Break New Ground

San Francisco
September 16–19, 2019

Safe Harbor

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.

Statements in this presentation relating to Oracle's future plans, expectations, beliefs, intentions and prospects are "forward-looking statements" and are subject to material risks and uncertainties. A detailed discussion of these factors and other risks that affect our business is contained in Oracle's Securities and Exchange Commission (SEC) filings, including our most recent reports on Form 10-K and Form 10-Q under the heading "Risk Factors." These filings are available on the SEC's website or on Oracle's website at http://www.oracle.com/investor. All information in this presentation is current as of September 2019 and Oracle undertakes no duty to update any statement in light of new information or future events.

Java/JDBC Scalability and Asynchrony: Reactive Extension and Fibers

Douglas Surber
Michael McMahon
Oracle Server Technology Development
Oracle Database JDBC
September 18, 2019

Oracle Database JDBC Drivers Available Today on Maven Central

- Oracle Database 19.3 release available now
  https://repo1.maven.org/maven2/com/oracle/ojdbc/
- Future releases including patch sets will be uploaded as released
- Selected older releases will be made available

ADBA Standardization Effort Terminated

- Oracle will not work on ADBA (Asynchronous Database Access)
- No work on the API
- No work on the JDBC reference implementation
- No work on any async database access Java standard
- The API and JDBC implementation are GPL, others can continue if they wish

Loom is the Future of Scalable Java

- The value of async code is scalability
- Sequential code is simpler but wastes resources on threads
- Async code is more complex but saves resources
- Project Loom introduces fibers
- Fibers are lightweight threads. Easily millions per JVM
- https://openjdk.java.net/projects/loom/

Scalable Database Access

- A fiber blocked waiting on a database uses negligible resources
- Fiber code is as scalable as async code but simpler
- Use exactly the same JDBC API we all know and love and get the scalability you need
- Frameworks and libraries must change to take advantage of fibers

Oracle Database JDBC Drivers are Fiber-Ready

- Oracle Database 20c JDBC drivers are fiber-ready
- No API change
- Your Oracle JDBC code using 20c will work with fibers without change

Fiber Demo

Oracle Database 20c JDBC Includes Async Extensions

- Oracle Database 20c JDBC drivers include a small number of proprietary extensions to support async database access
- The implementation uses NIO Selector so no threads block waiting on the database
- Minimal API. Not every use case supported
- No expectation that this wil ever be a Java standard. Fibers are the future for Java

oracle.jdbc.ConnectionBuilder

```
public Publisher<OracleConnection> buildPublisherOracle();
```

- Uses the reactive stream model exposed by java.util.concurrent.Flow
- Call Subscription.request to get a Connection
- Method names suffixed with "Oracle" to avoid name conflicts in the unlikely event this is added to the JDBC standard

oracle.jdbc.OraclePreparedStatement

```
public Publisher<Boolean> executeAsyncOracle();
```

- Async version of PreparedStatement.execute()
- Publisher emits a single Boolean result type.
- Calling thread not blocked
- But any other call to the Connection or its dependents will block while the async operation is in-flight

oracle.jdbc.OraclePreparedStatement

```
public Publisher<Long> executeUpdateAsyncOracle();
```

- Async version of PreparedStatement.executeUpdate()
- Publisher emits a single update count.
- Calling thread not blocked
- But any other call to the Connection or its dependents will block while the async operation is in-flight

oracle.jdbc.OraclePreparedStatement

```
public Publisher<Long> executeBatchAsyncOracle();
```

- Async version of PreparedStatement.executeBatch()
- Publisher emits multiple update counts, one per statement
- Calling thread not blocked
- But any other call to the Connection or its dependents will block while the async operation is in-flight

oracle.jdbc.OraclePreparedStatement

```
public Publisher<OracleResultSet> executeQueryAsyncOracle();
```

- Async version of PreparedStatement.executeQuery()
- Publisher emits a single result set.
- Calling thread not blocked
- But any other call to the Connection or its dependents will block while the async operation is in-flight

oracle.jdbc.OracleResultSet

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

- Publishes the reified rows of the ResultSet
- Function argument f reifies the rows.
  The OracleRow is only valid during the call to f
- publisherOracle closes the ResultSet

oracle.jdbc.OracleResultSet (continued)

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

- OracleRow extends Cloneable so you can copy the row if you want:

```
Publisher<OracleRow> rowPublisher =
    resultSet.publisherOracle(OracleRow::clone);
```


oracle.jdbc.OracleResultSet (continued)

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

- Or construct some other Object, ie an Employee, if appropriate:

```
Publisher<Employee> employeePublisher =
    resultSet.publisherOracle(this::mapRowToEmployee);
```


oracle.jdbc.OracleRow

```
public <T> T getObject(int index, Class<T> type);
public <T> T getObject(String name, Class<T> type);
public OracleRow clone();
```

