ORACLE

Developing Oracle Database applications in .NET

Overview

Witold Swierzy
EMEA Converged Database Expert
August 2023

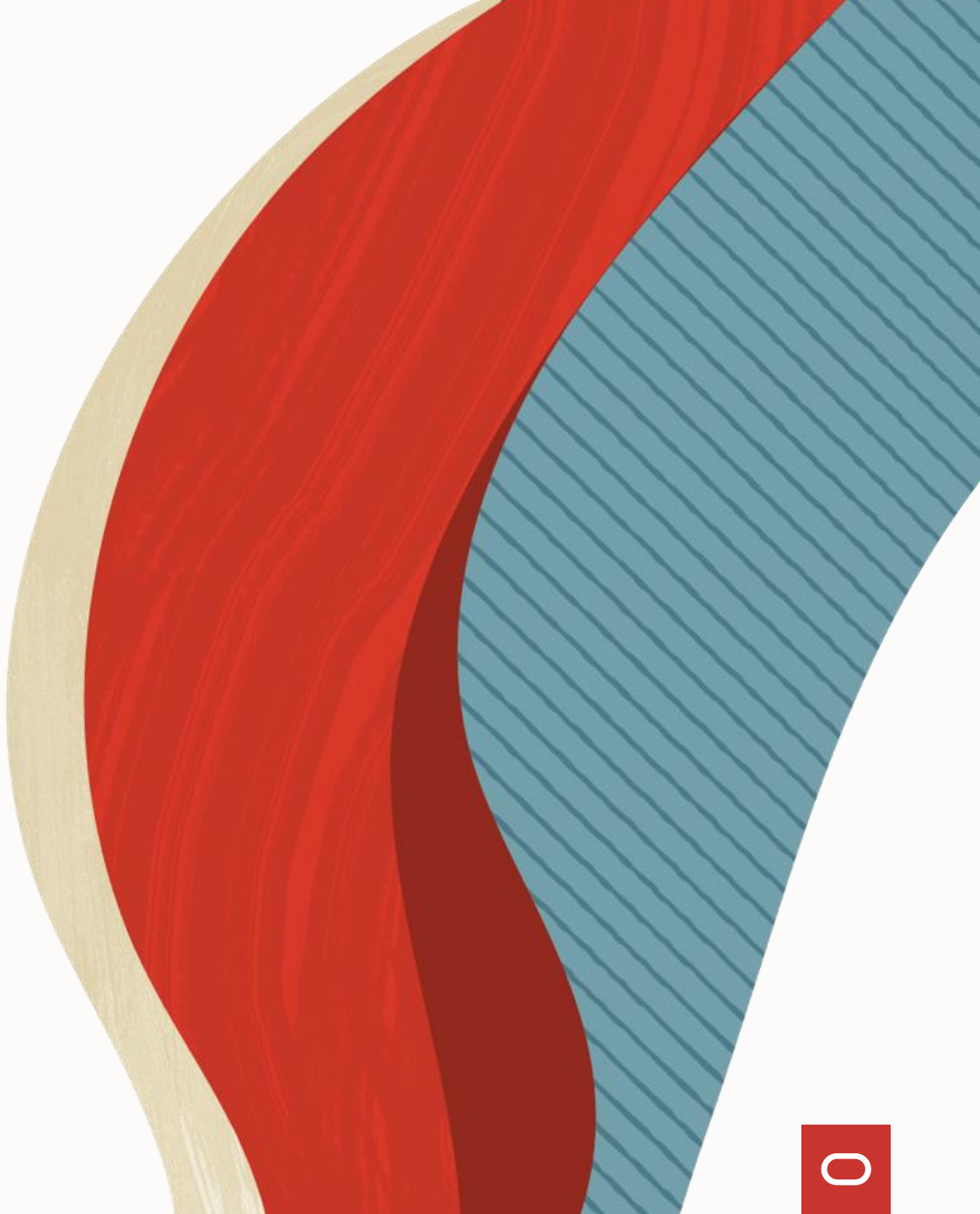
Safe harbor statement

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.

