International Journal of Mathematics and
Computer Science, **15**(2020), no. 1, 41–64

RDF-BF-Hypergraph Representation for Relational Database

Fayed F. M. Ghaleb[1], **Azza A. Taha**[1], **Maryam Hazman**[2], **Mahmoud Abd ElLatif**[3,4], **Mona Abbass**[2]

[1]Department of Mathematics
Faculty of Science
Ain Shams University
Cairo, Egypt

[2]Central Lab. for Agricultural Expert System
Agricultural Research Center
Giza, Egypt

[3]College of Business
University of Jeddah
Jeddah, Saudi Arabia

[4]Faculty of Computers and Artificial Intelligence
Helwan University
Helwan, Cairo, Egypt

email: ataha33@yahoo.com

(Received August 17, 2019, Accepted September 19, 2019)

Abstract

The semantic web is designed to provide machine accessible meaning for its constructs. Semantic web depends on Resource Description Framework (RDF), which is a standard model for data interchange on the Web. The RDF can be represented as graph which is known as RDF_graph data model. One of the goals of the semantic web

Key words and phrases: Hypergraph, Hypergraph Acyclicity, Relational Database Scheme, α-cycle, Semantic Web, RDF.
AMS (MOS) Subject Classifications: 68P15, 05C65, 97R50.
ISSN 1814-0432, 2020, http://ijmcs.future-in-tech.net

is to display the huge quantities of data that is stored in the Relational Databases (RDB) for computer processing. In order to integrate RDB into semantic web applications, RDB should be represented as RDF_graph data model, which is fundamental for developing the semantic web in the recent decade. In the last decades a special class of database schemes, called acyclic database schema was introduced. This class is preferred in order to minimize the time of answering query and decrease its space efficient access paths. Also the result of answering a query will be reduced specially in the case of representing RDB as graph data model. This paper introduces a Backward and Forward hypergrapg (BF-hypergraph) representation for the RDF schema that corresponds to RDB schema. The BF-hypergraph is a suitable model to represent the set of functional dependencies of RDB, since the domain and codomain of the functional dependency may contain more than one attribute. Finally, the paper proposes a model to represent the RDF-BF-hypergraph schema that corresponds to an acyclic/cyclic RDB schema according to its set of functional dependency. A formal representation of RDF_graph schema will preserve the integrity constraints and data semantics. Moreover, a formal representation of RDF_graph schema will facilitate the data conversion and will allow graph connectivity which is essential for querying the RDF using various traversal algorithms.

1 Introduction

The World Wide Web was basically designed for human usage, and in spite of the fact that everything on it is machine-readable, the data is not machine-understandable [1]. The semantic web and semantic web technologies provide a new approach for managing and processing data; the semantic web's main idea is the creation and usage of semantic metadata [2].

The basic data model for writing simple statements about web resources is the Resource Description Framework (RDF). One of the representations of the RDF is the graph data model, RDF_graph, which is a standard model for data interchange on the web. It is utilized for expressing data about data resources on the semantic web to become suitable for processing by applications [3]. Therefore, the good creation of the RDF is the base of the success of the semantic web [4].

One of the essential features of the RDF_graph model is its ability to interconnect resources of the RDF in an extensible way. The basic notions of graph theory like node, edge, path, neighborhood, connectivity, distance, and

degree play a central role in expressing the RDF with graph-like structure.

Introducing graphs as a modeling tool for a connected data has many advantages. Graph structures are visible to the user and permit a natural way of handling applications data. Queries can refer directly to this graph structure. Associated with graphs are graph operations in the query language algebra, like finding shortest paths and determining certain subgraphs using graph traversal algorithms [5, 6].

An active field of research during the last decade is making data hosted in relational databases (RDB) machine understandable to the semantic web [7]. Relational databases should be represented as RDF, in order to integrate relational databases into semantic web applications.

Relational database schemes have been defined by Codd [8] as a collection of table skeletons. These tables can be represented as hypergraphs, where every attribute of a database scheme $\mathbf{R}$ corresponds to a node in a hypergraph H, also every relation scheme R in $\mathbf{R}$ corresponds to a hyperedge in H [9, 10].

In the last decades a class of acyclic database scheme and different degrees of acyclicity has been introduced [11], such as α-acyclicity, β-acyclicity, and γ-acyclicity [10]. The least restrictive degree of hypergraph acyclicity is α-acyclicity and in the literature it has more studies than the other two acyclicity degrees. A database scheme is called α-acyclic if the corresponding hypergraph is [9, 12].

In the case of acyclic hypergraphs database the query optimization becomes easier than in the case of cyclic database, and might be recognized in linear time. Furthermore, acyclic hypergraphs is preferred in order to minimize the time of answering query and decrease its space efficient access paths. Also the result of answering query will be reduced specially in the case of representing RDB as graph data model [13, 14, 15, 16, 17].

In the literature, there have been some contributions to represent the RDB as a graph model. For example, Berners-Lee [18] introduced an informal model approach to represent the relational database as a graph model as follows:

- Each row of a table as a resource of the RDF,
- The RDB column names as an RDF predicates,
- Each cell with a literal value as the object of a data property,
- Each cell with a foreign key constraint as the object of an object property.

Applying the above set of rules automatically generate mappings between RDB and RDF_graph model, which is called a directed mapping. Byrne [19], presented the Simple Knowledge Organization System (SKOS) framework, which is used to transform the Royal Commission on the Ancient and Historical Monuments of Scotland (RCAHMS) thesauri to the semantic web. In SKOS framework the size of the RDF dataset has been reduced by about 2.8 million triples. Reducing graph size makes a colossal difference to performance in terms of storage, loading, and query times, but the result is not a graph according to the mathematical definition.

Most of the academic works that have been proposed to represent the RDB as RDF_graph data model are depending on transforming the data that is stored in the relational model not the schema. Furthermore, none of the existing models represents a formal graph model for the RDF_graph schema or instance. A formal representation of the RDF_graph schema should preserve the integrity constraints to ensure integrity and data semantics, which is captured from a database schema. Moreover, a formal representation of RDF_graph schema should facilitate the data conversion and should allow graph connectivity which is essential for querying the RDF using various traversal algorithms.

This paper introduces a Backward and Forward hypergrapg (BF-hypergraph) representation for the RDF schema that corresponds to RDB schema. The BF-hypergraph is a suitable model to represent the set of functional dependencies of RDB, since the domain and codomain of the functional dependency may contain more than one attribute. Finally, the paper proposes a model to represent the RDF-BF-hypergraph schema that corresponds to an acyclic/cyclic RDB schema according to its set of functional dependency to preserve the integrity constraints and data semantics. Moreover, a formal representation of RDF_graph schema should facilitate the data conversion and should allow graph connectivity which is essential for querying the RDF using various traversal algorithms. The model consists of four steps. In the first step a given database schema is represented as hypergraph by identifying its tables and their attributes. Second step will check for α-cyclicity by detecting the set of α-nodes. If the hypergraph is α-cyclic, then we will treat this cycle(s) to see if it can be removed or not, which is the third step. In the fourth step, an RDF-BF-hypergraph schema will be generated for the resulted acyclic undirected hypergraph in the case if the cycle(s) can be removed or for the original undirected hypergraph of step one. Moreover, two algorithms of the proposed model are introduced. The first algorithm converts a cyclic undirected hypergraph that corresponds to RDB schema

into acyclic one if it is possible, which is used in the third step. The second algorithm generates the RDF-BF-hypergraph schema that corresponds to an acyclic/cyclic RDB schema according to its set of functional dependency, which is used in the fourth step.

The paper is organized as follows: in Section 2, the basic definitions of graphs, RDF_graph, hypergraphs, and relational databases are given. In Section 3, a model to generate the RDF-BF-hypergraph schema is introduced. In Section 4, the result of the two introduced algorithms and a systematic discussion are given, Finally, Section 5, illustrates the conclusion of the paper.

2 Preliminaries

In this section the basic definitions of Graphs, RDF_Graph, hypergraphs, and relational databases are given.

2.1 Graphs and RDF_Graph

Definition 2.1. A graph G is a pair (V, E), where V is a set of nodes, and E is a set of edges. Each edge $e \in E$ is an unordered pairs $\{u, v\}, u, v \in V$ [21]. For example, Figure 1, shows a graph with set of nodes $V = \{A, B, C, D\}$ and set of edges $E = \{e_1, e_2, e_3, e_4, e_5\}$, where $e_1 = \{A, B\}, e_2 = \{B, C\}, e_3 = \{C, D\}, e_4 = \{A, C\}$, and $e_5 = \{B, D\}$.

Definition 2.2. Two edges e_i and e_j are said to be incident if they share a node [21]. For example, the two edges e_1 and e_2 of the graph of Figure 1 are incident.

Definition 2.3. A graph G is a multigraph if multiple edges are permitted between two nodes [21]. For example, Figure 2, shows a multigraph with set of nodes $V = \{A, B, C, D\}$ and set of edges $E = \{e_1, e_2, e_3, e_4, e_5, e_6\}$, where $e_1 = \{A, D\}, e_2 = \{A, C\}, e_3 = \{B, D\}, e_4 = \{B, C\}, e_5 = \{B, C\}$, and $e_6 = \{D, D\}$ and two multiple edges e_4 and e_5.

Definition 2.4. A directed graph or digraph D is an ordered pair (V, E) with V is a set of nodes, and E a set of ordered pairs of nodes, called edges, directed edges, or arrows as shown in Figure 3. An edge $e = (A, B)$ is considered directed from A to B; A is called the head and B is called the tail of the edge [22].

Definition 2.5. A path in a graph $G = (V, E)$ is a sequence of edges $e_1, ..., e_n$ where each edge e_i is incident to e_{i+1}, for $1 \leq i < n$ [21]. A path is said to be simple if $e_i \neq e_j$ for $i, j \leq n, i \neq j$. For example, the simple path from

node A to node D in the graph of Figure 1 is (e_1, e_2, e_3).

The RDF statements are triples <Sub><Pred><Obj>, consisting of three parts a subject <Sub>, a predicate <Pred> and an object <Obj>, where the subject is the resource to be described, the predicate is a property, and the object is a propertys value. For example, The RDF statement <kareem><is_a><doctor>, gives Kareem as the subject of the triple, "is_a" as the predicate of the triple, and doctor as the object of the triple. An RDF_graph T is the set of RDF triples represented as graph. Figure 4, shows the RDF graph data model [20].

Definition 2.6. let T be an RDF triple, the node-edge-labeled multigraph $\delta(T)$ is (V, E, l_v, l_e) where,

1. $V = \{v_x : x \in subj(T) \cup obj(T)\}, l_v(v_x) = x\}$ and
2. $E = \{e_{s,p,o} : (s, p, o) \in T\}$, such that there is an edge e from v_s to $v_o, l_e(e_{s,p,o}) = p$ [21].

For example, Figure 5, shows the node-edge-labeled multigraph for the RDF $T_1 = \{(Picasso, paints, Guernica), (Guernica, type, Paint), (Zapata, type, Paint), (Paints, range, Paint), (Paints, domain, Painter)\}$, where $V = \{v_1, v_2, v_3, v_4, v_5, v_6\}$,$l_v = \{Picasso, Guernica, Paint, Zapata, paints, Painter\}$, $E = \{e_1, e_2, e_3, e_4\}$, and $l_e = \{paints, type, range, domain\}$.
Note that the set of edge labels and node labels might not be disjoint.

Definition 2.7. Two nodes x, y are connected if there exists a path $e_1, ..., e_n$ with $x \in e_1$ and $y \in e_n$. The length of a path is the number of edges it contains [21]. For example, the length of the path (e_1, e_2, e_3) from node A to node D in the graph of Figure 2 is 3.

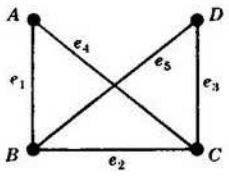


Figure 1: Simple Graph

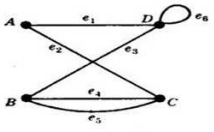


Figure 2: MultiGraph

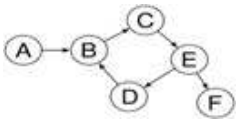


Figure 3: Directed Graph

2.2 Hypergraphs and Relational Databases

Definition 2.8. An undirected hypergraph $H = \{e_1, ..., e_n\}$ is a finite set of non-empty finite sets e_i, which is called its hyperedge (or simply edge), the

set $V(H) = \bigcup_{i=1}^{n} e_i$ of nodes of a hypergraph H, is defined to be the union

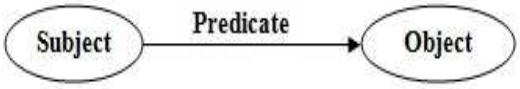


Figure 4: RDF triple

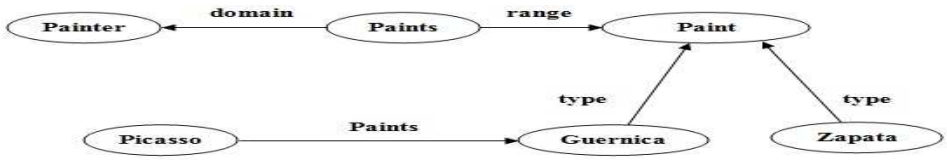


Figure 5: The node-edge-labeled multigraph for the RDF T_1 of definition 2.6

of all its hyperedges [23]. For example, Figure 6, shows a hypergraph with four edges $\{e_1, e_2, e_3, e_4\}$, where $e_1 = \{ABC\}$, $e_2 = \{CDE\}$,$e_3 = \{EFA\}$, and $e_4 = \{ACE\}$ (Note that ABC is abbreviation of $\{A, B, C\}$, etc.), and $V(H) = \{A, B, C, D, E, F\}$.

Definition 2.9. The size of a hypergraph $|H|$ is defined to be the number of edges in it [1]. For example, the size of the hypergraph H in Figure 6 is 4. Also, the size of an edge $|e|$ is defined to be the number of nodes in it. For example, the size of any edge in the hypergraph of Figure 6 is 3.

Definition 2.10. An edge $e \in H$ is said to be a singleton, if $|e| = 1$, i.e., if it contains exactly one node [10].

