Item 113
git_comments:

git_commits:

1. **summary:** [SPARK-24017][SQL] Refactor ExternalCatalog to be an interface
   message: [SPARK-24017][SQL] Refactor ExternalCatalog to be an interface ## What changes were proposed in this pull request? This refactors the external catalog to be an interface. It can be easier for the future work in the catalog federation. After the refactoring, `ExternalCatalog` is much cleaner without mixing the listener event generation logic. ## How was this patch tested? The existing tests Author: gatorsmile <gatorsmile@gmail.com> Closes #21122 from gatorsmile/refactorExternalCatalog.

github_issues:

github_issues_comments:

github_pulls:

1. **title:** [SPARK-24017] [SQL] Refactor ExternalCatalog to be an interface
   body: ## What changes were proposed in this pull request? This refactors the external catalog to be an interface. It can be easier for the future work in the catalog federation. After the refactoring, `ExternalCatalog` is much cleaner without mixing the listener event generation logic. ## How was this patch tested? The existing tests

github_pulls_comments:

1. cc @rxin @cloud-fan
2. **[Test build #89680 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/89680/testReport)** for PR 21122 at commit [`c62bba1`] (https://github.com/apache/spark/commit/c62bba1ed024c7d1d91da8f3d8035de8dc169302). * This patch **fails PySpark unit tests**. * This patch merges cleanly. * This patch adds the following public classes _(experimental)_: * `trait ExternalCatalog ` * ` // Returns the underlying catalog class (e.g., HiveExternalCatalog).` * `class ExternalCatalogWithListener(delegate: ExternalCatalog)`
3. retest this please
4. **[Test build #89681 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/89681/testReport)** for PR 21122 at commit [`c62bba1`] (https://github.com/apache/spark/commit/c62bba1ed024c7d1d91da8f3d8035de8dc169302). * This patch passes all tests. * This patch merges cleanly. * This patch adds the following public classes _(experimental)_: * `trait ExternalCatalog ` * ` // Returns the underlying catalog class (e.g., HiveExternalCatalog).` * `class ExternalCatalogWithListener(delegate: ExternalCatalog)`
5. cc @rdblue
6. To add some more color, the current `ExternalCatalog` is an abstract class and can already be implemented outside of Spark. However the problem is the listeners. Everytime we want to listen to one more event, we need to break the API(`createDatabase` and `doCreateDatabase`). this is very bad for a stable interface. The main goal of this PR is to pull out the listener stuff from `ExternalCatalog`, and make `ExternalCatalog` a pure interface.
7. Thanks for pointing this out, @henryr. This looks like a good change to support multiple catalogs. I think it looks fine, other than exposing `unwrapped` to get the Hive client. I think we're probably abusing that instead of passing the client so I think it makes sense to pass the client correctly instead of using `unwrapped`.
8. **[Test build #90184 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/90184/testReport)** for PR 21122 at commit [`b24071f`] (https://github.com/apache/spark/commit/b24071f6e6ad93aac49a08217f269999ba8ad644). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
9. LGTM
10. +1, LGTM, too. Thank you for choosing this path, @gatorsmile !

11. **[Test build #90218 has finished] (https://amplab.cs.berkeley.edu/jenkins/job/SparkPullRequestBuilder/90218/testReport)** for PR 21122 at commit [`9d97ebf`] (https://github.com/apache/spark/commit/9d97ebf34351acf53a9be5bfa286005f5ee9cb0c). * This patch passes all tests. * This patch merges cleanly. * This patch adds no public classes.
12. Thanks! Merged to master

github_pulls_reviews:

1. @gatorsmile . We had better skip the default implementation here. How do you think about that?
2. Based on your JIRA comment, can we put `@DeveloperApi` or `@InterfaceStability.Unstable` in this PR?
3. Maybe we can move it to `ExternalCatalogWithListener` and mark `SharedState.externalCatalog` as `ExternalCatalogWithListener`
4. This trait is still an internal one. We have not make it public yet.
5. I see. Thank you for confirming, @gatorsmile . Any milestone for making it public?
6. Hopefully, the next release.
7. Is there a better way to pass the Hive client? It looks like the uses of `unwrapped` actually just get the Hive client from the HiveExternalCatalog. If we can pass that through, it would prevent the need for this. I think that would be cleaner, unless there is a problem with that I'm missing.
8. Sure, let us get rid of it from this interface. We can do it in ExternalCatalogWithListener.scala
9. maybe we should define a util method to get hive client from `SparkSession`
10. Why are we passing the client using the catalog in the first place? Is this convenience, or is there a reason why we can't pass both separately? I don't think we should be doing this without a reason. I don't like needless casts.
11. The hive client is just for HiveExternalCatalog. It is meaningless for the other external catalogs.
12. @gatorsmile, the problem is that Hive classes access the Hive client from the HiveExternalCatalog in several places. Why is it getting the client this way? Shouldn't that client get passed separately?
13. nit. Maybe, this spacing change is not your intention.
14. We want to get rid of Hive dependency completely in the near future. Currently, in the source code, only HiveExternalCatalog needs to use/access the `client`. I might not get your point. Could you explain how to pass the client when we keep the client in `HiveExternalCatalog`?
15. There are several places that access the `client`, like `HiveSessionStateBuilder` and `SaveAsHiveFile`. My point is that assuming the catalog is a `HiveExternalCatalog`, casting, and accessing a field of the catalog to get the client isn't a good way to pass the client to where it is needed. Why doesn't this code pass both `client` and `catalog` to these classes to avoid this problem? I don't think it matters that Hive may be removed in the future. Unless removing it is going to happen in a week or two, I'm skeptical that it is going to be soon enough that we should ignore problems with the implementation.
16. In the 2.0 release, we did a lot of work to refactor the previous Hive integration and move the client to HiveExternalCatalog. This is still the ongoing work. If you have time, please help this effort too!
17. @gatorsmile, what is the reason for passing the client as a field of HiveExternalCatalog and not on its own? This requires casting the catalog to access the client, and there doesn't appear to be an obvious reason not to pass the client separately. Given that this is a problem -- the proposed interface was returning a wrapped catalog -- I'm trying to understand why it is this way and whether it should be changed.
18. The client is used for interacting with Hive metastore. Conceptually, it should be part of `HiveExternalCatalog`. If we want to pass it as a field of HiveExternalCatalog , we need to create the client in SharedState, which is in the core module instead of the hive module. Since we are getting rid of Hive from the code base, we do not want a Hive-speicific SharedState. Any better idea?
19. > Conceptually, it should be part of HiveExternalCatalog. Why? If this is true, then why is it used outside of the catalog? As far as a Hive-specific shared state, that sounds like the "right" way to do this. As long as this doesn't leak into the API, it isn't a big problem. But, practices like this that require breaking abstractions by casting to a known concrete class just to avoid passing multiple variables makes Spark more brittle. It is great to hear we want to eliminate Hive, but that doesn't mean the Hive code shouldn't be properly maintained while it is still supported.
20. We plan to get rid of `HiveSessionStateBuilder` and treat `Hive` tables as the regular data source tables. `Hive` should be the same as the other external data sources. For the other data sources, we do not have source-specific shared state and session state. Hive metastore is just used for global metastore. In the

future, it is also pluggable. Thus, it does not make sense to re-introduce Hive SharedState in the current stage.

21. Ideally Hive should not be a first-class citizen in Spark, it's just a data source and a catalog, and nothing more. We want to narrow down the scope of hive usage in Spark, and keeping it only in `HiveExternalCatalog` is a good choice. This is still an on-going effort, as @rdblue pointed out, there are still 2 places Spark uses Hive directly: 1. `HiveSessionStateBuilder`. It's mostly for the ADD JAR, once we move the ADD JAR functionality to `ExternalCatalog`, we can fix it 2. `SaveAsHiveFile`. It's the data source part so it should be allowed to use Hive there. One thing we can improve is the hive client reuse. We cast `HiveExternalCatalog` to get the existing hive client, maybe there is a better way to do it without the ugly casting.
22. The issue is out of scope this PR. We can continue addressing the related issues in the future PRs.
23. @gatorsmile, I agree that we don't need to block this PR on the problem, but could you create or post the issues that track solving these problems?

jira_issues:

jira_issues_comments: