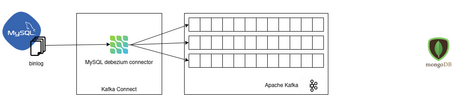
# POC

Se usan imágenes de prueba propuestas por debizium

<https://registry.hub.docker.com/search?q=debezium>

-kafka,zookeeper,conect, example-mysql



Creamos librería con springboot

@KafkaListener(

topics = {"${kafka.topic.product}"},

groupId = "${kafka.group}",

containerFactory = "kafkaListenerContainerFactory"

)

## Version1 mysql-mongo

0.Creamos imagen que vamos a ejecutar después en Docker compose

docker build -t com.poc.cdc/replicator-mongodb:0.0.1-SNAPSHOT .

1 .ejecutamos el docke compose up -d

2 docker ps -->verificamos conector de delbezium puerto 8003

<https://debezium.io/blog/2022/10/06/debezium-oracle-series-part-2/>

<https://debezium.io/blog/2022/09/30/debezium-oracle-series-part-1/>

3.registrar propiedades del conector en yml (springboot-debezium-mongo-replicator\environment\debezium)

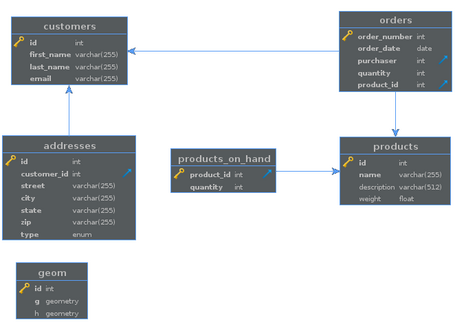
Al inicializar la base de datos MongoDB, esta se crea vacía. Teniendo en cuenta que se quiere replicar la información, y que el conector Debezium detecta los cambios que se producen cuando empieza a monitorizar, **¿qué pasa con los datos ya existentes en la base de datos MySQL? ¿Se tienen que replicar de forma manual a MongoDB? ¿Qué alternativas hay?**

a modo explicativo, el conector Debezium crea una instantánea de la base de datos que va a monitorizar y permite propagar los datos existentes como modificaciones realizadas. Es decir, los datos iniciales en MySQL serán propagados, en forma de evento de creación, a su topic respectivo para cada una de las tablas monitorizadas.

4 mejor creamos script sh->> ./init\_environment.sh

Si todo ha ido bien.

los topics, se han creado con la nomenclatura por defecto de forma automática en Apache Kafka. En este caso, son los siguientes:



**Kafka topic:**

dbserver1 (para cambios de estructura)

dbserver1.inventory.products (tabla products)

dbserver1.inventory.products\_on\_hand (tabla products\_on\_hand)

dbserver1.inventory.customers (tabla customers)

dbserver1.inventory.orders (tabla orders)

dbserver1.inventory.addresses (tabla addresses)

dbserver1.inventory.geom (tabla geom)

Example: Message emitted to the MySQL connector schema change topic

The following example shows a typical schema change message in JSON format. The message contains a logical representation of the table schema.

Prueba

Se va a verificar que ocurre al **actualizar** una fila de la entidad products:

* El estado actual del producto con ID 101 en MySQL.

Primeros pasos con Debezium 9

* El estado actual del producto con ID 101 en MongoDB.

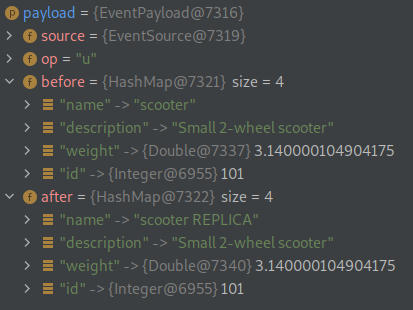
Primeros pasos con Debezium 10

* Se actualiza el nombre del producto ID 101 en MySQL.

UPDATE products SET name='scooter REPLICA' where id=101;

Primeros pasos con Debezium 11

* La aplicación Spring Boot captura el evento de cambio.



* Se actualiza el nombre del producto ID 101 en MongoDB.

Primeros pasos con Debezium 13

## Versión 2 oracle-mongo

Despues de toda la configuración creamos imagen con ella

docker commit -m "oracle configurada" -a "jcallejo" oracle21c oracle21configured

|  |
| --- |
| {  "name": "customers-connector",  "config": {  "connector.class": "io.debezium.connector.oracle.OracleConnector",  "tasks.max": "1",  "database.hostname": "dbz\_oracle21",  "database.port": "1521",  "database.user": "c##dbzuser",  "database.password": "dbz",  "database.dbname": "ORCLCDB",  "database.pdb.name": "ORCLPDB1",  "database.server.name": "server1",  "table.include.list": "C##DBZUSER.CUSTOMERS",  "database.history.kafka.bootstrap.servers": "kafka:9092",  "database.history.kafka.topic": "schema-changes"  }  } |

curl -i -X POST -H "Accept:application/json"-H "Content-Type:application/json" localhost:8083/connectors -d connector-oracle.json

