2023/2024

Ciências ULisboa
Faculdade de Ciências da Universidade de Lisboa

Base de Dados Avançadas

Project Report

Group 2

Artur Costa nº 62740

Enzo Souza nº 62744

Ivo Estrela nº 51051

Description of the Dataset

This dataset encompasses an extensive compilation of over 33,000 job listings. Each specific posting is equipped with 27 significant characteristics, encompassing details like job title, job description, salary, location, application URL, and work types. Additionally, there are distinct files for benefits, skills, and industries related to each job listing. Most positions are correlated with a corresponding company, the details of which are provided in a separate CSV file. This file includes attributes such as company description, headquarters location, number of employees, and follower count. To streamline and to not bloat our project we decided to not use some of the separate csv files mentioned above, with the remaining files being the following:

job_postings.csv

- **job_id**: The job ID as defined by LinkedIn (https://www.linkedin.com/jobs/view/ *job_id*)
- **company_id**: Identifier for the company associated with the job posting (maps to companies.csv)
- **title**: Job title.
- **description**: Job description. - **REMOVED**
- **max_salary**: Maximum salary
- **med_salary**: Median salary
- **min_salary**: Minimum salary
- **pay_period**: Pay period for salary (Hourly, Monthly, Yearly)
- **formatted_work_type**: Type of work
- **location:** Job location
- **applies**: Number of applications that have been submitted
- **original_listed_time**: Original time the job was listed
- **remote_allowed**: Whether job permits remote work
- **views**: Number of times the job posting has been viewed
- **job_posting_url**: URL to the job posting on a platform
- **application_url**: URL where applications can be submitted
- **application_type**: Type of application process
- **expiry**: Expiration date or time for the job listing
- **closed_time**: Time to close job listing
- **formatted_experience_level**: Job experience level
- **skills_desc**: Description detailing required skills for job
- **listed_time**: Time when the job was listed
- **posting_domain**: Domain of the website with application
- **sponsored**: Whether the job listing is sponsored or promoted.
- **work_type**: Type of work associated with the job
- **currency**: Currency in which the salary is provided.
- **compensation_type**: Type of compensation for the job.

companies.csv

- **company_id**: The company ID as defined by LinkedIn
- **name**: Company name
- **description**: Company description
- **company_size**: Company grouping based on number of employees (0 Smallest - 7 Largest)
- **country**: Country of company headquarters.
- **state**: State of company headquarters.
- **city**: City of company headquarters.
- **zip_code**: ZIP code of company's headquarters.
- **address**: Address of company's headquarters
- **url**: Link to company's LinkedIn page

employee_counts.csv

- **company_id**: The company ID
- **employee_count**: Number of employees at company
- **follower_count**: Number of company followers on LinkedIn
- **time_recorded**: Unix time of data collection

