Single Source of Truth (SSOT) for Service Oriented Architecture (SOA)

Candy Pang and Duane Szafron

Department of Computing Science, University of Alberta, Edmonton, Alberta, Canada
{cspang,dszafron}@ualberta.ca

Abstract. Enterprises have embraced Service Oriented Architecture (SOA) for years. With SOA, each business entity should be the Single Source of Truth (SSOT) of its data, and offer data services to other entities. Instead of sharing data through services, many business entities still share data through data replication. Replicating data causes inconsistencies and interoperability challenges. Even when there is a single authoritative source, that resolves inconsistencies, the data copies may end up being out-of-sync and cause errors. This paper describes how to use a SSOT service to eliminate data replication, enforce data autonomy, advocate data self-containment, and enhance data maintenance. Both mutable and immutable SSOT relationships (mappings) are considered. This paper describes the challenges, solutions, interactions and abstractions between the SSOT data service providers and the loosely coupled data consumers. It also assesses the performance and future usage of a SSOT service.

Keywords: Single Source of Truth, Service Composition, Service Oriented Architecture (SOA), Software Design Concept, Software Engineering.

1 Introduction

Before embracing the Service Oriented Architecture (SOA), enterprises or business entities used to share data through data replication. Replicating data across multiple systems gives rise to inconsistencies and interoperability challenges. In some cases, one of the systems is treated as the "authoritative" source or the Single Source of Truth (SSOT). The authoritative source's data is replicated to the clients' databases, resulting in data layer synchronization challenges. Independent client's data transformation or modification can result in data discrepancies that require manual intervention. A better solution is to hide the data layer from the clients in the SOA.

In the SOA, a SSOT should associate with clients through the service layer, instead of the data layer. This would alleviate data replication and the related problems. The authoritative source identified as the SSOT should provide data services for the clients. In this approach, clients maintain mappings to the SSOT data, but do not replicate the SSOT data. Unfortunately, many enterprises have yet to overhaul data layer replication into a SSOT service. Lack of experience in SSOT service implementation may have hindered enterprises from the migration.

This paper describes a SSOT service model for two common data sharing scenarios: mutable and immutable data sources. We use two motivating examples to illustrate the variants: (a) the management of Postal Codes (PC) and (b) the management of Electronic Patient Records (EPR). The proposed SSOT service model is useful for any business entity that maintains full ownership of its data, and does not want the clients to duplicate its data, but allows data access by restricted queries. The value of the model will be illustrated by the PC and EPR examples.

Most business applications use addresses. In Canada, Canada Post is the single authoritative agency that manages PCs for mail-delivery addresses. Each address should have exactly one PC, while each PC covers an area with multiple addresses. Most business applications collect address information from their customers, and store customers' addresses with PCs in their local databases. Periodically, business applications also replicate Canada Post's PCs to their local databases for data-validation purpose. For example, an application using billing addresses, which require PCs, may have a Billing_Address table and a Postal_Code table as shown in Fig. 1(a). The Postal_Code table contains all valid PCs periodically replicated from Canada Post. Before adding a new address to the Billing_Address table, the application checks the validity of the provided PC, i.e., whether the PC exists in the Postal_Code table. If so, the application will add the new address to the Billing_Address table. This process can only validate the existence of the PC, but it cannot validate whether the PC is correct for that address. There is a mapping between the PC and the billing address.

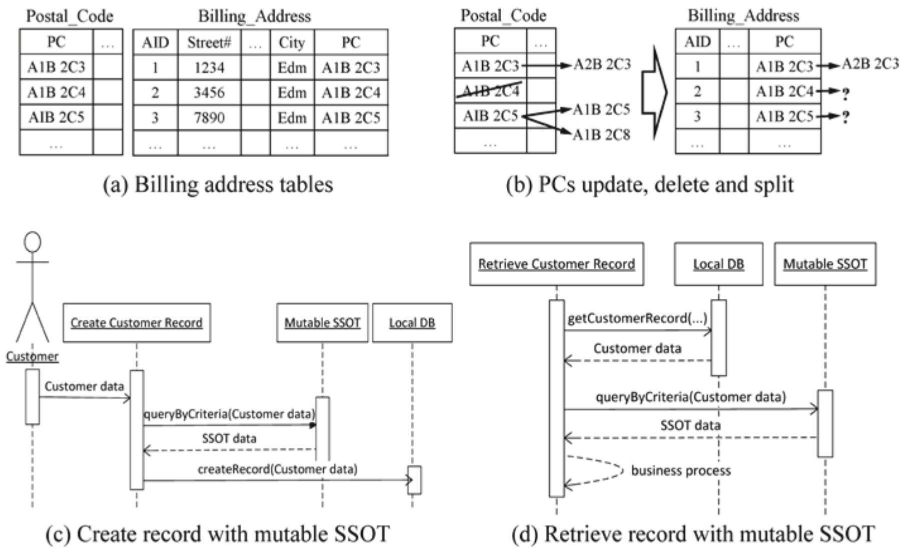


Fig. 1. The Postal Codes Use Case

A mapping that changes over time is called a mutable mapping, and a mapping that does not change is immutable. Canada Post changes PCs from time to time. PCs can be inserted, updated, deleted, split or merged. The application needs to synchronize the Postal_Code table with Canada Post, and, if necessary, to correct the PCs in the

