Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 506272, 9 pages
http://dx.doi.org/10.1155/2015/506272

Research Article

Modeling and Querying Business Data with Artifact Lifecycle

Danfeng Zhao,[1] Wei Zhao,[2] Le Sun,[1] and Dongmei Huang[1]

[1] *College of Information, Shanghai Ocean University, Shanghai 201306, China*
[2] *Information & Telecommunication Branch, Heilongjiang Electric Power Company Ltd., State Grid Corporation of China, Heilongjiang 150090, China*

Correspondence should be addressed to Danfeng Zhao; dfzhao@shou.edu.cn and Dongmei Huang; dmhuang@shou.edu.cn

Received 28 August 2014; Accepted 11 September 2014

Academic Editor: L. W. Zhang

Business data has been one of the current and future research frontiers, with such big data characteristics as high-volume, high-velocity, high-privacy, and so forth. Most corporations view their business data as a valuable asset and make efforts on the development and optimal utilization on these data. Unfortunately, data management technology at present has been lagging behind the requirements of business big data era. Based on previous business process knowledge, a lifecycle of business data is modeled to achieve consistent description between the data and processes. On this basis, a business data partition method based on user interest is proposed which aims to get minimum number of interferential tuples. Then, to balance data privacy and data transmission cost, our strategy is to explore techniques to execute SQL queries over encrypted business data, split the computations of queries across the server and the client, and optimize the queries with syntax tree. Finally, an instance is provided to verify the usefulness and availability of the proposed method.

1. Introduction

With the advent of Big Data, attentions from all walks of life gradually focus on exploiting their controllable data so as to realize a satisfactory profit. Against this background, data resource is widely recognized to be equal in status and value to mineral resource. In enterprise-led dataspace, data generated in business process are the most significant factor which will affect the performance of process execution. As business process is closely related to enterprise's business strategy and market competiveness, researches on business data will benefit enterprises in coping with the challenges brought by Big Data and are significant in predicting and responding to potential business risks in a timely way as well as offering business opportunities. Recently, research work on data management in business process has gradually become a research hotspot.

During business process execution, there is usually a large data transfer, which falls into the scope of Big Data. For example, currently China Unicom monthly stores more than 2 trillion records, data volume is over 525 TB, and the highest data volume has reached a peak of 5 PB [1]. China UnionPay daily handles more than 60 billion transactions; thereby the generated data are exceptionally large. Google supports such a great many of services as both processing over 20 petabytes (10^{15} bytes) of data and monitoring 7.2 billion pages per day [2]. Starting from 2005, NTDB (National Trauma Data Bank) has tracked more than half a million trauma patients by now and stored their records, and many service retailers collect data from multiple sales channels, catalogs, stores, and online interaction, such as Client-Side Click-to-Action [3]. Hence, Big Data is ubiquitous (business process data arises in enterprises (large or small)) and grows exponentially, which poses huge challenges in data management. To address this, the first priority is to build an adaptive data model, which provides basis and direction for efficient data acquisition. Secondly, we see it as the next big issue about devising a suitable query strategy for business data which is a prerequisite for data processing and analysis.

Data modeling is the foundation for dataspace building. The research work in the early days focused on dataspace modeling where its subject is individual [4, 5]. iDM (iMeMex data model) [6] is the first model which is able to represent all heterogeneous personal information into a single model. This data model uses database approach so easy to understand but introduce a new query language iQL, which is a little hard

for normal users to learn. UDM (unified data model) [7] uses the integrated IR-DB approach, which is able to represent the partial sections of a file but is also not able to support relational data query. Triple model [8] represents heterogeneous data in triple form, which is a simple and flexible solution but does not support the path expression queries, uncertainty, and lineage queries. PDM (probabilistic semantic model) [9] supports top-k query answering but it is difficult to obtain reliable probability functions. The methods above are based on personal dataspace. Unfortunately, in enterprise-led data space scenarios today there is rare research works on data modeling.

Query ability is the basis of the exploitation of Big Data's value. Query language iQL [6] realizes rules-based query optimization but ignores the evaluation of optimization cost. UDM [7] introduces a new query language, which is based on SQL query language with some extended core operations, called TALZBRA operation. Triple model [8] supports subject predicate object (SPO) query language that can be enhanced by RDF-based query language. DSSP (dataspace support platforms) supports some useful services on dataspace, helps to recognize the correlation among sources of dataspace, and provides a basic query schema upon these data sources. In enterprise-led dataspace, business process data is the key element in data modeling, which has such characteristics as large-volume, strong temporal correlation and stable lifecycle. These characteristics make it an extreme challenge for current query schemes.

Business data realistically records the whole execution process of a single task, including execution status, resource status and real-time usage, and correlation with other business process instances. Executing a business process would generate additional data for a variety of reasons such as monitoring for performance or business concerns, auditing, and compliance checking. Even business process schemas and enactments can be viewed as data so that they can be managed, queried, mined for process schemas, and analyzed [10]. An artifact is a kind of widely recognized business process data, representing key business entities. Artifact-centric approach [11] is the representative method in data-centric business process management and has been applied in various client engagements, including financial [12], supply chain, retailer [13], bank, pharmaceutical research [14], and cooperative work [15]. In this paper we firstly adopt artifact as a basic element, analyze its evolution process, and then model business data through corresponding artifact lifecycle. Secondly, we make efforts on devising a safe and quick query strategy in consideration of the privacy and storage distribution of artifacts.

The rest of the paper is organized as follows. In Section 2 we introduce the concept of artifact in field of workflow management and model business data with its lifecycle from the perspective of process. In Section 3 we propose a business data partition method concerning user interest, based on which we further present a cryptograph query for off-site storage data. Then, we give a detailed instance to verify the proposed method in Section 4. In the last section, we draw a conclusion.

2. Business Data Modeling

As before, artifacts describe the business-relevant data and their lifecycles which is an important property of business data and describes the whole dynamic process of business data. It also contains specific time information. To take advantage of these characteristics, our strategy is to model business data with its lifecycle, which aims to realize the completed description of dynamic business data. In this section, we introduce artifact-relevant notions and take artifact-centric process description method to model business data with artifact lifecycle. Furthermore, we adopt business process logic model to illustrate the lifecycle of business data and then measure the quality of above model.

