|  |  |
| --- | --- |
|  | Trabajo grupal o individual |

## Identificación del trabajo

|  |  |
| --- | --- |
| 1. **Módulo:** | 3 |
| 1. **Asignatura:** | Bases de Datos |
| 1. **RA:** | Relacionar Tablas creadas y poblarlas según indicaciones |
| 1. **Docente Online:** | **Jacob Felipe** |
| 1. **Fecha de entrega**: | 28 de noviembre de 2024 |
|  |  |
|  |  |

## Identificación del/los estudiante/s

|  |  |
| --- | --- |
| **Nombre y apellido** | **Carrera** |
| Christian Aránguiz Morales | Técnico en Programación |

## Introducción

En esta tercera iteración del proyecto de base de datos para la biblioteca de la Universidad Puls-AR, nuestro objetivo es consolidar las funcionalidades previamente desarrolladas, incorporando mejoras orientadas a un entorno profesional y manejando nuevas consultas para un análisis más profundo de los datos. A lo largo de esta actividad, exploraremos cómo optimizar la trazabilidad de los préstamos, obtener estadísticas relevantes, y trabajar con vistas que integren información clave del sistema, como los lectores y los libros que tienen en préstamo.

En el proceso, hemos tomado ciertas licencias para acercarnos a las prácticas comunes en ambientes profesionales. Por ejemplo, en lugar de eliminar registros relacionados con préstamos, optamos por actualizar un campo que refleja el estado del préstamo, lo que permite conservar el historial completo y garantizar la trazabilidad de los datos. Esta decisión responde tanto a las necesidades de escalabilidad del sistema como a la importancia de mantener registros históricos para análisis futuros.

Además, se han aplicado técnicas como la creación de vistas para centralizar información relevante y simplificar futuras consultas, así como el uso de funciones avanzadas para trabajar con fechas, cálculos y agrupaciones dinámicas. Estas mejoras aseguran que el sistema sea no solo funcional, sino también adaptable a nuevos requerimientos.

En resumen, esta iteración busca no solo cumplir con los requerimientos propuestos, sino también preparar la base de datos para un uso más robusto y profesional.

## Desarrollo

1. **Relación entre entidades base (Libros y Lectores).**

Tal como se mencionó (y profundizó) en la API anterior, la confección de las dos tablas troncales de la biblioteca de la Universidad Puls-AR (Libros y Lectores), no es ciertamente escalable ya que no se pueden *trackear* los estados de los préstamos realizados y quiénes los solicitaron.

Es por eso, que se decidió crear la tabla prestamos, la cuál será un punto intermedio entre ambas tablas anteriormente mencionadas:  
  
A black screen with white text

Description automatically generated

En el anterior diagrama, se puede apreciar que la tabla prestamos conecta a libro y lector para registrar los préstamos realizados. Cada Libro puede estar en múltiples préstamos (1, opcional), y cada Lector puede realizar múltiples préstamos (1, opcional). Sin embargo, cada registro en Préstamo debe referirse obligatoriamente a un único libro y un único lector (1:1, mandatorio). Esto permite rastrear qué libro fue prestado a quién y cuándo.

Dicha tabla fue creada de la siguiente manera:

A computer screen shot of a black screen

Description automatically generated

Ilustración 1: Creación tabla prestamos

A screenshot of a computer

Description automatically generated

Ilustración 2: descripción tabla prestamos

1. **Populación tabla prestamos.**

La tabla Préstamos se crea con el fin de conectar a las entidades Lectores y Libros, permitiendo registrar cada préstamo de manera detallada. En este caso, se han asignado libros a lectores según las reglas de negocio establecidas: 4 lectores con 5 libros cada uno, 3 lectores con 3 libros y 2 lectores con 1 libro. También se garantizó que un libro no fuera prestado y que un lector no tuviera libros asignados ([ID\_LIBRO = 10 Gracias por el fuego], [ID\_LECTOR = 10) (Juan Ignacio, Altarez]), cumpliendo con los requisitos especificados. A continuación, se presentan las consultas correspondientes.

