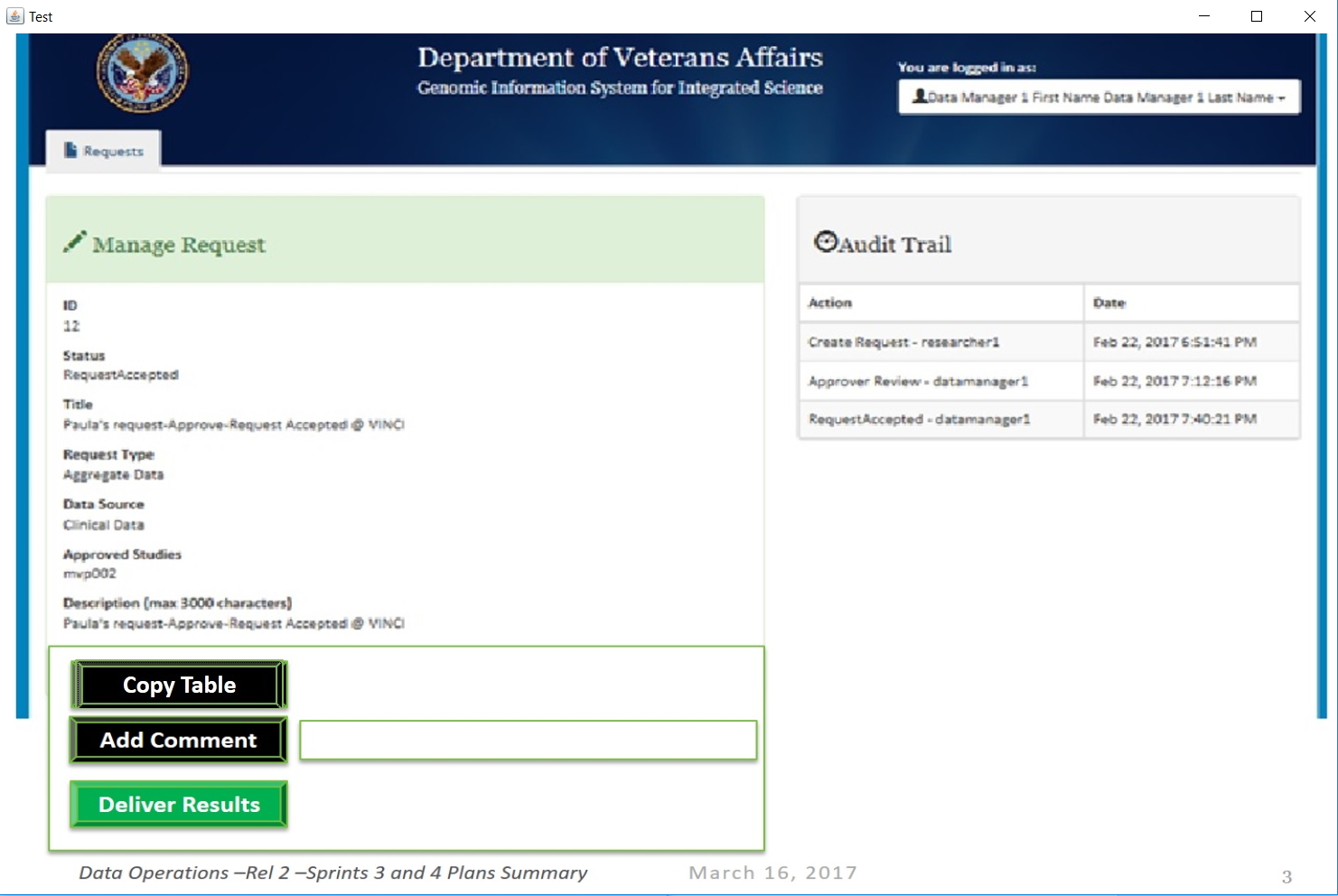
Genisis2 Data Operations – Table Copy Demonstration Prototype

