

Una técnica de baja tecnología para el aprendizaje de control de concurrencia en bancos de datos

Vinícius Medina Kern

UNIVALI, Cursos de Ciência e Engenharia de Computação

Rodovia SC 407, Km 4; 88122-00; São José-SC, Brasil

Teléfono +55 (48) 281-1500, Fax +55 (48) 281-1506

kern@eps.ufsc.br, <http://www.eps.ufsc.br/~kern>, <http://www.sj.univali.br/~kern>

INDEXADORES

Control de concurrencia, procesamiento de transacciones, juegos educativos, gerencia de banco de datos, constructivismo, aprendizaje

RESUMEN

Bancos de Datos es una disciplina fundamental en cursos de Computación. Las técnicas de control del acceso concurrente a bancos de datos están entre los tópicos más complejos y importantes de esa disciplina. Este artículo presenta una técnica de baja tecnología para el aprendizaje de control de concurrencia en bancos de datos. Primeramente son expuestos los conceptos sobre transacciones, escalonamientos y protocolos de control de concurrencia, con énfasis en las propiedades de recuperabilidad y serializabilidad de los escalonamientos. A continuación, un juego de carteles para simular el control de concurrencia en un sistema de gerencia de banco de datos es presentado. Los resultados del juego y sus consecuencias son comentados.

INTRODUCCIÓN

La disciplina de Bancos de Datos es fundamental en cursos universitarios de Computación y incluye diversos asuntos, entre ellos: proyecto, lenguajes de consulta, y técnicas de gerencia de bancos de datos. Los asuntos son extensos y complejos; los alumnos ni siempre consideran los libros suficientemente accesibles y claros. En nuestras clases usamos libros tradicionalmente adoptados y considerados como referencia: [1] [2] [3].

Actualmente, las actividades de proyecto y escrita de comandos de consulta son poco automatizadas, al paso que actividades de gerencia son llevadas a cabo con sistemas de gerencia de bancos de datos, con mucha automatización. El mercado de trabajo, en consecuencia, ofrece mucho más oportunidades para proyectistas y desarrolladores de programas aplicativos de banco de datos do que para desarrolladores de rutinas de gerencia. Sin embargo, la gerencia es parte importante de la teoría y práctica de bancos de datos.

Este artículo trata de presentar conceptos y un juego para facilitar el aprendizaje de una técnica de gerencia de bancos de datos: el control del acceso concurrente. El juego permite a los estudiantes simular el control de concurrencia usando instrumental de baja tecnología: carteles que representan operaciones de acceso a banco de datos. Así, los estudiantes llegan a ver el proceso como algo concreto, tangible.

La próxima sección presenta los conceptos fundamentales sobre transacciones y control de concurrencia. Los protocolos de control de concurrencia por bloqueos y ordenación de timestamps son descritos enseguida, juntamente con un juego de carteles para su simulación. Finalmente, los resultados de la aplicación de esta técnica son discutidos, y también las consecuencias del juego en el aprendizaje y la conexión del asunto con otras disciplinas de Computación.

TRANSACCIONES Y ESCALONAMIENTOS

El juego de control de concurrencia es practicado en clase por varios estudiantes, pero la práctica es precedida por el estudio de conceptos importantes. Una **transacción** es una unidad lógica de procesamiento de banco de datos que tiene las propiedades "ACID": **atomicidad**, **consistencia**,

aislamiento (isolation), y **durabilidad**. Eso significa que cada transacción tiene suceso o fracaso total (en este caso se debe deshacer grabaciones hechas por la transacción en el banco de datos), que lleva el banco de un estado consistente a otro estado consistente, que cada transacción no sabe lo que hacen las otras que ocasionalmente son procesadas al mismo tiempo, y que, una vez confirmada, el efecto de una transacción no puede ser deshecho.

La figura 1 presenta cuatro **escalonamientos** (ordenaciones de operaciones elementares) posibles para las transacciones T1 y T2, donde el tiempo corre desde arriba para abajo. T1 es una transferencia de fondos (50 unidades monetarias) entre cuentas X y Y. T2 es una retirada de 100 unidades monetarias de la cuenta X. Solo las operaciones de banco de datos son importantes para el control del acceso concurrente (leer y grabar ítem, confirmar transacción - COMMIT - o fallar - ROLLBACK), pero fueran agregadas operaciones aritméticas para ilustrar el significado de las transacciones.

