



(19) **United States**

(12) **Patent Application Publication**

**Wuest et al.**

(10) **Pub. No.: US 2020/0358842 A1**

(43) **Pub. Date: Nov. 12, 2020**

(54) **CLOUD INTELLIGENCE DATA MODEL AND FRAMEWORK**

(71) Applicant: **Sonrai Security Inc.**, New York, NY (US)

(72) Inventors: **Ben Wuest**, Fredericton (CA); **William Bird**, Fredericton (CA); **Brad Peters**, Fredericton (CA); **Albert Lockett**, Fredericton (CA)

(73) Assignee: **Sonrai Security Inc.**, New York, NY (US)

(21) Appl. No.: **16/939,890**

(22) Filed: **Jul. 27, 2020**

**Related U.S. Application Data**

(63) Continuation of application No. 16/505,038, filed on Jul. 8, 2019, now Pat. No. 10,728,307.

(60) Provisional application No. 62/742,747, filed on Oct. 8, 2018.

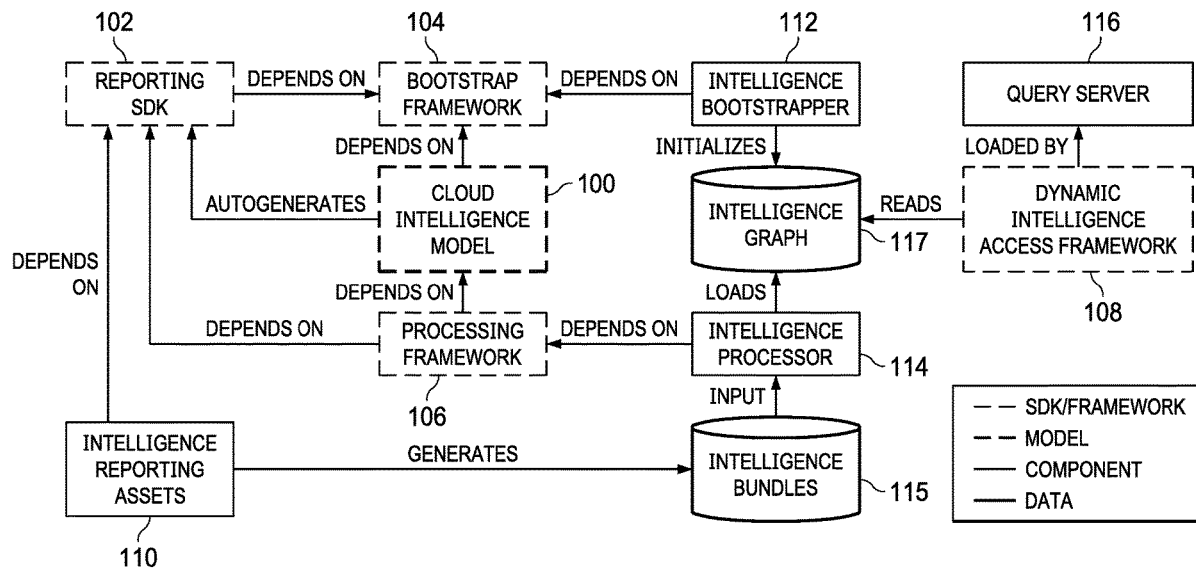
**Publication Classification**

(51) **Int. Cl.**  
**H04L 29/08** (2006.01)  
**G06F 16/2457** (2006.01)  
**G06F 11/34** (2006.01)

(52) **U.S. Cl.**  
CPC ..... **H04L 67/025** (2013.01); **H04L 67/1097** (2013.01); **G06F 16/2457** (2019.01); **G06F 11/34** (2013.01); **H04L 67/22** (2013.01); **H04L 67/306** (2013.01); **H04L 67/16** (2013.01); **H04L 67/36** (2013.01)

(57) **ABSTRACT**

A network-accessible service provides an enterprise with a view of all identity and data activity in the enterprise's cloud accounts. The service enables distinct cloud provider management models to be normalized with centralized analytics and views across large numbers of cloud accounts. The service enables an enterprise to model all activity and relationships across cloud vendors, accounts and third party stores. Display views of this information preferably can pivot on cloud provider, country, cloud accounts, application or data store. Using a domain-specific query language, the system enables rapid interrogation of a complete and centralized data model of all data and identity relationships. User reports may be generated showing all privileges and data to which a particular identity has access. Similarly, data reports shown all entities having access to an asset can be generated. Using the display views, a user can pivot all functions across teams, applications and data, geography, provider and compliance mandates, and the like.