Definition 2.11. A hypergraph H' is said to be a subhypergraph of a hypergraph H if $H' \subseteq H$ [23]. Note that, the suphypergraph H', in $H' \subseteq H$, is obtained from H by removing edges, and not removing nodes from edges. For example, Figure 7 shows that $H' = \{e_1, e_2, e_3\}$ is subhypergraph of the hypergraph in Figure 6 where $V(H) = V(H')$.

Definition 2.12. let V be a finite set of database attributes, a database scheme $\mathbf{R} = \{R_1, ..., R_p\}$, where each relation schemes R_i is a set of subsets of V, i.e., $R_i = \{A_{i1}, A_{i2}, ..., A_{in_i}\}$, and $i = 1, ..., p$: n_i is the number of the attributes of the relation scheme R_i. Each attribute A_{ij} is associated with a domain D_{ij}. This database scheme can be represented as hypergraph $H = \{B_1, B_2, ..., B_p\}$ where $B_i, i = 1, ..., p$ are its hyperedges such that, $B_i = \bigcup_{j=1}^{n_i} A_{ij}$ and $V(H) = \bigcup_{i=1}^{p} B_i$. For example, in the hypergraph of Figure 9, $H = \{\{S\#, Sname, Status, City\}, \{P\#, Pname, Color, Weight, City\}, \{S\#, P\#, Date, Qty\}\}$. The hypergraph corresponds to the SUPPLIER-PART database schemes [24] of Figure 8, where $\mathbf{R} = \{$*SUPPLIER, PART, SHIPMENT*$\}$. The database $\mathbf{R}$ consists of three relation schemes: a SUP-

PLIER relation scheme with attributes $\underline{S\#}$ (for supplier number), *Sname* (for supplier name), *Status*, and *City*; a $PART$ relation scheme with attributes $\underline{P\#}$ (for part number), *Pname* (for part name), *Color*, $Weight$, and *City*; and a $SHIPMENT$ relation scheme with attributes $\underline{S\#}$, $\underline{P\#}$, $\underline{Date}$, and *Qty* (for quantity).

Definition 2.13. Two edges e and f of a hypergraph are called properly intersecting if $e \nsubseteq f, f \nsubseteq e$ and $e \cap f \neq \emptyset$ [20]. For example, the two edges e_1, e_2 in Figure 6 are properly intersecting.

Definition 2.14. Let H be a hypergraph and $v \in V(H)$, the star $H(v)$, of the node v, is defined to be the set of all edges containing v ,i.e., $H(v) = \{e \in H | v \in e\}$. For example, the star of the node A in the hypergraph H of Figure 6 is $H(A) = \{e_1, e_3, e_4\}$ [23].

Definition 2.15. Let e and f be two properly intersecting hyperedges of a hypergraph H. A sequence $(e = e_1, ..., e_p = f)$, such that $p > 2$ is called a sequence of neighborhoods between e and f, if $e \cap f \subsetneq e_k \cap e_{k+1}$, for $k = 1, ..., p-1$ [25]. For example, (e_1, e_4, e_2) in Figure 6 is a sequence of neighborhoods connecting e_1 and e_2.

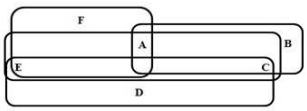


Figure 6: An undirected hypergraph with four edges

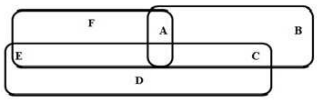


Figure 7: *H' = $\{e_1, e_2, e_3\}$* is subhypergraph of the hypergraph figure 6 and *V(H) = V(H)'*

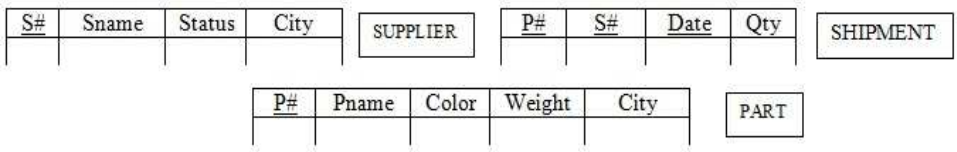


Figure 8: A cyclic SUPPLIER-PART database schemes

Definition 2.16. Let e and f be two properly intersecting hyperedges of a hypergraph H, the two edges e and f are $\alpha-$neighboring, if there is no sequence of neighborhoods between them [23]. For example, the two intersecting edges e_1 and e_2 of the hypergrapgh in Figure 7 are $\alpha-$neighboring, because there is no sequence of neighborhoods connecting e_1 and e_2, while the two intersecting edges e_1 and e_2 of the hypergrapgh in Figure 6 are not

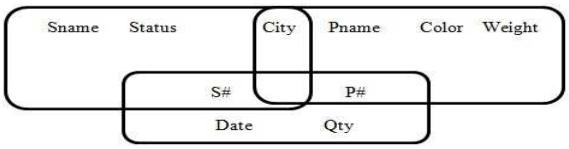


Figure 9: The undirected hypergraph corresponding to the cyclic database schemes of Figure 8

$\alpha-$neighboring, because (e_1, e_4, e_2) is a sequence of neighborhoods connecting e_1 and e_2.

Definition 2.17. An $\alpha-$path in a hypergraph H is a sequence of hyperedges $(e_1, ..., e_p)$, such that $\forall j, 1 \leq j < p; e_j$ and e_{j+1} are $\alpha-$neighboring [25]. For example, the sequence (e_1, e_2, e_3) in Figure 7 is $\alpha-$path.

Definition 2.18. An $\alpha-$cycle in a hypergrph H is an $\alpha-$path $(e_1, ..., e_p)$ such that $p > 3, e_1 = e_p, \nexists 1 \leq a < b < p$, such that $e_a \cap e_{a+1} \subset e_b \cap e_{b+1}$ [25]. For example, the sequence (e_1, e_2, e_3, e_1) in Figure 7 is $\alpha-$cycle.

Definition 2.19. An $\alpha - path(e_1, ..., e_q), q > 2$, is called quasi $\alpha-$cycle (α_Q-cycle) [26] in a hypergraph H if

1. The two edges e_1 and e_q are $\alpha-$neighboring and
2. $\nexists 1 \leq a < b < q$, such that $e_a \cap e_{a+1} \subset e_b \cap e_{b+1}$. For example, the sequence (e_1, e_2, e_3) in Figure 7 is α_Q-cycle.

Definition 2.20. Let H be a hypergraph, the set of all $\alpha-$cycles in H, is the set $\acute{C}_{\alpha-\text{cycle}} = \{C : C \text{ is an } \alpha-\text{cycle in } H\}$, and for every cycle $C = (e_1, ..., e_p) \in \acute{C}_{\alpha-cycle}, p > 3$, then the set $V_\alpha^C(H) = \cup_{i=1}^{p-1} e_i \cap e_{i+1}$ is called the set of $\alpha-$nodes of the $\alpha-$cycle C in a hypergraph H, and the set $V_\alpha(H) = \cup_{C \in \acute{C}_{\alpha-\text{cycle}}} V_\alpha^C(H)$ is called the set of $\alpha - nodes$ of the all cycles $\acute{C}_{\alpha-\text{cycle}}$ in a hypergraph H [26]. For example, $V_\alpha(H) = \{S\#, P\#, City\}$ is the set of $\alpha-$nodes of the hypergraph of Figure 9.

