Towards Technology-Enabled Learning of Relational Query Processing

Sourav S. Bhowmick [⋆] and Hui Li [⋄]
[⋆] Nanyang Technological University (NTU), Singapore, assourav@ntu.edu.sg
[⋄] Xidian University, China, hli@xidian.edu.cn

Abstract

*The database systems course has gained increasing prominence in academic institutions due to the convergence of widespread usage of relational database management system (*RDBMS*) in the commercial world, the growth of Data Science, and the increasing importance of lifelong learning. A key learning goal of learners taking such a course is to learn how* SQL *queries are processed in an* RDBMS *in* practice*. Most database courses supplement traditional modes of teaching with technologies such as off-the-shelf* RDBMS *to provide hands-on opportunities to learn database concepts used in practice. Unfortunately, these systems are not designed for effective and efficient pedagogical support for the topic of relational query processing. In this vision paper, we identify novel problems and challenges that need to be addressed in order to provide effective and efficient technological supports for learning this topic. We also identify opportunities for data-driven education brought by any effective solutions to these problems. Lastly, we briefly report the* TRUSS *system that we are currently building to address these challenges.*

1 Introduction

Learning is the acquisition of knowledge or skills through study, experience, or being taught [19]. It is not just listening and accepting what we are taught, but understanding and experiencing them. Education, on the other hand, is the acquisition of knowledge through a process of receiving or giving systematic instruction. Hence, although learning and education are closely related, the former has a broader scope and impact. Specifically, learning can be facilitated through education, personal development, schooling, training or experience. It is not limited to a certain age or period in life. Indeed, while formal education for young adult learners at universities has been the focus of educational provisions in the industrial age, the digital age is now seeing an increased experimentation of "lifelong learning" [6] with provisions such as work-study programmes for early career and mid-career individuals, and digital learning initiatives.

The growing demand for lifelong learning coupled with the widespread use of relational database management system (RDBMS) in the commercial world and the growth of Data Science as a discipline have generated increasing demand of database-related courses in academic institutions. Learners from diverse fields and experiences aspire to take these courses, even with limited Computer Science backgrounds [26]. In a computer science degree program, the key goal of a database systems course is to teach learners how to *build* a database system. On the other hand, the focus of the course in a data science program is to be able to *control* a database system effectively.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

To facilitate both these goals, it is paramount for learners to learn how SQL queries are processed in an RDBMS in practice. Traditionally, this learning goal is achieved through textbooks and lectures. Specifically, major database textbooks [17,36] introduce *general* (*i.e.,* not tied to any specific RDBMS) theories and principles associated with relational query processing and optimization using natural language-based narratives and visual examples. This allows a learner to gain a general understanding of SQL query execution strategies.

It is well-established in education that *effective* use of technology has a positive impact on learning [24]. It causes learners to be more motivated and engaged, thus, enabling them to retain more information. It also increases hands-on learning opportunities. In fact, technology is best used as *"a supplement to normal teaching rather than as a replacement for it"* [24]. Hence, in order to promote effective and efficient learning for diverse individuals in full recognition of the complexity of the topic of relational query processing, *learner-friendly* tools are paramount to augment the traditional modes of learning (*i.e.,* textbook, lecture). Indeed, database systems courses in major institutions around the world supplement traditional style of learning with the usage of off-the-shelf RDBMS. Unfortunately, these RDBMS are not designed for pedagogical support. Although they enable hands-on learning opportunities to build database applications and pose a wide variety of SQL queries over it, very limited effective and efficient learner-friendly support, beyond the visualization of *query execution plans*, is provided for understanding and experiencing the processing and optimization of these queries *in practice* by the underlying relational query engine.

Given the challenges faced by learners to learn SQL [34], there has been increasing research efforts to build tools and techniques to facilitate comprehension of complex SQL statements [15,25,29,30,32,33], automated grading of SQL queries [8], and so on. However, scant attention has been paid to explore technologies that can enable learning of relational query processing [21,31,42]. In this paper, we articulate a vision shaped by the following fundamental questions: (a) What are the key problems that we need to address to facilitate technology-enabled learning of relational query processing in practice? (b) How can the potential solutions to these problems support data-driven education with the goal of making teaching and learning practices more effective and efficient? Specifically, our vision calls for a *learning-centric*, *generic*, and *psychology-aware* approach grounded on the *theories* of motivation and learning to address these challenges. Note that by no means we claim that the list of problems discussed in this article is exhaustive. The pervasive desire here is to galvanize the data management community to explore this nascent inter-disciplinary topic at the intersection of learning sciences and data management by leveraging these problems as the seed.

The rest of the paper is organized as follows. In Section 2, we briefly introduce relevant *motivation theories* that are at the foundation of learning and may potentially impact the design of any learner-centric, technology-enabled solutions for learning. Section 3 introduces the key novel research challenges that need to be addressed in order to realize the vision. We briefly report the TRUSS system which we are currently building to address these challenges in Section 4. The last section concludes this paper.

2 Motivation Theories

Learning is impacted by *motivation*, which is the process that initiates, guides, and maintains goal-oriented behavior [27]. Specifically, motivation impacts how likely a learner is willing to learn. Since effective use of technology in learning has positive impact on motivation, we need to understand motivation theories and their impact on learning. These theories should underpin any effective technological solutions for facilitating learning. In this section, we first briefly describe key theories related to motivation proposed in the domain of education psychology. Next, we highlight how these theories can guide technology-enabled solutions for learning. In the next section, we shall identify novel research issues that are grounded on these theories to facilitate learning of relational query processing.

Motivation theories. Motivation can be broadly characterised into two types, *intrinsic* and *extrinsic* [37]. *Intrinsic motivation* is the act of doing an activity purely for the joy of doing it whereas *extrinsic motivation* is to do something due to the influence of external rewards or punishments (*e.g.,* good grades, jobs). Although the latter is not optimal for learning, research suggests that extrinsic motivation is prevalent [16, 37] and may pave the way to intrinsic motivation. That is, a learner may initiate learning due to external factors but the motivation may morph to an intrinsic one during task engagement.

Regardless of the type of motivation, *Achievement Goal Theory* [7] argues that all motivation are essentially linked to one's goals that can take two forms, namely *performance goals* and *mastery goals*. The former is caused by satisfaction of one's ego (*e.g.,* appearing superior to one's peers) whereas the latter is aligned with intrinsic motivation and is grounded on the pure desire to master a skill or concept. The *Expectancy Value Theory* [43] postulates that expectation and value of learning a skill or concept have direct impact on the performance and task choice of a learner. To elaborate further, an individual's effort and performance for a task are influenced by their expectation of success or failure. Note that expectations and values themselves are influenced by how a learner assess their competency and perceived task difficulty. If a learner has felt satisfaction in undertaking a similar task in the past, then it is more likely they will put effort to the current task. However, if past experience shows the task is either too difficult to be completed or not sufficiently difficult enough then they may not engage with it. The *Flow Theory* [35] describes a psychological state in which an individual is purely intrinsically motivated to learn without any external factors. Such state is said to occur when the task is neither too difficult to cause helplessness or frustration nor too easy to make a learner bored.

Motivation theories in technology-enabled learning frameworks. Every learner has intrinsic or extrinsic motivation to reach their learning goals. Hence, any technology-enabled learning framework should be cognizant of the above theories in order to cultivate motivation to learn. Specifically, technologies to supplement learning of relational query processing and optimization should support the followings:

- Learners may learn relational query processing due to intrinsic or extrinsic motivation. It is important that any technological framework support both and facilitate transformation of extrinsic motivation to intrinsic one during the course of interacting with it.
- Increase learners' satisfaction of successful completion of the task of learning relational query processing as well as facilitate flow state of learners to move to the Goldilock's zone (*i.e.,* learning relational query processing concepts is neither too difficult nor too easy).

