UNIVERSIDAD NACIONAL DE LA MATANZA

DEPARTAMENTO DE INGENIERÍA

E INVESTIGACIONES TECNOLÓGICAS

INGENIERIA EN INFORMATICA

**BASE DE DATOS**

Guia Practica de Transacciones

*Jefe de Cátedra: Ing. Verónica Ichazo*

*Docentes a cargo de curso:*

*Ing. Alfonso Palomares*

*Ing. Natalia Crespo*

*Ing. Guillermo Giannotti*

*Docentes a cargo de práctica:*

*Ing. Matías López*

*Ing. Fernando Ybarra*

*Ing. Hernán Jalil*

**2021**

**Laboratorio: Transacciones en MS SQLServer**

Para realizar esta práctica es necesario contar con la base de datos AdventureWorks. En el caso de no poseerla se podrá obtener libremente desde el site de Microsoft ([**Ver Anexo I**](#Anexo))

**1. Visualización de conexiones:**

1.1 Abrir una ventana de ejecución de consultas, la cual llamaremos *SQLQuery1*.

1.2 Ejecutar el siguiente comando para visualizar las conexiones existentes:

sp\_who2 'active'

*Pregunta 1:*

*¿Cuáles son los estados de las sesiones visualizadas?*

*Runnable, Background, sleeping,*

*¿Cuál es la sesión que corresponde a la consulta que estamos ejecutando?*

*52*

*¿Qué datos vemos de cada una de las sesiones?*

*Podemos ver las secciones activas, y tiene un process id para ver las querys que ejecutamos en un det tiempo*

1.3 Ejecutar el siguiente comando para visualizar el identificador único de la sesión conectada en *SQLQuery1*:

select @@SPID

51 (te devuelve el nro de sesión de process ID)

1.4 Abrir una nueva ventana de ejecución de consultas, la cual llamaremos SQLQuery2.

1.5 Ejecutar el siguiente comando para visualizar el identificador único de la sesión conectada en *SQLQuery2*:

select @@SPID

1.6 Crear el siguiente procedimiento en la ventana *SQLQuery2*:

create procedure dbo.p\_test\_i

as

select GETDATE() as inicio

declare @i int

set @i=0

while @i<100000

begin

set @i=@i+1

waitfor delay '0:00:01'

end

select GETDATE() as fin

------------------------------------

Escala de tiempo

Descripción generada automáticamente

1.7 Ejecutar el procedimiento en la ventana *SQLQuery2,* sin esperar una respuesta ya que se quedará en un estado de ejecución:

exec dbo.p\_test\_i

1.8 Situarse en la ventana *SQLQuery1* y Visualizar cómo vemos el *pid* de ejecución en la sesión 2:

sp\_who2 'active'

dbcc inputbuffer(<*nro\_pid>)*

*{nro\_pid:* es el número de pid obtenido que corresponde a la sesión 2.}

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

*‘5 mins-Pregunta 1:*

*¿Qué información nos muestra el comando inputbuffer?*

*Muestra el tipo de evento, los parámetros, y la información que muestra del SP*

*¿Qué informa el comando sp\_who?*

*SE ESTÁ EJECUTANDO UN STORE PROCEDURE DBO.P\_TEST\_I*

*¿Qué dato identifica unívocamente a cada sesión?*

*EL process id para ver las querys que ejecutamos en un det tiempo.*

1.9 Cerrar ambas ventanas para continuar con el resto de los ejercicios *SQLQuery1 y SQLQuery2.* A modo optativo, se deberá Indicar si se quieren grabar o no las sesiones.

**2. Rollback/Commit**

2.1 Abrir una nueva ventana *SQLQuery3.*

2.2 Testear las siguientes ejecuciones dentro de la misma ventana SQLQuery3, utilizando el mecanismo de seleccionar para indicar la sentencia a ejecutar (Primero ejecutar Test1, evaluar resultados luego Test2 y finalmente Test3).