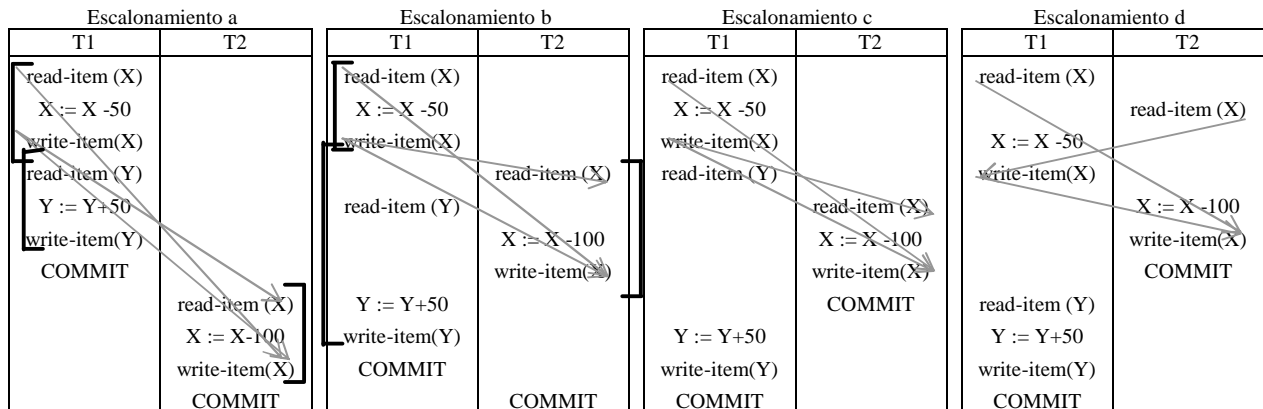


Figura 1. Varios escalonamientos alternativos para las transacciones T1 y T2: (a) serial y recuperable, (b) serializable y recuperable, (c) serializable y no recuperable, y (d) no-serializable

Una característica fundamental de los escalonamientos correctos es la **recuperabilidad** [4]. Un escalonamiento es recuperable si una falla puede ser deshecha, sin causar una inconsistencia definitiva en el banco de datos. Si una transacción debe ser deshecha pero ya está confirmada, entonces el escalonamiento no es recuperable. En la figura 1, el escalonamiento (c) no es recuperable, pues si T1 viniera a fallar después de T2 haber sido confirmada, la operación `write-item(X)` en T1 sería deshecha y, por consecuencia, T2 debería fallar también, pues lee X después de la citada grabación. Lamentablemente, T2 ya fue confirmada – la inconsistencia es definitiva (durable).

Otra característica importante de los escalonamientos es la **serializabilidad** [5]. La idea parte del principio de que un escalonamiento serial (sin intercalación de operaciones de transacciones distintas, como (a) en la figura 1) no tiene problemas de acceso concurrente. Un modo de garantizar la serializabilidad es asegurar que el escalonamiento es **conflicto-equivalente** a algún escalonamiento serial, es decir: la orden de ejecución de **operaciones conflictivas** es la misma que en un escalonamiento serial para las mismas transacciones.

Dos operaciones son conflictivas si son de transacciones distintas, si leen o graban el mismo ítem del banco de datos, y si por lo menos una de ellas es de grabación. Los pares de operaciones conflictivas son representadas por flechas en la figura 1. Los escalonamientos (b) y (c) son no-seriales pero conflicto-equivalentes al escalonamiento serial (a), por eso dichos **serializables**. El escalonamiento (d), al contrario, no es conflicto-equivalente a ningún de los escalonamientos seriales para T1 y T2, cuales sean, T1? T2 y T2? T1.

En resumen, solamente (a) y (b) en la figura 1 garantizan la integridad del banco de datos, pues resultan recuperables y serializables. El resultado de (c) es correcto si T1 y T2 son confirmadas, pero hay el riesgo de inconsistencia si T1 falla después de la confirmación de T2. Las operaciones COMMIT en (d) en realidad no pueden existir – T2 o ambas transacciones deben sufrir ROLLBACK (falla).

El control del acceso concurrente de varias transacciones simultaneas se hace a través de protocolos de control de concurrencia. El escalonamiento que va sendo producido es registrado en el **diario** o **histórico** del sistema, un documento secuencial grabado en disco duro.

