



Olá, aluno(a)!
Seja bem-vindo(a) à aula interativa!

Você entrará na reunião com a câmera e o microfone desligados.

Sua presença será computada através da enquete.
Fique atento(a) e não deixe de respondê-la!

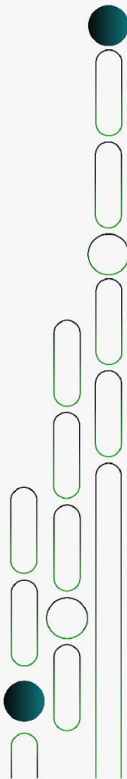


Aprenda com quem faz

> Performance & Otimização

Segunda Aula Interativa

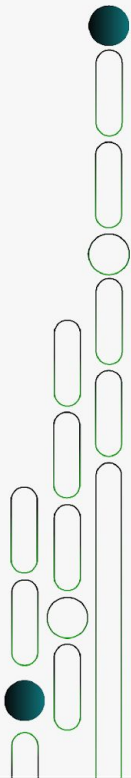
Prof. Adriano G. D. Ferreira





Nesta aula

- ❑ Centro de Desempenho SQL Server
- ❑ Export e Import Wizard do SQL Server
- ❑ Análise do Plano de Execução de Query e Operadores
- ❑ Dicas de Otimização
- ❑ Sobre o Desafio





Centro de Desempenho SQL Server

The screenshot shows the Microsoft Docs website for the 'Performance Center for SQL Server Database Engine and Azure SQL Database'. The page is in Portuguese and includes a sidebar with navigation links, a main content area with an introduction and a table of configuration options, and a footer with the URL.

Microsoft Docs | Documentação | Aprender | Q&A | Exemplos de Código

Documentos do SQL | Visão geral | Instalar | seguro | Desenvolver | Gerir | Analisar | Referência

Versão
SQL Server 2019
Filtrar por título

servidor SQL
Dicas de navegação de documentos
Versões anteriores 2005-2014

- Visão geral
- Clusters de Big Data
- Continuidade de negócios
- Design do banco de dados
- Desenvolvimento
- Recursos internos e arquitetura
- Instalação
- Migrar e carregar dados

Gerenciar, monitorar e ajustar

- Gerenciar
- Monitoramento
- Ajustar

Conceitos

- Ajuste automático
- Estimativa de cardinalidade
- Considerações para usar servidores de teste
- Processamento de consulta inteligente

Centro de desempenho

- Como fazer
- Assistente de Ajuste do Banco de Dados
- Planos de execução
- OLTP na memória

Performance Center para SQL Server Database Engine e Azure SQL Database

12/11/2018 • 3 minutos para o fim da leitura

Aplica-se a: SQL Server (todas as versões com suporte) Banco de dados SQL do Azure

Esta página fornece links para ajudá-lo a localizar as informações necessárias sobre o desempenho no Mecanismo de Banco de Dados do SQL Server e no Banco de Dados SQL do Azure.

lenda

- Indicates that the feature is available only in SQL Server Database Engine (both SQL Server running on-premises and SQL Server running in an Azure Virtual Machine).
- Indicates that the feature is available only in Azure SQL Database.
- Indicates that the feature is available in both SQL Server and Azure SQL Database.

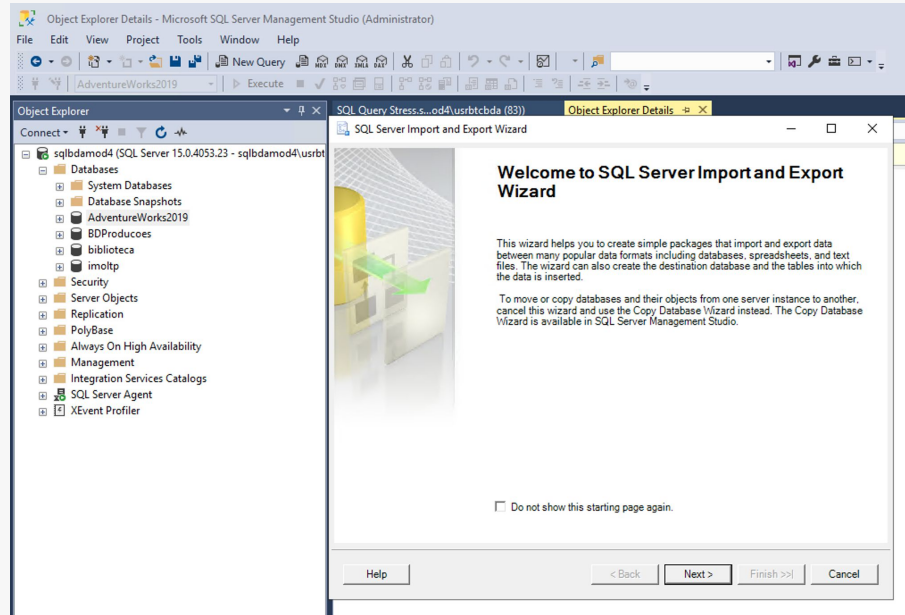
Opções de configuração para desempenho

O SQL Server oferece a capacidade de afetar o desempenho do mecanismo de banco de dados por meio de várias opções de configuração no nível do Mecanismo de banco de dados do SQL Server. Com o Banco de Dados SQL do Azure, a Microsoft realiza a maioria, mas não todas, essas otimizações para você.

