Universidade Federal de São Carlos - Campus Sorocaba
Ciências da Computação
Banco de Dados

Sistema de Gerenciamento de Aeroportos

Fase Final

Prof. Dra. Sahudy Montenegro González

Grupo 6

Nome: Bernardo Pinheiro Camargo	RA: 620343
Nome: Chady Chaito	RA: 420420
Nome: Vitor Pratali Camillo	RA: 620181

2018

Sumário

1 Especificação do problema

Um aeroporto foi construído e necessita um sistema de gerenciamento de banco de dados para armazenar, manipular e recuperar os dados de forma eficiente, já que existem muitos dados todos os dias. No aeroporto existe a área de estacionamento, os terminais, os voos, as companhias aéreas responsáveis pelos voos e serviços, como restaurantes, lojas das mais diversas, entre outros.

1.1 Objetivo do sistema

O objetivo do nosso projeto é a modelagem de um banco de dados a fim de resolver o problemas com a organização dos dados de um aeroporto. O sistema deverá armazenar informações referentes aos estacionamentos, terminais, voos, companhias aérea e serviços oferecidos pelo aeroporto e realizar as consultas necessárias.

Cada companhia aérea terá um nome que a identifica. Uma companhia aérea pode ter um ou vários voos programados para serem realizados.

Os voos programados são identificados por um código único, um horário e uma data que este voo acontecerá, além das siglas do aeroporto de origem e do aeroporto de destino, o status do voo (cancelado, previsto, atrasado, partindo, última chamada, embarque imediato, confirmado). Os voos poderão ser realizados por uma ou várias companhias aéreas. Além disso, os voo estão localizados em ou vários terminais.

Os terminais do aeroporto são identificados pelo nome e são compostos pelo tipo do terminal (carga, embarque ou desembarque) além de observações sobre sua localização. Um terminal possui um ou vários voos programados, além de possuir um ou vários estacionamentos e, dentro de um terminal, pode ter nenhum ou vários serviços sendo oferecidos. Além disso cada terminal possui um ou vários estacionamentos ligados a ele.

O estacionamento do aeroporto é identificado por um nome, além de mostrar a quantidade de vagas existentes, sua categoria (econômico, padrão e premium) e o valor por cada hora que o veículo fica estacionado. Um estacionamento pode ter um ou vários terminais ligados a ele.

Os diversos serviços, que são oferecidos pelo aeroporto, são compostos por um código único de cada serviço, o nome do estabelecimento, a categoria do serviço (alimentação, lojas, facilidades), um telefone de contato para cada serviço e o horário de funcionamento do estabelecimento. É importante ressaltar que cada serviço está localizado em um ou vários terminais dentro do aeroporto.

2 Requisitos de Dados

Nesta seção será apresentado os requisitos de dados, especificando quais as consultas que o sistema deverá realizar.

- **RD01.** O sistema deve consultar o voo pelo destino.

  - Atributos de visualização do resultado: código, destino, origem, companhia aérea, terminal, data, horário e status.

  - Atributos de busca (ou de condições/filtros): destino.

- **RD02.** O sistema deve consultar quantos voos existem por companhia aérea.

  - Atributos de visualização do resultado: nome da companhia, count (código companhia aérea).

  - Atributos de busca (ou de condições/filtros): código da companhia aérea

- **RD03.** O sistema deve consultar os serviços pelo nome e categoria.

  - Atributos de visualização do resultado: categoria, telefone, horário_funcionamento, nome.

  - Atributos de busca (ou de condiçõesfiltros): nome e categoria.

- **RD04.** O sistema deve consultar os voos por uma faixa de data.

  - Atributos de visualização do resultado: código, destino, origem, companhia aérea, terminal, data, horário e status.

  - Atributos de busca (ou de condiçõesfiltros):data.

- **RD05.** O sistema deve consultar a quantidade média de voos por dia durante um mês.

  - Atributos de visualização do resultado:data, avg (código voo).

  - Atributos de busca (ou de condiçõesfiltros): data.

- **RD06.** O sistema deve consultar os serviços dado um terminal.