2.1. Basic Definition

Definition 1. Artifact [16] is an objective data entity which records the business process. Artifact comprises both a unique immutable identity and self-describing mutable content.

Definition 2. An artifact lifecycle captures the end-to-end process of a specific artifact, from creation to completion and archiving.

Definition 3. Artiflow model (artifact logical flow) [17] is 5-tuple (N, S, R, C, Ru), where N is the name of model, S is a finite set of services, R is a finite set of repositories, C is a finite set of transport channels, and Ru is a finite set of business rules.

Definition 4. The states of artifact are a set, $\wedge_{i=1}^{m} IsDefine(A_i)$ (conjunction expression), where $IsDefine(A_i)$ is a mapping function that assigns a Boolean value $\{0, 1\}$ to each single attribute A_i in attribute set A, $A_i \in A$, $i \in \{1, m\}$, and m is the number of attributes in artifact. If the attribute is defined and has value, it will return 1; else it will return 0.

Definition 5. Service is 5-tuple (n, V_r, V_w, P, E), where n is the name of a certain service, $n \in S$; V_r, V_w are the finite set of artifact classes, where V_r is a set of artifacts which the service is about to read and V_w is a set of artifacts which the service is about to rewrite; P is the description of artifact states inputted by V; E is the description of activities on V.

Definition 6. Repository is 4-tuple $R = (\text{re}, R_a, R_r, C_t)$, where re is the name of repository; R_a, R_r are the set of stored and read artifacts, respectively; C_t is the reading condition for R_r.

Definition 7. Transport channel is 2-tuple (Cn, Cs) where Cn is the name of the channel; Cs is 3-tuple (prior service/repository name, rear service/repository name, channel type). Cs $\in$ $R \times S \times \{\text{Read, ReadOnly}\} \cup S \times R \times \{\text{Write}\}$, where R, S are the finite set of repository elements and service elements in Artiflow, respectively, and the set of transport channel types is described as {Read, ReadOnly, Write}.

2.2. Data Modeling with Artifact Lifecycle. As suggested in Definitions 2 and 3, Artiflow is a logical model that records

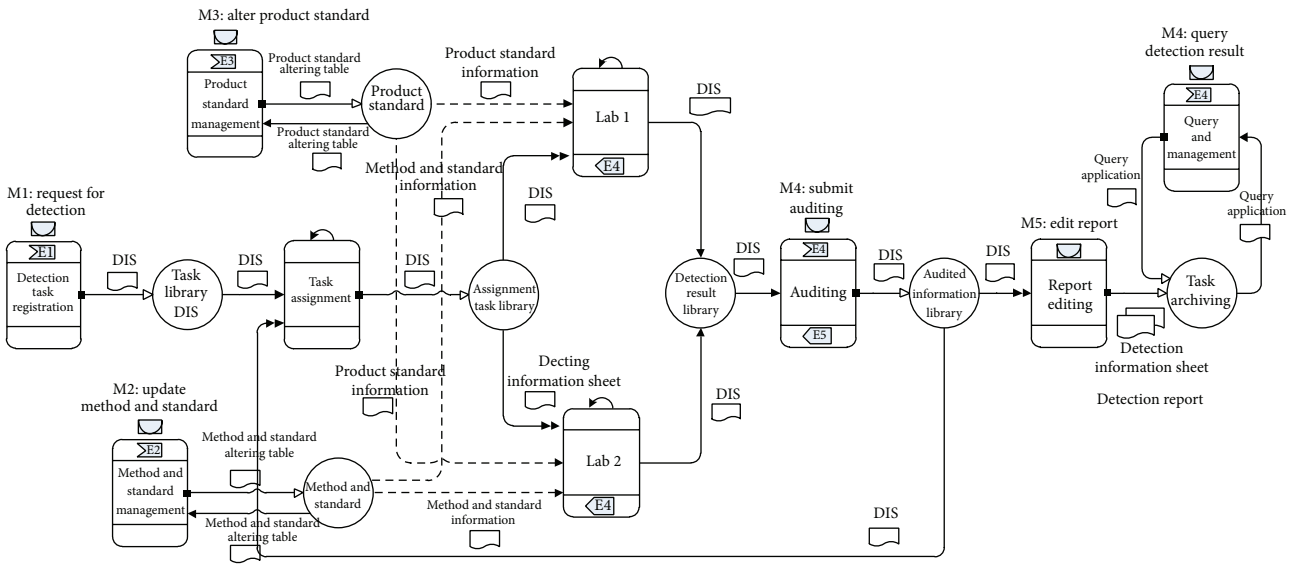

Figure 1: Lifecycle of business data in "monitoring information."

the artifact lifecycle, in which elements of repository, service, artifact type, and transport channel are abstracted to represent a realistic business process. Artiflow views business process as a graph, where nodes are either "service" or "repository." We formalize Artiflow to facilitate data analysis and illustrate it to facilitate process analysis. Figure 1 illustrates a quality inspection process instance of a certain enterprise where the main artifact is the "monitoring information sheet." The artifact captures the detected product's evolvement from creation to archiving, which includes all the business-relevant data in this process. The whole process comprises detection task registration, task assignment, task inspection, task audition, and so forth. Note that artifact "monitoring information sheet" is inseparable from the coordinate with such other artifacts as "product standard" and "method & standard" within its lifecycle. When "monitoring information sheet" completes its lifecycle, it will serve as a reference to form a new artifact—"detection information sheet (DIS, for short)."

In this figure, there are nine services ("task assignment," "auditing," etc.), seven repositories ("assignment task library," etc.), and serial transport channels between these repositories and services.

2.3. Model Quality Evaluation. Exactly, one business object can be achieved by implementing different business processes, while different business process corresponds to a different Artiflow model. However, we will measure the Artiflow based on two factors: (1) the number of services determines the flexibility of model. (2) The repository services read and update artifacts. It is in this context that we define following theorem to measure the quality of artifact models.

Theorem 8. *Given an Artiflow (N, S, R, C, Ru), it has j Artifacts where the number of attributes in any $Artifact_i$ is n_i. Suppose $|S^i|$ and $|R^i|$ represent the service amount and repository amount of corresponding $Artifact_i$, respectively; formula (1) is defined to calculate Artiflow's web service granularity and repository service proportion, so as to measure the quality of models:*

$$\pi = \frac{\sum_{i=1}^{j} \rho_i \left(\alpha \left(\left|S^i\right| / n_i\right) + \beta \left(1 - \left|R^i\right| / \left(\left|S^i\right| + \left|R^i\right|\right)\right)\right)}{\sum_{i=1}^{i} \alpha \left(\left|S^i\right| / n_i\right) + \beta \left(1 - \left|R^i\right| / \left(\left|S^i\right| + \left|R^i\right|\right)\right)}, \quad (1)$$

where α, β, and ρ_i are known.

Theorem Proving. For a given Artiflow (N, S, R, C, Ru), each Artifact_i comprises both a service sequence and a repository sequence, marked as $(S_x, R_u, \ldots, S_y, R_v)$, where service sequence is described as $S^i = (S_x, \ldots, S_y)$ and repository sequence is described as $R^i = (R_u, \ldots, R_v)$. Each Artifact_i also contains m attributes.

Suppose $|S^i|$ and $|R^i|$ represent the service amount and repository amount, respectively, then $|S^i|/m$ represents the granularity of services when dividing the whole lifecycle of artifact by its attribute number m. A larger value indicates there are more blocks that are divided and the granularity is less, which contributes to building a more flexible model.

In Artiflow, normally each artifact has a following repository to store its intermediate state, but there is exception that some services can directly communicate with each other and do not need intermediate repositories. Therefore, for the same Artifact, the few the repository elements are, the less the redundancy would be. $|R^i|/(|S^i| + |R^i|)$ represents the proportion of repository elements in both service and repository elements within its corresponding artifact lifecycle. The shorter the value is, the better the designed lifecycle would be.

The quality of Artifact_i is computed by the following formula:

$$\pi = \alpha \left(\frac{\left|S^i\right|}{n_i}\right) + \beta \left(1 - \frac{\left|R^i\right|}{\left|S^i\right| + \left|R^i\right|}\right), \quad (2)$$

where α and β are predefined constants, which is used to balance the different magnitude between values both before and after the plus.

Each Artiflow comprises multiple Artifacts, so the quality measurement formula for the whole Artiflow is $\pi = \sum_{i-1}^{j} \rho_i \pi_i / \sum_{i-1}^{j} \pi_i$, where j is the number of artifacts and $\sum_{i-1}^{j} \rho_i = 1$; ρ_i represents the importance of Artifact$_i$. The optimization of key Artifacts has a great impact on the whole model to great extent, while the optimization of less-valuable artifact does not contribute too much to the model efficiency. Note that ρ_i can be either given by user or obtained by data analysis.

By integrating with both repository element redundancy and service element granularity,

$$\begin{aligned} \pi &= \frac{\sum_{i=1}^{j} \rho_i \pi_i}{\sum_{i=1}^{j} \pi_i} \\ &= \frac{\sum_{i=1}^{j} \rho_i \left(\alpha \left(\left|S^i\right| / n_i\right) + \beta \left(1 - \left|R^i\right| / \left(\left|S^i\right| + \left|R^i\right|\right)\right)\right)}{\sum_{i=1}^{i} \alpha \left(\left|S^i\right| / n_i\right) + \beta \left(1 - \left|R^i\right| / \left(\left|S^i\right| + \left|R^i\right|\right)\right)} \end{aligned} \tag{3}$$

can be deduced and taken to measure the model quality.

3. Business Data Querying

Enterprises like Google, Amazon have provided plenty of cloud services, which provide an open storage solution for data like process data all over the world. But off-site storage is unsafe due to data privacy, even public cloud. In this case, these data need to be encrypted and then stored in database. But it is hard to make a trade-off between data security and query speed, which is because process data need to be frequent queried, modified, and transmitted. In this section we make study on partitioning encrypted artifacts and coming up with a superior query plan for cryptograph query that minimizes the execution cost.

3.1. Business Data Partition. In order to ensure the efficiency of business process, a superior data partition is on-demand. When using Bucket partition method, query result on cryptograph is actually a superset of true results generated by relevant operators and then filtered at the client after decryption. Thus, superior partition method is of great help and aims to minimize the work done as much as possible, such as minimizing the number of interferential results.

3.1.1. Data Analysis

Definition 9 (Bucket [18]). Mapping the domain of attribute A into another partitions set $\{p_1, \ldots, p_M\}$, where $p_i \cup p_j = \emptyset$, $1 \leqslant i, j \leqslant M$, each partition p_i is named as a Bucket; M is the Bucket number.

Definition 10 (the user interest on artifact). Querying on Artifact's attribute A of n times, respectively, while $q(a_i)$ represents any single result of queries q that contains value a_i, suppose $f(q(a_i))$ is the frequency of $q(a_i)$ occurring in n trials, as n increases, the frequency stabilizes at a certain value, which is expressed as $p(q(a_i))$. In other words, $p(q(a_i)$ is the probability of artifact attribute a_i emerged in query result dataset, called user interest.

Definition 11 (interferential artifact). (Intf-Artifact) is an artifact which is incorrect result but belong to cryptograph query result $q^*(a_i)$, named as INTFA$(q^*(a_i))$.

3.1.2. Min-Interference Partition. All Artifacts in each Bucket correspond to a given index number in Bucket-based cryptograph partition. Cryptograph query returns all the encrypted Artifacts in Bucket where true result exists. The rest in Bucket would be transmitted to users as Intf-Artifact, and then it should be deciphered and further filtered. Hence, Bucket partition method determines the number of Intf-Artifacts, which further effects the query processing cost.

Suppose a cryptograph relation contains n tuples {Artifact$_1$, Artifact$_2$, ..., Artifact$_n$} and k is a large integer; then we pose k random queries. Totally, there are k_i queries where their final query results are Artifact$_i$, and other l_i tuples are returned as the provisional result. In this case the expectation of Intf-Artifact is $l_i * (k_i/k)$.

There are n tuples in the relation at all, and then the expectation of total Intf-Artifacts is

$$l_1 * \left(\frac{k_1}{k}\right) + l_2 * \left(\frac{k_2}{k}\right) + \cdots + l_n * \left(\frac{k_n}{k}\right). \tag{4}$$

As for each Bucket containing n different attribute values, its user interest is $p(q(B)) = p(q(a_1)) + p(q(a_2)) + \cdots + p(q(a_n))$.

