Governor: Turning Open Government Data Portals into Interactive Databases

Chang Liu
c.liu@uwaterloo.ca
University of Waterloo
Waterloo, Ontario, Canada

Arif Usta
arif.usta@uwaterloo.ca
University of Waterloo
Waterloo, Ontario, Canada

Jian Zhao
jianzhao@uwaterloo.ca
University of Waterloo
Waterloo, Ontario, Canada

Semih Salihoğlu
semih.salihoglu@uwaterloo.ca
University of Waterloo
Waterloo, Ontario, Canada

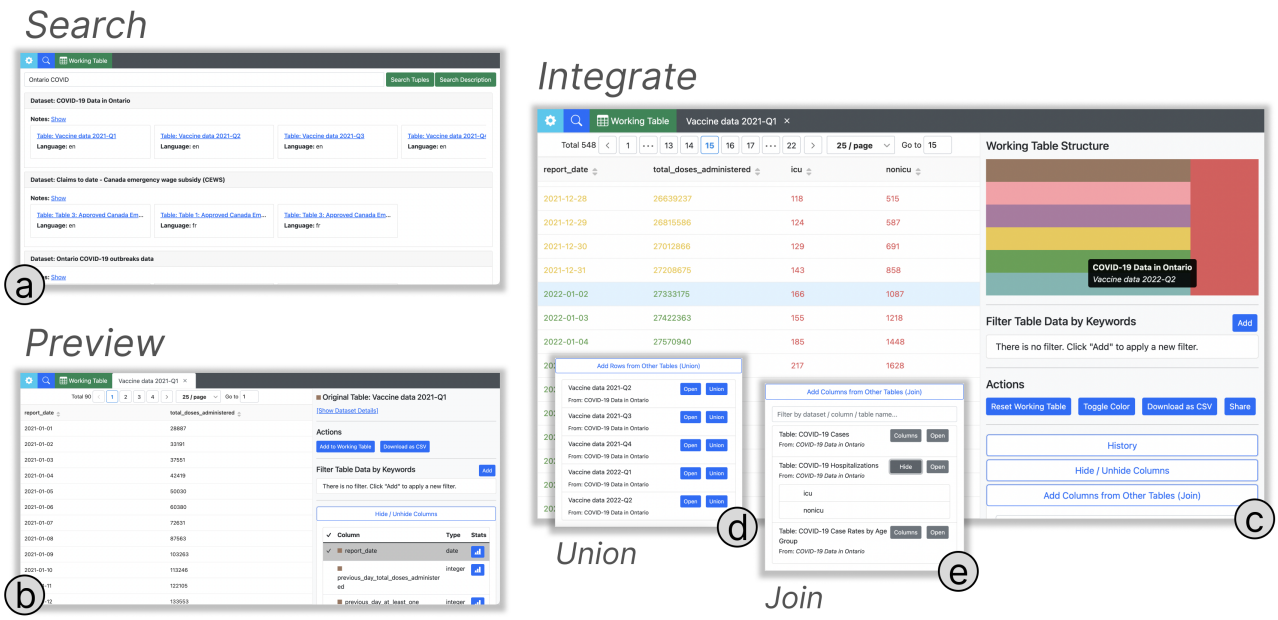

Figure 1: Governor: an interactive system to help users effectively search, preview, and integrate tables from OGDPs. The main interface of Governor consists of: (a) Search view which allows findind open data tables by the description of the dataset or the values stored in the table; (b) Original Table view which provides a preview of the original open data tables; (c,d,e) Working Table view which facilitates integrating multiple tables through automatically detecting unionable and joinable tables.

ABSTRACT

The launch of open governmental data portals (OGDPs) has popularized the open data movement of last decade. Although the amount of data in OGDPs is increasing, their functionalities are limited to finding datasets with titles/descriptions and downloading the actual files. This hinders the end users, especially those without technical skills, to find the open data tables and make use of them. We present Governor, an open-sourced[17] web application developed to make OGDPs more accessible to end users by facilitating searching actual records in the tables, previewing them directly without downloading, and suggesting joinable and unionable tables to users based on their latest working tables. Governor also manages the provenance of integrated tables allowing users and their collaborators to easily trace back to the original tables in OGDP. We evaluate Governor with a two-part user study and the results demonstrate its value and effectiveness in finding and integrating tables in OGDP.

CHI '23, April 23–28, 2023, Hamburg, Germany

CCS CONCEPTS

• **Information systems** → **Information integration**; *Search interfaces*; • **Human-centered computing** → **Interactive systems and tools**.

KEYWORDS

open data, data integration, database, user interface

ACM Reference Format:
Chang Liu, Arif Usta, Jian Zhao, and Semih Salihoğlu. 2023. Governor: Turning Open Government Data Portals into Interactive Databases. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI '23), April 23–28, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3544548.3580868

1 INTRODUCTION

The launch of open governmental data portals (OGDP), such as data.gov, open.canada.ca, data.gov.in, data.gov.uk has popularized the open data movement of the last decade. These portals publish large numbers of datasets about many aspects of governments and the countries these governments govern. Example topics include how governments distribute research funds, how much meat the country exports, or the CO_2 emissions of different factories in a country. The core goal of opening datasets to the public is to make governments more transparent and fulfill the information needs of different members of the society [10]. In their usage vision, these datasets can be analyzed by policy analysts, journalists, researchers, and engaged citizens to find rooms for improvement in how governments operate and identify corruption and systematic injustices in a country. In light of this vision, the potential societal value of making government datasets more open and easier to find, understand, and analyze for the users can be invaluable. However, as identified in prior studies on datasets and open data users [18, 25, 35, 39, 45], existing OGDPs fall short on delivering their potential value to their intended users due to several shortcomings on these portals that do not address some complex user needs. We next discuss several of these shortcomings that motivate our work.

Existing OGDPs, including dataset search engines, such as Google Dataset Search [21], have search interface that only supports metadata search which heavily relies on the accuracy and availability of metadata information of the datasets. Yet, low metadata quality negatively impacts users' dataset discovery to find and consume datasets [39]. Furthermore, there have been user studies [18, 35, 45] where OGDP users share their frustration of not being able to find desired datasets and indicate the need for improvement of search capabilities. In order to alleviate search ineffectiveness in OGDPs, one promising solution is to provide *tuple/record search* to complement metadata search. Many questions that are of interest to users can be answered through a few records. Consider the questions: "How many refugees from Afghanistan, Syria, Ukraine, etc. did Canada accept in 2022?" The answer to these questions can be found in a few tables with cells containing the country names but a search of keywords such as "refugees" in open.canada.ca returns more than 50 tables based on metadata, most of which are unrelated.

