SYSTEM AND METHOD FOR ADAPTIVELY TRACKING A PATTERN OF A TRANSACTION

[0001] The present invention relates generally to payment technology and, in particular, to a system and method for adaptively tracking a pattern of a transaction initiated by a transaction request.

Background

[0002] With some companies, aggregates for fraud prevention from the attackers using different payment instruments are used to detect lost or stolen cards for illicit earnings. Aggregates are used in different models to detect anomalies and decline potential fraud transactions. The aggregates are also used in rules to set hard limits on different data points to reduce finance loss. Rules are of the format - decline a transaction if the user has already done 100 transactions in the last 30 days, or if the user has done transactions to 100 different merchants in the last one week.

[0003] Conventionally, aggregates and rules are evaluated during each transaction. The aggregates are expected to be evaluated with least latency and minimal network and memory overload. Therefore, there is a need to ameliorate one or more of the above problems.

Summary

[0004] According to a first aspect of the present disclosure, there is provided a method for adaptively tracking a pattern of a transaction initiated by a transaction request, the transaction request including at least one data, the method comprising: querying, from a database, predetermined data corresponding to one or more features and/or combinations of features for a plurality of time periods; wherein the predetermined data is determined from historical transactions conducted over a period of time by the user; determining that data from the transaction request corresponds to a feature or combination of said one or more features and/or combinations of features; updating distinct counts for the feature or combination by: decreasing a distinct count for the feature or combination for a time period corresponding to the last historical transaction data; and increasing a distinct count for the

feature or combination for a time period corresponding to the transaction request; and adaptively tracking a spending pattern of a user in response to an outcome of the determination of the data from transaction request.

[0005] According to a second aspect of the present disclosure, there is provided a server for adaptively tracking a pattern of a transaction initiated by a transaction request, the server comprising: at least one processor; and at least one memory including computer program code; the at least

one memory and the computer program code configured to, with the at least one processor, cause the server at least to: query, from a database, predetermined data corresponding to one or more features and/or combinations of features for a plurality of time periods; wherein the predetermined data is determined from historical transactions conducted over a period of time by the user; determine that data from the transaction request corresponds to a feature or combination of said one or more features and/or combinations of features; updating distinct counts for the feature or combination by: decreasing a distinct count for the feature or combination for a time period corresponding to the last historical transaction data; and increasing a distinct count for the feature or combination for a time period corresponding to the transaction request; and adaptively tracking a pattern of a transaction in response to an outcome of the determination of the data from transaction request.

[0005a] The invention according to the above aspects advantageously provides the exact distinct count-based aggregates, in constant time without actually loading or scanning through the distinct values.

Brief Description of the Drawings

[0006] Embodiments and implementations are provided by way of example only, and will be better understood and readily apparent to one of ordinary skill in the art from the following written description, read in conjunction with the drawings, in which:

[0007] Fig. 1 is a block diagram of a system for managing a transaction in accordance with an aspect of the present disclosure.

[0008] Fig. 2 is a flow chart diagram of a method for adaptively tracking a pattern of a transaction initiated by a transaction request by the transaction processing server in the system of Fig. 1.

[0009] Fig. 3 shows a flow chart diagram of a sub-process in relation to historical transaction data in the method of Fig. 2.

[0010] Fig. 4 shows a flow chart diagram of a sub-process of the method of Fig. 2.

[0011] Fig 5a shows how aggregates are maintained for transactions for (user and merchant) and (user) as composite keys at day level and Fig 5b shows aggregates are updated for (user and merchant) and (user) as composite keys at day level shown in Fig 5a.

[0012] Fig. 6 shows a topographical view of features that are used to define the order of updates for multiple aggregates for a transaction.

[0013] Figs. 7 and 8 form a schematic block diagram of a computer system upon which a transaction processing server in the system of Fig. 1 can be practiced

[0014] Fig. 9 shows an example of a computing device to realize the transaction processing server shown in Fig. 1.

Detailed Description

Terms Description

[0015] Transaction request - A transaction request is a request generated by a user via, for example, a user device 120, in order to initiate a transaction. The transaction may be one that is for purchasing a product such as a good or service from a merchant. The transaction may also be one that is for utilising a credit-based financial product such as a credit card, a "buy now pay later" financial product, or other similar ways to purchase a good or service. The transaction request may comprise transaction data, the transaction data representing a merchant at which the user transacts, at least one of a product that the user wishes to purchase or utilise, and a category or product family of the product. The transaction data may be information to determine or update a rolling distinct or a distinct count. The transaction data may also include a transaction amount which is an amount that is required for payment for the product. The transaction request may also include user information which represents a type of transaction the user is allowed to transact. The user information may be information relating to what type of transaction that the user is allowed to transact, and/or what type of transaction that the user is blocked from transacting. The user information may be determined based on past transactions of the user for purchasing a product from a merchant, and/or for utilising a credit-based financial product to purchase a good or service. The user information may be available for users who are new customers to a merchant or credit-based financial product provider without any prior transaction history, wherein the indicated type of transaction in the user information may be based on a default setting (e.g. all types of transactions are allowed, or only debit transactions allowed, or credit transactions of up to a certain amount are allowed, or credit transactions involving payment instalments are/are not allowed, and other settings), or may be based on transaction history of the user from other similar merchants. The transaction request is initiated by the user device 120 and is propagated to the transaction processing server 110. In various embodiments, the transaction request may be propagated via an application programming interface ("API"). Example of APIs include the Representational State Transfer (REST) API, and the like.

[0016] Historical transactions – Historical transactions relate to past transactions that are initiated by a user. Historical transaction data may include information of past transactions relating to purchase of products, and/or relating to credit-based financial products used for purchasing goods and/or services. The information may include the transaction amount, merchant, product details, a category of the product, success of transaction e.g. whether the transaction is rejected or approved, date and time of the transaction, payment status of the transaction (e.g. paid, unpaid, still paying via payment instalments, etc) and other similar information. Historical transaction data may be available for existing customers of a merchant and/or provider who have previously purchased from the merchant and/or utilised a financial product from the provider, and such data may then be used by the payment processing server 110 for assessing whether a transaction request can proceed or not.

[0017] Rolling distinct –Rolling distinct is a data that is indicative of a transaction. In an embodiment, a rolling distinct for a feature or relating to a transaction can be modified e.g., decrement it for the last day the feature is used and increment it for the day of transaction. In an example, there may be two sets defined for a transaction type for composite keys or predetermined data - (feature1, feature2) and (feature1), and a distinct count of feature2 associated with a given (feature1) in the last 30 days. The rolling distinct for feature2 will be maintained in (feature1) set. Summation of this rolling distinct for the last 30-time blocks will result in a distinct count of feature2 associated with feature1. For example, feature1 can be user and feature2 can be merchants, and embodiments will be used to provide a distinct count of merchants associated with a user in the last 30 days. More details will be provided in the following.