If the user interest on ith artifact in a given Bucket is $p(q(a_i)/p(q(B))$, $(1 \leq i \leq n)$, then the number of Intf-Artifacts brought by above query is $|\text{INTFA}(q * (a_i))| = f_1 + f_2 + \cdots + f_{i-1} + f_{i+1} + \cdots + f_n$.

As for Bucket j $(1 \leq j \leq k)$, based on the user interest on artifact and the number of Intf-Artifacts in each Bucket, we can describe Bucket Intf-Artifact as follows:

$$\begin{aligned} &\text{WINTFA} \\ &= \sum_{j=1}^{M} \left[\!\left[p\left(q\left(B_j\right)\right)\right]\!\right] * \left|\text{INTFA}\left(\text{Bucket}_j\right)\right| \\ &= \sum_{j=1}^{M} \left[p\left(q\left(B_j\right)\right) * \sum_{i=1}^{n} \left(\frac{p\left(q\left(a_i\right)\right)}{p\left(q\left(B_j\right)\right)} \left|\text{INTFA}\left(q^*\left(a_i\right)\right)\right| \right) \right] \\ &= \sum_{j=1}^{M} \left[p\left(q\left(B_j\right)\right) * \sum_{i=1}^{n} \left(\frac{p\left(q\left(a_i\right)\right)}{p\left(q\left(B_j\right)\right)} \left(\sum_{i=1}^{n} f_i^j - f_i^j \right) \right) \right] \\ &= \sum_{j=1}^{M} \left[\sum_{i=1}^{n} p\left(q\left(a_i\right)\right) \sum_{i=1}^{n} f_i^j - \sum_{i=1}^{n} p\left(q\left(a_i\right)\right) f_i^j \right] \\ &= \sum_{j=1}^{M} \left[\sum_{i=1}^{n} p\left(q\left(a_i\right)\right) F_j - \sum_{i=1}^{n} p\left(q\left(a_i\right)\right) f_i^j \right]. \end{aligned} \tag{5}$$

From here we see that in the case of a fixed Bucket number, the smaller the value of formula (5) is, the more excellent the index would be. A larger value brings a heavy cost when querying and renders a low efficiency of Bucket partition. From the probability angle, Bucket where artifact with higher user interest exists should contain fewer Artifacts. Therefore, user interest on each artifact should be viewed as the weight

in the whole process. Moreover, when the index is being built, formula (5) is used to determine which Bucket we store each artifact in, which helps to obtain an optimal partition result.

3.2. Business Data Query. Cloud service stores encrypted artifact information and corresponding index information, while such other information as the partitioning of attributes, mapping function, and so forth are stored at client. When a user issues a query request, query q should be rewritten to its server-side cryptograph query q^*, which is then executed on cloud. The purpose of rewriting SQL queries is to split the query computation across the client and cloud.

3.2.1. Basic Definitions

Definition 12. $\xi_{(*,a]}(x)$ is a function which returns a set of all the Bucket ID where its right boundary value B_*^j.right is not greater than x when once partitioning Bucket; that is, $\xi_{(*,a]}(x) = \{\mathrm{BID}_*^v \mid B_*^v.\mathrm{right} \le x\}$.

Definition 13. $\xi_{[a,*)}(x)$ is a function which returns a set of all the Bucket ID where its left boundary value B_*^j.left is greater than x when once partitioning Bucket; that is, $\xi_{[a,*)}(x) = \{\mathrm{BID}_*^v \mid B_*^v.\mathrm{left} \ge x\}$.

Definition 14. $\xi_{(*,p(q(v_i))]}(x)$ is a function which returns a set of all the Bucket ID where its maximum artifact query probability $B_j^*.p\mathrm{right}$ is not greater than x when twice partitioning Bucket; that is, $\xi_{(*,p(q(v_i))]}(x) = \{\mathrm{BID}_j^* \mid B_j^*.p\mathrm{right} \le x\}$.

Definition 15. $\xi_{[p(q(a_i)),*)}(x)$ is a function which returns a set of all the Bucket ID where its minimum artifact query probability $B_j^*.p\mathrm{left}$ is not less than x when twice partitioning Bucket; that is, $\xi_{[p(q(a_i)),*)}(x) = \{\mathrm{BID}_j^* \mid B_j^*.p\mathrm{left} \ge x\}$.

Definition 16. $\delta_{\mathrm{cond}}(C)$ is a function that translates specific query conditions to encrypted ones.

Definition 17. Query rewriting function is described as $\delta_{\mathrm{query}}(q) \Rightarrow q^*$, where q is the original query and q^* is the cryptograph query.

3.2.2. Query Rewriting Rules. In view of grammatical rules, query condition cond includes: v, $A : A$, $\mathrm{cond}_1 \vee \mathrm{cond}_2$, $\mathrm{cond}_1 \wedge \mathrm{cond}_2$, where ":" is the operator, such as equal, less than, not greater than, greater than, and not less than. We list the rewrite formulas for various query conditions as shown in Formulas (6) to (8).

(1) $A : v$:

$$\begin{aligned}
&\delta_{\mathrm{cond}}(x = e) \Longrightarrow^1 A^* = \xi_e(x)\\
&\delta_{\mathrm{cond}}(x < e) \Longrightarrow^2 A^* \le \xi_e(x)\\
&\delta_{\mathrm{cond}}(x < e) \Longrightarrow^3 A^* \in \xi_{(*,e]}(x)\\
&\delta_{\mathrm{cond}}(x > e) \Longrightarrow^4 A^* \ge \xi_e(x)\\
&\delta_{\mathrm{cond}}(x > e) \Longrightarrow^5 A^* \in \xi_{[e,*)}(x),
\end{aligned} \tag{6}$$

where both Map 2 and Map 4 are order preserving, and both Map3 and Map 5 are random.

