Transacciones

1. Introducción

El **aislamiento** es una propiedad que define **cómo y cuando los cambios producidos por una operación se hacen visibles para las demás operaciones concurrentes**.

La mayor parte de los SGBDR ofrecen ciertos niveles de aislamiento que controlan el grado de bloqueo durante el acceso a los datos. El estándar ANSI/ISO SQL define los **niveles de aislamiento** como modos de operación que evitan tres fenómenos no deseados en concurrencia: lectura sucia (*dirty read*), lectura no repetible (*fuzzy read*) y lectura fantasma (*phantom row*).

- **Lectura sucia**: provoca que en una transacción se pueda leer una fila que ha sido cambiada, pero no confirmada, en otra.
- **Lectura no repetible**: ocurre cuando en el curso de una transacción una fila se lee dos veces y los valores no coinciden por haber sido modificada en otra transacción.
- **Lectura fantasma**: tiene lugar cuando una transacción opera sobre un determinado conjunto de filas, conjunto que modifica (insertando nuevos valores, por ejemplo) otra transacción y en la primera se ven los cambios introducidos por la segunda.

La mayor parte de los SGBD ofrecen ciertos niveles de aislamiento que tratan de evitar estos fenómenos controlando el grado de bloqueo durante el acceso a los datos. Estos niveles están definidos por el estándar SQL y son:

- **Read Uncommitted**
- **Read Committed** (bloqueo de escritura)
- **Repeatable Read** (bloqueos de lectura y escritura)
- **Serializable** (nivel de aislamiento más alto; bloqueos de lectura, escritura y rango)

Un nivel de aislamiento menor significa que los usuarios tienen un mayor acceso a los datos simultáneamente, con lo que aumentan tipos de efectos como la lectura sucia o la pérdida de actualizaciones. Por el contrario, un nivel de aislamiento mayor reduce los tipos de efectos debidos a la simultaneidad, pero requiere más recursos del sistema y aumenta las posibilidades de que una transacción bloquee a otra.

El nivel de aislamiento superior, ***Serializable***, garantiza que una transacción recuperará exactamente los mismos datos cada vez que repita una operación de lectura, aunque para ello aplicará un nivel de bloqueo que puede afectar a los demás usuarios en los sistemas multiusuario.

El nivel de aislamiento menor, ***Read Uncommitted***, puede recuperar datos que otras transacciones han modificado, pero no confirmado. En este nivel se pueden producir todos los efectos secundarios de simultaneidad, pero no hay bloqueos ni versiones de lectura, por lo que la sobrecarga se reduce.

A modo de resumen, a mayor nivel de aislamiento menos posibilidades hay de que se produzcan efectos no deseados (mayor seguridad, por tanto) pero a costa de obtener un menor grado de concurrencia y más consumo de recursos.

El nivel de aislamiento apropiado depende, por tanto, del equilibrio entre los requisitos de integridad de los datos de la aplicación y la sobrecarga de cada nivel de aislamiento.

La relación entre los niveles de aislamiento y los efectos de lectura se puede resumir como:

Nivel de aislamiento / Efecto de lectura	Lectura sucia	Lectura no repetible	Lectura fantasma
Read Uncommitted	Es posible	Es posible	Es posible
Read Committed	-	Es posible	Es posible
Repeatable Read	-	-	Es posible
Serializable	-	-	-

Para obtener el mayor nivel de aislamiento, un SGBD generalmente hace un bloqueo de los datos o implementa un control de concurrencia mediante versiones múltiples (MVCC). Los dos SGBD que vamos a utilizar en esta práctica, tienen las siguientes características en cuanto al control de concurrencia:

Oracle: Soporta MVCC. Soporta los niveles de aislamiento READ COMMITTED y and SERIALIZABLE tal como se definen en el estándard (https://docs.oracle.com/cd/B10501_01/server.920/a96524/c21cnsis.htm#2570).

HSQLdb: Soporta diferentes mecanismos de control de concurrencia. Por defecto, bloqueo en dos fases (2PL), y multiversion concurrency control (MVCC). 2PL es adecuado para aplicaciones con una conexión a la BBDD o aquellas que no acceden frecuentemente a una misma tabla para realizar escrituras concurrentes. Con múltiples conexiones simultáneas, puede usarse MVCC.

El estándar indica que el motor de la BBDD retorne un nivel de aislamiento mayor que el solicitado en caso de que el solicitado no se implemente. Por tanto, HyperSQL promociona READ UNCOMMITTED a READ COMMITTED y REPEATABLE READ a SERIALIZABLE (http://hsqldb.org/doc/guide/sessions-chapt.html)

Es aconsejable echar un vistazo al siguiente vídeo:

`https://www.youtube.com/watch?v=sxabCqWsFHg`

2. Objetivos

- Comprender y afianzar los conceptos de **transacción, concurrencia y aislamiento**.
- Conocer las diferentes posibilidades de configuración de un SGBD en función del grado que concurrencia que se desee.
- Conocer los diferentes grados de aislamiento de Oracle y HSQLDB.

