US 20160078120A1

(19) **United States**

(12) **Patent Application Publication**

**Pradeep et al.**

(10) **Pub. No.: US 2016/0078120 A1**

(43) **Pub. Date: Mar. 17, 2016**

(54) **EXTRACTING AND PROCESSING METRICS FROM SYSTEM GENERATED EVENTS**

(71) Applicant: **salesforce.com, inc.**, San Francisco, CA (US)

(72) Inventors: **Aakash Pradeep**, Union City, CA (US); **Adam Torman**, Walnut Creek, CA (US); **Alex Warshavsky**, Walnut Creek, CA (US); **Samarpan Jain**, Fremont, CA (US); **Soumen Bandyopadhyay**, Glen Park, CA (US); **Thomas William D'Silva**, Fremont, CA (US)

(21) Appl. No.: **14/688,989**

(22) Filed: **Apr. 16, 2015**

**Related U.S. Application Data**

(60) Provisional application No. 62/048,961, filed on Sep. 11, 2014.

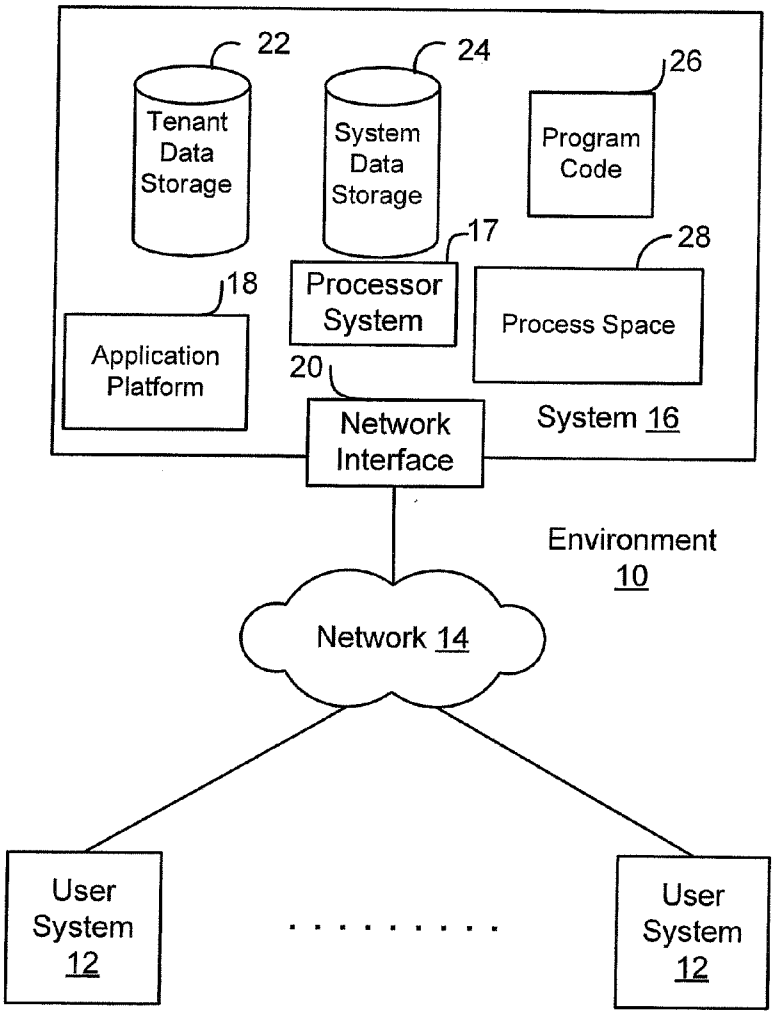
**Publication Classification**

(51) **Int. Cl.**
*G06F 17/30* (2006.01)

(52) **U.S. Cl.**
CPC .... ***G06F 17/30598*** (2013.01); ***G06F 17/30867*** (2013.01); ***G06F 17/30528*** (2013.01)

(57) **ABSTRACT**

In an example, a processing system of a database system may categorize event data taken from logged interactions of users with a multi-tenant information system to provide a metric. The processing system of the database system may periodically calculate the metric for a particular one of the tenants, and electronically store the periodically calculated metrics for accessing responsive to a query of the particular tenant.



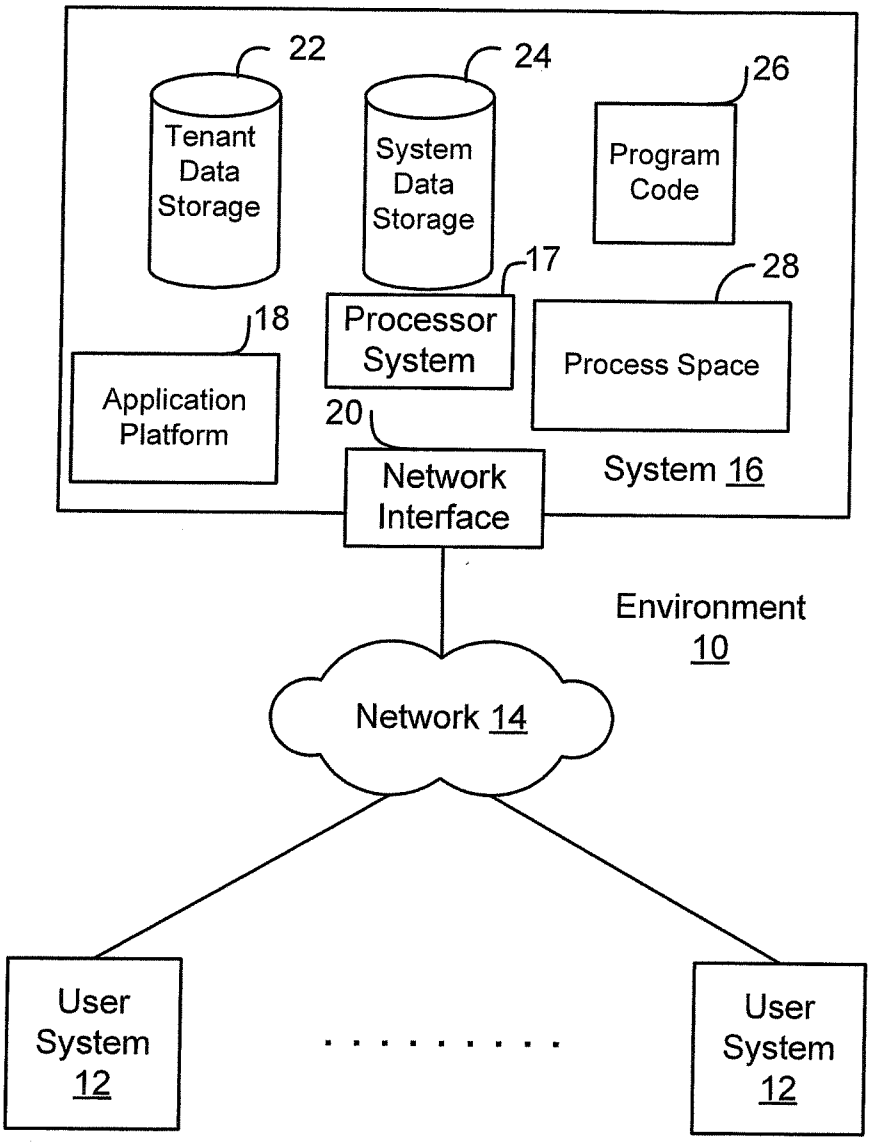

22
24
26
Tenant Data Storage
System Data Storage
Program Code
17
28
18
Processor System
Process Space
Application Platform
20
Network Interface
System 16
Environment 10
Network 14
User System 12
User System 12

*FIGURE 1A*

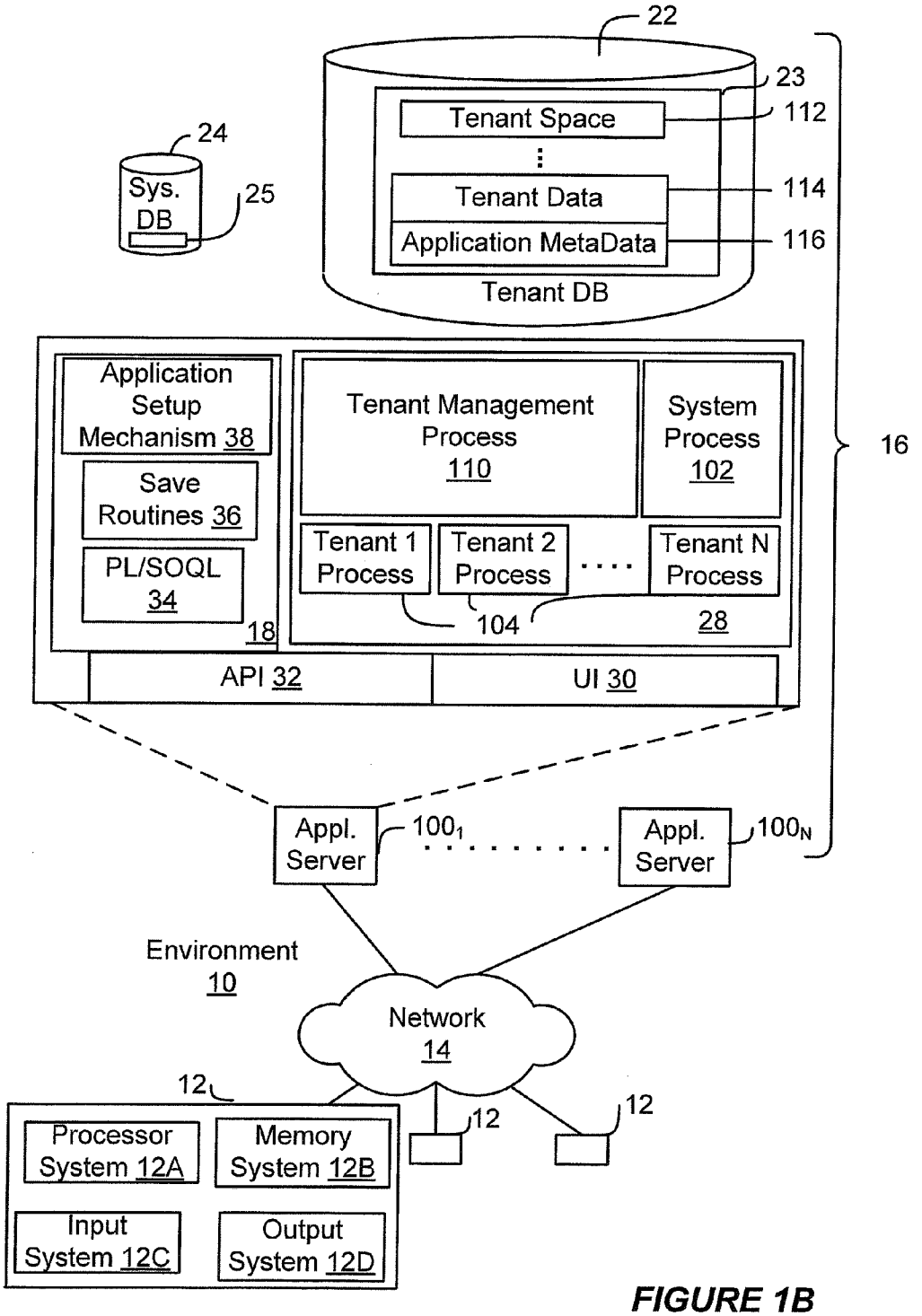

22
23
Tenant Space
112
24
Sys. DB
25
Tenant Data
114
Application MetaData
116
Tenant DB
Application Setup Mechanism 38
Save Routines 36
PL/SOQL 34
18
Tenant Management Process 110
System Process 102
Tenant 1 Process
Tenant 2 Process
Tenant N Process
104
28
16
API 32
UI 30
Appl. Server
100$_1$
Appl. Server
100$_N$
Environment 10
Network 14
12
12
12
Processor System 12A
Memory System 12B
Input System 12C
Output System 12D

