Colecciones de documentos

Tiempo estimado: 25min

Ya sabemos cómo crear bases de datos que, recordemos, no es más que un almacén de datos que tiene asignado un nombre identificativo. Ahora, vamos a ver cómo crear colecciones, uno de los tipos de objetos que se puede crear en una base de datos.

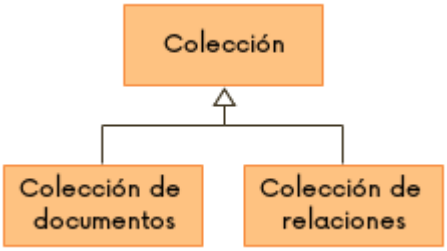
La lección comienza introduciendo el concepto de colección, ampliamente utilizado dentro de los sistemas de gestión de bases de datos de documentos. A continuación, se describe qué es un documento y el formato JSON. Y finalmente, se muestra cómo crear, listar, truncar y suprimir colecciones.

Al finalizar la lección, el estudiante sabrá:

- Qué es una colección.
- Qué tipos de colecciones hay en ArangoDB.
- Qué es un documento.
- Cómo crear, truncar y suprimir colecciones de documentos en ArangoDB.
- Qué es un objeto colección.
- Cómo obtener un objeto colección.
- Cómo conocer las colecciones de una base de datos.

Introducción

Una colección (*collection*) es un objeto contenedor donde almacenar datos. En ArangoDB, se distingue dos tipos de colecciones: las de documentos y las de relaciones.



Una colección de documentos (*document collection*) es aquella que almacena documentos. Mientras que una colección de relaciones (*edge collection*) almacena relaciones entre documentos. Recordemos que ArangoDB es un multimodelo. Las colecciones de documentos dan soporte a los almacenes clave-valor y a los de documentos. Las colecciones de relaciones a los almacenes de grafos. En este curso, nos vamos a centrar sólo en las colecciones de documentos.

Colecciones de documentos

Una colección de documentos (*document collection*) no es más que un contenedor donde almacenar documentos, donde un documento (*document*) es un objeto que representa un registro de datos. Cada documento está formado por uno o más campos, donde cada campo (*field*) representa una propiedad o atributo y su valor asociado.

Las colecciones de documentos son equivalentes al concepto de tabla de las bases de datos SQL. Pero a diferencia de las tablas que definen un esquema que todas sus filas deben cumplir, en las colecciones

de documentos, esto no es así. Las colecciones no presentan ningún esquema y, por lo tanto, los documentos, el concepto análogo a las filas en SQL, pueden tener distintos esquemas. Aunque por lo general tendrán un esquema idéntico o, al menos, parecido. Este modelo lo usan muchos productos de bases de datos de documentos como, por ejemplo, ArangoDB, CouchDB, MongoDB y RethinkDB.

Documentos

Un documento (*document*) es un objeto de datos, similar al concepto de fila en las bases de datos SQL. Una colección puede almacenar tantos documentos como sea necesario. Los documentos dividen sus datos en campos (*fields*), cada uno de las cuales contiene el valor de una determinada propiedad o atributo. En ArangoDB, cada documento es un objeto JSON.

JSON

Un formato de datos (*data format*) no es más que una manera de representar un objeto o registro de datos como, por ejemplo, una instancia de una clase C++, C#, Java, JavaScript, Python o, en nuestro caso, un documento de ArangoDB.