Furthermore, as reported by open data users in [35] finding datasets also involves the step of *data sensemaking*, which includes exploring dataset contents, e.g., column headers, descriptions and types of columns, and first few rows. Existing OGDPs primarily fulfill the functionality of publishing datasets in some raw file format, not offering features to make it easier for users to explore and understand the dataset. In dataset search setup, users are more interested in extending the current table they have in their hand rather than finding a specific one [25]. Morever, many user tasks often span over multiple datasets, which requires *data integration* features, also reported as a recommendation in a related user study [35]. However, in existing OGDPs, this can be accomplished through a tedious and manual task of finding related tables, downloading them, and putting them together in a spreadsheet software.

Finally, the prevalence of need for integrating datasets makes *provenance management* critical, as users are also interested in the history and processes of data collection [35]. Provenance is also vital in verifying an analysis based on a dataset [41]. In a recent user survey [35], the feature of dataset sharing in open data portals is identified as important, for which provenance can be instrumental, as provenance helps users assess trustworthiness of data. However, currently, once datasets are downloaded, users have to manually keep track of provenance information, which is challenging if not impossible especially for tasks involving many datasets.

In this paper, we argue that the above challenges would be addressed by the core functionalities of database management systems (DBMSs). For example, one can ask queries in DBMSs to search through records. Similarly, tables that integrate multiple other tables can be modeled as *views* [29] over the base tables. DBMSs also store information about how the views have been constructed, which can serve as provenance information. Indeed several systems in literature [42, 50] provide a subset of these features by storing the tables in data "lakes" in a DBMS and making this data accessible through a programming or query language, such as SQL. However, such systems are not accessible to many potential users of OGDPs who are not programmers. While several interactive data wrangling systems support non-programmers for transforming and cleaning data, such as Wrangler [34] and Rigel [26], they assume that users have already found the relevant data tables and integrated them. However, it is currently challenging for users to arrive at this step without laborious operations.

Hence, we argue that for OGDPs to deliver their potential value, they should evolve from mere publishing websites into interactive visual systems with database capabilities that are familiar to non-programmers. To achieve this overarching goal, we propose *Governor* (Figure 1), a browser-based system to address the above drawbacks of OGDPs. Governor models an OGDP as a database of published tables, which can be accessed and integrated interactively. Governor is designed for non-programmer users with the basic skills for exploring and integrating data tables, thus making it convenient to later wrangle and analyze the data through other tools such as Excel, Tableau, and Wrangler [34].

Governor uses the open.canada.ca portal as a test-bed and indexes all of the records in its tabular corpus. First, it allows users to search original records about specific entities, such as "Afghanistan", and preview the result tables/records in a single interface without downloading them and using a separate tool. Second, starting from one of the original tables, users can start data integration sessions. By interactively performing two core relational operations, unions and joins of tables, through a few clicks, users can construct a *Working Table* that integrates multiple tables. These data integration capabilities can be very useful when putting together frequently and periodically published tables, such as daily COVID-19 case tables. The data integration sessions are generally guided through the system's suggestions for tables to union and join with the Working Table. Lastly, Governor summarizes the integration task through a simple color-guided provenance summary, *Working Table Structure (WTS)*, which visually presents the tables that have been integrated

and how (see the top-right panel in Figure 1c). For fact-checking and verification, users can also view the original tables that have been integrated or go to the cells in the original table that correspond to the cells in the Working Table through a few clicks.

To assess Governor, we conducted a two-part user study, where in Part 1 we employed three definitive tasks to test the system with 12 participants and in Part 2 we devised an open-ended task to evaluate the system with additional 6 participants. This study helps evaluate the system in both controlled and natural settings, providing different perspectives to user experience with Governor.

In summary, our contributions in this paper include:

- The design and development of an in-browser interactive system, Governor, to assist non-programmers with finding and integrating data tables in OGDPs;
- Empirical findings of users' experiences and behaviors on completing various tasks with the system on real-world datasets.

2 RELATED WORK

In this section, we review prior work on four areas: (i) open data search tools; (ii) interactive data integration systems; (iii) systems and algorithms for finding related tables; and (iv) tools and visualizations for data transformation or cleaning. Table 1 shows the comparison between the closest prior systems and Governor.

2.1 Open Data Search Tools

Several systems support searching through large collections of enterprise or open data lakes. These systems support search through two search modes: (i) keyword queries, or (ii) *table queries*, in which users search for tables related to a query table.

Google Dataset Search (GDS) [21] is a large-scale search engine that indexes both public and private datasets. GDS crawls webpages that contain special HTML tags indicating that the page contains a dataset, and indexes the metadata about the dataset, which includes their descriptions, publisher information, or data licences. Unlike Governor, GDS is solely a search engine and does not index the contents of the records, nor supports data integration.

RONIN [40] is a data lake exploration system that supports both types of search modes we mentioned above. Users issue a keyword query, then get back a list of tables, and then navigate the data lake starting from a result table, which can be used as a table query to further get a set of related/joinable tables. Internally RONIN uses several indexing techniques, e.g,. the faiss index [32], to search for related/joinable tables. However, RONIN is not designed to integrate tables and provide provenance capabilities.

Auctus [23] is similar to RONIN and supports keyword and table queries. The latter are issued by manually uploading a table into the system. Auctus can also search the contents of temporal and spatial columns. Auctus also supports limited data integration, where the table can be joined or unioned with only one table at a time. To integrate multiple tables, users have to download and re-upload the intermediate data tables multiple times, which is tedious.

Toronto Open Data Search (TODS) [54] supports joining multiple tables; however, TODS does not support unions, another key database operation, which results in limitations for integrating data. Governor instead supports integrating large numbers of tables with both join and union operations, and is designed to manage these integrated tables and their provenance.

