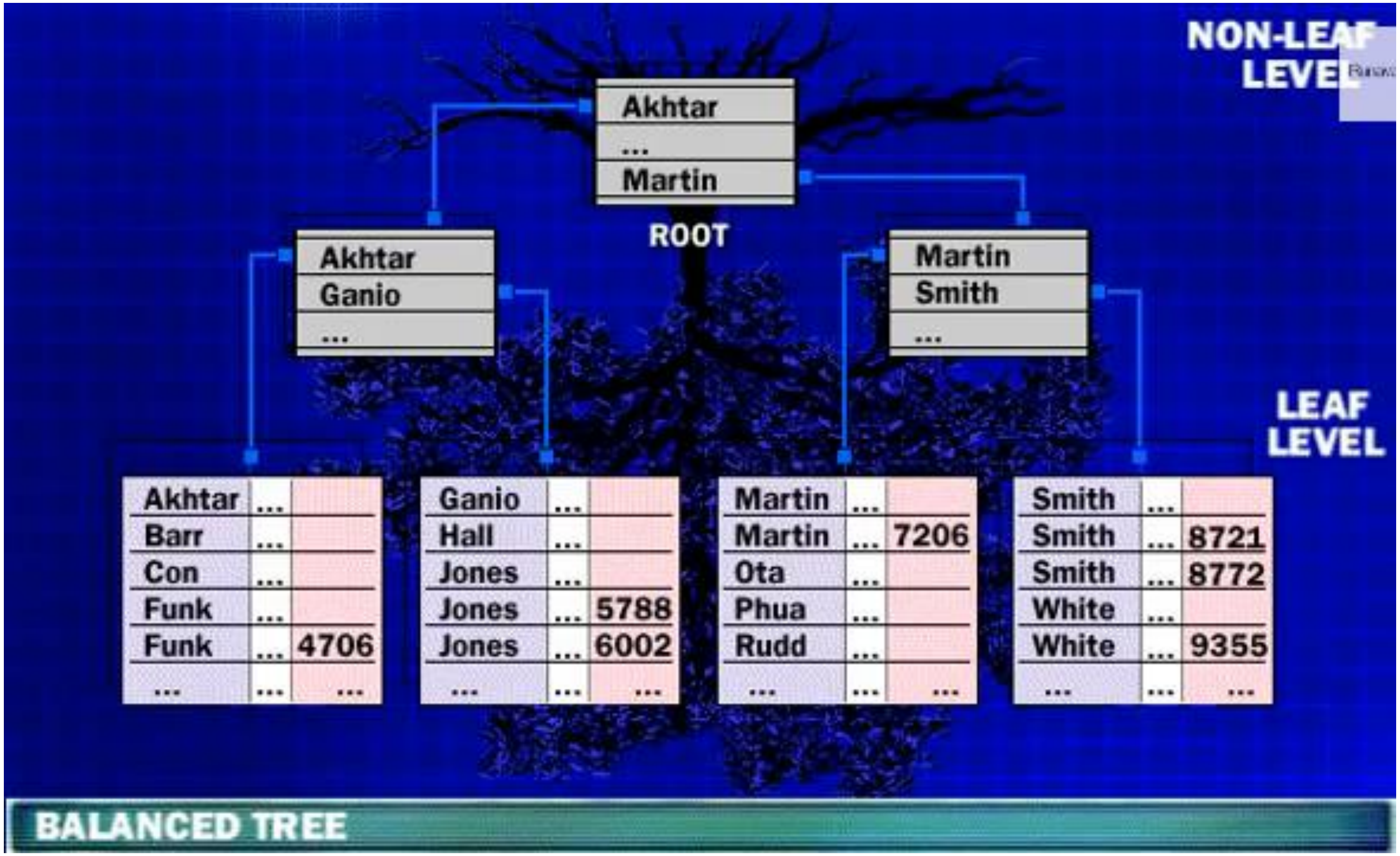
MS Sql Server Indexes

Author: Nemanja Kojic MScEE

Indexing

- Vital for system performance
- Improves query execution performance
- NOT one size fits all – trade offs must be made
- Penalties during INSERT/UPDATE – index update
- Two types of indexes:
  - Clustered Indexes
  - NonClustered Indexes

Index example



Akhtar
...
Martin

Akhtar
Ganio
...

Martin
Smith
...

Akhtar	...	
Barr	...	
Con	...	
Funk	...	
Funk	...	4706
...	...	...

Ganio	...	
Hall	...	
Jones	...	
Jones	...	5788
Jones	...	6002
...	...	...

Martin	...	
Martin	...	7206
Ota	...	
Phua	...	
Rudd	...	
...	...	...

Smith	...	
Smith	...	8721
Smith	...	8772
White	...	
White	...	9355
...	...	...

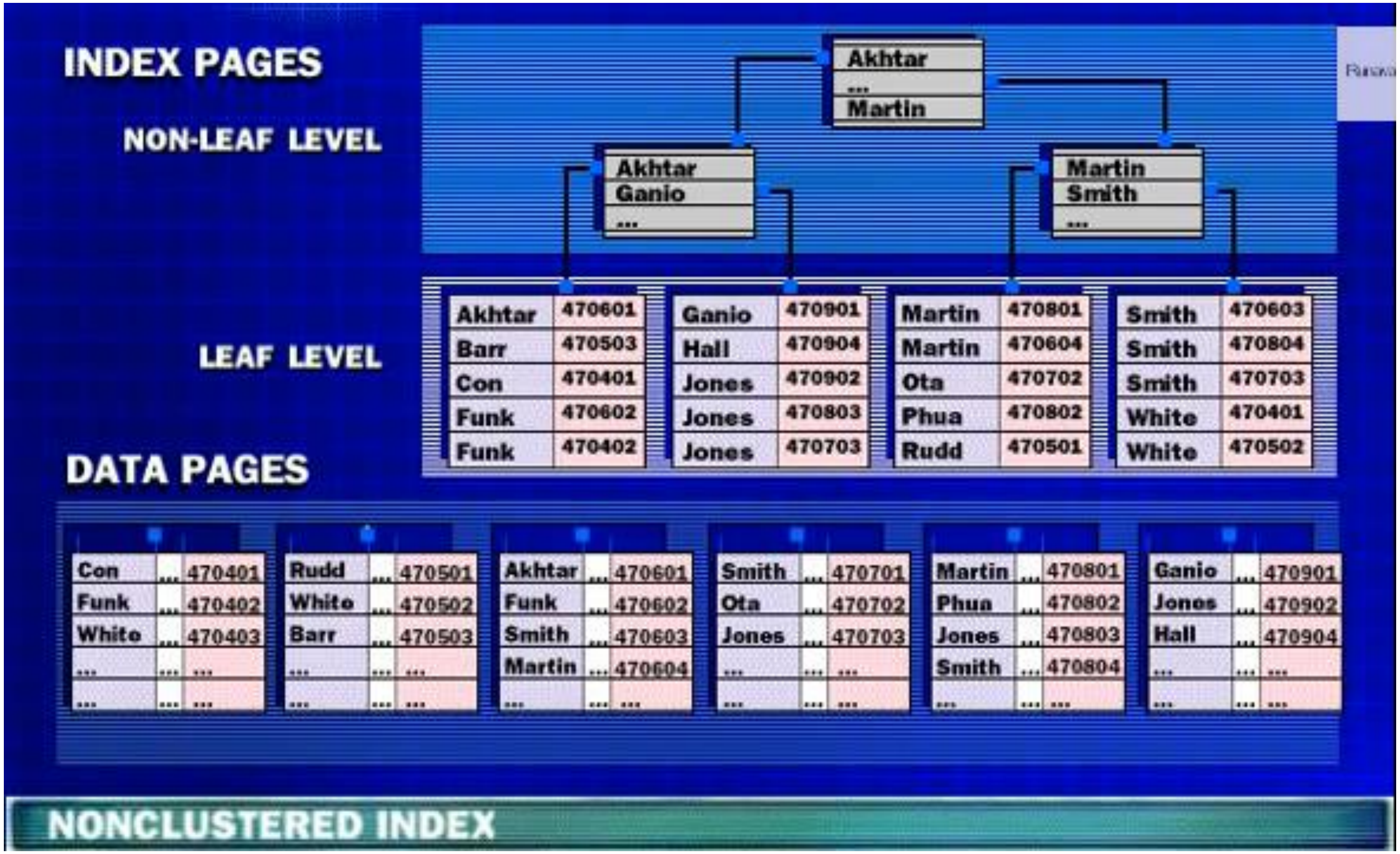
Non-Clustered Index

- Data in pages in random order
- Logical data order in index
- NonClustered index tree
  - Keys in sorted order
  - Leaf pages contain pointers to rows in data pages
- Typicaly created on column used in JOIN, WHERE, ORDER BY
- Good for tables whose values may be modified frequently

NonClustered Index (cont.)

