Pop Emilia

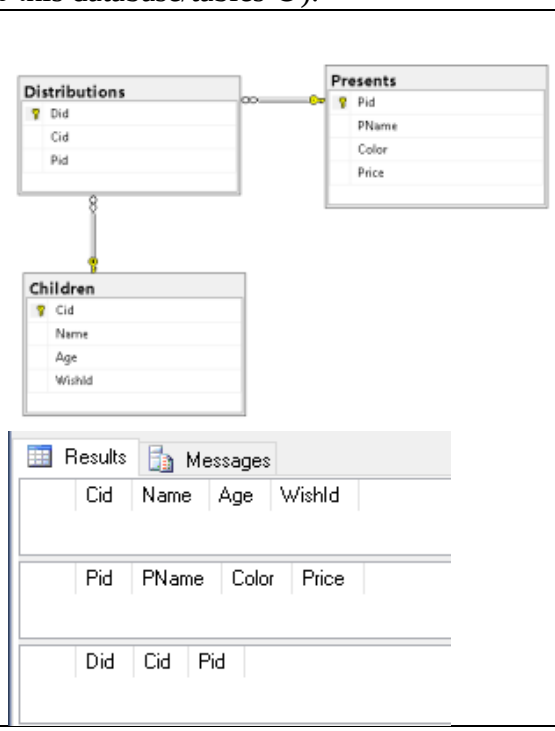
Laboratory 5 - Indexes

Work on 3 tables of the form Ta(aid, a2, ...), Tb(bid, b2, ...), Tc(cid, aid, bid, ...), where:

- aid, bid, cid, a2, b2 are integers;
- the primary keys are underlined;
- a2 is UNIQUE in Ta;
- aid and bid are foreign keys in Tc, referencing the corresponding primary keys in Ta and Tb, respectively.

In this laboratory you should create 3 tables that satisfy the conditions given in the request of your homework. Here, I consider the following database (please don't use this database/tables ☺):

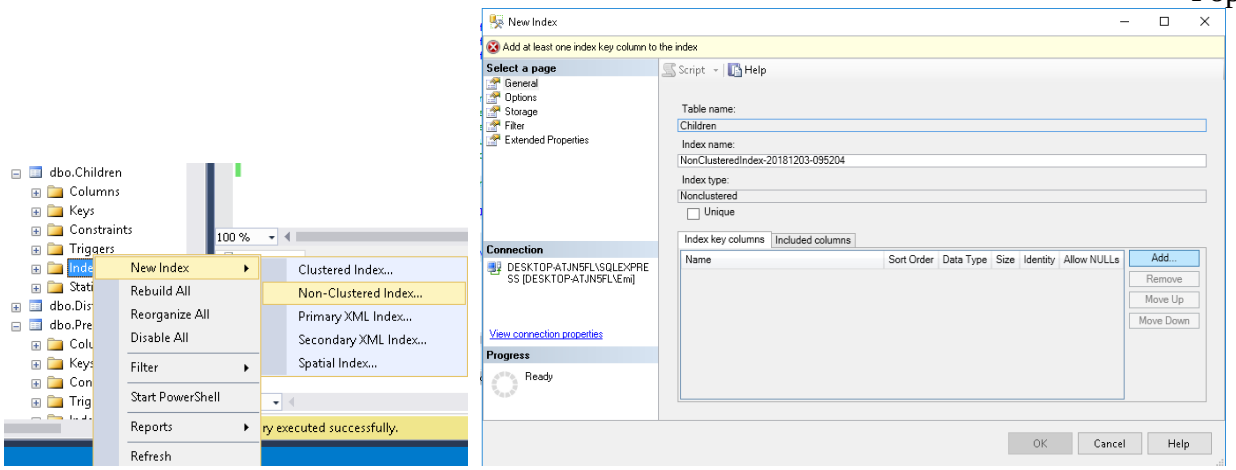
```
create database Lab5_IE
go
use Lab5_IE
go
CREATE TABLE Children(-- Ta
Cid INT PRIMARY KEY IDENTITY, -- aid
Name VARCHAR(50),
Age INT, -- a2
WishId INT UNIQUE
)
CREATE TABLE Presents(-- Tb
Pid INT PRIMARY KEY IDENTITY, -- bid
PName VARCHAR(50),
Color VARCHAR(50),
Price INT-- b2
)
CREATE TABLE Distributions(-- Tc
Did INT PRIMARY KEY IDENTITY, -- cid
Cid INT FOREIGN KEY REFERENCES Children(Cid), -- aid
Pid INT FOREIGN KEY REFERENCES Presents(Pid) -- bid
)
select * from Children
select * from Presents
select * from Distributions
```