3 Research Issues

In this section, we outline novel *learner-centric* research issues in the burgeoning topic of technology-enabled learning of relational query processing. It is worth noting that although technology engages and motivates learners, it is advantageous for learning only when it is aligned to with what is to be learned [24]. Hence, the issues we focus for technological support are learning about query execution plans, exploration of *alternative query plans* in a plan space, and cost estimation of a physical query plan. Observe that all these issues are typically covered by a database systems course. A common theme that cuts across these issues is the pervasive desire for motivation theory-aware tools and techniques that aim to *supplement* existing off-the-shelf RDBMS to facilitate learning of these issues. We begin with a brief background on relational query plans that learners typically encounter in a database systems course.

3.1 Query Plans

Given an arbitrary SQL query, an RDBMS generates a *query execution plan* (QEP) to execute it. A QEP consists of a collection of physical operators organized in form of a tree, namely the *physical operator tree* (*operator tree* for brevity). Figure 1(a) depicts an example QEP with a collection of physical operators. Each physical operator, *e.g.,* `SEQUENTIAL SCAN`, `INDEX SCAN`, takes as input one or more data streams and produces an output one. A QEP

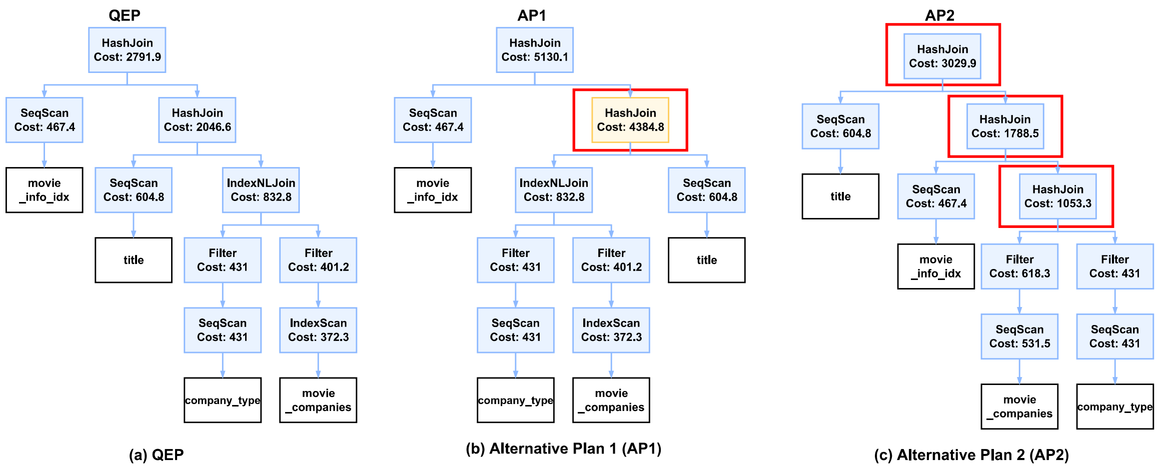


Figure 1: Examples of QEP and alternative query plans.

explicitly describes how the underlying RDBMS shall execute the given query. Notably, given an SQL query, there are many different query plans, other than the QEP, for executing it. We refer to these different plans (other than the QEP) as *alternative query plans* (AQP). Figures 1(b)-(c) depict two examples of AQP.

3.2 Understanding Query Execution Plans (QEPs)

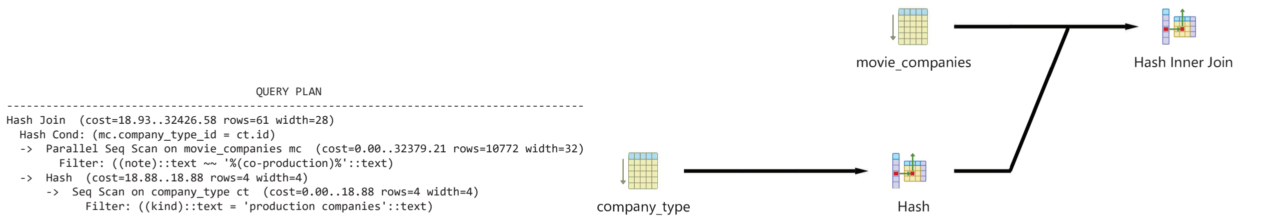
A key goal of learning relational query processing is to understand how SQL queries are processed in an RDBMS in practice. This can be achieved by understanding the content of the QEP of a given query. Major database textbooks (*e.g.,* [17, 36]) typically illustrate QEPs of simple SQL queries, their costs, and adverse impact on estimated cost if alternative physical operators are chosen (*e.g.,* merge join instead of hash join) for a QEP. However, they are not interactive and the variety of examples they can discuss is constraint by the page limit and cost. Hence, only a very few simple, static examples of QEPs are typically exposed to learners.

Off-the-shelf RDBMS (*e.g.,* PostgreSQL) provide the opportunity to greatly mitigate the limitations of textbooks. A learner may implement a database application in an RDBMS, pose queries over it, and peruse the associated QEPs to comprehend how they are processed by an industrial-strength query engine in practice. Such interactivity provides learners the freedom to formulate a large number of queries with diverse complexities and view the contents of corresponding QEPs in real-time.

Most existing RDBMS expose the QEP of an SQL query using *visual* or *textual* (*e.g.,* unstructured text, JSON, XML) format. Unfortunately, comprehending these semistructured textual formats to learn about query execution strategies of SQL queries in practice can be daunting for learners. In contrast to natural language-based narrations in database textbooks, they are not user-friendly and assume deep knowledge of vendor-specific implementation details. On the other hand, the visual format is relatively more user-friendly but hides important details. Consequently, in consistent with the *Expectancy-Value Theory*, the task difficulty may become a barrier for some learners to learn about query execution strategies in a specific RDBMS from these QEP formats.

Example 1: Doreen is an undergraduate student in a data science program who is currently enrolled in a database course. She wishes to understand the execution steps of the following SQL query in PostgreSQL on the IMDb benchmark dataset [1] by perusing the corresponding QEP in Figure 2(a) (partial view).

```
SELECT mc.note AS production_note,
   t.title AS movie_title,
   t.production_year AS movie_year
FROM company_type AS ct,
```



```
                                     QUERY PLAN
------------------------------------------------------------------------------------------
Hash Join  (cost=18.93..32426.58 rows=61 width=28)
  Hash Cond: (mc.company_type_id = ct.id)
  ->  Parallel Seq Scan on movie_companies mc  (cost=0.00..32379.21 rows=10772 width=32)
        Filter: ((note)::text ~~ '%(co-production)%'::text)
  ->  Hash  (cost=18.88..18.88 rows=4 width=4)
        ->  Seq Scan on company_type ct  (cost=0.00..18.88 rows=4 width=4)
              Filter: ((kind)::text = 'production companies'::text)
```

Figure 2: A QEP in PostgreSQL and its visual tree representation.

```
  movie_companies AS mc,
  movie_info_idx AS mi_idx,
  title AS t
WHERE ct.kind = 'production companies'
  and mc.note like '%(co-production)%'
  AND ct.id = mc.company_type_id
  AND t.id = mc.movie_id
      AND mc.movie_id = mi_idx.movie_id;
```

Unfortunately, Doreen finds it difficult to mentally construct a narrative of the overall execution steps by simply perusing it. This problem is further aggravated in more complex SQL queries. Hence, she switches to the visual tree representation of the QEP as shown in Figure 2(b). Although relatively succinct, it simply depicts the sequence of operators used for processing the query, hiding additional details about the query execution (*e.g.,* sequential scan, join conditions). In fact, Doreen needs to manually delve into details associated with each node in the tree for further information. ∎

Since natural language (NL)-based narratives aided with visual examples (as in textbooks and lectures) have been the traditional mode of learning for decades, we advocate that an intuitive natural language (NL)-based description of a QEP can greatly augment learning of the execution strategies of SQL queries by an RDBMS. The intuition is that NL-based descriptions may facilitate the flow state of learners (*Flow Theory*) to the Goldilock's zone. This can then effectively complement the current visual tree format generated by existing RDBMS. Specifically, a learner may either use the visual QEP to get a quick overview and then peruse the NL description or study them in parallel to acquire detailed understanding. To support this hypothesis, we surveyed 62 and 56 unpaid volunteers taking the undergraduate database course in NTU in two semesters (2019 and 2020). We use the TPC-H v2.17.3 benchmark and a rule-based natural language generation tool for QEPs [31] to generate natural language descriptions of QEPs for SQL queries formulated by the volunteers. The volunteers were asked: *"which query plan format they prefer for learning?"* 53.2% and 55.4% preferred NL-based description, respectively. Very few (3% and 5.4%, respectively) preferred the text format. Hence, there is clear evidence that learners prefer to use NL-based and visual tree-based formats for learning about QEP.

