Asserção 1

Como informado na descrição do projeto, apresentada na primeira entrega, cada barraca disponibilizada pelo evento pode ter no máximo um participante associado.Na definição do modelo, para representar essa associação, foi adicionado o campo cpf na tabela BARRACA. A asserção sugerida tem como objetivo garantir que exista apenas uma barraca por cpf.

Solução em SQL

A solução em SQL padrão possui o formato apresentado em aula, "não existe x, tal que não P(x)". Seguindo esse princípio, a consulta interna tem como objetivo retornar a quantidade de barracas por cpf,
e que tenham essa quantidade avaliada em > 1, para evitar um SELECT adicional em todos os resultados, para filtrar essa condição. Por fim, é verificado se não existe nenhum resultado que se encaixe nessa condição. A implementação da asserção se da seguinte maneira:

```
create assertion restricao_participante_barraca check
(not exists (
    SELECT COUNT(*) FROM BARRACA GROUP BY cpf
    HAVING COUNT(*) > 1)
);
```

Solução para o Oracle

Como informado em aula, e confirmado pelos teste, o Oracle não possui suporte para asserções. A maneira encontrada para realizar o mesmo controle foi utilizando uma CONSTRAINT UNIQUE sobre o campo cpf, que implica em que um valor só pode aparecer uma única vez na coluna especificada. A sintaxe apresentada é estruturada para ser executada após a criação da tabela BARRACA. Uma observação importante é que essa tratativa já havia sido implementada na primeira entrega, então ao tentar executar essa instrução sobre o banco já construído, será retornado um erro de CONSTRAINT já existente. A implementação da restrição se da seguinte maneira:

```
ALTER TABLE BARRACA
  ADD CONSTRAINT participante_barraca
    UNIQUE (cpf);
```

Asserção 2

Outra regra apresentada na descrição do projeto é que para cada veículo de comunicação cadastrado, é permitido ter no máximo dois jornalistas associados. Como um jornalista pode estar associado a diferentes veículos de comunicação, essa relação foi implementada por meio de uma tabela de relacionamento, TRABALHA_EM. A asserção sugerida tem como objetivo garantir que não existam mais de dois registros nessa relação por veículo de comunicação.

Solução em SQL

Segue a mesma estrutura da primeira, porém filtra os resultados da tabela TRABALHA_EM como maiores que 2, que é o limite implicado pela regra de negócio para essa restrição. A implementação da asserção se da seguinte maneira:

```
create assertion restricao_jornalista_veiculo check
(not exists (
      SELECT COUNT(*) FROM TRABALHA_EM GROUP BY codigo_vc
       HAVING COUNT(*) > 2)
);
```

Solução para o Oracle

Como já foi dito, foi necessário escolher outra abordagem para modelar a asserção para o Oracle. Nesse caso, foi escolhida a utilização de um TRIGGER. Ele é realizado no momento BEFORE do evento INSERT na tabela TRABALHA_EM, porque após verificar a regra, se ela for quebrada, é disparado um erro que irá cancelar o processo de inserção na tabela.

Aninhado em DECLARE temos a declaração de uma variável (qtdJornalistas), que vai armazenar o resultado de uma consulta, que retorna a quantidade de jornalistas associados ao veículo de comunicação que está sendo inserido na tabela de relacionamento.

A condição para disparar o erro é maior ou igual a 2 porque ela ocorre antes da inserção, então ela não contabiliza o registro que está sendo inserido. A condição maior está colocada porque essa verificação é feita apenas no INSERT, de forma que no UPDATE é possível cadastrar exceções, resultando em casos onde um veículo pode ter mais do que dois jornalistas associados. A implementação do gatilho se da seguinte maneira:

```
create or replace TRIGGER CHECK_JORNALISTA_VEICULO
   BEFORE INSERT ON TRABALHA_EM
   FOR EACH ROW
   DECLARE
      qtdJornalistas INT;
   BEGIN
      SELECT COUNT(*) INTO qtdJornalistas
      FROM TRABALHA_EM o
      WHERE o.codigo_vc = :new.codigo_vc;

      IF (qtdJornalistas >= 2)
      THEN
```

```
            RAISE_APPLICATION_ERROR( -20001, 'Numero maximo de jornalistas por veiculo
        excedido!' );
          END IF;
        END;
```

Casos de teste

Teste A (Asserção 2)
Alterar a relação TRABALHA_EM, para que existam três jornalistas associados ao mesmo veículo de comunicação. Esse teste tem como objetivo expor uma possível falha na solução sugerida, visto que ela trata apenas no momento de adicionar um novo registro. Por um lado isso pode ser visto como, uma falha, mas também pode ser uma solução que se tratada em conjunto com permissões de modificação, pode garantir que o sistema está aberto a exceções às regras padrões de limite de jornalistas.

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

```
INSERT INTO JORNALISTA (cpf, credencial, especialidade) VALUES ('66666666666', 2, 'Tecnologia');

INSERT INTO VEICULO_COMUNICACAO (codigo_vc, nome_vc, meio_comunicacao) VALUES (4, 'Veja',
'revista');

INSERT INTO TRABALHA_EM (cpf, codigo_vc) VALUES ('66666666666', 4);
INSERT INTO TRABALHA_EM (cpf, codigo_vc) VALUES ('66666666666', 2);
```

Ação de teste
```
UPDATE TRABALHA_EM SET codigo_vc = 4 WHERE codigo_vc IN (3,1);
```

Resultado
Atualização feita com sucesso na relação TRABALHA_EM, permitindo que exista 3 jornalistas associados ao mesmo veículo de comunicação.

Teste B (Asserção 1)
Inserir uma barraca com um participante que já está registrado em outra barraca. Esse teste pode representar um erro comum do mundo real onde uma pessoa está realizando o cadastro para mais de um indivíduo e pode inserir seu próprio documento para todas as reservas de barraca que estiverem sendo cadastradas.

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

```
INSERT INTO BARRACA (id_barraca, cpf, modelo, marca) VALUES (1, '11111111111', 'Modelo 1', 'Marca
1');
```

Ação de teste
```
INSERT INTO BARRACA (id_barraca, cpf, modelo, marca) VALUES (7, '11111111111', 'Modelo 1', 'Marca
2');
```

Resultado

Teste C (Asserção 2)
Inserir três registros na relação TRABALHA_EM referentes ao mesmo veículo de comunicação. Essa caso de teste representa um cenário padrão para o qual a restrição foi desenvolvida. Também tem como funcionalidade demonstrar o funcionamento do procedimento de cancelar uma inserção por meio do envio de um erro durante o processamento do gatilho.

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

```
INSERT INTO TRABALHA_EM (cpf, codigo_vc) VALUES ('11111111111', 2);
INSERT INTO TRABALHA_EM (cpf, codigo_vc) VALUES ('22222222222', 2);
```

Ação de teste
```
INSERT INTO TRABALHA_EM (cpf, codigo_vc) VALUES ('33333333333', 2);
```

Resultado