Gatilho 1

O primeiro gatilho apresentado tem como funcionalidade verificar se, ao excluir um painelista, o painel ao qual ele estava associado deve ser removido ou não. Como pode haver mais de um painelista por painel, essa relação foi representada por meio de uma relação APRESENTA_PN. O gatilho apresentado verifica os registros que estavam relacionados a essa painel, e em seguida, verifica se esses registros estão associados a apenas o painelista que foi removido. Ele remove os painéis que apresentarem essa condição. Depois, é apagada a relação entre painel e painelista da tabela de relacionamento.

Evento
Remover na relação PAINELISTA.

Condição
Sempre.

Ação

- Remover registros da relação PAINEL, quando essa não possuir nenhum registro com outro painelista na relação APRESENTA_PN;
- Remover registros da relação APRESENTA_PN que estejam associados ao cpf correspondente ao painelista removido.

Solução para o Oracle

Para remover as relações da tabela PAINEL, foi usada uma cláusula WHERE com IN em um SELECT aninhado. O primeiro SELECT aninhado (o mais externo) retorna todos os painéis que aparecem no máximo uma vez na relação APRESENTA_PN, filtrando essa busca pelo campo id_painel, que é selecionado de um conjunto de valores obtido pelo uso de IN em outro SELECT aninhado. Este último retorna todos os valores de id_painel que estão associados ao painelista removido.

Depois da primeira remoção, é executado outro DELETE para remover todos os registros da relação APRESENTA_PN que estão associados ao painelista. Um detalhe importante é que a referência no Oracle para a linha removida que disparou o TRIGGER é feita por :old. A implementação do gatilho se da seguinte maneira:

```
create or replace TRIGGER REMOVER_PAINEL
  AFTER DELETE ON PAINELISTA
  FOR EACH ROW
  BEGIN
    DELETE FROM PAINEL
    WHERE id_painel IN (
      SELECT id_painel
      FROM APRESENTA_PN
      WHERE id_painel IN (
        SELECT id_painel
        FROM APRESENTA_PN
        WHERE APRESENTA_PN.cpf = :old.cpf
      )
      GROUP BY id_painel
      HAVING COUNT(*) <= 1
    );
```

```
  DELETE FROM APRESENTA_PN
  WHERE APRESENTA_PN.cpf = :old.cpf;
  END;
```

Solução em SQL

A estrutura do TRIGGER apresentado para o SQL padrão é a mesma da apresentada para o Oracle, com exceção à forma de referenciar a linha que foi removida, que aqui é feita por: REFERENCING old row as oldRow, e acessada por oldRow.campoDesejado. Também foi utilizado BEGIN ATOMIC visto que são feitas múltiplas operações. A implementação do gatilho se da seguinte maneira:

```
create TRIGGER
REMOVER_PAINEL
AFTER DELETE
ON PAINELISTA
REFERENCING old row as oldRow
FOR EACH ROW
BEGIN ATOMIC
  DELETE FROM PAINEL
  WHERE id_painel IN (
    SELECT id_painel
    FROM APRESENTA_PN
    WHERE id_painel IN (
      SELECT id_painel FROM APRESENTA_PN
      WHERE APRESENTA_PN.cpf = oldRow.cpf
    )
    GROUP BY id_painel
    HAVING COUNT(*) <= 1
  )

  DELETE FROM APRESENTA_PN
  WHERE APRESENTA_PN.cpf = oldRow.cpf
END;
```

Previsões

Para usar o BEFORE no lugar de AFTER, não seria necessário alterar as consultas realizadas, porque elas não são feitas na mesma tabela em que o delete que causou o TRIGGER foi realizado.

Para utilizar o FOR EACH STATEMENT, seria necessário fazer uma consulta para saber todos os registros que foram afetados e usar o resultado obtido para filtrar a consulta de painéis afetados e para filtrar a remoção dos registro da relação APRESENTA_PN.

