

Complementos de Bases de Dados – Índices –

Engenharia Informática
2º Ano / 1º Semestre

Cláudio Miguel Sapateiro
claudio.sapateiro@estsetubal.ips.pt

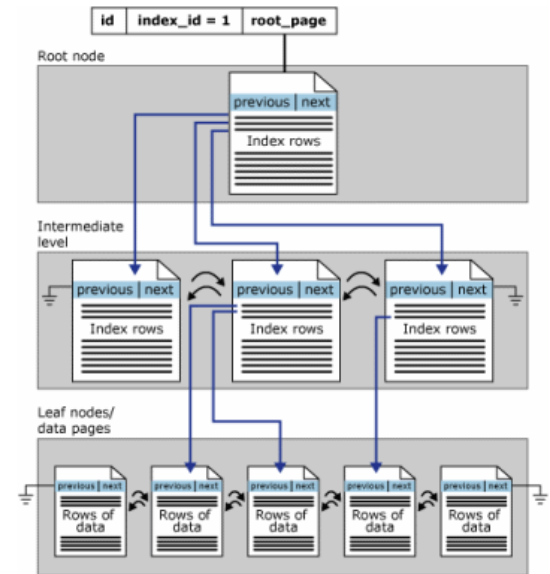
Índices - Parte 2

ÍNDICES EM MS SQL

Índices em SQL

Tipos de Índices MS SQL

- *clustered*
 - Ordenado (agrupado)
 - “armazena” a informação na estrutura de índice
- *non-clustered*
 - apenas “aponta” para a informação.
- Uma tabela apenas pode conter um índice *clustered* e até 999 *non-clustered*
- Uma tabela sem um índice *clustered* é denominada de *heap*

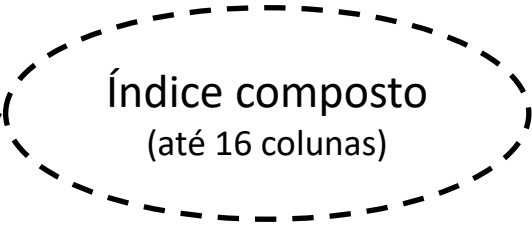


Índices em SQL

Índices MS SQL

- **Sintaxe**

```
CREATE [ UNIQUE ] [ CLUSTERED |  
NONCLUSTERED ] INDEX index_name  
ON <object> ( column [ ASC | DESC ] [ ,...n ] )
```



Índice composto
(até 16 colunas)

- Por defeito o SQLServer cria um índice *unique clustered* na PRIMARY KEY, senão houver já a criado outro índice *clustered*, nesse caso a PRIMARY KEY terá de ser indexada em modo *nonclustered*

Índices em SQL

Índices MS SQL

Exemplos

```
create table t1 (id int not null);
```

```
select * from sys.indexes where object_id = object_id('t1');
```

Results		Messages					
	object_id	name	index_id	type	type_desc	is_unique	data_sp
1	158623608	NULL	0	0	HEAP	0	1

```
alter table t1 add constraint pk_t1 primary key (id);
```

```
select * from sys.indexes where object_id = object_id('t1');
```

Results Messages

	object_id	name	index_id	type	type_desc	is_unique
1	174623665	pk_t1	1	1	<u>CLUSTERED</u>	1

Índices em SQL

Índices MS SQL

Exemplos

```
| create table t1 (id int primary key);  
  
select * from sys.indexes where object_id = object_id('t1');
```

Results		Messages				
	object_id	name	index_id	type	type_desc	is
1	206623779	PK_t1__3213E83F2FC91D62	1	1	<u>CLUSTERED</u>	1

Índices em SQL

Índices MS SQL

Exemplos

```
create table t2 (id int not null, cx int);  
  
create clustered index idx on dbo.t2 (cx asc);  
  
alter table t2 add constraint pk_t2 primary key(id);  
  
select * from sys.indexes where object_id = object_id('t2');
```

O que é expectável que aconteça ?

