Provenance-Based Visual Data Exploration with EVLIN

Houssem Ben Lahmar
University of Stuttgart
houssem.ben-lahmar@ipvs.uni-stuttgart.de

Melanie Herschel
University of Stuttgart
melanie.herschel@ipvs.uni-stuttgart.de

Michael Blumenschein
University of Konstanz
michael.blumenschein@uni-konstanz.de

Daniel A. Keim
University of Konstanz
keim@uni-konstanz.de

ABSTRACT

Tools for *visual data exploration* allow users to visually browse through and analyze datasets to possibly reveal interesting information hidden in the data that users are a priori unaware of. Such tools rely on both *query recommendations* to select data to be visualized and *visualization recommendations* for these data to best support users in their visual data exploration process.

EVLIN (exploring visually with lineage) is a system that assists users in visually exploring relational data stored in a data warehouse. EVLIN implements novel techniques for recommending both queries and their result visualization in an integrated and interactive way [3]. Recommendations rely on *provenance* (aka lineage) that describes the production process of *displayed data*.

The demonstration of EVLIN includes an introduction to its features and functionality through sample exploration sessions. Conference attendees will then have the opportunity to gain hands-on experience of provenance-based visual data exploration by performing their own exploration sessions. These sessions will explore real-world data from several domains. While exploration sessions use a Web-based visual interface, the demonstration also features a researcher console, where attendees may have a look behind the scenes to get a more in-depth understanding of the underlying recommendation algorithms.

1 VISUAL DATA EXPLORATION

Data exploration [8] helps users in finding interesting information in data sets when they do not know beforehand what useful information hides in their data. It thus supports humans in understanding and interpreting data in an investigative way. As manual data exploration is tedious, time-consuming, and it is easy to overlook interesting information, there is a need for tools supporting data exploration. These tools typically rely on different kinds of recommendations. Essentially, *query recommendation* guides users in their investigation of a data set D by suggesting queries as next exploration steps, given an initial query Q. Opposed to that, visualization recommendation commonly determines suited visualizations given a data set as input.

State-of-the-art. Most data and visualization recommendation techniques work independently from one another, meaning that the result of query recommendation, i.e., the data set $Q'(D)$ returned by executing a recommended query Q' over D, has no impact on the visualization recommendation process, and vice versa. This becomes apparent in Tab. 1 that summarizes works most closely related to ours. For each approach, it describes (i) the expressiveness of input queries (e.g., select-project-join (SPJ) queries, select-project-aggregate (SPA) queries, or cube queries corresponding to SPJA queries), (ii) the type of recommended output queries, (iii) the information used to compute query recommendations, (iv) the type of recommended visualization, and (v) the information used to compute visualization recommendations. This summary clearly shows that there is a gap between query recommendation systems such as YmalDB [5], SeeDB [12], or REACT [10] for expressive data exploration on the one hand, and visualization recommendation systems such as Voyager [14, 15] and Tableau's Show Me [9] on the other hand. Indeed, whereas the former may support the full range of typical OLAP queries, they do not offer any visualization recommendation. Typically, there is a one to one mapping between the result relation and a displayed table [5, 10] or bar chart [12]. Opposed to that, visualization recommendation solutions typically offer no or very limited support for query recommendation.

System	input query	recom. query	input for query recom.	recom. vis.	input for vis. recom.
YmalDB [5]	SPJ	SPJ	D, $Q(D)$	-	-
SeeDB [12]	cube	sub-cube	D, $Q(D)$	-	-
REACT [10]	cube	OLAP queries	previous exploration sessions' query history	-	-
Show Me [9]	SPA	-	-	diverse	data types
Voyager [14, 15]	SPA	Changed SELECT-clause	D, Q, metadata of D (schema, statistics)	diverse	data types, visual encoding channels
EVLIN	**cube**	**OLAP queries**	**D, Q, data & evolution provenance**	**diverse**	**$Q(D)$, evolution provenance**

Table 1: Summary of data exploration systems leveraging query recommendation or visualization recommendation

