Transacciones

Aislamiento

El **aislamiento** es una propiedad que define **cómo y cuándo se hacen visibles** para una transacción las operaciones concurrentes realizadas por otras.

La mayor parte de los SGBDR ofrecen ciertos **niveles de aislamiento** que controlan el grado de acceso a los datos y tratan de evitar tres fenómenos no deseados en concurrencia: lectura sucia (dirty read), lectura no repetible (fuzzy read) y lectura fantasma (phantom row).

El **estándar ANSI/ISO SQL** define los niveles de aislamiento:

- **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)

El nivel de aislamiento apropiado para cada aplicación depende de los requisitos de integridad de los datos de la aplicación y la sobrecarga que introduce de cada nivel de aislamiento.

Control de concurrencia

Para garantizar el nivel de aislamiento, los SGBD utilizan bloqueos o control de concurrencia mediante versiones múltiples (MVCC). El siguiente es el comportamiento general; distintos SGBD pueden implementar modificaciones.

• Basada en bloqueos

Nivel de aislamiento READ UNCOMMITTED (RU)

Operación	Lock	Se establece	Se libera
SELECT	Ninguno	No aplica	No aplica
UPDATE/DELETE/INSERT	X-lock	Al modificar/insertar fila	Fin de txn (Commit/Rollback)

Nivel de aislamiento READ COMMITTED (RC)

Operación	Lock	Se establece	Se libera
SELECT	S-lock sobre filas leída	Al leer cada fila	Tras leer la fila
UPDATE/DELETE/INSERT	X-lock sobre fila add/del/upd	Al modificar/insertar fila	Fin de txn (Com/Roll)

Nivel de aislamiento REPEATABLE READ (RR)

Operación	Lock	Se establece	Se libera
SELECT	S-lock sobre filas leídas	Al leer cada fila que cumpla la condición	Fin de txn (Com/Roll)
UPDATE/DELETE/INSERT	X-lock sobre fila add/del/upd	Al modificar/insertar fila	Fin de txn (Com/Roll)

Nivel de aislamiento SERIALIZABLE (Ser)

Operación	Lock	Se establece	Se libera
SELECT	S-lock sobre filas leídas y rango si la consulta actúa sobre rangos	Al leer filas o rangos	Fin de txn (Com/Roll)
UPDATE/DELETE/INSERT	X-lock	Al modificar filas/rangos	Fin de txn (Com/Roll)

- ## Basada en versiones

Nivel de aislamiento READ UNCOMMITTED (RU)

MVCC no puede funcionar en modo READ UNCOMMITTED.

Nivel de aislamiento READ COMMITTED (RC)

Cuándo obtiene la versión de los datos	Qué versión de datos ve	Validaciones en commit
Al inicio de cada sentencia SQL	Versiones confirmadas antes de la operación de lectura.	Otras txn no confirmaron updates o deletes sobre filas que esta txn actualiza (evita lost-update)

Nivel de aislamiento REPEATABLE READ (RR)

Cuándo obtiene la versión de los datos	Qué versión de datos ve	Validaciones en commit
Al inicio de la txn	Versiones confirmadas antes del comienzo de la txn	Otras txn no confirmaron updates o deletes sobre filas que esta txn actualiza

Nivel de aislamiento SERIALIZABLE (Ser)

Cuándo obtiene la versión de los datos	Qué versión de datos ve	Validaciones en commit
Al inicio de la txn	Versiones confirmadas antes del comienzo de la txn.	Otras txn no confirmaron updates o deletes sobre filas que esta txn actualiza

Oracle y HSQLDB

Los dos SGBD que vamos a utilizar en esta práctica, tienen las siguientes características en cuanto al control de concurrencia:

Por defecto, ambos trabajan en ***autocommit***, es decir, cada instrucción sql es una transacción por sí misma y se confirma automáticamente si termina bien. Por tanto, en ambos casos habrá que desactivar el modo Autocommit en ambas.

- ## Oracle

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

- ## HSQLDB

Soporta diferentes mecanismos de control de concurrencia (https://hsqldb.org/doc/2.0/guide/sessions-chapt.html). Por defecto, bloqueo en dos fases (2PL), pero también y multiversion concurrency control (MVCC).

En ambos casos, 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, HSQLDB promociona READ UNCOMMITTED a READ COMMITTED y REPEATABLE READ a SERIALIZABLE.

Objetivos

- Comprender y afianzar los conceptos de transacción, concurrencia y aislamiento.
- Conocer las 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.
- Para lograrlo, se deben probar los siguientes casos en los dos SGBD indicados y con dos niveles de aislamiento:
    a. **HSQL (LOCKS) con READ COMMITTED y SERIALIZABLE**

b. **HSQL (MVCC) con READ COMMITTED y SERIALIZABLE**
c. **Oracle con READ COMMITTED y SERIALIZABLE**

Entorno

Descargar la base de datos "Base de datos (hsqldb) para Agencia de Viajes" del Campus Virtual. Arrancarla en modo servidor. Abrir dos sesiones con la base de datos y desactivar `Autocommit mode` en ambas.

Arrancar SQL Developer y conectarse a Oracle con su usuario y contraseña. Crear la siguiente tabla (hay disponible un script)

London	450
Paris	320
Rome	280

Abrir dos sesiones de trabajo con modo autocommit desactivado.

Instrucciones para establecer el mecanismo de control de concurrencia en HSQLDB

SET DATABASE TRANSACTION CONTROL LOCKS

SET DATABASE TRANSACTION CONTROL MVCC

Instrucciones para cambiar el nivel de aislamiento

Para cambiar el nivel de aislamiento, es necesario, terminar la transacción actual y ejecutar la instrucción sql:

SET TRANSACTION ISOLATION LEVEL XXXX

Para consultar el nivel de aislamiento que está establecido puedes ejecutar la siguiente consulta:

CALL ISOLATION_LEVEL()

Transacciones a ejecutar

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

Importante: Asegúrate de volver a escribir los valores originales en las tablas antes de cada ejecución.

A. Dirty read

A.1

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

A.2

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

B. Unrepeteable read

B.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

B.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 Commit	SET TRANSACTION ISOLATION LEVEL XXXXXXX UPDATE Ttrips SET price = 500 WHERE destination = 'Rome' Commit

C. Phantom read

C.1

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 Commit	SET TRANSACTION ISOLATION LEVEL XXXXXXX INSERT INTO Ttrips VALUES ('Madrid',260) Commit