Billing_Address table. Since the PC that is mapped to a billing address can change over time, the mapping between PC and billing address is an example of a mutable mapping. Mutable mappings are often subject to synchronization errors. For example, Canada Post only provides PC changes to subscribers monthly. Therefore, the subscribers' Postal_Code tables are out-of-sync with Canada Post most of the time. As shown in Fig. 1(b), when a PC is updated (from A1B 2C3 to A2B 2C3), the corresponding mappings in the Billing_Address table can be updated. However, when a PC is deleted (from A1B 2C4 to none) or split (from A1B 2C5 to A1B 2C5 and AIB 2C8), there is no simple solution to correct the mappings in the Billing_Address table, by using the updated PC list.

As a second example, let us consider a regional Electronic Patient Record (EPR) system that manages patients' health numbers (PHNs), names and contacts. Each patient in the region receives services from multiple healthcare providers, with different specialties. Each healthcare provider obtains patient information directly from the patient during the patient's visit, containing information included in the EPR. Then, each provider independently stores the patient's information in its local database. In this model, a provider-specific patient record may be inconsistent with the regional EPR. In principle, each provider-specific patient record could be mapped to a corresponding EPR in the regional authoritative system. In this case, the mapping between an EPR and a provider-specific record is immutable. The mapping is immutable despite the fact that patient's data may change. For example, when a patient changes his/her name, the patient's EPR will be updated, but the patient's EPR is still mapped to the same provider-specific record. Therefore, the mapping is immutable.

The SOA paradigm can alleviate data synchronization issues. Clients using data that already exists in an authoritative source will not replicate the authoritative data in their local databases. Instead, the authoritative source acts as a SSOT service that provides clients the authoritative data. In the examples above, Canada Post serves as the SSOT service for PCs and a regional health authority provides the SSOT service for EPRs. Clients access PCs and EPRs by invoking SSOT services, without replicating SSOT data in their local databases. Therefore, clients do not need to manage and synchronize data with the SSOT. The SSOT can also shield its autonomy from the clients. We advocate the SSOT service model over data-replication.

This paper is structured as follows. Section 2 and 3 illustrate the challenges and solutions associated with the mutable and the immutable SSOT by the PC and the EPR use cases respectively. Section 4 evaluates the performance of the SSOT service. Section 5 describes the related works. Section 6 recommends future works, and Section 7 concludes the paper by enumerating SSOT's benefits.

2 Mutable SSOT Service

A SSOT service that manages mutable mappings is called a mutable SSOT service. In this case, the mapping between a SSOT record (each individual PC in our example) and a client application record (each billing address in our example) can change over time. Clients need to invoke the mutable SSOT service with query criteria. Therefore, a mutable SSOT service supports at least the query-by-criteria operation.

For example, in the PC use case, Canada Post maintains a mutable PC SSOT service that provides a single web service operation, PC-query. The PC-query operation takes PC query criteria as input. Clients may specify Street Number, Number Suffix, Unit/Suite Apartment, Street Name, Street Type, Street Direction, City, and/or Province as criteria. The PC-query operation returns a set of PCs that match the criteria.

This PC SSOT service can replace data replication. For example, in Fig. 1(a), the PC column in the Billing_Address table and the Postal_Code table are replicated data that can be dropped in favor of invoking the PC-query operation provided by the mutable PC SSOT service. The PC query criteria will come from the remaining columns (e.g. Street#, City) in the Billing_Address table. When the application needs the PC of a billing address, it will use the address data in the Billing_Address table as query criteria to retrieve the PC from the PC SSOT.

The rest of this section will use the PC use case to illustrate how clients can use a mutable SSOT service to replace data replication.

2.1 Client-Record Creation

Fig. 1(c) depicts how an application can use mutable SSOT service for data validation during record creation. In the PC use case, when the application receives a new billing address from a customer, the application queries the PC SSOT using the customer address. If the PC SSOT returns a single valid PC, then the address is valid. The application proceeds to create a new Billing_Address record for the customer.

If the PC SSOT returns more than one PC, then the customer address is not definitive. For example, if the customer address contains only the city field, then the PC SSOT will return all the PCs for the city. In principle, the application should not accept a non-definitive address as a billing address. Therefore, the application would seek additional address details from the customer. Similarly, if the PC SSOT returns no PC for the customer address, the application should alert the customer that the address is invalid and request the user to take remedial action.

In contrast, the data-replication model may allow non-definitive or invalid billing addresses in the application's database. Using a mutable SSOT service not only eliminates data replication, but also enhances data quality.

Different clients may have different processes for the SSOT response. The application in the PC use case expects a single valid PC from the response. Other applications may iterate the response records to select the most desired result. To support different clients' processes, the mutable SSOT query-by-criteria operation may return additional information. For example, the PC-query operation may return other address fields (Street Number, Unit/Suite Apartment, Street Name, etc.) in addition to the PC. Clients can use the additional address fields to filter the response records.

