# Práctica: Transacciones

CI-0127 Bases de Datos, Universidad de Costa Rica

Sivana Hamer

## Importante

Este documento recopila contenidos de diversos de sitios web especializados, académicos y documentos compartidos por universidades. Toda la información es utilizada con fines estrictamente académicos.

# Problemas de control de concurrencia

1. Se tiene el siguiente *schedule*:

| $T_1$ | $T_2$ |
|---|---|
| read($A$); | |
| write($C$); | |
| | write($C$); |

   (a) ¿Cuáles de los siguientes problemas de control de concurrencia tiene el *schedule*? Marque todas las opciones posibles. ☐ Lost update ☐ Dirty read ☐ Unrepeatable read ☐ Phantom read

   (b) Para cada uno de los problemas anteriores, detalle todas las instrucciones que generan cada problema respectivo. Debe indicar para cual transacción genera el problema respectivo.

   (c) Si se quisieran evitar todos los problemas de control de concurrencia de este, ¿cuál es el nivel de aislamiento (*isolation level*) mínimo que se debe seleccionar? ☐ Read uncommitted ☐ Read committed ☐ Repeatable read. ☐ Serializable.

   (d) Explique su escogencia del nivel de aislamiento.

2. Se tiene el siguiente *schedule*:

| $T_1$ | $T_2$ |
|---|---|
| read($A$); | |
| | write($B$); |
| write($C$); | |
| | read($D$); |

   (a) ¿Cuáles de los siguientes problemas de control de concurrencia tiene el *schedule*? Marque todas las opciones posibles. ☐ Lost update ☐ Dirty read ☐ Unrepeatable read ☐ Phantom read

   (b) Para cada uno de los problemas anteriores, detalle todas las instrucciones que generan cada problema respectivo. Debe indicar para cual transacción genera el problema respectivo.

   (c) Si se quisieran evitar todos los problemas de control de concurrencia de este, ¿cuál es el nivel de aislamiento (*isolation level*) mínimo que se debe seleccionar? ☐ Read uncommitted ☐ Read committed ☐ Repeatable read. ☐ Serializable.

   (d) Explique su escogencia del nivel de aislamiento.

3. Se tiene el siguiente *schedule*:

| $T_1$ | $T_2$ |
|---|---|
| read($A$); | |
| | read($A$); |
| read($A$); | |
| | read($B$); |
| read($A$); | |
| | read($A$); |
| write($B$); | |

   (a) ¿Cuáles de los siguientes problemas de control de concurrencia tiene el *schedule*? Marque todas las opciones posibles. ☐ Lost update ☐ Dirty read ☐ Unrepeatable read ☐ Phantom read

 

(b) Para cada uno de los problemas anteriores, detalle todas las instrucciones que generan cada problema respectivo. Debe indicar para cual transacción genera el problema respectivo.

(c) Si se quisieran evitar todos los problemas de control de concurrencia de este, ¿cuál es el nivel de aislamiento (*isolation level*) mínimo que se debe seleccionar? ☐ Read uncommitted ☐ Read committed ☐ Repeatable read. ☐ Serializable.

(d) Explique su escogencia del nivel de aislamiento.

4. Se tiene el siguiente *schedule*:

| $T_1$ | $T_2$ | $T_3$ | $T_4$ |
|---|---|---|---|
| read($A$); | | | |
| | write($B$); | | |
| | | | read($D$); |
| | | write($D$); | |
| | | write($C$); | |
| | | | read($C$); |
| | write($C$); | | |

(a) ¿Cuáles de los siguientes problemas de control de concurrencia tiene el *schedule*? Marque todas las opciones posibles. ☐ Lost update ☐ Dirty read ☐ Unrepeatable read ☐ Phantom read

(b) Para cada uno de los problemas anteriores, detalle todas las instrucciones que generan cada problema respectivo. Debe indicar para cual transacción genera el problema respectivo.

