Universidade de São Paulo
Escola de Artes, Ciências e Humanidades
Graduação em Sistemas de Informação

Adriano Barbieri	8921162
Douglas Mizuma	8920964
Laura Castro Vieira	8598822
Virgílio Fernandes Junior	7640870

Artefato 2A
Laboratório de Banco de Dados

Professor Luciano Araújo

São Paulo, SP

2016

Projeto de Laboratório de Bancos de Dados - Regras de negócio e visões

Regras de negócio - afirmações

1. Assentos e limite da aeronave

A primeira regra de negócio define que assentos não podem ser cadastrados além da capacidade de uma aeronave, que é definida no campo *Quantidade de Assentos* da tabela *Aeronave*.

SQL Padrão:

```
1  CREATE ASSERTION aeronave_assentos_check CHECK
2  (NOT EXIST (
3      SELECT * FROM (SELECT V.qtd_assentos, V.aer_codigo FROM aeronave V) B,
4      (SELECT aer_codigo, COUNT(ass_codigo) AS assentos FROM assento  GROUP BY aer_codigo) C
5      WHERE B.aer_codigo = C.aer_codigo and C.assento>B.qtd_assentos
6      )
7  )
```

Código:

```
1  DELIMITER $$
2  CREATE TRIGGER aeronave_assentos_check_insert
3  BEFORE INSERT ON assento
4  FOR EACH ROW
5  BEGIN
6      IF (SELECT COUNT(*) FROM (SELECT V.qtd_assentos FROM aeronave V
       WHERE V.aer_codigo = new.aer_codigo) B, (SELECT COUNT(*) AS
       assentos FROM assento A WHERE A.aer_codigo = new.aer_codigo) C
       WHERE B.qtd_assentos <= C.assentos) > 0
7      THEN
8          SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'aeronave nao pode
           ter mais assentos';
9      END IF;
10 END$$
11 DELIMITER ;
```

Transcrição:

O código acima descreve a afirmação através de um Delimitador $$ do MySQL. A regra de negócio é nomeada "aeronave_assentos_check_insert" e ocorre antes de cada criação de registro na tabela *Assento*. Os parâmetros utilizados para a verificação são:

- qtd_assentos: quantidade de assentos
- aer_codigo: código da aeronave

"BEFORE INSERT ON" é o trecho do código SQL que determina a afirmação. O MySQL não suporta afirmações, então foi criado um gatilho antes da inserção que garantirá que a condição seja cumprida.

A linha 6 do código verifica o campo *Quantidade de Assentos* do avião, nomeando-o B, e verifica a quantidade de registros pertencentes aquele avião que estão na tabela *Assento,* nomeando-a em C. Caso a tabela Assentos já possua a quantidade de registros determinada pela

Quantidade de Assentos, a linha 8 do código envia um mensagem de que aquele assento não poderá ser inserido.

Casos de teste:

- Teste 1: Inserção de uma companhia, de uma aeronave e verificação da quantidade de assentos, seguida da tentativa de inserção de um assento de uma aeronave cujos todos assentos já foram inseridos.

```
INSERT INTO companhia (registro, nome) VALUES ('1111', 'Companhia');
/* Affected rows: 1  Found rows: 0  Warnings: 0  Duration for 1 query:
0,031 sec. */
INSERT INTO aeronave (modelo, qtd_assentos, com_companhia) VALUES (123, 2,
1);
/* Affected rows: 1  Found rows: 0  Warnings: 0  Duration for 1 query:
0,063 sec. */
SELECT B.aer_codigo, COUNT(*) assentos, B.qtd_assentos FROM assento A,
aeronave B WHERE A.aer_aeronave = 1 AND B.aer_codigo = 1;
/* Affected rows: 0  Found rows: 1  Warnings: 0  Duration for 1 query:
0,000 sec. */
INSERT INTO assento(classe, aer_aeronave, numero) VALUES (1, 1, 3);
/* SQL Error (1644): aeronave nao pode ter mais assentos */
/* Affected rows: 0  Found rows: 0  Warnings: 0  Duration for 0 of 1 query:
0,000 sec. */
```

- Teste 2: Exclusão de um assento de uma aeronave, demonstração da exclusão e inserção de novo assento bem sucedida.

```
DELETE FROM assento WHERE aer_aeronave = 1 AND numero = 1;
/* Affected rows: 1  Found rows: 0  Warnings: 0  Duration for 1 query:
0,047 sec. */
SELECT COUNT(*) assentos, B.qtd_assentos FROM assento A, aeronave B WHERE
A.aer_aeronave = 1 AND B.aer_codigo = 1;
/* Affected rows: 0  Found rows: 1  Warnings: 0  Duration for 1 query:
0,000 sec. */
INSERT INTO assento(classe, aer_aeronave, numero) VALUES (1, 1, 1);
/* Affected rows: 1  Found rows: 0  Warnings: 0  Duration for 1 query:
0,047 sec. */
SELECT COUNT(*) assentos, B.qtd_assentos FROM assento A, aeronave B WHERE
A.aer_aeronave = 1 AND B.aer_codigo = 1;
/* Affected rows: 0  Found rows: 1  Warnings: 0  Duration for 1 query:
0,000 sec. */
```