Gatilho 2

O segundo trigger apresentado tem como função alterar o cadastro de um jornalista, quando um veículo de comunicação for removido. Quando um veículo é cadastrado, é entendido que ele possui um relacionamento de parceria com o evento. Quando um jornalista é cadastrado, seus ingressos já foram comprados pelo veículo associado. Dessa forma, ao remover um veículo do sistema, todos os dados relacionados a esse devem ser apagados, inclusive os jornalistas e as matérias. Porém, como os ingressos do jornalista já foram comprados, não é interessante descartar todas as informações sobre o proprietário desse ingresso. Para solucionar esse problema, quando o veículo for removido, os dados dos jornalistas associados são apagados e ele é cadastrado como um participante pagante comum.

Evento
Remover na relação VEICULO_COMUNICACAO.

Condição
Sempre.

Ação

- Inserir na relação PAGANTE todos os jornalistas associados ao veículo de comunicação removido;
- Remover esses mesmos jornalistas da relação ISENTO;
- Remover esses mesmos jornalistas da relação JORNALISTA;
- Remover os relacionamentos associados ao veículo de comunicação da relação TRABALHA_EM.

Solução para o Oracle

Para inserir em PAGANTE, são utilizados os valores de cpf associados ao veículo, obtidos pela relação TRABALHA_EM, e um valor arbitrário definido aqui como 350.00, que representa o valor em reais(R$) de ingresso para todos os dias do evento.

As remoções citadas na descrição são todas filtradas pela mesma cláusula WHERE que relaciona o cpf ao veículo removido, uma vez que esse campo está presente em todas as relações afetadas, com exceção da última que é feita diretamente pelo codigo_vc do veículo removido. A implementação do gatilho se da seguinte maneira:

```
create or replace TRIGGER
REMOVER_VEICULO_JORNALISTA
AFTER DELETE
ON VEICULO_COMUNICACAO
FOR EACH ROW
BEGIN
  INSERT INTO PAGANTE (cpf, valor_insc)

  SELECT cpf, 350.00
  FROM TRABALHA_EM
  WHERE TRABALHA_EM.codigo_vc = :old.codigo_vc;

  DELETE FROM MATERIA
```

```
  WHERE MATERIA.cpf IN (
    SELECT cpf FROM TRABALHA_EM
    WHERE TRABALHA_EM.codigo_vc = :old.codigo_vc
  );

  DELETE FROM ISENTO
  WHERE ISENTO.cpf IN (
    SELECT cpf FROM TRABALHA_EM
    WHERE TRABALHA_EM.codigo_vc = :old.codigo_vc
  );

  DELETE FROM JORNALISTA
  WHERE JORNALISTA.cpf IN (
    SELECT cpf FROM TRABALHA_EM
    WHERE TRABALHA_EM.codigo_vc = :old.codigo_vc
  );

  DELETE FROM TRABALHA_EM
  WHERE TRABALHA_EM.codigo_vc = :old.codigo_vc;
END;
```

Solução em SQL

A estrutura do TRIGGER apresentado para o SQL padrão é a mesma da apresentada para o Oracle, com exceção à forma de referenciar a linha que foi removida, que aqui é feita por: REFERENCING old row as oldRow, e acessada por oldRow.campoDesejado. Também foi utilizado BEGIN ATOMIC visto que são feitas múltiplas operações. A implementação do gatilho se da seguinte maneira:

```
create TRIGGER
REMOVER_VEICULO_JORNALISTA
AFTER DELETE
ON VEICULO_COMUNICACAO
REFERENCING old row as oldRow
FOR EACH ROW
BEGIN ATOMIC
  INSERT INTO PAGANTE (cpf, valor_insc)
  SELECT cpf, 350.00
  FROM TRABALHA_EM
  WHERE TRABALHA_EM.codigo_vc = oldRow.codigo_vc

  DELETE FROM ISENTO
  WHERE ISENTO.cpf IN (
    SELECT cpf FROM TRABALHA_EM
    WHERE TRABALHA_EM.codigo_vc = oldRow.codigo_vc
  )

  DELETE FROM JORNALISTA
  WHERE JORNALISTA.cpf IN (
    SELECT cpf FROM TRABALHA_EM
    WHERE TRABALHA_EM.codigo_vc = oldRow.codigo_vc
```

```
    )

    DELETE FROM MATERIA
    WHERE MATERIA.cpf IN (
      SELECT cpf FROM TRABALHA_EM
      WHERE TRABALHA_EM.codigo_vc = oldRow.codigo_vc
    )

    DELETE FROM TRABALHA_EM
    WHERE TRABALHA_EM.codigo_vc = oldRow.codigo_vc
  END;
```