(c) Si se quisieran evitar todos los problemas de control de concurrencia de este, ¿cuál es el nivel de aislamiento (*isolation level*) mínimo que se debe seleccionar? ☐ Read uncommitted ☐ Read committed ☐ Repeatable read. ☐ Serializable.

(d) Explique su escogencia del nivel de aislamiento.

5. Se tiene el siguiente *schedule*:

| $T_1$ | $T_2$ | $T_3$ |
|---|---|---|
| read($B$); | | |
| | | read($C$); |
| write($B$); | | |
| | | read($A$); |
| write($A$); | | |
| | | write($C$); |
| | read($C$); | |
| | write($C$); | |
| | | write($A$); |
| read($A$); | | |
| | write($B$); | |

(a) ¿Cuáles de los siguientes problemas de control de concurrencia tiene el *schedule*? Marque todas las opciones posibles. ☐ Lost update ☐ Dirty read ☐ Unrepeatable read ☐ Phantom read

(b) Para cada uno de los problemas anteriores, detalle todas las instrucciones que generan cada problema respectivo. Debe indicar para cual transacción genera el problema respectivo.

(c) Si se quisieran evitar todos los problemas de control de concurrencia de este, ¿cuál es el nivel de aislamiento (*isolation level*) mínimo que se debe seleccionar? ☐ Read uncommitted ☐ Read committed ☐ Repeatable read. ☐ Serializable.

 

(d) Explique su escogencia del nivel de aislamiento.

6. Se tiene el siguiente *schedule*:

| $T_1$ | $T_2$ | $T_3$ |
|---|---|---|
| | write($A$); | |
| | | read($B$); |
| write($C$); | | |
| | read($A$); | |
| read($B$); | | |
| | | write($B$); |
| | read($C$); | |
| | write($C$); | |
| | | write($A$); |
| read($B$); | | |

(a) ¿Cuáles de los siguientes problemas de control de concurrencia tiene el *schedule*? Marque todas las opciones posibles. ☐ Lost update ☐ Dirty read ☐ Unrepeatable read ☐ Phantom read

(b) Para cada uno de los problemas anteriores, detalle todas las instrucciones que generan cada problema respectivo. Debe indicar para cual transacción genera el problema respectivo.

(c) Si se quisieran evitar todos los problemas de control de concurrencia de este, ¿cuál es el nivel de aislamiento (*isolation level*) mínimo que se debe seleccionar? ☐ Read uncommitted ☐ Read committed ☐ Repeatable read. ☐ Serializable.

(d) Explique su escogencia del nivel de aislamiento.

 

# Protocolos de control de concurrencia

Para cada uno de los siguientes protocolos, seleccione **todos** los errores que pueden suceder.

1. Utilizando *candados básicos* puede suceder:

   ☐ Inconsistencia en estados de BD ☐ *Starvation* ☐ *Deadlocks* ☐ Concurrencia limitada
   ☐ *Cascading aborts*

2. Utilizando *2PL* puede suceder:

   ☐ Inconsistencia en estados de BD ☐ *Starvation* ☐ *Deadlocks* ☐ Concurrencia limitada
   ☐ *Cascading aborts*

3. Utilizando *strict or rigorous 2PL* puede suceder:

   ☐ Inconsistencia en estados de BD ☐ *Starvation* ☐ *Deadlocks* ☐ Concurrencia limitada
   ☐ *Cascading aborts*

4. Utilizando *granularidad múltiple* puede suceder:

   ☐ Inconsistencia en estados de BD ☐ *Starvation* ☐ *Deadlocks* ☐ Concurrencia limitada
   ☐ *Cascading aborts*

5. Utilizando *basic T/O* puede suceder:

   ☐ Inconsistencia en estados de BD ☐ *Starvation* ☐ *Deadlocks* ☐ Concurrencia limitada
   ☐ *Cascading aborts*

6. Utilizando *validation* puede suceder:

   ☐ Inconsistencia en estados de BD ☐ *Starvation* ☐ *Deadlocks* ☐ Concurrencia limitada
   ☐ *Cascading aborts*

 

# Serializabilidad

Para cada uno de los siguientes *schedules*, responda las preguntas.

