Trabalho 01
Pesquisa e Ordenação de Dados

Nome: Paulina Rehbein Matrícula: 202012842

Linguagens sorteadas: JavaScript, Go;
Banco de Dados sorteados: IBM DB2, MariaDB.

Linguagem Go:
A linguagem Go tem o método ***sort()*** para ordenar os dados de um array.

Um exemplo da implementação usando um *array* e o *sort()* para ordenar:

```go

package main

import (
    "fmt"
    "sort"
)

func main() {
    fmt.Println("Go Sorting Tutorial")

    myInts := []int{1,3,2,6,3,4}
    fmt.Println(myInts)

    // em Go é preciso usar sort.TipoDadosArray() ex:
    sort.Ints(myInts)
    fmt.Println(myInts)
}

```

O resultado no terminal é o seguinte:

```
Go Sorting Tutorial
[1 3 2 6 3 4]
[1 2 3 3 4 6]


...Program finished with exit code 0
Press ENTER to exit console.
```

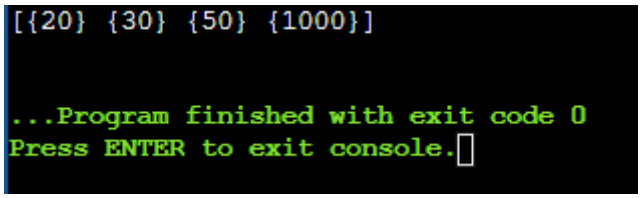
Caso o desenvolvedor queira customizar o método *sort()* ele pode usar as funções:

- **Len()** - retorna o tamanho do *array* de itens;
- **Swap()** - Uma função que troca a posição de dois elementos em um *array*;
- **Less()** - uma função que retorna um valor booleano dependendo se o item na posição i é menor que o item na posição j.

Exemplo do uso de ***Len(), Swap() e Less()***:

```go
package main

import (
    "fmt"
    "sort"
)

type Programmer struct {
    Age int
}

type byAge []Programmer

func (p byAge) Len() int {
    return len(p)
}

func (p byAge) Swap(i, j int) {
    p[i], p[j] = p[j], p[i]
}

func (p byAge) Less(i, j int) bool {
    return p[i].Age < p[j].Age
}

func main() {
    programmers := []Programmer{
        Programmer{Age: 30,},
        Programmer{Age: 20,},
        Programmer{Age: 50,},
        Programmer{Age: 1000,},
    }

    sort.Sort(byAge(programmers))

    fmt.Println(programmers)
}
```

Resultado:



A função ***sort()*** da Linguagem Go pode ser encontrada na documentação:
https://go.dev/src/sort/sort.go

Mais explicações sobre o ***sort()*** do Go:
https://pkg.go.dev/sort

Linguagem JavaScript:

O ***JavaScript*** usa a função ***sort()*** para ordenar dados, os elementos do array são classificados. A ordenação não é necessariamente estável (ou seja, elementos que se comparam iguais não permanecem necessariamente em sua ordem original).
Sintaxe da função:

```
arr.sort([funcaoDeComparacao]);
```

A função aceita um parâmetro que deve ser uma função que define a ordenação. **Se esse parâmetro for omitido, o array é ordenado de acordo com a pontuação de código Unicode de cada um dos caracteres, de acordo com a conversão de cada elemento para string.** Ou seja, se a função receber um array de números irá convertê-los para *String* e depois ordenar.
Link para ver na prática:
<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort>

A funcaoDeComparacao deve ser uma função que aceita dois argumentos x e y e retorna um valor negativo se x < y, zero se x = y, ou um valor positivo se x > y.

A ordenação é feita dentro do próprio array.
E é organizada em ordem crescente.

- **Qual tipo de ordenação é usada no JavaScript:**

Os dois tipos de algoritmos mais usados para ordenação genérica são Quick Sort e Merge Sort. O Quicksort geralmente é o mais rápido dos dois, mas o Merge

Sort tem algumas propriedades interessantes que podem torná-lo uma escolha geral melhor.

Seu interpretador JavaScript específico pode usar um desses algoritmos ou algo totalmente diferente. **O padrão ECMAScript não especifica qual algoritmo uma implementação em conformidade deve usar**. Até nega explicitamente a necessidade de estabilidade.

Documentação da função ***sort():***
https://262.ecma-international.org/5.1/#sec-15.4.4.11

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/sort

Sobre QuickSort:
https://en.wikipedia.org/wiki/Quicksort

Sobre Merge Sort:
https://en.wikipedia.org/wiki/Merge_sort

ECMAScript:
https://www.ecma-international.org/publications-and-standards/standards/ecma-262/

IBM DB12:
O IBM® DB2® Sort for z/OS® (também conhecido como DB2 Sort) é uma ferramenta DB2 que fornece processamento de classificação de utilitário de alta velocidade para dados armazenados em bancos de dados DB2 para z/OS.

O que o DB2 Sort faz?
O DB2 Sort pode melhorar o desempenho do processamento de classificação, especialmente em ambientes com grandes volumes de dados, grandes espaços de tabela ou grandes índices. Esses ambientes exigem uma abordagem de classificação mais sofisticada do que a abordagem usada por ferramentas usadas para fins de classificação geral.

