Banking Core - Database Schema Documentation

ER Diagram & Table Explanation

Version: 1.0.0
Last Updated: January 2026
Database: MySQL 8.0 (InnoDB)
Target Audience: DBMS Undergraduate Students, Faculty

1. Introduction to Database Design

Banking Core implements a **relational database model** designed for financial integrity and audit compliance. The schema follows these core principles:

1. **Double-Entry Accounting** – Every transaction creates balanced DEBIT/CREDIT pairs
2. **Append-Only Ledger** – Ledger entries are never modified or deleted
3. **Referential Integrity** – Logical foreign key relationships maintained at application level
4. **Optimistic Locking** – Version columns prevent concurrent modification conflicts
5. **Audit Trail** – Complete history of all financial and system events

Design Philosophy

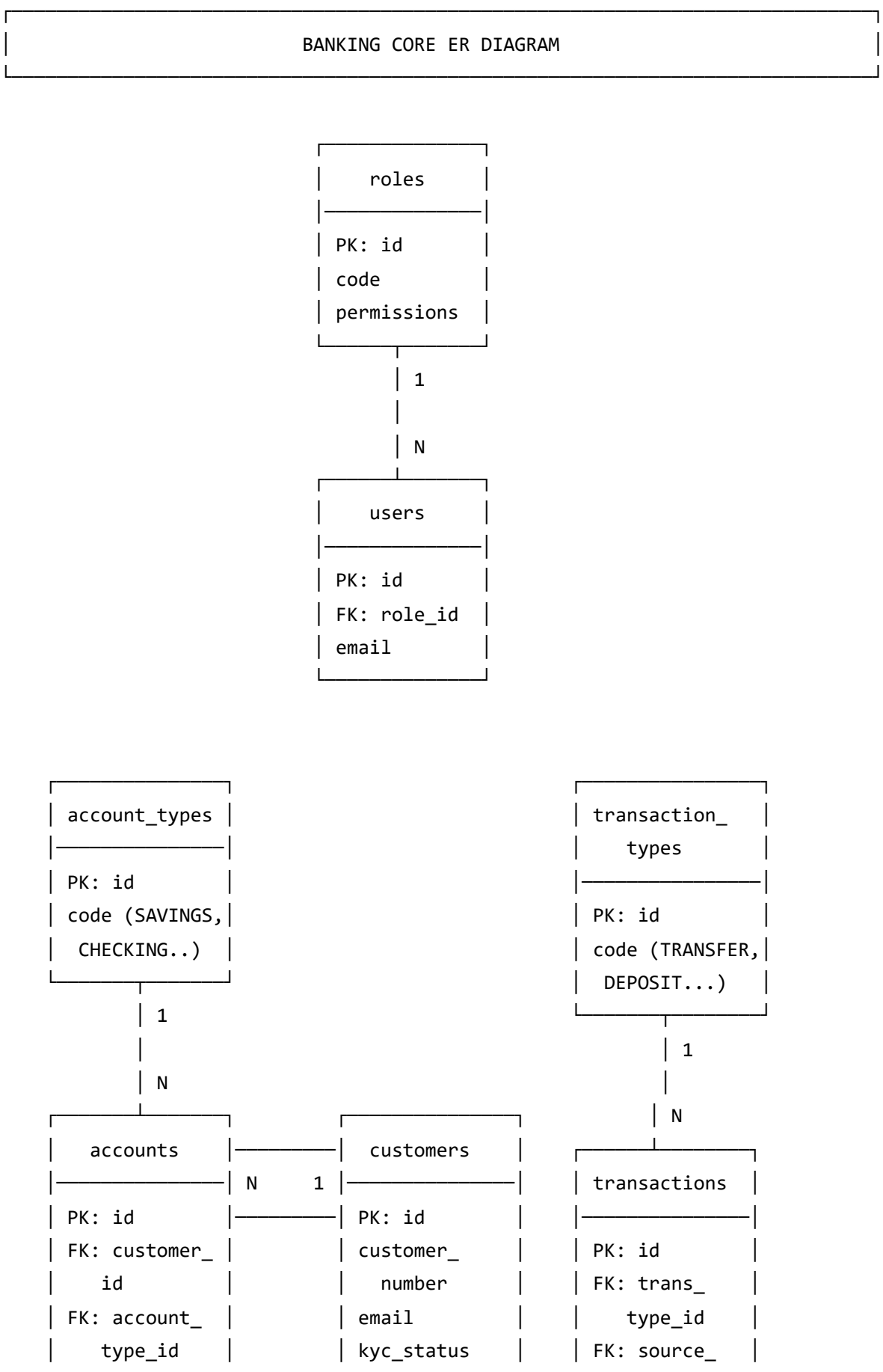
> **Correctness > Convenience > Speed**

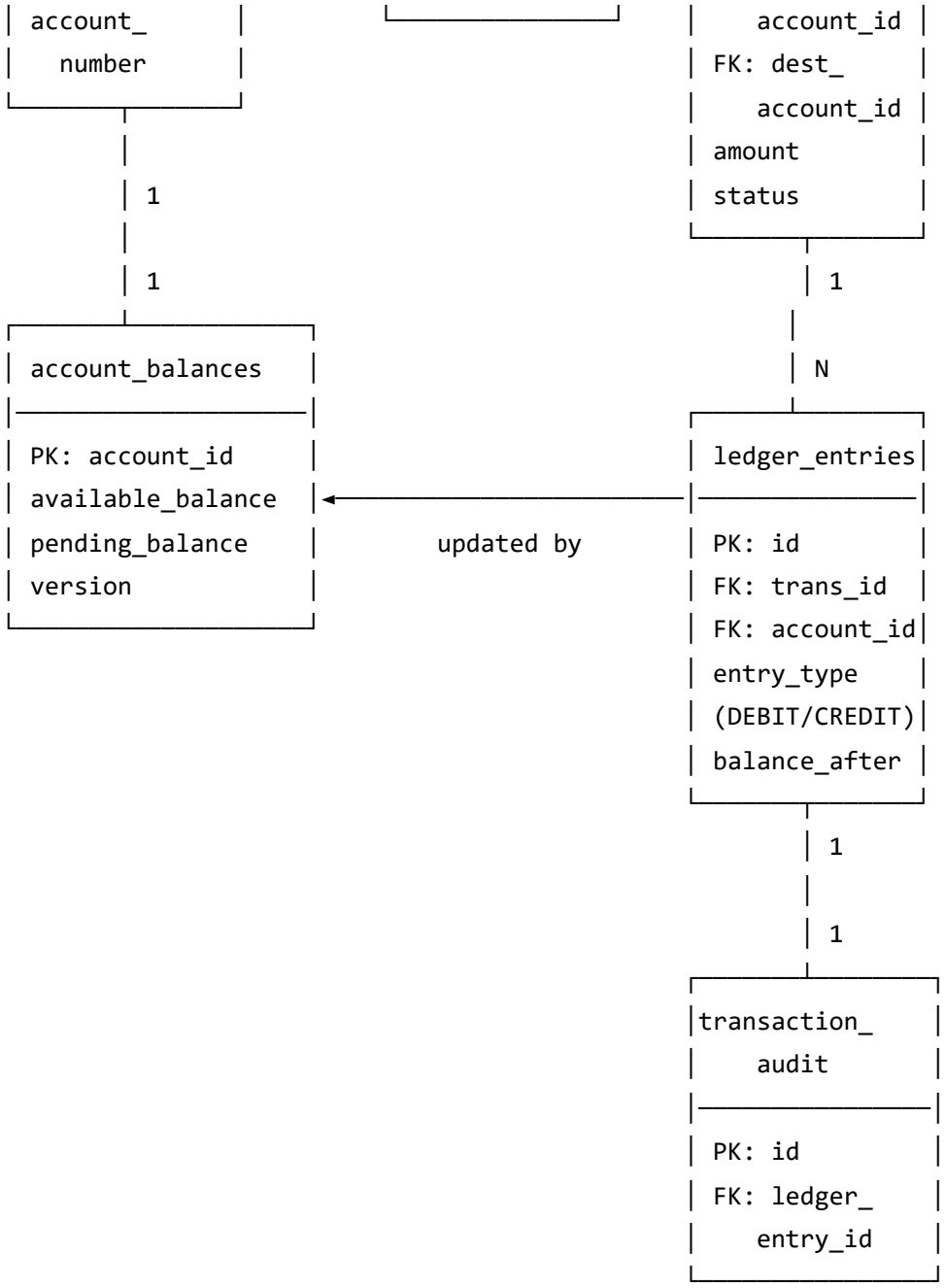
2. Overall ER Model Description

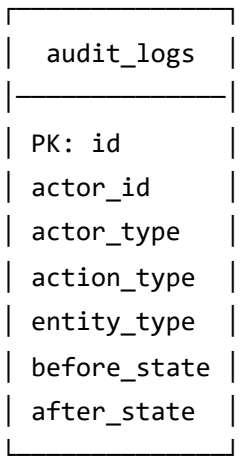
2.1 Entity Categories

Category	Entities	Purpose
Identity & Access	roles, users, customers	User management and authentication
Account Management	accounts, account_types, account_balances	Bank account lifecycle
Financial Core	transactions, transaction_types, ledger_entries	Money movement records
Audit & Compliance	transaction_audit, audit_logs	Regulatory compliance
System	system_config, events, outbox, idempotency_keys	Infrastructure support

2.2 ER Diagram (Chen Notation - Textual)




account_
number
1
1
account_balances
PK: account_id
available_balance
pending_balance
version
updated by
account_id
FK: dest_
account_id
amount
status
1
N
ledger_entries
PK: id
FK: trans_id
FK: account_id
entry_type
(DEBIT/CREDIT)
balance_after
1
1
transaction_
audit
PK: id
FK: ledger_
entry_id


audit_logs
PK: id
actor_id
actor_type
action_type
entity_type
before_state
after_state

2.3 Relationship Summary

Relationship	Type	Description
roles → users	1:N	One role has many users
customers → accounts	1:N	One customer has many accounts
account_types → accounts	1:N	One type classifies many accounts
accounts → account_balances	1:1	Each account has exactly one balance record
transaction_types → transactions	1:N	One type categorizes many transactions
accounts → transactions	1:N	Source/destination account references
transactions → ledger_entries	1:N	One transaction creates 2+ ledger entries
ledger_entries → transaction_audit	1:1	Each entry has one audit copy

3. Entity Descriptions

3.1 roles

Purpose: Defines user roles and their permissions for access control.

Column	Type	Nullable	Description
id	BIGINT	NO	Primary Key (AUTO_INCREMENT)
code	VARCHAR(50)	NO	Unique role identifier (ADMIN, BANKER, AUDITOR)
name	VARCHAR(100)	NO	Display name
description	TEXT	YES	Role description
permissions	JSON	YES	Array of permission strings
is_system	BOOLEAN	NO	TRUE if system-defined role
created_at	TIMESTAMP	NO	Creation timestamp

Column	Type	Nullable	Description
updated_at	TIMESTAMP	NO	Last update timestamp

Business Meaning: Controls what actions users can perform. System roles cannot be deleted.

Sample Data:

```
('ADMIN', 'Administrator', '["*"]', TRUE)
('BANKER', 'Banker', '["transactions:*"]', TRUE)
('AUDITOR', 'Auditor', '["read:*"]', TRUE)
```

3.2 users

Purpose: Internal staff accounts (bankers, admins, auditors).

Column	Type	Nullable	Description
id	BIGINT	NO	Primary Key
email	VARCHAR(255)	NO	Unique login email
password_hash	VARCHAR(255)	NO	bcrypt hashed password
first_name	VARCHAR(100)	NO	First name
last_name	VARCHAR(100)	NO	Last name
role_id	BIGINT	NO	FK → roles.id
status	VARCHAR(20)	NO	ACTIVE, INACTIVE, SUSPENDED
last_login_at	TIMESTAMP	YES	Last successful login
created_at	TIMESTAMP	NO	Account creation time
updated_at	TIMESTAMP	NO	Last modification

Foreign Keys:

- `role_id` → `roles(id)` (logical, enforced in application)

Business Meaning: Staff who operate the banking system. Not to be confused with customers.

3.3 customers

Purpose: Bank customers who can hold accounts and perform transactions.