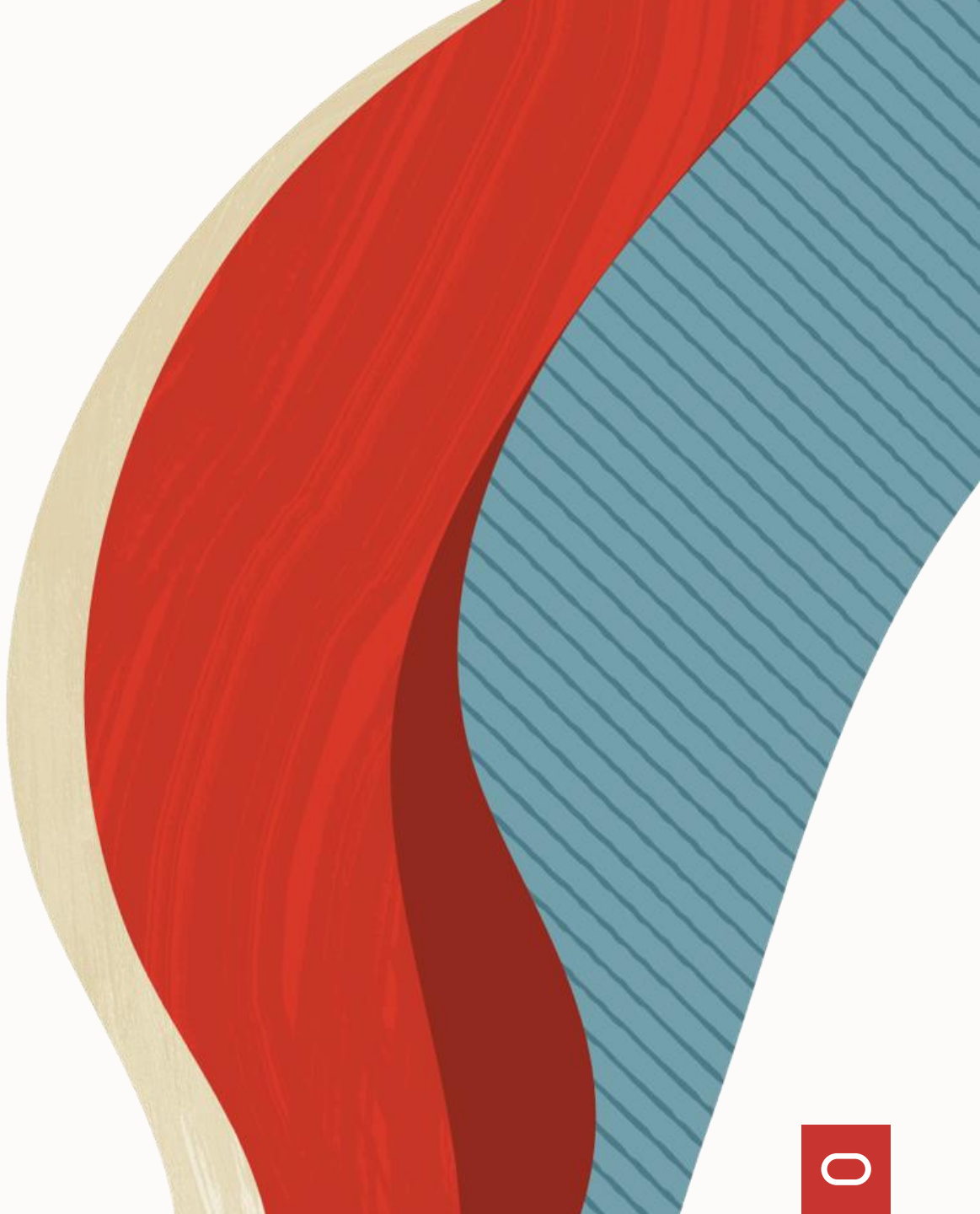
Agenda

- Fundamentals
- Transaction Event Queues
- HA/DR support



Agenda

- **Fundamentals**
- Transactional Event Queues
- HA/DR support



Categories of .NET Oracle Data Provider

ODP.NET Managed Driver	ODP.NET Unmanaged Driver	ODP.Net Core
Fully Managed ADO Provider	Based on Oracle client software installation	Multiplatform provider for MS .NET Core
Does not require installation of Oracle Client software.	Full functionality as a result of the fact, that it uses Oracle client software	Functionality similar to OP.NET Managed Driver
Equivalent of Oracle JDBC Thin Driver	Equivalent of Oracle JDBC OCI Driver	Subset of ODP.NET managed APIs Details here

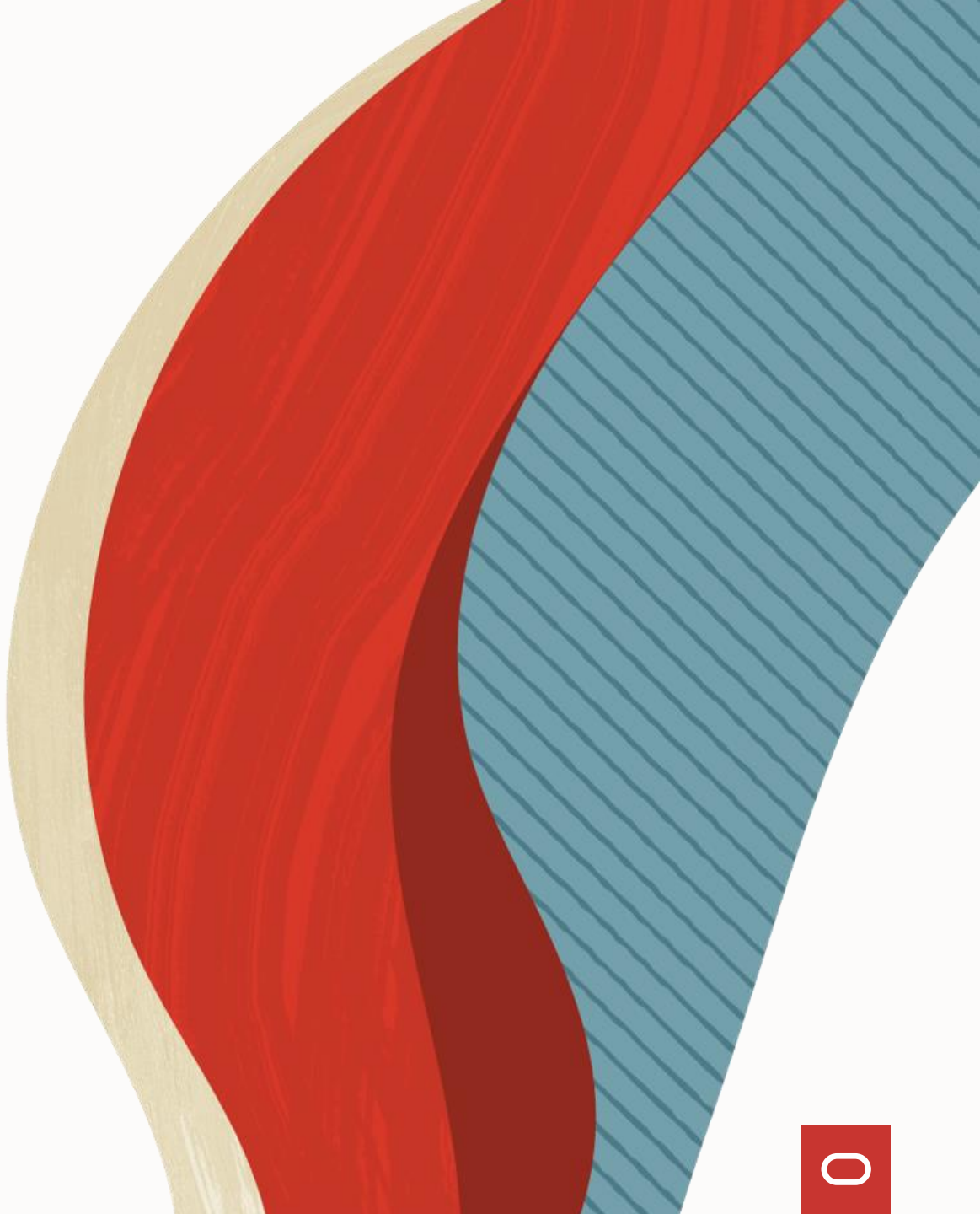
Drivers availability

Managed & Core drivers: Nuget

- Core Driver
- Managed driver

Unmanaged driver:

- Oracle downloads



Comparison of drivers

- APIs availability detailed comparison
- Configuration comparison
- Differences in support for distributed transactions

Connecting to the database

- OracleConnection class
- Username, password can be provided as string parameter to its constructor

```
OracleConnection dbConnection = new OracleConnection("User Id="+dbUsername+";" +
                                                     "Password="+dbPassword+";" +
                                                     "Data Source="+dbURL);

dbConnection.Open();
...
dbConnection.Close();
```

Transaction support

- Full Oracle transaction support
- Exception
  - .NET Core driver and distributed transaction. Documented here.
- OracleTransaction class

```
OracleTransaction sqlTxn = dbConnection.BeginTransaction();
sqlTxn.Commit();
sqlTxn.Rollback();
```

- By default database connection works in autocommit mode

Executing non-query statements (DMLs/DDLs/DCLs)

- OracleCommand class

```
OracleCommand sqlCmd = dbConnection.CreateCommand();
sqlCmd.CommandText = "SQL command";
sqlCmd.executeNonQuery();
```

Executing SELECT statements

- OracleCommand class

```
OracleCommand sqlCmd = dbConnection.CreateCommand();
sqlCmd.CommandText = "SELECT statement";
```

