BASES DE DATOS

Clase 15 - Lenguaje de consulta formal
Algebra Relacional

Profesora: Erika Gutiérrez Beltrán

Tema 2: Bases de datos relacionales

Objetivo

Transformar los modelos conceptuales a lógicos. Construir bases de datos para datos estructurados

Temario

- Utilización de los modelos en el mundo real ✅
- Transformación del modelado a la base de datos ✅
- Clases de entidades ✅
- Modelo relacional, restricciones de integridad y normalización ✅
- Lenguaje de consulta formal
- Lenguaje de consulta SQL ✅
- Arquitectura de los sistemas gestores de Bases de datos, OLTP, OLAP

Logros

Capturar datos desde una pantalla de usuario real y almacenarlos en una base de datos. Manipulación de estos datos.

Recordemos las propiedades ACID

A: Atomicidad, los cambios sobre los datos se realizan en su totalidad o los datos no son modificados si hay algún error. Esta propiedad garantiza que todas las transacciones se confirmen de manera exitosa.

C: Consistencia, cualquier cambio realizado sobre un registro o dato debe conducir a un estado válido de acuerdo a las restricciones.

I: Aislamiento – isolation. Un acambio sobre un dato o registro no debe afectar a otros.

D: Durabilidad, una vez ejecutado el cambio debe mantenerse en el tiempo.

Comandos básicos aprendidos hasta el momento

Como hemos visto en ocasiones anteriores, en PostgreSQL es posible acceder a datos de una tabla por medio de la instrucción SELECT

```
SELECT nombre, fecha_nacimiento, edad FROM "Usuario" WHERE id = 1;
```

Consultas combinadas – JOIN SQL

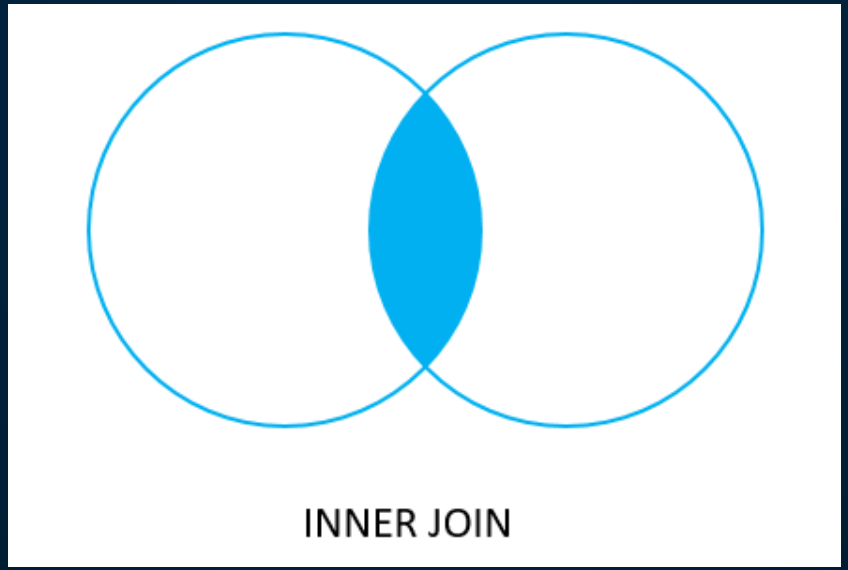
Sin embargo, si queremos acceder a múltiple información de varias tablas, no haremos una consulta por cada una de las tablas. SQL nos ofrece la posibilidad de hacer consultas combinadas entre tablas que estén relacionadas.

Las combinaciones que encontraremos y que aprenderemos en esta clase son:

- INNER JOIN
- LEFT JOIN
- LEFT OUTER JOIN
- LEFT OUTER JOIN - only
- RIGHT JOIN
- RIGHT OUTER JOIN
- RIGHT OUTER JOIN - only
- FULL OUTER JOIN
- FULL OUTER JOIN - only

INNER JOIN

Esta sentencia permite unir dos tablas para este ejemplo utilizaremos las tablas del esquema “Grupo 3” (recuerda reemplazar por tus datos disponibles en tu esquema y tablas). Esta consulta hará coincidir los valores de las columnas. Ver el ejemplo en el diagrama de Venn.