JSON (*JavaScript Object Notation*, Notación de Objetos de JavaScript), pronunciado *yeison*, es un formato para representar estructuras de datos mediante texto. Otro formato muy conocido es XML, pero éste lo hace mediante el uso de etiquetas. Ambos son formatos de representación de datos, pero cada uno de ellos lo hace de una manera distinta. Vamos a ilustrarlo mediante un ejemplo:

```
//JSON
{ "nombre": "The National",
  "añoFormación": 1999,
  "origen": "Cincinnati, OH; Brooklyn, NY",
  "géneros": ["Indie", Post-punk"],
  "sitioWeb": "americanmary.com",
  "miembros": [
    { "nombre": "Matt Berninger", "rol": "vocalista" },
    { "nombre": "Aaron Dessner", "rol": "guitarista, teclista" },
    { "nombre": "Bryce Dessner", "rol": "guitarrista, teclista" },
    { "nombre": "Bryan Devendorf", "rol": "batería" },
    { "nombre": "Scott Devendorf", "rol": "bajista" }
  ]
}

<!-- XML -->
<artista>
  <nombre>The National</nombre>
  <añoFormación>1999</añoFormación>
  <origen>Cincinnati, OH; Brooklyn, NY</origen>
  <géneros>
    <género>Indie</género>
    <género>Post-punk</género>
  </géneros>
  <sitioWeb>americanmary.com</sitioWeb>
  <miembros>
    <miembro>
      <nombre>Matt Berninger</nombre>
      <rol>vocalista</rol>
    </miembro>
    <miembro>
      <nombre>Aaron Dessner</nombre>
      <rol>guitarrista, teclista</rol>
    </miembro>
    <miembro>
      <nombre>Bryce Dessner</nombre>
      <rol>guitarrista, teclista</rol>
    </miembro>
    <miembro>
      <nombre>Bryan Devendorf</nombre>
      <rol>batería</rol>
    </miembro>
    <miembro>
      <nombre>Scott Devendorf</nombre>
```



```
        <rol>bajista</rol>
      </miembro>
    </miembros>
  </artista>
```

Como puede observar, la misma información se puede representar en formatos distintos, según las preferencias de cada uno. En nuestro caso, ArangoDB, los documentos se representan mediante objetos en formato JSON.

Un objeto JSON (*JSON object*) es un objeto formado por un conjunto ordenado de campos, en forma de pares clave-valor. Cada campo (*field*), como acabamos de ver, es un par clave-valor, donde la clave (*key*) identifica el campo y es siempre una cadena de texto y su valor (*value*) puede ser otro objeto JSON, una cadena de texto, un número, un *array*, un valor booleano o un valor nulo.

Su sintaxis es muy parecida a la de los objetos de JavaScript, concretamente es como sigue:

```
ObjetoJSON := { Miembro, Miembro... }
Miembro := Cadena : Valor
Valor := Array | Booleano | Cadena | Nulo | Número | ObjetoJSON
```

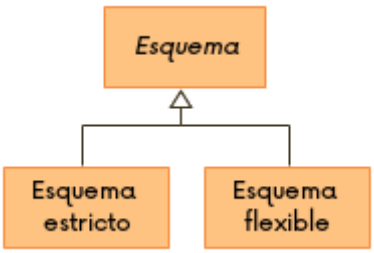
Tal como puede observar del ejemplo anterior y de la sintaxis ahora definida, JSON define un formato de representación claro y de fácil comprensión debido a su sintaxis textual. Requiere poca codificación y, por ende, poco procesamiento por parte del software. Se puede usar, pues, tanto para almacenar objetos de datos en disco como para serializarlos y transmitirlos a través de la red.

JSON se encuentra normalizado y su especificación se encuentra en json.org. Es independiente de JavaScript. Y se puede utilizar con otros lenguajes de programación como, por ejemplo, Java, C#, C++, etc.

Esquema de documento

Un esquema (*schema*) define el conjunto de campos que debe presentar un documento. Recordemos que los documentos se almacenan dentro de colecciones, las cuales representan contenedores donde almacenar documentos.

Conceptualmente, en bases de datos, independientemente de si son SQL o NoSQL, los esquemas se clasifican principalmente en estrictos y flexibles.



Un esquema estricto (*strict schema*), también conocido como esquema estático (*static schema*), es aquel que define exactamente qué campos debe presentar cada documento de la colección, así como los tipos de los valores. Todo documento debe cumplir necesariamente el esquema de su colección. No se puede añadir un documento a una colección si no cumple con el esquema definido en la colección en la que se almacena. Es el tipo de esquema que predomina en motores SQL y algunos NoSQL como, por ejemplo, Cassandra, PostgreSQL, SQL Server y SQLite.

