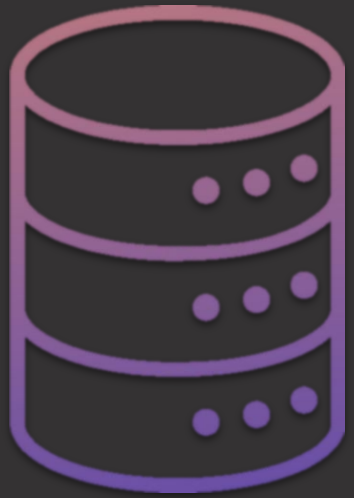
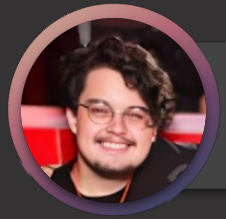
Database



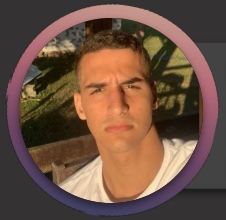
Grupo: Sherlock

Equipe



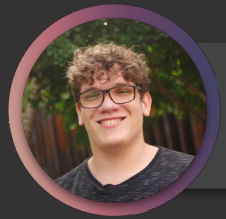
Antonio Henrique leitão Barros
Ciência da computação
Matrícula: 01647043

Daniel de Araújo Procópio da Cunha
Sistemas de informação
Matrícula: 01643124



Nickolas Mac'Hamilton Renaux Alves
Sistemas de informação
Matrícula: 01651053

Rafael José de Araújo Procópio da Cunha
Sistemas de informação
Matrícula: 01589882



Gabriel de Luna Cavalcanti
Ciência da computação
Matrícula: 01331511

Cassandra

É um banco de dados NoSQL do tipo colunar, seu estilo é “BASE”, no entanto possui algumas poucas características semelhantes ao “ACID”.

Prioriza disponibilidade de dados, logo é considerado “Basically Available”.

Não precisa ser consistente o tempo todo, sendo então “Soft-state”.

Consistente em momentos indeterminados, ou seja “Eventually consistent”

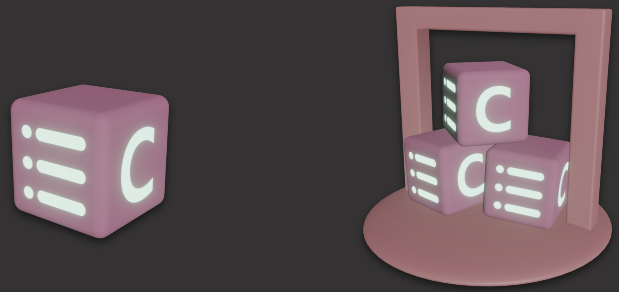
Suas funcionalidades foram arquitetadas com escalabilidade e alta avaliabilidade em mente, teoricamente, pode conter uma quantidade de nodos ilimitados em um cluster,e clusters são dispersos e os seus dados são transacionados entre diversos pontos.

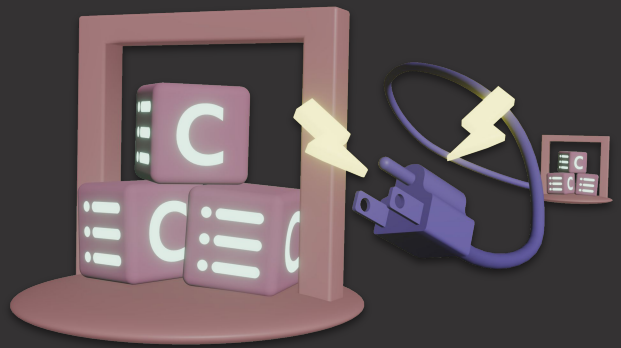
Desnormalizado e “schemaless”.

Necessita do JVM (Java Virtual Machine).

Workbenches

CAP & Estruturas





Nodes

Os **nodos** são instâncias individuais do Cassandra, cada **nodo** geralmente corresponde a uma máquina, seja ela uma unidade física ou virtualizada.

Racks

Agrupamento lógico de **nodos** em um mesmo datacenter, usados para definir o layout físico, comunicação e topologia.

Cluster

Os **clusters** são um conjunto de **nodos**, possivelmente pertencentes a **racks**, organizados de maneira lógica e de formas diversas, podendo estar a grandes distâncias geograficamente ou fisicamente próximos.

