

[0001] This application for letters patent disclosure document describes inventive aspects that include various novel innovations (hereinafter “disclosure”) and contains material that is subject to copyright, mask work, and/or other intellectual property protection. The respective owners of such intellectual property have no objection to the facsimile reproduction of the disclosure by anyone as it appears in published Patent Office file/records, but otherwise reserve all rights.

[0002] This application is a non-provisional of and claims priority under 35 USC §§ 119, 120 to: United States provisional patent application serial no. 61/594,063 filed February 2, 2012, entitled “CENTRALIZED PERSONAL INFORMATION PLATFORM APPARATUSES, METHODS AND SYSTEMS,” attorney docket no. P-42185PRV|VISA-122/01US, and United States patent application serial no. 13/520,481 filed July 3, 2012, entitled “Universal Electronic Payment Apparatuses, Methods and Systems,” attorney docket no. P-42051US02|20270-136US.

[0003] The entire contents of the aforementioned application(s) are expressly incorporated by reference herein.

[0004] The present innovations generally address apparatuses, methods and systems for consumer data management and analytics, and more particularly, include MULTI-SOURCE, MULTI-DIMENSIONAL, CROSS-ENTITY, MULTIMEDIA DATABASE PLATFORM APPARATUSES, METHODS AND SYSTEMS (“MMCDB”).

[0005] However, in order to develop a reader's understanding of the innovations, disclosures have been compiled into a single description to illustrate and clarify how

1 aspects of these innovations operate independently, interoperate as between individual
 2 innovations, and/or cooperate collectively. The application goes on to further describe
 3 the interrelations and synergies as between the various innovations; all of which is to
 4 further compliance with 35 U.S.C. §112.

5 BACKGROUND

6 **[0006]** Many people share information through online social media streams using
 7 various communication devices or applications. Moreover, consumers may access
 8 online stores using the Internet on a computer or mobile device to make purchases from
 9 various service providers or merchants.

10 BRIEF DESCRIPTION OF THE DRAWINGS

11 **[0007]** The accompanying appendices and/or drawings illustrate various non-
 12 limiting, example, innovative aspects in accordance with the present descriptions:

13 **[0008]** FIGURE 1 shows a block diagram illustrating example aspects of a
 14 centralized personal information platform in some embodiments of the MMCDB;

15 **[0009]** FIGURES 2A-F show block diagrams illustrating example aspects of data
 16 models within a centralized personal information platform in some embodiments of the
 17 MMCDB;

18 **[0010]** FIGURE 3 shows a block diagram illustrating example MMCDB
 19 component configurations in some embodiments of the MMCDB;

20 **[0011]** FIGURE 4 shows a data flow diagram illustrating an example search result
 21 aggregation procedure in some embodiments of the MMCDB;

22 **[0012]** FIGURE 5 shows a logic flow diagram illustrating example aspects of
 23 aggregating search results in some embodiments of the MMCDB, e.g., a Search Results
 24 Aggregation (“SRA”) component 500;

25 **[0013]** FIGURES 6A-D show data flow diagrams illustrating an example card-
 26 based transaction execution procedure in some embodiments of the MMCDB;

1 **[0014]** FIGURES 7A-E show logic flow diagrams illustrating example aspects of
 2 card-based transaction execution, resulting in generation of card-based transaction data
 3 and service usage data, in some embodiments of the MMCDB, e.g., a Card-Based
 4 Transaction Execution (“CTE”) component 700;

5 **[0015]** FIGURE 8 shows a data flow diagram illustrating an example procedure to
 6 aggregate card-based transaction data in some embodiments of the MMCDB;

7 **[0016]** FIGURE 9 shows a logic flow diagram illustrating example aspects of
 8 aggregating card-based transaction data in some embodiments of the MMCDB, e.g., a
 9 Transaction Data Aggregation (“TDA”) component 900;

10 **[0017]** FIGURE 10 shows a data flow diagram illustrating an example social data
 11 aggregation procedure in some embodiments of the MMCDB;

12 **[0018]** FIGURE 11 shows a logic flow diagram illustrating example aspects of
 13 aggregating social data in some embodiments of the MMCDB, e.g., a Social Data
 14 Aggregation (“SDA”) component 1100;

15 **[0019]** FIGURE 12 shows a data flow diagram illustrating an example procedure
 16 for enrollment in value-add services in some embodiments of the MMCDB;

17 **[0020]** FIGURE 13 shows a logic flow diagram illustrating example aspects of
 18 social network payment authentication enrollment in some embodiments of the
 19 MMCDB, e.g., a Value-Add Service Enrollment (“VASE”) component 1300;

20 **[0021]** FIGURES 14A-B show flow diagrams illustrating example aspects of
 21 normalizing aggregated search, enrolled, service usage, transaction and/or other
 22 aggregated data into a standardized data format in some embodiments of the MMCDB,
 23 e.g., a Aggregated Data Record Normalization (“ADRN”) component 1400;

24 **[0022]** FIGURE 15 shows a logic flow diagram illustrating example aspects of
 25 recognizing data fields in normalized aggregated data records in some embodiments of
 26 the MMCDB, e.g., a Data Field Recognition (“DFR”) component 1500;

27 **[0023]** FIGURE 16 shows a logic flow diagram illustrating example aspects of
 28 classifying entity types in some embodiments of the MMCDB, e.g., an Entity Type
 29 Classification (“ETC”) component 1600;

1 **[0024]** FIGURE 17 shows a logic flow diagram illustrating example aspects of
 2 identifying cross-entity correlation in some embodiments of the MMCDB, e.g., a Cross-
 3 Entity Correlation (“CEC”) component 1700;

4 **[0025]** FIGURE 18 shows a logic flow diagram illustrating example aspects of
 5 associating attributes to entities in some embodiments of the MMCDB, e.g., an Entity
 6 Attribute Association (“EAA”) component 1800;

7 **[0026]** FIGURE 19 shows a logic flow diagram illustrating example aspects of
 8 updating entity profile-graphs in some embodiments of the MMCDB, e.g., an Entity
 9 Profile-Graph Updating (“EPGU”) component 1900;

10 **[0027]** FIGURE 20 shows a logic flow diagram illustrating example aspects of
 11 generating search terms for profile-graph updating in some embodiments of the
 12 MMCDB, e.g., a Search Term Generation (“STG”) component 2000;

13 **[0028]** FIGURES 21A-E show user interface diagrams illustrating example
 14 features of user interfaces for an electronic virtual wallet in some embodiments of the
 15 MMCDB;

16 **[0029]** FIGURE 22 shows a block diagram illustrating example aspects of a
 17 merchant analytics platform in some embodiments of the MMCDB;

18 **[0030]** FIGURES 23A-B show data flow diagrams illustrating an example
 19 procedure to provide a user and/or merchant offers for products, services and/or the
 20 like, using user behavior patterns derived from card-based transaction data in some
 21 embodiments of the MMCDB;

22 **[0031]** FIGURE 24 shows a logic flow diagram illustrating example aspects of
 23 providing a user and/or merchant offers for products, services and/or the like, using
 24 user behavior patterns derived from card-based transaction data in some embodiments
 25 of the MMCDB, e.g., a Merchant Analytics (“MA”) component;

26 **[0032]** FIGURE 25 shows a logic flow diagram illustrating example aspects of
 27 generating a user behavior pattern analysis in some embodiments of the MMCDB, e.g., a
 28 User Behavioral Pattern Analytics (“UBPA”) component;

1 **[0033]** FIGURE 26 shows a logic flow diagram illustrating example aspects of
 2 identifying user behavioral patterns from aggregated card-based transaction data in
 3 some embodiments of the MMCDB, e.g., a User Pattern Identification (“UPI”)
 4 component;

5 **[0034]** FIGURES 27A-B show block diagrams illustrating example aspects of
 6 merchant analytics in a second set of embodiments of the MMCDB;

7 **[0035]** FIGURES 28A-C show data flow diagrams illustrating an example
 8 procedure for econometrical analysis of a proposed investment strategy based on card-
 9 based transaction data in some embodiments of the MMCDB;

10 **[0036]** FIGURE 29 shows a logic flow diagram illustrating example aspects of
 11 normalizing raw card-based transaction data into a standardized data format in some
 12 embodiments of the MMCDB, e.g., a Transaction Data Normalization (“TDN”)
 13 component;

14 **[0037]** FIGURE 30 shows a logic flow diagram illustrating example aspects of
 15 generating classification labels for card-based transactions in some embodiments of the
 16 MMCDB, e.g., a Card-Based Transaction Classification (“CTC”) component;

17 **[0038]** FIGURE 31 shows a logic flow diagram illustrating example aspects of
 18 filtering card-based transaction data for econometrical investment strategy analysis in
 19 some embodiments of the MMCDB, e.g., a Transaction Data Filtering (“TDF”)
 20 component;

21 **[0039]** FIGURE 32 shows a logic flow diagram illustrating example aspects of
 22 anonymizing consumer data from card-based transactions for econometrical investment
 23 strategy analysis in some embodiments of the MMCDB, e.g., a Consumer Data
 24 Anonymization (“CDA”) component;

25 **[0040]** FIGURES 33A-B show logic flow diagrams illustrating example aspects of
 26 econometrically analyzing a proposed investment strategy based on card-based
 27 transaction data in some embodiments of the MMCDB, e.g., an Econometrical Strategy
 28 Analysis (“ESA”) component;

1 **[0041]** FIGURE 34 shows a logic flow diagram illustrating example aspects of
 2 reporting business analytics derived from an econometrical analysis based on card-
 3 based transaction data in some embodiments of the MMCDB, e.g., a Business Analytics
 4 Reporting (“BAR”) component;

5 **[0042]** FIGURE 35 shows a logic flow diagram illustrating example aspects of
 6 sharing an analytical model generated using data acquired using the centralized
 7 personal information platform in some embodiments of the MMCDB, e.g., an Analytical
 8 Model Sharing (“AMS”) component;

9 **[0043]** FIGURE 36 shows a logic flow diagram illustrating example aspects of a
 10 metadata based interpretation engine of the MMCDB that generates standardized
 11 encryptmatics XML from structured data obtained from various sources via the
 12 centralized personal information platform, e.g., an Encryptmatics XML Converter
 13 (“EXC”) component;

14 **[0044]** FIGURE 37 shows a data flow diagram illustrating an example email data
 15 aggregation procedure, in one embodiment of the MMCDB;

16 **[0045]** FIGURE 38 shows a block diagram illustrating an example distributed
 17 linking node mesh, in one embodiment of the MMCDB;

18 **[0046]** FIGURES 39A-F show a block diagram illustrating an example distributed
 19 linking node mesh search, in one embodiment of the MMCDB;

20 **[0047]** FIGURES 40A-C show a block diagram illustrating an example distributed
 21 linking node mesh index creation, in one embodiment of the MMCDB;

22 **[0048]** FIGURE 41 shows a logic flow illustrating an example Encryptmatics XML
 23 Converter component, in one embodiment of the MMCDB;

24 **[0049]** FIGURE 42 shows a logic flow illustrating input language loading by an
 25 Encryptmatics XML Converter component, in one embodiment of the MMCDB;

26 **[0050]** FIGURES 43A-B show a logic flow illustrating input model conversion by
 27 an Encryptmatics XML Converter component, in one embodiment of the MMCDB;

1 **[0051]** FIGURE 44 shows a block diagram illustrating aspects of a tumblar data
 2 source manipulation / anonymization component, e.g., a TDS component, in one
 3 embodiment of the MMCDB;

4 **[0052]** FIGURE 45 shows a logic flow diagram illustrating an example tumblar
 5 data source manipulation / anonymization component, in one embodiment of the
 6 MMCDB;

7 **[0053]** FIGURE 46 shows an example data flow illustrating mesh aggregation and
 8 cluster querying, in one embodiment of a MMCDB;

9 **[0054]** FIGURE 47 shows an example logic flow illustrating cluster response
 10 analysis and transaction triggering, in one embodiment of a MMCDB;

11 **[0055]** FIGURE 48A-C illustrate an example MMCDB application embodiment,
 12 in one embodiment of the MMCDB; and

13 **[0056]** FIGURE 49 shows a block diagram illustrating embodiments of a MMCDB
 14 controller.

15 **[0057]** The leading number of each reference number within the drawings
 16 indicates the figure in which that reference number is introduced and/or detailed. As
 17 such, a detailed discussion of reference number 101 would be found and/or introduced
 18 in Figure 1. Reference number 201 is introduced in Figure 2, etc. The leading number of
 19 each reference number within the drawings indicates the figure in which that reference
 20 number is introduced and/or detailed. As such, a detailed discussion of reference
 21 number 101 would be found and/or introduced in Figure 1. Reference number 201 is
 22 introduced in Figure 2, etc.

DETAILED DESCRIPTION

MMCDB

[0058] The MULTI-SOURCE, MULTI-DIMENSIONAL, CROSS-ENTITY, MULTIMEDIA DATABASE PLATFORM APPARATUSES, METHODS AND SYSTEMS (hereinafter “MMCDB”) transform data aggregated from various computer resources, via MMCDB components, into updated entity profiles, social graphs and/or investment recommendations. The MMCDB components, in various embodiments, implement advantageous features as set forth below.

Centralized Personal Information Platform

[0059] FIGURE 1 shows a block diagram illustrating example aspects of a centralized personal information platform in some embodiments of the MMCDB. In various scenarios, originators 111 such as merchants 111b, consumers 111c (including, e.g., social networking sites), account issuers, acquirers 111a, and/or the like, desire to utilize information from payment network systems for enabling various features for consumers, and may provide input for the generation of a centralized personal information platform.

[0060] For all of the input types (e.g., consumer transactions 111b, social network interactions 111d (e.g., emails, reviews, text posts, photos, audio/video/multimedia, conversations, chats, etc.), financial institution activity 111a (e.g., acquirers, authorizations, denials, issuers, fraud detection, etc.), merchant activities 111b (e.g., offers, coupons, redemptions, etc.), and/or the like, the mesh server 105 may aggregate and store such inputs in consolidated database 104b.

[0061] The mesh server aggregation may be achieved by obtaining a feed of financial transactions (e.g., if the mesh server is also a pay network server), by obtaining complete feed access (e.g., firehose feeds), from social networks (e.g., Facebook, Twitter, etc.), using publically available data API's (e.g., Google search API), and/or the like.

1 **[0062]** In one embodiment, the feeds may be obtained via high-bandwidth
2 network connections. An example of the high-bandwidth network connections may
3 include multiple optical fiber connections to an Internet backplane such as the
4 multinational Equinix Exchange, New York International Internet eXchange (e.g.,
5 “NYIIX”), and/or the like.

6 **[0063]** The obtained feeds may be stored in fast storage array servers for
7 processing or access by other processing servers. Examples of the fast storage array
8 servers may include server blades such as those manufactured by Dell Computer (e.g.,
9 Dell model M820, M620, and/or the like), having multiple RAID fast SSD drives of type
10 SAS with memory cache of type L1, L2, L3, and/or the like. In another embodiment, the
11 feeds may be stored in a public cloud storage service (e.g., Amazon S3, and/or the like)
12 or private cloud (e.g., OpenStack Storage object and/or OpenStack Storage block storage
13 running on servers such as those described above).

14 **[0064]** In one embodiment, the fast storage servers may employ a distributed file
15 system that provides high-throughput access to stored data. Example file systems
16 suitable for this purpose may include the Hadoop Distributed File System (e.g.,
17 “HDFS”), Google BigTable, and/or the like. The file system may be implemented
18 substantially as a key/value store or, in other embodiments, as a structured file system
19 containing directories and files. In some embodiments, a hybrid key/value structured
20 file system may be used in order to utilize the capabilities of both a key/value store and a
21 structured file system. In one embodiment, the fast storage array servers may be
22 connected to one or mesh servers (e.g., 105) for feed processing.

23 **[0065]** In one embodiment, the mesh servers (e.g., 105) may be server blades such
24 as those described above. In another embodiment, the servers may be virtualized and
25 running on a private virtualization platform such as VMWare ESXi, Xen, OpenStack
26 Compute and/or the like. In still other embodiments, the servers may be virtualized
27 using a publically available cloud service such as Amazon EC2 (e.g., via an Amazon
28 Machine Image / “AMI”, and/or the like) or Rackspace (e.g., by providing a machine
29 image such as a VDI or OVA file suitable for creating a virtualized machine).

1 **[0066]** The mesh server may generate dictionary short code words for every type
 2 of input and associate with that short word with the input (e.g., a MD5 hash, etc. may
 3 generate a short word for every type of input, where the resulting short code is unique to
 4 each input). This short code to actual data input association, when aggregated, may
 5 form the basis of a mesh dictionary. An example of mesh dictionary entry substantially
 6 in the following form of XML is:

```

7   <dictionary_entry>
8       {id: "1h65323765gtyuf#uy76355",
9         type: email,
10        category: {cat1: "food", cat2: "dinner"},
11        from_addr: "john.doe@gmail.com",
12        to_addr: "jane.doe@gmail.com",
13        subject: "Korean BBQ this weekend?",
14        dictionary_keywords: "Korean, dinner, nyc",
15        content_hash: "7m865323476feeaniiji"}
16   </dictionary_entry>
```

17 **[0067]** Segmented portions, complete dictionaries, and/or updates thereto, may
 18 thus be sent *en masse* to mesh analytics clone servers; for example, such update may be
 19 done at off-network peak hours set to occur at dynamically and/or at set intervals. This
 20 allows the analytics servers to perform analytics operations, and it allows those analytics
 21 servers to operate on short codes even without the full underlying backend data being
 22 available. In so doing, dictionaries may be analysed using less space than the full
 23 underlying raw data would require. Additionally, dictionaries may be distributed
 24 between multiple servers. In one embodiment, the dictionaries are replicated across
 25 multiple servers and, periodically, synchronized. In one embodiment, any inconsistencies
 26 in distributed and/or separated dictionaries may be reconciled using demarcation
 27 protocol and/or controlled inconsistency reconciliation for replicated data (see D.
 28 Barbara H. Garcia-Molina, The case for controlled inconsistency in replicated data,"
 29 Proc. of the Workshop on Management of Replicated Data, Houston, TX, Nov. 1990; D.
 30 Barbara H. Garcia-Molina, The demarcation protocol a technique for maintaining
 31 arithmetic constraints in distributed database systems, CS-TR-320-91, Princeton
 32 University, April 1991; the contents of both which are herein expressly incorporated by

reference). In one embodiment, dictionaries may defer any analytic operations that require the backend data until when the caching of the dictionary is complete. It should be noted that throughout this disclosure, reference is made to “payment network server” or “pay network server.” It should be understood that such server may incorporate mesh servers, and it also contemplates that such mesh servers may include a network of mesh analytics clone servers, clustering node servers, clustering servers, and/or the like.

[0068] Features that entities may desire include application services 112 such as alerts 112a, offers 112c, money transfers 112n, fraud detection 112b, and/or the like. In some embodiments of the MMCDB, such originators may request data to enable application services from a common, secure, centralized information platform including a consolidated, cross-entity profile-graph database 101. For example, the originators may submit complex queries to the MMCDB in a structure format, such as the example below. In this example, the query includes a query to determine a location (e.g., of a user), determine the weather associated with the location, perform analyses on the weather data, and provide an exploded graphical view of the results of the analysis:

```

16  <int
17      Model_id  ="1"
18      environment_type="RT"
19      meta_data="./fModels/robotExample.meta"
20      tumblr_location="./fModels/robotExample.tumblr.location"
21      input_format="JSON"
22      pmmls="AUTONOMOUS_AGENTS.PMML"
23      Model_type ="AUTONOMOUS_AGENTS"
24  >
25  <vault  >
26
27  <door:LOCATION>
28      <lock name="DETERMINE LOCATION"
29          inkey="INPUT" inkeyname="lat"
30          inkey2="INPUT" inkeyname2="long"
31          function="ROUND"
32          fnct1-prec="-2"
33          function-1="JOIN"
34          fnct2-delim=":"

```

```

1      tumblar='LAT_LONG.key'
2      outkey="TEMP"  outkeyname="location"
3      type="STRING"
4  />
5  <lock name="DETERMINE WEATHER"
6      inkey="TEMP" inkeyname="location"
7      mesh='MESHRT.RECENTWEATHER'
8      mesh-query='HASH'
9      outkey="TEMP"  outkeyname="WEATHERDATA"
10     type="ARRAY"
11 />
12 <lock name="EXPLODE DATA"
13     inkey="TEMP" inkeyname="WEATHERDATA"
14     function="EXPLODE"
15     fnct-delim=":"
16     outkey="MODELDATA"  outkeystartindex=1
17 />
18 <lock name="USER SETTINGS"
19     inkey="INPUT" inkeyname="USERID"
20     mesh='MESHRT.AUTONOMOUSAGENT.SETTINGS'
21     mesh-query='HASH'
22     outkey="TEMP"  outkeyname="USERSETTINGS"
23     type="ARRAY"
24 />
25 <lock name="EXPLODE USER"
26     inkey="TEMP" inkeyname="USERSETTINGS"
27     function="EXPLODE"
28     fnct-delim=":"
29     outkey="USERDATA"  outkeystartindex=1
30 />
31 <lock name="RUN MODEL"
32     inkey="MODELDATA"
33     inkey1="USERDATA"
34     function="TREE"
35     fnc-pmml="AUTONOMOUS_AGENTS.PMML"
36     outkey="OUTPUT"  outkeyname="WEATHER"
37     type="NUMERIC"

```

```

1      />
2  </door>
3  </vault>

```

4 **[0069]** A non-limiting, example listing of data that the MMCDB may return based
5 on a query is provided below. In this example, a user may log into a website via a
6 computing device. The computing device may provide a IP address, and a timestamp to
7 the MMCDB. In response, the MMCDB may identify a profile of the user from its
8 database, and based on the profile, return potential merchants for offers or coupons:

```

9  -----
10 ----- Use Case 3 -----
11 -- User log into a website
12 -- Only IP address, GMT and day of week is passed to Mesh
13 -- Mesh matches profile based on Affinity Group
14 -- Mesh returns potential Merchants for offers or coupons based
15 on temporary model using suppression rules
16 -----
17 -- Test case 1 IP:24:227:206 Hour:9 Day:3
18 -- Test case 2 IP:148:181:75 Hour:4 Day:5
19 -----
20 ----- AffinityGroup Lookup -----
21 -----
22 Look up test case 1
23 [OrderedDict([('ISACTIVE', 'True'), ('ENTITYKEY',
24 '24:227:206:3:1'), ('XML', None), ('AFFINITYGROUPNAME',
25 '24:227:206:3:1'), ('DESCRIPTION', None), ('TYPEOF', None),
26 ('UUID', '5f8df970b9ff11e09ab9270cf67eca90'))],
27 OrderedDict([('ISACTIVE', 'True'), ('BASEUUID',
28 '4fbea327b9ff11e094f433b5d7c45677'), ('TOKENENTITYKEY',
29 '4fbea327b9ff11e094f433b5d7c45677:TOKEN:349:F'), ('BASETYPE',
30 'MODEL_002_001_00'), ('STATUS', 'ACTIVE'), ('ISSUEDDATE', None),
31 ('WEIGHT', '349'), ('CATEGORY', 'F'), ('DOUBLELINKED', None),
32 ('UUID', '6b6aab39b9ff11e08d850dc270e3ea06'))],
33 OrderedDict([('ISACTIVE', 'True'), ('BASEUUID',
34 '4fbea328b9ff11e0a5f833b5d7c45677'), ('TOKENENTITYKEY',
35 '4fbea328b9ff11e0a5f833b5d7c45677:TOKEN:761:1'), ('BASETYPE',

```

```

1  'MODEL_003_001_00'), ('STATUS', 'ACTIVE'), ('ISSUEDDATE', None),
2  ('WEIGHT', '761'), ('CATEGORY', '1'), ('DOUBLELINKED', None),
3  ('UUID', '68aaca40b9ff11e0ac799fd4e415d9de'))],
4  OrderedDict([('ISACTIVE', 'True'), ('BASEUUID',
5  '4fbea328b9ff11e0a5f833b5d7c45677'), ('TOKENENTITYKEY',
6  '4fbea328b9ff11e0a5f833b5d7c45677:TOKEN:637:2'), ('BASETYPE',
7  'MODEL_003_001_00'), ('STATUS', 'ACTIVE'), ('ISSUEDDATE', None),
8  ('WEIGHT', '637'), ('CATEGORY', '2'), ('DOUBLELINKED', None),
9  ('UUID', '6b6d1c38b9ff11e08ce10dc270e3ea06'))],
10 OrderedDict([('ISACTIVE', 'True'), ('BASEUUID',
11 '4fbea328b9ff11e0a5f833b5d7c45677'), ('TOKENENTITYKEY',
12 '4fbea328b9ff11e0a5f833b5d7c45677:TOKEN:444:3'), ('BASETYPE',
13 'MODEL_003_001_00'), ('STATUS', 'ACTIVE'), ('ISSUEDDATE', None),
14 ('WEIGHT', '444'), ('CATEGORY', '3'), ('DOUBLELINKED', None),
15 ('UUID', '6342aa53b9ff11e0bcd9fd4e415d9de'))],
16 OrderedDict([('ISACTIVE', 'True'), ('BASEUUID',
17 '4fbea328b9ff11e0a5f833b5d7c45677'), ('TOKENENTITYKEY',
18 '4fbea328b9ff11e0a5f833b5d7c45677:TOKEN:333:4'), ('BASETYPE',
19 'MODEL_003_001_00'), ('STATUS', 'ACTIVE'), ('ISSUEDDATE', None),
20 ('WEIGHT', '333'), ('CATEGORY', '4'), ('DOUBLELINKED', None),
21 ('UUID', '62bd26a2b9ff11e0bc239fd4e415d9de'))],
22 OrderedDict([('ISACTIVE', 'True'), ('BASEUUID',
23 '4fbea328b9ff11e0a5f833b5d7c45677'), ('TOKENENTITYKEY',
24 '4fbea328b9ff11e0a5f833b5d7c45677:TOKEN:307:5'), ('BASETYPE',
25 'MODEL_003_001_00'), ('STATUS', 'ACTIVE'), ('ISSUEDDATE', None),
26 ('WEIGHT', '307'), ('CATEGORY', '5'), ('DOUBLELINKED', None),
27 ('UUID', '6b6d1c39b9ff11e0986c0dc270e3ea06'))],
28 OrderedDict([('ISACTIVE', 'True'), ('BASEUUID',
29 '4fbea32db9ff11e09f3e33b5d7c45677'), ('TOKENENTITYKEY',
30 '4fbea32db9ff11e09f3e33b5d7c45677:TOKEN:801:Spend'), ('BASETYPE',
31 'MODEL_008_001_00'), ('STATUS', 'ACTIVE'), ('ISSUEDDATE', None),
32 ('WEIGHT', '801'), ('CATEGORY', 'Spend'), ('DOUBLELINKED', None),
33 ('UUID', '6b6d1c3ab9ff11e0a4ec0dc270e3ea06'))],
34 OrderedDict([('ISACTIVE', 'True'), ('BASEUUID',
35 '4fbea32eb9ff11e0b55133b5d7c45677'), ('TOKENENTITYKEY',
36 '4fbea32eb9ff11e0b55133b5d7c45677:TOKEN:1:Volume'), ('BASETYPE',
37 'MODEL_009_001_00'), ('STATUS', 'ACTIVE'), ('ISSUEDDATE', None),

```

```

1  ('WEIGHT', '1'), ('CATEGORY', 'Volume'), ('DOUBLELINKED', None),
2  ('UUID', '62a09df3b9ff11e090d79fd4e415d9de'))]]
3  Found a direct match
4  148:181:75:1:2
5  -- Failed to find a direct match
6  -- Try again with only IP address and hour
7  [OrderedDict([('ISACTIVE', 'True'), ('ENTITYKEY',
8  '148:181:75:1:1'), ('XML', None), ('AFFINITYGROUPNAME',
9  '148:181:75:1:1'), ('DESCRIPTION', None), ('TYPEOF', None)])]
10 -- Found match for case 2
11 -----
12 ----- Temporary model rules-----
13 -----
14 {1: {'LOWER': 10, 'BASETYPE': ['MODEL_002_001_00',
15 'MODEL_003_001_00'], 'attribute': 'WEIGHT', 'rule': 'NEAR', 'OP':
16 'PROX', 'type': 'TOKENENTITY', 'HIGHER': 10}, 2: {'type':
17 ['MERCHANT'], 'rule': 'FOLLOW'}, 3: {'rule': 'RESTRICTSUBTYPE',
18 'BASETYPE': ['MODEL_002_001_00', 'MODEL_003_001_00']}}
19 -----
20 ----- Temporary Model Output-----
21 ----- For Use Case 1 -----
22 -----
23 -- Number of Nodes:102
24 _____LIVRARIASICILIAN
25 _____GDPCOLTD
26 _____GOODWILLINDUSTRIES
27 _____DISCOUNTDE
28 _____BARELANCHOE
29 _____BLOOMINGDALES
30 _____PARCWORLDTENNIS
31 _____STRIDERITEOUTLET
32 _____PARCCEANOR
33 _____PONTOFRIO
34 _____FNACPAULISTA
35 _____FINISHLINE
36 _____WALMARTCENTRAL
37 _____BESNIINTERLARGOS

```

1 _____PARCLOJASCOLOMBO
2 _____SHOPTIMEINTER
3 _____BEDBATHBEYOND
4 _____MACYSWEST
5 _____PARCRIACHUELOFILIAL
6 _____JCPENNEYCORPINC
7 _____PARCLOJASRENNERFL
8 _____PARCPAQUETAESPORTES
9 _____MARISALJ
10 _____PARCLEADERMAGAZINE
11 _____INTERFLORA
12 _____DECATHLON
13 _____PERNAMBUCANASFL
14 _____KARSTADTDE
15 _____PARCCEAMCO
16 _____CHAMPS
17 _____ACCESSORIZE
18 _____BLOOMINGDALESDVRS
19 _____PARCLIVRARIACULTURA
20 _____PARCCEALOJA
21 _____ARQUIBANCADA
22 _____KITBAG
23 _____FREDERICKSOFHLWD
24 _____WALMART
25 _____PARCLOJASINSINUANTE
26 _____WALMARTCONTAGEM
27 _____FOOTLOCKER
28 _____PARCSANTALOLLA
29 _____RICARDOELETRO
30 _____PARCPONTOFRIO
31 _____DOTPAYPLPOLSKA
32 _____CAMICADO
33 _____KARSTADT
34 _____PARCRAMSONS
35 _____PARCGREGORY
36 _____GREMIOFBPA
37 _____WALMARTSJC

1	_____PRODIRECTSOCCERLTD
2	_____LAVIEENROSE
3	_____PARCMARISALJ
4	_____ORDERS
5	_____PARCNSNNATALNORTE
6	_____LOJASINSINUANTE
7	_____B
8	_____CITYCOUNTY
9	_____WALMARTPACAEMBU
10	_____SOHO
11	_____WALMARTOSASCO
12	_____FOSSILSTORESIINC
13	_____MENARDSCLIO
14	_____PARCPEQUENTE
15	_____BEALLS
16	_____THEHOMEDEPOT
17	_____VIAMIA
18	_____PARCLOJASRIACHUELO
19	_____PARCLOJASMILANO
20	_____NORDSTROM
21	_____WAILANACOFFEEHOUSE
22	_____LANCHOEBELLA
23	_____PUKET
24	_____WALMARTSTORESINC
25	_____PARCPERNAMBUCANASFL
26	_____SMARTSHOPPER
27	_____PARCMAGAZINELUIZASP
28	_____COLUMBIASPORTSWEARCO
29	_____BARELANCESTADA
30	_____DONATEEBAY
31	_____PARCRICARDOELETRO
32	_____PARCDISANTINNI
33	_____SCHUHCOUK
34	_____CEANOR
35	_____PARCCAMICADO
36	_____PARCCENTAUROCE
37	_____PARCMARLUIJOIAS

```

1      _____ALBADAH
2      _____MARTINEZ
3      _____MONEYBOOKERSLTD
4      _____MACYS
5      _____PARCRIOCENTER
6      _____PARCCASASBAHIA
7      _____PARCSUBMARINOLOJA
8      _____INC
9      _____SUBMARINOLOJA
10     _____LOJASRENNERFL
11     _____RIACHUELOFILIAL
12     _____PARCSOHNODOSPES
13     _____PINKBIJU
14     _____PARCCEAMRB
15     -----
16     ----- Temporary model Output -----
17     ----- For Use Case 2 -----
18     -----
19     -- Number of Nodes:3
20     _____KITBAG
21     _____COLUMBIASPORTSWEARCO
22     _____GREMIOFBPA
23     -----
24     ----- End of Example Use Case -----
25     -----

```

26 **[0070]** In some embodiments, the MMCDB may provide access to information on
 27 a need-to-know basis to ensure the security of data of entities on which the MMCDB
 28 stores information. Thus, in some embodiments, access to information from the
 29 centralized platform may be restricted based on the originator as well as application
 30 services for which the data is requested. In some embodiments, the MMCDB may thus
 31 allow a variety of flexible application services to be built on a common database
 32 infrastructure, while preserving the integrity, security, and accuracy of entity data. In
 33 some implementations, the MMCDB may generate, update, maintain, store and/or
 34 provide profile information on entities, as well as a social graph that maintains and

1 updates interrelationships between each of the entities stored within the MMCDB. For
 2 example, the MMCDB may store profile information on an issuer bank 102a (see profile
 3 103a), a acquirer bank 102b (see profile 103b), a consumer 102c (see profile 103c), a
 4 user 102d (see profile 103d), a merchant 102e (see profile 103e), a second merchant 102f
 5 (see profile 103f). The MMCDB may also store relationships between such entities. For
 6 example, the MMCDB may store information on a relationship of the issuer bank 102a
 7 to the consumer 102c shopping at merchant 102e, who in turn may be related to user
 8 102d, who might bank at the bank 102b that serves as acquirer for merchant 102f.

9 **[0071]** FIGURES 2A-F show block diagrams illustrating example aspects of data
 10 models within a centralized personal information platform in some embodiments of the
 11 MMCDB. In various embodiments, the MMCDB may store a variety of attributes of
 12 entities according to various data models. A few non-limiting example data models are
 13 provided below. In some embodiments, the MMCDB may store user profile attributes.
 14 For example, a user profile model may store user identifying information 201, user
 15 aliases 202, email addresses 203, phone numbers 204, addresses 205, email address
 16 types 206, address types 207, user alias types 208, notification statuses 209, ISO
 17 country 210, phone number types 211, contract information with the MMCDB 212, user
 18 authorization status 213, user profile status 214, security answer 215, security questions
 19 216, language 217, time zone 218, and/or the like, each of the above field types including
 20 one or more fields and field values. As another example, a user financial attributes
 21 model may store user identifying information 220, user financial account information
 22 221, account contract information 222, user financial account role 223, financial account
 23 type 224, financial account identifying information 225, contract information 226,
 24 financial account validation 227, financial account validation type 228, and/or the like.
 25 As another example, a user payment card attributes data model may include field types
 26 such as, but not limited to: user identifying information 230, user financial account
 27 information 231, user financial account role 232, account consumer applications 233,
 28 user consumer application 234, financial account type 235, financial account validation
 29 type 236, financial account information 237, consumer application information 238,
 30 consumer application provider information 239, and/or the like. As another example, a
 31 user services attributes data model may include field types such as, but not limited to:

1 user identifying information 240, user alias 241, consumer application user alias status
 2 242, user alias status 243, status change reason code 244, user contract 245, contract
 3 information 246, user service attribute value 247, consumer application attributes 248,
 4 account service attribute value, account contract 250, user profile status 261, contract
 5 business role 252, contract business 253, client information 254, contract role 255,
 6 consumer application 256, user activity audit 257, login results 258, and/or the like. As
 7 another example, a user services usage attributes data model may include field types
 8 such as, but not limited to: user identifying information 260, user alias 261, consumer
 9 application user alias status 262, status change reason code 263, user alias status 264,
 10 user consumer application 265, user login audit 266, login result 267, account service
 11 attribute value 268, account consumer application 269, consumer application 270,
 12 consumer application provider 271, login result 272, and/or the like. As another
 13 example, a user graph attributes data model may include field types such as, but not
 14 limited to: user identifying information 280, user contact 281, consumer application
 15 user alias status 282, relationship 283, and/or the like. In some embodiments, the
 16 MMCDB may store each object (e.g., user, merchant, issuer, acquirer, IP address,
 17 household, etc.) as a node in graph database, and store data with respect to each node in
 18 a format such as the example format provided below:

```

19  <Nodes Data>
20  ID, Nodes, Label
21  2fdc7e3fbd1c11e0be645528b00e8d0e, 2fdc7e3fbd1c11e0be645528b00e8d0e
22  , AFFINITYGROUPNAME: 49:95:0:3:1
23  32b1d53ebd1c11e094172557fb829fdf, 32b1d53ebd1c11e094172557fb829fdf
24  , TOKENENTITYKEY: 2b8494f0bd1c11e09c856d888c43f7c2: TOKEN: 0: F
25  2e6381e4bd1c11e0b9ffc929a54bb0fd, 2e6381e4bd1c11e0b9ffc929a54bb0fd
26  , MERCHANTNAME: _____MERCHANT_ABC
27  2fdc7e3dbd1c11e0a22d5528b00e8d0e, 2fdc7e3dbd1c11e0a22d5528b00e8d0e
28  , AFFINITYGROUPNAME: 49:95:0:1:1
29  2e6381e7bd1c11e091b7c929a54bb0fd, 2e6381e7bd1c11e091b7c929a54bb0fd
30  , MERCHANTNAME: _____MERCHANT_XYZ
31  2cf8cbabbd1c11e0894a5de4f9281135, 2cf8cbabbd1c11e0894a5de4f9281135
32  , USERNAME: 000060FF6557F103
  
```

```
1 2e6381debd1c11e0b336c929a54bb0fd,2e6381debd1c11e0b336c929a54bb0fd
2 ,MERCHANTNAME:_____MERCHANT_123
3 2e6381e0bd1c11e0b4e8c929a54bb0fd,2e6381e0bd1c11e0b4e8c929a54bb0fd
4 ,MERCHANTNAME:_____MERCHANT_FGH
5 2cf681c1bd1c11e0b8815de4f9281135,2cf681c1bd1c11e0b8815de4f9281135
6 ,USERNAME:000030C57080FFE8
7 2b8494f1bd1c11e0acbd6d888c43f7c2,2b8494f1bd1c11e0acbd6d888c43f7c2
8 ,MODELNAME:MODEL_003_001_00
9 32b44638bd1c11e0b01c2557fb829fdf,32b44638bd1c11e0b01c2557fb829fdf
10 ,TOKENENTITYKEY:2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:1
11 2fdc7e40bd1c11e094675528b00e8d0e,2fdc7e40bd1c11e094675528b00e8d0e
12 ,AFFINITYGROUPNAME:49:95:0:4:1
13 2b8494f0bd1c11e09c856d888c43f7c2,2b8494f0bd1c11e09c856d888c43f7c2
14 ,MODELNAME:MODEL_002_001_00
15 32b44639bd1c11e0b15b2557fb829fdf,32b44639bd1c11e0b15b2557fb829fdf
16 ,TOKENENTITYKEY:2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:2
17 32ce84febd1c11e0b0112557fb829fdf,32ce84febd1c11e0b0112557fb829fdf
18 ,TOKENENTITYKEY:2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:4
19 2e6381e3bd1c11e095b1c929a54bb0fd,2e6381e3bd1c11e095b1c929a54bb0fd
20 ,MERCHANTNAME:_____MERCHANT_789
21 34582a87bd1c11e080820167449bc60f,34582a87bd1c11e080820167449bc60f
22 ,TOKENENTITYKEY:2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:778:5
23 2e6381e5bd1c11e0b62cc929a54bb0fd,2e6381e5bd1c11e0b62cc929a54bb0fd
24 ,MERCHANTNAME:_____MERCHANT_456
25 2fdc7e3ebd1c11e088b55528b00e8d0e,2fdc7e3ebd1c11e088b55528b00e8d0e
26 ,AFFINITYGROUPNAME:49:95:0:2:1
27 32c4e80dbd1c11e09e442557fb829fdf,32c4e80dbd1c11e09e442557fb829fdf
28 ,TOKENENTITYKEY:2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:774:5
29 2e6381e1bd1c11e0bf28c929a54bb0fd,2e6381e1bd1c11e0bf28c929a54bb0fd
30 ,MERCHANTNAME:_____MERCHANT_WER
31 2cf681b8bd1c11e08be85de4f9281135,2cf681b8bd1c11e08be85de4f9281135
32 ,USERNAME:00002552FC930FF8
33 2cf8cba8bd1c11e09fbc5de4f9281135,2cf8cba8bd1c11e09fbc5de4f9281135
34 ,USERNAME:0000570FF1B46A24
35 32b4463abd1c11e0bdaa2557fb829fdf,32b4463abd1c11e0bdaa2557fb829fdf
36 ,TOKENENTITYKEY:2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:3
```

```

1  2cf8cbaebd1c11e0b6515de4f9281135,2cf8cbaebd1c11e0b6515de4f9281135
2  ,USERNAME:000064A20FF962D4
3  2e6381e6bd1c11e08087c929a54bb0fd,2e6381e6bd1c11e08087c929a54bb0fd
4  ,MERCHANTNAME:_____MERCHANT_496
5  2e6381e2bd1c11e0941dc929a54bb0fd,2e6381e2bd1c11e0941dc929a54bb0fd
6  ,MERCHANTNAME:_____MERCHANT_SDF
7  <Edge Data>Source,Target,Type,label, Weight
8  32ce84febd1c11e0b0112557fb829fdf,2e6381e6bd1c11e08087c929a54bb0fd
9  ,MODEL_003_001_00,2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:4,1
10 000
11 2fdc7e3ebd1c11e088b55528b00e8d0e,32ce84febd1c11e0b0112557fb829fdf
12 ,MODEL_003_001_00,2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:4,1
13 000
14 2e6381e2bd1c11e0941dc929a54bb0fd,34582a87bd1c11e080820167449bc60f
15 ,MODEL_003_001_00,2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:778:5,77
16 8
17 2b8494f1bd1c11e0acbd6d888c43f7c2,34582a87bd1c11e080820167449bc60f
18 ,MODEL_003_001_00,2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:778:5,77
19 8
20 2e6381e1bd1c11e0bf28c929a54bb0fd,32b44639bd1c11e0b15b2557fb829fdf
21 ,MODEL_003_001_00,2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:2,0
22 2e6381e0bd1c11e0b4e8c929a54bb0fd,32ce84febd1c11e0b0112557fb829fdf
23 ,MODEL_003_001_00,2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:4,1
24 000
25 32b44639bd1c11e0b15b2557fb829fdf,2e6381e6bd1c11e08087c929a54bb0fd
26 ,MODEL_003_001_00,2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:2,0
27 2e6381e1bd1c11e0bf28c929a54bb0fd,32ce84febd1c11e0b0112557fb829fdf
28 ,MODEL_003_001_00,2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:4,1
29 000
30 2e6381debd1c11e0b336c929a54bb0fd,32ce84febd1c11e0b0112557fb829fdf
31 ,MODEL_003_001_00,2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:4,1
32 000
33 2e6381e3bd1c11e095b1c929a54bb0fd,34582a87bd1c11e080820167449bc60f
34 ,MODEL_003_001_00,2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:778:5,77
35 8
36 2fdc7e40bd1c11e094675528b00e8d0e,32b44639bd1c11e0b15b2557fb829fdf
37 ,MODEL_003_001_00,2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:2,0

```

1 2b8494f1bd1c11e0acbd6d888c43f7c2, 32b4463abd1c11e0bdaa2557fb829fdf
2 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:3,0
3 2e6381e3bd1c11e095b1c929a54bb0fd, 32b4463abd1c11e0bdaa2557fb829fdf
4 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:3,0
5 2e6381e3bd1c11e095b1c929a54bb0fd, 32b1d53ebd1c11e094172557fb829fdf
6 ,MODEL_002_001_00, 2b8494f0bd1c11e09c856d888c43f7c2:TOKEN:0:F,0
7 2e6381e5bd1c11e0b62cc929a54bb0fd, 34582a87bd1c11e080820167449bc60f
8 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:778:5,77
9 8
10 2cf8cbabbd1c11e0894a5de4f9281135, 32b44638bd1c11e0b01c2557fb829fdf
11 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:1,1
12 000
13 2cf681b8bd1c11e08be85de4f9281135, 32b1d53ebd1c11e094172557fb829fdf
14 ,MODEL_002_001_00, 2b8494f0bd1c11e09c856d888c43f7c2:TOKEN:0:F,0
15 32b4463abd1c11e0bdaa2557fb829fdf, 2e6381e6bd1c11e08087c929a54bb0fd
16 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:3,0
17 2e6381debd1c11e0b336c929a54bb0fd, 32b44639bd1c11e0b15b2557fb829fdf
18 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:2,0
19 2e6381e1bd1c11e0bf28c929a54bb0fd, 32b44638bd1c11e0b01c2557fb829fdf
20 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:1,1
21 000
22 2e6381e5bd1c11e0b62cc929a54bb0fd, 32ce84febd1c11e0b0112557fb829fdf
23 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:4,1
24 000
25 2e6381e1bd1c11e0bf28c929a54bb0fd, 32b4463abd1c11e0bdaa2557fb829fdf
26 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:3,0
27 2e6381e2bd1c11e0941dc929a54bb0fd, 32b44639bd1c11e0b15b2557fb829fdf
28 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:2,0
29 2b8494f1bd1c11e0acbd6d888c43f7c2, 32c4e80dbd1c11e09e442557fb829fdf
30 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:774:5,77
31 4
32 2e6381e2bd1c11e0941dc929a54bb0fd, 32b1d53ebd1c11e094172557fb829fdf
33 ,MODEL_002_001_00, 2b8494f0bd1c11e09c856d888c43f7c2:TOKEN:0:F,0
34 2e6381e4bd1c11e0b9ffc929a54bb0fd, 32b4463abd1c11e0bdaa2557fb829fdf
35 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:3,0
36 2fdc7e3fbd1c11e0be645528b00e8d0e, 32b4463abd1c11e0bdaa2557fb829fdf
37 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:3,0

