LANTERN: Boredom-conscious Natural Language Description Generation of Query Execution Plans for Database Education

Peng Chen[§] Hui Li[§] Sourav S Bhowmick[‡] Shafiq R Joty[‡] Weiguo Wang [‡,§]
[‡]School of Computer Science and Engineering, Nanyang Technological University, Singapore
[§]State Key Laboratory of Integrated Services Networks, Xidian University, China
assourav|srjoty@ntu.edu.sg,hli@xidian.edu.cn,pchen97|wgwang@stu.xidian.edu.cn

ABSTRACT

The database systems course in an undergraduate computer science degree program is gaining increasing importance due to the continuous supply of database-related jobs as well as the rise of Data Science. A key learning goal of learners taking such a course is to understand how SQL queries are executed in an RDBMS in *practice*. Existing RDBMS typically expose a *query execution plan* (QEP) in visual or textual format, which describes the execution steps for a given query. However, it is often daunting for a learner to comprehend these QEPs containing vendor-specific implementation details. In this demonstration, we present a novel, *generic*, and *portable* system called LANTERN that generates a natural language-based description of the execution strategy chosen by the underlying RDBMS to process a query. It provides a *declarative framework* called POOL for subject matter experts (SME) to efficiently create and manipulate natural language descriptions of physical operators of any RDBMS. It then exploits POOL to generate NL description of a QEP by *integrating* rule-based and deep learning-based techniques to infuse language variability in the descriptions. Such NL generation strategy mitigates the impact of *boredom* on learners caused by repeated exposure of similar text generated by rule-based systems.

1 INTRODUCTION

"Database education is at an inflection point" [8]. Due to widespread usage of relational database management system (RDBMS) in the commercial world as well as the growth of Data Science as a discipline, there is an increasing demand of database-related courses in academic institutions. Learners from diverse field aspire to take these courses, even with limited Computer Science backgrounds [8]. Hence, learner-friendly tools that can facilitate learning and understanding of database concepts has the potential to augment the traditional modes of learning (*i.e.,* textbooks, lectures). This is evident from recent activities in the data management community such as research [11, 14], panels [8], and workshops (*e.g.,* https://dataedinitiative.github.io/).

One of the key goal for learners taking a database course is to understand the execution strategies of SQL queries by an RDBMS in practice. Given an SQL query, the query engine in an RDBMS produces a *query execution plan* (QEP), which represents the execution strategy of the given SQL query. Hence, such understanding can be gained by perusing the QEPs of queries. Major database textbooks introduce *general* (*i.e.,* not tied to any specific RDBMS) theories and principles associated with QEPs using natural language-based narratives and visual examples. This allows a learner to gain a general understanding of query execution strategies of SQL queries.

Most database courses complement text book-learning with hands-on interaction with an off-the-shelf commercial RDBMS (*e.g.,* PostgreSQL) to infuse knowledge about database techniques used in practice. A learner will typically implement a database application, pose queries over it, and peruse the associated QEPs to comprehend how they are processed by an industrial-strength query engine. These QEPs are exposed in *visual* or *textual* (*e.g.,* JSON, XML) format. Unfortunately, comprehending these formats in practice can be daunting for many learners as it demands knowledge of vendor-specific implementation details [11, 14]. This problem is further aggravated by the deployment of physical operators with different names by different vendors. In fact, a recent survey with real-world learners reveal that natural language-based descriptions of QEPs (*i.e.,* textbook style) are highly desirable and complements visual tree-based format [14]. Hence, *a learner-friendly tool that generates description of QEPs in natural language (NL) can enable learners to understand the query execution steps of specific queries in practice.*

In this demonstration, we present a novel, *generic*, and *portable* system called LANTERN [14] (naturaL lANguage descripTion of quERy plaNs) to support user-friendly NL-based description of the QEP of an arbitrary SQL query posed on an RDBMS. In particular, given an SQL query, instead of purely visual or semi-structured format of QEP, LANTERN generates *multi-faceted* NL-based description of the steps in the QEP. Under the hood, LANTERN provides a *declarative framework* called POOL to enable *subject matter experts* (SMEs) to create and manipulate NL descriptions (*i.e.,* labels) of physical operators, which are building blocks of QEPs. It implements a NL description generation framework that utilizes POOL to *integrate* rule-based and deep learning-based techniques in order to infuse language variability in the descriptions opportunely. This strategy mitigates the impact of *boredom* on learners that may arise due to repetitive statements in different NL descriptions generated by purely rule-based techniques [14]. In the next section, we elaborate on the design philosophy behind LANTERN.