Pontos

Positivos

- Basicamente disponível -> Muitos usuários podem acessar o banco de dados e garante disponibilidade até na presença de falhas (sistema distribuído).
- Arquitetura distribuída -> Dados são replicados em diversos **nodos**, que comunicam-se entre si através de ações como o “gossip protocol” e “hinted handoff”.
- Escalabilidade e “no-bottleneck” -> É possível aumentar o armazenamento e processamentos de forma horizontal onde cada nodo processa um certo workload.

Negativos

- Eventualmente consistente -> Seus dados são atualizados eventualmente.
- Complexidade da estrutura -> Seu “schema” deve ser cuidadosamente desenvolvido, já que desnormalização é uma prática comum no Cassandra, o que pode aumentar a complexidade da busca e processamento.
- Investimento em escalabilidade -> Escalar horizontalmente gera custos maiores, e também adiciona complexidade ao sistema.

DDL

CREATE KEYSPACE

ALTER KEYSPACE

DROP KEYSPACE

CREATE TABLE

ALTER TABLE

DROP TABLE

(ADD)

DML

INSERT

DELETE

UPDATE

(INTO)

(VALUES)

(SET)

(WHERE)

DQL

SELECT

(FROM)

(WHERE)

(COUNT)

(SUM)

(AVG)

(MIN)

(MAX)

(*)

Referências

O que é Cassandra

https://www.scylladb.com/learn/apache-cassandra/introduction-to-apache-cassandra/#:~:text=Cassandra%20is%20not%20a%20fully,commits%2C%20rollbacks%20or%20locking%20mechanisms.

Cassandra clusters

https://www.scylladb.com/glossary/cassandra-cluster/#:~:text=A%20Cassandra%20cluster%20is%20a,physical%20outages%20of%20underlying%20infrastructure.

“ACID” & “BASE”

https://aws.amazon.com/pt/compare/the-difference-between-acid-and-base-database/

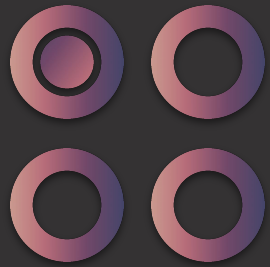
Comandos DDL, DML & DQL

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

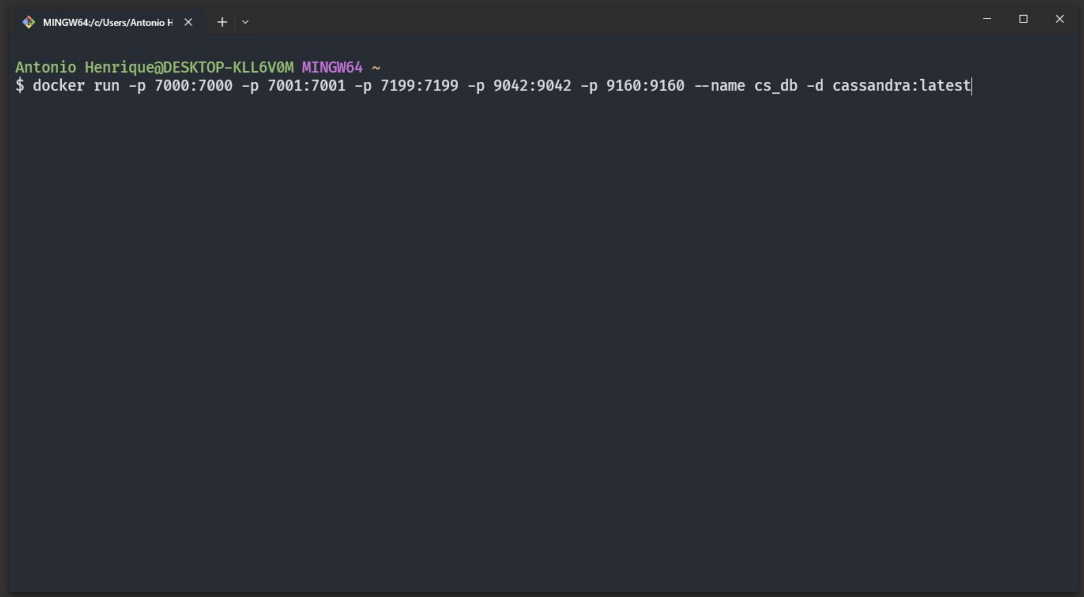
PDF

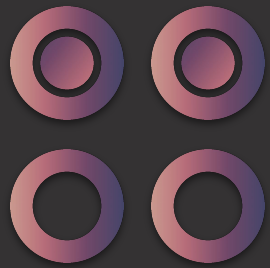
NoSQL - Como armazenar os dados de uma aplicação moderna.

