

CHANGE TRACKING FRAMEWORK

for Microsoft SQL Server

Sergey Vaselenko

CHANGE TRACKING FRAMEWORK

for Microsoft SQL Server

Written by Sergey Vaselenko

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

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

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)

<https://docs.microsoft.com/en-us/sql/relational-databases/track-changes/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

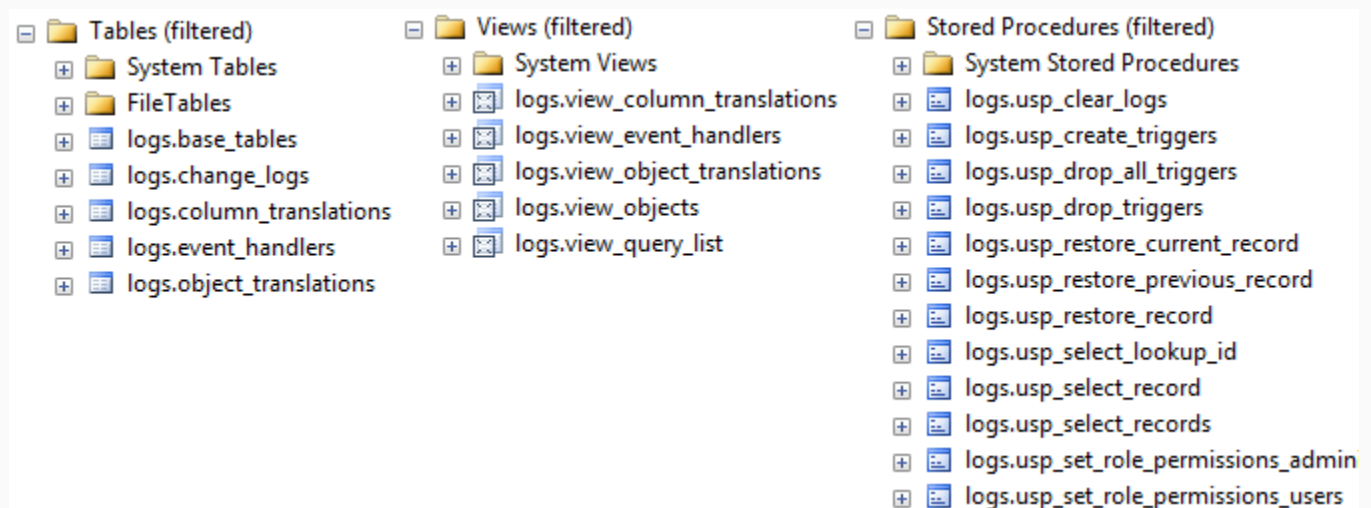
Unzip the downloaded package to a local drive.

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.

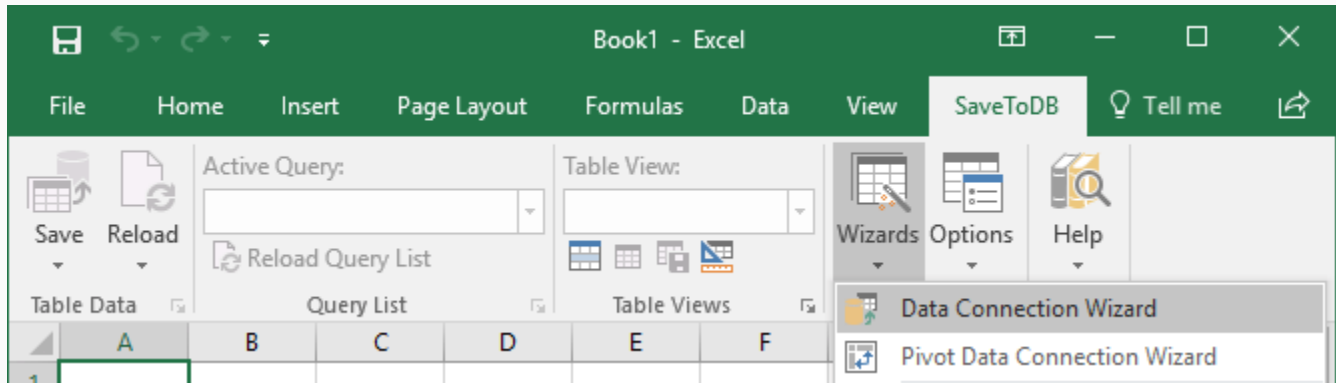
Note that the script removes frameworks objects, created change tracking triggers and procedures.