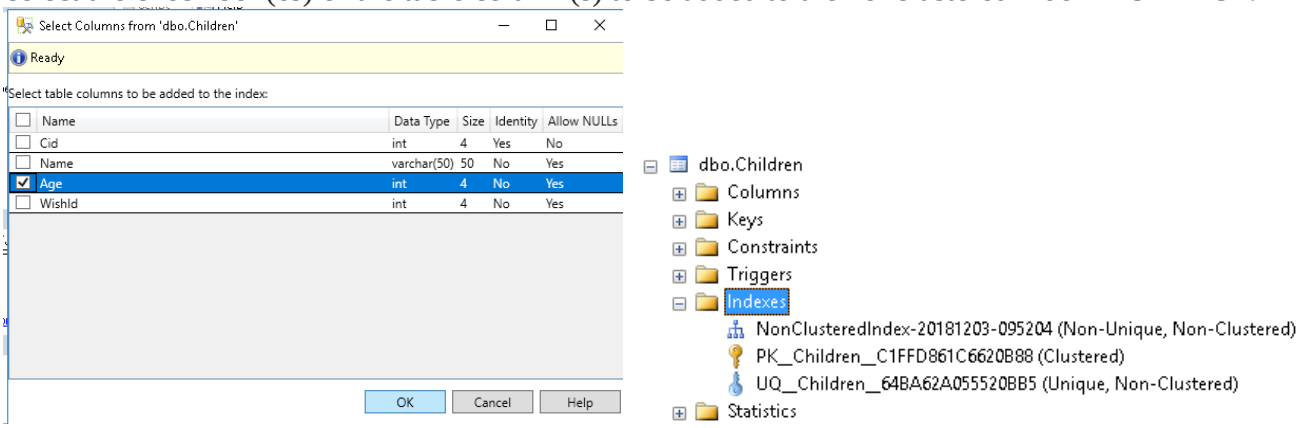
a. Write 5 queries on Ta such that their corresponding execution plans contain the following operators:
   - clustered index scan
   - clustered index seek;
   - nonclustered index scan;
   - nonclustered index seek;
   - key lookup.

To create a *nonclustered index* by using Object Explorer

- Choose the database -> Tables (folder) -> expand the table that will be used to create a non-clustered index -> Right-click the Indexes folder -> New Index -> select Non-Clustered Index…



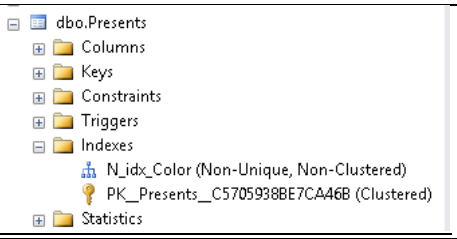
- In the New Index dialog box -> General page -> Index name box (=enter the name of the new index)
- Under Index key columns -> click Add… -> In the Select Columns from table_name dialog box -> select the check box(es) of the table column(s) to be added to the nonclustered index ->Ok -> Ok.



