Stray Animals Shelter Management System

ACID & Transaction Documentation

Database: PostgreSQL 16

Table of Contents

1. ACID Properties Overview

ACID ensures reliable database transactions. PostgreSQL provides full ACID compliance through Write-Ahead Logging (WAL) and Multi-Version Concurrency Control (MVCC).

Property	Definition
Atomicity	Transaction is all-or-nothing. All operations complete or none take effect.
Consistency	Transaction moves database from one valid state to another. All constraints must be satisfied.
Isolation	Concurrent transactions execute as if serial. Intermediate states are not visible to others.
Durability	Committed changes are permanent even after crashes. WAL ensures this.

2. Scenario 1: Complete Animal Adoption

2.1. Description

Administrator processes adoption for animal 7 (Buddy). Steps:

1. Verify animal is available (status = 'In Shelter')
2. Insert adoption record with status 'Completed'
3. Update animal.status to 'Adopted' via trigger
4. Log status change in audit_log

If any step fails, entire operation rolls back.

2.2. SQL Transaction

```
BEGIN;

SELECT animal_id, name, status
FROM   animal
WHERE  animal_id = 7
FOR UPDATE;

INSERT INTO adoption
  (animal_id, adopter_name, adopter_cnic, adopter_contact,
   adopter_address, employee_id, adoption_date, adoption_fee, status)
VALUES
  (7, 'Fareeha Malik', '35202-5544332-5', '0321-4433221',
   '88 Johar Town Block D, Lahore', 10, CURRENT_DATE, 2000, 'Completed');

COMMIT;
```

2.3. ACID Analysis

Property	How Satisfied
Atomicity	BEGIN...COMMIT wraps all operations. If trigger fails, entire transaction rolls back.
Consistency	Trigger enforces 'In Shelter' precondition. FK and CHECK constraints verify relationships.
Isolation	FOR UPDATE acquires exclusive row lock. Concurrent adoptions blocked until completion.
Durability	On COMMIT, WAL buffer flushed to disk. Adoption survives crashes.

3. Scenario 2: Rescue Mission Dispatch

3.1. Description

Dispatcher receives phone report of injured dog in Lahore. Steps:

1. Insert distress report with status 'Pending'
2. Verify assigned rescue team is active
3. Update report status to 'Assigned' and link team
4. Create rescue mission record

If team assignment or mission insert fails, report remains 'Pending'.

3.2. SQL Transaction

```
BEGIN;

INSERT INTO report
  (channel, reporter_name, reporter_contact, description,
   location_text, city_id, status)
VALUES
  ('Phone', 'Gulshan Parveen', '0321-7788990',
   'Injured dog near school gate, limping badly',
   'Gulshan-e-Ravi, Lahore', 1, 'Pending')
RETURNING report_id;

SELECT team_id, is_active
FROM   rescue_team
WHERE  team_id = 1
FOR SHARE;

UPDATE report
SET    assigned_team_id = 1,
       status = 'Assigned'
WHERE  report_id = :v_report_id;

INSERT INTO rescue_mission (report_id, team_id, dispatched_at, outcome)
VALUES (:v_report_id, 1, NOW(), 'Ongoing');

COMMIT;
```

3.3. ACID Analysis

Property	How Satisfied
Atomicity	All statements execute as single unit. If mission INSERT fails, report INSERT and UPDATE roll back.
Consistency	FK constraints ensure city_id and team_id reference valid records. CHECK constraints validate status.
Isolation	FOR SHARE prevents team deactivation between check and assignment.
Durability	After COMMIT, report and mission records written durably to WAL.

4. Scenario 3: Owner Animal Surrender

4.1. Description

Owner surrenders pet dog due to relocation abroad. Steps:

1. Insert animal as new shelter resident
2. Record surrender in animal_sale (amount = 0)
3. Schedule initial vet check-up in animal_care_log

All three records linked. If any step fails, entire operation rolls back.

4.2. SQL Transaction

```
BEGIN;

INSERT INTO animal
  (name, species_id, breed, gender, date_of_birth, colour,
   weight_kg, health_status, status, branch_id, intake_method)
VALUES
  ('Pepper', 1, 'Dachshund', 'F', '2020-05-10', 'Black/Tan',
   8.5, 'Healthy', 'In Shelter', 1, 'Owner Surrender')
RETURNING animal_id;

INSERT INTO animal_sale
  (animal_id, sale_type, counterparty_name,
   counterparty_contact, employee_id, transaction_date, amount)
VALUES
  (:v_animal_id, 'Owner Surrender', 'Nasreen Akhtar',
   '0345-1100998', 5, CURRENT_DATE, 0);

INSERT INTO animal_care_log
  (animal_id, employee_id, log_date, care_type, notes)
VALUES
  (:v_animal_id, 2, CURRENT_DATE, 'Check-up',
   'Initial assessment upon owner surrender intake');

COMMIT;
```

4.3. ACID Analysis

Property	How Satisfied
Atomicity	All three INSERTs wrapped in single transaction. Any failure rolls back entire transaction.
Consistency	FK, CHECK, and NOT NULL constraints validated at commit time.
Isolation	Under READ COMMITTED, new animal row invisible to others until commit.
Durability	On COMMIT, all three records durably stored via WAL.

5. Concurrency Control

5.1. Isolation Levels

PostgreSQL default isolation level is READ COMMITTED.

Level	Used For	Justification
READ COMMITTED	Care logs, reports, animal search	Prevents dirty reads, allows high concurrency.
REPEATABLE READ	Payroll, monthly hours	Ensures consistent view of shifts during computation.
SERIALIZABLE	Adoption, status updates	Prevents phantom reads with FOR UPDATE locks.

5.2. Locking Strategy

PostgreSQL MVCC allows readers and writers to coexist. Explicit locking:

- **FOR UPDATE**: Exclusive lock on animal before adoption/sale to prevent double adoption.
- **FOR SHARE**: Read rescue team status before assignment to prevent concurrent deactivation.
- Row-level locks allow concurrent operations on other rows.

5.3. Potential Issues

Issue	Risk	Mitigation
Double Adoption	HIGH	FOR UPDATE lock + trigger prevents adoption of non-'In Shelter' animals.
Lost Update on Status	MEDIUM	All status changes through triggers; SERIALIZABLE isolation.
Dirty Read on Reports	LOW	READ COMMITTED prevents dirty reads by default.
Phantom Read in Payroll	LOW	REPEATABLE READ prevents new shift rows appearing mid-calculation.

6. Durability Implementation

PostgreSQL implements durability through Write-Ahead Logging (WAL). Before data page modification, change is written to WAL log. After crash, database replays WAL to recover committed transactions.

6.1. Recommended Settings

```
synchronous_commit = on
wal_level          = replica
fsync              = on
checkpoint_timeout = 5min
max_wal_size       = 1GB
```

6.2. Backup Strategy

Method	Description
Logical Backup	pg_dump runs nightly for full logical backup. Stored on separate server, retained 30 days.
WAL Archiving	Continuous WAL segments archived to remote storage for Point-in-Time Recovery.
Replication	Hot-standby replica kept in sync via streaming replication for disaster recovery.