## Versión 1 oracle-oracle

# anexos

## mensaje Kafka generado por connector

|  |
| --- |
| {  "schema": { },  "payload": {  "source": {  "version": "2.2.1.Final",  "connector": "mysql",  "name": "mysql",  "ts\_ms": 1651535750218,  "snapshot": "false",  "db": "inventory",  "sequence": null,  "table": "customers",  "server\_id": 223344,  "gtid": null,  "file": "mysql-bin.000003",  "pos": 570,  "row": 0,  "thread": null,  "query": null  },  "databaseName": "inventory",  "schemaName": null,  "ddl": "ALTER TABLE customers ADD middle\_name varchar(255) AFTER first\_name",  "tableChanges": [  {  "type": "ALTER",  "id": "\"inventory\".\"customers\"",  "table": {  "defaultCharsetName": "utf8mb4",  "primaryKeyColumnNames": [  "id"  ],  "columns": [  {  "name": "id",  "jdbcType": 4,  "nativeType": null,  "typeName": "INT",  "typeExpression": "INT",  "charsetName": null,  "length": null,  "scale": null,  "position": 1,  "optional": false,  "autoIncremented": true,  "generated": true  },  {  "name": "first\_name",  "jdbcType": 12,  "nativeType": null,  "typeName": "VARCHAR",  "typeExpression": "VARCHAR",  "charsetName": "utf8mb4",  "length": 255,  "scale": null,  "position": 2,  "optional": false,  "autoIncremented": false,  "generated": false  },  {  "name": "middle\_name",  "jdbcType": 12,  "nativeType": null,  "typeName": "VARCHAR",  "typeExpression": "VARCHAR",  "charsetName": "utf8mb4",  "length": 255,  "scale": null,  "position": 3,  "optional": true,  "autoIncremented": false,  "generated": false  },  {  "name": "last\_name",  "jdbcType": 12,  "nativeType": null,  "typeName": "VARCHAR",  "typeExpression": "VARCHAR",  "charsetName": "utf8mb4",  "length": 255,  "scale": null,  "position": 4,  "optional": false,  "autoIncremented": false,  "generated": false  },  {  "name": "email",  "jdbcType": 12,  "nativeType": null,  "typeName": "VARCHAR",  "typeExpression": "VARCHAR",  "charsetName": "utf8mb4",  "length": 255,  "scale": null,  "position": 5,  "optional": false,  "autoIncremented": false,  "generated": false  }  ],  "attributes": [  {  "customAttribute": "attributeValue"  }  ]  }  }  ]  }  } |

| Table 1. Descriptions of fields in messages emitted to the schema change topic |
| --- |

|  |  |  |
| --- | --- | --- |
|  |  |  |
| **Item** | **Field name** | **Description** |
| 1 | source | The source field is structured exactly as standard data change events that the connector writes to table-specific topics. This field is useful to correlate events on different topics. |
| 2 | ts\_ms | Optional field that displays the time at which the connector processed the event. The time is based on the system clock in the JVM running the Kafka Connect task.  In the source object, ts\_ms indicates the time that the change was made in the database. By comparing the value for payload.source.ts\_ms with the value for payload.ts\_ms, you can determine the lag between the source database update and Debezium. |
| 3 | databaseName schemaName | Identifies the database and the schema that contains the change. The value of the databaseName field is used as the message key for the record. |
| 4 | ddl | This field contains the DDL that is responsible for the schema change. The ddl field can contain multiple DDL statements. Each statement applies to the database in the databaseName field. Multiple DDL statements appear in the order in which they were applied to the database.  Clients can submit multiple DDL statements that apply to multiple databases. If MySQL applies them atomically, the connector takes the DDL statements in order, groups them by database, and creates a schema change event for each group. If MySQL applies them individually, the connector creates a separate schema change event for each statement. |
| 5 | tableChanges | An array of one or more items that contain the schema changes generated by a DDL command. |
| 6 | type | Describes the kind of change. The value is one of the following:  CREATE  Table created.  ALTER  Table modified.  DROP  Table deleted. |
| 7 | id | Full identifier of the table that was created, altered, or dropped. In the case of a table rename, this identifier is a concatenation of *<old>*,*<new>* table names. |
| 8 | table | Represents table metadata after the applied change. |
| 9 | primaryKeyColumnNames | List of columns that compose the table’s primary key. |
| 10 | columns | Metadata for each column in the changed table. |
| 11 | attributes | Custom attribute metadata for each table change. |

# Configurar contenedor Oracle

### Descargar contenedor

<https://docs.oracle.com/en/database/oracle/oracle-database/21/deeck/index.html#DEEDK-GUID-E0F4869F-67A8-4624-8247-D3042BE391F2>

La imagen Docker de Oracle Database Enterprise Edition contiene Oracle Database 21c Enterprise Edition, con la opción de implementar Enterprise Edition o Standard Edition, **ejecutándose en Oracle Linux 7 (x86-64)**. Esta imagen contiene una base de datos predeterminada en una configuración multi-tenant (multi clinte), con una base de datos conectable.

