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 siguiente página (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 sobre qué gránulos.
2. En el 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 y X donde las transacciones trabajan con un nivel de aislamiento de `READ COMMITTED`. ¿Cómo acabaría 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 y X 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	RU(A)		
20			R(E)
30			RU(A)
40			W(A)
50		R(F)	
60	W(A)		
70		RU(E)	
80		W(E)	
90			RU(C)
100			W(C)
110	R(B)		
120		RU(B)	
130		W(B)	
140			R(E)
150		RU(D)	
160		W(D)	
170	R(D)		
180	COMMIT		
190		COMMIT	
200			COMMIT

Solución

1. En el horario propuesto encontramos las siguientes interferencias:

   Actualización perdida entre T1 y T3, gránulo A

   Lectura no repetible entre T2 y T3, gránulo E

   Análisis inconsistente entre T1 y T2, gránulos B y D

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

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

   Actualización perdida: `READ UNCOMMITTED`

   Lectura no repetible: `REPEATABLE READ`

   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, ya que, por ejemplo, T1 en la acción 170 lee el gránulo D, previamente escrito por T2 en la acción 160, y T1 acaba su ejecución confirmando antes que T2.

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

		UNLOCK(D))	
250	R(D)		
260	UNLOCK(D)		
270	COMMIT (UNLOCK(A))		
280			RU(A)
290			W(A)
300			LOCK(C,X)
310			RU(C)
320			W(C)
330			LOCK(E,S)
340			R(E)
350			UNLOCK(E)
360			COMMIT (UNLOCK(A) UNLOCK(C))

Podemos ver que, con READ COMMITTED, se evita la interferencia de actualización perdida, pero siguen apareciendo las interferencias de lectura no repetible y análisis inconsistente. Como el horario todavía tiene interferencias, no existe ningún horario en serie equivalente.

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

#Acc	T1	T2	T3
10	LOCK(A,X)		
20	RU(A)		
30			LOCK(E,S)
40			R(E)
50			LOCK(A,X)
60		LOCK(F,S)	
70		R(F)	

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

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

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 siguientes peticiones:

T1	T2	T3
SELECT * FROM DOG WHERE birth > 01-01-2015;	UPDATE DOG SET breed = 'Beagle' WHERE id_dog = 2;	SQL Statement
SELECT * FROM DOG WHERE birth > 01-01-2015;	UPDATE DOG SET breed = ‘Bulldog’ WHERE id_dog = 2;	
COMMIT	COMMIT	COMMIT

La tabla *DOG* contiene las siguientes filas:

DOG				
id_dog	**name**	**breed**	**birth**	**sex**
2	Rocky	Bulldog	01-06-2015	M
22	Chloe	Boxer	07-01-2018	F

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 confirmada.

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

3. Proponed, si es posible, un horario que incorpore sólo las sentencias de T1, junto con una sentencia SQL a ejecutar por T3 que cause una interferencia de tipo fantasma entre T1 y T3 (es decir, que produzca la aparición de un fantasma).

Nota: Es posible que 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.*

Solución

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

#acc	T1	T2
10		UPDATE DOG SET breed = 'Beagle' WHERE id_dog = 2;
20	SELECT * FROM DOG WHERE birth > 01-01-2015;	
30	SELECT * FROM DOG WHERE birth > 01-01-2015;	
40		UPDATE DOG SET breed = ‘Bulldog’ WHERE id_dog = 2;
50		COMMIT
60	COMMIT	

Se puede ver que el horario presenta una interferencia de lectura no confirmada porque la transacción T1 a leído el resultado de la actualización de T2 que no será el valor final a la base de datos (el valor final será el que deja T2 en la segunda actualización). Si las transacciones se hubiesen ejecutado en serie (sin solapamientos), T1 habría recuperado los mismos valores para las dos operaciones de consulta (los valores previos a la ejecución de T2, o bien el valor de la última sentencia de actualización de T2).

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

#acc	T1	T2
10		UPDATE DOG SET breed = ‘Beagle’ WHERE id_dog = 2;
20		UPDATE DOG SET breed = ‘Bulldog’ WHERE id_dog = 2;
30	SELECT * FROM DOG WHERE birth > 01-01-2015;	
40	SELECT * FROM DOG WHERE birth > 01-01-2015;	
50	COMMIT	
60		COMMIT

En este caso, el horario no presenta ninguna interferencia. De hecho, produce los mismos resultados que el horario serial T2;T1 pero no es un horario recuperable. T1 lee datos pendientes de confirmación en las acciones 30 y 40 (en concreto, lee la modificación efectuada por T2 en la acción 20) y T1 acaba su ejecución antes que T2.

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

#acc	T1	T3
10	SELECT * FROM DOG WHERE birth > 01-01-2015;	
20		INSERT INTO DOG VALUES (20, ‘Rakkun’, ‘Siberian Husky’, 01-02-2019, ‘M’)
30	SELECT * FROM DOG WHERE birth > 01-01-2015;	
40		COMMIT
50	COMMIT	
60		

La transacción T1 en el instante 10 hace la lectura de los perros nacidos después del 1 de enero de 2015, que según los datos del enunciado, son todos (dos filas). En el instante 30, T1 hace una nueva consulta (que devuelve las mismas filas de la primera consulta más una fila adicional), donde aparece un fantasma, puesto que aparece el perro con id=20 que ha

insertado T3 y que no aparecía en la primera lectura. De hecho en T3 podríamos tener cualquier inserción de un perro nacido con posterioridad al 1-1-2015.

No es un caso de lectura no repetible porque las dos filas de la primera SELECT vuelven a aparecer en la segunda con los mismos valores, no han cambiado, pero aparece una fila extra (la que inserta T3, el fantasma).

Pregunta 3 (20% puntuación)

Enunciado

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

1) Para poder hacer copias de seguridad de una BD siempre se tiene que parar la actividad de los usuarios y aplicaciones.
2) Dada una base de datos donde los usuarios y las aplicaciones solo hacen operaciones de lectura, solo se podrían producir interferencias de tipo lectura no confirmada.
3) Uno de los motivos de la popularidad de PostgreSQL es ser el único SGBD que implementa el modelo de control de concurrencia MVCC.
4) Las acciones de solo lectura R(G), en términos del SQL se corresponden con sentencias `SELECT`, por lo tanto, estas acciones solo pueden ser ejecutadas por transacciones de tipos `READ ONLY`.

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

1) **Falsa.** Las copias de seguridad pueden ser estáticas o dinámicas. Las primeras exigen que se pare la actividad de los usuarios y las aplicaciones (al menos, las actualizaciones), en cambio, las segundas, no.
2) **Falsa**. Si solo se hacen lecturas, será imposible que se produzcan interferencias, por lo tanto no haría falta que las transacciones definan ningún nivel de aislamiento.

3) **Falsa.** El modelo de control de concurrencia MVCC no solo se encuentra disponible en PostgreSQL, sino que también lo implementan otros SGBD como, por ejemplo, Oracle y SQL Server.

4) **Falsa**. Las acciones de solo lectura pueden ser ejecutadas tanto por transacciones `READ ONLY` como por transacciones `READ WRITE`.