Керування трансакціями

Відновлення

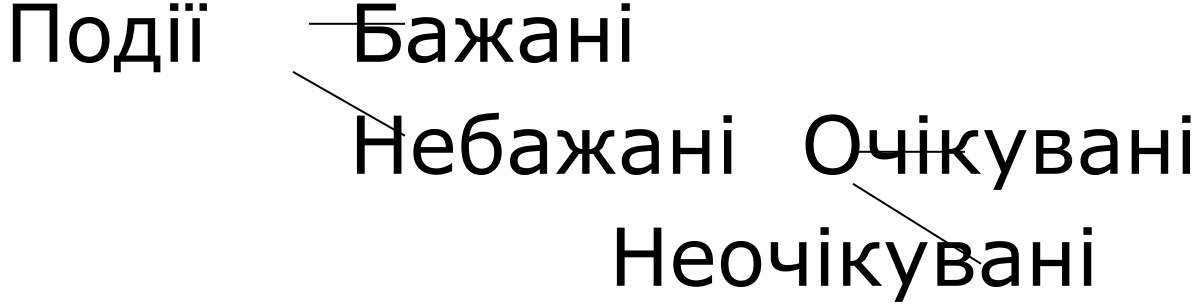
Трансакції

- Трансакція – логічна одиниця роботи БД

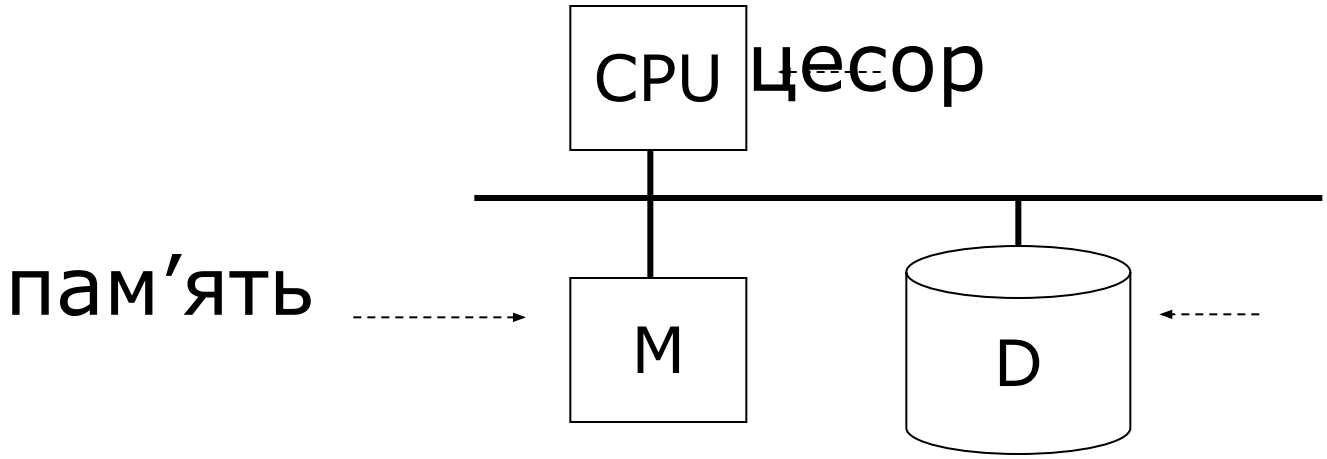
Властивості трансакцій (ACID)

- **Атомарність (Atomicity)**. Трансакції атомарні – виконується все або нічого.
- **Узгодженість (Consistency)**. Трансакції переводять базу даних з одного узгодженого стану в інший узгоджений стан.
- **Ізольованість (Isolation)**. Транзакції ізольовані одна від одної.
- **Довговічність (Durability)**. Якщо трансакція успішно закінчена, виконані нею дії оновлень даних зберігаються в БД постійно, навіть якщо в наступний момент відбудеться збій.

Режими відмов



Модель системи



- Бажані події: дивись документацію…
- Небажані очікувані події:
  - Системний збій
    - Втрата вмісту пам’яті
    - Зависання -> перезавантаження
  - Пошкодження носія
- Небажані неочікувані події
  - Все інше!