Column	Type	Nullable	Description
id	BIGINT	NO	Primary Key
customer_number	VARCHAR(20)	NO	Unique business identifier (CUS-0001)
email	VARCHAR(255)	NO	Login email
password_hash	VARCHAR(255)	NO	bcrypt password
first_name	VARCHAR(100)	NO	First name
last_name	VARCHAR(100)	NO	Last name
phone	VARCHAR(20)	YES	Contact phone
national_id	VARCHAR(50)	YES	Government ID number
date_of_birth	DATE	YES	DOB for verification
address_line1	VARCHAR(255)	YES	Street address
address_line2	VARCHAR(255)	YES	Additional address
city	VARCHAR(100)	YES	City
postal_code	VARCHAR(20)	YES	ZIP/Postal code
country	VARCHAR(50)	YES	Country
status	VARCHAR(20)	NO	PENDING, ACTIVE, SUSPENDED, CLOSED
kyc_status	VARCHAR(20)	NO	PENDING, VERIFIED, REJECTED
kyc_version	INT UNSIGNED	NO	KYC document version
created_by	BIGINT	YES	FK → users.id (banker who onboarded)
created_at	TIMESTAMP	NO	Registration time
updated_at	TIMESTAMP	NO	Last update

Foreign Keys:

- `created_by` → `users(id)` (the banker who created the customer)

Business Meaning: External individuals who use banking services. Must complete KYC before transacting.

3.4 account_types

Purpose: Classification of bank accounts.

Column	Type	Nullable	Description
id	BIGINT	NO	Primary Key
code	VARCHAR(50)	NO	Type code (SAVINGS, CHECKING, FIXED, INTERNAL)
name	VARCHAR(100)	NO	Display name
description	TEXT	YES	Description
min_balance	DECIMAL(18,4)	NO	Minimum required balance
is_active	BOOLEAN	NO	Whether new accounts can be opened
created_at	TIMESTAMP	NO	Creation time
updated_at	TIMESTAMP	NO	Last update

Business Meaning: Defines account products offered by the bank.

Sample Data:

Code	Name	Min Balance
SAVINGS	Savings Account	500.00 BDT
CHECKING	Checking Account	0.00 BDT
FIXED	Fixed Deposit	10,000.00 BDT
INTERNAL	Internal Account	0.00 BDT

3.5 accounts

Purpose: Individual bank accounts owned by customers.

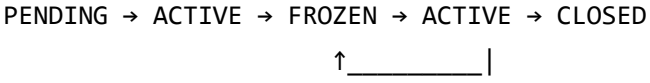
Column	Type	Nullable	Description
id	BIGINT	NO	Primary Key
account_number	VARCHAR(20)	NO	Unique account number (1001-0001-0001)
customer_id	BIGINT	NO	FK → customers.id
account_type_id	BIGINT	NO	FK → account_types.id
status	VARCHAR(20)	NO	PENDING, ACTIVE, FROZEN, CLOSED
opened_at	TIMESTAMP	YES	When account became ACTIVE
closed_at	TIMESTAMP	YES	When account was CLOSED
created_by	BIGINT	YES	FK → users.id (banker)
balance_locked	BOOLEAN	NO	Lock flag for admin operations
currency	VARCHAR(3)	NO	Currency code (BDT)
created_at	TIMESTAMP	NO	Record creation
updated_at	TIMESTAMP	NO	Last update

Foreign Keys:

- `customer_id` → `customers(id)`
- `account_type_id` → `account_types(id)`
- `created_by` → `users(id)`

Business Meaning: The core entity representing a customer's bank account.

Status Transitions:



3.6 account_balances

Purpose: Cached current balance for each account (performance optimization).

Column	Type	Nullable	Description
account_id	BIGINT	NO	Primary Key, FK → accounts.id
available_balance	DECIMAL(18,4)	NO	Spendable balance
pending_balance	DECIMAL(18,4)	NO	Pending transactions
hold_balance	DECIMAL(18,4)	NO	Held/reserved funds
currency	VARCHAR(3)	NO	Currency code
last_transaction_id	BIGINT	YES	FK → transactions.id
last_calculated_at	TIMESTAMP	YES	When balance was last computed
version	INT	NO	Optimistic locking version
updated_at	TIMESTAMP	NO	Last update

Foreign Keys:

- `account_id` → `accounts(id)` (1:1 relationship)
- `last_transaction_id` → `transactions(id)`

Business Meaning: Provides O(1) balance lookups instead of summing all ledger entries.