Test 1:

--Test1

begin transaction

create table Person.Test1(campo1 int)

insert into Person.Test1 values(9999)

rollback transaction

select \* from Person.Test1

Test 2:

--Test2

begin transaction

create table Person.Test1(campo1 int)

insert into Person.Test1 values(9999)

commit transaction

select \* from Person.Test1

--Test3

create table Person.Test2(campo1 int)

insert into Person.Test2 values(1234)

select \* from Person.Test2

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Es errónea ya que se deshace la operación. Test1

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamentetest 2 no

*‘5 mins Pregunta 2:*

*¿Qué sucede al realizar el rollback en una sesión con ddl?*

Se deshace la operación. Test1

*¿Qué sucede al realizar un commit con ddl?*

*Se mete el el dato ya que se comitió*

*¿Que característica especial tiene la tabla creada?*

*¿Cuál es el resultado de las ejecuciones?*

2.3 Abrir una nueva ventana *SQLQuery4.*

2.4 Testear las siguientes ejecuciones en la ventana SQLQuery4, de a un bloque de testeo por vez.

Test 1:

--Test1

begin transaction

insert into Person.Test1 values(-3)

insert into Person.Test1 values(-2)

insert into Person.Test1 values(-1)

rollback transaction

select \* from Person.Test1

Test 2:

--Test2

begin transaction

insert into Person.Test1 values(-3)

insert into Person.Test1 values(-2)

insert into Person.Test1 values(-1)

commit transaction

select \* from Person.Test1

*‘5 mins Pregunta 3:*

*¿Qué sucede al realizar el commit y qué sucede al realizar el rollback en la tabla Test1?*

*En el commit estás confirmando la transacción de los valores -3… y cuando tiene que confirmarla, hace un rollback*

*¿Cuál es el resultado de las ejecuciones?*

*Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente con confianza mediaeste no por rollback*

*Tabla

Descripción generada automáticamenteel test 2 confirma los valores sobre la tabla persona*

*¿Siempre deben ser explícitas las transacciones?*

*No, pero es el modo de trabajo recomendado para el tema transacciones. Se hacen implícitamente. Si no ponés el commit, se pone el autocommit.*

*¿Cuál es el comportamiento al utilizar c/u?*

*Básicamente uno hace el rollback, y otro la comitea xd.*

*Utilizar transacciones Explícitas -> try catch*

*Tiene tres transacciones de una sentencia c/u*

2.5 Crear el procedimiento en una nueva ventana, la cual llamaremos SQLQuery5:

create procedure Person.sp\_test1

as

declare @i int

set @i=0

while @i<100

begin

set @i=@i+1

insert into Person.Test1 values(@i)

waitfor delay '00:00:01'

end

2.6 Abrir una nueva ventana *SQLQuery6* y ejecutar el siguiente procedimiento, cancelarlo a los 10 segundos a través del botón de *Stop* (“Cancel Executing Query”):

exec Person.sp\_test1

2.7 En otra ventana llamada *SQLQuery7*, verificar cuántos registros posee la tabla Person.Test1.

Select \* from Person.Test1

*Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente*

*se pusieron 11 por los segundos que pasaron*

*‘5 mins Pregunta 4:*

*¿Cómo se inicia y se confirma la transacción en un procedimiento almacenado?*

*Porque se paró la ejecución del procedure. Esos inserts son transacciones implícitas que se van confirmados en los 10 seg. 1 por segundo*

2.8 En la ventana en la cual creamos el procedimiento *SQLQuery5*, cambiar el procedimiento de la siguiente forma. Recordar que se debe ejecutar el alter para hacer efectivo el cambio.