*FIGURE 1B*

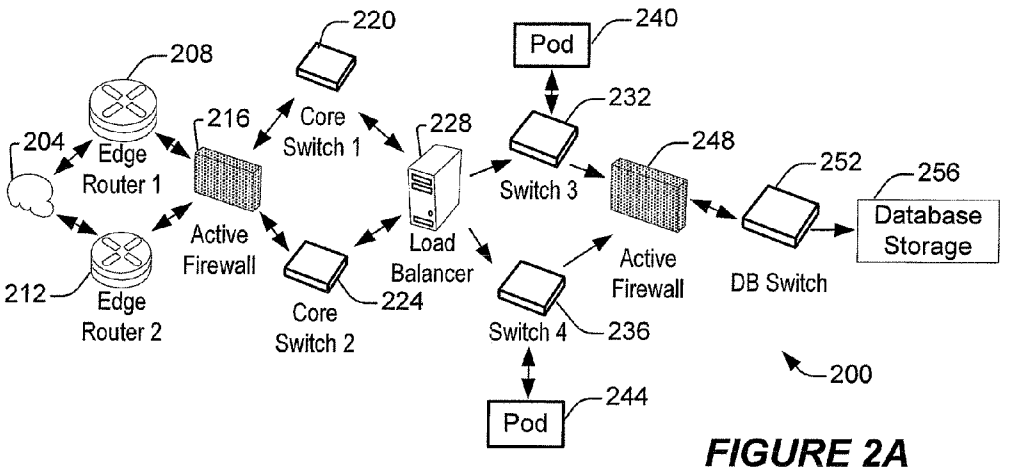



FIGURE 2A

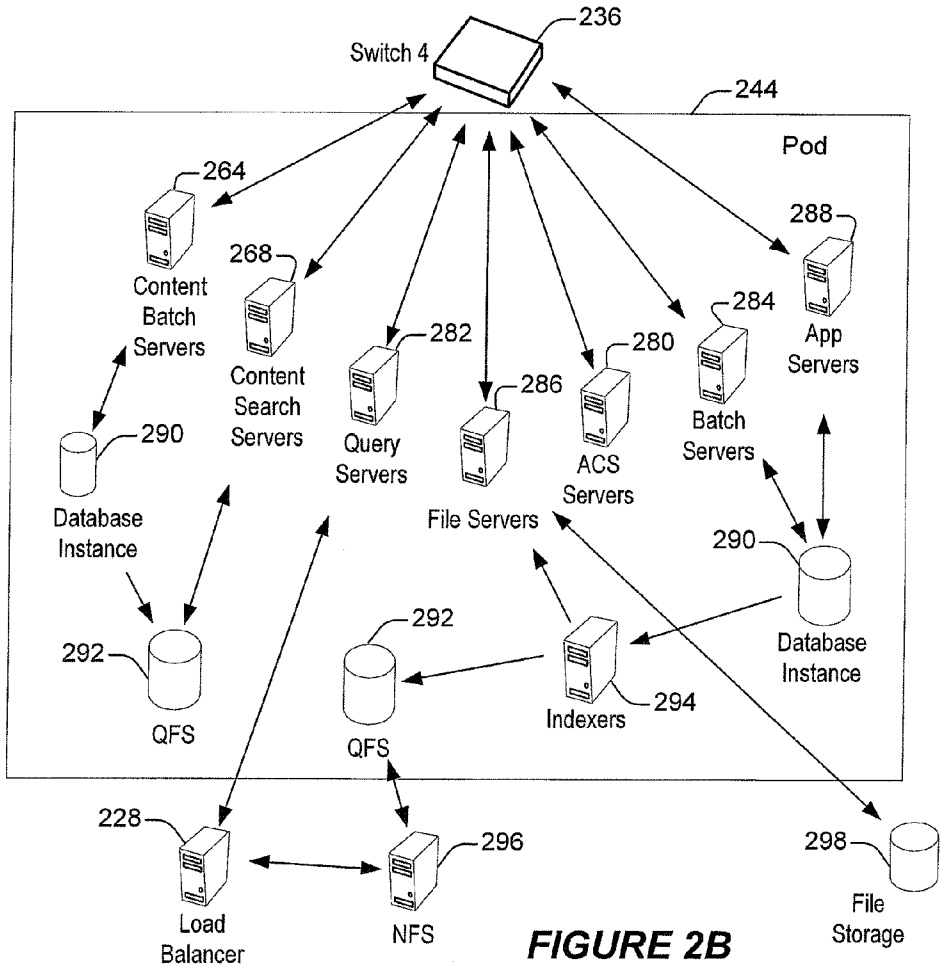



FIGURE 2B

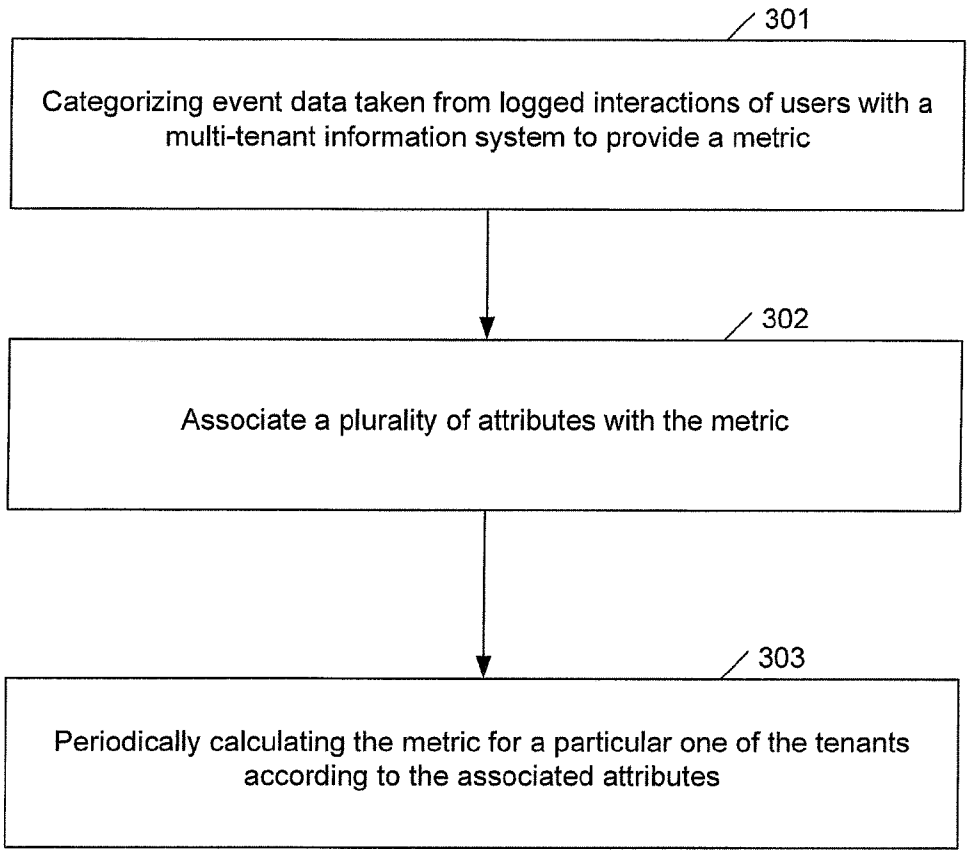

301
Categorizing event data taken from logged interactions of users with a multi-tenant information system to provide a metric
302
Associate a plurality of attributes with the metric
303
Periodically calculating the metric for a particular one of the tenants according to the associated attributes

*FIGURE 3*

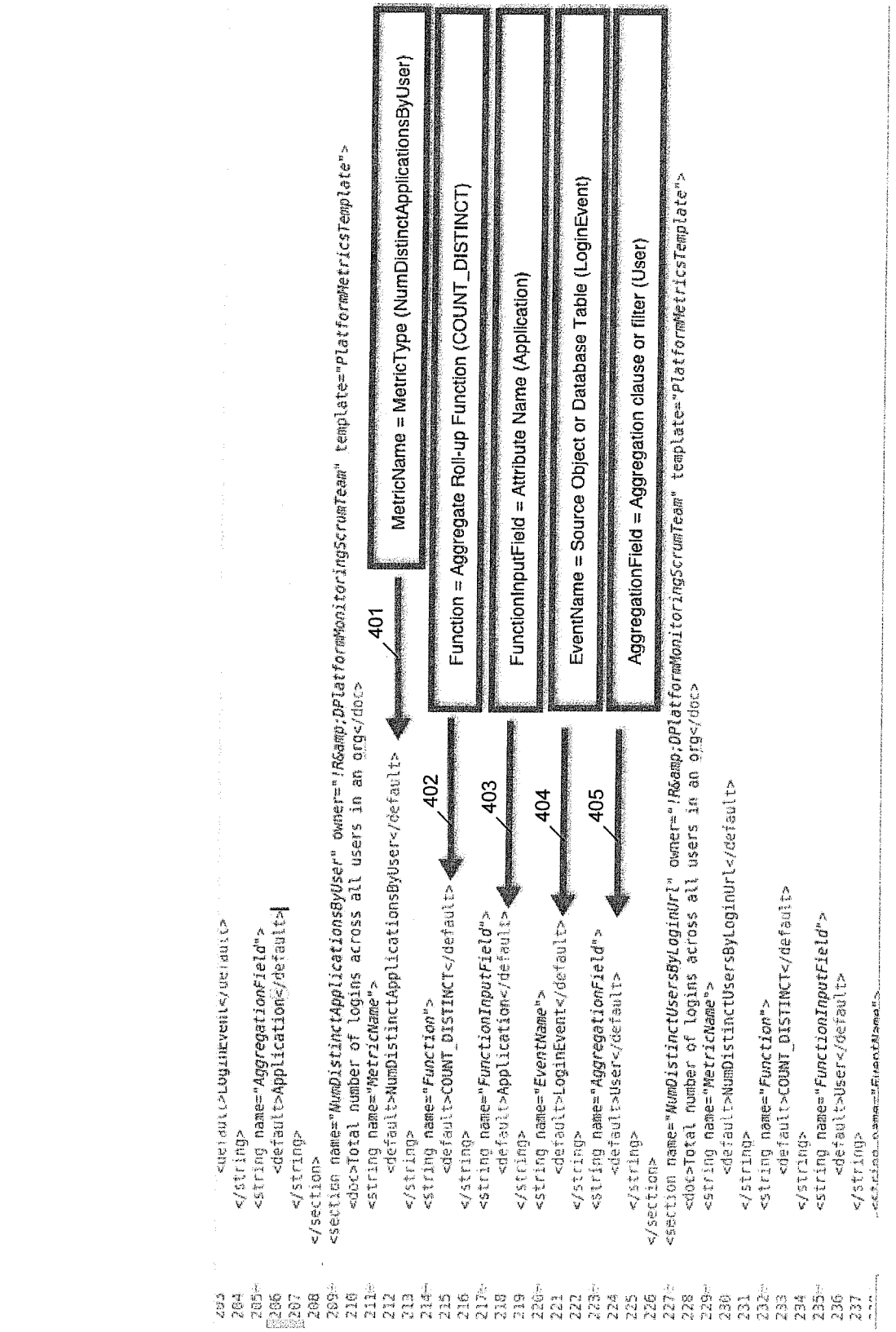
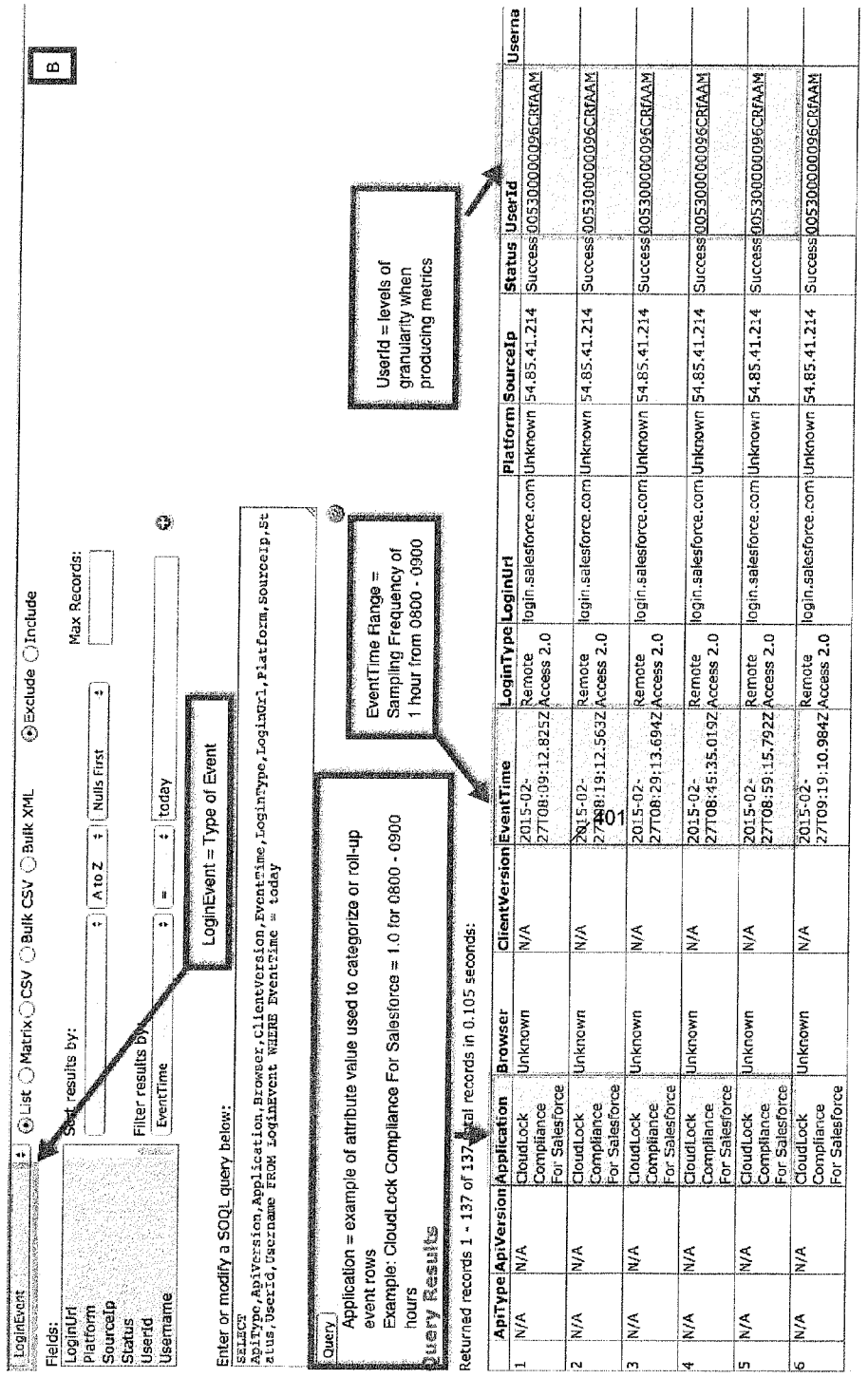

