Universitat Oberta de Catalunya uoc.edu

Uso de bases de datos

PEC 2: Acceso concurrente a bases de datos

Pregunta 1 (50% puntuación)

Enunciado

Sea un SGBD **sin ningún mecanismo de control de concurrencia**, y suponiendo que se produce el horario que se muestra en la página siguiente (donde R=lectura; RU=lectura con intención de actualización, W=escritura; las acciones se han numerado para facilitar su referencia).

Se pide, argumentando **brevemente** vuestra respuesta:

1. Indicad si hay interferencias en el horario propuesto, cuáles son, entre qué transacciones y para qué gránulos.
2. En caso de haber encontrado interferencias en el horario, indicad cuál es el nivel de aislamiento mínimo del SQL estándar que haría falta para evitar cada una de ellas por separado. ¿Cuál sería el nivel mínimo de aislamiento si las quisiéramos evitar todas simultáneamente?
3. ¿Es recuperable el horario anterior? Indicad cuáles son los horarios seriables equivalentes.
4. Aplicad un mecanismo de control de concurrencia basado en reservas S, X y donde las transacciones trabajan con un nivel de aislamiento de `READ COMMITTED`. ¿Cómo quedaría el horario? ¿Se evitan las interferencias del horario inicial? ¿Cuál es el horario en serie equivalente?
5. Aplicad un mecanismo de control de concurrencia basado reservas S, X y donde las transacciones trabajan con un nivel de aislamiento de `REPEATABLE READ`. ¿Cómo quedaría el horario? ¿Se evitan las interferencias del horario inicial? ¿Hay algún horario en serie equivalente?

Nota: Es posible que en los apartados 4 y 5, como consecuencia de aplicar las reservas indicadas, puedan existir diferentes propuestas de solución igualmente válidas dependiendo del orden de ejecución de las operaciones de las transacciones.

Criterios de evaluación

- *Todas las preguntas tienen el mismo peso.*
- *Las preguntas no contestadas no penalizan.*
- *Se valorará la calidad de la respuesta en relación a los contenidos de los módulos didácticos, y el hecho de no entrar en contradicciones en las explicaciones.*

- *Se tienen que justificar las respuestas para conseguir la puntuación máxima en todos los apartados.*
- *Las respuestas no argumentadas se considerarán como no contestadas.*

#Acc	T1	T2	T3
10		R(B)	
20			RU(B)
30			W(B)
40			RU(C)
50			W(C)
60		R(C)	
70	RU(A)		
80		RU(A)	
90	W(A)		
100		W(A)	
110			RU(E)
120			W(E)
130	R(D)		
140		R(E)	
150			RU(E)
160			W(E)
170	COMMIT		
180		COMMIT	
190			COMMIT
200			
210			

Solución

1. En el horario propuesto encontramos las interferencias

   Actualización perdida entre T1 y T2, gránulo A

   Análisis inconsistente entre T2 y T3, gránulos B y C

   Lectura no confirmada entre T2 y T3, gránulo E

#Acc	T1	T2	T3
10		R(B)	
20			RU(B)
30			W(B)
40			RU(C)
50			W(C)
60		R(C)	
70	RU(A)		
80		RU(A)	
90	W(A)		
100		W(A)	
110			RU(E)
120			W(E)
130	R(D)		
140		R(E)	
150			RU(E)
160			W(E)
170	COMMIT		
180		COMMIT	
190			COMMIT

2. Para evitar las interferencias encontradas, nos hacen falta los niveles mínimos siguientes:

   Actualización perdida `READ UNCOMMITTED`,

   Lectura no confirmada `READ COMMITTED`,

Análisis inconsistente `REPEATABLE READ`

Para evitar todas las interferencias a la vez, nos tendríamos que quedar con el nivel más restrictivo, es decir, `REPEATABLE READ`.

3. El horario propuesto no es recuperable, puesto que, por ejemplo, T2 en la acción 60 lee el gránulo C, previamente escrito por T3 en la acción 50, y T2 acaba su ejecución confirmando antes que T3.

   No hay ningún horario en serie equivalente puesto que el horario propuesto no es seriable al existir interferencias.