Chapter 2. Setup

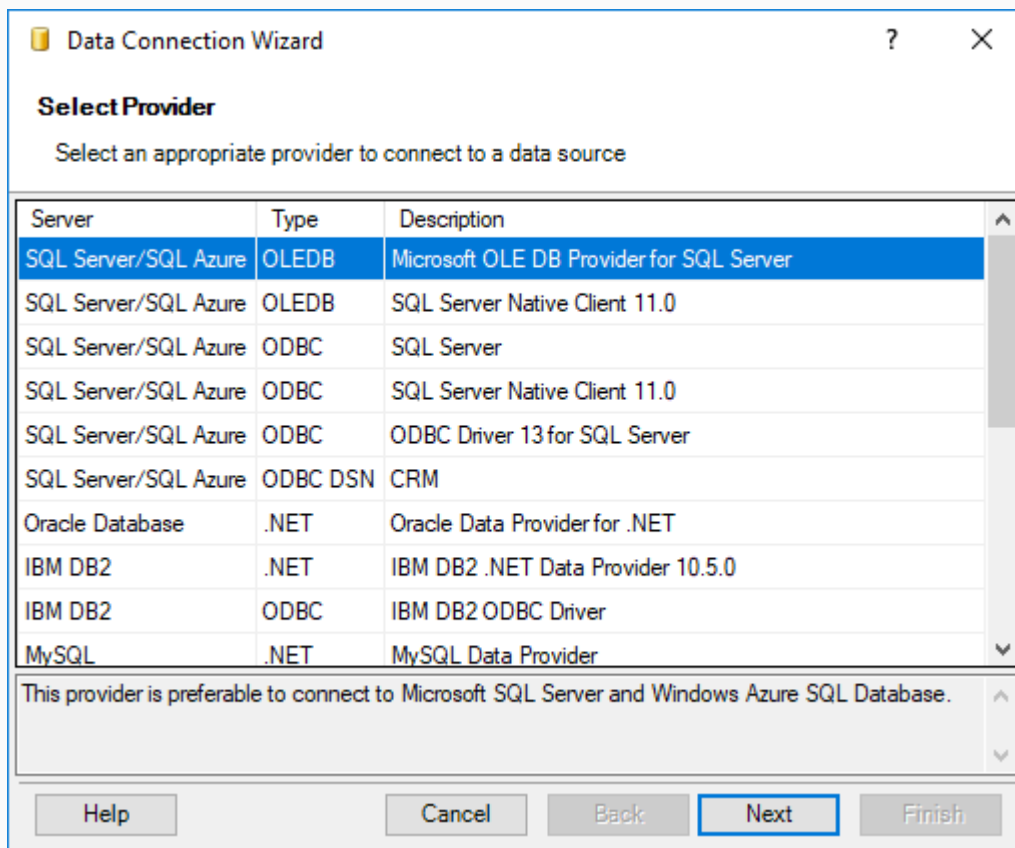
Creating Administrator's Workbook

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

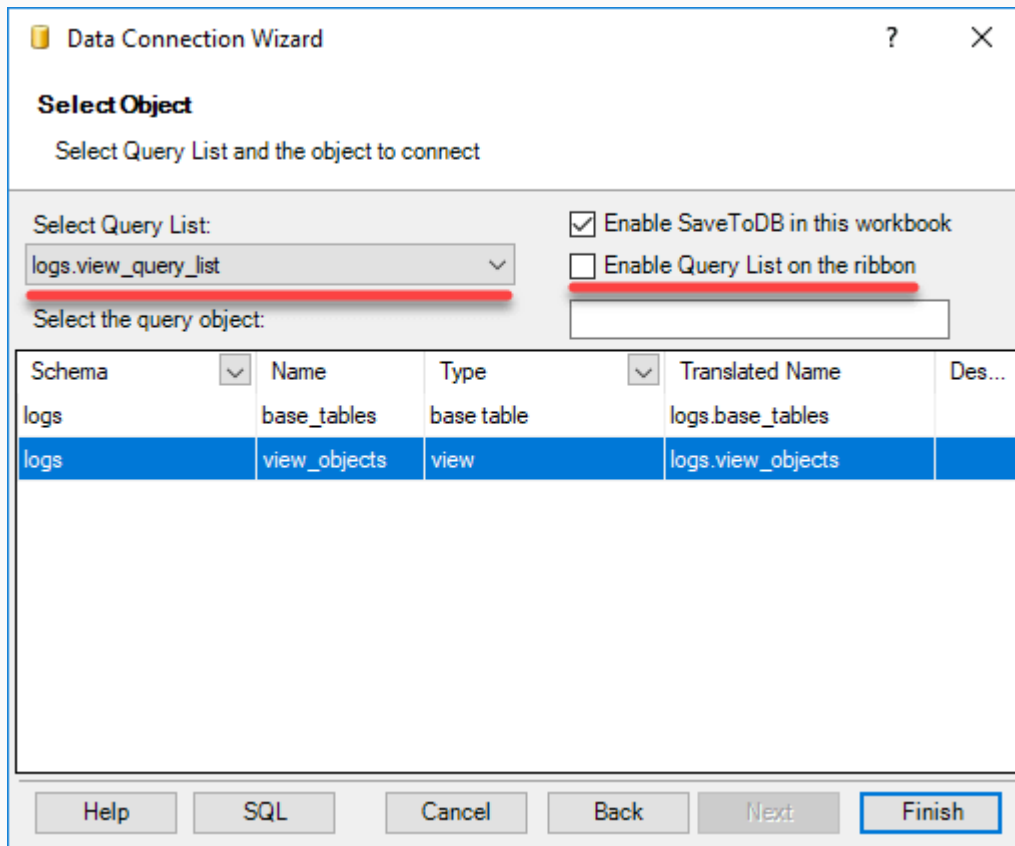
Create a new workbook and run **Wizards, Data Connection Wizard**:



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**:



The 'Data Connection Wizard' dialog box is shown with the 'Select Object' tab selected. The 'Select Query List' dropdown is set to 'logs.view_query_list'. The 'Enable SaveToDB in this workbook' checkbox is checked, and the 'Enable Query List on the ribbon' checkbox is unchecked. The 'Select the query object:' text box is empty. Below this is a table with columns: Schema, Name, Type, Translated Name, and Des... The table contains two rows: 'logs' with 'base_tables' (base table) and 'logs.view_objects' (view). The 'logs.view_objects' row is highlighted in blue. At the bottom are buttons for Help, SQL, Cancel, Back, Next, and Finish.

Data Connection Wizard

Select Object

Select Query List and the object to connect

Select Query List: logs.view_query_list

☒ Enable SaveToDB in this workbook

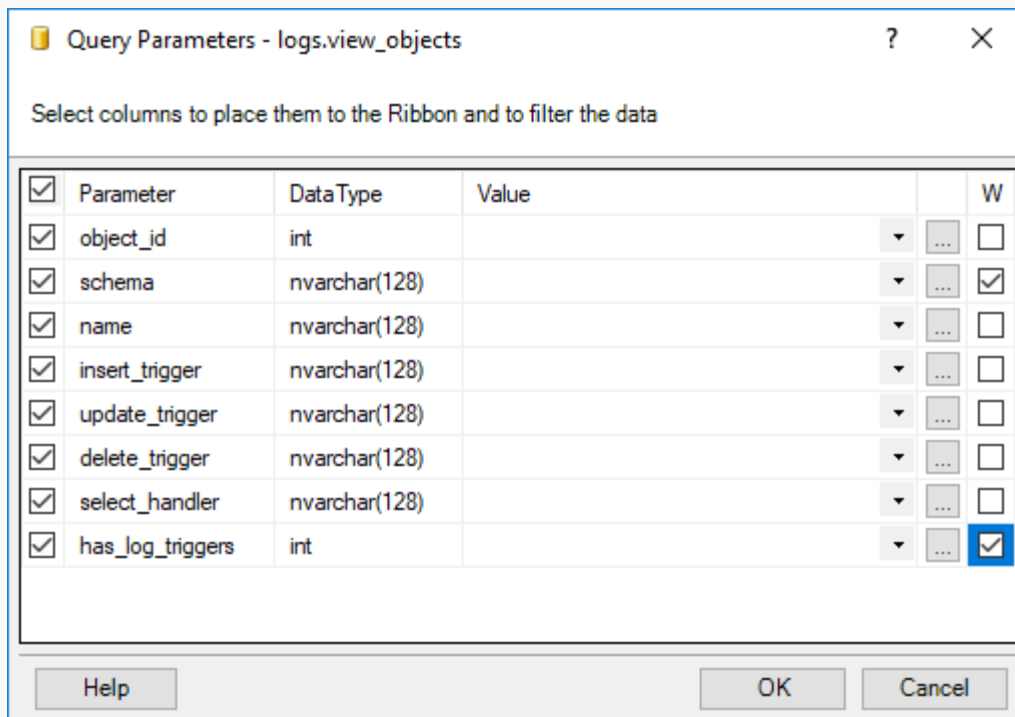
☐ Enable Query List on the ribbon

Select the query object:

Schema	Name	Type	Translated Name	Des...
logs	base_tables	base table	logs.base_tables	
logs	view_objects	view	logs.view_objects	

Help SQL Cancel Back Next Finish

In the next screen, select the **schema** and **has_log_triggers** field to place to the ribbon to filter data and click **OK**:



The 'Query Parameters - logs.view_objects' dialog box is shown. The title bar says 'Query Parameters - logs.view_objects'. The instruction 'Select columns to place them to the Ribbon and to filter the data' is at the top. Below is a table with columns: Parameter, DataType, Value, and W. The table contains eight rows, each with a checkbox in the 'Parameter' column. The 'has_log_triggers' row has its checkbox checked. At the bottom are buttons for Help, OK, and Cancel.

Query Parameters - logs.view_objects

Select columns to place them to the Ribbon and to filter the data

Parameter	DataType	Value	W
<input checked="" type="checkbox"/> object_id	int		<input type="checkbox"/>
<input checked="" type="checkbox"/> schema	nvarchar(128)		<input checked="" type="checkbox"/>
<input checked="" type="checkbox"/> name	nvarchar(128)		<input type="checkbox"/>
<input checked="" type="checkbox"/> insert_trigger	nvarchar(128)		<input type="checkbox"/>
<input checked="" type="checkbox"/> update_trigger	nvarchar(128)		<input type="checkbox"/>
<input checked="" type="checkbox"/> delete_trigger	nvarchar(128)		<input type="checkbox"/>
<input checked="" type="checkbox"/> select_handler	nvarchar(128)		<input type="checkbox"/>
<input checked="" type="checkbox"/> has_log_triggers	int		<input checked="" type="checkbox"/>

Help OK Cancel

Select cell **B3** to insert a table and click **OK**.

Input ? X

Select a cell to insert a query

\$B\$3

OK Cancel

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

The screenshot shows the Excel interface with the SaveToDB add-in ribbon. The 'Active Query' is 'logs.view_objects' and the 'schema' is 'has_log_triggers'. The 'Table View' is set to 'Table Data'. The table is displayed in the worksheet with columns: object_id, schema, name, insert_trigger, update_trigger, delete_trigger, and select_handler. The data is as follows:

object_id	schema	name	insert_trigger	update_trigger	delete_trigger	select_handler
1487134	dbo25	units				
129487590	dbo25	dimensions				
257488046	dbo25	form_permissions				
305488217	dbo25	dimension_calc_types				

Format the table as you like and save the workbook:

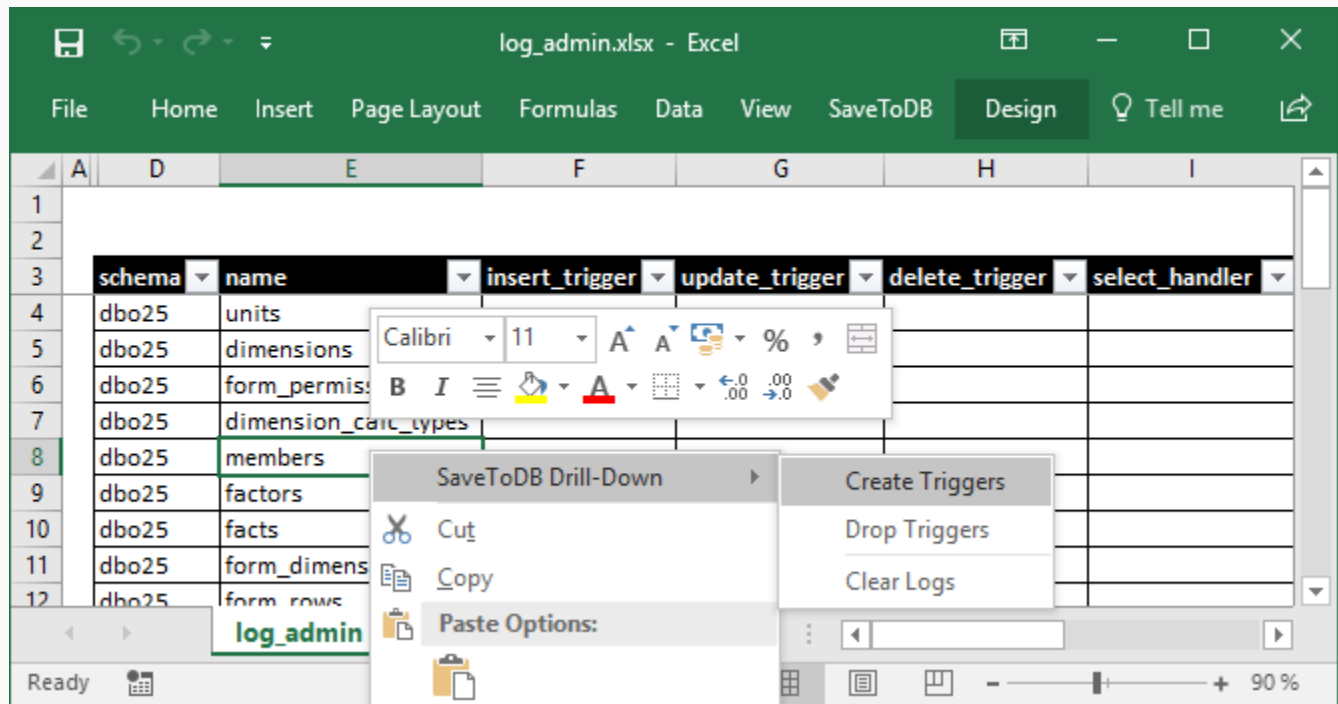
The screenshot shows the Excel interface with the log_admin table. The table has columns: schema, name, insert_trigger, update_trigger, delete_trigger, and select_handler. The data is as follows:

schema	name	insert_trigger	update_trigger	delete_trigger	select_handler
dbo25	units				
dbo25	dimensions				
dbo25	form_permissions				
dbo25	dimension_calc_types				
dbo25	members				
dbo25	factors				
dbo25	facts				
dbo25	form_dimensions				
dbo25	form_rows				

