# MICRO DATA MODEL ARCHITECTURE FOR AML SCORING RULE ENGINES

Walakulu Arachchige Hasitha Maduranga

(199344L)

MSc in Computer Science

Department of Computer Science and Engineering

University of Moratuwa
Sri Lanka

June 2022

# MICRO DATA MODEL ARCHITECTURE FOR AML SCORING RULE ENGINES

Walakulu Arachchige Hasitha Maduranga

(199344L)

Thesis submitted in partial fulfillment of the requirements for the degree of
MSc in Computer Science Specializing in Software Architecture

Department of Computer Science and Engineering

University of Moratuwa
Sri Lanka

June 2022

## Declaration

I declare that this is my own work and this thesis/dissertation does not incorporate without acknowledgement any material previously submitted for a Degree or Diploma in any other University or institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Also, I hereby grant to University of Moratuwa the non-exclusive right to reproduce and distribute my thesis/dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature : ---------- Date: ----------

The above candidate has carried out research for the Masters thesis/ Dissertation under my supervision.

Name of the supervisor : Eng. Prof. Indika Perera

Signature of the supervisor : ---------- Date: ----------

# Abstract

Online and mobile banking have become a primary service of today's banking and financial sector. Clients could do their primary transactional jobs without physically appearing on the bank. This facility is 24x7 available. So, detection of money laundering activities based on transactional data analysis is a key challengeable area in today's banking and financial sector.

Businesses are trying to prevent money laundering activities by applying rule-based techniques to the real time operational transactions which could not completely cure the problem because higher constraints on the operational transaction could inconvenience the legal customer base and lose the customer satisfaction over the time. So, the near-real time and traditional data warehousing approaches with post detection techniques becomes the most common approach to detect money laundering activities in today's banking and financial context.

Traditional data warehousing approaches loaded data from operational or transactional systems on a weekly or nightly basis. Near real-time and real-time data warehouse approaches use real-time ETL tools to load data into the data warehouse in predefined shorter time intervals which preserve a gap with real-time transactional data. In addition to that, running anomaly detection engines (rule based or machine learning models) on top of those massive amounts of data (either OLTP databases or warehouse database) will take another considerable time due to higher velocity of data. So, identifying money launderers by analyzing post detection techniques causes higher risk to the financial system because the money launderer may leave the financial system before the money launderer catches.

This report introduce a novel **data modelling architecture** named "Micro Data Model Architecture" and an associated supporting tool named "Micro Temporal Database Generator" for "scoring rule engines" to detect financial fraudulent activities earlier by removing the burden on operational data sources.

**Acknowledgement**

First, I must thank the Chancellor, Vice-Chancellor, and the rest of the management of the University of Moratuwa for giving me an opportunity to continue my higher studies in Computer Science with the guidance provided by the Department of Computer Science and Engineering University of Moratuwa. And also, thank them for continuously keeping the higher standard of the MSC program by keeping its curriculum up to date.

My deepest gratitude goes to my project supervisor, Prof.Indika Perera, for being interested in my research idea and constant support and guidelines rendered to me, right from the beginning, till the submission of the thesis report.

Further I thanks for the global industrial professionals, Mr.Boris Bialek, former Chief Architect of "Global Head of Banking Platform" in Switzerland (he is currently bare the post: Head of Innovation, EMEA at MongoDB) who shares his ideas and experience on real time transaction processing related to modern technology stack and allocate his valuable time to sharing his technical experiences with me. And, I would thank for management team of Creative Software Pvt.Ltd to giving me flexibility to studying the mastering program while employing. Also, I am grateful to all my teachers and my batch mates in MSC Program who supported and encouraged me throughout the completion of this project.

Finally, I express my deepest appreciation to my dearest wife and parents for their encouragement and support given throughout this complete.

# Table of Contents

## List of Figures



## List of Tables

**List of Abbreviations**

| | |
|---|---|
| AML | Anti-Money Laundering |
| FATF | Financial Action Task Force |
| FCA | Financial Conduct Authority (UK) |
| 5AMLD | Fifth Anti-Money Laundering Directive (European Union) |
| 6AMLD | Sixth Anti-Money Laundering Directive (European Union) |
| ICA | International Compliance Association |
| OLTP | Online Transaction Processing |
| OLAP | Online Analytical Processing |
| IMoLIN | International Money Laundering Layering Network |
| ICA | International Compliance Association |
| CDC | Change Data Capture |
| ACID | Atomicity,Consistency,Isolation,Durability |
| SC | Scoring Rule |

# 1 CHAPTER 1 - INTRODUCTION TO MICRO DATA MODEL ARCHITECTURE FOR AML SCORING RULES ENGINES

## 1.1 Introduction

Businesses are getting more complex today than it has ever been. People are fast moving from old technologies to the newest technologies to face the competitive business environment. Some businesses wrapped their old business inside the newer technology and others came with innovative business ideas capable of providing services to a massive client base which was not expected in the recent decade. Mobile and internet banking sector is one such innovative sector which uses various modern sophisticated technical concepts to facilitate better service for their customers (Ex: provide "mobile transactions" to replace traditional front desk queues in banks).

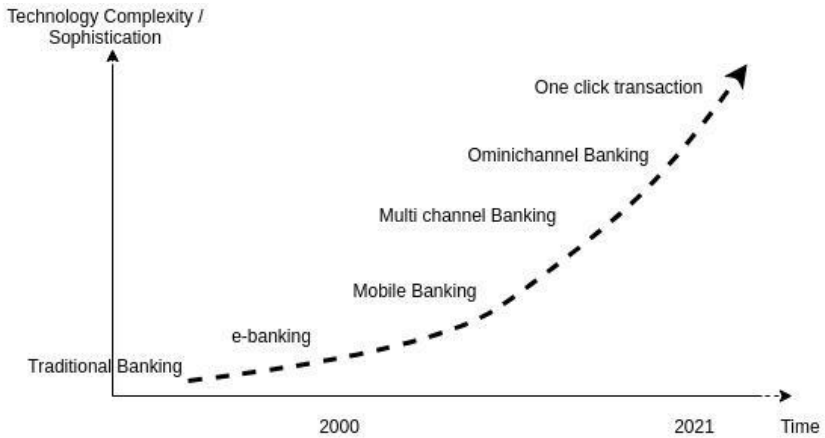


*Figure 1.1: Banking System Evolution Sketch over Time*

Figure 1.1 shows the rough sketch of how banking and financial systems evolve their business & technical concepts over the past decade. However those services do not come for free. It introduces novel risks and challenges to the banking and financial system. Detecting and preventing money laundering activity is one such risky, challenging and must-do operation in today's banks because money launderers could practice their illegal tasks irrespective of their physical locations.

This report focuses on introducing a novel **data modeling architectural solution** to optimize the data retrieval time for anti-money laundering "scoring rule engines" (Figure 2.3) by introducing a data storage architectural model.

## 1.2 Background & Motivation

According to ICA, hundreds of billions of dollars of criminally derived money is laundered through financial institutions, annually. Further they have claimed that, extensive nature of the money laundering process allows criminals to practice these illegal methods in a countless number of diverse ways (Refer Appendix-B for sample money laundering patterns) [3][5][6]. This situation is more complicated when consider the modern online and internet banking solutions which provide thousands of real-time transactions in seconds, irrespective of geographical boundaries.

To face this challenging problem, most of today's banks and financial organizations use two common architectural patterns to prevent & detect money laundering activities happening over their businesses boundaries. They are**, pre-validation techniques** based architectures and **post-detention techniques** based architectures.

In pre-validation techniques, financial institutes use a set of simple business rule sets to validate real-time monetary transactions. It mostly covers the 'real-time' scope of layers of figure 2.3. Those rules are defined by business investigators to match their unique business domain features/requirements and they directly affect the transaction completion time which is highly relates to customer satisfaction of the financial system. So, investigators always try to maintain simplicity & accuracy [7] of business rules to provide better customer satisfaction while baring money risk (specially, to avoid blocking genuine transactions).

Post detection techniques mainly include the near real-time approaches and offline data analysis approaches. Near real time analysis methods compose of both rule based scoring engines & predictive models which get inputs from operational transaction

data sources to predict the suspicious transactions after transaction completes its operational lifecycle. The offline analysis techniques use complex rule based models or data driven models to analyze & identify suspicious transaction records based on offline data sources like data warehouse. Those data sources load operational data on nightly or weekly basis on predefined time intervals. So there is a higher chance for money launderers to leave the financial system before they are caught by offline data analysis techniques.

In order to address the above situation, it is necessary to find solutions to minimize the suspicious activity identification time while keeping system performance at satisfactory level. So **designing a novel architectural solution to solve a real world problem,** makes me highly motivated to start my research in this area.

### 1.3 Problem in Brief

Internet and online banking have become a primary service of today's banking and financial sector. Customers could do their daily transactional activities without physically appearing at the bank front desk. This service supports executing transactions in seconds and it is available 24x7. So, customers are experiencing the ultimate benefit of technology enhancement which resulted from the past few decades.

However there are some negative effects as well. Money launderers use those modern sophisticated financial services to achieve their laundering process than they did previously. So, detecting and preventing money laundering activity has become one of the challenging and must-do operations (most of the governments and unions have been defining rules to follow for banks. Prevention of money laundering Act No 5 of 2006- Sri Lanka [1], FATF-Global, FCA-UK, 5AMLD and 6AMLD in -European Union [2], etc. Appendix-A for more details) in today's banking & financial industry.

Banks & financial institutes use transaction pre-validation techniques & post-detection techniques to address this challengeable issue. Pre-validation techniques use simple validation rules to validate real-time transactions and block illegal transactions before the transaction completes its operational lifecycle. Designing these types of transaction

blocking rules is part of the financial investigators' responsibility and they totally depend on their own unique business domains (Ex: Maximum transferable threshold varies over financial institutes).But they cannot completely address the money laundering issue due to its inherent limitations such as: Difficulty of maintaining transaction blocking rules over evolving customer habits over time, Difficulty of maintaining 100% accuracy to prevent blocking genuine transactions, Higher constraints on operational data sources can cause to degradation of system performance etc.

Post detection techniques are used to identify illegal transactions after a transaction completes its real-time processing cycle. Scoring Rule Engines, Predictive Models and offline analysis are the widely use detection techniques which operates on top of OLTP or data warehouse data sources. Even though these techniques support to address some of the pre-validation technique limitations, they introduce novel challenges to anti-money laundering research domain. They are:

**Challenge 1:** How to write complex transaction analysis queries without adding burden on OLTP systems
**Challenge 2:** How to speed up the suspicious activity detection process (most of the traditional post-detection solutions run on a weekly or monthly basis)
**Challenge 3:** How to explain machine learning models (results of machine learning models are difficult to explain)

This research focuses on addressing above **Challenge 1 & Challenge 2 in "Post Detection Solutions"**. Especially, the report focuses on "**scoring rules**" (Figure 2.3) section related optimizations.

### 1.4 Aims and Objectives

Aim of this research project to improve & optimize existing **near real-time scoring rule-based** monetary fraud detection **processes by introducing novel data modeling**

**architecture and an associated supporting tool to accelerate the data modeling process**.

To achieve that, the below objectives need to be satisfied:

- To evaluate the necessity of optimizing near real-time scoring rule-based engines for modern banking & financial industry
- To design a data modeling architectural approach to speed up the existing scoring rule-based fraud detections
- To implement the proposed approach using existing industrial tools
- To validate the proposed model
- To discuss the strengths and limitations of proposed approach

## 1.5 Proposed Solution

The proposed solution is to introduce novel architecture components to address above challenge 1 & challenge 2 (section 1.3), for anti-money laundering "scoring rule engines" by introducing a novel data modeling architecture.

The main idea of the proposed approach is to define a set of micro temporal databases by specifying **data storage period** and/or **data preferred condition** per each database. The real-time transactional data is expected to asynchronously replicate to each database based on their temporal time period and data preference. The scoring rule-based anomaly detection engines (out of scope) pointed to those temporal databases based on the transaction history record requirement or interest of data (Figure 1.2).

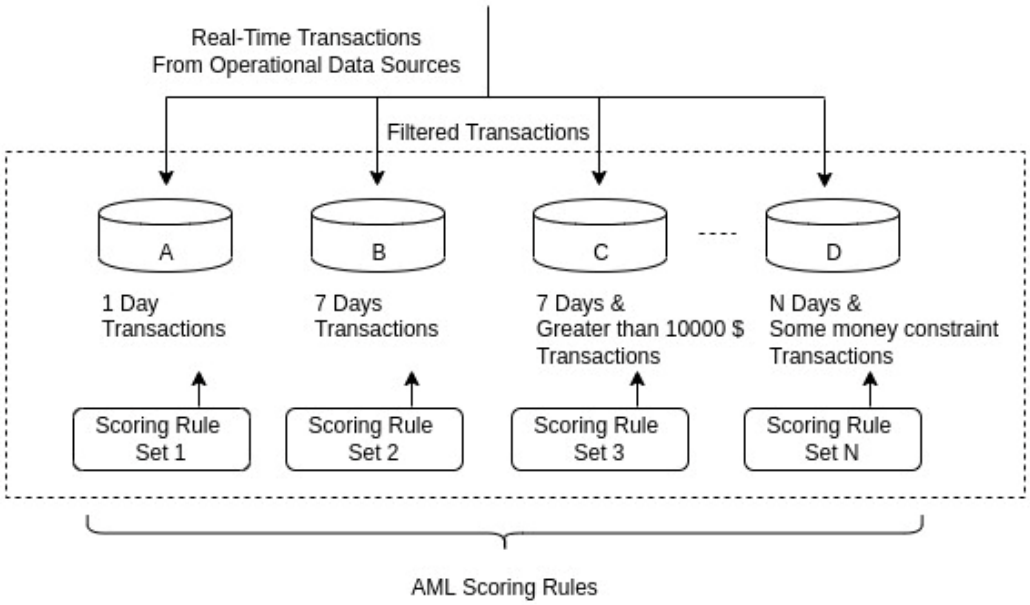


*Figure 1.2: Proposed Solution*

Figure 1.2 illustrates the top-level data modeling architecture. The A, B, C, D database components are known as "micro temporal" databases. They can **act as both temporal & cache databases at the same time**. The operational, real-time transactions are expected to filter and push into these micro databases after they complete the real-time OLTP lifecycle. It could achieve using asynchronous events or implementing change data capture mechanisms along with stream technologies (out of research scope). Scoring rules/engines are expected to point to those micro temporal databases for their near real-time anti-money laundering predictions.

Ex: "*Identify the customers with an unusual number of small amounts of cash-in followed by a cash-out in a short period of time (24 hour periods)*" is a well-known AML scoring rule. The traditional scoring rule base solutions evaluate this type of pattern on a weekly or monthly basis to minimize the OLTP operational burden. Our new approach suggest to create a micro temporal database which capable of storing up to last 24 hour temporal transaction data. Then we expect point the above scoring rule to micro temporal database instead of OLTP database and run the rule daily to identify suspicious activities as early as possible.

This approach helps to address the above (section 1.3) post-money laundering detection challenges in the following manner:

- Domain experts can complex transaction analysis queries on top of micro temporal databases without adding any burden on OLTP systems.

- Scoring rule engines can produce their results using micro temporal databases without waiting till month ends. Hence it reduce money launderer leaving risk from the financial system. Further, a lower number of queries on an individual micro temporal databases & flexibility of query optimization will further support to speedup of query execution for scoring rules engines to fetch necessary information within a shorter time than they traditionally operate on.

## 1.6 Rest of the Report

Second chapter gives in-depth details of the money laundering problem and it will discuss the current solutions implemented in the industry to address the situation. In addition to that, it will discuss the limitations of the current approaches and tools & techniques which can be used for the future improvement of the solution. The third chapter will explain the in-details design of the proposed architectural solution. Further it will discuss the necessary guidelines for users to fallow to get better results of this architectural model. Fourth chapter presents the temporal database modeling & generation tool which capable of generate temporal databases in real-time fashion, based on user inputs. Further it demonstrate the model validation details by generating simulated monetary transactional data using IBM's AmlSim tool. Last chapter will discuss the limitations and further improvement suggestions for the proposed architectural model.

## 1.7 Summary

This chapter introduces the money laundering context and the legal background related to the money laundering process. It overviews the current challenges faced by banking & financial industry in the scope of preventing & detecting money laundering activities

in day to day banking operations. Lastly it overviews the proposed solution to solve the challenges of near scoring rule engines in very briefly.

# 2 CHAPTER 2 - LITERATURE REVIEW

## 2.1 Introduction

Online and mobile banking has become one of the primary functions in today's banking & financial industry. So anti-money laundering has become one of the most active research areas in today's banking & financial sector.

First section of this chapter discusses in depth detail evolution of online and real-time banking facilities in modern banking and financial industry. Then the report explains the evolution of money laundering activities and pass researches conducted to detect and prevent money laundering activities happening over monetary transactional systems. More specially, it describes the current anti-money laundering solutions by clearly categorized in to three main groups namely, pre-validation solutions, post detection solutions and hybrid solutions.

Middle of the report explains the importance of temporal database for anti-money laundering systems while explaining the industrial supportability of different available tools techniques in the today's industry. Later of this section, discuss the database caching mechanisms to provide better performance over large volume of data.

Last section describes the two of the most famous transactional simulation tools which can use to generate simulated transactional data while keeping real world transaction properties. Lastly it evaluates the strength and limitations of each approach.

## 2.2 Banking Operations in the Modern Era

History of banking and the financial system goes to the old centuries of romance. Old days almost all the conventional banking activities were carried out manually. Customers need to physically appear in the bank for their day to day financial needs.

Today, most national and international banks, credit unions and other financial organizations provide some form of online banking facility. It opens a new

marketplace to financial institutes by adding extra value to their current operations [9]. Those services offered under a few subscription fees per customer or large financial service organizations provide those services under zero fees for customers. One of the major reason for that is, cost of bank operation drastically reduced by the online and mobile banking solutions with compared to the traditional banking(reducing burden on front desk counters of the banks) [9][10][11].Global change of traditional banking system started around the year 2007 and most of the significant changes were applied to product and service such as writing checks, paying bills, transferring funds, printing statements, and inquiring about account balances etc.(list of common operations are in the below list).

**Common Online Banking Operations**

- View Summary of customer account and his past transaction history
- Bill pay service
- Transfer money between accounts in the same bank or different banks
- Generate real time digital statements instead of old fashion paper based statements
- Instantly open new accounts
- Online money deposit services
- View images of proceed checks
- Supports for plan budget
- Etc.

## 2.3 Propagation of the Money Laundering Problems

International Compliance Association (ICA) defines money laundering as "Money laundering is the generic term used to describe the process by which criminals disguise the original ownership and control of the proceeds of criminal conduct by making such proceeds appear to have derived from a legitimate source" [3].The laundering process can takes up to three stages namely, **placement**(Criminals pump their illegal money in to the financial system), **layering** (Conceal the source of money by introducing series of transactions -washing), **integration** (Laundered money withdrawn and reintroduced

in to the legitimate economy).This landscape is diverse, and it is not a novel problem to the banking and financial industry. It started from the old bookkeeping systems and propagated to modern sophisticated financial software systems by evolving over the time.

Old days, banks and financial institutes used manual record verification and validation techniques to check whether transactions were in compliance with the rules and regulations which were introduced by the government and financial organization itself. This was carried out by expert human investigators based on their professional history experiences working on similar problem domain. That manual process took a considerable time to verify the given transaction legality and sometimes it took more than a week depending on the amount of workload in the bank. But this manual transaction record analysis time was not a serious issue at that time period because it is harder to execute complex money laundering transactional patterns with physically appearing on the bank front desk queue (Table 2.1).

With the improvement of technology (especially internet), banks and financial institutes tend to facilitate their customers in a finer way than they never had. They have promoted their customers to use online and mobile banking solutions as their newest golden facility which will help to totally remove busy front desk queues in traditional banking systems. Those features are not coming free of issues. One of the most challengeable issue is identifying money laundering transactions which are happening in diverse way over high-speed transactional systems [6][13][14]. This is mainly because customers can do their financial activities irrespective of physical geographical location. The International Money Laundering Information Network (IMoLIN) describes that by mentioning that the increase of sophistication and complexity of technology, badly happen to today's trends in money-laundering techniques [12].

Example Scenario: Splitting a big amount of money into small pieces and sharing them over multiple accounts is a famous money laundering pattern.

*Table 2.1:Money Laundering Scenario Comparison of Traditional Banking vs Modern Banking*

| Approach | System Behavior |
|---|---|
| Traditional Banking | Customers need to fill several slips to introduce money into the financial system. In that case, front desk bank officers could easily identify those tasks since the customer is physically appearing in the bank queue. |
| Online and mobile Banking | Customers do not physically appear in the bank and transaction completion time gets in seconds or milliseconds. So, there is a possibility to spread a big amount over multiple other accounts (by assuming there are no pre-validation conditions to keep system performance at good level) and leave the financial system before manual verifications or execute the post detection techniques. |

## 2.4 Anti-Money Laundering Solutions

The term "Anti-money laundering" (AML) is used to describe the process of preventing, detecting & reporting the money laundering activities happening over their operational business boundaries [4]. This process allows them to keep legal control of banking & financial operations according to local & global legal governance laws.

AML solutions started from the rule based techniques which applied constraints on real-time monetary transaction systems to block invalid transactions. Initially those rules were in very simple form like validating upper, lower limit of individual transactions and gradually evolved to identify complex transactional patterns using machine learning and data driven models. Based on the industrial applications, those solutions can be categorized into three broad solution categories namely (figure 2-1 ):

1. AML detection solution
2. AML prevention solutions
3. Hybrid solutions (combination of other two approaches).

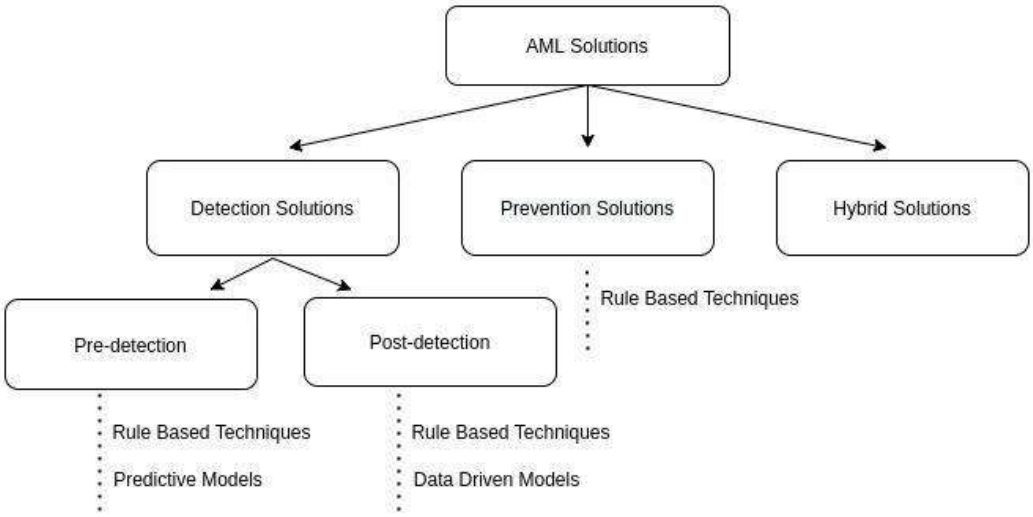


*Figure 2.1: AML Solution Approaches*

### 2.4.1 AML Prevention Solutions/Techniques

The main objective of this technique is to protect the financial system with **zero money losses** due to the money laundering activities. This technique is widely implemented using transactional business rules in the financial systems.

When considering the practical situation, it is very difficult to achieve zero loss figures due to the system usability and performance concerns. Majority of transactions in a financial system are legal while there are very few percentage of fraudulent transactions (formally known as class imbalance problem) [7][8][18][19].So, there is a **higher possibilities to block a genuine customer transaction due to lack of accuracy** of the blocking rule (false positive transaction blocks). This will lead to dissatisfied genuine customers which may lead to leaving the financial system very sooner. On the other hand, **higher constraints on the operational system causes degradation of system performance**. So, human experts (investigators) always try to keep only the rules which have closer to 100% accuracy.

Ex:" *Small amounts (5 $) of the same type (fund transfer), large number of transactions received to one account*" could not exactly figure out as a money laundering activity

because this account may belongs to social organization which received funds from the public. So, blocking such transactions creates a bad experience for customers. Due to that fact, businesspeople don't bother on implementing very strict transaction blocking algorithms on their financial system even if they have a risk of losing their money over such types of money laundering activities.

The other most common challenging part is, **difficult to maintain transaction blocking rules to match evolving customer habits over time**. This is commonly known as "concept of drift" in transaction fraud detection domain [7][16][17].So, application of AML prevention rules become limited to specific scope.

### 2.4.2 AML Detection Solutions/Techniques

AML detection techniques are used to identify the money laundering activities which are happening (via pre detection) or which have happened (post detection) through the financial system [7][20]. This is different from prevention techniques because prevention techniques try to stop (block) the money laundering action while detection techniques try to generate notification when the money laundering activity is happening or identify money laundering activities which happened through the system without immediately blocking the users' transaction. More commonly this technique is used by post data analysis solutions such as predictive models, data mining based techniques or real-time asynchronous pre-notification generation programs.

Monetary transaction processing systems generate massive amounts of transactions per day. So processing those massive amounts for detecting patterns requires powerful statistical models. These models may vary depending on the business domain & size of the business. But they can have common themes and patterns around them [6].

Once systems generate the suspicious notification, financial investigators immediately verify those system generated notifications by manually querying and analyzing the data points. This is generally conducted based on the experience of the investigator.

### 2.4.3 Hybrid Solutions/Techniques

Isolated applications of AML prevention solutions and detection solutions can covers only the limited scope in money laundering context. So it is necessary to have solutions which cover both of AML Detection Solution, AML Prevention Solution in a single system to cover higher context depth [7][8][15]. Hybrid solutions address that area.

These solutions use transaction validation techniques to block illegal transactions while minimizing the false blocking genuine transactions (apply only the 100% accuracy proved rules).They use fraud detection techniques to cover the missing validation coverage. Generally, these solutions are costly but safest solutions to reduce financial risk.

## 2.5 Anti-Money Laundering Control Layered Framework

Anti-money laundering related digital solutions spread over a larger research area. This is mainly because, ways of digital fraud are numerous and frauders dynamically adapt to the existing detection & prevention solutions [6][14][21].

Implementation of well architecture AML compliance solutions are complicated & costly. But it is required to have a compliance solution because, non-compliance financial solutions loose double digit cost for financial organizations due to higher fines cost in addition to compliance cost [6][13].So, banks and financial institutes tries to put their best effort to implement AML solutions, based on the existing industrial tools while researchers actively participating on finding innovative solutions.

There were considerably higher numbers of statistical & machine learning research conducted to implement fraud prevention & detection systems over past history. They were addressed the most common fraud detection problems like class imbalanced problem (percentage of legitimate transaction is far outnumber than fraudulent) [6][15][18][19][22] , transaction drift problem (frauders habits evolve over time) [16][17], verification latency issues (i.e.: Investigators cannot verify majority of suspicious alerts due to time & cost constraints )[7]. Some papers highlighted the

importance of implementing unsupervised based profiling methods to identify outliers [20] while some are focusing on supervised machine learning techniques like decision tree based approaches [15] ,neural network algorithms [16][23],etc. In addition to that, some research has designed the architectures to implement above approaches in real world industry [8][24][25][26].In higher level, all of the above solutions can be put into one or more combinations of real-time, near real-time or offline solutions categories (Figure 2.2).

