Grupo 6 - Aeroporto

Bernardo Camargo – 696969

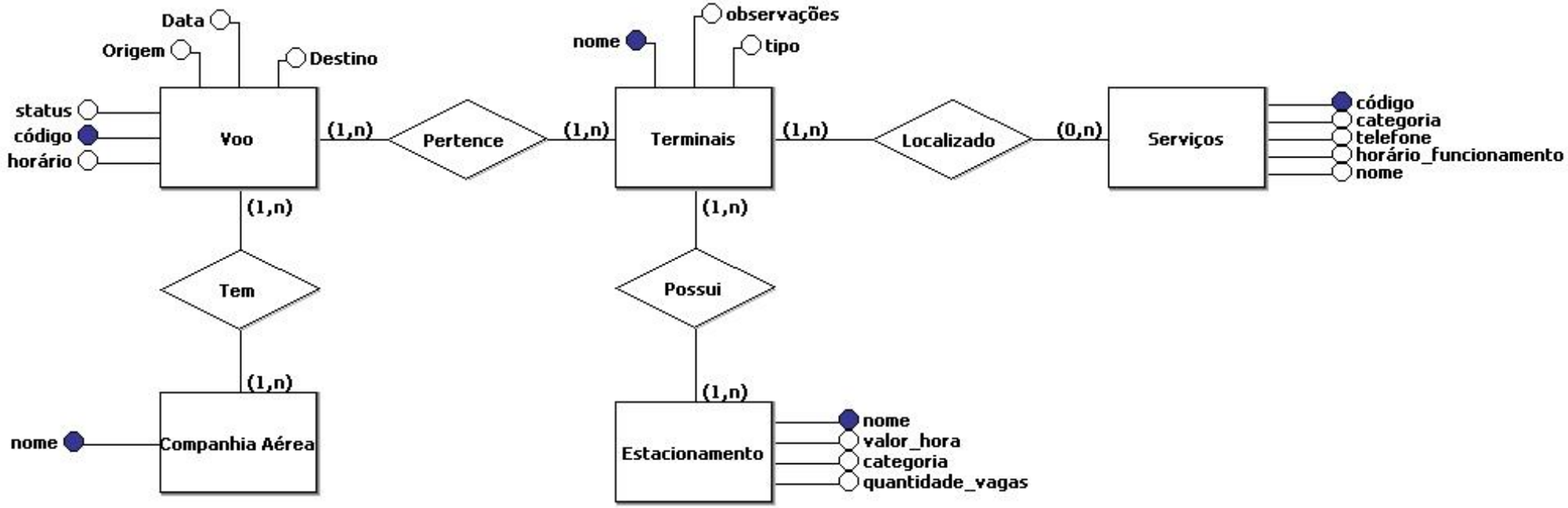
Chady Chaito – 613797

Vitor Pratali Camillo - 620181

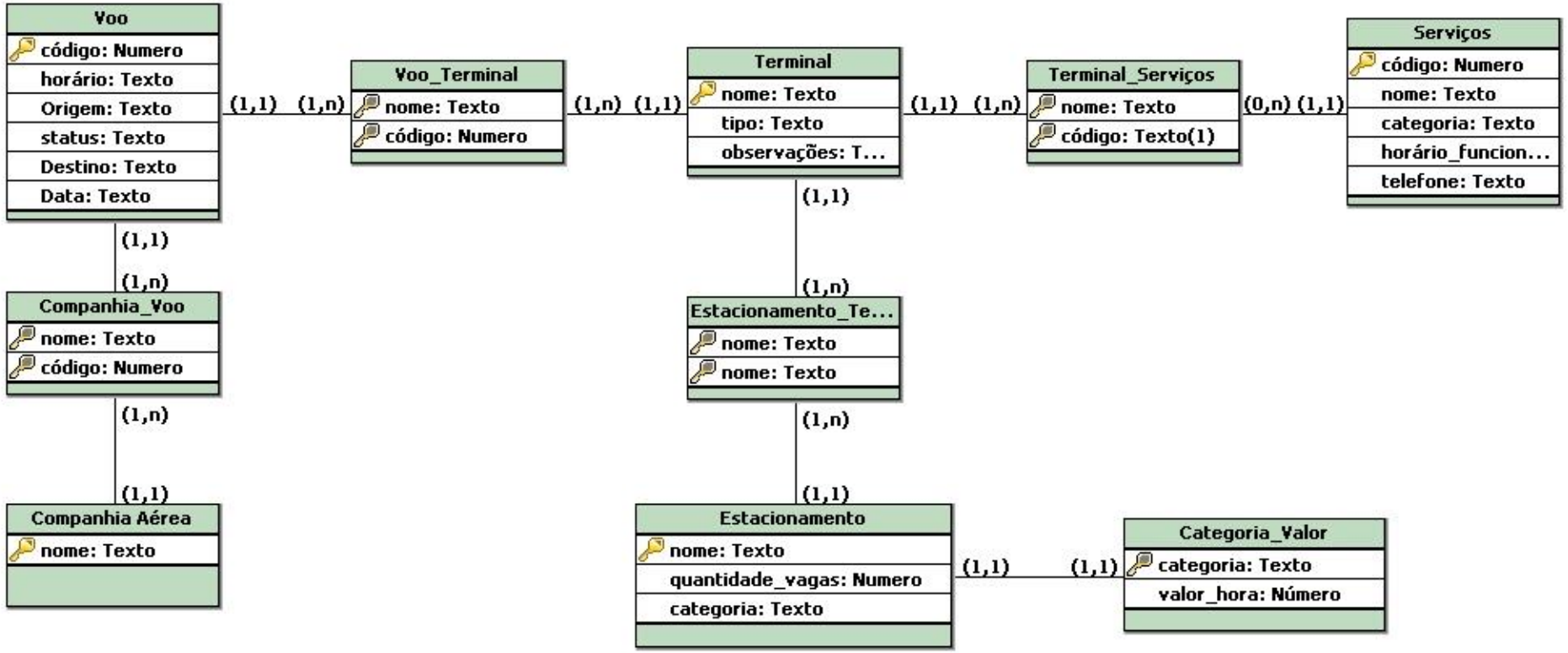
Introdução

- Sistema de gerenciamento de aeroporto
- Principais entidades voo, companhia aérea, terminal, estacionamento e serviços

MER


Data
Origem
Destino
status
código
horário
Voo
(1,n)
Pertence
(1,n)
nome
observações
tipo
Terminais
(1,n)
Localizado
(0,n)
Serviços
código
categoria
telefone
horário_funcionamento
nome
(1,n)
Tem
(1,n)
nome
Companhia Aérea
(1,n)
Possui
(1,n)
Estacionamento
nome
valor_hora
categoria
quantidade_vagas

Projeto lógico



Projeto Físico

- **Tabelas geradas:** *voo, companhia_aerea, terminal, estacionamento, servicos, companhia_voo, voo_terminal, estacionamento_terminal, categoria_valor e terminal_serviços.*
- **Cláusulas de integridade:** *NOT NULL , PRIMARY KEY , FOREIGN KEY , CASCADE e UNIQUE .*
- O esquema do banco de dados pode ser visto no *esquema.sql*.
- Os dados inseridos em nosso banco podem ser visto no *dados.sql*.

Consultas

- **Chady**
  - RD01 - O sistema deve consultar quantos voos existem por companhia aérea.
    - 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 <- voo |x| voo.codigo = companhia_voo.codigo companhia_voo

T2 <- T1 |x| t1.codigo = terminal_voo.codigo terminal_voo

T3 <- σ destino = <nome destino> (T2)

Π código, origem, destino, data, horarios, status, companhia aérea, terminal (T3)

Consultas

- **Vitor**
  - RD02 - O sistema deve consultar 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 <- companhia_aerea ⋈ companhia_voo

      t2 <- t1 ⋈ voo

      nome $\mathcal{F}$count(codigo)(t2)

Consultas

- **Bernardo**
  - RD03 - O sistema deve consultar 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 ← σservicos.categoria = <%categoria%>
      T2 ← σ servicos.nome = <%nome%>
      T3 ← T1 ∧ T2
      π nome,categoria,telefone,horario funcionamento (T3)

Consultas

- **Chady**
  - RD04 - O sistema deve consultar os voos por uma faixa de data.
    - 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.data BETWEEN <inicio.data> AND <data final>
```

    - AR:

T1 <- voo |x| voo.codigo = companhia_voo.codigo companhia_voo

T2 <- T1 |x| t1.codigo = terminal_voo.codigo terminal_voo

T3 <- σ data >= <data inicio> ^ data <= <data final> (T2)

Π código, origem, destino, data, horarios, status, companhia aérea, terminal (T3)

Consultas

- **Bernardo**
  - RD05 - O sistema deve consultar a quantidade me´dia de voos por dia durante um mês.
  - 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 ← F count(voo.codigo)(voo)

T2 ← ▫mes,ano,qtd voo(T1)

T3 ← ▫Favg(T1.qtd voo)(T2)

Consultas

- **Vitor**
  - RD06 - O sistema deve consultar 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 <- σ nome = <nome> (terminal)

      t2 <- t1 ⋈ terminal_servicos

      t3 <- t2 ⋈ servicos

      π nome, categoria, telefone, horario_funcionamento (t3)

Triggers

- Verificar status de um voo

```
-- 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();
```

Triggers

- Verificar o tipo do terminal

```
-- 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();
```

Triggers

- Verificar se a data inserida é válida

```
-- 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();
```

Obrigado!

Dúvidas?