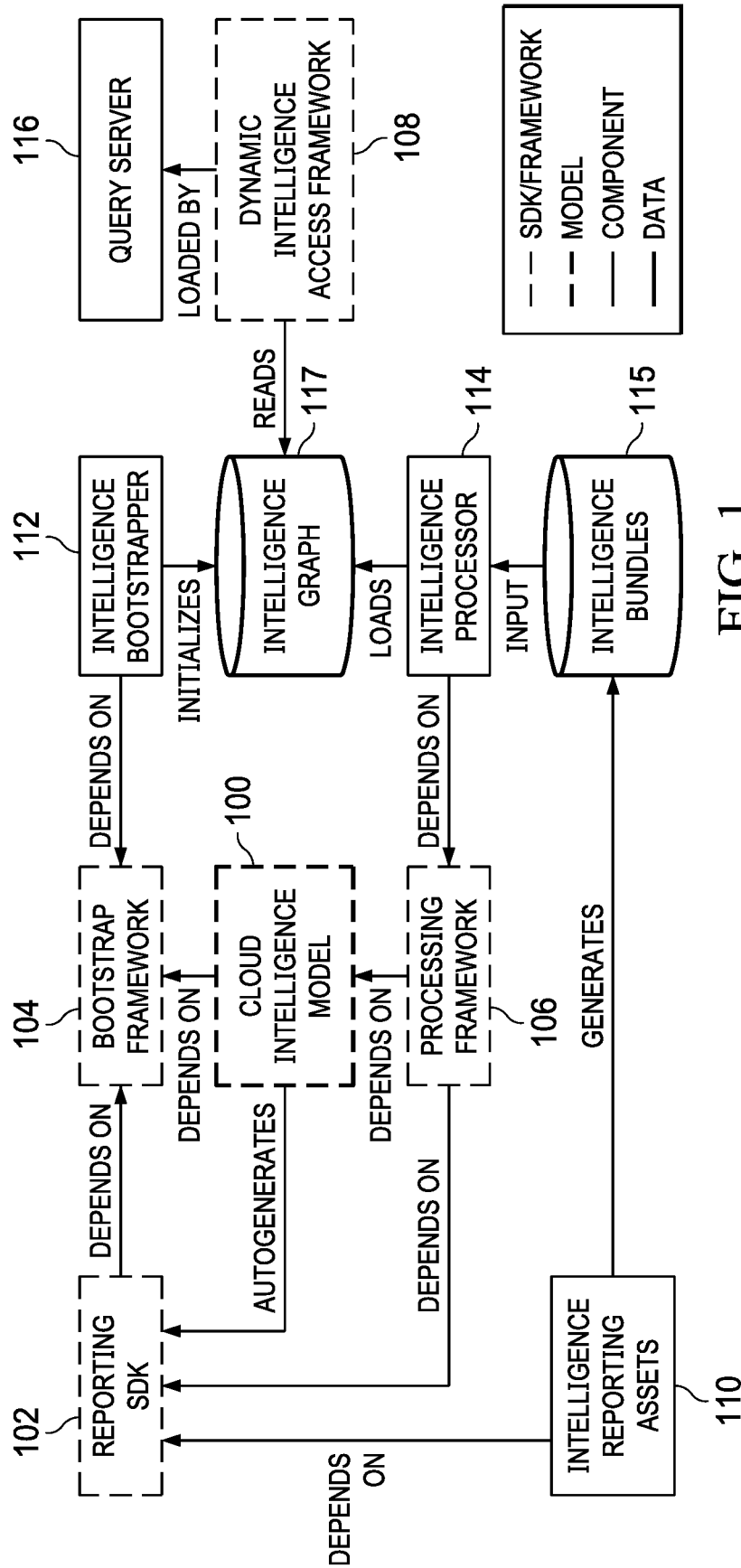


FIG. 1

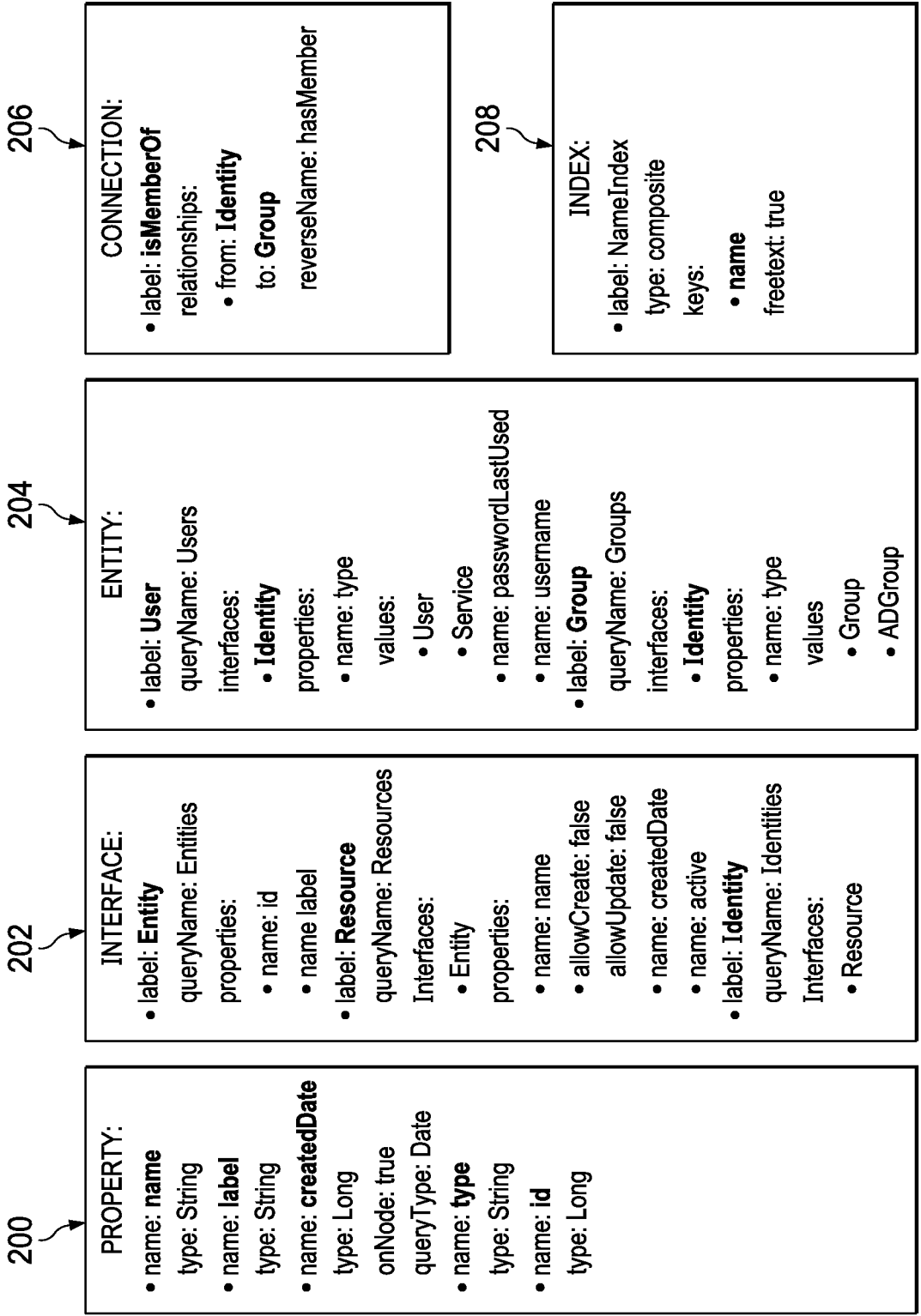


FIG. 2

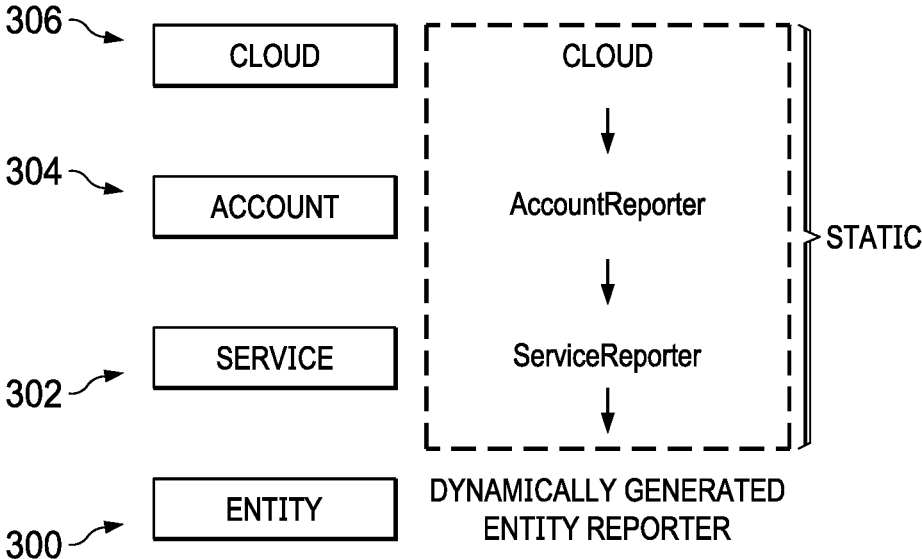


FIG. 3

```
public class UserReporter extends IdentityReporter<UserReporter> implements IReporter
{
    public enum Type {
        User,
        Service}
    public UserReporter(String uniqueName, ServiceReporter ServiceReporter)
        throws InvalidBuildArgumentException {
        super(EntityType.User, uniqueName, ServiceReporter);
    }
    public UserReporter withPasswordLastUsed(java.lang.Long value) {
        this.setProperty(PropertyType.passwordLastUsed, value);
        return this;
    }
    public UserReporter withUserName(java.lang.String value) {
        this.setProperty(PropertyType.userName, value);
        return this;
    }
    public UserReporter withType(Type value) {
        this.setProperty(PropertyType.type, value);
        return this;
    }
}
```

