Chapter Five

Big Data Storage Concepts

Topics

- Clusters
- File Systems and Distributed File Systems
- NoSQL
- Sharding
- Replication
- Sharding and Replication
- CAP Theorem
- ACID
- BASE

Clusters

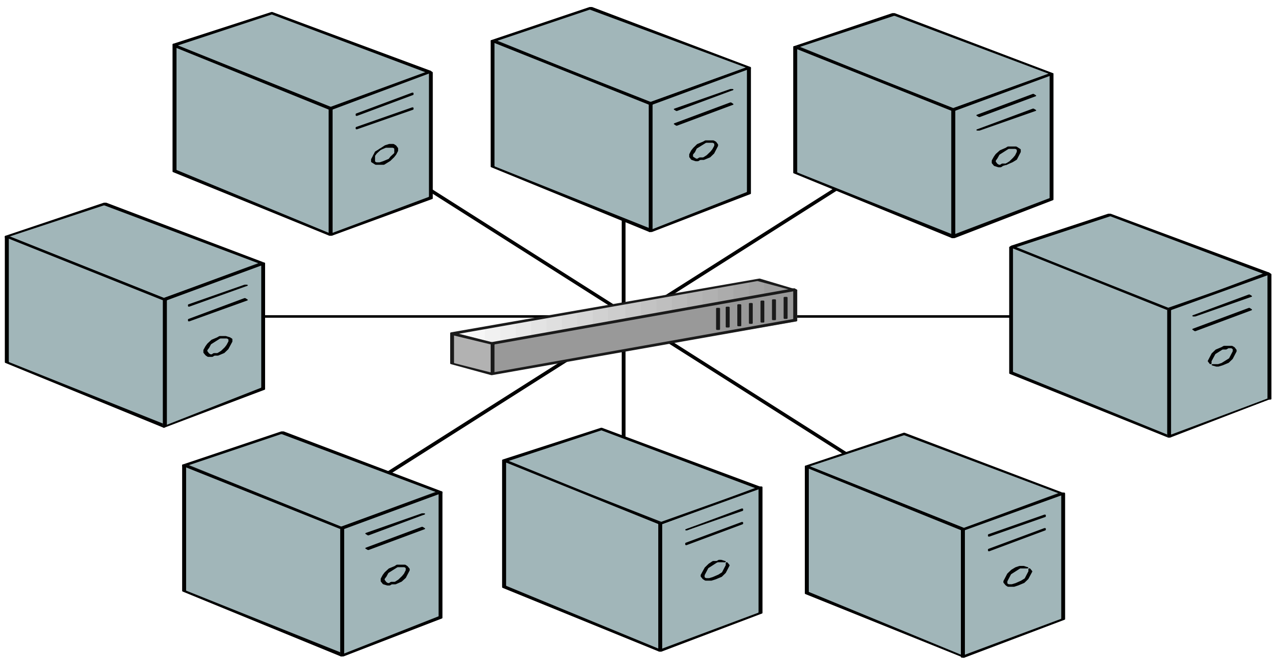


Figure 5.1 The symbol used to represent a cluster.

File System

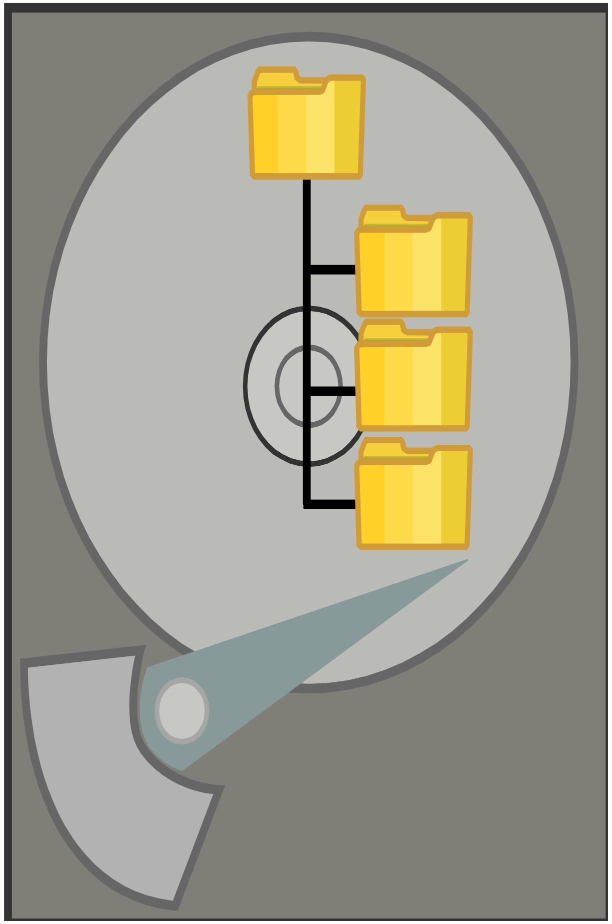


Figure 5.2 The symbol used to represent a file system.

Distributed file systems.

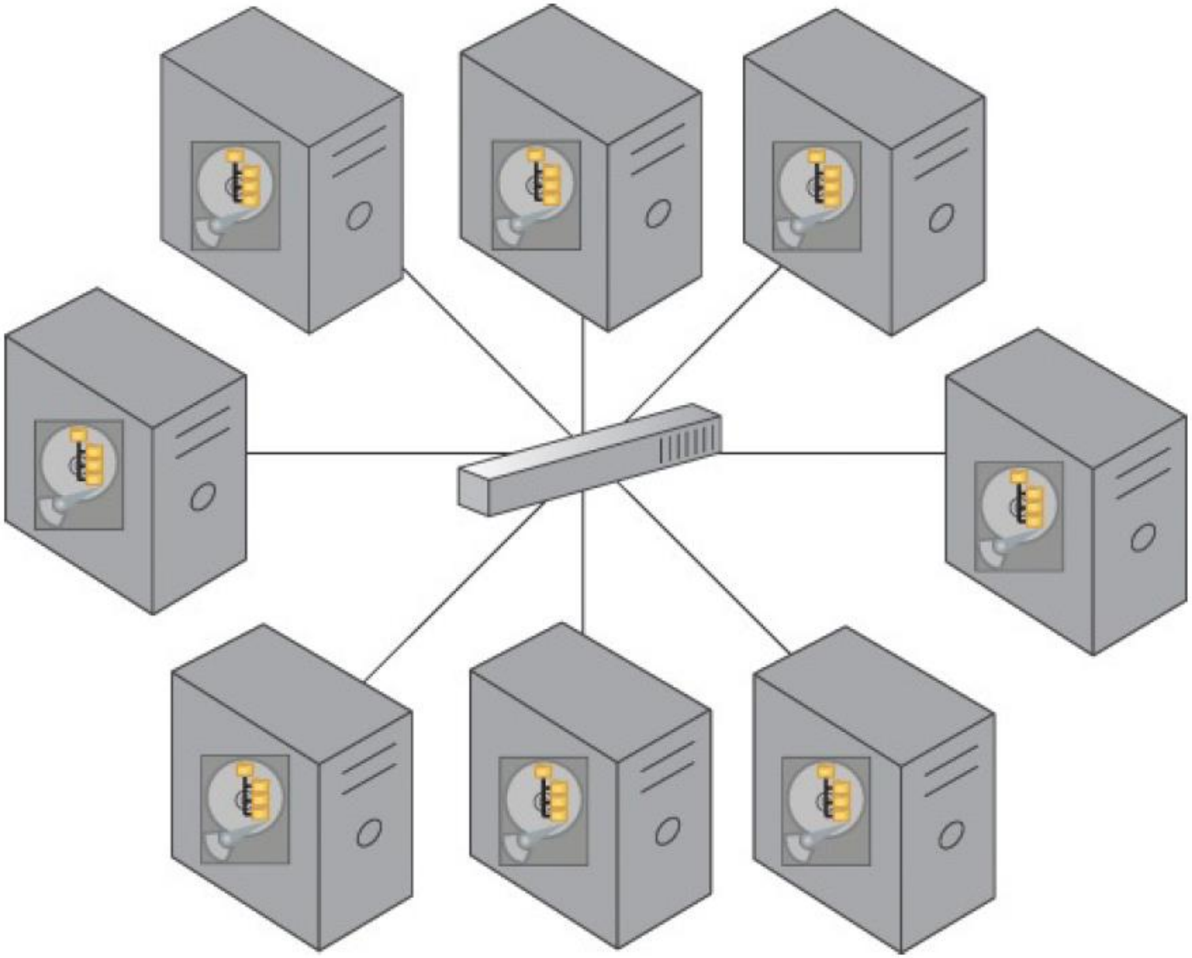


Figure 5.3 The symbol used to represent distributed file systems.

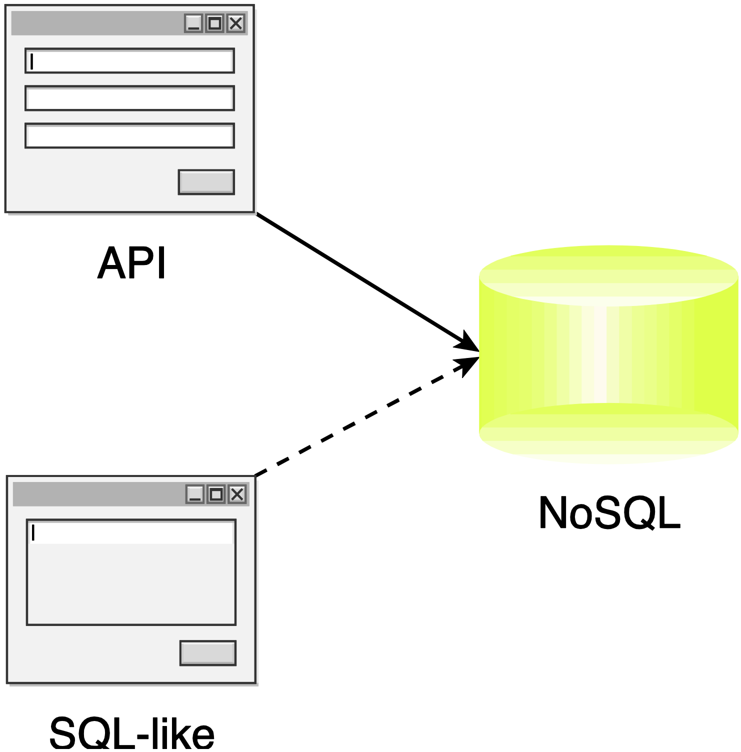


Figure 5.4 A NoSQL database can provide an API- or SQL-like query interface.