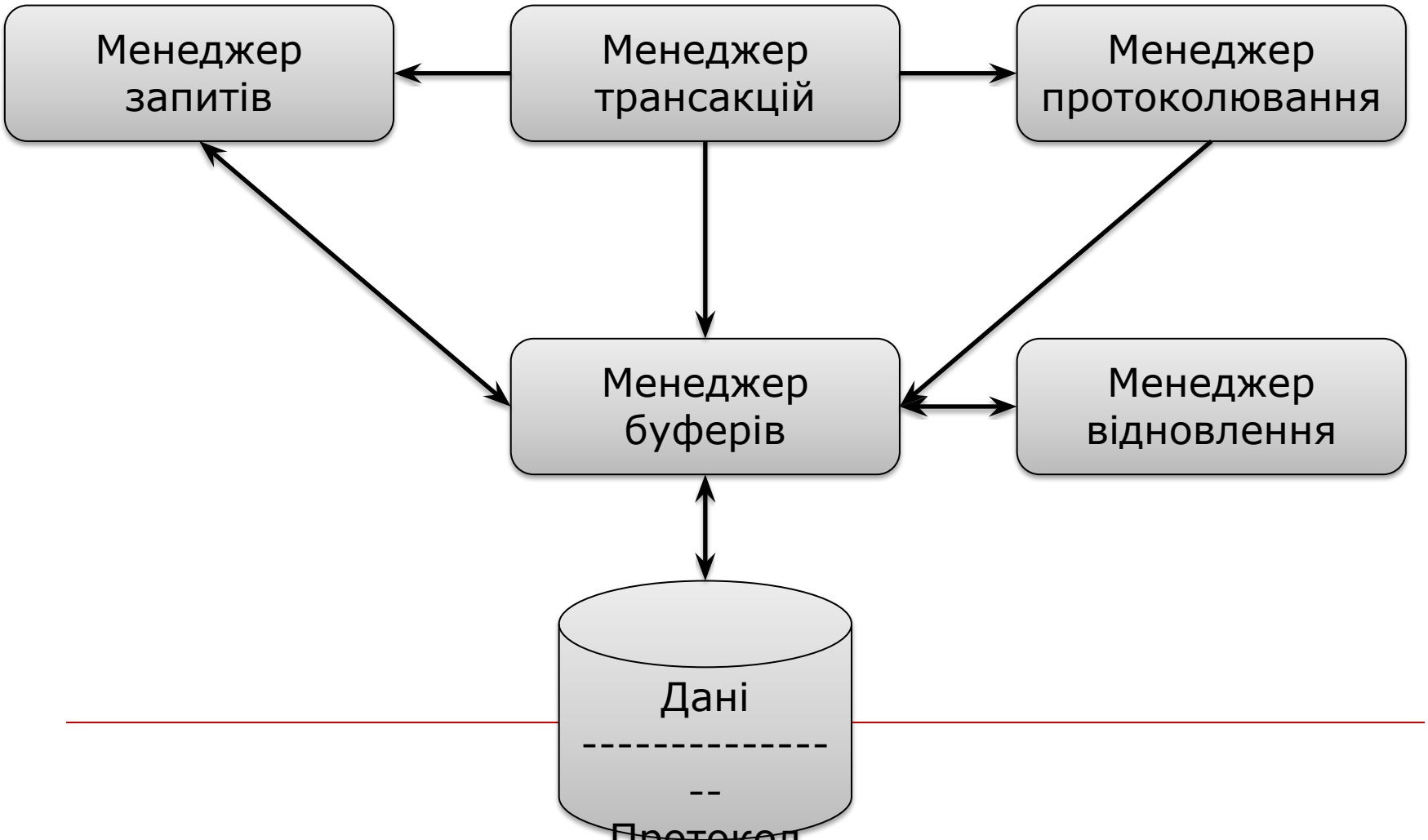
Режими відмов

- Помилкові елементи даних
- Пошкодження носія
- Катастрофа
- Збій системи

Менеджер трансакцій

- Відповідальність за коректну поведінку трансакцій покладається на **менеджер трансакцій**.

