KIP-508: Make Suppression State Queriable

Status

Current state: *Under Discussion*

Discussion thread: thread

JIRA: KAFKA-8403

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

From 2.1, Kafka Streams provides a way to suppress the intermediate state of KTable (KIP-328: Ability to suppress updates for KTables). The 'KTable#suppress' operator introduced in KIP-328 controls what updates downstream table and stream operations will receive. With this feature, the contents of the upstream table are disjointed into two groups, one for the intermediate state in the suppression buffer and the other for final states emitted to the downstream table. The user can query the associated value to a specific key in the downstream table by querying the upstream table (KIP-67: Queryable state for Kafka Streams), since all of the key-value mappings in the downstream table are also stored in the upstream table.

However, there is a limitation; if the user only wants to retrieve the associated value to a specified key (like `ReadOnlyKeyValueStore#get`), it is okay. But if what the user wants is getting an iterator to a suppressed view (like `ReadOnlyKeyValueStore#range` or `ReadOnlyKeyValueStore#all`), we stuck in trouble - since there is no way to identify which key is flushed out beforehand.

One available workaround is materializing the downstream table like `downstreamTable.filter(e -> true, Materialized.as("final-state"))`. However, this way is cumbersome.

Proposed Changes

This KIP proposes to add an option to make suppression state queriable by adding a queriable flag to Suppressed.

Public Interfaces

```
public interface Suppressed<K> extends NamedOperation<Suppressed<K>> {

    ...

    /**
     * Make the suppression queryable.
     *
     * @return The same configuration with query enabled.
     */
    Suppressed<K> enableQuery();

    /**
     * @return Returns true iff the query is enabled.
     */
    boolean isQueryEnabled();

    ....
```

The user can query the suppressed view with `Suppressed#name`, if `Suppressed.isQueryEnabled` is true.

Calling `Suppressed#enableQuery` without specifying name with `Suppressed#withName` is not allowed. For this case, `IllegalArgumentException` is thrown.

Compatibility, Deprecation, and Migration Plan

None.

Rejected Alternatives

KTable#suppress(Suppressed, Materialized<K, V, KeyValueStore>)

This approach feels more consistent with existing APIs with `Materialized` variant (e.g., `KTable#filter(Predicate)` - `KTable#filter(Predicate, Materialized)`) at first appearance. However, this approach introduces two concepts of the name for the same operation: `Suppressed#name` and `Materialized#name`. It is not feasible.

The current API for the `Materialized` variant is just a legacy of nameless operators before KIP-307. In this case, we already have `Suppressed` class and don't need to keep consistency with the old `Materialized` variant methods. So rejected.

KTable#suppress(Suppressed, String)

Another alternative is passing the state store name directly. This approach is neither consistent with the existing APIs nor has clear semantics, since it also introduces two concepts for the same operation. So rejected.