To create a *nonclustered index* on a table by using Transact-SQL

- Choose the database -> New Query -> write the code -> Execute

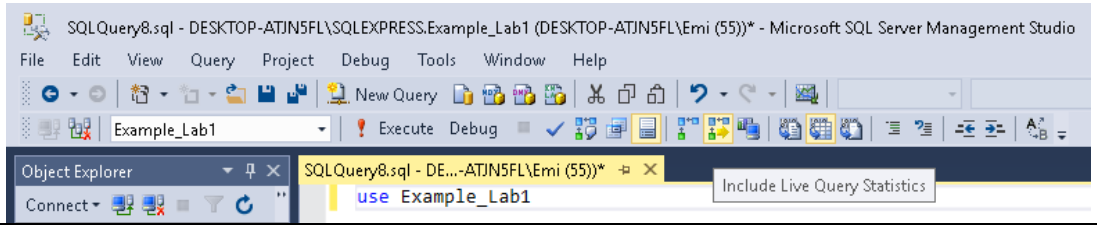
```
-- Find an existing index named N_idx_Color and delete it if found.
IF EXISTS (SELECT name FROM sys.indexes WHERE name = N'N_idx_Color')
    DROP INDEX N_idx_Color ON Presents;
GO
-- Create a nonclustered index called N_idx_Color on the Presents table using the Color column.
CREATE NONCLUSTERED INDEX N_idx_Color ON Presents(Color);
GO
```




```
Command(s) completed successfully.
```

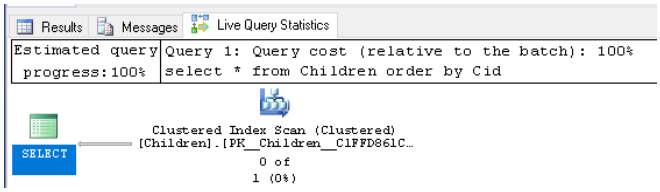
Check the Clustered/NonClustered indexes

– check **Include Live Query Statistics –** when a query is open (and the properties can be check for the exeuted query). After an operation (update, order by, …), the order of the records is modified. For example, the indexes become unordered (1,2,3 -> 3,1,2). To choose the best index you can work with Include Live Query Statistics.




```
-- check Include Live Query Statistics
-- by moving the mouse through the indexes, you can check properties of these

-- on Cid (the primary key) there is a clustered index
select * from Children order by Cid
```



Clustered Index Scan (Clustered)

Scanning a clustered index, entirely or only a range.

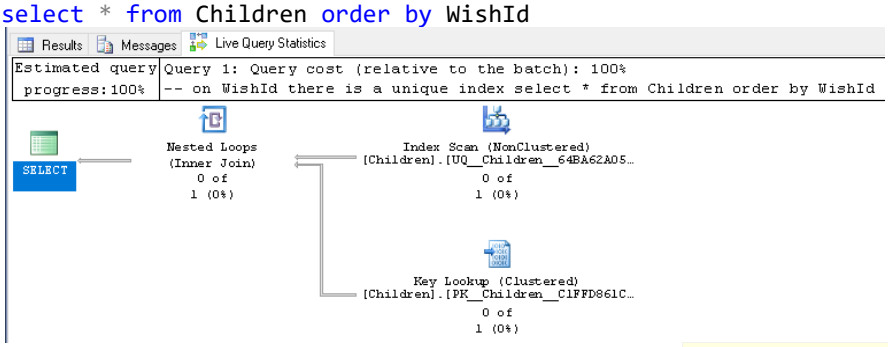
Estimated operator progress: 100%

Physical Operation	Clustered Index Scan
Logical Operation	Clustered Index Scan
Actual Execution Mode	Row
Estimated Execution Mode	Row
Storage	RowStore
Actual Number of Rows	0
Actual Number of Batches	0
Estimated I/O Cost	0.003125
Estimated Operator Cost	0.0032831 (100%)
Estimated Subtree Cost	0.0032831
Estimated CPU Cost	0.0001581
Estimated Number of Executions	1
Number of Executions	1
Estimated Number of Rows	1
Estimated Row Size	48 B
Actual Rebinds	0
Actual Rewinds	0
Ordered	True
Node ID	0

Object
[Lab4_MIE].[dbo].[Children].[PK__Children__C1FFD861C6620B88]

Output List
[Lab4_MIE].[dbo].[Children].Cid, [Lab4_MIE].[dbo].[Children].Name, [Lab4_MIE].[dbo].[Children].Age, [Lab4_MIE].[dbo].[Children].WishId

```
-- on WishId there is a unique index
select * from Children order by WishId
```



Nested Loops

For each row in the top (outer) input, scan the bottom (inner) input, and output matching rows.

