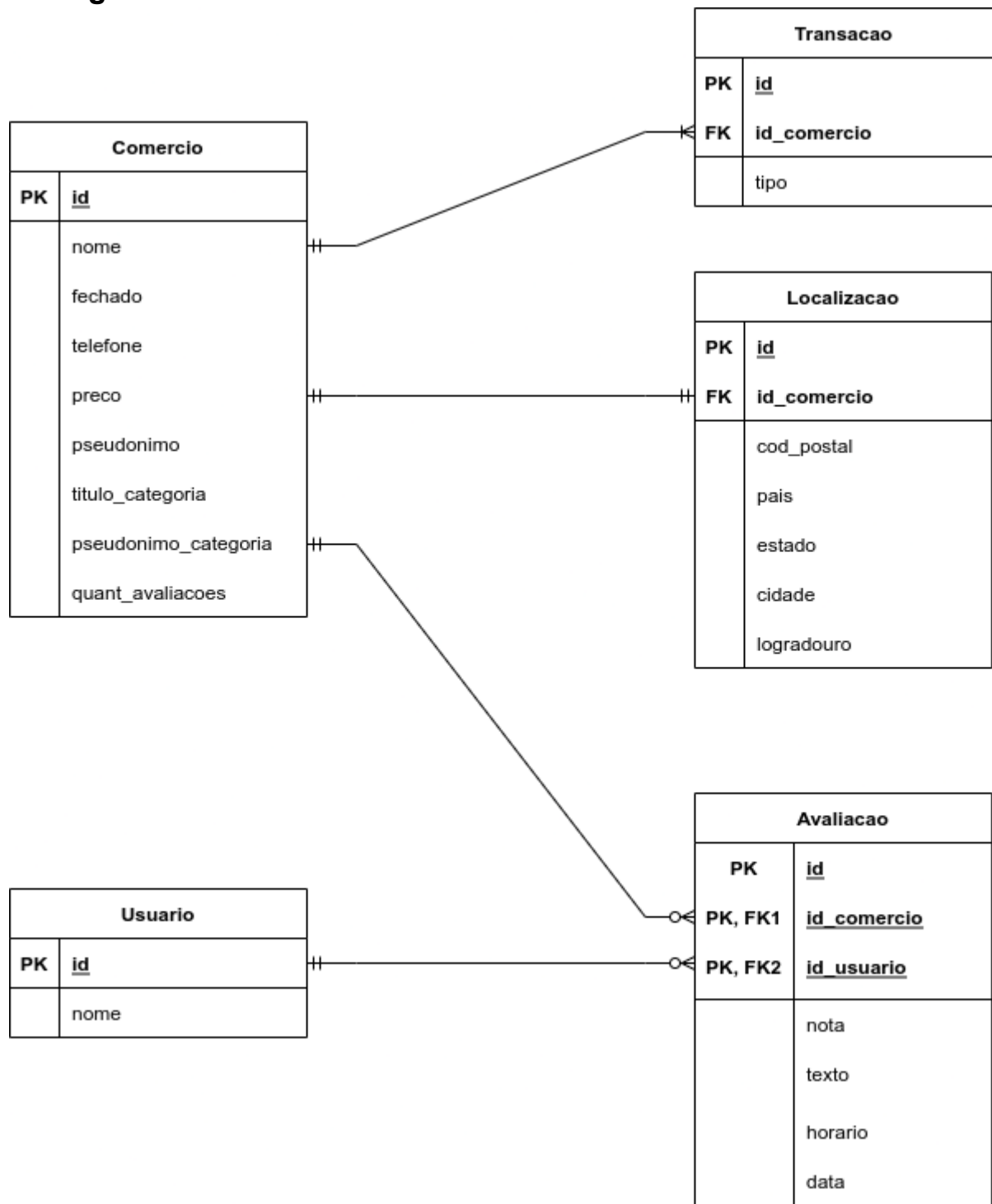


BANCO DE DADOS POPULADO COM YELP API

RELATÓRIO

Andrew Enrique Oliveira	2017020746
Luana de Cássia Freitas	2019009541
Lucas Lima Lordello	2019015941
Rolandro Aparecido Corrêa	2020016530

1. Diagrama relacional



2. Definição de grupos de usuários e suas permissões

- Criar cargo DBA

```
CREATE ROLE dba_yelp;  
GRANT ALL PRIVILEGES ON DATABASE "yelp" TO dba_yelp;  
GRANT ALL ON ALL TABLES IN SCHEMA public TO dba_yelp;
```

- Criar usuário DBA

```
CREATE USER dba WITH PASSWORD '12345';  
GRANT dba_yelp TO dba;
```

- Criar cargo Programador

```
CREATE ROLE programador_yelp;  
GRANT INSERT, SELECT, UPDATE ON TABLE public.avaliacao, public.comercio,  
public.localizacao, public.transacao, public.usuario TO programador_yelp;  
GRANT USAGE ON SCHEMA public TO programador_yelp;
```