alter procedure Person.sp\_test1

as

begin transaction

begin try

declare @i int

set @i=0

while @i<100

begin

set @i=@i+1

insert into Person.Test1 values(@i)

waitfor delay '00:00:01'

end

commit transaction

end try

begin catch

rollback transaction

end catch

2.9 Ejecutar nuevamente procedimiento en la ventana llamada SQLQuery6, cancelarlo a los 10 segundos a través del botón de stop:

select count(\*) as cant from Person.Test1

go

exec Person.sp\_test1

go

2.10 En otra verificar cuántos registros resultantes que posee la tabla Person.Test1.

select \* from Person.Test1

*Anotación: Cantidad de Registros= 25*

*Está esperando a que termine la otra transacción. Igual lo cancelé en el medio, no llegó a todo el commit*

*Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente*

*Los go hacen dividir en bloques. Pero ejecutan* *Interfaz de usuario gráfica, Tabla

Descripción generada automáticamente*

*‘5 mins Pregunta 5:*

*¿Cómo se controlan las transacciones dentro de un procedimiento almacenado?*

*¿Cómo puedo saber las transacciones abiertas?*

2.11 En otra ventana verificar las sesiones activas y las totales.

sp\_who2 'active' --Sesiones activas

go

sp\_who2 --Todas las sesiones

go

2.12 Cerrar la ventana SQLQuery6 (posee la ejecución del sp\_test1) y volver a verificar la cantidad de registros que posee la tabla Person.Test1 en otra ventana.

select \* from Person.Test1

*Anotación: Cantidad de Registros= ?*

*‘5 mins Pregunta 7:*

*¿Cómo eliminamos una sesión?*

*Con un kill no se qué*

*¿Qué sucede cuando se cancela manualmente una sesión que está en ejecución?*

*Te pregunta si querés commitear los cambios, sino se hace rollback*

2.13 Cerrar todas las ventanas de consultas existentes.

**3. Deadlock: yo te libero cuando el otro me libere aquel. Se queda en un estado indefinido. SQL MATA ALEATORIAMENTE A ALGUNA DE LAS TRANSACCIONES**

3.1 Abrir una nueva ventana de ejecución.

3.2 Borrar las tablas Person.Test1 y Person.Test2 y recrearlas con los siguientes registros:

Drop table Person.Test1,Person.Test2

create table Person.Test1 (campo1 int)

create table Person.Test2 (campo2 int)

insert into Person.Test1 values (999)

insert into Person.Test2 values (999)

select \* from Person.Test1

select \* from Person.Test2

3.3 Ejecutar las siguientes sentencias en “dos” ventanas diferentes, primero ejecutar Ventana1 y luego ejecutar Ventana2. Ejecutar ambas al mismo tiempo, ya que de lo contrario, no sucederá la excepción.

Ventana1:

BEGIN TRAN

UPDATE Person.Test1 SET campo1= 1000 where campo1=999

waitfor delay '00:00:10'

SELECT \* FROM Person.Test2

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Ventana2:

BEGIN TRAN

UPDATE Person.Test2 SET campo2= 1000 where campo2=999

waitfor delay '00:00:10'

SELECT \* FROM Person.Test1

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

*Pregunta 8*:

*¿Por qué circunstancias el motor puede cancelar una sesión que se encuentra en ejecución?*

3.4 Obtener el número de pid de cada una de las ventanas en las cuales realizaron su ejecución.

3.5 Volver a ejecutar los procesos de cada ventana de a uno por vez y verificar los lockeos que cada uno necesita.

sp\_lock <ingresar el número de pid>

ó

sp\_lock

*Pregunta 9:*

*¿Qué tipos de lockeos puede generar una sesión en una base de datos?*

*TIPOS:LOCK SHARED Y SEXCLUSIVE. HABÍA UNA TABLA, TAMBIÉN RESPETARLO*

3.6 Cerrar todas las ventanas de consultas existentes.

**4. Niveles de Aislamiento – VER APUNTE DE NIVEL DE AISLAMIENTO -> HACERLO, MUY IMPORTANTE**

4.1 Ejecutar el siguiente comando para verificar qué nivel de Isolation se está utilizando en la sesión activa.

DBCC USEROPTIONS

*Pregunta 10*:

¿Cuáles son los niveles de aislamiento que puede tomar una sesión en una base de datos?

