
ORACLE®

JavaOne™

ORACLE®

# Supercharge your Code to get Optimal Database Performance

Gerald Venzl
Senior Principal Product Manager
Oracle Database Development

@GeraldVenzl

Java Your (Next)

JavaOne ORACLE

# 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, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

# **Gerald's** Safe Harbor Statement

**Nothing said here is a guarantee for solving your particular problem.** While the techniques presented here can have a huge positive impact on performance it is not guaranteed that they will fix your particular problem.

**Performance is relative.** It depends on a huge amount of different factors and can't be compared 1:1. The numbers in this deck reflect a tiny 2 GB VirtualBox environment with only one virtual CPU, keep that in mind.

# Program Agenda

1. Introduction
2. What is performance
3. Oracle Database Performance Statistics
4. Commits – saver of your data
5. Row by row = slow by slow
6. Bind vari… – WHAT?

# Program Agenda

1. Introduction
2. What is performance
3. Oracle Database Performance Statistics
4. Commits – saver of your data
5. Row by row = slow by slow
6. Bind vari… – WHAT?

# Introduction

# Introduction

# Introduction

# Introduction

# Introduction

# Introduction


# Introduction


# Introduction


# Introduction


# Program Agenda

1. Introduction
2. What is performance
3. Oracle Database Performance Statistics
4. Commits – saver of your data
5. Row by row = slow by slow
6. Bind vari… – WHAT?

# What is performance

## A simple formula

**Performance = ?**

# What is performance

## A simple formula

**Performance = latency x throughput**

# What is performance

- Latency: The time to process a unit of work
- Throughput: The amount of units of work that can be processed in parallel
- A unit of work is either processed (CPU) or has to wait (I/O)
- Goals are:
  - **Process a unit of work as efficient as possible**
  - **Reduce wait time as much as possible**
  - **Avoid unnecessary resource consumption (CPU & I/O)**

# What is performance

## A simple formula

**Performance = latency x throughput**

**10,000 work units/s = ?**

# What is performance

## A simple formula

**Performance = latency x throughput**

**10,000 work units/s = 1ms x 10wu**

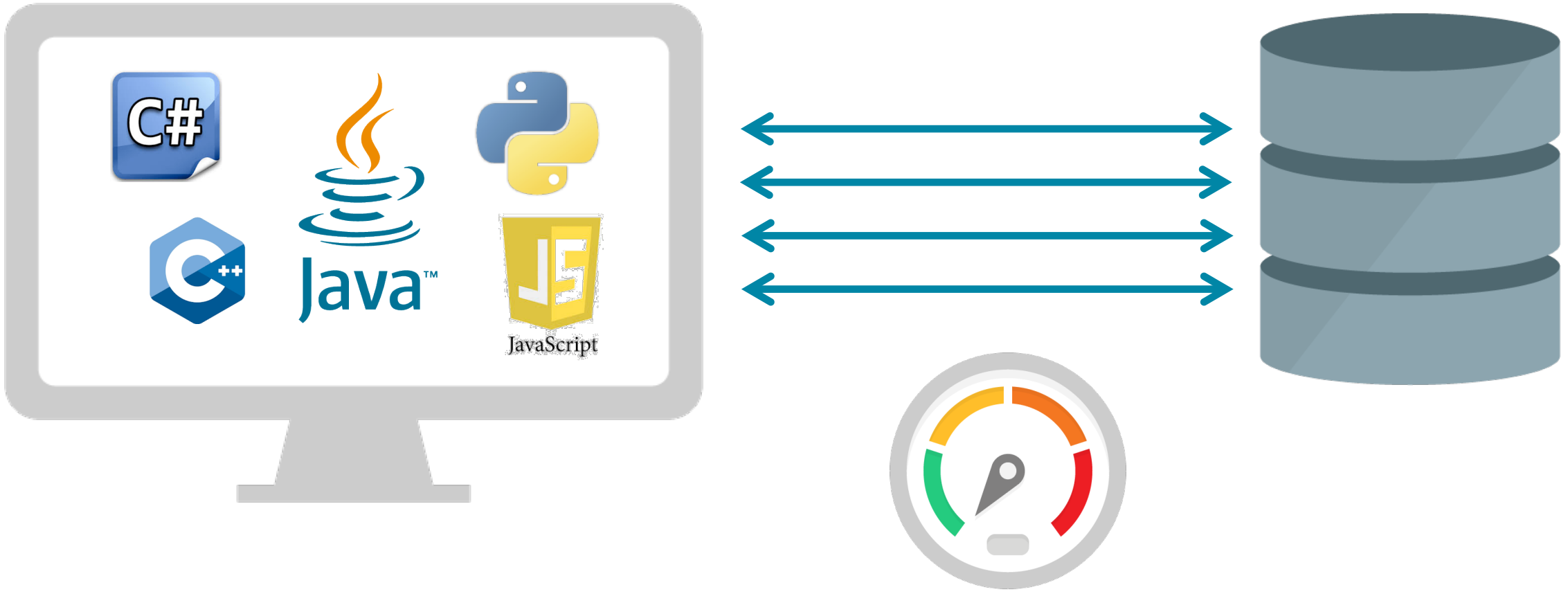
# What is performance

## A simple formula

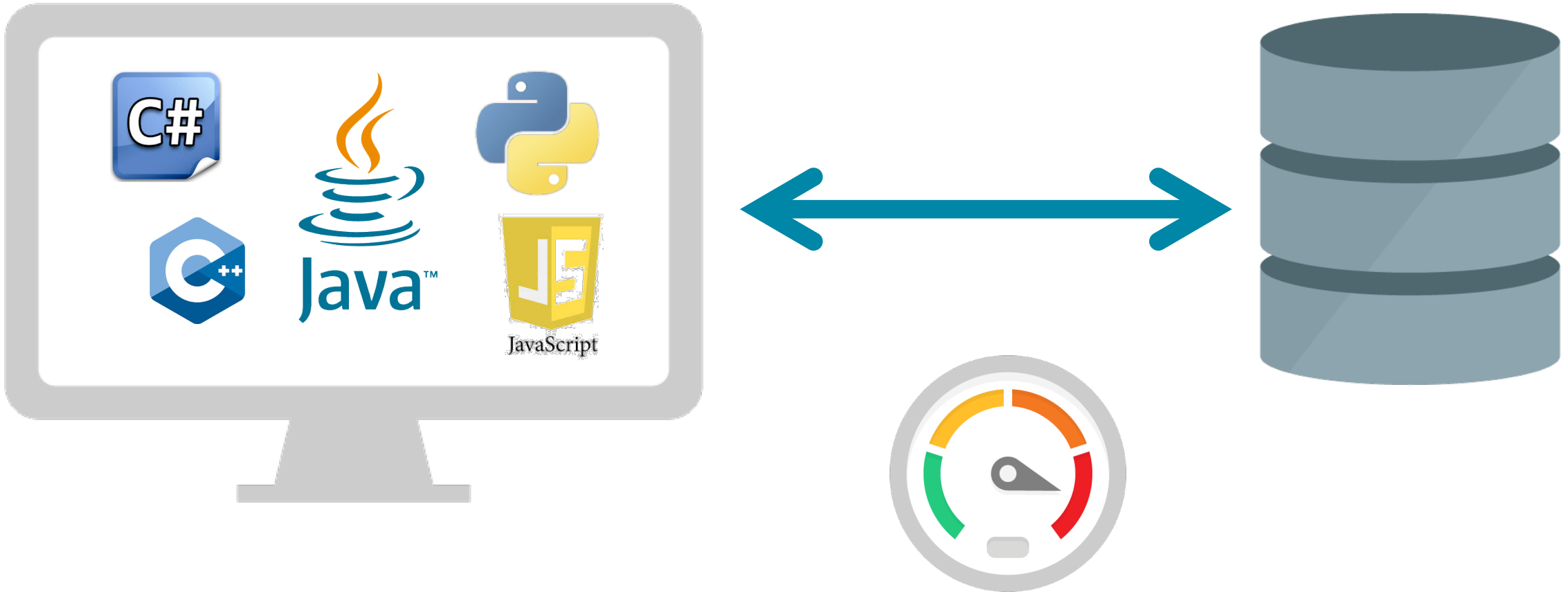
**Performance = latency x throughput**

**10,000 work units/s = 10ms x 100wu**

# What is performance



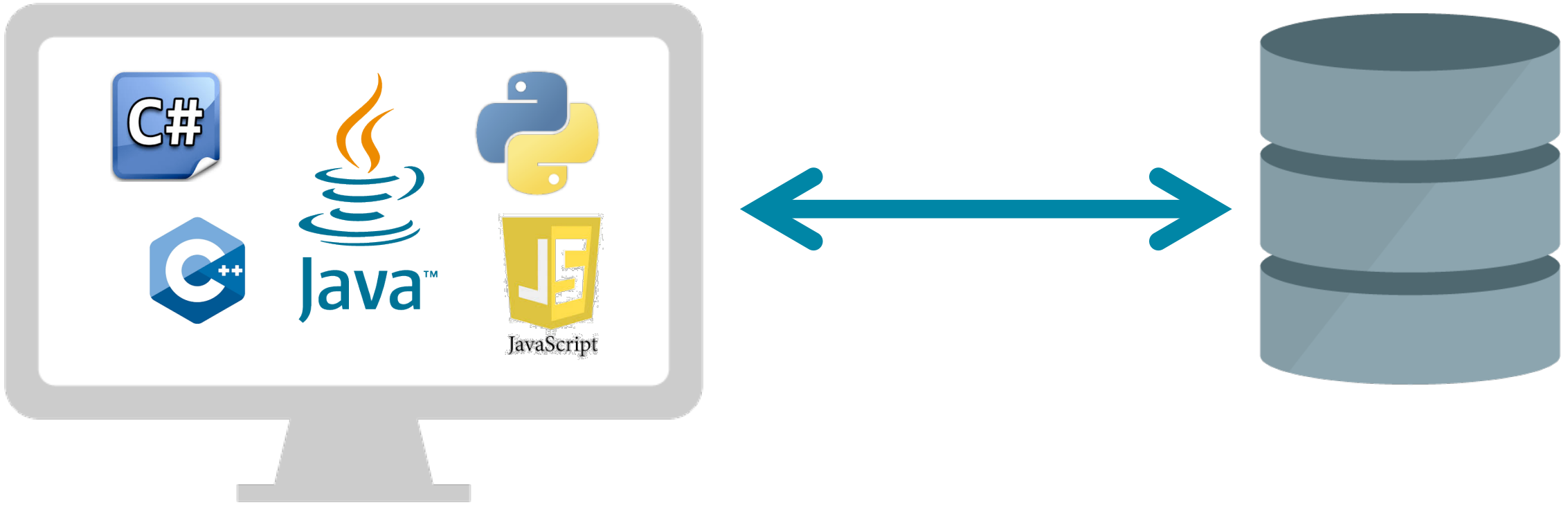
# What is performance



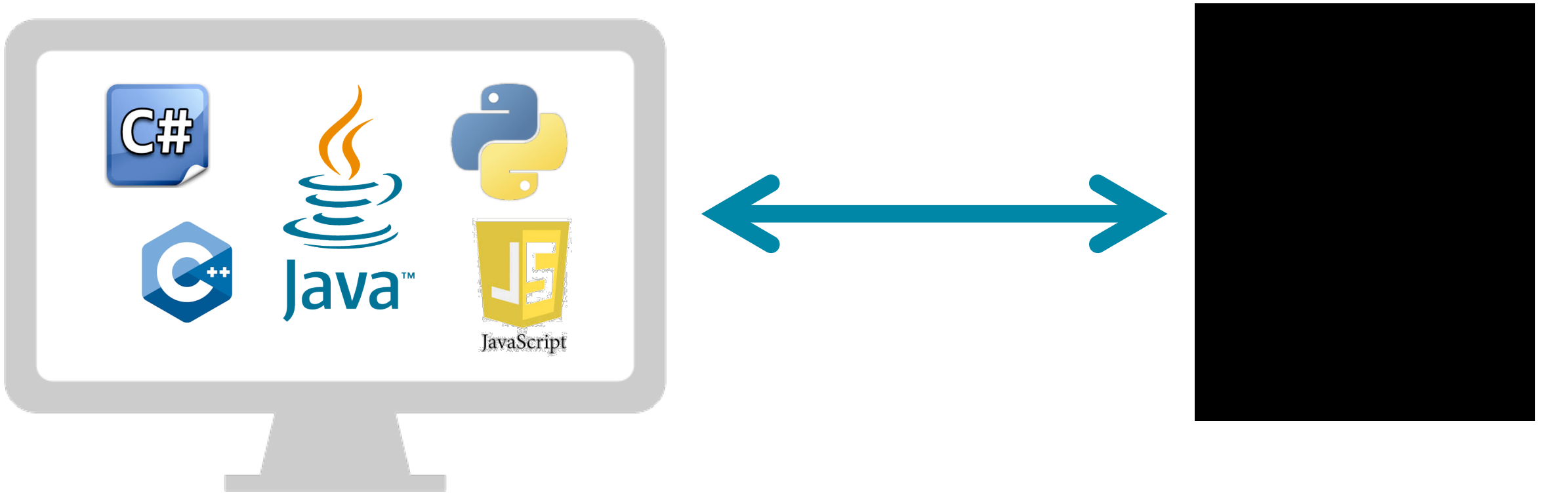
# Program Agenda

1. Introduction
2. What is performance
3. **Oracle Database Performance Statistics**
4. Commits – saver of your data
5. Row by row = slow by slow
6. Bind vari… – WHAT?