2.2 Data Integration Systems

Several researchers have developed interactive systems that have capabilities for data integration in data lakes. Voyager [20] is a data search and discovery system. Voyager is an extension to Jupyter Notebook. As such, Voyager is accessed through a programming language, such as Python. Voyager supports both keyword and table queries. Although not targeting data integration tasks, the results of these searches are tables that can be integrated with other tables in the user's programming language. Voyager assumes an interactive user experience, albeit the interaction happens through a programmer writing snippets of programs, instead of Governor's clicking-based web interface.

Juneau [50] is also a Jupyter Notebook extension. We refer to these notebooks as Juneau notebooks. Users can perform table queries to search source data lakes, and the results can be integrated with other tables in Juneau notebooks. However, the primary goal of Juneau is to manage data science pipelines. To achieve this, the Juneau notebooks, which contain both data integration and analytical codes are themselves stored in Juneau's server. The computation graphs of how the tables in Juneau notebooks were constructed (i.e., their provenance) is used to suggest related tables to users. Instead, Governor uses provenance information to facilitate fast fact-checking in the original tables published in OGDPs.

KGLac [31] is another system that supports table queries. Similar to Voyager and Juneau, KGLac targets programmers. KGLac generates a knowledge graph out of a data lake, whose nodes contain tables and columns and edges represent different relationships, such as how similar different columns are. This knowledge graph is stored in an Resource Description Framework (RDF) database [46], which can be queried through the SPARQL query language. In addition, users can use the KGLac library in Python to perform table queries and find related tables, which can be integrated in Python.

DICE [42] is a system in which users provide example tuples of a desired table to the system. DICE then finds SQL queries that could have generated those tuples. Although data integration is not the focus of DICE, these SQL queries can perform automatic data integration to generate a table that conforms to users' examples.

2.3 Techniques for Finding Related Tables

Effective data integration requires algorithms for suggesting related tables in data lakes. Prior research has studied the foundations of finding unionable and joinable tables with respect to a current table of interest. The literature assumes a setting in which a system supports table queries. Given a query column, Zhu et al. [51, 53] proposed efficient LSH indexes to find other columns that have high value overlap with the query column. Nargesian et al. [38] proposed a suite of indexing, searching, and ranking techniques for the problem of finding unionable tables. Some of these techniques are based on the values in the columns, some are based on ontology mappings of the columns of the tables (if they exist), and others are based on semantic similarity [19, 37]. KTabulator [47] uses information in knowledge graphs to recommend columns and rows for creating and extending ad-hoc tables from Wikipedia. These

Table 1: A comparison between features offered by Governor and closest related work. ✓denotes full support; ~ denotes partial support; × denotes not available.

	GDS [21]	RONIN [40]	Auctus [23]	TODS [54]	Voyager [20]	KGLac [31]	Governor
Metadata Search	✓	✓	✓	✓	✓	✓	✓
Tuple Search	×	×	~	×	✓	×	✓
Data Preview	×	✓	~	✓	✓	×	✓
Related Tables Suggestion	×	✓	✓	✓	✓	✓	✓
Data Integration	×	×	~	~	×	×	✓
Provenance Management	×	×	×	~	×	×	✓
Interactive User Interface	✓	✓	✓	✓	~	×	✓

techniques are complementary to our work. Governor uses a set overlap-based technique for finding joinable tables and searches for pairs of tables with the same schema to suggest unionable tables (see Section 5.1). Our design principle here was simplicity but our algorithms can be extended by other techniques from literature.

2.4 Data Visualization and Wrangling Tools

Data transformation and cleaning (a.k.a. data wrangling) is another key step in data analytics pipelines in practice. Wrangler [34] is an interactive tool that combines direct manipulation of visualized data with automatic inference of relevant data transformations, allowing users to iteratively explore a set of wrangling operations and preview their results. Rigel [26] allows for rapidly transforming tabular data by formulating data wrangling operations as direct mappings from data to row, column, and cell channels of the target table. Other tools have been proposed with the concept of "programming by example" such as Wrex [28] to facilitate non-programmers with data transformation tasks. These tools assume that users have already possessed relevant data tables and integrated them, which is currently challenging in existing OGDPs. Instead, Governor focuses on the prerequisite steps to search and integrate tables.

Our premise of facilitating fact-checking and auditing through provenance management in OGDPs has been articulated in other data science settings. For example, researchers have noted the importance of capturing provenance when teams of analysts share data and analytics [27, 33, 41]. Several systems have been proposed for visualizing and analyzing data wrangling pipelines. VisTrails [22] is an early scientific workflow and provenance management system for data simulation, visualization, and exploration. Heer et al. [30] studied the design of history mechanisms for visual data exploration within Tableau [13]. Tools have also been proposed for visualizing data science scripts such as Somnus [48] and Unravel [43]. A recent survey on the provenance analysis of user interaction and data visualization can be found in [49]. These systems assume more general data transformation capabilities and therefore propose more advanced solutions for provenance visualization than Governor. Instead, Governor's data transformation capabilities are limited to joining and unioning tables. As such, we designed Governor's color-guided provenance summary, the Working Table Structure, to intuitively present these operations.

3 USAGE SCENARIO

Shufan is a journalist who has been tasked with a specific question: Are the vaccinations reducing COVID-19 cases in Ontario?

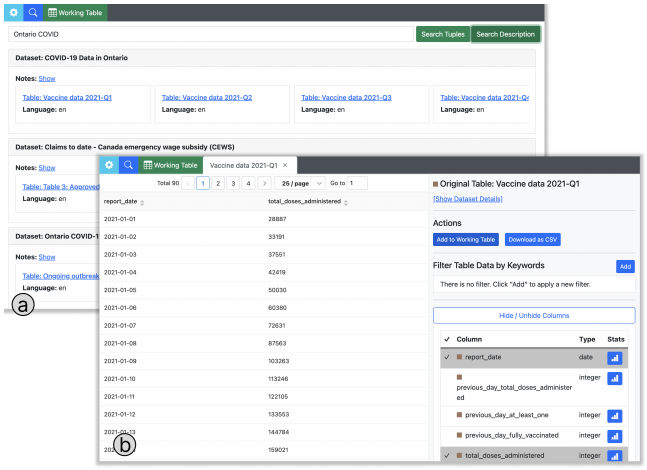

Figure 2: (a) Search results showing relevant tables grouped under their datasets given the query and (b) original table view when a particular table is clicked by the user.