Cassandra no docker

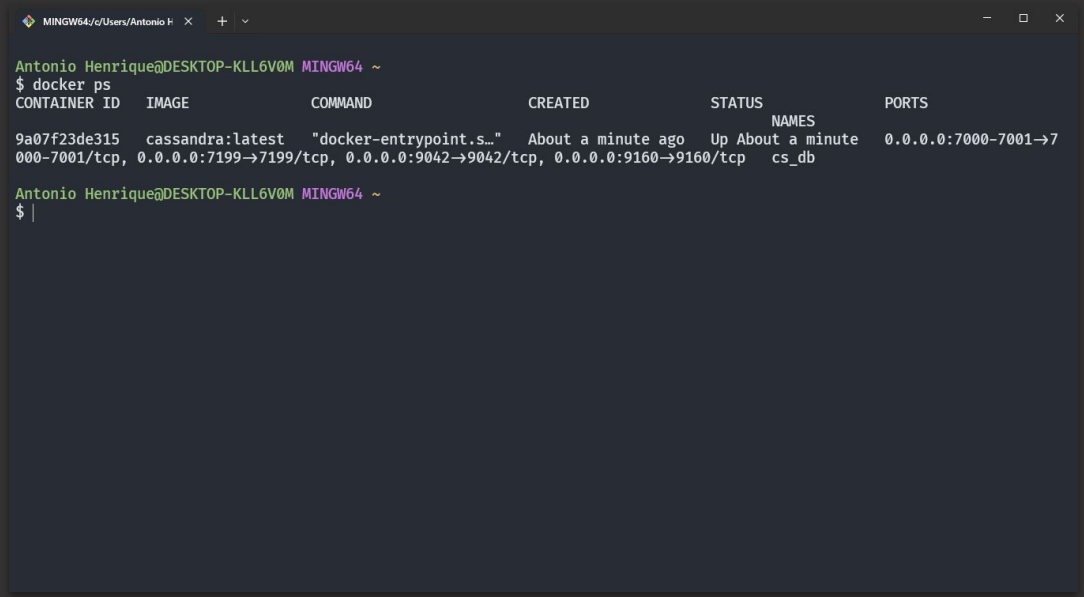


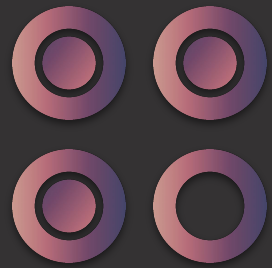
Cassandra no docker



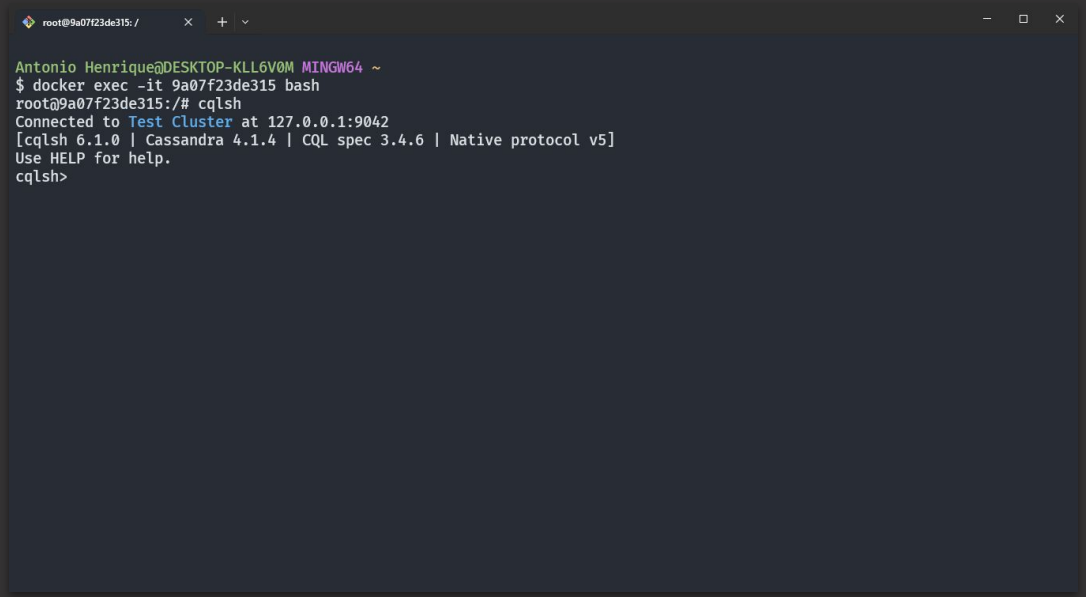


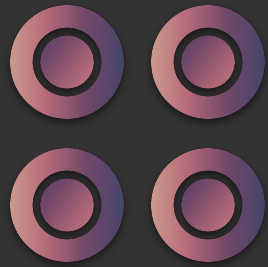
Cassandra no docker





Cassandra no docker



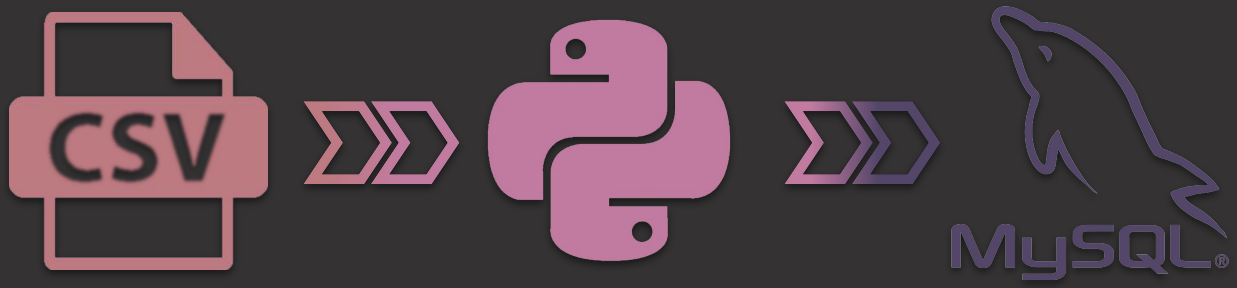


Dataset escolhido

Como um marco inicial, foi selecionado nosso “DataSet” por meio da plataforma KAGGLE, sendo escolhido um “dataset” contendo diversas estatísticas de vários os jogadores profissionais de CS:GO até o ano de 2022.

kaggle