2.2 Client-Record Retrieval

Since the SSOT data is excluded from the clients' local databases, each client needs to combine its local data with the SSOT data to compose the complete data records. The client data retrieval process is depicted in Fig. 1(d).

In the PC use case, the application first retrieves a Billing_Address record from its local database, then uses the local address data as query criteria to invoke the PC-query operation, and retrieve the up-to-date PC from the PC SSOT. If the PC of the billing address has changed since the last retrieval, then the PC SSOT will return a different PC from the last retrieval, but it will be the correct PC.

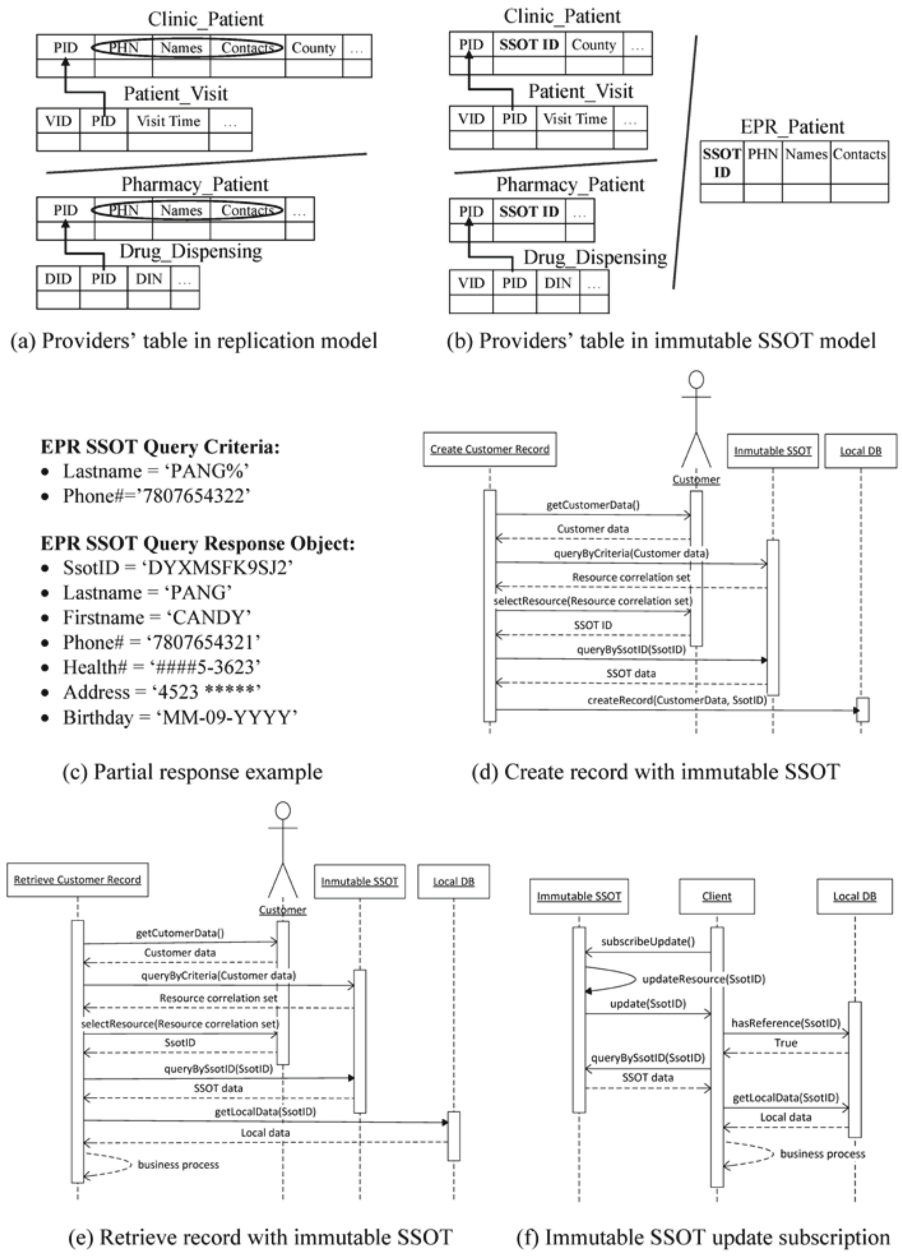
3 Immutable SSOT Service

A SSOT service that manages an immutable mapping is called an immutable SSOT service. The mapping cannot change over time. Each record in the client application has a permanent static relationship with a SSOT record. To establish this mapping, a client record is associated with a SSOT record using a unique permanent single source of truth identifier (SSOT-ID). A SSOT-ID is conceptually equivalent to the resource identifier in the Resource Description Framework (RDF) [1] and the unique identifier in the Representational State Transfer (REST) [2] architecture. Each record represents a resource in the SSOT. We will use the term resource instead of record since each SSOT resource may contain multiple child-records. For example, an EPR may contain multiple names, addresses and phone numbers as child-records. Each resource has a unique permanent identifier called the SSOT-ID. Clients use the query-by-SSOT-ID operation to retrieve resource details, which include the child-records.