Lemma 2.1. A hypergraph H is $\alpha-$acyclic if and only if it contains no $\alpha-$node [26].

The GYO algorithm [27] was introduced to determine the acyclicity of a hyperhraph $H = \{e_1, ..., e_n\}$ in linear time. The algorithm conveys to two rules:

- **Rule 1:** If an edge e_i contains an isolated node, delete this node from e_i.

- **Rule 2:** Delete edge e_i if there is another edge e_j such that $e_i \subseteq e_j$.

These two rules are applied repeatedly until no rules can be applied by getting either $GYO(H) = \emptyset$ and hence H is acyclic or $GYO(H) \neq \emptyset$ and hence H is cyclic.

Definition 2.21. A directed hypergraph $\overrightarrow{H} = (\overrightarrow{V}, \overrightarrow{E})$, where $\overrightarrow{V}$ is a finite set of nodes and $\overrightarrow{E} \subset \{(T(e), H(e)) : T(e), H(e) \in P(V) \setminus \emptyset \ \& \ T(e) \cap H(e) = \emptyset\}$ is the set of directed edges, where $P(V)$ is the power set of $V, T(e)$ and $H(e)$ are said to be the tail and the head of e respectively. The head and tail represent the set of nodes where the hyperedge ends and starts respectively [28]. It is clear that $|T(e)| > 0, |H(e)| > 0$. A directed hyperedge e is simple if $|T(e)| = 1$ and $|H(e)| = 1$. For example, Figure 10, shows a directed hypergraph $\overrightarrow{H}$ with a set of nodes $\overrightarrow{V} = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7\}$ and set of hyperedges $\overrightarrow{E} = \{e_1, e_2, e_3, e_4\}$, where $e_1 = (\{v_1, v_2\}, \{v_3, v_4\}), e_2 = (\{v_1, v_3\}, \{v_5\}), e_3 = (\{v_7\}, \{v_3, v_4\})$ and $e_4 = (\{v_5\}, \{v_6, v_7\})$.

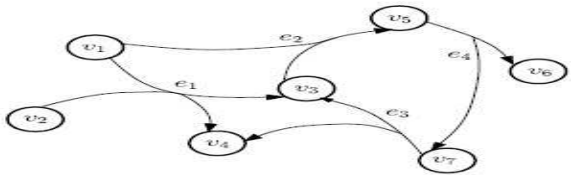


Figure 10: A BF-hypergraph H with 7 nodes and 4 hyperedges [29]

Definition 2.22. A Backward hyperedge, or B-hyperedge, is a hyperedge $e = (T(e), H(e))$ with $|H(e)| = 1$. Also a forward hyperedge, or F-hyperedge, is a hyperedge $e = (T(e), H(e))$ with $|T(e)| = 1$. For example, in the hypergraph of Figure 10, e_2 is a B-hyperedge, e_3 and e_4 are F-hyperedges in the hypergraph of Figure 10 [28]. A B-graph (or B-hypergraph) is a hypergraph whose hyperedges are B-hyperedges. An F-graph (or F-hypergraph) is a hypergraph whose hyperedges are F-hyperedges. A BF-graph (or BF-hypergraph) is a hypergraph whose hyperedges are either B-hyperedges or F-hyperedges. For example, the hypergraph of Figure 10 is BF-hypergraph.

Definition 2.23. Let a hypergraph $\overrightarrow{H} = (\overrightarrow{V}, \overrightarrow{E})$, be a directed hypergraph and let $s, t \in \overrightarrow{V}$. A simple directed hypergraph path P_{st} from s to t in $\overrightarrow{H}$ is a sequence $(v_1, e_1, v_2, e_2, ..., v_n, e_n, v_{n+1})$ consisting of

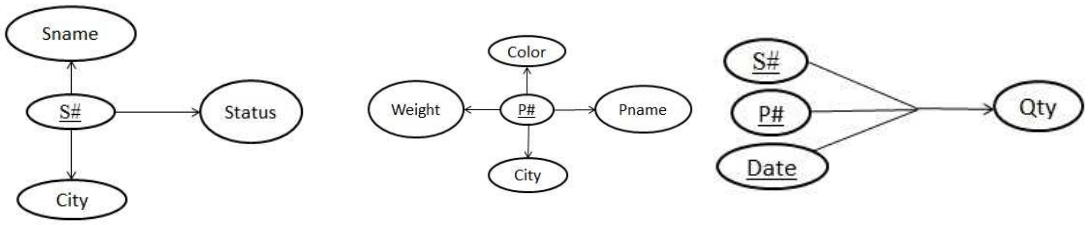
1. Nodes v_i, where $1 \leq i \leq n+1, v_i \in T(e_i)$ and
2. Distinct hyperedges e_j, where $1 \leq j \leq n$, such that $s = v_1, t = v_{n+1}$, and for every $1 \leq i \leq n, v_i \in T(e_i)$ and $v_{i+1} \in H(e_i)$. For exam-

ple, in Figure 10 $P_{v_1v_4} = (v_1, e_2, v_5, e_4, v_7, e_3, v_4)$ is a simple directed hypergraph path from v_1 to v_4.

Definition 2.24. A simple directed hypergraph path $P_{st} = (v_1, e_1, v_2, e_2, ..., v_n, e_n, v_{n+1})$ from $s = v_1$ to $t = v_{n+1}$ in hypergraph $\overrightarrow{H}$ is called a cycle if $|T(e_1)| > 1$ and $t \in T(e_1)$. The simple path P_{st} is called elementary if all the nodes v_i, where $1 \leq i \leq n+1$, are distinct [29]. For example, in Figure 10, $P_{v_1v_3} = (v_1, e_2, v_5, e_4, v_7, e_3, v_3)$ is a simple cycle in $\overrightarrow{H}$ (since $v_3 \in T(e_2) = \{v_1, v_3\}$).

Definition 2.25. Given a relation R, a set of attributes X in R is said to functionally determine another set of attributes Y, also in R, (written $X \longrightarrow Y$) if and only if each X value in R is associated with precisely one Y value in R; R is then said to satisfy the functional dependency $X \longrightarrow Y$ [30]. i.e., $\exists f : X \longrightarrow Y$.

Definition 2.26. Let V be the set of attributes of a RDB. A Functional Dependency $F(X, Y)$, with both X and Y are subsets of V, defines uniquely the value of the attributes in Y once the value of the attributes in X is given [31]. For example, the functional dependency (FD) of the database schema of Figure 8, is $\{P\#\} \longrightarrow \{\{Pname\}, \{City\}, \{Color\}, \{Weight\}\}$ for the PART relation scheme, $\{S\#\} \longrightarrow \{\{Sname\}, \{Status\}, \{City\}\}$ for the SUPPLIER relation scheme, and $\{S\#, P\#, Date\} \longrightarrow \{Qty\}$ for the SHIPMENT relation scheme.



