wasi-sql

Or wasi-relational or wasi-query (?)

Example golang source

```
func (db *sqlx.DB) GetEmployee(name string) (employee
Employee, err error) {

    return employee, db.Get(&employee, `

        SELECT

            *

        FROM employees

        WHERE name = ?

    `, name)

}
```

Example TS source

```
export const getEmployee = (config: ConnectionConfig) =>
  Query<Employee>(
    config,`
      SELECT
        Id, firstName, lastName
      FROM employees WHERE name = ?
    `);
```

Example as a component

```
(component

  (import "wasi:sql" (instance $db
     // syntax for defining parameter on import unknown
     // so don't judge this next line too closely :)
     (driver "MySQL")

    ...

     (export $query1 (func

          (query string "query:select * from employees where name = ?")

          (param string)

          (result (stream (handle $Employee)))
```

How do we get from high-level language to component?

- … To a SQL enabled Wasm Component?
- For the first case, golang could add support for WASI-SQL since it has a database/sql interface as a part of the SDK.
    - Expectation: golang compiler AOT recognizes the assembly of queries/execs for calls through this interface
    - Go programmer should not need to know about WASI-SQL (only go compiler)
- We will also need a dynamic interface so that the query string could potentially be dynamic. E.g. func Query(string $myQuery)

WIT

- Driver string that defines the syntax. If the syntax is supported by a host component, e.g. a single host could support multiple dialects and translate between them.
- Query and Exec function calls.
- Pros:
  - Type-safe
  - Optimizable
  - AOT
- Cons:
  - Must specify driver for correct parse of SQL dialect
  - Not DB agnostic

Other options

- https://pkg.go.dev/database/sql
    - Define interfaces for Drivers func (db *DB) Driver() driver.Driver
    - and connections, func (c *Conn) Close() error
    - and columns func (ci *ColumnType) Nullable() (nullable, ok bool)
    - and how to exec and query func (tx *Tx) Exec(query string, args ...any) (Result, error)
- Eventual goal: define computational DAGs for relational calc
    - AKA goals of wasi-data
    - This would mean better portability and extensibility
        - Potentially DB agnostic
        - Extension for custom operations or entire distributed algorithms
    - Create compatibility with other projects like substrate
    - This will take more time, careful articulation.