Funciones AQL de usuario

Tiempo estimado: 10min

En la lección anterior, presentamos las funciones AQL, concretamente las que vienen de fábrica con ArangoDB. Ahora, vamos a describir cómo podemos añadir nuevas funciones al catálogo de una base de datos.

Comenzamos recordando el concepto de función e introduciendo el concepto de determinismo. A continuación, se muestra cómo registrar, listar y suprimir funciones de usuario. Después, se presenta la función aql_warning() con la que propagar errores al motor de ejecución. Finalmente, se enumera algunos aspectos a tener en cuenta sobre las funciones de usuario.

Al finalizar la lección, el estudiante sabrá:

- Qué es una función de usuario.
- Qué es el determinismo.
- Cómo registrar, listar y suprimir funciones de usuario.
- Cómo propagar errores al motor de ejecución desde una función de usuario.

Introducción

ArangoDB permite que utilicemos funciones en AQL para realizar cálculos complejos, tanto en las cláusulas de restricción como en las de asignación. Atendiendo a si vienen de fábrica con la instancia o las definimos nosotros mismos, distinguimos entre funciones predefinidas y funciones de usuario.

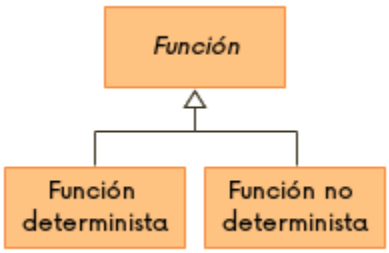
Una función de usuario (*user function* o *user-defined function*) es una función JavaScript con la que extender el catálogo de funciones disponible en una determinada base de datos. Se distingue dos conceptos, a la hora de registrar una nueva función: el espacio de nombres y el nombre de la función. Toda función debe tener un nombre (*name*) que la identifica de las demás. Por convenio y buenas prácticas, con objeto de evitar conflictos de nombres, se recomienda asignar las funciones de usuario a espacios de nombres (*namespaces*), un prefijo de función que sirve para agrupar funciones relacionadas con una funcionalidad, un aspecto o una organización. El espacio de nombres se separa del nombre de la función mediante un doble dos puntos (::).

Así pues, a la hora de invocar una función, atendiendo a si se ha definido en un espacio de nombres, usaremos uno de estos dos formatos:

```
función(argumentos)
espacioNombres::función(argumentos)
```

Determinismo de la función

El determinismo de una función (*function determinism*) indica si, bajo los mismos argumentos de entrada, la función devolverá siempre el mismo valor o puede devolver otro distinto. Se distingue entre funciones deterministas y no deterministas.



Una función determinista (*deterministic function*) es aquella que siempre devuelve el mismo valor para los mismos argumentos. En cambio, una función no determinista (*nondeterministic function*) puede devolver un valor distinto para los mismos argumentos, depende del momento de la llamada.

Por ejemplo, la función abs(), que devuelve el valor absoluto de un número, devolverá siempre el mismo valor para el mismo número. Así, si se le pasa -2, devolverá siempre 2; y si se le pasa -9, 9. En cambio, la función date_now() *no* es determinista, porque según el momento en el que la invoquemos devolverá un valor u otro.

Las funciones deterministas son cacheables, mejorando así el rendimiento de la instancia, pues si el resultado se encuentra cacheado, el motor no necesita volver a invocar la función.

Registro de función de usuario

El registro de función (*function register*) es la operación mediante la cual damos de alta una función en el catálogo de la base de datos actual. Se realiza mediante la función register() del módulo @arangodb/aql/functions, el cual, por convenio y buenas prácticas, se suele importar como aqlfns o aqlfunctions:

```
register(name, code, isDeterministic)
register(name, code)
```

Parámetro	Tipo de datos	Descripción
name	string	Nombre de la función, incluido el espacio de nombres.
code	string o function	Código JavaScript de la función.
isDeterministic	boolean	¿Función determinista? true, sí; false, no.

Si la función ya existe, será reemplazada.

He aquí un ejemplo ilustrativo:

```
127.0.0.1:8529@prueba> aqlfns = require("@arangodb/aql/functions")
{
  "unregister" : [Function "(nam" ...],
  "unregisterGroup" : [Function "(grou" ...],
  "register" : [Function "(name, code, isDeterministi" ...],
  "toArray" : [Function "(grou" ...]
}
127.0.0.1:8529@prueba> aqlfns.register("my::math::sum", function(x, y) { return x + y;
}, true)
false
127.0.0.1:8529@prueba> db._query("RETURN my::math::sum(123, 456)").toArray()
[
  579
]
127.0.0.1:8529@prueba>
```

Las funciones se registran en la colección de sistema _aqlfunctions de la base de datos. Pero cuidado, no hay que modificar esta colección directamente para evitar efectos secundarios.

Listado de funciones de usuario

Para conocer las funciones de usuario disponibles, hay que usar la función toArray() del módulo @arangodb/aql/functions:

```
toArray() : object[]
toArray(prefix) : object[]
```

Parámetro	Tipo de datos	Descripción
prefix	string	Espacio de nombres.

Supresión de función de usuario

Para suprimir una función de usuario, hay que usar la función unregister() del módulo @arangodb/aql/functions:

```
unregister(name)
```

Parámetro	Tipo de datos	Descripción

name	string	Nombre de la función a suprimir.

Si lo que se desea es suprimir todas las funciones de un espacio de nombres, hay que usar unregisterGroup():

```
unregisterGroup(prefix)
```

Parámetro	Tipo de datos	Descripción
prefix	string	Espacio de nombres.

Función aql_warning()

La función aql_warning() se encuentra disponible en el cuerpo de las funciones de usuario para propagar errores al motor de ejecución:

```
aql_warning(error, msg)
```

Parámetro	Tipo de datos	Descripción
error	number	Código de error. Disponible en la propiedad errors del módulo @arangodb.
msg	string	Mensaje de error.

Aspectos a tener en cuenta de las funciones de usuario

Hay algunos aspectos que tenemos que recordar cuando usamos funciones de usuario:

- No debemos de usarlas como primer criterio en una restricción FILTER.
- Pueden tener cualquier número de parámetros.
- Deberían devolver un valor de un tipo primitivo (null, boolean, number y string) o compuesto (object o array). No deberían devolver objetos de tipo function, Date o RegExp.

