KIP-806: Add session and window query over kv-store in IQv2

- Status
- Motivation
- Public Interfaces
  - WindowKeyQuery
  - WindowRangeQuery
- Proposed Changes
  - Examples

This page is meant as a template for writing a KIP. To create a KIP choose Tools->Copy on this page and modify with your content and replace the heading with the next KIP number and a description of your issue. Replace anything in italics with your own description.

Status

Current state: Adopted

Discussion thread: *here*

JIRA: KAFKA-13494

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

Motivation

Provide an implementation of the `Query` interface, introduced in KIP-796: Interactive Query v2 , to support session and window queries.

Public Interfaces

In this KIP we propose two new public classes:

WindowKeyQuery

This type used to search occurrences of keys in window stores in combination with IQv2. In particular the following mappings are established:

- WindowStore.fetch(K key, Instant timeFrom, Instant timeTo) WindowKeyQuery.withKeyAndWindowStartRange(key, from, to)

WindowRangeQuery

The query type is used to search per window aggregates of keys in window and session stores in combination with IQv2. In particular the following mappings are established:

- WindowStore.fetchAll(Instant timeFrom, Instant timeTo) WindowRangeQuery.withWindowStartRange(from, to)
- SessionStore.fetch(Bytes key) WindowRangeQuery.withKey(key)

There are multiple discussion points here:

- One reason we cannot use WindowKeyQuery to support SessionStore.fetch(key) is because SessionStore.fetch(key) returns a windowed key value iterator, KeyValueIterator<Windowed<Bytes>, byte[]>, whereas WindowKeyQuery is bound to Query<WindowStoreIterator<V>>.
- WindowRangeQuery does not currently support key ranges. We could add more cases here to get better coverage of WindowStore and SessionStore API. In this KIP, however, we focus on a subset of queries and defer support for additional cases to future KIPs.

Proposed Changes

The WindowKeyQuery class:

```
@Evolving
public class WindowKeyQuery<K, V> implements Query<WindowStoreIterator<V>> {

    public static <K, V> WindowKeyQuery<K, V> withKeyAndWindowStartRange(final K key, final Instant timeFrom,
final Instant timeTo);

    public K getKey();

    public Optional<Instant> getTimeFrom();

    public Optional<Instant> getTimeTo();
}
```

The WindowRangeQuery class:

```
@Evolving
public class WindowRangeQuery<K, V> implements Query<KeyValueIterator<Windowed<K>, V>> {

    public static <K, V> WindowRangeQuery<K, V> withKey(final K key);

    public static <K, V> WindowRangeQuery<K, V> withWindowStartRange(final Instant timeFrom, final Instant
timeTo);

    public K getKey();

    public Optional<Instant> getTimeFrom();

    public Optional<Instant> getTimeTo();
}
```

Examples

The following example illustrates the use of the WindowKeyQuey class.

```
final Instant timeTo = Instant.now();
final Instant timeFrom = timeTo.minusSeconds(60);
final WindowKeyQuery<GenericKey, ValueAndTimestamp<GenericRow>> query = WindowKeyQuery.
withKeyAndWindowStartRange(key, timeFrom, timeTo);

final StateQueryRequest<WindowStoreIterator<ValueAndTimestamp<GenericRow>>> request = inStore("rocksdb-window-
store").withQuery(query);
final StateQueryResult<WindowStoreIterator<ValueAndTimestamp<GenericRow>>> result = kafkaStream.query(request);
final WindowStoreIterator<ValueAndTimestamp<GenericRow>> iterator = result.getGlobalResult().getResult();
```

The following example illustrates the use of the WindowQuery class to query a window store.

```
final Instant timeTo = Instant.now();
final Instant timeFrom = timeTo.minusSeconds(60);

final WindowRangeQuery<GenericKey, ValueAndTimestamp<GenericRow>> query = WindowRangeQuery.withWindowStartRange
(timeFrom, timeTo);

final StateQueryRequest<KeyValueIterator<Windowed<GenericKey>, ValueAndTimestamp<GenericRow>>> request = inStore
("inmemory-window-store").withQuery(query);
final StateQueryResult<KeyValueIterator<Windowed<GenericKey>, ValueAndTimestamp<GenericRow>>> result =
kafkaStreams.query(request);
final KeyValueIterator<Windowed<GenericKey>, ValueAndTimestamp<GenericRow>> iterator = result.getGlobalResult().
getResult();
```

Compatibility, Deprecation, and Migration Plan

- Since this is a completely new set of APIs, no backward compatibility concerns are anticipated.
- Since nothing is deprecated in this KIP, users have no need to migrate unless they want to.