Exploratory Computing: What is there for the Database Researcher?

Marcello Buoncristiano [1], Giansalvatore Mecca [1], Elisa Quintarelli [2], Manuel Roveri [2], Donatello Santoro [1], and Letizia Tanca [2]

[1] Università della Basilicata – Potenza, Italy
[2] Politecnico di Milano – Milano, Italy

(Discussion Paper)

Abstract. The need for effective tools helping users to make sense of very big datasets has received a lot of attention lately. In this paper we propose a paradigm for *database exploration* which is in turn inspired by the *exploratory computing* vision [2]. We may describe exploratory computing as the step-by-step "conversation" of a user and a system that "help each other" to refine the data exploration process, ultimately gathering new knowledge that concretely fulfills the user needs. In this broad and innovative context, this paper proposes a model to concretely perform this kind of exploration over a database. The model is general enough to encompass most data models and query languages that have been proposed for data management in the last few years. At the same time, it is precise enough to provide a first formalization of the problem and reason about the research challenges posed to database researchers by this new paradigm of interaction.

1 Introduction

The need for effective tools helping users to make sense of very big datasets has received a lot of attention lately. However, the tools that are available for data analysis purposes typically address professional *data scientists*, who, besides a deep knowledge of the domain of interest, master one or more of the following disciplines: mathematics, statistics, computer science, computer engineering, and programming.

On the contrary, in our vision it is essential to support also different kinds of users who, for various reasons, may want to analyze the data and obtain new insight from them. Examples of these *data enthusiasts* [4] [8] are journalists, investors, or politicians: non-technical users who can draw great advantage from exploring the data, achieving new and essential knowledge, instead of reading tons of records coming out of the database as a query result.

The term *data exploration* generally refers to a data user being able to find her way through large amounts of data in order to gather the necessary information. A more technical definition comes from the field of statistics, introduced by Tukey [11]: with *exploratory data analysis* the researcher explores the data in many possible ways, including the use of graphical tools like boxplots or histograms, gaining knowledge from the way data are displayed.

Despite the emphasis on visualization, exploratory data analysis still assumes that the user understands at least the basics of statistics, while in this paper we propose a paradigm for *database exploration* which is in turn inspired by the *exploratory computing* vision [2]. We may describe exploratory computing as the step-by-step "conversation" of a user and a system that "help each other" to refine the data exploration process, ultimately gathering new knowledge that concretely fulfils the user needs. The process is seen as a conversation since the system provides active support: it not only answers user's requests, but also suggests one or more possible actions that may help the user to focus the exploratory session. This activity may entail the use of a wide range of different techniques, including the use of statistics and data analysis, query suggestion, advanced visualization tools, etc.

The closest analogy envisioned in [2] is that of a human-to-human dialogue, in which two people talk, and continuously make reference to their lives, priorities, knowledge and beliefs, leveraging them in order to provide the best possible contribution to the dialogue. In essence, through the conversation they are exploring themselves as well as the information that is conveyed through their words. This exploration process therefore means investigation, exploration-seeking, comparison-making, and learning altogether. It is therefore most appropriate for *big collections of semantically rich data*, which typically hide precious knowledge behind their complexity.

In this broad and innovative context, this paper intends to make a significant step further: it proposes a model to concretely perform this kind of exploration over a database. The model is general enough to encompass most data models and query languages that have been proposed for data management in the last few years. At the same time, it is precise enough to provide a first formalization of the problem and reason about the research challenges posed to database researchers by this new paradigm of interaction.

2 A Motivating Example

We illustrate the process of exploring a large and semantically rich dataset using as example the database of recordings of the fitness tracker AcmeBand, a wrist-worn smartband that continuously records user steps, and can also be used to track sleep.

Our user is allowed access to the measurements of a large fragment of AcmeBand users; in this example, for the sake of simplicity we shall assume that the database is a relational one, and focus on conjunctive queries as the query language of reference. However, we want to emphasize that the techniques discussed in this paragraph can be applied to any data model that is based on the primitives of object collection, attribute and attribute value, and object reference: as a consequence our approach may incorporate the majority of data models that have been proposed in the last few years to model rich data. In our simplified case, the database has the following structure:

(i) a AcmeUser(id, name, sex, age, cityId) table with user data;

(ii) a Location(id, cityName, state, region) table to record location data about users (here region may be east, west, north or south);

(iii) an Activity(id, type, date, start, length, userId) table to record step counts for user activities of the various kinds (like walks, runs, cycling etc.);

(iv) a Sleep(id, date, start, length, quality, userId) table to record user sleep and its quality (like deep sleep, restless etc.).

We notice that the database may be quite large, even if the number of users is limited and the timeframe for activities and sleep restricted. Our casual exploratory user intends to acquire some knowledge about fitness levels and sleep habits of this fragment of AcmeBand users.

We do not assume any a-priori knowledge of a database query language, nor the availability of pre-computed summaries as the ones that are typically used in OLAP applications. On the contrary, the system we have in mind should be able to guide and support our users throughout their (casual) information-seeking tasks.

2.1 Starting the Conversation

We envision this process as a *conversation* between the exploratory user and the system, where s/he provides some initial hints about her/his interests, and the system suggests "potentially interesting perspectives" over the data that may help to refine the search.

From the technical viewpoint, the conversation is modeled as a lattice of nodes. Each node is a *view* over the data, described intensionally as a conjunctive query over the database, and therefore represents a subset of database objects (*tuples*, in our relational case). To formalize this notion, we assume we are given a relational database schema $\mathcal{R} = \{R_1, \ldots, R_k\}$, and an instance I of $\mathcal{R}$, defined as usual. We introduce the notion of a *tuple-set* $\mathcal{T}^Q$ as the result of any conjunctive query Q over I, i.e., $\mathcal{T}^Q = Q(I)$.

In our example, we assume that the user does not have clear goals, and therefore expects the system to suggest some promising starting points for the exploration. However, we will see that our model can also handle the alternative scenario in which the user starts the conversation by formulating some explicit, albeit vague query.

Given our sample database, to start the conversation the system suggests some initial *relevant features*. We need to make this notion more precise, but for the time being consider that, in the relational context, we may assume that a feature is the set of values taken by one or more attributes in the tuple-sets of the view lattice, while relevance is defined starting from the statistical properties of these values. As concrete examples, consider that to trigger the exploration the system might suggest something along the following lines:

$(\mathcal{S}_1)$ "It might be interesting to explore the types of activities. In fact: *running* is the most frequent activity (over 50%), *cycling* the least frequent one (less than 20%)";

$(\mathcal{S}_2)$ "It might be interesting to explore the age of users. In fact: more than 60% of the users is in the 35-55 range";

$(\mathcal{S}_3)$ "It might be interesting to explore the quality of sleep periods. In fact: 65% of the sleep periods have bad quality".

2.2 A Model of Relevance

Before going on with our example, we want to discuss how relevant features are extracted. The notion of relevance is based on the frequency distribution of attributes in

the view lattice. To start the conversation, the system populates the lattice with a set of initial views. These will typically correspond to the database tables themselves, and joins thereof according to foreign keys. Broadly speaking, each view node can be seen also as a *concept* of the conversation, described by a sentence in natural language. For example, the node corresponding to the Activity table represents an "activity" concept (a *root* concept, from where the system may start the conversation), while the join of AcmeUser and Location represents the concept of "user location", and so on.

The system builds histograms for the attributes of these views, and looks for those that might have some interest for the user. We adopt a comparative notion of relevance which has an information-theoretic root, and look for attributes that show some significant deviation wrt an expected distribution. From the practical viewpoint, for each attribute we identify a number of reference distributions that correspond to expected – or *uninteresting* – ones, and deem the attribute *relevant* when its distribution shows some clear difference wrt the reference ones. To do this, we assume the availability of a statistical similarity test, denoted by $test(d, d')$, for value distributions d, d'. Examples of this test are statistical hypothesis tests (e.g., the *t-test* or the *chi-square test*).

For a root concept, the reference distribution might be the uniform one, although different other choices are possible according to the semantics of the attribute. In our example, the type attribute of Activity is considered as potentially relevant (suggestion $\mathcal{S}_1$ above), since its distribution shows significant statistical difference, according to $test$, w.r.t. to the uniform distribution. Note that $\mathcal{S}_1$ provides, along with a description of the relevant feature, some values as an explanation of these differences.

For views that are lower in the node lattice the notion of relevance is slightly more sophisticated. To introduce this, we need to formalize the lattice structure of views and tuple-sets. In our simple relational example we do so by reasoning about the relationship among conjunctive queries: we say that the tuple-set $\mathcal{T}^Q$ returned by a query Q over $\mathcal{R}$ *derives* from the tuple-set $\mathcal{T}^{Q'}$ returned by a query Q' if Q is obtained from Q' by adding one or more selections, one or more projections, one or more joins. We denote this by $Q \preceq Q'$, and, in turn, $\mathcal{T}^Q \preceq \mathcal{T}^{Q'}$.

In our example, the view AcmeUser $\bowtie$ Location derives from views AcmeUser and Location. Similarly, view $\sigma_{type='running'}$(AcmeUser $\bowtie$ Activity) derives from AcmeUser and Activity, but also from view $\sigma_{type='running'}$(Activity).

If we add, as a convention, a *top view* Q_{top} such that $Q \preceq Q_{top}$ for any query Q, it is possible to see that the relationship "derives from" among views over a schema $\mathcal{R}$ induces a join-semilattice.

Notice that the lattice also encodes a taxonomy: whenever the user imposes a restriction over the current view (concept), this amounts to somehow identifying one of its "sub-concepts". The lattice does not need to be pre-computed, and will be typically constructed in a dynamic fashion.

Given a tuple-set $\mathcal{T}$, and an attribute A in $\mathcal{T}$, we compute the distribution of values for A in $\mathcal{T}$, i.e., its $\mathcal{T}$-histogram.

To formalize this notion of relevance we notice that, whenever a query Q derives from Q', it is possible to keep track of the relationship among the attributes of $Q(I)$ and

those of $Q'(I)$.[3] To uniquely identify attributes within views, we denote attribute A in table R by the name R.A. We say that attribute R.A in $Q(I)$ *matches* attribute R.A of any $Q'(I)$ that is an ancestor or descendant of $Q(I)$ in the query lattice.

We can now formalize the notion of *relevance* for attributes in the view lattice. We say that attribute R.A of node $Q(I)$ is *relevant* if its distribution d is statistically different from the distribution of any matching attribute from any ancestor node. Note that the case of the root is a special one: we propose to assume the distribution of an attribute in the root node as the "default one" for that attribute; however the system administrator can modify this setting based on his/her knowledge of the domain of interest.

2.3 How the Conversation Goes On

We have now all the tools in place to present an example of a complex conversation. Assume as a first step the user is presented the items $\mathcal{S}_1$-$\mathcal{S}_3$ above, and selects item $\mathcal{S}_1$:

($\mathcal{S}_1$) "It might be interesting to explore the types of activities. In fact: *running* is the most frequent activity (over 50%), *cycling* the least frequent (less than 20%)".

This choice is interpreted by the system as an interaction with the lattice of views. The user has selected view Activity, and relevant feature type. The system shows a subset of values for the type attribute, the ones that justify its relevance. The user may pick one or more of these values: assume s/he selects value *running*. This is interpreted by the system as some interest in the set of tuples that correspond to running activities. As a consequence, a new view is generated and added to the lattice:

$$\sigma_{\mathsf{type}=\mathit{running}}(\mathsf{Activity})$$

The addition of a new node to the lattice triggers a new interaction, with the system trying to suggest new and relevant hints. To do this, it may add further nodes. Here, it finds that a relevant feature is related to the region of runners (users that perform running activities). Notice that this requires to compute a view:

$$\sigma_{\mathsf{type}=\mathit{running}}(\mathsf{Location} \bowtie \mathsf{AcmeUser} \bowtie \mathsf{Activity})$$

The system realizes that the distribution of the region attribute within this view is significantly different both from the uniform distribution, and from the ones of the matching attributes in ancestors. Assume, in fact, that users are evenly distributed across regions. On the contrary, runners are especially active in the west. Among other potentially relevant features, the system will suggest what follows:

($\mathcal{S}_{1.1}$) "It might be interesting to explore the region of users with *running* activities. In fact: while users are evenly distributed across regions, the 65% of users with *running* activities are in the *west*, only 15% in the *south*".

The user will select the suggestion ($\mathcal{S}_{1.1}$), and then pick up value *south*. The process is triggered again, and the system realizes that:

($\mathcal{S}_{1.1.1}$) "It might be interesting to explore the sex of users with *running* activities in the *south* region. In fact: while sex values are usually evenly distributed among users, only the 10% of users with *running* activities in the *south* are *women*".

[3] For the sake of simplicity, here we do not consider self-joins, i.e., joins of a table with itself. With a bit more work the definition can be extended to handle self-joins

It can be seen that the discovery of this new relevant feature requires to compute the following view and add the corresponding node to the lattice:

$$\sigma_{\mathsf{type}=\mathit{running},\mathsf{region}=\mathit{south},\mathsf{sex}=\mathit{female}}(\mathsf{Location} \bowtie \mathsf{AcmeUser} \bowtie \mathsf{Activity})$$

After s/he has selected the suggestion ($\mathcal{S}_{1.1.1}$), the user is effectively exploring the set of tuples corresponding to female runners in the south. S/he may decide to ask the system for further advice, or browse the actual database records, or use an externally computed distribution of values to look for relevant features. Assume, for example, s/he is interested in studying the quality of sleep. From an external Web site she downloads an Excel sheet stating that over 60% users complain about the quality of their sleep. When these data are imported into the system, the system computes a few more joins, and realizes that, unexpectedly, 85% of sleep periods of southern women that run are of good quality. Having learned new insight about the data, the user is satisfied.

From the technical viewpoint, the system has constructed a complex lattice of views. From the back-end perspective, the conversation consists in a walk of this lattice.

3 Research Challenges

The exploration model we have presented poses quite a number of technical challenges to database researchers. In this section, we overview some of these and highlight the literature that tackles some aspects. Eventually, we propose a technique that can advance us in the research path.

The most related research topic is *faceted search*, also called faceted navigation, which (often visually) supports the access to information items based on a taxonomy of *facets* [12] (roughly our *features*). In faceted search, items are initially classified according to a given taxonomy. The user can browse the set making use of the facets and of their values (e.g. the feature *Activity*, or the value *running*) and will inspect the set of items that satisfy the selection criteria. Faceted search has similar aims w.r.t. database exploration and would greatly benefit of the foundational model we propose, which can support all its complexity and scalability challenges.

Responsiveness To guarantee a satisfactory user experience we need a "reasonably fluent" conversation: users will not tolerate long computing times. This imposes strong constraints over the computation of the view lattice and related feature statistics. Let us first distinguish the case in which the database is a static snapshot from the one in which the database may receive updates. In the first case a portion of the view lattice and the related features may be pre-computed. In the second case, this might not be acceptable, at least in case of high update frequency.

Responsiveness is related to a research area that is traditional for databases: the DBMSs build and maintain *histograms* representing the distribution of attribute values to estimate the (possibly joint) selectivity of attribute values for use in query optimization [3]. Fast, incremental histogram computation is a good example of a technique that can be effectively employed for speeding up the assessment of relevance in a conversation step.

Summarization In both the above cases, fully pre-computing the lattice seems unreasonable, especially for truly big data. This imposes to study two different optimization directions.

On the one side, it suggests to work with data summaries, the size of which must be large enough to estimate statistical parameters and distributions, but practical from the computation viewpoint. To solve this problem many summarization techniques can be explored. Some works, for instance [1], have proposed to extract knowledge by means of data mining techniques, and to use it to intensionally describe sets in an approximate way. To summarize the contents of large relations and permit efficient retrieval of approximate query results, authors in [10] have used histograms, presenting a histogram algebra for efficiently executing queries on the histograms instead of on the data, and estimating the quality of the approximate answers. The latter research approaches can be used to support query response during the conversation.

On the other side, we need effective pruning techniques for the view lattice. A feature may be relevant if it is *different from*, or maybe *close to*, the user's expectations; in their turn, the users' expectations may derive from their previous background, common knowledge, previous exploration of other portions of the database, etc. In the following we suggest a technique for relevance assessment that allows to define notions of relevance more general than the one proposed by the motivating example.

CPUs vs GPUs Another open problem is if traditional CPU-RAM-disk architectures are fast enough for this purpose, or we need to resort to different ones. GPUs seem promising, provided that the memory limitations are handled, and avoid bottlenecks related to data transfers. Fast implementations of statistical computations over the GPU are a promising research direction [7] [9].

Fast Statistical Operators The previous points bring us to another fundamental requirement: the availability of fast operators to compute statistical properties of the data.

Surprisingly, despite years of data-mining research, there are very few research proposals towards the goal of fast computing statistical parameters, and comparing them. Subgroup discovery [6] [5], endeavors to discover the maximal subgroups of a set of objects that are statistically "most interesting" with respect to a property of interest. Subgroup discovery has been proposed in many flavors, differing as for the type of encompassed target variable, the description language, the adopted quality measure and the search strategy. Again, we believe that some of its techniques might be profitably adopted to assess the relevance of features in each of our exploration steps.

References

1. E. Baralis, P. Garza, E. Quintarelli, and L. Tanca. Answering XML queries by means of data summaries. *TODS*, 25(3), 2007.
2. N. D. Blas, M. Mazuran, P. Paolini, E. Quintarelli, and L. Tanca. Exploratory computing: a challenge for visual interaction. In *AVI*, pages 361–362, 2014.
3. P. B. Gibbons, Y. Matias, and V. Poosala. Fast incremental maintenance of approximate histograms. In *PVLDB*, pages 466–475, 1997.
4. P. Hanrahan. Analytic database technologies for a new kind of user: the data enthusiast. In *SIGMOD*, pages 577–578, 2012.
5. F. Herrera, C. J. Carmona, P. González, and M. J. del Jesús. An overview on subgroup discovery: foundations and applications. *Knowl. Inf. Syst.*, 29(3):495–525, 2011.
6. W. Klösgen. Explora: A multipattern and multistrategy discovery assistant. In *Advances in Knowledge Discovery and Data Mining*, pages 249–271. 1996.

7. U. Milic, I. Gelado, N. Puzovic, A. Ramirez, and M. Tomasevic. Parallelizing general histogram application for cuda architectures. In *Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIII)*, pages 11–18, 2013.
8. K. Morton, M. Balazinska, D. Grossman, and J. D. Mackinlay. Support the data enthusiast: Challenges for next-generation data-analysis systems. *PVLDB*, 7(6):453–456, 2014.
9. C. Nugteren, G. van den Braak, H. Corporaal, and B. Mesman. High performance predictable histogramming on gpus: Exploring and evaluating algorithm trade-offs. In *Proc. of the Fourth Workshop on GPGPU*, pages 1:1–1:8. ACM, 2011.
10. V. Poosala, V. Ganti, and Y. E. Ioannidis. Approximate query answering using histograms. *IEEE Data Eng. Bull.*, 22(4):5–14, 1999.
11. J. W. Tukey. *Exploratory data analysis*. Addison-Wesley, Reading,MA, 1977.
12. D. Tunkelang. *Faceted Search*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, 2009.