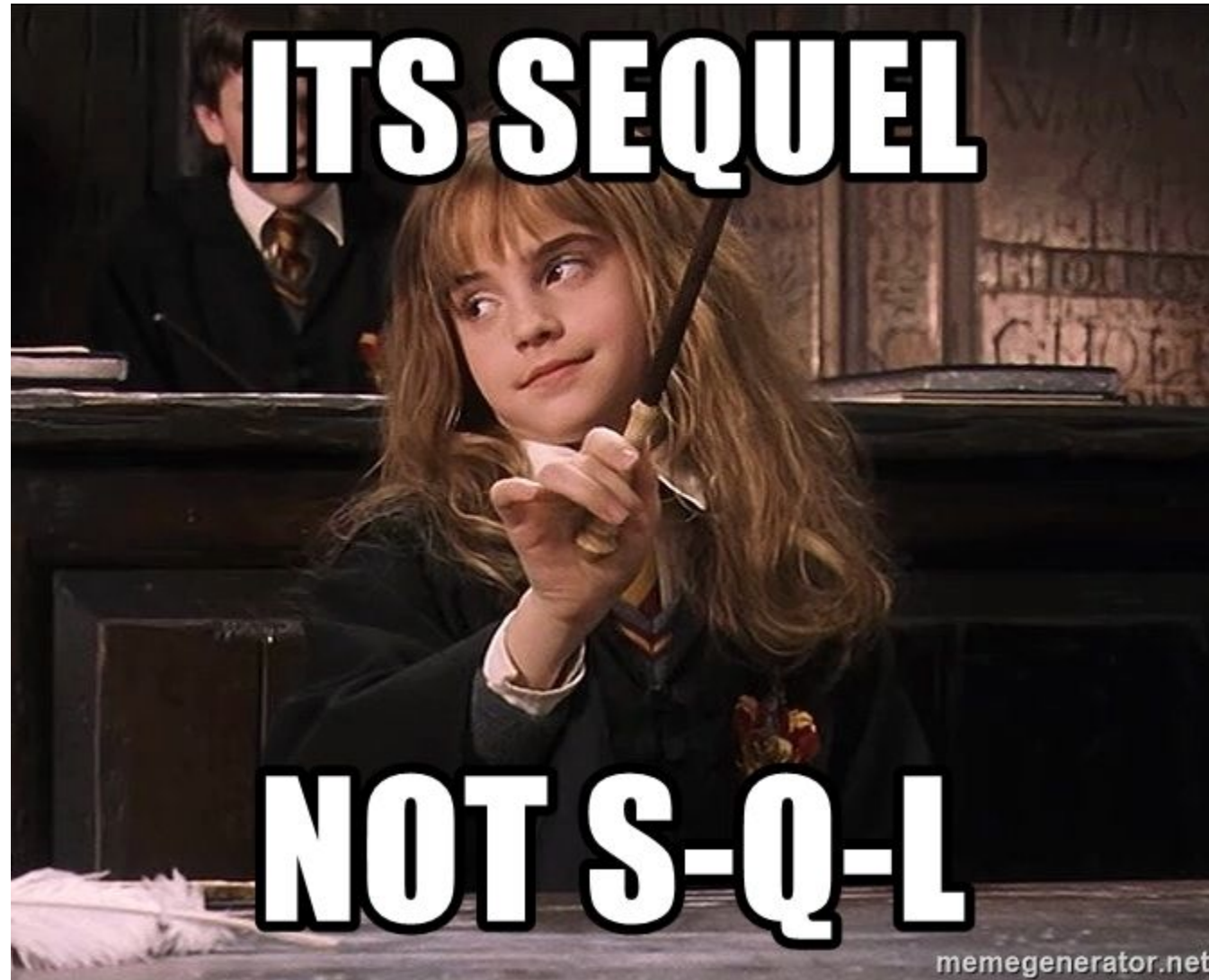
RDBMS: Relational Database Management Systems



RDBMS: Relational Database Management Systems



RDBMS: Relational Database Management Systems



SSMS: SQL Server Management Studio

SQL Server Management Studio (SSMS)
es una **interfaz gráfica de usuario (GUI)**
para administrar y trabajar con bases de
datos SQL Server.

