# DBAS 4002 - Assignment 3

# Record Logging & Insert/Delete Stored Procs

Assignment Value: *12*% of overall course mark.

Due Date: **See due date designated on the Assignment 3 dropbox on Brightspace.**

Late submissions will receive the standard late submission penalty as stated in the course outline.

#### Required Databases:

* Chinook

#### Assignment Instructions:

**Restore a new copy of the Chinook database, with a Database Name of Chinook\_A3.**

Create a single .sql script file containing all SQL statements required to meet all requirements listed below.

#### Submissions:

When you are finished, upload your script to Brightspace as your submission for Assignment 3, using a file name similar to: **[YourName]\_DBAS\_Assignment3.sql**.

#### Evaluation:

To insure the greatest chance of success on this assignment, be sure to check the marking rubric contained at the end of this document or in Brightspace. The rubric contains the criteria your instructor will be assessing when marking your assignment.

## Assignment Scenario

The Invoice Payments enhancements you made for The Chinook Company’s database system were a great success. Not only has the enhancement greatly improved their financial accounting system, they have noticed the database’s performance has increased when adding new Invoices and Payments. (Thank you, Stored Procedures!)

The company has realized they make better, more efficient use of their database if they convert more of its commonly-used functionality over to using more stored procedures. They’re also interested in tracking when records are added or removed from certain tables. They’ve hired you back on to tackle a new enhancement.

Being a music-based company, the data their employees have to administer the most are the **Tracks, Artists, Albums,** as well as **Genres** and **Mediatypes**. For each of these five tables, they have requested you create separate stored procedures for adding new records or deleting existing ones.

They want to log whenever a record is added or deleted, so they have asked you to create a new RecordLogging table, which will be used in conjunction with the new add/delete procedures, to log any changes to records in the desired tables.

## Requirements

Your solution should add the following to the existing Chinook database:

1. A new table called **RecordLogging** will be added. **See ERD below**.
2. A new stored procedure called **uspAddRecordLog** will be created, to add logging records and track data changes. The data they would like to track for each record changed is:
   1. The affected table’s name
   2. The record’s ID, if applicable. This would typically be the Primary Key for a newly-inserted record, or -1 for other cases.
   3. The action type of the change, ie. INSERT or DELETE.
   4. When logging an error, the SQL Server error number should be recorded, and the IsError flag set to True. For records that are not errors, set the error number to zero.
   5. Every log record, of any type, must be automatically time-stamped when it is added to the Logging table.
3. For each of the five tables (**Tracks, Artists, Albums, Genres** and **Mediatypes**), add a new stored procedure called **usp*<TableName>\_*Insert**. These stored procedures should:
   1. Accept values for all applicable fields, of the appropriate datatypes and sizes.
   2. Parameters should be nullable where appropriate. See each table’s constraints for details.
   3. Have an output parameter, which will return the ID of the newly-created record.

**Hint:** Research the SCOPE\_IDENTITY() function.

* 1. Contain a transaction and error handling, so that should the insert statement fail, it will be reverted. Otherwise, it will be committed.
  2. If the insert action is successful, a call to the **uspAddRecordLog** proc will be used to create a logging record. See screenshots below for details that should be captured for a successful insert.
  3. If the insert action is NOT successful, a different call to the **uspAddRecordLog** proc will be used to create a logging record to save basic error details. See screenshots below for details that should be captured for a failed insert.

1. For each of the five tables (**Tracks, Artists, Albums, Genres** and **Mediatypes**), add a new stored procedure called **usp*<TableName>\_*DeleteByID**. These stored procedures should:
   1. Accept the PK ID for the table as a parameter.
   2. Use the incoming ID value to attempt to delete the record associated with that ID.
   3. Contain a transaction and error handling, so that should the delete statement fail, it will be reverted. Otherwise, it will be committed.
   4. If the delete action is successful, a call to the **uspAddRecordLog** proc will be used to create a logging record. See screenshots below for details that should be captured for a successful delete.
   5. If the delete action is NOT successful, a different call to the **uspAddRecordLog** proc will be used to create a logging record to save basic error details. See screenshots below for details that should be captured for a failed delete.

## Executing Procs & Testing Statements

To ensure the accuracy of your work, your script should include executing calls to the new stored procedures, in the proper order, and include any supporting statements that, when run together, ensure all work is completed to the requirements. Note that during tests that add new test records, hard-coded values are acceptable.