¿Cuál es el nivel de aislamiento que utiliza por defecto MSSqlserver?

4.2 READ COMMITTED: Ejecutar las siguientes sentencias en dos ventanas diferentes. En una de ellas utilizamos la actualización de una persona y en la otra ventana realizamos pruebas de cómo visualizar esta modificación.

Ventana1:

SET TRANSACTION ISOLATION LEVEL READ COMMITTED

BEGIN TRAN

SELECT \*

FROM [Person].[Person]

WHERE FirstName='Terri' and LastName='Duffy'

UPDATE [Person].[Person]

SET MiddleName ='NN'

WHERE FirstName='Terri' and LastName='Duffy'

SELECT \*

FROM [Person].[Person]

WHERE FirstName='Terri' and LastName='Duffy'

*Pregunta 11:*

*¿Qué valores quedaron para la columna MiddleName? ¿Estos cambios perdurarán en la base de datos*?

Ventana2: Realizar los testeos de a un test por vez.

Test1: Utilizando el nivel de Isolation Read Uncommited

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

SELECT \*

FROM [Person].[Person]

WHERE FirstName='Terri' and LastName='Duffy'

Test2: Utilizando el hint (nolock)

SET TRANSACTION ISOLATION LEVEL READ COMMITTED

SELECT \*

FROM [Person].[Person] (nolock)

WHERE FirstName='Terri' and LastName='Duffy'

Test3: Utilizando el nivel de Isolation Read Commited

SET TRANSACTION ISOLATION LEVEL READ COMMITTED

SELECT \*

FROM [Person].[Person]

WHERE FirstName='Terri' and LastName='Duffy'

*Pregunta 12*:

¿Qué valores quedaron para la columna MiddleName según cada uno de los niveles de Isolation?

¿Qué hace el hint (nolock)?

¿Qué sucede al ejecutar sp\_who2 cuando estamos ejecutando el Test3 y cuál es el estado de las sesiones?

4.3 Cerrar todas las ventanas activas. -- /\* \*/

4.4 READ UNCOMMITTED: Ejecutar las siguientes sentencias en dos ventanas diferentes.

¿Qué valor nos devuelve la ventana 2: con el nuevo o anterior costo?

¿Qué sucedería si la ventana 1 termina en rollback?

Ventana1:

SET TRANSACTION ISOLATION LEVEL READ COMMITTED

BEGIN TRAN

SELECT StandardCost,P.\*

FROM Production.Product P

where ProductID=1

UPDATE Production.Product

SET StandardCost = 100

where ProductID=1

Ventana2:

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

BEGIN TRAN

SELECT StandardCost,P.\*

FROM Production.Product P

where ProductID=1

4.5 SERIALIZABLE: Ejecutar las siguientes sentencias en dos ventanas diferentes.

¿Qué sucede al querer realizar el update en la segunda ventana?

¿Qué lockeos tomó la ventana 1 y por qué no son compatibles con la ventana 2?

Ventana1:

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

BEGIN TRAN

SELECT StandardCost,P.\*

FROM Production.Product P

where ProductID=1

Ventana2:

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

BEGIN TRAN

SELECT StandardCost,P.\*

FROM Production.Product P

where ProductID=1

UPDATE Production.Product

SET StandardCost = 100

where ProductID=1

4.6 REPEATABLE READ: Ejecutar las siguientes sentencias en dos ventanas diferentes.

¿Qué sucede al querer realizar el update en la segunda ventana para el producto 2?

¿Qué sucede si en la ventana 2 se quiere cambiar el producto 1?

Ventana1:

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ

BEGIN TRAN

SELECT StandardCost,P.\*

FROM Production.Product P

where ProductID=1

Ventana2:

SET TRANSACTION ISOLATION LEVEL READ COMMITTED

BEGIN TRAN

SELECT StandardCost,P.\*

FROM Production.Product P

where ProductID=2

UPDATE Production.Product

SET StandardCost = 100

where ProductID=2

5. **Seteos de configuración**: El valor de la configuración LOCK\_TIMEOUT establecerá el tiempo de espera.

