Bases de Dados

2021/2022

Projeto - Enunciado 3

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

1. Base de Dados

Esquema

Utilizando a linguagem SQL (DDL), apresente as instruções para criar o esquema de Base de Dados correspondente ao esquema relacional apresentado no **Anexo A**. Certifique-se de que os tipos de dados e tamanhos de campo selecionados são os mais apropriados. Adicionalmente, devem ser especificadas restrições em cada campo, linha e tabela recorrendo às instruções **NOT NULL**, **CHECK**, **PRIMARY KEY**, **UNIQUE** e **FOREIGN KEY** conforme seja apropriado. Não use caracteres acentuados nem cedilhas.

Carregamento

Utilizando o esquema desenvolvido anteriormente como ponto de partida, crie as instruções para o preenchimento da base de dados de forma consistente. Deve ter em consideração que os registos devem garantir que todas as consultas SQL, apresentadas mais adiante, tenham um **resultado não vazio**. 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).

2. Restrições de Integridade

Apresente o código para implementar as restrições de integridade com as extensões procedimentais SQL (Stored Procedures e Triggers) no esquema de base de dados definido no ponto anterior:

(RI-1) Uma Categoria não pode estar contida em si própria

(RI-4) O número de unidades repostas num Evento de Reposição não pode exceder o número de unidades especificado no Planograma

(RI-5) Um Produto só pode ser reposto numa Prateleira que apresente (pelo menos) uma das Categorias desse produto

Quaisquer restrições de integridade **definidas** sem recorrer a extensões procedimentais (Stored Procedures e Triggers), devem ser implementados usando outros mecanismos, se apropriado. No entanto, mecanismos tais como como **ON DELETE CASCADE** e **ON UPDATE CASCADE** não são permitidos.

3. SQL

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

- Qual o nome do retalhista (ou retalhistas) responsáveis pela reposição do maior número de categorias?
- Qual o nome do ou dos retalhistas que são responsáveis por todas as categorias simples?
- Quais os produtos (ean) que nunca foram repostos?
- Quais os produtos (ean) que foram repostos sempre pelo mesmo retalhista?

4. Vistas

Cada evento de reposição representa a saída de stock de uma quantidade de produto sob a forma de vendas, desperdícios e roubos. Supondo que os desperdícios e roubos são inexistentes (todas as reposições de produto são causadas por vendas), crie uma vista que resuma as informações mais importantes sobre as vendas, combinando informações de diferentes tabelas do modelo. A vista deve ter o seguinte esquema:

Vendas (ean, cat, ano, trimestre, dia_mes, dia_semana, distrito, concelho, unidades)

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

- *unidades:* corresponde ao atributo com o mesmo nome da relação *evento_reposicao*
- *ean* e *cat*: correspondem às chaves primárias das relações produto e categoria, respectivamente
- *distrito e concelho*: correspondem aos atributos com o mesmo nome de *ponto_de_retalho*
- *ano, trimestre, mes* e *dia_semana:* atributos derivados do atributo *instante*

Pode utilizar a função extract()[2] do POSTGRES para obter anos, trimestres, dias da semana, etc a partir de *"timesptamps"*.

[1] Não pode usar instruções SQL que não fazem parte do SQL standard (como por exemplo a `instrução LIMIT`).
[2] https://www.postgresqltutorial.com/postgresql-date-functions/postgresql-extract/

5. Desenvolvimento da Aplicação

Crie um protótipo de aplicação web em scripts Python CGI e páginas HTML que permita:
a) Inserir e remover categorias e sub-categorias;
b) Inserir e remover um retalhista, com todos os seus produtos, garantindo que esta operação seja atómica;
c) Listar todos os eventos de reposição de uma IVM, apresentando o número de unidades repostas por categoria de produto;
d) Listar todas as sub-categorias de uma super-categoria, a todos os níveis de profundidade.

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. Pode-se utilizar CSS, embora não seja objeto de avaliação.

6. Consultas OLAP

Usando a vista desenvolvida para a Questão 4, escreva duas consultas SQL que permitam analisar o número total de artigos vendidos:

1. num dado período (i.e. entre duas datas), por dia da semana, por concelho e no total
2. num dado distrito (i.e. “Lisboa”), por concelho, categoria, dia da semana e no total

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

7. 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.