UTN_DB



ALUMNOS			
	Column Name	Data Type	Allow Nulls
🔑	ID_ALUMNO	int	<input type="checkbox"/>
	NOMBRE	varchar(100)	<input type="checkbox"/>
	APELLIDO	varchar(100)	<input type="checkbox"/>
	PROMEDIO	float	<input checked="" type="checkbox"/>
	ESTA_ACTIVO	bit	<input type="checkbox"/>
	FECHA_INSCRIPCION	date	<input type="checkbox"/>
	FECHA_GRADUACION	date	<input checked="" type="checkbox"/>
	TELEFONO	varchar(20)	<input checked="" type="checkbox"/>
	SE_RECIBIO	bit	<input type="checkbox"/>
	ID_MATERIA	int	<input checked="" type="checkbox"/>

MATERIAS			
	Column Name	Data Type	Allow Nulls
🔑	ID_MATERIA	int	<input type="checkbox"/>
	DESCRIPCION	varchar(200)	<input type="checkbox"/>
	CUATRIMESTRE	int	<input type="checkbox"/>
			<input type="checkbox"/>



¿Qué es una COLUMNA DE IDENTIDAD?

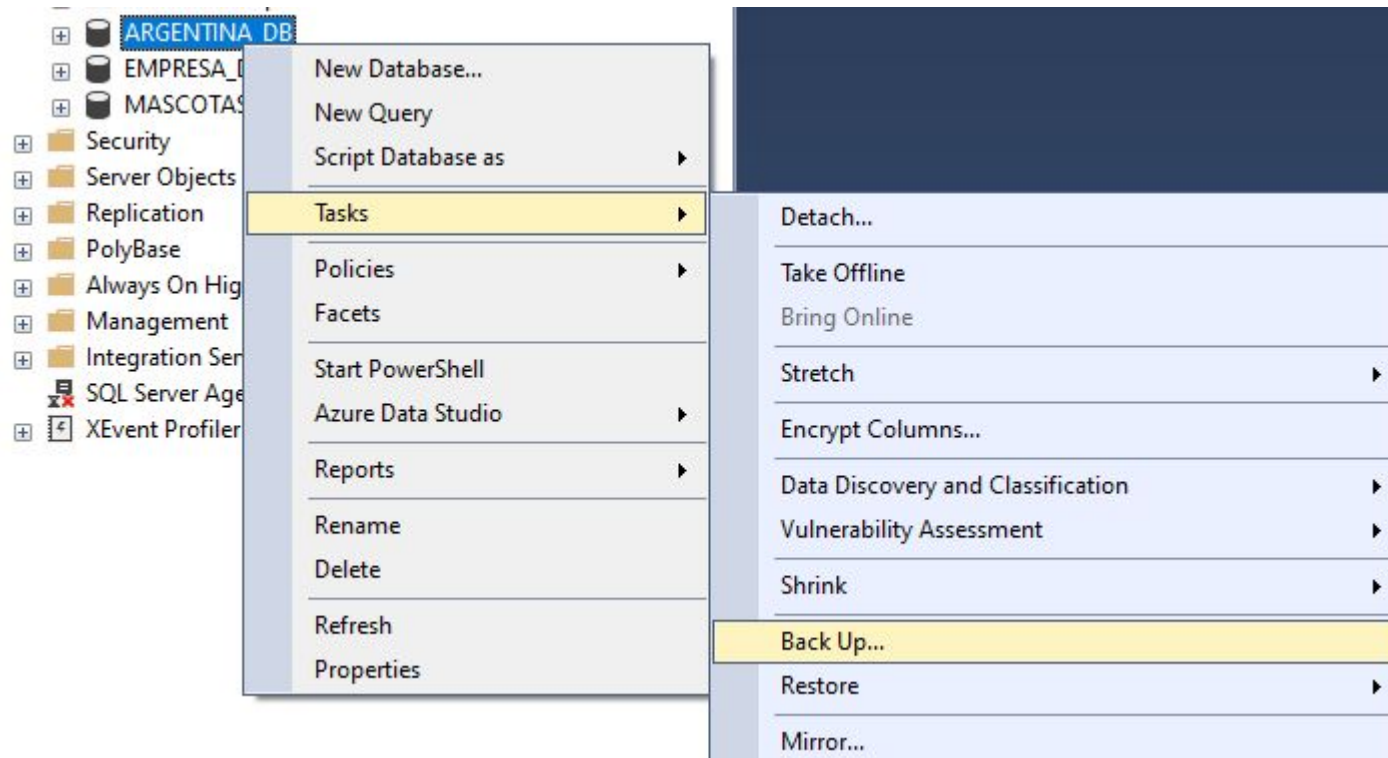
Una **columna de identidad** (*identity column*) es una columna cuyos valores son generados automáticamente por la base de datos.