Opções de configuração de disco	<input type="checkbox"/> Distribuição de disco e RAID
Opções de configuração de arquivos de dados e log	<input type="checkbox"/> Colocar dados e arquivos de registro em unidades separadas Exibir ou alterar os locais padrão para dados e arquivos de registro (SQL Server Management Studio)
Opções de configuração de TempDB	<input type="checkbox"/> Melhorias de desempenho na configuração do mecanismo de banco de dados TempDB - TempDB usando SSDs em VMs do Azure para armazenar o TempDB do SQL Server e o disco de extensões de pool de buffer e práticas recomendadas de desempenho para disco temporário para SQL Server em máquinas virtuais do Azure

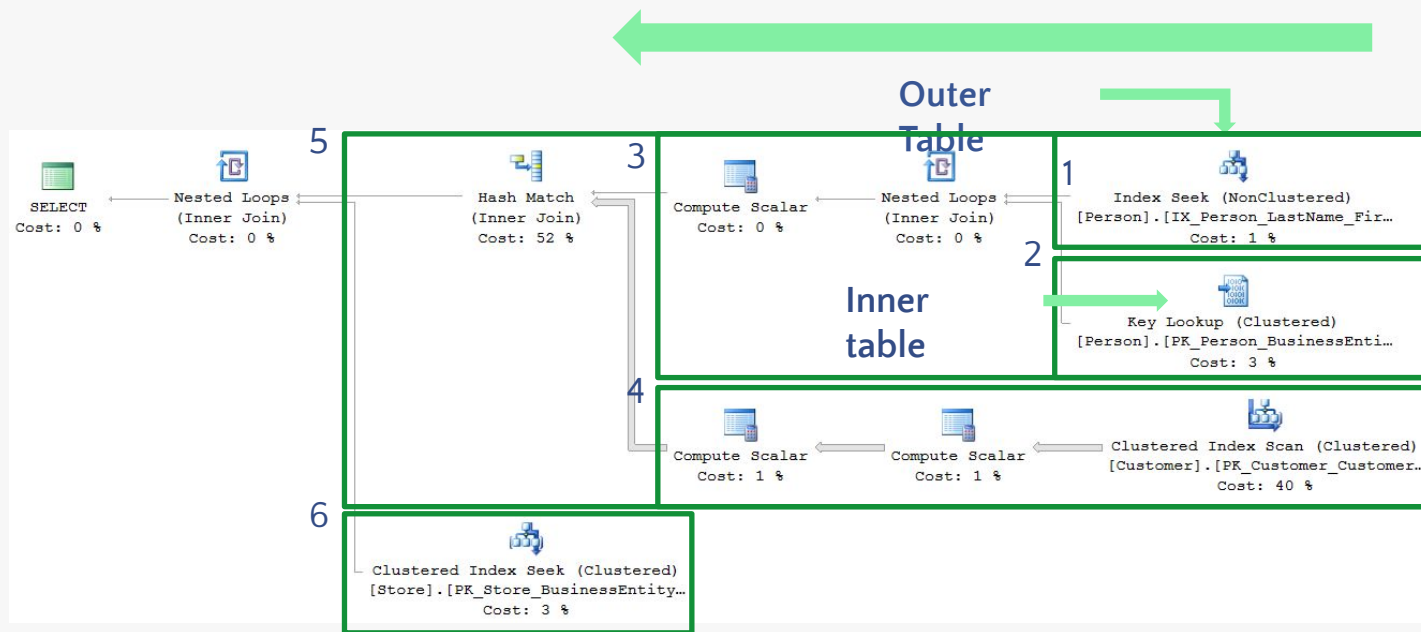


Export e Import Wizard





Análise do Plano de Execução (SQL)



Operadores

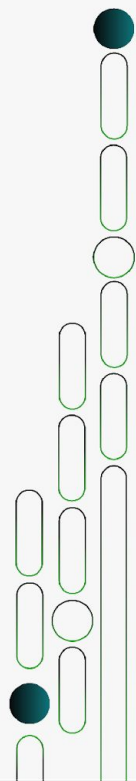
- **Operadores Seek**

- ✓ *Index Seek*

- ✓ *Clustered Index Seek*



- O operador *Clustered Index Seek* usa a capacidade dos índices para recuperar linhas de um índice clusterizado (clustered index);
 - O operador *Index Seek* usa a capacidade de busca de índices para recuperar linhas de um índice não clusterizado.



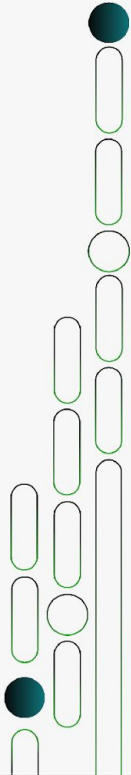
Operadores

- **Operadores Lookup**

- ✓ *Row Identifier (RID) Lookup*

- ✓ *Key Lookup*

- *RID Lookup* é uma pesquisa em uma tabela **HEAP** usando um identificador de linha fornecido (RID) para pesquisar a linha na tabela e o nome da tabela na qual a linha é pesquisada.
 - *Key Lookup* é uma pesquisa em uma tabela com um índice clusterizado. É sempre acompanhada por um operador de loops aninhados. Esse operador pode indicar que a consulta pode se beneficiar se for criado um índice mais abrangente.



