DataTweener: A Demonstration of a Tweening Engine for Incremental Visualization of Data Transforms

Meraj Khan[§] Larry Xu[‡] Arnab Nandi[§] Joseph M. Hellerstein[‡]
The Ohio State University[§] University of California, Berkeley[‡]
{khan.485,nandi.9}@osu.edu, {larry.xu,hellerstein}@berkeley.edu

ABSTRACT

With the development and advancement of new data interaction modalities, data exploration and analysis has become a highly interactive process situating the user in a session of successive queries. With rapidly changing results, it becomes difficult for the end user to fully comprehend transformations, especially the transforms corresponding to complex queries. We introduce "data tweening" as an informative way of visualizing structural data transforms, presenting the users with a series of incremental visual representations of a resultset transformation. We present transformations as ordered sequences of basic structural transforms and visual cues. The sequences are generated using an automated framework which utilizes differences between the consecutive resultsets and queries in a query session. We evaluate the effectiveness of tweening as a visualization method through a user study.

1. INTRODUCTION

Data analysis tasks often involve queries which produce large changes in results for relatively small changes in the queries. Consider the case of the pivot transformation, a very popular and useful transformation for analytical purposes. A small change in the query, e.g., changing from the query Q_1 to Q_2 causes a large change in the resultset, T_1 to T_{12} as seen in Figure 3.

```
Q1 = SELECT Id, Year, Dept
FROM StudentEnrollment E;

Q2 = SELECT * FROM (
SELECT Year, Dept
FROM StudentEnrollment E)
PIVOT (COUNT(Dept) FOR Dept IN ("ECE", "CSE"));
```

Such large jumps in the result space can leave the users disoriented as they can clearly notice something changed, but it is difficult to understand what the exact changes were and how they were caused. In such cases, *tweening* (an incremental visualization of the transformation) can help users get the right context and make the transformation easier to understand. This change can be brought about by a single text query on traditional query interfaces or through a touch gesture on modern direct-manipulation touch-based interfaces [2, 3, 6]. Direct-manipulation interfaces require continuous visual feedback during user interaction to keep the user engaged [7]. Hence, for these modern interfaces, tweening is a hard requirement.

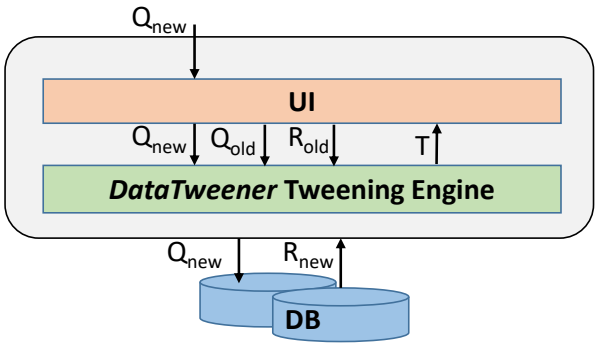

Figure 1: System design for a tweening-based query interface.

We provide a formalized a visual grammar of basic structural transforms and visual cues in our work *Data Tweening* [4], wherein we also presented a framework to generate ordered sequences of micro-operations from this grammar to encode resultset transformations. The visual grammar is designed considering the design principles of animated graphics [1, 9] and other works in perception research [8]. We visualize these ordered sequences as animated transforms on the query interface described here.

2. SYSTEM

The core of the data tweening based query interface is the tweening engine incorporated in the client. When a user issues a new query Q_{new}, the UI passes the query to the tweening engine, which then saves the old query Q_{old} and the old resultset R_{old}, and issues Q_{new} to the database for execution. Upon receiving the result R_{new} from the database, the tweening engine utilizes the old and new queries and corresponding resultsets to generate the tweening sequence T. This tweening sequence is then visualized on the UI as an animated transition from R_{old} to R_{new}. This is illustrated in Figure 1.

A tweening sequence is composed of basic transforms and visual cues defined on table entities (list of table cells, columns,

This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-nd/4.0/. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org.
Proceedings of the VLDB Endowment, Vol. 10, No. 12
Copyright 2017 VLDB Endowment 2150-8097/17/08.

