# 2nd LAB SESSION ON TRANSACTIONS

**Given Name: Núria Family name: Canals Nevado**

**Given Name: Marc Family name: Duran López**

1. (40%) Consider the No Steal / Force policy:

1. Provide the pseudo code of the *read*, *write*, *commit* and *abort* operations, so that we guarantee recoverability in **case of power failure**. Use as basis those in pages 19 and 20 for the steal / no force policy.

**procedure read(t: transaction\_id, p: page\_id, v: page\_value)**

**if search(p) = 0 then**

//Si la pàgina no està a la memòria principal, portar la pàgina del disc a la memòria.

**fetch(p);**

**end if**

//Llegeix el valor de la pàgina i l’assigna a la variable v

**v := content(p);**

**endProcedure**

**procedure write(t: transaction\_id, p: page\_id, v: new\_value)**

**if search(p) = 0 then**

//Si la pàgina no està a la memòria principal, portar la pàgina del disc a la memòria.

**fetch(page\_id);**

**end if**

//Registrar el log abans de realitzar l’escriptura

**write\_log('u', t, p, content(p), v);**

//Actualitzar la pàgina amb el nou valor

**content(p) := v;**

//Marcar la pàgina dirty per indicar que ha sigut modificada

**dirty(p) := 1;**

**endProcedure**

**procedure commit(t: transaction\_id)**

**for each page\_id in transaction\_buffer(t) do**

//Per cada pàgina que la transacció té la memòria, desmarcar la pàgina amb “pin” per indicar que ja no està sent utilitzada per la transacció

**unpin(page\_id);**

**if dirty(page\_id) then**

//Si la pàgina està dirty, escriure els canvis al disc

**flush(page\_id);**

**end if**

**end for**

//Registrar el log del commit

**write\_log('c', t);**

**endProcedure**

**procedure abort(t: transaction\_id)**

**for each page\_id in transaction\_buffer(t) do**

//Per cada pàgina que la transacció té en memòria, desmarcar la pàgina amb “pin” per indicar que ja no està sent utilitzada per la transacció

**unpin(page\_id);**

**end for**

//Registrar el log de la cancelació

**write\_log('a', t);**

**endProcedure**

1. Under what circumstances that policy may be interesting (e.g., What are its **cons and pros**? **What kind of systems** can you think of that would suit it?)

La política de "No Steal / Force" és una política de control de la concurrència i recuperació que té els seus propis avantatges i desavantatges i és adequada per a certs tipus de sistemes i escenaris. A continuació, descriurem algunes de les circumstàncies en les quals aquesta política pot ser interessant:

**Avantatges:**

* **Recuperabilitat garantida:** Garanteix la recuperabilitat de la base de dades en cas de fallades del sistema o de l'energia. Totes les modificacions realitzades per una transacció es guarden en un emmagatzematge secundari (disc) abans de permetre el commit, la qual cosa significa que no es perden canvis no confirmats en cas d'una fallada.
* **Senzillesa d'implementació:** Segueix un enfocament directe d'escriure canvis al disc abans del commit. Això facilita la comprensió i la implementació de la política en sistemes de bases de dades.
* **Menys bloquejos:** En comparació amb altres polítiques de control de la concurrència més restrictives, aquesta tendeix a requerir menys bloquejos, causant així, una major concurrència i un millor rendiment en sistemes amb una càrrega de treball elevada.

**Desavantatges:**

* **Cost addicional d'escriptura al disc:** El principal desavantatge d'aquesta política és que pot generar un cost addicional d'escriptura al disc. Cada vegada que una transacció realitza una escriptura, es requereix una escriptura addicional al disc per garantir la recuperabilitat. Això pot ser costós en termes de rendiment, especialment en sistemes amb una gran quantitat d'escriptures.
* **Bloqueig prolongat de pàgines:** Com a part de la política, les pàgines de dades estan marcades com a ocupades amb el “pin” per transaccions actives. Això significa que una pàgina pot estar bloquejada per un període prolongat, el que podria afectar negativament altres transaccions que necessitin accedir a la mateixa pàgina.

**Sistemes que poden beneficiar-se:**

* **Sistemes financers:** En sistemes financers i bancaris, on la integritat de les dades és crítica i s'han d'evitar pèrdues de dades, aquesta política pot ser una elecció adequada per garantir la recuperabilitat en cas de fallades.
* **Sistemes de registre i auditoria:** En aquests entorns, és necessari mantenir un registre detallat de totes les transaccions i canvis de dades, aquesta política pot proporcionar una traçabilitat completa i assegurar que tots els canvis es registrin de manera fiable.
* **Sistemes de bases de dades de petita escala:** Sistemes on el cost addicional d'escriptura al disc no és un problema crític i la simplicitat d'implementació és una avantatge, aquesta política pot ser adequada.

En resum, la política de No Steal / Force és adequada per a sistemes on la garantia de recuperabilitat és fonamental i on el cost addicional d'escriptura al disc no sigui un gran inconvenient ja que pot afectar al rendiment si hi ha moltes escriptures.

