MU5IN852
Bases de Données Large Echelle

data streaming et requêtes

Octobre 2021

Objectifs

- Aperçu du data streaming
- Notion de fenêtre sur des flux
- Notion de requêtes continues
  - Jointures sur des fenêtres
- Perspectives

Références

- Conférence internationale SIGMOD 2018
  - Titre de l'article

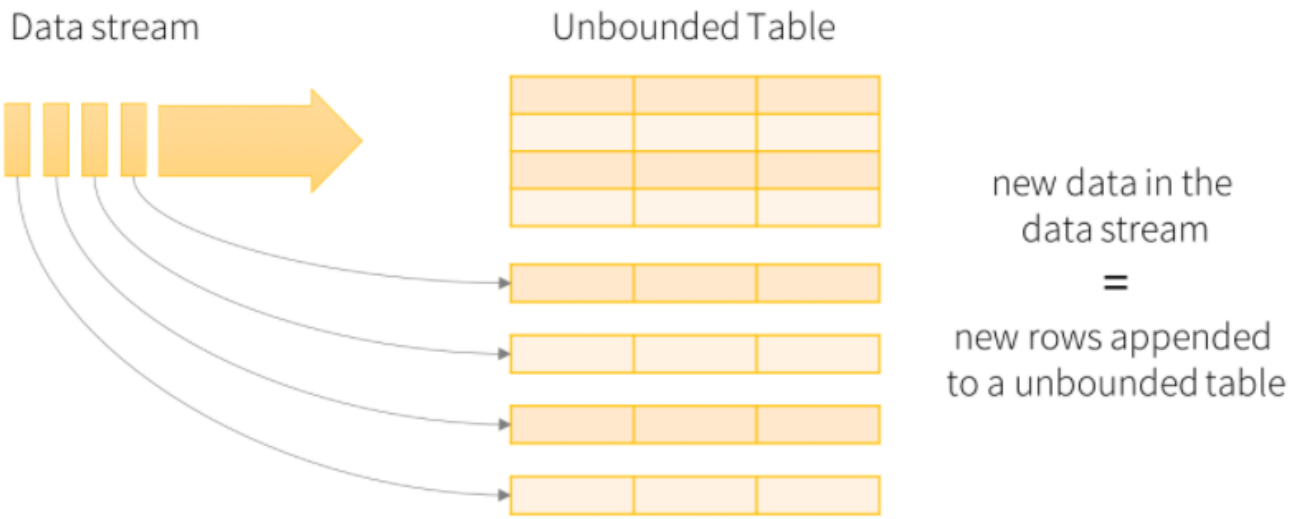
    Structured Streaming: A Declarative API for Real-Time Applications in Apache Spark
  - Auteurs

    Armbrust et al : Databricks, Stanford Univ
  - URL

    https://databricks.com/wp-content/uploads/2018/12/sigmod_structured_streaming.pdf

Contexte : Flux

- Données produites en continu
  - Ensemble ordonné de tuples, de taille infinie
    - Ordre partiel si la source est distribuée
  - Estampille
    - attribut date d'événement ou date d'arrivée



Data stream as an unbounded table

Motivations et Défis

- Requêtes incrémentales complexes à exprimer
  - Besoin de langage déclaratif
- Chaîne de traitement intégrée
- Défis opérationnels
  - Pannes et retards dus aux stragglers
    - tâches « à la traine »
  - Mise à jour des applis traitant un stream
  - Redimensionnement des ressources allouées
- Métrique de performance
  - Débit : nombre de tuples traités par minute

