

Lab 7 de Bases de Datos

Hecho por

Carlos Sanchez (C17226) y Carlos Ramirez (B96360)

EJERCICIO 1

6. Sesión A

-- Comandos --

```
set implicit_transactions off;  
set transaction isolation level read  
uncommitted;  
begin transaction t1; PRINT  
@@TRANCOUNT  
Select avg(Nota) from Lleva;
```

-- Preguntas --

a) ¿Qué efecto tiene este comando?

Crea una transacción con el nivel de aislamiento de read uncommitted, imprime la cantidad de transacciones abiertas en la sesión actual, y muestra el promedio de notas en la tabla lleva

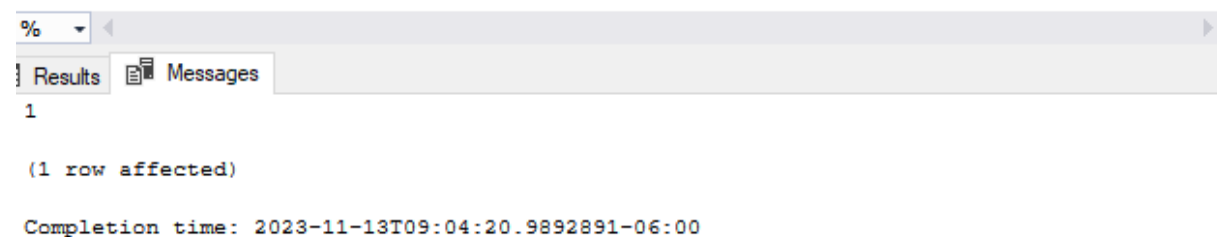
b) ¿Qué hace este comando?

Imprime la cantidad de transacciones abiertas en la sesión actual, y muestra el promedio de notas en la tabla lleva

c) ¿Qué información ofrece la variable @@TRANCOUNT

Es la cantidad de transacciones creadas en la sesión actual

```
use C17226;  
set implicit_transactions off;  
set transaction isolation level read  
uncommitted;  
begin transaction t1; PRINT  
@@TRANCOUNT  
Select avg(Nota) from Lleva;
```



1

(1 row affected)

Completion time: 2023-11-13T09:04:20.9892891-06:00

7. Sesión B

-- Comandos --

```
set implicit_transactions off;  
begin transaction t2;  
PRINT @@TRANCOUNT  
Select * from sys.sysprocesses
```

```

where open_tran = 1;
Update Lleva
set
Nota = Nota*(0.8)
where Nota is not
null;

```

-- Preguntas --

a) Que valor se despliega el PRINT? Por que?

Despliega la cantidad de transacciones creadas en la sesion actual, porque esa es la variable de una sesion en especifico que da informacion de un atributo de la sesion actual

b) Que retorna esta consulta? Que informacion ofrece?

Retorna informacion sobre los procesos que tienen una transaccion abierta, pero solo si tiene una abierta, si se crean mas transacciones en la sesion no se mostrara nada, si se vuelve a ejecutar no devolvera nada

c) En que se diferencia de @@TRANCOUNT?

Permite ver las transacciones y su informacion en varias sesiones del mismo usuario, pero solo si cada sesion tiene una sola transaccion abierta, tranccount solo permite ver la cantidad de transacciones abiertas en una sola sesion del usuario

```

-- 7
use C17226;
set implicit_transactions off;
begin transaction t2;
PRINT @@TRANCOUNT
Select * from sys.sysprocesses
where open_tran = 1;
Update Lleva
set
Nota = Nota*(0.8)
where Nota is not
null;

PRINT
@@TRANCOUNT
select * from Lleva;

```

100 %

Results Messages

1

(1 row affected)

(3 rows affected)

Completion time: 2023-11-13T09:13:48.0766521-06:00

8. Sesion A

-- Comandos --

```
Select avg(Nota) from Lleva;
```

-- Preguntas --

a) ¿Cuál es el resultado de esta consulta?

Da el promedio de las notas de la tabla lleva pero con el valor

b) ¿Es este resultado igual al obtenido antes de ejecutar t2 (comparar con captura de pantalla anterior)?

No, es diferente

c) ¿Leyó t1 el cambio hecho por t2?

Si

d) ¿Qué tipo de problema ejemplifica esta situación?

Ejemplifica el problema de dirty read, porque la transacción no ha finalizado, y es capaz de hacer roll back, lo que dejaría con datos inconsistentes en la transacción que leyó y se quedó con los datos modificados

e) ¿Es este el comportamiento esperado, dado el nivel de aislamiento especificado?

Si por que no se ha protegido la información con los niveles de aislamiento que se proveen en las transacciones, como el read committed

9. Sesión A

-- Comandos --

rollback transaction t2;

-- Preguntas --

a) ¿Qué efecto tiene este comando?

Hace que los datos modificados se deshagan y vuelvan al estado anterior

10. Sesión A

-- Comandos --

Select avg(Nota) from Lleva;

Commit transaction t1;

-- Preguntas --

a) ¿Cuál es el resultado de esta consulta?

El promedio de las notas actualmente

b) ¿Difiere este valor del leído antes de que t2 hiciera rollback?

Si

c) ¿Por qué?

Si porque el valor antes del rollback era cuando t2 hizo una modificación a las notas haciendo que tuvieran valores diferentes, pero con el rollback los cambios hechos son devueltos y el resultado es diferente antes del rollback

EJERCICIO 2

12. Sesión A

-- Comandos --

```
set implicit_transactions off;  
set transaction isolation level  
read committed;  
begin transaction t3;  
Select avg(Nota) from Lleva;
```

-- Preguntas --

a) ¿Qué hace este comando?

Setea las transacciones implícitas en falso, y activa el nivel de aislamiento de transacciones a read committed, inicia una transacción y lee datos

13. Sesión B

-- Comandos --

```
set implicit_transactions off;  
begin transaction t4;  
Update Lleva  
set  
Nota = Nota*(0.8)  
where Nota is not  
null;
```

14. Sesión A

-- Comandos --

```
Select max(Nota) from Lleva;
```

-- Preguntas --

a) ¿Qué paso?

El DBMS se queda trabado completando la consulta, pero nunca se va a completar porque los datos que se quieren leer están protegidos por el nivel de aislamiento de read committed de la transacción t3 que no se ha acabado todavía, y esto no permite que nadie pueda leer datos que la transacción está modificando actualmente, por lo que es comportamiento esperado

15. Sesión B

-- Comandos --

```
Select * from  
sys.sysprocesses where  
open_tran = 1 commit transaction t4
```

-- Preguntas --

a) ¿Qué hace este comando? Revise el contenido de las columnas waitresource y lastwaittype.

Regresa una tabla con los datos de transacciones actualmente corriendo y abiertas del usuario

b) ¿Qué indican estos valores?

Estos valores indican el tipo de dato por el que una transacción dada está esperando para poder seguir, y también indica el evento por el cual el proceso está actualmente esperando

c) ¿Qué efecto tiene este comando sobre la BD?

imprime los procesos del usuario, hace commit de la transacción read committed de t4 que modificó datos, y lee los datos que la transacción estaba sosteniendo

16. Sesión A

-- Comandos --

commit transaction t3;

-- Preguntas --

¿Qué efecto tuvo el commit de t4 sobre la consulta que estaba en espera aquí?

El commit hace que se pueda ver la información que esa transacción estaba intentando leer desbloqueándola

¿El comportamiento observado (t3 calcula el promedio de las notas antes de que t4 actualice las notas pero calcula la nota máxima después de que t4 hace los cambios) es consistente con el nivel de aislamiento de la transacción t3?

Si lo es porque cada transacción acaba y una espera a que otra acabe para no tener problemas

¿Es este schedule equivalente a algún schedule serial de las transacciones t3 y t4?

Jus+fique.

Si lo sería porque la t3 queda haciendo la consulta, mientras que t4 después de que t3 lee valores empieza a modificar los datos leídos, registra esos datos con el commit y acaba

EJERCICIO 3

18. Sesión A

-- Comandos --

set implicit_transactions off;

set transaction isolation level

repeatable read;

begin transaction t5;

```
Select avg(Nota) from
```

```
Lleva;
```

```
-- Preguntas --
```

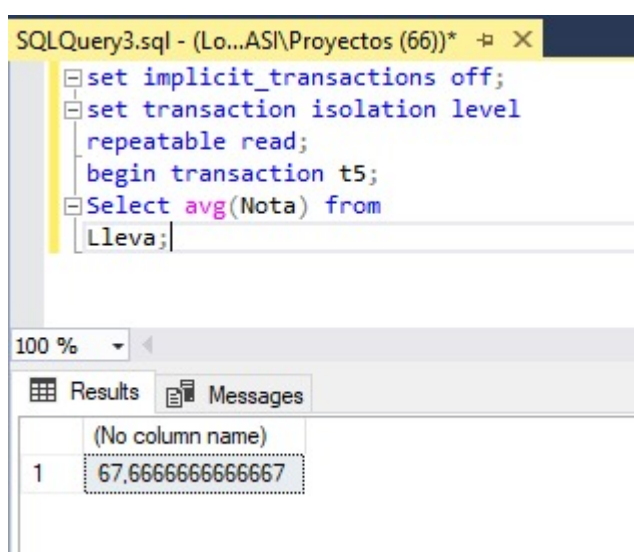
a) ¿Qué hace este comando?

SET IMPLICIT_TRANSACTIONS OFF;: Esta instrucción desactiva el modo de transacciones implícitas. Cuando este modo está activado, cada sentencia de transacción (como INSERT, UPDATE, DELETE) inicia automáticamente una nueva transacción si no hay ninguna transacción activa

SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;: Establece el nivel de aislamiento de la transacción en "REPEATABLE READ" (lectura repetible). Este nivel de aislamiento asegura que las transacciones concurrentes no puedan modificar ni insertar nuevos datos que afecten a la transacción en curso

BEGIN TRANSACTION T5;: Inicia una nueva transacción con el nombre "T5". Esto marca el inicio de una transacción atómica. Si algo sale mal, todas las operaciones realizadas dentro de esta transacción se revertirán

SELECT AVG(Nota) FROM Lleva;: Realiza una consulta para calcular el promedio de la columna "Nota" en la tabla "Lleva". Esto devuelve el valor promedio de las notas en la tabla.



19. Sesión B

-- Comandos --

```
set implicit_transactions off;
```

```
begin transaction t6;
```

```
Insert into Lleva
```

```
(CedEstudiante, SiglaCurso,
```

```
NumGrupo, Semestre, Anho, Nota)
```

```
values('4444444444', 'KFC', 2, 2,
```

```
3200, 85);
```

```
commit transaction t6;
```

-- Preguntas --

a) ¿Este comando se ejecuta o queda en espera?

Se ejecuta

b) ¿Modifica este insert los valores/tuplas leídos por t5?

No, no afecta a los valores leídos por t5

c) ¿Es correcto entonces que se permita el insert bajo el nivel de aislamiento de t5?

Sí, es correcto. El repeatable read bloquea lecturas pero permite la escritura. Se puede hacer el insert sin esperar a que T5 termine.

d) ¿Qué hubiera pasado si en vez de un insert, t6 hubiese ejecutado un update de Nota sobre la tabla Lleva? ¿Qué esperaríamos que sucediera?

Si se ejecuta update en t6 y se actualiza uno de los valores que se usan en t5, habría un conflicto. La t5 bloquea las lecturas y no permite que otras transacciones realicen actualizaciones en los mismos datos hasta que t5 se complete.

```
-- Comandos --
set implicit_transactions off;
begin transaction t6;
Insert into Lleva
(CedEstudiante, SiglaCurso,
NumGrupo, Semestre, Anho, Nota)
values('4444444444', 'KFC', 2, 2,
3200, 85);
```

100 %

Results Messages

	CedEstudiante	SiglaCurso	NumGrupo	Semestre	Anho	Nota
1	1234567809	ABC	2	2	3200	40
2	1234567809	KFC	2	2	3200	99
3	1234567890	ABC	1	1	2020	64
4	4444444444	KFC	2	2	3200	85

20. Sesión A

-- Comandos --

Select avg(Nota) from Lleva;

commit transaction t5;

-- Preguntas --

a) ¿Qué resultado obtiene?

b) ¿Es este resultado igual al que obtuvo antes de ejecutar t6 (comparar con captura de pantalla anterior)?