Shufan decides to answer this question by correlating 1.5 years of COVID-19 hospitalization cases with administered vaccination doses. Knowing that this information is periodically published by the government, he decides to construct a dataset using Governor.

He types "Ontario COVID" on Governor's landing page and clicks "Search Description" (Figure 2a), which returns back, among other results, a dataset of COVID-19 Data in Ontario with many tables in it, since this information is published quarterly. He clicks on the Vaccine data 2021-Q1 table, which opens the table in Original Table View (Figure 2b). He notices 90 tuples in this table, one for each day of the first quarter of 2021, with a total_doses_administered column storing the number of vaccines administered in that day.

Shufan decides to enrich this table with data from other quarters and also hospitalization cases. He clicks on the "Add to Working Table" button (Figure 2b) on the right panel which copies this data to the "Working Table" view. Next, he inspects the "Add Rows From Other Tables (Union)" sub-panel in the Actions Panel, where Governor lists its suggestions of other tables (Figure 3a) that have the same schema. These are the tables in open.canada.ca that can be union with the Working Table. There he sees multiple suggestions for each quarter. He clicks on all the tables of vaccine data from Q2-2021 to Q2-2022 and generates a table (Figure 3b) with hundreds of

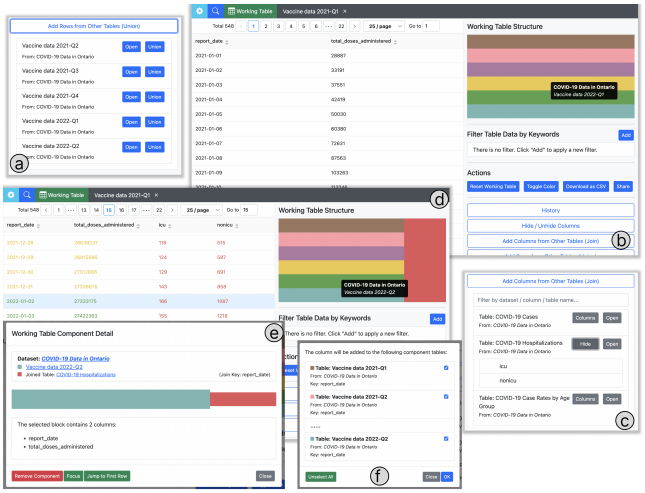

Figure 3: (a) The panel for suggested unionable tables given the working table, (b) Working table along with WTS on top-right of the view after performing unions, (c) The panel for suggested joinable tables and columns given the updated working table, (d) Working table with color-coded rows, (e) Working table component view showing the provenance of the original tables constituting the bottom-most component. (f) Join confirmation pop-up indicating which of the previous unioned tables' rows can be populated with the columns that the user selects.

tuples and two columns; report_date, and total_doses_administered (hiding several other columns).

Next, Shufan decides to integrate hospitalization data into the Working Table. Since these data are in separate files with different schema, Shufan inspects the "Add Columns From Other Tables (Join)" panel and sees several tables there (Figure 3c), including "COVID-19 Hospitalizations". The tables here all have close to perfect overlaps with a key (or almost key) column in the Working Table, i.e., that is unique for each tuple in the table. In Shufan's current table, this is the report_date column. Shufan inspects the columns that he can gather from COVID-19 Hospitalizations and infers that icu and nonicu columns must be reporting the number of hospitalizations that were (icu) and were not (nonicu) in intensive care units. He clicks on these columns, upon which a pop up menu indicates which of the 6 previous unioned tables' rows can be populated with these columns (Figure 3f). Shufan sees that each of the 6 unioned tables can be populated and clicks OK, which extends the Working Table with two new columns (and no new rows) (Figure 3d). This operation effectively joins each of the previously unioned 6 tables with COVID-19 Hospitalizations.

Shufan next decides to do his analyses. He downloads the integrated table as a csv file and opens it in Excel. He plots a line chart of total_doses_administered and icu and sees a sharp decline starting in the second quarter of 2022, though without a clear effect on hospitalization. He thinks that it would still be worth raising the question of what has caused this decline in an article. He prepares his article that contains the chart and sends it with a shareable Governor link to the editors for fact-checking.

Upon opening the link, the editor sees the integrated table (Figure 3d) and inspects the Working Table Structure (WTS) panel (at the top-right corner of Figure 3d) to understand the tables that has been integrated. She sees that many tables have been unioned thanks to color coding, and after inspecting the names of these tables, understands that they correspond to quarterly vaccination data. She knows that Shufan's article mentions a decline in vaccinations in the second quarter of 2022, so clicks on the rectangle for Q2-2022 on WTS, which opens the "Working Table Component Detail" pop-up (Figure 3e). She then clicks on the link to Vaccine data 2022-Q2 and opens it in the Original Table View. Here she eyeballs the total_doses_administered column of the 92 tuples in this table and notices that they are in the few thousands, which indeed looks very low. She double checks the line chart in the article, and sees that it plots numbers in the few thousands for the second quarter of 2022. She then clicks on the Show Dataset Details link (at the top-right of Figure 2f) and reads the dataset details to verify that these are COVID 19 vaccination data published by the ministry of health. She fact-checks the data in a few other quarters and lets Shufan know that she has verified the article's data.

4 DESIGN GOALS

Governor's design was guided by three goals driven by our approach of providing DBMS capabilities to OGDPs through a non-programming interface to address the drawbacks of existing OGDPs we discussed in the introductory section.

G1: Improve dataset search: Finding the desired datasets is a 2-step process [35] that involves search and data sensemaking. In previous user studies [35, 45] done with open data users, it was reported that users often find it challenging to locate what they are looking for. An ideal data discovery tool should be designed to provide effective features to enhance user experience for both these processes. Existing OGDPs provide search functionality over many attributes stored under metadata of the dataset; including title, description, publisher name. However, this is not sufficient for searching particular entities such as persons, universities, or companies. Values identifying such entities often are not present in dataset descriptions. Hence, our first design goal was to support searching over records to complement metadata search. After search, users spend time studying the contents of the dataset, which has to be done through third party applications outside the portal. We further wanted to implement features to ease data sensemaking, including previewing the data without downloading, listing column names along with descriptions (if available), and performing basic spreadsheet functionalities such as filtering.

G2: Support joining and unioning tables through a user-friendly interface: In contrast to general web search engines, in a typical dataset search session in OGDPs users tend to grow what they find by adding external information rather than trying to find new relevant information [25]. The results of a recent user study [35] with open data users affirms that many user tasks require access to multiple datasets. Tools to discover relationships between datasets to link and ideally integrate them are highly recommended as additional features for OGDPs. However, existing OGDPs do

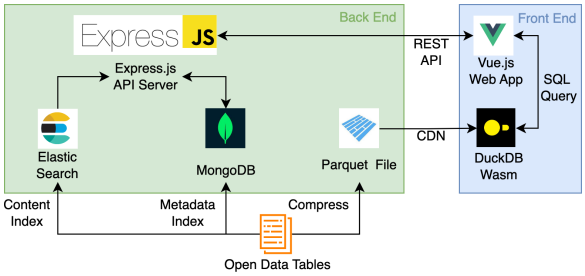

Figure 4: Governor System Architecture.

not have features to find links between related datasets, let alone integrate them. The publication style in OGDPs also have two properties that increase the importance of detecting and integrating related datasets: (i) periodic publishing: tables being published at certain time intervals, e.g., every month, week; and (ii) normalization [44]: some information is published in a normalized form, i.e., partitioned into multiple tables to avoid data redundancy. Manually finding related tables to integrate one by one is not practical, which is what users have to go through using existing OGDPs. Therefore, our second design goal is to have a functionality that supports automatically detecting and integrating related tables through a user-friendly interface. Particularly, we design table integration as a combination of join and union operations, where join is handy to detect related tables that share common columns while union is effective to capture related tables that are periodically published.

G3: Support exploring provenance of the information in integrated tables: As suggested in a recent user survey [35], an important use case for OGDP datasets is to be able share data which may include information from multiple tables to other potentially interested users. For instance, a journalist publishing an article based on data she gathered through an OGDP would also want to affiliate provenance of the data to her article to increase trustworthiness. The outcomes of the analyses based on the information gathered by integrating tables from OGDPs can be sensitive and require fact-checking, for which provenance is reported to be instrumental to ensure credibility [41]. Hence, our third design goal is to provide a functionality to users so that they can track provenance of the data gathered by integrating tables.

5 GOVERNOR SYSTEM

In this section, we give a detailed description ofGovernor's implementation. Many of the existing OGDPs use the CKAN system to host their portals. For example, each of the largest OGDPs that publish in English, open.canada.ca (Canada), data.gov (US), data.gov.uk (UK) use the CKAN system. Although we have used open.canada.ca in our study, Governor can work with any CKAN-based portals. We discuss how to run Governor on a different portal in Section 5.3 and provide documentation in our GitHub repository [17]. As a background, we note that CKAN-based OGDPs are comprised of a set of datasets, each of which contains multiple data files, e.g., csv files, and a metadata file providing descriptions of the data files. Governor indexes only the tabular files.

Figure 4 shows an overview of Governor's architecture. The front end of Governor is a single-page Vue.js [15] web application. In the front end, we use the DuckDB-Wasm system, which runs in the browser and handles the data transformation operations in the Working Table or the original tables in the Original Table View (e.g., sorting or filtering). The back end of Governor combines multiple technologies and provides a unified interface for the front end via a HTTP server. The back end fetches the open datasets from the government data portal through CKAN[3] API, indexes the dataset with MongoDB [9] and Elasticsearch[7] for querying, and serves the data files by converting them into a compressed Apache Parquet [2] format using Apache Arrow [1], as illustrated in Figure 4.

5.1 Core Functionalities

Governor's core functionalities aim to support non-programmers to interactively search, explore, and integrate datasets on OGDPs.

5.1.1 Searching Datasets (G1): We implemented two different search modes in Governor: *Search Tuples* and *Search Description*. Users can perform a search by typing a keyword and select one of the search modes. The search results, which are a set of tables, are grouped by dataset and contain the notes, subjects, and release date of each dataset, which can be seen in Figure 2a.

In the back end, we use MongoDB to index the metadata description of each dataset and the tables under it as one document (~30K in total). We use Elasticsearch to implement tuple search. We index each tuple t of each table T in open.canada.ca as one document (~323M in total). The document contains the table ID of T and all of the values in t's column as a single array value. An alternative design can store each column value separately but for tables with many columns, this design runs into Elasticsearch's limitations on number of different document fields. When users ask a tuple search query, we first retrieve Elasticsearch results, then go to MongoDB to retrieve the metadata information about table IDs and the datasets these tables are from.

5.1.2 Table Integration (G2): Governor supports gathering more data into the Working Table (WT henceforth) through the two core data integration operations of join and union. We first describe the conceptual components of the WT and the user interactions that Governor supports. Then we discuss how the system detects and suggests joinable and unionable tables. In Appendix A discuss our choice of using DuckDB-Wasm to achieve interactive speeds when performing these operations.

Conceptual Components of WT: We divide the tables in WT into 2 conceptual classes:

- *Unioned tables* contain the original table the user populated WT with and any other table that the user unioned this table with to add new rows to WT. Unioned tables have exactly the same schema.
- *Joined tables* are those that are joined with one or more of the unioned tables to add new colums to the WT.

At any point in time any subset of the columns of unioned and joined tables are visible in WT. We refer to these as the *unhidden unioned/joined table columns*. Users can perform the following data integration operations.

- Joins are performed through the "Add Columns from Other Tables (Join)" panel that presents to the user a list of of suggested tables $\mathcal{SJ} = \{J_1, ..., J_k\}$ and the columns of these tables that can be

integrated to WT through joining (Figure 3c). These tables do not share the same schema as the unioned tables and is joinable with at least one unioned table U_i on join columns $U_i.k_i = J.k_j$, where $J.k_j \neq J.c$. We discuss how Governor populates its joinable table suggestions momentarily. When a user clicks on a column $J.c$ from a suggested table J, the user is provided with the list of unioned tables $\mathcal{U}_J = \{U_{J1}, ..., U_{Jt}\} \subseteq \mathcal{U}$, where $\mathcal{U}$ is the set of previously unioned tables that J can be joined with. For example, in our usage scenario, our journalist Shufan was presented with 6 possible unioned tables (one for each quarter of Vaccine data from Q2-2021 to Q2-2022) after clicking on the icu and nonicu columns of the suggested Hospitalization table (Figure 3f). The user clicks on any subset $\mathcal{U}'_{\mathcal{J}} \subseteq \mathcal{U}_J$ and clicks OK. This joins (on the join columns) each of the unioned tables in $\mathcal{U}'_{\mathcal{J}}$ with J and fills the new c column for each row r of these tables with values from J (based on J's row(s) that r joins with).

- Union a new table U_{k+1} (assuming currently there are k unioned tables) to WT from the suggested list of unionable tables under the "Add Rows From Other Tables (Union)" panel (see Figure 3a for an example). We discuss how Governor populates its unionable table suggestions momentarily. This will add the rows $t_1, ..., t_m$ of U_{k+1} to WT and display the values of the unhidden unioned table columns of these rows. If the $WT.c$ is column that was integrated into the table from a joined table, i.e., one of the previous unioned tables was joined with a table J to add c, the system will put the value "Unfilled" for the c column of $t_1, ..., t_m$.

Detecting and Suggesting Unionable and Joinable Tables: For unionability, we opted for a schema-based approach, where we required the schemas of unionable tables (i.e., sets of column names) to be the same with the unioned tables in WT. Our goal here was to choose a simple definition of unionability that can detect periodically published tables which overwhelmingly have same schemas. This is the most common case of unionable tables that we observed across open data portals. More advanced definitions of unionability have been proposed [19, 37, 38], such as when subsets of columns have high content similarity. We leave the robust implementations of these techniques to future work. In our implementation, we preprocess the entire corpus to find pairs of tables that have the same schema and put these pairs into MongoDB. During a user session, the "Add Rows from Other Tables (Union)" panel is populated by querying MongoDB with the first unioned table.

We also preprocess the corpus to find joinable table-column pairs. We use the containment score (a.k.a *overlap set similarity metric*) to compute a joinability score between table-column pairs. This is a frequently used joinability metric in the literature [24, 40, 51–53]. The containment score is defined as $\frac{|Q.c \cap c_i|}{|Q.c|}$ where Q_c is the query column that the system needs to find joinable columns for, and $c_i \in C$ are the set of other candidate columns from a corpus of columns C. This containment score ranges from 0 to 1, where 1 has perfect containment. We store table-column pairs $T_1.c_1$ and $T_2.c_2$ that satisfy the following three constraints in MongoDB; (i) the containment score of $T_1.c_1$ and $T_2.c_2$ is high (>0.7 by default), to ensure the join will not result in many NULL values; (ii) either c_1 or c_2 has to be a key (or almost a key >80% of values being distinct) to ensure that WT will not grow much after the join; and (iii) T_1 and T_2 have to be under the same dataset. We observed that there are many pairs of tables from unrelated datasets having entirely different subjects that accidentally share columns with many value overlaps, e.g., date attributes that are spanning same period of time for unrelated tables. The third constraint prevents Governor to suggest such pairs of tables. After applying these constraints, we found ~200K many pairs in open.canada.ca. During a user session, the "Add Columns from Other Tables (Join)" panel is populated by querying MongoDB for each of the unioned table's key/almost key columns.

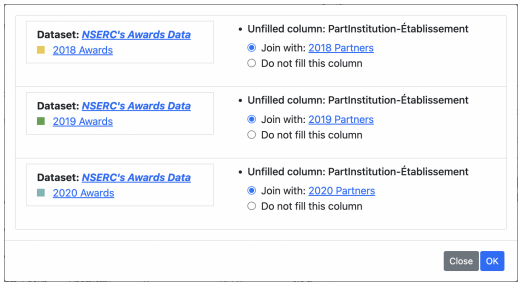

Figure 5: Join plan panel that suggests how to fill each non-joined unionable table inside the working table.

In many use cases, users may need to join each unioned table with a separate table to enrich WT with a single common column. We observed that this is in fact the more frequent scenario in open.canada.ca and happens on datasets that contain periodically published and normalized tables. For example each NSERC Awards dataset for each year (e.g., 2022 Awards), joins with a different NSERC Partners dataset (2022 Partners). We provide a feature to streamline this workflow. After the users adds a column c to one unioned table U_i, the system detects if c can be added to other unioned tables either by joining with T_1 or by other tables that contain the column c, as shown in Figure 5. We call this panel the *join plan panel.*

5.1.3 Provenance Management and Exploration (G3). The main visual cue provided by Governor to distinguish different tables that have been integrated is color coding. When a table is added to WT via a join or union, the system automatically assigns a color to it and uses the color consistently throughout the entire user interface. First, when the user clicks on "Toggle Color" from the Actions panel (Figure 3b), the cells of WT are colored according to colors of the integrated tables. Second, color coding is used in the Working Table Structure (WTS) panel (Figure 3b). WTS is a color-guided provenance summary of the integration that happened to construct WT. The layout of WTS represents the structure of WT. The unioned tables are shown as horizontal rectangles on the left. To the right of these horizontal rectangles are the joined tables. For each "joined column", i.e., a column that does not exist in the schemas of the unioned tables, there is a vertical stack of smaller boxes. If the contents of a joined column come from the same table, this will appear as a single vertical rectangle. Figure 6 shows an example, with two joined columns where the first one comes from different joined tables (except the yellow unioned table is not yet joined with any table for this column), while the right one comes from a single joined table.

Figure 6: Another WTS panel that shows 4 unioned tables that are extended with two columns that come from joined tables: (i) the first joined column comes from 3 joined tables. The yellow unioned table is not yet joined with any table to fill these values; and (ii) the second joined column comes from a single joined table that joins with all 4 of the unioned tables.

Finally, Governor provides several ways for users to go back to the original table from the Working Table. First, when clicking on a block/rectangle from WTS, the Working Table Component Detail (Figure 3e) panel pops up, which shows the title of the table and the columns under it. The users can click on the title to open the original table. Second, when the users click on a cell in WT, Governor opens and locates the cell from the original table.