401
MetricName = MetricType (NumDistinctApplicationsByUser)
402
Function = Aggregate Roll-up Function (COUNT_DISTINCT)
403
FunctionInputField = Attribute Name (Application)
404
EventName = Source Object or Database Table (LoginEvent)
405
AggregationField = Aggregation clause or filter (User)

FIGURE 4A




| | ApiType | ApiVersion | Application | Browser | ClientVersion | EventTime | LoginType | LoginUrl | Platform | SourceIp | Status | UserId | Userna |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | N/A | N/A | CloudLock Compliance For Salesforce | Unknown | N/A | 2015-02-27T08:09:12.825Z | Remote Access 2.0 | login.salesforce.com | Unknown | 54.85.41.214 | Success | 00530000009GRfAAM | |
| 2 | N/A | N/A | CloudLock Compliance For Salesforce | Unknown | N/A | 2015-02-27T08:19:12.563Z | Remote Access 2.0 | login.salesforce.com | Unknown | 54.85.41.214 | Success | 00530000009GRfAAM | |
| 3 | N/A | N/A | CloudLock Compliance For Salesforce | Unknown | N/A | 2015-02-27T08:29:13.694Z | Remote Access 2.0 | login.salesforce.com | Unknown | 54.85.41.214 | Success | 00530000009GRfAAM | |
| 4 | N/A | N/A | CloudLock Compliance For Salesforce | Unknown | N/A | 2015-02-27T08:45:35.019Z | Remote Access 2.0 | login.salesforce.com | Unknown | 54.85.41.214 | Success | 00530000009GRfAAM | |
| 5 | N/A | N/A | CloudLock Compliance For Salesforce | Unknown | N/A | 2015-02-27T08:59:15.792Z | Remote Access 2.0 | login.salesforce.com | Unknown | 54.85.41.214 | Success | 00530000009GRfAAM | |
| 6 | N/A | N/A | CloudLock Compliance For Salesforce | Unknown | N/A | 2015-02-27T09:19:10.984Z | Remote Access 2.0 | login.salesforce.com | Unknown | 54.85.41.214 | Success | 00530000009GRfAAM | |

FIGURE 4B

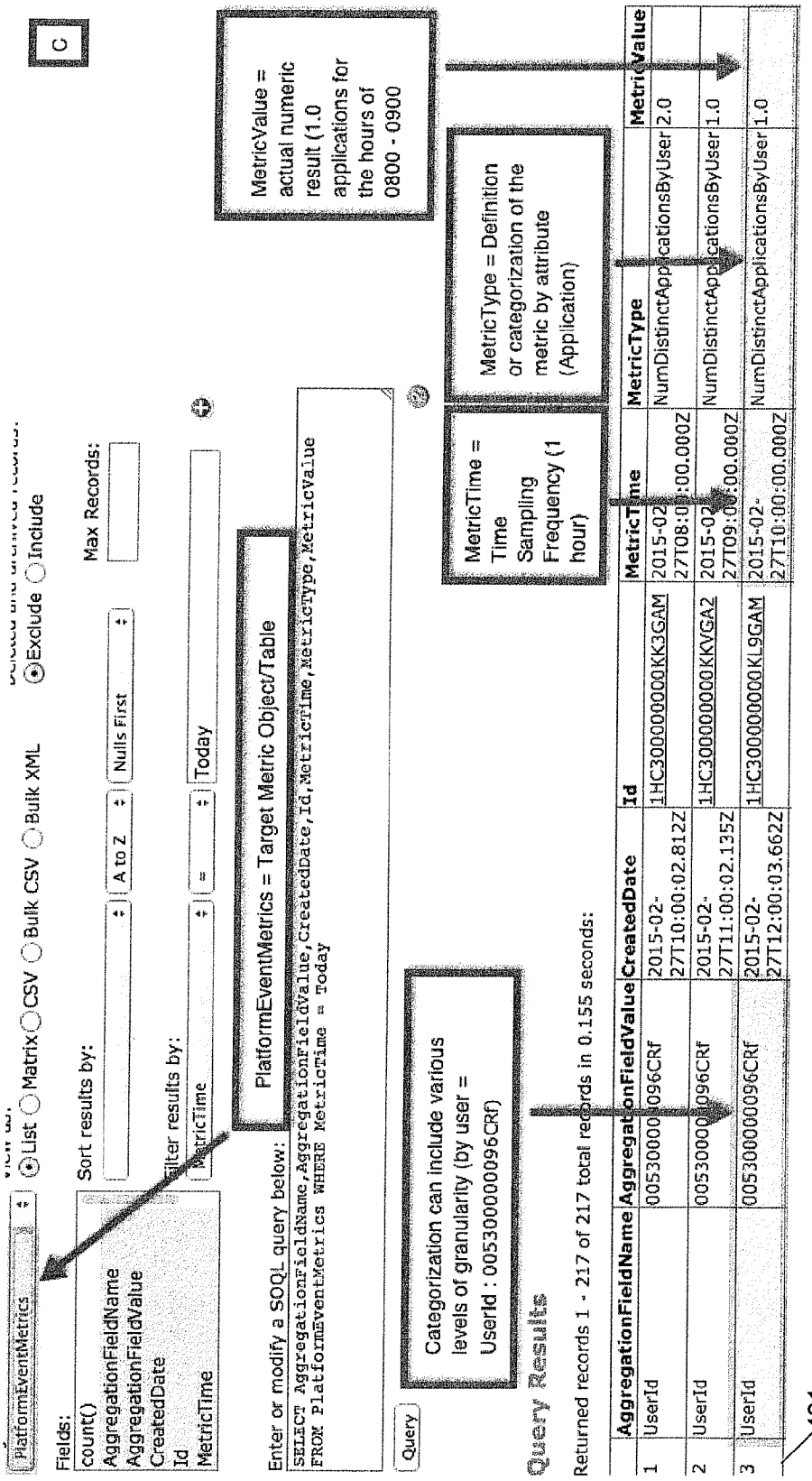

PlatformEventMetrics
List
Matrix
CSV
Bulk CSV
Bulk XML
Exclude
Include
C
Fields:
count()
AggregationFieldName
AggregationFieldValue
CreatedDate
Id
MetricTime
Sort results by:
A to Z
Nulls First
Max Records:
Filter results by:
MetricTime
=
Today
PlatformEventMetrics = Target Metric Object/Table
Enter or modify a SOQL query below:
SELECT AggregationFieldName,AggregationFieldValue,CreatedDate,Id,MetricTime,MetricType,MetricValue
FROM PlatformEventMetrics WHERE MetricTime = Today
MetricValue = actual numeric result (1.0 applications for the hours of 0800 - 0900
MetricType = Definition or categorization of the metric by attribute (Application)
MetricTime = Time Sampling Frequency (1 hour)
Query
Categorization can include various levels of granularity (by user = UserId : 0053000000096CRf)
Query Results
Returned records 1 - 217 of 217 total records in 0.155 seconds:
AggregationFieldName
AggregationFieldValue
CreatedDate
Id
MetricTime
MetricType
MetricValue
1 UserId 0053000000096CRf 2015-02-27T10:00:02.812Z 1HC300000000KK3GAM 2015-02-27T08:00:00.000Z NumDistinctApplicationsByUser 2.0
2 UserId 0053000000096CRf 2015-02-27T11:00:02.135Z 1HC300000000KKVGA2 2015-02-27T09:00:00.000Z NumDistinctApplicationsByUser 1.0
3 UserId 0053000000096CRf 2015-02-27T12:00:03.662Z 1HC300000000KL9GAM 2015-02-27T10:00:00.000Z NumDistinctApplicationsByUser 1.0
421

