# OraBench - Benchmark Framework for Oracle Database Drivers.

build passing release v1.0.0 release date june github repo or version not found

Table of Contents

## 1. Introduction

`OraBench` can be used to determine the performance of different Oracle database drivers under identical conditions. The framework parameters for a benchmark run are stored in a central configuration file.

The currently supported database drivers are:

| Driver | Programming Language(s) |
|---|---|
| cx_Oracle | Python 3 |
| godror | Go |
| JDBC.jl | Julia |
| Oracle JDBC | Java & Kotlin |
| Oracle ODPI-C | C++ (gcc) |
| Oracle.jl | Julia |
| oranif | Elixir & Erlang |

The following Oracle database versions are provided in a benchmark run via Docker container:

| Shortcut | Oracle Database Version |
|---|---|
| db_18_4_xe | Oracle Database 18c 18.4 (Express Edition) - Linux x86-64 |
| db_19_3_ee | Oracle Database 19c 19.3 - Linux x86-64 |
| db_21_3_ee | Oracle Database 21c 21.3 - Linux x86-64 |

The results of the benchmark runs are collected in either csv (comma-separated values) or tsv (tab-separated values) files.

---

## 2. Framework Tools

### 2.1 Benchmark Configuration

The benchmark configuration file controls the execution and output of a benchmark run. The default name for the configuration file is `priv/properties/ora_bench.properties`. A detailed description of the configuration options can be found here. For reasons of convenience the following files are generated:

- the configuration file `priv/ora_bench_c.propperties` for C++ (gcc),
- the configuration file `priv/ora_bench_erlang.properties` with a corresponding map for Erlang, and
- the configuration file `priv/ora_bench_python.propperties` for Python 3.
- the configuration file `priv/ora_bench_toml.propperties` for Julia.

All the file names specified here are also part of the configuration file and can be changed if necessary.

### 2.2 Installation

The easiest way is to download a current release of **`OraBench`** from the GitHub repository. You can find the necessary link here.

**`OraBench`** is tested under Ubuntu.

Git is needed to download the repository and for compilation the following software components are needed:

- Elixir
- Erlang
- gcc Windows or gcc Ubuntu
- Go
- Gradle Build Tool
- Java, e.g. the open-source JDK
- Julia
- Kotlin
- Oracle Instant Client
- Python 3
- rebar3

For changes to the **`OraBench`** repository it is best to use an editor (e.g. Vim) or a suitable IDE. For using the Docker Image based databases in operational mode, Docker Desktop must also be installed. For the respective software versions, please consult the document release notes.

The whole software environment for the operation and further development of OraBench can be created most easily by using a Docker container (version 1.1.0 from here).

Alternatively, in an Ubuntu 20.04 based environment, e.g.: in a virtual machine, the two following scripts can be used to install the necessary software:

- `scripts/1.1.0/run_install_4-vm_wsl2_1.sh`

- `scripts/1.1.0/run_install_4-vm_wsl2_2.sh`
  - run `sudo apt update`
  - run `sudo apt install git`
  - run `git clone https://github.com/KonnexionsGmbH/ora_bench` (cloning the **`OraBench`** repository)
  - run `cd ora_bench/scripts/kxn_dev`
  - run `./run_install_4_vm_wsl2_1.sh`
  - close the Ubuntu shell and reopen it again
  - run `cd ora_bench/scripts/kxn_dev`
  - run `./run_install_4_vm_wsl2_2.sh`