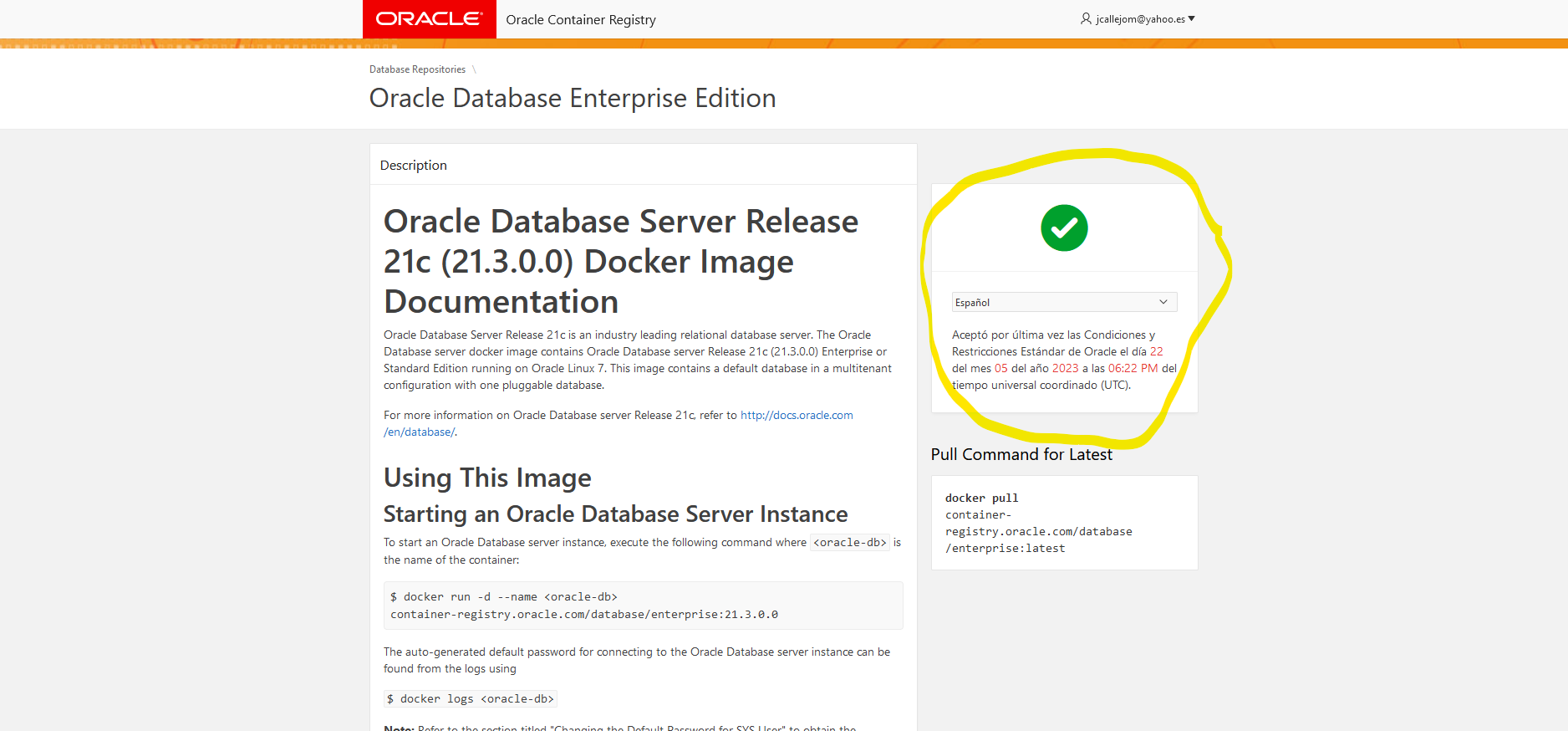
Para comenzar, necesitaremos un entorno de base de datos Oracle en ejecución. Una de las formas más sencillas de hacer esto es usar Docker mediante la implementación de un contenedor que ejecute la base de datos de Oracle. **Oracle ha publicado dichos contenedores en su registro de contenedores** , lo que permite que cualquiera pueda ejecutar la base de datos y probarla. Se puede omitir esta si tiene la intención de incorporar cambios de una base de datos Oracle existente.