DYNAMICALLY GENERATED RESTRICTION ON TYPE

DYNAMICALLY GENERATED FROM PROPERTIES READ FROM INTELLIGENCE MODEL

FIG. 4

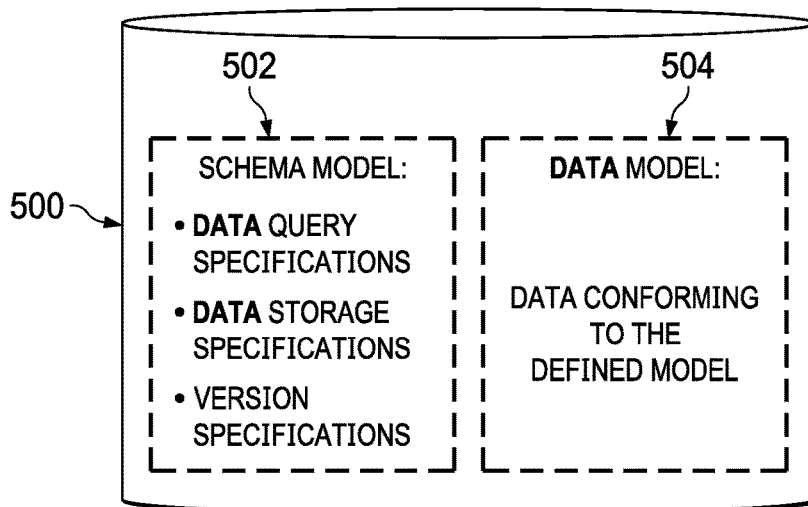


FIG. 5

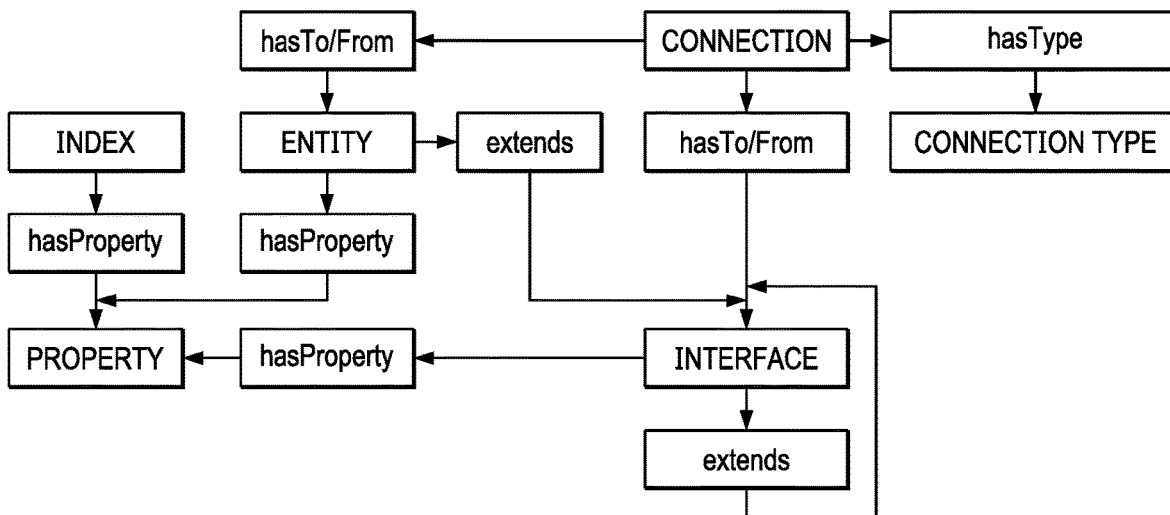


FIG. 6

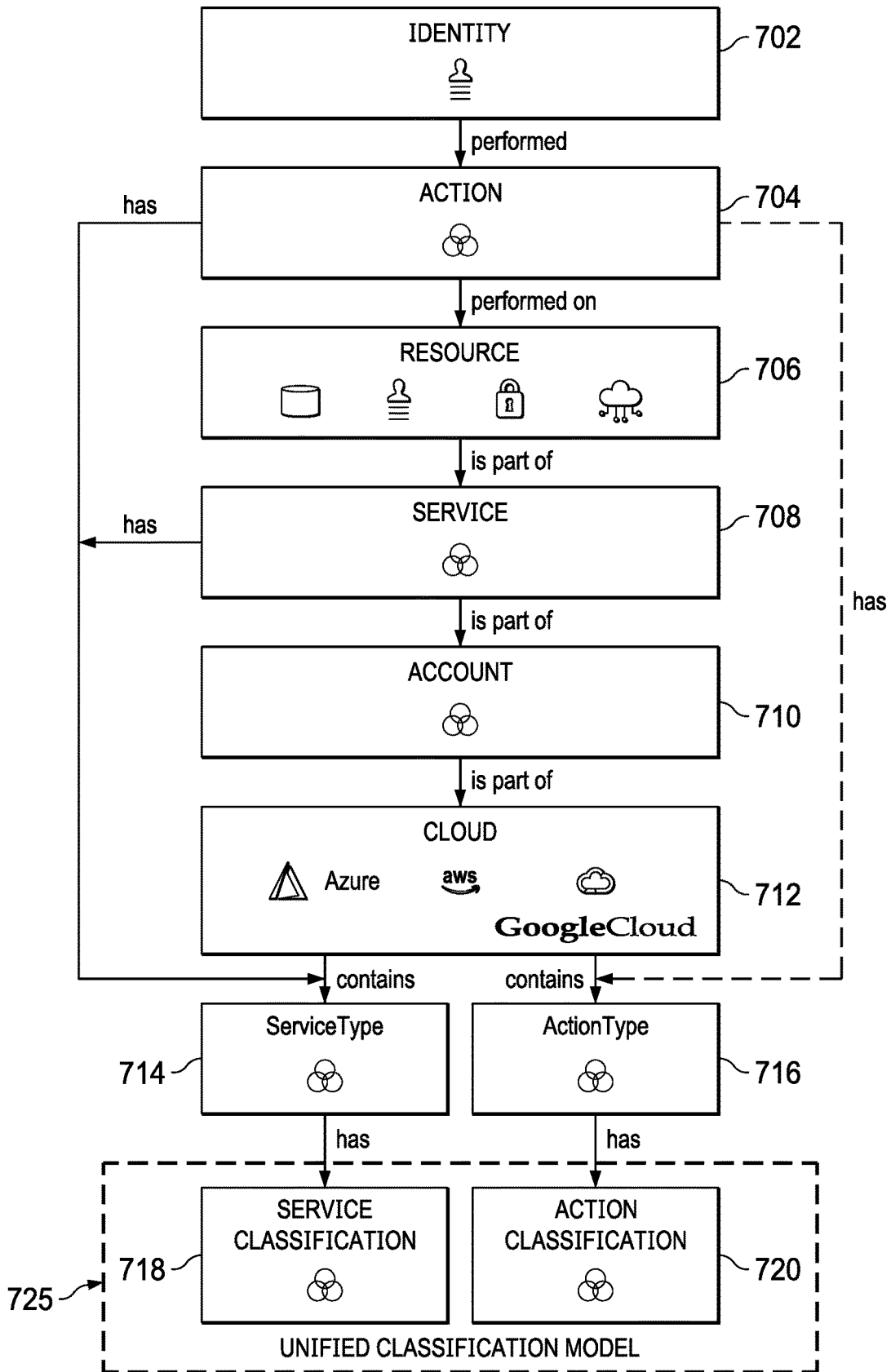


FIG. 7

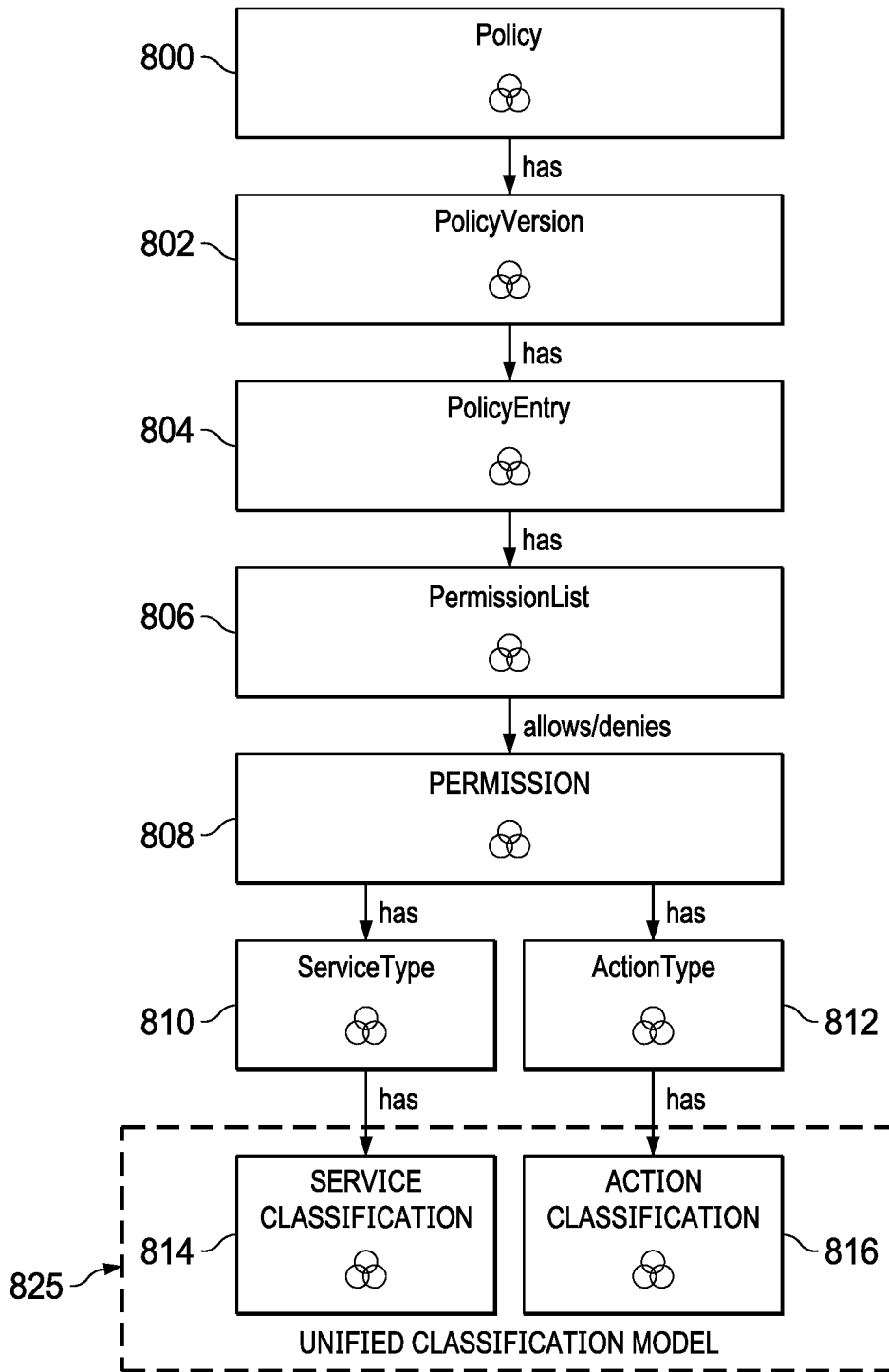


FIG. 8



CREATE	DELETE	MESSAGE	RESUME
ACTIVITY	STOP	ARCHIVE	ANALYTIC
ACTION	AUDIT	RESTORE	LOGIN
READ	START	AUTHENTICATION	FAILURE
UPDATE	CONFIGURE	ACCEPT	LOGOUT

FIG. 9

SecurityService	SQL	STREAMING	MountedDisk
SERVICE	MIGRATION	TRANSFER	VirtualHost
DATA	AI	BusinessProcess	VPS
ObjectStore	EMAIL	OperatingSystem	MediaProcessing
ANALYTIC	INDEX	DIRECTORY	MESSAGING
GEOLOCATION	HDFS	TimeSeries	WAREHOUSE
CONFIGURATION	FileSystem	LOCATION	AppDevelopment

FIG. 10

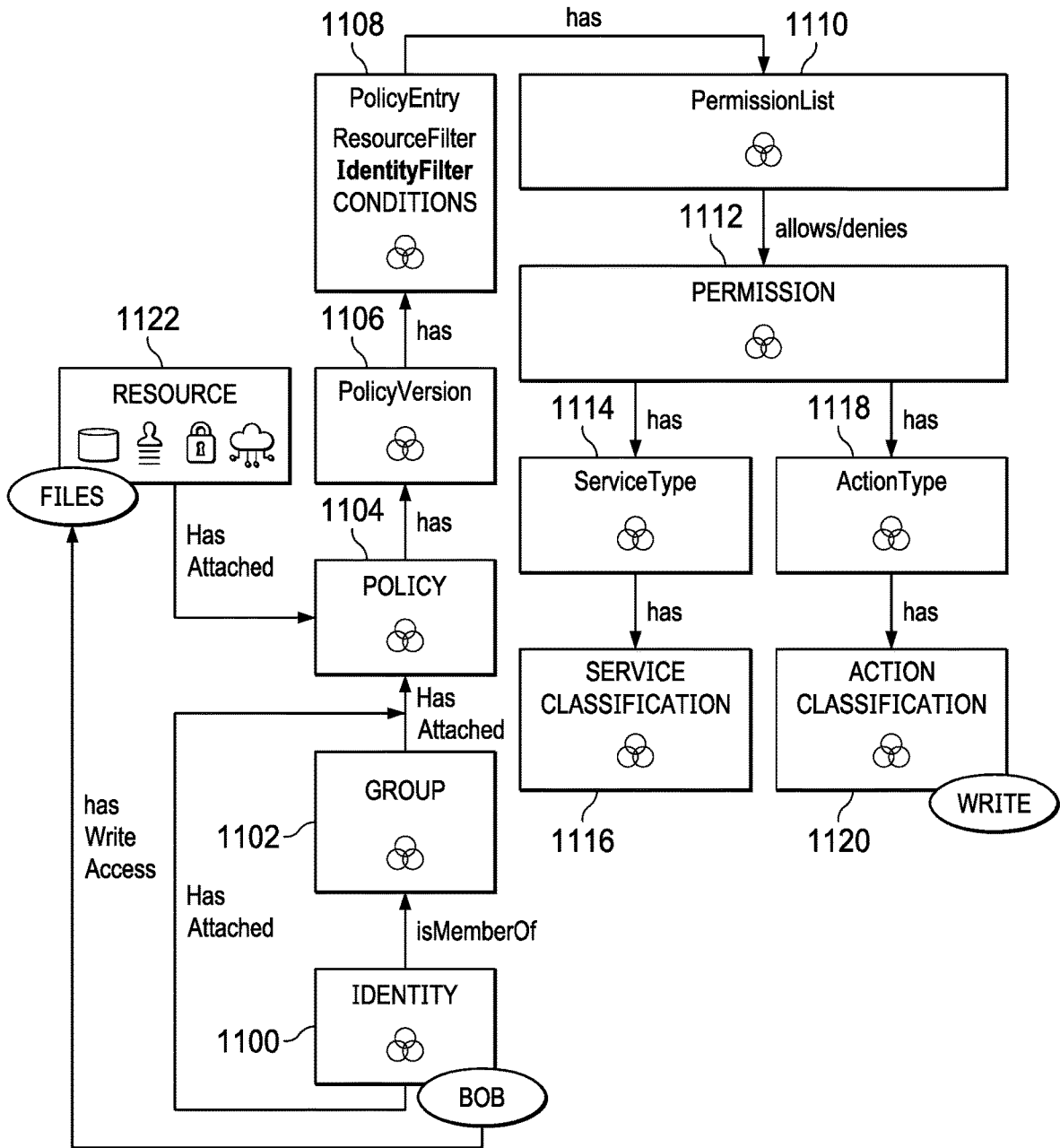


FIG. 11

```
{  
  "type": "Entity",  
  "name": "john.doe",  
  "author": "audit",  
  "properties": {  
    "name": "john.doe",  
    "active": true,  
    "type": "User"  
  },  
  "objectType": "User",  
  "createdDate": 1532722273000,  
  "cloud": "aws",  
  "service": "service",  
  "account": "456234566655"  
}
```

FIG. 12

## CLOUD INTELLIGENCE DATA MODEL AND FRAMEWORK

### BACKGROUND

#### Technical Field

[0001] This application relates generally to cloud compute infrastructures and, in particular, to techniques to model and manage data across multiple cloud deployments.

#### Brief Description of the Related Art

[0002] Cloud computing is an information technology delivery model by which shared resources, software and information are provided on-demand over a network (e.g., the publicly-routed Internet) to computers and other devices. This type of delivery model has significant advantages in that it reduces information technology costs and complexities, while at the same time improving workload optimization and service delivery. In a typical use case, an application is hosted from network-based resources and is accessible through a conventional browser or mobile application. Cloud compute resources typically are deployed and supported in