Data Lineage Discovery in Databases Based on Knowledge Graph Link Prediction

Jakub Dutkiewicz[1], Paweł Misiorek[1,*] and Robert Wrembel[1]

[1]Poznan University of Technology, Poznań, Poland

Abstract

Discovering missing data lineage relationships (links) in large data repositories is an important problem in the domain of data integration, but research on automated methods for identifying such links is limited. It is because real business data and database schemas are unavailable. As a consequence, benchmarks for testing discovery methods and grounded evaluations of such methods do not exist. In this paper, we propose a novel data lineage link discovery algorithm, which is based on inductive link prediction in a Knowledge Graph (KG). To this end, benchmark data and database objects (tables and views) are generated by means of an LLM-based framework, to represent the set of missing data lineage scenarios. Then, as part of our contribution, we propose a semantic model that is used to transform the lineage scenarios into KGs, where database objects are modeled as vertices and data lineage links are modeled as a subset of graph edges. The model that we propose aims to discover missing links in KGs in a novel inductive setting where the test set consists of database objects different from those used for training. As the prediction model, we use a Graph Neural Network (GNN) that utilizes information from the vertex graph neighborhood to learn patterns of data lineage link occurrences. We run a set of experiments using an adjusted evaluation method for knowledge graph link prediction, tailored to the case of data lineage graphs. The experiments show promising results, measured using standard machine learning metrics (AUROC, Hits@n), on the proposed benchmark dataset.

Keywords

data lineage, semantic transformations, knowledge graphs, inductive link prediction

1. Introduction

Data lineage refers to the set of techniques that document the life-cycle of data, from their origin, through transformations they undergo on the way to be uploaded into a destination system [1]. A special class is *object lineage*, which describes a chain of dependencies between database (DB) objects. Such dependencies represent scenarios where one object (e.g., a table) is used to create another object (e.g., a materialized view). The dependencies between such objects are often many-to-many, which is a typical scenario in data warehouses (DWs) [2].

Maintaining these relationships is essential for ensuring DB integrity and optimizing resource usage by avoiding the creation of unnecessary, redundant objects, which increase maintenance costs and degrade performance. While some dependencies are automatically managed by a Database Management System (DBMS), many remain unknown. For instance, when a DB object is created from temporal data generated by a stored function or when a permanent table is derived from a temporary one, the underlying relationship between such objects is often lost. We define these hidden or missing relationships as *broken object lineage* (or *broken lineage* for short). Discovering broken lineage is challenging due to large number of DB/DW objects and much larger number of possible relationships between them.

Notice that, relationships between DB objects in a natural way can be modeled as a graph, with nodes representing DB objects and edges representing links between these objects.

There is an extensive research on methods for discovering missing edges in graphs [3], and specifically concerning the algorithms for link prediction in knowledge graphs (KG) [4, 5, 6]. However, none of

Published in the Proceedings of the Workshops of the EDBT/ICDT 2026 Joint Conference (March 24-27, 2026), Tampere, Finland

*Corresponding author.

jakub.dutkiewicz@put.poznan.pl (J. Dutkiewicz); pawel.misiorek@put.poznan.pl (P. Misiorek); robert.wrembel@put.poznan.pl (R. Wrembel)

0000-0002-7954-7484 (J. Dutkiewicz); 0000-0001-5223-240X (P. Misiorek); 0000-0001-6037-5718 (R. Wrembel)

these approaches has been applied to the data lineage discovery problem. Therefore, a natural research direction is to use graphs to represent and transform knowledge about database structure and content in a way focused on preserving information concerning lineage between DB objects.

The semantic ontology-based transformation framework enables a flexible way to achieve this, with the ability to test different modeling schemas and approaches. The challenge is to model all expert domain knowledge about rules and evidence of lineage relationships between DB tables using KG nodes and labeled links. Due to the fact that KGs are human-readable structures, the proposed approach leads to an explainable machine learning solution, which is desirable from the perspective of practical applications. We chose the inductive link prediction approach [5, 6, 7] with the goal to discover broken lineage links in fully inductive settings. In this approach, a completely new (previously unseen) DB is transformed into KG nodes, without the need of dedicated model training or fine-tuning [6].

The paper **contributes** the following novel artifacts: (1) a procedure for creating broken lineage scenarios, resulting in a benchmark DB schema and dataset; (2) a semantic model used to transform the data lineage scenarios into KGs; and (3) a data lineage-oriented modification of the inductive link prediction experimental evaluation protocol used in the literature. Based on these novel features, we provide an end-to-end evaluation of the performance (measured by AUROC and Hits@10) of the selected state-of-the-art GNN-based algorithm [8] for the discovery of broken lineage in relational DBs. To the best of our knowledge, this is the first application of KGs and inductive link prediction applied to discovering data lineage (in general) and broken lineage (in particular).