- MS Sql Server:
  CREATE INDEX -> nonClustered by default
- Allowed more than index on a db table
- MS Sql Server 2008:
  up to 999 nonClustered indexes per table

Non-Clustered Index example



Akhtar	470601
Barr	470503
Con	470401
Funk	470602
Funk	470402

Ganio	470901
Hall	470904
Jones	470902
Jones	470803
Jones	470703

Martin	470801
Martin	470604
Ota	470702
Phua	470802
Rudd	470501

Smith	470603
Smith	470804
Smith	470703
White	470401
White	470502

Con	...	470401
Funk	...	470402
White	...	470403
...	...	...
...	...	...

Rudd	...	470501
White	...	470502
Barr	...	470503
...	...	...
...	...	...

Akhtar	...	470601
Funk	...	470602
Smith	...	470603
Martin	...	470604
...	...	...

Smith	...	470701
Ota	...	470702
Jones	...	470703
...	...	...
...	...	...

Martin	...	470801
Phua	...	470802
Jones	...	470803
Smith	...	470804
...	...	...

Ganio	...	470901
Jones	...	470902
Hall	...	470904
...	...	...
...	...	...

Non-Clustered Index - summation

- Create index on columns which are:
  - Frequently used in search criteria
  - Used to JOIN different tables
  - Used as foreign key fields
  - Of having high selectability
  - Used in ORDER BY clause
  - Of type XML (primary and secondary indexes)

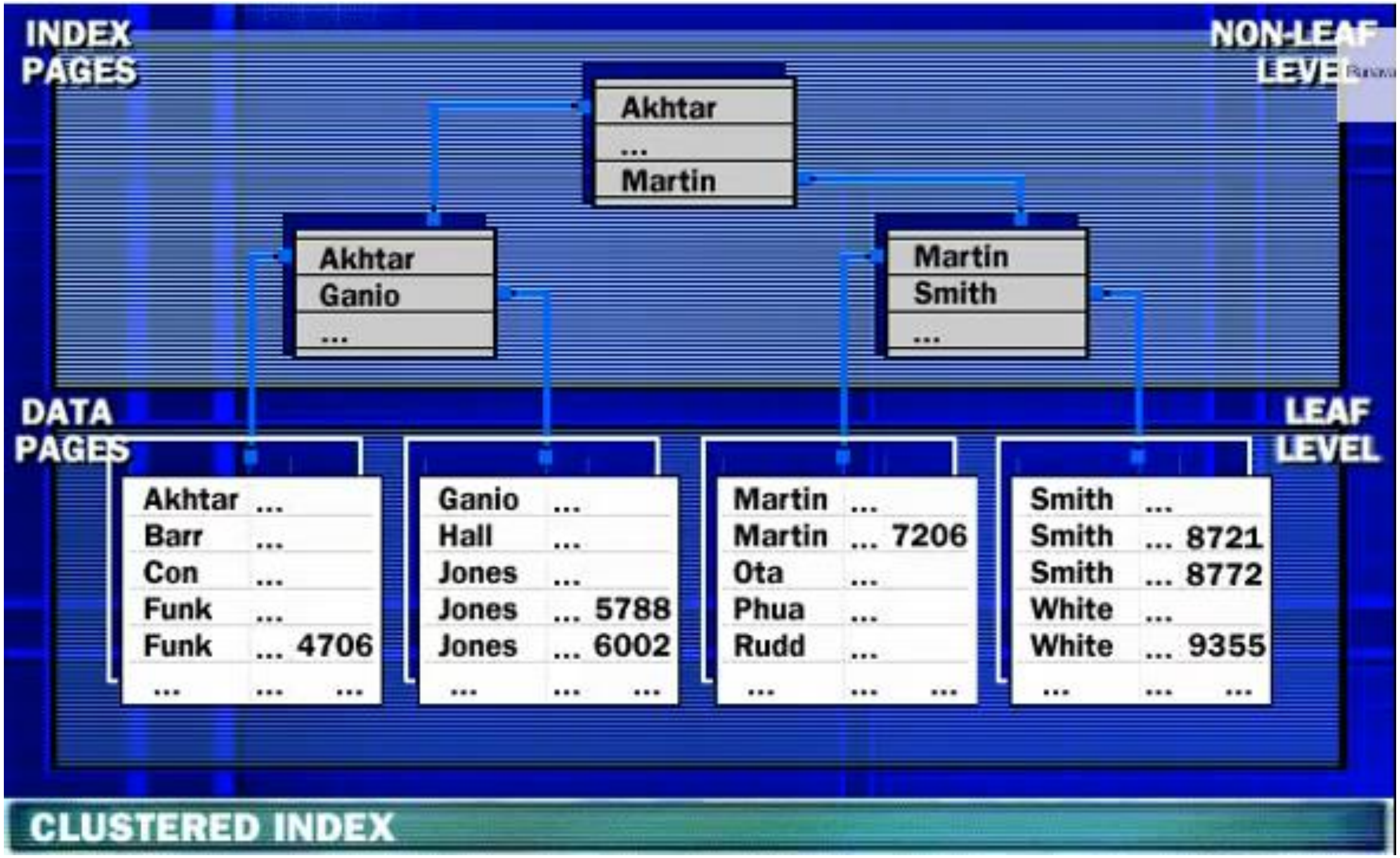
Clustered Index

- Re-orders data rows to match the index
  (rows in sort order on disk)
- Only one clustered index per table!
- Leaf level of the index tree - actual data rows
- Good for
  sequential access, and
  range selection

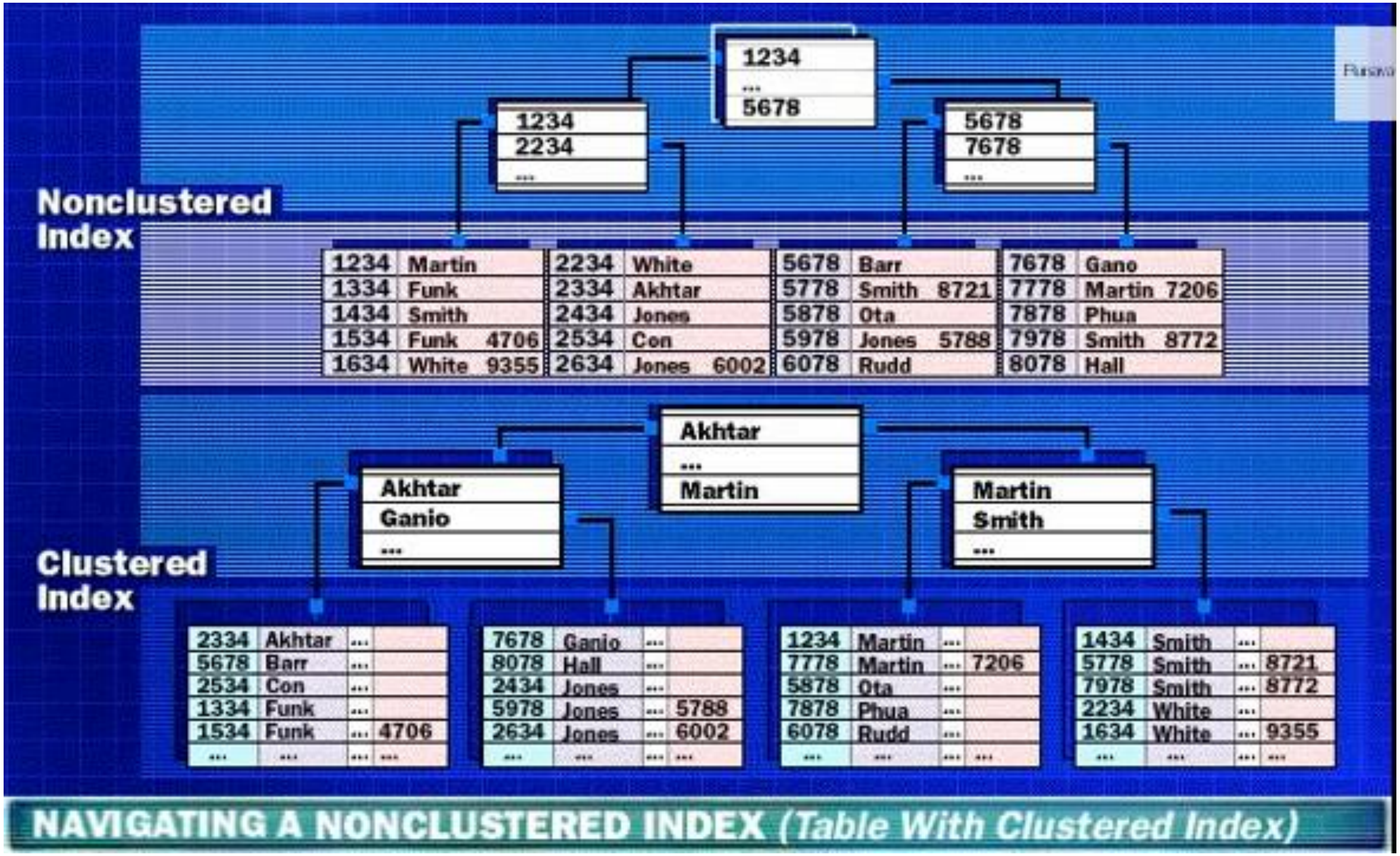
Clustered Index (cont.)

