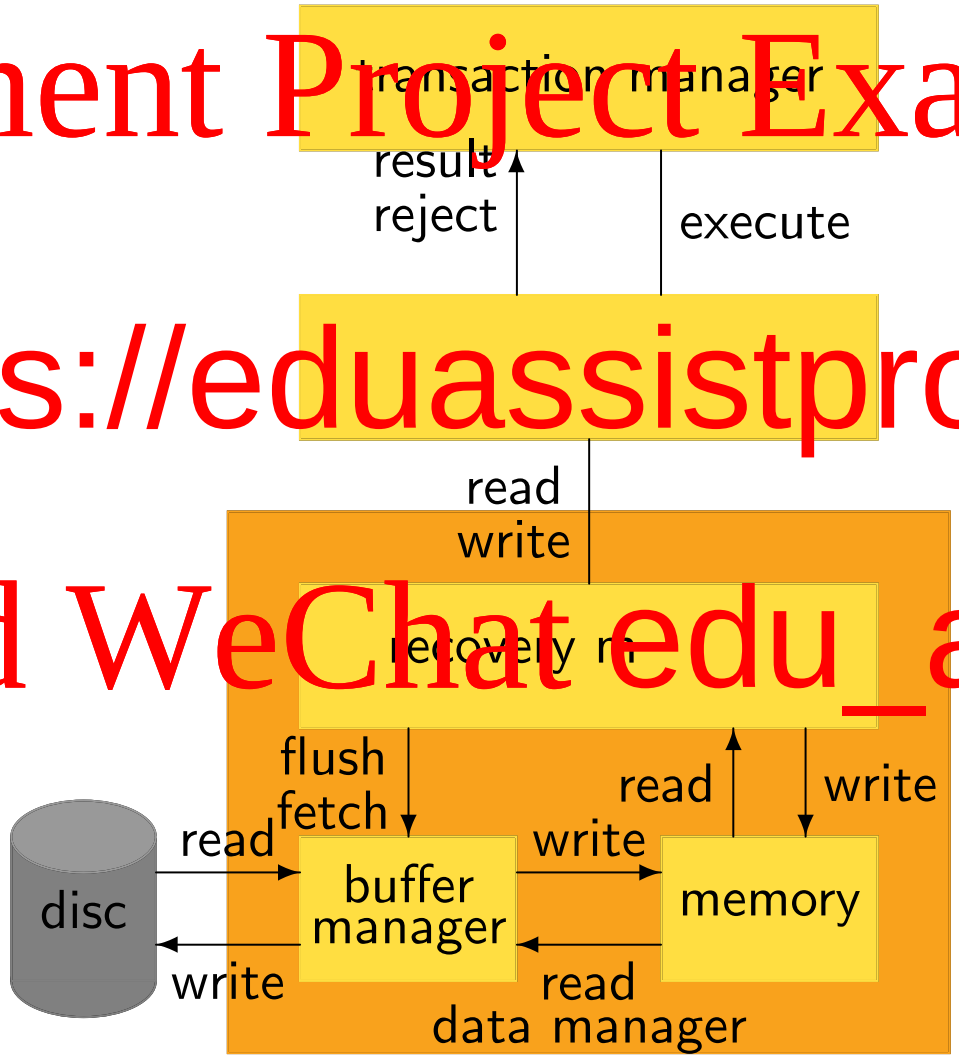
Recovery

Imperial College Lond

DBMS Architecture



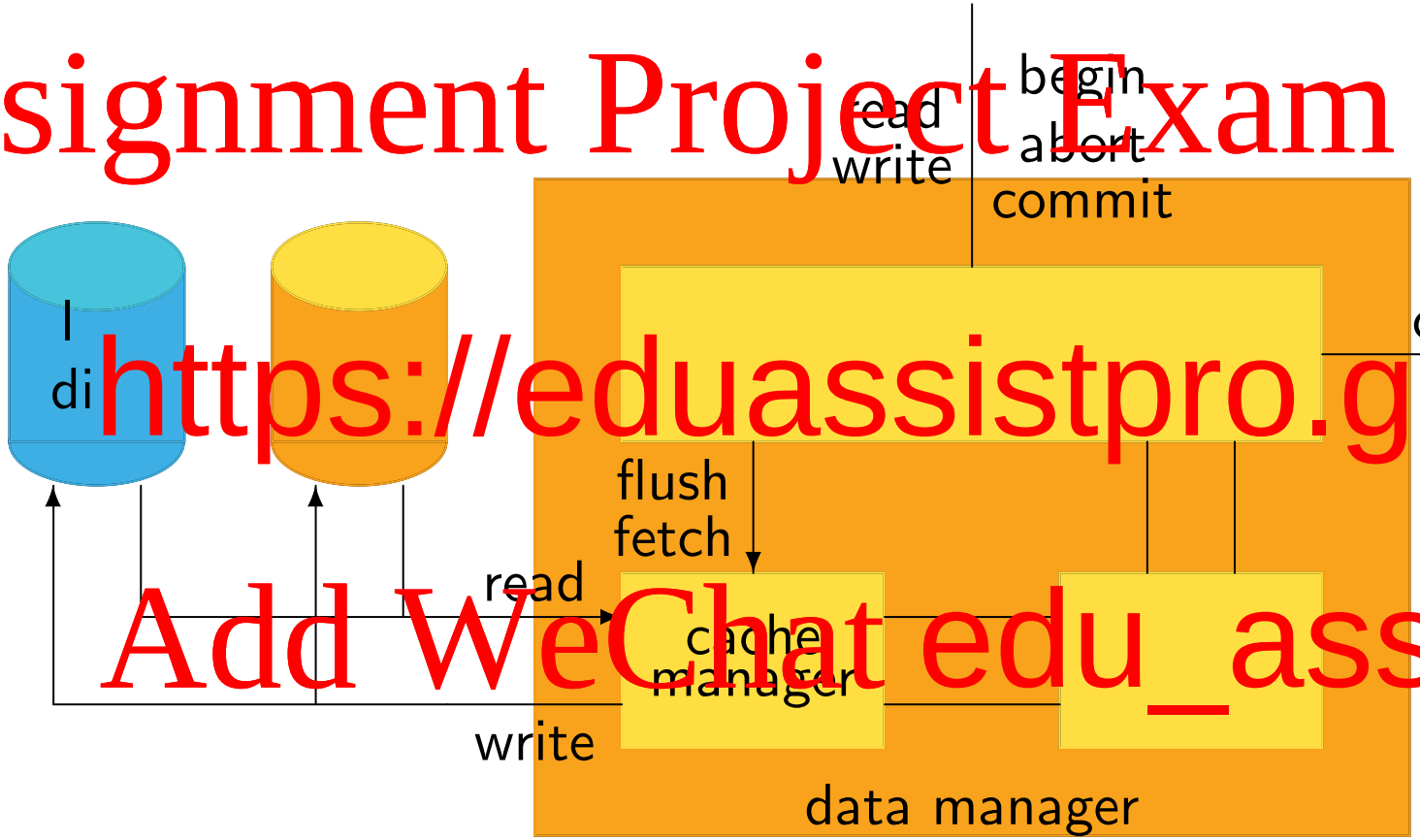
Recovery Manager (RM)

protect the DBMS against failures

- **sys**
  1.
  2. *OR*
  3. sufficient information such that (1) and (2) *recovery*

