PROJETO DE BASES DE DADOS

Relatório Parte 3

Grupo 40 – Turno L12 – Sexta-feira 14:30		
Aluno		Esforço (horas)
Miguel Mano	99286	30
Pedro Rodrigues	99300	30
Stanislaw Talejko	99330	30

Professor do Laboratório: Rodrigo Sousa

main.sql

main.sql

```sql
/*  Trabalho realizado pelo grupo 40  */
/*  99286 99300 99330 */
DROP VIEW IF EXISTS Vendas;
DROP TABLE IF EXISTS evento_reposicao;
DROP TABLE IF EXISTS responsavel_por;
DROP TABLE IF EXISTS retalhista;
DROP TABLE IF EXISTS planograma;
DROP TABLE IF EXISTS prateleira;
DROP TABLE IF EXISTS instalada_em;
DROP TABLE IF EXISTS ponto_de_retalho;
DROP TABLE IF EXISTS IVM;
DROP TABLE IF EXISTS tem_categoria;
DROP TABLE IF EXISTS produto;
DROP TABLE IF EXISTS tem_outra;
DROP TABLE IF EXISTS super_categoria;
DROP TABLE IF EXISTS categoria_simples;
DROP TABLE IF EXISTS categoria;

CREATE TABLE categoria (
    nome                    VARCHAR(50)                 NOT NULL UNIQUE,
    PRIMARY KEY(nome));

CREATE TABLE categoria_simples (
    nome                    VARCHAR(50)                 NOT NULL,
    PRIMARY KEY(nome),
    FOREIGN KEY(nome)
        REFERENCES categoria(nome));

CREATE TABLE super_categoria (
    nome                    VARCHAR(50)                 NOT NULL,
    PRIMARY KEY(nome),
    FOREIGN KEY(nome)
        REFERENCES categoria(nome));
```

```sql
CREATE TABLE tem_outra (
    super_categoria         VARCHAR(50)                 NOT NULL,
    categoria               VARCHAR(50)                 NOT NULL,
    PRIMARY KEY(categoria),
    FOREIGN KEY(super_categoria)
        REFERENCES categoria(nome),
    FOREIGN KEY(categoria)
        REFERENCES categoria(nome)
    CHECK(super_categoria != categoria));

CREATE TABLE produto (
    ean                     INTEGER                     NOT NULL UNIQUE,
    cat                     VARCHAR(50)                 NOT NULL,
    descr                   VARCHAR(100),
    PRIMARY KEY(ean),
    FOREIGN KEY(cat)
        REFERENCES categoria(nome));

CREATE TABLE tem_categoria (
    ean                     INTEGER                     NOT NULL,
    nome                    VARCHAR(50)                 NOT NULL,
    PRIMARY KEY(ean, nome),
    FOREIGN KEY(ean)
        REFERENCES produto(ean),
    FOREIGN KEY(nome)
        REFERENCES categoria(nome));

CREATE TABLE IVM (
    num_serie               INTEGER                     NOT NULL,
    fabricante              VARCHAR(50)                 NOT NULL,
    PRIMARY KEY(num_serie, fabricante));

CREATE TABLE ponto_de_retalho (
    nome                    VARCHAR(50)                 NOT NULL,
    distrito                VARCHAR(50)                 NOT NULL,
    concelho                VARCHAR(50)                 NOT NULL,
    PRIMARY KEY(nome));
```

```sql
CREATE TABLE instalada_em (
    num_serie               INTEGER                 NOT NULL,
    fabricante              VARCHAR(50)             NOT NULL,
