Universidade Tecnológica Federal do Paraná

Universidade Tecnológica Federal do Paraná - Campus Dois Vizinhos

Especialização em Ciência de Dados

Projeto Integrador

Alexandre Hannisch

Bruno Faustino Amorim

Leonardo Garcia

Nicolas Soffi

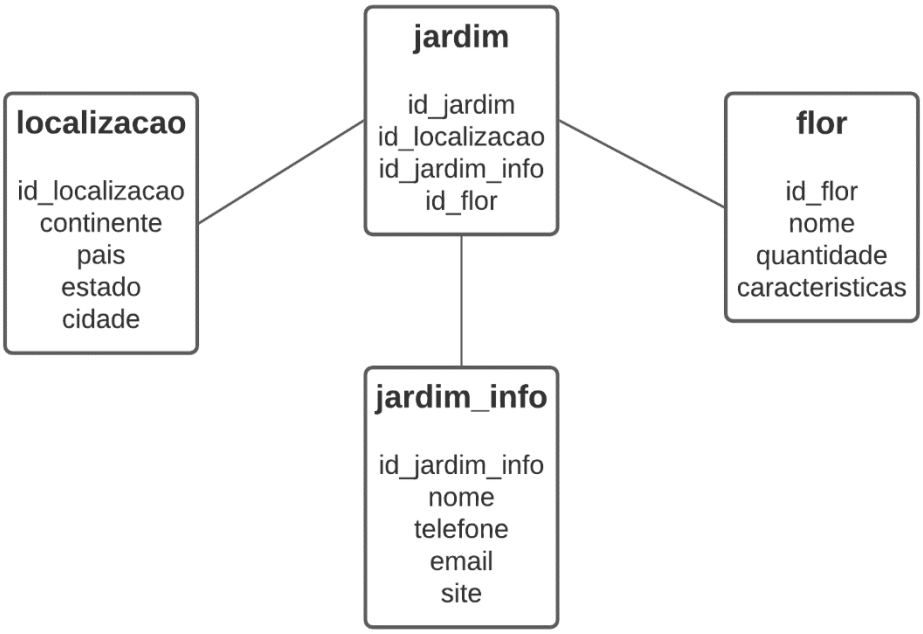
Dois Vizinhos - PR

2021

Universidade Tecnológica Federal do Paraná

Processamento Analítico de Dados

Projeto logico (desenho estrutural):



Projeto físico (Criação das tabelas)

```
create schema jardim;

create table jardim.jardim_info
(
	id_jardim_info serial primary key,
	nome VARCHAR,
	telefone VARCHAR,
	email VARCHAR,
	site VARCHAR
);

create table jardim.localizacao
(
	id_localizacao serial primary key,
	continente varchar,
	pais VARCHAR,
	estado varchar,
	cidade VARCHAR
);

create table jardim.flor
(
	id_flor serial primary key,
	nome varchar,
	quantidade INTEGER,
	caracteristicas FLOAT[]
);

create table jardim.jardim
(
	id_jardim serial primary key,
	id_jardim_info INTEGER,
	id_localizacao INTEGER,
	id_flor INTEGER,
	CONSTRAINT fk_jardim_info
	FOREIGN KEY(id_jardim)
	  REFERENCES jardim.jardim_info(id_jardim_info),
	CONSTRAINT fk_localizacao
	FOREIGN KEY(id_localizacao)
	  REFERENCES jardim.localizacao(id_localizacao),
	CONSTRAINT fk_flor
	FOREIGN KEY(id_flor)
	  REFERENCES jardim.flor(id_flor)
);
```

Descrever no documento quais são as hierarquias identificada

Localização: (all) ≤ (continente) ≤ (pais) ≤ (estado) ≤ (cidade)

Descrever no documento quais são as medidas identificadas

Flor: (quantidade)

Criar 4 grupos de consultas analíticas no data warehouse e suas características

Consulta 1:

```
select j.id_jardim, ji.nome, l.pais, f.nome
from
jardim.jardim as j, jardim.jardim_info as ji,
jardim.localizacao as l, jardim.flor as f
where
j.id_jardim_info = ji.id_jardim_info
and j.id_localizacao = l.id_localizacao
and j.id_flor = f.id_flor
and l.pais = 'Brasil'
```