[0018] Aggregate – Aggregate represents data for one combination of data points. One set in aerospike represents one aggregate. Two sets will be used to identify if a rolling distinct needs to be updated and the day for which it needs to be decremented. More details can be found in the following. An aggregate may be represented by a series of symbols. On receiving a new transaction, firstly it will update data points in (feature1, feature2) set including count, amount, and so on. Increment operations are used to update all data points configured for one set in an aerospike call. Aerospike returns the new incremented value in the increment operation call. This output of new count in (feature1, feature2) will be used to identify if rolling distinct needs to be updated for this transaction.

[0019] Payment network server – The payment network server is a server that hosts software application programs for processing payment of a transaction request. The payment network server may be one that is used to process payment transactions. The payment network server communicates with a token vault (if required), and any other servers (e.g., an issuer server, an acquirer server) to facilitate payment of transaction requests generated by user devices. Payment network servers may use a variety of different protocols and procedures in order to process the transaction requests. Transactions that may be performed via a payment network server include product or service purchases, credit purchases, debit transactions, fund transfers, account withdrawals, etc. Payment network servers may be configured to process transactions via cash-substitutes, which may include payment cards, letters of credit, checks, payment accounts, etc. The payment network server is operated by a payment network operator. The payment network operator may be an entity (e.g. a company or organization) who operates to process transactions, clear and settle funds for payments between two entities (e.g. two banks). The payment network server may include one or more computing devices that are used for processing transactions.

[0020] Payment account – A financial account that can be used for performing transactions. A payment account may include a payment card, such as a credit card, debit card, prepaid card, savings account, checking account, charge card, membership card, promotional card, frequent flyer card, identification card, gift card, and/or may also relate to a digital wallet (which is also known as an e-wallet), and/or may also relate to a 'buy now pay later' financial product. A digital wallet is typically used to store various forms of electronic money to aid in completing e-commerce transactions (or wallet-based transactions). For example, it is possible to link / register a payment card to a digital wallet to perform a wallet-based transaction. It will be understood that each type of these payment account can be used as a method of payment for the system described herein. The payment account includes account information such as an account number, account owner, and the like.

[0021] A user account – an account of a user that is registered with the transaction processing server 110. The user account is associated with the user information for the purpose of purchasing goods and/or services (or a product). The user account includes a payment account of the user.

[0022] A merchant account – an account of a merchant that is registered with the transaction processing server 110. The merchant account includes a payment account of the merchant. In various embodiments, the transaction processing server 110 may be a server of the merchant, such that a merchant account is not necessary.

Example Implementations

[0023] Where reference is made in any one or more of the accompanying drawings to steps and/or features, which have the same reference numerals, those steps and/or features have for the purposes of this description the same function(s) or operation(s), unless the contrary intention appears.

[0024] It is to be noted that the discussions contained in the "Background" section and that above relating to prior art arrangements relate to discussions of devices which form public knowledge through their use. Such should not be interpreted as a representation by the present inventor(s) or the patent applicant that such devices in any way form part of the common general knowledge in the art.

[0025] Fig. 1 shows a system 100 for managing a transaction comprising a transaction processing server 110, merchant devices 120A to 120N, user devices 130A to 130N, an acquirer server 138, a payment network server 140, and an issuer server 142.

[0026] The transaction processing server 110 is a server that hosts software application programs 233 to function as a transaction management system. The structural context of the transaction processing server 110 is discussed below in relation to Figs. 7 and 8.

[0027] In the illustrative embodiment, the transaction processing server 110 provides an interface to enable communication with each of the devices 120, 130, 150 and the server 138. The transaction processing server 110 provides an application programming interface (“API”) to facilitate such communication. Such APIs may be part of a user interface that may include graphical user interfaces (GUIs), Web-based interfaces, programmatic interfaces such as application programming interfaces (APIs) and/or sets of remote procedure calls (RPCs) corresponding to interface elements, messaging interfaces in which the interface elements correspond to messages of a communication protocol, and/or suitable combinations thereof. Examples of APIs include the REST API, and the like.

[0028] The transaction processing server 110 is configured to receive and send commands and data from and to the user devices 130A to 130N and the merchant devices 120A to 120N.

[0029] The transaction processing server 110 is also configured to receive a transaction request initiated by a user from the user devices 130A to 130N. The transaction request may also be received from merchant devices 120A to 120N in scenarios wherein the transaction request is sent from a user device to a merchant device, and then forwarded from the merchant device to the transaction processing server 110. The transaction processing server also determines if the transaction request includes any features (eg. merchant) identified in an aggregate, and is necessary for calculating a distinct count.

[0030] The transaction processing server 110 then determines if the user is authorized to proceed with the transaction based on a transaction data included in the transaction request and user information, the transaction data representing at least one of a product, and a category of the product, the user information representing a type of transaction the user is allowed to transact. The transaction processing server 110 then proceeds with the transaction in response to the determination if the user is authorized to proceed with the transaction.

[0031] In one arrangement, the transaction management system implemented in the transaction processing server 110 is compatible with existing payment apps, digital wallets, payment gateways, and the like. In another arrangement, the transaction management system implemented in the transaction processing server 110 is an additional function on a payment app or digital wallet.

[0032] The transaction processing server 110 may be operated by any entity (e.g. a company or organization), which may include a merchant, an acquirer, a payment network server provider, and the like. If the acquirer operates the transaction processing server 110, then the acquirer may integrate the acquirer server 138 with the transaction processing server 110. If the payment network server provider operates the transaction processing server 110, then the payment network server provider may integrate the payment network server 140 with the transaction processing server 110.

[0033] The merchant devices 120A to 120N are devices that are used by merchants who are registered with the transaction processing server 110. The merchants offer goods and/or services (or products). Examples of the merchant devices 120A to 120N are tablets, laptops, desktop

computers, smartphones, point-of-sales terminals, and the like. In the present disclosure, the merchant devices 120A to 120N do not refer specifically to point-of-sales terminals. The merchant devices 120A to 120N respectively connect with the transaction processing server 110 via connections 271. The connections 271 may be wired or wireless connections. Hereinafter, the merchant devices 120A to 120N are collectively referred to as the merchant devices 120 (when referring to all of the merchant devices) and the merchant device 120 (when referring to a single merchant device). As described above, the merchant devices 120 can communicate with the transaction processing server 110. The merchant devices 120 can provide commands to the transaction processing server 110 in the form of voice, text, and the like.

[0034] The user devices 130A to 130N are devices that are used by consumers who are registered with the transaction processing server 110. The consumers are users who wish to buy goods or services from the merchant that offers goods or services (see the discussion on the merchant devices 120). Examples of the user devices 130A to 130N are tablets, laptops, desktop computers, smartphones, and the like. The user devices 130A to 130N respectively connect with the transaction processing server 110 via connections 270. The connections 270 may be wired or wireless connections. Hereinafter, the user devices 130A to 130N are collectively referred to as the user devices 130 (when referring to all of the user devices) and the user device 130 (when referring to a single user device). As described above, the user devices 130 can communicate with the transaction processing server 110. The user devices 130 can provide commands to the transaction processing server 110 in the form of voice, text, and the like.

[0035] Both the merchant devices and user devices may include identifiers to identify each merchant or user device for the purposes of updating a distinct count or rolling distinct.

[0036] The transaction processing server 110 is in communication with the acquirer server 138. The acquirer server 138, in turn, is in communication with the payment network server 140. The payment network server 140, in turn, is in communication with the issuer server 142.

[0037] The acquirer server 138 generally is associated with an acquirer who may be an entity (e.g. a company or organization) which issues (e.g. establishes, manages, administers) a payment account (e.g. a financial bank payment account) of the merchant. Examples of the acquirer include a bank and/or other financial institution. The acquirer server 138 may include one or

more computing devices that are used to establish communication with another server by exchanging messages with and/or passing information to the other servers 110 and 140.

[0038] The payment network server 140 is a server that hosts software application programs for processing payment of a transaction request using a payment method specified by the user device 130. The payment network server 140 is one that is used to process transactions. As would be understood, the payment network server 140 communicates with a token vault (if required), and any other servers (e.g., an issuer server) to facilitate payment of transaction requests.

[0039] The issuer server 142 generally is associated with an issuer and may include one or more computing devices that are used to perform a payment transaction. The issuer may be an entity (e.g. a company or organization) which issues (e.g. establishes, manages, administers) a payment account (e.g. a financial bank payment account) of a user. The issuer server may also be able to communicate with the transaction processing server 110 to provide information such as user information, historical transaction data, credit risk limit and other similar data that may be utilised by the transaction processing server 110 for determining whether a transaction request can proceed or not.

[0040] The transaction processing server 110 of the system 100 can replace a physical payment terminal (e.g., a point-of-sale terminal). Therefore, instead of a consumer interacting with a physical payment terminal, the consumers through their user devices 130 can interact with the transaction processing server 110 for purchasing goods and services. In various embodiments, the transaction processing server 110 is one that is configured to query, from a database, a predetermined data. The predetermined data (or features or composite keys) is one that is determined from historical transactions conducted over a period of time (e.g., 1 week or 30 days from the date of transaction). The operation of the transaction processing server 110 will be discussed in more detail below.

Operations of the transaction processing server 110

<u>On-Boarding of Users and Merchants</u>

[0041] Before a user or a merchant can use the transaction processing server 110, the user or the merchant may be required to register with the transaction processing server 110. The registration

step is called on-boarding. The merchant device 120 and the user device 130 may separately perform on-boarding to the transaction processing server 110.

[0042] The on-boarding process for a user is performed by the user through one of the user devices 130. In one arrangement, the user downloads an app of the transaction management system to the user device 130. In another arrangement, the user accesses a website of the transaction management system on the user device 130. As described below in relation to Figs. 8 and 9, the API is part of the software application program 233. Once the user accesses the app or website on the user device 130, the user is able to interact with the transaction processing server 110 to register. The on-boarding process for a merchant is similarly performed.

[0043] A user relating a merchant on-boards the merchant by registering with the transaction processing server 110 through an app, a website, and the like of the transaction processing system on the merchant device 120. For ease of discussion, a user (e.g., a CEO, a CFO, a CTO, a company secretary, or any authorized person) relating to the merchant is referred to simply as a merchant. A merchant uses a merchant device 120 to register with the transaction processing server 110. Details of the registration include, for example, name of the merchant, merchant type, payment server to use, merchant payment account, merchant devices 120 that are authorized to send data for transaction requests such as transaction data, user information, user credit risk limit, historical transaction data type of payment, and the like. The details of the registration include any other data required by the acquirer associated with the merchant.

[0044] In an arrangement, only the merchant may be required to on-board the payment processing server 110, so that any transaction involving the merchant's products would automatically go through the transaction processing server 110. In this case, the user may not be required to do any on-boarding. Transaction requests relating to the merchant's products may be sent directly from a user device to the payment processing server 110, or sent from the merchant device to the payment processing server 110.

[0045] In an arrangement, the transaction processing server 110 may set up rules on rolling distinct criteria for certain users, based on certain features or transaction request information. For example, rolling distinct may be a new data point, representing the count of distinct features used in a historical transaction. In other words, if a rolling distinct representing merchants for a user is 3 for a day, it means the user has done transactions to 3 different merchants on that day and the

user has not performed any transactions to these 3 merchants after that. So the count of distinct features for the last x days can be answered by adding rolling distinct for the last x days.

[0046] Advantageously, a pattern of transactions can be adaptively tracked based on the rolling distinct, and is used to monitor preferences of a user or a success rate of a promotion that is run by a merchant.

[0047] Once on-boarded, the merchant would have a merchant account that stores all the registration details and rules regarding transactions i.e user information and credit risk limits for new/existing users. Authorized merchant devices 120 are capable of sending transaction data to the transaction processing server 110 and, if applicable, initiating payment of the broadcast transaction records that are associated with particular user accounts. Historical transaction data may be shared by the merchant with the transaction processing server 110. In another arrangement, historical transaction data may be gathered and compiled from other sources for the transaction processing server 110 to assess whether a transaction request can proceed.

[0048] A user (e.g., a consumer) may on-board, via the user device 130, by registering with the transaction processing server 110 through an app, a website, and the like of the transaction processing server 110. A user uses a user device 130 to register with the transaction processing server 110. Details of the registration include, for example, name of the user, user payment account, user devices 130 that are authorized to request a transaction, pre-approved maximum transaction amount that the user has set, allowed payment scheme that the user has set, and the like. Once on-boarded, the transaction processing server 110 creates a user account for the user. In one arrangement, it may be possible for a user to create multiple user accounts on the transaction processing server 110. Authorized user devices 130 are then capable of transmitting a transaction request to the transaction processing server 110 for purchasing a product from a merchant.

[0049] In an arrangement, only the merchant may be required to on-board the payment processing server 110, so that any transaction involving the merchant's products would automatically go through the transaction processing server 110. In this case, the user may not be required to do any on-boarding. Transaction requests relating to the merchant's products may be sent directly from a user device to the payment processing server 110, or sent from the merchant device to the payment processing server 110.

[0050] In one alternative arrangement, a user can use an existing user account, which is issued by an issuer (e.g., a bank, a financial institution, etc.) or an entity (e.g., Mastercard®, etc.), to use the transaction processing server 110. In this alternative arrangement, the transaction processing server 110 can connect into a core user management system of the issuer to enable the user to utilise the transaction processing server 110. Further, in this alternative arrangement, the user can use a Secure Remote Commerce (SRC) compatible app such as Masterpass® to access the transaction processing server 110.