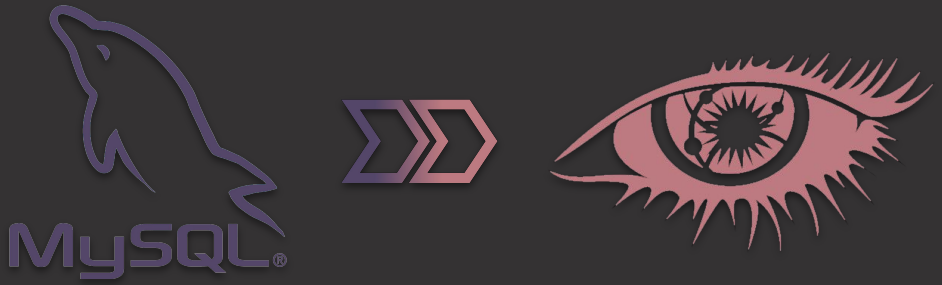
Dataset para MySQL



Ao termos nosso Dataset e seu arquivo CSV em posse, tratamos de importar tais dados por meio do Python ao MYSQL workbench. Para isto, tivemos que definir inicialmente como iríamos dispor nossas “tables” de forma que o relacionamento entre elas respeitassem cardinalidades e operações lógicas.

Usamos nosso arquivo CSV para popular nosso banco de dados MySQL.
Com todos os dados em mãos, passamos para a próxima fase: o ajuste de código para que fosse passado ao banco de dados NoSQL Cassandra.

