Básandote en el ejemplo de ejecución de Oracle visto en clase, vas a hacer dos Ejemplos de ejecución. Para cada ejemplo, ejecuta en dos sesiones concurrentes, las mismas operaciones de consulta y actualización usando la BDejemplo. El objetivo es provocar dos comportamientos distintos cuando usas los dos niveles de aislamiento : Read Commited y Serializable

- Ejemplo 1: las dos sesiones con Read Commited

- Ejemplo 2: : las dos sesiones con Serializable

Para entregar: Haz una tabla con los pasos seguidos en el tiempo, y dos columnas, una para cada sesión, indicando cada paso dado y resultado obtenido. Las respuestas a las preguntas las contestas dentro de la misma tabla. Separa los dos ejemplos en dos tablas:

======= EJEMPLO 1 ===========

PASOS: Síguelos en el orden descrito:

- Abre dos conexiones (en el enunciado: Sesión 1 es la 1ª que abriste)

- abriendo sql Developer dos veces y conectándote con tu usuario

- después abre un editor: menú Herramientas + Hoja de Trabajo SQL

- en la Sesión 1:

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

set autocommit off; (siempre puedes comprobarlo con show autocommit)

- en la Sesión 2:

SET TRANSACTION ISOLATION LEVEL READ COMMITTED;

set autocommit off;

- en la Sesión 1:

Una consulta de una tabla (por ejemplo COMPRAS) que solo devuelva una fila

Actualiza la misma fila y el mismo atributo

Consulta esa fila y atributo: ¿Qué valor ves?

- en la Sesión 2:

- Consulta esa fila y atributo: ves lo mismo?

- Actualiza la misma fila y el mismo atributo con otro valor

-Que sucede?

Oracle no debería responder : Porqué?

- en la Sesión 1:

- Escribe un commit

|  |  |
| --- | --- |
| SESSION 1 | SESSION 2 |
| SET TRANSACTION ISOLATION LEVEL READ COMMITTED;  transaction ISOLATION correcto.  SELECT \* from compras WHERE importe=50;  DNI NUMT NUMF FECHA TIENDA IMPORTE  -------- ---------- ---------- ---------- -------------------- ----------  00000005 50000400 1 501 tienda1 50  UPDATE compras set fecha=505 WHERE importe = 50;  1 filas actualizadas.  SELECT \* from compras where importe=50;  DNI NUMT NUMF FECHA TIENDA IMPORTE  -------- ---------- ---------- ---------- -------------------- ----------  00000005 50000400 1 505 tienda1 50 | SET TRANSACTION ISOLATION LEVEL READ COMMITTED;  transaction ISOLATION correcto. |
|  | SELECT \* from compras where importe=50;  DNI NUMT NUMF FECHA TIENDA IMPORTE  -------- ---------- ---------- ---------- -------------------- ----------  00000005 50000400 1 501 tienda1 50  UPDATE compras set fecha=508 where importe = 50;  << SE QUEDA BLOQUEADO EL PROMPT >> |
| COMMIT; |  |
|  | 1 filas actualizadas |
| SELECT \* from compras where importe=50;  DNI NUMT NUMF FECHA TIENDA IMPORTE  -------- ---------- ---------- ---------- -------------------- ----------  00000005 50000400 1 505 tienda1 50 |  |
|  | COMMIT; |
| SELECT \* from compras where importe=50;  DNI NUMT NUMF FECHA TIENDA IMPORTE  -------- ---------- ---------- ---------- -------------------- ----------  00000005 50000400 1 508 tienda1 50 |  |
|  | SELECT \* from compras where importe=50;  DNI NUMT NUMF FECHA TIENDA IMPORTE  -------- ---------- ---------- ---------- -------------------- ----------  00000005 50000400 1 508 tienda1 50 |

- Consulta esa fila y atributo: que ves?, de donde ha salido?

Se ve el dato actualizado por la transacción 2.

- Se ha perdido la actualización de sesión 1?

Si, por que cuando la transacción 2 ha intentado actualizar se queda esperando a que la 1 haga commit de su actualización, cuando esto ocurre la transacción 2 actualiza eliminando el dato puesto por la 1.

- Cómo solucionarlo? Ver la ejecución del ejemplo 2 a continuación

Con un nivel de aislamiento serializable se consigue que los datos de otras sesiones no afecten a la actual hasta que esta no realice un commit.

======= EJEMPLO 2 ===========

? Repetir los mismos pasos, teniendo en ambas sesiones

SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;

tabla.

|  |  |
| --- | --- |
| SESSION 1 | SESSION 2 |
| SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;  transaction ISOLATION correcto. | SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;  transaction ISOLATION correcto. |
| SELECT \* from compras WHERE importe=50;  DNI NUMT NUMF FECHA TIENDA IMPORTE  -------- ---------- ---------- ---------- -------------------- ----------  00000005 50000400 1 501 tienda1 50  UPDATE compras set fecha=505 WHERE importe = 50;  1 filas actualizadas.  SELECT \* from compras where importe=50;  DNI NUMT NUMF FECHA TIENDA IMPORTE  -------- ---------- ---------- ---------- -------------------- ----------  00000005 50000400 1 505 tienda1 50 |  |
|  | SELECT \* from compras where importe=50;  DNI NUMT NUMF FECHA TIENDA IMPORTE  -------- ---------- ---------- ---------- -------------------- ----------  00000005 50000400 1 501 tienda1 50  UPDATE compras set fecha=508 where importe = 50;  << SE QUEDA BLOQUEADO EL PROMPT >> |
| COMMIT; |  |
|  | Informe de error:  Error SQL: ORA-08177: no se puede serializar el acceso para esta transacción  08177. 00000 - "can't serialize access for this transaction"  \*Cause: Encountered data changed by an operation that occurred after  the start of this serializable transaction.  \*Action: In read/write transactions, retry the intended operation or  transaction. |
| SELECT \* from compras where importe=50;  DNI NUMT NUMF FECHA TIENDA IMPORTE  -------- ---------- ---------- ---------- -------------------- ----------  00000005 50000400 1 505 tienda1 50 |  |
|  | COMMIT; |
| SELECT \* from compras where importe=50;  DNI NUMT NUMF FECHA TIENDA IMPORTE  -------- ---------- ---------- ---------- -------------------- ----------  00000005 50000400 1 505 tienda1 50 |  |
|  | SELECT \* from compras where importe=50;  DNI NUMT NUMF FECHA TIENDA IMPORTE  -------- ---------- ---------- ---------- -------------------- ----------  00000005 50000400 1 505 tienda1 50 |

- Se ha perdido algo ahora?.

No se ha perdido ninguna actualización

- Cómo lo ha solucionado Oracle?

Oracle no ha permitido la actualización de la sesión 2 por que no ha podido garantizar la serialización de la operación.