The rest of this section will illustrate the immutable SSOT service through the EPR use case. Assume that a clinic application and a pharmacy application both need patients' health numbers (PHNs), names and contacts, along with their own provider-specific data. Traditionally, in the data-replication model, the clinic application would have a Clinic_Patient table and a Patient_Visit table, and the pharmacy application would have a Pharmacy_Patient table and a Drug_Dispensing table, as shown in Fig. 2(a). The clinic application assigns a county to each patient using the patient's home address, while the pharmacy application does not. In the data-replication model, the PHNs, names and contacts located in the clinic's and pharmacy's databases may have errors or be inconsistent with the authoritative EPR system. When patients move, patients must notify the EPR authority, the clinic and the pharmacy individually.

Actually, PHNs, names and contacts are readily available in the regional EPR system. An immutable EPR SSOT service can eliminate data replication from the clinic and the pharmacy databases. After eliminating the replicated EPR data, Fig. 2(b) shows the new clinic application and pharmacy application tables. In the new Clinic_Patient and Pharmacy_Patient tables, the EPR columns are replaced by the SSOT-ID column. Unlike the PC use case, immutable SSOT clients do not routinely use the query-by-criteria operation to obtain SSOT data. Instead, clients can invoke the query-by-SSOT-ID operation to retrieve SSOT data from the immutable SSOT service.

Each immutable SSOT service should provide at least four operations: (a) query-by-criteria, (b) query-by-SSOT-ID, (c) update subscription, and (d) deletion subscription. The potential risks of using the immutable SSOT service query-by-criteria operation are illustrated in Section 3.1. The rest of the sub-sections describe different usages of the immutable SSOT operations.



EPR SSOT Query Criteria:
- Lastname = 'PANG%'
- Phone#='7807654322'

EPR SSOT Query Response Object:
- SsotID = 'DYXMSFK9SJ2'
- Lastname = 'PANG'
- Firstname = 'CANDY'
- Phone# = '7807654321'
- Health# = '####5-3623'
- Address = '4523 *****'
- Birthday = 'MM-09-YYYY'

(c) Partial response example

Fig. 2. The Electronic Health Record Use Case

3.1 Query-by-Criteria

An immutable SSOT service could provide the same query-by-criteria operation as a mutable SSOT service. Clients could query the SSOT service by criteria and receive a correlated set of results matching the criteria. However, in some situations, the immutable SSOT query-by-criteria operation may sustain a privacy violation risk. In the EPR use case, if the query-by-criteria operation returns all resources matching any given set of criteria, then any client can browse the EPR data, which violate patients' privacy. For example, a client may specify Firstname='JOHN' as the query criteria for the EPR SSOT query-by-criteria operation. In response, the EPR SSOT returns all patients with first name equals 'JOHN'. The large response set may violate many patients' privacy, since many of the set may not be patients of the querying facility.

To avoid browsing, the query-by-criteria operation could specify a maximum number of returned records (i.e. query-limit). If the response set is larger than the query-limit, the service would return an error. At which point the client needs to refine the query criteria and queries again.

A safer query-by-criteria operation may also demand more than one query criterion to avoid brute force hacking. For example, the EPR SSOT query-by-criteria operation could reject single criterion queries to avoid PHN cracking by querying with randomly generated PHN's until a valid resource is returned.

For further privacy protection, the query-by-criteria response set can be filtered to contain partial data. The partial data must include the complete SSOT-ID and sufficient information to identify a SSOT resource. Fig. 2(c) shows a set of query criteria for the EPR SSOT query-by-criteria operation, and one partial response record. The partial record includes the SSOT-ID and only part of the PHN, address and birthday. Using the partial record, a client should be able to determine whether the EPR maps to the targeted patient. Once a partial record is selected, the client can use the SSOT-ID to retrieve the resource details using the query-by-SSOT-ID operation.

3.2 Query-by-SSOT-ID

If the query-by-criteria operation returns only partial data for privacy protection, the immutable SSOT service must provide a query-by-SSOT-ID operation, which takes a SSOT-ID as input. In response to a query-by-SSOT-ID request, the immutable SSOT provides details of the resource corresponding to the provided SSOT-ID, but only the details that the client is authorized to see. This allows the SSOT service to distinguish between client access permissions, providing different information to different facilities, such as pharmacies, clinics and acute-care facilities.

To protect data privacy, the immutable SSOT should include a proper auditing mechanism to detect, identify and stop improper browsing behavior or unlawful use of data. In the EPR use case, if a client continually invokes the EPR SSOT query-by-SSOT-ID operation with randomly generated or guessed SSOT-IDs, the EPR SSOT auditing mechanism should detect and deter the client.

Each SSOT-ID is effectively a foreign key to a remote SSOT resource. Since the SSOT and clients are loosely coupled, the foreign key constraints in the client data cannot be enforced at the SSOT. An alternate foreign key constraint handling mechanism will be discussed in the later sub-sections.

