



www.databasejournal.com/features/mssql/article.php/3829861

[Back to Article](#)

Obtaining data changes with Change Tracking in SQL Server 2008
July 20, 2009

In the [first article](#) of this series, we enabled change tracking on the HumanResources.Department table. Now how do we get the changed data? With the CHANGETABLE function, we can get change tracking information. This function provides two modes: CHANGES and VERSION. In the CHANGES mode, the CHANGETABLE(CHANGES table , last_sync_version) takes the name of the table being tracked and a version number, and returns all the changes to the table after the version. In the VERSION mode, the CHANGETABLE(VERSION table, primary_key_column_name [, ...n] , (primary_key_column_value [, ...n])) takes a table name, primary key column(s), and primary key column value(s). The key values identify a row in the table. The VERSION mode returns the current version and change context that is associated with the specific row.

Data changes need to be compared to a baseline. When you first synchronize your client (a .NET application and/or another SQL Server table) with the HumanResources.Department table, you obtain an initial data set from all rows of the table. You also get the baseline version number, i.e., the maximum version number of all the rows, at that time and record it for the next synchronization request. At the next synchronization request, the system determines which rows have been modified since the baseline version, and a new baseline version is saved for the next synchronization. The algorithm is shown below.

```
-- Obtain the current synchronization version. This will be used
--the next time CHANGETABLE(CHANGES...) is called.
SET @synchronization_version = CHANGE_TRACKING_CURRENT_VERSION();

-- If this is the first synchronization session
IF (@sync_initialized = 0)
BEGIN
    -- Initialize from the table
    SELECT DepartmentID, Name, GroupName, ModifiedDate
    FROM HumanResources.Department
END
ELSE
BEGIN
    -- Obtain incremental changes by using the synchronization version
    --obtained the last time the data was synchronized.
    SELECT CT.SYS_CHANGE_OPERATION,
           DepartmentID, Name, GroupName, ModifiedDate
    FROM HumanResources.Department
    RIGHT OUTER JOIN
        CHANGETABLE(CHANGES HumanResources.Department,
                    @last_synchronization_version) AS CT
    ON
        P.DepartmentID = CT.DepartmentID
END
```

In our example, let's first query all the rows from the HumanResources.Department table to get a baseline and get a baseline version.

```
DECLARE @synchronization_version bigint
-- Obtain the current synchronization version. This will be used
--the next time CHANGETABLE(CHANGES...) is called.
SET @synchronization_version = CHANGE_TRACKING_CURRENT_VERSION()
```

Instant Cloud Servers Running:

PHP on Windows

with FastCGI

*Outstanding PHP Performance
on a Windows 2008 Server!*

GOGRID NOW ►

\$50 Free Credit!

Information-Driven Business Center

- Oracle Data Guard 11g - The Next Era in Data Protection and Availability**
 Oracle Data Guard 11g can address both High Availability and Disaster Recovery requirements, and is the ideal complement to Oracle Real Application Clusters (Oracle RAC). Straightforward to implement and manage, Data Guard has the requisite knowledge of the Oracle database to reliably protect a standby database from corruptions that attempt to propagate from a primary database. »
- Oracle Database 11g: Transparent Solutions for Security and Compliance**
 Historically organizations have relied on network and application security, leaving their databases and the sensitive information inside exposed. With Oracle Database 11g, organizations can now deploy reliable data security solutions that do not require any changes to existing applications. »
- Oracle Advanced Compression White Paper**
 Oracle Database 11g introduces the Advanced Compression Option to help customers reduce the resources and costs of managing large data volumes. The introduction of these exciting new technologies come at an opportune time as terabyte-sized databases are becoming prevalent in enterprise data centers. »

Visit the Oracle Information-Driven Business Center **ORACLE®**

Hot List

internet.com

**Develop on the Best-engineered,
Most Interoperable Platform for
Mission-critical Computing**

Learn More >>

20/07/2009

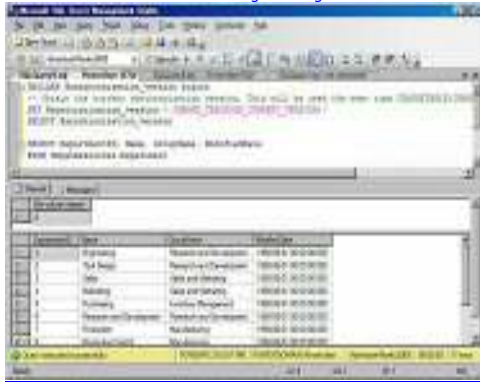
MS SQL Server Articles & Tutorials ...

```
SELECT @synchronization_version
```

```
-- Initialize from the base table
```

```
SELECT DepartmentID, Name, GroupName, ModifiedDate  
FROM HumanResources.Department
```

[Click for larger image](#)



As shown above, the baseline version number is 0 since we just enabled change tracking in the database.