2. Related Work

2.1. Data lineage/provenance

A few types of solutions for storing/retrieving data provenance in DBs have been proposed, namely: annotation-based, inversion-based, and lineage graphs. In the *annotation-based* solutions input data are annotated with information about their provenance (for example a location in which they are stored) and the annotations are propagated throughout the whole data processing workflow, until the final destination. Historically, first annotation-based solutions include Polygen [9] and DBNotes [10], whereas more recent works include [11, 12]. A common approach to annotation based data provenance representation is the *semiring* model [13, 14]. The *inversion-based* solutions invert queries trying to infer original data that produced a given result, e.g., [15, 16]. Finally, in the *lineage graph* approach, provenance is registered during query processing and saved in the form of a lineage graph for further analysis [17]. Paper [18] summarizes several most relevant works on provenance techniques.

Another path includes solutions for tracking provenance in scripting languages (e.g., Python and R), e.g., [19, 20, 21]. [22] proposes automatic Python processing without modifying existing scripts. Similar assumptions were made in [23], dedicated for data science applications. A completely different approach was proposed by the authors of [24], who advocate for allowing humans to express the desired provenance through a provenance schema.

In order to support the exchange of provenance data between different systems, the Open Provenance Model (OPM) was created [25]. The W3C standard PROV [26] aims at facilitating provenance data exchange over the Web.

2.2. Inductive link prediction for knowledge graphs

The surveys presented in [3] and [4] summarize the research on link prediction methods for graphs and KGs, respectively. In contrast to transductive KG link prediction methods (e.g., TransE [27] and BERT-ConvE [28]), which focus on completing missing relations between known entities, inductive KG link prediction aim to discover missing triples involving unseen entities [5, 6].

Most existing KG link prediction methods are embedding-based [27, 28]. These models independently project each triple into a specific vector space and thus learn dedicated embedding vectors for each KG entity and relation. More advanced approaches focus on the structural features in order to capture

complex semantic information and achieve better prediction performance [29]. However, all transductive methods have difficulty predicting missing links involving emerging entities in a KG.

To address this problem, solutions for inductive link prediction have recently been proposed [5, 6, 7, 8, 30]. The first inductive reasoning models inferred missing relations mainly by learning logical rules from KGs [31]. The authors of GraIL [6] proposed the first framework for utilizing local subgraph structures to predict relations in a fully inductive setting. Several extensions of this approach have since appeared [7, 8, 30]. In particular, in [30] the authors proposed a model that combines relational context, which captures the relation types of edges adjacent to a given entity pair, with relational paths, which characterize the relative position between the two entities in the KG. Other extensions include an inductive relation prediction model based on a global relation graph with topological information [5], as well as a path-based model that utilizes Siamese neural networks [8]. To the best of our knowledge, this work is the first to apply this method to KGs obtained through an ontology-grounded transformation of the real-world problem of broken lineage link discovery.

3. Preliminaries

Our research focuses on the **transformation** of database schemas and objects representing data lineage scenarios into knowledge graphs (KGs). Specifically, we use RDF as the framework for representing KGs, motivated by the fact that it is the standard way to represent ontology-modeled KGs. However, our method is not limited to RDF-based KGs and can be adapted to other types of KGs, such as property graphs used in Neo4j graph databases.

An **ontology-grounded KG** is defined as

$$\mathcal{G} = (\mathcal{E}, \mathcal{R}, \mathcal{T}, \mathcal{C}, \mathcal{A}),$$

where $\mathcal{E}$ is a set of entities (corresponding to graph nodes), $\mathcal{R}$ is a set of relations, $\mathcal{T} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$ is a set of triples (corresponding to graph edges), $\mathcal{C}$ is a set of ontology classes (the allowed types of entities), and $\mathcal{A}$ is a set of ontology definitions describing object proprieties corresponding to elements of $\mathcal{R}$ (types of links), as well as properties' domain and range constraints, and other semantic rules if applicable.

Let us assume the application of two disjoint entity sets:

$$\mathcal{E}_{\text{train}} \cap \mathcal{E}_{\text{test}} = \emptyset,$$

where $\mathcal{E}_{\text{train}}$ contains entities observed during training and $\mathcal{E}_{\text{test}}$ contains previously unseen entities, and the training and test graphs defined as follows:

$$\mathcal{G}_{\text{train}} = (\mathcal{E}_{\text{train}}, \mathcal{R}, \mathcal{T}_{\text{train}}, \mathcal{C}, \mathcal{A}),$$

$$\mathcal{G}_{\text{test}} = (\mathcal{E}_{\text{test}}, \mathcal{R}, \mathcal{T}_{\text{test}}, \mathcal{C}, \mathcal{A}).$$

Then the objective of **inductive link prediction for the KG** is to learn a scoring function

$$f : \mathcal{E}_{\text{test}} \times \mathcal{R} \times \mathcal{E}_{\text{test}} \to \mathbb{R}$$

using $\mathcal{G}_{\text{train}}$ such that f can predict missing triples that are not present in $\mathcal{T}_{\text{test}}$.

The visualization of fully inductive link prediction procedure for the case of broken data lineage discovery is presented on Figure 1. The figure illustrates a graph-based representation of a database, in which nodes correspond to database objects (e.g., tables, columns, and rows) and edges encode relationships between them. In the fully inductive setting, the training and test sets contain disjoint node sets while operating over a shared relation vocabulary. Examples of broken lineage scenarios are shown in Figure 2.

4. Dataset preparation

This section overviews a test DB schema and data. Scripts used to create the DB and fill it in with data are available trough a public GitHub repository[1]. The test DB is based on the Northwind database [32]. The initial DB was extended with: (1) the set of lineage transformations and (2) lineage tracking structures. We provide 10 different lineage scenario types, listed below.

- Basic *data copy* - data are copied from one object to another. The only operation allowed in such a scenario is filter-based selection.
- *Data transformation and aggregation* - basic data transformations are used, such as algebraic operations and string operations. Scenarios of this type allow for group based aggregations.
- *Join-based* data creation - new data are created based on joining multiple database objects; standard types of joins are allowed.
- *Derived views* - the main object in a scenario of this type is a view object. Data are transformed between views, and finally, data are stored within a new database object.
- *Temporary table* utilization - this type of scenario involves lineage, which is based on multiple temporary objects. The lineage pathway involves missing database objects, we simulate the expiration of temporary objects in the test data.
- *Recursive queries* and hierarchical data - involve common table expressions and recursive queries.
- *Data deduplication* - employs queries designed for data deduplication.
- *Partitioned data* processing - involve scripts and queries, which generate multiple new objects, based on filtering conditions.
- *Materialized summary tables* (materialized views) - this type of scenarios simulates the lineage created by reporting tasks (for example: lineage created during a monthly sales report generation).
- *Tabular-based objects* - this type of scenarios includes new DB objects created with use of tabular functions.

For each of aforementioned types, we generate 5 unique scenario instances, totaling to 50 lineage scenarios. Each follows a standard Extract-Transform-Load (ETL) pattern. We utilized Gemini 3 to generate the T-SQL scripts.

Figure 2 depicts three example scenarios. All 50 scenarios are available in the code repository. The train/test split is performed at the scenario level rather than the instance level. For each scenario type, four instances are allocated for training and one for testing, ensuring the evaluation data remain structurally distinct.

The characteristics of the generated scenarios, w.r.t. the number of various DB objects created and the number of relationships between them are summarized in Table 1.

In the next step, we convert the DB into a KG. The total number of lineage links between DB records directly dictates the graph size, while the number of DB object provenance links reflects the number of operations for each scenario type.

5. KG-based Data Lineage Discovery

The core of the proposed method is the semantic model-based transformation of a relational DB schema and data into a KG, used for broken lineage link discovery. We propose this semantic model by means of the ontology illustrated in Figure 3. As a good practice in semantic modeling, we introduce the ontology classes as subclasses of standard classes of state-of-the-art ontologies (presented in the figure using dark blue color). Our classes concern the description terminology of tabular data and datasets. In particular, we ground our model on the CSVW [33] and DSD [34] ontologies.

[1] https://github.com/dudenzz/lineage

Table 1
Generated scenario statistics. DOs - the total number of views and tables in a DB schema (+ the number of temporary tables). DOLin - the number of lineage relationships between database objects. DataLin - the number of lineage relationships between DB records.

Scenario type	DOs	DOLin	DataLin
Baseline Database	31(+0)	0	0
Basic Data Copy	42(+5)	19	603
Transformation lineage	48(+4)	19	1386
Join-based lineage	44(+3)	19	787
Data deduplication	44(+1)	16	2272
Views	45(+0)	15	1040
Summary tables	39(+0)	9	163
Recursive queries	39(+0)	12	182
Partitioning	45(+0)	14	1757
Temporary tables	36(+9)	14	1387
All scenarios	151 (+27)	141	1914
All training scenarios	128 (+22)	112	7955
All test scenarios	53 (+5)	29	9868

Our core model consists of five classes, presented using light blue color, linked by relationships, as shown in Figure 3. The crucial links for modeling lineage are represented by the relations *valueDerivedFrom* and *columnDerivedFrom*, inspired by the *isDerivedFrom* property from the widely used provenance ontology PROV-O [35].

Notice that the semantic model plays a crucial role in the final performance of the broken data lineage predictor. The same model, i.e., the same classes and properties (relationships), must be used to transform all training data lineage scenarios, as well as every relational DB with missing data lineage links, which are the target of discovery in the testing procedure. The ontology used in our test is available in the public repository[2].

The simplified illustration of KGs obtained as a result of ontology-based transformation of a DB schema and data is illustrated in Figure 1. The transformation is executed using a Python script that connects to the Database Management System and transforms the DB into a KG. The script uses SQL queries to navigate through the DB and SPARQL queries to retrieve the created nodes in the KG. We limit the database representation to the set of database objects defined in the ontology. Although the present study focuses on data lineage in relational databases, the proposed method is not limited to this setting. With appropriate modifications to the ontology used for transformation into a KG, the approach can be applied to semi-structured and tabular datasets of other types.

In order to build the inductive link predictor, we use the path-based model proposed in [8], which depends on KG properties (relations) and path embeddings. The objective of the proposed GNN is to capture entity-invariant topological patterns from the graph, therefore, it does not use node embeddings. The model has been proven to obtain results close to the state of the art for the general KG link prediction task. We chose it due to its simplicity, as well as the availability and reproducibility of the code. More details on the method can be found in the original paper [8].

The proposed approach is designed to predict the existence of link p between two given nodes (s, o), by analyzing connections between them. From the set of available paths, three are randomly selected, denoted as P_1, P_2, and P_3. In order to prevent data leakage, a path that consists only of link p is excluded from the set. The paths are represented as a sequence of node identifiers. The network maps these paths into a shared 300-dimensional embedding space. These sequences are then processed by two subsequent, recurrent bidirectional Long-Short Term Memory layers (BiLSTM), as shown in Figure 4. We use a combination of a max pooling layer and concatenation layer to create an input to a fully connected layer, which creates a final embedding of the three selected paths. This idea is based on the distributional hypothesis - the semantic information of the node is stored within its structural context.

[2]https://github.com/dudenzz/lineage

On the other end of the network, the predicate identifier is embedded into a separate 300-dimensional space. Then both embeddings are normalized. A dot product calculated between the node and predicate embeddings serves as a scoring function - the higher the value of that function, the more likely it is for this link to exist in a graph. A single pass of the network processes a batch of a size $|B|$. For this type of training, we use all connections in the graph as positive samples and we sample the same number of false connections, which do not appear in the graph.

6. Experiments

For the experimental evaluation, we prepared a set of transformation scripts, and an application to execute them on a clean DB instance. These transformations include both the lineage scenarios and the KG instantiation. All related code and artifacts are available in the code repository.

In our experiments we follow a fully inductive setting [6]. We use the state-of-the-art evaluation protocol and methodology utilized by research presented in [5, 6, 8, 7] adapted to the case of broken data lineage discovery. While the original methods treat node classes and relation types as irrelevant, we restrict the evaluation exclusively to the lineage predicate, i.e., the links *valueDerivedFrom* highlighted in Figure 3. For negative sampling, we generate triples (s, p, o) where both the subject and object are nodes of the class that constitutes the domain and range of *valueDerivedFrom* according to the ontology. For every positive sample, we generate 50 negative samples. These negative examples represent false provenance information between valid database tuples.

The GNN-based experiments were run on a server-class machine with the following specifications: CPU: Xeon Gold 6132; GPU: 2x Tesla T4 + CUDA 12.6; RAM: 504GB; Local storage: Micron_5300_MTFD 500GB; DBMS: SQL Server 2022. With this setup, we achieved a training time of roughly 16 hours per training session. It should be noted, that the training process was split into two parts: the batch generation and the neural network training. The batch generation, which accounts for approximately 50% of the computation time was performed on the CPU, while the neural network was trained on the GPU. The prediction model was trained on all of the scenario types. We used a batch size of 32 samples and a single epoch for training.

We trained the *General model* using KGs obtained from all types of scenarios, and two *Dedicated models* using only KGs representing the selected scenario types, namely *Basic data copy* and *Join-based lineage*. We excluded scenarios with fewer than 20 test samples, namely *Materialized summary tables* and *Recursive queries*. The dedicated models were evaluated exclusively on test data representing their corresponding scenarios, whereas the general model was evaluated separately on testing data from each scenario and on test data combined across all scenarios. The training sets and test sets statistics can be found in Table 1. For each evaluation setting, we measured the AUROC and Hits@10 metrics. Due to the randomness involved in the evaluation process, every experiment was repeated 10 times; the highest and lowest achieved metric values were excluded from the evaluation. The final reported metric value is an average of the values obtained in the remaining eight executions.

The exact values of the evaluation metrics depend largely on the inherent difficulty of the link discovery task for a given scenario. As this study constitutes the first investigation of missing link discovery in the context of data lineage, a direct quantitative comparison with existing studies on link prediction in knowledge graphs is not feasible. For reference, an AUROC value of 0.5, corresponding to a random predictor, can serve as a general baseline.

The highest AUROC scores are observed for the simplest scenario, *Basic data copy*, reaching 0.92 for the dedicated model and 0.86 for the general model. These results demonstrate that the model effectively captures lineage relationships of this type. In contrast, for more complex broken-lineage scenarios, the general model achieves AUROC values close to 0.6, indicating limited but non-negligible predictive capability. The lowest performance is obtained for the most challenging scenario, *Temporary tables*, where lineage discovery is evaluated for ETL processes involving temporary tables that are not observable in the test data. An AUROC value of 0.49 corresponds to the random baseline and indicates that the proposed model fails to capture lineage relationships in this setting. Since the final model is

based on GNN, the explainability of predicted links and prediction errors is limited.

Table 2
Performance metrics across different data generation scenarios.

Scenario	AUROC	Hits@10
Dedicated models		
Basic data copy	0.92	0.87
Join-based lineage	0.75	0.61
General model		
Basic data copy	0.86	0.65
Join-based lineage	0.63	0.31
Transformation lineage	0.60	0.28
Data deduplication	0.59	0.29
Derived views	0.59	0.28
Partitioning	0.59	0.31
Temporary tables	0.49	0.21
All scenarios	0.60	0.28

7. Conclusion and Future Work

To the best of our knowledge, we propose the first approach to the problem of broken lineage discovery that applies a GNN-based inductive link prediction model. The key element of the solution novelty is the transformation of the original problem into a graph, by means of ontology-based semantic modeling. The obtained promising results demonstrate that the proposed transformation framework may establish a new practical research branch in the domain of broken data lineage discovery in relational databases, and further in the future - in NoSQL storage systems.