benefits.csv

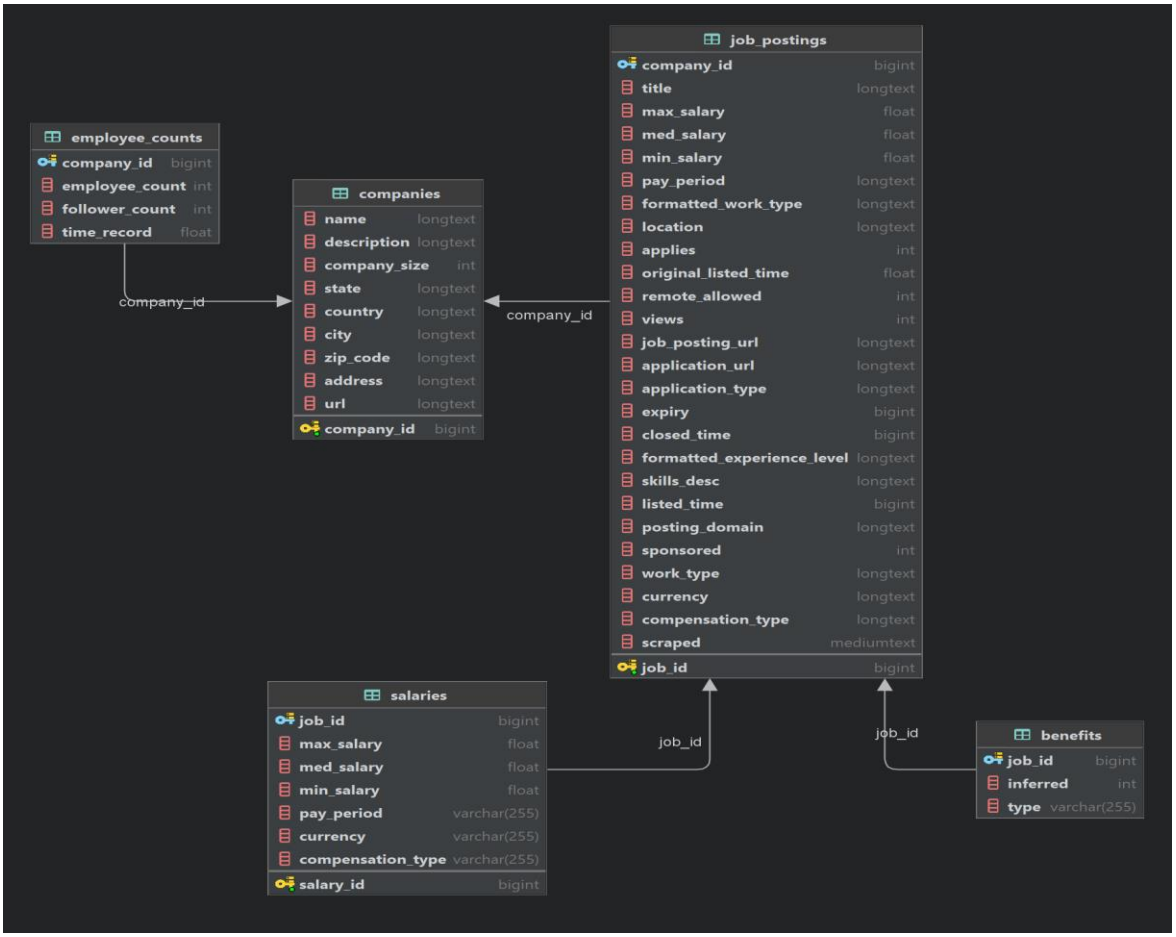
- **job_id**: The job ID
- **type**: Type of benefit provided (401K, Medical Insurance, etc)
- **inferred**: Whether the benefit was explicitly tagged or inferred through text by LinkedIn

salaries.csv

- **job_id**: The job ID
- **salary_id**: The salary id
- **max_salary:** Maximum salary
- **med_salary:** Median salary
- **min_salary:** Minimum salary
- **pay_period:** Pay period for salary (Hourly, Monthly, Yearly)
- **currency:** Type of currency
- **compensation_type:** Method of compensation

Scheme for both Databases

MySQL



MongoDB

Discussion

Queries Simples:

3a - 1

Esta consulta busca os salários que são pagos anualmente, com um valor superior a 10000 e com um salário máximo superior a 50000.

MySQL - Results with no optimization:

```
Length of the result for 3a-1 :  119
SQL Execution time: 0.010532140731811523
(1, 'SIMPLE', 'sa', None, 'ALL', None, None, None, None, 259,
1.11, 'Using where')
Avg total time SQL-Query 3a-1  =  0.00583195686340332
```

For optimization of this query, we decided to use a composed index in pay_period, max_salary, min_salary to help the query. Here are the following results:

```
Length of the result for 3a-1 :  119
SQL Execution time: 0.004030704498291016
(1, 'SIMPLE', 'sa', None, 'range', 'index_salary2',
'index_salary2', '1028', None, 119, 33.33, 'Using where; Using
 index')
Avg total time SQL-Query 3a-1  =  0.00576314926147461
```

Since the table salaries is small, the difference in execution time is negligible. Nevertheless, the number of rows analyzed dropped from 259 (All rows) to 119.