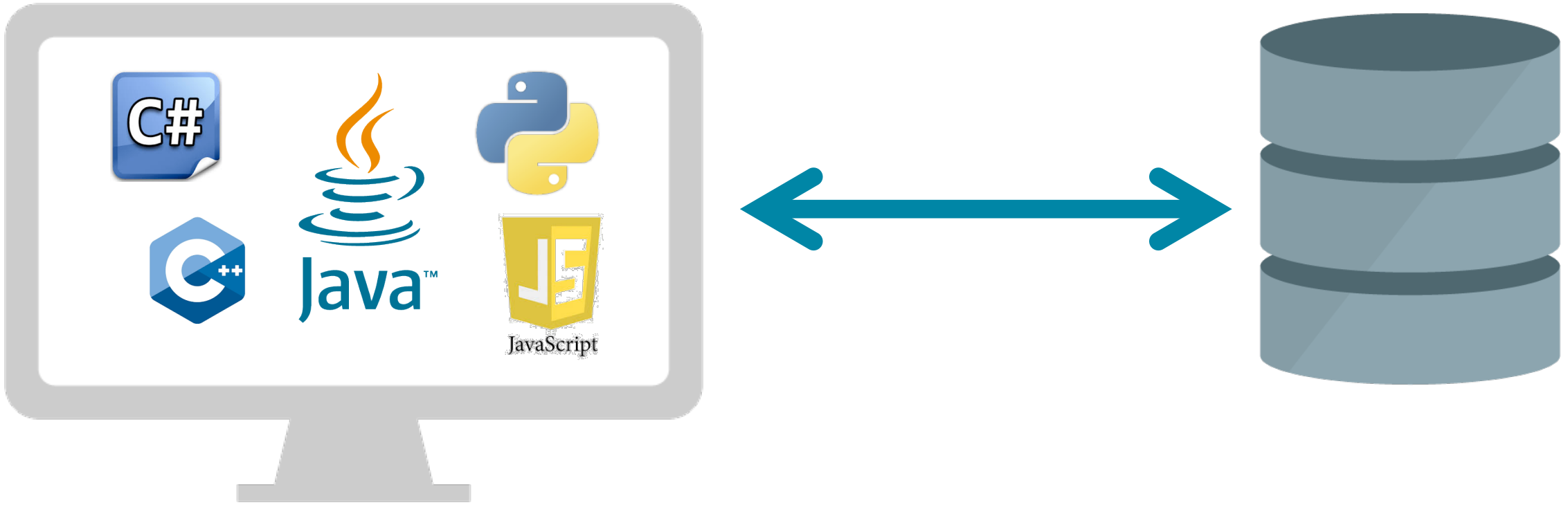
# The database as a



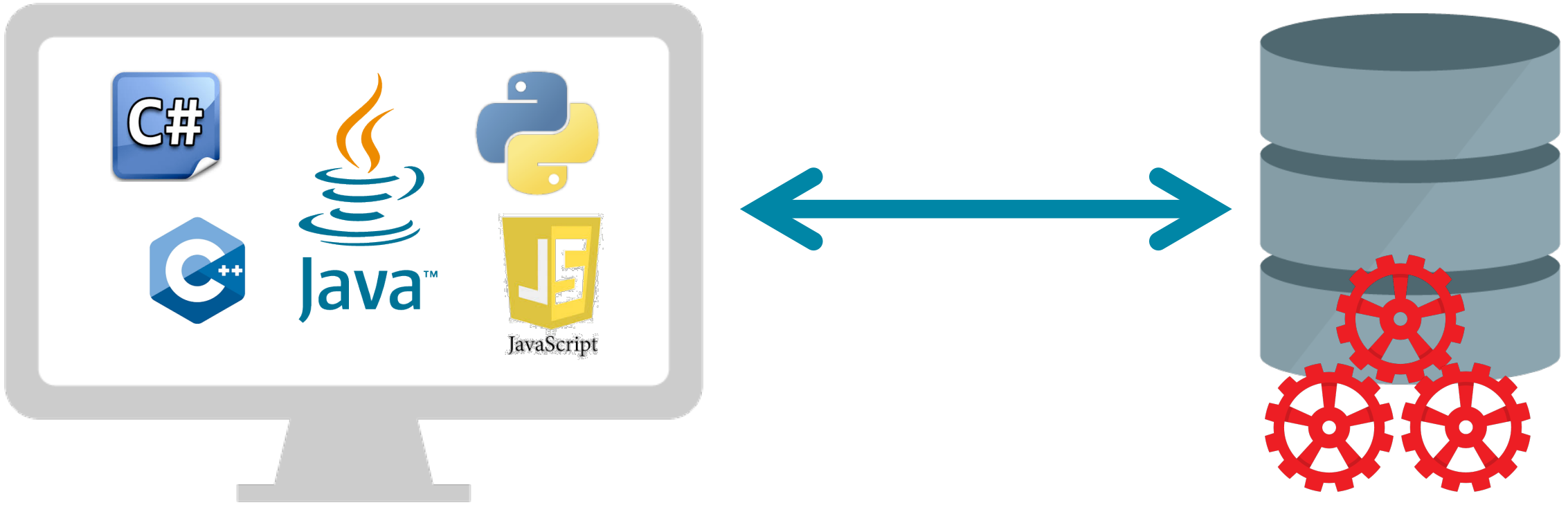
# The database as a **black box**



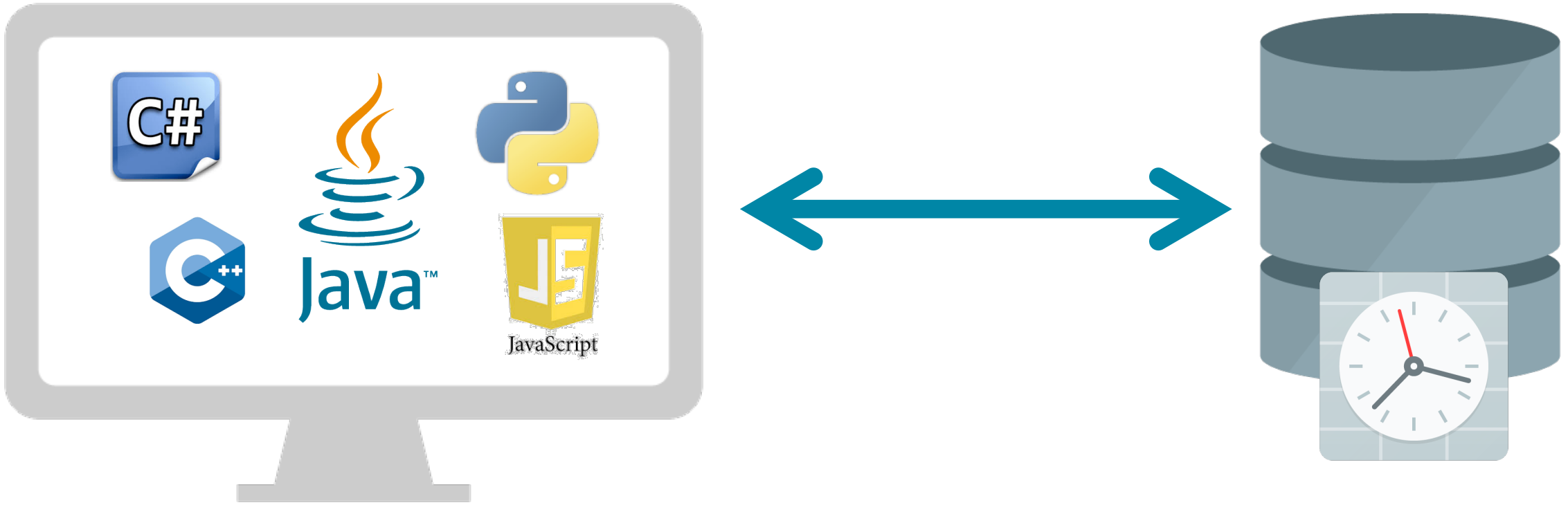
# A database is either



# A database is either **working**

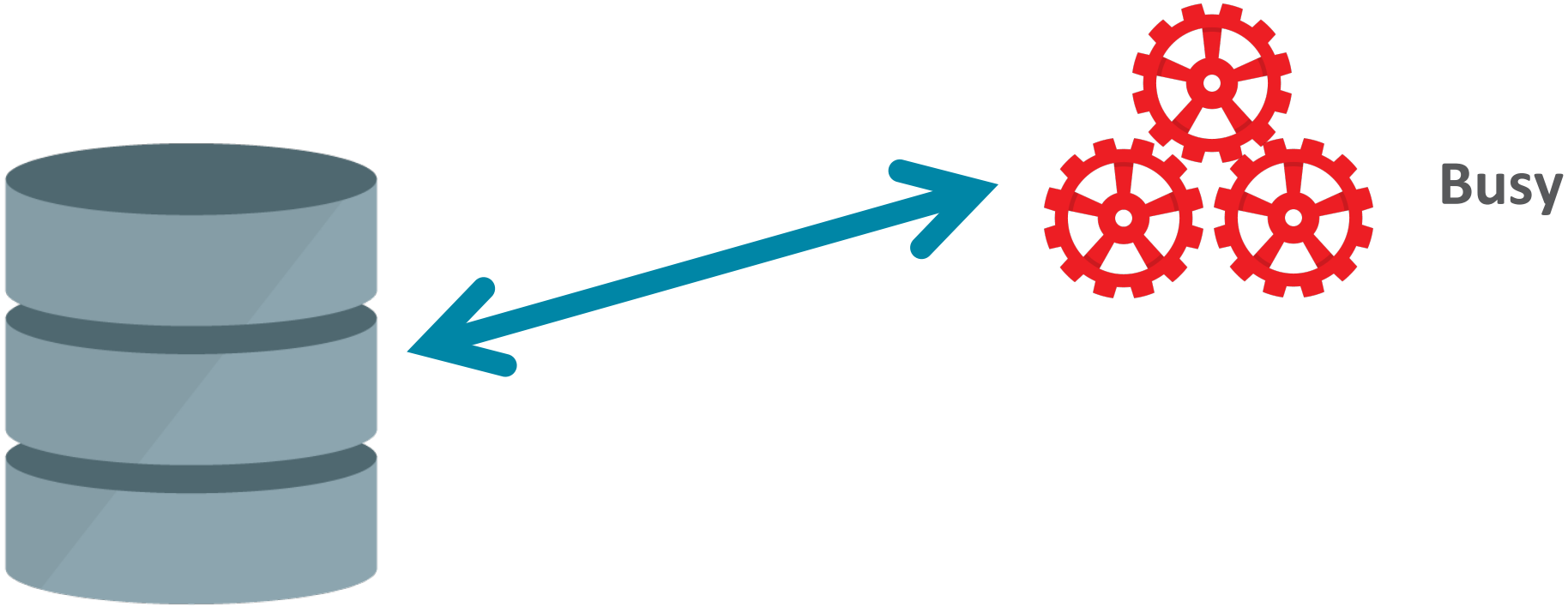


# A database is either working **or idle**

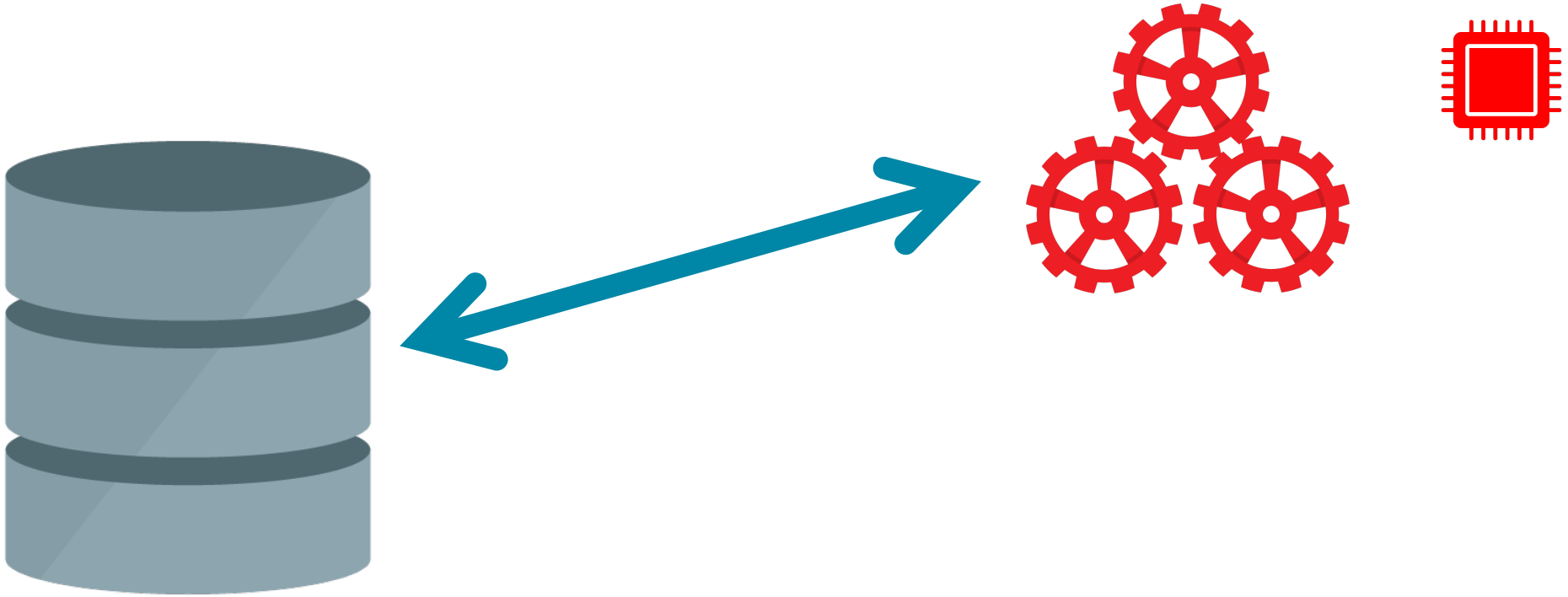


# The working states of the database

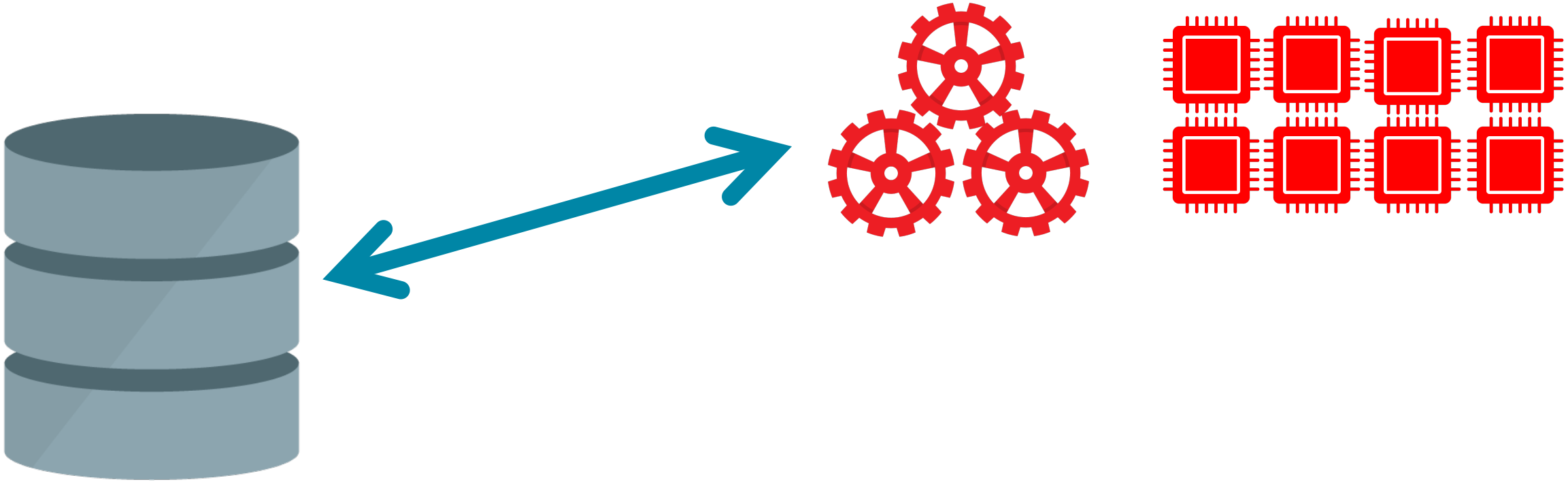
# The working states of the database



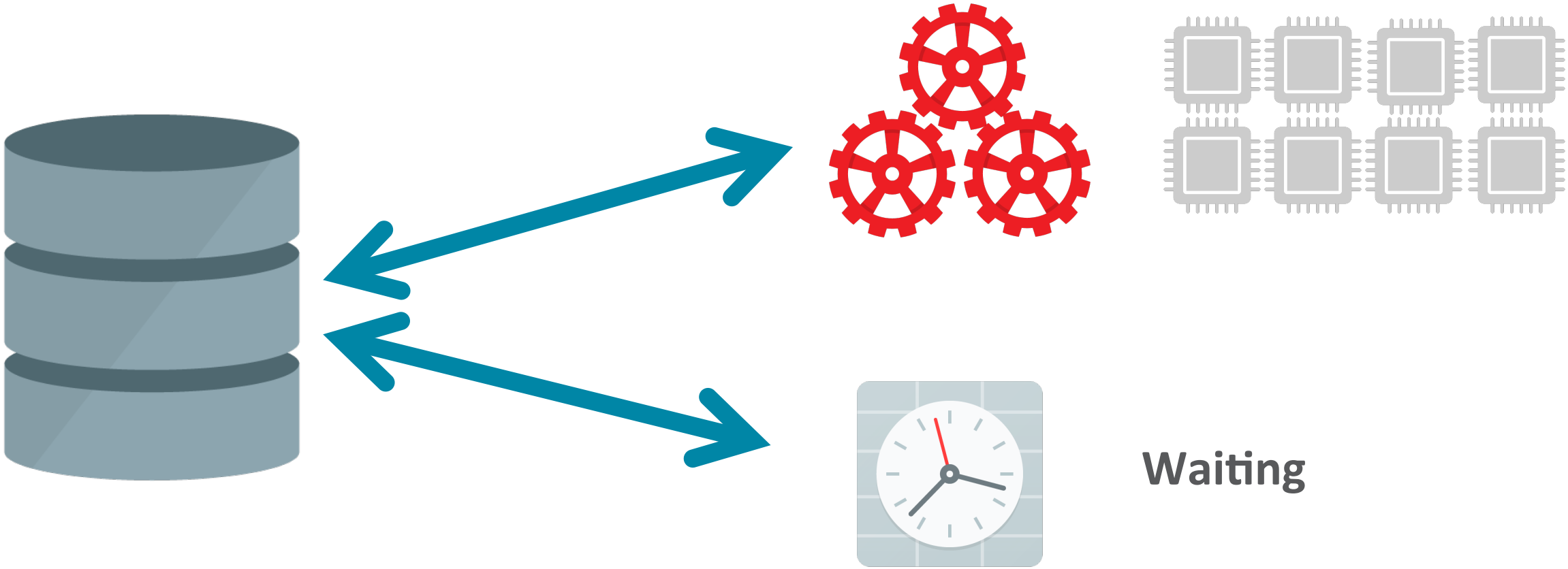
# The working states of the database



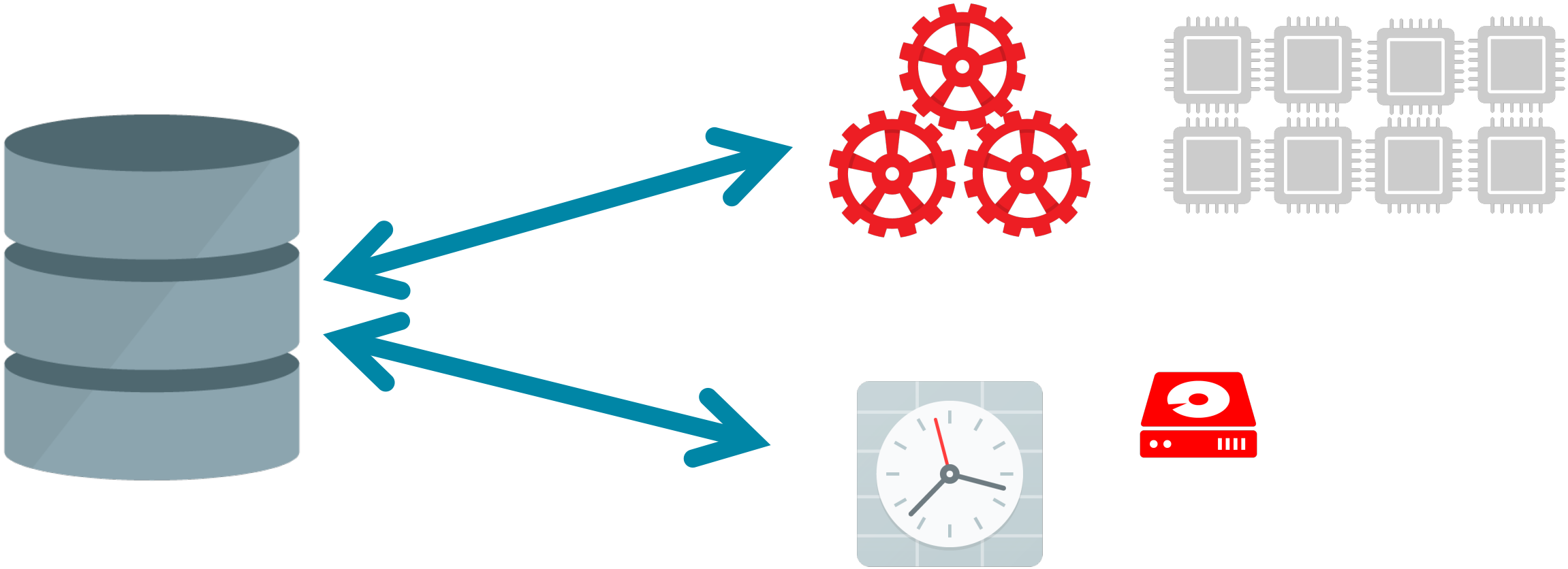
# The working states of the database



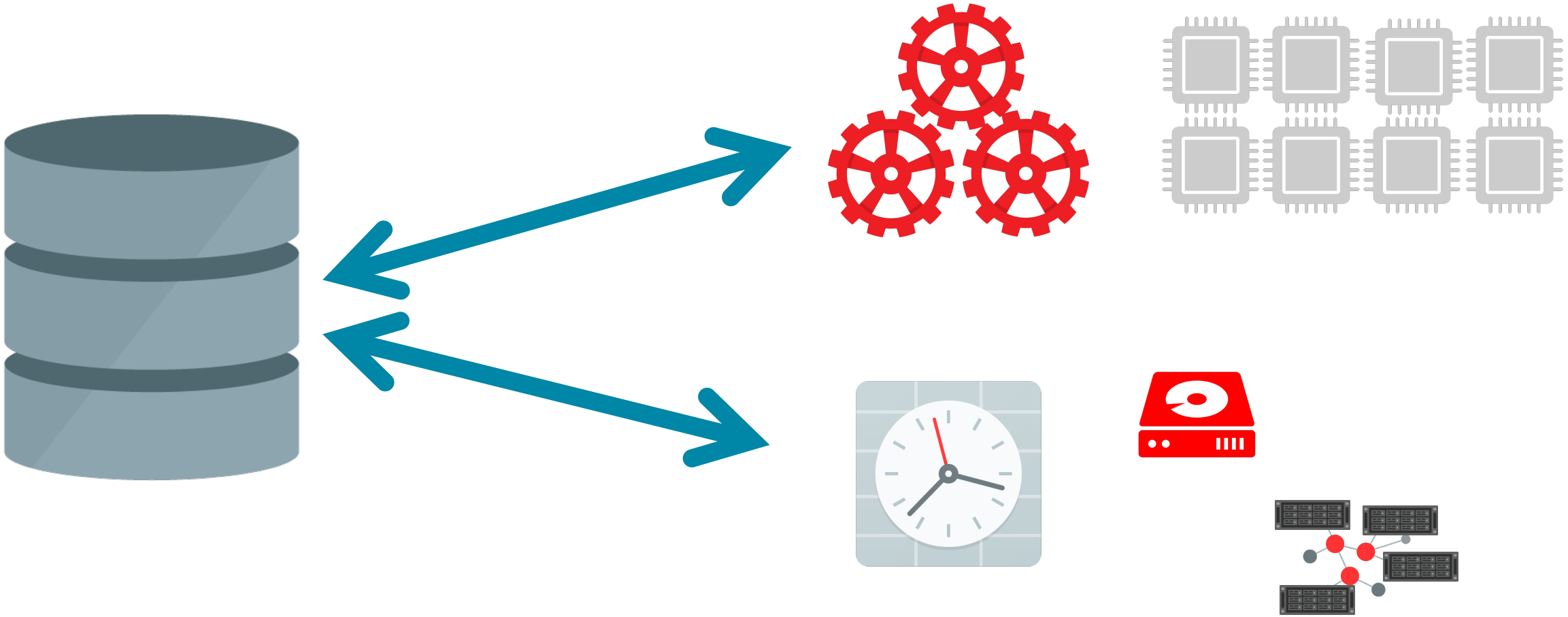
# The working states of the database



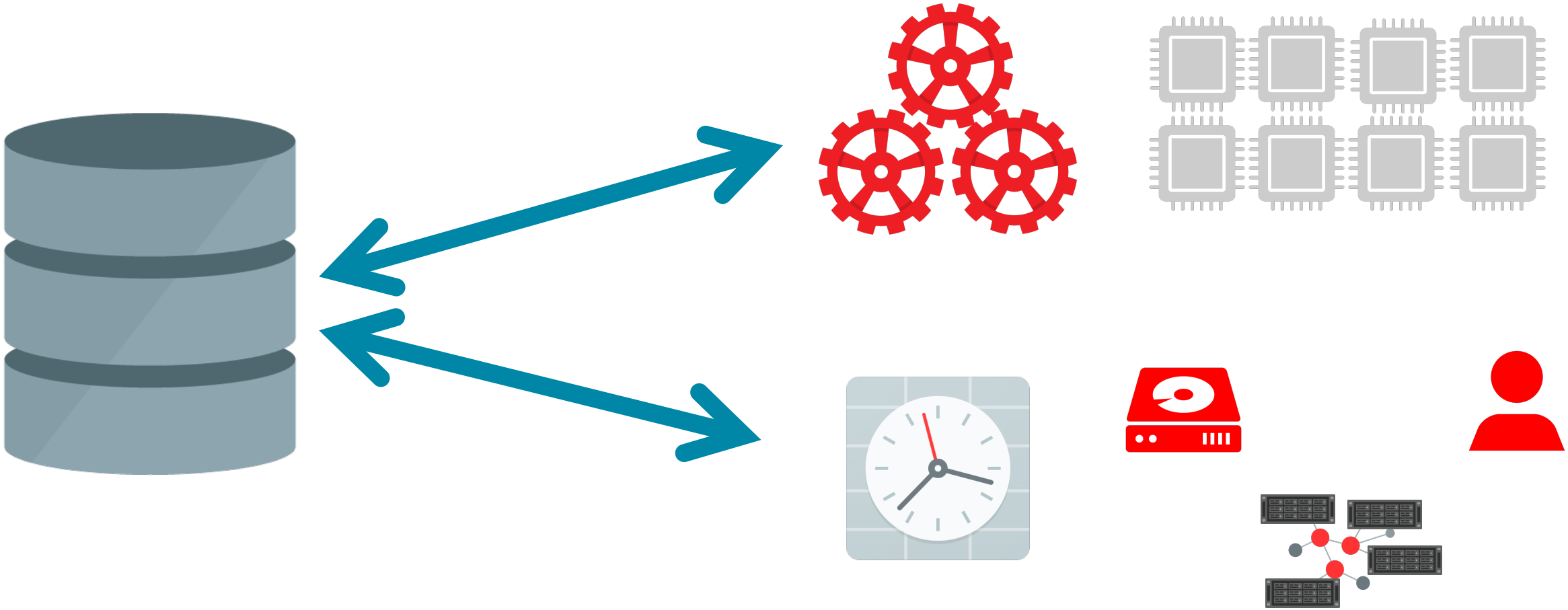
# The working states of the database



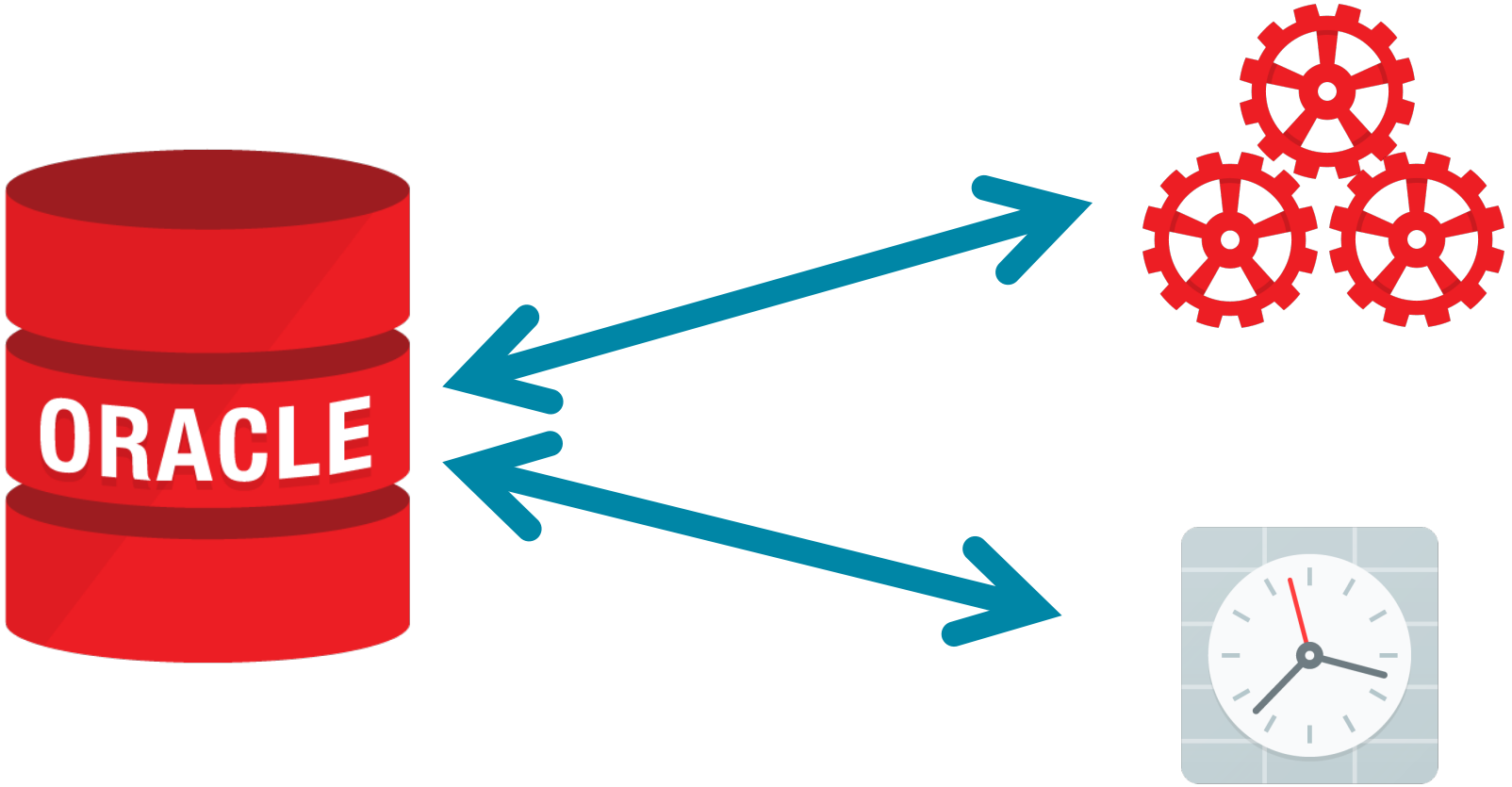
# The working states of the database



# The working states of the database

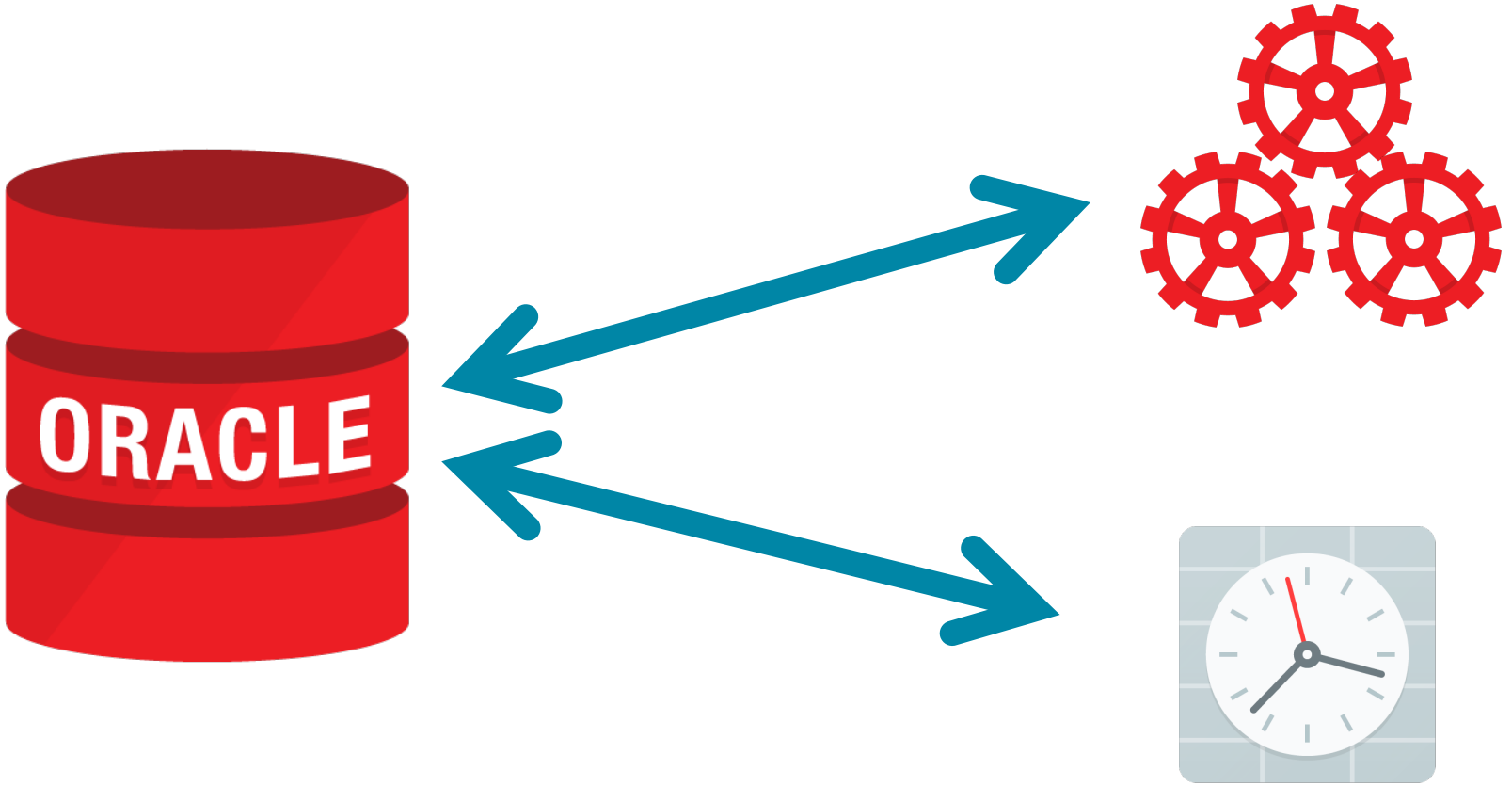


# The working states of the **Oracle** Database




= Statistics

# The working states of the **Oracle** Database




= Statistics

= Wait Classes

# Oracle Database statistics

- **1804** statistics in total in Oracle DB 12.2.0.1
- Query: v$sysstat, v$mystat, v$sesstat (v$statname)
- For example:

```
SELECT n.name, s.value
 FROM v$mystat s, v$statname n
  WHERE s.statistic#=n.statistic#
   AND value > 0
    ORDER BY n.name;
```