1 2e6381e1bd1c11e0bf28c929a54bb0fd, 32b1d53ebd1c11e094172557fb829fdf
2 ,MODEL_002_001_00, 2b8494f0bd1c11e09c856d888c43f7c2:TOKEN:0:F, 0
3 2fdc7e40bd1c11e094675528b00e8d0e, 32ce84febd1c11e0b0112557fb829fdf
4 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:4, 1
5 000
6 2cf8cba8bd1c11e09fbc5de4f9281135, 32c4e80dbd1c11e09e442557fb829fdf
7 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:774:5, 77
8 4
9 2e6381e2bd1c11e0941dc929a54bb0fd, 32b44638bd1c11e0b01c2557fb829fdf
10 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:1, 1
11 000
12 2e6381e4bd1c11e0b9ffc929a54bb0fd, 32b1d53ebd1c11e094172557fb829fdf
13 ,MODEL_002_001_00, 2b8494f0bd1c11e09c856d888c43f7c2:TOKEN:0:F, 0
14 2e6381e5bd1c11e0b62cc929a54bb0fd, 32b44639bd1c11e0b15b2557fb829fdf
15 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:2, 0
16 32b1d53ebd1c11e094172557fb829fdf, 2e6381e6bd1c11e08087c929a54bb0fd
17 ,MODEL_002_001_00, 2b8494f0bd1c11e09c856d888c43f7c2:TOKEN:0:F, 0
18 2b8494f1bd1c11e0acbd6d888c43f7c2, 32b44639bd1c11e0b15b2557fb829fdf
19 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:2, 0
20 2e6381e3bd1c11e095b1c929a54bb0fd, 32b44638bd1c11e0b01c2557fb829fdf
21 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:1, 1
22 000
23 2fdc7e3dbd1c11e0a22d5528b00e8d0e, 32ce84febd1c11e0b0112557fb829fdf
24 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:4, 1
25 000
26 2cf681c1bd1c11e0b8815de4f9281135, 32b44638bd1c11e0b01c2557fb829fdf
27 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:1, 1
28 000
29 2cf681c1bd1c11e0b8815de4f9281135, 32b1d53ebd1c11e094172557fb829fdf
30 ,MODEL_002_001_00, 2b8494f0bd1c11e09c856d888c43f7c2:TOKEN:0:F, 0
31 2e6381e3bd1c11e095b1c929a54bb0fd, 32b44639bd1c11e0b15b2557fb829fdf
32 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:2, 0
33 2fdc7e3fbd1c11e0be645528b00e8d0e, 32b1d53ebd1c11e094172557fb829fdf
34 ,MODEL_002_001_00, 2b8494f0bd1c11e09c856d888c43f7c2:TOKEN:0:F, 0
35 32b44638bd1c11e0b01c2557fb829fdf, 2e6381e6bd1c11e08087c929a54bb0fd
36 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:1, 1
37 000

1 2cf8cbaebd1c11e0b6515de4f9281135, 32ce84febd1c11e0b0112557fb829fdf
2 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:4, 1
3 000
4 2e6381e6bd1c11e08087c929a54bb0fd, 32b1d53ebd1c11e094172557fb829fdf
5 ,MODEL_002_001_00, 2b8494f0bd1c11e09c856d888c43f7c2:TOKEN:0:F, 0
6 2e6381e7bd1c11e091b7c929a54bb0fd, 34582a87bd1c11e080820167449bc60f
7 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:778:5, 77
8 8
9 2e6381e1bd1c11e0bf28c929a54bb0fd, 34582a87bd1c11e080820167449bc60f
10 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:778:5, 77
11 8
12 2e6381e5bd1c11e0b62cc929a54bb0fd, 32b1d53ebd1c11e094172557fb829fdf
13 ,MODEL_002_001_00, 2b8494f0bd1c11e09c856d888c43f7c2:TOKEN:0:F, 0
14 2b8494f0bd1c11e09c856d888c43f7c2, 32b1d53ebd1c11e094172557fb829fdf
15 ,MODEL_002_001_00, 2b8494f0bd1c11e09c856d888c43f7c2:TOKEN:0:F, 0
16 2b8494f1bd1c11e0acbd6d888c43f7c2, 32b44638bd1c11e0b01c2557fb829fdf
17 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:1, 1
18 000
19 2e6381e6bd1c11e08087c929a54bb0fd, 32b4463abd1c11e0bdaa2557fb829fdf
20 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:3, 0
21 2b8494f1bd1c11e0acbd6d888c43f7c2, 32ce84febd1c11e0b0112557fb829fdf
22 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:4, 1
23 000
24 2cf681c1bd1c11e0b8815de4f9281135, 32b44639bd1c11e0b15b2557fb829fdf
25 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:2, 0
26 2cf681c1bd1c11e0b8815de4f9281135, 32b4463abd1c11e0bdaa2557fb829fdf
27 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:3, 0
28 2e6381e2bd1c11e0941dc929a54bb0fd, 32b4463abd1c11e0bdaa2557fb829fdf
29 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:3, 0
30 2e6381e3bd1c11e095b1c929a54bb0fd, 32ce84febd1c11e0b0112557fb829fdf
31 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:4, 1
32 000
33 2e6381e6bd1c11e08087c929a54bb0fd, 32ce84febd1c11e0b0112557fb829fdf
34 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:4, 1
35 000

```
1 2e6381e6bd1c11e08087c929a54bb0fd, 34582a87bd1c11e080820167449bc60f
2 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:778:5, 77
3 8
4 2e6381e6bd1c11e08087c929a54bb0fd, 32b44638bd1c11e0b01c2557fb829fdf
5 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:1, 1
6 000
7 2fdc7e3ebd1c11e088b55528b00e8d0e, 32b44639bd1c11e0b15b2557fb829fdf
8 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:2, 0
9 2e6381e5bd1c11e0b62cc929a54bb0fd, 32b4463abd1c11e0bdaa2557fb829fdf
10 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:3, 0
11 2e6381e4bd1c11e0b9ffc929a54bb0fd, 34582a87bd1c11e080820167449bc60f
12 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:778:5, 77
13 8
14 2e6381e4bd1c11e0b9ffc929a54bb0fd, 32b44638bd1c11e0b01c2557fb829fdf
15 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:1, 1
16 000
17 34582a87bd1c11e080820167449bc60f, 2e6381e6bd1c11e08087c929a54bb0fd
18 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:778:5, 77
19 8
20 2e6381e6bd1c11e08087c929a54bb0fd, 32b44639bd1c11e0b15b2557fb829fdf
21 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:2, 0
22 2e6381e5bd1c11e0b62cc929a54bb0fd, 32b44638bd1c11e0b01c2557fb829fdf
23 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:1, 1
24 000
25 2fdc7e3fbd1c11e0be645528b00e8d0e, 32b44638bd1c11e0b01c2557fb829fdf
26 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:1, 1
27 000
28 2cf681b8bd1c11e08be85de4f9281135, 32b44639bd1c11e0b15b2557fb829fdf
29 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:2, 0
30 2e6381e4bd1c11e0b9ffc929a54bb0fd, 32b44639bd1c11e0b15b2557fb829fdf
31 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:2, 0
32 2cf681b8bd1c11e08be85de4f9281135, 32b4463abd1c11e0bdaa2557fb829fdf
33 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:3, 0
34 2e6381e4bd1c11e0b9ffc929a54bb0fd, 32ce84febd1c11e0b0112557fb829fdf
35 ,MODEL_003_001_00, 2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:4, 1
36 000
```

```

1  2e6381e2bd1c11e0941dc929a54bb0fd,32ce84febd1c11e0b0112557fb829fdf
2  ,MODEL_003_001_00,2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:4,1
3  000
4  2fdc7e3dbd1c11e0a22d5528b00e8d0e,32b44639bd1c11e0b15b2557fb829fdf
5  ,MODEL_003_001_00,2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:0:2,0
6  2cf681b8bd1c11e08be85de4f9281135,32b44638bd1c11e0b01c2557fb829fdf
7  ,MODEL_003_001_00,2b8494f1bd1c11e0acbd6d888c43f7c2:TOKEN:1000:1,1
8  000

```

9 **[0072]** In alternate examples, the MMADB may store data in a JavaScript Object
10 Notation (“JSON”) format. The stored information may include data regarding the
11 object, such as, but not limited to: commands, attributes, group information, payment
12 information, account information, etc., such as in the example below:

```

13  {'MERCHANT': {'TYPEOFTYPES': ['MERCHANTS', 'SYNTHETICNETWORKS'],
14  'FUNCTIONS': {'ENTITYCREATION': 'putNetwork'}
15  , 'UNIQUEATTRIBUTES': ['MERCHANTNAME'],
16  'TOKENENTITIESRELATIONSHIPS': [], 'ATTRIBUTES': {'MERCHANT': (2,
17  'STRING', 0, 'VALUE'), 'MERCH_ZIP_CD': (7, 'STRING', 0, 'VALUE'),
18  'MERCH_NAME': (8, 'STRING', 0, 'VALUE'), 'MERCHANTNAME': (3,
19  'STRING', 0, 'VALUE'), 'ACQ_CTRY_NUM': (4, 'STRING', 0, 'VALUE'),
20  'ACQ_PCR': (6, 'STRING', 0, 'VALUE'), 'ACQ_REGION_NUM': (5,
21  'STRING', 0, 'VALUE'), 'ISACTIVE': (0, 'BOOL', 1, 'VALUE'),
22  'ENTITYKEY': (1, 'STRING', 0, 'VALUE')}}
23  }
24  , 'AFFINITYGROUP': {'TYPEOFTYPES': ['AFFINITYGROUPS'],
25  'FUNCTIONS': {'ENTITYCREATION': 'putNetwork'}
26  , 'UNIQUEATTRIBUTES': ['AFFINITYGROUPNAME'],
27  'TOKENENTITIESRELATIONSHIPS': [], 'ATTRIBUTES': {'XML': (2,
28  'STRING', 0, 'VALUE'), 'DESCRIPTION': (4, 'STRING', 0, 'VALUE'),
29  'ENTITYKEY': (1, 'STRING', 0, 'VALUE'), 'TYPEOF': (5, 'STRING',
30  0, 'VALUE'), 'AFFINITYGROUPNAME': (3, 'STRING', 0, 'VALUE'),
31  'ISACTIVE': (0, 'BOOL', 1, 'VALUE')}}
32  }
33  , 'CASCADINGPAYMENT': {'TYPEOFTYPES': ['CASCADINGPAYMENT'],
34  'FUNCTIONS': {'ENTITYCREATION': 'putNetwork'}}

```

```

1  , 'UNIQUEATTIBUTES': ['CASCADINGPAYMENTNAME'],
2  'TOKENENTITIESRELATIONSHIPS': ['GROUP'], 'ATTRIBUTES': {'STATUS':
3  (2, 'STRING', 0, 'VALUE'), 'EXPDT': (6, 'DATETIME', 0, 'VALUE'),
4  'GROUP': (3, 'STRING', 0, 'VALUE'), 'RESTRICTIONS': (7, 'DICT',
5  0, 'VALUE'), 'CASCADINGPAYMENTNAME': (4, 'STRING', 0, 'VALUE'),
6  'STARTDT': (5, 'DATETIME', 0, 'VALUE'), 'ISACTIVE': (0, 'BOOL',
7  1, 'VALUE'), 'ENTITYKEY': (1, 'STRING', 0, 'VALUE')}
8  }
9  , 'GROUP': {'TYPEOFTYPES': [], 'FUNCTIONS': {'ENTITYCREATION':
10 'putNetwork'}}
11 , 'UNIQUEATTIBUTES': ['GROUPNAME'], 'TOKENENTITIESRELATIONSHIPS':
12 {}
13 , 'ATTRIBUTES': {'GROUPNAME': (2, 'STRING', 0, 'VALUE'),
14 'DESCRIPTION': (2, 'STRING', 0, 'VALUE'), 'ISACTIVE': (0, 'BOOL',
15 1, 'VALUE'), 'ENTITYKEY': (1, 'STRING', 0, 'VALUE')}
16 }
17 , 'USERS': {'TYPEOFTYPES': [], 'FUNCTIONS': {'ENTITYCREATION':
18 'putNetwork'}}
19 , 'UNIQUEATTIBUTES': ['USERSID'], 'TOKENENTITIESRELATIONSHIPS':
20 {}
21 , 'ATTRIBUTES': {'USERSID': (2, 'STRING', 0, 'VALUE'),
22 'ISACTIVE': (0, 'BOOL', 1, 'VALUE'), 'ENTITYKEY': (1, 'STRING',
23 0, 'VALUE')}
24 }
25 , 'TWITTERUSER': {'TYPEOFTYPES': ['TOKENENTITY'], 'FUNCTIONS':
26 {'ENTITYCREATION': 'putWGTNetwork'}}
27 , 'UNIQUEATTIBUTES': ['USERNAME'], 'TOKENENTITIESRELATIONSHIPS':
28 ['USER'], 'ATTRIBUTES': {'USERNAME': (2, 'STRING', 0, 'VALUE'),
29 'CITY': (5, 'STRING', 0, 'VALUE'), 'ENTITYKEY': (1, 'STRING', 0,
30 'VALUE'), 'USERLINK': (6, 'STRING', 0, 'VALUE'), 'FULLNAME': (4,
31 'STRING', 0, 'VALUE'), 'USERTAG': (3, 'STRING', 0, 'VALUE'),
32 'ISACTIVE': (0, 'BOOL', 1, 'VALUE')}
33 }
34 , 'COUPON': {'TYPEOFTYPES': ['COUPON'], 'FUNCTIONS':
35 {'ENTITYCREATION': 'putNetwork'}}
36 , 'UNIQUEATTIBUTES': ['COUPONNAME'],
37 'TOKENENTITIESRELATIONSHIPS': ['MERCHANT'], 'ATTRIBUTES':

```

```

1  {'STATUS': (2, 'STRING', 0, 'VALUE'), 'MERCHANT': (3, 'STRING',
2  0, 'VALUE'), 'TITLE': (5, 'STRING', 0, 'VALUE'), 'NOTES': (7,
3  'STRING', 0, 'VALUE'), 'UPDATEDBY': (11, 'STRING', 0, 'VALUE'),
4  'ENTITYKEY': (1, 'STRING', 0, 'VALUE'), 'DESCRIPTION': (6,
5  'STRING', 0, 'VALUE'), 'CREATEDBY': (10, 'STRING', 0, 'VALUE'),
6  'LASTUPDATEDT': (9, 'DATETIME', 0, 'VALUE'), 'EXPDT': (13,
7  'DATETIME', 0, 'VALUE'), 'RESTRICTIONS': (14, 'DICT', 0,
8  'VALUE'), 'COUPONNAME': (4, 'STRING', 0, 'VALUE'), 'CREATIONDT':
9  (8, 'DATETIME', 0, 'VALUE'), 'STARTDT': (12, 'DATETIME', 0,
10 'VALUE'), 'ISACTIVE': (0, 'BOOL', 1, 'VALUE')}
11 }
12 , 'MEMBERSHIP': {'TYPEOFTYPES': ['MEMBERSHIPS'], 'FUNCTIONS':
13 {'ENTITYCREATION': 'putNetwork'}}
14 , 'UNIQUEATTIBUTES': ['MEMBERSHIPNAME'],
15 'TOKENENTITIESRELATIONSHIPS': ['MERCHANT'], 'ATTRIBUTES':
16 {'STATUS': (2, 'STRING', 0, 'VALUE'), 'MERCHANT': (3, 'STRING',
17 0, 'VALUE'), 'RESTRICTIONS': (7, 'DICT', 0, 'VALUE'),
18 'MEMBERSHIPNAME': (4, 'STRING', 0, 'VALUE'), 'STARTDT': (5,
19 'DATETIME', 0, 'VALUE'), 'EXPDT': (6, 'DATETIME', 0, 'VALUE'),
20 'ISACTIVE': (0, 'BOOL', 1, 'VALUE'), 'ENTITYKEY': (1, 'STRING',
21 0, 'VALUE')}
22 }
23 , 'USERSECURITY': {'TYPEOFTYPES': ['SECURITY'], 'FUNCTIONS':
24 {'ENTITYCREATION': 'putNetwork'}}
25 , 'UNIQUEATTIBUTES': ['USERSECURITYNAME'],
26 'TOKENENTITIESRELATIONSHIPS': ['USER'], 'ATTRIBUTES': {'STATUS':
27 (2, 'STRING', 0, 'VALUE'), 'EXPDT': (6, 'DATETIME', 0, 'VALUE'),
28 'USERSECURITYNAME': (4, 'STRING', 0, 'VALUE'), 'USER': (3,
29 'STRING', 0, 'VALUE'), 'RESTRICTIONS': (7, 'DICT', 0, 'VALUE'),
30 'STARTDT': (5, 'DATETIME', 0, 'VALUE'), 'ISACTIVE': (0, 'BOOL',
31 1, 'VALUE'), 'ENTITYKEY': (1, 'STRING', 0, 'VALUE')}
32 }
33 , 'MCC': {'TYPEOFTYPES': ['MCC'], 'FUNCTIONS': {'ENTITYCREATION':
34 'putWGTNetwork'}}
35 , 'UNIQUEATTIBUTES': ['MCCNAME', 'MCC'],
36 'TOKENENTITIESRELATIONSHIPS': ['MCCSEG'], 'ATTRIBUTES':
37 {'MCCSEG': (4, 'STRING', 0, 'VALUE'), 'MCC': (2, 'STRING', 0,

```

```

1  'VALUE'), 'MCCNAME': (3, 'STRING', 0, 'VALUE'), 'ISACTIVE': (0,
2  'BOOL', 1, 'VALUE'), 'ENTITYKEY': (1, 'STRING', 0, 'VALUE'))
3  }
4  , 'ZIPCODE': {'TYPEOFTYPES': ['LOCATION'], 'FUNCTIONS':
5  {'ENTITYCREATION': 'putNetwork'}}
6  , 'UNIQUEATTIBUTES': ['ZIPCODE'], 'TOKENENTITIESRELATIONSHIPS':
7  [], 'ATTRIBUTES': {'STATE': (4, 'STRING', 0, 'VALUE'),
8  'POPULATION': (3, 'STRING', 0, 'VALUE'), 'ZIPCODE': (2, 'STRING',
9  0, 'VALUE'), 'ISACTIVE': (0, 'BOOL', 1, 'VALUE'), 'ENTITYKEY':
10 (1, 'STRING', 0, 'VALUE')}
11 }
12 , 'PAYMENTCARD': {'TYPEOFTYPES': ['PAYMENTCARDS'], 'FUNCTIONS':
13 {'ENTITYCREATION': 'putNetwork'}}
14 , 'UNIQUEATTIBUTES': ['CARDNUMBER'],
15 'TOKENENTITIESRELATIONSHIPS': ['USER'], 'ATTRIBUTES': {'EXPDATE':
16 (5, 'DATETIME', 0, 'VALUE'), 'ENTITYKEY': (1, 'STRING', 0,
17 'VALUE'), 'CARDTYPE': (4, 'STRING', 0, 'VALUE'), 'CARDNUMBER':
18 (2, 'STRING', 0, 'VALUE'), 'USER': (3, 'STRING', 0, 'VALUE'),
19 'ISACTIVE': (0, 'BOOL', 1, 'VALUE')}
20 }
21 , 'GENERICTOKEN': {'TYPEOFTYPES': ['COUPON'], 'FUNCTIONS':
22 {'ENTITYCREATION': 'putNetwork'}}
23 , 'UNIQUEATTIBUTES': ['GENERICTOKENNAME'],
24 'TOKENENTITIESRELATIONSHIPS': ['MERCHANT'], 'ATTRIBUTES':
25 {'STATUS': (2, 'STRING', 0, 'VALUE'), 'MERCHANT': (3, 'STRING',
26 0, 'VALUE'), 'TITLE': (5, 'STRING', 0, 'VALUE'), 'NOTES': (7,
27 'STRING', 0, 'VALUE'), 'UPDATEDBY': (11, 'STRING', 0, 'VALUE'),
28 'ENTITYKEY': (1, 'STRING', 0, 'VALUE'), 'DESCRIPTION': (6,
29 'STRING', 0, 'VALUE'), 'CREATEDBY': (10, 'STRING', 0, 'VALUE'),
30 'LASTUPDATEDT': (9, 'DATETIME', 0, 'VALUE'), 'EXPDT': (13,
31 'DATETIME', 0, 'VALUE'), 'RESTRICTIONS': (14, 'DICT', 0,
32 'VALUE'), 'STARTDT': (12, 'DATETIME', 0, 'VALUE'), 'CREATIONDT':
33 (8, 'DATETIME', 0, 'VALUE'), 'GENERICTOKENNAME': (4, 'STRING', 0,
34 'VALUE'), 'ISACTIVE': (0, 'BOOL', 1, 'VALUE')}
35 }
36 , 'USER': {'TYPEOFTYPES': ['USERS', 'SYNTHETICNETWORKS'],
37 'FUNCTIONS': {'ENTITYCREATION': 'putNetwork'}}

```

```

1  , 'UNIQUEATTIBUTES': ['USERNAME'], 'TOKENENTITIESRELATIONSHIPS':
2  ['USERS'], 'ATTRIBUTES': {'USERNAME': (5, 'STRING', 0, 'VALUE'),
3  'USERS': (2, 'STRING', 0, 'VALUE'), 'FIRSTNAME': (3, 'STRING', 0,
4  'VALUE'), 'LASTNAME': (4, 'STRING', 0, 'VALUE'), 'ENTITYKEY': (1,
5  'STRING', 0, 'VALUE'), 'ISACTIVE': (0, 'BOOL', 1, 'VALUE')}
6  }
7  , 'TWEETS': {'TYPEOFTYPES': ['TOKENENTITY'], 'FUNCTIONS':
8  {'ENTITYCREATION': 'putWGTNetwork'}}
9  , 'UNIQUEATTIBUTES': ['TWEETID'], 'TOKENENTITIESRELATIONSHIPS':
10 ['TWITTERUSER'], 'ATTRIBUTES': {'Title': (4, 'STRING', 0,
11 'VALUE'), 'RawTweet': (5, 'STRING', 0, 'VALUE'), 'DATETIME': (3,
12 'STRING', 0, 'VALUE'), 'CLEANEDTWEET': (6, 'STRING', 0, 'VALUE'),
13 'ENTITYKEY': (1, 'STRING', 0, 'VALUE'), 'TWEETID': (2, 'STRING',
14 0, 'VALUE'), 'ISACTIVE': (0, 'BOOL', 1, 'VALUE')}
15 }
16 , 'MODEL': {'TYPEOFTYPES': ['MODELS'], 'FUNCTIONS':
17 {'ENTITYCREATION': 'putNetwork'}}
18 , 'UNIQUEATTIBUTES': ['MODELNAME'], 'TOKENENTITIESRELATIONSHIPS':
19 ['USER', 'MERCHANT', 'PAYMENTCARD'], 'ATTRIBUTES': {'XML': (2,
20 'STRING', 0, 'VALUE'), 'MODELNAME': (3, 'STRING', 0, 'VALUE'),
21 'DESCRIPTION': (4, 'STRING', 0, 'VALUE'), 'ENTITYKEY': (1,
22 'STRING', 0, 'VALUE'), 'TYPEOF': (5, 'STRING', 0, 'VALUE'),
23 'ISACTIVE': (0, 'BOOL', 1, 'VALUE')}
24 }
25 , 'MCCSEG': {'TYPEOFTYPES': ['MCCSEG'], 'FUNCTIONS':
26 {'ENTITYCREATION': 'putWGTNetwork'}}
27 , 'UNIQUEATTIBUTES': ['MCCSEGID'], 'TOKENENTITIESRELATIONSHIPS':
28 {}
29 , 'ATTRIBUTES': {'MCCSEGID': (2, 'STRING', 0, 'VALUE'),
30 'MCCSEGNAME': (3, 'STRING', 0, 'VALUE'), 'ISACTIVE': (0, 'BOOL',
31 1, 'VALUE'), 'ENTITYKEY': (1, 'STRING', 0, 'VALUE')}
32 }
33 , 'TOKENENTITY': {'TYPEOFTYPES': ['TOKENENTITY'], 'FUNCTIONS':
34 {'ENTITYCREATION': 'putWGTNetwork'}}
35 , 'UNIQUEATTIBUTES': ['TOKENENTITYKEY'],
36 'TOKENENTITIESRELATIONSHIPS': {}

```

```

1  , 'ATTRIBUTES': {'STATUS': (4, 'STRING', 0, 'VALUE'),
2  'ISSUEDDATE': (5, 'STRING', 0, 'VALUE'), 'DOUBLELINKED': (8,
3  'BOOL', 1, 'VALUE'), 'BASEUUID': (1, 'STRING', 0, 'VALUE'),
4  'WEIGHT': (6, 'STRING', 0, 'VALUE'), 'BASETYPE': (3, 'STRING', 0,
5  'VALUE'), 'CATEGORY': (7, 'STRING', 0, 'VALUE'), 'ISACTIVE': (0,
6  'BOOL', 1, 'VALUE'), 'TOKENENTITYKEY': (2, 'STRING', 0, 'VALUE')}
7  }
8  }

```

[0073] FIGURE 3 shows a block diagram illustrating example MMCDB component configurations in some embodiments of the MMCDB. In some embodiments, the MMCDB may aggregate data from a variety of sources to generate centralized personal information. The may also aggregate various types of data in order to generate the centralized personal information. For example, the MMCDB may utilize search results aggregation component(s) 301 (e.g., such as described in FIGS. 4-5) to aggregate search results from across a wide range of computer networked systems, e.g., the Internet. As another example, the MMCDB may utilize transaction data aggregation component(s) 302 (e.g., such as described in FIGS. 6-9) to aggregate transaction data, e.g., from transaction processing procedure by a payment network. As another example, the MMCDB may utilize service usage data aggregation component(s) 303 (e.g., such as described in FIGS. 6-9) to aggregate data on user's usage of various services associated with the MMCDB. As another example, the MMCDB may utilize enrollment data component(s) 304 (e.g., such as described in FIGS. 12-13) to aggregate data on user's enrollment into various services associated with the MMCDB. As another example, the MMCDB may utilize email data component(s) 305a (e.g., such as described in FIG. 37) to aggregate data regarding the user's email correspondence history into various services associated with the MMCDB. As another example, the MMCDB may utilize social data aggregation component(s) 305 (e.g., such as described in FIGS. 10-11) to aggregate data on user's usage of various social networking services accessible by the MMCDB. In one embodiment, the aggregated data may be used to generate dictionary entries. Further detail regarding the generation of dictionary entries may be found throughout this specification, drawings, and claims and particularly with reference to Fig. 1 and Fig. 46.

1 **[0074]** In some embodiments, the MMCDB may acquire the aggregated data, and
2 normalize the data into formats that are suitable for uniform storage, indexing,
3 maintenance, and/or further processing via data record normalization component(s)
4 306 (e.g., such as described in FIGS. 14A-B). The MMCDB may extract data from the
5 normalized data records, and recognize data fields, e.g., the MMCDB may identify the
6 attributes of each field of data included in the normalized data records via data field
7 recognition component(s) 307 (e.g., such as described in FIG. 15). For example, the
8 MMCDB may identify names, user ID(s), addresses, network addresses, comments
9 and/or specific words within the comments, images, blog posts, video, content within
10 the video, and/or the like from the aggregated data. In some embodiments, for each
11 field of data, the MMCDB may classify entity types associated with the field of data, as
12 well as entity identifiers associated with the field of data, e.g., via component(s) 308
13 (e.g., such as described in FIG. 16). For example, the MMCDB may identify an Internet
14 Protocol (IP) address data field to be associated with a user ID john.q.public (consumer
15 entity type), a user John Q. Public (consumer entity type), a household (the Public
16 household - a multi-consumer entity type / household entity type), a merchant entity
17 type with identifier Acme Merchant Store, Inc. from which purchases are made from the
18 IP address, an Issuer Bank type with identifier First National Bank associated with the
19 purchases made from the IP address, and/or the like. In some embodiments, the
20 MMCDB may utilize the entity types and entity identifiers to correlate entities across
21 each other, e.g., via cross-entity correlation component(s) 309 (e.g., such as described in
22 FIG. 17). For example, the MMCDB may identify, from the aggregated data, that a
23 household entity with identifier H123 may include a user entity with identifier John Q.
24 Public and social identifier john.q.public@facebook.com, a second user entity with
25 identifier Jane P. Doe with social identifier jpdoe@twitter.com, a computer entity with
26 identifier IP address 192.168.4.5, a card account entity with identifier ****1234, a bank
27 issuer entity with identifier AB23145, a merchant entity with identifier Acme Stores, Inc.
28 where the household sub-entities make purchases, and/or the like. In some
29 embodiments, the MMCDB may utilize the entity identifiers, data associated with each
30 entity and/or correlated entities to identify associations to other entities, e.g., via entity
31 attribute association component(s) 310 (e.g., such as described in FIG. 18). For
32 example, the MMCDB may identify specific purchases made via purchase transactions

1 by members of the household, and thereby identify attributes of members of the
 2 household on the basis of the purchases in the purchase transactions made by members
 3 of the household. Based on such correlations and associations, the MMCDB may update
 4 a profile for each entity identified from the aggregated data, as well as a social graph
 5 interrelating the entities identified in the aggregated data, e.g., via entity profile-graph
 6 updating component(s) 311 (e.g., such as described in FIGS. 19, 40, 41A-E and 42A-C).
 7 In some embodiments, the updating of profile and/or social graphs for an entity may
 8 trigger a search for additional data that may be relevant to the newly identified
 9 correlations and associations for each entity, e.g., via search term generation
 10 component(s) 313-314 (e.g., such as described in FIG. 20). For example, the updating of
 11 a profile and/or social graph may trigger searches across the Internet, social networking
 12 websites, transaction data from payment networks, services enrolled into and/or
 13 utilized by the entities, and/or the like. In some embodiments, such updating of entity
 14 profiles and/or social graphs may be performed continuously, periodically, on-demand,
 15 and/or the like.

16 **[0075]** FIGURE 4 shows a data flow diagram illustrating an example search result
 17 aggregation procedure in some embodiments of the MMCDB. In some
 18 implementations, the pay network server may obtain a trigger to perform a search. For
 19 example, the pay network server may periodically perform a search update of its
 20 aggregated search database, e.g., 410, with new information available from a variety of
 21 sources, such as the Internet. As another example, a request for on-demand search
 22 update may be obtained as a result of a user wishing to enroll in a service, for which the
 23 pay network server may facilitate data entry by providing an automated web form filling
 24 system using information about the user obtained from the search update. In some
 25 implementations, the pay network server may parse the trigger to extract keywords
 26 using which to perform an aggregated search. The pay network server may generate a
 27 query for application programming interface (API) templates for various search engines
 28 (e.g., Google™, Bing®, AskJeeves, market data search engines, etc.) from which to
 29 collect data for aggregation. The pay network server may query, e.g., 412, a pay network
 30 database, e.g., 407, for search API templates for the search engines. For example, the
 31 pay network server may utilize PHP/SQL commands similar to the examples provided

1 above. The database may provide, e.g., 413, a list of API templates in response. Based
 2 on the list of API templates, the pay network server may generate search requests, e.g.,
 3 414. The pay network server may issue the generated search requests, e.g., 415a-c, to
 4 the search engine servers, e.g., 401a-c. For example, the pay network server may issue
 5 PHP commands to request the search engine for search results. An example listing of
 6 commands to issue search requests 415a-c, substantially in the form of PHP commands,
 7 is provided below:

```

8  <?PHP
9  // API URL with access key
10 $url =
11 ["https://ajax.googleapis.com/ajax/services/search/web?v=1.0&"
12   . "q=" $keywords
13   "&key=1234567890987654&userip=datagraph.cpip.com"];
14
15 // Send Search Request
16 $ch = curl_init();
17 curl_setopt($ch, CURLOPT_URL, $url);
18 curl_setopt($ch, CURLOPT_RETURNTRANSFER, 1);
19 curl_setopt($ch, CURLOPT_REFERER, "datagraph.cpip.com");
20 $body = curl_exec($ch);
21 curl_close($ch);
22
23 // Obtain, parse search results
24 $json = json_decode($body);
25 ?>
```

26 **[0076]** In some embodiments, the search engine servers may query, e.g., 417a-c,
 27 their search databases, e.g., 402a-c, for search results falling within the scope of the
 28 search keywords. In response to the search queries, the search databases may provide
 29 search results, e.g., 418a-c, to the search engine servers. The search engine servers may
 30 return the search results obtained from the search databases, e.g., 419a-c, to the pay
 31 network server making the search requests. An example listing of search results 419a-c,
 32 substantially in the form of JavaScript Object Notation (JSON)-formatted data, is
 33 provided below:

```

1  {"responseData": {
2    "results": [
3      {
4        "GsearchResultClass": "GwebSearch",
5        "unescapedUrl": "http://en.wikipedia.org/wiki/John_Q_Public",
6        "url": "http://en.wikipedia.org/wiki/John_Q_Public",
7        "visibleUrl": "en.wikipedia.org",
8        "cacheUrl":
9        "http://www.google.com/search?q\u003dcache:TwrPfhd22hYJ:en.wikipe
10       dia.org",
11        "title": "\u003cb\u003eJohn Q. Public\u003c/b\u003e -
12       Wikipedia, the free encyclopedia",
13        "titleNoFormatting": "John Q. Public - Wikipedia, the free
14       encyclopedia",
15        "content": "\[1\] In 2006, he served as Chief Technology
16       Officer..."
17      },
18      {
19        "GsearchResultClass": "GwebSearch",
20        "unescapedUrl": "http://www.imdb.com/name/nm0385296/",
21        "url": "http://www.imdb.com/name/nm0385296/",
22        "visibleUrl": "www.imdb.com",
23        "cacheUrl":
24        "http://www.google.com/search?q\u003dcache:1i34KkqnsooJ:www.imdb.
25       com",
26        "title": "\u003cb\u003eJohn Q. Public\u003c/b\u003e",
27        "titleNoFormatting": "John Q. Public",
28        "content": "Self: Zoolander. Socialite \u003cb\u003eJohn Q.
29       Public\u003c/b\u003e..."
30      },
31      ...
32    ],
33    "cursor": {
34      "pages": [
35        { "start": "0", "label": 1 },
36        { "start": "4", "label": 2 },
37        { "start": "8", "label": 3 },

```

```

1      { "start": "12","label": 4 }
2    ],
3    "estimatedResultCount": "59600000",
4    "currentPageIndex": 0,
5    "moreResultsUrl":
6    "http://www.google.com/search?oe\u003dutf8\u0026ie\u003dutf8..."
7  }
8  }
9  , "responseDetails": null, "responseStatus": 200}

```

10 **[0077]** In some embodiments, the pay network server may store the aggregated
 11 search results, e.g., 420, in an aggregated search database, e.g., 410a.

12 **[0078]** FIGURE 5 shows a logic flow diagram illustrating example aspects of
 13 aggregating search results in some embodiments of the MMCDB, e.g., a Search Results
 14 Aggregation (“SRA”) component 500. In some implementations, the pay network server
 15 may obtain a trigger to perform a search, e.g., 501. For example, the pay network server
 16 may periodically perform a search update of its aggregated search database with new
 17 information available from a variety of sources, such as the Internet. As another
 18 example, a request for on-demand search update may be obtained as a result of a user
 19 wishing to enroll in a service, for which the pay network server may facilitate data entry
 20 by providing an automated web form filling system using information about the user
 21 obtained from the search update. In some implementations, the pay network server
 22 may parse the trigger, e.g., 502, to extract keywords using which to perform an
 23 aggregated search. The pay network server may determine the search engines to search,
 24 e.g., 503, using the extracted keywords. Then, the pay network server may generate a
 25 query for application programming interface (API) templates for the various search
 26 engines (e.g., Google™, Bing®, AskJeeves, market data search engines, etc.) from which
 27 to collect data for aggregation, e.g., 504. The pay network server may query, e.g., 505, a
 28 pay network database for search API templates for the search engines. For example, the
 29 pay network server may utilize PHP/SQL commands similar to the examples provided
 30 above. The database may provide, e.g., 505, a list of API templates in response. Based
 31 on the list of API templates, the pay network server may generate search requests, e.g.,
 32 506. The pay network server may issue the generated search requests to the search

1 engine servers. The search engine servers may parse the obtained search results(s), e.g.,
 2 507, and query, e.g., 508, their search databases for search results falling within the
 3 scope of the search keywords. In response to the search queries, the search databases
 4 may provide search results, e.g., 509, to the search engine servers. The search engine
 5 servers may return the search results obtained from the search databases, e.g., 510, to
 6 the pay network server making the search requests. The pay network server may
 7 generate, e.g., 511, and store the aggregated search results, e.g., 512, in an aggregated
 8 search database.

9 **[0079]** FIGURES 6A-D show data flow diagrams illustrating an example card-
 10 based transaction execution procedure in some embodiments of the MMCDB. In some
 11 implementations, a user, e.g., 601, may desire to purchase a product, service, offering,
 12 and/or the like (“product”), from a merchant. The user may communicate with a
 13 merchant server, e.g., 603, via a client such as, but not limited to: a personal computer,
 14 mobile device, television, point-of-sale terminal, kiosk, ATM, and/or the like (e.g., 602).
 15 For example, the user may provide user input, e.g., purchase input 611, into the client
 16 indicating the user’s desire to purchase the product. In various implementations, the
 17 user input may include, but not be limited to: keyboard entry, card swipe, activating a
 18 RFID/NFC enabled hardware device (e.g., electronic card having multiple accounts,
 19 smartphone, tablet, etc.), mouse clicks, depressing buttons on a joystick/game console,
 20 voice commands, single/multi-touch gestures on a touch-sensitive interface, touching
 21 user interface elements on a touch-sensitive display, and/or the like. For example, the
 22 user may direct a browser application executing on the client device to a website of the
 23 merchant, and may select a product from the website via clicking on a hyperlink
 24 presented to the user via the website. As another example, the client may obtain track 1
 25 data from the user’s card (e.g., credit card, debit card, prepaid card, charge card, etc.),
 26 such as the example track 1 data provided below:

```

27 %B123456789012345^PUBLIC/J.Q.^99011200000000000000**901*****?*
28 (wherein '123456789012345' is the card number of 'J.Q. Public'
29 and has a CVV number of 901. '990112' is a service code, and ***
30 represents decimal digits which change randomly each time the
31 card is used.)

```

1 **[0080]** In some implementations, the client may generate a purchase order
 2 message, e.g., 612, and provide, e.g., 613, the generated purchase order message to the
 3 merchant server. For example, a browser application executing on the client may
 4 provide, on behalf of the user, a (Secure) Hypertext Transfer Protocol (“HTTP(S)”) GET
 5 message including the product order details for the merchant server in the form of data
 6 formatted according to the eXtensible Markup Language (“XML”). Below is an example
 7 HTTP(S) GET message including an XML-formatted purchase order message for the
 8 merchant server:

```

9  GET /purchase.php HTTP/1.1
10 Host: www.merchant.com
11 Content-Type: Application/XML
12 Content-Length: 1306
13 <?XML version = "1.0" encoding = "UTF-8"?>
14 <purchase_order>
15     <order_ID>4NFU4RG94</order_ID>
16     <timestamp>2011-02-22 15:22:43</timestamp>
17     <user_ID>john.q.public@gmail.com</user_ID>
18     <client_details>
19         <client_IP>192.168.23.126</client_IP>
20         <client_type>smartphone</client_type>
21         <client_model>HTC Hero</client_model>
22         <OS>Android 2.2</OS>
23         <app_installed_flag>true</app_installed_flag>
24     </client_details>
25     <purchase_details>
26         <num_products>1</num_products>
27         <product>
28             <product_type>book</product_type>
29             <product_params>
30                 <product_title>XML for
31 dummies</product_title>
32                 <ISBN>938-2-14-168710-0</ISBN>
33                 <edition>2nd ed.</edition>
34                 <cover>hardbound</cover>
35                 <seller>bestbuybooks</seller>

```

```

1          </product_params>
2          <quantity>1</quantity>
3      </product>
4  </purchase_details>
5  <account_params>
6      <account_name>John Q. Public</account_name>
7      <account_type>credit</account_type>
8      <account_num>123456789012345</account_num>
9      <billing_address>123 Green St., Norman, OK
10 98765</billing_address>
11      <phone>123-456-7809</phone>
12      <sign>/jpg</sign>
13      <confirm_type>email</confirm_type>
14
15      <contact_info>john.q.public@gmail.com</contact_info>
16  </account_params>
17  <shipping_info>
18      <shipping_address>same as billing</shipping_address>
19      <ship_type>expedited</ship_type>
20      <ship_carrier>FedEx</ship_carrier>
21      <ship_account>123-45-678</ship_account>
22      <tracking_flag>true</tracking_flag>
23      <sign_flag>false</sign_flag>
24  </shipping_info>
25  </purchase_order>

```

26 **[0081]** In some implementations, the merchant server may obtain the purchase
 27 order message from the client, and may parse the purchase order message to extract
 28 details of the purchase order from the user. The merchant server may generate a card
 29 query request, e.g., 614 to determine whether the transaction can be processed. For
 30 example, the merchant server may attempt to determine whether the user has sufficient
 31 funds to pay for the purchase in a card account provided with the purchase order. The
 32 merchant server may provide the generated card query request, e.g., 615, to an acquirer
 33 server, e.g., 604. For example, the acquirer server may be a server of an acquirer
 34 financial institution (“acquirer”) maintaining an account of the merchant. For example,

1 the proceeds of transactions processed by the merchant may be deposited into an
 2 account maintained by the acquirer. In some implementations, the card query request
 3 may include details such as, but not limited to: the costs to the user involved in the
 4 transaction, card account details of the user, user billing and/or shipping information,
 5 and/or the like. For example, the merchant server may provide a HTTP(S) POST
 6 message including an XML-formatted card query request similar to the example listing
 7 provided below:

```

8  POST /cardquery.php HTTP/1.1
9  Host: www.acquirer.com
10 Content-Type: Application/XML
11 Content-Length: 624
12 <?XML version = "1.0" encoding = "UTF-8"?>
13 <card_query_request>
14     <query_ID>VNEI39FK</query_ID>
15     <timestamp>2011-02-22 15:22:44</timestamp>
16     <purchase_summary>
17         <num_products>1</num_products>
18         <product>
19             <product_summary>Book - XML for
20 dummies</product_summary>
21             <product_quantity>1</product_quantity?
22         </product>
23     </purchase_summary>
24     <transaction_cost>$34.78</transaction_cost>
25     <account_params>
26         <account_name>John Q. Public</account_name>
27         <account_type>credit</account_type>
28         <account_num>123456789012345</account_num>
29         <billing_address>123 Green St., Norman, OK
30 98765</billing_address>
31         <phone>123-456-7809</phone>
32         <sign>/jqp</sign>
33     </account_params>
34     <merchant_params>
35         <merchant_id>3FBCR4INC</merchant_id>

```

```

1         <merchant_name>Books & Things, Inc.</merchant_name>
2
3         <merchant_auth_key>1NNF484MCP59CHB27365</merchant_auth_ke
4 y>
5         </merchant_params>
6     </card_query_request>

```

7 **[0082]** In some implementations, the acquirer server may generate a card
 8 authorization request, e.g., 616, using the obtained card query request, and provide the
 9 card authorization request, e.g., 617, to a pay network server, e.g., 605. For example, the
 10 acquirer server may redirect the HTTP(S) POST message in the example above from the
 11 merchant server to the pay network server.