(2) $A : A$:

$$\begin{aligned}
&\delta_{\mathrm{cond}}(A_i < A_j)\\
&\quad \Longrightarrow^1 \vee \left(A_i^* = \mathrm{Bid}_{A_i}(p_k) \wedge A_j^* \ge \xi_{A_i}(p.\mathrm{left})\right)\\
&\delta_{\mathrm{cond}}(A_i < A_j)\\
&\quad \Longrightarrow^2 \vee \left(A_j^* = \mathrm{Bid}_{A_j}(p_l) \wedge A_i^* \ge \xi_{A_j}(p.\mathrm{right})\right)\\
&\delta_{\mathrm{cond}}(A_i < A_j) \Longrightarrow^3 \vee \left(\xi_{A_i}(p_k.\mathrm{left}) \le \xi_{A_j}(p_l.\mathrm{right})\right)\\
&\delta_{\mathrm{cond}}(A_i < A_j)\\
&\quad \Longrightarrow^4 \vee \left(A_i^* = \mathrm{Bid}_{A_i}(p_k) \wedge A_j^* = \mathrm{Bid}_{A_j}(p_l)\right),
\end{aligned} \tag{7}$$

where $p_k \in \mathrm{partition}(A_i)$, $p_l \in \mathrm{partition}(A_j)$, and $p_l.\mathrm{high} \ge p_k.\mathrm{low}$.

When the condition is $A_i < A_j$, in Map 1 A_i is order preserving, while in Map 3 both A_i and A_j are order preserving. Meanwhile, in Map 2 A_j is order preserving, and in Map 4 both A_i and A_j are random.

(3) $\mathrm{cond}_1 \vee / \wedge \mathrm{cond}_2$:

$$\begin{aligned}
&\delta_{\mathrm{cond}}(\mathrm{cond}_1 \vee \mathrm{cond}_2)\\
&\quad \Longrightarrow \delta_{\mathrm{cond}}(\mathrm{cond}_1) \vee \delta_{\mathrm{cond}}(\mathrm{cond}_2),\\
&\delta_{\mathrm{cond}}(\mathrm{cond}_1 \wedge \mathrm{cond}_2)\\
&\quad \Longrightarrow \delta_{\mathrm{cond}}(\mathrm{cond}_1) \wedge \delta_{\mathrm{cond}}(\mathrm{cond}_2).
\end{aligned} \tag{8}$$

For instance, suppose there are two artifact plaintext tables in cloud database, which are app (aid, aname, time, content, cid) check (cid, aid, result), respectively, where the range of attribute *aid* is divided into 6 partitions, including $\mathrm{id}_{\mathrm{app.aid}}([0, 100]) = 3$; $\mathrm{id}_{\mathrm{app.aid}}((100, 200]) = 7$; $\mathrm{id}_{\mathrm{app.aid}}((200, 300]) = 5$; $\mathrm{id}_{\mathrm{app.aid}}((300, 400]) = 1$; $\mathrm{id}_{\mathrm{check.aid}}([0, 200]) = 2$; $\mathrm{id}_{\mathrm{check.aid}}((200, 400]) = 6$.

Given above partition results, we rewrite the following query conditions based on above formulas:

$$\begin{aligned}
&\delta_{\mathrm{cond}}(\mathrm{aid} = 256) \Longrightarrow \mathrm{aid}^* = 5,\\
&\delta_{\mathrm{cond}}(\mathrm{aid} < 180) \Longrightarrow \mathrm{aid}^* \in \{3, 7\},\\
&\delta_{\mathrm{cond}}(\mathrm{aid} > 240) \Longrightarrow \mathrm{aid}^* \in \{5, 1\}.
\end{aligned} \tag{9}$$

δ_{cond} (app.did = check.did) $\Rightarrow$ (app*.did* = 3 $\wedge$ check*.did* = 2) $\vee$ (app*.did* = 7 $\wedge$ check*.did* = 2) $\vee$ (app*.did* = 5 $\wedge$ check*.did* = 6) $\vee$ (app*.did* = 1 $\wedge$ check*.did* = 6).

δ_{cond} (app.did < check.did) $\Rightarrow$ (app*.did* = 3 $\wedge$ check*.did* = 2) $\vee$ (app*.did* = 3 $\wedge$ check*.did* = 6) $\vee$ (app*.did* = 7 $\wedge$ check*.did* = 2) $\vee$ (app*.did* = 7 $\wedge$ check*.did* = 6) $\vee$ (app*.did* = 5 $\wedge$ check*.did* = 6) $\vee$ (app*.did* = 1 $\wedge$ check*.did* = 6).

3.2.3. Query Optimization Principles. Because data is encrypted and stored in various places, in order to reduce the transmission cost and improve the query efficiency, we

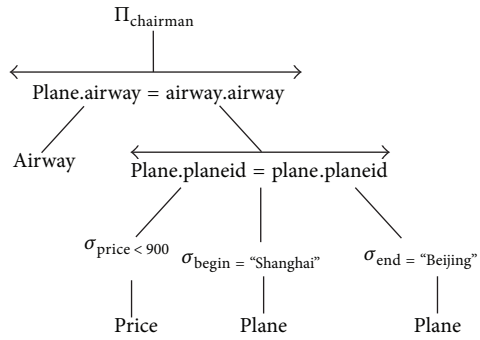

Figure 2: Initial syntax tree.

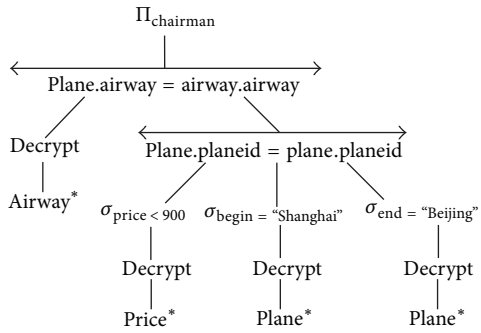

Figure 3: Syntax tree applied to cloud DB.

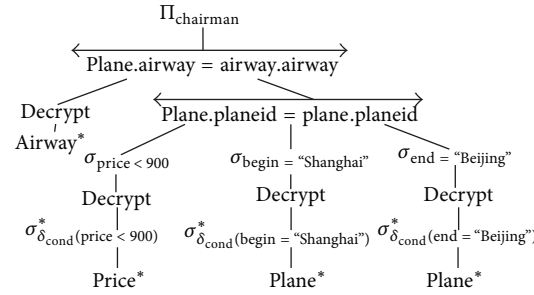

Figure 4: Rewriting syntax tree.

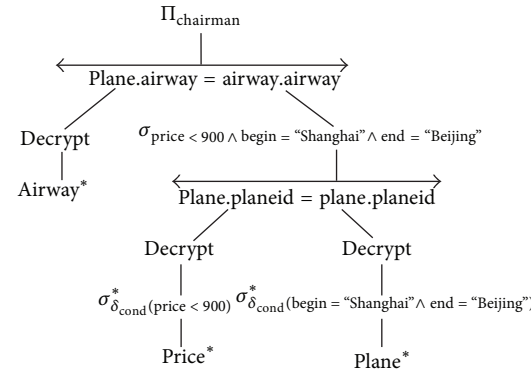

Figure 5: Moving selections in syntax tree.

should run operations on cloud services as much as possible, and the answers can be computed with little effort by the client.

For clear expression, operation procedure is expressed by using the syntax tree. The decryption operation splits the tree into cryptograph operations and plaintext operations. Because any single operation on the original tree ends with the selection after decryption; thereby the principle of query optimization by using syntax tree is to iteratively pull up the selection.

For example, given a selection "SELECT chairman FROM Airway, Price, Plane WHERE price < 900 AND begin = "shanghai" AND end = "beijing" AND Price.planeid = Plane.planeid AND Plane.airway = Airway.airway", we take query tree to illustrate how to optimize this query and describe its detailed procedures.

In Figure 2 the SQL statement is converted into an initial syntax tree. If the enterprise use cloud services or other offsite storage platforms, we need to first decrypt the cryptograph then query the data at client, as shown in Figure 3, where cryptograph database on cloud is bounded by the dotted line. Query objects (Price, Plane, and Airway) are converted to cryptograph tables (Price*, Plane* and Airway*) in the cloud database.

Operations on syntax tree are performed from bottom to up. In Figure 3 the first step is to execute selection, while the following steps include rewriting the condition of selection operations, converting it to a selection on cryptograph in cloud database and then decrypting and further filtering the result at client. A new syntax tree is derived as shown in Figure 4.

According to optimization principles described above, we should iteratively pull up selections. Therefore, by both exchanging the positions between selection operations (price < 900, begin = "shanghai" and end = "beijing") and join operation and then combing corresponding conditions, we obtain a new syntax tree, as shown in Figure 5.

Moreover, based on operation rewriting rules, join operation in Figure 5 should be converted into two parts, including the join on cryptograph in the cloud database and the selection on decrypted provisional results, as shown in Figure 6. Repeat the above steps, rewrite all kinds of operations, and continuously exchange the positions between selection operations and other operations, till all the selections cannot be pulled up. As a result, we get the ultima syntax tree as shown in Figure 7. Operations within dotted line would be executed on cloud service, whereas user only needs to execute the last selection. From here we see that the above method takes full advantage of cloud service to reduce the cost of transmitting and postprocessing and improve the efficiency of artifact querying in business process.

4. Case Study

In this section, we will introduce a business instance of a certain enterprise. Based on the method in Section 2, we complete the data modeling with artifact lifecycle from a given process instance and illustrate the query process through query tree mentioned in Section 3.

An enterprise's process of equipment purchase/scrap involves the following steps. At first equipment division fills out the equipment purchase/scrap application and hands it to

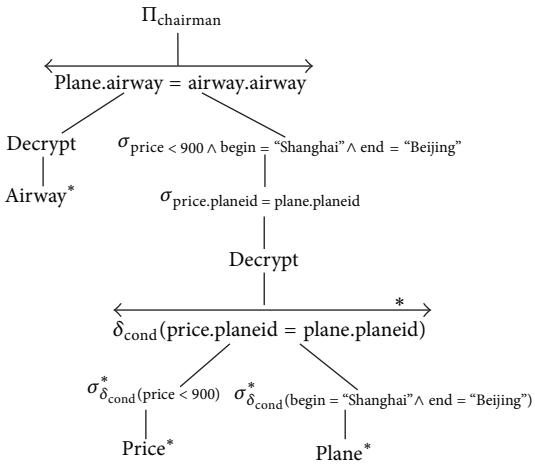

Figure 6: Rewriting syntax tree.

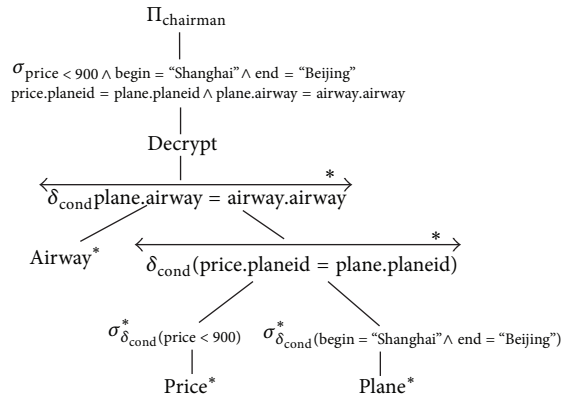

Figure 7: The ultima syntax tree.

department managers and company's leadership for approval. If the application is consented, then we should archive it; else we withdraw it. Purchasing department does purchase according to a copy of application, and when the purchase is completed, documents should be archived. Equipment division scraps the equipment based on specific methods and standards and then archives the processing results. Assets department regularly verifies company's assets based on purchase/scrap equipment information. Archive department has permission to query all the archived information.

This process involves multiple departments and multiple sets of information. If we manage the data alone, as business data are complicated, and even one attribute has difference value in different event, thereby it is difficult to manage. If we manage the process alone, only the department activities will be involved while business data in the process will be ignored. In this context, we analyze the process concerning both data and process and describe this instance with an Artiflow (N, S, R, C, Ru), where

N: "EP/S";

S: {FilloutEPA, Audit1, Audit2, Query, Purchase, Asset Verification...};

R: {NewEPA, PrimaryEPA, FinalEPA, Unapro-ved EPA, PO, FAL...};

C: {FtoN, NtoA,...};

Ru: {constraint (EPA) = (FilloutEPA, Audit1, Audit2)...}.

The model contains multiple artifacts: "EPA" is for describing equipment purchase application, while its lifecycle starts from filling, auditing to archiving. When having been archived, it will provide asset verification and support query processing. "ESA" is for describing equipment scrap application, while its lifecycle starts from filling, auditing to archiving. It is associated with another artifact, called "method & standard." The lifecycle of "PO"/"SL" captures process from purchase/scrap to application archive. The whole process is shown in Figure 8.

Artifact Example.

Artifact: (C, **A**, τ, Q, s, F)

C = "EPA";

A: {EquipmentName, PurchaseAmount, UnitPrice, ApplicationDate, Applicant, AuditingComment, AuditingDate};

Q: {empty table (initial state s), basic information filling, delivery auditing, auditing completion, audited application archiving (terminate state F)};

τ: EquipmentName: verchar; PurchaseAmount: In; UnitPrice: Int; ApplicationDate: Date; Applicant: Verchar; AuditingComment: Verchar; AuditingDate: Date.

Service Example.

service = (n, V_r, V_w, P, E), where:

n = "Audit2";

V_r: {EPA};

V_w: {EPA};

P: DEFINED (EquipmentName) ∧ DEFINED (PurchaseAmount) ∧ DEFINED (UnitPrice) ∧ DEFINED (ApplicationDate) ∧ DEFINED (Applicant) ∧ ¬DEFINED (AuditingComment) ∧ ¬DEFINED (AuditingDate).

E: DEFINED (EquipmentName) ∧ DEFINED (PurchaseAmount) ∧ DEFINED (UnitPrice) ∧ DEFINED (ApplicationDate) ∧ DEFINED (Applicant) ∧ DEFINED (AuditingComment) ∧ DEFINED (AuditingDate).

Repository Example.

R = (re, R_a, R_r, C_t).

re: "FinalEPA";

R_a: {EPA};

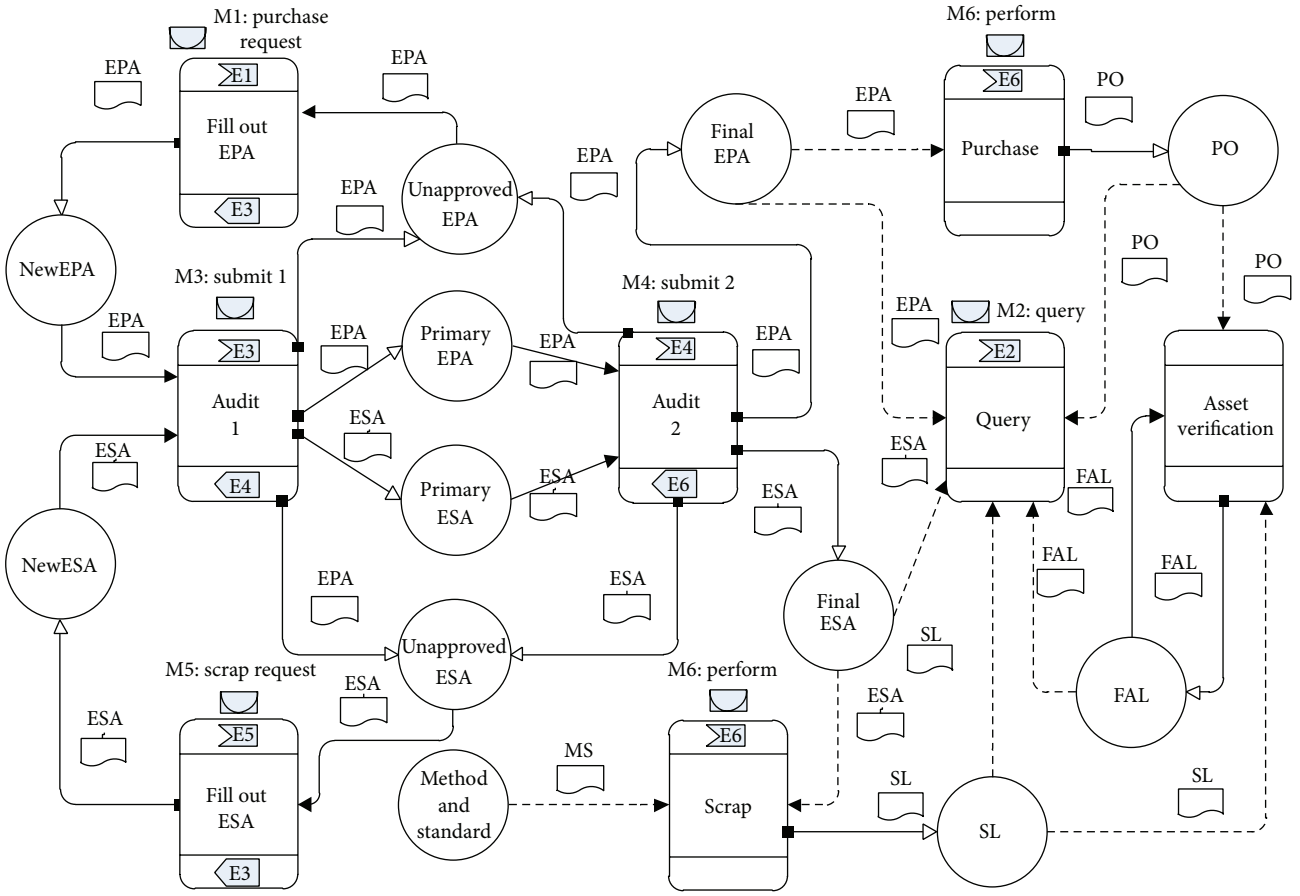

Figure 8: Lifecycle of business data in equipment purchase/scrap process.

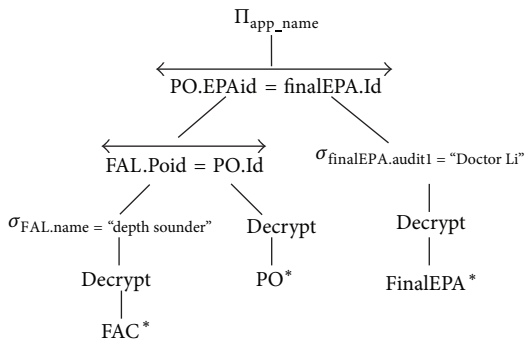

Figure 9: Initial query on EP/S.