¿Qué sucede cuando ha pasado ese tiempo de lock?

Verificar el siguiente ejemplo:

Ventana1:

SET TRANSACTION ISOLATION LEVEL READ COMMITTED

BEGIN TRAN

UPDATE Person.Test1

SET CAMPO1=1000

WHERE CAMPO1=999

Ventana2:

SET TRANSACTION ISOLATION LEVEL READ COMMITTED

BEGIN TRAN

UPDATE Person.Test1

SET CAMPO1=1000

WHERE CAMPO1=999

*Pregunta de Evaluación 11*:

¿Cuánto espera el motor de base de datos por un lockeo en un objeto?

**6. Locks:** Verificamos ahora los lockeos que establece cada uno de los niveles de aislamiento. Identificar cada uno de los lockeos. ¿En qué momento se liberan los lockeos establecidos?

Test 1:

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE

begin tran

select StandardCost,p.\*

from Production.Product P

where StandardCost between 100 and 150

SELECT resource\_associated\_entity\_id, resource\_type,request\_mode, request\_session\_id

FROM sys.dm\_tran\_locks

where request\_session\_id=@@SPID

ROLLBACK TRAN

Test 2:

SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED

begin tran

select StandardCost,p.\*

from Production.Product P

where StandardCost between 100 and 150

SELECT resource\_associated\_entity\_id, resource\_type,request\_mode, request\_session\_id

FROM sys.dm\_tran\_locks

where request\_session\_id=@@SPID

ROLLBACK TRAN

Test 3:

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ

begin tran

select StandardCost,p.\*

from Production.Product P

where StandardCost between 100 and 150

SELECT resource\_associated\_entity\_id, resource\_type,request\_mode, request\_session\_id

FROM sys.dm\_tran\_locks

where request\_session\_id=@@SPID

ROLLBACK TRAN

*Pregunta 13:*

*¿Cuándo se liberan los objetos tomados por una sesión?*

**ANEXO 1**

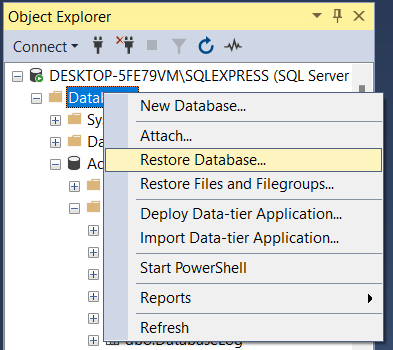
Acceder al siguiente Link:

<https://docs.microsoft.com/en-us/sql/samples/adventureworks-install-configure?view=sql-server-ver15&tabs=ssms>

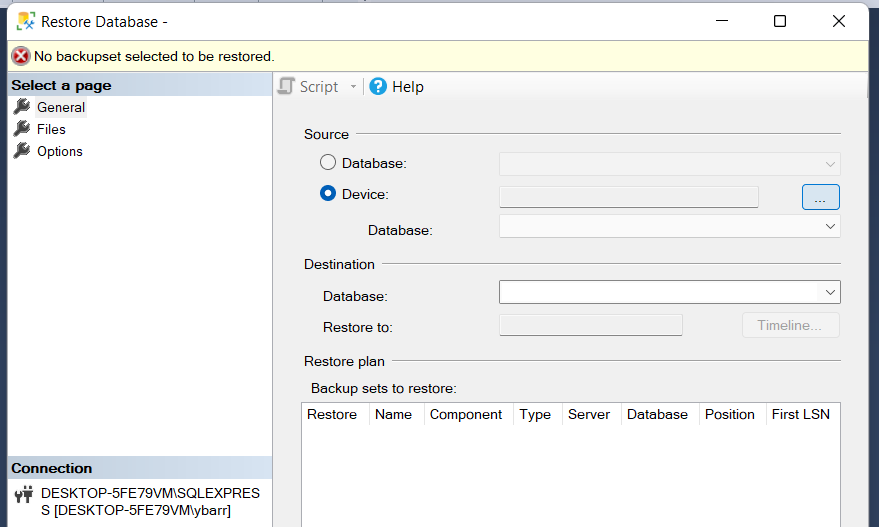
Seleccionar: [AdventureWorks2017.bak](https://github.com/Microsoft/sql-server-samples/releases/download/adventureworks/AdventureWorks2017.bak)