Una columna de identidad NO es una clave primaria y no asegura que el dato sea único.

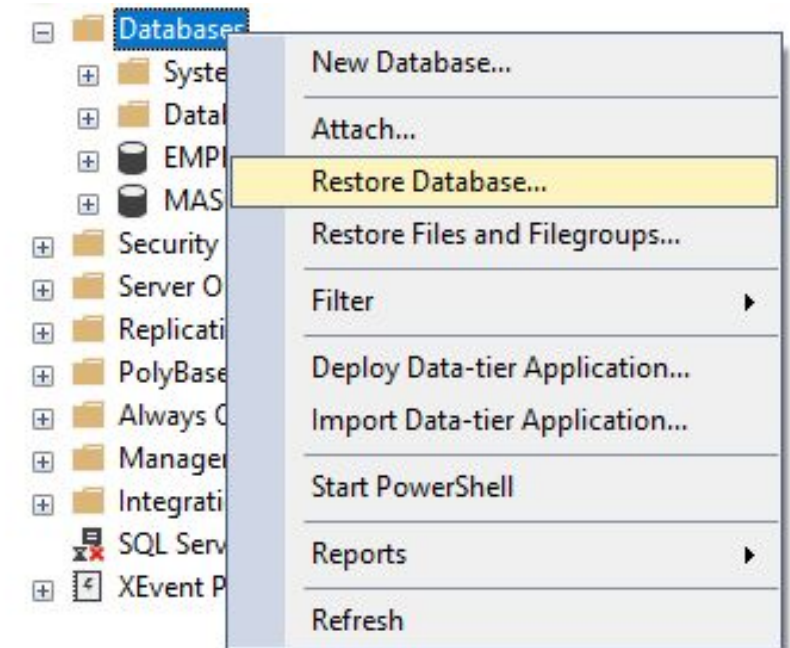
Sin embargo, muchas veces una clave primaria también se marca como una columna de identidad.

Exportar e importar una base de datos

EXPORTAR



IMPORTAR



Ejercicios



- Ejercicio I01 - Empleados y registrados - **Punto 1.**

https://codeutnfra.github.io/programacion_2_laboratorio_2_apuntes/



02.

SQL



¿Qué es SQL?

SQL (Structured Query Language) es un lenguaje de programación diseñado para administrar datos en un sistema de gestión de bases de datos relacionales (RDBMS).



Clasificación de SQL

DDL - Data Definition Language

Aquellas instrucciones que modifican la estructura de objetos una base de datos.

CREATE - ALTER - DROP - TRUNCATE

DML - Data Manipulation Language

Aquellas instrucciones que permiten consultar, agregar, modificar y eliminar los datos dentro de las tablas.

SELECT - INSERT - UPDATE - DELETE



03.

Data Manipulation Language (DML)

Sentencias DML

INSERT

Crear nuevas filas en una tabla.

SELECT

Consultar los datos de una tabla.

UPDATE

Modificar uno o varios datos de una o más filas en una tabla.

DELETE

Eliminar una o más filas de una tabla.



CREATE

C



READ

R



UPDATE

U



DELETE

D

INSERT: Crear registros

INSERT INDIVIDUAL

```
INSERT INTO nombre_tabla (columna1, [columna2, ...]) VALUES (valor1, [valor2, ...]);
```

```
INSERT INTO personas (nombre,apellido) VALUES ('José', 'Pérez');
```

INSERT MÚLTIPLE

```
INSERT INTO nombre_tabla (columna1, [columna2, ...]) VALUES  
(valor1, [valor2, ...]), (valor1, [valor2, ...]), ...;
```

```
INSERT INTO personas (nombre,apellido)  
VALUES ('Ernesto','Giménez'),('Alberto','Gómez');
```



CREATE



Ejercicios



- Ejercicio I01 - Empleados y registrados - **Puntos 1, 2 y 3.**

https://codeutnfra.github.io/programacion_2_laboratorio_2_apuntes/

SELECT: Realizar consultas

```
SELECT columna1,[columna2, ...] FROM nombre_tabla;
```

```
SELECT nombre FROM personas;
```

