Data & Knowledge Engineering 111 (2017) 39–57

Contents lists available at ScienceDirect

Data & Knowledge Engineering

journal homepage: www.elsevier.com/locate/datak

Maximal object+: An acyclic semantic structure on the universal relation model

In-Joong Kim, Kyu-Young Whang*, Hyuk-Yoon Kwon

Department of Computer Science, Korea Advanced Institute of Science and Technology (KAIST), 291 Daehak-ro, Yuseong-gu, Daejeon 305-338, Republic of Korea

ARTICLE INFO

Keywords:
Maximal object+
Semantic structure
Universal relation
USQL
Data models
Database semantics

ABSTRACT

The universal relation model allows us to find the results on a virtual relation that joins all the relations in a relational database if the user specifies only selection and projection conditions. It automatically finds actual relations in which the selection and projection conditions are performed and possible join paths among the relations using the database schema. When there are cycles in the database schema graph, however, the universal relation model may lose some results since it adds an unintentional condition to the interpretation of a query. Here, the *unintentional condition* is an equality condition that intersects multiple query interpretations by different join paths, and thus, it returns only part of the results that the user intends. This paper proposes a new semantic structure, *maximal object+*, that completely removes cycles in a universal relation. A maximal object+ is a largest acyclic connected component in the database schema graph where the entire set of relations in the component has the lossless join property. Here, the important point is that a maximal object+ allows only the lossless join property indicated by functional dependencies excluding the lossless join property indicated by only multivalued dependencies. To show the advantage of a maximal object+, we compare the effectiveness of the query processing method based on maximal object+ with that of the existing query processing method based on the maximal object by performing experiments on the synthetic and real datasets. As a result, we show that our method significantly outperforms the one based on the maximal object in terms of mean recall when a cycle exists in a maximal object while maintaining comparable efficiency. Specifically, our method improves the mean recall by up to 8.15 times for the dataset whose schema involves cycles.

1. Introduction

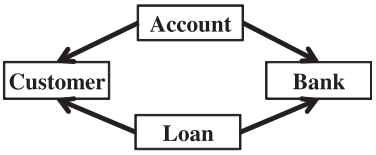
The user of a relational database normally writes a query in Structured Query Language (SQL) to extract information from the database. Here, the user should specify the relations to access and the relationships among those relations (i.e., join paths). For this, the user must know the database schema. This constraint degrades the usability of a database when the schema of the database is complex [12]. To improve the usability of relational databases, the universal relation model [6] has been proposed. The universal relation model finds the results on a virtual relation that joins all the relations in a relational database if the user specifies *only selection and projection conditions* [6]. Fig. 1(a) shows the schema of the BANK database. The schema graph has a cycle if we ignore the directions of the edges. From now on, we use the term “cycle” for a cycle obtained by ignoring the directions of the edges of a graph. Here, the attributes underlined by a straight line represent primary keys; the attributes underlined by a dotted line foreign

* Corresponding author.
E-mail addresses: ijkim@mozart.kaist.ac.kr (I.-J. Kim), kywhang@mozart.kaist.ac.kr (K.-Y. Whang), hykwon@mozart.kaist.ac.kr (H.-Y. Kwon).

http://dx.doi.org/10.1016/j.datak.2017.06.004
Received 5 April 2016; Received in revised form 11 March 2017; Accepted 25 June 2017
Available online 27 June 2017
0169-023X/

Customer ($\underline{Cid}$, $Cname$)
Bank ($\underline{Bid}$, $Bname$)
Account ($\underline{C\_id}$, $\underline{B\_id}$, $Balance$)
Loan ($\underline{C\_id}$, $\underline{B\_id}$, $Amount$)

(a) The database schema.



(b) The database schema graph.

U (Cid, Bid, $Cname$, $Bname$, $Balance$, $Amount$)

(c) The schema of the universal relation U.

Fig. 1. The BANK database.

retrieve ⟨attribute list⟩
where ⟨condition⟩

Fig. 2. The query format for the universal relation.

retrieve $Bname$
where $Cname$ = 'Smith'

Fig. 3. $Query_1$: a query on the universal relation for the BANK database.

keys. Fig. 1(b) shows the schema graph of the BANK database. Fig. 1(c) shows the schema of the universal relation for the BANK database. Fig. 2 shows the query format of the Universal SQL (simply, USQL) [21] on the universal relation. Here, the user specifies selection conditions in the where clause and projection conditions in the retrieve clause. ⟨attribute list⟩ consists of (a_1, a_2, ⋯, a_m) where a_i is an attribute. ⟨*condition*⟩ is a boolean expression that connects arithmetic expressions over attributes. Fig. 3 shows a query example, $Query_1$. The meaning of $Query_1$ is that "Find the names of the banks where a customer whose name is 'Smith' is related".

When using the universal relation model, the system should determine the join paths which are used together with the selection and projection conditions in the query to interpret the user's intention correctly. Fagin et al. [6] have proposed a method, which we call UR method, that finds all the possible join paths among the relations containing all the attributes used in the query (i.e., selection and projection conditions). When there are no cycles in the database schema graph, we have only one join path [6]; hence, the UR method can interpret the user's intention correctly. For example, if we remove the *Loan* relation from the BANK database, the UR method interprets $Query_1$ as follows. The *Customer* relation contains *Cname* and the *Bank* relation contains *Bname*. Then, a join path that connects the two relations is *Customer* − *Account* − *Bank*. The UR method adds the join condition representing the join path to the where clause of $Query_1$, and thus, the where clause becomes " **where** *Customer*. *Cid* = *Account*. *Cid* and *Account*. *Bid* = *Bank*. *Bid* and *Customer*. *Cname*= 'Smith'". The interpreted meaning of $Query_1$ is that "Find the names of the banks where a customer whose name is 'Smith' has an account".

When there are cycles in the database schema graph, however, the UR method proposed by Fagin et al. [6] cannot correctly interpret users' intention. In this case, we have two or more join paths that connect the relations containing all the attributes used in the query. Hence, we can interpret a query in multiple ways where a join path corresponds to an interpretation. For example, we can interpret $Query_1$ as two different join paths: 1) *Customer* − *Account* − *Bank* or 2) *Customer* − *Loan* − *Bank*. The former means that "Find the names of the banks where a customer whose name is 'Smith' has an account'; the latter "Find the names of the banks from which a customer whose name is 'Smith' borrows some money". However, due to the cycle, the UR method adds an equality condition that intersects the two interpretations of the query. The join path for $Query_1$ on the schema graph has a cycle: *Bank* − *Loan* − *Customer* − *Account* − *Bank*. The where clause of $Query_1$ that contains the join path as a join condition is "**where** *Customer*. *Cid* = *Account*. *Cid* and *Account*. *Bid* = *Bank*. *Bid* and *Customer*. *Cid* = *Loan*. *Cid* and *Loan*. *Bid* = *Bank*. *Bid* and *Customer*. *Cname*= 'Smith'". This interpretation of the query means that "Find the names of the banks where a customer whose

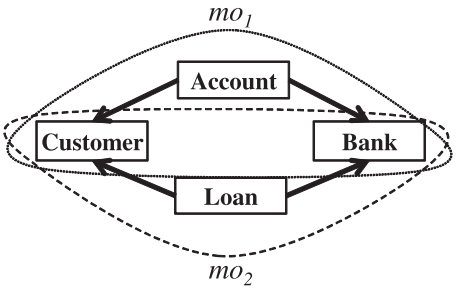


Fig. 4. Maximal objects for the BANK database schema.

name is 'Smith' has an account and borrows money from". The interpreted meaning is equivalent to the intersection of two interpretations of the query by different join paths. That is, the 'AND operation' adds the following meaning to the interpretation of the query: "the customer who has an account on a bank is the same as the customer who borrows money from the bank", which is not intended by the user. What the user would intend is "Find the names of the banks where a customer whose name is 'Smith' has an account **or** borrows money from". We call the 'equality condition' by the AND operation as the *unintentional condition*.

Maier et al. [15] have proposed the notion of the maximal object to remove cycles in universal relations. The *maximal object* is the maximal set of relations where the set of relations has the lossless join property [15]. Maier et al. [15] have also proposed a new method to interpret the query using the maximal object. First, it splits the schema graph into a set of maximal objects. Next, it interprets the query on each maximal object. Finally, it unions the results obtained by evaluating the query on each maximal object. Fig. 4 shows maximal objects for the BANK database. These maximal objects are obtained as follows. R_{set1}:={*Customer*, *Account*, *Bank*} has the lossless join property; R_{set2}:={*Customer*, *Loan*, *Bank*} has the lossless join property. Furthermore, $R_{set1} \cup \{Loan\}$ does not have the lossless join property since neither $R_{set1} \cap \{Loan\} \rightarrow R_{set1}$ nor $R_{set1} \cap \{Loan\} \rightarrow \{Loan\}$ holds[1] [26]. Also, $R_{set2} \cup \{Account\}$ does not have the lossless join property since neither $R_{set2} \cap \{Account\} \rightarrow R_{set2}$ nor $R_{set2} \cap \{Account\} \rightarrow \{Account\}$ holds [26]. Hence, since $R_{set1}(R_{set2})$ is a maximal set of relations that has the lossless join property, $R_{set1}(R_{set2})$ is a maximal object mo_1 (mo_2). There are no cycles in the two maximal objects.

We show that, despite the intent of the maximal object, the maximal object cannot completely remove cycles in a universal relation. Our work is the first that makes this observation. Existing works [15,17,21] do not consider this problem despite that it is a serious problem. Fig. 5(a) shows the schema of the purchase database. Fig. 5(b) shows the schema graph and maximal objects for the database. The set of all relations in the purchase database {*Customer*, *Item*, *Nation*, *Orders*} has the lossless join property. Hence, by the definition of the maximal object, all relations in the set and all primary key-foreign key (simply, PK-FK) relationships connecting those relations become a maximal object mo_1. As the result, the maximal object has a cycle. We elaborate in detail on these cases where the maximal object cannot remove cycles in Section 2.