2 DESIGN PHILOSOPHY

LANTERN is designed to provide learners natural language (NL) description of the QEPs of their relational queries on any RDBMS. Its design is based on the following four principles:

(1) Portability and generalizability of LANTERN: The NL generation framework should be *generalizable*. That is, it should be easily deployable on any domain-specific applications (*e.g.,* movies, hospitals). Furthermore, it should be easily portable across different RDBMS (*e.g.,* PostgreSQL, DB2, SQLServer). This will significantly reduce the cost of its deployment in different learning institutes and environments where different application-specific examples and RDBMS software may be used to teach database systems.

(2) Rich support for SQL of different complexities: LANTERN should be able to generate NL description of *any* SQL statements posed by learners. That is, it should be *orthogonal* to the complexities of SQL queries.

(3) Multi-faceted presentation of NL description: Different learners may wish to view the NL description of a QEP in different modes. Some may simply view it in text while others may wish to view it in visual tree mode where the nodes are annotated with corresponding NL descriptions. Hence, LANTERN should support multi-faceted presentation of the NL description of a given QEP to cater to different needs of learners.

(4) Boredom-conscious NL description generation: A rule-based NL description generation framework (*e.g.,* NEURON [11]) may often result in learners feeling boredom[1] after reading the descriptions for several queries due to the repeated exposure of same/similar text to describe the QEPS [14]. They reported that they started skipping several sentences in the descriptions [14]. This is, in fact, consistent with research in psychology that have found that repetition of text messages can lead to annoyance and boredom [4] resulting in purposeful avoidance [6] and content blindness [7]. Hence, LANTERN should be able to inject diversity in the NL description judiciously in order to mitigate the impact of boredom.

3 SYSTEM OVERVIEW

The architecture of LANTERN is outlined in Figure 1(a). Here we briefly describe the key components. The reader may refer to [14] for details.

3.1 The GUI Module

LANTERN exposes two visual interfaces, one for learners and another for *subject matter experts* (SME). The former is for end users who use LANTERN to view NL descriptions of QEPS. The latter is for database experts who create and manipulate NL descriptions of various physical operators in different RDBMS using a declarative language. Figure 1(b) depicts a screenshot of the *learner view*. The *C1* panel displays the database schema of a specific application of interest. A learner submits his/her query in *C2* panel. Upon clicking on Submit, *C3* and *C4* panels display the NL descriptions in different presentation modes. Clicking on the Compare button shows the differences between NL descriptions generated by rule-based and deep learning-based techniques. The panel *C5* shows some example queries for learners to explore. Panels *C6-C7* support the question-answering framework of NEURON [11]. We show it here for the sake of completeness. In this paper, we do not focus on it as it is already demonstrated in [11]. The Panel *C8* is used to capture comments and feedback from learners. Clicking on Switch DB on the top-right corner enables us to change the underlying RDBMS (*e.g.,* from PostgreSQL to SQL Server) to showcase portability of LANTERN. Clicking on POEM takes us to the *SME view* (*i.e.,* interface for SMES), which we shall discuss in the next subsection.

3.2 POOL Module

No publicly-available QEP-to-NL description datasets exist to train deep learning-based frameworks for generating NL descriptions of arbitrary QEPS. Creating such a dataset is challenging in practice as the natural language labeling of QEPS needs to be performed by trained SMES in order to ensure accuracy of the annotations. Given that there can be numerous QEPS in practice, it is prohibitively expensive to deploy such experts for labeling QEPS.