- MS Sql Server INSERTS data according to the way a clustered index was created
- Most often:
  PRIMARY KEY => Clustered Index
- Every table SHOULD have clustered index
- w/o clustered index:
  records added to the end of the last page
- w/ clustered index:
  data added to suitable position dictated by the index

Clustered Index example



Clustered and Non-Clustered Index combined



1234	Martin	
1334	Funk	
1434	Smith	
1534	Funk	4706
1634	White	9355

2234	White	
2334	Akhtar	
2434	Jones	
2534	Con	
2634	Jones	6002

5678	Barr	
5778	Smith	8721
5878	Ota	
5978	Jones	5788
6078	Rudd	

7678	Gano	
7778	Martin	7206
7878	Phua	
7978	Smith	8772
8078	Hall	

2334	Akhtar	...	
5678	Barr	...	
2534	Con	...	
1334	Funk	...	
1534	Funk	...	4706
...	...	...	...

7678	Ganio	...	
8078	Hall	...	
2434	Jones	...	
5978	Jones	...	5788
2634	Jones	...	6002
...	...	...	...

1234	Martin	...	
7778	Martin	...	7206
5878	Ota	...	
7878	Phua	...	
6078	Rudd	...	
...	...	...	...

1434	Smith	...	
5778	Smith	...	8721
7978	Smith	...	8772
2234	White	...	
1634	White	...	9355
...	...	...	...

NAVIGATING A NONCLUSTERED INDEX ***(Table With Clustered Index)***

Covering indexes

- Extending functionality of nonCls indexes
- Adding non-key columns to the leaf level
- Index covers more types of queries
- Covering Indexes = Indexes w/ incl. columns
- Great performance benefits

Filtering indexes

- NonClustered index with a record filter
- Covers a subsed of records in a table
- Reduces storage space for index
- Better performance
- Decreased INSERT penalty

Index selectivity and Density

- Selectivity:
  number of distinct key values in the table
- PRIMARY KEY, UNIQUE – perfectly selective
- The higher selective Index, the better perform.
- Density:
  number of duplicate key values in the table
- Query optimizer: index seek, index scan

Fill factor

- Tuning storage and performance
- Fill factor = % of space for data in leaf pages
- Remainder of the page for future growth
- E.g. Fill factor=80% => 20% page empty
- Reserved space between index rows (rather than at the of the index)
- Applied on CREATE or REBUILD INDEX

Fill factor - guidelines

- Depends on how data are accessed
- Data inserted at the end of the table =>
  FILL FACTOR = 90%-100%
- Data inserted anywhere =>
  FILL FACTOR = 60%-80%
- The lower FF, the higher storage for the index
- In general: appropriate FF requires a lot of testing and probing

Creating indexes – Best Practices

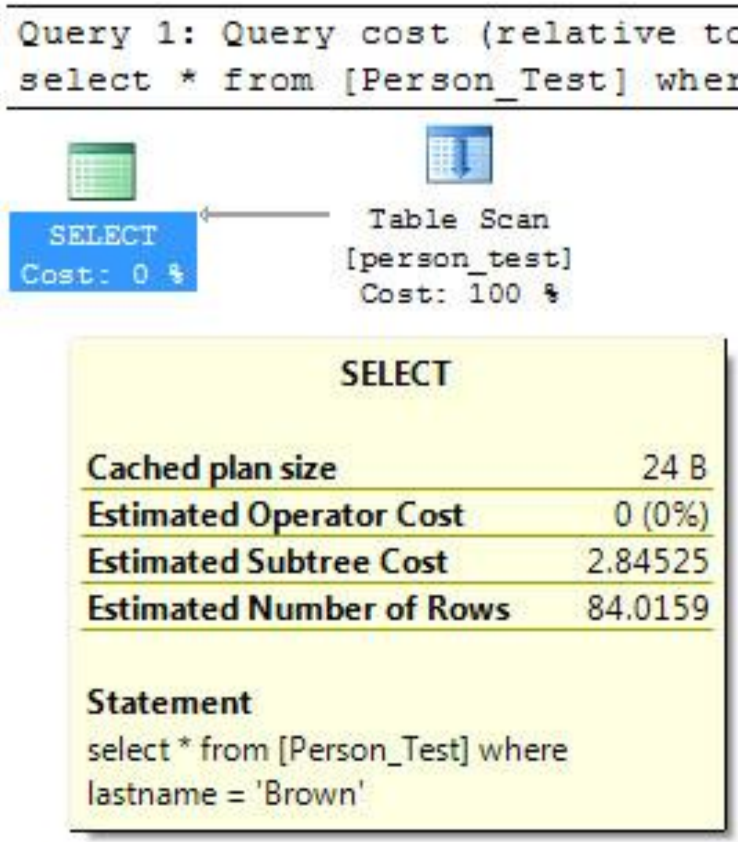
- Keep indexes narrow (one or few columns)
- Clustered index on every table
- Clustered index on a highly selective column
- Clustered index on a column that is never upd.
- Default: clustered index on PRIMARY KEY col.
- Be aware of penalties during INSERT/UPDATE
- Eliminate duplicate indexes.
- Check the default FILL FACTOR
- Non-clustered indexes can be created in different file groups, which may increase performance

Order of fields on each index?

- Bad order => index is not useful
- Most selective columns go first
- Sql Server knows data distribution only for the first column!
- Don’t place column from clustered index to a non-clustered index

EXAMPLES

Table with NO indexes select * …



```
Query 1: Query cost (relative to the batch): 100%
select * from [Person_Test] where lastname = 'Brown'
```

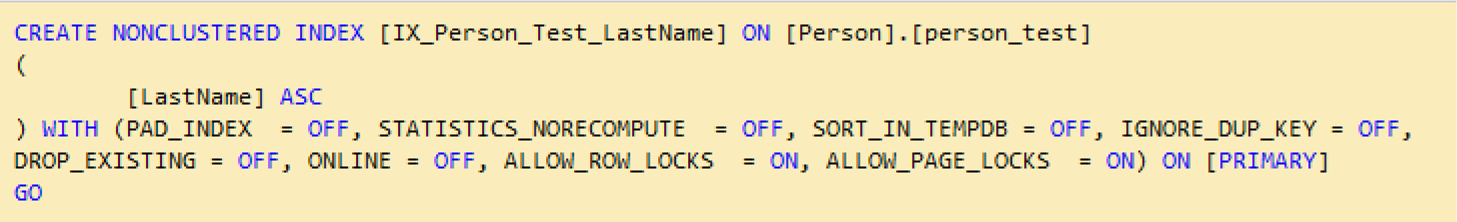
SELECT

Cached plan size	24 B
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	2.84525
Estimated Number of Rows	84.0159

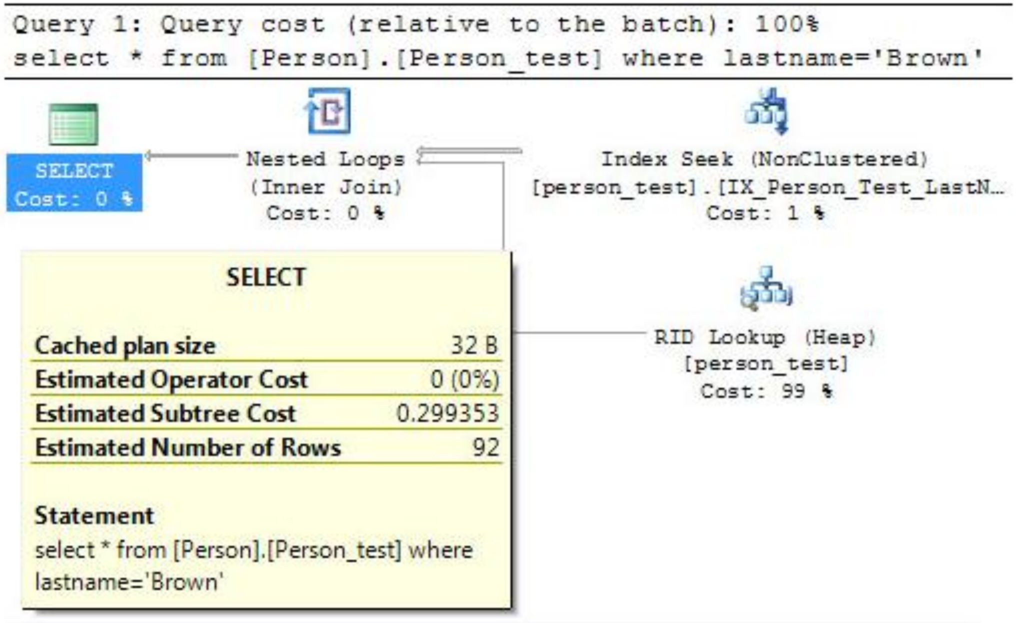
Statement

select * from [Person_Test] where lastname = 'Brown'

