SeeDB: What's interesting about my query?

1. SYSTEM ARCHITECURE

1.1 SEEDB overview

Figure 1 shows the architecture of our system. Currently, SEEDB is a wrapper around a database (PostgreSQL in this case). While optimization opportunities are restricted by virtue of being outside the DBMS, we believe that it allows quick iteration and permits SEEDB to be used with different backends.

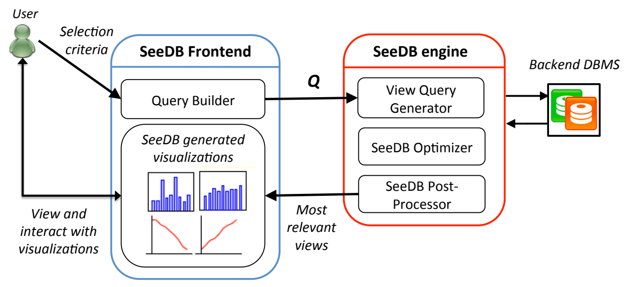

Figure 1: SeeDB Architecture

The SEEDB frontend allows the user to pose queries via various means including raw SQL, an easy-to-use query builder and a set of queries pre-written for specific tasks (more in **??**). Once the user issues a query Q to SEEDB the SEEDB optimizer determines potential views possible for Q and explores options to optimize the computation of these views (Section **??**). These options include traditional multi-query optimization techniques as well as pruning heuristics. The View Query Generator module of SEEDB then uses this information to rewrite Q as multiple queries where each query is either a view or a precursor to multiple views. These queries are sent to the DBMS and the results are processed by the Post-processor to generate data for individual views and determine utility for each view. The views are then ranked using a metric that depends both on the view utility and complexity of the view (we prefer views with high utility and low complexity). The top-k views with highest utility are then picked and shipped to the frontend. The post-processor also determines the best way to package view information to allow quick future interaction at the frontend. The SeeDB frontend uses view information sent by the SeeDB engine to pick appropriate visualizations and show the top views along with appropriate metadata. The user can then examine these views and perform further processing (Section **??**).

1.2 Basic Framework

Given a user query Q, the basic version of SEEDB computes all possible view queries by adding a single aggregate and a single or multiple attribute group-by operator to Q. Each of these view queries is executed independently at the backend along with an equivalent aggregate+group-by query on the complete underlying dataset. These two query results produce a “distribution” for the attribute that has been aggregated (Section **??**). These two distributions are compared using the chosen distance metric (Section **??**) and the top k views with the largest distance are chosen. These views are sent to the frontend where they are visualized and presented to the user (Section 1.4).

1.3 Optimizations

Since SEEDB executes many similar queries, we clearly can do better than the basic algorithm presented above, and SEEDB implements various such optimizations. Strategies include:

1. *Rewrite view query*: Since similar group-by and aggregate queries are executed on the results of user query Q and the underlying dataset, we can combine these queries into one query. As shown in Figure **??**.a this achieves a speed-up of Y%.

2. *Single Group-by Multiple Aggregates*: A large number of view queries have the same group-by clause but aggregates on different attributes. A straightforward optimization combines all view queries with the same group-by clause into a single view query. This rewriting provides a speed up linear in the number of aggregate attributes. (Figure **??**. b).

3. *Multiple Aggregate Computation*: Similar to data cubes ([]), SEEDB seeks to compute group-bys for a large set of attributes. One optimization is to combine queries with different group-by attributes into a single query with multiple group-bys. For instance, consider view queries $V(Q, A_1, G_1)$, $V(Q, A_1, G_2) \ldots V(Q, A_1, G_n)$. Instead of executing them individually, we can rewrite them into a single view query $V(Q, A_1, (G_1, G_2 \ldots G_n))$. While this strategy reduces query execution time, SEEDB must spend more time combining the results and obtaining separate aggregates for individual G_i's. This is reminiscent of data cube algorithms. As shown in Figure **??**.c the speed up depends closely on the number

of distinct values for each of the group-by attributes and the memory constraints of the DBMS. A variation of this approach also implemented on SEEDB is to send the results of the multiple group-by query to the front end and ask the SEEDB frontend to compute utility and select views. The advantage of this approach is that it allows for more efficient interactive exploration of the views.

4. *Sampling*: The final optimization we study for the purpose of this demo is sampling. Instead of running queries on the entire dataset, we run queries on subsets of the data at the expense of reduced accuracy. Figure **??**.d shows the effects of this optimization. As expected, the sampling technique and size of the sample can affect the accuracy of the generated views. As a result, the user can specify whether inaccuracies are acceptable and therefore, whether SEEDB can use sampling.