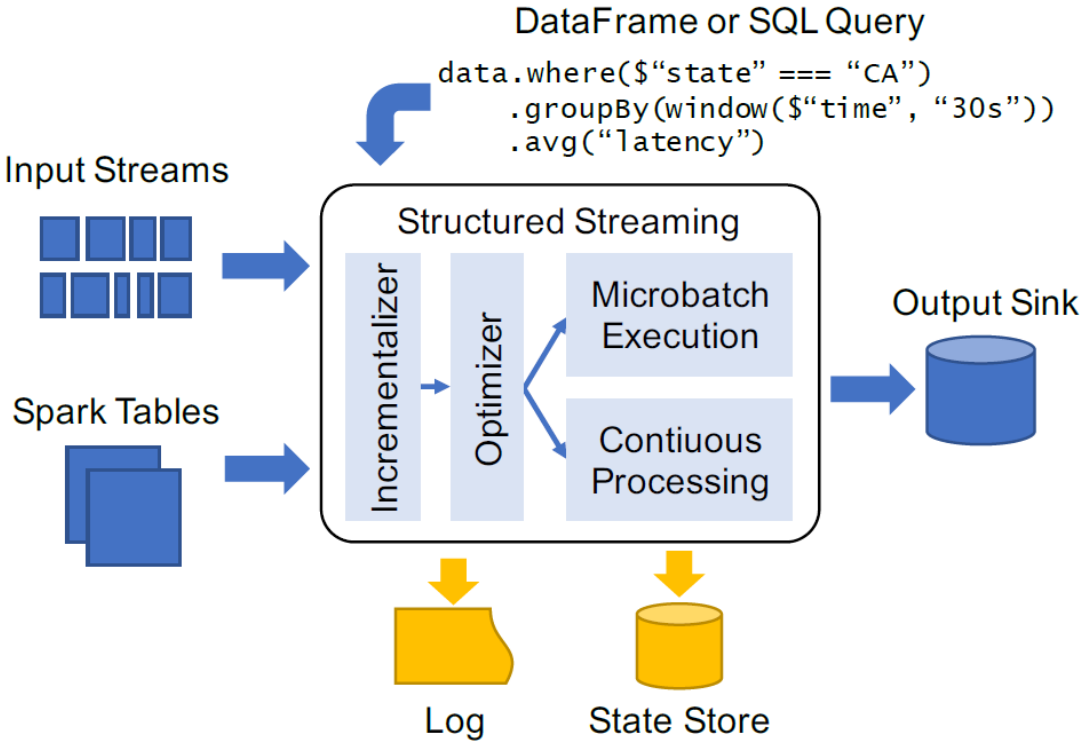
  versus
  - Latence : temps de réponse d'une requête
    - date du résultat de la requête – date d'arrivée de la donnée

Système de streaming

- Système de gestion des flux et des requêtes
  - Scalable : architecture distribuée
- Gestion des flux entrants
  - Tolérance aux pannes
    - Stockage temporaire des flux
    - Possibilité de répéter l'arrivée d'un flux
  - Propriété sémantique
    - Chaque tuple arrive une et une seule fois : « exaclty once »
  - Exemple : Kafka
- Gestion des flux sortants
  - Tolérance aux pannes
    - Ecriture indempotente : 1 ou plusieurs invocations d'une écriture produit le même résultat

Architecture du système Structured Streaming

- Données mixtes :
  - Flux dynamiques et/ou tables statiques
- Requêtes déclaratives



Exécution de requêtes

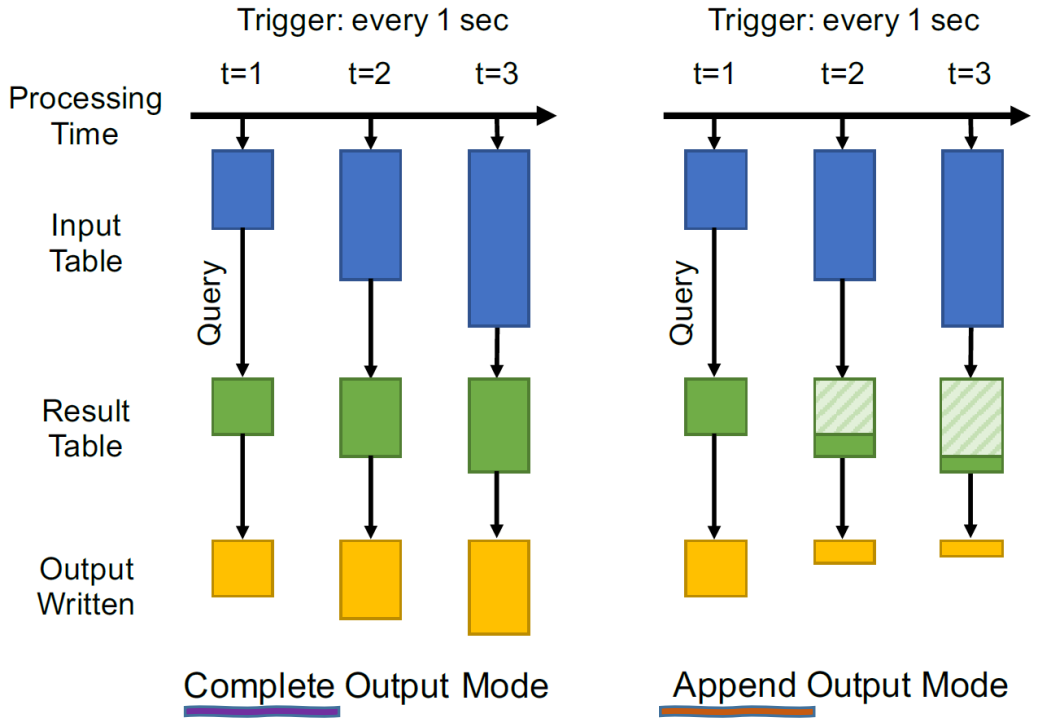
- Requête posée sur toutes les données *d* telles que
  - *d.date <= t*
- Modes d'exécution
  - **Périodique** : exécution toutes les *n* secondes