Figure 2.2 shows the extended version of layered control flow of a fraud detection system [7] to demonstrate the general components of a AML controlling framework and their controlling boundaries .**This research focuses on the bold colored box ( Scoring Rules) to design an architectural data model to improve existing scoring engines performances.**

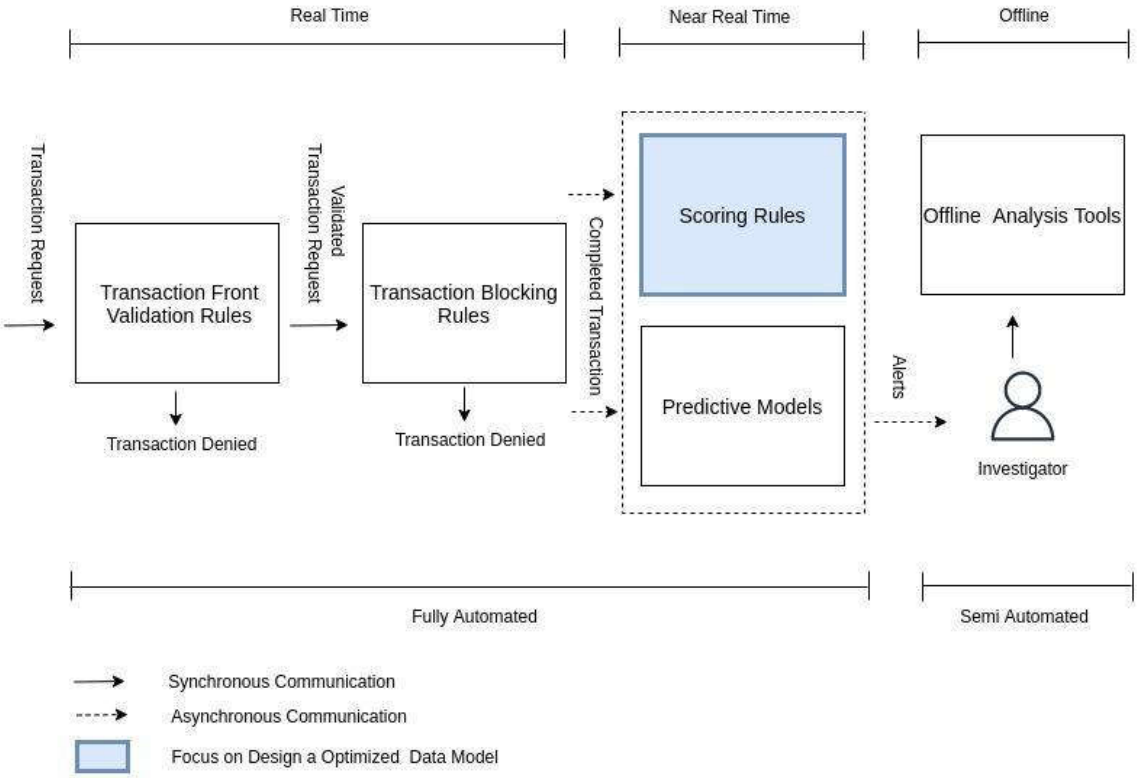


*Figure 2.2: Extended version of layered control flow of a fraud detection system*

### 2.5.1 Real-time Controlling Layers

Real-time components of the FDS controlling flow responsible for early validation and block, clearly identified transaction patterns before monetary transactions complete their cycles. “Transaction Front Validation Rules” and “Transaction Blocking Rules” are the most common components implemented in this layer.

Transaction front validation processes are responsible to perform conventional security checking activities like:

- Does the sender's account is in active mode?
- Does the sender have the correct permission levels? (Are there any partial operation blocks? Ex: Fund transfer block)
- Does the user account have sufficient funds?

Transaction blocking rules triggers only if transaction pass its front validations. These rules are like if-then-else statements to block clearly identified unallowable operations. These rules can evaluate in parallel and block the transaction if one of validation rule fails its requirements. So accuracy of each rule should kept in 100 % to prevent blocking genuine transactions. Example for such rules are:

- Does fund transfer to known blacklisted account (ex: Prevent fund transfer to the blacklisted company)
- Does the sum of daily transaction amount (including current transaction) exceed the user's daily fund transfer limit?
- Does this transaction exceed the user's upper transferable amount (for a single transaction)?
- Does the current transaction exceed the user's maximum transferable amount for the day?

The above rules have no hard boundaries. So some rules can be interchangeably used in above two components. Rule design & implementation totally depend on financial applications and preferences of subject experts in a particular organization.

### 2.5.2 Near Real-time Controlling Layers

Near real-time components of the FDS controlling flow start its execution process after real-time transaction flow is completed. The components in this layer operate as fraud detection components and try to generate alerts based on the criticalness of the money laundering activity. “Scoring Rules” and “Predictive Models” are the most common detection components used in this layer [7][8].

“Scoring Rules” are similar to transaction blocking rules which are designed by domain experts. However they may consume more time or cost to complete its evaluation process. After the evaluation, scoring rule assigns some associated risk value for each transaction and sends it to the alerting system. Most of the time, scoring rule engines use OLTP database or warehouse database to collects necessary history data for its operations. So it may have some considerable time based on the amount of data and performance of algorithm.

The “Predictive Models” use statistical and machine learning models to calculate fraud scores. Then it follows the same process which “Scoring rule engines” does.

### 2.5.3 Offline Controlling Layers

Offline components of the FDS controlling flow represent the advanced tools & techniques used to identify money laundering activities by semi-automated fashion (automatic tools + human action) .Traditionally data warehouse and mining based solutions exist in this layer. Sometimes, investigators use these tools to verify the auto generated alerts (comes from previous layers) before taking a further action on the suspicious activity.

## 2.6 Conventional and Temporal Databases

### 2.6.1 Conventional Databases

Conventional (non-temporal) databases were initially designed to model the current real world status of data. So they are called snapshot databases [27].These databases get updated according to the real world state changes. So the history of the previous state get losses, except for some special scenarios like maintaining audit data.

Conventional databases used the traditional database backups or transaction log files (or both) to recover important missing data in case of necessity. But those history data collection approaches cannot use for real time business analysis purposes due to various challengeable reasons like,

- Complicated backup loading process takes too much of time
- Log files may be missed due to various reasons like database management systems erase older files to maintain system security concerns,etc [34] .

### 2.6.2 Temporal Databases

Temporal databases or time-oriented databases are special purpose databases which are designed to store time varying data (time dependent data) [28].These databases can be in uni-temporal, bi-temporal or tri-temporal forms according to the number of temporal aspects (valid time , transaction time , decision time) taken for the design of the database [35].The literature of this research area is very active and a huge amount of research papers has been published over the last three decades [29][31] to address various aspect of this domain ex: Important of temporal data management [29][31][34][36] , time incorporation methods for database [32][33], temporal and real time databases [30]).

### 2.6.3 Relational Support for Temporal Databases

SQL2 supports DATE, TIME, TIMESTAMP, INTERVAL and PERIOD data types to provide temporal nature to relational databases. So , we can easily convert database relation to “valid time relation” by adding “valid start time & valid end time” attribute to existing relation or “transaction time relation” by adding “transaction start time & transaction end time” to existing relation. Modeling of bi-temporal relations can be achieved by adding above 4 attributes to the existing table [35].The key challengeable issue with modeling temporal databases with relational modeling is rapid evolution of schema. Because it needs to deal with schema changes in addition to dealing with the data store [35].

### 2.6.4 NoSQL Support for Temporal Database

NOSQL databases are gaining popularity due to their simplicity , scalability & performance concerns when comes to processing vast amount of data [38][39][40].These databases could came with four general types namely , key-value store, column family, graph and document databases [40][41][42].Many NOSQL databases do not have built in support for temporal aspects [37][39].But are support to incorporate temporal nature by versioning the necessary temporal attributes. Since the NOSQL database schema is loosely coupled with semi structured and no-structured data, NOSQL temporal databases not having issue like complex schema evaluations( like in traditional temporal relational modeling.)

Figure 2.3 shows a sample of bi-temporal modeling for “person” salary.

```
[
  {
    "Name": "John Smith",
    "Salary": "85,000",
    "StartTransactionTime": "2011-11-12T13:19:28.0000000Z",
    "EndTransactionTime": "2012-10-30T13:07:39.0000000Z",
    "StartValidTime": "2011-01-01T00:00:00.0000000Z",
    "EndValidTime": "2011-12-31T11:59:59.0000000Z"
  },
  {
    "Name": "John Smith",
    "Salary": "100,000",
    "StartTransactionTime": "2012-10-30T13:07:39.0000000Z",
    "EndTransactionTime": null,
    "StartValidTime": "2012-01-01T00:00:00.0000000Z",
    "EndValidTime": "2012-12-31T11:59:59.0000000Z"
  },
]
```


*Figure 2.3:Bi-Temporal Data in JSON [Monger, Morgan & Mata-Toledo, Ramon & Gupta, Pranshu. (2012)]*

## 2.7 Database Caching

### 2.7.1 What is caching?

Cache is a high speed data storage layer which provides a subset of data from primary storage to speed up the data access time rather than accessing data in its primary storage layers. Applicability of cache is diverse and some of the common use cases are: database caching [42], CDN caching [44][45], DNS caching [46],Web Caching [47] , API caching, etc.

### 2.7.2 Database Caching

Speed and throughput of the database directly implicated to the applications overall performance. Database caching provides a way to improve the throughput and performance of applications by caching the frequent access data [43][48].

Database driven applications can have different types of cache implementation techniques namely caching strategies. The applicability of correct strategy depends on data & data access pattern [49][50].For example, a gaming leaderboard application needs to use a completely different caching strategy than trending news display applications.

Below are some of commonly used caching strategies:

**Cache Aside (Lazy Loading):** This is the most commonly used caching strategy. In this strategy, application tries to retrieve data from cache when it receive data retrieval request. If the requested data item found in cache (cache hit), application serves that data to original client request. Otherwise (cache miss) application query data from the primary data source to server client request. Finally, application update its cache with latest fetched data

Key Points:

- Cache aside systems are resilient to cache failure because the unavailability of cache clusters may not kill the complete system. When cache cluster goes down, system can continue operation with primary data source with degraded performance.
- Since cache and primary data models are independent, cache can maintain lightweight models to further optimize the system performance.
- When cache aside, the most commonly used data write strategy is direct database writing strategy. So the initial data reads may get delayed (because cache databases load data only after cache miss).
- Data can be in state format [51].To face that challenge, TTL attribute or programmatically cache entry invalidation (application programmer invalidate cache entry when update operation performs) techniques can be used.

**Read-Through:** In read through caching approach, application read its data read requests directly from cache instead of regular primary data source. Cache has a special reader component which knows how to propagate those read requests to primary databases [11].When application receive a data retrieval request, application directly query that data from cache database like in cache aside pattern. If data found in cache, cache responded with its cache data.If application requested data is not exist in cache, cache provider component take the responsibility to fetch data from primary data source and update cache database accordingly.

Key Points:

- Lesser chance to maintain two different data models for cache & primary source because data loading logic is controlled by library or caching provider.

- Read through strategy has the same performance issue like in cache aside pattern when loading data item 1st time due to usage of direct database writes.
- Data can be in stale format. Write through strategy can combined with this approach to fix this issue.

**Write-Through:** In write through caching approach, application writes its data write request directly to cache instead of regular primary data source. Cache has a special writer component which knows how to propagate those writes to primary databases [52].This data update process happens in synchronous fashion and cache providers take responsibility of simultaneous data write to both cache & primary database [53].

Key Points:

- No stale data (guaranteed data consistency) because all data writes and updates going through the cache layer.
- There can be latency gaps between cache layer and primary data layer.
- Ensure only the acceleration of read operation. Write operations get delayed due to synchronous writes to both cache & primary database.

**Write Around:** In this strategy, applications redirect data write requests to the primary database directly. Caching mechanism comes into play when applications receive data read requests. When an application receives a data read request immediately after write operation always results in cache miss and data will be read from the backing store .

Key Points:

- Write around strategy usually comes in combination with two other previous strategies ( write around + cache aside or write around + read through).
- Good to use in situations where there are no subsequent read operations after writes.

**Write Back (Write Behind): I**n write back caching approach, application writes its data write request to cache similar to write-through approach. But caches are not trying to update primary data sources immediately after update cache. Caches return success

response to application and it updates the primary data sources in predefined schedules (asynchronous mode).

Key Points:

- Write operation time is improved due to asynchronous primary database update mechanism. So this is good for write intensive applications.
- Cash disk failure before primary database update can cause data losses

## 2.8 Problem of Real World Transaction Data

The lack of real-world monetary transaction data availability becomes one of the main problems in the anti-money laundering detection research area [61][62][63][64][65]. This is mainly because of three reasons:

1. Financial transaction data has Intrinsic private nature [63][64]
2. Lack of financial transaction data quality [65]
3. Financial Data Expose Can Cause Indirect Damages To Financial Institute

### 2.8.1 Intrinsic Private Nature of Financial Transaction Data

Most financial system related data includes personally identifiable information (PII) such as customer names, social security numbers, contact information, etc. [54].They are very sensitive because these data can be used to distinguish one person from another and unethical persons can easily misuse them to attack those individuals. So there are plenty of financial compliance rules and regulations has been publish over the globe (ex: GDPR in Europe [55], NYDFS financial cybersecurity rules in New York [56], Consumer Data Right in Australia [57], etc.) .In order to face this challenge, few of organizations has tried to use data sanitization [59] and anonymization techniques to expose financial data to research sector due to higher cost of those processes.

### 2.8.2 Lack of Financial Transaction Data Quality

Operational transaction systems are the most common systems which involve processing real-time financial data. They are generally focused on providing smooth support for data to day business transactions. So, these systems commonly use relational databases due to its inherent ACID property support features. Most of these databases are designed to store only the business operational data (known as snapshot database) rather than tracking necessary transactional metadata to data analysis purposes .As a result, usage of transactional data to analysis process becomes questionable related to the data quality.

### 2.8.3 Financial Data Expose Can Cause Indirect Damages to Financial Institute

The Banking & financial industry is a very competitive business area and thousands of same type institutes can exist in a single geographic region, ex: In 2020 there were 4377 FDIC insured registered banks in the United states [58].It is natural to have higher competition between those same types business to gain more profit over others.

With the popularity of data mining and machine learning techniques, most of today's businesses use computerized systems to find relationships between data points [60].So exposing financial data can leak the bank secrets to other organizations in the same field could damage existing market position (change the image of brand/product in customer mind) .

## 2.9 Synthetic Financial Transaction Data

Synthetic data generation is one of most active research area because unavailability of real-world dataset affected to most of data centric researches. Some of such common research areas are machine learning based model validation [73] and statistical testing [74] based researches.

Synthetic financial data is artificially generated data using computerized algorithms to simulate real world transactional data. The JPMorgan Chase & Co has publish a

research by specifying opportunities and challenges in synthetic financial data generation due to its privacy [72]. Those synthetic data address the above mentioned (section 2.8) limitations of real world monetary data availability issues.

The main disadvantage of synthetic transaction data is, they are unable to represent all different transaction patterns in real word data due to evolving customer behavior over time. But they are very useful for the researches like, improving fraud detection accuracy of specific known fraudulent activity, measuring efficiency/performance of anomaly detection algorithm, comparing existing architectural performances, etc.

**PaySim** [65] and AMLSim [66] are two of the most popular synthetic transaction generation tools which are used by a considerable number of researchers.

### 2.9.1 PaySim Transaction Data

Lack of real mobile monetary transaction data was one of the foremost problems in the scientific research community over the past two decade. PaySim project started with the aim of providing a method to generate simulated monetary transaction data to scientific research communities which preserve similar real world mobile transactions characteristics [66].

PaySim uses multi-agent based technology to simulate real human behavior. It uses mathematical statistical methods to generate a very similar data set to the original dataset which was derived from an undisclosed African country.

The researchers have selected five of the most common transaction types (Table 2.2) for their research simulations and simulators ran for several times using random seeds to generate one month of real-time(synthetic) transactional data(744 steps).The generated synthetic transaction data was visualized and it got a similar shape to real dataset. Figure 2.4 shows the visualization of simulated synthetic transaction (blue color) for TRANSFER transaction type with respect to original data (red color).

The main limitation of this tool is inability **to generate fraudulent transactions for given well known money laundering topologies**, because it was out of original research scope.

*Table 2.2:PaySim Supported Transaction Types*

| Transaction Type | Usage |
|---|---|
| CASH-IN | Deposit money in to account which increase account balance<br>Ex: Client top up account via merchant |
| CASH-OUT | Withdraw cash from the account which decreases the account balance<br>Ex: Client moves money out via merchant |
| DEBIT | Sending money from mobile money service to the bank.<br>Ex: Client moves money in to bank |
| PAYMENT | Exchange cash for something (good or service)<br>Ex:Paying for good or service to merchant |
| TRANSFER | Sending money from one account to other similar kind of user<br>Ex: Client send money to another client |

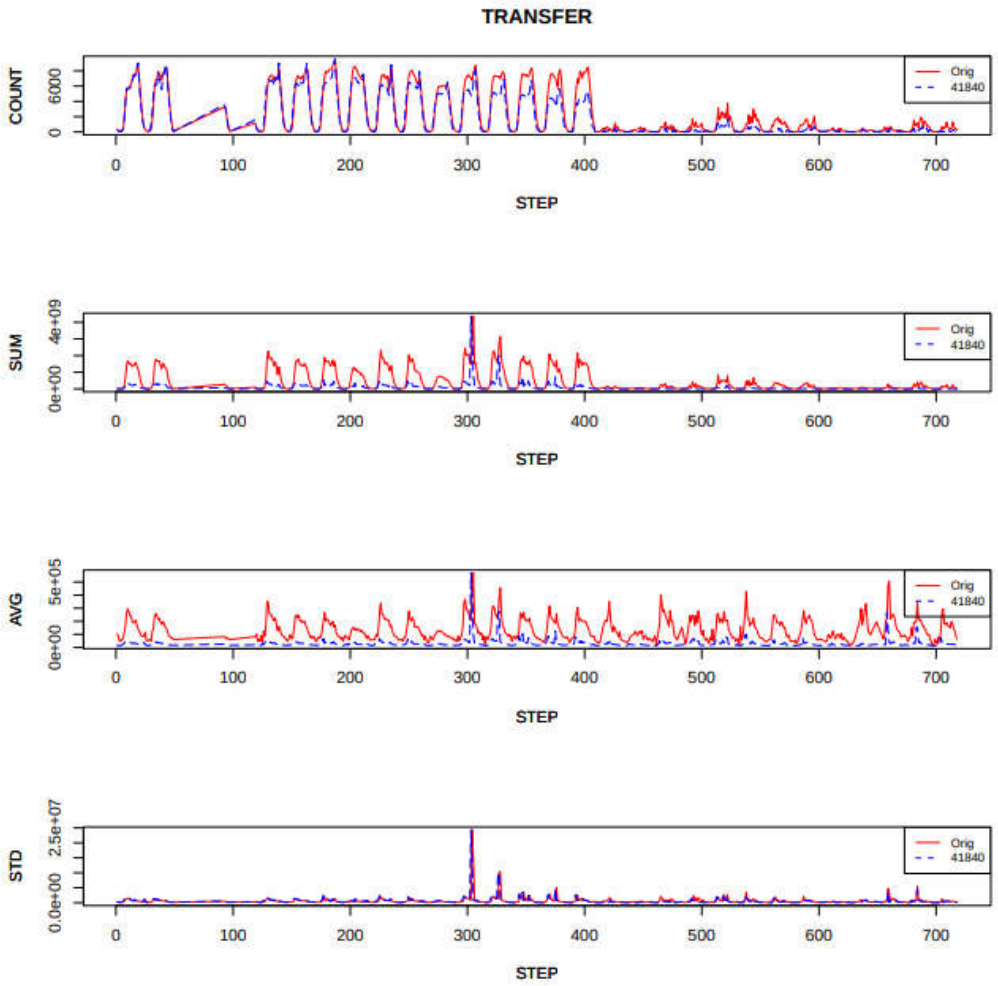


*Figure 2.6: Visualization of transaction type TRANSFER*

### 2.9.2 AMLSim Transaction Data

Anti-money laundering model validation can be considered as another challenging aspect of the transaction fraud detection domain. This is mainly because of lack of money laundering transaction data for known money laundering topologies due to inherent class imbalance problem. International Business Machines (IBM) Corporation's anti-money laundering simulator (AMLSim) is specially designed to address above issue by providing capability to generate synthetic financial transactions for known money laundering topologies.

### 2.9.2.1 Supporting Typologies

AMLSim simulator is supported to generate simulated transactions for eight well known money laundering topologies: fan_in, fan_out, bipartite, stack bipartite, random, cycle, scatter_gather and gather_scatter.

**Fan_in**

This topology consists of multiple accounts "ai" sending large numbers of X amounts of money to a single main account "m" in a predefined period of time. Figure 2.7 shows this topology structure.

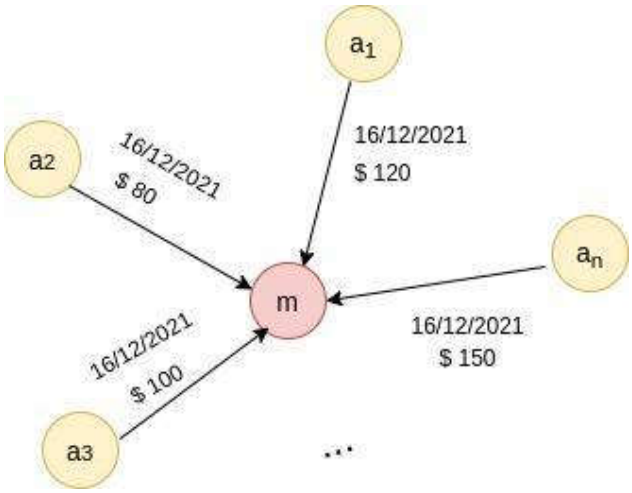


*Figure 2.7: Fan-in Topology*

**Fan_out**

This topology consists of a single main account "m", distributing money to multiple accounts "ai" over a predefined interval of time. Figure 2.8 shows this topology structure.

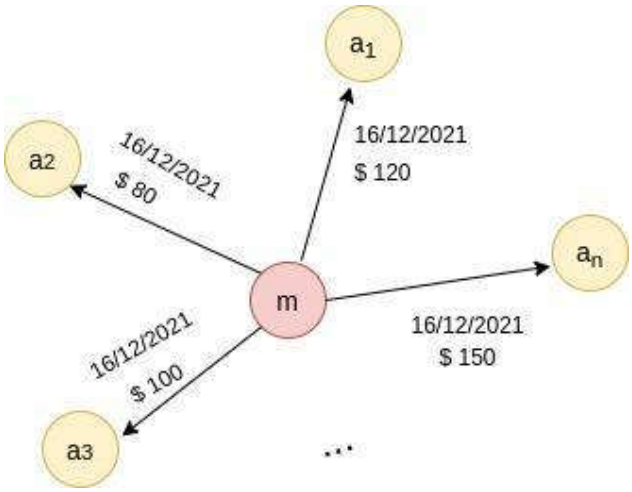


*Figure 2.8: Fan-out Topology*

**Bipartite**

This topology consists of more than two accounts “mi”, distributing money into the same set of multiple accounts “ai” over a predefined interval of time. So the two main accounts have an unobvious relationship with each other. Figure 2.9 shows this topology structure.

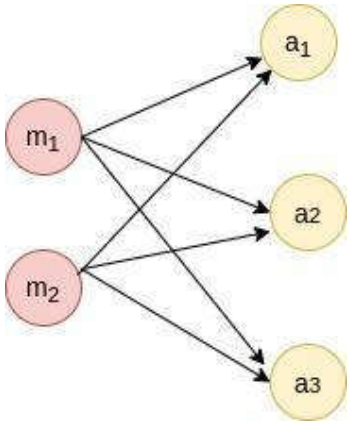


*Figure 2.9: Bipartite Topology*

**Stack_bipartite**

This topology is an extension of bipartite topology which includes a combination of bipartite networks. Figure 2.10 shows this topology structure.

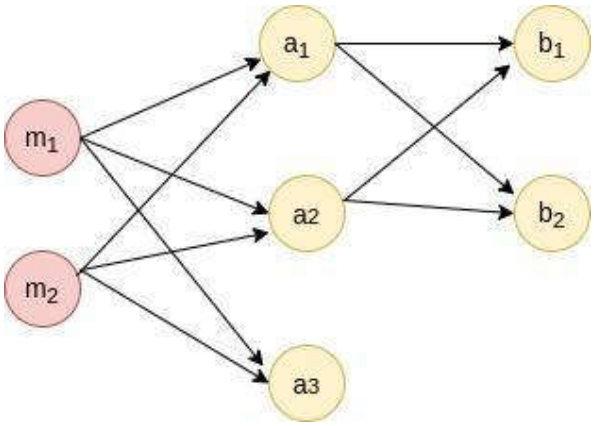


*Figure 2.10: Stack_bipartite Topology*

**Cycle**

In cycle topology, a main account "m" sends a large amount of money to neighboring account "a1".Then the beneficiary account sends some percentage of money to its neighboring account "a2".This process repeats until the main account, "m" receive some percentage of money back. Figure 2.11 shows this topology structure.

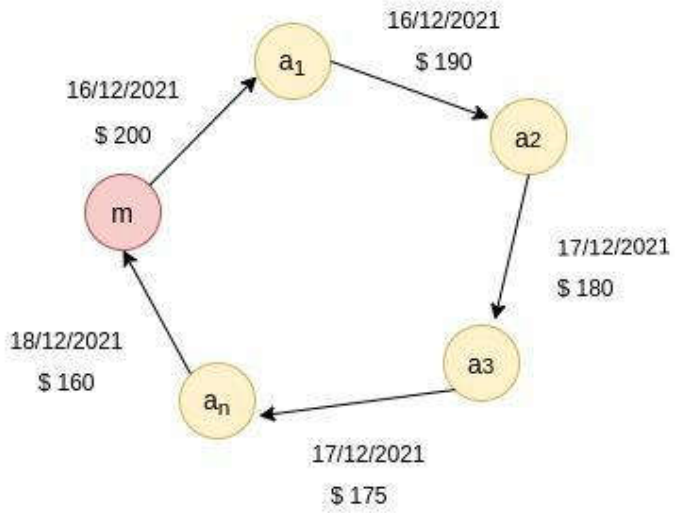


*Figure 2.11: Cycle Topology*

**Random**

This topology is similar to cycle topology but it selects the neighboring nodes in random fashions. There are no restrictions to receive funds to the same main account "m" at the end of transferring process. Figure 2.12 shows this topology structure.

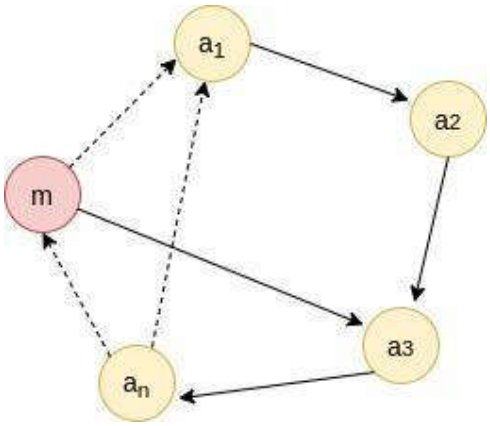


*Figure 2.12: Random Topology*

**Scatter_gather**

This topology consists of the combination of 'fan_in' and 'fan_out' topologies.The main account 'm' spread money to multiple intermediate accounts 'ai'.Then the intermediate accounts keep their margins amounts and send their remaining to single beneficiary 'b'. Figure 2.13 shows this topology structure.

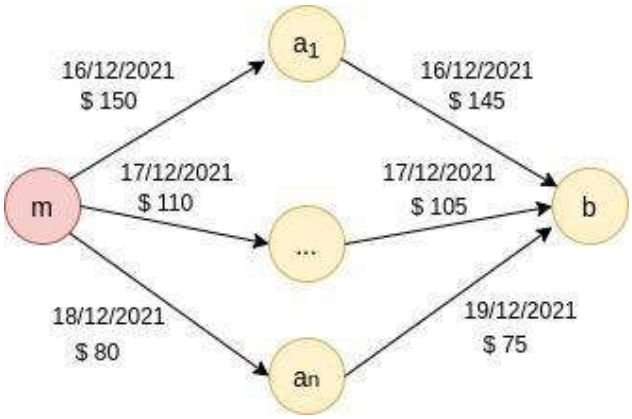


*Figure 2.13: Scatter_gather Topology*

**Gather_scatter**

In this topology, multiple accounts "ai" send money to an intermediate main account. Then the intermediate main account keeps its margin amount and redistributes money to multiple accounts "bi". Figure 2.14 shows this topology structure.

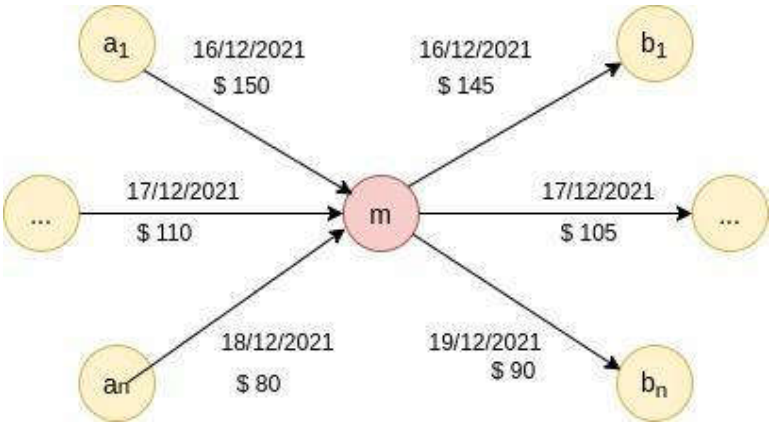


*Figure 2.14: Gather_scatter Topology*

### 2.9.2.2 Data Generate Process

The AMLSim simulator's architecture has two major components namely "Transaction Graph Generator" and "Transaction Simulator" (Figure 2.15).The simulator runs those two components one after other to complete the simulation process.

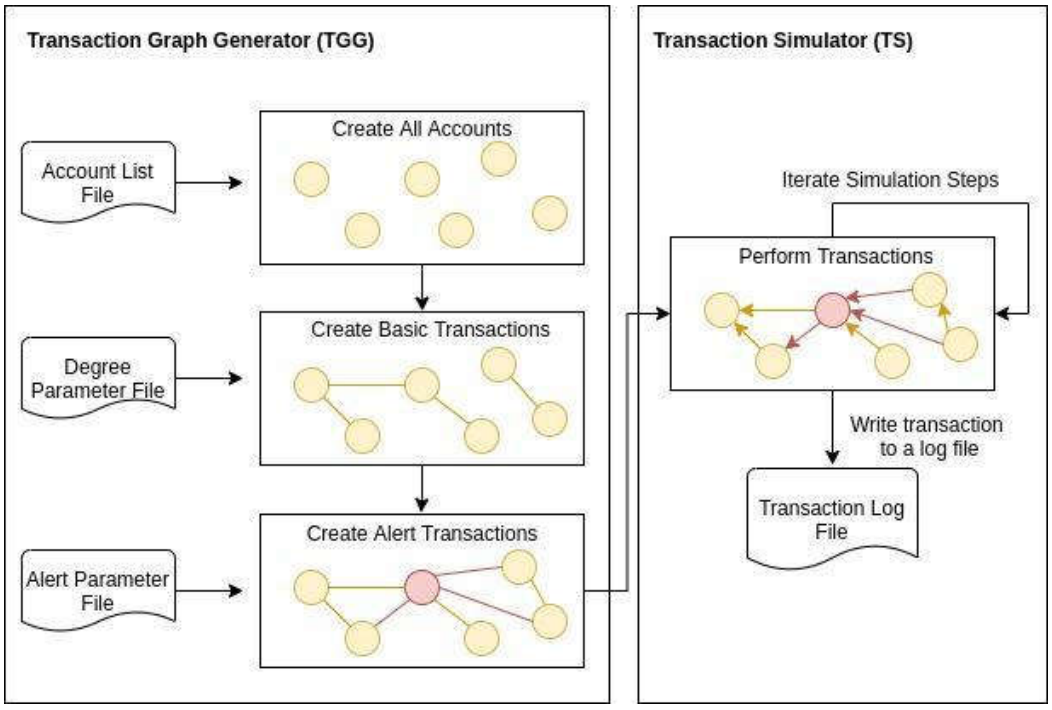


*Figure 2.15: An overview of the AMLSim Architecture*

Source: https://github.com/IBM/AMLSim/wiki/Overview

**Transaction Graph Generator (TGG)**

Transaction Graph Generator is a python program which takes the responsibility of creating the transaction network structure. This process has three major steps:

**Step 1: Create All Bank Accounts**

TGG takes the account details file as input and it generates all the necessary bank accounts based on user's specified parameters. Below Table 2.1 is the tabular view of sample account.csv file.

*Figure 2.16: AMLSim Account Config File*

| count | min_balance | max_balance | country | business_type | model | bank_id |
|---|---|---|---|---|---|---|
| 6000 | 100 | 200 | US | I | 1 | bank_a |
| 100 | 5000 | 10000 | US | I | 1 | bank_b |
| 2000 | 100 | 200 | US | I | 2 | bank_b |

The count parameter tells the number of accounts that should be created with the given attribute list. The min_balance & max_balance attributes tells the minimum & maximum balance for an account when initially created the account. The country, business_type tells additional account related descriptions which could be used in the fraud detection stage. The model attribute represent the account behavior just like money laundering topologies. This allows to take one of the predefined values "0-single, 1-fan-out, 2-fan-in, 3-mutual, 4-forward, 5-periodical".The last parameter, bank_id is optional and it represent the account belonging bank name.

**Step 2: Construct the Base Transaction Graph**

After constructing all the accounts, TGG takes the degree parameter file as input and creates the basic transactions between constructed accounts. The degree file contains three parameters: Count, In-degree, Out-degree. It describes, how many accounts have in-degree incoming transactions and how many accounts have out-degree outgoing transactions. Below is the tabular view of sample degree.csv file.

*Figure 2.17: AMLSim Degree Config File*

| Count | In-degree | Out-degree |
|---|---|---|
| 1 | 2421 | 2470 |
| 2 | 12018 | 12231 |
| 4 | 78149 | 78268 |

**Step 3: Incorporate Suspicious Transactions to Base Transaction Graph**

Finally, TGG injects the money laundering transactions to base transaction graph based on alert parameter file.

*Figure 2.18: AMLSim Topology Config File*

| count | type | schedule_id | min_accounts | max_accounts | min_amount | max_amount | min_period | max_period | bank_id | is_sar |
|---|---|---|---|---|---|---|---|---|---|---|
| 2 | fan_in | 2 | 5 | 6 | 2700 | 3000 | 5 | 20 | bank_a | "True" |
| 2 | fan_out | 2 | 5 | 10 | 100 | 200 | 10 | 30 | bank_a | "True" |
| 2 | cycle | 2 | 7 | 11 | 100 | 200 | 5 | 20 | bank_a | "True" |

The count parameter defines the number of alert sub graphs to be generated for the specified transaction set. Type defines the money laundering topology. Schedule_id denotes the order of transactions. The min_accounts , max_accounts define the minimum and maximum number of invalid accounts to be generated. The min_amount, max_amount defines the minimum and maximum of initial transaction amounts. The min_period, max_period defines minimum & maximum overall transaction period in number of days. The bank_id is a optional field and it denotes the bank from which the account transfers from. The is_sar parameter tells weather generated SAR is true or false.

Once the above 3 steps are completed, the simulator switches to other component for the transaction simulation process.

**Transaction Simulator (TS)**

The transaction simulation component is an extended version of PaySim simulator library. It is responsible for generating time series of synthetic transaction data using multi-agent based simulation.

## 2.10 Summary

This chapter discussed in-depth details of the imagination of the anti-money laundering systems and problems and solutions given by academic and industrial practitioners in the existence context. More specifically, first sections described the banking operation evolution over the time from manual bookkeeping to online and real time system.

Middle sections elaborated the industrial usages and academic researches going on the similar contextual nature and the end sections described some important tools, techniques and concepts used in the industry to support real time monetary transactional data analysis. Lastly it described the industrial available transactional data generation tools which can use in the solution implementation and testing phases..

# 3 CHAPTER 3 - METHODOLOGY

## 3.1 Introduction

This chapter discusses in detail descriptions of the proposed architectural solution to address challenge 1 & challenge 2 (section 1.3) for anti-money laundering "scoring rule engines".

The beginning sections of this chapter will briefly discuss the most essential business and technical factors which need to take account while engineering a novel risk accounted architectural solution.

The middle sections of the chapter overview the components of the proposed architecture along with the functionalities of each component. Finally, it will discuss the design process of the proposed solution by giving several practical examples.

## 3.2 Factors to Consider

Micro data modelling architecture is a data modelling architecture for anti-money laundering systems that specifically targeted the near real-time scoring rule engines. It suggests a novel approach to optimize existing near real-time, rule-based fraud detection engines while removing the operation burden on OLTP systems. The key concept of this architectural solution is to early identify money launders by considering the business risk in the solution design phase. It accounts for both the technical and business factors as its solution design considerations rather than traditional fully technical architecture-based solutions. This section states the essential **business & technical considerations** that need to evaluate when designing both business and technical parties accepted solution.

### 3.2.1 Business Factors Force for Design Considerations

Businesspeople always think to improve their profit by minimizing the risk of the business. They are not much worrying about whatever modern technologies which IT

people using on. But they worried about the profit and growth of the business and the legal/ organizational policy compliance. Related to the money laundering context, businesspeople focus on:

**a. Need to identify money launders before leaving the financial system:**

There are no hard and fast rules to detect all types of money laundering activities that could happen over financial systems. But most of the technical architectures has been designed by focusing to identify almost all money laundering activities happening over organization business boundary. So we can call them as traditional generalized architectures. In those generalized architectures, almost all the near real-time scoring rules are pointed to a single monolithic database (as a unit).So, individual scoring rules are difficult to adjust for specific money laundering patterns with higher risk. Due to this reason, each scoring rule needs to filter data on large massive datasets (probably the entire database) which takes considerable time. As a result, money lauders have considerable time to leave the financial system. So, businesspeople focus on minimizing business risk by identifying money launders as soon as possible. In other words, they are focusing on architectures that take business risk factors when designing technical architecture.

**b. Need to prevent damage which may cause to the original legal customer:**

Due to the variability of money laundering activities, technical people try to implement complex constraints on the operational transaction systems which may cause a bad customer experience to the genuine customer base. Businesspeople always try to maintain good customer satisfaction by focusing on minimizing such issues.

**c. Cost of system migration should be minimum**

Implementation of novel solutions should have good support from existing available technologies and they should support migrating existing solutions with minimal cost.

### 1.1.1 Technical Factors Force for Design Considerations

Technical people think to build fully-featured systems to identify all the money laundering activities which are happening through the financial institute while

minimizing the damage to the OLTP systems. Below are the most important technical factors related to the anti-money laundering context:

**a. Ability to data analysis quickly while keeping system performance at a good level:**

Banking and financial systems use relational databases for most of their OLTP systems due to inherent ACID property support. In general, most financial organizations use those OLTP databases as data sources for scoring rule engines. These databases have performance issues when executing complex analysis queries on top of them. This problem could not solve by replacing data-warehouse databases because it introduces novel issues due to the massive amount of data. So novel design architectures should support optimizing existing scoring rule engines while maintaining OLTP performance at a good level.

**b. The system should support to easy integration of heterogeneous OLTP systems:**

Banking and financial systems consist of several granular systems to support different sorts of services to their customers. Those systems may be implemented by the company's internal IT department or outsourced organizations. Those systems may use different tools and technologies as their requirement or company preferences. So the novel architectural solution should support to easily integrate with those heterogeneous systems.

**c. One system downtime should not affect to whole scoring rule analysis process:**

Traditional scoring rule engines use OLTP database or data warehouse database as their primary data sources. There can have situations where those data sources may not work as expected due to internal or external reasons like errors, system maintenance, etc. This could cause cancel the preplanned scoring rule execution process which gives the money launderer a bonus time to conceal from laws. In order to minimize this situation, newer systems should design to loose coupling with existing other systems to minimize the impact of whole system downtime ( at least part of scheduling tasks should be able to run).

**d. Good implementation support**

The system should support to implement in smaller financial institutes to large enterprise level organizations without totally changing the initial design.

## 3.3 Micro Data Model Architecture In-Depth

Micro Data Model Architecture is primarily designed for the banking and financial industry to address the near real-time scoring rule related issues (**1.3 Problem in Brief)**. It wrapped the enterprise key business concerns **(3.2.1-business concerns)** inside the technical design **(3.2.2-technical concerns)** to provide both business and technical people accepted solution.

Figure 3.1 shows the top-level data modelling architecture. The symbols A, B, C represent the micro temporal databases that act as the main character in this design. Each of these micro temporal databases expects to receive the transactional records in a real-time fashion and keep store records until predefined time intervals as like a cache database. The transactional data injection processes can be built using the change capturing process (CDC) [67][68] with stream technologies[69][70] which is out of this research scope.

In this architecture, temporal micro databases have been modelled by taking money laundering risks to address above discussed technical & business concerns. Section 3.4 describes this modelling method in more detail.

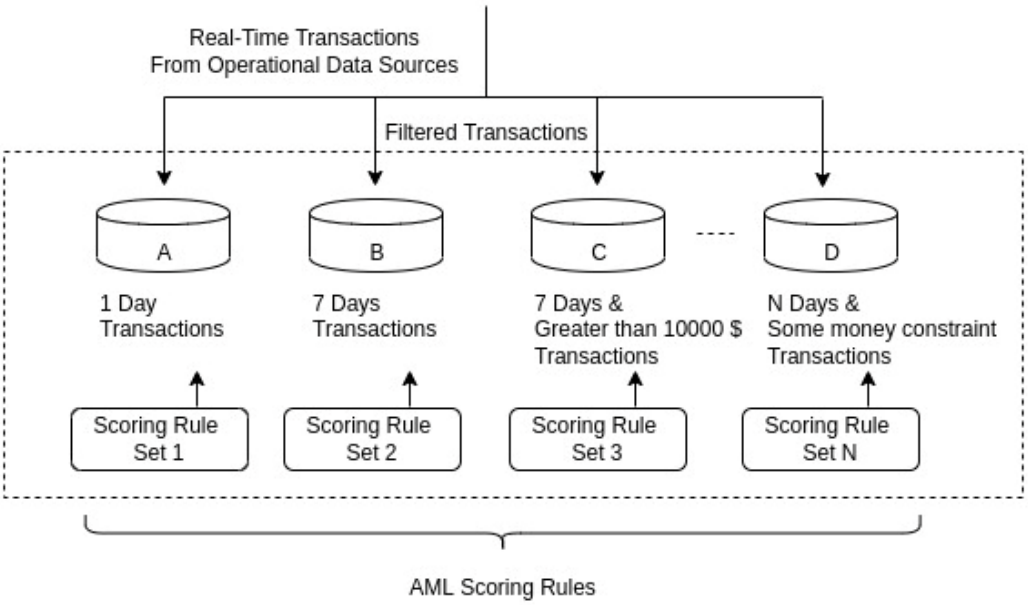


*Figure 3.1: Micro Database Architecture*

## 3.4 Architectural Modeling

Micro database architecture modeling consists of two higher-level activities. They are:

1. Scoring Rule Organization
2. Data model Organization

### 3.4.1 Scoring Rule Organization

Scoring-rule organization is the process of grouping existing money laundering detection rules according to given guidelines. This is a special task that should be done by a domain expert person in the financial organization. In general, the organizational financial investigators can take this responsibility who already participating in the new scoring-rule design process in traditional financial fraud detection systems.

The higher level scoring rule organization process can be further divided into two sub-stages namely: Initial Rule Group Split & Rule Group Specialization.

**Stage 01: Initial Rule Group Split**

In this step we are grouping scoring rules by considering the history data requirement of each rule. Figure 3.2 Venn diagram shows the generalized grouping idea. The G1, G2, G3, Gn represent the rule groups. They have been grouped according to history transaction record requirements. Rules in the G1 group is looking for minimal history transactional records. The rest of the groups (G2, G3, and G3 orderly) expect a higher amount of transaction history records.

The number of scoring rule groups is a variable that depends on organizational business rules and other financial constraints. So this could be taken as organizations' financial investigators' responsibility who design rules for their unique business domain.

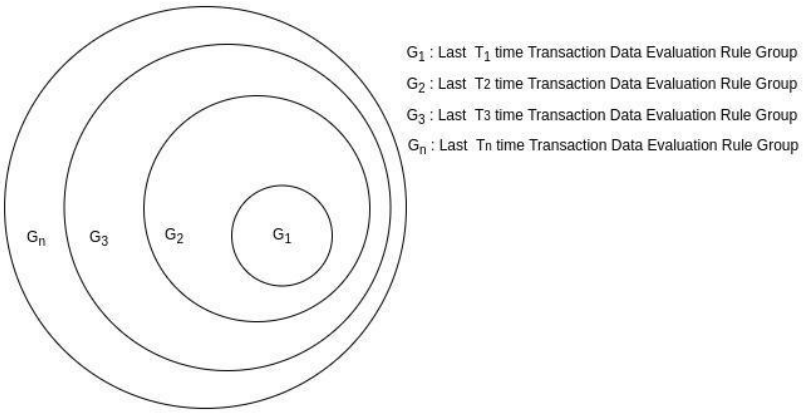


*Figure 3.2: Initial Rule Group Splitting*

Below is a simplified example of the idea where a financial investigator group rules G1, G2, G3 according to 1 day,7 days, 30 days' time frames.

Rule Group 1: Rules with temporal record requirement of last 24 hour time frames
Rule Group 2: Rules with temporal record requirement of last 7 day time frames
Rule Group 3: Rules with temporal record requirement of last 30 daytime frames

**Stage 02: Rule Group Specialization**

In this stage, we are identifying the specialized hidden rule groups inside the above-identified rule groups **by taking the risk of money laundering patterns**. This is important because these specialized groups have more power to change data modelling decisions which we will discuss more in section 3.4.2.

Figure 3.1 demonstrate this concept. The Gn represent an arbitrarily selected scoring rule group.Gi represents the newly identified, specialized scoring rule group from Gn.Gj represent the remaining rules in Gn, after extracting the Gi rule group.

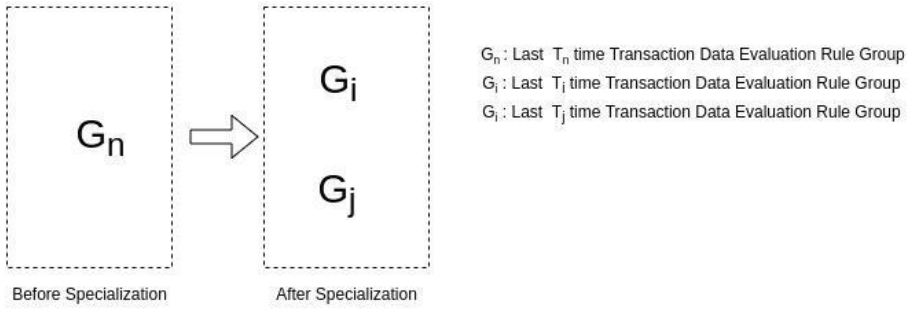


*Figure 3.1: Rule Group Specialization*

Below are two examples that explain this concept further.

Ex 01: Splitting bigger amounts of money into smaller portions and distributing them to peer account is a well-known money laundering pattern. Normally these types of peer transferring processes happen roughly over a week to conceal from other money

laundering detection rules (ex: maximum transaction limit per day, higher number of small transactions over the short time frame, etc.).**Keeping all one-week time frame rules in a single group can lead to losing concentration on these types of frequently Occurring patterns.** To address this case, we can form a new scoring rule group to cover that specialized situation.

Ex 02: Generally, utility bill payments happen on monthly basis. So, there could have a scoring rule to identify money launders that execute an unusual number of monthly utility payments. This rule belongs to the 30-day transaction history record required group. If the financial investigator has been noticed in history transactions that there is a higher possibility of losing money using this pattern, he can focus on forming a new rule group for this pattern.

### 3.4.2 Data Model Organization

Data model organization is the process of identifying and defining micro temporal databases by taking identified rule groups as inputs. These micro databases should have two main characteristics.

1. Temporal Database Behavior: Databases should have the ability to maintain historical records, instead of modifying existing records like in snapshot databases.
2. Database Cache Behavior: Micro databases should store the temporal data up to a certain time extent. When new transaction data comes, they should override the first inserted record like in the FIFI cache replacement policy. This way we can keep a limited data amount in each micro database.

There are three different ways that we can model these micro temporal databases. They are

1. Data Cache Time Focus Organization
2. Data Cache Size Focus Organization
3. Transaction Risk Focus Organization

**Data Cache Time Focus Organization**

The data cache time focus organization method takes the scoring rule group as input and decides the number of micro databases based on the lifespan of scoring rule groups. Each record in each database has the same TTL value as its living micro database. This approach keeps only a single micro database (or cluster) for all same caching time period rule groups. In other terms, **this method does not take any scoring rule group as specialized**.

Ex: Consider a situation where we receive below 4 rule groups as input from the above scoring rule organization (section 3.4.1) step:

**Rule Group 1:** Scoring rules with temporal record requirement of last 24-hour time frame

**Rule Group 2:** Scoring rules with temporal record requirement of last 7-day time frame

**Rule Group 3:** Scoring rules with temporal record requirement of last 30-day time frame

**Rule Group 4:** Specialized rule with temporal record requirement of last 30-day time frame

Based on the time frames of above scoring rule groups, we can clearly identify three separate lifespan groups. So, we need only three micro temporal databases to cover the above four rule groups. The temporal time period for each database is orderly gets 1 day, 7days & 30 days. Below are the database-rule groups mapping details.

**Database 1:**

Store the last 24-hour transactional records. So, all the scoring rule groups with temporal record requirements of the last 24 hour time frame are pointed to this database. According to our example, Rule Group 1 pointed to this database.

**Database 2:**

Store the last 7 days transactional records. So, all the scoring rule groups with temporal record requirements of the last 7 day time frame are pointed to this database. According to our example, Rule Group 2 pointed to this database.

**Database 3:**

Store the last 30 days transactional records. So, all scoring rule groups with temporal record requirements of the last 30-day time frame are pointed to this database. According to our example, both Rule Group 3 &Rule Group 4 pointed to this database.

**Data Cache Size Focus Organization**

The data cache size focus organization method takes the scoring rule group as input and decides the number of micro databases based on the lifespan of scoring rule groups exactly similar to the above "data cache time focus organization". But the actual data storage for each temporal database is defined by "database size" constraints to address high-speed database size growth. The actual implementation of this process can be done by either one of the following methods:

**Method 1:** Define the temporal database size at the database storage capacity level. Individual transactional records do not have their own minimum lifespan. They have only a maximum lifespan which is similar to its living micro database. But it does not guarantee to live up to that period.

Ex:

**Database 1:**

Configured to store 5 GB for last 24-hour time frame records

**Database 2:**

Configured to store 20 GB for last 7 days' time frame records

**Database 3:**

Configured to store 100 GB for last 30 days' time frame records

**Method 2:** Define the temporal database size at the database record count level. This can be done either for the individual user level or commonly for all users.

Ex:

**Database 1:**

Configured to store 1 million transactions for last 24-hour time frame records

**Database 2:**

Configured to store 10 million for last 7 days' time frame records

**Database 3:**

Configured to store 100 million for last 30 days' time frame records

The most important part of this design is to take specific consideration on transaction incoming rate when deciding the database size. Otherwise, there is a higher possibility to lose some of the older records before scoring rules complete the evaluation processes.

Problems:

- Can miss some important history data if transaction incoming rate increases
- It is difficult to define a size for the databases without having prior stats

**Transaction Risk Focus Organization**

Transaction Risk Focus Organization method takes the scoring rule groups as input and decides the number of micro databases by **taking both lifespan of scoring rule groups and associated risk** of each scoring rule group. The primary objective of this method is to isolate riskier data to their own temporal databases to further optimize the data processing engines. For the data caching behavior, they can use either "**cache time**" based storage or "**cache size**" based storage which is discussed in the above two approaches.

Ex: Consider a situation where we receive below 4 rule groups as input from the above scoring rule organization (section 3.4.1) step:

**Rule Group 1:** Scoring rules with temporal record requirement of last 24-hour time frame

**Rule Group 2:** Scoring rules with temporal record requirement of last 7-day time frame

**Rule Group 3:** Scoring rules with temporal record requirement of last 30-day time frame

**Rule Group 4:** Specialized rule with temporal record requirement of last 30-day time frame

Cache Time-based Example:

**Database 1:**

Store the last 24-hour transactional records. All **non-specialized scoring rule groups** with temporal record requirements of the last 24 hour time frame are pointed to this database. According to our example, Rule Group 1 pointed to this database.

**Database 2:**

Store the last 7 days transactional records. All **non-specialized scoring rule groups** with temporal record requirements of the last 7 day time frame are pointed to this database. According to our example, Rule Group 2 pointed to this database.

**Database 3:**

Store the last 30 days transactional records. All the **non-specialized** scoring rule groups with temporal record requirements of the last 30-day time frame are pointed to this database. According to our example, Rule Group 3 pointed to this database.

**Database 4:**

Store the last 30 days transactional records. **Specialized scoring rule group**/s with temporal record requirements of the last 30-day time frame is pointed to this database. According to our example, Rule Group 4 pointed to this database.

### 3.5 Data Injection & Remove Process

Data injection & removal processes have higher depths (can take as separate research areas) and they are kept outside of this research scope. But having some basic idea of those concepts will support readers to get understand this architecture better. So, we try to give a rough idea of those processes by going through basic details in the below paragraphs.

Real-time transactional records are expected to be forwarded to those micro databases in a real-time fashion. This can achieve using multiple ways. One such method is to use the change data capturing (CDC) [67][68] process along with stream technologies[69][70] to identify and inject transaction records into micro temporal databases.

Implementation of removing invalid data processes can also implement in several ways including

- Store transaction record time along with other transaction attributes. A separate scheduling process evaluates the validity of each record in a periodic fashion and removes invalidated records. (or can setup trigger to execute when we insert new record)
- Use databases inbuilt Time To Live features (TTL index in Mongo DB).
- Use database in-built event triggers

## 3.6 Summary

This chapter discussed in-depth details of the proposed micro data modelling architecture and its design process. The design of this architecture has two key design activities: Rule Group Organization & Data Model Organization. Chapter discuss those activities by giving several real-world examples to make a better understanding of each activity.

The rule group organization activity is the process of grouping anti-money laundering rules into set of groups based on their temporal history record requirements. There can have some special money laundering patterns which can cause higher damages to financial organizations (already identified by organizations).Since higher damages are higher riskier to organizations, the ruling group design process try to identify rules which can detect those actions and create special rule groups which can use in the data model organization activity to design more optimized data models by considering business risk actors to its data model. This specialization group creation process is known as “Rule Group Specialization”.

The data model organization is the process of design & organizing micro databases by taking input as above identified rule groups. This process can achieve in three different ways: Data Cache Time Focus Organization, Data Cache Size Focus Organization, Transaction Risk Focus Organization. This research encourages to use of the 'Transaction Risk Focus Organization' organization to build a risk-driven solution that satisfied both organizational business and technical factors.

# 4 CHAPTER 4 –MICRO TEMPORAL DATABASE GENERATOR

## 4.1 Introduction

Converting the OLTP schema into temporal database schema is a complicated & error-prone process that can take considerable time. Since micro database architecture design is an abstract design concept, it allows users to use different database types (different database vendor's products) as micro databases irrespective of the relational OLTP database type. So the database design & implementation process can become further complicated.

This chapter introduces an automated tool to replace the manual temporal database schema design & database implementation process to remove its unnecessary complications.

## 4.2 Introduction to Micro Temporal Database Generator

Micro temporal database architecture is an abstract design concept. So the design and implementation of real-world micro temporal databases afford users to make independence decisions based on their data & technology preferences. Some of them are:

- Selection of database type
- Selection of database vendor
- Selection of database cache maintenance strategy

**Selection of Database Type**

Traditional monetary OLTP databases are designed to use relational databases due to their natural transactional support. But it is not wise to use them for data analysis applications because the relations with constraints based queries take considerable time than relations without constraint queries. Due to this reason, most of the modern data

analysis solutions tries to avoid such constraints or use NoSQL based database as their primary data sources.

Since micro temporal architecture is technology-neutral, users can build NoSQL type databases as micro temporal databases without changing the existing OLTP database engine. So they can take the ultimate benefit of using both technologies. Additionally, micro temporal database design provides flexibility to easily migrate into newer technologies to align organizational products to newer technologies.

**Selection of Database Vendor**

Databases can be categorized into two main types: relational or non-relational. Implementation of each category type can have several forms based on various decision factors (ex: storage optimize implementations, query optimize implementations). Database vendors are the organizations who provide different branded database products specifying underlying specifications (specialities). As an example, MySQL and Oracle are two relational databases owned by the company named Oracle Corporation. Although both of those two databases support standard SQL syntaxes, Oracle provides a different set of features for advanced processing(ex: different index types to support query optimizations)[71]. So the micro temporal database implementers can take the advantage of loose coupling architectural attribute to select different vendors' products for each database independently.

**Selection of Database Cache Maintenance Strategy**

Databases of micro temporal design architecture has two special properties. They are:

- Temporal Behaviour : Capable to store history data
- Caching Behaviour: Each database should hold data only for a given specific time period

Database caching behaviour maintenance methodologies are diverse. Some database engines (vendors) support automatically clean history data based on predefined configurable values (e: TTL) while others provide partial support or do not provide any support to clean historical records. So the micro database architecture

implementers need to think about this behaviour and select the appropriate technologies based on their organizational preferences.

Designing and implementation of micro temporal databases to match real word organizational requirements are extremely complicated and time-consuming process because:

- Manual OLTP schema to temporal mapping can have human mistakes
- Manual mapping need to do repetitively if the customer need several micro temporal databases (this is the general behaviour)
- Different database vendors support different query syntax even with the same database type (MySQL vs PostgreSQL)

**Micro database generator is a tool to address the above real-world industrial implementation concerns.** This project build as is an open-source project which provide anyone to use and modify code as they preferred.

### 4.3 Architecture

Micro temporal database generator architecture has three components: “Rest API”, “Temporal Schema Mapper” and “Database Generator”.

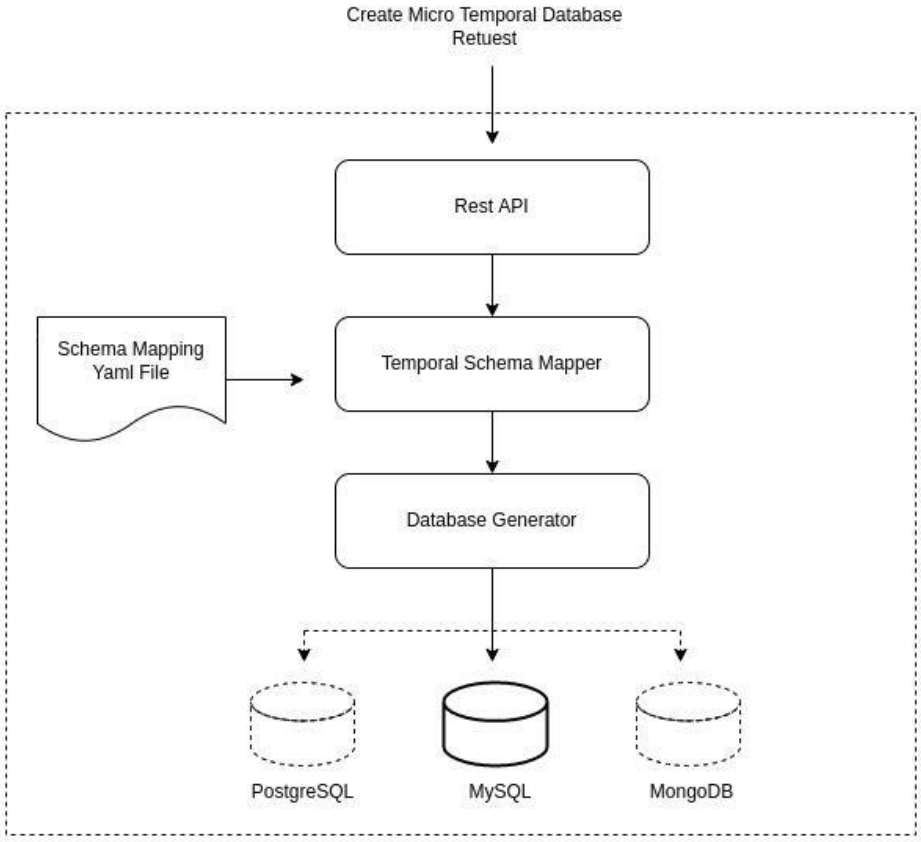


*Figure 4.1: Architecture of Micro Temporal Database Generator*

### 4.3.1 Rest API

This is the triggering point of the database creation process. End users or external applications can pass “database name prefix” & “temporal time period” to micro database generator application through restful web API.Then the API returns the complete name of the generated database and its data caching time period as API response.

### 4.3.2 Template Schema Mapper

The main responsibility of schema mapper is to create temporal schema model based on user’s inputs instructions given in the “schema_mapper.yaml” file and validate the attributes against tools’ guidelines. Appendix D shows the sample generated schema for relational type, temporal database with data caching period of 7 day.

Figure 4.2 shows the schema mapping for relational transactions (Refer appendix C for full version).The"tableName" attribute defines the name of the relational OLTP table. Same table name will use in the temporal database if underlying temporal database is also a relational database (as default implementation). The "tableType" attribute use to define weather this table should convert in to temporal table by adding additional temporal attributes or keep it as "LOOKUP" table. The "isMainTable" attribute is use to define main table in the OLTP table schema. Usually this is the transaction table where it stores monetary transaction data as append only data. The "columns" represent the set of all column name and their properties of the relational OLTP table. Users are expected to provide underlying temporal database engine supported data types, if temporal database use different relational database engine other than OLTP database engine type/vendor. The "indexes" represent the set of all indexes in current OLTP database table. Users should use "PRIMARY_KEY" index, only for the main tables' primary key. All other indexes are optional and if user provides them they should mark as "INDEX".

```yaml
tables:
  - tableName: Transaction
    tableType: TEMPORAL
    isMainTable: true
    columns:
      - columnName: txn_id
        isNullable: false
        dataType: INT
      - columnName: txn_amt
        isNullable: false
        dataType: DOUBLE
      - columnName: orig_acc
        isNullable: false
        dataType: INT
      - columnName: bene_acc
        isNullable: false
        dataType: INT
      - columnName: txn_timestamp
        isNullable: false
        dataType: DATETIME
      - columnName: txn_type_id
        isNullable: false
        dataType: INT
    indexes:
      - indexType: PRIMARY KEY
        indexName: txn_id_idx
        columnName: txn_id
      - indexType: INDEX
        indexName: wallet_id_a
        columnName: orig_acc
      - indexType: INDEX
        indexName: wallet_id_b
        columnName: bene_acc
      - indexType: INDEX
        indexName: txn_type_id
        columnName: txn_type_id
```

*Figure 4.2: Temporal Schema Mapping For Transaction Table*

### 4.3.3 Database Generator

The responsibility of database generator is to take the mapped schema object and convert it into temporal database and implement in the preconfigured database server.

## 4.4 Extendibility of Design

Micro temporal database generator currently supports to convert OLTP database to another MySQL based temporal database. But it supports extending this tool to any kind of database using template design pattern. Figure 4.3 represent the extendibility of the database generator tool using template design pattern.

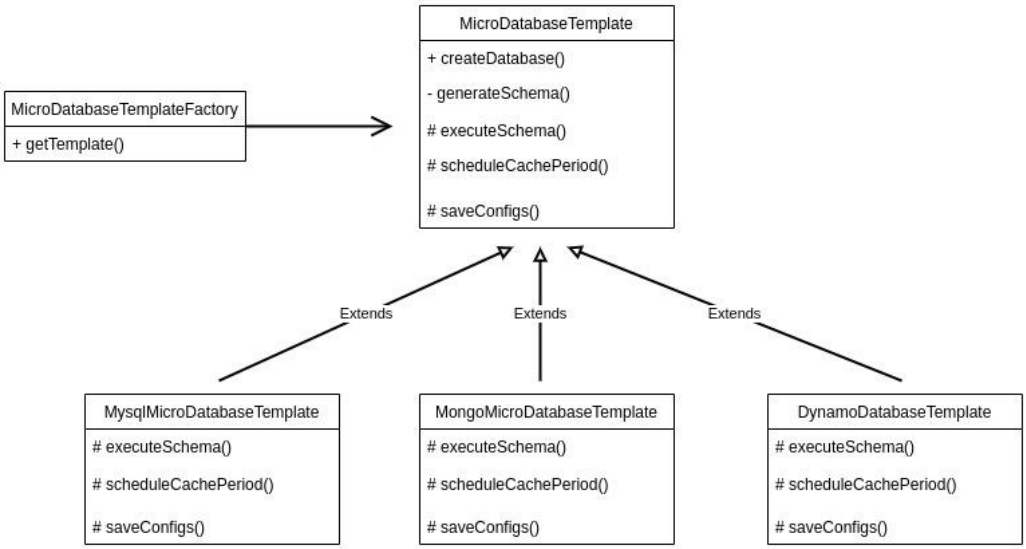


*Figure 4.3: Extendibility of Micro Database Generator Using Template Method Pattern*

Figure 4.4 shows the API contract for the above design. Since the source code of this project is expected to publish under open source license, future developers can use this contract to extend this tool support to implement different database vendors' products without changing the overall architectural design.

```java
public abstract class MicroDatabaseTemplate {

    /**
     * <p>
     * The main responsibility of this method to convert the OLTP to temporal mapping into target database schema.
     * Current implementation work as default behavior.It converts temporal mapping to SQL DDL statements and save
     * them in outputs directory .
     * </p>
     * <p>
     * <b>Hint:</b> For NOSQL databases : Generate index documents or collection mappings based on underlying
     * database engine.
     * </p>
     *
     * @param databaseName   temporal database name
     * @param databaseSchema schema mapping object
     */
    void generateSchema(String databaseName, DatabaseSchema databaseSchema) {

    }

    /**
     * <p>
     * The main intention of this method to connect to database server and create database and
     * implement the  previously (@code generateSchema step) generated schema in it.
     * </p>
     * <p>
     * <b>Hint:</b> For NOSQL databases : Create indexes or collection etc.
     * </p>
     *
     * @param databaseName temporal database name
     */
    abstract void executeSchema(String databaseName);

    /**
     * <p>The main intention of this method to provide a mechanism to  maintain data caching time period.
     * Since the different database engine support different mechanisms, users need to provide their preference
     * implementation.
     * </p>
     * <p>
     * Ex:
     *      <ul>
     *          <li>Database Events  in MySql</li>
     *          <li>Database triggers</li>
     *          <li>External crone triggers in AWS EventBridge</li>
     *          <li>NOSQL databases has TTL</li>
     *      </ul>
     * </p>
     *
     * @param databaseName temporal database name
     * @param cachePeriod  cache time period in days
     * @param tableNames   temporal table mapping
     */
    abstract void scheduleCachePeriod(String databaseName, int cachePeriod, List < String > tableNames);

    /**
     * Save configurations in a file system or database for later use
     */
    abstract void saveConfigs();
```

*Figure 4.4: Micro Database Generator API Contract*

## 4.5 Model Validation & Testing

Model validation process is use to confirm the generated micro databases has capability to produce same OLTP results under simulated transaction dataset. The overall process consist of four steps. They are:

1. Generate Micro Temporal Databases
2. Data Design
3. Data Preprocessing & Loading
4. Scoring Rule Design
5. Model Validation & Testing

### 4.5.1 Generate Micro Temporal Databases

As the 1st step of the validation process, we need to generate the micro temporal databases using the automated micro temporal database generation tool. The automated tool expects to have the properly mapped YAML file based on the guidelines provided in Appendix C. Then we have used the provided RESTful web API to generate temporal databases for 7 days,14 days, 28 days, data cache time periods.

Figure 4.1 shows the postman request & response to generate temporal database with 7 day data caching period.

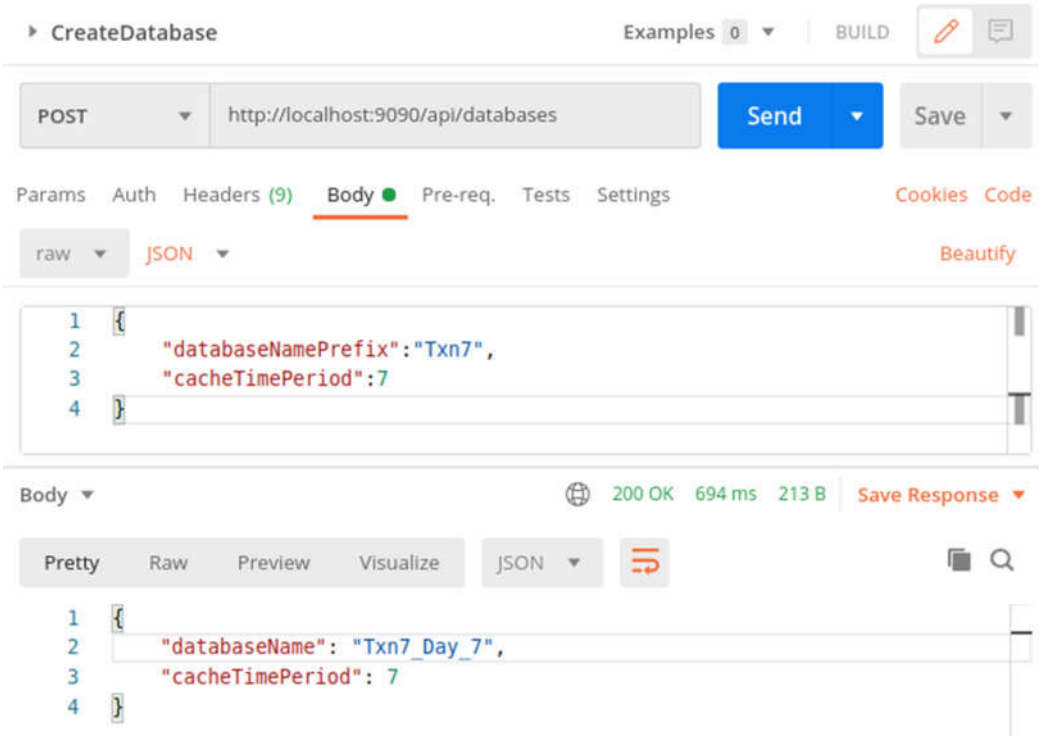


*Figure 4.1: Generate Temporal Database Sample Request & Response*

### 4.5.2 Data Design

Real world data takes a vital palace when comes to model validation phase. But publicly availability of transactional data is very limited (Refer: 2.8 Problem of Real World Transaction Data).So we have used the **IBM's** multi agent based, simulated transaction generator tool (**AMLSim**) to generate simulated monetary transactions.

According to the AMLSim tool, users have flexibility to configure several parameters including: supported transaction types, account types and account features, suspicious pattern types, transaction generation time period etc. We have modified the transactionTypes.csv, accounts.csv and altertPatterns.csv files in "1K" parameter files in AMLSim project. Other parameters kept as default values.

AMLSim Transaction Type Configuration

Table 4.1 shows the contents of transaction type configuration file

*Table 4.1: Transaction Types Configuration Details*

| Type | Frequency |
|---|---|
| TRANSFER | 1 |
| PAYMENT | 1 |
| CASH_IN | 1 |
| CASH_OUT | 1 |

AMLSim Account Configuration

Table 4.2 shows the contents of account configuration file. It has configured to generate 1446 accounts belongs to business types of:

- Individual Account
- Merchant Account
- Utility Account

*Table 4.2: Account Details Configuration*

| count | min_balance | max_balance | country | business_type | model | bank_id |
|---|---|---|---|---|---|---|
| 200 | 50000 | 100000 | US | I | 1 | Bank |

| | | | | | | |
|---|---|---|---|---|---|---|
| 200 | 50000 | 100000 | US | I | 2 | Bank |
| 200 | 50000 | 100000 | US | I | 3 | Bank |
| 200 | 50000 | 100000 | US | I | 4 | Bank |
| 200 | 50000 | 100000 | US | I | 4 | Bank |
| 200 | 50000 | 100000 | US | I | 5 | Bank |
| 200 | 50000 | 100000 | US | M | 1 | Bank |
| 46 | 50000 | 100000 | US | U | 2 | Bank |

AMLSim Alert Pattern Configuration

Table 4.3 shows the contents of alert pattern configuration file. We have configured it to generate transactions to cover: fan_in , fan_out, scatter_gather, gather_scatter AML topologies.

*Table 4.3: Alert Pattern Configuration*

| count | type | schedule_id | min_accounts | max_accounts | min_amount | max_amount | min_period | max_period |
|---|---|---|---|---|---|---|---|---|
| 3 | fain_in | 2 | 10 | 15 | 100 | 200 | 1 | 5 |
| 3 | fan_out | 2 | 10 | 15 | 100 | 200 | 5 | 20 |
| 2 | scatter_gather | 0 | 10 | 20 | 100 | 200 | 5 | 10 |
| 3 | scatter_gather | 0 | 15 | 30 | 100 | 200 | 5 | 30 |
| 2 | gather_scatter | 0 | 10 | 20 | 100 | 200 | 5 | 10 |

AMLSim simulator tool ran to cover 720 days and it was generated 379158 transactional records using 1445 customer accounts. The same data generation process ran over 5 rounds with random seed to generate the five different data sets. Figure 4.2 & figure 4.3 shows the first few rows of “account.csv” & “transaction.csv” files for the 1st round of data generation, based on the above configurations.

| | acct_id | dsply_nm | type | acct_stat | acct_rptng_crncy | prior_sar_count | branch_id | open_dt | close_dt | initial_deposit | ... | street_addr | city |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | C_0 | I | A | USD | False | 1 | 2022-01-01T00:00:00Z | 4759-11-29T00:00:00Z | 92221.09 | ... | 48764 Howard Forge Apt. 421 | Vanessaside |
| 1 | 1 | C_1 | I | A | USD | False | 1 | 2022-01-01T00:00:00Z | 4759-11-29T00:00:00Z | 87897.72 | ... | 387 Grimes Green Apt. 801 | Pagetown |
| 2 | 2 | C_2 | I | A | USD | False | 1 | 2022-01-01T00:00:00Z | 4759-11-29T00:00:00Z | 71028.58 | ... | 15871 Arnold Squares Apt. 858 | Port Carrie |
| 3 | 3 | C_3 | I | A | USD | True | 1 | 2022-01-01T00:00:00Z | 4759-11-29T00:00:00Z | 62945.84 | ... | 471 Erika Curve | North Megan |
| 4 | 4 | C_4 | I | A | USD | False | 1 | 2022-01-01T00:00:00Z | 4759-11-29T00:00:00Z | 75563.74 | ... | 477 Miller Ridge Apt. 795 | East Allen |

*Figure 4.2: Generated Accounts*

| | tran_id | orig_acct | bene_acct | tx_type | base_amt | tran_timestamp | is_sar | alert_id |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 599 | 4 | PAYMENT | 201.85 | 2022-01-01T00:00:00Z | False | -1 |
| 1 | 2 | 161 | 30 | PAYMENT | 541.37 | 2022-01-01T00:00:00Z | False | -1 |
| 2 | 3 | 1002 | 53 | PAYMENT | 374.96 | 2022-01-01T00:00:00Z | False | -1 |
| 3 | 4 | 1201 | 49 | CASH_IN | 559.50 | 2022-01-01T00:00:00Z | False | -1 |
| 4 | 5 | 314 | 18 | TRANSFER | 318.26 | 2022-01-01T00:00:00Z | False | -1 |

*Figure 4.3: Generated Transactions*

### 4.5.3 Data Preprocessing & Loading

Figure 4.4 shows our OLTP database schema model and figure 4.5 shows the underlying converted temporal schema model. They have common attributes as well as **several distinct attributes and features**. Specially, there are some additional timestamp related attributes in **all the temporal tables** in temporal databases. So we need to prepare two separate datasets for those two types of databases. Further, AMLSim generated data is in CSV format and they are not in the form of ready to insert to any of database. So, in the preprocessing step, AMLSim generated data is preprocessed to match OLTP and temporal database schemas. Further, it will generate SQL "INSERT" script files which can instantly install on OLTP & temporal databases using a SQL client software.

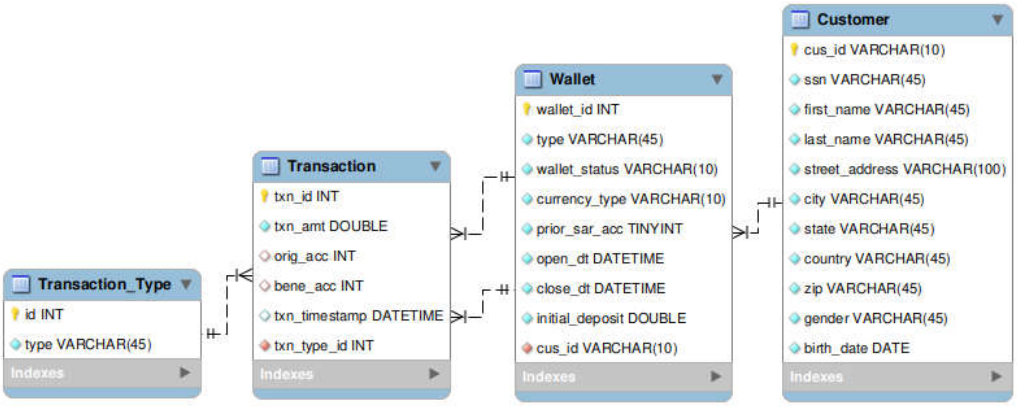


*Figure 4.4: OLTP Database Schema Model*

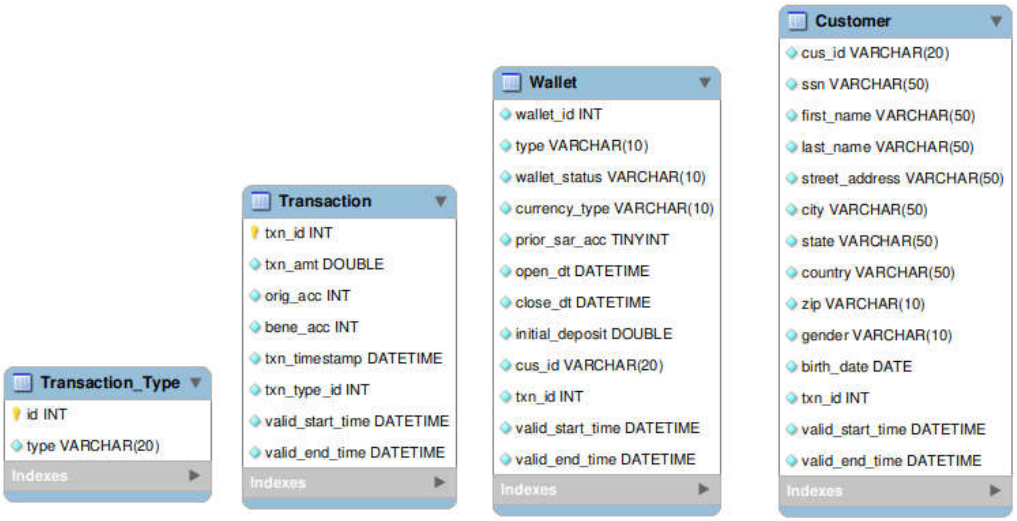


*Figure 4.5: Temporal Database Schema Model*

Appendix D gives more details steps for the “preprocessing and loading” phase using Jupiter notebooks and python scripts.

### 4.5.4 Scoring Rule Design

There is no restriction to use same schema architecture for both OLTP databases and temporal databases. So the temporal database schemas can be in completely different form, than OLTP design. Due to this fact, most of existing scoring rules might need to modify their algorithms to meet the newer temporal schema designs in most of real world practical situations.

**The transaction scoring rule design process is out of this research scope** and we have designed & implemented the sample of four scoring rules which suit to point to the both of OLTP and TEMPORAL databases for sake of proofing the model validity.

**Scoring Rule 1 (SC1):** Identify accounts with unusual frequency of payments for good or service for a single merchant (when transaction type is PAYMENT)
**Scoring Rule 2 (SC2):** Identify accounts with unusual frequency of fund receive from same or distinct accounts (when transaction type is TRANSFER)
**Scoring Rule 3 (SC3):** Identify accounts who transfer unusual frequency of small amount of funds to set of different (or same) accounts over period of time
**Scoring Rule 4 (SC4):** Identify accounts who practice the CASH_IN fallowed by CASH_OUT in unusual frequency of time over specified time period

### 4.5.5 Model Validate

The micro database generator tool provides a RESTful web service endpoint for each of above four scoring rules to trigger & find suspicious account details. These endpoints take the "databaseName" as external user input and scoring rule will evaluate against given database.

In order to prove the micro temporal model validity, we ran the same scoring rule against OLTP database and three of micro temporal databases (with data cache time period of 7 days, 14 days, 28 days) and check the similarity of results with respect to relevant data cache time periods of each database.

Note: All simulated transactions are generated for the future periods because temporal databases automatically removes older records based on their configured cache time periods. So we have generated simulated transactions for the period of "2022-01-01 00:00:00" to "2023-12-21 00:00:00".

**Scoring Rule 1: Evaluation Results**

Scoring Rule 1 Evaluate over "OLTP Database"


```
{
	"suspiciousReports": [{
			"dateFrom": "2023-12-21 00:00:00",
			"dateTo": "2023-12-14 00:00:00",
			"suspicions": [{
				"accountId": 1352,
				"frequency": 7,
				"transactions": [result hide to keep clarity]
			}]
		},
		{
			"dateFrom": "2023-12-14 00:00:00",
			"dateTo": "2023-12-07 00:00:00",
			"suspicions": [{
				"accountId": 1352,
				"frequency": 7,
				"transactions": [result hide to keep clarity]
			}]
		},
		{
			"dateFrom": "2023-12-07 00:00:00",
			"dateTo": "2023-11-30 00:00:00",
			"suspicions": [{
				"accountId": 1352,
				"frequency": 7,
				"transactions": [result hide to keep clarity]
			}]
		},
		{
			"dateFrom": "2023-11-30 00:00:00",
			"dateTo": "2023-11-23 00:00:00",
			"suspicions": [{
				"accountId": 1352,
				"frequency": 7,
				"transactions": [result hide to keep clarity]
			}]
		}
	]
}
```


According to the above results, SC1 has identified account number 1357 as suspicious for each evaluation week. The financial investigators will identify this suspicious record at the end of each month because this rule is actually evaluated in monthly basis (to reduce OLTP database load) even it is expected to evaluate on weekly basis.

Scoring Rule 1 Evaluate over Temporal “Txn28_Day_28 Database”


```
{
        "suspiciousReports": [{
                        "dateFrom": "2023-12-21 00:00:00",
                        "dateTo": "2023-12-14 00:00:00",
                        "suspicions": [{
                                "accountId": 1352,
                                "frequency": 7,
                                "transactions": [result hide to keep clarity]
                        }]
                },
                {
                        "dateFrom": "2023-12-14 00:00:00",
                        "dateTo": "2023-12-07 00:00:00",
                        "suspicions": [{
                                "accountId": 1352,
                                "frequency": 7,
                                "transactions": [result hide to keep clarity]
                        }]
                },
                {
                        "dateFrom": "2023-12-07 00:00:00",
                        "dateTo": "2023-11-30 00:00:00",
                        "suspicions": [{
                                "accountId": 1352,
                                "frequency": 7,
                                "transactions": [result hide to keep clarity]
                        }]
                },
                {
                        "dateFrom": "2023-11-30 00:00:00",
                        "dateTo": "2023-11-23 00:00:00",
                        "suspicions": [{
                                "accountId": 1352,
                                "frequency": 7,
                                "transactions": [result hide to keep clarity]
                        }]
                }
        ]
}
```


According to the above results, SC1 has identified account number 1357 as suspicious for each evaluation week. This proves the temporal database with 28 cache time period can produce equally results which it produced with OLTP database.

