Test 2 Detailed Topics

TP09

r be a relation:

n(r) = number of tuples in r b(r) = number of blocks occupied by r f(r) = number of tuples per block in r = n(r) / b(r) V(A,r) = number of distinct values of attribute A in r

Values of A are uniformly distributed.

on average, each value of A appears n(r) / V(A,r) times in r.

|σA=c(r)| = n(r) / V(A,r)

if A is a key, then V(A,r) = n(r) and |σA=c(r)| = 1

$$|\sigma_{A<k}(r)| = n_r \cdot \frac{k-1}{N}$$

$$|\sigma_{A\geq k}(r)| = n_r \cdot \frac{N-k+1}{N}$$

5.1 Join on attribute A

Example:

sql Copy code

```
r ⋈ s ON r.A = s.A
```

Slide formula

$$|r \bowtie s| = \min\left(\frac{n_r \cdot n_s}{V(A,r)}, \frac{n_r \cdot n_s}{V(A,s)}\right)$$

If A is key in r → $|r \bowtie s| \leq n_s$

If A is key in s → $|r \bowtie s| \leq n_r$

TP11

Two-Phase Locking (2PL)

A transaction has two phases:

1. Growing phase: acquires locks, releases none
2. Shrinking phase: releases locks, acquires none

If it releases a lock and later acquires another -> NOT 2PL

Strict 2PL

A transaction must hold all its **exclusive locks** until commit/abort.

Rigorous 2PL All locks (S and X) are held until commit/abort.

schedule is serializable if equivalent to a serial schedule. schedule is strict. recovery is easy (because no transaction can read uncommitted data so there's no cascading aborts).

Strict -> Cascadeless -> Recoverable

TP12

Write-Ahead Logging (WAL) - Before a data page is written to disk, all log records describing changes to that page must be written to stable storage.

AFTER A CRASH:

UNDO TRANSACTIONS THAT HAVE TI START BUT NOT TI COMMIT

REDO TRANSACTIONS THAT HAVE TI COMMIT.

Immediate update: changes are written to disk as soon as they are made. Deferred update: changes are written to disk only at commit time.

Normal checkpoint:

1. Stop updates
2. Flush all log records
3. Flush all dirty pages
4. Write <checkpoint {T1, T2, ...}> to log

- Everything before the checkpoint is guaranteed to be on disk.

Fuzzy checkpoint (modern systems):

- Do not stop all transactions
- Log contains a pointer to the last fuzzy checkpoint record
- During recovery, start from the last checkpoint and go forward
- Transactions active at checkpoint must be undone if not committed
- Transactions that committed after checkpoint must be redone

T13 - Distributed Transactions

- A distributed transaction runs on multiple sites / databases
- Must still satisfy ACID properties

Problem:

- Sites can fail independently
- Messages can be lost or delayed

Two-Phase Commit (2PC) Protocol

- Coordinator sends PREPARE to all participants

Each participant writes <Ti, PREPARE> to log and replies YES (can commit) or NO (must abort))

After voting YES, participant is in uncertain state: it has promised to commit but not yet committed.

IF ALL VOTE YES:

- coordinator writes to log
- sends COMMIT to all participants

IF ANY VOTE NO:

- coordinator writes to log
- sends ABORT to all participants
- Participants follow coordinator's decision and log it

Main Drawbacks of 2PC:

2PC IS BLOCKING

- If the coordinator crashes after sending PREPARE but before sending COMMIT/ABORT, participants cannot decide on their own
- they must wait or ask other participants (termination protocol)
- Coordinator fails after senting COMMIT to site A, but before sending it to site B. what does B do?

1. Is site B prepared?
    - NO -> ABORT
    - YES -> cannot decide
2. Runs termination protocol: asks other participants
    - If any says ABORT -> ABORT
    - If all say COMMIT -> COMMIT
    - If some are uncertain -> wait