Results		Messages					
	object_id	name	index_id	type	type_desc	is_unique	d
1	254623950	idx	1	1	CLUSTERED	0	1
2	254623950	pk_t2	2	2	NONCLUSTERED	1	1

Índices em SQL

Índices MS SQL

Exemplos

```
CREATE TABLE Sales(  
    ID INT IDENTITY(1,1)  
    ,ProductCode VARCHAR(20)  
    ,Price FLOAT(53)  
    ,DateTransaction DATETIME);
```

EXEC InsertIntoSales

Select * from sales

	ID	ProductCode	Price	DateTransaction
1	1	A12CB0	50	2015-11-16 16:29:12.620
2	2	A12CB1	51	2015-11-16 16:29:12.620
3	3	A12CB2	52	2015-11-16 16:29:12.620
4	4	A12CB3	53	2015-11-16 16:29:12.623
5	5	A12CB4	54	2015-11-16 16:29:12.623

```
CREATE PROCEDURE InsertIntoSales  
AS  
SET NOCOUNT ON  
BEGIN  
    DECLARE @PC VARCHAR(20)='A12CB'  
    DECLARE @Price INT = 50  
    DECLARE @COUNT INT = 0  
    WHILE @COUNT<200000  
    BEGIN  
        SET @PC=@PC+CAST(@COUNT AS VARCHAR(20))  
        SET @Price=@Price+@COUNT  
        INSERT INTO Sales VALUES  
            (@PC,@Price,GETDATE())  
        SET @PC='A12CB'  
        SET @Price=50  
        SET @COUNT+=1  
    END  
END
```


Índices em SQL

Índices MS SQL

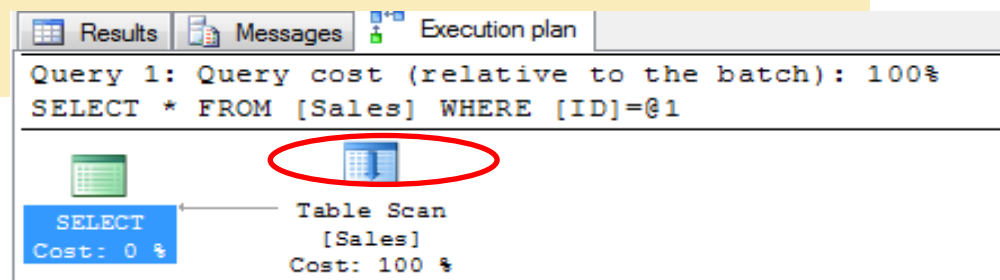
Exemplos

```
SET STATISTICS IO ON  
SELECT * FROM Sales WHERE ID=189923
```

ID	ProductCode	Price	DateTransaction
189923	A12CB189922	189972	2011-03-21 12:07:48.310

Table 'Sales'. Scan count 1, logical reads 1129, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.

(1 row(s) affected)



Índices em SQL

Índices MS SQL

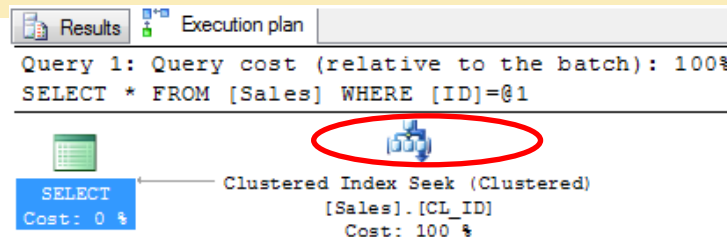
Exemplos (*clustered*)

```
CREATE CLUSTERED INDEX CL_ID ON SALES(ID);
```

```
SET STATISTICS IO ON  
SELECT * FROM Sales WHERE ID=189923
```

ID	ProductCode	Price	DateTransaction
189923	A12CB189922	189972	2011-03-21 12:07:48.310

Table 'Sales'. Scan count 1, logical reads 3, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.



