Query Real-Time Kafka Streams with Oracle SQL

Melli Annamalai
Senior Principal Product Manager
Oracle
October 23, 2018
melliyal.Annamalai@oracle.com
Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle’s products may change and remains at the sole discretion of Oracle Corporation.
Agenda

• Demo: Monitor temperature readings from food distribution centers
• Apache Kafka Concepts
• SQL Connector for Kafka
  – Database objects
  – SQL syntax examples
  – Performance and scale

• Roadmap
Streaming Data

In real-time: Process data streams as they occur
Streaming Data Examples

• Sensor data
• Clickstream data
• Machine/equipment
• Network monitoring
• Pipeline monitoring
• Fleet monitoring
• ......
Demo
Monitoring Temperature at Food Distribution Centers
Apache Kafka Concepts
Apache Kafka

Producers
Entities producing streaming data

Kafka Cluster
Stores and manages streaming data in a distributed, replicated, fault-tolerant cluster

Consumers
Applications that read and process messages
Kafka Topics

Messages with a common format

Kafka Cluster
Stores and manages streaming data in a distributed, replicated, fault-tolerant cluster

Producers
Entities producing streaming data

Consumers
Applications that read and process messages
Kafka Topic Example

MSGNUMBER,
MSGTIMESTAMP,
SENSORUNITID,
SENSORTYPEID,
TEMPERATURESETTING,
TEMPERATUREREADING

1293847, 2018-02-02 00:02:09, 804, 122, 68557, 34, 000, 32, 618
1293849, 2018-02-02 00:02:10, 226, 129, 65144, 34, 000, 33, 282
1293851, 2018-02-02 00:02:10, 104, 170, 39436, 34, 000, 33, 809
1293853, 2018-02-02 00:02:09, 459, 104, 9223, 34, 000, 31, 624
1293855, 2018-02-02 00:02:10, 222, 127, 55628, 34, 000, 31, 622
1293857, 2018-02-02 00:02:10, 392, 183, 2986, 34, 000, 34, 273
1293859, 2018-02-02 00:02:10, 279, 166, 34301, 34, 000, 32, 546
1293861, 2018-02-02 00:02:09, 433, 162, 34858, 34, 000, 34, 091
1293863, 2018-02-02 00:02:09, 792, 149, 81568, 34, 000, 36, 590
1293865, 2018-02-02 00:02:10, 013, 163, 20631, 34, 000, 35, 723
1293868, 2018-02-02 00:02:09, 948, 170, 94548, 34, 000, 34, 111
1293869, 2018-02-02 00:02:10, 018, 174, 29056, 34, 000, 36, 404
1293880, 2018-02-02 00:02:10, 813, 161, 71207, 34, 000, 32, 124
1293882, 2018-02-02 00:02:09, 398, 121, 93751, 34, 000, 32, 565
1293886, 2018-02-02 00:02:09, 389, 168, 59378, 34, 000, 32, 172
1293888, 2018-02-02 00:02:412, 114, 79177, 34, 000, 31, 725
1293890, 2018-02-02 00:02:09, 829, 144, 81195, 34, 000, 35, 618
1293892, 2018-02-02 00:02:10, 004, 182, 58496, 34, 000, 31, 415
1293894, 2018-02-02 00:02:10, 105, 114, 62482, 34, 000, 33, 363
Kafka Topic Partitions

Subset of messages in a Topic, managed by a broker

Kafka Cluster
Stores and manages streaming data in a distributed, replicated, fault-tolerant cluster

Producers
Entities producing streaming data

Consumers
Applications that read and process messages
Kafka Topic Partitions

Producers
Entities producing streaming data

Kafka Cluster
Stores and manages streaming data in a distributed, replicated, fault-tolerant cluster

Consumers
Applications that read and process messages
A Kafka Consumer Application
Should read all partitions

Kafka Cluster
Stores and manages streaming data in a distributed, replicated, fault-tolerant cluster

Producers
Entities producing streaming data

Consumers
Applications that read and process messages
Kafka Topic Partitions

Producers
Entities producing streaming data

Kafka Cluster
Stores and manages streaming data in a distributed, replicated, fault-tolerant cluster