En definitiva, se trata de comprobar cómo se comportan los diferentes SGBD con los diferentes niveles de aislamiento.

Para lograrlo, **se deben probar los siguientes casos en los dos SGBD indicados y con dos niveles de aislamiento**:

  - **HSQL (LOCKS) con READ COMMITTED y SERIALIZABLE**
  - **HSQL (MVCC) con READ COMMITTED y SERIALIZABLE**
  - **Oracle con READ COMMITTED y SERIALIZABLE**

3. Entorno

- **HSQLDB**:
  - Arrancar el servidor HSQLdb. Para ello, acceder al campus virtual, descargar y descomprimir el archivo "**Base de datos (hsqldb) para Agencia de Viajes**", y ejecutar *data/startup*.
  - Ejecutar a continuación data/runManagerSwing.bat dos veces (**dos sesiones**).
  - En el menú *Options*, **desactivar *Autocommit mode*** en ambas.
- **Oracle**:
  - Arrancar SQL Developer y conectarse con su usuario.
  - Crear la siguiente tabla (hay disponible un script)
    CREATE TABLE TTRIPS (Destination VARCHAR2(200), Price NUMBER(5));
  - Añadir los siguientes viajes:
    - London, 450
    - Paris, 320
    - Rome, 280
  - Desde SQLDeveloper Herramientas – Preferencias - Base de Datos – Avanzada - verificar que **"Confirmación automática" está desactivada**
  - Desde SQLDeveloper Herramientas – Preferencias - Base de Datos –Hoja de trabajo - verificar que las dos primeras casillas **están activadas** (sobre todo "nueva hoja de trabajo para utilizar conexiones no compartidas").
  - Abrir una nueva hoja de trabajo (**dos sesiones**).

4. Instrucciones SQL

Para establecer un **modo de control de transacciones** en SQL escribe lo siguiente:

```
SET DATABASE TRANSACTION CONTROL LOCKS
SET DATABASE TRANSACTION CONTROL MVCC
```

Para cambiar entre diferentes **niveles de aislamiento** (`SET TRANSACTION ISOLATION LEVEL XXXX`) **recuerda ejecutar antes *commit* o *rollback*.**

Para cada ejercicio, anota los resultados de cada operación. ¿Se bloquea algún proceso durante la ejecución? ¿Cuál? ¿Cuándo? ¿Por qué? ¿Qué anomalías se producen en cada ejecución?

Importante: Anota los valores iniciales antes de cada ejecución o asegúrate de volver a escribir los valores originales en las tablas antes de cada ejecución.

5. Dirty read

1. Script 1

TRANSACTION 1 (Ej: Travel agency)	TRANSACTION 2 (Ej: Booking center)
SET TRANSACTION ISOLATION LEVEL XXXXXXX SELECT * from Ttrips	SET TRANSACTION ISOLATION LEVEL XXXXXXX UPDATE Ttrips SET price = price + 10

Finaliza la transacción del script 2 con rollback y el del script 1 con commit.

2. Script 2

TRANSACTION 1 (Ej: Travel agency)	TRANSACTION 2 (Ej: Booking center)
SET TRANSACTION ISOLATION LEVEL XXXXXXX SELECT * from Ttrips SELECT * from Ttrips	SET TRANSACTION ISOLATION LEVEL XXXXXXX UPDATE Ttrips SET price = price + 10

Finaliza la transacción del script 2 con rollback y el del script 1 con commit. ¿ Cuál es el contenido final de la tabla?

6. Unrepeteable read

1. Script 1

TRANSACTION 1 (Ej: Travel agency)	TRANSACTION 2 (Ej: Booking center)
SET TRANSACTION ISOLATION LEVEL XXXXXXX SELECT price FROM Ttrips WHERE destination = 'Paris' SELECT price FROM Ttrips WHERE destination = 'Paris' **Commit** SELECT price FROM Ttrips WHERE destination = 'Paris'	SET TRANSACTION ISOLATION LEVEL XXXXXXX UPDATE Ttrips SET price = 350 WHERE destination = 'Paris' **Commit**

2. Script 2

TRANSACTION 1 (Ej: Travel agency)	TRANSACTION 2 (Ej: Booking center)
SET TRANSACTION ISOLATION LEVEL XXXXXXX SELECT * FROM Ttrips WHERE destination = 'Rome' SELECT * FROM Ttrips WHERE destination = 'Rome'; UPDATE Ttrips SET price=price+10 SELECT * FROM Ttrips	SET TRANSACTION ISOLATION LEVEL XXXXXXX UPDATE Ttrips SET price = 500 WHERE destination = 'Rome' **Commit**

7. Phantom read

TRANSACTION 1 (Ej: Travel agency)	TRANSACTION 2 (Ej: Booking center)
SET TRANSACTION ISOLATION LEVEL XXXXXXX SELECT * FROM Ttrips WHERE price BETWEEN 250 AND 350 SELECT * FROM Ttrips WHERE price BETWEEN 250 AND 350	SET TRANSACTION ISOLATION LEVEL XXXXXXX INSERT INTO Ttrips VALUES ('Madrid',260) **Commit**