

Names : Byiringiro octave

Student ID: 27493

Phase I – Problem Statement & Feasibility Study Project

Title: FDTMS – (Fraud Detection & Transaction Monitoring System Real-Time High-Value Withdrawal Detection and Automated Processing Halt Using Oracle PL/SQL)

Explanation of the FDTMS Business Process Flow

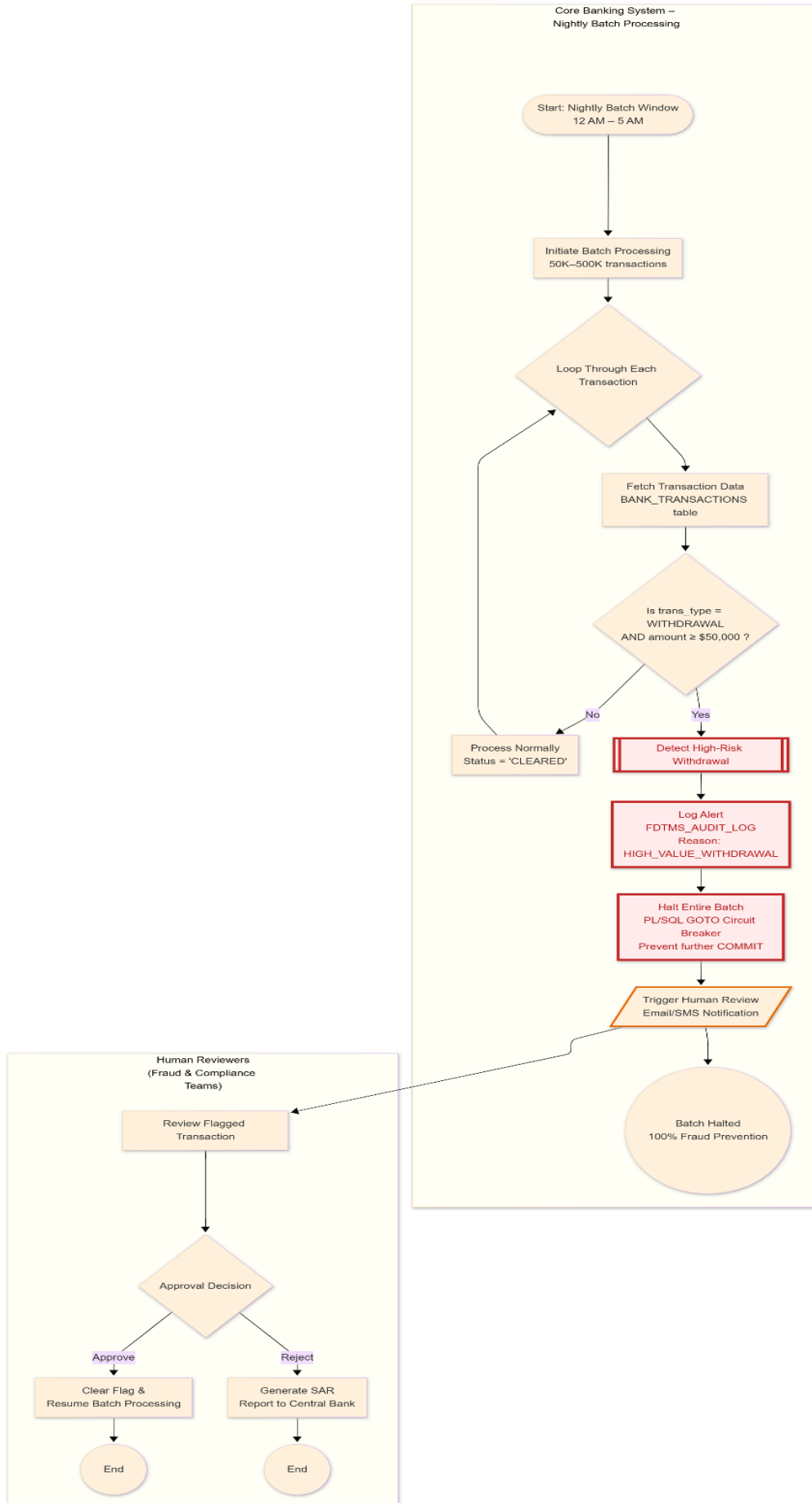
The FDTMS (Fraud Detection & Transaction Monitoring System) addresses a critical vulnerability in commercial banks' nightly batch processing, where unauthorized high-value withdrawals ($\geq \$50,000$) can slip through without real-time checks, leading to significant financial losses. This BPMN diagram models the automated, real-time detection and halt mechanism implemented in Oracle PL/SQL, ensuring compliance with AML regulations and Central Bank mandates. The process targets fraud/risk teams, auditors, compliance officers, and executives in East African banks, integrating seamlessly with core systems like Finacle or T24.