- **media failures** loss of stable storage

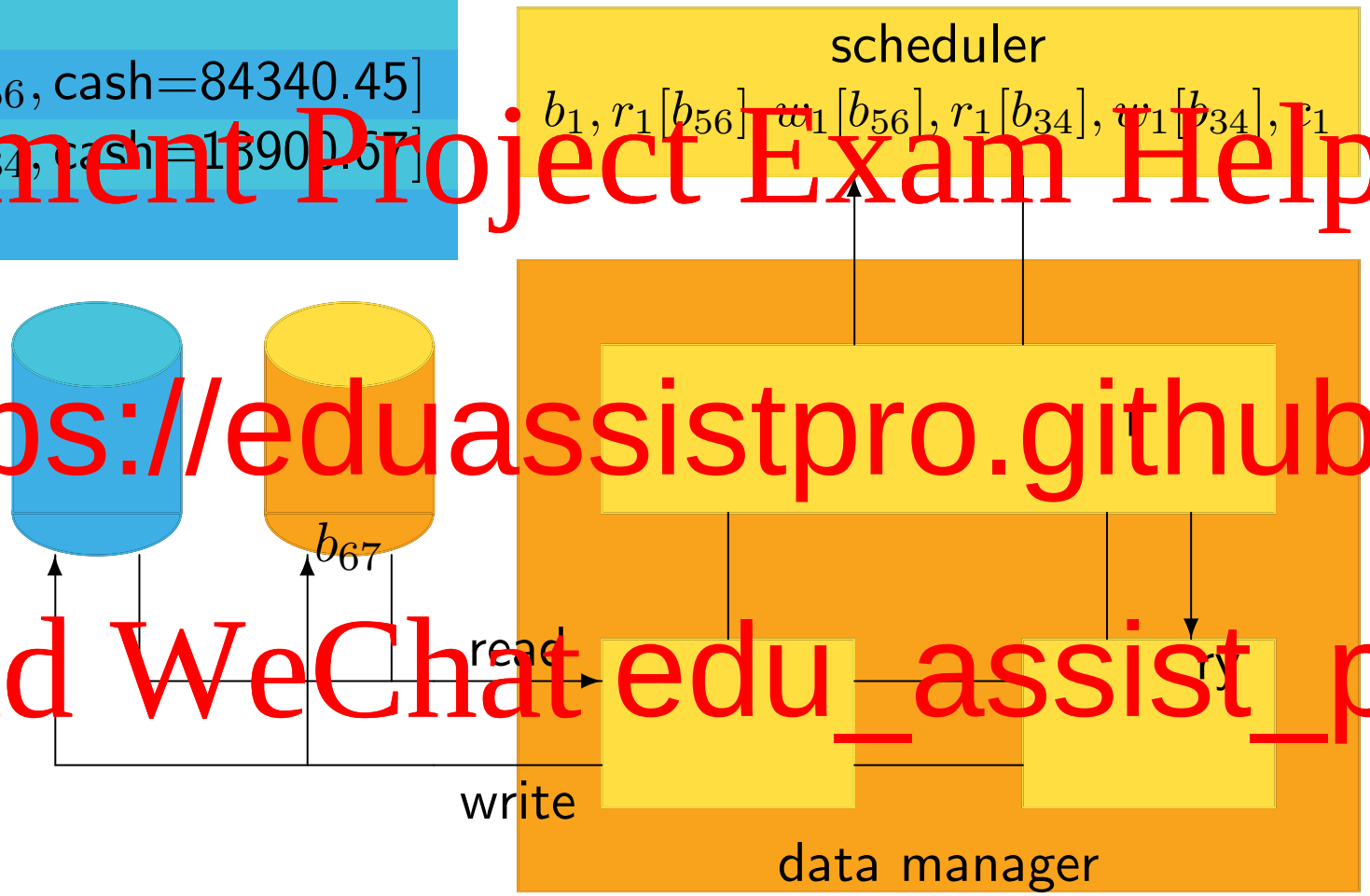
Enhanced Data Manager Architecture



- Need to cache log as well

Need to REDO

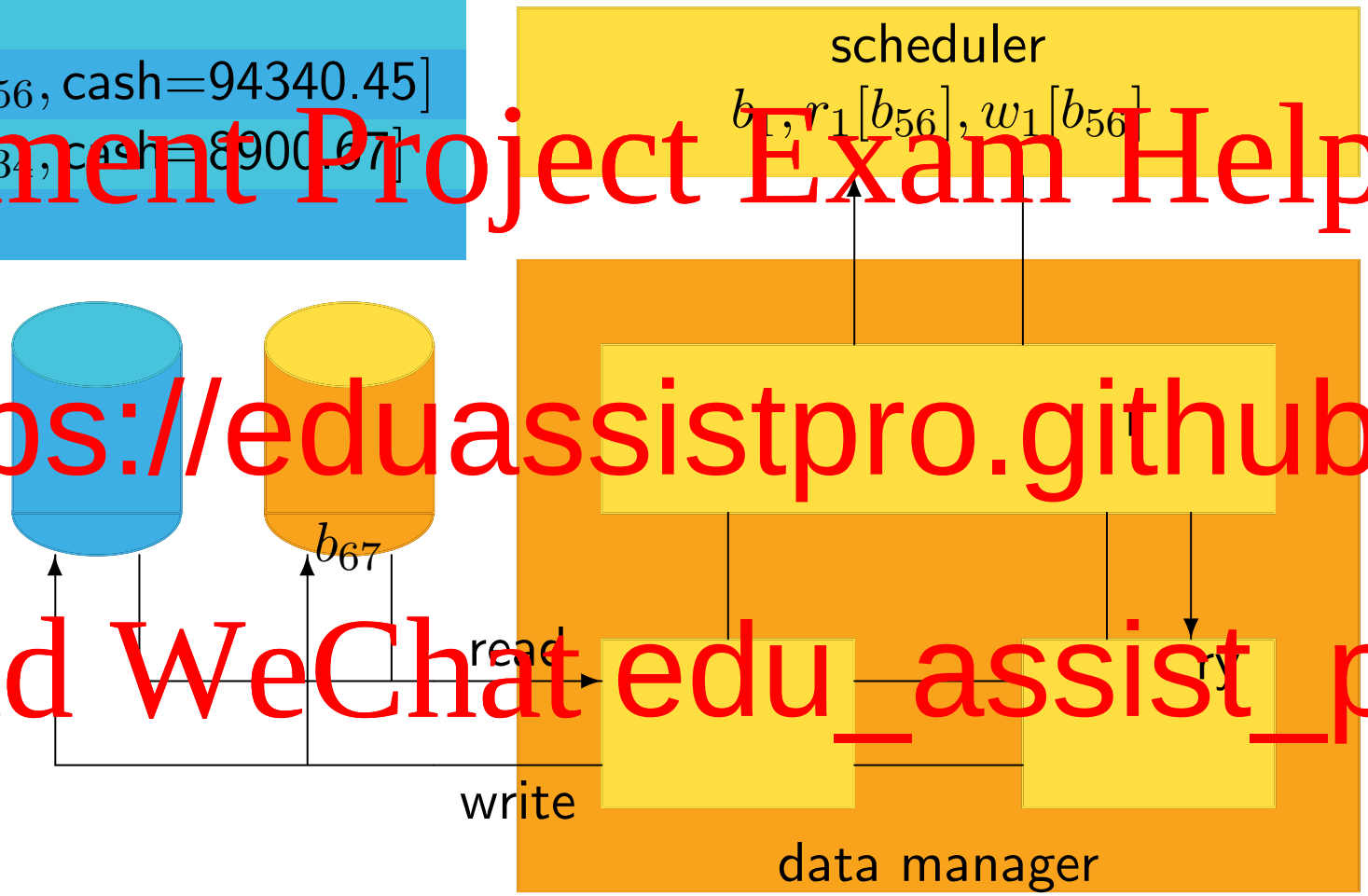
LOG	b_1
REDO	$w_1[b_{56}, \text{cash}{=}84340.45]$
REDO	$w_1[b_{34}, \text{cash}{=}13900.67]$
LOG	c_1



- REDO required if committed transactions not in stable storage
- must write all REDO to log before commit of transaction

Need to UNDO

LOG	b_1
UNDO	$w_1[b_{56}, \text{cash}=94340.45]$
UNDO	$w_1[b_{34}, \text{cash}=8900.67]$
LOG	c_1



- UNDO required if non-committed transactions in stable storage
- Must flush UNDO to log before corresponding write to data

Quiz 1: Contents of Data Disc After a Transaction

branch

sortcode	bname	cash
56	'Wimbledon'	94340.45
34	'Goodge St'	8900.67
67	'Strand'	34005.00

BEGIN TRANSACTION T1
UPDATE branch
SET cash=
WHERE so

UPDATE
SET cash=
WHERE so
COMMIT TRANSACTION T1

branch ①

sortcode	bname	cash
56	'Wimbledon'	94340.45
34	'Goodge St'	8900.67
67	'Strand'	34005.00

branch ③

sortcode	bname	cash
56	'Wimbledon'	84340.45
34	'Goodge St'	8900.67
67	'Strand'	34005.00

ranch ④

ame	cash
imbledon	84340.45
oodge St'	18900.67
trand'	34005.00

What must the contents of the branch after the transaction commits?

