Oracle OpenWorld 2019

SAN FRANCISCO

ORACLE

Performance & Scalability for Java Applications using an RDBMS: What's New

Nirmala Sundarappa
Principal Product Manager
Oracle JDBC and UCP

Kuassi Mensah
Director, Product Management
Oracle JDBC and UCP

Ilesh Garish
CMTS
Oracle JDBC and UCP

Safe Harbor

Agenda

19.3 JDBC driver & some companion jars on Central Maven

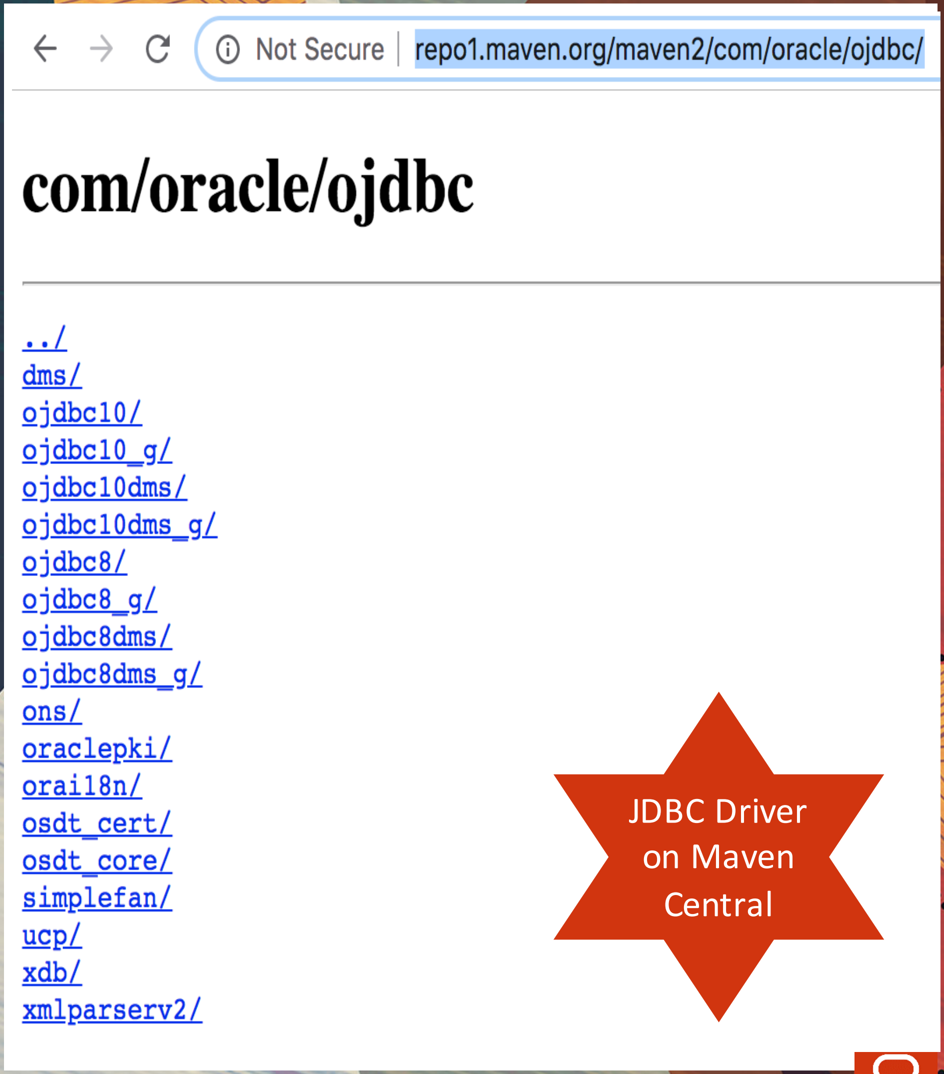
https://repo1.maven.org/maven2/com/oracle/ojdbc/

<groupId>**com.oracle.ojdbc**</groupId>

<artifactId>**ojdbc8**</artifactId>

<version>**19.3.0.0**</version>

Older releases will be available later



Not Secure | repo1.maven.org/maven2/com/oracle/ojdbc/

com/oracle/ojdbc

../
dms/
ojdbc10/
ojdbc10_g/
ojdbc10dms/
ojdbc10dms_g/
ojdbc8/
ojdbc8_g/
ojdbc8dms/
ojdbc8dms_g/
ons/
oraclepki/
orai18n/
osdt_cert/
osdt_core/
simplefan/
ucp/
xdb/
xmlparserv2/

JDBC Driver on Maven Central

What’s New

- Eclipse Plugin for Autonomous Databases
- Asynchronous Extentions for JDBC driver
- Reactive Streams Ingest (RSI) library
- JSON data type support
- Connection URL enhancements
  - Easy Connect Plus
  - Multiple ways to set TNS_ADMIN
  - Support for HTTPS_PROXY
- New *ojdbc.properties* file