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:

Query Parameters - Create Triggers

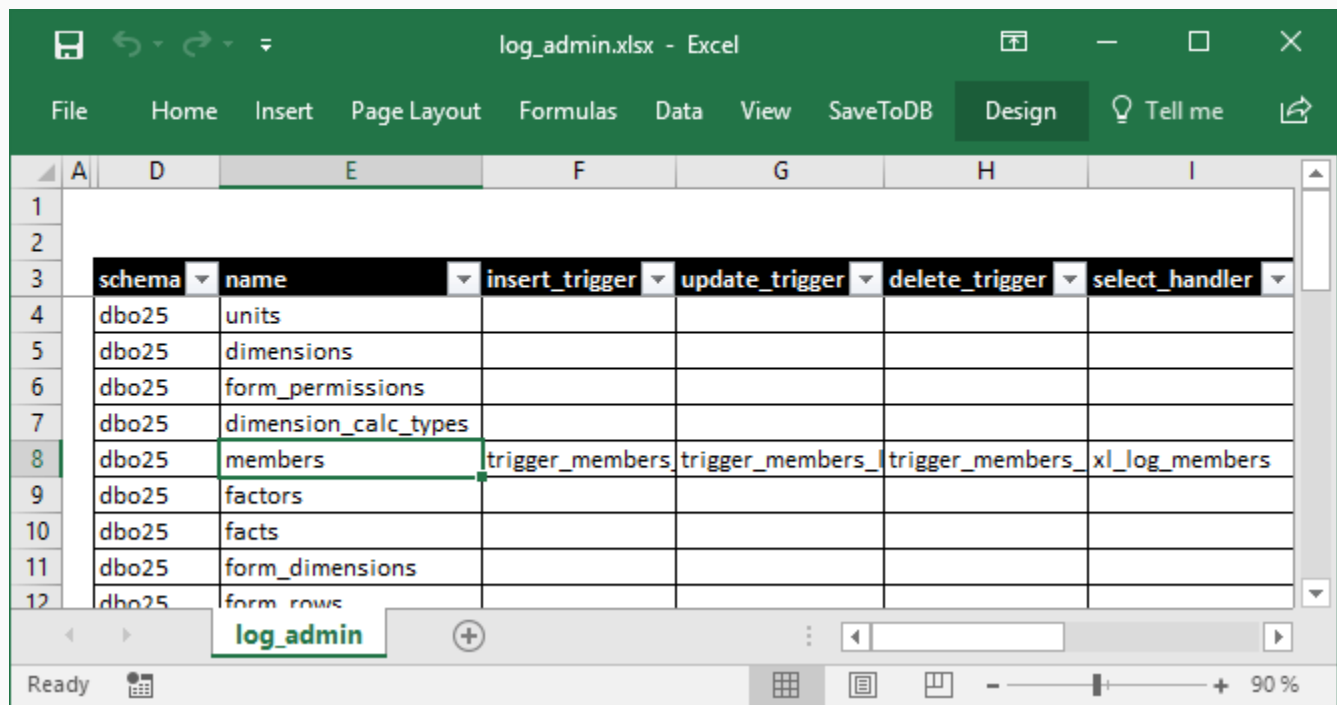
Edit procedure parameters

Parameter	DataType	Value	W
schema	nvarchar(128)	dbo25	<input checked="" type="checkbox"/>
name	nvarchar(128)	members	<input checked="" type="checkbox"/>
execute_script	bit	1	<input checked="" type="checkbox"/>

Help OK Cancel

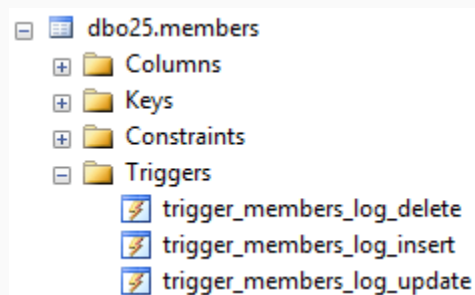
Check the target table and click **OK**.

The SaveToDB add-in updates the table, and you see actual triggers for the captured tables:



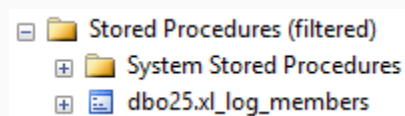
	schema	name	insert_trigger	update_trigger	delete_trigger	select_handler
4	dbo25	units				
5	dbo25	dimensions				
6	dbo25	form_permissions				
7	dbo25	dimension_calc_types				
8	dbo25	members	trigger_members	trigger_members	trigger_members	xl_log_members
9	dbo25	factors				
10	dbo25	facts				
11	dbo25	form_dimensions				
12	dbo25	form_rows				

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:



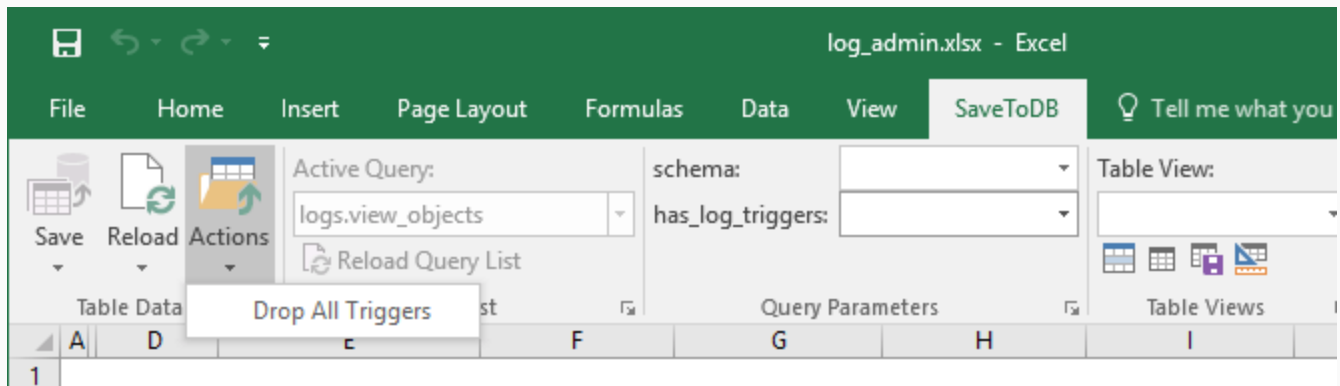
Note that the framework creates procedures in the schemas of underlying tables.

