Actualización de documentos

Tiempo estimado: 20min

Una vez sabemos cómo insertar y extraer documentos de las colecciones, vamos a examinar detenidamente cómo modificar documentos existentes.

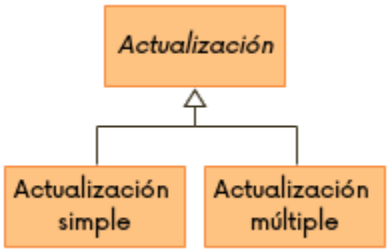
Comenzamos introduciendo el concepto de actualización de documentos y los distintos tipos de actualización existentes. A continuación, se presenta detalladamente las operaciones de actualización completa y parcial. Después, se presenta la operación UPSERT. Y finalmente, se describe cómo utilizar el comando RETURN de AQL para indicar los documentos a devolver en una consulta de modificación.

Al finalizar la lección, el estudiante sabrá:

- Para qué se usa el campo _rev de los documentos.
- Cómo reemplazar o actualizar documentos mediante la API simple y AQL.
- Cómo realizar UPSERTs mediante AQL.
- Cómo utilizar el comando RETURN y las variables NEW y OLD en las consultas AQL de modificación para devolver un resultado concreto.

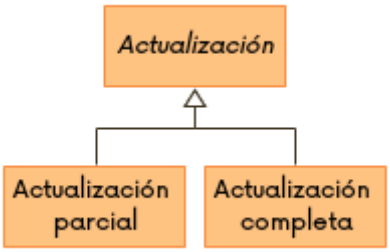
Introducción

La actualización de documentos (*document update*) es la operación mediante la cual se actualiza uno o más documentos de una colección. Atendiendo al número de documentos que puede actualizar la operación, se distingue entre actualización simple o múltiple.



La actualización simple (*simple update*) sólo puede actualizar un único documento. Mientras que la actualización múltiple (*multiple update*), varios.

Por otra parte, se puede distinguir entre actualizaciones completas o parciales, atendiendo a qué se modifica de cada documento.



Una actualización completa (*full update*) reemplaza un documento por otro. Mientras que una actualización parcial (*partial update*) actualiza sólo algunos campos del documento.

Vamos a examinar detenidamente los cuatro tipos de actualización en lo que resta de lección.

Campos claves

Recordemos que los campos claves, _key e _id, no se pueden cambiar de ninguna manera.

Campo de revisión

El campo de revisión (*revision field*), representado mediante el campo _rev, contiene un valor que se actualiza tras cada cambio sufrido por un documento. Lo fija inicialmente ArangoDB cuando se inserta el documento y lo modifica automáticamente cada vez que se actualiza parcial o completamente. Al igual que _key e _id, tampoco se puede modificar explícitamente.

Actualizaciones completas

Una actualización completa (*full update*) es una actualización, simple o compuesta, que reemplaza un documento por otro. Se puede realizar mediante la API simple o AQL.

Permisos

Para reemplazar un documento por otro, es necesario que el usuario tenga concedido el permiso de lectura/escritura sobre la base de datos en la que se encuentra la colección.

Actualización completa mediante API simple

La actualización completa se puede realizar mediante los métodos replace() y replaceByExample() del objeto colección.

Método replace()

El método replace() reemplaza un documento a partir de una clave:

```
replace(selector, doc)
replace(selector, doc, opts)
```

Parámetro	Tipo de datos	Descripción
selector	string u object	Clave del documento a reemplazar.
doc	object	Documento a asignar.
opts	object	Opciones de actualización: • waitForSync (boolean). ¿Actualización síncrona? • overwrite (boolean). ¿Mantener el valor del campo _rev del documento? true, sí; false, no. • returnNew (boolean). ¿Devolver el documento resultante? true, sí; false, no. • returnOld (boolean). ¿Devolver el documento reemplazado? true, sí; false, no. • silent (boolean). ¿No devolver nada? true, sí; false, no.

El parámetro selector puede ser una cadena, en cuyo caso deberá indicar la clave principal o el identificador del documento a reemplazar. He aquí un ejemplo ilustrativo:

```
db.bands.replace("bands/the-psychedelic-furs", {
  name: "The Psychedelic Furs",
  origin: "London, UK"
})
```

También es posible indicar la clave o el identificador mediante un objeto. Ejemplo:

```
db.bands.replace({_key: "the-psychedelic-furs"}, {
  name: "The Psychedelic Furs",
  origin: "London, UK"
})
```

Si el documento a reemplazar no existe, el método propaga una excepción ArangoError.

