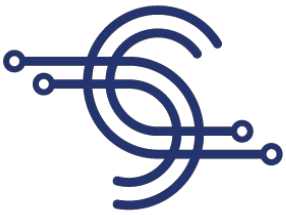


Conflexion
Custom Modeling Solutions for
Planning, Scheduling, Analysis

P.O. Box 824
Auburn, AL 36831-0824
(334) 444-3492
http://conflexion.com

# Simio Output Tables

## Problem Description

We have a Simio model that uses an external process requiring access to the complete contents of a Simio output table. The external process output could be any of: Adding new rows, Changing the attributes of existing rows, and/or Deleting existing rows. In this example Simio will hold the *golden* copy of the data, but we envision other cases in which the *golden* copy would be externally with Simio pushing updates asynchronously and pulling data as needed.

## System Requirements

To run this example as-is you'll need:

- Simio with a better-than-personal license (too many process steps)
- A SQL Server on which you can create a database. The Simio processes work with any database you can connect with DbReadWrite but the stored procedures for this example are SQL Server specific.

## High-Level Process

In this example model we'll use Simio's DbReadWrite user extension to replicate the output table into a SQL Server database. The Simio process outline is:

1. Simio pushes each row via Search + DbWrite steps
2. Execute the external process (simulated by a SQL Server stored procedure)
3. Simio pulls each row via Search + DbRead steps
4. Simio pulls deletions via a DbQuery step in a loop
5. Simio pulls additions via a DbQuery step in a loop (DbRead is not possible because it requires a search condition)

## Getting Started (initialize the system)

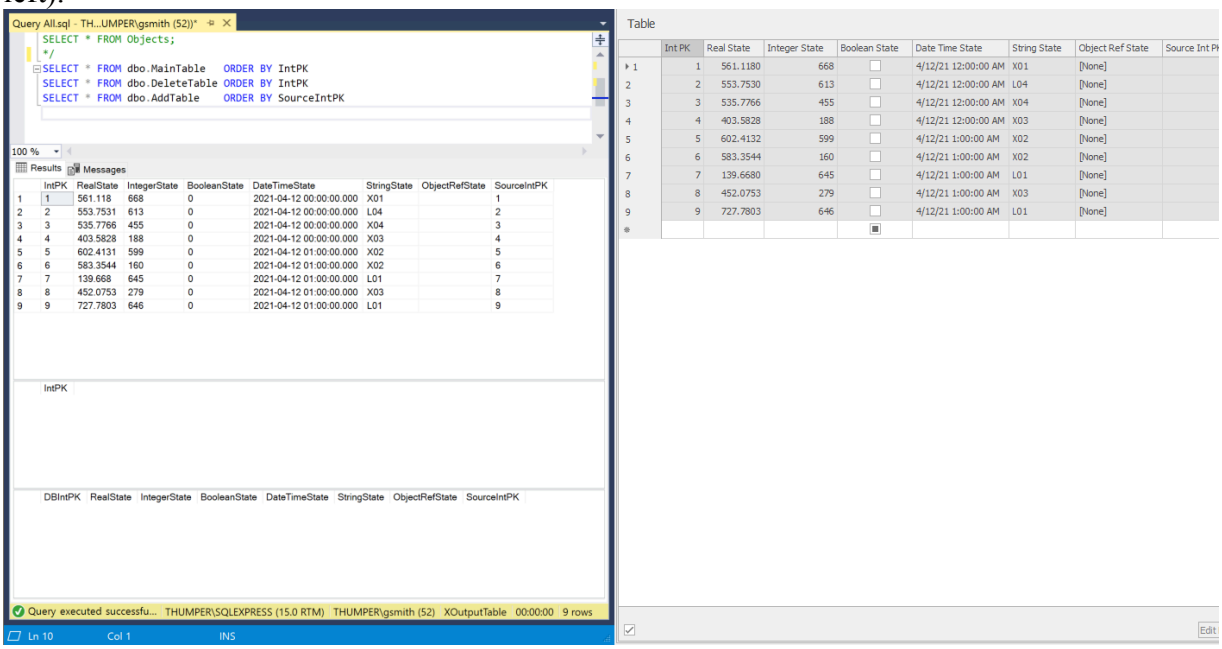
- Execute the *InitializeDatabase.sql* script in SQL Server Management Studio to create the tables and example stored procedures.
- Set the DbConnect Parameters (change as required for your system)
    - ConnectionString: Server=localhost\SQLEXPRESS;Database=XOutputTable;Trusted_Connection=True;
    - Provider Name: SqlClient Data Provider

- Configure the Simio Data Importer to work with your database server.