4. Con un nivel de aislamiento `READ COMMITTED`, el horario queda de la manera siguiente:

#Acc	T1	T2	T3
10		LOCK(B,S)	
20		R(B)	
30		UNLOCK(B)	
40			LOCK(B,X)
50			RU(B)
60			W(B)
70			LOCK(C,X)
80			RU(C)
90			W(C)
100		LOCK(C,S)	
110	LOCK(A,X)		
120	RU(A)		
130	W(A)		
140			LOCK(E,X)
150			RU(E)
160			W(E)
170	LOCK(D,S)		
180	R(D)		
190	UNLOCK(D)		

200			RU(E)
210			W(E)
220	COMMIT UNLOCK(A)		
230			COMMIT UNLOCK(B) UNLOCK(C) UNLOCK(E)
240		R(C)	
250		UNLOCK(C)	
260		LOCK(A,X)	
270		RU(A)	
280		W(A)	
290		LOCK(E,S)	
300		R(E)	
310		UNLOCK(E)	
320		COMMIT UNLOCK(A)	
330			
340			
350			
360			

En comparación con las interferencias que tenía el horario original podemos ver como con `READ COMMITTED` se evitan todas las interferencias excepto la inferencia de análisis inconsistente. Por lo tanto, no hay ningún horario en serie equivalente.

5. Con un nivel de aislamiento `REPEATABLE READ`, el horario queda tal como se indica a continuación:

#Acc	T1	T2	T3
10		LOCK(B,S)	
20		R(B)	

30			LOCK(B,X)
40		LOCK(C,S)	
50		R(C)	
60	LOCK(A,X)		
70	RU(A)		
80		LOCK(A,X)	
90	W(A)		
100	LOCK(D,S)		
110	R(D)		
120	COMMIT UNLOCK(A) UNLOCK(D)		
130		RU(A)	
140		W(A)	
150		LOCK(E,S)	
160		R(E)	
170		COMMIT UNLOCK(E) UNLOCK(B) UNLOCK(C) UNLOCK(A)	
180			RU(B)
190			W(B)
200			LOCK(C,X)
210			RU(C)
220			W(C)
230			LOCK(E,X)
240			RU(E)
250			W(E)
260			RU(E)
270			W(E)

280			COMMIT UNLOCK(E) UNLOCK(B) UNLOCK(C)
290			

Este horario ha acabado sin abrazos mortales y gracias al nivel de aislamiento aplicado está libre de interferencias. Por lo tanto, existe un horario en serie que da resultados equivalentes al horario obtenido. El horario en serie equivalente sería T1;T2;T3.

Pregunta 2 (30% puntuación)

Enunciado

Dadas las tablas que se muestran a continuación (extraídas de la base de datos de la práctica, con algunas simplificaciones), suponed que queremos ejecutar varias transacciones. Éstas ejecutarán las peticiones siguientes:

T1	T2	T3
SELECT * FROM BAND WHERE style='Heavy';	UPDATE BAND SET style = 'Heavy' WHERE id_band = 3;	SENTENCIA SQL
SELECT * FROM BAND WHERE style='Heavy';	UPDATE BAND SET year_formed = 1980 WHERE id_band = 1;	
COMMIT	COMMIT	COMMIT

BAND				
id_band	**name**	**year_formed**	**style**	**origin**
1	Metallica	1981	Heavy	US
3	Free	1968	Rock	England

Suponiendo que el gránulo es la fila y que el SGBD **no implementa ningún mecanismo de control de concurrencia**, responded a las siguientes preguntas (tenéis que razonar brevemente vuestras respuestas). Cada apartado se tiene que responder de forma independiente, obviando las respuestas dadas en los apartados previos.

1. Proponed, si es posible, un horario que incorpore todas las sentencias SQL de T1 y T2 y que solo contenga una interferencia de tipo lectura no repetible.

2. Proponed, si es posible, un horario que incorpore todas las sentencias SQL de T1 y T2, que no contenga ninguna interferencia, pero que sea no recuperable.

3. Proponed, si es posible, un horario y una sentencia SQL a ejecutar por T3 que cause una interferencia de tipo fantasma entre T1 y T3 (que cause la aparición de un fantasma).

