How Databases Guarantee Durability

Durability is a fundamental property of modern databases, ensuring that once a transaction is committed, the data persists even in the face of system crashes, power failures, or unexpected restarts. This document explores the primary mechanisms used by databases to provide durability guarantees as part of the ACID properties.

1. Write-Ahead Logging (WAL)

Write-Ahead Logging is the most fundamental and widely-used technique for ensuring durability in modern databases. It follows a simple but powerful principle that protects data integrity.

How It Works:

Every change made to the database is first recorded in a sequential log file (the WAL) on persistent disk storage before being applied to the actual database pages. The WAL contains enough information to redo (reapply) or undo (reverse) each transaction.

If a crash occurs, the database reviews the WAL to recover all committed transactions and roll back incomplete ones, restoring data consistency. This process ensures no committed data is lost.

Key Advantages:

- **Efficient Sequential Writes:** Log records are written sequentially, which is much faster than random disk updates
- **Fast Recovery:** The system can quickly replay the log after crashes to restore committed transactions
- **Point-in-Time Recovery:** Historical transactions can be replayed to restore data to any previous state
- **Reduced Disk I/O:** Only log records need immediate flushing, while database pages can be written lazily

2. Shadow Paging

Shadow paging is an alternative copy-on-write technique that maintains two versions of database pages simultaneously to ensure durability without requiring complex logging mechanisms.

How It Works:

The system maintains two sets of pointers (page tables): the **shadow page table** pointing to the original, unmodified database pages representing the stable pre-transaction state, and the **current page table** tracking new copies of modified pages.

When a transaction modifies data, changes are written to new page copies rather than updating pages in-place. Upon commit, the current page table atomically replaces the shadow, making all changes permanent at once.

If a failure or crash happens before the commit, the system simply discards the uncommitted new pages and continues from the untouched shadow pages, eliminating the need for undo and redo operations.

Characteristics:

- **Atomic Commits:** Changes become visible all at once with a single pointer switch
- **Simple Recovery:** No log scanning or replay needed after crashes
- **Storage Fragmentation:** Can lead to scattered data across disk over time
- **Less Common Today:** Performance challenges on modern storage systems limit widespread adoption

3. Checkpointing

Checkpointing is a critical optimization technique that works alongside logging mechanisms to speed up recovery and maintain system performance. Without checkpoints, recovering from a crash would require scanning the entire log file from the beginning, which could take extremely long.

How It Works:

During a checkpoint operation, the database performs several coordinated actions:

- Forces all log records from memory to stable storage
- Flushes all modified database pages to disk
- Writes a special checkpoint record in the log marking this savepoint

When recovery is needed after a crash, the system only needs to scan the log from the last checkpoint forward, since all transactions committed before that point are already permanently saved. This dramatically reduces recovery time while maintaining durability guarantees.

Benefits:

- **Faster Recovery:** Limits the amount of log that needs to be processed
- **Log Management:** Allows old log entries before checkpoints to be safely discarded
- **Performance Balance:** Reduces recovery time without significantly impacting normal operations

4. Storage Synchronization Techniques

Beyond the core techniques, databases use additional mechanisms to ensure data truly reaches persistent storage and can survive various types of failures.

Key Mechanisms:

- **fsync Operations:** Force data from operating system buffers to physical disk, guaranteeing persistence
- **Checksums:** Verify data integrity after writes to detect corruption
- **Storage Redundancy:** Replication across multiple disks or servers protects against hardware failures
- **Battery-Backed Caches:** Provide additional protection for in-flight writes during power failures

Together, these mechanisms ensure that once a transaction is committed, its changes are permanent and will survive any subsequent system failure, including crashes, power outages, and hardware malfunctions.

Summary Comparison

Technique	Durability Mechanism	Common Usage
Write-Ahead Logging	Logs all changes before application; supports redo/undo; sequential I/O	Most modern databases (PostgreSQL, MySQL, Oracle)
Shadow Paging	Copy-on-write pages with atomic pointer switch at commit	Older/simpler databases (SQLite, LMDB)
Checkpointing	Periodic savepoints; limits recovery time and log scan	WAL-based databases (all major systems)
Storage Synchronization	Forces data to physical disk (fsync); checksums, replication	All production systems (critical for durability)

Conclusion

Database durability is achieved through a combination of complementary techniques that work together to protect committed data. Write-Ahead Logging provides the foundation for recovery, checkpointing optimizes performance, and storage synchronization ensures data reaches physical media.

Understanding these mechanisms is essential for database administrators, developers, and anyone working with critical data systems. Each technique has specific trade-offs in terms of performance, complexity, and recovery guarantees, and modern databases often combine multiple approaches to achieve optimal durability characteristics.