  ou
  - Continue : exécution à chaque nouveau tuple entrant

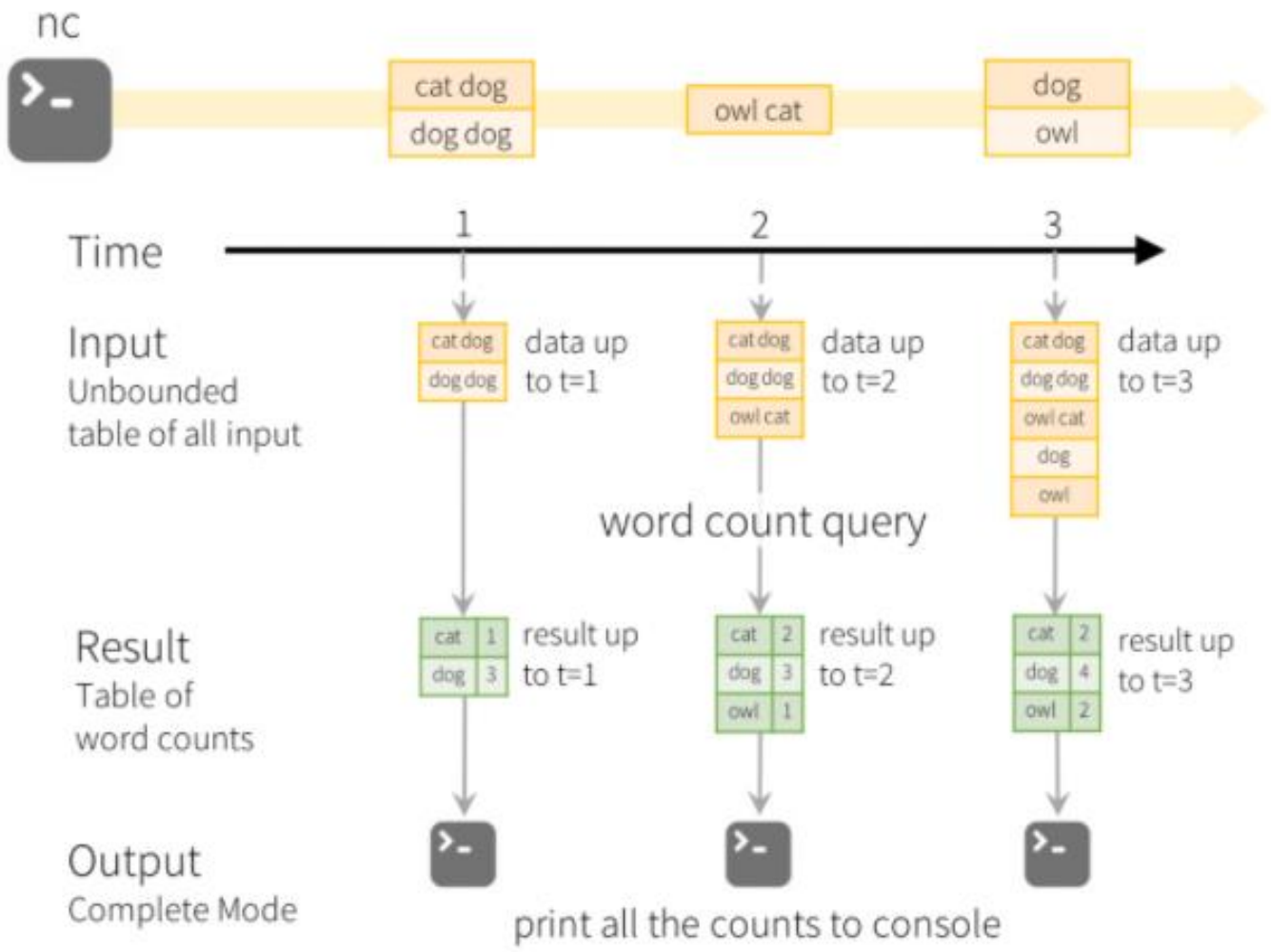
Exécution périodique : mode de sortie

Exécution périodique de la requête avec trois modes de sortie possibles :

