Article

Sam Duncan · May 6, 2021 4m read

Open Exchange

Monitoring BI cube usage and cleaning up unused cubes

When you have been using cubes for business intelligence in a namespace for some time, you may find that there are many cubes in the namespace, only some of which are actively being used. However, it can be difficult to tell which cubes users are or are not querying, and maintaining unused cubes can be costly both in terms of storage and of computation to keep them up to date. This article provides some suggestions and examples for monitoring which cubes are in active use, and for removing cubes that you determine are no longer necessary.

Monitoring using ^DeepSee.AuditQueryCode

As described in the IRIS BI documentation, you can set an ObjectScript statement as the value of the ^DeepSee.AuditQueryCode global in a namespace, and it will be executed each time an MDX query is run in that namespace. Most often, you will want to set the global to a statement that calls a custom routine or method to record the data you are interested in. For instance, you might want to call a routine that logs the query text, the name of the cube being queried, the username of the user running the query, the current time, and the ID of the process running the query in a new object of a custom class you have created. This Open Exchange application provides an example of this kind of class and how to set it up.

Making use of your monitoring data

Once you are logging usage data as described above, you have several options, such as SQL queries, for viewing the data you have logged. One useful option could be to create a new cube based on the persistent class where you have logged the data. This will allow you to explore the data in the Analyzer, filtering the data and drilling down on areas you are interested in. Each fact in this cube will be represent a single time that an MDX query was run. While it may not make sense to include the full text of each MDX query as a level of the cube, you can make it available in a detail listing. The same Open Exchange application includes an example cube based on the BI query usage data.

Cleaning up unused cubes

If you have been monitoring cube usage for some time and see that some cubes are not being used, you may want to remove them. This will not affect the underlying data in the source classes that the cubes are based on, but will remove the cube model, fact table, dimension tables, and associated indices. To remove a cube:

- Determine whether the cube is versioned; if it is, you may want to remove past versions of the cube as well.
- Determine whether any remaining cubes have relationships to the cube you are removing. If so, first consider whether removing the cube will interfere with queries against the related cubes that may be making use of those relationships. If you still want to remove the cube, delete the relationships from the remaining cubes and recompile them. In versions prior to IRIS 2020.1, if a

cube that you deleted a relationship from was the dependent cube in a relationship, rebuild it.

- If there is a subject area based on the cube, you should not remove the base cube unless you also remove the subject area. If another cube lists the cube to be removed under the DependsOn keyword (for example, because they formally share a dimension), you should not delete the cube that appears in the DependsOn unless you first remove the dependency and any references to it from the other cube, and confirm that you can successfully compile the latter.
- If you use the Cube Manager, open it in the Management Portal. If there is a registered group that includes the cube you are removing, take note of the settings for any other cubes in the group and then unregister the group. Save the cube registry and confirm that the registered and unregistered groups load without errors.
- Delete the cube data and indices using %DeepSee.Utils:%KillCube(cubeName), where cubeName is the logical name of the cube.
- Delete the cube class using $System.OBJ.Delete(cubeClass), where cubeClass is the fully-qualified cube class name.
- Run $System.OBJ.DeletePackage(cubeClass) to remove the fact class, dimension and star tables, and other associated classes.
- In the Cube Manager, if you unregistered a group that included the removed cube and there were other cubes in the group, you can now reregister them (it's possible that they will now be split into multiple groups) and configure their update schedules as before. Confirm that cube registry events are completing as scheduled.
- If you had pivot table or dashboard definitions based on the removed cube, consider deleting them to prevent users from encountering errors when trying to load them.

The steps to remove a subject area are similar, but you cannot run %KillCube() on a subject area (and this would be unnecessary, as subject areas do not independently store facts or indices).

#Analytics #Monitoring #InterSystems IRIS Analytics (DeepSee)
Check the related application on InterSystems Open Exchange