Copiar el archivo en la carpeta Backup de SQL Server. (Utilizar PowerShell de Windows con permisos de Administrador)

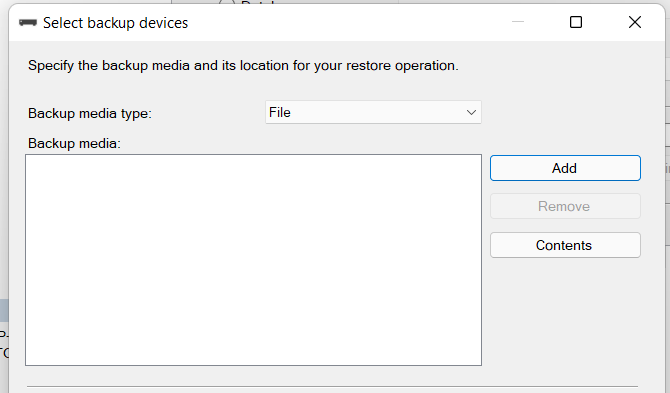
Abrir SSMS, hacer clic en el botón derecho del mouse sobre “Databases”. Seleccionar la opción “Restore Database”



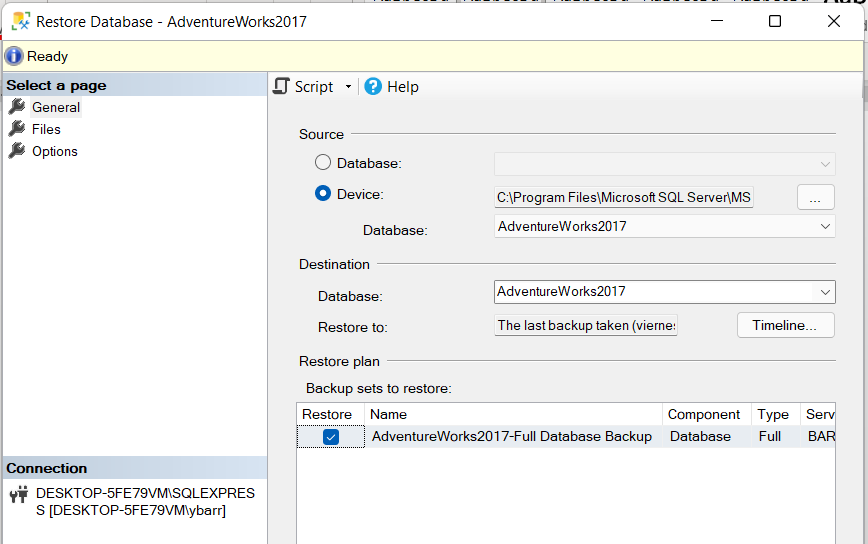
Seleccionar la opción “Device”



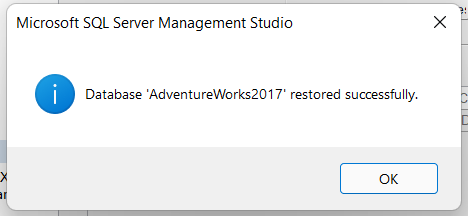
Presionar en el botón con “3 puntos”.



Presionar en el botón “Add” y elegir el archivo “[AdventureWorks2017.bak](https://github.com/Microsoft/sql-server-samples/releases/download/adventureworks/AdventureWorks2017.bak)”



Confirmar con “ok”



El sistema informara el correcto “Restore”