En cambio, un esquema flexible (*flexible schema*), también conocido como esquema dinámico (*dynamic schema*) o sin esquema (*schemaless*), es más flexible y no define ningún tipo de restricción en los campos que debe presentar el documento. Esto quiere decir que en una colección se podrá almacenar documentos con esquemas distintos y tipos de datos distintos para campos homónimos. Aunque generalmente, por convenio, se suele seguir un esquema muy similar. Se utiliza principalmente en sistemas de gestión de bases de datos NoSQL como, por ejemplo, ArangoDB, CouchDB, MongoDB y RethinkDB. Lo que favorece el desarrollo rápido de aplicaciones.

Identificador de documento

La integridad de datos (*data integrity*) es la cualidad que debe presentar la base de datos para asegurar que describe con exactitud la realidad que almacena o describe. Para poder ayudar al motor

de bases de datos a mantener la integridad, hay que definir restricciones de integridad (*integrity constraints*), el mecanismo de control mediante el cual se limita el conjunto de valores que se puede almacenar o introducir en la base de datos. Consiguiendo así que sólo se almacenen datos correctos.

Cada vez que se introduce, modifica o elimina un documento de una colección, que tiene asociada restricciones de integridad, el motor de base de datos verifica que las restricciones se cumplen. Si es así, acepta la operación; en otro caso, se produce lo que se conoce formalmente como violación de integridad (*integrity violation*) o fallo de integridad (*integrity failure*), cancelándose la operación.

En ArangoDB, así como en muchos sistemas de gestión de bases de datos NoSQL, se utiliza la restricción de clave principal (*primary key constraint*), la cual garantiza que un documento será único en la colección. Se conoce como clave (*key*) a aquel campo que identifica un documento como único.

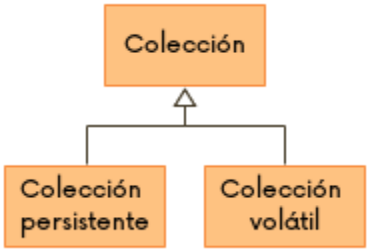
En ArangoDB, se distingue dos tipos de claves, la clave principal y el identificador, ambas simples. Una clave simple (*simple key*) es aquella que está formada por un campo; mientras que una clave compuesta (*comsposite key*), por varios.

Por un lado, tenemos la clave principal (*primary key*), el campo _key del documento, que contiene un valor único para todo documento de la colección. Nunca dos documentos almacenados en la misma colección podrán tener el mismo valor en su clave primaria. Por otro lado, se utiliza el identificador (*identifier* o *handle*), el campo _id, el cual contiene un valor único para *toda* la base de datos. Mientras que la clave principal puede fijarla el usuario, el identificador lo fija automáticamente ArangoDB, mediante el nombre de la colección y la clave del documento, siguiendo el formato colección/clave.

Una vez insertado un documento en una colección, no se puede cambiar su clave principal y, por ende, tampoco su identificador.

Colecciones de documentos

Tal como hemos visto anteriormente, una colección de documentos (*document collection*) es aquella que almacena documentos. Atendiendo a dónde se almacenan los documentos, se distingue entre colecciones persistentes y volátiles.



Una colección persistente (*persistent collection*) es aquella que almacena sus documentos en disco. En cambio, una colección volátil (*volatile collection*) lo hace en memoria, de tal manera que cuando la instancia se detenga, formal o informalmente, su contenido se perderá.

El acceso a una colección volátil es más rápido que sobre una colección persistente. Pero tiene el inconveniente que sus datos se pierden cuando la instancia se detiene.

ArangoDB soporta ambos tipos de colecciones. Se crea un tipo u otro atendiendo a la propiedad isVolatile de la colección.

Creación de colecciones de documentos

La creación de una colección (*collection creation*) es la operación mediante la cual se crea una nueva colección en la base de datos. Se puede utilizar el *shell* o la interfaz web de ArangoDB.