Operadores

- **Operadores Scan**

✓ *Table Scan*

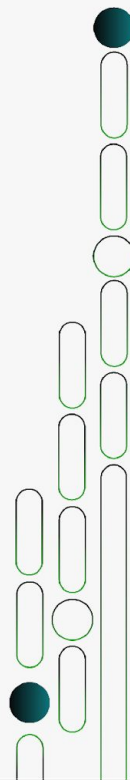
✓ *Clustered Index Scan*

✓ *Index Scan*

✓ *Columnstore Index Scan*



- O operador *Table scan* percorre todas as linhas da tabela para retornar as linhas desejadas na query.
- Demais operadores percorrem toda a estrutura do índice para retornar as linhas desejadas na query.



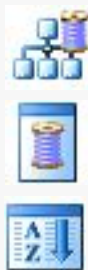
Operadores

- **Outros operadores**

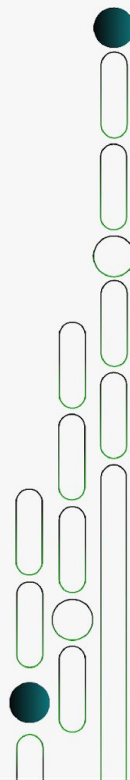
✓ *Table Spool*

✓ *Index Spool*

✓ *Sort*



- *Table Spool* coloca uma cópia de cada linha em uma tabela de spool oculta que é armazenada no banco de dados *tempdb* e existe apenas durante o tempo de vida da query.
- *Index Spool* coloca uma cópia de cada linha em um arquivo de spool oculto (armazenado no *tempdb* e existindo apenas durante o tempo de vida da consulta) e cria um índice não cluster ☐ permite usar a capacidade de busca de índices (*Index Seek*).





Dicas para Otimizar Plano de Query

Revisar e otimizar os JOINS

- Nested loops apenas para conjuntos pequenos de dados.

Rever operadores SCANS e RID

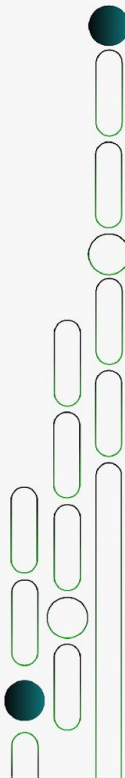
- Removê-los usando índices que contemplem toda a query.

Rever operadores Key Lookups

- Removê-los usando índices que contemplem toda a query.

Outras considerações

- Eliminar spools.





Dicas para Otimizar Plano de Query

Limitar a quantidade de JOINS

Limitar a quantidade de linhas nos JOINS

Criar índices nas colunas dos JOINS

Fazer JOINS usando as colunas mais únicas

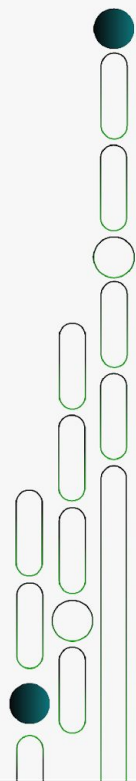
Fazer JOINS com colunas do mesmo tipo de dados (evitar conversão implícita)

Evitar usar SELECT *

- Sempre selecione explicitamente as colunas que você deseja retornar.

Usar a sintaxe ANSI nos JOINS

- SELECT fname, lname, department
FROM names **INNER JOIN** departments
ON names.employeeid = departments.employeeid





Dicas para Otimizar Plano de Query

Evitar lógicas negativas, como !=, <>, NOT (...)

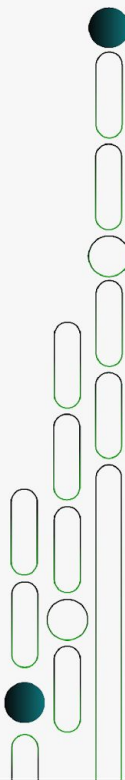
- Isso introduz contenção adicional, porque muitas vezes resulta na avaliação de cada linha (index scan).

Só usar ORDER BY se for estritamente necessário

- Se estiver coberto por um índice, garanta que a ordenação do índice é a mesma da cláusula ORDER BY.

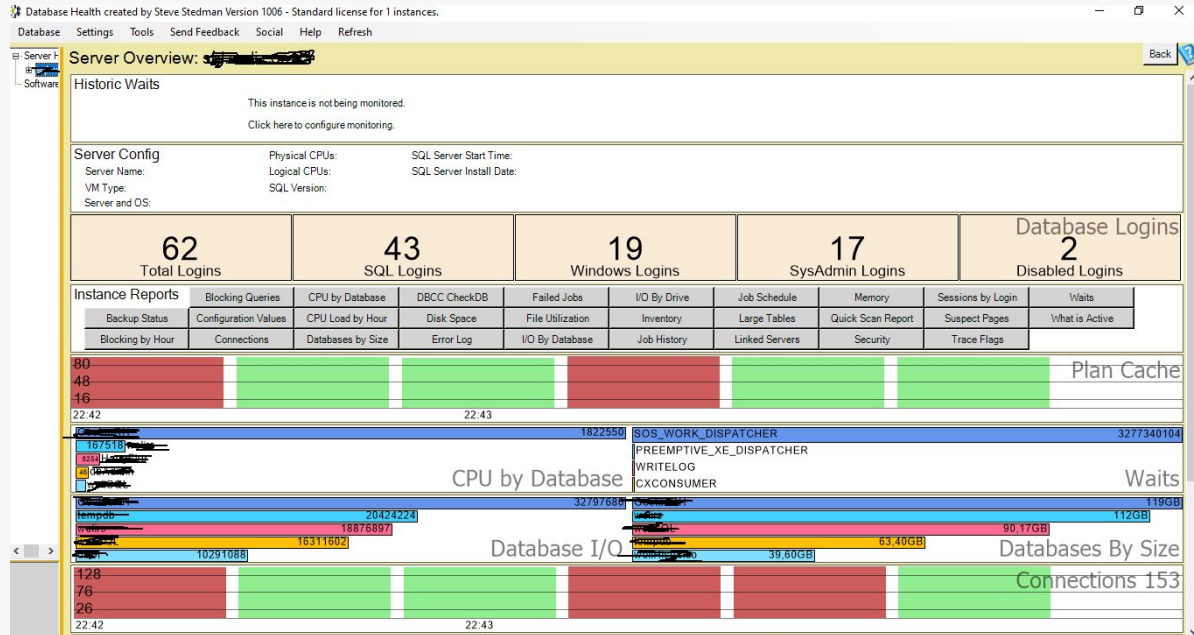
Evitar o operador LIKE com caracteres curingas ('%VALOR%'), pois sempre causará um table scan