Una vez que los estudiantes comprendan estos principios y conceptos, les falta aún comprender **como** hace un sistema de gerencia de banco de datos para escalonar múltiples operaciones de varias transacciones concurrentes. Las técnicas de control de concurrencia más populares usan **bloqueos**. En este artículo describimos también un protocolo de control del acceso concurrente que no usa bloqueos, sino ordenación de timestamps (que hace comparaciones basadas en valores de tiempo).

Un bloqueo es la reserva de un ítem de banco de datos para acceso por una transacción. En la figura 1, (a) y (b) tienen ilustrados por corchetes los bloqueos y desbloqueos de los ítems. Todo acceso a ítem de banco de datos demanda bloqueo, que puede ser compartido por varias transacciones si visa lectura (**read-lock**). Accesos para grabación, o para lectura y grabación, demandan bloqueo exclusivo (**write-lock**, concedido a una sola transacción, lo que hace que otras tengan que esperar para lograr el acceso). Para garantizar la serializabilidad, el **protocolo de dos fases** [6] hay que ser respetado: Hay que obtener todos los bloqueos antes del primer desbloqueo en cada transacción (según los corchetes en la figura 1).

Una variación importante del protocolo recién explicado es el **protocolo de dos fases conservador**. En ello, se exige que una transacción solo pueda procesar su primer operación de leer o grabar si antes logra reservar todos los bloqueos necesarios en la transacción. De esta forma se evita la situación de **deadlock**, en la cual dos o más transacciones no pueden seguir porque necesitan de bloqueos que otra transacción detiene, y la liberación no puede ocurrir para no violar el protocolo de dos fases. Hay ejemplos de deadlock en la sección Resultados, más adelante en este artículo. Un deadlock solo puede ser solucionado si se aborta alguna transacción [2].

En el **protocolo de 2 fases estricto**, una transacción no libera sus bloqueos hasta que sea confirmada o abortada. Este criterio garantiza también la recuperabilidad del escalonamiento – en realidad, una recuperabilidad más restrictiva que la exigencia básica de confirmar lecturas solamente después de confirmar grabaciones anteriores.

Aparte de los varios protocolos de bloqueos, hay dos modalidades posibles. En el bloqueo **múltiple**, recién referido, varias transacciones pueden compartir bloqueos para lectura, pero una sola transacción puede detener un bloqueo para grabación. Una modalidad más simple, el bloqueo **binario**, no hace esa distinción, y por eso es más restrictiva.

El control del acceso concurrente puede ser hecho también sin el uso de bloqueos. En nuestra experiencia, los alumnos estudian también el **protocolo de ordenación de timestamps**. Un timestamp es un identificador atribuido a cada transacción, y refleja el momento de sumisión de la transacción, así que, cuanto mayor el timestamp, más nueva es la transacción.

Esta técnica produce escalonamientos conflicto-equivalentes al escalonamiento serial que coincide con la orden en la cual las transacciones fueron iniciadas. Para tanto, mantiene variables `read_TS` y `write_TS` para cada ítem del banco de datos, que significan: timestamp de la transacción más joven que ha leído el ítem, y timestamp de la transacción más joven que ha grabado el ítem.

Para cada operación `read-item` el algoritmo testa si las operaciones conflictivas con esa lectura siguen la orden correcta (el timestamp de la transacción debe ser mayor o igual a `write_TS`). Para una operación `write-item` el algoritmo testa ambos `read_TS` y `write_TS`, pues una grabación representa conflicto con lecturas y otras grabaciones del mismo ítem. Si algún teste falla, la transacción que intenta leer o grabar es abortada.

EL JUEGO DE CONTROL DE CONCURRENCIA

El juego de control de concurrencia usa carteles para representar operaciones de `iniciar transacción`, `COMMIT` y `ROLLBACK`, lecturas (`read-item`), grabaciones (`write-item`) y deshacer de grabaciones (`undo write`) de determinados ítems (X, Y, y Z, por ejemplo). Esos carteles son usados siempre que se juega, pues representan las operaciones grabadas en el diario o histórico del sistema.

Hay otros carteles que se usan conforme el protocolo de control de concurrencia adoptado. No son operaciones de banco de datos, ni tampoco quedan grabadas en el diario, pero son usados en el juego con propósito ilustrativo. Cuando se escalona por ordenación de timestamps, pequeños carteles con números marcan las actualizaciones de los valores `read_TS` y `write_TS` de los diversos ítems de banco de datos. Cuando se escalona usando bloqueos, el juego se hace registrando cada bloqueo o desbloqueo.