El comodín “*” trae todas las columnas de la tabla.

```
SELECT * FROM personas;
```

Las consultas se pueden filtrar por medio de la **sentencia WHERE**.

```
SELECT nombre FROM personas WHERE id = 1;
```



READ

R

SELECT: Realizar consultas



INSERT usando SELECT

```
INSERT INTO nombre_tabla (columna1, [columna2, ...])  
SELECT columna1, [columna2, ...] FROM nombre_tabla2;
```

```
INSERT INTO personas2 (nombre, apellido)  
SELECT nombre, apellido  
FROM personas;
```



CREATE



Operadores de comparación

Operador	Descripción	Ejemplo
=	Igual	<code>SELECT * FROM personas WHERE apellido = 'perez';</code>
!=	Distinto	<code>SELECT * FROM personas WHERE pais != 'Argentina';</code>
>	Mayor que	<code>SELECT * FROM personas WHERE fecha_nacimiento > '1999-12-31';</code>
>=	Mayor o igual que	<code>SELECT * FROM personas WHERE salario >= 50000;</code>
<	Menor que	<code>SELECT * FROM alumnos WHERE fecha_alta < '2010-01-01';</code>
<=	Menor o igual que	<code>SELECT * FROM alumnos WHERE promedio <= 7.5;</code>
IS NULL	El valor es nulo	<code>SELECT * FROM personas WHERE telefono IS NULL;</code>
IS NOT NULL	El valor no es nulo	<code>SELECT * FROM personas WHERE profesion IS NOT NULL;</code>

Operadores matemáticos

Operador	Descripción
+	Suma
-	Resta
*	Multiplicación
/	División
%	Módulo

```
SELECT costo_bruto + impuestos AS "Costo neto"  
FROM productos;
```

```
SELECT nombre, apellido, sueldo AS "Sueldo Bruto",  
sueldo*0.83 AS "Sueldo Neto" FROM personas;
```

Cláusula LIKE

Comodín %

Representa cualquier número de caracteres, sin importar cuáles.

```
SELECT nombre, apellido, sueldo FROM personas WHERE nombre LIKE '%a%';
```

Comodín _

Representa un carácter (sólo uno), cualquiera sea.

```
SELECT nombre, apellido, sueldo FROM personas WHERE nombre LIKE '_na%';
```

```
SELECT nombre, apellido, sueldo FROM personas WHERE nombre LIKE 'J__n C%s';
```

Ejercicios



- Ejercicio I01 - Empleados y registrados - **Punto 4.**

https://codeutnfra.github.io/programacion_2_laboratorio_2_apuntes/

Operadores lógicos

AND

Se deben cumplir ambas condiciones.

```
SELECT nombre, apellido, sueldo FROM personas  
WHERE sueldo > 1000 AND direccion IS NOT NULL;
```

OR

Se debe cumplir al menos una de las condiciones.

```
SELECT nombre, apellido, sueldo FROM personas  
WHERE sueldo <= 1000 OR direccion IS NULL;
```


Otros operadores

Operador	Descripción	Ejemplo
BETWEEN	Incluido entre	<pre>SELECT nombre, apellido, sueldo FROM personas WHERE sueldo BETWEEN 1000 AND 2000;</pre>
NOT BETWEEN	No incluido entre	<pre>SELECT nombre, apellido, sueldo FROM personas WHERE sueldo NOT BETWEEN 1000 AND 2000;</pre>
IN	Compara que el valor coincida con al menos uno de una serie de valores.	<pre>SELECT nombre, apellido, sueldo FROM personas WHERE sueldo IN (1000,2000);</pre>
NOT IN	Compara que el valor NO coincida con al menos uno de una serie de valores.	<pre>SELECT nombre, apellido, sueldo FROM personas WHERE sueldo NOT IN (1000,2000);</pre>

Operadores ORDER BY

SQL File 1* x personas

```
1 SELECT nombre, apellido
2 FROM personas
3 ORDER BY apellido;
```

Filter: Export:

	nombre	apellido
►	Federico	Estevez
	Ernesto	Giménez
	Alberto	Gómez
	Gerardo	González
	Juan	López
	Gonzalo	Martínez
	Jorge	Pérez