As stated in Section 5, for our experiments we used a path-based GNN model [8] for inductive link prediction, which has been proven to provide results which are closed to the state of the art for the general link prediction task, but at the same time, it has certain limitations from the perspective of processing time and the KG size.

Due to the promising outcomes that we obtained, in the future we plan to test and adapt other GNN models, including those proposed in [5, 7, 36]. Moreover, the experimental evaluation protocol may also become a subject of future research, with the ultimate goal of reflecting the requirements of practical real-world missing lineage discovery scenarios as accurately as possible, including the case of applying GNN models that enable testing all possible lineage candidates for a given KG.

Future research directions will focus also on extending the proposed semantic ontology model with other concepts of the relational data model like: datatype definition, attribute length, integrity constraints, and other characteristics.

Finally, a comprehensive evaluation and comparison of various solutions will be performed. The solutions will include: (1) our KG-based solution, (2) the state-of-the-art methods (see Section 2), adjusted to the problem of discovering broken lineage, and (3) alternative methods based on statistical/mathematical modeling [37] and a ML-based method [38].

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] What is data lineage? IBM documentation, https://www.ibm.com/topics/data-lineage, Accessed Jan, 2026.

[2] A. A. Vaisman, E. Zimányi, Data Warehouse Systems - Design and Implementation, Second Edition, Data-Centric Systems and Applications, Springer, 2022.
[3] D. Arrar, N. Kamel, A. Lakhfif, A comprehensive survey of link prediction methods, The Journal of Supercomputing 80 (2023).
[4] A. Rossi, D. Barbosa, D. Firmani, A. Matinata, P. Merialdo, Knowledge graph embedding for link prediction: A comparative analysis, ACM Transactions on Knowledge Discovery from Data 15 (2021).
[5] C. Mu, L. Zhang, J. Li, Z. Wang, L. Tian, M. Jia, Inductive link prediction via global relational semantic learning, Information Systems 130 (2025).
[6] K. K. Teru, E. G. Denis, W. L. Hamilton, Inductive relation prediction by subgraph reasoning, in: Int. Conf. on Machine Learning (ICML), 2020.
[7] J. Wang, W. Li, F. Liu, Z. Wang, A. M. Luvembe, Q. Jin, Q. Pan, F. Liu, ConeE: Global and local context-enhanced embedding for inductive knowledge graph completion, Expert Systems with Applications 246 (2024).
[8] C. Zhang, X. Liu, Inductive link prediction in knowledge graphs using path-based neural networks, in: Int. Joint Conf. on Neural Networks (IJCNN), 2024.
[9] Y. R. Wang, S. E. Madnick, A polygen model for heterogeneous database systems: The source tagging perspective, in: Int. Conf. on Very Large Data Bases (VLDB), 1990.
[10] L. Chiticariu, W.-C. Tan, G. Vijayvargiya, DBNotes: a post-it system for relational databases based on provenance, in: Int. Conf. on Management of Data (SIGMOD), 2005.
[11] J. N. Foster, T. J. Green, V. Tannen, Annotated XML: queries and provenance, in: ACM SIGMOD-SIGACT-SIGART symposium on Principles of Database Systems (PODS), 2008.
[12] D. Dosso, S. B. Davidson, G. Silvello, Data provenance for attributes: attribute lineage, in: USENIX Conf. on Theory and Practice of Provenance, TAPP, USENIX Association, 2020.
[13] T. J. Green, G. Karvounarakis, V. Tannen, Provenance semirings, in: ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), 2007.
[14] T. J. Green, V. Tannen, The semiring framework for database provenance, in: ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS), ACM, 2017.
[15] Y. Cui, J. Widom, J. L. Wiener, Tracing the lineage of view data in a warehousing environment, ACM Transactions on Database Systems 25 (2000).
[16] M. Yamada, H. Kitagawa, T. Amagasa, A. Matono, Augmented lineage: traceability of data analysis including complex UDF processing, The VLDB Journal 32 (2023).
[17] A. Kashliev, Storage and querying of large provenance graphs using NoSQL DSE, in: Int. Conf. on Big Data Security on Cloud, Int. Conf. on High Performance and Smart Computing, and Int. Conf. on Intelligent Data and Security (BigDataSecurity/HPSC/IDS), IEEE, 2020.
[18] P. Senellart, Provenance and probabilities in relational databases, SIGMOD Record 46 (2018).
[19] M. Stamatogiannakis, P. Groth, H. Bos, Looking inside the black-box: Capturing data provenance using dynamic instrumentation, in: Int. Provenance and Annotation Workshop on Provenance and Annotation of Data and Processes, Springer, 2014.
[20] J. F. Pimentel, J. Freire, L. Murta, V. Braganholo, A survey on collecting, managing, and analyzing provenance from scripts, ACM Computing Surveys 52 (2019).
[21] A. Chapman, P. Missier, G. Simonelli, R. Torlone, Capturing and querying fine-grained provenance of preprocessing pipelines in data science, VLDB Endowment 14 (2020).
[22] J. F. Pimentel, L. Murta, V. Braganholo, J. Freire, noWorkflow: a tool for collecting, analyzing, and managing provenance from python scripts, Proc. VLDB Endowment 10 (2017).
[23] M. H. Namaki, A. Floratou, F. Psallidas, S. Krishnan, A. Agrawal, Y. Wu, Y. Zhu, M. Weimer, Vamsa: Automated provenance tracking in data science scripts, in: ACM SIGKDD Int. Conf. on Knowledge Discovery & Data Mining (KDD), 2020.
[24] Y. Lou, M. Cafarella, Enabling useful provenance in scripting languages with a human-in-the-loop, in: Workshop on Human-In-the-Loop Data Analytics (HILDA) @SIGMOD, 2022.
[25] L. Moreau, J. Freire, J. Futrelle, R. E. McGrath, J. Myers, P. Paulson, The Open Provenance Model: An Overview, in: Provenance and Annotation of Data and Processes (IPAW), volume 5272 of