# Oracle database wait classes

- **Administrative:** Waits resulting from DBA commands that cause users to wait (index rebuild, etc.)
- **Application:** Waits resulting from user application code (lock waits caused by row level locking or explicit lock commands)
- **Cluster:** Waits related to Oracle Real Application Clusters resources (global cache resources such as 'gc cr block busy')
- **Commit:** This wait class only comprises one wait event - wait for redo log write confirmation after a commit (that is, 'log file sync')
- **Concurrency:** Waits for internal database resources (latches, cursor pins, etc.)
- **Configuration:** Waits caused by inadequate configuration of database or instance resources (undersized log files, shared pool, etc.)
- **Idle:** Waits that signify the session is inactive, waiting for work ('SQL*Net message from client')
- **Network:** Waits related to network messaging ('SQL*Net more data to dblink')
- **Other:** Waits which should not typically occur on a system (for example, 'wait for EMON to spawn')
- **Queueing:** Delays in obtaining additional data in a pipelined environment. (parallel queries, or DBMS_PIPE PL/SQL packages)
- **Scheduler:** Resource Manager related waits ('resmgr: become active', etc.)
- **System I/O:** Waits for background process I/O (for example, DBWR wait for 'db file parallel write')
- **User I/O:** Waits for user I/O (for example 'db file sequential read')