  - Atributos de visualização do resultado:categoria, telefone, horário_funcionamento, nome.

  - Atributos de busca (ou de condiçõesfiltros): nome.

3 Projeto conceitual

Nesta seção será apresentado o projeto conceitual, que consta do Modelo Entidade-Relacionamento (MER) e também a descrição do tipo de cada atributo por entidade.

3.1 Modelo Entidade-Relacionamento (MER)

Abaixo, na Figura 1, se encontra o MER que é usado para descrever, de maneira conceitual, como será o armazenamento dos dados no sistema.

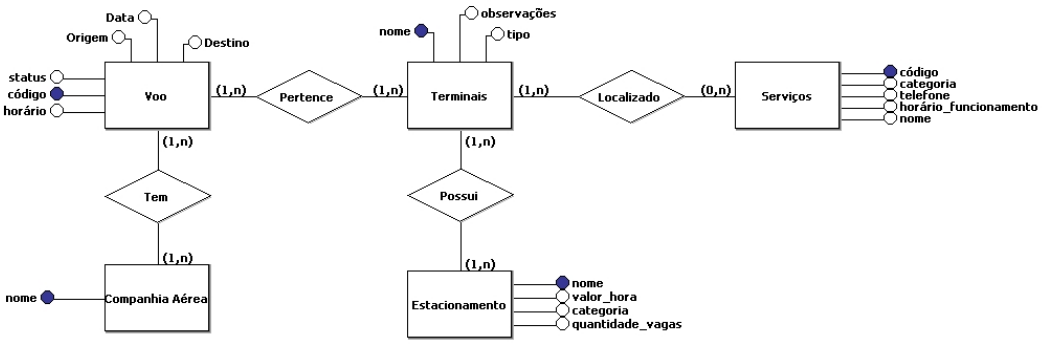

Figura 1: Modelo EntidadeRelacionamento

3.2 Descrição do tipo de atributos por entidade

Nesta seção será apresentado os atributos por cada entidade, descrevendo o tipo do atributo e a restrição deste. Esses dados podem ser vistos na Tabela 1.

Tipo Entidade	Atributo	Tipo	Restrição
Voo	Código	Chave Primária	Obrigatório
	Horário	Monovalorado	Obrigatório
	Origem	Monovalorado	Obrigatório
	Destino	Monovalorado	Obrigatório
	Data	Monovalorado	Obrigatório, >= data corrente
	Status	Monovalorado	Obrigatório, entre cancelado, previsto, atrasado, partindo, última chamada, embarque imediato ou confirmado
Companhia Aérea	Nome	Chave Primária	Obrigatório
Terminais	Nome	Chave Primária	Obrigatório
	Tipo	Monovalorado	Obrigatório, entre carga, embarque ou desembarque
	Observações	Monovalorado	Opcional
Estacionamento	Valor_hora	Monovalorado	Obrigatório
	Categoria	Monovalorado	Obrigatório
	Quantidade_vagas	Monovalorado	Obrigatório
Serviços	Código	Chave Primária	Obrigatório
	Categoria	Monovalorado	Obrigatório
	Telefone	Monovalorado	Opcional
	Horário_funcionamento	Monovalorado	Obrigatório
	Nome	Monovalorado	Obrigatório

Tabela 1: Atributos por entidade

4 Projeto Lógico

Nesta seção será apresentado o projeto lógico, quais são as tabelas que foram geradas a partir do projeto conceitual e também a descrição da forma que foi gerado este modelo lógico e se este projeto está na terceira forma de normalização.