Менеджер трансакцій



Базові операції трансакцій

- В процесі виконання трансакція має справу з трьома адресними просторами:
  - простір **дискових блоків**, що містять елементи даних БД;
  - простір **віртуальної або оперативної пам’яті**, що керується менеджером буферів;
  - власний **локальний адресний простір** трансакції.

Базові операції трансакцій

- **Input (X)**: дисковий блок, що містить значення X копіюється в буфер оперативної пам’яті.
- **Output (X)**: копіювати блок з елементом даних X на диск
- **Read (x,t)**: копіювати елемент X в локальну змінну t трансакції
- **Write (x,t)**: копіювати X значення локальної змінної t в буфер пам’яті, що відповідає елементу X бази даних

Ключова проблема

- Незакінчені трансакції

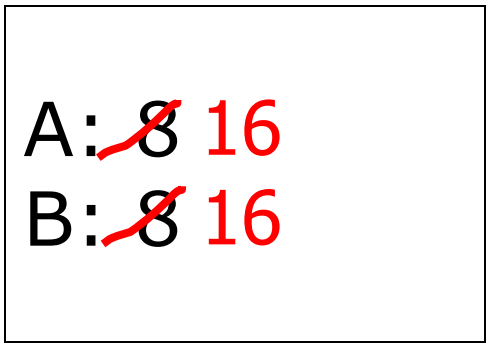
Наприклад Constraint: A=B

T_1: $A \leftarrow A \times 2$

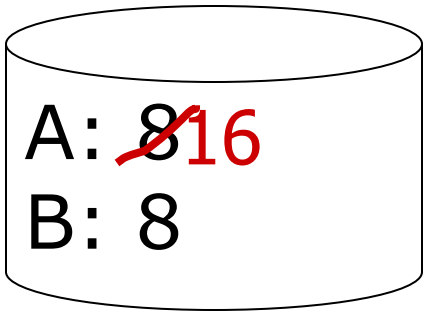
$B \leftarrow B \times 2$

T1: Read (A,t); t ← t×2
Write (A,t);
Read (B,t); t ← t×2
Write (B,t);
Output (A);
Output (B);

збій!



пам’ять



диск

Вирішення проблеми

- Потрібна **атомарність**
  - виконати всі дії трансакції або не виконати жодної

Рішення – протоколювання

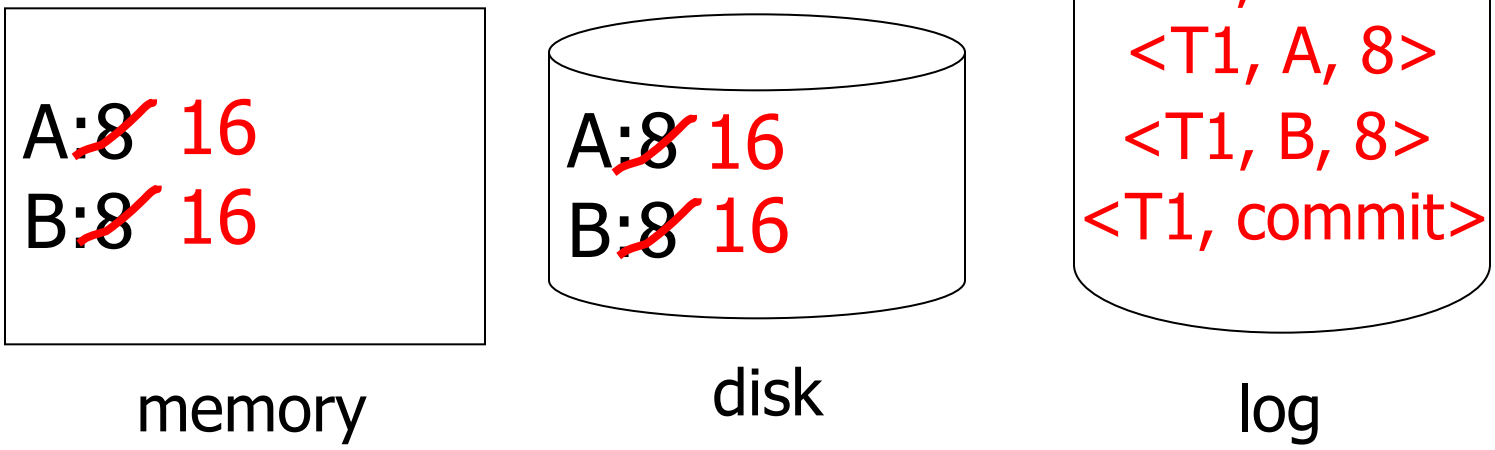
- Протокол, чи журнал – це **послідовність** так званих **записів** протоколу, кожний із яких несе в собі інформацію про певну дію, що виконується трансакцією.
- В протоколі можуть перемішуватись записи декількох трансакцій.