Table w/ non-clust. index on LastName select *



```
CREATE NONCLUSTERED INDEX [IX_Person_Test_LastName] ON [Person].[person_test]
(
        [LastName] ASC
) WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF,
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
GO
```



```
Query 1: Query cost (relative to the batch): 100%
select * from [Person].[Person_test] where lastname='Brown'
```

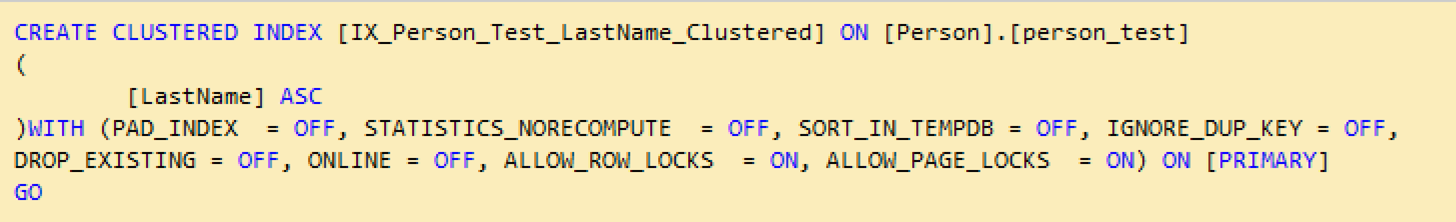
SELECT

Cached plan size	32 B
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0.299353
Estimated Number of Rows	92

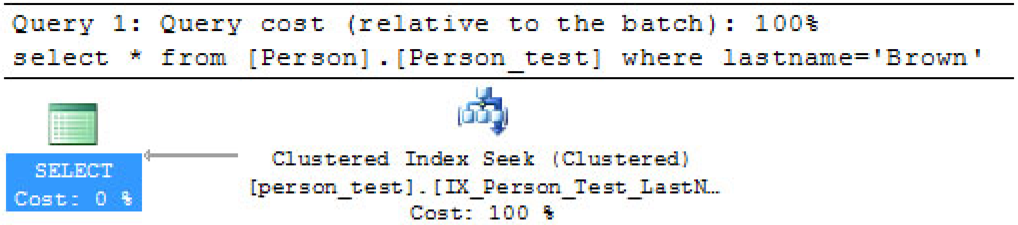
Statement

select * from [Person].[Person_test] where lastname='Brown'

Table w/ clust. index on LastName select * …

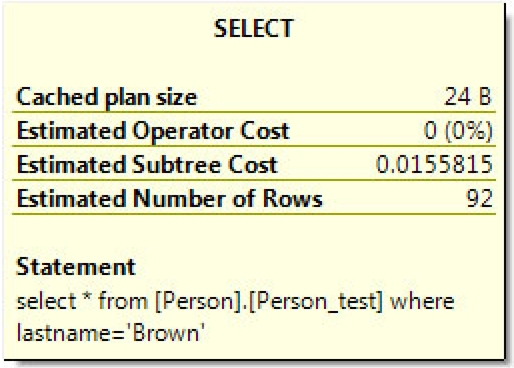


```
CREATE CLUSTERED INDEX [IX_Person_Test_LastName_Clustered] ON [Person].[person_test]
(
        [LastName] ASC
)WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF,
DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
GO
```



```
Query 1: Query cost (relative to the batch): 100%
select * from [Person].[Person_test] where lastname='Brown'
```

SELECT
Cost: 0 %
Clustered Index Seek (Clustered)
[person_test].[IX_Person_Test_LastN…
Cost: 100 %



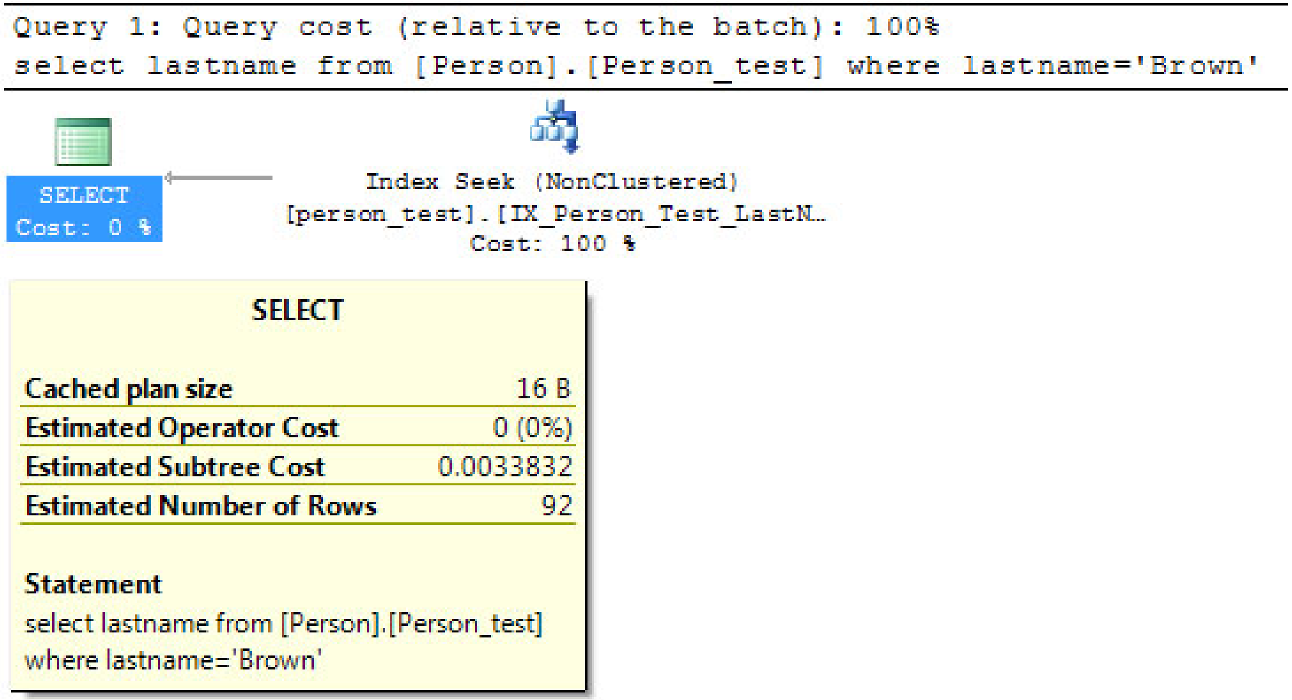
SELECT

Cached plan size	24 B
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0.0155815
Estimated Number of Rows	92

Statement

select * from [Person].[Person_test] where lastname='Brown'

Table w/ non-clust. index on LastName selecting LastName



```
Query 1: Query cost (relative to the batch): 100%
select lastname from [Person].[Person_test] where lastname='Brown'
```

SELECT
Cost: 0 %
Index Seek (NonClustered)
[person_test].[IX_Person_Test_LastN…
Cost: 100 %

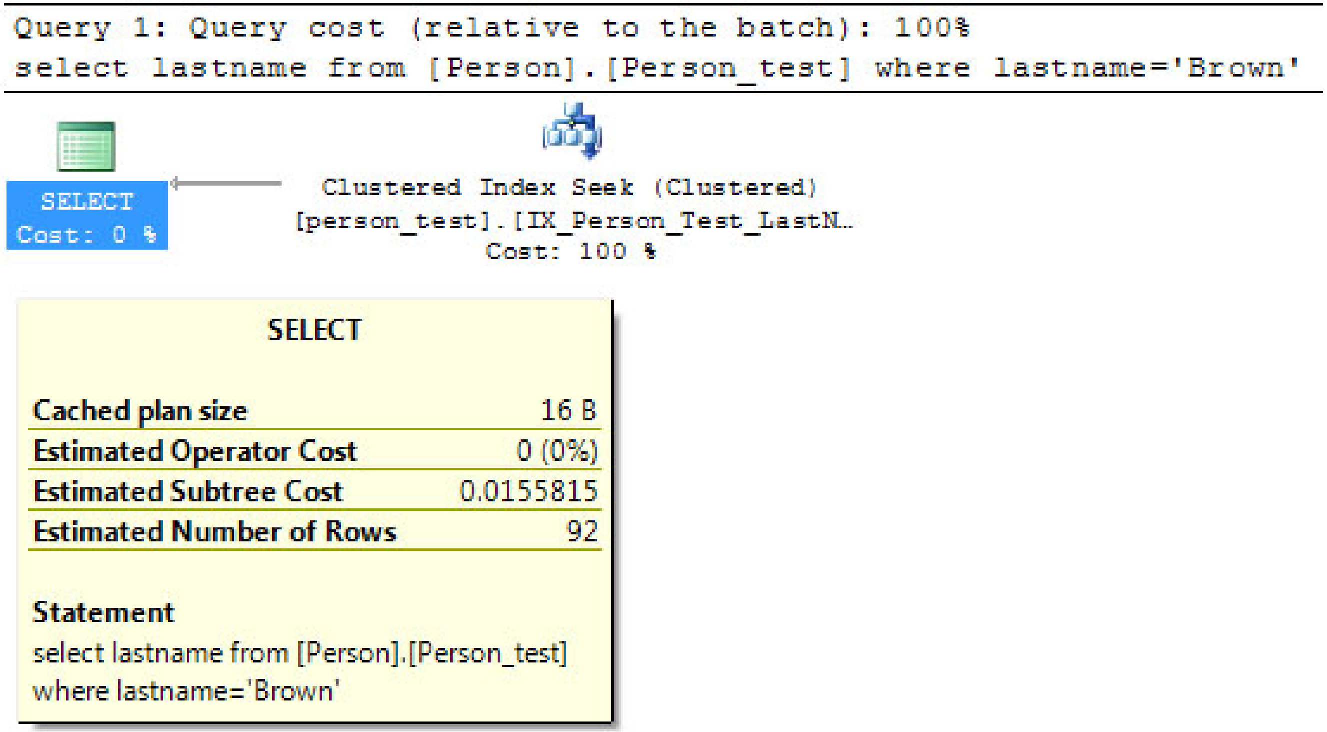
SELECT

Cached plan size	16 B
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0.0033832
Estimated Number of Rows	92