Sharding

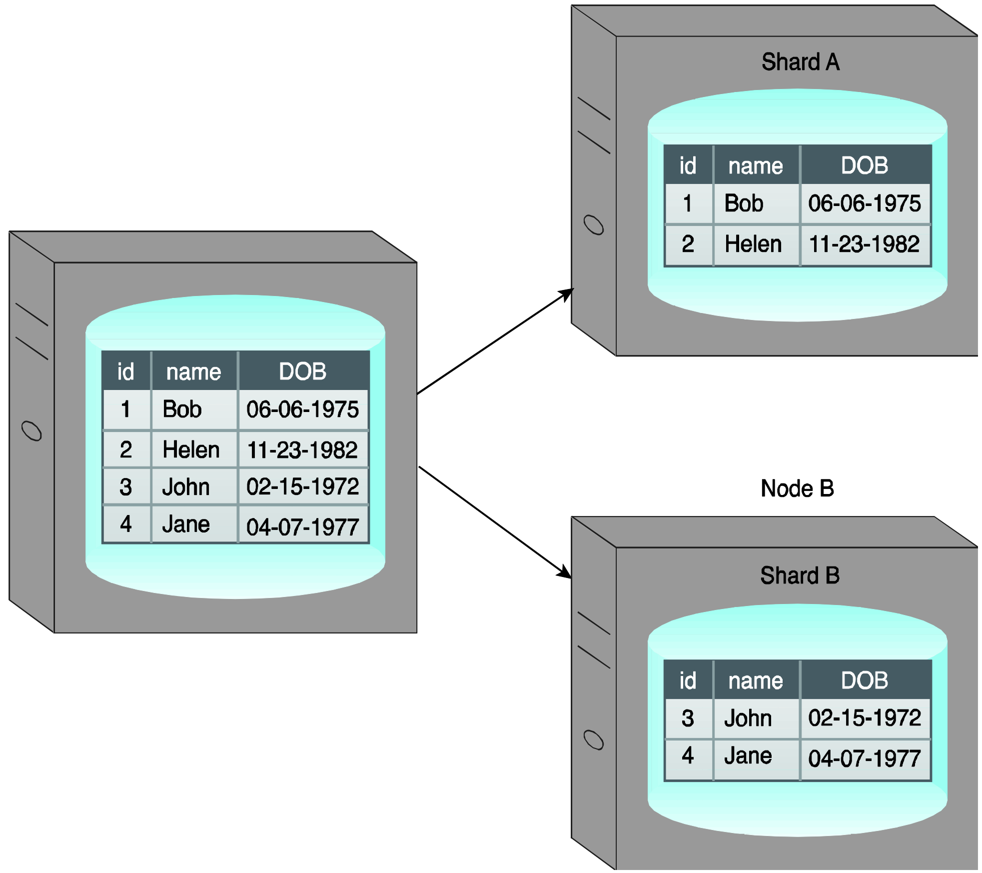


Figure 5.5 An example of sharding where a dataset is spread across Node A and Node B, resulting in Shard A and Shard B, respectively.

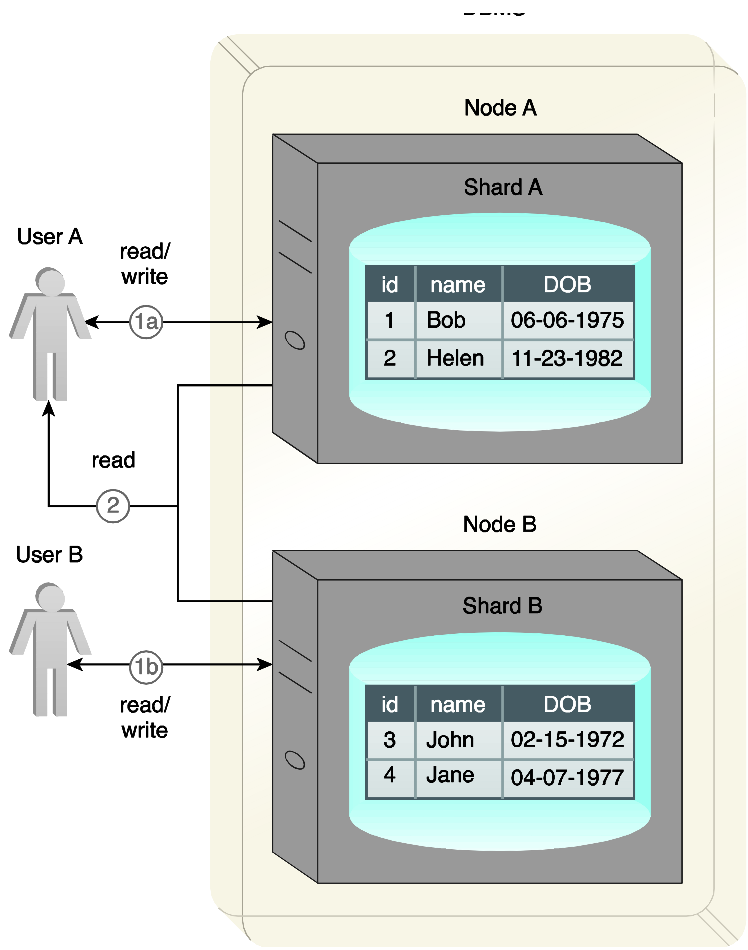


Figure 5.6 A sharding example where data is fetched from both Node A and Node B.

Exercise1 :

1. What are the benefits of Sharding?

2. What are the drawbacks of Sharding?

Replication

Replication stores multiple copies of a dataset, known as *replicas*, on multiple nodes

Replication methods:

- Master-slave
- Peer-to-pear

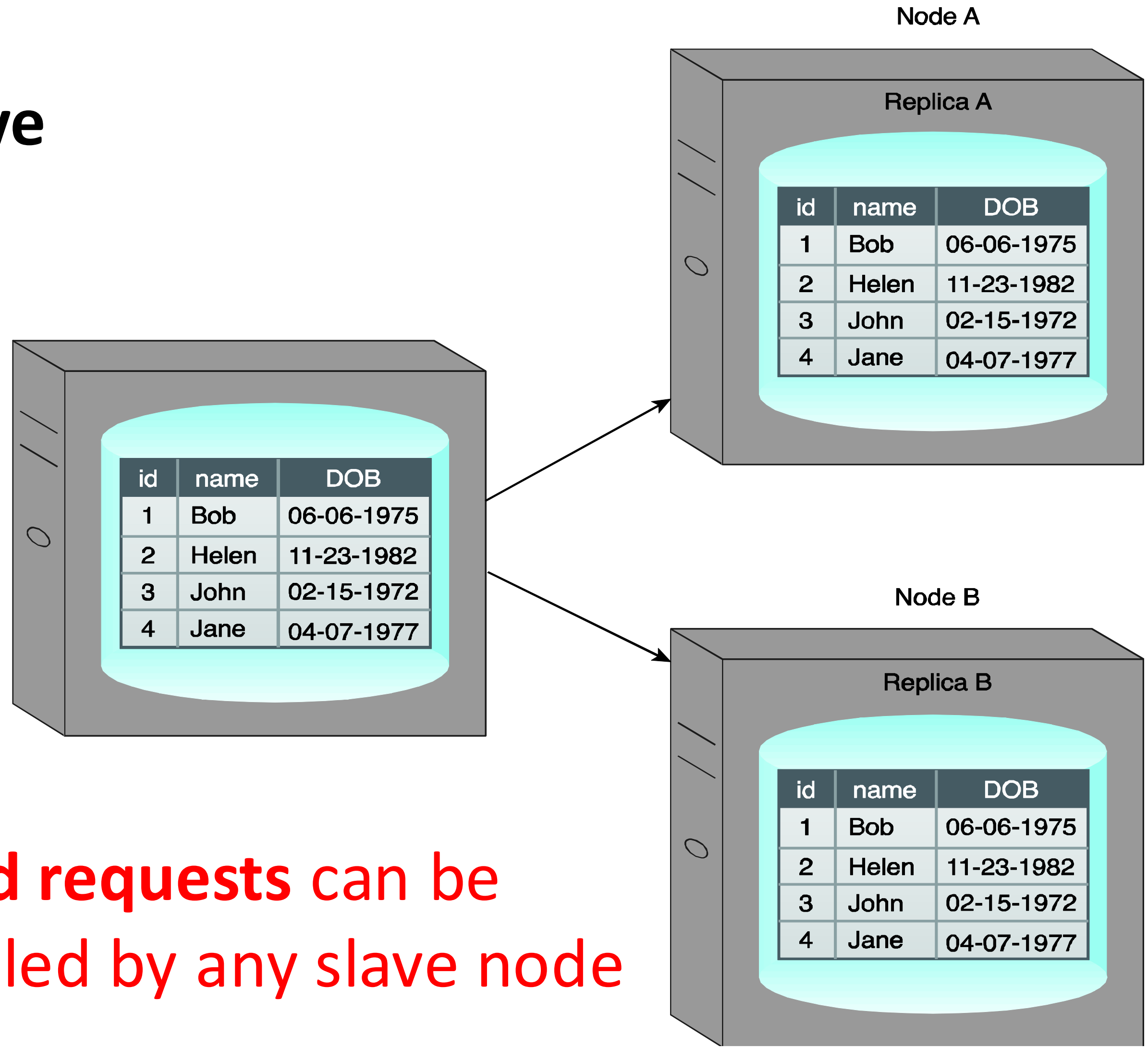
Exercise2:

1. What are the benefits of **Replication**?

2. What are the drawbacks of **Replication**?

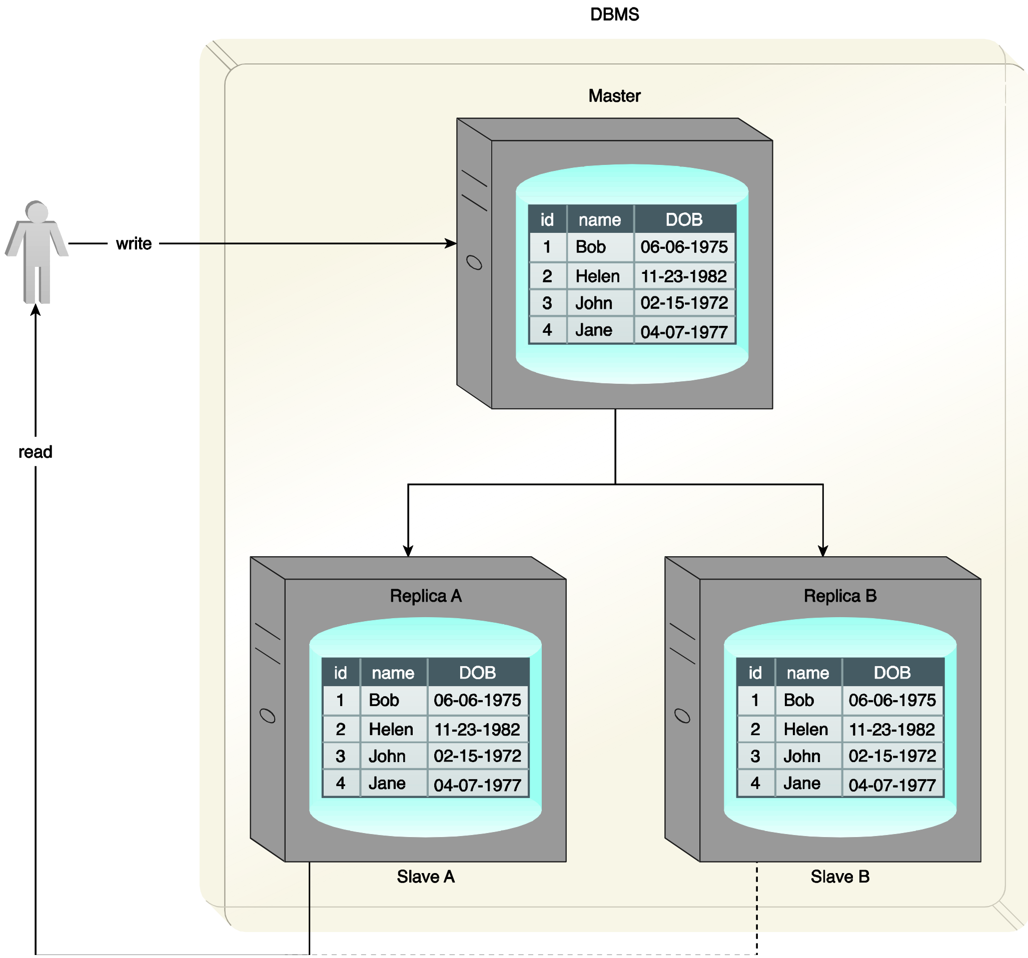
Master-Slave

Write to master node



Read requests can be fulfilled by any slave node

Figure 5.7 An example of replication where a dataset is replicated to Node A and Node B, resulting in Replica A and Replica B.



"Copyright © 2016 Pearson India Education Services Pvt. Ltd".

Figure 5.8 An example of master-slave replication where Master A is the single point of contact for all writes, and data can be read from Slave A and Slave B.

Exercise3:

1. What are the benefits of **Master-Slave Replication**?

2. What are the drawbacks of **Master-Slave Replication**?

Another Issue: read inconsistency

Solution?

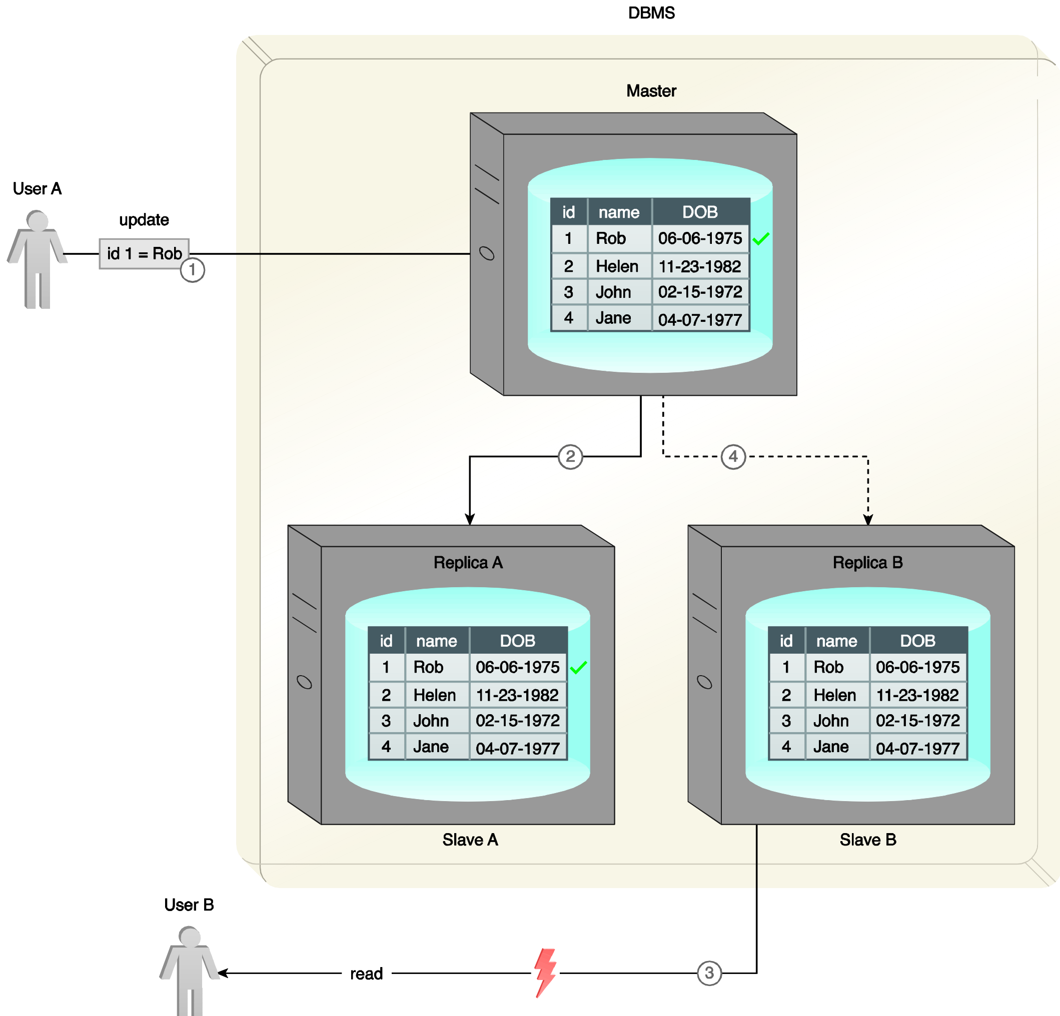


Figure 5.9 An example of master-slave replication where **read inconsistency occurs**

Peer-to-Peer

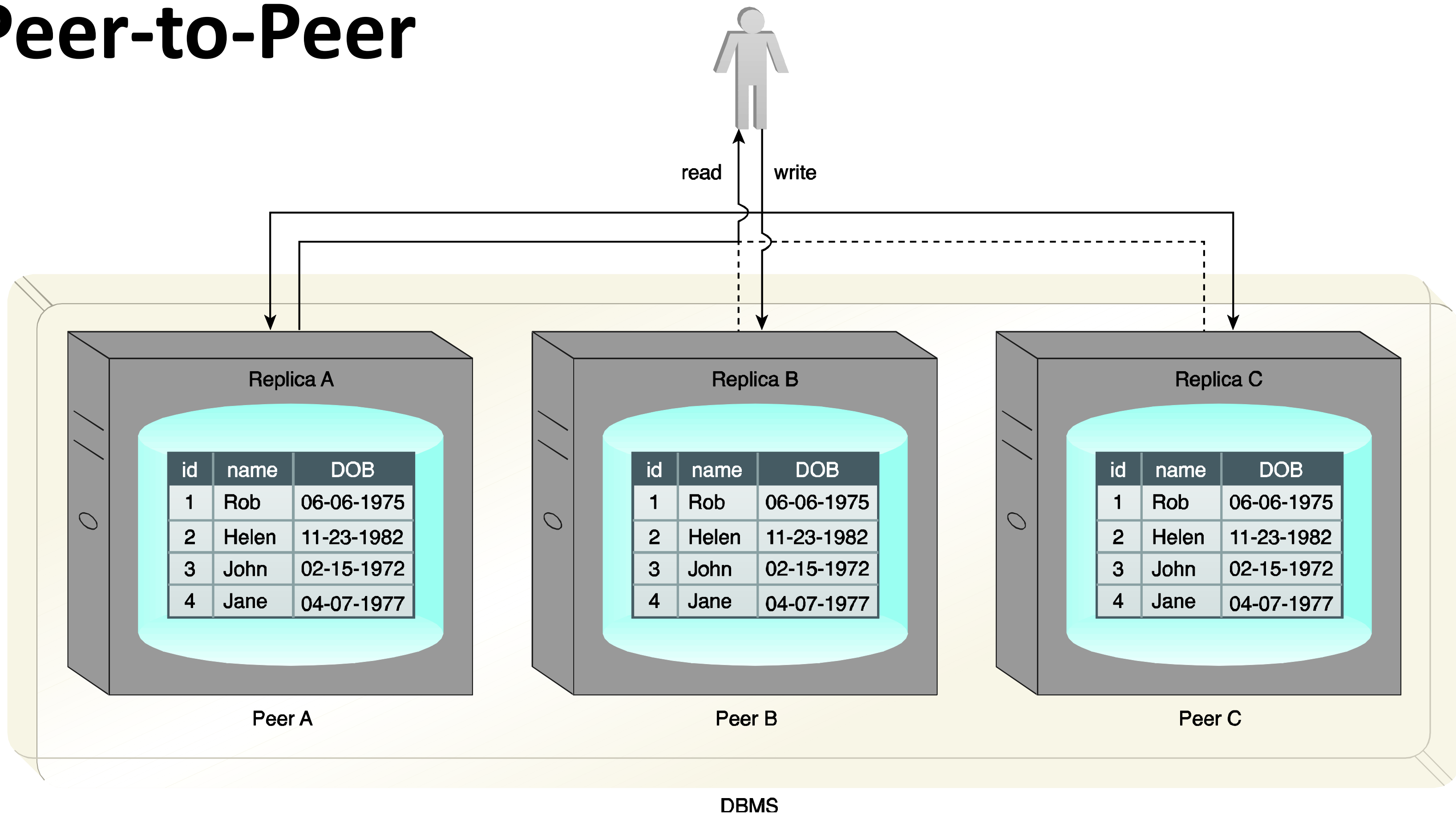


Figure 5.10 Writes are copied to Peers A, B and C simultaneously. Data is read from Peer A, but it can also be read from Peers B or C.

