Spicing Up Your Data

The Chemistry of ACID in Database Management

Anton Nazarov

Does your data deserve a stable relationship and to be loved too?

Yes!

Any database is a real heart of any application!

ACID

Anton Nazarov

Atomicity

Atomicity

Atomicy

Atomicity ensures that each transaction is treated as a single, indivisible unit of work.

All operations within the transaction succeed and are committed.

OR

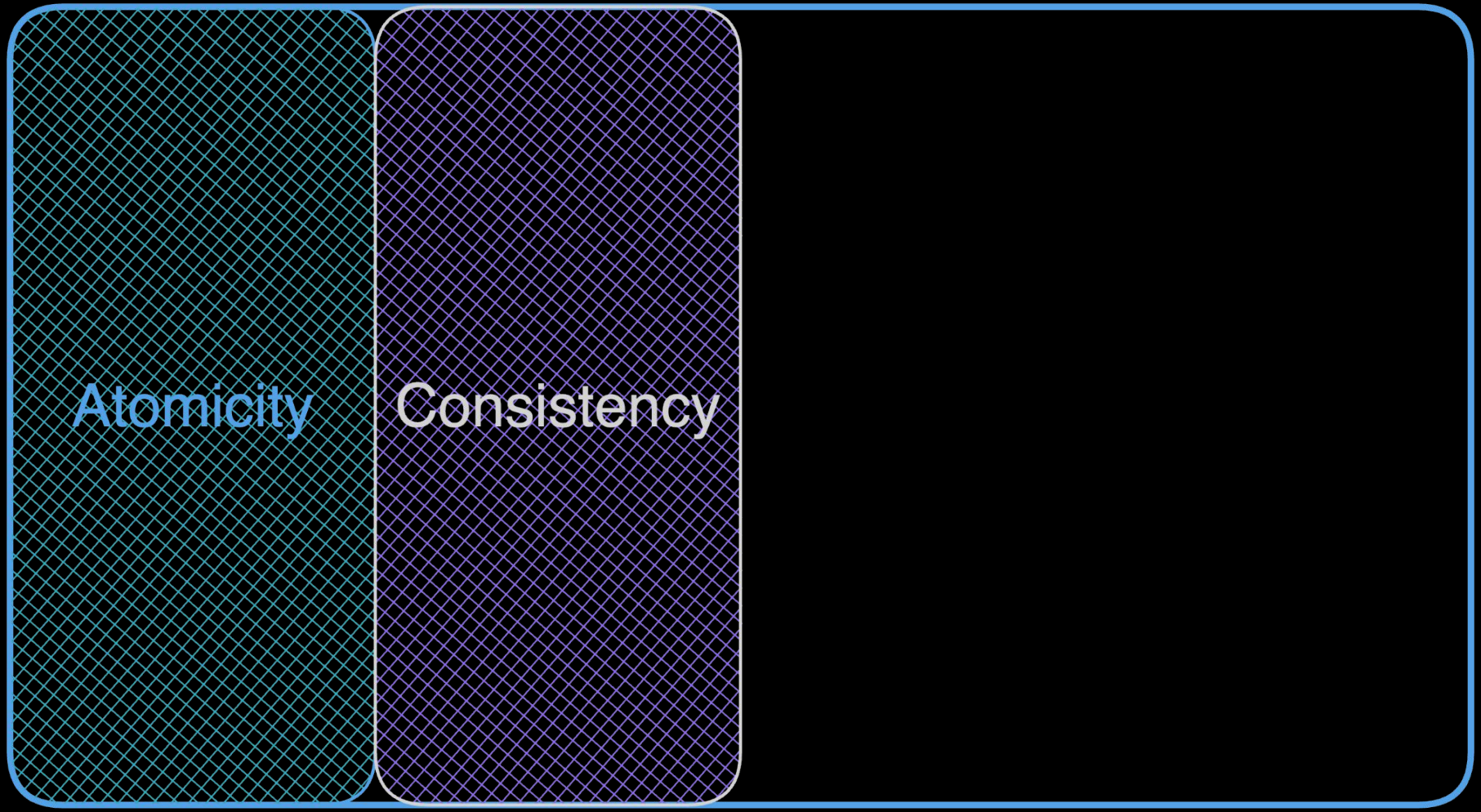
If any operation fails, the entire transaction is aborted and the database is restored to its state before the transaction started.

How to touch atomicity?

