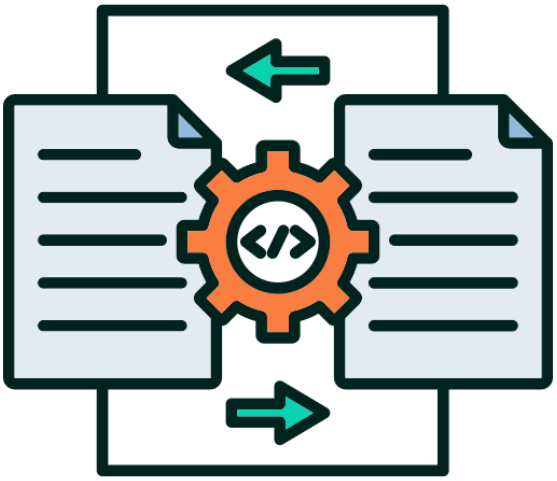

72.39 - Autómatas, Teoría de Lenguajes y Compiladores

Proyecto Especial - Entrega N°1

Diseño e implementación de un lenguaje

Profesores:

Arias Roig, Ana María

Golmar, Mario Agustín

Integrantes del grupo: “JSON2SQL”

64396	Menshikoff, Katia	kmenshikoff@itba.edu.ar
63622	Santamarina Balbin, Manuel José	msantamarinabalbin@itba.edu.ar
63162	Mesa Rubio, Santiago	smesarubio@itba.edu.ar
61067	Sapino, Matías	msapino@itba.edu.ar

Índice

1. Introducción

Este proyecto se centra en el desarrollo de un compilador que convierte consultas JSON a SQL. El objetivo principal es proporcionar una herramienta que permita a los desarrolladores y analistas de datos trabajar de manera eficiente entre ambos mundos, facilitando la interoperabilidad entre bases de datos NoSQL y SQL.

2. Repositorio

El repositorio donde se encontrará la solución está aquí.

3. Propuesta

El proyecto consiste en diseñar e implementar un compilador que transforma estructuras de consultas JSON en sentencias SQL. Este compilador permitirá a los usuarios enviar consultas en formato JSON, que luego se traducirán a SQL para interactuar con bases de datos relacionales.

El compilador contará con las siguientes funcionalidades:

- Análisis léxico y sintáctico: Identificación y validación de las estructuras JSON.
- Generación de AST (Árbol de Sintaxis Abstracta): Representación intermedia de la consulta JSON.
- Transformación a SQL: Traducción del AST a una consulta SQL válida.
- Manejo de errores: Informar al usuario sobre errores de sintaxis y semántica en la consulta JSON.
- Optimización básica: Implementación de optimizaciones simples para generar consultas SQL más eficientes.
- Soporte para distintas operaciones SQL: SELECT, INSERT, UPDATE, DELETE.

4. Construcciones

1. **Validación de sintaxis JSON:** Asegura que la entrada JSON esté correctamente formateada.
2. **Generación básica de consultas SELECT:** Traducción de consultas simples de JSON a SQL.
3. **Soporte para condiciones WHERE:** Adición de cláusulas WHERE en la traducción.
4. **Traducción de consultas INSERT:** Conversión de estructuras JSON para inserciones de datos.
5. **Soporte para JOIN:** Traducción de consultas que involucran la combinación de tablas.
6. **Optimización de consultas:** Implementación de optimizaciones sencillas para consultas SQL generadas.
7. **Manejo avanzado de errores:** Detección y reporte de errores semánticos más complejos.

8. **Traducción de consultas CREATE TABLE:** Conversión de estructuras JSON para la creación de tablas en SQL.

5. Casos de prueba

Casos de aceptación:

1. Consulta SELECT básica sin filtros.
2. Consulta SELECT con condiciones WHERE.
3. Consulta SELECT con múltiples condiciones WHERE (AND/OR).
4. Consulta INSERT con una fila de datos.
5. Consulta INSERT con múltiples filas.
6. Consulta SELECT con JOIN entre dos tablas.
7. Consulta SELECT con función de agregación (COUNT).
8. Consulta DELETE con condición WHERE.
9. Consulta UPDATE con una condición WHERE.
10. Consulta SELECT con ordenamiento (ORDER BY).

Casos de rechazo:

1. JSON malformado (estructura incorrecta).
2. Consulta SELECT sin especificar tablas.
3. Consulta INSERT sin valores.
4. Consulta UPDATE sin condiciones WHERE.
5. Combinación de funciones complejas en SELECT.

6. Ejemplos

a. Consulta JSON para seleccionar todos los empleados de una tabla.

JSON:

```
{
  "select": “all”,
  "from": "employees"
}
```

SQL:

```
SELECT * FROM employees;
```

b. Consulta JSON para seleccionar empleados que trabajan en el departamento de ventas y tienen un salario mayor a 50,000.

JSON:

```
{
  "select": ["name", "salary"],
  "from": "employees",
  "where": {
    "department": "sales",
    "salary": {"gt": 50000}
  }
}
```

SQL:

```
SELECT name, salary FROM employees WHERE department = 'sales' AND
salary > 50000;
```

c. Consulta JSON para crear una tabla de empleados con columnas de nombre, salario, y departamento.

JSON:

```
{
  "create_table": "employees",
  "columns": {
    "name": "VARCHAR(100)",
    "salary": "INT",
    "department": "VARCHAR(50)"
  }
}
```

SQL:

```
CREATE TABLE employees (
  name VARCHAR(100),
  salary INT,
  department VARCHAR(50)
);
```