Es diferente, antes daba 67.67

c) ¿Qué tipo de problema de concurrencia ejemplifica esta situación?

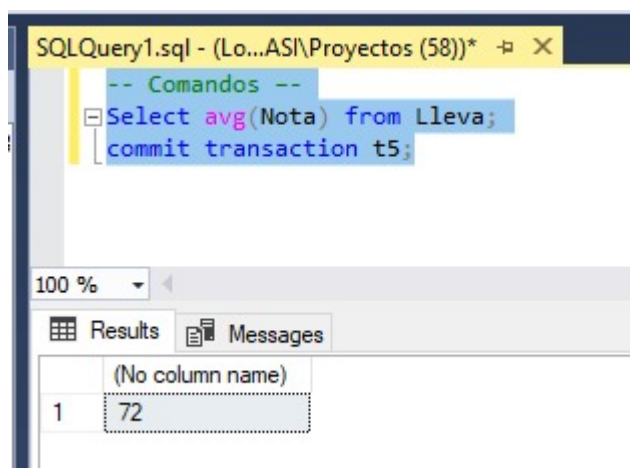
Lecturas fantasma. Bajo el nivel de aislamiento de repeatable read, las transacciones pueden experimentar lecturas que incluyen filas que no existían cuando comenzó la transacción.

d) ¿Concuerda este comportamiento con el nivel de aislamiento de la transacción t5? Explique.

Sí, el comportamiento concuerda con el nivel de aislamiento de repeatable read. Este nivel permite que las transacciones realicen lecturas pero no garantiza que las filas que se leyeron al inicio de la transacción permanezcan inalteradas hasta que se complete la transacción. En este caso, la transacción t5 al hacer su Select del promedio de notas otra vez (y después de ejecutar t6), entonces el promedio es diferente porque incluye la nota insertada por la transacción t6.

e) ¿Es este schedule equivalente a algún schedule serial de las transacciones t5 y t6? Justifique.

No, este schedule no es equivalente a ningún schedule serial de T5 y T6. Un schedule serial implica ejecutar las transacciones de manera secuencial sin solapamientos. En este caso, las transacciones T5 y T6 están ejecutando operaciones concurrentes, lo que significa que no es equivalente a un schedule serial.



EJERCICIO 4

22. Sesión A

-- Comandos

```
set implicit_transactions off;
```

```
set transaction isolation level
```

```
serializable;
```

```
begin transaction t7;
```

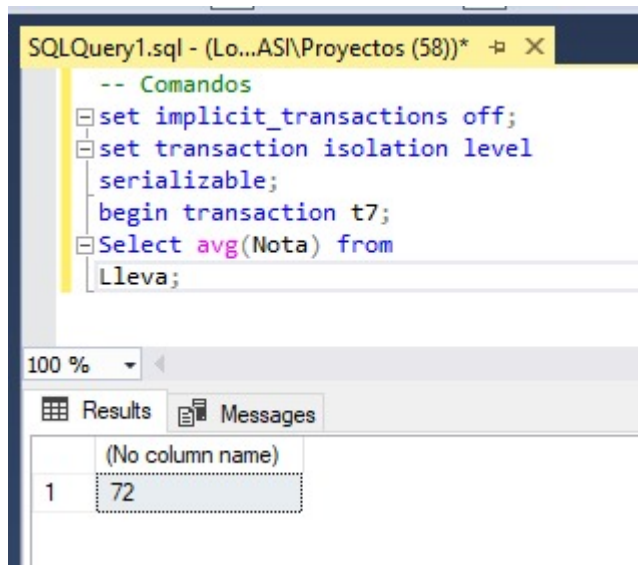
```
Select avg(Nota) from
```

```
Lleva;
```

-- Preguntas --

a) ¿Qué hace este comando?

Es igual que la definición del ejercicio anterior, con la diferencia que se usa un nivel de aislamiento serializable. Esto implica que en t7 exista un nivel de aislamiento más alto, que garantiza que las transacciones concurrentes no puedan afectar la transacción en curso (la t7), ya que impone bloqueos de lectura y escritura en los datos involucrados.