El primer corchete de la figura 1 (a), por ejemplo, corresponde a carteles `write-lock(X)` y `unlock(X)`. La representación por carteles tiene el intento de ilustrar claramente donde se obtiene o libera cada bloqueo.

Varios estudiantes pueden jugar. En nuestra experiencia el número de jugadores ha variado de 3 a 10. Hay una serie de funciones que pueden ser distribuidas. Un **cartel de control** simula la atención momentánea de la unidad de procesamiento de la computadora. Dos o más transacciones son disparadas, por hipótesis, al mismo tiempo, pero necesariamente en alguna orden establecida. En cada transacción hay un cartel `iniciar transacción` y uno o más carteles de lectura y/o grabación. Usualmente se escogen transacciones con muchas operaciones conflictivas.

El estudiante responsable por la política de **timesharing** (división de tiempo) recibe el cartel de control y hace el primer movimiento: Dedicar tiempo de la unidad de procesamiento a la transacción que tiene la vez, pasando el cartel de control al estudiante que hace el **servicio de operación** (o sea, que presenta la próxima operación a escalonar en esta transacción). Ese pasa el cartel al responsable por aplicar el criterio del **protocolo de control de concurrencia**, para que decida cual bloqueo es necesario, y para que verifique si la transacción lo tiene o hay que obtenerlo.

Esa decisión depende del protocolo de control de concurrencia adoptado – si se usa el **protocolo de dos fases** básico o uno de sus variantes más restrictivas. Si no se puede obtener un bloqueo y al mismo tiempo respetar el criterio de dos fases, la transacción pierde la vez y el control retorna a la función `timesharing`. Cuando se usa el protocolo de ordenación de timestamps, la decisión es siempre por escalonar la operación o abortar la transacción.

Si es posible obtener el bloqueo o la transacción ya lo tiene, entonces la operación de lectura o grabación es escalonada y se puede hacer una serie de preguntas: ¿Hay que seguir escalonando en la misma transacción o llamar el `timesharing`? ¿Se puede desbloquear algo? ¿Se puede confirmar la transacción? ¿Hay motivo para postergar un desbloqueo si la transacción ya hizo los accesos a un ítem? ¿Hay motivo para postergar un `COMMIT` si la transacción ya hizo todas las lecturas y grabaciones?

Para responder a estas preguntas los estudiantes necesitan dominar plenamente los conceptos de propiedades de las transacciones, recuperabilidad, y serializabilidad. Además, deben pensar sobre la performance del sistema conforme se escogen políticas de control de concurrencia, criterio de recuperabilidad, y tamaño del fragmento de tiempo concedido a cada turno a cada transacción.

RESULTADOS

Al empezar el juego, la primera pregunta importante que surge es: ¿Cuál es el tamaño adecuado de la porción de tiempo concedida a cada transacción? Si la porción es muy larga, los escalonamientos serán seriales, lo que inhibe la concurrencia y es malo para la performance. Si son muy cortos, quizás la concurrencia también sea truncada – tal vez dejando de confirmar una transacción y forzando la demora en cascada de otras transacciones. Es necesario establecer una **política de timesharing**.

En nuestra experiencia usualmente adoptamos el criterio de que cada transacción puede escalonar, a cada vez, **una** operación de lectura o grabación. Juzgamos que es una buena política para una simulación, pues operaciones de lectura y grabación de banco de datos son las más costosas, y posiblemente operaciones de iniciar, confirmar o cancelar transacciones son relativamente mucho más baratas – y por eso, cuanto antes sean escalonadas, mejor será el nivel de concurrencia.

Con esta política establecida, y conociendo los conceptos y protocolos de control de concurrencia, es posible para los alumnos decidir, a cada momento, cual es la acción adecuada para producir un escalonamiento. La figura 2 ilustra un escalonamiento recuperable producido usando el protocolo por ordenación de timestamps.

Es interesante notar la preservación de la propiedad de recuperabilidad en T2. Si el `COMMIT` fuera grabado en el histórico luego después de la lectura, el posterior `ROLLBACK` de T1, que demanda el `UNDO` (deshacer) de la grabación de Y, tornaría la lectura en T2 inválida. Sin embargo, si T2 hubiera sido confirmada (`COMMIT`), el usuario recibiría una respuesta con un valor posiblemente incorrecto de Y. Entonces, T2 es abortado en cascada en consecuencia del `ROLLBACK` de T1, y una nueva transacción T6 es iniciada para hacer el trabajo que T2 no pudo hacer. El `ROLLBACK` es transparente para el usuario.