Consulta o id, nome, país e nome das flores que existem nos jardins do Brasil

Consulta 2:

```
select l.continente, f.nome, sum(f.quantidade)
from
jardim.jardim as j, jardim.jardim_info as ji,
jardim.localizacao as l, jardim.flor as f
where
j.id_jardim_info = ji.id_jardim_info
and j.id_localizacao = l.id_localizacao
and j.id_flor = f.id_flor
group by
l.continente, f.nome
order by
l.continente
```

Consulta a soma da quantidade de flores de cada espécie por continente

Consulta 3:

```
select f.nome, avg(f.quantidade)
from
jardim.jardim as j, jardim.jardim_info as ji,
jardim.localizacao as l, jardim.flor as f
where
j.id_jardim_info = ji.id_jardim_info
and j.id_localizacao = l.id_localizacao
and j.id_flor = f.id_flor
group by
f.nome
```

Consulta a média da quantidade de flores por espécie nos jardins existentes na base de dados

Consulta 4:

```
select f.nome, l.pais, max(f.quantidade) as maior_quantidade
from
jardim.jardim as j, jardim.jardim_info as ji,
jardim.localizacao as l, jardim.flor as f
where
j.id_jardim_info = ji.id_jardim_info
and j.id_localizacao = l.id_localizacao
and j.id_flor = f.id_flor
and f.nome in ('bluebell','buttercup','iris','daffodil','crocus')
group by
f.nome, l.pais
order by
f.nome, maior_quantidade desc
```

Consulta a maior quantidade de flores por espécie e país, dada as espécies bluebell, buttercup, íris, daffodil e crocus.

Universidade Tecnológica Federal do Paraná

Introdução ao Big Data

Qual é o NoSQL escolhido e por que da sua escolha?

Optamos pelo MongoDB, pois contém um grupo de documentos que é a coleção, existe dentro de um banco de dados, pode possuir um conjunto de diferentes campos, os documentos possuem uma estrutura similar, não precisa de um esquema rígido, escala bem horizontalmente, é aprimorado para dados no volume de petabytes e segue o modelo BASE.

Qual é o modelo de dados do NoSQL escolhido?

Escolhemos o modelo orientado a documentos que é uma extensão do modelo chave-valor, onde o conceito básico do dado é o documento e cada documento pertence a uma coleção que por sua vez pertence a um database. Outras características é que é possível gerenciar informações de dados semi-estruturados.

Por que seria interessante migrar do PostgreSQL ao NoSQL escolhido? detalhar a motivação.

Seria interessante migrar para o MongoDB pois o foco é em flexibilidade e desempenho, é projetado para dados não-estruturados, é possível escalar horizontalmente por meio de sharding que particiona os dados por intervalos e atribui dados em varias instâncias e oferece grande disponibilidade pela reaplicação de dados que é uma solução que permite servidos de banco de dados tenham os mesmos dados, assim deixando mais fácil o balanceamento de carga e a redundância.

Como implantar as tabelas de dimensão e de fato no NoSQL escolhido?

É possível ter as tabelas de fato e dimensão da maneira abaixo através das várias coleções e gerar referencias entre os documentos delas, como abaixo:

Qual seria a sintaxe das consultas analíticas no NoSQL escolhido? apresente exemplos.

Abaixo algumas sintaxes de consultas no MongoDB:

db.localizacao.find({‘pais’:’Brasil’})

db.flor.fin().sort({‘nome’,’quantidade’})

db.jardim.find().limit(5)

db.localizacao.distinct(‘cidade’)

db.flor.find({quantidade: { $gte: 10000}})

Recuperação de Informação Baseada em Conteúdo

Nosso DW contém a modelagem para análise de dados de plantas. Uma das tabelas possui os vetores de características já calculados a armazenados. Assim, podemos utilizar consultas por similaridade para comparar as imagens, por exemplo, usando cálculos de funções de distância.

