Aula Prática 8

Objetivos

Índices de Base de Dados Relacional – baseado em SQL Server.

Nota: As submissões devem seguir o template de resposta facultado.

Problema 8.1

Este problema tem como referência a tabela *Production.WorkOrder* da base de dados *AdventureWorks2012*. Deve descarregar o ficheiro *AdventureWorks2012.bak*[4] e restaurar a base de dados seguindo o tutorial: *Restore to SQL Server*[5]. A tabela *Production.WorkOrder* tem um *Clustered Unique index* associado à PK *WorkOrderID*[6].

Utilizando as ferramentas ***SQL Server Profiler*** e ***Query Execution Plan***[7], **registe e discuta** os valores obtidos (index/query/rows/cost/pag. reads/...) para cada uma das experiências abaixo. Recomenda-se que apresente os resultados obtidos na forma de tabela contendo os seguintes elementos:

#	Query	Rows	Cost	Pag. Reads	Time (ms)	Index used	Index Op.
1	select * from Production.WorkOrder	72591	.484	531	1171	…	Clustered Index Scan
2	…	…	…	…	…	…	…

Nota: antes de executar cada uma das queries deve executar as seguintes instruções:

```
DBCC FREEPROCCACHE;
DBCC DROPCLEANBUFFERS;
```

Experiências:

```
#1.   Index:  WorkOrderID (PK)
      Query: select * from Production.WorkOrder

#2.   Index:  WorkOrderID (PK)
      Query: select * from Production.WorkOrder where WorkOrderID=1234

#3.   Index:  WorkOrderID (PK)
      Query1: SELECT * FROM Production.WorkOrder
              WHERE WorkOrderID between 10000 and 10010

      Query2: SELECT * FROM Production.WorkOrder
              WHERE WorkOrderID between 1 and 72591
```

[4] https://github.com/Microsoft/sql-server-samples/releases/download/adventureworks/AdventureWorks2012.bak

[5] https://learn.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-ver16&tabs=ssms#restore-to-sql-server

[6] https://github.com/CarlosCosta-UA/BD-UA/blob/main/aula8/adventure_works_2012_clustered_idx.JPG

[7] Disponíveis no SQL Server Management Studio.

```
#4.    Index: WorkOrderID (PK)

       Query: SELECT * FROM Production.WorkOrder
              WHERE StartDate = '2012-05-14'


#5.    Index: ProductID

       Query: SELECT * FROM Production.WorkOrder WHERE ProductID = 757


#6.    Index:  ProductID Covered (StartDate)

       Query1: SELECT WorkOrderID, StartDate FROM Production.WorkOrder
               WHERE ProductID = 757
       Query2: SELECT WorkOrderID, StartDate FROM Production.WorkOrder
               WHERE ProductID = 945
       Query3: SELECT WorkOrderID FROM Production.WorkOrder
               WHERE ProductID = 945 AND StartDate = '2011-12-04'


#7.    Index: ProductID and StartDate

       Query: SELECT WorkOrderID, StartDate FROM Production.WorkOrder
              WHERE ProductID = 945 AND StartDate = '2011-12-04'


#8.    Index: Composite (ProductID, StartDate)

       Query: SELECT WorkOrderID, StartDate FROM Production.WorkOrder
              WHERE ProductID = 945 AND StartDate = '2011-12-04'
```

Problema 8.2

Tenha como base a seguinte tabela:

```
CREATE TABLE mytemp (
       rid BIGINT /*IDENTITY (1, 1)*/ NOT NULL,
       at1 INT NULL,
       at2 INT NULL,
       at3 INT NULL,
       lixo varchar(100) NULL
);
```

a) Defina *rid* como chave primária do tipo *Clustered Index*.

b) Registe os tempos de introdução de 50000 novos registos (tuplos) na tabela utilizando o código abaixo:

```
-- Record the Start Time
DECLARE @start_time DATETIME, @end_time DATETIME;
SET @start_time = GETDATE();
PRINT @start_time

-- Generate random records
DECLARE @val as int = 1;
DECLARE @nelem as int = 50000;

SET nocount ON

WHILE @val <= @nelem
BEGIN
    DBCC DROPCLEANBUFFERS;                          -- need to be sysadmin

    INSERT mytemp (rid, at1, at2, at3, lixo)
```

```
            SELECT cast((RAND()*@nelem*40000) as int),  cast((RAND()*@nelem) as int),
                   cast((RAND()*@nelem) as int),  cast((RAND()*@nelem) as int),
                   'lixo...lixo...lixo...lixo...lixo...lixo...lixo...lixo...lixo';
            SET @val = @val + 1;
        END

        PRINT 'Inserted ' + str(@nelem) + ' total records'

        -- Duration of Insertion Process
        SET @end_time = GETDATE();
        PRINT 'Milliseconds used: ' + CONVERT(VARCHAR(20), DATEDIFF(MILLISECOND,
@start_time, @end_time));
```

Qual a percentagem de fragmentação dos índices e de ocupação das páginas dos índices?

c) Varie o *fillfactor* (por exemplo: 65, 80 e 90) do *clustered index* e veja o efeito nos tempos de inserção.

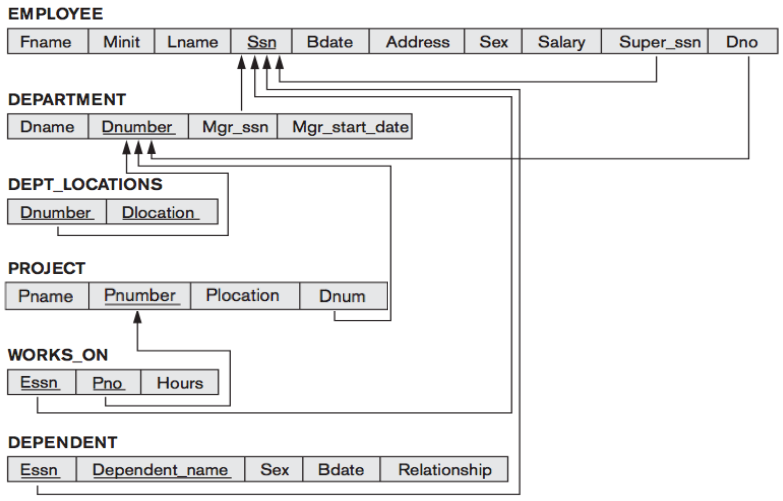
d) Altere a tabela `mytemp` para que o atributo *rid* passe a ser do tipo *identity*. Volte a medir os tempos de inserção[8].

e) Crie um índice para cada atributo da tabela `mytemp`. Compare os tempos de inserção obtidos, sem e com todos os índices. O que pode concluir?

Nota: Os resultados obtidos neste exercício podem variar de acordo com o tipo de computador/máquina virtual (por exemplo: hardware HDD/SSD) e com a carga da máquina no momento em que está a decorrer a experiência;

Problema 8.3

Tendo como base o esquema da base de dados apresentado na figura abaixo (desenvolvido nas aulas teóricas):



a) Defina os índices que achar conveniente para cada uma das relações. Tenha em atenção que usualmente temos necessidade de efetuar as seguintes consultas à base de dados:

[8] Tem de alterar o código fornecido na alínea b) para esta nova situação.

i. O funcionário com determinado número ssn;
ii. O(s) funcionário(s) com determinado primeiro e último nome;
iii. Os funcionários que trabalham para determinado departamento;
iv. Os funcionários que trabalham para determinado projeto;
v. Os dependentes de determinado funcionário;
vi. Os projetos associados a determinado departamento;