Eclipse Plugin for Autonomous Database

Use case

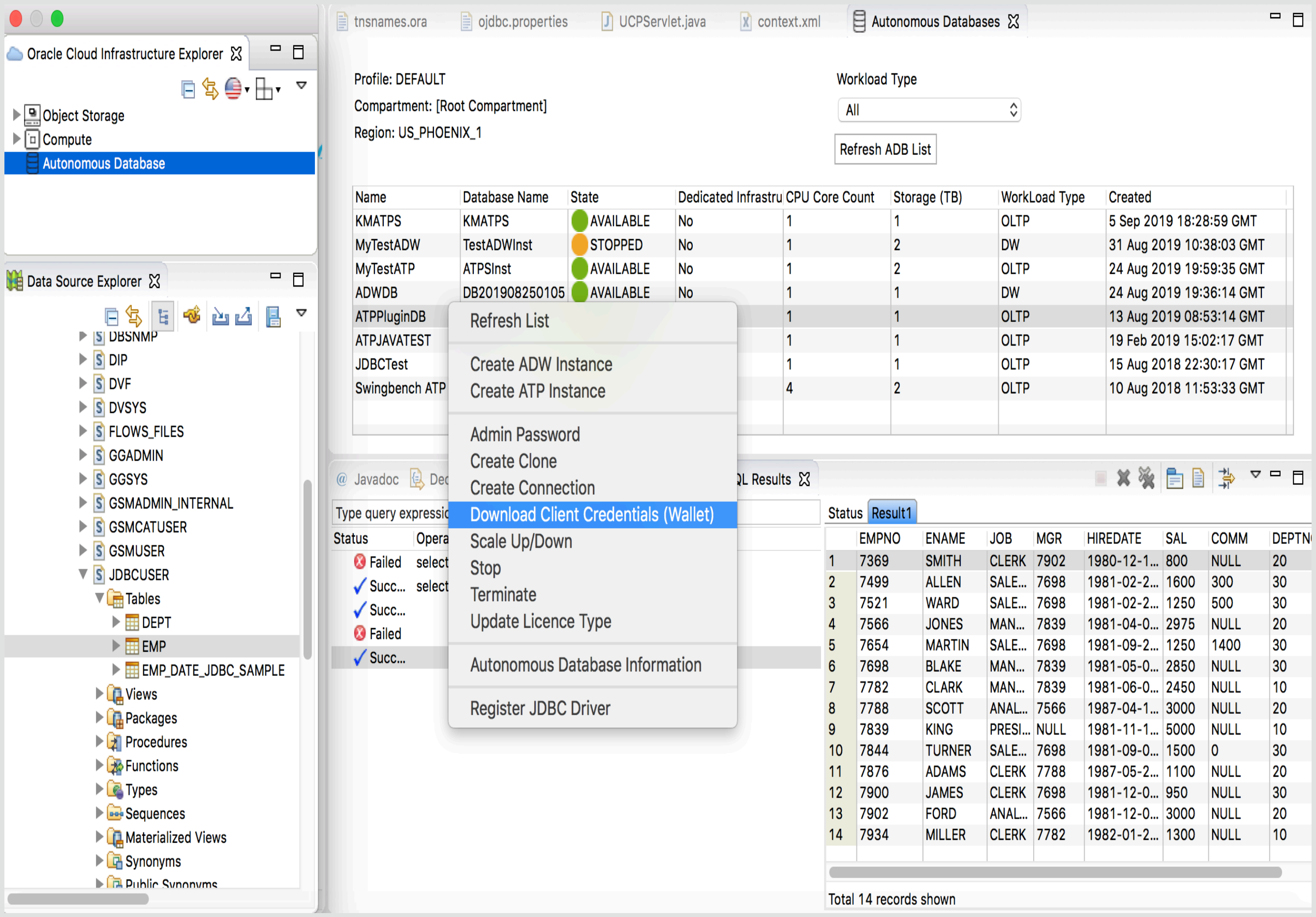
Java developers should be able to connect to the Autonomous Database (ATP/ADW) through Eclipse.
(Integrated Development experience)

Solution

A new Eclipse Plugin called as **"Oracle Cloud Infrastructure Toolkit for Eclipse"** is developed.