Es posible reemplazar varios documentos, para ello, hay que utilizar la siguiente sobrecarga del método replace():

```
replace(selectors, docs)
replace(selectors, docs, opts)
```

Parámetro	Tipo de datos	Descripción

selectors	object[]	Indica las claves de los documentos a reemplazar.
docs	object[]	Documentos a asignar.
opts	object	Opciones de actualización.

Método replaceByExample()

Si se necesita reemplazar un documento a partir de campos no claves, se puede utilizar el método replaceByExample():

```
replaceByExample(filter, doc)
replaceByExample(filter, doc, opts)
```

Parámetro	Tipo de datos	Descripción
filter	object	Condición de búsqueda.
doc	object	Documento a asignar.
opts	object	Opciones de actualización.

Ejemplo:

```
db.bands.replaceByExample({name: "The Psychedelic Furs"}, {
  name: "The Psychedelic Furs",
  origin: "London, UK"
})
```

Recuerde que los campos _key e _id se mantendrán. Por lo que no es necesario indicarlos en el nuevo documento.

Actualización completa mediante AQL

Para reemplazar un documento mediante AQL, hay que utilizar el comando REPLACE:

```
REPLACE selector
WITH documento
IN colección

REPLACE documento
IN colección
```

El selector debe indicar la clave del documento a reemplazar, en forma de cadena de texto u objeto. Se puede especificar de manera separada o dentro del nuevo documento.

Ejemplos ilustrativos:

```
REPLACE {_key: "the-psychedelic-furs"}
WITH {name: "The Psychedelic Furs", origin: "UK"}
IN bands

REPLACE {_key: "the-psychedelic-furs", name: "The Psychedelic Furs", origin: "UK"}
IN bands
```

Recordemos que para ejecutar la consulta, se debe usar el método _query() del objeto db y, si es necesario, se puede usar una consulta parametrizada.

Si necesitamos realizar el filtro de un documento a partir de campos no claves, se puede utilizar las cláusulas FOR y FILTER sin problemas:

```
FOR b IN bands
FILTER b.name == "The Psychedelic Furs"
REPLACE {_key: b._key}
WITH {name: "The Psychedelic Furs", origin: "UK"}
IN bands
```

No olvide que el comando REPLACE siempre espera un campo clave y una cláusula IN específica, además de la específica de la cláusula FOR.

Actualizaciones parciales

Una actualización parcial (*partial update*) es una actualización que modifica el valor de uno o más campos, dejando los demás sin modificar. Se puede realizar mediante la API simple o AQL.

Permisos

Para realizar una actualización parcial, es necesario que el usuario tenga concedido el permiso de lectura/escritura sobre la base de datos en la que se encuentra la colección.

Actualización parcial mediante API simple

Se puede utilizar los métodos update() y updateByExample() del objeto colección.

Método update()

El método update() permite modificar un documento a partir de su clave:

```
update(selector, fields)
update(selector, fields, opts)
```

Parámetro	Tipo de datos	Descripción
selector	string u object	Clave del documento a modificar.
fields	object	Campos a modificar.
opts	object	Opciones de actualización: • waitForSync (boolean).¿Actualización síncrona? • returnNew (boolean). ¿Devolver el documento tras la actualización? • returnOld (boolean). ¿Devolver el documento antes de la actualización? • keepNull (boolean). ¿Mantener los campos nulos? true, sí; false, no, suprimir aquellos cuyo valor sea null.

El parámetro selector puede ser una cadena, en cuyo caso deberá indicar la clave principal o el identificador del documento a reemplazar. También es posible indicar la clave o el identificador mediante un objeto.