(a) FD F-hypergraph for SUPPLIER relation

(b) FD F-hypergraph for PART relation

(c) FD B-hypergraph for SHIPMENT relation

Figure 11:

Definition 2.27. A set F of all Functional Dependencies on the attribute set N can be represented by a directed hypergraph $\overrightarrow{H} = (\overrightarrow{V}, \overrightarrow{E})$, with $\overrightarrow{V} =$

N and $\overrightarrow{E} = \{(X, Y \setminus X) : F(X, Y) \in F, Y \nsubseteq X\}$ [31]. For example, Figure 11 (a), represents the set of functional dependencies of the SUPPLIER relation, Figure 11 (b), represents the set of functional dependencies of the PART relation, and Figure 11 (c), represents the set of functional dependencies of the SHIPMENT relation in the database schema of Figure 8.

3 The Proposed Model

In this section we propose a model that represents RDF-BF-hypergraph schema that corresponds to an acyclic/cyclic RDB schema according to its set of functional dependency.

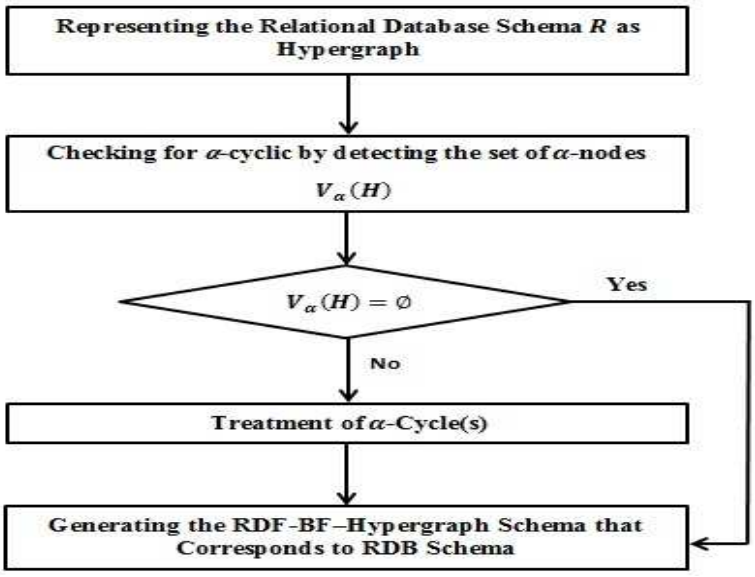


Figure 12: The four steps of the proposed model

The model consists of four steps. In the first step a given database schema is represented as hypergraph by identifying its tables and their attributes. Second step will check for α-cyclicity by detecting the set of α-nodes. If the hypergraph is α-cyclic, then we will treat this cycle(s) to see if it can be removed or not, which is the third step. In the fourth step, an RDF-BF-hypergraph schema will be generated for the resulted acyclic undirected hypergraph in the case if the cycle(s) can be removed or for the original undirected hypergraph of step one. Moreover, two algorithms of the proposed model are introduced. The first algorithm converts a cyclic undirected hypergraph that corresponds to RDB schema into acyclic one if it is possible,

which is used in the third step. The second algorithm generates the RDF-BF-hypergraph schema that corresponds to an acyclic/cyclic RDB schema according to its set of functional dependency, which is used in the fourth step. The four steps are illustrated in Figure 12.

3.1 Representing the Relational Database Schema R as Hypergraph

In this step, the relational database schemes is represented as hypergraphs, by creating a node for every attribute of a database scheme **R** and creating a hyperedge for every relation scheme R in **R**.

3.2 Checking for α-cycles

In this step, the algorithm $A_0(H)$ which is proposed by Ghaleb et. al [26] is used. Algorithm $A_0(H)$ depends on the output of the GYO algorithm to detect the set of $\alpha-$nodes which is based on the existence of quasi $\alpha-$cycle(s) in a given hypergraph H. This algorithm returns the set of $\alpha-$nodes, $V_\alpha(H)$ if the hypergraph is cyclic, or returns $V_\alpha(H) = \emptyset$ if the hypergraph is acyclic.

3.3 Treatment of α-cycle(s)

In this step, the following proposed algorithm, $\alpha_{Rem}(H,\mathbf{n},K)$ is introduced to convert a cyclic hypergraph H, which corresponds to RDB into an acyclic one if it is possible or return failure.

Note that taking a database schemes in the third normal form will guarantee that every non-key attribute A in R is fully functionally dependent on the primary key of R, and no non-key attribute of R is transitively dependent on the primary key.

Algorithm 1 $\alpha_{Rem}(H,\boldsymbol{n},K)$

Input: An undirected hypergraph H which corresponds to a cyclic database schemes in the third normal form, the set of $\alpha-$nodes $V_\alpha(H)$ which is returned from $A_0(H)$ algorithm, and a set K, which is the set of all keys of this database schemes.

Output: An $\alpha-$acyclic hypergraph or failure.

Begin

Assume the set of $\alpha-$nodes $V_\alpha(H) = \{v_1, ..., v_n\}$

$H' = H$
Flag = true
while Flag **do**
  for $i = 1$ to n **do**
    if $v_i \in K$ **then**
      continue
    else
      rename v_i to table_name_v_i, where table_name is the name of the table it belongs to
    end if
  end for
  $V_\alpha(H') = A_0(H')$ // calling algorithm A_0 on H'
  Assume the output of $A_0(H')$ is the set of $\alpha-$nodes $V_\alpha(H') = \{v_1, ..., v_m\}$
  if $V_\alpha(H') = \emptyset$ **then**
    Flag = false // H' becomes $\alpha-$acyclic
  else if $(V_\alpha(H') \neq \emptyset$ && $\forall v \in V_\alpha(H') : v \in K)$ **then**
    Flag = false and Return failure
    // all the $\alpha-$nodes are keys which cannot be renamed
  else
    $\alpha_{Rem}(H', m, K)$
  end if
end while
return H'
End.

3.4 Generating the RDF-BF-Hypergraph schema that corresponds to RDB schema

In the last step, the proposed algorithm, $RDF - BF(V(H), F)$ is introduced to generate the RDF-BF-hypergraph schema that corresponds to an acyclic/cyclic RDB schema according to its set of functional dependency.

4 Results and Discussion

In this section, the result of the proposed model is illustrated through the following example. Consider the undirected hypergraph of Figure 9, which corresponds to database scheme of Figure 8. After applying algorithm $A_0(H)$ to this hypergraph, we get the set of $\alpha-$nodes: $V(H) = \{S\#, P\#, City\}$ [26]. Then after applying algorithm $\boldsymbol{\alpha_{Rem}}(H,\boldsymbol{n},K)$, the attribute City in both the SUPPLIER relation scheme and the PART relation scheme will be renamed to be Supplier_City and Part_City respectively. This gives an $\boldsymbol{\alpha}-$acyclic undirected hypergraph of Figure 14, which corresponding to the acyclic database schemes of Figure 13.