POOL (Physical Operator Object Language) addresses this challenge by providing a declarative interface to enable SMES to create and manipulate NL descriptions of physical operators in a commercial RDBMS. All QEPS are essentially constructed from this set of operators, which is orders of magnitude smaller than a training set containing QEPS, making it viable to obtain NL descriptions (*i.e.,* labels) from SMES. The NL descriptions of operators involved in a QEP are stitched together automatically by the *boredom-conscious NL description generator module* (see below) to generate the description of the corresponding QEP.

The data model underlying POOL is called POEM (Physical Operator ObjEct Model), which is a simple and flexible graph model where all entities are objects. Each object represents a physical operator of a relational query engine. Each object has a unique *object identifier* (*oid*) from the type *oid*. Objects are either atomic or complex. Atomic objects do not have any outgoing edges. Each object is associated with the following attributes: *source*, *name*, *alias*, *defn*, *desc*, *type*, *cond*, and *target*. The *source* refers to the specific relational engine that an operator belongs to. We can create different operator objects for different RDBMS by changing the *source*. The *name* refers to the name of a physical operator in the source and the *type* captures whether it is an unary or binary operator. *Alias* is an optional alternative name for an operator. The *defn* attribute stores the definition of an operator. The *desc* attribute stores a NL description of the operation performed by an operator. There can be more multiple *desc* associated with an object. The *cond* attribute takes a Boolean value to indicate whether a specified condition (*e.g.,* join condition) should be appended to the NL description of an operator. Values of all attributes are taken from the atomic type string (possibly empty). There is a directed edge between an object pair (p_a, p_c) iff p_a is used to describe p_c. For example, (p_{hash}, p_{hashjoin}) of PostgreSQL has a directed link as the former is used to describe hash join. In such cases, the *target* attribute value of p_{hash} is 'HASH JOIN'.

POOL supports SQL-like statements to manipulate the operator objects. For example, the COMPOSE statement can be used to generate the NL description *template* of an operator. For instance, the template of HASH JOIN operator on PostgreSQL can be generated using it. Specifically, it combines the descriptions associated with the HASH and HASH JOIN operators to generate the following template: *"hash <T> and perform hash join on table <R> and <T> under condition (<C>) to get intermediate table <TN>"* where *"hash <T>"* is the *desc* value associated with the HASH operator. Note that each tag (*e.g.,* <*T*>, <*C*>, etc.) in *desc* acts as a place holder and have a specific meaning as reported in [14]. Particularly, it enables us to make LANTERN orthogonal to any specific application domain. These place holders are replaced by specific relation and attribute names, and predicates associated with a query on a database schema to generate application-specific NL description.

POEM objects are stored in two relations with the following schema: POperators(oid, source, name, alias, type, defn, cond, targetid) and PDesc(oid, desc) as an object may have multiple descriptions. POOL statements are transformed to corresponding SQL statements on these relations. A wrapper is implemented that takes

[1]Watt and Vodanovich [15] describe boredom as a dislike of repetition or of routine.