1. Se tiene el siguiente *schedule*:

| $T_1$ | $T_2$ |
|---|---|
| read($A$); | |
| write($C$); | |
| | write($C$); |

   (a) ¿El *schedule* es serial? ☐ Si ☐ No

   (b) Dibuje el gráfico de dependencias del *schedule*.

   (c) ¿El *schedule* es serializable en conflictos (*conflict serializable*)? ☐ Si ☐ No

   (d) Si su respuesta a la pregunta anterior es "Si", provea el ejecución serial equivalente. En caso contrario provea los ciclos que indican que no es serializable en conflictos.

   (e) Seleccione todas las transacciones que se deben remover para hacer que la transacción sea serializable en conflictos.
   ☐ $T_1$ ☐ $T_2$ ☐ Ya es serializable en conflictos.

2. Se tiene el siguiente *schedule*:

| $T_1$ | $T_2$ |
|---|---|
| read($A$); | |
| | write($B$); |
| write($C$); | |
| | read($D$); |

   (a) ¿El *schedule* es serial? ☐ Si ☐ No

   (b) Dibuje el gráfico de dependencias del *schedule*.

   (c) ¿El *schedule* es serializable en conflictos (*conflict serializable*)? ☐ Si ☐ No

   (d) Si su respuesta a la pregunta anterior es "Si", provea el ejecución serial equivalente. En caso contrario provea los ciclos que indican que no es serializable en conflictos.

   (e) Seleccione todas las transacciones que se deben remover para hacer que la transacción sea serializable en conflictos.
   ☐ $T_1$ ☐ $T_2$ ☐ Ya es serializable en conflictos.

3. Se tiene el siguiente *schedule*:

| $T_1$ | $T_2$ |
|---|---|
| read($A$); | |
| | read($A$); |
| read($A$); | |
| | read($B$); |
| read($A$); | |
| | read($A$); |
| write($B$); | |

   (a) ¿El *schedule* es serial? ☐ Si ☐ No

(b) Dibuje el gráfico de dependencias del *schedule*.

(c) ¿El *schedule* es serializable en conflictos (*conflict serializable*)? ☐ Si ☐ No

(d) Si su respuesta a la pregunta anterior es "Si", provea el ejecución serial equivalente. En caso contrario provea los ciclos que indican que no es serializable en conflictos.

(e) Seleccione todas las transacciones que se deben remover para hacer que la transacción sea serializable en conflictos.
☐ $T_1$ ☐ $T_2$ ☐ Ya es serializable en conflictos.

4. Se tiene el siguiente *schedule*:

| $T_1$ | $T_2$ | $T_3$ | $T_4$ |
|---|---|---|---|
| read($A$); | | | |
| | write($B$); | | |
| | | | read($D$); |
| | | write($D$); | |
| | | write($C$); | |
| | | | read($C$); |
| | write($C$); | | |

(a) ¿El *schedule* es serial? ☐ Si ☐ No

(b) Dibuje el gráfico de dependencias del *schedule*.

(c) ¿El *schedule* es serializable en conflictos (*conflict serializable*)? ☐ Si ☐ No

(d) Si su respuesta a la pregunta anterior es "Si", provea el ejecución serial equivalente. En caso contrario provea los ciclos que indican que no es serializable en conflictos.

(e) Seleccione todas las transacciones que se deben remover para hacer que la transacción sea serializable en conflictos.
☐ $T_1$ ☐ $T_2$ ☐ $T_3$ ☐ $T_4$ ☐ Ya es serializable en conflictos.

