Base de Datos. 1C2009.
Clase práctica de Transacciones
Docente: Alejandro Eidelsztein

RECUPERABILIDAD EN BASES DE DATOS

1. LECTURA ENTRE TRANSACCIONES (*Ti READ FROM Tj*):

Decimos que *Ti lee de Tj* en H si Tj es la transacción que última escribió sobre X, pero no abortó, al tiempo que Ti lee X.

O dicho en otra forma más rigurosa, si:

1. Wj(X) < Ri(X)[1]
2. Aj /< Ri(X)[2]
3. Si hay algún Wk(X) tal que Wj(X) < Wk(X) < Ri(X), entonces Ak < Ri(X)

2. IMAGEN ANTERIOR DE UN WRITE (*BEFORE IMAGE*):

La *imagen anterior* de una operación Write(X,val) es el valor que tenía X justo antes de esta operación.
Podemos asumir que el SGBD implementa el abort restaurando las imágenes anteriores de todos los writes de una transacción.

3. EL PROBLEMA DE LA RECUPERABILIDAD:

Veamos algunos ejemplos:

Ejemplo 1:

Sea la siguiente historia:

H1= Write1(X,2); Read2(X); Write2(Y,3); Commit2.

Supongamos que inicialmente los item X e Y tienen un valor igual a 1.
Ahora supongamos que T1 aborta –y por lo tanto debería hacer un rollback y volver X al valor anterior- y entonces luego T2 debería también abortar y hacer un rollback –porque leyó un valor sucio que le dejó T1- pero si lo hacemos estaríamos violando la semántica del commit y esto trae confusión.
Llegamos a una situación en el que estado consistente anterior de la BD es *irrecuperable*.
Para evitar esta situación deberíamos demorar el commit de T2.

Ejemplo 2:

Sea H2= Write1(X,2); Read2(X); Write2(Y,3); Abort1.

Supongamos un caso similar al anterior pero donde T1 abortó y por lo tanto todavía podemos recuperar el estado consistente anterior abortando T2.
Pero sin embargo esto nos puede llevar a la situación no deseada de *aborts en cascada*.
Para evitar esta situación deberíamos demorar cada Read(X) hasta que los Ti que previamente hayan hecho un Write(X,val) hayan abortado o commiteado.

Ejemplo 3:

Sea H3= Write1(X,2); Write2(X,3); Abort1; Abort2

Supongamos que inicialmente el item X tiene un valor igual a 1.
Aquí vemos que la imagen anterior de Write2(X,3) es 2, escrito por T1.
Sin embargo el valor de X, después de que Write2(X,3) es deshecho, debería ser 1 que es el valor inicial de X, dado que ambos updates de X fueron abortados -como si no se hubieran ejecutado ninguna de las dos transacciones-

[1] p < q denota que la operación p precede a q en el orden de ejecución.

[2] p /<q denota que la operación p no precede a q en el orden de ejecución.

Si embargo aunque el estado anterior todavía podría recuperarse, dado que no hubo commit y todo puede deshacerse, igualmente hemos llegado a una situación de confusión. El problema es que dejó de funcionar la implementación del abort (como restauración de las imágenes anteriores de los writes de una transacción)
Podemos evitar este problema pidiendo que la ejecución de un Write(X,val) sea demorado hasta que la transacción que previamente escribió X hayan commiteado o abortado.
Si pedimos lo mismo con respecto al Read(X) decimos que tenemos una *ejecución estricta*.

4. CLASIFICACIÓN DE HISTORIAS SEGÚN RECUPERABILIDAD:

Ahora veamos las definiciones:

4.1. Historias Recuperables:

Decimos que H es *recuperable* (RC), si cada transacción commitea después del commitment de todas las transacciones (otras que si misma) de las cuales lee.

O en forma equivalente:

Siempre que Ti lee de Tj (i≠j) en H y Ci $\in$ H y Cj < Ci

4.2. Historias que evitan aborts en cascada:

Decimos que H *evita aborts en cascada* (avoids cascading aborts) (ACA), si cada transacción puede leer solamente aquellos valores que fueron escritos por transacciones que commiteadas (o por si misma)

O en forma equivalente:

Siempre que Ti lee X de Tj (i≠j) en H y Ci $\in$ H y Cj < Ri(X)

4.3. Historias estrictas:

Decimos que H es estricta (ST), si ningún item X puede ser leído o sobrescrito hasta que la transacción que previamente escribió X haya finalizado abortando o commiteando.

O en forma equivalente:

Siempre que Wj(X) < Oi(X) (i≠j), o Aj < Oi(X) o Cj < Oi(X), donde Oi(X) es Ri(X) o Wi(X)

Ejemplo 4:

Dados:

T1= W(X) W(Y) W(Z) C
T2= R(U) W(X) R(Y) W(Y) C

H7 = W1(X) W1(Y) R2(U) W2(X) R2(Y) W2(Y) **C2** W1(Z) **C1**
H8 = W1(X) W1(Y) R2(U) W2(X) R2(Y) W2(Y) W1(Z) **C1** **C2**
H9 = W1(X) W1(Y) R2(U) W2(X) W1(Z) **C1** R2(Y) W2(Y) **C2**
H10= W1(X) W1(Y) R2(U) W1(Z) **C1** W2(X) R2(Y) W2(Y) **C2**

Vemos que:

H7 no es RC porque T2 lee Y de T1 pero C2 < C1.

