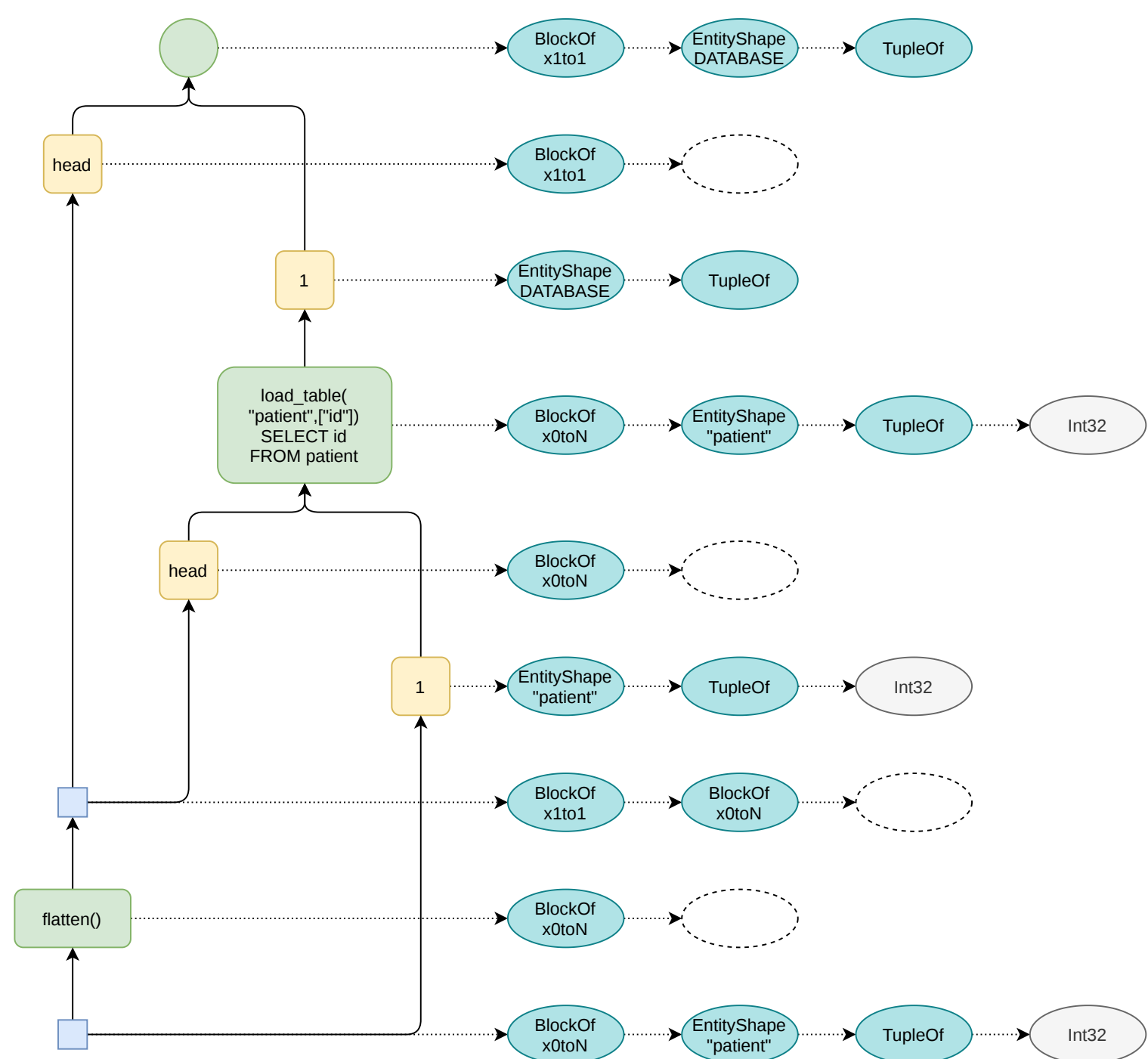


```
chain_of(
  with_elements(load_postgres_table(("public", "patient"), ["id"], [Int32])),
  flatten())
```



The diagram illustrates a complex computational graph for a neural network architecture, likely for a database query execution plan. The graph is composed of several interconnected components, including nodes, edges, and shaded regions.

Key Components and Flow:

- Inputs and Initial Operations:** The graph starts with inputs like `head` and `1`. These feed into operations such as `load_table("patient", ["id"]) SELECT id FROM patient` and `load_table("patient", ["mn"], ["id"]) SELECT mn FROM patient WHERE id = ?`.
- Shaded Regions:**
 - A large **blue shaded region** encompasses the `load_table` operations and the `cardinality(x1to1)` operation.
 - A large **orange shaded region** encompasses the `cardinality(x1to1)` operation and the `output()` operation.
- Intermediate Operations:**
 - `cardinality(x1to1)` and `output()` are key intermediate operations.
 - The graph includes multiple `head` and `1` nodes, which likely represent different parts of the data or specific values.
 - Operations like `column(1)` and `output()` are also present.
- Final Output and Connections:**
 - The graph concludes with a final `output()` operation.
 - Connections to external data sources or databases are shown, such as `EntityShape DATABASE` and `BlockOf x1to1`.
 - The final output is represented by a `TupleOf` node, which is connected to a `String` node.

The graph uses a color-coded system to distinguish between different types of nodes and operations, with green for data loading, blue for cardinality/output, and orange for other operations. The shaded regions highlight specific parts of the graph that are likely of interest for analysis or optimization.







```
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))))
```



