This e-book describes the change tracking framework for Microsoft SQL Server.

You may add change tracking features to a database and track changes using SSMS or Microsoft Excel.

Also, you may extend the Microsoft Excel context menu to allow business users to track and rollback changes.
# Contents

<table>
<thead>
<tr>
<th>Contents</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>2</td>
</tr>
<tr>
<td>Chapter 1. Installation</td>
<td>3</td>
</tr>
<tr>
<td>Download</td>
<td>3</td>
</tr>
<tr>
<td>Download Package</td>
<td>3</td>
</tr>
<tr>
<td>Installation</td>
<td>3</td>
</tr>
<tr>
<td>Uninstalling</td>
<td>3</td>
</tr>
<tr>
<td>Chapter 2. Setup</td>
<td>4</td>
</tr>
<tr>
<td>Creating Administrator’s Workbook</td>
<td>4</td>
</tr>
<tr>
<td>Creating Change Tracking Triggers</td>
<td>7</td>
</tr>
<tr>
<td>Useful Operations</td>
<td>9</td>
</tr>
<tr>
<td>Attaching Context Menus</td>
<td>9</td>
</tr>
<tr>
<td>Chapter 3. Usage</td>
<td>10</td>
</tr>
<tr>
<td>Context Menu</td>
<td>10</td>
</tr>
<tr>
<td>Task Panes</td>
<td>10</td>
</tr>
<tr>
<td>Restoring Records</td>
<td>12</td>
</tr>
<tr>
<td>ID Lookup</td>
<td>13</td>
</tr>
<tr>
<td>Translation</td>
<td>14</td>
</tr>
<tr>
<td>Chapter 4. Permissions</td>
<td>15</td>
</tr>
<tr>
<td>Administrator Permissions</td>
<td>15</td>
</tr>
<tr>
<td>User Permissions</td>
<td>15</td>
</tr>
<tr>
<td>Chapter 5. Database Objects</td>
<td>16</td>
</tr>
<tr>
<td>Roles</td>
<td>16</td>
</tr>
<tr>
<td>Schemas</td>
<td>16</td>
</tr>
<tr>
<td>Tables</td>
<td>16</td>
</tr>
<tr>
<td>Views</td>
<td>17</td>
</tr>
<tr>
<td>Procedures</td>
<td>18</td>
</tr>
<tr>
<td>Triggers</td>
<td>20</td>
</tr>
<tr>
<td>Conclusion</td>
<td>22</td>
</tr>
<tr>
<td>About the Author</td>
<td>23</td>
</tr>
</tbody>
</table>
Introduction

The change tracking framework provides features that track DML changes (insert, update, and delete operations) in a database.

You may:

- Create and drop change tracking triggers.
- Create and drop procedures to select changes.
- Restore changed records.

You may execute the operations in SSMS, your application, or in Microsoft Excel using the SaveToDB add-in.

Moreover, you may easily extend Excel context menu to track changes in Excel.

The framework includes the following blocks:

- Database tables, views, and procedures to store and manage changes;
- Configuration objects to manage change tracking in Microsoft Excel for administrators;
- Configuration objects to track and restore changes in Microsoft Excel for end users.

You may download and install the framework in a couple of minutes.

The framework uses features of the SaveToDB Express add-in for Microsoft Excel. So, you may use it for free.

Microsoft SQL Server includes built-in features to track changes. You may read the following article, for example:

Track Data Changes (SQL Server)


We have created the change tracking framework for SQL Server to get the following features:

1. To see historical data in Microsoft SQL Express (not supported by the Change Data Capture feature).
2. To easily integrate and use the solution in Microsoft Excel (not supported by SQL Server).
3. To allow business users to manage change tracking features and rollback changes using Microsoft Excel.

We get a reliable and high-performance solution that you may use for free.

Try the framework. You may save a lot of time and make your users happier.

Best regards,

Sergey Vaselenko

November 8, 2017
Chapter 1. Installation

Download

You may download the change tracking framework at https://www.savetodb.com/download.htm. The application also requires SaveToDB 7.14 or higher. You may download the add-in here too.

Download Package

Unzip the downloaded package to a local drive.

The download package contains the ChangeTrackingFramework folder with the following files:

- change-tracking-framework-install.sql
- change-tracking-framework-remove.sql
- change-tracking-framework-for-sql-server.pdf

Installation

To install the change tracking framework using SQL Server Management Studio (SSMS), open and execute (F5) the change-tracking-framework-install.sql script. The script creates the following framework objects:

Uninstalling

To uninstall the framework, open and execute the change-tracking-framework-remove.sql script.

