THE UNIVERSITY of EDINBURGH

Advanced Database Systems

Spring 2026

Q&A Session 4

If you require this document in an alternative format, such as large print or a coloured background, please contact milos.nikolic@ed.ac.uk

ADMINISTRIVIA

Practice Worksheet 4 is now available on Learn

ADBS Exam

- Date & time available at https://exams.is.ed.ac.uk/
- **Important: Bring your calculator!**
- Before the exam:
  - Check which calculators are permitted at https://edin.ac/4bOaZ9H
  - Ensure your device complies with exam regulations

Surveys, Surveys, Surveys…

Course survey – please complete

- Your feedback really matters
- Helps us improve the course for future students (just like past feedback shaped this one)

UG students – please complete the NSS

- The School takes your feedback seriously
- Coffee vouchers available at upcoming events

Question 1

Q1: Two-Phase Commit with Logging

Scenario

1 coordinator, 3 participants

Communication delays

Coordinator → participants: **30 ms** (simultaneous delivery)

P1 → Coord: **5 ms** P2 → Coord: **10 ms** P3 → Coord: **15 ms**

Logging cost

Log write + flush: **10 ms**

Questions

(a) Total time for successful commit

(b) Crash after YES vote

(c) Crash scenarios (YES vs NO, with/without presumed abort)

(d) Coordinator crash scenarios

Recap: 2PC (Normal Mode)

Goal: Ensure all participants either commit or abort consistently

Before 2PC begins:

- Transaction operations are typically already executed locally
- Changes are not yet durable / visible (pending commit)

Phase 1: Prepare (Voting)

- Coordinator sends **PREPARE**
- Participants decide if they can commit and respond to coordinator:
  - **YES** (ready to commit)
  - **NO** (cannot commit)

Phase 2: Commit (Decision)

- Coordinator sends final decision to all participants
  - All YES → **COMMIT**
  - Any No → **ABORT**
- Participants confirm completion by returning **ACK**

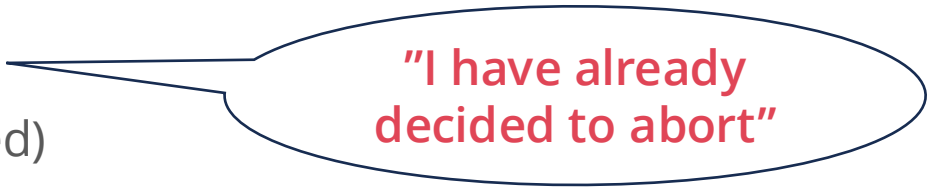
Participant Logging & States

If participant votes **YES**:

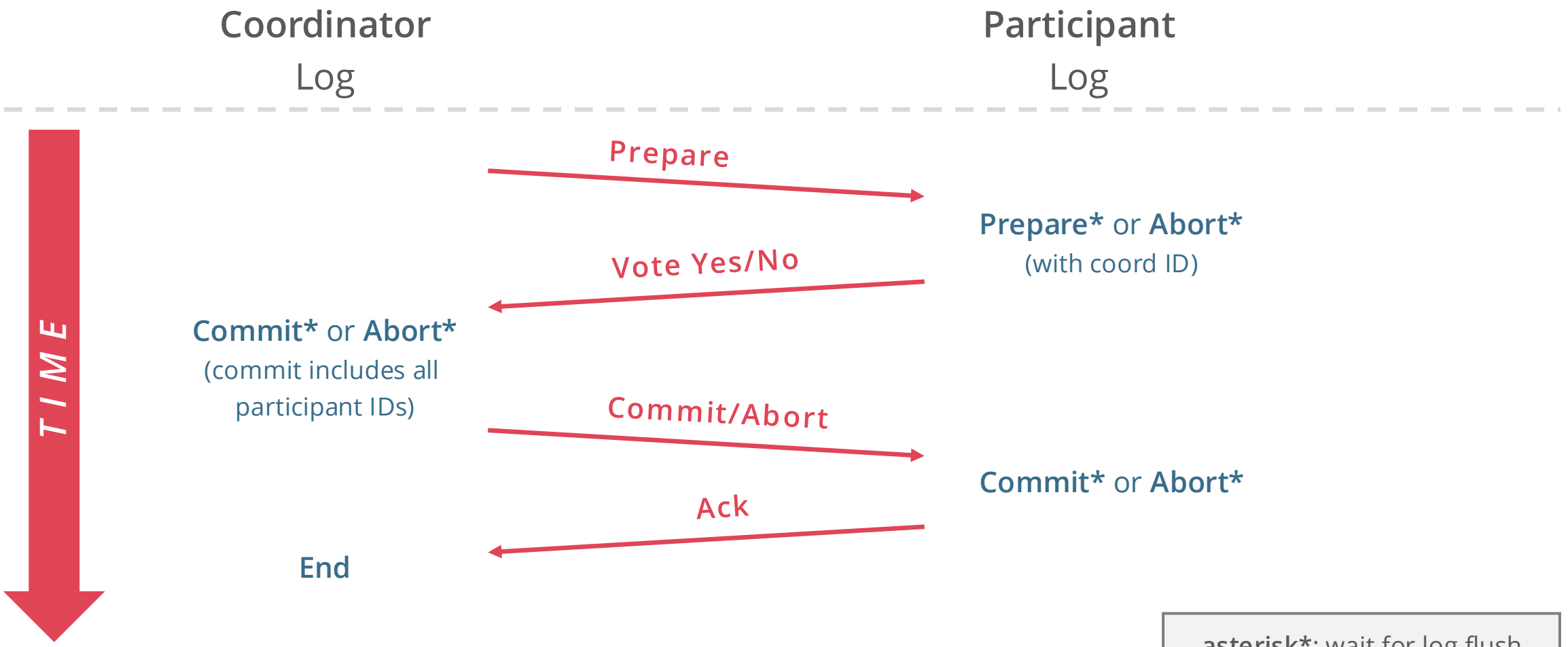
- Writes **PREPARE** record to stable storage
- Keeps locks and resources
- Enters uncertain state – relinquished the right to unilaterally abort, waits for coordinator's decision
- Must be able to commit after recovery

If participant votes **NO**:

- Aborts immediately, undoing local changes (if needed)
- Releases locks
- Sends **NO** to coordinator
- Flushes **ABORT** record



2PC In a Nutshell



asterisk*: wait for log flush before sending next msg

2PC Optimisation: Presumed Aborts

Core idea

If the coordinator has no record of a transaction → presume **ABORT**

What this eliminates (abort case)

Coordinator skips forced **ABORT** log write

Participants do not send **ACKs** after **ABORT**

Coordinator does no cleanup for aborted transactions (no need to write **END** records for aborts)

Why this works

Participants that vote **NO**:

- Abort immediately
- Typically write no durable record

Therefore: missing state safely ⇒ **ABORT**

Savings apply only to abort cases. Commit path unchanged (still requires logging + messages)

Q1A: Timing

Communication delays

Coordinator → participants: **30 ms** (simultaneous delivery)

P1 → Coord: **5 ms** P2 → Coord: **10 ms** P3 → Coord: **15 ms**

Logging cost

Log write + flush: **10 ms**

Question

Under 2PC with logging, how long does the entire protocol take for a successful commit?

Q1A: Timing

Communication delays

Coordinator → participants: **30 ms** (simultaneous delivery)

P1 → Coord: **5 ms** P2 → Coord: **10 ms** P3 → Coord: **15 ms**

Logging cost

Log write + flush: **10 ms**

Question

Under 2PC with logging, how long does the entire protocol take for a successful commit?

Phase 1: C sends **PREPARE** + Ps flush **PREPARE** + Ps send **YES** + C flushes **COMMIT**

= 30 + 10 + max { 5, 10, 15 } + 10 = 65 ms

Phase 2: C sends **COMMIT** + Ps flush **COMMIT** + Ps send **ACK** + C flushes **END**

= 30 + 10 + max { 5, 10, 15 } + 10 = 65 ms

Total: 130 ms

Q1B: Failure After YES Vote

Question

- Participant
  - Voted **YES**
  - Crashes and restarts
- All others voted **YES**

What happens?