Let's insert a new row into the table and update an existing row.

```
INSERT INTO HumanResources.Department  
(Name, GroupName, ModifiedDate)  
VALUES ('Product Design', 'Research and Development', GETDATE())
```

```
UPDATE HumanResources.Department  
SET GroupName='Accounting'  
WHERE DepartmentID=10
```

To obtain the changes since the last version 0, run the following script.

```
DECLARE @synchronization_version bigint  
-- Obtain the current synchronization version. This will be used the next time CHANGETABLE(CHANGES...) is called.  
SET @synchronization_version = CHANGE_TRACKING_CURRENT_VERSION()  
SELECT @synchronization_version  
  
-- Obtain incremental changes by using the synchronization version obtained the last time the data was synchronized.  
SELECT CT.SYS_CHANGE_VERSION, CT.SYS_CHANGE_OPERATION, CT.DepartmentID, Name, GroupName, ModifiedDate  
FROM HumanResources.Department D  
RIGHT OUTER JOIN  
    CHANGETABLE(CHANGES HumanResources.Department, 0) AS CT  
ON  
    D.DepartmentID = CT.DepartmentID
```

[Click Here](#)



MARKETPLACE

[Help Desk Software by IssueTrak](#)

Easy to use, customizable, powerful and flexible. Free online demo. Try it now!
www.issuetrak.com

[Adobe Solutions for Higher Education](#)

Teach Students the Skills They Need for Career Success. Download Free Curriculum
www.tryit.adobe.com

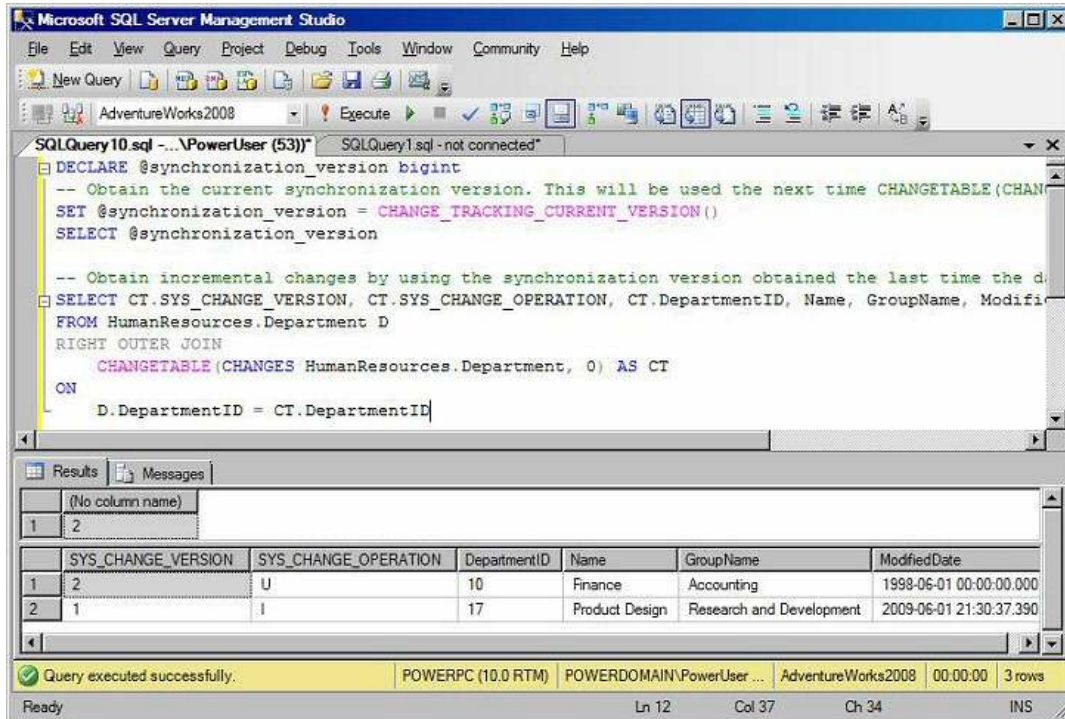
[Take Advantage of Super HP Rebates!](#)

HP Thin Client Discounts From SYNEX. Find Out More About This Offer Now!
www.SYNEX.com

[DOWNLOAD CRYSTAL REPORTS SERVER 2008 - FREE](#)