Issue: write inconsistencies that occur as a result of a simultaneous update of the same data

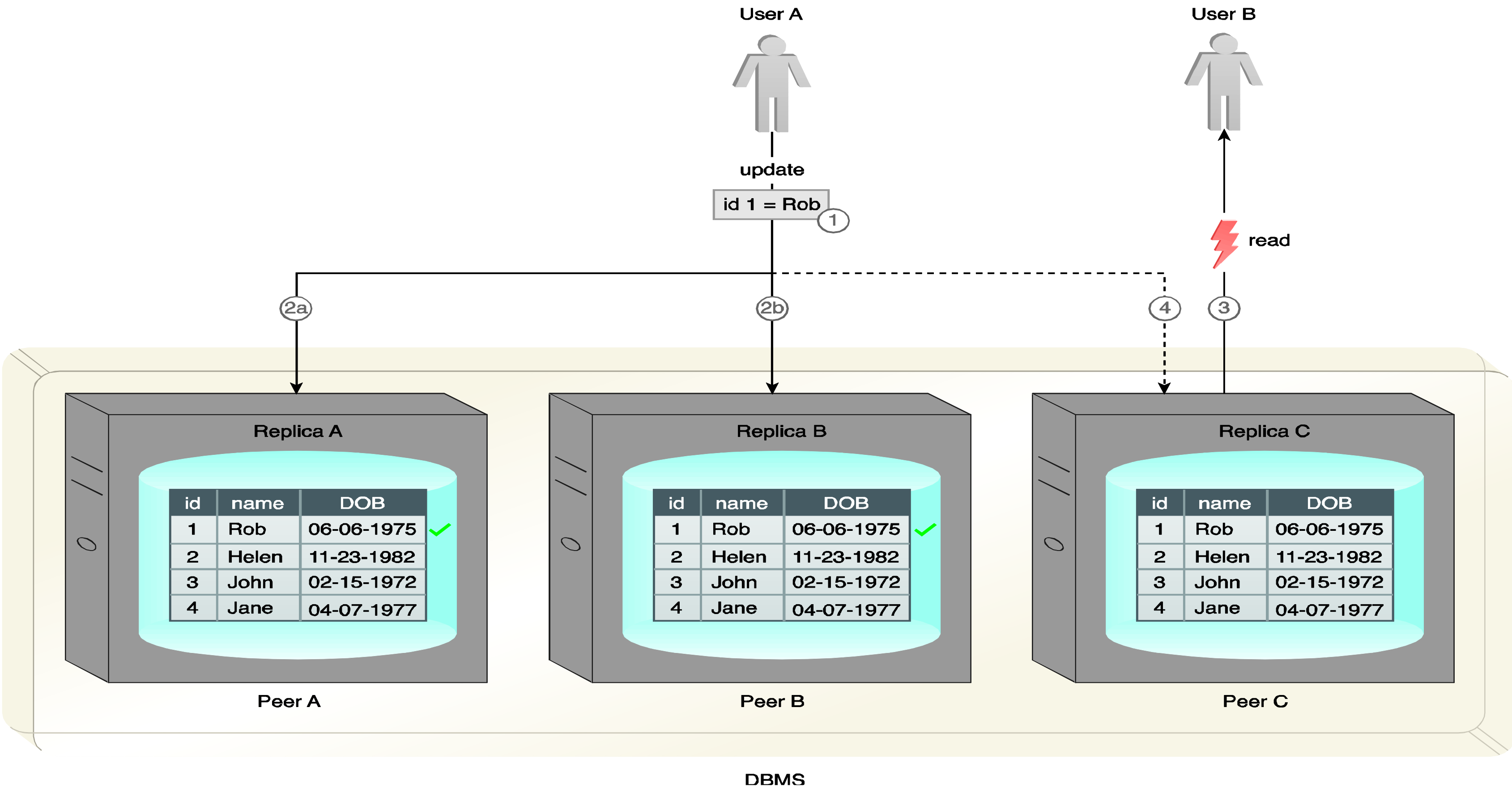


Figure 5.11 An example of peer-to-peer replication where an inconsistent read occurs.

Exercise4:

How to address write inconsistencies in Peer-to-Peer replication? (Hint: Concurrency)

Sharding and Replication

fault tolerance

Space efficiency

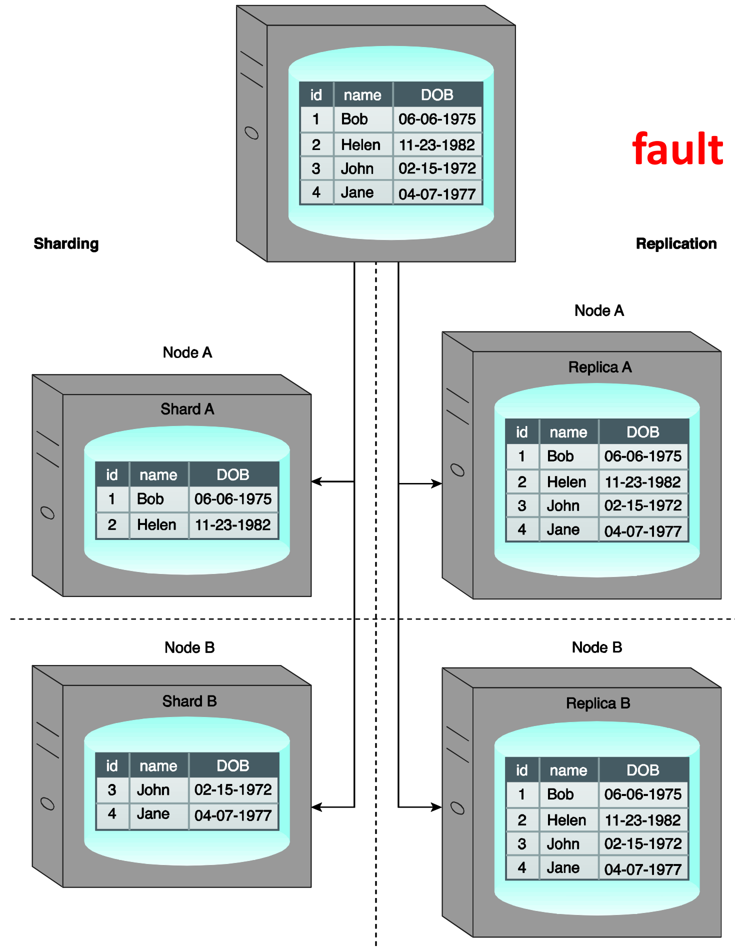


Figure 5.12 A comparison of sharding and replication that shows how a dataset is distributed between two nodes with the different approaches.

Sharding and Master-Slave Replication

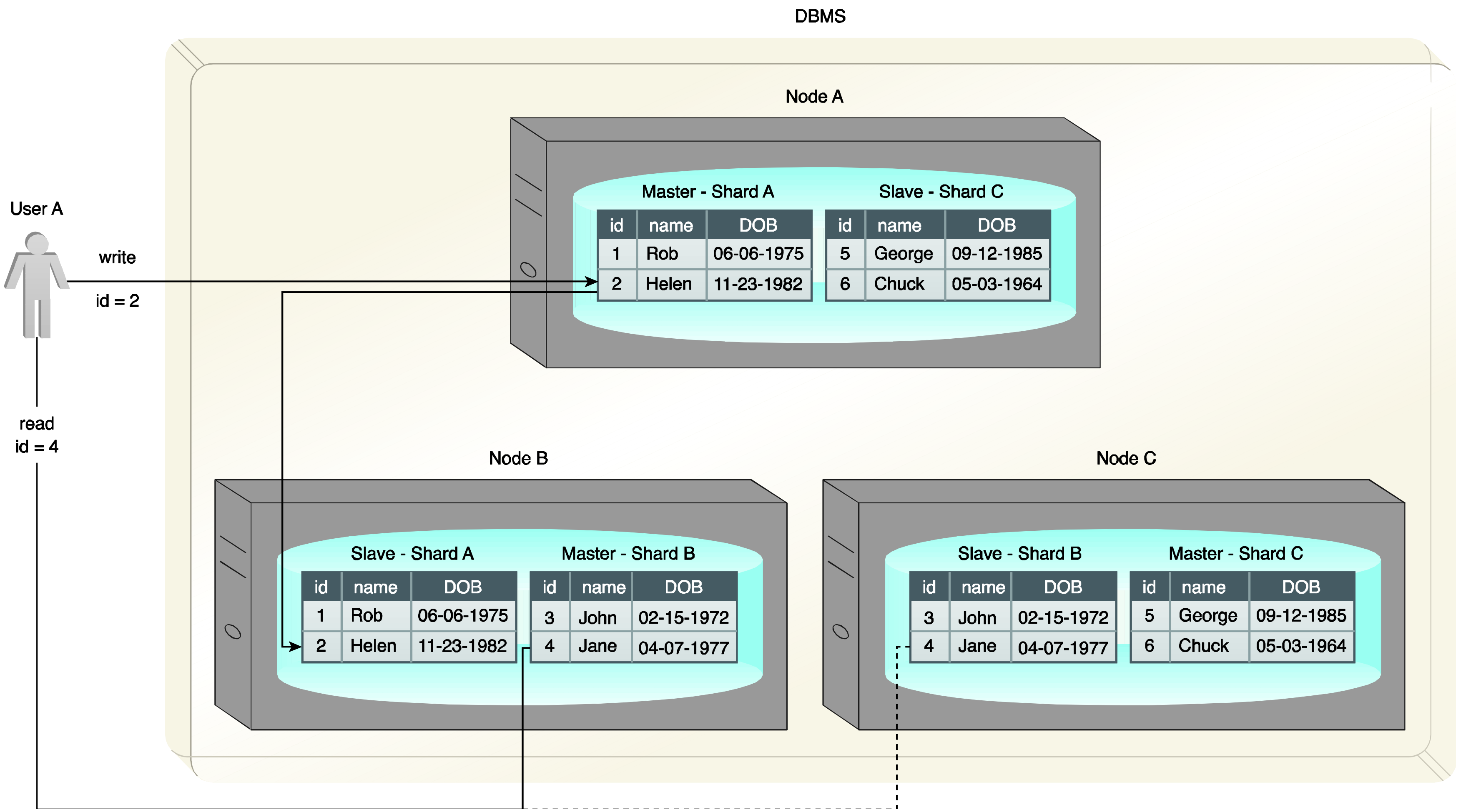


Figure 5.13 An example that shows the combination of sharding and master-slave replication.

