Overview of ARIES

DataLab
Introduction to Database Systems
2019 Spring
What we expected

In page 20:
- Tx1 is a winner tx
- Tx2 is a loser tx

Flush
However…

- Steal
  - Due to buffer management, dirty pages may be flushed to disk before txs commit
  -> The changes made by loser txs must be UNDO
However…

- Steal

- No Force
  - Due to performance reason, dirty page won’t be flush immediately after txs commit
  -> The changes made by *winner* txs must be REDO
Three Phases of ARIES

- Analysis Phase
  - Find the earliest possibly start point of dirty page
  - Find loser txs

- REDO Phase
  - Repeat history (both winner and loser changes)
  - Recovery exact page status when the failure occurred

- UNDO Phase
  - Rollback loser txs changes
Three Phases of ARIES

- **Analysis Phase**
  - Find the earliest possibly start point of dirty page
  - Find loser txs

- **REDO Phase**
  - Repeat history (both winner and loser changes)
  - Recovery exact page status when the failure occurred
Three Phases of ARIES

- **Analysis Phase**
  - Find the earliest possibly start point of dirty page
  - Find loser txs

- **REDO Phase**
  - Repeat history *(both* winner and loser changes)*
  - Recovery exact page status when the failure occurred

- **UNDO Phase**
  - Rollback *loser* txs changes

```
log
---
Analyse | REDO | UNDO
```
Logs in ARIES
Physical Log Record

• Record format :
  - Set Value Record:
    \(<\text{Op Code}, \text{txNum}, \text{fileName}, \text{blockNum}, \text{offset}, \text{sqlType}, \text{oldVal}, \text{newVal}>\)
  - Index Page Insert/Delete Record:
    \(<\text{Op Code}, \text{txNum}, \text{fileName}, \text{blockNum}, \text{insertSlot}, \text{insertKey}, \text{insertRidBlkNum}, \text{insertRidId}>\)

• REDO :

• UNDO :
Physical Log Record

- Record format:
  - Set Value Record:
    `<Op Code, txNum, fileName, blockNum, offset, sqlType, oldVal, newVal>`
  - Index Page Insert/Delete Record:
    `<Op Code, txNum, fileName, blockNum, insertSlot, insertKey, insertRidBlkNum, insertRidId>`

- **REDO**:
  - Apply `newVal` to the page

- **UNDO**:
Physical Log Record

• Record format:
  - Set Value Record:
    \(<\text{Op Code}, \text{txNum}, \text{fileName}, \text{blockNum, offset, sqlType, oldVal, newVal}>\)
  - Index Page Insert/Delete Record:
    \(<\text{Op Code, txNum, fileName, blockNum, insertSlot, insertKey, insertRidBlkNum, insertRidId}>\)

• REDO:
  - Apply newVal to the page

• UNDO:
  - Apply oldVal to the page
Physical Log Record

• Record format:
  - Set Value Record:
    \(<\text{Op Code}, \text{txNum}, \text{fileName}, \text{blockNum, offset, sqlType, oldVal, newVal}>\>
  - Index Page Insert/Delete Record:
    \(<\text{Op Code, txNum, fileName, blockNum, insertSlot, insertKey, insertRidBlkNum, insertRidId}>\>

• REDO:
  - Apply newVal to the page

• UNDO:
  - Apply oldVal to the page
  - Append its Compensation Log
Compensation Log Record

• CLRs are used to replay UNDO actions

• Feature :
  - REDO-ONLY Physical Log Record
  - UNDO procedure's REDO Log

• Record format :
  - Set Value Clr and Index Page Insert/Delete Clr:
    \[ \langle \text{Op Code}, \text{UndoTxNum} \ldots, \text{UndoNextLSN} \rangle \]

• REDO :
  - Apply oldVal to the page

• UNDO :
  - Do nothing
Why Clr is Redo Only?

[0] <Start 1>
[1] <SetVal, 1, Page 20, 0, 1>
[2] <SetVal, 1, Page 20, 1, 2>
[3] <SetVal, 1, Page 20, 2, 3>

Crash Here!

Redo

Analyse

tx1 is a loser tx
Why Clr is Redo Only?

[0] <Start 1>
[1] <SetVal, 1, Page 20, 0, 1>
[2] <SetVal, 1, Page 20, 1, 2>
[3] <SetVal, 1, Page 20, 2, 3>

Crash Here!

Why Clr is Redo Only?

[0] <Start 1>
[1] <SetVal, 1, Page 20, 0, 1>
[2] <SetVal, 1, Page 20, 1, 2>
[3] <SetVal, 1, Page 20, 2, 3>

Crash Here!

[4] <SetValClr 1, Page 20, 3, 2>

Crash Again!

Redo

Analyse
Why Clr is Redo Only?

[0] <Start 1>
[1] <SetVal 1, Page 20, 0, 1>
[2] <SetVal 1, Page 20, 1, 2>
[3] <SetVal 1, Page 20, 2, 3>

Crash Here!

[4]<SetValClr 1, Page 20, 3, 2 >

Crash Again!

[5]<SetValClr 1, Page 20, 2, 3 >

Why Clr is Redo Only?