S#	Sname	Status	Supplier_City
S1	Doaa	20	Giza
S2	Noha	40	Giza
S3	Joudy	60	Cairo

P#	S#	Date	Qty
P1	S1	13/9/2017	100
P2	S1	7/11/2018	200
P1	S2	29/3/2015	300

P#	Pname	Color	Weight	Part_City
P1	Nut	Red	12	Cairo
P2	Bolt	Blue	10	Cairo
P3	Screw	Green	13	Giza
P4	Bolt	Red	15	Aswan

Figure 13: An instance of acyclic database of the database schemes of figure 8

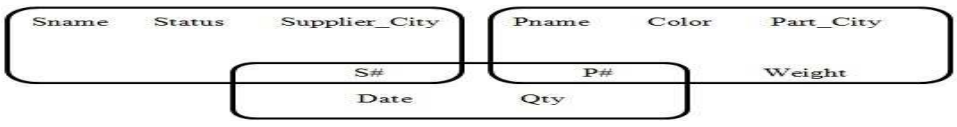


Figure 14: The acyclic hypergraph corresponding to the acyclic database schemes of Figure 13

After applying algorithm $\boldsymbol{RDF-BF(V(H), F)}$ to the resulted set of nodes of the acyclic undirected hypergraph of Figure 14, we will get the RDF-BF-Hypergraph schema with set of nodes that represent each attribute name:

Algorithm 2 *RDF-BF*$(V(H), F)$

Input: The set of nodes $V(H)$ of the undirected hypergraph H which is the output of the algorithm $\boldsymbol{\alpha_{Rem}}(H,\boldsymbol{n},K)$ in the case of the possibility of removing the $\boldsymbol{\alpha}-$cycle(s) or the set of nodes $\boldsymbol{V(H)}$ of the original hypergraph of step one and

The set of functional dependencies $\boldsymbol{F}$ of the form $\boldsymbol{X \longrightarrow Y}$

Output: An $\boldsymbol{RDF-BF}-$Hypergraph $\overrightarrow{\boldsymbol{H}}$.

Begin

Assume the set of all nodes of $\boldsymbol{H}$ is $\boldsymbol{V(H) = \{v_1, ..., v_n\}}$ and let$(\overrightarrow{\boldsymbol{H}}) = (\overrightarrow{\boldsymbol{V}}, \overrightarrow{\boldsymbol{E}})$.

1: $\overrightarrow{\boldsymbol{V}} = \emptyset$ and $\overrightarrow{\boldsymbol{E}} = \emptyset$
2: **for** $\boldsymbol{i = 1}$ to $\boldsymbol{n}$ **do**
3: $\quad \overrightarrow{\boldsymbol{V}} = \overrightarrow{\boldsymbol{V}} \cup \{\boldsymbol{v_i}\}$
4: **end for**
5: **for all** $\boldsymbol{X \longrightarrow Y \in F}$ **do**
6: $\quad$ **if** $|\boldsymbol{X}| = 1$ && $|\boldsymbol{Y}| = 1$ **then**
7: $\quad\quad \overrightarrow{\boldsymbol{E}} = \overrightarrow{\boldsymbol{E}} \cup \boldsymbol{e} = (\{\boldsymbol{x}\}, \{\boldsymbol{y}\})$
// Add a directed edge from $\boldsymbol{x \in X}$ to $\boldsymbol{y \in Y}$
// i.e., $\boldsymbol{\{x\} \in T(e), \{y\} \in H(e)}$,
// where $\boldsymbol{X = \{x_1, ..., x_m\}}$
8: $\quad$ **else if** $(|\boldsymbol{X}| > 1$ && $|\boldsymbol{Y}| = 1)$ **then**
9: $\quad\quad \overrightarrow{\boldsymbol{E}} = \overrightarrow{\boldsymbol{E}} \cup \boldsymbol{e} = (\{\boldsymbol{x_1, ..., x_m}\}, \{\boldsymbol{y}\})$ && $\overrightarrow{\boldsymbol{V}} = \overrightarrow{\boldsymbol{V}} \cup \{\boldsymbol{B_j}\}$
// Add a directed B-hyperedge from all elements in X to the element
// in Y and create a new connector node $\boldsymbol{B_j}$ to connect all
// $\boldsymbol{\{x_1, ..., x_m\} \in X}$ to $\boldsymbol{Y}$ i.e., $\boldsymbol{\{x_1, ..., x_m\} \in T(e), y \in H(e)}$
10: $\quad$ **else if** $|\boldsymbol{X}| > 1$ && $|\boldsymbol{Y}| > 1$ **then**
11: $\quad\quad \overrightarrow{\boldsymbol{E}} = \overrightarrow{\boldsymbol{E}} \cup \boldsymbol{e} = (\{\boldsymbol{x_1, ..., x_m}\}, \{\boldsymbol{y_1, ..., y_k}\})$ && $\overrightarrow{\boldsymbol{V}} = \overrightarrow{\boldsymbol{V}} \cup \{\boldsymbol{BF_l}\}$
// Add a directed BF-hyperedge from all elements in X to
// all elements in Y and create a new connector node
// $\boldsymbol{BF_l}$ to connect all $\boldsymbol{\{x_1, ..., x_m\} \in X}$ to $\boldsymbol{\{y_1, ..., y_k\} \in Y}$
// i.e., $\boldsymbol{\{x_1, ..., x_m\} \in T(e), \{y_1, ..., y_k\} \in H(e)}$),
// where $\boldsymbol{X = \{x_1, ..., x_m\}}$ and $\boldsymbol{Y = \{y_1, ..., y_k\}}$
12: $\quad$ **end if**
13: **end for**
return $\overrightarrow{\boldsymbol{H}}$
End.

$\overrightarrow{\boldsymbol{V}} = \{\boldsymbol{S\#}, \boldsymbol{Sname}, \boldsymbol{Status}, \boldsymbol{Supplier\_City}, \boldsymbol{P\#}, \boldsymbol{Pname}, \boldsymbol{Color}, \boldsymbol{Weight}, \boldsymbol{Part\_City}, \boldsymbol{Date}, \boldsymbol{Qty}\}$, and the set of directed hyperedges $\overrightarrow{\boldsymbol{E}} = (\{\boldsymbol{S\#}\}, \{\boldsymbol{Sname}\}), (\{\boldsymbol{S\#}\}, \{\boldsymbol{Status}\}), (\{\boldsymbol{S\#}\}, \{\boldsymbol{Supplier\_City}\}), (\{\boldsymbol{P\#}\}, \{\boldsymbol{Pname}\}), (\{\boldsymbol{P\#}\}, \{\boldsymbol{Color}\}), (\{\boldsymbol{P\#}\}, \{\boldsymbol{Weight}\}), (\{\boldsymbol{P\#}\}, \{\boldsymbol{Part\_City}\}), (\{\boldsymbol{S\#}, \boldsymbol{P\#}, \boldsymbol{Date}, \boldsymbol{B_1}\}, \{\boldsymbol{Qty}\})$, as shown in Figure 15.

