THE UNIVERSITY OF MELBOURNE

COMP90050 Advanced Database Systems

Winter Semester, 2023

Lecturer: Farhana Choudhury (PhD)

Live lecture – week 4

Q/A

- Last quiz questions
- Group presentations happening
- Group report submission

Crash recovery

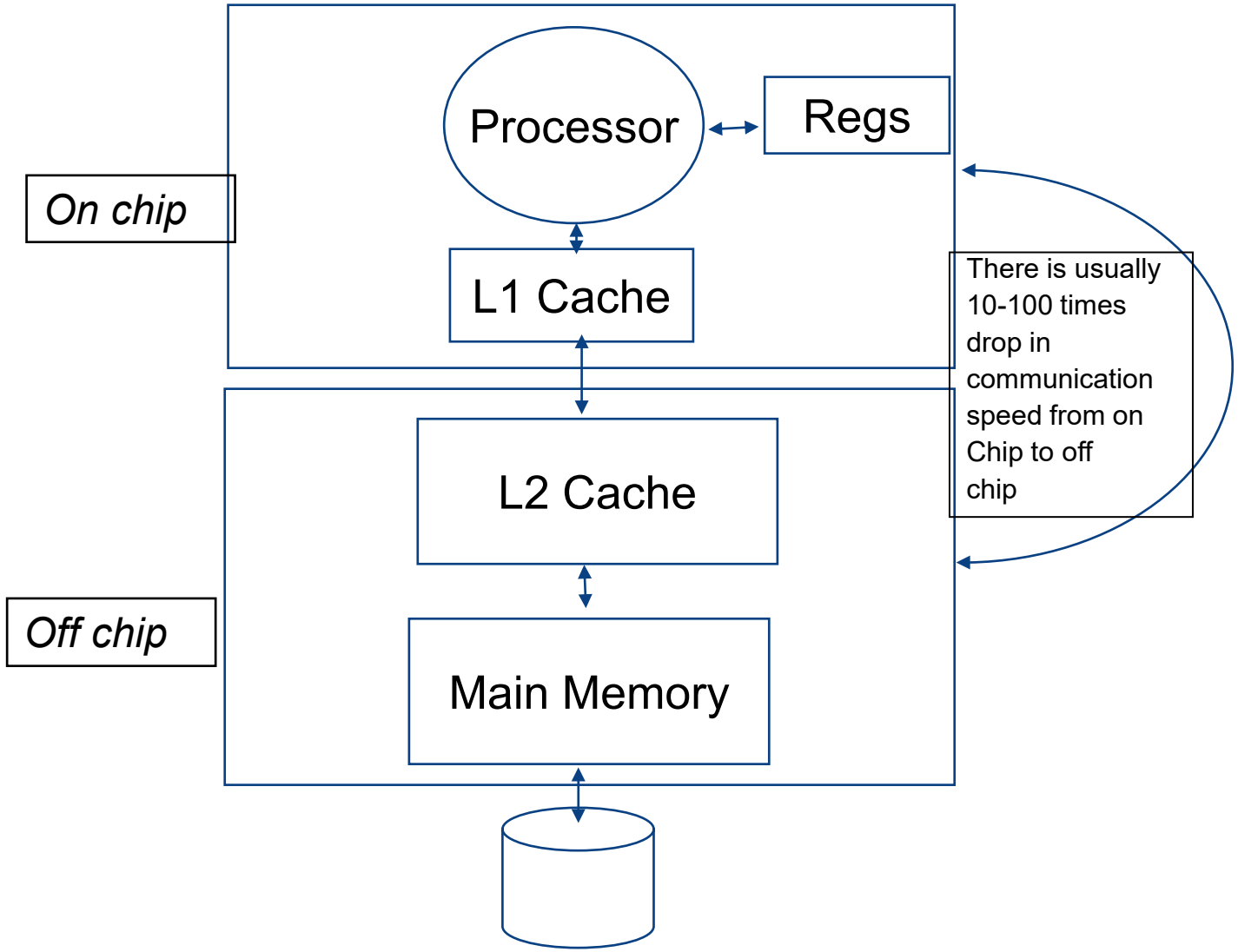
What needs to be recovered if a crash happens?

- Has it been made durable - good!
- If not durable – what additional information are needed to recover them?