    local_                  VARCHAR(50)             NOT NULL,
    PRIMARY KEY(num_serie, fabricante),
    FOREIGN KEY(num_serie, fabricante)
        REFERENCES IVM(num_serie, fabricante),
    FOREIGN KEY(local_)
        REFERENCES ponto_de_retalho(nome));

CREATE TABLE prateleira (
    nro                     INTEGER                 NOT NULL,
    num_serie               INTEGER                 NOT NULL,
    fabricante              VARCHAR(50)             NOT NULL,
    altura                  INTEGER                 NOT NULL,
    nome                    VARCHAR(50)             NOT NULL,
    PRIMARY KEY(nro, num_serie, fabricante),
    FOREIGN KEY(num_serie, fabricante)
        REFERENCES IVM(num_serie, fabricante),
    FOREIGN KEY(nome)
        REFERENCES categoria(nome));

CREATE TABLE planograma (
    ean                     INTEGER                 NOT NULL,
    nro                     INTEGER                 NOT NULL,
    num_serie               INTEGER                 NOT NULL,
    fabricante              VARCHAR(50)             NOT NULL,
    faces                   INTEGER                 NOT NULL,
    unidades                INTEGER                 NOT NULL,
    loc                     VARCHAR(50)             NOT NULL,
    PRIMARY KEY(ean, nro, num_serie, fabricante),
    FOREIGN KEY(ean)
        REFERENCES produto(ean),
    FOREIGN KEY(nro, num_serie, fabricante)
        REFERENCES prateleira(nro, num_serie, fabricante));

CREATE TABLE retalhista (
    tin                     INTEGER                 NOT NULL UNIQUE,
    name_                   VARCHAR(50)             NOT NULL UNIQUE,    /* RI-RE7 unique(name)*/
    PRIMARY KEY(tin));

CREATE TABLE responsavel_por (
    nome_cat                VARCHAR(50)             NOT NULL,
    tin                     INTEGER                 NOT NULL,
    num_serie               INTEGER                 NOT NULL,
    fabricante              VARCHAR(50)             NOT NULL,
    PRIMARY KEY(num_serie, fabricante),
    FOREIGN KEY(num_serie, fabricante)
        REFERENCES IVM(num_serie, fabricante),
    FOREIGN KEY(tin)
        REFERENCES retalhista(tin),
    FOREIGN KEY(nome_cat)
        REFERENCES categoria(nome));

CREATE TABLE evento_reposicao (
    ean                     INTEGER                 NOT NULL,
    nro                     INTEGER                 NOT NULL,
    num_serie               INTEGER                 NOT NULL,
    fabricante              VARCHAR(50)             NOT NULL,
    instante                TIMESTAMP               NOT NULL DEFAULT CURRENT_TIMESTAMP,
    unidades                INTEGER                 NOT NULL,
    tin                     INTEGER                 NOT NULL,
    PRIMARY KEY(ean, nro, num_serie, fabricante, instante),
    FOREIGN KEY(ean, nro, num_serie, fabricante)
        REFERENCES planograma(ean, nro, num_serie, fabricante),
    FOREIGN KEY(tin)
        REFERENCES retalhista(tin));
```

