



­

Assignment Document:

HierarchyID Sql Server

Version: < Sql Server 2008>/ASSIGNMENT/xxxx/x.x

Date: 03-08-2011

Cognizant

500 Glen Pointe Center West

Teaneck, NJ 07666

Ph: 201-801-0233  
[www.cognizant.com](http://www.cognizant.com)

Contents

[Topic: <Hierarchyid> 3](#_Toc300170881)

[Hands-On Exercises 3](#_Toc300170882)

[Hands-On Exercise 1: <To display entire table with NodePath> 3](#_Toc300170883)

[Hands-On Exercise 2: <To return the path of the root element using GetRoot ()> 4](#_Toc300170887)

[Hands-On Exercise 3: <To return the level of all items using GetLevel ()> 4](#_Toc300170889)

[Hands-On Exercise 4: <To return the parent of an item n-level above it> 5](#_Toc300170890)

[Hands-On Exercise 5: <To return the path of a child underneath parent) 5](#_Toc300170891)

[Hands-On Exercise 6: <To determine whether an item is a child of the item specified by the parameter parent > 6](#_Toc300170893)

[Hands-On Exercise 7: < to find all the employees who are at level 3 using GetLevel() > 6](#_Toc300170894)

[Hands-On Exercise 8: < to traverse up the hierarchy and find employees above a particular employee> 7](#_Toc300170896)

[Hands-On Exercise 9: < to traverse down the hierarchy and find employees above a particular employee> 7](#_Toc300170898)

Topic: <HierarchyID>

Total Completion Time: 28 minutes

### Hands-On Exercises

Hands-On Exercise 1: <To display entire table with NodePath>

Estimated Completion Time: 1 Minute

Step 1: Use AdventureWorks.

Step 2: Create a table dbo.MyEmployees and populate it with values

Step 3: Create a unique clustered index on column NodeID.

Step4: Copy paste the below code in SSMS Query window to display the entire table along with NodePath.

CREATE TABLE dbo.MyEmployees

(

EmployeeID smallint NOT NULL,

NodeID HierarchyID NOT NULL,

FirstName nvarchar (30) NOT NULL,

Lastname nvarchar (40) NOT NULL,

Title nvarchar (50) NOT NULL,

);

-- Populate the table with values.

INSERT INTO dbo.MyEmployees values ( 1,hierarchyid::GetRoot(), N'Ken', N'Sánchez', N'Chief Executive Officer')

INSERT INTO dbo.MyEmployees values ( 2,'/1/', N'Brian', N'Welcker', N'Vice President of Sales')

INSERT INTO dbo.MyEmployees values( 3,'/1/2/', N'Stephen', N'Jiang', N'North American Sales Manager')

INSERT INTO dbo.MyEmployees values ( 4,'/1/2/2/', N'Michael', N'Blythe', N'Sales Representative' )

INSERT INTO dbo.MyEmployees values ( 5,'/1/3/1/', N'Linda', N'Mitchell', N'Sales Representative')

INSERT INTO dbo.MyEmployees values ( 6,'/1/3/', N'Syed', N'Abbas', N'Pacific Sales Manager')

INSERT INTO dbo.MyEmployees values( 7,'/1/2/1/', N'Lynn', N'Tsoflias', N'Sales Representative')

INSERT INTO dbo.MyEmployees values ( 8,'/1/1/', N'David',N'Bradley', N'Marketing Manager')

INSERT INTO dbo.MyEmployees values ( 9,'/1/1/1/', N'Mary', N'Gibson', N'Marketing Specialist')

GO

CREATE UNIQUE CLUSTERED INDEX idx\_node on MyEmployees (NodeID);

/\*displays the entire table with nodepath \*/

SELECT nodeid.ToString() AS nodepath,

employeeid,firstname+' '+lastname AS employeename,

title

FROM MyEmployees;

Hands-On Exercise 2: <To return the path of the root element using GetRoot ()>

Estimated Completion Time: 1 Minute

Step 1: Use AdventureWorks database

Step 2: Use dbo.MyEmployees table

Step 3: Copy paste the below code in SSMS query window to display the path of the root element.

/\*returns the path of the root element \*/

SELECT HierarchyID :: GetRoot (),

firstname + ' ' + lastname AS EmployeeName FROM MyEmployees;

Hands-On Exercise 3: <To return the level of all items using GetLevel ()>

Estimated Completion Time: 1 Minute

Step 1: Use AdventureWorks database

Step 2: Use dbo.MyEmployees table

Step 3: Copy paste the below code in SSMS query window to display the level of the items.