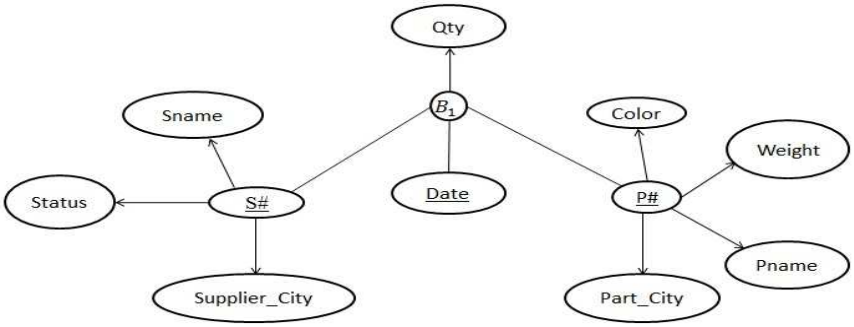


Figure 15: the RDF-BF-Hypergraph schema that corresponds to the acyclic database schema of Figure 13

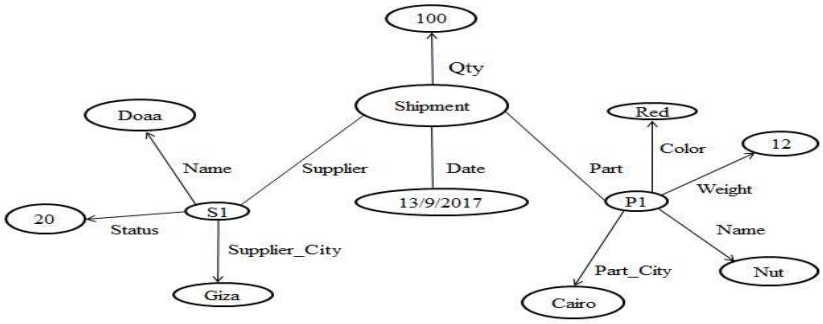


Figure 16: The RDF-BF-hypergraph instance of the RDF-BF-hypergraph schema of Figure 15

A formal representation of RDF_graph schema will preserve the integrity constraints to ensure integrity and data semantics, which is captured from a database schema. Moreover, a formal representation of RDF_graph schema will facilitate the data conversion and will allow graph connectivity which is essential for querying RDF by using concepts, techniques and various traversal algorithms. The RDF-BF-hypergraph instance can be easily obtained as follows:

- Each node will have a label that corresponds to attribute name's value. Each hyperedge will have a label that corresponds to that attribute

name [32].

- In the case of the many-to-many relational joins, each B-connector will be replaced by the name of the bridge table as shown in Figure 16 in which the B-connector $\boldsymbol{B_1}$ of Figure 15, is replaced by shipment which is the name of the bridge table of the database of Figure 8.
- In the case of one-to-many relation joins, the hyperedge will connect the value of the primary key in "many" table to the referenced value of the foreign key in the one table through a label that corresponds to the name of "one" table, for example Figure 19 shows the instance of the RDF-BF-hypergraph for one-to-many relation joins of the RDF-BF-Hypergraph schema of Figure 18, that corresponds to the acyclic database schema of Figure 17.

EMP_ID	EMP_NAME	SALARY	DEPT_ID
1	Lorin	2000	1
2	Yasmine	2500	1

EMPLOYEE

DEPT_ID	DEPT_NAME	MGR
1	Computer Science	Adel
2	Information Technology	Salma

DEPARTMENT

Figure 17: An instance of acyclic database

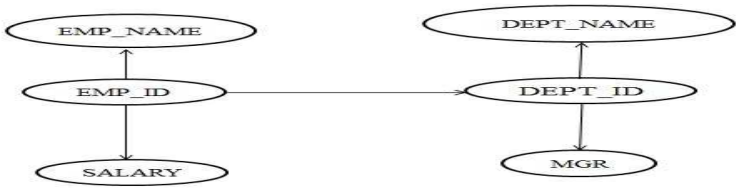


Figure 18: The RDF-BF-Hypergraph schema that corresponds to the acyclic database schema of Figure 17

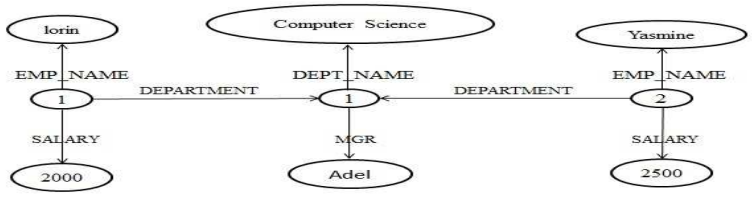


Figure 19: The RDF-BF-hypergraph instance of the RDF-BF-hypergraph schema of Figure 18

One of the earliest languages for the purposes of querying RDF graphs is Simple Protocol and RDF Query Language (SPARQL) [33]. In SPARQL

queries are expressed as a graph and query evaluation relies on graph matching between the query and the database [34]. SPARQL shares a conceptual core, which consists of two natural operations in the context of querying graphs:

- Graph pattern matching, where a query pattern can be viewed as a graph $\boldsymbol{P = (V_p, E_p)}$. Here, the nodes and edges describe query conditions that a subgraph of a given $\boldsymbol{G}$ must satisfy in order to be matched. Given a graph $\boldsymbol{G}$, the pattern matching problem is to find all possible subgraphs of $\boldsymbol{G}$ that match a given pattern $\boldsymbol{P}$ [35].

- Graph navigation that relies on path existence, which asks if there is some directed path between two nodes, irrespective of edge labels [36].

Figure 20: The RDF-BF-hypergraph instance of the cyclic database of Figure 8

Assume that we have a query to find all suppliers that ships all parts in City Giza from the RDF-BF-hypergraph instance of Figure 20 that corresponds to the cyclic database of Figure 8. By applying the two operations of SPARQL then City is a constant that will only match edges with the corresponding label in the original graph [36]. Hence the result will consist of two steps. In the first step those subgraphs of the original graph, which consist of all such valid matches corresponding to the label "City" will be return. In the second step the supgraphs that match all parts in giza city are chosen to be the final result. Whereas for the RDF-BF-hypergraph instance of Figure 19 that corresponds to the RDF-BF-hypergraph schema of Figure 18, the result will be reduced to those subgraphs which consist of all matches corresponding to the label "Part_City". Hence in the case of acyclic RDF-BF-hypergraph instance, the time and space that needed for query answering will be reduced and finally the semantics of the RDB schema will be maintained.

5 Conclusion and Future Work

The semantic web has an initiative that aims to improve the current state of the World Wide Web to make data machine-understandable. Semantic web depends on RDF-graph data model which is a standard model for data interchange on the web. Even web contents that are generated automatically from databases are usually presented without the original structural information found in databases. In the last decades a special class of database schemes, called acyclic database schema, was introduced. In the case of acyclic hypergraphs the query optimization becomes easier than in the case of cyclicity and might be recognized in linear time. This paper introduces a BF-hypergraph representation for the RDF schema that corresponds to RDB schema. The BF-hypergraph is a suitable model to represent the set of functional dependencies of RDB since the domain and codomain of the functional dependency may contain more than one attribute. We propose a model, consisting of four steps, to represent the RDF-BF-hypergraph schema that corresponds to an acyclic/cyclic RDB schema according to its set of functional dependency. In the first step a given database schema is represented as hypergraph by identifying its tables and their attributes. The second step will check for $\boldsymbol{\alpha}$-cyclicity by detecting the set of $\boldsymbol{\alpha}$-nodes. If the hypergraph is $\boldsymbol{\alpha}$-cyclic, then we will treat this cycle(s) to see if it can be removed or not, which is the third step. In the fourth step an RDF-BF-hypergraph schema will be generated for the resulted acyclic undirected hypergraph in the case if the cycle(s) can be removed or for the original undirected hypergraph of step one. Moreover, two algorithms of the proposed model are introduced. The first algorithm converts a cyclic undirected hypergraph that corresponds to RDB schema into acyclic one if it is possible, which is used in the third step. The second algorithm generates the RDF-BF-hypergraph schema that corresponds to an acyclic/cyclic RDB schema according to its set of functional dependency, which is used in the fourth step. A formal representation of the RDF-graph schema will preserve the integrity constraints to ensure integrity and data semantics. Moreover, a formal representation of RDF-graph schema will facilitate the data conversion and will allow graph connectivity which is essential for querying the RDF by using concepts, techniques and various traversal algorithms. In the case of acyclic RDF-BF-hypergraph instance, the time and space needed for query answering will be reduced and, finally, the semantics of the RDB schema will be maintained. For future work, we will extend our model to propose a tool to generate the RDF-BF-hypergraph schema and instance from an existing database.