## Running the Example

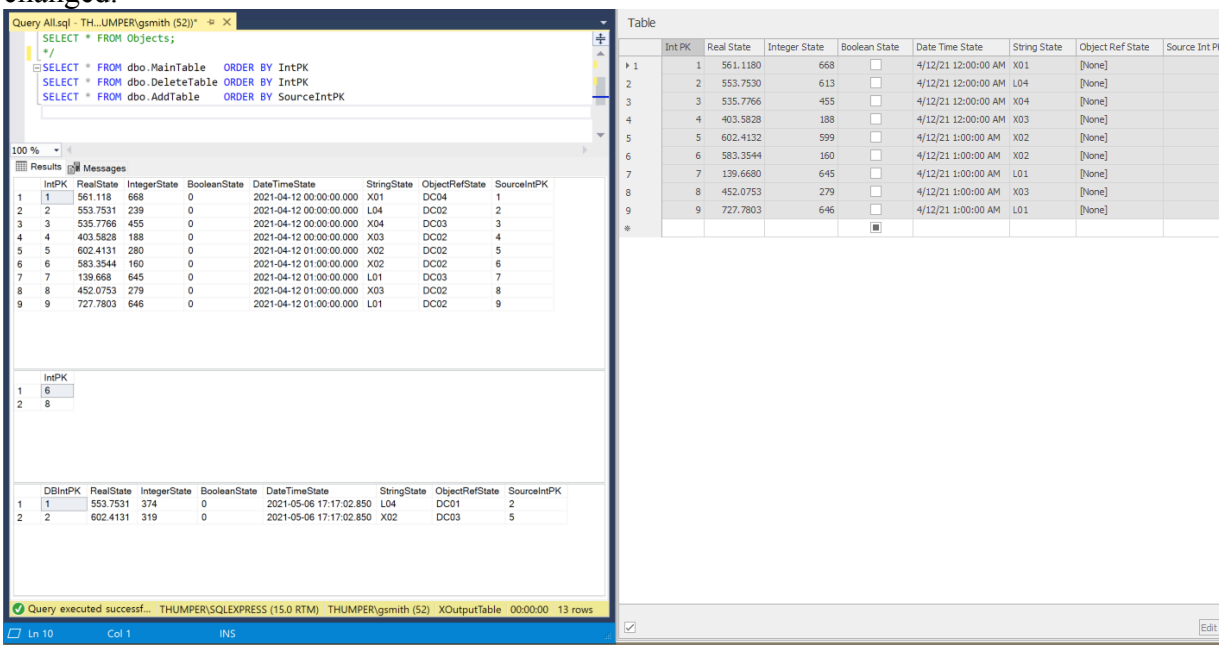
1. Open the Simionly.spfx model in Simio
2. Depress Simio's run button. The process steps outlined have a breakpoint between each of the listed steps so that you can examine the database and Simio tables to see how each part works. Depress the run button again after examining the tables at each breakpoint.

    a. **Post Write** – Simio has generated 9 rows into the output table (image on right) these have been written, using Search + DbWrite, to the database table (image on left).