3.3 Client-record Creation

Fig. 2(d) depicts the role of an immutable SSOT service during client record creation. Using the clinic application in the EPR use case as an illustration, when a patient first visits the clinic, the clinic application needs the patient's SSOT-ID. The patient provides personal data to the clinic. The clinic application invokes the EPR SSOT query-by-criteria operation with the patient's data. From the returned set of partial EPRs, the clinic and patient together identify the correct EPR. With the SSOT-ID from the selected partial EPR, the clinic application invokes the query-by-SSOT-ID operation to retrieve the portion of the patient's EPR that is permitted to the clinic. The clinic application then assigns a county to the patient according to the patient's home address in the EPR. With the patient's SSOT-ID and assigned county, the clinic application adds a new record to the Clinic_Patient table for the patient. Similarly, when a patient first visits the pharmacy, the pharmacy application uses the EPR SSOT query-by-criteria operation to retrieve a partial EPR of the patient. The pharmacy application gets the patient's SSOT-ID from the partial EPR. Since the pharmacy data does not depend on data in the EPR, the pharmacy application can add a new record to the Pharmacy_Patient table for the patient with the patient's SSOT-ID.

3.4 Client-record Retrieval

As with the mutable SSOT service, the immutable SSOT clients need to combine the SSOT data with the local data to compose complete data records. The data retrieval process is depicted in Fig. 2(e).

In the EPR use case, when a patient revisits the clinic or the pharmacy, the clinic and pharmacy applications use the EPR SSOT query-by-criteria operation to retrieve the patient's SSOT-ID. With the SSOT-ID, the applications fetch the permission-filtered EPR with query-by-SSOT-ID. Using the SSOT-ID again, the applications retrieve patient's local data from the local databases, i.e. the Clinic_Patient, Patient_Visit, Pharmacy_Patient and Drug_Dispensing tables in Fig. 2(b). Finally, the applications combine the EPR and local data to instantiate a complete patient record.

3.5 Constructing SSOT-IDs

Since the SSOT and the clients are loosely coupled, the clients rely on the SSOT-IDs to be unique and permanent. Therefore, it is important to select a proper data type, length, format and representation for the SSOT-ID. For example, the Canadian Social Insurance Number (SIN) has 9 digits. The first digit of the SIN represents the owner's residential status. Similarly, a SSOT-ID can have embedded representations. The design of the SSOT-ID deserves extraordinary attention to ensure uniqueness and permanency. For example, the SIN is unique, but not permanent. When an owner's residential status changes, a new SIN may be assigned. Therefore, SIN is not a good SSOT-ID candidate. In addition, public data items are not good SSOT-ID candidates.

An SSOT-ID does not need to be a single value. It can also be a composite value, as long as it is unique and permanent. Part of the SSOT-ID can be a fixed-length sequential number, to ensure its uniqueness. Since the SSOT-ID will be shared between systems, it is recommended that the SSOT-ID take a different format from the SSOT database standard, so that the SSOT database standard will not be exposed.

Clients store the SSOT-IDs in their local databases. The SSOT-IDs are guaranteed to be unique and permanent. Therefore, clients may consider using the SSOT-IDs as the primary keys in their local databases. If the SSOT-ID data type and size do not match the local database standard, we recommend that the local database create its own local primary key, and use the SSOT-ID as a foreign key. In the EPR use case, the Clinic_Patient table could have used the EPR SSOT-ID as the primary key. Since the primary key of the Clinic_Patient table will be a foreign key of the Patient_Visit table, the clinic application created its local primary key (PID in the Clinic_Patient table), to preserve data type consistency between tables.

The clients should also consider whether the sequence of the primary keys matter to the local business logic. In the EPR use case, it is likely that a portion of the EPR SSOT-ID is a sequential number. The clinic serves only a relatively small number of patients in the regional EPR system. Therefore, only a small number of the EPR SSOT-IDs will be imported into the clinic's database. If the clinic application uses the EPR SSOT-IDs as its primary key, then the primary key will have a lot of gaps in its sequence. In addition, the order of the primary keys will not represent the order in which patients' records are added to the clinic's database.

3.6 Update Subscription

Data updates for an immutable SSOT may affect clients. In the EPR use case, the clinic application assigns a patient's county based on a patient's home address. When a patient moves, the clinic application may assign a different county for the patient. In this case, data updates in the EPR SSOT affect the clinic application. On the other hand, none of the pharmacy application local data depends on the EPR SSOT data. Therefore, data updates in the EPR SSOT do not affect the pharmacy application.

The SSOT cannot determine how data updates will affect the loosely coupled clients. Clients are responsible for managing their own data. Therefore, the immutable SSOT must provide an update subscription operation. If a client is concerned about data updates in the SSOT, then the client is responsible for subscribing to the SSOT update service through the update subscription operation.