La figura 2 muestra la construcción del escalonamiento de cuatro transacciones posicionando carteles de `inicia transacción`, lecturas (`read-item`) y grabaciones (`write-item`), confirmaciones (`COMMIT`) y cancelamientos (`ROLLBACK`). En ese caso también fue usado un cartel para deshacer una grabación cancelada (`undo write (Y)`), respetando la propiedad de la atomicidad de la transacción T1.

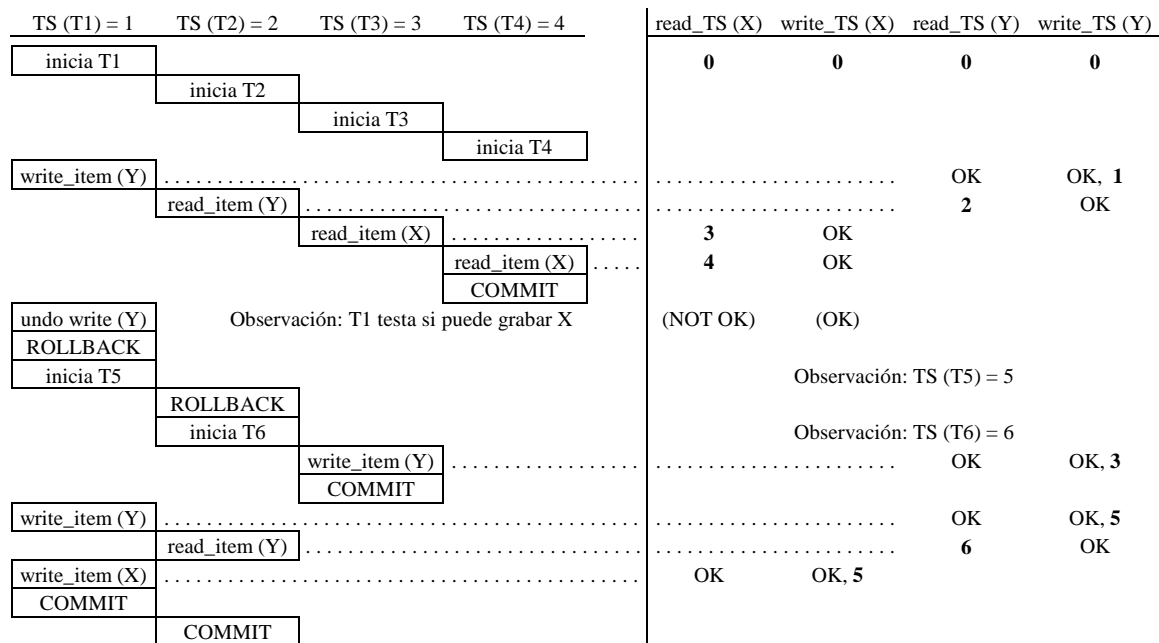


Figura 2. Escalonamiento recuperable por ordenación de timestamps para las transacciones simultaneas T1, T2, T3 y T4 (y sucesoras T5 y T6)

Los números a la derecha también son dispuestos en la mesa de juego como carteles que registran la evolución de los valores `read_TS` y `write_TS`. Las anotaciones “OK” y “NOT OK” no son carteles, solamente ilustran el teste hecho por los alumnos en cada momento. Los carteles a la izquierda son una representación del histórico del sistema.

La figura 3 muestra cuatro escalonamientos por técnicas de bloqueo para las mismas transacciones de la figura 2. En estos experimentos los alumnos usan opciones variadas de modalidad de bloqueo (binario o múltiplo), protocolo (2 fases básico, conservador o estricto) y criterio de recuperabilidad (recuperable, que evita abortar en cascada, o estricto). Los carteles más oscuros representan bloqueos y desbloqueos – operaciones de memoria, que no son registradas en el histórico.

En el experimento con bloqueo múltiplo en 2 fases conservador, índice (a) en la figura 3, es interesante observar la postergación del `COMMIT` de T2 para preservar el carácter recuperable del escalonamiento. T1 graba Y antes de la lectura en T2, y por eso la confirmación de T1 debe ocurrir antes de la confirmación de T2. Si T1 falla, T2 debe ser abortado en cascada (en consecuencia).