The majority of NL interfaces for RDBMS [28], however, have focused either on translating natural language sentences to SQL queries or narrating SQL queries in a natural language. Scant attention has been paid for generating natural language descriptions of QEPs [31, 42], which is a challenging problem. Although deep learning techniques, which can learn task-specific representation of input data, are particularly effective for natural language processing, it has a major upfront cost. These techniques need massive training sets of labeled examples to learn from. Such training sets in our context are prohibitively expensive to create as they demand database experts to translate thousands of QEPs of a wide variety of SQL queries. Even labeling using crowdsourcing is challenging as accurate natural language descriptions demand experts who understand QEPs. Note that accuracy is critical here as low quality translation may adversely impact individuals' learning.

3.3 Learning Impact of Alternative Choices on QEP

Natural language description of a QEP enables a learner to understand the execution steps of a query. This may pique the interest of a learner to raise further questions related to query processing centred around a specific QEP. Since major database textbooks typically discuss the adverse impact of choosing alternative physical operators or join ordering on the estimated cost, a learner may also like to delve deeper into the impact of these alternative choices on the estimated query processing cost of their queries.

Example 2: Reconsider Example 1. Doreen's course lectures and textbook discuss the impact of physical operator choices and join ordering on the selection of a QEP. Hence, after perusing the content of the QEP, she wonders what will be the impact on the cost be if the hash join is replaced by a merge join? Is the estimated cost of the alternative query plan substantially higher compared to the QEP? How much is the impact on the estimated cost if the join ordering is changed? In this context, a narrative that explains why the QEP is chosen by connecting its content with knowledge garnered from database textbooks will greatly benefit her learning. ∎

Unfortunately, as stated earlier, off-the-shelf RDBMS are not developed for pedagogical support. Typically, they do not expose the impact of alternative choices of various physical operators or join ordering on the QEP in a *user-friendly* manner to aid learning. Note that such information is invaluable to learners as it not only facilitates hands-on inquire-driven learning on the impact of a choice of a physical operator or a specific join ordering on the estimated cost of a QEP but it also enables them to comprehend why a QEP is chosen by the underlying RDBMS. However, an RDBMS typically demands a learner to manually pose SQL queries with various constraints on *configuration parameters* (*e.g.*, `enable_hashjoin`, `enable_nestloop` in PostgreSQL) to view the corresponding QEP containing specific physical operators. Furthermore, one has to manually compare the generated plan with the original QEP to understand the impact. Notably, a database course may not introduce these configuration parameters while exposing syntax and semantics of SQL. It is also impractical to assume that learners will be familiar with them when many are taking the course for the first time. Clearly, based on the *Expectancy-Value* and *Flow* theories, a learner-friendly framework that can facilitate exploration of the impact of various physical operators and join ordering on a QEP can greatly motivate learners to deeper engagement and learning of this topic.

Intuitively, given an SQL query and learner-specified *preferences* (*e.g.,* merge join, index scan, specific join ordering), the goal is to automatically visualize the impact of these choices on the selected QEP. In this context, it is important to generate a natural language-based explanation that goes beyond the conventional least-cost-based explanation to connect established knowledge related to usage scenarios of different physical operators from textbooks with the specified preferences. For instance, examples of some established knowledge are: (a) index scan is the optimal access path for low selectivity whereas sequential scans perform better in high selectivity [11]; (b) merge join is preferred if the join inputs are large and are sorted on their join column [2]; (c) nested-loop join is ideal when one join input is small (*e.g.,* fewer than 10 rows) and the other join input is large and indexed on its join columns [2]. Such knowledge in the form of explanations will naturally facilitate learners' understanding of relational query processing. For instance, consider Example 2. Explanations such as (b) will help Doreen to understand why a hash join was chosen by the relational query engine.

The problem is challenging from several fronts. While under-the-hood it is straightforward to generate an SQL query involving the learner-specified preferences and retrieve the corresponding QEP, automatically generating appropriate visualization framework to aid learning is challenging. First, how should the results be presented in consistent with motivation theories to motivate learners to explore and learn? Note that a learner may want to view the impact of multiple physical operators and join ordering together instead of just a single operator or join ordering. Simply generating an NL description of the AQP is insufficient since this will demand a learner to manually compare the description of the original QEP with it in order to understand the impact of various operators and join ordering. Naturally, this becomes tedious especially for complex queries. Second, how can we generate NL explanations that augment learning by connecting with textbook knowledge? It demands sophisticated text extraction, analytics, and summarization framework that connects the alternative query plans with relevant established knowledge embedded in online resources.

3.4 Exploration of Informative Alternative Query Plans

In the preceding challenge, a learner has clear preferences that they wish to explore with respect to a QEP. However, this may not always be the case. Some learners may not have clear idea of what alternative query plans they are interested in.

Example 3: Meng is another undergraduate student pursuing a degree in computer science and a classmate of Doreen in the database course. He also formulates the query in Example 1. After viewing the QEP, he wonders what are the different alternative query plans (AQPs) considered by the underlying RDBMS during the QEP selection process. Specifically, are there alternative plan(s) that have similar (resp. different) structure and physical operators but very different (resp. similar) estimated cost? If there are, then how they look like? ∎

Off-the-shelf RDBMS do not expose a *representative* set of alternative query plans considered by the underlying query optimizer during the selection of a QEP in a *user-friendly* manner to aid learning. Hence, due to the lack of easy access to such information in RDBMS, based on the *Expectancy-Value Theory*, learners may restrict themselves to the simple and limited number of examples that are typically exposed in textbooks and lectures or simply abandon the effort. Clearly, a learner-friendly framework that can facilitate retrieval and exploration of "informative" alternative query plans associated with a given query can greatly aid in answering Meng's questions related to the query optimization process.

Selecting a set of *informative* AQPs to facilitate learning is a technically challenging problem. First, what is an "informative" AQP in the context of learning? To elaborate further, reconsider Example 3. Figures 1(b)-(c) depict two alternative plans for the query where the physical operator/join order differences are highlighted with red rectangles and significant cost differences are shown using yellow nodes. Specifically, *AP1* has very similar structure as the QEP but different join order involving `title` and `company_type` relations and significantly different estimated cost. *AP2*, on the other hand, displays similar estimated cost as the QEP but different join order. *Which of these alternative plans should be revealed to Meng?* The overarching goal here is to choose alternative plan(s) that may enhance Meng's knowledge of the QEP selection process (*i.e.,* informative) as well as motivate him to learn and explore. Certainly, any *informativeness* measure needs to be cognizant of plans that a learner have already viewed for her query (including the QEP) in order to avoid the exposure of highly similar information. It should also facilitate retrieval of plans that learners may be interested in as far as query optimization is concerned. Hence, it is paramount to take feedback from learners on the *types* of AQPs that are potentially of interest to them and devise a mechanism to quantify *informativeness* of a plan by mapping the knowledge acquired from the feedback to a *utility* measure. Subsequently, we need to design techniques that can select informative plans that *maximize* the *utility* as we cannot simply rely only on the estimated cost of alternative plans. Second, the number of candidate AQPs for a given SQL query is exponential in the worst case [13]. Hence, it is prohibitively expensive to scan all these plans to select informative ones. Note that the selection of AQP cannot be integrated into the plan enumeration step of the underlying query optimizer. We need to know the QEP when computing AQP as they are selected with respect to the QEP a learner has seen.