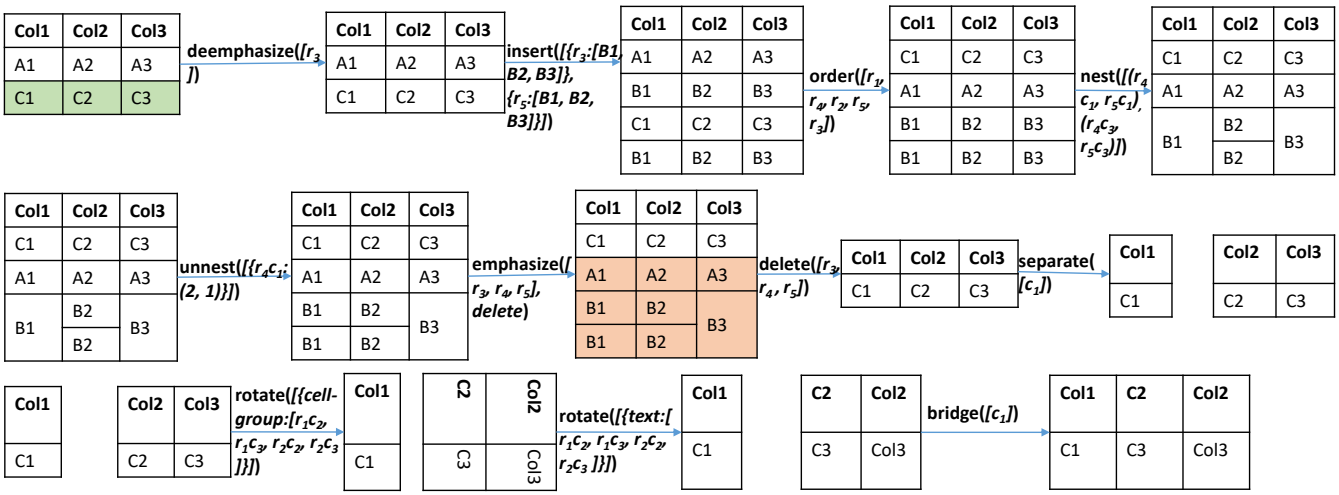

Figure 2: Micro-operations in the Visual Grammar

or rows). Figure 2 shows the effect of these micro-operations on a table.

The tweening sequence is generated using a fixed rulebase defined in Table 1. The rulebase defines the ordered sequences of basic transforms corresponding to changes in different SQL clauses. If two consecutive queries differ in multiple clauses, the tweening sequences for respective changes are concatenated to form the complete tweening sequence. In addition to the structural transforms, the sequences include visual cues which are inserted before / after certain transforms for providing hints to upcoming and current transforms in the sequence. The standard visual cues are inserted into the tweening sequence as per the following rules.

- *delete* is preceded by *emphasize*, and followed by *deemphasize* on the table entities on which delete is to be applied.
- *insert* is followed by *emphasize*, and then *deemphasize* on the newly inserted table entities.
- *order* is preceded by *annotate* showing a note with sorting column, and order (ascending or descending).

More difficult query changes like pivot are tweened through a different set of rules as shown in Figure 3. The tweening of pivot operation follows the natural logical construct of pivot, first showing the aggregation and then following it with the cross-tabulation. In certain cases where at least one of the queries from a consecutive pair of queries in a session is an aggregation query, the tweening generation becomes non-trivial as the tweening engine would need tuples constituting aggregation of either or both the queries to generate a meaningful animation. In such cases, instead of issuing the new query Q_{new} to the database, the tweening engine substitutes it with a simplified form of the expression- $disagg(Q_{old}) \bigcup disagg(Q_{new})$, where $disagg(Q)$ represents the SQL query which would return the constituent tuples forming the resultset corresponding to query Q. Our work in [4] details how and why this works. It also covers a few optimizations over these standard techniques which we do not implement in this demo.

3. QUERY INTERFACE

We present a web-based query interface with two interaction modalities – *text* and *multitouch* on two independent screens. On the text interface, the user can type in the desired SQL query in a text console, and see the result visualized through tweening in the same window. The multitouch interface defines intuitive touch gestures for basic relational data transforms- select, project, aggregate, and cross-tab. Each of these transform visualizations are composed of basic transforms and visual cues described in the earlier section. We restrict the type of queries that can be issued in a query session to a subset of SQL. We require the FROM clause in all the queries for a given session to be exactly the same.