Minimize IT cost with auto job scheduling and report delivery. Great for businesses of all sizes.
[Crystal Reports Server 2008 - Power](#)

[Advertise Here](#)



The screenshot shows the Microsoft SQL Server Management Studio interface. The SQL query window displays the following code:

```

DECLARE @synchronization_version bigint
-- Obtain the current synchronization version. This will be used the next time CHANGETABLE (CHAN
SET @synchronization_version = CHANGE_TRACKING_CURRENT_VERSION()
SELECT @synchronization_version

-- Obtain incremental changes by using the synchronization version obtained the last time the d
SELECT CT.SYS_CHANGE_VERSION, CT.SYS_CHANGE_OPERATION, CT.DepartmentID, Name, GroupName, Modifi
FROM HumanResources.Department D
RIGHT OUTER JOIN
    CHANGETABLE (CHANGES HumanResources.Department, 0) AS CT
ON
    D.DepartmentID = CT.DepartmentID
  
```

The Results pane shows the following data:

| | (No column name) |
|---|------------------|
| 1 | 2 |

| | SYS_CHANGE_VERSION | SYS_CHANGE_OPERATION | DepartmentID | Name | GroupName | ModifiedDate |
|---|--------------------|----------------------|--------------|----------------|--------------------------|-------------------------|
| 1 | 2 | U | 10 | Finance | Accounting | 1998-06-01 00:00:00.000 |
| 2 | 1 | I | 17 | Product Design | Research and Development | 2009-06-01 21:30:37.390 |

The status bar at the bottom indicates: Query executed successfully. POWERPC (10.0 RTM) POWERDOMAIN\PowerUser ... AdventureWorks2008 00:00:00 3 rows. Ready Ln 12 Col 37 Ch 34 INS.

As you can see in the figure, the current version has increased to 2. The row with DepartmentID=10 was updated as indicated by the value "U" in the SYS_CHANGE_OPERATION column. The row with DepartmentID=17 was inserted as indicated by the value "I" in the SYS_CHANGE_OPERATION column.

If you have column tracking enabled on the HumanResources.Department table, you can get the data from only the columns that were changed.

```

DECLARE @synchronization_version bigint
-- Obtain the current synchronization version. This will be used the next time CHANGETABLE (CHANGES...) is called.
SET @synchronization_version = CHANGE_TRACKING_CURRENT_VERSION()
SELECT @synchronization_version

-- Obtain incremental changes by using the synchronization version obtained the last time the data was synchronized.
SELECT CT.SYS_CHANGE_VERSION, CT.SYS_CHANGE_OPERATION, CT.DepartmentID,
CASE CHANGE_TRACKING_IS_COLUMN_IN_MASK
    WHEN 1 THEN Name ELSE NULL END as Name,
CASE CHANGE_TRACKING_IS_COLUMN_IN_MASK(COLUMNPROPERTY
    (OBJECT_ID('HumanResources.Department'), 'GroupName', 'ColumnId'), SYS_CHANGE_COLUMNS)
    WHEN 1 THEN GroupName ELSE NULL END AS GroupName,
    ModifiedDate
FROM HumanResources.Department D
RIGHT OUTER JOIN
    CHANGETABLE (CHANGES HumanResources.Department, 0) AS CT
ON
    D.DepartmentID = CT.DepartmentID
  
```

The screenshot shows the Microsoft SQL Server Management Studio interface. The main window displays a T-SQL query in a file named 'SQLQuery4.sql'. The query is designed to track changes in the 'HumanResources.Department' table using the 'CHANGETABLE' function. It declares a synchronization version, obtains the current version, and then selects incremental changes. The results pane shows two rows of data:

| | SYS_CHANGE_VERSION | SYS_CHANGE_OPERATION | DepartmentID | Name | GroupName | ModifiedDate |
|---|--------------------|----------------------|--------------|----------------|--------------------------|-------------------------|
| 1 | 2 | U | 10 | NULL | Accounting | 1998-06-01 00:00:00.000 |
| 2 | 1 | I | 17 | Product Design | Research and Development | 2009-06-24 13:21:51.530 |

The status bar at the bottom indicates 'Query executed successfully.' and shows the execution time as 00:00:00 for 3 rows.

As you can see in the figure, the value of the Name column was not returned for the first row as it was not updated in the update statement. In this example, the Name column is only a nvarchar column and doesn't take much storage. I only use the column here for demonstration purposes. LOB columns would be better candidates for column tracking as the overhead for enabling column tracking can be small compared to the benefits of retrieving LOB data through the network.