[0] <Start 1>
[1] <SetVal, 1, Page 20, 0, 1>
[2] <SetVal, 1, Page 20, 1, 2>
[3] <SetVal, 1, Page 20, 2, 3>

Crash Here!

[4]<SetValClr 1, Page 20, 3, 2>

Crash Again!

[5]<SetValClr 1, Page 20, 2, 3>

Crash Again!
Why CLR is Redo Only?

[0] <Start 1>
[1] <SetVal, 1, Page 20, 0, 1>
[2] <SetVal, 1, Page 20, 1, 2>
[3] <SetVal, 1, Page 20, 2, 3>

Crash Here!

[4]<SetValClr 1, Page 20, 3, 2>

Crash Again!

[5]<SetValClr 1, Page 20, 2, 3>

Crash Again!

[6]<SetValClr 1, Page 20, 3, 2>

// Append Undo { Undo { Undo [3] Redo log } Redo log} Redo log

... WTF
Which level of REDO am I Undoing?
Clr is Redo Only!

[0] <Start 1>
[1] <SetVal, 1, Page 20, 0, 1>
[2] <SetVal, 1, Page 20, 1, 2>
[3] <SetVal, 1, Page 20, 2, 3>

Crash Here!

[4]<SetValClr 1, Page 20, 3, 2 >

Crash Again!

Redo

Undo

Analyse
Why Clr need UndoNext ?

• Clr without UndoNext :

[0] <Start 1>
[1] <SetVal , 1 , Page 20 , 0 , 1>
[2] <SetVal , 1 , Page 20 , 1 , 2>
[3] <SetVal , 1 , Page 20 , 2 , 3>

   Crash Here !


   Crash Again !

[7]<SetValClr 1, Page 20 , 1 , 0 >
Why Clr need UndoNext?

• Clr with UndoNext:

  [0] <Start 1>
  [1] <SetVal, 1, Page 20, 0, 1>
  [2] <SetVal, 1, Page 20, 1, 2>
  [3] <SetVal, 1, Page 20, 2, 3>

  Crash Here!


  Crash Again!

  [6]<SetValClr 1, Page 20, 1, 0, [1] >
Why Clr need UndoNext?

- UndoNext helps skip logs which have been Undone by Redo

  [0] <Start 1>
  [1] <SetVal 1, Page 20, 0, 1>
  [2] <SetVal 1, Page 20, 1, 2>
  [3] <SetVal 1, Page 20, 2, 3>

  **Crash Here!**


  **Crash Again!**

  [7]<SetValClr 1, Page 20, 1, 0, [1] >
Logical Log Record

• Record format :
  - Logical - Start Record
    <OP Code, txNum>
  - Record File Insert/Delete End Record :
    <Op Code, txNum, fileName, blockNum, slotId, logicalStartLSN>
  - Index Insert/Delete End Record :
    <Op Code, txNum, tblName, fldName, searchKey, recordBlockNum, recordSlotId, logicalStartLSN>

• REDO :
  - Do nothing

• UNDO :
  - Undo *competed* logical log *logically*
  - Undo *partial* logical log *physically*
  - Append *Logical Abort* log record
Rollback a competed Logical Log Record

[0] <Start 1>
[1] <LogicalStart, 1 >
[2] <Index Page Insert, 1, ... >
[3] <SetVal, 1, Page 2, 1, 2>
[4] <SetVal, 1, Page 20, 2, 3>

Crash Here!
Rollback a completed Logical Log Record

[0] <Start 1>

[1] <LogicalStart, 1 >

[2] <Index Page Insert, 1 , … >

[3] <SetVal, 1 , Page 2 , 1, 2>

[4] <SetVal, 1 , Page 20 , 2, 3>


Logical operations

Crash Here!
Rollback a competed Logical Log Record

[0] <Start 1>

[1] <LogicalStart, 1 >

[2] <Index Page Insert, 1, … >

[3] <SetVal, 1, Page 2, 1, 2>

[4] <SetVal, 1, Page 20, 2, 3>


Physical operations

Logical operations

Crash Here!
Rollback a completed Logical Log Record

[0] <Start 1>

[1] <LogicalStart, 1 >

[2] <Index Page Insert , 1 , … >

[3] <SetVal , 1 , Page 2 , 1 , 2>

[4] <SetVal , 1 , Page 20 , 2 , 3>


Crash Here!

[6] <Start 2 >

[7] <LogicalStart, 2 >

[8] <Index Page Delete , 2 , … >

[9] <SetVal , 2 , Page 2 , 2 , 1>

[10] <SetVal , 2 , Page 20 , 2 , 3>


[12] <Logical Abort 1 , [1]>

29
Rollback a competed Logical Log Record

[0] <Start 1>
[1] <LogicalStart, 1 >
[2] <Index Page Insert, 1, … >
[3] <SetVal, 1, Page 2, 1, 2>
[4] <SetVal, 1, Page 20, 2, 3>

Crash Here!

[6] <Start 2>

[7] <LogicalStart, 2 >

[8] <Index Page Delete, 2, … >
[9] <SetVal, 2, Page 2, 2, 1>
[10] <SetVal, 2, Page 20, 2, 3>

[12] <Logical Abort 1, [1]>

Physical operations

Logical operations