- We **disable autocommit** to start a transaction explicitly.
- We **update** an employee's salary and check for errors.
  - If there's an error, we **rollback** the transaction.
  - If **successful**, we **update** the department's budget and check for errors.
    - If any occur, we **rollback**.
- If all operations succeed, we **commit** the transaction to finalise changes.

Result: All data will either be saved entirely or not at all.

```
-- Disable autocommit to start a transaction explicitly
SET AUTOCOMMIT = 0;

START TRANSACTION;
UPDATE employees
SET salary = 60000
WHERE employee_id = 101;

-- Check for errors and rollback if necessary
IF @@ERROR <> 0 THEN
    ROLLBACK;
ELSE

    UPDATE departments
    SET budget = budget + 10000
    WHERE department_id = 1;

    -- Check for errors and rollback if necessary
    IF @@ERROR <> 0 THEN
        ROLLBACK;
    ELSE
        -- Commit the transaction if all operations are successful
        COMMIT;
    END IF;
END IF;
```


Atomicity
Consistency

Consistency

It ensures that only valid data is written to the database. Data written to the database must satisfy:

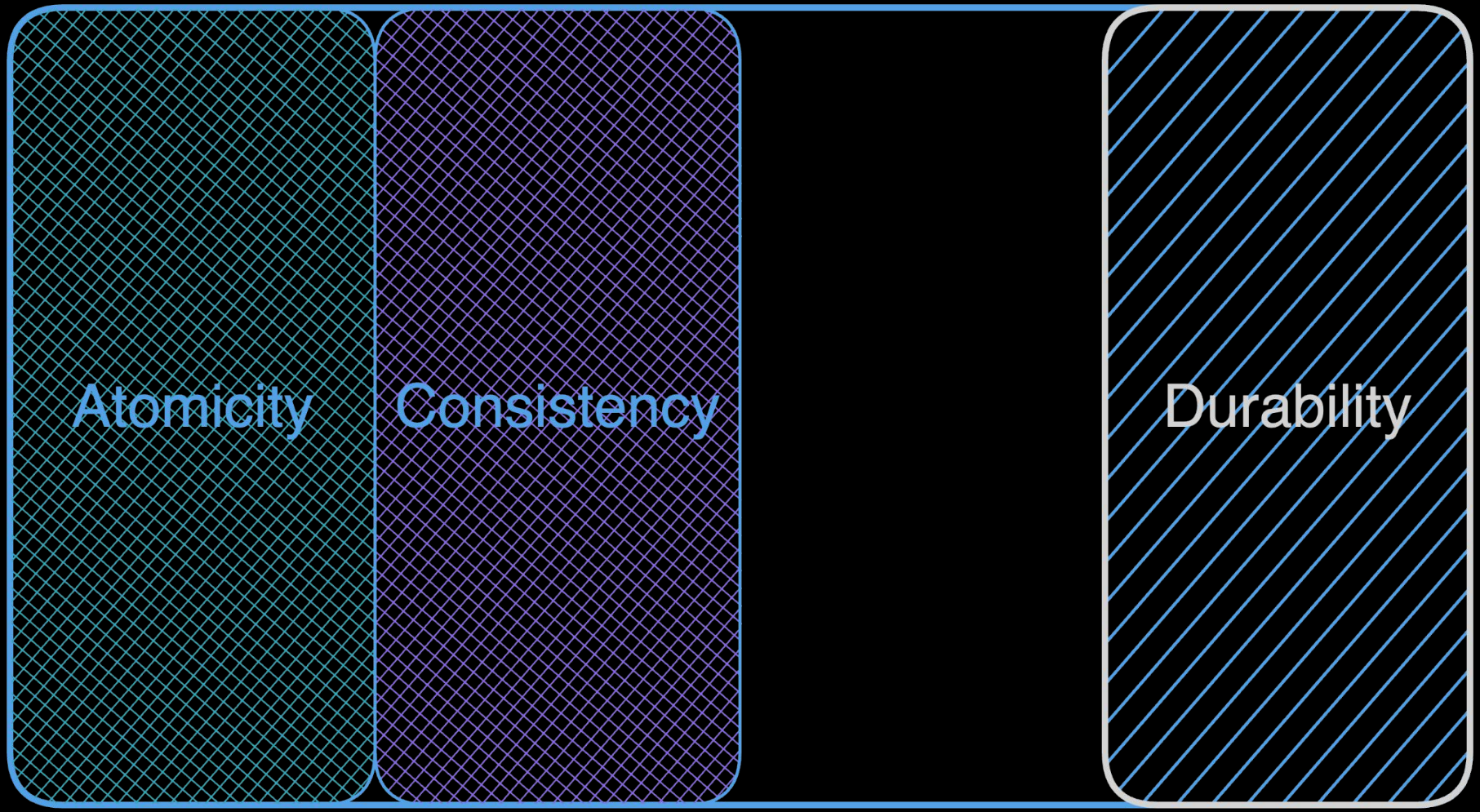
- Defined rules
- Constraints as
  - Unique key
  - Referential integrity
  - Data type constraint
  - …
- Relationships