Permisos

Para crear una colección, el usuario debe tener concedido el permiso de lectura/escritura en la base de datos donde se creará la colección.

Creación de colecciones de documentos mediante arangosh

Para crear una colección de documentos mediante el *shell* de ArangoDB, hay que utilizar el método _create() del objeto predefinido db:

```
_create(name)
_create(name, props)
```

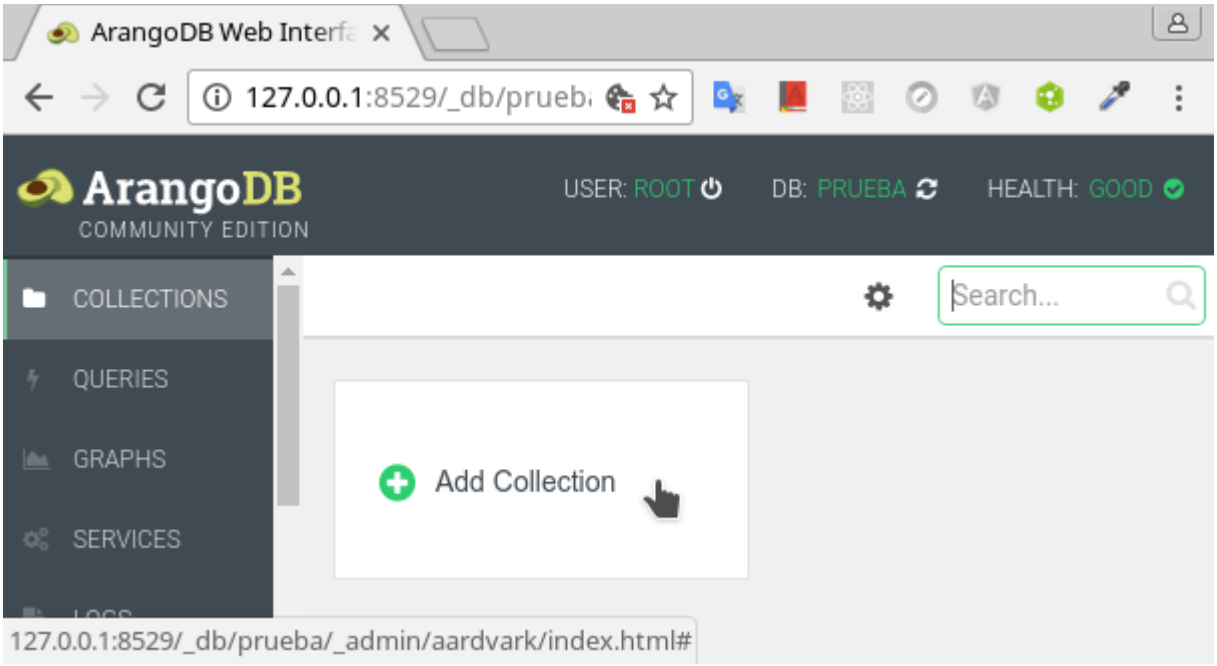
Parámetro	Tipo de datos	Descripción
name	string	Nombre de la colección de documentos a crear.
props	object	Propiedades de la colección. Las más frecuentes son: • waitForSync (boolean). ¿Colección síncrona? true, sí; false, no, asíncrona. Valor predeterminado: false. • journalSize (number). Tamaño del archivo de *log* de la colección. Valor predeterminado: el indicado en la configuración de la instancia. • isVolatile (boolean). ¿Es una colección volátil? true, sí; false, no, es persistente. Valor predeterminado: false.