Відновлення з використання протоколу

- Для усунення згубних наслідків збою системи одні трансакції будуть виконані повторно, і зміни, виконані ними й раніше збережені в БД зафіксуються заново. Іншим транзакціям прийдеться відмінити свої дії, наче ці трансакції взагалі й не активувалися.

Undo logging

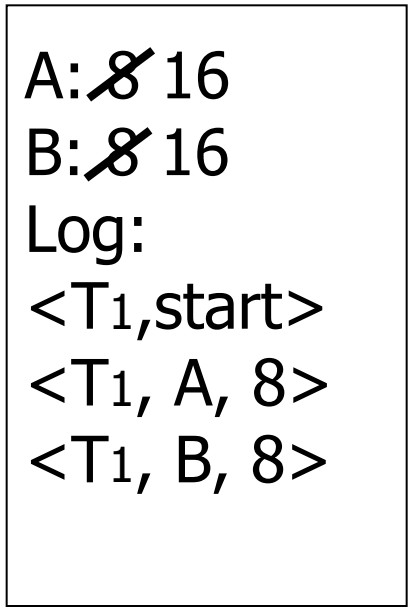
T1: Read (A,t); t ← t×2 A=B
Write (A,t);
Read (B,t); t ← t×2
Write (B,t);
Output (A);
Output (B);



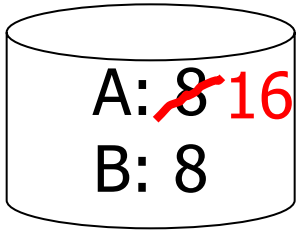
“Ускладнення”