Consumers
Applications that read and process messages
Kafka Topic Partition Offsets

Logical Position of a Message within a Partition

Kafka Cluster
Stores and manages streaming data in a distributed, replicated, fault-tolerant cluster

Producers
Entities producing streaming data

Consumers
Applications that read and process messages
Logical Position (Offset) Maps to Sequence Numbers

- Offset uniquely identifies a message
- Consumers can read from an Offset
- Starting offset is 0 (at the beginning of a topic)
Using Data from Kafka Topics

- Kafka Consumer application
- Kafka Streams API
- Stream Kafka topic data into HDFS/Object store/databases using Kafka connectors
- KSQL: Streaming SQL engine for real-time data processing of Kafka topics
Kafka Consumer Example
With Automatic Offset Commits

Properties props = new Properties();
props.put("bootstrap.servers", "localhost:9092");
props.put("group.id", "test");
props.put("enable.auto.commit", "true");
props.put("auto.commit.interval.ms", "1000");
props.put("key.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
props.put("value.deserializer", "org.apache.kafka.common.serialization.StringDeserializer");
KafkaConsumer<String, String> consumer = new KafkaConsumer<>(props);
consumer.subscribe(Arrays.asList("foo", "bar"));
while (true) {
  ConsumerRecords<String, String> records = consumer.poll(100);
  for (ConsumerRecord<String, String> record : records)
    System.out.printf("offset = %d, key = %s, value = %s\n", record.offset(), record.key(), record.value());
}

Oracle Database as a Kafka Consumer

Enable Oracle SQL access to Kafka Topics

Producers
Entities producing streaming data

Kafka Cluster
Stores and manages streaming data in a distributed, replicated, fault-tolerant cluster

Oracle Database
External tables and views
Oracle SQL and Kafka

1. Integrate Kafka messages with Oracle Database applications

2. Enrich Kafka data with Oracle Database table data
   – Feed enriched topics back to Kafka
1. Integrate Kafka with Database Applications

• Query Kafka messages
  – Integrate and analyze with Oracle Database data
  – Use the full richness of Oracle SQL

• Join data in a Kafka time interval with an Oracle Database table

• Load into Oracle Database table using Oracle SQL
2. Enrich Kafka Data with Oracle Database Table Data

34.3, Seattle, LastOutageDate; 36.2, Austin, LastOutageDate; 34.2, Philadelphia, LastOutageDate
SQL Connector for Kafka
Apache Kafka + Oracle Database

Kafka Cluster

Partition 1
Partition 2
Partition 3
Partition 4

Database applications query views to access Kafka messages

External tables
Views on the external tables
Database Objects

• Views over Kafka Topics

• Views are created per Topic

• There can be multiple sets of views per Topic: One set per application (Consumer Group)

<table>
<thead>
<tr>
<th>Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>KAFKA$PARTITION</td>
<td></td>
<td>NUMBER(38)</td>
</tr>
<tr>
<td>KAFKA$OFFSET</td>
<td></td>
<td>NUMBER(38)</td>
</tr>
<tr>
<td>KAFKA$TIMESTAMP</td>
<td></td>
<td>NUMBER(38)</td>
</tr>
<tr>
<td>MSGNUMBER</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>MSGTIMESTAMP</td>
<td></td>
<td>TIMESTAMP(6)</td>
</tr>
<tr>
<td>SENSORTYPEID</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>SENSORUNITID</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>TEMPERATURESETTING</td>
<td></td>
<td>NUMBER(6,3)</td>
</tr>
<tr>
<td>TEMPERATUREREADING</td>
<td></td>
<td>NUMBER(6,3)</td>
</tr>
</tbody>
</table>
Database Objects