Optimistic Locking:

```
UPDATE account_balances
SET available_balance = ?, version = version + 1
WHERE account_id = ? AND version = ?
```

3.7 transaction_types

Purpose: Classification of financial transactions.

Column	Type	Nullable	Description
id	BIGINT	NO	Primary Key
code	VARCHAR(50)	NO	Type code (TRANSFER, DEPOSIT, WITHDRAWAL)
name	VARCHAR(100)	NO	Display name
description	TEXT	YES	Description
requires_approval	BOOLEAN	NO	Whether manager approval needed
created_at	TIMESTAMP	NO	Creation time
updated_at	TIMESTAMP	NO	Last update

Business Meaning: Categorizes transactions for reporting and workflow control.

3.8 transactions

Purpose: Header record for each financial transaction.

Column	Type	Nullable	Description
id	BIGINT	NO	Primary Key
transaction_reference	VARCHAR(50)	NO	Unique UUID reference
transaction_type_id	BIGINT	NO	FK → transaction_types.id
amount	DECIMAL(18,4)	NO	Transaction amount
currency	VARCHAR(3)	NO	Currency code
description	VARCHAR(255)	YES	User-provided description
status	VARCHAR(20)	NO	PENDING, COMPLETED, FAILED, REVERSED
source_account_id	BIGINT	YES	FK → accounts.id (sender)
destination_account_id	BIGINT	YES	FK → accounts.id (receiver)
processed_at	TIMESTAMP	YES	When transaction completed

Column	Type	Nullable	Description
created_by	BIGINT	YES	FK → users.id or customers.id
created_at	TIMESTAMP	NO	Creation time
updated_at	TIMESTAMP	NO	Last update

Foreign Keys:

- `transaction_type_id` → `transaction_types(id)`
- `source_account_id` → `accounts(id)` (nullable for deposits)
- `destination_account_id` → `accounts(id)` (nullable for withdrawals)

Business Meaning: The master record of a money movement. Each transaction generates 2 ledger entries (DEBIT and CREDIT).

3.9 ledger_entries

Purpose: Individual debit/credit entries implementing double-entry accounting.

Column	Type	Nullable	Description
id	BIGINT	NO	Primary Key
transaction_id	BIGINT	NO	FK → transactions.id
account_id	BIGINT	NO	FK → accounts.id
entry_type	VARCHAR(10)	NO	DEBIT or CREDIT
amount	DECIMAL(18,4)	NO	Entry amount (always positive)
currency	VARCHAR(3)	NO	Currency code
balance_after	DECIMAL(18,4)	NO	Account balance after this entry
description	VARCHAR(255)	YES	Entry description
entry_date	DATE	NO	Accounting date
created_at	TIMESTAMP	NO	Creation time

Foreign Keys:

- `transaction_id` → `transactions(id)`
- `account_id` → `accounts(id)`

Business Meaning: The immutable record of account movements. NEVER updated or deleted.

Double-Entry Rule:

```
For every transaction:
  SUM(DEBIT amounts) = SUM(CREDIT amounts)
```

3.10 transaction_audit

Purpose: Immutable copy of ledger entries for regulatory compliance.

Column	Type	Nullable	Description
id	BIGINT	NO	Primary Key
ledger_entry_id	BIGINT	NO	FK → ledger_entries.id
transaction_id	BIGINT	NO	FK → transactions.id
account_id	BIGINT	NO	FK → accounts.id
entry_type	VARCHAR(10)	NO	DEBIT or CREDIT
amount	DECIMAL(18,4)	NO	Amount
balance_after	DECIMAL(18,4)	NO	Balance after entry
audit_timestamp	TIMESTAMP	NO	When audit record created

Foreign Keys:

- `ledger_entry_id` → `ledger_entries(id)`

Business Meaning: Provides a separate audit trail that can be verified against ledger_entries.

3.11 audit_logs

Purpose: System-wide activity logging for non-financial events.