- **Complete** : Résultat complet à chaque instant t
- **Append** : Résultat = seulement les nouveaux tuples
- Update : Résultat = les tuples à modifier ou à ajouter



Exemple :
la requête « word count » sur un flux

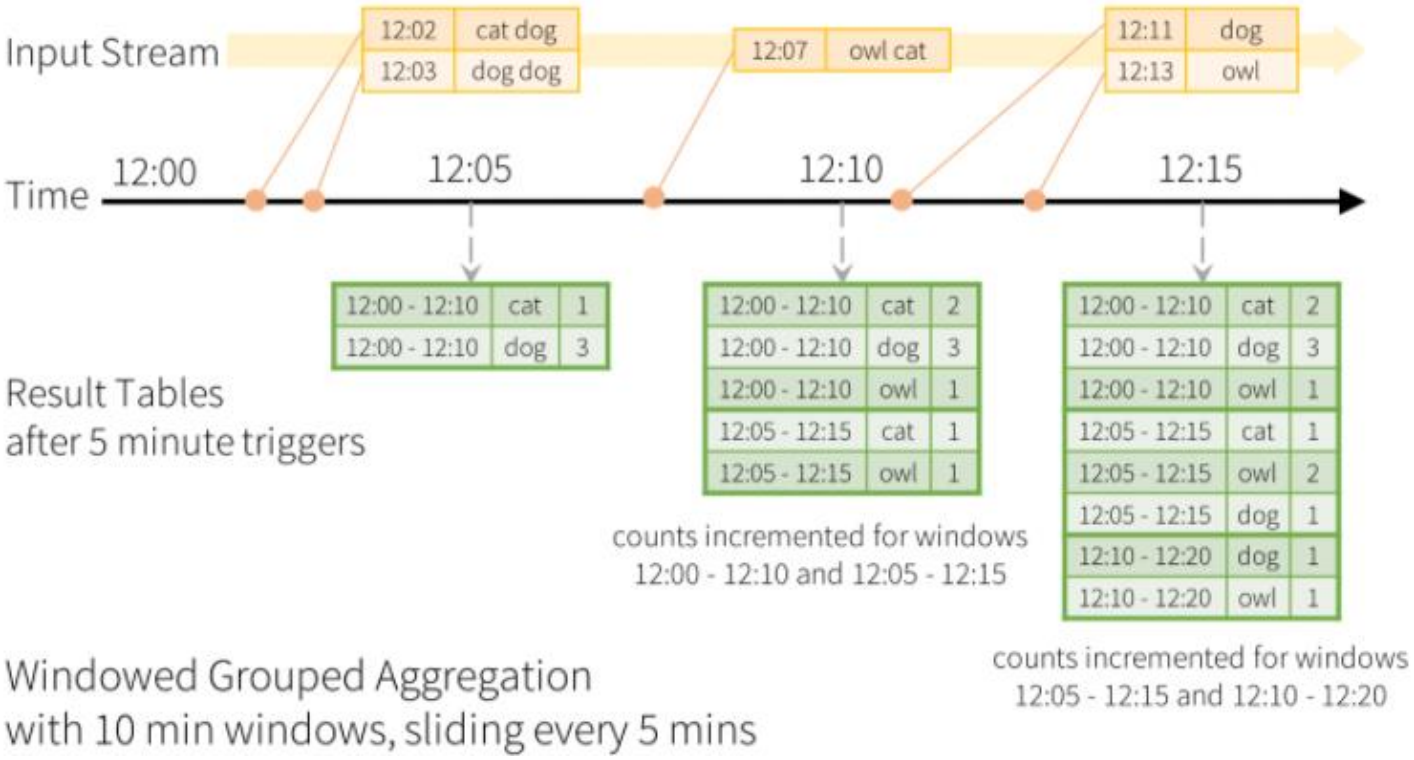


Fenêtrage temporel

- basé sur la date de l'événement
  - Attribut du flux entrant
- Syntaxe : GROUP BY WINDOW attr taille décalage
  - attribut de type date
  - taille de la fenêtre
  - décalage entre les dates de début de deux fenêtres consécutives
- Recouvrement partiel des fenêtres consécutives
  - si décalage < taille

