📘 BANKING SYSTEM - AUDIT LOG & SECURITY MODEL

This document explains the logging mechanisms and security enforcement strategies used in the SQL-based banking system.

✅ SECTION A: AUDIT LOGGING MECHANISM

An `AuditLog` table is maintained to track all major user/system actions across the database. This promotes:

- Transparency

- Debuggability

- Compliance & rollback tracking

🔧 Table: AuditLog

Column	Data Type	Description
LogID	INT (PK)	Unique identifier for each log entry
ActionPerformed	VARCHAR(100)	Type of action (e.g., Create, Update, Transfer)
PerformedBy	VARCHAR(50)	User or process performing the action
Details	VARCHAR(255)	Additional context (Account ID, Customer, etc.)
ActionTime	DATETIME	Timestamp of when the action occurred

🧠 Where it's used:

Procedure	What Is Logged
CreateCustomer	Customer creation action
OpenAccount	Account creation for a customer

DepositMoney	Deposit action with account ID
WithdrawMoney	Withdrawal action and amount
TransferMoney	Both from and to account transactions
ApplyForLoan	Loan request submitted
FreezeAccount / UnfreezeAccount	Account status change log

🔁 How:

Every stored procedure includes an `INSERT INTO AuditLog(...)` line after successful execution.

🔐 SECTION B: SECURITY ENFORCEMENT MODEL

Security is enforced through a combination of:

1. Account Freezing (Soft Lock)

- Business rule: Frozen accounts **cannot send or receive funds**

- Controlled via: `Status` column in `Accounts` table

- Tracked in: `AccountFreezeRecords` table

🔧 Table: AccountFreezeRecords

Column	Data Type	Description
FreezeID	INT (PK)	Unique freeze record ID
AccountID	INT (FK)	Account that is frozen
Reason	VARCHAR(255)	Reason for freezing
FrozenAt	DATETIME	Time the account was frozen
UnfrozenAt	DATETIME (NULLABLE)	Time it was unfrozen (if applicable)

💡 **Notes:**

Procedure	Check	Action Taken
DepositMoney	Reject if account is frozen	Error raised
WithdrawMoney	Reject if account is frozen	Error raised
TransferMoney	Reject if either account is frozen	Error raised

2. Input Validations

All stored procedures include strict validations:

- No duplicate emails or IDs

- No overdrafts or zero transfers

- Account status checks before any transaction

- Foreign key validations (e.g., account must exist)

3. UserRoles Table (Optional - For future enhancement)

Designed to support admin-level access controls in the future.

🔧 **Table: UserRoles**

Column	Data Type	Description
UserID	INT (PK)	Unique system user ID
Username	VARCHAR(50)	Username of the account
Role	VARCHAR(20)	Role type: Admin, Manager, Customer, etc.

Usage ideas:

- Restrict procedures using `IF @Role = 'Admin' THEN ...`

- Future RBAC (Role-Based Access Control) expansion

🛡️ **Summary:**

The system is designed with security-first principles:

- Every action is traceable

- Unauthorized operations are blocked
- Logs help maintain accountability

These make the system enterprise-ready and scalable for large deployments.