LNISA, Springer, 2008.
[26] P. Missier, K. Belhajjame, J. Cheney, The W3C PROV family of specifications for modelling provenance metadata, in: Int. Conf. on Extending Database Technology (EDBT), 2013.
[27] A. Bordes, N. Usunier, A. Garcia-Durán, J. Weston, O. Yakhnenko, Translating embeddings for modeling multi-relational data, in: Int. Conf. on Neural Information Processing Systems (NIPS) Volume 2, 2013.
[28] X. Liu, H. Hussain, H. Razouk, R. Kern, Effective use of BERT in graph embeddings for sparse knowledge graph completion, in: ACM/SIGAPP Symposium on Applied Computing (SAC), 2022.
[29] W. L. Hamilton, R. Ying, J. Leskovec, Representation learning on graphs: Methods and applications, IEEE Data Eng. Bull. 40 (2017) 52–74.
[30] H. Wang, H. Ren, J. Leskovec, Relational message passing for knowledge graph completion, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, KDD '21, 2021, p. 1697–1707.
[31] C. Meilicke, M. Fink, Y. Wang, D. Ruffinelli, R. Gemulla, H. Stuckenschmidt, Fine-grained evaluation of rule- and embedding-based systems for knowledge graph completion, in: The Semantic Web – ISWC 2018, 2018, pp. 3–20.
[32] Microsoft, GitHub, Northwind and pubs sample databases for Microsoft SQL Server, https://github.com/microsoft/sql-server-samples/tree/master/samples/databases/northwind-pubs, Accessed Jan, 2026.
[33] W3C Document, CSVW Namespace Vocabulary Terms, https://www.w3.org/ns/csvw, Accessed Jan, 2026.
[34] Institute for Application-oriented Knowledge Processeing, Johannes Kepler University Linz, Austria, The Data Source Description Vocabulary, https://w3id.org/dsd, Accessed Jan, 2026.
[35] W3C Recommendation, PROV-O: The PROV Ontology, https://www.w3.org/TR/prov-o/, Accessed Jan, 2026.
[36] Y. Chen, P. Mishra, L. Franceschi, P. Minervini, P. Stenetorp, S. Riedel, REFACTOR GNNS: revisiting factorisation-based models from a message-passing perspective, in: Int. Conf. on Neural Information Processing Systems (NIPS), 2022.
[37] W. Andrzejewski, P. Boiński, R. Wrembel, On fixing broken lineage, in: ProvenanceWeek @SIGMOD, 2025.
[38] P. Boiński, W. Andrzejewski, M. Grocholewski, T. Gruszczyński, R. Wrembel, Leveraging machine learning techniques for discovering broken lineage links between database objects, in: Information Systems Development (ISD), 2025.

