Transactions & recovery

CMT220
Databases & Modelling

Cardiff School of **Computer Science & Informatics**
http://www.cs.cf.ac.uk

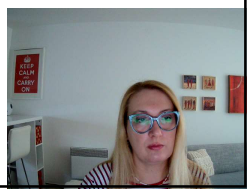
CARDIFF UNIVERSITY
PRIFYSGOL CAERDYDD

1

1

Lecture

- in the previous lecture we learned about database integrity
- database integrity is paramount and DBMS often include the ability to handle transactions to maintain the integrity of data
- in this lecture we introduce the concept of transaction as a basic unit of consistent and reliable computing in database systems
- we will also learn about the capability of a database to recover from various types of failures

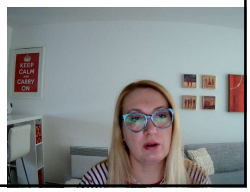


CARDIFF UNIVERSITY
PRIFYSGOL CAERDYDD

2

Transactions

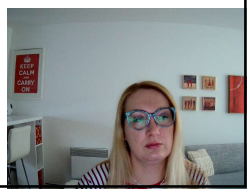
- a transaction is an action, or a series of actions, carried out by a single user or an application program, which reads or updates the contents of a database
- a transaction is a logical unit of work that transforms the database from one consistent state to another consistent state
- transactions are the units of:
  - recovery
  - consistency
  - integrity



3

ACID properties

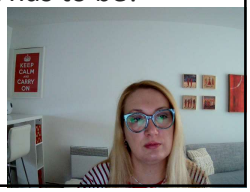
- a transaction must satisfy the ACID properties:
  - **A**tomicity
  - **C**onsistency
  - **I**solation
  - **D**urability



4

ACID properties

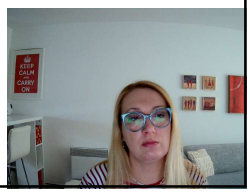
- atomicity
  - transactions do not have parts
  - transactions cannot be executed partially
- consistency
  - transactions take the database from one consistent state into another
  - note: midway through a transaction the database might not be consistent, but at the and it has to be!



5

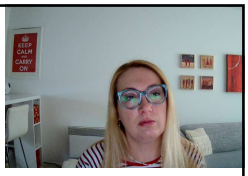
ACID properties

- isolation
  - the effects of a transaction are not visible to other transactions until it has completed
- durability
  - once a transaction is completed, its updates survive, even if there is a subsequent system crash



6

Example



- transaction: "transfer £50 from account A to account B"

1. read(A)
2. A = A – 50
3. write(A)
4. read(B)
5. B = B + 50
6. write(B)

- atomicity: should not take money from A without giving it to B
- consistency: no money lost or gained overall
- isolation: other queries should not see A or B change until completion
- durability: the money does not go back to A

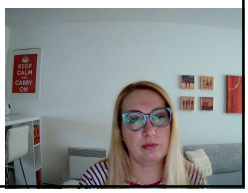
CARDIFF UNIVERSITY PRIFYSGOL CAERDYDD

7

7

Transaction manager

- transaction manager enforces the ACID properties
  - it schedules the operations of transactions
  - `COMMIT` and `ROLLBACK` are used to ensure atomicity
  - locks or timestamps are used to ensure consistency and isolation for concurrent transactions
  - a log is kept to ensure durability in the event of system failure

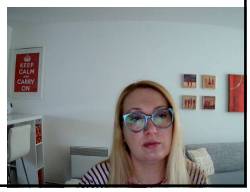


CARDIFF UNIVERSITY PRIFYSGOL CAERDYDD

8

COMMIT and ROLLBACK

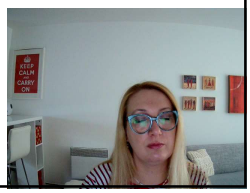
- `COMMIT` signals the successful end of a transaction
  - any changes made by the transaction are saved
  - these changes become visible to other transactions
- `ROLLBACK` signals the unsuccessful end of a transaction
  - any changes made by the transaction are undone
  - as if the transaction never occurred



9

Recovery

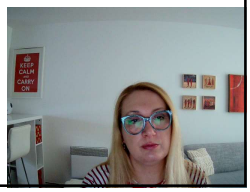
- in theory, transactions should be durable
- ... but in practice we cannot always prevent all types of failures, e.g.
  - system crash
  - power failure
  - disk crash
  - user mistake
  - sabotage
  - natural disaster



10

Transaction log

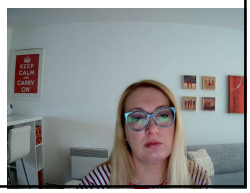
- transaction log records details of all transactions
  - any changes transactions make to the database
  - how to undo these changes
  - when transactions complete and how
- the log is stored on disk, not in memory
  - if the system crashes, then the log is preserved
- write ahead log rule
  - the entry in the log must be recorded before COMMIT



