PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL

MODELAÇÃO E IMPLEMENTAÇÃO DE BANCO DE DADOS NoSQL COM APACHE CASSANDRA

Sistema de Reservas Aeroportuárias

AUGUSTO PERONI BALDINO

Prof. Eduardo Henrique P. de Arruda

Porto Alegre
2025

Sumário

1 Introdução

Este relatório apresenta a modelagem e implementação de um banco de dados NoSQL utilizando Apache Cassandra para um sistema de reservas aeroportuárias. O trabalho segue a metodologia de modelagem orientada por consultas (query-driven design) com foco em desempenho e alta disponibilidade.

2 Definição das Consultas

2.1 Sequência Q1 → Q2 (Keyspace: `booking`)

2.1.1 Q1: Consulta de Reservas por Voo

Descrição: Buscar todas as reservas de um voo específico em uma data/hora de partida, incluindo detalhes dos passageiros.

2.1.2 Q2: Consulta de Reservas por ID

Descrição: Buscar informações específicas de uma reserva através do seu ID único.

2.2 Sequência Q3 → Q4 (Keyspace: `flights`)

2.2.1 Q3: Consulta de Voos por Companhia Aérea

Descrição: Buscar todos os voos de uma companhia aérea específica em uma data determinada.

2.2.2 Q4: Consulta de Informacões de Aeroportos

Descrição: Buscar informações detalhadas de aeroportos específicos através do código do aeroporto.

3 Esquema Lógico

3.1 Keyspace: `booking`

3.1.1 Tabela: `bookings_by_flight`

- Partition Key: (flightno, departure)
- Clustering Key: booking$_id$
- Atributos: passenger$_id$, $first_name$, $last_name$, $seat$

3.1.2 Tabela: `bookings_by_id`

- Partition Key: booking$_id$
- Atributos: flightno, departure, first$_name$, $last_name$, $seat$, $passenger_id$

3.2 Keyspace: `flights`

3.2.1 Tabela: `flights_by_airline`

- Partition Key: (airline$_c$*ode*, *departure*$_d$*ate*)
- Clustering Key: flightno
- Atributos: departure$_t$*ime*, *origin*, *destination*

3.2.2 Tabela: `airports_info`

- Partition Key: airport$_c$*ode*
- Atributos: name, city, country

3.2.3 Tabela: `airlines_info`

- Partition Key: airline$_c$*ode*
- Atributos: estimated$_n$*ame*

4 Comandos CQL DDL (Data Definition Language)

4.1 Criação dos Keyspaces

```
-- Keyspace para sistema de reservas
CREATE KEYSPACE booking WITH REPLICATION = {
  'class': 'SimpleStrategy',
  'replication_factor': 1
};

-- Keyspace para informacoes de voos e aeroportos
CREATE KEYSPACE flights WITH REPLICATION = {
  'class': 'SimpleStrategy',
  'replication_factor': 1
};
```

4.2 Criação das Tabelas - Keyspace `booking`

```
-- Tabela para consulta Q1: reservas por voo
CREATE TABLE booking.bookings_by_flight (
  flightno text,
  departure timestamp,
  booking_id int,
  passenger_id int,
  first_name text,
  last_name text,
  seat text,
  PRIMARY KEY ((flightno, departure), booking_id)
);

```

```
-- Tabela para consulta Q2: reservas por ID
CREATE TABLE booking.bookings_by_id (
  booking_id int,
  flightno text,
  departure timestamp,
  seat text,
  last_name text,
  first_name text,
  passenger_id int,
  PRIMARY KEY (booking_id)
);
```

4.3 Criação das Tabelas - Keyspace `flights`

```
-- Tabela para consulta Q3: voos por companhia area
CREATE TABLE flights.flights_by_airline (
  airline_code text,
  departure_date date,
  departure_time timestamp,
  flightno text,
  origin text,
  destination text,
  PRIMARY KEY ((airline_code, departure_date), flightno)
);

-- Tabela para consulta Q4: informacoes de aeroportos
CREATE TABLE flights.airports_info (
  airport_code text,
  name text,
  city text,
  country text,
  PRIMARY KEY (airport_code)
);

-- Tabela adicional: informacoes de companhias aereas
CREATE TABLE flights.airlines_info (
  airline_code text,
  estimated_name text,
  PRIMARY KEY (airline_code)
);
```

5 Comandos CQL DML (Data Manipulation Language)

5.1 Inserções no Keyspace `booking`

```
-- Inseres para consulta Q1
INSERT INTO booking.bookings_by_flight (
  flightno, departure, booking_id, passenger_id,
  first_name, last_name, seat
```

```
) VALUES (
  'SP3100', '2020-11-17 08:40:14', 181727, 4239,
  'Anthony Tyler', 'Quinn', '16F'
);

INSERT INTO booking.bookings_by_flight (
  flightno, departure, booking_id, passenger_id,
  first_name, last_name, seat
) VALUES (
  'SP3100', '2020-11-17 08:40:14', 181728, 2524,
  'B. J.', 'Thomas', '14A'
);
```

```
-- Inseres para consulta Q2
INSERT INTO booking.bookings_by_id (
  booking_id, flightno, departure, seat,
  last_name, first_name, passenger_id
) VALUES (
  181727, 'SP3100', '2020-11-17 08:40:14', '16F',
  'Quinn', 'Anthony Tyler', 4239
);

INSERT INTO booking.bookings_by_id (
  booking_id, flightno, departure, seat,
  last_name, first_name, passenger_id
) VALUES (
  181728, 'SP3100', '2020-11-17 08:40:14', '14A',
  'Thomas', 'B. J.', 2524
);
```