Scoring Rule 1 Evaluate over Temporal “Txn14_Day_14 Database”


```
{
	"suspiciousReports": [{
			"dateFrom": "2023-12-21 00:00:00",
			"dateTo": "2023-12-14 00:00:00",
			"suspicions": [{
				"accountId": 1352,
				"frequency": 7,
				"transactions": [

				]
			}]
		},
		{
			"dateFrom": "2023-12-14 00:00:00",
			"dateTo": "2023-12-07 00:00:00",
			"suspicions": [{
				"accountId": 1352,
				"frequency": 7,
				"transactions": [

				]
			}]
		},
		{
			"dateFrom": "2023-12-07 00:00:00",
			"dateTo": "2023-11-30 00:00:00",
			"suspicions": []
		},
		{
			"dateFrom": "2023-11-30 00:00:00",
			"dateTo": "2023-11-23 00:00:00",
			"suspicions": []
		}
	]
}
```


According to the above results, SC1 has identified account number 1357 as suspicious for 3rd and 4th evaluation weeks. This proves the temporal database with 14 day cache time period can produce respective equally results for the last two weeks which it produced with OLTP database.

Scoring Rule 1 Evaluate over Temporal “Txn7_Day_7 Database”


```
{
	"suspiciousReports": [{
			"dateFrom": "2023-12-21 00:00:00",
```

```
                        "dateTo": "2023-12-14 00:00:00",
                        "suspicions": [{
                                "accountId": 1352,
                                "frequency": 7,
                                "transactions":[result hide to keep clarity]
                        }]
                },
                {
                        "dateFrom": "2023-12-14 00:00:00",
                        "dateTo": "2023-12-07 00:00:00",
                        "suspicions": []
                },
                {
                        "dateFrom": "2023-12-07 00:00:00",
                        "dateTo": "2023-11-30 00:00:00",
                        "suspicions": []
                },
                {
                        "dateFrom": "2023-11-30 00:00:00",
                        "dateTo": "2023-11-23 00:00:00",
                        "suspicions": []
                }
        ]
}
```


According to the above results, SC1 has identified account number 1357 as suspicious for the 4th evaluation week. This proves the temporal database with 7 day cache time period can produce respective equally results for the last week (4th week) which it produced with OLTP database.

**Scoring Rule 1: Summary**

*Table 4.4: Scoring Rule 1 - Experiment 1 - Summary*

| Scoring Rule | Database Name | Expected Execution Period | Actual Execution Period | Result (AccountId:Frequency) |
|---|---|---|---|---|
| SC 1 | OLTP | 7 Day | 28 Day | 4th Week: 1352:7 |
| | | | | 3rd Week: 1352:7 |
| | | | | 2nd Week: 1352:7 |
| | | | | 1st Week: 1352:7 |
| | Tnx28_Day_28 | 7 Day | 28 Day | 4th Week: 1352:7 |

| | | | | |
|---|---|---|---|---|
| | | | | 3rd Week: 1352:7 |
| | | | | 2nd Week: 1352:7 |
| | | | | 1st Week: 1352:7 |
| | Tnx14_Day_14 | 7 Day | 14 Day | 4th Week: 1352:7 |
| | | | | 3rd Week: 1352:7 |
| | Tnx7_Day_7 | 7 Day | 7 Day | 4th Week : 1352:7 |

Table 4.4 shows summary results when SC1 pointed to above four database. Based on that:

- SC1 identified the account number 1352 as suspicious from the 1st week results when it pointed to OLTP database. Same result gave when SC1 pointed to Txn28_Day_28 database because this pattern target to cover a moth period and we mark a month as 28 days (7x 4).
- Txn14_Day_14 produces same result for last 2 weeks' time period. Results for first two weeks are empty because this database keep data only for past 14 day period.
- Txn7_day_7 produces same result for last 1 week time period. Results for other three weeks are empty because this database keep data only for past 7 day period.

Based on the overall above results, temporal databases produces the same OLTP results base on its temporal time period. So by using temporal design, **we can identify this type of patterns in three times faster than evaluating them on monthly basis by pointing OLTP systems.**

**Scoring Rule 2: Evaluation Results**

Scoring Rule 2 Evaluate over "OLTP Database"


```
{
       "suspiciousReports": [
         {
```

```
                        "dateFrom": "2023-12-21 00:00:00",
                        "dateTo": "2023-12-14 00:00:00",
                        "suspicions": []
                },
                {
                        "dateFrom": "2023-12-14 00:00:00",
                        "dateTo": "2023-12-07 00:00:00",
                        "suspicions": []
                },
                {
                        "dateFrom": "2023-12-07 00:00:00",
                        "dateTo": "2023-11-30 00:00:00",
                        "suspicions": [{
                                "accountId": 27,
                                "frequency": 11,
                                "transactions": [results hide to keep clarity]
                        },
                        {
                                "accountId": 21,
                                "frequency": 11,
                                "transactions": [results hide to keep clarity]
                        }
                        ]
                },
                {
                        "dateFrom": "2023-11-30 00:00:00",
                        "dateTo": "2023-11-23 00:00:00",
                        "suspicions": [{
                                "accountId": 25,
                                "frequency": 11,
                                "transactions": [results hide to keep clarity]
                        }]
                }
        ]
}
```


According to the above results, SC2 has not recorded any suspicious records for 3rd and 4th evaluation weeks.SC2 identified account number 21 & 27 as suspicious for the 2nd week of evaluation results. Account number 25 was recorded for the period of first 7 days. The financial investigators will identify this suspicious record at the end of each month because this rule is actually evaluated in monthly basis (to reduce OLTP database load) even it is expected to evaluate on weekly basis.

Scoring Rule 2 Evaluate over Temporal "Txn28_Day_28 Database"


```
{
        "suspiciousReports": [{
                        "dateFrom": "2023-12-21 00:00:00",
                        "dateTo": "2023-12-14 00:00:00",
                        "suspicions": []
                },
```

```
                {
                        "dateFrom": "2023-12-14 00:00:00",
                        "dateTo": "2023-12-07 00:00:00",
                        "suspicions": []
                },
                {
                        "dateFrom": "2023-12-07 00:00:00",
                        "dateTo": "2023-11-30 00:00:00",
                        "suspicions": [{
                                "accountId": 27,
                                "frequency": 11,
                                "transactions": [results hide to keep clarity]
                                },
                                {
                                "accountId": 21,
                                "frequency": 11,
                                "transactions": [results hide to keep clarity]
                                }
                        ]
                },
                {
                        "dateFrom": "2023-11-30 00:00:00",
                        "dateTo": "2023-11-23 00:00:00",
                        "suspicions": [{
                                "accountId": 25,
                                "frequency": 11,
                                "transactions": [results hide to keep clarity]
                        }]
                }
        ]
}
```


According to the above results, SC2 has not record any suspicious records for 3rd and 4th week of the month.SC2 identified account number 27 & 21 as suspicious in the 2nd week report and account number 25 in 1st week report. This proves the temporal database with 28 cache time period can produce equally results which it produced with OLTP database.

Scoring Rule 2 Evaluate over Temporal "Txn14_Day 14 Database"


```
{
        "suspiciousReports": [{
                        "dateFrom": "2023-12-21 00:00:00",
                        "dateTo": "2023-12-14 00:00:00",
                        "suspicions": []
                },
                {
```

```
                        "dateFrom": "2023-12-14 00:00:00",
                        "dateTo": "2023-12-07 00:00:00",
                        "suspicions": []
                },
                {
                        "dateFrom": "2023-12-07 00:00:00",
                        "dateTo": "2023-11-30 00:00:00",
                        "suspicions": []
                },
                {
                        "dateFrom": "2023-11-30 00:00:00",
                        "dateTo": "2023-11-23 00:00:00",
                        "suspicions": []
                }
        ]
}
```


According to the above results, SC2 has not record any suspicious records for 3rd and 4th week of the month. This proves the temporal database with 14 day cache time period can produce respective equally results for the last two weeks which it produced with OLTP database.

Scoring Rule 2 Evaluate over Temporal "Txn7_Day_7 Database"


```
{
        "suspiciousReports": [{
                        "dateFrom": "2023-12-21 00:00:00",
                        "dateTo": "2023-12-14 00:00:00",
                        "suspicions": []
                },
                {
                        "dateFrom": "2023-12-14 00:00:00",
                        "dateTo": "2023-12-07 00:00:00",
                        "suspicions": []
                },
                {
                        "dateFrom": "2023-12-07 00:00:00",
                        "dateTo": "2023-11-30 00:00:00",
                        "suspicions": []
                },
                {
                        "dateFrom": "2023-11-30 00:00:00",
                        "dateTo": "2023-11-23 00:00:00",
                        "suspicions": []
                }
        ]
}
```

According to the above results, SC2 has not record any suspicious records for last week (4th week) of the month. This proves the temporal database with 7 day cache time period can produce respective equally results for the last weeks which it produced with OLTP database.

**Scoring Rule 2: Summary**

*Table 4.5: Scoring Rule 2 - Experiment 1 - Summary*

| Scoring Rule | Database Name | Expected Execution Period | Actual Execution Period | Result (AccountId:Frequency) |
|---|---|---|---|---|
| SC 2 | OLTP | 7 Day | 28 Day | 4th Week:<br>Empty |
| | | | | 3rd Week:<br>Empty |
| | | | | 2nd Week:<br>21:11<br>27:11 |
| | | | | 1st Week:<br>25:11 |
| | Tnx28_Day_28 | 7 Day | 28 Day | 4th Week:<br>Empty |
| | | | | 3rd Week:<br>Empty |
| | | | | 2nd Week:<br>21:11<br>27:11 |
| | | | | 1st Week:<br>25:11 |
| | Tnx14_Day_14 | 7 Day | 14 Day | 4th Week:<br>Empty |
| | | | | 3rd Week:<br>Empty |
| | Tnx7_Day_7 | 7 Day | 7 Day | 4th Week :<br>Empty |

Table 4.5 shows summary results when SC2 pointed to above four database. Based on that:

- SC2 identified the account number 21 & 27 as suspicious for 2nd week result of OLTP pointed database. It produced the account number 25 as suspicious for the 1st week evaluation results that pointed to same OLTP database. Same

result gave when SC2 pointed to Txn28_Day_28 database because this pattern cover a moth period and we mark a month as 28 days (7x 4).

- Txn14_Day_14 and Txn7_day_7 databases pointed evaluations has not identified any records relevant to their temporal time frames.
- Scoring rule pointed to Txn14_Day_14 database produce empty results for first two weeks because this database keep data only for the past 14 days period.
- Scoring rule pointed to Txn7_day_7 produces empty results for first three weeks because this database keep data only for last 7 day period.

Based on the overall above results, temporal databases produces the same OLTP results base on its temporal time period.

**Scoring Rule 3: Evaluation Results**

Scoring Rule 3 Evaluate over "OLTP Database"


```
{
	"suspiciousReports": [{
			"dateFrom": "2023-12-21 00:00:00",
			"dateTo": "2023-12-07 00:00:00",
			"suspicions": [{
				"accountId": 918,
				"frequency": 8,
				"transactions": [results hide to keep clarity]
			}]
		},
		{
			"dateFrom": "2023-12-07 00:00:00",
			"dateTo": "2023-11-23 00:00:00",
			"suspicions": []
		}
	]
}
```


According to the above results, SC3 has recorded account number 918 as suspicious for the period of 3rd & 4th week time frame. There are no any suspicious records for first two week time period. The financial investigators will identify this suspicious record at the end of each month because this rule is actually evaluated in monthly basis

(to reduce OLTP database load) even it is expected to evaluate on 14 day period basis (biweekly).

Scoring Rule 3 Evaluate over Temporal "Txn28_Day_28 Database"


```
{
        "suspiciousReports": [{
                        "dateFrom": "2023-12-21 00:00:00",
                        "dateTo": "2023-12-07 00:00:00",
                        "suspicions": [{
                                "accountId": 918,
                                "frequency": 8,
                                "transactions": [results hide to keep clarity]
                        }]
                },
                {
                        "dateFrom": "2023-12-07 00:00:00",
                        "dateTo": "2023-11-23 00:00:00",
                        "suspicions": []
                }
        ]
}
```


According to the above results, SC3 has recorded account number 918 as suspicious for the period of 3rd & 4th week time frame. There are no any suspicious records for first two week time period. This proves the temporal database with 28 cache time period can produce equally results which it produced with OLTP database.

Scoring Rule 3 Evaluate over Temporal "Txn14_Day_14 Database"


```
{
        "suspiciousReports": [{
                        "dateFrom": "2023-12-21 00:00:00",
                        "dateTo": "2023-12-07 00:00:00",
                        "suspicions": [{
                                "accountId": 918,
                                "frequency": 8,
                                "transactions": [results hide to keep clarity]
                        }]
                },
                {
                        "dateFrom": "2023-12-07 00:00:00",
                        "dateTo": "2023-11-23 00:00:00",
                        "suspicions": []
                }
        ]
}
```

According to the above results, SC3 has recorded account number 918 as suspicious for the period of 3rd & 4th week time frame. There are no any suspicious records for first two week time period. This proves the temporal database with 14 day cache time period can produce respective equally results for last 14 day period which it produced with OLTP database.

**Scoring Rule 3: Summary**

*Table 4.6: Scoring Rule 3 - Experiment 1 - Summary*

| Scoring Rule | Database Name | Expected Execution Period | Actual Execution Period | Result (AccountId:Frequency) |
|---|---|---|---|---|
| SC 3 | OLTP | 14 Day | 28 Day | 3rd & 4th Week: 918:8 |
| | | | | 1st & 2nd Week: Empty |
| | Tnx28_Day_28 | 14 Day | 28 Day | 3rd & 4th Week: 918:8 |
| | | | | 1st & 2nd Week: Empty |
| | Tnx14_Day_14 | 14 Day | 14 Day | 3rd & 4th Week: 918:8 |
| | | | | 1st & 2nd Week: Empty |
| | Tnx7_Day_7 | NA | NA | NA |

Table 4.6 shows the summary results when SC3 pointed to above three databases. Based on that:

- SC3 has identified the account number 918 as suspicious for last two weeks period evaluation results (between 3rd & 4th week) when it pointed to all three applicable databases.
- There is no any suspicious records reported for 1st & 2nd week period when SC3 pointed to OLTP database or Tnx28_Day_28 temporal database.

Based on the overall above results, temporal databases produces the same OLTP results base on its temporal time period.

**Scoring Rule 4: Evaluation Results**

Scoring Rule 4 Evaluate over “OLTP Database”


```
{
   "dateFrom": "2023-12-21 00:00:00",
   "dateTo": "2023-12-14 00:00:00",
   "suspicions": [{
         "accountId": 140,
         "frequency": 2
      }, {
         "accountId": 23,
         "frequency": 2
      },{
         "accountId": 281,
         "frequency": 2
      },{
         "accountId": 28,
         "frequency": 2
      },{
         "accountId": 33,
         "frequency": 2
      },{
         "accountId": 35,
         "frequency": 2
      },{
         "accountId": 297,
         "frequency": 2
      },{
         "accountId": 192,
         "frequency": 2
      },{
         "accountId": 200,
         "frequency": 2
      },{
         "accountId": 346,
         "frequency": 2
      },{
         "accountId": 91,
         "frequency": 2
      },{
         "accountId": 94,
         "frequency": 2
      },{
         "accountId": 994,
         "frequency": 2
      },{
         "accountId": 104,
         "frequency": 2
      }
   ]
}
```


According to the above results, SC4 has recorded account numbers of 140 , 23 , 281 , 28, 33 , 35 , 297 , 192 , 200 , 346 as suspicious for last 7 day time period.

Scoring Rule 4 Evaluate over Temporal “Txn28_Day_28 Database”


```
{
  "dateFrom": "2023-12-21 00:00:00",
  "dateTo": "2023-12-14 00:00:00",
  "suspicions": [{
        "accountId": 140,
        "frequency": 2
    }, {
        "accountId": 23,
        "frequency": 2
    },{
        "accountId": 281,
        "frequency": 2
    },{
        "accountId": 28,
        "frequency": 2
    },{
        "accountId": 33,
        "frequency": 2
    },{
        "accountId": 35,
        "frequency": 2
    },{
        "accountId": 297,
        "frequency": 2
    },{
        "accountId": 192,
        "frequency": 2
    },{
        "accountId": 200,
        "frequency": 2
    },{
        "accountId": 346,
        "frequency": 2
    },{
        "accountId": 91,
        "frequency": 2
    },{
        "accountId": 94,
        "frequency": 2
    },{
        "accountId": 994,
        "frequency": 2
    },{
        "accountId": 104,
        "frequency": 2
    }
  ]
}
```


According to the above results, SC4 has recorded account numbers of 140 , 23 , 281 , 28, 33 , 35 , 297 , 192 , 200 , 346 as suspicious for last 7 day time period.

Scoring Rule 4 Evaluate over Temporal “Txn14_Day_14 Database”


```
{
  "dateFrom": "2023-12-21 00:00:00",
  "dateTo": "2023-12-14 00:00:00",
  "suspicions": [{
        "accountId": 140,
        "frequency": 2
    }, {
        "accountId": 23,
        "frequency": 2
    },{
        "accountId": 281,
        "frequency": 2
    },{
        "accountId": 28,
        "frequency": 2
    },{
        "accountId": 33,
        "frequency": 2
    },{
        "accountId": 35,
        "frequency": 2
    },{
        "accountId": 297,
        "frequency": 2
    },{
        "accountId": 192,
        "frequency": 2
    },{
        "accountId": 200,
        "frequency": 2
    },{
        "accountId": 346,
        "frequency": 2
    },{
        "accountId": 91,
        "frequency": 2
    },{
        "accountId": 94,
        "frequency": 2
    },{
        "accountId": 994,
        "frequency": 2
    },{
        "accountId": 104,
        "frequency": 2
    }
  ]
}
```


According to the above results, SC4 has recorded account numbers of 140 , 23 , 281 , 28, 33 , 35 , 297 , 192 , 200 , 346 as suspicious for last 7 day time period.

Scoring Rule 4 Evaluate over Temporal “Txn7_Day_7 Database”


```
{
  "dateFrom": "2023-12-21 00:00:00",
  "dateTo": "2023-12-14 00:00:00",
  "suspicions": [{
        "accountId": 140,
        "frequency": 2
    }, {
        "accountId": 23,
        "frequency": 2
    },{
        "accountId": 281,
        "frequency": 2
    },{
        "accountId": 28,
        "frequency": 2
    },{
        "accountId": 33,
        "frequency": 2
    },{
        "accountId": 35,
        "frequency": 2
    },{
        "accountId": 297,
        "frequency": 2
    },{
        "accountId": 192,
        "frequency": 2
    },{
        "accountId": 200,
        "frequency": 2
    },{
        "accountId": 346,
        "frequency": 2
    },{
        "accountId": 91,
        "frequency": 2
    },{
        "accountId": 94,
        "frequency": 2
    },{
        "accountId": 994,
        "frequency": 2
    },{
        "accountId": 104,
        "frequency": 2
    }
  ]
}
```


According to the above results, SC4 has recorded account numbers of 140 , 23 , 281 , 28, 33 , 35 , 297 , 192 , 200 , 346 as suspicious for last 7 day time period.

**Scoring Rule 4: Summary**

*Table 4.7: Scoring Rule 4 - Experiment 1 - Summary*

| Scoring Rule | Database Name | Expected Execution Period | Actual Execution Period | Result (AccountId:Frequency) |
|---|---|---|---|---|
| SC 4 | OLTP | 7 Day | 7 Day | 4th Week: 140:2 , 23:2 , 281:2 , 28:2, 33:2 , 35:2 , 297:2 , 192:2 , 200:2 , 346:2 , 91:2 , 94:2 , 994:2 , 104:2 |
| | Tnx28_Day_28 | 7 Day | 7 Day | 4th Week: 140:2 , 23:2 , 281:2 , 28:2, 33:2 , 35:2 , 297:2 , 192:2 , 200:2 , 346:2 , 91:2 , 94:2 , 994:2 , 104:2 |
| | Tnx14_Day_14 | 7 Day | 7 Day | 4th Week: 140:2 , 23:2 , 281:2 , 28:2, 33:2 , 35:2 , 297:2 , 192:2 , 200:2 , 346:2 , 91:2 , 94:2 , 994:2 , 104:2 |
| | Tnx7_Day_7 | 7 Day | 7 Day | 4th Week: 140:2 , 23:2 , 281:2 , 28:2, 33:2 , 35:2 , 297:2 , 192:2 , 200:2 , 346:2 , 91:2 , 94:2 , 994:2 , 104:2 |

Table 4.7 shows the summary results when SC4 pointed to above four databases. Based on that:

- SC4 produces same results when it point to any of the above databases (OLTP or temporal databases)

Based on the overall above results, temporal databases produces the same OLTP results base on its temporal time period.

Special Note: We have run the above process for a series of five randomly generated transactional datasets and all of them produced a similar result to the above. A summary of all test case results has attached to appendix E.

## 4.6 Summary

First sections of this chapter described the necessity of automated tool for micro temporal database creation for industrial usage. Then the chapter introduced the micro temporal database generator tool and its capabilities. More especially it describe the extendibility of tool to generate any type of temporal database using its given template contract and open source licensing.

Last sections of the chapter described the temporal database architecture model validation process & associated results.

# 5 CHAPTER 5 – Discussion

## 5.1 Study Outcome

Micro Data Model Architecture support to mitigate financial organizations' risk by allowing scoring rule engines to produce suspicious transaction report in early stage than it produces results with traditional OLTP databases. The loosely coupled architectural attribute gives users to select any kind of databases technology as temporal databases. So the users have full freedom to use modern database technologies to optimize the data analysis process. Further, it supports optimizing each individual temporal database based on the requirements of each individual rule group (very similar to microservices) which was not allowed to do when working with OLTP databases. Additionally, this architecture enables business experts to write complex queries on top of micro databases which was completely blocked in traditional OLTP methodology due to the extra burden on OLTP databases cause database performance issues.

Traditional scoring rule engines had scoring rule scheduling issues when OLTP database call maintenance jobs because the whole set of scoring rules was pointed to a single data source. So there were cancellations and rescheduling take a place. So the money launders had extra time to continue their operations without any disturbs. But the micro database modelling architecture introduces a set of temporal databases instead of the single data source. So, there is a lesser chance to cancel the overall scoring rule evaluation process at a given point of time. Hence it provides further risk reduction compared to the traditional approaches.

The micro data model architecture consider the financial transaction risk when design & model the micro databases. This feature guided the technology experts to work with business experts to produce most optimized data model by combining technical and business expert knowledge together.

## 5.2 Experimental Limitations

This research has fallowing limitations:

- Unavailability of real world transactional data cause to use synthetic transactional data for model validations
- Unavailability of suitable scoring rule engines cause lower test coverage
- Performance and cost evaluation is excluded to reduce research scope to complete the research in the due time frame

## 5.3 Future Work

Since performance and cost aspects are excluded at this level, I need to cover that part before introducing this architectural model to real industrial use.

Micro data model architecture is live on the middle part of full data analysis architecture. So I hope to build a prototype version of full architecture by embedding this component for industrial presentations. That can make aware of this concept in the real world industry.

The current implementation provides the facility to generate only the MySQL based temporal databases. Since we have already defined the API contract I am expected to implement this contract for few more different databases including NoSQL databases.

## 5.4 Summary

This chapter discussed the research outcomes, approach limitations and future work plans related to extending of this idea.

# 6 REFERENCES

[1]Parliament Of the Democratic Socialist Republic Of Sri Lanka, "Prevention Of Money Laundering Act , No. 5 of 2006", Central Bank Of Sri Lanka, 2006. [Online]. Available: http://fiusrilanka.gov.lk/docs/ACTs/PMLA/Money_Laundering_Act_2006-5_(English).pdf. [Accessed: 17- Dec- 2019].

[2]"Anti-money laundering and counter terrorist financing", European Commission - European Commission, 2020. [Online]. Available: https://ec.europa.eu/info/business-economy-euro/banking-and-finance/financial-supervision-and-risk-management/anti-money-laundering-and-counter-terrorist-financing_en. [Accessed: - 12 Aug- 2019].

[3]"What is Money Laundering? | ICA", Int-comp.org, 2020. [Online]. Available: https://www.int-comp.org/careers/your-career-in-aml/what-is-money-laundering/. [Accessed: 01- Feb- 2020].

[4]"Money laundering", En.wikipedia.org, 2020. [Online]. Available: https://en.wikipedia.org/wiki/Money_laundering. [Accessed: 06- Aug- 2019].

[5]"Types of Suspicious Activities or Transactions – Financial Intelligence Unit", Fiubelize.org, 2019. [Online]. Available: http://fiubelize.org/types-of-suspicious-activities-or-transactions/. [Accessed: 1- Sep- 2019].

[6]Bolton, Richard & Hand, David. (2002). Statistical Fraud Detection: A Review. Statistical Science. 17. 10.1214/ss/1042727940.

[7]A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi and G. Bontempi, "Credit Card Fraud Detection: A Realistic Modeling and a Novel Learning Strategy," in IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 8, pp. 3784-3797, Aug. 2018, doi: 10.1109/TNNLS.2017.2736643.

[8]Carcillo, Fabrizio & Dal Pozzolo, Andrea & Le Borgne, Yann-Aël & Caelen, Olivier & Mazzer, Yannis & Bontempi, Gianluca. (2017). SCARFF : a Scalable Framework for Streaming Credit Card Fraud Detection with Spark. Information Fusion. 41. 10.1016/j.inffus.2017.09.005.

[9]D. Lacmanovic, I. Lacmanovic and B. Markoski, "Mobile Banking - financial services technology," 2012 Proceedings of the 35th International Convention MIPRO, 2012, pp. 1451-1455.

[10]B. Burnham, "The Internet's Impact on Retail Banking", strategy+business, 2020. [Online]. Available: https://www.strategy-business.com/article/17575?gko=855fb. [Accessed: 20- May- 2020].

[11]Tănase, R & Şerbu, Răzvan. (2010). Operational risk and e-banking. Bulletin of the Transilvania University of Brasov. Series V : Economic Sciences. 3 (52).

[12]"UN Instruments", Imolin.org, 2020. [Online]. Available: https://www.imolin.org/imolin/courmaye.html. [Accessed: 20- May- 2020].

[13]"The 5 Most Pressing AML Challenges", Sentinels.ai, 2022. [Online]. Available: https://www.sentinels.ai/news/the-five-most-pressing-aml-challenges. [Accessed: 13- December- 2021].

[14]John, Samuel & Anele, C. & Okokpujie, Kennedy & Olajide, Funminiyi & Kennedy, Chinyere. (2016). Realtime Fraud Detection in the Banking Sector Using Data Mining Techniques/Algorithm. 1186-1191. 10.1109/CSCI.2016.0224.

[15]Sahin, Yusuf & Bulkan, Serol & Duman, Ekrem. (2013). A cost-sensitive decision tree approach for fraud detection. Expert Systems with Applications. 40. 5916–5923. 10.1016/j.eswa.2013.05.021.

[16]A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi and G. Bontempi, "Credit card fraud detection and concept-drift adaptation with delayed supervised information," 2015 International Joint Conference on Neural Networks (IJCNN), 2015, pp. 1-8, doi: 10.1109/IJCNN.2015.7280527.

[17]D. Malekian and M. R. Hashemi, "An adaptive profile based fraud detection framework for handling concept drift," 2013 10th International ISC Conference on Information Security and Cryptology (ISCISC), 2013, pp. 1-6, doi: 10.1109/ISCISC.2013.6767338.

[18]S. Makki, Z. Assaghir, Y. Taher, R. Haque, M. Hacid and H. Zeineddine, "An Experimental Study With Imbalanced Classification Approaches for Credit Card Fraud Detection," in IEEE Access, vol. 7, pp. 93010-93022, 2019, doi: 10.1109/ACCESS.2019.2927266.