A	B	C	D
④	① or ④	①,③ or ④	①,②,③ or ④

Quiz 2: Contents of Log Disc After a Transaction

Data Disc Before Transaction

branch

sortcode	bname	cash
56	'Wimbledon'	94340.45
34	'	
67	'	

BEGIN TRANSACTION T1
UPDATE branch
SET cash=cash-10000.00
WHERE sortcode=56

UPDATE branch

Data Disc At Commit Time

branch

sortcode	bname	cash
56	'Wimbledon'	94340.45
	odge St'	18900.67
	and'	34005.00

What m

A
REDO r_{56}
REDO r_{34}
UNDO r_{56}
UNDO r_{34}

B
REDO r_{56}
UNDO r_{34}

C
UN

6

Before and after images

before image

branch

sortcode	bname	cash
56	'Wimbledon'	94340.45
3	'	
6	'	3

$w_1[b_{56}]$

$\Downarrow$

branch

sortcode	bname	cash
56	'Wimbledon'	84340.45
34	'Goodge St'	8900.67
67	'Strand'	34005.00

after image

- before image allows RM to **undo** $w_1[b_{56}]$
- after image allows RM to **redo** $w_1[b_{56}]$

Database Logs

LOG	b_1
REDO	$w_1[b_{56}, \text{cash}=84340.45]$
REDO	$w_1[b_{34}, \text{cash}=18900.67]$

LOG	b_1
UNDO	$w_1[b_{56}, \text{cash}=94340.45]$
UNDO	$w_1[b_{34}, \text{cash}=8900.67]$

LOG

LOG	b_1
UNDO	$w_1[b_{56}, \text{cash}$
REDO	$w_1[b_{56}, \text{cash}$
UNDO	$w_1[b_{34}, \text{cash}=8900.67$
REDO	$w_1[b_{34}, \text{cash}=18900.67]$
LOG	c_1

What must a complete REDO/UNDO log contain?

Must contain

- REDO information for each update
- UNDO information for each update
- co

Might co

- begin of each transaction
  - can be inferred from first REDO/
  - presence useful to stop search of
- abort of each transaction
  - can be inferred from lack of commit
  - presence useful to indicate UNDO already done

Rules for log and data updates

write ahead logging (WAL)

Redo rul

- co
- never respond to scheduler before log written

Undo rule:

- flushing uncommitted data → flush l

Basic Recovery Procedure

$\Rightarrow w_v[o_1], c_v, w_x[o_2], w_y[o_1], c_y, w_z[o_2] \Rightarrow$

1. UN
   -
   - Collect set of incomplete transactions $I = \{x, z\}$
   - Perform UNDO for any transaction in
2. REDO — *Scan forward through the log*
   -
   - Perform REDO for any transaction in $v \quad 1 \qquad y \quad 1$

$\Rightarrow \text{UNDO}(w_z[o_2]), \text{UNDO}(w_x[o_2]), \text{REDO}(w_v[o_1]), \text{REDO}(w_y[o_1]) \Rightarrow$

Example of Recovery

Log

LOG	b_4
LOG	o_1
UNDO	$w_1[b_{56}, \text{cash}{=}94340.45]$
REDO	$w_1[b_{56}, \text{cash}{=}84340.45]$
LOG	
UNDO	
REDO	
UNDO	$_2$ $_{67}$
REDO	$w_2[b_{67}, \text{cash}{=}36005.25]$
LOG	b_7
LOG	c_2
UNDO	$w_1[b_{34}, \text{cash}{=}8900.67]$
REDO	$w_1[b_{34}, \text{cash}{=}18900.67]$
UNDO	$w_7[b_{67}, \text{cash}{=}36005.25]$
REDO	$w_7[b_{67}, \text{cash}{=}37005.25]$
LOG	c_7
LOG	c_4