At first glance, it may seem that we can select $k > 1$ alternative query plans where k is a value specified by a learner. Although this is a realistic assumption for many top-k problems, learners may not necessarily be confident to specify the value of k always. They may prefer to *iteratively* view one plan-at-a-time and only cease exploration once they are satisfied with the understanding of the query optimization process for a specific query. Hence, k may not only be unknown *apriori* but also the selection of an AQP at each iteration to enhance learning of different plan choices depends on the plans viewed by a learner thus far. Clearly, it does not increase learners' understanding of the query optimization process or motivate them to use the framework if a plan with highly similar information of an already viewed plan is revealed to them in the subsequent iterations. This demands for a flexible solution framework that can select informative AQPs in absence or presence of the k value.

3.5 Understanding Cost Estimation of a Physical Query Plan

The preceding subsections introduce research issues that aim to facilitate learning of the execution strategy of an SQL query and interesting plan choices a relational query optimizer makes in practice. Another key knowledge that a learner needs to acquire is the cost estimation procedure of these plans.

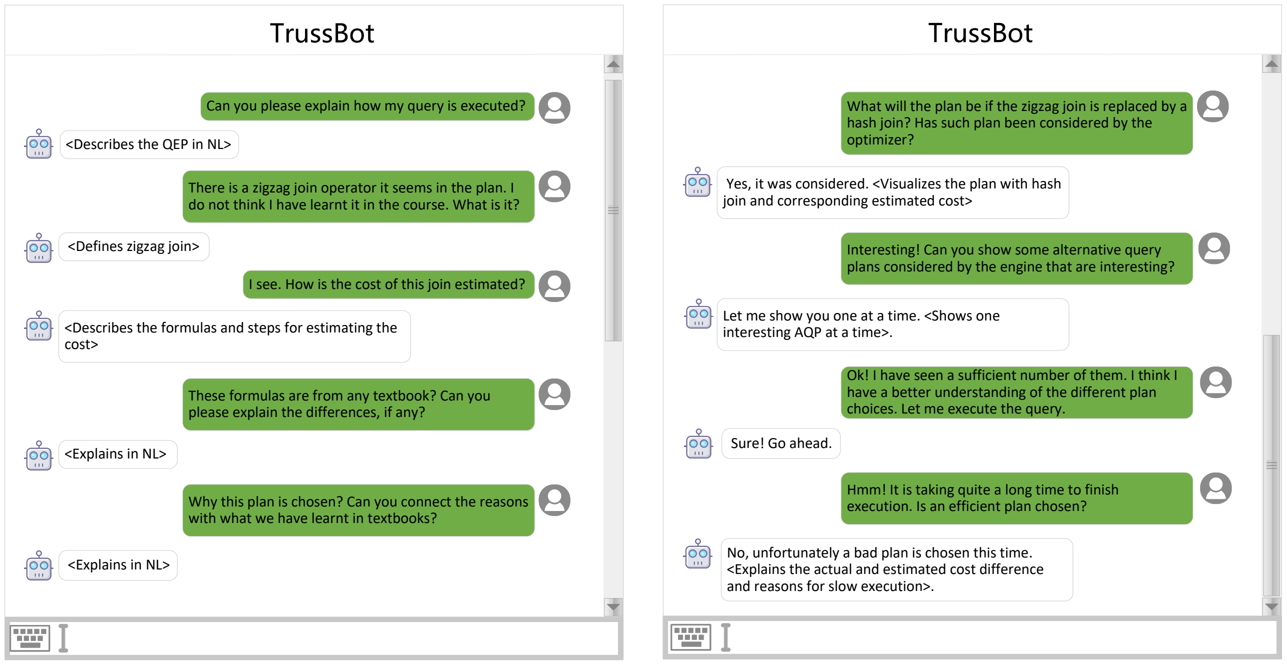
Example 4: Doreen and Meng have learnt from textbooks and lectures how the cost of a physical query plan can be estimated. However, the number of queries considered in these modes of learning and their complexities are limited. They are motivated to experience cost estimation of plans associated with a wider variety of queries. Hence, they pose several queries with different degrees of complexity on the IMDb dataset in PostgreSQL. They can view the overall estimated cost of a QEP as well as cost of different subtrees (*e.g.,* Figure 1). However, they cannot view step-by-step details of the input parameters and the formulas used by the underlying query optimizer to compute these numbers. For instance, in Figure 1(a), why is the cost of the first `HASH JOIN` *2046.6*? By undertaking a back-of-the-envelope calculation using formulas learnt in the course, they could not replicate this value. Are some of the principles and formulas to compute cost different from what they have learnt from textbooks and lectures? If so, then why?

Doreen and Meng also wonder what the intermediate result sizes of different operations are here? When they execute one of the queries, they have to wait for a considerable amount of time to view the results. Does the estimated time cost of the QEP differ significantly from the actual cost? Why? ∎

Existing RDBMS do not provide any learner-friendly support to facilitate such learning. Consequently, based on motivation theories, learners may not pursue this direction of inquiry using an RDBMS, surrendering valuable opportunity for hands-on acquisition of knowledge of the cost estimation process. It is challenging, however, to expose an interface to facilitate such learning and exploration. First, it demands automated analysis of the code base of the underlying query optimizer to extract various formulas used for cost estimation. These formulas may not necessarily be identical across all RDBMS or textbooks. For instance, in [17], the cost of a selection involving inequality condition is approximated to be 1/3 of the input size independent of the selection condition. On the other hand, in [36], more accurate measure is used for estimating the selection cost. Furthermore, a specific RDBMS may implement variants of these formulas. Second, it is paramount to connect these formulas with specific input parameters for a query to reveal how the cost of a plan is estimated while emphasizing the similarity and differences with textbook knowledge. A framework that can support this in a palatable manner to facilitate learning is non-trivial as it may demand a sophisticated natural language generation framework that connects analysis of the code base with textbook content. Third, superior visualization and NL-based framework are necessary to explain to learners the reasons for the differences in estimated and actual cost of a query. Although tools such as [21] allow one to visualize the cost of different plans over the plan space, they are not designed for explaining the *cost differences* for a *specific* query in a palatable manner.

3.6 A Unifying Framework: Chatting with a Relational Query Engine

Although addressing the aforementioned issues has the potential to facilitate learning of relational query processing by providing learner-friendly platforms, a set of isolated platforms that address these different issues will make it cumbersome for learners to navigate and take advantage of them. For instance, learners may find it overwhelming to operate three independent technology-enabled learning platforms targeting NL description generation of QEP, exploration of AQP, and cost analysis of query plans, respectively. This may deter them to use such technology for learning. Given that young adults often interact through chat apps (*e.g.,* Whatsapp, WeChat), a natural language interaction framework (*i.e.,* chatbot) that can unify these solutions may bring practical benefits to learners. A


TrussBot

Can you please explain how my query is executed?

<Describes the QEP in NL>

There is a zigzag join operator it seems in the plan. I do not think I have learnt it in the course. What is it?

<Defines zigzag join>

I see. How is the cost of this join estimated?

<Describes the formulas and steps for estimating the cost>

These formulas are from any textbook? Can you please explain the differences, if any?

<Explains in NL>

Why this plan is chosen? Can you connect the reasons with what we have learnt in textbooks?

<Explains in NL>

TrussBot

What will the plan be if the zigzag join is replaced by a hash join? Has such plan been considered by the optimizer?