Fenêtre temporelle avec recouvrement

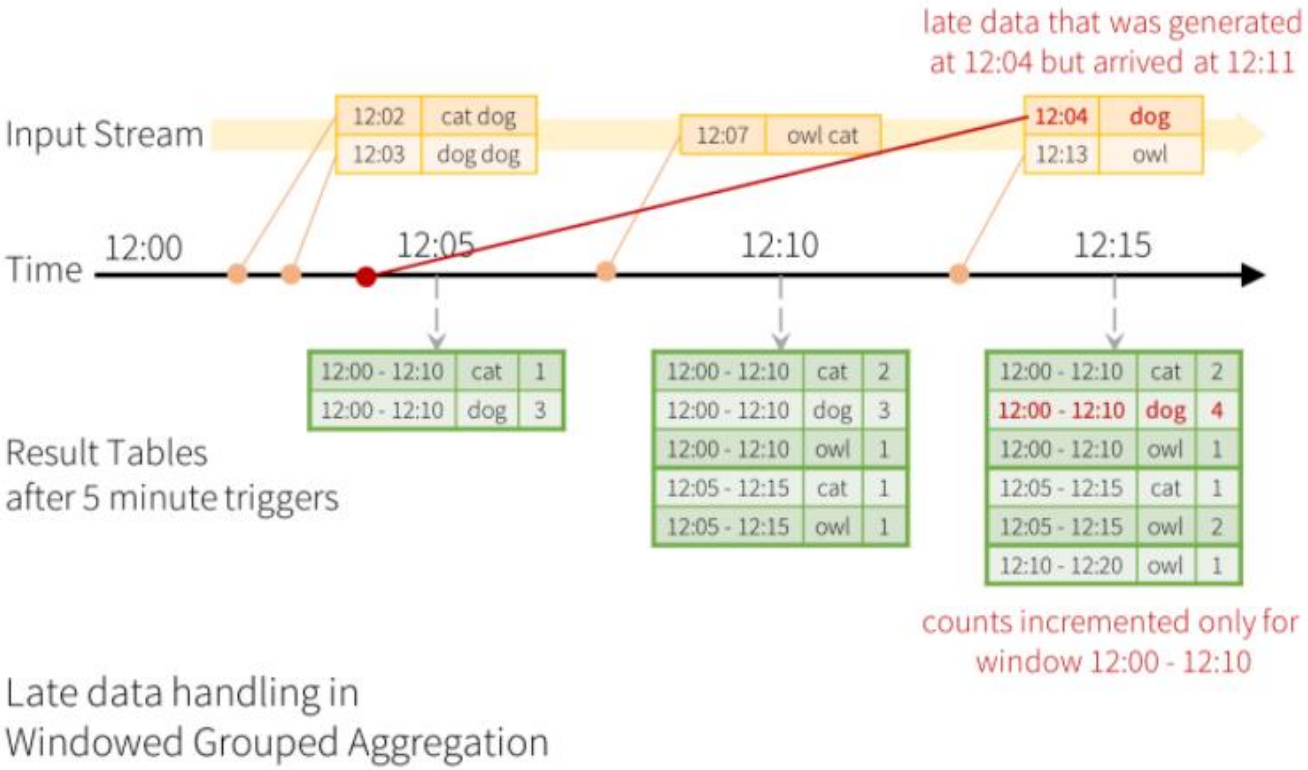
- Taille (ou durée) de la fenêtre : 10 min
- Décalage : 5 min



Ordre d'arrivée des données

- Hypothèse :
  - L'ordre d'arrivée peut être différent de l'ordre obtenu en triant les données par date croissante (l'attribut date servant d'estampille temporelle).
    - date d'arrivée ≠ date du tuple
    - les données peuvent arriver « en retard » ou « en avance »
- Problème :
  - Peut-on garantir la complétude des résultats sur des flux potentiellement infinis ?