FIGURE 4C

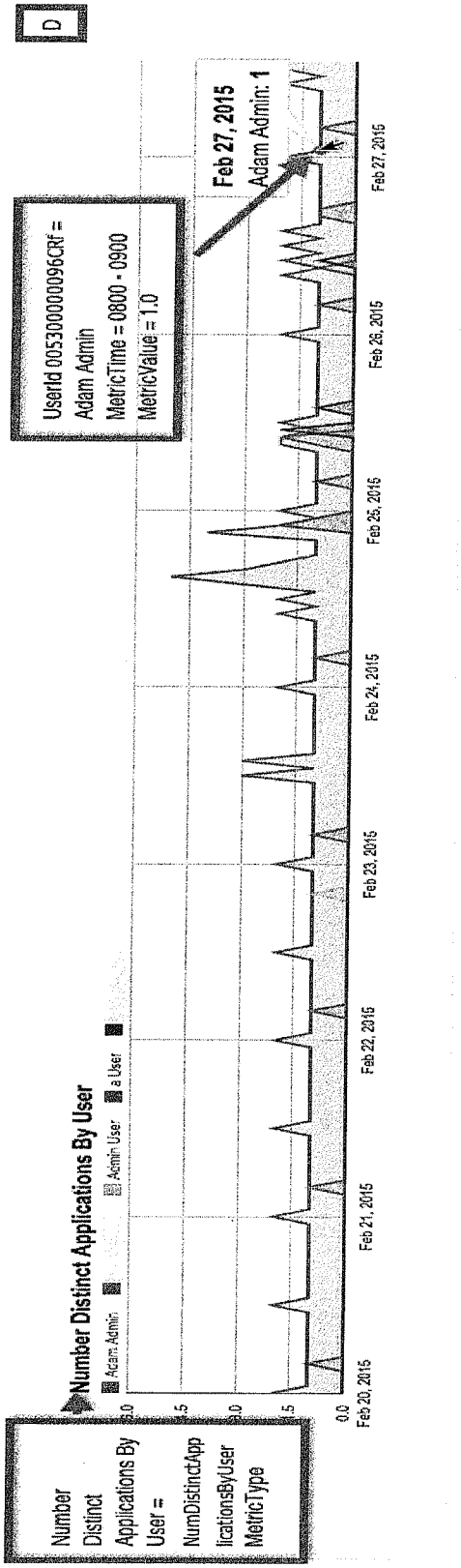
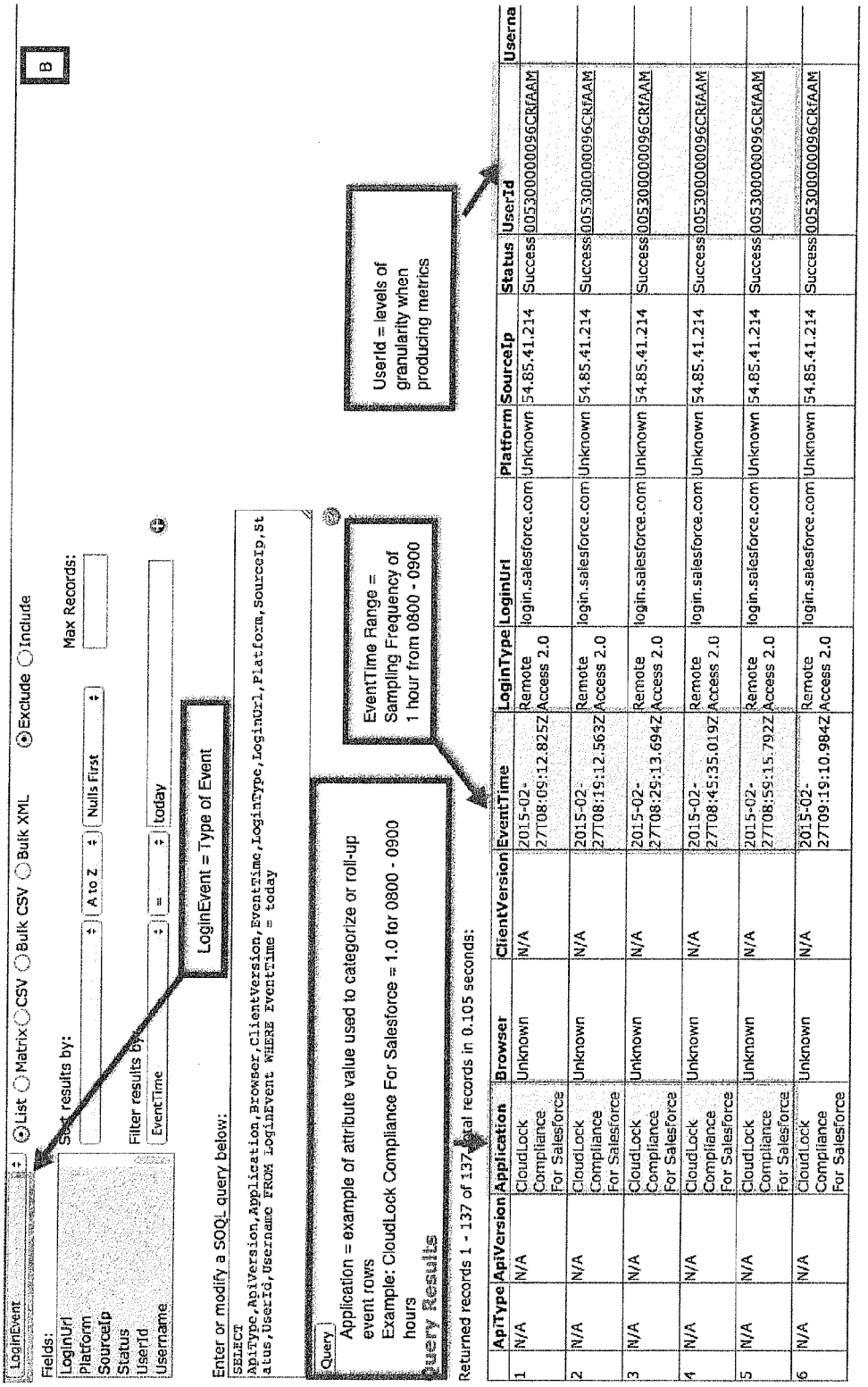

D
Number Distinct Applications By User
Adam Admin
Admin User
a User
UserId 00530000000096CRf =
Adam Admin
MetricTime = 0800 - 0900
MetricValue = 1.0
Feb 27, 2015
Adam Admin: 1
Number Distinct Applications By User = NumDistinctApplicationsByUser MetricType
6.0
4.5
3.0
1.5
0.0
Feb 20, 2015
Feb 21, 2015
Feb 22, 2015
Feb 23, 2015
Feb 24, 2015
Feb 25, 2015
Feb 26, 2015
Feb 27, 2015

FIGURE 4D




| | ApiType | ApiVersion | Application | Browser | ClientVersion | EventTime | LoginType | LoginUrl | Platform | SourceIp | Status | UserId | Userna |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | N/A | N/A | CloudLock Compliance For Salesforce | Unknown | N/A | 2015-02-27T08:09:12.825Z | Remote Access 2.0 | login.salesforce.com | Unknown | 54.85.41.214 | Success | 00530000096CRfAAM | |
| 2 | N/A | N/A | CloudLock Compliance For Salesforce | Unknown | N/A | 2015-02-27T08:19:12.563Z | Remote Access 2.0 | login.salesforce.com | Unknown | 54.85.41.214 | Success | 00530000096CRfAAM | |
| 3 | N/A | N/A | CloudLock Compliance For Salesforce | Unknown | N/A | 2015-02-27T08:29:13.694Z | Remote Access 2.0 | login.salesforce.com | Unknown | 54.85.41.214 | Success | 00530000096CRfAAM | |
| 4 | N/A | N/A | CloudLock Compliance For Salesforce | Unknown | N/A | 2015-02-27T08:45:35.019Z | Remote Access 2.0 | login.salesforce.com | Unknown | 54.85.41.214 | Success | 00530000096CRfAAM | |
| 5 | N/A | N/A | CloudLock Compliance For Salesforce | Unknown | N/A | 2015-02-27T08:59:15.792Z | Remote Access 2.0 | login.salesforce.com | Unknown | 54.85.41.214 | Success | 00530000096CRfAAM | |
| 6 | N/A | N/A | CloudLock Compliance For Salesforce | Unknown | N/A | 2015-02-27T09:19:10.984Z | Remote Access 2.0 | login.salesforce.com | Unknown | 54.85.41.214 | Success | 00530000096CRfAAM | |

*FIGURE 4B*

# EXTRACTING AND PROCESSING METRICS FROM SYSTEM GENERATED EVENTS

## CLAIM OF PRIORITY

**[0001]** This application claims the benefit of U.S. Provisional Patent Application 62/048,961 entitled Extracting and Processing Metrics from System Generated Events, by Aakash Pradeep et al., filed Sep. 11, 2014, the entire contents of which are incorporated herein by reference.

## COPYRIGHT NOTICE



## CROSS REFERENCE TO RELATED APPLICATIONS

**[0003]** The following commonly owned, co-pending non-provisional United States Patents and Patent Applications, including the present application, may be related to each other. Each of the other patents/applications are incorporated by reference herein in its entirety:

**[0004]** U.S. patent application Ser. No. 14/688,917 entitled EXTRACTION AND CAPTURE OF INFORMATION FROM CUSTOMIZABLE HEADER, filed Apr. 16, 2015.

## FIELD OF THE INVENTION

**[0005]** One or more implementations relate generally to extracting and processing metrics from system generated events in a database network system environment.

## BACKGROUND

**[0006]** The subject matter discussed in the background section should not be assumed to be prior art merely as a result of its mention in the background section. Similarly, a problem mentioned in the background section or associated with the subject matter of the background section should not be assumed to have been previously recognized in the prior art. The subject matter in the background section merely represents different approaches, which in and of themselves may also be inventions.

**[0007]** Many multi-tenant database system today produce system generated events such as logins, downloads, transactions, or anything that can be construed as an application event associated with a specific customer instance. These systems provide for various forms of electronic communication between customers and host system.

**[0008]** It has become common practice for institutions to extract electronic communication information from these systems. Given the affordability of data storage and the potential necessity of such information, many institutions retain electronic copies of information. For example, in industries such as financial and legal services as well as many others, it is often important to maintain a "paper trail". Information such as what information is being accessed by whom, and when it is being accessed may be required for audits or investigations. In certain cases, electronic records may even be subpoenaed, and file storage devices that store such information may be subject to computer forensic searches.

## BRIEF DESCRIPTION OF THE DRAWINGS

**[0009]** In the following drawings like reference numbers are used to refer to like elements. Although the following figures depict various examples, the one or more implementations are not limited to the examples depicted in the figures.

**[0010]** The included drawings are for illustrative purposes and serve to provide examples of possible structures and operations for the disclosed inventive systems, apparatus, methods and computer-readable storage media. These drawings in no way limit any changes in form and detail that may be made by one skilled in the art without departing from the spirit and scope of the disclosed implementations.

**[0011]** FIG. **1**A shows a block diagram of an example environment in which an on-demand database service can be used according to some implementations.

**[0012]** FIG. **1**B shows a block diagram of example implementations of elements of FIG. **1**A and example interconnections between these elements according to some implementations.

**[0013]** FIG. **2**A shows a system diagram of example architectural components of an on-demand database service environment according to some implementations.

**[0014]** FIG. **2**B shows a system diagram further illustrating example architectural components of an on-demand database service environment according to some implementations.

**[0015]** FIG. **3** is an operational flow diagram illustrating a process that may be performed by a processing system of a database system for extracting and processing metrics from system generated events.

**[0016]** FIG. **4**A is an example of code to define a metric.

**[0017]** FIG. **4**B is an example of raw event data.

**[0018]** FIG. **4**C is an example of a graphical user interface to query a platform event metric object.

**[0019]** FIG. **4**D is an example of viewing a result of the query to the platform event metric object using a visualization tool.

## DETAILED DESCRIPTION

**[0020]** Examples of systems, apparatus, computer-readable storage media, and methods according to the disclosed implementations are described in this section. These examples are being provided solely to add context and aid in the understanding of the disclosed implementations. It will thus be apparent to one skilled in the art that the disclosed implementations may be practiced without some or all of the specific details provided. In other instances, certain process or method operations, also referred to herein as "blocks," have not been described in detail in order to avoid unnecessarily obscuring the disclosed implementations. Other implementations and applications also are possible, and as such, the following examples should not be taken as definitive or limiting either in scope or setting.