- Se tiver que usar LIKE, torne o primeiro caracter um literal ('VALOR%').





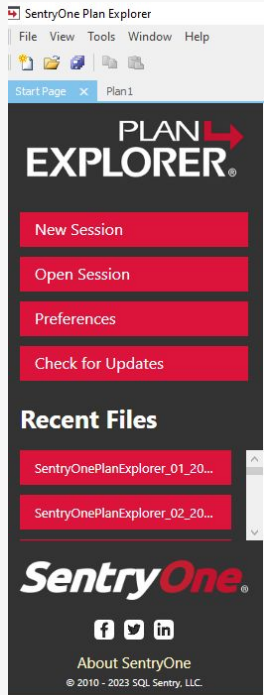
DBs Relacionais: Performance



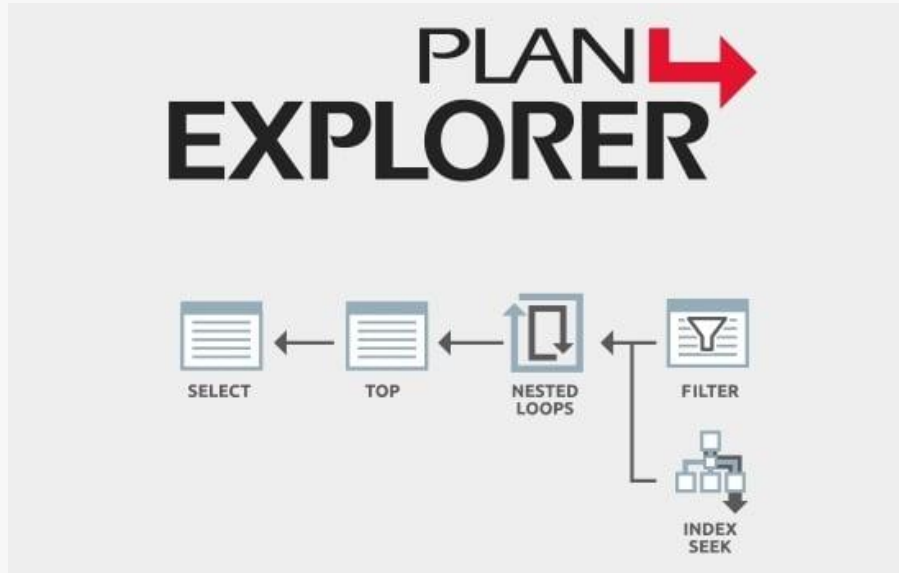
Fonte: <http://databasehealth.com/>



Dbs Relacionais: Performance



- Integrado ao Plano de Execução (Execution Plan) do Sql Server.



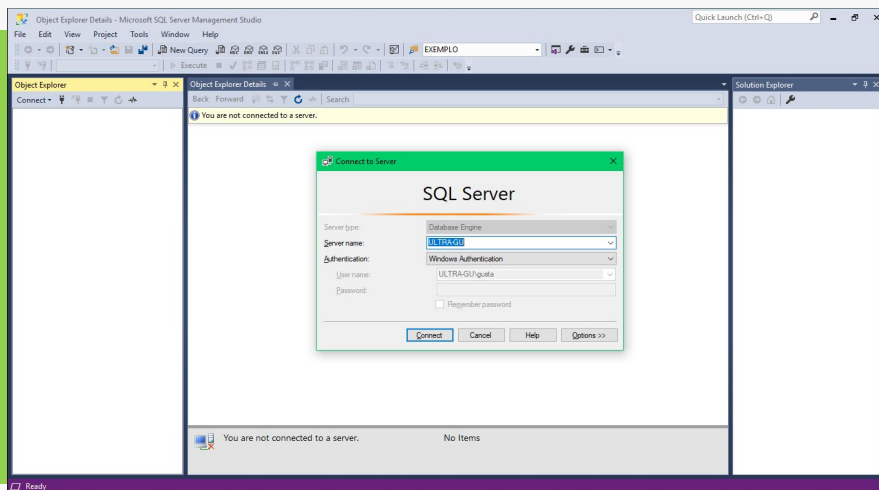
<https://www.sentryone.com/plan-explorer>



Análise do Plano de Execução



Demo





Primeiro: Criar o banco 'Desafio' e fazer as cargas:

The screenshot shows the Microsoft SQL Server Management Studio interface. The left pane displays the 'Object Explorer' with a tree view of databases under the 'adrianoogdf' server. The right pane shows a script titled 'ScriptCriandoDbPr...OGDPAdriano (55)' with the following SQL code:

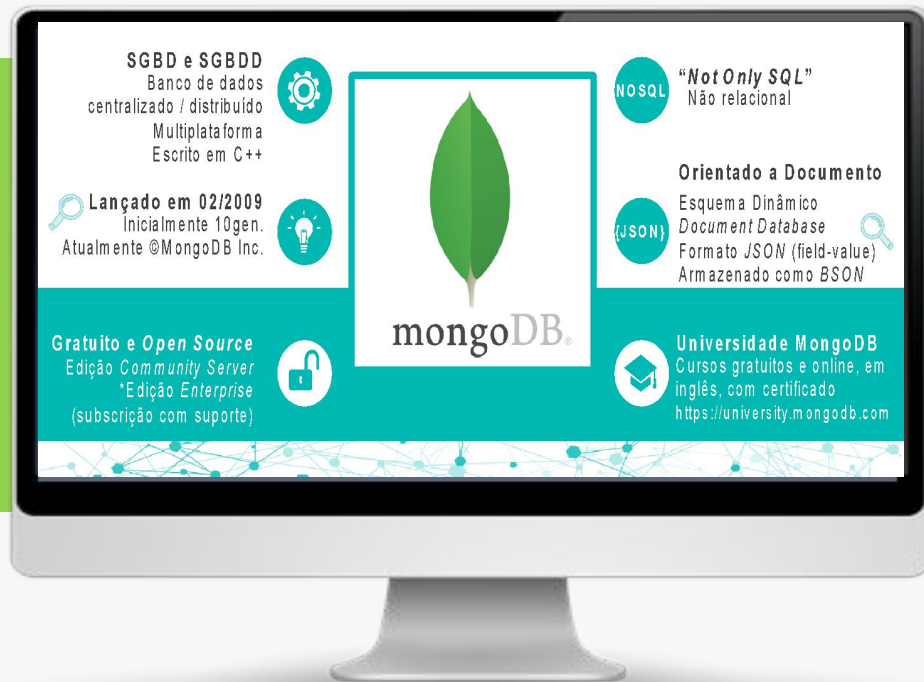
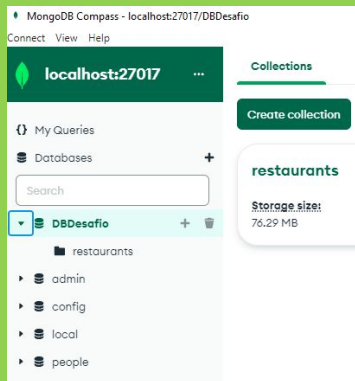
```
-- ScriptDesafio Aula.04 - Performance e Otimização:
-- Pergunta.01
CREATE DATABASE BDProducoes
GO
USE [BDProducoes]
GO

CREATE TABLE [dbo].[Autoria](
    [cod_Autoria] [int] IDENTITY(1,1) NOT NULL,
    [cod_titulo] [int] NOT NULL,
    [cod_pessoa] [int] NOT NULL,
    CONSTRAINT [PK_Autoria] PRIMARY KEY CLUSTERED
    (
        [cod_Autoria] ASC
    )
)
GO
--
CREATE TABLE [dbo].[Avaliacao](
    [cod_titulo] [int] NOT NULL,
    [classificacao_media] [int] NOT NULL,
    [qtd_votos] [int] NOT NULL
)
GO
CREATE TABLE [dbo].[Direcao](
```

The status bar at the bottom indicates 'Connected. (1/1)' and 'adrianoogdf (15.0 RTM)'.



Análise do Plano de Execução





Análise do Plano de Execução (MongoDB)

```
{
  "queryPlanner" : {
    "plannerVersion" : 1,
    "namespace" : "test_blog.users",
    "indexFilterSet" : false,
    "parsedQuery" : {
      "username" : {
        "$eq" : "USER_9"
      }
    },
    "winningPlan" : {
      "stage" : "COLLSCAN",
      "filter" : {
        "username" : {
          "$eq" : "USER_9"
        }
      },
      "direction" : "forward"
    },
    "rejectedPlans" : []
  },
  "serverInfo" : {
    "host" : "Laptop",
    "port" : 27017,
    "version" : "3.4.4",
    "gitVersion" : "888390515874a9debd1b6c5d36559ca86b44babd"
  },
  "ok" : 1
}
```

← The query

← Query stage

← Direction of query

← Server Info



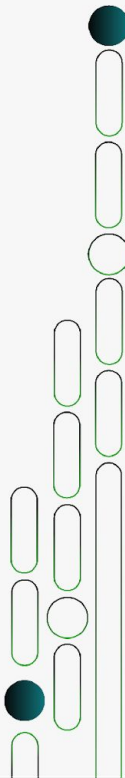
Dicas para Otimizar Plano de Query (MongoDB)

Ordem dos Campos no Índice:

- **1º. Igualdade**
- **2º. Ordenação**
- **3º. Range**

```
exp.find({ "address.zipcode": { $gt: '50000' }, cuisine: 'Sushi' }).sort({ stars: -1 })
```

```
db.restaurants.createIndex({ "cuisine": 1, "stars": 1, "address.zipcode": 1 })
```





Explain Query (MongoDB Compass)

Local

4 DBS 3 COLLECTIONS

FAVORITE

HOST
localhost:27017

CLUSTER
Standalone

EDITION
MongoDB 4.0.10 Community

Filter your data

admin

config

local

test

restaurant

restaurants

test.restaurants

Explain Plan

DOCUMENTS 1.0m

TOTAL SIZE 173.0MB

AVG. SIZE 181B

INDEXES 2

TOTAL SIZE 14.6MB

AVG. SIZE 7.3MB

Documents

Aggregations

Schema

Explain Plan

Indexes

Validation

FILTER { 'cuisine': 'Sichuan' }

OPTIONS

EXPLAIN

RESET

VIEW DETAILS AS

VISUAL TREE

RAW JSON

Query Performance Summary

Documents Returned: 23482

Actual Query Execution Time (ms): 1656

Index Keys Examined: 0

Sorted in Memory: no

Documents Examined: 1000000

No index available for this query.

