

Methods and systems for facilitating data store migration

Technical Field

[1] Embodiments generally relate to methods, systems, and computer-readable media for facilitating data store migration, and in some embodiments, for facilitating data store schema changes to accommodate read and write applications.

Background

[2] Any discussion of documents, acts, materials, devices, articles or the like which has been included in the present specification is not to be taken as an admission that any or all of these matters form part of the prior art base or were common general knowledge in the field relevant to the present disclosure as it existed before the priority date of each claim of this application.

Summary

[3] Some embodiments relate to a method comprising: providing a first data store interface package associated with a first data store, the first data store package comprising a first schema compatible with the structural representation of data in the first data store; importing, by a first application, the first data store interface package; configuring the first application to write to the first data store in accordance with the first data store interface package; determining data to be written to the first data store; writing data to the first data store according to the first schema; importing, by a second application or Application Protocol Interface (API), the first data store interface package; configuring the second application or Application Protocol Interface (API) to read from the first data store in accordance with the first data store interface package; and after data has been written to the first data store, reading data from the first data store according to the first schema.

[4] Some embodiments relate to a method comprising: determining a first data store interface package associated with a first data store, the first data store package comprising a first schema compatible with the structural representation of data in the first data store;

determining data to be written to the first data store; determining first schema and first data store identifier from the first data store interface package; writing data to the first data store according to the first schema; and sending a notification that the data has been written to the first data store to one or more of: (i) a data store migration module; (ii) a second application to be configured to read from the first data store; and (iii) an application protocol interface (API) to be configured to facilitate reading from the first data store by a second application.

[5] Determining a first data store interface package may comprise importing a published version of the first data store interface package or receiving the first data store interface package from the data store migration module.

[6] In some embodiments, prior to receiving the first data store package, the method comprises writing data to a second data store according to a second schema.

[7] Some embodiments relate to a method comprising: a method comprising: determining a first data store interface package associated with a first data store, the first data store package comprising a first schema compatible with the structural representation of data in the first data store; determining first schema and first data store from the first data store interface package; and reading at least a subset of the data from the first data store according to the first schema.

[8] In some embodiments, the method further comprises: receiving a request for the at least a subset of the data from a second application; and providing the at least the subset of data to the second application.

[9] Determining the first data store interface package may comprise importing a published version of the first data store interface package or receiving the first data store interface package from a data store migration module.

[10] In some embodiments, prior to determining the first data store interface package, the method comprises reading data from a second data store according to a second schema.

[11] Some embodiments relate to a method comprising: determining a first data store interface package associated with the first data store, the first data store package comprising a first schema compatible with the structural representation of data in the first data store, wherein a first application is to be configured to write to the first data store and an application protocol interface, API, is to be configured to facilitate reading from the first data store by a second application; providing the first data store interface package to the first application to cause the first application to write data to the first data store according to the first schema; receiving a notification that the data has been written to the first data store; responsive to receiving the notification, provide the first data store interface package to the API to cause the API to read at least a subset of the data from the first data store according to the first schema.

[12] Some embodiments relate to a method comprising: determining a first data store interface package associated with the first data store, the first data store package comprising a first schema compatible with the structural representation of data in the first data store, wherein a first application is to be configured to write to the first data store and an application protocol interface, API, is to be configured to facilitate reading from the first data store by a second application; publish the first data store interface package so that it is accessible to the first application and the API; receiving a notification that the data has been written to the first data store; and responsive to receiving the notification, sending a message to the API to configure the API to read data from the first data store according to the first schema based on the published first data store interface package. The method may further comprise deploying the first data store.

[13] In some embodiments, prior to providing the first data store interface package to the first application, the first application is configured to write data to a second data store according to a second schema, and the API is configured to read data from the second data store according to a second schema or according to a third schema different from the first and second schema, and wherein the API is configured to continue to read data from the second data store until the first data store interface package is provided to the API.

[14] In some embodiments, the first data store interface package comprises read and write functionality for interfacing with the first data store.

[15] In some embodiments, the first data store interface package comprises a round-trip test to assess consistency of the read and write functionality.

[16] The first data store may be a table, such as a DynamoDB table.

[17] Some embodiments relate to a system comprising: one or more processors; and memory comprising computer executable instructions, which when executed by the one or more processors, cause the system to perform any one of the described methods.

[18] Some embodiments relate to a computer-readable storage medium storing instructions that, when executed by a computer, cause the computer to perform the method of any one of the described methods.

Brief Description of Drawings

[19] Various ones of the appended drawings merely illustrate example embodiments of the present disclosure and cannot be considered as limiting its scope.

[20] Figures 1A to 1C are schematic illustrations is a schematic illustration of an example process of performing schema migration, according to some embodiments;

[21] Figures 2A to 2C are schematic illustrations of an example process of performing schema migration, according to some embodiments;

[22] Figure 3 is a block diagram of an example architecture comprising a system configured to perform the process of Figures 2A to 2C, according to some embodiments; and

[23] Figure 4 is a process flow diagram of a method of facilitating database migration, according to some embodiments;

[24] Figure 5 is a process flow diagram of a method of facilitating database migration, according to some embodiments; and

[25] Figure 6 is a process flow diagram of a method of facilitating database migration, according to some embodiments.

Description of Embodiments

[26] Embodiments generally relate to methods, systems, and computer-readable media for facilitating data store migration, and in some embodiments, for facilitating data store schema changes to accommodate read and write applications.

[27] Schema such as database or data store schema relates to how data is organised within the database or data store; that is, the structural representation of data. For example, a schema may include logical constraints such as table names, fields, data types and relationships between objects. Data may be written to and/or read from the database or data store in accordance with the schema of the respective database or data store. In some embodiments, the schema refers to the structural representation of data but not the semantics of the data represented.

[28] The schema may be enforced by a database management system, such as in SQL-based systems. For example, this may be the approach taken for relational database or data store such as SQL-based databases and data stores. However, some database management systems only support partial enforcement of schema when data is written and no enforcement when data is read (“Schema-in-Write”), or vice versa (“Schema-on-Read”). For example, with Schema-in-Write, the application writing the data is responsible for structuring the data being stored in the database or data store. For example, with Schema-on-Read, the application reading the data is responsible for structuring the retrieved data from the database or data store. This may be the case for non-relational databases or data stores, No-SQL or ‘document stores’ databases, such as DynamoDB.

[29] Referring to Figures 1A to 1C, there is shown a process for schema migration. Schema migration may involve changing from a first table or data store having a first schema

to a second table or data store having a second schema. As illustrated, a first application may be configured to write to a first data store, such as a table, and an Application Programming Interface (API) may be configured to read from the first data store. The first application and the API may be configured to conform to the first schema of the data store when writing to or reading from the first table.

[30] To change or migrate from using the first table having the first schema to the second table having the second schema, as a first step, as illustrated in Figure 1A, the API creates or deploys the second data store (data store #2). The second data store may be deployed alongside the first data store (data store #1). At this point, the first application is still writing to the first data store, and the second application is still configured to read from the first data store. This is because the second data store has not yet been populated with data. In creating or deploying the second data store, the API may be tasked with defining the second schema for the second data store.

[31] Referring to Figure 1B, the first application is configured to begin writing to the second data store. In order for the first application to write to the second data store in accordance with the second schema, the first application may be modified such that it writes to the data store in a format that matches or conforms to the second schema of the second table. In the meantime, the second application is still configured to read from the first data store.

[32] Where the first and second schemas are the same and a read/write interface between the first and second data stores does not change, but the first and second data stores are different, the first application (e.g. its source code) may be configured to update or change the version label of the data store to be used – e.g, version 1.0.3 is changed to the next major version, 2.0.0. However, where the first and second schemas are different and a read/write interface between the first and second data stores does change, the first application (e.g. its source code) may be configured to update or change the version label of the data store to be used, but may also be configured to accommodate the changes to the interface. For example, if the second schema was an expansion of the first schema, expanded to accommodate new or additional data, the first application may need to be modified to determine and provide or

write the new or additional data according to the second schema. In some embodiments, there may be a change in interface but no change or difference between the first and second schemas. For example, this may relate to how the data received from the application is translated or transformed into a form compatible for storing in the respective data store, and how the data retrieved from the data store is translated or transformed into a form compatible with the API.

[33] Referring to Figure 1C, once the first application has populated the second data store, the second application is configured to read from the second data store. To facilitate this, the second application may be modified such that it is accommodated to request and read data in the format of the second schema of the second table. As with the modification of the first application described above, this may involve modifications to the source code of the API. The first table may then be removed.

[34] This architecture requires the first application to write to a table or data store from which the second application reads. Having writing and reading functionality in two different places can be a source of friction. For example, the table schema expected by the first application and the table schema expected by the second application may diverge, and potentially cause problems for the second application in reading from the table or data store. To ensure that the expected table schema is consistent or identical across the first application and the second application tends to involve testing that spans both the first and second applications, and can be quite involved. Further, change management in terms of performing the process of Figures 1A to 1C to change the table can be complicated and can be particularly impactful for the second application when something goes awry.

[35] Described embodiments provide an improved process for performing data store or schema migration, such as facilitating changing from a first data store having a first data schema to a second data store having a second (and different) schema. The improved process may facilitate data store schema changes accommodating both read and write applications with minimal service interruption, and lessening or removing responsibility for managing schema changes, interface changes and/or database changes and improving or ensuring the first and/or second applications or APIs are operable with the respective data stores. This may

involve configuring or causing the applications and/or API to use a data store interface package for interfacing with the data store associated with the data store interface package. The data store interface package may enforce schema on read and on write.

[36] In some embodiments, a data store interface package is created or determined for a respective data store. The data store interface package may comprise an identifier of the respective data store to which it relates. The data store interface package may comprise or define a schema for the respective data store. The schema may define how data is to be represented structurally in the data store. For example, the schema may define that there are two data fields to be populated on writing to the data store, and that data in two fields will be returned to an application or API reading from the data store. The data store interface package may comprise an interface configured to translate or transform data between interface data representations, compatible with the first application and API, and schema data representation, compatible with the first data store. In other words, the interface may transform data to be written to the data store from the form received by the application into a form compatible with the data store, and to translate or transform data to be read by an application or API from the form stored in the data store into a form compatible with the application or API. For example, the interface may comprise functions and/or type definition, which when incorporated by the applications and/or API facilitate effective interoperability between the application and/or API and the data store.

[37] An example schema (in the Python language) may be:

```
class Contact:
    contactId: UUID
    businessId: str
```

[38] An example instance of the type may be:

```
Contact(contactId=UUID("a042505a-0f8d-4f7a-bae6-f1ad4e8751ca"),
        businessId="1ba6e35")
```

[39] An example representation in the data store (in JSON) may be:

```
{  
  "contactId": "a042505a-0f8d-4f7a-bae6-f1ad4e8751ca",  
  "businessId": "1ba6e35"  
}
```

[40] The data store interface package is associated or coupled to the respective data store to which it relates. For example, the data store interface package may be explicitly named or used in a list of dependencies for the application and/or API. Each version of the data store interface package is associated with a data store exclusive or unique to that version. This effectively makes explicit the version of the schema to be used when reading from or writing to the data store.

[41] By providing or publishing a data store interface package associated with a respective data store which can be imported by applications to use when writing and/or reading to/from the data store, the data representation within the data store is held or maintained independently from the consumer and producer of the data (for example, the first or second application and/or API). That is, neither the consumer nor the producer owns or is responsible for the data representation within the data store. In some embodiments, publication of and/or management of the data store interface package is managed by a data store migration module, as discussed in more detail below. The data store interface package, being independent of both the consumer and producer of the data may be capable of more effectively coordinating across the distinct components of the consumer and producer.

[42] The improved process, according to some embodiments, is illustrated in Figures 2A to 2C.

[43] Similar to the scenario of Figures 1A to 1C, Figures 2A to 2C illustrate an example process for data store or schema migration which involves changing from a first table or data store having a first schema to a second table or data store having a second schema. As illustrated, a first application may be configured to write to a first data store, such as a table, and an Application Programming Interface (API) may be configured to read from the first data store. The first application and the API may be configured to conform to the first schema of the data store when writing to or reading from the first table.

[44] A first data store interface package is published, and a respective first data store, such as a first table, is deployed. For example, in some embodiments, the first data store interface package may be published and the first data store deployed by a database management engine configured to manage the database. The first data store interface package comprises or defines a first schema to be used when writing to and reading from the first data store. The first data store interface package may comprise a first data store identifier, and/or a first interface for transforming the data between interface data representations, compatible with the first application and API, and schema data representation, compatible with the first data store. The first data store interface package may be associated with or coupled to the first data store.

[45] To write to the first data store, the first application imports the data store interface package which informs it of the first schema to be used when writing to the first data store. Similarly, to read from the first data store, the API imports the data store interface package which informs it of the first schema to be used when reading from the first data store. By importing the data store interface package, the application and API may be capable of interacting with the data store in the manner specified by the data store interface data package.

[46] The database management engine may also publish a second data store interface package and deploy a respective second data store. The database management engine may be configured to deploy multiple data stores, each with a respective data store interface package including a data store specific schema. In some embodiments, the database management engine may be configured to publish a data store interface package and deploy a respective data store in response to a trigger from a data store migration module. For example, the data store migration module may be configured to determine a suitable time for performing data store or schema migration.

[47] For example, and as illustrated in Figure 1B, a second data store interface package is published and a respective second data store deployed. The first application is prompted to change from writing to the first data store to writing to the second data store. To that end, the first application imports the second data store interface package, determines the second

schema and the second data store from the second data store interface package, and writes the relevant data to the second data store according to the second schema. At that time, the API continues to read data from the first data store according to the first schema.

[48] Once the relevant data has been written to the second data store, the API is updated to use the second data store interface package. For example, the first application or database management engine may inform the API that the second data store or table has been populated. The API then connects to the second data store, and configures itself to read from the second data store according to the second schema, as determined from the second data store interface package.

[49] Figure 3 illustrates an example network or architecture 300 comprising a system 302, according to some embodiments. The system 302 may be an accounting system or platform for managing accounts associated with a plurality of businesses or organisations. The accounting system 302 may be connected to or otherwise access a database 306 configured to store or record data associated with accounting records and financial accounts of the organisations. For example, the accounting system 302 may directly access the database 306, or may access the database via communications network 310.

[50] In some embodiments, the system 302 is configured to communicate with one or more financial institutions (e.g., banks) 304 or other third party financial systems, for example, via the communications network 310. The system 302 may be configured to receive financial information or banking data, such as financial records and/or financial documents, for example, as may be associated with transactions monitored by the system 302. For example, the system 302 may be arranged to receive bank feeds associated with transactions (for example, as may be required to be reconciled by the accounting system 202), and/or invoices or credit notes or receipts associated with transactions to be reconciled from third party entities. The system 302 may be configured to access and import the financial information over the communications network 310 through a bank feed or document (such as user- or accountant-created document), or over the communications network 310 via an application Programming interface (API). Accessing the financial information may comprise

providing account credentials of the plurality of users to obtain access to the financial information for the plurality of users.

[51] In some embodiments, items of information stored in the database 306 may be referred to as objects, and may each be associated with an object id. Examples of objects that may be stored in the database 306 include organisations, contacts, apps, business type, employees, etc.

[52] As depicted in Figure 3, a plurality of servers and/or computing devices 308 associated with the plurality of organisations may be configured to communicate with the accounting platform 304 across the communications network 310 to avail of the accounting services provided by the accounting platform 304. Similarly, a plurality of servers 312 may be configured to provide additional services to the accounting platform 304, for example to enhance or compliment the services provided by the accounting platform to the organisations, which may be customers (including individuals) of the accounting platform 404. For example, the servers 310 may be hosted by partners of the accounting platform 304, which may be pre-approved or vetted by the accounting platform 304.

[53] The system 302 comprises one or more processors 314, program memory 316, and data memory 318. The processor(s) 314 may include an integrated electronic circuit that performs the calculations such as a microprocessor, graphic processing unit, for example. Memory 316 and 318 may comprise both volatile and non-volatile memory for storing executable program code, and data, respectively. Memory 316 comprises program code (for example modules or engines) which when executed by the processor 314, provides the various computational capabilities and functionality of the system 302. Memory 318 comprises data accessible to the program code to facilitate the program code in executing its functionality. In some embodiments, any part of the data of data memory 318 may be also or instead stored at a remote location from the system 302, but accessible to the system 302 to allow the system to perform its functionality. For example, data of data memory 318 may be stored in database 306.

[54] The system 302 comprises a network interface 320 to facilitate communications with components of the system 302 across the communications network 310. The network interface 320 may comprise a combination of network interface hardware and network interface software suitable for establishing, maintaining and facilitating communication over a relevant communication channel.

[55] In some embodiments, the system 304 may be implemented as a distributed system comprising multiple server systems configured to communicate over a network to provide the functionality of the system 302. For example, one or more of the program code (for example modules or engines) may be deployed on one or more disparate or remote servers, which may cooperate to provide the functionality of the system 302 described.

[56] Program memory 316 may comprise one or more builder modules 322. The builder module(s) 322 are configured to determine connections or relationships between entities in raw data to facilitate the construction and/or development of a graph data structure (“graph”) 324. For example, the raw data may be collated and/or stored in database 306, such as accounting and/or financial information. The relationship may be represented by an edge of the graph data structure 324 and the entities are represented by vertices interconnected by the edge.

[57] The edge(s) and/or vertices may be determined directly, or identified, from the raw data. The edge(s) and/or vertices may be determined from information already stored in the graph 324, and/or from other external sources. The edge(s) and/or vertices may be determined by inference. Accordingly, in some embodiments, the builder module(s) 204 may comprise an inference engine 326 configured to infer one or more edges and/or vertices based on the raw data, data from the graph 324 and/or other data sources. The inference engine 326 may be configured to make inferences about a specific topic or theme. In some embodiments, one or more builder modules 322 may be configured to determine connections or associations between objects of a first type and objects of a second type. For example, the first and second object types may be different object types or the same object types.

[58] For example, in some embodiments, one or more builder modules 322 may be configured to determine connections between organisations and apps being used by the organisations. The builder module(s) 322 may request or subscribe to receive application connection data, such as notifications or events indicative of app or API connection or call requests by organisations to connect to avail of services and/or products provided by servers 312 to the organisations using the accounting platform 302. In some embodiments, the builder module(s) 322 may request or fetch the application connection data, which may be maintained and stored in database 306.

[59] In some embodiments, the builder module(s) 322 and/or the accounting system or platform 302 may be configured to monitor or track activity on the API of the servers 312. The builder module(s) 322 and/or accounting system 302 may be configured to track which organisation is making a call to which API (or which service provided by server 312) and when. The builder module(s) 322 and/or accounting system 302 may be configured to track when an organisation is making calls to an API of a server 312, how many calls the organisation is making calls to the API of the server 312 and/or how regularly. Using this application connection information, the builder module(s) 322 (or the inference engine 326 of the builder module 322) may determine or infer that a particular organisation has installed a particular app. In some embodiments, the builder module(s) 322 (or the inference engine 326 of the builder module 322) may determine or infer that a particular organisation has uninstalled a particular app based on a lack of API calls over a period of time.

[60] In some embodiments, one or more builder modules 322 are configured to determine connections, such as entity relationships, between organisations and contacts, and contacts and contacts, such as to determine where the contact entity is the same entity as the organisation entity or another contact entity, or does business with the organisation entity or another contact entity, for example. For example, the raw data used by the system 302 may comprise contact and organisation data. Each organisation that uses or has an account with the accounting platform may have or maintain a contact list or contact address book. For example, the contacts in the contact list may be customers and/or suppliers of the organisation. In some embodiments, the accounting platform 302 may maintain a contact table comprising details of at least some of the contact lists of organisations that have an

account with the accounting platform 302. In some embodiments, the creation of a new contact entry in a contact list or address book, for example, by a user, and/or in response to the reconciling of a transaction between an organization and a contact using a recommended or suggested contact (which may be automatic), may instigate an instruction (e.g. a streamed event) to be sent to a builder module 322 to create an edge between the organisation and the contact. For example, the builder module 322 may be subscribed to receive events indicative of such actions or occurrences.

[61] Once an edge between a pair of vertices has been determined, the builder module 322 may generate message(s) (for example, an event) for transmitting to a write interface of a graph interface module 328 for mapping the edge and vertices information to relevant table(s) of the graph data structure 324. The message comprises information indicative of the edge and the vertex pair. In some embodiments, fields, such as “attributes” of the edge element may be populated or annotated with an indication of a strength of the determination or inference, contributing evidence, and/or dates/versions of inference.

[62] The graph interface module 328 may be configured to communicate or cooperate with one or more first application or service modules 330 to provide services, views and/or products, for example to organisations or users of the accounting platform 302. The service module(s) 330 may be configured to provide service or views based on or built on information derived from the graph 324. A read interface of the graph interface module 328 is configured to receive read and process read requests received from the service module(s) 330 and to provide the required information to the service module(s) 330.

[63] In some embodiments, the first or service application module 330 may be configured to determine data for writing to the database 333, such as a first data store 338, or a second data store 340. The first or service application module 330 may determine the data from the graph data structure 324. For example, the data may comprise object type correlation data. The object type correlation data may comprise a plurality of pairs of objects of a first object type considered to be correlated, such as pairs of applications or “app” considered to be relatively highly correlated with one another. The first or service application module 330

may be configured to write to data stores or tables of the database 330 in batches. The write request may conform to the schema associated with the respective data store.

[64] In some embodiments, the first or service application module 330 may be configured to determine or derive information for writing to the first data store (or second data store, or subsequent data stores) according to one or more of the methods described in International (PCT) patent application PCT/NZ2022/050152 entitled “Methods and systems for building and/or using a graph structure”, filed on 24 November 2022, the entire content of which is incorporated herein by reference.

[65] In some embodiments, the second application module 334 may comprise an application (“app”) recommendation module configured to generate recommendations, such as to organisations or users of an accounting platform, as to which apps they may be interested in installing. The app recommendation module may be configured to generate recommendations based on data determined from a data store of the database. For example, the application recommendation module may be configured to determine that a user has an interest in a particular app, (for example, the user may have installed the particular app, or may be searching for a particular app), and based on the user’s interest in that app, the app recommendation module may request, from the data store (in some embodiments, via API) a list of all (or any) apps linked or associated with that particular app. Similarly, app recommendation module may be configured to determine that a user has an interest in a particular app, (for example, the user may have installed the particular app, or may be searching for a particular app), and based on the user’s interest in that app, the app recommendation module may send a read request to the data store (in some embodiments via the API), for a list of all (or any) apps associated or linked to that particular app. The read request may conform to the schema associated with the respective data store. For example, the app recommendation module may recommend App 1 to Org Z because the app recommendation module determines that the user associated with Org Z was interested in App 2, and that App 1 and App 2 are associated in the data store.

[66] In some embodiments, the second application module 334 may comprise a business register module. The business register module may be configured to provide details of a

business register based on data stored in the data store. For example, the data may comprise one or more contacts, entity names and/or aliases associated with an entity identifier. For example, the association may be “is associated with”, “is the same as”, “works at”, “works for” etc. The business register module may be configured to receive a request for organisation or contact details from a user, such as a user of the accounting platform associated with a particular organization. The business register module may be configured to send a read request to the data store (in some embodiments via the API), wherein the request conforms to the schema associated with the respective data store.

[67] In some embodiments, the second application module 334 may comprise a correlation module configured to determine correlation data. The correlation data may be indicative of a correlation or a level of correlation between a first object of first object type and a second or other object(s) of the first object type. The first object type may be organisations, contacts, apps, business type, employees, etc. For example, the first object type may be a contact’s business type and the second object type may be an organisation. The correlation module may be configured to make recommendations to a user based on the correlation data. The correlation module may be configured to send a read request to the data store for correlation data (in some embodiments via the API), wherein the request conforms to the schema associated with the respective data store. For example, on determining that a user has stored, or has recently added, a renovation business as a service provider to its contact list, the correlation module may recommend one or more decorator businesses to the user, on the basis that if the user has recently had some renovation work done, they may be interested in engaging a decorator.

[68] Program memory may comprise a database management engine 332. The database management engine 332 may be configured to manage data stored in database 330. The database management engine 332 may comprise a data store migration module 334. The data store migration module 334 may be configured to facilitate data store schema changes to accommodate read and write applications. For example, the database management engine 332 and/or data store migration module 334 may be configured to generate, publish and/or make available, data store interface packages, such as first data store interface package 342, and second data store interface package 344. The data store package(s) 342, 344 may be

associated with, coupled to a respective data store 338, 340. The data store(s) 338, 340 may be tables, such as DynamoDB tables. The data store interface package(s) 342, 344 may comprise read and write functionality for interfacing with the respective data store 338, 340.

[69] The data store interface package 342, 344 may comprise round-trip test functionality to assess consistency of the read and write functionality. For example, round-trip test functionality may comprise computer executable code configured to cause test input data to be written to the associated data store using write functionality of the data store interface package, reading the data from the data store using the read functionality of the data store interface package, and comparing the data written with the data read to determine an assessment of the integrity of the data (e.g. whether values are present and correct, whether the structure is unchanged, etc.); in other words, whether the same data was read as was written.

[70] Referring now to Figure 4, there is shown a process flow of the method 400 of facilitating database schema changes, according to some embodiments. For example, the method 400, or at least some of the steps of the method 400, may be implemented by the processor(s) 314 of the system 302 executing code (for example, first application module 330) stored in program memory 316.

[71] At 402, a first application module 330 may be configured to write, and/or may write data to a second data store 340, such as a table, according to a second schema. Examples of the first application are discussed in more detail above.

[72] At 404, the first application module 330 determines the first data store interface package 342. For example, the first application module 330 may receive the first data store interface package 342 from the data store migration module 332 or may retrieve or import it from data memory 318, for example, following receipt of a prompt to do so from the data store migration module 342.

[73] At 406, the first application module 330 determines data to write to the first data store 338. For example, the first application module 330 may determine data to be written to

the first data store 338 from the graph data structure 324, as discussed above. In some embodiments, the first application module 330 may be configured to determine or derive information for writing to the first data store (or second data store, or subsequent data stores) according to one or more of the methods described in International (PCT) patent application PCT/NZ2022/050152 entitled “Methods and systems for building and/or using a graph structure”, filed on 24 November 2022, the entire content of which is incorporated herein by reference.

[74] At 408, the first application module 330 determines the first schema and the first data store 338 from the first data store interface package 342. The first application module 330 may configure itself to write data to the first data store 338 according to the first schema. The first application module 330 may import functionality and/or code from the first data store interface package 342 to enable the first application module 330 to write data to the first data store in a manner that is compatible with the first data store. For example, the first application module 330 may use an interface defined or provided in the first data store interface package 342 to transform data from an interface representation to a data store representation in accordance with the first schema.

[75] In some embodiments, facilitating a change from a first data store to a second data store does not involve a change in schema. For example, both the first and second data store may represent data structurally according to a common or a same schema.

[76] In some embodiments, facilitating a change from a first data store to a second data store does not involve a change in interface. For example, both the first and second data store may use the same interface; that is the same functionality to convert or transform between interface data (such as that provided to the interface by the first application, or read by the second application or API) and schema data (such as that stored in the data store).

[77] In some embodiments, facilitating a change from a first data store to a second data store involves a change in schema and/or a change in interface. In such embodiments, modifications to the first application, second application and/or API may be necessary to accommodate the change. For example, it may be necessary to determine what data is to be

provided to the first data store where a change in schema involves the expansion of the scheme to include an additional data field. It may be necessary to provide the data in a different format to accommodate the interface of the data store interface package.

[78] At 410, the first application module 330 writes the data to the first data store 338 according to the first schema.

[79] Referring now to Figure 5, there is shown a process flow of the method 500 of facilitating database schema changes, according to some embodiments. For example, the method 500, or at least some of the steps of method 500, may be implemented by the processor(s) of the system executing code (for example, the database management engine 331 or data store migration module 332) stored in program memory 316.

[80] At 502, the data store migration module 332 may determine a first data store interface package 342 associated with a first data store 338. The first data store package 342 may comprise a first schema associated with the first data store 338. The first data store 338 may be a table, such as DynamoDB table.

[81] The first data store interface package 342 may comprise read and write functionality for interfacing with the first data store 338, as discussed above. In some embodiments, the first application module 330 is to be configured to write to the first data store 338 and a second application module 334 is to be configured to read from the first data store 338. In some embodiments, the first application module 330 is to be configured to write to the first data store and an Application Programming Interface, API, 336 is to be configured to facilitate reading from the first data store 338 by a second application module 334.

[82] At 504, the data store migration module 332 may provide the first data store interface package 342 to the first application module 330 to cause the first application module 330 to write data to the first data store 342 according to the first schema. In other embodiments, the data store migration module 332 may cause the first application module 330 to retrieve or import the first data store interface package 342, for example, from a data

memory 318. For example, the data store migration module 332 may publish the first data store interface package 342, and may deploy the first data store 338.

[83] Prior to being configured to read from the first data store 338 according to the first schema, the second application module 334 or API 336 may have been configured to read from the second data store 340 according to a second or third schema, as discussed below. The second application module 334 or API 336 may continue to read from the second data store 340 according to a second or third schema while the first application module 330 populates the first data store 338 with data.

[84] At 506, in some embodiments, the data store migration module 332 determines that the data has been written to the first data store 338. The data store migration module 332 may receive a notification that the data has been written to the first data store 338, for example, from the first application module 330.

[85] At 508, in response to determining that the data has been written to the first data store, the data store migration module 332 may provide, or otherwise make accessible (for example, through publication), the first data store interface package 342 to the second application module 334 or API 346 to configure the second application module 334 or API 346 to read data from the first data store 338 according to the first schema. For example, the second application module 334 or API 346 may connect to the first data store 338. In some embodiments, the data store migration module 332 may provide, or otherwise make accessible (for example, through publication), the first data store interface package 342 to the second application module 334 or API 336 before determining that the data has been written to the first data store 338. In such an embodiment, once the data store migration module 332 has determined that the data has been written to the first data store 338, the data store migration module 332 may then provide an instruction or message to the second application module 334 or API 336 to configure itself to read data from the first data store 338 according to the first schema. In some embodiments, the data store migration module 332 may cause the second application module 334 or API 336 to retrieve or import the first data store interface package 342, for example, from a data memory 318, and may configure itself to read data from the first data store 338 according to the first schema.

[86] Referring now to Figure 6, there is shown a process flow of the method 600 of facilitating database schema changes, according to some embodiments. For example, the method 600, or at least some steps of method 600, may be implemented by the processor(s) of the system executing code (for example, API 336 or second application module 334) stored in program memory 316.

[87] At 602, a second application module 334, or an API 336, may read data, such as a subset of data, from the second data store 340. Examples of the second application are discussed in more detail above. In some embodiments, the second application module 334 or API 336 may read the subset of data from the second data store 340 according to the second schema.

[88] At 604, the second application module 334 or API 336 determines the first data store interface package. For example, the second application module 334 or API 336 may receive the first data store interface package 342 from the data store migration module 332 or may retrieve it from data memory 318, for example, following receipt of a prompt to do so from the data store migration module 332.

[89] At 606, the second application module 335 or API 336 determines the first data store 338 and the first schema from the first data store interface package. The second application module 335 or API 336 may configure itself to read data from the first data store 338 according to the first schema. The second application module 335 or API 336 may import functionality and/or code from the first data store interface package 342 to enable the second application module 335 or API 336 to read data from the first data store in a manner that is compatible with the first data store. For example, the second application module 335 or API 336 may use an interface defined or provided in the first data store interface package 342 to transform data from a schema representation to an interface representation in accordance with the first schema.

[90] At 608, in some embodiments, the API 336 may receive a request for a subset of data from a requesting application.