23. Sesión B

-- Comandos --

```
set implicit_transactions off;
```

```
begin transaction t8;
```

```
Insert into Lleva
```

```
(CedEstudiante, SiglaCurso,
```

```
NumGrupo, Semestre, Anho, Nota)
```

```
values('4444444444', 'ABC', 1, 1,
```

```
2020, 85);
```

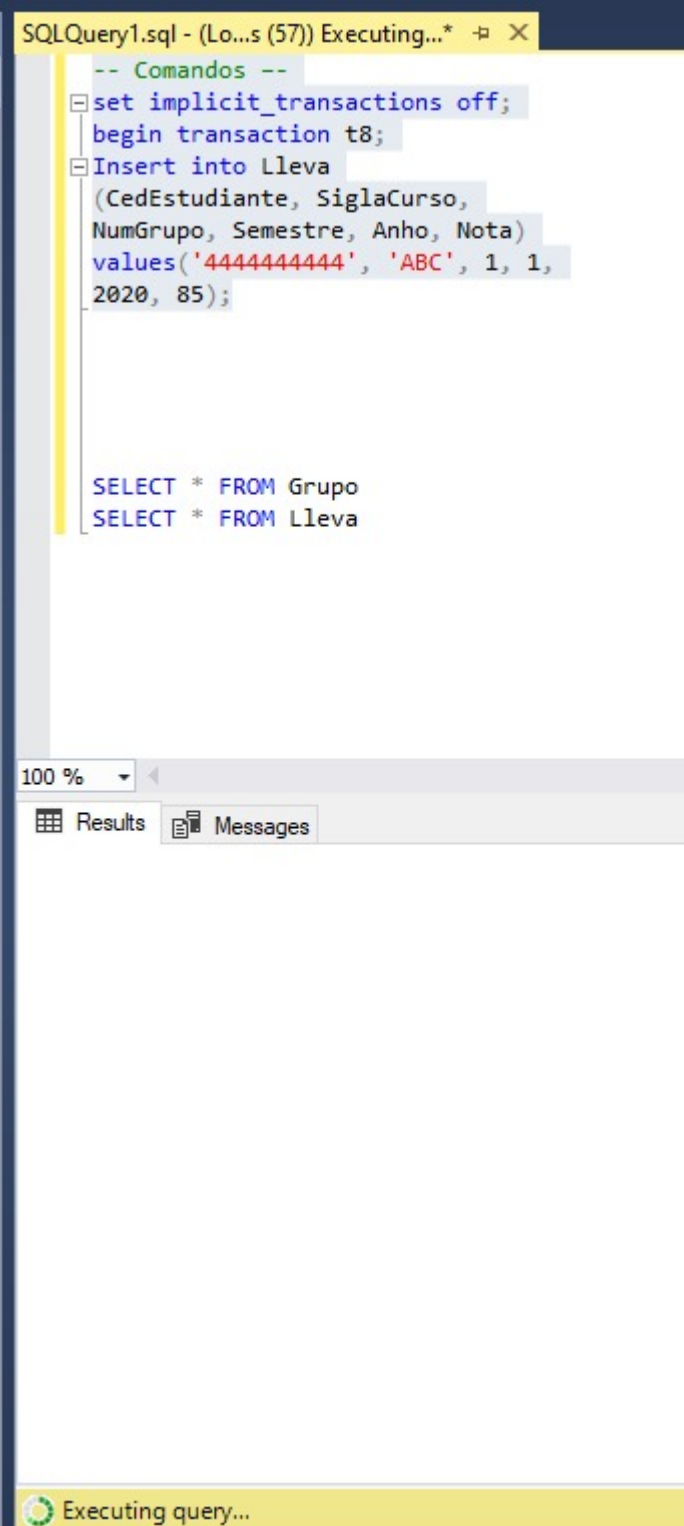
-- Preguntas --

a) ¿Este comando se ejecuta o queda en espera?

En espera

b) ¿Es esperado este comportamiento de acuerdo con el nivel de aislamiento de t7?