Contribution. EVLIN bridges the gap between query and visualization recommendation, seamlessly integrating both query and visualization recommendation for a streamlined, interactive user-experience. It relies on a novel recommendation strategy that leverages provenance to recommend queries and interactive visualizations in relation to each other [3]. The underlying techniques as well as the implementation focus on visually exploring relational data stored in a data warehouse. That is, we assume an input data set to conform to a snowflake schema. This demonstration focuses on the usability and interactivity of EVLIN in letting users explore these data. Through various real-world scenarios, we showcase that provenance-based recommendations for visual data exploration allow to effectively reveal interesting information. An example exploration session showing the functionality of EVLIN is available as an online video in [1].

Structure. Sec. 2 highlights the innovative aspects of EVLIN. The audience experience is first addressed in Sec. 3 where we discuss an exploration session in detail. On-site details such as the intended audience, sample scenarios, and a summary of the audience experience beyond the sample exploration session of Sec. 3 are then covered in Sec. 4.

2 EVLIN CONTRIBUTIONS

This section briefly summarizes the scientific contributions of EVLIN. We refer interested readers to [3] for more technical details, which we leave out here due to space constraints.

Leveraging data and evolution provenance. EVLIN captures two types of provenance: *data provenance* (more specifically why-provenance) and *evolution provenance* [7]. Data provenance records which data in the database D was used to derive the query result $Q(D)$ of a given query Q. In our context, evolution provenance [3] captures the explored dataset history as well as user interactions and visual encoding parameters of corresponding visualizations, thus tracking how a current visualization was derived. Thus, our model of evolution provenance extends the query history model used for query recommendation in REACT [10].

Provenance-based recommendation. We have developed a novel query recommendation algorithm that takes into account data provenance of data that has been explored and interacted with during an exploration session via the visual front-end. For an input SPJA SQL query, the computed recommendations follow typical data warehouse operations such as drill-down, roll-up, or slice. To identify adequate visualizations for the results of recommended queries, we have further developed a recommendation strategy that takes into account both interactions and visualizations captured as evolution provenance. While our query recommendation is similar in spirit to REACT [10], which records query histories from exploration sessions, our system leverages a richer provenance model and recommends not only queries but also their result visualizations.

Recommendation data space coverage and conciseness. To the best of our knowledge, EVLIN is the first system that recommends visualizations of query results for all queries typical in data warehouse navigation. Indeed, recommendations include roll-up, slice/dice, and drill-down queries, including drill-down queries that navigate to dimensions not considered by previous queries. In addition, data can be clustered by different characteristics or measures to zoom-in to more detailed distributions. To reduce the number of recommendations that are ultimately presented to the user (both for efficiency and usability reasons) while avoiding the loss of potentially relevant recommendations, we have explored how to leverage integrity constraints such as functional dependencies to prune redundant recommendations [3].

Visualization of quantified recommendation quality. Given the high diversity and possibly large number of recommended queries (and associated visualizations) produced by EVLIN, we propose to support users to navigate through the exploration space by quantifying the "interestingness" of recommended next exploration steps. The computed scores are then visualized in an interactive impact matrix, pointing users to potentially interesting data visualizations for different data warehouse operations.

3 SYSTEM FUNCTIONALITY

The above contributions are implemented as part of the EVLIN Web application that supports the exploration of multidimensional relational data stored in a data warehouse D with schema $schema(D)$, given an initial SQL query Q. EVLIN users are expected to have initial basic knowledge about the schema and dimensions of the data warehouse. The query Q, as well as all queries subsequently recommended take the form

SELECT $f(m), A$ FROM $rel(Q)$ WHERE $cond$ GROUP BY A

where m is a measure in the fact table, $A = \{a_1, \ldots, a_n\}$ is a set of attributes, f is an aggregation function, $rel(Q)$ refers to one or more relations in $schema(D)$, and $cond$ is a conjunction of predicates.

