

RELATÓRIO (PARTE 1, 2 e 3)

PROJETO FINAL - BANCO DE DADOS II

Geovany Carlos Mendes - 2018013746

Leonardo Vieira Ferreira - 2018009799

Thiago Silva Pereira - 2018008209

1. Definições

Nesse primeiro relatório para o trabalho de banco de dados II, coordenado pela docente Vanessa Cristina Oliveira de Souza, é apresentado as primeiras 3 etapas do projeto, onde é feito a seleção da API a ser trabalhada, Implementação do Banco e Desenvolvimento da aplicação para consumir os dados.

A API selecionada para o trabalho foi a API do GitHub, que comporta informações sobre sua plataforma como: usuários, repositórios, licenças, Informes entre outros. Com isso é montado o modelo relacional apresentado em 1.1 e todas as definições do banco descritas nesta seção. O modelo foi pensado para melhor dispor informações que podem ser consideradas úteis e essenciais para gerar um relatório sobre o tema. Índices e funções também são construídos com esse objetivo, focando a eficiência do resultado final.

A aplicação construída foi toda construída em javascript, na qual optamos por uma API que recebe requisições de um cliente e os dados são armazenados nos bancos. Cada solicitação tem sua própria rota que é chamada a partir de um outro cliente http

Sequelize: um ORM para realizar o mapeamento entre as tabelas e dados capturados da API;

Axios: uma biblioteca para realizar as requisições HTTP na API do Github;

Mongoose: um modelador de objetos para determinar as estruturas que seriam enviadas ao mongo e ter controle sobre o formato dos documentos;

Outras tecnologias utilizadas:

Insomnia: um cliente HTTP para testar as requisições a API que construímos;

MongoDB compass: uma GUI (Graphic User Interface) para gerenciamento do mongo.

MongoDB Atlas: um gerenciador para o mongoDB na nuvem.

Como a aplicação tinha o intuito de carregar o banco para prover uma quantidade de dados para análise, foram gerados dois scripts para cada entidade importante no banco (Issues, Licenses, Repos e Users), uma para persistir os dados no mongo e outro no postgres.

Na seção 2.0 é apresentado o modelo orientado a objetos também derivado da API selecionada. Finaliza-se com as seções 3 e 4 onde são apresentados as impressões baseadas nas tabelas geradas para o modelo relacional postgresQL e as coleções do mongo respectivamente.