Sharding and Peer-to-Peer Replication

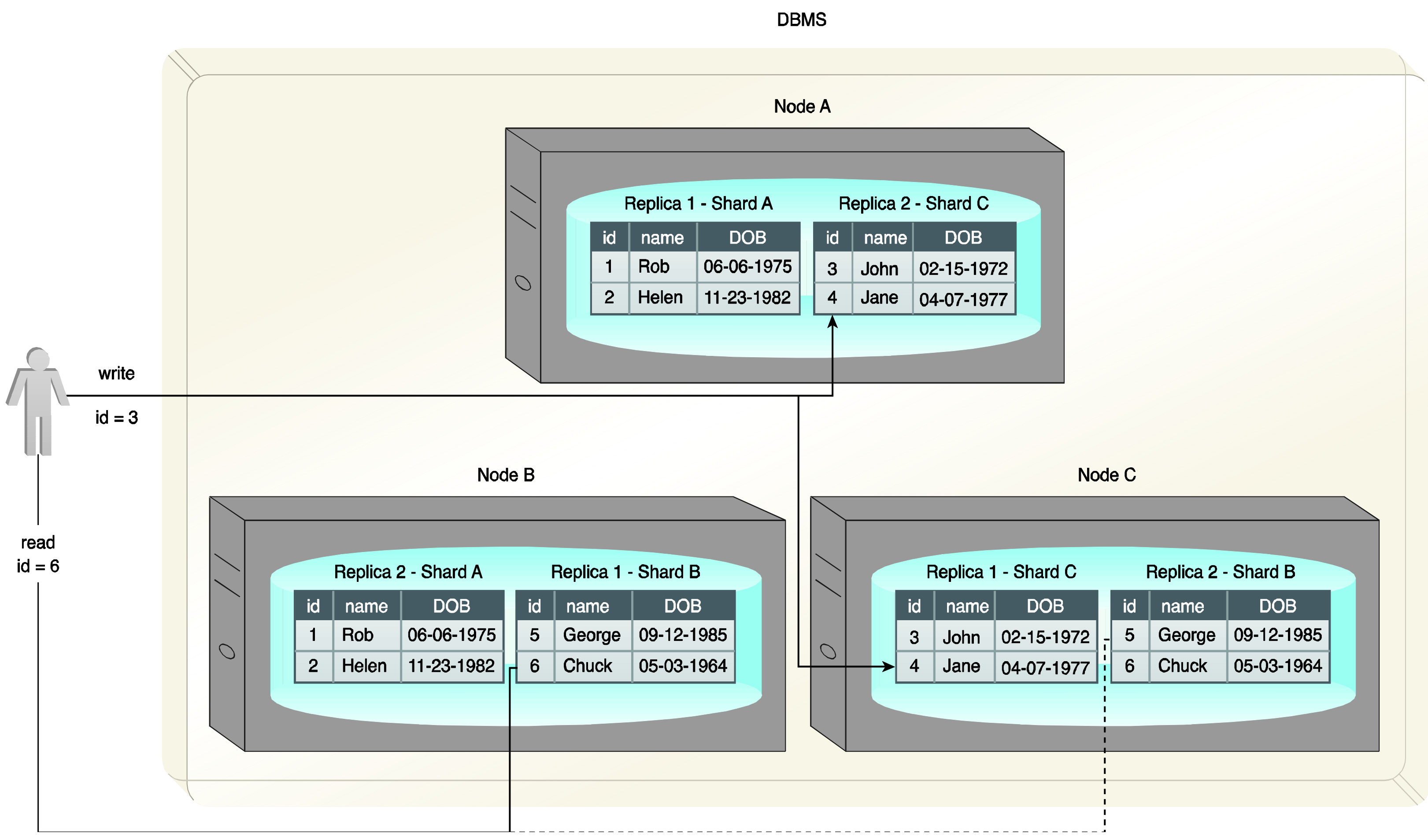


Figure 5.14 An example of the combination of sharding and peer-to-peer replication.

Exercise5:

- What is the difference between *Sharding Master-Slave Replication* and *Sharding Peer-to-Peer Replication*?

CAP

Consistency – A read from any node results in the same data across multiple nodes

Availability – A read/write request will always be acknowledged in the form of a success or a failure

Partition tolerance – The database system can tolerate *communication* outages that split the cluster into multiple silos and can still service read/write requests

Consistency

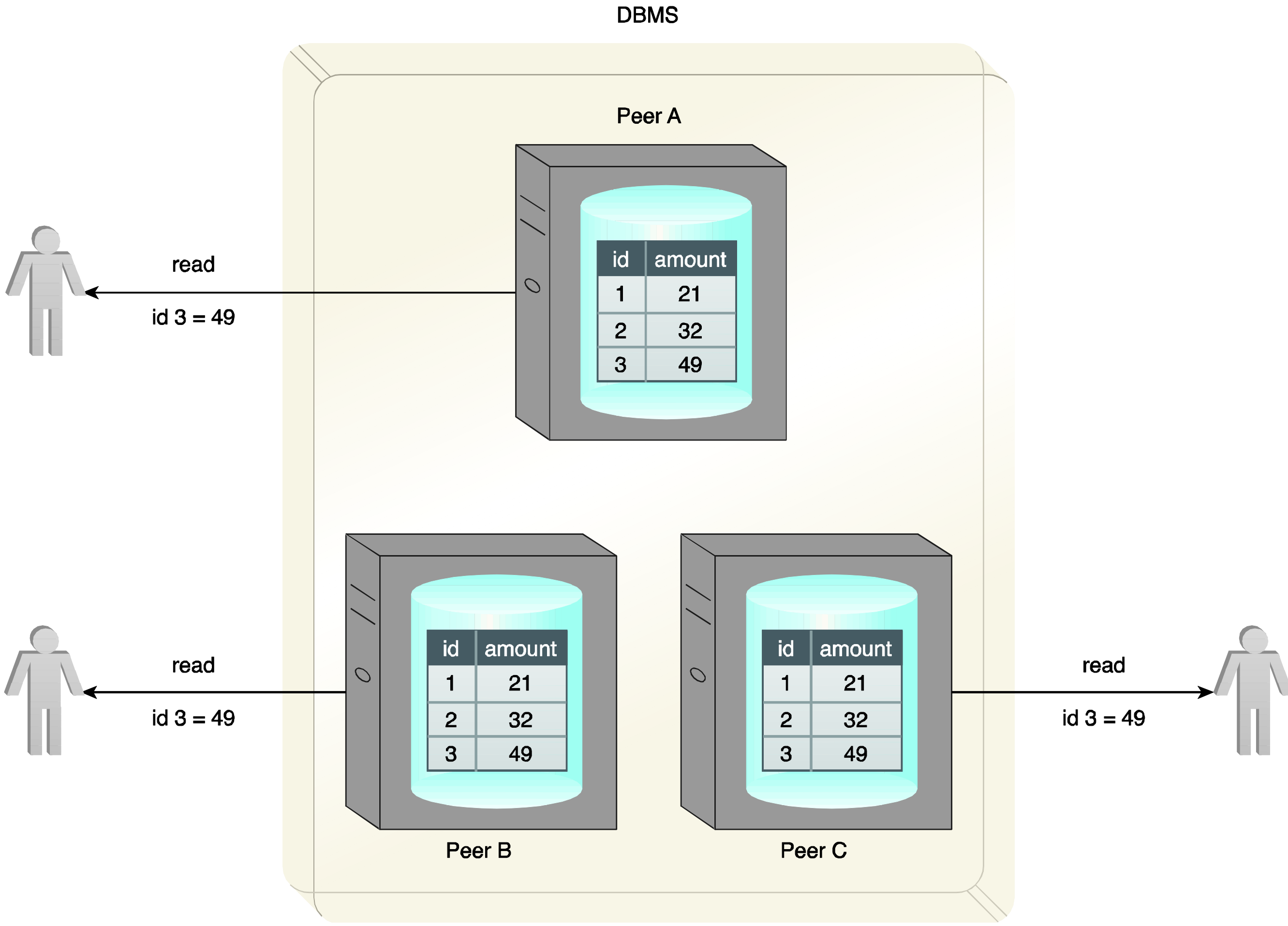


Figure 5.15 Consistency: all three users get the same value for the amount column even though three different nodes are serving the record.

Availability and **Partition tolerance**

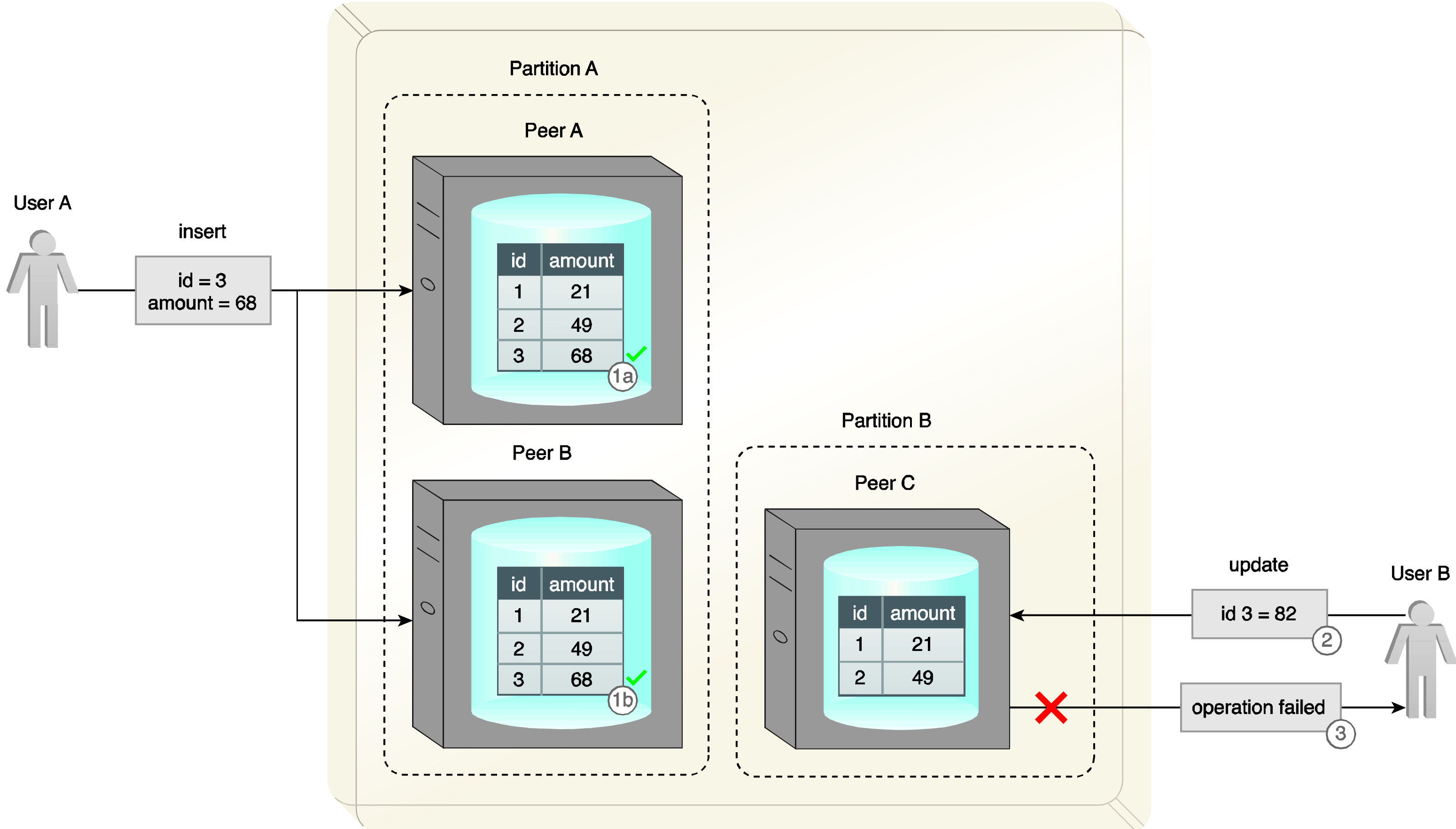


Figure 5.16 Availability and partition tolerance: in the event of a communication failure, requests from both users are still serviced (1, 2). However, with User B, the update fails as the record with id = 3 has not been copied over to Peer C. The user is duly notified (3) that the update has failed.