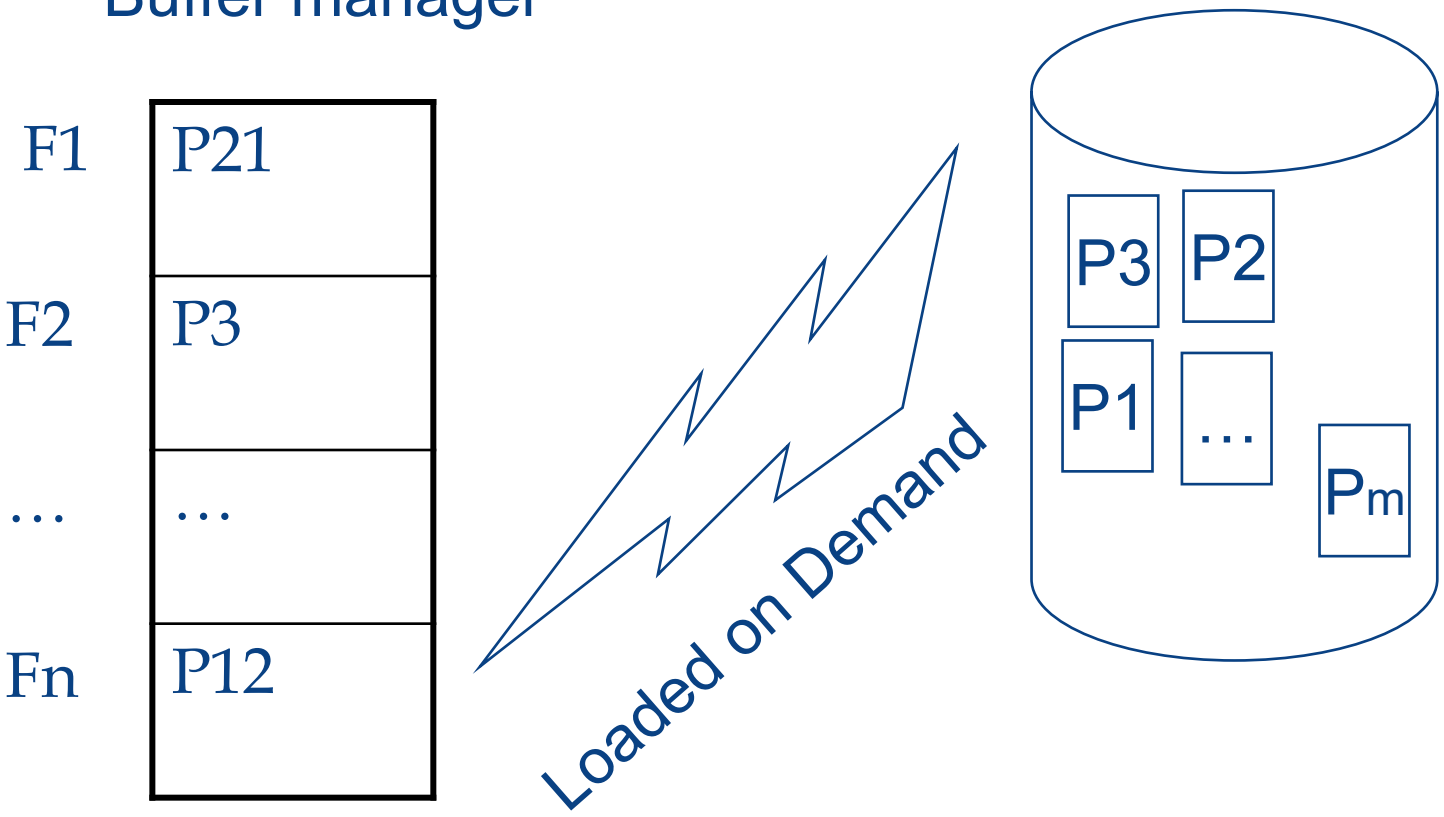
So at first, we need to know what happens during the usual execution of a system.

Memory hierarchy – week 1's lecture

What makes some data non-durable?



Buffer manager



Some activities !

PollEv.com/farhanachoud585

Combining this knowledge with other topics we learnt before

Let's assume that a bitmap index is in memory
while the hard disk became unavailable

If the system is designed to still run* while the disk is being recovered,
Can we still get answers for queries "How many people are in income level L1"?

Record Num	Name	State	Income_level
0	John	VIC	L1
1	Diana	NSW	L2
2	Xiaolu	WA	L1
3	Anil	VIC	L4
4	Peter	NSW	L3