5.2 Inserções no Keyspace `flights`

```
-- Inseres para consulta Q3
INSERT INTO flights.flights_by_airline (
  airline_code, departure_date, departure_time,
  flightno, origin, destination
) VALUES (
  'SP', '2020-11-17', '2020-11-17 08:40:14',
  'SP3100', 'GRU', 'BSB'
);

INSERT INTO flights.flights_by_airline (
  airline_code, departure_date, departure_time,
  flightno, origin, destination
) VALUES (
  'AA', '2020-11-17', '2020-11-17 10:30:00',
  'AA1001', 'GRU', 'MIA'
);
```

```
-- Inseres para consulta Q4
INSERT INTO flights.airports_info (
```

```
  airport_code, name, city, country
) VALUES (
  'GRU', 'Guarulhos International Airport', 'Sao Paulo', 'Brazil'
);

INSERT INTO flights.airports_info (
  airport_code, name, city, country
) VALUES (
  'BSB', 'Brasilia International Airport', 'Brasilia', 'Brazil'
);

-- Inseres de companhias aereas
INSERT INTO flights.airlines_info (
  airline_code, estimated_name
) VALUES (
  'SP', 'SATA Air Acores'
);

INSERT INTO flights.airlines_info (
  airline_code, estimated_name
) VALUES (
  'AA', 'American Airlines'
);
```

6 Comandos CQL DQL (Data Query Language)

6.1 Consultas do Keyspace booking

```
-- Q1: Buscar todas as reservas do voo SP3100 na data/hora
   especifica
SELECT * FROM booking.bookings_by_flight
WHERE flightno = 'SP3100'
AND departure = '2020-11-17 08:40:14';

-- Variacao: Buscar apenas nomes dos passageiros
SELECT first_name, last_name, seat FROM booking.
   bookings_by_flight
WHERE flightno = 'SP3100'
AND departure = '2020-11-17 08:40:14';
```

```
-- Q2: Buscar informacoes especificas da reserva 181727
SELECT * FROM booking.bookings_by_id
WHERE booking_id = 181727;

-- Variacao: Buscar multiplas reservas
SELECT * FROM booking.bookings_by_id
WHERE booking_id IN (181727, 181728);
```

6.2 Consultas do Keyspace flights

```
-- Q3: Buscar todos os voos da companhia SP em 17/11/2020
SELECT * FROM flights.flights_by_airline
WHERE airline_code = 'SP'
AND departure_date = '2020-11-17';

-- Variacao: Buscar apenas horarios e destinos
SELECT flightno, departure_time, destination
FROM flights.flights_by_airline
WHERE airline_code = 'SP'
AND departure_date = '2020-11-17';
```

```
-- Q4: Buscar informacoes do aeroporto GRU
SELECT * FROM flights.airports_info
WHERE airport_code = 'GRU';

-- Variacao: Buscar multiplos aeroportos
SELECT * FROM flights.airports_info
WHERE airport_code IN ('GRU', 'BSB', 'MIA');

-- Consulta adicional: Informacoes de companhias aereas
SELECT * FROM flights.airlines_info
WHERE airline_code = 'SP';
```

7 Justificativas de Design

7.1 Escolhas de Partition Keys

Explanação sobre como as *partition keys* foram escolhidas para distribuir os dados adequadamente e otimizar as consultas (ver documento original).

- **bookings_by_flight** `(flightno, departure)`: agrupa todas as reservas de um voo específico em uma partição, ideal para operações de *check-in*.
- **bookings_by_id** `(booking_id)`: permite acesso direto a reservas individuais, com *lookup* de complexidade $O(1)$.
- **flights_by_airline** `(airline_code, departure_date)`: organiza voos por companhia e data, facilitando consultas operacionais.

7.2 Desnormalização

O modelo utiliza desnormalização intencional, armazenando dados duplicados em diferentes tabelas para otimizar consultas, seguindo o princípio *"uma tabela por consulta"* do Cassandra.

- Dados duplicados em múltiplas tabelas para otimizar consultas específicas.
- Elimina a necessidade de *JOINs* complexos.
- *Trade-off*: maior uso de armazenamento em troca de um ganho significativo em desempenho.

7.3 Estratégia de Clustering

As clustering keys foram escolhidas para ordenar dados dentro das particoes e permitir filtros eficientes sem hotspots.

8 Implementação e Resultados

8.1 Configuração do Ambiente

Ambiente configurado utilizando Docker com Apache Cassandra para facilitar testes e replicacao.

8.2 Performance das Consultas

Resumo dos resultados: consultas por partition key completa apresentam performance ótima; consultas por chave primaria simples apresentaram performance excelente.

9 Conclusão

O modelo desenvolvido atende aos requisitos de um sistema aeroportuário moderno, proporcionando escalabilidade, disponibilidade e desempenho necessários para operações críticas.

O modelo implementado atende plenamente aos requisitos de um sistema de reservas aeroportuário, conforme os seguintes aspectos:

- **Performance**: consultas otimizadas com acesso direto via *partition keys*.
- **Escalabilidade**: arquitetura distribuída que cresce horizontalmente.
- **Disponibilidade**: tolerância a falhas nativa do Cassandra.
- **Organização**: separação lógica em *keyspaces* distintos.

A abordagem de modelagem orientada a consultas mostrou-se eficaz para atender aos padrões de acesso específicos do domínio aeroportuário, proporcionando uma base sólida para operações críticas como *check-in*, gestão de voos e atendimento ao cliente.