Statement

select lastname from [Person].[Person_test] where lastname='Brown'

Table w/ clust. Index on LastName selecting LastName



```
Query 1: Query cost (relative to the batch): 100%
select lastname from [Person].[Person_test] where lastname='Brown'
```

SELECT
Cost: 0 %
Clustered Index Seek (Clustered)
[person_test].[IX_Person_Test_LastN...
Cost: 100 %

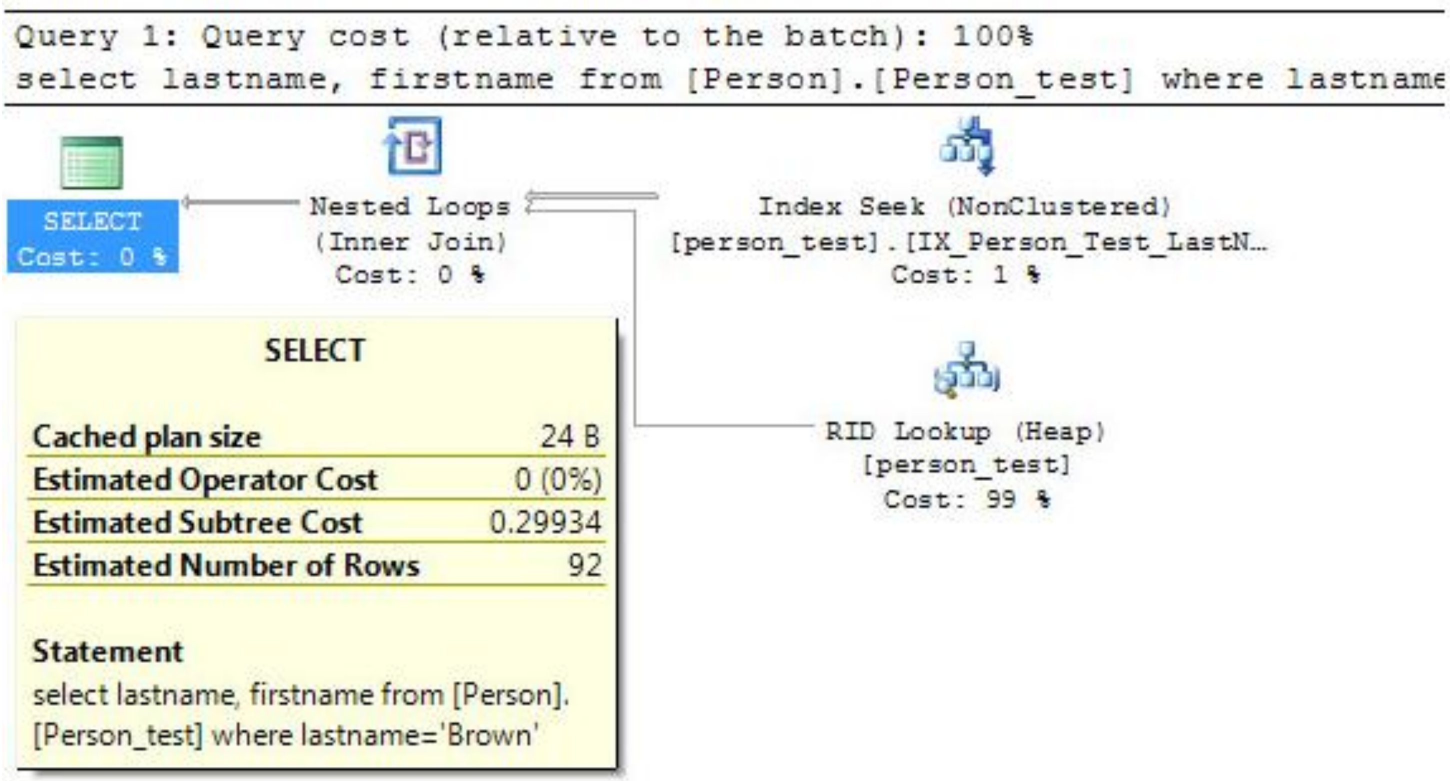
SELECT

Cached plan size	16 B
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0.0155815
Estimated Number of Rows	92

Statement

select lastname from [Person].[Person_test] where lastname='Brown'

Table w/ non-clust. Index on LastName selecting LastName and FirstName



```
Query 1: Query cost (relative to the batch): 100%
select lastname, firstname from [Person].[Person_test] where lastname
```

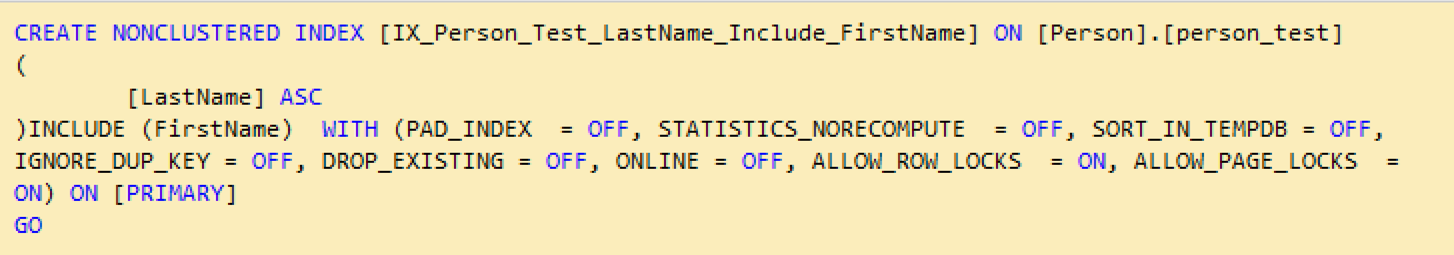
SELECT

Cached plan size	24 B
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0.29934
Estimated Number of Rows	92

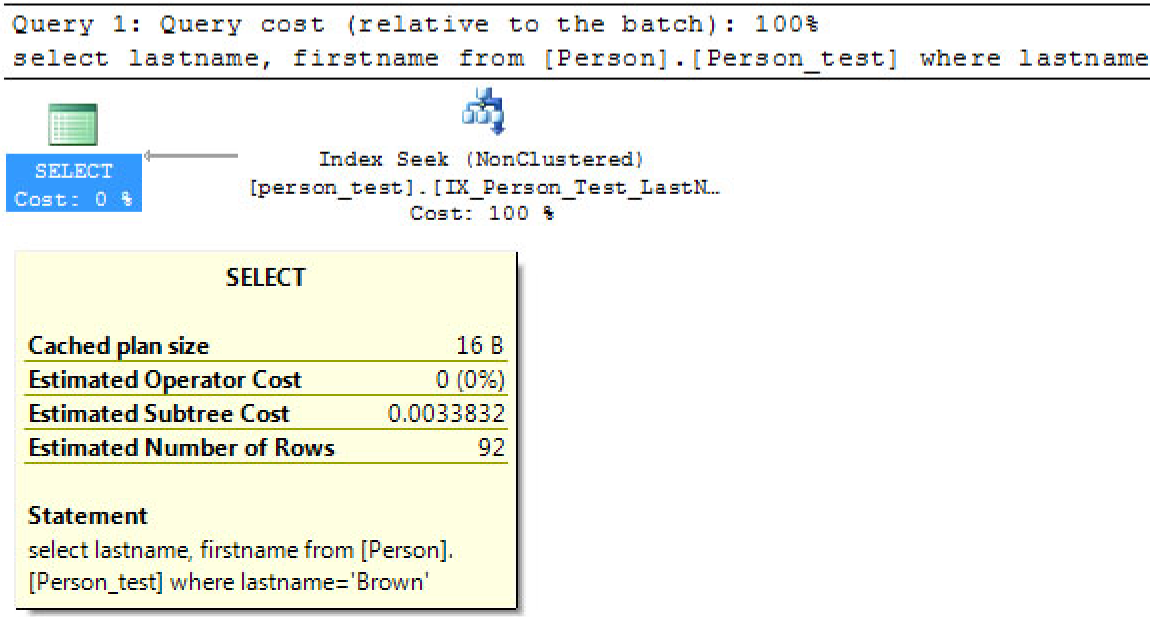
Statement

select lastname, firstname from [Person].[Person_test] where lastname='Brown'