Criterios de evaluación

- *Todas las preguntas tienen el mismo peso.*
- *Las preguntas no contestadas no penalizan.*
- *Se valorará la calidad de la respuesta en relación a los contenidos de los módulos didácticos, y el hecho de no entrar en contradicciones en las explicaciones.*
- *Se tienen que justificar las respuestas para conseguir la puntuación máxima en todos los apartados.*
- *Las respuestas no argumentadas se considerarán como no contestadas.*

Solución

1) A continuación, proponemos un horario por T1 y T2 que verifica las condiciones indicadas en el enunciado:

#acc	T1	T2
10		UPDATE BAND SET style = 'Heavy' WHERE id_band = 3;
20	SELECT * FROM BAND WHERE style='Heavy';	
30		UPDATE BAND SET year_formed = 1980 WHERE id_band = 1;
40	SELECT * FROM BAND WHERE style='Heavy';	
50		COMMIT
60	COMMIT	

2) A continuación, proponemos un horario por T1 y T2 que verifica las condiciones indicadas en el enunciado:

#acc	T1	T2
10		UPDATE BAND SET style = 'Heavy' WHERE id_band = 3;
20		UPDATE BAND SET year_formed = 1980 WHERE id_band = 1;
30	SELECT * FROM BAND WHERE style='Heavy';	
40	SELECT * FROM BAND WHERE style='Heavy';	
50	COMMIT	
60		COMMIT

El horario no presenta interferencias, de hecho, da los mismos resultados que el horario en serie T2; T1, pero no es un horario recuperable. T1 lee datos pendientes de confirmación en las acciones 30 y 40 y T1 acaba su ejecución antes de que T2.

3) A continuación, proponemos un horario por T1 y T3 que verifica las condiciones indicadas en el enunciado:

#acc	T1	T3
10	SELECT * FROM BAND WHERE style='Heavy';	
20		UPDATE BAND SET style = 'Heavy' WHERE id_band = 3;
30	SELECT * FROM BAND WHERE style='Heavy';	
40		COMMIT
50	COMMIT	
60		

La transacción T1, en el instante 10, hace la lectura de las bandas de Heavy que, con los datos del enunciado, es la banda con identificador 1. En el instante 30 T1 hace una lectura donde aparece un fantasma, puesto que aparece la banda identificador 3 que ha actualizado T3 y que no aparecía en la primera lectura.

Otra solución sería hacer un INSERT de una nueva banda de Heavy, que también provoca que aparezca un fantasma en el instante 30.

Pregunta 3 (20% puntuación)

Enunciado

- Utilizando la tabla siguiente:

```
CREATE TABLE employee(
      id SMALLINT PRIMARY KEY,
      name VARCHAR(255));
```

y los siguientes datos iniciales:

```
INSERT INTO employee VALUES( 1, 'Pere');
```

Explicad por qué PostgreSQL no bloquea T1 a la hora de ejecutar el UPDATE.

#acc	T1	T2
10	BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE;	
20		BEGIN TRANSACTION ISOLATION LEVEL SERIALIZABLE;
30		SELECT * FROM employee WHERE id = 1
40	UPDATE employee SET name = 'Carlos' WHERE id = 1	
50		COMMIT
60	COMMIT	

70		

- Indicad si las afirmaciones siguientes son ciertas o falsas. Razonad brevemente vuestra respuesta.

    1) Una transacción con nivel de aislamiento READ COMMITTED no puede cumplir el PR2F estricto.

    2) Utilizando PostgreSQL, una transacción con nivel de aislamiento READ UNCOMMITTED puede tener interferencias que se pueden evitar con READ COMMITTED.

Criterios de evaluación

- *Todas las preguntas tienen el mismo peso.*
- *Las preguntas no contestadas no penalizan.*
- *Las respuestas sin argumentación no serán evaluadas.*
- *Se valorará la calidad de la respuesta en relación a los contenidos de los módulos didácticos, y el hecho de no entrar en contradicciones en las explicaciones.*

Solución

- El motivo por el cual T1 no bloquea es porque PostgreSQL internamente utiliza MVCC (*Multiversion concurrency control*). En MVCC, las acciones de lectura (R(G)) nunca generan reservas ni tampoco ven nunca bloqueada la ejecución.

1) **Falso**. Una transacción que sólo escriba cumplirá el PR2F estricto.

2) **Falso.** PostgreSQL internamente utiliza 3 niveles de aislamiento (READ COMMITTED, REPEATABLE READ y SERIALIZABLE). Por lo tanto, cuando indicamos READ UNCOMMITTED, PostgreSQL lo transforma internamente a READ COMMITTED.