UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT
GRADUAÇÃO – CIÊNCIA DA COMPUTAÇÃO

ANA PAULA CARNEIRO ATHAYDE
HENRIQUE FRANCO SCUR

TRABALHO DE BANCO DE DADOS 2: RESERVA DE PASSAGENS

JOINVILLE
2022

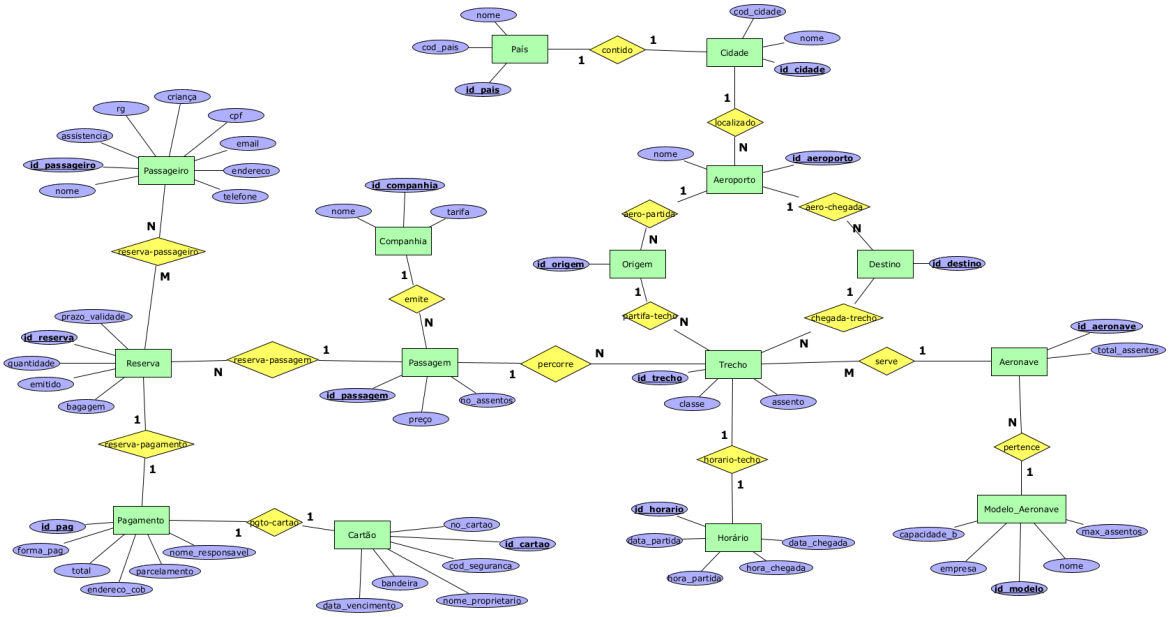
1. SUMÁRIO

2. TEMA

Sistema de reservas para uma companhia de aviação, uma companhia de aviação deseja gerenciar as reservas de passagens de seus aviões:

- Uma reserva é identificada por um código gerado pelo sistema no computador. A reserva é feita para um único passageiro, do qual se conhece apenas o nome. A reserva compreende um conjunto de trechos de voos, que acontecerão em determinada data/hora. Para cada trecho, a reserva é feita em uma classe (econômica, executiva, etc.).
- Um voo é identificado por um código e possui uma origem e um destino. Por exemplo, o voo 595 sai de Porto Alegre com destino a São Paulo. Um vôo é composto por vários trechos, correspondendo às escalas intermediárias do voo. Assim, o voo 595 pode ser composto de dois trechos, um de Porto Alegre a Curitiba, e outro de Curitiba a São Paulo. Cabe salientar que há cidades que são servidas por vários aeroportos. Por isso,é importante informar ao passageiro que faz a reserva em qual aeroporto o voo passa.
- Às vezes, ao fazer a reserva, os clientes querem saber qual é o tipo de aeronave que será utilizada em determinados trechos do voo. Alguns poucos voos, principalmente internacionais, têm troca de aeronave em determinadas escalas.
- Nem todos os voos operam em todos os dias da semana. Inclusive, certos voos têm pequenas mudanças de horário em certos dias da semana.
- Cada reserva possui um prazo de validade. Caso os bilhetes não tenham sido emitidos até esgotar-se o prazo da reserva, a mesma é cancelada. Reservas podem ser prorrogadas.
- Como o check-in de todos os voos está informatizado, a companhia possibilita a reserva de assento para passageiros. Reservas de assento podem ser feitas com até três meses de antecedência.
- Além de efetuar reservas, o sistema deve servir para vários tipos de consultas que os clientes podem querer fazer:
    - Possibilidade de viagem de uma cidade ou de um aeroporto para outro;
    - Restringir as possibilidades de viagem por determinados dias da semana;
    - Restringir as possibilidades de viagem por determinados horários de saída e
    - chegada;
    - Disponibilidade de vagas em um trecho de vôo;
    - Disponibilidade de determinados assentos em um trecho de voo.

