Résumé de l'exercice Kafka Streams

Cet exercice montre comment utiliser Kafka Streams pour détecter, enrichir, et analyser en temps réel les changements du nombre d’épisodes des saisons de séries, avec des jointures et des agrégations temporelles.

# 1. Type Structuré : season\_length

CREATE TYPE season\_length AS STRUCT<  
 season\_id INT,  
 episode\_count INT  
>;  
• Représente une saison avec son identifiant et le nombre d’épisodes.  
• Utilisé dans les colonnes 'before' et 'after' pour indiquer l’état avant/après d’un changement.

# 2. Stream : production\_changes

CREATE STREAM production\_changes (  
 rowkey STRING KEY,  
 title\_id INT,  
 change\_type STRING,  
 before season\_length,  
 after season\_length,  
 created\_at STRING  
) WITH (  
 KAFKA\_TOPIC='production\_changes',  
 VALUE\_FORMAT='JSON',  
 TIMESTAMP='created\_at',  
 TIMESTAMP\_FORMAT='yyyy-MM-dd''T''HH:mm:ss.SSSX',  
 PARTITIONS=4  
);  
• Représente les événements de modification en temps réel.  
• Contient les colonnes : rowkey, title\_id, change\_type, before, after, created\_at.  
• Source principale des événements.

# 3. Table : titles

CREATE TABLE titles (  
 id INT PRIMARY KEY,  
 title STRING  
) WITH (  
 KAFKA\_TOPIC='titles',  
 VALUE\_FORMAT='JSON',  
 PARTITIONS=4  
);  
• Table de référence contenant les titres.  
• Sert à enrichir les événements avec le nom du titre.

# 4. Requêtes de test et filtrage

SELECT \* FROM production\_changes  
WHERE created\_at < '2023-04-14 12:00:00'  
EMIT CHANGES;  
  
SELECT \* FROM production\_changes  
WHERE change\_type LIKE 'season%'  
EMIT CHANGES;  
• Permettent de filtrer les événements selon la date ou le type de changement.

# 5. Stream : season\_length\_changes

CREATE STREAM season\_length\_changes WITH (  
 KAFKA\_TOPIC='season\_length\_changes',  
 VALUE\_FORMAT='JSON',  
 PARTITIONS=4,  
 REPLICAS=1  
) AS  
SELECT  
 rowkey,  
 title\_id,  
 created\_at,  
 COALESCE(after->season\_id, before->season\_id) AS season\_id,  
 before->episode\_count AS old\_episode\_count,  
 after->episode\_count AS new\_episode\_count  
FROM production\_changes  
WHERE change\_type = 'season\_length'  
EMIT CHANGES;  
• Filtre les événements où le type de changement est 'season\_length'.  
• Extrait les anciennes et nouvelles valeurs.

# 6. Jointure simple avec la table titles

SELECT title FROM season\_length\_changes  
JOIN titles ON season\_length\_changes.title\_id = titles.id  
EMIT CHANGES;  
• Permet d’associer le nom du titre à chaque événement de changement.

# 7. Stream enrichi : season\_length\_changes\_enriched

CREATE STREAM season\_length\_changes\_enriched WITH (  
 KAFKA\_TOPIC='SEASON\_LENGTH\_CHANGES\_ENRICHED',   
 PARTITIONS=4,   
 REPLICAS=1,   
 TIMESTAMP='created\_at',  
 TIMESTAMP\_FORMAT='yyyy-MM-dd HH:mm:ss'  
) AS  
SELECT  
 s.rowkey,  
 s.title\_id,  
 s.created\_at,  
 s.season\_id,  
 s.old\_episode\_count,  
 s.new\_episode\_count,  
 t.title  
FROM season\_length\_changes s  
JOIN titles t ON s.title\_id = t.id  
EMIT CHANGES;  
• Combine les informations détaillées du changement avec le nom du titre.

# 8. Agrégation horaire : season\_length\_change\_counts

CREATE TABLE season\_length\_change\_counts  
WITH (KAFKA\_TOPIC='season\_length\_change\_counts') AS  
SELECT  
 title,  
 COUNT(\*) AS change\_count,  
 LATEST\_BY\_OFFSET(new\_episode\_count) AS episode\_count  
FROM season\_length\_changes\_enriched  
WINDOW TUMBLING (SIZE 1 HOUR)  
GROUP BY title  
EMIT CHANGES;  
• Agrège les changements par heure et par titre.  
• Affiche aussi le dernier nombre d’épisodes connu par titre.