<https://container-registry.oracle.com/>

|  |  |
| --- | --- |
|  | Todas las imágenes de contenedores de Oracle están preconstruidas utilizando Oracle con multiusuario. Esto significa que seguiremos las configuraciones de acuerdo con una arquitectura multiusuario. Es posible que se necesiten algunos ajustes menores si está utilizando una instalación que no utiliza multiusuario. |

Para este ejercicio, vamos a utilizar esta imagen de contenedor

PREVIO hay que aceptar la licencia por imagen sino aunque hagamos login no dejara bajarla. ORACLE



docker login container-registry.oracle.com

docker pull [container-registry.oracle.com/database/enterprise:latest](http://container-registry.oracle.com/database/enterprise:latest)

|  |
| --- |
| docker run -d --name <container\_name> \  -p <host\_port>:1521 -p <host\_port>:5500 \  -e ORACLE\_SID=<your\_SID> \  -e ORACLE\_PDB=<your\_PDBname> \  -e ORACLE\_PWD=<your\_database\_password> \  -e INIT\_SGA\_SIZE=<your\_database\_SGA\_memory\_MB> \  -e INIT\_PGA\_SIZE=<your\_database\_PGA\_memory\_MB> \  -e ORACLE\_EDITION=<your\_database\_edition> \  -e ORACLE\_CHARACTERSET=<your\_character\_set> \  -e ENABLE\_ARCHIVELOG=true \  -v [<host\_mount\_point>:]/opt/oracle/oradata \  container-registry.oracle.com/database/enterprise:21.3.0.0  Parameters:  --name: The name of the container (default: auto generated  -p: The port mapping of the host port to the container port.  Two ports are exposed: 1521 (Oracle Listener), 5500 (OEM Express)  -e ORACLE\_SID: The Oracle Database SID that should be used (default:ORCLCDB)  -e ORACLE\_PDB: The Oracle Database PDB name that should be used (default: ORCLPDB1)  -e ORACLE\_PWD: The Oracle Database SYS, SYSTEM and PDBADMIN password (default: auto generated)  -e INIT\_SGA\_SIZE: The total memory in MB that should be used for all SGA components (optional)  -e INIT\_PGA\_SIZE: The target aggregate PGA memory in MB that should be used for all server processes attached to the instance (optional)  -e ORACLE\_EDITION: The Oracle Database Edition (enterprise/standard, default: enterprise)  -e ORACLE\_CHARACTERSET: The character set to use when creating the database (default: AL32UTF8)  -e ENABLE\_ARCHIVELOG: To enable archive log mode when creating the database (default: false). Supported 19.3 onwards.  -v /opt/oracle/oradata  The data volume to use for the database. Has to be writable by the Unix "oracle" (uid: 54321) user inside the container  If omitted the database will not be persisted over container recreation.  -v /opt/oracle/scripts/startup | /docker-entrypoint-initdb.d/startup  Optional: A volume with custom scripts to be run after database startup.  For further details see the "Running scripts after setup and on  startup" section below.  -v /opt/oracle/scripts/setup | /docker-entrypoint-initdb.d/setup  Optional: A volume with custom scripts to be run after database setup.  For further details see the "Running scripts after setup and on startup" section below. |

Contenedor con tres variables nombre del contenedor user password

User system … password: el del contenedor

docker exec -it oracle21c sqlplus / as sysdba

Si en Docker compose exponemos también este puerto 5500 obtenemos un cliente Oracle

<https://localhost:5500/em>

modificación de contenedor

### Configurar Oracle: archived logs

Oracle guarda grupos completos de registros de archived (los registros de transacciones de la base de datos) en uno o más destinos fuera de línea, conocidos colectivamente como el registro de rehacer archivado o los registros de archivo. Los cambios principales en una base de datos principal se replican en entornos de reserva lógicos o físicos mediante registros de archivo.

Un cambio de registro ocurre cuando un registro de rehacer se llena y se archiva. Debezium ingiere los cambios en los registros de rehacer y archivar. Debezium requiere acceso al registro de archivo para terminar de procesar las entradas de rehacer cuando se archiva el registro de rehacer.

La imagen de registro del contenedor de Oracle utilizada en la sección [Instalar Oracle](https://debezium-io.translate.goog/blog/2022/09/30/debezium-oracle-series-part-1/?_x_tr_sl=auto&_x_tr_tl=es&_x_tr_hl=es#install_oracle) no tiene habilitado el registro de archivo. Si usa otra imagen o un entorno preexistente, debe verificar si el registro de archivo está habilitado. Para verificar el estado, use el SYS usuario y la contraseña definida durante la instalación para ORACLE\_PWD conectarse a la ORCLCDBbase de datos y ejecutar la siguiente consulta:

SELECT LOG\_MODE FROM V$DATABASE;

Si la columna contiene ARCHIVELOG, el registro de archivo está habilitado. Si la columna contiene el valor NOARCHIVELOG, el registro de archivo no está habilitado y es necesaria una configuración adicional.

Al configurar el registro de archivo de Oracle, no solo debemos habilitar la función de registro, sino que también debemos especificar una ubicación en el disco para almacenar los registros. Si está utilizando un entorno preexistente, deberá consultarlo con el administrador de su base de datos. La mayoría de los servidores de bases de datos almacenan archivos de registro de archivado utilizando rutas especiales, y necesitará saber si se utiliza Oracle Automatic Storage Management (ASM) o qué volumen tiene espacio adecuado en el servidor de la base de datos.