   Although other optimizations are possible, particularly related to pre-computing aggregate results, we discuss them in the full paper currently in preparation.

1.4 User Interface

«figure goes here after I get it from Angela/Sashko»

The SEEDB frontend serves two purposes - to input a query, and to visualize and analyze the resulting views. We provide the user with three options for specifying an input query: (a) as raw SQL, (b) through a query builder that can allow users unfamiliar with SQL to formulate queries through an easy-to-use form-based interface, (c) through pre-defined query templates, e.g., queries that select outliers in a particular column. These templates are particularly useful since users are interested in anomalous data points.

Once a user submits a query to SEEDB the SEEDB engine evaluates various views and sends the most promising ones to the frontend. The frontend then determines the best ways to visualize these views (e.g. depending on data types being represented, number of distinct values etc.) and displays the visualizations. The user can then examine these diverse views at a glance, explore specific views in detail and view metadata for each view (e.g. size of result, sample data, value with maximum change, statistics etc.). The user can also slice-and-dice views further by selecting particular values of grouping attributes to explore. The user does this simply by selecting the relevant attribute values in the view. This automatically modifies the selection query and displays views for the subset of data selected. The user can of course revert back to the original views and continue exploring the data.

2. DEMO WALKTHROUGH

We propose to demonstrate the following aspects of SEEDB as part our scenarios: (i) the utility and versatility of SEEDB, (ii) the latency of SEEDB operating on various dataset sizes, along with the relative impact of optimizations within SEEDB.

Demonstrating Utility and Versatility: We will allow conference attendees to interact with three diverse datasets, capturing three potential ways SEEDB may be used. Attendees will be both (a) able to pose queries, view results, and interact further with these datasets in real time, and (b) be able to browse a selection of queries that we have pre-selected to be those that give unexpected or surprising results. The datasets that we will allow the conference attendees to browse include the following:

- **Store Orders dataset** [**?**]: This dataset is often used by Tableau [**?**] as a canonical dataset for business intelligence applications. It consists of information about orders placed in a store including products, prices, ship dates, geographical information, profits, and so on. This dataset is well-studied by users learning how to use Tableau, with several web-pages dedicated to discovering interesting trends hidden in this dataset [**?**]. Attendees using SEEDB will be able to identify very quickly the same insights and trends that Tableau users have discovered over many years. This dataset will also enable us to demonstrate how SEEDBcan correctly deal with numeric, categorical, time series and geographic data.
- **Election Contribution dataset** []: This dataset is an example of a real-world dataset that is typically analyzed by non-expert data analysts, such as journalists or historians. This dataset will enable us to demonstrate to the attendees how non-experts can quickly arrive at interesting visualizations via the intuitive user interface.
- **Medical dataset []:** This dataset is an example of a real-world dataset that a researcher (here, a clinical researcher) might use over the course of his/her work. This data has a schema that is more complex than the the election or store one, and is of larger size too.

Aditya: If this stuff is already in the user interface, I would delete it. No point repeating the workflow For all of these datasets, the interaction workflow will be the following: the conference attendee can either opt for a pre-selected query, or formulate a query of their own using the query builder tool or using SQL. Then, SEEDB will search through the entire space of possible visualizations using techniques described in Section **??**, and returns the top-k (attendees will be allowed to set this value when forming the query, otherwise the default is set as 10.). For the purposes of the demonstration, we will also allow attendees to view the bottom-k visualizations, enabling them to see why SEEDB actually displayed the chosen set of visualizations over other ones.

Demonstrating Speed and Optimizations: This demonstration scenario will use an enhanced user interface, with three additional "knobs" or "dials" that attendees can fix, described below. For this demonstration scenario, we will focus on the Medical dataset, which is a large dataset with a large schema.

- Fraction of dataset: The first knob allows attendees to change the fraction of the medical dataset selected. Clearly, as the dataset size increases, the latency of SEEDB increases, so attendees will be able to see how much of an impact the size of the dataset has on the latency of SEEDB.
- Fraction of schema: The second knob allows attendees to change the number of columns in the medical dataset selected. As before, as the number of columns increases, the latency of SEEDB increases, so attendees will be able to see how much of an impact the number of columns has on the latency of SEEDB.
- Optimizations selected: The last knob allows attendees to change the optimizations (from Section **??**) that are being used within SEEDB, allowing attendees to see which optimizations actually have an impact.

This demonstration scenario, in addition to having a different user interface, will also have a different result interface, with additional statistics reported when the visualizations are generated, including (but not limited to):

- Time to completion
- Time to first visualization
- Aditya: more here...

3. REFERENCES