Oracle Cloud Infrastructure Toolkit for Eclipse

- Access both ATP and ADW
- Download the client credentials
- Test the connection
- Browse the schema
- Other Database operations
  - Create New ATP or ADW
  - Start/Stop/Clone
  - Scale Up/Down
  - Change the ADMIN password



Profile: DEFAULT
Compartment: [Root Compartment]
Region: US_PHOENIX_1

Workload Type: All

Name	Database Name	State	Dedicated Infrastru	CPU Core Count	Storage (TB)	WorkLoad Type	Created
KMATPS	KMATPS	● AVAILABLE	No	1	1	OLTP	5 Sep 2019 18:28:59 GMT
MyTestADW	TestADWInst	● STOPPED	No	1	2	DW	31 Aug 2019 10:38:03 GMT
MyTestATP	ATPSInst	● AVAILABLE	No	1	2	OLTP	24 Aug 2019 19:59:35 GMT
ADWDB	DB201908250105	● AVAILABLE	No	1	1	DW	24 Aug 2019 19:36:14 GMT
ATPPluginDB				1	1	OLTP	13 Aug 2019 08:53:14 GMT
ATPJAVATEST				1	1	OLTP	19 Feb 2019 15:02:17 GMT
JDBCTest				1	1	OLTP	15 Aug 2018 22:30:17 GMT
Swingbench ATP				4	2	OLTP	10 Aug 2018 11:53:33 GMT

Type query expressio

Status	Opera
✗ Failed	select
✓ Succ...	select
✓ Succ...	
✗ Failed	
✓ Succ...	

Status Result1

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTN
1	7369	SMITH	CLERK	7902	1980-12-1...	800	NULL	20
2	7499	ALLEN	SALE...	7698	1981-02-2...	1600	300	30
3	7521	WARD	SALE...	7698	1981-02-2...	1250	500	30
4	7566	JONES	MAN...	7839	1981-04-0...	2975	NULL	20
5	7654	MARTIN	SALE...	7698	1981-09-2...	1250	1400	30
6	7698	BLAKE	MAN...	7839	1981-05-0...	2850	NULL	30
7	7782	CLARK	MAN...	7839	1981-06-0...	2450	NULL	10
8	7788	SCOTT	ANAL...	7566	1987-04-1...	3000	NULL	20
9	7839	KING	PRESI...	NULL	1981-11-1...	5000	NULL	10
10	7844	TURNER	SALE...	7698	1981-09-0...	1500	0	30
11	7876	ADAMS	CLERK	7788	1987-05-2...	1100	NULL	20
12	7900	JAMES	CLERK	7698	1981-12-0...	950	NULL	30
13	7902	FORD	ANAL...	7566	1981-12-0...	3000	NULL	20
14	7934	MILLER	CLERK	7782	1982-01-2...	1300	NULL	10