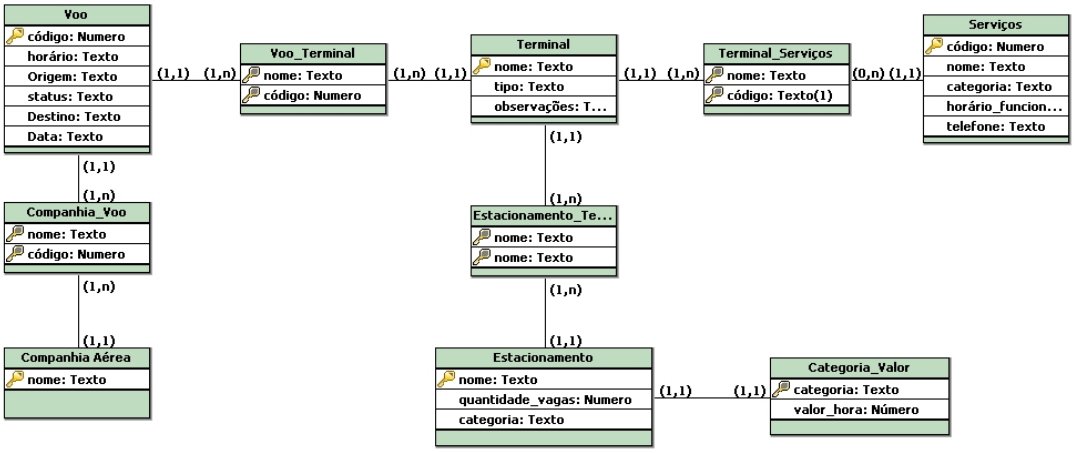

Figura 2: Modelo Lógico

A transformação entre os modelos foi feita com o auxílio da ferramenta BrModelo e alguns ajustes foram feitos manualmente para que o modelo lógico representasse, fielmente, nosso Modelo Entidade-Relacionamento além das modificações necessárias para que o projeto atendesse a 3ª Forma Normal.

Como podemos visualizar a partir de nosso Modelo Lógico, o projeto encontra-se na 3ª Forma Normal. Explicação:

- O Projeto encontra-se na 1ª Forma Normal pois não possui chaves multivaloradas;
- O Projeto encontra-se na 2ª Forma Normal pois está na 1ª Forma Normal e nenhum atributo possui dependência parcial de superchaves;
- O Projeto encontra-se na 3ª Forma Normal pois está na 2ª Forma Normal e não possui dependência entre atributos não-chave

Para que fosse necessário chegar a 3ª Forma Normal foi necessário modificar os atributos categoria e valor_hora que faziam parte da entidade Estacionamento, para uma nova entidade de nome Categoria_Valor sendo categoria a chave primária e valor_hora um atributo não chave. Esta mudança foi necessária pois dada uma categoria já era esperado o valor_hora, tendo assim uma dependência entre atributos nãochave dentro da entidade Estacionamento.

5 Projeto físico do banco de dados

A criação do banco de dados acontece, primeiramente, com a criação das tabelas através do SQL. As tabelas geradas são: **voo**, **companhia_aerea**, **terminal**, **estacionamento**, **servicos**, **companhia_voo**, **voo_terminal**, **estacionamento_terminal**, **categoria_valor** e **terminal_serviços**. Os *scripts* gerados para a criação do banco de dados pode ser encontrado no arquivo *esquema.sql*.

Em relacao as politicas de integridade dos dados em nosso sistema, consiste em cláusula com **NOT NULL**, **PRIMARY KEY**, **FOREIGN KEY**, **CASCADE** e **UNIQUE**.

Na tabela **voo**, temos o atributo **código** como **PRIMARY KEY**, que é o identificados único para diferenciar cada voo. Temos também o **horário**, a **origem**, **destino**, **data** e **status** do voo, nos quais são colunas do tipo **VARCHAR** e para estes atribuímos a cláusula **NOT NULL**, de forma a serem obrigatórios em uma possível inserção ou atualização.

Na tabela **companhia_aerea**, temos apenas um atributo, que é o **nome** da companhia aérea, onde este é do tipo **VARCHAR**, de forma a ser obrigatório, **NOT NULL** e ainda é **PRIMARY KEY**.

Na tabela **terminal**, temos o **nome** do terminal sendo do tipo **VARCHAR**, sendo obrigatório, ou seja, **NOT NULL** e ainda sendo **PRIMARY KEY**, como sendo o identificador único. Além desse atributo, temos o **tipo** do terminal, ou seja, se é de embarque, desembarque ou carga, é do tipo **VARCHAR** e obrigatório, **NOT NULL**. E por fim, temos o atributo **observacoes** onde é um tipo **TEXT** do que podemos colocar um texto para referenciar o terminal.