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:

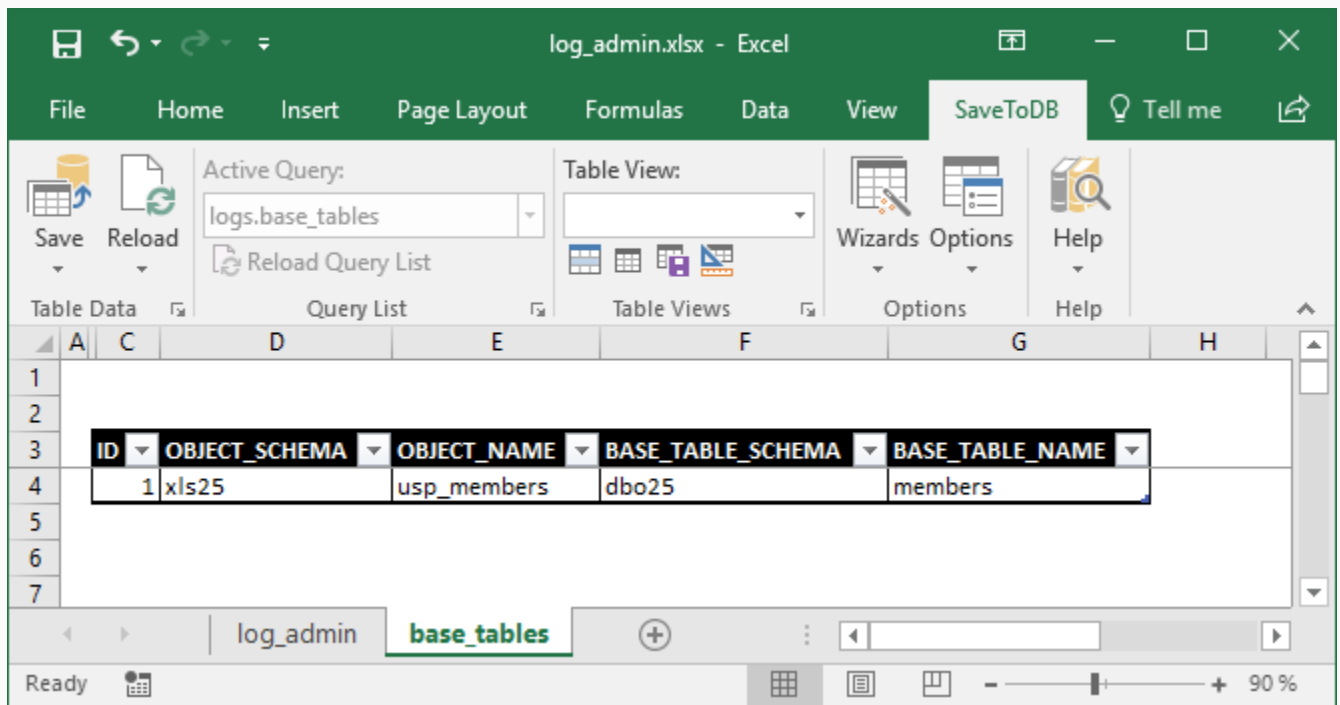


Attaching Context Menus

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.



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.

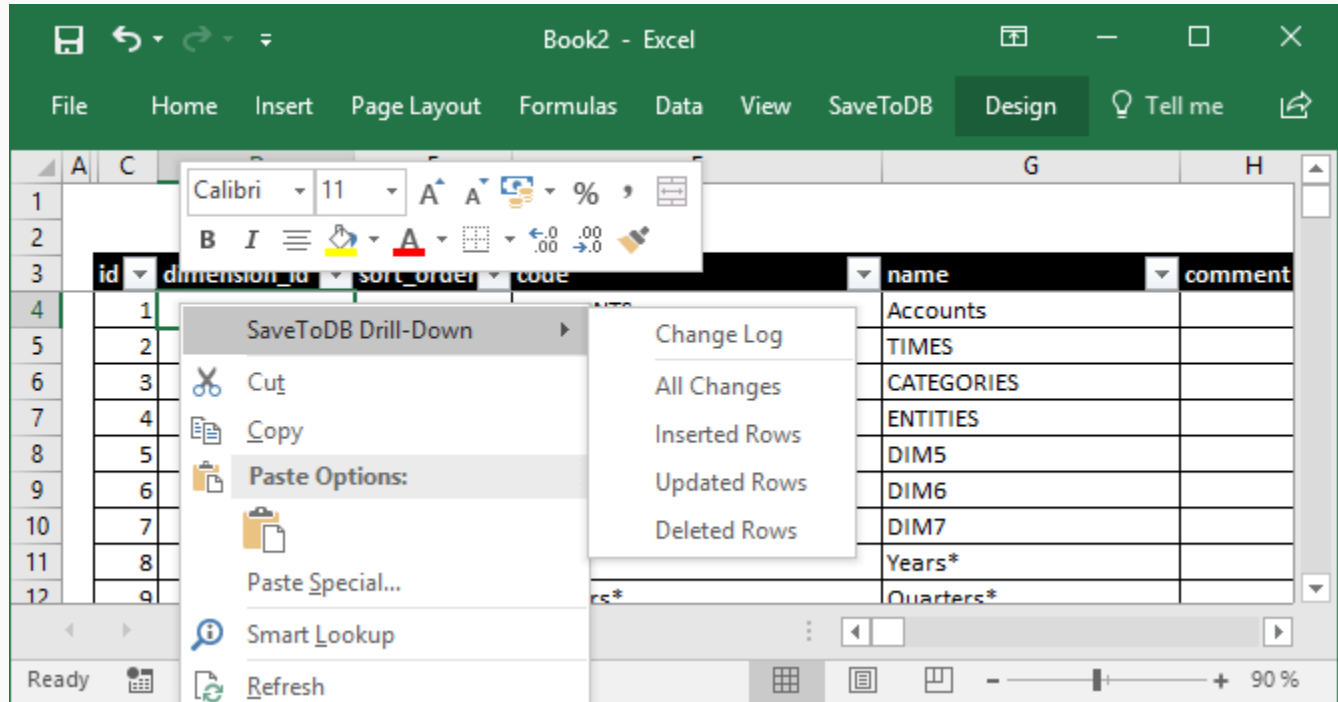
They need click **Reload**, **Reload Data and Configuration** to load new context menu items.

