Clase Auxiliar 5 CC52K Base de datos 2

SQL Embedded

Profesor: Eduardo Godoy

Profesor Aux: Marcelo Vega (mvega@dcc.uchile.cl)

22 de Mayo del 2006

Transacciones: Test Acid

1.- Isolation (del test ACID)

IMPORTANTE: Una BD debe asegurar que se cumpla un nivel de isolation que impida que ocurran problemas cuando 2 o más transacciones intentan modificar el mismo valor. Sin embargo, no tienen porque soportar (ni por parte de las BD ni de los drivers JDBC) todos los niveles de isolation. Por lo general soportar el Grado 1 y 3.

Problemas que se pueden producir:

Dirty Reads:

Una transacción lee datos ingresados por otra transacción que aun no ha hecho commit de esos datos.

Transacción 1	Transacción 2
begin	
SELECT X /* X = 0 */	
X = X + 10 /* X = 10 */	
UPDATE X /* X = 10 en BD pero no se ha hecho COMMIT */	begin
	SELECT X /* X = 10 */
rollback /* X = 0 en BD */	X = X + 10 /* X = 20 */
	UPDATE X /* X = 20 en BD pero no se ha hecho COMMIT */
	commit /* X = 20 en BD y ya se hizo COMMIT */

Non repeatable reads:

Una transacción lee 2 veces una misma tupla. Lee una vez una tupla, pero luego otra transacción modifica ese valor, por lo cual vuelve a leer por segunda vez la misma tupla pero ahora tiene otro valor.

Transacción 1	Transacción 2
begin	
SELECT X /* X = 0 */	**begin**
	X = 20
	UPDATE X /* X = 20 en BD pero no se ha hecho COMMIT */
	commit /* X = 20 en BD. Ahora ya se hizo COMMIT */
SELECT X /* X = 20 */	
…	
commit	

Phantom reads:

Una transacción a leer tuplas que no existían al iniciar la transacción, esto producto que otra transacción agrego nuevos valores

Transacción 1	Transacción 2
begin	
SELECT WHERE condition	begin
	INSERT /* Inserta nuevas filas en BD (aun sin COMMIT), algunas cumpliendo “condition” */
	commit /* Las Inserciones ya están con commit*/
SELECT WHERE condition /* Ahora devuelve más filas */	
…	
commit	

P1.- Ud se encuentra trabajando en una empresa y el gerente le comenta.

"Estoy muy contento porque nuestra empresa va a la vanguardia tecnológica. Gracias a que tenemos un alumno en práctica, hemos desarrollado nuestro propia RDBM (según API estándar de SQL) con lo cual podemos reemplazar nuestra motor de BD actual y ahorrarnos los costos de las licencias.
El sistema ha funcionado a la perfección. Yo he probado en mi computador cada característica, y no he tenido ningún problema, por lo cual he decidido que ya podemos reemplazar la BD".

Dejando de lado el aspecto económico, indique que comentarios le haría sobre la BD (Hint: Considere que el alumno en práctica esta haciendo su práctica 1, y no es alumno destacado).

a.- Explayarse sobre como se administra un sistema de archivos para una BD, e indicarle que es difícil hacerlo eficiente (para inserción, validación de tipos, búsqueda de datos, modificación etc) , por lo cual es posible que este sistema no sea precisamente el más óptimo ni seguro.
b.- Cualquier sistema que administre datos debe cumplir con el requisito ACID. Por lo cual propondría hacer una copia completa del sistema, y realizar test para cada característica ACID
c.- Que la RDBM funcione correctamente en su computador no es mucha gracia. Lo importante es que funcione correctamente cuando hay transacciones concurrentes.

P2.- Producto de sus comentarios, el Gerente quiere demostrarle que esta BD cumple con la característica ACID

i.- "*Esta BD cumple con la característica de ATOMICIDAD. Por ejemplo, tengo una tabla telefonos(id_empleado, numero_telefono) en donde se guardan los teléfonos de los empleados. He realizado inserciones de datos como por ejemplo:*

INSERT INTO teléfonos VALUES(10, 022344565);
INSERT INTO teléfonos VALUES(10, 095685489);

y cada inserción se ha realizado exitosamente de manera atómica."

Discuta con el gerente el significado de la ATOMICIDAD y proponga algún otro ejemplo para medir esta característica.

Atomicidad no se refiere a ingresar un solo dato, sino que se refiere a que cada transacción debe realizarse completamente o sino no debe hacerse nada. Una prueba podría ser ingresar 1.000.000 de registros del tipo telefono con id y numero, y luego hacer update a cero el telefono y presionar reset mientras ocurre esta consulta (si esto no toma el tiempo suficiente como para poder presionar reset, se aumenta el número de registros hasta que se pueda realizar el experimento). Al volver la BD no debería haber ningún registro modificado.

ii.- ***"Esta BD cumple con la característica de CONCISTENCIA. De hecho el campo numero_telefono de la tabla teléfonos esta definido como NOT NULL, y si yo ingreso una tupla sin el telefono, la BD no me deja ingresar el valor".***

Discuta con el gerente el significado de la CONSISTENCIA y proponga algún otro ejemplo para medir esta característica.