Na tabela **estacionamento**, temos o nome do estacionamento sendo **PRIMARY KEY**, do tipo **VARCHAR** e sendo obrigatório, **NOT NULL**. Temos também, o atributo **categoria**, no qual se refere a categoria do estacionamento, se este é econômico, padrão ou premium, assim este é do tipo **VARCHAR**, sendo obrigatório, **NOT NULL**, e tendo um valor único, **UNIQUE**. O último atributo, temos o **qtd_vagas** que refere-se a quantidade de vagas que existe em cada estacionamento, sendo este atributo um **SMALLINT** e obrigatório, **NOT NULL**.

Na tabela **servicos**, temos o identificador único de cada serviço, que seria pelo atributo **codigo**, que é do tipo **SERIAL** e **PRIMARY KEY**. Além deste atributo, temos temos os atributos **nome**, **categoria**, **telefone** e **horario_funcionamento**, onde todos são do tipo **VARCHAR**, sendo todos obrigatórios, **NOT NULL**, em excessão do telefone que é opcional.

Na tabela **companhia_voo**, temos o atributo **nome** que é **VARCHAR** e o atributo **codigo** que é **INTEGER**, ambos são chaves primárias desta tabela e chaves estrangeiras da tabela **companhia_aerea** e **voo**, respectivamente. Como política de integridades de dados usamos o **CASCADE** em caso de atualização ou exclusão de algum dado das tabelas c**ompanhia_aerea** e **voo**.

Na tabela **voo_terminal**, temos o atributo **nome** que é **VARCHAR** e o atributo **codigo** que é **INTEGER**, ambos são chaves primárias desta tabela e chaves estrangeiras da tabela **terminal** e **voo**, respectivamente. Como política de integridades de dados usamos o **CASCADE** em caso de atualização ou exclusão de algum dado das tabelas terminal e voo.

Na tabela **estacionamento_terminal**, temos o atributo **nome_terminal** que é **VARCHAR** e o atributo **nome_estacionamento** que é **VARCHAR**, ambos são chaves primárias desta tabela e chaves estrangeiras da tabela terminal e estacionamento, respectivamente. Como política de integridades de dados usamos o **CASCADE** em caso de

atualização ou exclusão de algum dado das tabelas terminal e estacionamento.

Na tabela **categoria_valor**, temos o atributo **categoria** que é **VARCHAR** e o atributo **valor_hora** que é **SMALLINT** e **NOT NULL**. O atributo categoria é **PRIMARY KEY** desta tabela e também é chave estrangeira da tabela estacionamento. Como política de integridades de dados usamos o **CASCADE** em caso de atualização ou exclusão de algum dado da tabela estacionamento.

Na tabela **terminal_servicos**, temos o atributo **nome** que é **VARCHAR** e o atributo **codigo** que é **INTEGER**, ambos são chaves primárias desta tabela e chaves estrangeiras da tabela **terminal** e **servicos**, respectivamente. Como política de integridades de dados usamos o **CASCADE** em caso de atualização ou exclusão de algum dado das tabelas terminal e servicos.

É importante ressaltar que os casos de integridade do nosso banco de dados, como o status do voo, data do voo e tipo do terminal serão testados a partir das triggers, como pode ser visto na seção 7 deste documento.

Em relação a alimentacao do banco de dados, foi necessario criar vários dados verídicos para inserção nas tabelas. Os scripts de insercao de dados no banco de dados se encontram no arquivo dados.sql.

