<https://docs.microsoft.com/en-us/sql/relational-databases/tables/lesson-1-3-optimizing-the-neworg-table>

# Lesson 1-3 - Optimizing the NewOrg Table

2017-3-6 3 min to read Contributors

* [BYHAM](https://github.com/BYHAM)

* [Craig Guyer](https://github.com/craigg-msft)

The **NewOrd** table that you created in the [Populating a Table with Existing Hierarchical Data](https://docs.microsoft.com/en-us/sql/relational-databases/tables/lesson-1-2-populating-a-table-with-existing-hierarchical-data) task contains all the employee information, and represents the hierarchical structure by using a **hierarchyid** data type. This task adds new indexes to support searches on the **hierarchyid** column.

## Clustered Index

The **hierarchyid** column (**OrgNode**) is the primary key for the **NewOrg** table. When the table was created, it contained a clustered index named **PK\_NewOrg\_OrgNode** to enforce the uniqueness of the **OrgNode** column. This clustered index also supports a depth-first search of the table.

## Nonclustered Index

This step creates two nonclustered indexes to support typical searches.

#### To index the NewOrg table for efficient searches

1. To help queries at the same level in the hierarchy, use the [GetLevel](https://docs.microsoft.com/en-us/sql/t-sql/data-types/getlevel-database-engine) method to create a computed column that contains the level in the hierarchy. Then, create a composite index on the level and the **Hierarchyid**. Run the following code to create the computed column and the breadth-first index:

Copy

ALTER TABLE NewOrg

ADD H\_Level AS OrgNode.GetLevel() ;

CREATE UNIQUE INDEX EmpBFInd

ON NewOrg(H\_Level, OrgNode) ;

GO

1. Create a unique index on the **EmployeeID** column. This is the traditional singleton lookup of a single employee by **EmployeeID** number. Run the following code to create an index on **EmployeeID**:

Copy

CREATE UNIQUE INDEX EmpIDs\_unq ON NewOrg(EmployeeID) ;

GO

1. Run the following code to retrieve data from the table in the order of each of the three indexes:

Copy

SELECT OrgNode.ToString() AS LogicalNode,

OrgNode, H\_Level, EmployeeID, LoginID

FROM NewOrg

ORDER BY OrgNode;

SELECT OrgNode.ToString() AS LogicalNode,

OrgNode, H\_Level, EmployeeID, LoginID

FROM NewOrg

ORDER BY H\_Level, OrgNode;

SELECT OrgNode.ToString() AS LogicalNode,

OrgNode, H\_Level, EmployeeID, LoginID

FROM NewOrg

ORDER BY EmployeeID;

GO

1. Compare the result sets to see how the order is stored in each type of index. Only the first four rows of each output follow.

Here is the result set.

Depth-first index: Employee records are stored adjacent to their manager.

LogicalNode OrgNode H\_Level EmployeeID LoginID

/ 0x 0 1 zarifin

/1/ 0x58 1 2 tplate

/1/1/ 0x5AC0 2 4 schai

/1/1/1/ 0x5AD6 3 9 jwang

/1/1/2/ 0x5ADA 3 10 malexander

/1/2/ 0x5B40 2 5 elang

/1/3/ 0x5BC0 2 6 gsmits

/2/ 0x68 1 3 hjensen

/2/1/ 0x6AC0 2 7 sdavis

/2/2/ 0x6B40 2 8 norint

**EmployeeID**-first index: Rows are stored in **EmployeeID** sequence.

LogicalNode OrgNode H\_Level EmployeeID LoginID

/ 0x 0 1 zarifin

/1/ 0x58 1 2 tplate

/2/ 0x68 1 3 hjensen

/1/1/ 0x5AC0 2 4 schai

/1/2/ 0x5B40 2 5 elang

/1/3/ 0x5BC0 2 6 gsmits

/2/1/ 0x6AC0 2 7 sdavis

/2/2/ 0x6B40 2 8 norint

/1/1/1/ 0x5AD6 3 9 jwang

/1/1/2/ 0x5ADA 3 10 malexander

Note

For diagrams that show the difference between a depth-first index and a breadth-first index, see [Hierarchical Data (SQL Server)](https://docs.microsoft.com/en-us/sql/relational-databases/hierarchical-data-sql-server).

#### To drop the unnecessary columns

1. The **ManagerID** column represents the employee/manager relationship, which is now represented by the **OrgNode** column. If other applications do not need the **ManagerID** column, consider dropping it by using the following statement:

Copy

ALTER TABLE NewOrg DROP COLUMN ManagerID ;

GO

1. The **EmployeeID** column is also redundant. The **OrgNode** column uniquely identifies each employee. If other applications do not need the **EmployeeID** column, consider dropping the index and then the column by using the following code:

Copy

DROP INDEX EmpIDs\_unq ON NewOrg ;

ALTER TABLE NewOrg DROP COLUMN EmployeeID ;

GO

#### To replace the original table with the new table

1. If your original table contained any additional indexes or constraints, add them to the **NewOrg** table.
2. Replace the old **EmployeeDemo** table with the new table. Run the following code to drop the old table, and then rename the new table with the old name:

Copy

DROP TABLE EmployeeDemo ;

GO

sp\_rename 'NewOrg', EmployeeDemo ;

GO

1. Run the following code to examine the final table:

Copy

SELECT \* FROM EmployeeDemo ;

## Next Task in Lesson

[Summary: Converting a Table to a Hierarchical Structure](https://docs.microsoft.com/en-us/sql/relational-databases/tables/lesson-1-4-summary-converting-a-table-to-a-hierarchical-structure)