educación IT
CODE YOUR FUTURE

Combinación de consultas: *UNION*

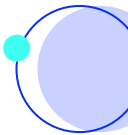
Combinación de consultas: *UNION*

Para **comparar los resultados de varias consultas y combinarlos** en un nuevo resultado basado en esa comparación, existe (entre otros) el operador ***UNION***.

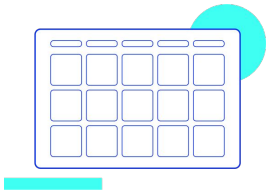
Dado que se compararán varias consultas, es necesario que:

- **En cada resultado exista la misma cantidad de campos**.
- Los campos a comparar tengan **tipos de datos compatibles** *(no es necesario que tengan el mismo nombre).*

Debemos **diferenciar** la comparación de los resultados de varias consultas y su combinación y, por otro, la **combinación de datos de varias tablas en una consulta**: para esto último usamos el operador ***JOIN***, que establece una relación por uno o varios campos con otras tablas.



Las dos consultas se pueden hacer sobre la misma tabla, pero podría ser cualquier consulta siempre y cuando se respete que la salida contenga la misma cantidad de campos y que los tipos de datos de los campos sean compatibles para la comparación.
En general, esto implica campos numéricos con campos numéricos y campos de texto con campos de texto.



UNION - UNION ALL

La sintaxis para **combinar dos consultas** mediante la cláusula ***UNION*** es:

```
consulta1 UNION [ALL] consulta2;
```

Esta sintaxis agrega el resultado de la ***consulta2*** al resultado de la ***consulta1***. Los registros que quedan duplicados, se eliminan (no se muestran en el resultado). **Para conservar los registros duplicados**, se utiliza la cláusula ***UNION ALL*** en lugar de ***UNION***.

La **primera consulta** que interviene se escribe **sin el punto y coma (;) al final**, ya que la consulta completa está formada por las consultas que intervienen y el operador ***UNION***, finalizando **después de la segunda consulta**. Esto es así para todas las combinaciones de consultas.

Gráficamente, podemos ver que el **resultado** está formado por todos los registros de la primera y de la segunda consulta:

Ejemplos de combinación de consultas

Veremos a continuación, algunos ejemplos donde se suponen **2 tablas**:

- Una tabla con el nombre ***nenes*** que contiene todos los nacimientos de bebés de sexo masculino, ocurridos durante el año 2020 en la República Argentina.
- Otra tabla con el nombre ***nenas*** que contiene todos los nacimientos de bebés de sexo femenino, ocurridos durante el año 2020 en la República Argentina.

Ejemplo 1

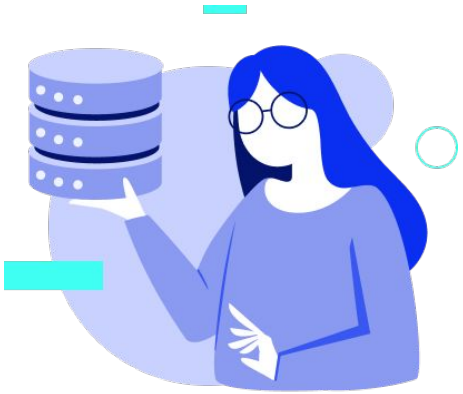
Se necesita obtener una **lista completa de todos los bebés nacidos en el año 2020 en la República Argentina**. La sentencia SQL sería la siguiente:

```
SELECT * FROM nenes
UNION
SELECT * FROM nenas;
```

Ejemplo 2

En el siguiente ejemplo, se suponen las mismas tablas que en el caso anterior. Y se necesita obtener una lista completa de **todos los bebés nacidos en el año 2020 en la República Argentina**, en la provincia de ***Córdoba***. La sentencia SQL sería la siguiente:

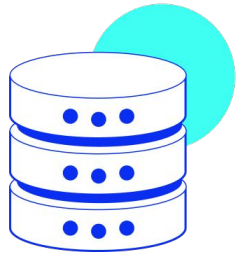
```
SELECT * FROM nenes WHERE provincia = ‘Córdoba’
UNION
SELECT * FROM nenas WHERE provincia = ‘Córdoba’;
```



Ejemplo 3

Ahora, se suponen las mismas tablas que en el caso anterior. Y se necesita obtener una lista completa de **todos los bebés nacidos en el año 2020 en la República Argentina**: bebés de **sexo masculino** nacidos en la provincia de ***Córdoba*** y bebas de **sexo femenino** nacidas en la provincia de ***La Pampa***. La sentencia SQL sería:

```
SELECT * FROM nenes WHERE provincia = ‘Córdoba’
UNION
SELECT * FROM nenas WHERE provincia = ‘La Pampa’;
```



Ejemplo 4

En este último ejemplo, se suponen las mismas tablas que en el caso anterior. Y se necesita obtener una lista completa de **todos los bebés nacidos durante el mes de agosto del año 2020 en la República Argentina**. La sentencia SQL sería la siguiente:

```
SELECT * FROM nenes WHERE MONTH(fecha_nacimiento) = 8
UNION
SELECT * FROM nenas WHERE MONTH(fecha_nacimiento) = 8;
```

Recuerda

La cláusula ***UNION*** **anula automáticamente los registros duplicados entre las tablas**. En el caso de querer mostrar los registros duplicados, la cláusula que deberás utilizar es ***UNION ALL***.

Esta es una forma de verificar si existen registros duplicados entre tablas. Es decir, si al utilizar la cláusula ***UNION*** se obtiene **la misma cantidad de registros resultantes** que al utilizar la cláusula ***UNION ALL***, esto mostraría que no existen registros duplicados entre las tablas consultadas.

Nota: a estas consultas también se las conoce con el nombre de ***consultas de unión externa***.

Uso de NULL en unión

Cuando se utiliza UNION en SQL, es un requisito asegurarse de que ambas partes de la consulta tengan la misma estructura de columnas, para, por ejemplo, evitar perder información que puede ser relevante, por lo que es necesario, a veces usar NULL en el código.

- **Uniformidad en las columnas:** al usar UNION, todas las consultas deben tener el mismo número de columnas y el mismo tipo de datos en cada columna. Si una de las consultas tiene columnas que no existen en la otra, necesitamos rellenar esas columnas con valores NULL en la consulta que no tiene esos datos.

En la siguiente *slide* podrás ver un ejemplo:

Ejemplo

Cuando trabajamos con tablas relacionadas, como las mostradas en este ejemplo (Empleados y Departamentos), puede surgir la necesidad de generar una consulta que incluya información completa de ambas tablas o que combine valores de diferentes atributos.

En estas situaciones, el uso de NULL resulta esencial para completar los datos en columnas que no están presentes en una de las tablas involucradas.

TABLA Empleados:

- ID
- Nombre
- Apellido
- DepartmentoID

TABLA Departamentos:

- DepartmentoID
- NombreDepartamento

Combinación de datos de empleados y departamentos

Se desea combinar datos de ambas tablas en una consulta para obtener una lista que incluya tanto empleados como departamentos. Aunque las tablas tienen una estructura diferente, puedes usar NULL para crear columnas vacías en las partes que no tengan información relevante.

```
SELECT
    ID,
    Nombre,
    Apellido,
    DepartamentoID,
    NULL AS NombreDepartamento
FROM EMPLEADOS
UNION
SELECT
    NULL AS ID,
    NULL AS Nombre,
    NULL AS Apellido,
    DepartamentoID,
    NombreDepartamento
FROM DEPARTAMENTOS;
```

¡Sigamos trabajando!