- OracleDataReader class for the result set processing

```
OracleDataReader reader = sqlCmd.executeReader();
while (reader.Read()) {
    // result set processing, example
    Console.WriteLine(reader.getString(0)+reader.getString(1));
}
...
```

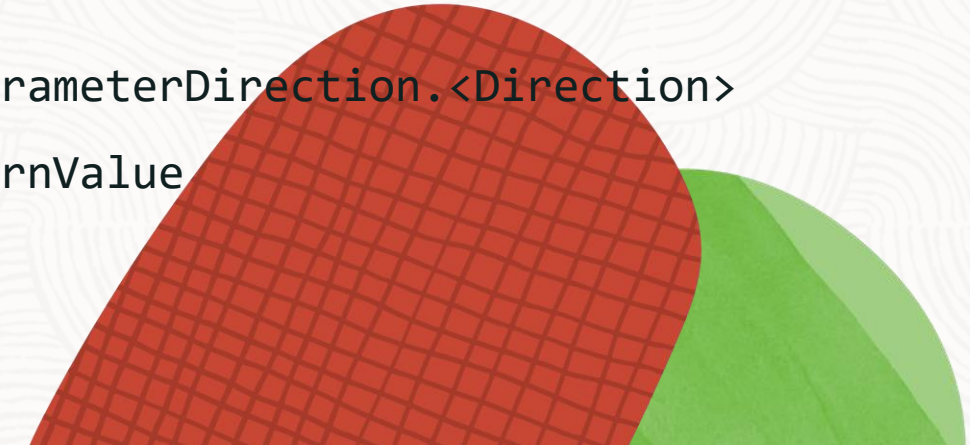
Stored procedures and functions execution

- OracleCommand class

```
OracleCommand sqlCmd = dbConnection.CreateCommand();
sqlCmd.CommandText = "name of the stored program";
sqlCMd.CommandType = System.Data.CommandType.StoredProcedure;
```

- Parameters can be defined using OracleParameter class

```
OracleParameter param
= new OracleParameter("name_of_stored_program_parameter",
                      OracleDbType.Type,
                      length);
param.Direction = System.Data.ParameterDirection.<Direction>
//Direction: Input, Output, ReturnValue
```



Data Type mappings

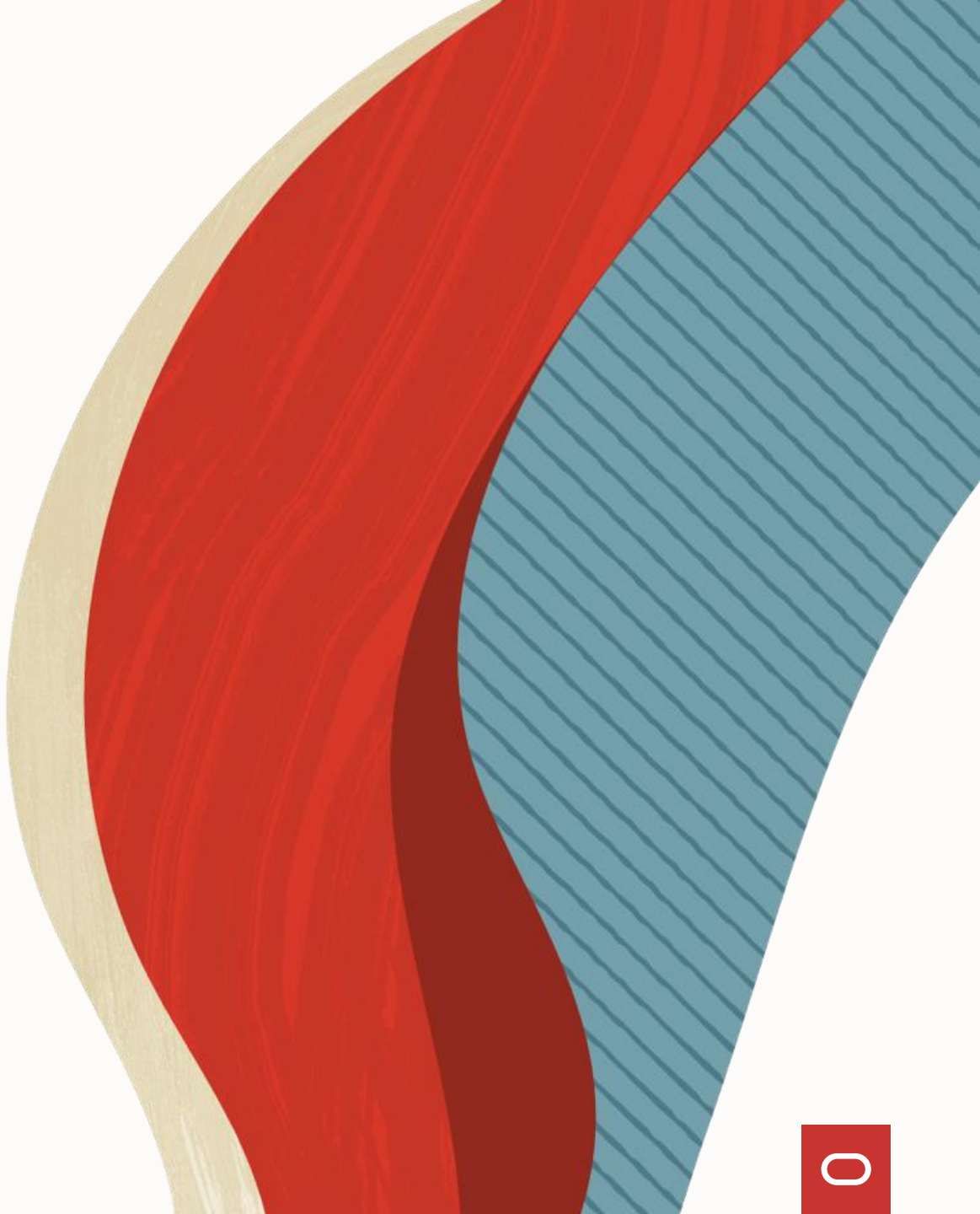
- OracleDbType class contains set of constants providing information about expected Oracle data type
- Examples:
  - Varchar2
  - Blob
  - Raw
  - Date
  - ...

