

# Przetwarzanie strumieni danych w systemach Big Data

część 4 – dualizm strumieni i tabel, SQL

Krzysztof Jankiewicz

# Plan

- Wyższe poziomy abstrakcji
- Narracje
- Fundamenty
- Perspektywy materializowane
- Dualizm strumieniowo tabelowy
- Typy strumieni
- Znaczenie metadanych

# Wyższe poziomy abstrakcji

- Przetwarzanie strumieni danych – podstawowy poziom abstrakcji
  - traktowanych jako nieskończona kolekcja (sekwencja) zdarzeń
  - za pomocą API przypominającego przetwarzanie kolekcji
- Wyższe poziomy abstrakcji z reguły opierają się na podejściu "relacyjnym"
  - strumień traktowany jest jako strumień poleceń DML
    - operacji wstawiania danych – nowe dane
    - operacji aktualizacji danych – najnowsze wersje danych
  - przykłady
    - TableAPI – oparty na metodach odpowiadających klauzulom SQL
    - SQL – całkowicie oparty na "variantach" poleceń SQL, w których występują zarówno ograniczenia jak i rozszerzenia standardu SQL (wynikające głównie z natury danych)

# Narracje

- Wyższe poziomy abstrakcji funkcjonują na poziomie dwóch światów
  - strumieni
  - tabel
- Stosowane narracje
  - Apache Kafka Streams
    - Strumienie jako *record stream* – każdy wiersz to nowa dana
    - Tabele jako *changelog stream* – aktualizacje na poziomie klucza
  - Apache Flink
    - tabele w bazie danych utrzymywane są na podstawie strumienia poleceń DML - *changelog stream*
    - perspektywy materializowane definiowane są za pomocą poleceń SQL. Ich aktualizacja wymaga ciągłej analizy (wykonywania zapytania) i przetwarzania strumienia zmian z tabel bazowych
    - zmiany w tabelach tworzą strumień poleceń DML

# Fundamenty

- Mechanizmy łączące tabele i strumienie zmian nie powstały wraz systemami przetwarzania strumieni danych
- Dzienniki powtórzeń (*redo logs*) – przykłady:
  - Relacyjne bazy danych
  - HDFS (*EditLog*)
  - Redis (*append-only file*)
- Mechanizmy replikacji danych
- Mechanizmy CDC (*change data capture*)
  - Typy
    - Log-based CDC
    - Trigger-based CDC
    - ETL-based CDC
    - API-based CDC
  - Przykłady implementacji
    - Oracle CDC - Oracle GoldenGate
    - MongoDB CDC - MongoDB Change Streams
    - Microsoft CDC - Microsoft SQL Server CDC
    - Debezium (np. dla bazy danych PostgreSQL, Oracle, MySQL, Cassandra, MongoDB)

# Perspektywy materializowane

- Perspektywa – polecenie SQL
- Perspektywa materializowana (MV)
  - polecenie SQL
  - zmaterializowany wynik – tabela
- Zmiany w tabelach bazowych => nieaktualna perspektywa materializowana
- Sposoby odświeżania
  - pełne (COMPLETE) – zapytanie SQL odbudowuje pełną zawartość MV
  - przyrostowe (FAST) – aktualizuje fragmenty MV na podstawie zmian w tabelach bazowych
- Odświeżanie przyrostowe wymaga wiedzy na temat zmian w tabelach bazowych np. logów perspektyw materializowanych
- Relacje pomiędzy tabelami a strumieniami
  - Log MV – strumień utrzymywany na podstawie zmian w tabeli
  - MV – tabela utrzymywana na podstawie strumienia zmian

# Oracle – log perspektywy materializowanej

```
CREATE TABLE example_table (  
  id NUMBER PRIMARY KEY,  
  name VARCHAR2(50),  
  age NUMBER  
);
```

```
INSERT INTO example_table (id, name,  
VALUES (1, 'John', 25);
```

```
INSERT INTO example_table (id, name,  
VALUES (2, 'Jane', 30);
```

```
INSERT INTO SELECT * FROM MLOG$_example_table;
```

```
UPDATE example_table  
SET age = 26 WHERE id = 1;
```

```
DELETE FROM example_table  
WHERE id = 3;
```

```
INSERT INTO example_table  
(id, name, age)  
VALUES (4, 'Kate', 35);
```

ID	NAME	AGE	M_ROW\$\$	SNAPTIME\$\$	DMLTYPE\$\$	OLD_NEW\$\$	CHANGE_VECTOR\$\$	XID\$\$
1	John	25	AKJBV6ADCAAAACUAAA	01-JAN-00	U	U	08	8974266703940797171
3	Bob	40	AKJBV6ADCAAAACUAAC	01-JAN-00	D	O	00	8974266703940797171
4	Kate	35	AKJBV6ADCAAAACWAAA	01-JAN-00	I	N	FE	8974266703940797171

