

Laserfiche Forms – Tables/Collections CRUD Operations with SQL

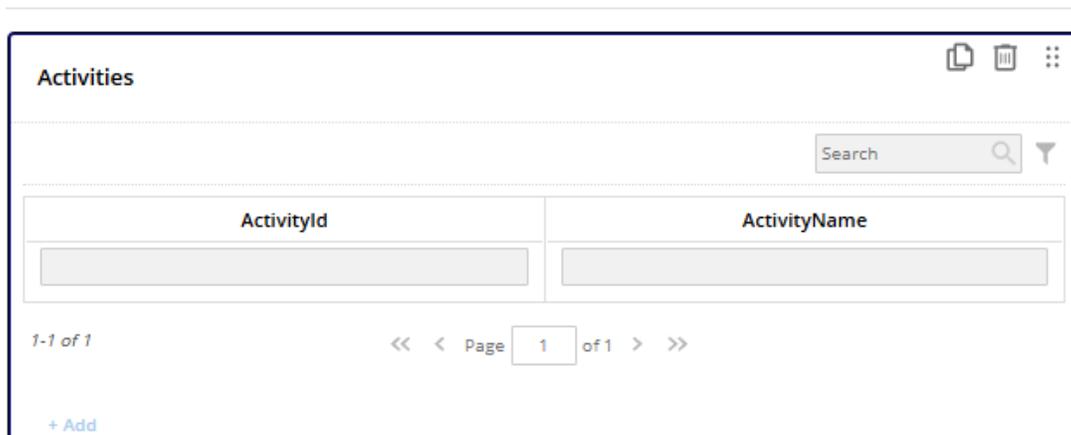
This document outlines how to create a table or collection in a form that will update a SQL database table including inserts, updates, and deletes (CRUD - **C**reate, **R**ead, **U**ppdate, **D**elete) using Laserfiche Forms Modern Designer and Workflow.

The sample uses a table called Activities. The table should be created with a unique identifier column set up as integer to auto increment, in this case ActivityId.

```
CREATE TABLE [dbo].[Activities](
    [ActivityId] [int] IDENTITY(1,1) NOT NULL,
    [ActivityName] [varchar](200) NULL
CONSTRAINT [PK_Activities] PRIMARY KEY CLUSTERED
(
    [ActivityId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF, ALLOW_ROW_LOCKS =
ON, ALLOW_PAGE_LOCKS = ON, OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
```

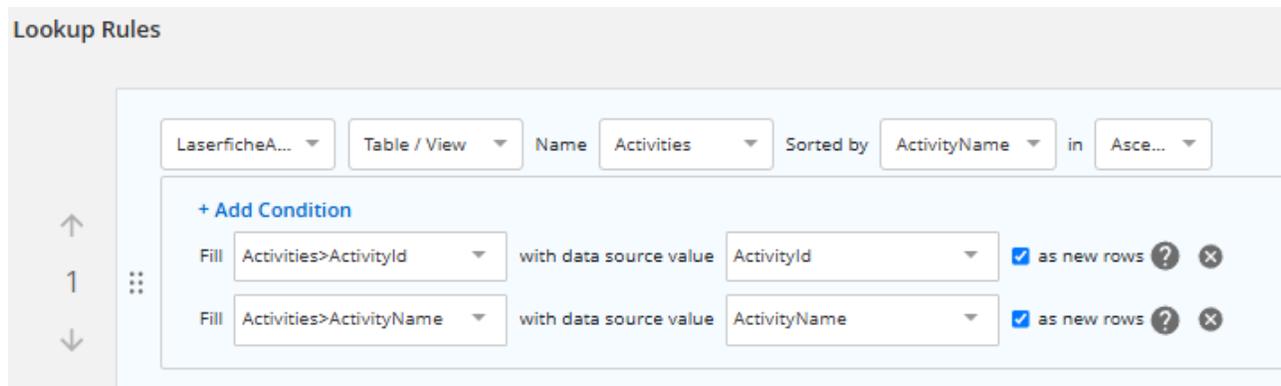
Create a form using the modern designer then add a table with fields for ActivityId and ActivityName. The ActivityId can be set to readonly and can optionally be hidden using field rules

CRUD Sample



ActivityId	ActivityName
------------	--------------

Create a Lookup Rule for the table to be populated from the Activities table with results “as new sets”.



Lookup Rules

LaserficheA... Table / View Name Activities Sorted by ActivityName in Asce...

