1 Experiments

Note that, since the way we did the transaction generator (coming from wisabi database client's behavior), the average number of transactions per day per card is ~ 1, and therefore to be able to generate a transaction set with anomalous situations more close to reality, a reasonable time interval size for the generated transaction stream would be having T around some weeks or month(s).

1.1 1st option: Real time-event stream simulation

Since we do not have the material time to run each experiment for a interval time T of some weeks or a month the idea is to do time scaling of the time event stream.

Take the stream of a certain time interval size T and map it into a smaller time interval T' where $T' << T$ so that:

- **Shorter experimental time**: Reduced time to test the system behavior. Instead of T, only T' time to test it.
- **Stress testing**: We do not test the system under a real-case scenario considering its number of cards c, instead we are testing it under a higher load to what it would correspond, but having c cards, and therefore c filter's subgraph.
- **Graph database size VS amount of filters' subgraphs**: Although we simulate a higher load, the number of filters' subgraph corresponds to c the number of cards. The benefit is that we do not need to have such a big graph database.

The consequences for the experiments and metrics:

- **Diefficiency metrics** (continuous delivery of results): If we give the input stream to the system respecting the temporal timestamps, note that no matter the system characteristics, that a result (an alert in our case), will not be possible to be produced until the event causing it arrives to the system. Therefore the emission of events is expected to be really similar in this case, for any system variation. Only in the case when the stream load is high enough we expect to see some differences??
- **Response time**: having in mind the previous considerations, we think in measuring the possible differences of behavior of the different system capabilities in terms of the mean response time. The mean response time (`mrt`) would be the average time that the system spends since it receives the transactions involved in an alert until the time it emits the alert.

Problems derived to pay attention to:

- Shrinking the timestamps to a smaller time interval, produces the emergence of not real fraud patterns that before did not exist due to their real and "correct" larger time distance. Example:

  - Consider the original size of the time interval of the input stream $T = 120h$ (5 days) and $T' = 24h$.
  - Consider two consecutive regular transactions of a certain client performed in two different ATMs `ATM-x` and `ATM-y` with `t_min`= 8h (minimum time difference to traverse the distance from `ATM-x` to `ATM-y`) and `t_diff`= 24h (time difference between the first and the second transaction).
  - → Note that with the scaling the time difference `t_diff` would be of 5 times less, that is, `t_diff` $= 4.8h$. Therefore this will make `t_diff'` $= 4.8h <$ `t_min` $= 8h$.

- → (*) Solution A: **introduce the scaling factor as a input parameter** and consider it also for the fraud checking so to properly **scale the `t_min` variable** (`t_min` $= 8h \rightarrow$ `t_min'` $= \frac{8}{5}h = 1.6h$) and therefor:
  - Before scaling: `t_diff` $= 24h >$ `t_min` $= 8h$.
  - After scaling (scale factor $= \frac{1}{5}$): `t_diff` $= \frac{24}{5} = 4.8h >$ `t_min` $= \frac{8}{5} = 1.6h$.

- → Solution B: conserve the original timestamps, and consider the mapped-reduced timestamps for simulating the arrival times of the transactions into the system while taking the original timestamps for the checking of the frauds.

1.2 2nd option: real timestamp omission

Do not consider the real-time simulation, by omitting the transaction timestamps in the sense that we do not consider them to simulate a real case scenario where each transaction arrives to the system at the time indicated by its timestamp. Instead all the stream comes (ordered by timestamp) but directly (almost) at the same time to the system. With this approach:

- **No real case simulation**
- **Measure the load the system can take**: for the different system variations given a same stream.
- **Diefficiency metrics**: since time arrival of the transactions to the system is now ignored, and all the transactions come one after the other, a result to be produced do not need to wait for the real timestamp of the transaction. Therefore, we could see the differences in continuously delivering results of the different systems under the same input stream load (more clear than before).

Some (other) references:

- Apache Flink: distributed processing engine for stateful computation of data streams.