Mongo - Results with no optimization:

```
Length of the result for 3a-1 :  119
Mongo Execution time: 0.09665179252624512
Avg total time Mongo-Query 3a-1  =  0.013785266876220703
```

We used the same optimization strategy to produce a slight improvement in execution time:

```
Length of the result for 3a-1 :  119
Mongo Execution time: 0.01770639419555664
Avg total time Mongo-Query 3a-1  =  0.006117701530456543
```

3a - 2

A próxima utiliza uma expressão de tabela comum e a função de janela ROW_NUMBER() para classificar as empresas com base no número de ofertas de emprego em cada localização. O resultado inclui apenas as empresas com o maior número de ofertas de emprego em cada local.

Relativamente à otimização do MySQL e do Mongo, nesta query não conseguimos realizar nenhuma otimização.

Queries Complexas

3b – 1

Esta consulta recupera informações sobre empresas, incluindo o seu ID, nome, número de empregados e o número de ofertas de emprego em que o título contém "er". Também filtra as empresas com mais de 5 ofertas de emprego e ordena os resultados pela contagem de ofertas por ordem decrescente. A utilização de LEFT JOINs garante que as empresas sem entradas correspondentes nas tabelas benefits ou employee_counts continuam a ser incluídas nos resultados.

MySQL results with no optimization:

```
Length of the result for 3b-1 :  174
SQL Execution time: 0.1979987621307373
(1, 'SIMPLE', 'jp', None, 'ALL', None, None, None, None, 16123, 11.11, 'Using where; Using temporary; Using
filesort')
(1, 'SIMPLE', 'c', None, 'eq_ref', 'PRIMARY', 'PRIMARY', '8', 'mysql.jp.company_id', 1, 100.0, None)
(1, 'SIMPLE', 'ec', None, 'ref', 'company_id', 'company_id', '9', 'mysql.c.company_id', 1, 100.0, None)
Avg total time SQL-Query 3b-1  =  0.35470578670501707
```

MySQL results with optimization

For optimization of this query, we decided to use two composed indexes. One in company_id(companies) and the other in company_id(employee_counts). Both help the query. The following results didn't show a significant increasing performance:

```
Length of the result for 3b-1 :  174
SQL Execution time: 0.19666409492492676
(1, 'SIMPLE', 'jp', None, 'ALL', None, None, None, None, 16123, 11.11, 'Using where; Using temporary; Using
filesort')
(1, 'SIMPLE', 'c', None, 'eq_ref', 'PRIMARY,idx_companies_company_id', 'PRIMARY', '8', 'mysql.jp.company_id', 1,
100.0, None)
(1, 'SIMPLE', 'ec', None, 'ref', 'idx_employee_counts_company_id', 'idx_employee_counts_company_id', '9', 'mysql.c
.company_id', 1, 100.0, None)
Avg total time SQL-Query 3b-1  =  0.2749918222427368
```

Mongo results with no optimization:

```
Length of the result for 3b-1 :  174
Mongo Execution time: 130.22568154335022
```

Mongo results with optimization

For optimization of this query, we decided to use six composed indexes. One in title_index, one in company_id_index, one in job_id_index, one in company_company_id_index, one in group_index and the last one in project_index. All the indexes help the query. The following results show a significant increase in the performance of the query (31 seconds):

```
Length of the result for 3b-1 :  174
Mongo Execution time: 99.83905220031738
```

3b - 2

Essa consulta calcula os valores médio, mínimo e máximo da coluna max_salary da tabela salaries, considerando apenas as linhas em que os anúncios de emprego correspondentes têm um max_salary maior que 5000. O RIGHT JOIN garante que todas as linhas da tabela job_postings sejam incluídas, e as linhas correspondentes da tabela salaries sejam incluídas com valores NULL se não houver correspondência.