# Oracle database wait classes

- For example:
- ```
  SELECT sid, event, p1, p2, p3
    FROM v$session_wait
     ORDER BY sid, event;
  ```
  ```

# Program Agenda

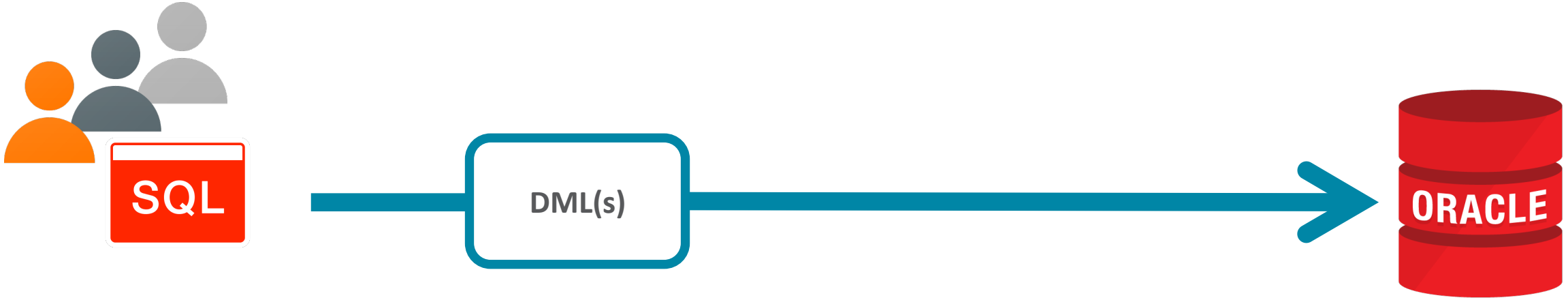
1. Introduction
2. What is performance
3. Oracle Database Performance Statistics
4. Commits – saver of your data
5. Row by row = slow by slow
6. Bind vari… – WHAT?

# Commits – saver of your data

## How does Oracle make sure your data is stored on disk?

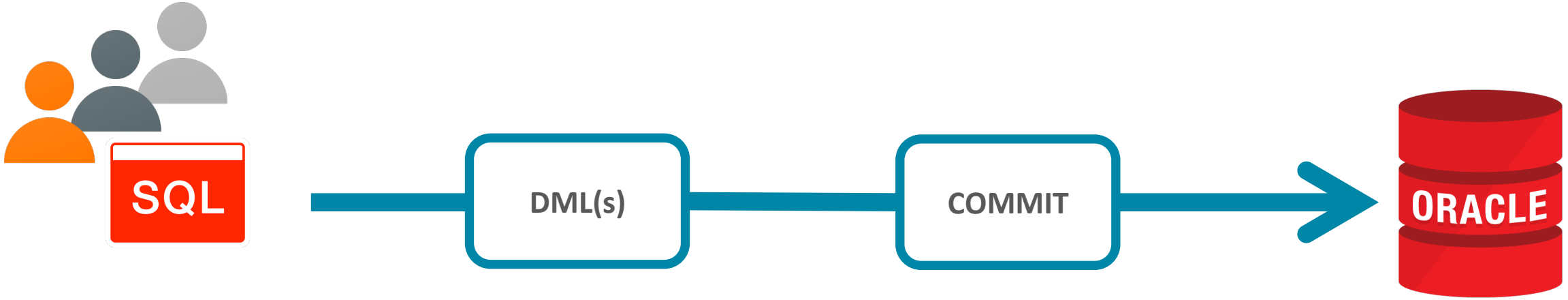
# Commits – saver of your data

## How does Oracle make sure your data is stored on disk?



# Commits – saver of your data

## How does Oracle make sure your data is stored on disk?



# Commits – saver of your data

## How does Oracle make sure your data is stored on disk?

INSERT INTO purchase (…)

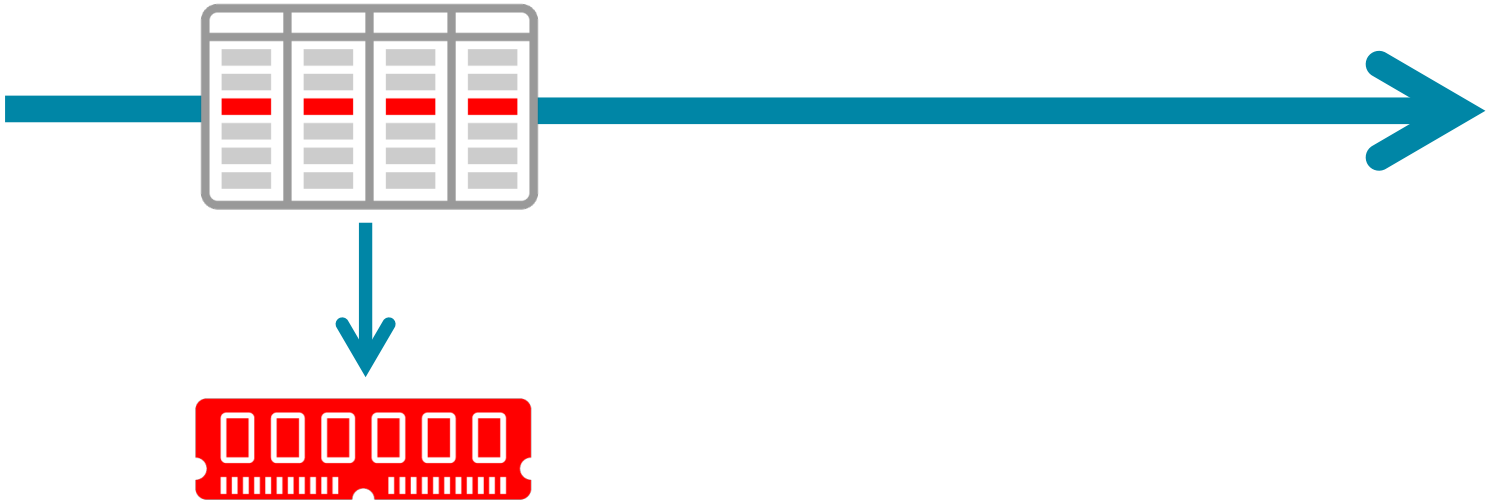
VALUES (:1, :2, :3, …);

# Commits – saver of your data

## How does Oracle make sure your data is stored on disk?

INSERT INTO purchase (…)

VALUES (:1, :2, :3, …);

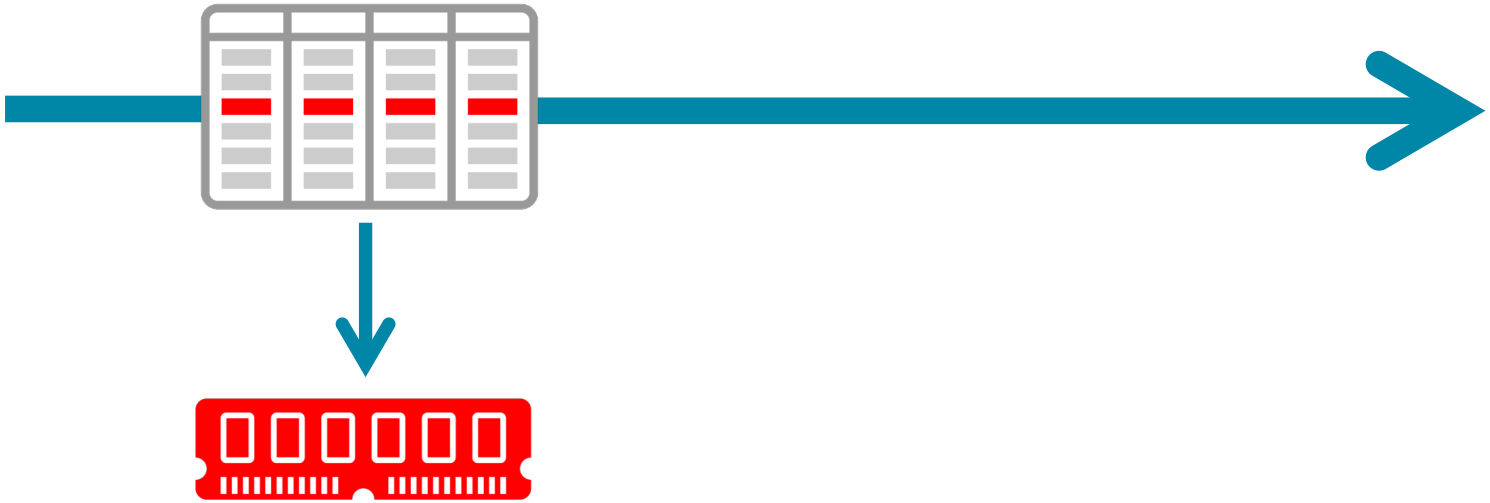


# Commits – saver of your data

## How does Oracle make sure your data is stored on disk?

INSERT INTO purchase (…)

VALUES (:1, :2, :3, …);



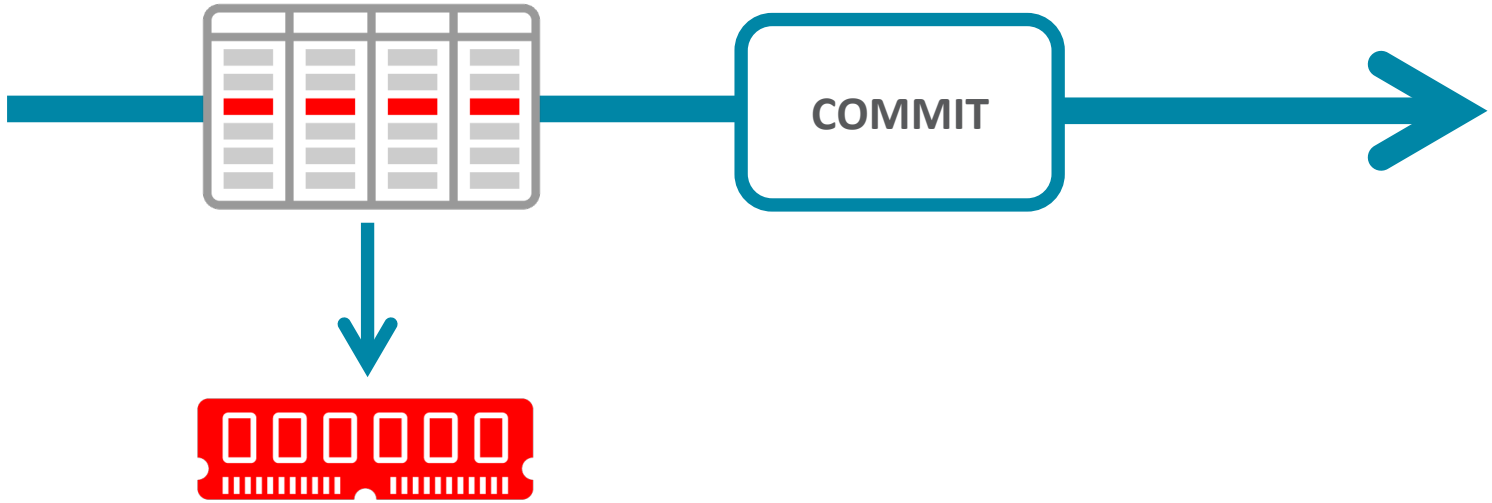