Column	Type	Nullable	Description
id	BIGINT	NO	Primary Key
actor_id	BIGINT	YES	User/Customer who performed action
actor_type	VARCHAR(20)	NO	'user', 'customer', 'system'
actor_role	VARCHAR(50)	YES	Role at time of action
action_type	VARCHAR(50)	NO	ACCOUNT_CREATED, LOGIN, PASSWORD_CHANGED, etc.
entity_type	VARCHAR(50)	NO	ACCOUNT, CUSTOMER, USER, SESSION
entity_id	BIGINT	YES	ID of affected entity
before_state	JSON	YES	Entity state before change
after_state	JSON	YES	Entity state after change
metadata	JSON	YES	Additional context (IP, user agent)
created_at	TIMESTAMP	NO	Event timestamp

Indexes:

- `idx_audit_actor (actor_id, actor_type)`
- `idx_audit_entity (entity_type, entity_id)`
- `idx_audit_action (action_type)`
- `idx_audit_created (created_at)`

Business Meaning: Captures all significant system events for compliance and debugging.

3.12 system_config

Purpose: Runtime configuration parameters.

Column	Type	Nullable	Description
id	BIGINT	NO	Primary Key
config_key	VARCHAR(100)	NO	Unique configuration key
config_value	TEXT	YES	Configuration value
value_type	VARCHAR(20)	NO	STRING, NUMBER, BOOLEAN, JSON
description	TEXT	YES	Human-readable description
is_sensitive	BOOLEAN	NO	If TRUE, value hidden in UI
created_at	TIMESTAMP	NO	Creation time
updated_at	TIMESTAMP	NO	Last update

Unique Constraint: `config_key`

Sample Data:

```
('teller.cash_account_number', 'BANK-CASH-001', 'STRING')
```

3.13 idempotency_keys

Purpose: Prevents duplicate transaction processing.

Column	Type	Nullable	Description
idempotency_key	VARCHAR(64)	NO	Primary Key (UUID)
request_hash	VARCHAR(64)	YES	Hash of request body
response_status	INT	YES	HTTP status of original response
response_body	JSON	YES	Cached response
created_at	TIMESTAMP	NO	When key was created
expires_at	TIMESTAMP	YES	When key can be reused

Business Meaning: If a client retries a request with the same idempotency key, the system returns the cached response instead of executing again.

3.14 events

Purpose: Event store for event sourcing pattern.

Column	Type	Nullable	Description
id	BIGINT	NO	Primary Key
event_type	VARCHAR(100)	NO	TRANSFER_COMPLETED, DEPOSIT_COMPLETED, etc.
aggregate_type	VARCHAR(50)	NO	TRANSACTION, ACCOUNT, CUSTOMER
aggregate_id	BIGINT	NO	ID of the affected aggregate
payload	JSON	YES	Event data
created_at	TIMESTAMP	NO	Event timestamp

Business Meaning: Immutable log of all business events. Can be used to rebuild state.

3.15 outbox

Purpose: Outbox pattern for reliable event publishing.

Column	Type	Nullable	Description
id	BIGINT	NO	Primary Key
event_type	VARCHAR(100)	NO	Event type
aggregate_type	VARCHAR(50)	NO	Aggregate type
aggregate_id	BIGINT	NO	Aggregate ID
payload	JSON	YES	Event payload
status	VARCHAR(20)	NO	PENDING, PUBLISHED, FAILED

Column	Type	Nullable	Description
created_at	TIMESTAMP	NO	Creation time

Business Meaning: Events are written here within the same transaction as business data. A separate process reads and publishes them, ensuring exactly-once delivery.

4. Relationship Descriptions

4.1 roles → users (1:N)

Property	Value
Participating Entities	roles, users
Cardinality	One role has many users
Optionality	Role required for user (mandatory)
Enforced Via	`users.role_id` column
Business Interpretation	Each staff member is assigned exactly one role

4.2 customers → accounts (1:N)

Property	Value
Participating Entities	customers, accounts
Cardinality	One customer can have many accounts
Optionality	Customer required for account (mandatory)
Enforced Via	`accounts.customer_id` column
Business Interpretation	Bank customers may hold multiple accounts (savings, checking, etc.)