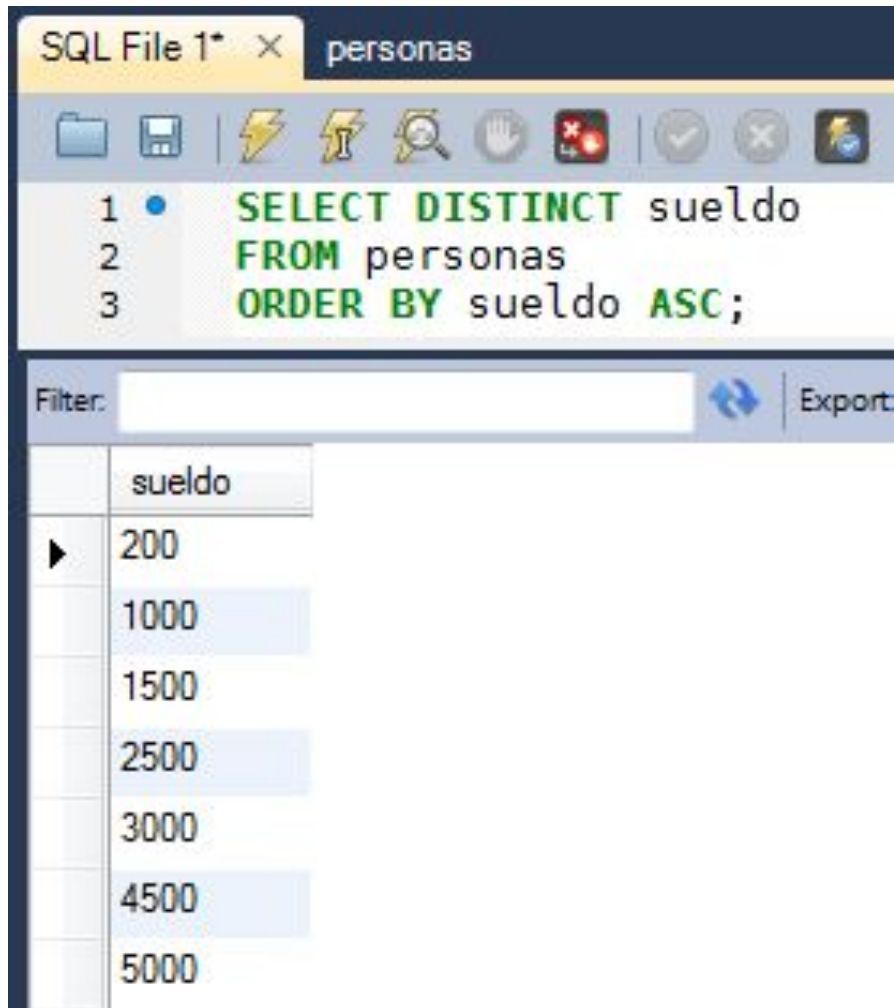
SQL File 1* x personas

```
1 SELECT nombre, apellido
2 FROM personas
3 ORDER BY apellido DESC;
```

Filter: Export:

	nombre	apellido
►	Jorge	Pérez
	Gonzalo	Martínez
	Juan	López
	Gerardo	González
	Alberto	Gómez
	Ernesto	Giménez
	Federico	Estevez

Operadores DISTINCT y TOP



The screenshot shows a SQL IDE window with a toolbar and a query editor. The query editor contains the following SQL code:

```
1 SELECT DISTINCT sueldo
2 FROM personas
3 ORDER BY sueldo ASC;
```

Below the query editor, there is a 'Filter:' input field and an 'Export' button. The results are displayed in a table with a single column named 'sueldo'.

sueldo
200
1000
1500
2500
3000
4500
5000

```
SELECT TOP(3) sueldo
FROM personas
ORDER BY sueldo DESC;
```

Ejercicios



- Ejercicio I01 - Empleados y registrados - **Puntos 5 y 6.**

https://codeutnfra.github.io/programacion_2_laboratorio_2_apuntes/

Consultas con relaciones entre tablas

JOIN o INNER JOIN

Retorna la **intersección** de dos conjuntos de datos (tablas). Trae las filas que coinciden en la condición del JOIN.

```
SELECT s.sector, p.puesto FROM sector s INNER JOIN puestos p ON s.id = p.idSector;
```

LEFT JOIN o LEFT OUTER JOIN

Retorna la **intersección** de dos tablas y todos los datos de la tabla de la izquierda.