12 **[0083]** In some implementations, the pay network server may determine whether
 13 the user has enrolled in value-added user services. For example, the pay network server
 14 may query 618 a database, e.g., pay network database 407, for user service enrollment
 15 data. For example, the server may utilize PHP/SQL commands similar to the example
 16 provided above to query the pay network database. In some implementations, the
 17 database may provide the user service enrollment data, e.g., 619. The user enrollment
 18 data may include a flag indicating whether the user is enrolled or not, as well as
 19 instructions, data, login URL, login API call template and/or the like for facilitating
 20 access of the user-enrolled services. For example, in some implementations, the pay
 21 network server may redirect the client to a value-add server (e.g., such as a social
 22 network server where the value-add service is related to social networking) by providing
 23 a HTTP(S) REDIRECT 300 message, similar to the example below:

```

24 HTTP/1.1 300 Multiple Choices
25 Location:
26 https://www.facebook.com/dialog/oauth?client_id=snpa_app_ID&redir
27 ect_uri= www.paynetwork.com/purchase.php
28 <html>
29     <head><title>300 Multiple Choices</title></head>
30     <body><h1>Multiple Choices</h1></body>
31 </html>

```

1 **[0084]** In some implementations, the pay network server may provide payment
 2 information extracted from the card authorization request to the value-add server as
 3 part of a value add service request, e.g., 620. For example, the pay network server may
 4 provide a HTTP(S) POST message to the value-add server, similar to the example below:

```

5  POST /valueservices.php HTTP/1.1
6  Host: www.valueadd.com
7  Content-Type: Application/XML
8  Content-Length: 1306
9  <?XML version = "1.0" encoding = "UTF-8"?>
10 <service_request>
11     <request_ID>4NFU4RG94</order_ID>
12     <timestamp>2011-02-22 15:22:43</timestamp>
13     <user_ID>john.q.public@gmail.com</user_ID>
14     <client_details>
15         <client_IP>192.168.23.126</client_IP>
16         <client_type>smartphone</client_type>
17         <client_model>HTC Hero</client_model>
18         <OS>Android 2.2</OS>
19         <app_installed_flag>true</app_installed_flag>
20     </client_details>
21     <account_params>
22         <account_name>John Q. Public</account_name>
23         <account_type>credit</account_type>
24         <account_num>123456789012345</account_num>
25         <billing_address>123 Green St., Norman, OK
26 98765</billing_address>
27         <phone>123-456-7809</phone>
28         <sign>/jpg</sign>
29         <confirm_type>email</confirm_type>
30
31     <contact_info>john.q.public@gmail.com</contact_info>
32 </account_params>
33 <!--optional-->
34 <merchant>
35     <merchant_id>CQN3Y42N</merchant_id>
36     <merchant_name>Acme Tech, Inc.</merchant_name>

```

```

1         <user_name>john.q.public</user_name>
2         <cardlist>
3     www.acme.com/user/john.q.public/cclist.xml<cardlist>
4         <user_account_preference>1 3 2 4 7 6
5     5<user_account_preference>
6         </merchant>
7     </service_request>

```

8 **[0085]** In some implementations, the value-add server may provide a service
9 input request, e.g., 621, to the client. For example, the value-add server may provide a
10 HTML input/login form to the client. The client may display, e.g., 622, the login form
11 for the user. In some implementations, the user may provide login input into the client,
12 e.g., 623, and the client may generate a service input response, e.g., 624, for the value-
13 add server. In some implementations, the value-add server may provide value-add
14 services according to user value-add service enrollment data, user profile, etc., stored on
15 the value-add server, and based on the user service input. Based on the provision of
16 value-add services, the value-add server may generate a value-add service response, e.g.,
17 626, and provide the response to the pay network server. For example, the value-add
18 server may provide a HTTP(S) POST message similar to the example below:

```

19     POST /serviceresponse.php HTTP/1.1
20     Host: www.paynet.com
21     Content-Type: Application/XML
22     Content-Length: 1306
23     <?XML version = "1.0" encoding = "UTF-8"?>
24     <service_response>
25         <request_ID>4NFU4RG94</order_ID>
26         <timestamp>2011-02-22 15:22:43</timestamp>
27         <result>serviced</result>
28         <servcode>943528976302-45569-003829-04</servcode>
29     </service_response>

```

30 **[0086]** In some implementations, upon receiving the value-add service response
31 from the value-add server, the pay network server may extract the enrollment service
32 data from the response for addition to a transaction data record. In some

1 implementations, the pay network server may forward the card authorization request to
 2 an appropriate pay network server, e.g., 628, which may parse the card authorization
 3 request to extract details of the request. Using the extracted fields and field values, the
 4 pay network server may generate a query, e.g., 629, for an issuer server corresponding
 5 to the user's card account. For example, the user's card account, the details of which the
 6 user may have provided via the client-generated purchase order message, may be linked
 7 to an issuer financial institution ("issuer"), such as a banking institution, which issued
 8 the card account for the user. An issuer server, e.g., 608a-n, of the issuer may maintain
 9 details of the user's card account. In some implementations, a database, e.g., pay
 10 network database 607, may store details of the issuer servers and card account numbers
 11 associated with the issuer servers. For example, the database may be a relational
 12 database responsive to Structured Query Language ("SQL") commands. The pay
 13 network server may execute a hypertext preprocessor ("PHP") script including SQL
 14 commands to query the database for details of the issuer server. An example PHP/SQL
 15 command listing, illustrating substantive aspects of querying the database, is provided
 16 below:

```

17  <?PHP
18  header('Content-Type: text/plain');
19  mysql_connect("254.93.179.112", $DBserver, $password); // access
20  database server
21  mysql_select_db("ISSUERS.SQL"); // select database table to
22  search
23  //create query for issuer server data
24  $query = "SELECT issuer_name issuer_address issuer_id ip_address
25  mac_address auth_key port_num security_settings_list FROM
26  IssuerTable WHERE account_num LIKE '%" $accountnum";
27  $result = mysql_query($query); // perform the search query
28  mysql_close("ISSUERS.SQL"); // close database access
29  ?>
```

30 **[0087]** In response to obtaining the issuer server query, e.g., 629, the pay network
 31 database may provide, e.g., 630, the requested issuer server data to the pay network
 32 server. In some implementations, the pay network server may utilize the issuer server

1 data to generate a forwarding card authorization request, e.g., 631, to redirect the card
 2 authorization request from the acquirer server to the issuer server. The pay network
 3 server may provide the card authorization request, e.g., 632, to the issuer server. In
 4 some implementations, the issuer server, e.g., 608, may parse the card authorization
 5 request, and based on the request details may query 633 a database, e.g., user profile
 6 database 609, for data of the user's card account. For example, the issuer server may
 7 issue PHP/SQL commands similar to the example provided below:

```

8  <?PHP
9  header('Content-Type: text/plain');
10 mysql_connect("254.93.179.112", $DBserver, $password); // access
11 database server
12 mysql_select_db("USERS.SQL"); // select database table to search
13 //create query for user data
14 $query = "SELECT user_id user_name user_balance account_type FROM
15 UserTable WHERE account_num LIKE '%" $accountnum";
16 $result = mysql_query($query); // perform the search query
17 mysql_close("USERS.SQL"); // close database access
18 ?>
```

19 **[0088]** In some implementations, on obtaining the user data, e.g., 634, the issuer
 20 server may determine whether the user can pay for the transaction using funds available
 21 in the account, e.g., 635. For example, the issuer server may determine whether the user
 22 has a sufficient balance remaining in the account, sufficient credit associated with the
 23 account, and/or the like. If the issuer server determines that the user can pay for the
 24 transaction using the funds available in the account, the server may provide an
 25 authorization message, e.g., 636, to the pay network server. For example, the server
 26 may provide a HTTP(S) POST message similar to the examples above.

27 **[0089]** In some implementations, the pay network server may obtain the
 28 authorization message, and parse the message to extract authorization details. Upon
 29 determining that the user possesses sufficient funds for the transaction, the pay network
 30 server may generate a transaction data record from the card authorization request it
 31 received, and store, e.g., 639, the details of the transaction and authorization relating to
 32 the transaction in a database, e.g., pay network database 607. For example, the pay

1 network server may issue PHP/SQL commands similar to the example listing below to
 2 store the transaction data in a database:

```

3  <?PHP
4  header('Content-Type: text/plain');
5  mysql_connect("254.92.185.103", $DBserver, $password); // access
6  database server
7  mysql_select("TRANSACTIONS.SQL"); // select database to append
8  mysql_query("INSERT INTO PurchasesTable (timestamp,
9  purchase_summary_list, num_products, product_summary,
10 product_quantity, transaction_cost, account_params_list,
11 account_name, account_type, account_num, billing_addres, zipcode,
12 phone, sign, merchant_params_list, merchant_id, merchant_name,
13 merchant_auth_key)
14 VALUES (time(), $purchase_summary_list, $num_products,
15 $product_summary, $product_quantity, $transaction_cost,
16 $account_params_list, $account_name, $account_type, $account_num,
17 $billing_addres, $zipcode, $phone, $sign, $merchant_params_list,
18 $merchant_id, $merchant_name, $merchant_auth_key)"); // add data
19 to table in database
20 mysql_close("TRANSACTIONS.SQL"); // close connection to database
21 ?>
```

22 **[0090]** In some implementations, the pay network server may forward the
 23 authorization message, e.g., 640, to the acquirer server, which may in turn forward the
 24 authorization message, e.g., 640, to the merchant server. The merchant may obtain the
 25 authorization message, and determine from it that the user possesses sufficient funds in
 26 the card account to conduct the transaction. The merchant server may add a record of
 27 the transaction for the user to a batch of transaction data relating to authorized
 28 transactions. For example, the merchant may append the XML data pertaining to the
 29 user transaction to an XML data file comprising XML data for transactions that have
 30 been authorized for various users, e.g., 641, and store the XML data file, e.g., 642, in a
 31 database, e.g., merchant database 604. For example, a batch XML data file may be
 32 structured similar to the example XML data structure template provided below:

```

33  <?XML version = "1.0" encoding = "UTF-8"?>
```

```

1  <merchant_data>
2      <merchant_id>3FBCR4INC</merchant_id>
3      <merchant_name>Books & Things, Inc.</merchant_name>
4      <merchant_auth_key>1NNF484MCP59CHB27365</merchant_auth_ke
5  y>
6      <account_number>123456789</account_number>
7  </merchant_data>
8  <transaction_data>
9      <transaction 1>
10         ...
11     </transaction 1>
12     <transaction 2>
13         ...
14     </transaction 2>
15         .
16         .
17         .
18     <transaction n>
19         ...
20     </transaction n>
21 </transaction_data>

```

22 **[0091]** In some implementations, the server may also generate a purchase receipt,
 23 e.g., 643, and provide the purchase receipt to the client. The client may render and
 24 display, e.g., 644, the purchase receipt for the user. For example, the client may render
 25 a webpage, electronic message, text / SMS message, buffer a voicemail, emit a ring tone,
 26 and/or play an audio message, etc., and provide output including, but not limited to:
 27 sounds, music, audio, video, images, tactile feedback, vibration alerts (e.g., on vibration-
 28 capable client devices such as a smartphone etc.), and/or the like.

29 **[0092]** With reference to FIGURE 6C, in some implementations, the merchant
 30 server may initiate clearance of a batch of authorized transactions. For example, the
 31 merchant server may generate a batch data request, e.g., 645, and provide the request,
 32 e.g., 646, to a database, e.g., merchant database 604. For example, the merchant server
 33 may utilize PHP/SQL commands similar to the examples provided above to query a

1 relational database. In response to the batch data request, the database may provide the
 2 requested batch data, e.g., 647. The server may generate a batch clearance request, e.g.,
 3 648, using the batch data obtained from the database, and provide, e.g., 641, the batch
 4 clearance request to an acquirer server, e.g., 610. For example, the merchant server may
 5 provide a HTTP(S) POST message including XML-formatted batch data in the message
 6 body for the acquirer server. The acquirer server may generate, e.g., 650, a batch
 7 payment request using the obtained batch clearance request, and provide the batch
 8 payment request to the pay network server, e.g., 651. The pay network server may parse
 9 the batch payment request, and extract the transaction data for each transaction stored
 10 in the batch payment request, e.g., 652. The pay network server may store the
 11 transaction data, e.g., 653, for each transaction in a database, e.g., pay network database
 12 607. For each extracted transaction, the pay network server may query, e.g., 654-655, a
 13 database, e.g., pay network database 607, for an address of an issuer server. For
 14 example, the pay network server may utilize PHP/SQL commands similar to the
 15 examples provided above. The pay network server may generate an individual payment
 16 request, e.g., 656, for each transaction for which it has extracted transaction data, and
 17 provide the individual payment request, e.g., 657, to the issuer server, e.g., 608. For
 18 example, the pay network server may provide a HTTP(S) POST request similar to the
 19 example below:

```

20  POST /requestpay.php HTTP/1.1
21  Host: www.issuer.com
22  Content-Type: Application/XML
23  Content-Length: 788
24  <?XML version = "1.0" encoding = "UTF-8"?>
25  <pay_request>
26      <request_ID>CNI4ICNW2</request_ID>
27      <timestamp>2011-02-22 17:00:01</timestamp>
28      <pay_amount>$34.78</pay_amount>
29      <account_params>
30          <account_name>John Q. Public</account_name>
31          <account_type>credit</account_type>
32          <account_num>123456789012345</account_num>

```

```

1           <billing_address>123 Green St., Norman, OK
2 98765</billing_address>
3           <phone>123-456-7809</phone>
4           <sign>/jqp</sign>
5       </account_params>
6       <merchant_params>
7           <merchant_id>3FBCR4INC</merchant_id>
8           <merchant_name>Books & Things, Inc.</merchant_name>
9
10          <merchant_auth_key>1NNF484MCP59CHB27365</merchant_auth_ke
11 y>
12      </merchant_params>
13      <purchase_summary>
14          <num_products>1</num_products>
15          <product>
16              <product_summary>Book - XML for
17 dummies</product_summary>
18              <product_quantity>1</product_quantity?
19          </product>
20      </purchase_summary>
21  </pay_request>

```

22 **[0093]** In some implementations, the issuer server may generate a payment
23 command, e.g., 658. For example, the issuer server may issue a command to deduct
24 funds from the user's account (or add a charge to the user's credit card account). The
25 issuer server may issue a payment command, e.g., 659, to a database storing the user's
26 account information, e.g., user profile database 608. The issuer server may provide a
27 funds transfer message, e.g., 660, to the pay network server, which may forward, e.g.,
28 661, the funds transfer message to the acquirer server. An example HTTP(S) POST
29 funds transfer message is provided below:

```

30 POST /clearance.php HTTP/1.1
31 Host: www.acquirer.com
32 Content-Type: Application/XML
33 Content-Length: 206
34 <?XML version = "1.0" encoding = "UTF-8"?>

```

```

1   <deposit_ack>
2       <request_ID>CNI4ICNW2</request_ID>
3       <clear_flag>true</clear_flag>
4       <timestamp>2011-02-22 17:00:02</timestamp>
5       <deposit_amount>$34.78</deposit_amount>
6   </deposit_ack>

```

7 **[0094]** In some implementations, the acquirer server may parse the funds
8 transfer message, and correlate the transaction (e.g., using the request_ID field in the
9 example above) to the merchant. The acquirer server may then transfer the funds
10 specified in the funds transfer message to an account of the merchant, e.g., 662.

11 **[0095]** FIGURES 7A-E show logic flow diagrams illustrating example aspects of
12 card-based transaction execution, resulting in generation of card-based transaction data
13 and service usage data, in some embodiments of the MMCDB, e.g., a Card-Based
14 Transaction Execution (“CTE”) component 700. In some implementations, a user may
15 provide user input, e.g., 701, into a client indicating the user’s desire to purchase a
16 product from a merchant. The client may generate a purchase order message, e.g., 702,
17 and provide the generated purchase order message to the merchant server. In some
18 implementations, the merchant server may obtain, e.g., 703, the purchase order
19 message from the client, and may parse the purchase order message to extract details of
20 the purchase order from the user. Example parsers that the merchant client may utilize
21 are discussed further below with reference to FIGURE 49. The merchant may generate a
22 product data query, e.g., 704, for a merchant database, which may in response provide
23 the requested product data, e.g., 705. The merchant server may generate a card query
24 request using the product data, e.g., 704, to determine whether the transaction can be
25 processed. For example, the merchant server may process the transaction only if the
26 user has sufficient funds to pay for the purchase in a card account provided with the
27 purchase order. The merchant server may optionally provide the generated card query
28 request to an acquirer server. The acquirer server may generate a card authorization
29 request using the obtained card query request, and provide the card authorization
30 request to a pay network server.

1 **[0096]** In some implementations, the pay network server may determine whether
 2 the user has enrolled in value-added user services. For example, the pay network server
 3 may query a database, e.g., 707, for user service enrollment data. For example, the
 4 server may utilize PHP/SQL commands similar to the example provided above to query
 5 the pay network database. In some implementations, the database may provide the user
 6 service enrollment data, e.g., 708. The user enrollment data may include a flag
 7 indicating whether the user is enrolled or not, as well as instructions, data, login URL,
 8 login API call template and/or the like for facilitating access of the user-enrolled
 9 services. For example, in some implementations, the pay network server may redirect
 10 the client to a value-add server (e.g., such as a social network server where the value-add
 11 service is related to social networking) by providing a HTTP(S) REDIRECT 300
 12 message. In some implementations, the pay network server may provide payment
 13 information extracted from the card authorization request to the value-add server as
 14 part of a value add service request, e.g., 710.

15 **[0097]** In some implementations, the value-add server may provide a service
 16 input request, e.g., 711, to the client. The client may display, e.g., 712, the input request
 17 for the user. In some implementations, the user may provide input into the client, e.g.,
 18 713, and the client may generate a service input response for the value-add server. In
 19 some implementations, the value-add server may provide value-add services according
 20 to user value-add service enrollment data, user profile, etc., stored on the value-add
 21 server, and based on the user service input. Based on the provision of value-add
 22 services, the value-add server may generate a value-add service response, e.g., 717, and
 23 provide the response to the pay network server. In some implementations, upon
 24 receiving the value-add service response from the value-add server, the pay network
 25 server may extract the enrollment service data from the response for addition to a
 26 transaction data record, e.g., 719-720.

27 **[0098]** With reference to FIGURE 7B, in some implementations, the pay network
 28 server may obtain the card authorization request from the acquirer server, and may
 29 parse the card authorization request to extract details of the request, e.g., 720. Using
 30 the extracted fields and field values, the pay network server may generate a query, e.g.,
 31 721-722, for an issuer server corresponding to the user's card account. In response to

1 obtaining the issuer server query the pay network database may provide, e.g., 722, the
 2 requested issuer server data to the pay network server. In some implementations, the
 3 pay network server may utilize the issuer server data to generate a forwarding card
 4 authorization request, e.g., 723, to redirect the card authorization request from the
 5 acquirer server to the issuer server. The pay network server may provide the card
 6 authorization request to the issuer server. In some implementations, the issuer server
 7 may parse, e.g., 724, the card authorization request, and based on the request details
 8 may query a database, e.g., 725, for data of the user's card account. In response, the
 9 database may provide the requested user data. On obtaining the user data, the issuer
 10 server may determine whether the user can pay for the transaction using funds available
 11 in the account, e.g., 726. For example, the issuer server may determine whether the user
 12 has a sufficient balance remaining in the account, sufficient credit associated with the
 13 account, and/or the like, but comparing the data from the database with the transaction
 14 cost obtained from the card authorization request. If the issuer server determines that
 15 the user can pay for the transaction using the funds available in the account, the server
 16 may provide an authorization message, e.g., 727, to the pay network server.

17 **[0099]** In some implementations, the pay network server may obtain the
 18 authorization message, and parse the message to extract authorization details. Upon
 19 determining that the user possesses sufficient funds for the transaction (e.g., 730, option
 20 "Yes"), the pay network server may extract the transaction card from the authorization
 21 message and/or card authorization request, e.g., 733, and generate a transaction data
 22 record using the card transaction details. The pay network server may provide the
 23 transaction data record for storage, e.g., 734, to a database. In some implementations,
 24 the pay network server may forward the authorization message, e.g., 735, to the acquirer
 25 server, which may in turn forward the authorization message, e.g., 736, to the merchant
 26 server. The merchant may obtain the authorization message, and parse the
 27 authorization message o extract its contents, e.g., 737. The merchant server may
 28 determine whether the user possesses sufficient funds in the card account to conduct the
 29 transaction. If the merchant server determines that the user possess sufficient funds,
 30 e.g., 738, option "Yes," the merchant server may add the record of the transaction for the
 31 user to a batch of transaction data relating to authorized transactions, e.g., 739-740.

1 The merchant server may also generate a purchase receipt, e.g., 741, for the user. If the
 2 merchant server determines that the user does not possess sufficient funds, e.g., 738,
 3 option “No,” the merchant server may generate an “authorization fail” message, e.g.,
 4 742. The merchant server may provide the purchase receipt or the “authorization fail”
 5 message to the client. The client may render and display, e.g., 743, the purchase receipt
 6 for the user.

7 **[00100]** In some implementations, the merchant server may initiate clearance of a
 8 batch of authorized transactions by generating a batch data request, e.g., 744, and
 9 providing the request to a database. In response to the batch data request, the database
 10 may provide the requested batch data, e.g., 745, to the merchant server. The server may
 11 generate a batch clearance request, e.g., 746, using the batch data obtained from the
 12 database, and provide the batch clearance request to an acquirer server. The acquirer
 13 server may generate, e.g., 748, a batch payment request using the obtained batch
 14 clearance request, and provide the batch payment request to a pay network server. The
 15 pay network server may parse, e.g., 749, the batch payment request, select a transaction
 16 stored within the batch data, e.g., 750, and extract the transaction data for the
 17 transaction stored in the batch payment request, e.g., 751. The pay network server may
 18 generate a transaction data record, e.g., 752, and store the transaction data, e.g., 753, the
 19 transaction in a database. For the extracted transaction, the pay network server may
 20 generate an issuer server query, e.g., 754, for an address of an issuer server maintaining
 21 the account of the user requesting the transaction. The pay network server may provide
 22 the query to a database. In response, the database may provide the issuer server data
 23 requested by the pay network server, e.g., 755. The pay network server may generate an
 24 individual payment request, e.g., 756, for the transaction for which it has extracted
 25 transaction data, and provide the individual payment request to the issuer server using
 26 the issuer server data from the database.

27 **[00101]** In some implementations, the issuer server may obtain the individual
 28 payment request, and parse, e.g., 757, the individual payment request to extract details
 29 of the request. Based on the extracted data, the issuer server may generate a payment
 30 command, e.g., 758. For example, the issuer server may issue a command to deduct
 31 funds from the user’s account (or add a charge to the user’s credit card account). The

1 issuer server may issue a payment command, e.g., 759, to a database storing the user's
 2 account information. In response, the database may update a data record
 3 corresponding to the user's account to reflect the debit / charge made to the user's
 4 account. The issuer server may provide a funds transfer message, e.g., 760, to the pay
 5 network server after the payment command has been executed by the database.

6 **[00102]** In some implementations, the pay network server may check whether
 7 there are additional transactions in the batch that need to be cleared and funded. If
 8 there are additional transactions, e.g., 761, option "Yes," the pay network server may
 9 process each transaction according to the procedure described above. The pay network
 10 server may generate, e.g., 762, an aggregated funds transfer message reflecting transfer
 11 of all transactions in the batch, and provide, e.g., 763, the funds transfer message to the
 12 acquirer server. The acquirer server may, in response, transfer the funds specified in the
 13 funds transfer message to an account of the merchant, e.g., 764.

14 **[00103]** FIGURE 8 shows a data flow diagram illustrating an example procedure to
 15 aggregate card-based transaction data in some embodiments of the MMCDB. In some
 16 implementations, the pay network server may determine a scope of data aggregation
 17 required to perform the analysis, e.g., 811. The pay network server may initiate data
 18 aggregation based on the determined scope. The pay network server may generate a
 19 query for addresses of server storing transaction data within the determined scope. The
 20 pay network server may query, e.g., 812, a pay network database, e.g., 807a, for
 21 addresses of pay network servers that may have stored transaction data within the
 22 determined scope of the data aggregation. For example, the pay network server may
 23 utilize PHP/SQL commands similar to the examples provided above. The database may
 24 provide, e.g., 813, a list of server addresses in response to the pay network server's
 25 query. Based on the list of server addresses, the pay network server may generate
 26 transaction data requests, e.g., 814. The pay network server may issue the generated
 27 transaction data requests, e.g., 815a-c, to the other pay network servers, e.g., 805b-d.
 28 The other pay network servers may query, e.g., 817a-c, their pay network database, e.g.,
 29 807a-d, for transaction data falling within the scope of the transaction data requests. In
 30 response to the transaction data queries, the pay network databases may provide
 31 transaction data, e.g., 818a-c, to the other pay network servers. The other pay network

1 servers may return the transaction data obtained from the pay network databases, e.g.,
 2 819a-c, to the pay network server making the transaction data requests, e.g., 805a. The
 3 pay network server, e.g., 805a, may store the aggregated transaction data, e.g., 820, in
 4 an aggregated transactions database, e.g., 810a.

5 **[00104]** FIGURE 9 shows a logic flow diagram illustrating example aspects of
 6 aggregating card-based transaction data in some embodiments of the MMCDB, e.g., a
 7 Transaction Data Aggregation (“TDA”) component 900. In some implementations, a
 8 pay network server may obtain a trigger to aggregate transaction data, e.g., 901. For
 9 example, the server may be configured to initiate transaction data aggregation on a
 10 regular, periodic, basis (e.g., hourly, daily, weekly, monthly, quarterly, semi-annually,
 11 annually, etc.). As another example, the server may be configured to initiate transaction
 12 data aggregation on obtaining information that the U.S. Government (e.g., Department
 13 of Commerce, Office of Management and Budget, etc) has released new statistical data
 14 related to the U.S. business economy. As another example, the server may be
 15 configured to initiate transaction data aggregation on-demand, upon obtaining a user
 16 investment strategy analysis request for processing. The pay network server may
 17 determine a scope of data aggregation required to perform the analysis, e.g., 902. For
 18 example, the scope of data aggregation may be pre-determined. As another example,
 19 the scope of data aggregation may be determined based on a received user investment
 20 strategy analysis request. The pay network server may initiate data aggregation based
 21 on the determined scope. The pay network server may generate a query for addresses of
 22 server storing transaction data within the determined scope, e.g., 903. The pay network
 23 server may query a database for addresses of pay network servers that may have stored
 24 transaction data within the determined scope of the data aggregation. The database
 25 may provide, e.g., 904, a list of server addresses in response to the pay network server’s
 26 query. Based on the list of server addresses, the pay network server may generate
 27 transaction data requests, e.g., 905. The pay network server may issue the generated
 28 transaction data requests to the other pay network servers. The other pay network
 29 servers may obtain and parse the transaction data requests, e.g., 906. Based on parsing
 30 the data requests, the other pay network servers may generate transaction data queries,
 31 e.g., 907, and provide the transaction data queries to their pay network databases. In

1 response to the transaction data queries, the pay network databases may provide
 2 transaction data, e.g., 908, to the other pay network servers. The other pay network
 3 servers may return, e.g., 909, the transaction data obtained from the pay network
 4 databases to the pay network server making the transaction data requests. The pay
 5 network server may generate aggregated transaction data records from the transaction
 6 data received from the other pay network servers, e.g., 910, and store the aggregated
 7 transaction data in a database, e.g., 911.

8 **[00105]** FIGURE 10 shows a data flow diagram illustrating an example social data
 9 aggregation procedure in some embodiments of the MMCDB. In some
 10 implementations, the pay network server may obtain a trigger to perform a social data
 11 search. For example, the pay network server may periodically perform an update of its
 12 aggregated social database, e.g., 1010, with new information available from a variety of
 13 sources, such as the social networking services operating on the Internet. As another
 14 example, a request for on-demand social data update may be obtained as a result of a
 15 user wishing to enroll in a service, for which the pay network server may facilitate data
 16 entry by providing an automated web form filling system using information about the
 17 user obtained from the social data update. In some implementations, the pay network
 18 server may parse the trigger to extract keywords using which to perform an aggregated
 19 social data update. The pay network server may generate a query for application
 20 programming interface (API) templates for various social networking services (e.g.,
 21 Facebook®, Twitter™, etc.) from which to collect social data for aggregation. The pay
 22 network server may query, e.g., 1012, a pay network database, e.g., 1007, for social
 23 network API templates for the social networking services. For example, the pay network
 24 server may utilize PHP/SQL commands similar to the examples provided above. The
 25 database may provide, e.g., 1013, a list of API templates in response. Based on the list of
 26 API templates, the pay network server may generate social data requests, e.g., 1014. The
 27 pay network server may issue the generated social data requests, e.g., 1015a-c, to the
 28 social network servers, e.g., 1001a-c. For example, the pay network server may issue
 29 PHP commands to request the social network servers for social data. An example listing
 30 of commands to issue social data requests 1015a-c, substantially in the form of PHP
 31 commands, is provided below:

```

1  <?PHP
2  header('Content-Type: text/plain');
3
4  // Obtain user ID(s) of friends of the logged-in user
5  $friends =
6  json_decode(file_get_contents('https://graph.facebook.com/me/frie
7  nds?access_token='.$cookie['oauth_access_token']), true);
8  $friend_ids = array_keys($friends);
9
10 // Obtain message feed associated with the profile of the logged-
11 in user
12 $feed =
13 json_decode(file_get_contents('https://graph.facebook.com/me/feed
14 ?access_token='.$cookie['oauth_access_token']), true);
15
16 // Obtain messages by the user's friends
17 $result = mysql_query('SELECT * FROM content WHERE uid IN ( '
18 .implode($friend_ids, ',') . ')');
19 $friend_content = array();
20 while ($row = mysql_fetch_assoc($result))
21 $friend_content [] $row;
22 ?>

```

23 **[00106]** In some embodiments, the social network servers may query, e.g., 1017a-c,
24 their databases, e.g., 1002a-c, for social data results falling within the scope of the social
25 keywords. In response to the queries, the databases may provide social data, e.g.,
26 1018a-c, to the search engine servers. The social network servers may return the social
27 data obtained from the databases, e.g., 1019a-c, to the pay network server making the
28 social data requests. An example listing of social data 1019a-c, substantially in the form
29 of JavaScript Object Notation (JSON)-formatted data, is provided below:

```

30
31 [ "data":      [
32                 {      "name": "Tabatha Orloff",
33                       "id": "483722"},
34                 {      "name": "Darren Kinnaman",

```

```

1         "id": "86S743"},
2     {     "name": "Sharron Jutras",
3         "id": "O91274"}
4 ] }
```

5 **[00107]** In some embodiments, the pay network server may store the aggregated
6 search results, e.g., 1020, in an aggregated search database, e.g., 1010a.

7 **[00108]** FIGURE 11 shows a logic flow diagram illustrating example aspects of
8 aggregating social data in some embodiments of the MMCDB, e.g., a Social Data
9 Aggregation (“SDA”) component 1100. In some implementations, the pay network
10 server may obtain a trigger to perform a social search, e.g., 1101. For example, the pay
11 network server may periodically perform an update of its aggregated social database
12 with new information available from a variety of sources, such as the Internet. As
13 another example, a request for on-demand social data update may be obtained as a
14 result of a user wishing to enroll in a service, for which the pay network server may
15 facilitate data entry by providing an automated web form filling system using
16 information about the user obtained from the social data update. In some
17 implementations, the pay network server may parse the trigger, e.g., 1102, to extract
18 keywords and/or user ID(s) using which to perform an aggregated search for social data.
19 The pay network server may determine the social networking services to search, e.g.,
20 1103, using the extracted keywords and/or user ID(s). Then, the pay network server
21 may generate a query for application programming interface (API) templates for the
22 various social networking services (e.g., Facebook®, Twitter™, etc.) from which to
23 collect social data for aggregation, e.g., 1104. The pay network server may query, e.g.,
24 1105, a pay network database for search API templates for the social networking
25 services. For example, the pay network server may utilize PHP/SQL commands similar
26 to the examples provided above. The database may provide, e.g., 1105, a list of API
27 templates in response. Based on the list of API templates, the pay network server may
28 generate social data requests, e.g., 1106. The pay network server may issue the
29 generated social data requests to the social networking services. The social network
30 servers may parse the obtained search results(s), e.g., 1107, and query, e.g., 1108, their
31 databases for social data falling within the scope of the search keywords. In response to

1 the social data queries, the databases may provide social data, e.g., 1109, to the social
 2 networking servers. The social networking servers may return the social data obtained
 3 from the databases, e.g., 1110, to the pay network server making the social data requests.
 4 The pay network server may generate, e.g., 1111, and store the aggregated social data,
 5 e.g., 1112, in an aggregated social database.

6 **[00109]** FIGURE 12 shows a data flow diagram illustrating an example procedure
 7 for enrollment in value-add services in some embodiments of the MMCDB. In some
 8 implementations, a user, e.g., 1201, may desire to enroll in a value-added service. Let us
 9 consider an example wherein the user desires to enroll in social network authenticated
 10 purchase payment as a value-added service. It is to be understood that any other value-
 11 added service may take the place of the below-described value-added service. The user
 12 may communicate with a pay network server, e.g., 1203, via a client such as, but not
 13 limited to: a personal computer, mobile device, television, point-of-sale terminal, kiosk,
 14 ATM, and/or the like (e.g., 1202). For example, the user may provide user input, e.g.,
 15 enroll input 1211, into the client indicating the user's desire to enroll in social network
 16 authenticated purchase payment. In various implementations, the user input may
 17 include, but not be limited to: a single tap (e.g., a one-tap mobile app purchasing
 18 embodiment) of a touchscreen interface, keyboard entry, card swipe, activating a
 19 RFID/NFC enabled hardware device (e.g., electronic card having multiple accounts,
 20 smartphone, tablet, etc.) within the user device, mouse clicks, depressing buttons on a
 21 joystick/game console, voice commands, single/multi-touch gestures on a touch-
 22 sensitive interface, touching user interface elements on a touch-sensitive display, and/or
 23 the like. For example, the user may swipe a payment card at the client 1202. In some
 24 implementations, the client may obtain track 1 data from the user's card as enroll input
 25 1211 (e.g., credit card, debit card, prepaid card, charge card, etc.), such as the example
 26 track 1 data provided below:

```
27 %B123456789012345^PUBLIC/J.Q.^9901120000000000000000**901*****?*
28 (wherein '123456789012345' is the card number of 'J.Q. Public'
29 and has a CVV number of 901. '990112' is a service code, and ***
30 represents decimal digits which change randomly each time the
31 card is used.)
```

1 **[00110]** In some implementations, using the user's input, the client may generate
 2 an enrollment request, e.g., 1212, and provide the enrollment request, e.g., 1213, to the
 3 pay network server. For example, the client may provide a (Secure) Hypertext Transfer
 4 Protocol ("HTTP(S)") POST message including data formatted according to the
 5 eXtensible Markup Language ("XML"). Below is an example HTTP(S) POST message
 6 including an XML-formatted enrollment request for the pay network server:

```

7  POST /enroll.php HTTP/1.1
8  Host: www.merchant.com
9  Content-Type: Application/XML
10 Content-Length: 718
11 <?XML version = "1.0" encoding = "UTF-8"?>
12 <enrollment_request>
13     <cart_ID>4NFU4RG94</order_ID>
14     <timestamp>2011-02-22 15:22:43</timestamp>
15     <user_ID>john.q.public@gmail.com</user_ID>
16     <client_details>
17         <client_IP>192.168.23.126</client_IP>
18         <client_type>smartphone</client_type>
19         <client_model>HTC Hero</client_model>
20         <OS>Android 2.2</OS>
21         <app_installed_flag>true</app_installed_flag>
22     </client_details>
23     <!--account_params> <optional>
24         <account_name>John Q. Public</account_name>
25         <account_type>credit</account_type>
26         <account_num>123456789012345</account_num>
27         <billing_address>123 Green St., Norman, OK
28 98765</billing_address>
29         <phone>123-456-7809</phone>
30         <sign>/jqp</sign>
31         <confirm_type>email</confirm_type>
32
33     <contact_info>john.q.public@gmail.com</contact_info>
34 </account_params-->
35 <checkout_purchase_details>
36     <num_products>1</num_products>

```

```

1          <product>
2              <product_type>book</product_type>
3              <product_params>
4                  <product_title>XML for
5  dummies</product_title>
6                  <ISBN>938-2-14-168710-0</ISBN>
7                  <edition>2nd ed.</edition>
8                  <cover>hardbound</cover>
9                  <seller>bestbuybooks</seller>
10             </product_params>
11             <quantity>1</quantity>
12         </product>
13     </checkout_purchase_details>
14 </enrollment_request>

```

15 **[00111]** In some implementations, the pay network server may obtain the
16 enrollment request from the client, and extract the user's payment detail (e.g., XML
17 data) from the enrollment request. For example, the pay network server may utilize a
18 parser such as the example parsers described below in the discussion with reference to
19 FIGURE 49. In some implementations, the pay network server may query, e.g., 1214, a
20 pay network database, e.g., 1204, to obtain a social network request template, e.g., 1215,
21 to process the enrollment request. The social network request template may include
22 instructions, data, login URL, login API call template and/or the like for facilitating
23 social network authentication. For example, the database may be a relational database
24 responsive to Structured Query Language ("SQL") commands. The merchant server
25 may execute a hypertext preprocessor ("PHP") script including SQL commands to query
26 the database for product data. An example PHP/SQL command listing, illustrating
27 substantive aspects of querying the database, e.g., 1214-1215, is provided below:

```

28 <?PHP
29 header('Content-Type: text/plain');
30 mysql_connect("254.93.179.112", $DBserver, $password); // access
31 database server
32 mysql_select_db("SOCIALAUTH.SQL"); // select database table to
33 search

```

```

1  //create query
2  $query = "SELECT template FROM EnrollTable WHERE network LIKE '%"
3  $socialnet";
4  $result = mysql_query($query); // perform the search query
5  mysql_close("SOCIALAUTH.SQL"); // close database access
6  ?>

```

7 **[00112]** In some implementations, the pay network server may redirect the client
8 to a social network server by providing a HTTP(S) REDIRECT 300 message, similar to
9 the example below:

```

10 HTTP/1.1 300 Multiple Choices
11 Location:
12 https://www.facebook.com/dialog/oauth?client_id=snpa_app_ID&redir
13 ect_uri= www.paynetwork.com/enroll.php
14 <html>
15     <head><title>300 Multiple Choices</title></head>
16     <body><h1>Multiple Choices</h1></body>
17 </html>

```

18 **[00113]** In some implementations, the pay network server may provide payment
19 information extracted from the card authorization request to the social network server
20 as part of a social network authentication enrollment request, e.g., 1217. For example,
21 the pay network server may provide a HTTP(S) POST message to the social network
22 server, similar to the example below:

```

23 POST /authenticate_enroll.php HTTP/1.1
24 Host: www.socialnet.com
25 Content-Type: Application/XML
26 Content-Length: 1306
27 <?XML version = "1.0" encoding = "UTF-8"?>
28 <authenticate_enrollment_request>
29     <request_ID>4NFU4RG94</order_ID>
30     <timestamp>2011-02-22 15:22:43</timestamp>
31     <user_ID>john.q.public@gmail.com</user_ID>
32     <client_details>
33         <client_IP>192.168.23.126</client_IP>

```

```

1         <client_type>smartphone</client_type>
2         <client_model>HTC Hero</client_model>
3         <OS>Android 2.2</OS>
4         <app_installed_flag>true</app_installed_flag>
5     </client_details>
6     <account_params>
7         <account_name>John Q. Public</account_name>
8         <account_type>credit</account_type>
9         <account_num>123456789012345</account_num>
10        <billing_address>123 Green St., Norman, OK
11 98765</billing_address>
12        <phone>123-456-7809</phone>
13        <sign>/jqp</sign>
14        <confirm_type>email</confirm_type>
15        <contact_info>john.q.public@gmail.com</contact_info>
16    </account_params>
17 </authenticate_enrollment_request>

```

18 **[00114]** In some implementations, the social network server may provide a social
19 network login request, e.g., 1218, to the client. For example, the social network server
20 may provide a HTML input form to the client. The client may display, e.g., 1219, the
21 login form for the user. In some implementations, the user may provide login input into
22 the client, e.g., 1220, and the client may generate a social network login response, e.g.,
23 1221, for the social network server. In some implementations, the social network server
24 may authenticate the login credentials of the user, and access payment account
25 information of the user stored within the social network, e.g., in a social network
26 database. Upon authentication, the social network server may generate an
27 authentication data record for the user, e.g., 1222, and provide an enrollment
28 notification, e.g., 1224, to the pay network server. For example, the social network
29 server may provide a HTTP(S) POST message similar to the example below:

```

30 POST /enrollnotification.php HTTP/1.1
31 Host: www.paynet.com
32 Content-Type: Application/XML
33 Content-Length: 1306

```

```

1  <?XML version = "1.0" encoding = "UTF-8"?>
2  <enroll_notification>
3      <request_ID>4NFU4RG94</order_ID>
4      <timestamp>2011-02-22 15:22:43</timestamp>
5      <result>enrolled</result>
6  </enroll_notification>

```

7 **[00115]** Upon receiving notification of enrollment from the social network server,
8 the pay network server may generate, e.g., 1225, a user enrollment data record, and
9 store the enrollment data record in a pay network database, e.g., 1226, to complete
10 enrollment. In some implementations, the enrollment data record may include the
11 information from the enrollment notification 1224.

12 **[00116]** FIGURE 13 shows a logic flow diagram illustrating example aspects of
13 enrollment in a value-added service in some embodiments of the MMCDB, e.g., a Value-
14 Add Service Enrollment (“VASE”) component 1300. In some implementations, a user,
15 e.g., 1201, may desire to enroll in a value-added service. Let us consider an example
16 wherein the user desires to enroll in social network authenticated purchase payment as
17 a value-added service. It is to be understood that any other value-added service may
18 take the place of the below-described value-added service. The user may communicate
19 with a pay network server via a client. For example, the user may provide user input,
20 e.g., 1301, into the client indicating the user’s desire to enroll in social network
21 authenticated purchase payment. In various implementations, the user input may
22 include, but not be limited to: a single tap (e.g., a one-tap mobile app purchasing
23 embodiment) of a touchscreen interface, keyboard entry, card swipe, activating a
24 RFID/NFC enabled hardware device (e.g., electronic card having multiple accounts,
25 smartphone, tablet, etc.) within the user device, mouse clicks, depressing buttons on a
26 joystick/game console, voice commands, single/multi-touch gestures on a touch-
27 sensitive interface, touching user interface elements on a touch-sensitive display, and/or
28 the like. In some implementations, using the user’s input, the client may generate an
29 enrollment request, e.g., 1302, and provide the enrollment request to the pay network
30 server. In some implementations, the MMCDB may provide an enrollment button
31 which may take the user to an enrollment webpage where account info may be entered

1 into web form fields. In some implementations, the pay network server may obtain the
 2 enrollment request from the client, and extract the user's payment detail from the
 3 enrollment request. For example, the pay network server may utilize a parser such as
 4 the example parsers described below in the discussion with reference to FIGURE 49. In
 5 some implementations, the pay network server may query, e.g., 1304, a pay network
 6 database to obtain a social network request template, e.g., 1305, to process the
 7 enrollment request. The social network request template may include instructions,
 8 data, login URL, login API call template and/or the like for facilitating social network
 9 authentication. In some implementations, the pay network server may provide payment
 10 information extracted from the card authorization request to the social network server
 11 as part of a social network authentication enrollment request, e.g., 1306. In some
 12 implementations, the social network server may provide a social network login request,
 13 e.g., 1307, to the client. For example, the social network server may provide a HTML
 14 input form to the client. The client may display, e.g., 1308, the login form for the user.
 15 In some implementations, the user may provide login input into the client, e.g., 1309,
 16 and the client may generate a social network login response for the social network
 17 server. In some implementations, the social network server may authenticate the login
 18 credentials of the user, and access payment account information of the user stored
 19 within the social network, e.g., in a social network database. Upon authentication, the
 20 social network server may generate an authentication data record for the user, e.g., 1311,
 21 and provide an enrollment notification to the pay network server, e.g., 1313. Upon
 22 receiving notification of enrollment from the social network server, the pay network
 23 server may generate, e.g., 1314, a user enrollment data record, and store the enrollment
 24 data record in a pay network database, e.g., 1315, to complete enrollment. The pay
 25 network server may provide an enrollment confirmation, and provide the enrollment
 26 confirmation to the client, which may display, e.g., 1317, the confirmation for the user.