Estimated operator progress: 100%

Physical Operation	Nested Loops
Logical Operation	Inner Join
Estimated Execution Mode	Row
Actual Number of Rows	0
Estimated I/O Cost	0
Estimated Operator Cost	0.0000042 (0%)
Estimated Subtree Cost	0.0065704
Estimated CPU Cost	0.0000042
Estimated Number of Executions	1
Number of Executions	1
Estimated Number of Rows	1
Estimated Row Size	48 B
Node ID	0

Output List
[Lab4_MIE].[dbo].[Children].Cid, [Lab4_MIE].[dbo].[Children].Name, [Lab4_MIE].[dbo].[Children].Age, [Lab4_MIE].[dbo].[Children].WishId

Outer References
[Lab4_MIE].[dbo].[Children].Cid

Index Scan (NonClustered)

Scan a nonclustered index, entirely or only a range.

Estimated operator progress: 100%

Physical Operation	Index Scan
Logical Operation	Index Scan
Estimated Execution Mode	Row
Storage	RowStore
Actual Number of Rows	0
Estimated Operator Cost	0.0032831 (50%)
Estimated I/O Cost	0.003125
Estimated CPU Cost	0.0001581
Estimated Subtree Cost	0.0032831
Number of Executions	1
Estimated Number of Executions	1
Estimated Number of Rows	1
Estimated Row Size	15 B
Ordered	True
Node ID	1

Object
[Lab4_MIE].[dbo].[Children].[UQ__Children__64BA62A055520BB5]

Output List
[Lab4_MIE].[dbo].[Children].Cid, [Lab4_MIE].[dbo].[Children].WishId

Key Lookup (Clustered)

Uses a supplied clustering key to lookup on a table that has a clustered index.

Estimated operator progress: 100%

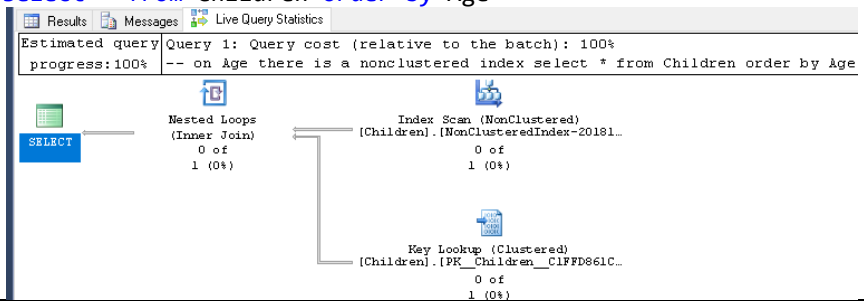
Physical Operation	Key Lookup
Logical Operation	Key Lookup
Estimated Execution Mode	Row
Storage	RowStore
Actual Number of Rows	0
Estimated Operator Cost	0.0032831 (50%)
Estimated I/O Cost	0.003125
Estimated CPU Cost	0.0001581
Estimated Subtree Cost	0.0032831
Number of Executions	0
Estimated Number of Executions	1
Estimated Number of Rows	1
Estimated Row Size	40 B
Ordered	True
Node ID	3

Object
[Lab4_MIE].[dbo].[Children].[PK__Children__C1FFD861C6620B88]

Output List
[Lab4_MIE].[dbo].[Children].Name, [Lab4_MIE].[dbo].[Children].Age

Seek Predicates
Seek Keys[1]: Prefix: [Lab4_MIE].[dbo].[Children].Cid = Scalar Operator([Lab4_MIE].[dbo].[Children].[Cid])

```
-- on Age there is a nonclustered index
select * from Children order by Age
```



Index on primary key = it is created automatically when the primary key is created = index clustered
Index on unique key = it is created automatically when the unique constraint is created = index nonclustered