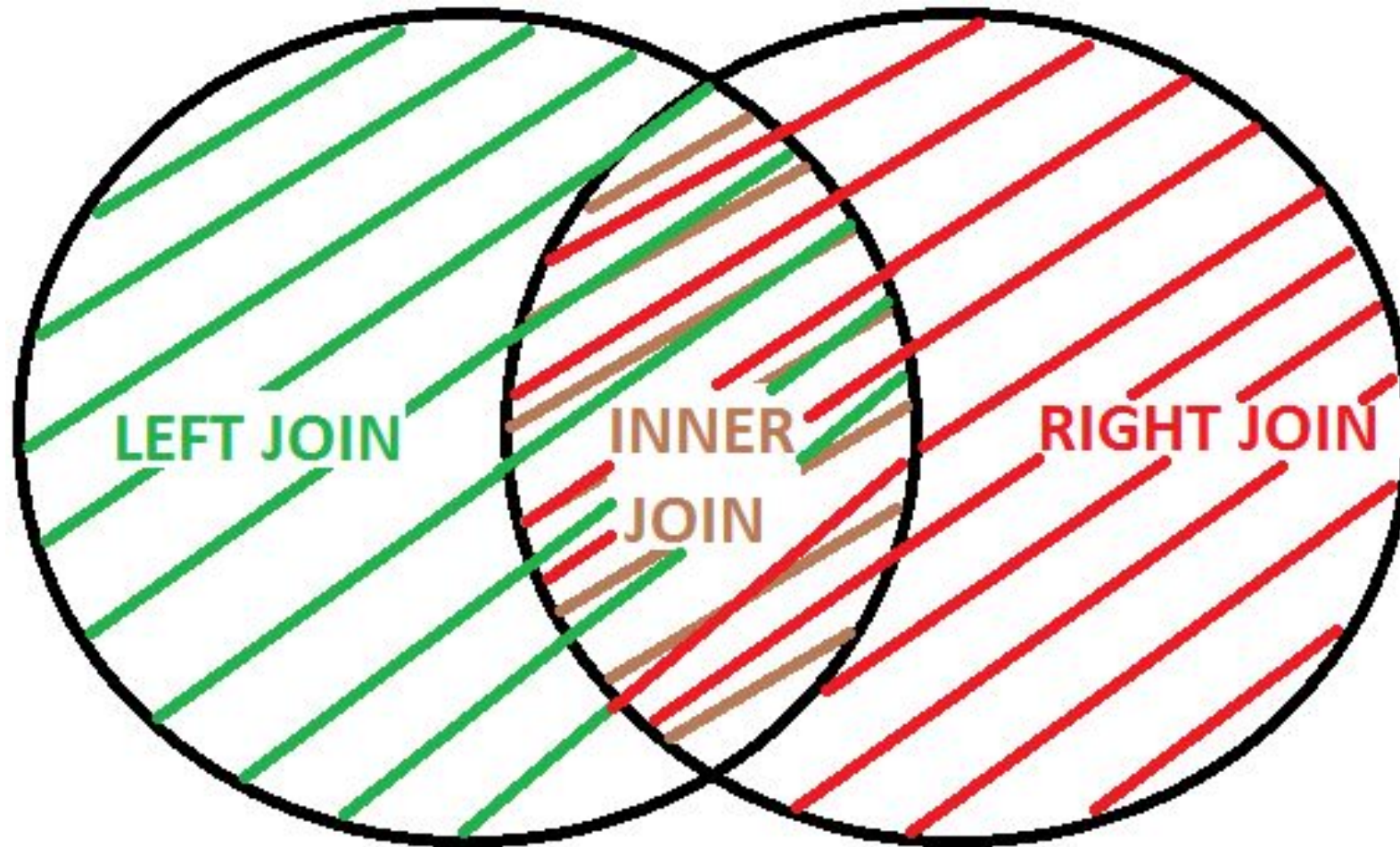
```
SELECT s.sector, p.puesto FROM sector s LEFT JOIN puestos p ON s.id = p.idSector;
```

RIGHT JOIN o RIGHT OUTER JOIN

Retorna la **intersección** de dos tablas y todos los datos de la tabla de la derecha.