After a SSOT resource is updated, the SSOT will send an update message with the SSOT-ID of the updated resource to the subscribers. When the subscriber receives the update message, the subscriber can check whether the SSOT-ID is referenced locally. If not, the subscriber can ignore the update message. If the SSOT-ID is referenced locally, then the subscriber can fetch the resource details using the query-by-SSOT-ID operation. Based on the latest resource details, the subscriber may update its local data accordingly. The update subscription process is depicted in Fig. 2(f).

In the EPR use case, the clinic application would subscribe to the EPR SSOT update service. When an EPR is updated, the clinic application will receive an update message with the SSOT-ID of the updated EPR. The clinic application determines whether the SSOT-ID is referenced locally. If so, it retrieves the patient details from the EPR SSOT using the query-by-SSOT-ID operation. Then the clinic application can determine whether the patient's latest home address matches the clinic-assigned county in the local database. If not, it updates the local database accordingly. On the other hand, the pharmacy application is not affected by EPR updates. Therefore, the pharmacy application would not subscribe to the EPR SSOT update service. Notice that the pharmacy still relies on the SSOT for the latest EPR patient information. However, it does not subscribe for updates since it does not need to update its own local database, when EPR data changes.

3.7 Deletion Subscription

Just like any other data, the SSOT data can be deleted. Since the SSOT is loosely coupled with its clients, clients cannot put a foreign key constraint on the SSOT to restrain the SSOT from deleting data. Instead, the SSOT provides a deletion service. When a resource is deleted from the SSOT, the SSOT will send a deletion message to the subscriber. Clients need to determine whether they should subscribe to the SSOT deletion service using the deletion subscription operation.

In the EPR use case, deleting a patient from the EPR SSOT may create broken links in the Clinic_Patient and Pharmacy_Patient tables. Therefore, the clinic and pharmacy applications should both subscribe to the EPR SSOT deletion service.

If the SSOT physically deletes the resource, then the deletion message should contain the last version of the resource before the deletion and the reason for deletion. Clients can use this information to determine how to handle the deleted data.

In the EPR use case, a patient may move from the region and be deleted from the EPR SSOT. The clinic and pharmacy applications will receive a deletion message from the EPR SSOT with the last version of the patient's EPR. The applications may store or ignore the EPR in the deletion message. Depending on the reason for deletion, the applications can delete, archive or mark the patient's local record inactive.

Alternatively, the SSOT may logically delete a resource. In this situation, the deletion message will only contain the resource SSOT-ID and the reason for deletion. The client can still fetch the logically deleted resource using the query-by-SSOT-ID operations. The query-by-SSOT-ID operation would return the corresponding resource but flagged as deleted. The query-by-criteria operation could exclude logically deleted resources from the response correlated set, or provide them and flag them as deleted.

In the EPR use case, if patients are only logically deleted from the EPR SSOT, the clinic and pharmacy applications may mark the patient inactive in their local databases. If EPR records are only logically deleted, then clients should always check the deleted flags on the EPR, since logically deleted records may later be undeleted.

3.8 Additional Operations

An SSOT creation subscription operation is not recommended. If clients can subscribe to SSOT data creation, then clients can replicate the whole SSOT database, which violates the purpose of using SSOT. Clients should only link to the SSOT resources related to their operational mandates, but not replicate the SSOT data.

In addition to the four mandatory operations, an immutable SSOT service might provide additional resource create, retrieve, update and delete (CRUD) operations.

In the EPR use case, if a patient has not registered with the regional EPR SSOT, then the clinic and pharmacy applications would not find the patient through the query-by-criteria operation. If the EPR SSOT also provides a patient creation operation, then the authorized clinic or pharmacy personnel can create SSOT EPRs as needed.

If the clinic or pharmacy personnel are not authorized to create SSOT EPRs, then the un-registered patients need to register with the EPR authority later. In the meantime, the applications can create a local temporary file to store the patient's data. An optional column (TEMP FILE#) can be added to the Clinic_Patient and Pharmacy_Patient tables to keep track of the temporary file number. In the absence of the SSOT ID and existence of the TEMP FILE#, the applications will not query the EPR SSOT for patient's information, but retrieve data from the local temporary file. After the patient registers with the EPR SSOT, the applications can insert the EPR SSOT-ID into the Clinic_Patient and Pharmacy_Patient tables, and delete the temporary file. At this point, the application returns to normal processing.

4 Performance and Quality-of-Service (QoS)

Despite the data-synchronization challenges, the data-replication model has a performance advantage over the SSOT service model. In the SSOT service model, client applications make extra service invocations to the SSOT during record creation and retrieval, which may affect user experience. Therefore, we implemented experiments to evaluate how the extra SSOT service invocations might affect user wait times.

The experiments evaluated delay caused by the SSOT service invocations. They were conducted on the institution's network supporting ~40,000 students. The SSOT service was hosted on a workstation in a departmental subnet. In the experiments, client applications accessed the SSOT service through Wi-Fi on the institute's public network during regular office hours. This condition simulated an enterprise network supporting multiple sub-divisions.

