Bases de Dados 9090

2022/2023

Projeto - Enunciado 3

A terceira parte do projeto consiste no desenvolvimento de restrições de integridade complexas, concepção de consultas SQL avançadas, criação de um protótipo de aplicação web e concepção de consultas OLAP.

0. Carregamento da Base de Dados

Carregue o esquema de Base de Dados apresentado no **Anexo A**. Crie as instruções para o seu preenchimento de forma consistente, garantindo que todas as consultas SQL e OLAP, apresentadas mais adiante, produzam um **resultado não vazio**. Deve ainda garantir que as moradas inseridas são nacionais e seguem o formato Português, terminando com código postal: XXXX-XXX e de seguida a localidade.

A criação de registos e o carregamento podem ser realizados através do método que lhe pareça mais adequado (manualmente, a partir de uma folha Excel, através de um script SQL, Python ou outro).

1. Restrições de Integridade

Tendo por base o esquema de base de dados definido no ponto anterior, apresente o código para implementar as seguintes restrições de integridade, se necessário, com recurso a extensões procedimentais SQL (Stored Procedures e Triggers):

(RI-1) Nenhum empregado pode ter menos de 18 anos de idade

(RI-2) Um 'Workplace' é obrigatoriamente um 'Office' ou 'Warehouse' mas não pode ser ambos

(RI-3) Uma 'Order' tem de figurar obrigatoriamente em 'Contains'.

O uso de extensões procedimentais (Stored Procedures e Triggers) deve ser reservado a restrições de integridade que não podem ser implementadas usando outros mecanismos mais simples. No entanto, os mecanismos **ON DELETE CASCADE** e **ON UPDATE CASCADE** não são permitidos.

2. SQL

Apresente a consulta SQL mais sucinta[1] para cada uma das seguintes questões:

1. Qual o número e nome do(s) cliente(s) com maior valor total de encomendas pagas?
2. Qual o nome dos empregados que processaram encomendas em todos os dias de 2022 em que houve encomendas?[2]
3. Quantas encomendas foram realizadas mas não pagas em cada mês de 2022?

3. Vistas

Crie uma vista que resuma as informações mais importantes sobre as vendas de produtos, combinando informações de diferentes tabelas do esquema de base de dados. A vista deve ter o seguinte esquema:

product_sales(sku, order_no, qty, total_price, year, month, day_of_month, day_of_week, city)

No esquema da vista, há as seguintes correspondências entre os seus atributos e os das tabelas:

- *sku*, *order_no*: correspondem à chave primária da tabela contains, mas apenas devem ser incluídas orders que foram pagas
- *qty*: corresponde ao atributo da tabela contains
- *total_price*: produto de qty e price
- *year*, *month*, *day_of_month*, *day_of_week*: atributos derivados do atributo date
- *city*: atributo derivado da morada de customer[3]

4. Desenvolvimento de Aplicação

Crie um protótipo de aplicação web, consistindo em scripts Python CGI e páginas HTML que permita:
a) Registar e remover produtos e fornecedores;
b) Alterar preços de produtos e respectivas descrições
c) Registar e remover clientes
d) Realizar encomendas
e) Simular o pagamento de encomendas

---

[1] Não pode usar instruções SQL que não fazem parte do SQL standard (como por exemplo a instrução `LIMIT`).
[2] Pode utilizar a função EXTRACT() do POSTGRES para obter dias, meses, etc, a partir de datas ou timestamps.
[3] Pode utilizar a função SUBSTRING() especificando um padrão POSIX para extrair a cidade após o código postal da morada.

A solução deve prezar pela segurança, prevenindo ataques por SQL INJECTION. Além disso, a **atomicidade das operações** sobre a base de dados deve estar assegurada com recurso a transações. Embora não seja fundamental para a avaliação, pode-se utilizar CSS ou outras "frameworks" para melhorar o aspecto gráfico.
A aplicação deve estar disponível online, no servidor sigma do IST (na área web de um dos alunos do grupo), desde a data de entrega do projeto até à data da discussão.

5. Consultas OLAP

Usando a vista desenvolvida para a Questão 4, escreva duas consultas SQL que permitam analisar:

1. As quantidade e valores totais de venda de cada produto em 2022, globalmente, por cidade, por mês, dia do mês e dia da semana
2. O valor médio diário das vendas de todos os produtos em 2022, globalmente, por mês e dia da semana

A solução submetida deve usar as instruções ROLLUP, CUBE, GROUPING SETS ou as cláusulas UNION of GROUP BY.