3.1 SQL Console Interface

The users can type in a query in the console at the bottom right of the browser window. When the user types in a query, the visualization area is updated from the old resultset to the new resultset through an animated transition. The users start by issuing a SELECT query on one of the datasets available in the database. After the first query, the user needs to make sure they use the same FROM clause as that in the first query in all the subsequent queries. An exception to this rule is the pivot query which is required to have the table being pivoted in the FROM clause expression of the other queries in the session. Figure 4 shows the console area and the visualization area for this interface. The result for the query typed in the console is shown in the visualization area. The query execution is triggered by completion of the SQL statement in the console. If the user issues an invalid or unsupported SQL statement, an appropriate error message is shown on the top right corner of the screen.

3.2 Multitouch Interface

On the multitouch interface, the users can select a database table at the start of the session. The users can then issue queries on the selected dataset through touch gestures. For the demonstration, we provide support for basic relational operations along with cross-tabulation. The gestures for the respective operations are described below:

Table 1: Query-based tweening rulebase

Differing Clause	Change	Tweening Sequence
`SELECT`		delete(columns), insert(columns), order(columns)
`WHERE`		delete(rows), insert(rows), order(rows)
`GROUP BY`	New Group By Clause	order(rows), highlight(rows, cause=aggregation), insert(aggregateColumn), separate(groups), nest(rows), bridge(groups)
`GROUP BY`	Removed Group By Clause	delete(aggregateColumn), unnest(rows)

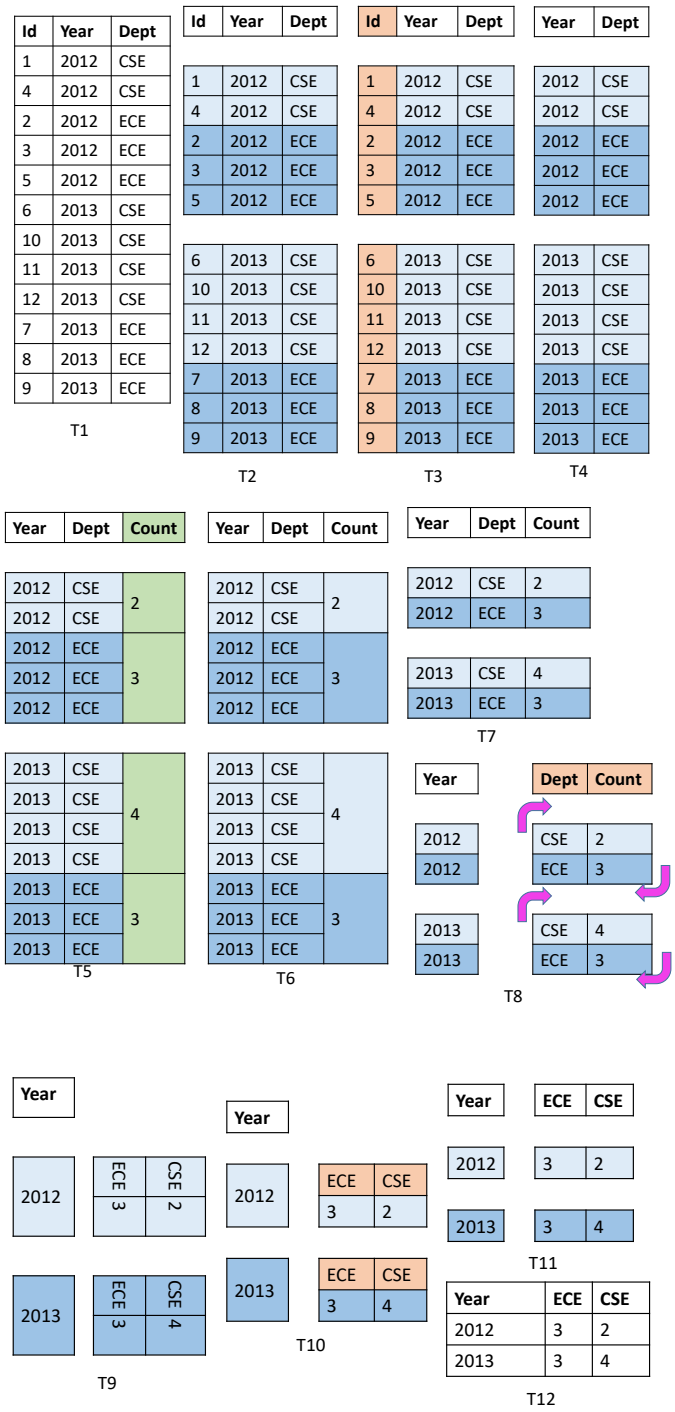

Figure 3: Incremental Visualization of Pivot Transformation introduced in Section 1

