T. Grust et al. (Hrsg.): Datenbanksysteme für Business, Technologie und Web (BTW 2019),
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2019 503

RelaX: A Webbased Execution and Learning Tool for Relational Algebra

Johannes Kessler[1] Michael Tschuggnall[2] Günther Specht[3]

Abstract: The relational model and especially the relational algebra is the fundament of each relational database system and thus content of almost every database lecture. Even though there exist a few tools allowing to experiment with relational algebra, a common way to learn it is still by formulating queries on paper, without the option of checking them for syntax or even executing them. To fill this gap and to support students in their learning process, we propose RelaX, a webbased tool which is capable of executing arbitrary relational algebra statements on arbitrary datasets. By drawing interactive operator trees corresponding to the queries, it is also possible to compute the final result in a step-by-step manner. Finally, RelaX is also equipped to execute SQL queries and to automatically translate them to relational algebra.

Keywords: relational algebra, database systems, learning tool

1 Motivation

The relational model and the relational algebra have been proposed in 1970 by E. F. Codd [Co70] and have evolved to be the theoretical fundament of almost every modern relational database system. Through its procedural structure it is very well suited to transmit an understanding of the basic operations of the relational model, and is thus often utilized for (academic) teaching purposes as a solid base for a subsequent introduction to SQL. While there exist numerous possibilities to learn the latter using various database systems, query tools and corresponding datasets, relational algebra is still mostly taught purely theoretically due to the lack of proper execution tools. The two fundamental differences to SQL in terms of learning support can be summarized as follows: First, while the SQL standard defines a clear syntax with minor variations in current database systems, there is no official, standardized syntax for relational algebra. This leads to the problem that notations often differ significantly from one text book or author to another, making the verification of correctness a cumbersome procedure for students as well as for tutors. Second, SQL can be executed and tested directly on different database systems. On the contrary, tools that allow the execution of relational algebra statements are rare (e.g., *radb* [Ya18], *IRA* [Mu18] or *Relational* [To18]) and come with certain limitations. These include restrictions

[1] Department of Computer Science, Universität Innsbruck
[2] Department of Computer Science, Universität Innsbruck, michael.tschuggnall@uibk.ac.at
[3] Department of Computer Science, Universität Innsbruck, guenther.specht@uibk.ac.at

doi:10.18420/btw2019-32

to predefined datasets (*IRA*) or general usability flaws in terms of tool installation/execution (*radb*, *Relational*), editing queries (*IRA*) or creating, importing and sharing datasets. In addition, advanced but yet important operators like `GROUP BY` (γ) are not supported by any of those tools. Finally, the execution of relational algebra statements is usually done by transforming queries to SQL and utilizing a relational database system in the background (*IRA, radb, Relational*), making results difficult to understand.

2 RelaX

With *RelaX*[4] we created a webbased learning tool for relational algebra, which addresses the previously mentioned problems by allowing to execute arbitrary relational algebra statements on predefined or custom datasets, offering a rich set of operators.

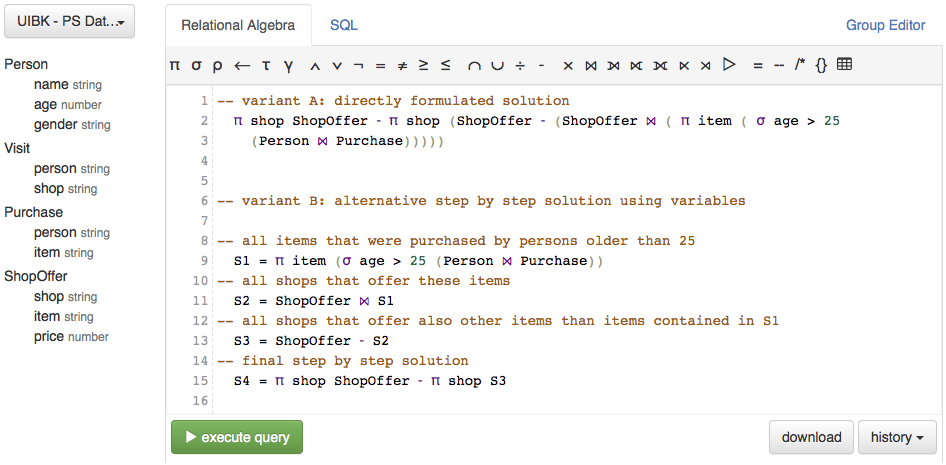

Fig. 1: The RelaX editor.

Syntax and Editor

Analogously to SQL the statements are thereby treated as *source code*, which allows comfortable editing with syntax highlighting, auto completion and the possibility of stating comments. For query formulation, it is possible to directly use the mathematical notation as well as relying on corresponding key words. For example, when using a selection, both the use of σ as well as `sigma` is possible. With the proposed editor shown in Figure 1 it is thus possible to develop step-by-step solutions, while at the same time addressing and avoiding syntax errors directly during editing. For example, variant B in Figure 1 constructs the final solution $S4$ using the three intermediate steps $S1 - S3$.