1.1 Modelo Relacional

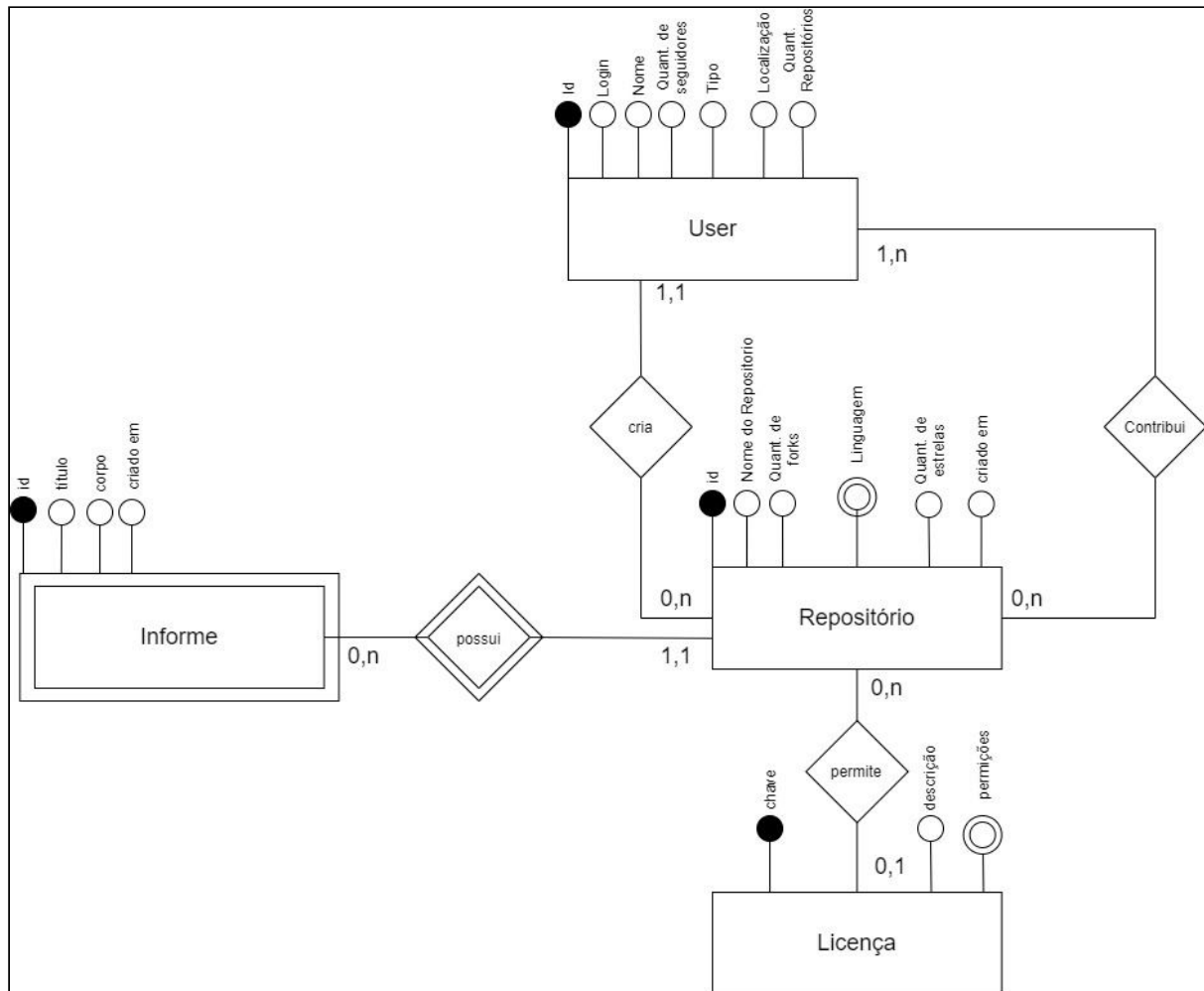


Imagem 01: Modelo Relacional desenvolvido para consumo da API

O modelo relacional foi construído em volta dos repositórios e usuários como informações principais, já que essas duas tabelas representam a base do GitHub. a partir desse meio foi desenvolvida ligação com outras informações que complementam a aplicação, como os informes de cada repositório ou as licenças utilizadas. É interessante observar que o informe se comporta como um entidade fraca de repositório, sendo seu id e escopo dependente do repositório. Outra peculiaridade que torna o modelo mais completo é a presença de atributos multivalorados, que podem vir a gerar novas tabelas em bancos relacionais, por exemplo.

1.2 Grupos de Usuários

Como o objetivo do projeto é descrito pela criação de um relatório ad-hoc para leitura e filtro de informações de uma API, nesse caso a GitHub API, é apenas importante para gerência de usuários que exista uma *role*, 'classe' de privilégios no postgres, para suprir as necessidades da aplicação (imagem 1), e outra *role* que cumpra os papéis de uma dba para a gerência do banco (imagem 2).

```
/*Criando role DBA BD*/
CREATE ROLE dba_github_database;
GRANT ALL PRIVILEGES ON DATABASE github_database TO dba_github_database WITH GRANT OPTION;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA gitdatabase TO dba_github_database WITH GRANT OPTION;
GRANT ALL PRIVILEGES ON SCHEMA gitdatabase TO dba_github_database WITH GRANT OPTION;
CREATE USER geovany WITH PASSWORD 'b2NvcnZvZGlzc2VudW5jYW1haXM=';
GRANT dba_github_database TO geovany;
ALTER ROLE dba_github_database WITH CREATEROLE;
ALTER ROLE geovany CREATEROLE;
```

Imagem 01: Código usado para a criação da role de DBA e um usuário geovany com as permissões da role.

```
--SCRIPT PARA O TRABALHO:
/*Criando role aplicação*/
CREATE ROLE aplicacion;
GRANT INSERT, SELECT, UPDATE, DELETE ON ALL TABLES IN SCHEMA gitdatabase TO aplicacion;
GRANT USAGE ON SCHEMA gitdatabase TO aplicacion;
CREATE USER app WITH PASSWORD 'VrCv%(6;9&';
GRANT aplicacion TO app;
```

Imagem 02: Código usado para criar a role da aplicação e um usuário app com as permissões da role.

1.3 Índices

Durante a otimização do banco foram criados diversos índices baseados nos filtros que pensado ser os mais recorrentemente usados, assim criando a necessidade de uso mais rápido e eficiente proveniente dos índices, mesmo com o custo de memória. Todos os índices criados são listados na tabela 01.

