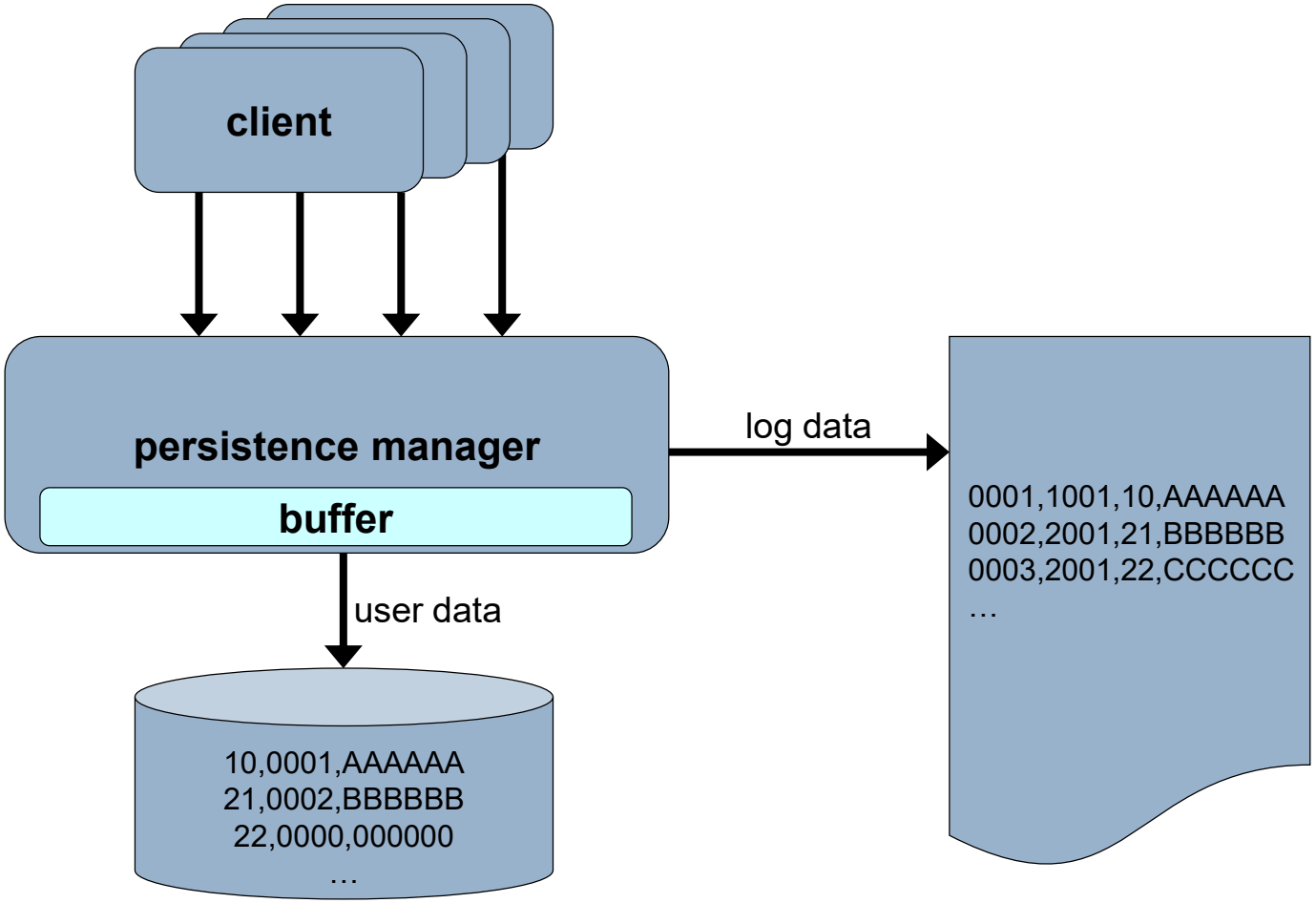
Logging and Recovery

DIS Exercise Course

Task

- implementing a simplified „database system“
- several clients access one single persistence
- buffer management
  - Non-Atomic
  - No-Steal
  - No-Force
- logging during normal operation
  - physical state logging
- crash recovery
  - redo recovery only
  - re-issuing lost modifications

Architecture



Persistence Manager

- persisting pages
  - in files, not in DB2
  - advice: one file per page
- page structure: [PageID, LSN, Data]
  - PageID: Page identifier
  - LSN: Log Sequence Number
  - Data: user data
- buffer: delayed writes
  - persisting data of **commited** transactions, when there are more than five pages in the buffer (No-Force)
  - check for a „full" buffer after every write access
  - refreshing pages in the buffer is possible
  - no dirty pages are persisted (No-Steal)
  - user data are overwritten directly (Non-Atomic)
- modifying operations are logged immediately
- requirement: support of crash recovery

Persistence Manager: Logging

- physical state logging
- granulate: Page
- new states (after-images) of modified objects are written to the log file
- record types
  - BOT and commit record
  - modification records (after-images)
- structure of log entries: [LSN, TAID, PageID, Redo]
  - LSN: Log Sequence Number (monotonically ascending)
  - TAID: transaction ID
  - PageID: Page ID
  - Redo: information required for redo

Clients

- 5 clients of the same kind
  - concurrent threads with ClientIDs 1..5
- access to persistence manager
  - persistence manager as Singleton: a single instance
  - clients write pages like this:
    `beginTransaction() write() write()… commit()`
  - no concurrent access on the same object → no locks required
    (Client 1: pages 10..19, Client 2: pages 20..29 etc.)
  - TAID is assigned by the persistence manager

Crash Recovery

- a record in the DB is either
  - up-to-date
  - or stale (noforce) → redo
- analysis phase
  - read the entire log sequentially
  - determine winner TAs
- redo phase
  - read the entire log sequentially
  - selective redo (redo winners)
- recovery progress is logged by updating the LSNs of the completed redo steps in the corresponding pages

Crash Recovery

- redo is only required, if
  page LSN < LSN of redo log record
- page LSN is updated on redo (grows monotonically)

```
if LSN(LS)  >  LSN(Page) then
  redo (modification from LS);
  LSN(Page) := LSN(LS);
end;
```