U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

Bases de Dados, MIEIC 2012/13

João Mendes Moreira, Carla Teixeira Lopes, Sérgio Nunes, Tiago Boldt

Exame de 18 de Junho de 2013 – Parte prática

A duração TOTAL do exame (incluindo TUDO!) é de 2h30m

As perguntas 1 e 2 devem ser respondidas em folha de exame

1. **(5 valores)** Construa um modelo conceptual de dados em UML para armazenar informação de uma garagem automóvel. Indique todas as restrições que possam ser úteis para a construção da base de dados.

   A garagem pretende manter informação sobre os estacionamentos, em particular os detalhes sobre a hora de entrada e saída, bem como o custo total de cada utilização. Os estacionamentos podem ser de três tipos: normal, via verde e assinatura. No caso de um estacionamento usando a via verde é necessário armazenar informação sobre o identificador. No caso dos estacionamentos associados a assinaturas é necessário relacionar o estacionamento com a assinatura respetiva.

   Relativamente a cada assinatura interessa saber qual o cliente e veículo associado, bem como detalhes sobre a data de início, duração e valor total. As assinaturas têm um lugar associado e são pagas mensalmente. Estas mensalidades ficam em pagamento até haver uma indicação de que o pagamento foi realizado. O sistema deve permitir armazenar detalhes sobre cada cliente e sobre cada veículo, nomeadamente o modelo e a marca (caso existam), bem como o tipo de veículo (motociclo, comercial, etc).

2. **(2 valores)** Considere a relação R(A,B,C,D,E) com as dependências funcionais F={A->B, AC->D, BD->C, B->CE, E->A}. Normalize essa relação na terceira Forma Normal.

A pergunta 3 deve ser respondida através do SIGEX
Notas MUITO Importantes para submissão das respostas:

i. A resposta a cada alínea deve ser guardada num ficheiro SEPARADO com o nome no seguinte formato: *<nome_da_alinea>.sql* **EM MAIÚSCULAS!**
   ***Exemplo**: alínea 4a) deverá ser respondida no ficheiro "4A.sql".*
ii. Devem finalizar TODAS as INSTRUÇÕES SQL que desenvolverem em resposta às questões com ponto e virgula (;);
   Exemplo: "SELECT * FROM TABELA;"
iii. Todas as alíneas devem ser submetidas via SIGEX integradas num único ficheiro .Zip
iv. **O incumprimento destas notas poderá incorrer na anulação das alíneas em causa ou do próprio teste**.

3. O CICA pretende desenvolver um sistema de gestão de bugs para as suas aplicações e servidores onde estas estão instaladas. O sistema irá guardar informações relativas às aplicações existentes, servidores, aplicações instaladas em servidores, pessoas na equipa e Bugs existentes. Um bug mantém uma descrição do mesmo, bem como informação relativa à sua prioridade (entre 1 e 5, sendo 1 o mais prioritário), o estado ("aberto" ou "fechado") e se o mesmo apresenta uma vulnerabilidade ("sim" ou "não"). Bugs que sejam vulnerabilidades (campo "vulnerabilidade" da tabela Bug igual a "sim") introduzem falhas de segurança nos servidores onde as respetivas aplicações estão instaladas. Sempre que isto acontece, o campo "vulnerável" do servidor tem o valor "sim".

   Modelo Relacional
   Aplicacao(idAplicacao, nome, descricao)
   Servidor(idServidor, idResponsavel -> Pessoa, vulneravel, hostname)
   AplicacaoServidor(idAplicacao -> Aplicacao, Servidor -> Servidor)
   Pessoa(idPessoa, nome, mail)
   Bug(idBug, idAplicacao -> Aplicacao, idResponsavel -> Pessoa, descricao, prioridade, estado, vulnerabilidade)

   As tabelas correspondentes podem ser criadas usando o script criação.txt.

   a) **(1 valor)** Liste o hostname e nome do responsável para cada servidor vulnerável

hostname	nome
porto	Joao Almeida
lisboa	Joao Almeida
lagos	Ana Goncalves
alu1	Mario Manuel
cica	Ana Goncalves

   b) **(1 valor)** Liste os hostnames dos servidores com aplicações vulneráveis a bugs, a descrição dos bugs que tornam o servidor vulnerável e o nome do responsável pelo servidor, ordenado por hostname.