**[0021]** In the following detailed description, references are made to the accompanying drawings, which form a part of the description and in which are shown, by way of illustration, specific implementations. Although these disclosed implementations are described in sufficient detail to enable one skilled in the art to practice the implementations, it is to be understood that these examples are not limiting, such that other implementations may be used and changes may be

made to the disclosed implementations without departing from their spirit and scope. For example, the blocks of the methods shown and described herein are not necessarily performed in the order indicated in some other implementations. Additionally, in some other implementations, the disclosed methods may include more or fewer blocks than are described. As another example, some blocks described herein as separate blocks may be combined in some other implementations. Conversely, what may be described herein as a single block may be implemented in multiple blocks in some other implementations. Additionally, the conjunction "or" is intended herein in the inclusive sense where appropriate unless otherwise indicated; that is, the phrase "A, B or C" is intended to include the possibilities of "A," "B," "C," "A and B," "B and C," "A and C" and "A, B and C."

**[0022]** Some implementations described and referenced herein are directed to systems, apparatus, computer-implemented methods and computer-readable storage media for extracting and processing metrics from system generated events.

**[0023]** In some implementations, a processing system of a database system is provided. The processing system may extract and store certain content from users interacting with a multi-tenant database system. The processing system may also determine a profile as well as an authenticity of customer users accessing information of the multi-tenant database system. The processing system may also process extracted metrics to determine a profile of the user accessing the information. The profile may be used, for example, to determine whether a particular access by a purported user may be an access by a different user, based on the profile of the purported user.

**[0024]** In some embodiments, the information generated for each customer instance is captured passively. That is, the information is captured without any interruption to the user. The information may also be stored on a database and examined at a later time by an administrator. For example, a generic Application Program Interface (API) may capture the information and store it on a distributed database, such as a non-relational, distributed database. In some instances, the information may be retrieved from the database and processed. This processing of the information may help product a "fingerprint" that identifies and/or authenticates a user. In some implementations, for each user instance, information such as user identification, time stamp, browser used, version of the browser, IP address, referring URL and an assortment of other information may be collected and stored. Once collected, the processing system may categorize the information to produce a variety of metrics. A metric may be used individually or in combination with another metric in a requested query by, or for, a tenant.

**[0025]** In some embodiments, the processing system may identify an aggregate function on an event attribute for a duration. The processing system may identify a threshold according to available aggregation function results. The processing system may compare a new aggregate function result to the identified threshold. The processing system may transmit an alert responsive to a result of the comparison.

**[0026]** In some embodiments, one example metric for categorization may be the number of instances of logins per user. The instances of logins may be processed to determine an average number of login events. After an average has been determined, a graph may be produced where outliers may be easily determined based on the graphical depiction of the results. In some cases, excessive number of login events may suggest a security breach. That is, if the average number of logins per day for a user is five, and one user is determined to have logged in 15 times, an indication of a potential security breach may be suggested since there are not many instances where that many additional logins are necessary. In some embodiments the processing system may provide an alert in such an instance, where the number of logins exceeds a predetermined threshold amount based on an average number of logins.

**[0027]** Another example metric for categorization may be the hours when a user is logged in. In general, one can expect users to be logged in during work hours, typically between 9 am and 5 pm, or thereabouts. Any instances of outliers, again, may suggest a security breach that triggers an alert. However, work hours may vary from user to user. Thus, while a common trend may exist among most users, this common trend may not be representative of all users. Accordingly, it may be helpful to determine not only a trend among a set of users, but the trend of a single user over a period of time, since a single user's daily activity may be inconsistent from a group of users with which the single user is associated.

**[0028]** Another example metric for categorization may be the location of a user by way of IP addresses. Since IP addresses provide an indication of the location of a user, analyzing and graphing a user based on IP address ranges may provide an indication of the authenticity of each instance of the user login. For example, when a user is determine to have logged in from an IP address associated with a first location, and then logs in a short time later from an IP address associated with a second location that is far removed from the first location, it is likely that the login was performed by two different people using a same account, thus suggesting a security breach. Similarly, the use of two different types of browsers over a short period of time may also suggest different users logging into the same account. Typically, users have the habit of using the same browser on a personal home or work computer. Thus, a change in the type of browser, e.g. brand, or the version of the browser may suggest multiple users. Both instances may trigger an alert being provided.

**[0029]** In some implementations, the users described herein are users (or "members") of an interactive online "enterprise social network," also referred to herein as an "enterprise social networking system," an "enterprise collaborative network," or more simply as an "enterprise network." Such online enterprise networks are increasingly becoming a common way to facilitate communication among people, any of whom can be recognized as enterprise users. One example of an online enterprise social network is Chatter®, provided by salesforce.com, inc. of San Francisco, Calif. salesforce.com, inc. is a provider of enterprise social networking services, customer relationship management (CRM) services and other database management services, any of which can be accessed and used in conjunction with the techniques disclosed herein in some implementations. These various services can be provided in a cloud computing environment as described herein, for example, in the context of a multi-tenant database system. Some of the described techniques or processes can be implemented without having to install software locally, that is, on computing devices of users interacting with services available through the cloud. While the disclosed implementations may be described with reference to Chatter® and more generally to enterprise social networking, those of ordinary skill in the art should under-

stand that the disclosed techniques are neither limited to Chatter® nor to any other services and systems provided by salesforce.com, inc. and can be implemented in the context of various other database systems such as cloud-based systems that are not part of a multi-tenant database system or which do not provide enterprise social networking services.

I. Example System Overview

**[0030]** FIG. **1**A shows a block diagram of an example of an environment **10** in which an on-demand database service can be used in accordance with some implementations. The environment **10** includes user systems **12**, a network **14**, a database system **16** (also referred to herein as a "cloud-based system"), a processor system **17**, an application platform **18**, a network interface **20**, tenant database **22** for storing tenant data **23**, system database **24** for storing system data **25**, program code **26** for implementing various functions of the system **16**, and process space **28** for executing database system processes and tenant-specific processes, such as running applications as part of an application hosting service. In some other implementations, environment **10** may not have all of these components or systems, or may have other components or systems instead of, or in addition to, those listed above.

**[0031]** In some implementations, the environment **10** is an environment in which an on-demand database service exists. An on-demand database service, such as that which can be implemented using the system **16**, is a service that is made available to users outside of the enterprise(s) that own, maintain or provide access to the system **16**. As described above, such users generally do not need to be concerned with building or maintaining the system **16**. Instead, resources provided by the system **16** may be available for such users' use when the users need services provided by the system **16**; that is, on the demand of the users. Some on-demand database services can store information from one or more tenants into tables of a common database image to form a multi-tenant database system (MTS). The term "multi-tenant database system" can refer to those systems in which various elements of hardware and software of a database system may be shared by one or more customers or tenants. For example, a given application server may simultaneously process requests for a great number of customers, and a given database table may store rows of data such as feed items for a potentially much greater number of customers. A database image can include one or more database objects. A relational database management system (RDBMS) or the equivalent can execute storage and retrieval of information against the database object(s).

**[0032]** Application platform **18** can be a framework that allows the applications of system **16** to execute, such as the hardware or software infrastructure of the system **16**. In some implementations, the application platform **18** enables the creation, management and execution of one or more applications developed by the provider of the on-demand database service, users accessing the on-demand database service via user systems **12**, or third party application developers accessing the on-demand database service via user systems **12**.

**[0033]** In some implementations, the system **16** implements a web-based customer relationship management (CRM) system. For example, in some such implementations, the system **16** includes application servers configured to implement and execute CRM software applications as well as provide related data, code, forms, renderable web pages and documents and other information to and from user systems **12** and to store to, and retrieve from, a database system related data, objects, and Web page content. In some MTS implementations, data for multiple tenants may be stored in the same physical database object in tenant database **22**. In some such implementations, tenant data is arranged in the storage medium(s) of tenant database **22** so that data of one tenant is kept logically separate from that of other tenants so that one tenant does not have access to another tenant's data, unless such data is expressly shared. The system **16** also implements applications other than, or in addition to, a CRM application. For example, the system **16** can provide tenant access to multiple hosted (standard and custom) applications, including a CRM application. User (or third party developer) applications, which may or may not include CRM, may be supported by the application platform **18**. The application platform **18** manages the creation and storage of the applications into one or more database objects and the execution of the applications in one or more virtual machines in the process space of the system **16**.

**[0034]** According to some implementations, each system **16** is configured to provide web pages, forms, applications, data and media content to user (client) systems **12** to support the access by user systems **12** as tenants of system **16**. As such, system **16** provides security mechanisms to keep each tenant's data separate unless the data is shared. If more than one MTS is used, they may be located in close proximity to one another (for example, in a server farm located in a single building or campus), or they may be distributed at locations remote from one another (for example, one or more servers located in city A and one or more servers located in city B). As used herein, each MTS could include one or more logically or physically connected servers distributed locally or across one or more geographic locations. Additionally, the term "server" is meant to refer to a computing device or system, including processing hardware and process space(s), an associated storage medium such as a memory device or database, and, in some instances, a database application (for example, OODBMS or RDBMS) as is well known in the art. It should also be understood that "server system" and "server" are often used interchangeably herein. Similarly, the database objects described herein can be implemented as part of a single database, a distributed database, a collection of distributed databases, a database with redundant online or offline backups or other redundancies, etc., and can include a distributed database or storage network and associated processing intelligence.

**[0035]** The network **14** can be or include any network or combination of networks of systems or devices that communicate with one another. For example, the network **14** can be or include any one or any combination of a LAN (local area network), WAN (wide area network), telephone network, wireless network, cellular network, point-to-point network, star network, token ring network, hub network, or other appropriate configuration. The network **14** can include a TCP/IP (Transfer Control Protocol and Internet Protocol) network, such as the global internetwork of networks often referred to as the "Internet" (with a capital "I"). The Internet will be used in many of the examples herein. However, it should be understood that the networks that the disclosed implementations can use are not so limited, although TCP/IP is a frequently implemented protocol.

**[0036]** The user systems **12** can communicate with system **16** using TCP/IP and, at a higher network level, other common Internet protocols to communicate, such as HTTP, FTP, AFS, WAP, etc. In an example where HTTP is used, each user

system **12** can include an HTTP client commonly referred to as a "web browser" or simply a "browser" for sending and receiving HTTP signals to and from an HTTP server of the system **16**. Such an HTTP server can be implemented as the sole network interface **20** between the system **16** and the network **14**, but other techniques can be used in addition to or instead of these techniques. In some implementations, the network interface **20** between the system **16** and the network **14** includes load sharing functionality, such as round-robin HTTP request distributors to balance loads and distribute incoming HTTP requests evenly over a number of servers. In MTS implementations, each of the servers can have access to the MTS data; however, other alternative configurations may be used instead.

**[0037]** The user systems **12** can be implemented as any computing device(s) or other data processing apparatus or systems usable by users to access the database system **16**. For example, any of user systems **12** can be a desktop computer, a work station, a laptop computer, a tablet computer, a handheld computing device, a mobile cellular phone (for example, a "smartphone"), or any other Wi-Fi-enabled device, wireless access protocol (WAP)-enabled device, or other computing device capable of interfacing directly or indirectly to the Internet or other network. The terms "user system" and "computing device" are used interchangeably herein with one another and with the term "computer." As described above, each user system **12** typically executes an HTTP client, for example, a web browsing (or simply "browsing") program, such as a web browser based on the WebKit platform, Microsoft's Internet Explorer browser, Netscape's Navigator browser, Opera's browser, Mozilla's Firefox browser, or a WAP-enabled browser in the case of a cellular phone, PDA or other wireless device, or the like, allowing a user (for example, a subscriber of on-demand services provided by the system **16**) of the user system **12** to access, process and view information, pages and applications available to it from the system **16** over the network **14**.

**[0038]** Each user system **12** also typically includes one or more user input devices, such as a keyboard, a mouse, a trackball, a touch pad, a touch screen, a pen or stylus or the like, for interacting with a graphical user interface (GUI) provided by the browser on a display (for example, a monitor screen, liquid crystal display (LCD), light-emitting diode (LED) display, among other possibilities) of the user system **12** in conjunction with pages, forms, applications and other information provided by the system **16** or other systems or servers. For example, the user interface device can be used to access data and applications hosted by system **16**, and to perform searches on stored data, and otherwise allow a user to interact with various GUI pages that may be presented to a user. As discussed above, implementations are suitable for use with the Internet, although other networks can be used instead of or in addition to the Internet, such as an intranet, an extranet, a virtual private network (VPN), a non-TCP/IP based network, any LAN or WAN or the like.

