Cassandra

CARACTERÍSTICAS

- Open source e gratuito
- Escalável e alta tolerância a falhas de hardware
- Alta disponibilidade
- Replicação em diferentes datacenters de forma eficiente

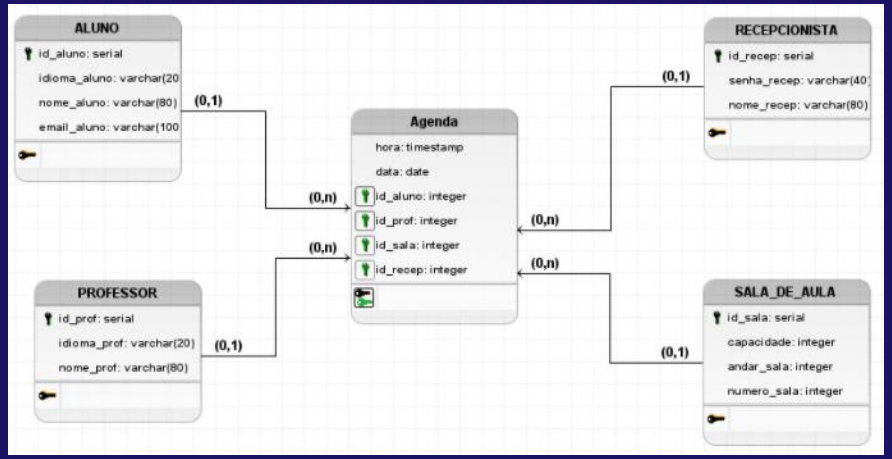
REPRESENTAÇÕES DE MODELOS

Cassandra

```
session.execute('USE naruhodo;')
```

```
session.execute('CREATE TABLE professor (id_prof int PRIMARY KEY, nome_prof text, idioma_prof text);')
session.execute('CREATE TABLE aluno (id_aluno int PRIMARY KEY, nome_aluno text, email_aluno text,  idioma_aluno text);')
session.execute('CREATE TABLE recepcionista (id_recep int PRIMARY KEY, nome_recep text, senha_recep text);')
session.execute('CREATE TABLE sala_de_aula (id_sala int PRIMARY KEY, numero_sala int, andar_sala int, capacidade int);')
session.execute('CREATE TABLE agenda (hora timestamp, data date, id_aluno int, id_prof int, id_sala int, id_recep int, PRIMARY KEY (id_aluno, id_prof, id_sala, id_recep));
```

Postgres



CONEXÃO VIA PYSPARK

É necessária a instalação da bibliotecas cassandra-driver.

O import de cluster e auth servem para a conexão com o astra.

O import query é para a execução dos comandos no banco de dados.

```
!pip install cassandra-driver
```

```
import cassandra; print (cassandra.__version__)
```

```
from cassandra.cluster import Cluster
from cassandra.auth import PlainTextAuthProvider
from cassandra.query import SimpleStatement

cloud_config= {
    'secure_connect_bundle': 'secure-connect-naruhodo.zip'
}
auth_provider = PlainTextAuthProvider('naruhodo','123456')
cluster = Cluster(cloud=cloud_config, auth_provider=auth_provider)
session = cluster.connect()
```

INCLUSÃO DE DADOS

```
session.execute("INSERT INTO professor (id_prof, nome_prof, idioma_prof) VALUES (%s, %s, %s);", (10101,'Jorge', 'ingles'))
session.execute("INSERT INTO professor (id_prof, nome_prof, idioma_prof) VALUES (%s, %s, %s);", (12121,'Luiz', 'ingles'))
session.execute("INSERT INTO professor (id_prof, nome_prof, idioma_prof) VALUES (%s, %s, %s);", (13131,'Vitoria', 'ingles'))
session.execute("INSERT INTO professor (id_prof, nome_prof, idioma_prof) VALUES (%s, %s, %s);", (14141,'Ingrid', 'ingles'))
session.execute("INSERT INTO professor (id_prof, nome_prof, idioma_prof) VALUES (%s, %s, %s);", (15151,'Maristela', 'ingles'))
session.execute("INSERT INTO professor (id_prof, nome_prof, idioma_prof) VALUES (%s, %s, %s);", (16161,'Carla', 'espanhol'))
session.execute("INSERT INTO professor (id_prof, nome_prof, idioma_prof) VALUES (%s, %s, %s);", (17171,'Brenda', 'espanhol'))
session.execute("INSERT INTO professor (id_prof, nome_prof, idioma_prof) VALUES (%s, %s, %s);", (18181,'Andre', 'espanhol'))
session.execute("INSERT INTO professor (id_prof, nome_prof, idioma_prof) VALUES (%s, %s, %s);", (19191,'Luna', 'frances'))
session.execute("INSERT INTO professor (id_prof, nome_prof, idioma_prof) VALUES (%s, %s, %s);", (20202,'Stella', 'frances'))
```