## 2.3 Benchmark Operation

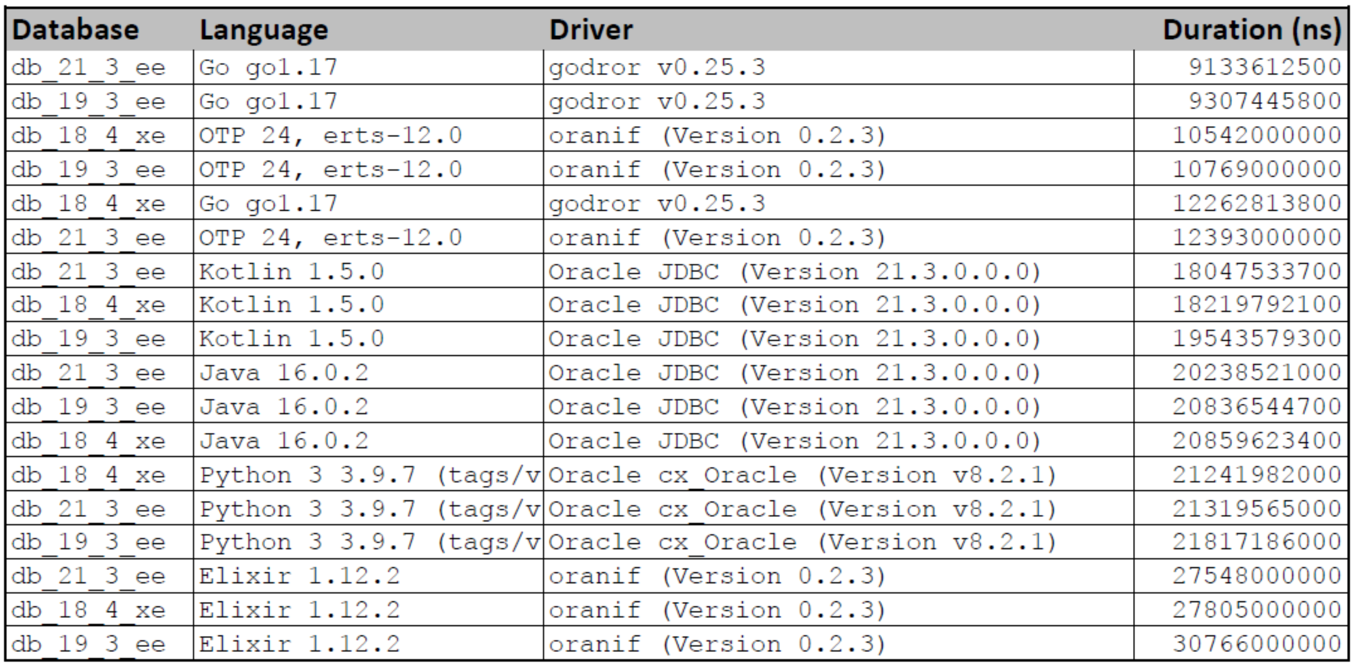
### 2.3.1 Script `run_ora_bench`

This script executes the `run_properties_standard` script for each of the databases listed in chapter Introduction with standard properties. At the beginning of the script it is possible to exclude individual databases or drivers from the current benchmark. The run log is stored in the `run_ora_bench.log` file.

## 2.4 Benchmark Results

In a file defined by the configuration parameters `file.result.delimiter`, `file.result.header` and `file.result.name`, the results of the benchmark run with the actions `benchmark`, `trial` and `query` are stored. In the file directory `priv/statistics` reference statistics files are available per version of **`OraBench`**.

Excerpts from a sample file can be seen in the following image:



| Database | Language | Driver | Duration (ns) |
|---|---|---|---|
| db_21_3_ee | Go go1.17 | godror v0.25.3 | 9133612500 |
| db_19_3_ee | Go go1.17 | godror v0.25.3 | 9307445800 |
| db_18_4_xe | OTP 24, erts-12.0 | oranif (Version 0.2.3) | 10542000000 |
| db_19_3_ee | OTP 24, erts-12.0 | oranif (Version 0.2.3) | 10769000000 |
| db_18_4_xe | Go go1.17 | godror v0.25.3 | 12262813800 |
| db_21_3_ee | OTP 24, erts-12.0 | oranif (Version 0.2.3) | 12393000000 |
| db_21_3_ee | Kotlin 1.5.0 | Oracle JDBC (Version 21.3.0.0.0) | 18047533700 |
| db_18_4_xe | Kotlin 1.5.0 | Oracle JDBC (Version 21.3.0.0.0) | 18219792100 |
| db_19_3_ee | Kotlin 1.5.0 | Oracle JDBC (Version 21.3.0.0.0) | 19543579300 |
| db_21_3_ee | Java 16.0.2 | Oracle JDBC (Version 21.3.0.0.0) | 20238521000 |
| db_19_3_ee | Java 16.0.2 | Oracle JDBC (Version 21.3.0.0.0) | 20836544700 |
| db_18_4_xe | Java 16.0.2 | Oracle JDBC (Version 21.3.0.0.0) | 20859623400 |
| db_18_4_xe | Python 3 3.9.7 (tags/v | Oracle cx_Oracle (Version v8.2.1) | 21241982000 |
| db_21_3_ee | Python 3 3.9.7 (tags/v | Oracle cx_Oracle (Version v8.2.1) | 21319565000 |
| db_19_3_ee | Python 3 3.9.7 (tags/v | Oracle cx_Oracle (Version v8.2.1) | 21817186000 |
| db_21_3_ee | Elixir 1.12.2 | oranif (Version 0.2.3) | 27548000000 |
| db_18_4_xe | Elixir 1.12.2 | oranif (Version 0.2.3) | 27805000000 |
| db_19_3_ee | Elixir 1.12.2 | oranif (Version 0.2.3) | 30766000000 |

In detail, the following information is available in the result files:

| Column | Format | Content |
|---|---|---|
| release | alphanumeric | config param `benchmark.release` |

| Column | Format | Content |
|---|---|---|
| benchmark id | alphanumeric | config param `benchmark.id` |
| benchmark comment | alphanumeric | config param `benchmark.comment` |
| host name | alphanumeric | config param `benchmark.host.name` |
| no. cores | integer | config param `benchmark.number.cores` |
| os | alphanumeric | config param `benchmark.os` |
| user name | alphanumeric | config param `benchmark.user.name` |
| database | alphanumeric | config param `benchmark.database` |
| language | alphanumeric | config param `benchmark.language` |
| driver | alphanumeric | config param `benchmark.driver` |
| trial no. | integer | `0` if action equals `benchmark`, trial no. otherwise |
| SQL statement | alphanumeric | SQL statement if action equals `query`, empty otherwise |
| core multiplier | integer | config param `benchmark.core.multiplier` |
| fetch size | integer | config param `connection.fetch.size` |
| transaction size | integer | config param `benchmark.transaction.size` |
| bulk length | integer | config param `file.bulk.length` |
| bulk size | integer | config param `file.bulk.size` |
| batch size | integer | config param `benchmark.batch.size` |
| action | alphanumeric | one of `benchmark`, `query` or `trial` |
| start day time | yyyy-mm-dd hh24:mi:ss.ffffffff | current date and time at the start of the action |
| end day time | yyyy-mm-dd hh24:mi:ss.ffffffff | current date and time at the end of the action |
| duration (sec) | integer | time difference in seconds between start time and end time of the action |
| duration (ns) | integer | time difference in nanoseconds between start time and end time of the action |

### 2.5 Bulk File

The bulk file in `csv` or `tsv` format is created in the `run_create_bulk_file` script if it does not already exist. The following configuration parameters are taken into account:

- `file.bulk.delimiter`
- `file.bulk.header`

- `file.bulk.length`
- `file.bulk.name`
- `file.bulk.size`

The data column in the bulk file is randomly generated with a unique key column (MD5 hash code).