- Criar usuários Programador

```
CREATE USER programador1 WITH PASSWORD '12345';  
GRANT programador_yelp TO programador1;  
CREATE USER programador2 WITH PASSWORD '12345';  
GRANT programador_yelp TO programador2;
```

3. Definição de índices e suas justificativas

```
CREATE INDEX cidadex ON public.localizacao USING btree (cidade);
```

Ao executar a consulta para buscar os usuários que fizeram avaliação de negócios em Los Angeles (código abaixo), foi observado um aumento de desempenho de aproximadamente 18% no tempo de execução, como demonstram os prints apresentados nesta seção.

```
Explain Analyze SELECT * FROM public.usuario WHERE id IN  
(SELECT id_usuario FROM public.avaliacao WHERE avaliacao.id_comercio IN  
(SELECT id_comercio FROM public.localizacao WHERE cidade = 'Los Angeles'))
```

- Antes da criação do índice – tempo de execução: 2.856 ms

Data Output Notifications Explain Messages

	QUERY PLAN	
	text	
1	Hash Semi Join (cost=209.84..282.17 rows=693 width=32) (actual time=1.847..2.784 rows=646 loops=1)	
2	[...] Hash Cond: ((usuario.id)::text = (avaliacao.id_usuario)::text)	
3	[...] -> Seq Scan on usuario (cost=0.00..53.43 rows=2943 width=32) (actual time=0.009..0.346 rows=2943 loops=1)	
4	[...] -> Hash (cost=201.18..201.18 rows=693 width=23) (actual time=1.783..1.785 rows=693 loops=1)	
5	[...] Buckets: 1024 Batches: 1 Memory Usage: 46kB	
6	[...] -> Hash Semi Join (cost=31.18..201.18 rows=693 width=23) (actual time=0.773..1.615 rows=693 loops=1)	
7	[...] Hash Cond: ((avaliacao.id_comercio)::text = (localizacao.id_comercio)::text)	
8	[...] -> Seq Scan on avaliacao (cost=0.00..153.29 rows=3429 width=46) (actual time=0.004..0.558 rows=3429 loops=1)	
9	[...] -> Hash (cost=28.29..28.29 rows=231 width=23) (actual time=0.296..0.296 rows=231 loops=1)	
10	[...] Buckets: 1024 Batches: 1 Memory Usage: 21kB	
11	[...] -> Seq Scan on localizacao (cost=0.00..28.29 rows=231 width=23) (actual time=0.084..0.239 rows=231 loops=1)	
12	[...] Filter: ((cidade)::text = 'Los Angeles'::text)	
13	[...] Rows Removed by Filter: 912	
14	Planning Time: 0.386 ms	
15	Execution Time: 2.856 ms	

- Depois da criação do índice – tempo de execução: 2.389 ms

Query Editor Query History Data Output Notifications Explain Messages

	QUERY PLAN	
	text	
1	Hash Semi Join (cost=208.51..280.84 rows=693 width=32) (actual time=1.539..2.328 rows=646 loops=1)	
2	[...] Hash Cond: ((usuario.id)::text = (avaliacao.id_usuario)::text)	
3	[...] -> Seq Scan on usuario (cost=0.00..53.43 rows=2943 width=32) (actual time=0.007..0.275 rows=2943 loops=1)	
4	[...] -> Hash (cost=199.84..199.84 rows=693 width=23) (actual time=1.486..1.488 rows=693 loops=1)	
5	[...] Buckets: 1024 Batches: 1 Memory Usage: 46kB	
6	[...] -> Hash Semi Join (cost=29.84..199.84 rows=693 width=23) (actual time=0.496..1.318 rows=693 loops=1)	
7	[...] Hash Cond: ((avaliacao.id_comercio)::text = (localizacao.id_comercio)::text)	
8	[...] -> Seq Scan on avaliacao (cost=0.00..153.29 rows=3429 width=46) (actual time=0.004..0.517 rows=3429 loops=1)	
9	[...] -> Hash (cost=26.96..26.96 rows=231 width=23) (actual time=0.154..0.155 rows=231 loops=1)	
10	[...] Buckets: 1024 Batches: 1 Memory Usage: 21kB	
11	[...] -> Bitmap Heap Scan on localizacao (cost=10.07..26.96 rows=231 width=23) (actual time=0.080..0.112 rows=231 l...	
12	[...] Recheck Cond: ((cidade)::text = 'Los Angeles'::text)	
13	[...] Heap Blocks: exact=4	
14	[...] -> Bitmap Index Scan on cidadex (cost=0.00..10.01 rows=231 width=0) (actual time=0.073..0.073 rows=231 loops=1)	
15	[...] Index Cond: ((cidade)::text = 'Los Angeles'::text)	
16	Planning Time: 0.694 ms	
17	Execution Time: 2.389 ms	

4. Definição de views

Foi criada uma view que exibe os dez melhores comércios de acordo com o valor médio de avaliações dos usuários, visto ser uma consulta pertinente para visualização em uma aplicação web em um contexto onde essa aplicação usaria o banco de dados.

```
CREATE OR REPLACE VIEW "10 Melhores Comercios por Avaliacao" AS
SELECT comercio.nome, comercio.preco, avg(avaliacao.nota)
FROM public.comercio INNER JOIN public.avaliacao
ON comercio.id = avaliacao.id_comercio
GROUP BY comercio.nome, comercio.preco
ORDER BY avg(avaliacao.nota) DESC
LIMIT 10;
```