Ejemplo ilustrativo:

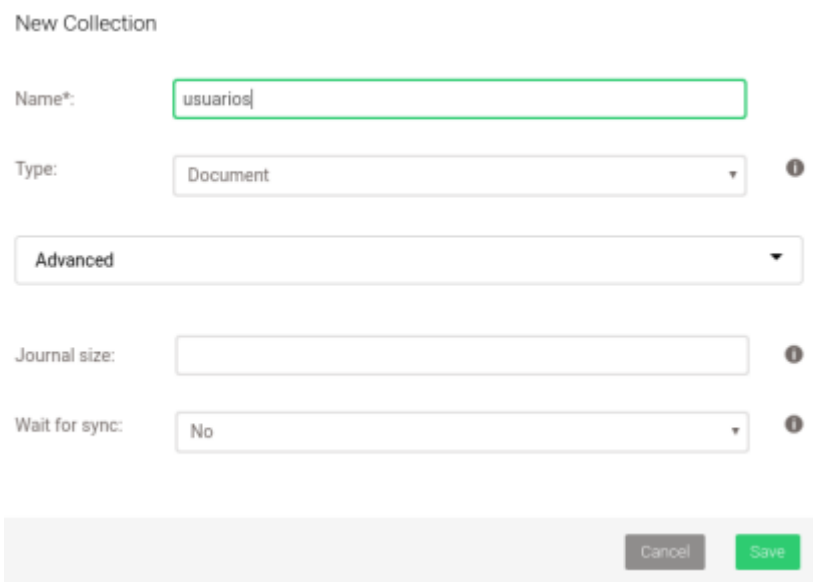
```
127.0.0.1:8529@prueba> db._create("usuarios", {waitForSync: true})
[ArangoCollection 213703, "usuarios" (type document, status loaded)]
127.0.0.1:8529@prueba>
```

Creación de colecciones de documentos mediante interfaz web

También es posible crear una colección mediante la interfaz web, para ello, hay que ir a COLLECTIONS y hacer clic en + Add Collection:



A continuación, rellenar el formulario, seleccionando Document como Type, y hacer clic en Save:



Para mostrar las opciones, hacer clic en Advanced.

Listado de colecciones

Para conocer qué colecciones tiene una determinada base de datos, podemos utilizar tanto el *shell* como la interfaz web de ArangoDB.

Permisos

Para poder listar las colecciones de una base de datos, el usuario debe tener concedido el permiso de lectura/escritura en la base de datos.

Listado de colecciones mediante arangosh

Para obtener la lista de colecciones mediante el *shell*, hay que utilizar el método _collections() del objeto db:

```
_collections() : ArangoCollection[]
```

Listado de colecciones mediante interfaz web

Para conocer las colecciones de una base de datos mediante la interfaz web, ir a COLLECTIONS.

Objeto colección

Un objeto colección (*collection object*) es una instancia de la clase ArangoCollection que representa una determinada colección. El método db._collections() devuelve un *array* de instancias de ArangoCollection, una para cada una de las colecciones. Si deseamos acceder al objeto de una determinada colección, mediante arangosh, no hay más que indicar el nombre de la colección como propiedad del objeto db. Veamos un ejemplo ilustrativo:

```
127.0.0.1:8529@prueba> db.usuarios
[ArangoCollection 214309, "usuarios" (type document, status loaded)]
127.0.0.1:8529@prueba>
```

Truncamiento de colección

El truncamiento de colección (*collection truncate*) es la operación mediante la cual se vacía o suprime el contenido de una colección, manteniendo la colección y sus índices, no así sus documentos. Se puede hacer mediante el *shell* y la interfaz web de ArangoDB.

Permisos

Para truncar una colección, el usuario debe tener concedido el permiso de lectura/escritura sobre la base de datos donde se encuentra la colección.