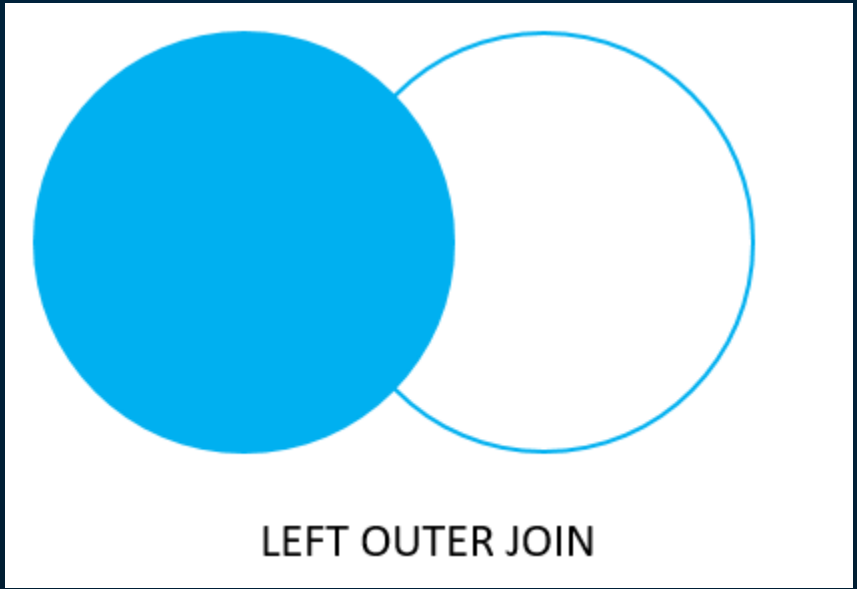
INNER JOIN

El INNER JOIN cruza dos tablas, y es necesario contar con una columna en común, ya que se comparan ambas columnas para encontrar coincidencias. Si no hay coincidencias, los registros no se muestran.

```sql
SELECT
    pelicula.id_pelicula,
    pelicula.titulo,
    pelicula.sinopsis,
    pelicula.cartel,
    genero.id_genero
FROM
     "Grupo 3"."PELICULA" as pelicula
INNER JOIN "Grupo 3"."GENERO-PELICULA" as genero
    ON pelicula.id_pelicula = genero.id_pelicula;
```

LEFT JOIN

Esta combinación selecciona datos de la tabla izquierda y compara con los valores de la tabla derecha. Cuando se hace la consulta, siempre aparecen todos registros de la tabla izquierda, y para el caso de la derecha solo se visualizarán los registros que coincidan.



LEFT JOIN

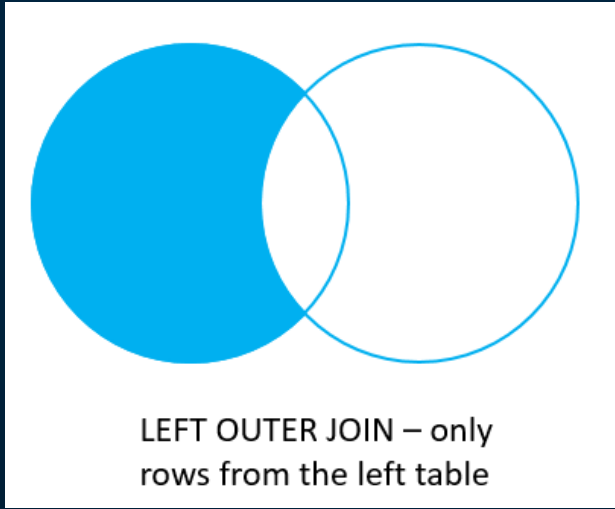
Cuando no se encuentran coincidencias en la segunda tabla, los valores de los registros serán NULL.

```
SELECT
    pe.id_perfil, pe.correo, pe.nombre, pf.id_perfil, pf.id_pelicula
FROM
     "Grupo 3"."PERFIL" as pe
LEFT JOIN "Grupo 3"."PELICULA-FAVORITA" as pf
    ON pf.id_perfil = pe.id_perfil;
```