[0051] Pre-approved maximum transaction amount relates to an amount that the user pre-approves. For example, if a transaction amount is above a pre-approved maximum transaction amount of $50, then a transaction request on the transaction processing server 110 that involves a transaction amount of more than $50 may require further confirmation from the user in order to proceed. Such confirmation may be requiring the user to verify a code sent via SMS to the user device (or another device registered to the user) to proceed, or other similar verification methods. This advantageously enables the user to safeguard against unauthorised transactions that may be made by a fraudulent user using the user's account.

[0052] Allowed payment scheme includes details relating to payment instalments or other credit-based methods of transacting.

[0053] In one arrangement, a user token is generated upon creation of the user account so that the details of the user payment account are not stored on the transaction processing server 110. The generation of the user token is performed by the transaction processing server 110 in conjunction with a token vault (not shown). The token vault may be operated by a third party or an institution (e.g., a bank) to which the user payment method is related.

Managing transactions

[0054] The discussion now turns to a method of managing a transaction initiated by a user. The transaction may be initiated by a transaction request, which is sent from a user device to the transaction processing server.

[0055] Fig. 2 is a flow chart diagram of a method of for adaptively tracking a pattern of a transaction initiated by a transaction request by the transaction processing server in the system of Fig. 1. The method shown in Fig. 2 begins with step 302 by querying, from a database, a

predetermined data; the predetermined data is one that is determined from historical transactions conducted over a period of time by the user. Step 304 follows step 302 in which the method includes determining if data from the transaction request corresponds to the predetermined data. Step 306 follows step 304 in which the method adaptively tracks a pattern of a transaction initiated by a transaction request in response to an outcome of the determination of the data from transaction request.

[0056] Fig. 3 shows a flow chart diagram of the method 400 of managing a transaction initiated by a user. The method 400 is a software application program 233 (which contains the transaction management system) that is executable by the processor 205 of the transaction processing server 110.

[0057] The method 400 commences at step 310 by determining if a user is authorized to proceed with a transaction based on a transaction data included in a transaction request and user information, the transaction data representing at least one of a product, a merchant and a category of the product, the user information representing a type of transaction the user is allowed to transact. In various examples, at least one of the transaction data and the user information can be referred to as a feature of a transaction. The transaction is initiated by a user via generating and sending a transaction request from a user device 130 to the transaction processing server 110. The transaction request may also be sent from the user device 130 to the merchant device 120, and then forwarded to the transaction processor server 110.

[0058] The transaction data may indicate at least one of a product, a merchant and a category of the product, that the user intends to purchase via the transaction request. For example, the transaction request may be for purchasing a can of tomatoes, and the transaction data thus indicates the can of tomatoes as the product, and a category for the product such as canned food, groceries, or other similar categories. The transaction data may also indicate a method of payment that the user intends to utilise. For example, the user may wish to purchase the can of tomatoes utilising a ‘buy now pay later’ financial product, and the transaction data indicates the ‘buy now pay later’ financial product as the method of payment. In another embodiment, the transaction data may indicate at least one of a financial product that the user intends to use for a purchase, and a category of the financial product. For example, the transaction data may indicate that the user intends to utilise a financial product for paying instalments over a period of time, and the category of the financial product may be indicated as ‘pay later products’.

[0059] User information may be included in the transaction request, and/or retrieved from a merchant device 120 or from a database (not shown). The user information represents a type of transaction that the user is allowed to transact. For example, the user information relating to the user initiating the transaction may indicate that the user is authorised to purchase on a credit basis e.g. by using a credit-based financial product. At step 312, the transaction processing server proceeds with the transaction in response to the determination if the user is authorized to proceed with the transaction. The determination may be based on comparison of the user information and the transaction data. For example, the transaction processing server 110 may allow the purchase of the canned tomatoes to be completed since the user information indicates that the user is authorised to pay for the canned tomatoes using instalment payment. In an event where a particular product, product category, or payment method represented in the transaction data is blacklisted based on the user information, the transaction processing server 110 may not allow the transaction to be completed.

[0060] In various embodiments, the step of determining whether the user is authorized to proceed with the transaction may comprise consideration of other types of data. For example, Fig. 4 illustrates a flow chart diagram of a method 500 for updating a rolling distinct. The method 500 is a software application program 233 (which contains the transaction management system) that is executable by the processor 205 of the transaction processing server 110.

[0061] The method 500 commences at step 502. At step 504, a transaction request is received. At step 506, aggregates that are defined for the transaction type is arranged in topographical order. For example, the aggregates may be arranged in a predetermined order based on dependency. At step 508, increment operations may be prepared for each aggregate based on count or amount over a predetermined period of time, eg. day, month or year. At step 510, it is determined if a distinct datapoint is defined for each aggregate.

[0062] At step 512, when it is determined that a distinct datapoint (or rolling distinct) is defined for an aggregate, it is determined if it is the first time the aggregate appears or count 1 for the aggregate.

[0063] At step 514, when it is determined that it is the first time the aggregate appears, get an aerospike record from an aerospike database. If the count is anything but 1, it suggests the

association of feature1 and feature2 has already been seen for the time block and hence the distinct count need not be updated now. This may correspond to step 302 in Fig. 2.

[0064] At step 516, the method comprises deriving the last time the combination of features are used together in a transaction. This step is carried out to determine, for example, if the user has transacted at the merchants before.

[0065] At step 518 the method comprises preparing increment and decrement operations for the current datapoints relating to the transaction request that is received in step 504. This step may correspond to the steps in Fig. 4. More details will be provided in Fig 5a and Fig 5b below.

[0066] In response to step 510, when it is determined that a distinct datapoint (or rolling distinct) is not defined for an aggregate, it proceeds to step 520 where the method executes prepared operations in aerospike.

[0067] At step 526, it is determined if there is any aggregate dependent on the aggregate for distinct count.

[0068] At step 528, when it is determined that if there is aggregate dependent on the aggregate for distinct count, it proceeds to step 530. In an example, the method continues to monitor if there is another occurrence of the aggregate in the transactions that follow.

[0069] At step 532, when it is determined that if there is no aggregate dependent on the aggregate for distinct count, it proceeds to step 532.

[0070] Fig 5a shows aggregates maintained for transactions for (user and merchant) and (user) composite keys, at day level and Fig 5b shows updated state of aggregates shown in Fig 5a. One transaction type requires aggregates for multiple composite keys or features. One set is maintained for each composite key in aerospike. Each record maintains aggregates at various time periods, for example, day, month, year level, to deliver aggregates for various durations like last 7 days, last 3 months, current year, and so on. On receiving a transaction request, all sets or historical transactions are updated with a new count, amount or other data points configured, for all the time levels.

[0071] As mentioned in the above, maintaining a rolling distinct or predetermined data for a feature, may require the below two changes to keep the definition intact:

1. Decrement it for the last day or last transaction in which the feature is used.
2. Increment it for the day of transaction or when the transaction request is received.