• Metadata tables
  – DBMS_KAFKA_CLUSTER
  – DBMS_KAFKA_PARTITION
  – DBMS_KAFKA_LOAD_METRICS
  – DBMS_KAFKA_APPLICATION

• Message schema table

<table>
<thead>
<tr>
<th>Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>MSGNUMBER</td>
<td>NOT NULL</td>
<td>NUMBER(38)</td>
</tr>
<tr>
<td>MSGTIMESTAMP</td>
<td>NULL</td>
<td>TIMESTAMP(6)</td>
</tr>
<tr>
<td>SENSORTYPEID</td>
<td></td>
<td>NUMBER(38)</td>
</tr>
<tr>
<td>SENSORUNITID</td>
<td></td>
<td>NUMBER(38)</td>
</tr>
<tr>
<td>TEMPERATURESETTING</td>
<td></td>
<td>NUMBER(6,3)</td>
</tr>
<tr>
<td>TEMPERATUREREADING</td>
<td></td>
<td>NUMBER(6,3)</td>
</tr>
</tbody>
</table>
Query Streaming Data with a View

No Need to Persist or Stage Data in a Table

<table>
<thead>
<tr>
<th>Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>KAFKA$PARTITION</td>
<td></td>
<td>NUMBER(38)</td>
</tr>
<tr>
<td>KAFKA$OFFSET</td>
<td></td>
<td>NUMBER(38)</td>
</tr>
<tr>
<td>KAFKA$TIMESTAMP</td>
<td></td>
<td>NUMBER(38)</td>
</tr>
<tr>
<td>MSGNUMBER</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>MSGTIMESTAMP</td>
<td></td>
<td>TIMESTAMP(6)</td>
</tr>
<tr>
<td>SENSORTYPEID</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>SENSORUNITID</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>TEMPERATURESETTING</td>
<td></td>
<td>NUMBER(6,3)</td>
</tr>
<tr>
<td>TEMPERATUREREADING</td>
<td></td>
<td>NUMBER(6,3)</td>
</tr>
</tbody>
</table>
Join Kafka Topic View with Oracle Database Table

<table>
<thead>
<tr>
<th>Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>KAFKA$PARTITION</td>
<td></td>
<td>NUMBER(38)</td>
</tr>
<tr>
<td>KAFKA$OFFSET</td>
<td></td>
<td>NUMBER(38)</td>
</tr>
<tr>
<td>KAFKA$TIMESTAMP</td>
<td></td>
<td>NUMBER(38)</td>
</tr>
<tr>
<td>MSGNUMBER</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>MSGTIMESTAMP</td>
<td></td>
<td>TIMESTAMP(6)</td>
</tr>
<tr>
<td>SENSORTYPEID</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>SENSORUNITID</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>TEMPERATURESETTING</td>
<td></td>
<td>NUMBER(6,3)</td>
</tr>
<tr>
<td>TEMPERATUREREADING</td>
<td></td>
<td>NUMBER(6,3)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Null?</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>SENSORUNITID</td>
<td>NOT NULL</td>
<td>NUMBER(38)</td>
</tr>
<tr>
<td>LATITUDE</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>LONGITUDE</td>
<td></td>
<td>NUMBER</td>
</tr>
<tr>
<td>CITY</td>
<td></td>
<td>VARCHAR2(30)</td>
</tr>
</tbody>
</table>

View on Kafka Topics

Find cities where the temperature fluctuation is more than normal

Database table
Offset Management

34.2 34.5 34.1 34.6 34.3
Offset Management

Commit offset RECORD_OFFSET()
Offsets of each Consumer Group application are managed independently.
SQL Syntax Examples
1. REGISTER_CLUSTER

SQL> BEGIN

    dbms_kafka.register_cluster('SENS2',
    '<Zookeeper URL>:2181',
    '<Kafka broker URL>:9092',
    'DBMSKAFKA_DEFAULT_DIR',
    'DBMSKAFKA_LOCATION_DIR'
    'Testing DBMS_KAFKA');

END;
DECLARE
    views_created INTEGER;
    view_prefix VARCHAR2(128);
BEGIN
    DBMS_KAFKA.CREATE_KAFKA_VIEWS('SENS2',
        'MONITORAPP', 'sensor2', 'sensor_messages_shape',
        views_created, view_prefix);
END;
/