LEFT JOIN

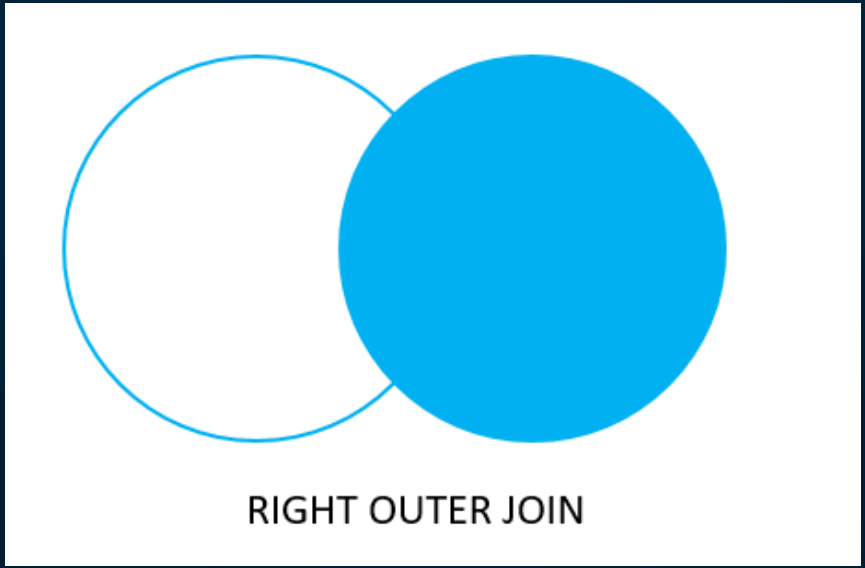
LEFT OUTER JOIN es lo mismo que LEFT JOIN (la palabra OUTER en la sentencia es opcional), por lo que puede utilizar cualquiera de los dos sin problema.

Para obtener solo las filas que se encuentran en la tabla izquierda en el WHERE agregar la restricción IS NULL



RIGHT JOIN

Esta combinación selecciona datos de la tabla derecha y compara con los valores de la tabla izquierda. Cuando se hace la consulta, siempre aparecen todos registros de la tabla derecha, y para el caso de la izquierda solo se visualizarán los registros que coincidan.



RIGHT JOIN

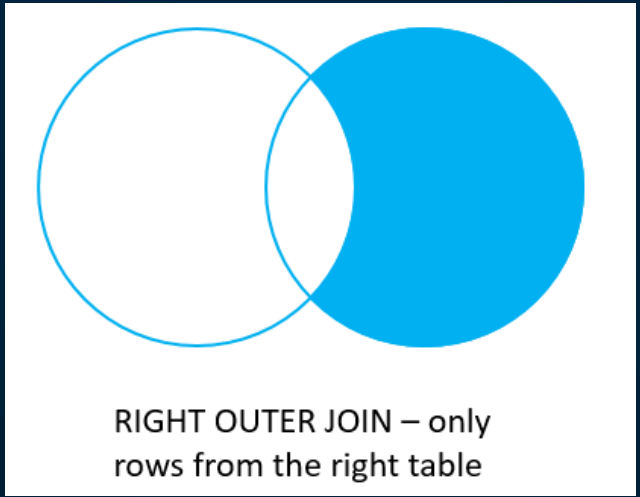
Cuando no se encuentran coincidencias en la tabla de la derecha (primera tabla), los valores de los registros serán NULL.

```
SELECT
    pe.id_perfil, pe.correo, pe.nombre, pf.id_perfil, pf.id_pelicula
FROM
     "Grupo 3"."PERFIL" as pe
RIGHT JOIN "Grupo 3"."PELICULA-FAVORITA" as pf
    ON pe.id_perfil = pf.id_perfil;
```

