

```
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 on the left is: `SELECT min(id) FROM patient WHERE id = ?`. The graph on the right shows the execution flow, with nodes representing data structures and operations. The graph is divided into a blue-shaded region for the main query execution and a red-shaded region for the 'cardinality' operation. The output is a 'String'.

**Query:** `SELECT min(id) FROM patient WHERE id = ?`

**Dataflow Graph:**

- Input:** A variable `id` (green circle) is passed to the `cardinality(x1to1)` node (red box) and the `load_table("patient", ["id"]) SELECT id FROM patient` node (green box).
- Cardinality:** The `cardinality(x1to1)` node outputs a `BlockOf x1to1` (teal oval) and a `BlockOf x0toN` (teal oval).
- Load Table:** The `load_table` node outputs a `BlockOf x0toN` (teal oval) and a `BlockOf x1to1` (teal oval).
- Aggregation:** The `BlockOf x1to1` is flattened (`flatten()`) and then the `min` operation is applied (`min()`), resulting in a `BlockOf x0toN` (teal oval).
- Output:** The final result is a `String` (grey oval).







```
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))))
```



















