Objetivo: Crear un esquema genérico de auditoría para auditar los inserts updates y deletes sobre tablas.

1. Crear la siguiente tabla de auditoría:

NombreTabla: auditoria

Campos:  
 auditID ENTERO PK (Crear sequence/SERIAL/IDENTITY/AUTOINCREMENT) nombreTabla VARCHAR(30) NOT NULL  
 operacion CHAR (VALIDAR (I: Inserted, O: Old Updated,N:New Updated,D: Deleted))

rowData VARCHAR(255) NOT NULL

Usuario VARCHAR(30) DEFAULT USER (ver según motor)

fecha DATETIME DEFAULT CURRENT (ver según motor)

create table auditoria(

auditID int identity (1,1) primary key,

nombreTabla varchar(30) not null,

operacion char(1) check (operacion in ('I','O','N','D')),

rowData varchar(255) not null,

usuario varchar(30) default suser\_sname(),

fecha datetime default getDate())

1. Crear un procedure altaAuditoria que inserte en la tabla auditoría una fila con los parámetros de entrada enviados.

Parametros de entrada: nombreTabla , operacion, rowData

Especificación del cuerpo del procedure

Debe insertar en la tabla auditoría una fila con los datos recibidos.

Que sucede con el campo usuario y fecha de la tabla auditoría si no los incluyen en el insert?

CREATE PROCEDURE altaAuditoria @nombreTabla varchar(30), @operacion char(1) , @rowData varchar(255)

AS

BEGIN

INSERT INTO auditoria(nombreTabla, operacion,rowData)

VALUES (@nombreTabla,@operacion,@rowData)

END

1. Crear un trigger de insert sobre la tabla manufact con las siguientes características:

FOREACH ROW (TENER EN CUENTA RESOLUCIÓN EN SQLSERVER)

Ejecuta el Procedure altaAuditoria pasándole los parámetros necesarios a partir de los datos INSERTADOS en la tabla.

El valor NOMBRETABLA deberá ser “manufact”

El valor operacion deberá ser “I”(i mayúscula)

El campo rowData lo debe componer concatenando los valores INSERTADOS en la tabla separados por “|” cada valor.

Ej: “ANZ”|”Adfasdflsflds”|”12”|”CA”

CREATE TRIGGER ins\_manufact on manufact

AFTER INSERT

AS

DECLARE

@manu\_code char(3),

@manu\_name varchar(15),

@lead\_time smallint,

@state CHAR(2),

@rowData varchar(255)

BEGIN

DECLARE curInsertados CURSOR FOR

SELECT manu\_code, manu\_name, lead\_time, state

FROM inserted

OPEN curInsertados

FETCH NEXT FROM curInsertados

INTO @manu\_code, @manu\_name, @lead\_time, @state

WHILE @@fetch\_status=0

BEGIN

SET @rowData= @manu\_code + ' | '+ @manu\_name + ' | '+ cast (@lead\_time as nvarchar)+' | '+@state

EXEC altaAuditoria 'manufact', 'I', @rowData

FETCH NEXT FROM curInsertados

INTO @manu\_code, @manu\_name, @lead\_time, @state

END

CLOSE curInsertados

DEALLOCATE curInsertados

END

1. Crear un trigger de DELETE sobre la tabla manufact con las siguientes características:

FOREACH ROW (TENER EN CUENTA RESOLUCIÓN EN SQLSERVER)

Ejecuta el procedure altaAuditoria pasándole los parámetros necesarios a partir de los datos de la fila/s BORRADAS en la tabla.

El valor nombreTabla deberá ser “manufact”

El valor operacion deberá ser “D”

El campo rowData lo debe componer concatenando los valores de la fila BORRADA de la tabla origen y separándolo por “|” cada valor.

Ej: “ANZ”|”Adfasdflsflds”|”12” |”CA”

CREATE TRIGGER del\_manufact ON manufact

AFTER DELETE

AS

DECLARE

@manu\_code char(3),

@manu\_name varchar(15),

@lead\_time smallint,

@state char(2),

@rowData varchar(255)

BEGIN

DECLARE curBorrados CURSOR FOR

SELECT manu\_code, manu\_name, lead\_time, state

FROM deleted

OPEN curBorrados

FETCH NEXT FROM curBorrados INTO @manu\_code, @manu\_name, @lead\_time,@state

WHILE @@fetch\_status=0

BEGIN

SET @rowData= @manu\_code + ' | '+ @manu\_name + ' | '+ cast (@lead\_time as nvarchar)+' | '+@state