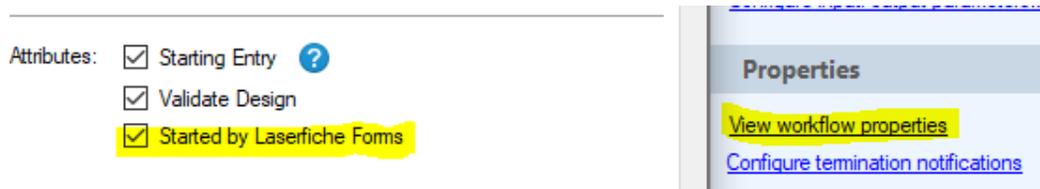
+ Add Condition

Fill Activities>ActivityId with data source value ActivityId as new rows ? x

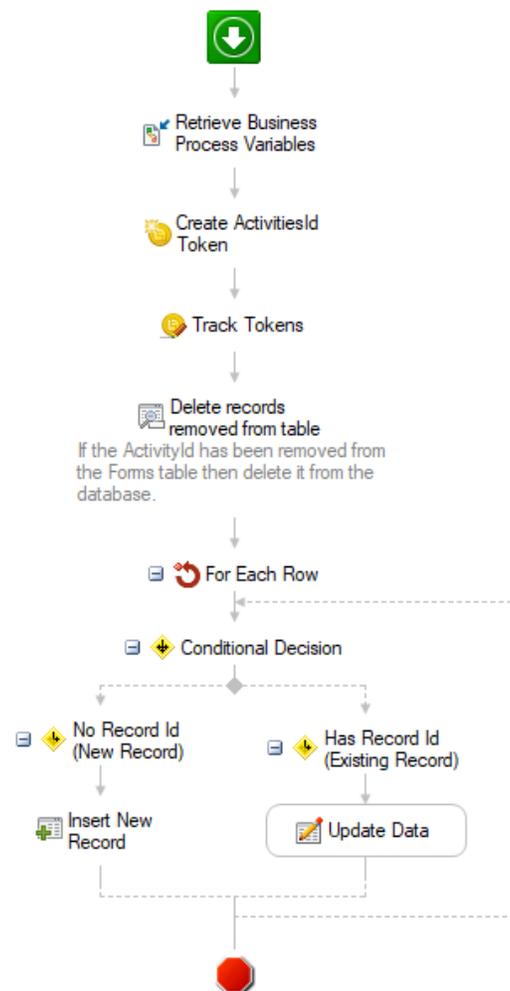
Fill Activities>ActivityName with data source value ActivityName as new rows ? x

Create a Workflow in Workflow Designer to update the SQL data using the data from the Forms table.

Set the Workflow Properties to be “Started by Laserfiche Forms”

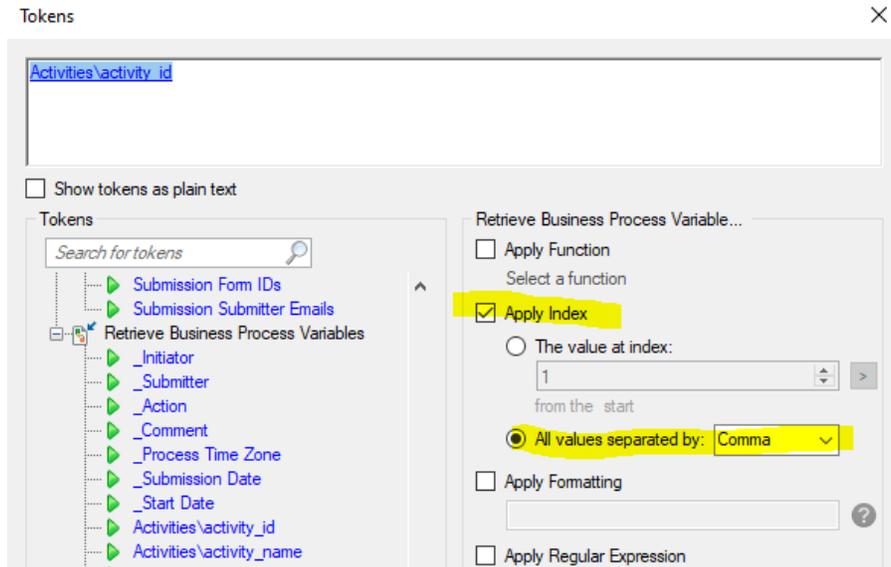


Overall Workflow



- **Retrieve Business Process Variables**
 - Set this step to retrieve the Activities table from the form
- **Create ActivityIds Token**
 - This token will get the activity_ids from the ActivitiesTable
 - Use the token builder dialog to Apply Index with comma separated values so that all record ids will be contained in a single string
 - This string will be passed to a stored procedure so that it can compare all the records returned from the form against the database table in order to determine which records have been deleted from the form table so that they can be deleted from the SQL table.