Check indexes

```
-- check the indexes (nonclustered) for the database used
SELECT TableName = t.name, IndexName = ind.name, IndexId = ind.index_id, ColumnId = ic.index_column_id,
     ColumnName = col.name, ind.*, ic.*, col.*
FROM sys.indexes ind
INNER JOIN sys.index_columns ic ON  ind.object_id = ic.object_id and ind.index_id = ic.index_id
INNER JOIN sys.columns col ON ic.object_id = col.object_id and ic.column_id = col.column_id
INNER JOIN sys.tables t ON ind.object_id = t.object_id
WHERE ind.is_primary_key = 0 AND ind.is_unique = 0 AND ind.is_unique_constraint = 0
     AND t.is_ms_shipped = 0
ORDER BY t.name, ind.name, ind.index_id, ic.index_column_id;
```

Results | Messages

	TableName	IndexName	IndexId	ColumnId	ColumnName	object_id	name	index_id	type	type_desc	is_unique	data
1	Children	NonClusteredIndex-20181203-095204	3	1	Age	565577053	NonClusteredIndex-20181203-095204	3	2	NONCLUSTERED	0	1
2	Presents	N_idx_Color	2	1	Color	613577224	N_idx_Color	2	2	NONCLUSTERED	0	1

```
-- all the indexes from table Children
select i2.name, i1.user_scans, i1.user_seeks, i1.user_updates,i1.last_user_scan,i1.last_user_seek,
i1.last_user_update
from sys.dm_db_index_usage_stats i1
inner join sys.indexes i2 on i1.index_id = i2.index_id
where i1.object_id = OBJECT_ID('Children') and i1.object_id = i2.object_id
```

Results | Messages

	name	user_scans	user_seeks	user_updates	last_user_scan	last_user_seek	last_user_update
1	NonClusteredIndex-20181203-095204	2	0	0	2018-12-03 10:57:06.990	NULL	NULL
2	UQ__Children__64BA62A055520BB5	3	0	0	2018-12-03 10:59:33.433	NULL	NULL
3	PK__Children__C1FFD861C6620B88	4	0	0	2018-12-03 10:58:24.263	NULL	NULL

```
-- all the indexes from the current database
SELECT OBJECT_NAME(A.[OBJECT_ID]) AS [OBJECT NAME], I.[NAME] AS [INDEX NAME], A.LEAF_INSERT_COUNT,
       A.LEAF_UPDATE_COUNT, A.LEAF_DELETE_COUNT
FROM   SYS.DM_DB_INDEX_OPERATIONAL_STATS (NULL,NULL,NULL,NULL ) A
       INNER JOIN SYS.INDEXES AS I ON I.[OBJECT_ID] = A.[OBJECT_ID] AND I.INDEX_ID = A.INDEX_ID
WHERE  OBJECTPROPERTY(A.[OBJECT_ID],'IsUserTable') = 1
```

Results | Messages

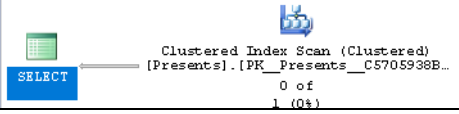
	OBJECT NAME	INDEX NAME	LEAF_INSERT_COUNT	LEAF_UPDATE_COUNT	LEAF_DELETE_COUNT
1	Presents	PK__Presents__C5705938BE7CA46B	0	0	0
2	Children	UQ__Children__64BA62A055520BB5	432	0	0
3	Children	PK__Children__C1FFD861C6620B88	432	0	0
4	Presents	N_idx_Color	0	0	0
5	Children	PK__Children__C1FFD861C6620B88	0	0	0
6	Children	UQ__Children__64BA62A055520BB5	0	0	0
7	Presents	PK__Presents__C5705938BE7CA46B	0	0	0
8	Distributions	PK__Distribu__C031221883C8CCCE	0	0	0
9	sysdiagrams	PK__sysdiagr__C2B05B61B7EE2DE6	1	0	0
10	sysdiagrams	UK_principal_name	1	0	0
11	Presents	N_idx_Color	0	0	0
12	Children	NonClusteredIndex-20181203-095204	0	0	0

b. Write a query on table Tb with a WHERE clause of the form *WHERE b2 = value* and analyze its execution plan. Create a nonclustered index that can speed up the query. Recheck the query's execution plan (operators, SELECT's *estimated subtree cost*).

```
select *
from Presents
where Price=8
```