Total 14 records shown

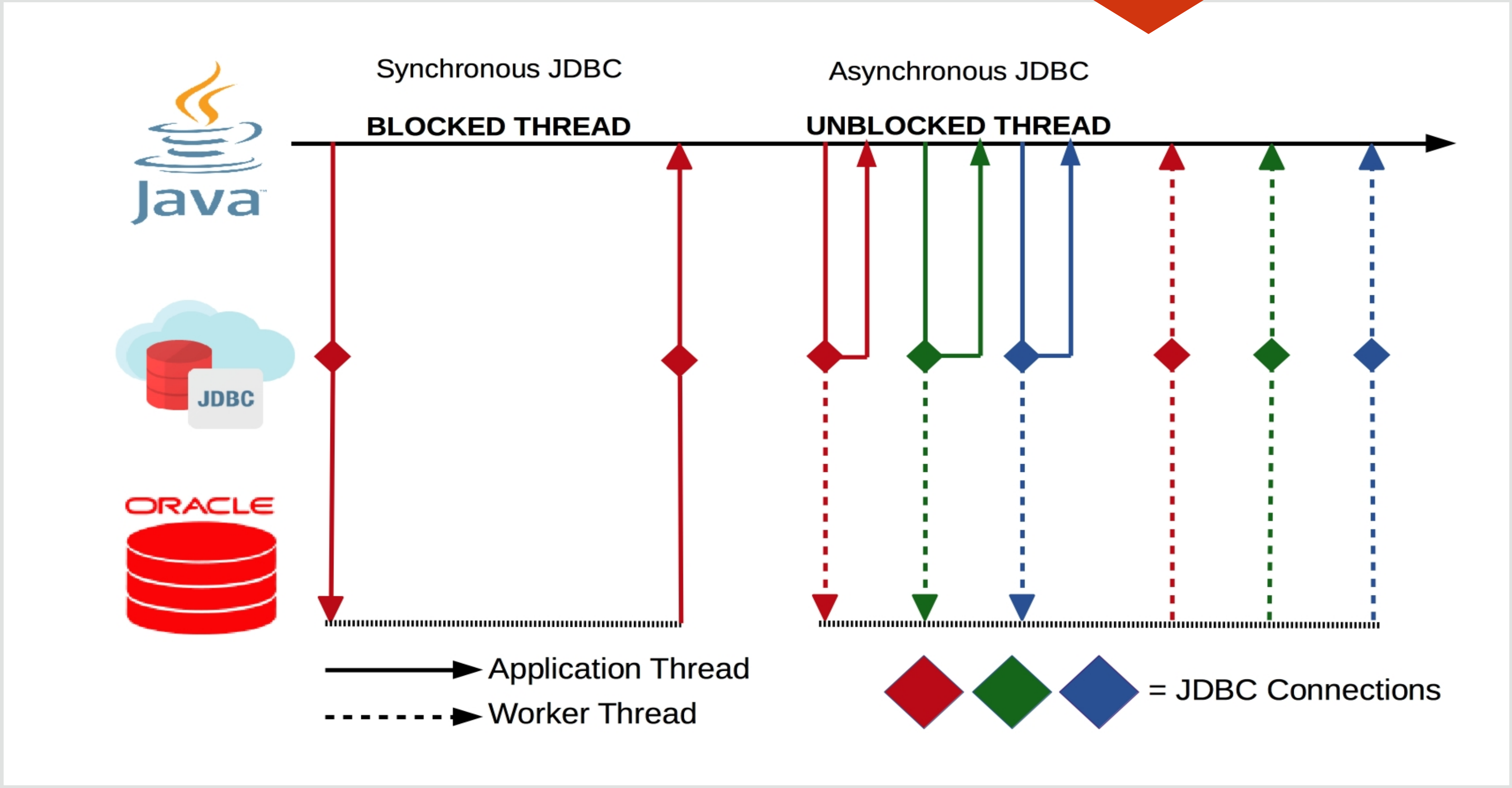
Asynchronous Extensions for JDBC driver

- What is this?
  - New APIs for asynchronous database access
  - These APIs won't block a thread when opening connections, executing SQL, or reading and writing LOBs
- Why do we need?
  - Non-Blocking calls use less threads
  - Less threads means less memory usage
- How does it look like?
  - Exposed as reactive-streams with *java.util.concurrent.Flow* interfaces
  - Reactive-Streams control the rate at which applications receive data
  - The Flow interfaces allow for inter-op compatibility with popular libraries such as Reactor, RxJava, and Akka-Streams.

