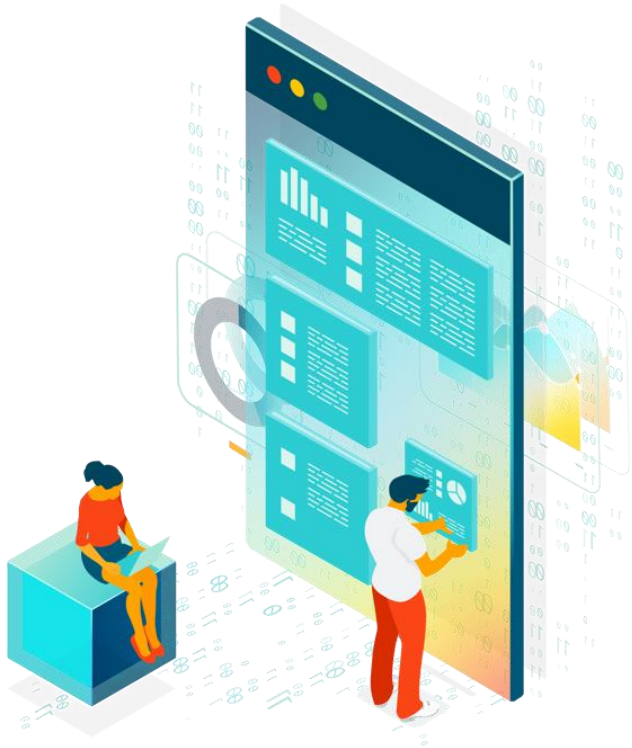
ArangoDB

for

Altair IoT Studio



ALTAIR

ArangoDB: Definition

ArangoDB is an open-source, multi-model database, capable of storing data in document, graph and key/value formats.

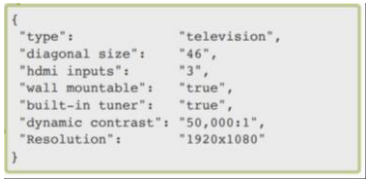
Every piece of data in ArangoDB is a JSON document with an identifier.

Reference: ArangoDB

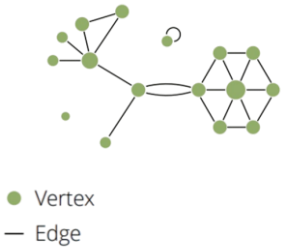
ArangoDB: Database basic concepts

- **Collections**: Collections are containers for storing documents in ArangoDB. They act as tables in traditional relational databases but are schema-less, allowing flexibility in storing various types of documents within the same collection.

- **Documents**: Documents are individual records stored within collections. They are represented in JSON format. Each document can have its own structure.



- **Graphs**: Data structure for representing relationships between entities. They consist of nodes (vertices) and edges (relationships), allowing for the modeling of complex interconnected data. Graphs enable efficient traversal and querying of interconnected data, making them ideal for applications involving networks, social connections, and recommendation systems.



Reference: ArangoDB

ALTAIR

ArangoDB: System attributes

All documents contain special attributes at the top-level that start with an underscore:

- The **document key** is stored as a string in the **`_key`** attribute.
  - Example: `01HW5M4G8704HJTJWR4X3TBXRD`
- The **document identifier** is stored as a string in the **`_id`** attribute.
  - Example: `things/01HW5M4G8704HJTJWR4X3TBXRD`
- The **document revision** is stored as a string in the **`_rev`** attribute.

Edge documents in edge collections have two additional system attributes:

- The document identifier of the source vertex stored in the **`_from`** attribute.
  - Example: `things/01HW5M4G8704HJTJWR4X3TBXRD`
- The document identifier of the target vertex stored in the **`_to`** attribute.
  - Example: `things/01HW5MJFV8ER817FSJ0XP9VYA7`