Abramos una terminal al contenedor de la base de datos de Oracle. Queremos conectarnos al contenedor de la base de datos usando SQL\*Plus para usar un cliente que permita desmontar y reiniciar fácilmente la base de datos. Entonces, en una nueva terminal, ejecute:

docker exec -it dbz\_oracle21 -e ORACLE\_SID=ORCLCDB sqlplus sys as sysdba

docker exec -e ORACLE\_SID=ORCLCDB -it oracle21c sqlplus sys as sysdba

|  |  |
| --- | --- |
|  | Si se conecta a un entorno de Oracle existente, también puede conectarse ssh al shell del servidor de la base de datos para ejecutar SQL\*Plus. |

El comando anterior iniciará SQL\*Plus de Oracle, un cliente SQL de línea de comandos de Oracle. El cliente le pedirá su contraseña, que será la misma que la ORACLE\_PWDvariable de entorno o oraclepwsi está utilizando el contenedor de registro de Oracle. Utilice la contraseña de usuario de su entorno SYScuando se conecte a un entorno de Oracle existente.

Necesitamos establecer dos parámetros de la base de datos:

db\_recovery\_file\_dest\_size

El número de bytes disponibles para almacenar registros de archivo.

|  |  |
| --- | --- |
|  | Supongamos que el tamaño de los registros de archivo existentes y el siguiente registro que se archivará supera este valor configurado. En ese caso, el proceso del archived de la base de datos de Oracle se bloqueará. Si todos los registros de rehacer requieren archivado y el proceso de archivado está bloqueado, la base de datos evita cambios hasta que el proceso de archivado se desbloquee. La eliminación de registros de archivo antiguos con la RMANutilidad desbloquea el proceso de archivado, lo que permite archivar cualquier registro de rehacer pendiente. Por lo tanto, generalmente es una buena idea elegir un tamaño decente según la política de retención de su base de datos. |

db\_recovery\_file\_dest

La ubicación en el disco donde se almacenan los registros de archivo.

|  |  |
| --- | --- |
|  | El usuario de la base de datos de Oracle, a menudo denominado Oracle usuario, debe poder leer y escribir en esta ubicación. |

Para establecer estos valores, ejecutaremos los siguientes comandos SQL dentro de la ventana del terminal SQL\*Plus donde ya nos hemos conectado a la base de datos como usuario SYS. Nuevamente, si se está conectando a un entorno preexistente, consulte con el administrador de su base de datos antes de continuar aquí.

ALTER SYSTEM SET db\_recovery\_file\_dest\_size = 10G;

ALTER SYSTEM SET db\_recovery\_file\_dest = '/opt/oracle/oradata/ORCLCDB' scope=spfile;

SHUTDOWN IMMEDIATE

STARTUP MOUNT

ALTER DATABASE ARCHIVELOG;

ALTER DATABASE OPEN;

ARCHIVE LOG LIST;

Las ALTERdeclaraciones anteriores ajustan los parámetros de la base de datos, especificando que la retención de registros de archivo es de hasta un máximo de 10 GB y ahí /opt/oracle/oradata/ORCLCDBes donde se almacenan los registros.

El resultado final de SQL\*Plus debería mostrar lo siguiente:

SQL> Database log mode Archive Mode

Automatic archival Enabled

Archive destination USE\_DB\_RECOVERY\_FILE\_DEST

Oldest online log sequence 1

Next log sequence to archive 3

Current log sequence 3

La configuración de los registros de archivo está completa y, cuando se produce un cambio de registro de la base de datos, el proceso Oracle ARCH convertirá el registro de rehacer en un registro de archivo almacenado en la ubicación especificada en el disco.

|  |  |
| --- | --- |
|  | Oracle admite la noción de múltiples destinos de registros de archivo, lo que permite el almacenamiento de un registro de rehacer en diferentes ubicaciones de archivos. Múltiples ubicaciones de almacenamiento son comunes cuando se usa Oracle DataGuard para transferir copias de los registros de archivo a un servidor secundario para recuperación ante desastres o inteligencia comercial. No cubriremos cómo configurar esto en esta publicación de blog, pero vale la pena señalar que la funcionalidad existe y puede ser útil. |

### Configurar Oracle: Redo logs

El registro de transacciones de Oracle se conoce como registro de rehacer(redo). Estos registros son vitales, ya que se utilizan en caso de bloqueo de la base de datos o falla de los medios para recuperarse en un punto de control. Desafortunadamente, las imágenes de contenedor de Oracle a menudo usan una configuración de registro redo que no es útil para Debezium.

Existen dos [estrategias de minería de registros](https://debezium.io/documentation/reference/stable/connectors/oracle.html#oracle-property-log-mining-strategy) para el conector Oracle de Debezium. La estrategia controla cómo interactúa el conector con Oracle LogMiner y cómo el conector incorpora los cambios de esquema y tabla.