Illustration du problème

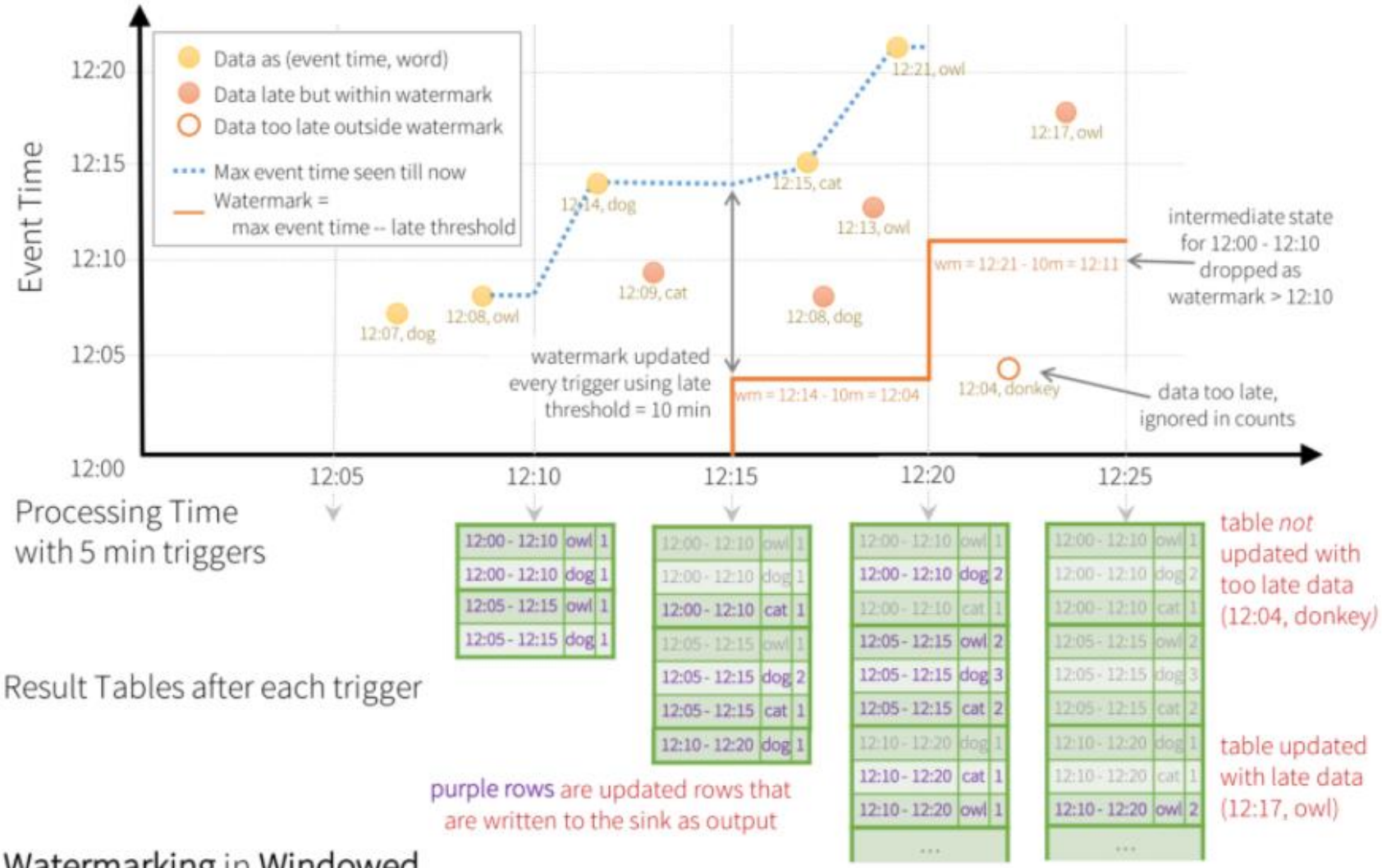


Borner le flux pour ignorer les données trop tardives ?