User-defined Oracle database data types

- Nested tables
- Collections
- Mapping rules documented here

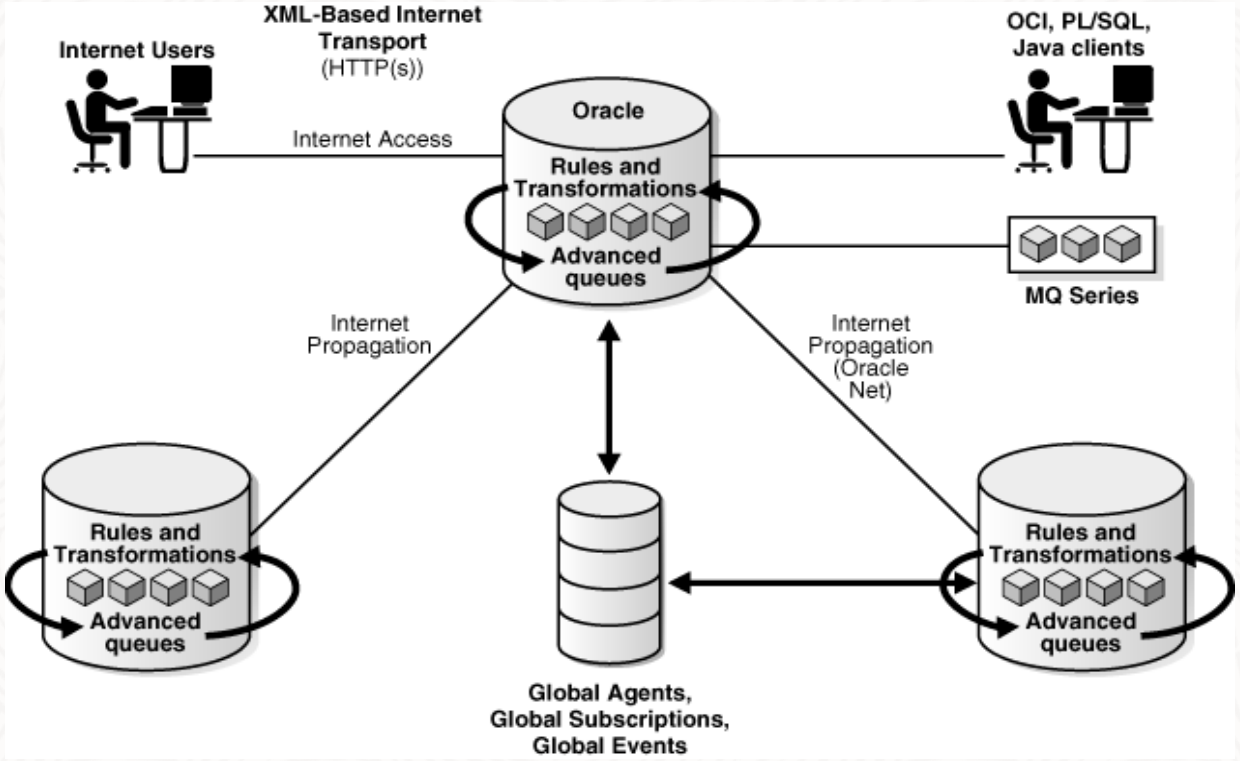
Agenda

- Fundamentals
- **Transactional Event Queues**
- HA/DR support



Oracle Database
Transactional Event Queues

- Database-integrated message queuing functionality
- Provides the message management and communication needed for application integration. In an integrated environment, messages travel between the Oracle Database server, applications, and users



Oracle Database
Transactional Event Queues

- All .NET data providers support TEQ
- A .NET application can utilize TEQ functionality in two ways
  - By using PL/SQL TEQ interface
  - By using native support for TEQ
- Examples in this presentation use the following queue definition

```
begin
  dbms_aqadm.create_queue_table (
  queue_table        => 'sample_message_queue_table',
  queue_payload_type => 'RAW');

  dbms_aqadm.create_queue (
  queue_name  => 'sample_message_queue',
  queue_table => 'sample_message_queue_table' );

  dbms_aqadm.start_queue(queue_name => 'sample_message_queue');
end;
/
```

Transactional Event Queues

Using PL/SQL TEQ interface in a .NET application

- TEQ PL/SQL interface
  - DBMS_AQADM package to manage queues
  - DBMS_AQ package to send/propagate/dequeue/consume messages
- .NET data providers support PL/SQL stored procedures calls
- This is the simplest and oldest way a .NET application can utilize TEQ Oracle Database functionality
- Example uses PL/SQL code as well as .NET code attached to this presentation

Transactional Event Queues

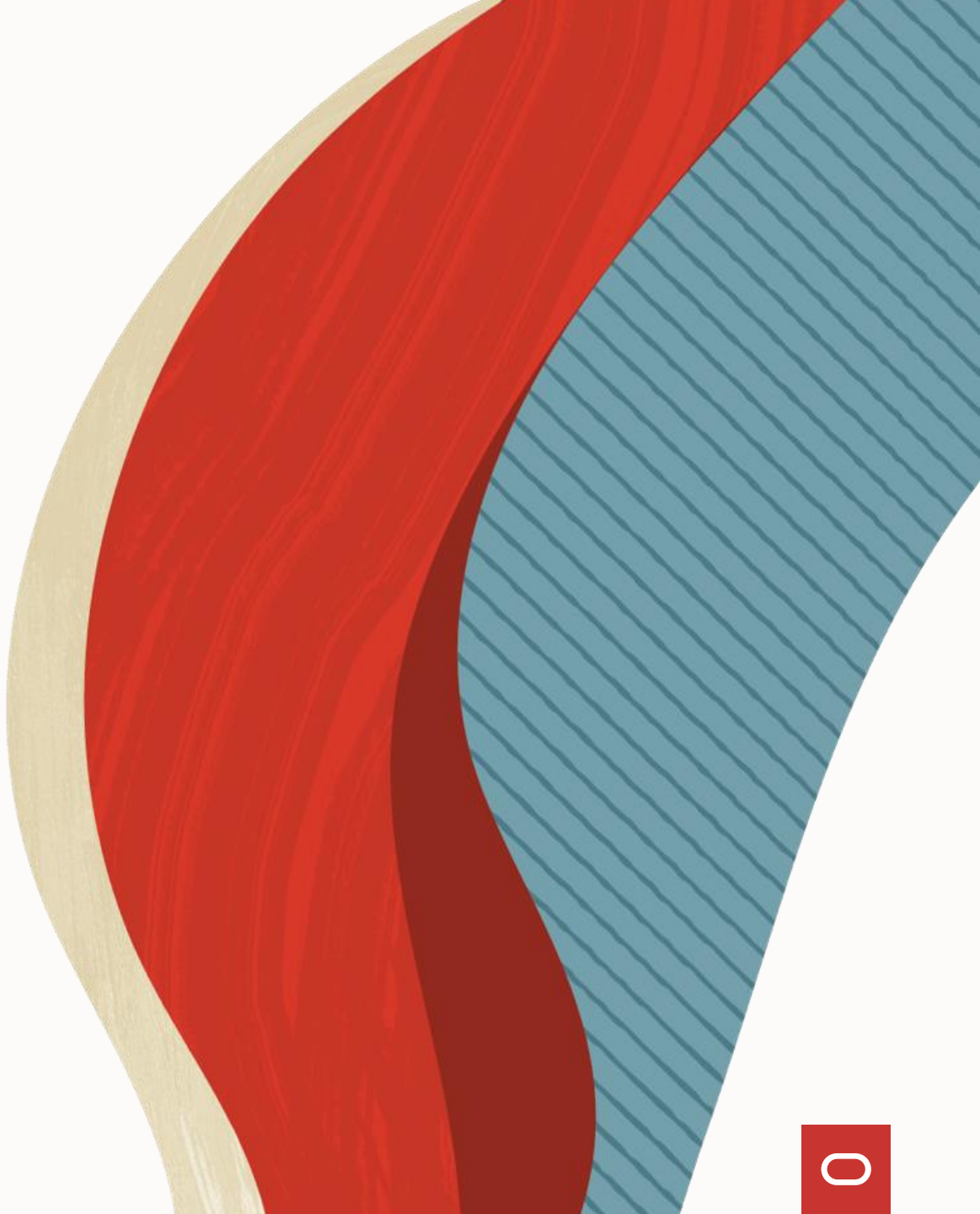
Using .NET provider native support for TEQ