With respect to syntax we rely on the definitions given by Kemper and Eickler [KE15] as well as Garcia-Molina et al. [GMUW08]. During the process of creating *RelaX* with the aim

[4] RelaX stands for Relational Algebra eXecutor and is available at http://dbis-uibk.github.io/relax

of supporting every possible relational algebra operator/formula, we came across several ambiguities as well as missing information in literature. For example, even for a simple operator like the natural join there exist serveral definitions, which either merge attributes with same names, inherit the corresponding attribute from the left or right table or use both attributes for the result. Among several other similar problems, the lack of definitions of proper scoping and precedence of operators had to be solved. With *RelaX*, we resolved these issues by folllowing the semantic of SQL whenever possible.

Execution

For the execution, we laid the main focus on transparency and reproducability, i.e., such that users can reconstruct the final result in a step-by-step manner. Consequently, statements are not transformed to SQL and executed in an external database system to retrieve results, but rather interpreted and executed directly in relational algebra. For this, a statement is transformed to and visualized as an interactive operator tree. For example, the statement of variant B from Figure 1 results in the operator tree depicted in Figure 2. Thereby, users can click on every node of the tree to not only inspect the intermediate result up to this node, but also to verify the corresponding schema and view other information like which attributes have been used for a natural join.

As we are interested in providing a transparent tool with respect to understandability, statements are not optimized at all and rather executed as a beginner would except it. Therefore, all operations are implemented such that they preserve the order of tuples of the original relation. For joins, a nested loop join is implemented as it is the most intuitive variant. Finally, also grouping and aggregation is implemented in a comprehensible way.

3 Operators, Datasets and SQL support

RelaX supports all common unary operators like selection (σ), projection (π) or renaming (ρ, $\leftarrow$) and can handle several join variants (e.g., $\bowtie$, $\ltimes$, $\rtimes$) as well as set operations ($\cup$, $\cap$, $-$) and division ($\div$). Besides a grouping operator (γ), also the sorting operator τ as proposed by Garcia-Molina et al. [GMUW08] has been implemented. To allow users to construct a step-by-step solution, we introduced variables as can be seen in variant B of Figure 1. A comprehensive description on all supported operators, their usage as well as syntax in general is provided in the help section of the tool.

With respect to datasets – where statements can be executed on – we provide several predefined, commonly used datasets that can be used directly (e.g., the university schema from Kemper and Eickler [KE15]). Moreover, it is possible to include custom datasets using Github-Gists[5], SQL dumps or by editing them directly in an embedded spreadsheet editor.

[5] https://gist.github.com/

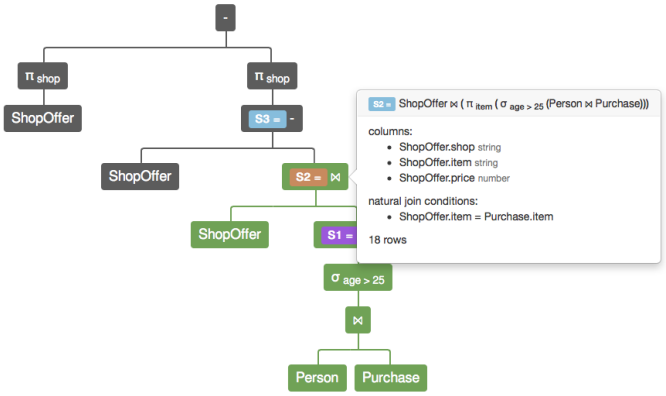

Fig. 2: Automatically computed operator tree, optionally showing intermediate results.

Finally, *RelaX* also supports the formulation of SQL statements[6] that are automatically transformed to relational algebra, which then can be executed and inspected using the operator tree. With this functionality, students can more easily learn SQL if the concepts of relational algebra are understood, but also vice versa.

4 Conclusion

With *RelaX* we created a webbased learning tool for relational algebra which allows the execution of arbitrary statements on predefined or custom datasets. Conceptualized for the use in teaching we integrated several functions and means such as a comprehensive editor and operator tree visualization, all of which should help students understand relational algebra better.

References

[Co70] Codd, E. F.: A Relational Model of Data for Large Shared Data Banks. Commun. ACM, 13(6):377–387, 1970.

[GMUW08] Garcia-Molina, Hector; Ullman, Jeffrey D.; Widom, Jennifer: Database Systems - The Complete Book. Pearson Prentice Hall, New Jersey, 2nd edition edition, 2008.

[KE15] Kemper, Alfons; Eickler, André: Datenbanksysteme - Eine Einführung, 10. Auflage. De Gruyter Oldenbourg, 2015.

[Mu18] Muehe, Henrik: IRA - Interaktive Relationale Algebra. http://db.in.tum.de/people/sites/muehe/ira/, visited September 2018.

[To18] Tomaselli, Salvo: Educational Tool for Relational Algebra. https://github.com/ltworf/relational/, visited September 2018.

[Ya18] Yang, Jun: IRA - Interaktive Relationale Algebra. https://users.cs.duke.edu/~junyang/radb/, visited September 2018.

[6] with the exception of correlated subqueries and recursive statements