Non-Blocking JDBC

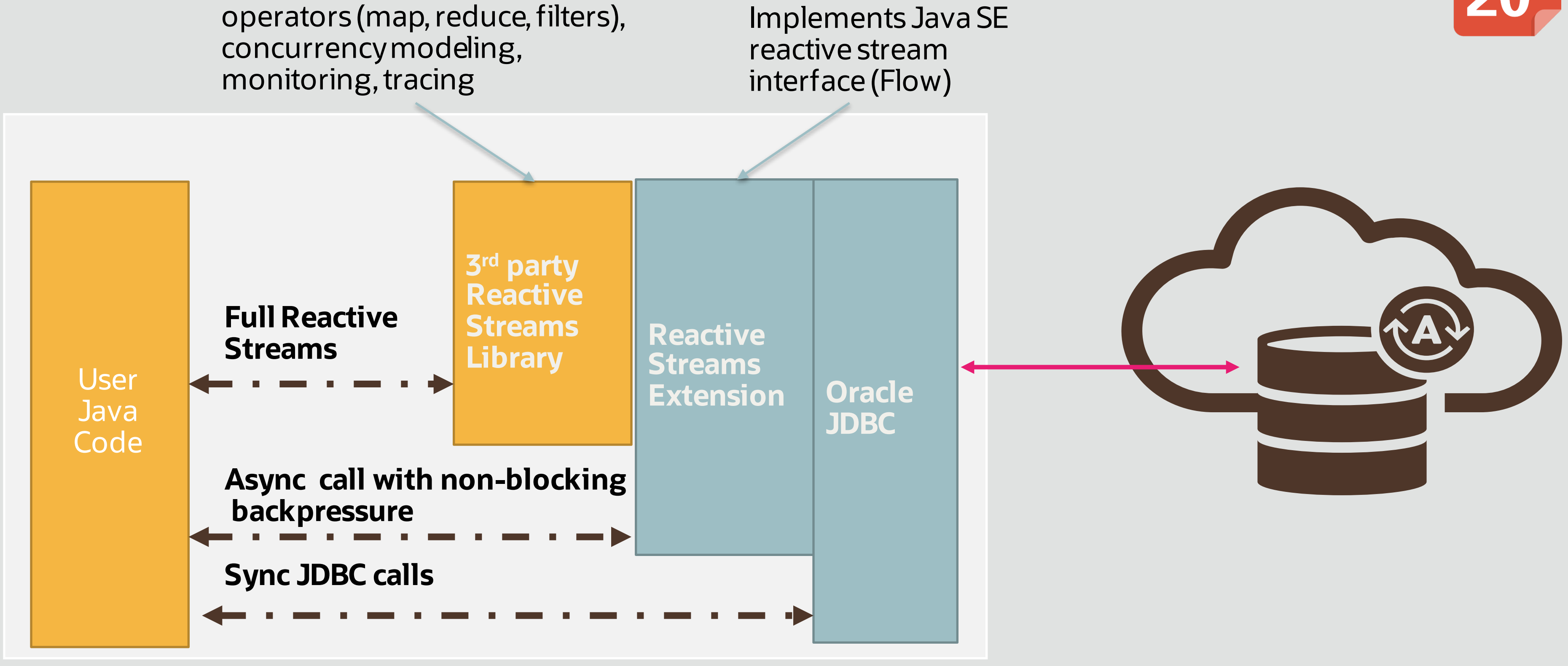
Learn more in DEV6323

NEW IN 20c



Database Access with Oracle JDBC

NEW IN 20c


operators (map, reduce, filters), concurrency modeling, monitoring, tracing
Implements Java SE reactive stream interface (Flow)
User Java Code
Full Reactive Streams
3rd party Reactive Streams Library
Reactive Streams Extension
Oracle JDBC
Async call with non-blocking backpressure
Sync JDBC calls

Reactive and Asynchronous JDBC APIs

NEW IN 20c

Connection Creation (OracleConnectionBuilder):

```
Publisher<OracleConnection> buildConnectionPublisherOracle()
```

SQL Execution (OraclePreparedStatement):

```
Publisher<Boolean> executeAsyncOracle()
Publisher<Long> executeUpdateAsyncOracle()
Publisher<Long> executeBatchAsyncOracle()
Publisher<OracleResultSet> executeQueryAsyncOracle()
```

Row Fetching (OracleResultSet):

```
Publisher<T> publisherOracle(Function<OracleRow, T> f)
```

LOB I/O (OracleBlob):

```
Publisher<byte[]> publisherOracle(long position)
Subscriber<byte[]> subscriberOracle(long position)
```

LOB I/O (OracleClob):

```
Publisher<String> publisherOracle(long position)
Subscriber<String> subscriberOracle(long position)
```

Connection Closing (OracleConnection):

```
Publisher<Success> closeAsyncOracle()
```

Transaction Closing (OracleConnection):

```
Publisher<Success> commitAsyncOracle()
Publisher<Success> rollbackAsyncOracle()
```

Reactive Streams Ingest (RSI) Library

Use case

Requirement to persist large amount of records in the form of rows in a table in the Oracle Database table.
Example: IoT agents or Telco Applications etc.,

Solution

A new Java library called as **“Reactive Streams Ingest (RSI)”** uses direct path to ingest the data in a fast and non-blocking way.

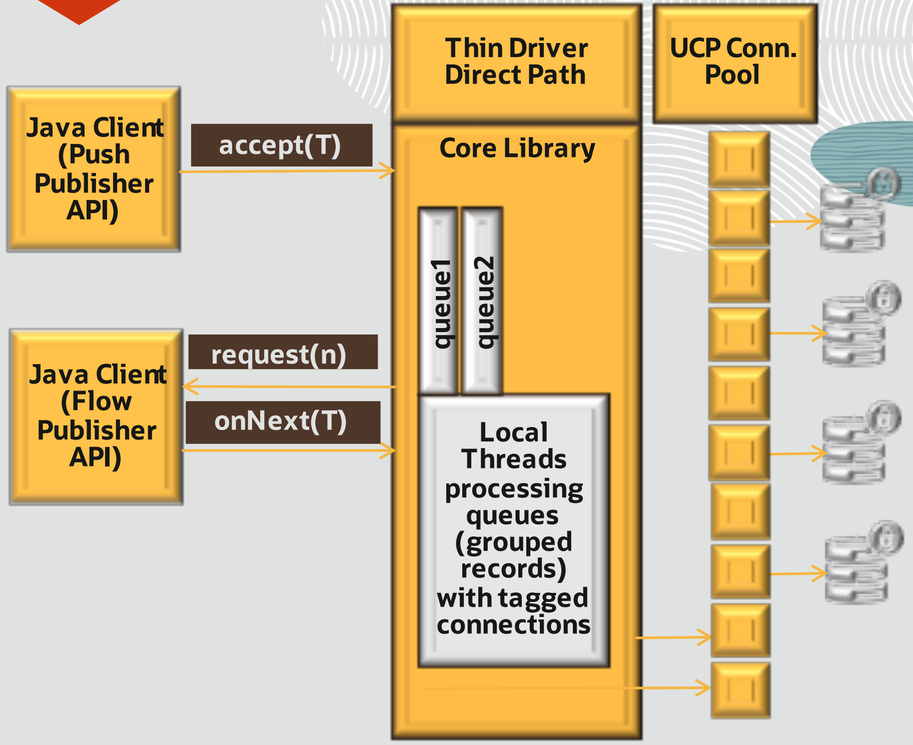
Introducing the Reactive Streams Ingestion Library

Fastest insert method for the Oracle Database through Direct Path.