11

System failures

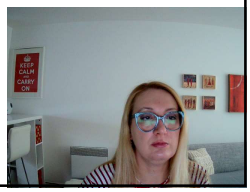
- a system failure affects all running transactions
  - software crash
  - power failure
- the physical media (disks) are not damaged
- at various times a DBMS takes a checkpoint
  - all committed transactions are written to disk
  - a record is made (on disk) of the transactions that are currently running



12

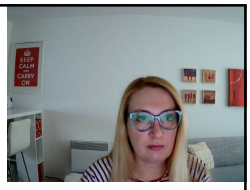
System recovery

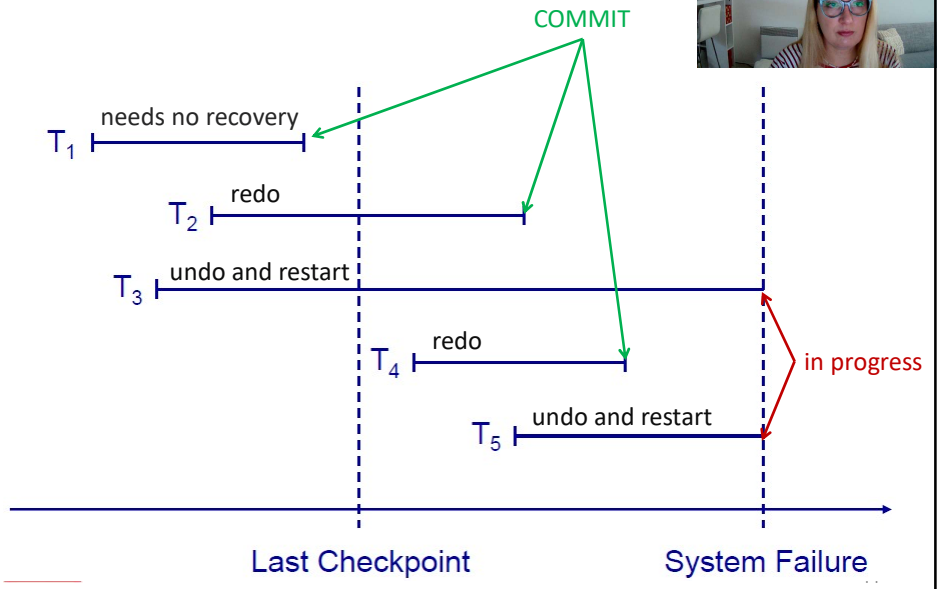
- any transaction that was running at the time of failure needs to be undone and restarted
- any transaction that committed since the last checkpoint needs to be redone



13

Types of transactions





14

Transaction recovery

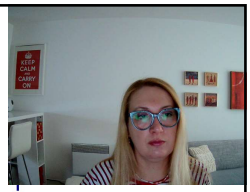
- two lists of transaction
  1. UNDO: all transaction that were in progress at the last checkpoint
  2. REDO: empty list
- for each entry in the log (at the last checkpoint) do:
  - if a BEGIN TRANSACTION entry is found for transaction T, then add T to UNDO
  - if a COMMIT entry if found for T, then move T from UNDO to REDO

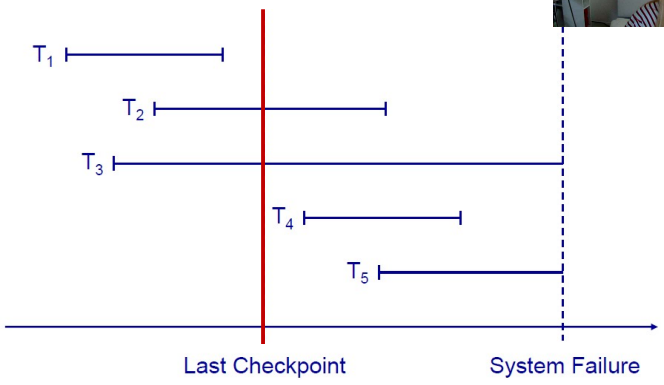
CARDIFF UNIVERSITY PRIFYSGOL CAERDYDD

15

15

Transaction recovery





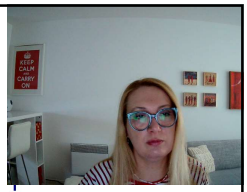
- UNDO: T_2, T_3
- REDO:
- active transactions: T_2, T_3
- add them to UNDO