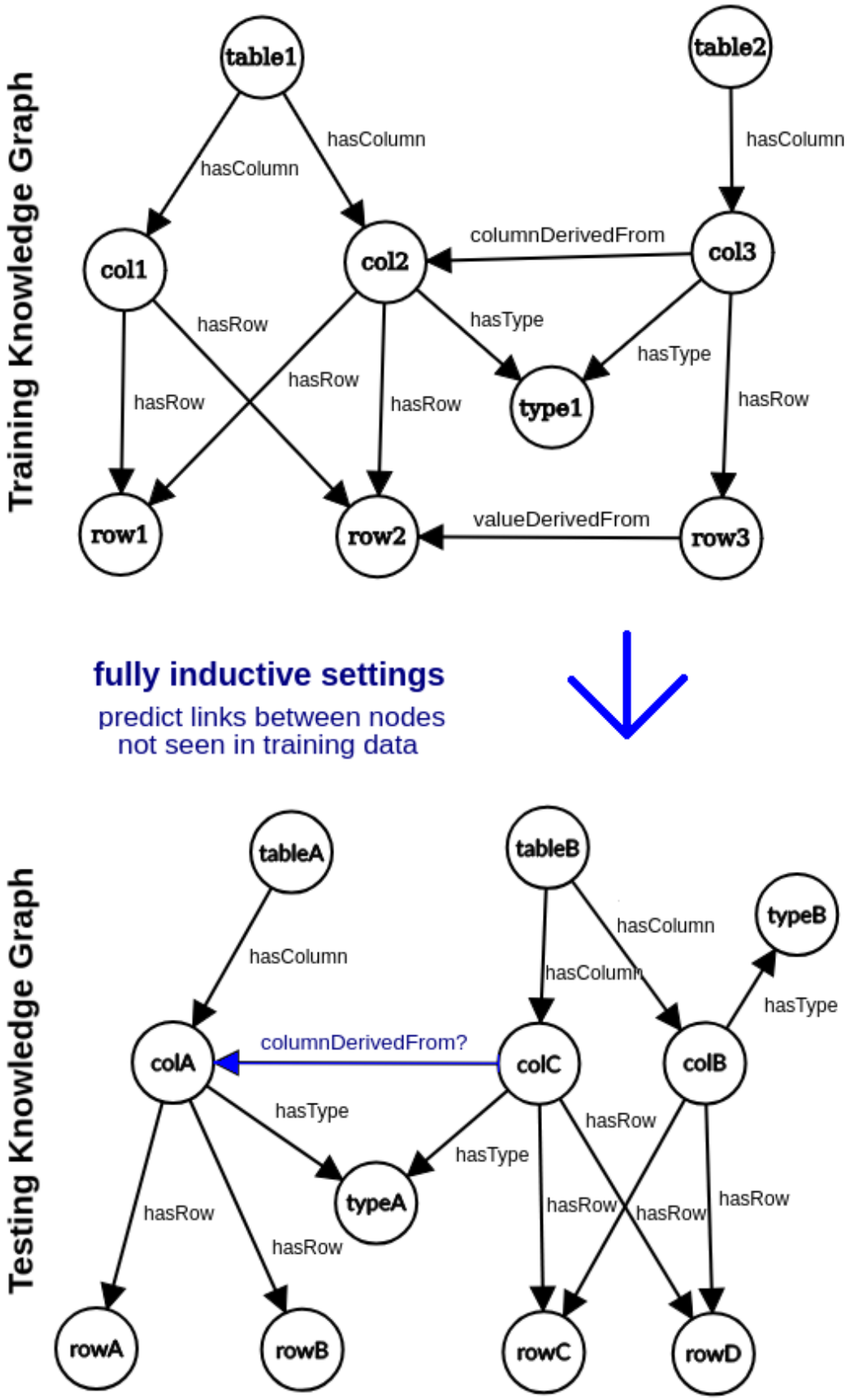

Figure 1: Inductive lineage link prediction visualization for KG obtained as a result of ontology-based transformation of the broken data lineage scenario data.

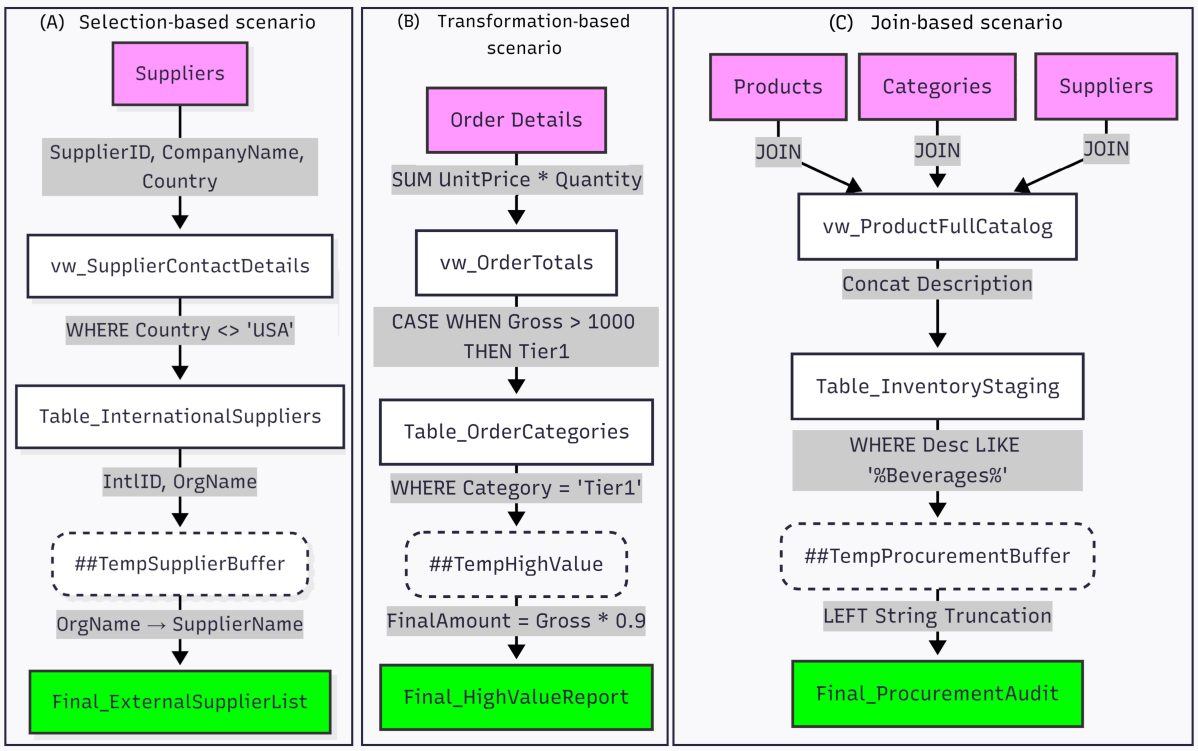

Figure 2: Three example lineage scenarios based on: (A) a simple selection, (B) a transformation, (C) a join.