Yes, it was considered. <Visualizes the plan with hash join and corresponding estimated cost>

Interesting! Can you show some alternative query plans considered by the engine that are interesting?

Let me show you one at a time. <Shows one interesting AQP at a time>.

Ok! I have seen a sufficient number of them. I think I have a better understanding of the different plan choices. Let me execute the query.

Sure! Go ahead.

Hmm! It is taking quite a long time to finish execution. Is an efficient plan chosen?

No, unfortunately a bad plan is chosen this time. <Explains the actual and estimated cost difference and reasons for slow execution>.

Figure 3: An example of interaction between a learner and the chatbot.

possible interaction between a learner and a hypothetical chatbot designed for interacting with a relational query engine are shown in Figure 3.

Building a high-quality chatbot for a relational query engine to facilitate learning is non-trivial and challenging. In addition to the challenges mentioned earlier for addressing individual components, a chatbot brings new challenges with respect to correct parsing and interpretation of a learner's statement, constructing correct (syntactically and semantically) responses in a natural language, engaging learners in conversations, and so on. While these challenges are long recognized in building a generic chatbot [20], the domain-specific nature of the problem brings in interesting flavor to it. Different from natural conversations, a learner's questions usually have concrete objectives, actively requesting information, and an answer to every question should facilitate understanding of relational query processing. Furthermore, a learner's questions at each step are not only closely related to the chatbot's current answers, but also need to take into account the context of the previous parts of the conversation. For example, consider the first three questions in Figure 3 from the learner. These questions are actively requesting information related to relational query processing. Observe that the third question is related to the preceding parts of the conversation.

Any chatbot needs to consider two kinds of information in a learner's question: (1) the intent of the question (*e.g.,* understanding cost computation) and (2) the content of the question (*e.g.,* cost computation steps of zigzag join in a QEP). To this end, we can construct a *query processing knowledge graph* semi-automatically to represent a collection of relational query processing concepts. Then the intent and content can be determined by *mapping* the question to different concepts in the knowledge graph. Note that similar idea of knowledge graph has been recently exploited in the context of question generation for multi-party court debates for judicial education [44]. Once the intent and content of a question are determined, the chatbot invokes the relevant component (Section 3.2-3.5) to retrieve the answer for the specific question. The result returned by it is then *transformed* into a natural language (supported by visual representations, if necessary) and presented to the learner.

Table 2: Learning-centric issues.

Issue	Questions to address
Rational for the impact of technologies on learning	Will learners work more efficiently, more effectively, more intensely? Will the technology help them to learn for longer, more deeply, more productively?
Role of technology in learning	Will it help learners to gain access to learning content? Will the technology provide feedback?
Technology should support effective interaction for learning	Does it support effective interaction with learners?
Identify what learners will stop doing	What it will replace or how the technology activities will be additional to what learners would normally experience?

3.7 Learning-centric, Generic, and Psychology-Awareness of Solutions

In addition to the challenges within each aforementioned issues, any solution to them must ensure the following features.

- **Learning-centric.** The role, impact, and interaction of the platforms designed to address aforementioned issues have to be learning-centric, *i.e.,* they bring about improvement in learning. Table 2 lists the learning-centric issues [24] that any technology-enabled solution needs to address. For instance, consider the last issue. An effective solution to NL descriptions of QEPs will provide learners an additional interface to learn about query execution strategies to what they would normally experience. Similarly, consider the second issue. A solution to user-friendly exploration of AQPs will enable learners to gain easy access to informative plans to aid learning of the query optimization process.
- **Generalizability.** Solutions must be *generalizable* to different RDBMS and applications. This will significantly reduce the cost of its deployment in different learning institutes and environments where different application-specific examples and RDBMS may be used to teach database systems. For example, the natural language generation framework should be generalizable. Ideally we would like to generate natural language descriptions of QEPs using one application-specific dataset (*e.g.,* movies) and then use it for other applications (*e.g.,* hospital) on any off-the-shelf RDBMS.
- **Psychology-awareness.** Any technology-enabled learning framework has to be *learner-centric*, *i.e.,* it has to be cognizant of the psychology of learners. Any deployable solution has to be palatable and engaging to learners so that they are motivated to learn and explore. Hence, these solutions need to be consistent with various cognitive psychology and motivation theories to have practical impact. For example, the NL descriptions for different queries must not use the same language to describe various operations in QEPs. Similarly, highly similar AQPs should not be exposed to the learners. Otherwise, learners may feel bored after viewing several AQPs or reading the NL descriptions for several queries. In fact, this is consistent with research in psychology that have found that repetition of messages can lead to annoyance and boredom [12] resulting in purposeful avoidance [22], content blindness [23], and even lower motivation [38].

3.8 Towards Data-driven Education

As remarked in Section 1, learning can be facilitated by education. Hence, technological platforms that address the aforementioned challenges may pave the way for *data-driven* education due to rich access to *interaction log* data of learners. Such log data may consists of access times of learners, history of queries formulated by learners, temporal information related to various interactions, among others. This provides a rich data source for building data-driven techniques to facilitate education by analyzing these data at both individual and group levels and correlating them with the performances of learners in tests (*i.e.,* academic outcomes). A non-exhaustive list of questions that can be answered by exploiting the log data to facilitate data-driven education is as follows:

- How do the type and complexity of SQL queries posed by learners evolve over time? What are the activity

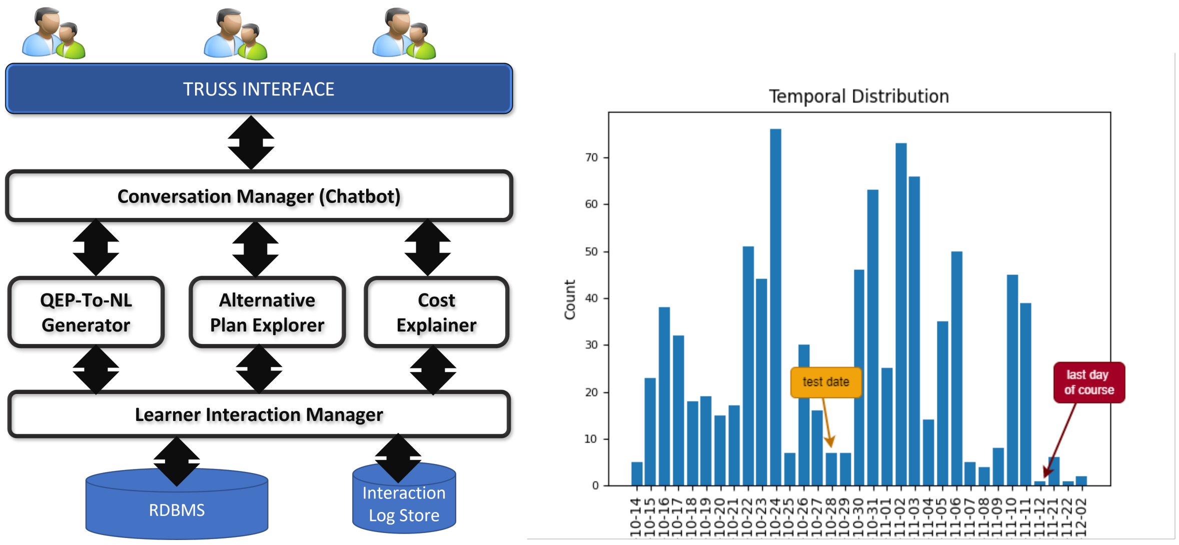


Figure 4: (a) Architecture of TRUSS (left); (b) No. of queries versus time (right).

patterns of learners during a semester? Answers to these questions may provide insights on motivation and learning habits of learners.