CAP theorem - *"A distributed database system, running on a cluster, can only provide two of the following three properties"*

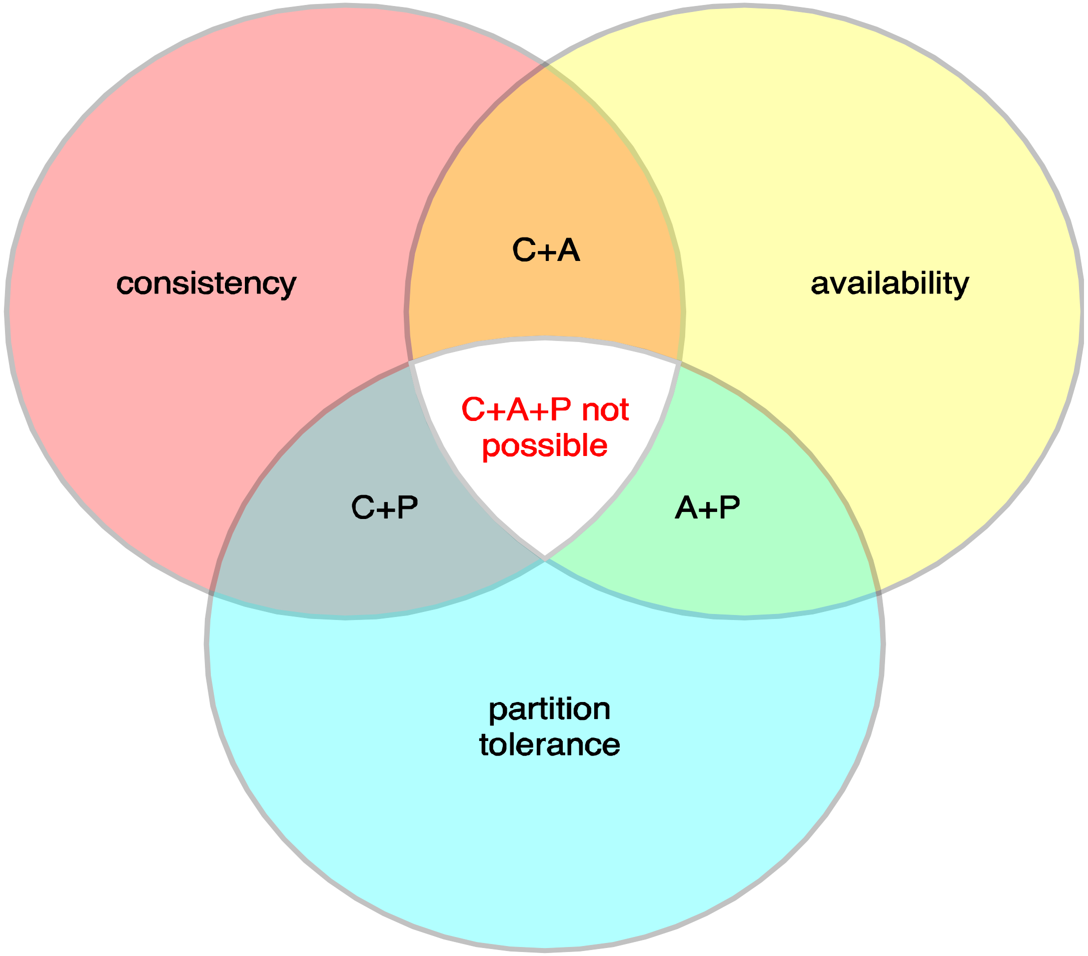


Figure 5.17 A Venn diagram summarizing the CAP theorem.

Exercise6: Prove CAP theorem

1. If consistency (C) and availability (A) are required, why partition tolerance (P) is not possible?
2. If consistency (C) and partition tolerance (P) are required, why availability (A) is not possible?
3. If availability (A) and partition tolerance (P) are required, why consistency (C) is not possible?

Exercise7: Choice among CAP?

Why CAP is generally a choice between choosing either C+P or A+P?

ACID: a database design principle

ACID is a transaction management style that leverages **concurrency controls** to ensure **consistency** is maintained.

- atomicity
- consistency
- isolation
- durability

Atomicity ensures that all operations will always **succeed or fail completely(*all-or-none*)**.

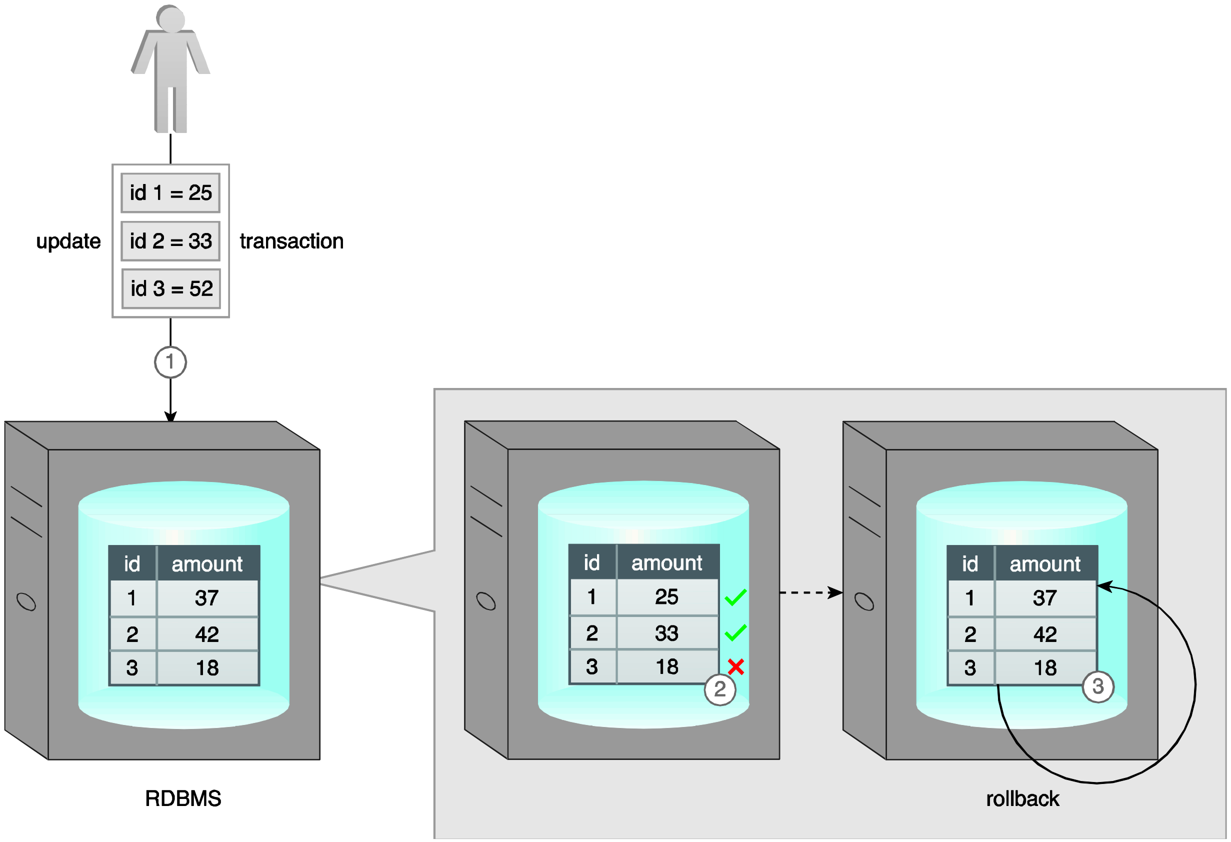


Figure 5.18 An example of the atomicity property of ACID is evident here.

Consistency ensures that the database will always remain in a consistent state by ensuring that ***only data that conforms to the constraints of the database schema*** can be written to the database.

