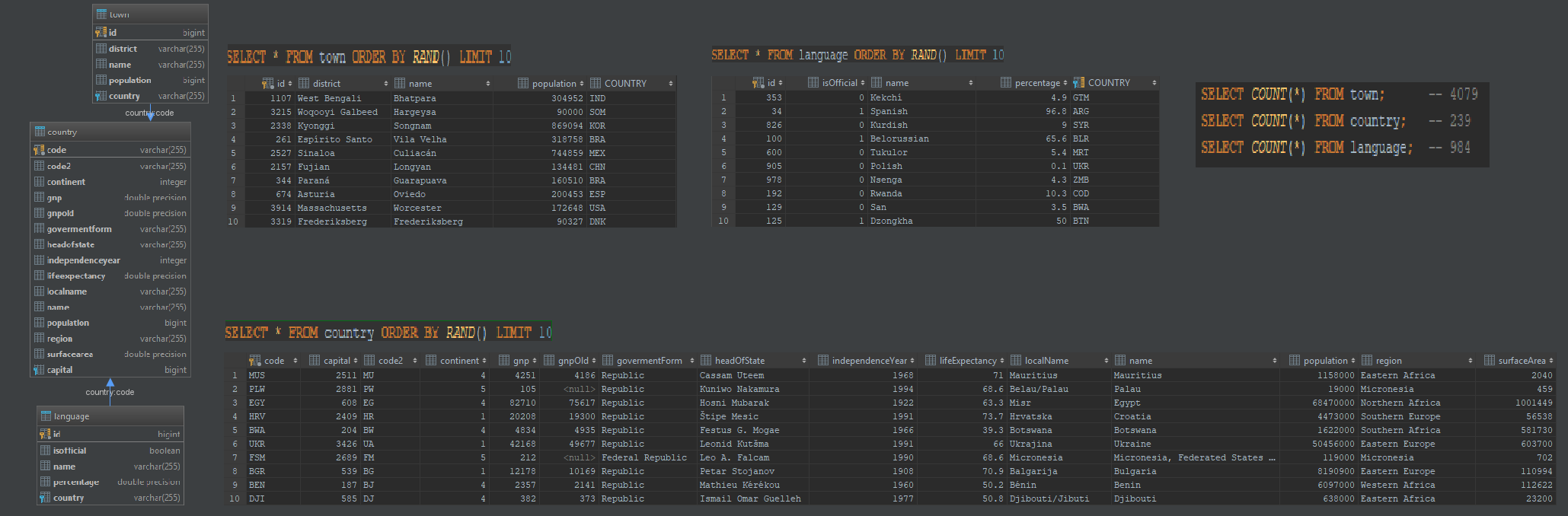
**Test report**

1. **Database schema:**
2. **Queries**

* **Most popular languages by region (version which works on every database engine)**

SELECT language.name AS language\_name, MAX(language\_aggregate.region) AS region, MAX(language\_aggregate.language\_population) AS language\_population FROM (

SELECT language\_sum.region, MAX(language\_sum.language\_population) AS language\_population FROM (

SELECT lang.name, cntr.region, sum((lang.percentage / 100) \* cntr.population) AS language\_population

FROM language lang

JOIN country cntr

ON lang.country = cntr.code

GROUP BY lang.name, cntr.region) language\_sum

GROUP BY language\_sum.region) language\_aggregate

JOIN country

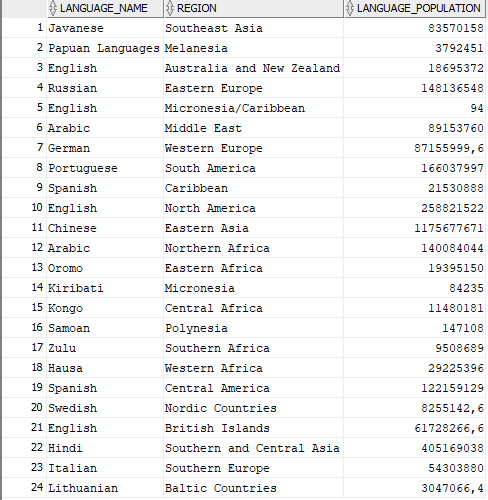
ON country.region = language\_aggregate.region