MySQL with no optimization:

```
Length of the result for 3b-2 :  1
SQL Execution time: 0.32944560050964355
(1, 'SIMPLE', 'jp', None, 'ALL', None, None, None, None, 16123, 33.33, 'Using where')
(1, 'SIMPLE', 's', None, 'ref', 'job_id', 'job_id', '9', 'mysql.jp.job_id', 1, 100.0, 'Using
index')
Avg total time SQL-Query 3b-2  =  0.03912396430969238
```

MySQL with optimization

For optimization of this query, we decided to use two composed indexes. One in job_id(salaries) and the other in job_id(job_postings). Both help the query. The following results didn't show a significant increasing performance:

```
Length of the result for 3b-2 :  1
SQL Execution time: 0.03265810012817383
(1, 'SIMPLE', 'jp', None, 'ALL', None, None, None, None, 16123, 33.33, 'Using where')
(1, 'SIMPLE', 's', None, 'ref', 'idx_salaries_job_id', 'idx_salaries_job_id', '9', 'mysql.jp
.job_id', 1, 100.0, 'Using index')
Avg total time SQL-Query 3b-2  =  0.032817935943603514
```

Mongo with no optimization:

```
Length of the result for 3b-2 :  1
Mongo Execution time: 5.0137858390808105
Avg total time Mongo-Query 3b-2  =  6.815635752677918
```

Mongo with optimization

We used the same optimization strategy (using the same indexes as MysSQL) to produce a big improvement in execution time (6 seconds):

```
Length of the result for 3b-2 :  1
Mongo Execution time: 0.06915068626403809
Avg total time Mongo-Query 3b-2  =  0.06299724578857421
```

Updates and Inserts

UPDATE benefits SET type = 'test' WHERE type = 'Medical insurance' LIMIT 10;

Esta consulta atualiza até 10 linhas na tabela benefits, alterando o valor da coluna type de 'Medical insurance' para 'test'. O LIMIT 10 garante que apenas um máximo de 10 linhas sejam atualizadas, mesmo que haja mais linhas que satisfaçam a condição.

INSERT INTO companies (company_id, name, description, company_size, state, country, city, zip_code, address, url) VALUES (1, 'Empresas Empresas', 'Fazemos tudo e mais alguma coisa', 15, 'CA', 'USA', 'Los Angeles', '2625-136', 'Rua 29 de Fevereiro', 'https://www.example.com');

Esta consulta insere na tabela companies um novo objeto com os valores especificados na query.

Relativamente ao UPDATE e ao INSERT, verificámos que as duas operações foram bem sucedidas paras as respetivas base de dados.

Algumas das trade offs relativamente às nossas estratégias de otimização que podemos ter são um aumento do espaço necessário de armazenamento, a complexidade da query pode aumentar substancialmente sem haver ganhos ao nível da performance e os índices podem aumentar variações na performance o que dificulta a medição da mesma.

No fim nós reparámos que conseguimos melhorar mais a performance no Mongo do que no MySQL, visto que o nosso esquema é relacional estando mais adaptado ao MySQL (que é uma base de dados relacional).

Description of how to replicate the project

No nosso projeto, utilizamos a plataforma de virtualização Docker para inicializar as bases de dados MySQL e Mongo. Para inicializar basta entrar na pasta “advanced_databases/job_positions” e escolher o sistema operativo adequado e correr ou o start.sh (em Linux) ou o start.bat (em Windows). Depois das bases de dados já estarem inicializadas no Docker, deve-se correr na pasta “advanced_databases/src/insert.py” para colocar os dados nas tabelas. Para limpar as tabelas basta correr o “advanced_databases/src/drop.py”.