COLLSCAN

nReturned: 23482

Execution Time: 1454 ms

Documents Examined: 1000000

DETAILS



Explain Query (MongoDB Compass)

MongoDB Compass - localhost:27017/DBDesafio.restaurants

Connect View Help

localhost:27017 Explain Plan DBDesafio.restau...

My Queries

Databases

Search

DBDesafio

restaurants

admin

config

local

people

DBDesafio.restaurants

1.0m 1
DOCUMENTS INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

Filter



MongoDB Compass - localhost:27017/DBDesafio.restaurants

Connect View Collection Help

localhost:27017 Documents DBDesafio.restau...

My Queries Databases Search

DBDesafio restaurants

admin config local people

DBDesafio.restaurants

1.0m 1 DOCUMENTS INDEXES

Documents Aggregations Schema Explain Plan Indexes Validation

Filter Type a query: { field: 'value' } Reset Find More Options

ADD DATA EXPORT COLLECTION 1 - 20 of 1000000

```
{
  "_id": ObjectId('63db19cf51aa3c4b172bd3dd'),
  "name": "Chefs Table at the Edgewater",
  "cuisine": "Italian",
  "stars": 1.6,
  "address": Object
}
```

```
{
  "_id": ObjectId('63db19cf51aa3c4b172bd3de'),
  "name": "Altius",
  "cuisine": "Japanese",
  "stars": 3.1,
  "address": Object
}
```

```
{
  "_id": ObjectId('63db19cf51aa3c4b172bd3df'),
  "name": "Red Hill Station",
  "cuisine": "Mediterranean",
  "stars": 3.4
}
```

>_MONGOOSH



Export Collections

```
Prompt de Comando
17/01/2023 16:08      34.790.912 mongos.exe
17/01/2023 16:08      471.257.088 mongos.pdb
03/11/2022 12:25      22.986.864 mongostat.exe
03/11/2022 12:25      22.400.684 mongotop.exe
15 arquivo(s) 1.647.854.120 bytes
3 pasta(s) 744.039.383.040 bytes disponíveis

C:\Program Files\MongoDB\Server\6.0\bin>mongodbinport
'mongodbinport' não é reconhecido como um comando interno
ou externo, um programa operável ou um arquivo em lotes.

C:\Program Files\MongoDB\Server\6.0\bin>mongoimport
2023-02-01T22:54:43.030-0300 no collection specified
2023-02-01T22:54:43.162-0300 using filename '' as collection
2023-02-01T22:54:43.163-0300 error validating settings: invalid collection name: collection name cannot be an empty string

C:\Program Files\MongoDB\Server\6.0\bin>mongoimport --drop --collection=restaurants --db=DBDesafio --file="D:\_BootCampAnalistaDeDados\MongoDbsImports\restaurants.json"
2023-02-01T23:02:54.950-0300 connected to: mongodb://localhost/
2023-02-01T23:02:54.987-0300 dropping: DBDesafio.restaurants
2023-02-01T23:02:57.952-0300 [#####] DBDesafio.restaurants 16.5MB/144MB (11.5%)
2023-02-01T23:03:00.950-0300 [#####] DBDesafio.restaurants 35.8MB/144MB (24.9%)
2023-02-01T23:03:03.950-0300 [#####] DBDesafio.restaurants 56.0MB/144MB (39.0%)
2023-02-01T23:03:06.950-0300 [#####] DBDesafio.restaurants 76.0MB/144MB (52.9%)
2023-02-01T23:03:09.950-0300 [#####] DBDesafio.restaurants 95.7MB/144MB (66.6%)
2023-02-01T23:03:12.951-0300 [#####] DBDesafio.restaurants 115MB/144MB (79.8%)
2023-02-01T23:03:15.950-0300 [#####] DBDesafio.restaurants 134MB/144MB (93.0%)
2023-02-01T23:03:17.433-0300 [#####] DBDesafio.restaurants 144MB/144MB (100.0%)
2023-02-01T23:03:17.433-0300 1000000 document(s) imported successfully. 0 document(s) failed to import.

C:\Program Files\MongoDB\Server\6.0\bin>
```

`mongoimport --drop --collection=restaurants --db=DBDesafio --file="D:_BootCampAnalistaDeDados\MongoDbsImports\restaurants.json"`

`mongoimport --drop --collection=people --db=people --file="D:_BootCampAnalistaDeDados\MongoDbsImports\people.json"`



Verificando indexação DBDesafio criado:

```
mongosh
Current sessionID: 87e71cb81785bfb636994a2c
Connecting to:   mongodb://127.0.0.1:27017
Using MongoDB:  4.4.1
Using Mongosh Beta: 0.5.0

For more information about mongosh, please see our docs: https://docs.mongodb.com/mongodb-shell/

> show dbs
BDDesafio      33.1 MB
BDModulo2      8.19 kB
Configuracoes  279 kB
Graduacao      8.19 kB
admin          147 kB
config         36.9 kB
local          73.7 kB
test           101 MB

> use BDDesafio
switched to db BDDesafio
> db.restaurants.find({'cuisine': 'Sichuan', stars : { $gt : 4 } }).sort({ stars: -1 }).explain()
```