Como o DB2 Sort funciona com utilitários e ferramentas do DB2
O DB2 Sort se comunica com utilitários e ferramentas do DB2 e, em seguida, aloca recursos do sistema dinamicamente para otimizar cada solicitação de classificação. Quando várias classificações são executadas em paralelo, o DB2 Sort escolhe o uso mais eficaz dos recursos para cada classificação, o que pode aumentar potencialmente o número de classificações que podem ser executadas simultaneamente. Durante a execução dessas classificações, o DB2 Sort também

monitora e ajusta a alocação de recursos do sistema para otimizar o processamento da CPU, o desempenho de E/S e o uso da memória.

Como as ordenações paralelas são tratadas

Quando os utilitários do DB2 requerem processamento de classificação, várias classificações de um único utilitário geralmente são executadas em paralelo para reduzir o tempo decorrido do aplicativo. Esse método de classificação é conhecido como **paralelismo intra-regional**.

O DB2 Sort se comunica com o utilitário DB2 para escolher o número ideal de classificações a serem executadas em paralelo.

O DB2 Sort avalia a disponibilidade de recursos do sistema e se comunica com o utilitário DB2 para otimizar a alocação de recursos para cada classificação com base nas características dos dados. Ao escolher o uso mais eficaz dos recursos para cada classificação, o DB2 Sort pode aumentar o número de classificações que podem ser executadas simultaneamente. Este maior grau de paralelismo pode resultar em menor tempo decorrido para a aplicação. O DB2 Sort também sincroniza o processamento de ordenações simultâneas, reduzindo o tempo de espera associado à transferência de registros entre os utilitários e o DB2 Sort. Essa sincronização reduz o tempo de CPU e o tempo decorrido do processamento de classificação. Os utilitários LOAD, CHECK INDEX, REBUILD INDEX e REORG TABLESPACE podem se beneficiar da maneira como o DB2 Sort trata as classificações paralelas.

Classificação de Dados:

- **Sorts for GROUP BY and ORDER BY**
  Essas classificações são indicadas por SORTC_ORDERBY e SORTC_GROUPBY em PLAN_TABLE.
  O desempenho da ordenação pela cláusula GROUP BY é aprimorado quando a consulta acessa uma única tabela e quando a coluna GROUP BY não possui índice.

- **Classifica para remover duplicatas**
  Esse tipo de classificação é usado para processar uma consulta com SELECT DISTINCT, com uma função definida como COUNT(DISTINCT COL1), ou para remover duplicatas no processamento de UNION. É indicado por SORTC_UNIQ em PLAN_TABLE.

- **Classificações usadas no processamento de junção**
  Para junção híbrida (MÉTODO 4) e junção de loop aninhado (MÉTODO 1), a tabela composta pode ser classificada para tornar a junção mais eficiente.

- **Classificações para processamento de subconsultas**
  Quando uma subconsulta IN ou NOT IN não correlacionada está presente na consulta, os resultados da subconsulta são classificados e colocados em um arquivo de trabalho para referência posterior pela consulta pai.

Links da documentação:
https://www.ibm.com/docs/en/db2-sort-for-zos/2.1.0?topic=SSLJYE_2.1.0/topics/cnkucon_overview_architecture.html

https://www.ibm.com/docs/en/db2-sort-for-zos/2.1.0?topic=SSLJYE_2.1.0/topics/cnkucon_overview.html

https://www.ibm.com/docs/en/db2/9.7?topic=explain-sort-input-argument

https://www.ibm.com/docs/en/db2-for-zos/12?topic=access-sorts-data

MariaDB
No banco de dados MariaDB Use a cláusula **ORDER BY** para ordenar um conjunto de resultados, como os retornados de uma instrução SELECT. Você pode especificar apenas uma coluna ou usar qualquer expressão com funções. Se você estiver usando a cláusula GROUP BY, poderá usar funções de agrupamento em ORDER BY. O pedido é feito após o agrupamento.

A ordem de classificação natural é a ordenação de strings em ordem alfabética, enquanto os números são tratados como números. Essa compreensão da classificação está mais próxima da compreensão humana do que de uma máquina. Você pode encontrar um exemplo desse recurso no gerenciador de arquivos do Windows. Lá os arquivos são classificados em ordem natural. Tente criar quatro pastas “b1”, “a11”, “a2”, “a1”.

Existem várias linguagens de programação que possuem ordenação natural. Em PHP é a função nativa natsort, enquanto como um módulo de terceiros em Python é natsort, em Perl é Sort::Naturally e em Matlab é sort_nat. É possível obter o mesmo comportamento usando algumas soluções alternativas, como adicionar zeros à esquerda.

No MariaDB 10.7.0, baseado no MDEV-4742, a classificação natural está disponível através da função natural_sort_key().

A função NATURAL_SORT_KEY é usada para ordenação mais próxima da ordenação natural. As strings são classificadas em ordem alfabética, enquanto os números são tratados de forma que, por exemplo, 10 seja maior que 2, enquanto em outras formas de classificação, 2 seria maior que 10, assim como z é maior que y.

Existem várias implementações de classificação natural, diferindo na maneira como lidam com zeros à esquerda, frações, i18n, negativos, decimais e assim por diante.

A implementação do MariaDB ignora zeros à esquerda ao realizar a classificação.

Você também pode usar NATURAL_SORT_KEY com colunas geradas. O valor não é armazenado permanentemente na tabela. Ao usar uma coluna gerada, a coluna virtual deve ser maior que a coluna base para atender aos números incorporados na string e no MDEV-24582.

Links da documentação:

https://mariadb.org/10-7-preview-feature-natural-sort/

https://www.mariadbtutorial.com/mariadb-basics/mariadb-order-by/

https://mariadb.com/kb/en/order-by/

https://mariadb.com/kb/en/natural_sort_key/