5. Se tiene el siguiente *schedule*:

| $T_1$ | $T_2$ | $T_3$ |
|---|---|---|
| read($B$); | | |
| | | read($C$); |
| write($B$); | | |
| | | read($A$); |
| write($A$); | | |
| | | write($C$); |
| | read($C$); | |
| | write($C$); | |
| | | write($A$); |
| read($A$); | | |
| | write($B$); | |

(a) ¿El *schedule* es serial? ☐ Si ☐ No

(b) Dibuje el gráfico de dependencias del *schedule*.

(c) ¿El *schedule* es serializable en conflictos (*conflict serializable*)? ☐ Si ☐ No

(d) Si su respuesta a la pregunta anterior es "Si", provea el ejecución serial equivalente. En caso contrario provea los ciclos que indican que no es serializable en conflictos.

 

(e) Seleccione todas las transacciones que se deben remover para hacer que la transacción sea serializable en conflictos.
☐ $T_1$ ☐ $T_2$ ☐ $T_3$ ☐ Ya es serializable en conflictos.

6. Se tiene el siguiente *schedule*:

| $T_1$ | $T_2$ | $T_3$ |
|---|---|---|
| | write($A$); | |
| | | read($B$); |
| write($C$); | | |
| | read($A$); | |
| read($B$); | | |
| | | write($B$); |
| | read($C$); | |
| | write($C$); | |
| | | write($A$); |
| read($B$); | | |

(a) ¿El *schedule* es serial? ☐ Si ☐ No

(b) Dibuje el gráfico de dependencias del *schedule*.

(c) ¿El *schedule* es serializable en conflictos (*conflict serializable*)? ☐ Si ☐ No

(d) Explique formalmente porque es o no es un *schedule* serializable en conflictos.

(e) Seleccione todas las transacciones que se deben remover para hacer que la transacción sea serializable en conflictos.
☐ $T_1$ ☐ $T_2$ ☐ $T_3$ ☐ Ya es serializable en conflictos.

 

# Candados

Para cada una de los siguientes *schedules*, indique que candado simple se debe pedir. Asuma que despues de obtener un candado, no se libera.

1. Se tiene el siguiente schedule:

| | $T_1$ | $T_2$ | $T_3$ |
|---|---|---|---|
| $t_1$ | $read(A)$; | | |
| $t_2$ | | $read(B)$; | |
| $t_3$ | | | $write(A)$; |
| $t_4$ | | $write(B)$; | |
| $t_5$ | | | $write(C)$; |
| $t_6$ | $read(A)$; | | |

   (a) En $t_1$ se pide: ☐ $S-LOCK(A)$ ☐ $X-LOCK(A)$ ☐ $S-LOCK(B)$ ☐ $X-LOCK(B)$ ☐ $S-LOCK(C)$ ☐ $X-LOCK(C)$ ☐ Nada

   (b) En $t_2$ se pide: ☐ $S-LOCK(A)$ ☐ $X-LOCK(A)$ ☐ $S-LOCK(B)$ ☐ $X-LOCK(B)$ ☐ $S-LOCK(C)$ ☐ $X-LOCK(C)$ ☐ Nada

   (c) En $t_3$ se pide: ☐ $S-LOCK(A)$ ☐ $X-LOCK(A)$ ☐ $S-LOCK(B)$ ☐ $X-LOCK(B)$ ☐ $S-LOCK(C)$ ☐ $X-LOCK(C)$ ☐ Nada

   (d) En $t_4$ se pide: ☐ $S-LOCK(A)$ ☐ $X-LOCK(A)$ ☐ $S-LOCK(B)$ ☐ $X-LOCK(B)$ ☐ $S-LOCK(C)$ ☐ $X-LOCK(C)$ ☐ Nada

   (e) En $t_5$ se pide: ☐ $S-LOCK(A)$ ☐ $X-LOCK(A)$ ☐ $S-LOCK(B)$ ☐ $X-LOCK(B)$ ☐ $S-LOCK(C)$ ☐ $X-LOCK(C)$ ☐ Nada

