Implementación y gestión de una base de datos

SQL Injection y Prepared Statement

{desafío} latam_

Implementar la conexión a una base de datos PostgreSQL en una aplicación Node utilizando buenas prácticas.

Programar instrucciones para la obtención y manipulación de información desde una base de datos utilizando buenas prácticas, acorde al entorno Node.js.

{desafío} latam_

- Unidad 1:
  Implementación y gestión de una base de datos
- Unidad 2:
  Transacciones y API REST
- Unidad 3:
  Trabajo práctico

¿Qué aprenderás en esta sesión?

- *Identifica los riesgos de SQL Injection y las medidas para mitigarlo en una consulta de obtención de datos.*

- *Programa instrucciones de obtención de datos utilizando Prepared Statements para mitigar el riesgo de SQL Injection.*

- *Reconocer el uso del Prepared Statement en PostgreSQL a través del paquete pg para optimizar consultas.*

{desafío} latam_

¿Has oído antes el término Prepared Statement? ¿En qué contexto?

/* Prepared Statements */

Prepared Statements

La base de datos PostgreSQL implementa el concepto de declaraciones preparadas, que hace referencia al manejo de una memoria caché que existirá en la base de datos y estará asociada a una conexión. El manejo realizado por este caché será en forma transparente para la aplicación y se encuentra enfocado en optimizar los tiempos de respuestas al repetir la misma consulta a la base de datos.

Para indicar a la base de datos PostgreSQL que activaremos el caché en la consulta debemos:

- Crear un objeto que permita definir parámetros de configuración a la consulta generada, en este caso crearemos el objeto llamado queryObj, que tendrá un atributo name que corresponde al nombre que recibirá la base de datos, para "cachear" la consulta y el atributo text, que indicará la sentencia SQL que ejecutará el método query, al realizar la consulta a la base de datos.

- Al incluir este parámetro la primera vez que se ejecuta la consulta, se almacenará el plan de ejecución, todos los procesos asociados al análisis y la planificación que requiera la ejecución de la consulta.

Prepared Statements

- La segunda vez que se ejecute con el mismo nombre, la base de datos descarta los procesos asociados y ejecutará directamente la consulta.

Para implementar esta técnica solo debemos incluir una propiedad al JSON como argumento de la consulta. A continuación, te muestro un código de ejemplo en donde se incluye la propiedad name al JSON como argumento:

```
const queryObj = {
     const queryObj = {
     name: 'fetch-user', // prepared statement
     text: `SELECT
               att_id,
               app_clients.cli_name||' '||app_clients.cli_lastname as Cliente,
               att_date,
               att_detail,
               app_employee.emp_name||' '||app_employee.emp_lastname as Ejecutivo
               FROM app_attentions
                   INNER JOIN app_clients
                   ON app_attentions.cli_id = app_clients.cli_id
                   INNER JOIN app_employee
                   ON app_attentions.emp_id = app_employee.emp_id
           `,
   }
}
```

Prepared Statements

Como puedes notar la consulta de ejemplo es enorme y esto es porque la técnica de Prepared Statement es justamente aplicada a consultas de alto cómputo para la base de datos, en una menor escala sería imperceptible notar la diferencia.

¿Qué tanto impacto hace Prepared Statement? A continuación te mostraré la ejecución correspondiente a la consulta de ejemplo previa, en un caso sin prepared statement y en otro sí. El tiempo de la consulta se encontrará al final de la tabla.

Comentemos en palabras simples

{desafío} latam_

/* Prueba sin usar Prepared statement */

Prueba sin usar Prepared statement

La ejecución de la consulta sin utilizar Prepared Statements, como se puede ver en la imagen, se demoró 6.107 milisegundos:

(index)	att_id	cliente	att_date	att_detail	ejecutivo
0	7	'Martin Verdugo'	2020-03-13T03:00:00.000Z	'Sin Servicio'	'Ana Maria Fuentes'
1	6	'Martin Verdugo'	2020-02-08T03:00:00.000Z	'Como puedo cambiarme a un plan con fibra óptica'	'Benjamin Inostroza'
2	5	'Martin Verdugo'	2020-02-04T03:00:00.000Z	'Sin Servicio'	'Benjamin Inostroza'
3	4	'Martin Verdugo'	2020-01-22T03:00:00.000Z	'Como descargar los manuales del telephone'	'Ana Maria Fuentes'
4	3	'Martin Verdugo'	2020-01-12T03:00:00.000Z	'Como pagar mi cuenta'	'Ana Maria Fuentes'
5	2	'Martin Verdugo'	2019-12-24T03:00:00.000Z	'Como cambiar la clave del WIFI'	'Ana Maria Fuentes'
6	1	'Martin Verdugo'	2019-12-24T03:00:00.000Z	'Como cambiar la clave del WIFI'	'Ana Maria Fuentes'
7	10	'Alejandra Morande'	2019-12-22T03:00:00.000Z	'Como puedo cambiarme a un plan con fibra óptica'	'Ana Maria Fuentes'
8	9	'Alejandra Morande'	2019-12-13T03:00:00.000Z	'Sin Servicio'	'Ana Maria Fuentes'
9	8	'Alejandra Morande'	2019-11-03T03:00:00.000Z	'Como solicito traslado de domicilio'	'Ana Maria Fuentes'
10	14	'Cesar Orellana'	2020-02-22T03:00:00.000Z	'Como cambiar la clave del WIFI'	'Ana Maria Fuentes'
11	13	'Cesar Orellana'	2020-02-14T03:00:00.000Z	'Sin Servicio'	'Ana Maria Fuentes'
12	12	'Cesar Orellana'	2020-01-20T03:00:00.000Z	'Que es 4G'	'Ana Maria Fuentes'
13	11	'Cesar Orellana'	2019-12-18T03:00:00.000Z	'Como puedo cambiarme a un plan con fibra óptica'	'Ana Maria Fuentes'
14	15	'Ingrid Rivera'	2020-03-22T03:00:00.000Z	'Como cambiar la clave del WIFI'	'Ana Maria Fuentes'

calculate-time: 6.107ms

/* Prueba utilizando Prepared Statement */

Prueba utilizando Prepared Statement

En esta consulta se utilizó Prepared Statement y devolvió un tiempo de ejecución de 5.059 milisegundos, como se puede ver en la imagen:

(index)	att_id	cliente	att_date	att_detail	ejecutivo
0	7	'Martin Verdugo'	2020-03-13T03:00:00.000Z	'Sin Servicio'	'Ana Maria Fuentes'
1	6	'Martin Verdugo'	2020-02-08T03:00:00.000Z	'Como puedo cambiarme a un plan con fibra óptica'	'Benjamin Inostroza'
2	5	'Martin Verdugo'	2020-02-04T03:00:00.000Z	'Sin Servicio'	'Benjamin Inostroza'
3	4	'Martin Verdugo'	2020-01-22T03:00:00.000Z	'Como descargar los manuales del telephone'	'Ana Maria Fuentes'
4	3	'Martin Verdugo'	2020-01-12T03:00:00.000Z	'Como pagar mi cuenta'	'Ana Maria Fuentes'
5	2	'Martin Verdugo'	2019-12-24T03:00:00.000Z	'Como cambiar la clave del WIFI'	'Ana Maria Fuentes'
6	1	'Martin Verdugo'	2019-12-24T03:00:00.000Z	'Como cambiar la clave del WIFI'	'Ana Maria Fuentes'
7	10	'Alejandra Morande'	2019-12-22T03:00:00.000Z	'Como puedo cambiarme a un plan con fibra óptica'	'Ana Maria Fuentes'
8	9	'Alejandra Morande'	2019-12-13T03:00:00.000Z	'Sin Servicio'	'Ana Maria Fuentes'
9	8	'Alejandra Morande'	2019-11-03T03:00:00.000Z	'Como solicito traslado de domicilio'	'Ana Maria Fuentes'
10	14	'Cesar Orellana'	2020-02-22T03:00:00.000Z	'Como cambiar la clave del WIFI'	'Ana Maria Fuentes'
11	13	'Cesar Orellana'	2020-02-14T03:00:00.000Z	'Sin Servicio'	'Ana Maria Fuentes'
12	12	'Cesar Orellana'	2020-01-20T03:00:00.000Z	'Que es 4G'	'Ana Maria Fuentes'
13	11	'Cesar Orellana'	2019-12-18T03:00:00.000Z	'Como puedo cambiarme a un plan con fibra óptica'	'Ana Maria Fuentes'
14	15	'Ingrid Rivera'	2020-03-22T03:00:00.000Z	'Como cambiar la clave del WIFI'	'Ana Maria Fuentes'

calculate-time: 5.059ms

Prueba utilizando Prepared Statement

Como puedes notar, si existe un cambio, aunque pareciera poco, hay que pensar que esto es escalable a consultas aún más complejas y este 17,16% de mejora puede ser muy importante.

El paquete pg ofrece varias herramientas como esta, entre otra de las populares está el parseo de tipos, el cual consiste en utilizar el interpretador de tipos de PostgreSQL y formatear nuestros datos según su parser. Si quieres saber más sobre esta técnica y otras, en la documentación oficial de pg, creado por el estadounidense Brian Carlson encontrarás todo lo referente a esta librería.

¿Por qué usar Prepared Statement?

Próxima sesión...

- *Utiliza sentencias para la captura y procesamiento de errores de conexión y estados de base de datos utilizando el entorno Node.js*

- *Utiliza sentencias para la captura y manejo de errores durante las operaciones de manipulación acorde al entorno Node.js.*

{desafío} latam_

{desafío}
latam_

Academia de
talentos digitales