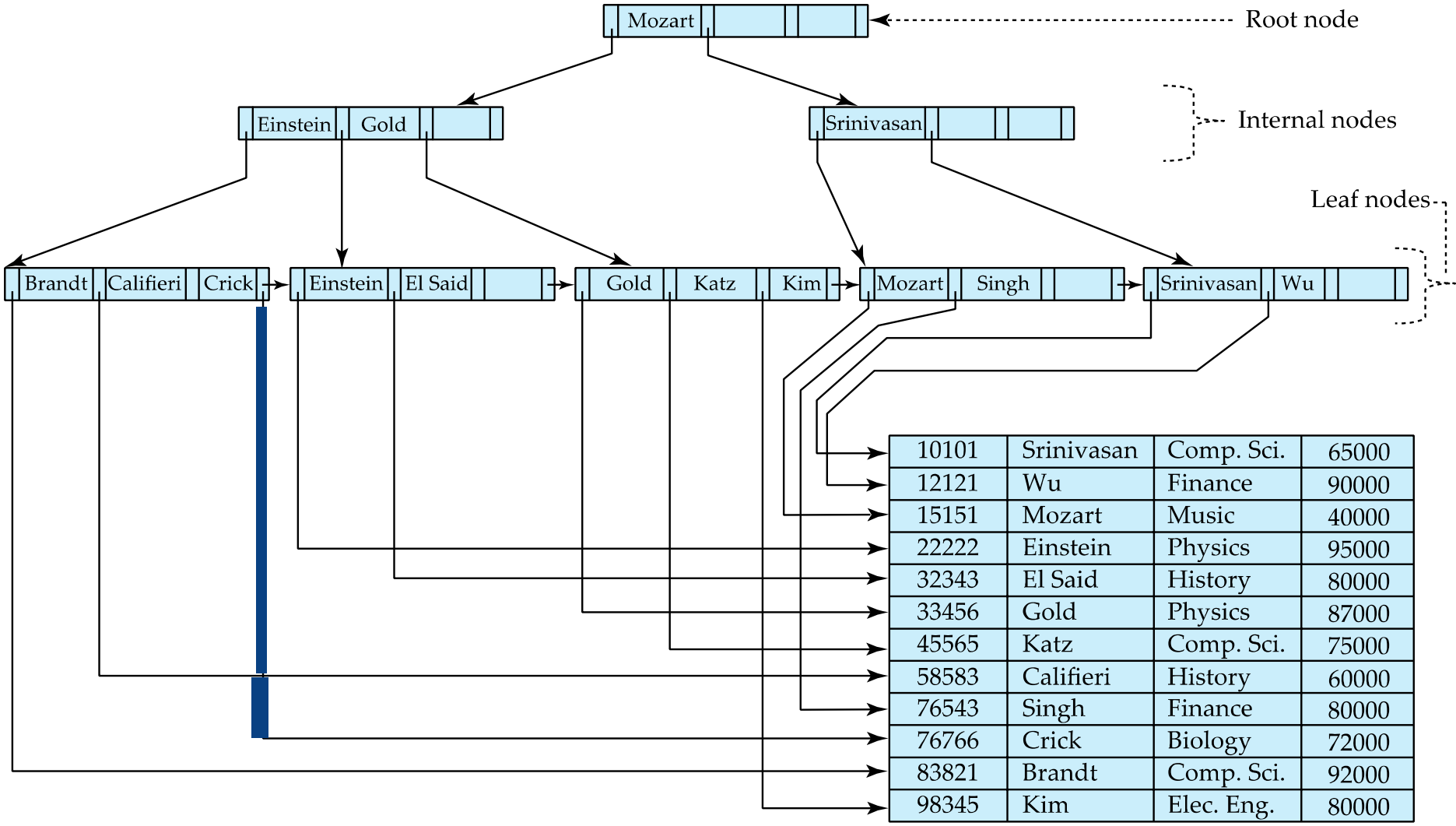
Bitmap for income level	Income_level
L1	1 0 1 0 0
L2	0 1 0 0 0
L3	0 0 0 0 1
L4	0 0 0 1 0
L5	0 0 0 0 0

Combining this knowledge with other topics we learnt before

Let's assume that the nodes of a B+ index is in memory, but leaf nodes point to the records stored on disk. The hard disk became unavailable

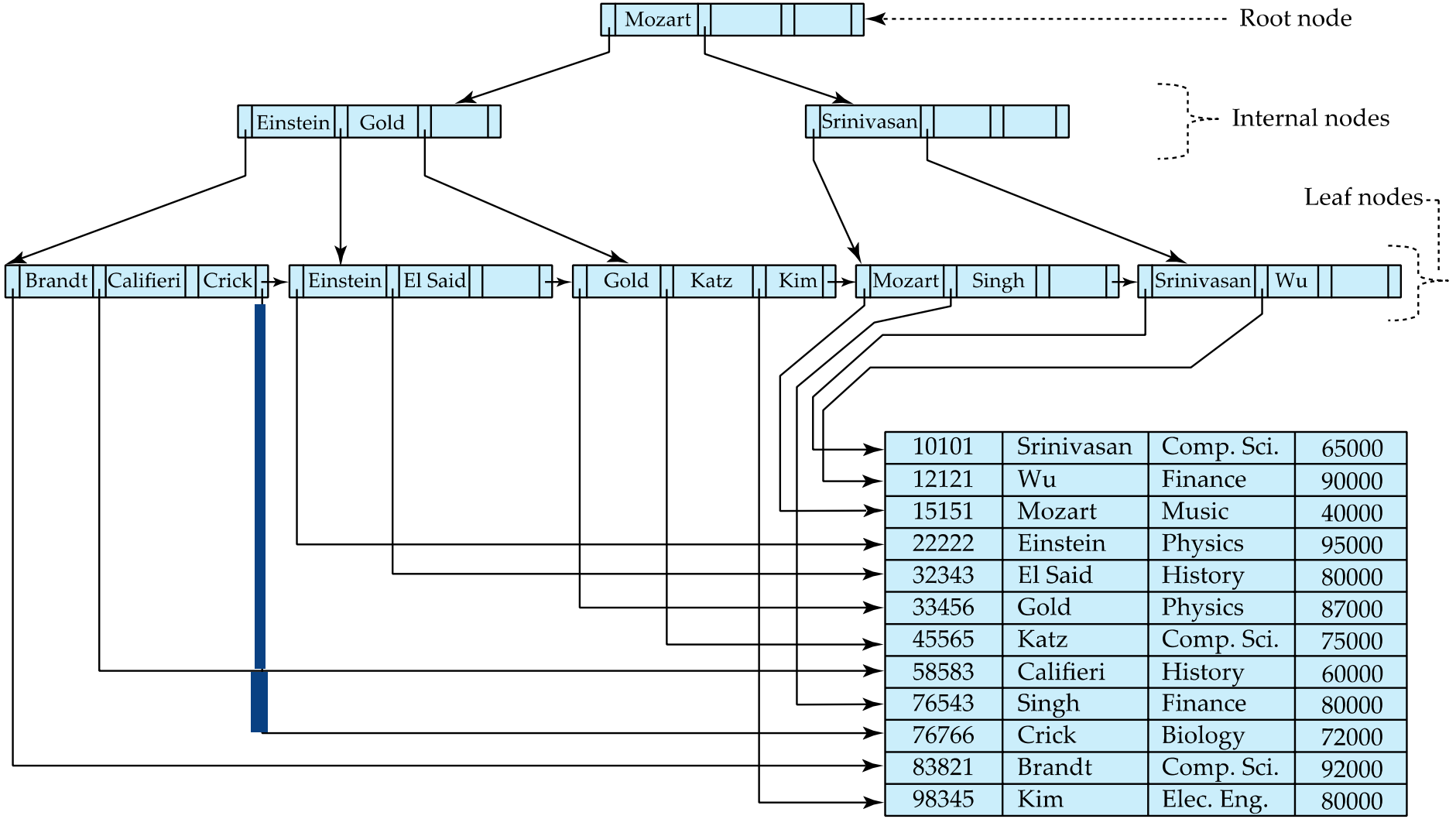
If the system is designed to still run* while the disk is being recovered,
Can we still get answers for queries "What is the salary of Brandt?"