3. PROJETO CONCEITUAL



4. MAPEAMENTO CONCEITUAL → LÓGICO

- Pagamento(#id_pag, forma_pag, total, endereco_cob, parcelamento, nome_responsavel, &id_cartao)
    - id_cartao: chave estrangeira da entidade cartão para a relação pgto-cartao.
- Cartao(#id_cartao, no_cartao, cod_seguranca, nome_proprietario, bandeira, data_vencimento)
- Reserva(#id_reserva, prazo_validade, quantidade, emitido, bagagem, &id_pag, &id_passagem)
    - id_pag: chave estrangeira da entidade pagamento para a relação reserva-pagamento;
    - id_passagem: chave estrangeira da entidade passagem para a relação reserva-passagem.
- Passageiro(#id_passageiro, nome, assistencia, rg, cpf, crianca, email, endereco, telefone)
- Companhia(#id_companhia, nome, tarifa)
- Passagem(#id_passagem, preco, no_assentos, &id_companhia)
    - id_companhia: chave estrangeira da entidade companhia para a relação emite.
- Trecho(#id_trecho, classe, assento, &id_horario, &id_passagem, &id_aeronave, &id_origem, &id_destino)
    - id_horario: chave estrangeira da entidade horario para a relação horario-trecho;
    - id_passagem: chave estrangeira da entidade passagem para a relação percorre;
    - id_aeronave: chave estrangeira da entidade aeronave para a relação serve;
    - id_origem: chave estrangeira da entidade origem para a relação partida-trecho;
    - id_destino: chave estrangeira da entidade destino para a relação chegada-trecho.
- Horario(#id_horario, data_partida, hora_partida, data_chegada, hora_chegada)
- Origem(#id_origem, &id_aeroporto)
    - id_aeroporto: chave estrangeira da entidade aeroporto para a relação aero-partida.
- Destino(#id_destino, &id_aeroporto)
    - id_aeroporto: chave estrangeira da entidade aeroporto para a relação aero-chegada.
- Aeroporto(#id_aeroporto, nome, &id_cidade)
    - id_cidade: chave estrangeira da entidade cidade para a relação localizado.

- Cidade(#id_cidade, nome, cod_cidade, &id_pais)
    - id_pais: chave estrangeira da entidade pais para a relação contido.
- Pais(#id_pais, nome, cod_pais)
- Aeronave(#id_aeronave, total_assentos, &id_modelo)
    - id_modelo: chave estrangeira da entidade modelo_aeronave para a relação pertence.
- Modelo_aeronave(#id_modelo, nome, max_assentos, empresa, capacidade_b)
- Reserva_Passageiro(#&id_reserva, #&id_passageiro)
    - id_reserva: chave estrangeira da entidade reserva para a relação reserva-passageiro;
    - id_passageiro: chave estrangeira da entidade passageiro para a reserva-passageiro.

5. IMPLEMENTAÇÃO DE FUNÇÕES E GATILHOS

Funções:

1. Retornar cartões com vencimento depois do dia 13/12/2022;
2. Retornar a quantidade máxima de assentos de todos os modelos de aeronave;
3. Retornar o nome de todas as companhias;
4. Retornar o nome das cidades pertencentes ao país Brasil;
5. Retornar se cartão com o número passado já existe;
6. Retornar se cpf com o número passado já existe;
7. Verificar data de vencimento do cartão;
8. Verificar se existe algum trecho com o horário passado;

Gatilhos:

9. Antes de adicionar um cartão verificar se existe outro com o mesmo número;
10. Não permitir adicionar dois cpfs iguais;
11. Não permitir mudar vencimentos de cartão para uma data anterior a atual;
12. Não permitir deletar um horário que já foi utilizado em um trecho.

1. Retornar cartões com vencimento depois do dia 13/12/2022:

```
CREATE OR REPLACE FUNCTION verificarVencimentoMaiorData (dataChecar date)
RETURNS char(16) AS $$
BEGIN
        RETURN 'SELECT no_cartao FROM cartao WHERE data_vencimento >
'||dataChecar|| ';';
END;
$$
LANGUAGE plpgsql;

SELECT * FROM verificarVencimentoMaiorData (‘2022-12-13’);
```

2. Retornar a quantidade máxima de assentos de todos os modelos de aeronave:

```
CREATE OR REPLACE FUNCTION quantidaMaxModAero()
RETURNS integer AS $$
BEGIN
        RETURN 'SELECT max_assentos  FROM modelo_aeronave;';
END;
$$
LANGUAGE plpgsql;
```