5.2 Supporting Functionalities

Governor also provides some other basic functionalities to ease the data tables exploration and integration process. **Hiding/Unhiding Columns:** In the open.canada.ca's tabular data corpus, 50% of the tables contain more than 11 columns. The large number of columns can be overwhelming. By default we show 5 columns, picking those with the largest number of unique values. Optionally, the users can pick the columns to show manually by clicking on the column title from "Hide / Unhide Column" section of the action panel.

Basic Spreadsheet Features: Both the Original Table and the Working Table views consist of a few basic spreadsheet featues, including filtering, sorting, and basic column statistics. These features can help users lookup for simple facts without having to download the table and open it in a spreadsheet software

5.3 Extensibility of Governor

Although we have based our work on open.canada.ca, Governor can work on any data portal that uses the CKAN system, which has been one of the most popular software for hosting OGDPs. CKAN is used by 148 OGDPs [39], such as data.gov [4], data.gov.uk [6], and data.gov.sg [5]. These CKAN-powered portals use the same data structure for storing metadata and can be accessed through the same API. However, the attributes stored in the metadata can be customized according to the requirement of each publisher. For example, as a dual-language portal open.canada.ca stores a language field for each file.

Governor can be configured with a single JSON configuration file, which makes it easy for a system administrator to adapt Governor to different CKAN-based portals without writing any new code. Instead, the system administrator only needs to modify the configuration file to specify the metadata fields that should be indexed for search and displayed on the front end, as well as the URL of the CKAN endpoint. Then, by running a single script, Governor can fetch the metadata from the CKAN API automatically, crawl all the data files in parallel, and create all the required indices, allowing the server to be started immediately. To demonstrate this, in the supplementary material, we provide the source code of Governor, which includes the indexing script, and the JSON configuration file to deploy Governor on data.gov.sg.

6 EVALUATION

We conducted a two-part user study with two groups of users to assess the effectiveness and usefulness of Governor from different perspectives. The general purpose of the study was to investigate how people use the system to perform ad hoc search and create integrated tables from multiple original datasets to fulfill their goals, and understand the strengths and weaknesses of the tool. Specifically, Part 1 employed a more controlled setting with three definitive tasks, which ensured that every novel feature of the tool could be assessed. Part 2 adopted a more natural setting with an open-ended task, which allowed us to investigate the flexibility and generalizability of Governor. Also, compared to Part 1, Part 2 was more exploratory and thus we aimed to obtain more qualitative user feedback with in-depth interviews.

As discussed in Section 2, two classes of existing tools partially provide Governor functions: (i) tools that aim to make it easier for users to search and explore open data tables, and (ii) tools for integrating open data tables. A combination of open data search tools with standard spreadsheet software such as Excel would enable our use cases and form a baseline, but switching between tools would require tedious and frequent copy-pasting. Thus, this would not be a fair comparison to Governor. Additionally, the tools that require users to write code to perform data integration [20, 31, 50] would not be comparable to Governor, because our target audience is non-programmers. Detailed comparison has been summarized in Table 1. The closest to our work would be Auctus [23] and Toronto Open Data Search (TODS) [54]. However, the table integration feature of both tools is limited, as discussed in Section 2. Auctus only allows the table to be joined or unioned with one other table, so in our case, users have to frequently download and upload intermediate tables to achieve the same functionality in Governor. TODS, on the other hand, does not support unions, which significantly restricts possible integration tasks. In sum, there exists no single comparable tool that can support all the core functionalities in our scenario to allow users to complete our desired tasks reasonably. A cumbersome combination of tools with much manual effort involved would offer less valued insights into the user experience of Governor. Thus, we decided not to include a baseline in our study design and aimed to qualitatively compare the user experience of Governor with other approaches that participants used before.

6.1 Participants and Apparatus

For study Part 1, we recruited twelve participants (P1-P12) via mailing lists at a local university and by reaching out to people from the open data community. Six participants are between age 18–34, five between age 35–54, and one between 55–74. Six participants are males and six are females. All of the participants have a bachelor's degree or higher (three Bachelors, seven Masters, and two PhDs) whose backgrounds include information technology, electrical engineering, law, public policy, and social services. On a 5-point Likert

scale (1: no familiarity; 5: advanced user of such software), their self-reported familiarity with spreadsheet software (e.g., Excel, LibreOffice Calc, Numbers, Google Sheet) had a median of 4 and a mode of 4; their self-reported familiarity with RDBMS (e.g., Oracle, PostgreSQL, MySQL, SQLite, Db2) had a median of 3 and a mode of 3. Six participants have prior experience working with OGDPs.

For Part 2, we recruited six additional participants (P13-P18) using the same methods. All of the participants are between age 18-34. Four participants are males and two are females. All of the participants have a bachelor's degree or higher (two Bachelors, two Masters, and two PhDs) whose backgrounds include information technology, statistics, and health science. On the same 5-point Likert scale above, their self-reported familiarity with spreadsheet software had a median of 3 and a mode of 3; their self-reported familiarity with RDBMS had a median of 4 and a mode of 4. Two participants have prior experience working with OGDPs.

We conducted the study via remote video conferencing software. Governor was deployed as a web application and participants accessed it from their personal computers. Each participant received a $20 gift card for their time and effort.

6.2 Tasks and Design

To evaluate Governor, we designed the following tasks for our study, where each task is designed to assess one or more design goals introduced in Section 4. Particularly, Part 1 included T1-3 and Part 2 consisted of T4.

T1: Finding Datasets (G1): Participants needed to find out how much money has been granted by a province-level research fund to a researcher. This task requires participants to employ the search and table previewing feature of Governor, including finding and opening the table, adding additional columns, and applying a filter. T1 requires the participants to find an answer from an open data table, which allows us to validate if participants could quickly find the record they are looking for via search.

T2: Unioning Tables (G2): Participants needed to create an integrated table about how the wait time for travelers at a border office changed over the year 2015 and 2016 by unioning eight tables which are published quarterly together and applying a filter based on the name of the border office. This task requires the participants to create an integrated table, which allows us to evaluate if participants could easily integrate multiple tables with Governor.