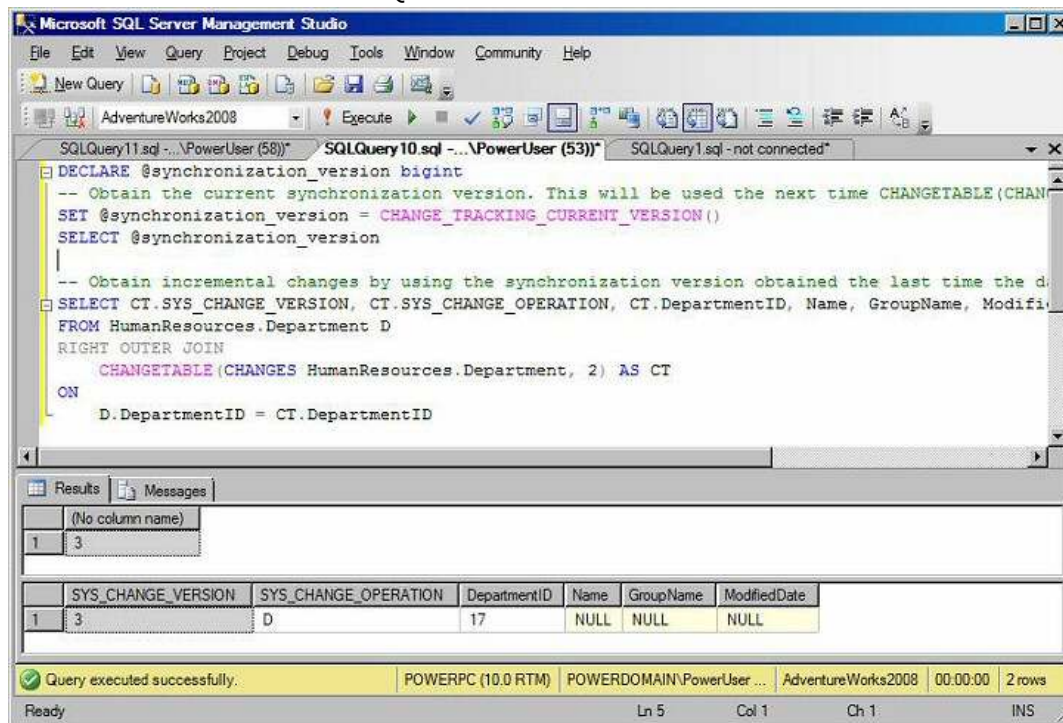
Let's delete the newly added row.

```
DELETE FROM HumanResources.Department
WHERE DepartmentID=17
```

To obtain the changes since the last version 2, run the previous script with a new version 2.

```
DECLARE @synchronization_version bigint
-- Obtain the current synchronization version. This will be used the next time CHANGETABLE (CHANGES...) is called.
SET @synchronization_version = CHANGE_TRACKING_CURRENT_VERSION()
SELECT @synchronization_version

-- Obtain incremental changes by using the synchronization version obtained the last time the data was synchronized.
SELECT CT.SYS_CHANGE_VERSION, CT.SYS_CHANGE_OPERATION, CT.DepartmentID, Name, GroupName, ModifiedDate
FROM HumanResources.Department D
RIGHT OUTER JOIN
    CHANGETABLE (CHANGES HumanResources.Department, 2) AS CT
ON
    D.DepartmentID = CT.DepartmentID
```

As shown in the figure, the value "D" in the SYS_CHANGE_OPERATION column indicates the row with DepartmentID=17, was deleted.

Because the change tracking information can be purged before the next synchronization runs if the AUTO_CLEANUP option is turned on, we also need to compare @last_synchronization_version with the minimum version saved in the change tracking table for the specified table. If @last_synchronization_version is less than `CHANGE_TRACKING_MIN_VALID_VERSION(OBJECT_ID('HumanResources.Department'))`, we need to reinitialize and get a new baseline containing all the rows from the HumanResources.Department base table. Therefore, the condition

```
IF (@sync_initialized = 0)
```

needs to be changed to

```
IF (@last_synchronization_version < CHANGE_TRACKING_MIN_VALID_VERSION(OBJECT_ID('HumanResources.Department')))
```