1. (30%) Given a DBMS without any concurrency control mechanism, let’s suppose that we have the following history (actions have been numbered just to facilitate referencing them):

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

Let’s suppose now that the DBMS is based on an **optimistic technique** that validates readings at commit time. How would result the same history? **Is any transaction cancelled?**

Com és la tècnica optimista, cada transacció anirà fent localment i quan arribi al commit validarà amb les altres si hi ha hagut alguna interferència.

Per aquest motiu, un cop executada l’ultima acció abans dels commits, la 170, obtindrem el següent:

RS(T1) ={A,F,D,E}

RS(T2)={E,C}

RS(T3)={A,F}

WS(T1)={A,E}

WS(T2)={C}

WS(T3)={A,F}

En la següent acció, **la 180**, T2 fa commit. Al ser el primer en fer-ho, no causarà cap interferència perquè no ha de comprovar amb cap altre intersecció les seves accions. Per això, s’afegeix al set of commited transactions:

setOfCommitedTx(T1)={T2}

setOfCommitedTx(T3)={T2}

Al acabar aquesta acció, **en la 190**, T1 ha de validar si hi ha algun conflicte amb alguna transacció que ja ha finalitzat exitosament, en aquest cas, ha de comprovar si hi ha hagut alguna interferència amb T2 que és la única que ha fet commit abans que ella. D’aquesta forma:

Es comprova la interferència: RS(T1) ∩ WS(T2) = Ø, com és buida, T1 fa commit i es manté el setOfCommitedTx(T1)={T2} i s’afegeix al setOfCommitedTx(T3)={T2, T1}.

Finalment, arribem a l’última acció, **la 200**, on T3 ha de validar, en aquest cas veiem que ha de tenir en compte les dues transaccions ja finalitzades, la T1 i T2. Per aquest motiu, es comproven les interferències:

RS(T3) ∩ RS(T2) = Ø, no causa cap interferència, però, RS(T3) ∩ RS(T1) = {A}. La validació amb la transacció 1 no retorna un conjunt null, hi ha un conflicte amb el grànol A, per aquest motiu, T3 s’aborta.

1. (30%) Given a DBMS without any concurrency control mechanism, let’s suppose that we have the following history (actions have been numbered just to facilitate referencing them):

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

Let’s suppose now that the DBMS is based on a **dynamic timestamping** technique. How would result the same history? **Is any transaction cancelled?**

Mirant el contingut de RS i WS a cada acció:

En l'acció 70 de la seqüència, es presenta una situació en la qual hi pot haver una interferencia entre dues transaccions, T1 i T3. La raó d'aquest conflicte radica en el fet que T3 vol llegir una part de dades, "grànul A," que ha estat prèviament modificada per T1.

Ara bé, en aquest punt, cap de les dues transaccions (ni T1 ni T3) té assignada un timestamp. Per aclarir l'ordre en què aquestes transaccions haurien de ser processades i resoldre el possible conflicte, s’assignen marques de temps de manera dinàmica.

Ja que les dues transaccions no tenen marques de temps prèvies, s'assignen valors que compleixen amb la condició TS(T1) < TS(T3). Això significa que es vol assegurar que la marca de temps de T1 sigui menor que la de T3 per indicar que T1 té prioritat sobre T3 en l'accés als recursos.

Per tant, en aquesta situació concreta, es realitza la següent assignació dinàmica de marques de temps:

* TS(T1) = 1 (per a la transacció T1)
* TS(T3) = 2 (per a la transacció T3)

Aquestes assignacions de marques de temps ajuden a establir una jerarquia d'execució i resolen el possible conflicte entre T1 i T3 en l'accés al grànul A.

Quan arribem a l'acció 150, ens trobem novament en una situació similar, però amb diferents transaccions. En aquest cas, la transacció T2 vol llegir el valor grànul E que ha estat modificat per T1.

En aquest moment, és important notar que la marca de temps de T1 ja té un valor assignat i no és nul, mentre que la marca de temps de T2 encara és nul·la, és a dir, no té un valor assignat.

Per gestionar aquesta situació i establir una jerarquia d'execució per resoldre possibles conflictes, es decideix assignar una marca de temps a T2. Aquesta marca de temps es fa de manera que s'asseguri que TS(T1) < TS(T2), és a dir, que la marca de temps de T1 sigui més petita que la de T2.

Finalment, com a resultat d'aquesta assignació dinàmica de marques de temps, a l'acció 150, es defineixen les següents marques de temps:

* TS(T1) = 1 (per a la transacció T1)
* TS(T2) = 3 (per a la transacció T2)

Aquestes marques de temps ajuden a establir l'ordre d'execució i asseguren que T1 tingui prioritat sobre T2 en l'accés al grànul E, evitant possibles conflictes.

Ja que sempre es verifica que, quan una transacció T està executant una acció en una unitat de temps determinada, la seva marca de temps (TS) és major que les marques de temps de totes les altres transaccions actives (TS(T) > TS(Ti) ), no hi ha cap necessitat de cancel·lar cap de les transaccions.