USE AdventureWorks

GO

/\*returns the level of all the elements \*/

SELECT NodeID.GetLevel() as HierarchyLevel, NodeID.ToString() as nodepath,

firstname + ' ' +lastname as employeename from MyEmployees;

Hands-On Exercise 4: <To return the parent of an item n-level above it>

Estimated Completion Time: 3 Minutes

Step 1: Use AdventureWorks database

Step 2: Use dbo.MyEmployees table

Step 3: Copy paste the below code in SSMS query window to return the parent of an item n-level above it

USE AdventureWorks

GO

SELECT

nodeid,

nodeid.ToString() AS nodepath,

nodeid.GetAncestor (1).ToString() AS Parent,

firstname + ' ' +lastname AS employee\_name FROM MyEmployees;

SELECT

nodeid,

nodeid.ToString() AS nodepath,

nodeid.GetAncestor (2).ToString() AS Parent,

firstname + ' ' +lastname AS employee\_name FROM MyEmployees;

Hands-On Exercise 5: <To return the path of a child underneath parent)

Estimated Completion Time: 3 Minutes

Step 1: Use AdventureWorks database

Step 2: Use dbo.MyEmployees table

Step 3: Copy paste the below code in SSMS query window to return the path of a child underneath parent.

SELECT

nodeid,

nodeid.ToString()AS nodepath,

nodeid.GetDescendant (null,null).ToString() AS FirstChild,

firstname + ' ' +lastname AS employee\_name FROM MyEmployees;

Hands-On Exercise 6: <To determine whether an item is a child of the item specified by the parameter parent >

Estimated Completion Time: 3 Minutes

Step 1: Use AdventureWorks database

Step 2: Use dbo.MyEmployees table

Step 3: Copy paste the below code in SSMS query window to determine whether an item is a child of the item specified by the parameter parent.

select nodeid.ToString() AS nodepath,

nodeid.IsDescendantOf (HierarchyID::Parse('/1/')) as IsDescendantOf,

firstname + ' ' +lastname as EmployeeName

from MyEmployees;

Hands-On Exercise 7: < to find all the employees who are at level 3 using GetLevel() >

Estimated Completion Time: 5 Minutes

Step 1: Use AdventureWorks database

Step 2: Use dbo.MyEmployees table

Step 3: Copy paste the below code in SSMS query window to find all the employees who are at level 3 using GetLevel()

USE AdventureWorks

select NodeID.GetLevel() as HieararchyLevel, NodeId.ToString() AS NodePath,

firstname + ' ' +lastname as employeename,title from MyEmployees

where NodeID.GetLevel() = 3;

Hands-On Exercise 8: < to traverse up the hierarchy and find employees above a particular employee>

Estimated Completion Time: 5 Minutes

Step 1: Use AdventureWorks database

Step 2: Use dbo.MyEmployees table

Step 3: Copy paste the below code in SSMS query window to < to traverse up the hierarchy and find employees above a particular employee

USE AdventureWorks

declare @senior as HierarchyId

Select @senior = NodeID from MyEmployees where employeeid = 6

select NodeID.GetLevel() as HieararchyLevel, NodeId.ToString() AS NodePath,employeeid,

firstname + ' ' +lastname as employeename,title from MyEmployees

where @senior.IsDescendantOf(NodeID) = 1 and NodeID.ToString() <> HierarchyID::GetRoot();

Hands-On Exercise 9: < to traverse down the hierarchy and find employees above a particular employee>

Estimated Completion Time: 5 Minutes

Step 1: Use AdventureWorks database

Step 2: Use dbo.MyEmployees table

Step 3: Copy paste the below code in SSMS query window to traverse down the hierarchy and find employees above a particular employee

declare @senior1 as HierarchyId

Select @senior1 = NodeID from MyEmployees where employeeid = 6

select NodeID.GetLevel() as HieararchyLevel, NodeId.ToString() AS NodePath,employeeid,

firstname + ' ' +lastname as employeename,title from MyEmployees

where NodeID.IsDescendantOf(@senior1) = 1;

Summary

You have just learnt:

* GetAncestor() method finds the nth ancestor of the given child node
* GetDescendant () method gets the descendant of a given node.
* GetLevel () finds the Level of the current node.
* GetRoot () returns the root of the hierarchy tree and this is a static method if you are using it within CLR.
* IsDescendant () returns true/false (BIT) if the node is a descendant of the parent.