Q1B: Failure After YES Vote

Question

- Participant
  - Voted **YES**
  - Crashes and restarts
- All others voted **YES**

What happens?

On recovery

- Participant finds **PREPARE** record
- Enters prepared (uncertain) state
- Cannot decide alone → must contact coordinator

Coordinator

- Cannot finish until all **ACKs** received
- Knows transaction committed

Outcome

- Coordinator replies **COMMIT**
- Participant writes **COMMIT** record and sends **ACK**

Q1C.i: Crash After YES Vote

Question

P1 votes **YES**, then crashes before decision

P2 votes **NO** → transaction aborts

What does P1 do after recovery

1. Without presumed abort
2. With presumed abort

Q1C.i: Crash After YES Vote

Question

P1 votes **YES**, then crashes before decision

P2 votes **NO** → transaction aborts

What does P1 do after recovery

1. Without presumed abort
2. With presumed abort

P1 finds **PREPARE** record

Must contact coordinator

Coordinator replies: **ABORT**

P1 aborts transaction locally

Same behaviour w/ or w/o presumed abort

Q1C.II: Crash After No Vote

Question

P2 votes **NO**, then crashes before decision

What does P2 do after recovery

1. Without presumed abort
2. With presumed abort

Q1C.II: Crash After No Vote

Question

P2 votes **NO**, then crashes before decision

What does P2 do after recovery

1. Without presumed abort
2. With presumed abort

Without presumed abort

- P2 has **ABORT** record log
- May resend **NO** vote
  - Not required for correctness
  - Helps coordinator finish faster
- Aborts locally

With presumed abort

- **ABORT** record may be missing
- No record → **presume abort**
- P2 aborts locally and sends no messages

Q1D: Coordinator Crashes Early

Scenario

Coordinator sends **PREPARE**

Crashes before receiving votes

Using presumed abort

Questions

What does coordinator do after recovery?

What does a **NO** voter do?

What does a **YES** voter do?

Q1D.i: Coordinator Recovery

Scenario

Coordinator sends **PREPARE**

Crashes before receiving votes

Using presumed abort

Questions

What does coordinator do after recovery?

What does a **NO** voter do?

What does a **YES** voter do?

Q1D.i: Coordinator Recovery

Scenario

- Coordinator sends **PREPARE**
- Crashes before receiving votes
- Using presumed abort

Questions

- What does coordinator do after recovery?
- What does a **NO** voter do?
- What does a **YES** voter do?

No **COMMIT** record in log

With presumed abort

- No record → ABORT

Coordinator

- Aborts locally
- Writes **ABORT** record
- Does not notify participants

Q1D.II: Participant Voted NO

Scenario

Coordinator sends **PREPARE**

Crashes before receiving votes

Using presumed abort

Questions

What does coordinator do after recovery?

What does a **NO** voter do?

What does a **YES** voter do?

Q1D.II: Participant Voted NO

Scenario

- Coordinator sends **PREPARE**
- Crashes before receiving votes
- Using presumed abort

Questions

- What does coordinator do after recovery?
- What does a **NO** voter do?
- What does a **YES** voter do?

NO vote = veto

Participant

- Already knows outcome = ABORT
- Aborts locally
- Ignores coordinator failure

Q1D.iii: Participant Voted YES

Scenario

Coordinator sends **PREPARE**

Crashes before receiving votes

Using presumed abort

Questions

What does coordinator do after recovery?

What does a **NO** voter do?

What does a **YES** voter do?

Q1D.iii: Participant Voted YES

Scenario

- Coordinator sends **PREPARE**
- Crashes before receiving votes
- Using presumed abort

Questions

- What does coordinator do after recovery?
- What does a **NO** voter do?
- What does a **YES** voter do?

Participant

- Voted **YES** → prepared state
- Cannot decide alone
- Periodically try to contact coordinator

Coordinator (after recovery)

- Returns **ABORT**

Participant

- Aborts transaction