2. Aeronave não pode estar em dois voos simultaneamente

A segunda afirmação é necessária para impedir que uma aeronave seja reservada para dois voos no mesmo horário. A aeronave pode ser utilizada em diversos voos, desde que não simultaneamente. Para que essa verificação seja feita, é preciso considerar o tempo total de cada voo, contando com seu fim, para que a aeronave fique restrita a ele durante toda sua duração.

SQL Padrão:

```
1  CREATE ASSERTION voo_mesmo_horario_insert CHECK
2  (NOT EXIST
3   (SELECT V.data_hora_ini, V.data_hora_fim from voo V1, voo V2
4    WHERE V1.aer_codigo=V2.aer_codigo
5    AND V1.voo_codigo<>V2.voo_codigo
6    AND V1.data_hora_ini<V2.data_hora_fim
7    AND V2.data_hora_ini<V1.data_hora_fim
8    )
9   )
10 )
```

Código:

```
1  DELIMITER $$
2  CREATE TRIGGER voo_mesmo_horario_insert
3  BEFORE INSERT ON voo
4  FOR EACH ROW
5  BEGIN
6      IF (SELECT COUNT(*) FROM (SELECT V.data_hora_ini, V.data_hora_fim
       FROM voo V WHERE V.aer_codigo = new.aer_codigo) C WHERE NOT (new.
       data_hora_ini > C.data_hora_fim OR C.data_hora_ini > new.
       data_hora_fim)) > 0
7      THEN
8          SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'aeronave nao pode
           estar em dois voos ao mesmo tempo';
9      END IF;
10 END$$
11 DELIMITER ;
```

Transcrição:

A segunda afirmação também foi especificada através de um delimitador $$ do MySQL. A regra de negócio é nomeada "voo_mesmo_horario" e dessa vez ocorre antes de cada criação de registro na tabela Voo. Os campos usados na query são:

- data_hora_ini: horário de início de um voo
- data_hora_fim: horário do fim de um voo
- aer_codigo: código da aeronave

Ao inserir um voo, a linha 6 do código verifica se há algum voo usando a mesma aeronave no horário pretendido. Para cada registro da tabela voo, o código verifica se o código da aeronave é o mesmo. Caso seja, ele verifica se os horários não são conflitantes verificando data de início e fim do voo. No caso de existir um voo conflitante, uma mensagem é exibida (linha 8) e não ocorre a inserção. Adicionalmente, o código pode ser estendido para casos de alteração de um Voo, além de inserção, através do uso de uma outra afirmação com BEFORE UPDATE no lugar de BEFORE INSERT.

Casos de teste:

- Teste 1: Inserção de um Voo bem sucedido com início às 14 horas e fim às 20 horas.

```
INSERT INTO voo(origem, destino, data_hora_ini, data_hora_fim, aer_codigo,
disponivel, preco) VALUES ('GRU', 'SSS', '2017-05-20-14:00:00',
'2017-05-20-20:00:00', 1, 1, 100.00);
```

- Teste 2: Inserção de um Voo usando o mesmo avião com início às 19 horas.

```
INSERT INTO voo(origem, destino, data_hora_ini, data_hora_fim, aer_codigo,
disponivel, preco) VALUES ('GRU', 'SSS', '2017-05-20-19:00:00',
'2017-05-20-23:00:00', 1, 1, 100.00);
/* SQL Error (1644): aeronave nao pode estar em dois voos ao mesmo tempo */
/* Affected rows: 0  Found rows: 0  Warnings: 0  Duration for 0 of 1 query:
0,000 sec. */
```

- Teste 3: Inserção de um Voo usando o mesmo avião com início às 13 horas, mas durando até às 15 horas.

```
INSERT INTO voo(origem, destino, data_hora_ini, data_hora_fim, aer_codigo,
disponivel, preco) VALUES ('GRU', 'SSS', '2017-05-20-13:00:00',
'2017-05-20-15:00:00', 1, 1, 100.00);
/* SQL Error (1644): aeronave nao pode estar em dois voos ao mesmo tempo */
/* Affected rows: 0  Found rows: 0  Warnings: 0  Duration for 0 of 1 query:
0,000 sec. */
```