Índices em SQL

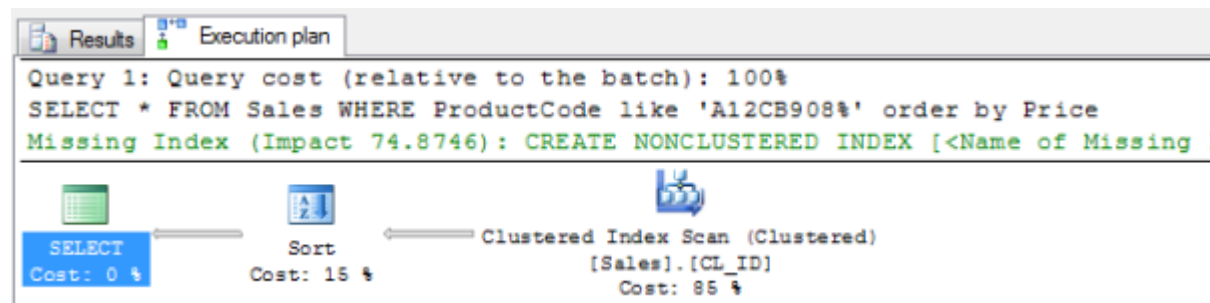
Índices MS SQL

Exemplos

```
SET STATISTICS IO ON  
SELECT * FROM Sales WHERE ProductCode like 'A12CB908%' order by Price
```

(111 row(s) affected)

Table 'Sales'. Scan count 1, logical reads 1130, physical reads 0, read-ahead reads 0, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.



Índices em SQL

Índices MS SQL

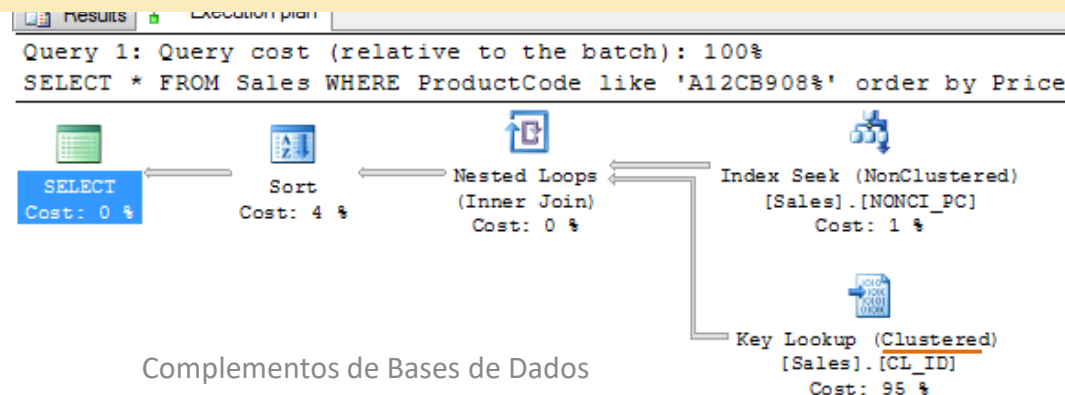
Exemplos (*nonclustered* & *clustered*)

```
CREATE NONCLUSTERED INDEX NONCI_PC ON SALES(ProductCode);
```

```
SET STATISTICS IO ON  
SELECT * FROM Sales WHERE ProductCode like 'A12CB908%' order by Price
```

(111 row(s) affected)

Table 'Sales'. Scan count 1, logical reads 351, physical reads 0, read-ahead reads 7, lob logical reads 0, lob physical reads 0, lob read-ahead reads 0.



Índices em SQL

Índices MS SQL

Exemplos (*nonclustered* & *heap*)

```
DROP INDEX Sales.CL_ID;
```

```
SET STATISTICS IO ON
```

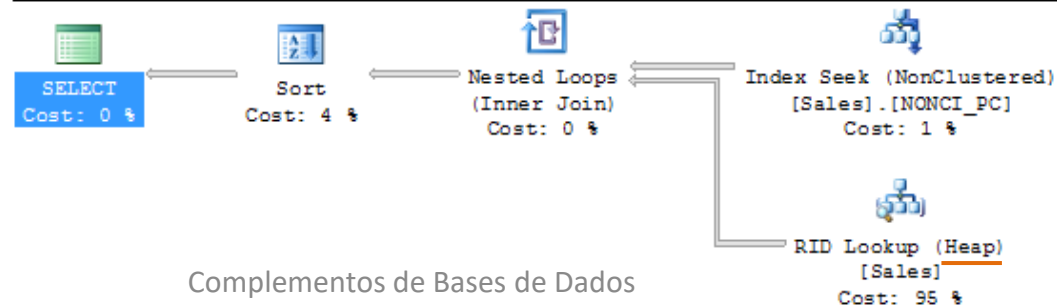
```
SELECT * FROM Sales WHERE ProductCode like 'A12CB908%' order by Price
```