Note that the script removes frameworks objects, created change tracking triggers and procedures.
Chapter 2. Setup

Creating Administrator’s Workbook

You may use Microsoft Excel to manage change tracking framework objects.

Create a new workbook and run Wizards, Data Connection Wizard:

Select the first data provider to connect to Microsoft SQL Server and click Next:
Select the `logs.view_query_list` in the **Query List**, uncheck the checkbox as shown, and select `logs.view_objects`:

In the next screen, select the **schema** and **has_log_triggers** field to place to the ribbon to filter data and click **OK**:
Select cell **B3** to insert a table and click **OK**.

The SaveToDB add-in inserts the table. You see the database objects including columns to display triggers:

Format the table as you like and save the workbook:
Creating Change Tracking Triggers

The framework creates three triggers to log INSERT, UPDATE, and DELETE operations for captured tables.

Use the context menu to create and drop triggers:

You have to set 1 into the `execute_script` field to confirm creating triggers:

Check the target table and click OK.
The SaveToDB add-in updates the table, and you see actual triggers for the captured tables:

In this example, we have created triggers for the dbo25.members table. Its triggers look like:

The change tracking framework also creates a stored procedure to select changes.

The procedures have the `xl_log_` prefix. You may use filters in SSMS to filter change-tracking objects:

Note that the framework creates procedures in the schemas of underlying tables.

So, if a user has SELECT and EXECUTE permissions for a schema, he can execute a new procedure.

The framework configures such procedures as SaveToDB context menu items for underlying tables.
Useful Operations

You may use the Actions menu to drop all triggers at once and the ribbon parameters to filter objects:

![Excel screenshot showing the Actions menu with Drop All Triggers selected.]

Attaching Context Menus

The framework configures procedures to show changes as SaveToDB context menu items for underlying tables. To attach context menu items to other objects, edit the link data in the `logs.base_tables` table.

You may do this in Excel. Just connect to the table, edit the data and click the Save button.

![Excel screenshot showing the logs.base_tables table with data.]  

In this example, we have attached `dbo25.members` context menu items to the `xls25.usp_members` procedure.

As a result, users may track changes using the context menu of the `xls25.usp_members` procedure.

They need click Reload, Reload Data and Configuration to load new context menu items.

Note that the target object must select all primary key fields of the base table with the same names.
Chapter 3. Usage

Context Menu

The framework creates procedures to select changes of specific rows or entire tables by change types:

Click Reload, Reload Data and Configuration to refresh the context menu after creating the procedures.

Task Panes

The SaveToDB add-in shows changes in the Excel task panes. Initially, the log can be empty:
You may configure columns shown in the task pane. For example, you may leave the most useful columns:

Let’s change the name of the Accounts member and save the changes.

Click the **Reload** menu item in the Change Log task pane (or run it from the context menu once again):

You see the changes.
The SaveToDB add-in also opens a new task pane to show record details (non-empty values):

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>1</td>
</tr>
<tr>
<td>dimension_id</td>
<td>1</td>
</tr>
<tr>
<td>sort_order</td>
<td>1</td>
</tr>
<tr>
<td>code</td>
<td>ACCOUNTS</td>
</tr>
<tr>
<td>name</td>
<td>Accounts2 [Accounts]</td>
</tr>
<tr>
<td>calc_type_id</td>
<td>0</td>
</tr>
<tr>
<td>row_color</td>
<td>11</td>
</tr>
<tr>
<td>is_active</td>
<td>1</td>
</tr>
</tbody>
</table>

Change Action: updated
Change Date: Nov 7 2017 1:32PM
Change User: dbo

You may see old values in square brackets.

**Restoring Records**

The context menu of the log records contains more items:

You may restore the current or previous version of the active row.

In this example, we want to restore the previous name. So, choose the previous record.
You must set 1 in the confirm field (to prevent unwanted changes) and click OK:

The add-in executes the logs.usp_restore_record procedure to restore changes and updates data:

**ID Lookup**

The task pane shows table rows as is.

Use the ID Lookup context menu item to find the related row in the foreign key table.

For example, this screen shows a dimension row of the account member:
Translation

The change tracking framework supports translation. Select the UI and data languages in the SaveToDB Options:

![SaveToDB Options]

To apply changes, click **Reload, Reload Data and Configuration** and restart Excel.

Voilà:
Chapter 4. Permissions

Administrator Permissions

Users of the **log_administrators** role have all user permissions and additionally can:

- Create and drop change tracking triggers;
- Clear logs.