Combining this knowledge with other topics we learnt before



10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	80000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	60000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

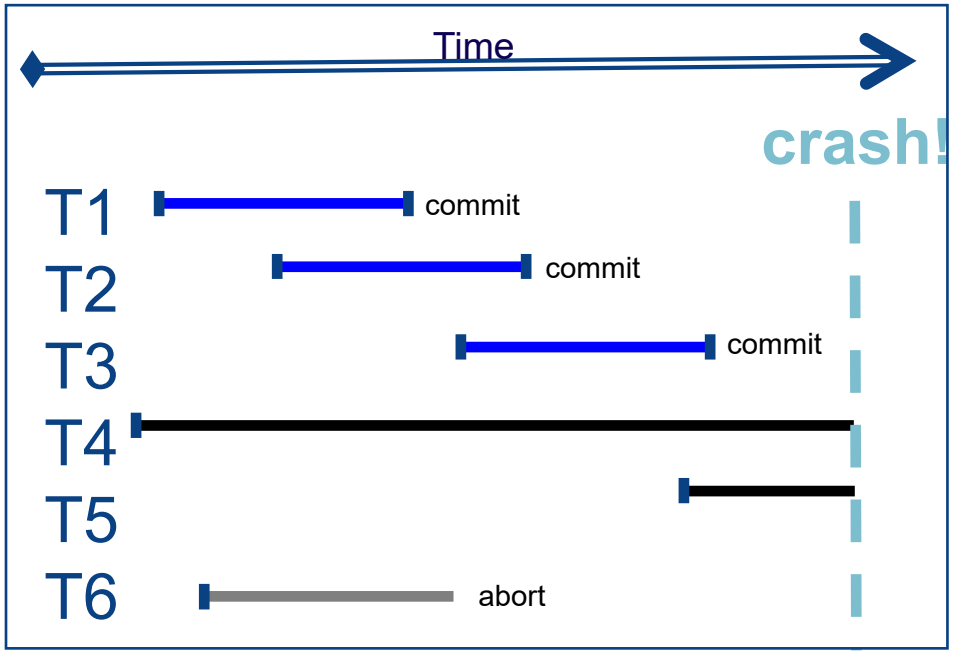
Combining this knowledge with other topics we learnt before



10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	80000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	60000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000

B+ tree and the record 83821 is in memory, while the hard disk became unavailable

What we want if a crash happens



ARIES for crash recovery – most DBMSs use this algorithm (or its variant)

What additional information we need to recover from a crash

Dirty Page table (WAL in place)

Page #	Oldest LSN (least Recent LSN)

X-table

Xid	Status	Last LSN

Log

Prev LSN	Tid	Type	Page	Length	Offset	Old Value	New Value

How exactly are they used to recover?
- using ARIES algorithm

We have learnt crash recovery

What needs to be recovered if a crash happens?

- Has it been made durable - good!
- If not durable – what additional information are needed to recover them?

Data pages in the buffer

Crash manager maintains both durability and atomicity

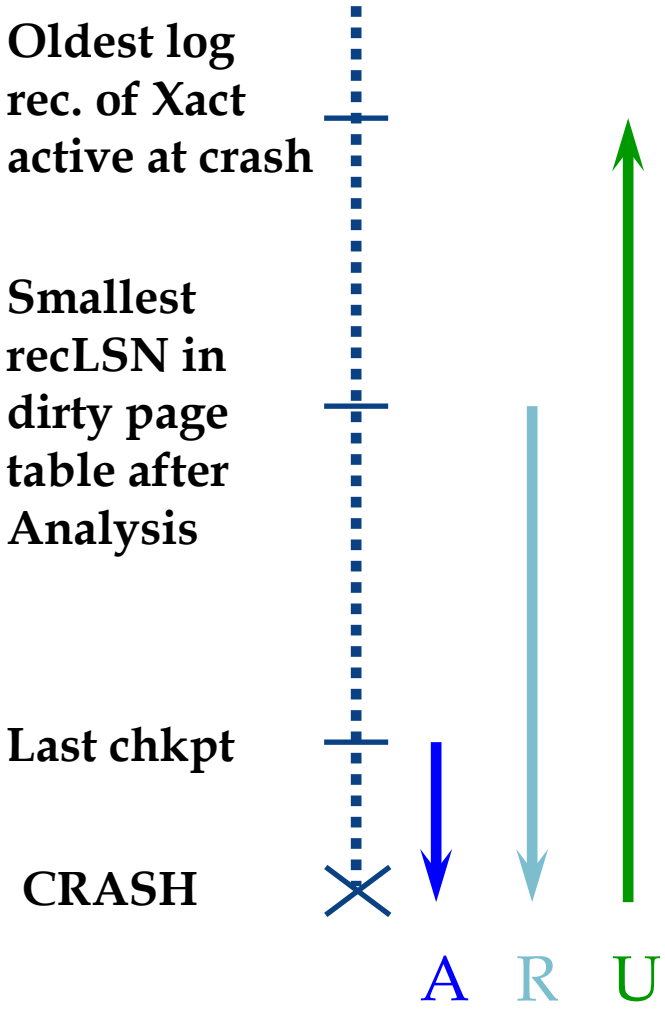
The changes by committed transactions – make them durable
The changes by aborted/running transactions - undo

