# Telco ACIDShield

This project showcases how ACID principles protect critical telecom operations, guaranteeing data integrity and service reliability.

hussein-mohamed7

# Meet Ahmed

## Ahmed clicks “Upgrade Internet” What could possibly go wrong!

Double Charge:
Ahmed sees: Two bills for one service.
The Problem: Transaction stopped midway.

Negative Balance:
His account shows a balance below zero.
The Problem: Rules weren’t enforced

Bundle Clash:
Two bundles are charged, but only one works.
The Problem: Two requests collide

Missing Upgrade:
He pays for speed but nothing changes.
The Problem: Change wasn’t saved

# How “Telco ACIDShield” Stops the Chaos!

"ACID: Four principles that keep transactions reliable — Atomicity, Consistency, Isolation, and Durability."

Double Charge:
Fix: Atomicity – All or nothing.
The whole process runs or none of it does.

Negative Balance:
Fix: Consistency – Data stays valid.
No rules are broken during updates.

Bundle Clash:
Fix: Isolation – No interference.
Requests are handled one at a time.

Missing Upgrade:
Fix: Durability – Changes stick.
Even a crash can’t erase what’s confirmed.

# Project Overview
# Purpose & Workflow

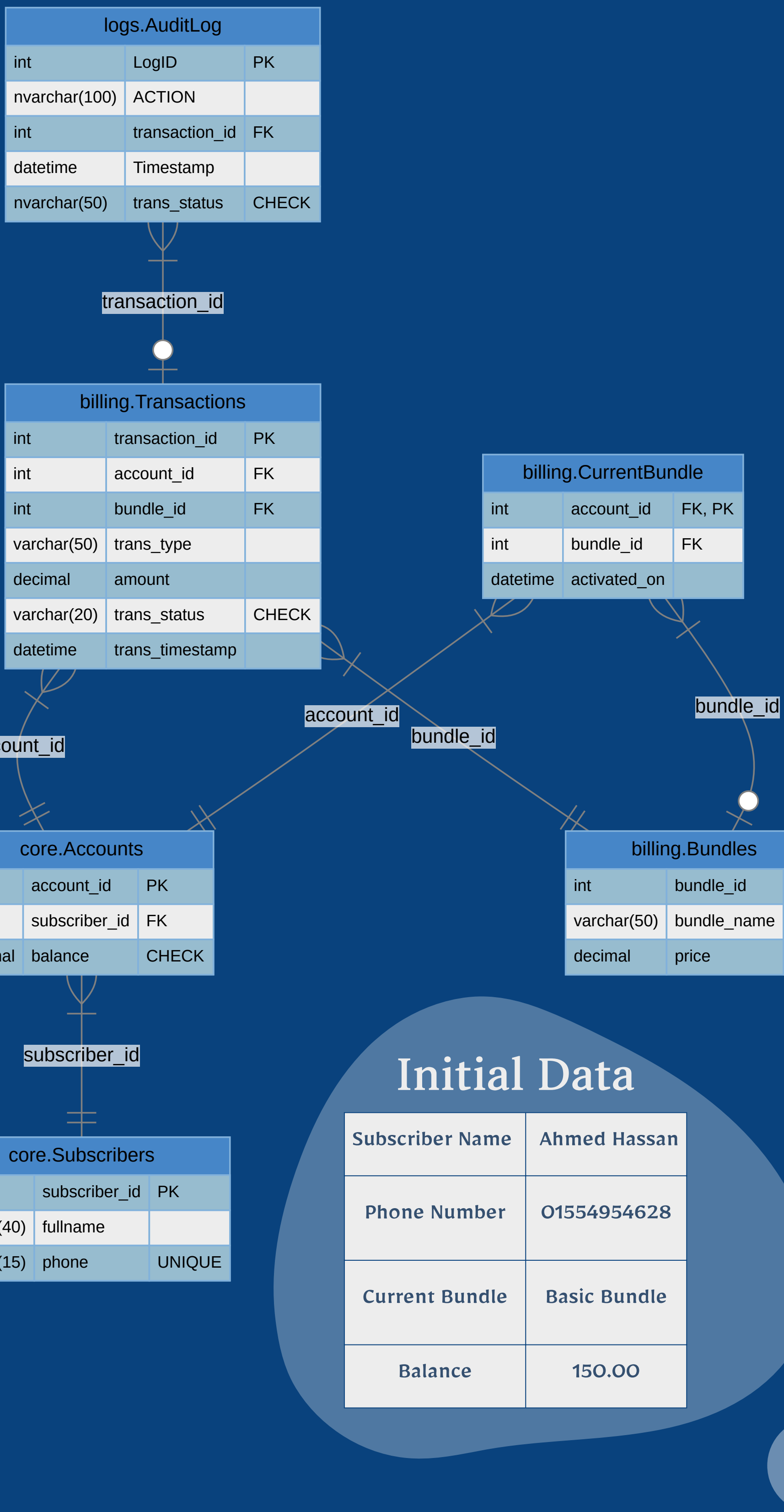
## Goal & Outcome

- Build a small telecom system designed to protect users from billing errors, bundle issues, and incorrect account balances.
- Demonstrate how ACID principles protect business logic and ensure system reliability.

## Workflow Steps:

- Database & Schema Design
- Initial Data Seeding
- Stored Procedure: SubscribeToBundle
- ACID Enforcement for Transaction Safety

