Benchmarking databases on small to medium size projects

Florian Nemling[a], **Renad Quku**[a]

[a] **Institute for Information Systems Engineering (E194)**
Compilers and Languages (E194-5)

Forschungsmethoden SS20 – Gruppe 2

Intro

The scope of this project is to compare three well-known relational database management systems - MySQL, PostgreSQL and OracleDB – for three different scenarios.
We make use of the “New Passenger Car Registrations by Makes; monthly time series from January 2000 onwards” [1], which includes one main table with more than seventeen thousand records and three lookup-tables.
We benchmark these database management systems on the following three criteria:

1. Response time of a single simple query avoiding caching,
2. Handling of simultaneous request (starting from 50 requests)
3. Response time and efficiency of grouped functions.

Methods and Material

QUERY	DESCRIPTION	SCOPE
1	Query for individual data records	Fast and Furious
2	Many concurrent data requests	MultiUser – Rush Hour
3	Analytical query aggregating several thousand records	Tower to Management

Why use theses 3 databases

Oracle is undoubtedly one of the leading product in the **enterprise world.**

MySQL is one of the most popular database systems driven by a substantial **community** powering **websites and applications**.

PostgreSQL has an academic heritage and still has strong roots in the **academic community**. Moreover, being bundled with numerous *NIX distributions it has become quite **ubiquitous**.

The database systems are being benchmarked in three different scenarios.

Scenario 1: Query for individual data records / very small sets
This scenario is relevant for use cases commonly seen in application development where quick response times are essential. The goal here is to achieve the best response times.

Scenario 2: Concurrent data requests
This scenario is relevant for a database that needs to service concurrent requests. The databases goal should be to service all requests with good response without outliers.

Scenario 3: Analytical query
This scenario focuses on good performance on queries accessing several thousand rows and use of grouped aggregate functions. .

Benchmarking Procedure:
The benchmark numbers for scenarios 1 & 3 are based on samples of 10 queries.
The concurrency performance in scenario 2 was collected with 200 executions and 50 parallel requests.
All databases were running on virtual docker containers as provided by the software producers with two cores and 2 GB RAM.
All Numbers are execution times in seconds, without result transfer or logon/logoff overhead.

SQL & Powershell

```
SELECT TF.NAME, TPM.NAME, TZM.NAME
FROM T_NEUZULASSUNGEN NZ
LEFT JOIN T_FAHRZEUGE TF ON nz.FAHRZEUGEN = tf.CODE
LEFT JOIN T_PKW_MARKEN TPM on tpm.CODE = nz.PKW_MARKEN
LEFT JOIN T_ZEIT_MONATSWERTE TZM on tzm.CODE = nz.ZEIT_MONATSWERTEN
where TPM.NAME LIKE :'quoted';
```

```
SELECT TZM.NAME, sum(NEUZULASSUNGEN), LAG (sum(neuzulassungen)) over
(partition by SUBSTR(TZM.NAME, 0, INSTR(TZM.NAME, ' ')-1) order by
tzm.name)
FROM T_NEUZULASSUNGEN NZ
LEFT JOIN T_FAHRZEUGE TF ON nz.FAHRZEUGEN = tf.CODE
LEFT JOIN T_PKW_MARKEN TPM on tpm.CODE = nz.PKW_MARKEN
LEFT JOIN T_ZEIT_MONATSWERTE TZM on tzm.CODE = nz.ZEIT_MONATSWERTEN
where tzm.name is not null
GROUP BY TZM.NAME;
```

```
1..$no_executions | ForEach-Object -Parallel {

    $PG_results = $using:PG_results

    ########################################
    ###  Define here the needed variables  ###
    ########################################

    $x = get-random -Minimum 0 -Maximum 23


    $ex_time = & "$PG_CLIENT" "--username=$PG_USER" "--dbname=$PG_DB" "-vquoted=""$whereclauses[$x]""" "--file=pg_simple.sql"  |
          Where-Object { $_ -match 'Time:'}[0] |
          foreach-object {$_.Split(' ')[1]}

    $ex_time = [System.Convert]::ToDecimal($ex_time,[cultureinfo]::GetCultureInfo('de_DE'))
    $val = @{"pg_concurrent.sql"= $ex_time/1000}
    $pg_results.Add($val)

} -ThrottleLimit $throttle
```

Outcome and Discussion

seconds		Min	Max	Avg
Fast and Furious	Oracle	0,0000	0,0100	0,0090
	MySQL	0,0104	0,0168	0,0138
	PostgreSQL	0,0027	0,0057	0,0035
Multi-User – Rush Hour	Oracle	0,0600	1,9700	0,6007
	MySQL	0,0101	1,5326	0,2937
	PostgreSQL	0,0034	0,0579	0,0115
Tower to Management	Oracle	0,0400	0,0500	0,0420
	MySQL	0,0447	0,0779	0,0572
	PostgreSQL	0,0209	0,0349	0,0259

Fast and Furious

- PostgreSQL is the clearly the fastest
- Oracle results for this testcase not really reliable, because of the limitation of the return value with just 2 decimal places
- Still slower than Postgres
- MySQL slowet with an average of 14 ms

Multi-User – Rush Hour

- PostgreSQL is the clearly the fastest when a lot of users are accessing the system at the same time
- Least needed time: Postgres (3 ms) vs. Oracle (60 ms)
- Postgres as its worst scenario is still faster than Oracle
- Oracle the worst in this scenario: it takes at cases up to 2 seconds
- Oracle is the most stable RDBMS: Average time just 0,7 ms slower than Best Case
- MySQL is also weak: In some cases it takes up to 1.5 seconds

Tower to Management

- PostgreSQL is the clearly the fastest when using aggregate functions in the queries
- Postgres needs around 16 ms less than Oracle and around 32 ms less than MySQL
- MySQL is clearly the slowest for reporting tasks
- MySQL needs at its worst case around 30 ms more than Oracle and around 40 ms more than PostgreSQL

Contact

Florian Nemling, Renad Quku
florian@nemling.eu , renadquk@gmail.com
Technische Universität Wien
Institute for Information Systems Engineering (E194)
Compilers and Languages (E194-5)
Argentinierstraße 8,1040 Wien - AT