27 **[00117]** FIGURES 14A-B show flow diagrams illustrating example aspects of
 28 normalizing aggregated search, enrolled, service usage, transaction and/or other
 29 aggregated data into a standardized data format in some embodiments of the MMCDB,
 30 e.g., a Aggregated Data Record Normalization ("ADRN") component 1400. With
 31 reference to FIGURE 14A, in some implementations, a pay network server ("server")

1 may attempt to convert any aggregated data records stored in an aggregated records
 2 database it has access to in a normalized data format. For example, the database may
 3 have a transaction data record template with predetermined, standard fields that may
 4 store data in pre-defined formats (e.g., long integer / double float / 4 digits of precision,
 5 etc.) in a pre-determined data structure. A sample XML transaction data record
 6 template is provided below:

```

7   <?XML version = "1.0" encoding = "UTF-8"?>
8   <transaction_record>
9       <record_ID>00000000</record_ID>
10      <norm_flag>>false</norm_flag>
11      <timestamp>yyyy-mm-dd hh:mm:ss</timestamp>
12      <transaction_cost>$0,000,000,00</transaction_cost>
13      <merchant_params>
14          <merchant_id>00000000</merchant_id>
15          <merchant_name>TBD</merchant_name>
16
17          <merchant_auth_key>0000000000000000</merchant_auth_key>
18      </merchant_params>
19      <merchant_products>
20          <num_products>000</num_products>
21          <product>
22              <product_type>TBD</product_type>
23              <product_name>TBD</product_name>
24              <class_labels_list>TBD<class_labels_list>
25              <product_quantity>000</product_quantity>
26              <unit_value>$0,000,000.00</unit_value>
27              <sub_total>$0,000,000.00</sub_total>
28              <comment>normalized transaction data record
29  template</comment>
30          </product>
31      </merchant_products>
32      <user_account_params>
33          <account_name>JTBD</account_name>
34          <account_type>TBD</account_type>
35          <account_num>0000000000000000</account_num>
36          <billing_line1>TBD</billing_line1>

```

```

1         <billing_line2>TBD</billing_line2>
2         <zipcode>TBD</zipcode>
3         <state>TBD</state>
4         <country>TBD</country>
5         <phone>00-00-000-000-0000</phone>
6         <sign>TBD</sign>
7     </user_account_params>
8 </transaction_record>

```

9 **[00118]** In other embodiments, the transaction data record template may contain
10 integrated logic, regular expressions, executable meta-commands, language commands
11 and/or the like in order to facilitate properly matching aggregated data with the location
12 and format of the data in the template. In some embodiments, the template may
13 contain logic in a non-template language, such as PHP commands being included in an
14 XML file. As such, in one example, a language key may be used by the template (e.g.,
15 “php:<command>”, “java:<function>”, and/or the like). In so doing, the matching
16 template may match a vast array of disparate data formats down into a normalized and
17 standardized format. An example transaction data template record substantially in the
18 form of XML is as follows:

```

19 <?XML version = "1.0" encoding = "UTF-8"?>
20 <transaction_record>
21     <record_ID
22         default_value=false_return_error
23         match_length=8
24         format=integer
25         regex_search="(?!=\s|^)\d+(?!=\s|$)"
26         start_search_offset="50bytes">00000000</record_ID>
27 <norm_flag>false</norm_flag>
28 <timestamp
29     default_value="MySQL:'NOW()'"
30     format_after_matching="php:mktime($value);">
31     yyyy-mm-dd hh:mm:ss</timestamp>
32 <transaction_cost>$0,000,000,00</transaction_cost>
33 <merchant_params>
34     <merch_id>00000000</merch_id>

```

```

1         <merch_name>TBD</merch_name>
2         <merch_auth_key>0000000000000000</merch_auth_key>
3     </merchant_params>
4     <merchant_products>
5         <num_products
6             min_quantity=1
7             max_quantity=30>000</num_products>
8         <product>
9             <product_type
10                 from_group="array( 'BOOK' , ' CD' , ' DVD' ) ">
11                 TBD</product_type>
12             <product_name>TBD</product_name>
13             <class_labels_list>TBD<class_labels_list>
14             <product_quantity>000</product_quantity>
15             <unit_value>$0,000,000.00</unit_value>
16             <sub_total>$0,000,000.00</sub_total>
17             <comment>normalized transaction data record
18 template</comment>
19         </product>
20     </merchant_products>
21     <user_account_params>
22         <account_name>JTBD</account_name>
23         <account_type>TBD</account_type>
24         <account_num>0000000000000000</account_num>
25         <billing_line1>TBD</billing_line1>
26         <billing_line2>TBD</billing_line2>
27         <zipcode>TBD</zipcode>
28         <state>TBD</state>
29         <country>TBD</country>
30         <phone>00-00-000-000-0000</phone>
31         <sign>TBD</sign>
32     </user_account_params>
33 </transaction_record>

```

34 **[00119]** In some implementations, the server may query a database for a
35 normalized data record template, e.g., 1401. The server may parse the normalized data

1 record template, e.g., 1402. In some embodiments, the parsing may parse the raw data
 2 record (such as using a parser as described herein and with respect to Fig. 49). In other
 3 embodiments, the parser may parse a dictionary entry containing a subset of the
 4 complete data. Based on parsing the normalized data record template, the server may
 5 determine the data fields included in the normalized data record template, and the
 6 format of the data stored in the fields of the data record template, e.g., 1403. The server
 7 may obtain transaction data records for normalization. The server may query a
 8 database, e.g., 1404, for non-normalized records. In one embodiment, no querying is
 9 required as the normalization of records may occur in flight (e.g., in real time as data is
 10 received). For example, the server may issue PHP/SQL commands to retrieve records
 11 that do not have the 'norm_flag' field from the example template above, or those where
 12 the value of the 'norm_flag' field is 'false'. Upon obtaining the non-normalized
 13 transaction data records, the server may select one of the non-normalized transaction
 14 data records, e.g., 1405. The server may parse the non-normalized transaction data
 15 record, e.g., 1406, and determine the fields present in the non-normalized transaction
 16 data record, e.g., 1407. For example, the server may utilize a procedure similar to one
 17 described below with reference to FIGURE 15. The server may compare the fields from
 18 the non-normalized transaction data record with the fields extracted from the
 19 normalized transaction data record template. For example, the server may determine
 20 whether the field identifiers of fields in the non-normalized transaction data record
 21 match those of the normalized transaction data record template, (e.g., via a dictionary,
 22 thesaurus, etc.), are identical, are synonymous, are related, and/or the like. Based on
 23 the comparison, the server may generate a 1:1 mapping between fields of the non-
 24 normalized transaction data record match those of the normalized transaction data
 25 record template, e.g., 1409. The server may generate a copy of the normalized
 26 transaction data record template, e.g., 1410, and populate the fields of the template
 27 using values from the non-normalized transaction data record, e.g., 1411. The server
 28 may also change the value of the 'norm_flag' field to 'true' in the example above. The
 29 server may store the populated record in a database (for example, replacing the original
 30 version), e.g., 1412. The server may repeat the above procedure for each non-
 31 normalized transaction data record (see e.g., 1413), until all the non-normalized
 32 transaction data records have been normalized.

1 **[00120]** With reference to FIGURE 14B, in some embodiments, the server may
 2 utilize metadata (e.g., easily configurable data) to drive an analytics and rule engine that
 3 may convert any structured data into a standardized XML format (“encryptmatics”
 4 XML). The encryptmatics XML may then be processed by an encryptmatics engine that
 5 is capable of parsing, transforming and analyzing data to generate decisions based on
 6 the results of the analysis. Accordingly, in some embodiments, the server may
 7 implement a metadata-based interpretation engine that parses structured data,
 8 including, but not limited to: web content (see e.g., 1421), graph databases (see e.g.,
 9 1422), micro blogs, images or software code (see e.g., 1424), and converts the structured
 10 data into commands in the encryptmatics XML file format. For example, the structured
 11 data may include, without limitation, software code, images, free text, relational
 12 database queries, graph queries, sensory inputs (see e.g., 1423, 1425), and/or the like. A
 13 metadata based interpretation engine engine, e.g., 1426, may populate a data/command
 14 object, e.g., 1427, based on a given record using configurable metadata, e.g., 1428. The
 15 configurable metadata may define an action for a given glyph or keyword contained
 16 within a data record. The engine may then process the object to export its data structure
 17 as a collection of encryptmatics vaults in a standard encryptmatics XML file format, e.g.,
 18 1429. The encryptmatics XML file may then be processed to provide various features by
 19 an encryptmatics engine, e.g., 1430.

20 **[00121]** In some embodiments, the server may obtain the structured data, and
 21 perform a standardization routine using the structured data as input (e.g., including
 22 script commands, for illustration). For example, the server may remove extra line
 23 breaks, spaces, tab spaces, etc. from the structured data, e.g. 1431. The server may
 24 determine and load a metadata library, e.g., 1432, using which the server may parse
 25 subroutines or functions within the script, based on the metadata, e.g., 1433-1434. In
 26 some embodiments, the server may pre-parse conditional statements based on the
 27 metadata, e.g., 1435-1436. The server may also parse data 1437 to populate a
 28 data/command object based on the metadata and prior parsing, e.g., 1438. Upon
 29 finalizing the data/command object, the server may export 1439 the data/command
 30 object as XML in standardized encryptmatics format.

1 **[00122]** FIGURE 15 shows a logic flow diagram illustrating example aspects of
 2 recognizing data fields in normalized aggregated data records in some embodiments of
 3 the MMCDB, e.g., a Data Field Recognition (“DFR”) component 1500. In some
 4 implementations, a server may recognize the type of data fields included in a data
 5 record, e.g., date, address, zipcode, name, user ID, email address, payment account
 6 number (PAN), CVV2 numbers, and/or the like. The server may select an unprocessed
 7 data record for processing, e.g., 1501. The server may parse the data record rule, and
 8 extract data fields from the data record, e.g., 1502. The server may query a database for
 9 data field templates, e.g., 1503. For example, the server may compare the format of the
 10 fields from the data record to the data record templates to identify a match between one
 11 of the data field templates and each field within the data record, thus identifying the
 12 type of each field within the data record. In one embodiment, the data field templates
 13 may be implemented as a collection of regular expressions, a set of interpreted or
 14 compiled language commands that when run against the candidate match return
 15 boolean true or false if the candidate matches, and/or the like. The server may thus
 16 select an extracted data field from the data record, e.g., 1504. The server may select a
 17 data field template for comparison with the selected data field, e.g., 1505, and compare
 18 the data field template with the selected data field, e.g., 1506, to determine whether
 19 format of extracted data field matches format of data field template, e.g., 1507. If the
 20 format of the selected extracted data field matches the format of the data field template,
 21 e.g., 1508, option “Yes,” the server may assign the type of data field template to the
 22 selected data field, e.g., 1509. If the format of the extracted data field does not match
 23 the format of the data field template, e.g., 1508, option “No,” the server may try another
 24 data field template until no more data field templates are available for comparison, see
 25 e.g., 1510. If no match is found, the server may assign “unknown” string as the type of
 26 the data field, e.g., 1511. The server may store the updated data record in the database,
 27 e.g., 1512. The server may perform such data field recognition for each data field in the
 28 data record (and also for each data record in the database), see e.g., 1513.

29 **[00123]** FIGURE 16 shows a logic flow diagram illustrating example aspects of
 30 classifying entity types in some embodiments of the MMCDB, e.g., an Entity Type
 31 Classification (“ETC”) component 1600. In some implementations, a server may apply

1 one or more classification labels to each of the data records. For example, the server
 2 may classify the data records according to entity type, according to criteria such as, but
 3 not limited to: geo-political area, number of items purchased, and/or the like. The
 4 server may obtain transactions from a database that are unclassified, e.g., 1601, and
 5 obtain rules and labels for classifying the records, e.g., 1602. For example, the database
 6 may store classification rules, such as the exemplary illustrative XML-encoded
 7 classification rule provided below:

```

8   <rule>
9       <id>PURCHASE_44_45</id>
10      <name>Number of purchasers</name>
11      <inputs>num_purchasers</inputs>
12      <operations>
13          <1>label = 'null'</1>
14          <2>IF (num_purchasers > 1) label = 'household'</2>
15      </operations>
16      <outputs>label</outputs>
17  </rule>

```

18 **[00124]** The server may select an unclassified data record for processing, e.g.,
 19 1603. The server may also select a classification rule for processing the unclassified data
 20 record, e.g., 1604. The server may parse the classification rule, and determine the
 21 inputs required for the rule, e.g., 1605. Based on parsing the classification rule, the
 22 server may parse the normalized data record template, e.g., 1606, and extract the values
 23 for the fields required to be provided as inputs to the classification rule. The server may
 24 parse the classification rule, and extract the operations to be performed on the inputs
 25 provided for the rule processing, e.g., 1607. Upon determining the operations to be
 26 performed, the server may perform the rule-specified operations on the inputs provided
 27 for the classification rule, e.g., 1608. In some implementations, the rule may provide
 28 threshold values. For example, the rule may specify that if the number of products in
 29 the transaction, total value of the transaction, average luxury rating of the products sold
 30 in the transaction, etc. may need to cross a threshold in order for the label(s) associated
 31 with the rule to be applied to the transaction data record. The server may parse the
 32 classification rule to extract any threshold values required for the rule to apply, e.g.,

1 1609. The server may compare the computed values with the rule thresholds, e.g., 1610.
 2 If the rule threshold(s) is crossed, e.g., 1611, option “Yes,” the server may apply one or
 3 more labels to the transaction data record as specified by the classification rule, e.g.,
 4 1612. For example, the server may apply a classification rule to an individual product
 5 within the transaction, and/or to the transaction as a whole. In other embodiments,
 6 the rule may specify criteria that may be present in the mesh in order to generate a new
 7 entity (e.g., to create a deduced concept or deduced entity). For example, if a given set
 8 of mesh aggregated data contain references the a keyword iPhone, a rule may specify
 9 that “iPhone” is to be created as a deduced node within the mesh. This may be done in a
 10 recursive manner, such as when the creation of the meta-concept of an “iPhone” may
 11 subsequently be combined with created meta-concepts of “iMac” and “iPod” in order to
 12 create a master deduced concept of “Apple Computer”, which is thereafter associated
 13 with “iPhone,” “iMac,” and “iPod”. In so doing, the rules may allow the mesh, given the
 14 aggregated content available as well as inputs (such as category inputs) to automatically
 15 create meta-concepts based on rules that are themselves unaware of the concepts. In
 16 one embodiment, a rule for the creation of a meta-concept, substantially in the form of
 17 XML is:

```

18 <rule id="create_deduced_concept_5" type="deduced_concept">
19   <criteria>
20     <number_keyword_references>
21       <is type="greater_than" value="50" />
22       <isnot type="greater_than" value="500" />
23     </number_keyword_references>
24   </criteria>
25   <if_criteria_met value="create_entity" />
26 </rule>

```

27 **[00125]** In the example above, a new deduced entity may be added to the mesh if
 28 the number of other entites referencing a given keyword is greater than 50 but less than
 29 500. In one embodiment, the criteria may be specified as a scalar value as shown above.
 30 In other embodiments, the criteria may reference a percentage size of the mesh
 31 references (such as greater than 5% but less than 10%). In so doing, entities may be
 32 added only when they reach a certain absolute threshold, or alternatively when they

1 reach a threshold with respect to the mesh itself. In other embodiments, the criteria
 2 may be a function (such as a Python procedure) that may be performed in order to
 3 determine if a new meta-entity should be created. In such an embodiment, the rule may
 4 take advantage of any language features available (e.g., language method/functions) as
 5 well as external data sources (such as by querying Wikipedia for the presence of a page
 6 describing the candidate meta-concept, performing a Google Search and only creating
 7 the meta concept if greater than a given number of results are returned, and/or the like).
 8 In one embodiment, deduced entries may be created based on a specified or relative
 9 frequency of occurrence matches (e.g., keyword matches, transaction occurrences, and/or
 10 the like) within a certain time quantum (e.g., 5 orders for an item within a
 11 day/week/month, 100 tweets a minute about a topic, and/or the like). Deduced entities
 12 may become actual mesh entities (and actual mesh entities may be come deduced
 13 entities) through the application of similar rules. For example, if an entity is deduced
 14 but subsequently the data aggregation shows a sufficient social media discussion
 15 regarding a deduced concept, the concept may be changed from a deduced concept to a
 16 mesh concept. In so doing, the mesh can adapt to evolving entities that may initially
 17 exist only by virtue of their relationship to other nodes, but may ultimately become
 18 concepts that the mesh may assign to actual entities.

19 **[00126]** In some implementations, the server may process the transaction data
 20 record using each rule (see, e.g., 1613). Once all classification rules have been processed
 21 for the transaction record, e.g., 1613, option “No,” the server may store the transaction
 22 data record in a database, e.g., 1614. The server may perform such processing for each
 23 transaction data record until all transaction data records have been classified (see, e.g.,
 24 1615).

25 **[00127]** FIGURE 17 shows a logic flow diagram illustrating example aspects of
 26 identifying cross-entity correlation in some embodiments of the MMCDB, e.g., a Cross-
 27 Entity Correlation (“CEC”) component 1700. In some implementations, a server may
 28 recognize that two entities in the the MMCDB share common or related data fields, e.g.,
 29 date, address, zipcode, name, user ID, email address, payment account number (PAN),
 30 CVV2 numbers, and/or the like, and thus identify the entities as being correlated. The
 31 server may select a data record for cross-entity correlation, e.g., 1701. The server may

1 parse the data record rule, and extract data fields from the data record, e.g., 1702-1703.
 2 The server may select an extracted data field from the data record, e.g., 1704, and query
 3 a database for other data records having the same data field as the extracted data field,
 4 e.g., 1705. From the list of retrieved data records from the database query, the server
 5 may select a record for further analysis. The server may identify, e.g., 1707, an entity
 6 associated with the retrieved data record, e.g., using the ETC 1600 component discussed
 7 above in the description with reference to FIGURE 16. The server may add a data field
 8 to the data record obtained for cross-entity correlation specifying the correlation to the
 9 retrieved selected data record, e.g., 1708. In some embodiments, the server may utilize
 10 each data field in the data record obtained for cross-entity correlation to identify
 11 correlated entities, see e.g., 1709. The server may add, once complete, a “correlated” flag
 12 to the data record obtained for cross-entity correlation, e.g., 1710, e.g., along with as
 13 timestamp specifying the time at which the cross-entity correlation was performed. For
 14 example, such a timestamp may be used to determine at a later time whether the data
 15 record should be processed again for cross-entity correlation. The server may store the
 16 updated data record in a database.

17 **[00128]** FIGURE 18 shows a logic flow diagram illustrating example aspects of
 18 associating attributes to entities in some embodiments of the MMCDB, e.g., an Entity
 19 Attribute Association (“EAA”) component 1800. In some implementations, a server
 20 may associate attributes to an entity, e.g., if the entity id a person, the server may
 21 identify a demographic (e.g., male/female), a spend character, a purchase preferences
 22 list, a merchants preference list, and/or the like, based on field values of data fields in
 23 data records that are related to the entity. In some implementations, a server may
 24 obtain a data record for entity attribute association, e.g., 1801. The server may parse the
 25 data record rule, and extract data fields from the data record, e.g., 1802-1803. The
 26 server may select an extracted data field from the data record, e.g., 1804, and identify a
 27 field value for the selected extracted data field from the data record, e.g., 1805. The
 28 server may query a database for demographic data, behavioral data, and/or the like, e.g.,
 29 1806, using the field value and field type. In response, the database may provide a list of
 30 potential attributes, as well as a confidence level in those attribute associations to the
 31 entity, see e.g., 1807. The server may add data fields to the data record obtained for

1 entity attribute association specifying the potentially associated attributes and their
 2 associated confidence levels, e.g., 1808. In some embodiments, the server may utilize
 3 each data field in the data record obtained for cross-entity correlation to identify
 4 correlated entities, see e.g., 1809. The server may store the updated data record in a
 5 database, e.g., 1810.

6 **[00129]** FIGURE 19 shows a logic flow diagram illustrating example aspects of
 7 updating entity profile-graphs in some embodiments of the MMCDB, e.g., an Entity
 8 Profile-Graph Updating (“EPGU”) component 1900. In some implementations, a server
 9 may generate/update a profile for an entity whose data is stored within the MMCDB.
 10 The server may obtain an entity profile record for updating, e.g., 1901. The server may
 11 parse the entity profile record, and extract an entity identifier data field from the data
 12 record, e.g., 1902. The server may query a database for other data records that are
 13 related to the same entity, e.g., 1903, using the value for the entity identifier data field.
 14 In response, the database may provide a list of other data records for further processing.
 15 The server may select one of the other data records to update the entity profile record,
 16 e.g., 1904. The server may parse the data record, and extract all correlations,
 17 associations, and new data from the other record, e.g., 1905. The server may compare
 18 the correlations, attributes, associations, etc., from the other data record with the
 19 correlations, associations and attributes from the entity profile. Based on this
 20 comparison, the server may identify any new correlations, associations, etc., and
 21 generate an updated entity profile record using the new correlations, associations; flag
 22 new correlations, associations for further processing, e.g., 1907. In some embodiments,
 23 the server may utilize each data record obtained for updating the entity profile record as
 24 well as its social graph (e.g., as given by the correlations and associations for the entity),
 25 see e.g., 1909. The server may store the updated entity profile record in a database, e.g.,
 26 1908.

27 **[00130]** FIGURE 20 shows a logic flow diagram illustrating example aspects of
 28 generating search terms for profile-graph updating in some embodiments of the
 29 MMCDB, e.g., a Search Term Generation (“STG”) component 2000. In some
 30 implementations, a server may generate/update a profile for an entity whose data is
 31 stored within the MMCDB, by performing search for new data, e.g., across the Internet

1 and social networking services. The server may obtain an entity profile record for
 2 updating, e.g., 2001. The server may parse the entity profile record, and extract data
 3 field types and field values from the entity profile record, e.g., 2002. The server may
 4 query a database for other data records that are related to the same entity, e.g., 2003,
 5 using the values for the extracted data fields. In response, the database may provide a
 6 list of other data records for further processing. The server may parse the data records,
 7 and extract all correlations, associations, and data from the data records, e.g., 2004.
 8 The server may aggregate all the data values from all the records and the entity profile
 9 record, e.g., 2005. Based on this, the server may return the aggregated data values as
 10 search terms to trigger search processes (see e.g., FIG.3, 301-305), e.g., 2006.

11 **Electronic Virtual Wallet User Interface**

12 **[00131]** FIGURES 21A-E show user interface diagrams illustrating example
 13 features of user interfaces for an electronic virtual wallet in some embodiments of the
 14 MMCDB. With reference to FIGURE 21A, in some embodiments, a virtual wallet mobile
 15 app, e.g., 2111, executing on a device, e.g., 2100, of a user may include an app interface
 16 providing various features for the user. For example, the device may include a camera
 17 via which the app may acquire image frames, video data, live video, and/or the like, e.g.,
 18 2116. The app may be configured to analyze the incoming data, and search, e.g., 2112,
 19 for a product identifier, e.g., 2114, such as barcodes, QR codes and/or the like.

20 **[00132]** In some embodiments, the app may be configured to automatically detect,
 21 e.g., 2112, the presence of a product identifier within an image or video frame grabbed
 22 by the device (e.g., via a webcam, in-built camera, etc.). For example, the app may
 23 provide a “hands-free” mode of operation wherein the user may move the device to
 24 bring product identifiers within the field of view of the image/video capture mechanism
 25 of the device, and the app may perform image/video processing operations to
 26 automatically detect the product identifier within the field of view. In some
 27 embodiments, the app may overlay cross-hairs, target box, and/or like alignment
 28 reference markers, e.g., 2115, so that a user may align the product identifier using the
 29 reference markers to facilitate product identifier recognition and interpretation.

1 **[00133]** In some embodiments, the detection of a product identifier may trigger
2 various operations to provide products, services, information, etc. for the user. For
3 example, the app may be configured to detect and capture a QR code having embedded
4 merchant and/or product information, and utilize the information extracted from the
5 QR code to process a transaction for purchasing a product from a merchant. As other
6 examples, the app may be configured to provide information on related products,
7 quotes, pricing information, related offers, (other) merchants related to the product
8 identifier, rewards/loyalty points associated with purchasing the product related to the
9 product identifier, analytics on purchasing behavior, alerts on spend tracking, and/or
10 the like.

11 **[00134]** In some embodiments, the app may include user interface elements to
12 allow the user to manually search, e.g., 2113, for products (e.g., by name, brand,
13 identifier, etc.). In some embodiments, the app may provide the user with the ability to
14 view prior product identifier captures (see, e.g., 2117a) so that the user may be able to
15 better decide which product identifier the user desires to capture. In some
16 embodiments, the app may include interface elements to allow the user to switch back
17 and forth between the product identification mode and product offer interface display
18 screens (see, e.g., 2117b), so that a user may accurately study deals available to the user
19 before capturing a product identifier. In some embodiments, the user may be provided
20 with information about products, user settings, merchants, offers, etc. in list form (see,
21 e.g., 2117c) so that the user may better understand the user's purchasing options.
22 Various other features may be provided for in the app (see, e.g., 2117d). In some
23 embodiments, the user may desire to cancel product purchasing; the app may provide
24 the user with a user interface element (e.g., 2118) to cancel the product identifier
25 recognition procedure and return to the prior interface screen the user was utilizing.

26 **[00135]** With reference to FIGURE 21B, in some embodiments, the app may
27 include an indication of the location (e.g., name of the merchant store, geographical
28 location, information about the aisle within the merchant store, etc.) of the user, e.g.,
29 2121. The app may provide an indication of a pay amount due for the purchase of the
30 product, e.g., 2122. In some embodiments, the app may provide various options for the
31 user to pay the amount for purchasing the product(s). For example, the app may utilize

1 GPS coordinates associated with the device to determine the merchant store within
2 which the user is present, and direct the user to a website of the merchant. In some
3 embodiments, the app may be configured to make an application programming
4 interface (“API”) call to participating merchants to directly facilitate transaction
5 processing for purchases. In some embodiments, a merchant-branded app may be
6 developed with an in-app purchasing mode, which may directly connect the user into
7 the merchant’s transaction processing system. For example, the user may choose from a
8 number of cards (e.g., credit cards, debit cards, prepaid cards, etc.) from various card
9 providers, e.g., 2123a. In some embodiments, the app may provide the user the option
10 to pay the purchase amount using funds included in a bank account of the user, e.g., a
11 checking, savings, money market, current account, etc., e.g., 2123b. In some
12 embodiments, the user may have set default options for which card, bank account, etc.
13 to use for the purchase transactions via the app. In some embodiments, such setting of
14 default options may allow the user to initiate the purchase transaction via a single click,
15 tap, swipe, and/or other remedial user input action, e.g., 2123c. In some embodiments,
16 when the user utilizes such an option, the app may utilize the default settings of the user
17 to initiate the purchase transaction. In some embodiments, the app may allow the user
18 to utilize other accounts (e.g., Google™ Checkout, Paypal™ account, etc.) to pay for the
19 purchase transaction, e.g., 2123d. In some embodiments, the app may allow the user to
20 utilize rewards points, airline miles, hotel points, electronic coupons, printed coupons
21 (e.g., by capturing the printed coupons similar to the product identifier) etc., to pay for
22 the purchase transaction, e.g., 2123e. In some embodiments, the app may provide an
23 option to provide express authorization before initiating the purchase transaction, e.g.,
24 2124. In some embodiments, the app may provide a progress indicator provide
25 indication on the progress of the transaction after the user has selected an option to
26 initiate the purchase transaction, e.g., 2125. In some embodiments, the app may
27 provide the user with historical information on the user’s prior purchases via the app,
28 e.g., 2127a. In some embodiments, the app may provide the user with an option to share
29 information about the purchase (e.g., via email, SMS, wall posting on Facebook®, tweet
30 on Twitter™, etc.) with other users and/or control information shared with the
31 merchant, acquirer, payment network etc., to process the purchase transaction, e.g.,
32 2127b. In some embodiments, the app may provide the user an option to display the

1 product identification information captured by the client device (e.g., in order to show a
2 customer service representative at the exit of a store the product information), e.g.,
3 2127c. In some embodiments, the user, app, device and or purchase processing system
4 may encounter an error in the processing. In such scenarios, the user may be able to
5 chat with a customer service representative (e.g., VerifyChat 2127d) to resolve the
6 difficulties in the purchase transaction procedure.

7 **[00136]** In some embodiments, the user may select to conduct the transaction
8 using a one-time anonymized credit card number, see e.g., 2123f. For example, the app
9 may utilize a pre-designated anonymized set of card details (see, e.g., “AnonCard1,”
10 “AnonCard2”). As another example, the app may generate, e.g., in real-time, a one-time
11 anonymous set of card details to securely complete the purchase transaction (e.g., “Anon
12 It 1X”). In such embodiments, the app may automatically set the user profile settings
13 such that the any personal identifying information of the user will not be provided to the
14 merchant and/or other entities. In some embodiments, the user may be required to
15 enter a user name and password to enable the anonymization features.

16 **[00137]** With reference to FIGURE 21C, in some embodiments, the user interface
17 elements of the app may be advantageously arranged to provide the user the ability to
18 process a purchase with customized payment parameters with a minimum number of
19 user inputs applied to the user’s device. For example, if the user has a QR pay code, e.g.,
20 2132, within the viewing angle of a camera included in the user’s mobile device, the user
21 may activate a user interface element to snap the QR code. In some embodiments, the
22 user may control the field of view of the camera using a user interface zoom element,
23 e.g., 2133. In some embodiments, the user interface may be designed such that the user
24 may touch an image of a QR code displayed on the screen to capture the QR code (see
25 e.g., 2134). For example, the position of the user’s touch may be utilized as an input by
26 an image processing module executing on the user’s device to process the displayed
27 video frame (and/or adjacent video frames), and extract the QR code from the frame(s)
28 based on the user’s input. For example, the user’s touch may provide an approximate
29 center point of the QR code. Using this information, the image processing module may
30 be able to better perform an automated QR code image recognition, and accordingly

1 capture the correct QR code (e.g., if portions of many QR codes are displayed within the
2 video frame(s)) selected by the user for capture and processing.

3 **[00138]** In some embodiments, the app may utilize predetermined default settings
4 for a particular merchant, e.g., 2131, to process the purchase based on the QR code (e.g.,
5 in response to the user touching an image of a QR code displayed on the screen of the
6 user device). However, if the user wishes to customize the payment parameters, the
7 user may activate a user interface element 2135 (or e.g., press and continue to hold the
8 image of the QR code 2132). Upon doing so, the app may provide a pop-up menu, e.g.,
9 2137, providing a variety of payment customization choices, such as those described
10 with reference to FIGURE 21B. The user may, e.g., drag the user's finger to the
11 appropriate settings the user prefers, and release the user's finger from the touchscreen
12 of the user's mobile device to select the setting for payment processing. In alternate
13 embodiments, the payment settings options, e.g., 2137, and QR capture activation
14 button, e.g., 2136 may be included in the user interface along with a window for
15 capturing the QR code via the mobile device's camera. In alternate embodiments, the
16 user's mobile device may generate a hybrid QR code-payment settings graphic, and the
17 POS terminal (or user's trusted computing device) may capture the entire graphic for
18 payment processing. In some embodiments, the app may provide a user interface
19 element 2138 for the user to minimize the payment options settings user interface
20 elements. In some embodiments, the app may provide additional user interface
21 elements, e.g., 2139, to display previous purchases, data shared about those purchases,
22 purchase receipts (e.g., via barcodes) and customer support options (e.g., VerifyChat).

23 **[00139]** With reference to FIGURE 21D, in some embodiments, the user may be
24 able to view and/or modify the user profile and/or settings of the user, e.g., by activating
25 user interface element 2122 (of FIGURE 21B). For example, the user may be able to
26 view/modify a user name (e.g., 2141a-b), account number (e.g., 2142a-b), user security
27 access code (e.g., 2143a-b), user pin (e.g., 2144a-b), user address (e.g., 2145a-b), social
28 security number associated with the user (e.g., 2146a-b), current device GPS location
29 (e.g., 2147a-b), user account of the merchant in whose store the user currently is (e.g.,
30 2148a-b), the user's rewards accounts (e.g., 2149a-b), and/or the like. In some
31 embodiments, the user may be able to select which of the data fields and their associated

1 values should be transmitted to facilitate the purchase transaction, thus providing
 2 enhanced data security for the user. For example, in the example illustration in
 3 FIGURE 21D, the user has selected the name 2141a, account number 2142a, security
 4 code 2143a, merchant account ID 2148a and rewards account ID 2149a as the fields to
 5 be sent as part of the notification to process the purchase transaction. In some
 6 embodiments, the user may toggle the fields and/or data values that are sent as part of
 7 the notification to process the purchase transactions. In some embodiments, the app
 8 may provide multiple screens of data fields and/or associated values stored for the user
 9 to select as part of the purchase order transmission. In some embodiments, the app
 10 may obtain the GPS location of the user. Based on the GPS location of the user, the app
 11 may determine the context of the user (e.g., whether the user is in a store, doctor's
 12 office, hospital, postal service office, etc.). Based on the context, the app may present
 13 the appropriate fields to the user, from which the user may select fields and/or field
 14 values to send as part of the purchase order transmission.

15 **[00140]** For example, a user may go to doctor's office and desire to pay the co-pay
 16 for doctor's appointment. In addition to basic transactional information such as
 17 account number and name, the app may provide the user the ability to select to transfer
 18 medical records, health information, which may be provided to the medical provider,
 19 insurance company, as well as the transaction processor to reconcile payments between
 20 the parties. In some embodiments, the records may be sent in a Health Insurance
 21 Portability and Accountability Act (HIPAA)-compliant data format and encrypted, and
 22 only the recipients who are authorized to view such records may have appropriate
 23 decryption keys to decrypt and view the private user information.

24 **[00141]** With reference to FIGURE 21E, in some embodiments, the app executing
 25 on the user's device may provide a "VerifyChat" feature for fraud prevention (e.g., by
 26 activating UI element 2127d in FIGURE 21B). For example, the MMCDB may detect an
 27 unusual and/or suspicious transaction. The MMCDB may utilize the VerifyChat feature
 28 to communicate with the user, and verify the authenticity of the originator of the
 29 purchase transaction. In various embodiments, the MMCDB may send electronic mail
 30 message, text (SMS) messages, Facebook® messages, Twitter™ tweets, text chat, voice
 31 chat, video chat (e.g., Apple FaceTime), and/or the like to communicate with the user.

1 For example, the MMCDB may initiate a video challenge for the user, e.g., 2151a. For
2 example, the user may need to present him/her-self via a video chat, e.g., 2152a. In
3 some embodiments, a customer service representative, e.g., agent 2155a, may manually
4 determine the authenticity of the user using the video of the user. In some
5 embodiments, the MMCDB may utilize face, biometric and/or like recognition (e.g.,
6 using pattern classification techniques) to determine the identity of the user, e.g., 2154a.
7 In some embodiments, the app may provide reference marker (e.g., cross-hairs, target
8 box, etc.), e.g., 2153a, so that the user may the video to facilitate the MMCDB's
9 automated recognition of the user. In some embodiments, the user may not have
10 initiated the transaction, e.g., the transaction is fraudulent. In such embodiments, the
11 user may cancel, e.g., 2158a, the challenge. The MMCDB may then cancel the
12 transaction, and/or initiate fraud investigation procedures on behalf of the user. In
13 some embodiments, the app may provide additional user interface elements, e.g., to
14 display previous session 2156a, and provide additional customer support options (e.g.,
15 VerifyChat 2157a).

16 **[00142]** In some embodiments, the MMCDB may utilize a text challenge procedure
17 to verify the authenticity of the user, e.g., 2151b. For example, the MMCDB may
18 communicate with the user via text chat, SMS messages, electronic mail, Facebook®
19 messages, Twitter™ tweets, and/or the like. The MMCDB may pose a challenge
20 question, e.g., 2152b, for the user. The app may provide a user input interface
21 element(s) (e.g., virtual keyboard 2153b) to answer the challenge question posed by the
22 MMCDB. In some embodiments, the challenge question may randomly selected by the
23 MMCDB automatically; in some embodiments, a customer service representative 2155b
24 may manually communicate with the user. In some embodiments, the user may not
25 have initiated the transaction, e.g., the transaction is fraudulent. In such embodiments,
26 the user may cancel, e.g., 2158b, the text challenge. The MMCDB may then cancel the
27 transaction, and/or initiate fraud investigation procedures on behalf of the user. In
28 some embodiments, the app may provide additional user interface elements, e.g., to
29 display previous session 2156b, and provide additional customer support options (e.g.,
30 VerifyChat 2157b).

Merchant Analytics Platform

1 **[00143]** FIGURE 22 shows a block diagram illustrating example aspects of a
 2 merchant analytics platform in first set of embodiments of the MMCDB. In some
 3 implementations, a user, e.g., 2201, may desire to purchase products from a merchant.
 4 For example, the user may utilize a card (e.g., a credit card, debit, card, prepaid card,
 5 charge card, etc.) to purchase products, services, and/or other offerings (“products”)
 6 from a merchant 2202. In some implementations, the user may exhibit consumption
 7 patterns. For example, the user may often buy a similar set of products simultaneously
 8 each time the user shops. In some implementations, the purchasing patterns of the user
 9 may be reflected in the card transactions conducted by the user. For example, the
 10 consumption patterns may reflect in card transaction records of the transactions
 11 conducted by the user, which may be mined by a card company, e.g., 2203. In some
 12 implementations, information as to the general preferences of the user, purchasing
 13 preferences of the user, cost-sensitivities of the user, etc. may be gleaned from studying
 14 the aggregated card transaction records pertaining to the user. For example, analysis of
 15 the aggregated user card transaction records may indicate a preference for shopping
 16 within a particular geographical area, at particular times, with particular merchants, for
 17 particular products types, categories, brand names, quantities, and/or the like. As
 18 another example, analysis of the aggregated card transaction records may indicate
 19 correlations between purchases of the user. For example, the analysis may provide the
 20 ability to predict (with a known confidence level) that a user may purchase product B
 21 given that the user has purchased (or intends to purchase) product A (or products A,
 22 and/or C, and/or D, etc.). Thus, in some implementations, analysis of the aggregated
 23 card transaction records of a user may allow the MMCDB to provide suggestions to the
 24 merchant and/or user as to products that the user is likely to be interested in
 25 purchasing. For example, a user may desire suggestions as to what products, services,
 26 offerings, deals that user may be interested in, e.g., 2204. In some implementations, the
 27 MMCDB may provide such suggestions, e.g., 106, to the user on a real-time basis (e.g.,
 28 as the user is scanning products at a point-of-sale terminal, as the user is performing a
 29 price check, as the user is paying for a purchase, etc., as the user walks by a merchant
 30 where the MMCDB determines that products of interest to the user are available, etc.).

1 In some implementations, a merchant, e.g., 2202, may desire to understand customer
 2 behavior better so that the merchant may determine which products to provide for
 3 customers to generate maximum retail sales, generate customer loyalty, etc, e.g., 2205.
 4 In some implementations, the MMCDB may provide merchant analytics reports to the
 5 merchant including recommendations of product, service, discount, Groupon® offers,
 6 and/or other offers that the merchant can make to the user based on the user's
 7 behavioral patterns, e.g., 2206.

8 **[00144]** FIGURES 23A-B show data flow diagrams illustrating an example
 9 procedure to provide a user and/or merchant offers for products, services and/or the
 10 like, using user behavior patterns derived from card-based transaction data in some
 11 embodiments of the MMCDB. In some implementations, a user, e.g., 2301, may desire
 12 to purchase a product, service, offering, and/or the like ("product"), from a merchant.
 13 The user may communicate with a merchant server, e.g., 2303, via a client such as, but
 14 not limited to: a personal computer, mobile device, television, point-of-sale terminal,
 15 kiosk, ATM, pharmacy store, store counter, and/or the like (e.g., client 2302). For
 16 example, the user may provide user input, e.g., purchase input 2311, into the client
 17 indicating the user's desire to purchase the product. In various implementations, the
 18 user input may include, but not be limited to: keyboard entry, card swipe, activating a
 19 RFID/NFC enabled hardware device (e.g., electronic card having multiple accounts,
 20 smartphone, tablet, etc.), mouse clicks, depressing buttons on a joystick/game console,
 21 voice commands, single/multi-touch gestures on a touch-sensitive interface, touching
 22 user interface elements on a touch-sensitive display, and/or the like. For example, the
 23 user may direct a browser application executing on the client device to a website of the
 24 merchant, and may select a product from the website via clicking on a hyperlink
 25 presented to the user via the website. As another example, the client may obtain track 1
 26 data from the user's card (e.g., credit card, debit card, prepaid card, charge card, etc.),
 27 such as the example track 1 data provided below:

```
28 %B123456789012345^PUBLIC/J.Q.^99011200000000000000**901*****?*
29 (wherein '123456789012345' is the card number of 'J.Q. Public'
30 and has a CVV number of 901. '990112' is a service code, and ***
31 represents decimal digits which change randomly each time the
32 card is used.)
```

1 **[00145]** In some implementations, the client may generate a purchase order
 2 message, e.g., 2312, and provide, e.g., 2313, the generated purchase order message to
 3 the merchant server, e.g., 2303. For example, a browser application executing on the
 4 client may provide, on behalf of the user, a (Secure) Hypertext Transfer Protocol
 5 (“HTTP(S)”) GET message including the product order details for the merchant server
 6 in the form of data formatted according to the eXtensible Markup Language (“XML”).
 7 Below is an example HTTP(S) GET message including an XML-formatted purchase
 8 order message for the merchant server:

```

9  GET /purchase.php HTTP/1.1
10 Host: www.merchant.com
11 Content-Type: Application/XML
12 Content-Length: 1306
13 <?XML version = "1.0" encoding = "UTF-8"?>
14 <purchase_order>
15     <order_ID>4NFU4RG94</order_ID>
16     <timestamp>2011-02-22 15:22:43</timestamp>
17     <user_ID>john.q.public@gmail.com</user_ID>
18     <client_details>
19         <client_IP>192.168.23.126</client_IP>
20         <client_type>smartphone</client_type>
21         <client_model>HTC Hero</client_model>
22         <OS>Android 2.2</OS>
23         <app_installed_flag>true</app_installed_flag>
24     </client_details>
25     <purchase_details>
26         <num_products>1</num_products>
27         <product>
28             <product_type>book</product_type>
29             <product_params>
30                 <product_title>XML for
31 dummies</product_title>
32                 <ISBN>938-2-14-168710-0</ISBN>
33                 <edition>2nd ed.</edition>
34                 <cover>hardbound</cover>
35                 <seller>bestbuybooks</seller>

```

```

1           </product_params>
2           <quantity>1</quantity>
3       </product>
4   </purchase_details>
5   <account_params>
6       <account_name>John Q. Public</account_name>
7       <account_type>credit</account_type>
8       <account_num>123456789012345</account_num>
9       <billing_address>123 Green St., Norman, OK
10  98765</billing_address>
11       <phone>123-456-7809</phone>
12       <sign>/jpg</sign>
13       <confirm_type>email</confirm_type>
14
15       <contact_info>john.q.public@gmail.com</contact_info>
16   </account_params>
17   <shipping_info>
18       <shipping_address>same as billing</shipping_address>
19       <ship_type>expedited</ship_type>
20       <ship_carrier>FedEx</ship_carrier>
21       <ship_account>123-45-678</ship_account>
22       <tracking_flag>true</tracking_flag>
23       <sign_flag>false</sign_flag>
24   </shipping_info>
25 </purchase_order>

```

26 **[00146]** In some implementations, the merchant server may, in response to
 27 receiving the purchase order message from the client, generate, e.g., 2314, a request for
 28 merchant analytics from a pay network server, e.g., 2305, so that the merchant may
 29 provide product offerings for the user. For illustration, in the example above, the
 30 merchant server may add an XML-encoded data structure to the body of the purchase
 31 order message, and forward the message to the pay network server. An example XML-
 32 encoded data snippet that the merchant server may add to the body of the purchase
 33 order message before forwarding to the pay network server is provided below:

```

34 <analytics_request>

```

```

1      <request_ID>NEUI4BGF9</request_ID>
2      <details>
3          <type>products OR services OR discounts</type>
4          <deliver_to>user AND merchant</deliver_to>
5          <timeframe>realtime</timeframe>
6          <user_priority>high</user_priority>
7          <data_source>appended</data_source>
8      </details>
9      <merchant_params>
10         <merchant_ID>3FBCR4INC</merchant_id>
11         <merchant_name>Books & Things, Inc.</merchant_name>
12
13         <merchant_auth_key>1NNF484MCP59CHB27365</merchant_auth_ke
14 y>
15     </merchant_params>
16 </analytics_request>

```

17 **[00147]** The merchant server may provide the merchant analytics request, e.g.,
18 2315, to the pay network server. In some implementations, the pay network server may
19 extract the merchant and user profile information from the merchant analytics request.
20 For illustration, the pay network server may extract values of the ‘merchant_ID’ and
21 ‘user_ID’ fields from the merchant analytics request in the examples above. Using the
22 merchant and user profile information, the pay network server may determine whether
23 the merchant and/or user are enrolled in the merchant analytics program. In some
24 implementations, the pay network server may provide the results of merchant analytics
25 only to those entities that are enrolled in the merchant analytics program. For example,
26 the server may query a database, e.g., pay network database 2307, to determine whether
27 the user and/or merchant are enrolled in the merchant analytics program. In some
28 implementations, the pay network server may generate a query the database for user
29 behavior patterns of the user for merchant analytics, e.g., 2317. For example, the
30 database may be a relational database responsive to Structured Query Language (“SQL”)
31 commands. The pay network server may execute a hypertext preprocessor (“PHP”)
32 script including SQL commands to query the database for user behavior patterns of the

1 user. An example PHP/SQL command listing, illustrating substantive aspects of
 2 querying the database, is provided below:

```

3  <?PHP
4  header('Content-Type: text/plain');
5  mysql_connect("254.93.179.112", $DBserver, $password); // access
6  database server
7  mysql_select_db("USERS.SQL"); // select database table to search
8  //create query for issuer server data
9  $query = "SELECT behavior_profile_XML FROM UserBehaviorTable
10 WHERE userid LIKE '%' $user_id";
11 $result = mysql_query($query); // perform the search query
12 mysql_close("USERS.SQL"); // close database access
13 ?>

```

14 **[00148]** In response to obtaining the issuer server query, e.g., 2317, the pay
 15 network database may provide, e.g., 2318, the requested behavior patterns data to the
 16 pay network server. For example, the user behavior patterns data may comprise pair-
 17 wise correlations of various variables to each other, and/or raw user transaction
 18 patterns. An example XML-encoded user behavior pattern data file is provided below:

```

19
20 <?XML version = "1.0" encoding = "UTF-8"?>
21 <last_updated>2011-02-22 15:22:43</timestamp>
22 <user_ID>john.q.public@gmail.com</user_ID>
23 <pair_correlation_data>
24     <pair><time>AM</time><pdt>A</pdt><confidence>0.65</confid
25 ence></pair>
26     <pair><pdt>B</pdt><pdt>A</pdt><confidence>0.95</confidenc
27 e></pair>
28     <pair><zip>98456</zip><pdt>A</pdt><confidence>0.25</confi
29 dence></pair>
30     <pair><time>PM</time><zip>98465</zip><confidence>0.45</co
31 nfidence></pair>
32 </pair_correlation_data>
33 <raw_data>
34     <transaction>

```

```

1      <timestamp>2011-02-21 15:32:01</timestamp>
2      <product>
3          <product_type>book</product_type>
4          <product_params>
5              <product_title>XML for dummies</product_title>
6              <ISBN>938-2-14-168710-0</ISBN>
7              <edition>2nd ed.</edition>
8              <cover>hardbound</cover>
9              <seller>bestbuybooks</seller>
10         </product_params>
11         <quantity>1</quantity>
12     </transaction>
13     .
14     .
15     .
16     <transaction>
17         . . .
18     </transaction>
19 </raw_data>

```

[00149] In some implementations, the pay network server may identify products, services and/or other offerings likely desired by the user based on pre-generated user behavioral pattern analysis and user profile, e.g., 2319. The pay network server may generate a query, e.g., 2320, for merchants that may be able to provide the identified products, services, and/or offerings for the user. For example, the pay network server may generate a query based on the GPS coordinates of the user (e.g., obtained from the user's smartphone), the merchant store in which the user currently is present, etc., for merchants in the vicinity of the user who may have products included within the identified products likely desired by the user. In some implementations, the pay network server may also generate a query for offers (e.g., discount offers, Groupon® offers, etc.) that the merchant may be able to offer for the users. For example, the pay network server may utilize PHP/SQL commands similar to those provided above to query a database. In response, the database may provide, e.g., 2321, the requested merchant and/or offer data to the pay network server. In some implementations, the

1 pay network server may generate a real-time merchant analytics report for the
 2 merchant, e.g., 2322. In some implementations, the pay network server may generate a
 3 real-time geo-sensitive product offer packet for the user, e.g., including such items as
 4 (but not limited to): merchant names, location, directions, offers, discounts, interactive
 5 online purchase options, instant mobile wallet purchase ability, order hold placing
 6 features (e.g., to hold the items for pick up so as to prevent the items going out of stock,
 7 e.g., during seasonal shopping times), and/or the like. In some implementations, the
 8 pay network server may provide the merchant analytics report, e.g., 2324, to the
 9 merchant server, and may provide the real-time geo-sensitive product offer packet, e.g.,
 10 2327, to the client. In some implementations, the merchant server may utilize the pay
 11 network server's merchant analytics report to generate, e.g., 2325, offer(s) for the user.
 12 The merchant server may provide the generated offer(s), e.g., 2326, to the user. In some
 13 implementations, the client may render and display, e.g., 2328, the real-time geo-
 14 sensitive product offer packet from the pay network server and/or purchase offer(s)
 15 from the merchant to the user.