To create and drop triggers, users must also have such permissions in the target schema.

To clear logs, users must also have UPDATE permissions on the captured tables.

You may find **log_administrators** permissions in the **logs.usp_set_role_permissions_administrators** procedure.

User Permissions

Users of the **log_users** role can:

- Select log records;
- Restore records from the log.

To select log records, users must also have SELECT permissions on the captured tables.

To restore records, the user must also have UPDATE permissions on the captured tables.

You may find **log_users** permissions in the **logs.usp_set_role_permissions_users** procedure.

You may use the **sp_addrolemember** procedure to assign a role to a user:

```sql
EXEC sp_addrolemember 'log_users', 'pa_user_01'
```
Chapter 5. Database Objects

Roles

The change tracking framework creates the following roles:

- \texttt{log_administrators}
- \texttt{log_users}

\textbf{log_administrators}

Assign this role to users who can create triggers, drop triggers and clear logs.

See actual database permissions in the \texttt{logs.usp_set_role_permissions_administrators} procedure.

\textbf{log_users}

Assign this role to business users who will use the change tracking functions.

See actual database permissions in the \texttt{logs.usp_set_role_permissions_users} procedure.

Schemas

\textbf{logs}

The change tracking framework creates its objects in the logs schema.

Tables

The change tracking framework contains the following tables in the logs schema:

- \texttt{base_tables}
- \texttt{change_logs}
- \texttt{column_translations}
- \texttt{event_handlers}
- \texttt{object_translations}

\textbf{base_tables}

This user table contains a configuration used to attach context menu items to database objects.

See \texttt{Attaching Context Menus}.

\textbf{change_logs}

This application table contains change tracking data for all captured tables. Do not edit it.

The table contains primary keys and both current and previous row values of source records in an XML format.

\textbf{column_translations}

This application table contains column translation data. You may change and add translations.
event_handlers
This application table contains event handler configuration.
The framework inserts and deletes handlers of every captured table here.
Do not edit it.

object_translations
This application table contains object translation data. You may change and add translations.

Views
The change tracking framework contains the following views in the logs schema:

- view_column_translations
- view_event_handlers
- view_object_translations
- view_objects
- view_query_list

view_column_translations
This view selects column translation configuration for the SaveToDB add-in.

view_event_handlers
This view selects event handler configuration for the SaveToDB add-in.

view_object_translations
This view selects object translation configuration for the SaveToDB add-in.

view_objects
This view selects database objects and related change tracking framework information.
Use it to configure change tracking triggers. See Setup.

view_query_list
This view selects change tracking framework objects to connect with Microsoft Excel.
See Creating Administrator’s Workbook.
**Procedures**

The change tracking framework contains the following views in the logs schema:

- `usp_clear_logs`
- `usp_create_triggers`
- `usp_drop_all_triggers`
- `usp_drop_triggers`
- `usp_restore_current_record`
- `usp_restore_previous_record`
- `usp_restore_record`
- `usp_select_lookup_id`
- `usp_select_record`
- `usp_select_records`
- `usp_set_role_permissions_administrators`
- `usp_set_role_permissions_users`

**usp_clear_logs**

This procedure clears change tracking records of the specified target table. A user must also have the UPDATE permission on the target table.

You may run it from the context menu of the logs.view_objects view.

See [Creating Change Tracking Triggers](#).

**usp_create_triggers**

This procedure creates change tracking triggers of the specified target table. A user must also have the ALTER permission on the target table.

You may run it from the context menu of the logs.view_objects view.

See [Creating Change Tracking Triggers](#) and [Triggers](#).

**usp_drop_all_triggers**

This procedure drops all change tracking triggers and clears all logs. A user must also have the ALTER permission on altered tables.

You may run it from the Actions menu of the logs.view_objects view.

See [Useful Operations](#).

**usp_drop_triggers**

This procedure drops change tracking triggers of the specified target table. A user must also have the ALTER permission on the target table.

You may run it from the context menu of the logs.view_objects view.

See [Creating Change Tracking Triggers](#).
**usp_restore_current_record**
This procedure restores the current version of the change log record.
See [Restoring Records](#).
The procedure executes the usp_restore_record procedure with the predefined parameter.

**usp_restore_previous_record**
This procedure restores the previous version of the change log record.
See [Restoring Records](#).
The procedure executes the usp_restore_record procedure with the predefined parameter.

**usp_restore_record**
This procedure restores a row from the change log record.
A user must have the UPDATE permission on the captured table.