H8 es RC pero no es ACA porque T2 lee Y de T1 antes que T1 commitee.

H9 es ACA pero no es ST porque T2 sobreescribe el valor de X escrito por T1 antes que T1 termine.

H10 es ST.

5. TEOREMA DE RECUPERABILIDAD:

ST $\subset$ ACA $\subset$ RC.

Este teorema nos dice que las propiedades de ST son más restrictivas que las de ACA y que las de esta son a su vez es más que restrictivas que las de RC.

6. RECUPERABILIDAD y SERIALIZABILIDAD:

El concepto de recuperabilidad es ortogonal al concepto de serializabilidad, o sea que una historia H puede ser no RC, RC, ACA o ST y a la vez ser SR o no SR.
El conjunto SR intersecta los conjuntos RC, ACA y ST pero es incomparable a cada uno de ellos.
Las historias seriales son ST y SR.

Ejemplo 5:

Dadas las mismas historias del ejemplo anterior, vemos que:

SG(H7) = SG(H8) = SG(H9) = SG(H10) = (T1) ---------- X,Y ---------- > (T2)

No hay ciclos y por lo tanto son todas SR.

En la siguiente figura podemos ver el diagrama de Venn correspondiente:

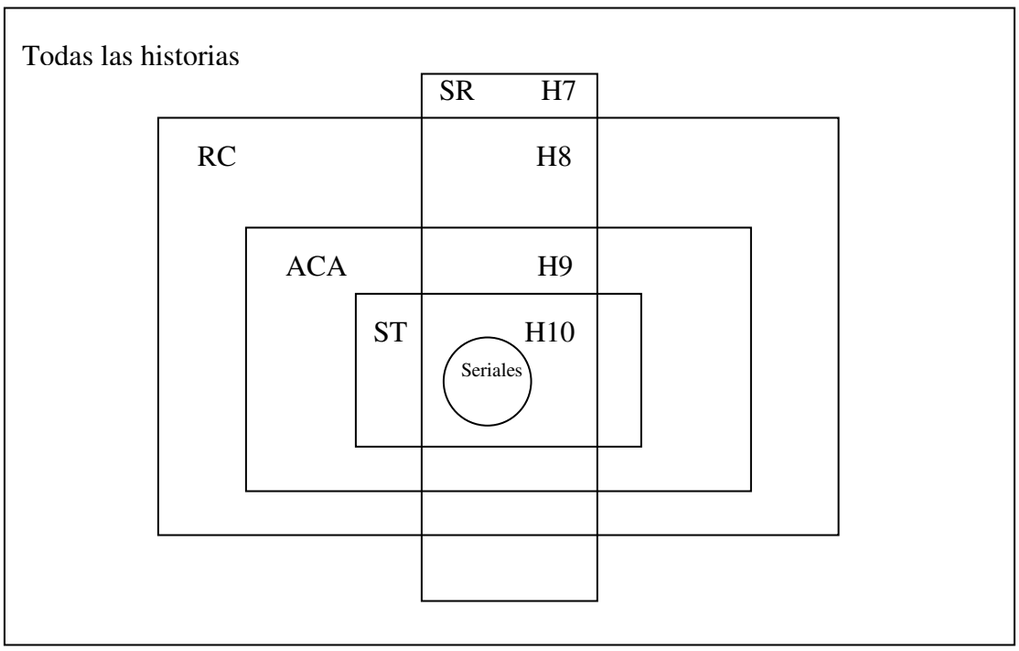

Figura 1.

Ejemplo 6:

Dados:

T1= R(X) W(Y) C
T2= W(Y) W(X) C

H= R1(X) W2(Y) W2(X) C2 W1(Y) C1

Hacemos el SG(H) : (T1) ---------- X ---------- > (T2)
< ---------- Y ----------

y vemos que H no es SR aunque es ST.

7. LOCKING Y RECUPERABILIDAD:

Hay una variante del protocolo 2PL que además de serializabilidad garantiza recuperabilidad.

7.1 PROTOCOLO 2PL ESTRICTO (STRICT 2PL)

T cumple con *2PL estricto* si cumple con 2PL y además no libera ninguno de sus locks exclusivos (WriteLocks) hasta después de commitear o abortar.

Observamos que:

Este protocolo garantiza historias estrictas (ST) con respecto a recuperabilidad, y serializables (SR) con respecto a serializabilidad, o sea que *todo H que cumpla con 2PL estricto (todas sus Ti son 2PL estrictas) es ST y SR.*

Sin embargo, 2PL estricto no garantiza que estemos libres de *deadlocks*.

Nota del docente:

Algunos de los ejemplos de este apunte fueron tomados de los siguientes libros:

"Concurrency Control and Recovery in Database Systems", de Philip A. Bernstein, Vassos Hadzilacos y Nathan Goodman, 1987.

"Introducción a las Bases de Datos Relacionales", de Alberto Mendelzon y Juan Ale, 2000.

"Principles of Database and Knowledge-Base Systems", Volume I, de Jeffrey Ullman, 1988.