<https://debezium.io/documentation/reference/stable/connectors/oracle.html#oracle-property-log-mining-strategy>

redo\_log\_catalog

El diccionario de datos se escribirá periódicamente en los registros de rehacer, lo que provocará una mayor generación de registros de archivo con el tiempo. Esta configuración permite realizar un seguimiento de los cambios de DDL, por lo que si el esquema de una tabla cambia, esta será la estrategia ideal para ese propósito.

online\_catalog

El diccionario de datos no se escribirá periódicamente en los registros de rehacer, dejando la generación de registros de archivo coherente con el comportamiento actual. Oracle LogMiner extraerá los cambios sustancialmente más rápido; sin embargo, este rendimiento tiene el costo de **no** realizar un seguimiento de los cambios de DDL. Si el esquema de una tabla se mantiene constante, esta será la estrategia ideal para ese propósito.

Al usar el modo online\_catalog, puede omitir este paso por completo de manera segura.

Cuando se utiliza el modo redo\_log\_catalog (el valor predeterminado), el tamaño del registro de rehacer es fundamental para reducir la frecuencia de los cambios de registro. La sesión de LogMiner se reinicia y el diccionario de datos se reconstruye en los registros de rehacer cuando se produce un cambio de registro. LogMiner vuelve a leer el diccionario y lo utiliza para realizar un seguimiento de los cambios de DDL cuando se reinicia la sesión, lo que puede provocar un ligero retraso inicial de la sesión mientras se rellenan las tablas del diccionario. En general, obtiene un mejor rendimiento cuando el registro de rehacer es lo suficientemente grande como para escribir el diccionario de datos en un solo archivo de registro.

Las imágenes de registro de contenedor de Oracle vienen configuradas con un tamaño de registro de rehacer de 200MB. Este tamaño predeterminado es demasiado pequeño cuando se usa la estrategia de minería predeterminada, por lo que lo ajustaremos para que los registros usen un tamaño de 400MB.

|  |  |
| --- | --- |
|  | Cuando se trabaja con Oracle instalado sin multicliente, el uso 400MBaún puede ser un poco pequeño, ya que existe una gran cantidad de tablas base en la base de datos raíz, pero no existen en las bases de datos conectables cuando se habilita el multicliente. Úselo 500MBen su lugar si está incorporando cambios desde un entorno de Oracle sin multiusuario como mínimo.  Independientemente de la tenencia múltiple, estos valores deberían ser mucho más significativos en su entorno de producción. Su DBA podrá utilizar la guía de dimensionamiento de Oracle para determinar el mejor valor en función de la frecuencia de cambio de registro y la carga en el sistema. |

Antes de realizar cualquier cambio, es esencial examinar el estado actual de su entorno. En el mismo terminal donde habilitó el registro de archivo, ejecute el siguiente SQL para determinar los tamaños de registro actuales.

SELECT GROUP#, BYTES/1024/1024 SIZE\_MB, STATUS FROM V$LOG ORDER BY 1;

La imagen de registro del contenedor de Oracle devolverá el siguiente resultado:

GROUP# SIZE\_MB STATUS

---------- ---------- ----------------

1 200 INACTIVE

2 200 INACTIVE

3 200 CURRENT

Esta salida nos dice que hay 3 grupos de registro, y cada grupo consume 200MB de espacio por registro. Además, el estado asociado con cada grupo es crucial ya que representa el estado actual de ese registro.

A continuación se describe el status en detalle:

INACTIVE

Esto significa que Oracle ha inicializado el registro y no está actualmente en uso.

ACTIVE

Esto significa que Oracle ha inicializado el registro y actualmente está en uso. El registro de rehacer es necesario y está en uso en caso de falla para que la base de datos pueda recuperarse de manera segura.

CURRENT

Esto significa que Oracle está escribiendo actualmente en este registro. Cuando se trabaja con Oracle Real Application Clusters (RAC), se pueden marcar varios registros como actuales, lo que representa un registro por nodo de clúster.

UNUSED

Esto significa que Oracle no ha inicializado el registro y no está en uso.

Ahora, utilizando la misma ventana de terminal, ejecute el siguiente SQL para determinar los nombres de archivo y las ubicaciones de los registros de rehacer.

SELECT GROUP#, MEMBER FROM V$LOGFILE ORDER BY 1, 2;

La imagen de registro del contenedor de Oracle devolverá el siguiente resultado:

GROUP# MEMBER

---------- ---------------------------------------------------

1 /opt/oracle/oradata/ORCLCDB/redo01.log

2 /opt/oracle/oradata/ORCLCDB/redo02.log

3 /opt/oracle/oradata/ORCLCDB/redo03.log

Podemos deducir de esto que cada grupo de registro consta de un solo registro de rehacer. Oracle admite la noción de varios registros por grupo, lo que permite lo que se denomina multiplexación. Por lo general, solo verá esto en un entorno de producción y ocasionalmente en un entorno de prueba, pero es raro ver esto en un entorno de desarrollo o contenedor.