**Buffer Cache**

# Commits – saver of your data

## How does Oracle make sure your data is stored on disk?

INSERT INTO purchase (…)

VALUES (:1, :2, :3, …);

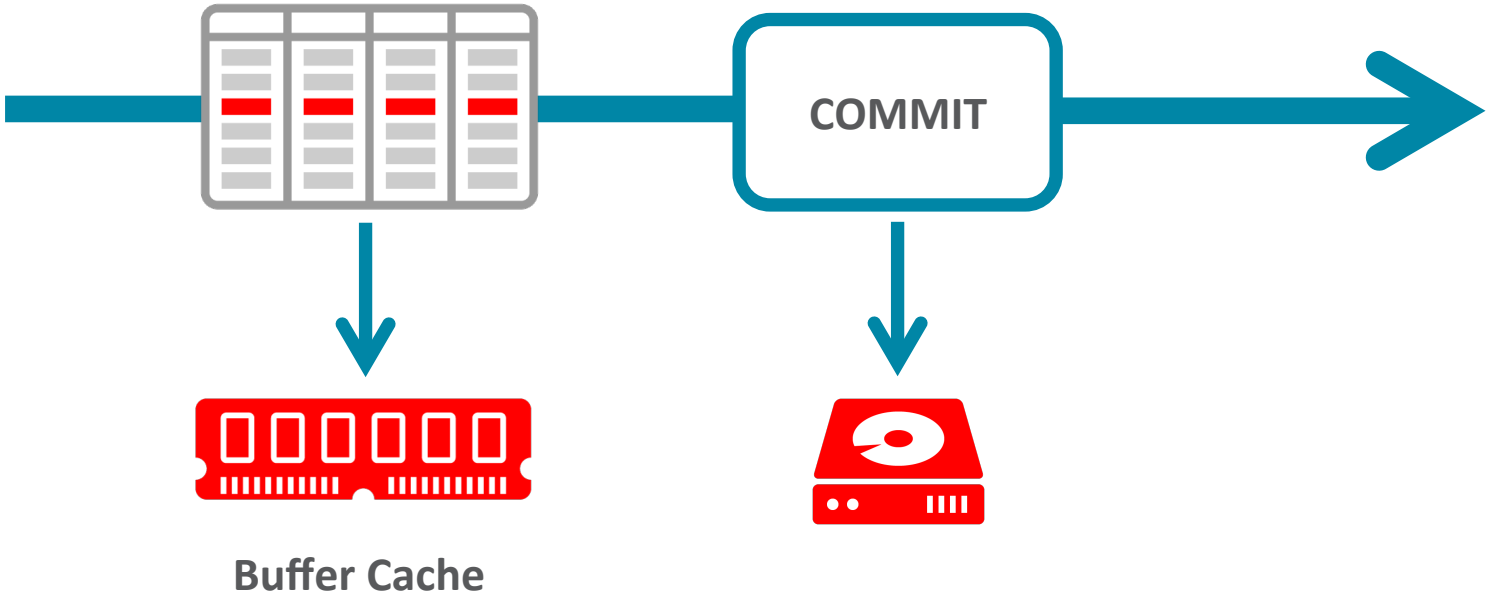


# Commits – saver of your data

## How does Oracle make sure your data is stored on disk?

INSERT INTO purchase (…)

VALUES (:1, :2, :3, …);

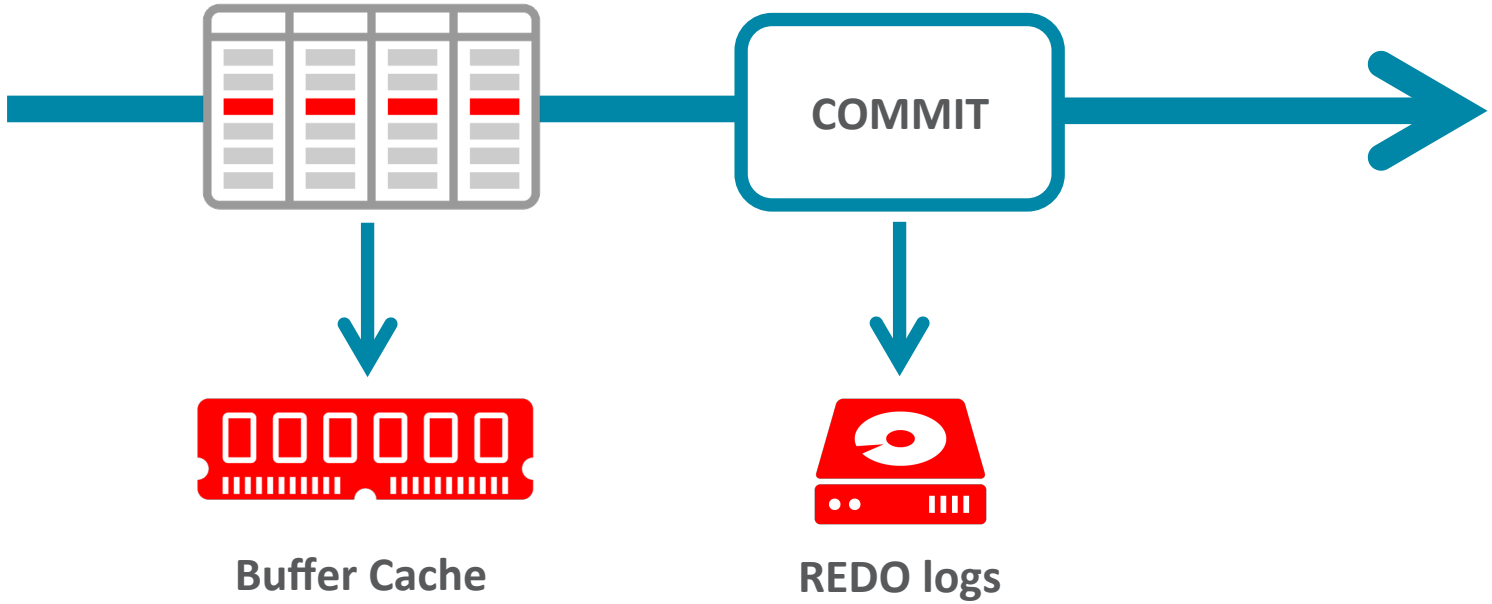


# Commits – saver of your data

## How does Oracle make sure your data is stored on disk?

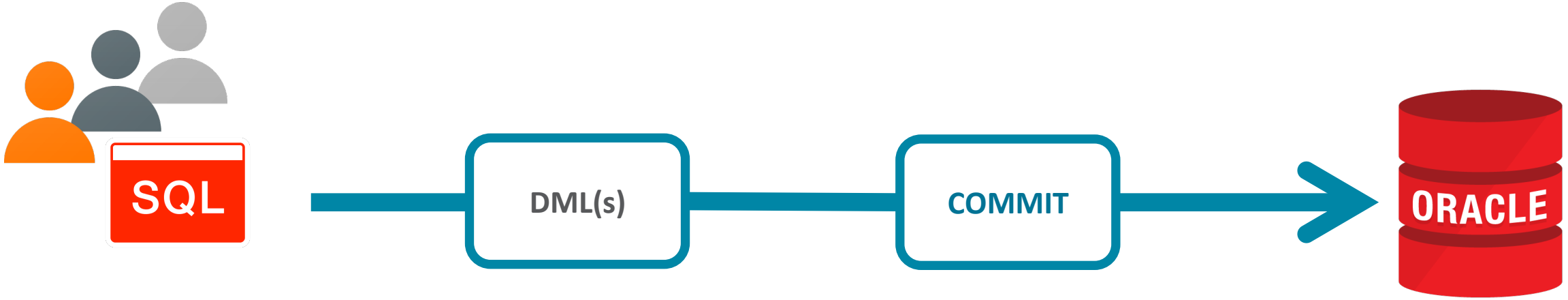
INSERT INTO purchase (…)

VALUES (:1, :2, :3, …);



# Commits – saver of your data

## What happens on a COMMIT?



# Commits – saver of your data

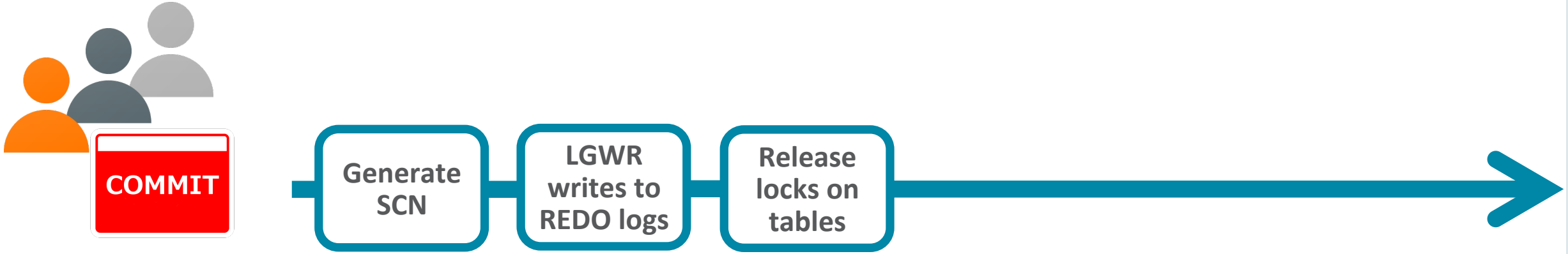
## What happens on a COMMIT?

# Commits – saver of your data

## What happens on a COMMIT?

# Commits – saver of your data

## What happens on a COMMIT?

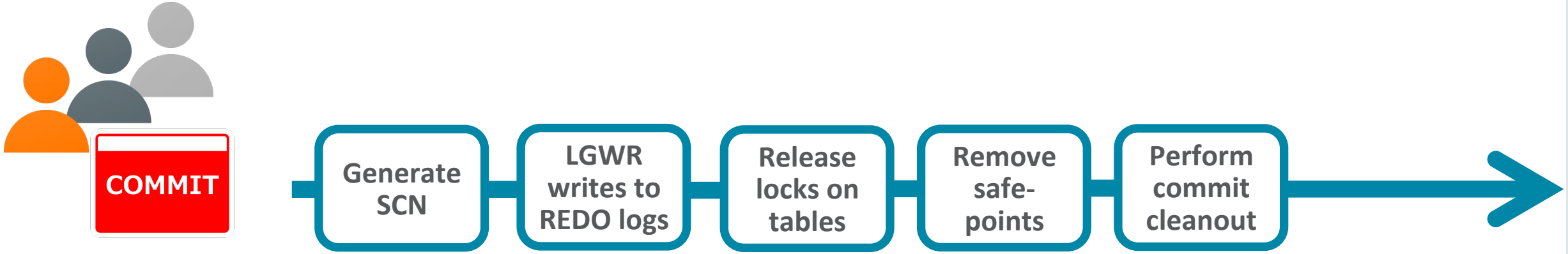


# Commits – saver of your data

## What happens on a COMMIT?


# Commits – saver of your data

## What happens on a COMMIT?

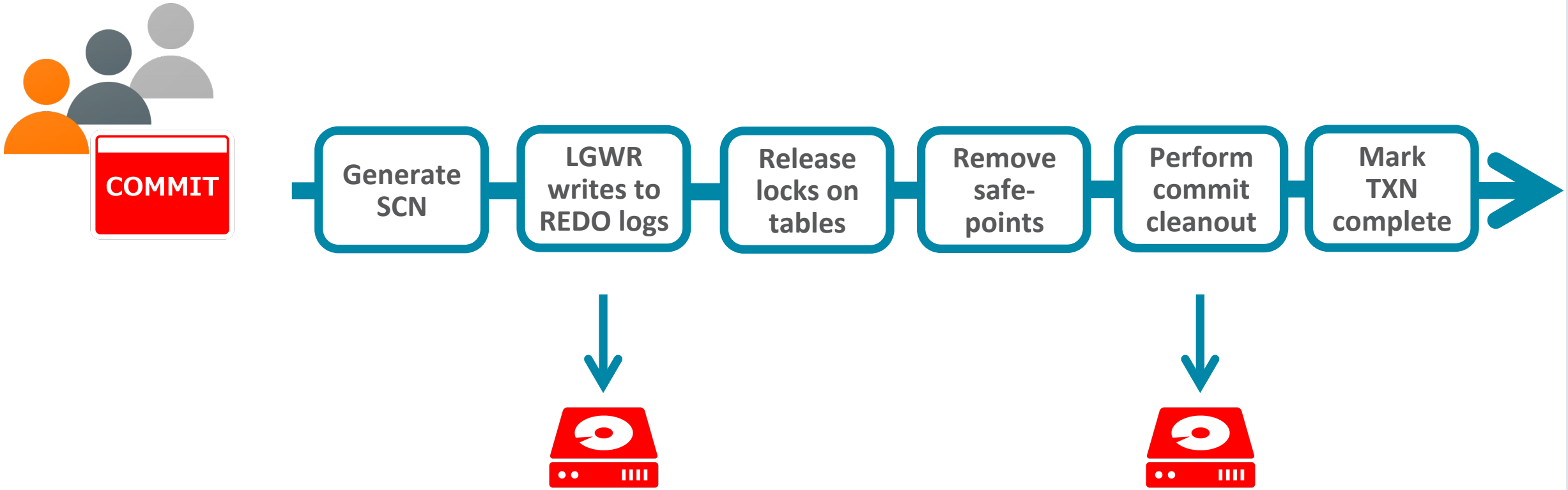


# Commits – saver of your data

## What happens on a COMMIT?


# Commits – saver of your data

## What happens on a COMMIT?



# Commits – saver of your data