6 Especificação de consultas em Álgebra Relacional e SQL

- Consulta do voo pelo destino.
  - SQL
    - SELECT A.codigo, A.origem, A.destino,
      A.data, A.horario, A.status,
      B.nome "companhia aerea",
      C.nome "terminal"
      FROM voo A
      INNER JOIN companhia_voo B ON B.CODIGO = A.CODIGO
      INNER JOIN voo_terminal C ON C.CODIGO = A.CODIGO
      WHERE A.destino = <nome destino>
  - AR
    - T1 $\leftarrow voo \bowtie voo.codigo = companhia\_voo.codigo(companhia\_voo)$
      $T2 \leftarrow T1 \bowtie T1.codigo = terminal\_voo.codigo(terminal\_voo)$
      $T3 \leftarrow \sigma destino = <nomedestino>(T2)$

      $\prod código, origem, destino, data, horarios, status, companhiaaérea, terminal(T3)$

- Consulta de quantos voos existem por companhia aérea.
  - SQL
    - SELECT ca.nome, COUNT(v.codigo)
      FROM companhia_aerea as ca, companhia_voo as cv, voo as v
      WHERE ca.nome = cv.nome AND cv.codigo = v.codigo
      GROUP BY ca.nome;
  - AR
    - T1 $\leftarrow companhia\_aereacompanhia\_voo$
      $T2 \leftarrow T1 \bowtie voo$
      $\prod nome\mathscr{F}count(codigo)(T2)$

- Consulta os serviços pelo nome e categoria.
  - SQL
    - SELECT se.nome, se.categoria, se.telefone, se.horario_funcionamento
      FROM servicos AS se
      WHERE se.categoria LIKE <%categoria%>
      AND se.nome LIKE <%nome%>
  - AR
    - T1 $\leftarrow \sigma servicos.categoria = <\%categoria\%>$
      $T2 \leftarrow \sigma servicos.nome = <\%nome\%>$
      $T3 \leftarrow T1 \wedge T2$
      $\prod nome, categoria, telefone, horario\_funcionamento$

- Consulta os voos por uma faixa de data
  - SQL
    - SELECT A.codigo, A.origem, A.destino, A.data, A.horario, A.status,
      B.nome, C.nome
      FROM voo AS A
      INNER JOIN companhia_voo AS B ON B.CODIGO = A.CODIGO
      INNER JOIN voo_terminal AS C ON C.CODIGO = A.CODIGO
      WHERE A.data BETWEEN <inicio.data> AND <final.data>
  - AR
    - T1 $\leftarrow voo \bowtie voo.codigo = companhia\_voo.codigo(companhia\_voo)$
      $T2 \leftarrow T1 \bowtie T1.codigo = terminal\_voo.codigo(terminal\_voo)$
      $T3 \leftarrow data \geq <datainicio> \wedge data \leq <datafinal>(T2)$
      $\prod código, origem, destino, data, horarios, status, companhiaaérea, terminal(T3)$

- Consulta a quantidade média de voos por mês durante um ano
  - SQL
    - SELECT voos.ano, AVG(voos.qtd_voo) AS
      media_voos_p_mes
      FROM (
      SELECT date_part('Month', voo.data) AS mes,
      date_part('Year', voo.data) AS ano,
      COUNT(voo.codigo) AS qtd_voo
      FROM voo
      GROUP BY date_part('Month', voo.data),
      date_part('Year', voo.data)
) AS voos
      GROUP BY voos.ano

- AR
  - T1 $\leftarrow \mathscr{F} count(voo.codigo)(voo)$
    $T2 \leftarrow \prod mes, ano, qtd\_voo(T1)$
    $T3 \leftarrow \prod \mathscr{F} avg(T1.qtd\_voo)(T2)$

- Consulta os serviços dado um terminal
  - SQL
    - SELECT s.nome, s.categoria, s.telefone,
      s.horario_funcionamento
      FROM terminal as t, terminal_servicos as ts, servicos as s
      WHERE t.nome = <nome terminal> AND t.nome = ts.nome
      AND ts.codigo = s.codigo

  - AR
    - T1 $\leftarrow \sigma nome = <nome>(terminal)$
      $T2 \leftarrow T1 \bowtie terminal_servicos$
      $T3 \leftarrow T2 \bowtie servicos$
      $\prod nome, categoria, telefone, horario_funcionamento(T3)$