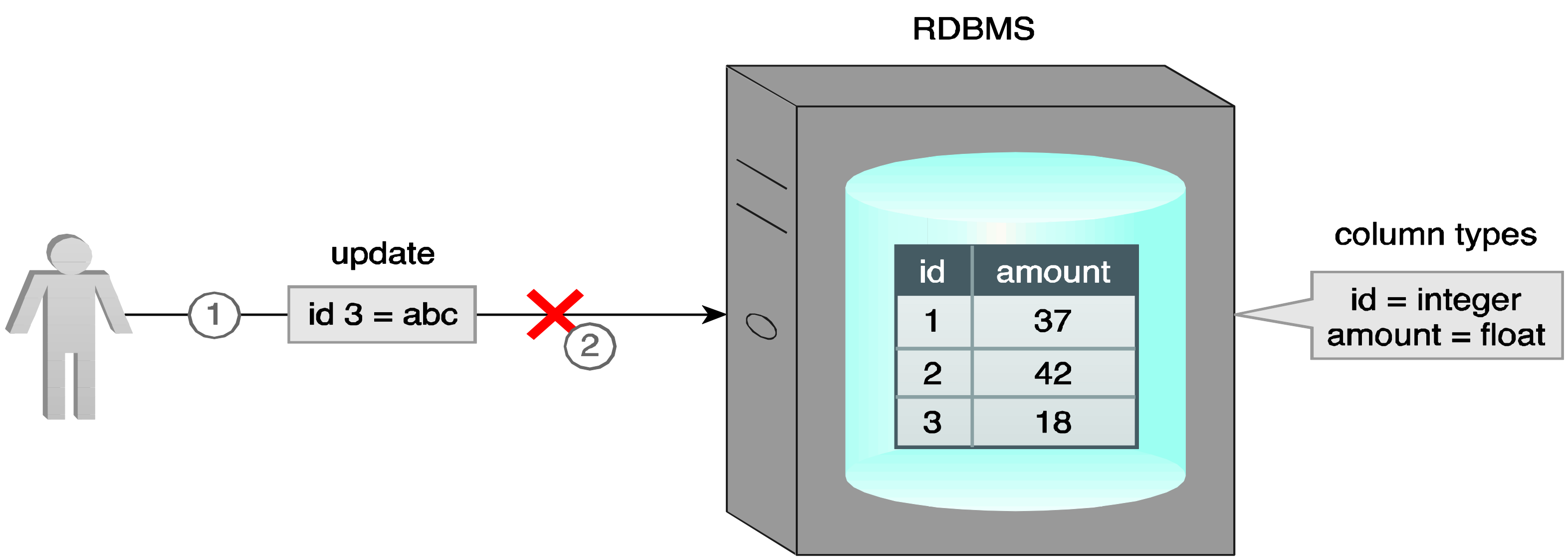


Figure 5.19 An example of the consistency of ACID.

Isolation ensures that the results of a ***transaction are not visible*** to other operations until it is complete.

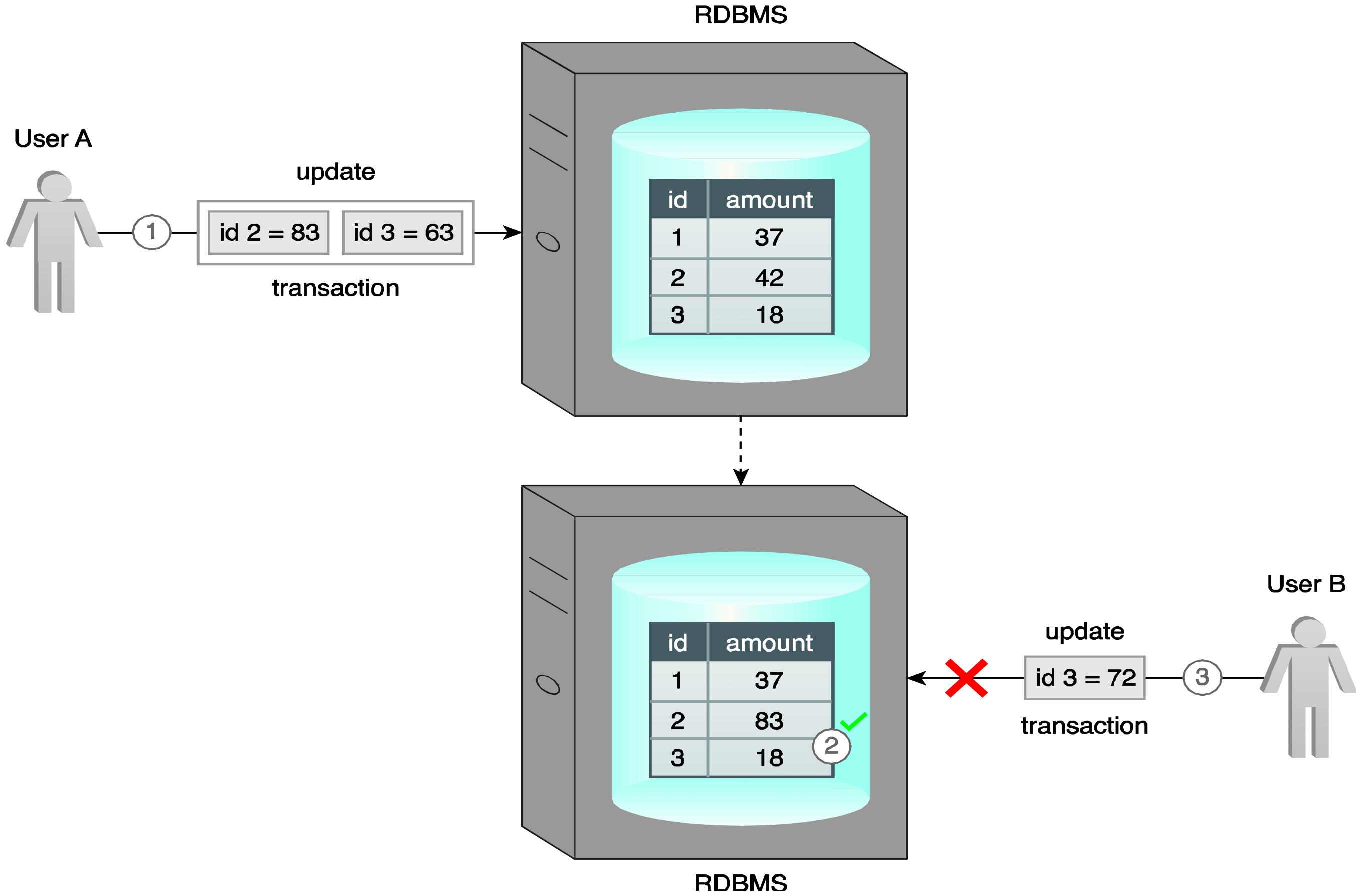


Figure 5.20 An example of the isolation property of ACID.

Durability ensures that the results of an operation are permanent.

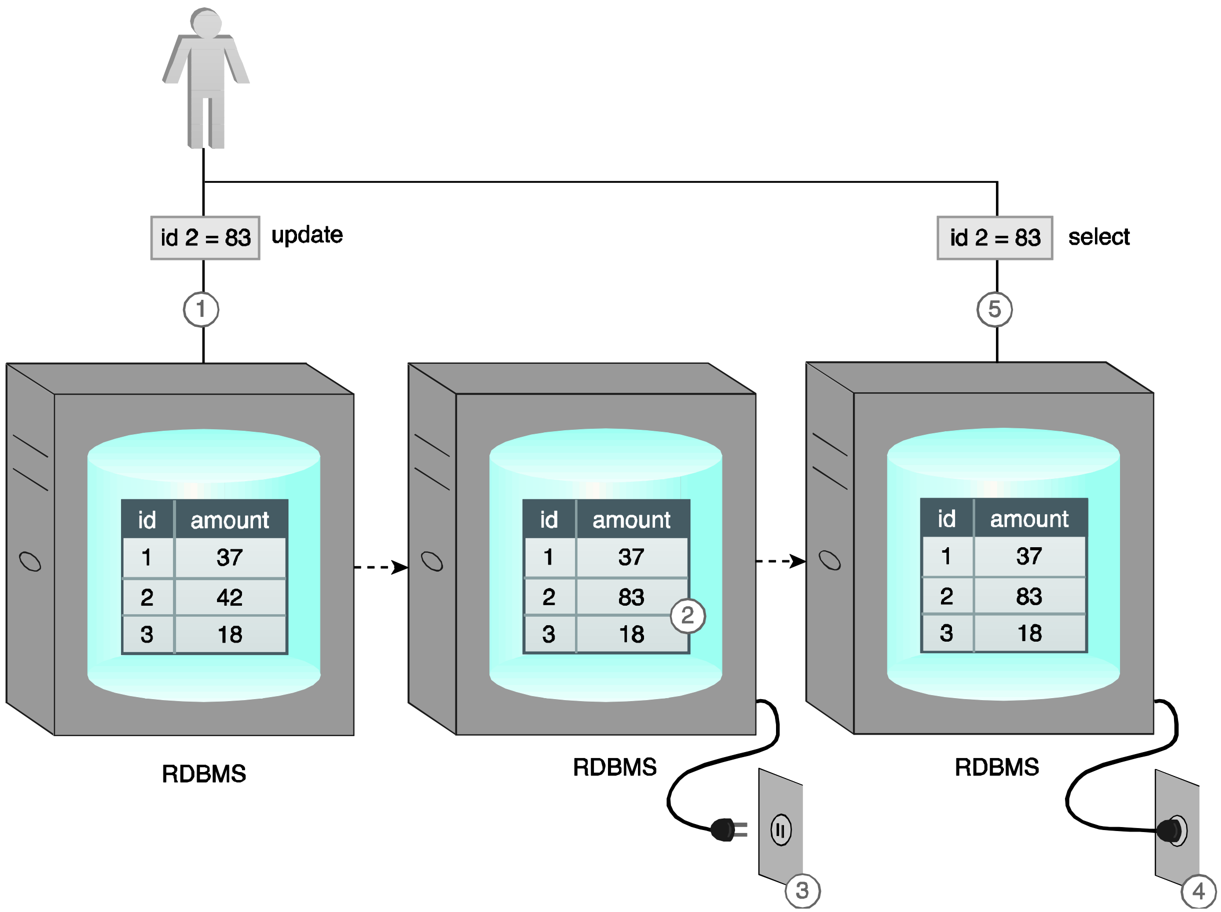


Figure 5.21 The durability characteristic of ACID.

BASE

BASE is a *database design principle* based on the CAP theorem and leveraged by database systems that use *distributed technology*.

BASE stands for:

- basically available
- soft state
- eventual consistency

Database supporting BASE favors **availability** over consistency

Basically Available - always acknowledge a client's request, either in the form of the requested data or a success/failure notification.