EXEC altaAuditoria 'manufact', 'D', @rowData

FETCH NEXT FROM curBorrados

INTO @manu\_code, @manu\_name, @lead\_time,@state

END

CLOSE curBorrados

DEALLOCATE curBorrados

END

1. Crear un trigger de UPDATE sobre la tabla manufact con las siguientes características:

FOREACH ROW (TENER EN CUENTA RESOLUCIÓN EN SQLSERVER)

Ejecutar dos veces el procedure altaAuditoria pasándole los parámetros necesarios a partir de los datos INSERTADOS en la tabla.

El valor NOMBRETABLA deberá ser “manufact”

El valor operacion deberá ser “O” (Old Fila vieja antes de la modificación)

El campo rowData lo debe componer concatenando los valores de la fila ORIGINAL anterior a la MODIFICACION de la tabla origen y separándolo por “|” cada valor.

Ej: “ANZ”|”Adfasdflsflds”|”12” |”CA”

El valor NOMBRETABLA deberá ser “MANUFACT”

El valor operacion deberá ser “N” (NEW Fila nueva producto de la modificación)

El campo rowData lo debe componer concatenando los valores de la fila NUEVA posterior a la MODIFICACION de la tabla origen y separándolo por “|” cada valor.

Ej: “ANZ”|”Adfasdflsflds”|”12”

CREATE OR ALTER TRIGGER upd\_manufact ON manufact

AFTER update

AS

DECLARE

@rowDataB varchar(255),

@rowDataI varchar(255),

@manu\_codeB char(3),

@manu\_nameB varchar(15),

@lead\_timeB smallint,

@stateB char(2),

@manu\_codeI char(3),

@manu\_nameI varchar(15),

@lead\_timeI smallint,

@stateI char(2)

BEGIN

DECLARE curBorrados CURSOR FOR SELECT manu\_code, manu\_name, lead\_time, state FROM deleted

DECLARE curInsertados CURSOR FOR SELECT manu\_code, manu\_name, lead\_time, state FROM inserted

OPEN curBorrados

OPEN curInsertados

FETCH NEXT FROM curBorrados INTO @manu\_codeB, @manu\_nameB, @lead\_timeB, @stateB

FETCH NEXT FROM curInsertados INTO @manu\_codeI, @manu\_nameI, @lead\_timeI, @stateI

while @@fetch\_status=0

begin

set @rowDataB= @manu\_codeB + ' | '+ @manu\_nameB + ' | '+ cast (@lead\_timeB as nvarchar)

exec altaAuditoria 'manufact', 'O', @rowDataB

set @rowDataI= @manu\_codeI + ' | '+ @manu\_nameI + ' | '+ cast (@lead\_timeI as nvarchar)

exec altaAuditoria 'manufact', 'N', @rowDataI

FETCH NEXT FROM curBorrados INTO @manu\_codeB, @manu\_nameB, @lead\_timeB, @stateB

FETCH NEXT FROM curInsertados INTO @manu\_codeI, @manu\_nameI, @lead\_timeI, @stateI

end

CLOSE curBorrados

DEALLOCATE curBorrados

CLOSE curInsertados

DEALLOCATE curInsertados

END

1. Realizar las siguientes pruebas
   1. INSERTAR el siguiente registro “XXX”,”Xtra large”,”23”
   2. UPDATE el reg donde manu\_code=”XXX” cambiando la descripción por “Extra Large”

Que valores contiene la tabla AUDITORIA?

* 1. INSERTAR NUEVAMENTE el siguiente registro “XXX”,”Xtra large”,”23”

Que sucedió?

* 1. INSERTAR el siguiente registro “ZZZ”,”Zampini SA”,”11”
  2. BORRAR el registro cuyo manu\_code=”ZZZ”

Que nuevos registros contiene la tabla AUDITORIA?