Eso es correcto, la característica de consistencia nos asegura que no habrán datos que rompan alguna restricción impuesta en un constraint. Sin embargo, no es solo cada vez que se inserta,

también debe asegurarla en cualquier escenario, como por ejemplo transacciones incompletas, o transacciones concurrentes que produzcan quiebres en las reglas.

Por ejemplo, asumamos que existe una tabla Empleado con un campo id. El campo id_empleado de la tabla telefono es una foreing_key del campo id de la tabla Empleado, y esta seteado ON DELETE CASCADE).
Creo un empleado y 1.000.000 o más registros en la tabla telefono para este empleado. Hago un DELETE de este empleado en la tabla EMPLEADO (y un COMMIT). Por estar seteado ON DELETE CASCADE, todos sus teléfonos deberian ser borrados. Mientras ocurre esta instrucción SQL, presiono RESET.

iii.- "*Mire, he lanzado 2 threads que modifican el mismo dato, y no tuve ningún problema de concurrencia. Podemos entonces decir que al menos esta BD cumple con la característica de ISOLATION".* Comente sobre este resultado y si realmente esto nos asegura el nivel deseado de ISOLATION, además de algunos problemas que podría ocurrir.

El nivel de isolation implica que no habrá problemas si 2 transacciones quieren modificar datos al mismo tiempo. Sin embargo hay muchas más cosas que analizar.

- Si todas las transacciones se realiza de manera secuencial, la BD sería muy lenta.
- Si se paralelizan y se comporta correctamente cuando se modifican datos concurrentes, igual pueden producirse problemas del tipo.
  - Dirty read
  - Non repeatable reads
  - Phantom reads.

iv.- ***"Mire acá, después de cada instrucción SQL que he realizado, los datos quedan. De hecho mire, voy a reiniciar mi PC, y luego le mostraré que todos los datos están. Por lo tanto cumple con la característica de DURABILIDAD."***
Comente sobre el significado de esta característica, sobre escenarios sobre los cuales debe cumplirse, y ejemplos que podrían ayudar a testearla.

Durabilidad se refiere a que la BD debe siempre quedar en el estado del último COMMIT hecho, independiente de cualquier escenario, no solo en un reinicio/apagado controlado, sino en los siguientes escenarios:

- Fallas de hardware
- Fallas de software
- Cortes de luz, entre otros.

P3.- El gerente para demostrar que su BD es una buena idea, ahora busca demostrar que la BD actual es muy mala.

"Observe el funcionamiento erróneo de la BD que usamos actualmente. Realicé inserción masiva de datos acerca de las horas trabajadas por los empleados (un registro por dia), y aunque aún no he hecho ningún COMMIT, puedo ver los valores. He logrado solucionar esto cambiado en todas las transacciones los niveles de Isolation al nivel máximo, pero esto me ha producido que el sistema funcione extremadamente lento". **Comente**

El que mientras aun no haya realizado COMMIT se puedan ver los valores, dependerá del nivel de isolation. Esto puede producir errores como DIRTY READ, NON REPEATABLE READ y PHANTOM READS. Esto puede ser o no un error dependiendo del contexto.

Por ejemplo, si al gerente le interesa saber la cantidad de horas que trabajó un determinado empleado en una semana, permitir un DIRTY READ podría ser un error. Imaginemos que para ese empleado, se están ingresando los registros con la cantidad de horas del lunes, martes, miércoles, y

en ese momento (antes que se inserten las del jueves y viernes) el gerente hace una consulta para ver cuantas horas trabajo en la semana. Entonces el gerente pensaría que ese empleado ha trabajado muchas menos horas que las que realmente trabajo.
Esto se corrige cambiando el nivel de isolation para evitar que se produzcan estos errores. Sin embargo, esto puede resultar mucho mas caro en tiempo y carga para la BD.
Como otro ejemplo, si al gerente le interesara saber el promedio de horas trabajadas por los empleados, un DIRTY READ podría no ser un error, pues en un promedio al gerente no le interesa que sea exacto, por lo cual si algunas tuplas aun no han sido ingresadas el valor no se verá fuertemente afectado.
Por lo tanto, la solución no es ni dejar todo en un nivel de isolation máximo para evitar los problemas mencionados antes, ni dejar los niveles al mínimo para asegurar que la BD funcione lo más rápidamente posible. Lo que hay que hacer es equilibrar los niveles de acuerdo a las necesidades de nuestro sistema.

7.- El gerente encontró que la BD actual no cumple con todos los niveles de isolation, por lo que dice que no cumple el test ACID ¿es eso correcto? (no haga caso al auxiliar cuando le dice que las BD deben cumplir con todos los niveles de isolation).

Las BD (ni los Driver) deben cumplir con todos los niveles de isolation. Sólo deben asegurar que no se produzcan problemas al intentar modificar el mismo dato.