# ERD: System Structure & Initial Data




## Initial Data

| | |
|---|---|
| Subscriber Name | Ahmed Hassan |
| Phone Number | 01554954628 |
| Current Bundle | Basic Bundle |
| Balance | 150.00 |

# SubscribeToBundle

Stored Procedure

## The Structure and function of the procedure

### Step 1

Get account ID and bundle ID

### Step 2

Check if account exists

### Step 3

Check if bundle exists and get price

### Step 4

Check balance is enough

### Step 5

Remove old subscription (if any)

### Step 6

Deduct balance

### Step 7

Subscribe to new bundle

# SubscribeToBundle

Stored Procedure

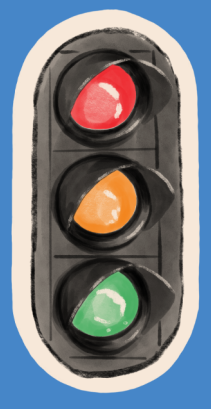
How ACID is applied to the procedure

## A-Atomicity

- The procedure uses BEGIN TRANSACTION and COMMIT/ROLLBACK.
- If any step fails, the entire transaction is rolled back.

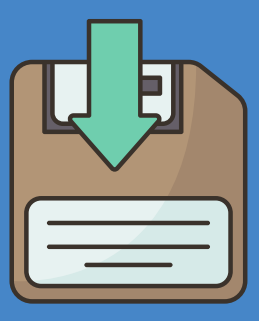
## C-Consistency

- All constraints (like foreign keys and balance check) are enforced.
- Data remains valid before and after the transaction.



## I-Isolation

- SET TRANSACTION ISOLATION LEVEL SERIALIZABLE ensures full isolation.
- Prevents dirty, non-repeatable, and phantom reads.



## D-Durability

- Once COMMIT is called, changes are permanently saved.
- Even if a crash happens, committed changes remain.

# Telco ACIDShield

**Made it easy for Ahmed to click 'Upgrade Internet' without hesitation**

```
SELECT profile FROM network WHERE platform = 'LinkedIn';
-- connect('linkedin.com/in/hussein-mohamed7')
```

**hussein-mohamed7**

```
SELECT repo FROM github WHERE author = 'me';
-- source('github.com/husseinMohamed7/
Telco-ACIDShield')
```

**husseinMohamed7/ Telco-ACIDShield**