**[0039]** The users of user systems **12** may differ in their respective capacities, and the capacity of a particular user system **12** can be entirely determined by permissions (permission levels) for the current user of such user system. For example, where a salesperson is using a particular user system **12** to interact with the system **16**, that user system can have the capacities allotted to the salesperson. However, while an administrator is using that user system **12** to interact with the system **16**, that user system can have the capacities allotted to that administrator. Where a hierarchical role model is used, users at one permission level can have access to applications, data, and database information accessible by a lower permission level user, but may not have access to certain applications, database information, and data accessible by a user at a higher permission level. Thus, different users generally will have different capabilities with regard to accessing and modifying application and database information, depending on the users' respective security or permission levels (also referred to as "authorizations").

**[0040]** According to some implementations, each user system **12** and some or all of its components are operator-configurable using applications, such as a browser, including computer code executed using a central processing unit (CPU) such as an Intel Pentium® processor or the like. Similarly, the system **16** (and additional instances of an MTS, where more than one is present) and all of its components can be operator-configurable using application(s) including computer code to run using the processor system **17**, which may be implemented to include a CPU, which may include an Intel Pentium® processor or the like, or multiple CPUs.

**[0041]** The system **16** includes tangible computer-readable media having non-transitory instructions stored thereon/in that are executable by or used to program a server or other computing system (or collection of such servers or computing systems) to perform some of the implementation of processes described herein. For example, computer program code **26** can implement instructions for operating and configuring the system **16** to intercommunicate and to process web pages, applications and other data and media content as described herein. In some implementations, the computer code **26** can be downloadable and stored on a hard disk, but the entire program code, or portions thereof, also can be stored in any other volatile or non-volatile memory medium or device as is well known, such as a ROM or RAM, or provided on any media capable of storing program code, such as any type of rotating media including floppy disks, optical discs, digital versatile disks (DVD), compact disks (CD), microdrives, and magneto-optical disks, and magnetic or optical cards, nanosystems (including molecular memory ICs), or any other type of computer-readable medium or device suitable for storing instructions or data. Additionally, the entire program code, or portions thereof, may be transmitted and downloaded from a software source over a transmission medium, for example, over the Internet, or from another server, as is well known, or transmitted over any other existing network connection as is well known (for example, extranet, VPN, LAN, etc.) using any communication medium and protocols (for example, TCP/IP, HTTP, HTTPS, Ethernet, etc.) as are well known. It will also be appreciated that computer code for the disclosed implementations can be realized in any programming language that can be executed on a server or other computing system such as, for example, C, C++, HTML, any other markup language, Java™, JavaScript, ActiveX, any other scripting language, such as VBScript, and many other programming languages as are well known may be used. (Java™ is a trademark of Sun Microsystems, Inc.).

**[0042]** FIG. **1**B shows a block diagram of example implementations of elements of FIG. **1**A and example interconnections between these elements according to some implementations. That is, FIG. **1**B also illustrates environment **10**, but FIG. **1**B, various elements of the system **16** and various interconnections between such elements are shown with more specificity according to some more specific implementations.

Additionally, in FIG. **1**B, the user system **12** includes a processor system **12**A, a memory system **12**B, an input system **12**C, and an output system **12**D. The processor system **12**A can include any suitable combination of one or more processors. The memory system **12**B can include any suitable combination of one or more memory devices. The input system **12**C can include any suitable combination of input devices, such as one or more touchscreen interfaces, keyboards, mice, trackballs, scanners, cameras, or interfaces to networks. The output system **12**D can include any suitable combination of output devices, such as one or more display devices, printers, or interfaces to networks.

**[0043]** In FIG. **1**B, the network interface **20** is implemented as a set of HTTP application servers $\mathbf{100}_1$-$\mathbf{100}_N$. Each application server **100**, also referred to herein as an "app server", is configured to communicate with tenant database **22** and the tenant data **23** therein, as well as system database **24** and the system data **25** therein, to serve requests received from the user systems **12**. The tenant data **23** can be divided into individual tenant storage spaces **112**, which can be physically or logically arranged or divided. Within each tenant storage space **112**, user storage **114** and application metadata **116** can similarly be allocated for each user. For example, a copy of a user's most recently used (MRU) items can be stored to user storage **114**. Similarly, a copy of MRU items for an entire organization that is a tenant can be stored to tenant storage space **112**.

**[0044]** The process space **28** includes system process space **102**, individual tenant process spaces **104** and a tenant management process space **110**. The application platform **18** includes an application setup mechanism **38** that supports application developers' creation and management of applications. Such applications and others can be saved as metadata into tenant database **22** by save routines **36** for execution by subscribers as one or more tenant process spaces **104** managed by tenant management process **110**, for example. Invocations to such applications can be coded using PL/SOQL **34**, which provides a programming language style interface extension to API **32**. A detailed description of some PL/SOQL language implementations is discussed in commonly assigned U.S. Pat. No. 7,730,478, titled METHOD AND SYSTEM FOR ALLOWING ACCESS TO DEVELOPED APPLICATIONS VIA A MULTI-TENANT ON-DEMAND DATABASE SERVICE, by Craig Weissman, issued on Jun. 1, 2010, and hereby incorporated by reference in its entirety and for all purposes. Invocations to applications can be detected by one or more system processes, which manage retrieving application metadata **116** for the subscriber making the invocation and executing the metadata as an application in a virtual machine.

**[0045]** The system **16** of FIG. **1**B also includes a user interface (UI) **30** and an application programming interface (API) **32** to system **16** resident processes to users or developers at user systems **12**. In some other implementations, the environment **10** may not have the same elements as those listed above or may have other elements instead of, or in addition to, those listed above.

**[0046]** Each application server **100** can be communicably coupled with tenant database **22** and system database **24**, for example, having access to tenant data **23** and system data **25**, respectively, via a different network connection. For example, one application server $\mathbf{100}_1$ can be coupled via the network **14** (for example, the Internet), another application server $\mathbf{100}_{N-1}$ can be coupled via a direct network link, and another application server $\mathbf{100}_N$ can be coupled by yet a different network connection. Transfer Control Protocol and Internet Protocol (TCP/IP) are examples of typical protocols that can be used for communicating between application servers **100** and the system **16**. However, it will be apparent to one skilled in the art that other transport protocols can be used to optimize the system **16** depending on the network interconnections used.

**[0047]** In some implementations, each application server **100** is configured to handle requests for any user associated with any organization that is a tenant of the system **16**. Because it can be desirable to be able to add and remove application servers **100** from the server pool at any time and for various reasons, in some implementations there is no server affinity for a user or organization to a specific application server **100**. In some such implementations, an interface system implementing a load balancing function (for example, an F5 Big-IP load balancer) is communicably coupled between the application servers **100** and the user systems **12** to distribute requests to the application servers **100**. In one implementation, the load balancer uses a least-connections algorithm to route user requests to the application servers **100**. Other examples of load balancing algorithms, such as round robin and observed-response-time, also can be used. For example, in some instances, three consecutive requests from the same user could hit three different application servers **100**, and three requests from different users could hit the same application server **100**. In this manner, by way of example, system **16** can be a multi-tenant system in which system **16** handles storage of, and access to, different objects, data and applications across disparate users and organizations.

**[0048]** In one example storage use case, one tenant can be a company that employs a sales force where each salesperson uses system **16** to manage aspects of their sales. A user can maintain contact data, leads data, customer follow-up data, performance data, goals and progress data, etc., all applicable to that user's personal sales process (for example, in tenant database **22**). In an example of a MTS arrangement, because all of the data and the applications to access, view, modify, report, transmit, calculate, etc., can be maintained and accessed by a user system **12** having little more than network access, the user can manage his or her sales efforts and cycles from any of many different user systems. For example, when a salesperson is visiting a customer and the customer has Internet access in their lobby, the salesperson can obtain critical updates regarding that customer while waiting for the customer to arrive in the lobby.

**[0049]** While each user's data can be stored separately from other users' data regardless of the employers of each user, some data can be organization-wide data shared or accessible by several users or all of the users for a given organization that is a tenant. Thus, there can be some data structures managed by system **16** that are allocated at the tenant level while other data structures can be managed at the user level. Because an MTS can support multiple tenants including possible competitors, the MTS can have security protocols that keep data, applications, and application use separate. Also, because many tenants may opt for access to an MTS rather than maintain their own system, redundancy, up-time, and backup are additional functions that can be implemented in the MTS. In addition to user-specific data and tenant-specific data, the system **16** also can maintain system level data usable by

multiple tenants or other data. Such system level data can include industry reports, news, postings, and the like that are sharable among tenants.

**[0050]** In some implementations, the user systems **12** (which also can be client systems) communicate with the application servers **100** to request and update system-level and tenant-level data from the system **16**. Such requests and updates can involve sending one or more queries to tenant database **22** or system database **24**. The system **16** (for example, an application server **100** in the system **16**) can automatically generate one or more Structured Query Language (SQL) statements (for example, one or more SQL queries) designed to access the desired information. System database **24** can generate query plans to access the requested data from the database. The term "query plan" generally refers to one or more operations used to access information in a database system.

**[0051]** Each database can generally be viewed as a collection of objects, such as a set of logical tables, containing data fitted into predefined or customizable categories. A "table" is one representation of a data object, and may be used herein to simplify the conceptual description of objects and custom objects according to some implementations. It should be understood that "table" and "object" may be used interchangeably herein. Each table generally contains one or more data categories logically arranged as columns or fields in a viewable schema. Each row or element of a table can contain an instance of data for each category defined by the fields. For example, a CRM database can include a table that describes a customer with fields for basic contact information such as name, address, phone number, fax number, etc. Another table can describe a purchase order, including fields for information such as customer, product, sale price, date, etc. In some MTS implementations, standard entity tables can be provided for use by all tenants. For CRM database applications, such standard entities can include tables for case, account, contact, lead, and opportunity data objects, each containing pre-defined fields. As used herein, the term "entity" also may be used interchangeably with "object" and "table."

**[0052]** In some MTS implementations, tenants are allowed to create and store custom objects, or may be allowed to customize standard entities or objects, for example by creating custom fields for standard objects, including custom index fields. Commonly assigned U.S. Pat. No. 7,779,039, titled CUSTOM ENTITIES AND FIELDS IN A MULTI-TENANT DATABASE SYSTEM, by Weissman et al., issued on Aug. 17, 2010, and hereby incorporated by reference in its entirety and for all purposes, teaches systems and methods for creating custom objects as well as customizing standard objects in a multi-tenant database system. In some implementations, for example, all custom entity data rows are stored in a single multi-tenant physical table, which may contain multiple logical tables per organization. It is transparent to customers that their multiple "tables" are in fact stored in one large table or that their data may be stored in the same table as the data of other customers.

**[0053]** FIG. **2**A shows a system diagram illustrating example architectural components of an on-demand database service environment **200** according to some implementations. A client machine communicably connected with the cloud **204**, generally referring to one or more networks in combination, as described herein, can communicate with the on-demand database service environment **200** via one or more edge routers **208** and **212**. A client machine can be any of the examples of user systems **12** described above. The edge routers can communicate with one or more core switches **220** and **224** through a firewall **216**. The core switches can communicate with a load balancer **228**, which can distribute server load over different pods, such as the pods **240** and **244**. The pods **240** and **244**, which can each include one or more servers or other computing resources, can perform data processing and other operations used to provide on-demand services. Communication with the pods can be conducted via pod switches **232** and **236**. Components of the on-demand database service environment can communicate with database storage **256** through a database firewall **248** and a database switch **252**.

