SQLc: Efficient SQL Query Generation in TypeScript

Leverage SQLc to simplify database access in TypeScript

What is SQLc?

SQLc is a code generation tool that creates type-safe SQL query interfaces for Go and TypeScript by parsing SQL queries.
Automates database access by generating code from SQL.
Reduces the boilerplate and minimizes runtime SQL errors.

Key Features:

- Type safety
- Compile-time verification of queries
- Supports multiple programming languages including Go and TypeScript.

SQLc vs ORM

SQLc:

- **Approach**: Uses raw SQL queries with type-safe code generation.
- **Advantages**:
  - Full control over SQL queries.
  - Better performance since there's no abstraction layer.
  - Easier to optimize complex queries.
  - Compile-time query validation.
- **Drawbacks**:
  - Requires writing SQL manually.

ORM:

- **Approach**: Abstracts database interactions using objects and models.
- **Advantages**:
  - Easier to start for simple CRUD operations.
  - No need to write SQL for basic operations.
  - Handles relationships between tables.
- **Drawbacks**:
  - Can generate inefficient queries, especially for complex operations.
  - Performance overhead due to abstraction.
  - Harder to debug and optimize complex queries.

Conclusion: SQLc is a better choice for performance, query control, and type safety, while ORMs offer convenience at the cost of flexibility and performance.

Benefits of SQLc

Type Safety: Strongly typed, ensuring SQL query correctness.

Productivity: Less time writing repetitive code and writing queries only once.

Performance: Code generated from SQL is highly optimized.

Error Prevention: Catch SQL-related errors at compile-time.

SQLc for TypeScript Developers

SQLc’s support for **TypeScript** allows developers to integrate type-safe SQL query execution in their TypeScript applications.

It provides an efficient and structured way to interact with databases without manually writing query-related code.

How It Works:

- Write SQL queries in `.sql` files.
- SQLc generates corresponding TypeScript code that safely executes those queries.

SQLc Code Generation Workflow

Step 1: Write SQL queries in .sql files.
Step 2: Run SQLc CLI to generate TypeScript code.
Step 3: Use the generated TypeScript code to call the database in a type-safe manner.

Code Example:

// user_queries.sql

```
-- name: getUserByIdQuery :one
SELECT * FROM users WHERE id = $1;
```

// query.ts

```
export const getUserByIdQuery = (client: Client, args: …) => { /* Generated code */ }
```

TypeScript SQLc Example

Scenario: A Node.js API using TypeScript to fetch data from a PostgreSQL database.

SQLc generates the type-safe TypeScript code to execute SQL queries without needing to write raw SQL in the application code.

```
import { getUserByIdQuery } from './generated/query';

const user = await getUserByIdQuery(client, {id: 1});
console.log(user);
```

How to Get Started with SQLc

Step 1: Install SQLc CLI.

```
brew install sqlc
```

Step 2: Create your SQL query files.

Step 3: Generate TypeScript code.

```
sqlc generate
```

Step 4: Use the generated TypeScript code in your application.

Helpful Links: https://docs.sqlc.dev

Conclusion: Why SQLc?

- SQLc simplifies database interaction by generating type-safe SQL query interfaces in TypeScript.
- It enhances developer productivity and reduces SQL-related runtime errors.
- Perfect for developers seeking an efficient and structured way to interact with databases in modern TypeScript applications.

Questions and Discussion

Open the floor for questions about SQLc and TypeScript integration.