- Журнал пишеться спочатку в пам’ять
- Не пишеться на диск при кожній дії

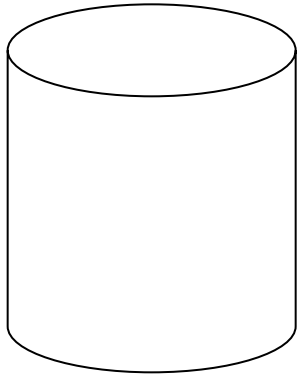
memory



DB



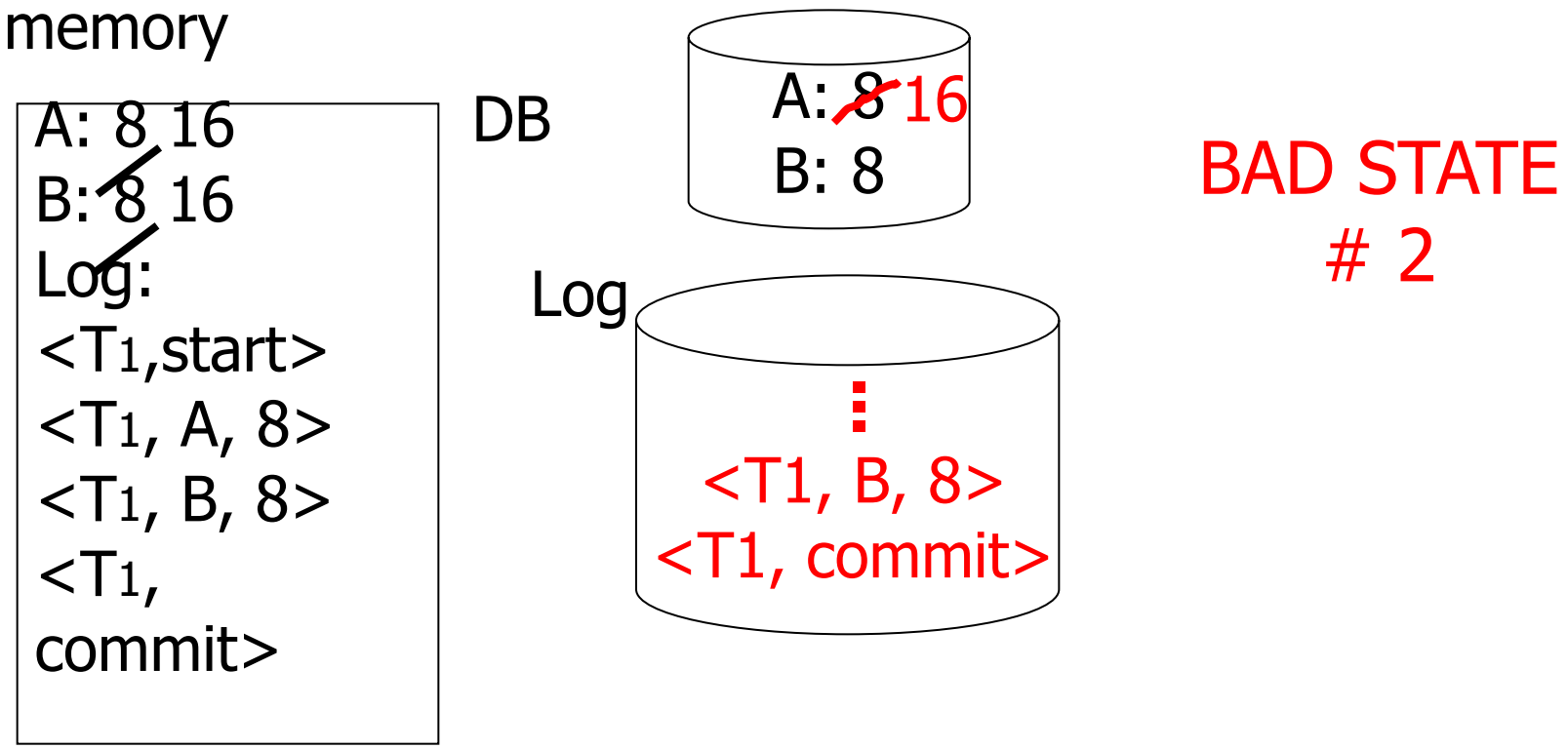
Log



BAD STATE # 1

“Ускладнення”

- Журнал пишеться спочатку в пам’ять
- Не пишеться на диск при кожній дії



Правила протоколювання в режимі Undo

1. Для кожної дії створювати запис журналу, що містить старе значення елементу даних

2. U_1 Якщо трансакція T змінює елемент *x*, запис журналу виду <T,x,v> повинен бути занесений в протокол **до** збереження нового значення на диску (write ahead logging:WAL)

3. U_2 При фіксації результатів трансакції T запис <T, commit> слід заносити в протокол тільки після «скидання» всіх змінених елементів БД на диск, причому інтервал між дисковими операціями і записом <T, commit> повинен бути як найменшим.

Відновлення: протокол Undo

- Перша задача менеджера відновлення полягає в розподілі всіх трансакцій, згаданих в журналі, по двом категоріям – зафіксовані й незафіксовані.
- Менеджер відновлення повинен здійснювати сканування журналу, починаючи з кінця і рухатись до початку.

Відновлення: протокол Undo

- Для кожної Ti із записом <Ti, start> в журналі:

  - If <Ti,commit> or <Ti,abort> in log, do nothing
  - Else For all <Ti, *X*, *v*> in log:
    write *(X, v)*
    output (*X*)
    Write <Ti, abort> to log

Відновлення: протокол Undo

(1) Дано S = множина транзакцій в журналі з
<Ti, start>, але без
<Ti, commit> (або <Ti, abort>), тобто незакінчені

(2) Для кожної<Ti, X, v> в журналі,
в зворотному порядку (останні → ранні) виконати:
- if Ti ∈ S then
  - write (X, v)
  - output (X)

(3) Для кожної Ti ∈ S виконати
- записати<Ti, abort> в журнал

Питання

- Чи можна зробити запис <Ti, abort> (крок 3) в будь-якій послідовності?
  - Наприклад: T1 і T2 обидві пишуть A
  - T1 виконується перед T2
  - T1 і T2 обидві rolled-back
  - <T1, abort> записаний, а <T2, abort> - ні

Питання

- Що буде коли відбудеться збій під час виконання відновлення?