References

[1] O. Lassila, R. Swick, Resource Description Framework (RDF) Model and Syntax Specification, W3C Recommendation, World Wide Web, http://www.w3.org/TR/1999/REC-rdf-syntax-19990222, 1999.

[2] J. Davies, R. Studer, P. Warren, Semantic Web Technologies Trends and Research in Ontology-based Systems, 1st edition, John Wiley & Sons Ltd, 2006.

[3] E. Miller, R. Swick, D. Brickley, Resource Description Framework (RDF) / W3C Semantic Web Activity, World Wide Web, http//www.w3.org/RDF/, 2004.

[4] T. Berners-Lee, Semantic Web Road map, World Wide Web, http://www.w3.org/DesignIssues/Semantic.html, 1998.

[5] R. Angles, C. Gutierrez, Survey of graph database models, ACM Computer Survey, **40,** no. 1, (2008), 1–39.

[6] R. Virgilio, A. Maccioni, R. Torlone, Converting Relational to Graph Databases, Proceedings of the First International Workshop on Graph Data Management Experience and Systems, (2013), 1–3.

[7] F. Michel, J. Montagnat, C. Farozucker, A survey of RDB to RDF translation approaches and tools, Research Report number I3S/RR 2013-04-FR, 25 pages, I3S laboratory, Sophia Antipolis, France, 2013.

[8] E. F. Codd, A relational model of data for large shared data banks, Communication of the ACM, **13,** no. 6, (1970), 377–387.

[9] M. C. Yeow, J. Lijun, L. Andrew, K. H. Anthony, Arboricity: An acyclic hypergraph decomposition problem motivated by database theory, Discrete Applied Mathematics, **160,** (2012), 100–107.

[10] R. Fagin, Degrees of acyclicity for hypergraphs and relational database schemes, Communication of the ACM, **30,** no. 3, (1983), 514-550.

[11] C. Beeri, R. Fagin, M. David, Y. Mihalis, On the desirability of acyclic database schemes, Communication of the ACM, **30,** no. 3, (1983), 479–513.

[12] M. David, D. Ullman, Connections in acyclic hypergraphs, Theoretical Computer Science, **32,** (1984), 185–199.

[13] C. Beeri, R. Fagin, D. Maie, A. O. Mendelzon, J. D. Ullman, M. Yannakakis, Properties of acyclic database schemes, In: Proceedings of the thirteenth annual ACM symposium on Theory of computing, (1981), 355–362.

[14] S. Chokkalingam, Database and query analysis tools for MySQL: exploting hypertree and hypergraph decompositions Master Thesis, Faculty of the Russ College of Engineering and Technology Ohio University, Ohio, USA, 2006.

[15] K. Ahmet and O. Dan, Covers of Query Results. In: ICDT 2018, 21st International Conference on Database Theory, 2018.

[16] X. Hu, K. Yi, Instance and Output Optimal Parallel Algorithms for Acyclic Joins. In: Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, (2019).

[17] H. Chen, Y. Yoshida, Testability of Homomorphism Inadmissibility: Property Testing Meets Database Theory. In: Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, (2019).

[18] T. Berners-Lee, Relational Databases on the Semantic Web, Internet note, URL http://www.w3.org/DesignIssues/RDB-RDF.html, 22, 2006 (originally published September 1998).

[19] K. Byrne, Having Triplets Holding Cultural Data as RDF, Proceedings of the ECDL 2008 Workshop on Information Access to Cultural Heritage, Aarhus, Denmark, (2008), 1–13.

[20] G. Klyne, J. Carroll, Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation, World WideWeb, http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/, 2004.

[21] J. Hayes, C. Gutierrez, Bipartite graphs as intermediate model for RDF, In: Proceedings of the 3th International Semantic Web Conference (ISWC), 3298, (2004), 47-61.

[22] J. Bang-Jensen, and G. Gregory, directed graphs: Theory, Algorithms and Applications, 1st edition, Springer, 2000.

[23] J. Brault-Baron, Hypergraph acyclicity revisited, ACM Computing Surveys (CSUR), **49.,**Issue 3, (2016), 1–32.

[24] C. J. Date, An Introduction to Database System, 3rd edition, 1, Narosa Publishing House, 2002.

[25] J. A. Philippe, N. N. Samba, On the notion of cycles in hypergraphs, Discrete Mathematics, **309,** (2009), 6535–6543.

[26] F. M. Ghaleb, A. A. Taha, M. Hazman, M. Abd ElLatif, M. Abbass, On Quasi Cycles in Hypergraph Databases, in submission.

[27] M.H. Graham, On the universal relation, Technical Report, University of Toronto, 1979.

[28] M. R. Moyano, F. Jose, Strong Connectivity in Directed Hypergraphs and its Application to the Atomic Decomposition of Ontologies, Ph.D. Thesis, Artificial intelligent department Polytechnic university of Madrid, 2016.

[29] M. Thakur, R. Tripathi, Linear connectivity problems in directed hypergraphs, Theoretical Computer Science, **410,** (2009) 2592–2618.

[30] R. El masri, B. Sh. Navathe, Fundamental of Database Systems, 6th edition, Addison-Wesley, 2010.

[31] G. Gallo, G. Longo, S. Nguyen, S. Pallottino, Directed Hypergraphs and Applications, Journal of Discrete Applied Mathematics - Special issue: combinatorial structures and algorithms, **42,** Issue 2-3, (1993), 177-201.

[32] A. Antonio, M. E. Vidal, Directed hyper-graphs for RDF documents, Technical Journal of the Faculty of Engineering, **33,** no. 1, (2010), 59–67.

[33] E. Prudhommeaux, A. Seaborne, SPARQL Query Language for RDF. W3C Recommendation, http://www.w3.org/TR/rdf-sparql-query/, 2008.

[34] S. Harris, A. Seaborne, SPARQL 1.1 Query Language. W3C Recommendation, http://www.w3.org/TR/sparql11-query/, 2013.

[35] A. Gubichev, M. Then, Graph Pattern Matching Do We Have to Reinvent the Wheel?, Proceedings of Workshop on Graph Data management Experiences and Systems, (2014), 1–7.

[36] R. Angles, M. Arenas, P. Barcelo, A. Hogan, J. L. Reutter, D. Vrgoc, Foundations of Modern Query Languages for Graph Databases, Comput. Surveys, **50,** no. 5, (2017).