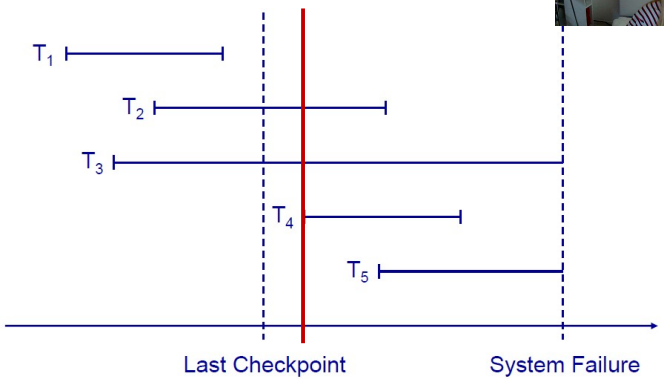
CARDIFF UNIVERSITY PRIFYSGOL CAERDYDD

16

16

Transaction recovery

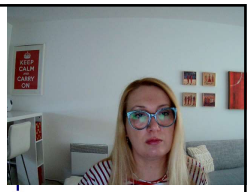


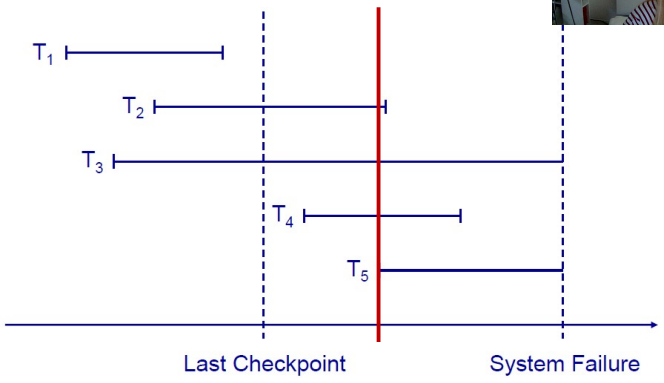


- UNDO: T_2, T_3, T_4
- REDO:
- T_4 begins
- add it to UNDO

17

Transaction recovery

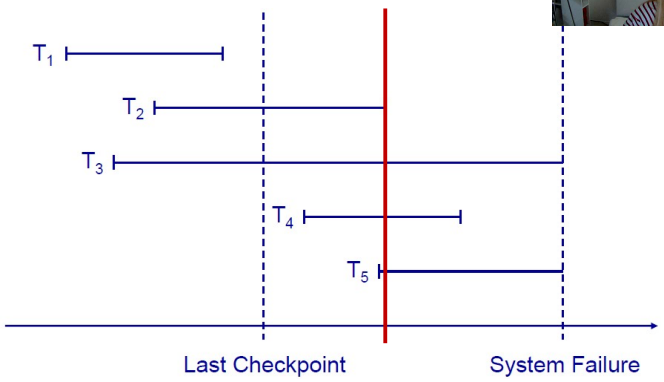




- UNDO: T_2, T_3, T_4, T_5
- REDO:
- T_5 begins
- add it to UNDO

18

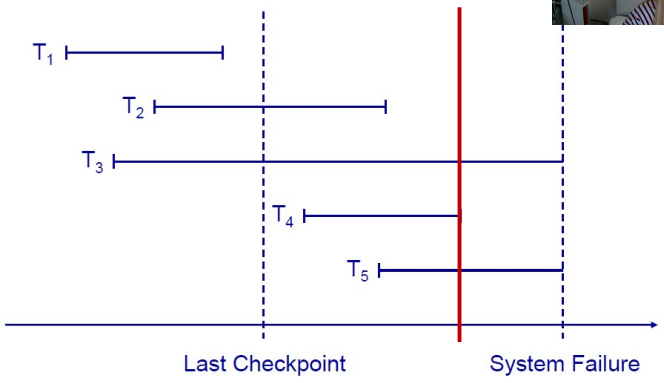
Transaction recovery



- UNDO: T_3, T_4, T_5
- REDO: T_2
- T_2 commits
- move it to REDO

19

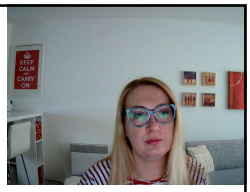
Transaction recovery



- UNDO: T_3, T_5
- REDO: T_2, T_4
- T_4 commits
- move it to REDO

20

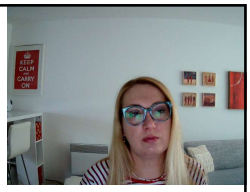
Forwards and backwards recovery



- backwards recovery
  - we need to undo some transactions
  - working backwards through the log we undo any operation by a transaction on the UNDO list
  - this returns the database to a consistent state
- forwards recovery
  - some transactions need to be redone
  - working forwards through the log we redo any operation by a transaction on the REDO list
  - this brings the database up to date

21

Media failures

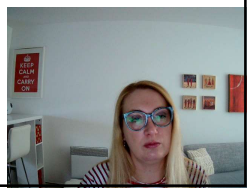


- system failures are not too severe
  - only information since the last checkpoint is affected
  - this can be recovered from the transaction log
- media failures (e.g. disk crash) are more serious
  - the data stored on disk is damaged
  - the transaction log itself may be damaged

22

Backups

- backups are needed to recover from media failure
  - the transaction log and entire contents of the database is written to secondary storage (e.g. tape)
  - time consuming and thus often requires down time
- backup frequency
  - frequent enough to minimise information loss
  - not too frequent as to cause problems
  - daily backup (e.g. overnight) is common

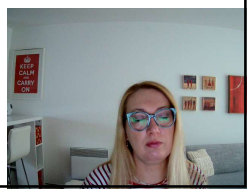


23

Recovery from media failure

1. use the last backup to restore the database
2. use the transaction log to redo any changes made since the last backup

- if the transaction log is damaged, we cannot do step 2
- store the transaction log and the database on separate physical devices
- this minimises the risk of losing both



24