16 **[00150]** FIGURE 24 shows a logic flow diagram illustrating example aspects of
 17 providing a user and/or merchant offers for products, services and/or the like, using
 18 user behavior patterns derived from card-based transaction data in some embodiments
 19 of the MMCDB, e.g., a Merchant Analytics ("MA") component. In some
 20 implementations, the MMCDB may obtain a trigger to perform merchant analytics. For
 21 example a user may desire to purchase a product, service, offering, and/or the like
 22 ("product"), from a merchant (e.g., start scanning products in the checkout counter of
 23 the merchant's store), or may initiate a purchase transaction (e.g., attempt to pay for
 24 products purchased at the merchant store). In some implementations, the MMCDB
 25 may extract, e.g., 2402, the merchant and user profile information from the merchant
 26 analytics request. For example, the MMCDB may extract fields such as, but not limited
 27 to: user_ID, user_name, timestamp, merchant_ID, merchant_name, merchant_type,
 28 and/or the like. Using the merchant and/or user profile information, the MMCDB may
 29 generate a query the database for user behavior patterns, e.g., 2403, of the user for
 30 merchant analytics. In some implementations, the MMCDB may identify products,
 31 services and/or other offerings likely desired by the user based on pre-generated user

1 behavioral pattern analysis and user profile, e.g., 2404. The MMCDB may identify, e.g.,
 2 2405, merchants that may be able to provide the identified products, services, and/or
 3 offerings for the user. For example, the MMCDB may generate a query based on the
 4 GPS coordinates of the user (e.g., obtained from the user's smartphone), the merchant
 5 store in which the user currently is present, etc., for merchants in the vicinity of the user
 6 who may have products included within the identified products likely desired by the
 7 user. In some implementations, the pay network server may also determine offers (e.g.,
 8 discount offers, Groupon® offers, etc.), e.g., 2406, that the merchant may be able to
 9 offer for the users. In some implementations, the MMCDB may generate a real-time
 10 merchant analytics report for the merchant, e.g., 2407. In some implementations, the
 11 MMCDB may generate, e.g., 2408, a real-time geo-sensitive product offer packet for the
 12 user, e.g., including such items as (but not limited to): merchant names, location,
 13 directions, offers, discounts, interactive online purchase options, instant mobile wallet
 14 purchase ability, order hold placing features (e.g., to hold the items for pick up so as to
 15 prevent the items going out of stock, e.g., during seasonal shopping times), and/or the
 16 like. In some implementations, the MMCDB may provide the merchant analytics report
 17 to the merchant server, and may provide the real-time geo-sensitive product offer
 18 packet to the client, e.g., 2409.

19 **[00151]** FIGURE 25 shows a logic flow diagram illustrating example aspects of
 20 generating a user behavior pattern analysis in some embodiments of the MMCDB, e.g., a
 21 User Behavioral Pattern Analytics ("UBPA") component. In some implementations, the
 22 MMCDB may select, e.g., 2501, a user (e.g., via user ID) for behavioral pattern analysis.
 23 The MMCDB may store, e.g., 2502, card-based transaction data records for each card-
 24 based transaction performed by the user, e.g., via a Card-Based Transaction Execution
 25 component. The MMCDB may aggregate such card-based transaction data records of
 26 the user, e.g., 2503. For example, the MMCDB may utilize a Transaction Data
 27 Aggregation component such as that described above with reference to FIGS. 8-9. In
 28 various implementations, the MMCDB may aggregate card transaction records of the
 29 user according to criteria including, but not limited to: geographical location of card use,
 30 time of card use, type of purchase, quantity of purchase, transaction value, merchant
 31 type, merchant name, spending category (e.g., such as the North American Industry

1 Classification System (NAICS) codes for spending categories), and/or the like. The
 2 MMCDB may analyze the aggregated card transaction data, e.g., 2504, to determine
 3 user behavioral patterns, e.g., via a User Pattern Identification (“UPI”) component such
 4 as described below with reference to FIGURE 26. In some implementations, the
 5 MMCDB may provide user behavioral patterns obtained from the analysis for use by
 6 other MMCDB components and/or affiliated entities, e.g., 2505.

7 **[00152]** FIGURE 26 shows a logic flow diagram illustrating example aspects of
 8 identifying user behavioral patterns from aggregated card-based transaction data in
 9 some embodiments of the MMCDB, e.g., a User Pattern Identification (“UPI”)
 10 component. In some implementations, a pay network server (“server”) may obtain a
 11 user ID of a user for whom the server is required to generate user behavioral patterns,
 12 e.g., 2601. The server may query a database, e.g., a pay network database, for
 13 aggregated card transaction data records of the user, e.g., 2602. The server may also
 14 query, e.g., 2603, the pay network database for all possible field value that can be taken
 15 by each of the field values (e.g., AM/PM, zipcode, merchant_ID, merchant_name,
 16 transaction cost brackets, etc.). Using the field values of all the fields in the transaction
 17 data records, the server may generate field value pairs, for performing a correlation
 18 analysis on the field value pairs, e.g., 2604. An example field value pair is: ‘time’ is ‘AM’
 19 and ‘merchant’ is ‘Walmart’. The server may then generate probability estimates for
 20 each field value pair occurring in the aggregated transaction data records. For example,
 21 the server may select a field value pair, e.g., 2605. The server may determine the
 22 number of records within the aggregated transaction data records where the field value
 23 pair occurs, e.g., 2606. The server may then calculate a probability quotient for the field
 24 value pair by dividing the number determined for the occurrences of the field value pair
 25 by the total number of aggregate transaction data records, e.g., 2607. The server may
 26 also assign a confidence level for the probability quotient based on the sample size, e.g.,
 27 total number of records in the aggregated transaction data records, e.g., 2608. The
 28 server may generate and store an XML snippet, such as described above with reference
 29 to FIGURES 23A-B, including the field value pair, the probability quotient, and the
 30 confidence level associated with the probability quotient, e.g., 2609. The server may
 31 perform such a computation for each field value pair (see 2610) generated in 2604.

1 **[00153]** FIGURES 27A-B show block diagrams illustrating example aspects of
 2 merchant analytics in a second set of embodiments of the MMCDB. With reference to
 3 FIGURE 27A, in some implementations, the MMCDB may provide merchant analytics
 4 reports to various users in order to facilitate their making calculated investment
 5 decisions. For example, a stock investor may desire business analytics to determine
 6 which stocks the investor should invest in, how the investor should modify the investor's
 7 portfolio, and/or the like, e.g., 2701. In another example, a retailer may desire to
 8 understand customer behavior better so that the retailer may determine which products
 9 to provide for customers to generate maximum retail sales, e.g., 2702. In another
 10 example, a serviceperson providing services to customers may desire to understand
 11 which services the customer tend to prefer, and/or a paying for in the marketplace, e.g.,
 12 2703. In another example, a service provider may desire to understand the geographical
 13 areas where business for the serviceperson is likely to be concentrated, e.g., 2704. In
 14 some implementations, a credit card company may have access to a large database of
 15 card-based transactions. The card-based transaction may have distributed among them
 16 information on customer behavior, demand, geographical distribution, industry sector
 17 preferences, and/or the like, which may be mined in order to provide investors, retailer,
 18 service personnel and/or other users business analytics information based on analyzing
 19 the card-based transaction data. In some implementations, the MMCDB may take
 20 specific measures in order to ensure the anonymity of users whose card-based
 21 transaction data are analyzed for providing business analytics information for users.
 22 For example, the MMCDB may perform business analytics on anonymized card-based
 23 transaction data to provide solutions to questions such as illustrated in 2701-2704.

24 **[00154]** With reference to FIGURE 27B, in some implementations, the MMCDB
 25 may obtain an investment strategy to be analyzed, e.g., 2711, for example, from a user.
 26 The MMCDB may determine, e.g., 2712 the scope of the investment strategy analysis
 27 (e.g., geographic scope, amount of data required, industry segments to analyze, type of
 28 analysis to be generated, time-resolution of the analysis (e.g., minute, hour, day, month,
 29 year, etc.), geographic resolution (e.g., street, block, zipcode, metropolitan area, city,
 30 state, country, inter-continental, etc.). The MMCDB may aggregate card-based
 31 transaction data in accordance with the determined scope of analysis, e.g., 2713. The

1 MMCDB may normalized aggregated card-based transaction data records for uniform
 2 processing, e.g., 2714. In some implementations, the MMCDB may apply classification
 3 labels to card-based transaction data records, e.g., 2715, for investment strategy
 4 analysis. The MMCDB may filter the card-based transaction data records to include
 5 only those records as relevant to the analysis, e.g., 2716. For example, the MMCDB may
 6 utilize the classification labels corresponding to the transaction data records to
 7 determine which records are relevant to the analysis. In some implementations, the
 8 MMCDB may anonymize transaction data records for consumer privacy protection prior
 9 to investment strategy analysis, e.g., 2717. The MMCDB may perform econometrical
 10 investment strategy analysis, e.g., 2718, and generate an investment strategy analysis
 11 report based on the investment strategy analysis, e.g., 2719. The MMCDB may provide
 12 the investment strategy analysis report for the user requesting the investment strategy
 13 analysis.

14 **[00155]** FIGURES 28A-C show data flow diagrams illustrating an example
 15 procedure for econometrical analysis of a proposed investment strategy based on card-
 16 based transaction data in some embodiments of the MMCDB. With reference to
 17 FIGURE 28A, in some implementations, a user, e.g., 2801, may desire to obtain an
 18 analysis of an investment strategy. For example, the user may be a merchant, a retailer,
 19 an investor, a serviceperson, and/or the like provider or products, services, and/or other
 20 offerings. The user may communicate with a pay network server, e.g., 2805a, to obtain
 21 an investment strategy analysis. For example, the user may provide user input, e.g.,
 22 analysis request input 2811, into a client, e.g., 2802, indicating the user's desire to
 23 request an investment strategy analysis. In various implementations, the user input
 24 may include, but not be limited to: keyboard entry, mouse clicks, depressing buttons on
 25 a joystick/game console, voice commands, single/multi-touch gestures on a touch-
 26 sensitive interface, touching user interface elements on a touch-sensitive display, and/or
 27 the like. In some implementations, the client may generate an investment strategy
 28 analysis request, e.g., 2812, and provide, e.g., 2813, the generated investment strategy
 29 analysis request to the pay network server. For example, a browser application
 30 executing on the client may provide, on behalf of the user, a (Secure) Hypertext Transfer
 31 Protocol ("HTTP(S)") GET message including the investment strategy analysis request

1 in the form of XML-formatted data. Below is an example HTTP(S) GET message
 2 including an XML-formatted investment strategy analysis request:

```

3  GET /analysisrequest.php HTTP/1.1
4  Host: www.paynetwork.com
5  Content-Type: Application/XML
6  Content-Length: 1306
7  <?XML version = "1.0" encoding = "UTF-8"?>
8  <analysis_request>
9      <request_ID>EJ39FI1F</request_ID>
10     <timestamp>2011-02-24 09:08:11</timestamp>
11     <user_ID>investor@paynetwork.com</user_ID>
12     <password>*****</password>
13     <request_details>
14         <time_period>year 2011</time_period>
15         <time_interval>month-to-month</time_interval>
16         <area_scope>United States</area>
17         <area_resolution>zipcode</area_resolution>
18         <spend_sector>retail<sub>home
19 improvement</sub></spend_sector>
20     </request_details>
21     <client_details>
22         <client_IP>192.168.23.126</client_IP>
23         <client_type>smartphone</client_type>
24         <client_model>HTC Hero</client_model>
25         <OS>Android 2.2</OS>
26         <app_installed_flag>true</app_installed_flag>
27     </client_details>
28 </analysis_request>

```

29 **[00156]** In some implementations, the pay network server may parse the
 30 investment strategy analysis request, and determine the type of investment strategy
 31 analysis required, e.g., 2814. In some implementations, the pay network server may
 32 determine a scope of data aggregation required to perform the analysis. The pay
 33 network server may initiate data aggregation based on the determined scope, for
 34 example, via a Transaction Data Aggregation (“TDA”) component such as described

1 above with reference to FIGURE 9. The pay network server may query, e.g., 2816, a pay
 2 network database, e.g., 2807, for addresses of pay network servers that may have stored
 3 transaction data within the determined scope of the data aggregation. For example, the
 4 pay network server may utilize PHP/SQL commands similar to the examples provided
 5 above. The database may provide, e.g., 2817, a list of server addresses in response to the
 6 pay network server's query. Based on the list of server addresses, the pay network
 7 server may issue transaction data requests, e.g., 2818b-n, to the other pay network
 8 servers, e.g., 2805b-n. The other the pay network servers may query their transaction
 9 databases, e.g., 2810b-n, for transaction data falling within the scope of the transaction
 10 data requests. In response to the transaction data queries, e.g., 2819b-n, the transaction
 11 databases may provide transaction data, e.g., 2820b-n, to the other pay network servers.
 12 The other pay network servers may return the transaction data obtained from the
 13 transactions databases, e.g., 2821b-n, to the pay network server making the transaction
 14 data requests, e.g., 2805a.

15 **[00157]** With reference to FIGURE 28B, the pay network server 2805a may
 16 aggregate, e.g., 2823, the obtained transaction data records, e.g. via the TDA
 17 component. The pay network server may normalize, e.g., 2824, the aggregated
 18 transaction data so that all the data share a uniform data structure format, e.g., via a
 19 Transaction Data Normalization ("TDN") component such as described below with
 20 reference to FIGURE 29. The pay network server may generate, e.g., 2825-2828, one or
 21 more classification labels for each of the transaction data records, e.g., via a Card-Based
 22 Transaction Classification ("CTC") component such as described below with reference to
 23 FIGURE 30. The pay network server may query for classification rules, e.g., 2826, a
 24 database, e.g., pay network database 2807. Upon obtaining the classification rules, e.g.,
 25 2827, the pay network server may generate, e.g., 2828, classified transaction data
 26 records using the classification rules, e.g., via the CTC component. The pay network
 27 server may filter, e.g., 2829, relevant transaction data records using the classification
 28 labels, e.g., via a Transaction Data Filtering ("TDF") component such as described below
 29 with reference to FIGURE 31. The pay network server may anonymize, e.g., 2830, the
 30 transaction data records, e.g., via a Consumer Data Anonymization ("CDA") component
 31 such as described below with reference to FIGURE 32.

1 **[00158]** With reference to FIGURE 28C, the pay network server may, in some
 2 implementations, store aggregated, normalized, classified, filtered, and/or anonymized
 3 data records, e.g., 2832, in a database, e.g., transactions database 2810a. In some
 4 implementations, the pay network server may econometrically analyze, e.g., 2833,
 5 aggregated, normalized, classified, filtered, and/or anonymized data records, e.g., via an
 6 Econometrical Strategy Analysis (“ESA”) component such as described below with
 7 reference to FIGURE 33. The pay network server may prepare a report customized to
 8 the client used by the user. The pay network server may provide a reporting rules query
 9 to a database, e.g., pay network database 2807, for reporting rules to use in preparing
 10 the business analytics report. Upon obtaining the reporting rules, e.g., 2835, the pay
 11 network server may generate a business analytics report customized to the client, e.g.,
 12 2836, for example via a Business Analytics Reporting (“BAR”) such as described below
 13 with reference to FIGURE 34. The pay network server may provide the business
 14 analytics report, e.g., 2837, to the client, e.g., 2802. The client may render and display,
 15 e.g., 2838, the business analytics report for the user.

16 **[00159]** FIGURE 29 shows a logic flow diagram illustrating example aspects of
 17 normalizing raw card-based transaction data into a standardized data format in some
 18 embodiments of the MMCDB, e.g., a Transaction Data Normalization (“TDN”)
 19 component. In some implementations, a pay network server (“server”) may attempt to
 20 convert any transaction data records stored in a database it has access to in a
 21 normalized data format. For example, the database may have a transaction data record
 22 template with predetermined, standard fields that may store data in pre-defined formats
 23 (e.g., long integer / double float / 4 digits of precision, etc.) in a pre-determined data
 24 structure. A sample XML transaction data record template is provided below:

```

25   <?XML version = "1.0" encoding = "UTF-8"?>
26   <transaction_record>
27       <record_ID>00000000</record_ID>
28       <norm_flag>>false</norm_flag>
29       <timestamp>yyyy-mm-dd hh:mm:ss</timestamp>
30       <transaction_cost>$0,000,000,00</transaction_cost>
31       <merchant_params>
32           <merchant_id>00000000</merchant_id>

```

```

1         <merchant_name>TBD</merchant_name>
2         <merch_auth_key>0000000000000000</merch_auth_key>
3     </merchant_params>
4     <merchant_products>
5         <num_products>000</num_products>
6         <product>
7             <product_type>TBD</product_type>
8             <product_name>TBD</product_name>
9             <class_labels_list>TBD<class_labels_list>
10            <product_quantity>000</product_quantity>
11            <unit_value>$0,000,000.00</unit_value>
12            <sub_total>$0,000,000.00</sub_total>
13            <comment>normalized transaction data record
14 template</comment>
15        </product>
16    </merchant_products>
17    <user_account_params>
18        <account_name>JTBD</account_name>
19        <account_type>TBD</account_type>
20        <account_num>0000000000000000</account_num>
21        <billing_line1>TBD</billing_line1>
22        <billing_line2>TBD</billing_line2>
23        <zipcode>TBD</zipcode>
24        <state>TBD</state>
25        <country>TBD</country>
26        <phone>00-00-000-000-0000</phone>
27        <sign>TBD</sign>
28    </user_account_params>
29 </transaction_record>

```

30 **[00160]** In some implementations, the server may query a database for a
31 normalized transaction data record template, e.g., 2901. The server may parse the
32 normalized data record template, e.g., 2902. Based on parsing the normalized data
33 record template, the server may determine the data fields included in the normalized
34 data record template, and the format of the data stored in the fields of the data record
35 template, e.g., 2903. The server may obtain transaction data records for normalization.

1 The server may query a database, e.g., 2904, for non-normalized records. For example,
 2 the server may issue PHP/SQL commands to retrieve records that do not have the
 3 'norm_flag' field from the example template above, or those where the value of the
 4 'norm_flag' field is 'false'. Upon obtaining the non-normalized transaction data records,
 5 the server may select one of the non-normalized transaction data records, e.g., 2905.
 6 The server may parse the non-normalized transaction data record, e.g., 2906, and
 7 determine the fields present in the non-normalized transaction data record, e.g., 2907.
 8 The server may compare the fields from the non-normalized transaction data record
 9 with the fields extracted from the normalized transaction data record template. For
 10 example, the server may determine whether the field identifiers of fields in the non-
 11 normalized transaction data record match those of the normalized transaction data
 12 record template, (e.g., via a dictionary, thesaurus, etc.), are identical, are synonymous,
 13 are related, and/or the like. Based on the comparison, the server may generate a 1:1
 14 mapping between fields of the non-normalized transaction data record match those of
 15 the normalized transaction data record template, e.g., 2909. The server may generate a
 16 copy of the normalized transaction data record template, e.g., 2910, and populate the
 17 fields of the template using values from the non-normalized transaction data record,
 18 e.g., 2911. The server may also change the value of the 'norm_flag' field to 'true' in the
 19 example above. The server may store the populated record in a database (for example,
 20 replacing the original version), e.g., 2912. The server may repeat the above procedure
 21 for each non-normalized transaction data record (see e.g., 2913), until all the non-
 22 normalized transaction data records have been normalized.

23 **[00161]** FIGURE 30 shows a logic flow diagram illustrating example aspects of
 24 generating classification labels for card-based transactions in some embodiments of the
 25 MMCDB, e.g., a Card-Based Transaction Classification ("CTC") component. In some
 26 implementations, a server may apply one or more classification labels to each of the
 27 transaction data records. For example, the server may classify the transaction data
 28 records, according to criteria such as, but not limited to: geo-political area, luxury level
 29 of the product, industry sector, number of items purchased in the transaction, and/or
 30 the like. The server may obtain transactions from a database that are unclassified, e.g.,
 31 3001, and obtain rules and labels for classifying the records, e.g., 3002. For example,

1 the database may store classification rules, such as the exemplary illustrative XML-
2 encoded classification rule provided below:

```

3   <rule>
4       <id>NAICS44_45</id>
5       <name>NAICS - Retail Trade</name>
6       <inputs>merchant_id</inputs>
7       <operations>
8           <1>label = 'null'</1>
9           <1>cat = NAICS_LOOKUP(merchant_id)</1>
10          <2>IF (cat == 44 || cat ==45) label = 'retail
11 trade'</2>
12      </operations>
13      <outputs>label</outputs>
14  </rule>

```

15 **[00162]** The server may select an unclassified data record for processing, e.g.,
16 **3003**. The server may also select a classification rule for processing the unclassified
17 data record, e.g., **3004**. The server may parse the classification rule, and determine the
18 inputs required for the rule, e.g., **3005**. Based on parsing the classification rule, the
19 server may parse the normalized data record template, e.g., **3006**, and extract the values
20 for the fields required to be provided as inputs to the classification rule. For example, to
21 process the rule in the example above, the server may extract the value of the field
22 'merchant_id' from the transaction data record. The server may parse the classification
23 rule, and extract the operations to be performed on the inputs provided for the rule
24 processing, e.g., **3007**. Upon determining the operations to be performed, the server
25 may perform the rule-specified operations on the inputs provided for the classification
26 rule, e.g., **3008**. In some implementations, the rule may provide threshold values. For
27 example, the rule may specify that if the number of products in the transaction, total
28 value of the transaction, average luxury rating of the products sold in the transaction,
29 etc. may need to cross a threshold in order for the label(s) associated with the rule to be
30 applied to the transaction data record. The server may parse the classification rule to
31 extract any threshold values required for the rule to apply, e.g., **3009**. The server may
32 compare the computed values with the rule thresholds, e.g., **3010**. If the rule

1 threshold(s) is crossed, e.g., 3011, option “Yes,” the server may apply one or more labels
 2 to the transaction data record as specified by the classification rule, e.g., 3012. For
 3 example, the server may apply a classification rule to an individual product within the
 4 transaction, and/or to the transaction as a whole. In some implementations, the server
 5 may process the transaction data record using each rule (see, e.g., 3013). Once all
 6 classification rules have been processed for the transaction record, e.g., 3013, option
 7 “No,” the server may store the transaction data record in a database, e.g., 3014. The
 8 server may perform such processing for each transaction data record until all
 9 transaction data records have been classified (see, e.g., 3015).

10 **[00163]** FIGURE 31 shows a logic flow diagram illustrating example aspects of
 11 filtering card-based transaction data for econometrical investment strategy analysis in
 12 some embodiments of the MMCDB, e.g., a Transaction Data Filtering (“TDF”)
 13 component. In some implementations, a server may filter transaction data records prior
 14 to econometrical investment strategy analysis based on classification labels applied to
 15 the transaction data records. For example, the server may filter the transaction data
 16 records, according to criteria such as, but not limited to: geo-political area, luxury level
 17 of the product, industry sector, number of items purchased in the transaction, and/or
 18 the like. The server may obtain transactions from a database that are classified, e.g.,
 19 3101, and investment strategy analysis parameters, e.g., 3102. Based on the analysis
 20 parameters, the server may generate filter rules for the transaction data records, e.g.,
 21 3103. The server may select a classified data record for processing, e.g., 3104. The
 22 server may also select a filter rule for processing the classified data record, e.g., 3105.
 23 The server may parse the filter rule, and determine the classification labels required for
 24 the rule, e.g., 3106. Based on parsing the classification rule, the server may parse the
 25 classified data record, e.g., 3107, and extract values for the classification labels (e.g.,
 26 true/false) required to process the filter rule. The server may apply the classification
 27 labels values to the filter rule, e.g., 3108, and determine whether the transaction data
 28 record passes the filter rule, e.g., 3109. If the data record is admissible in view of the
 29 filter rule, e.g., 3110, option “Yes,” the server may store the transaction data record for
 30 further analysis, e.g., 3112. If the data record is not admissible in view of the filter rule,
 31 e.g., 3110, option “No,” the server may select another filter rule to process the

1 transaction data record. In some implementations, the server may process the
 2 transaction data record using each rule (see, e.g., 3111) until all rules are exhausted. The
 3 server may perform such processing for each transaction data record until all
 4 transaction data records have been filtered (see, e.g., 3113).

5 **[00164]** FIGURE 32 shows a logic flow diagram illustrating example aspects of
 6 anonymizing consumer data from card-based transactions for econometrical investment
 7 strategy analysis in some embodiments of the MMCDB, e.g., a Consumer Data
 8 Anonymization (“CDA”) component. In some implementations, a server may remove
 9 personal information relating to the user (e.g., those fields that are not required for
 10 econometrical investment strategy analysis) and/or merchant from the transaction data
 11 records. For example, the server may truncate the transaction data records, fill
 12 randomly generated values in the fields comprising personal information, and/or the
 13 like. The server may obtain transactions from a database that are to be anonymized,
 14 e.g., 3201, and investment strategy analysis parameters, e.g., 3202. Based on the
 15 analysis parameters, the server may determine the fields that are necessary for
 16 econometrical investment strategy analysis, e.g., 3203. The server may select a
 17 transaction data record for processing, e.g., 3204. The server may parse the transaction
 18 data record, e.g., 3205, and extract the data fields in the transactions data records. The
 19 server may compare the data fields of the transaction data record with the fields
 20 determined to be necessary for the investment strategy analysis, e.g., 3206. Based on
 21 the comparison, the server may remove any data fields from the transaction data record,
 22 e.g., those that are not necessary for the investment strategy analysis, and generate an
 23 anonymized transaction data record, e.g., 3207. The server may store the anonymized
 24 transaction data record in a database, e.g., 3208. In some implementations, the server
 25 may process each transaction data record (see, e.g., 3209) until all the transaction data
 26 records have been anonymized.

27 **[00165]** FIGURES 33A-B show logic flow diagrams illustrating example aspects of
 28 econometrically analyzing a proposed investment strategy based on card-based
 29 transaction data in some embodiments of the MMCDB, e.g., an Econometrical Strategy
 30 Analysis (“ESA”) component. With reference to FIGURE 33A, in some
 31 implementations, the server may obtain spending categories (e.g., spending categories

1 as specified by the North American Industry Classification System (“NAICS”)) for which
 2 to generate estimates, e.g., 3301. The server may also obtain the type of forecast (e.g.,
 3 month-to-month, same-month-prior-year, yearly, etc.) to be generated from the
 4 econometrical investment strategy analysis, e.g., 3302. In some implementations, the
 5 server may obtain the transaction data records using which the server may perform
 6 econometrical investment strategy analysis, e.g., 3303. For example, the server may
 7 select a spending category (e.g., from the obtained list of spending categories) for which
 8 to generate the forecast, e.g., 3304. For example, the forecast series may be several
 9 aggregate series (described below) and the 12 spending categories in the North
 10 American Industry Classification System (NAICS) such as department stores, gasoline,
 11 and so on, that may be reported by the Department of Commerce (DOC).

12 **[00166]** To generate the forecast, the server may utilize a random sample of
 13 transaction data (e.g., approximately 6% of all transaction data within the network of
 14 pay servers), and regression analysis to generate model equations for calculating the
 15 forecast from the sample data. For example, the server may utilize distributed
 16 computing algorithms such as Google MapReduce. Four elements may be considered in
 17 the estimation and forecast methodologies: (a) rolling regressions; (b) selection of the
 18 data sample (“window”) for the regressions; (c) definition of explanatory variables
 19 (selection of accounts used to calculate spending growth rates); and (d) inclusion of the
 20 explanatory variables in the regression equation (“candidate” regressions) that may be
 21 investigated for forecasting accuracy. The dependent variable may be, e.g., the growth
 22 rate calculated from DOC revised sales estimates published periodically. Rolling
 23 regressions may be used as a stable and reliable forecasting methodology. A rolling
 24 regression is a regression equation estimated with a fixed length data sample that is
 25 updated with new (e.g., monthly) data as they become available. When a new data
 26 observation is added to the sample, the oldest observation is dropped, causing the total
 27 number of observations to remain unchanged. The equation may be estimated with the
 28 most recent data, and may be re-estimated periodically (e.g., monthly). The equation
 29 may then be used to generate a one-month ahead forecast for year-over-year or month
 30 over month sales growth.

1 **[00167]** Thus, in some implementations, the server may generate N window
 2 lengths (e.g., 18 mo, 24 mo, 36 mo) for rolling regression analysis, e.g., 3305. For each
 3 of the candidate regressions (described below), various window lengths may be tested to
 4 determine which would systemically provide the most accurate forecasts. For example,
 5 the server may select a window length may be tested for rolling regression analysis, e.g.,
 6 3306. The server may generate candidate regression equations using series generated
 7 from data included in the selected window, e.g., 3307. For example, the server may
 8 generate various series, such as, but not limited to:

9 **[00168]** Series (1): Number of accounts that have a transaction in the selected
 10 spending category in the current period (e.g., month) and in the prior period (e.g.,
 11 previous month / same month last year);

12 **[00169]** Series (2): Number of accounts that have a transaction in the selected
 13 spending category in the either the current period (e.g., month), and/or in the prior
 14 period (e.g., previous month / same month last year);

15 **[00170]** Series (3): Number of accounts that have a transaction in the selected
 16 spending category in the either the current period (e.g., month), or in the prior period
 17 (e.g., previous month / same month last year), but not both;

18 **[00171]** Series (4): Series (1) + overall retail sales in any spending category from
 19 accounts that have transactions in both the current and prior period;

20 **[00172]** Series (5): Series (1) + Series (2) + overall retail sales in any spending
 21 category from accounts that have transactions in both the current and prior period; and

22 **[00173]** Series (6): Series (1) + Series (2) + Series (3) + overall retail sales in any
 23 spending category from accounts that have transactions in both the current and prior
 24 period.

25 **[00174]** With reference to FIGURE 33B, in some implementations, the server may
 26 calculate several (e.g., six) candidate regression equations for each of the series. For
 27 example, the server may calculate the coefficients for each of the candidate regression
 28 equations. The server may calculate a value of goodness of fit to the data for each
 29 candidate regression equations, e.g., 3308. For example, two measures of goodness of

1 fit may be used: (1) out-of-sample (simple) correlation; and (2) minimum absolute
 2 deviation of the forecast from revised DOC estimates. In some implementations,
 3 various measures of goodness of fit may be combined to create a score. In some
 4 implementations, candidate regression equations may be generated using rolling
 5 regression analysis with each of the N generated window lengths (see, e.g., 3309). In
 6 some implementations, upon generation of all the candidate regression equations and
 7 their corresponding goodness of fit scores, the equation(s) with the best score is chosen
 8 as the model equation for forecasting, e.g., 3310. In some implementations, the
 9 equation(s) with the highest score is then re-estimated using latest retail data available,
 10 e.g., from the DOC, e.g., 3311. The rerun equations may be tested for auto correlated
 11 errors. If the auto correlation test is statistically significant then the forecasts may
 12 include an autoregressive error component, which may be offset based on the
 13 autocorrelation test.

14 **[00175]** In some implementations, the server may generate a forecast for a
 15 specified forecast period using the selected window length and the candidate regression
 16 equation, e.g., 3312. The server may create final estimates for the forecast using DOC
 17 estimates for prior period(s), e.g., 3313. For example, the final estimates (e.g., F_t^Y -
 18 year-over-year growth, F_t^M - month-over-month growth) may be calculated by averaging
 19 month-over-month and year-over-year estimates, as follows:

$$\begin{aligned}
 20 \quad D_t^Y &= (1 + G_t^Y) \bullet R_{t-12} \\
 21 \quad D_t^M &= (1 + G_t^M) \bullet A_{t-1} \\
 22 \quad D_t &= \text{Mean}(D_t^M, D_t^Y) \\
 23 \quad B_{t-1}^Y &= (1 + G_{t-1}^Y) \bullet R_{t-13} \\
 24 \quad B_{t-1}^M &= A_{t-1} \\
 25 \quad B_{t-1} &= \text{Mean}(B_{t-1}^M, B_{t-1}^Y) \\
 26 \quad F_t^Y &= \frac{D_t}{R_{t-12}} - 1
 \end{aligned}$$

$$F_t^M = \frac{D_t}{B_{t-1}} - 1$$

[00176] Here, G represents the growth rates estimated by the regressions for year (superscript Y) or month (superscript M), subscripts refer to the estimate period, t is the current forecasting period); R represents the DOC revised dollar sales estimate; A represents the DOC advance dollar estimate; D is a server-generated dollar estimate, B is a base dollar estimate for the previous period used to calculate the monthly growth forecast.

[00177] In some implementations, the server may perform a seasonal adjustment to the final estimates to account for seasonal variations, e.g., 3314. For example, the server may utilize the X-12 ARIMA statistical program used by the DOC for seasonal adjustment. The server may then provide the finalized forecast for the selected spending category, e.g., 3315. Candidate regressions may be similarly run for each spending category of interest (see, e.g., 3316).

[00178] FIGURE 34 shows a logic flow diagram illustrating example aspects of reporting business analytics derived from an econometrical analysis based on card-based transaction data in some embodiments of the MMCDB, e.g., a Business Analytics Reporting (“BAR”) component. In some implementations, the server may customize a business analytics report to the attributes of a client of the user requesting the investment strategy analysis, e.g., 3401. The server may obtain an investment strategy analysis request from a client. The request may include details about the client such as, but not limited to: client_type, client_IP, client_model, client_OS, app_installed_flag, and/or the like. The server may parse the request, e.g., 3402, and determine the type of client (e.g., desktop computer, mobile device, smartphone, etc.). Based on the type of client, the server may determine attributes of the business analytics report, including but not limited to: report size; report resolution, media format, and/or the like, e.g., 3403. The server may generate the business analytics report according to the determined attributes, e.g., 3404. The server may compile the report into a media format according to the attributes of the client, e.g., 3405, and provide the business analytics report for the client, e.g., 3406. Optionally, the server may initiate actions

1 (e.g., generate a market data feed, trigger an investment action, trigger a wholesale
 2 purchase of goods for a retailer, etc.) based on the business analytics report and/or data
 3 utilized in preparing the business analytics report, e.g., 3407.

4 **Analytical Model Sharing**

5 **[00179]** Thus, as seen from the discussion above, in various embodiments, the
 6 MMCDB facilitates the creation of analytical models using which the data aggregated by
 7 the Centralized Personal Information Platform of the MMCDB may be utilized to
 8 provide business or other intelligence to the various users of the MMCDB. Examples of
 9 analytical models include the components discussed above in the discussion with
 10 reference to FIGURES 24 and 33A-B. In some implementations, the MMCDB may
 11 facilitate the sharing of such analytical models among various users and/or other
 12 entities or components associated with the MMCDB. For example, a developer of an
 13 analytical model such as the real-time offer merchant analytics report-generating
 14 component of FIGURE 24 may distribute the analytical model to other users of the
 15 MMCDB. Optionally, the model may be described according to an encryptmatics XML
 16 format, as discussed in detail further below in this disclosure. In some embodiments,
 17 the analytical model may be provide without providing the model data sets based on
 18 which the model was developed, so as to protect the privacy of the consumers whose
 19 data were included in the model data set. In alternate embodiments, the MMCDB may
 20 utilize a consumer data anonymization component such as that described above with
 21 reference to FIGURE 32, before sharing the model data set along with the analytical
 22 model.

23 **[00180]** FIGURE 35 shows a logic flow diagram illustrating example aspects of
 24 sharing an analytical model generated using data acquired using the centralized
 25 personal information platform in some embodiments of the MMCDB, e.g., an Analytical
 26 Model Sharing (“AMS”) component. In some embodiments, the MMCDB may obtain an
 27 analytical model provided for sharing with other users, e.g., 3501. The MMCDB may
 28 parse the analytical model, e.g., using one of the parsers described below with reference
 29 to FIGURE 49. The MMCDB may, based on the parsing of the model, identify any
 30 model data set used to develop the analytical model, that is included in the model

1 provided for sharing, e.g., 3502. The MMCDB may determine, if such a model dataset is
 2 included, whether the model dataset includes private data that may be shared on an
 3 open exchange, e.g., personally identifiable information, e.g., 3503. If the data is
 4 allowed to be shared, e.g., 3504, option “No,” the MMCDB may provide the analytical
 5 model for sharing, e.g., to a model exchange website, without any further processing,
 6 e.g., 3505. If however, the model dataset include data that may not be shared, e.g.,
 7 3504, option “Yes,” the MMCDB may determine whether the model dataset may be
 8 excluded from the model, while still allowing the model to be properly used, e.g., 3506.
 9 If the model dataset is excludable, e.g., 3507, option “Yes,” the MMCDB may delete the
 10 model dataset from the analytical model, e.g., 3508, and provide the modified analytical
 11 model for sharing, e.g., 3505. If however, the model dataset is not excludable, e.g.,
 12 3507, option “No,” the MMCDB may determine whether the dataset can be anonymized
 13 while still allowing the analytical model to function properly. If the dataset can be
 14 anonymized, e.g., 3509, option “Yes,” the MMCDB may anonymize the model dataset,
 15 e.g., using the CDA component discussed above with reference to FIGURE 32, e.g.,
 16 3510. Then, the MMCDB may provide the modified analytical model including the
 17 anonymized dataset for sharing, e.g., 3505. If however, the model dataset cannot be
 18 anonymized, e.g., 3509, option “No,” the MMCDB may generate and provide a sharing
 19 rejection message to the provider of the analytical model for sharing, e.g., 3511.

20

Encryptmatics XML Data Converter

21 **[00181]** In some embodiments, the MMCDB may utilize metadata (e.g., easily
 22 configurable data) to drive an analytics and rule engine that may convert any structured
 23 data obtained via the centralized personal information platform, discussed above in this
 24 disclosure, into a standardized XML format (“encryptmatics” XML). See FIGURE 14B
 25 for additional description. The encryptmatics XML may then be processed by an
 26 encryptmatics engine of the MMCDB that is capable of parsing, transforming and
 27 analyzing data to generate decisions based on the results of the analysis. Accordingly, in
 28 some embodiments, the MMCDB may implement a metadata-based interpretation
 29 engine that parses structured data, including, but not limited to: web content, graph
 30 databases, micro blogs, images or software code, and convert the structured data into

1 commands in the encryptmatics XML file format. The structured data may include,
 2 without limitation, software code, images, free text, relational database queries, graph
 3 queries, sensory inputs, and/or the like. As an example, the MMCDB may convert
 4 software code written in SAS integrated system of software products provided by SAS
 5 Institute, Inc., into a standard encryptmatics XML structure. The example below shows
 6 SAS code and encryptmatics XML that serves a similar purpose to the SAS code -
 7 defining a module's inputs, outputs, and function calls:

```

8 // SAS
9 filename myFTL "my.378.FTL";
10 data MyFile;
11     length yyddd $5. ;
12     infile myFTL lrecl=50000;
13     input @21 yyddd ;
14 run;
15
16 // Encryptmatics XML
17 <lock name="F: Transaction Date : yyddd"
18     inkeyid="0" inkeystart="21" inkeystop="25"
19     outkeyid="31" outkeyindex="1"
20     function="INSTANT"
21     type="STRING"
22 />
```

23 **[00182]** In the encryptmatics XML examples herein, a “key” represents a collection
 24 of data values. A “tumbler” represents a hash lookup table that may also allow wild card
 25 searches. A “lock” represents a definition including one or more input data sources,
 26 data types for the input sources, one or more data output storage variables, and
 27 functions/modules that may be called to process the input data from the input data
 28 sources. A “door” may refer to a collection of locks, and a vault may represent a model
 29 package defining the input, output, feature generation rules and analytical models.
 30 Thus, the encryptmatics XML may be thought of as a framework for calling functions
 31 (e.g., INSTANT - returns the raw value, LAG - return a key from a prior transaction,
 32 ADD - add two keys together, OCCURRENCE - returns the number of times a key value

1 occurred in prior transactions, AVG - returns an average of past and current key values,
2 etc.) and data lookups with a shared storage space to process a grouped data stream.

3 **[00183]** In some embodiments, a metadata based interpretation engine may
4 populate a data/command object (e.g., an encryptmatics XML data structure defining a
5 “vault”) based on a given data record, using configurable metadata. The configurable
6 metadata may define an action for a given glyph or keyword contained within a data
7 record. The MMCDB may obtain the structured data, and perform a standardization
8 routine using the structured data as input (e.g., including script commands, for
9 illustration). For example, the MMCDB may remove extra line breaks, spaces, tab
10 spaces, etc. from the structured data. The MMCDB may determine and load a metadata
11 library, using which the MMCDB may parse subroutines or functions within the script,
12 based on the metadata. In some embodiments, the MMCDB may pre-parse conditional
13 statements based on the metadata. The MMCDB may also parse data to populate a
14 data/command object based on the metadata and prior parsing. Upon finalizing the
15 data/command object, the MMCDB may export the data/command object as XML in
16 standardized encryptmatics format. For example, the engine may process the object to
17 export its data structure as a collection of encryptmatics vaults in a standard
18 encryptmatics XML file format. The encryptmatics XML file may then be processed to
19 provide various features by an encryptmatics engine.