Table w/ non-clust. Index on LastName including FirstName



```
CREATE NONCLUSTERED INDEX [IX_Person_Test_LastName_Include_FirstName] ON [Person].[person_test]
(
        [LastName] ASC
)INCLUDE (FirstName)  WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF, SORT_IN_TEMPDB = OFF,
IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF, ONLINE = OFF, ALLOW_ROW_LOCKS  = ON, ALLOW_PAGE_LOCKS  =
ON) ON [PRIMARY]
GO
```



```
Query 1: Query cost (relative to the batch): 100%
select lastname, firstname from [Person].[Person_test] where lastname
```

SELECT
Cost: 0 %
Index Seek (NonClustered)
[person_test].[IX_Person_Test_LastN...
Cost: 100 %

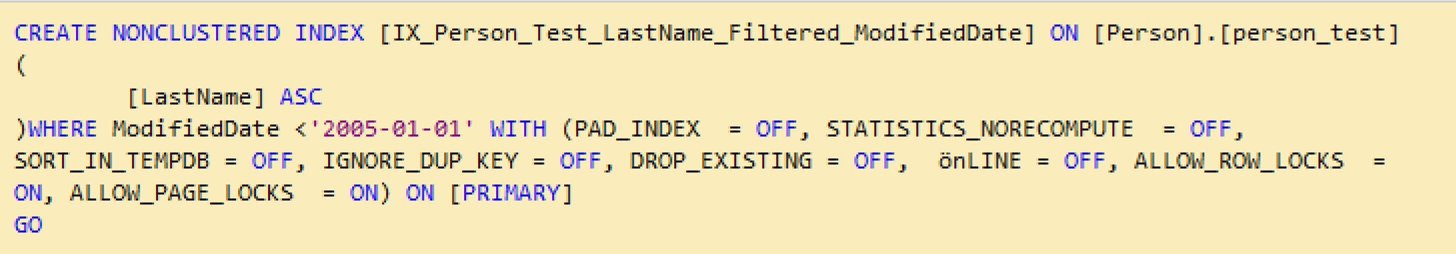
SELECT

Cached plan size	16 B
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0.0033832
Estimated Number of Rows	92

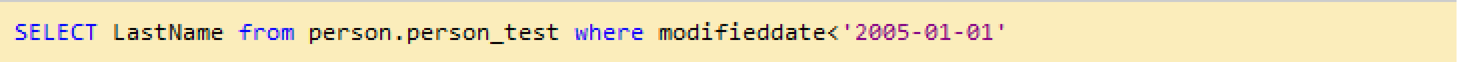
Statement

select lastname, firstname from [Person].[Person_test] where lastname='Brown'

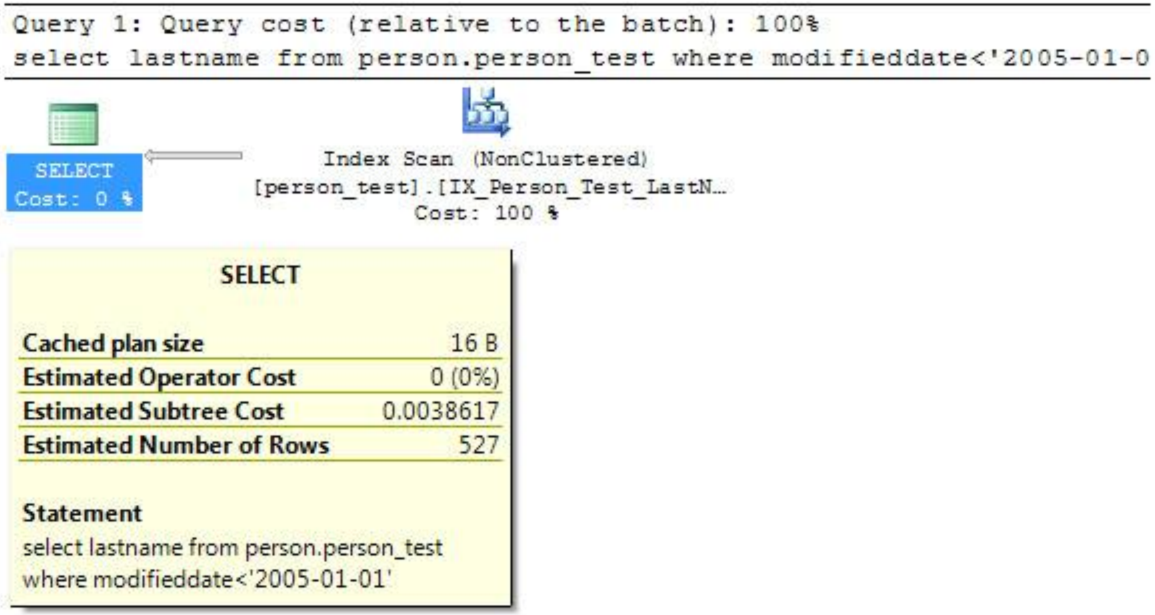
Table with filtered index



```
CREATE NONCLUSTERED INDEX [IX_Person_Test_LastName_Filtered_ModifiedDate] ON [Person].[person_test]
(
        [LastName] ASC
)WHERE ModifiedDate <'2005-01-01' WITH (PAD_INDEX  = OFF, STATISTICS_NORECOMPUTE  = OFF,
SORT_IN_TEMPDB = OFF, IGNORE_DUP_KEY = OFF, DROP_EXISTING = OFF,  önLINE = OFF, ALLOW_ROW_LOCKS  =
ON, ALLOW_PAGE_LOCKS  = ON) ON [PRIMARY]
GO
```



```
SELECT LastName from person.person_test where modifieddate<'2005-01-01'
```



```
Query 1: Query cost (relative to the batch): 100%
select lastname from person.person_test where modifieddate<'2005-01-0
```

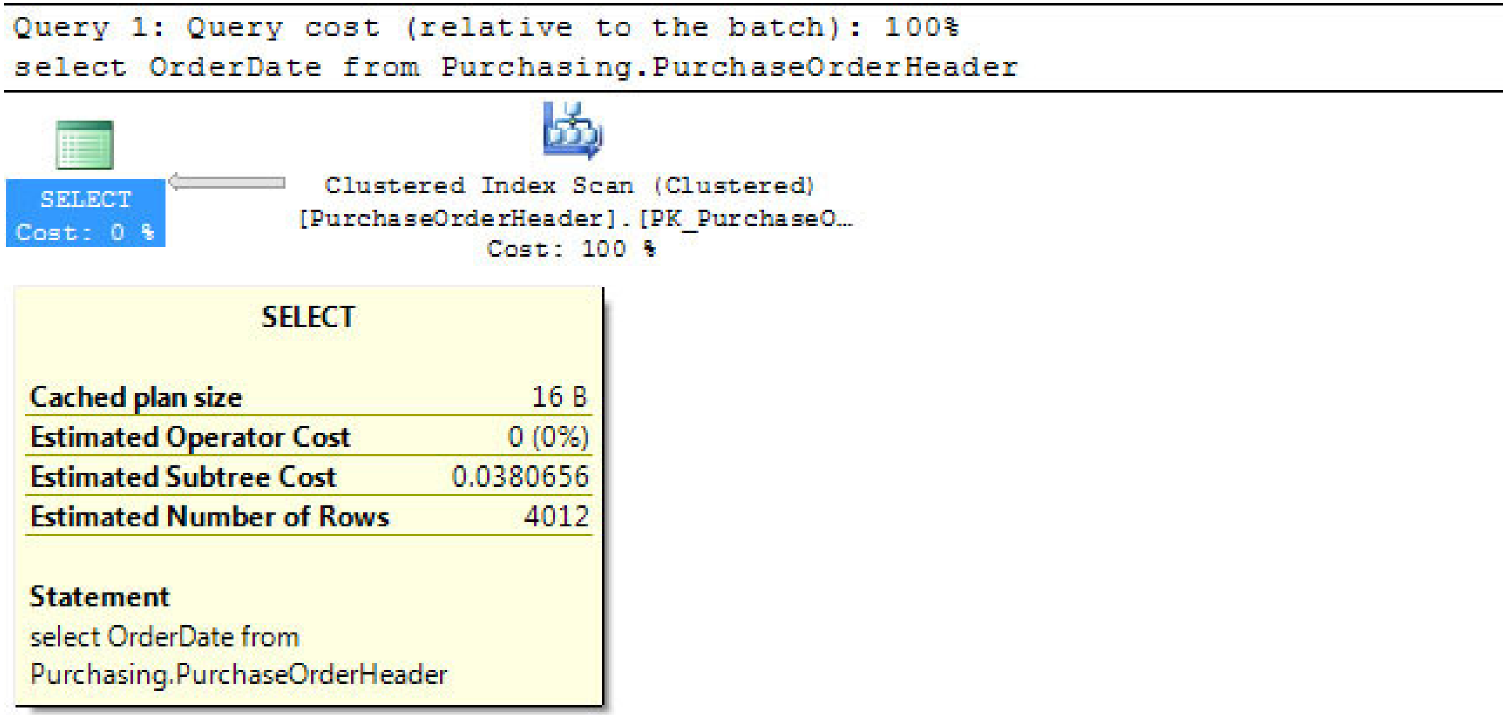
SELECT
Cost: 0 %
Index Scan (NonClustered)
[person_test].[IX_Person_Test_LastN...
Cost: 100 %