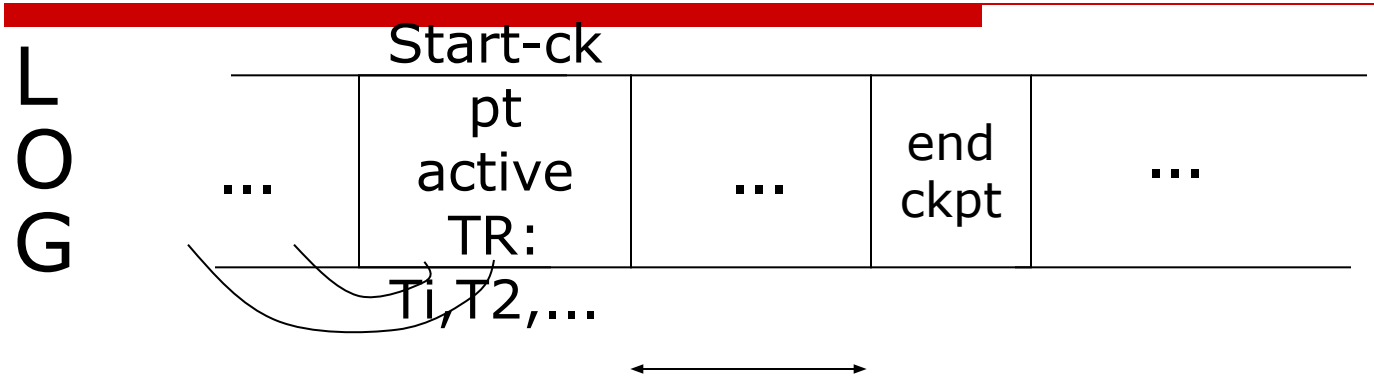
Введення контрольних точок

- Для введення контрольної точки системі потрібно:
  1. Призупинити прийом запитів на активізацію нових трансакцій.
  2. Дочекатися, поки всі діючі трансакції не виконають операції фіксації чи переривання і не занесуть в журнал записи <T, commit> або <T, abort>.
  3. Здійснити «скидання» журналу на диск командою flush log.
  4. «Скинути» всі буфери на диск
  5. Занести в протокол запис виду <CKPT>.
  6. Відновити прийом запитів.
- Недоліки?

Динамічне введення контрольних точок

1. Внести в протокол запис <START CKPT $T_1 \ldots T_n$ > і через команду flush log «скинути» протокол на диск; $T_1 \ldots T_n$ - всі активні трансакції на момент введення контрольної точки
2. Дочекатися моменту фіксації або переривання всіх трансакцій з $T_1 \ldots T_n$ не забороняючи можливості старту нових трансакцій.
3. При завершенні всіх трансакцій з $T_1 \ldots T_n$ зберегти в протоколі запис <END CKPT> і виконати flush log.

Динамічні контрольні точки



чекаємо завершення трансакцій

Відновлення з використанням контрольних точок

1. Якщо спочатку менеджеру відновлення зустрівся запис виду <END CKPT>, це свідчить про те, що всі незавершені трансакції почалися після процедури введення контрольної точки, тобто знаходяться після запису <START CKPT $T_1 \ldots T_n$ >. Тому достатньо здійснити сканування до запису <START CKPT $T_1 \ldots T_n$ >.

Відновлення з використанням контрольних точок

2. Якщо першим зустрівся запис <START CKPT $T_1 \ldots T_n$ >, значить, збій відбувся в період виконання процедури введення контрольної точки. Незавершеними будуть тільки ті трансакції із числа $T_1 \ldots T_n$, які не встигли здійснити фіксацію своїх результатів до виникнення ситуації відмови.