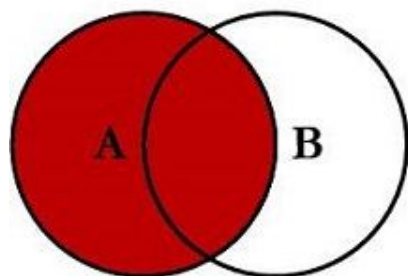
```
SELECT s.sector, p.puesto FROM sector s RIGHT JOIN puestos p ON s.id = p.idSector;
```

Consultas con relaciones entre tablas

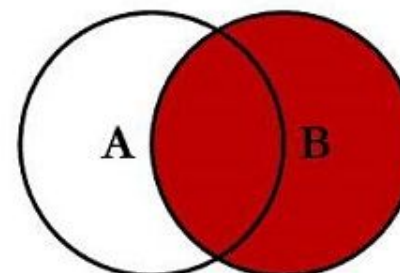


Consultas con relaciones entre tablas

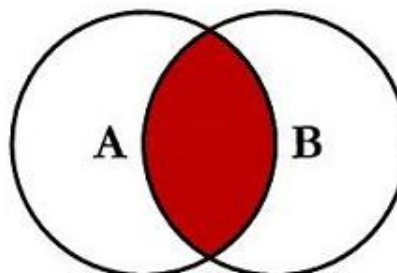
SQL JOINS



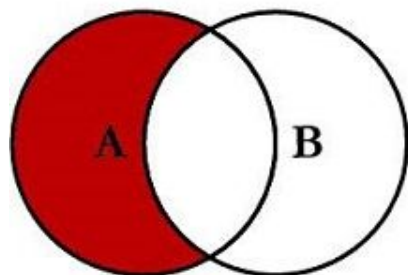
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key
```



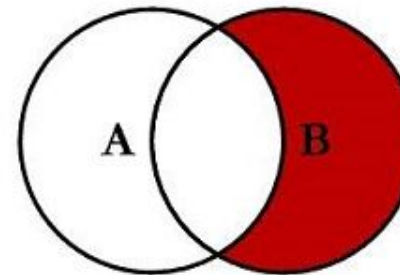
```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key
```



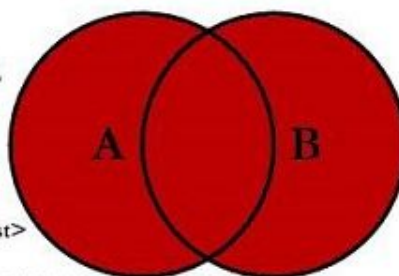
```
SELECT <select_list>  
FROM TableA A  
INNER JOIN TableB B  
ON A.Key = B.Key
```



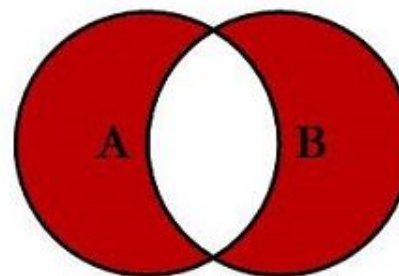
```
SELECT <select_list>  
FROM TableA A  
LEFT JOIN TableB B  
ON A.Key = B.Key  
WHERE B.Key IS NULL
```



```
SELECT <select_list>  
FROM TableA A  
RIGHT JOIN TableB B  
ON A.Key = B.Key  
WHERE A.Key IS NULL
```



```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.Key = B.Key
```



```
SELECT <select_list>  
FROM TableA A  
FULL OUTER JOIN TableB B  
ON A.Key = B.Key  
WHERE A.Key IS NULL  
OR B.Key IS NULL
```


Consultas con relaciones entre tablas

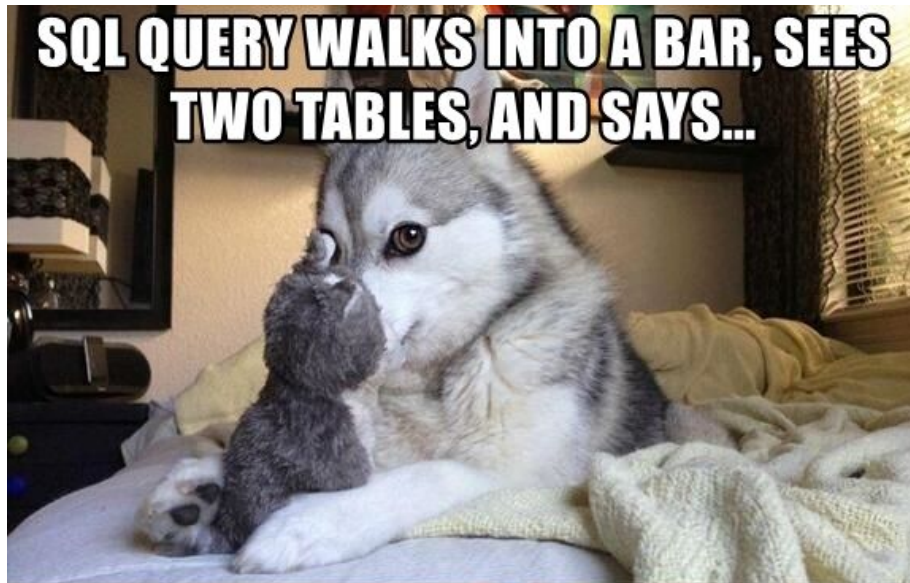
JOIN con más de dos tablas