hostname	descricao	nome
alu1	code injection na pagina de login	Mario Manuel
alu1	falha seguranca	Mario Manuel
alu2	falha seguranca	Joao Almeida
lagos	code injection na pagina de login	Ana Goncalves
lisboa	falha seguranca	Joao Almeida
porto	code injection na pagina de login	Joao Almeida

c) **(1.5 valor)** Listar os servidores com hostname começado por "alu", geridos pelo funcionário com email joao.almeida@cica.pt, com um ou mais bugs associados.

hostname
alu2

d) **(1.5 valor)** Listar o nome da aplicação com mais bugs associados.

nome
redmine

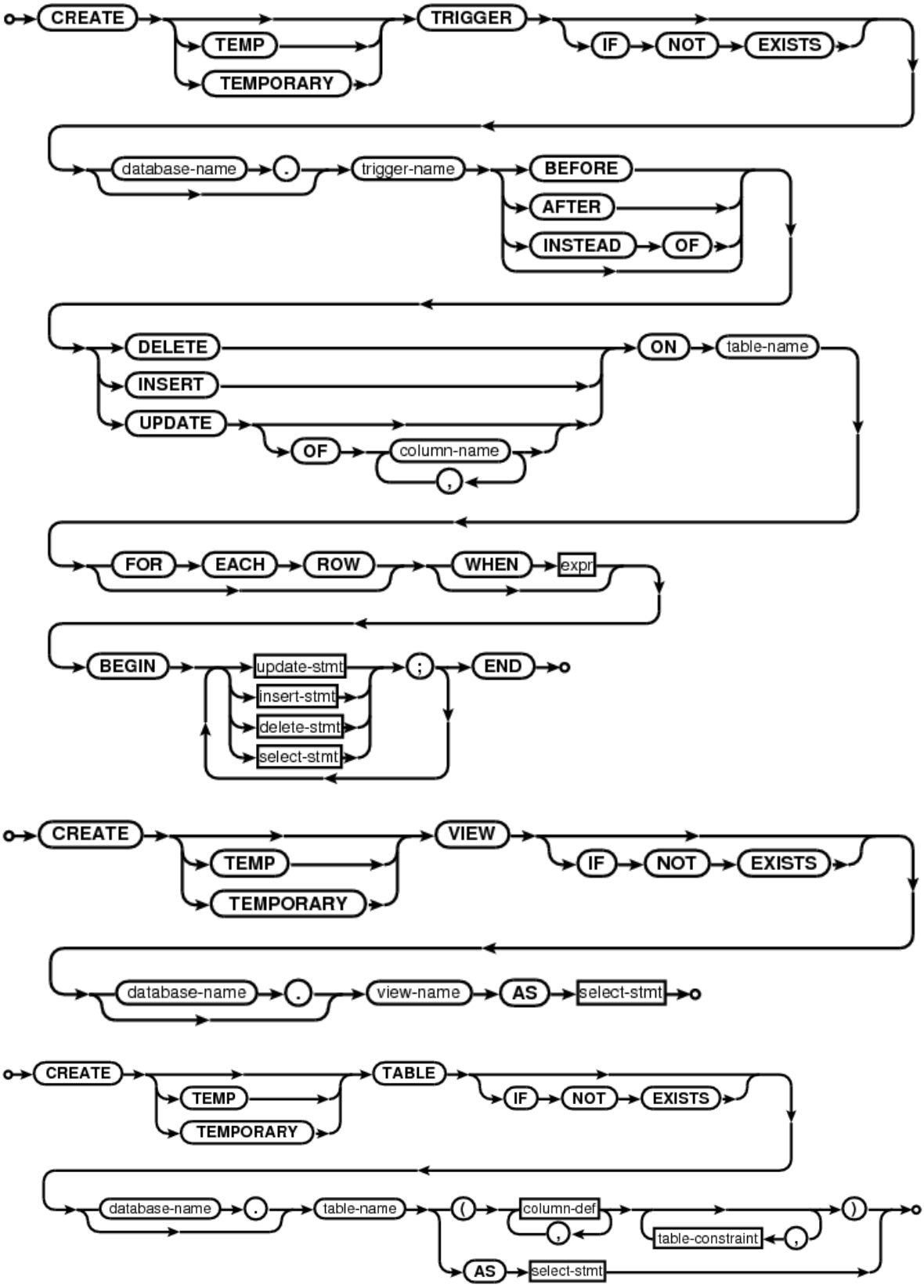
e) **(1.5 valor)** Crie um trigger que ao adicionar uma aplicação a um servidor, o marque como vulnerável caso essa aplicação tenha bugs que introduzam vulnerabilidades.