- Available in all .NET providers
- Fully transactional, can use OracleTransaction object to manage the transactional context
- Uses OracleAQQueue class

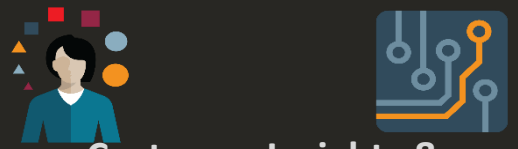
```
OracleTransaction txn = dbConnection.BeginTransaction();
OracleAQQueue sample_message_queue = new OracleAQQueue("sample_message_queue",
                                                       dbConnection);
sample_message_queue.MessageType = OracleAQMessageType.Raw;
OracleAQMessage msgIn = new OracleAQMessage();
msgIn.Payload = Encoding.UTF8.GetBytes("Another test payload");
sample_message_queue.Enqueue(msgIn);
Console.WriteLine("A message has been put into the test queue");
txn.Commit();
txn = dbConnection.BeginTransaction();
OracleAQMessage msgOut = sample_message_queue.Dequeue();
txn.Commit();
Console.WriteLine(Encoding.UTF8.GetString(msgOut.Payload as byte[]));
```

Agenda

- Fundamentals
- Transactional Event Queues
- **HA/DR support**



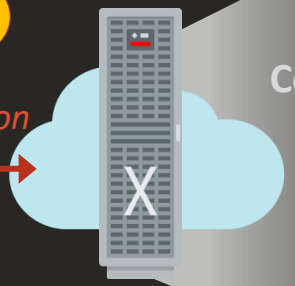
Oracle Maximum Availability Architecture (MAA)