MySQL para Cassandra



De uma maneira mais simples, após termos feito algumas pesquisas, optamos por unir todas as tabelas que havíamos feito, tendo em vista que o Cassandra é conhecido por ser um banco de dados orientado à PESQUISA, apenas importando que os dados estejam ali presentes, e sejam destinados ao seu devido dono.

Após criarmos nosso KEYSPACE e sua “table” “Player_info”, passamos o processo de transferência de DATA, do MySQL para o Cassandra, tendo em vista que já havíamos “preparado o terreno” uma vez que determinamos nossa “table” com nomes que eram comuns às nossas tabelas de MySQL.

Na sequência foi só chamarmos os dados por meio de um “select * from {tabela}”, passando cada tabela por um processo de ajuste da função “transfer_data”, que leu cada e posicionou seus dados devidamente em nossa tabela “player_info” do Cassandra.

CREATE & USE

```
CREATE KEYSPACE cs_db WITH replication = {'class':'SimpleStrategy', 'replication_factor' : 1};

USE cs_db;

CREATE TABLE IF NOT EXISTS player_info (
        player_id INT PRIMARY KEY,
        nickname VARCHAR,
        real_name VARCHAR,
        age INT,
        country VARCHAR,
        current_team VARCHAR,
        teams SET<VARCHAR>,
        total_kills INT,
        total_deaths INT,
        headshot_percentage DOUBLE,
        maps_played INT,
        rounds_played INT,
        kills_to_death_diff VARCHAR,
        total_opening_kills INT,
        total_opening_deaths INT,
        opening_kill_ratio DOUBLE,
        opening_kill_rating DOUBLE,
        first_kill_in_won_rounds DOUBLE,
        rating DOUBLE,
        rifle_kills INT,
        sniper_kills INT,
        smg_kills INT,
        pistol_kills INT,
        grenade_kills INT,
        other_kills INT,
        damage_per_round DOUBLE,
        grenade_dmg_per_round DOUBLE,
        kills_per_round DOUBLE,
        assists_per_round DOUBLE,
        deaths_per_round DOUBLE,
        saved_by_teammate_per_round INT,
        saved_teammates_per_round INT,
        rounds_with_kills DOUBLE,
        team_win_percent_after_first_kill DOUBLE,
        zero_kill_rounds INT,
        one_kill_rounds INT,
        two_kill_rounds INT,
        three_kill_rounds INT,
        four_kill_rounds INT,
        five_kill_rounds INT
    );
```

Replicação

SimpleStrategy

Replica os dados nos próximos servidores sem levar em conta a topologia da rede. Em desenvolvimento não há problemas em utilizar esse tipo replicação

NetworkTopologyStrategy

Muito mais complexo, pois permite configurar réplicas por **racks**. Em termos de durabilidade e alta disponibilidade esse tipo de replicação é indispensável.