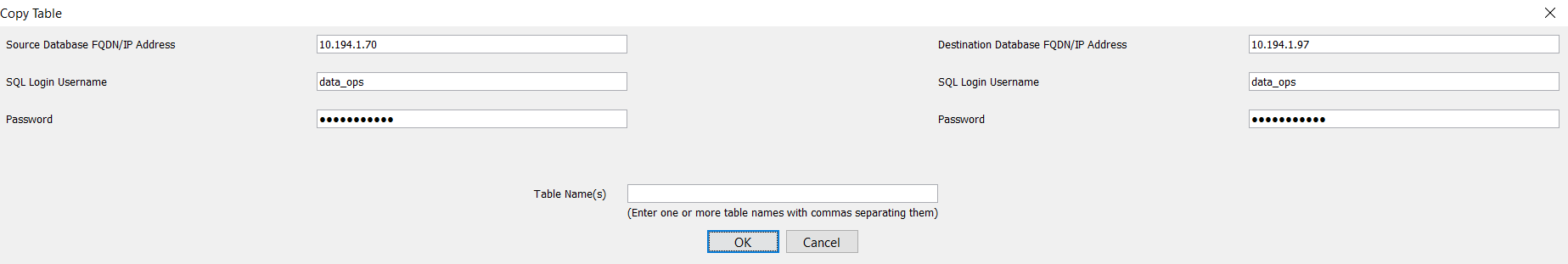
3/24/17

Demonstration Prototype Flow

First, the program will create a new table copy frame to set up for the prototype. A pane is then added to the frame with the mock-up image and button for users to press.



When the "Copy Table" button is pressed, it will generate a pop up menu with pre-initialized test log in information so that all the user should enter are the table names that they would want copied. After the information is deemed to be correct, press the ‘OK’ button.



Java Code

When the information is entered, it will take in the entered text as variables storing everything except the passwords as strings, and storing the passwords as char arrays. Due to the use of linked servers in sql server management studio, the source username and passwords are not required, just the name of the linked server.

**if** (result == JOptionPane.***OK\_OPTION***)

//when all the information is correct in the fields, press the default pop-up okay option

{

//get the text input into the fields and initialize variables for parameterization

String sIP = sourceIP.getText();

//no longer needed due to use of Linked Servers

//String sUsername = sourceUsername.getText();

//char[] sPasswords = sourcePassword.getPassword();

//String sPassword = new String(sPasswords);

String dIP = destIP.getText();

String dUsername = destUsername.getText();

**char**[] dPasswords = destPassword.getPassword();

String dPassword = **new** String(dPasswords);

The table field will be stored as a string and then turned into an array of strings separating where the user entered a comma. The table names now stored in a string array will be trimmed to remove any trailing white space after checking for an empty table field.

// get tables entered into the field as a string

String tables = tableNames.getText();

//convert table field string into string array splitting at commas

String[] table = tables.split(",");

//remove trailing white spaces and error check for empty table field

**for** (**int** i = 0;i<table.length;i++)

{

**if**(table[0].equals(""))

{

JOptionPane.*showMessageDialog*(**null**, "No Table(s) Found", "Error", JOptionPane.***ERROR\_MESSAGE***);

JOptionPane.*showMessageDialog*(**null**, "The copy process has been canceled", "Canceled", JOptionPane.***ERROR\_MESSAGE***);

**return**;

}

table[i] = table[i].trim();

System.***out***.println(table[i]);

}

The table array is iterated through and copied one at a time by a for loop that runs through the SQL scripts detailed below. The variables received from the text fields are used to parameterize sqlcmd calls.

// starting with the first table in the string array, use a for loop to run one table at a time through the parameterized copy process

**for** (**int** i = 0;i<table.length;i++){

//initialize different commands to be run

String sourceTableCount\_cmd ="sqlcmd -U "+dUsername+" -P "+dPassword+" -S "+dIP+" -d "+ "MANAGEMENT" +" -i precount.sql -v ip = "+sIP+" tb = "+table[i];

String sourceChecksum\_cmd ="sqlcmd -U "+dUsername+" -P "+dPassword+" -S "+dIP+" -d "+ "MANAGEMENT" +" -i prechecksum.sql -v ip = "+sIP+" tb = "+table[i];

String tableCopy\_cmd ="sqlcmd -U "+dUsername+" -P "+dPassword+" -S "+dIP+" -d "+ "MANAGEMENT" +" -i tablecopy.sql -v ip = "+sIP+" tb = "+table[i];

String destTableCount\_cmd ="sqlcmd -U "+dUsername+" -P "+dPassword+" -S "+dIP+" -d "+ "MANAGEMENT" +" -i postcount.sql -v tb = "+table[i];

String destChecksum\_cmd ="sqlcmd -U "+dUsername+" -P "+dPassword+" -S "+dIP+" -d "+ "MANAGEMENT" +" -i postchecksum.sql -v tb = "+table[i];