Customer Insights &
Expert Recommendations

Reference Architectures

Replication



Production Site

Replicated Site

HA Features, Configurations & Operational Practices

Deployment Choices

Generic Systems

Engineered Systems

DBCS ExaCS/ExaCC

Autonomous DB

Continuous Availability

Application Continuity

Global Data Services

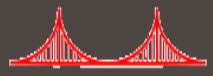
Data Protection

Flashback

RMAN + ZDLRA

Active Replication

Active Data Guard

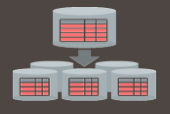


GoldenGate

Scale Out

RAC

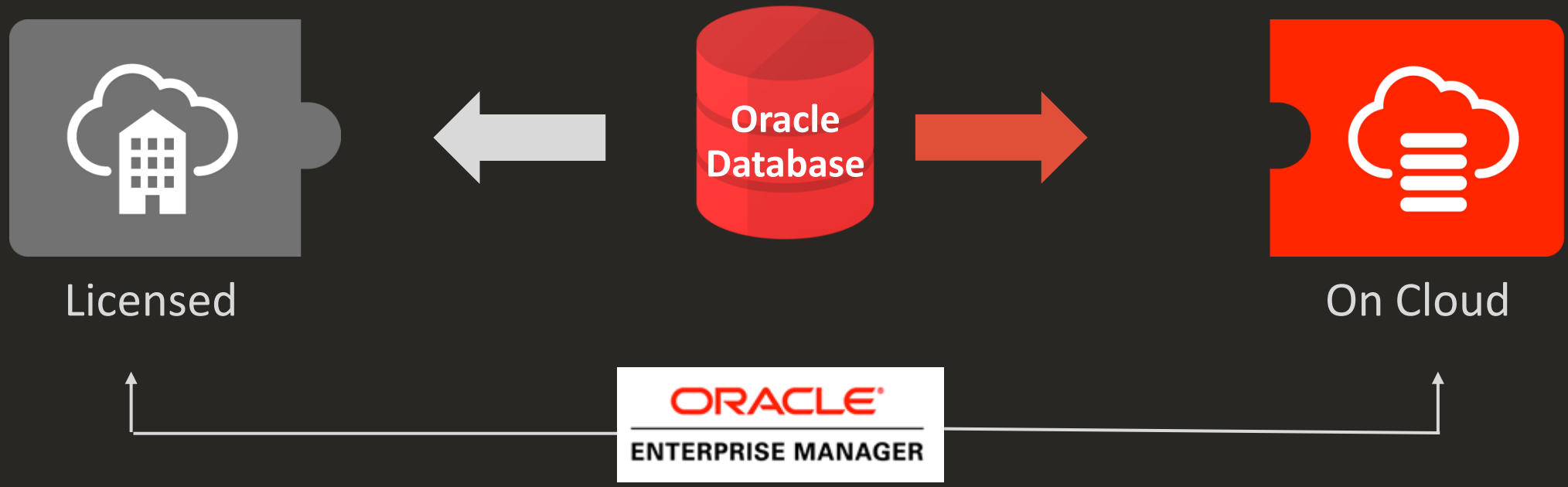
ASM



Sharding

Oracle HA/MAA – The Broader Picture

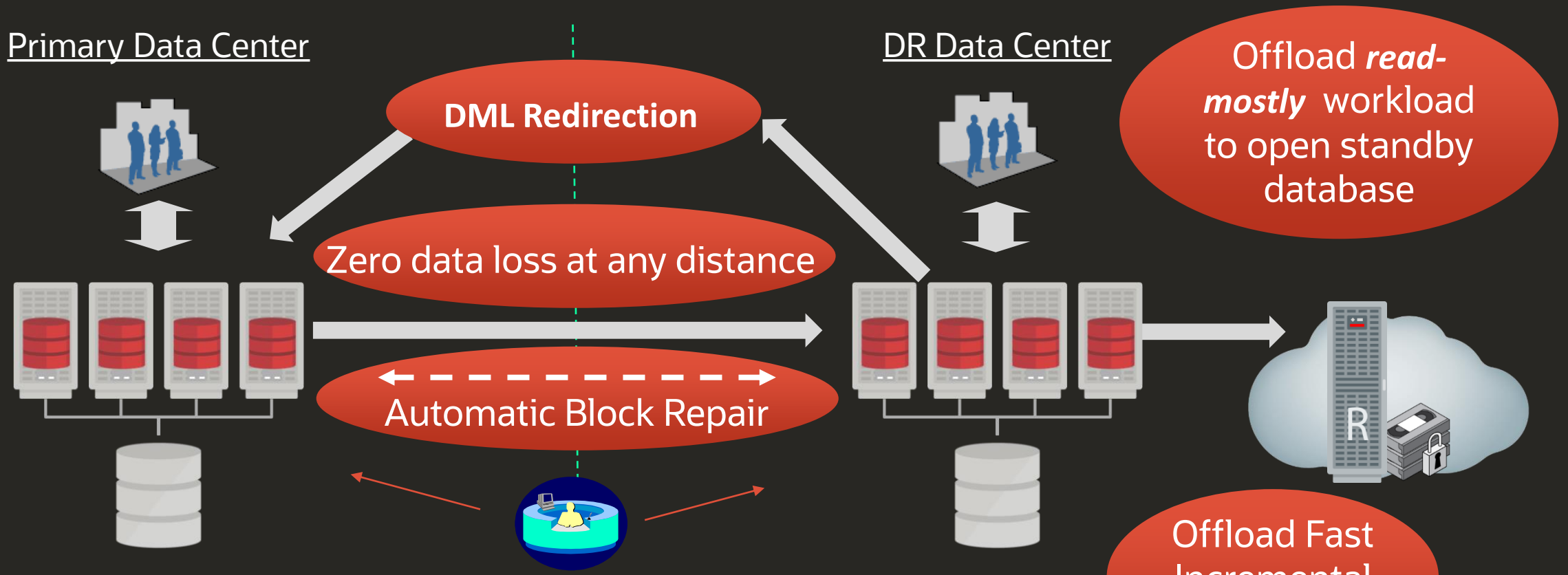
Designed to Address the Complete Range of Business Requirements



Common Platform – Licensed, Cloud, and Hybrid Cloud

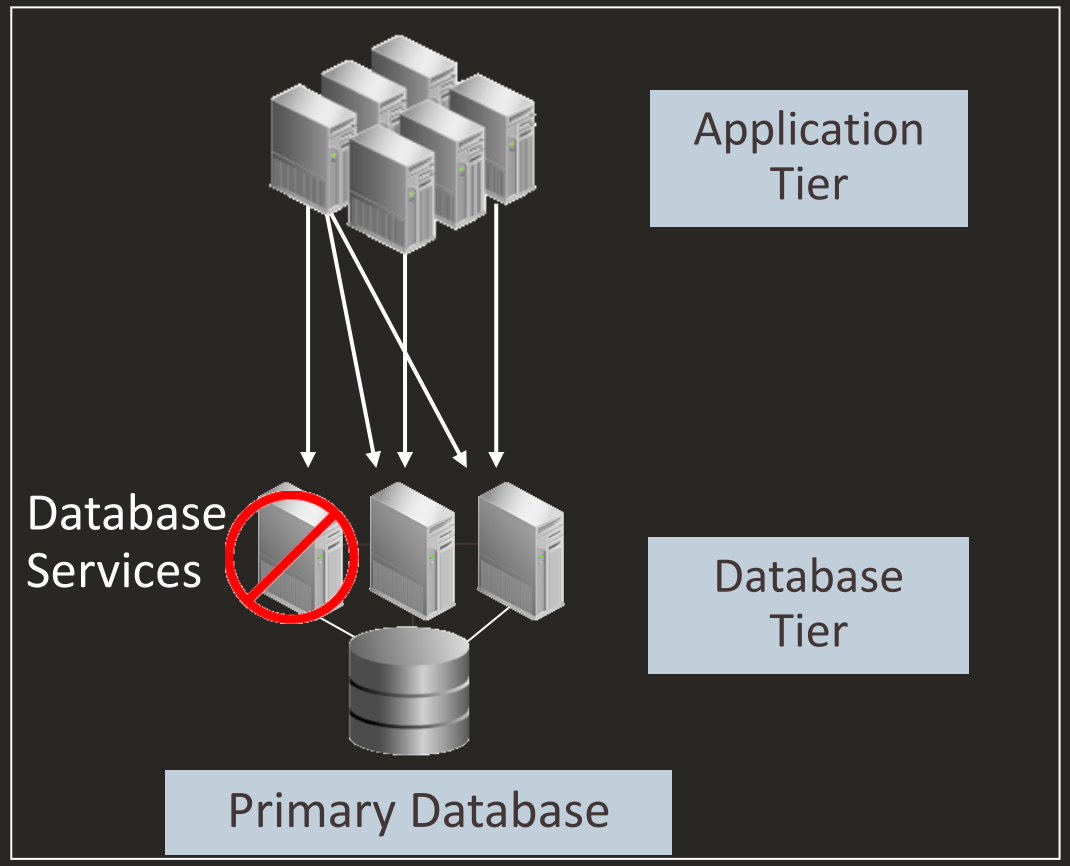
Big Differentiator

Active Data Guard: Advanced Capabilities



Oracle Real Application Clusters (RAC)

Node Failure, Instance Failure, Rolling Maintenance

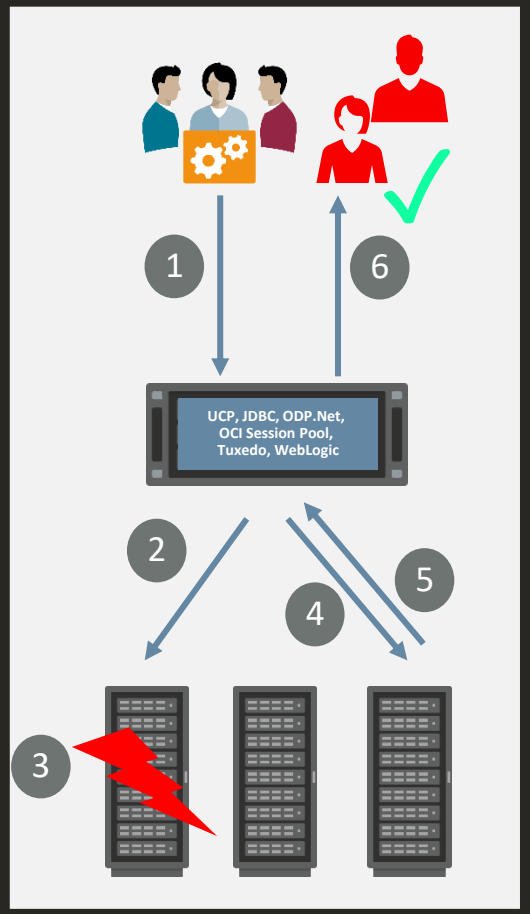


- Utilizes two or more instances of an Oracle Database concurrently
- Very Scalable
  - All instances active; Add capacity online; Ideal for database consolidation
- Highly Available
  - Auto-failover of services to an already running instance; Outage is transparent to user, in-flight transactions succeed; Zero downtime rolling maintenance