[91] At 610, the API 336 or second application module 334 reads the subset of data from the first data store 338 according to the first schema. In the case of the API 336 reading the subset of data, the API 336 may provide the subset of data to the requesting application, at 612.

[92] It will be appreciated by persons skilled in the art that numerous variations and/or modifications may be made to the above-described embodiments, without departing from the broad general scope of the present disclosure. The present embodiments are, therefore, to be considered in all respects as illustrative and not restrictive.

CLAIMS

1. A method comprising:

providing a first data store interface package associated with a first data store, the first data store package comprising a first schema compatible with the structural representation of data in the first data store;

importing, by a first application, the first data store interface package;

configuring the first application to write to the first data store in accordance with the first data store interface package;

determining data to be written to the first data store;

writing data to the first data store according to the first schema;

importing, by a second application or Application Programming Interface (API), the first data store interface package;

configuring the second application or API to read from the first data store in accordance with the first data store interface package; and

after data has been written to the first data store, reading data from the first data store according to the first schema.

2. A method comprising:

determining a first data store interface package associated with a first data store, the first data store package comprising a first schema compatible with the structural representation of data in the first data store;

determining data to be written to the first data store;

determining first schema and first data store identifier from the first data store interface package;

writing data to the first data store according to the first schema; and

sending a notification that the data has been written to the first data store to one or more of:

(i) a data store migration module;

(ii) a second application to be configured to read from the first data store; and

(iii) an application Programming interface (API) to be configured to facilitate reading from the first data store by a second application.

3. The method of claim 2, wherein determining a first data store interface package comprises importing a published version of the first data store interface package.

4. The method of claim 2, wherein determining a first data store interface package comprises receiving the first data store interface package from the data store migration module.

5. The method of any one of the preceding claims, wherein prior to receiving the first data store package, writing data to a second data store according to a second schema.

6. A method comprising:

determining a first data store interface package associated with a first data store, the first data store package comprising a first schema compatible with the structural representation of data in the first data store;

determining first schema and first data store from the first data store interface package; and

reading at least a subset of the data from the first data store according to the first schema.

7. The method of claim 6, further comprising:

receiving a request for the at least a subset of the data from a second application; and

providing the at least the subset of data to the second application.

8. The method of claim 6 or 7, wherein determining the first data store interface package comprises importing a published version of the first data store interface package.

9. The method of claim 6 or 7, wherein determining the first data store interface package comprises receiving the first data store interface package from a data store migration module.

10. The method of any one of claims 6 to 9, wherein prior to determining the first data store interface package, reading data from a second data store according to a second schema.

11. A method comprising:

determining a first data store interface package associated with the first data store, the first data store package comprising a first schema compatible with the structural representation of data in the first data store, wherein a first application is to be configured to write to the first data store and an application Programming interface, API, is to be configured to facilitate reading from the first data store by a second application;

providing the first data store interface package to the first application to cause the first application to write data to the first data store according to the first schema;

receiving a notification that the data has been written to the first data store;

responsive to receiving the notification, provide the first data store interface package to the API to cause the API to read at least a subset of the data from the first data store according to the first schema.

12. A method comprising:

determining a first data store interface package associated with the first data store, the first data store package comprising a first schema compatible with the structural representation of data in the first data store, wherein a first application is to be configured to write to the first data store and an application Programming interface, API, is to be configured to facilitate reading from the first data store by a second application;

publish the first data store interface package so that it is accessible to the first application and the API;

receiving a notification that the data has been written to the first data store; and

responsive to receiving the notification, sending a message to the API to configure the API to read data from the first data store according to the first schema based on the published first data store interface package.

13. The method of any one of claims 11 or 12, further comprising deploying the first data store.

14. The method of any one of claims 11 to 13, wherein prior to providing the first data store interface package to the first application, the first application is configured to write data to a second data store according to a second schema, and the API is configured to read data from the second data store according to a second schema or according to a third schema different from the first and second schema, and wherein the API is configured to continue to read data from the second data store until the first data store interface package is provided to the API.

15. The method of any one of the preceding claims, wherein the first data store interface package comprises read and write functionality for interfacing with the first data store.
16. The method of any one of the preceding claims, wherein the first data store interface package comprises a round-trip test to assess consistency of the read and write functionality.
17. The method of any one of the preceding claims, wherein the first data store is a table.
18. The method of any one of the preceding claims, wherein the first data store is a DynamoDB table.
19. A system comprising:
 - one or more processors; and
 - memory comprising computer executable instructions, which when executed by the one or more processors, cause the system to perform the method of any one of claims 1 to 18.
20. A computer-readable storage medium storing instructions that, when executed by a computer, cause the computer to perform the method of any one of claims 1 to 18.