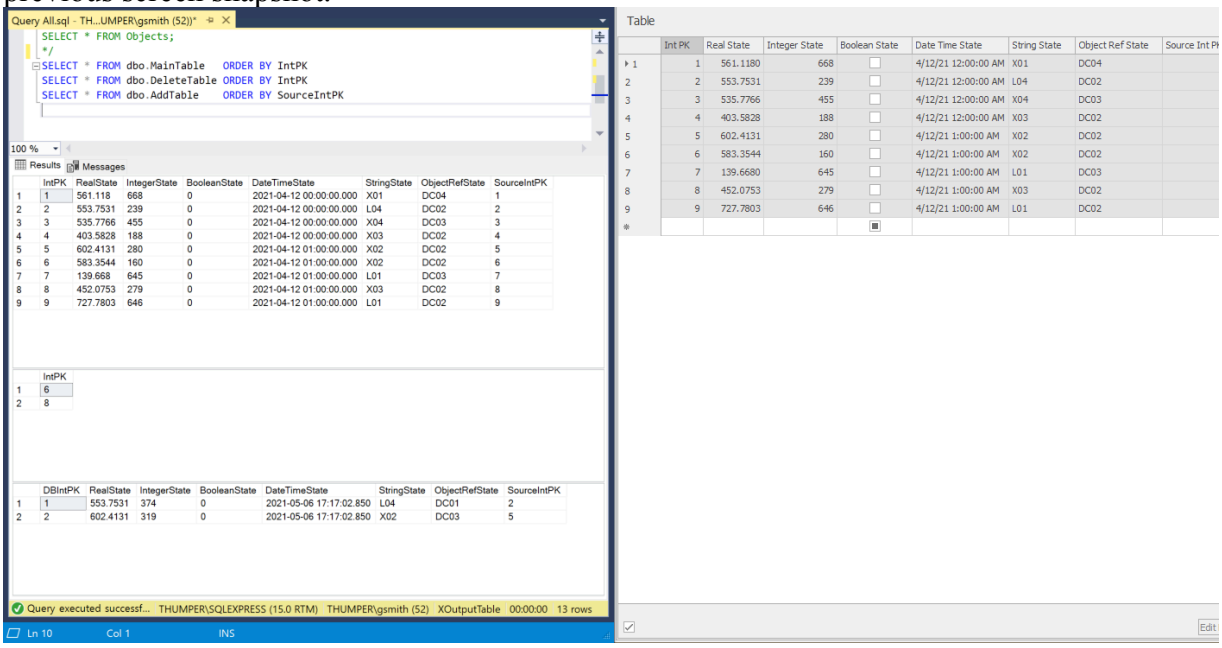



    b. **Post Raptomize** (Random simulation of Optimizer). The database stored procedure *Raptomize* has selected 2 random rows for deletion (6 and 8), 2 different random rows to be parents for adding rows (2 added and the "parents" in the main table are edited – hence difference between the database tables and the Simio output table for rows 2 and 5). Note that the Simio output table has not

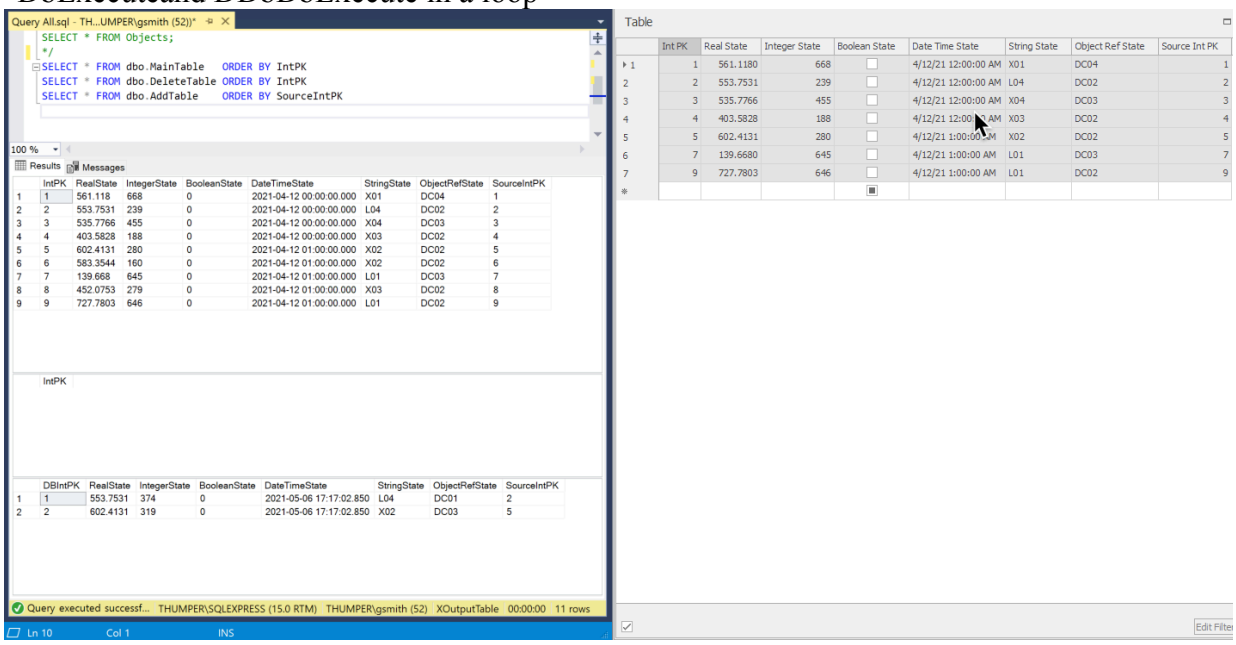
changed.




c. **Post Read** The database MainTable has been imported, via Search + DbRead, into the Simio output table. Note that the ObjectRefState attributes are populated and that the edited rows, 2 and 5, have a different IntegerState value than in the previous screen snapshot.



d. **Post Delete** The rows from the database DeleteTable table ( 6 and 8 ) have been deleted from that table and from Simio's output table. These use DbQuery and DbExecuteand DDbDbExecute in a loop




e. **Post Add** The rows from the database AddTable (DbIntPK 1 and 2, based on IntPK rows 2 and 5, respectively) have been deleted from that table and added to the Simio output table with the new IntPK's 10 and 11