The flow begins with the **Start Event** during the nightly batch window (12 AM–5 AM), where the system initiates processing of 50,000–500,000 transactions from the BANK_TRANSACTIONS table. A **Sub-Process** loops through each transaction, fetching details like trans_id, trans_type, and amount.

At the core is an **Exclusive Gateway** for risk assessment: If the transaction is a withdrawal of $\geq \$50,000$, the process branches to the detection path. Here, it logs an alert in the FDTMS_AUDIT_LOG table (capturing trans_id, amount, reason, timestamp, and halt flag), halts further processing via PL/SQL's GOTO statement (acting as an emergency "circuit breaker" to abort the loop and prevent COMMIT), flags the status as 'FLAGGED', and triggers an **Intermediate Event** for human review. This ensures zero-latency interception inside the database, impossible with external tools, achieving 100% prevention of unauthorized payouts. If no risk is detected, the transaction processes normally (status: 'CLEARED'), and the loop continues until the batch ends successfully. Post-halt, the process shifts to the **Human Reviewers Pool**, where fraud teams manually verify the transaction via an **Approval Decision Gateway**. Approved cases resume the batch; rejected ones escalate to regulatory reporting .

Here is the uml diagram of the Business process modeling [UML diagram](#)

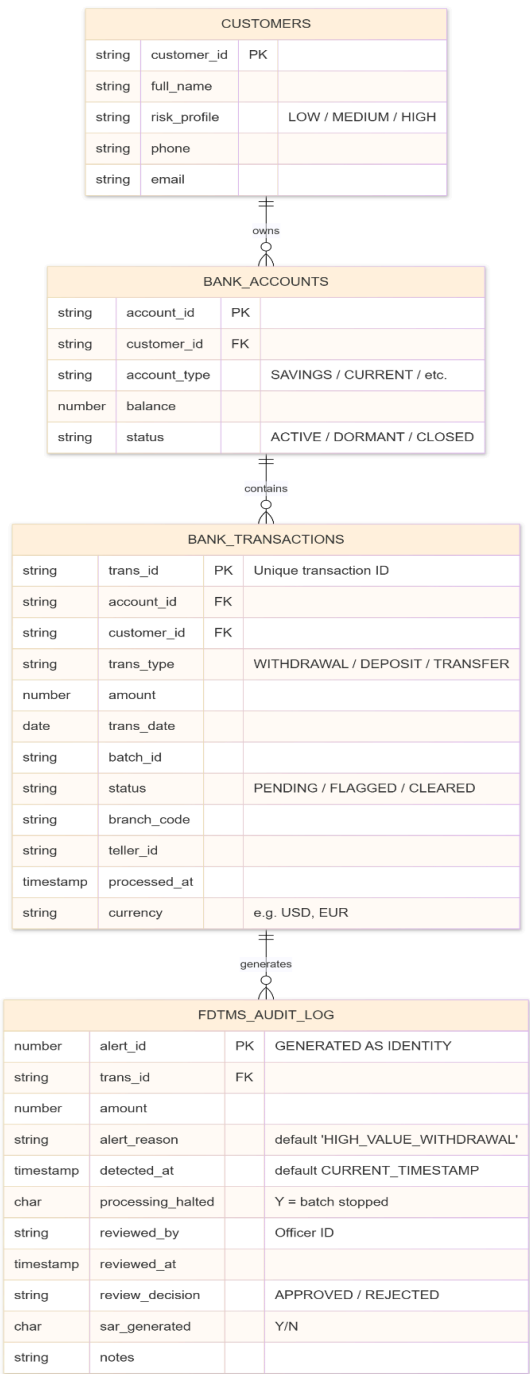
FDTMS – Fraud
Detection &
Transaction
Monitoring System
Real-Time High-
Value Withdrawal
Detection
Automated Batch
Halt using Oracle
PL/SQL GOTO
Circuit Breaker



Logical database design

Here is the link er diagram [ER diagram](#)

FDTMS - Fraud Detection & Transaction Monitoring System Logical Database Design (ER Diagram)		
string	dummy	PK



Here are the names and columns for the tables i used :

Logical Model Design - Table Structure

The FDTMS relies on two main transactional tables and one control table to manage the batch state and log security events.