We propose the *maximal object+* that completely removes cycles in a universal relation. A maximal object+ is a largest acyclic connected component in the database schema graph where the entire set of relations in the component has the lossless join property. Here, the important point is that a maximal object+ allows only the lossless join property indicated by functional dependencies (simply, FDs)[2]. That is, a maximal object+ does not allow the lossless join property that is not indicated by functional dependencies but indicated by multivalued dependencies (simply, MVDs). The salient points of the maximal object+ are as follows. 1) It eliminates the equality condition that the user does not intend. 2) It interprets the query not to generate the meaningless results by a lossy join since it consists of the set of relations that has the lossless join property. 3) It does not generate Cartesian product results that are incurred by multivalued dependencies since it allows only the lossless join property indicated by FDs.

To show the advantage of a maximal object+, we compare the effectiveness of the USQL query processing method based on the maximal object+ with that of the existing USQL query processing method based on the maximal object [15] by performing experiments on the synthetic and real datasets. As a result, we show that our method significantly outperforms the one based on the maximal object in terms of mean recall, especially when a cycle exists in a maximal object while maintaining comparable efficiency. Specifically, our method improves the mean recall by up to 8.15 times for the dataset whose schema involves cycles.

The technical contributions of the paper are as follows:

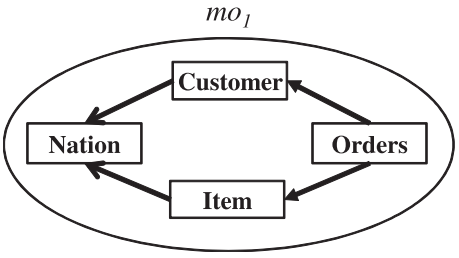
- We have identified for the first time that the maximal object may include a cycle.
- We have proposed a new semantic structure, called maximal object+, which completely removes cycles from a universal relation.
- We have shown that our method significantly outperforms the one based on the maximal object in terms of mean recall when a cycle exists in a maximal object while maintaining comparable efficiency.

The rest of this paper is organized as follows. Section 2 presents our research goal and explains that the maximal object does not satisfy the research goal. Section 3 proposes a maximal object+ that always removes cycles in the universal relation while the set of

[1] {*Cid*, *Bid*} ({*Cid*, *Bid*}) is not the key of the *Account* (*Loan*) relation.
[2] These notations will be further elaborated in Section 2.

Customer (*Cid*, *Nid*, *Cname*)
Item (*Iid*, *Nid*, *Iname*)
Nation (*Nid*, *Nname*)
Orders (*Oid*, *Cid*, *Iid*)

(a) The database schema.



(b) The schema graph and the maximal objects.

Fig. 5. The schema, schema graph, and maximal objects for the purchase database.

relations in a maximal object+ has the lossless join property indicated by FDs. Section 4 describes related work. Section 5 presents the experimental results. Section 6 summarizes and concludes the paper.

2. Motivation

In this section, we describe our research goal and explain that the maximal object does not satisfy the research goal. We first formally define the notation that is used throughout the paper.

Let us assume a universal relation that consists of a set of l attributes $U:=\{A_1, A_2, \ldots, A_l\}$ and a relational database D that consists of a set of n relations $\{R_1, R_2, \ldots, R_n\}$ where $U:=\bigcup_{i=1}^{n} R_i$[3] holds. We denote a PK-FK relationship where a foreign key of R_i references a primary key of R_j by $R_i \rightarrow R_j$. Table 1 shows the notations used in the paper.

The goal of the paper is to propose a semantic structure that satisfies the following two conditions in the database schema graph: 1) the semantic structure does not contain (undirected) cycles; 2) the set of the relations in the semantic structure has the lossless join property indicated by FDs. Condition 1 is required since a semantic structure containing a cycle adds an unintentional condition to the interpretation of the query as explained in Section 1. Condition 2 is required since the join of relations in a set of relations that does not have the lossless join property indicated by FDs but has the lossless join property indicated by MVDs generates Cartesian product results. Here, $\{R_i, R_j\}$ has the lossless join property indicated by FDs if and only if either $R_i \cap R_j$[4] $\rightarrow (R_i - R_j)$ or $R_i \cap R_j \rightarrow (R_j - R_i)$ holds [26]; $\{R_i, R_j\}$ has the lossless join property indicated by MVDs if and only if either $R_i \cap R_j \twoheadrightarrow (R_i - R_j)$ or $R_i \cap R_j \twoheadrightarrow (R_j - R_i)$ holds [26]. We call Condition 1 the *acyclic requirement* and Condition 2 the *FD-based losslessness requirement*.

In Definition 1, we define the MVD connection. In Lemma 1, we prove that the semantic structure that does not have an MVD connection and is a connected graph always satisfies Condition 2, i.e., the FD-based losslessness requirement.

Definition 1. An *MVD connection* is defined as a structure of $R_i \rightarrow R_j \leftarrow R_k$ on a schema graph. □

Lemma 1. *If the schema graph for a set of the relations does not contain an MVD connection and is a connected graph, the set of relations always has the lossless join property indicated by FDs with respect to F.*

Proof. We prove the Lemma by mathematical induction.

(Base case) For three relations R_1, R_2, and R_3, we have the following four cases for possible connections among the relations based on PK-FK relationships: 1) $R_1 \rightarrow R_2 \rightarrow R_3$, 2) $R_1 \leftarrow R_2 \leftarrow R_3$, 3) $R_1 \leftarrow R_2 \rightarrow R_3$, and 4) $R_1 \rightarrow R_2 \leftarrow R_3$.

We show that cases 1, 2, and 3 represent the set of the relations has the lossless join property indicated by FDs with respect to F. In contrast, case 4 represents an MVD connection so that the set of the relations has the lossless join property indicated by MVDs but not by FDs with respect to F. In case 1, there is a PK-FK relationship from R_1 to R_2. Since $R_1 \cap R_2$ contains the primary key of R_2, $R_1 \cap R_2 \rightarrow (R_2 - R_1)$ holds. Hence, $\{R_1, R_2\}$ has the lossless join property indicated by FDs with respect to F [26]. In addition, since there is a PK-FK relationship from R_2 to R_3, there is a PK-FK relationship from $R_1 \cap R_2$ to R_3. Hence, $\{R_1, R_2, R_3\}$ has the lossless join property indicated by FDs with respect to F [26]. The same holds for case 2 since it is symmetric to case 1. In case 3, since there is a PK-FK relationship from R_2 to R_1, $\{R_1, R_2\}$ has the lossless join property indicated by FDs with respect to F. In addition, since there is a PK-FK relationship from R_2 to R_3, there is a PK-FK relationship from $R_1 \cup R_2$ to R_3. Hence, $\{R_1, R_2, R_3\}$ has

[3] $\bigcup_{i=1}^{n} R_i$ represents the set of attributes that belong to relations $R_1, R_2, \ldots, R_n$.
[4] $R_i \cap R_j$ represents the set of attributes that belong to both relations; $R_i \cup R_j$ the set of attributes that belong to either R_i or R_j.

Table 1
The notations.

Symbols	Definitions
G	A directed graph that models each relation in the database as a node and each PK-FK relationship as a directed edge
$PK-FK_{ij}$	The PK-FK relationship from R_i to R_j
V_i	The node mapped to a relation R_i in G
E_{ij}	The edge mapped to $PK-FK_{ij}$ in G
$V(G)$	The set of all V_i's in G
$E(G)$	The set of all E_{ij}'s in G
$R(V_i)$	The relation R_i mapped to V_i in G
$V(R_i)$	The node V_i in G mapped to R_i
$PK-FK(E_{ij})$	The PK-FK relationship $PK-FK_{ij}$ mapped to E_{ij} in G
$E(PK-FK_{ij})$	The edge E_{ij} in G mapped to $PK-FK_{ij}$
F	The set of functional dependencies on the set of relations in a semantic structure that includes the functional dependencies among the attributes in each relation and the ones among the relations. The functional dependencies among relations considered are only those represented (i.e., implied) by the PK-FK relationships [8].

the lossless join property indicated by FDs with respect to F. In case 4, since there is a PK-FK relationship from R_1 to R_2, $\{R_1, R_2\}$ has the lossless join property indicated by FDs with respect to F. However, there is no PK-FK relationship from $R_1 \cup R_2$ to R_3. In addition, since there is a PK-FK relationship from R_3 to R_1. $\{R_2, R_3\}$ has the lossless join property indicated by FDs with respect to F. However, there is no PK-FK relationship from $R_2 \cup R_3$ to R_1, either. Hence, $\{R_1, R_2, R_3\}$ does not have the lossless join property indicated by FDs with respect to F. Meanwhile, both $R_2 \twoheadrightarrow (R_1 - R_2)$ and $R_2 \twoheadrightarrow (R_3 - R_2)$ hold for the following reason: since there is a PK-*FK relationship from* R_1 to R_2 and R_3 to R_2, there is a one-to-many relationship from R_2 to R_1 and one from R_2 to R_3. Hence, $\{R_1, R_2, R_3\}$ has the lossless join property indicated by MVDs [26].

(Hypothesis) For the schema graph of $R_{set} = \{R_1, R_2, \cdots, R_i\} (i \geq 3)$ where the schema graph is connected and does not have an MVD connection, R_{set} has the lossless join property indicated by FDs with respect to F.

(Induction) We add another relation R_{i+1} to R_{set} so as not to have an MVD connection. Then, the following two cases are possible: a) connecting a relation $R_k (1 \leq k \leq i)$ in R_{set} to the relations in R_{i+1} based on a PK-FK relationship and b) connecting R_{i+1} to a relation in R_{set} based on a PK-FK relationship. In the case a, whichever relation $R_k (1 \leq k \leq i)$ is connected to R_{i+1}, the relations in $R_{set} \cup R_{i+1}$ does not have an MVD connection. Let us suppose that $S = \cup_{R_j \in R_{set}} R_j$. Since there is a PK-FK relationship from R_k to R_{i+1}, there is a PK-FK relationship from S to R_{i+1}. Hence, $\{R_1, R_2, \cdots, R_{i+1}\}$ has the lossless join property indicated by FDs with respect to F. In the case b, we need to connect R_{i+1} to a relation R_d in R_{set} so as not to have an MVD connection. Then, R_d must be the one having a directed path to every relation in R_{set}. Here, there is a PK-FK relationship from R_{i+1} to R_d. In addition, for all $R_j \in \{R_1, R_2, \cdots, R_i\}$, $R_d \rightarrow R_j$ holds since R_d has a directed path to R_j. Therefore, the primary key of S is the same as that of R_d. Since there is a PK-FK relationship from R_{i+1} to R_d, there is a PK-FK relationship from R_{i+1} to S. Therefore, $\{R_1, R_2, \cdots, R_{i+1}\}$ has the lossless join property indicated by FDs with respect to F. □

Maier et al. [15] have proposed the maximal object to remove cycles in a universal relation. In Definition 2, we first present the definition of the maximal object [15]. Then, we show that the maximal object does not always satisfy the acyclic requirement and FD-based losslessness requirement.

Definition 2. [15] In a database D, a set of relations and PK-FK relationships among the relations $(\{R_i, 1 \leq i \leq l\}, \{PK-FK_{ij}, 1 \leq i, j \leq l\})$ is defined as a *maximal object* if it satisfies the following two conditions: (1) (maximality) $\nexists(R_i)((V(R_i) \in V(G))$ AND $(R_i \notin \{R_j, 1 \leq j \leq l\})$ AND $S = \cup_{\{1 \leq k \leq l\}} R_k$ AND $LOSSLESS(S, R_i))$ (2) $\{PK-FK_{ij}, 1 \leq i, j \leq l\} = \{PK-FK(e_{ij}) \mid R_i, R_j \in \{R_k, 1 \leq k \leq l\}$ AND G has an edge e_{ij} between $V(R_i)$ and $V(R_j)$.$\}$. For a maximal object mo, we denote the set of all relations in mo as $mo.R_{set}$ and the set of all the PK-FK relationships as $mo.PK-FK_{set}$. In addition, we denote the set of all the maximal objects in D as $MOset(D)$.□

Condition 1 of Definition 2 means that the set of all the relations in a maximal object is the maximal set of relations that has the lossless join property. Here, if two input relations form a lossless join, the predicate *LOSSLESS* is true; otherwise, it is false. There are three versions of *LOSSLESS*: 1) $LOSSLESS(R_i, R_j)$ if and only if $(R_i \cap R_j) \rightarrow (R_i - R_j)$ or $(R_i \cap R_j) \rightarrow (R_j - R_i)$ (the "FDs only" rule), 2) $LOSSLESS(R_i, R_j)$ if and only if $(R_i \cap R_j) \twoheadrightarrow (R_i - R_j)$ or $(R_i \cap R_j) \twoheadrightarrow (R_j - R_i)$ (the "MVD only" rule), and 3) $LOSSLESS(R_i, R_j)$ if and only if (a) $(R_i \cap R_j) \rightarrow (R_i - R_j)$ or $(R_i \cap R_j) \rightarrow (R_j - R_i)$ or (b) $(R_i \cap R_j) \twoheadrightarrow (R_i - R_j) | (R_j - R_i)$ but NOT $[(R_i - R_j) \rightarrow (R_i \cap R_j)$ or $(R_j - R_i) \rightarrow (R_i \cap R_j)]$. The first version of *LOSSLESS* means that the set of two relations R_i and R_j has the lossless join property indicated by FDs. The second version means that the set of two relations R_i and R_j has the lossless join property indicated by MVDs. The third version means that the set of two relations R_i and R_j that has the lossless join property indicated by FDs or has the lossless join property indicated by MVDs except those induced by FDs. Condition (2) of Definition 2 means that a maximal object contains all the PK-FK relationships among the relations in the maximal object.

We show that the maximal object does not always satisfy the acyclic requirement and FD-based losslessness requirement. If we use the second or third versions of the LOSSLESS predicate, the maximal object can contain an MVD connection since those predicates allow a set of relations that has the lossless join property indicated by MVDs. Hence, we use the first version of the LOSSLESS predicate to avoid a maximal object having an MVD connection.

Table 2
Maximal objects as the number of MVD connections and existence of cycles vary in a database schema graph.

Case number	Database schema graph		Maximal object	
	Existence of a cycle	Number of MVD connections	Existence of a cycle	Number of MVD connections
1	No	0	No	0
2		n (n>0)		
3	Yes	0	Yes	0
4		1		1
5		n (n>1)	No	0

Table 2 shows the number of an MVD connection and existence of cycles in a maximal object for a database schema graph. In Case 1, there is no cycle or MVD connection in the database schema graph. In this case, by Lemma 1, the set of relations has the lossless join property indicated by FDs. In this case, the set of all relations becomes one maximal object, having no cycles or MVD connections.

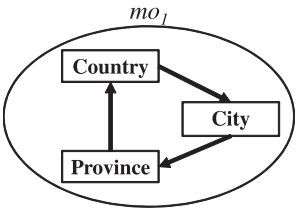
In Case 2, there is no cycle and there are one or more MVD connections in the database schema graph. Let us consider n relations. If there is one MVD connection, the relations are connected as $R_1 \rightarrow R_2 \rightarrow \cdots \rightarrow R_{i-1} \rightarrow R_i \leftarrow R_{i+1} \leftarrow \cdots \leftarrow R_n (2 \le i \le n-1)$. Here, the MVD connection is $R_{i-1} \rightarrow R_i \leftarrow R_{i+1}$. Let us create a set of relations R_{set1}: $\{R_1\}$. From R_2 to R_i, we add a relation R_j $(2 \le j \le i)$ to R_{set1} one by one as long as $R_{set1} \cup \{R_j\}$ has the lossless join property indicated by FDs. Then, R_{set1} becomes $\{R_1, R_2, \cdots, R_i\}$. Likewise, let us create a set of relations R_{set2}: $\{R_n\}$. From R_{n-1} to R_i, we add a relation R_j $(n-1 \ge j \ge i)$ to R_{set2} one by one as long as $R_{set2} \cup \{R_j\}$ has the lossless join property indicated by FDs. Then, R_{set2} becomes $\{R_i, R_{i+1}, \cdots, R_n\}$. The relations in R_{set1} are connected as $R_1 \rightarrow R_2 \rightarrow \cdots \rightarrow R_i$ and those in R_{set2} as $R_i \leftarrow R_{i+1} \leftarrow \cdots \leftarrow R_n$. Since the relations in $R_{set1}(R_{set2})$ do not have an MVD connection, $R_{set1}(R_{set2})$ has the lossless join property indicated by FDs by Lemma 1. Here, if we add a relation $R_{i+1}(R_{i-1})$ to $R_{set1}(R_{set2})$, the set does not have the lossless join property indicated by FDs for the following reason. First, let us consider $R_{set1} \cup \{R_{i+1}\}$. Let $S = \cup_{R_k \in R_{set1}} R_k$. Since there is no PK-FK relationship from a relation R_j $(1 \le j \le i)$ in R_{set1} to a relation R_{i+1}, there is no PK-FK relationship from S to R_{i+1}. There is a PK-FK relationship from R_{i+1} to R_i. However, since the primary key of R_i is not the same as that of S, there is no PK-FK relationship from R_{i+1} to S. Hence, neither $S \cap R_{i+1} \rightarrow (S - R_{i+1})$ nor $S \cap R_{i+1} \rightarrow (R_{i+1} - S)$ holds. Therefore, $R_{set1} \cup \{R_1\}$ does not have the lossless join property indicated by FDs [26]. Likewise, $R_{set2} \cup \{R_{i-1}\}$ does not have the lossless join property indicated by FDs, either. As the result, since we cannot add any relations to R_{set1} or R_{set2}, each one of R_{set1} and R_{set2} satisfies Condition (1) of Definition 2. Therefore, there are two maximal objects that do not have MVD connections for the schema graph. One maximal object is R_{set1} and the other one is R_{set2}. When there are n (≥2) MVD connections, we obtain $n + 1$ maximal objects. Those maximal objects do not have an MVD connection.

In Case 3, there is a cycle in the database schema graph and there is no MVD connection in the cycle. In this case, the maximal object for the database schema graph has a cycle. Here, the set of all relations in the cycle has the lossless join property indicated by FDs. The reason is as follows. Suppose that the number of relations in one cycle is n. If there is no MVD connection in the cycle, the relations are connected as $R_1 \rightarrow R_2 \rightarrow \cdots \rightarrow R_n \rightarrow R_1$. Since the relations in $\{R_1, R_2, \cdots R_n\}$ do not have an MVD connection, $\{R_1, R_2, \cdots R_n\}$ has the lossless join property indicated by FDs by Lemma 1.

In Case 4, there is a cycle in the database schema graph and there is only one MVD connection in the cycle. In Case 2, we have already shown that an MVD connection that is not in a cycle cannot be made part of a maximal object. Therefore, in Case 4, we deal with an MVD connection only when it is in a cycle. In this case, the maximal object for the database schema graph has a cycle, and this cycle has an MVD connection for the following reason. Suppose that the number of relations in the cycle is n. The relations are connected as $R_1 \rightarrow R_2 \rightarrow \cdots \rightarrow R_{i-1} \rightarrow R_i \leftarrow R_{i+1} \leftarrow \cdots \leftarrow R_n \rightarrow R_1$ $(2 \le i \le n-1)$. The MVD connection is $R_{i-1} \rightarrow R_i \leftarrow R_{i+1}$. Here, let us consider the set of all relations except the relation R_i that is contained in the MVD connection, R_{set1}: $\{R_1, R_2, \cdots, R_{i-1}, R_{i+1}, \cdots, R_n\}$. Then, the relations are connected as $R_{i+1} \leftarrow \cdots \leftarrow R_n \rightarrow R_1 \rightarrow R_2 \rightarrow \cdots \rightarrow R_{i-1}$. Since the relations in R_{set1} do not have an MVD connection, R_{set1} has the lossless join property indicated by FDs by Lemma 1. Let $S = \cup_{R_k \in R_{set1}} R_k$. Since there is a PK-FK relationship from S to R_i, the set of all relations in the cycle, i.e., $R_{set1} \cup \{R_i\}$, has the lossless join property indicated by FDs.