Checkpointing

Periodically, the DBMS creates a checkpoint with current *Xact table* and *dirty page table*

Store logs and checkpoint records in a safe place

Crash Recovery: Big Picture



- Start from a checkpoint (found via master record).
- Three phases. Need to:
  - Figure out which Xacts committed since checkpoint, which failed (Analysis).
  - REDO ***all*** actions.
    - (repeat history)
  - UNDO effects of failed Xacts.

Some activities !

PollEv.com/farhanachoud585

Tutorial exercises on ARIES

Crash recovery in practice

Choose the right recovery model for your application

Types of recovery models in MS SQL server:

a. Simple: No logs, but has backups. Recovery is done from the last backup
b. Full: Uses logs plus backups, regular checkpoints
c. Bulk logged: Logs are not maintained for each individual writes, but for multiple writes together*

*For some operations

Crash recovery in practice

Choose the right recovery model for your application

1. Find the current recovery model

```
SELECT name, recovery_model_desc
FROM sys.databases
WHERE name = 'model' ;
```

2. Change the model if needed

3. **Checkpoint interval:** Specify target recovery interval, for example, if a crash happens, recover within 1 min. The system will then determine how often it needs to create checkpoints based on that value.

Some more on previously learnt topics

Disk writes for consistency:

Either entire block is written **correctly** on disk or the contents of the block is unchanged. To achieve disk write consistency we can do –

- ***Duplex*** write
- **Logged** write

Some more on previously learnt topics

Disk writes for consistency:

Either entire block is written **correctly** on disk or the contents of the block is unchanged. To achieve disk write consistency we can do –

Duplex write:

- Each block of data is written in two places *sequentially*
- If one of the writes fail, system can issue another write
- Each block is associated with a version number. The block with the latest version number contains the most recent data.
- While reading - we can determine error of a disk block by its ***CRC***.
- It always guarantees at least one block has consistent data.

Logged write- similar to duplex write, except one of the writes goes to a log. This method is very efficient if the changes to a block are small.

The first one of the writes goes to a log. The second overwrites the old regular data block. In short, all modifications need to be logged before they are applied.

So if a failure occurs, system knows where we are left to take proper action, i.e., even during a single block write.

Some more on previously learnt topics – nested transactions

Commit rule

- A subtransaction can either commit or abort, however, **commit cannot take place unless the parent itself commits.**

- Subtransactions have A, C, and I properties but not D property unless all its ancestors commit.

- Commit of a sub transaction makes its results available only to its parents.

Roll back Rules

If a subtransaction rolls back, all its children are forced to roll back.

Visibility Rules

Changes made by a subtransaction are visible to the parent only when the subtransaction commits. All objects of parent are visible to its children. Implication of this is that the **parent should not modify objects while children are accessing them.** This is not a problem as parent does not run in parallel with its children.

What if a crash happens during a nested transaction?

We have also seen backups and recovery in practice

Strategy plan based on:

- Goals and requirement of your organization/task
- The nature of your data and usage pattern
- Constraint on resources

Design backup strategy:

- Full disk backup vs partial - Are changes likely to occur in only a small part of the database or in a large part of the database?
- How frequently data changes
  - If frequent: use differential backup that captures only the changes since the last full database backup
- Space requirement of the backups – depends on the resource
- Multiple past instances of backup – useful if point-in-time recovery is needed

