dbTouch in Action
Database Kernels for Touch-based Data Exploration

Erietta Liarou
EPFL
erietta.liarou@epfl.ch

Stratos Idreos
Harvard University
stratos@seas.harvard.edu

***Abstract*—A fundamental need in the era of data deluge is *data exploration through interactive tools*, i.e., being able to quickly determine data and patterns of interest. dbTouch is a new research direction towards a next generation of data management systems that inherently support data exploration by allowing touch-based interaction. Data is represented in a visual format, while users can touch those shapes and interact/query with gestures. In a dbTouch system, the whole database kernel is geared towards quick responses in touch input; the user drives query processing (not just query construction) via touch gestures, dictating how fast or slow data flows through query plans and which data parts are processed at any time. dbTouch translates the gestures into interactive database operators, reacting continuously to the touch input and analytics tasks given by the user in real-time such as sliding a finger over a column to scan it progressively; zoom in with two fingers over a column to progressively get sample data; rotate a table to change the physical design from row-store to column-store, etc. This demo presents the first dbTouch prototype over iOS for iPad.**

I. Introduction

Interactive Data Exploration. dbTouch [9] expresses a novel research direction that addresses the critical need of data exploration in the big data era [4], i.e., when we are in search for interesting patterns often not knowing a priori exactly what we are looking for. For example, an astronomer wants to browse parts of the sky to look for interesting effects, while a data analyst of an IT business browses daily data of monitoring streams to figure out user behavior patterns. What both cases have in common is a daily stream of big data, i.e., in the order of multiple Terabytes and the need to observe *something interesting and useful*. Interactive data exploration aims to allow for instant access to the data, i.e., without expensive initialization steps, while at the same time allowing the user to extract knowledge from the data by selectively and interactively investigating parts of the data.

dbTouch. The vision of dbTouch was recently proposed [9] with the goal to lead towards interactive data exploration by bringing together touch interfaces and core database research, i.e., to allow users to *touch and manipulate data* in intuitive ways in search for interesting patterns. dbTouch proposes to rethink database kernel designs towards architectures tailored for touch-based data exploration.

In the dbTouch context, data is visualized in various shapes and forms while users can interact with those shapes via touch input. For example, an attribute of a given table may be represented by a column shape. Then, a user will be able to apply gestures on this shape to get a feeling of the data stored in the column, and to run any kind of simple or complex query.

The fundamental concepts of query, query plan and data flow are redefined. The user's touch, the gesture evolution, i.e., the speed and the direction of the gesture, determine the data to be processed next and the actions that need to be performed. The system does not try to consume all data; instead, it analyzes only parts of the data at a time, continuously refining the answers and continuously reacting to user input. Every single user touch on a data object can be seen as a request to run an operator or a collection of operators over a part of the data. At the same time, as a gesture, i.e., a collection of touches, evolves and varies in direction and speed, users can determine the data they need to analyze, reacting to intermediate results as they appear on screen. The database system does not have control of the data flow anymore.

Beyond Query Construction. Compared to pioneering state-of-the-art drag and drop systems such as Tableau, dbTouch shares a lot of motivation but goes a step further in terms of how interactive it is. Such systems help primarily with query construction. dbTouch is not only about query construction but mainly about more fine grained data exploration. For example, in a drag and drop system a user may construct a query by dragging a column onto an aggregate operation (which is indeed a very intuitive action). Still though, the back-end system is a standard database engine that needs to consume and process the whole column in order to provide the result. With big data this becomes a bottleneck. dbTouch is about making this step interactive and adaptive, allowing users to "touch the column data", processing just a few data entries at a time and ask for more data to be processed as they observe running results. Here we use the notion of touch gestures as our starting gesture input example but any kind of interactive input means would apply. For example, the same dbTouch techniques can be applied in a desktop interface where one applies gestures via a mouse or a touch-pad on visual data objects. The dbTouch vision is that through the integration of intuitive gestures and a new class of interactive database kernels we enable new ways for users to have a *quick look and feel* of their data in a natural and interactive way.

From Gestures to Query Processing. dbTouch creates new research opportunities both in the area of database architectures and in the area of visualization. From a database researcher's point of view there are several fundamental questions to answer. Most critical questions have to do with

how we translate touch gestures to database query processing algorithms/operators, i.e., how does dbTouch react in terms of storage and access patterns when we slide a finger or when we zoom-in with two fingers over a column or a table? What is the equivalent of a query? Several challenges arise. For example, when sliding a finger to scan or to run an aggregation over a column, users may choose to slide faster or slower or change the slide speed numerous times or even pause for some time, while they are observing the running results.

The interactive and continuously changing mode in dbTouch is drastically different than what state-of-the-art database systems support; it requires rethinking of core algorithms, while new concepts arise when it comes to interactive query processing. In traditional systems, once a query is posed, the database controls the data flow, i.e., it is in full control regarding which data it processes and in what order, such as to compute the result to the query. In dbTouch, however, these concepts are blurred; a query is a session of one or more continuous gestures and the system needs to react to every touch, while the user is now in control of the data flow.

Other than translating gestures to database algorithms, there are numerous optimization issues in dbTouch. For example, when a user slides a finger over a column with a varying slide speed, then we would like to find a good way and timing to extrapolate the gesture movement and to efficiently access, prefetch and precompute the anticipated data to avoid stalling once the query session resumes or when it moves faster.

Exploiting dbTouch. The dbTouch system is primarily designed to work as an exploration tool, i.e., to speed up understanding of data. For example, a user may choose to load a small data sample directly on a tablet mobile dbTouch system or use the tablet as an interface to a cloud based dbTouch system over the complete data set.

Contributions and Demo. Paper [9] presents the basic dbTouch vision, the basic architecture and sets the research path and critical milestones. Here, we present a demo that showcases the key aspects of our first dbTouch prototype. The demo allows the audience to experience interactive data exploration in an iPad and to engage in a contest with other demo participants to compare this experience against a typical modern database system. The dbTouch prototype provides various interactive operators and gestures such as sliding a finger over a column to scan it progressively or to run aggregations; zoom in or out with two fingers over a column to progressively get sample data; rotate a table to change the physical design from row-store to column-store, etc.

II. DBTOUCH

This section gives more details on the dbTouch prototype. The main challenge is to redesign database kernels such that they support near instant reaction to touch input.

Visualizing Data. Objects appear on the touch screen while a user can apply gestures on these objects. For simplicity, our initial design assumes a straightforward visualization option, i.e., data appears in a relational-like way; tables are represented as (fat) rectangles and attributes are represented as columns.

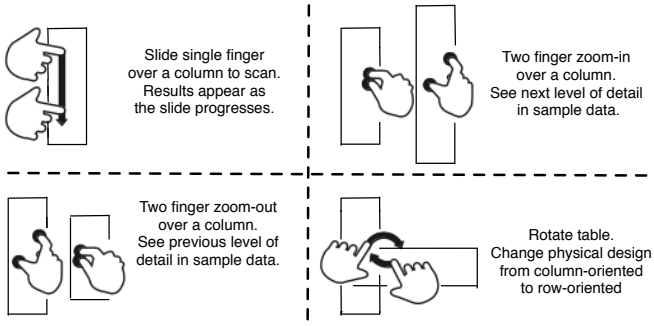


Fig. 1. Examples of dbTouch gestures.

Figure 1 shows several examples of column representations and gestures. The objects are abstract representations of the data. For example, in Figure 1 we see that the actual data is not visible. A single column of a height of only a few centimeters may represent an attribute in a table with several millions of tuples. Thus, the actual data becomes visible, only by applying one or more gestures on a data object. The various objects convey the schema information at a high level view, giving a glimpse to the user regarding what kind of data is available for inspection via gestures.

Interactive Exploration. One of the main requirements when inspecting data is to get a quick feeling regarding the quality of the data and possible patterns and properties. Not necessarily all data is required at this stage while at the same time this is not an exact science; it involves a lot of intuition from the analyst which is why many people claim that (big) data analysis is a kind of art [5]. In this way, the key goal of dbTouch is to assist users with *data exploration*.

dbTouch provides an *interactive* feeling to the users, by giving them the illusion that they are indeed "touching the data" in real time. This increases user satisfaction and ease of using the system. To achieve this goal, dbTouch operators are not monolithic; they do not consume big piles of data at a time, resulting in significant delays in response times. Instead, dbTouch provides *incremental* and *adaptive* operators that give quick results back to the user. In turn, users reflect on those results and adjust their gestures accordingly.

The interactive data exploration requires a combination of both low level query processing actions and visualization techniques of properly visualizing data and intermediate results. Good query processing techniques increase the response times of the system when reacting to user requests (gestures), while good visualization techniques increase the response time of the user when reacting to results produced by the system.

dbTouch in Action. A snapshot of the dbTouch prototype in action is shown in Figure 2. In general, several objects may be visible at any time, representing data (columns and tables) stored in the database. The user has the option to touch and manipulate whole tables or to visualize and work on the columns of a table independently for more fine grained access and analysis. In the example of Figure 2, the user sees 3 independent columns, each one visualized as a separate rectangle and with a different color.

Slide: Scan and Aggregates. The most basic action when having a first exploratory look at a new set of data is achieved

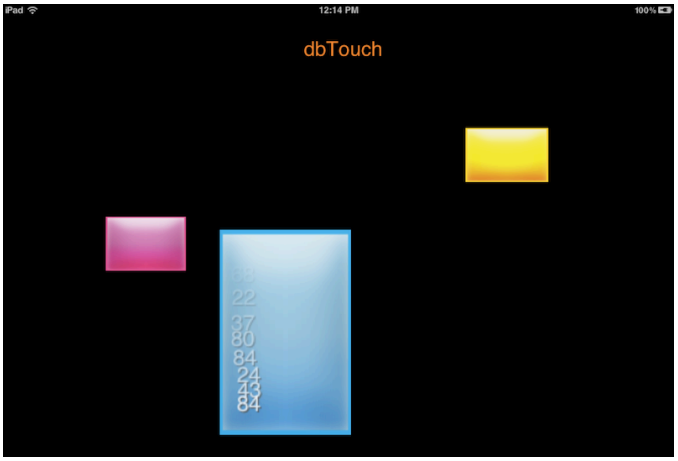


Fig. 2. Screenshot of dbTouch.

either by having access at the actual data values or by running simple aggregations. dbTouch realizes this goal by the *slide* operator which is triggered by the slide gesture.

The upper left part of Figure 1 depicts an example of the slide gesture; the idea is that the user slides a single finger over the data object to *point* at data to be inspected. Figure 2 shows a slide gesture in action on the actual dbTouch prototype. The gesture is applied on the blue data object in (the middle of) Figure 2 where data values appear as the gesture progresses. The screen-shot is taken directly from the iPad where we tested our prototype using the screen-shot functionality in XCode. Naturally, the actual user finger is not visible in Figure 2; in this case the user slides a finger starting from the top of the blue object all the way to the bottom of the object.

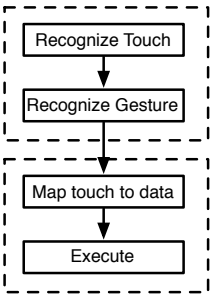
Touching Data: From Touch to Tuple Identifiers. Data objects represent tables and columns. A key step is in translating the location of a touch over a data object to a tuple identifier of the table or the column represented by the object, i.e., determining which data entry corresponds to the touch. For example, with a single tap over a data object dbTouch shows a single data entry which appears and fades away (as in Figure 2). A slide gesture can be seen as multiple continuous single taps in successive positions of a data object. In order to translate the location of a touch to a tuple identifier, dbTouch exploits the *view* concept of modern touch-based operating systems by correlating the size of the visual object with the cardinality of the underlying data objects.

Storage Layout and Touch. The dbTouch vision does not pose any particular restrictions on the underlying storage model. It can be a row-store, a column-store or a hybrid format. Our current design is based on a flexible hybrid format; each relational table is stored as a collection of column-groups and following common practices of most modern systems each column is a fixed-width array [1]. For expert users part of the exploration process may also involve changing the schema or the storage layout for organization or for performance reasons which can be achieved by drag and drop actions to bring objects together or to separate them.

Operators. The key element in the design of dbTouch is the design of underlying operators which can quickly react to continuous touch input. Consider every tap as if it is a query in a typical DBMS. Our current prototype supports slide operations which fetch data dynamically following the speed and direction of the gesture. Scans and aggregates run on top of sample data as a single slide gesture will not touch all rowIDs. A slower or faster slide touches more or less tuples respectively. Zoom in and zoom out gestures allow to increase or decrease the granularity of the working data set.

Implementation. Our dbTouch prototype is essentially a prototype database kernel implemented on top of iOS in objective C. In many ways it resembles a hybrid kernel where data is stored in fixed-width dense arrays or matrixes which can be seen as collections of >1 of such arrays. In this way, in terms of data storage dbTouch resembles modern systems. The critical difference in dbTouch is the way data is accessed and the way query processing actions are triggered and processed which requires rethinking of the core database design.

Similarly to the typical SQL input-parser-optimizer-execution flow of modern database systems, a dbTouch system goes through a flow that begins with a touch input, and continues with gesture recognition. Then, depending on the gesture, the location touched, the size of the object touched, etc., the proper execution methods are called.



The figure on the left-hand side shows a high level view of this task flow and how the various system layers interact. This flow is not per query as it is in database systems; instead dbTouch goes through these steps for every touch input on a data object.

Complex Queries. Any kind of query is possible with dbTouch. For example, a user may set-up any kind of complex query (which may include joins, group by operators, etc) and then use slide gestures to drive the flow of data in an interactive way, while observing running results. Setting up a query is purely a HCI issue; for example a join can be defined by bringing together and overlapping two columns or simply by double tapping and presenting a menu with the available options. The challenge in terms of database architecture design is how do we interactively provide instant results when the actual query processing starts. To provide instant and interactive results dbTouch raises the challenge to design interactive operators which may sacrifice correctness for speed [9].

III. Demonstration

We have developed a dbTouch prototype over iOS SDK 5.1 (9B176) for iPad. The demo focuses on exposing the exploration properties of dbTouch.

Set-up and Scenarios. The audience will get direct access to the dbTouch prototype via an iPad that we will provide and will get the chance to use the system and interact via gestures to query and explore data. We will provide various data sets with a varying set of properties and patterns. The audience will have the task of discovering these properties.

Exploration Contest dbTouch Vs. DBMS. In addition, the audience can participate in an exploration contest. We will provide a laptop installed with a traditional DBMS loaded with the same data sets as dbTouch. Two audience members will simultaneously start exploring the data set; one member will be using the tablet dbTouch prototype, while the other member will be using the SQL interface of the DBMS on the laptop. Both members will be free to perform any kind of query processing actions, i.e., to apply any kind of supported gestures in dbTouch and to fire any kind of standard SQL queries in the DBMS. The winner is the one who can first figure out the data properties and patterns.

Available dbTouch Gestures. Our prototype supports exploration via zoom-in and zoom-out gestures that increase or decrease a data object size allowing more or less fine grained access to the underlying data. The slide gesture allows to walk through a data object in order to perform various aggregations or simply to scan data or even to join data from > 1 data objects. Each user touch can be tuned to result in touching just one or multiple tuples (interactive summaries).

IV. Related Work

dbTouch takes inspiration from several research areas.

Data Exploration. Several researchers argue towards exploration based database kernels, e.g., with sampling based kernels [2], [11], [14], adaptive indexing [8] and adaptive data loading [7], [3]. Overall, this is a quite promising and largely unexplored research area. dbTouch complements ongoing research efforts by providing a promising alternative when it comes to how users interact with an exploration based database kernel. Ideas such as sampling, adaptive indexing or loading can be exploited in the dbTouch context with new challenges on how to adapt to the dynamic touch patterns in gestures.

Online Aggregation. Online aggregation [6], [13] is also related to dbTouch. In online aggregation the system continuously returns results as they are created. A confidence metric is also calculated and reported, allowing the user to terminate query processing when confidence reaches satisfactory levels. Online aggregation techniques can certainly be exploited in dbTouch; dbTouch brings additional challenges as the user drives the speed of requesting more data and determines the data to be processed dynamically.

Visual Analytics. The idea that simple text mode can hurt usability is not new. Polaris is the pioneering system from Stanford University for visual analytics [15], [16]. In Polaris, there are two distinct features to ease usability and exploration: (a) users can synthesize SQL queries by drag and drop actions and (b) results appear directly in the proper visual format. For example, results can appear directly in a bar graph or in other graph formats depending on the kind of data. The ideas pioneered in Polaris and later commercialized in the Tableau system are directly in line with the dbTouch vision. In addition to Tableau, there are more commercial systems exploiting similar ideas. In particular Data Clarity and VisuaLinks from Visual Analytics Inc. and the Visual Analytics platform by SAS. All these systems have the same high level goal; they try to provide an easy way to construct queries graphically and to visualize results.

What dbTouch brings is the idea of building database kernels to inherently support touch interfaces and interactive exploration at their core, taking visual analytics one major step further by allowing systems to increase their interactive and exploratory character. In dbTouch users do not simply create queries which will then run in a typical back-end system; instead they drive the actual low level query processing actions which results in a more interactive data exploration system.

Gesture-based Systems. The vision of gesture based systems is a new path expressed by dbTouch [9] and also by the vision of keyboard-free systems [12], [10]. Both visions appeared concurrently and have the same high level goal; keyboard free systems research is mostly focused on creating novel gesture-based languages for users to be able to pose expressive queries [12], [10], while dbTouch is mostly focused on the interactive database architectures aspect of the vision. A notable example from past research is the system Timber, proposed in the early 1980s [17]; it essentially shares the same high level motivation with dbTouch and keyboard free systems, allowing for data exploration via "browsing relations" interactively.

V. Summary

In this paper, we present a demonstration of dbTouch [9]; a new research direction towards touch based database kernels to ease interactive data exploration in the big data era.

References

[1] D. Abadi, P. Boncz, S. Harizopoulos, S. Idreos, and S. Madden. The Design and Implementation of Modern Column-Oriented Database Systems. *Foundations and Trends in Databases*, 5(3), 2012.

[2] S. Agarwal, A. Panda, B. Mozafari, S. Madden, and I. Stoica. Blink and it's done: Interactive queries on very large data. In *PVLDB*, 2012.

[3] I. Alagiannis, R. Borovica, M. Branco, S. Idreos, and A. Ailamaki. Nodb: efficient query execution on raw data files. In *SIGMOD*, 2012.

[4] S. Chaudhuri. What next? a half-dozen data management research goals for big data and cloud. In *PODS*, 2012.

[5] P. Hanrahan. Analytic database technologies for a new kind of user - the data enthusiast. In *SIGMOD*, 2012.

[6] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. In *SIGMOD Conference*, 1997.

[7] S. Idreos, I. Alagiannis, R. Johnson, and A. Ailamaki. Here are my data files. here are my queries. where are my results? In *CIDR*, 2011.

[8] S. Idreos, M. Kersten, and S. Manegold. Database Cracking. CIDR'07.

[9] S. Idreos and E. Liarou. dbtouch: Analytics at your fingertips. CIDR'13.

[10] L. Jiang, M. Mandel, and A. Nandi. Gesturequery: A multitouch database query interface. *PVLDB*, 6(12):1342–1345, 2013.

[11] M. L. Kersten, S. Idreos, S. Manegold, and E. Liarou. The researcher's guide to the data deluge: Querying a scientific database in just a few seconds. *PVLDB*, 4(12):1474–1477, 2011.

[12] A. Nandi. Querying without keyboards. In *CIDR*, 2013.

[13] N. Pansare, V. R. Borkar, C. Jermaine, and T. Condie. Online aggregation for large mapreduce jobs. *PVLDB*, 4(11):1135–1145, 2011.

[14] L. Sidirourgos, M. L. Kersten, and P. A. Boncz. Sciborq: Scientific data management with bounds on runtime and quality. In *CIDR*, 2011.

[15] C. Stolte, D. Tang, and P. Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Trans. Vis. Comput. Graph.*, 8(1):52–65, 2002.

[16] C. Stolte, D. Tang, and P. Hanrahan. Query, analysis, and visualization of hierarchically structured data using polaris. In *KDD*, 2002.

[17] M. Stonebraker and J. Kalash. Timber: A sophisticated relation browser. In *VLDB*, pages 1–10, 1982.