ICs.sql

A RI-1 encontra-se na main.sql realizado através do CHECK na tabela tem_outra.

```sql
/* Exercicio 2.2 */

CREATE OR REPLACE FUNCTION ri_4()
RETURNS TRIGGER
AS $$
BEGIN
    IF NEW.unidades > (SELECT unidades FROM planograma WHERE ean = NEW.ean) THEN
        RAISE EXCEPTION 'O numero de unidades repostas num evento de reposicao tem de ser menor ou igual ao numero de unidades especificado no planograma.';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER ri_4
BEFORE UPDATE OR INSERT ON evento_reposicao
FOR EACH ROW EXECUTE PROCEDURE ri_4();

/* Exercicio 2.3 */

CREATE OR REPLACE FUNCTION ri_5()
RETURNS TRIGGER
AS $$
BEGIN
    IF New NOT IN produto OR NEW.ean NOT IN produto OR NEW.nro NOT IN prateleira THEN
    RAISE EXCEPTION 'Um produto so pode ser reposto numa prateleira que apresente uma das categorias desse produto.';
    END IF;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER ri_5
BEFORE UPDATE OR INSERT ON evento_reposicao
FOR EACH ROW EXECUTE PROCEDURE ri_5();

```

queries.sql

queries.sql

```sql
/* Exercicio 3.1 - Qual o nome do retalhista (ou retalhistas) responsaveis pela reposicao do maior numero de categorias? */

 with counts_cat as
 (

SELECT tin, COUNT(tin) AS num_cat
FROM responsavel_por
GROUP BY tin
)
SELECT DISTINCT name_
FROM retalhista NATURAL JOIN /* changed*/counts_cat
WHERE num_cat = (SELECT MAX(num_cat) FROM counts_cat);
/* Exercicio 3.2 - Qual o nome do retalhista ou dos retalhistas que sao responsaveis por todas as categorias simples? */

with os AS (
    SELECT tin
    FROM (responsavel_por INNER JOIN categoria_simples ON responsavel_por.nome_cat = categoria_simples.nome)
)
SELECT DISTINCT name_
FROM retalhista NATURAL JOIN os;
/* Exercicio 3.3 - Quais os produtos (ean) que nunca foram repostos? */


SELECT DISTINCT ean
FROM produto
WHERE ean NOT IN (SELECT ean FROM evento_reposicao AS repostos);
```

```sql

/* Exercicio 3.4 - Quais os produtos (ean) que foram repostos sempre pelo mesmo retalhistasta? */

CREATE OR REPLACE FUNCTION  num_retalhista_repor(ean_aux INTEGER)
RETURNS INTEGER AS
$$
DECLARE total INTEGER;
BEGIN
        SELECT COUNT(*) INTO total
            FROM evento_reposicao
            WHERE ean = ean_aux
            GROUP BY tin;
        RETURN  total;
END
$$  LANGUAGE    plpgsql;

SELECT DISTINCT ean
FROM produto
WHERE num_retalhista_repor(ean) = 1;

```

view.sql

```sql
DROP VIEW IF EXISTS Vendas;
CREATE VIEW Vendas (ean, cat, ano, trimestre, mes, dia_mes, dia_semana, distrito, concelho, unidades)
As
Select e.ean, cat, extract (YEAR From instante) as ano, extract (QUARTER From instante) as trimestre, extract (MONTH From instante) as mes,
 extract(DAY From instante) as dia_mes, extract(DOW From instante) as dia_semana, distrito, concelho, unidades
From evento_reposicao AS e NATURAL JOIN produto INNER JOIN
instalada_em
ON e.num_serie = instalada_em.num_serie AND e.fabricante = instalada_em.fabricante  INNER JOIN
ponto_de_retalho
ON instalada_em.local_ = ponto_de_retalho.nome
```

index.sql

Para melhor compreensão deste ponto, estudamos a função de dispersão (hash) que recebe como parâmetro um valor e determina o contentor. Sabe-se que no mesmo contentor pode haver diferentes valores e que as entradas são pesquisadas sequencialmente. É importante destacar a ideia de que são os melhores para seleção por igualdade. Já a árvore B+ (btree), apresenta outras vantagens como o facto de não ser necessário ter os blocos fisicamente juntos no disco e de que as folhas do índice se encontram sempre ordenadas. É também distinguível pela sua segurança e simplicidade. Por outro lado, como se trata de um banco de dados pequenos, tem-se noção que a eficiência deste índice será menor do que os índices comuns.

Assim sendo, após estas informações e alguns testes sobre o tempo de execução chegamos à conclusão de que estes seriam os melhores índices na prática, embora a teoria não o comprove totalmente.

```sql
DROP INDEX IF EXISTS nome_retalhista_index;
DROP INDEX IF EXISTS nome_retalhista_index2;
DROP INDEX IF EXISTS nome_cat_index;
DROP INDEX IF EXISTS count_ean_index;

/* Exercicio 7.1 */
CREATE INDEX nome_retalhista_index2 ON retalhista USING HASH(name_);
CREATE INDEX nome_retalhista_index ON responsavel_por USING HASH(TIN);

SELECT DISTINCT R.name_
FROM retalhista R, responsavel_por P
WHERE R.tin = P.tin and P.nome_cat = 'Peru';

/* Exercicio 7.2 */
CREATE INDEX nome_cat_index ON tem_categoria USING HASH(nome);
CREATE INDEX count_ean_index ON produto USING HASH(ean);

SELECT T.nome, count(T.ean)
FROM produto P, tem_categoria T
WHERE p.cat = T.nome and P.descr like 'P%'
GROUP BY T.nome;
```