Previsões

Para usar o BEFORE no lugar de AFTER, não seria necessário alterar as consultas realizadas, porque elas não são feitas na mesma tabela em que o delete que causou o TRIGGER foi realizado.

Para utilizar o FOR EACH STATEMENT, seria necessário fazer uma consulta para saber todos os registros que foram afetados e usar o resultado obtido para filtrar cada uma das cláusulas DELETE efetuadas e para o INSERT também, de forma que elas fossem acionadas apenas para os registros que de fato foram afetados pela remoção.

Casos de teste

Teste A (Gatilho 2)
Excluir um jornalista que já estava cadastrado como pagante. Esse caso de teste tem como objetivo expor uma falha na solução apresentada, que não leva em conta um cenário atípico do problema. É possível que, mesmo que um participante esteja credenciado como jornalista, o veículo de comunicação empregador não tenha garantido acesso a todos os dias do evento, e ele tenha comprado ingressos para os dias restantes.

Dessa forma, caso esse jornalista seja desvinculado pela remoção do veículo de comunicação associado, o gatilho tentará inserir o registro do participante na tabela PAGANTE, o que irá resultar em um erro de restrição de chave primária exclusiva, porque o cpf em questão já existe na relação alvo. Uma possível solução para esse problema seria verificar a existência do participante nessa relação, e se esta for confirmada, apenas atualizar o valor de inscrição para somar os dias que haviam sido pagos pelo veículo de comunicação.

Inicialização das tabelas*
* considerando o banco já populado com o script da Entrega 1

INSERT INTO PARTICIPANTE (cpf, nome, data_nasc, email, telefone, genero, ocupacao, categoria_ins, valor_barraca, PAGANTE, ISENTO, id_endereco, id_caravana) VALUES ('12345678912', 'Carla Vieira', TO_DATE('24/04/1998', 'dd/mm/yyyy'), 'viecarla@email.com', '15929764991', 'F', 'Jornalista', 'isento com barraca', 150, 0, 2, 2, 1);

INSERT INTO PAGANTE (cpf, valor_insc) VALUES ('12345678912', 450.00);

INSERT INTO ISENTO (cpf, PALESTRANTE, PAINELISTA, JORNALISTA, EXPOSITOR) VALUES ('12345678912', 0, 0, 1,0);

INSERT INTO JORNALISTA (cpf, credencial, especialidade) VALUES ('12345678912', 1, 'Ciencia');

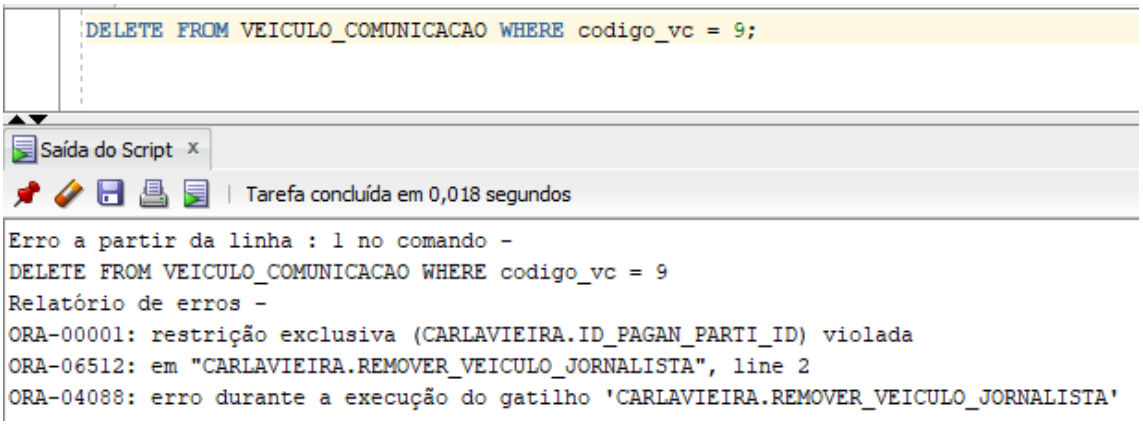
INSERT INTO MATERIA (id_materia, titulo, assunto, data, cpf) VALUES (1, 'Como fazer seus miolos explodirem', 'Neurologia aplicada', TO_DATE('12/01/2018', 'dd/mm/yyyy'), '12345678912');