20 **[00184]** As an example, using such a metadata based interpretation engine, the
21 MMCDB can generate the encryptmatics XML code, provided below, from its equivalent
22 SAS code, provided beneath the encryptmatics XML code generated from it:

```
23 // SAS function code
24 myInput    = filename("../data/30x.raw", "fixed",
25   "../metaData/ftl_302.meta");
26
27 data myout;
28   set myInput;
29   auth_amt  = float(myInput.auth_amt);
30   auth_amt2 = log(auth_amt);
31   run;
32
```

```

1  proc freq data = myout;
2      tables auth_amt2  ;
3      run;
4
5
6  // Equivalent encryptmatics XML function code
7  <init>
8  loop=mainLoop
9      <input>
10         keyname=myinput
11         file=../data/30x.raw
12         format=fixed
13         meta_data= ../metaData/ftl_302.meta
14     </input>
15     <output>
16         keyname=myout
17         file=VARRAY
18         format=VARRAY
19         meta_data= {'auth_amt2': (1, 0, 'String'), 'auth_amt': (0, 0,
20 'String') }
21     </output>
22 </init>
23 <vault>
24     <door>
25         <lock>
26             outkey=myout
27             outkeyname=auth_amt
28             inkey=myinput
29             inkeyname=auth_amt
30             function=float
31             type=String
32         </lock>
33         <lock>
34             outkey=myout
35             outkeyname=auth_amt2
36             inkey=myout
37             inkeyname=auth_amt

```

```

1      function=log
2      type=String
3      </lock>
4      </door>
5  </vault>
6      <init>
7      summary_level=2
8      loop=mainLoop
9      <input>
10         keyname=myout
11         file=VARRAY:myout
12         format=array
13         meta_data= {'auth_amt2': (1, 0, 'String'), 'auth_amt': (0,
14 0, 'String')}
15     </input>
16     <output>
17         keyname=_output
18         file=stdout
19         format=VARRAY
20         meta_data= {'_aggl': (0, 0, 'object')}
21     </output>
22 </init>
23 <vault>
24     <door>
25         <lock>
26             outkey=aggl
27             outkeyname=auth_amt2
28             inkey=myout
29             inkeyname=auth_amt2
30             function=instant
31             type=String
32         </lock>
33     </door>
34     <door>
35         <lock>
36             outkey=_output
37             outkeyname=_aggl

```

```

1      function=aggfreq
2      type=object
3      parser=noparse
4      groupkeyname=agg1
5      </lock>
6  </door>
7  </vault>

```

8 **[00185]** As another example, using such a metadata based interpretation engine,
 9 the MMCDB can generate the encryptmatics XML code, provided below, from its
 10 equivalent SAS code, provided beneath the encryptmatics XML code generated from it:

```

11  // SAS function code
12  myInput          = filename("../data/vnd.test.json", "JSON",
13  "../metaData/enrollment.meta");
14  myTumblar = tumblarname("../tumblars/enrollment.exp.tumblar");
15
16  data myOut;
17      set myInput;
18      customer_ipaddresstmp = tumble(myInput.customer_ipaddress ,
19  customer_ipaddress );
20      customer_ipaddress    =  myOut.customer_ipaddresstmp/1000;
21      cvv_resulttmp = tumble(myInput.cvv_result , cv_result );
22      cv_result      =  myOut.cvv_resulttmp/1000;
23      keep customer_ipaddress cv_result;
24      run;
25
26  proc model data = myOut out=Scored;
27      features = customer_ipaddress cv_result ;
28      weights  = 1,1 ;
29      type     = 'bayes' ;
30      run;
31
32  proc print data = Scored;
33      run;
34
35

```

```

1  // Equivalent encryptmatics XML function code
2  <init>
3  loop=mainLoop
4    <input>
5      keyname=myinput
6      file=../data/vnd.test.json
7      format=JSON
8      meta_data= ../metaData/ftl_302.meta  <br/>
9    </input>
10   <output>
11     keyname=myout
12     file=VARRAY
13     format=VARRAY
14     meta_data= {'cv_result': (1, 0, 'String'),
15 'customer_ipaddress': (0, 0, 'String')}
16   </output>
17   <constant>
18     indexname=_constant_1000
19     value=1000
20     type=float
21   </constant>
22 </init>
23 <vault>
24   <door>
25     <lock>
26       outkey=_old_myout
27       outkeyname=customer_ipaddresstmp
28       inkey=myinput
29       inkeyname=customer_ipaddress
30       function=INSTANT
31       type=String
32       tumblar=customer_ipaddress
33     </lock>
34     <lock>
35       outkey=_old_myout
36       outkeyname=customer_ipaddress
37       inkey=_old_myout

```

```
1      inkeyname=customer_ipaddresstmp
2      inkey2=_constants
3      inkey2name=_constant_1000
4      function=divide
5      type=String
6  </lock>
7  <lock>
8      outkey=_old_myout
9      outkeyname=cvv_resulttmp
10     inkey=myinput
11     inkeyname=cv_result
12     function=INSTANT
13     type=String
14     tumblar=cv_result
15 </lock>
16 <lock>
17     outkey=_old_myout
18     outkeyname=cv_result
19     inkey=_old_myout
20     inkeyname=cvv_resulttmp
21     inkey2=_constants
22     inkey2name=_constant_1000
23     function=divide
24     type=String
25 </lock>
26 <lock>
27     outkey=myout
28     outkeyname=customer_ipaddress
29     inkey=_old_myout
30     inkeyname=customer_ipaddress
31     function=instant
32     type=String
33 </lock>
34 <lock>
35     outkey=myout
36     outkeyname=cv_result
37     inkey=_old_myout
```

```

1      inkeyname=cv_result
2      function=instant
3      type=String
4  </lock>
5  </door>
6 </vault>
7  <init>
8  loop=mainLoop
9      <input>
10         keyname=myout
11         file=VARRAY:myout
12         format=array
13         meta_data= {'cv_result': (1, 0, 'String'),
14 'customer_ipaddress': (0, 0, 'String')}
15     </input>
16     <output>
17         keyname=scored
18         file=VARRAY
19         format=VARRAY
20         meta_data= {'_mdl1': (0, 0, 'float')}
21     </output>
22 </init>
23 <vault>
24     <door>
25         <lock>
26             outkey=mdl1
27             outkeyname=customer_ipaddress
28             inkey=myout
29             inkeyname=customer_ipaddress
30             function=instant
31             type=String
32         </lock>
33         <lock>
34             outkey=mdl1
35             outkeyname=cv_result
36             inkey=myout
37             inkeyname=cv_result

```

```

1      function=instant
2      type=String
3  </lock>
4  </door>
5  <door>
6      <lock>
7          outkey=scored
8          outkeyname=_mdl1
9          function=_mdl1
10         type=float
11         fnc-weights=1.0,1.0
12         function=SUMPBROB
13         model=bayes
14         parser=noparse
15         groupkeyname=mdl1
16     </lock>
17 </door>
18 </vault>
19 <init>
20 outputall=True
21 loop=print
22 <input>
23     keyname=scored
24     file=VARRAY:scored
25     format=array
26     meta_data= {'_mdl1': (0, 0, 'float')}
27 </input>
28 <output>
29     keyname=_output
30     file=stdout
31     format=VARRAY
32     meta_data= {'_mdl1': (0, 0, 'String')}
33 </output>
34 </init>
35 <vault>
36     <door>
37         <lock>

```

```

1      outkey=_output
2      outkeyname=_mdl1
3      inkey=scored
4      inkeyname=_mdl1
5      function=instant
6      type=String
7  </lock>
8  </door>
9  </vault>

```

10 **[00186]** FIGURE 36 shows a logic flow diagram illustrating example aspects of a
11 metadata based interpretation engine of the MMCDB that generates standardized
12 encryptmatics XML from structured data obtained from various sources via the
13 centralized personal information platform, e.g., an Encryptmatics XML Converter
14 (“EXC”) component. In some embodiments, the MMCDB may obtain a structured data
15 object for conversion into encryptmatics XML format, e.g., 3601. The MMCDB may
16 parse the structured data, e.g., 3602. For example, the MMCDB may utilize parsers
17 such as the example parsers discussed below with reference to FIGURE 49. In some
18 embodiments, the MMCDB may determine and load a metadata library, using which the
19 MMCDB may parse subroutines or functions within the script, based on the metadata.
20 In some embodiments, the MMCDB may pre-parse conditional statements based on the
21 metadata. The MMCDB may also parse data to populate a data/command object based
22 on the metadata and prior parsing. The MMCDB may obtain the structured data, and
23 perform a standardization routine using the structured data as input (e.g., including
24 script commands, for illustration). For example, the MMCDB may optionally eliminate
25 superfluous characters, e.g., extra line breaks, spaces, tabs, etc., to generate a modified
26 structured data object, e.g., 3603. The MMCDB may extract a glyph or keywords from
27 the modified structured data, e.g., 3604. The MMCDB may, using the metadata library,
28 lookup a database (e.g., a metadata library) for an encryptmatics XML metadata code
29 snippet corresponding to the extracted glyph or keyword, e.g., 3605, and append the
30 retrieved encryptmatics XML metadata to a metadata object, e.g., 3606. The MMCDB
31 may perform such a routine until all glyphs or keywords are extracted and processed
32 from the modified structured data, see e.g., 3607. Then, the MMCDB may, upon

1 finalizing the data/command object, export the data/command object as XML in
 2 standardized encryptmatics file format, e.g., 3608. For example, the engine may
 3 process the object to export its data structure as a collection of encryptmatics vaults in a
 4 standard encryptmatics XML file format. In some embodiments, the MMCDB may
 5 execute an application defined by the exported encryptmatics file, e.g., on other
 6 structured data available in the centralized personal information platform, e.g., 3609.

7 **[00187]** Thus, in some embodiments, the MMCDB may gradually convert the
 8 entire centralized personal information platform from structured data into standardized
 9 encryptmatics XML format. The MMCDB may also generate structured data as an
 10 output from the execution of the standardized encryptmatics XML application, and add
 11 the structured data to the centralized personal information platform databases, e.g.,
 12 3610. In some embodiments, the MMCDB may recursively provides structured data
 13 generated as a result of execution of the encryptmatics XML application as input into
 14 the EXC component, e.g. 3611.

15 **[00188]** FIGURE 37 shows a data flow diagram illustrating an example email data
 16 aggregation procedure in some embodiments of the MMCDB. In some
 17 implementations, the pay network server may obtain a trigger to extract one or more
 18 monitorable user email addresses and generate an email access API query in order to
 19 monitor a user's email activity and aggregate the content thereof. For example, the pay
 20 network server may periodically perform an update of its aggregated database, e.g.,
 21 3710a, with new information available from the user's email account activity operating
 22 on the Internet. In one embodiment, the information aggregated is the raw content of
 23 email messages, including header information containing the server delivery path
 24 through which the message has passed. As another example, a request for email data
 25 aggregation update may be obtained as a result of a user wishing to enroll in a service,
 26 for which the pay network server may facilitate data entry by providing an automated
 27 web form filling system using information about the user obtained from the email data
 28 aggregation update. In some implementations, the pay network server may parse the
 29 trigger to extract access credentials with which to perform an email data aggregation
 30 update. The pay network server may generate a query for application programming
 31 interface (API) templates for various email provider services (e.g., Gmail™, Yahoo

1 Mail™, etc.) from which to collect email data for aggregation. In some embodiments,
 2 the aggregation templates will be configured to provide access to the user's email
 3 account at the email service provider. In other embodiments, the aggregation templates
 4 will be configured to provide a mechanism to parse retrieved user email into a more
 5 suitable format for processing. In still other embodiments, the templates may indicate
 6 that an email transfer protocol (such as POP, IMAP, and/or the like) should be
 7 employed. In some instances, the email transfer protocol may be used over a secondary
 8 secured connection (such as SSH, PGP, and/or the like).

9 **[00189]** In one embodiment, the pay network server may query, e.g., 3712, a pay
 10 network database, e.g., 3707, for email aggregation API templates for the email provider
 11 services. For example, the pay network server may utilize PHP/SQL commands similar
 12 to the examples provided above. The database may provide, e.g., 3713, a list of email
 13 access API templates in response. Based on the list of API templates, the pay network
 14 server may generate email aggregation requests, e.g., 3714. The pay network server may
 15 issue the generated email aggregation requests, e.g., 3715a-c, to the email network
 16 servers, e.g., 3701a-c. For example, the pay network server may issue PHP commands
 17 to request the email provider servers for email data. An example listing of commands to
 18 issue email aggregation data requests 3715a-c, substantially in the form of PHP
 19 commands, is provided below:

```

20  <?php
21
22  $aggregated_email = "";
23  $mail = imap_open('{server.com:110/pop3}',
24                  $user,
25                  $password);
26  $headers = imap_headers($mail);
27
28  for ($i=1; $i<=count($headers); $i++) {
29      $structure = imap_fetchstructure($mail, $i);
30      $structure_parts = $structure->parts;
31      $number_parts = count($structure_parts);
32      for ($j=0; $j<=$number_parts; $j++) {
33          $text = imap_fetchbody($mail,$i,$j);

```

```

1             $aggregated_email .=
2                 nl2br(htmlspecialchars($text))."<br>";
3         }
4     }
5
6     ?>

```

7 **[00190]** In some embodiments, the email provider servers may query, e.g., 3717a-c,
8 their databases, e.g., 3710a-c, for email aggregation results falling within the scope of
9 the email aggregation request. In response to the queries, the databases may provide
10 email data, e.g., 3718a-c, to the email provider servers. The email provider servers may
11 return the email data obtained from the databases, e.g., 3719a-c, to the pay network
12 server making the email aggregation requests. An example listing of email data 3719a-c,
13 substantially in the form of JavaScript Object Notation (JSON)-formatted data, is
14 provided below:

```

15     ["data":[
16         {"headers": "Delivered-To: MrSmith@gmail.com
17     Received: by 10.36.81.3 with SMTP1 id e3cs239nzb; Tue, 5 Mar 2020
18     15:11:47 -0800 (PST)
19     Return-Path:
20     Received: from mail.emailprovider.com (mail.emailprovider.com
21     [111.111.11.111]) by mx.gmail.com with SMTP id
22     h19si826631rnb.2005.03.29.15.11.46; Tue, 5 Mar 2020 15:11:47 -
23     0800 (PST)
24     Message-ID: <20050329231145.62086.mail@mail.emailprovider.com>
25     Received: from [11.11.111.111] by mail.emailprovider.com via
26     HTTP; Tue, 5 Mar 2020 15:11:45 PST
27     Date: Tue, 5 Mar 2020 15:11:45 -0800 (PST)
28     From: Mr Jones
29     Subject: Dinner at Patsy's this weekend?
30     To: Mr Smith",
31         "from_addr": "MrJones@gmail.com",
32         "from_name": "Mr Jones",
33         "to_addr": "MrSmith@gmail.com",
34         "subject": "Dinner at Patsy's this weekend?"

```

```

1      "date": "Tue, 5 Mar 2020 15:11:45 -0800 (PST)",
2      "msg_content": "Hey Frank,\n\nWould you like to meet at
3      Patsy's for dinner on Saturday night? Their chicken parm is as
4      good as my mom's (and that's pretty good!).\n\nRafael"
5  },
6  {
7      ...
8  }
9  ]]

```

10 **[00191]** In some embodiments, the pay network server may store the aggregated
 11 email data results, e.g., 3720, in an aggregated database, e.g., 3710a.

12 **[00192]** FIGURE 38 is a block diagram illustrating an example structure of a
 13 distributed linking node mesh, in one embodiment of the MMCDB. In one
 14 embodiment, the linking node mesh may be represented as a modified graph data
 15 structure that contains nodes for entities and edges that represent the associations
 16 between the nodes. In one embodiment, the nodes are actual observable entities, such
 17 as a user 3801, a business 3803, an item 3807, a review on an online web site, e.g., 3813,
 18 3816, and/or the like. The graph mesh may also contain deduced entities that have been
 19 inserted by the MMCDB as a result of the aggregation described herein, such as the
 20 aggregation of data from email, search queries, location aggregation, and/or the like.
 21 Non-limiting examples of deduced entity nodes that may be inserted into the graph
 22 mesh include a deduced item, i.e., 3810 or a deduced opportunity, i.e., 3805. A deduced
 23 item may be an item that the mesh determines exists based on scanning emails, e.g.,
 24 3809, to determine that a concept that occurs with some frequency in emails is
 25 associated with a concept that is not yet linked into the mesh graph. An example
 26 deduced item may be a user's mother's chicken parmesan 3810. In one embodiment,
 27 there may also be deduced opportunities added to the mesh graph, e.g. 3805. A
 28 deduced opportunity may be an opportunity that is determined based on aggregated
 29 transaction data, e.g., 3804. For example, through the use of aggregated transaction
 30 data it may be determined by the MMCDB that the price of a given set of items declines
 31 at a restaurant, e.g., 3803, during the week, e.g., 3805. This decline in pricing may then
 32 allow the MMCDB to determine that there exists a special weekday menu with lower

1 prices. In so doing, an opportunity for use by the mesh in providing recommendations
 2 or information to users may be created, e.g., 3805, in order to facilitate searching the
 3 mesh.

4 **[00193]** In one embodiment, the mesh graph may also contain service items, e.g.,
 5 3807, such as a restaurants chicken parmesan or other menu item. The service item and
 6 its link to the business 3803, e.g., 3806, 3808, may be determined using a forward web
 7 crawl (such as by crawling from a business home page to its menu pages), or by a
 8 reverse web crawl, such as by crawling using an Optical Character Recognition scanned
 9 menu forwarded through an email exchange and aggregated by an email aggregating
 10 component of the MMCDB.

11 **[00194]** In one embodiment, the mesh graph may additionally contain meta
 12 concepts, e.g., 3810, 3812, 3815. Meta-concepts are conceptual nodes added to the
 13 graph by MMCDB that define not a specific entity (such as a user or a business) nor a
 14 specific deduced entity (such as a deduced item or a deduced opportunity), but rather
 15 indicate an abstract concept to which many more nodes may relate. For example,
 16 through web crawling, e.g., 3814, or email aggregation, e.g., 3817, user reviews may be
 17 imported as nodes within the mesh graph, e.g., 3813, 3816. Nodes may be anonymous,
 18 e.g., 3813, linked to a specific user's friend (such as to provide specific user
 19 recommendations based on a social graph link), e.g., 3816, and/or the like. These
 20 reviews may be analyzed for positive concepts or words such as "delightful meal" or
 21 "highly recommended" and thereafter be determined by the MMCDB to be a positive
 22 review and linked to a mesh meta-concept of the kind positive review, e.g., 3815. In so
 23 doing, the MMCDB allows disparate aggregated inputs such as email aggregation data,
 24 location aggregation data, web crawling or searching aggregated data, and/or the like to
 25 be used to roll up concepts into conceptual units.

26 **[00195]** In one embodiment, these conceptual meta concepts, e.g., 3815, may be
 27 further linked to actual items, e.g., 3807. In so doing connections can be formed
 28 between real world entities such as actual reviews of items, to meta-concepts such as a
 29 positive review or beneficial location, and further linked to actual items as a location.
 30 Further meta-concepts may include activities such as dinner, e.g., 3812, a non-entity
 31 specific item (e.g., not a restaurant's chicken parmesan and not a mother's chicken

1 parmesan, but chicken parmesan as a concept), e.g., 3811. The connection of actual
 2 entity nodes with deduced entity nodes and meta-concept nodes allows the mesh to
 3 answer a virtually limitless number of questions regarding a given nodes connections
 4 and probable outcomes of a decision.

5 **[00196]** In one embodiment, nodes within the mesh graph are connected by edges
 6 that have no magnitude. In another embodiment, the edges themselves may have meta-
 7 data associated with them that enable faster or better querying of the mesh. Example
 8 meta data that may be stored at a graph edge include a relative magnitude of connection
 9 between nodes, path information regarding other nodes available from the edge, and/or
 10 the like. In still other embodiments, intermediate or link nodes, e.g., 3804, 3806, 3808,
 11 3814, 3817, 3809, may be inserted by the MMCDB into the mesh graph. These
 12 intermediate nodes may function as the equivalent of an edge, in that they may describe
 13 a relationship between two nodes. In one embodiment, the link nodes may contain
 14 information about the nodes that they connect to. In so doing, the number of nodes in
 15 the graph that need to be searched in order to find a given type, magnitude or value of
 16 connection may be reduced logarithmically. Additionally, the link nodes may contain
 17 data about how the relationship between the nodes it links was established, such as by
 18 indicating the connection was established via search aggregation, email aggregation,
 19 and/or the like.

20 **[00197]** In one embodiment, the distributed linking node mesh may be stored in a
 21 modified open source database such as Neo4j, OrientDB, HyperGraphDB, and/or the
 22 like. An example structure substantially in the form of XML suitable for storing a
 23 distributed linking node mesh is:

```

24 <mesh>
25   <nodes>
26     <node id="1" kind="entity" type="user">
27       <name="John Doe">
28     </node>
29     <node id="2" kind="entity" type="item">
30       <name="iPhone" />
31     </node>
32     <node id="3" kind="deduced_entity" type="business">
```

```

1      <name="Apple Computer" />
2      <attribute type="keyword" value="iPhone" />
3      <deduced_from value="frequency_keyword" />
4  </node>
5  <node>
6      ...
7  </node>
8 </nodes>
9 <link_nodes>
10     <linknode id="78" in_node="1" out_node="3" value="55" />
11     <linknode id="97" in_node="1" out_node="2" value="124" />
12     ...
13 </link_nodes>
14 <edges>
15     <edge from_node="1" to_node="78" />
16     <edge from_node="78" to_node="3" />
17     ...
18 </edges>
19 </mesh>

```

20 **[00198]** An example query suitable for querying a distributed linking node mesh is:

```

21  START user=node(5,4,1,2,3)
22  MATCH user-[:affinity]->"iphone"
23  WHERE entity.manufacturer =~ 'Apple.*', link.strength >= 40
24  RETURN user, user.affinity

```

25 **[00199]** In another embodiment, an example query suitable for querying a
 26 distributed linking node mesh is:

```

27  ##MODEL QUERY Language (JSON FORMAT)
28  {
29      1: {'LOWER': 100, 'BASETYPE': ['MODEL_001_001_00',
30  'MODEL_002_001_00', 'MODEL_003_001_00', 'MODEL_004_001_00'] ,
31  'attribute': 'WEIGHT', 'rule': 'NEAR', 'OP': 'PROX', 'type':
32  'TOKENENTITY', 'HIGHER': 100}
33  , 2: {'type': ['USER', 'MCC'], 'rule': 'FOLLOW'}

```

```

1   , 3: {'rule': 'RESTRICTSUBTYPE', 'BASETYPE': ['MODEL_001_001_00',
2     'MODEL_002_001_00', 'MODEL_003_001_00', 'MODEL_004_001_00']}]
3   }

```

4 **[00200]** FIGURES 39A-E are an example block diagram illustrating a distributed
5 linking node mesh search, in one embodiment, of the MMCDB. The graph presented in
6 Fig. 39A-E is similar to the graph presented in Fig. 38 but nodes of different type are
7 represented similarly for ease of understanding. In one embodiment, a user 3901 may
8 desire to find a good deal on dinner with friends at a restaurant nearby. The MMCDB
9 may be configured with a capability to extract sub-concepts from a natural form query
10 question, such as by natural language processing. Example tools suitable for this type of
11 processing may include OpenNLP, Graph Expression, FreeLing, and/or the like.

12 **[00201]** In one embodiment, the query portion relating to finding a good deal is
13 performed as a MMCDB search to arrive arrive at a result of a deduced opportunity for
14 lower prices during weekdays, e.g., 3902. The search may then progress to extract the
15 concept of a good deal merged with a restaurant nearby. Using an integrated location
16 capability of a user's device, the user's current location may additionally be provided to
17 the MMCDB for use in this portion of the query process, to produce a result containing a
18 deduced opportunity for lower prices (e.g., a "good deal") at a business nearby wherein
19 the lower prices are linked to the business nearby with a certain degree of weight, e.g.,
20 3903. In one embodiment, the search may progress to find results for the concept of a
21 dinner (e.g., meta-concept dinner 3904), which is itself linked through intermedia
22 nodes to the business found in the previous portion of the search, e.g., 3905. In one
23 embodiment, the search may then progress to find connections that indicate that the
24 user 3901 will like the restaurant, e.g., 3906, and that the user's friends will similarly
25 like the restaurant, e.g., 3907. The intermediate searches performed may be then
26 merged to produce a unitary result, e.g., 3908, for a restaurant meeting the full criteria.
27 In cases where no single entity meets all the criteria, the most important criteria to a
28 user may be first determined using its own MMCDB search, such as a search that
29 determines that a user 3901 has never traveled to a nearby popular location area for
30 dinner and therefore concluding that location is very important to the user. In one

1 embodiment, multiple results 3908 may be returned and ranked for acceptability to
2 both the user and his/her friends, enabling the user to then choose a preferred location.

3 **[00202]** FIGURE 39F shows an alternative embodiment of a distributed linking
4 node mesh search. Here, mesh user 3901 wants to determine the probability that a user
5 will buy Korean BBQ before a certain time, e.g., 3909. The distributed linking mesh
6 may be queried. For example, the user's previous purchases of Korean BBQ (e.g., 3910,
7 3911), may be linked to a meta-concept that indicates an affinity for Korean BBQ, e.g.,
8 3911. The affinity may be similarly linked to an entity indicating a purchase frequency
9 for Korean BBQ, e.g., 3912. Similarly, by aggregating data from the user's email
10 correspondence (i.e., calendar updates, and/or the like), the mesh may have an entity
11 representing the user's schedule including free time, e.g., 3913. Both the purchase
12 frequency 3912 and the user schedule 3913 may be linked to a mesh meta-concept of
13 free time, e.g., 3914, which may indicate that the entities are related when the user has
14 free time (e.g., the individual may be more likely to go for Korean BBQ when she is not
15 working). By querying the distributed linking node mesh for interrelations between
16 entities built from aggregation techniques (and deduced or input entities), a profile of
17 the user's future behavior may be similarly built. For example, if the user schedule
18 indicates that the user is free on both Wednesday and Thursday afternoons, and the
19 aggregated purchase history indicates an affinity to purchase Korean BBQ both on those
20 days (based on the purchase transaction entities) and when the user is free (based on
21 the meta-concept of free time), then a mesh search can return a probability based on the
22 respective weights of the constituent entity relationships with respect to the user.

23 **[00203]** FIGURES 40A-C are an example block diagram illustrating index creation
24 in a distributed linking node mesh, in one embodiment of the MMCDB. In one
25 embodiment, a non-indexed graph is exploded to form a chain of relationships from a
26 single entity's perspective, e.g., 4001. A furthest node graph traversal is then
27 performed, e.g., 4002, whereby the linking nodes are sequentially removed and replaced
28 with a single edge which has a magnitude of the connection between the two nodes, e.g.,
29 4002a, 4002b. A nearest node graph traversal may then be performed, e.g., 4003,
30 whereby the magnitude of links further from the nearest node is modified by a factor of
31 previous links. Modification proceed from nearest to furthest nodes, e.g., 4003a. In the

1 example illustrated, a modification is made to the second edge encountered to make its
 2 value as a relation of magnitude with User X incident on both the relationship of User X
 3 to Business Y and of Business Y to Deduced Opportunity L. This procedure may
 4 produce a flattened graph from a single entity's perspective, e.g., 4004. The graph may
 5 then be further modified to a single perspective indexed graph, e.g., 4005, to reduce the
 6 number of hops in the graph from a given entity to any other entity within the indexed
 7 graph, so as to advantageously speed searching of the graph from the entity's
 8 perspective, e.g., 4005a. In one embodiment, the output of similar indexes performed
 9 from other entity perspectives, e.g., 4006b, may be linked 4006c to the generated
 10 perspective 4006a. In so doing, the index may form a graph that simultaneously allows
 11 easy searching from the perspective of a single entity while maintaining connection
 12 between entities of different perspectives.

13 **[00204]** FIGURE 41 is an example block diagram illustrating aspects of an
 14 Encryptmatics XML converter component. In one embodiment, models may be input in
 15 a number of disparate language, e.g., 4101. Languages may include interpreted
 16 languages such as Python, PHP, and/or the like, e.g., 4102, intermediate compiled
 17 languages such as .Net, Java, and/or the like, e.g., 4103, compiled languages such as
 18 C++, COBOL, and/or the like, e.g., 4104. A user defined language may also be input,
 19 e.g., 4105. In one embodiment, the user defined language will be input with a language
 20 mapper, e.g., 4106 that defines a mapping of the user defined language's functions,
 21 methods, supported types, and/or the like to a language known by the MMCDB. In still
 22 other embodiments, a native meta-data language, e.g., 4107, may be input.

23 **[00205]** In one embodiment, languages other than a native meta-data language are
 24 passed to a meta-data language conversion component 4108, such as an Encryptmatics
 25 XML converter. The converter may convert the language to a meta-data language 4109.
 26 In one embodiment, the meta data language may describe data sources 4110 including a
 27 private data store (not available to the provided model), an anonymized data store that
 28 is based on the private data store (available to the provided model), and/or a public data
 29 store. In one embodiment, the meta-data language may be deconverted 4111 to produce
 30 data queries and model logic 4112 that is parseable by the MMCDB interpreter.

1 **[00206]** FIGURE 42 is an example logic flow illustrating input language loading by
 2 an Encryptmatics XML converter component, in one embodiment of a MMCDB. In one
 3 embodiment, an input language definition is received, e.g., 4201. The language
 4 definition may be a file containing information about the functions available within the
 5 input language. In one embodiment, the language definition is source code for the
 6 language to be loaded into the MMCDB. In still other embodiments, the language
 7 definition is an executable binary file suitable for the MMCDB to issue sample
 8 commands and receive output. In one embodiment, the current mesh language
 9 definition may be retrieved 4202. The mesh language may be an XML based meta-data
 10 language that allows the description of data sources, data source manipulations (both
 11 visible and not visible to the input model) and a model to be run against the data
 12 sources. A model may be a series of logic commands describing manipulations or
 13 conditional logic to apply to input or source data sources in order to reach a result. In
 14 one embodiment, language loading may facilitate the user providing the description of
 15 data sources, data source manipulations, the model, and/or the like in a language with
 16 which the user is already familiar. The MMCDB may then used a loaded language
 17 definition to convert the language to a common meta-data based (e.g., XML based,
 18 JSON based, and/or the like) language with which to then parse and execute commands
 19 from.

20 **[00207]** In one embodiment, the first unprocessed mesh language operation is
 21 extracted from the mesh language definition. An example operation may be “TRIM”,
 22 which may strip whitespace from the beginning and end of an input string. A
 23 determination is made if the mesh operation has an equivalent operation in the input
 24 language, e.g., 4204. Such a determination may be made by executing a sample
 25 command against the input binary and observing the output to determine if an error
 26 occurred. In other embodiments, a publically available language definition web site may
 27 be crawled to determine which function(s) within an input language likely map to the
 28 mesh operation equivalent(s). In some instances, there will be a one-to-one mapping
 29 between the input language and the meta-data based mesh language. If there is not a
 30 one-to-one equivalence, e.g., 4205, a determination is made (using a procedure similar
 31 to that employed above) to determine if a combination of input language functions may

1 equate to a mesh language operation, e.g., 4206. For example, an input language that
 2 supports both a left-trim (strip space to left of string) and a right-trim operation (strip
 3 space to right of string) may be considered to support a mesh TRIM through a
 4 combination applying both the left-trim and right-trim operations, producing a
 5 substantially equivalent output result.

6 **[00208]** In one embodiment, if no matching combination is found, e.g., 4207, the
 7 mesh operation may be marked as unavailable for the input language, e.g., 4208 and the
 8 next unprocessed mesh operation may then be considered. If a matching combination is
 9 found, e.g., 4207, an upper bound test may be employed to test the upper bound
 10 behavior of the input language operation and compare that to the upper bound behavior
 11 of an equivalent mesh operation, e.g., 4209. For example, some languages may perform
 12 floating point rounding to a different degree of precision at upper bounds of input. By
 13 testing this case, a determination may be made if the equivalent input language function
 14 will produce output equivalent to the mesh operation at upper bounds. In one
 15 embodiment, a lower bound test may be employed to test the lower bound behavior of
 16 the input language operation and compare that to the lower bound behavior of an
 17 equivalent mesh operation, e.g., 4210. For example, some languages may perform
 18 floating point rounding to a different degree of precision at lower bounds of input. By
 19 testing this case, a determination may be made if the equivalent input language function
 20 will produce output equivalent to the mesh operation at upper bounds. In one
 21 embodiment, other custom tests may then be performed that may be dependent on the
 22 mesh operation or the input language operation(s), e.g., 4211. If the results of the test
 23 cases above produce output that is different than the expected output for the equivalent
 24 mesh operation, e.g., 4212, an offset spanning function may be generated to span the
 25 difference between the languages. For example, in the example above if the rounding
 26 function in the input language is determined to produce different behavior than the
 27 equivalent mesh operation at a lower bound, a function may be provided in the input or
 28 mesh language to modify any output of the given input language operations to create an
 29 equivalent mesh language operation output. For example, a given floating point number
 30 may be rounded to a given level of significant digits to produce equivalent behavior.

1 **[00209]** In one embodiment, the offset spanning function may not be capable of
 2 completely mapping the input language operation(s) to the mesh language operation,
 3 e.g., 4214. In one embodiment, previous versions of the mesh language definition, e.g.,
 4 4215, may then be tested using a procedure substantially similar to that described above
 5 to determine if they may completely map the input language, e.g., 4216. If the previous
 6 version of the mesh language definition completely maps the input language, the mesh
 7 language definition version for the input language may be set to the previous version,
 8 e.g., 4217. For example, a previous version of the mesh language definition may contain
 9 different capabilities or function behaviors that allow it to completely map to an input
 10 language. If previous versions of the mesh input language do not completely map to the
 11 input language, language clipping parameters may be generated, e.g., 4218. Language
 12 clipping parameters are input limitations that are placed on an input language such that
 13 any inputs within the input limitations range will produce compliant mesh operation
 14 output. Inputs outside that range may generate an error. In one embodiment, language
 15 clipping parameters may include limits to the upper bound or lower bound of acceptable
 16 input. Such limits may be determined by iteratively testing increasing or decreasing
 17 inputs in order to find an input range that maps completely to the mesh operation.

18 **[00210]** In one embodiment, the current mesh operation, input language
 19 operation(s) any spanning functions or language clipping parameters, the mesh
 20 language version, and/or the like may be stored in an input language definition
 21 database, e.g., 4219. If there are more unprocessed mesh language operations, e.g.,
 22 4220, the procedure may repeat.

23 **[00211]** FIGURES 43A-B show an example logic flow for input model conversion,
 24 in one embodiment of an MMCDB. In one embodiment, a language command file is
 25 received, e.g., 4301. The language command file may contain instructions in any
 26 language which has been loaded into the MMCDB (e.g., Fig. 42). The input language
 27 command file may contain instructions that may describe a set of manipulations that
 28 may be performed on a data set (e.g., a data set that is input as part of the input
 29 language command file, a data set that is loaded from an external data source, and/or
 30 the like). In one embodiment, input language definitions corresponding to the language
 31 of the input language command file is retrieved, e.g., 4302. A mesh language definition,

1 which may specify operations that are available within the mesh language, may also be
 2 retrieved, e.g., 4303. Non-conditional logic flow blocks in the input language command
 3 file may be determined, e.g., 4304. A non-conditional logic block represents the
 4 outermost logic contained within an input language command file. For example, if a file
 5 contains no conditional logic (i.e., no if/then/else blocks, and/or the like), then the
 6 outermost logic may be the complete set of input language commands themselves. In
 7 one embodiment, a run block is created for each outermost non-conditional logic flow
 8 block. The metadata run blocks are then populated with logic commands further in the
 9 procedure. In one embodiment, any variables that are initialized within the logic block
 10 corresponding to the run block are determined, e.g., 4306. A variable initialization
 11 template may then be determined, e.g., 4307. In one embodiment, the input language
 12 definition is used to determine if an equivalent meta-data based variable type is
 13 available in the mesh language definition for each of the variables initialized in the input
 14 language command file, e.g., 4308. If all variable types are not available, a model input
 15 error may be raised, e.g., 4309.

16 **[00212]** In one embodiment, the variable initialization template and the input
 17 language definition are used to create a constants block based on the variable
 18 initialization template, e.g., 4310. Within the constants block, any constants that were
 19 included in the input language command file may be stored as structured XML. An
 20 example constants block, substantially the form of XML is as follows:

```
21   <constant>
22     indexname="0"
23     value='row by row'
24     Type="string"
25   </constant>
```

26 **[00213]** In one embodiment, there may be multiple constant blocks defined
 27 corresponding to multiple constant values in the input language command file. In other
 28 embodiments, constants may be collapsed to one block.

29 **[00214]** In one embodiment, the input datasources may then be determined based
 30 on the input language command file, e.g., 4311. For example, an input datasource may

1 be defined directly in the input language command file (such as by declaring a variable
 2 as an array to values in the input language command file). In other embodiments, the
 3 inputs may be external to the input language command file, such as a third party library
 4 or loaded from an external source file (such as a comma delimited file, via a SQL query
 5 to an ODBC compliant database, and/or the like). A mesh language input datasource
 6 template may then be retrieved, e.g., 4312, to provide a structure to the MMCDDB to use
 7 in formatting the inputs as meta-data. The datasources may be scanned to determine if
 8 they are available to the model (such as by executing “ls -l” on a POSIX compliant Unix
 9 system), e.g., 4313. If the datasources are available to the model, then a meta data
 10 language input block may be created using the input datasource template, the language
 11 definition, and the input language command file, e.g., 4314. An example input block
 12 substantially in the form of XML is:

```
13 <input>
14     keyname="test_by"
15     file="<ecyptmatics install>/test_by.egd"
16     format="ecdataformat"
17     meta_data={'col8': (7, 0, 'string'), '__header': True, 'col_2':
18 (1,0,'int'), 'col_3': (2,0,'int'), 'col_1': (0,0,'int'), 'col_6':
19 (5,0,'julian'), 'col_7': (6,0,'float'), 'col_4':
20 (3,0,'ordinaldate'), 'col_5': (4,0,'date')}]
21 </input>
```

22 **[00215]** In one embodiment, a mesh language output template is determined, e.g.,
 23 4315 and an output block is created using a procedure substantially similar to that
 24 described above with respect to the constant and input blocks, e.g., 4316. An example
 25 output block, substantially in the form of XML is:

```
26 <output>
27     keyname="myout"
28     file="stdout"
29     format="delimited"
30     meta_data={'col_2': (2, 0, 'String'), 'col_2_1': (4, 0,
31 'String'), 'test': (0, 0, 'String'), 'col_3': (3, 0, 'String'),
32 'col_1': (1, 0, 'String'), 'sum_col_7': (5, 0, 'String')}
```

```

1      delimiter= "csv"
2      </output>

```

3 **[00216]** In one embodiment, the constant block, input block, and output block are
 4 added to a newly created initialization block and the initialization block is added to the
 5 current run block, e.g., 4317. An example run block with a complete initialization block
 6 included therein, substantially in the form of XML is as follows:

```

7      <run>
8      <init>
9      processor=process
10     <input>
11         keyname="test_by"
12         file="<ecryptmatics install>/test/data/test_by.egd"
13         format="ecdataformat"
14         delimiter="csv"
15         meta_data={'col_8': (7, 0, 'string'), '__header': True,
16 'col_2': (1, 0, 'int'), 'col_3': (2, 0, 'int'), 'col_1': (0, 0,
17 'int'), 'col_6': (5, 0, 'julian'), 'col_7': (6, 0, 'float'),
18 'col_4': (3, 0, 'ordinaldate'), 'col_5': (4, 0, 'date')}
19     </input>
20     <output>
21         keyname="myout"
22         file="stdout"
23         format="delimited"
24         meta_data={'col_2': (2, 0, 'String'), 'col_2_1': (4, 0,
25 'String'), 'test': (0, 0, 'String'), 'col_3': (3, 0, 'String'),
26 'col_1': (1, 0, 'String'), 'sum_col_7': (5, 0, 'String')}
27         delimiter= "csv"
28     </output>
29     <constant>
30         indexname="0"
31         value='row by row'
32         type="string"
33     </constant>
34 </init>
35 </run>

```

1 **[00217]** In one embodiment, a vault block will then be created, e.g., 4318. A logic
 2 command block will be extracted from the input logic command file, e.g., 4319. A logic
 3 command block is a logic block that is a non-outermost non-conditional logic flow. A
 4 door block may then be added to the vault block, e.g., 4320. A logic command,
 5 representing a discrete logic operation, may then be extracted from the logic command
 6 block, e.g., 4321. The logic command may be a tumbler, e.g., 4322, in which case a
 7 tumbler key may be looked up in a tumbler database and the tumbler may be processed,
 8 e.g., 4323. Further detail with respect to tumbler processing may be found with respect
 9 to Figs. 44-45. The logic command may then be mapped to a mesh language equivalent
 10 by using the language definition file, e.g., 4324. A mesh template logic command
 11 template, containing formatting information for a logic command, may be retrieved,
 12 e.g., 4325. In one embodiment, a lock block may be created using the mesh language
 13 definition, the language definition, and the logic command, e.g., 4326. The created lock
 14 block may be added to the current door block, e.g., 4327. In one embodiment, if there
 15 are more logic commands, e.g., 4328, the procedure may continue. If there are more
 16 logic command blocks, e.g., 4329, the procedure may similarly continue. In one
 17 embodiment, if there are more outermost non-conditional logic flow blocks in the input
 18 language command file, e.g., 4330, the procedure may continue with respect to Fig. 43A.

19 **[00218]** FIGURE 44 is an example block diagram illustrating a tumbler data source
 20 manipulation and anonymization component, e.g., a TDS Component. In one
 21 embodiment, a user model may call a tumbler as part of a logic command block
 22 processing (e.g., in order to perform a hash table lookup, to provide third-party data, to
 23 import anonymized transaction data, and/or the like). In one embodiment, portions of
 24 the data manipulation may not be visible, e.g., 4401, to the user model in order to
 25 maintain privacy for the record owners, to preserve business secrets, and/or the like. In
 26 one embodiment, the data source to be anonymized is loaded into a key/value table, e.g.,
 27 4402. The entire matrix may be considered as a tumblar key. In other embodiments, a
 28 single cell within the matrix may be a tumblar key. In still other embodiments, the
 29 matrix may take the form of a $n \times n$ matrix of arbitrary size (e.g., a $4 \times 4 \times 4 \times 4$ matrix,
 30 and/or the like) In one embodiment, the keys or values may be pointers to underlying
 31 data records. In another embodiment, the keys or values may themselves be the data for

1 manipulation. Commands may be read from the tumblar file (which may, in some
 2 embodiments, have a format substantially similar to an input language command file,
 3 e.g., 4301). The commands may change some values in the matrix to other values, such
 4 as may be done to anonymize user payment card information, e.g., 4403. In other
 5 embodiments, data may be removed from the matrix and replaced with other data
 6 values, e.g., 4404. When indicated by the tumblar file, when a set number such as 5
 7 anonymization operations have been performed, or when the tumblar key has reached a
 8 certain value, the tumblar key may be considered visible to the user model, e.g., 4405.
 9 In so doing, the current keychain may be visible to the user model, e.g., 4407.
 10 Additional operations may then be performed on the key, extending the keychain, e.g.,
 11 4408. A keychain is a representation of current and past values of a key/value store. In
 12 one embodiment, the keychain 4409 may be returned. The keychain may contain an $n \times$
 13 n sized matrix (i.e., a single 2D matrix, a 3D collection of 2D matrix, a 4D matrix, and/or
 14 the like), e.g., 4409a, 4409b.