Tabela 01: Tabela de índices criados para o postgres

| | schemaname name | tablename name | indexname name | tablespace name | indexdef text |
|---|--------------------|-------------------|-------------------|--------------------|--|
| 1 | gitdatabase | repositories | index_full_name | [null] | CREATE INDEX index_full_name ON gitdatabase.repositories USING btree (full_name) |
| 2 | gitdatabase | users | index_user_login | [null] | CREATE INDEX index_user_login ON gitdatabase.users USING btree (login) |
| 3 | gitdatabase | users | index_user_name | [null] | CREATE INDEX index_user_name ON gitdatabase.users USING btree (name) |
| 4 | gitdatabase | users | index_user_local | [null] | CREATE INDEX index_user_local ON gitdatabase.users USING btree (location) |

Índices como o nome de usuário e nome do repositório são essenciais, uma vez que esse é o principal artefato fornecido pelo GitHub e portanto os mais pesquisados. no caso do full_name do repositório, ainda é possível usá-lo para encontrar o dono do repositório com mais facilidade em alguns casos, já que compõem de 'criador/repositório'.

Outras alternativas de busca interessante e que também podem ser recorrentes, é um filtro de usuários por localidade ou busca por login, já que alguns usuários podem não ter nome.

1.4 Triggers

Um trigger pensado para facilitar as atualizações futuras quanto a adição e remoção de funcionários, foi criado para o incremento ou decremento do número de repositórios públicos criados por determinado usuário. a imagem 3 representa o código usado na criação do trigger.

```
CREATE OR REPLACE FUNCTION att_num_repos() RETURNS TRIGGER AS $$ BEGIN
IF (TG_OP = 'INSERT') THEN
UPDATE gitdatabase.users SET qt_repos = qt_repos + 1 WHERE login = NEW.owner;
ELSEIF (TG_OP = 'DELETE') THEN
UPDATE gitdatabase.users SET qt_repos = qt_repos - 1 WHERE login = OLD.owner;
END IF;
RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER t_att_num_repos AFTER INSERT OR DELETE ON gitdatabase.repositories FOR EACH ROW
EXECUTE PROCEDURE att_num_repos();
```

Imagem 03: Código utilizado na construção do trigger.

2. Modelo Orientado a Objetos

```
"Issue":{  
  "title":"String",  
  "body":"String",  
  "created_at":"Date"  
}
```

Imagem 04: Modelo JSON tabela Issue

```
"License":{  
  "key":"String",  
  "description":"String",  
  "permissions":[  
  ]  
}
```

Imagem 05: Modelo JSON tabela License

```
"Permission":{  
  "name":"String"  
}
```

Imagem 06: Modelo JSON tabela Permission

```
"Repository":{  
  "full_name":"String",  
  "language":"String",  
  "forks":"Int",  
  "created_at":"Date",  
  "licenseid":"String",  
  "issues":[  
  ]  
}
```

Imagem 07: Modelo JSON tabela Repository

```
"User":{  
  "login":"String",  
  "name":"String",  
  "followers":"Number",  
  "type":"String",  
  "location":"String",  
  "public_repos":[  
  ],  
  "subscriptions":[  
  ]  
});
```

Imagem 08: Modelo JSON tabela User

3. Tabelas

| | | |
|---|-------------------------------|---|
|  | total_issues bigint |  |
| 1 | 3987 | |

| | | |
|---|---------------------------------|---|
|  | total_licenses bigint |  |
| 1 | 13 | |

| | | |
|---|------------------------------------|---|
|  | total_permissions bigint |  |
| 1 | 57 | |

| | | |
|---|-------------------------------------|---|
|  | total_repositories bigint |  |
| 1 | 33623 | |

| | | |
|---|------------------------------|---|
|  | total_users bigint |  |
| 1 | 1000 | |

| | | |
|---|------------------------------------|---|
|  | total_users_repos bigint |  |
| 1 | 20465 | |

Imagem 09: Count das tabelas do utilizadas e populadas no Postgres

4. Coleções





| Collection Name | Documents | Avg. Document Size | Total Document Size | Num. Indexes | Total Index Size | Properties |
|-----------------|-----------|--------------------|---------------------|--------------|------------------|---|
| issues | 2,942 | 1.1 KB | 3.2 MB | 1 | 76.0 KB |  |
| licenses | 13 | 466.2 B | 5.9 KB | 2 | 44.0 KB |  |
| repositories | 48,154 | 232.5 B | 10.7 MB | 1 | 468.0 KB |  |
| users | 1,100 | 7.2 KB | 7.7 MB | 2 | 60.0 KB |  |

Imagem 10: Count das coleções do utilizadas e populadas no mongoDB

LINK PARA O VÍDEO:

https://drive.google.com/file/d/1Y3iEvNuy3SItpLlyle0UBJV5_vQxSiJ/view?usp=sharing