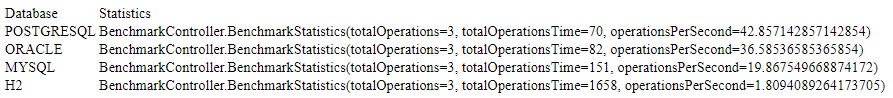
JOIN language

ON language.country = country.code

GROUP BY language.name, country.region

HAVING SUM((language.percentage / 100) \* country.population) = MAX(language\_aggregate.language\_population);





* **Most popular languages by region (version which works only on Oracle and PostgreSQL)**

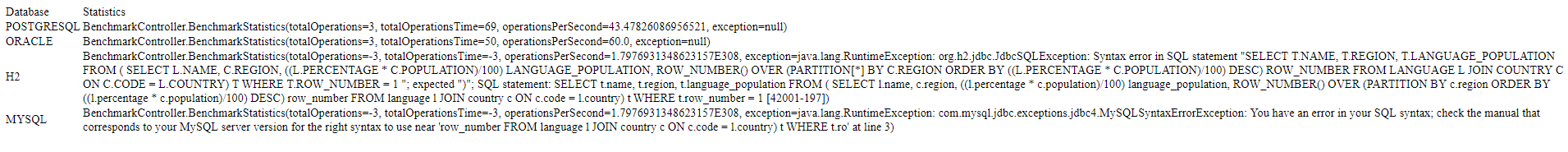
SELECT t.name, t.region, t.language\_population

FROM (

SELECT l.name, c.region, ((l.percentage \* c.population)/100) language\_population, ROW\_NUMBER() OVER (PARTITION BY c.region ORDER BY ((l.percentage \* c.population)/100) DESC) row\_number

FROM language l

JOIN country c ON c.code = l.country) t WHERE t.row\_number = 1

****

* **Most popular languages by region (version for Mongo)**

**db.language.aggregate([{$lookup: {from:"country",localField:"country",foreignField:"code",as:"countryObj"}},{$project:{countryObj: 1, name: 1, percentage: 1,totalPopulation: {$multiply: [{ $divide: ["$percentage", 100]}, { $arrayElemAt: ["$countryObj.population", 1]}]}}},{$sort:{"countryObj.region":1,"totalPopulation":1}},{$group:{\_id:"countryObj.region",docs:{ $push:"$$ROOT"}}},{$project:{"top\_one": {$slice:["$docs", 1]}}}]);**

****

* **Cities from 10 biggest countries**

SELECT town.\*

FROM country

JOIN town

ON town.country = country.code

WHERE (

SELECT COUNT(\*)

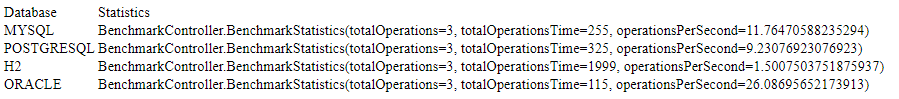
FROM country cntr

WHERE cntr.surfacearea >= country.surfacearea

) <= 10

ORDER BY town.population DESC;





* **Cities from 10 biggest countries (mongodb version)**

**db.country.aggregate([ {$lookup: {from: "town", localField: "code", foreignField: "country", as: "countryTown"}}, {$sort: {surfacearea: -1}}, {$limit: 10} ]);**

****

* **Top 3 cities from every region based on population**

SELECT town.\*, country.region

FROM town

JOIN country

ON town.country = country.code

WHERE (

SELECT COUNT(\*)