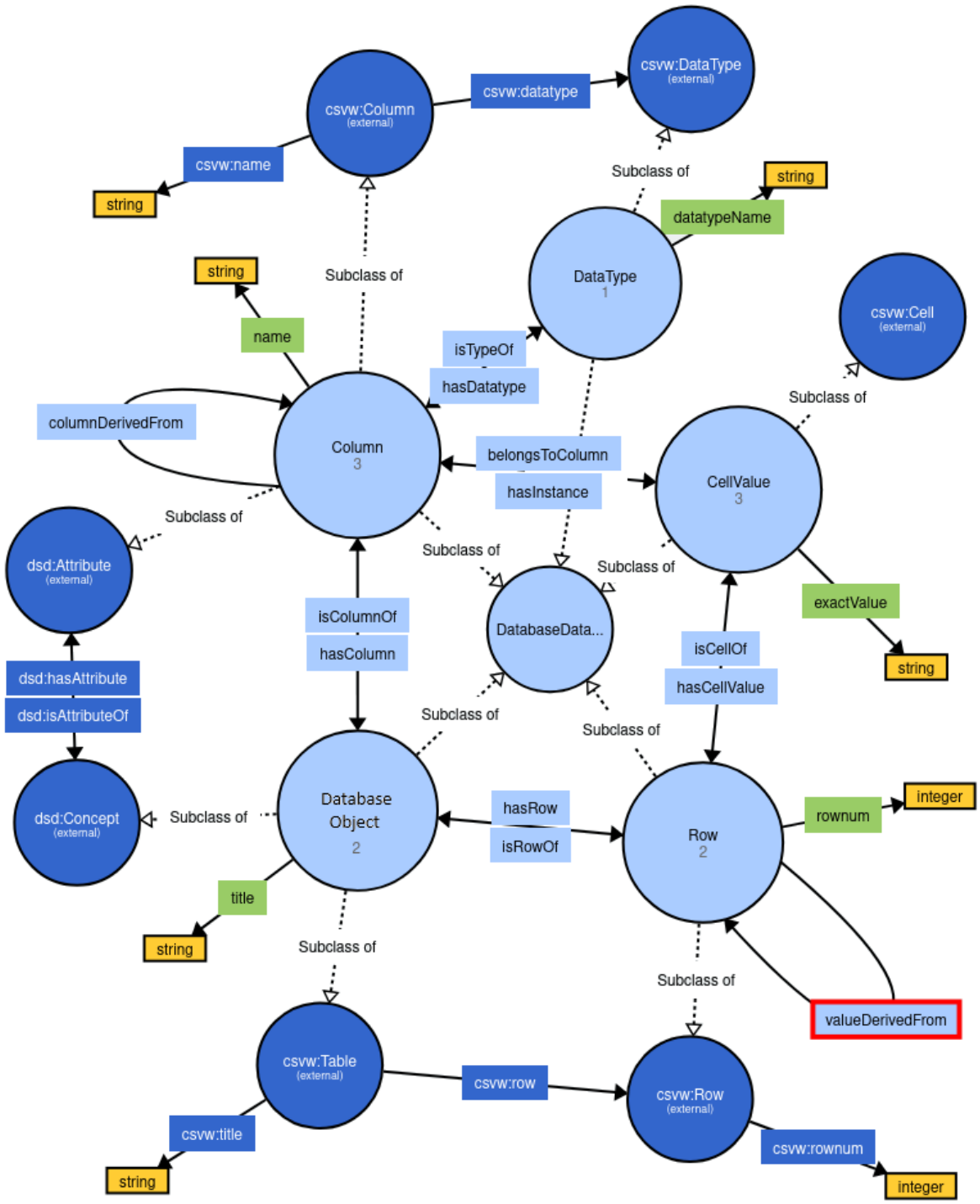

Figure 3: The ontology used to transform the data on database objects to the knowledge graph (the graphic was generated by WebVOWL).

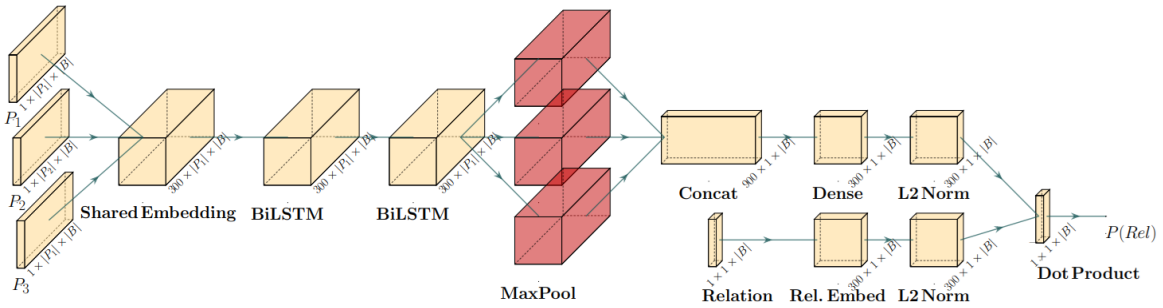

Figure 4: The GNN-architecture used in experiments. The diagram reflects a siamese neural network proposed in [8].