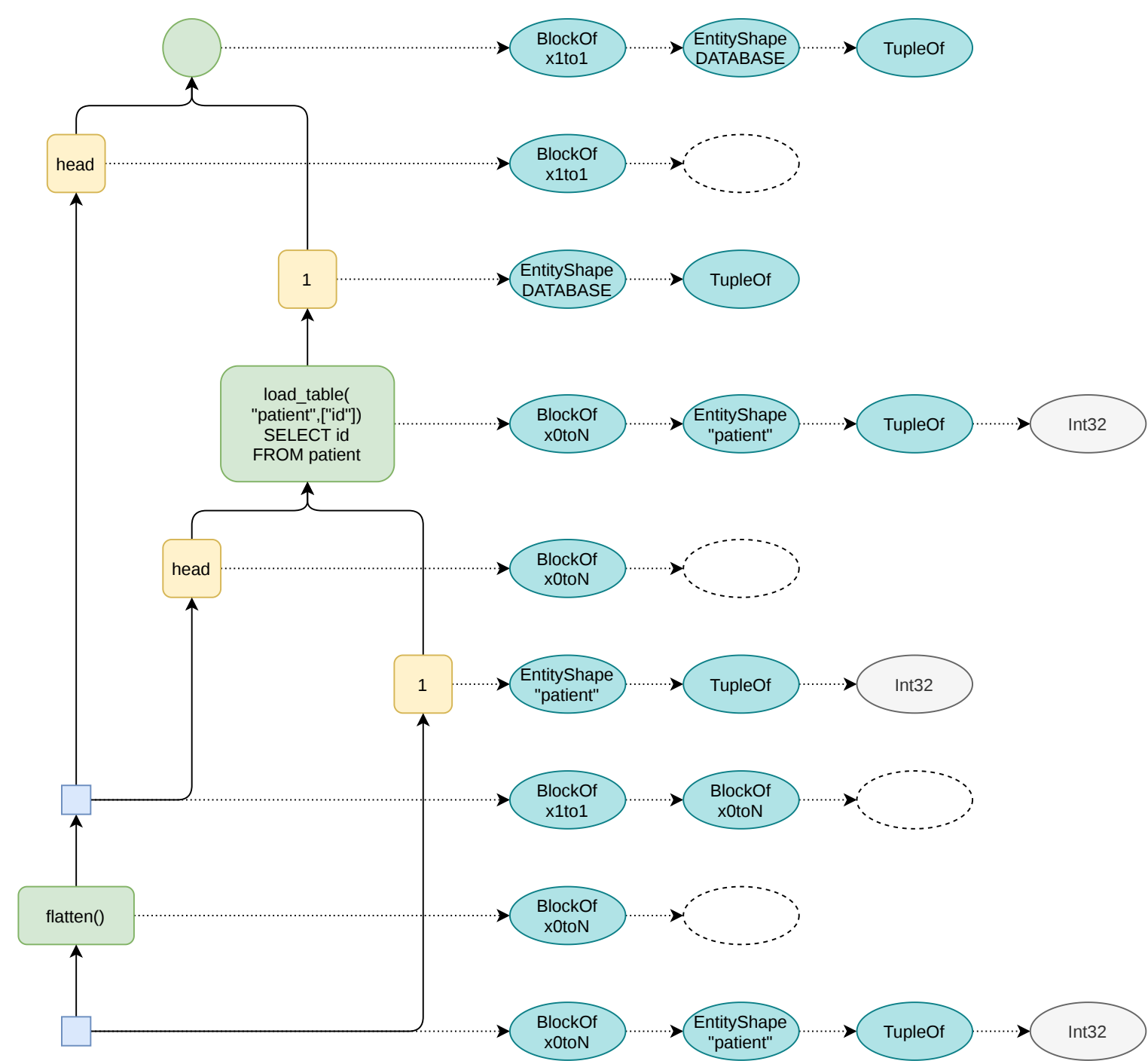


```
chain_of(
  with_elements(load_postgres_table(("public", "patient"), ["id"], [Int32])),
  flatten())
```



[illegible]







The diagram illustrates the mapping of a query plan to a relational algebra expression. The query plan on the left is a directed graph with nodes representing database operations. The relational algebra expression on the right is a corresponding graph using formal constructs. Arrows indicate the mapping between the two.

**Query Plan Nodes (Left):**

- flatten() (green box):** The root of the query plan.
- column(1) (green box):** Connected to the root via a join node (blue circle).
- output() (green box):** Connected to the join node.
- cardinality(x1to1) (green box):** Connected to the join node.
- load\_table("patient", ["min"], ["id"]) SELECT min FROM patient WHERE id = ? (green box):** Connected to the join node.
- load\_table("patient", ["id"]) SELECT id FROM patient (green box):** Connected to the join node.
- head (yellow box):** Multiple instances connected to various nodes.
- 1 (yellow box):** Multiple instances connected to various nodes.
- head (yellow box):** Connected to the root.

**Relational Algebra Expression Nodes (Right):**

- BlockOf x1to1, EntityShape DATABASE, TupleOf:** Connected to the root.
- BlockOf x0toN, EntityShape "patient", TupleOf:** Connected to the root.
- BlockOf x0toN:** Multiple instances connected to various nodes.
- EntityShape "patient", TupleOf:** Multiple instances connected to various nodes.
- Int32, String:** Leaf nodes representing data types.

The diagram shows a complex mapping where the query plan's operations are translated into relational algebra constructs, maintaining the same logical flow and data dependencies.

