On a busy system, between getting the current version and obtaining the changes from the `CHANGETABLE()` function, the `HumanResources.Department` table can be changed by other sessions, or the change tracking information can be removed by the cleanup process and causes `CHANGE_TRACKING_MIN_VALID_VERSION` to increase. Therefore, it is better to use snapshot isolation to ensure that all change tracking information is consistent during the transaction. The complete algorithm is shown below.

```
SET TRANSACTION ISOLATION LEVEL SNAPSHOT;
BEGIN TRAN
    DECLARE @synchronization_version bigint
    -- Obtain the current synchronization version. This will be used the next time CHANGETABLE(CHANGE...) is called.
    SET @synchronization_version = CHANGE_TRACKING_CURRENT_VERSION()
    SELECT @synchronization_version

    IF (@last_synchronization_version < CHANGE_TRACKING_MIN_VALID_VERSION(OBJECT_ID('HumanResources.Department')))
    BEGIN
        -- Initialize from the base table
        SELECT DepartmentID, Name, GroupName, ModifiedDate
        FROM HumanResources.Department

    END
    ELSE
    BEGIN
        -- Obtain incremental changes by using the synchronization version obtained the last time the data was synchronized.
        SELECT CT.SYS_CHANGE_OPERATION, CT.DepartmentID, Name, GroupName, ModifiedDate
        FROM HumanResources.Department D
        RIGHT OUTER JOIN
            CHANGETABLE(CHANGE HumanResources.Department, @last_synchronization_version) AS CT
        ON
            D.DepartmentID = CT.DepartmentID

    END
COMMIT TRAN
```

Enabling snapshot isolation on a database adds non-trivial performance overhead to the database. If you prefer not to use snapshot isolation, an alternative algorithm to obtain changes is shown below.

```
DECLARE @synchronization_version bigint

SET @synchronization_version = CHANGE_TRACKING_CURRENT_VERSION()
SELECT @synchronization_version

IF (@last_synchronization_version < CHANGE_TRACKING_MIN_VALID_VERSION(OBJECT_ID('HumanResources.Department')))
    -- Initialize from the base table
    SELECT DepartmentID, Name, GroupName, ModifiedDate
    FROM HumanResources.Department
ELSE
BEGIN
    -- Obtain incremental changes by using the synchronization version obtained the last time the data was synchronized.
    SELECT CT.SYS_CHANGE_OPERATION, CT.DepartmentID, Name, GroupName, ModifiedDate
    FROM HumanResources.Department D
    RIGHT OUTER JOIN
        CHANGETABLE (CHANGES HumanResources.Department, @last_synchronization_version) AS CT
    ON
        D.DepartmentID = CT.DepartmentID
    WHERE (CT.SYS_CHANGE_CREATION_VERSION <= @synchronization_version)
END
```

Instead of using snapshot isolation, this algorithm uses the new version to make sure no changes since the new version are returned from the CHANGETABLE(CHANGES ...) function. However, this algorithm doesn't solve the problem when CHANGE_TRACKING_MIN_VALID_VERSION changes and @last_synchronization_version becomes invalid.

Summary

This article illustrated how to obtain data changes using the CHANGETABLE function. Two algorithms were presented. You can implement these two algorithms in your .NET application using Sync Services.

» [See All Articles by Columnist Yan Pan](#)



WebMediaBrands



20/07/2009

MS SQL Server Articles & Tutorials ...

Search:

[WebMediaBrands Corporate Info](#)

Copyright 2009 WebMediaBrands Inc. All Rights Reserved.

[Legal Notices](#), [Licensing](#), [Reprints](#), [Permissions](#), [Privacy Policy](#).

[Advertise](#) | [New sletters](#) | [Shopping](#) | [E-mail Offers](#) | [Freelance Jobs](#) **new!**

Solutions

Whitepapers and eBooks

Whitepaper: Introducing the Azure Services Platform
Whitepaper: Introduction to Microsoft .NET Services for Developers
Whitepaper: Securing Microsoft's Cloud Infrastructure
Internet.com eBook: Becoming a Better Project Manager
Microsoft Partner Portal: What Azure Means for Microsoft Partners

Internet.com eBook: Web Development Frameworks for Java
Internet.com eBook: Developing a Content Management System Strategy
Article: Ten Things IT Professionals Should Know About Windows 7
Article: New BlackBerry Enterprise Server Eases Administration for IT Pros
MORE WHITEPAPERS, EBOOKS, AND ARTICLES

Webcasts

Get Started with .NET Services

MORE WEBCASTS, PODCASTS, AND VIDEOS

Downloads and eKits

Amyuni: PDF & PDF/A Engine for .NET and ActiveX Apps
Red Gate Free Trial: SQL Toolbelt
Iron Speed Designer Application Generator

Download Award-Winning Red Gate SQL Backup Pro
MORE DOWNLOADS, EKITS, AND FREE TRIALS

Tutorials and Demos

Internet.com Hot List: Get the Inside Scoop on the Hottest IT and Developer
Products
Internet.com Hot List: Java EE 6 Platform Highlights the JavaOne
Conference
MORE TUTORIALS, DEMOS AND STEP-BY-STEP GUIDES