[19]Dal Pozzolo, Andrea & Caelen, Olivier & Le Borgne, Yann-Aël & Waterschoot, Serge & Bontempi, Gianluca. (2014). Learned lessons in credit card fraud detection from a practitioner perspective. Expert Systems with Applications. 41. 4915–4928. 10.1016/j.eswa.2014.02.026.

[20]Bolton, Richard & Hand, David. (2001). Unsupervised Profiling Methods for Fraud Detection. Conference on Credit Scoring and Credit Control. 7.

[21]B. Baesens, V. Van Vlasselaer, and W. Verbeke, Fraud analytics using descriptive, predictive, and social network techniques: a guide to data science for fraud detection. John Wiley & Sons, 2015.

[22]Y. Liu, Z. Tang and W. Zheng, "Suspicious Bank Card Transaction Recognition Based on K-means Clustering and Random Forest Algorithm," 2019 International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS), 2019, pp. 332-336, doi: 10.1109/ICIIBMS46890.2019.8991451.

[23]Ghosh and Reilly, "Credit card fraud detection with a neural-network," 1994 Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences, 1994, pp. 621-630, doi: 10.1109/HICSS.1994.323314.

[24]A. Thennakoon, C. Bhagyani, S. Premadasa, S. Mihiranga and N. Kuruwitaarachchi, "Real-time Credit Card Fraud Detection Using Machine Learning," 2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2019, pp. 488-493, doi: 10.1109/CONFLUENCE.2019.8776942.

[25]R. D. Camino, R. State, L. Montero and P. Valtchev, "Finding Suspicious Activities in Financial Transactions and Distributed Ledgers," 2017 IEEE International Conference on Data Mining Workshops (ICDMW), 2017, pp. 787-796, doi: 10.1109/ICDMW.2017.109.

[26]"WSO2", Wso2.com, 2022. [Online]. Available: https://wso2.com/whitepapers/fraud-detection-and-prevention-a-data-analytics-approach. [Accessed: 16- December- 2021].

[27]"Temporal and Conventional Databases", Dcs.ed.ac.uk, 2022. [Online]. Available: http://www.dcs.ed.ac.uk/home/tz/phd/thesis/node10.htm. [Accessed: 20- December- 2021].

[28]Jensen, C. S., Dyreson, C. E., Bohlen, M., Clifford, J., Elmasri, R., Gadia, S. K., Grandi, F., Hayes, P., Jajodia, S., Käfer, W., Kline, N., Lorentzos, N., Mitsopoulos, Y., Montanari, A., Nonen, D., Peressi, E., Pernici, B., Roddick, J. F., Sarda, N. L., ... Wiederhold, G. (1998). The consensus glossary of temporal database concepts - february 1998 version. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 1399, 367-405. https://doi.org/10.1007/bfb0053710

[29]Jensen, Christian & Christian, Copyright & Jensen, S. & Snodgrass, Richard & (codirector, Christian & Böhlen, Michael & Busatto, Renato & Gregersen, Heidi & Torp, Kristian & (codirector, Richard & Datta, Anindya & Ram, Sudha. (1997). Temporal Data Management.

[30]G. Ozsoyoglu and R. T. Snodgrass, "Temporal and real-time databases: a survey," in IEEE Transactions on Knowledge and Data Engineering, vol. 7, no. 4, pp. 513-532, Aug. 1995, doi: 10.1109/69.404027.

[31]E. J. Coburn, "A conceptual view of temporal databases," Proceedings of the 1990 Symposium on Applied Computing, 1990, pp. 170-173, doi: 10.1109/SOAC.1990.82162.

[32]James Clifford and David S. Warren. 1983. Formal semantics for time in databases. ACM Trans. Database Syst. 8, 2 (June 1983), 214–254. DOI:https://doi.org/10.1145/319983.319986

[33]Richard Snodgrass. 1987. The temporal query language TQuel. ACM Trans. Database Syst. 12, 2 (June 1987), 247–298. DOI:https://doi.org/10.1145/22952.22956

[34]M. Kvet and M. Kvet, "Temporal database management: Temporal registration," 2017 International Conference on Information and Digital Technologies (IDT), 2017, pp. 227-233, doi: 10.1109/DT.2017.8024301.

[35]"Temporal database - Wikipedia", En.wikipedia.org, 2022. [Online]. Available: https://en.wikipedia.org/wiki/Temporal_database. [Accessed: 21- December- 2021].

[36] A. STEINER, "A Generalisation Approach to Temporal Data Models and their Implementations", Ph.D, SWISS FEDERAL INSTITUTE OF TECHNOLOGY ZURICH, 1998.

[37]Eshtay, Mohammed & Sleit, Azzam & Aldwairi, Monther. (2019). Implementing bi-temporal properties into various NoSQL database categories. International Journal of Computing Science and Mathematics. 18. 45-52.

[38]Monger, Morgan & Mata-Toledo, Ramon & Gupta, Pranshu. (2012). TEMPORAL DATA MANAGEMENT IN NoSQL DATABASES.

[39]Eshtay, Mohammed & Sleit, Azzam & Aldwairi, Monther. (2019). Implementing bi-temporal properties into various NoSQL database categories. International Journal of Computing Science and Mathematics. 18. 45-52.

[40]K. Sahatqija, J. Ajdari, X. Zenuni, B. Raufi and F. Ismaili, "Comparison between relational and NOSQL databases," 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2018, pp. 0216-0221, doi: 10.23919/MIPRO.2018.8400041.

[41]Mukherjee, Sourav. (2019). The battle between NoSQL Databases and RDBMS. 10.15680/IJIRSET.2019.0805107.

[42]R. Hecht and S. Jablonski, "NoSQL evaluation: A use case oriented survey," 2011 International Conference on Cloud and Service Computing, 2011, pp. 336-341, doi: 10.1109/CSC.2011.6138544.

[42]"What is Caching and How it Works | AWS", Amazon Web Services, Inc., 2021. [Online]. Available: https://aws.amazon.com/caching. [Accessed: 25- December- 2021].

[43]"Database Caching", Amazon Web Services, Inc., 2021. [Online]. Available: https://aws.amazon.com/caching/database-caching. [Accessed: 25- December- 2021].

[44]Stamos, Konstantinos & Pallis, George & Vakali, Athena. (2008). Caching Techniques on CDN Simulated Frameworks. 10.1007/978-3-540-77887-5_5.

[45]D. Sarkar, N. Rakesh and K. K. Mishra, "Content delivery networks: Insights and recent advancement," 2016 Fourth International Conference on Parallel, Distributed and Grid Computing (PDGC), 2016, pp. 1-5, doi: 10.1109/PDGC.2016.7913113.

[46]"What Is DNS Cache and How to Flush It - KeyCDN Support", KeyCDN, 2021. [Online]. Available: https://www.keycdn.com/support/dns-cache. [Accessed: 25- December- 2021].

[47]"Web Caching", Amazon Web Services, Inc., 2021. [Online]. Available: https://aws.amazon.com/caching/web-caching/. [Accessed: 25- December - 2021].

[48]K. Li, K. Guo and H. Guo, "Financial Big Data Hot and Cold Separation Scheme Based on HBase and Redis," 2019 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), 2019, pp. 1612-1617, doi: 10.1109/ISPA-BDCloud-SustainCom-SocialCom48970.2019.00237.

[49]"Caching Strategies and How to Choose the Right One", CodeAhoy, 2021. [Online]. Available: https://codeahoy.com/2017/08/11/caching-strategies-and-how-to-choose-the-right-one/. [Accessed: 25- December- 2021].

[50]"Caching strategies - Amazon ElastiCache", docs.aws.amazon.com, 2021. [Online]. Available: https://docs.aws.amazon.com/AmazonElastiCache/latest/mem-ug/Strategies.html [Accessed 25 December 2021].

[51]"About Handling Stale Data", 2021. [Online]. Available: https://www.eclipse.org/eclipselink/documentation/2.4/concepts/cache004.htm#:~:text=Stale%20data%20is%20an%20artifact,committed%20to%20the%20data%20source.&text=Depending%20on%20how%20you%20configure,in%20use%20within%20another%20process. [Accessed 25 December 2021].

[52]"Cache Usage Patterns", Ehcache.org, 2021. [Online]. Available: https://www.ehcache.org/documentation/3.3/caching-patterns.html#cache-as-sor. [Accessed 25 December 2021].

[53]"Cache configuration | Open CAS", Open-cas.github.io, 2021. [Online]. Available: https://open-cas.github.io/cache_configuration.html?fbclid=IwAR3FVNolFBo_gfKX7XIRpkxw9h4s7p90m_qYPZmZwOAHnCCor5CNvN2jAVc#:~:text=Write-Around%20is%20similar%20to,not%20often%20subsequently%20re-read. [Accessed 25 December 2021].

[54] "2. PII Data in Corporate Lending", Docs.oracle.com, 2022. [Online]. Available: https://docs.oracle.com/cd/F41496_01/html/PII/PII_Data.htm. [Accessed 26 December 2021].

[55]"General Data Protection Regulation (GDPR) – Official Legal Text", General Data Protection Regulation (GDPR), 2021. [Online]. Available: https://gdpr-info.eu/. [Accessed 26 December 2021].

[56]"5 Secure Steps to NYDFS Compliance | Resource Library", Resource Library, 2021. [Online]. Available: https://www.imperva.com/resources/resource-library/ebooks/5-secure-steps-to-nydfs-compliance. [Accessed 26 December 2021].

[58]"BankFind Suite", Banks.data.fdic.gov, 2021. [Online]. Available: https://banks.data.fdic.gov/explore/historical?displayFields=STNAME%2CTOTAL%2CBRANCHES%2CNew_Char&selectedEndDate=2020&selectedReport=CBS&selectedStartDate=1934&selectedStates=0&sortField=YEAR&sortOrder=desc. [Accessed 26 December 2021].

[59]S. R. M. Oliveira and O. R. Zaiane, "Protecting sensitive knowledge by data sanitization," Third IEEE International Conference on Data Mining, 2003, pp. 613-616, doi: 10.1109/ICDM.2003.1250990.

[60]M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim and V. Verykios, "Disclosure limitation of sensitive rules," Proceedings 1999 Workshop on Knowledge and Data Engineering Exchange (KDEX'99) (Cat. No.PR00453), 1999, pp. 45-52, doi: 10.1109/KDEX.1999.836532.

[61]"Overcoming the biggest transaction monitoring challenges", Napier.ai, 2022. [Online]. Available: https://www.napier.ai/post/overcoming-transaction-monitoring-challenges. [Accessed 26 December 2021].

[62]Zareapoor, Masoumeh & Shamsolmoali, Pourya. (2015). Application of Credit Card Fraud Detection: Based on Bagging Ensemble Classifier. Procedia Computer Science. 48. 679-686. 10.1016/j.procs.2015.04.201.

[63]Www2.deloitte.com, 2021. [Online]. Available: https://www2.deloitte.com/content/dam/Deloitte/in/Documents/finance/Forensic/in-forensic-AML-Survey-report-2020-noexp.pdf. [Accessed: 26- Dec- 2021].

[64]Diva-portal.org, 2021. [Online]. Available: https://www.diva-portal.org/smash/get/diva2:834221/FULLTEXT01.pdf. [Accessed: 26- Dec- 2021].

[65]Lopez-Rojas, Edgar Alonso & Elmir, Ahmad & Axelsson, Stefan. (2016). PAYSIM: A FINANCIAL MOBILE MONEY SIMULATOR FOR FRAUD DETECTION.

[66]["Overview · IBM/AMLSim Wiki", GitHub, 2021. [Online]. Available: https://github.com/IBM/AMLSim/wiki/Overview. [Accessed: 26- Dec- 2021].

[67] M. J. Eccles, D. J. Evans and A. J. Beaumont, "True Real-Time Change Data Capture with Web Service Database Encapsulation," 2010 6th World Congress on Services, Miami, FL, 2010, pp. 128-131, doi: 10.1109/SERVICES.2010.59.

[68] "What are the Different Methods of Change Data Capture (CDC)? | HVR Software", HVR Software, 2020. [Online]. Available: https://www.hvr-software.com/blog/change-data-capture/. [Accessed: 25- May- 2020].

[69]"What Is Real-Time Stream Processing? - Hazelcast", Hazelcast, 2022. [Online]. Available: https://hazelcast.com/glossary/real-time-stream-processing/. [Accessed: 10- Jan- 2022].

[70]Confluent.io, 2022. [Online]. Available: https://www.confluent.io/learn/data-streaming/. [Accessed: 10- Jan- 2022].

[71]"MySQL vs. Oracle Comparison", Db-engines.com, 2022. [Online]. Available: https://db-engines.com/en/system/MySQL%3BOracle. [Accessed 26 January 2022].

[72] Assefa, Samuel A.. "Generating Synthetic Data in Finance: Opportunities, Challenges and Pitfalls." InfoSciRN: Data Protection (Topic) (2020): n. pag.

[73] S. D. P. Mendonça, Y. P. D. S. Brito, C. G. R. D. Santos, R. D. A. D. Lima, T. D. O. D. Araújo and B. S. Meiguins, "Synthetic Datasets Generator for Testing Information Visualization and Machine Learning Techniques and Tools," in IEEE Access, vol. 8, pp. 82917-82928, 2020, doi: 10.1109/ACCESS.2020.2991949.

[74] G. Soltana, M. Sabetzadeh and L. C. Briand, "Synthetic data generation for statistical testing," 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE), 2017, pp. 872-882, doi: 10.1109/ASE.2017.8115698.

# 7 APPENDIX A: ANTI-MONEY LAUNDERING LAWS AND REGULATIONS IN GLOBE

| Organization | Description |
|---|---|
| Financial Action Task Force (FATF) | The Financial Action Task Force recommendations, the international anti-money laundering and combating the financing of terrorism and proliferation standards, and define methodology to assess the effectiveness of anti-money laundering systems.<br><br>Source: "Documents - Financial Action Task Force (FATF)", Fatf-gafi.org, 2020. [Online]. Available: http://www.fatf-gafi.org/publications/fatfrecommendations/documents/fatf-recommendations.html. [Accessed: 17- May- 2020]. |
| European Union | The European Union has adopted strong legislation to combat money laundering and terrorist financing and is contributing to these international efforts. The European Commission will consider the transposition of the EU's Acquits communautaire and, in cooperation with a network of competent authorities, will ensure the effective application of this law. Some of well-known guide lines are **4AMLD** , **5AMLD, 6AMLD** .<br>Source: "Anti-money laundering and counter terrorist financing", European Commission - European Commission, 2020. [Online]. Available: https://ec.europa.eu/info/business-economy-euro/banking-and-finance/financial-supervision-and-risk-management/anti-money-laundering-and-counter-terrorist-financing_en. [Accessed: 17- May- 2020]. |
| UK: Financial Conduct Authority (FCA) | An independent non-governmental agency responsible for the regulation of the UK financial services industry. Its purpose is to protect consumers by ensuring the integrity and stability of the market. |

| | |
|---|---|
| | Source: "Money Laundering Regulations", FCA, 2020. [Online]. Available: https://www.fca.org.uk/firms/financial-crime/money-laundering-regulations. [Accessed: 17- May- 2020]. |
| US: Bank Secrecy Act | The Bank Secrecy Act (also known as the Currency and Foreign Transaction Reporting Act) is the leading anti-money laundering regulation in the United States and is governed by the Financial Crimes Enforcement Network. The scope of the law covers both money laundering and financial crimes.<br>Source: [7]"Bank Secrecy Act (BSA) \| OCC", Occ.treas.gov, 2020. [Online]. Available: https://www.occ.treas.gov/topics/supervision-and-examination/bsa/index-bsa.html. [Accessed: 17- May- 2020]. |
| Hong Kong Monetary Authority (HKMA) | The Hong Kong Monetary Authority is responsible for the stability of Hong Kong's banking system and monetary policy. HKMA oversees the AML / CFT risk management system of the Accreditation Body (IA), which follows international standards and practices, taking into account the risks faced by the banking industry and the IA individual is exposed.<br><br>Source: H. Authority, "Hong Kong Monetary Authority - Anti-Money Laundering and Counter-Financing of Terrorism", Hong Kong Monetary Authority, 2020. [Online]. Available: https://www.hkma.gov.hk/eng/key-functions/banking/anti-money-laundering-and-counter-financing-of-terrorism/. [Accessed: 17- May- 2020]. |
| Monetary Authority of Singapore (MAS) | The Monetary Authority of Singapore (MAS) is Singapore's central bank and integrated financial regulator. MAS' AML policy covers due diligence procedures, KVC verification process and reporting and monitoring activates which need to fallow by their banking and financial institutes. |

| | |
|---|---|
| | Source: "Who We Are", Mas.gov.sg, 2020. [Online]. Available: https://www.mas.gov.sg/who-we-are. [Accessed: 17- May- 2020]. |
| Australia: AUSTRAC | The Australian Transaction Reporting and Analysis Center is the Australian Government's financial intelligence agency established to monitor financial transactions to identify money laundering, terrorist financing, fraud and other financial crimes.<br>Source: [10]"Homepage \| AUSTRAC", Austrac.gov.au, 2020. [Online]. Available: https://www.austrac.gov.au/. [Accessed: 17- May- 2020]. |

# 8 APPENDIX B: ANTI-MONEY LAUNDERING PATTERNS AND SAMPLE SCENARIOS (MOBILE BANKING)

Nature of the money laundering process make allow criminals to practice these illegal methods in a countless umber of diverse ways. Below are some sample scenarios which need to consider while development of anti-money laundering patterns for mobile banking systems.

**Pattern 1 Type 1: Detection of Multiple Instances of transaction for same From/To combination.**

- Need to identify clients which transfer money to another clients with abnormal frequency.
- Need to identify clients which cash in into mobile banking system and convert them in to property by paying bills for merchants in abnormal frequency.
- There may be some programs to pay money over specified intervals. Those programs can be identified using properly analyzing the transaction date timestamp.

**Example Dataset 1: Customer top-up money in unusual frequency**

Assumptions**:** General cash in frequency for customer does not increase 4 time within 1 hour.

*Table 1: Pattern 1 Type 1 Dataset 1*

| Sender(Retailer )Acc. | Receiver (Customer)Acc. | Transaction Type | Transaction Amount | Transaction Reference | Transaction Date |
|---|---|---|---|---|---|
| 7 | 15 | Cash In | 2000 | 1652072396316 | 2019-06-21 05:06:00 |
| 7 | 15 | Cash In | 1000 | 1652072396317 | 2019-06-21 05:07:00 |
| 7 | 15 | Cash In | 1000 | 1652072396318 | 2019-06-21 05:08:00 |
| 7 | 15 | Cash In | 1000 | 1652072396319 | 2019-06-21 05:09:08 |
| 7 | 15 | Cash In | 1000 | 1652072396320 | 21. 5:09 :20 |

Note that the total number of cash in (same From/To) within an hour is 5.So this retailer and customer could take as suspicions based on their agreement customer.

**Example Dataset 2: Customer transfer money in unusual frequency**

Assumptions**:** Maximum allowable transaction count for same from/to is 4 for within 1 hour

*Table 2: Pattern 1 Type 1 Dataset 2*

| Sender Acc. | Receiver Acc. | Transaction Type | Transaction Amount | Transaction Reference | Transaction Date |
|---|---|---|---|---|---|
| 17 | 18 | Transfer | 750 | 1652072396321 | 2019-06-21 05:09:20 |
| 17 | 18 | Transfer | 500 | 1652072396322 | 2019-06-21 05:09:21 |
| 17 | 18 | Transfer | 1100 | 1652072396323 | 2019-06-21 05:09:23 |
| 17 | 18 | Transfer | 1000 | 1652072396324 | 2019-06-21 05:09:25 |
| 17 | 18 | Transfer | 200 | 1652072396325 | 2019-06-21 05:09:27 |

- Total Number of money transfer from one customer to another within an hour is 4.So this could be a practice of money laundering activity.
- Notice that the transaction time was happen in very close time interval which is difficult to trigger using human being. So this could be a suspicious activity which performed using programmatic manner.

**Pattern 2 Type1: Detection of customers with Same NIC/SSN/DL & Different Names, Addresses**

- Most of the banking and financial systems support to for their customer to create new accounts using different documents. But those documents should represent same person.

**Example Dataset 1: Conflicting Client Information**

*Table 3: Pattern 2 Type 2 Dataset 1*

| Wallet Acc. | Mobile | Wallet Type | Name | Address | NIC |
|---|---|---|---|---|---|

| | | | | | |
|---|---|---|---|---|---|
| 3 | 071-276-8527 | Retailer | Kasun G | Pothupitiya,Pinnaduwa,Galle | 871012713V |
| 7 | 071-572-4336 | Retailer | Banuka P | No 25,Edirisinha Rd,Colombo 6 | 871012713V |
| 8 | 072-667-5761 | Merchant | Banuka PW | No 25,Edirisinha Rd,Colombo 6 | 871012713V |
| 9 | 071-543-0194 | Retailer | Banuka PW | No 25,Edirisinha Rd,Colombo 6 | 871012713V |
| 10 | 071-251-2766 | Retailer | Banuka PW | No 25,Edirisinha Rd,Colombo 6 | 871012713V |
| 105 | 072-2549557 | Merchant | Banuka PW | No 25,Edirisinha Rd,Colombo 6 | 871012713V |

Notice that the client with same NIC but having different contact numbers, names and addresses for his/her different account types (clients allowed to have multiple account types).We cannot directly categorized as money launder, but can take as suspicious.

**Pattern 3 Type 1: Client with deposits and withdrawals at multiple locations (agents).**

The most observed stages of money laundering were placement and layering and the most common techniques for money laundering were structuring and smurfing. Structuring normally involves multiple cash deposits or withdrawals at amounts below the reporting threshold and smurfing is defined as multiple deposits of cash, and/or low-value monetary instruments purchased from various banks or money services businesses by various individuals.

To verify this situation, we need to analysis three cases:

1. Particular customers cash in frequency within specified time.
2. Cash out frequency of that particular customer within specified time.
3. Number of different retails who help to cash out for that particular customer.

**Example Dataset 1: Customer Cash In and Cash Out**

*Table 4: Pattern 4 Type 1 Dataset 1*

| Wallet Acc. | Transaction Type | Transaction Amount | Transaction Reference | Transaction Date |
|---|---|---|---|---|
| 26 | Cash In | 700 | 1652072396337 | 2019-06-11 10:01:10 |
| 26 | Cash In | 500 | 1652072396338 | 2019-06-11 10:01:12 |
| 26 | Cash In | 1000 | 1652072396339 | 2019-06-11 10:05:12 |
| 26 | Cash In | 500 | 1652072396340 | 2019-06-11 11:05:12 |
| 26 | Cash In | 800 | 1652072396341 | 2019-06-11 11:06:12 |
| 26 | Cash In | 5000 | 1652072396349 | 2019-06-12 12:48:12 |
| 26 | Cash In | 5000 | 1652072396349 | 2019-06-12 12:58:12 |

*Table 7: Pattern 4 Type 1 Dataset 2*

| Wallet Acc. | Transaction Type | Transaction Amount | Transaction Reference | Transaction Date |
|---|---|---|---|---|
| 26 | Cash Out | 1000 | 1652072396342 | 2019-06-09 11:06:12 |
| 26 | Cash Out | 1000 | 1652072396345 | 2019-06-09 11:09:12 |
| 26 | Cash Out | 500 | 1652072396344 | 2019-06-09 12:07:12 |
| 26 | Cash Out | 500 | 1652072396346 | 2019-06-09 12:14:12 |
| 26 | Cash Out | 800 | 1652072396347 | 2019-06-09 12:18:12 |
| 26 | Cash Out | 2500 | 1652072396351 | 2019-06-09 12:48:22 |
| 26 | Cash Out | 3000 | 1652072396352 | 2019-06-09 12:49:12 |
| 26 | Cash Out | 2500 | 1652072396353 | 2019-06-09 12:55:12 |

Note that the customer account no 26, executed **cash in (**Pattern 4 Type 1 Dataset 1**)** operation followed by **cash out** (Pattern 4 Type 1 Dataset 2) exceed the general acceptable level. When evaluate the agents which support to those operation, it gives different agent accounts registered under different geographical areas (Pattern 4 Type 1 Dataset 3).

*Table 5: Pattern 4 Type 1 Dataset 3*

| Transaction Reference | Customer Acc. | Retailer Acc. |
|---|---|---|
| 1652072396342 | 26 | 1 |
| 1652072396345 | 26 | 16 |
| 1652072396344 | 26 | 14 |
| 1652072396346 | 26 | 25 |
| 1652072396347 | 26 | 25 |
| 1652072396351 | 26 | 14 |
| 1652072396352 | 26 | 16 |
| 1652072396353 | 26 | 25 |

So the customer with account number 26 could take as suspicious.

**Pattern 4 Type 1: Identify the clients with unusual number of payee (when sender) or payers (when receiver)**

- Same from to combination checked in earlier (Pattern 1). But it cannot identify when there are different clients send to one receiver or one client send to multiple receivers.
- Also this pattern not worried weather client receive multiple number of same amount transaction or multiple number of variable amount transactions (RS: 100, 102, 132, 98, etc.).Separate pattern need to identify such transactions.

**Example Dataset 1: Client with unusual payers**

*Table 6: Pattern 5 Type 1 Dataset 1*

| Receiver Acc. | Sender Acc. | Transaction Amount | Transaction Type | Transaction Reference | Transaction Date |
|---|---|---|---|---|---|
| 15 | 19 | 200 | Transfer | 1652072396329 | 2016-06-09 09:57:50 |
| 15 | 45 | 102 | Transfer | 1652072396359 | 2016-06-09 12:58:22 |
| 15 | 47 | 101 | Transfer | 1652072396360 | 2016-06-09 12:18:23 |
| 15 | 48 | 103 | Transfer | 1652072396361 | 2016-06-09 12:59:24 |
| 15 | 23 | 100 | Transfer | 1652072396362 | 2016-06-09 12:59:25 |
| 15 | 26 | 101 | Transfer | 1652072396363 | 2016-06-09 12:59:25 |
| 15 | 27 | 102 | Transfer | 1652072396364 | 2016-06-09 12:59:26 |
| 15 | 29 | 105 | Transfer | 1652072396365 | 2016-06-09 12:59:26 |

| | | | | | |
|---|---|---|---|---|---|
| 15 | 50 | 102 | Transfer | 1652072396366 | 2016-06-09 12:59:26 |
| 15 | 50 | 120 | Transfer | 1652072396367 | 2016-06-09 12:59:27 |
| 15 | 52 | 115 | Transfer | 1652072396368 | 2016-06-09 12:59:27 |

Notice that above customer (receiver account number=15) received different amount of same type of transactions (transfer) within a day from unusual number of senders.

**Example Dataset 2: Pay For unusual number of Good or Service**

Assumption: Generally a customers does not pay more than 4 institute payments within a day (constrain is subjected to change over financial organizational preferences).

*Table 7 : Pattern 5 Type 1 Dataset 2*

| Sender Acc. | Receiver Acc. | Transaction Type | Transaction Amount | Transaction Reference | Transaction Date |
|---|---|---|---|---|---|
| 19 | 108 | Institute | 10000 | 1652072396409 | 2016-06-11 07:16:20 |
| 19 | 109 | Institute | 11000 | 1652072396410 | 2016-06-11 07:16:22 |
| 19 | 110 | Institute | 11000 | 1652072396412 | 2016-06-11 07:16:23 |
| 19 | 103 | Institute | 10000 | 1652072396374 | 2016-06-11 10:59:30 |
| 19 | 104 | Institute | 10000 | 1652072396375 | 2016-06-11 11:59:20 |

Observe that customer account number 19, pay for 4 distinct institutes in the same day. Therefore he/she could be suspicious. In addition to this case, there can be situation where one customer pay for same merchant in unaccepted frequency.

**Pattern 5 Type 1: Need to detect clients with unusual number of Cash-in followed by a Cash-out in a short period of time.**

Pattern 4 Type 1 concentrate on customers who cash out using different locations. But here concentrate on, customers who cash in and cash out in same location.