El objetivo es ajustar la BYTEScolumna de la V$LOGtabla para que tenga un valor de 400MB. Desafortunadamente, la única forma de hacer este ajuste es descartar y volver a crear el grupo de registro, y esto solo es posible si el STATUSdel grupo es INACTIVEo UNUSED. Dado que el grupo de registros 1 estaba INACTIVEarriba, comenzaremos con él, pero puede realizar este procedimiento de manera segura en los grupos de registros en cualquier orden.

En la terminal donde se ejecuta SQL\*Plus, ejecute lo siguiente:

ALTER DATABASE CLEAR LOGFILE GROUP 1;

ALTER DATABASE DROP LOGFILE GROUP 1;

ALTER DATABASE ADD LOGFILE GROUP 1 ('/opt/oracle/oradata/ORCLCDB/redo01.log') size 400M REUSE;

Esto eliminará y volverá a crear el grupo de registro con el tamaño de 400MB. Usaremos el mismo nombre de archivo de registro en la MEMBERcolumna de la VLOGFILEtabla. Si la base de datos utiliza multiplexación, con varios archivos de registro por grupo de registros, utilice una lista de nombres de archivos separados por comas para registrar cada archivo de registro.

Continúe con el procedimiento anterior para todos los grupos de registro, cambiando el grupo de registro y los nombres de archivo en consecuencia hasta que todos los INACTIVEgrupos UNUSEDtengan un tamaño de 400MB. Una vez que todo lo que queda por cambiar son los que son CURRENT, puede emitir un cambio de registro en la base de datos para avanzar la base de datos al siguiente registro de rehacer usando el siguiente SQL:

ALTER DATABASE OPEN;

ALTER SYSTEM SWITCH LOGFILE;

Si vuelve a verificar el tamaño de los registros en el V$LOG, verá que el resultado es similar al siguiente:

SQL> SELECT GROUP#, BYTES/1024/1024 SIZE\_MB, STATUS FROM V$LOG ORDER BY 1;

GROUP# SIZE\_MB STATUS

---------- ---------- ----------------

1 400 CURRENT

2 400 UNUSED

3 200 ACTIVE

Ahora debemos esperar a que la base de datos cambie el estado del grupo de registro 3 a INACTIVE. El cambio puede tardar varios minutos, así que tenga paciencia y vuelva a comprobar el tamaño periódicamente. Una vez que el estado llegue a INACTIVE, modifique el grupo de registro final y el nombre de archivo utilizando el mismo procedimiento.

Una última revisión de la V$LOGtabla después del grupo de registro final, veremos que todo se ve en orden:

SQL> SELECT GROUP#, BYTES/1024/1024 SIZE\_MB, STATUS FROM V$LOG ORDER BY 1;

GROUP# SIZE\_MB STATUS

---------- ---------- ----------------

1 400 CURRENT

2 400 UNUSED

3 400 UNUSED

En este punto, hemos modificado todos los tamaños de registros de rehacer, reduciendo la frecuencia de los cambios de registro cuando Debezium ejecuta los pasos de creación del diccionario de datos.

### Configurar Oracle: Supplemental Logging

Los registros de rehacer de Oracle se utilizan principalmente para la recuperación de instancias y medios porque los datos necesarios para esas operaciones se registran automáticamente. LogMiner no se puede utilizar de forma predeterminada porque Oracle no proporciona datos de registro complementarios listos para usar. Dado que Debezium se basa en LogMiner, el registro complementario debe estar habilitado como mínimo para que Debezium realice cualquier captura de datos modificados para Oracle.

Se pueden utilizar dos estrategias diferentes para establecer el registro complementario:

1. Registro complementario de la base de datos
2. Registro suplementario de tabla

Para que Debezium interactúe con LogMiner y trabaje con filas encadenadas y varios arreglos de almacenamiento, el registro complementario de la base de datos debe estar habilitado en un nivel mínimo. Para habilitar este nivel, ejecute el siguiente SQL en el terminal SQL\*Plus actual:

ALTER DATABASE ADD SUPPLEMENTAL LOG DATA;

Cubriremos el registro complementario basado en tablas en una sección posterior cuando analicemos la configuración del conector.

ADEMAS LE QUITAMOS EL EL LIMITE DE LOGIN

ALTER PROFILE DEFAULT LIMIT FAILED\_LOGIN\_ATTEMPTS UNLIMITED;

### Configurar Oracle: User setup

Para que el conector Debezium capture eventos de cambio, debe establecer una conexión JDBC con la base de datos y ejecutar una serie de API de LogMiner. Una cuenta de usuario requerirá permisos específicos para acceder a estas API de LogMiner y recopilar datos de las tablas capturadas.

Cuando se utiliza una arquitectura multiusuario, como se encuentra con la imagen de registro de contenedor de Oracle, existen efectivamente dos bases de datos con las que tendremos que trabajar, ORCLCDB(la base de datos de contenedor o raíz) y ORCLPDB1(la base de datos conectable). Todas las tablas capturadas se crearán y mantendrán desde la PDB, pero habrá momentos en los que el conector necesitará acceder a la base de datos raíz para leer tablas específicas del sistema.