data centers that run one or more network applications, typically using a virtualized architecture wherein applications run inside virtual servers, or virtual machines, which are mapped onto physical servers in the data center. The virtual machines typically run on top of a hypervisor, which allocates physical resources to the virtual machines.

[0003] Enterprises moving to cloud deployments typically use multiple cloud accounts across a number of providers (e.g., Amazon® Web Services, Microsoft® Azure and Google® Cloud Platform) in a number of ways. They migrate existing workloads to reduce costs, build new customer facing applications, and move employee backend processes to a continuous integrations/continuous delivery model. Large data science workloads also are transitioning to the cloud in all sizes of companies, and the processing of such workloads requires large clusters of compute and storage, sometimes for short time periods.

[0004] The rapid adoption of cloud technology has left Security, Compliance and Development Operations (DevOps) teams struggling to keep pace. Indeed, securing cloud data across a single cloud provider is hard enough, but securing data across a multi-cloud deployment is a significant challenge to even the most talented Security and DevOps teams. Making the problem even more of a challenge is that the agility of the cloud quickly leads to an explosion of cloud accounts, data stores, and data movement. Unfortunately, existing low-level tools lack a cohesive security model for identities and data movement, and none work across multiple cloud providers. Further, hackers have not overlooked the new attack vectors introduced by rapid cloud adoption. Every day, the media reports stories of significant cloud vulnerabilities and data breaches. Compounding this problem further, is that business often have to comply with not one, but potentially multiple government or industry regulations around data security. Moreover, rapid growth in the cloud has led to mind-numbing complexities and inefficiencies for DevOps and Security teams alike.