- Uses the generic getter methods added to JDBC 4.2
- OracleRows passed to the reify method are backed by internal data structures. No copying
- Call clone() if you want to copy or construct another object

oracle.jdbc.OracleConnection

```
public Publisher<Success> closeAsyncOracle();
public Publisher<Success> commitAsyncOracle();
public Publisher<Success> rollbackAsyncOracle();
```

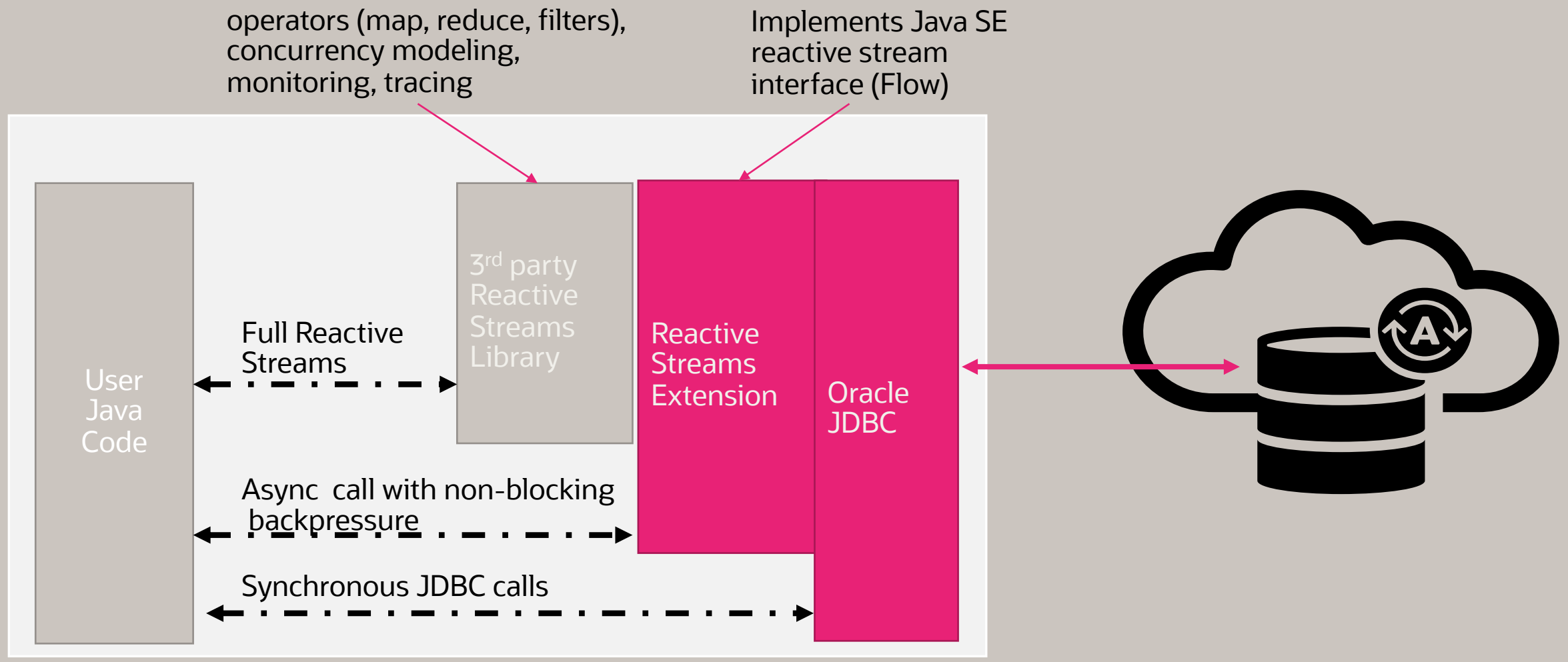
- Async versions of Connection.close(), Connection.commit(), and Connection.rollback()
- Publisher emits a single Success enum.

Notes

- Support for reading and writing BFILE, BLOB, and CLOB
- Only one async operation at a time. Subsequent calls to the Connection or its dependents block except for cancel, isClosed, etc
- This is not ADBA. Cannot queue operations
- 80/20 rule. 80% of the benefit for 20% (or less) of the API
- Compatible with reactive stream libraries that support Flow
- In the long term, fibers are the answer

Async Demo

Database Access with Oracle JDBC



Q & A

What's Ahead

Wednesday

5:00-5:45 Developing and Deploying Oracle Database Applications in Kubernetes

Thursday

9:00-9:45 Microservice Essentials: Kubernetes and Ecosystem, Data, and Transaction Patterns

12:15-1:00 A Database Proxy for Transparent HA, Performance, Routing, and Security

Session Survey

Help us make the content even better. Please complete the session survey in the Mobile App.