6. Índices

Apresente as instruções de criação do(s) índice(s) SQL para melhorar os tempos de consulta para cada um dos casos listados abaixo, explicando quais são as operações que seriam otimizadas e como.

Indique, com a devida justificação, que tipo de índice(s), sobre qual(is) atributo(s) e sobre qual(is) tabela(s) faria sentido criar, de forma a agilizar a execução de cada consulta. Suponha que o tamanho das tabelas exceda a memória disponível em várias ordens de magnitude.

Suponha que não existam índices nas tabelas, além daqueles implícitos ao declarar chaves primárias e estrangeiras.

6.1 -

```
SELECT order_no
FROM orders
JOIN contains USING (order_no)
JOIN product USING (SKU)
WHERE price > 50 AND
EXTRACT(YEAR FROM date) = 2023
```

6.2 -

```
SELECT order_no, SUM(qty*price)
FROM contains
JOIN product USING (SKU)
WHERE name LIKE 'A%'
GROUP BY order_no;
```

Entrega

O projeto será avaliado com base no relatório apresentado e na discussão. O relatório deve conter todas as respostas aos itens solicitados acima. Na tabela abaixo estão listados os pontos atribuídos a cada parte do trabalho.

Item	**Pontos**
Restrições de Integridade	2
Consultas SQL	4
Vista	1
Aplicação	6
Análise de dados	4
Índices	3

Relativamente à aplicação, os seguintes aspetos, ainda que não fundamentais para a avaliação, serão apreciados sob a forma de um bónus:

Aspeto gráfico polido da aplicação	Bónus +1.0
Lógica de paginação para listas	Bónus +1.0

Submeta um ficheiro **`entrega-bd-02-GG.zip`**[4], onde **GG** é o número do grupo, estruturado segundo uma das seguintes opções:

[4] ⚠ O formato do arquivo deve ser exclusivamente ZIP ou GZ. Outros formatos de arquivo não serão aceites.

1. **Opção Recomendada:**

`E3-report-GG.ipynb` (onde GG é o número do grupo)	**Um ficheiro Jupyter Notebook correspondendo ao preenchimento do template "E3-report.ipynb" disponibilizado na página da disciplina, com o nome modificado para incluir o número do grupo** Deverá preencher a informação da "folha de rosto" com a indicação do **número do grupo**, o **número e nome** dos alunos que o constituem, tal como **a percentagem relativa de contribuição** de cada aluno com o respectivo **esforço (horas)**, o **turno** a que o grupo pertence e o **nome do docente de laboratório** responsável. Deverá preencher cada uma das secções subsequentes: 0. Instruções SQL para popular a base de dados (opcionalmente pode colocá-los num ficheiro "populate.sql" separado, que é importado no report e incluído no zip da entrega) 1. Implementação de **Restrições de Integridade** em SQL 2. Código para **Consultas SQL** 3. Código para criação de **Vistas** 4. Explicação da **Aplicação** desenvolvida, incluindo a descrição do código (todo o código da aplicação deve ser incluído na entrega) 5. Código para **Consultas OLAP** 6. Respostas aos **Índices** O ficheiro **"E3-report.ipynb"** pode ser importado para o ambiente de trabalho disponibilizado para as aulas de laboratório[5] (basta colocar na pasta work/), que serve de ambiente de teste para as partes em SQL. Deve popular a base de dados de forma a assegurar que o resultado das suas queries é não-vazio. Deve ainda certificar-se que todo o código SQL é executável no ambiente de trabalho.
`web/`	Pasta com os arquivos Python e HTML da aplicação.

2. **Opção Alternativa:**

GG-relatorio.pdf	Deverá incluir uma folha de rosto com a indicação "**Projeto de BD - Parte 3**", o **número do grupo**, o **número e nome** dos alunos que o constituem, tal como **a percentagem relativa de contribuição** de cada aluno com o respectivo **esforço**

[5] https://github.com/bdist/db-workspace

(onde GG é o número do grupo)	**(horas)**, o **turno** a que o grupo pertence e o **nome do docente de laboratório** responsável. O relatório deverá conter uma **explicação da arquitetura da aplicação web, incluindo um link para uma versão de trabalho**, e as **relações entre os vários ficheiros na pasta web/arquivos**. Deverá ainda incluir as respostas relativas aos **índices** (secção 6). ⚠ Os grupos devem garantir que a aplicação funciona online até depois da discussão.
populate.sql	Ficheiro para preencher a base de dados com os dados de teste.
ICs.sql	Ficheiro para criar as restrições de integridade (triggers e procedimentos armazenados).
queries.sql	Ficheiro com as consultas SQL.
view.sql	Ficheiro com as instruções para criar a view
analytics.sql	Ficheiro com as consultas de análise de dados OLAP
web/	Pasta com os arquivos Python e HTML da aplicação.