   (f) En $t_6$ se pide: ☐ $S-LOCK(A)$ ☐ $X-LOCK(A)$ ☐ $S-LOCK(B)$ ☐ $X-LOCK(B)$ ☐ $S-LOCK(C)$ ☐ $X-LOCK(C)$ ☐ Nada

2. Se tiene el siguiente schedule:

| | $T_1$ | $T_2$ | $T_3$ |
|---|---|---|---|
| $t_1$ | | write($A$); | |
| $t_2$ | | | read($B$); |
| $t_3$ | write($C$); | | |
| $t_4$ | | read($A$); | |
| $t_5$ | read($B$); | | |
| $t_6$ | | | write($B$); |
| $t_7$ | | read($C$); | |
| $t_8$ | | write($C$); | |
| $t_9$ | | | write($A$); |
| $t_{10}$ | read($B$); | | |

   Indique para cada instrucción que candado simple se debe pedir. Debe indicar una de las siguientes opciones en cada espacio: $S-LOCK(A)$, $X-LOCK(A)$, $S-LOCK(B)$, $X-LOCK(B)$, $S-LOCK(C)$, $X-LOCK(C)$ y Nada. Asuma que después de obtener un candado, no se libera. La opción de nada indica que no hay que pedir un candado.

 

| | $T_1$ | $T_2$ | $T_3$ |
|---|---|---|---|
| $t_1$ | | ——— | |
| $t_2$ | | | ——— |
| $t_3$ | ——— | | |
| $t_4$ | | ——— | |
| $t_5$ | ——— | | |
| $t_6$ | | | ——— |
| $t_7$ | | ——— | |
| $t_8$ | | ——— | |
| $t_9$ | | | ——— |
| $t_{10}$ | ——— | | |

# Manejo de deadlocks: Detección

1. Se tiene las siguientes transacciones que piden candados simples:

| | $T_1$ | $T_2$ | $T_3$ |
|---|---|---|---|
| $t_1$ | **S-LOCK(A);** | | |
| $t_2$ | | **X-LOCK(B);** | |
| $t_3$ | | | **X-LOCK(C);** |
| $t_4$ | | **S-LOCK(A);** | |
| $t_5$ | | **X-LOCK(C);** | |
| $t_6$ | | | **X-LOCK(A);** |
| $t_7$ | **S-LOCK(D);** | | |
| $t_8$ | **S-LOCK(A);** | | |

   (a) Para cada tiempo, indique si cada candado otorgará (*granted*) o bloqueará (*blocked*) el candado.
      i. En $t_1$: ☐ Otorgará ☐ Bloqueará
      ii. En $t_2$: ☐ Otorgará ☐ Bloqueará
      iii. En $t_3$: ☐ Otorgará ☐ Bloqueará
      iv. En $t_4$: ☐ Otorgará ☐ Bloqueará
      v. En $t_5$: ☐ Otorgará ☐ Bloqueará
      vi. En $t_6$: ☐ Otorgará ☐ Bloqueará
      vii. En $t_7$: ☐ Otorgará ☐ Bloqueará
      viii. En $t_8$: ☐ Otorgará ☐ Bloqueará

   (b) Para las transacciones anteriores, dibuje el *wait-for graph*.

   (c) ¿Existe un deadlock en el *schedule* anterior? ☐ Si ☐ No

   (d) Explique porque hay o no hay un deadlock.

2. Se tiene las siguientes transacciones que piden candados simples:

| | $T_1$ | $T_2$ | $T_3$ |
|---|---|---|---|
| $t_1$ | | $X-LOCK(A)$ | |
| $t_2$ | | | $S-LOCK(B)$ |
| $t_3$ | $X-LOCK(C)$ | | |
| $t_4$ | | Nada | |
| $t_5$ | $S-LOCK(B)$ | | |
| $t_6$ | | | $X-LOCK(B)$ |
| $t_7$ | | $S-LOCK(C)$ | |
| $t_8$ | | $X-LOCK(C)$ | |
| $t_9$ | | | $X-LOCK(A)$ |
| $t_{10}$ | Nada | | |

   (a) Para el *schedule* anterior, dibuje el *wait-for graph*.