f) **(1.5 valor)** Crie um trigger que na inserção de um bug que represente uma vulnerabilidade marque como vulneráveis todos os servidores que usam essa mesma aplicação (atualize o campo "vulnerável" no servidor) e garanta que o valor da prioridade do bug é 1.

Sintaxe básica de SQL

INSTRUÇÃO SELECT	OPERADORES	JOIN, UNION, INTERSECT e EXCEPT
SELECT * FROM tabela;	SELECT * FROM tabela WHERE col1 [NOT] BETWEEN valor1 AND valor2;	SELECT * FROM tabela1, tabela2 WHERE condições;
SELECT col1,col2 FROM tabela;	SELECT * FROM tabela WHERE col1 [NOT] IN (valor1,valor2,…) ;	SELECT * FROM tabela1 INNER JOIN tabela2 ON condições;
SELECT col1,col2 FROM tabela WHERE condições;	SELECT * FROM tabela WHERE col1 [NOT] IN (SELECT …);	SELECT * FROM tabela1 INNER JOIN tabela2 ON condições WHERE condições;
SELECT col1,col2 FROM tabela WHERE condições ORDER BY col1 ASC,col2 DESC;	SELECT * FROM tabela WHERE col1 > valor1 AND col1 < valor2;	SELECT * FROM tabela1 LEFT JOIN tabela2 ON condições;
SELECT DISTICT col1,col2 FROM tabela;	SELECT * FROM tabela WHERE col1 < valor1 OR col1 > valor2;	SELECT col1 from tabela1 UNION SELECT col2 from tabela2;
SELECT col1, agregação(col2) FROM tabela GROUP BY col1;	SELECT * FROM tabela WHERE col1 = valor1;	SELECT col1 FROM tabela1 INTERSECT SELECT col2 FROM tabela2;
SELECT col1, agregação(col2) FROM tabela GROUP BY col1 HAVING agregação(col2) > valor1;	SELECT * FROM tabela WHERE col1 = 'texto1';	SELECT col1 from tabela1 EXCEPT SELECT col2 from tabela2;
SELECT AVG(col1), SUM(col1), COUNT(*), MIN(col1), MAX(col1) FROM tabela;	SELECT * FROM tabela WHERE col1 <> valor1;	
INSTRUÇÃO INSERT	**INSTRUÇÃO DELETE**	**INSTRUÇÂO UPDATE**
INSERT INTO tabela [(atributos)] VALUES (valores);	DELETE FROM tabela WHERE condições;	UPDATE tabela SET atributo = valor, … WHERE condições;

Mais informação abaixo.


CREATE
TEMP
TEMPORARY
TRIGGER
IF
NOT
EXISTS
database-name
.
trigger-name
BEFORE
AFTER
INSTEAD
OF
DELETE
INSERT
UPDATE
OF
column-name
,
ON
table-name
FOR
EACH
ROW
WHEN
expr
BEGIN
update-stmt
insert-stmt
delete-stmt
select-stmt
;
END
CREATE
TEMP
TEMPORARY
VIEW
IF
NOT
EXISTS
database-name
.
view-name
AS
select-stmt
CREATE
TEMP
TEMPORARY
TABLE
IF
NOT
EXISTS
database-name
.
table-name
(
column-def
,
table-constraint
,
)
AS
select-stmt