INSERT INTO manufact (manu\_code,manu\_name,lead\_time,state)

VALUES ('XXX', 'Xtra large', 23,'CA')

UPDATE manufact

SET manu\_name = 'Extra Large'

WHERE manu\_code='XXX'

INSERT INTO manufact (manu\_code,manu\_name,lead\_time,state)

VALUES ('XCX', 'Xtra large', 23,'CA')

DELETE FROM manufact WHERE manu\_code ='XCX'

SELECT \* FROM auditoria

1. Crear la siguiente tabla de log de errores:

NombreTabla: errorAudit

Campos:

errorID ENTERO PK (Crear sequence o SERIAL, IDENTITY o

AUTOINCREMENT)

SqlError INTEGER

isamError INTEGER

errorInfo CHAR(70)

nombreTabla VARCHAR(30) NOT NULL

operacion CHAR

rowData VARCHAR(255) NOT NULL

usuario VARCHAR(30) DEFAULT USER

fecha DATETIME YEAR TO FRACTION DEFAULT CURRENT

errorStatus CHAR (Validar P:Pendiente, F: Finalizado) DEFAULT “P”

drop table errorAudit;

CREATE TABLE errorAudit(

errorID int IDENTITY(1,1),

ErrorNum int,

ErrorLine int,

ErrorInfo varchar(1000),

nombreTabla varchar(255) not null,

operacion char(1),

rowData varchar(255),

usuario varchar(30) DEFAULT suser\_sname(),

fecha datetime DEFAULT getDate(),

errorStatus char CHECK(errorStatus in('P','F')) DEFAULT 'P')

1. Modificar el procedure altaAuditoria o los triggers (dependiendo del motor) agregando un bloque de manejo de excepciones que ante cualquier error que ocurra en la operación de auditoría, grabe una fila en la tabla errorAudit ingresando los datos asociados al error y los datos de la fila fallida en la inserción de la tabla AUDITORÍA.

El campo errorStatus no se incluye en el INSERT y se actualizará por DEFAULT en “P”.

OPCION 1 creo procedure errorAuditoria y modifico trigger con TRY y CATCH

CREATE PROCEDURE errorAuditoria

@nombreTabla varchar(30), @operacion char(1) , @rowData varchar(255)

AS

BEGIN

INSERT INTO errorAudit (ErrorNumber,ErrorLine,errorInfo,

nombreTabla,operacion,rowData)

VALUES (error\_number(),error\_line(),error\_message(),

@nombreTabla, @operacion, @rowData)

END

ALTER TRIGGER ins\_manufact on manufact

AFTER INSERT

AS

DECLARE

@manu\_code char(3),

@manu\_name varchar(15),

@lead\_time smallint,

@state CHAR(2),

@rowData varchar(255)

BEGIN

DECLARE curInsertados CURSOR FOR

SELECT manu\_code, manu\_name, lead\_time, state

FROM inserted

OPEN curInsertados

FETCH NEXT FROM curInsertados

INTO @manu\_code, @manu\_name, @lead\_time, @state

WHILE @@fetch\_status=0

BEGIN

SET @rowData= @manu\_code + ' | '+ @manu\_name + ' | '+ cast (@lead\_time as nvarchar)+' | '+@state

BEGIN TRY

EXEC altaAuditoria 'manufact', 'I', @rowData

END TRY

BEGIN CATCH

EXEC errorAuditoria 'manufact', 'I', @rowData

END CATCH

FETCH NEXT FROM curInsertados

INTO @manu\_code, @manu\_name, @lead\_time, @state

END

CLOSE curInsertados

DEALLOCATE curInsertados

END

Opcion 2

Modifico sp altaAuditoria con un TRY y CATCH. En otros motores me anda perfecto

ALTER PROCEDURE altaAuditoria

@nombreTabla varchar(30), @operacion char(1) , @rowData varchar(255)

AS

BEGIN TRY

INSERT INTO auditoria(nombreTabla, operacion,rowData)

VALUES (@nombreTabla,@operacion,@rowData)

END TRY

BEGIN CATCH

INSERT INTO errorAudit (ErrorNum,ErrorLine,errorInfo,

nombreTabla,operacion,rowData)

VALUES (error\_number(),error\_line(),error\_message(),@nombreTabla,

@operacion, @rowData)

END CATCH

ALTER PROCEDURE altaAuditoria – este funciona

@nombreTabla varchar(30), @operacion char(1) , @rowData varchar(255)

AS

begin

declare @xactAbortAux BIT = 0, @options INTEGER;

-- PRINT 'Obtengo los valores de sesion seteados'

SELECT @options = @@OPTIONS;

-- PRINT 'Me fijo con que valor vino la var de sesion XACT\_ABORT'

IF ( (16384 & @options) = 16384 ) -- si vino en ON

Set @xactAbortAux = 1; -- guardo el valor en una aux

-- PRINT 'Seteo XACT\_ABORT en OFF'