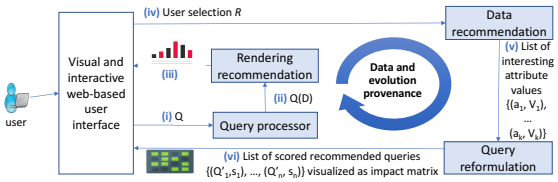


Figure 1: EVLIN system overview

Fig. 1 depicts the general processing EVLIN implements: (i) A user triggers an exploration session by issuing Q. (ii) The *query processor* executes the query and returns its result, denoted $Q(D)$. (iii) $Q(D)$ is then input to *rendering recommendation* that determines an adequate visualization to render the query result. (iv) The user can interact with the data via the graphical user interface, selecting a data-subset of interest, denoted R. (v) Based on this interaction, *data recommendation* identifies which attributes and values within their domain may be of interest to the user at the next step of his data exploration session. (vi) For each data recommendation, *query reformulation* determines variations Q' of the query Q that correspond to different exploration queries over a data cube in a data warehouse (slice, drill-down, roll-up, etc.). The interestingness of these queries is quantified based on the data underlying Q and Q' and the consistency of possible visualizations for $Q'(D)$ wrt evolution provenance. The result of this step is a set of recommended queries and respective visualizations for each recommended attribute of step (v) with associated interestingness scores. This information is visualized as an impact matrix to the user. The user then chooses one particular query to explore next. Upon selection of this query Q', the next iteration of the exploration process starts.

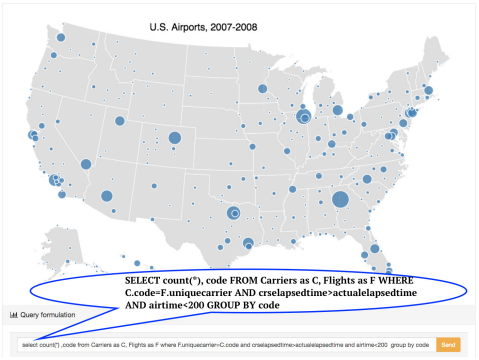
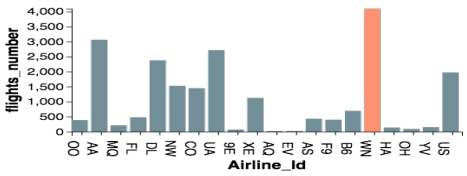


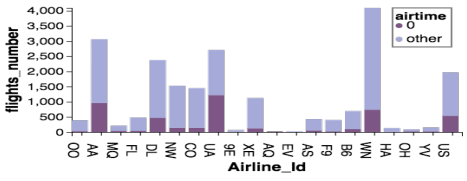
Figure 2: Initial input interface and sample query Q

In what follows, we detail functionalities, interfaces, and interactions that the audience will experience, focusing on the rendering recommendation, data recommendation, and query reformulation components. Due to space constraints, an in-depth discussion of the underlying algorithms is out of the scope of this paper, and we refer interested readers to [3] for details.

An initial user-specified query Q is input via a graphical user interface, as illustrated in Fig. 2. While our current implementation requires full-text SQL (loaded from a file or typed in a text field), a more user-friendly interaction similar to Voyager is planned. Our sample scenario considers a database D of domestic US flights



(a) Example of recommended visualization for $Q(D)$



(b) Visualization of recommended query result $Q'(D)$

Figure 3: Sample visualizations rendered using EVLIN

and Q that determines the number of short flights per airline which arrive ahead of schedule (a possible indicator for airline quality). Using the same scenario, a more extensive exploration session than the one described in the following is showcased in an online video in [1]. It includes for instance visualizations other than bar charts and further data warehouse operations.

3.1 Rendering recommendation

The rendering recommendation component takes as input a query result $Q_i(D)$ and evolution provenance P_e to determine a suited visualization of $Q_i(D)$. As described in [3], evolution provenance encompasses information about (i) past queries, (ii) the set of visualization resources used to render results of these queries, and (iii) information of past user interactions such as the selected data regions or specific sub-results. Essentially, this information are collected for each exploration step of an exploration session, where steps are delimited by transitioning from visualizing a query Q to visualizing a query Q'. This results in evolution provenance being a directed graph where nodes represent individual exploration steps and edges represent transitions from one step to the next. For a given $Q_i(D)$, its corresponding evolution provenance P_e includes all meta-data associated to graph nodes on the path from the initial query Q to the node representing the query Q_i.