Disc Before Recovery

branch		cash
		84340.45
		18900.67
		34005.00

Dis

sor		cash
56	'Wimbledon'	94340.45
34	'Goodge St'	8900.67
67	'Strand'	37005.25

Omitting the REDO Log

If no REDO records kept

must flush committed transactions to data disc

1. C
2. Sca
3. commit entry $\rightarrow$ add to C
4. undo entry for member of $C \rightarrow$ add king changes to the data.
5. perform undo entry for object not of member D

Omitting the Undo Log

If no UNDO records kept

transact

- add
- commit is followed by flush or **unfix** of fixed objects

Omitting UNDO and REDO

atomic commit → out of place updating

Quiz 3: Contents of Disc Before Commit if no UNDO log

branch

sortcode	bname	cash
56	'Wimbledon'	94340.45
34	'Goodge St'	8900.67
67	'Strand'	34005.00

branch ①

sortcode	bname	cash
56	'Wimbledon'	94340.45
34	'Goodge St'	8900.67
67	'Strand'	34005.00

branch ③

sortcode	bname	cash
56	'Wimbledon'	84340.45
34	'Goodge St'	8900.67
67	'Strand'	34005.00

```
BEGIN TRANSACTION T1
    UPDATE branch
    SET cash=
    WHERE so

    UPDATE
    SET cash=
    WHERE so
COMMIT TRANSACTION T1
```

branch ④

ame	cash
imbledon	84340.45
oodge St'	18900.67
trand'	34005.00

What must the contents of the branch transaction commits?

A	B	C	D
①	① or ④	④	①,②,③ or ④

Quiz 4: Contents of Disc After Commit if no REDO log

branch		
sortcode	bname	cash
56	'Wimbledon'	94340.45
34	'Goodge St'	8900.67
67	'Strand'	34005.00

BEGIN TRANSACTION T1
UPDATE branch
SET cash=
WHERE so

UPDATE
SET cash=
WHERE so
COMMIT TRANSACTION T1

branch ①		
sortcode	bname	cash
56	'Wimbledon'	94340.45
34	'Goodge St'	8900.67
67	'Strand'	34005.00

branch ③		
sortcode	bname	cash
56	'Wimbledon'	84340.45
34	'Goodge St'	8900.67
67	'Strand'	34005.00

branch ④

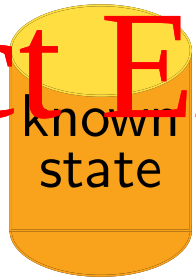
ame	cash
imbledon	84340.45
oodge St'	18900.67
trand'	34005.00

What must the contents of the branch transaction commits?

A	B	C	D
①	① or ④	④	①,②,③ or ④

Checkpointing

$\ldots, w_x[o_1] \Rightarrow$ 

$\Rightarrow cp \Rightarrow$ 

$\Rightarrow w_y[o_1], \ldots$

- For
- Recovery limited to only look back to checkp efore!)
  - speeds the recovery operation
  - limits the size of log
- The more consistent this known state
  - the easier it is to recover
  - the longer it takes to perform the checkpoint

Commit Consistent Checkpoint

Generating a Commit Consistent Checkpoint

1. Sto
2. Fin
3. Flu
4. Write a checkpoint to stable log.

- recovery now only needs to scan back to
- possible long hold-up at checkpoint ✖

Cache Consistent Checkpoint

Generating a Cache Consistent Checkpoint

1. Suspend all transactions
2. Flush all dirty cache objects to disc
3. Wr

Recove

1. perform UNDOs of non-committed trans
2. perform UNDO of non-committed tran ey were active at *cp*
3. perform REDOs of committed transactions after *cp*

- could still have delay whilst flushing cached objects

Worksheet: Cache Consistent Checkpoint

LOG	b_7
UNDO	$w_7[b_{67}, \text{cash=34005.25}]$
REDO	$w_7[b_{67}, \text{cash=37005.25}]$
LOG	b_2
UNDO	
REDO	
LOG	
UNDO	
REDO	$w_6[a_{101}, \text{rate=6.00}]$
LOG	b_1
UNDO	$w_1[b_{56}, \text{cash=94340.45}]$
REDO	$w_1[b_{56}, \text{cash=84340.45}]$
LOG	a_7
LOG	$cp\{1, 2, 6\}$
	$\vdots$