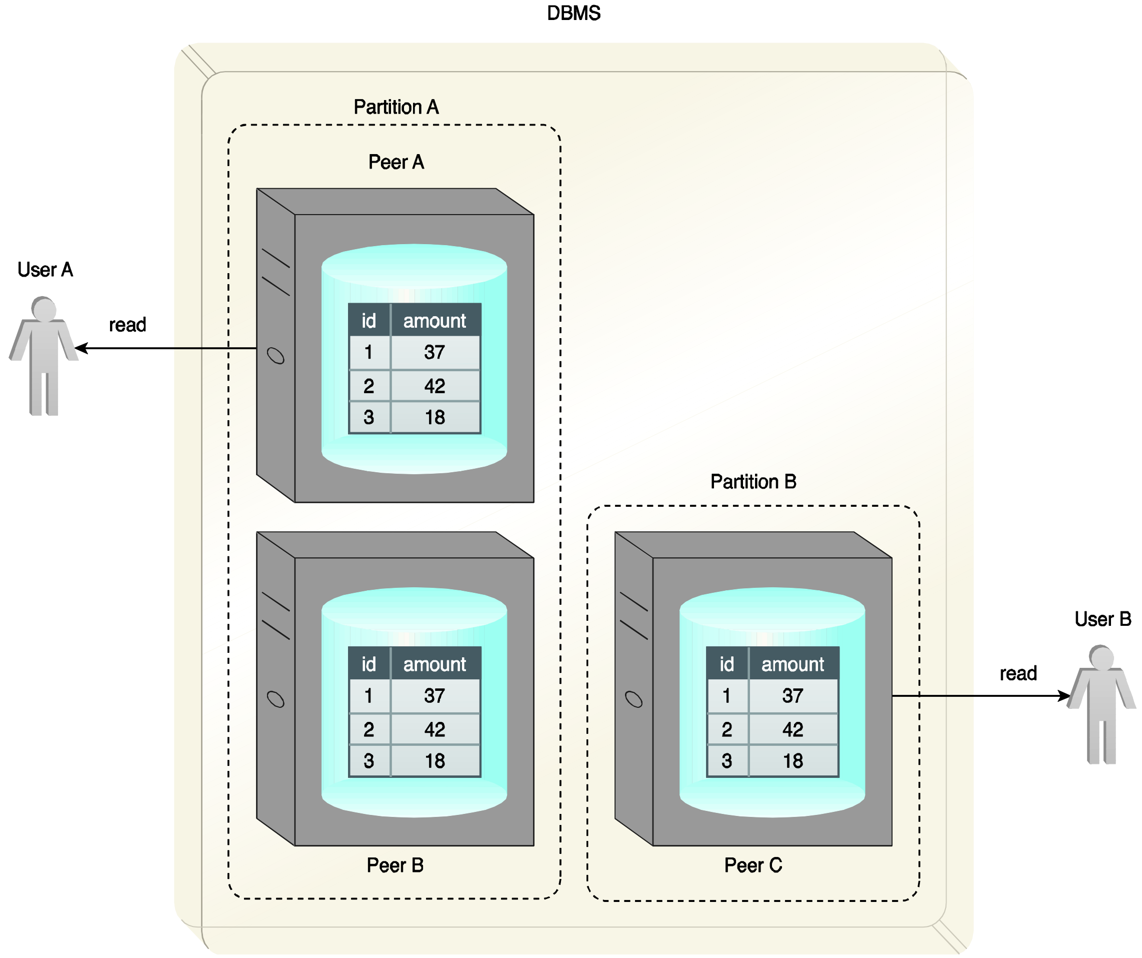


Figure 5.23 User A and User B receive data despite the database being partitioned by a network failure.

Soft state means that a database may be in an inconsistent state when data is read

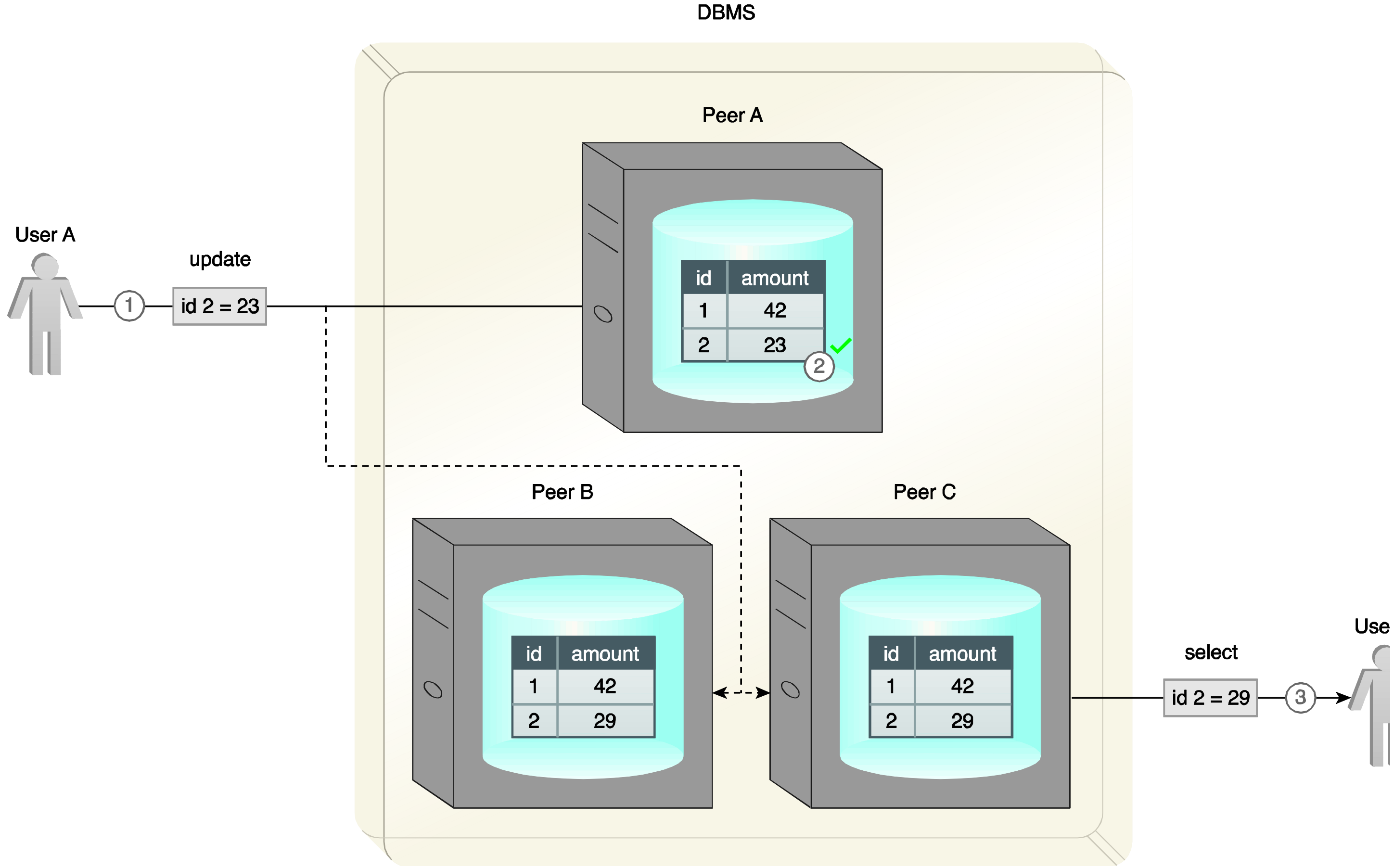


Figure 5.24 An example of the soft state property of BASE is shown here.

Eventual consistency is the state in which reads by different clients, immediately following a write to the database, may not return consistent results

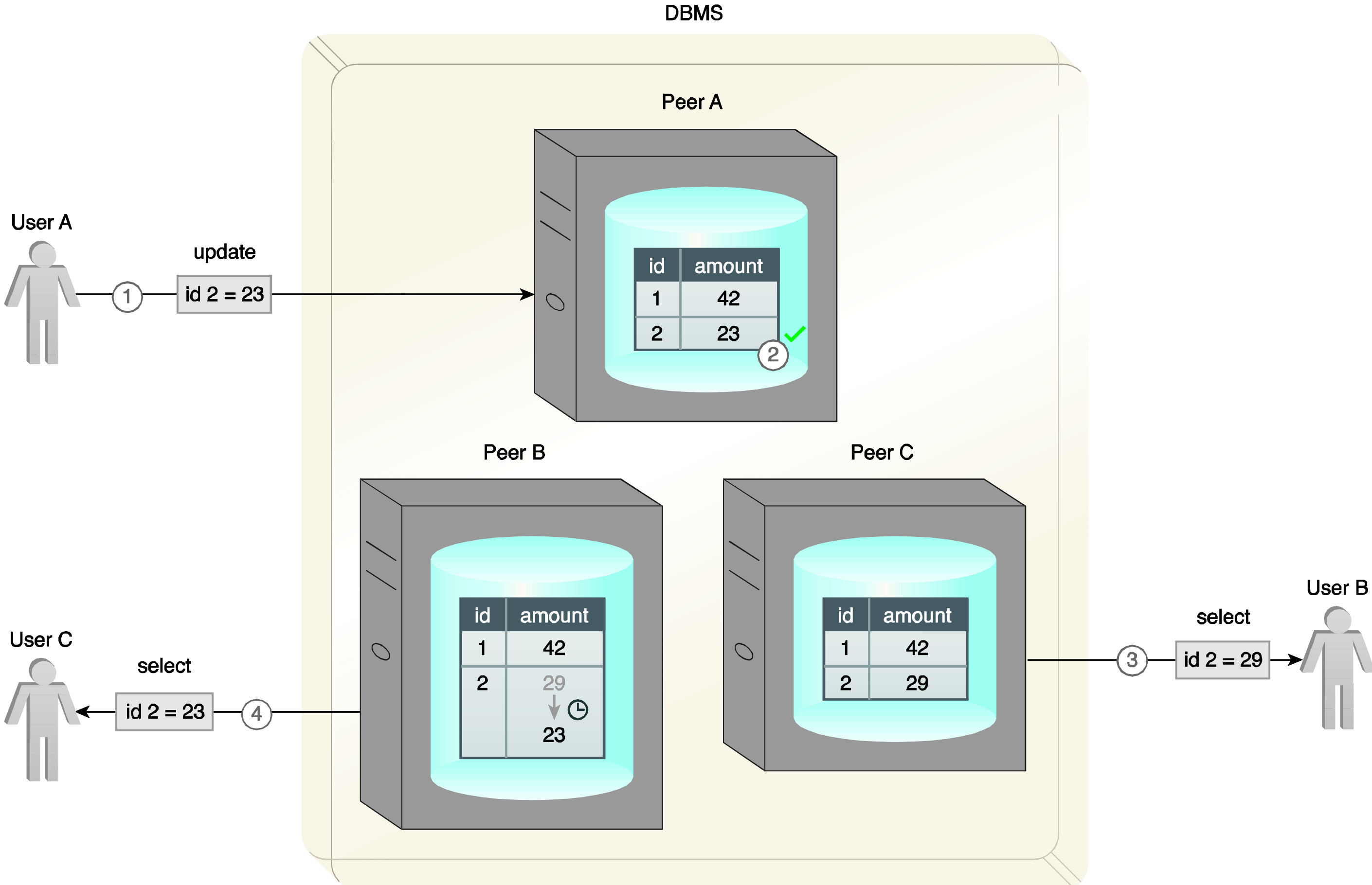


Figure 5.25 An example of the eventual consistency property of BASE.