Cross-cluster queries and schema changes

Article • 03/07/2022

When running a cross-cluster query, the cluster that performs the initial query interpretation must have the schema of the entities referenced on the remote clusters.

For example, take the following cross-cluster query:

Kusto

```
Table1 | join (cluster("Cluster2").database("MyDatabase").Table2 ) on
KeyColumn
```

In order for the query to run on *Cluster1*, the columns and their data types of *Table2* must be known. To get that information, a special command is sent from *Cluster1* to *Cluster2*. Sending the command can be an expensive network operation; hence, once the entities are retrieved, they're cached so that future queries referencing the same entities require fewer network operations.

Any changes to the schema of the remote entity may result in unwanted effects. For example, new columns aren't recognized or deleted columns may cause a 'Partial Query Error' instead of a semantic error.

To reduce the possibility of this issue arising, cached schemas expire one hour after retrieval, so that any queries run after that time will work with the up-to-date schema. Alternatively, you can refresh the schema manually by using the Clear Cross Cluster Schema Cache command.

Important

If the clusters are in different tenants, you may need to edit the `trustedExternalTenants` property. Non-trusted external tenants may get an **Unauthorized error (401)** failure. For more information, see **How to allow principals from another tenant to access your cluster**.

Feedback

Cross-database and cross-cluster queries

Article • 01/16/2023

Queries execute with one specific database being "in context". This database is used by default to check permissions, and every entity reference in the query that has no explicit cluster or database qualification is resolved against this default database.

- In Kusto Explorer, the default database is the one selected in the Connections panel, and the current cluster is the connection containing that database.
- When using the client library, the current cluster and the default database are specified by the `Data Source` and `Initial Catalog` properties of the connection strings, respectively.

Queries

To access tables from any database other than the default, the *qualified name* syntax must be used.

To access database in the current cluster.

Kusto

```
database("<database name>").<table name>
```

Database in remote cluster.

Kusto

```
cluster("<cluster name>").database("<database name>").<table name>
```

database name is case-sensitive

cluster name is case-insensitive and can be of one of the following forms:

- Well-formed URL, such as `http://contoso.kusto.windows.net:1234/`. Only HTTP and HTTPS schemes are supported.
- Fully qualified domain name (FQDN), such as `contoso.kusto.windows.net`. This string is equivalent to `https://`**`contoso.kusto.windows.net`**`/`.

- Short name (host name [and region] without the domain part), such as `contoso` or `contoso.westus`. These strings are interpreted as `https://`**`contoso`**`.kusto.windows.net/` and `https://`**`contoso.westus`**`.kusto.windows.net/`.

> Note
>
> Cross-database access is subject to the usual permission checks. To execute a query, you must have read permission to the default database and to every other database referenced in the query (in the current and remote clusters).

Qualified name can be used in any context in which a table name can be used.

All of the following are valid.

Kusto

```
database("OtherDb").Table | where ...

union Table1, cluster("OtherCluster").database("OtherDb").Table2 | project
...

database("OtherDb1").Table1 | join
cluster("OtherCluster").database("OtherDb2").Table2 on Key | join Table3 on
Key | extend ...
```

> Important
>
> If the clusters are in different tenants, you may need to edit the `trustedExternalTenants` property. Non-trusted external tenants may get an **Unauthorized error (401)** failure. For more information, see **How to allow principals from another tenant to access your cluster**.

When *qualified name* appears as an operand of the union operator, then wildcards can be used to specify multiple tables and multiple databases. Wildcards aren't permitted in cluster names.

Kusto

```
union withsource=TableName *, database("OtherDb*").*Table,
cluster("OtherCluster").database("*").*
```

ⓘ Note

> - The name of the default database is also a potential match, so database("*")specifies all tables of all databases including the default.
> - For more ionformation on how schema changes affect cross-cluster queries, see Cross-cluster queries and schema changes

Access restriction

Qualified names or patterns can also be included in restrict access statement, Wildcards in cluster names aren't permitted.

Kusto

```
restrict access to (my*, database("MyOther*").*,
cluster("OtherCluster").database("my2*").*);
```

The above will restrict the query access to the following entities:

- Any entity name starting with *my...* in the default database.
- Any table in all the databases named *MyOther...* of the current cluster.
- Any table in all the databases named *my2...* in the cluster *OtherCluster.kusto.windows.net*.

Functions and Views

Functions and views (persistent and created inline) can reference tables across database and cluster boundaries. The following code is valid.

Kusto

```
let MyView = Table1 join database("OtherDb").Table2 on Key | join
cluster("OtherCluster").database("SomeDb").Table3 on Key;
MyView | where ...
```

Persistent functions and views can be accessed from another database in the same cluster.

Tabular function (view) in `OtherDb`.

Kusto

```
.create function MyView(v:string) { Table1 | where Column1 has v ...  }
```

Scalar function in `OtherDb`.

Kusto

```
.create function MyCalc(a:double, b:double, c:double) { (a + b) / c }
```

In default database.

Kusto

```
database("OtherDb").MyView("exception") | extend
CalCol=database("OtherDb").MyCalc(Col1, Col2, Col3) | take 10
```

Limitations of cross-cluster function calls

Tabular functions or views can be referenced across clusters. The following limitations apply:

- Remote functions must return tabular schema. Scalar functions can only be accessed in the same cluster.
- Remote functions can accept only scalar arguments. Functions that get one or more table arguments can only be accessed in the same cluster.
- Remote functions' result schema must be fixed (known in advance without executing parts of the query). This precludes the use of query constructs such as the `pivot` plugin. (Note that some plugins, such as the `bag_unpack` plugin, supports a way to indicate the result schema statically, and in this form it *can* be used in cross-cluster function calls.)
- For performance reasons, the schema of remote entities is cached by the calling cluster after the initial call. Therefore, changes made to the remote entity may result in a mismatch with the cached schema information, potentially leading to query failures. For more information, see Cross-cluster queries and schema changes.

The following cross-cluster call is valid.

Kusto

```
cluster("OtherCluster").database("SomeDb").MyView("exception") | count
```

The following query calls a remote scalar function `MyCalc`. This call violates rule #1, so it's not valid.

Kusto

```
MyTable | extend
CalCol=cluster("OtherCluster").database("OtherDb").MyCalc(Col1, Col2, Col3)
| take 10
```

The following query calls remote function `MyCalc` and provides a tabular parameter. This call violates rule #2, so it's not valid.

Kusto

```
cluster("OtherCluster").database("OtherDb").MyCalc(datatable(x:string,
y:string)["x","y"] )
```

The following query calls remote function `SomeTable` that has a variable schema output based on the parameter `tablename`. This call violates rule #3, so it's not valid.

Tabular function in `OtherDb`.

Kusto

```
.create function SomeTable(tablename:string) { table(tablename)  }
```

In default database.

Kusto

```
cluster("OtherCluster").database("OtherDb").SomeTable("MyTable")
```

The following query calls remote function `GetDataPivot` that has a variable schema output based on the data (pivot() plugin has dynamic output). This call violates rule #3, so it's not valid.

Tabular function in `OtherDb`.

Kusto

```
.create function GetDataPivot() { T | evaluate pivot(PivotColumn) }
```

Tabular function in the default database.

Kusto

```
cluster("OtherCluster").database("OtherDb").GetDataPivot()
```

Displaying data

Statements that return data to the client are implicitly limited by the number of records returned, even if there's no specific use of the `take` operator. To lift this limit, use the `notruncation` client request option.

To display data in graphical form, use the render operator.