RAC and Shard awareness (routing capabilities) through native UCP.

Extremely simple to configure and use.

Streaming capability: unblockingly receive data from a large group of clients.


Java Client (Push Publisher API)
accept(T)
Java Client (Flow Publisher API)
request(n)
onNext(T)
Thin Driver Direct Path
UCP Conn. Pool
Core Library
queue1
queue2
Local Threads processing queues (grouped records) with tagged connections

NEW IN 20c

JDBC Support for Native JSON Datatype

NEW IN 20c

- The `oracle.sql.JSON` package contains classes and interfaces that allow Java applications to support JSON-P, a Java API for JSON processing
- A simpler and richer type system
  - Support for dates, timestamps
  - No constraint check (IS JSON)
- Improved performance for Java applications
  - Faster access to nested JSON values

Easy Connect Plus

Problem

Easy Connection URL was easy to use but was not useful in many scenarios.

- TCPS connections
- Not recommended for RAC
- Limited to single hostname/port number

Solution

Easy Connect Plus that removes all the limitations and allows.

- TCPS connections,
- RAC aware
- Multiple hostname/port number

Easy Connect Plus

NEW IN 19c

Easy Connect

- Allowed only TCP connections with hostname, port number, and service name

```
jdbc:oracle:thin:@//myhost:myport/myservice.oracle.com
```

Easy Connect Plus

- Allows TCP and TCPS connections. It also takes security related properties in the URL

```
jdbc:oracle:thin:@tcps://myhost:1522/myservice.oracle.com?wallet_location=/wallet&oracle.net.ssl_server_cert_dn=\"CN=adwc.uscom-east-1.oraclecloud.com,OU=Oracle BMCS US,O=Oracle Corporation,L=Redwood City,ST=California,C=US\”
```

Easy Connect Plus

NEW IN 19c

Easy Connect

- Did not allow connection properties in the URL.
- No HA capabilities

```
jdbc:oracle:thin:@//myhos
t:myport/myservice.oracle
.com
```

Easy Connect Plus

- Allows connection properties in the URL
- Can be used for HA

Example:

```
jdbc:oracle:thin:@tcps://salesse
rver1:1521/sales.us.example.com?
connect_timeout=60&
transport_connect_timeout=30&ret
ry_count=3
```

Easy Connect Plus

NEW IN 19c

Easy Connect

- Allowed only **Single host name and port number** in the URL.
- Not recommended for RAC

Example:

```
jdbc:oracle:thin:@//myhost:
myport/myservice.oracle.com
```

Easy Connect Plus

- Allows **Multi host and Multi port number** in the URL
- Can be used with RAC

Example:

```
jdbc:oracle:thin:@tcps://salesse
rver1:1521, salesserver1,
salesserver3:1522/sales.us.examp
le.com
```

New *ojdbc.properties* file

- New file **ojdbc.properties** for setting connection properties
  - Comes handy for ATP and ADW connections
  - Included as part of `wallet_<dbname>.zip` download for ATP/ADW
- Default location is where the *tnsnames.ora* is present
  - Specify the location with $TNS_ADMIN property
- The property file can be given a custom name
  - ojdbc_orcl.properties, ojdbc_<TNSalias>.properties
- The property file can be placed at any other place
  - **`oracle.jdbc.config.file`** can be set by specifying the location

Multiple ways to set TNS_ADMIN

NEW IN 19c

Before 19c

- ONLY way is to set as system property
- Example: `-Doracle.net.tns_admin`

19c and later

- Can be set as part of URL
- Example: `jdbc:oracle:thin:@//myhost:1521/orcl?`**`TNS_ADMIN=/home/oracle/network/admin/`**
- Can be set as a connection property.

Support for HTTPS Proxy Configuration

- Support HTTPS Proxy settings in the connection URL
- Can connect to Oracle cloud through internal corporate network
- Should be used only for **testing purposes** but not in production
- Example:

```
(DESCRIPTION=(ADDRESS=(HTTPS_PROXY=sales-proxy)(HTTPS_PROXY_PORT=8080)(PROTOCOL=TCPS)(HOST=sales2-svr)(PORT=443))(CONNECT_DATA=(SERVICE_NAME=sales.us.example.com)))
```

Performance Capabilities

- Client side result set cache
- Use Universal Connection Pool
- Other best practices
  - Disable AUTO COMMIT
  - Use Prepared Statements
  - Enable Statement Caching
  - Enable Array fetching
  - Use Network data compression