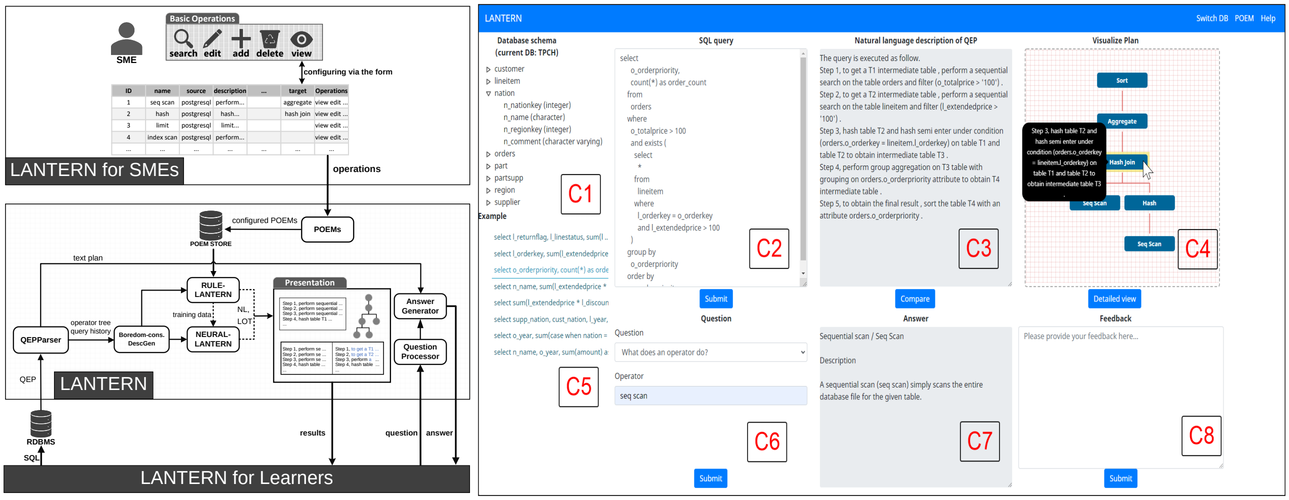

Figure 1: LANTERN: (a) Architecture (left). (b) GUI for learners (right).

the results of SQL queries as input and returns POEM objects or strings as output.

Form-based interaction. We provide an interactive form that encapsulates aforementioned features supported in the POOL. The form provides basic operations over POEM, such as search, edit, add and deletion. These operations are converted internally into corresponding POEM statements.

3.3 Boredom-conscious NL Description Generator Module

Given an SQL query Q from a learner, this module is responsible for generating the natural language description of the QEP of Q. To this end, it realizes the following components.

QEPParser. It first converts the QEP into a *physical operator tree* (*operator tree* for brevity), which is exploited by subsequent modules. When a user submits an SQL query through the frontend, it retrieves and parses the corresponding QEP to construct an operator tree. Each node in an operator tree contains relevant information associated with a plan such as the physical operator (*e.g.,* HASH JOIN), name of the relation being processed by the node, alias given to intermediate results (*e.g.,* subqueries), column(s) used for grouping or sorting, name of the index being processed by the node, subplan ids, filtering conditions used during a join or a table scan, predicates used for index-based search, and the number of rows left after an operation, etc. Next, it records the physical operators involved in the QEP (the same operator that appears multiple times in the same QEP is only counted once).

RULE-LANTERN. It realizes a rule-based framework that leverages the narration (descriptions by SMEs) of various operators defined using POOL to generate a NL description of the QEP of an SQL query [14]. It first extends the operator tree to a *language annotated operator tree* (LOT) by annotating the nodes with corresponding NL descriptions from POEM store and assigning a unique *identifier* to the output of each operator (*i.e.,* intermediate results) so that it can be appropriately referred to in the translation of its parent. Then, it traverses the LOT in a post-order manner to generate a sequence of steps containing the NL description by replacing the place holders in NL templates/descriptions associated with operators with corresponding values.

NEURAL-LANTERN. Recall that NL descriptions of RULE-LANTERN can be repetitive and lack variability that may cause boredom among learners [14]. NEURAL-LANTERN is a deep learning-based framework designed to mitigate this challenge. To address the paucity of training data, it adopts Kipf *et al.* [10] to generate a set of SQL queries given a particular schema and database instance. A collection of QEPs corresponding to these queries are then generated. Each QEP is decomposed into a set of *acts*, each of which corresponds to a set of operators in an operator tree (*i.e.,* subtree). For each act, RULE-LANTERN is invoked to generate the corresponding NL description. Then for each result, it infuses language variability in the generated description by exploiting a group of paraphrasing tools [1–3] and pretrained word embeddings (*e.g.,* [5, 12]). to acquire a set of synonymous sentences. As a result, the number of training samples in our datasets is enlarged by approximately 3X.

The translation model of NEURAL-LANTERN follows the *Seq2Seq* structure [13]. The encoder RNN encodes each word in an act into the corresponding hidden state using an LSTM layer. It uses an LSTM decoder with an attention mechanism to let the decoder focus on the relevant portion of the encoder while generating a token. It adopts both static (*Word2Vec* and *GloVe*) and contextual word embeddings (*ELMo* and *BERT*) in decoder.