Como resultado da consulta obtém-se o nome dos comércios, seu preço e a avaliação média.

```
SELECT * FROM "10 Melhores Comercios por Avaliacao";
```

	Data Output	Notifications	Messages	Explain
	 nome character varying (255) 🔒	preco character varying (4) 🔒	avg numeric 🔒	
1	Fisherman's Outlet	\$\$	5.0000000000000000	
2	Eight Korean BBQ	\$\$	5.0000000000000000	
3	Nini's Deli	\$	5.0000000000000000	
4	Fat Ducks Deli & Bakery	\$	5.0000000000000000	
5	OBAO	\$\$	5.0000000000000000	
6	Flub A Dub Chub's	\$	5.0000000000000000	
7	Dirt Dog	\$\$	5.0000000000000000	
8	Fogo de Chão	\$\$\$	5.0000000000000000	
9	Lady Yum	\$\$	5.0000000000000000	
10	La Nonna	\$\$	5.0000000000000000	

5. Triggers

Esse trigger tem como propósito fazer uma chamada à função quant_usuarios() após a remoção de algum usuário para informar a quantidade de usuários ainda registrados no banco de dados.

```
CREATE OR REPLACE FUNCTION quant_usuarios()
RETURNS TRIGGER LANGUAGE plpgsql
AS $$
BEGIN
    SELECT count(*) FROM usuario;
END;
```

```

$$;
CREATE TRIGGER confere_usuarios
    AFTER DELETE ON usuario
    FOR EACH ROW
    EXECUTE PROCEDURE quant_usuarios();

```

6. Procedures

Procedure criado com objetivo de inserir um usuário e uma avaliação do mesmo ao informar todos os valores necessários para tal nos parâmetros de `insereUsuario()`.

```

CREATE OR REPLACE PROCEDURE insereUsuario(id_com character varying(22),
id_user character varying(22), id_aval character varying(22), stars int,
comentario text, dia date, hora time, username character varying(22))
LANGUAGE SQL
AS $$
    INSERT INTO usuario VALUES (id_user, username);
    INSERT INTO avaliacao VALUES (id_com, id_user, id_aval, stars,
comentario, dia, hora);
$$;

```

7. Funções

Essa função realiza a inserção dos dados de um comércio nas tabelas comércio e localização, visto que as tabelas são relacionadas de modo que um registro de localização depende da existência do comércio correspondente no banco de dados.

```

CREATE OR REPLACE FUNCTION insereComercio(character varying(22), character
varying(255), boolean, character varying(20), character varying(4), character
varying(255), character varying(255), character varying(255), int, int,
character varying(20), character varying(255), character varying(255),
character varying(255), character varying(255))
RETURNS bool AS
$BODY$
    DECLARE
        id_comercio alias for $1;
        nome alias for $2;
        fechado alias for $3;
        telefone alias for $4;
        preco alias for $5;
        pseudo alias for $6;
        titulo_cat alias for $7;
        pseudo_cat alias for $8;
        quant_aval alias for $9;
        id_local alias for $10;
        cod_post alias for $11;
        pais alias for $12;
        estado alias for $13;
        city alias for $14;
        lograd alias for $15;
        ok bool;
BEGIN



```


```

        if lograd IN (SELECT logradouro FROM localizacao)
        AND city IN (SELECT cidade FROM localizacao WHERE logradouro =
lograd) then
            RAISE NOTICE 'Localização já existente.';
            ok = false;
        else
            INSERT INTO comercio VALUES (id_comercio, nome, fechado,
telefone, preco, pseudo, titulo_cat, pseudo_cat, quant_aval);
            INSERT INTO localizacao VALUES (id_comercio, id_local,
cod_post, pais, estado, city, lograd);
            RAISE NOTICE 'Comércio e localização inseridos com sucesso.';
            ok = true;
        END if;
    RETURN ok;
END;
$BODY$
LANGUAGE 'plpgsql' VOLATILE;

```

8. Print com o count das tabelas

Query Editor		Query History		
1	SELECT count(*) FROM public.comercio;			
2				
3				
Data Output		Notifications	Messages	Explain
	count bigint 			
1	1143			

Query Editor		Query History		
1	SELECT count(*) FROM public.localizacao;			
2				
3				
Data Output		Notifications	Messages	Explain
	count bigint			
1	1143			

Query Editor Query History

```
1 SELECT count(*) FROM public.transacao;  
2  
3
```

Data Output Notifications Messages Explain

	count bigint	
1	1813	

Query Editor Query History

```
1 SELECT count(*) FROM public.usuario;  
2  
3
```

Data Output Notifications Messages Explain

	count bigint	
1	2943	

Query Editor Query History

```
1 SELECT count(*) FROM public.avaliacao;  
2  
3
```

Data Output Notifications Messages Explain

	count bigint	
1	3429	