FROM town twn

JOIN country cntr

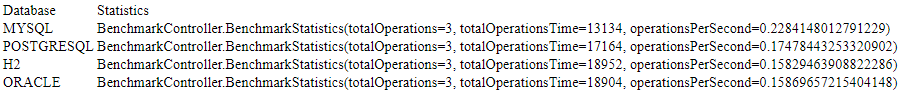
ON twn.country = cntr.code

WHERE twn.population >= town.population AND country.region = cntr.region

) <= 3

ORDER BY country.region;





* **Top 3 cities from every region based on population (mongo version)**

db.town.aggregate([{$lookup:{from:"country",localField:"country",foreignField:"code",as:"countryObj"}},{$sort:{"countryObj.region":1,"population":1}},{$group:{\_id:"countryObj.region",docs:{$push:"$$ROOT"}}},{$project:{"top\_three": {$slice:["$docs", 3]}}}]);



1. **Conclusions**

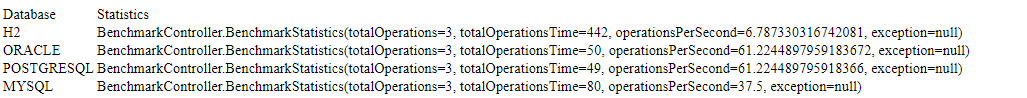
Three of the testes databases can be used in production environment, Oracle, MySQL, PostgreSQL. H2 is here only for contrast. As you can see H2 have worst execution times for every query, in first case H2 was about 20 times worst than other databases.

Let’s then focus on these three databases mentioned before, in each case different database won, but in general, Oracle had the most stable results, it of course depends on many factors, for example all of these engines works simultaneously on my local machine, which of course is not the best testing environment.

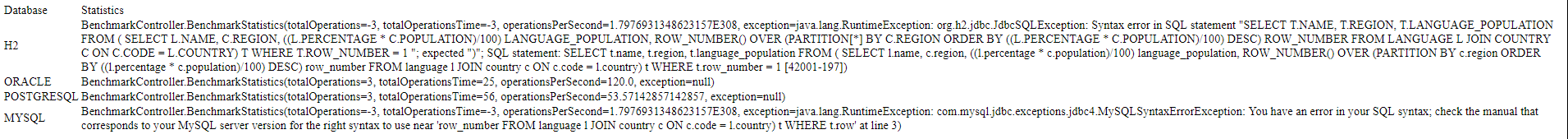
Another case are indexes, in each queries there was some subqueries, this means at least square complexity, indexes on COUNTRY.POPULATION, COUNTRY.SURFACEAREA and TOWN.POPULATION would for sure speed up a bit all the queries, there can be more indexes of course, but I think, these on mentioned numeric values, are most valuable, as I often searched using these values. Additionally, it is difficult to find two same values for mentioned columns, and that’s good from index point of view.

Times after indexes was added on mentioned columns:

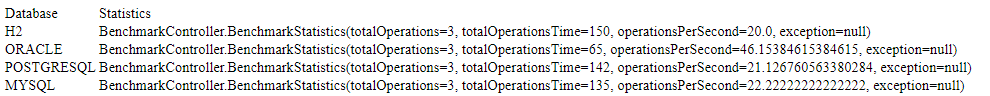
* **Most popular languages by region (version which works on every database engine)**

****

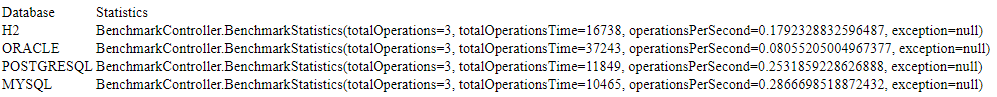
* **Most popular languages by region (version which works only on Oracle and PostgreSQL)**

****

* **Cities from 10 biggest countries**

****

* **Top 3 cities from every region based on population**



With indexes on tables I had an average 25% time improvement. But of course this have some impact on tables initialization, on every application bootstrap, databases are filled with new data, now it takes more time than before.