TEXT CLASS REVIEW

TEMAS A TRATAR EN EL CUE:

- Modificación de datos en una base de datos.
- Actualización de datos.
- Borrado de datos.

MODIFICACIÓN DE DATOS EN UNA BASE DE DATOS

Inserción de datos

Instrucción INSERT y paso de parámetros.

Para insertar datos en la base de datos, podemos usar el ***SQL INSERT INTO***:

```
INSERT INTO [table_name] ([column1], [column2], [column3], ...)
VALUES ([value1], [value2], [value3], ...);
```

Para concretar esta consulta, insertamos nuestros propios valores en la tabla previamente creada:

```
const query = `
INSERT INTO users (email, firstName, lastName, age)
VALUES ('jose@test.com', 'josé', 'perez', 25)`;
```

Y finalmente, ejecutamos la consulta en la base de datos:

```
cliente.query(query, (err, res) => {
    if (err) {
        console.error(err);
        return;
    }
    console.log('Se ha insertado el registro satisfactoriamente');
    cliente.end();
});
```

Obtener la cantidad de registros insertados.

La recuperación de datos de una base de datos se realiza por medio de la declaración SELECT:

```
SELECT [column1], [column2], ...
FROM [table_name]
WHERE [condition];
```

Para seleccionar las columnas específicas, o todos los campos de una tabla, se utiliza el comodín *; también podemos crear las consultas personalizadas.

En la siguiente consulta, seleccionamos todas las filas y todas las columnas de la tabla ***users*** de la base de datos:

```
const query = `
SELECT *
FROM users
`;
```

Ejecutando esta consulta en la base de datos, esto se realiza por medio de un ***cliente*** previamente instanciado :

```
cliente.query(query, (err, res) => {
    if (err) {
        console.error(err);
        return err;
    }
    for (let row of res.rows) {
        console.log(row);
    }
    cliente.end();
});
```

Recuperación del último id insertado

Para ello, buscamos en la base de datos por orden descendente, y le indicamos que nos recupere (1), el último id insertado.

```
const searchLastRows = async (count) => {
          const query = `
            SELECT * FROM "users"
```

```
            ORDER BY "id" DESC
            LIMIT $1;
    `;
    await cliente.connect();    // creamos la conexion
    try {
                // envió de a  consulta
        const { rows } =  await cliente.query(query, [count]);
        return rows;
    } finally {
        await cliente.end();    // cerrando conexión
    }
};

searchLastRows(1)  // Obtener el ultimo registro insertado
    .then(result => console.table(result))
    .catch(error => console.error(error.stack));
```

ACTUALIZACIÓN DE DATOS

Instrucción UPDATE y paso de parámetros

En la actualización de datos de una tabla de algunos o todos los campos que ya existen, podemos usar la sintaxis o declaración **UPDATE** :

```
UPDATE [table_name]
SET [column1] = [value1], [column2] = [value2], ...
WHERE [condition];
```

Se establecen para cada valor actualizado, de cada columna con la palabra clave **SET**. Después de la cláusula **WHERE**, también se puede definir la condición de qué entradas deben actualizarse. Por ejemplo:

```
const query = `
UPDATE users
SET age = 22
WHERE email="[email protected]"
`;
```

Procedemos a ejecutar la consulta en la base de datos:

```
cliente.query(query, (err, res) => {
    if (err) {
```

```
        console.error(err);
        return;
    }
    if (err) {
        console.error(err);
        return;
    }
    console.log('Se actualizaron los datos satisfactoriamente');
          return.res;
    cliente.end();
});
```

Obtener cantidad de registros actualizados

En la consulta anterior pudimos retornar el objeto result(), y lo mostramos en la pantalla por medio de una tabla o fila. Por ejemplo:

```
// Resultados de las filas en una tabla
console.table(result.rows);

// Resultados de los datos actualizados como filas
console.log(result.rows);
```

BORRADO DE DATOS

La instrucción DELETE y paso de parámetros

En la eliminación de un registro de la tabla se utiliza la declaración DELETE:

```
DELETE FROM [table_name]
WHERE [condition];
```

Un ejemplo es:

```
const query = `
DELETE FROM users
WHERE email="jose@test.com"
`;
cliente.query(query, (err, res) => {
    if (err) {
        console.error(err);
        return;
    }
```

```
    if (err) {
        console.error(err);
        return res;
    }
    console.log('Registro eliminado satisfactoriamente');
    cliente.end();
});
```

Esta consulta o declaración se debe manipular con cuidado, ya que podría eliminar accidentalmente más de lo que se desea.

OBTENER EL NÚMERO DE REGISTROS ELIMINADOS

Para obtener la cantidad de registros eliminados, debemos hacer uso del objeto de resultado `result`, que se devuelve cada vez que se usa el método de consulta en nodejs. A continuación un ejemplo:

```
{
  fieldCount: 0,
  affectedRows: 1,
  insertId: 0,
  serverStatus: 34,
  warningCount: 0,
  message: '',
  protocol41: true,
  changedRows: 0
}
```

Como podemos observar, la segunda propiedad es `affectedRows`, que se traduce en *"filas afectadas"*. Por lo tanto, para obtener el número de filas eliminadas (o en este caso, "afectadas por un delete"), debemos hacer uso del valor de esta propiedad. Una forma sencilla de acceder es la siguiente:

```
console.log(result.affectedRows)
```

Capturando errores en manipulación de datos

Crearemos una función que actualiza un usuario según su id, la realizaremos como una función asíncrona, y capturaremos los errores por medio de un try/catch. Por ejemplo, la función ***updateUser:***

```
const updateUser = async (email, fname, lname, userId) => {
    // creamos la consulta para el query
    const query = `UPDATE "users"
                   SET "name" = $1, "role" = $2
                   WHERE "id" = $3`;
    try {
        // obtenemos la conexión
        await cliente.connect();
        // Enviamos la consulta
        await cliente.query(query, [email, fname, lname, userId]);
        return true;
    } catch (error) { // Capturamos los errores de la consulta
        console.error(error.stack);
        return false;
    } finally {
        await cliente.end();  // cerramos la conexión
    }
};
email, fname, lname, userId
updateUser('jose@test.com', 'José', 'Perez', '1').then(result => {
  // email, fname, lname, userId
    if (result) {
        console.log('Usuario Actualizado');
    }
});
```