Using this input, rendering recommendation aims at maximizing the visual similarity of a recommended visualization with those seen and interacted with previously for similar queries (intuitively, such that users easily recognize the same information as seen previously and thus understand the meaning of visualizations faster). This first requires determining similar queries among those in P_e. We quantify this similarity using a token-based similarity function between Q_i and a query $Q_p \in P_e$, weighted by the inverse of the shortest distance from Q_i to the previously seen Q_p on the path defining P_e. Among all queries of P_e with a similarity to Q_i above a predefined threshold θ_V, we now take the most frequent encoding parameters used to visualize information that is also selected by Q_i. These define a preliminary skeleton for the recommended visualization of $Q_i(D)$. In case no prior visualizations can be used, we resort to the effectiveness metrics adopted by Voyager [14, 15] to construct the visualization. Finally, the visualization skeleton is possibly updated based on the set of constraints defined in [11], that reduce conflict between visualizations.

As an example, Fig. 3a displays the recommended bar-chart visualization for our sample query Q that counts short flights arriving ahead of schedule for each airline. For this initial rendering, $P_e = \emptyset$, thus, the recommendation is solely based on the effectiveness metrics.

The user can inspect (by mouse hovering) the data and select a region R of interest by clicking to pursue the exploration. In Fig. 3a, the user has selected the highest bar that designates the overly punctual flights of the airline with code WN. The selected region is highlighted in a different color. At this point, P_e is updated to include the initial query Q in its query history, the user interaction event (selection of bar with code=WN) as well as visual encoding parameters of the displayed chart.

Recommending the visualization of Fig. 3b (explained later) relies on the updated P_e to generate a similar visualization of same features, e.g., same axis scale for the y-axis, same order of airline codes on the x-axis, or choice of a stacked bar chart to maintain same heights as seen previously (e.g., in the bar-chart of Fig. 3a).

3.2 Data recommendation

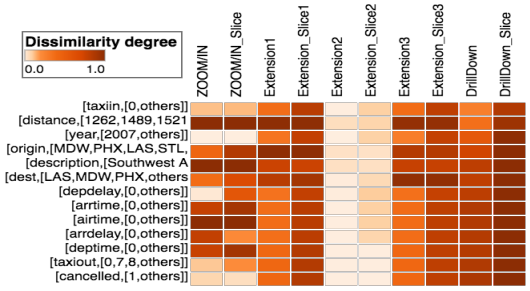


Figure 4: Impact matrix for running example

Selecting a particular sub-result $R \subseteq Q(D)$ of the data via the graphical user interface triggers the data recommendation component. It takes as input R, D, and Q to determine a set of attribute-value pairs $P = \{(a_1, v_1), \ldots, (a_n, v_n)\}$. Before passing these to query reformulation, the attribute-value pairs in P are grouped by attribute, resulting in the final output $G = \{(a_1, V_1), \ldots, (a_k, V_k)\}$. Essentially, data recommendation determines which data to explore next while query reformulation determines how these data will be explored.

To compute P, we leverage the data provenance of R, denoted $P_d(R)$, using the Perm provenance management system [6]. This provenance corresponds to all tuples in D that have contributed to producing R (i.e., why-provenance [4]). An attribute-value pair is then recommended if it satisfies one of the two following conditions: (i) it is widely present in $P_d(R)$ and more massively present in the database D or (ii) it is widely present in $P_d(R)$ but rarely present in D. We verify these conditions by first requiring a minimum frequency $f_{a,v}$ for an attribute-value pair in $P_d(R)$, i.e., $f_{a,v}(P_d(R)) \geq \theta_L$, where θ_L is a predefined threshold. We then compare this frequency to the frequency of the same attribute-value pair in the whole database D using the support measure defined by $support_{a,v}(R) = \left|\log_e\left(\frac{f_{a,v}(P_d(R))}{f_{a,v}(D)}\right)\right|$. Finally, only those attribute-value pairs with a lineage-based support above a given threshold θ_{supp} are retained for recommendation. In our prototype, threshold values of θ_L and θ_{supp} have been set to 0.1 and 0.7 respectively, which proved to be practical for the use cases we considered. However, setting these in general is an interesting avenue for future research. Using the method described above, it is possible that two distinct entries in P, i.e., (a, v) and (a', v') yield redundant query reformulations.