Flux borné : Watermaking

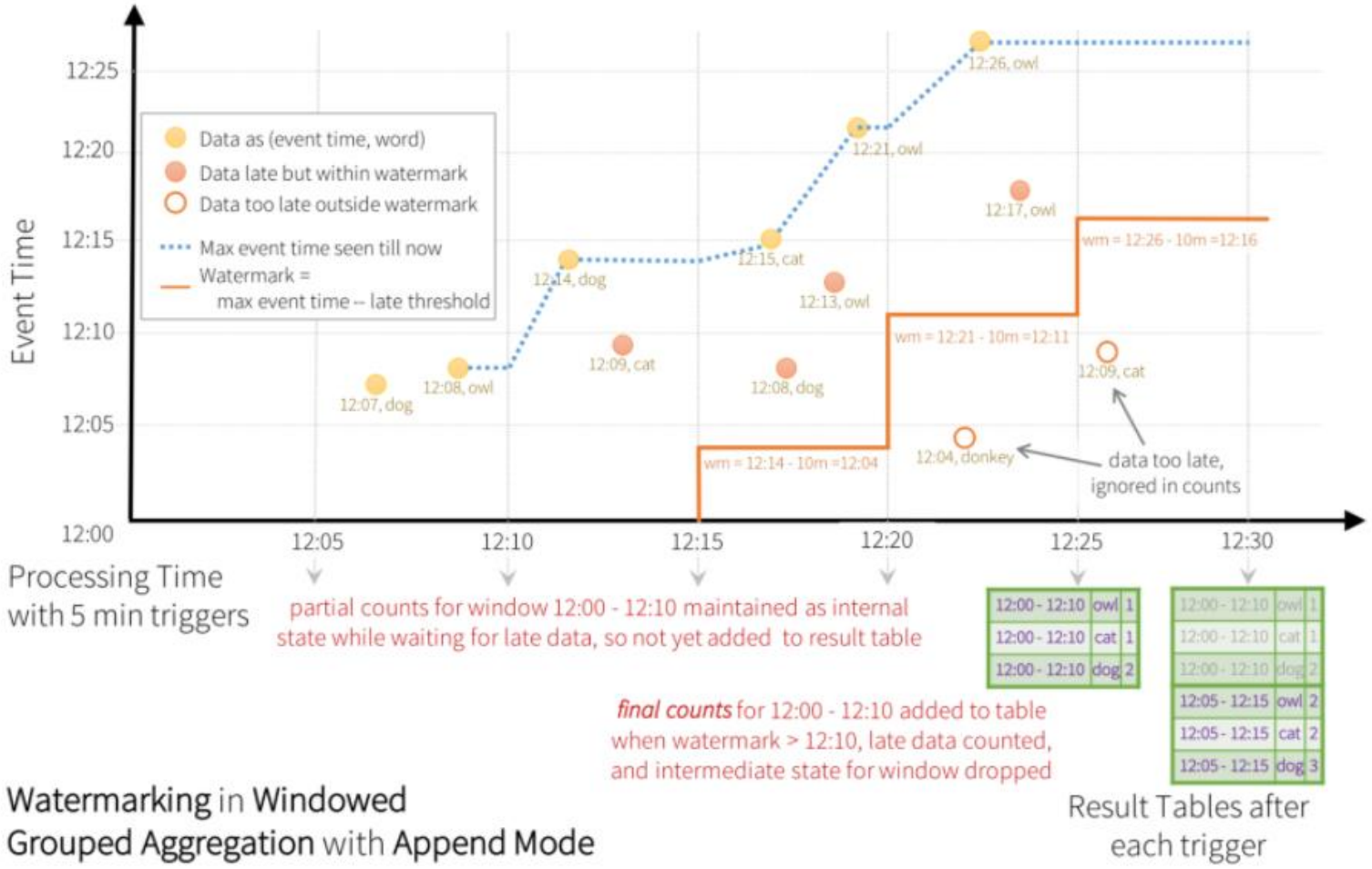
- Solution : spécifier une **contrainte** sur la date d'événement par rapport à la date courante
  - **Retard toléré**
    - **taille** de l'intervalle de **validité** d'un flux
  - Date max = valeur max parmi les tuples déjà arrivés
    - c'est la borne supérieure de l'intervalle de validité
    - peut être supérieure à la date courante…
  - Condition requise
    - date tuple > date max – retard toléré
- Syntaxe : WITH WATERMARK attribut, retard
  - attribut de type date
  - retard toléré

Illustration du watermarking (1/2)



Watermarking in **Windowed Grouped Aggregation** with **Update Mode**

Illustration du watermarking (2/2)