7 Triggers

Foram feitas tres triggers, uma para verificar quando esta havendo um UPDATE ou INSERT na tabela voo, para garantir que a nova tupla a ser inserida ou a tupla que esta sendo modificada tenha os dados dentro dos padroes estabelecidos, no caso, o voo deve ter uma data válida e que seja maior ou igual a data autal, ela pode ser observada na Figura 3.

```
-- Verificar a DATA do voo
CREATE OR REPLACE FUNCTION verificar_data() RETURNS TRIGGER AS $data$
    BEGIN
        IF NEW.data < CURRENT_DATE THEN
            RAISE EXCEPTION 'Data inserida é inválida';
        END IF;
        RETURN NEW;
    END
$data$ LANGUAGE plpgsql;

CREATE TRIGGER verificar_data BEFORE INSERT OR UPDATE ON voo
    FOR EACH ROW EXECUTE PROCEDURE verificar_data();
```

Figura 3: Trigger para verificar data

A segunda trigger foi feita para verificar quando esta havendo um UPDATE ou INSERT na tabela voo, para garantir que a nova tupla a ser inserida ou a tupla que esta sendo modificada tenha os dados dentro dos padroes estabelecidos, no caso, o voo tem que ter os dados corretos no campo status, que são: Cancelado, Previsto, atrasado, partindo, ultima chamada, embarque imediato, confirmado, decolado e desembarcando. Esta pode ser vista na figura 4.

```
-- Verifica status --
CREATE OR REPLACE FUNCTION verificar_status() RETURNS TRIGGER AS $status$
    BEGIN
        IF NOT (NEW.status = 'Cancelado' OR NEW.status = 'Previsto' OR NEW.status = 'Atrasado' OR NEW.status = 'Partindo'
            OR NEW.status = 'Ultima Chamada' OR NEW.status = 'Embarque Imediato' OR NEW.status = 'Confirmado'
            OR NEW.status = 'Decolado' OR NEW.status = 'Desembarcando') THEN
            RAISE EXCEPTION 'O status inserido não é permitido';
        END IF;
        RETURN NEW;
    END
$status$ LANGUAGE plpgsql;

CREATE TRIGGER verificar_status BEFORE INSERT OR UPDATE ON voo
    FOR EACH ROW EXECUTE PROCEDURE verificar_status();
```

Figura 4: Trigger para verificar status

A terceira trigger foi feita para verificar quando esta havendo um UPDATE ou INSERT na tabela terminal, para garantir que a nova tupla a ser inserida ou a tupla que esta sendo modificada tenha os dados dentro dos padroes estabelecidos, no caso, o tipo de terminal deve estar entre umas das opções de Carga, Embarque, Desembarque, Embarque/Desembarque. Esta pode ser vista na figura 5.

```
-- Tipo terminal --
CREATE OR REPLACE FUNCTION tipoTerminal() RETURNS TRIGGER AS $tipo$
    BEGIN
        IF NOT (NEW.tipo = 'Carga' OR NEW.tipo = 'Embarque' OR NEW.tipo = 'Desembarque' OR NEW.tipo = 'Embarque/Desembarque') THEN
            RAISE EXCEPTION 'O tipo do terminal inserido não é permitido';
        END IF;
        RETURN NEW;
    END
$tipo$ LANGUAGE plpgsql;

CREATE TRIGGER tipoTerminal BEFORE INSERT OR UPDATE ON terminal
    FOR EACH ROW EXECUTE PROCEDURE tipoTerminal();
```

Figura 5: Trigger para verificar tipo do terminal

8 Considerações finais

Conforme o idealizado, na fase 3 deste projeto foi desenvolvido o projeto fisico do banco de dados onde conceitualmente foi desenvolvida a Algebra Relacional para cada consulta e entao implementado em SQL, alem, e claro, da implementacao dos codigos que geram as tabelas do banco com suas devidas restricoes de dados.

Para fins de testes tambem foram criadas rotinas de insercao de dados em cada tabela do Banco, sendo que esses dados possuem informacoes que fazem sentido, nao somente qualquer cadeia de caracteres aleatorios.

Tambem foi necessaria a utilizacao de Triggers para verificar se os dados estao nos padroes pre-estabelecidos e entao manter a consistencia dos dados das tabelas voo e terminal quando houver um Insert ou Update.

As maiores dificuldades desse projeto apareceram ao desenvolver as Algebras Relacionais, principalmente na consulta de quantidade média de voos por mês durante um ano