Country (*CountryId*, *CityId*, *CountryName*)
City (*CityId*, *ProvinceId*, *CityName*)
Province (*ProvinceId*, *CountryId*, *ProvinceName*)

(a) The database schema.



(b) The schema graph and the maximal object mo_1.

Fig. 6. The schema, schema graph, and maximal object for the geographic database.

In Case 5, there is a cycle in the database schema graph and there are two or more MVD connections in the cycle. In this case, a maximal object for the database schema graph has neither a cycle nor an MVD connection for the following reason. Suppose that the number of relations in the cycle is n. If there are two MVD connections, the relations are connected as $R_1 \rightarrow R_2 \leftarrow R_3 \leftarrow \cdots \leftarrow R_{i-1} \rightarrow R_i \leftarrow R_{i+1} \leftarrow \cdots \leftarrow R_n \rightarrow R_1 (4 \leq i \leq n-1)$. Here, the MVD connections are $R_1 \rightarrow R_2 \leftarrow R_3$ and $R_{i-1} \rightarrow R_i \leftarrow R_{i+1}$. Here, we can find two sets of connected relations where the relations in each set do not have an MVD connection: (1) R_{set1}: $\{R_2, R_3, \cdots, R_i\}$ and (2) R_{set2}: $\{R_i, R_{i+1}, \cdots, R_n, R_1, R_2\}$. The relations in R_{set1} are connected as $R_2 \leftarrow R_3 \leftarrow \cdots \leftarrow R_{i-1} \rightarrow R_i$; those in R_{set2} as $R_i \leftarrow R_{i+1} \leftarrow \cdots \leftarrow R_n \rightarrow R_1 \rightarrow R_2$. Since neither R_{set1} nor R_{set2} has an MVD connection, each set has the lossless join property indicated by FDs by Lemma 1.

If we add R_1 or R_{i+1} (R_3 or R_{i-1}) to $R_{set1}(R_{set2})$, each set does not have the lossless join property indicated by FDs. The reason is as follows. First, consider $R_{set1} \cup \{R_1\}$. Let $S = \cup_{R_k \in R_{set1}} R_k$. $S \cap R_1 \rightarrow (S - R_1)$ does not hold since there is no PK-FK relationship from S to R_1. In addition, $S \cap R_1 \rightarrow (R_1 - S)$ does not hold either since there is no PK-FK relationship from R_1 to S due to the following reason. There is a PK-FK relationship from R_1 to R_2. However, since R_2 does not have a directed path to R_j, for all $R_j \in (R_{set1} - \{R_2\})$, the primary key of R_2 is not the same as that of S. Thus, there is no PK-FK relationship from R_1 to S. Therefore, $R_{set1} \cup \{R_1\}$ does not have the lossless join property indicated by FDs [26]. Likewise, $R_{set1} \cup \{R_{i+1}\}$, $R_{set2} \cup \{R_3\}$, and $R_{set2} \cup \{R_{i-1}\}$ do not have the lossless join property indicated by FDs, either. Hence, since we cannot add any relations to R_{set1} or R_{set2}, each set satisfies Condition (1) of Definition 2. Therefore, there are two maximal objects that do not have an MVD connection for the schema graph; R_{set1} and R_{set2}. When there are n ($\geq$2) MVD connections in the cycle, we obtain $n + 1$ maximal objects in a similar way. Those maximal objects do not have a cycle nor MVD connection.

As shown in Table 2, a maximal object has a cycle in Cases 3 and 4. Examples 1 and 2 show an example of each case.

Example 1. Let us consider the geographic database in Fig. 6(a). In the database, the *CityId* attribute of the *Country* relation represents the capital of a country; the *ProvinceId* attribute of the *City* relation the province where a city belongs; the *CountryId* attribute of the *Province* relation the country where a province belongs. Fig. 6(b) shows the schema graph of the database. The schema graph has a cycle, but does not have an MVD connection. Since the set of all relations in the database {*Country*, *City*, *Province*} has the lossless join property indicated by FDs, we obtain a maximal object mo_1: ({*Country*, *City*, *Province*}, {*Country* → *City*, *City* → *Province*, *Province* → *Country*}), which contains the set of all relations and all PK-FK relationships. Here, mo_1 has a cycle.

Consider a query $Query_2$: **retrieve** *CityName* **where** *CountryName* = 'USA'. We can interpret $Query_2$ as following either the path *Country* − *City* or the path *City* − *Province* − *Country*. The former means that "Find the name of the capital of a country whose name is 'USA'; the latter "Find the names of the cities that belong to a country whose name is 'USA'. Therefore, the meaning of $Query_2$ is that "Find the name of the capital of a country whose name is 'USA' or the names of the cities who belong to a country whose name is 'USA'. However, if we interpret $Query_2$ using the maximal object, the where clause of the query becomes "**where** *Country*. *CountryName* = 'USA' and *Country*. *CityId* = *City*. *CityId* and *City*. *ProvinceId* = *Province*. *ProvinceId* and *Province*. *CountryId* = *Country*. *CountryId*", which means that "Find the name of the city that belongs to a country whose name is 'USA' and is the capital of a country whose name is 'USA" This way, the equality condition, i.e., "the name of the city which belongs to a country is the same as that of the capital", which is an unintentional condition, is added to the interpretation of the query because of the cycle. This is contrary to the user's intention. □

Example 2. Consider the purchase database in Fig. 5(a). As shown in Fig. 5(b), the schema graph of the database has a (undirected)

Algorithm FindMaixmalObjectPlusSet
Input: *MOset*: the set of maximal objects
Output: *MOPset*: the set of maximal objects+
Algorithm:

Step 1. /* Initialize data structures. */
  1.1. $MOPset := \{\}$ /* the set of maximal objects+ */
Step 2. /* For each maximal object *mo* in *MOset*, remove a cycle. */
  2.1. For each $mo \in MOset$
    2.1.1. If *mo* has a cycle *CY*
      2.1.1.1. If *CY* does not contain an MVD connection
        2.1.1.1.1. For each $PK\text{-}FK_{ij}$ in *CY*
          2.1.1.1.1.1. $mop := \{mo.R_{set}, mo.PK\text{-}FK_{set} - \{PK\text{-}FK_{ij}\}\}$
          2.1.1.1.1.2. $MOPset := MOPset \bigcup \{mop\}$
      2.1.1.2. Else if *CY* has an MVD connection *MC*
        2.1.1.2.1. For each $PK\text{-}FK_{ij}$ participating in *MC*
          2.1.1.2.1.1. $mop := \{mo.R_{set}, mo.PK\text{-}FK_{set} - \{PK\text{-}FK_{ij}\}\}$
          2.1.1.2.1.2. $MOPset := MOPset \bigcup \{mop\}$
    2.1.2. Else /* If *mo* has no cycles */
      2.1.2.1. $MOPset := MOPset \bigcup \{mo\}$
Step 3. /* If some element in *MOPset* has a cycle, run this algorithm recursively. */
  3.1. If some element in *MOPset* has a cycle
    3.1.1. $MOPset := FindMaximalObjectPlusSet(MOPset)$
Step 4. /* Return the set of maximal objects+. */
  4.1. return *MOPset*

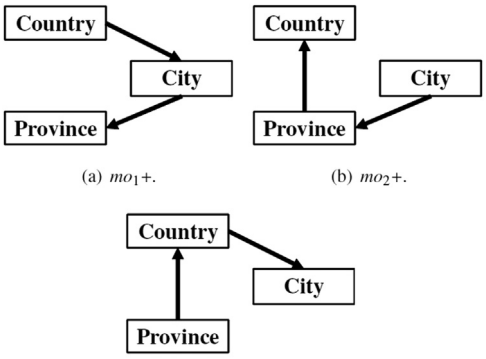
Fig. 7. Algorithm *FindMaximalObjectPlusSet*.

cycle and one MVD connection. Since the set of all relations in the database has the lossless join property indicated by FDs, we obtain a maximal object mo_1: ({*Customer*, *Nation*, *Item*, *Orders*}, {*Customer* → *Nation*, *Item* → *Nation*, *Orders* → *Item*, *Orders* → *Customer*}), which contains the set of all relations and all PK-FK relationships. Here, mo_1 has a cycle.

Consider a query $Query_3$: **retrieve** *Nname* **where** *Cname* = 'Smith'. We can interpret $Query_3$ as following either the join path *Customer* − *Nation* or the join path *Customer* − *Orders* − *Item* − *Nation*. The former means that "Find the name of the country that a customer whose name is 'Smith' belongs to"; the latter "Find the name of countries that items that a customer whose name is 'Smith' orders belong to". However, if we interpret $Query_3$ using the maximal object, the where clause of the query becomes "**where** *Customer*. *Cname* = 'Smith' and *Customer*. *Nid* = *Nation*. *Nid* and *Customer*. *CountryId* = *Orders*. *CountryId* and *Orders*. *Iid* = *Item*. *Iid* and *Item*. *Nid* = *Nation*. *Nid*", which means that "Find the name of a country that a customer whose name is 'Smith' belongs to and that the items ordered by a customer whose name is 'Smith' belongs to". This way, the equality condition, i.e., "The country that an item belongs to is the same as the country that a customer belongs to", which is an unintentional condition, is added to the interpretation of the query because of the cycle. This is contrary to the user's intention.□

3. An acyclic semantic structure on the universal relation model

In Section 3.1, we define a new semantic structure, maximal object+, that completely removes cycles in a universal relation. We also propose an algorithm to find the maximal objects+. In Section 3.2, we describe an algorithm that interprets the queries using a maximal object+.



(a) mo_1+. (b) mo_2+.

(c) mo_3+.

Fig. 8. Maximal objects+ for the geographic database.