	$\vdots$
UNDO	$w_6[a_{119}, \text{rate=5.50}]$
REDO	$w_6[a_{119}, \text{rate=6.00}]$
	34005.00]
	36005.25]
	8
L	
U	0.67]
R	00.67]
LOG	b_9
UNDO	$w_9[b_{67}, \text{cash=36005.00}]$
REDO	$w_9[b_{67}, \text{cash=20000.00}]$
LOG	c_9

Fuzzy Checkpointing

Generating a Fuzzy Checkpoint

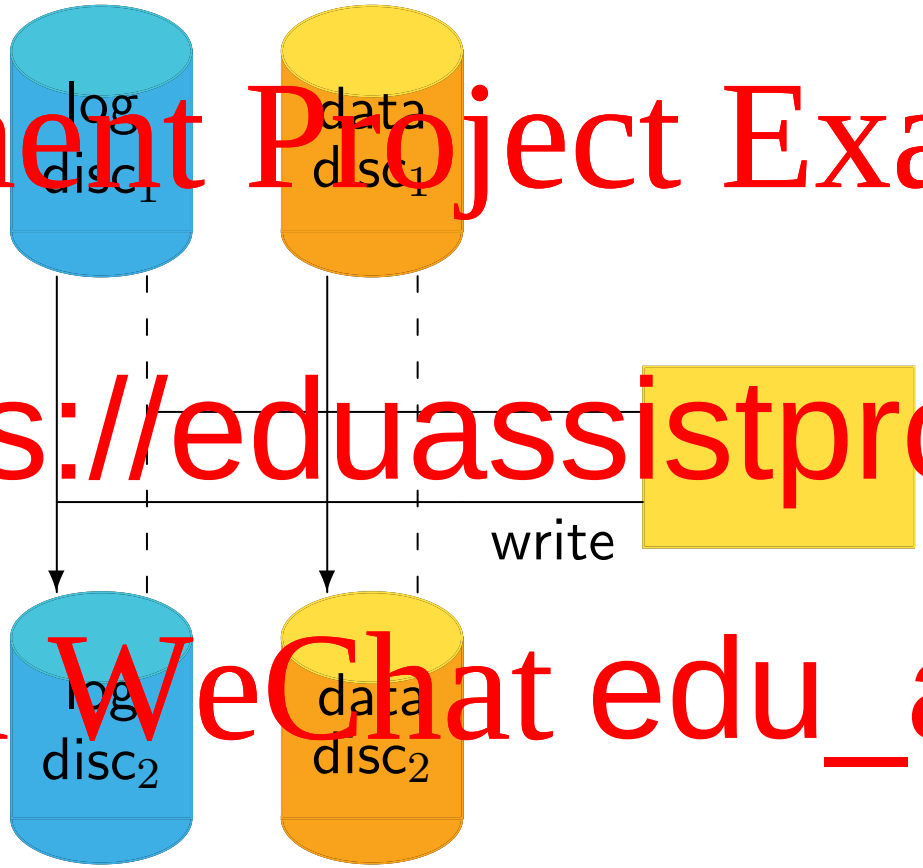
1. Suspend all transactions
2. Flush any dirty cache objects to disc not flushed in previous *cp*
3. Wr