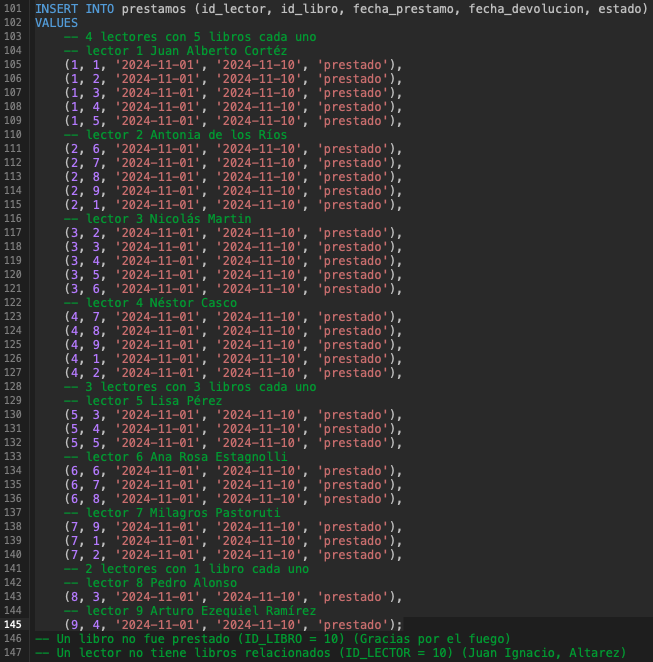


Ilustración 3: Se vuelca la tabla prestamos según las instrucciones mencionadas en la consigna 2.

A screenshot of a computer

Description automatically generated

Ilustración 4: OK en insert

1. **Consultas a tablas para saber qué libro se presta más.**

Para conocer la popularidad de los libros en términos de préstamos, se realizó una consulta que cuenta cuántas veces cada libro ha sido prestado. Se utilizó la función **COUNT()** sobre los registros de la tabla **prestamos**, relacionándolos con los nombres de los libros en la tabla **libros** mediante un **JOIN**. Los alias **loans** y **books** se asignaron a las tablas **prestamos** y **libros**, respectivamente, para simplificar su uso en la consulta. Asimismo, se crearon alias descriptivos para las columnas resultantes, como "**Libro Prestado"** y "**Veces Prestado",** facilitando la comprensión de los datos mostrados. Los resultados se **agrupan por título del libro** y se ordenan en **orden** **descendente** según **la cantidad de préstamos**, mostrando primero los libros más solicitados.

A screenshot of a computer

Description automatically generated

Ilustración 5: Consulta que extrae datos de dos tablas para contar cantidad de libros en préstamo según su ID

1. Préstamos de libros por lector.
2. cantidad de libros que tiene en préstamo cada lector.

Primero, se realiza una consulta **SELECT** que, al igual que en la anterior, utiliza un **JOIN** entre las tablas **lectores** y **prestamos** para relacionarlas. A continuación, se cuentan los libros en préstamo por cada lector, asignando un alias descriptivo a la columna resultante como "**Cantidad de Libros**". Dado que todos los libros inicialmente se encuentran en estado "**prestado**", esta consulta refleja la cantidad total de libros activos por lector:

A screenshot of a computer

Description automatically generated

Ilustración 6: cantidad de libros prestados por usuario.

1. Simulación de devolución.

En este paso, se simula la devolución de un libro por parte de un lector que inicialmente tenía 5 libros en préstamo. Para ello, en lugar de eliminar el registro correspondiente, se actualiza el campo estado de la tabla **Préstamos**, cambiando su valor de "prestado" a "devuelto". Este enfoque permite mantener el historial del préstamo en la base de datos, reflejando correctamente que el libro ya no está en préstamo activo sin perder información sobre la transacción original:

A screen shot of a computer program

Description automatically generated

Ilustración 7: Se actualiza estado del libro 1 para el lector 1

A screenshot of a computer

Description automatically generated

Ilustración 8: diferencia después de la actualización con la misma consulta de la ilustración 6

1. Promedio de edad de los lectores, más viejo y más joven.

Por temas de diseño, PostgreSQL, al realizar una operación de resta entre dos columnas o valores de tipo **DATE**, devuelve un intervalo expresado en días, esto debido a que este motor de bases de datos trata las fechas como el intervalo de días transcurridos desde el **1 de enero de 2000.**