(111 row(s) affected)

Table 'Sales'. Scan count 1, logical reads 114, physical reads 0,
read-ahead reads 0, lob logical reads 0,
lob physical reads 0, lob read-ahead reads 0.

Query 1: Query cost (relative to the batch): 100%

```
SELECT * FROM Sales WHERE ProductCode like 'A12CB908%' order by Price
```



Índices em SQL

Cover vs Composite index

- **Composite:**
 - Indexa um conjunto de colunas
- **Cover:**
 - Indica num índice non-clustered, informação adicional para ser armazenada juntamente no índice

```
SELECT OrderID, OrderDate  
FROM Sales  
WHERE OrderID = 12345;
```



```
CREATE NONCLUSTERED INDEX ix_orderid  
ON dbo.Sales (OrderID)  
INCLUDE (OrderDate);
```

Índices em SQL

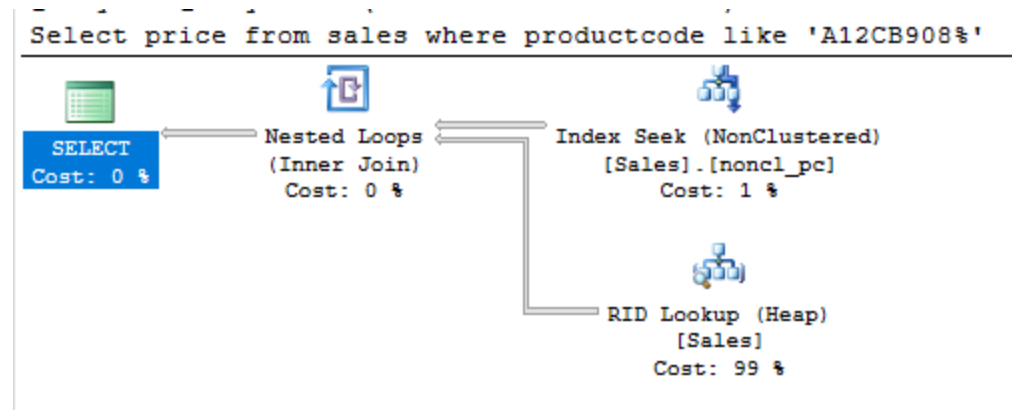
Cover

Exemplo

```
create nonclustered index noncl_pc on dbo.sales(productcode);
```

```
SET STATISTICS IO ON
```

```
Select price from sales where productcode like 'A12CB908%';
```



Índices em SQL

Cover

Exemplo

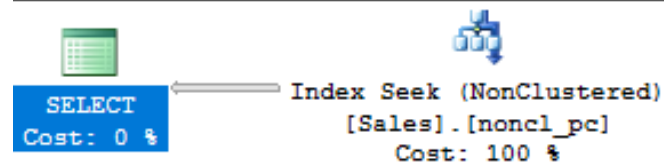
```
create nonclustered index noncl_pc on sales(productcode) include(price);
```

```
drop index sales.noncl_pc;
```

```
SET STATISTICS IO ON
```

```
Select price from sales where productcode like 'A12CB908%';
```

Select price from sales where produc



Índices em SQL

Filtered index

- Exclui do índice um conjunto de registo de acordo com um critério
 - Tem o potencial de melhorar a performance, considerando o numero de entradas e operações de atualização

```
CREATE NONCLUSTERED INDEX ix_trackingnumber  
ON Sales.SalesOrderDetail(CarrierTrackingNumber)  
WHERE CarrierTrackingNumber IS NOT NULL;
```

Índices em SQL

Exercício: Índice com múltiplas chaves de pesquisa

- Exemplo:

```
SELECT account_number  
FROM account  
WHERE branch_name = 'Perryridge'  
AND balance = 1000
```

- Estratégias possíveis:
 - Usar um índice para *branch_name* e testar o valor de *balance*;
 - Usar um índice para *balance* e testar o valor de *branch_name*;
 - Usar índices para *branch_name* e *balance* e intersectar os resultados.