SELECT

Cached plan size	16 B
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0.0038617
Estimated Number of Rows	527

Statement
select lastname from person.person_test
where modifieddate<'2005-01-01'

Building indexes in Asc vs. Desc Order selecting all records


SELECT
Cost: 0 %
Clustered Index Scan (Clustered)
[PurchaseOrderHeader].[PK_PurchaseO…
Cost: 100 %

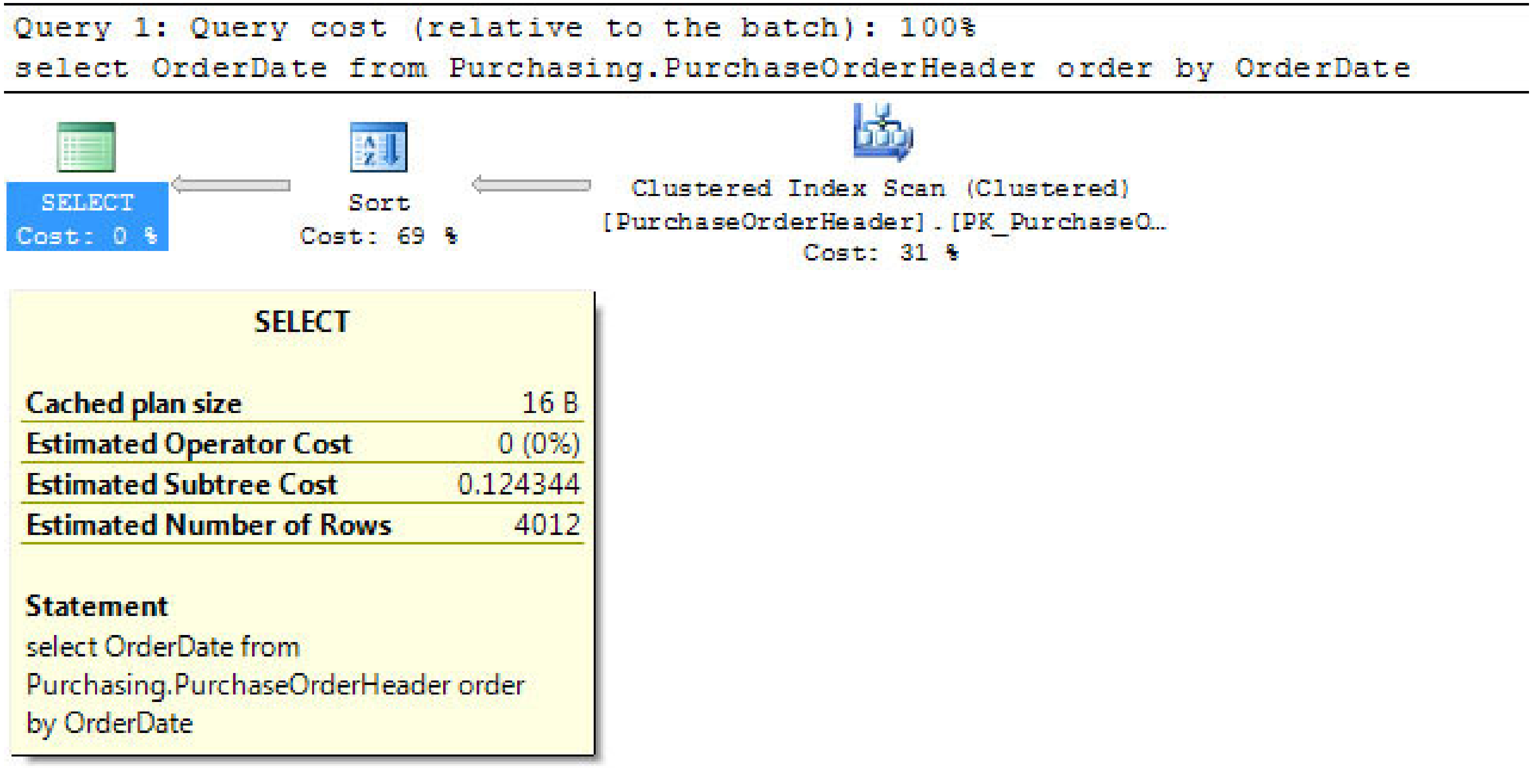
```
Query 1: Query cost (relative to the batch): 100%
select OrderDate from Purchasing.PurchaseOrderHeader
```

SELECT

Cached plan size	16 B
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0.0380656
Estimated Number of Rows	4012

Statement
select OrderDate from Purchasing.PurchaseOrderHeader

Building indexes in Asc vs. Desc Order select w/ ORDER BY ASC, no INDEX



```
Query 1: Query cost (relative to the batch): 100%
select OrderDate from Purchasing.PurchaseOrderHeader order by OrderDate
```

SELECT
Cost: 0 %
Sort
Cost: 69 %
Clustered Index Scan (Clustered)
[PurchaseOrderHeader].[PK_PurchaseO…
Cost: 31 %

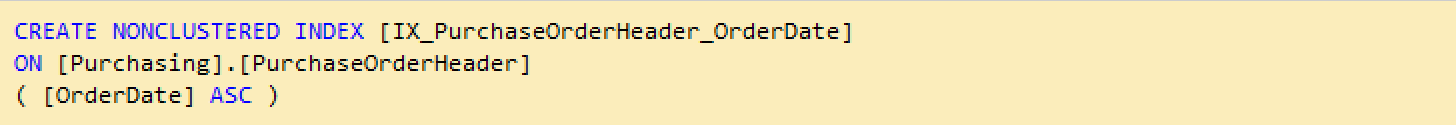
SELECT

Cached plan size	16 B
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0.124344
Estimated Number of Rows	4012

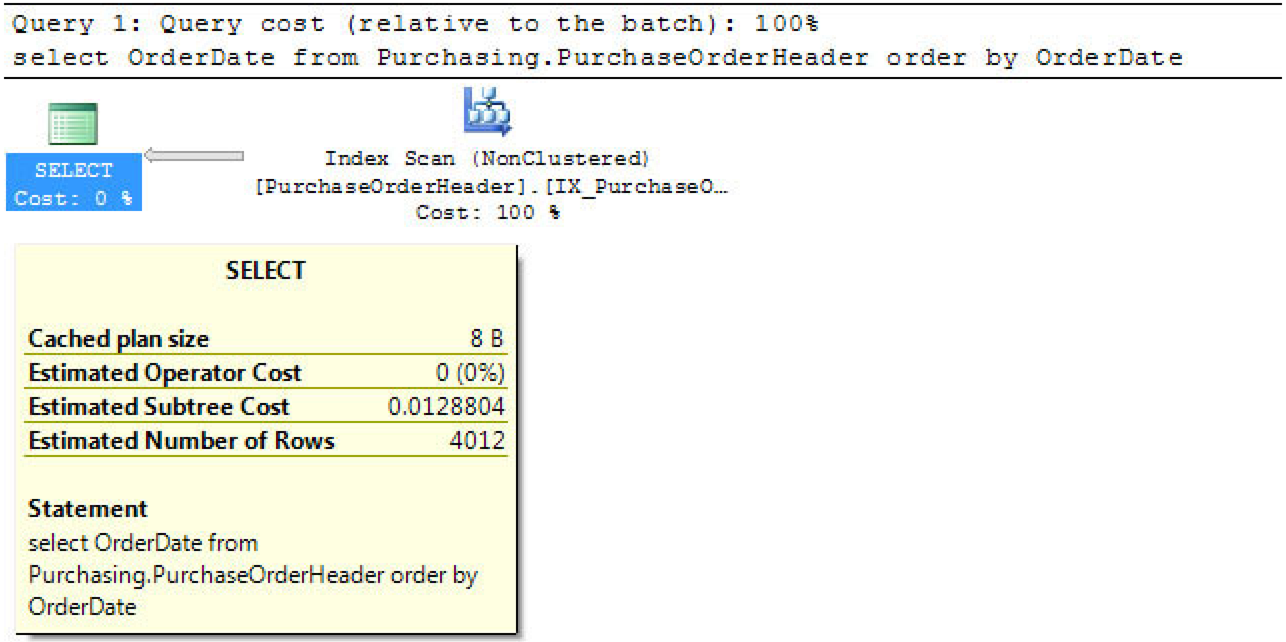
Statement

select OrderDate from Purchasing.PurchaseOrderHeader order by OrderDate

Building indexes in Asc vs. Desc Order select w/ ORDER BY ASC, with INDEX



```
CREATE NONCLUSTERED INDEX [IX_PurchaseOrderHeader_OrderDate]
ON [Purchasing].[PurchaseOrderHeader]
( [OrderDate] ASC )
```



```
Query 1: Query cost (relative to the batch): 100%
select OrderDate from Purchasing.PurchaseOrderHeader order by OrderDate
```