He aquí un ejemplo ilustrativo que muestra la actualización del campo origin y la añadidura del campo website:

```
127.0.0.1:8529@prueba> db.bands.document("the-stone-roses")
{
  "_key" : "the-stone-roses",
  "_id" : "bands/the-stone-roses",
  "_rev" : "_USCD7YW---",
  "name" : "The Stone Roses",
  "origin" : "UK"
}
127.0.0.1:8529@prueba> db.bands.update({_key: "the-stone-roses"}, {origin:
"Manchester, UK", website: "thestoneroses.org"})
{
  "_id" : "bands/the-stone-roses",
  "_key" : "the-stone-roses",
  "_rev" : "_USCEmhO---",
  "_oldRev" : "_USCD7YW---"
}
127.0.0.1:8529@prueba> db.bands.document("the-stone-roses")
{
  "_key" : "the-stone-roses",
  "_id" : "bands/the-stone-roses",
  "_rev" : "_USCEmhO---",
  "name" : "The Stone Roses",
  "origin" : "Manchester, UK",
  "website" : "thestoneroses.org"
}
127.0.0.1:8529@prueba>
```

Si el documento a actualizar no existe, el método propaga una excepción ArangoError.

Método updateByExample()

Si se necesita modificar uno o más documentos a partir de campos no claves, se puede usar el método

updateByExample():

```
updateByExample(filter, fields)
updateByExample(filter, fields, opts)
```

Parámetro	Tipo de datos	Descripción
filter	object	Condición de búsqueda.
fields	object	Campos a actualizar.
opts	object	Opciones de actualización. Las mismas que las del método update().

Ejemplo ilustrativo:

```
db.bands.updateByExample({name: "BMX Bandits"}, {origin: "Bellshill, Scotland"})
```

Actualización parcial mediante AQL

La actualización parcial es muy similar a la completa, pero se utiliza el comando UPDATE, en vez de REPLACE:

```
UPDATE selector
WITH campos
IN colección

UPDATE campos
IN colección
```

El selector debe indicar la clave del documento a reemplazar, en forma de cadena de texto u objeto. Se puede especificar de manera separada o dentro del documento final.

Ejemplos ilustrativos:

```
UPDATE {_key: "the-psychedelic-furs"}
WITH {origin: "London, UK", website: "thepsychedelicfurs.com"}
IN bands

UPDATE {
  _key: "the-psychedelic-furs",
  origin: "London, UK",
  website: "thepsychedelicfurs.com"
} IN bands
```

Es posible hacer actualizaciones, simples o múltiples, a partir de documentos obtenidos de una selección, tal como muestra los siguientes ejemplos:

```
FOR b IN bands
FILTER b.name == "BMX Bandits"
UPDATE b
WITH {origin: "Scotland"}
IN bands

FOR b IN bands
FILTER b.name == "BMX Bandits"
UPDATE {_key: b._key}
WITH {origin: "Scotland"}
IN bands
```

UPSERTs

Una operación UPSERT (*update-insert operation*) es aquella que se comporta como una inserción cuando el documento buscado no existe, o bien como una actualización parcial cuando existe. Se puede realizar mediante AQL.

Permisos

Para realizar un UPSERT, es necesario que el usuario tenga concedido el permiso de lectura/escritura sobre la base de datos en la que se encuentra la colección.

UPSERT mediante AQL

En AQL, se puede realizar esta operación mediante el comando UPSERT:

```
UPSERT filtro
INSERT documento
UPDATE campos
IN colección
```

Veamos un ejemplo ilustrativo:

```
UPSERT {name: "The Mission"}
INSERT {_key: "the-mission", name: "The Mission", origin: "Leeds, UK"}
UPDATE {origin: "Leeds, UK"}
IN bands
```

Comando RETURN

Recordemos que el comando RETURN de AQL indica qué debe devolver la consulta. Se utiliza principalmente en las consultas de selección, pero también se puede utilizar con las consultas de modificación como, por ejemplo, INSERT, REPLACE, UPDATE y REMOVE.

En un INSERT, se puede utilizar la variable predefinida NEW para indicar que devuelva el documento insertado. En un UPDATE o REPLACE, se puede utilizar tanto NEW como OLD: NEW representa el documento tal cual queda tras la modificación; y OLD, el documento antes de la modificación. Y en las supresiones, que veremos en breve en una lección posterior, pueden utilizar OLD para referenciar al documento suprimido.

Veamos unos ejemplos ilustrativos autoexplicativos:

```
INSERT {
  name: "Badly Drawn Boy",
  origin: "England"
} INTO bands
RETURN NEW

FOR b IN bands
FILTER b.name == "Bombay Bicycle Club"
UPDATE b
WITH {origin: "England"}
IN bands
RETURN {
  oldOrigin: OLD.origin,
  newOrigin: NEW.origin
}

FOR b IN bands
FILTER b.name == "The Bluetones"
REMOVE b
IN bands
RETURN OLD._key
```