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

Chapter 3. Usage

Context Menu

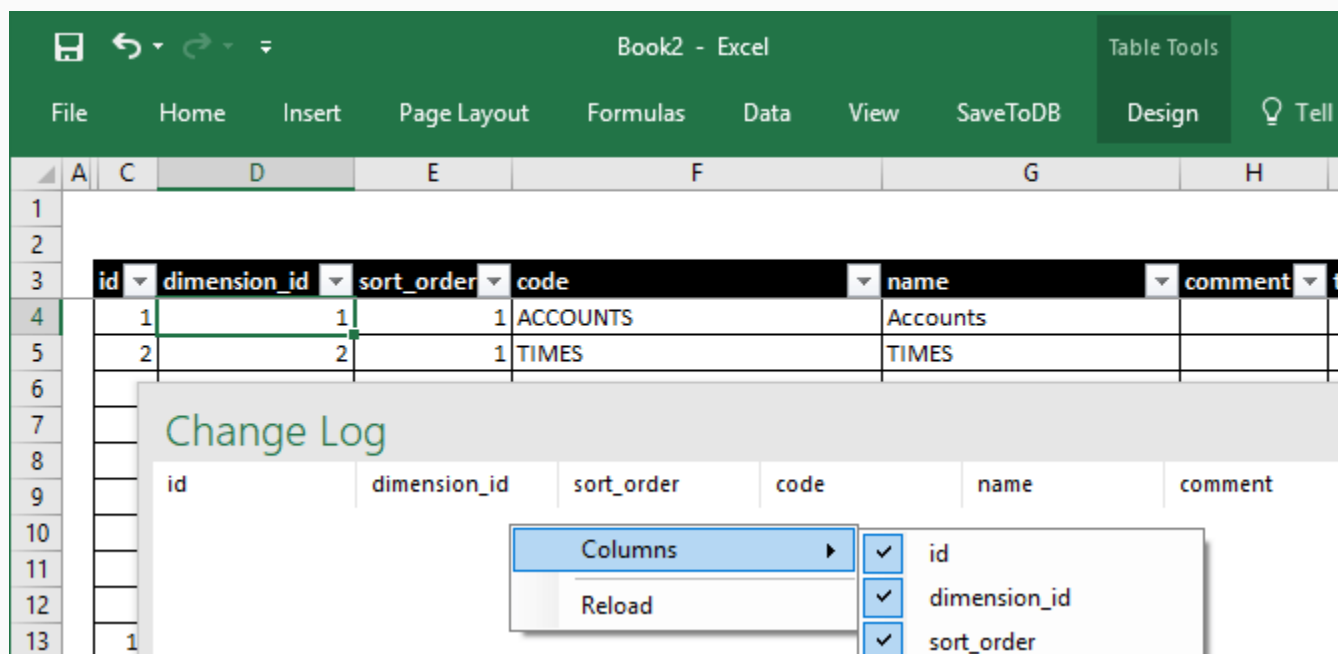
The framework creates procedures to select changes of specific rows or entire tables by change types:



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:

Change Log				
id	name	Change Action	Change Date	Change User

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

The screenshot shows the Microsoft Excel interface with the 'SaveToDB' task pane open. The 'Active Query' is 'dbo25.members'. The 'Table View' shows a table with the following data:

id	dimension_id	sort_order	code	name	comment
1	1	1	ACCOUNTS	Accounts2	
2	2	2	TIMES	TIMES	
3	3	3	CATEGORIES	CATEGORIES	
4	4	4	ENTITIES	ENTITIES	

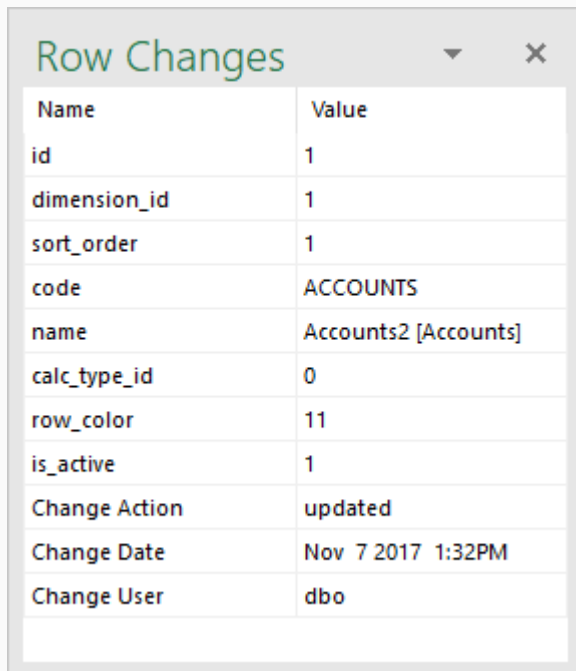
The 'Reload' button in the task pane is highlighted. The Excel status bar shows 'Ready' and '90 %' zoom.

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

Change Log				
id	name	Change Action	Change Date	Change User
1	Accounts2	updated	07.11.2017 13:32	dbo

You see the changes.

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



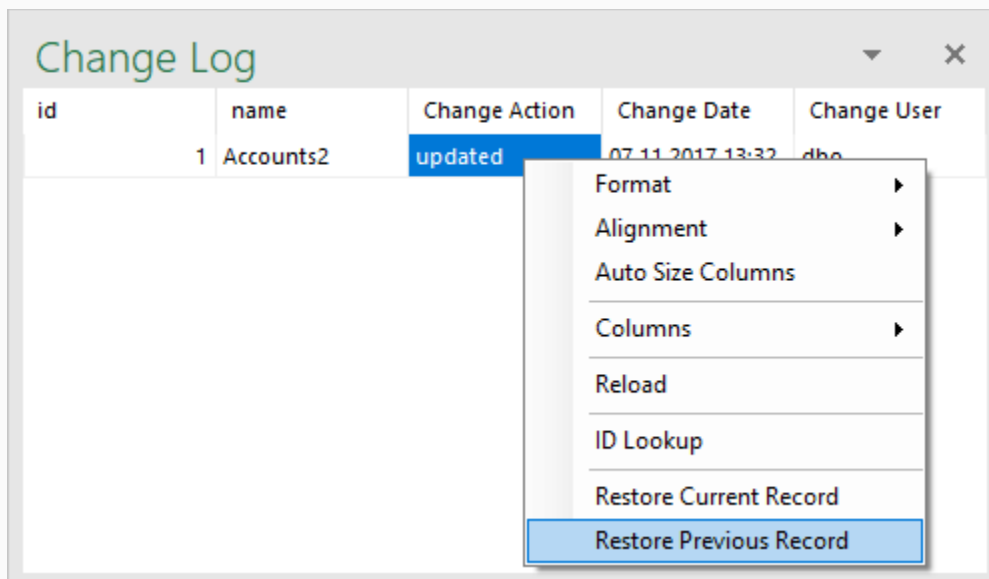
The 'Row Changes' task pane displays a table with two columns: 'Name' and 'Value'. The table contains the following data:

Name	Value
id	1
dimension_id	1
sort_order	1
code	ACCOUNTS
name	Accounts2 [Accounts]
calc_type_id	0
row_color	11
is_active	1
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:



The 'Change Log' task pane displays a table with five columns: 'id', 'name', 'Change Action', 'Change Date', and 'Change User'. The first row contains the following data:

id	name	Change Action	Change Date	Change User
1	Accounts2	updated	07.11.2017 13:32	dbo

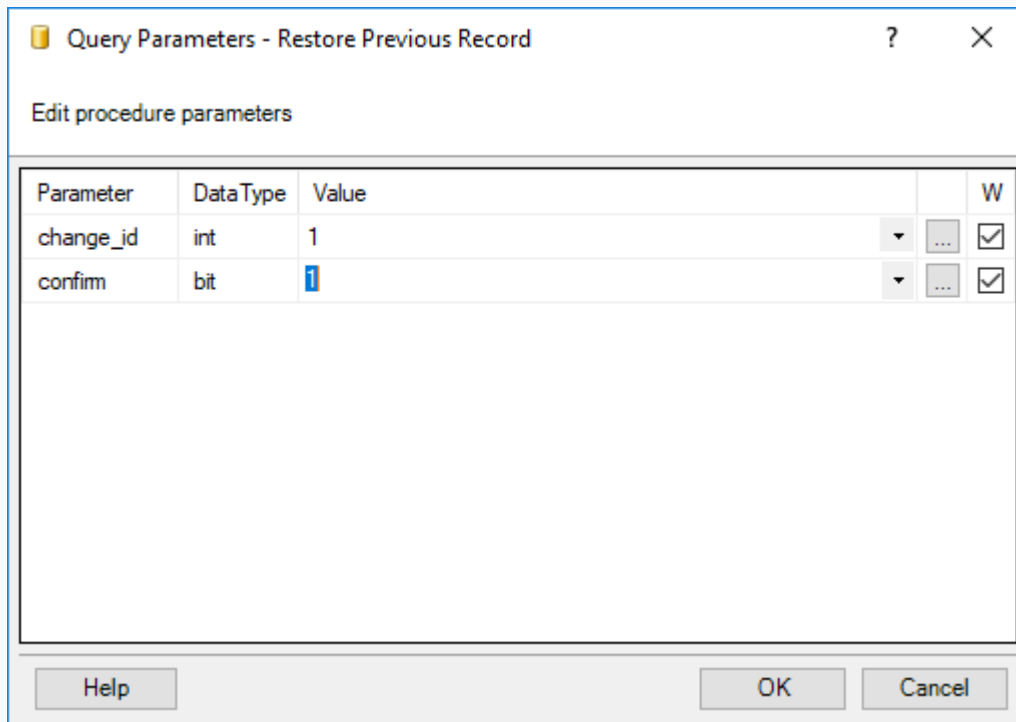
A context menu is open over the 'updated' cell, showing the following options:

- Format
- Alignment
- Auto Size Columns
- Columns
- Reload
- ID Lookup
- Restore Current Record
- Restore Previous Record

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**:



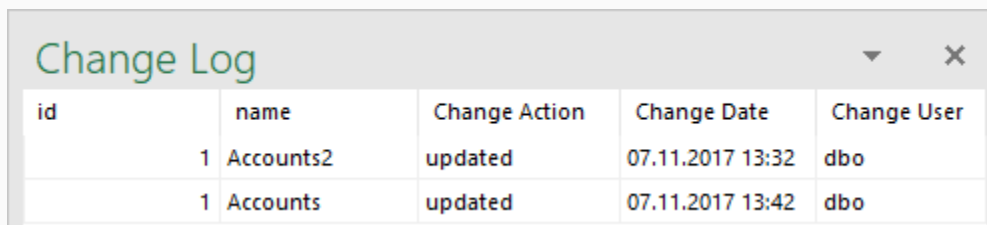
Query Parameters - Restore Previous Record

Edit procedure parameters

Parameter	Data Type	Value		W
change_id	int	1	...	<input checked="" type="checkbox"/>
confirm	bit	1	...	<input checked="" type="checkbox"/>

Help OK Cancel

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



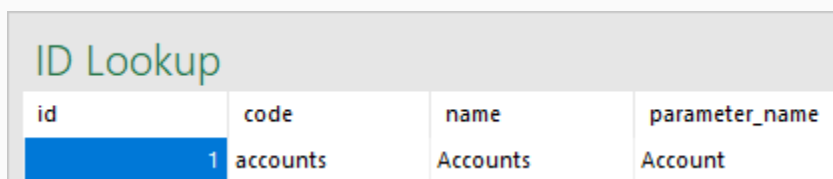
id	name	Change Action	Change Date	Change User
1	Accounts2	updated	07.11.2017 13:32	dbo
1	Accounts	updated	07.11.2017 13:42	dbo

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.