The experiment results show that if the SSOT service and the client applications are located within the same enterprise network, then the service invocation costs less than 20 milliseconds per call. This indicates that users should not notice any performance deterioration. If the SSOT service is available across a wide area network, the performance primarily depends on the transmission delay between the public SSOT service and the clients. Clients should benchmark the transmission delay to determine the actual performance effect.

In dynamic web service composition [3], clients select web service providers according to their published quality-of-service (QoS) [4]. Even though, SSOT's clients will likely access the SSOT service statically, the SSOT service should publish the following QoS metrics per operation:

- **Operational hours:** the regular SSOT servicing hours.
- **Maintenance schedule:** the changes and release schedule.
- **Reliability:** the SSOT's ability to perform the operation without errors.
- **Request-processing time:** the maximum and average time required for the SSOT service to complete the operations.

The SSOT may publish additional QoS metrics, such as capacity, performance, robustness, accuracy and more [3]. Clients can design their usage of the SSOT service according to the SSOT's QoS metrics.

If the SSOT service is part of a larger enterprise or jurisdiction, then the SSOT service usually has the same operational hours and maintenance schedules as their clients. Public SSOT services, like Canada Post, are usually available 24x7.

5 Related Work

Most SSOT is implemented on the data layer. Ives et al. [5] suggest synchronizing distributed data on the data layer. Ives et al. propose a "Collaborative Data Sharing System (CDSS) [that] models the exchange of data among sites as update propagation among peers, which is subject to transformation (schema mapping), filtering (based on policies about source authority), and local revision or replacement of data." Since data across multiple sites are continuously "updated, cleaned and annotated", cross-site synchronization has to deal with issues such as data correctness, schema and terminology consistence, and timing. These data layer synchronization hurdles highlight the advantage of our SSOT service model that eliminates data layer synchronization.

Others try to implement SSOT using an Enterprise Service Bus (ESB) [6], in which a SSOT is defined. Clients duplicate the SSOT data locally, and subscribe to ESB for SSOT updates. Whenever the SSOT is updated, clients synchronize with the SSOT by repeating the changes in their local copies. Our SSOT model totally avoids data duplication at the clients' site.

Instead of a data-centric model for SSOT, some research has turned to artifact-centric modeling [7]. An artifact is a set of name-value-pairs related to a business process or task, where data represents business objects. In the artifact-centric model, each artifact instance is shared between all process participants. The participants get information from the artifact and change the state of the artifact to accomplish the process goal. Since the artifacts are shared between process participants, access and transaction control is necessary. Hull [8] suggests using artifact-centric hubs to facilitate communication and synchronization between the participants. Our SSOT service model does not require complicated facilitation or a centralized hub.

Other researchers have proposed the Personal Information Management (PIM) [9] model. In our model, the SSOT does not have any knowledge about its clients' data. However, the PIM model finds, links, groups and manages clients' data references to the source. PIM is a centralized data management model, while SSOT is a distributed data management model.

Finally, Ludwig et al. [10] propose a decentralized approach to manage distributed service configurations. The proposed solution uses RESTful services to exchange configuration data between hosts, and a subscription mechanism to manage changes. This approach endorses a data perspective similar to an SSOT service, where each data source maintains self-contained autonomous data. Data is not synchronized across multiple sites. Sources and clients are statically bound.

6 Future Work

A mutable SSOT service has one standard operation: query-by-criteria. An immutable SSOT service has four standard operations: query-by-criteria, query-by-SSOT-ID, update subscription, and deletion subscription. Based on these standard operations, we defined Web Service Definition Language (WSDL) extensions for the mu-table and immutable SSOT services with corresponding templates. Because of the limited space, we do not include the SSOT WSDL and templates in this paper.

Based on the WSDL extensions and templates, development tools can easily be created to generate SSOT service source code with corresponding client access code. These tools can simplify development for programmers, which encourages the use of SSOT services to replace data-replication. Eclipse is an excellent platform to implement such tools. There are three additional future work topics.

Maintenance and Upgrades: Every active service goes through changes. Besides regular change management, services may experience unexpected emergency incidents. When these incidents occur, the clients should be alerted about their occurrences, side-effects, recovery progress and estimated recovery times. As suggested by Ludrig et al. [10] a subscription service can be used to communicate change, maintenance and emergency notices. Protocols and language extensions can be defined for various types of change, maintenance and emergency activities. Since SSOT introduces a single point of failure, a cloud infrastructure specifically designed for SSOT can improve its availability.

Local Temporary Cache: If the cost of the SSOT service invocation is a concern, clients can cache the SSOT resources temporarily. Once a SSOT resource is obtained, there is a good chance that the client needs the same SSOT resource for related processes. Therefore, temporarily caching the SSOT resource will likely reduce service invocations. The cached SSOT resource can be flushed after a timeout period. The SSOT update or deletion message should also flush the related resource from the cache. The caching functionality would ideally be implemented as middleware that supports the SSOT service architecture.