How to touch consistency?

- We **START TRANSACTION**.
- We deduct $100 from sender's account.
- We credit $100 to the recipient's account.
- If both updates (deduction and credit) succeed without errors, we **COMMIT** the transaction.
- If there's an error during either update, such as insufficient funds or a database error, we roll back the transaction using **ROLLBACK**.

Result: Consistency here ensures that the total amount of money in the system remains unchanged.

```sql
-- Start transaction
START TRANSACTION;

-- Deduct $100 from sender's account
UPDATE accounts
SET balance = balance - 100
WHERE account_id = 101;

-- Credit $100 to recipient's account
UPDATE accounts
SET balance = balance + 100
WHERE account_id = 102;

-- Commit or rollback transaction based on success
COMMIT; -- If both updates succeed
ROLLBACK; -- If any update fails
```


Atomicity
Consistency
Durability

Durability

Durability ensures that the effects of committed transactions persist in the database, even if the system crashes immediately after the transaction completes.

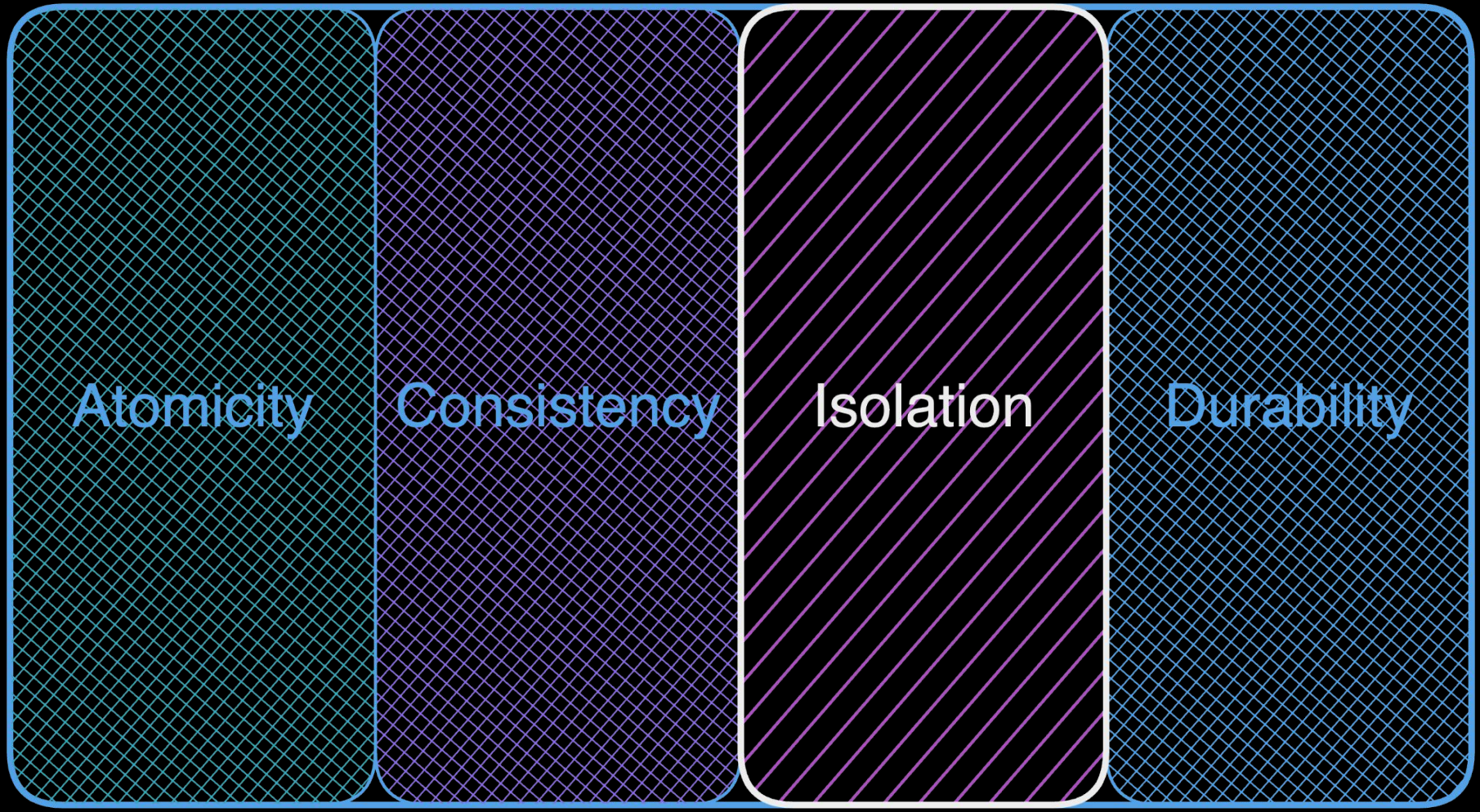
How to touch durability?