Verificando indexação DBDesafio: Questão

11:

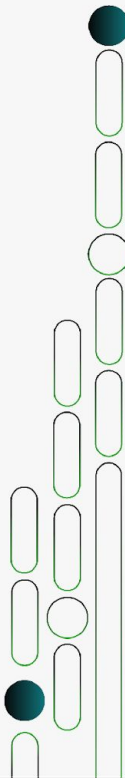
```
> use BDDesafio
switched to db BDDesafio
> db.restaurants.find({'cuisine': 'Sichuan', stars : { $gt : 4 } }).sort({ stars: -1 }).explain()
{
  queryPlanner: {
    plannerVersion: 1,
    namespace: 'BDDesafio.restaurants',
    indexFilterSet: false,
    parsedQuery: {
      '$and': [ { cuisine: { '$eq': 'Sichuan' } }, { stars: { '$gt': 4 } } ]
    },
    winningPlan: {
      stage: 'SORT',
      sortPattern: { stars: -1 },
      memLimit: 104857600,
      type: 'simple',
      inputStage: {
        stage: 'COLLSCAN',
        filter: { '$and': [ { cuisine: [Object] }, { stars: [Object] } ] },
        direction: 'forward'
      }
    },
    rejectedPlans: []
  },
  serverInfo: {
    host: 'SERVER-GU',
    port: 27017,
    version: '4.4.1',
    gitVersion: 'ad91a93a5a31e175f5cbf8c69561e788bbc55ce1'
  },
  ok: 1
}
```



Verificando indexação DBDesafio: Questão 11: Índices

```
> db.restaurants.createIndex({ "cuisine": 1 , "stars" : -1 })_
```

índice com 'cuisine' e 'stars' -1, para tratar um 'CollScan' ('Collection Scan').





Verificando indexação DBDesafio: Questão 11: Índices

```
> db.restaurants.find({ cuisine: 'Sichuan', stars: { $gt: 4 } }).sort({ stars: -1 }).explain()
{
  queryPlanner: {
    plannerVersion: 1,
    namespace: 'DBDesafio.restaurants',
    indexFilterSet: false,
    parsedQuery: {
      '$and': [ { cuisine: { '$eq': 'Sichuan' } }, { stars: { '$gt': 4 } } ]
    },
    winningPlan: {
      stage: 'FETCH',
      inputStage: {
        stage: 'IXSCAN',
        keyPattern: { cuisine: 1, stars: -1 },
        indexName: 'cuisine_1_stars_-1',
        isMultiKey: false,
        multiKeyPaths: { cuisine: [], stars: [] },
        isUnique: false,
        isSparse: false,
        isPartial: false,
        indexVersion: 2,
        direction: 'forward',
        indexBounds: {
          cuisine: [ ['Sichuan', 'Sichuan'] ],
          stars: [ ['inf.0', 4] ]
        }
      }
    },
    rejectedPlans: []
  },
  serverInfo: {
    host: 'SERVER-GU',
    port: 27017,
    version: '4.4.1',
    gitVersion: 'ad91a93a5a31e175f5cbf8c69561e788bbc55ce1'
  },
  ok: 1
}
```

Com o índice, tenho agora um 'IXSCAN', que é situação melhor que 'CollScan'.
O índice trata primeiro a condição de igualdade, depois o 'sort' e a situação de Range.



Verificando indexação DBDesafio: Questão 12: Índices

```
> db.restaurants.createIndex({ "cuisine": 1 })
{
  createdCollectionAutomatically: false,
  numIndexesBefore: 1,
  numIndexesAfter: 2,
  ok: 1
}
> db.restaurants.createIndex({ "stars" : -1 })
{
  createdCollectionAutomatically: false,
  numIndexesBefore: 2,
  numIndexesAfter: 3,
  ok: 1
}
>
```

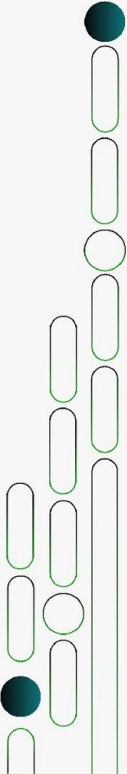
A criação de dois índices para tratar primeiro o filtro: 'cuisine: 'Sushi' e depois o sort: 'stars: 1'.
Tratando duas operações com índices diferentes e não a partir de um índice composto.



Verificando indexação DBDesafio: Questão 12: Índices

```
mongosh
{
  stage: 'FETCH',
  filter: { cuisine: { '$eq': 'Sushi' } },
  inputStage: {
    stage: 'IXSCAN',
    keyPattern: { stars: -1 },
    indexName: 'stars_-1',
    isMultiKey: false,
    multiKeyPaths: { stars: [] },
    isUnique: false,
    isSparse: false,
    isPartial: false,
    indexVersion: 2,
    direction: 'forward',
    indexBounds: { stars: [ '[MaxKey, MinKey]' ] }
  },
  rejectedPlans: [
    {
      stage: 'SORT',
      sortPattern: { stars: -1 },
      memLimit: 104857600,
      type: 'simple',
      inputStage: {
        stage: 'FETCH',
        inputStage: {
          stage: 'IXSCAN',
          keyPattern: { cuisine: 1 },
          indexName: 'cuisine_1',
          isMultiKey: false,
          multiKeyPaths: { cuisine: [] },
          isUnique: false,
          isSparse: false,
          isPartial: false,
          indexVersion: 2,
          direction: 'forward',
          indexBounds: { cuisine: [Array] }
        }
      }
    }
  ],
  serverInfo: {
    host: 'SERVER-GU',
    port: 27017,
    version: '4.4.1',
    gitVersion: 'ad91a93a5a1e179f9cbf8c69561e7800bc55ce1'
  },
  ok: 1
}
```

- Os dois índices criados resolveram a situação, agora com o 'IXScan'.
- Importante testar e validar sempre.
- Fica mais barato fazer o 'fetch' do que fazer 'sort' em memória, não precisando de um índice composto para isto.
- Precisa da análise em conjunto das 'queries' envolvidas.