# Dualizm strumieniowo tabelowy

- Domyślnie strumień (bez dodatkowych metadanych) może być traktowany jako strumień operacji wstawienia (*record stream*)

taxi	godzina	dzielnica	akcja
taxi1	18:00	Wilda	start
taxi2	18:10	Łazarz	start
taxi1	18:15	Piątkowo	stop
taxi3	18:20	Jeżyce	start

- Na podstawie tego strumienia można utrzymywać tabele np.:

- ostatni stan
- liczba akcji

taxi	ostatni stan
taxi1	wolna
taxi2	zajęta
taxi3	zajęta

taxi	liczba akcji
taxi1	2
taxi2	1
taxi3	1

- Tabele

- utrzymują stan (*stateful*),
- często są wynikiem agregacji

- Zmiany w tabelach mogą być źródłem strumieni.

Strumienie takie reprezentują zmiany (*changelog stream*) – to oznacza, że ich interpretacja podczas przetwarzania takiego strumienia powinna być inna

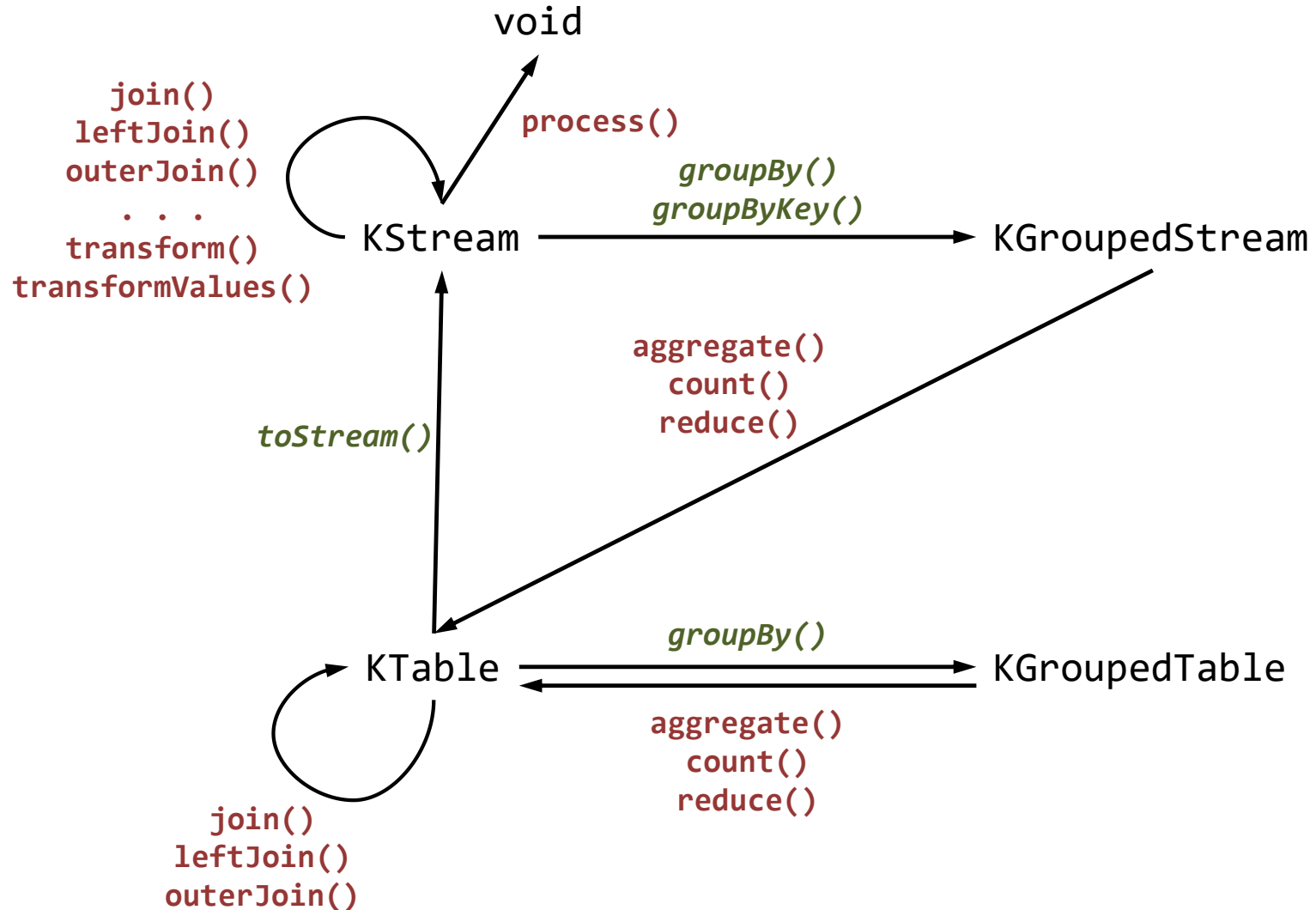


# Kafka a tabele

- Apache Kafka posiada specjalny scalony typ tematu (*compacted topics*), który utrzymuje pojedynczą wartość dla danego klucza
- Apache Kafka może implementować takie tematy przy wykorzystaniu wbudowanej bazy danych *in-memory* RocksDB opartej na modelu klucz-wartość.

```
kafka-topics.sh --create \  
  --bootstrap-server ${CLUSTER_NAME}-w-0:9092 \  
  --replication-factor 2 --partitions 3 --topic kafka-compacted \  
  --config cleanup.policy=compact
```

# Kafka Streams



# Kafka – ksqlDB

```
CREATE STREAM scoreEvents (house VARCHAR, character VARCHAR,  
                           score INT, ts VARCHAR)  
  WITH (kafka_topic='kafka-input', value_format='json',  
        partitions=3);
```









```
CREATE TABLE latest_scores_tab AS  
select house, LATEST_BY_OFFSET(character) character,  
          LATEST_BY_OFFSET(score) score,  
          LATEST_BY_OFFSET(ts) ts  
from scoreEvents  
group by house
```

# Flink

```
%flink.sql
CREATE OR REPLACE TEMPORARY TABLE source_table (
  `code` VARCHAR(10),
  `value` INT,
  `ts` TIMESTAMP(3),
  `pt` AS PROCTIME(),
  WATERMARK FOR `ts` AS `ts` - INTERVAL '1' SECOND
)
WITH (
  'connector' = 'faker',
  'number-of-rows' = '20',
  'rows-per-second' = '1',
  'fields.code.expression' = '#{number.numberBetween ''51'', ''54''}',
  'fields.value.expression' = '#{number.numberBetween ''1'', ''9''}',