Results | Messages | Live Query Statistics

Estimated query progress:100% | Query 1: Query cost (relative to the batch): 100% (@1 tinyint)SELECT * FROM [Presents] WHERE [Price]=@1




```
-- Find an existing index named N_idx_Price and delete it if found.
IF EXISTS (SELECT name FROM sys.indexes WHERE name = N'N_idx_Price')
    DROP INDEX N_idx_Price ON Presents;
GO
-- Create a nonclustered index called N_idx_Price on the Presents table using the Price column.
CREATE NONCLUSTERED INDEX N_idx_Price ON Presents(Price);
GO
```

```
select * from Presents where Price=8
```

Results | Messages | Live Query Statistics

Estimated query progress:100% | Query 1: Query cost (relative to the batch): 100% SELECT * FROM [Presents] WHERE [Price]=@1


```
-- list the index plan
SET NOCOUNT ON;
GO
SET SHOWPLAN_ALL ON;
GO
--
SELECT * FROM Presents WHERE Price BETWEEN 6 AND 15;
GO
--
SET SHOWPLAN_ALL OFF
GO
```

Results | Messages

	StmtText	StmtId	NodeId	Parent	PhysicalOp	LogicalOp	Argument
1	SELECT * FROM Presents WHERE Price BETWEEN 6 AND 15;	1	1	0	NULL	NULL	1
2	|--Clustered Index Scan(OBJECT:([Lab4_MIE].[dbo].[Presents].[PK__Presents__C5705938...	1	2	1	Clustered Index Scan	Clustered Index Scan	OBJECT:([Lab4_MIE].[db

Results | Messages

	Pid	PName	Color	Price
1	1	Kitty Cat	blue	9
2	2	Apples	red	8

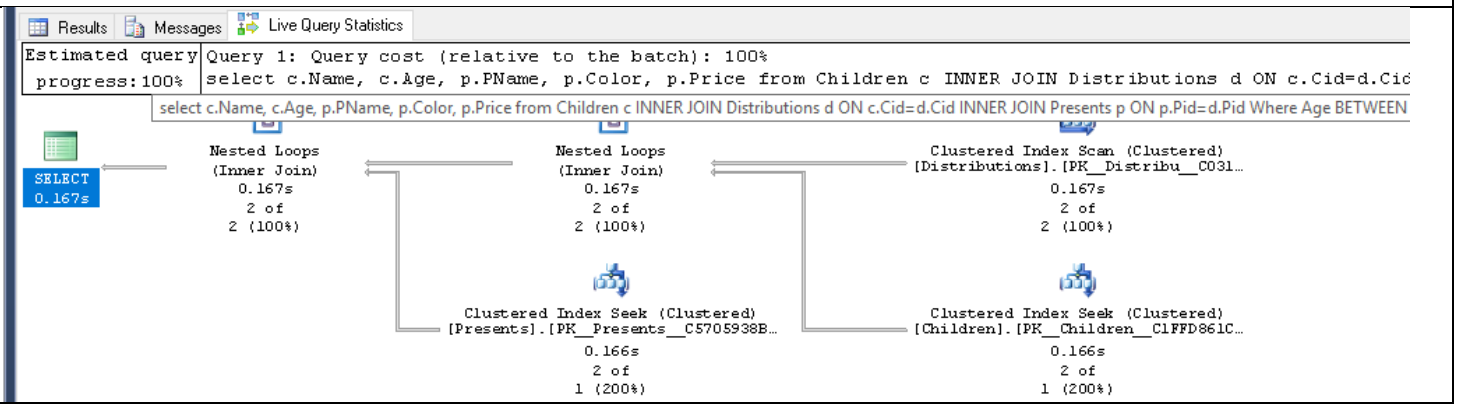
	Rows	Executes	StmtText	StmtId	NodeId	Parent	PhysicalOp	LogicalOp	Argument
1	2	1	SELECT * FROM [Presents] WHERE [Price]>=@1 AND [...	1	1	0	NULL	NULL	NULL
2	2	1	|--Clustered Index Scan(OBJECT:([Lab4_MIE].[dbo].[Pre...	1	2	1	Clustered Index Scan	Clustered Index Scan	OBJECT:([Lab4_MIE].[dbo].[Presen