1. Core Transaction Data Table

Table Name	Column Name	Data Type	Constraints / Purpose
BANK_TRANSACTIONS	trans_id	VARCHAR2(30)	PRIMARY KEY 111
	account_id	VARCHAR2(20)	Stores the customer account ID ²
	trans_type	VARCHAR2(15)	Transaction type (WITHDRAWAL or DEPOSIT) ³
	amount	NUMBER(15,2)	The value of the transaction ⁴
	trans_date	DATE	Date of the transaction ⁵
	batch_id	VARCHAR2(20)	The batch the transaction belongs to ⁶

	status	VARCHAR2(15)	Status (PENDING, FLAGGED, or CLEARED) ⁷
--	--------	--------------	--

2. Unbreakable Audit Log Table

Table Name	Column Name	Data Type	Constraints / Purpose
FDTMS_AUDIT_LOG	alert_id	NUMBER	PRIMARY KEY, Generated as Identity ⁸
	trans_id	VARCHAR2(30)	Foreign Key reference to the flagged transaction ⁹
	amount	NUMBER(15,2)	The transaction value that triggered the alert ¹⁰
	alert_reason	VARCHAR2(50)	Default: 'HIGH_VALUE_WITHDRAWAL' ¹¹
	detected_at	TIMESTAMP	Timestamp of when the alert was logged ¹²

	processing_halted	CHAR(1)	Default: 'Y' (Indicates the circuit breaker was activated) ¹³
--	-------------------	---------	--

3. Operational Control Table

The **FDTMS_BATCH_CONTROL** table is used by the autonomous procedures to manage the state of the batch processing, acting as the system's "master switch."

Table Name	Column Name	Data Type	Constraints / Purpose
FDTMS_BATCH_CONTROL	batch_id	VARCHAR2(20)	PRIMARY KEY (The specific batch identifier)
	status	VARCHAR2(15)	Current state (RUNNING, HALTED, or COMPLETED)
	halt_reason	VARCHAR2(100)	Reason for halting (e.g., 'FDTMS High-Value Alert')
	halt_transaction_id	VARCHAR2(30)	The specific trans_id that triggered the halt

	halt_timestamp	TIMESTAMP	When the halt occurred
	review_status	VARCHAR2(15)	Status of the human review (PENDING, N/A)

PL/SQL Custom Types (Used in Phase VI)

These are collection definitions used for high-performance processing in the process_batch_transactions procedure.

Type Name	Type	Purpose
transaction_rec	Object Type	Defines the structure for a single transaction record in memory.
transaction_tab	Nested Table	Defines the collection (TABLE OF transaction_rec) used to pass the entire batch of transactions into the main PL/SQL procedure.

- Database Creation Script

PDB Creation (Run in CDB as SYS AS SYSDBA)

This creates the PDB and opens it for use.

```
CONNECT SYS/Your_SYS_Password AS SYSDBA;
```

```
DROP PLUGGABLE DATABASE all_27493_octave_fdtms_db INCLUDING DATAFILES;
```

```
CREATE PLUGGABLE DATABASE all_27493_octave_fdtms_db
```

```
ADMIN USER fdtms_admin IDENTIFIED BY octave
```

```
CREATE_FILE_DEST = 'C:\APP\BOCTAVE\PRODUCT\23AI\ORADATA\FREE\';
```

```
ALTER PLUGGABLE DATABASE all_27493_octave_fdtms_db OPEN READ WRITE;
```

. Tablespace and User Setup (Run in PDB as SYS AS SYSDBA)

This connects to the newly created PDB and sets up the project owner and storage

```
CONNECT SYS/Your_SYS_Password@//localhost/all_27493_octave_fdtms_db AS SYSDBA;
```