Client Side Result set Cache

NEW IN 19c

- Result sets of frequent/identical queries are cached in the driver's memory on the client side
- Cache invalidation when the data changes on server side
- To be enabled on Server side
  - Set CLIENT_RESULT_CACHE_SIZE and CLIENT_RESULT_CACHE_LAG
- Disable by setting *oracle.jdbc.enableQueryResultCache* to false
- Enabled by default on client side
  - Three levels: Query level, Table annotations, Server initialization parameter
  - Add SQL hints at the query level.
  - SELECT /** result_cache */ first_name, last_name from employees where employee_id < 150

Universal Connection Pool (UCP)

- Oracle's **feature-rich Java connection pool**
- **Profoundly enhanced** for faster connection management
- **Works seamlessly** with Oracle Real Application Clusters (RAC), Active Data Guard (ADG) and Global Data Services (GDS)
- **Tested and proven solution** to achieve scalability and high-availability during planned and unplanned database downtimes.
- **Easy to configure** with any application containers such as Tomcat, IBM WebSphere, Web Logic Server etc.,
- **UCP properties** help in tuning the behavior of the connection pool based on the application needs.

Universal Connection Pool (ucp.jar)

```
<Context docBase="ATPWebApp" path="/ATPWebApp"
   reloadable="true" source="org.eclipse.jst.jee.server:samplejdbcpage">

 <Resource name="tomcat/UCP_atp" auth="Container"
   factory="oracle.ucp.jdbc.PoolDataSourceImpl"
   type="oracle.ucp.jdbc.PoolDataSource"
   description="UCP Pool in Tomcat"
   connectionFactoryClassName="oracle.jdbc.pool.OracleDataSource"
   minPoolSize="5"
   maxPoolSize="50"
   initialPoolSize="15"
   user="hr"
   password="hr"
   url="jdbc:oracle:thin:@databasename_medium?TNS_ADMIN=/Users/test/lib"
 />

</Context>
```

- IBM WebSphere
- IBM Liberty
- Apache Tomcat
- NEC WebOTX
- Red Hat WildFly (JBoss)
- Hibernate
- Spring
- custom

Scalability Features

- Horizontal Scaling through Sharding
- Scaling through Multi-tenant database
- JDBC Driver for Sharding

JDBC Driver for Sharding

NEW IN 20c

- A new JDBC driver that enables Java connectivity to a sharded database without the need for an application to furnish a sharding key.
- Makes Oracle sharding transparent to Java apps as much as possible.
- Use this driver
  - when an application doesn't know the sharding key upfront
  - When there is a mixed queries to direct shard and cross shard

JDBC Driver for Sharding
Connection Property

NEW IN 20c

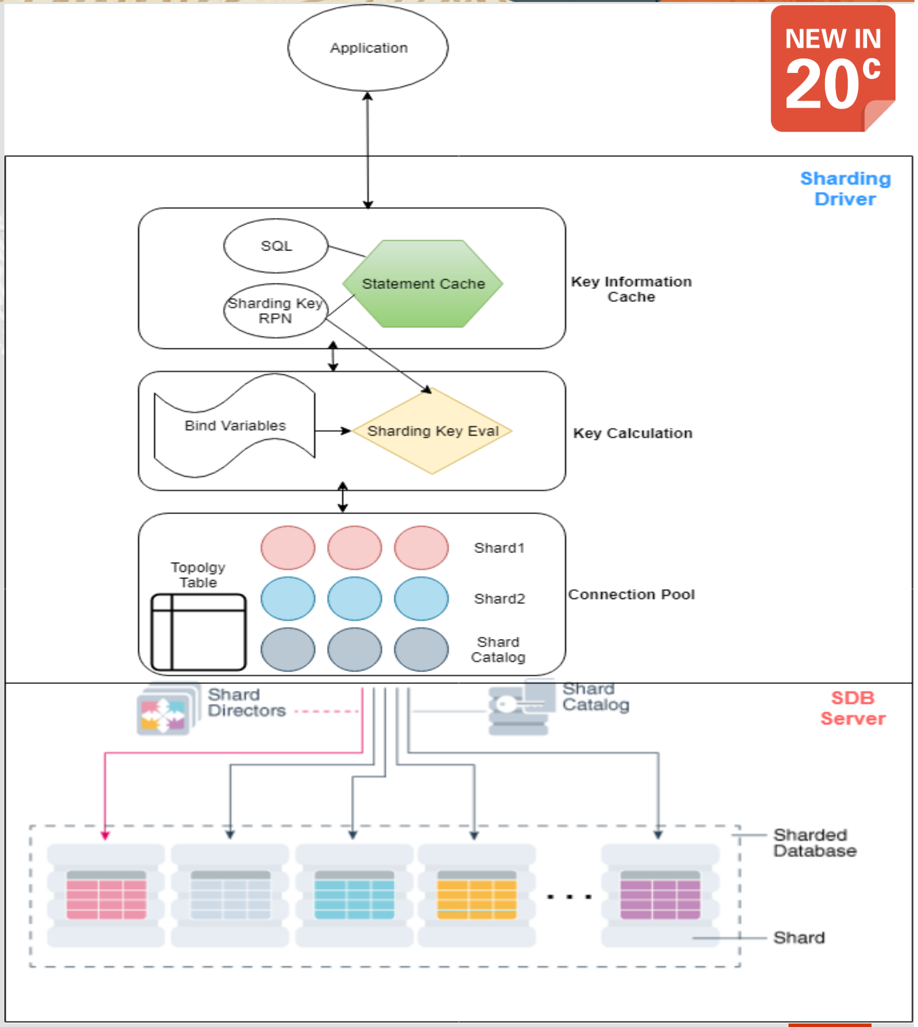
- The new connection property is "oracle.jdbc.useShardingDriverConnection" and set it value to "true".
- It can be set as part of the connection URL or as a connection property
- The default value is "false"

JDBC Driver for Sharding Query Execution

- An application executes a query using the driver. e.g.

  SELECT * FROM MY_TABLE WHERE ID = ?
- The driver connects to the catalog and gets the sharding key information for the query.
- The driver computes the key and executes it using a direct shard connection.
- The driver caches the query and sharding key information.

NEW IN 20c



JDBC Driver for Sharding with Replay Capability

NEW IN 20c

- "oracle.jdbc.shardingReplayEnable" connection property needs to be set to use the replay data source internally.

- Default is '*false*' and uses plain vanilla Thin connection.

- Replay data source is required for achieving Application Continuity in Java applications.

Benefits of the Sharding Driver

NEW IN 20c

- Derives the sharding key from the SQL statement. No need to make an extra API call to pass the sharding key.
- No need to worry about the configuration of the UCP.
- No need to check-in/check-out a physical connection for every new sharding key, as the driver automatically does it.
- No need to separate cross-shard statements from single-shard statements and create separate connection pools for them, as the driver maintains those connections pools.
- The driver enables the prepared statement caching and route to the direct shard based on the key used in the SQL statement.
- Simplifies application and optimizes the performance of it without any code changes.

Limitations of the Sharding Driver

NEW IN 20c

- Supported for Thin driver. No support for Thick and KPRB driver.
- Some Oracle JDBC extension API such as Direct Path Load, ICC, DMS, etc. are not supported.
- Single direct shard transaction in auto commit off mode is not supported.
- PL/SQL execution always happens through the catalog database.

Sharding Driver Sample

NEW IN 20c

```
public class ShardingDriverSample {

  public static void main(String[] args) throws SQLException {

    ShardingDriverSample sample = new ShardingDriverSample();

    sample.bindQuerySample();

  }

  private void bindQuerySample() throws SQLException {

    OracleConnection conn = getGsmConnection();

    executeQueryWithBindAndReadRows(conn, "SELECT * FROM MY_SHARD_TABLE where ID = ?", 10);

    System.out.println("Direct shard execution percentage:" +
conn.getPercentageQueryExecutionOnDirectShard());

  }

  private OracleConnection getGsmConnection() throws SQLException {

    Properties prop = new Properties();

    prop.setProperty("oracle.jdbc.useShardingDriverConnection", "true");

    ...

    return ds.getConnection(); }
```

```
 private void executeQueryWithBindAndReadRows(Connection
dbConnection, String sql, int noOfTime)

    throws SQLException {


              while (noOfTime-- > 0) {

                PreparedStatement statement =
dbConnection.prepareStatement(sql);

                statement.setInt(1, 15);

                ResultSet rs = statement.executeQuery();

                 ...

              }

 }

}
```

Summary of RPN Expression format

NEW IN 20c

Stack Instruction	OpCode	Argument(s)	Remarks
<push empty tuple>	1		
<push empty key>	2		
<push short>	3	b1 b2	Big Endian
<push SQL type>	4	b1 b2	Big Endian
<push bind val>	5	b1 b2	Big Endian
<push SQL value>	6	b1…bn	Bytes of Oracle Datum value.
<append key value>	7		Build a subkey.
<append tuple key>	8		Build a key
<return tuple>	9		Returns key(s)

RPN Expression Sample

NEW IN 20c

SQL: SELECT * FROM MY_TABLE WHERE ID = ?;

RPN expression:

<push empty tuple>

<push empty key>

<push short> Index of the bind variable

<push SQL type> 00 02

<push bind val>

<append key value>

<append tuple key>

<return tuple>

Thank You

Oracle JDBC & UCP

Product Management and Product Development
www.oracle.com/jdbc