One view per partition
### 3. Query with Oracle SQL

#### SQL Example

```sql
SQL> select count(*) from KV_SENS2_MONITORAPP_SENSOR2_0;
```

```sql
SQL> select timestamp, sensorunitid, temperaturereading from KV_SENS2_MONITORAPP_SENSOR2_0;
```

<table>
<thead>
<tr>
<th>TIMESTAMP</th>
<th>SENSORUNITID</th>
<th>TEMPERATUREREADING</th>
</tr>
</thead>
<tbody>
<tr>
<td>19-OCT-18 11:07:40.545000000 AM US/EASTERN</td>
<td>7</td>
<td>32.893</td>
</tr>
<tr>
<td>19-OCT-18 11:07:40.545000000 AM US/EASTERN</td>
<td>5</td>
<td>33.826</td>
</tr>
<tr>
<td>19-OCT-18 11:07:40.545000000 AM US/EASTERN</td>
<td>0</td>
<td>31.025</td>
</tr>
<tr>
<td>19-OCT-18 11:07:40.545000000 AM US/EASTERN</td>
<td>1</td>
<td>33.771</td>
</tr>
<tr>
<td>19-OCT-18 11:07:40.545000000 AM US/EASTERN</td>
<td>5</td>
<td>33.159</td>
</tr>
<tr>
<td>19-OCT-18 11:07:40.545000000 AM US/EASTERN</td>
<td>3</td>
<td>32.645</td>
</tr>
<tr>
<td>19-OCT-18 11:07:40.545000000 AM US/EASTERN</td>
<td>6</td>
<td>36.065</td>
</tr>
<tr>
<td>19-OCT-18 11:07:40.545000000 AM US/EASTERN</td>
<td>2</td>
<td>32.095</td>
</tr>
<tr>
<td>19-OCT-18 11:07:40.545000000 AM US/EASTERN</td>
<td>6</td>
<td>31.5</td>
</tr>
<tr>
<td>19-OCT-18 11:07:40.545000000 AM US/EASTERN</td>
<td>9</td>
<td>34.972</td>
</tr>
<tr>
<td>19-OCT-18 11:07:30.544000000 AM US/EASTERN</td>
<td>0</td>
<td>32.32</td>
</tr>
<tr>
<td>19-OCT-18 11:07:30.544000000 AM US/EASTERN</td>
<td>1</td>
<td>31.83</td>
</tr>
<tr>
<td>19-OCT-18 11:07:30.544000000 AM US/EASTERN</td>
<td>1</td>
<td>32.093</td>
</tr>
<tr>
<td>19-OCT-18 11:07:30.544000000 AM US/EASTERN</td>
<td>6</td>
<td>32.093</td>
</tr>
</tbody>
</table>

---


Copyright © 2018, Oracle and/or its affiliates. All rights reserved.
Record Offset and Set Offset

SQL> execute dbms_kafka.record_offset('KV_SENS2_MONITORAPP_SENSOR2_0');

SQL> execute dbms_kafka.set_offset('KV_SENS2_MONITORAPP_SENSOR2_0');
Loading into Oracle Database

DECLARE
    rows_loaded number;
BEGIN
    dbms_kafka.load_table('SENS2','LOADAPP','sensor2','sensor_messages_shape',rows_loaded);
    dbms_output.put_line('rows loaded: ' || rows_loaded);
END;
/
Performance and Scale
Scaling in Kafka

• Increase number of Partitions

• Increase Consumer instances
Kafka Consumer Groups

Producers
Entities producing streaming data

Kafka Cluster
Stores and manages streaming data in a distributed, replicated, fault-tolerant cluster

Consumers
Applications that read and process messages

One Consumer reads from all Partitions

Topic with 4 Partitions
**Kafka Consumer Groups**

**Producers**  
Entities producing streaming data

**Kafka Cluster**  
Stores and manages streaming data in a distributed, replicated, fault-tolerant cluster

**Consumers**  
Applications that read and process messages

Topic with 4 Partitions
Kafka Consumer Groups

Producers
Entities producing streaming data

Kafka Cluster
Stores and manages streaming data in a distributed, replicated, fault-tolerant cluster