RIGHT JOIN

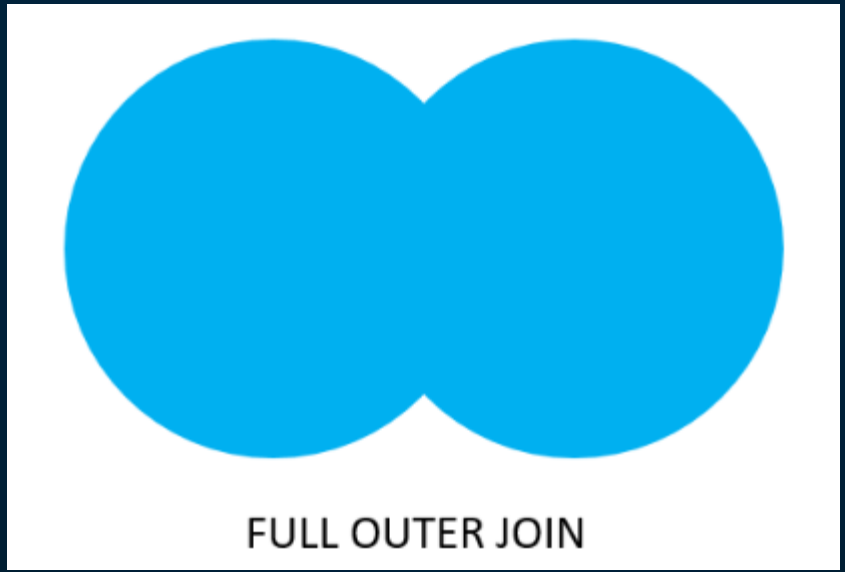
RIGHT OUTER JOIN es lo mismo que RIGHT JOIN (la palabra OUTER en la sentencia es opcional), por lo que puede utilizar cualquiera de los dos sin problema.

Para obtener solo las filas que se encuentran en la tabla derecha en el WHERE agregar la restricción IS NULL



FULL OUTER JOIN

Una sentencia de este tipo devuelve la unión o la combinación completa, tanto filas de derecha como de izquierda, con los valores que coinciden. Si no hay coincidencias los valores serán completados con NULL.



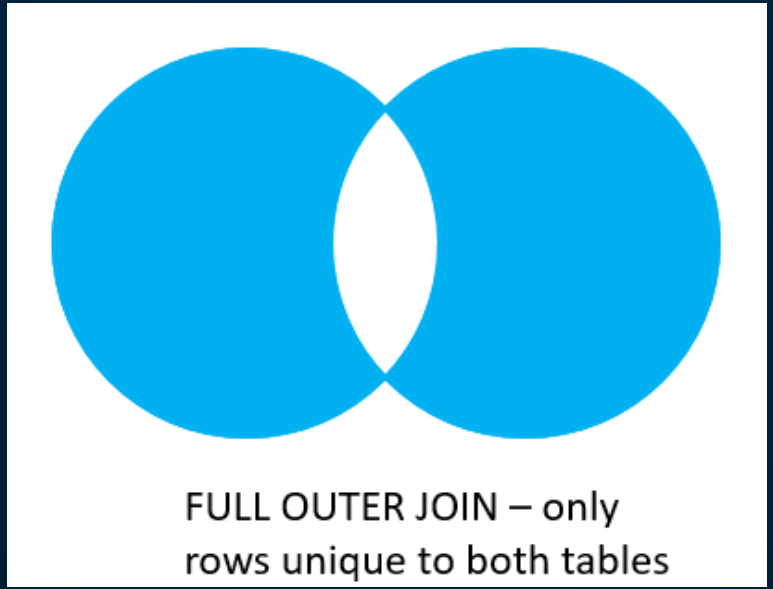
FULL OUTER JOIN

Cuando no se encuentran coincidencias en la tabla de la derecha o la tabla de la izquierda, los valores de los registros serán NULL.

```
SELECT
    pe.id_perfil, pe.correo, pe.nombre, pf.id_perfil, pf.id_pelicula
FROM
     "Grupo 3"."PERFIL" as pe
FULL JOIN "Grupo 3"."PELICULA-FAVORITA" as pf
    ON pe.id_perfil = pf.id_perfil;
```

FULL OUTER JOIN

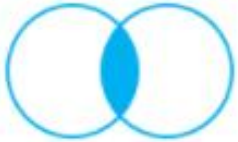
Para obtener solo las coincidencias externas, agregue IS NULL en la sentencia.