```
SELECT pe.nombre, pe.apellido, s.sector, pu.puesto
FROM sector s
INNER JOIN puestos pu ON s.id = pu.idSector
INNER JOIN personas pe ON pu.id = pe.idPuesto;
```

UNION

```
SELECT s.sector, p.puesto
FROM sector s LEFT JOIN puestos p ON s.id = p.idSector
UNION
SELECT s.sector, p.puesto
FROM sector s RIGHT JOIN puestos p ON s.id = p.idSector;
```

Consultas con relaciones entre tablas



Ejercicios



- Ejercicio I01 - Empleados y registrados - **Punto 7.**

https://codeutnfra.github.io/programacion_2_laboratorio_2_apuntes/

Consultas con relaciones entre tablas



Consultas con relaciones entre tablas



Funciones de agregación

Operador	Descripción	Ejemplo
MAX()	Busca el valor máximo.	<code>SELECT MAX(sueldo) FROM personas;</code>
MIN()	Busca el valor mínimo.	<code>SELECT MIN(sueldo) FROM personas;</code>
AVG()	Saca el promedio de los valores, descartando nulos.	<code>SELECT AVG(sueldo) FROM personas;</code>
SUM()	Suma todos los valores.	<code>SELECT SUM(sueldo) FROM personas;</code>
COUNT()	Cuenta la cantidad de ocurrencias.	<code>SELECT COUNT(*) FROM personas WHERE sueldo >= 50000;</code>
CONCAT()	Concatena cadenas de texto.	<code>SELECT CONCAT(nombre, ' ', apellido) AS nombre_completo FROM personas;</code>

Ejercicios



- Ejercicio I01 - Empleados y registrados - **Punto 8.**

https://codeutnfra.github.io/programacion_2_laboratorio_2_apuntes/

GROUP BY y HAVING

```
SELECT campos FROM tabla [WHERE condicion]
GROUP BY campoAgrupacion
[HAVING condicionAgrupamiento];
```

```
SELECT pu.puesto, AVG(pe.sueldo)
FROM personas pe INNER JOIN puestos pu ON pe.idPuesto=pu.id
GROUP BY pu.id;
```

```
SELECT pu.puesto, AVG(pe.sueldo) AS promedioSueldo
FROM personas pe INNER JOIN puestos pu ON pe.idPuesto=pu.id
GROUP BY pu.id
HAVING promedioSueldo > 1000;
```

Resumen

JOIN

Intersección entre tablas relacionadas.

WHERE

Aplicar filtros a una consulta o comando.

GROUP BY

Agrupar los resultados.

HAVING

Filtrar en base a un agrupamiento de datos.

ORDER BY

Ordenar los datos.



Ejercicios



- Ejercicio I01 - Empleados y registrados - **Punto 9.**

https://codeutnfra.github.io/programacion_2_laboratorio_2_apuntes/

UPDATE: Modificar registros

```
UPDATE nombre_tabla  
SET columna1=valor1, [columna2=valor2, ...]  
WHERE [...];
```

```
UPDATE personas  
SET nombre='Jorge'  
WHERE id = 1;
```



UPDATE

U

Si no se coloca el WHERE, se actualizan TODOS los registros de la tabla.

DELETE: Eliminar registros

```
DELETE FROM nombre_tabla WHERE [...];
```

```
DELETE FROM personas WHERE nombre = 'José';
```



DELETE

D

Si no se coloca el WHERE, se eliminan TODOS los registros de la tabla.

Mucho muy importante...



Ejercicios



- Ejercicio I01 - Empleados y registrados - **Puntos 10 al 16.**
- Ejercicio I02 - La base del bufet

https://codeutnfra.github.io/programacion_2_laboratorio_2_apuntes/