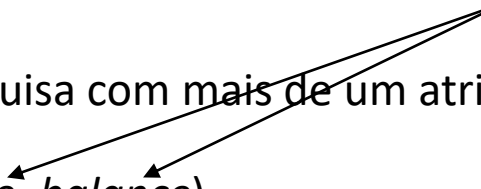
Índices em SQL

Exercício: Índices com múltiplas chaves de pesquisa

- Índices com chaves compostas, estratégias possíveis:
 - Chaves de pesquisa com mais de um atributo
Exemplo:
(branch_name, balance)
branch_name = 'Perryridge' AND balance = 1000;

• mais eficiente que índices separados.

• também é eficiente em:
branch_name = 'Perryridge' AND balance < 1000;

• não é eficiente em:
branch_name < 'Perryridge' AND balance = 1000; —————> Porquê?
- Na criação do índice a ordem interessa!
- 

Índices em SQL

Exemplo: Índices com múltiplas chaves de pesquisa

```
set statistics IO on  
select *  
from sales2  
where productcode = 'A12BC49000'  
and price = 49700
```

```
(100 row(s) affected)  
Table 'sales2'. Scan count 1, logical reads 1079,  
  
(1 row(s) affected)
```

Índices em SQL

Exemplo: Índices com múltiplas chaves de pesquisa

```
create nonclustered index noncl_pc on sales2(productcode,price);
```

```
set statistics IO on
```

```
select *
```

```
from sales2
```

```
where productcode = 'A12BC49000'
```

```
and price = 49700;
```

```
(100 row(s) affected)
```

```
Table 'sales2'. Scan count 1, logical reads 103, physical
```

```
(1 row(s) affected)
```

```
select *
```

```
from sales2
```

```
where productcode > 'A12BC4900'
```

```
and price = 49700;
```

```
(100 row(s) affected)
```

```
Table 'sales2'. Scan count 1, logical reads 355, physical
```

```
(1 row(s) affected)
```

```
|
```

Índices em SQL

Exemplo: Índices com múltiplas chaves de pesquisa

```
create nonclustered index noncl_pc on sales3(price,productcode);
```

```
set statistics IO on
```

```
select *
```

```
from sales3
```

```
where productcode = 'A12BC49000'
```

```
and price = 49700;
```

(100 row(s) affected)

Table 'sales3'. Scan count 1, logical reads 103, physic

(1 row(s) affected)

```
select *
```

```
from sales3
```

```
where productcode > 'A12BC4900'
```

```
and price = 49700;
```

(100 row(s) affected)

Table 'sales3'. Scan count 1, logical reads 103, physic

(1 row(s) affected)

Síntese

Operation	Memory-optimized hash, index	Memory-optimized nonclustered index	Disk-based (non)clustered index
Index Scan, retrieve all table rows.	Yes	Yes	Yes
Index seek on equality predicate(s) (=).	Yes (Full key required.)	Yes*	Yes
Index seek on inequality predicates (>, <, <=, >=, BETWEEN).	No (results in an index scan)	Yes*	Yes
Retrieve rows in a sort-order matching the index definition.	No	Yes	Yes
Retrieve rows in a sort-order matching the reverse of the index definition.	No	No	Yes

<https://msdn.microsoft.com>

Síntese

ÍNDICES E DESEMPENHO

Existem essencialmente três operações efetuadas através de índices:

- Consulta apenas do índice: Neste caso o acesso é muito rápido, dispensando o acesso às páginas da tabela. Acessos nesta categoria incluem consultas por igualdade de chave primária ou de outra coluna única.
- Consulta por intervalo no índice: Neste caso o acesso percorre todas as folhas do índice que respeitam o critério de pesquisa no intervalo de valores. O resultado é um conjunto de endereços de registos.
- Consulta da tabela usando entrada do índice: Neste caso a página que contém o endereço encontrado no índice é lida. Esta situação é comum nos casos em que se acede a colunas que não estão no índice. Uma situação limite desta operação consulta todos os endereços da tabela, pelo que o varrimento sequencial pode ser a opção mais económica.