3.1. Maximal object+

Definition 3. In a database D, we define a set of relations and PK-FK relationships ($\{R_i, 1 \le i \le l\}$, $\{PK - FK_{ij},\ 1 \le i, j \le l\}$) as a *maximal object+* and denote it as $MOP(D)$ if it satisfies the following four conditions: (1) $mo \in MOset(D)$ AND $\{R_i, 1 \le i \le l\} = mo.R_{set}$ AND $\{PK - FK_{ij},\ 1 \le i, j \le l\} \subseteq mo.\ PK - FK_{set}$. (2) $G' = G(\{V(R_i)|R_i \in \{R_i, 1 \le j \le l\}\}, \{E(PK{-}FK_{ij})|PK{-}FK_{ij} \in \{PK{-}FK_{ij}, 1 \le i, j \le l\}\})$ does not have an MVD connection. (3) G' is acyclic. (4) G' is connected. □

In Definition 3, to guarantee that the maximal object+ satisfies the FD-based losslessness requirement, we add Conditions 1, 2, and 4. To guarantee that the maximal object+ satisfies the acyclic requirement, we add Condition 3. Condition 3 lets a maximal object+ does not contain directed cycles in the schema graph such as the one in the schema graph of geographic database in Fig. 6(b).

Fig. 7 shows the *FindMaximalObjectPlusSet* algorithm that finds a set of maximal objects+ for the schema graph of a relational database. The algorithm finds maximal objects+ by removing the cycles and MVD connection in maximal objects. The algorithm takes the set of maximal objects, *MOset*, as the input. It returns the set of maximal objects+, *MOPset*, as the output. In Step 1, we initialize the set of maximal objects+, *MOPset*. In Step 2, we remove a cycle in each maximal object. 1) If the maximal object has a cycle that does not contain an MVD connection, we generate maximal objects+ by removing each PK-FK relationship in the cycle in the maximal object. 2) If the maximal object has a cycle that contains an MVD connection, we generate maximal objects+ by removing each PK-FK relationship participating in the MVD connection of the maximal object. In the other cases, each maximal object itself becomes a maximal object+. We store the generated maximal objects+ in *MOPset*. In Step 3, if an element in *MOPset* has a cycle (that is, if the maximal object has two or more cycles), we call the algorithm recursively. In Step 4, we return *MOPset*.

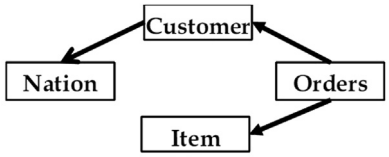
Lemma 2 proves that a maximal object+ defined in Definition 3 satisfies the acyclic requirement and the FD-based losslessness requirement in Section 2.

Lemma 2. *A maximal* object+ *is acyclic, and the set of relations in a maximal* object+ *has the lossless join property indicated by FDs with respect to F.*

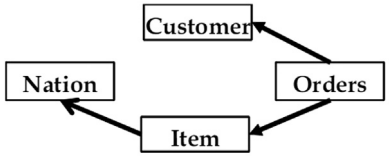
Proof. We examine all the cases in Table 2. In Cases 1, 2, and 5, the maximal object satisfies all the conditions for a maximal object +. In Case 3, we obtain maximal objects+ by removing each PK-FK relationship in the cycle in the maximal object. Hence, each maximal object+ does not a cycle. As the result, we obtain n maximal objects+ when there are n PK-FK relationships in the cycle. In addition, since the maximal object in Case 3 does not have an MVD connection, a maximal object+ does not have an MVD connection, either. In Case 4, we obtain two maximal objects+ by removing each of the two PK-FK relationships participating in an MVD connection in the cycle of the maximal object. Hence, a maximal object+ has neither a cycle nor an MVD connection. Thus, the set of the relations in a maximal object+ has the lossless join property indicated by FDs with respect to F by Lemma 1. □

Example 3. Let us consider the geographic database in Fig. 6(a) once again. The set of relations that satisfies Condition (1) of Definition 3 is {*Country*, *City*, *Province*}; the set of PK-FK relationships that satisfies Condition (1) of Definition 3 is a subset of {*Country* → *City*, *City* → *Province*, *Province* → *Country*}. By removing each PK-FK relationship in this set, we obtain the following sets of PK-FK relationships that satisfy Conditions 2, 3, and 4: {*Country* → *City*, *City* → *Province*}, {*City* → *Province*, *Province* → *Country*}, {*Province* → *Country*, *Country* → *City*}. Hence, maximal objects+ for the geographic database are mo_{1+}, mo_{2+}, and mo_{3+} as shown in Figs. 8(a)–(c), respectively. While the maximal object mo_1 in Fig. 6 has a cycle, maximal objects+ mo_{1+}, mo_{2+}, and mo_{3+} in Fig. 8 do not have a cycle. In addition, the set of relations in each maximal object+ has the lossless join property indicated by FDs with respect to *F* by Lemma 2. □

Example 4. Let us consider the purchase database in Fig. 5(a) once again. The set of relations that satisfies Condition (1) of



(a) mo_1+.



(b) mo_2+.

Fig. 9. Maximal objects+ for the purchase database.

retrieve $t_1.Bname$
where $t_1.Bid = t_2.Bid$ and
$t_1.Cname =$ 'Smith' and $t_2.Cname =$ 'Miller'

Fig. 10. $Query_4$: a USQL join query on the BANK database.

Definition 3 is {*Nation*, *Item*, *Customer*, *Orders*}; the set of PK-FK relationships that satisfies Condition (1) of Definition 3 is a subset of {*Customer* → *Nation*, *Item* → *Nation*, *Orders* → *Item*, *Orders* → *Customer*}. By removing each PK-FK relationship in the MVD connection *Customer* → *Nation* ← *Item* from this set, we obtain the following sets of PK-FK relationships that satisfy Conditions 2, 3, and 4: {*Customer* → *Nation*, *Orders* → *Customer*, *Orders* → *Item*}, {*Item* → *Nation*, *Orders* → *Item*, *Orders* → *Customer*}. Hence, maximal objects+ for the geographic database are mo_{1+}, mo_{2+} as shown in Figs. 9(a) and (b). While the maximal object mo_1 in Fig. 5 has a cycle, maximal objects+ mo_{1+} and mo_{2+} in Fig. 9 do not have a cycle. In addition, the set of relations in each maximal object+ has the lossless join property indicated by FDs with respect to *F* by Lemma 2. □

3.2. Query interpretation using the maximal object+

In this section, we describe a method that interprets USQL queries using a maximal object+. For this, we use the algorithm "*Algorithm 2*" that has been proposed by Maier et al. [15] by substituting the maximal object in *Algorithm 2* with a maximal object+. First, we explain *Algorithm 2* [15]. Then, we show two examples that interpret the queries using a maximal object+.

Maier et al. [15] have used the USQL as the query for universal relations. The USQL specifies only the selection, projection, and join conditions on semantic structures of the relational database [15]. We can join semantic structures to figure out the relationship among the semantic structures. The *tuple variable* is used in USQL queries to specify join conditions between semantic structures. A tuple variable is mapped to one semantic structure. Tuple variables can be used to express the join of two distinct semantic structures or the self-join of the same semantic structure. We call a USQL query that has only one tuple variable, which does not require a join of semantic structures, *USQL selection query*; a USQL query that has two or more tuple variables, which requires a join of semantic structures, *USQL join query*.

An example of a USQL selection query is shown in Fig. 3 in Section 1; an example of a USQL join query is shown in Fig. 10. Fig. 10 shows a USQL join query, $Query_4$, on the BANK database whose schema is shown in Fig. 1(a). $Query_4$ means that "Find the names of the banks where both a customer whose name is 'Smith' and a customer whose name is 'Miller' are related". Here, a tuple variable t_1 (t_2) in $Query_4$ can be mapped to a maximal object mo_1 or mo_2 for the BANK database schema of Fig. 4. Using the combinations that t_1 and t_2 are mapped to the maximal objects, we obtain the following four interpretations. 1) When both t_1 and t_2 are mapped to mo_1, $Query_4$ is interpreted as a self-join of mo_1, which means that "Find the names of the banks that both a customer whose name is 'Smith' and a customer whose name is 'Miller' has an account". 2) When t_1 is mapped to mo_1 and t_2 to mo_2, $Query_4$ is interpreted as a join of mo_1 and mo_2, which means that "Find the names of the banks where a customer whose name is 'Smith' has accounts and a customer whose name is 'Miller' borrows money from". 3) When t_1 is mapped to mo_2 and t_2 to mo_1, $Query_4$ is interpreted by a join of mo_1 and mo_2, which means that "Find the names of the banks where a customer whose name is 'Smith' borrows money from and a customer whose name is 'Miller' has accounts". 4) When both t_1 and t_2 are mapped to mo_2, $Query_4$ is interpreted by a self-join of mo_2, which means that "Find the names of the banks from which both a customer whose name is 'Smith'

Algorithm *Algorithm 2*
Input: (1) Q: a USQL query mentioning tuple variables $t_1, t_2, \cdots, t_k$
(2) $MOset$: a set of maximal objects $m_1, m_2, \cdots, m_q$
Output: E_5: an algebraic expression
Algorithm:
Step 1. /* For each tuple variable t_i in Q, find the set of maximal objects, M_i, that contains $\{B \mid t_i.B$ appears in $Q\}$. */
1.1. For each t_i in Q
1.1.1. $X_i := \{B \mid t_i.B$ appears in $Q\}$
1.1.2. $M_i := \{\}$ /* the set of maximal objects that contain all attributes of X_i */
1.1.3. For each m_j in $MOset$
1.1.3.1. $U_j :=$ the union of relation attributes in m_j
1.1.3.2. If $X_i \subseteq U_j$ $M_i := M_i \cup \{m_j\}$
Step 2. /* For each maximal object m_i in $MOset$, construct an algebraic expression J_i that joins the relations in m_i. */
2.1. For each m_i in $MOset$
2.1.1. $J_i :=$ the algebraic expression for the natural join of all the relations in m_i.
Step 3. /* For each tuple variable t_i in Q, construct an algebraic expression K_i by unioning algebraic expressions that represent maximal objects in M_i. */
3.1. For each t_i in Q
3.1.1. $K_i := \bigcup_{m_j \in M_i} J_j$ that maps to m_j
Step 4. /* Construct an algebraic expression E_5 that interprets Q by adding selection, join, and projection conditions into the Cartesian product of all K_is, and then, by finding its minimal equivalent expression */
4.1. $E_2 := K_1 \times K_2 \times \cdots \times K_k$
4.2. Construct E_3 by applying selection and join conditions to E_2.
4.3. Construct E_4 from E_3 by applying the projection condition.
4.4. Construct E_5 from E_4 by finding a minimal equivalent expression [2] that is equivalent to E_4.
Step 5. /* Return the algebraic expression E_5. */
5.1. Return E_5.

Fig. 11. The *Algorithm 2* of Maier et al. [15].

and a customer whose name is 'Miller' borrow money". The meaning of $Query_4$ is the union of the four interpretations. The USQL selection query cannot represent this meaning since it uses only one semantic structure.

Fig. 11 shows the algorithm "*Algorithm 2*" proposed by Maier et al. [15] that interprets the USQL query using the maximal objects. The algorithm gets a USQL query, Q, that has k tuple variables, and a set of q maximal objects, $MOset$, as the input. It returns an algebraic expression E_5, which interprets Q using the maximal objects. In Step 1, for each tuple variable t_i in Q, we find the set of maximal objects containing all the attributes in $\{B|t_i.B$ appears in $Q\}$. In Step 2, for each maximal object m_i, we create an algebraic expression J_i by connecting all the relations in m_i using natural join operations. In Step 3, for each tuple variable t_i, we construct an algebraic expression K_i by unioning the algebraic expression J_j that is mapped to a maximal object m_j in the set of maximal objects M_i for t_i. In Step 4, we construct an algebraic expression E_5 that interprets Q using the maximal objects. First, we create the Cartesian product of all K_i for Q. Next, we apply the selection, projection, and join conditions of Q to the Cartesian product. Finally, we construct a minimal equivalent expression [2] E_5 for Q. In Step 5, we return the minimal expression E_5.

Example 5. Let us interpret a USQL selection query $Query_1$ as shown in Fig. 3 in Section 1 on the BANK database using Algorithm 2. We assume that there is only one implicit tuple variable t_1 in $Query_1$. In Step 1, X_1 becomes $\{Bname, Cname\}$. Since both mo_1 and mo_2 in Fig. 4 contain $\{Bname, Cname\}$, M_1 becomes $\{mo_1, mo_2\}$. In Step 2, J_1 becomes $Customer \bowtie Account \bowtie Bank$; J_2 $Customer \bowtie Loan \bowtie Bank$. In Step 3, K_1 becomes $J_1 \cup J_2$. In Step 4, E_2 becomes K_1; E_3 becomes $\sigma_{cname='smith'}(K_1)$; E_4 becomes $\Pi_{Bname}(\sigma_{cname='smith'}(K_1))$; E_5 becomes $\Pi_{Bname}(\sigma_{cname='smith'}(Customer \bowtie Account \bowtie Bank)) \cup \Pi_{Bname}(\sigma_{cname='smith'}(Customer \bowtie Loan \bowtie Bank))$. In Step 5, E_5 is returned. □

Example 6. Let us interpret a query on the geographic database in Fig. 6(a) in Section 2 using a maximal object+. Fig. 8 shows three maximal objects+ for the geographic database: mo_{1+}, mo_{2+}, and mo_{3+}. Let us consider a query $Query_5$: **retrieve** *t*. *CityName* **where** *t*. *CountryName*='USA' in Example 1 once again. Using a maximal object+, we can interpret $Query_5$ as "Find the name of the capital of a country whose name is 'USA' or the names of the cities that belong to the country whose name is 'USA'", which is the union of the interpretation using mo_{1+} or mo_{3+} and the one using mo_{2+}. The where clause of the interpreted query becomes "**where** (*Country*. *Ciid* = *City*. *Ciid* and *Country*. *CountryName* = 'USA') or (*City*. *Pid* = *Province*. *Pid* and *Province*. *Cid* = *Country*. *Cid* and *Country*. *CountryName* = 'USA')". The meaning of the interpreted query is the same as the original meaning that is intended by the user while it is not when maximal objects are used instead (as explained in Section 2). The reason is that the maximal object+ does not have a cycle, while it exists in the maximal object. □

Example 7. Let us interpret a query on the purchase database in Fig. 5(a) in Section 2 using a maximal object+. Fig. 9 shows two maximal objects+ for the purchase database: mo_{1+} and mo_{2+}. Let us consider a query $Query_6$: **retrieve** *t*. *Nname* **where** *t*. *Cname* = 'Smith' in Example 2 once again. Using a maximal object+, we interpret the query as "Find the name of a country that a customer whose name is 'Smith' belongs to or that items that a customer whose name is 'Smith' orders belong to'", which is the union of the interpretation using mo_{1+} and the one using mo_{2+}. The where clause of the interpreted query becomes '**where** (*Customer*. *Nid* = *Nation*. *Nid* and *Customer*. *Cname* = 'Smith') or (*Customer*. *Cid* = *Orders*. *Cid* and *Orders*. *Iid* = *Item*. *Iid* and *Item*. *Nid* = *Nation*. *Nid* and *Customer*. *Cname* = 'Smith')". The meaning of the interpreted query is the same as the original meaning that is intended by the user while it is not when maximal objects are used instead (as explained in Section 2). The reason is that the maximal object+ does not have a cycle, while it exists in the maximal object. □

4. Related work

Fagin et al. [6] have defined the universal relation under the assumption that there is no cycle in the database schema graph. For the cases where there are cycles in the database schema graph, existing methods to remove the cycle are classified into the following two categories: 1) the relation renaming method [6] and 2) the semantic structure method [15,17,21]. The former renames a relation in a cycle; the latter proposes new semantic structures to divide the schema graph in the unit of the semantic structure.

Fagin et al. [6] have proposed a relation renaming method. It removes a cycle according to the following steps. 1) The method selects a relation in the cycle and renames it as two independent relations with different names. It also renames the attributes in the renamed relation. 2) The method identifies two relations connected to a relation on a cycle to be renamed and connects each of them to each different renamed relation after renaming it. This method has the following drawbacks. 1) The database schema obtained by renaming relations may not be unique [16]. 2) Users must remember the names of the attributes in the renamed relation. Hence, the more complex the database schema is, the lower the usability of the database is.

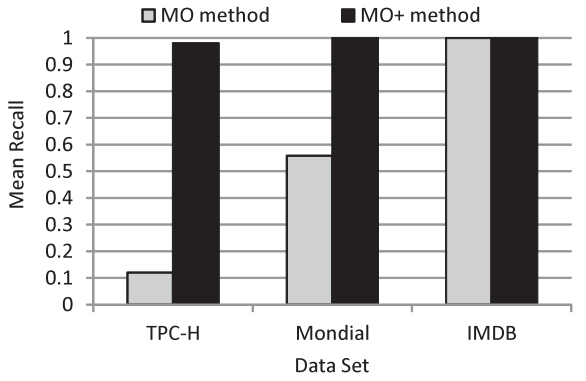
There are three semantic structure methods: 1) the maximal object [15], 2) Context [21], and 3) the maximal join tree [17]. Although the maximal object has been proposed to remove the cycles in universal relations, it fails to remove cycles completely as explained in Section 2. Context is a maximal structure of relations that has the lossless join property indicated by FDs or MVDs and the structure does not have cycles. Although Context does not have cycles, it allows a set of relations that has the lossless join property indicated by MVDs. Hence, it has a drawback of providing meaningless results such as the Cartesian product results when it interprets USQL queries. The maximal join tree [17] is a maximal tree of relations that has the lossless join property. Although Mason et al. [17] remove cycles by defining the maximal join tree as a tree, the maximal join tree has the following drawbacks compared to a maximal object+: 1) they do not provide the theoretical foundation on the definition and usability of the maximal join tree. 2) They do not distinguish the difference between the meaning of the lossless join property indicated by FDs and that indicated by MVDs.

In the big data era, the keyword query is becoming a useful tool to search data since most of the data contains text data nowadays. Webpage big data contain lots of text data with or without a schema. Using a keyword query on webpage data has been actively studied for decades [4,5,9]. Graph big data such as the RDF data contain lots of (short) text data. Keyword queries on graph data can be used to find graph substructures that have specified keywords [11,25]. Meanwhile, relational big data have more and more text data than they had before since the applications for storing text data in relational database have become prevalent [13]. Keyword queries on graph data can be used to find trees of tuples that collectively contains specified keywords [1,14].

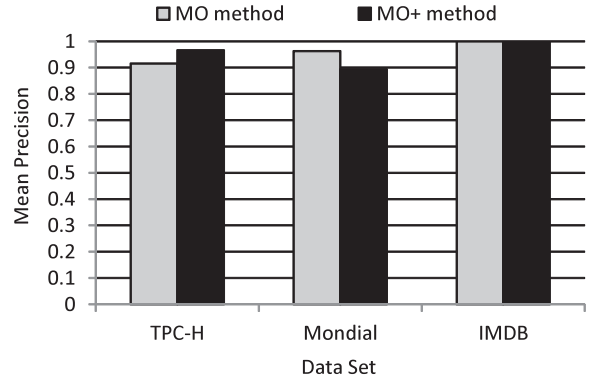
Techniques for processing keyword queries naturally employ USQL techniques. Semantic units can be used to process keyword queries, which consists of a set of keywords, on the relational databases. Like a USQL query, a user can create a keyword query even though the user does not know the details of the database schema. However, unlike a USQL query, a user does not need to specify target attributes for the selection or projection conditions. Since a keyword can match values of multiple attributes, a keyword query corresponds to a set of USQL queries. DBXplorer [1] models a keyword query on a relational database as a set of USQL selection queries on the universal relation and uses the universal relation as a semantic unit for interpreting a keyword query. For a keyword query, DBXplorer finds a set of USQL selection queries matching the keyword query, converts the USQL selection queries into SQL queries, and then, obtains the keyword query results by evaluating the SQL queries. However, since it uses the universal relation as a semantic unit, it cannot properly interpret the user intention when there are cycles in the database schema. Mason et. al. [17] models a keyword query on a relational database as a set of USQL selection or join queries on the universal relation and uses the maximal join tree as a semantic unit for efficiently interpreting a keyword query. SRT-Rank [8] uses the Strongly Related Tree (SRT) as a

Table 3
The statistics of the experimental datasets.

Dataset	Size	# relations	# tuples
TPC-H	100 MB	8	866,602
Mondial	9 MB	28	17,115
IMDB	516 MB	6	1,673,074



(a) Mean recall.



(b) Mean precision.

Fig. 12. The mean recall and mean precision as the query processing methods and the datasets are varied.

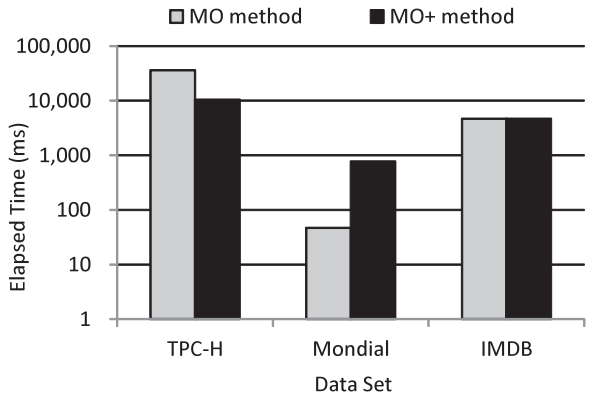


Fig. 13. The actual query time as the query processing methods and the datasets are varied.

semantic unit for effectively ranking the keyword query results. SRT is different from a maximal object+ (maximal join tree) since SRT is a semantic structure defined on the query graph while the maximal object+ is one defined on the schema graph.

Ramon Lawrence and Ken Barker proposed a semantic query language with which users can compose queries by using the semantics of a database instead of the structure of a database [10]. Their proposed method represents the semantics for the given database as a context view, and then, lets a user to specify queries on the context view. Their proposed method determines the join path for the given query using a join graph and a depth-first search based algorithm.

Semantic units can be used to process queries on a semantic web [23]. In semantic web, the resource definition framework (RDF) data model is used to represent data. RDF data model describes data as a form of a triple (subject, predicate, object). The triple denotes that the subject has the predicate relationship with the object. If we map the RDF data to relational data, we can find semantic units in the relational schema of mapped data. By utilizing the semantic units identified, we can process queries on the semantic web. Mapping the RDF data to relational data has been actively studied in the literature [3,19,20,22].

5. Performance evaluation

5.1. Experimental data and environment

In this section, we compare the effectiveness and efficiency of the USQL query processing method based on a maximal object+ (in short, the MO+ method) with that of the existing USQL query processing method based on the maximal object (in short, the MO method). We do not compare Context [21] and maximal join tree [17] with the MO+ method since it may contain meaningless results such as the Cartesian product results as indicated in Section 4. The MO+ (MO) method translates a USQL query using the maximal objects+ (maximal objects) into SQL queries, and then, obtains query results by evaluating those SQL queries.

We perform experiments on the synthetic and real datasets. We use TPC-H [24] as a synthetic dataset and use Mondial [18] and IMDB [7] as real datasets. We generate the dataset for TPC-H by using the TPC-H DBGen tool [24]. Table 3 shows the statistics for the datasets. The TPC-H dataset represents retail information such as customers, suppliers, and parts and consists of eight relations. The Mondial dataset represents geographical information and consists of 28 relations. The IMDB dataset represents information on movies and consists of six relations. The database schema for the TPC-H dataset has one undirected cycle and that for Mondial dataset has more than ten undirected and directed cycles while that for the IMDB dataset has none.

We use the following queries for the experiments. For each dataset, we use 50 queries generated by eight graduate students of Computer Science Department of the authors' institution. To minimize bias with queries, we choose the students who do not participate in this project. They write down search intentions, which specify the information they want to retrieve, and USQL queries to represent the search intentions. An example of the search intentions is "Find the name of a country that a customer whose name is 'Smith' belongs to". The USQL query to represent the search intention is "retrieve t1.n_name where t1.c_name = 'Smith'". In Appendix A, we show five representative USQL queries generated for each dataset.

We measure both effectiveness and efficiency with an emphasis on the former. We use the actual query time to measure the efficiency. To measure the effectiveness of each method, we define the *correct answers* for a USQL query as the results obtained by evaluating the SQL queries corresponding to the users' search intention for the USQL query. We use mean precision and mean recall to calculate the effectiveness. The mean precision (recall) for a set of queries is the average of precisions (recalls) for all the queries in the set.

We have performed all experiments on a machine with Intel Core i5 760 2.80 GHz CPU and 4GB RAM running Linux Fedora Core 14. We use MySQL 5.5.19.

5.2. Experimental results

Fig. 12 shows the mean recall and mean precision of the MO and the MO+ methods. The mean recall of the MO+ method is 8.15, 1.79, and 1.00[5] times that of the MO method for the TPC-H, Mondial, and IMDB datasets, respectively. The mean precision of the MO+ method is similar to that of the MO method for the TPC-H, Mondial, and IMDB datasets. That is, the mean precision of the MO+ method is 0.97, 0.89, and 1.00, and that of the MO method is 0.92, 0.96, and 1.00 for the TPC-H, Mondial, and IMDB datasets, respectively.[6]

The MO+ method produces more correct answers for the TPC-H and Mondial datasets than the MO+ method does due to the following reason. The MO+ method gives near perfect recall since it interprets a USQL query on each possible semantically meaningful alternative path that connects the relations containing attributes used in the query.[7] In contrast, the maximal objects for

[5] The mean recall of the MO+ method is the same as that of the MO method for IMDB dataset.

[6] If the method generates no result for a given query whose correct answer exists, we can either set the precision of the method for the query to 1.0 or ignore the query when calculating the mean precision. Here, we use the latter approach, which is advantageous to the MO method. If we used the former, the mean precision of the MO+ method would be 2.35, 1.33, and 1.00 times that of the MO method for the TPC-H, Mondial, and IMDB datasets, respectively. (The mean precision of the MO+ method is 0.95, 0.89, and 1.00; that of the MO method is 0.40, 0.67, and 1.00 for the TPC-H, Mondial, and IMDB datasets, respectively.)

[7] The MO+ method finds semantically meaningful alternative paths that do not contain an MVD connection. If the intention of the user for the given USQL query is to find a path that contains an MVD connection, the MO+ method fails to generate correct answers. However, as we observe that the search intention of only one query out of 50 queries for TPC-H dataset is to find a path that contains an MVD connection, queries with such search intention occur very rarely in practice. The only one query for the TPC-H dataset is the USQL query "retrieve t1.nation.n_name where t1.customer.c_name = 'A' and t2.supplier.s_name = 'B' and t1.customer.c_nationkey = t2.supplier.s_nationkey" with a search intention "Find the name of nation where both customers whose name is 'A' and suppliers whose name is 'B' belong". The search intention finds a path *Customer* → *Nation* ← *Supplier*, which has an MVD connection defined in Definition 1. This MVD connection is excluded in the maximal objects+ for TPC-H.

the TPC-H and Mondial datasets have a cycle while the maximal objects+ do not. Consequently, as explained in Sections 2 and 3.2, the MO method adds an unintentional condition to the interpretation of a USQL query on the TPC-H or Mondial datasets because of the cycle, excluding a large portion of correct answers to be produced. Both the MO and the MO+ methods generate the same correct answers for the IMDB dataset. Since there is no cycle in the schema graph, the maximal objects+ are the same as the maximal objects.

Fig. 13 shows the actual query time of the MO and the MO+ methods. The results of the experiments show that the relative efficiency depends on the characteristics of the cycles in the database schema graph. Specifically, the MO+ method is 3.45, 0.06, and 1.00 times faster than the MO method for the TPC-H, Mondial, and IMDB datasets, respectively. In general, the MO+ method generates more SQL queries than the MO method does because it generates a query for each possible semantically meaningful alternative path while the MO method generates queries that require satisfying all the paths in the cycle at the same time due to the unintentional equality condition. For the TPC-H dataset, the MO+ method is still faster than the MO method since the overhead of processing the unintentional condition is significant joining all the relations in the cycle. On the other hand, for the Mondial dataset, the MO+ method is slower than the MO method since it generates far more SQL queries than the MO method does due to the complex schema graph involving many cycles. For the IMDB dataset, the actual query time of the MO+ method is almost the same as that of the MO method since both the maximal objects and the maximal objects+ for the IMDB dataset are the same. In summary, we conclude that both methods have comparable efficiency.

6. Conclusions

The universal relation model has been proposed to improve the usability of relational databases [6]. The maximal object has been proposed to implement the universal relation in a semantically correct way [15]. The maximal object is not supposed to include a cycle since it can make query interpretation ambiguous [15]. In this paper we have identified for the first time that the maximal object may include a cycle, and have proposed a new semantic structure, called maximal object+, which completely removes cycles in a universal relation. A maximal object+ is a largest connected acyclic component in the database schema graph where the entire set of relations in the component has the lossless join property indicated by FDs. A maximal object+ only allows the set of relations that has the lossless join property indicated by FDs. It does not allow the set of relations that has the lossless join property not indicated by FDs but indicated by MVDs. The salient points of a maximal object+ are as follows. 1) It eliminates the equality condition that the user does not intend. 2) It interprets the query so as not to generate meaningless results by a lossy join since it consists of the set of relations that has the lossless join property. 3) It does not generate Cartesian product results that are incurred by MVDs since it allows only the lossless join property indicated by FDs. In Lemma 2, we have shown that the maximal object+ is acyclic and that the set of the relations in a maximal object+ always has the lossless join property indicated by FDs.

We have performed experiments on the synthetic and real datasets. As a result, we show that our method significantly outperforms the one based on the maximal object in terms of mean recall when a cycle exists in a maximal object while maintaining comparable efficiency. Specifically, our method improves the mean recall of the query processing method by up to 8.15 times for the dataset whose schema involves cycles.

A maximal object+ can be utilized to improve the effectiveness of query interpretation by using it as a structure to represent a semantically close relationship among relations. We have theoretically shown that a maximal object+ interprets a query for each possible semantically meaningful alternative path by completely removing cycles from the universal relation.

In the further study, we are going to take the following two points into account: 1) types of cycles in the database schema graph and 2) other measures to complement a maximal object+ to fit a user intent. There are different types of cycles in the database schema graph: self-cycle, directed cycle, and undirected cycle. Taking types of cycles into account will help interpret an USQL query to fit user intent. In addition, other measures related to query workloads or semantic information can be utilized to complement a maximal object+.

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by Korean Government (MSIP) (No. 2016R1A2B4015929).

Appendix A. USQL queries and search intensions used in the experiments

Figs. 14–16 show five representative USQL queries and their search intentions for each dataset used in the experiments.

• Query1
(USQL query) retrieve t1.n_name where t1.c_name = 'Customer#000010326'
(Search intention) Find the name of the nation where the customer 'Customer#000010326' lives.

• Query2
(USQL query) retrieve t1.p_name where t1.c_name = 'Customer#000001012'
(Search intention) Find the name of a part that the customer 'Customer#000001012' orders.

• Query3
(USQL query) retrieve t1.partsupp.ps_availqty where t1.supplier.s_name = 'Supplier#000000903' and t1.part.p_name = 'steel white brown peru aquamarine'
(Search intention) Find the available quantity of the 'steel white brown peru aquamarine' part that the supplier 'Supplier#000000903' supplies.

• Query4
(USQL query) retrieve t1.supplier.s_name where t1.part.p_brand = 'Brand#32' and t2.part.p_brand = 'Brand#44' and t1.supplier.s_suppkey = t2.supplier.s_suppkey
(Search intention) Find the name of a supplier who supplies both the part whose brand is 'Brand#32' and the part whose brand is 'Brand#44.'

• Query5
(USQL query) retrieve t1.supplier.s_name where t1.customer.c_name = 'Customer#000014171' and t2.customer.c_name = 'Customer#000009394' and t1.supplier.s_suppkey = t2.supplier.s_suppkey
(Search intention) Find the name of a supplier who receives orders from both the customer 'Customer#000014171' and the customer 'Customer#000009394.'

Fig. 14. Five USQL queries and their search intentions for the TPC-H dataset.

• Query1
(USQL query) retrieve t1.religion.name where t1.country.name = 'Mexico'
(Search intention) Find the name of a religion that prevails in the country 'Mexico.'

• Query2
(USQL query) retrieve t1.country.name where t1.language.name = 'Russian';
(Search intention) Find the name of a country whose language is 'Russian.'

• Query3
(USQL query) retrieve t1.continent.name where t1.country.name = 'Turkey'
(Search intention) Find the name of the continent that the country 'Turkey' belongs to.

• Query4
(USQL query) retrieve t1.country.name where t2.country.name = 'Japan' and t1.economy.gdp > t2.economy.gdp
(Search intention) Find the name of a country whose GDP is larger than that of the country 'Japan.'

• Query5
(USQL query) retrieve t1.religion.name where t1.country.name = 'China' and t2.country.name = 'Thailand' and t1.religion.name = t2.religion.name
(Search intention) Fine the name of a religion that prevails in both the country 'China' and the country 'Thailand.'

Fig. 15. Five USQL queries and their search intentions for the Mondial dataset.

• Query1
(USQL query) retrieve t1.movie.title where t1.person.name = 'Robbins, Tim'
(Search intention) Find the title of a movie in which a person whose name is 'Washington, Denzel' plays as an actor.

• Query2
(USQL query) retrieve t1.person.name where t1.movie.title = 'Moneyball'
(Search intention) Find the name of a person who plays in the movie having the title 'Moneyball.'

• Query3
(USQL query) retrieve t1.character.name where t1.movie.title = 'The Shawshank Redemption'
(Search intention) Find the name of a character who appears in the movie having the title 'The Shawshank Redemption.'

• Query4
(USQL query) retrieve t1.person.name where t1.role.name = 'director' and t2.role.name = 'actor' and t1.person.id = t2.person.id
(Search intention) Find the name of a person who is an actor and director.

• Query5
(USQL query) retrieve t1.person.name where t1.movie.title = 'The Dark Knight' and t2.movie.title = 'Brokeback Mountain' and t1.person.id = t2.person.id
(Search intention) Find the name of a person who plays in both the movie 'The Dark Knight' and the movie 'Brokeback Mountain.'

Fig. 16. Five USQL queries and their search intentions for the IMDB dataset.

References

[1] S. Agrawal, S. Chaudhuri, G. Das, DBXplorer: a system for keyword-based search over relational databases, in: Proceedings of the IEEE International Conference on Data Engineering(ICDE), San Jose, CA, USA, 2002, pp. 5–16.
[2] A.V. Aho, Y. Sagiv, J.D. Ullman, Efficient optimization of a class of relational expressions, ACM Trans. Datab. Syst. (TODS) 4 (4) (1979) 435–454.
[3] S. Bagui, J. Bouressa, Mapping RDF and RDF-schema to the entity relationship model, J. Emerg. Trends Comput. Inform. Sci. 5 (12) (2014) 953–961.
[4] S. Brin, L. Page, Reprint of: the anatomy of a large-scale hypertextual web search engine, Comput. Netw. 56 (18) (2012) 3825–3833.
[5] H. Chu, M. Rosenthal, Search engines for the world wide web: a comparative study and evaluation methodology, in: Proceedings of the Annual Meeting of American Society for Information Science, Baltimore, MD, USA, 1996, pp. 127–135.
[6] R. Fagin, A. Mendelzon, J.D. Ullman, A simplified universal relation assumption and its properties, ACM Trans. Datab. Syst. (TODS) 7 (3) (1982) 343–360.
[7] IMDb.com, Internet Movie Database, 2016. ⟨http://www.imdb.com⟩. (Accessed 21 February 2016).
[8] I. Kim, K. Whang, H. Kwon, SRT-rank: ranking keyword query results in relational databases using the strongly related tree, IEICE Trans. Inform. Syst. (9) (2014) 2398–2414 (E97-D).
[9] S. Lawrence, Context in web search, IEEE Data Eng. Bull. 23 (3) (2000) 25–32.
[10] R. Lawrence, K. Barker, Querying relational databases without explicit joins, Concept. Model. New Inform. Syst. Technol. 2465 (2002) 278–291.
[11] W. Le, F. Li, A. Kementsietsidis, S. Duan, Scalable keyword search on large RDF data, IEEE Trans. Knowl. Data Eng. 26 (11) (2014) 2774–2788.
[12] F. Leymann, A survey of the universal relation model, Data Knowl. Eng. 4 (4) (1989) 305–320.
[13] Y. Luo, X. Lin, W. Wang, X. Zhou, SPARK: top-k keyword query in relational databases, in: Proceedings of the International Conference on Management of Data, ACM SIGMOD, Beijing, China, 2007, pp. 115–126.
[14] Y. Luo, W. Wang, X. Lin, X. Zhou, J. Wang, K. Li, Spark2: top-k keyword query in relational databases, IEEE Trans. Knowl. Data Eng. 23 (12) (2011)

1763–1780.
[15] D. Maier, J.D. Ullman, Maximal objects and the semantics of universal relation databases, ACM Trans. Datab. Syst. (TODS) 8 (1) (1983) 1–14.
[16] D. Maier, D. Rozenshtein, J. Stein, Representing roles in universal scheme interfaces, IEEE Trans. Softw. Eng. 11 (7) (1985) 644–652.
[17] T. Mason, L. Wang, R. Lawrence, AutoJoin: providing freedom from specifying joins, in: Proceedings of the International Conference on Enterprise Information Systems (ICEIS), Miami, USA, vol. 5, 2005, pp. 31–38.
[18] W. May, Information Extraction and Integration with Florid: The Mondial Case Study, Universität Freiburg, Technical Report 131, Freiburg im Breisgau, Germany, 1999.
[19] Z. Pan, J. Heflin, DLDB: Extending Relational Databases to Support Semantic Web Queries, Practical and Scalable Semantic Systems, 2003.
[20] S. Ramanujan, A. Gupta, L. Khan, S. Seida, B. Thuraisingham, A relational wrapper for RDF reification, in: Proceedings of the Third IFIP WG 11.11, International Conference on Trust Management (IFIPTM), West Lafayette, USA, 2009. pp. 196–214.
[21] R.D. Semmel, J. Mayfield, Automated query formulation using an entity-relationship conceptual schema, J. Intell. Inform. Syst. 8 (3) (1997) 267–290.
[22] W. Teswanichand S. Chittayasothorn, A transformation of RDF documents and schemas to relational databases, in: Proceedings of the IEEE PacificRim Conference on Communications, Computers, and Signal Processing, Victoria, B.C., Canada, 2007, pp. 38–41.
[23] B. Tim, J. Hendler, O. Lassila, The semantic web, Sci. Am. 284 (5) (2001) 28–37.
[24] TPC, TPC-H, 2016. ⟨http://www.tpc.org/tpch⟩. (Accessed 21 February 2016).
[25] T. Tran, H. Wang, S. Rudolph, P. Cimiano, Top-k exploration of query candidates for efficient keyword search on graph-shaped (RDF) data, in: Proceedings of the IEEE International Conference on Data Engineering(ICDE), Shanghai, China, 2009, pp. 405–416.
[26] J.D. Ullman, Principles of Database and Knowledge-Base Systems 1, Computer Science Press, New York, 1998.

In-Joong Kim received the B.S. in computer engineering from Hongik University in 2004 and the M.S. in computer science from Korea Advanced Institute of Science and Technology (KAIST) in 2006, and the Ph.D. in computer science from KAIST in 2015. His research interests include information retrieval, search engines, and big data analytics.

Kyu-Young Whang earned his Ph.D. from Stanford University in 1984. From 1983 to 1991, he was a Research Staff Member at IBM T. J. Watson Research Center. In 1990, he joined KAIST, where he currently is a Distinguished Professor at Department of Computer Science. His research interests encompass database systems/storage systems, search engines, object-oriented databases, geographic information systems, big data management, and data mining. He was the general chair of VLDB2006. He served as an Editor-in-Chief of the VLDB Journal from 2003 to 2009. He is a Fellow of ACM and IEEE. He served as the chair of IEEE Technical Committee on Data Engineering from 2013 to 2014.

Hyuk-Yoon Kwon received the B.S. in computer science and statistics from University of Seoul (UOS) in 2005, the M.S. in computer science from Korea Advanced Institute of Science and Technology (KAIST) in 2007, and the Ph.D. in computer science from KAIST in 2013. His research interests include GIS, information retrieval, and big data analytics.