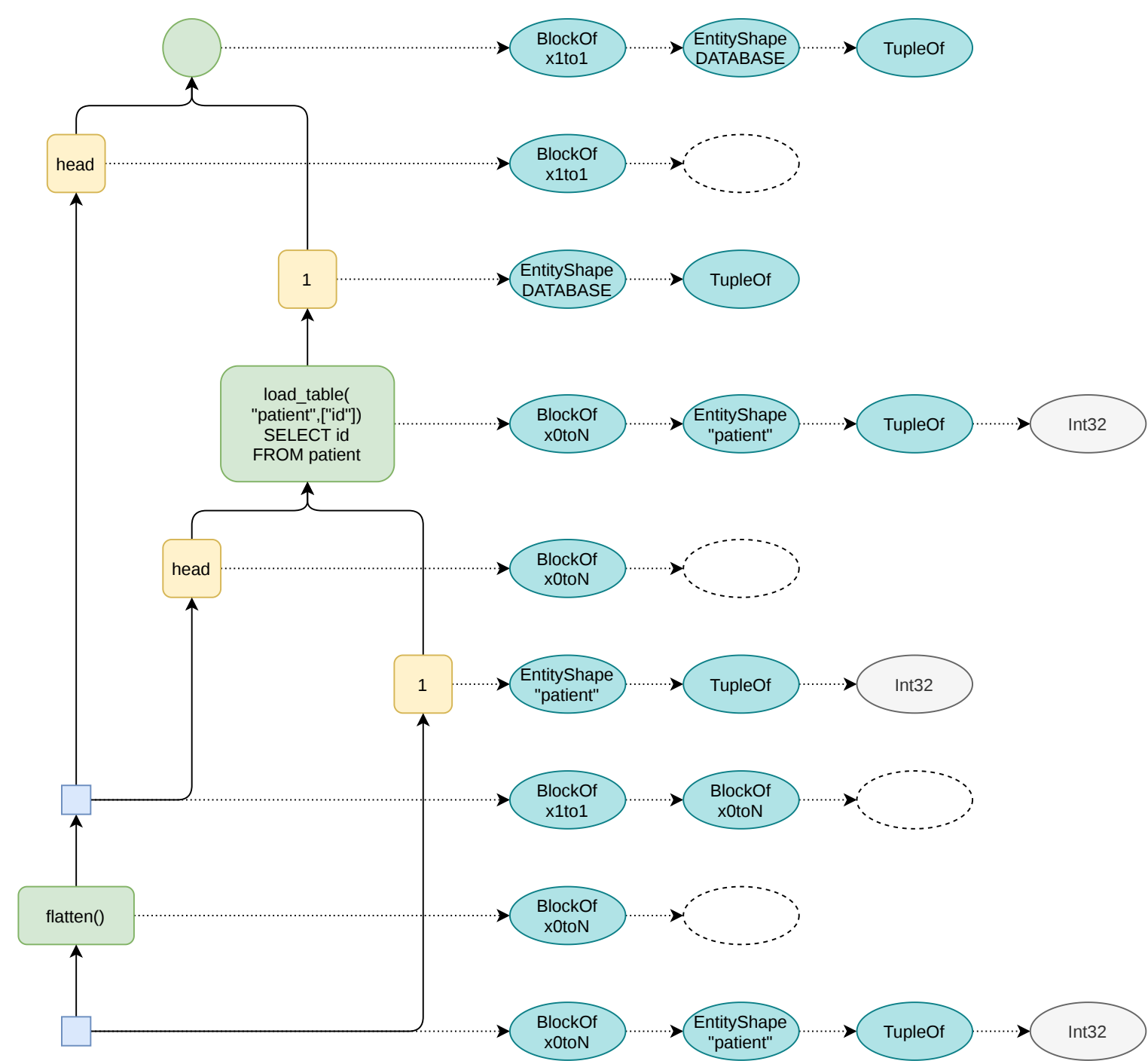


```
chain_of(
  with_elements(load_postgres_table(("public", "patient"), ["id"], [Int32])),
  flatten())
```



The diagram illustrates a complex computational graph, likely representing a neural network architecture or a data processing pipeline. The graph is composed of several interconnected nodes and edges, with a large blue shaded region and a large orange shaded region.

Key Components and Operations:

- Input/Output Nodes:** The graph starts with an input node (green circle) and ends with an output node (green circle). There are also intermediate nodes like "head" (yellow rectangle), "1" (yellow rectangle), "output()" (green rectangle), "column(1)" (green rectangle), "cardinality(x1to1)" (green rectangle), "load_table(...)" (green rectangle), and "atten()" (green rectangle).
- Shaded Regions:**
 - Blue Shaded Region:** Contains the "load_table(...)" node and the "cardinality(x1to1)" node.
 - Orange Shaded Region:** Contains the "head" node and the "1" node.
- Operations and Data Flow:**
 - load_table(...):** A node that takes "patient" as input and outputs "id". It is connected to "head" and "1".
 - cardinality(x1to1):** A node that takes "x1to1" as input and outputs "1". It is connected to "head" and "1".
 - output():** A node that takes "head" and "1" as inputs and outputs "output()".
 - column(1):** A node that takes "output()" as input and outputs "column(1)".
 - head:** A node that takes "load_table(...)" and "cardinality(x1to1)" as inputs and outputs "head".
 - 1:** A node that takes "load_table(...)" and "cardinality(x1to1)" as inputs and outputs "1".
 - atten():** A node that takes "head" and "1" as inputs and outputs "atten()".
- Right Side Detail:** The right side of the image shows a detailed view of the operations, with nodes like "BlockOf", "EntityShape", "TupleOf", and "Int32", connected by dotted lines. This section provides a more granular view of the data flow and operations within the graph.







```
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))))
```