**usp_select_lookup_id**
This procedure selects rows from tables referenced by foreign keys.
See [ID Lookup](#).
A user must have the SELECT permission on the captured and referenced tables.

**usp_select_record**
This procedure selects change details of a change log record.
See [Task Panes](#).
A user must have the SELECT permission on the captured table.

**usp_select_records**
This procedure selects change log records.
Generated procedures like dbo25.xl_log_members execute this procedure to select changes.
See [Creating Change Tracking Triggers](#) and [Task Panes](#).
A user must have the SELECT permission on the captured table.

**usp_set_role_permissions_administrators**
This procedure sets permissions for the log_administrators role. See [Administrator Permissions](#).

**usp_set_role_permissions_users**
This procedure sets permissions for the log_users role. See [User Permissions](#).
Triggers

The `logs.usp_create_triggers` procedure generates three triggers for every captured table like these:

- `trigger_members_log_insert`
- `trigger_members_log_update`
- `trigger_members_log_delete`

Below are the trigger examples.

**Insert Trigger**

```sql
CREATE TRIGGER [dbo25].[trigger_members_log_insert]
    ON [dbo25].[members]
    AFTER INSERT
AS
BEGIN
 SET NOCOUNT ON
 INSERT INTO logs.change_logs
 (object_id, id, inserted, deleted, change_type, change_date, change_user)
 SELECT
  369488445
  , inserted.[id]
  , (SELECT * FROM inserted FOR XML RAW)
  , NULL
  , 1
  , GETDATE()
  , USER_NAME()
 FROM
inserted
END
```

**Update Trigger**

```sql
CREATE TRIGGER [dbo25].[trigger_members_log_update]
    ON [dbo25].[members]
    AFTER UPDATE
AS
BEGIN
 SET NOCOUNT ON
 INSERT INTO logs.change_logs
 (object_id, id, inserted, deleted, change_type, change_date, change_user)
 SELECT
  369488445
  , inserted.[id]
  , (SELECT * FROM inserted FOR XML RAW)
  , (SELECT * FROM deleted FOR XML RAW)
  , 3
  , GETDATE()
  , USER_NAME()
 FROM
inserted
END
```
**Delete Trigger**

```
CREATE TRIGGER [dbo25].[trigger_members_log_delete]
ON [dbo25].[members]
AFTER DELETE
AS
BEGIN
SET NOCOUNT ON
INSERT INTO logs.change_logs
    (object_id, id, inserted, deleted, change_type, change_date, change_user)
SELECT
    369488445
, deleted.[id]
, NULL
, (SELECT * FROM deleted FOR XML RAW)
, 2
, GETDATE()
, USER_NAME()
FROM deleted
END
```

**Change Log Table**

Triggers save records to the logs.change_logs table that has the following declaration:

```
CREATE TABLE [logs].[change_logs] (
    [change_id] int IDENTITY(1,1) NOT NULL
, [object_id] int NOT NULL
, [id] int NULL
, [keys] nvarchar(445) NULL
, [inserted] xml NULL
, [deleted] xml NULL
, [change_type] tinyint NOT NULL
, [change_date] datetime NOT NULL
, [change_user] nvarchar(128) NOT NULL
, CONSTRAINT [PK_change_logs_logs] PRIMARY KEY ([change_id])
)
```

**Comments**

All triggers of all tables add new change tracking records to the single logs.change_logs table.

If a table has an identity field, the triggers use this field. This is the fastest case to track and select changes.

Otherwise, the triggers pack the key field values into XML. The maximum length is 445 characters.

You cannot use the framework for tables with larger key lengths.

Triggers pack inserted and deleted values into XML.

Also, triggers save the time and user of the changes.
Conclusion

To add change tracking features to an example database, we have made the following steps:

1. Install the change tracking framework.
2. Create administrator’s workbook.
3. Create triggers for captured tables.

You may repeat these steps for any database in a couple of minutes.

However, you get a reliable and high-performance solution. For free.

Moreover, you get the possibility to work with logs in Microsoft Excel using the free SaveToDB Express add-in.

And, you may extend your existing Microsoft Excel application to allow users track changes and restore records.

I hope you try it and will like it.

Best regards,

Sergey Vaselenko
About the Author

My name is Sergey Vaselenko.

I am from Russia, Moscow.

My passion is creating software.

I am a founder and CEO of Gartle Technology Corporation and a leading developer of the SaveToDB add-in.

You are welcome to contact me at

www.facebook.com/sergey.vaselenko
www.linkedin.com/in/vaselenko/