	Microsoft SQL Server 2005 XML Showplan
1	<ShowPlanXML xmlns="http://schemas.microsoft.com/sqlserver/2004/07/showplan" Version="1.5" Build="13.0.1601.5"><BatchSequence><Batch><Statements><StmtSimple S...

c. Create a view that joins at least 2 tables. Check whether existing indexes are helpful by using *dynamic management* objects; if this is not the case, reassess existing indexes / examine the cardinality of the tables.

```
create view test
as
select c.Name, c.Age, p.PName, p.Color, p.Price
from Children c INNER JOIN Distributions d ON c.Cid=d.Cid
INNER JOIN Presents p ON p.Pid=d.Pid
Where Age BETWEEN 7 and 14 OR Price>5
go
```

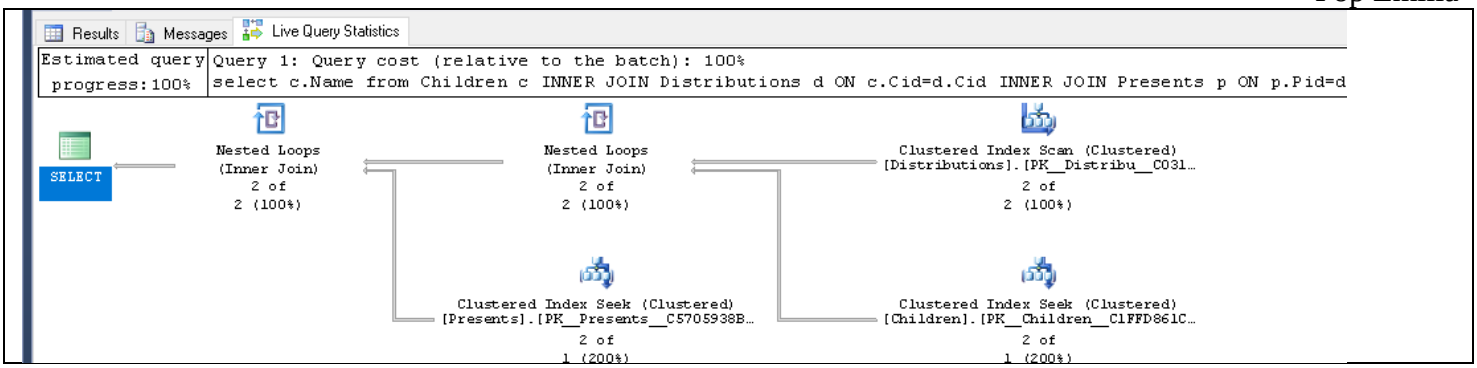
select * from test



Usualy you should apply indexes on the fields involved in Where conditions or Order By or Group By.

```
-- Find an existing index named N_idx_Name and delete it if found.
IF EXISTS (SELECT name FROM sys.indexes WHERE name = N'N_idx_Name')
    DROP INDEX N_idx_Name ON Children;
GO
-- Create a nonclustered index called N_idx_Color on the Presents table using the Color column.
CREATE NONCLUSTERED INDEX N_idx_Name ON Children(Name);
GO
```

```
select c.Name
from Children c INNER JOIN Distributions d ON c.Cid=d.Cid
INNER JOIN Presents p ON p.Pid=d.Pid
Where Age BETWEEN 7 and 14 OR Price>5
```



sp_indexes, sp_helpindex – functions used to check indexes

```
-- CHECK THE INDEXES for the given table
EXEC sp_helpindex 'Children'
```

Results | Messages

	index_name	index_description	index_keys
1	N_idx_Name	nonclustered located on PRIMARY	Name
2	NonClusteredIndex-20181203-095204	nonclustered located on PRIMARY	Age
3	PK__Children__C1FFD861C6620B88	clustered, unique, primary key located on PRIMARY	Cid
4	UQ__Children__64BA62A055520BB5	nonclustered, unique, unique key located on PRIMA...	WishId