In an example, two sets are defined for a transaction type for composite keys having different features, e.g., (i) (feature1, feature2) and (ii) (feature1), and a distinct count of feature2 associated with a given (feature1) in the last 30 days is needed. The predetermined data or a rolling distinct for feature2 will be maintained in (feature1) set. A summation of this rolling distinct for the last 30 time blocks will result in a distinct count of feature2 associated with feature1. For example, feature1 can be user and feature2 can be merchants, and the example will be used to provide a distinct count of merchants associated with a user in the last 30 days.

[0072] Aggregate from both the sets will be used to identify if rolling distinct needs to be updated and the day for which it needs to be decremented. On receiving a new transaction or a transaction request, firstly it will update data points in (feature1, feature2) set including count, amount, and so on. Increment operations are used to update all data points configured for one set in an aerospike call. Aerospike returns the new incremented value in the increment operation call. This output of new count in (feature1, feature2) will be used to identify if the rolling distinct needs to be updated for this transaction.

[0073] The output of increment operation for count suggests the number of transactions where feature1 is associated with feature2. The output 1 suggests this is the first combination of feature1 and feature2 seen for the time block and, thus the rolling distinct, representing distinct feature2, needs to be updated. If the count is anything but 1, it suggests the association of feature1 and feature2 has already been seen for the time block and hence the distinct count need not be updated now.

[0074] Secondly, the set (feature1) will be updated for count, amount, rolling distinct, and others. If the count is 1 in the above set, the rolling distinct will be incremented for the transaction time blocks. It will query the (feature1, feature2) record and derive the prior to last time block. Operation to decrement rolling distinct for this prior time block will also be added in the same aerospike update call, along with other operations.

[0075] The updated count value is used from increment operation to handle race condition. The one thread, updating the record, for the first time for a new day, will perform the extra aerospike call and update rolling distinct.

[0076] One transaction type can consist of multiple aggregates with different distinct count requirements, and the aggregate (feature1, feature2) needs to be updated before (feature1). The topological sorting of all aggregates will be used to derive the sequence of aggregates, in which they need to be updated.

[0077] In Fig 5a, one of the predetermined data is user and the other is a merchant. That is, feature 1 is user and feature 2 is merchant and is represented (user1, mex). On 2021-03-01, user 1 visited mex1 and mex 2. The count is 4 since the user visited merchants 4 times. However, the only merchant that user 1 visited for the last time is mex 1. As such, the rolling distinct on 2021-03-01 is 1. On 2021-03-02, user 1 visited mex2, mex 3 and mex 4. A query is done, from a database, a predetermined data (or unique merchants that the user has visited). The predetermined data is one that is determined from historical transactions conducted over a period of time by the user. It is then determined if data from the transaction request (or merchant data) corresponds to the predetermined data (or unique merchant that the user has visited). The only merchant that user 1 visited for the last time (or count = 1) is mex 2 and mex 3. As such, the rolling distinct on 2021-03-01 is 2. On 2021-03-04, user 1 only visited mex4 and it is not the last time user 1 visited mex 4 hence the rolling distinct is 0. On 2021-03-10, user 1 only visited mex4 and it is the last time user 1 visited mex 4 hence the rolling distinct is 1.

[0078] In Fig 5B, it shows how the rolling distinct is updated. As mentioned in the example of Fig 5A, On 2021-03-02, user 1 visited mex2, mex 3 and mex 4. The only merchant that user 1 visited for the last time (or= 1) is mex 2 and mex 3. As such, the rolling distinct on 2021-03-02 is 2. In Fig 5B, on 2021-03-10, user 1 went to mex 3 again, additionally to mex 4. As such, the only merchant that user 1 visited for the last time on 2021-03-02 is now only mex 2. That is, the count of a rolling distinct (or distinct count) on 2021-03-02 is decreased. And since this is the first visit to mex3 (count = 1 for user1.mex3) on 2021-03-10, rolling distinct is increased for 2021-03-10.

[0079] Fig. 6 shows a topographical view of features that are used to identify the dependency ordering of aggregates of a transaction and the sequence to update all aggregates.

[0080] The features that are predetermined to track a pattern of a transaction may be user, merchant), (user, card, device), (user, card), (user), (merchant, device), (merchant). Here (datapoint1, datapoint2) represents transactions using both the two datapoint alues, i.e. (user, card) represents a pattern of transactions done by a user using a particular card. (user,card) as shown in 606 may use count from (user,card,device) as shown in 604, to decide if rolling distinct (representing distinct count of device used by the user using the card) needs to be updated and for which day needs to be decremented. (user) as shown in 608 can refer to count from (user, card, device) as shown in 604 to calculate rolling distinct representing distinct number of combinations of card and device used by the user. It can also use count from (user, card) as shown in 606 to calculate rolling distinct representing distinct count of cards used by the user. It can use (user, merchant) as shown in 602 to maintain rolling distinct representing distinct count of merchants visited by the user. (merchant) as shown in 612 can use count of transactions from (merchant, device) as shown in 610 to update rolling distinct representing distinct count of devices used by the merchant. It can be used for the count of distinct combinations of multiple features as well. If the record from (feature1, feature2, feature3, feature4) will be used to update rolling_distinct in (feature1, feature2), the sum of rolling_distinct will represent the distinct count of (feature3, feature4) combinations.

Structural Context of the transaction processing server 110

[081] Figs. 7 and 8 depict the transaction processing server 110, upon which the transaction management system described above can be practiced.

[082] As seen in Fig. 7, the transaction processing server 110 includes: a computer module 201; input devices such as a keyboard 202, a mouse pointer device 203, a scanner 226, a camera 227, and a microphone 280; and output devices including a printer 215, a display device 214 and loudspeakers 217. An external Modulator-Demodulator (Modem) transceiver device 216 may be used by the computer module 201 for communicating to and from a communications network 220 via a connection 221. The communications network 220 may be a wide-area network (WAN), such as the Internet, a cellular telecommunications network, or a private WAN. Where the connection 221 is a telephone line, the modem 216 may be a traditional “dial-up” modem. Alternatively, where the connection 221 is a high capacity (e.g., cable) connection, the modem 216 may be a broadband modem. A wireless modem may also be used for wireless connection to the communications network 220.

[083] The input and output devices may be used by an operator who is interacting with the transaction processing server 110. For example, the printer 215 may be used to print reports relating to the status of the transaction processing server 110.

[084] The transaction processing server 110 uses the communications network 220 to communicate with the merchant devices 120 and the user devices 130 to receive commands and data. The transaction processing server 110 also uses the communications network 220 to communicate with the merchant devices 120 and the user devices 130 to send notification messages or broadcast transaction records.