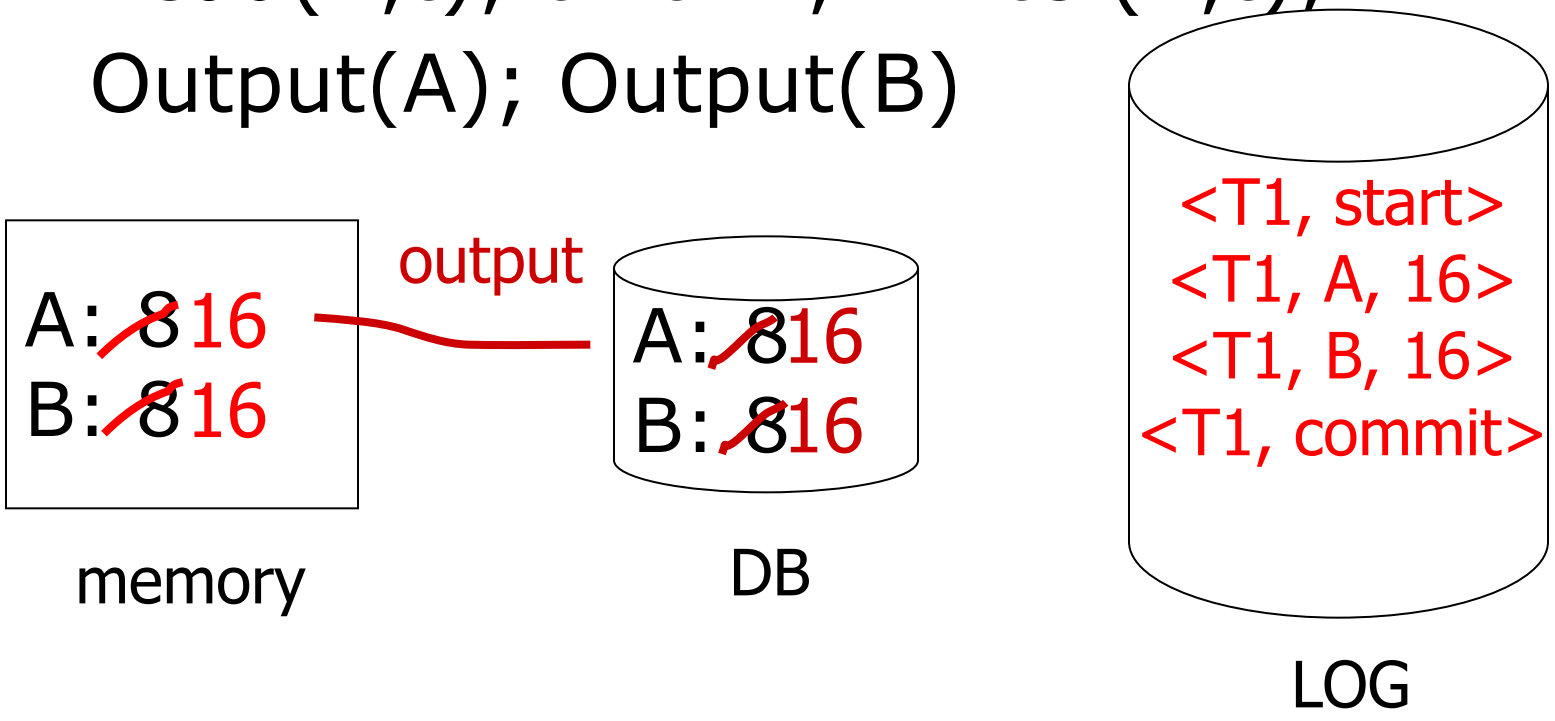
<T1, START>
<T1, A, 5>
<T2, START>
<T2, B, 10>
<START CKPT T1, T2>
<T3, START>
<T1, D, 20>
<T1, COMMIT>
<T3, E, 25>
<T2, COMMIT>
<END CKPT>
<T3, F, 30>

Протоколювання в режимі REDO

1. Для кожної дії створювати запис журналу, що містить нове значення елементу даних
2. R_1 Перш ніж трансакція змінить X на дискові, необхідно внести в протокол всі записи оновлення X, включаючи запис оновлення <T, x, v> і запис фіксації <T, commit>

Протоколювання в режимі REDO

T1: Read(A,t); t←t×2; Write (A,t);
Read(B,t); t←t×2; Write (B,t);
Output(A); Output(B)