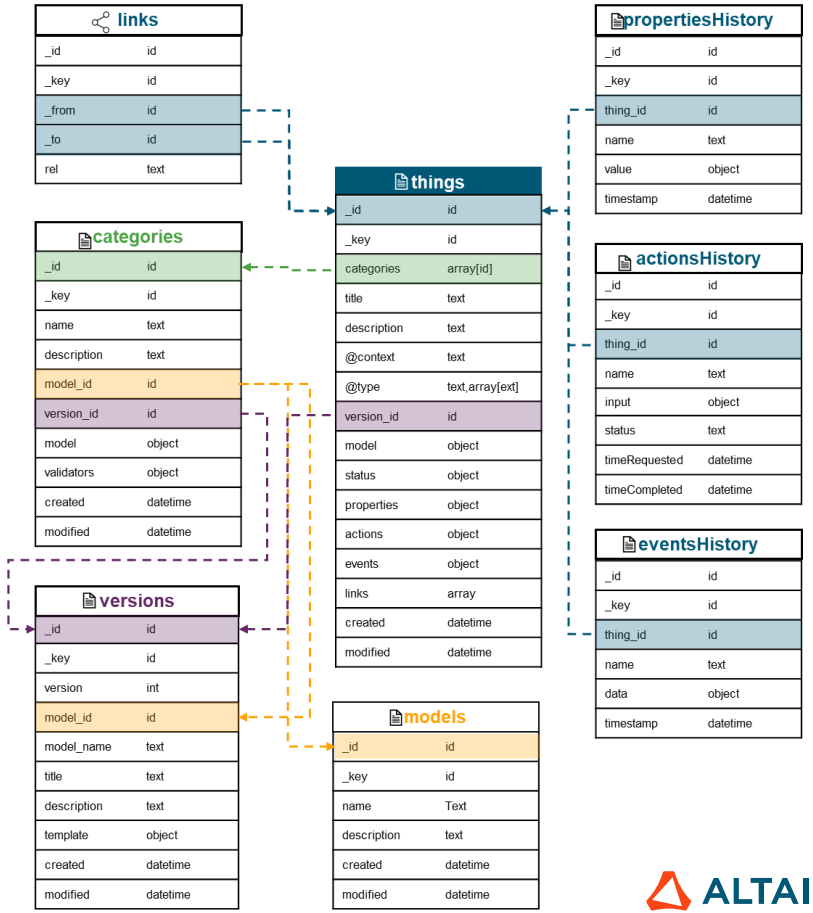
Reference: ArangoDB

ALTAIR

AnythingDB: Database schema

These are the ArangoDB collections used by AnythingDB to store all the documents (Things, Categories, history values...).

The fields of each collection show the attributes used in each of its documents.



Reference: ArangoDB

ALTAIR

Understanding AQL (ArangoDB Query Language)

- It’s mainly a **declarative language**: a query expresses what result should be achieved but not how it should be achieved.
- It aims to be **human-readable**.
- It's designed to handle complex query patterns and to support ArangoDB's various data models.
- It supports reading and modifying collection data, but **it doesn’t support data-definition operations** such as creating and dropping databases, collections and indexes.
- It is a **pure data manipulation language (DML)**, not a data definition language (DDL) or a data control language (DCL).

Reference: ArangoDB

AQL: Main AQL Operations

- **FOR**: Iterates over a collection, the elements of an array, or traverse a graph.

Syntax:
```
FOR variableName IN expression
```

Example:
```
FOR t IN things
```

- **FILTER**: Restricts the results to elements that match arbitrary logical conditions.

Syntax:
```
FILTER expression
```

Example:
```
FOR t IN things
    FILTER t.status.temp > 20
    FILTER 'switch' IN t.`@type`
```

- **RETURN**: Produces the result of a query. It can determine the structure and content of the output returned.

Syntax:
```
RETURN expression
```

Example:
```
FOR t IN things
    RETURN t
```

Reference: ArangoDB

AQL: Main AQL Operations

- **SORT**: Specifies one or multiple sort criteria and directions to control the order of query results or the elements of arrays.

Syntax:

```
SORT expression direction
```

Example:

```
FOR t IN things
    SORT t.name, t.created DESC
    RETURN t
```

- **LIMIT**: Reduces the number of results to at most the specified number and optionally skip results using an offset for pagination

Syntax:

```
LIMIT count
LIMIT offset, count
```

Example:

```
FOR t in things
    LIMIT 50
    RETURN t
```

Reference: ArangoDB

AQL: Main AQL Operations

- **LET**: Assigns an arbitrary value to a variable.

Syntax:

```
LET variableName = expression
```

Example:

```
LET categoryName = 'motionSensors'

LET category = FIRST(
    FOR c IN categories
        FILTER c.name == categoryName
        RETURN c
)

FOR t IN things
    FILTER category._id IN t.categories
    RETURN t
```

Reference: ArangoDB

AQL: Some Examples

- **Example 1:** Get a maximum of 10 Things based on their status values and type.

```
FOR t IN things
    SORT t.created
    FILTER 'weather_station' IN t.`@type`
    FILTER t.status.temp > 20 AND t.status.humidity < 0.5
    LIMIT 10
    RETURN {
        id: t.uid,
        name: t.name,
        status: t.status
    }
```

Reference: ArangoDB

AQL: Some Examples

- **Example 2:** Gets all the Properties history values of all the Things belonging to a given Category.

```
FOR c IN categories
    FILTER c.name == 'Vehicles'

    FOR p IN propertiesHistory
        SORT p.timestamp DESC
        FILTER c._id IN DOCUMENT(p.thing_id).categories
        RETURN p
```

Reference: ArangoDB

References

- ArangoDB documentation: https://docs.arangodb.com/
- ArangoDB Graph Course: https://university.arangodb.com/courses/graph-course-for-beginners/

Reference: ArangoDB