[085] The computer module 201 typically includes at least one processor unit 205, and at least one memory unit 206. For example, the memory unit 206 may have semiconductor random access memory (RAM) and semiconductor read only memory (ROM). The computer module 201 also includes a number of input/output (I/O) interfaces including: an audio-video interface 207 that couples to the video display 214, loudspeakers 217 and microphone 280; an I/O interface 213 that couples to the keyboard 202, mouse 203, scanner 226, camera 227 and optionally a joystick or other human interface device (not illustrated); and an interface 208 for the external modem 216 and printer 215. In some implementations, the modem 216 may be incorporated within the computer module 201, for example within the interface 208. The computer module 201 also has a local network interface 211, which permits coupling of the computer system 110 via a connection 223 to a local-area communications network 222, known as a Local Area Network (LAN). As illustrated in Fig. 7, the local communications network 222 may also couple to the wide network 220 via a connection 224, which would typically include a so-called “firewall” device or device of similar functionality. The local network interface 211 may comprise an Ethernet circuit card, a Bluetooth® wireless arrangement or an IEEE 802.11 wireless arrangement; however, numerous other types of interfaces may be practiced for the interface 211.

[086] The I/O interfaces 208 and 213 may afford either or both of serial and parallel connectivity, the former typically being implemented according to the Universal Serial Bus (USB) standards and having corresponding USB connectors (not illustrated). Storage devices 209 are provided and typically include a hard disk drive (HDD) 210. Other storage devices such as a floppy disk drive and a magnetic tape drive (not illustrated) may also be used. An optical disk drive 212 is typically provided to act as a non-volatile source of data. Portable

memory devices, such optical disks (e.g., CD-ROM, DVD, Blu-ray Disc™), USB-RAM, portable, external hard drives, and floppy disks, for example, may be used as appropriate sources of data to the transaction processing server 110.

[087] The components 205 to 213 of the computer module 201 typically communicate via an interconnected bus 204 and in a manner that results in a conventional mode of operation of a computer system known to those in the relevant art. For example, the processor 205 is coupled to the system bus 204 using a connection 218. Likewise, the memory 206 and optical disk drive 212 are coupled to the system bus 204 by connections 219. Examples of computers on which the described arrangements can be practised include IBM-PC's and compatibles, Sun Sparcstations, Apple Mac™ or like computer systems.

[088] The methods of operating the transaction processing server 110, as shown in the processes of Figs. 2, 3, 4, and 6 to be described, may be implemented as one or more software application programs 233 executable within the transaction processing server 110. In particular, the steps of the methods shown in Figs. 2, 3, 4, and 6 are effected by instructions 231 (see Fig. 8) in the software (e.g., computer program codes) 233 that are carried out within the transaction processing server 110. The software instructions 231 may be formed as one or more code modules, each for performing one or more particular tasks. The software may also be divided into two separate parts, in which a first part and the corresponding code modules performs the operation of the transaction processing server 110 and a second part and the corresponding code modules manages the API and corresponding user interfaces in the merchant devices 120, the user devices 130, and on the display 214. In other words, the second part of the software manages the interaction between (a) the first part and (b) any one of the merchant devices 120, the user devices 130, and the operator of the server 110.

[089] The software may be stored in a computer readable medium, including the storage devices described below, for example. The software is loaded into the transaction processing server 110 from the computer readable medium, and then executed by transaction processing server 110. A computer readable medium having such software or computer program recorded on the computer readable medium is a computer program product. The use of the computer program product in the transaction processing server 110 preferably effects an advantageous apparatus for processing transaction requests for functioning as a transaction management system.

[090] The software (e.g., computer program codes) 233 is typically stored in the HDD 210 or the memory 206. The software 233 is loaded into the transaction processing server 110 from a computer readable medium (e.g., the memory 206), and executed by the processor 205. Thus, for example, the software 233 may be stored on an optically readable disk storage medium (e.g., CD-ROM) 225 that is read by the optical disk drive 212. A computer readable medium having such software or computer program recorded on it is a computer program product. The use of the computer program product in the server 110 preferably effects an apparatus for processing transaction requests for functioning as a transaction management system.

[091] In some instances, the application programs 233 may be supplied to the user encoded on one or more CD-ROMs 225 and read via the corresponding drive 212, or alternatively may be read by the user from the networks 220 or 222. Still further, the software can also be loaded into the transaction processing server 110 from other computer readable media. Computer readable storage media refers to any non-transitory tangible storage medium that provides recorded instructions and/or data to the transaction processing server 110 for execution and/or processing by the processor 205. Examples of such storage media include floppy disks, magnetic tape, CD-ROM, DVD, Blu-ray™ Disc, a hard disk drive, a ROM or integrated circuit, USB memory, a magneto-optical disk, or a computer readable card such as a PCMCIA card and the like, whether or not such devices are internal or external of the computer module 201. Examples of transitory or non-tangible computer readable transmission media that may also participate in the provision of software, application programs, instructions and/or data to the computer module 201 include radio or infra-red transmission channels as well as a network connection to another computer or networked device, and the Internet or Intranets including e-mail transmissions and information recorded on Websites and the like.

[092] The second part of the application programs 233 and the corresponding code modules mentioned above may be executed to implement one or more API of the transaction processing server 110 with associated graphical user interfaces (GUIs) to be rendered or otherwise represented upon the display 214 or the display of the merchant device 120 and the user device 130. Through manipulation of typically the keyboard 202 and the mouse 203, an operator of the server 110 and the application may manipulate the interface in a functionally adaptable manner to provide controlling commands and/or input to the applications associated with the GUI(s). Similarly, on the merchant devices 120 and the user devices 130, a user of those devices 120 and 130 manipulate the input devices (e.g., touch screen, keyboard, mouse, etc.) of those devices

120, 130 in a functionally adaptable manner to provide controlling commands and/or input to the applications associated with the GUI(s). Other forms of functionally adaptable user interfaces may also be implemented, such as an audio interface utilizing speech prompts output via the loudspeakers 217 and user voice commands input via the microphone 280. These other forms of functionally adaptable user interfaces may also be implemented on the merchant devices 120 and the user devices 130.

[093] Fig. 8 is a detailed schematic block diagram of the processor 205 and a "memory" 234. The memory 234 represents a logical aggregation of all the memory modules (including the HDD 209 and semiconductor memory 206) that can be accessed by the computer module 201 in Fig. 7.

[094] When the computer module 201 is initially powered up, a power-on self-test (POST) program 250 executes. The POST program 250 is typically stored in a ROM 249 of the semiconductor memory 206 of Fig. 7. A hardware device such as the ROM 249 storing software is sometimes referred to as firmware. The POST program 250 examines hardware within the computer module 201 to ensure proper functioning and typically checks the processor 205, the memory 234 (209, 206), and a basic input-output systems software (BIOS) module 251, also typically stored in the ROM 249, for correct operation. Once the POST program 250 has run successfully, the BIOS 251 activates the hard disk drive 210 of Fig. 7. Activation of the hard disk drive 210 causes a bootstrap loader program 252 that is resident on the hard disk drive 210 to execute via the processor 205. This loads an operating system 253 into the RAM memory 206, upon which the operating system 253 commences operation. The operating system 253 is a system level application, executable by the processor 205, to fulfil various high level functions, including processor management, memory management, device management, storage management, software application interface, and generic user interface.