[0005] Accordingly, there is a need to provide tools and methods that enable enterprises that use multi-cloud deployments to obtain a comprehensive view of all identity and data activity across the enterprise's cloud accounts.

### BRIEF SUMMARY

[0006] This disclosure provides a cloud data control intelligence framework for modeling, reporting, storing and querying cloud resources and the connections among them. The framework preferably leverages a unified cloud intelligence data model. The framework is dynamic in that adjustments are made to the intelligence data model based on changes occurring in the underlying cloud resources. Further, key assets related to the reporting, storing and querying of resources dynamically update to reflect changes in the underlying intelligence model. In one embodiment, the framework provides a cloud risk control system that provides an enterprise the ability to continuously manage and interact with modern cloud environments, even as such environments themselves change and evolve.

[0007] The foregoing has outlined some of the more pertinent features of the disclosed subject matter. These features should be construed to be merely illustrative. Many other beneficial results can be attained by applying the disclosed subject matter in a different manner or by modifying the invention as will be described.

### BRIEF DESCRIPTION OF THE DRAWINGS

[0008] For a more complete understanding of the subject matter and the advantages thereof, reference is now made to the following descriptions taken in conjunction with the accompanying drawings, in which:

[0009] FIG. 1 depicts a framework architecture for a cloud risk intelligence platform according to this disclosure;

[0010] FIG. 2 is a representative data model implemented in one embodiment;

[0011] FIG. 3 depicts a representative base reporting framework;

[0012] FIG. 4 depicts a representative user entity reporter;

[0013] FIG. 5 depicts a representative data store;

[0014] FIG. 6 depicts a representative data model schema;

[0015] FIG. 7 depicts how a unified classification model ties to specific service and action types, and the corresponding actions from a given cloud provider;

[0016] FIG. 8 depicts how the unified classification model ties to a permission model in a cloud intelligence model;

[0017] FIG. 9 lists examples of normalized action types;

[0018] FIG. 10 lists examples of normalized service types;

[0019] FIG. 11 depicts a representative example of how normalized pathing analytics are used to distill information from the cloud intelligence model down to deliverable cloud intelligence; and

[0020] FIG. 12 depicts an example of a JSON code fragment generated by dynamic entity generation code in one embodiment.

### DETAILED DESCRIPTION

[0021] As described, cloud computing is a model of service delivery for enabling on-demand network access to a shared pool of configurable computing resources (e.g. networks, network bandwidth, servers, processing, memory, storage, applications, virtual machines, and services) that can be rapidly provisioned and released with minimal management effort or interaction with a provider of the service. Available services models that may be leveraged in whole or in part include: Software as a Service (SaaS) (the provider's applications running on cloud infrastructure); Platform as a service (PaaS) (the customer deploys applications that may

be created using provider tools onto the cloud infrastructure); Infrastructure as a Service (IaaS) (customer provisions its own processing, storage, networks and other computing resources and can deploy and run operating systems and applications). Typically, a cloud computing infrastructure may comprise co-located hardware and software resources, or resources that are physically, logically, virtually and/or geographically distinct. Communication networks used to communicate to and from the platform services may be packet-based, non-packet based, and secure or non-secure, or some combination thereof. Typically, the cloud computing environment has a set of high level functional components that include a front end identity manager, a business support services (BSS) function component, an operational support services (OSS) function component, and the compute cloud components themselves.

**[0022]** According to this disclosure, the services platform described below may itself be part of the cloud compute infrastructure, or it may operate as a standalone service that executes in association with third party cloud compute services, such as Amazon® AWS, Microsoft® Azure, IBM® SoftLayer®, and others.

**[0023]** Each of the functions described herein may be implemented in a hardware processor, as a set of one or more computer program instructions that are executed by the processor(s) and operative to provide the described function.

**[0024]** The server-side processing is implemented in whole or in part by one or more web servers, application servers, database services, and associated databases, data structures, and the like.

**[0025]** More generally, the techniques described herein are provided using a set of one or more computing-related entities (systems, machines, processes, programs, libraries, functions, or the like) that together facilitate or provide the described functionality described above. In a typical implementation, a representative machine on which the software executes comprises commodity hardware, an operating system, an application runtime environment, and a set of applications or processes and associated data, networking technologies, etc., that together provide the functionality of a given system or subsystem. As described, the functionality may be implemented in a standalone machine, or across a distributed set of machines.

**[0026]** A front-end of the below-described infrastructure (e.g., a customer console or portal) is also representative of a web site (e.g., a set of one or more pages formatted according to a markup language). Interaction with the portal may also take place in an automated manner, or programmatically, and the portal may interoperate with other identity management devices and systems.

**[0027]** As will be described below, and in a representative use case, an enterprise has relationships with multiple cloud providers, with each cloud provider typically implementing a network-accessible cloud computing infrastructure. This is sometimes referred to herein as a “multi-cloud” deployment. An enterprise multi-cloud deployment typically is one in which there are multiple cloud accounts, data stores, and data movement within and across the various cloud deployments provided by the multiple cloud providers. As will be described, and according to this disclosure, a Cloud Data Control (CDC) service provides an enterprise (typically, a service customer or “subscriber”) the ability to generate and use a complete risk model of all identity and data relationships, including activity and movement across cloud

accounts, cloud providers and third party data stores. Typically, the risk model is maintained by the CDC service provider and exposed to the enterprise customer via one or more display(s), typically web-accessible dashboards. Using the service, an enterprise subscriber obtains continuous visibility into a wide range of security concerns including multi-cloud security monitoring, data sovereignty, data exposure detection, audit tampering and identity governance. Data managed by the data model enables the service to provide the subscriber data risk dashboards that include, without limitation, (i) views by cloud accounts, geography, data and protection, user and identity, compliance, and public exposure; (ii) security alerts (e.g., over-privileged users with access to PII, failed privilege escalation attempts, audit functions disabled by user, unusual data movement, separation of duties violations, data movement to public network, shared credential violations, etc.), (iii) compliance dashboards indicating data sovereignty, data movement and identity relationships (e.g., GDPR, HIPAA, PCI dashboards, data sovereignty monitoring, data asset inventory, customized controls and compliance dashboards, monitoring PII data movement, etc.)

**[0028]** The CDC service typically is implemented by a service provider “as-a-service” on behalf of a participating enterprise customer. In a typical use case, the enterprise customer subscribes to the CDCaaS solution described herein. The enterprise includes its own on-premises infrastructure (networks, servers, endpoints, databases, etc.), internal IT teams (e.g., Security, Compliance, DevOps, etc.), as well as its relationships with one or more cloud providers that provide cloud-based infrastructure. Except to the extent the enterprise internal systems and the cloud provider infrastructure(s) interoperate with the CDC service (typically via data exchange), the subscriber and cloud provider infrastructures are external to the CDC service, which typically is operated and managed separately.

**[0029]** FIG. 1 depicts a preferred framework architecture for a cloud risk management platform that provides the CDC service according to an embodiment of this disclosure. The architecture comprises a number of software and data components or subsystems. The functional components need not be implemented as distinct elements, as various components may be combined with one another. Further, the components may execute independently of one another, and they may execute in different locations or using multiple computing entities. Typically, a computing entity that supports a component or data store is a computer or computing system that comprises one or more processors, storage and memory, network interfaces, etc. As depicted, typically the platform comprises cloud intelligence data model **100**, a reporting Software Development Kit (SDK) **102**, a set of one or more code frameworks, namely a graph bootstrap framework **104**, an intelligence processing framework **106**, and a dynamic intelligence framework **108**, as well as various system components comprising intelligence reporting assets **110**, an intelligence bootstrapper **112**, an intelligence processor **114**, and a query server **116**. As noted, these components typically are implemented in software executing on hardware processing elements. One or more components may be combined with one another or execute as separate processes. The particular relationships and dependencies between and among the various components of the framework are depicted in FIG. 1 for explanatory purposes and are not intended to be limiting.