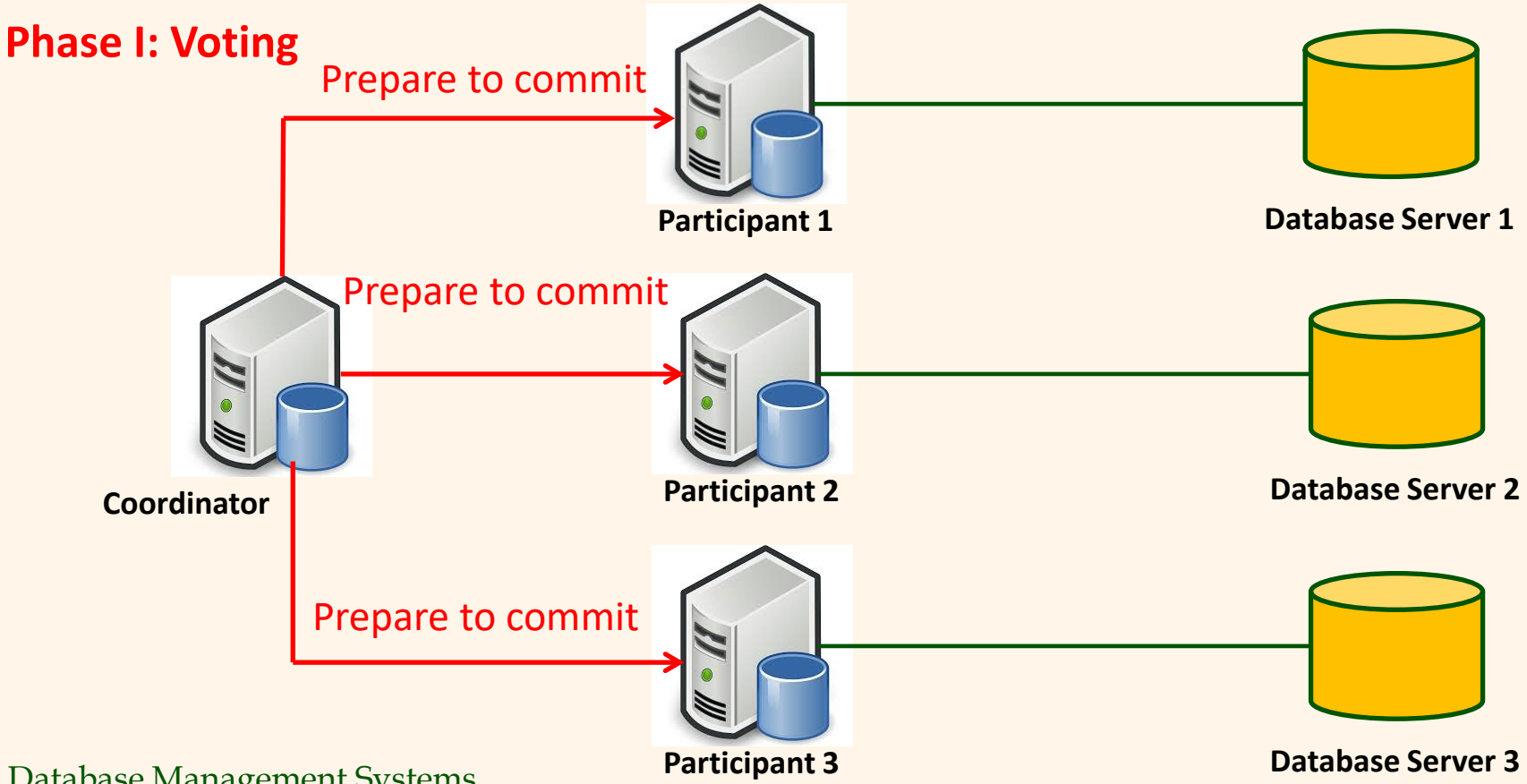
Resource: https://learn.microsoft.com/en-us/sql/relational-databases/backup-restore/back-up-and-restore-of-sql-server-databases?view=sql-server-ver16

DBMSs in the era of Networking

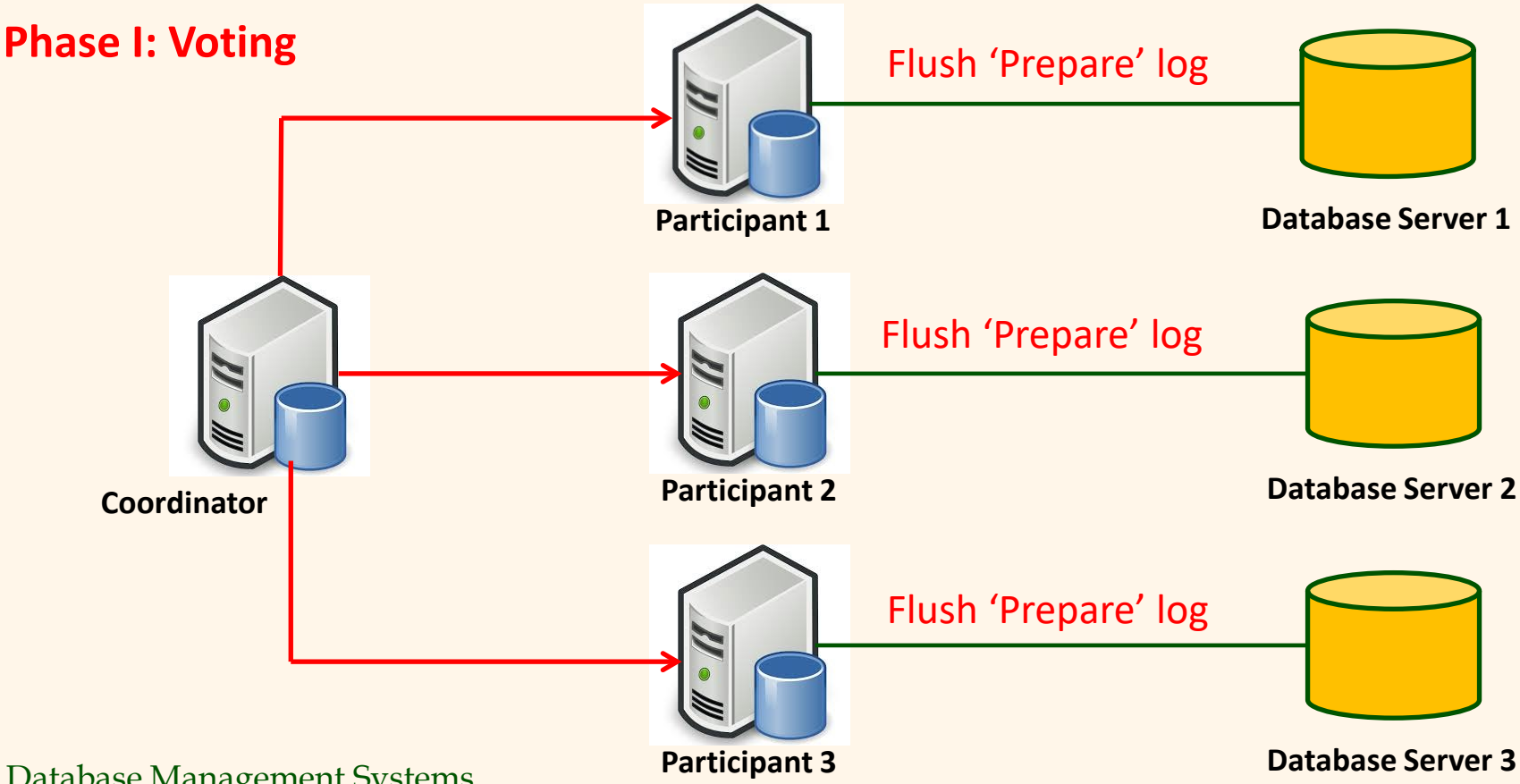
- Atomicity, concurrency, replication rule in distributed transaction processing