Boredom-conscious NL description generator. This component integrates the aforementioned components to implement boredom-conscious NL description generation for a given QEP P. Specifically, the goal is to track the *repetition rate* of physical operators in a user's query history over a time duration T days (by default $T = 1$) and select an appropriate NL description generation scheme (either RULE-LANTERN or NEURAL-LANTERN) based on it. It computes the ratio of the number of operators in P to the total number of operators contained in all QEPs in the user's query history. If the ratio is below a predefined *repetition threshold* (40% by default), it invokes RULE-LANTERN to generate the NL description

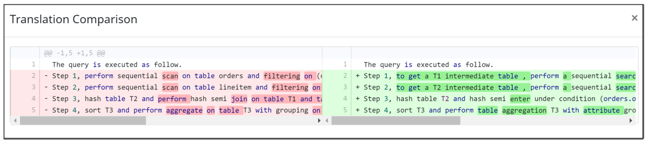

Figure 2: Translation comparison (left: RULE-LANTERN, right: NEURAL-LANTERN.

of P. Otherwise, it invokes NEURAL-LANTERN to generate it, which infuses variability in the description. This avoids a learner viewing repetitive text after submitting several queries within T, thereby mitigating boredom as reported in a user study in [14].

3.4 Presentation Module

Finally, this module is responsible for displaying the NL description of a QEP in multiple modes as follows: (a) *Document view*: The NL description is displayed in form of a text document (*e.g.,* Panel *C3* in Figure 1(b)). (b) *Annotated visual tree view:* This mode *integrates* the visual tree view of a QEP with the NL description output. Specifically, the visual operator tree is shown by default and the NL description corresponding to each step is added as an annotation to the relevant node in the tree (Panel *C4*). A user can view the NL description of an operation by simply clicking on the corresponding node in the tree. (c) *Comparative view:* In this mode, a learner can comparatively view the differences between document views of RULE-LANTERN and NEURAL-LANTERN. The differences in the descriptions are highlighted in the text (*e.g.,* Figure 2).

4 RELATED SYSTEMS

Translating natural language queries to SQL have been studied for decades [9]. LANTERN compliments these efforts by providing a natural language description of a QEP. Most germane to this work is NEURON [11], which is a rule-based system to generate NL descriptions of QEPs. Broadly, NEURON only realizes the second design principle in Section 2 whereas LANTERN achieves all four. More specifically, LANTERN differs from it in the following key ways. Firstly, instead of exploiting purely rule-based solution that may give rise to boredom among learners due to similar/same descriptions, LANTERN integrates rule-based and deep learning-based solutions to inject variability in the NL descriptions opportunely. Secondly, NEURON is tightly integrated with PostgreSQL and hence is not portable across RDBMS. LANTERN implements a declarative framework called POOL for labeling physical operators of different RDBMS to enhance its portability. Thirdly, [11] demonstrated the NL description of a query in document-style text mode only. LANTERN is more flexible by implementing multi-faceted visualization of the NL descriptions. Consequently, all key modules of LANTERN (*i.e.,* POOL, boredom-conscious NL description generator, and presentation modules) are novel and have not been demonstrated in any prior venues.

5 DEMONSTRATION PLAN

LANTERN is implemented in Python. Our demonstration will make use of the TPC-H benchmark and IMDB datasets as representation of two different applications. We shall use two different RDBMS, PostgreSQL and SQL Server, to demonstrate the portability of LANTERN. To this end, the *QEPParser* component generates the operator trees for both these RDBMS. By default, LANTERN uses PostgreSQL. The datasets are replicated on both these RDBMS for ease of demonstration. The audience can pose their own ad-hoc queries on these datasets. The goal of our demonstration is to allow the audience to experience the following interactive features of LANTERN. A **video** to show the demonstration plan of LANTERN using example use cases is available at https://youtu.be/0pNOOnR-Ezk.