3. Retornar o nome de todas as companhias:

```
CREATE OR REPLACE FUNCTION nomeCompanhias()
```

```
RETURNS character varying(50)  AS $$
BEGIN
	RETURN 'SELECT nome FROM companhia;';
END;
$$
LANGUAGE plpgsql;
```

4. Retornar o nome das cidades pertencentes ao país Brasil:

```
CREATE OR REPLACE FUNCTION cidadesPais(nomePais character varying(50))
RETURNS character varying(50) AS $$
BEGIN
	RETURN 'SELECT cidade.nome FROM cidade  JOIN pais ON pais.id_pais =
cidade.id_pais WHERE pais.nome = '||nomePais ||';';
END;
$$
LANGUAGE plpgsql;
```

5. Retornar se cartão com o número passado já existe:

```
CREATE OR REPLACE FUNCTION verificarNuCartao()
RETURNS TRIGGER AS $$
BEGIN
	IF TG_OP = 'INSERT' THEN
		IF (SELECT count(1) FROM cartao WHERE no_cartao=new.no_cartao) > 0
THEN
			RAISE EXCEPTION 'Numero de cartão já cadastrado';
		END IF;
	ELSIF TG_OP = 'UPDATE' THEN
		IF (new.no_cartao <> old.no_cartao) THEN
			IF (SELECT count(1) FROM cartao WHERE
no_cartao=new.no_cartao) > 0 THEN
			RAISE EXCEPTION 'Numero de cartão já cadastrado';
		END IF;
	END IF;
END IF;
RETURN new;
END;
$$
LANGUAGE plpgsql;
```

6. Antes de adicionar um cartão verificar se existe outro com o mesmo número:

```
CREATE TRIGGER  verificarNuCartao
BEFORE INSERT OR UPDATE ON cartao
FOR ROW EXECUTE PROCEDURE verificarNuCartao();
```

7. Retornar se cpf com o número passado já existe:

```
CREATE OR REPLACE FUNCTION verificarCPF()
RETURNS TRIGGER AS $$
BEGIN
	IF TG_OP = 'INSERT' THEN
		IF (SELECT count(1) FROM passageiro WHERE cpf =new.cpf) > 0 THEN
			RAISE EXCEPTION 'CPF já cadastrado';
		END IF;
	ELSIF TG_OP = 'UPDATE' THEN
		IF (new.cpf  <> old.cpf ) THEN
			IF (SELECT count(1) FROM passageiro WHERE cpf =new.cpf) > 0
THEN
			RAISE EXCEPTION 'CPF já cadastrado';
		END IF;
	END IF;
END IF;
RETURN new;
END;
$$
LANGUAGE plpgsql;
```

8. Não permitir adicionar dois cpfs iguais:

```
CREATE TRIGGER  verificarCPF
BEFORE INSERT OR UPDATE ON passageiro
FOR ROW EXECUTE PROCEDURE verificarCPF();
```

9. Verificar data de vencimento do cartão:

```
CREATE OR REPLACE FUNCTION verificaVencimento()
RETURNS TRIGGER AS $$
DECLARE
	dataAtual date = 'SELECT GETDATE()';
BEGIN
	IF TG_OP = 'INSERT' THEN
		IF (new.data_vencimento < dataAtual) THEN
			RAISE EXCEPTION 'Data que deseja inserir e menor que a data
atual';
		END IF;
	ELSIF TG_OP = 'UPDATE' THEN
		IF (new.data_vencimento  <> old.data_vencimento ) THEN
			IF (new.data_vencimento < dataAtual) THEN
				RAISE EXCEPTION 'Data que deseja inserir e menor que a
data atual';
			END IF;
	END IF;
END IF;
```

```
RETURN new;
END;
$$
LANGUAGE plpgsql;
```

10. Não permitir mudar vencimentos de cartão para uma data anterior a atual:

```
CREATE TRIGGER  verificaVencimento
BEFORE INSERT OR UPDATE ON cartao
FOR ROW EXECUTE PROCEDURE verificaVencimento();
```

11. Verificar se existe algum trecho com o horário passado:

```
CREATE OR REPLACE FUNCTION verificaHorario()
RETURNS TRIGGER AS $$
BEGIN
	IF((SELECT COUNT(1) FROM horario JOIN trecho ON horario.id_horario =
trecho.id_horario WHERE horario.id_horario = old.horario.id_horario) > 0) THEN
		RAISE EXCEPTION 'O horario ja esta relacionado a um trecho';
	END IF;
	RETURN OLD;
END;
$$
LANGUAGE plpgsql;
```

12.Não permitir deletar um horário que já foi utilizado em um trecho:

```
CREATE TRIGGER verificaHorario
BEFORE DELETE ON horario
FOR EACH ROW EXECUTE PROCEDURE verificaHorario();
```