  'fields.ts.expression' = '#{date.next ''2'' ''SECONDS''}';
);
```

FLINK JOB FINISHED

```
%flink.sql
SELECT `code`, sum(`value`) as sum_val, count(*) as how_many
FROM source_table
GROUP BY `code`;
```

 settings

code	sum_val	how_many
52	66	17
53	9	3

# Typy strumieni

- Znamy już dwa typy strumieni *record* i *changelog*
- Operując na mechanizmach znanych z Apache Flink powiedzielibyśmy, o dwóch typach strumieni
  - **Strumień tylko wstawiający (*append-only stream*)** – dynamiczna tabela obsługiwana tylko i wyłącznie za pomocą zmian INSERT odwzorowywana jest na strumień składający się z tylko wstawianych wierszy.
  - **Strumień z aktualizacjami (*upsert stream*)** – zawiera dwa typy komunikatów:
    - **aktualizujące** (*upsert messages*) oraz
    - **usuwające** (*delete messages*).
  - Dynamiczna tabela konwertowana na strumień ze aktualizacjami wymaga unikalnego klucza. W takim przypadku
    - zmiany INSERT oraz UPDATE generują komunikaty **aktualizujące**,
    - natomiast zmiany wynikające z DELETE generują komunikaty **usuwające**.
  - Odbiorca takiego strumienia powinien mieć wiedzę na temat klucza, aby w sposób poprawny interpretować i obsługiwać pojawiające się w strumieniu komunikaty.

# Typy strumieni

- W Apache Flink mamy do czynienia z jeszcze jednym typem strumieni
  - **Strumień z wycofaniami (*retract stream*)** – zawiera dwa typy komunikatów/zdarzeń
    - ***dodających*** (*add messages*) i
    - ***wycofujących*** (*retract messages*).

Tabela dynamiczna jest konwertowana na strumień z wycofaniami generując

- ***dodające*** komunikaty dla każdej operacji INSERT,
- ***wycofujące*** komunikaty dla operacji DELETE
- operacja UPDATE generuje natomiast ***wycofujący*** komunikat dla poprzedniego stanu oraz ***dodający*** komunikat dla nowego stanu.

# Metadane

- Tworzenie strumieni *changelog* zwykle ma miejsce w oparciu o tabele dokonujące agregacji.
- Tabele takie określając klucz (GROUP BY, PRIMARY KEY) wprowadzają dodatkowe informacje do utrzymywanej tabeli
- Znajomość tych metadanych jest kluczowa przy poprawnej interpretacji danych pojawiających się w strumieniu
  - system przetwarzania strumieni danych – kolejny krok
  - zewnętrzny odbiorca – ujście