This is for instance the case when functional dependencies exist between attributes. To avoid redundant recommendations, we employ data profiling algorithms to determine functional dependencies [2] of the form $a \rightarrow a'$ and prune a'. The row labels of Fig. 4 show the set G, including for instance $(cancelled, \{1\})$ or $(dest, \{LAS, MDW, PHX\})$ that result from relevant attribute pairs $(cancelled, 1)$ and $\{(dest, LAS), (dest, MDW), (dest, PHX)\} \subseteq P$, respectively.

3.3 Query reformulation

The data recommendations are input to the query reformulation component that produces, for each $(a, V) \in G$, a set of queries corresponding to variations of the original query Q. Each variation reflects an operation typical when querying data warehouses. Our system supports variations implementing slice (and dice), drill-down, including the navigation to dimensions not considered in the initial query Q (which we call extension afterward), roll-up, and grouping or clustering the original results of Q by further attributes (zoom-in). The variations are systematically constructed based on Q, (a, V), and the specific data warehouse operation. For instance, the query reformulation for slice will add conditions of the form AND $a = v_1$ OR ... OR $a = v_n$ to the initial WHERE clause of Q, where $\{v_1, \ldots, v_n\} = V$, whereas a roll-up, drill-down changes the attribute set A in the SELECT and GROUP BY clauses of Q to a higher or lower granularity.

To assist users in choosing the next query for the next exploration step, we assign a utility score s to each query variation. Developing and evaluating suited scoring functions to quantify the interestingness of query variations based on their result data and candidate visualization properties is currently actively researched. As a proof-of-concept for our visual data exploration, we have currently implemented Kullback-Leibler divergence function [13] as a utility function. It quantifies the divergence of a's value distribution in $Q'(D)$ from its distribution in D.

The mapping of (a, V)-pairs to sets of scored recommended queries is visualized as an impact matrix. Fig. 4 shows the impact matrix that results from the query and interaction depicted in Fig. 2 and Fig. 3a. Each line of the matrix corresponds to an (a, V)-pair and each column corresponds to a type of query variation. The cell colors encode the divergence score. From this example, we see for instance that the zoom-in query for (airtime, 0) obtains a high interestingness score. Upon clicking this interesting cell, the corresponding query variation is set to be the new query Q, which is then executed before its result is visualized as recommended by the rendering recommendation component (see Fig. 3b).

The visualization of Fig. 3b shows that a significant number of flights satisfying the initially intended quality criteria for airlines, i.e., arrival ahead of schedule, has a flight duration equal to zero. Based on this insight, users may decide to revise or refine the airline quality criteria.

4 SCENARIOS AND USER EXPERIENCE

The demonstration will rely on scenarios from different domains. The domains are chosen so that some basic knowledge about database schemas and attributes can be assumed. Possible supported scenarios could leverage for instance the following datasets.

Flights. The first dataset describes US domestic flights[1]. It contains information about two million flights done by more than 1500 airline companies between 2007 and 2008. It includes further information about 3300 airports and almost 4500 plane types used for the covered flights. The facts recorded for each flight include various numerical attributes such delays, cancellation, arrival and departure time etc.

Movies. The second dataset describes one million ratings made by 6000 users of the MovieLens platform[2] on 4000 movies. This database stores various information about users and movies.

Soccer. The third dataset is the European soccer league database[3]. It contains detailed information about more than 25,000 fixtures between 2008 and 2016 in 11 European championships.