[095] The operating system 253 manages the memory 234 (209, 206) to ensure that each process or application running on the computer module 201 has sufficient memory in which to execute without colliding with memory allocated to another process. Furthermore, the different types of memory available in the server 110 of Fig. 7 must be used properly so that each process can run effectively. Accordingly, the aggregated memory 234 is not intended to illustrate how particular segments of memory are allocated (unless otherwise stated), but rather to provide a general view of the memory accessible by the server 110 and how such is used.

[096] As shown in Fig. 8, the processor 205 includes a number of functional modules including a control unit 239, an arithmetic logic unit (ALU) 240, and a local or internal memory 248, sometimes called a cache memory. The cache memory 248 typically includes a number of storage registers 244 - 246 in a register section. One or more internal busses 241 functionally interconnect these functional modules. The processor 205 typically also has one or more interfaces 242 for communicating with external devices via the system bus 204, using a connection 218. The memory 234 is coupled to the bus 204 using a connection 219.

[097] The application program 233 includes a sequence of instructions 231 that may include conditional branch and loop instructions. The program 233 may also include data 232 which is used in execution of the program 233. The instructions 231 and the data 232 are stored in memory locations 228, 229, 230 and 235, 236, 237, respectively. Depending upon the relative size of the instructions 231 and the memory locations 228-230, a particular instruction may be stored in a single memory location as depicted by the instruction shown in the memory location 230. Alternately, an instruction may be segmented into a number of parts each of which is stored in a separate memory location, as depicted by the instruction segments shown in the memory locations 228 and 229.

[098] In general, the processor 205 is given a set of instructions which are executed therein. The processor 205 waits for a subsequent input, to which the processor 205 reacts to by executing another set of instructions. Each input may be provided from one or more of a number of sources, including data generated by one or more of the input devices 202, 203, data received from an external source across one of the networks 220, 202, data retrieved from one of the storage devices 206, 209 or data retrieved from a storage medium 225 inserted into the corresponding reader 212, all depicted in Fig. 7. The execution of a set of the instructions may in some cases result in output of data. Execution may also involve storing data or variables to the memory 234.

[099] The disclosed transaction management arrangements use input variables 254, which are stored in the memory 234 in corresponding memory locations 255, 256, 257. The transaction management arrangements produce output variables 261, which are stored in the memory 234 in corresponding memory locations 262, 263, 264. Intermediate variables 258 may be stored in memory locations 259, 260, 266 and 267.

[0100] Referring to the processor 205 of Fig. 8, the registers 244, 245, 246, the arithmetic logic unit (ALU) 240, and the control unit 239 work together to perform sequences of micro-operations needed to perform “fetch, decode, and execute” cycles for every instruction in the instruction set making up the program 233. Each fetch, decode, and execute cycle comprises: a fetch operation, which fetches or reads an instruction 231 from a memory location 228, 229, 230; a decode operation in which the control unit 239 determines which instruction has been fetched; and an execute operation in which the control unit 239 and/or the ALU 240 execute the instruction.

[0101] Thereafter, a further fetch, decode, and execute cycle for the next instruction may be executed. Similarly, a store cycle may be performed by which the control unit 239 stores or writes a value to a memory location 232.

[0102] Each step or sub-process in the processes of Figs. 2, 3, 4, and 6 is associated with one or more segments of the program 233 and is performed by the register section 244, 245, 247, the ALU 240, and the control unit 239 in the processor 205 working together to perform the fetch, decode, and execute cycles for every instruction in the instruction set for the noted segments of the program 233.

[0103] It is to be understood that the structural context of the transaction processing server 110 is presented merely by way of example. Therefore, in some arrangements, one or more features of the transaction processing server 110 may be omitted. Also, in some arrangements, one or more features of the transaction processing server 110 may be combined together. Additionally, in some arrangements, one or more features of the transaction processing server 110 may be split into one or more component parts.

[0104] Fig. 9 shows an alternative implementation of the transaction processing server 110. In the alternative implementation, the transaction processing server 110 may be generally described as a physical device comprising at least one processor 902 and at least one memory 904 including computer program code. The at least one memory 904 and the computer program code are configured to, with the at least one processor 902, cause the transaction processing server 110 to perform the operations described in Figs. 2, 3, 4, and 6. The transaction processing server 110 may also include a transaction processing module 906, payment monitoring module 908, a registered user module 910, a registered merchant module 912, and credit risk limit module 914.

The memory 904 stores computer program code that the processor 902 compiles to have each of the transaction processing module 906, the payment monitoring module 908, the registered user detail module 910, the registered merchant detail module 912, and the credit risk limit module 912 performs their respective functions. It will be appreciated that the processor 902 may also be configured to perform the functions performed by each of the transaction processing module 906, the payment monitoring module 908, the registered user detail module 910, the registered merchant detail module 912, and the credit risk limit module 912. In this arrangement, the transaction processing server 110 may only have a single processor 902 for performing the above-mentioned functions.

[0105] With reference to the discussion above regarding the user devices 130, the registered user module 910 manages the on-boarding (see the on-boarding discussion above) and storing of users who are consumers who wish to buy products from registered merchants. With reference to the discussion above regarding the merchant devices 120, the registered merchant module 912 manages the on-boarding (see the on-boarding discussion above) and storing of merchants which offer products for sale.

[0106] With reference to Fig. 1, the transaction processing module 906 receives a transaction request (see step 302 of Fig. 2 discussed above) from the user devices 130 or merchant devices 120 and determines if the user is authorized to proceed with the transaction based on a transaction data included in the transaction request and user information, the transaction data representing at least one of a product, and a category of the product, the user information representing a type of transaction the user is allowed to transact (see step 302 of Fig. 2 discussed above). The transaction processing module 906 proceeds with the transaction in response to the determination if the user is authorized to proceed with the transaction (see step 304 of Fig. 2 discussed above),

[0107] The transaction processing module 906 may also be configured to retrieve historical transaction data relating to the user (see step 402 of Fig. 3 as discussed above), and compare the retrieved historical transaction data and the transaction data to determine if the user is authorized to proceed with the transaction (see step 404 of Fig. 3 as discussed above).

[0108] With reference to Fig. 4, the credit risk limit module 914 may be configured to determine a credit risk level of the user, the credit risk level indicating an amount that the user is allowed to

transact (see step 502 of Fig. 4 discussed above). The transaction processing module 906 may be configured to determine if the user is authorized to proceed with the transaction based on a transaction amount included in the transaction request and the credit risk level of the user, the transaction amount representing an amount of the product (see method 500 of Fig. 4 above).

[0109] With reference to Fig. 6, the transaction processing module 906 may be configured to receive a payment request indicating a plurality of payments over the period of time, the payment request including a payment amount to be paid for each of the plurality of payments (see step 702 of Fig. 6 discussed above), and determine if the payment request can proceed (see step 704 of Fig. 6 shown above), The payment monitoring module 908 may be configured to monitor the plurality of payments over the period of time (see steps 710, 712, 714, 716 of Fig. 6 shown above).