**[0054]** As shown in FIGS. **2**A and **2**B, accessing an on-demand database service environment can involve communications transmitted among a variety of different hardware or software components. Further, the on-demand database service environment **200** is a simplified representation of an actual on-demand database service environment. For example, while only one or two devices of each type are shown in FIGS. **2**A and **2**B, some implementations of an on-demand database service environment can include anywhere from one to several devices of each type. Also, the on-demand database service environment need not include each device shown in FIGS. **2**A and **2**B, or can include additional devices not shown in FIGS. **2**A and **2**B.

**[0055]** Additionally, it should be appreciated that one or more of the devices in the on-demand database service environment **200** can be implemented on the same physical device or on different hardware. Some devices can be implemented using hardware or a combination of hardware and software. Thus, terms such as "data processing apparatus," "machine," "server" and "device" as used herein are not limited to a single hardware device, rather references to these terms can include any suitable combination of hardware and software configured to provide the described functionality.

**[0056]** The cloud **204** is intended to refer to a data network or multiple data networks, often including the Internet. Client machines communicably connected with the cloud **204** can communicate with other components of the on-demand database service environment **200** to access services provided by the on-demand database service environment. For example, client machines can access the on-demand database service environment to retrieve, store, edit, or process information. In some implementations, the edge routers **208** and **212** route packets between the cloud **204** and other components of the on-demand database service environment **200**. For example, the edge routers **208** and **212** can employ the Border Gateway Protocol (BGP). The BGP is the core routing protocol of the Internet. The edge routers **208** and **212** can maintain a table of IP networks or 'prefixes', which designate network reachability among autonomous systems on the Internet.

**[0057]** In some implementations, the firewall **216** can protect the inner components of the on-demand database service environment **200** from Internet traffic. The firewall **216** can block, permit, or deny access to the inner components of the on-demand database service environment **200** based upon a set of rules and other criteria. The firewall **216** can act as one or more of a packet filter, an application gateway, a stateful filter, a proxy server, or any other type of firewall.

**[0058]** In some implementations, the core switches **220** and **224** are high-capacity switches that transfer packets within the on-demand database service environment **200**. The core switches **220** and **224** can be configured as network bridges that quickly route data between different components within

the on-demand database service environment. In some implementations, the use of two or more core switches **220** and **224** can provide redundancy or reduced latency.

**[0059]** In some implementations, the pods **240** and **244** perform the core data processing and service functions provided by the on-demand database service environment. Each pod can include various types of hardware or software computing resources. An example of the pod architecture is discussed in greater detail with reference to FIG. **2**B. In some implementations, communication between the pods **240** and **244** is conducted via the pod switches **232** and **236**. The pod switches **232** and **236** can facilitate communication between the pods **240** and **244** and client machines communicably connected with the cloud **204**, for example via core switches **220** and **224**. Also, the pod switches **232** and **236** may facilitate communication between the pods **240** and **244** and the database storage **256**. In some implementations, the load balancer **228** can distribute workload between the pods **240** and **244**. Balancing the on-demand service requests between the pods can assist in improving the use of resources, increasing throughput, reducing response times, or reducing overhead. The load balancer **228** may include multilayer switches to analyze and forward traffic.

**[0060]** In some implementations, access to the database storage **256** is guarded by a database firewall **248**. The database firewall **248** can act as a computer application firewall operating at the database application layer of a protocol stack. The database firewall **248** can protect the database storage **256** from application attacks such as structure query language (SQL) injection, database rootkits, and unauthorized information disclosure. In some implementations, the database firewall **248** includes a host using one or more forms of reverse proxy services to proxy traffic before passing it to a gateway router. The database firewall **248** can inspect the contents of database traffic and block certain content or database requests. The database firewall **248** can work on the SQL application level atop the TCP/IP stack, managing applications' connection to the database or SQL management interfaces as well as intercepting and enforcing packets traveling to or from a database network or application interface.

**[0061]** In some implementations, communication with the database storage **256** is conducted via the database switch **252**. The multi-tenant database storage **256** can include more than one hardware or software components for handling database queries. Accordingly, the database switch **252** can direct database queries transmitted by other components of the on-demand database service environment (for example, the pods **240** and **244**) to the correct components within the database storage **256**. In some implementations, the database storage **256** is an on-demand database system shared by many different organizations as described above with reference to FIGS. **1**A and **1**B.

**[0062]** FIG. **2**B shows a system diagram further illustrating example architectural components of an on-demand database service environment according to some implementations. The pod **244** can be used to render services to a user of the on-demand database service environment **200**. In some implementations, each pod includes a variety of servers or other systems. The pod **244** includes one or more content batch servers **264**, content search servers **268**, query servers **282**, file force servers **286**, access control system (ACS) servers **280**, batch servers **284**, and app servers **288**. The pod **244** also can include database instances **290**, quick file systems (QFS) **292**, and indexers **294**. In some implementations, some or all communication between the servers in the pod **244** can be transmitted via the switch **236**.

**[0063]** In some implementations, the app servers **288** include a hardware or software framework dedicated to the execution of procedures (for example, programs, routines, scripts) for supporting the construction of applications provided by the on-demand database service environment **200** via the pod **244**. In some implementations, the hardware or software framework of an app server **288** is configured to execute operations of the services described herein, including performance of the blocks of various methods or processes described herein. In some alternative implementations, two or more app servers **288** can be included and cooperate to perform such methods, or one or more other servers described herein can be configured to perform the disclosed methods.

**[0064]** The content batch servers **264** can handle requests internal to the pod. Some such requests can be long-running or not tied to a particular customer. For example, the content batch servers **264** can handle requests related to log mining, cleanup work, and maintenance tasks. The content search servers **268** can provide query and indexer functions. For example, the functions provided by the content search servers **268** can allow users to search through content stored in the on-demand database service environment. The file force servers **286** can manage requests for information stored in the Fileforce storage **298**. The Fileforce storage **298** can store information such as documents, images, and basic large objects (BLOBs). By managing requests for information using the file force servers **286**, the image footprint on the database can be reduced. The query servers **282** can be used to retrieve information from one or more file systems. For example, the query system **282** can receive requests for information from the app servers **288** and transmit information queries to the NFS **296** located outside the pod.

**[0065]** The pod **244** can share a database instance **290** configured as a multi-tenant environment in which different organizations share access to the same database. Additionally, services rendered by the pod **244** may call upon various hardware or software resources. In some implementations, the ACS servers **280** control access to data, hardware resources, or software resources. In some implementations, the batch servers **284** process batch jobs, which are used to run tasks at specified times. For example, the batch servers **284** can transmit instructions to other servers, such as the app servers **288**, to trigger the batch jobs.

**[0066]** In some implementations, the QFS **292** is an open source file system available from Sun Microsystems® of Santa Clara, Calif. The QFS can serve as a rapid-access file system for storing and accessing information available within the pod **244**. The QFS **292** can support some volume management capabilities, allowing many disks to be grouped together into a file system. File system metadata can be kept on a separate set of disks, which can be useful for streaming applications where long disk seeks cannot be tolerated. Thus, the QFS system can communicate with one or more content search servers **268** or indexers **294** to identify, retrieve, move, or update data stored in the network file systems **296** or other storage systems.

**[0067]** In some implementations, one or more query servers **282** communicate with the NFS **296** to retrieve or update information stored outside of the pod **244**. The NFS **296** can allow servers located in the pod **244** to access information to access files over a network in a manner similar to how local storage is accessed. In some implementations, queries from

the query servers **282** are transmitted to the NFS **296** via the load balancer **228**, which can distribute resource requests over various resources available in the on-demand database service environment. The NFS **296** also can communicate with the QFS **292** to update the information stored on the NFS **296** or to provide information to the QFS **292** for use by servers located within the pod **244**.

**[0068]** In some implementations, the pod includes one or more database instances **290**. The database instance **290** can transmit information to the QFS **292**. When information is transmitted to the QFS, it can be available for use by servers within the pod **244** without using an additional database call. In some implementations, database information is transmitted to the indexer **294**. Indexer **294** can provide an index of information available in the database **290** or QFS **292**. The index information can be provided to file force servers **286** or the QFS **292**.

II. Extracting and Processing Metrics from System Generated Events

**[0069]** Data taken from logged interactions of users with a multi-tenant information system may be raw event data. The raw event data may include data collected by logged interactions of any number of event types. One example of an event type is a login event, which occurs every time a user logs into an application of the multi-tenant information system. A login might produce raw event data indicating an Application Program Interface (API) type and version if an API is used for the login, a browser and client version if a User Interface (UI) is used for the login, a time of the login, a platform used for the login, an IP address from which the login took place, an identity of the user, whether the login was successful or not, or the like, or combinations thereof.

**[0070]** The amount of raw event data captured from just the login event type for a single user of a single tenant may be significant. For instance, a single user of a single tenant may login say 10 times in an hour, multiple times a week. This is one user of one tenant—there may be a multitude of other users of the same tenant plus a multitude of other user of another tenant all creating login events. And login events may be only one of more than one event type captured by the logging system. Manually querying the raw event data or a login event object, by a tenant (or for the tenant), may consume processing cycles of one or more machines involved with the query, possibly overwhelming computing resources or making those resources partially unavailable for other parallel tasks.

**[0071]** In an example, a group by function is applied at an interval to a login event object to create a platform event metrics object, i.e. to roll up all of the grouped events for the interval. The group by function may be applied to all raw event data available at the interval, or only to the raw event data collected since the previous occurrence of the interval. The platform event metrics object may be exposed to a tenant. The tenant may query the platform event metric object, which may consume less processing cycles than querying the login event object.

**[0072]** FIG. **3** is an operational flow diagram illustrating a process that may be performed by a processing system of a database system for extracting and processing metrics from system generated events.

**[0073]** In block **301**, the processing system categorizes data taken from logged interactions of users with a multi-tenant information system to provide a metric. The metric may be used to group events of the raw event data.

**[0074]** In an example, the metric may be based on number of logins, number of logins for a particular user, number of logins from a particular browser, number of logins in a specified timeframe, login duration, login IP address, login browser type, or the like, or combinations thereof. In an example, the processing system may be configured to allow a metric to be predefined in a declarative way with reference to components or attributes. In an example, components of a metric include a metric time, e.g. an interval, and a name-value pair.

**[0075]** FIG. **4**A shows an example of metric definition. The example eXtensible Markup Language (XML) code defines a metric "Number of Distinct Applications By User". The XML code defines a Metric Name **401** (also referred to as Metric Type), as well as other attributes of the metric, e.g. in this case function, function input field, event name, and aggregation field.

**[0076]** Referring again to FIG. **3**, in block **302** the processing system may associate a plurality of attributes with the metric. In one example, all of the attributes are associated with the metric in order to predefine the metric before, i.e. not responsively, to receiving a query from a tenant. The attributes may include timeframe, source object, database table, target field, or the like, or combinations thereof.

**[0077]** FIG. **4**B is an example of raw event data. FIG. **4**B shows an example of a Salesforce Object Query Language (SOQL) query **401** of the login event object selecting an application attribute equal to "CloudLock Compliance For Salesforce". The login event in FIG. **4**B represents raw log data where each reference to the CloudLock Compliance for Salesforce represents an individual login. This raw event data may be rolled up into a variety of aggregates or metrics as illustrated in FIG. **4**C.

**[0078]** Referring again to FIG. **3**, in block **303** the processing system periodically calculates the metric for a particular one of the tenants according to the associated attributes. In one example, the metric may be calculated every hour. In one example, a query by a tenant may be performed on a result of one of the periodic calculations, e.g. on the most recent periodic calculation results. In other words, the result may be pre-calculated without beforehand knowledge that a query will be received.

**[0079]** FIG. **4**C shows an example of an SOQL query of the platform event metric object. The query results **421** apply an attribute for a particular user to the result of pre-calculating the metric at the interval. Rows of the table of query results **421** corresponds to events, and columns correspond to fields. Each field may correspond to one of the attributes of the metric.