Debido a lo anteriormente mencionado, para hacer cálculos con fechas, por ejemplo obtener la edad de una persona, que por lo general sería restar su la fecha actual, contra la fecha de nacimiento, previamente se deben hacer los siguientes *parseos*:

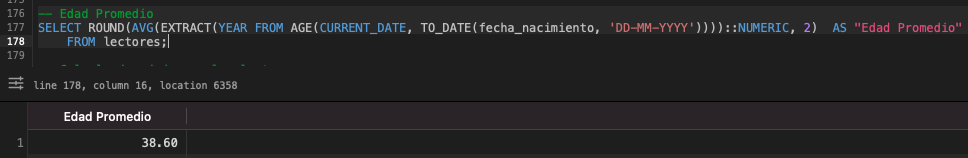
1. Transformar el tipo de la campo fecha\_nacimiento a **DATE** usando el método **TO\_DATE($var, ‘format’):**
2. Luego, utilizar el método **AGE(timestamp, timestamp)** para obtener la edad exacta expresada en días
3. Para finalmente llamar a los métodos **EXTRACT()** y **YEAR FROM ()** para en conjunto extraer exclusivamente los años de diferencia y no los días

|  |
| --- |
| EXTRACT(YEAR FROM AGE(CURRENT\_DATE, TO\_DATE(fecha\_nacimiento, 'DD-MM-YYYY'))) |

Con este preámbulo explicado, podemos continuar con los pasos:

1. Promedio de edad:

Para obtener el promedio del año utilizamos la función AVG**()** que permite obtener el promedio de cierta cantidad de datos en una columna, además para que no aparezca un decimal demasiado largo, se agrega una función round(): **round(number [, decimal\_places)**



1. Lector más viejo.

A screen shot of a computer

Description automatically generated

1. Lector más joven.

A black background with white text

Description automatically generated

1. Vista libros\_prestados.

Para facilitar la consulta de los préstamos activos en la biblioteca, se creó una vista llamada **libros\_prestados**. Esta vista relaciona las tablas **prestamos**, **lectores** y **libros** mediante **JOINs**, y filtra únicamente los registros donde los préstamos están en estado "prestado". La vista muestra el nombre del lector, el título del libro, su editorial y el ISBN, proporcionando una visión clara y útil de los préstamos actuales. Además, se realizaron ajustes en los nombres de las columnas utilizando ALTER VIEW para simplificar su comprensión y presentación.

A screenshot of a computer

Description automatically generated

## Conclusión

La tercera iteración del proyecto de base de datos para la biblioteca de la Universidad Puls-AR permitió afianzar conocimientos clave sobre diseño y manejo de datos relacionales en PostgreSQL. Cada consigna aportó aprendizajes específicos: desde el análisis de las relaciones entre entidades y la creación de vistas, hasta el uso de cálculos avanzados para extraer estadísticas como edades y promedios. Estas tareas no solo consolidaron lo ya implementado, sino que también introdujeron herramientas prácticas para un manejo más robusto del sistema.

Aprendimos la importancia de preservar datos históricos mediante la actualización de estados en lugar de eliminaciones, lo que mejora la trazabilidad y asegura la integridad de la información. Además, trabajar con funciones dinámicas para fechas y cálculos estadísticos reforzó habilidades en la transformación y análisis de datos. Finalmente, la creación de la vista libros\_prestados demostró cómo estructurar consultas complejas de manera profesional, simplificando el acceso a información crítica.

Cada consigna cumplió un rol fundamental en la construcción de un sistema más robusto y adaptable, preparando la base de datos para desafíos futuros y garantizando que el sistema pueda cumplir con requerimientos reales de gestión bibliotecaria.

## Bibliografía

* AGE PSQL: <https://neon.tech/postgresql/postgresql-date-functions/postgresql-age>
* ALTER VIEW: <https://neon.tech/postgresql/postgresql-views/postgresql-alter-view#introduction-to-the-postgresql-alter-view-statement>