**Example Dataset 1: unusual cash in fallowed by cash out**

*Table 8: Pattern 1 Type 1 Dataset 1*

| Customer Acc. | Retailer Acc. | Transaction Amount | Transaction Type | Transaction Reference | Transaction Time |
|---|---|---|---|---|---|
| 26 | 25 | 13000 | Cash In | 1652072396413 | 2016-06-09 07:16:23 |
| 26 | 25 | 12000 | Cash In | 1652072396414 | 2016-06-09 07:26:23 |
| 26 | 25 | 5000 | Cash In | 1652072396415 | 2016-06-09 08:20:00 |

*Table 12: Pattern1 Type 1 Dataset 2*

| Customer Acc. | Retailer Acc. | Transaction Amount | Transaction Type | Transaction Reference | Transaction Time |
|---|---|---|---|---|---|
| 26 | 25 | 13000 | Cash Out | 1652072396346 | 2016-06-09 07:16:23 |
| 26 | 25 | 12000 | Cash Out | 1652072396347 | 2016-06-09 07:26:23 |
| 26 | 25 | 5000 | Cash Out | 1652072396416 | 2016-06-09 08:25:00 |

Notice the color codes used in above two tables. Customer **cash in** and after a small time period, again **cash out** .So this activity could be suspicious.

# 9 APPENDIX C: GUIDELINES FOR MICRO TEMPORAL DATABSE GENERATOR

**Q1) How to convert my relational OLTP database to a TEMPORAL database?**

OLTP to micro TEMPORAL database conversion is a three-step process.

**Step 01: Configure Database Server**

As the 1st step, the you needs to configure your database server details in "application.yaml file. The below screenshot shows how it is configured when the database server is run on the same local machine.


**Step 02: Configure Schema Mapping**

In this step, the user needs to map the current OLTP attributes in "schema_mapping.yaml" file by fallowing the guidelines given in 'Schema Mapping Guidelines' table. The below 'Schema Mapping File' figure shows the sample schema mapping file for the OLTP schema model.

*Table 9.1: Schema Mapping Guidelines*

| Guideline ID | Description |
|---|---|
| G-001 | OLTP's monetary transaction table is considered the main table and there should have only one main table. In most cases this is the transaction table.<br>Maintainable should mark with 'isMainTable:TRUE' and all other tables should mark with isMainTable:FALSE. |

| | |
|---|---|
| G-oo2 | There can be only two types of tables in the mapping file. They are "TEMPORAL" tables and "LOOKUP" tables.<br>All the tables which expect to have temporal properties should mark as "TEMPORAL" and all other tables should mark as "LOOKUP" tables. |
| G-003 | There can have only two type of indexes. They are general indexes which allows null and empty values ("INDEX") and "PRIMARY KEY" indexes.'PRIMARY KEY' value can use to mark the unique key in main table (transaction table) and primary keys in "LOOKUP" tables. All the other table indexes should mark with "INDEX" indexes (including the other key indexes in main table). |
| G-004 | tableName, columnName, indexName will directly reflect the same name in temporal database schema. They should mark with the relevant expected names. |
| G-005 | Column dataType will directly reflect in temporal schema. So they should give the related matching data types for your expected temporal database. |

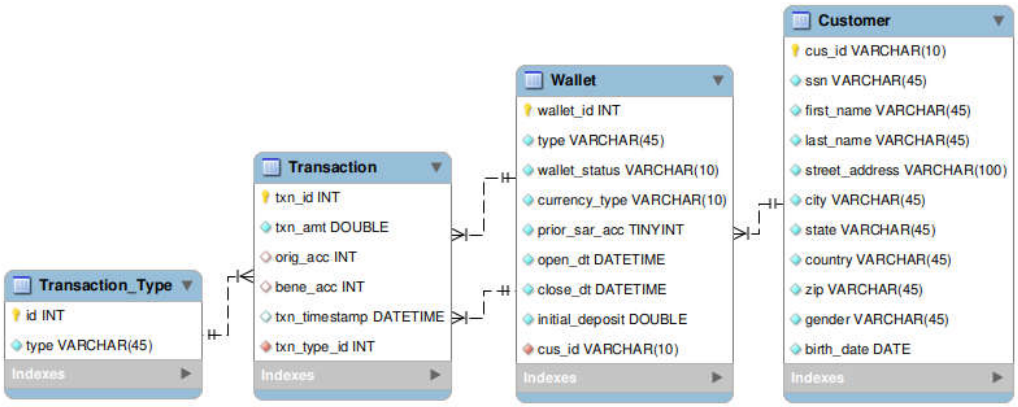


*Figure 9.1: OLTP Schema Model*

```
---
tables:
 - tableName: Customer
   tableType: TEMPORAL
```

```
  isMainTable: FALSE
  columns:
    - columnName: cus_id
      isNullable: false
      dataType: VARCHAR(20)
    - columnName: ssn
      isNullable: false
      dataType: VARCHAR(50)
    - columnName: first_name
      isNullable: false
      dataType: VARCHAR(50)
    - columnName: last_name
      isNullable: false
      dataType: VARCHAR(50)
    - columnName: street_address
      isNullable: false
      dataType: VARCHAR(50)
    - columnName: city
      isNullable: false
      dataType: VARCHAR(50)
    - columnName: state
      isNullable: false
      dataType: VARCHAR(50)
    - columnName: country
      isNullable: false
      dataType: VARCHAR(50)
    - columnName: zip
      isNullable: false
      dataType: VARCHAR(10)
    - columnName: gender
      isNullable: false
      dataType: VARCHAR(10)
    - columnName: birth_date
      isNullable: false
      dataType: date
  indexes:
    - indexType: INDEX
      indexName: cus_id_idx
      columnName: cus_id
- tableName: Transaction
  tableType: TEMPORAL
  isMainTable: TRUE
  columns:
    - columnName: txn_id
      isNullable: false
      dataType: INT
    - columnName: txn_amt
      isNullable: false
      dataType: DOUBLE
    - columnName: orig_acc
      isNullable: false
      dataType: INT
    - columnName: bene_acc
      isNullable: false
      dataType: INT
    - columnName: txn_timestamp
      isNullable: false
```

```
      dataType: DATETIME
    - columnName: txn_type_id
      isNullable: false
      dataType: INT
  indexes:
    - indexType: PRIMARY KEY
      indexName: txn_id_idx
      columnName: txn_id
    - indexType: INDEX
      indexName: wallet_id_a_idx
      columnName: orig_acc
    - indexType: INDEX
      indexName: wallet_id_b_idx
      columnName: bene_acc
    - indexType: INDEX
      indexName: txn_type_id_idx
      columnName: txn_type_id
- tableName: Wallet
  tableType: TEMPORAL
  isMainTable: FALSE
  columns:
    - columnName: wallet_id
      isNullable: false
      dataType: INT
    - columnName: type
      isNullable: false
      dataType: VARCHAR(10)
    - columnName: wallet_status
      isNullable: false
      dataType: VARCHAR(10)
    - columnName: currency_type
      isNullable: false
      dataType: VARCHAR(10)
    - columnName: prior_sar_acc
      isNullable: false
      dataType: TINYINT
    - columnName: open_dt
      isNullable: false
      dataType: DATETIME
    - columnName: close_dt
      isNullable: false
      dataType: DATETIME
    - columnName: initial_deposit
      isNullable: false
      dataType: DOUBLE
    - columnName: cus_id
      isNullable: false
      dataType: VARCHAR(20)
  indexes:
    - indexType: INDEX
      indexName: wallet_id_idx
      columnName: wallet_id
- tableName: Transaction_Type
  tableType: LOOKUP
  isMainTable: FALSE
  columns:
    - columnName: id
```

```
      isNullable: false
      dataType: INT
    - columnName: type
      isNullable: false
      dataType: VARCHAR(20)
  indexes:
    - indexType: PRIMARY KEY
      indexName: txn_type_id_idx
      columnName: id
```

*Figure 9.2: Schema Mapping File*

*Table 9.2 : Schema Mapping Attribute Definitions*

| **Attribute** | **Meaning** |
|---|---|
| tableName | Name of the relational table which expect to create in temporal database |
| tableType | Define weather table should create as "TEMPORAL" table or "LOOKUP" table |
| isMainTable | Define "TRUE" for transaction table. |
| columns.columnName | Name of the temporal database column |
| columns.isNullable | Define weather column can have null values or not |
| columns.dataType | Data type & size which support in the expected database |
| indexes.indexType | Type of the index. Can have one of "PRIMARY_KEY" or "INDEX" value |
| indexes.indexName | Name of the index |
| indexes.columnName | Index column name |

**Step 03: Generate Micro Temporal Databases**

Users can use a simple Restful API endpoint to generate any amount of databases. Q2 covers this area.

Figure xx shows the schema diagram for above OLTP mapping.

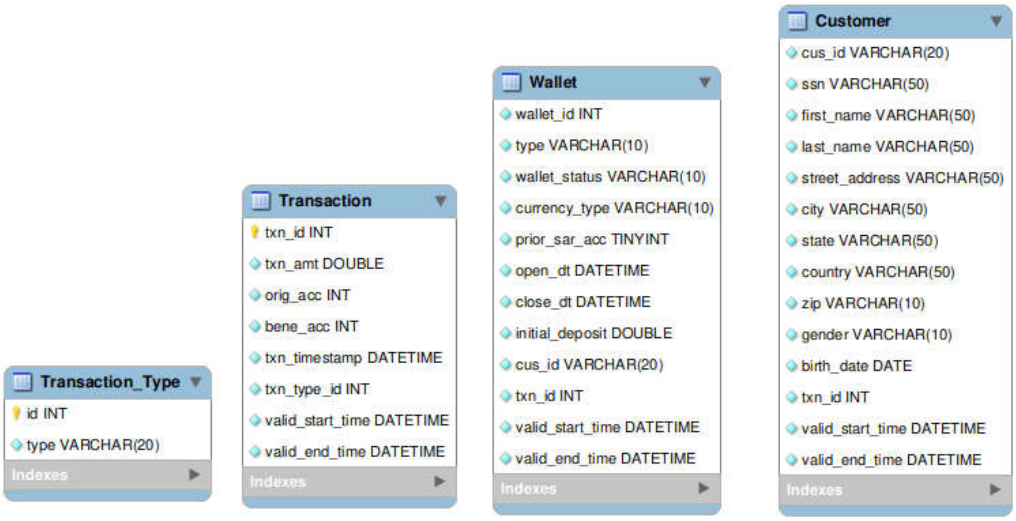


*Figure 9.3 : Temporal Schema Model*

**How to use micro temporal databases to generate micro temporal databases for 7 days, 21 days and 28 day time periods?**

Micro temporal database generator tool provides a simple Restful web service API to generate temporal databases by getting ‘database name prefix’ and ‘temporal time period’ as user inputs. The API response contains the full name of the generated database and the data caching time period. Below are the requests and responses received when you call API to generate databases for 7, 21, and 28 day cache time periods.

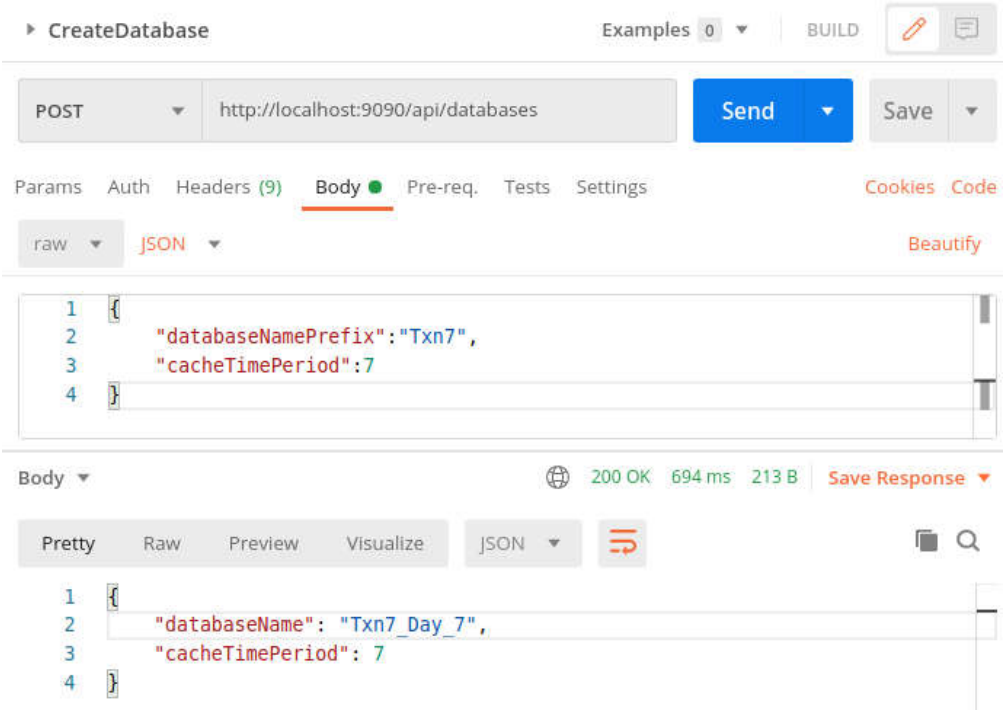


*Figure 9.4: Create Temporal Database as 7 Day Data Cache*

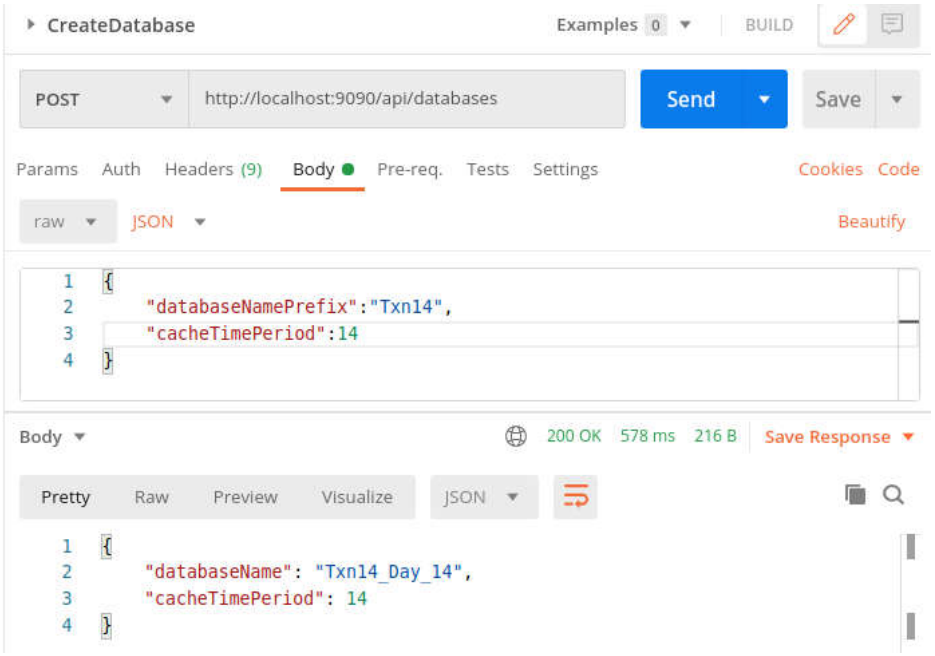


*Figure 9.5: Create Temporal Database as 14 Day Data Cache*

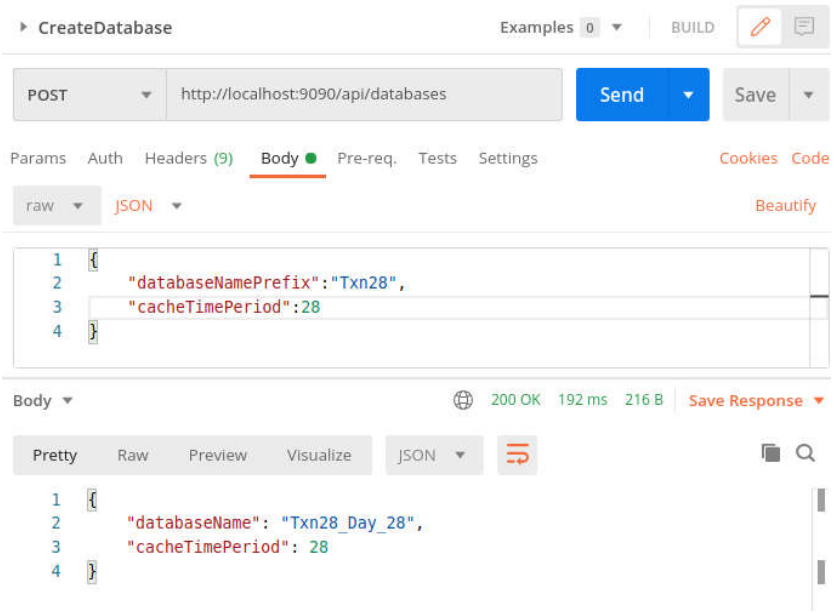


*Figure 9.6: Create Temporal Database As 28 Day Data Cache*

**Q 3) How can we verify the databases & Schedules are properly created?**

You can use any SQL client application to verify this information. Below we provide, how to verify using MySQL workbench client.

**Check Databases:** Connect to the database server using MySQL workbench and use schema view to verify the created databases.

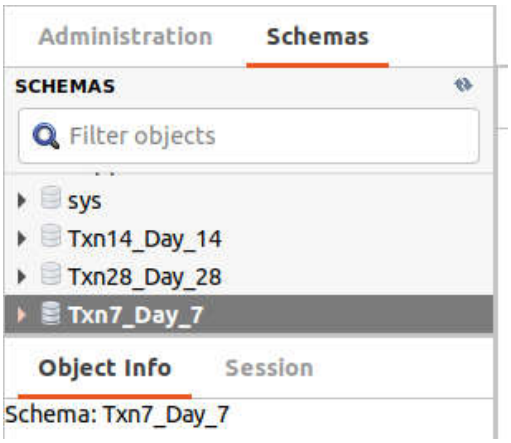


*Figure 9.7: Verify Database*

**Check Tables:** Right-click on the one of temporal database and select "Schema Inspector". Select the "Tables "tab view under that menu. Below "Verify Tables" figure shows the table view for the Txn7_Day_7 database.

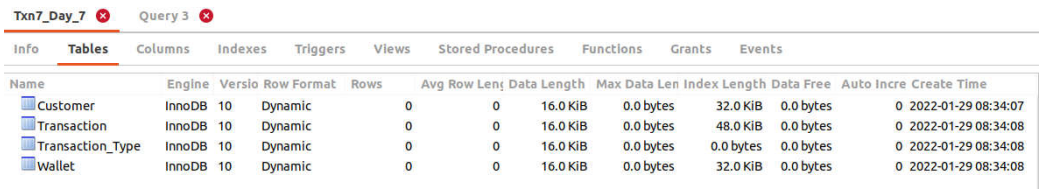


*Figure 9.8: Verify Tables*

**Check Event Triggers:** Right-click on the one of temporal database and select "Schema Inspector". Then, select the 'Events' tab to view created event triggers. . Below "Verify Events" figure shows the event view for the Txn7_Day_7 database.


*Figure 9.9: Verify Events*

**Q 4) Can I see the generated temporal schema?**

Micro temporal database generator saves a copy of each temporal schema under the **micro-db-generator/outputs** directory. Below figure shows the screenshot of the location in view of the IntelliJ code editor.

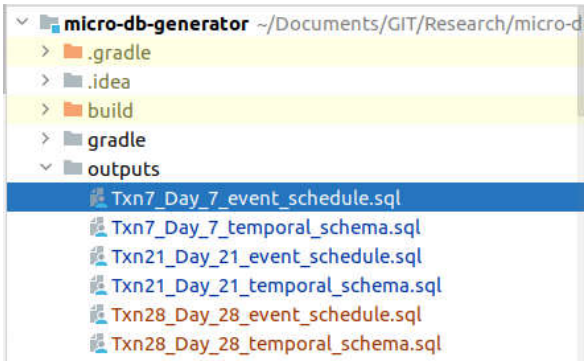


*Figure 9.10: Temporal Schema Script Location*

The temporal schema consists of database **table schema** & **event schedule schema.** Below is sample schemas for the Txn7_Day7 database.

```
CREATE DATABASE IF NOT EXISTS Txn7_Day_7;
USE Txn7_Day_7;

/*------------------DDL Script For Customer Table-----------------------*/
CREATE TABLE IF NOT EXISTS Customer
```

```
(cus_id VARCHAR(20) NOT NULL
, ssn VARCHAR(50) NOT NULL
, first_name VARCHAR(50) NOT NULL
, last_name VARCHAR(50) NOT NULL
, street_address VARCHAR(50) NOT NULL
, city VARCHAR(50) NOT NULL
, state VARCHAR(50) NOT NULL
, country VARCHAR(50) NOT NULL
, zip VARCHAR(10) NOT NULL
, gender VARCHAR(10) NOT NULL
, birth_date date NOT NULL
, txn_id INT NOT NULL
, valid_start_time DATETIME NOT NULL
, valid_end_time DATETIME NOT NULL
,INDEX cus_id_idx ( cus_id)
, INDEX txn_id_idx ( txn_id)
);
/*------------------DDL Script For Wallet Table-----------------------*/
CREATE TABLE IF NOT EXISTS Wallet
(wallet_id INT NOT NULL
, type VARCHAR(10) NOT NULL
, wallet_status VARCHAR(10) NOT NULL
, currency_type VARCHAR(10) NOT NULL
, prior_sar_acc TINYINT NOT NULL
, open_dt DATETIME NOT NULL
, close_dt DATETIME NOT NULL
, initial_deposit DOUBLE NOT NULL
, cus_id VARCHAR(20) NOT NULL
, txn_id INT NOT NULL
, valid_start_time DATETIME NOT NULL
, valid_end_time DATETIME NOT NULL
,INDEX wallet_id_idx ( wallet_id)
, INDEX txn_id_idx ( txn_id)
);
/*------------------DDL Script For Transaction Table-----------------------*/
CREATE TABLE IF NOT EXISTS Transaction
(txn_id INT NOT NULL
, txn_amt DOUBLE NOT NULL
, orig_acc INT NOT NULL
, bene_acc INT NOT NULL
, txn_timestamp DATETIME NOT NULL
, txn_type_id INT NOT NULL
, valid_start_time DATETIME NOT NULL
, valid_end_time DATETIME NOT NULL
,PRIMARY KEY txn_id_idx ( txn_id)
, INDEX wallet_id_a_idx ( orig_acc)
, INDEX wallet_id_b_idx ( bene_acc)
, INDEX txn_type_id_idx ( txn_type_id)
);
/*------------------DDL Script For Transaction_Type Table-----------------------*/
CREATE TABLE IF NOT EXISTS Transaction_Type
(id INT NOT NULL
, type VARCHAR(20) NOT NULL
,PRIMARY KEY txn_type_id_idx ( id)
);
CREATE DATABASE IF NOT EXISTS Txn7_Day_7;
```

```
USE Txn7_Day_7;

/*------------------DDL Script For Customer Table-----------------------*/
CREATE TABLE IF NOT EXISTS Customer
(cus_id VARCHAR(20) NOT NULL
, ssn VARCHAR(50) NOT NULL
, first_name VARCHAR(50) NOT NULL
, last_name VARCHAR(50) NOT NULL
, street_address VARCHAR(50) NOT NULL
, city VARCHAR(50) NOT NULL
, state VARCHAR(50) NOT NULL
, country VARCHAR(50) NOT NULL
, zip VARCHAR(10) NOT NULL
, gender VARCHAR(10) NOT NULL
, birth_date date NOT NULL
, txn_id INT NOT NULL
, valid_start_time DATETIME NOT NULL
, valid_end_time DATETIME NOT NULL
,INDEX cus_id_idx ( cus_id)
, INDEX txn_id_idx ( txn_id)
);
/*------------------DDL Script For Wallet Table-----------------------*/
CREATE TABLE IF NOT EXISTS Wallet
(wallet_id INT NOT NULL
, type VARCHAR(10) NOT NULL
, wallet_status VARCHAR(10) NOT NULL
, currency_type VARCHAR(10) NOT NULL
, prior_sar_acc TINYINT NOT NULL
, open_dt DATETIME NOT NULL
, close_dt DATETIME NOT NULL
, initial_deposit DOUBLE NOT NULL
, cus_id VARCHAR(20) NOT NULL
, txn_id INT NOT NULL
, valid_start_time DATETIME NOT NULL
, valid_end_time DATETIME NOT NULL
,INDEX wallet_id_idx ( wallet_id)
, INDEX txn_id_idx ( txn_id)
);
/*------------------DDL Script For Transaction Table-----------------------*/
CREATE TABLE IF NOT EXISTS Transaction
(txn_id INT NOT NULL
, txn_amt DOUBLE NOT NULL
, orig_acc INT NOT NULL
, bene_acc INT NOT NULL
, txn_timestamp DATETIME NOT NULL
, txn_type_id INT NOT NULL
, valid_start_time DATETIME NOT NULL
, valid_end_time DATETIME NOT NULL
,PRIMARY KEY txn_id_idx ( txn_id)
, INDEX wallet_id_a_idx ( orig_acc)
, INDEX wallet_id_b_idx ( bene_acc)
, INDEX txn_type_id_idx ( txn_type_id)
);
/*------------------DDL Script For Transaction_Type Table-----------------------*/
CREATE TABLE IF NOT EXISTS Transaction_Type
```

```
(id INT NOT NULL
, type VARCHAR(20) NOT NULL
,PRIMARY KEY txn_type_id_idx ( id)
);
```

```
CREATE EVENT IF NOT EXISTS Txn7_Day_7_Customer_Event
ON SCHEDULE EVERY 7 Day
DO
DELETE FROM Customer WHERE valid_end_time < NOW();

CREATE EVENT IF NOT EXISTS Txn7_Day_7_Wallet_Event
ON SCHEDULE EVERY 7 Day
DO
DELETE FROM Wallet WHERE valid_end_time < NOW();

CREATE EVENT IF NOT EXISTS Txn7_Day_7_Transaction_Event
ON SCHEDULE EVERY 7 Day
DO
DELETE FROM Transaction WHERE valid_end_time < NOW();

CREATE EVENT IF NOT EXISTS Txn7_Day_7_Customer_Event
ON SCHEDULE EVERY 7 Day
DO
DELETE FROM Customer WHERE valid_end_time < NOW();

CREATE EVENT IF NOT EXISTS Txn7_Day_7_Wallet_Event
ON SCHEDULE EVERY 7 Day
DO
DELETE FROM Wallet WHERE valid_end_time < NOW();

CREATE EVENT IF NOT EXISTS Txn7_Day_7_Transaction_Event
ON SCHEDULE EVERY 7 Day
DO
DELETE FROM Transaction WHERE valid_end_time < NOW();
```

**Q 5) Can I convert OLTP with NoSQL databases into micro TEMPORAL databases using this tool?**

Short Answer: NO

Long Answer:

Most of the industrial monetary transactions databases are built on top of relational databases due to the natural transactional support (ACID properties). So this tool is designed to support converting the relational OLTP database schema into any kind of

micro temporal database (SQL or NoSQL). But as the initial version, we are providing support to cover OLTP into MySQL micro temporal databases.