String copyStatusUpdate\_cmd ="sqlcmd -U "+dUsername+" -P "+dPassword+" -S "+dIP+" -d "+ "MANAGEMENT" +" -i copyStatusUpdate.sql -v tb = "+table[i];

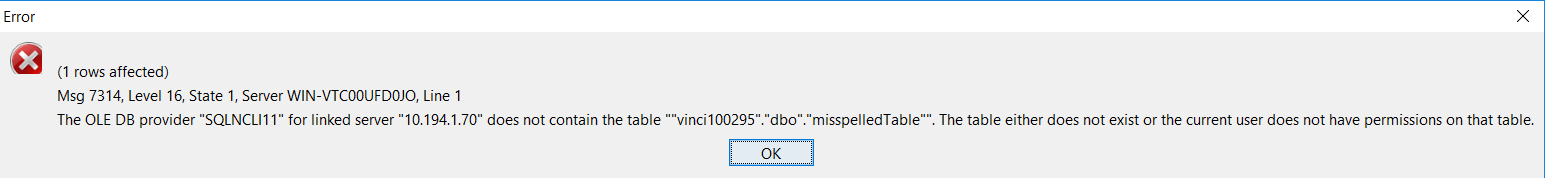
String managementInsert\_cmd ="sqlcmd -U "+dUsername+" -P "+dPassword+" -S "+dIP+" -d "+ "MANAGEMENT" +" -i managementInsert.sql -v tb = "+table[i];

String precopyStatusUpdate\_cmd ="sqlcmd -U "+dUsername+" -P "+dPassword+" -S "+dIP+" -d "+ "MANAGEMENT" +" -i beforeCopyError.sql";

String postcopyStatusUpdate\_cmd ="sqlcmd -U "+dUsername+" -P "+dPassword+" -S "+dIP+" -d "+ "MANAGEMENT" +" -i afterCopyError.sq";

String dropCorrupted\_cmd ="sqlcmd -U "+dUsername+" -P "+dPassword+" -S "+dIP+" -d "+ "MANAGEMENT" +" -i dropCorrupted.sql -v tb = "+table[i];

The calls are made from the Java Runtime class and stored in a process to read the logs from. If the process causes sqlcmd to output a message( which would only happen in this case if an error had occurred, denoted by the Msg keyword), the process is stopped and the prototype is returned to the state before clicking the “Copy Table" button.



Errors are checked for in every call to sqlcmd. Below is an example of how all SQL scripts are called.

//insert the table name into the management table to create a row for logging while checking for errors thrown by sqlcmd

System.***out***.println("insert");

//call to runtime with parameterized sqlcmd call

Process process = Runtime.*getRuntime*().exec(managementInsert\_cmd);

**int** status = process.waitFor();

System.***out***.println(status);

//read the logs into a string to look for errors that occurred during the call

BufferedReader input = **new** BufferedReader(**new** InputStreamReader(process.getInputStream()));

**while** ((line = input.readLine()) != **null**)

{ inputLog=inputLog+line+"\n";

}

input.close();

System.***out***.println(inputLog);

//in case of errors, display message and return out of the for loop

**if**(inputLog.toLowerCase().contains("Msg".toLowerCase()))

{

JOptionPane.*showMessageDialog*(**null**, inputLog, "Error", JOptionPane.***ERROR\_MESSAGE***);

JOptionPane.*showMessageDialog*(**null**, "The copy process has been canceled", "Canceled", JOptionPane.***ERROR\_MESSAGE***);

process = Runtime.*getRuntime*().exec(precopyStatusUpdate\_cmd);

status = process.waitFor();

**return**;

}

System.***out***.println(inputLog);

//also check for less common cmd errors mostly for testing

BufferedReader error = **new** BufferedReader(**new** InputStreamReader(process.getErrorStream()));

**while** ((line = error.readLine()) != **null**)

{

errorLog=errorLog+line+"\n";

}

error.close();

System.***out***.println(errorLog);

**if**(errorLog.toLowerCase().contains("error".toLowerCase()))

{

JOptionPane.*showMessageDialog*(**null**, errorLog, "Error", JOptionPane.***ERROR\_MESSAGE***);

JOptionPane.*showMessageDialog*(**null**, "The copy process has been canceled","Canceled", JOptionPane.***ERROR\_MESSAGE***);

process = Runtime.*getRuntime*().exec(precopyStatusUpdate\_cmd);

status = process.waitFor();

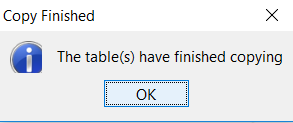
**return**;

}

After running through all the SQL script calls, and there are no errors the user is notified that the copy of that table has successfully been copied from the source server to the destination server.