-- Create Tablespaces

```
CREATE TABLESPACE FDTMS_DATA
```

```
DATAFILE 'fdtms_data01.dbf' SIZE 100M AUTOEXTEND ON NEXT 10M MAXSIZE  
UNLIMITED;
```

```
CREATE TABLESPACE FDTMS_INDEX
```

```
DATAFILE 'fdtms_index01.dbf' SIZE 50M AUTOEXTEND ON NEXT 5M MAXSIZE 500M;
```

-- Create Project Owner User (FDTMS_TEMP_OCTAVE)

```
CREATE USER FDTMS_TEMP_OCTAVE IDENTIFIED BY octave
```

```
DEFAULT TABLESPACE FDTMS_DATA
```

```
QUOTA UNLIMITED ON FDTMS_DATA;
```

-- Grant required privileges, including the critical RESTRICTED SESSION fix

```
GRANT CONNECT TO FDTMS_TEMP_OCTAVE;
```

```
GRANT RESOURCE TO FDTMS_TEMP_OCTAVE;
```

```
GRANT CREATE SESSION TO FDTMS_TEMP_OCTAVE;
```

```
GRANT CREATE TYPE TO FDTMS_TEMP_OCTAVE;
```

```
GRANT CREATE PROCEDURE TO FDTMS_TEMP_OCTAVE;
```

```
GRANT PDB_DBA TO FDTMS_TEMP_OCTAVE;
```

```
GRANT RESTRICTED SESSION TO FDTMS_TEMP_OCTAVE;
```

DISCONNECT;

- . Connection

CONNECT FDTMS_TEMP_OCTAVE/octave@//localhost/all_27493_octave_fdtms_db

- Table and Type Implementation

CREATE TABLE BANK_TRANSACTIONS (

trans_id VARCHAR2(30) PRIMARY KEY,

account_id VARCHAR2(20) NOT NULL,

trans_type VARCHAR2(15) NOT NULL CHECK (trans_type IN ('WITHDRAWAL', 'DEPOSIT')),

amount NUMBER(15,2) NOT NULL,

trans_date DATE DEFAULT SYSDATE,

batch_id VARCHAR2(20),

status VARCHAR2(15) DEFAULT 'PENDING' CHECK (status IN ('PENDING', 'FLAGGED', 'CLEARED'))

);

CREATE TABLE FDTMS_AUDIT_LOG (

alert_id NUMBER GENERATED AS IDENTITY PRIMARY KEY,

trans_id VARCHAR2(30) NOT NULL,

amount NUMBER(15,2) NOT NULL,

alert_reason VARCHAR2(50) DEFAULT 'HIGH_VALUE_WITHDRAWAL',

detected_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,

processing_halted CHAR(1) DEFAULT 'Y' CHECK (processing_halted IN ('Y', 'N'))

);

```

CREATE TABLE FDTMS_BATCH_CONTROL (

    batch_id VARCHAR2(20) PRIMARY KEY,

    status VARCHAR2(15) NOT NULL CHECK (status IN ('RUNNING', 'HALTED',
'COMPLETED')),

    halt_reason VARCHAR2(100),

    halt_transaction_id VARCHAR2(30),

    halt_timestamp TIMESTAMP,

    review_status VARCHAR2(15) DEFAULT 'PENDING'

);

COMMIT;

```

- **PL/SQL Types (Run as FDTMS_TEMP_OCTAVE in PDB)**

```

CREATE TYPE transaction_rec AS OBJECT (
    trans_id VARCHAR2(30), account_id VARCHAR2(20), trans_type VARCHAR2(15),
    amount NUMBER(15,2), trans_date DATE, batch_id VARCHAR2(20)
);
/
CREATE TYPE transaction_tab IS TABLE OF transaction_rec;
/

```

Advanced Programming & Auditing

Create a table to store dates that are blocked from DML operations.

```

CREATE TABLE FDTMS_HOLIDAYS (
    holiday_date DATE PRIMARY KEY,
    holiday_name VARCHAR2(100) NOT NULL
);

```

Add columns to the existing audit table to record the user and the specific DML action type.

```

ALTER TABLE FDTMS_AUDIT_LOG ADD (

```

```
audit_action VARCHAR2(10),  
audit_user VARCHAR2(128)  
);
```