Truncamiento de colección mediante arangosh

Para realizar un truncamiento mediante el *shell*, se puede usar el método truncate() del objeto colección o bien el método _truncate() del objeto db:

```
db.colección.truncate()
db._truncate(name)
```

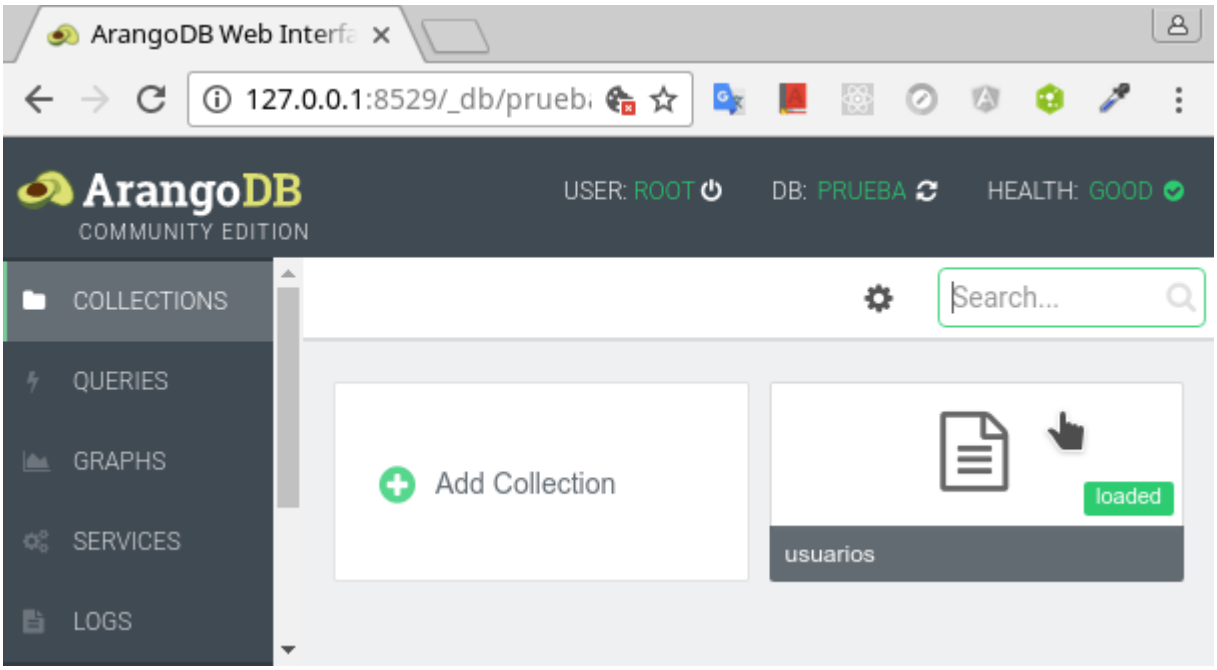
Parámetro	Tipo de datos	Descripción
name	string	Nombre de la colección a truncar.

He aquí unos ejemplos ilustrativos:

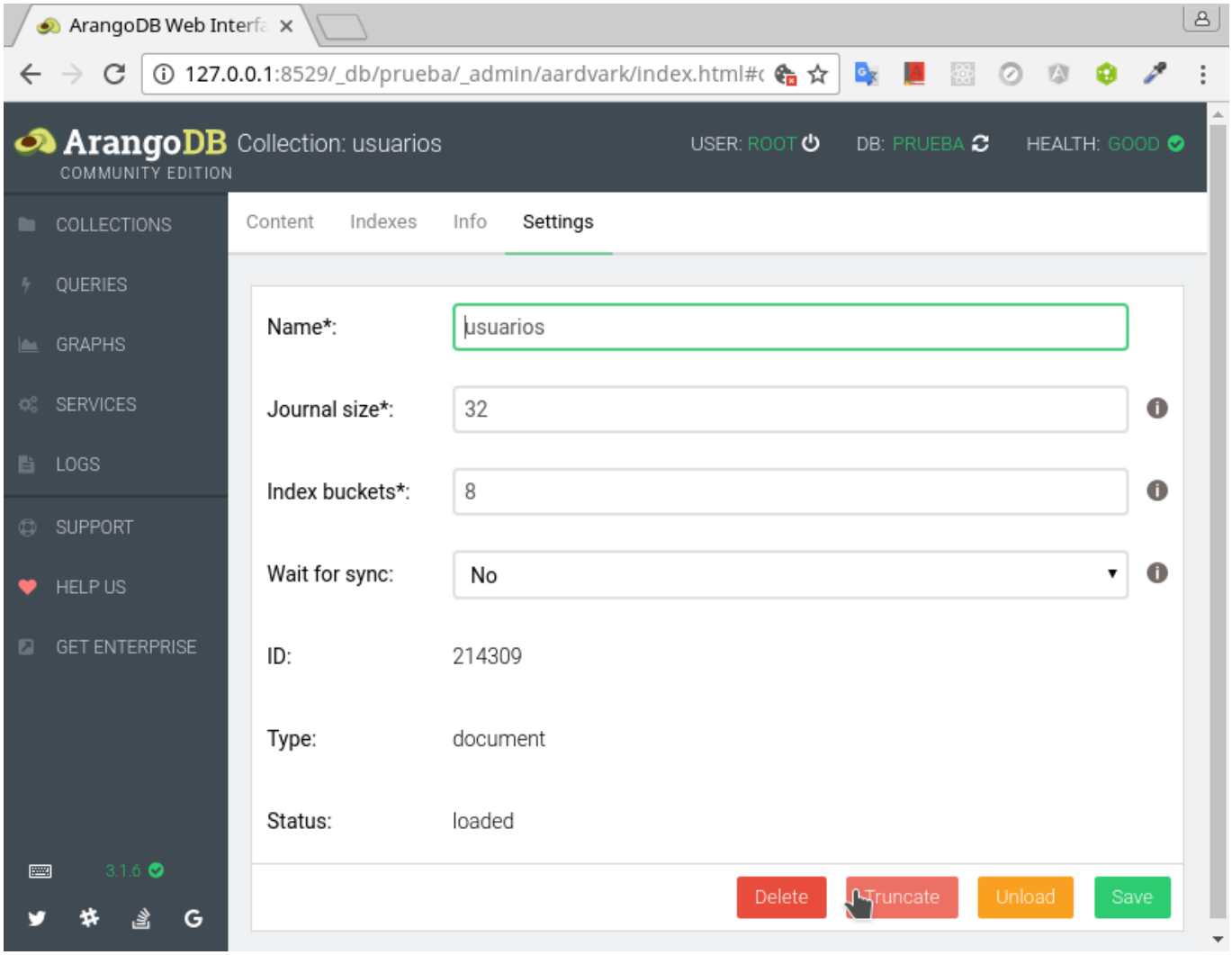
```
db.usuarios.truncate()
db._truncate("usuarios")
```

Truncamiento de colección mediante interfaz web.

Para truncar una colección mediante la interfaz web, ir a COLLECTIONS y hacer clic en la colección en cuestión:



A continuación, ir a Settings y hacer clic en Truncate:



Supresión de colecciones

La supresión de una colección (*collection drop*) es la operación mediante la cual se elimina una determinada colección y, por ende, los documentos almacenados en ella y sus índices. Se puede realizar mediante el *shell* y la interfaz web.

Permisos

Para suprimir una colección, el usuario debe tener concedido el permiso de lectura/escritura sobre la base de datos donde se encuentra la colección.

Supresión de colección mediante arangosh

Para suprimir una colección mediante el *shell* de ArangoDB, se puede utilizar el método drop() del objeto colección o el método _drop() del objeto db:

```
db.colección.drop()
db.colección.drop(opts)
db._drop(name)
db._drop(name, opts)
```

Parámetro	Tipo de datos	Descripción
name	string	Nombre de la colección a suprimir.
opts	object	Opciones de supresión: • isSystem (boolean). Para suprimir colecciones de sistema, esta opción debe fijarse a true. Valor predeterminado: false.

Ejemplos:

```
db.usuarios.drop()
db._drop("usuarios")
```

Supresión de colección mediante interfaz web

La supresión de una colección mediante la interfaz web de ArangoDB es tan sencilla como el truncamiento. Hay que ir a COLLECTIONS, seleccionar la colección, ir a Settings y finalmente hacer clic en Delete.