categories.html

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>List accounts - Flask </title>
</head>
<body style="padding:20px">
{% if cursor %}
    <table border="2px">
        <thead>
            <tr>
                <th>nome </th>
                <th>Adicionar Categoria </th>
                <th>Remover Categoria </th>
            </tr>
        </thead>
        <tbody>
            {% for record in cursor %}
                <tr>
                    <td>{{ record[0] }} </td>
                    <td><a href="categories/remove?nome={{ record[0] }}">Remover Catergoria </a> </td>
                </tr>
            {% endfor %}
        </tbody>
    </table>
{% else %}
    <p> Erro: não foi possível obter dados da base de dados! </p>
{% endif %}
<div>
    <button onclick="window.location.href='add_category?nome={{ params.get('nome') }}'">Adicionar categoria</button>
</div>

</body>
</html>
```

add_category.html

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <title>Add Category - Flask </title>
</head>

<body>
    <h3>Nome da categoria {{ params.get("nome") }} </h3>
    <form action="categories/add" method="post">
        <p>Nova Categoria: <input type="text" name="nome" /> </p>
        <p><input type="submit" value="add"> </p>
    </form>
    </div>
    <div>
        <button onclick="window.location.href='categories'>Voltar</button>
<div>
</body>
</html>
```

retailer.html

```
<!DOCTYPE html>
<html>

<head>
    <meta charset="utf-8">
    <title>List retailers - Flask </title>
</head>

<body style="padding:20px">
    {% if cursor %}
    <table border="2px">
        <thead>
            <tr>
                <th>nome</th>
                <th>tin</th>
                <th>Adicionar Categoria </th>
                <th>Remover Categoria </th>
            </tr>
        </thead>
        <tbody>
            {% for record in cursor %}
            <tr>
                <td>{{ record[0] }} </td>
                <td>{{ record[1] }} </td>
                <td><a href="retailer/remove?nome={ { record[0] }& { record[1] }}">Remover retalhista </a> </td>
            </tr>
            {% endfor %}
        </tbody>
    </table>
    {% else %}
    <p> Erro: não foi possível obter dados da base de dados! </p>
    {% endif %}
    <div>
        <button
            onclick="window.location.href='add_retailer?nome,tin={{ params.get('nome') }}&{ params.get('tin')}}'">Adicionar
            retalhista</button>
    </div>
</body>

</html>
```

add_retailer.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Add retailer - Flask </title>
</head>
<body>
    <form action="retailer/add" method="post">
    <p>Novo retailer: <input type="text" name="nome" /> </p>
    <p>tin retailer: <input type="text"  tin="tin"/> </p>
    <p><input type="submit" value="add" > </p>
    </form>
    </div>
    <div>
        <button onclick="window.location.href='retailer'>Voltar</button>
    <div>
</body>
</html>
```

listEvents.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>List retailers - Flask </title>
</head>
<body style="padding:20px">
{% if cursor %}
    <table border="2px">
        <thead>
            <tr>
                <th>cat</th>
                <th>unidades</th>

            </tr>
        </thead>
        <tbody>
            {% for record in cursor %}
                <tr>
                    <td>{{ record[0] }} </td>
                    <td>{{ record[1] }} </td>
                </tr>
            {% endfor %}
        </tbody>
    </table>
{% else %}
    <p> Erro: não foi possível obter dados da base de dados! </p>
{% endif %}
</body>
</html>
```

listCategoriesFromSuper.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>List retailers - Flask </title>
</head>
<body style="padding:20px">
{% if cursor %}
    <table border="2px">
        <thead>
            <tr>
                <th>nome</th>

            </tr>
        </thead>
        <tbody>
            {% for record in cursor %}
                <tr>
                    <td>{{ record[0] }} </td>
                </tr>
            {% endfor %}
        </tbody>
    </table>
{% else %}
    <p> Erro: não foi possível obter dados da base de dados! </p>
{% endif %}
</body>
</html>
```