If you are from a development background, you can generate any kind of micro temporal database by implementing our API contract document in the below figure.

```
public abstract class MicroDatabaseTemplate {

    /**
     * <p>
     * The main responsibility of this method to convert the OLTP to temporal mapping into target database schema.
     * Current implementation work as default behavior.It converts temporal mapping to SQL DDL statements and save
     * them in outputs directory .
     * </p>
     * <p>
     * <b>Hint:</b> For NOSQL databases : Generate index documents or collection mappings based on underlying
     * database engine.
     * </p>
     *
     * @param databaseName   temporal database name
     * @param databaseSchema schema mapping object
     */
    void generateSchema(String databaseName, DatabaseSchema databaseSchema) {

    }

    /**
     * <p>
     * The main intention of this method to connect to database server and create database and
     * implement the  previously (@code generateSchema step) generated schema in it.
     * </p>
     * <p>
     * <b>Hint:</b> For NOSQL databases : Create indexes or collection etc.
     * </p>
     *
     * @param databaseName temporal database name
     */
    abstract void executeSchema(String databaseName);

    /**
     * <p>The main intention of this method to provide a mechanism to  maintain data caching time period.
     * Since the different database engine support different mechanisms, users need to provide their preference
     * implementation.
     * </p>
     * <p>
     * Ex:
     *      <ul>
     *          <li>Database Events  in MySql</li>
     *          <li>Database triggers</li>
     *          <li>External crone triggers in AWS EventBridge</li>
     *          <li>NOSQL databases has TTL</li>
     *      </ul>
     * </p>
     *
     * @param databaseName temporal database name
     * @param cachePeriod  cache time period in days
     * @param tableNames   temporal table mapping
     */
    abstract void scheduleCachePeriod(String databaseName, int cachePeriod, List < String > tableNames);

    /**
     * Save configurations in a file system or database for later use
     */
    abstract void saveConfigs();
```

*Figure 9.11: Temporal Schema Generator Template Contract*

# 10 APPENDIX D: DATA PREPROCESSING AND LOADING FOR OLTP AND TEMPORAL DATABASES

## Data Preprocessing

We have provided the Jupiter notebooks to preprocess data for both OLTP and temporal databases. The "Generate_oltp_data_scripts.ipynb" notebook (refer the below figure) is use to generate SQL insert scripts for OLTP database and "Generate_template_data_scripts.ipynb" notebook is use to generate SQL scripts for each temporal database .

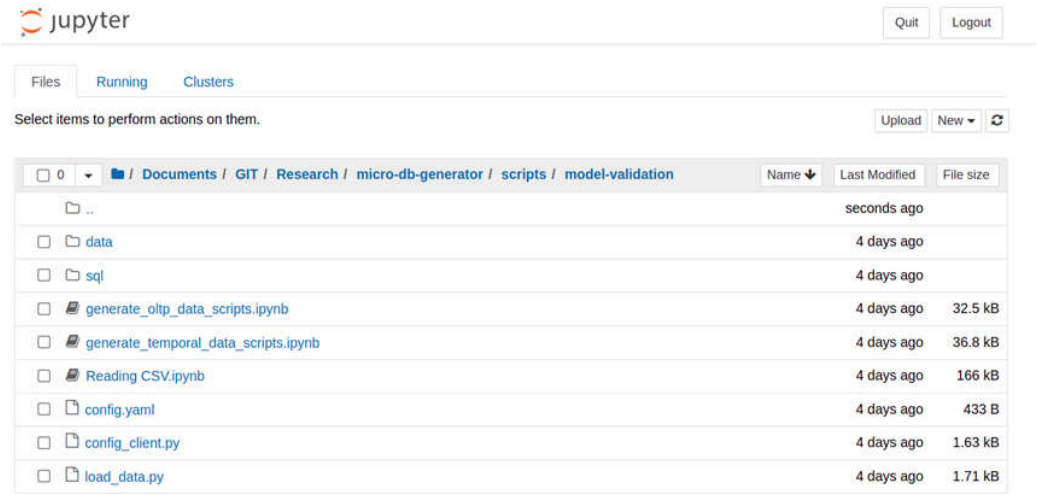


*Figure 10.1: Data Processing File Structure*

After executing the OLTP data generation notebook, "sql/OLTP" folder will be populated with all the necessary SQL insert statement files as follows.

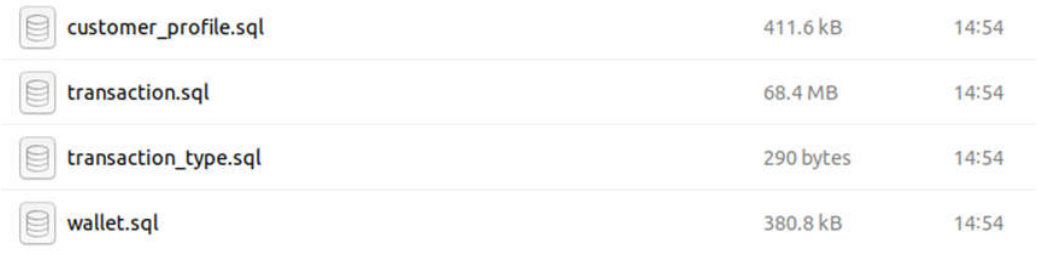


*Figure 10.2: SQL Data Insert Scripts for OLTP Database*

In order to generate transactions for different databases (with different temporal time periods), users need to set the relevant number of temporal days to 'temporalTimePeriod' variable in "Generate_template_data_scripts.ipynb" notebook script. After successful execution, you can see the generated script files under 'sql/TEMPORAL' folder.

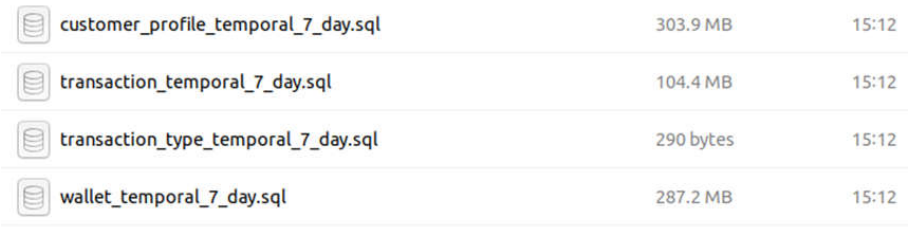


*Figure 10.3: SQL Data Insert Scripts for Temporal Database*

**Data Loading**

We have provided a separate python script 'load_data.py' to load temporal and OLTP data into respective databases. Users need to provide database configurations in 'config.yaml' file. Below is the sample yaml configuration file for OLTP database with three temporal databases.

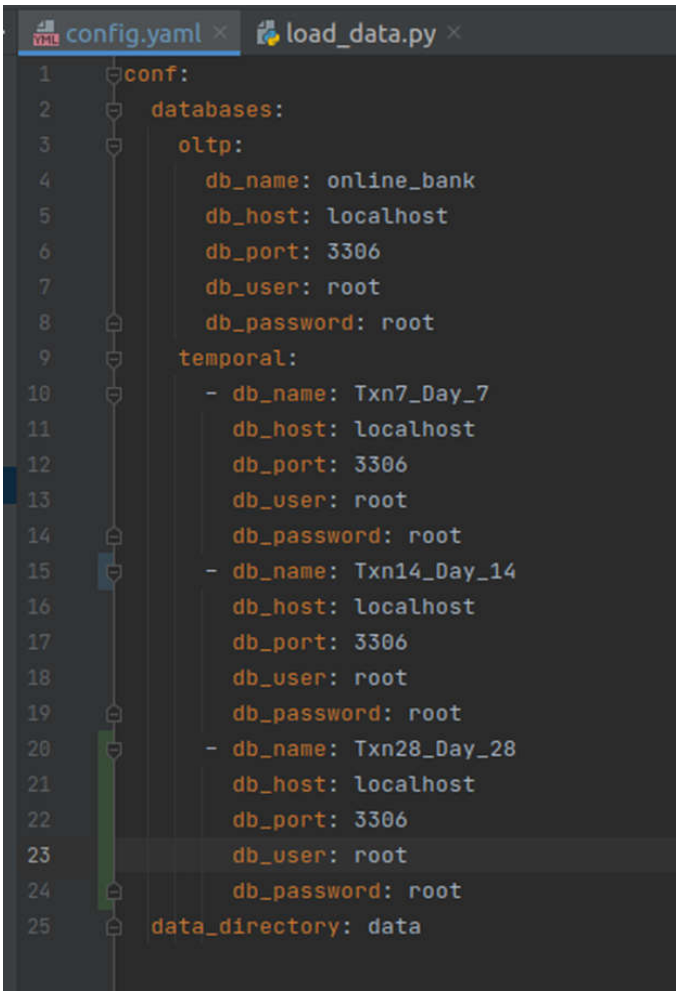


*Figure 10.4: Data Loading Configuration File*

Once everything is configured, 'load_data.py' python script file can execute like a regular python script. Successful start of data import process will show a message like below figure.

```
Database Config Loaded For :online_bank
Query Execution Started...:sql/OLTP/transaction_type.sql
Query Execution Ended...:sql/OLTP/transaction_type.sql
Query Execution Started...:sql/OLTP/customer_profile.sql
Query Execution Ended...:sql/OLTP/customer_profile.sql
Query Execution Started...:sql/OLTP/wallet.sql
Query Execution Ended...:sql/OLTP/wallet.sql
Query Execution Started...:sql/OLTP/transaction.sql
```

*Figure 10.5: Data Loading Started*

# 11 APPENDIX E: MODEL VALIDATION TEST RESULTS SUMMARY

**Experiment 1: Summary**

*Table 11.1: Scoring Rule 1 - Experiment 1 - Summary*

| Scoring Rule | Database Name | Expected Execution Period | Actual Execution Period | Result (AccountId:Frequency) |
|---|---|---|---|---|
| SC 1 | OLTP | 7 Day | 28 Day | 4th Week: 1352:7 |
| | | | | 3rd Week: 1352:7 |
| | | | | 2nd Week: 1352:7 |
| | | | | 1st Week: 1352:7 |
| | Tnx28_Day_28 | 7 Day | 28 Day | 4th Week: 1352:7 |
| | | | | 3rd Week: 1352:7 |
| | | | | 2nd Week: 1352:7 |
| | | | | 1st Week: 1352:7 |
| | Tnx14_Day_14 | 7 Day | 14 Day | 4th Week: 1352:7 |
| | | | | 3rd Week: 1352:7 |
| | Tnx7_Day_7 | 7 Day | 7 Day | 4th Week : 1352:7 |

*Table 11.2: Scoring Rule 2 - Experiment 1 - Summary*

| Scoring Rule | Database Name | Expected Execution Period | Actual Execution Period | Result (AccountId:Frequency) |
|---|---|---|---|---|
| SC 2 | OLTP | 7 Day | 28 Day | 4th Week: Empty |
| | | | | 3rd Week: Empty |
| | | | | 2nd Week: 21:11 |

| | | | | |
|---|---|---|---|---|
| | | | | 27:11 |
| | | | | 1st Week: 25:11 |
| | Tnx28_Day_28 | 7 Day | 28 Day | 4th Week: Empty |
| | | | | 3rd Week: Empty |
| | | | | 2nd Week: 21:11 27:11 |
| | | | | 1st Week: 25:11 |
| | Tnx14_Day_14 | 7 Day | 14 Day | 4th Week: Empty |
| | | | | 3rd Week: Empty |
| | Tnx7_Day_7 | 7 Day | 7 Day | 4th Week : Empty |

*Table 11.3: Scoring Rule 3 - Experiment 1 - Summary*

| Scoring Rule | Database Name | Expected Execution Period | Actual Execution Period | Result (AccountId:Frequency) |
|---|---|---|---|---|
| SC 3 | OLTP | 14 Day | 28 Day | 3rd & 4th Week: 918:8 |
| | | | | 1st & 2nd Week: Empty |
| | Tnx28_Day_28 | 14 Day | 28 Day | 3rd & 4th Week: 918:8 |
| | | | | 1st & 2nd Week: Empty |
| | Tnx14_Day_14 | 14 Day | 14 Day | 3rd & 4th Week: 918:8 |
| | Tnx7_Day_7 | NA | NA | NA |

*Table 11.4: Scoring Rule 4 - Experiment 1 - Summary*

| Scoring Rule | Database Name | Expected Execution Period | Actual Execution Period | Result (AccountId:Frequency) |
|---|---|---|---|---|
| SC 4 | OLTP | 7 Day | 7 Day | 4th Week: 140:2 , 23:2 , 281:2 , 28:2, 33:2 , 35:2 , 297:2 , 192:2 , 200:2 , 346:2 , 91:2 , 94:2 , 994:2 , 104:2 |
| | Tnx28_Day_28 | 7 Day | 7 Day | 4th Week: 140:2 , 23:2 , 281:2 , 28:2, 33:2 , 35:2 , 297:2 , 192:2 , 200:2 , 346:2 , 91:2 , 94:2 , 994:2 , 104:2 |
| | Tnx14_Day_14 | 7 Day | 7 Day | 4th Week: 140:2 , 23:2 , 281:2 , 28:2, 33:2 , 35:2 , 297:2 , 192:2 , 200:2 , 346:2 , 91:2 , 94:2 , 994:2 , 104:2 |
| | Tnx7_Day_7 | 7 Day | 7 Day | 4th Week: 140:2 , 23:2 , 281:2 , 28:2, 33:2 , 35:2 , 297:2 , 192:2 , 200:2 , 346:2 , 91:2 , 94:2 , 994:2 , 104:2 |

## Experiment 2: Summary

*Table 11.5: Scoring Rule 1 - Experiment 2 - Summary*

| Scoring Rule | Database Name | Expected Execution Period | Actual Execution Period | Result (AccountId:Frequency) |
|---|---|---|---|---|
| SC 1 | OLTP | 7 Day | 28 Day | 4th Week: 1352:7 |
| | | | | 3rd Week: 1352:7 |
| | | | | 2nd Week: 1352:7 |
| | | | | 1st Week: 1352:7 |
| | Tnx28_Day_28 | 7 Day | 28 Day | 4th Week: 1352:7 |
| | | | | 3rd Week: 1352:7 |
| | | | | 2nd Week: 1352:7 |
| | | | | 1st Week: 1352:7 |
| | Tnx14_Day_14 | 7 Day | 14 Day | 4th Week: 1352:7 |
| | | | | 3rd Week: 1352:7 |
| | Tnx7_Day_7 | 7 Day | 7 Day | 4th Week: 1352:7 |

*Table 11.6: Scoring Rule 2 - Experiment 2 - Summary*

| Scoring Rule | Database Name | Expected Execution Period | Actual Execution Period | Result (AccountId:Frequency) |
|---|---|---|---|---|
| SC 2 | OLTP | 7 Day | 28 Day | 4th Week: Empty |
| | | | | 3rd Week: Empty |
| | | | | 2nd Week: 33:11 |
| | | | | 1st Week: 34:13 |
| | | 7 Day | 28 Day | 4th Week: Empty |

| | | | | |
|---|---|---|---|---|
| | Tnx28_Day_28 | | | 3rd Week: Empty |
| | | | | 2nd Week: 33:11 |
| | | | | 1st Week: 34:13 |
| | Tnx14_Day_14 | 7 Day | 14 Day | 4th Week: Empty |
| | | | | 3rd Week: Empty |
| | Tnx7_Day_7 | 7 Day | 7 Day | 4th Week: Empty |

*Table 11.7: Scoring Rule 3 - Experiment 2 - Summary*

| Scoring Rule | Database Name | Expected Execution Period | Actual Execution Period | Result (AccountId:Frequency) |
|---|---|---|---|---|
| SC 3 | OLTP | 14 Day | 28 Day | 3rd & 4th Week: 326:8 |
| | | | | 1st & 2nd Week: Empty |
| | Tnx28_Day_28 | 14 Day | 28 Day | 3rd & 4th Week: 326:8 |
| | | | | 1st & 2nd Week: Empty |
| | Tnx14_Day_14 | 14 Day | 14 Day | 3rd & 4th Week: 326:8 |
| | Tnx7_Day_7 | NA | NA | NA |

*Table 11.8: Scoring Rule 4 - Experiment 2 - Summary*

| Scoring Rule | Database Name | Expected Execution Period | Actual Execution Period | Result (AccountId:Frequency) |
|---|---|---|---|---|

| | | | | |
|---|---|---|---|---|
| SC 4 | OLTP | 7 Day | 7 Day | 4th Week: 769:2 , 23:2 , 281:2 , 28:2 , 33:2 , 35:2 , 192:2 , 200:2 , 346:2 , 91:2 , 94:2 , 104:2 |
| | Tnx28_Day_28 | 7 Day | 7 Day | 4th Week: 769:2 , 23:2 , 281:2 , 28:2 , 33:2 , 35:2 , 192:2 , 200:2 , 346:2 , 91:2 , 94:2 , 104:2 |
| | Tnx14_Day_14 | 7 Day | 7 Day | 4th Week: 769:2 , 23:2 , 281:2 , 28:2 , 33:2 , 35:2 , 192:2 , 200:2 , 346:2 , 91:2 , 94:2 , 104:2 |
| | Tnx7_Day_7 | 7 Day | 7 Day | 4th Week: 769:2 , 23:2 , 281:2 , 28:2 , 33:2 , 35:2 , 192:2 , 200:2 , 346:2 , 91:2 , 94:2 , 104:2 |

**Experiment 3: Summary**

*Table 11.9: Scoring Rule 1 - Experiment 3 - Summary*

| Scoring Rule | Database Name | Expected Execution Period | Actual Execution Period | Result (AccountId:Frequency) |
|---|---|---|---|---|
| SC 1 | OLTP | 7 Day | 28 Day | 4th Week: 1352:7 |
| | | | | 3rd Week: 1352:7 |
| | | | | 2nd Week: 1352:7 |
| | | | | 1st Week: 1352:7 |

| | | | | |
|---|---|---|---|---|
| | Tnx28_Day_28 | 7 Day | 28 Day | 4th Week: 1352:7 |
| | | | | 3rd Week: 1352:7 |
| | | | | 2nd Week: 1352:7 |
| | | | | 1st Week: 1352:7 |
| | Tnx14_Day_14 | 7 Day | 14 Day | 4th Week: 1352:7 |
| | | | | 3rd Week: 1352:7 |
| | Tnx7_Day_7 | 7 Day | 7 Day | 4th Week : 1352:7 |

*Table 11.10: Scoring Rule 2 - Experiment 3 - Summary*

| Scoring Rule | Database Name | Expected Execution Period | Actual Execution Period | Result (AccountId:Frequency) |
|---|---|---|---|---|
| SC 2 | OLTP | 7 Day | 28 Day | 4th Week: 22:11 |
| | | | | 3rd Week: Empty |
| | | | | 2nd Week: Empty |
| | | | | 1st Week: 34:11 |
| | Tnx28_Day_28 | 7 Day | 28 Day | 4th Week: 22:11 |
| | | | | 3rd Week: Empty |
| | | | | 2nd Week: Empty |
| | | | | 1st Week: 34:11 |
| | Tnx14_Day_14 | 7 Day | 14 Day | 4th Week: 22:11 |
| | | | | 3rd Week: Empty |
| | Tnx7_Day_7 | 7 Day | 7 Day | 4th Week : 22:11 |

*Table 11.11: Scoring Rule 3 - Experiment 3 - Summary*

| Scoring Rule | Database Name | Expected Execution Period | Actual Execution Period | Result (AccountId:Frequency) |
|---|---|---|---|---|
| SC 3 | OLTP | 14 Day | 28 Day | 3rd & 4th Week: Empty |
| | | | | 1st & 2nd Week: Empty |
| | Tnx28_Day_28 | 14 Day | 28 Day | 3rd & 4th Week: Empty |
| | | | | 1st & 2nd Week: Empty |
| | Tnx14_Day_14 | 14 Day | 14 Day | 3rd & 4th Week: Empty |
| | Tnx7_Day_7 | NA | NA | NA |

*Table 11.12: Scoring Rule 4 - Experiment 3 - Summary*

| Scoring Rule | Database Name | Expected Execution Period | Actual Execution Period | Result (AccountId:Frequency) |
|---|---|---|---|---|
| SC 4 | OLTP | 7 Day | 7 Day | 4th Week: 23:2 , 281:2 , 28:2 , 33:2 , 35:2 , 295:2 , 297:2 , 192:2 , 200:2 , 346:2 , 91:2 , 94:2 , 104:2 |
| | Tnx28_Day_28 | 7 Day | 7 Day | 4th Week: 23:2 , 281:2 , 28:2 , 33:2 , 35:2 , 295:2 , 297:2 , 192:2 , 200:2 , 346:2 , 91:2 , 94:2 , 104:2 |
| | Tnx14_Day_14 | 7 Day | 7 Day | 4th Week: 23:2 , 281:2 , 28:2 , 33:2 , |

| | | | | |
|---|---|---|---|---|
| | | | | 35:2 , 295:2 , 297:2 , 192:2 , 200:2 , 346:2 , 91:2 , 94:2 , 104:2 |
| | Tnx7_Day_7 | 7 Day | 7 Day | 4th Week: 23:2 , 281:2 , 28:2 , 33:2 , 35:2 , 295:2 , 297:2 , 192:2 , 200:2 , 346:2 , 91:2 , 94:2 , 104:2 |

**Experiment 4: Summary**

*Table 11.13: Scoring Rule 1 - Experiment 4 - Summary*

| Scoring Rule | Database Name | Expected Execution Period | Actual Execution Period | Result (AccountId:Frequency) |
|---|---|---|---|---|
| SC 1 | OLTP | 7 Day | 28 Day | 4th Week: 1352:7 |
| | | | | 3rd Week: 1352:7 |
| | | | | 2nd Week: 1352:7 |
| | | | | 1st Week: 1352:7 |
| | Tnx28_Day_28 | 7 Day | 28 Day | 4th Week: 1352:7 |
| | | | | 3rd Week: 1352:7 |
| | | | | 2nd Week: 1352:7 |
| | | | | 1st Week: 1352:7 |
| | Tnx14_Day_14 | 7 Day | 14 Day | 4th Week: 1352:7 |
| | | | | 3rd Week: 1352:7 |
| | Tnx7_Day_7 | 7 Day | 7 Day | 4th Week : 1352:7 |

*Table 11.14: Scoring Rule 2 - Experiment 4 - Summary*

| Scoring Rule | Database Name | Expected Execution Period | Actual Execution Period | Result (AccountId:Frequency) |
|---|---|---|---|---|
| SC 2 | OLTP | 7 Day | 28 Day | 4th Week: 12:11 |
| | | | | 3rd Week: Empty |
| | | | | 2nd Week: 33:11 |
| | | | | 1st Week: 12:11 25:11 |
| | Tnx28_Day_28 | 7 Day | 28 Day | 4th Week: 12:11 |
| | | | | 3rd Week: Empty |
| | | | | 2nd Week: 33:11 |
| | | | | 1st Week: 12:11 25:11 |
| | Tnx14_Day_14 | 7 Day | 14 Day | 4th Week: 12:11 |
| | | | | 3rd Week: Empty |
| | Tnx7_Day_7 | 7 Day | 7 Day | 4th Week : 12:11 |

*Table 11.15: Scoring Rule 3 - Experiment 4 - Summary*

| Scoring Rule | Database Name | Expected Execution Period | Actual Execution Period | Result (AccountId:Frequency) |
|---|---|---|---|---|
| SC 3 | OLTP | 14 Day | 28 Day | 3rd & 4th Week: Empty |
| | | | | 1st & 2nd Week: Empty |
| | | 14 Day | 28 Day | 3rd & 4th Week: Empty |

| | | | | |
|---|---|---|---|---|
| | Tnx28_Day_28 | | | 1st & 2nd Week: Empty |
| | Tnx14_Day_14 | 14 Day | 14 Day | 3rd & 4th Week: Empty |
| | Tnx7_Day_7 | NA | NA | NA |

*Table 11.16: Scoring Rule 4 - Experiment 4 - Summary*

| Scoring Rule | Database Name | Expected Execution Period | Actual Execution Period | Result (AccountId:Frequency) |
|---|---|---|---|---|
| SC 4 | OLTP | 7 Day | 7 Day | 4th Week: 140:2 , 23:2 , 281:2 , 28:2, 33:2 , 35:2 , 297:2 , 192:2 , 200:2 , 346:2 , 91:2 , 94:2 , 104:2 |
| | Tnx28_Day_28 | 7 Day | 7 Day | 4th Week: 140:2 , 23:2 , 281:2 , 28:2, 33:2 , 35:2 , 297:2 , 192:2 , 200:2 , 346:2 , 91:2 , 94:2 , 104:2 |
| | Tnx14_Day_14 | 7 Day | 7 Day | 4th Week: 140:2 , 23:2 , 281:2 , 28:2, 33:2 , 35:2 , 297:2 , 192:2 , 200:2 , 346:2 , 91:2 , 94:2 , 104:2 |
| | Tnx7_Day_7 | 7 Day | 7 Day | 4th Week: 140:2 , 23:2 , 281:2 , 28:2, 33:2 , 35:2 , 297:2 , 192:2 , 200:2 , 346:2 , 91:2 , 94:2 , 104:2 |

## Experiment 5: Summary

*Table 11.17: Scoring Rule 1 - Experiment 5 - Summary*

| Scoring Rule | Database Name | Expected Execution Period | Actual Execution Period | Result (AccountId:Frequency) |
|---|---|---|---|---|
| SC 1 | OLTP | 7 Day | 28 Day | 4th Week: 1352:7 |
| | | | | 3rd Week: 1352:7 |
| | | | | 2nd Week: 1352:7 |
| | | | | 1st Week: 1352:7 719:7 |
| | Tnx28_Day_28 | 7 Day | 28 Day | 4th Week: 1352:7 |
| | | | | 3rd Week: 1352:7 |
| | | | | 2nd Week: 1352:7 |
| | | | | 1st Week: 1352:7 719:7 |
| | Tnx14_Day_14 | 7 Day | 14 Day | 4th Week: 1352:7 |
| | | | | 3rd Week: 1352:7 |
| | Tnx7_Day_7 | 7 Day | 7 Day | 4th Week: 1352:7 |

*Table 11.18: Scoring Rule 2 - Experiment 5 - Summary*

| Scoring Rule | Database Name | Expected Execution Period | Actual Execution Period | Result (AccountId:Frequency) |
|---|---|---|---|---|
| SC 2 | OLTP | 7 Day | 28 Day | 4th Week: 62:11 |
| | | | | 3rd Week: Empty |
| | | | | 2nd Week: 21:11 |
| | | | | 1st Week: 14:11 |

| | | | | |
|---|---|---|---|---|
| | Tnx28_Day_28 | 7 Day | 28 Day | 4th Week: 62:11 |
| | | | | 3rd Week: Empty |
| | | | | 2nd Week: 21:11 |
| | | | | 1st Week: 14:11 |
| | Tnx14_Day_14 | 7 Day | 14 Day | 4th Week: 62:11 |
| | | | | 3rd Week: Empty |
| | Tnx7_Day_7 | 7 Day | 7 Day | 4th Week: 62:11 |

*Table 11.19: Scoring Rule 3 - Experiment 5 - Summary*

| Scoring Rule | Database Name | Expected Execution Period | Actual Execution Period | Result (AccountId:Frequency) |
|---|---|---|---|---|
| SC 3 | OLTP | 14 Day | 28 Day | 3rd & 4th Week: Empty |
| | | | | 1st & 2nd Week: Empty |
| | Tnx28_Day_28 | 14 Day | 28 Day | 3rd & 4th Week: Empty |
| | | | | 1st & 2nd Week: Empty |
| | Tnx14_Day_14 | 14 Day | 14 Day | 3rd & 4th Week: Empty |
| | Tnx7_Day_7 | NA | NA | NA |

*Table 11.20: Scoring Rule 4 - Experiment 5 - Summary*

| Scoring Rule | Database Name | Expected Execution Period | Actual Execution Period | Result (AccountId:Frequency) |
|---|---|---|---|---|
| SC 4 | OLTP | 7 Day | 7 Day | 4th Week: 768:2 , 134:2 , 18:2 , 34:2 , |

| | | | | |
|---|---|---|---|---|
| | | | | 964:2 , 327:2 , 343:2 , 103:2 |
| | Tnx28_Day_ 28 | 7 Day | 7 Day | 4th Week: 768:2 , 134:2 , 18:2 , 34:2 , 964:2 , 327:2 , 343:2 , 103:2 |
| | Tnx14_Day_ 14 | 7 Day | 7 Day | 4th Week: 768:2 , 134:2 , 18:2 , 34:2 , 964:2 , 327:2 , 343:2 , 103:2 |
| | Tnx7_Day_7 | 7 Day | 7 Day | 4th Week: 768:2 , 134:2 , 18:2 , 34:2 , 964:2 , 327:2 , 343:2 , 103:2 |