Boredom-conscious NL description generation and exploration. An audience can generate and explore NL description of the QEP of his/her query through the *learners view* (Figure 1(b)). One can select a database schema in panel *C1* and input an SQL query in *C2*. She may also select one from the example queries in *C5*. The panel *C3* displays the NL description of her query in document-style view. In particular, the audience will be encouraged to fire several queries and peruse the NL descriptions of the corresponding QEPs to appreciate similar descriptions generated by RULE-LANTERN as well as how NEURAL-LANTERN injects variability in the descriptions. One may click the Compare button to visualize the differences of RULE-LANTERN and NEURAL-LANTERN output (Figure 2). One may also view the NL descriptions in annotated visual tree mode (panel *C4*). One can click the switch DB link to switch to a different RDBMS (*e.g.,* from PostgreSQL to SQL Server). We believe all these interactions will trigger interesting discussions on the benefits of multi-faceted presentation of NL descriptions to aid learning as well as the impact of similar descriptions on boredom.

Label generation and portability using POOL interface. We demonstrate how one can declaratively create labels for physical operators in a different RDBMS (SQL Server) using the form-based interface of POOL. Specifically, clicking on the POEM link takes a user to the POEM GUI. The audience can create and manipulate the NL descriptions of different physical operator objects associated with the specific RDBMS either using form-based interface or by directly writing statements using POOL. After this, we shall return back to the *learner view*. We shall input queries on the database applications hosted on SQL Server now and generate the corresponding NL descriptions of QEPs. This will demonstrate the portability of LANTERN across different RDBMS.

REFERENCES

[1] Paraphrasing tool. https://paraphrasing-tool.com/.
[2] Prepostseo paraphrasing tool. https://www.prepostseo.com/paraphrasing-tool.
[3] Quillbot paraphraser. https://quillbot.com/.
[4] J. T. Cacioppo and R. E. Petty. Effects of Message Repetition and Position on Cognitive Response, Recall, and Persuasion. *Journal of Personality and Social Psychology* ,37, 1: 97-109, 1979.
[5] J. Devlin, M. Chang, K. Lee, K. Toutanova. Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv e-prints*, 2018.
[6] M. R. Hastall and S. Knobloch-Westerwick. Severity, Efficacy, and Evidence Type as Determinants of Health Message Exposure. *Health Communication*, 28, 4: 378-388, 2013.
[7] G. Hervet, K. Guerard, S. Tremblay, M. Saber Chtourou. Is Banner Blindness Genuine? Eye Tracking Internet Text Advertising. *Applied Cognitive Psychology*, 25, 5: 708-716, 2011.
[8] Z. Ives, et al. VLDB Panel Summary: "The Future of Data(base) Education: Is the Cow Book Dead?". *SIGMOD Rec.*, 50(3), 2021.
[9] Hyeonji Kim, Byeong-Hoon So, Wook-Shin Han, and Hongrae Lee. 2020. Natural language to SQL: Where are we today? *Proc. VLDB Endow.* 13, 10 (2020), 1737–1750.
[10] A. Kipf, et al. Learned Cardinalities: Estimating Correlated Joins with Deep Learning. *In CIDR*, 2019.
[11] S. Liu, et al. NEURON: Query Optimization Meets Natural Language Processing For Augmenting Database Education. *In SIGMOD*, 2019.
[12] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, L. Zettlemoyer. 2018. Deep Contextualized Word Representations. *In NAACL*, 2018.
[13] I. Sutskever, O. Vinyals, Q. V Le. Sequence to Sequence Learning with Neural Networks. *In NeurIPS*, 2014.
[14] Weiguo Wang, Sourav S. Bhowmick, Hui Li, Shafiq R. Joty, Siyuan Liu, and Peng Chen. 2021. Towards Enhancing Database Education: Natural Language Generation Meets Query Execution Plans. In *SIGMOD*. ACM, 1933–1945.
[15] J. D. Watt , S. J. Vodanovich. Boredom Proneness and Psychosocial Development. *The Journal of Psychology*, 133/3, pp. 303-314, 1999.