Em uma das nossas dimensões temos nossos vetores de características armazenados:

```
--============================================================================================
-- VETOR DE CARACTERISTICAS
--============================================================================================

select * from jardim.flor limit 10;
```

Data Output | Explain | Messages

	id_flor [PK] integer	nome character varying	quantidade integer	caracteristicas double precision[]
1	1	bluebell	798396	164403076092e-05,0.1627756620994623 4,0.052071068139186494,0.971890113073481}
2	2	bluebell	476189	164403076092e-05,0.1627756620994623 4,0.052071068139186494,0.971890113073481}
3	3	bluebell	636864	3011279068293069,0.1627758255901544 4,0.05207548308454056,0.9718895834936611}
4	4	buttercup	351991	5006270770855e-06,0.162775660016296 8,0.05208026106759719,0.9718900214467171}
5	5	buttercup	939921	91089837310129e-05,0.162775646091569,0.05208314826076169,0.9718900242512742}
6	6	buttercup	15076	271793015352022,0.1627755370022700 2,0.05208344822246062 4,0.9718898962005392}
7	7	coltsfoot	96391	3604490016064378,0.1627827186757991 3,0.05219274345703225,0.9718692801954626}
8	8	coltsfoot	25130	230834795250096,0.1627754635863614 4,0.05211024184867897 6,0.9718899326486079}
9	9	coltsfoot	75783	1475287317563438,0.1627043370184207 4,0.05261463536510773,0.9719620264265388}
10	10	cowslip	50749	0223186212314580 6,0.1627754323882644 2,0.052061743611795 3,0.9718900138638885}

```
--============================================================================================
-- CRIANDO FUNÇÕES DE DISTÂNCIA
--============================================================================================

create or replace function l1(elem1 float[], elem2 float[]) returns float as $$
declare
 size integer;
 somat float;
begin
 select cardinality(caracteristicas) into size from jardim.flor limit 1;
 somat := 0;
 for i in 1..size loop
    somat := somat + ABS(elem1[i] - elem2[i]);
 end loop;
 return somat;
end $$
language plpgsql;
```

```
18
19  create or replace function KNN(qc float[], k integer)
20   returns table (id integer, distance float) as $$
21   begin
22    return query
23        select id_flor, l1(caracteristicas,qc) as distance
24        from jardim.flor
25        order by l1(caracteristicas,qc) LIMIT k;
26   end $$
27   language plpgsql;
28
29
30  create or replace function RangeQuery(qc float[], radius float)
31   returns table (id integer, distance float) as $$
32   begin
33    return query
34        select id_flor, l1(caracteristicas,qc) as distance
35        from jardim.flor
36        where l1(caracteristicas,qc) <= radius;
37   end $$
38   language plpgsql;
```

Acima, realizamos a criação das funções de similaridade para os cálculos de distância.

Escolhemos como ponto de comparação a planta bluebell de id = 1

```
41  ---------------------------------------------------------------------------------------------
42  -- PROCESSANDO DISTÂNCIAS
43  ---------------------------------------------------------------------------------------------
44
45  -- Escolheremos a flor: bluebell
46  SELECT cast(caracteristicas as  float[]) FROM jardim.flor WHERE id_flor = 1
47
48  select id_flor, distance, nome from KNN((SELECT cast(caracteristicas as  float[]) as caracteristicas FROM jardim.flor
49  WHERE id_flor = 1), 5) as caracteristicas
50  inner join jardim.flor as flor on flor.id_flor = caracteristicas.id;
51
52  Data Output  Explain  Messages                                                    FROM jardi
```

	id_flor [PK] integer	distance double precision	nome character varying
1	2	0	bluebell
2	1	0	bluebell
3	3	13.065001082643509	bluebell
4	5	14.661564013620568	buttercup
5	15	15.217282363192387	crocus

```
51
52  select id_flor, distance, nome from RangeQuery((SELECT cast(caracteristicas as  float[]) as caracteristicas FROM jardim.flor
53  WHERE id_flor = 1), 5) as caracteristicas
54  inner join jardim.flor as flor on flor.id_flor = caracteristicas.id;
55
56  Data Output  Explain  Messages
```

	id_flor [PK] integer	distance double precision	nome character varying
1	1	0	bluebell
2	2	0	bluebell

Realizando as consultas vemos que a consulta por KNN identificou as flores buttercup e crocus como sendo as mais similares quando comparadas a nossa flor consultada (bluebell). Abaixo, temos imagens de exemplo: bluebell e crocus, respectivamente.