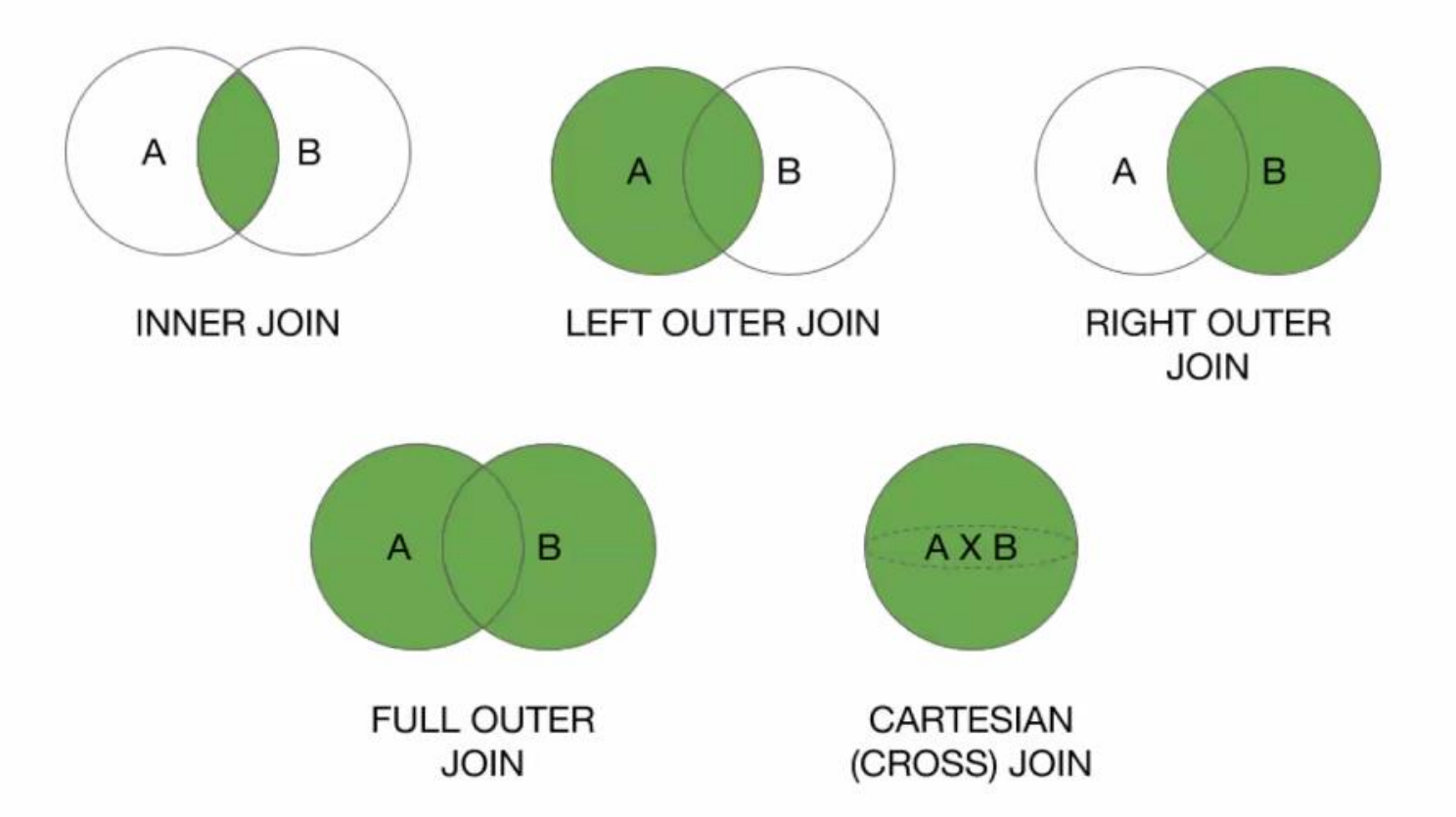


Jointure de flux

- Problème:
  - A la date courante, est-ce qu'on a assez d'information pour déterminer le résultat de la jointure ?
    - Peut-on joindre les tuples qui viennent d'arriver?

- Plusieurs aspects à considérer
  - Dynamicité :
    - Jointure entre 1 flux et une table
    - Jointure entre 2 flux
  - Type de jointure
    - Jointure standard (innerjoin)
    - Jointure externe

Rappel sur les jointures externes

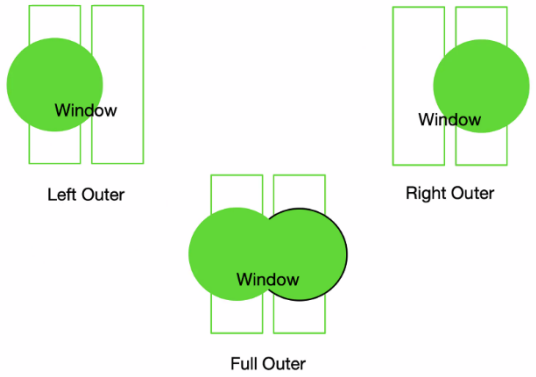


Jointure entre flux et table

- Jointure (innerjoin) possible
  - Chaque tuple du flux peut être comparé avec ceux de la table
- Jointure externe avec le flux : possible
  - le résultat contient un tuple du flux sans correspondance avec la table
- Impossible de calculer une jointure externe
  - On ne sais pas si un tuple de la table va joindre ou non avec les prochains tuples du flux

Jointure entre flux et flux

- Watermaking nécessaire
  - Préciser l'intervalle de tolérance sur chaque flux



Source : EDBT 2020 Tutorial: Declarative Languages for Big Streaming Data

Extensibilité

- Fonction définie par l'utilisateur
- Evaluation incrémentale sur des fenêtre avec recouvrement
  - **Etat à maintenir** entre deux évaluations consécutives de la même requête

Biblio et perspectives

- Spark Structured Streaming
  - (rappel) article SIGMOD 2018 : Structured Streaming
  - Programming guide
    - https://spark.apache.org/docs/latest/structured-streaming-programming-guide.html
- Académique :
  - VLDB 2015 : Google DataFlow Model
  - BigData 2018 : BigSR: real-time expressive RDF stream reasoning on modern Big Data platforms
    - https://ieeexplore.ieee.org/document/8621947
  - EDBT 2020 Tutorial: Declarative Languages for Big Streaming Data
    - https://openproceedings.org/2020/conf/edbt/paper_T1.pdf