Transparent Application Continuity

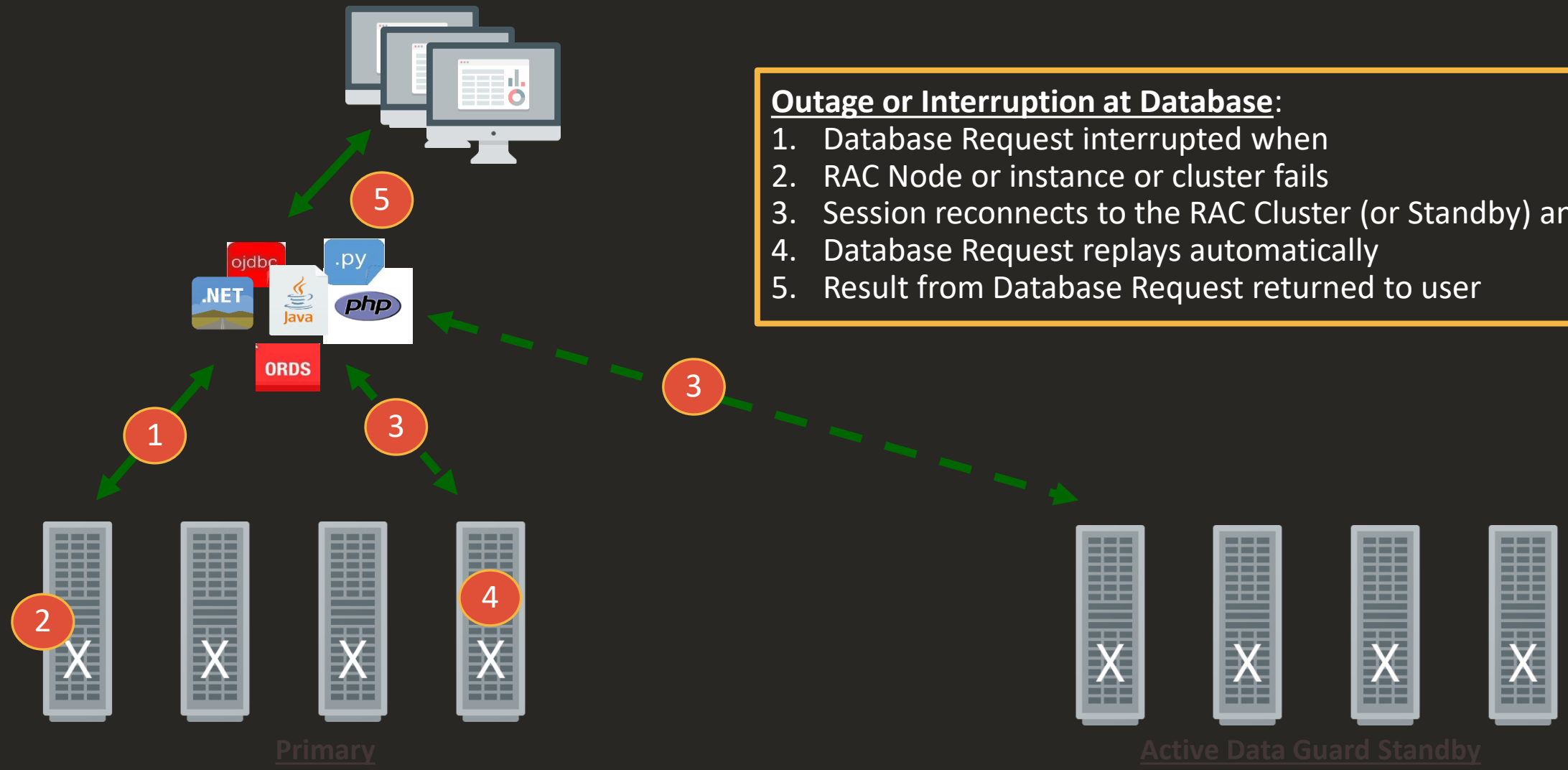
Preserving and Replaying Database Requests Across Interruptions

- Failures in the database stack & connectivity lead to
  - Interruptions in user sessions
  - Unknown state of transactions
- Application Continuity masks errors from applications by recovering session state and replaying in-flight requests
  - Replay performed on a surviving RAC instance or Data Guard standby
    - If transaction already committed: no action;
    - if replay successful: app continues;
    - if request non-recoverable: app handles errors usual way
  - Eliminates the need to create custom exception code
- Application Continuity extends Oracle's HA capabilities end-to-end – from bottom to top, without code changes.



Continuous Availability

Unplanned Outages without Client Interruptions, with Application Continuity



Outage or Interruption at Database:

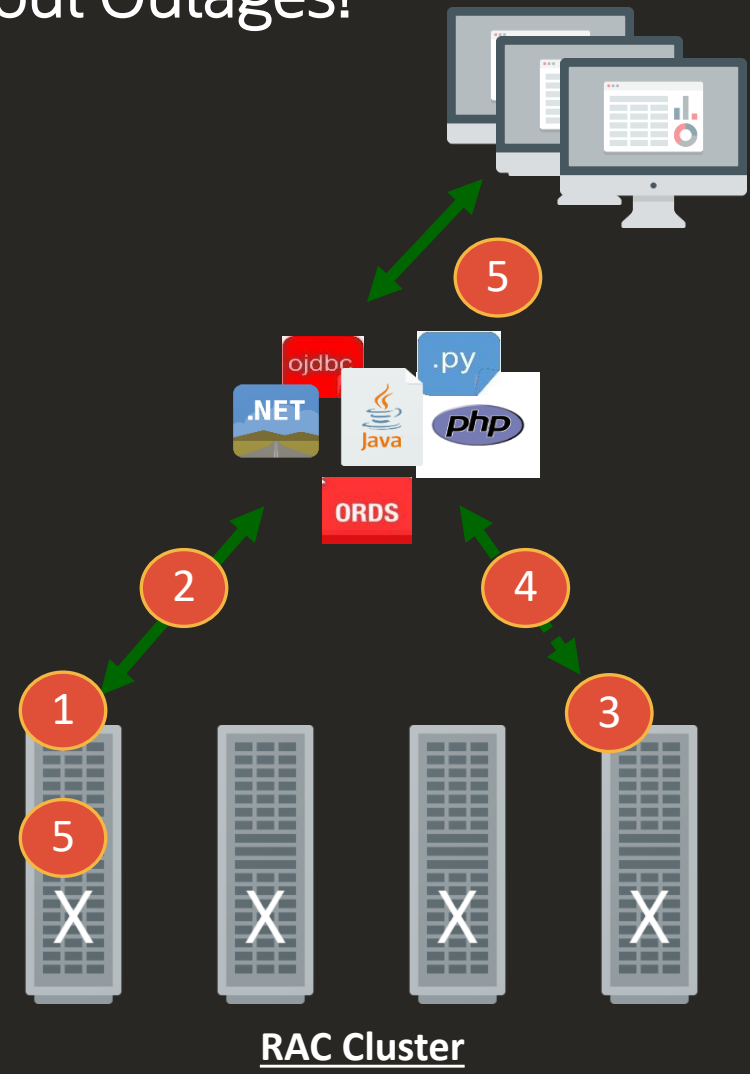
1. Database Request interrupted when
2. RAC Node or instance or cluster fails
3. Session reconnects to the RAC Cluster (or Standby) and
4. Database Request replays automatically
5. Result from Database Request returned to user

Continuous Availability

Planned Maintenance (patching, etc) without Outages!