**[0080]** FIG. **4**C shows an example of performing a SOQL query over rolled-up aggregate information taken from the raw event data. That information may be rolled-up based on any number of attributes or fields from the raw event data. For instance, the ability to determine how many distinct applications were used in the past hour versus the total number of times a particular user logged in. Here, the query results **421** show that for the time interval, e.g. one hour, a metric value for the metric number of distinct applications by user is equal to one. The metric value for this time interval can be combined with metric values from other time intervals to determine a trend over time for the metric number of distinct applications by user. If a different metric was checked for the same raw data, e.g. the metric number of logins by user, a

metric value may have a different count, say four, despite the fact that a user may have only logged into one application, e.g. CloudLock for Compliance.

**[0081]** The tool illustrated in FIG. **4**C allows a tenant to select a metric or a combination of metrics, optionally select any number of attributes, and receive a result responsive to the selection (the result may be used for forensic analysis, for example). The result may be a list of relevant data from relevant logged events, or the result may be an image from a visualization tool. FIG. **4**D shows an example of the query results **421** mapped to a visualization tool.

**[0082]** In an example, the processing system may be configured to, responsive to receiving a query selecting an attribute value from a particular one of the tenants, applying the attribute value to a result of a pre-calculation of the metric for the particular one of the tenants. The pre-calculation may be performed prior to receiving the query without beforehand knowledge of the query. The processing system may provide, to the particular one of the tenants, a result of the application of the attribute value to the result of the pre-calculation of the metric for the particular one of the tenants.

**[0083]** In an example, the processing system may be configured to generate a profile for a user indicated by event data taken from logged user interactions with a multi-tenant information system. The profile may be based on a result of periodic calculations of the metric. The processing system may electronically store the generated profile in a data store. The processing system may compare information taken from a most recently logged interaction to the generated profile. The processing system may determine whether to transmit an alert response to a result of the comparison.

**[0084]** In an example, the categorization by the processing system may include selecting an event type, such as login event. The processing system may be configured to filter events of the event data according to the selected event type.

**[0085]** In an example, the processing system may be configured to identify an average number of logins for a duration using a result of one of the periodic calculations, e.g. according to a most recent one of the period calculations. The processing system may be configured to identify a threshold according to the average. The processing system may be configured to count a number of logins during a time period corresponding to the duration. The processing system may be configured to determine whether the count is greater than the determined threshold, and transmit an alert responsive to the determination.

**[0086]** In an example, the processing system may be configured to identify a first trend for a time of day logged in for a group of users of the tenant. The processing system may be configured to identify a second trend for a time of day logged in for a user of the group of users. The processing system may be configured to determine whether to transmit an alert using the first and second trends.

**[0087]** In an example, the processing system may be configured to detect a change in IP address between successive logins for a user. The processing system may be configured to determine whether a first duration between logins associated with the detected change is less than a second duration, and transmit an alert response to the determination.

**[0088]** In an example, the processing system may be configured to detect a change in browser type or browser version between successive logins for a user, determining whether to transmit an alert according to the detected change.

**[0089]** In an example, the processing system may be configured to provide a declarative ability to define system generated events that can be rolled-up into an aggregated metric, which may ensure that it is easy to both create and maintain metrics over time by declaring a series of attributes about the metric. In an example, the attributes include metric name, source object or database table, aggregate function, target field, aggregation clause, interval, timeframe, or the like, or combinations thereof. In an example, a series of system generated metrics may be defined declaratively based on common and repeatable characteristics.

**[0090]** Co-pending U.S. patent application Ser. No. 14/688,917 entitled EXTRACTION AND CAPTURE OF INFORMATION FROM CUSTOMIZABLE HEADER describes features for collecting data, which in an example may include unstructured data. In an example, any of the processes/operations described herein may be combined with any of the processes/operations described in the co-pending application. For instance, in an example, a processing system periodically calculates a metric based on structured data, e.g. event data, and unstructured data collected using features described in the co-pending application. A query may be received from a particular one of the tenants, the query applying a selected attribute value to a result of a pre-calculation of the metric based on the structured and unstructured data. The pre-calculation may be performed prior to receiving the query without beforehand knowledge of the query. The processing system may provide a result of the application of the attribute value to the particular tenant.

**[0091]** The system and apparatus described above may use dedicated processor systems, micro controllers, programmable logic devices, microprocessors, or any combination thereof, to perform some or all of the operations described herein. Some of the operations described above may be implemented in software and other operations may be implemented in hardware. Any of the operations, processes, and/or methods described herein may be performed by an apparatus, a device, and/or a system substantially similar to those as described herein and with reference to the illustrated figures.

**[0092]** The processing system may execute instructions or "code" stored in memory. The memory may store data as well. The processing system may include, but may not be limited to, an analog processor, a digital processor, a microprocessor, a multi-core processor, a processor array, a network processor, or the like. The processing system may be part of an integrated control system or system manager, or may be provided as a portable electronic device configured to interface with a networked system either locally or remotely via wireless transmission.

**[0093]** The processor memory may be integrated together with the processing system, for example RAM or FLASH memory disposed within an integrated circuit microprocessor or the like. In other examples, the memory may comprise an independent device, such as an external disk drive, a storage array, a portable FLASH key fob, or the like. The memory and processing system may be operatively coupled together, or in communication with each other, for example by an I/O port, a network connection, or the like, and the processing system may read a file stored on the memory. Associated memory may be "read only" by design (ROM) by virtue of permission settings, or not. Other examples of memory may include, but may not be limited to, WORM, EPROM, EEPROM, FLASH, or the like, which may be implemented in solid state semiconductor devices. Other memories may comprise moving

parts, such as a known rotating disk drive. All such memories may be "machine-readable" and may be readable by a processing system.

**[0094]** Operating instructions or commands may be implemented or embodied in tangible forms of stored computer software (also known as "computer program" or "code"). Programs, or code, may be stored in a digital memory and may be read by the processing system. "Computer-readable storage medium" (or alternatively, "machine-readable storage medium") may include all of the foregoing types of memory, as well as new technologies of the future, as long as the memory may be capable of storing digital information in the nature of a computer program or other data, at least temporarily, and as long as the stored information may be "read" by an appropriate processing system. The term "computer-readable" may not be limited to the historical usage of "computer" to imply a complete mainframe, mini-computer, desktop or even laptop computer. Rather, "computer-readable" may comprise storage medium that may be readable by a processor, a processing system, or any computing system. Such media may be any available media that may be locally and/or remotely accessible by a computer or a processor, and may include volatile and non-volatile media, and removable and non-removable media, or any combination thereof.

**[0095]** A program stored in a computer-readable storage medium may comprise a computer program product. For example, a storage medium may be used as a convenient means to store or transport a computer program. For the sake of convenience, the operations may be described as various interconnected or coupled functional blocks or diagrams. However, there may be cases where these functional blocks or diagrams may be equivalently aggregated into a single logic device, program or operation with unclear boundaries.

**[0096]** While one or more implementations have been described by way of example and in terms of the specific embodiments, it is to be understood that one or more implementations are not limited to the disclosed embodiments. To the contrary, it is intended to cover various modifications and similar arrangements as would be apparent to those skilled in the art. Therefore, the scope of the appended claims should be accorded the broadest interpretation so as to encompass all such modifications and similar arrangements.

**1**. A database system, comprising:

a processing system; and

a memory device coupled to the processing system, the memory device having instructions stored thereon that, in response to execution by the processing system, cause the processing system to perform operations comprising:

categorizing event data taken from logged interactions of users with a multi-tenant information system to provide a metric;

associating a plurality of attributes with the metric, wherein the plurality of attributes include interval and at least one of aggregation function, aggregation clause, or aggregation filter;

periodically calculating the metric for a particular one of the tenants according to the associated attributes; and

electronically storing the periodically calculated metrics.

**2**. The database system of claim **1**, wherein the operations further comprising:

responsive to receiving a query from the particular one of the tenants, the query selecting an attribute value, applying the selected attribute value to a result of a pre-calculation of the metric for the particular one of the tenants;

wherein the pre-calculation is performed prior to receiving the query without beforehand knowledge of the query; and

providing, to the particular one of the tenants, a result of the application of the attribute value.

**3**. The database system of claim **1**, wherein the operations further comprise:

generating a profile for one of the users based on the periodic calculations; and

electronically storing the generated profile.

**4**. The database system of claim **3**, wherein the operations further comprise:

comparing information taken from a most recent logged interaction to the generated profile; and

determining whether to transmit an alert responsive to a result of the comparison.

**5**. The database system of claim **1**, wherein the categorization comprises:

selecting an event type; and

filtering events of the event data according to the selected event type.

**6**. The database system of claim **5**, wherein the event type comprises login event.

**7**. The database system of claim **1**, wherein the metric corresponds to at least one event attribute including at least one of number of logins, login duration, login IP address, or login browser type.

**8**. The database system of claim **7**, wherein the operations further comprise:

identifying an average number of logins for a duration using a most recent one of the periodic calculations;

identifying a threshold according to the average;

counting a number of logins during a time period corresponding to the duration using a most recent one of the periodic calculations;

determining whether the count is greater than the determined threshold; and

transmitting an alert responsive to the determination.

**9**. The database system of claim **7**, wherein the operations further comprise:

identifying a first trend for a time of day logged in for a group of users of the tenant;

identifying a second trend for a time of day logged in for a user of the group of users; and

determining whether to transmit an alert using the first and second trends.

**10**. The database system of claim **7**, wherein the operations further comprise:

detecting a change in IP address between successive logins for a user;

determining whether a first duration between logins associated with the detected change is less than a second duration; and

transmitting an alert response to the determination.

**11**. The database system of claim **7**, wherein the operations further comprise:

detecting a change in browser type or browser version between successive logins for a user; and

determining whether to transmit an alert according to the detected change.

**12**. The database system of claim **1**, wherein the plurality of attributes include at least one of timeframe, source object, database table, or target field.

**13**. A method, comprising:

in a first phase:

categorizing event data taken from logged interactions of users with a multi-tenant information system to provide a metric;

associating a plurality of attributes with the metric, wherein the plurality of attributes include interval and at least one of aggregation function, aggregation clause, or aggregation filter;

periodically calculating, using a processing system of a database system, the metric for a particular one of the tenants according to the associated attributes;

electronically storing the periodically calculated metrics; and

in a second phase that is after the first phase, the second phase initiated by receipt of a query from the particular one of the tenants;

applying, using the processing system of the database system, a selected attribute value of the received query to one of the stored metrics; and

providing a result of the application of the selected attribute value of the received query to the one of the stored metrics.

**14**. The method of claim **13**, further comprising:

generating a profile for one of the users based on the periodic calculations; and

electronically storing the generated profile.

**15**. The method of claim **14**, further comprising:

comparing information taken from a most recent logged interaction to the generated profile; and

determining whether to transmit an alert responsive to a result of the comparison.

**16**. The method of claim **13**, wherein the categorization comprises:

selecting an event type; and

filtering events of the event data according to the selected event type.

**17**. The method of claim **16**, wherein the event type comprises login event.

**18**. The method of claim **13**, wherein the metric corresponds to at least one event attribute including at least one of number of logins, login duration, login IP address, or login browser type.

**19**. The method of claim **18**, further comprising:

identifying an average number of logins for a duration using a most recent one of the periodic calculations;

identifying a threshold according to the average;

counting a number of logins during a time period corresponding to the duration using a most recent one of the periodic calculations;

determining whether the count is greater than the determined threshold; and

transmitting an alert responsive to the determination.

**20**. The method of claim **18**, further comprising:

identifying a first trend for a time of day logged in for a group of users of the tenant;

identifying a second trend for a time of day logged in for a user of the group of users; and

determining whether to transmit an alert using the first and second trends.

* * * * *