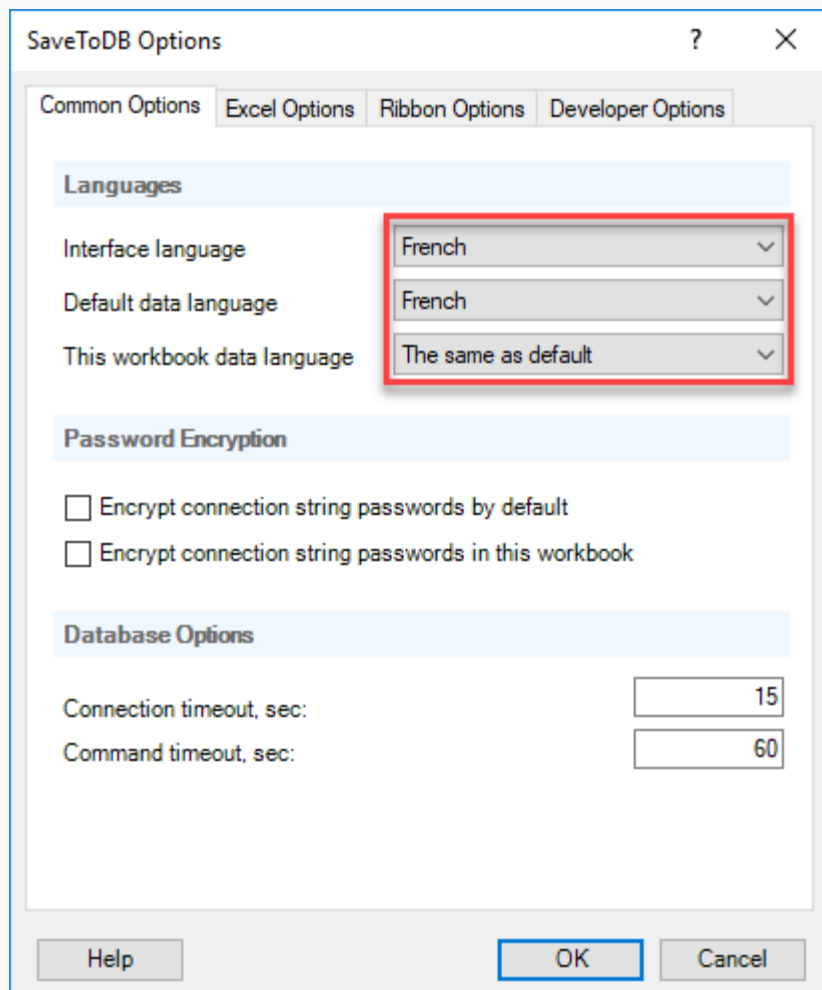
For example, this screen shows a dimension row of the account member:



id	code	name	parameter_name
1	accounts	Accounts	Account

Translation

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



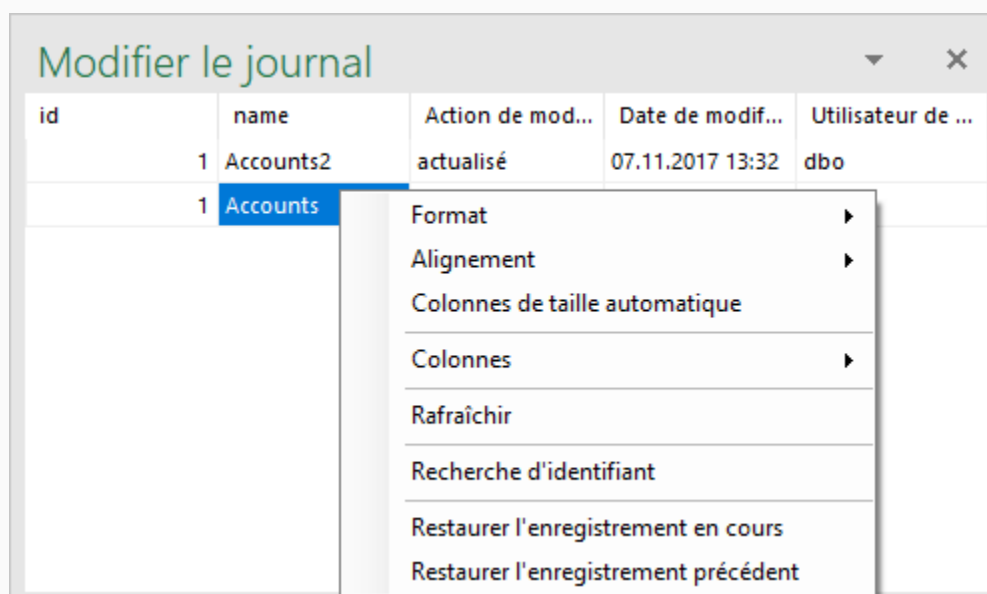
The image shows the 'SaveToDB Options' dialog box with the 'Common Options' tab selected. The 'Languages' section contains three dropdown menus: 'Interface language' (French), 'Default data language' (French), and 'This workbook data language' (The same as default). These three dropdowns are highlighted with a red rectangle. Below the 'Languages' section is the 'Password Encryption' section with two unchecked checkboxes: 'Encrypt connection string passwords by default' and 'Encrypt connection string passwords in this workbook'. The 'Database Options' section contains two text boxes: 'Connection timeout, sec:' (15) and 'Command timeout, sec:' (60). At the bottom are 'Help', 'OK', and 'Cancel' buttons.

Languages	
Interface language	French
Default data language	French
This workbook data language	The same as default

Database Options	
Connection timeout, sec:	15
Command timeout, sec:	60

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

Voilà:



The image shows a window titled 'Modifier le journal' (Modify the log) with a table of log entries. The table has columns: id, name, Action de mod..., Date de modif..., and Utilisateur de ... The first row shows '1 Accounts2' with action 'actualisé' and user 'dbo'. The second row shows '1 Accounts' which is selected, and a context menu is open over it. The context menu options are: Format, Alignement, Colonnes de taille automatique, Colonnes, Rafraîchir, Recherche d'identifiant, Restaurer l'enregistrement en cours, and Restaurer l'enregistrement précédent.

id	name	Action de mod...	Date de modif...	Utilisateur de ...
1	Accounts2	actualisé	07.11.2017 13:32	dbo
1	Accounts			

- Format
- Alignement
- Colonnes de taille automatique
- Colonnes
- Rafraîchir
- Recherche d'identifiant
- Restaurer l'enregistrement en cours
- Restaurer l'enregistrement précédent

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:

```
EXEC sp_addrolemember 'log_users', 'pa_user_01'
```

Chapter 5. Database Objects

Roles

The change tracking framework creates the following roles:

- log_administrators
- log_users

log_administrators

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

See actual database permissions in the **logs.usp_set_role_permissions_administrators** procedure.

log_users

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

See actual database permissions in the **logs.usp_set_role_permissions_users** procedure.

Schemas

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:

- base_tables
- change_logs
- column_translations
- event_handlers
- object_translations

base_tables

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

See [Attaching Context Menus](#).

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.

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

```
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

```
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/