En el bloqueo binario en 2 fases conservador, figura 3 (b), un criterio de recuperabilidad más restrictivo fue usado para asegurar que ninguna transacción venga a fallar en cascada. El impedimento de leer ítem grabado y no confirmado garantiza que no hay fallas en cascada. T2 solo puede leer Y después de T1 haber sido confirmada.

Ambos escalonamientos 3 (a) y (b) usan el protocolo de 2 fases conservador, que es deadlock-free (libre de deadlock). Si el protocolo usado es de 2 fases básico (no conservador), es posible que dos o más transacciones lleguen a un conflicto entre obtener y liberar bloqueos. Eso pasa en los casos de la figura 3 (c) y (d).

Hay que usar un protocolo para la prevención de situaciones de deadlock si el protocolo de control de concurrencia es susceptible. En el caso de la figura 3 (c), el criterio para decidir cual transacción abortar es comparar dos transacciones en espera y decidir por abortar la más joven – en este caso, T3. Los protocolos wait-die y wound-die [2] usan este criterio.

El criterio usado en el escalonamiento de la figura 3 (d) para elegir cual transacción abortar está basado en timeout (tiempo excesivo): la transacción T1 es la primera a perder tiempo y sigue sin poder procesar sus operaciones. Así, será también la primera a atingir el tiempo-límite de timeout, y por eso es abortada.