15 **[00219]** In one embodiment, a tumblar file may be substantially in the form of
 16 XML as follows:

```

17   <xml>
18   <run>
19   <init>
20     processor=process
21     tumblar_name=None
22     tumblar_path=c:\lm\Ecryptmatics\ecryptmatics\test\tumblars\flare
23     tumblarkey=flare
24     <input>
25       keyname="flares"
26       file="<ecryptmatics install>/test/data/flare.data1"
27       format="delimited"
28       delimiter=" "
29       meta_data={'evolution': (4, 0, 'Evolution'),
30 'prev_activity': (5, 0, 'Previous 24 hour flare activity code'),
31 'area': (8, 0, 'Area'), 'are_largest_spot': (9, 0, 'Area of the
32 largest spot'), 'histocially_complex': (6, 0, 'Historically-
33 complex'), 'complex': (7, 0, "Did region become historically
```

```

1  complex  on this pass across the sun's disk"), 'class_cd': (0, 0,
2  'Code for class (modified Zurich class)'), 'activity': (3, 0,
3  'Activity'), 'spot_dict_cd': (2, 0, 'Code for spot
4  distribution'), 'largests_spot_cd': (1, 0, 'Code for largest spot
5  size'), '__header': False}
6      </input>
7      <output>
8      keyname="myout "
9      file="stdout "
10     format="delimited"
11     meta_data={'evolution': (3, 0, 'String'), 'area': (1, 0,
12 'String'), 'complex': (2, 0, 'String'), 'activity': (0, 0,
13 'String')})
14     delimiter= "csv"
15     </output>
16 </init>
17 <vault>
18     <door>
19         <lock>
20             outkey="myout "
21             outkeyname="activity"
22             inkey="flares"
23             inkeyname="activity"
24             function="tumble"
25             type="String"
26             tumblar-masks="*"
27             fnc-tumblar-key-table="flare.activity"
28             tumblar-default="None"
29         </lock>
30         <lock>
31             outkey="myout "
32             outkeyname="area"
33             inkey="flares"
34             inkeyname="area"
35             function="tumble"
36             type="String"
37             tumblar-masks="*"

```

```

1      fnc-tumblr-key-table="flare.area"
2      tumblr-default="None"
3      </lock>
4      <lock>
5      outkey="myout "
6      outkeyname="complex"
7      inkey="flares"
8      inkeyname="complex"
9      function="tumble"
10     type="String"
11     tumblr-masks="*"
12     fnc-tumblr-key-table="flare.complex"
13     tumblr-default="None"
14     </lock>
15     <lock>
16     outkey="myout "
17     outkeyname="evolution"
18     inkey="flares"
19     inkeyname="evolution"
20     function="tumble"
21     type="String"
22     tumblr-masks="*"
23     fnc-tumblr-key-table="flare.evolution"
24     tumblr-default="None"
25     </lock>
26 </door>
27 </vault>
28 </run>
29 </xml>

```

30 **[00220]** FIGURE 45 is an example logic flow showing a tumblr data source
31 anonymization component, e.g., a TDS component, in one embodiment of a MMCDB.
32 In one embodiment, a user unaccessible data source request and a user generated model
33 containing tumblr data source manipulations may be received, e.g., 4501. In one
34 embodiment, a tumblr key may be extracted, e.g., 4502. A tumblr definition
35 corresponding to the tumblr key may be retrieved from a tumblr database, e.g., 4503.

1 A tumblar definition may contain manipulations (e.g., functions, methods, and/or the
 2 like) that may be performed on a given source file before the data is made available for
 3 use in a user model. In one embodiment, an input/starting key name may be
 4 determined (e.g., by inspecting an init block or by inspecting the input key values in the
 5 first lock of the first door of the first vault in the first run block in the tumblar file), e.g.,
 6 4505. An unprocessed internal tumblar data operation may be extracted including an
 7 input and an output key, e.g., 4506. An internal tumblar operation may be an operation
 8 that is performed before a user model has access to the data store, such as data
 9 manipulations that anonymize data. Manipulation operations may include bit shifting,
 10 replacing or masking certain field values in the data set, swapping data values between
 11 records (such as may be done to maintain a total of all values or the average of all values
 12 while not revealing to the user model the underlying data). In one embodiment, the
 13 current map located at the input key may be duplicated and stored, e.g., 4507. The
 14 operation may then be performed on the data copy, e.g., 4508. In so doing, a chain (e.g.,
 15 a key chain) of values may be created for a single data point. If the current output key is
 16 visible to the user model (such as if the output key is > a given value such as 31, the
 17 output has undergone a given number of operations, and/or the like), e.g., 4509, then
 18 any maps equal to or greater than the current map may be marked as visible to the user
 19 model, e.g., 4510. Manipulation operations may continue on the data and an
 20 unprocessed external tumblar data operation (e.g., an operation visible to the user
 21 model) may be extracted, e.g., 4511. The current map may be duplicated, e.g., 4512, and
 22 stored as a new map also visible to the user model, e.g., 4512. In one embodiment, the
 23 external tumblar data operation may then be applied to the copied map, e.g., 4513. If
 24 there are no more processed external tumblar data operations, e.g., 4514, the user model
 25 visible portion of the keychain may be returned, e.g., 4515.

26 **[00221]** FIGURE 46 is an example data flow illustrating mesh aggregation and
 27 cluster querying, in one embodiment of a MMCDB. In one embodiment, a firehose
 28 server 4601 provides firehose input, e.g., 4602 to a mesh server 4603. A firehose server
 29 may be a server capable of accepting input from one or more data sources (e.g., Twitter
 30 or Facebook posts, transaction records, email data, and/or the like) at a high relative
 31 flow rate. In one embodiment, the firehose server may perform some manipulations on

1 the received data before it is input to the mesh server 4603. An example firehose input
 2 4602, substantially in the form of XML formatted data is:

```

3   <firehose_input>
4     <input type="email" id="1">
5       <dictionary_entry>
6         {id: "1h65323765gtyuf#uy76355",
7           type: email,
8           category: {cat1: "food", cat2: "dinner"},
9           from_addr: "john.doe@gmail.com",
10          to_addr: "jane.doe@gmail.com",
11          subject: "Korean BBQ this weekend?",
12          dictionary_keywords: "Korean, dinner, nyc",
13          content_hash: "7m865323476feeaniiji"}
14       </dictionary_entry>
15       <datetime>Jan 20, 2020 15:23:43 UTC</datetime>
16       <from_addr>john.doe@gmail.com</from_addr>
17       <to_addr>jane.doe@gmail.com</to_addr>
18       <subject>Korean BBQ this weekend?</subject>
19       <content>
20 Received: by 10.36.81.3 with SMTP1 id e3cs239nzb; Tue, 5 Mar 2020
21 15:11:47 -0800 (PST)
22 Return-Path:
23 Received: from mail.emailprovider.com (mail.emailprovider.com
24 [111.111.11.111]) by mx.gmail.com with SMTP id
25 h19si826631rnb.2005.03.29.15.11.46; Tue, 5 Mar 2020 15:11:47 -
26 0800 (PST)
27 Message-ID: <20050329231145.62086.mail@mail.emailprovider.com>
28 Received: from [11.11.111.111] by mail.emailprovider.com via
29 HTTP; Tue, 5 Mar 2020 15:11:45 PST
30 Date: Tue, 5 Mar 2020 15:11:45 -0800 (PST)
31 From: John Doe <john.doe@gmail.com>
32 Subject: Korean BBQ this weekend?
33 To: Jane Doe <jane.doe@gmail.com>
34 Hi Jane,
35 Would you like to meet up in New York city this weekend for
36 Korean BBQ? I know this great place down on Spring Street.
```

```

1   John
2       </content>
3   </input>
4   <input type="tweet" id="2">
5       ...
6   </input>
7   <input type="purchase_transaction" id="3">
8       ...
9   </input>
10  <input type="web_search" id="4">
11      ...
12  </input>
13  <input id="n">
14      ...
15  </input>
16  </firehost_input>

```

17 **[00222]** In one embodiment, the mesh structure may then be updated, e.g., 4604.
 18 Further detail regarding updating the mesh structure can be found throughout this
 19 specification, drawing and claims, and particularly with reference to Figs. 15-19. In one
 20 embodiment, a clustering node 4605 may send a cluster categories request 4606 to the
 21 mesh server. A cluster categories request may contain a category or deduced concept
 22 that is to be added to the mesh. In one embodiment, the category may have no pre-
 23 existing associations in the mesh (e.g., the category to be added may be an orphan
 24 category). An example cluster categories request 4606, substantially in the form of an
 25 HTTP(S) POST message including XML is:

```

26  POST /cluster_categories.php HTTP/1.1
27  Host: www.meshserver.com
28  Content-Type: Application/XML
29  Content-Length: 667
30  <?XML version = "1.0" encoding = "UTF-8"?>
31  <cluster_categories_request>
32      <cluster operation="add">
33          <concept value="iphone" />
34          <concept_related_concept value="apple" />

```

```

1      <concept_keyword value="64GB" />
2      <concept_keyword value="Steve Jobs" />
3  </cluster>
4  <cluster>
5      ...
6  </cluster>
7  </cluster_categories_request>

```

8 **[00223]** In an alternative embodiment, an example cluster categories request
9 4606, substantially in the form of an HTTP(S) POST message including XML is:

```

10 POST /cluster_categories.php HTTP/1.1
11 Host: www.meshserver.com
12 Content-Type: Application/XML
13 Content-Length: 667
14 <?XML version = "1.0" encoding = "UTF-8"?>
15 <cluster_categories_request>
16     <cluster operation="add">
17         <concept value="portable music player" />
18         <manufacturer>Apple Computer</manufacturer>
19         <model>iPod Touch 32GB</model>
20         <size>32GB</size>
21     </cluster>
22     <cluster>
23         ...
24     </cluster>
25 </cluster_categories_request>

```

26 **[00224]** In one embodiment, the cluster categories request above may be modified
27 by the MMCDDB as a result of aggregated data. For example, a request to create a cluster
28 for an iPod of a given size may be supplemented with alternative models / sizes. In so
29 doing, the mesh may expand a recommendation, graph entity, and/or the like to
30 encompass concepts that are connected with the primary concept. In one embodiment,
31 this modified cluster may take the form a the form of XML substantially similar to:

```

32     <cluster>
33         <concept value="portable music player" />

```

```

1      <manufacturer>Apple Computer</manufacturer>
2      <model>
3          <1>iPod Touch 32GB</1>
4          <2>iPod Touch 64GB</2>
5          <3>iPod Touch 128GB</3>
6          <4>iPhone 32GB</4>
7          <5>iPhone 64GB</5>
8          <6>iPhone 128GB</6>
9      </model>
10     <size>32GB OR 64GB OR 128GB</size>
11 </cluster>

```

12 **[00225]** In one embodiment, the mesh structure may be updated in response to the
13 cluster categories request, e.g., 4604. In one embodiment, a user 4607 may use his/her
14 mobile device to indicate that they wish to purchase an item based on cluster concepts,
15 e.g., a user bid/buy input 4608. For example, a user may query “I want the TV that AV
16 Geeks thinks is best and I’ll pay \$1,500 for it”. In one embodiment, the query may be
17 substantially in the form of a language input such as the above, which may be parsed
18 using natural language processing packages such as FreeLing, LingPipe, OpenNLP,
19 and/or the like. In other embodiments, the user may be presented with a structured
20 query interface on their mobile device that allows a restricted set of options and values
21 from which to build a bid/buy input 4608. For example, a user may be given a list of
22 categories (such as may be built by querying a categories database as described with
23 respect to Fig. 49) from which to choose when making a bid/buy input. In one
24 embodiment, a clustering server 4609 may receive the user bid/buy input 4608 and
25 generate a consumer cluster based bid request, e.g., 4610 and provide same to a
26 clustering node 4605. An example consumer cluster based bid request 4610,
27 substantially in the form of an HTTP(S) POST message including XML is:

```

28 POST /consumer_bid_request.php HTTP/1.1
29 Host: www.meshserver.com
30 Content-Type: Application/XML
31 Content-Length: 667
32 <?XML version = "1.0" encoding = "UTF-8"?>
33 <consumer_cluster_based_bid_request>

```

```

1      <datetime>Jan 21, 2020 5:34:09 UTC</datetime>
2      <user_id>43246</user_id>
3      <request>
4          <type>bid</type>
5          <item>
6              <item_query>LCD Television</item_query>
7              <type_desired value="best" />
8              <cluster_source value="AV Geeks.com" />
9              <cluster_min_expertise_level value="top2prct" />
10             <max_price value="1500.00" currency="USD" />
11             <expire_request value="30days" />
12             <payment type="credit">
13                 <card_type>VISA</card_type>
14                 <card_num>98765436598766732</card_num>
15                 <card_exp>0525</card_exp>
16             </payment>
17             <shipping_address>
18                 <addr1>100 Main St.</addr1>
19                 <city>Anytown</city>
20                 <state>CA</state>
21                 <zip>90145</zip>
22             </shipping_address>
23         </item>
24     </request>
25 </consumer_cluster_based_bid_request>

```

26 **[00226]** In an alternative embodiment, the consumer cluster based bid request
27 may be generated using the user interface described herein and with respect to Fig. 48A-
28 B. In one embodiment, the consumer cluster based bid request 4610, generated using
29 the interface may be substantially in the form of an HTTP(S) POST message including
30 XML:

```

31 POST /consumer_bid_request.php HTTP/1.1
32 Host: www.meshserver.com
33 Content-Type: Application/XML
34 Content-Length: 667
35 <?XML version = "1.0" encoding = "UTF-8"?>

```

```

1   <consumer_cluster_based_bid_request>
2     <datetime>Jan 21, 2020 5:34:09 UTC</datetime>
3     <user_id>43246</user_id>
4     <request>
5       <type>bid</type>
6       <item>
7         <item_query>headphones</item_query>
8         <quantity value="2" />
9         <requirement value="rated_top_3" />
10        <cluster_source value="consumerreports.com" />
11        <max_price value="249.95" currency="USD" />
12        <expire_request value="January 15, 2020" />
13        <payment type="credit">
14          <card_type>VISA</card_type>
15          <card_num>98765436598766732</card_num>
16          <card_exp>0525</card_exp>
17        </payment>
18        <shipping_address>
19          <addr1>100 Main St.</addr1>
20          <city>Anytown</city>
21          <state>CA</state>
22          <zip>90145</zip>
23        </shipping_address>
24      </item>
25    </request>
26  </consumer_cluster_based_bid_request>

```

27 **[00227]** In one embodiment, in response to the consumer cluster based bid request
 28 4610, the clustering node 4605 may generate a cluster request 4611. A cluster request
 29 may be a request to search the mesh in order to find results (e.g., items matching a
 30 cluster's buying habits, merchants offering an item, alternative items for purchase,
 31 friends that have already purchased items, items the user already owns – based on, for
 32 example, past purchase transactions – that may satisfy the request, and/or the like). An
 33 example query suitable for querying a distributed linking node mesh is:

```

34   START user=node(5,4,1,2,3)

```

```

1 MATCH entity-[:affinity]->"consumer_reports"
2 WHERE entity.recommended >= '3',
3     entity.recommendation.item.type ~= "headphones"
4 RETURN entity.recommendation.item.name,
5     entity.recommendation.item.model,
6     entity.recommendation.item.averageprice

```

7 **[00228]** In one embodiment, the mesh server may provide a cluster request
8 response 4612. An example cluster request response 4612 substantially in the form of
9 an HTTP(S) POST message including XML is:

```

10 POST /cluster_request_response.php HTTP/1.1
11 Host: www.clusteringnode.com
12 Content-Type: Application/XML
13 Content-Length: 667
14 <?XML version = "1.0" encoding = "UTF-8"?>
15 <cluster_request_response>
16     <requested_item>
17         <item_query>LCD Television</item_query>
18         <type_desired value="best" />
19         <cluster_source value="AV Geeks.com" />
20         <cluster_min_expertise_level value="top2prct" />
21         <max_price value="1500.00" currency="USD" />
22     </requested_item>
23     <cluster_results>
24         <num_users_meeting_cluster value="2541" />
25         <average_user_feedback_ranking value="94%" />
26         <cluster_user_purchases>
27             <item rank="1">
28                 <desc>Sony Bravada 50" LCD 645</desc>
29                 <model>KDL50EX645</model>
30             </item>
31             <item rank="2">
32                 <desc>Sony Bravada 50" LCD 655</desc>
33                 <model>KDL50EX655</model>
34             </item>
35             <item>

```

```

1      ...
2      </item>
3      </cluster_user_purchases>
4  </cluster_results>
5  </cluster_request_response>

```

6 **[00229]** In an alterative embodiment, an example cluster request response 4612
7 substantially in the form of an HTTP(S) POST message including XML is:

```

8  POST /cluster_request_response.php HTTP/1.1
9  Host: www.clusteringnode.com
10 Content-Type: Application/XML
11 Content-Length: 667
12 <?XML version = "1.0" encoding = "UTF-8"?>
13 <cluster_request_response>
14   <requested_item>
15     <item_query>headphones</item_query>
16     <quantity value="2" />
17     <requirement value="rated_top_3" />
18     <cluster_source value="consumerreports.com" />
19     <max_price value="249.95" currency="USD" />
20     <expire_request value="January 15, 2020" />
21   </requested_item>
22   <cluster_results>
23     <cluster_consumer_reports_ranking>
24       <item consumer_reports_rank="1">
25         <desc>Panasonic Technics Pro DJ</desc>
26         <model>RP-DH1250</model>
27         <avg_price>$235.55</avg_price>
28       </item>
29       <item consumer_reports_rank="2">
30         <desc>Coby In Ear Headphones</desc>
31         <model>CVEM76PNK</model>
32         <avg_price>$245.55</avg_price>
33       </item>
34       <item consumer_reports_rank="3">
35         <desc>Shure E2c-n Sound Isolating Earphones</desc>

```

```

1         <model>SHE2CN</model>
2         <avg_price>$249.95</avg_price>
3     </item>
4     </cluster_consumer_reports_ranking>
5 </cluster_results>
6 </cluster_request_response>

```

7 **[00230]** In one embodiment, the clustering node 4605 may then process the cluster
8 response and create transaction triggers. Further details regarding cluster request
9 response 4612 processing may be found throughout the specification, drawings and
10 claims and particularly with reference to **Fig. 47**, e.g., a **CRA Component**.

11 **[00231]** In one embodiment, a lead cluster order request may be generated for
12 merchants that were identified as a result of the cluster response analysis, e.g., 4613. In
13 other embodiments, a default list of merchants may be used. A lead cluster order
14 request may contain information relating to the identified purchase that the user 4607
15 wishes to engage in. In the example above, for example, the analysis may have
16 determined that based on the aggregated AV Geeks user expert preference information,
17 the user should purchase Sony television model KDL50EX645 or KDL50EX655. The
18 analysis may also have determined that a given merchant sells those models of television
19 (such as by using aggregated sales transaction data as described herein). A request may
20 then be sent to the merchant indicating a purchase item, a user lead that may execute
21 the purchase and a price the user is willing to pay. In one embodiment, the user identity
22 is not provided or is anonymized such that the merchant does not have information
23 sufficient to determine the actual identity of the user but may determine if they wish to
24 execute the sale to the user. An example lead cluster order request 4614, substantially in
25 the form of an HTTP(S) POST message containing XML data:

```

26 POST /lead_cluster_order_request.php HTTP/1.1
27 Host: www.merchantserver.com
28 Content-Type: Application/XML
29 Content-Length: 667
30 <?XML version = "1.0" encoding = "UTF-8"?>
31 <lead_cluster_order_request>
32     <lead validFor="30_days">

```

```

1      <type>television</type>
2      <items join="OR">
3          <item model="KDL50EX645" />
4          <item model="KDL50EX655" />
5      </items>
6      <user_information>
7          <name>John Doe</name>
8          <email>john.doe@gmail.com</email>
9          <phone>865-765-3465</phone>
10     </user_information>
11     <payment type="credit">
12         <card_type>VISA</card_type>
13         <card_num>98765436598766732</card_num>
14         <card_exp>0525</card_exp>
15     </payment>
16     <shipping_address>
17         <addr1>100 Main St.</addr1>
18         <city>Anytown</city>
19         <state>CA</state>
20         <zip>90145</zip>
21     </shipping_address>
22 </lead>
23 <lead>
24     ...
25 </lead>
26 </lead_cluster_order_request>

```

27 **[00232]** In one embodiment, a merchant may accept the order and generate a lead
28 cluster order accept/reject response. In other embodiments, the merchant may indicate
29 that they wish to hold the lead opportunity open and may accept at a later time if no
30 other merchant has filled the lead cluster order request. In still other embodiments, the
31 merchant response may contain a counteroffer for the user (e.g., \$1600), which the user
32 may then accept or decline. In one embodiment, the user receives an order acceptance
33 confirmation 4617 indicating that their order has been fulfilled.

1 **[00233]** In one embodiment, a user may cancel a cluster based bid request prior to
 2 the merchant fulfilling the order. For example, a user may transmit a user cancel input
 3 4618 to clustering server 4609. The clustering server may forward the cancel message
 4 to the clustering node 4605, e.g., 4619, which may in turn forward the cancel message to
 5 the merchant(s) server 4615, e.g., 4620.

6 **[00234]** FIGURE 47 is an example logic flow illustrating cluster response analysis
 7 and transaction triggering, e.g., a CRA component, in one embodiment of a MMCDB. In
 8 one embodiment, a cluster request response is received, e.g., 4701. Cluster criteria (i.e.,
 9 user requesting cluster, the criteria for the cluster, payment/shipping information for
 10 the user purchase bid, and/or the like) may be extracted from the cluster request
 11 response, e.g., 4702. In one embodiment, the cluster criteria is examined to determined
 12 if it meets the minimum cluster criteria, e.g., 4703. Examples of minimum cluster
 13 criteria include minimum feedback ranking of users in cluster, minimum years of
 14 expertise of users in cluster, median value of items returned, and/or the like. If the
 15 cluster criteria is not greater than the minimum cluster criteria, the user may be
 16 prompted to adjust the minimum criteria and a search may be re-run, e.g., 4704. In
 17 other embodiments, the criteria may be adjusted automatically by the MMCDB or a
 18 third-party database may be queried to determine new minimum criteria (e.g., a user
 19 expertise ranking service, a user review site, and/or the like).

20 **[00235]** In one embodiment, candidate purchase items may be extracted from the
 21 cluster request response, e.g., 4705. A merchant database may be queried to determine
 22 merchants selling the candidate purchase items. An example merchant database query,
 23 substantially in the form of PHP/SQL commands is provided below:

```

24 <?PHP
25 header('Content-Type: text/plain');
26 mysql_connect("localhost",$DBserver,$password);
27 mysql_select_db("merchants.sql");
28 $query = "SELECT merchant_id, merchant_name, price,
29 quantity_on_hand FROM merchants WHERE merchant_item_id LIKE '%"
30 $cluster_returned_model_num";
31 $result = mysql_query($query); // perform the search query
32 mysql_close("merchants.sql"); // close database access

```

1 ?>

2 **[00236]** In one embodiment, a maximum price the user is willing to pay is
 3 determined, e.g., 4707. An average selling price of the candidate purchase items may be
 4 determine (such as by querying a merchant table containing price history, querying a
 5 price history table, performing a live crawl of a merchant's web site, and/or the like). If
 6 the user's maximum price is not within a given range of the average merchant item
 7 price, e.g., 4709, a price trend database may be queried, e.g., 4710. A price trend
 8 database may contain historical information relating to the price of an item over time. If
 9 the price trend (i.e., the linear extrapolation of the historical prices, and/or the like)
 10 shows that the average price of the item will be within 40% of the user's maximum price
 11 before the user purchase bid expires, e.g., 4711, the user purchase bid request may be
 12 held, e.g., 4712, so that the cluster response analysis may be re-run again before the bid
 13 expires. In another embodiment, even if the user's price will not be within a range of the
 14 average price of an item at the queried merchants, the user procedure may continue if
 15 the user has been marked as a high priority bid user (e.g., a frequent bidder, a new
 16 bidder, and/or the like), e.g., 4713. In one embodiment, the first merchant that has
 17 stock of the item may be selected, e.g., 4714. If the merchant has received greater than a
 18 set amount of bids in a time period, e.g., 4715, another merchant may be selected. In so
 19 doing, one merchant may not be overwhelmed with bids. In one embodiment, a lead
 20 cluster order request is created and transmitted to the merchant, e.g., 4716.

21 **[00237]** FIGURES 48A-C show user interface diagrams illustrating example
 22 aspects of a discovery shopping mode of a virtual wallet application in some
 23 embodiments of the MMCDB. In some embodiments, the virtual wallet application may
 24 provide a 'discovery shopping' mode for the user. For example, the virtual wallet
 25 application may obtain information on aggregate purchasing behavior of a sample of a
 26 population relevant to the user, and may provide statistical/aggregate information on
 27 the purchasing behavior for the user as a guide to facilitate the user's shopping. For
 28 example, with reference to FIGURE 48A, the discovery shopping mode 4801 may
 29 provide a view of aggregate consumer behavior, divided based on product category (see
 30 4802). Thus, the virtual wallet application may provide visualization of the magnitude of

1 consumer expenditure in particular market segment, and generate visual depictions
2 representative of those magnitudes of consumer expenditure (see 4803-4806). In some
3 embodiments, the virtual wallet application may also provide an indicator (see 4809) of
4 the relative expenditure of the user of the virtual wallet application (see blue bars); thus
5 the user may be able to visualize the differences between the user's purchasing behavior
6 and consumer behavior in the aggregate. The user may be able to turn off the user's
7 purchasing behavior indicator (see 4810). In some embodiments, the virtual wallet
8 application may allow the user to zoom in to and out of the visualization, so that the user
9 may obtain a view with the appropriate amount of granularity as per the user's desire
10 (see 4807-4808). At any time, the user may be able to reset the visualization to a default
11 perspective (see 4811).

12 **[00238]** Similarly, the discovery shopping mode 4821 may provide a view of
13 aggregate consumer response to opinions of experts, divided based on opinions of
14 experts aggregated form across the web (see 4802). Thus, the virtual wallet application
15 may provide visualizations of how well consumers tend to agree with various expert
16 opinion on various product categories, and whose opinions matter to consumers in the
17 aggregate (see 4823-4826). In some embodiments, the virtual wallet application may
18 also provide an indicator (see 4829) of the relative expenditure of the user of the virtual
19 wallet application (see blue bars); thus the user may be able to visualize the differences
20 between the user's purchasing behavior and consumer behavior in the aggregate. The
21 user may be able to turn off the user's purchasing behavior indicator (see 4830). In
22 some embodiments, the virtual wallet application may allow the user to zoom in to and
23 out of the visualization, so that the user may obtain a view with the appropriate amount
24 of granularity as per the user's desire (see 4827-4828). At any time, the user may be able
25 to reset the visualization to a default perspective (see 4831).

26 **[00239]** With reference to FIGURE 48B, in some implementations, the virtual
27 wallet application may allow users to create targeted shopping rules for purchasing (see
28 FIGURE 48A, 4812, 4822). For example, the user may utilize the consumer aggregate
29 behavior and the expert opinion data to craft rules on when to initiate purchases
30 automatically. As an example, rule 4841 specifies that the virtual wallet should sell the
31 users iPad2 if its consumer reports rating falls below 3.75/5.0, before March 1, provided

1 a sale price of \$399 can be obtained. As another example, rule 4842 specifies that the
 2 virtual wallet should buy an iPad3 if rule 4841 succeeds before February 15. As another
 3 example, rule 4843 specifies that the wallet should buy a Moto Droid Razr from the
 4 Android Market for less than \$349.99 if its Slashdot rating is greater than 3.75 before
 5 February 1. Similarly, numerous rules with a wide variety of variations and
 6 dependencies may be generated for targeted shopping in the discovery mode. In some
 7 implementations, the virtual wallet user may allow the user to modify a rule. For
 8 example, the wallet may provide the user with an interface similar to 4846 or 4847. The
 9 user may utilize tools available in the rule editor toolbox to design the rule according to
 10 the user's desires. In some implementations, the wallet may also provide a market status
 11 for the items that are subject to the targeted shopping rules.

12 **[00240]** With reference to FIGURE 48C, in some implementations, the virtual
 13 wallet application may provide a market watch feature, wherein the trends associated
 14 with items subject to targeted shopping rules may be tracked and visually represented
 15 for the user. For example, the visualization may take, in some implementations, the
 16 form of a ticker table, wherein against each item 4851(A)-(E) are listed a product
 17 category or cluster of expert opinions to which the product is related 4852, pricing
 18 indicators, including, but not limited to: price at the time of rule creation 4852, price at
 19 the time of viewing the market watch screen 4853, and a target price for the items (A)-
 20 (E). Based on the prices, the market watch screen may provide a trending symbol (e.g.,
 21 up, down, no change, etc.) for each item that is subject to a targeted shopping rule.
 22 Where an item satisfied the targeted rule (see item (E)), the virtual wallet may
 23 automatically initiate a purchase transaction for that item once the target price is
 24 satisfied.

25 **MMCDB Controller**

26 **[00241]** FIGURE 49 shows a block diagram illustrating embodiments of a MMCDB
 27 controller. In this embodiment, the MMCDB controller 4901 may serve to aggregate,
 28 process, store, search, serve, identify, instruct, generate, match, and/or facilitate
 29 interactions with a computer through various technologies, and/or other related data.

1 **[00242]** Typically, users, which may be people and/or other systems, may engage
 2 information technology systems (e.g., computers) to facilitate information processing.
 3 In turn, computers employ processors to process information; such processors 4903
 4 may be referred to as central processing units (CPU). One form of processor is referred
 5 to as a microprocessor. CPUs use communicative circuits to pass binary encoded signals
 6 acting as instructions to enable various operations. These instructions may be
 7 operational and/or data instructions containing and/or referencing other instructions
 8 and data in various processor accessible and operable areas of memory 4929 (e.g.,
 9 registers, cache memory, random access memory, etc.). Such communicative
 10 instructions may be stored and/or transmitted in batches (e.g., batches of instructions)
 11 as programs and/or data components to facilitate desired operations. These stored
 12 instruction codes, e.g., programs, may engage the CPU circuit components and other
 13 motherboard and/or system components to perform desired operations. One type of
 14 program is a computer operating system, which, may be executed by CPU on a
 15 computer; the operating system enables and facilitates users to access and operate
 16 computer information technology and resources. Some resources that may be employed
 17 in information technology systems include: input and output mechanisms through
 18 which data may pass into and out of a computer; memory storage into which data may
 19 be saved; and processors by which information may be processed. These information
 20 technology systems may be used to collect data for later retrieval, analysis, and
 21 manipulation, which may be facilitated through a database program. These information
 22 technology systems provide interfaces that allow users to access and operate various
 23 system components.

24 **[00243]** In one embodiment, the MMADB controller 4901 may be connected to
 25 and/or communicate with entities such as, but not limited to: one or more users from
 26 user input devices 4911; peripheral devices 4912; an optional cryptographic processor
 27 device 4928; and/or a communications network 4913.

28 **[00244]** Networks are commonly thought to comprise the interconnection and
 29 interoperation of clients, servers, and intermediary nodes in a graph topology. It should
 30 be noted that the term “server” as used throughout this application refers generally to a
 31 computer, other device, program, or combination thereof that processes and responds to

1 the requests of remote users across a communications network. Servers serve their
 2 information to requesting “clients.” The term “client” as used herein refers generally to
 3 a computer, program, other device, user and/or combination thereof that is capable of
 4 processing and making requests and obtaining and processing any responses from
 5 servers across a communications network. A computer, other device, program, or
 6 combination thereof that facilitates, processes information and requests, and/or
 7 furthers the passage of information from a source user to a destination user is
 8 commonly referred to as a “node.” Networks are generally thought to facilitate the
 9 transfer of information from source points to destinations. A node specifically tasked
 10 with furthering the passage of information from a source to a destination is commonly
 11 called a “router.” There are many forms of networks such as Local Area Networks
 12 (LANs), Pico networks, Wide Area Networks (WANs), Wireless Networks (WLANs), etc.
 13 For example, the Internet is generally accepted as being an interconnection of a
 14 multitude of networks whereby remote clients and servers may access and interoperate
 15 with one another.

16 **[00245]** The MMCDB controller 4901 may be based on computer systems that may
 17 comprise, but are not limited to, components such as: a computer systemization 4902
 18 connected to memory 4929.

19 **Computer Systemization**

20 **[00246]** A computer systemization 4902 may comprise a clock 4930, central
 21 processing unit (“CPU(s)” and/or “processor(s)” (these terms are used interchangeable
 22 throughout the disclosure unless noted to the contrary)) 4903, a memory 4929 (e.g., a
 23 read only memory (ROM) 4906, a random access memory (RAM) 4905, etc.), and/or an
 24 interface bus 4907, and most frequently, although not necessarily, are all interconnected
 25 and/or communicating through a system bus 4904 on one or more (mother)board(s)
 26 4902 having conductive and/or otherwise transportive circuit pathways through which
 27 instructions (e.g., binary encoded signals) may travel to effectuate communications,
 28 operations, storage, etc. The computer systemization may be connected to a power
 29 source 4986; e.g., optionally the power source may be internal. Optionally, a
 30 cryptographic processor 4926 and/or transceivers (e.g., ICs) 4974 may be connected to

1 the system bus. In another embodiment, the cryptographic processor and/or
 2 transceivers may be connected as either internal and/or external peripheral devices
 3 4912 via the interface bus I/O. In turn, the transceivers may be connected to antenna(s)
 4 4975, thereby effectuating wireless transmission and reception of various
 5 communication and/or sensor protocols; for example the antenna(s) may connect to: a
 6 Texas Instruments WiLink WL1283 transceiver chip (e.g., providing 802.11n, Bluetooth
 7 3.0, FM, global positioning system (GPS) (thereby allowing MMCDB controller to
 8 determine its location)); Broadcom BCM4329FKUBG transceiver chip (e.g., providing
 9 802.11n, Bluetooth 2.1 + EDR, FM, etc.); a Broadcom BCM4750IUB8 receiver chip (e.g.,
 10 GPS); an Infineon Technologies X-Gold 618-PMB9800 (e.g., providing 2G/3G
 11 HSDPA/HSUPA communications); and/or the like. The system clock typically has a
 12 crystal oscillator and generates a base signal through the computer systemization's
 13 circuit pathways. The clock is typically coupled to the system bus and various clock
 14 multipliers that will increase or decrease the base operating frequency for other
 15 components interconnected in the computer systemization. The clock and various
 16 components in a computer systemization drive signals embodying information
 17 throughout the system. Such transmission and reception of instructions embodying
 18 information throughout a computer systemization may be commonly referred to as
 19 communications. These communicative instructions may further be transmitted,
 20 received, and the cause of return and/or reply communications beyond the instant
 21 computer systemization to: communications networks, input devices, other computer
 22 systemizations, peripheral devices, and/or the like. It should be understood that in
 23 alternative embodiments, any of the above components may be connected directly to
 24 one another, connected to the CPU, and/or organized in numerous variations employed
 25 as exemplified by various computer systems.

26 **[00247]** The CPU comprises at least one high-speed data processor adequate to
 27 execute program components for executing user and/or system-generated requests.
 28 Often, the processors themselves will incorporate various specialized processing units,
 29 such as, but not limited to: integrated system (bus) controllers, memory management
 30 control units, floating point units, and even specialized processing sub-units like
 31 graphics processing units, digital signal processing units, and/or the like. Additionally,

1 processors may include internal fast access addressable memory, and be capable of
 2 mapping and addressing memory 4929 beyond the processor itself; internal memory
 3 may include, but is not limited to: fast registers, various levels of cache memory (e.g.,
 4 level 1, 2, 3, etc.), RAM, etc. The processor may access this memory through the use of a
 5 memory address space that is accessible via instruction address, which the processor
 6 can construct and decode allowing it to access a circuit path to a specific memory
 7 address space having a memory state. The CPU may be a microprocessor such as:
 8 AMD's Athlon, Duron and/or Opteron; ARM's application, embedded and secure
 9 processors; IBM and/or Motorola's DragonBall and PowerPC; IBM's and Sony's Cell
 10 processor; Intel's Celeron, Core (2) Duo, Itanium, Pentium, Xeon, and/or XScale;
 11 and/or the like processor(s). The CPU interacts with memory through instruction
 12 passing through conductive and/or transportive conduits (e.g., (printed) electronic
 13 and/or optic circuits) to execute stored instructions (i.e., program code) according to
 14 conventional data processing techniques. Such instruction passing facilitates
 15 communication within the MMCDB controller and beyond through various interfaces.
 16 Should processing requirements dictate a greater amount speed and/or capacity,
 17 distributed processors (e.g., Distributed MMCDB), mainframe, multi-core, parallel,
 18 and/or super-computer architectures may similarly be employed. Alternatively, should
 19 deployment requirements dictate greater portability, smaller Personal Digital Assistants
 20 (PDAs) may be employed.

21 **[00248]** Depending on the particular implementation, features of the MMCDB may
 22 be achieved by implementing a microcontroller such as CAST's R8051XC2
 23 microcontroller; Intel's MCS 51 (i.e., 8051 microcontroller); and/or the like. Also, to
 24 implement certain features of the MMCDB, some feature implementations may rely on
 25 embedded components, such as: Application-Specific Integrated Circuit ("ASIC"),
 26 Digital Signal Processing ("DSP"), Field Programmable Gate Array ("FPGA"), and/or the
 27 like embedded technology. For example, any of the MMCDB component collection
 28 (distributed or otherwise) and/or features may be implemented via the microprocessor
 29 and/or via embedded components; e.g., via ASIC, coprocessor, DSP, FPGA, and/or the
 30 like. Alternately, some implementations of the MMCDB may be implemented with

1 embedded components that are configured and used to achieve a variety of features or
2 signal processing.

3 **[00249]** Depending on the particular implementation, the embedded components
4 may include software solutions, hardware solutions, and/or some combination of both
5 hardware/software solutions. For example, MMCDB features discussed herein may be
6 achieved through implementing FPGAs, which are a semiconductor devices containing
7 programmable logic components called "logic blocks", and programmable
8 interconnects, such as the high performance FPGA Virtex series and/or the low cost
9 Spartan series manufactured by Xilinx. Logic blocks and interconnects can be
10 programmed by the customer or designer, after the FPGA is manufactured, to
11 implement any of the MMCDB features. A hierarchy of programmable interconnects
12 allow logic blocks to be interconnected as needed by the MMCDB system
13 designer/administrator, somewhat like a one-chip programmable breadboard. An
14 FPGA's logic blocks can be programmed to perform the operation of basic logic gates
15 such as AND, and XOR, or more complex combinational operators such as decoders or
16 mathematical operations. In most FPGAs, the logic blocks also include memory
17 elements, which may be circuit flip-flops or more complete blocks of memory. In some
18 circumstances, the MMCDB may be developed on regular FPGAs and then migrated into
19 a fixed version that more resembles ASIC implementations. Alternate or coordinating
20 implementations may migrate MMCDB controller features to a final ASIC instead of or
21 in addition to FPGAs. Depending on the implementation all of the aforementioned
22 embedded components and microprocessors may be considered the "CPU" and/or
23 "processor" for the MMCDB.

24

Power Source

25 **[00250]** The power source 4986 may be of any standard form for powering small
26 electronic circuit board devices such as the following power cells: alkaline, lithium
27 hydride, lithium ion, lithium polymer, nickel cadmium, solar cells, and/or the like.
28 Other types of AC or DC power sources may be used as well. In the case of solar cells, in
29 one embodiment, the case provides an aperture through which the solar cell may
30 capture photonic energy. The power cell 4986 is connected to at least one of the

1 interconnected subsequent components of the MMCDB thereby providing an electric
 2 current to all subsequent components. In one example, the power source 4986 is
 3 connected to the system bus component 4904. In an alternative embodiment, an outside
 4 power source 4986 is provided through a connection across the I/O 4908 interface. For
 5 example, a USB and/or IEEE 1394 connection carries both data and power across the
 6 connection and is therefore a suitable source of power.

7 **Interface Adapters**

8 **[00251]** Interface bus(es) 4907 may accept, connect, and/or communicate to a
 9 number of interface adapters, conventionally although not necessarily in the form of
 10 adapter cards, such as but not limited to: input output interfaces (I/O) 4908, storage
 11 interfaces 4909, network interfaces 4910, and/or the like. Optionally, cryptographic
 12 processor interfaces 4927 similarly may be connected to the interface bus. The interface
 13 bus provides for the communications of interface adapters with one another as well as
 14 with other components of the computer systemization. Interface adapters are adapted
 15 for a compatible interface bus. Interface adapters conventionally connect to the
 16 interface bus via a slot architecture. Conventional slot architectures may be employed,
 17 such as, but not limited to: Accelerated Graphics Port (AGP), Card Bus, (Extended)
 18 Industry Standard Architecture ((E)ISA), Micro Channel Architecture (MCA), NuBus,
 19 Peripheral Component Interconnect (Extended) (PCI(X)), PCI Express, Personal
 20 Computer Memory Card International Association (PCMCIA), and/or the like.

21 **[00252]** Storage interfaces 4909 may accept, communicate, and/or connect to a
 22 number of storage devices such as, but not limited to: storage devices 4914, removable
 23 disc devices, and/or the like. Storage interfaces may employ connection protocols such
 24 as, but not limited to: (Ultra) (Serial) Advanced Technology Attachment (Packet
 25 Interface) ((Ultra) (Serial) ATA(PI)), (Enhanced) Integrated Drive Electronics ((E)IDE),
 26 Institute of Electrical and Electronics Engineers (IEEE) 1394, fiber channel, Small
 27 Computer Systems Interface (SCSI), Universal Serial Bus (USB), and/or the like.

28 **[00253]** Network interfaces 4910 may accept, communicate, and/or connect to a
 29 communications network 4913. Through a communications network 4913, the MMCDB
 30 controller is accessible through remote clients 4933b (e.g., computers with web

1 browsers) by users 4933a. Network interfaces may employ connection protocols such as,
 2 but not limited to: direct connect, Ethernet (thick, thin, twisted pair 10/100/1000 Base
 3 T, and/or the like), Token Ring, wireless connection such as IEEE 802.11a-x, and/or the
 4 like. Should processing requirements dictate a greater amount speed and/or capacity,
 5 distributed network controllers (e.g., Distributed MMCDB), architectures may similarly
 6 be employed to pool, load balance, and/or otherwise increase the communicative
 7 bandwidth required by the MMCDB controller. A communications network may be any
 8 one and/or the combination of the following: a direct interconnection; the Internet; a
 9 Local Area Network (LAN); a Metropolitan Area Network (MAN); an Operating
 10 Missions as Nodes on the Internet (OMNI); a secured custom connection; a Wide Area
 11 Network (WAN); a wireless network (e.g., employing protocols such as, but not limited
 12 to a Wireless Application Protocol (WAP), I-mode, and/or the like); and/or the like. A
 13 network interface may be regarded as a specialized form of an input output interface.
 14 Further, multiple network interfaces 4910 may be used to engage with various
 15 communications network types 4913. For example, multiple network interfaces may be
 16 employed to allow for the communication over broadcast, multicast, and/or unicast
 17 networks.

18 **[00254]** Input Output interfaces (I/O) 4908 may accept, communicate, and/or
 19 connect to user input devices 4911, peripheral devices 4912, cryptographic processor
 20 devices 4928, and/or the like. I/O may employ connection protocols such as, but not
 21 limited to: audio: analog, digital, monaural, RCA, stereo, and/or the like; data: Apple
 22 Desktop Bus (ADB), IEEE 1394a-b, serial, universal serial bus (USB); infrared; joystick;
 23 keyboard; midi; optical; PC AT; PS/2; parallel; radio; video interface: Apple Desktop
 24 Connector (ADC), BNC, coaxial, component, composite, digital, Digital Visual Interface
 25 (DVI), high-definition multimedia interface (HDMI), RCA, RF antennae, S-Video, VGA,
 26 and/or the like; wireless transceivers: 802.11a/b/g/n/x; Bluetooth; cellular (e.g., code
 27 division multiple access (CDMA), high speed packet access (HSPA(+)), high-speed
 28 downlink packet access (HSDPA), global system for mobile communications (GSM),
 29 long term evolution (LTE), WiMax, etc.); and/or the like. One typical output device may
 30 include a video display, which typically comprises a Cathode Ray Tube (CRT) or Liquid
 31 Crystal Display (LCD) based monitor with an interface (e.g., DVI circuitry and cable)

1 that accepts signals from a video interface, may be used. The video interface composites
 2 information generated by a computer systemization and generates video signals based
 3 on the composited information in a video memory frame. Another output device is a
 4 television set, which accepts signals from a video interface. Typically, the video interface
 5 provides the composited video information through a video connection interface that
 6 accepts a video display interface (e.g., an RCA composite video connector accepting an
 7 RCA composite video cable; a DVI connector accepting a DVI display cable, etc.).

8 **[00255]** User input devices 4911 often are a type of peripheral device 512 (see
 9 below) and may include: card readers, dongles, finger print readers, gloves, graphics
 10 tablets, joysticks, keyboards, microphones, mouse (mice), remote controls, retina
 11 readers, touch screens (e.g., capacitive, resistive, etc.), trackballs, trackpads, sensors
 12 (e.g., accelerometers, ambient light, GPS, gyroscopes, proximity, etc.), styluses, and/or
 13 the like.

