

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



The diagram illustrates a computational graph for a neural network architecture, showing the flow of data and operations. The graph is divided into two main sections: a central computational graph and a series of symbolic representations on the right.

**Central Computational Graph:**

- Inputs:** The graph starts with a `head` node (yellow) and a `cardinality(x1to1)` node (green, highlighted with a red border). These feed into a `load_table("patient", ["id"], SELECT id FROM patient)` node (green).
- Intermediate Operations:** The `load_table` node feeds into a `head` node (yellow), which then feeds into a `load_table("patient", ["mn"], ["id"], SELECT mn FROM patient WHERE id = ?)` node (green).
- Output and Aggregation:** The `load_table` node feeds into a `head` node (yellow), which then feeds into a `cardinality(x1to1)` node (green, highlighted with a red border). This node feeds into a `head` node (yellow), which then feeds into a `output()` node (green).
- Final Output:** The `output()` node feeds into a `head` node (yellow), which then feeds into a `column(1)` node (green).

**Symbolic Representations (Right Side):**

- BlockOf:** Multiple instances of `BlockOf` nodes (teal) are shown, representing different parts of the graph. Some are connected to `EntityShape` nodes (teal) and `TupleOf` nodes (teal).
- EntityShape:** These nodes (teal) represent the shape of the data, often connected to `TupleOf` nodes.
- TupleOf:** These nodes (teal) represent the tuple of data, often connected to `Int32` or `String` nodes (gray).
- Int32 and String:** These nodes (gray) represent the final output types of the graph.

The graph shows a complex flow of data and operations, with a large blue shaded region highlighting a specific part of the computation. The symbolic representations on the right provide a detailed view of the data types and shapes involved in the computation.







```
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))))
```



