Schema Synchronization: For existing applications to adopt a SSOT service model, the existing client applications need to map their local data to the SSOT data. Both the SSOT and the clients can make use of existing ontology studies, which define data syntax and semantics for specific industries. For example, HL7 [11] is defined for the health industry; RosettaNet [12] is defined for e-business; EDIFACT [13] is defined for electronic data interchange. Moving toward standard languages will benefit the survival and the long term growth of the industry. The SSOT service model does not define the communication architecture or protocol between the clients (e.g. between the clinic application and the pharmacy application in the EPR use case), but adopting the SSOT service model can simplify communication between the clients. For example, the pharmacy can verify a patient's prescription with the clinic by the patient SSOT-ID and avoid multiple drug dispensing.

7 Conclusion

The SSOT service model described in this paper addresses the data-synchronization problems that arise due to data-layer replication across distributed systems. On the other hand, the SSOT service model introduces a single point of failure in the system. Depending on the Service Level Agreement (SLA), the SSOT may need support from multi-site configurations or cloud-infrastructure with fail-over capability. Although the data-replication model does not have a single point of failure, it suffers from data-synchronization and data-inconsistency issues. Data synchronization usually involves defining a custom peer-to-peer data exchange agreement. The custom agreement tightly couples the data provider and consumer, which makes switching providers very costly. Nonetheless, data synchronization usually happens during the overnight maintenance windows. Data becomes stale between synchronizations. Furthermore, data replication keeps a full copy of the provider's data at the clients' sites. If clients use only a small portion of the provider's data, then the clients are wasting resources. An added benefit of the SSOT service model is that it can control what data each client is authorized to access, while data replication makes all data available to clients.

The SSOT service model allows the provider and clients to be loosely coupled. Clients do not need to pledge infrastructure resources for the foreign data. The SSOT service model also provides up-to-date data. Overall, we believe that the SSOT service model can be used to eliminate data replication, enforce data autonomy, advocate data self-containment, ease data maintenance and enhance data protection. In the long term, these properties will also increase business adaptability.

Within large enterprises or government agencies, managing large amounts of data as a single entity is problematic. Decomposing a large data set into smaller autonomous and independently managed data sets can increase flexibility. As in the EPR case, once the EPR SSOT service is established, a new patient related service can be created without defining and creating its own patient data set. The new service does not need to negotiate with other parties regarding data acquisition or synchronization. The new service can loosely couple with the EPR SSOT and be established quickly. In addition, the SSOT service model allows each individual service to be self-contained and maintain its local database. For example, the clinic application and the pharmacy application in the EPR use case maintain their individual local databases without sharing data with the EPR system. This characteristic is very important in the health industry, where patients' privacy is closely monitored.

The SSOT service model is also applicable to the financial industry. Banking, investment and insurance businesses are often integrated under one corporation. However, legislation may require each of these businesses to be separate entities. The SSOT service model allows the corporation to create a customer SSOT to register each customer once. Banking, investment and insurance services can run as separate entities, while being loosely coupled with the customer SSOT service. With the SSOT service model, new financial services can be introduced more quickly. Similarly, the SSOT service model can benefit any jurisdiction that provides multiple services.

References

1. Lasila, O., Swick, R.R.: World Wide and Web Consortium: Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation (1998)
2. Fielding, R.T.: Chapter 5 Representational State Transfer (REST), Architectural Styles and the Design of Network-based Software Architectures, Doctoral dissertation, University of California, Irvine (2000)
3. Dustdar, S., Schreiner, W.: Survey on Web services Composition. International Journal on Web and Grid Services 1, 1–30 (2005)
4. Ran, S.: A Model for Web Services Discovery With QoS. ACM SIGecom Exchanges 4(1), 1–10 (2003)
5. Ives, Z., Khandelwal, N., Kapur, A., Cakir, M.: ORCHESTRA: Rapid, Collaborative Sharing of Dynamic Data. In: The 2nd Biennial Conference on Innovative Data Systems Research (CIDR 2005), Asilomar, CA, USA (2005)
6. Schmidt, M.-T., Hutchison, B., Lambros, P., Phippen, R.: The Enterprise Service Bus: Making servie-oriented architecture real. IBM Systems Journal 44(4), 781–797 (2005)
7. Nigam, A., Caswell, N.: Business artifacts: An approach to operational specification. IBM Systems Journal 47(3), 428–445 (2003)
8. Hull, R.: Artifact-Centric Business Process Models: Brief Survey of Research Results and Challenges. In: OTM 2008, Monterrey, Mexico (2008)
9. Jones, W.: Personal Information Management. Annual Review of Information Science and Technology 41(1), 453–504 (2007)
10. Ludwig, H., Laredo, J., Bhattacharya, K., Pasquale, L., Wassermann, B.: REST-Based Management of Loosely Coupled Services. In: The 18th International Conference on World Wide Web (WWW 2009), Madrid, Spain (2009)
11. HL7 Health Level Seven International, `http://www.hl7.org`
12. RosettaNet (1999), `http://www.rosettanet.org`
13. EDIFACT, United Nations Directories for Electronic Data Interchange for Administration, Commerce and Transport,
`http://www.unece.org/trade/untdid/welcome.htm`