

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



The diagram illustrates the execution of a query plan for the query: `load_table('patient',['id']) SELECT id FROM patient`. The plan is shown as a sequence of operators (green boxes) and their outputs (blue ovals).

Query Plan:

- load_table('patient',['id']) SELECT id FROM patient** (Green box)
 - Input: `head` (Yellow box)
 - Output: `BlockOf x1to1` (Blue oval) → `EntityShape DATABASE` (Blue oval) → `TupleOf` (Blue oval)
- load_table('patient',['min'],['id']) SELECT min FROM patient WHERE id = ?** (Green box)
 - Input: `head` (Yellow box)
 - Output: `BlockOf x0toN` (Blue oval) → `EntityShape 'patient'` (Blue oval) → `TupleOf` (Blue oval) → `Int32` (Grey oval)
- cardinality(x1to1)** (Green box)
 - Input: `head` (Yellow box)
 - Output: `BlockOf x0toN` (Blue oval) → `EntityShape 'patient'` (Blue oval) → `TupleOf` (Blue oval) → `String` (Grey oval)
- output()** (Green box)
 - Input: `head` (Yellow box)
 - Output: `BlockOf x1to1` (Blue oval) → `BlockOf x0toN` (Blue oval) → `TupleOf` (Blue oval) → `String` (Grey oval)
- column(1)** (Green box)
 - Input: `head` (Yellow box)
 - Output: `BlockOf x0toN` (Blue oval) → `BlockOf x1to1` (Blue oval) → `TupleOf` (Blue oval) → `String` (Grey oval)
- flatten()** (Green box)
 - Input: `head` (Yellow box)
 - Output: `BlockOf x0toN` (Blue oval) → `BlockOf x0toN` (Blue oval) → `TupleOf` (Blue oval) → `String` (Grey oval)
- flatten()** (Green box)
 - Input: `head` (Yellow box)
 - Output: `BlockOf x0toN` (Blue oval) → `BlockOf x0toN` (Blue oval) → `TupleOf` (Blue oval) → `String` (Grey oval)

The diagram also shows the flow of data through various operators like `head`, `flatten()`, `output()`, `column()`, and `cardinality()`. The outputs are shown as `BlockOf` and `EntityShape` objects, which are then converted to `TupleOf` and finally to the final data type (e.g., `Int32`, `String`).