SET XACT\_ABORT OFF;

BEGIN TRY

INSERT INTO auditoria(nombreTabla, operacion,rowData)

VALUES (@nombreTabla,@operacion,@rowData)

END TRY

BEGIN CATCH

print 'entro al catch del Alta auditoria'

INSERT INTO errorAudit (ErrorNum,ErrorLine,errorInfo,

nombreTabla,operacion,rowData)

VALUES (error\_number(),error\_line(),error\_message(),@nombreTabla,

@operacion, @rowData)

END CATCH

-- print ('Despues del CATCH');

If @xactAbortAux = 1

begin

-- print 'Vuelvo la variable XACT\_ABORT a su valor original

SET XACT\_ABORT ON

end

end

--Prueba

-- Creo table auditoria con un constraint de chequeo inválido para que falle la inserción en dicha tabla.

drop table auditoria

create table auditoria(

auditID int identity (1,1) primary key,

nombreTabla varchar(30) not null,

operacion char(1) check (operacion in ('X')),

rowData varchar(255) not null,

usuario varchar(30) default suser\_sname(),

fecha datetime default getDate())

INSERT INTO manufact (manu\_code,manu\_name,lead\_time,state)

VALUES ('yyy', 'Xtra large', 23,'CA')

select \* from errorAudit

1. Realizar las siguientes pruebas
   1. Borre y recree la secuencia sq\_pk\_auditoria comenzando con el anteúltimo número existente en la tabla auditoría campo auditID
   2. Realizar los siguientes INSERTs del siguiente

‘WWW’,’Wahington WWW’,12

‘TTT’,’Tamina SA’,12

Falló la transacción original?

Se grabó la información en la tabla auditoría?

Que sucedió?

* 1. Ejecutar un DELETE completo de la tabla MANUFACT sin cláusula WHERE.

Cuál fue el resultado del mismo?

Que información se grabó en la tabla AUDITORÍA?

Se grabaron errores en la tabla errorAudit? Por qué?

Falló la transacción original?

Se grabó la información en la tabla auditoría?

Que sucedió?

* 1. Ejecutar un DELETE completo de la tabla MANUFACT sin cláusula WHERE.

Cuál fue el resultado del mismo?

Que información se grabó en la tabla AUDITORÍA?

Se grabaron errores en la tabla errorAudit? Por qué?

1. Realizar un procedimiento REPROCESA\_DATA

Objetivo: Actualizar la tabla AUDITORIA con todas las filas de auditoría que no pudieron ser insertadas, que se grabaron en la tabla errorAudit y que su estado sea Pendiente (errorStatus=P).

Por cada fila de la tabla errorAudit cuyo campo ERRSTATUS sea igual a P se deberán realizar las siguientes operaciones:

* INSERTAR en la tabla AUDITORIA los datos de la fila original.
* MODIFICAR el campo ERRSTATUS de la fila procesada con el valor “F”

Tener en cuenta:

* Que no se pueda modificar en forma concurrente por otro usuario o proceso la fila que se está tratando en este PROCEDURE.
* El proceso completo de reproceso debe tomarse como un bloque, o se procesan todas las filas o no se procesa ninguna.

1. Realizar las siguientes pruebas
   1. Consultar de la tabla errorAudit todas las filas cuyo estado sea Pendiente de reprocesar.

Cuantas filas son?

* 1. Ejecutar el procedimiento reprocesaAltaAuditoria

Evaluar las tablas auditoria y errorAudit e indique el resultado obtenido.

**Opcional**

1. Crear los triggers de INSERT para las tablas orders e items, tomando como referencia el trigger de la tabla manufact.
2. Realice una prueba de los mismos con la siguiente info.
   1. Inserte en la tabla ORDERS una nueva orden basándose en el SELECT de la tabla ORDERS para los order\_num IN (1001,1002)., tener en cuenta que el campo order\_num debe respetar una secuencia. Verifique los números de orden generados
      1. Crear una sequence sq\_pk\_orders que comience con el nro 2001
      2. La primer Orden deberá ser igual a la 1001 y la siguente a la 1002
   2. Verifique los números de orden generados
   3. Inserte en la tabla ITEMS los ítems para la 1ra orden generada, basándose en los ítems de la Orden 2002.
   4. Inserte en la tabla ITEMS los ítems para la 2da orden generada, basándose en los ítems de la Orden 2002.

Que nuevas filas observa en la tabla AUDITORIA.