**[0030]** The cloud intelligence model **100** is central to the framework, as it enables the CDC service to provide a subscriber a view of all identity and data activity in the enterprise's cloud accounts. Preferably, there is a cloud intelligence model developed and maintained for each subscriber to the service. Typically, this model is decoupled from the actual technical implementation in the reporting SDK **102**, the code frameworks, and the processing components, although each of which depend on this model closely. In a representative, but non-limiting embodiment, the model **100** is a cloud environment data model for a particular subscriber that is based on observed patterns across multiple cloud environments. As will be described, this solution provides a unified approach to modelling data, identity, infrastructure and protection. Preferably, the model **100** comprises an object model (e.g., all cloud entities and their corresponding properties, the allowed connections between and among cloud entities, and multi-level interfaces for the cloud entities), storage properties (e.g., index, types, etc.) for all or some of the above, and query properties of the object model.

**[0031]** Several of the components depicted in FIG. 1 typically are supported using computing systems and services technologies configured as follows. The cloud intelligence model **100** is pre-processed to generate the reporting SDK **102**, which it then embedded into the code comprising the intelligence reporting assets subsystem **110**; the intelligence reporting assets **110** (which include the embedded reporting SDK as auto-generated from the pre-processed cloud intelligence model) in turn reside within the cloud computing resources (systems, sub-systems, etc.) that are to be monitored by the cloud intelligence system of this disclosure. Thus, for example, the intelligence reporting assets **110** are configured to execute within or in association with a particular cloud computing system or resource (e.g., an Amazon® container running DynamoDB, etc.) and, as a result of that monitoring the intelligence reporting assets **110** generate the intelligence bundles **115**. An intelligence bundle **115** thus is generated for each particular cloud compute deployment that is being monitored. The intelligence bundle **115** includes the information about the cloud accounts, resources, etc. that the subscriber has provisioned in or is other using in each such cloud compute deployment. In a representative implementation, the service provider deploys a Docker container or the like (that maintains the intelligence reporting assets **110** and the embedded reporting SDK **102**) in the third party cloud computing deployment for support of the data collection. Typically, data collection with respect to a particular cloud deployment (and for a particular customer) occurs during an initial discovery phase, and then runs continuously thereafter (e.g., nightly). As noted, the intelligence reporting assets subsystem **110** collects that data from the external cloud deployment(s) and generates the one or more intelligence bundles **115**. As noted, a bundle is associated with an enterprise subscriber and encapsulates the subscriber's data (e.g., identity and data activity, etc.) retrieved from each cloud deployment being used by the subscriber. The intelligence processor **114** receives an intelligence bundle and processes it under the control of the processing framework **106**, thereby generating an intelligence graph **117** (subscriber-specific). Because the processing framework depends on the cloud intelligence model **100**, and because the intelligence processor **114** depends on the processing framework **106**, the model **100** is embedded/

instantiated in the intelligence graph **117**. An initial version of the intelligence graph is initialized by the intelligence bootstrapper **112**, and the intelligence graph is updated (by the intelligence processor **114**) as the intelligence bundle **115** is updated (by changes in the local cloud compute deployment, as reported by the intelligence reporting asserts **110**). The intelligence graph **117** is configured to be queried by a query server **116**, which executes on an application server subsystem and, together with a web server, preferably exposes a query-able display interface (e.g., as a set of web pages) to authenticated and authorized enterprise users. Typically, an enterprise user interacts with the query server sub-system using a client browser or mobile app.

**[0032]** FIG. 1 depicts the various dependencies and interactions among the components and data elements that are described above. As shown, the bootstrap framework **104**, the processing framework **106** and the intelligence processor **114** depend on the cloud intelligence data model **100**. As described, the intelligence processor **114** also receives as input the intelligence bundle(s) **115** that are generated by the cloud-deployed intelligence reporting assets subsystem **110** (which as noted also includes the embedded reporting SDK itself generated from the model). The processing framework **106** processes the data model (in the manner described below in more detail) and provides results of such processing to the intelligence processor subsystem **114**, which in turn generates the intelligence graph **117** that is exposed to a dynamic intelligence access framework **108** loaded by the query server **116** to thereby respond to data queries. As noted above, typically each cloud intelligence data model (one per subscriber) has an associated intelligence graph **117**. The intelligence graph **117** is a knowledge-based structure of edges and nodes that is generated by an intelligence bootstrapper component **112**, which in turn depends on a bootstrap framework **104** that receives the data model as input. Upon initialization, the data model (and associated data) is stored inside the intelligence graph.

**[0033]** Generalizing, these subsystems and data structures interact in the manner depicted in FIG. 1 to collect, organize, manage and display the data to the subscriber via the above-described query-able dashboards. Further details regarding these components are provided below.

**[0034]** As noted above, and to provide the CDC service to a participating subscriber, the system generates and manages a cloud intelligence data model for each subscriber. As noted above, the data model is stored inside the intelligence graph upon startup. A representative data model schema that supports this data model is now described.

**[0035]** In particular, FIG. 2 depicts a sample from the model **100** with respect to objects referred to herein as Users and Groups. As depicted, the model preferably defines properties **200**, interfaces **202**, entities **204**, connections **206** and indexes **208**. Within each of these definitions are pieces of information that are collected, managed and leveraged by the various parts of the framework. The following provides additional examples of these representative portions of the data model.

#### Property Definition

**[0036]** The following is a representative scheme for the property definition (AttributeDescription): name|Name of the property; type|the storage type for the property; onNode|directs storage to either put the property on the entity or

not; queryType|the type to use in the query interface (e.g., in FIG. 2, the createdDate is stored as Long but queried as Date).

#### Interface Definition

**[0037]** Interface definitions are primarily used for reporting and querying data. They need not be stored in the data store. The concept of interfaces allows the hierarchy of a query to change the entities that are stored in the actual data store. Preferably, there are layers of inheritance that allow the framework to look for all entities that conform to a particular interface. For example, the sample in FIG. 2 directs the query server to return both User and Groups when the query asks for all contained Identities.

**[0038]** Preferably, Interfaces can also extend Interfaces. This is shown in FIG. 2, where the chain of User—Identity—Resource—Entity is represented from an inheritance point of view. The following is a representative scheme for the Interface definition (Attribute|Description): label|the name/label of the interface; queryName|the name used by the query/access framework; interfaces|any interfaces that the given interface extends; and properties|any properties that exist on the given interfaces, and any interface or entity that extends this interface will inherit these properties.

#### Entity Definition

**[0039]** Entity definitions define entities that are used in reporting, querying and storage. They extend Interfaces but preferably do not extend each other. The following is a representative scheme for the Entity definition (Attribute|Description): label|the name/label of the interface; queryName|the name used by the query/access framework; interfaces|any interfaces that the given interface extends; and properties|any properties that exist on the given interfaces, and any interface or entity that extends this interface will inherit these properties.

**[0040]** In addendum to the properties defined above, properties preferably enforce types on the reporting and query layers. For instance, in the User entity defined in FIG. 2, the type is restricted to User or Service. This ensures integrity in the reporting and query sections of the framework.

#### Connection Definition

**[0041]** The connection definitions allow the query framework to expose queries to the user, and for the storage framework to appropriately store relationships in the data model. Each connection preferably has a label and can contain multiple relationships (meaning multiple entities can use the same connection identifier). In the above example, which is merely representative, a relationship between Identity and Group is established, thereby defining that anything that extends Identity can have a “isMemberOf” connection with Group Entity. The following is a representative scheme for the Connection definition (Attribute | Description): label | the name of the connection relationships All the relationships that use this connect. Each relationship entry contains a: fromNode, a toNode, and a reverseName.

#### Index Definition

**[0042]** Index definitions are primarily used by the bootstrap and storage layers. They define what properties need to be indexed and how to support the use cases placed on the intelligence framework. The following is a representative

scheme for the Index definition (Attribute|Description): label|the name of the index; type|the index type; keys|properties included by the framework (must be referenced in the property definition); and freeText|a flag identifying if the index is free text or not.

**[0043]** As referenced above, the reporting SDK depicted in FIG. 1 provides a means for reporting data that can be consistently consumed by the framework. Preferably, the reporting SDK is dynamically-generated from the intelligence model such that, as updates are made to the model, new reporting assets are easily produced using a newly-generated SDK. As implemented, typically the reporting SDK is embedded within the intelligence reporting assets that are deployed in the cloud compute environment to be monitored.