INSERT INTO VEICULO_COMUNICACAO (codigo_vc, nome_vc, meio_comunicacao) VALUES (9, 'Canal Tech', 'youtube');

INSERT INTO TRABALHA_EM (cpf, codigo_vc) VALUES ('12345678912', 9);

Ação de teste
DELETE FROM VEICULO_COMUNICACAO WHERE codigo_vc = 9;

Resultado



Teste B (Gatilho 1)
Excluir um painelista. Esse caso de teste tem como objetivo exibir a execução do gatilho, de forma que, após a exclusão de um painelista, os registros na tabela de APRESENTA_PN que possuem como apresentador o painelista excluído, devem também ser excluídos. No entanto, o registro do painelista na tabela de participantes não foi excluído.

Inicialização das tabelas*
* considerando o banco já populado com o script da Entrega 1

Nesse caso, não será necessário inserção de dados para realização do teste.

Ações de teste
DELETE FROM PAINELISTA WHERE cpf = '55555555555';

Resultado

```
DELETE FROM PAINELISTA WHERE cpf = '55555555555';

select * from apresenta_pn;
```

Saída do Script | Resultado da Consulta

SQL | Todas as Linhas Extraídas: 8 em 0,001 segundos

	CPF	ID_PAINEL
1	77777777777	1
2	66666666666	2
3	77777777777	2
4	77777777777	3
5	66666666666	4
6	77777777777	4
7	66666666666	5
8	77777777777	5

```
select * from participante;
```

Saída do Script | Resultado da Consulta

SQL | Todas as Linhas Extraídas: 15 em 0,006 segundos

	CPF	NOME	DATA_NASC	EMAIL
1	11111111111	Maria Maravilha	12/03/88	maria@email.com
2	22222222222	João Joalheiro	19/01/75	joao@email.com
3	33333333333	Lucas Loiro	23/06/99	lucas@email.com
4	44444444444	Pedro Pedroso	17/07/74	pedro@email.com
5	55555555555	Bruna Bala	28/02/88	bruna@email.com
6	66666666666	Vitória Valente	04/04/86	vitoria@email.com

Teste C (Gatilho 2)

O caso de teste sugerido tem como intuito mostrar uma deficiência do gatilho em tratar um caso atípico do sistema. Como foi informado na descrição do problema, um jornalista pode estar associado a mais de um veículo de comunicação. Da forma que o gatilho foi implementado, ele remove todos os jornalistas associados ao veículo excluído sem considerar que eles possam estar associados a outro veículo. Para resolver esse problema, teriam que ser verificadas as possíveis demais ocorrências do jornalista na relação TRABALHA_EM.

Inicialização das tabelas*

* considerando o banco já populado com o script da Entrega 1

INSERT INTO JORNALISTA (cpf, credencial, especialidade) VALUES ('33333333333', 2, 'Tecnologia');
INSERT INTO VEICULO_COMUNICACAO (codigo_vc, nome_vc, meio_comunicacao) VALUES (9, 'Canal Tech', 'youtube');

INSERT INTO TRABALHA_EM (cpf, codigo_vc) VALUES ('33333333333', 1);
INSERT INTO TRABALHA_EM (cpf, codigo_vc) VALUES ('33333333333', 9);

Ações de teste

DELETE FROM VEICULO_COMUNICACAO WHERE codigo_vc = 9;

Resultado