   (b) ¿Existe un deadlock en el *schedule* anterior? ☐ Si ☐ No

   (c) Explique porque hay o no hay un deadlock.

 

# Manejo de deadlocks: Prevención

Para cada uno de los siguientes protocolos, indique si se otorga el candado ($O$), bloquea el candado o si se encuentra la transacción bloqueada($B$), aborta una transacción ($A$) o se encuentra muerta la transacción ($M$). Asuma que después de obtener un candado, no se libera. Además asuma que las transacciones se crearon de tal manera que las estampillas de tiempo son: $T_1 < T_2 < T_3$.

1. Se tiene las siguientes llamadas a candados:

| | $T_1$ | $T_2$ | $T_3$ |
|---|---|---|---|
| $t_1$ | **S-LOCK(C);** | | |
| $t_2$ | | | **X-LOCK(C);** |
| $t_3$ | | **X-LOCK(A);** | |
| $t_4$ | | **S-LOCK(B);** | |
| $t_5$ | **X-LOCK(A);** | | |
| $t_6$ | | | **X-LOCK(A);** |
| $t_7$ | | | **S-LOCK(B);** |

   (a) Si *no* se utiliza alguna política de prevención:
      i. En $t_1$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$
      ii. En $t_2$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$
      iii. En $t_3$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$
      iv. En $t_4$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$
      v. En $t_5$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$
      vi. En $t_6$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$
      vii. En $t_7$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$

   (b) Si se utiliza la política *wait-die*:
      i. En $t_1$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$
      ii. En $t_2$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$
      iii. En $t_3$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$
      iv. En $t_4$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$
      v. En $t_5$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$
      vi. En $t_6$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$
      vii. En $t_7$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$

   (c) Si se utiliza la política *wound-wait*:
      i. En $t_1$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$
      ii. En $t_2$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$
      iii. En $t_3$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$
      iv. En $t_4$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$
      v. En $t_5$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$
      vi. En $t_6$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$
      vii. En $t_7$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$

2. Se tienen las siguientes llamadas a candados:

   (a) Si *no* se utiliza alguna política de prevención:
      i. En $t_5$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$ ☐ $N$
      ii. En $t_6$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$ ☐ $N$
      iii. En $t_7$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$ ☐ $N$

 

| | $T_1$ | $T_2$ | $T_3$ |
|---|---|---|---|
| $t_1$ | | $X - LOCK(A)$ | |
| $t_2$ | | | $S - LOCK(B)$ |
| $t_3$ | $X - LOCK(C)$ | | |
| $t_4$ | | Nada | |
| $t_5$ | $S - LOCK(B)$ | | |
| $t_6$ | | | $X - LOCK(B)$ |
| $t_7$ | | $S - LOCK(C)$ | |
| $t_8$ | | $X - LOCK(C)$ | |
| $t_9$ | | | $X - LOCK(A)$ |
| $t_{10}$ | Nada | | |

iv. En $t_8$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$ ☐ $N$
v. En $t_9$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$ ☐ $N$
vi. En $t_{10}$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$ ☐ $N$

(b) Si se utiliza la política *wait-die*:

i. En $t_5$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$ ☐ $N$
ii. En $t_6$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$ ☐ $N$
iii. En $t_7$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$ ☐ $N$
iv. En $t_8$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$ ☐ $N$
v. En $t_9$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$ ☐ $N$
vi. En $t_{10}$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$ ☐ $N$

(c) Si se utiliza la política *wound-wait*:

i. En $t_5$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$ ☐ $N$
ii. En $t_6$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$ ☐ $N$
iii. En $t_7$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$ ☐ $N$
iv. En $t_8$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$ ☐ $N$
v. En $t_9$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$ ☐ $N$
vi. En $t_{10}$: ☐ $O$ ☐ $B$ ☐ $A$ ☐ $M$ ☐ $N$