

```
chain_of(
    with_elements(load_postgres_table(("public", "patient"), ["id"], [Int32])),
    flatten())
```



The diagram illustrates the transformation of a SQL query into a dataflow graph. The query is: `SELECT min(id) FROM patient WHERE id = ?`. The graph is divided into a blue-shaded region for the main query execution and a red-shaded region for the `cardinality` operation.

**Query Execution (Blue Region):**

- The query is split into two parts: `load_table("patient",["id"]) SELECT id FROM patient` and `load_table("patient",["min"],["id"]) SELECT min FROM patient WHERE id = ?`.
- The `load_table("patient",["id"]) SELECT id FROM patient` part is executed first, producing a `BlockOf x1to1` and an `EntityShape "patient"`.
- The `load_table("patient",["min"],["id"]) SELECT min FROM patient WHERE id = ?` part is executed next, producing a `BlockOf x0toN` and an `EntityShape "patient"`.
- The results are combined into a `TupleOf` and then a `String`.

**Cardinality Operation (Red Region):**

- The `cardinality(x1to1)` operation is applied to the `BlockOf x1to1` result, producing a `BlockOf x0toN` and an `EntityShape "patient"`.
- The result is then combined into a `TupleOf` and finally a `String`.

**Final Output:**

- The final result is a `String`, which is the output of the query.







```
chain_of(with_elements(load_postgres_table(("public", "patient"), ["id"], [Int32])),
  flatten(),
  with_elements(
    chain_of(
      load_postgres_table(("public", "patient"), ["mrn"], [String], ["id"]),
      block_cardinality(x1to1))),
  flatten(),
  with_elements(
    chain_of(
      output(),
      column(1))))
```



