SELEÇÃO DE DADOS

```
[46] statement = SimpleStatement("SELECT * FROM professor", fetch_size=10)
     for user_row in session.execute(statement):
         print(user_row)
```

```
Row(id_prof=13131, idioma_prof='ingles', nome_prof='Vitoria')
Row(id_prof=18181, idioma_prof='espanhol', nome_prof='Andre')
Row(id_prof=10101, idioma_prof='ingles', nome_prof='Jorge')
Row(id_prof=20202, idioma_prof='frances', nome_prof='Stella')
Row(id_prof=14141, idioma_prof='ingles', nome_prof='Ingrid')
Row(id_prof=17171, idioma_prof='espanhol', nome_prof='Brenda')
Row(id_prof=19191, idioma_prof='frances', nome_prof='Luna')
Row(id_prof=15151, idioma_prof='ingles', nome_prof='Maristela')
Row(id_prof=16161, idioma_prof='espanhol', nome_prof='Carla')
Row(id_prof=12121, idioma_prof='ingles', nome_prof='Luiz')
```

```
[86] statement = SimpleStatement("SELECT * FROM sala_de_aula", fetch_size=10)
     for user_row in session.execute(statement):
         print(user_row)
```

```
Row(id_sala=201, andar_sala=2, capacidade=5, numero_sala=1)
Row(id_sala=203, andar_sala=2, capacidade=15, numero_sala=3)
Row(id_sala=102, andar_sala=1, capacidade=10, numero_sala=2)
Row(id_sala=101, andar_sala=1, capacidade=5, numero_sala=1)
Row(id_sala=202, andar_sala=2, capacidade=10, numero_sala=5)
Row(id_sala=103, andar_sala=1, capacidade=5, numero_sala=3)
```

```
[84] statement = SimpleStatement("SELECT * FROM recepcionista", fetch_size=10)
     for user_row in session.execute(statement):
         print(user_row)
```

```
Row(id_recep=4040, nome_recep='Geovana', senha_recep='senha4')
Row(id_recep=3030, nome_recep='Larissa', senha_recep='senha3')
Row(id_recep=2020, nome_recep='Débora', senha_recep='senha2')
Row(id_recep=1010, nome_recep='Glória', senha_recep='senha1')
```

EXCLUSÃO DE DADOS

```
session.execute("DELETE FROM aluno WHERE id_aluno = 5;")
session.execute("DELETE FROM professor WHERE id_prof = 15151;")
```

ATUALIZAÇÃO DE DADOS

```
session.execute("UPDATE professor SET nome_prof = 'Marianne' WHERE id_prof = 20202;")
session.execute("UPDATE aluno SET idioma_aluno  = 'frances' WHERE id_aluno = 2;")
```

PONTOS POSITIVOS

- Escalabilidade de recursos
- Consistência configurável
- Lida bem com falha de hardware
- Writes rápidos

PONTOS NEGATIVOS

- Não é possível utilizar joins
- Não é possível implementar chaves estrangeiras
- Reads podem ser lentos

Referências

Documentação comandos CQL
https://docs.datastax.com/en/cql-oss/3.3/cql/cql_reference/cqlCommandsTOC.html

Playlist Tutorial Cassandra
https://www.youtube.com/playlist?list=PLalrWAGybpB-L1PGA-NfFu2uiWHEsdscD

Stack Overflow
https://stackoverflow.com