7.1 -

```
SELECT DISTINCT R.nome
FROM retalhista R, responsavel_por P
WHERE R.tin = P.tin    and P. nome_cat = 'Frutos'
```

7.2 -

```
SELECT T.nome, count(T.ean)
FROM produto P,  tem_categoria T
WHERE p.cat = T.nome and P.desc like ‘A%’
GROUP BY T.nome
```

Relatório

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**
SQL	5
Restrições de Integridade	2
Aplicação	6
Análise de dados	4
Índices	3

O relatório deve começar com uma folha de rosto com o título “**Dados Projeto de**”, o **name e nº, de alunosa contribuição de cada membro em porcentagem relativa, juntamente com** a **esforço (em horas)** que cada membro colocou no projeto, o **número do grupo**, o grupo **turno** e o nome do professor do laboratório. Além da folha de rosto, o relatório deve ter, no máximo, **8 páginas**.

Entrega

A submissão em Fénix deve ser um **zip** com a seguinte estrutura:

GG-relatorio.pdf (onde GG é o número do grupo)	O relatório em **pdf** onde **GG** é o número do grupo, contendo uma **explicação da arquitetura da aplicação web com um link para uma versão de trabalho**, as **relações entre os vários arquivos** e os **índices**. Você não deve incluir as instruções usadas para preencher o banco de dados. O relatório não precisa incluir o componente OLAP. ⚠ Os grupos devem garantir que o aplicativo funcione online até depois da discussão.
queries.sql	Ficheiro com as consultas SQL.
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).
view.sql	Ficheiro com as instruções para criar a view
analytics.sql	Ficheiro com as consultas de análise de dados
web/	Pasta com os arquivos Python e HTML.

O projeto deve ser entregue em ZIP formatado com o nome entregay-03-GG.zip[3] (onde **GG** é o número do grupo), através Fénix até às 23h59 da data de entrega.

Nota: penalidadesse aplicam a gruposque não sigam as instruções de entrega. Os elementos de avaliação que não forem encontrados de acordo com as instruções acima prescritas **não serão levados em consideração para a classificação**.

[3] ⚠ O formato do arquivo deve ser exclusivamente ZIP ou GZ. Outros formatos de arquivo (como RAR) não serão aceitos.

Anexo A - Esquema Relacional

categoria(<u>nome</u>)

- **RI-RE1**: O valor do atributo nome de qualquer registo da relação categoria tem de existir em na relação categoria_simples ou na relação super_categoria

categoria_simples(<u>nome</u>)

- nome: FK(categoria)
  RI-RE2: O valor do aributo nome de qualquer registo de categoria_simples não pode existir em super_categoria

super_categoria(<u>nome</u>)

- nome: FK(categoria)
- **RI-RE3**: O valor do atributo nome de qualquer registo tem de existir no atributo super_categoria da relação constituída

tem_outra(super_categoria, <u>categoria</u>)

- super_categoria: FK(super_categoria)
- categoria: FK(categoria)
- **RI-RE4** não podem existir valores repetidos dos atributos super_categoria e categoria numa sequência de registos relacionados pela FK categoria
- **RI-RE5**: Para qualquer registo desta relação, verifica-se que os atributos super_categoria e categoria são distintos.

produto(<u>ean</u>, cat, descr)

- cat: FK(categoria)
- **RI-RE6**: O valor do atributo ean existente em qualquer registo da relação produto tem de existir também no atributo ean da relação tem_categoria

tem_categoria(ean, nome)

- ean: FK(produto)
- nome: FK(categoria)

IVM(<u>num_serie, fabricante</u>)

ponto_de_retalho(<u>nome</u>, distrito, concelho<u>)</u>

instalada_em(<u>num_serie, fabricante</u>, local)

- num_serie, fabricante: FK(IVM)
- local: FK(ponto_de_retalho)

prateleira(<u>nro, num_serie, fabricante</u>, altura, nome)

- num_serie, fabricante: FK(IVM)
- nome: FK(categoria)

planograma(ean, nro, num_serie, fabricante, faces, unidades, loc)

- ean: FK(produto)
- nro, num_serie, fabricante: FK(prateleira)

retalhista(tin, name)

- **RI-RE7**: unique(name)

responsavel_por(nome_cat, tin, num_serie, fabricante)

- num_serie, fabricante: FK(IVM)
- tin: FK(retalhista)
- nome_cat: FK(categoria)

evento_reposicao(ean, nro, num_serie, fabricante, instante, unidades, tin)

- ean, nro, num_serie, fabricante: FK(planograma)
- tin: FK(retalhista)
- **RI-RE7**8: o valor do atributo unidades de um registo X desta relação não pode exceder o valor do atributo unidades do registo da relação planograma referido pelo registo X.