- How do learners learn relational query processing? Numerous studies in cognitive psychology show that *spacing* (*i.e.,* distributing practice over more sessions) significantly improves long-term learning compared to *massing* (*i.e.,* practice in longer sessions) [9, 10, 39, 41]. The interaction data may enable us to build models to predict learners demonstrating massing, thereby enabling timely intervention to nudge them to more effective learning habits.
- Research in education posits that technology can be used effectively as a short but focused intervention to improve learning especially when there is regular and frequent usage over a period of several weeks [24]. In our context, it is expected that the platforms are also for focused usage over a period of few weeks. Do they help learners to perform better in tests and coursework? Is there any correlation between frequency of engagement with a platform and performance? Answer to this may provide data-driven insights to the effectiveness of these tools in learning.
- Do learners continue to use the platforms even after the end of a database course? This may indicate intrinsic motivation to learn relational query processing.
- Can the queries posed by learners over time shed light on the difficulties they face with respect to the learning and understanding of relational query processing and optimization? Answer to this question may facilitate the design of more effective and efficient pedagogical strategies to improve effectiveness of teaching.

In summary, addressing the aforementioned research issues provide us a unique opportunity to take a data-driven approach to the education of relational query processing that may otherwise be infeasible through traditional mode of teaching.

4 The TRUSS System

Figure 4(a) depicts the high-level architecture of the TRUSS (**T**echnology-enabled Lea**R**ning of Q**U**ery Proce**SS**ing) system that we are currently building to address the challenges introduced in the preceding section. The *QEP-to-NL Generator*, *Alternative Plan Explorer*, and *Cost Explainer* components aim to address the challenges in

Sections 3.2, 3.3 & 3.4, and 3.5, respectively. The *Conversation Manager* is to realize the unifying chatbot (Section 3.6) and the *Learner Interaction Manager* is responsible to facilitate data-driven education (Section 3.8). In this section, we briefly describe our recent efforts to build three frameworks, NEURON [31] and LANTERN [14], that aim to address the problem described in Section 3.2 (*i.e., QEP-to-NL Generator*), and MOCHA [40] that takes an initial step to address the problem in Section 3.3 (*i.e., Alternative Plan Explorer*). The reader may refer to [14, 31, 40, 42] for details on these frameworks.

4.1 NEURON and LANTERN

NEURON [5, 31] is the first system that exploits a rule-based interpretation engine to generate NL description for QEP in PostgreSQL. Specifically, given the QEP of a SQL query, NEURON first parses and transforms the QEP of an SQL query into an operator tree where each node contains relevant information associated with a plan (*e.g.,* filter conditions). Next, it traverses the tree and generates a NL description of the node based on NL templates and the information it carries. It also supports a preliminary *natural language question answering* system that allows a user to seek answers to a variety of concepts and features associated with a QEP.

NEURON is a rule-based framework that is tightly integrated with PostgreSQL. Hence, it is not generalizable (Section 3.7). LANTERN [3, 14, 42] addresses this limitation by not only making the solution generalizable but also psychology-aware. It incorporates a *declarative framework* called POOL to empower *subject matter experts* (SMEs) create and manipulate the NL descriptions (*i.e.,* labels) of physical operators, which are the building blocks of QEPs. The data definition in POOL allows one to declaratively create physical operator objects associated with a specific RDBMS. For example, one can create the definition of hash join operator in PostgreSQL (pg) as follows.

```
CREATE POPERATOR hashjoin FOR pg
(ALIAS = null,
TYPE = 'binary',
DEFN = null,
DESC = 'perform hash join',
COND = 'true',
TARGET = null)
```

In particular, the TYPE attribute can take either 'unary' or 'binary' value. The DESC attribute allows one to specify a natural language description of the operation performed by the operator. The COND attribute takes a Boolean value to indicate whether a specified condition (*e.g.,* join condition) should be appended to the natural language description of an operator. Values of all attributes are taken from the atomic type `string` (possibly empty). Note that no relation or condition is specified in DESC. This is because these are added automatically to DESC by exploiting TYPE and COND attributes of an operator. For instance, since TYPE is 'binary' in the above definition, two variables representing join relations will be added automatically to the description of `hashjoin`. Lastly, the TARGET attribute allows one to specify the operator name which is supported by the defined operator. For example, TARGET is set to 'hash join' for the definition of `hash` operator.

The key goals of the data manipulation component of POOL are to provide syntactical means to support (a) retrieval of specific properties (*i.e.,* attributes) of physical operators using SQL-like SELECT-FROM-WHERE syntax, (b) generation of the *template* for natural language description of an operator using the COMPOSE clause, and (c) update properties of physical operator objects using UPDATE and REPLACE clauses. Specifically, the COMPOSE clause uses the *desc*, *type*, and *cond* attributes of operators to generate the template. For example, the template generation for the `hash` operator can be specified as follows.

```
COMPOSE hash FROM pg
```

The above statement will return the template *"hash \$$R_1$\$"*, which can be subsequently used by LANTERN to generate specific description of the `hash` operator in a QEP. Also, observe that R_1 is appended based on the *type* attribute of the hash object. An example to generate the NL description template of the hash join operator is as follows.

```
COMPOSE hash, hashjoin FROM pg
USING hashjoin.desc = 'perform hash join'
```

The above statement generates the following template: *"hash R_1 and perform hash join on R_2 and R_1 on condition $cond$"*.

The update statement can be exploited to assign definition or description of an operator from one commercial database to another, thereby making it more efficient for an SME to specify properties of physical operators. The following example demonstrates how the description of hash join in PostgreSQL is transferred to the hash join operator in DB2.

```
UPDATE db2
SET desc = (SELECT desc
 FROM pg WHERE pg.name = 'hashjoin')
WHERE db2.name = 'hsjoin'
```

It can also be used along with the REPLACE clause to transfer definition or description of an operator object to another within the *same* source. For example, one can transfer the description of hash join to nested loop join by replacing the word 'hash' with 'nested loop' as follows.

```
UPDATE pg
SET desc = REPLACE((SELECT desc FROM pg AS pg2
WHERE pg2.name = 'hashjoin'), 'hash', 'nested loop')
WHERE pg.name = 'nested loop join'
```

Note that the REPLACE clause takes three parameters as input, namely, the description or definition of an operator object, the string in it that needs to be replaced (*e.g.,* 'hash'), and its new replacement string (*e.g.,* 'nested loop').

Once the physical operator objects for different RDBMS are created in POOL and stored, the physical operator tree of a given query in any RDBMS can be augmented by automatically annotating relevant nodes with NL descriptions by leveraging the COMPOSE statement and replacing the place holders in NL templates with specific relations, attribute names, and predicates relevant to the query. The NL description generation framework then utilizes this augmented operator tree and *integrates* a rule-based and deep learning-based techniques. In particular, the latter infuses language variability in the descriptions opportunely. This strategy has been shown to mitigate the impact of boredom on learners that may arise due to repetitive statements in different NL descriptions [42].

4.2 MOCHA

MOCHA (iMpact of Operator CHoices visuAlizer) [4, 40] aids learner-friendly interaction and visualization of the impact of alternative physical operator choices on a selected QEP for a given SQL query. It is built on top of PostgreSQL. Given an SQL query and learner-specified *operator preferences* (*e.g.,* merge join, index scan), MOCHA automatically visualizes the impact of these choices on the selected QEP. Specifically, it exploits the *planner method configuration*[1] feature of PostgreSQL to generate AQPs based on a user input. The configuration parameters in this feature provide a way to enforce the query optimizer to choose a query plan with certain user-specified physical operators. By default, all parameters are turned on during query processing. A query request is sent to PostgreSQL using the default settings to retrieve the QEP of a query.

[1] www.postgresql.org/docs/9.2/runtime-config-query.html#RUNTIME-CONFIG-QUERY-CONSTANTS.

In order to retrieve AQPs, a learner may select a subset of the configuration parameters (through a user-friendly visual interface) based on the physical operators that she intend to view in these plans. In this case, the corresponding parameters are set to "true" (*e.g.,* `SET enable_mergejoin = true`) in the query request. MOCHA supports two modes for generating alternative plans, namely, *single mode* and *multiple mode*. In the former mode, MOCHA sends a query request to PostgreSQL in which the *unselected* parameters are set to "false" to generate an AQP containing the operators corresponding to the selected parameters that are relevant to the processing of the query. In the latter mode, every selected parameter is either set to "true" or "false" to create all possible combinations of these parameters. MOCHA iterates through these combinations and sends corresponding query requests to PostgreSQL. It only maintains all *distinct* plans retrieved from these requests. To facilitate learning, it provides a learner-friendly GUI to detect and visualize various structural and cost differences between a selected AQP and the QEP.

MOCHA also generates a natural language-based explanation that goes beyond the conventional least-cost-based explanation to connect established knowledge related to usage scenarios of different physical operators that a learner has learnt from textbooks with the operators in a QEP. The current version manually extracts usage scenarios of different physical operators from the relevant literature. This is feasible since there is a small number of physical operators in PostgreSQL. Then a set of documents containing these usage scenarios is indexed using an inverted index where each document is associated with a single physical operator. For a given QEP, it identifies relevant operators and retrieves associated predicates and join conditions, if any. The text explanation is then generated for an operator by utilizing a rule-based template, the inverted index to retrieve corresponding usage scenario, and database statistics information (*e.g.,* selectivity). The generated explanation is visually displayed on the visual interface of MOCHA. For example, an explanation could be *"the* QEP *uses index scan on the `lineitem` table as it is faster due to the high selectivity of the predicate (*i.e., *`l_orderkey = orders.o_orderkey`)".*

In the future, we intend to generalize MOCHA to accommodate major RDBMS, support visualization of the impact of join ordering, and automate the manual extraction of usage information of various physical operators in these RDBMS. More importantly, we wish to deploy MOCHA in our learning environment and investigate its impact on students taking the database systems course.

4.3 Usage and Impact of NEURON

NEURON and LANTERN are currently deployed in database systems courses in NTU and Xidian University. We now briefly describe our initial efforts to measure NEURON's impact on the learning of QEP. To this end, we introduced it to students taking the undergraduate database systems course (*CZ4031*) in NTU in the August semester of 2021. 166 students were enrolled in this course. These students are pursuing a variety of degrees such as computer science, computer engineering, data science and analytics, and business and computing. In particular, the topic of query processing and optimization was covered in 4 weeks (September-October) over eight 1-hour lectures. On October 28th, the students took a test on the topic of query processing and optimization. The following message was sent to the students on 14th October, 2021: *"If you wish to understand query execution plans (*QEP*) generated by PostgreSQL for different* SQL *queries, you may use the software called* NEURON *at `https://neuron.scse.ntu.edu.sg/`.* NEURON *translates the* QEP *of a query to natural language description."* We did not nudge students any further on using NEURON or give them any hints on whether questions related to natural language descriptions of query plans will appear in the test. Hence, students were not aware of any assessment-related rewards if they used the tool. The goal here is to observe whether learners will use it without any nudging or grades-related rewards and whether they will benefit from it.

We observe the usage of NEURON from 14th October to 2nd December by analyzing the log file. Note that 12th November was the last date of the course culminating with the submission of a course project. Since a user needs to access NEURON by logging using an email address, we were able to match majority of the email addresses that accessed it with those registered for the course. There were 69 distinct learners (41.5%) using NEURON during this period. Figure 4(b) reports the number of queries posed by learners over time. In total, 888

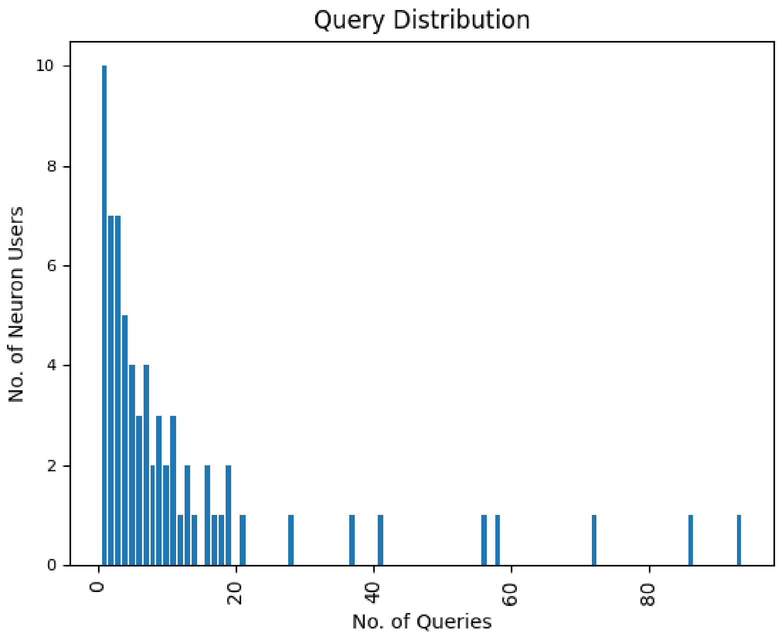


Figure 5: No. of queries versus no. of users.

valid SQL queries (484 distinct queries) were executed on NEURON. The distribution of the number of queries versus the number of distinct learners who posed that number of queries is shown in Figure 5. Observe that more than 85% of them posed more than one query and the maximum number of queries posed by a single user is 93! Prior to October 28th (test date), the number of distinct users is 48 and the number of queries posed is 409 (211 distinct queries). Hence, the usage of NEURON continued even after the test as learners may be using it to understand QEPs for their project work. Some students used it even after the official end date of the course probably demonstrating intrinsic motivation to explore QEPs.

To investigate whether NEURON may benefit learners to understand QEP, we use the test as a proxy. In the test, a question was specifically set to this end. The students were asked to explain in natural language the visual format of a QEP (in PostgreSQL) for an SQL query on IMDb database involving two joins, scans, and sort operations. The question carried 10 marks. Observe that it matches very closely to NEURON's goal. This enables us to evaluate more accurately possible impact of NEURON on the test performance.

In order to avoid any bias, a teaching assistant (TA) who is not involved with NEURON graded the answers to this question. The TA was given the solution and was allowed to set the marking scheme for the question. The number of students who took the test is 162. The average score of students who used (resp. not used) NEURON prior to test is 8.43 (resp. 7.07). The maximum, minimum, and median scores of these two groups are (10, 6.5, 8) and (10, 0, 7.5), respectively. Among the non-NEURON users, the percentage of students with scores lower than the minimum score in the NEURON user group (*i.e.,* 6.5) is 21.31%. Furthermore, the percentage of NEURON users (resp. non-NEURON users) with scores higher than the average score of 8 is 47.5% (resp. 35.25%). Some of the common errors made by students are (a) not describing in natural language; (b) incorrect sequence of steps; (c) not including filter conditions in the scan operations; and (d) unclear specifications of operators and intermediate results. Observe that these errors could have been mitigated with the usage of NEURON.

In summary, while we cannot infer causal link from the initial results, it is possible NEURON may improve learning of query execution strategies. We are still in the early stages of understanding the impact of this tool on learning. We intend to use NEURON and LANTERN in future semesters to gather sufficient longitudinal data for a more detailed investigation of technology-enabled learning and their impact on data-driven education.

5 Conclusions

Impact of digital technologies on learning has consistently shown positive benefits when they are aligned. With the advent of data science and lifelong learning, there has been growing interest in the database course from adult learners with diverse background. This necessitates us to revisit the traditional way we teach this course by supplementing it with technological support to improve learning. This paper contributes a vision of technology-enabled learning of the topic of relational query processing in a database systems course. Specifically, our vision attempts to carve out a substantially new research topic that is at the intersection of learning sciences and data management to improve learning of relational query processing. To the best of our knowledge, this vision has not been systematically investigated before, prior to our recent publications.

Measures of success. Successful realisation of this vision will improve learning and understanding of the complex topic of relational query processing. But several non-trivial and novel research challenges as articulated in the paper need to be overcome to realize it. Adoption of the potential solutions by real-world learners as a supplement to traditional modes of learning will be another measure of success.

Wider applicability. We focused on the topic of relational query processing since it is one of the most challenging topic in a database systems course. Nevertheless, it is easy to see that our vision of technology-enabled learning can be extended to other topics such as enabling technologies to facilitate learning of SQL queries.

References

[1] The IMDb database. https://relational.fit.cvut.cz/dataset/IMDb.

[2] Advanced query tuning concepts. https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms191426(v=sql.105)?redirectedfrom=MSDN, 2012.

[3] LANTERN software. https://howardlee.cn/lantern/.

[4] MOCHA software. https://howardlee.cn/mocha/.

[5] NEURON software. https://howardlee.cn/#/.

[6] USESCO Insititute of Lifelong Learning. UNESCO Global Network of Learning Cities. Accessible at https://uil.unesco.org/lifelong-learning/learning-cities, 2019.

[7] E. M. Anderman, H. Patrick. Achievement goal theory, conceptualization of ability/intelligence, and classroom climate. *In Handbook of research on student engagement*, Springer: Boston, MA, 2012.

[8] A. Bhangdiya, B. Chandra, B. Kar, B. Radhakrishnan, K. V. M. Reddy, S. Shah, S. Sudarshan. The XDa-TA system for automated grading of SQL query assignments. *In ICDE*, 2015.

[9] E. L. Bjork, R. Bjork. Making things hard on yourself, but in a good way. *Psychology in the Real World*, 59-68, 2011.

[10] R. A. Bjork, J. Dunlosky, N. Kornell. Self-regulated learning: Beliefs, techniques, and illusions. *Annual review of psychology*, 64, 417–444, 2013.

[11] R. Borovica-Gajic, S. Idreos, A. Ailamaki, M. Zukowski, C. Fraser. Smooth scan: robust access path selection without cardinality estimation. *The VLDB Journal*, 27(4):521-545, 2018.

[12] 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.

[13] S. Chaudhuri. An Overview of Query Optimization in Relational Systems. *In PODS*, 1998.

[14] P. Chen, H. Li, S. S. Bhowmick, S. R. Joty, W. Wang. LANTERN: Boredom-conscious Natural Language Description Generation of Query Execution Plans for Database Education. *In SIGMOD*, 2022.

[15] J. Danaparamita, W. Gatterbauer. QueryViz: Helping Users Understand SQL Queries and Their Patterns. *In EDBT*, 2011.

[16] E. L. Deci, R. J. Vallerand, L. G. Pelletier, R. M. Ryan. Motivation and education: The self-determination perspective. Educational psychologist, 26(3-4), 325-346, 1991.

[17] H. Garcia-Molina, J. D. Ullman, J. Widom. Database Systems: The Complete Book. *Prentice-Hall*, 2002.

[18] M. Gawade, M. L. Kersten. Stethoscope: A platform for interactive visual analysis of query execution plans . *PVLDB*, 5(12), 2012.

[19] R. Gross. Psychology: The Science of Mind and Behaviour. *Hachette UK*, ISBN 978-1-4441-6436-7.

[20] J. Gao, M. Galley, L. Li. Neural Approaches to Conversational AI. *In ACL*, 2018.

[21] J. R. Haritsa. The Picasso Database Query Optimizer Visualizer. *In PVLDB*, 3(2), 2010.

[22] 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.

[23] 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.

[24] S. Higgins, Z. M. Xiao, M. Katsipatak. The Impact of Digital Technology on Learning: A Summary for the Education. *Education Endowment Foundation*, 2012.

[25] Y. Hu, Z. Miao, Z. Leong, H. Lim, Z. Zheng, S. Roy, K. Stephens-Martinez, J. Yang. I-Rex: An Interactive Relational Query Debugger for SQL. *In ACM Technical Symposium on Computer Science Education (SIGCSE)*, 2022.

[26] Z. Ives, J. Gehrke, J. Giceva, A. Kumar, R. Pottinger. VLDB Panel Summary: "The Future of Data(base) Education: Is the Cow Book Dead?". *SIGMOD Rec.*, 50(3), 2021.

[27] A. E. Kazdin. Motivation: an overview. *Encyclopedia of Psychology*, American Psychological Association, ISBN 978-1-55798-187-5, 2000.

[28] H. Kim, B.-H. So, W.-S. Han, H. Lee. Natural Language to SQL: Where Are We Today? *PVLDB*, 13(10), 2020.

[29] A. Kokkalis, P. Vagenas, A. Zervakis, A. Simitsis, G. Koutrika, Y. E. Ioannidis. Logos: A System for Translating Queries into Narratives. *In SIGMOD*, 2012.

[30] A. Leventidis, J. Zhang, C. Dunne, W. Gatterbauer, H. V. Jagadish, M. Riedewald. QueryVis: Logic-based Diagrams help Users Understand Complicated SQL Queries Faster. *In SIGMOD*, 2020.

[31] S. Liu, S. S. Bhowmick, W. Zhang, S. Wang, W. Huang, S. Joty. NEURON: Query Optimization Meets Natural Language Processing For Augmenting Database Education. *In SIGMOD*, 2019.

[32] Z. Miao, S. Roy, J. Yang. Explaining Wrong Queries Using Small Examples. *In SIGMOD*, 2019.

[33] D. Miedema, G. Fletcher. SQLVis: Visual Query Representations for Supporting SQL Learners. *In VL/HCC*, 2021.

[34] D. Miedema, E. Aivaloglou, G. Fletcher. Identifying SQL Misconceptions of Novices: Findings from a Think-Aloud Study. *In ICER*, 2021.

[35] J. Nakamura, M. Csikszentmihalyi. Flow theory and research. *Oxford Handbook of Positive Psychology*, Oxford University Press, 2009.

[36] R. Ramakrishna, J. Gehrke. Database Management Systems. *McGraw-Hill, Inc.*, USA, 2020.

[37] R. M. Ryan, E. L. Deci. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary educational psychology*, 25(1), 54-67, 2000.

[38] D. W. Schumann, R. E. Petty, D. S. Clemons. Predicting the Effectiveness of Different Strategies of Advertising Variation: A Test of the Repetition-Variation Hypotheses. *Journal of Consumer Research*, 17, 2: 192, 1990.

[39] N. C. Soderstrom, R. A. Bjork. Learning versus performance: An integrative review. *Perspectives on Psychological Science*, 10(2), 176-199, 2015.

[40] J. Tan, D. Yeo, R. Neoh, H.-E. Chua, S. S. Bhowmick. MOCHA: A Tool for Visualizing Impact of Operator Choices in Query Execution Plans for Database Education. *PVLDB*, 15(12), 2022.

[41] K. Taylor, D. Rohrer. The effects of interleaved practice. *Applied Cognitive Psychology*, 24(6), 2010.

[42] W. Wang, S. S. Bhowmick, H. Li, S. Joty, S. Liu, P. Chen. Towards Enhancing Database Education: Natural Language Generation Meets Query Execution Plans. *In SIGMOD*, 2021.

[43] A. Wigfield, J. S. Eccles. Expectancy-value theory of achievement motivation. *Contemporary educational psychology*, 25(1), 68-81, 2000.

[44] C. Zhu, C. Ji, Y. Zhang, X. Liu, A. Jatowt, S. S. Bhowmick, C. Sun, T. Zhao. Towards Automatic Support for Leading Court Debates: A Novel Task Proposal & Effective Approach of Judicial Question Generation. *To Appear in Neural Computing and Applications (NCAA)*, Springer, 2022.