Відновлення: протокол REDO

1. Ідентифікувати всі завершені трансакції
2. Сканувати протокол в напрямку від початку до кінця. Зустрівши запис <T, x, v>:
   - ігнорувати його, якщо T – незавершена
   - зберегти на дискові значення v для елемента x, якщо трансакція T завершена.
3. Для кожної незавершеної трансакції T зберегти в журналі запис <T, abort>
4. Виконати команду flush log

Відновлення: протокол REDO

- Для кожної Ti з <Ti, commit> в журналі:
  - Для всіх<Ti, X, v> в журналі:

    Write(X, v)

    Output(X)

Відновлення: протокол REDO

1. Дано S = множина трансакцій з <Ti, commit> в журналі
2. Для кожної <Ti, X, v> в журналі, в порядку появи (ранні →останні) виконати:

   - if Ti ∈ S then Write(X, v)
     Output(X)

Введення контрольних точок

1. Внести в журнал запис виду <START CKPT $T_1 \ldots T_n$>, де - $T_1 \ldots T_n$ всі активні (незафіксовані) трансакції і виконати «скидання» протоколу на диск командою flush log.
2. Зберегти на дискові значення всіх елементів бази даних, змінених в буферах пам’яті, але не записаних на диск тими трансакціями, які на момент включення в журнал запису <START CKPT…> виявилися завершеними.
3. Зберегти в протоколі запис <END CKPT> і виконати flush log

<T1, START>
<T1, A, 5>
<T2, START>
<T1, COMMIT>
<T2, B, 10>
<START CKPT T2>
<T2, C, 15>
<T3, START>
<T3, E, 25>
<END CKPT>
<T2, COMMIT>
<T3, COMMIT>

Відновлення: протокол REDO

1. Якщо спочатку менеджеру відновлення зустрівся запис виду <END CKPT>, це свідчить про те, що всі зміни, внесені трансакціями, які закінчилися до відповідного запису <START CKPT $T_1 ... T_n$> записані на диск і їх відновлювати не потрібно.

Відновлення: протокол REDO

- Але будь-яка трансакція зі списку $T_1 ... T_n$, можливо ще не встигла записати дані на диск, навіть якщо в журналі є запис <T, commit>. Тому для відновлення потрібно розглянути трансакції зі списку $T_1 ... T_n$ або ті, які стартували після процедури введення контрольної точки.

Відновлення: протокол REDO

2. Якщо першим зустрівся <START CKPT $T_1 \ldots T_n$>. В такому випадку неможливо гарантувати, що всі трансакції, завершені на момент внесення <START CKPT $T_1 \ldots T_n$>, змогли зберегти свої оновлення даних на диск. Тому треба перейти до попереднього <END CKPT>, знайти відповідний йому запис <START CKPT $S_1 \ldots S_m$>, а потім повторити збереження результатів оновлень всіх зафіксованих трансакцій Sj зі або тих, які стартували після внесення а протокол <START CKPT $S_1 \ldots S_m$>

Протоколювання: undo/redo

- Режим undo вимагає, щоб змінені дані зберігалися на диску безпосередньо по закінченню трансакції.
- Режим redo передбачає зберігання модифікованих блоків даних в буферах до моменту, поки трансакція не буде зафіксована і записи протоколу не будуть “скинуті” на диск.

Протоколювання: undo/redo

- Перш ніж трансакція T зможе модифікувати елемент X бази даних на диску, необхідно записати в журнал на диску запис оновлення **<Ti, Xid, New X val, Old X val>**

Відновлення: undo/redo

1. Повторити операції всіх зафіксованих трансакцій в порядку від ранніх до останніх.
2. Відмінити результати всіх незавершених трансакцій в зворотному порядку.

Введення контрольних точок

1. Внести в протокол запис виду <START CKPT T1…Tn> і виконати “скидання” протоколу на диск командою flush log.
2. Зберегти на диску інформацію всіх буферів, що містять “бруд”, тобто змінені значення, не “скинуті” на диск.
3. Зберегти в журналі запис <END CKPT> і виконати команду flush log

<T1, START>
<T1, A, 4, 5>
<T2, START>
<T1, COMMIT>
<T2, B, 9, 10>
<START CKPT T2>
<T2, C, 14, 15>
<T3, START>
<T3, E, 20, 25>
<END CKPT>
<T2, COMMIT>
<T3, COMMIT>