Fonte: Feliz Gouveia, *Fundamentos de Bases de Dados*, FCA – Editora de Informática

Síntese

(Algumas) Linhas orientadoras para Indexação

- Para tabelas frequentemente atualizadas utilizar poucas colunas indexadas
- Numa tabela com muitos dados, mas taxa de atualização baixa, é recomendada a utilização de vários índices de acordo com o tipo de *queries* previsto
- Colunas indexadas em modo *clustered index* devem ter valores de pequena dimensão, não nulos e preferencialmente/tendencialmente únicos
- Em regra quantos mais valores duplicados existirem na coluna indexada pior é a performance da indexação
- Em índices compostos, interessa a ordem das colunas; colunas cujo os valores serão testados na clausula WHERE das *queries* devem ser colocados em primeiro no índice
(colunas com os valores “mais únicos” devem ser colocados no fim)

(Algumas) Linhas orientadoras para Indexação

- É muitas vezes desnecessário indexar tabelas de reduzida dimensão
- Em geral, quando não é a chave de organização do dados deve ainda assim indexar-se a chave primária
- É normalmente pertinente indexar chaves estrangeiras
- Regularmente constituem-se índices secundários em atributos nestas condições:
 - selection or join criteria;
 - ORDER BY;
 - GROUP BY;
 - operações que envolvem ordenação como a UNION ou o DISTINCT
 - atributos utilizados em computação de funções,
exemplo:
SELECT branchNo, **AVG**([salary](#))
FROM Staff
GROUP BY [branchNo](#);

(Algumas) Linhas orientadoras para Indexação

- Generalizando a anterior, adicionar índices secundário a atributos que permitam gerar planos de execução de *queries*, *index-only*
- Evitar indexação de atributos que são frequentemente alvo de atualizações
 - Embora contudo um campo indexado (e.g. EmployeeNo) possa auxiliar atualização de outros campos
- Evitar indexar atributos cujo o resultado da *querie* traga mais de 25% dos registos da relação (nestes casos pode não ser grande o ganho em estar indexado vs não indexado)
- Evitar indexar atributos do tipo *char* de grande comprimento
- Deve-se fazer ensaio sobre se o índice, traz melhorias desempenho, se estas são significativas ou se estão a degradar desempenho noutras *queries* ou operações
 - Recorrer a ferramentas de monitorização avaliação para avaliar desempenho
- ❖ Pode ser útil em operações de carregamento ou manutenção desativar os índices
- ❖ Alguns SGBD requerem uma atualização explícita das estatísticas no catalogo para que os planos de execução passem a considerar a presença de um novo índice
- ❖ DOCUMENTAR as escolhas de índices

mini Sumário

1. Índices em MS SQL
2. Linhas orientadoras para indexação

05:00



Exercícios

1. Quantos *clustered indexes* podem existir numa tabela?
2. Ter mais índices na tabela é sempre melhor? Porquê?
3. Distinga *composite index* de *cover index*.

References

- <http://www.codeproject.com/Articles/190263/Indexes-in-MS-SQL-Server>
- <https://www.simple-talk.com/sql/learn-sql-server/sql-server-index-basics/>
- <https://www.simple-talk.com/sql/performance/14-sql-server-indexing-questions-you-were-too-shy-to-ask/>
- <https://www.simple-talk.com/sql/performance/execution-plan-basics/>
- <http://www.sqlshack.com/sql-server-query-execution-plans-understanding-reading-plans/>
- DTA; Execution Plan » Monitorização e tuning

Complementos de Bases de Dados – Índices –

Engenharia Informática
2º Ano / 1º Semestre

Cláudio Miguel Sapateiro
claudio.sapateiro@estsetubal.ips.pt