Atomicity in Distributed Transaction Processing

- The two-phase commit protocol (2PC) can help achieve atomicity in distributed transaction processing

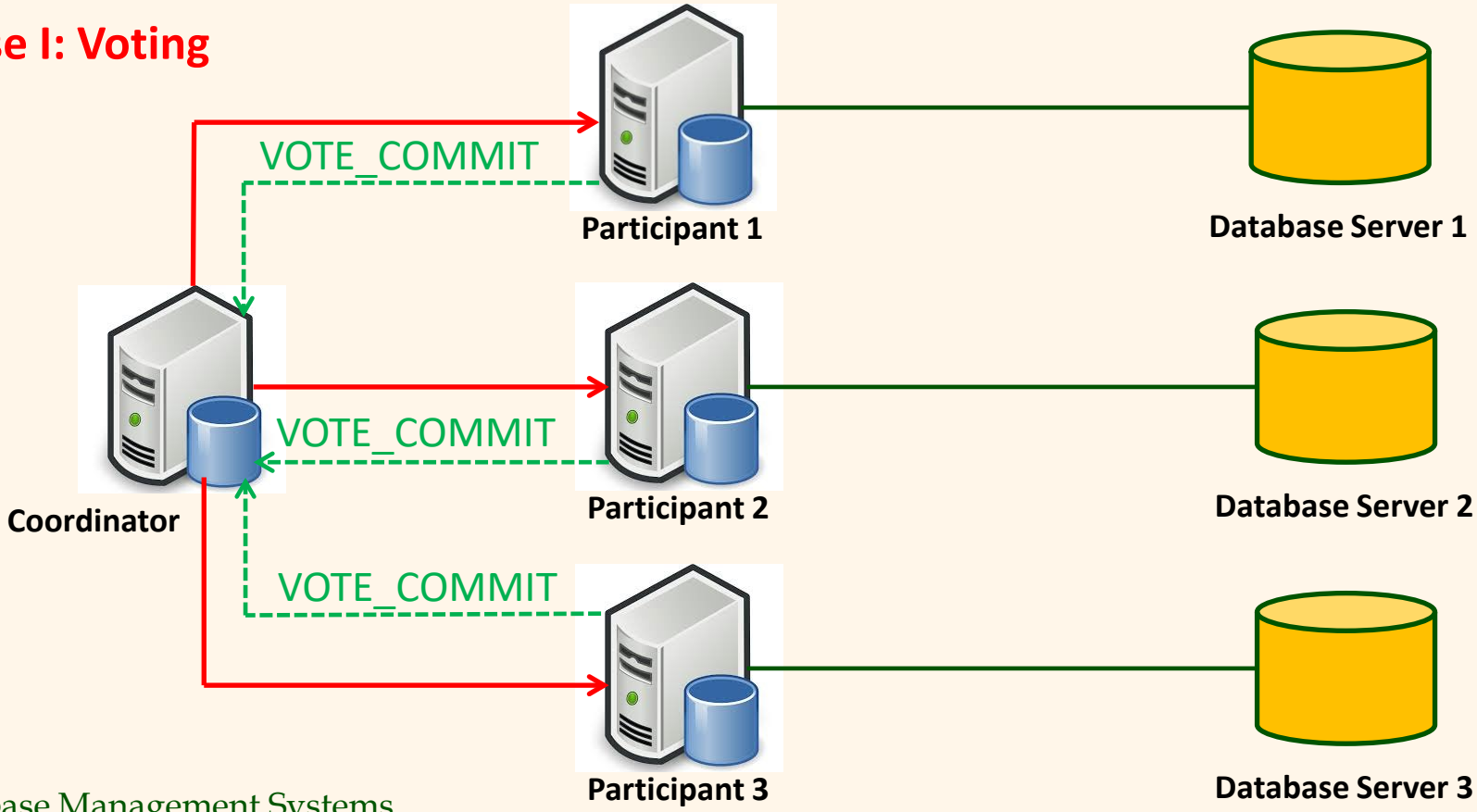


The Two-Phase Commit Protocol

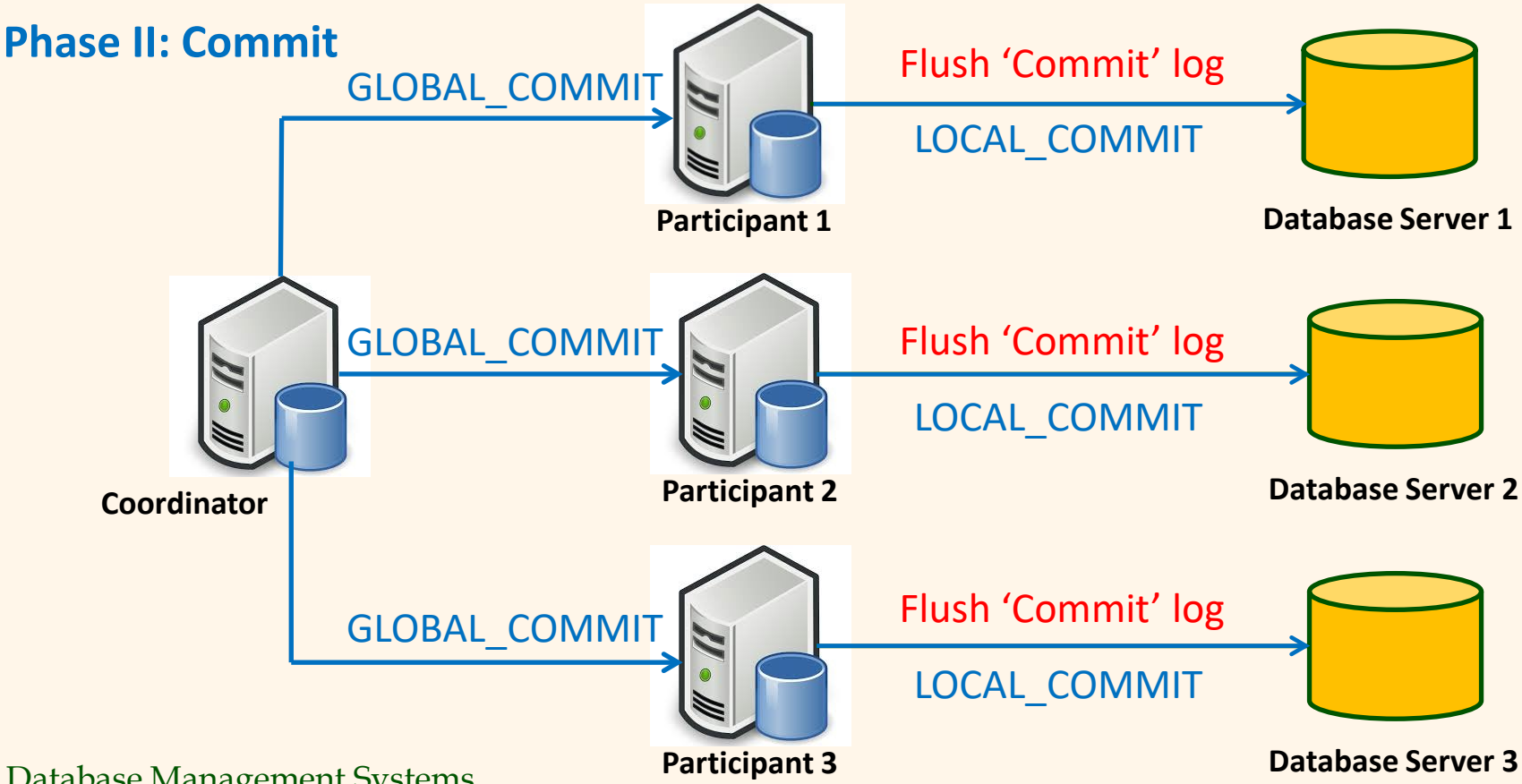


The Two-Phase Commit Protocol

Phase I: Voting



The Two-Phase Commit Protocol



The CAP Theorem

- The limitations of distributed databases can be described in the so called the CAP theorem
  - **C**onsistency: every node always sees the same data at any given instance (i.e., strict consistency)
  - **A**vailability: the system continues to operate, even if nodes crash, or some hardware or software parts are down due to upgrades
  - **P**artition Tolerance: the system continues to operate in the presence of network partitions

CAP theorem: any distributed database with shared data, can have *at most two* of the three desirable properties, C, A or P

CAP -> PACELC

- A more complete description of the space of potential tradeoffs for distributed system:
  - If there is a **partition (P)**, how does the system trade off **availability and consistency (A and C)**; **else (E)**, when the system is running normally in the absence of partitions, how does the system trade off **latency (L) and consistency (C)**?

Abadi, Daniel J. "Consistency tradeoffs in modern distributed database system design." Computer-IEEE Computer Magazine 45.2 (2012): 37.

Different design choices based on CAP theorem: Examples

- Different data may require different consistency and availability
- Example:
  - Shopping cart: high availability, responsive, can sometimes suffer anomalies/inconsistencies
  - Product information need to be available, slight variation in inventory is sufferable
  - Checkout, billing, shipping records must be consistent

Final exam discussion

Core Concepts of Database management system