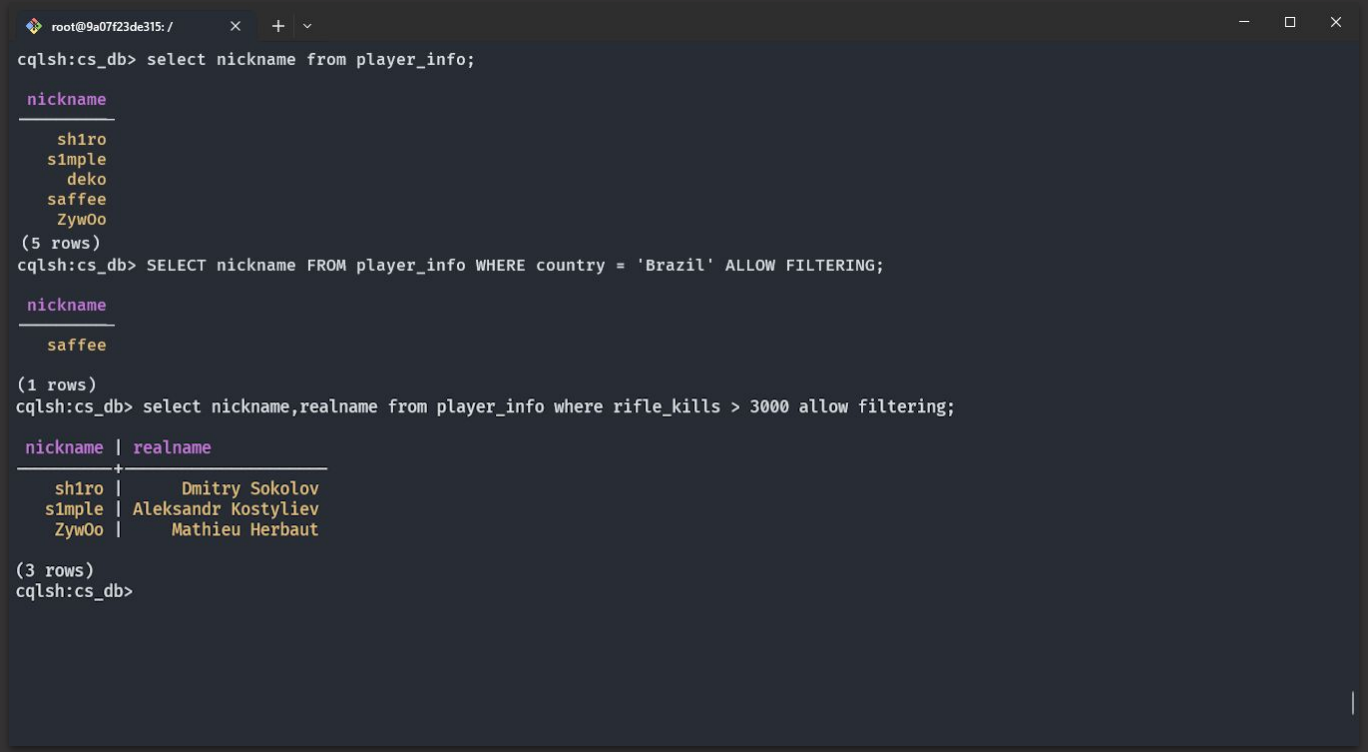
Fator

Determina a quantas de cópias(réplicas) de cada informação existente em um cluster, garantindo avaliabilidade e tolerância a problemas.

INSERT VALUES

```
INSERT INTO player_info (
    player_id,
    nickname,
    real_name,
    age,
    country,
    current team,
    teams,
    total_kills,
    total_deaths,
    headshot_percentage,
    damage_per_round,
    grenade_dmg_per_round,
    maps_played,
    rounds_played,
    kills_to_death_diff,
    kills_per_round,
    assists_per_round,
    deaths_per_round,
    saved_by_teammate_per_round,
    saved_teammates_per_round,
    rounds_with_kills,
    kill_death_diff,
    total_opening_kills,
    total opening deaths,
    opening_kill_ratio,
    opening_kill_rating,
    team_win_after_first_kill,
    first_kill_in_won_rounds,
    zero_kill_rounds,
    one_kill_rounds,
    two_kill_rounds,
    three_kill_rounds,
    four_kill_rounds,
    five_kill_rounds,
    rifle_kills,
    sniper_kills,
    smg_kills,
    pistol_kills,
    grenade_kills,
    other_kills,
    rating
) VALUES
(
        18835,
        'saffee',
        'Rafael Costa',
        27, 'Brazil',
        'FURIA',
        {'FÚRIA, paiN'},
        8482,
        6517,
        30.9,
        80.2,
        3.0,
        404,
        10701,
        1.3,
        0.79,
        0.1,
        0.61,
        0.08,
        0.1,
        5488,
        1965,
        1309,
        892,
        1.47,
        1.14,
        76.1,
        17.3,
        5213,
        3288,
        1548,
        524,
        114,
        14,
        2545,
        4412,
        264,
        1209,
        52,
        28,
        1.22
);
```

SELECT


root@9a07f23de315: /
cqlsh:cs_db> select nickname from player_info;
nickname
sh1ro
s1mple
deko
saffee
ZywOo
(5 rows)
cqlsh:cs_db> SELECT nickname FROM player_info WHERE country = 'Brazil' ALLOW FILTERING;
nickname
saffee
(1 rows)
cqlsh:cs_db> select nickname,realname from player_info where rifle_kills > 3000 allow filtering;
nickname | realname
sh1ro | Dmitry Sokolov
s1mple | Aleksandr Kostyliev
ZywOo | Mathieu Herbaut
(3 rows)
cqlsh:cs_db>

OBRIGADO!!!