14 **[00256]** Peripheral devices 4912 may be connected and/or communicate to I/O
 15 and/or other facilities of the like such as network interfaces, storage interfaces, directly
 16 to the interface bus, system bus, the CPU, and/or the like. Peripheral devices may be
 17 external, internal and/or part of the MMCDB controller. Peripheral devices may
 18 include: antenna, audio devices (e.g., line-in, line-out, microphone input, speakers, etc.),
 19 cameras (e.g., still, video, webcam, etc.), dongles (e.g., for copy protection, ensuring
 20 secure transactions with a digital signature, and/or the like), external processors (for
 21 added capabilities; e.g., crypto devices 528), force-feedback devices (e.g., vibrating
 22 motors), network interfaces, printers, scanners, storage devices, transceivers (e.g.,
 23 cellular, GPS, etc.), video devices (e.g., goggles, monitors, etc.), video sources, visors,
 24 and/or the like. Peripheral devices often include types of input devices (e.g., cameras).

25 **[00257]** It should be noted that although user input devices and peripheral devices
 26 may be employed, the MMCDB controller may be embodied as an embedded, dedicated,
 27 and/or monitor-less (i.e., headless) device, wherein access would be provided over a
 28 network interface connection.

29 **[00258]** Cryptographic units such as, but not limited to, microcontrollers,
 30 processors 4926, interfaces 4927, and/or devices 4928 may be attached, and/or

1 communicate with the MMCDB controller. A MC68HC16 microcontroller,
 2 manufactured by Motorola Inc., may be used for and/or within cryptographic units. The
 3 MC68HC16 microcontroller utilizes a 16-bit multiply-and-accumulate instruction in the
 4 16 MHz configuration and requires less than one second to perform a 512-bit RSA
 5 private key operation. Cryptographic units support the authentication of
 6 communications from interacting agents, as well as allowing for anonymous
 7 transactions. Cryptographic units may also be configured as part of the CPU. Equivalent
 8 microcontrollers and/or processors may also be used. Other commercially available
 9 specialized cryptographic processors include: Broadcom's CryptoNetX and other
 10 Security Processors; nCipher's nShield; SafeNet's Luna PCI (e.g., 7100) series;
 11 Semaphore Communications' 40 MHz Roadrunner 184; Sun's Cryptographic
 12 Accelerators (e.g., Accelerator 6000 PCIe Board, Accelerator 500 Daughtercard); Via
 13 Nano Processor (e.g., L2100, L2200, U2400) line, which is capable of performing 500+
 14 MB/s of cryptographic instructions; VLSI Technology's 33 MHz 6868; and/or the like.

15 Memory

16 **[00259]** Generally, any mechanization and/or embodiment allowing a processor to
 17 affect the storage and/or retrieval of information is regarded as memory 4929. However,
 18 memory is a fungible technology and resource, thus, any number of memory
 19 embodiments may be employed in lieu of or in concert with one another. It is to be
 20 understood that the MMCDB controller and/or a computer systemization may employ
 21 various forms of memory 4929. For example, a computer systemization may be
 22 configured wherein the operation of on-chip CPU memory (e.g., registers), RAM, ROM,
 23 and any other storage devices are provided by a paper punch tape or paper punch card
 24 mechanism; however, such an embodiment would result in an extremely slow rate of
 25 operation. In a typical configuration, memory 4929 will include ROM 4906, RAM 4905,
 26 and a storage device 4914. A storage device 4914 may be any conventional computer
 27 system storage. Storage devices may include a drum; a (fixed and/or removable)
 28 magnetic disk drive; a magneto-optical drive; an optical drive (i.e., Blu-ray, CD
 29 ROM/RAM/Recordable (R)/ReWritable (RW), DVD R/RW, HD DVD R/RW etc.); an
 30 array of devices (e.g., Redundant Array of Independent Disks (RAID)); solid state

1 memory devices (USB memory, solid state drives (SSD), etc.); other processor-readable
 2 storage mediums; and/or other devices of the like. Thus, a computer systemization
 3 generally requires and makes use of memory.

4 **Component Collection**

5 **[00260]** The memory 4929 may contain a collection of program and/or database
 6 components and/or data such as, but not limited to: operating system component(s)
 7 4915 (operating system); information server component(s) 4916 (information server);
 8 user interface component(s) 4917 (user interface); Web browser component(s) 4918
 9 (Web browser); database(s) 4919; mail server component(s) 4921; mail client
 10 component(s) 4922; cryptographic server component(s) 4920 (cryptographic server);
 11 the MMADB component(s) 4935; and/or the like (i.e., collectively a component
 12 collection). These components may be stored and accessed from the storage devices
 13 and/or from storage devices accessible through an interface bus. Although non-
 14 conventional program components such as those in the component collection, typically,
 15 are stored in a local storage device 4914, they may also be loaded and/or stored in
 16 memory such as: peripheral devices, RAM, remote storage facilities through a
 17 communications network, ROM, various forms of memory, and/or the like.

18 **Operating System**

19 **[00261]** The operating system component 4915 is an executable program
 20 component facilitating the operation of the MMADB controller. Typically, the operating
 21 system facilitates access of I/O, network interfaces, peripheral devices, storage devices,
 22 and/or the like. The operating system may be a highly fault tolerant, scalable, and
 23 secure system such as: Apple Macintosh OS X (Server); AT&T Plan 9; Be OS; Unix and
 24 Unix-like system distributions (such as AT&T's UNIX; Berkley Software Distribution
 25 (BSD) variations such as FreeBSD, NetBSD, OpenBSD, and/or the like; Linux
 26 distributions such as Red Hat, Ubuntu, and/or the like); and/or the like operating
 27 systems. However, more limited and/or less secure operating systems also may be
 28 employed such as Apple Macintosh OS, IBM OS/2, Microsoft DOS, Microsoft Windows
 29 2000/2003/3.1/95/98/CE/Millennium/NT/Vista/XP/Win7 (Server), Palm OS, and/or

1 the like. An operating system may communicate to and/or with other components in a
 2 component collection, including itself, and/or the like. Most frequently, the operating
 3 system communicates with other program components, user interfaces, and/or the like.
 4 For example, the operating system may contain, communicate, generate, obtain, and/or
 5 provide program component, system, user, and/or data communications, requests,
 6 and/or responses. The operating system, once executed by the CPU, may enable the
 7 interaction with communications networks, data, I/O, peripheral devices, program
 8 components, memory, user input devices, and/or the like. The operating system may
 9 provide communications protocols that allow the MMCDB controller to communicate
 10 with other entities through a communications network 4913. Various communication
 11 protocols may be used by the MMCDB controller as a subcarrier transport mechanism
 12 for interaction, such as, but not limited to: multicast, TCP/IP, UDP, unicast, and/or the
 13 like.

14 **Information Server**

15 **[00262]** An information server component 4916 is a stored program component
 16 that is executed by a CPU. The information server may be a conventional Internet
 17 information server such as, but not limited to Apache Software Foundation's Apache,
 18 Microsoft's Internet Information Server, and/or the like. The information server may
 19 allow for the execution of program components through facilities such as Active Server
 20 Page (ASP), ActiveX, (ANSI) (Objective-) C (++), C# and/or .NET, Common Gateway
 21 Interface (CGI) scripts, dynamic (D) hypertext markup language (HTML), FLASH, Java,
 22 JavaScript, Practical Extraction Report Language (PERL), Hypertext Pre-Processor
 23 (PHP), pipes, Python, wireless application protocol (WAP), WebObjects, and/or the like.
 24 The information server may support secure communications protocols such as, but not
 25 limited to, File Transfer Protocol (FTP); HyperText Transfer Protocol (HTTP); Secure
 26 Hypertext Transfer Protocol (HTTPS), Secure Socket Layer (SSL), messaging protocols
 27 (e.g., America Online (AOL) Instant Messenger (AIM), Application Exchange (APEX),
 28 ICQ, Internet Relay Chat (IRC), Microsoft Network (MSN) Messenger Service, Presence
 29 and Instant Messaging Protocol (PRIM), Internet Engineering Task Force's (IETF's)
 30 Session Initiation Protocol (SIP), SIP for Instant Messaging and Presence Leveraging

1 Extensions (SIMPLE), open XML-based Extensible Messaging and Presence Protocol
 2 (XMPP) (i.e., Jabber or Open Mobile Alliance's (OMA's) Instant Messaging and
 3 Presence Service (IMPS)), Yahoo! Instant Messenger Service, and/or the like. The
 4 information server provides results in the form of Web pages to Web browsers, and
 5 allows for the manipulated generation of the Web pages through interaction with other
 6 program components. After a Domain Name System (DNS) resolution portion of an
 7 HTTP request is resolved to a particular information server, the information server
 8 resolves requests for information at specified locations on the MMCDB controller based
 9 on the remainder of the HTTP request. For example, a request such as
 10 `http://123.124.125.126/myInformation.html` might have the IP portion of the request
 11 `"123.124.125.126"` resolved by a DNS server to an information server at that IP address;
 12 that information server might in turn further parse the http request for the
 13 `"/myInformation.html"` portion of the request and resolve it to a location in memory
 14 containing the information `"myInformation.html."` Additionally, other information
 15 serving protocols may be employed across various ports, e.g., FTP communications
 16 across port 21, and/or the like. An information server may communicate to and/or with
 17 other components in a component collection, including itself, and/or facilities of the
 18 like. Most frequently, the information server communicates with the MMCDB database
 19 4919, operating systems, other program components, user interfaces, Web browsers,
 20 and/or the like.

21 **[00263]** Access to the MMCDB database may be achieved through a number of
 22 database bridge mechanisms such as through scripting languages as enumerated below
 23 (e.g., CGI) and through inter-application communication channels as enumerated below
 24 (e.g., CORBA, WebObjects, etc.). Any data requests through a Web browser are parsed
 25 through the bridge mechanism into appropriate grammars as required by the MMCDB.
 26 In one embodiment, the information server would provide a Web form accessible by a
 27 Web browser. Entries made into supplied fields in the Web form are tagged as having
 28 been entered into the particular fields, and parsed as such. The entered terms are then
 29 passed along with the field tags, which act to instruct the parser to generate queries
 30 directed to appropriate tables and/or fields. In one embodiment, the parser may
 31 generate queries in standard SQL by instantiating a search string with the proper

1 join/select commands based on the tagged text entries, wherein the resulting command
 2 is provided over the bridge mechanism to the MMADB as a query. Upon generating
 3 query results from the query, the results are passed over the bridge mechanism, and
 4 may be parsed for formatting and generation of a new results Web page by the bridge
 5 mechanism. Such a new results Web page is then provided to the information server,
 6 which may supply it to the requesting Web browser.

7 **[00264]** Also, an information server may contain, communicate, generate, obtain,
 8 and/or provide program component, system, user, and/or data communications,
 9 requests, and/or responses.

10

User Interface

11 **[00265]** Computer interfaces in some respects are similar to automobile operation
 12 interfaces. Automobile operation interface elements such as steering wheels, gearshifts,
 13 and speedometers facilitate the access, operation, and display of automobile resources,
 14 and status. Computer interaction interface elements such as check boxes, cursors,
 15 menus, scrollers, and windows (collectively and commonly referred to as widgets)
 16 similarly facilitate the access, capabilities, operation, and display of data and computer
 17 hardware and operating system resources, and status. Operation interfaces are
 18 commonly called user interfaces. Graphical user interfaces (GUIs) such as the Apple
 19 Macintosh Operating System's Aqua, IBM's OS/2, Microsoft's Windows
 20 2000/2003/3.1/95/98/CE/Millennium/NT/XP/Vista/7 (i.e., Aero), Unix's X-Windows
 21 (e.g., which may include additional Unix graphic interface libraries and layers such as K
 22 Desktop Environment (KDE), mythTV and GNU Network Object Model Environment
 23 (GNOME)), web interface libraries (e.g., ActiveX, AJAX, (D)HTML, FLASH, Java,
 24 JavaScript, etc. interface libraries such as, but not limited to, Dojo, jQuery UI,
 25 MooTools, Prototype, script.aculo.us, SWFObject, Yahoo! User Interface, any of which
 26 may be used and provide a baseline and means of accessing and displaying information
 27 graphically to users.

28 **[00266]** A user interface component 4917 is a stored program component that is
 29 executed by a CPU. The user interface may be a conventional graphic user interface as
 30 provided by, with, and/or atop operating systems and/or operating environments such

1 as already discussed. The user interface may allow for the display, execution,
 2 interaction, manipulation, and/or operation of program components and/or system
 3 facilities through textual and/or graphical facilities. The user interface provides a facility
 4 through which users may affect, interact, and/or operate a computer system. A user
 5 interface may communicate to and/or with other components in a component
 6 collection, including itself, and/or facilities of the like. Most frequently, the user
 7 interface communicates with operating systems, other program components, and/or the
 8 like. The user interface may contain, communicate, generate, obtain, and/or provide
 9 program component, system, user, and/or data communications, requests, and/or
 10 responses.

11 **Web Browser**

12 **[00267]** A Web browser component 4918 is a stored program component that is
 13 executed by a CPU. The Web browser may be a conventional hypertext viewing
 14 application such as Microsoft Internet Explorer or Netscape Navigator. Secure Web
 15 browsing may be supplied with 128bit (or greater) encryption by way of HTTPS, SSL,
 16 and/or the like. Web browsers allowing for the execution of program components
 17 through facilities such as ActiveX, AJAX, (D)HTML, FLASH, Java, JavaScript, web
 18 browser plug-in APIs (e.g., Firefox, Safari Plug-in, and/or the like APIs), and/or the like.
 19 Web browsers and like information access tools may be integrated into PDAs, cellular
 20 telephones, and/or other mobile devices. A Web browser may communicate to and/or
 21 with other components in a component collection, including itself, and/or facilities of
 22 the like. Most frequently, the Web browser communicates with information servers,
 23 operating systems, integrated program components (e.g., plug-ins), and/or the like; e.g.,
 24 it may contain, communicate, generate, obtain, and/or provide program component,
 25 system, user, and/or data communications, requests, and/or responses. Also, in place of
 26 a Web browser and information server, a combined application may be developed to
 27 perform similar operations of both. The combined application would similarly affect the
 28 obtaining and the provision of information to users, user agents, and/or the like from
 29 the MMADB enabled nodes. The combined application may be nugatory on systems
 30 employing standard Web browsers.

Mail Server

[00268] A mail server component 4921 is a stored program component that is executed by a CPU 4903. The mail server may be a conventional Internet mail server such as, but not limited to sendmail, Microsoft Exchange, and/or the like. The mail server may allow for the execution of program components through facilities such as ASP, ActiveX, (ANSI) (Objective-) C (++), C# and/or .NET, CGI scripts, Java, JavaScript, PERL, PHP, pipes, Python, WebObjects, and/or the like. The mail server may support communications protocols such as, but not limited to: Internet message access protocol (IMAP), Messaging Application Programming Interface (MAPI)/Microsoft Exchange, post office protocol (POP3), simple mail transfer protocol (SMTP), and/or the like. The mail server can route, forward, and process incoming and outgoing mail messages that have been sent, relayed and/or otherwise traversing through and/or to the MMCDB.

[00269] Access to the MMCDB mail may be achieved through a number of APIs offered by the individual Web server components and/or the operating system.

[00270] Also, a mail server may contain, communicate, generate, obtain, and/or provide program component, system, user, and/or data communications, requests, information, and/or responses.

Mail Client

[00271] A mail client component 4922 is a stored program component that is executed by a CPU 4903. The mail client may be a conventional mail viewing application such as Apple Mail, Microsoft Entourage, Microsoft Outlook, Microsoft Outlook Express, Mozilla, Thunderbird, and/or the like. Mail clients may support a number of transfer protocols, such as: IMAP, Microsoft Exchange, POP3, SMTP, and/or the like. A mail client may communicate to and/or with other components in a component collection, including itself, and/or facilities of the like. Most frequently, the mail client communicates with mail servers, operating systems, other mail clients, and/or the like; e.g., it may contain, communicate, generate, obtain, and/or provide program component, system, user, and/or data communications, requests, information, and/or

1 responses. Generally, the mail client provides a facility to compose and transmit
2 electronic mail messages.

3

Cryptographic Server

4 **[00272]** A cryptographic server component 4920 is a stored program component
5 that is executed by a CPU 4903, cryptographic processor 4926, cryptographic processor
6 interface 4927, cryptographic processor device 4928, and/or the like. Cryptographic
7 processor interfaces will allow for expedition of encryption and/or decryption requests
8 by the cryptographic component; however, the cryptographic component, alternatively,
9 may run on a conventional CPU. The cryptographic component allows for the
10 encryption and/or decryption of provided data. The cryptographic component allows for
11 both symmetric and asymmetric (e.g., Pretty Good Protection (PGP)) encryption and/or
12 decryption. The cryptographic component may employ cryptographic techniques such
13 as, but not limited to: digital certificates (e.g., X.509 authentication framework), digital
14 signatures, dual signatures, enveloping, password access protection, public key
15 management, and/or the like. The cryptographic component will facilitate numerous
16 (encryption and/or decryption) security protocols such as, but not limited to: checksum,
17 Data Encryption Standard (DES), Elliptical Curve Encryption (ECC), International Data
18 Encryption Algorithm (IDEA), Message Digest 5 (MD5, which is a one way hash
19 operation), passwords, Rivest Cipher (RC5), Rijndael, RSA (which is an Internet
20 encryption and authentication system that uses an algorithm developed in 1977 by Ron
21 Rivest, Adi Shamir, and Leonard Adleman), Secure Hash Algorithm (SHA), Secure
22 Socket Layer (SSL), Secure Hypertext Transfer Protocol (HTTPS), and/or the like.
23 Employing such encryption security protocols, the MMCDB may encrypt all incoming
24 and/or outgoing communications and may serve as node within a virtual private
25 network (VPN) with a wider communications network. The cryptographic component
26 facilitates the process of “security authorization” whereby access to a resource is
27 inhibited by a security protocol wherein the cryptographic component effects authorized
28 access to the secured resource. In addition, the cryptographic component may provide
29 unique identifiers of content, e.g., employing and MD5 hash to obtain a unique
30 signature for an digital audio file. A cryptographic component may communicate to

12

13 **[00273]** The MMCDDB database component 4919 may be embodied in a database
14 and its stored data. The database is a stored program component, which is executed by
15 the CPU; the stored program component portion configuring the CPU to process the
16 stored data. The database may be a conventional, fault tolerant, relational, scalable,
17 secure database such as Oracle or Sybase. Relational databases are an extension of a flat
18 file. Relational databases consist of a series of related tables. The tables are
19 interconnected via a key field. Use of the key field allows the combination of the tables
20 by indexing against the key field; i.e., the key fields act as dimensional pivot points for
21 combining information from various tables. Relationships generally identify links
22 maintained between tables by matching primary keys. Primary keys represent fields that
23 uniquely identify the rows of a table in a relational database. More precisely, they
24 uniquely identify rows of a table on the “one” side of a one-to-many relationship.

[00274] Alternatively, the MMCDDB database may be implemented using various standard data-structures, such as an array, hash, (linked) list, struct, structured text file (e.g., XML), table, and/or the like. Such data-structures may be stored in memory and/or in (structured) files. In another alternative, an object-oriented database may be used, such as Frontier, ObjectStore, Poet, Zope, and/or the like. Object databases can include a number of object collections that are grouped and/or linked together by

1 common attributes; they may be related to other object collections by some common
 2 attributes. Object-oriented databases perform similarly to relational databases with the
 3 exception that objects are not just pieces of data but may have other types of capabilities
 4 encapsulated within a given object. If the MMCDB database is implemented as a data-
 5 structure, the use of the MMCDB database 4919 may be integrated into another
 6 component such as the MMCDB component 4935. Also, the database may be
 7 implemented as a mix of data structures, objects, and relational structures. Databases
 8 may be consolidated and/or distributed in countless variations through standard data
 9 processing techniques. Portions of databases, e.g., tables, may be exported and/or
 10 imported and thus decentralized and/or integrated.

11 **[00275]** In one embodiment, the database component 4919 includes several tables
 12 4919a-w. A Users table 4919a may include fields such as, but not limited to: user_id,
 13 ssn, dob, first_name, last_name, age, state, address_firstline, address_secondline,
 14 zipcode, devices_list, contact_info, contact_type, alt_contact_info, alt_contact_type,
 15 and/or the like. The Users table may support and/or track multiple entity accounts on a
 16 MMCDB. A Devices table 4919b may include fields such as, but not limited to:
 17 device_id, user_id, client_ip, client_type, client_model, operating_system, os_version,
 18 app_installed_flag, and/or the like. An Apps table 4919c may include fields such as, but
 19 not limited to: app_id, app_name, app_type, OS_compatibilities_list, version,
 20 timestamp, developer_id, and/or the like. An Accounts table 4919d may include fields
 21 such as, but not limited to: account_id, account_firstname, account_lastname,
 22 account_type, account_num, account_balance_list, billingaddress_line1,
 23 billingaddress_line2, billing_zipcode, billing_state, shipping_preferences,
 24 shippingaddress_line1, shippingaddress_line2, shipping_zipcode, shipping_state,
 25 and/or the like. A Merchants table 4919e may include fields such as, but not limited to:
 26 merchant_id, merchant_name, merchant_address, ip_address, mac_address,
 27 auth_key, port_num, security_settings_list, and/or the like. An Issuers table 4919f
 28 may include fields such as, but not limited to: issuer_id, issuer_name, issuer_address,
 29 ip_address, mac_address, auth_key, port_num, security_settings_list, and/or the like.
 30 An Acquirers table 4919g may include fields such as, but not limited to: acquirer_id,
 31 account_firstname, account_lastname, account_type, account_num,

1 account_balance_list, billingaddress_line1, billingaddress_line2, billing_zipcode,
 2 billing_state, shipping_preferences, shippingaddress_line1, shippingaddress_line2,
 3 shipping_zipcode, shipping_state, and/or the like. A Gateways table 4919h may
 4 include fields such as, but not limited to: gateway_id, gateway_name, merchant_id,
 5 issuer_id, acquirer_id, user_id, and/or the like. A Transactions table 4919i may include
 6 fields such as, but not limited to: transaction_id, order_id, user_id, timestamp,
 7 transaction_cost, purchase_details_list, num_products, products_list, product_type,
 8 product_params_list, product_title, product_summary, quantity, user_id, client_id,
 9 client_ip, client_type, client_model, operating_system, os_version, app_installed_flag,
 10 user_id, account_firstname, account_lastname, account_type, account_num,
 11 billingaddress_line1, billingaddress_line2, billing_zipcode, billing_state,
 12 shipping_preferences, shippingaddress_line1, shippingaddress_line2,
 13 shipping_zipcode, shipping_state, merchant_id, merchant_name, merchant_auth_
 14 key, and/or the like. A Batches table 4919j may include fields such as, but not limited
 15 to: batch_id, parent_batch_id, transaction_id, account_id, user_id, app_id,
 16 batch_rules, and/or the like. A Ledgers table 4919k may include fields such as, but not
 17 limited to: ledger_id, transaction_id, user_id, merchant_id, issuer_id, acquirer_id,
 18 aggregation_id, ledger_name, ledger_value, and/or the like. A Products table 4919l
 19 may include fields such as, but not limited to: product_id, product_name, sku, price,
 20 inventory_level, stores_carrying, unit_of_measure, and/or the like. A Offers table
 21 4919m may include fields such as, but not limited to: offer_id, merchant_id,
 22 offered_to_user_id, offer_type, offer_description, start_date, end_date,
 23 num_times_redeemed, and/or the like. A Behavior table 4919n may include fields such
 24 as, but not limited to: behavior_id, user_id, behavior_description, behavior_type,
 25 behavior_value, date_time_behavior, and/or the like. An Analytics table 4919o may
 26 include fields such as, but not limited to: analytics_id, batch_id, user_id,
 27 transaction_id, generated_graph, generated_results_set, generated_results_set_json,
 28 input_data_set, date_time_generated, and/or the like. A Market Data table 4919p may
 29 include fields such as, but not limited to: market_data_id, index_name, index_value,
 30 last_updated_index_datetime, and/or the like. An Input Languages table 4919q may
 31 include fields such as, but not limited to: input_language_id, function_name,
 32 function_definition, parent_input_language_id, mesh_language_id, user_id,

1 tumbler_id, aggregation_id, and/or the like. A Mesh Language table 4919r may include
 2 fields such as, but not limited to: mesh_language_id, operation_name,
 3 operation_min_test_case, operation_max_test_case, operation_custom_test_case,
 4 mesh_language_version, mesh_language_updated_date, and/or the like. A Tumblers
 5 table 4919s may include fields such as, but not limited to: tumbler_id,
 6 user_visible_model_commands, non_user_visible_model_commands, input_key,
 7 output_key, and/or the like. An Aggregation table 4919t may include fields such as, but
 8 not limited to: aggregation_id, aggregation_data_source, key, value,
 9 parent_aggregation_id, and/or the like. A Category table 4919u may include fields such
 10 as, but not limited to: category_id, mesh_id, user_id, category_name, category_type,
 11 entity_name, is_present_in_mesh, and/or the like. A Mesh table 4919v may include
 12 fields such as, but not limited to: mesh_id, mesh_node, mesh_node_value,
 13 mesh_edge, mesh_edge_value, mesh_link, mesh_link_value, attributes, tags,
 14 keywords, and/or the like. A Price Trends table 4919w may include fields such as, but
 15 not limited to: price_trends_id, merchant_id, date_price_observed,
 16 number_observations, observed_price, next_check_date, inventory_quantity, and/or
 17 the like.

18 **[00276]** In one embodiment, the MMCDB database may interact with other
 19 database systems. For example, employing a distributed database system, queries and
 20 data access by search MMCDB component may treat the combination of the MMCDB
 21 database, an integrated data security layer database as a single database entity.

22 **[00277]** In one embodiment, user programs may contain various user interface
 23 primitives, which may serve to update the MMCDB. Also, various accounts may require
 24 custom database tables depending upon the environments and the types of clients the
 25 MMCDB may need to serve. It should be noted that any unique fields may be designated
 26 as a key field throughout. In an alternative embodiment, these tables have been
 27 decentralized into their own databases and their respective database controllers (i.e.,
 28 individual database controllers for each of the above tables). Employing standard data
 29 processing techniques, one may further distribute the databases over several computer
 30 systemizations and/or storage devices. Similarly, configurations of the decentralized
 31 database controllers may be varied by consolidating and/or distributing the various

1 database components 4919a-w. The MMCDB may be configured to keep track of various
2 settings, inputs, and parameters via database controllers.

3 **[00278]** The MMCDB database may communicate to and/or with other
4 components in a component collection, including itself, and/or facilities of the like.
5 Most frequently, the MMCDB database communicates with the MMCDB component,
6 other program components, and/or the like. The database may contain, retain, and
7 provide information regarding other nodes and data.

8

The MMCDBs

9 **[00279]** The MMCDB component 4935 is a stored program component that is
10 executed by a CPU. In one embodiment, the MMCDB component incorporates any
11 and/or all combinations of the aspects of the MMCDB that was discussed in the
12 previous figures. As such, the MMCDB affects accessing, obtaining and the provision of
13 information, services, transactions, and/or the like across various communications
14 networks. The features and embodiments of the MMCDB discussed herein increase
15 network efficiency by reducing data transfer requirements the use of more efficient data
16 structures and mechanisms for their transfer and storage. As a consequence, more data
17 may be transferred in less time, and latencies with regard to transactions, are also
18 reduced. In many cases, such reduction in storage, transfer time, bandwidth
19 requirements, latencies, etc., will reduce the capacity and structural infrastructure
20 requirements to support the MMCDB's features and facilities, and in many cases reduce
21 the costs, energy consumption/requirements, and extend the life of MMCDB's
22 underlying infrastructure; this has the added benefit of making the MMCDB more
23 reliable. The generation of the mesh graph and dictionary entries by the MMCDB has
24 the technical effect of allowing more transaction, search, enrollment and email data to
25 be analyzed and queried by the MMCDB user without a corresponding increase in data
26 storage server / processing infrastructure. For example, by utilizing the aggregated data
27 record normalization 306, data field recognition 307, entity type classification 308,
28 cross-entity correlation 309, and entity attribute 310 components of the MMCDB, raw
29 aggregated data may be stored in a more efficient manner and indexed and searched in a
30 manner requiring less physical infrastructure and supporting faster querying with

1 reduced latency (e.g., through the use of a distributed linking node mesh search
 2 component). Aspects of the MMCDB facilitate faster transaction processing by reducing
 3 consumer decision latency (e.g., through the provision of customized offers requiring
 4 decreased user input and therefore reduced data transfer requirements). Similarly,
 5 many of the features and mechanisms are designed to be easier for users to use and
 6 access, thereby broadening the audience that may enjoy/employ and exploit the feature
 7 sets of the MMCDB; such ease of use also helps to increase the reliability of the
 8 MMCDB. In addition, the feature sets include heightened security as noted via the
 9 Cryptographic components 4920, 4926, 4928 and throughout, making access to the
 10 features and data more reliable and secure.

11 **[00280]** The MMCDB component may transform data aggregated from various
 12 computer resources via MMCDB components into updated entity profiles and/or social
 13 graphs, and/or the like and use of the MMCDB. In one embodiment, the MMCDB
 14 component 4935 takes inputs such as aggregated data from various computer resources,
 15 and transforms the inputs via various components (e.g., SRA 4941, CTE 4942, TDA
 16 4943 SDA 4944, VASE 4945, DFR 4946, ETC 4947, CEC 4948, EAA 4949, EPGU 4950,
 17 STG 4951, MA 4952, UBPA 4953, UPI 4954, TDN 4955, CTC 4956, TDF 4957, CDA
 18 4958, ESA 4959, BAR 4960, AMS 4961, ADRN 4962, EXC 4963, CRA 4964, and/or the
 19 like), into outputs such as updated entity profiles and social graphs within the MMCDB.

20 **[00281]** The MMCDB component enabling access of information between nodes
 21 may be developed by employing standard development tools and languages such as, but
 22 not limited to: Apache components, Assembly, ActiveX, binary executables, (ANSI)
 23 (Objective-) C (++), C# and/or .NET, database adapters, CGI scripts, Java, JavaScript,
 24 mapping tools, procedural and object oriented development tools, PERL, PHP, Python,
 25 shell scripts, SQL commands, web application server extensions, web development
 26 environments and libraries (e.g., Microsoft's ActiveX; Adobe AIR, FLEX & FLASH;
 27 AJAX; (D)HTML; Dojo, Java; JavaScript; jQuery(UI); MooTools; Prototype;
 28 script.aculo.us; Simple Object Access Protocol (SOAP); SWFObject; Yahoo! User
 29 Interface; and/or the like), WebObjects, and/or the like. In one embodiment, the
 30 MMCDB server employs a cryptographic server to encrypt and decrypt communications.
 31 The MMCDB component may communicate to and/or with other components in a

1 component collection, including itself, and/or facilities of the like. Most frequently, the
 2 MMADB component communicates with the MMADB database, operating systems,
 3 other program components, and/or the like. The MMADB may contain, communicate,
 4 generate, obtain, and/or provide program component, system, user, and/or data
 5 communications, requests, and/or responses.

6

Distributed MMADBs

7 **[00282]** The structure and/or operation of any of the MMADB node controller
 8 components may be combined, consolidated, and/or distributed in any number of ways
 9 to facilitate development and/or deployment. Similarly, the component collection may
 10 be combined in any number of ways to facilitate deployment and/or development. To
 11 accomplish this, one may integrate the components into a common code base or in a
 12 facility that can dynamically load the components on demand in an integrated fashion.

13 **[00283]** The component collection may be consolidated and/or distributed in
 14 countless variations through standard data processing and/or development techniques.
 15 Multiple instances of any one of the program components in the program component
 16 collection may be instantiated on a single node, and/or across numerous nodes to
 17 improve performance through load-balancing and/or data-processing techniques.
 18 Furthermore, single instances may also be distributed across multiple controllers
 19 and/or storage devices; e.g., databases. All program component instances and
 20 controllers working in concert may do so through standard data processing
 21 communication techniques.

22 **[00284]** The configuration of the MMADB controller will depend on the context of
 23 system deployment. Factors such as, but not limited to, the budget, capacity, location,
 24 and/or use of the underlying hardware resources may affect deployment requirements
 25 and configuration. Regardless of if the configuration results in more consolidated
 26 and/or integrated program components, results in a more distributed series of program
 27 components, and/or results in some combination between a consolidated and
 28 distributed configuration, data may be communicated, obtained, and/or provided.
 29 Instances of components consolidated into a common code base from the program
 30 component collection may communicate, obtain, and/or provide data. This may be

1 accomplished through intra-application data processing communication techniques
 2 such as, but not limited to: data referencing (e.g., pointers), internal messaging, object
 3 instance variable communication, shared memory space, variable passing, and/or the
 4 like.

5 **[00285]** If component collection components are discrete, separate, and/or
 6 external to one another, then communicating, obtaining, and/or providing data with
 7 and/or to other component components may be accomplished through inter-application
 8 data processing communication techniques such as, but not limited to: Application
 9 Program Interfaces (API) information passage; (distributed) Component Object Model
 10 ((D)COM), (Distributed) Object Linking and Embedding ((D)OLE), and/or the like),
 11 Common Object Request Broker Architecture (CORBA), Jini local and remote
 12 application program interfaces, JavaScript Object Notation (JSON), Remote Method
 13 Invocation (RMI), SOAP, process pipes, shared files, and/or the like. Messages sent
 14 between discrete component components for inter-application communication or within
 15 memory spaces of a singular component for intra-application communication may be
 16 facilitated through the creation and parsing of a grammar. A grammar may be
 17 developed by using development tools such as lex, yacc, XML, and/or the like, which
 18 allow for grammar generation and parsing capabilities, which in turn may form the basis
 19 of communication messages within and between components.

20 **[00286]** For example, a grammar may be arranged to recognize the tokens of an
 21 HTTP post command, e.g.:

22 `w3c -post http://... Value1`

23 **[00287]** where Value1 is discerned as being a parameter because “http://” is part of
 24 the grammar syntax, and what follows is considered part of the post value. Similarly,
 25 with such a grammar, a variable “Value1” may be inserted into an “http://” post
 26 command and then sent. The grammar syntax itself may be presented as structured data
 27 that is interpreted and/or otherwise used to generate the parsing mechanism (e.g., a
 28 syntax description text file as processed by lex, yacc, etc.). Also, once the parsing
 29 mechanism is generated and/or instantiated, it itself may process and/or parse
 30 structured data such as, but not limited to: character (e.g., tab) delineated text, HTML,

1 structured text streams, XML, and/or the like structured data. In another embodiment,
 2 inter-application data processing protocols themselves may have integrated and/or
 3 readily available parsers (e.g., JSON, SOAP, and/or like parsers) that may be employed
 4 to parse (e.g., communications) data. Further, the parsing grammar may be used
 5 beyond message parsing, but may also be used to parse: databases, data collections, data
 6 stores, structured data, and/or the like. Again, the desired configuration will depend
 7 upon the context, environment, and requirements of system deployment.

8 **[00288]** For example, in some implementations, the MMCDB controller may be
 9 executing a PHP script implementing a Secure Sockets Layer (“SSL”) socket server via
 10 the information sherver, which listens to incoming communications on a server port to
 11 which a client may send data, e.g., data encoded in JSON format. Upon identifying an
 12 incoming communication, the PHP script may read the incoming message from the
 13 client device, parse the received JSON-encoded text data to extract information from the
 14 JSON-encoded text data into PHP script variables, and store the data (e.g., client
 15 identifying information, etc.) and/or extracted information in a relational database
 16 accessible using the Structured Query Language (“SQL”). An exemplary listing, written
 17 substantially in the form of PHP/SQL commands, to accept JSON-encoded input data
 18 from a client device via a SSL connection, parse the data to extract variables, and store
 19 the data to a database, is provided below:

```

20  <?PHP
21  header('Content-Type: text/plain');
22
23  //set ip address and port to listen to for incoming data
24  $address = '192.168.0.100';
25  $port = 255;
26
27  //create a server-side SSL socket, listen
28  //for/accept incoming communication
29  $sock = socket_create(AF_INET, SOCK_STREAM, 0);
30  socket_bind($sock, $address, $port)
31      or die('Could not bind to address');
32  socket_listen($sock);
33  $client = socket_accept($sock);
  
```

```

1
2  //read input data from client device in 1024 byte
3  //blocks until end of message
4  do {
5      $input = "";
6      $input = socket_read($client, 1024);
7      $data .= $input;
8  } while($input != "");
9
10 // parse data to extract variables
11 $obj = json_decode($data, true);
12
13 // store input data in a database
14 mysql_connect("10.1.1.1",$srvr,$pass); // access database server
15 mysql_select("CLIENT_DB.SQL"); // select database to append
16 mysql_query("INSERT INTO UserTable (transmission) VALUES
17 ($data)"); // add data to UserTable table in a CLIENT database
18 mysql_close("CLIENT_DB.SQL"); // close connection to database
19 ?>

```

20 **[00289]** Also, the following resources may be used to provide example
 21 embodiments regarding SOAP parser implementation:

```

22  http://www.xav.com/perl/site/lib/SOAP/Parser.html
23  http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?
24  topic=/com.ibm.IBMDI.doc/referenceguide295.htm

```

25 **[00290]** and other parser implementations:

```

26  http://publib.boulder.ibm.com/infocenter/tivihelp/v2r1/index.jsp?
27  topic=/com.ibm.IBMDI.doc/referenceguide259.htm

```

28
 29 **[00291]** all of which are hereby expressly incorporated by reference.

30 **[00292]** In order to address various issues and advance the art, the entirety of this
 31 application for MMCDDB (including the Cover Page, Title, Headings, Field, Background,
 32 Summary, Brief Description of the Drawings, Detailed Description, Claims, Abstract,

1 Figures, Appendices, and otherwise) shows, by way of illustration, various embodiments
2 in which the claimed innovations may be practiced. The advantages and features of the
3 application are of a representative sample of embodiments only, and are not exhaustive
4 and/or exclusive. They are presented only to assist in understanding and teach the
5 claimed principles. It should be understood that they are not representative of all
6 claimed innovations. As such, certain aspects of the disclosure have not been discussed
7 herein. That alternate embodiments may not have been presented for a specific portion
8 of the innovations or that further undescribed alternate embodiments may be available
9 for a portion is not to be considered a disclaimer of those alternate embodiments. It will
10 be appreciated that many of those undescribed embodiments incorporate the same
11 principles of the innovations and others are equivalent. Thus, it is to be understood that
12 other embodiments may be utilized and functional, logical, operational, organizational,
13 structural and/or topological modifications may be made without departing from the
14 scope and/or spirit of the disclosure. As such, all examples and/or embodiments are
15 deemed to be non-limiting throughout this disclosure. Also, no inference should be
16 drawn regarding those embodiments discussed herein relative to those not discussed
17 herein other than it is as such for purposes of reducing space and repetition. For
18 instance, it is to be understood that the logical and/or topological structure of any
19 combination of any program components (a component collection), other components
20 and/or any present feature sets as described in the figures and/or throughout are not
21 limited to a fixed operating order and/or arrangement, but rather, any disclosed order is
22 exemplary and all equivalents, regardless of order, are contemplated by the disclosure.
23 Furthermore, it is to be understood that such features are not limited to serial execution,
24 but rather, any number of threads, processes, services, servers, and/or the like that may
25 execute asynchronously, concurrently, in parallel, simultaneously, synchronously,
26 and/or the like are contemplated by the disclosure. As such, some of these features may
27 be mutually contradictory, in that they cannot be simultaneously present in a single
28 embodiment. Similarly, some features are applicable to one aspect of the innovations,
29 and inapplicable to others. In addition, the disclosure includes other innovations not
30 presently claimed. Applicant reserves all rights in those presently unclaimed
31 innovations including the right to claim such innovations, file additional applications,
32 continuations, continuations in part, divisions, and/or the like thereof. As such, it

1 should be understood that advantages, embodiments, examples, functional, features,
2 logical, operational, organizational, structural, topological, and/or other aspects of the
3 disclosure are not to be considered limitations on the disclosure as defined by the claims
4 or limitations on equivalents to the claims. It is to be understood that, depending on the
5 particular needs and/or characteristics of a MMCDB individual and/or enterprise user,
6 database configuration and/or relational model, data type, data transmission and/or
7 network framework, syntax structure, and/or the like, various embodiments of the
8 MMCDB, may be implemented that enable a great deal of flexibility and customization.
9 For example, aspects of the MMCDB may be adapted for restaurant dining, online
10 shopping ,brick-and-mortar shopping, secured information processing, and/or the like.
11 While various embodiments and discussions of the MMCDB have been directed to
12 electronic purchase transactions, however, it is to be understood that the embodiments
13 described herein may be readily configured and/or customized for a wide variety of
14 other applications and/or implementations.

What is claimed is:

1. A merchant analytics platform processor-implemented method comprising:
 - obtaining, by a pay network server, a hypertext transfer protocol (HTTP) GET message from a merchant server, the HTTP GET message including a request for a merchant analytics recommendation including a user identifier;
 - parsing, by the pay network server, the HTTP GET message to extract the user identifier; upon obtaining the HTTP GET message and extracting the user identifier, querying, by the pay network server, a distributed linking node mesh for entities correlated with the user identifier;
 - receiving, by the pay network server, aggregated user entity correlation data;
 - generating, by the pay network server, a user behavior profile based on the aggregated user entity correlation data;
 - determining, by the pay network server, a product or service using the user behavior profile;
 - based on the determination of the product or service, generating, by the pay network server, an HTTP POST message including an indication of the product or service; and
 - providing, by the pay network server, the HTTP POST message to the merchant server in response to the request for the merchant analytics recommendation;
 - wherein the distributed linking node mesh includes a node representing an observable entity and a node representing a deduced entity derived through aggregating information associated with the user.
2. The method of claim 1, wherein the user identifier is user payment account information.
3. The method of claim 2, additionally comprising:
 - querying a transaction database for a user identity associated with the user payment account information.
4. The method of claim 1, additionally comprising:
 - anonymizing the aggregated user entity correlation data.
5. The method of claim 4, wherein anonymizing includes masking user payment account information.

6. The method of claim 4, wherein anonymizing includes swapping aggregated user entity correlation data.
7. The method of claim 6, wherein swapping aggregated user entity correlation data preserves the sum of the user entity correlation components.
8. The method of claim 4, wherein anonymizing includes applying multiple functions to the aggregated user entity correlation data based on the user entity correlation data type.
9. The method of claim 1, wherein generating a user behavior profile includes determining an entity not present in the distributed node linking mesh.
10. The method of claim 9, additionally comprising:
inserting the determined entity into the distributed node linking mesh.
11. The method of claim 10, additionally comprising:
updating the distributed node linking mesh; and
querying the distributed node linking mesh for updated entities correlated with the user identifier.
12. The method of claim 1, wherein the aggregated user entity correlation data includes aggregated search results data.
13. The method of claim 11, wherein the aggregated user entity correlation data includes aggregated transaction data.
14. The method of claim 11, wherein the aggregated user entity correlation data includes aggregated service usage data.
15. The method of claim 1, wherein the aggregated user entity correlation data includes aggregated enrollment data.
16. The method of claim 1, wherein the aggregated user entity correlation data includes

aggregated email data.

17. The method of claim 1, wherein the aggregated user entity correlation data includes aggregated social media data.

18. The method of claim 1, additionally comprising:
 querying a merchant inventory database to determine a current inventory level of the product or service.

19. The method of claim 18, additionally comprising:
 determining a second product or service using the user behavior profile.

20. A merchant analytics platform processor-implemented method comprising:
 obtaining, by a pay network server, a hypertext transfer protocol (HTTP) GET message from a merchant server, the HTTP GET message including a request for a merchant analytics recommendation including a user identification package, wherein the user identification package includes user contact information and an approximate user location;
 parsing, by the pay network server, the HTTP GET message to extract the user identification package;
 upon obtaining the HTTP GET message and extracting the user identification package,
 querying, by the pay network server, a distributed linking node mesh for entities correlated with the user identification package;
 receiving, by the pay network server, aggregated user entity correlation data;
 querying, by the pay network server, an anonymization database for at least one anonymization operation applicable to the aggregated user entity correlation data;
 applying, by the pay network server, a first at least one anonymization operation to the aggregated user entity correlation data;
 determining, by the pay network server, that the aggregated user entity correlation data is not sufficiently anonymized;
 applying, by the pay network server, a second at least one anonymization operation to the aggregated user entity correlation data;
 querying, by the pay network server, a user behavior template database for a user behavior template model;

generating, by the pay network server, a user behavior profile based on the user behavior template model and the aggregated user entity correlation data;

determining, by the pay network server, using the user behavior profile, a product or service having the highest likelihood of being purchased by the user;

querying, by the pay network server, a merchant inventory database to determine a current inventory level of the product or service;

based on the determination of the product or service and the current inventory level of the product or service, generating, by the pay network server, an HTTP POST message including an indication of the product or service; and

providing, by the pay network server, the HTTP POST message to the merchant server in response to the request for the merchant analytics recommendation;

wherein the distributed linking node mesh includes a node representing an observable entity and a node representing a deduced entity derived through aggregating information associated with the user.