Es esperado, pues la transacción en la ventana B, que intenta realizar un INSERT, realiza una espera indefinida debido a los bloqueos impuestos por el nivel de aislamiento serializable definido en t7.



The screenshot displays the SQL Server Enterprise Manager interface. At the top, a query window titled "SQLQuery1.sql - (Local) (57) Executing..." is open. The window contains the following SQL code:

```
-- Comandos --
set implicit_transactions off;
begin transaction t8;
Insert into Lleva
(CedEstudiante, SiglaCurso,
NumGrupo, Semestre, Anho, Nota)
values('4444444444', 'ABC', 1, 1,
2020, 85);

SELECT * FROM Grupo
SELECT * FROM Lleva
```

Below the query window, there is a toolbar with a "Results" button (represented by a grid icon) and a "Messages" button (represented by a document icon). The status bar at the bottom of the window shows a green circular icon and the text "Executing query...".

24. Sesión A

-- Comandos --

```
Select avg(Nota) from Lleva;
```

```
commit transaction t7;
```

-- Preguntas --

a) ¿Qué resultado obtiene?

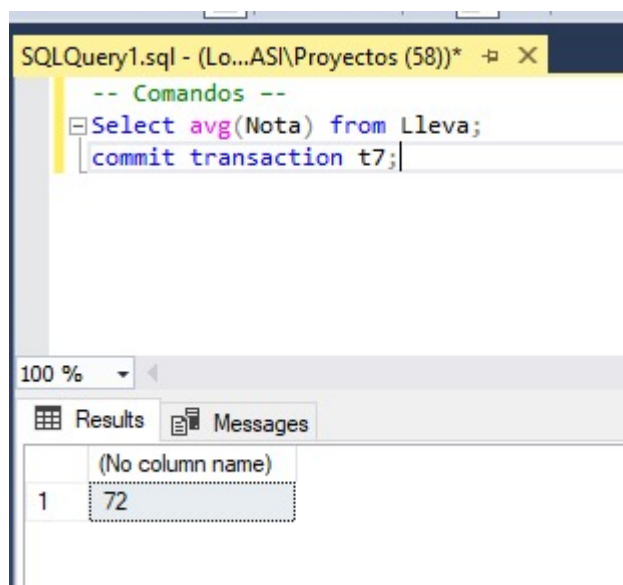
72

b) ¿Es este resultado igual al que obtuvo antes de ejecutar t8?

Igual

c) ¿Es este schedule equivalente a algún schedule serial de las transacciones t7 y t8?
Justifique.

Sí. Es un schedule serial entre t7 y t8, pues solo cuando a la t7 se le hizo su commit, la t8 pudo salir de su espera infinita y realizar su insertado. Entonces, t8 no hizo nada hasta que terminó definitivamente t7, gracias al nivel de aislamiento serializable definido en t7.



25. Sesión B

-- Comandos --

```
commit transaction t8;
```

-- Preguntas --

a) ¿Pasó algo?

Hizo un commit de la t8, pero no hizo ninguna operación adicional

b) ¿Qué efecto tuvo el commit de t7 sobre el insert que estaba en espera aquí?

Ligado a la respuesta anterior, el insert en t8 se realizó justo inmediatamente después de que t7 fuera commiteada. Porque una vez la t7 hizo su commit, la espera infinita se rompió y t8 por fin pudo realizar su insertado, sin necesidad de hacer un commit para la t8 para realizar dicho insertado. Esto afectó la tabla Lleva y el promedio de notas en dicha tabla.

SQLQuery1.sql - (Lo...ASI\Proyectos (57))*

```
-- Comandos --  
commit transaction t8;  
  
SELECT * FROM Lleva  
  
Select avg(Nota) from Lleva;
```

100 %

Results Messages

	CedEstudiante	SiglaCurso	NumGrupo	Semestre	Anho	Nota
1	1234567809	ABC	2	2	3200	40
2	1234567809	KFC	2	2	3200	99
3	1234567890	ABC	1	1	2020	64
4	4444444444	ABC	1	1	2020	85
5	4444444444	KFC	2	2	3200	85

	(No column name)
1	74,6