POSTGRESQL JOINS

POSTGRESQL TUTORIAL

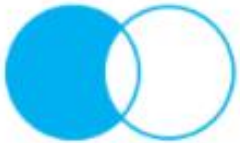
```
SELECT * FROM a
INNER JOIN b ON a.key = b.key
```



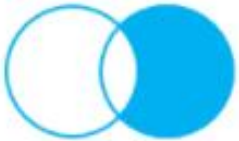
```
SELECT * FROM a
LEFT JOIN b ON a.key = b.key
```

```
SELECT * FROM a
RIGHT JOIN b ON a.key = b.key
```

```
SELECT * FROM a
LEFT JOIN b ON a.key = b.key
WHERE b.key IS NULL
```



```
SELECT * FROM a
RIGHT JOIN b ON a.key = b.key
WHERE a.key IS NULL
```

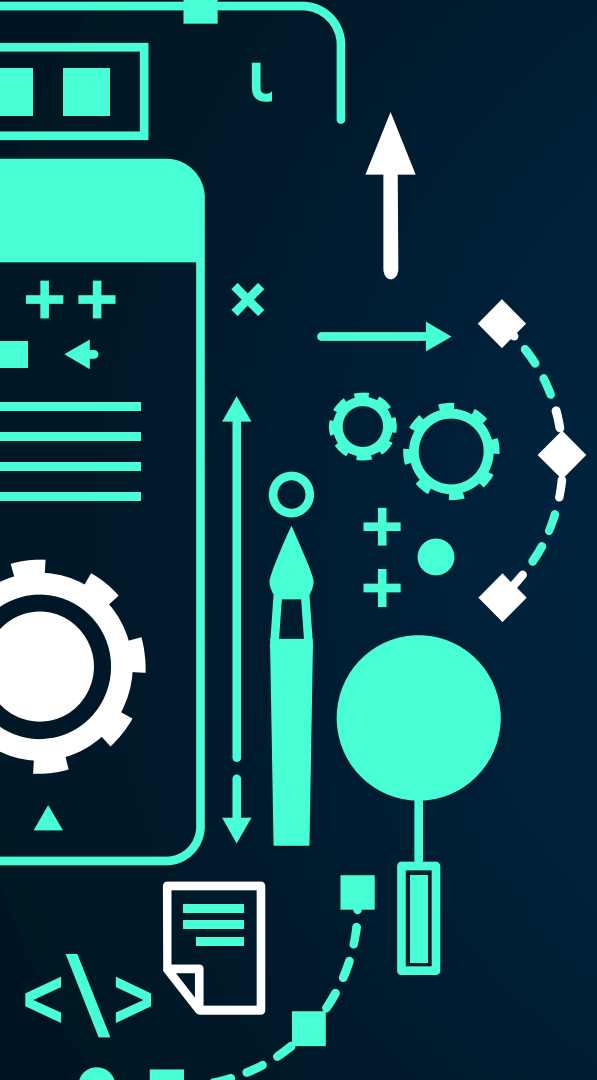


```
SELECT * FROM a
FULL JOIN b ON a.key = b.key
```

```
SELECT * FROM a
FULL JOIN b ON a.key = b.key
WHERE a.key IS NULL OR b.key IS NULL
```

REFERENCIAS

- Libro: Fundamentos de base de datos 5 edición, Silberschatz
- Tutorial de PostgreSQL: https://www.postgresqltutorial.com/postgresql-joins/
- Extending the database relational model to capture more meaning, IBM Research Laboratory



Gracias!