Database views

One view reads from all Partitions

Topic with 4 Partitions
Kafka Consumer Groups

Producers
Entities producing streaming data

Kafka Cluster
Stores and manages streaming data in a distributed, replicated, fault-tolerant cluster

Database views

Partition 1
Partition 2
Partition 3
Partition 4

Topic with 4 Partitions

Each view reads from two Partitions
Kafka Consumer Groups

Producers
Entities producing streaming data

Kafka Cluster
Stores and manages streaming data in a distributed, replicated, fault-tolerant cluster

Topic with 4 Partitions

Database views

Four views, one per Partition
Kafka Partitions and Database Views

• Number of views can be 1......\textit{total number of partitions}\textgreater
• Scale by increasing the number of views

• Can have multiple sets of views for the same Topic
  – A set of views is a ‘Consumer Group’
  – \textbf{Offsets for each Consumer Group managed independently}
Factors that Impact Performance

• Degree of parallelism in the database

• Network bandwidth
  – Between Kafka brokers and database system
Kafka and Oracle Products and Cloud Services
Oracle Big Data SQL

Kafka Cluster → Hadoop Cluster with Apache Hive → Big Data SQL external table

Shipping since 2017
Query Hive table from Oracle Database

Hive table reads from Kafka

```sql
CREATE TABLE TRAFFIC_JSON_TAB
(
  ID VARCHAR2(256),
  CLIENTID VARCHAR2(256),
  ...),
  EVENTTIMEASSTRING VARCHAR2(256),
  SENDER VARCHAR2(256),
  TYPE VARCHAR2(256),
  PROPERTIES VARCHAR2(256),
  DIRECTION VARCHAR2(256),
  RECEIVEDTIME NUMBER(38,0),
  RECEIVEDTIMEASSTRING VARCHAR2(256),
  PAYLOAD VARCHAR2(4000)
)

ORGANIZATION EXTERNAL
(
  TYPE ORACLE_HIVE DEFAULT DIRECTORY DEFAULT_DIR
  ACCESS PARAMETERS
  (
    com.oracle.bigdata.cluster=bigdatalite
    com.oracle.bigdatatablename=oracletest.traffic_json_tab
  )
)
```
Oracle Event Hub Cloud Service

Enterprise Kafka features in Event Hub
- Scalable and highly available
- Messaging/Event broker
- Security
- Schema registry
- Connectors: object storage, databases, other external systems

Access from Database Cloud Service via Database views
Oracle Event Hub Cloud Service Use Cases
Single Infra for Data Movement, Application Messaging, Events Broker, Stream Analytics

<table>
<thead>
<tr>
<th>Kafka Use cases</th>
<th>Oracle PaaS Service Add-Ons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Movement into data lakes (Firehose) and DB Change propagation (Kafka Connect)</td>
<td>Big Data Cloud</td>
</tr>
<tr>
<td>Application messaging for Microservices; replacement for JMS; Events Broker (Kafka Connect, Kafka Streams)</td>
<td>Integration Cloud</td>
</tr>
<tr>
<td>Event Processing &amp; Analytics (Kafka Streams and KSQL)</td>
<td>Streaming Analytics</td>
</tr>
<tr>
<td>IoT Events and Logs transport (Kafka Produce/Consume, Kafka Connect)</td>
<td>Mobile Cloud</td>
</tr>
</tbody>
</table>

Data Lake Firehose  
Application Messaging  
IoT Use Case  
Database Change Use Case
Roadmap
Planned Release Roadmap

• Addition to Big Data Connectors product suite

• BYOL for Cloud deployments
  – Note: Big Data Connectors are included in Big Data Cloud Service and Big Data Cloud@Customer

• Cloud Managed Services
  – Autonomous Data Warehouse Cloud Service (ADWCS)
Peek into Planned Functionality Roadmap

• Support Kafka Streams
  – Enrich Kafka Topics using Oracle SQL and database tables

• Retrieval based on timestamp information