4.3 account_types → accounts (1:N)

Property	Value
Participating Entities	account_types, accounts
Cardinality	One type classifies many accounts
Optionality	Type required for account (mandatory)
Enforced Via	`accounts.account_type_id` column
Business Interpretation	Each account is categorized by its product type

4.4 accounts → account_balances (1:1)

Property	Value
Participating Entities	accounts, account_balances
Cardinality	One account has exactly one balance record
Optionality	Balance required for active account
Enforced Via	`account_balances.account_id` is PK
Business Interpretation	Cached balance for performance

4.5 transactions → ledger_entries (1:N)

Property	Value
Participating Entities	transactions, ledger_entries
Cardinality	One transaction creates 2+ ledger entries
Optionality	Entries required for completed transaction
Enforced Via	`ledger_entries.transaction_id` column
Business Interpretation	Double-entry accounting requires balanced pairs

5. Constraints & Integrity Rules

5.1 Primary Key Constraints

All tables use `BIGINT AUTO_INCREMENT` primary keys except:

- `account_balances` uses `account_id` as PK (1:1 with accounts)
- `idempotency_keys` uses `idempotency_key` VARCHAR as PK

5.2 Unique Constraints

Table	Column(s)	Purpose
users	email	Prevent duplicate logins
customers	email	Prevent duplicate logins
customers	customer_number	Business identifier
accounts	account_number	Account identifier
system_config	config_key	Configuration lookup

5.3 Check Constraints (Application-Enforced)

Rule	Enforcement
`amount > 0`	Stored procedure validation
`available_balance >= 0`	Stored procedure validation
`ledger_entries` are append-only	No UPDATE/DELETE allowed
`DEBIT sum = CREDIT sum`	Stored procedure logic

5.4 Referential Integrity

Foreign keys are enforced at the **application level** (not database level) for flexibility:

```
ON DELETE: Application prevents deletion of referenced records
ON UPDATE: Cascaded via application logic
```

6. Normalization Justification

6.1 First Normal Form (1NF) ✅

- All tables have atomic column values
- No repeating groups
- Each row uniquely identified by primary key

6.2 Second Normal Form (2NF) ✅

- All non-key attributes depend on the entire primary key
- No partial dependencies (all tables have single-column PKs)

6.3 Third Normal Form (3NF) ✅

- No transitive dependencies
- Example: `customers.city` doesn't depend on `postal_code`

6.4 Denormalization Decisions

Table	Denormalization	Justification
account_balances	Cached balance	O(1) lookups vs. summing ledger
ledger_entries	balance_after	Avoid recalculating historical balances
transactions	source/dest account IDs	Direct access without ledger join

7. DBMS Alignment with Banking Requirements

7.1 ACID Compliance

Property	Implementation
Atomicity	Stored procedures with START TRANSACTION / COMMIT
Consistency	Application-level constraints + validation
Isolation	SELECT ... FOR UPDATE row locking
Durability	InnoDB with write-ahead logging

7.2 Financial Precision

```
amount DECIMAL(18,4) NOT NULL
```

- 18 digits total, 4 decimal places
- Prevents floating-point rounding errors
- Supports values up to 99,999,999,999,999.9999

7.3 Audit Compliance

- `transaction_audit` provides immutable financial log
- `audit_logs` captures all system events
- `before_state` / `after_state` JSON for change tracking

7.4 Concurrency Control

```
-- Optimistic locking
SELECT ... FOR UPDATE;
UPDATE account_balances SET version = version + 1 WHERE version = ?;
```

8. Summary

Table Count by Category

Category	Tables	Count
Identity & Access	roles, users, customers	3
Account Management	accounts, account_types, account_balances	3
Financial Core	transactions, transaction_types, ledger_entries	3
Audit & Compliance	transaction_audit, audit_logs	2
System Infrastructure	system_config, events, outbox, idempotency_keys	4
Total		**15**

Key Design Highlights

1. **Double-Entry Accounting** via ledger_entries (DEBIT/CREDIT pairs)
2. **Append-Only Ledger** for financial integrity
3. **Optimistic Locking** via version columns
4. **Idempotency** for safe transaction retries
5. **Event Sourcing** via events table
6. **Outbox Pattern** for reliable messaging

This documentation is suitable for DBMS coursework, viva examinations, and project reports.