A entrega terá de ser feita através do Fénix até às 23h59 da data de entrega.

IMPORTANTE: Serão aplicadas penalizações aos grupos que não cumprirem o formato de submissão. Os elementos de avaliação que não forem encontrados de acordo com as instruções acima prescritas **não serão levados em consideração para a classificação**. Não serão aceites submissões fora do prazo.

Anexo A - Esquema SQL

```
DROP TABLE IF EXISTS customer CASCADE;
DROP TABLE IF EXISTS orders CASCADE;
DROP TABLE IF EXISTS pay CASCADE;
DROP TABLE IF EXISTS employee CASCADE;
DROP TABLE IF EXISTS process CASCADE;
DROP TABLE IF EXISTS department CASCADE;
DROP TABLE IF EXISTS workplace CASCADE;
DROP TABLE IF EXISTS works CASCADE;
DROP TABLE IF EXISTS office CASCADE;
DROP TABLE IF EXISTS warehouse CASCADE;
DROP TABLE IF EXISTS product CASCADE;
DROP TABLE IF EXISTS contains CASCADE;
DROP TABLE IF EXISTS supplier CASCADE;
DROP TABLE IF EXISTS delivery CASCADE;

CREATE TABLE customer(
      cust_no INTEGER PRIMARY KEY,
      name VARCHAR(80) NOT NULL,
      email VARCHAR(254) UNIQUE NOT NULL,
      phone VARCHAR(15),
      address VARCHAR(255)
);

CREATE TABLE orders(
      order_no INTEGER PRIMARY KEY,
      cust_no INTEGER NOT NULL REFERENCES customer,
      date DATE NOT NULL
      --order_no must exist in contains
);

CREATE TABLE pay(
      order_no INTEGER PRIMARY KEY REFERENCES orders,
      cust_no INTEGER NOT NULL REFERENCES customer,
);

CREATE TABLE employee(
      ssn VARCHAR(20) PRIMARY KEY,
      TIN VARCHAR(20) UNIQUE NOT NULL,
      bdate DATE,
      name VARCHAR NOT NULL
      --age must be >=18
);

CREATE TABLE process(
```

```
      ssn VARCHAR(20) REFERENCES employee,
      order_no INTEGER REFERENCES orders,
      PRIMARY KEY (ssn, order_no)
);

CREATE TABLE department(
      name VARCHAR PRIMARY KEY
);

CREATE TABLE workplace(
      address VARCHAR PRIMARY KEY,
      lat NUMERIC(8, 6) NOT NULL,
      long NUMERIC(9, 6) NOT NULL,
      UNIQUE(lat, long)
      --address must be in warehouse or office but not both
);

CREATE TABLE office(
      address VARCHAR(255) PRIMARY KEY REFERENCES workplace
);

CREATE TABLE warehouse(
      address VARCHAR(255) PRIMARY KEY REFERENCES workplace
);

CREATE TABLE works(
      ssn VARCHAR(20) REFERENCES employee,
      name VARCHAR(200) REFERENCES department,
      address VARCHAR(255) REFERENCES workplace,
      PRIMARY KEY (ssn, name, address)
);

CREATE TABLE product(
      SKU VARCHAR(25) PRIMARY KEY,
      name VARCHAR(200) NOT NULL,
      description VARCHAR,
      price NUMERIC(10, 2) NOT NULL,
      ean NUMERIC(13) UNIQUE
);

CREATE TABLE contains(
      order_no INTEGER REFERENCES orders,
      SKU VARCHAR(25) REFERENCES product,
      qty INTEGER,
      PRIMARY KEY (order_no, SKU)
);
```

```
CREATE TABLE supplier(
      TIN VARCHAR(20) PRIMARY KEY,
      name VARCHAR(200),
      address VARCHAR(255),
      SKU VARCHAR(25) REFERENCES product,
      date DATE
);

CREATE TABLE delivery(
      address VARCHAR(255) REFERENCES warehouse,
      TIN VARCHAR(20) REFERENCES supplier,
      PRIMARY KEY (address, TIN)
);
```