Recove

Recovery works like cache consistent checkpoin h penultimate *cp*

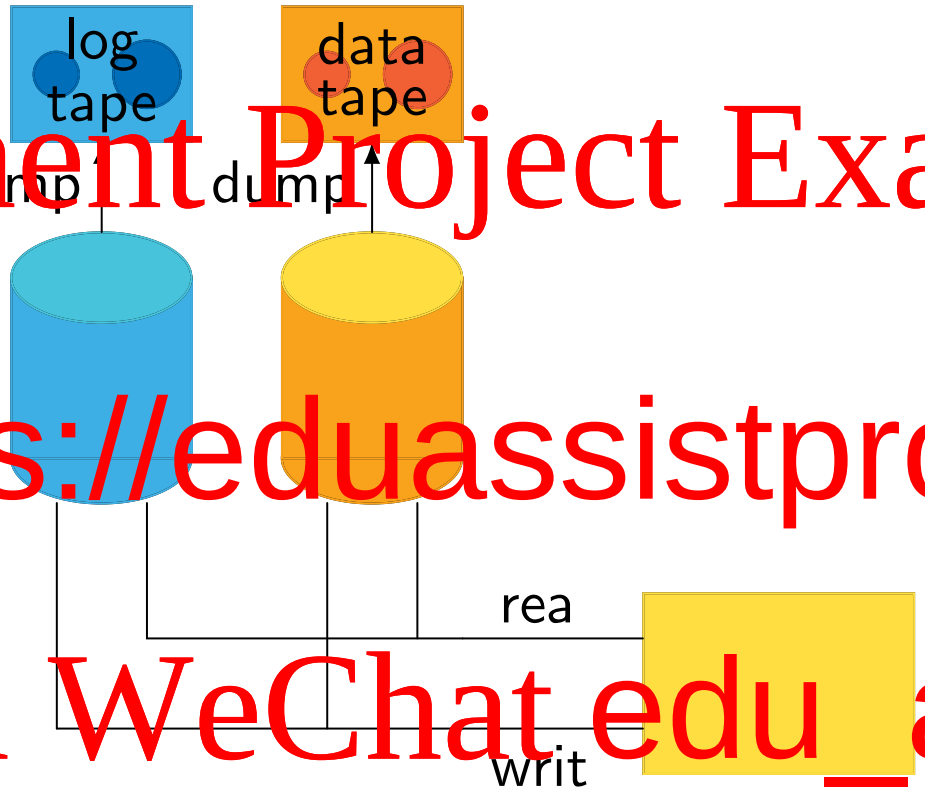
1. perform UNDOs of non-committed trans penultimate *cp*
2. perform UNDO of non-committed transactions before penultimate *cp* if they were active at *cp*
3. perform REDOs of committed transactions after penultimate *cp*

Media Failures: Mirroring (RAID-1)



- Keep more than one active copy of data and log
- Writes sent to both
- Read from either

Media Failures: Dumping

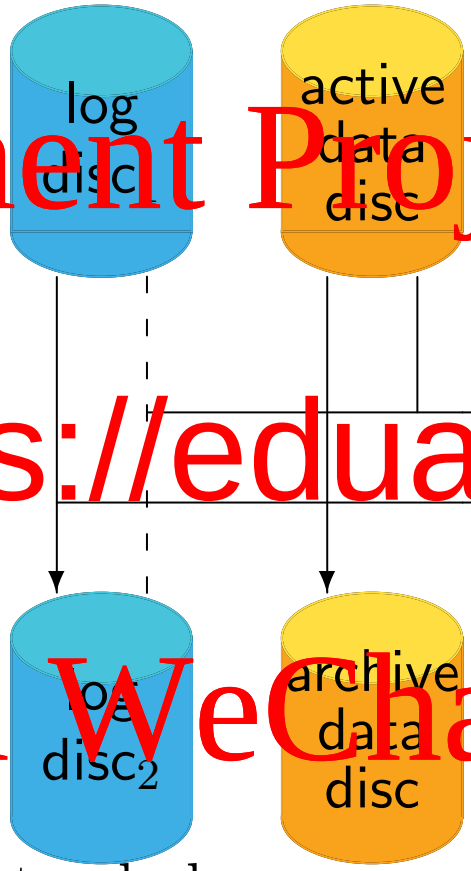


- 'tape' might also be a external file server, removable HD, *etc.*
- To use normal OS backup procedure
  - DBMS must not be still running
  - raw partition must not be used

Checkpoints and Dumps

- Du
- Res
  1. https://eduassistpro.github.i
  2. undo transactions active at the archive time
  3. redo transactions that committed after th
- commit consistent checkpoint obvious ch

Media Failures: Archive Database



- mirror log, but only have one active database
- periodically archive updates onto archive database
- failure of active database disc involves restore of archive database using logs

THE END

- Co es
- Rev
- 201
- Older exam questions mostly apply, but ther s on RA and SQL, less on concurrency