**[0044]** Referring to FIG. 3, at the base of the SDK reporting framework is the notion that any entity **300** in the data model preferably also exists as part of a service **302**, an account **304** and a cloud **306** in the framework. This means that an entity (see, FIG. 2, **204**) typically is reported with respect to (in association with) a service, account and/or cloud. As depicted, a dynamically-generated entity reporter component exposes entity-specific information, whereas the respective service, account and cloud-specific is exposed by the respective ServiceReporter, AccountReporter and CloudReporter components, which components provide static information. Referring now to FIG. 4, a portion of the dynamically-generated entity reporter component is shown. This portion (which is part of the reporting SDK **102**) is generated and embedded in the deployed intelligence reporting assets **110**. This particular portion is the SDK code used to collect and report information regarding user identity. The code snippets that are dynamically-generated are shown. Thus, e.g., there are several code snippets (“PasswordLastUsed,” “UserName,” “Type”) that (in this example) depend on the “properties” that are read from the cloud intelligence model. Taken together, FIGS. 3 and 4 depict how the intelligence model is used to enable dynamic generation of the reporting code for a particular cloud data source that is to be monitored.

**[0045]** In operation, preferably dynamic entity generation code reads all the Property, Interface, Entity and Connection definitions from the intelligence model to produce a set of reporters that produce intelligence, preferably in a JSON standard format such as depicted below in a representative snippet as depicted in FIG. 12.

**[0046]** By reading the model, a User Entity Reporter is produced through templates that are written in a given computer language. As previously described, FIG. 4 is an example reporter in the Java Programming Language.

**[0047]** Preferably, the dynamically-generated assets in the reporting SDK implement a reporter interface, which interprets the data produced by any reporter and produces a standard format. This allows a consistent way for the reporting SDK to report data.

**[0048]** Preferably, the code frameworks that are part of the framework provide capabilities built upon the Reporting SDK and Object Model to bootstrap an intelligence graph **117** (see FIG. 1) according to the specifications of the cloud intelligence model, process intelligence reported to store that data in the intelligence graph, and enable query on the data in the intelligence graph. As used herein, and generalizing, a code framework refers to software providing some basic functionality that is selectively changed by additional

user-written code, thus providing application-specific software. A code framework of this type provides a mechanism to build and deploy applications. A code framework may include support programs, compilers, code libraries, tool sets, and application programming interfaces (APIs) that bring together all different components to enable development of a system.

**[0049]** The code frameworks, which preferably dynamically adjust according to the cloud intelligence model, provide a support mechanism underlying the other cloud risk control system processing components, as is now described. Preferably, and as depicted in FIG. 1, these code frameworks comprise the data bootstrap framework **104**, the data processing framework **106**, and the dynamic data access framework **108**, all of which are now further described.

**[0050]** The bootstrap framework **104** is responsible for building a data model from the model definition that contains sufficient information for the dynamic intelligence access framework **108** to build a schema for querying the contained data and a model to allow processing components to store the data. A data store **500** bootstrapped by the framework **104** preferably contains two sections (one for the model schema **502**, and the other for the actual subscriber-specific data **504** comprising the data model), as depicted schematically in FIG. 5. The model schema **502** typically comprises data query specification, data storage specifications, version specifications, and so forth.

**[0051]** The bootstrap framework **104** preferably also provides several capabilities used by the intelligence bootstrapper component **112** to prepare a data store, namely: initialization of an empty data store with the model schema, translation of the intelligence model into the model schema, and initialization of the data schema based on the information in the model schema.

**[0052]** The model schema generated (see, e.g., FIG. 6) preferably is a static model used by the framework. This is the preferred schema for which all the information provided by the intelligence model is entered, and the various elements (Property, Interface, Entity, Connection, Index) are as specified in FIG. 2. As further described, the query framework (namely, the query server and the dynamic intelligence access framework) reads this model schema to dynamically generate a query schema used to query the data contained in the data section of the model.

**[0053]** Preferably, the data processing framework **106** is built (depends) upon the reporting SDK, which is automatically generated from the cloud intelligence model **100**. The processing framework **106** reads intelligence and stores it in the framework data store. The processing framework **106** validates incoming intelligence against the framework data store to which connects, e.g., by examining its contained schema model.

**[0054]** The dynamic intelligence access framework **108** ties these other frameworks together. Because the framework data store contains the schema model that tabulates all the information from the model (including entities, connections, interfaces and connections), the dynamic data access framework **108** builds up a domain-specific query language based on this information, and this functionality allows the system to reliably and efficiently query the system as the model and/or data changes. The dynamic data access framework also provides the generic capability to drive system APIs, User Interfaces and System components.

**[0055]** The system components (which preferably built upon the code frameworks) provide a cohesive cloud intelligence control framework that reports intelligence from one or more cloud environments, processes and store the intelligences, and enables querying and analysis of the data. Because of the dynamic nature of the entire framework, updated components dynamically adjust to changes from the cloud intelligence model.

**[0056]** The framework provides significant advantages. It provides a unified cloud intelligence model and query framework. A cloud risk control system that leverages the unified cloud model can assess resources across multiple clouds and accounts in a unified fashion. Asking questions like “show me over-permissioned users” are consistent regardless of where the intelligence comes from. The framework is dynamic and responds to model updates. The techniques herein provide for updating code (e.g., SDK reporter code paths) and internal data and processing components as the cloud landscape evolves. The framework allows a cloud risk control system to continually evolve as new capabilities are introduced by cloud providers, and as new workloads are introduced to cloud infrastructures.

**[0057]** Generalizing, the cloud intelligence model described above (sometimes also referred to as a Cloud Risk Control (CRC) data model) unifies the view of Identity, Data, Protection and Infrastructure across multiple clouds and accounts. There are two components to this model that provide context to Cloud Risk Control (CRC). They are a unified classification model of cloud actions and services, and normalized pathing analytics. Further details of these aspects are now described.

**[0058]** The unified classification model allows for interrogation and analytics-related Cloud Risk Control to operate across cloud policies and controls that are decoupled from the actual individual cloud action and service types. The language of the unified classification model can be expressed in various ways, such as depicted in FIGS. 7-10. For example, FIG. 7 depicts how a unified classification model ties to specific service and action types, and the corresponding actions from a given cloud provider. In FIG. 7, which is an example, Identity **702** is performed on an Action **704**, which in turn is performed with respect to a Resource **706** that is part of a Service **708**. Service **708** is part of an Account **710**, which in turn is an account that is provisioned by a cloud provider **712**. In this example, Action **704** has a ServiceType **714** and ActionType **716**. The ServiceType **714** in this example has a Service Classification **718**, and the ActionType **716** has an Action Classification **720**. The classifications comprise a part of a Unified Classification Model that is common to the system. In another example, FIG. 8 depicts how the unified classification model ties to a permission model in a cloud intelligence model. In this example, Policy **800** has a PolicyVersion **802**, which in turn has a PolicyEntry **804**. The PolicyEntry **804** has a PermissionList **806** that allows or denies individual Permissions, which of which is represented as Permission **808**. The Permission **808** has an associated ServiceType **810** and ActionType **812**, with these types having associated Service- and Action-Classifications **814** and **816**. Once again, these classifications **814** and **816** comprise part of the Unified Classification Model. FIG. 9 lists examples of normalized action types (e.g., ActionType **716** in FIG. 7, or ActionType



**812** in FIG. 8). FIG. 10 lists examples of normalized service types (e.g., ServiceType **714**, in FIG. 7, or ServiceType **810** in FIG. 8).

**[0059]** Normalized pathing analytics distill the information from the cloud intelligence model (as instantiated in the intelligence graph) down to deliverable cloud intelligence. FIG. 11 depicts an example of how a relation of an Identity (Bob) is determined to have WRITE access to a Resource (Files). In this example, Identity **1100** is a member of a Group **1102**. The Identity **1100** has an attached Policy **1104**, which in this example is also attached to the Group **1102**. The Policy **1104** has a PolicyVersion **1106** that has a PolicyEntry **1108**. The PolicyEntry **1108** has an associated PermissionList **1110** comprising permissions. Permission **1112** has a ServiceType **1114** having an associated Service Classification **1116**, as well as an ActionType **1118** having an associated Action Classification **1118**. The Policy **1104** manages one or more resources, such as Resource **1120**. Within the collected data there are many paths (several are displayed here) which can identify that “BOB” has “WRITE” access to the Resource “FILES.” Thus, and using the above-described schema, e.g., one path is as follows: Bob→Policy→PolicyVersion→Policy Entry (FILES)→PermissionList (allows)→Permission→ActionType→WRITE. Another path is BOB→Group→Policy→PolicyVersion→Policy Entry (FILES)→PermissionList (allows)→Permission→ActionType→WRITE. Still another path is: FILES Policy→PolicyVersion→Policy Entry (BOB)→PermissionList (allows)→Permission→ActionType→WRITE. As is self-evident from this simple example scenario, this type of data management can get very complicated in multi-cloud, multi-account environments.