```
PreparedStatement stmt = conn.prepareStatement(
                    "INSERT INTO TEST (id, text, created_tms, last_upd_tms) VALUES
                          (?,?,SYSDATE,SYSDATE)");

long start = System.currentTimeMillis();

for(int i=1;i<=rows;i++) {
       stmt.setInt(1, i);
       stmt.setString(2, "This is the row with the value of " + i);
       stmt.executeUpdate();
       conn.commit();
}

long end = System.currentTimeMillis();
System.out.println("Elapsed time(ms) for commit after every row: " + (end-start));
```

# Commits – saver of your data

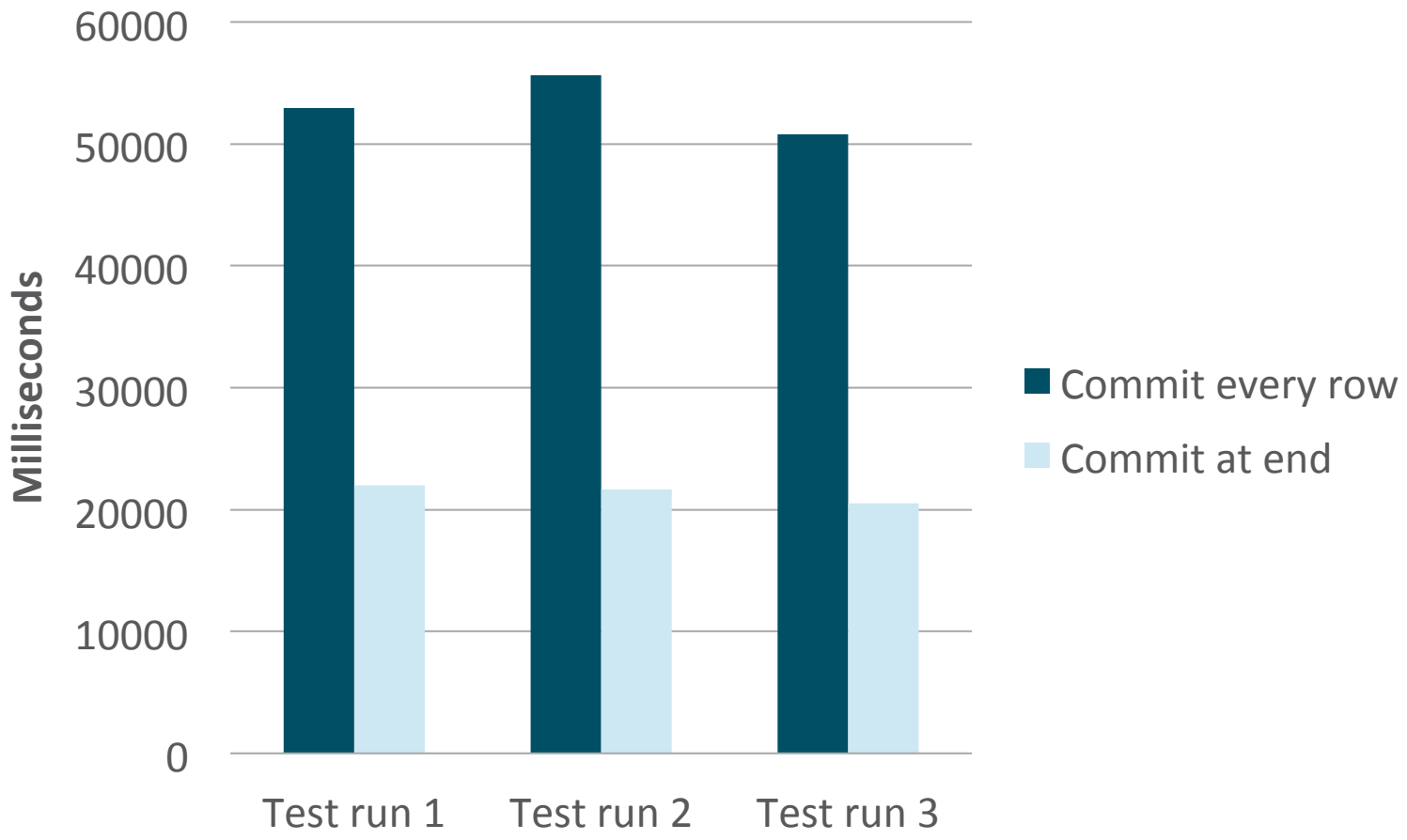
```
PreparedStatement stmt = conn.prepareStatement(
                    "INSERT INTO TEST (id, text, created_tms, last_upd_tms) VALUES
                          (?,?,SYSDATE,SYSDATE)");

long start = System.currentTimeMillis();

for(int i=1;i<=rows;i++) {
       stmt.setInt(1, i);
       stmt.setString(2, "This is the row with the value of " + i);
       stmt.executeUpdate();
}
conn.commit();

long end = System.currentTimeMillis();
System.out.println("Elapsed time(ms) for commit at the end: " + (end-start));
```

# Commits – saver of your data




- Three test runs inserting 10k rows
- Dark blue axis shows elapsed time when committing after every row
- Light blue axis shows elapsed time when committing only once after all data is loaded

# Commits – saver of your data

## When to commit?

- Commit when your business logic requires you to
  - Non restart-able work units usually require commit
    - Payment processing, order confirmation, ATM withdrawal, etc.
  - Restart-able work units normally don't require a commit
    - Batch loads, data transformation, etc.
- Commit means a round trip to the database
- **Remember**: Autocommit is set to true by default in a lot of drivers

# Program Agenda

1. Introduction
2. What is performance
3. Oracle Database Performance Statistics
4. Commits – saver of your data
5. Row by row = slow by slow
6. Bind vari… – WHAT?

# Row by row – Quiz

- What would you rather do on a table with 10 million rows:

# Row by row – Quiz

- What would you rather do on a table with 10 million rows:

```
PreparedStatement stmt = conn.prepareStatement(
       "SELECT value FROM PURCHASE WHERE tms > '2016-09-01'");

ResultSet rslt = stmt.executeQuery();

while (rslt.next()) {
       val += rslt.getInt(1);
}
```

# Row by row – Quiz

- What would you rather do on a table with 10 million rows:

```
PreparedStatement stmt = conn.prepareStatement(
       "SELECT value FROM PURCHASE WHERE tms > '2016-09-01'");

ResultSet rslt = stmt.executeQuery();

while (rslt.next()) {
       val += rslt.getInt(1);
}
```

```
PreparedStatement stmt = conn.prepareStatement(
       "SELECT SUM(value) FROM PURCHASE WHERE tms > '2016-09-01'");

ResultSet rslt = stmt.executeQuery();
rslt.next();
val = rslt.getInt(1);
```

# Row by row – Quiz

- What would you rather do on a table with 10 million rows:

```
PreparedStatement stmt = conn.prepareStatement(
       "SELECT value FROM PURCHASE WHERE tms > '2016-09-01'");

ResultSet rslt = stmt.executeQuery();

while (rslt.next()) {
       val += rslt.getInt(1);
}
```

```
PreparedStatement stmt = conn.prepareStatement(
       "SELECT SUM(value) FROM PURCHASE WHERE tms > '2016-09-01'");

ResultSet rslt = stmt.executeQuery();
rslt.next();
val = rslt.getInt(1);
```

# Row by row = Slow by slow

- Databases work best with set based processing
- Process as much as possible on the side where it makes most sense
- Generally a single SQL statement execution is fastest
- Avoid unnecessary round trips

# Row by row = Slow by slow

```
PreparedStatement stmt = conn.prepareStatement(
                    "INSERT INTO TEST (id, text, created_tms, last_upd_tms) VALUES
                          (?,?,SYSDATE,SYSDATE)");

long start = System.currentTimeMillis();

for(int i=1;i<=rows;i++) {
       stmt.setInt(1, i);
       stmt.setString(2, "This is the row with the value of " + i);
       stmt.executeUpdate();
}
conn.commit();

long end = System.currentTimeMillis();
System.out.println("Elapsed time(ms) for row by row insert: " + (end-start));
```

# Row by row = Slow by slow

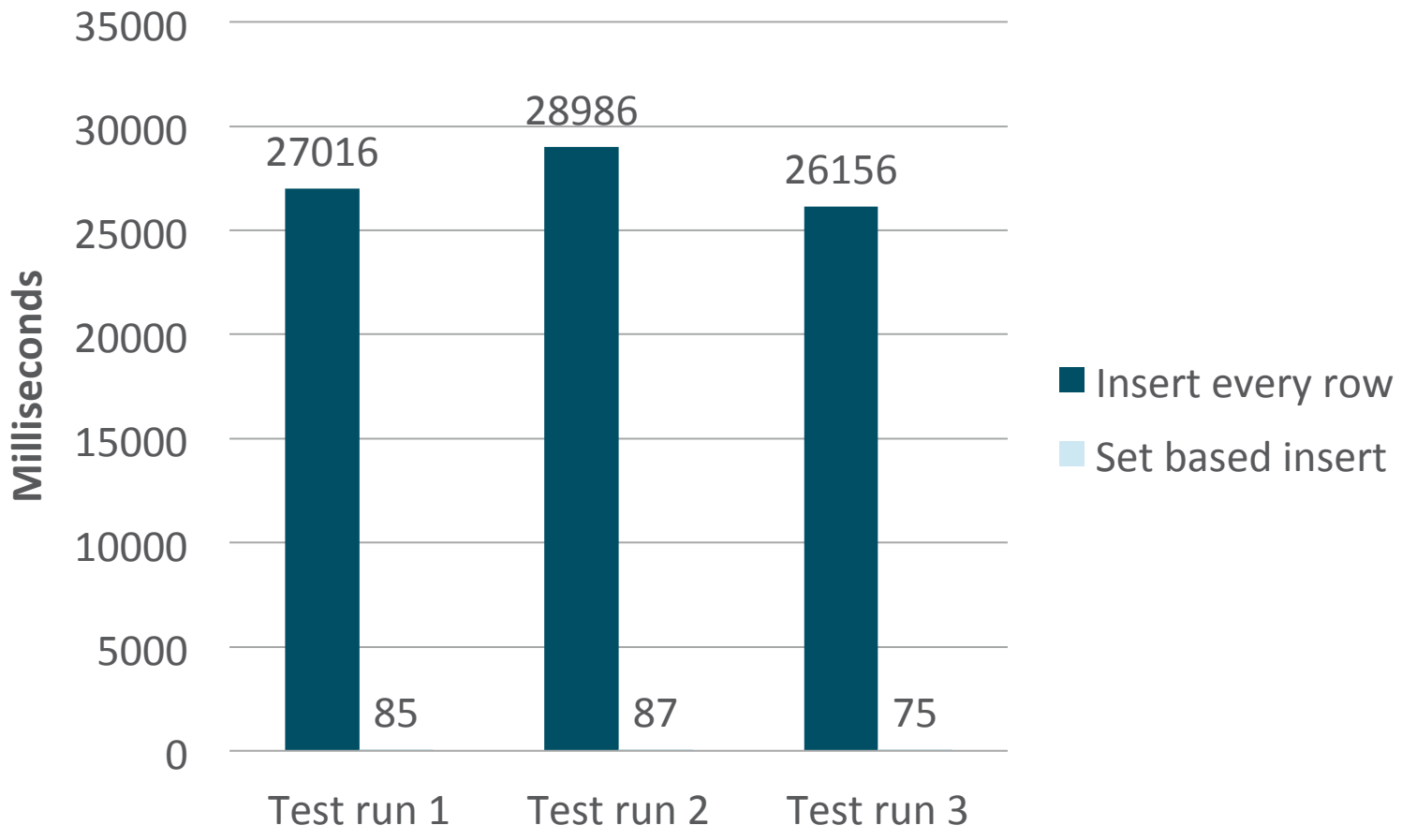
```
PreparedStatement stmt = conn.prepareStatement(
                    "INSERT INTO TEST (id, text, created_tms, last_upd_tms) VALUES
                          (?,?,SYSDATE,SYSDATE)");