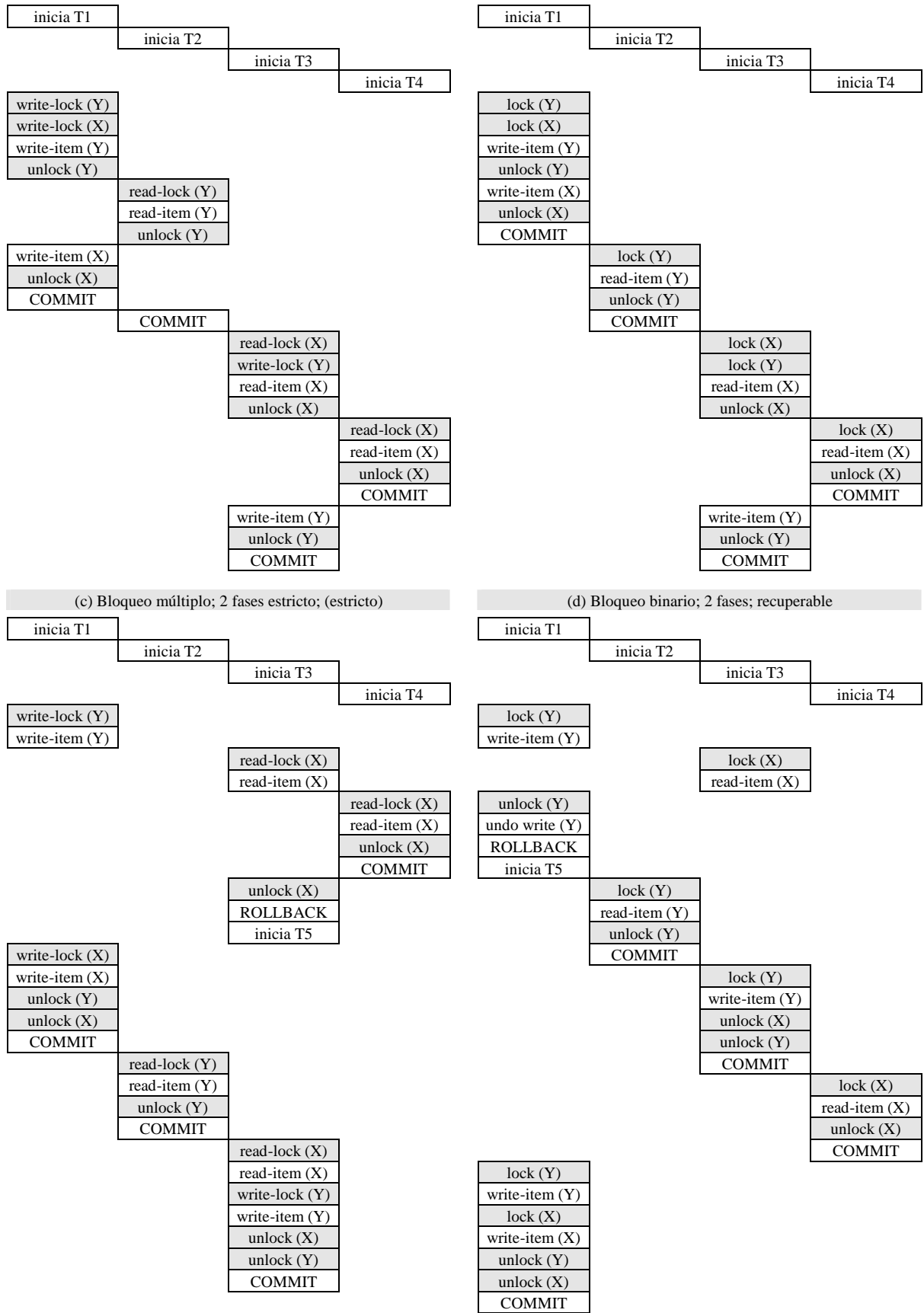


Figura 3. Escalonamientos para las para las mismas transacciones de la figura 2 conforme variadas modalidades de bloqueo, protocolos de control de concurrencia y criterios de recuperabilidad

CONCLUSIÓN

Mientras juegan, los alumnos tienen la oportunidad de observar la dinámica del proceso de escalonamiento (conforme el andar del cartel de control). A cada movimiento, contemplan la composición del histórico del sistema y se preguntan sobre el sentido de cada decisión de escalonamiento. Esa forma de aprender es concordante con las ideas de aprendizaje constructivista de Vygotsky [7] y Piaget [8].

Al comparar varias soluciones para el escalonamiento de un mismo conjunto de transacciones simultáneas, los alumnos pueden preguntarse: ¿Cuál política de escalonamiento es más adecuada para cada circunstancia? ¿Cuales son las ventajas de cada protocolo y cada criterio? ¿Cuál protocolo es “mejor”? De esa forma, usando instrumentos de baja tecnología y ejercitando conceptos vistos en clase, mejoran los chances de que los alumnos lleguen a **pensar** – un presupuesto de la educación superior, pero tantas veces malogrado.

A pesar de la técnica usada ser de baja tecnología, el ejercicio permite a los alumnos comprender, analizar, y posteriormente diseñar y implementar una solución computacional para el problema de control del acceso concurrente. Esa es una de las tareas que cumplen en la disciplina.

La participación en el juego hizo aumentar el interés de los alumnos por el asunto de la gerencia de bancos de datos. Su desempeño mediano también ha mejorado desde la introducción del juego.

Una extensión posible de este trabajo es criar un ambiente computacional para la simulación de las técnicas. Sin embargo, el carácter de baja tecnología y la posibilidad de trabajo en grupo son dos puntos fuertes del abordaje. Es posible, también, invertir más en técnicas de recuperación de fallas. En los ejemplos de la figura 3 se usa un cartel de deshacer grabación. En realidad, la técnica de recuperación de fallas es más compleja do que el ejemplo trasparece, y el juego de carteles podría ser extendido para contemplar más esta técnica de gerencia de banco de datos.

La participación en el juego ha permitido a los estudiantes acercarse más del asunto, profundizar su entendimiento, y mejorar el nivel de aprendizaje. El profesor tiene, mientras coordina el juego, oportunidades para llamar los alumnos al ejercicio de los conocimientos relativos a otras disciplinas, por ejemplo sistemas operativos, lenguajes de programación, algoritmos y estructuras de datos.

REFERENCIAS

- [1] C. J. Date. **An introduction to database systems**, séptima edición. Addison-Wesley, 1999.
- [2] R. Elmasri y S. B. Navathe. **Fundamentals of database systems**. Addison-Wesley, 1994.
- [3] R. Ramakrishnan. **Database management systems**, segunda edición. WCB/McGraw-Hill, 2000.
- [4] D. Lomet y B. Salzberg. “Access method concurrency control with recovery”, **ACM SIGMOD conf. on the management of data**, 1992.
- [5] J. Gray, R. Lorie, y G. Putzulo. “Granularity of locks and degrees of consistency in a shared database”. In: G. Nijssen (ed.), **Modelling in data base management systems**. North-Holland, 1976.
- [6] K. Eswaran, J. Gray, R. Lorie, y I. Traiger. “The notions of consistency and predicate locks in a data base system”, **Communications of the ACM** 19 (11), p. 624-633, 1976.
- [7] L. S. Vygotsky. **A formação social da mente**. São Paulo: Martins Fontes, 1984.
- [8] J. Piaget. **Psicologia da inteligência**. Rio de Janeiro: Zahar, 1977.