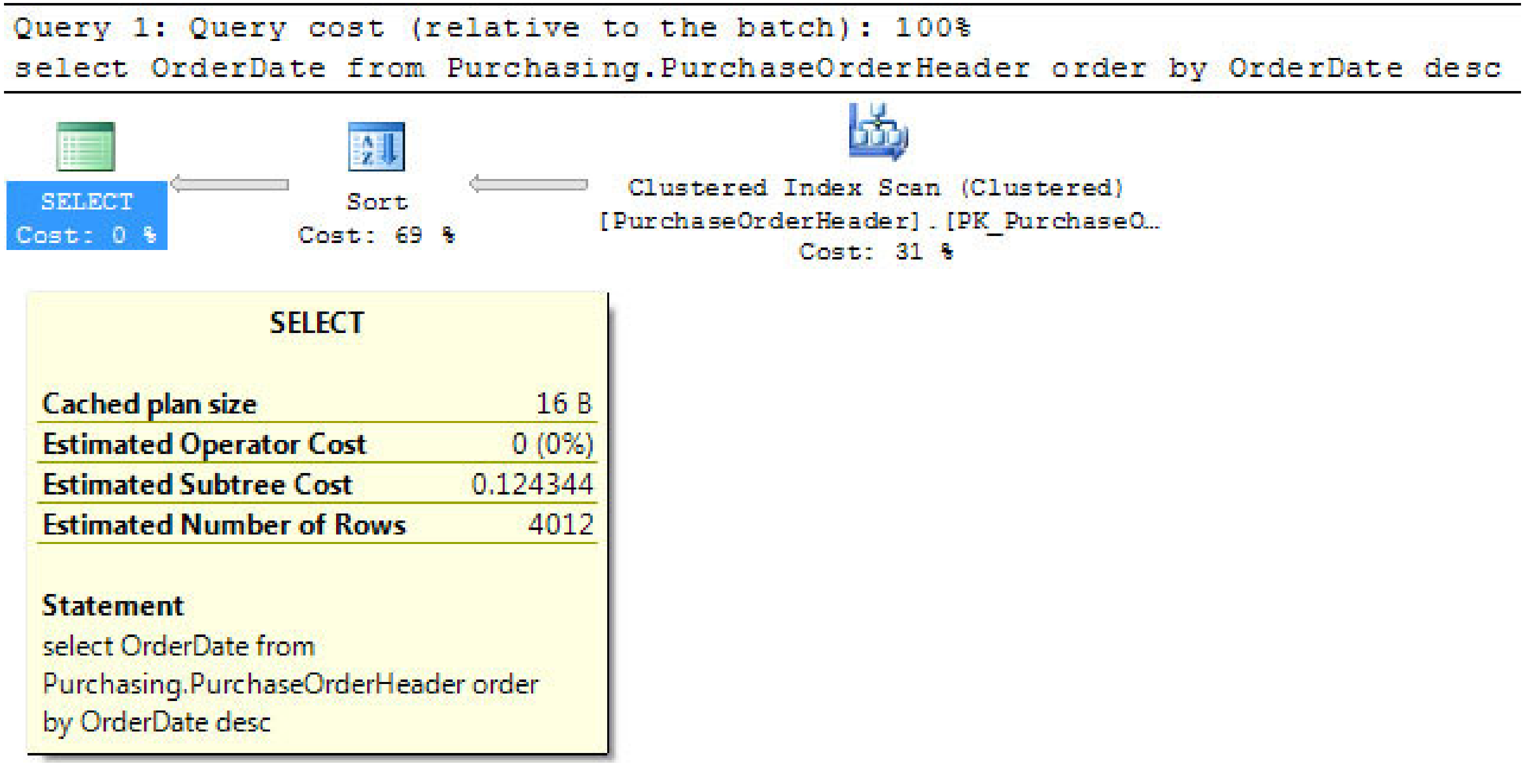
SELECT
Cost: 0 %
Index Scan (NonClustered)
[PurchaseOrderHeader].[IX_PurchaseO…
Cost: 100 %

SELECT

Cached plan size	8 B
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0.0128804
Estimated Number of Rows	4012

Statement
select OrderDate from Purchasing.PurchaseOrderHeader order by OrderDate

Building indexes in Asc vs. Desc Order select w/ ORDER BY DESC, no INDEX



```
Query 1: Query cost (relative to the batch): 100%
select OrderDate from Purchasing.PurchaseOrderHeader order by OrderDate desc
```

SELECT
Cost: 0 %
Sort
Cost: 69 %
Clustered Index Scan (Clustered)
[PurchaseOrderHeader].[PK_PurchaseO…
Cost: 31 %

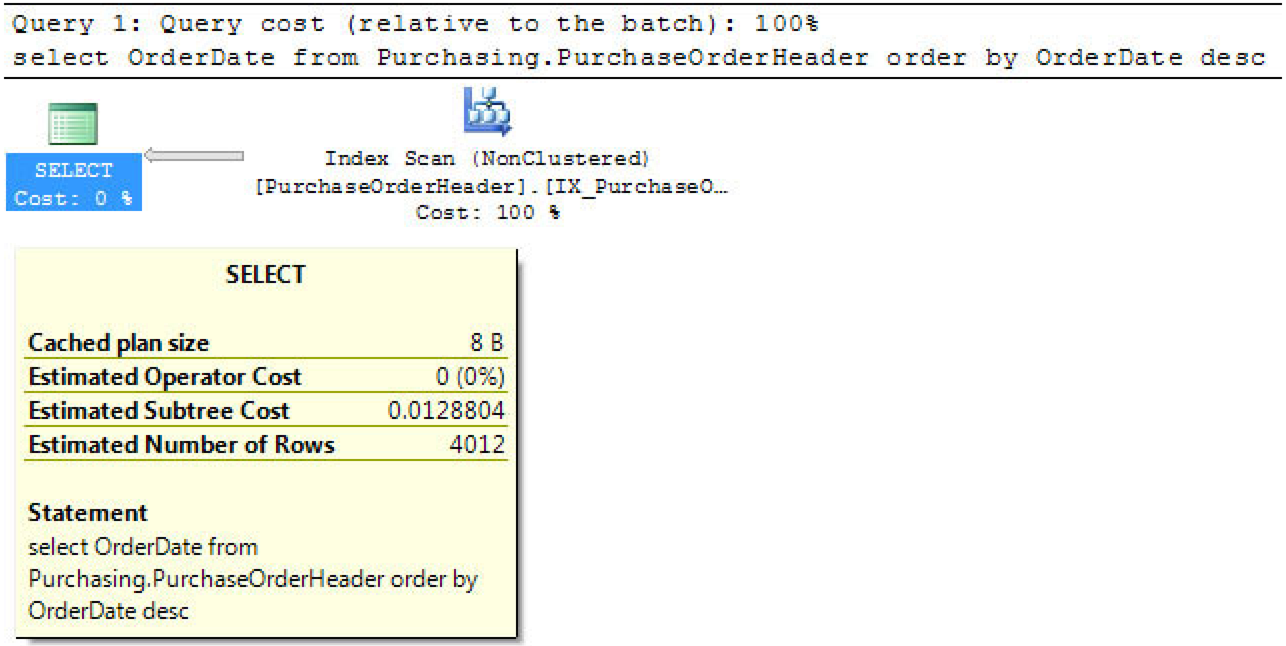
SELECT

Cached plan size	16 B
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0.124344
Estimated Number of Rows	4012

Statement
select OrderDate from Purchasing.PurchaseOrderHeader order by OrderDate desc

Building indexes in Asc vs. Desc Order select w/ ORDER BY DESC, with INDEX

```
CREATE NONCLUSTERED INDEX [IX_PurchaseOrderHeader_OrderDate]
ON [Purchasing].[PurchaseOrderHeader]
( [OrderDate] DESC )
```



```
Query 1: Query cost (relative to the batch): 100%
select OrderDate from Purchasing.PurchaseOrderHeader order by OrderDate desc
```

SELECT
Cost: 0 %
Index Scan (NonClustered)
[PurchaseOrderHeader].[IX_PurchaseO…
Cost: 100 %

SELECT

Cached plan size	8 B
Estimated Operator Cost	0 (0%)
Estimated Subtree Cost	0.0128804
Estimated Number of Rows	4012

Statement

select OrderDate from Purchasing.PurchaseOrderHeader order by OrderDate desc