[0110] The arrangements described are applicable to the computer and data processing industries and particularly for the payment technology.

[0111] The foregoing describes only some embodiments of the present invention, and modifications and/or changes can be made thereto without departing from the scope and spirit of the invention, the embodiments being illustrative and not restrictive.

CLAIMS:

1. A computer-implemented method for adaptively tracking a pattern of a transaction initiated by a transaction request, the transaction request including at least one data, the method comprising:

   querying, from a database, a predetermined data; the predetermined data is one that is determined from historical transactions conducted over a period of time by the user;

   determining if data from the transaction request corresponds to the predetermined data; and

   adaptively tracking a spending pattern of a user in response to an outcome of the determination of the data from transaction request.

2. The method according to claim 1, when it is determined that the data from the transaction request corresponds to the predetermined data, further comprising. decreasing a count of a distinct count of the predetermined data from the last historical transaction data.

3. The method according to claim 2, when it is determined that the data from the transaction request corresponds to the predetermined data, further comprising. increasing a count of the distinct count of the predetermined data from the transaction request.

4. The method according to any one of claims 2-3, further comprising:
   obtaining historical transaction data relating to historical transactions that were conducted over the period of time by the user; and

   calculating the distinct count of the predetermined data based on the obtained historical transaction data.

5. The method according to any one of the above claims, further comprising:
determining a type of transaction based on the transaction request.

6. The method according to claim 5, where the type of transaction indicates that a further predetermined data is to be used for determining if data from the transaction request corresponds to the predetermined data.

7. The method according to claim 6, further comprising:
ranking the predetermined data and the further predetermined data to determine an order of updating a respective distinct count of the predetermined data and the further predetermined data.

8. The method according to any one of the preceding claims, further comprising identifying at least one of a user information or a merchant information, wherein the step of determining if data from the transaction request corresponds to the predetermined data is performed based on the at least one of the user information or the merchant information.

9. A server for adaptively tracking a pattern of a transaction initiated by a transaction request, the server comprising: at least one processor; and at least one memory including computer program code; the at least one memory and the computer program code configured to, with the at least one processor, cause the server at least to:

query, from a database, a predetermined data; the predetermined data is one that is determined from historical transactions conducted over a period of time by the user;

determine if data from the transaction request corresponds to the predetermined data; and

adaptively tracking a pattern of a transaction in response to an outcome of the determination of the data from transaction request.

10. The server according to claim 9, wherein the at least one memory and the computer program code is further configured with the at least one processor to: decreasing a count of a distinct count of the predetermined data from the last historical transaction data when it is determined that the data from the transaction request corresponds to the predetermined data.

11. The server according to claim 10, wherein the at least one memory and the computer program code is further configured with the at least one processor to:

increasing a count of the distinct count of the predetermined data from the transaction request when it is determined that the data from the transaction request corresponds to the predetermined data.

12. The server according to any one of claims 10-11, wherein the at least one memory and the computer program code is further configured with the at least one processor to:

obtain historical transaction data relating to historical transactions that were conducted over the period of time by the user; and

calculate the distinct count of the predetermined data based on the obtained historical transaction data.

13. The server according to any one of claims 10-12, wherein the at least one memory and the computer program code is further configured with the at least one processor to determine a type of transaction based on the transaction request.

14. The server according to any one of claims 13, where the type of transaction indicates that a further predetermined data is to be used for determining if data from the transaction request corresponds to the predetermined data.

SYSTEM AND METHOD FOR ADAPTIVELY TRACKING A PATTERN OF A TRANSACTION

Abstract

The present disclosure relates to payment technology. In one aspect, the present disclosure provides a method of managing a transaction. The method comprises adaptively tracking a pattern of a transaction initiated by a transaction request, the transaction request including at least one data, the method comprising: querying, from a database, a predetermined data; the predetermined data is one that is determined from historical transactions conducted over a period of time by the user; determining if data from the transaction request corresponds to the predetermined data; and adaptively tracking a spending pattern of a user in response to an outcome of the determination of the data from transaction request.

Fig. 2

CLAIMS:

1. A computer-implemented method for adaptively tracking a pattern of a transaction initiated by a transaction request, the transaction request including at least one data, the method comprising:

   querying, from a database, predetermined data corresponding to one or more features and/or combinations of features for a plurality of time periods; wherein the predetermined data is determined from historical transactions conducted over a period of time by the user;

   determining that data from the transaction request corresponds to a feature or combination of said one or more features and/or combinations of features;

   updating distinct counts for the feature or combination by:

      decreasing a distinct count for the feature or combination for a time period corresponding to the last historical transaction data; and

      increasing a distinct count for the feature or combination for a time period corresponding to the transaction request;

   and

   adaptively tracking a spending pattern of a user in response to an outcome of the determination of the data from transaction request.

2. The method according claim 1, further comprising:
   obtaining historical transaction data relating to historical transactions that were conducted over the period of time by the user; and

calculating the distinct count of the predetermined data based on the obtained historical transaction data.

3. The method according to any one of the above claims, further comprising: determining a type of transaction based on the transaction request.

4. The method according to claim 3, where the type of transaction indicates that a further predetermined data is to be used for determining if data from the transaction request corresponds to the predetermined data.

5. The method according to claim 4, further comprising:
ranking the predetermined data and the further predetermined data to determine an order of updating a respective distinct count of the predetermined data and the further predetermined data.

6. The method according to any one of the preceding claims, further comprising identifying at least one of a user information or a merchant information, wherein the step of determining if data from the transaction request corresponds to the predetermined data is performed based on the at least one of the user information or the merchant information.

7. A server for adaptively tracking a pattern of a transaction initiated by a transaction request, the server comprising: at least one processor; and at least one memory including computer program code; the at least one memory and the computer program code configured to, with the at least one processor, cause the server at least to:

query, from a database, predetermined data corresponding to one or more features and/or combinations of features for a plurality of time periods; wherein the predetermined data is determined from historical transactions conducted over a period of time by the user;

determine that data from the transaction request corresponds to the a feature or combination of said one or more features and/or combinations of features;

updating distinct counts for the feature or combination by:

decreasing a distinct count for the feature or combination for a time period corresponding to the last historical transaction data; and

increasing a distinct count for the feature or combination for a time period corresponding to the transaction request;

and

adaptively tracking a pattern of a transaction in response to an outcome of the determination of the data from transaction request.

8. The server according to claim 7, wherein the at least one memory and the computer program code is further configured with the at least one processor to:

obtain historical transaction data relating to historical transactions that were conducted over the period of time by the user; and

calculate the distinct count of the predetermined data based on the obtained historical transaction data.

9. The server according to any one of claims 7-8, wherein the at least one memory and the computer program code is further configured with the at least one processor to determine a type of transaction based on the transaction request.

10. The server according to any one of claims 9, where the type of transaction indicates that a further predetermined data is to be used for determining if data from the transaction request corresponds to the predetermined data.