If there are more tables in the array it will start the for-loop again, if there is not then the user is notified that all their tables have been successfully copied.



Below are the SQL scripts called in the for-loop for table copy.

SQL server scripts

• Inserting a column in a management table for new entry:

$tb = Table name

INSERT INTO [MANAGEMENT].[dbo].[Management\_table] (ReqID,DatabaseName, Table\_Name, dateandtime) VALUES ((SELECT (max(ReqID)+1) from [MANAGEMENT].[dbo].[Management\_table]), 'vinci100295', '$(tb)', CURRENT\_TIMESTAMP );

• Number of rows before copy:

$ip: Server name

$tb: Table name

update [MANAGEMENT].[dbo].[Management\_table] set rowcount\_before\_copy= (select count(\*) from [$(ip)].[vinci100295].[dbo].[$(tb)]) where ReqID = (select MAX(ReqID) from [MANAGEMENT].[dbo].[Management\_table])

• Number of rows after copy:

$tb = Table name

update [MANAGEMENT].[dbo].[Management\_table] set rowcount\_after\_copy= (select count(\*) from [vinci100295].[dbo].[$(tb)]) where ReqID = (select MAX(ReqID) from [MANAGEMENT].[dbo].[Management\_table])

• Checksum Before copy:

$tb = Table name

update Management\_table set checksum\_before\_copy = (select SUM( CAST( checksum(\*) AS BIGINT )) from [$(ip)].[vinci100295].[dbo].[$(tb)]) where ReqID = (select MAX(ReqID) from [MANAGEMENT].[dbo].[Management\_table])

• Checksum after copy:

$tb = Table name

update Management\_table set checksum\_after\_copy = (select SUM( CAST( checksum(\*) AS BIGINT )) from [vinci100295].[dbo].[$(tb)]) where ReqID = (select MAX(ReqID) from [MANAGEMENT].[dbo].[Management\_table])

• Status before copy if process completed:

update [MANAGEMENT].[dbo].[Management\_table] set status\_before\_copy = 'Completed' where ReqID = (select MAX(ReqID) from [MANAGEMENT].[dbo].[Management\_table])

• Status before copy if process failed:

update [MANAGEMENT].[dbo].[Management\_table] SET status\_before\_copy = 'Copy Failed'

WHERE ReqID = (select MAX(ReqID) from [MANAGEMENT].[dbo].[Management\_table])

• Table copy:

$ip: Server Name

$tb = Table name

SELECT \* INTO [vinci100295].[dbo].[$(tb)]

FROM [$(ip)].[vinci100295].[dbo].[$(tb)]

• Status after copy if process completed:

update [MANAGEMENT].[dbo].[Management\_table] set status\_after\_copy = 'Completed' where ReqID = (select MAX(ReqID) from [MANAGEMENT].[dbo].[Management\_table]);

• Status after copy if process failed:

update [MANAGEMENT].[dbo].[Management\_table] SET status\_after\_copy = 'Copy Failed'

WHERE ReqID = (select MAX(ReqID) from [MANAGEMENT].[dbo].[Management\_table])

• Copy Status to check copy successful or copy corrupted

UPDATE [MANAGEMENT].[dbo].[Management\_table]

SET CopyStatus =

CASE

(select checksum\_before\_copy - checksum\_after\_copy

from [MANAGEMENT].[dbo].[Management\_table]

where ReqID = (select MAX(ReqID) from [MANAGEMENT].[dbo].[Management\_table]))

when 0 THEN 'Copy Successful'

ELSE 'Copy Corrupted'

END

WHERE ReqID = (select MAX(ReqID) from [MANAGEMENT].[dbo].[Management\_table])

• Drop table if table copy is corrupted:

$tb = Table name

IF EXISTS(SELECT Table\_Name FROM [MANAGEMENT].[dbo].[Management\_table] where CopyStatus = 'Copy Corrupted' and ReqID = (select max(ReqID) from [MANAGEMENT].[dbo].[Management\_table]))

drop table [vinci100295].[dbo].[$(tb)]

• Copy Status if copy successful

update [MANAGEMENT].[dbo].[Management\_table] set CopyStatus = 'Copy Successful' where ReqID = (select MAX(ReqID) from [MANAGEMENT].[dbo].[Management\_table])