long start = System.currentTimeMillis();

for(int i=1;i<=rows;i++) {
       stmt.setInt(1, i);
       stmt.setString(2, "This is the row with the value of " + i);
       stmt.addBatch();
}
stmt.executeBatch();
conn.commit();

long end = System.currentTimeMillis();
System.out.println("Elapsed time(ms) for set based insert: " + (end-start));
```

# Row by row = Slow by slow




- Three test runs inserting 10k rows
- Dark blue axis shows elapsed time of individual inserts
- Light blue axis shows elapsed time of set based inserts

# Row by row = Slow by slow

## What about errors?

- Use DML error logging
  - Errors don’t abort batch
  - Errors are logged into separate error table including the input data

- SQL syntax: INSERT INTO TEST … **LOG ERRORS**;

# Program Agenda

1. Introduction
2. What is performance
3. Oracle Database Performance Statistics
4. Commits – saver of your data
5. Row by row = slow by slow
6. Bind vari… – WHAT?

# Bind vari… – WHAT?

## What are bind variables

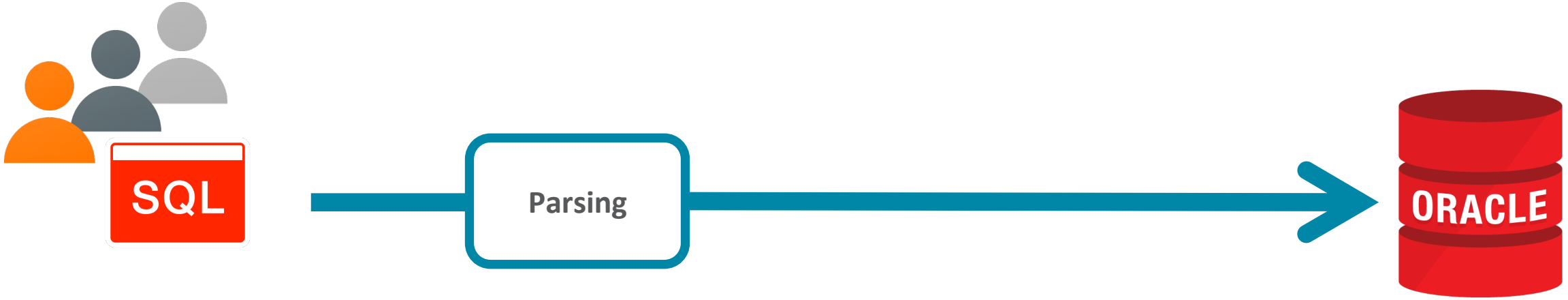
- SQL statements are strings
- But data manipulated via SQL may be of a different types (Number, Date, …)
- A bind variable allows you to bind data to an explicit date type
  - Without having to convert the data type from string
  - And without having to parse the SQL statement again

# SQL processing

## How does Oracle actually process your SQLs?

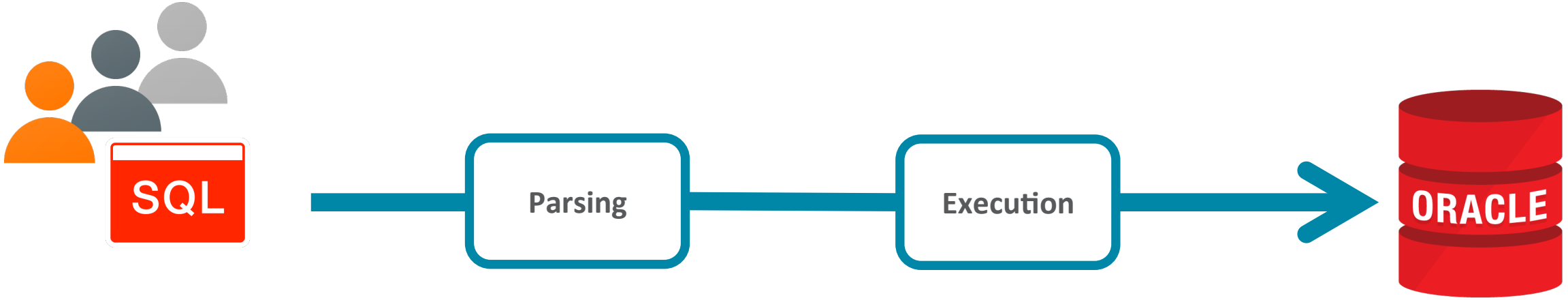
# SQL processing

## How does Oracle actually process your SQLs?



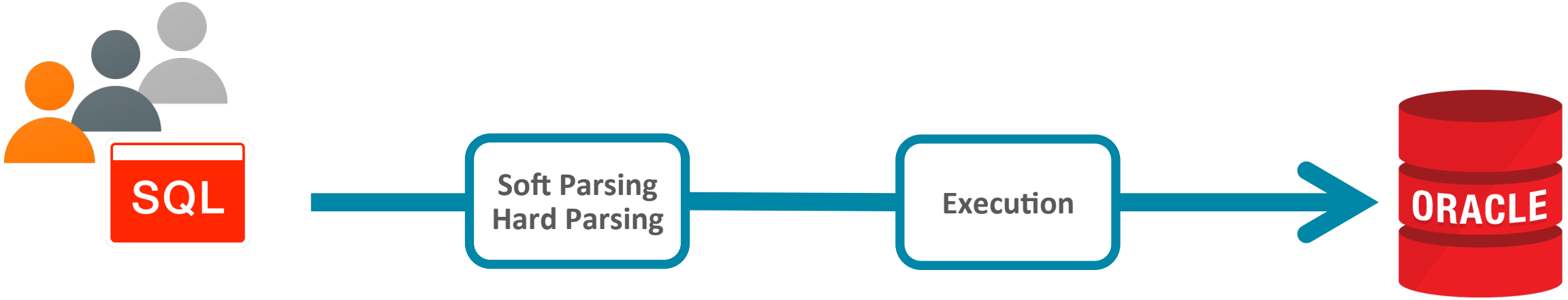
# SQL processing

## How does Oracle actually process your SQLs?



# SQL processing

## How does Oracle actually process your SQLs?



# SQL processing – Parsing

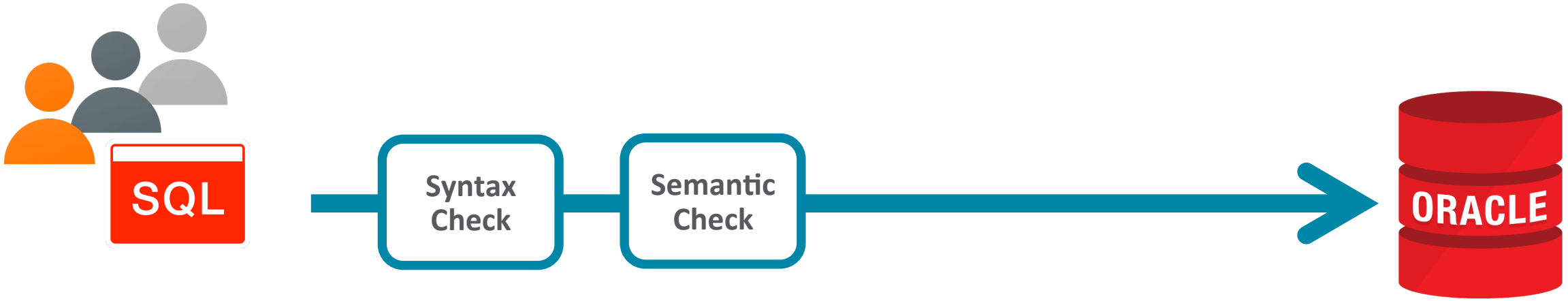
## How does Oracle actually process your SQLs?


Is the SQL syntax correct?

I.e. did you type FROM and co correctly?

# SQL processing – Parsing

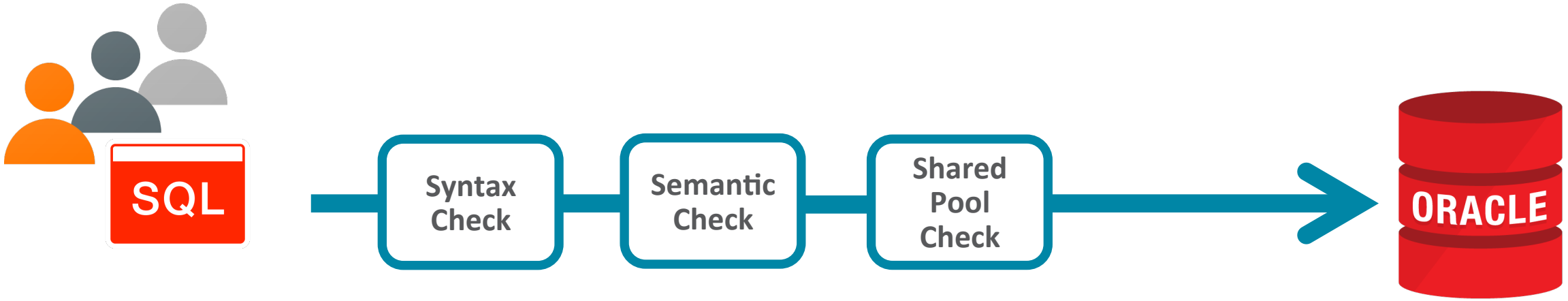
## How does Oracle actually process your SQLs?




Does the tables exist;
do you have permissions,…

# SQL processing – Parsing

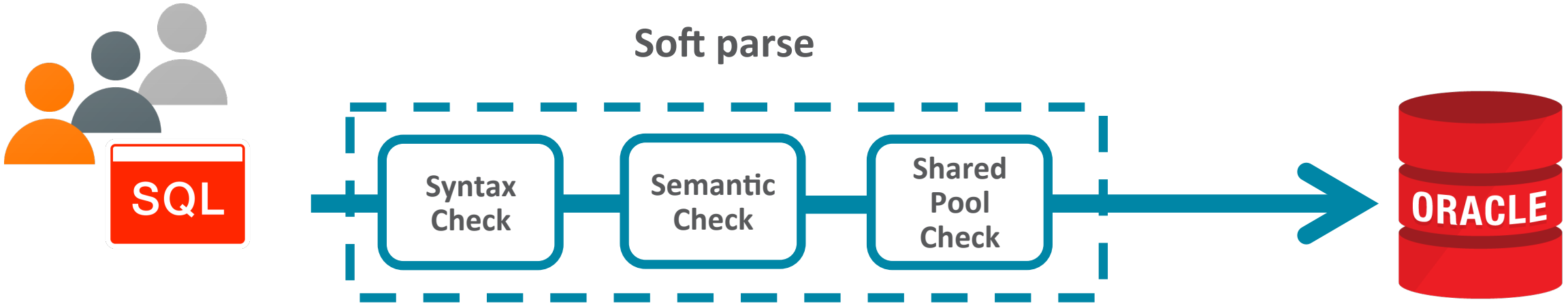
## How does Oracle actually process your SQLs?




Create SQL ID and check whether it has **already been executed before**

# SQL processing – Parsing

## How does Oracle actually process your SQLs?



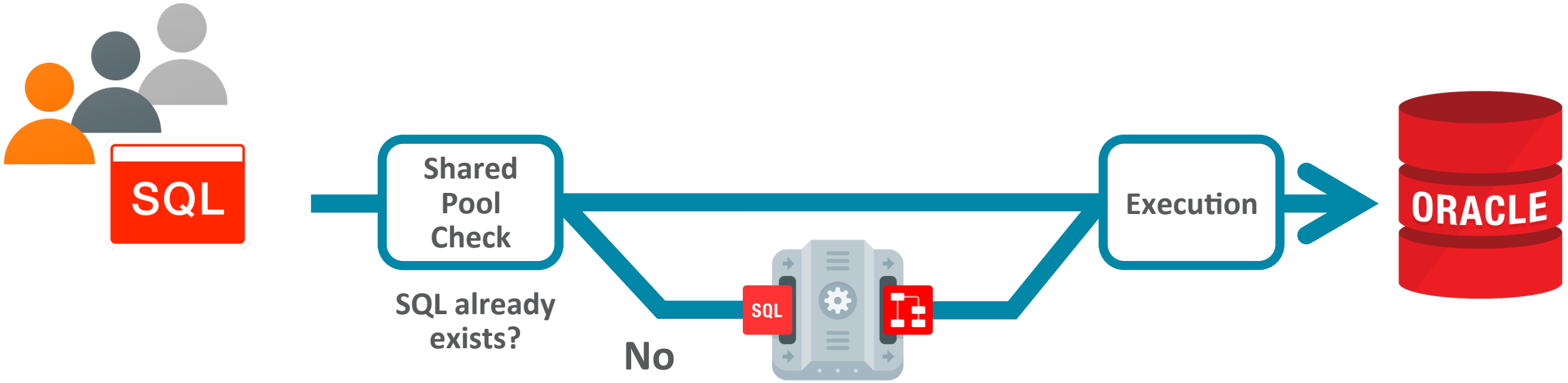

**Always happens for any given query**
**Very fast!**
**Done by your session process**
**(happens concurrently for each session)**

# SQL processing – Parsing

## How does Oracle actually process your SQLs?

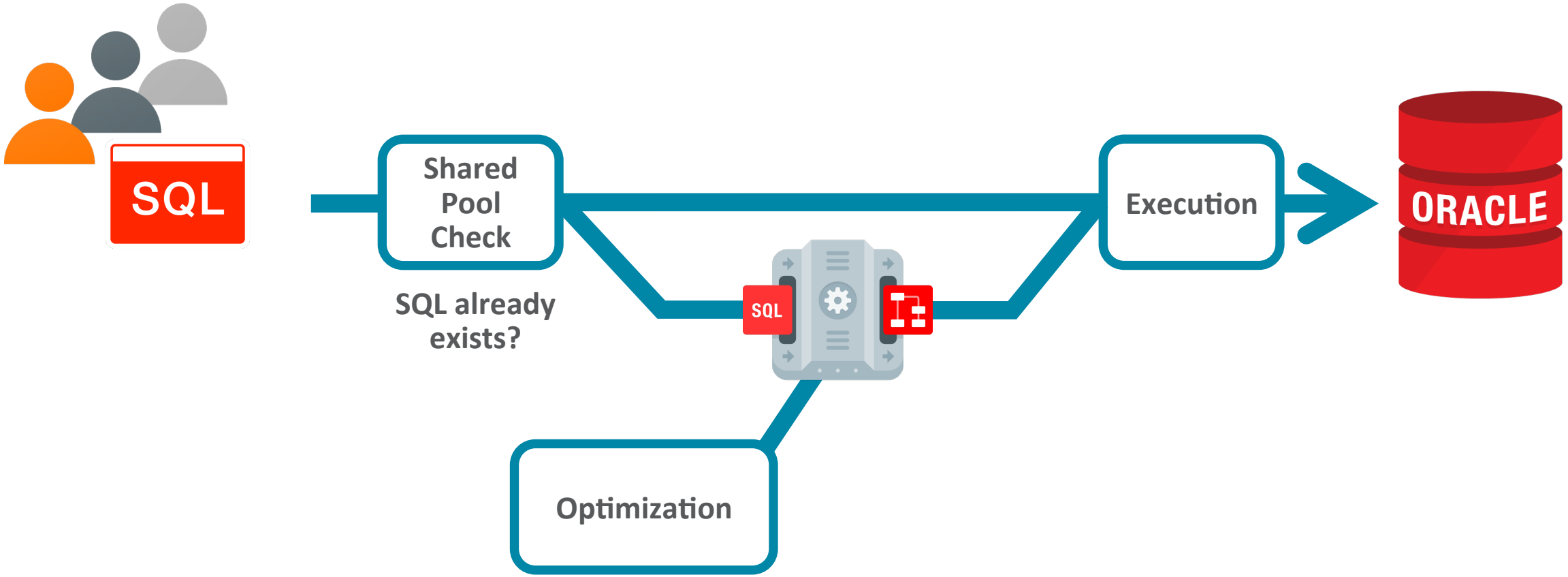
# SQL processing – Parsing

## How does Oracle actually process your SQLs?



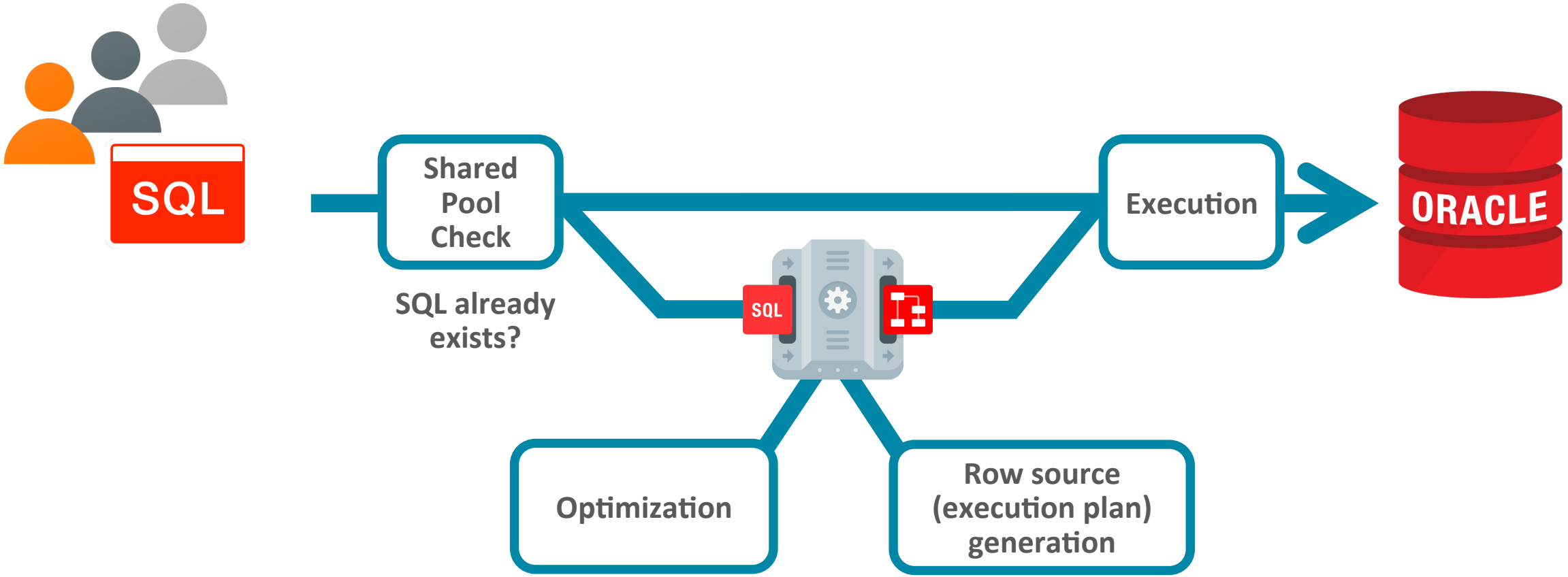
# SQL processing – Parsing

## How does Oracle actually process your SQLs?



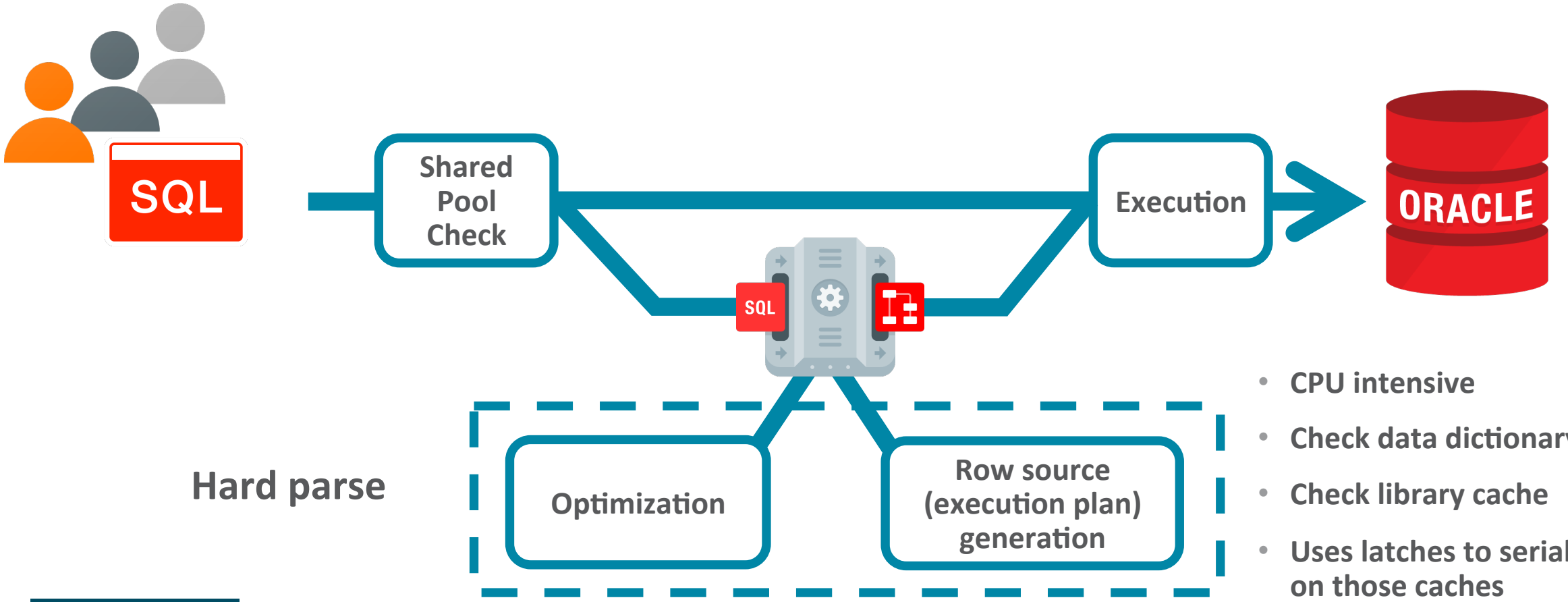
# SQL processing – Parsing

## How does Oracle actually process your SQLs?



# SQL processing – Parsing

## How does Oracle actually process your SQLs?




**Hard parse**

- CPU intensive
- Check data dictionary
- Check library cache
- Uses latches to serialize on those caches

# SQL processing – Bind variables

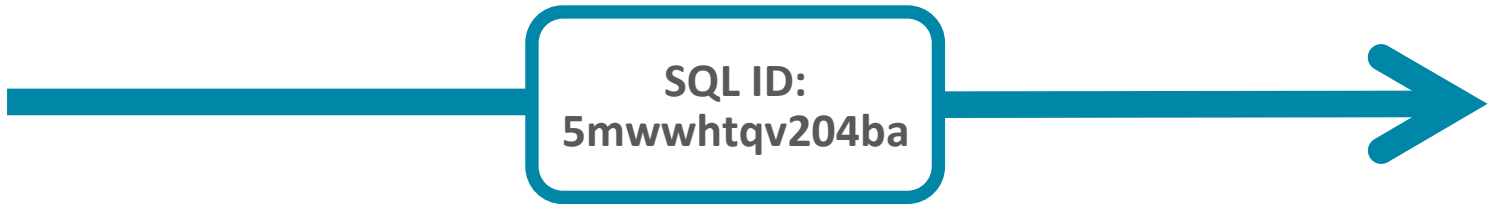
## How does Oracle actually process your SQLs?

SELECT text

FROM TEST

WHERE

**id = 453;**



# SQL processing – Bind variables

## How does Oracle actually process your SQLs?

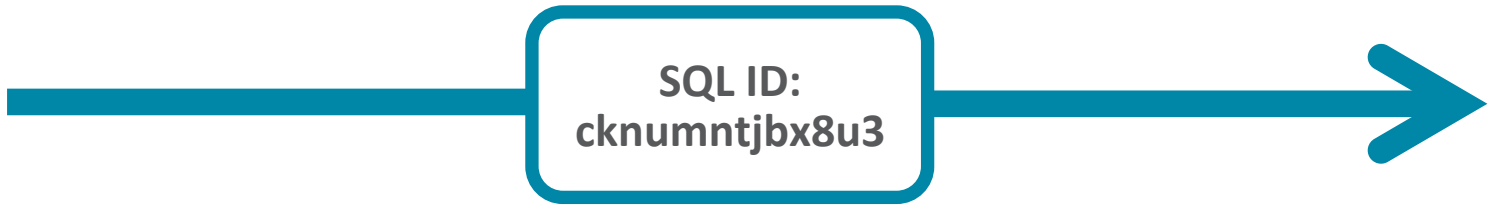
```
SELECT text
FROM TEST
WHERE
id = 453;
```