- The token is used in this manner to avoid doing a For Each Row loop.
- Passing this token to a Custom Query delete statement directly resulted in “conversion failed when converting the nvarchar value '2, 3, 4' to data type int” errors, so it was passed to a stored procedure instead so that the aggregated string could be parsed correctly.



- **Track Tokens**

- Optional step that is used for troubleshooting so that the token in the previous step can be viewed in the log.

- **Delete records removed from table**

- Custom query – takes token containing all ids returned from form table and passes it to a stored procedure which executes the deletes (**see *Activities_Delete stored procedure code below***).

Activity Name	
Delete records removed from table	
Activity Description	
If the ActivityId has been removed from the Forms table then delete it from the database.	
Data Source	
LaserficheAuxiliary	
Database Type: Sql (Direct)	
Database: [REDACTED]	
Authentication: [REDACTED]	
Custom Query	
Query:	
EXECUTE Activities_Delete @ActivityIds = @A	
Parameters:	
Name	Value
@A	% (ActivityIds)

- **For Each Row**

- Loop through each row of the forms table

Activity Name

For Each Row

Activity Description

Runs the contained activities for each row returned by

Get Rows From

Activity: Retrieve Business Process Variables
 Rows: Activities
[Select...](#)

- **No Record Id (New Record)**

- Check to see if the row returned from the table has a populated Id
- If there is no Id then it must be a new record; Ids are automatically generated in SQL when a record is inserted because of the Identity column that was set up in the table build.

Activity Name

No Record Id (New Record)

Activity Description

Contains activities that will run if the branch c

Condition

If all of these conditions are true
 %(ForEachRow_activity_id) is empty

- **Insert New Record**

- Don't need to insert the Id column as it is autogenerated by SQL, only insert other column(s).
- Use the Insert Data tool

Data to Insert

Select the table and then configure the values to insert in it.

Activities

Column	Value
ActivityName	%(ForEachRow_activity_name)

- **Has Record Id (Existing Record)**

- If the row returned from a table has Id populated then it already exists in the database and should be updated

Activity Name
Has Record Id (Existing Record)
Activity Description
Contains activities that will run if the branch conc
Condition
If all of these conditions are true %(ForEachRow_activity_id) is not empty

- **Update Data**

- Use the Update Data tool

Rows to Update				
Find the rows to be updated.				
ExcursionActivities				
<table border="1"> <thead> <tr> <th>Column</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>ActivityId</td> <td>%(ForEachRow_activity_id)</td> </tr> </tbody> </table>	Column	Value	ActivityId	%(ForEachRow_activity_id)
Column	Value			
ActivityId	%(ForEachRow_activity_id)			
Add...				
New Values				
Update the rows found with these new values:				
<table border="1"> <thead> <tr> <th>Column</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>ActivityName</td> <td>%(ForEachRow_activity_name)</td> </tr> </tbody> </table>	Column	Value	ActivityName	%(ForEachRow_activity_name)
Column	Value			
ActivityName	%(ForEachRow_activity_name)			

Save and publish the Workflow without any starting rules.

Create the Stored Procedure used by the “Delete records removed from table step”

```

CREATE PROC [dbo].[Activities_Delete]
(@ActivityIds VARCHAR(MAX))

AS

WITH ActivityIds AS
(SELECT
    value AS ActivityId
FROM
    STRING_SPLIT(@ActivityIds,','))

DELETE FROM
    Activities
WHERE
    ActivityId NOT IN (SELECT ActivityId FROM ActivityIds)
    
```

In the Forms Process Designer, add a Workflow Service Task to run the workflow after the Form submission.



The form table should now be able to create, update, and delete records from a SQL Table.

Activities

Search 🔍 ⌵

ActivityId	ActivityName	
<input type="text" value="3"/>	<input type="text" value="Activity One"/>	×
<input type="text" value="4"/>	<input type="text" value="Activity Two"/>	×

[+ Add](#)