**[0060]** In accordance with the techniques herein, the pathing analytics distill this information down to enable easy interrogation using the query server. In a preferred embodiment, the intelligence graph for a particular enterprise customer of the service is supported in a graph database. A graph database uses graph structures for semantic queries with nodes, edges and properties to represent and store data. A graph (one or more edges and/or relationships) relates data items in the store to a collection of nodes and edges, wherein the edges represent the relationship between and among the nodes. The relationships enable data in the store to be linked together directly and, depending on the query, retrieved in one or just a few operations. Relationships also can be visualized using graph databases, making them useful for heavily inter-connected data.

**[0061]** As previously noted, the enterprise-specific data model and associated data is stored in the knowledge graph initially (at startup) and then configured to be queried. As the underlying information (in the various cloud environments changes), the enterprise’s intelligence graph is updated, preferably continuously, e.g., via the intelligence reporting assets subsystem. At query time, the enterprise user (e.g., an authorized person) executes a query from the query server **116**. The query server loads the dynamic intelligence access framework, which in turn reads the intelligence graph for the enterprise, with the graph being configured according to the cloud model. Because the access framework contains the schema model and thus the all of the information in the model, the dynamic access framework can configure and use a domain specific query language (e.g., Cypher) based on this information. A declarative graph query language of this type allows for expressive and efficient querying and updat-

ing of the graph. Use of declarative graph query language users to focus on structuring queries that are domain-specific (relevant) without having to managed underlying database access requirements.

**[0062]** The techniques herein provide significant advantages. A representative embodiment of the framework is a cloud data control service that finds and continuously monitors an enterprise’s cloud-supported resources and all entities with access to them. This is enabled across cloud providers, cloud account and third party data stores. By providing this comprehensive view, the service enables users (e.g. DevOps and security personnel) to achieve improved data security and reduced risk (including public data exposure risks, configuration and privilege risks, crown jewel monitoring, anomalous data movements, anomalous user/developer activity, etc.), ensure compliance (e.g., GDPR compliance, data sovereignty monitoring, HIPAA, PCI and other compliance reporting, data asset inventory discovery and monitoring), and increase DevOps efficiency.

**[0063]** The approach provides an enterprise with a total view of all identity and data activity in its cloud accounts. The system enables cloud provider management models to be normalized with centralized analytics and views across large numbers of cloud accounts (e.g., AWS/GCP accounts, Azure subscriptions/resource groups, etc.) As previously described, a cloud data control service implemented using the framework enables an enterprise customer to model all activity and relationships across cloud vendors, accounts and third party stores. Display views of this information preferably can pivot on cloud provider, country, cloud accounts, application or data store. Using a Cloud Query Language (CQL), the system enables rapid interrogation of the complete and centralized data model of all data and identity relationships. User reports may be generated showing all privileges and data to which a particular identity has access. Similarly, data reports shown all entities having access to an asset (and the access history) can be generated. Using the display views, user can pivot all functions across teams, applications and data, geography, provider and compliance mandates, and the like.

**[0064]** Using the approach herein, a cloud data control (CDC) service provides a complete risk model of all identity and data relationships, including activity and movement across cloud accounts, cloud providers and third party data stores. Data risk dashboards include, without limitation, (i) views by cloud accounts, geography, data and protection, user and identity, compliance, and public exposure; (ii) security alerts (e.g., over-privileged users with access to PII, failed privilege escalation attempts, audit functions disabled by user, unusual data movement, separation of duties violations, data movement to public network, shared credential violations, etc.), (iii) compliance dashboards indicating data sovereignty, data movement and identity relationships (e.g., GDPR, HIPAA, PCI dashboards, data sovereignty monitoring, data asset inventory, customized controls and compliance dashboards, monitoring PII data movement, etc.)

**[0065]** While the above description sets forth a particular order of operations performed by certain embodiments, it should be understood that such order is exemplary, as alternative embodiments may perform the operations in a different order, combine certain operations, overlap certain operations, or the like. References in the specification to a given embodiment indicate that the embodiment described may include a particular feature, structure, or characteristic,

but every embodiment may not necessarily include the particular feature, structure, or characteristic.

**[0066]** While the disclosed subject matter has been described in the context of a method or process, the subject disclosure also relates to apparatus for performing the operations herein. This apparatus may be specially constructed for the required purposes, or it may comprise a general-purpose computing entity selectively activated or reconfigured by a stored computer program stored. Such a computer program may be stored in a computer readable storage medium, such as, but is not limited to, any type of disk including an optical disk, a CD-ROM, and a magnetic-optical disk, flash memory, a read-only memory (ROM), a random access memory (RAM), a magnetic or optical card, or any type of non-transitory media suitable for storing electronic instructions.

**[0067]** While given components of the system have been described separately, one of ordinary skill will appreciate that some of the functions may be combined or shared in given instructions, program sequences, code portions, and the like.

**[0068]** A given implementation of the computing platform is software that executes on a hardware platform running an operating system such as Linux. A machine implementing the techniques herein comprises a hardware processor, and non-transitory computer memory holding computer program instructions that are executed by the processor to perform the above-described methods.

**[0069]** The functionality may be implemented with other application layer protocols besides HTTP/HTTPS, or any other protocol having similar operating characteristics.

**[0070]** There is no limitation on the type of computing entity that may implement the client-side or server-side of the connection. Any computing entity (system, machine, device, program, process, utility, or the like) may act as the client or the server.

**[0071]** While given components of the system have been described separately, one of ordinary skill will appreciate that some of the functions may be combined or shared in given instructions, program sequences, code portions, and the like. Any application or functionality described herein may be implemented as native code, by providing hooks into another application, by facilitating use of the mechanism as a plug-in, by linking to the mechanism, and the like.

**[0072]** The platform functionality may be co-located or various parts/components may be separately and run as distinct functions, perhaps in one or more locations (over a distributed network).

**[0073]** The techniques herein provide for improvements to another technology or technical field, namely, data analytics tooling, applications and systems, as well as improvements to cloud computing infrastructures that support such functions and technologies.

**[0074]** A cloud risk control system as described and depicted may be implemented within a cloud compute infrastructure, or as an adjunct to one or more third party cloud compute infrastructures. The cloud risk control system may be implemented in whole or in part by a service provider on behalf of entities (e.g., enterprise customers) that use third party cloud computing resources. A typical implementation provides for cloud risk control-as-a-service in the manner described herein. Portions of the cloud risk control system may execute in an on-premises manner within or in association with an enterprise. The cloud risk control system preferably comprises a web-accessible portal (e.g., an extranet application) that is accessible via a browser or mobile app via HTTP/HTTPS, or other protocol.

**[0075]** Communications between devices and the cloud risk control system are preferably authenticated and secure (e.g., over SSL/TLS).

What is claimed is as follows:

1. A computing system accessible over a network, comprising:
  - a processor;
  - computer memory holding computer program instructions executed by the processor to provide a service to one or more subscribers, the computer program instructions configured, with respect to each of the one or more subscribers, to:
    - receive a subscriber-specific data set from each of one or more cloud computing infrastructures, wherein each of the cloud computing infrastructures has cloud-specific service types or action types, the cloud-specific service types or action types conforming to a unified classification model common to all of the cloud computing infrastructures;
    - initialize a data model conforming to a schema model;
    - store the data model and the data set within a knowledge graph;
    - as changes in the one or more cloud computing infrastructures occur, dynamically update the data model and knowledge graph responsive to the changes; and
    - responsive to one or more queries, selectively retrieve information from the knowledge graph.

\* \* \* \* \*