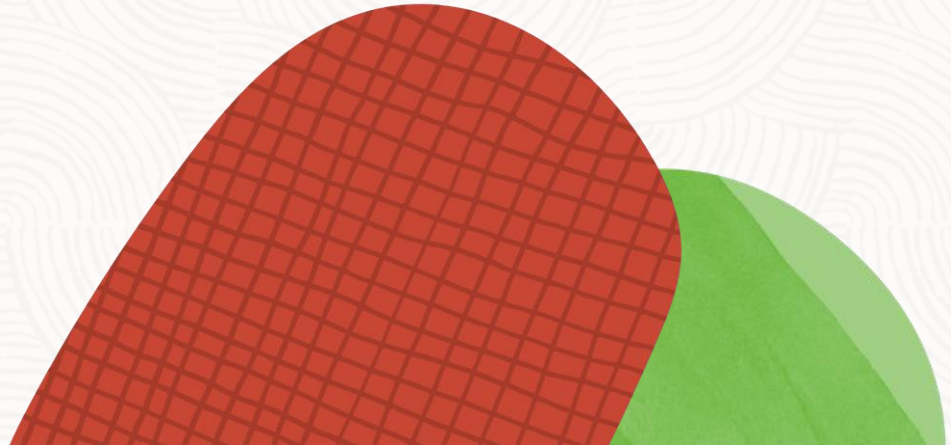
Planned Maintenance (no Outages!):

1. Database Service is relocated or stopped
2. Sessions connected via that service drain their work
3. Service starts on other RAC node
4. Session reconnects to Service on other node, and
5. Maintenance activities can start on the node



Application Continuity

- All ODP.NET provider types, core, managed, and unmanaged, support Application Continuity and Transparent Application Continuity
- Can be enabled by setting the following connection attribute
  - Application Continuity=true
  - This attribute is set to TRUE by default
  - Source: Oracle **23c** .NET DP documentation
- Available with Oracle RAC, Oracle RAC One Node, Oracle Active Data Guard, and both dedicated and shared Oracle Autonomous Database.



Summary

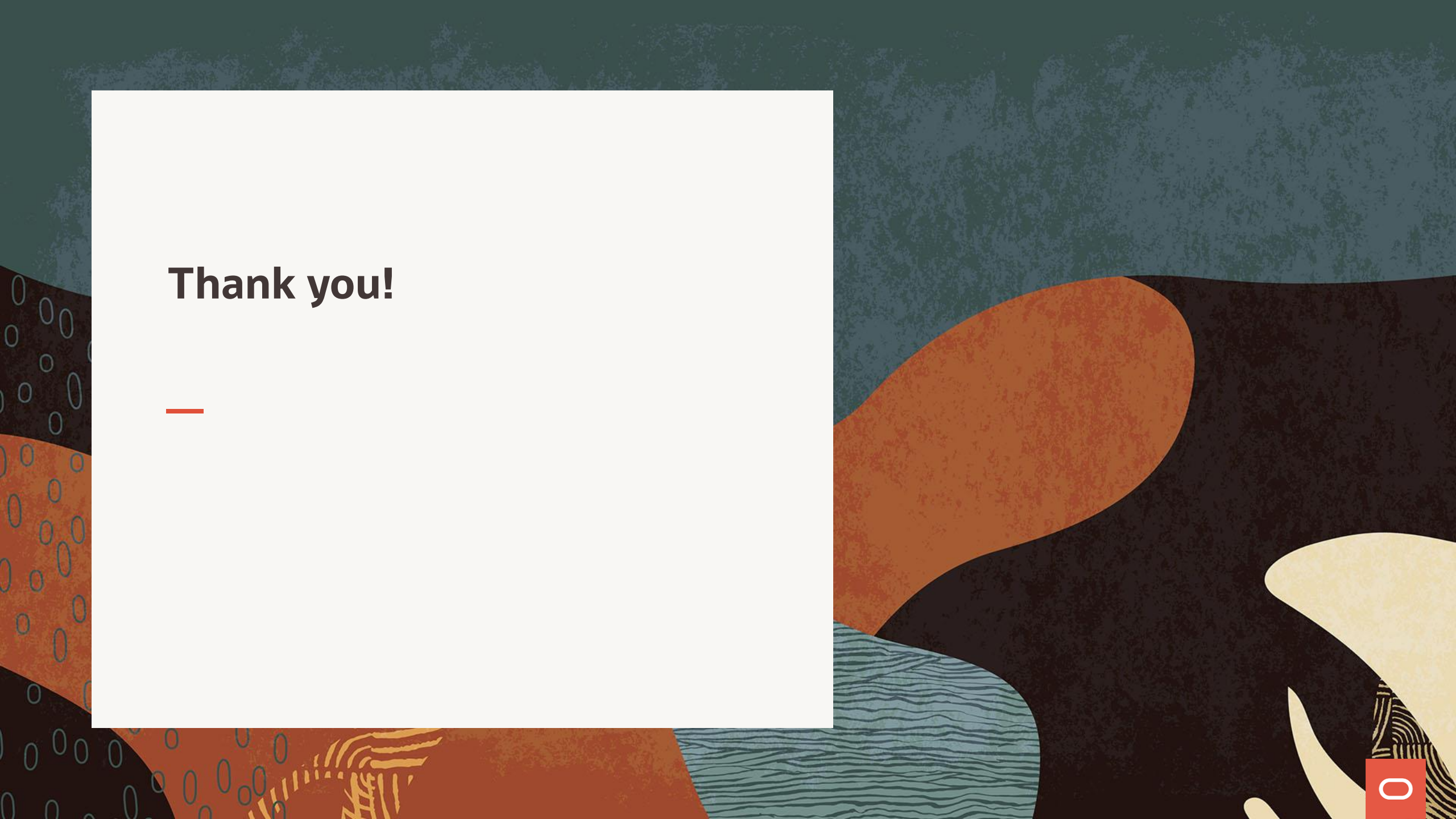
Oracle Database fully supports .NET application development

- Full ACID transaction support
- DML/DDL/DCL/Queries and result set processing
- Full support for Advanced Queuing/Transactional Event Queues
- Full support for modern, NOSQL datatypes
  - JSON, XML
- Full support for HA/DR architectures
  - Development of mission-critical apps

Interresting links

- Oracle **23c** .NET DP documentation
- APIs availability detailed comparison
- Configuration comparison
- Differences in support for distributed transactions
- Nuget Core Driver
- Nuget Managed driver
- Unmanaged Oracle Driver downloads

Thank you!




ORACLE