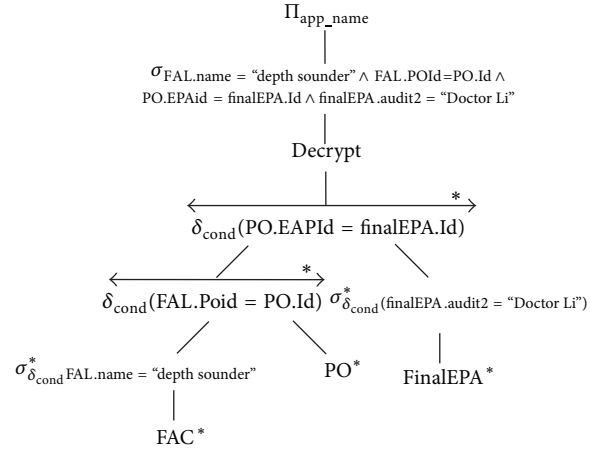

Figure 10: Ultima query on EP/S.

R_r: {EPA};

C_t: *IsDefine*(AuditingComment).

There is a repository named "FinalEPA," which reads and stores artifact "EPA" only if "AuditingComment" has been assigned.

Given a query "SELECT app_name FROM FAL, PO, FinalEPA WHERE Fal.name = 'depthsounder' AND FAL .POid = PO.Id AND PO.EPAid = FinalEPA.Id AND FinalEPA .Audit2 = 'Doctor Li'", it can be converted into a syntax tree as shown in Figure 9, which can be further converted into a new syntax tree shown in Figure 10. Queries will be issued on this syntax tree.

5. Conclusion

There is no doubt that more and more large datasets will be poured out during business process execution; meanwhile, these business data are extremely valuable. In this case, we modeled business data through its lifecycle from the perspective of process, which ensures the integrity of dynamic business data. Furthermore, we present the notion of user interest on business data, which has a superior function in

counting minimum interferential tuples during data partition and ensuring a lower cost of postprocessing brought by data partition. Considering current business data are mostly stored on cloud, we proposed a query rewriting strategy for off-site encrypted data which has a significant advantage in reducing the postprocessing cost. Currently there is little research on business data modeling and querying in the true sense. Our research lays great foundation for business data's application in enterprise. That is the initial step of business data management architecture, and we will further research on business data analysis with its lifecycle, to fully dig the significant value of business data.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by National Natural Science Foundation of China (61272098) and Science and Technology Development Foundation of Shanghai Ocean University.

References

[1] W. Huang, Z. Chen, W. Dong, H. Li, B. Cao, and J. Cao, "Mobile internet big data platform in china unicom," *Tsinghua Science and Technology*, vol. 19, no. 1, pp. 95–101, 2014.

[2] S. Sagiroglu and D. Sinanc, "Big data: a review," in *Proceedings of the International Conference on Collaboration Technologies and Systems (CTS '13)*, pp. 42–47, San Diego, Calif, USA, May 2013.

[3] M. Chui, B. Brown, J. Bughin et al., *Big Data: The Next Frontier for Innovation, Competition, and Productivity*, McKinsey Global Institute, 2011.

[4] M. Singh and S. K. Jain, "A survey on dataspace," in *Advances in Network Security and Applications*, vol. 196 of *Communications in Computer and Information Science*, pp. 608–621, Springer, 2011.

[5] K. Belhajjame, N. W. Paton, S. M. Embury, A. A. A. Fernandes, and C. Hedeler, "Incrementally improving dataspaces based on user feedback," *Information Systems*, vol. 38, no. 5, pp. 656–687, 2013.

[6] J.-P. Dittrich and M. A. Vaz Salles, "IDM: a unified and versatile data model for Personal Dataspace Management," in *Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB '06)*, pp. 367–378, September 2006.

[7] S. Pradhan, "Towards a novel desktop search technique," in *Database and Expert Systems Applications*, pp. 192–201, Springer, Berlin, Germany, 2007.

[8] M. Zhong, M. Liu, and Q. Chen, "Modeling heterogeneous data in dataspace," in *Proceedings of the IEEE International Conference on Information Reuse and Integration (IEEE IRI '08)*, pp. 404–409, Las Vegas, Nev, USA, July 2008.

[9] A. Sarma, X. Dong, and A. Halevy, "Data modeling in dataspace support platforms," in *Conceptual Modeling: Foundations and Applications*, pp. 122–138, Springer, Berlin, Germany, 2009.

[10] R. Hull and J. Su, *Report on NSF Workshop on Data-Centric Workflows*, 2012, http://dcw2009.cs.ucsb.edu/report.pdf.

[11] R. Hull, J. Su, and R. Vaculin, "Data management perspectives on business process management," in *Proceedings of the ACM SIGMOD Conference on Management of Data (SIGMOD '13)*, pp. 943–947, June 2013.

[12] K. Bhattacharya, N. S. Caswell, S. Kumaran, A. Nigam, and F. Y. Wu, "Artifact-centered operational modeling: lessons from customeengagements," *IBM Systems Journal*, vol. 46, no. 4, pp. 703–721, 2007.

[13] K. Bhattacharya, R. Guttman, K. Lyman et al., "A model-driven approach to industrializing discovery processes in pharmaceutical research," *IBM Systems Journal*, vol. 44, no. 1, pp. 145–162, 2005.

[14] R. Vaculín, R. Hull, T. Heath, C. Cochran, A. Nigam, and P. Sukaviriya, "Declarative business artifact centric modeling of decision and knowledge intensive business processes," in *Proceedings of the 15th IEEE International EDOC Enterprise Computing Conference (EDOC '11)*, pp. 151–160, September 2011.

[15] R. Vaculín, R. Hull, M. Vuković, T. Heath, N. Mills, and Y. Sun, "Supporting collaborative decision processes," in *Proceedings of the IEEE 10th International Conference on Services Computing (SCC '13)*, pp. 651–658, July 2013.

[16] A. Nigam and N. S. Caswell, "Business artifacts: an approach to operational specification," *IBM Systems Journal*, vol. 42, no. 3, pp. 428–445, 2003.

[17] G. Liu, X. Liu, H. Qin et al., "Automated realization of business workflow specification," in *Proceedings of the 1st International Workshop on SOA, Globalization, People, and Work (SG-PAW '09)*, pp. 8–9, 2009.

[18] H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra, "Executing SQL over encrypted data in the database-service-provider model," in *Proceedings of the ACM SIGMOD International Conference on Managment of Data*, pp. 216–227, June 2002.