- Check logs and system behavior
- Test the durability of transactions,
- Ensuring that your database reliably maintains data integrity and persistence

```sql
-- Disable autocommit to start a transaction explicitly
SET AUTOCOMMIT = 0;

-- Start transaction
START TRANSACTION;

-- Insert data into a test table
INSERT INTO test_table (id, name) VALUES (1, 'Test Data');

-- Simulate a system crash or failure
-- Manually stop MySQL service or kill MySQL process

-- Restart MySQL and allow it to recover
-- Check MySQL error log for recovery information

-- Verify data durability
SELECT * FROM test_table WHERE id = 1;

-- Commit or rollback the transaction (depending on test outcome)
COMMIT; -- If data is durable and present after recovery
ROLLBACK; -- If data is lost or not recoverable
```

Logs

- **Transaction logging**, changes which are first recorded in logs before applying them to databases, ensuring recovery from crashes.
- **Write-Ahead Logging (WAL)** ensures durability by logging changes before updating the database, even after committing transactions.


Atomicity
Consistency
Isolation
Durability

Isolation - Concurrency Control

Isolation prevents concurrent transactions from interfering with each other, avoiding issues like:

- Dirty reads
- Non-repeatable reads
- Phantom reads

Databases offer mechanisms like

- Locking (row-level, table-level)
- Multiversion concurrency control (MVCC)
- Timestamp ordering

Isolation levels

- **Read Uncommitted**: Allows transactions to read data that has been modified but not yet committed by other transactions..
- **Read Committed**: Ensures that transactions only read data that has been committed by other transactions.
- **Repeatable Read**: Ensures that once a transaction reads data, it will see the same data again in subsequent reads within the same transaction.
- **Serializable**: Provides the highest level of isolation by ensuring that transactions are executed as if they were serialized, one after the other.

What popular databases that support ACID?

SQL related databases:

- MySQL
- MariaDB
- PostgreSQL
- Oracle
- SQL Server

Note: There could be various options for **ACID** support across different databases.

For example: **MySQL** and **MariaDB** have their own storage engines (**MyISAM** is **deprecated** since 2010 and **no longer available for MySQL**, **Memory**), which not support all **ACID** properties due to design, use cases, and other factors.

What about NoSQL databases?

Atomicity: NoSQL databases **may** offer atomicity at the document or operation leve.

What about NoSQL databases?

Consistency: NoSQL databases typically offer eventual consistency rather than strong consistency. This means that updates to the database propagate asynchronously, and there may be a period where different replicas or partitions of the data are inconsistent.

What about NoSQL databases?

Isolation: Isolation levels in NoSQL databases vary and may not provide the same strict guarantees as traditional relational databases. Concurrent operations may affect each other, and isolation levels may be configurable based on application requirements.

What about NoSQL databases?

Durability: NoSQL databases ensure durability by persisting data to disk or other storage media, but the timing and guarantees around durability can vary. Some databases may offer asynchronous durability, where acknowledgments are returned to clients before data is fully persisted.

Take care of your hearts and data inside.

Thank you

Tony Nazarov

LinkedIn:
https://linkedin.com/in/tonynazarov

Email:
tonynazarov.nz@gmail.com

Presentations:
https://linktr.ee/tonynazarov.nz

SCAN ME

My Contacts: