ARENA: Alternative Relational Query Plan Exploration for Database Education

Hu Wang
Xidian University
China
wh2mail@163.com

Hui Li
Xidian University
China
hli@xidian.edu.cn

Sourav S Bhowmick
Nanyang Technological University
Singapore
assourav@ntu.edu.sg

Baochao Xu
Xidian University
China
bcxu_x@stu.xidian.edu.cn

ABSTRACT

A key learning goal of learners taking a database systems course is to understand how SQL queries are processed in an RDBMS in practice. To this end, comprehension of different *alternative query plans* (AQPs) that may be considered during the selection of the query execution plan (QEP) of a query is paramount. In this demonstration, we present a novel and *generic* system called ARENA that facilitates exploration of *informative* alternative query plans of a given SQL query to aid the comprehension of QEP selection. Under the hood, ARENA addresses a novel problem called *informative plan selection problem* (TIPS) which aims to discover alternative plans from the underlying plan space so that the *plan informativeness* is maximized. We demonstrate various innovative features of ARENA emphasizing the important role it can play in supplementing database education.

CCS CONCEPTS

• **Information systems** → **Database query processing**.

KEYWORDS

database education, query execution plan, alternative query plan

ACM Reference Format:
Hu Wang, Hui Li, Sourav S Bhowmick, and Baochao Xu. 2023. ARENA: Alternative Relational Query Plan Exploration for Database Education. In *Companion of the 2023 International Conference on Management of Data (SIGMOD-Companion '23), June 18–23, 2023, Seattle, WA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3555041.3589713

1 INTRODUCTION

Database systems courses in major institutions around the world supplement traditional style of learning (*e.g.,* lecture, textbook) with the usage of off-the-shelf RDBMS (*e.g., PostgreSQL*) to infuse knowledge about database techniques used in *practice*. Unfortunately, these RDBMS are not designed for pedagogical support [1]. Although they enable hands-on learning opportunities to build database applications and pose a wide variety of SQL queries over them, very limited learner-friendly support is provided beyond them. This has led to increasing research activities to build tools and techniques to supplement traditional modes of learning in database courses [1, 2, 4, 5, 9].

A key learning goal in any database systems course is to understand how SQL queries are processed in practice. A relational query engine produces a *query execution plan* (QEP), which represents an execution strategy of an SQL query. Major database textbooks introduce the *general* (*i.e.,* not tied to any specific RDBMS software) theories and principles behind the generation of a QEP using natural language-based narratives. In particular, these textbooks typically elaborate on the selection process of the QEP of an SQL query from a set of *alternative query plans* (AQPs) by comparing their estimated costs. Scant attention, however, has been paid to explore technologies that can supplement the learning of this selection process. Note that major RDBMS typically only expose a QEP to an end user in a user-friendly manner. Consider the following motivating scenario.

Example 1.1. Georgia is an undergraduate student who is currently enrolled in a database systems course. She issued the following SQL query in *PostgreSQL* on the IMDB dataset (https://relational.fit.cvut.cz/dataset/IMDb).

```
SELECT t.title AS movie_title
FROM keyword AS k,  movie_info AS mi,
     movie_keyword AS mk, title AS t
WHERE t.production_year > 2005
      AND t.id = mi.movie_id
      AND t.id = mk.movie_id
      AND mk.movie_id = mi.movie_id
      AND k.id = mk.keyword_id;
```

The corresponding QEP is depicted in Figure 1(a). After perusing the content of the QEP, she wonders how some of the different AQPs look like? Specifically, are there alternative plans with similar estimated cost as the QEP? How are plans with significantly higher cost look like? Examples of such AQPs are shown in Figures 1(b)-(d). ∎

Observe that it is challenging for a learner like Georgia to explore the AQPs using an off-the-shelf RDBMS. Although an RDBMS (*e.g., PostgreSQL*) may allow a learner to manually pose SQL queries with various constraints on *configuration parameters* to view the corresponding QEPs containing specific physical operators (*e.g., SET enable_hashjoin = true*), such strategy demands not only familiarity of the configuration parameters but also he/she must have a clear idea of plans of interest. Often this is impractical to assume for learners who are taking a database systems course for the first time.

SIGMOD-Companion '23, June 18–23, 2023, Seattle, WA, USA

ACM ISBN 978-1-4503-9507-6/23/06.
https://doi.org/10.1145/3555041.3589713

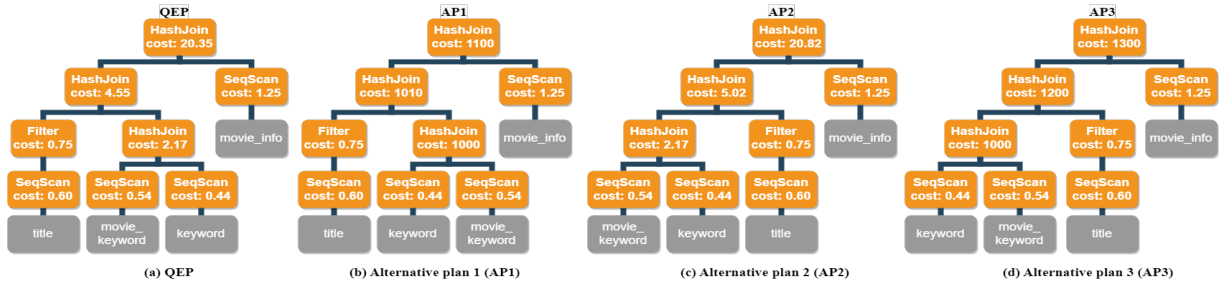

Figure 1: Example of QEP and alternative query plans (AQPs).

Clearly, a user-friendly tool that can facilitate retrieval and exploration of *representative* AQPs associated with a given query can greatly aid in answering Georgia's questions. In this demonstration, we present a novel system called ARENA (AlteRnative quEry plaN ExplorAtion) to address the *informative plan selection* (TIPS) problem. ARENA facilitates learners to retrieve, view, and explore *informative* alternative query plans effortlessly. Given an SQL query and a parameter $k > 0$, instead of showing only the QEP, it selects k *informative* AQPs for display and exploration. Under the hood, it addresses two variants of TIPS, namely the *batch* TIPS (B-TIPS) and *incremental* TIPS (I-TIPS) problems, to cater for scenarios where k is specified or unspecified by a learner, respectively.

2 SYSTEM OVERVIEW

The architecture of ARENA is depicted in Figure 2. It consists of the following key components.

2.1 ARENA GUI

ARENA exposes a browser-based visual interface (GUI) that enables a user to retrieve and view information related to the QEP and *informative* alternative query plans of her input query in a user-friendly manner. Figure 3 shows a screenshot of the ARENA GUI. It consists of several components organized into three columns. The left-most column contains three panels: database schema visualizer (C1), configuration panel for various parameters (C2), and controller of the display mode (C3). In the middle column, a learner can type in his/her SQL query (C4) or select a predefined query from a drop-down list (C5). Upon clicking on Execute, the C6 panel displays the QEP and the AQPs, as well as associated information (detailed later) that highlights the differences between the former and the latter. Clicking on a specific plan in C6 leads to its visualization in the right column (C7). In addition, if Compare Plan in C3 is enabled, a pop-up window showing the differences between the QEP and the selected AQP are displayed (Figure 4). Specifically, the left panel lists down the differences between both plans (w.r.t. operators, estimated cost, join order). Clicking on any item in this list will trigger the right panel to highlight the corresponding regions of the plans.

2.2 Candidate Plan Set Retriever

In order to retrieve *informative* AQPs for a given SQL query, we need to first obtain a large number of candidate AQPs. To this end, we adopt ORCA [6], which is a modular top-down query optimizer based on the cascades optimization framework. We adopt it for the following reasons. First, it is a stand-alone optimizer and is independent of the RDBMS a learner interacts with. Consequently, it facilitates

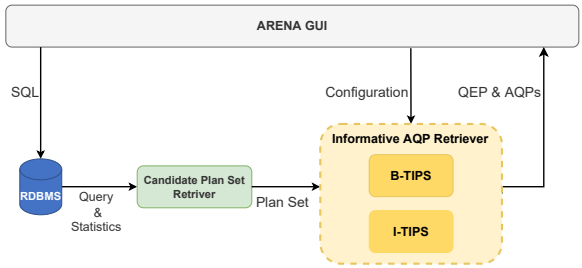

Figure 2: Architecture of ARENA.

portability of ARENA across different RDBMS. Second, it interacts with the RDBMS through a standard XML interface such that the effort to support a new RDBMS is minimized. The only task one needs to undertake is to rewrite the parser that transforms a new format of query plan (*e.g.,* XML in SQL Server, JSON in PostgreSQL) into the standard XML interface of ORCA. Consequently, this makes ARENA compatible with many RDBMS as a parser can be easily written without touching the internals of the target RDBMS. Specifically, we have implemented this interface on top of *PostgreSQL* and *Greenplum*.

2.3 Informative AQP Retriever

This module is the core engine of ARENA. Note that the candidate plan space can be prohibitively large. Hence, it is ineffective to expose all plans. Instead, it is paramount to select those that are *informative*. However, how do we quantify *informativeness* in order to select plans? For instance, in Figures 1(b)-(d), which plans should be revealed to Georgia? Here we informally introduce the notion of *plan informativeness* to address this issue. The reader may refer to [8] for its formal treatment.

Consider *AP1* or *AP3*. By viewing them, Georgia will learn that the estimated cost is sensitive to the join order and for some plans it can be significantly higher than the cost of the QEP. On the other hand, *AP2* reveals that changing the join order may not always lead to significantly different estimated cost from the QEP. In fact, *AP2* and the QEP have very similar cost. Observe that *AP1* and *AP3* convey highly similar information w.r.t. the QEP (*i.e.,* both plans have different join order and substantially higher cost). Hence, {*AP1*, *AP2*} or {*AP2*, *AP3*} are potentially most informative sets for Georgia. Observe that no matter what order these plans are presented to her, she will learn different information from them. In other words, any informativeness measure should consider the plans (including the QEP) that an individual has already viewed for his/her query so that highly similar plans are not exposed to them. It should also be cognizant of individuals' interests in this context. Furthermore, query optimizers search and filter based on cost. In contrast, as shown in

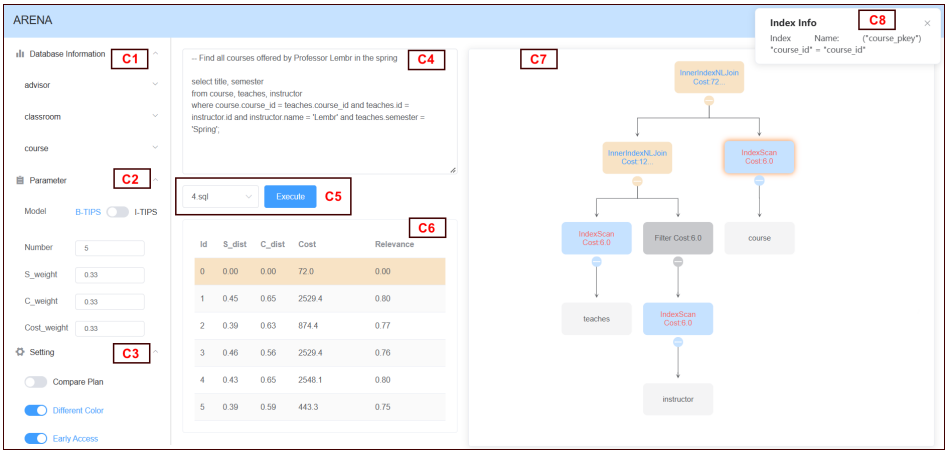

Figure 3: A screenshot of the ARENA GUI.

the above example, simply seeking plan(s) with the lowest cost is insufficient in this scenario. Specifically, the content, structure, and cost of a query plan w.r.t. the QEP and already-viewed plans play pivotal roles in determining informativeness of AQPs.

Intuitively, the notion of *plan informativeness* (informativeness for brevity) is captured using *distance* (computed by exploiting *differences* between plans w.r.t structure (*S_dist*), content (*C_dist*) and estimated cost (*Cost*)) and *relevance* of alternative plans w.r.t. the QEP or the set of alternative plans viewed by a user thus far (see C6 panel in Figure 3). Specifically, the former is used to avoid selecting similar plans with negligible marginal informativeness.

The goal of this module is to automatically select a small number of alternative plans that maximize the plan informativeness (*i.e.,* the TIPS problem). These plans are referred to as *informative alternative (query) plans*. Under the hood, it addresses two flavors of the TIPS problem, namely *batch* and *incremental*. The *batch TIPS* (B-TIPS) problem facilitates a user to view top-k informative plans other than the QEP whereas the *incremental TIPS* (I-TIPS) problem enables one to *iteratively* view an informative plan besides those that have been already shown to him/her. I-TIPS also enables users to provide feedback on the current plan and the subsequent plan is then selected by exploiting it. Formally, given a QEP π^* and a budget k, the B-TIPS problem aims to find top-k alternative plans Π_{OPT} such that the plan informativeness of the plans in $\Pi_{OPT} \cup \{\pi^*\}$ is maximized, *i.e.,* $\Pi_{OPT} = \text{argmax}_{\Pi} U_{\pi^*}(\Pi \cup \{\pi^*\})$ subject to $|\Pi| = k$ and $\Pi \subseteq \Pi^* \setminus \{\pi^*\}$ where Π^* denotes the space of all possible plans and $U_{\pi^*}(\cdot)$ denotes the plan informativeness of the plan set. On the other hand, I-TIPS selects an AQP iteratively and the selection process is influenced by the user feedback (*i.e.,* a rating on the usefulness of a plan), if any, on the current plans viewed thus far. Under the hood, the user feedback is modeled using a *feedback function* f. ARENA automatically updates the plan informativeness of other AQPs based on f so that the subsequent selection of AQP(s) is directed towards plans that are aligned with f. The reader may refer to [8] for further details on these two problems.

It is worth emphasizing the role of I-TIPS in database education environment. Observe that B-TIPS demands the value of k as input from learners. Furthermore, it returns the same results for the same query from different learners. Our engagement with learners reveal that they may not necessarily be confident to specify k always. One may prefer to *iteratively* view one plan-at-a-time and only cease exploration once he/she is satisfied with the understanding of the alternative plan choices for a specific query. Hence, k may not only be unknown *apriori* but also the selection of an AQP at each iteration depends on the plans viewed by them thus far. I-TIPS is designed to address this issue. Specifically, in this mode ARENA only selects one AQP-at-a-time that maximizes plan informativeness based on the AQPs that have been shown to a learner. One can annotate whether the current AQP is informative for learning and ARENA will automatically select the subsequent AQP based on the feedback.

it can be prohibitively expensive to scan all AQPs to select informative ones. How can we design efficient techniques to select informative plans? Importantly, this cannot be integrated into the enumeration step of the query optimizer as informative AQPs need to be selected based on the QEP an individual has seen. We note that B-TIPS is NP-hard, but a 2-approximation result can be obtained by the greedy solution proposed in [8]. Intuitively, it calculates the informativeness of each plan when selecting the i-th AQP and then select the one with the largest value. Since the plan informativeness decreases monotonically with the increase of $|\Pi|$, ARENA adopts a max-heap to filter plans with very small informativeness to avoid unnecessary computation. When the i-th AQP is selected, the heap records the informativeness of each plan when it is selected in i'-th ($i' \leq i$) iteration. If it is less than the current solution, this iteration can be terminated. Observe that the worst-case time complexity is $O(k\Pi^*)$. However, it is significantly more efficient than several baseline methods in practice [8].

3 RELATED SYSTEMS

Research on tools and techniques to supplement learning of relational query processing is still in its infancy [1]. NEURON [4] and LANTERN [9] generate a natural language description of a QEP to facilitate its understanding. MOCHA [7] is a tool for learner-friendly interaction and visualization of the impact of alternative physical operator choices on a selected QEP for a given SQL query. Given an SQL query and learner-specified *operator preferences* (*e.g.,* merge join, index scan), it automatically visualizes the impact of these choices on the selected QEP. Hence, MOCHA demands a learner to

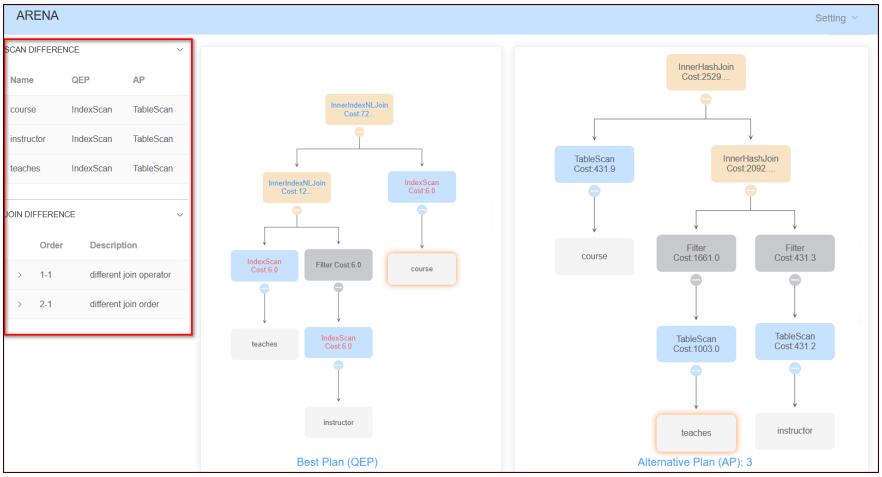

Figure 4: Comparison of QEP and an alternative query plan.

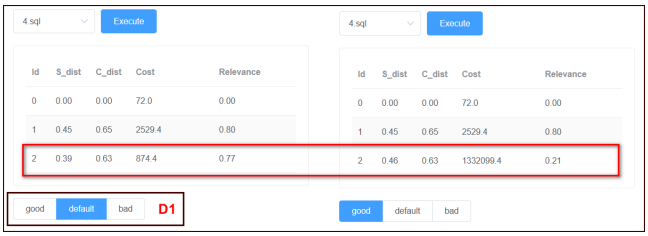

Figure 5: Results retrieved by I-TIPS.

have clear preferences of what physical operators they wish to explore w.r.t. a QEP. In contrast, ARENA is a generic tool that does not demand any operator preferences. Picasso [3] depicts various visual diagrams of different QEPs and their costs over the entire selectivity space. In summary, ARENA complements these efforts by facilitating exploration of informative alternative query plans.

4 DEMONSTRATION OVERVIEW

ARENA is implemented using *Python* and *PostgreSQL*. Our demonstration will make use of several publicly-available datasets such as IMDB. Users can pose their own ad-hoc queries on these datasets using ARENA. A short **video** to illustrate the main features of ARENA using example use cases is available at https://youtu.be/VDVU_co47YM. ARENA can be accessed via https://dbedu.xidian.edu.cn to supplement database education. Specifically, we shall demonstrate the followings.

Interactive exploration of k informative AQPs using B-TIPS. An audience can view the schema of the selected database in the C1 panel and specify or preselect an SQL query in the C4 panel through the GUI of ARENA (Figure 3). One can adjust various parameters in the C2 panel including the number of AQPs (*i.e.*, k) and observe and explore their impact on the informative AQP retrieval process by B-TIPS. The table in C6 shall display the QEP (with *id* 0) and k AQPs (with other ids). It also shows information related to the distance and relevance measures for each AQP. One can click on a specific plan to view it in C7. Audiences can view the differences between the QEP and a selected AQP conveniently by enabling the options in C3 (Figure 4). Specifically, the QEP and the AQP are juxtaposed and the differences can be viewed in the left panel (red rectangle). One can also interact with these differences by clicking on any item and ARENA shall highlight the corresponding fragments of the plans.

Interactive, personalized exploration with I-TIPS. Users can use I-TIPS similar to B-TIPS except that they can click on the Execute button as many times as they want to retrieve a new AQP iteratively. Furthermore, in contrast to B-TIPS, a radio button below the table in C6, (D1 in Figure 5) is exposed. Users can choose whether they are satisfied with the current AQP (default means no adjustment is required) and ARENA will reveal the next result based on the feedback. Figure 5 shows the differences when default and good are chosen as feedback (highlighted with a red rectangle).

ACKNOWLEDGEMENT

This work is partially supported by the National Natural Scientific Foundation of China (No. 61972309, 62272369) and China 111 Project (No. B23046).

REFERENCES

[1] S. S. Bhowmick, H. Li. Towards Technology-Enabled Learning of Relational Query Processing. *IEEE Data Engineering Bulletin*, 45(3), 2022.
[2] Wolfgang Gatterbauer, Cody Dunne, H. V. Jagadish, and Mirek Riedewald. 2022. Principles of Query Visualization. *IEEE Data Engineering Bulletin* 45(3) (2022).
[3] Jayant R. Haritsa. 2010. The Picasso Database Query Optimizer Visualizer. *Proc. VLDB Endow.* 3, 2 (2010), 1517–1520.
[4] S. Liu et al. NEURON: Query Optimization Meets Natural Language Processing For Augmenting Database Education. *In SIGMOD*, 2019.
[5] D. Miedema, G. Fletcher. SQLVis: Visual Query Representations for Supporting SQL Learners. *In VL/HCC*, 2021.
[6] M. A. Soliman et al. Orca: a modular query optimizer architecture for big data. *In SIGMOD*, 2014.
[7] Jess Tan, Desmond Yeoh, Rachael Neoh, Huey-Eng Chua, and Sourav S. Bhowmick. 2022. MOCHA: A Tool for Visualizing Impact of Operator Choices in Query Execution Plans for Database Education. *Proc. VLDB Endow.* 15, 12 (2022), 3602–3605.
[8] Hu Wang, Hui Li, and Sourav S Bhowmick. 2022. ARENA: Towards Informative Alternative Query Plan Selection for Database Education. *arXiv preprint arXiv:2210.13722* (2022).
[9] 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.