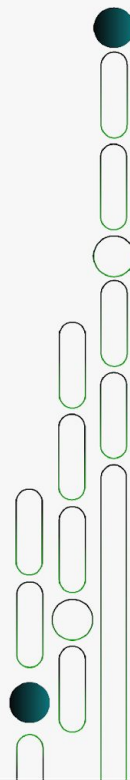




Verificando indexação DBDesafio: Sort: name

```
> db.restaurants.find({ "stars": { $gte: 4 }, cuisine: 'Italian' }).sort({ name: 1 }).explain()
```

‘find’ por cozinha ‘cuisine’ e ordenação ‘sort’ por nome ‘name’.



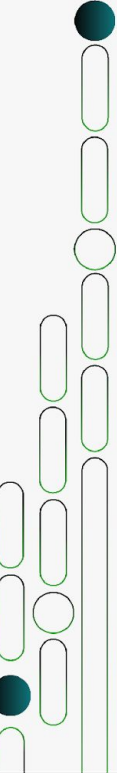


Verificando indexação DBDesafio: Sort: name

```
> db.restaurants.find({ "stars": { $gte: 4 }, cuisine: 'Italian' }).sort({ name: 1 }).explain()
{
  queryPlanner: {
    plannerVersion: 1,
    namespace: 'DBDesafio.restaurants',
    indexFilterSet: false,
    parsedQuery: {
      '$and': [ { cuisine: { '$eq': 'Italian' } }, { stars: { '$gte': 4 } } ]
    },
    winningPlan: {
      stage: 'SORT',
      sortPattern: { name: 1 },
      memLimit: 104857600,
      type: 'simple',
      inputStage: {
        stage: 'FETCH',
        filter: { stars: { '$gte': 4 } },
        inputStage: {
          stage: 'IXSCAN',
          keyPattern: { cuisine: 1 },
          indexName: 'cuisine_1',
          isMultiKey: false,
          multikeyPaths: { cuisine: [] },
          isUnique: false,
          isSparse: false,
          isPartial: false,
          indexVersion: 2,
          direction: 'forward',
          indexBounds: { cuisine: [ '['Italian', 'Italian']' ] }
        }
      }
    },
    rejectedPlans: [
      {
        stage: 'SORT',
        sortPattern: { name: 1 },
        memLimit: 104857600,
        type: 'simple',
        inputStage: {
          stage: 'FETCH',
          filter: { cuisine: { '$eq': 'Italian' } },
          inputStage: {
            stage: 'IXSCAN',
            keyPattern: { stars: -1 },
            indexName: 'stars_1',
            isMultiKey: false,
            multikeyPaths: { stars: [] },

```

Índice criado anteriormente ajudando 'query' com o 'IxScan'.



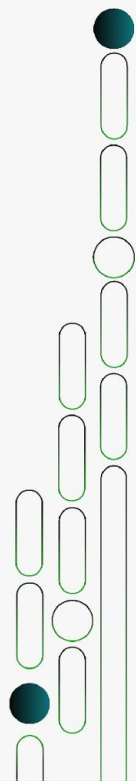


Listando índices criados no MongoDB:

```
> db.restaurants.getIndexes()
[
  { v: 2, key: { _id: 1 }, name: '_id_' },
  { v: 2, key: { cuisine: 1 }, name: 'cuisine_1' },
  { v: 2, key: { stars: -1 }, name: 'stars_-1' }
]
>
```

Comando para verificar índices criados.

Um pelo campo id e dois utilizados para filtrar pelo campo 'name'.





Sobre o Desafio

Objetivos

Exercitar os seguintes conceitos trabalhados no Módulo:

- Análise e otimização de plano de queries no SQL Server e MongoDB
- Cargas e expurgos massivos de dados no SQL Server e MongoDB

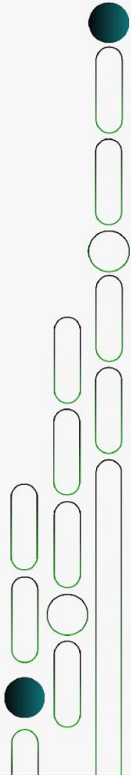
Enunciado

O projeto para o qual você foi contratado no Módulo 2 entra agora em sua segunda fase, onde será feita a otimização das 'queries' que foram implementadas no 'SQL Server'.

Atividades

1. Criar o banco **BDProducoes** e seu schema físico usando o script abaixo;

...





Adriano G. Dias Ferreira
DBA Sr | Data Administrator | Data Architect | Cloud Computing |
Data Quality | Data Analyst | BI | DW | Coordenador e Professor de
Graduação e Pós-Graduação

Linked-in: <https://www.linkedin.com/in/adrianogdiasferreira>

Lattes: <http://lattes.cnpq.br/3393239069676217>

Email: adrianogdf@gmail.com

**Obrigado pela
participação!**

Bons estudos!

Sucesso!