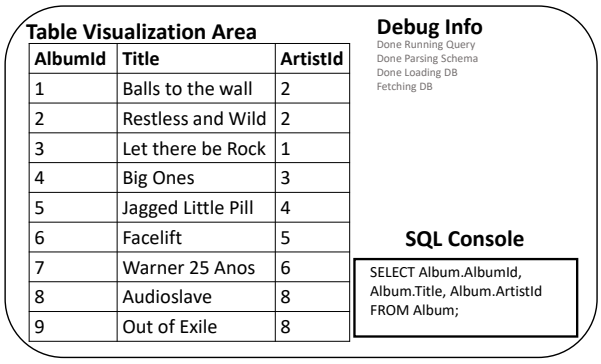

Figure 4: SQL console query interface

- SELECTION - The query interface provides two mechanisms for selection. Users can either *pinch-in* (Figure 5) rows to delete them or they can specify an attribute-specific WHERE clause predicate in the pop-up dialog opened by tapping on a column header. The predicates are defined in the same syntax as that in standard SQL. For deleting rows using the pinch-in gesture, users need to tap on two cells in the same column separated by at least one row and swipe inwards. This would cause all the rows in between the two touchpoints (excluding the rows containing them) to be deleted.

- PROJECTION - The users can delete a column with the pinch-in touch gesture (Figure 5). This works exactly like the pinch-in gesture for selection. The users can select two distinct cells in the same row and swipe inwards to delete the block of columns between the two points.

- AGGREGATION - The users can initiate the aggregation gesture by a single finger swipe directed from the measure column to the group-by column after which they would be prompted to pick the aggregation function. The swipe gesture for aggregation and the final output for a sample table is shown in Figure 6.

- CROSS-TABULATION - A cross-tab can be performed by an arc-shaped swipe gesture originating at a cell in the column to be transposed to the attribute headers and directed upwards towards the table headers while holding down a cell in the pivot column. The gesture is depicted in Figure 7. To maintain a continuous visual feedback, once the system figures out the cross-tab query during the swipe gesture, it starts showing the transition to the new resultset. For example, for the cross-tab query demonstrated in Figure 7, an animated transition from T_7 to T_{12} in Figure 3 is played.

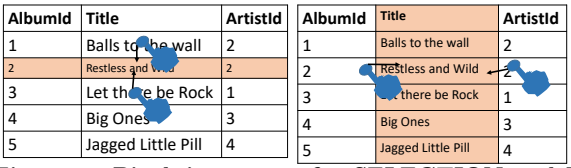

Figure 5: Pinch-in gesture for SELECTION and PROJECTION on multitouch interface

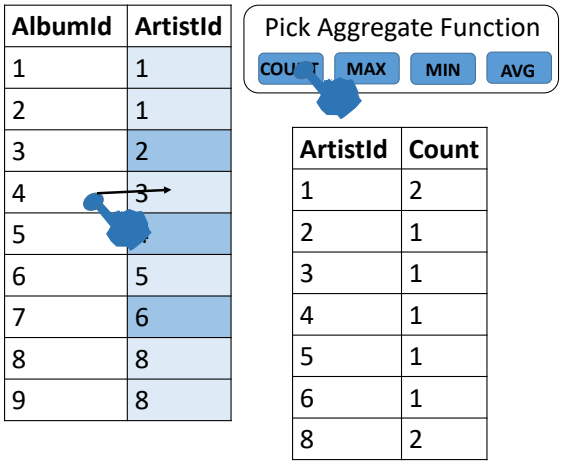

Figure 6: Aggregation on multitouch interface

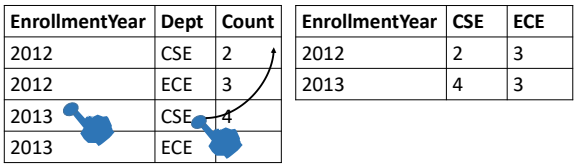

Figure 7: Cross-tabulation on multitouch interface

4. USER STUDY

To study the effectiveness of tweening as a visualization method and its role in helping people understand complex transformations, we conducted a user study [5]. We recruited 20 undergraduate and graduate students from different backgrounds to participate in an in-persosn study. The users were asked to rate our pivot tweening sequence for clarity and understandability on a scale of $0 - 10$. We also asked them if the tweening helped them understand the pivot transformation better. Figure 8 shows the rating for pivot tweening by each user. 18 out of the 20 users said they found tweening helpful in understanding pivot. The users rated the tweening sequence high both for clarity (mean - 8.23, standard error - 1.24) and understandability (mean - 8.48, standard error - 1.31) Through this study, we conclude users find tweening helpful in understanding complex transformations and hence is an effective visualization method.

5. DEMONSTRATION

For the demonstration, we will present a web app on a touch-screen laptop. The users can opt to either use the console-based interface or the multitouch interface. The users can select a publicly available datasets hosted on an in-browser sqlite database. Users can see complex data transformations broken down and visualized into simple and smooth transitions between distinct resultsets. This is in

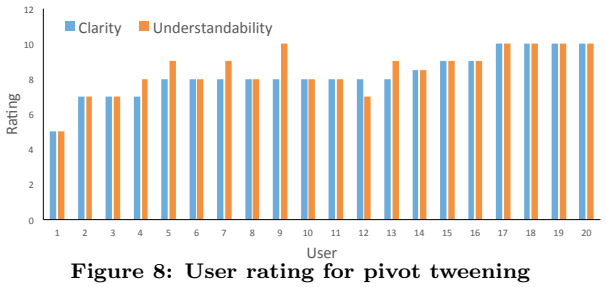

Figure 8: User rating for pivot tweening

contrast to the standard sql resultset visualization where the changes in resultsets are abrupt and discontinuous. For example, the pivot transformation corresponding to a change in query Q_1 to Q_2 introduced in Section 1 would be displayed as an animated transition stepping through the frames shown in Figure 3. The same transformation on the standard query interface would be displayed as an abrupt change from T_1 to T_{12}. A demo video (**http://go.osu.edu/datatweener**) is provided for elucidation.

6. CONCLUSION AND FUTURE WORK

This work describes an initial implementation of a query interface with support for data-tweening. Users can get real-time feedback in the form of animated transition for a subset of SQL queries. We have shown that the visual grammar has the expressivity to encode a wider SQL subset than supported by the current implementation. In the future, we plan to build a full-fledged query interface with tweening support, and add playback capability for all actions. We also plan a comparative study to see if users perform any better on data analysis tasks with a tweening enabled query interface over traditional query interfaces. Another interesting future work is extending tweening to very large datasets.

Acknowledgements We acknowledge the support of the U.S. National Science Foundation under awards IIS-1422977, IIS-1527779, CAREER IIS-1453582, and IIS-1564351.

7. REFERENCES

[1] Heer, Jeffrey and Robertson, George G. Animated Transitions in Statistical Data Graphics. *TVCG*, pages 1240–1247, 2007.

[2] Idreos, Stratos and Liarou, Erietta. dbTouch: Analytics at your Fingertips. *CIDR*, 2013.

[3] Jiang, Lilong and Nandi, Arnab. Designing interactive query interfaces to teach database systems in the classroom. *SIGCHI*, pages 1479–1482, 2015.

[4] Khan, Meraj and Xu, Larry, and Nandi, Arnab and Hellerstein, Joseph M. Data tweening: Incremental visualization of data transforms. *VLDB*, pages 661–672, 2017.

[5] Kosara, Robert and others. Thoughts on User Studies: Why, How, and When? *CGA*, pages 20–25, 2003.

[6] Nandi, Arnab and Jiang, Lilong and Mandel, Michael. Gestural Query Specification. *VLDB*, pages 289–300, 2013.

[7] Shneiderman, Ben. Direct Manipulation: A Step Beyond Programming Languages. *Sparks of innovation in HCI*, page 17, 1993.

[8] Todorovic, Dejan. Gestalt Principles. *Scholarpedia*, 2008.

[9] Tversky, Barbara and Morrison, and Julie Bauer and Betrancourt, Mireille. Animation: Can It Facilitate? *IJHCS*, pages 247–262, 2002.