Your testing script should include:

* A call to each table’s Insert and Delete stored procedures, using VALID data (ie. That should not causes errors). The first call should be to the table’s Insert proc, to create a new test record. The second call should be to the table’s Delete proc, and should remove the test record created in the previous test. Note that the success/failure should be logged in the Logging table.
* Calls to the Insert procs for Album and Track, with INVALID data (ie. Should cause an error). These will be used to verify that error records are correctly created in the Logging table.

## ERD for Proposed Modifications

#### Insert/Delete Procs & Logging Enhancement

Figure 1: New RecordLogging table

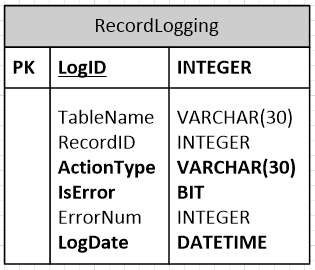
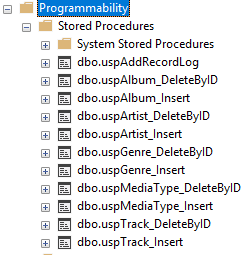


Figure 2: List of New Procs



## Screenshots

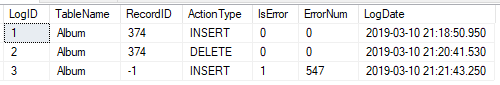
RecordLogging table after a successful insert into the Album table, using the **uspAlbum\_Insert** proc.



RecordLogging table after successfully deleting that same test record using the **uspAlbum\_DeleteByID** proc.



RecordLogging table after a **failed** insert into the Album table, using the **uspAlbum\_Insert** proc. Note: Error 547 is a Foreign Key Constraint violation.



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Chinook – Logging & Add/Delete Procs** | | | | | Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |  |
| **Criteria** | | **Insufficient (0 pts)** | **Needs Development**  **(1-2 pts)** | **Sufficient (3-4 pts)** | **Excellent (5 pts)** |  | **X** |
| **DDL for RecordLogging** | | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | New RecordLogging table successfully created and matches all requirements for fields, constraints and auto-numbering. |  |  |
| **uspAddRecordLog** | | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | **uspAddNewInvoice** works as expected to add new RecordLogging records. All required parameters are included, have the correct datatype, correct defaults where appropriate, and are used correctly in the procedure. LogDate is system-generated, either through the proc or a table constraint. |  |  |
| **Successes vs. Errors** | | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | All successful insert/delete actions are logged correctly, with all the expected data recorded. All failed actions (ie. Errors) are logged correctly, with all expected data recorded. |  |  |
| **Insert Procedures** | | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | All five **usp*<Tablename>*\_Insert** procedures work as expected to add new records to the appropriate tables. All required input parameters are included, have the correct datatype, correct defaults where appropriate, and are used correctly in the procedure. Output parameter to return the new record ID is present and works as expected. Procedure contains calls to logging stored proc, for both successful and error scenarios. |  | 2 |
| **Delete Procedures** | | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | All five **usp*<Tablename>*\_DeleteByID** procedures work as expected to delete records from the appropriate tables. All required input parameters are included, have the correct datatype, correct defaults where appropriate, and are used correctly in the procedure. Procedure contains calls to logging stored proc, for both successful and error scenarios. |  | 2 |
| **Executing & Testing Script** | | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | Statements are included to correctly execute all required stored procedures, in the proper order, to test each stored procedure with valid test data. Required invalid tests are included and work as expected. |  |  |
| **Transactions Use** | | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | Use of transactions shows a strong understanding of the concept, and transactions are correctly applied in all reasonably appropriate places in the procedures. |  |  |
| **Error Handling** | | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | Use of error handling shows a strong understanding of the concept, and error handling techniques are correctly applied in all reasonably appropriate places in the procedures. |  |  |
| **Script Usability** | | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | Script demonstrates proper batching (GO) whenever needed. Script runs without errors. |  |  |
| **Comments &**  **Best Practices**  (Must complete at least 60% of functional reqs) | | Little to no effort was made or contains too many errors / omissions. | A reasonable effort was made, but there are multiple areas for improvement. | A good effort was made, but at least one error or omission exists. | Organizational or explanatory comments are used extensively, most are meaningful and easily understood. A consistent naming convention was used for most of the program and deviated very little. Source code was clean, consistently well-formatted and easy to read. |  |  |
|  |  | |  |  | **Total:** |  | **/60** |