We expect the demonstration to attract a broad audience, generally interested in interactive data analysis or visual data exploration. Having some proficiency in SQL is crucial to be able to follow and express SQL queries that trigger an exploration session.

The audience experience will be similar to the sample scenario used throughout Sec. 3. However, wheres the example limits to one exploration step, attendees will have the opportunity to run exploration sessions spanning multiple exploration steps similarly to the user experience shown in [1].

In addition to experiencing the system's main functionality, the demonstration provides a tour "behind the scenes". This includes seeing how user interaction translates to a provenance query, which scores are computed for data recommendation, which attribute-values are pruned along the way, and which queries (with associated scores) are recommended. Ultimately, after hands-on data exploration experience using EVLIN, the audience will have gained a better understanding of the explored data set and possibly even discovered new insights on the underlying data set.

Acknowledgments. We thank the German Research Foundation (DFG) for supporting projects A03 and D03 of SFB-TRR 161.

REFERENCES

[1] 2017. EVLIN Demo. (2017). https://youtu.be/L_59cXZu0Uk
[2] Ziawasch Abedjan, Lukasz Golab, and Felix Naumann. 2015. Profiling relational data: a survey. *VLDB Journal* 24, 4 (8 2015), 557–581.
[3] Houssem Ben Lahmar and Melanie Herschel. 2017. Provenance-based Recommendations for Visual Data Exploration. In *TaPP*.
[4] James Cheney, Laura Chiticariu, and Wang Chiew Tan. 2009. Provenance in Databases: Why, How, and Where. *Found. Trends databases* 1, 4 (4 2009), 379–474.
[5] Marina Drosou and Evaggelia Pitoura. 2013. YmalDB: Exploring relational databases via result-driven recommendations. *VLDB Journal* 22, 6 (12 2013), 849–874.
[6] Boris Glavic and Gustavo Alonso. 2009. The Perm Provenance Management System in Action. In *SIGMOD*. 1055–1058.
[7] Melanie Herschel, Ralf Diestelkämper, and Houssem Ben Lahmar. 2017. A Survey on Provenance: What for? What Form? What from? *VLDB Journal* 26, 6 (12 2017), 881–906.
[8] Stratos Idreos, Olga Papaemmanouil, and Surajit Chaudhuri. 2015. Overview of Data Exploration Techniques. In *SIGMOD*. 277–281.
[9] Jock Mackinlay, Pat Hanrahan, and Chris Stolte. 2007. Show Me: Automatic Presentation for Visual Analysis. *TVCG* 13 (11 2007), 1137–44.
[10] Tova Milo and Amit Somech. 2016. REACT: Context-Sensitive Recommendations for Data Analysis. In *SIGMOD*. 2137–2140.
[11] Zening Qu and Jessica Hullman. 2016. Evaluating Visualization Sets: Tradeoffs Between Local Effectiveness and Global Consistency. In *BELIV*. 44–52.
[12] Manasi Vartak, Sajjadur Rahman, Samuel Madden, Aditya Parameswaran, and Neoklis Polyzotis. 2015. SEEDB: Efficient Data-Driven Visualization Recommendations to Support Visual Analytics. *Proc. VLDB Endow.* 8, 13 (9 2015), 2182–2193.
[13] Larry Wasserman. 2013. *All of statistics: a concise course in statistical inference.*
[14] Kanit Wongsuphasawat, Dominik Moritz, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. 2016. Voyager: Exploratory Analysis via Faceted Browsing of Visualization Recommendations. *TVCG* 22, 1 (2016), 649–658.
[15] Kanit Wongsuphasawat, Zening Qu, Dominik Moritz, Riley Chang, Felix Ouk, Anushka Anand, Jock D. Mackinlay, Bill Howe, and Jeffrey Heer. 2017. Voyager 2: Augmenting Visual Analysis with Partial View Specifications. In *CHI*. 2648–2659.

[1] https://stat-computing.org/dataexpo/2009/

[2] http://grouplens.org/datasets/movielens/

[3] https://www.kaggle.com/hugomathien/soccer