```
SELECT text
FROM TEST
WHERE
id = 879;
```

# SQL processing – Bind variables

## How does Oracle actually process your SQLs?

```
SELECT text
FROM TEST
WHERE
id = ?;
```

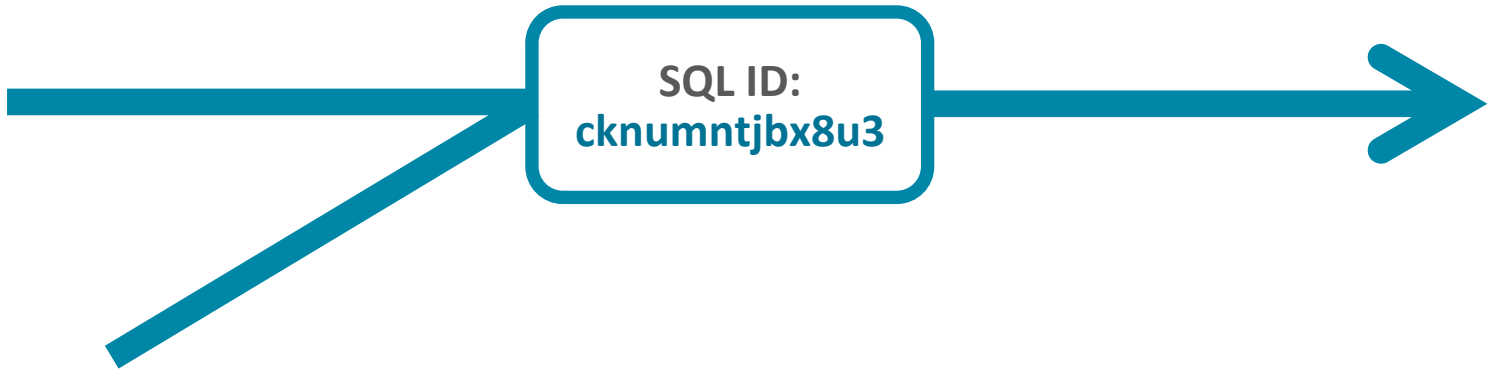


# SQL processing – Bind variables

## How does Oracle actually process your SQLs?

```
SELECT text
FROM TEST
WHERE
id = ?;
```

```
SELECT text
FROM TEST
WHERE
id = ?;
```



# SQL processing – Bind variables

## How does Oracle actually process your SQLs?

```
SELECT sql_id, sql_text
FROM v$sql
WHERE sql_text
LIKE '%TEST%';
```

| SQL_ID | SQL_TEXT |
|---|---|
| cknumntjbx8u3 | SELECT text FROM TEST WHERE id = :1 |
| 5mwwhtqv204ba | SELECT text FROM TEST WHERE id = 453 |
| 06jc0z1kcuu6b | SELECT text FROM TEST WHERE id = 879 |
| 3y3unjhrpp9nm | SELECT sql_id, sql_text FROM v$sql WHERE sql_text LIKE '%TEST%' |

# Bind vari… -WHAT?


- Three test runs selecting 1k rows
- Dark blue axis shows elapsed time of SELECT using literals
- Light blue axis shows elapsed time of SELECT using bind variable

# Useful resources

- Performance Tuning Guide
- SQL Tuning Guide
- Database Development Guide
- JDBC Developer's Guide
- Universal Connection Pool for JDBC Developer's Guide

# Where is the source?

**github.com/gvenzl/Oracle-JavaOneSuperchargeCodeOptimalDBPerf**

# Safe Harbor Statement

The preceding 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, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

JavaOne™

ORACLE®


ORACLE®