Por lo tanto, en una arquitectura multiinquilino, primero debemos configurar los dos espacios de tabla que usará nuestra cuenta de usuario. Para crear estos espacios de tabla, ejecute el siguiente SQL desde el terminal SQL\*Plus:

CONNECT sys/oraclepw@ORCLCDB as sysdba;

CREATE TABLESPACE logminer\_tbs DATAFILE '/opt/oracle/oradata/ORCLCDB/logminer\_tbs.dbf'

SIZE 25M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;

CONNECT sys/oraclepw@ORCLPDB1 as sysdba;

CREATE TABLESPACE logminer\_tbs DATAFILE '/opt/oracle/oradata/ORCLCDB/ORCLPDB1/logminer\_tbs.dbf'

SIZE 25M REUSE AUTOEXTEND ON MAXSIZE UNLIMITED;

|  |  |
| --- | --- |
|  | Si la implementación no se realiza en una base de datos de Oracle con multiusuario habilitado, ORCLPDB1no es necesario crear el segundo espacio de tabla dentro de la base de datos. Además, asegúrese de que la ruta proporcionada para el tablespace, las credenciales y el SID de la base de datos sean correctos para su instalación. Es posible que deba consultar con su DBA para que el tablespace se cree correctamente. |

Una vez que existen los espacios de tabla, ahora es el momento de crear la cuenta de usuario. Si está utilizando un entorno de múltiples inquilinos, el nombre de usuario debe usar el prefijo de usuario común para que Oracle lo cree tanto en la base de datos raíz de CDB como en la base de datos conectable de PDB; de lo contrario, el nombre de usuario puede ser cualquier cosa. Dado que estamos trabajando con una instalación de base de datos multiusuario con nuestro contenedor, crearemos una cuenta de usuario llamada c##dbzuser.

CONNECT sys/oraclepw@ORCLCDB as sysdba;

CREATE USER c##dbzuser IDENTIFIED BY dbz DEFAULT TABLESPACE LOGMINER\_TBS

QUOTA UNLIMITED ON LOGMINER\_TBS

CONTAINER=ALL;

CREATE USER c##dbzuser IDENTIFIED BY dbz DEFAULT TABLESPACE LOGMINER\_TBS QUOTA UNLIMITED ON LOGMINER\_TBS CONTAINER=ALL;

La cuenta de usuario requiere varios permisos. En el momento de esta publicación, la lista de permisos incluía lo siguiente:

GRANT CREATE SESSION TO c##dbzuser CONTAINER=ALL;

GRANT SET CONTAINER TO c##dbzuser CONTAINER=ALL;

GRANT SELECT ON V\_$DATABASE TO c##dbzuser CONTAINER=ALL;

GRANT FLASHBACK ANY TABLE TO c##dbzuser CONTAINER=ALL;

GRANT SELECT ANY TABLE TO c##dbzuser CONTAINER=ALL;

GRANT SELECT\_CATALOG\_ROLE TO c##dbzuser CONTAINER=ALL;

GRANT EXECUTE\_CATALOG\_ROLE TO c##dbzuser CONTAINER=ALL;

GRANT SELECT ANY TRANSACTION TO c##dbzuser CONTAINER=ALL;

GRANT SELECT ANY DICTIONARY TO c##dbzuser CONTAINER=ALL;

GRANT LOGMINING TO c##dbzuser CONTAINER=ALL;

GRANT CREATE TABLE TO c##dbzuser CONTAINER=ALL;

GRANT LOCK ANY TABLE TO c##dbzuser CONTAINER=ALL;

GRANT CREATE SEQUENCE TO c##dbzuser CONTAINER=ALL;

GRANT EXECUTE ON DBMS\_LOGMNR TO c##dbzuser CONTAINER=ALL;

GRANT EXECUTE ON DBMS\_LOGMNR\_D TO c##dbzuser CONTAINER=ALL;

GRANT SELECT ON V\_$LOG TO c##dbzuser CONTAINER=ALL;

GRANT SELECT ON V\_$LOG\_HISTORY TO c##dbzuser CONTAINER=ALL;

GRANT SELECT ON V\_$LOGMNR\_LOGS TO c##dbzuser CONTAINER=ALL;

GRANT SELECT ON V\_$LOGMNR\_CONTENTS TO c##dbzuser CONTAINER=ALL;

GRANT SELECT ON V\_$LOGMNR\_PARAMETERS TO c##dbzuser CONTAINER=ALL;

GRANT SELECT ON V\_$LOGFILE TO c##dbzuser CONTAINER=ALL;

GRANT SELECT ON V\_$ARCHIVED\_LOG TO c##dbzuser CONTAINER=ALL;

GRANT SELECT ON V\_$ARCHIVE\_DEST\_STATUS TO c##dbzuser CONTAINER=ALL;

GRANT SELECT ON V\_$TRANSACTION TO c##dbzuser CONTAINER=ALL