T3.1: Locating Provenance (G3): Participants were presented with a pre-defined table with four columns via a shared link about how much funding was spent by charitable organizations in Canada on political activities in 2019 , which was created by joining three tables. Then, we asked the participants the following questions:

- TQ1: Can you describe what operations were performed to construct this table?
- TQ2: From how many different tables does this table contain information from? What are these tables?
- TQ3: Can you open the original table that contains the "Description" column of charitable organizations?
- TQ4: Can you show the following cell in the original table?
  - "Legal Name: SecondStreet.org" (from one of the joined tables)
  - "5030:6000" (from unioned table)

This task tests whether the participants can understand a predefined table by interacting with the provenance features of Governor, such as color-coding and working table component detail modal. To test if the participants can locate a cell manually, we disable the "Locate in Original Table" feature in this task.

T3.2: Provenance + Integration (G2 and G3): Participants are required to extend the table presented in T3.1 to contain the data from year 2017 to 2019. This task requires participants to understand the structure of the table from T3.1, and also interact with both the joining and unioning features of Governor. T3.2 allows us to evaluate both table integration and provenance information features of Governor.

T4: Open-ended Data Integration (G1, G2 and G3): Participants are asked to prepare a dataset by unioning or joining at least three tables about research funding awarded to academics or universities of their choice and explain the dataset they have created. This task gives participants the opportunity to explore tables of their interest, starting from many possible different tables published by different research funding agencies, e.g., NSERC, CIHR, SSHERC, or provincial agencies, find these tables by different search modes, perform both joins and unions, and apply different filters. The participants also needed to imagine a usage scenario themselves to explain their final tables. T4 allows us to evaluate Governor comprehensively in an open-ended manner.

6.3 Procedure

The study Part 1 and Part 2, while having different tasks, followed a similar procedure as described below. We began the user study with an introduction session. The experimenter first demonstrated the process of searching a keyword from the open.canada.ca portal and showed the dataset consisted of data from many years which are published periodically in different data tables. Then, the experimenter downloaded a specific table and opened it in Excel to show the participants the columns and values stored in the table. Next, the experimenter performed the same search in Governor and opened the table directly to demonstrate the table previewing feature, along with hiding and unhiding columns. Then, the table was added to the working table and the experimenter demonstrated the table integration features by unioning and joining tables. Finally, the provenance features of Governor were shown by introducing the WTS, color toggling, etc.

Participants were then instructed to perform two practice tasks. The first asked participants to search a dataset and identify an item in a table. The second required them to create a table similar to that in the introduction session (but with a different dataset). Detailed, step-by-step instructions are provided on how to perform these tasks, and the experimenter could answer any questions raised by participants. However, the participants were encouraged to think about how these tasks could be completed without first looking at the instructions. The above procedure assured that the participants got some familiarity with the system and had adequate skills and knowledge to complete the actual tasks. We note that we ensured that no datasets that would appear in the actual tasks for participants were used during the introduction of Governor and practice tasks.

Next, participants were asked to perform all the actual tasks described above. We did not give any direction or hint to complete the task unless the participants got stuck for a long time. After finishing each task, participants filled in the NASA TLX questionnaire based on their experience. Finally, we conducted a semi-structured interview to collect their feedback. The studies lasted between 60 to 80 minutes for each participant. We screen-captured the task sessions and audio-recorded the interviews.

7 RESULTS

In this section, we report our results from the user study, including both quantitative measures and qualitative feedback. Recall that P1-12 performed tasks T1-T3, while P13-18 performed T4.

7.1 Task Performance

We first discuss participants' performance on our tasks and describe our observations of their behaviors.

T1: Finding Datasets: On average, participants spent 3 min 3 s (σ = 1 min 19 s) for T1. Of all the participants, ten perfectly completed the task by meeting all the requirements, and two partially met the requirements by incorrectly reporting the total cost of the project as the government commitment. However, the mistakes due to misinterpretation of the column name and descriptions were mostly due to the ambiguity in the original open dataset and were less of an indicator of the effectiveness of Governor.

An ideal solution to T1 would be: (1) use the researcher's name as a keyword to perform a tuple search, (2) open the summary table for the research fund, and (3) unhide the column for the amount of funding. Of the ten participants who completed the task well, only three followed a very similar approach, while the other seven searched for the research fund's name via tuple or description search. This reveals the flexibility of Governor, supporting multiple ways to find the record. Interestingly, three participants added the original table to the Working Table before applying the filter or unhiding the column. It is encouraging that the flexibility offered by Governor allows most participants to be effective in completing the task, even if they sometimes went off the "optimal" path. During the process, one participant was confused about the difference between the tuple and description search modes and required clarification. These two search modes can potentially be merged into one universal search in the future.

T2: Unioning Tables: On average, participants spent 5 min 1 s (σ = 1 min 50 s) for T2. All of the participants were able to complete T2 successfully. However, five participants got stuck a bit trying to figure out if they had the correct columns and two participants were confused because the data table for 2015 Q1 also contains the last few hours of data from December 31st of 2014. However, similar to T1, this is largely due to the non-descriptive column names and the data cleanness issue which existed in the original open dataset.

In this task, the participant can start by adding the data from one of the eight quarterly-published tables to the Working Table. Then, by expanding the panel "Add Rows from Other Tables (Union)", the system can suggest the seven remaining data tables. The participant needs to click the "Union" button for each of the seven tables to add them to the working table. All of the participants followed this approach. However, three participants complained during the process that the table integration requires too many clicks, or the reloading after each click is annoying, and asked if we have an alternative way to integrate all eight tables at once.

T3.1: Locating Provenance: On average, participants spent 8 min 48 s (σ = 3 min 23 s) for the T3.1. All of the participants were able to answer TQ1 and TQ2 with the information provided by Governor and open the correct table for TQ3. Among the twelve participants, six found the required answer with the "Working Table Structure", three of them also turned on the color-coding of the table to determine the provenance of each column. Four participants found the answer with the "Working Table Component Detail", while two participants got the required answer from the history panel. This indicates that by providing the same provenance information in different ways, the system makes it intuitive for the participants to understand the provenance of data. Even without memorizing all the features of the system, the participant can still complete the tasks with the subset of features that they are familiar with.

On the other hand, the locating subtask (TQ4) did not go smoothly for most of the participants. An optimal way to locate the cell from the original table is to open the original table and filter the data in it with the unique identifier ("BN") of the row that contains the cell, and then unhide the required column. However, only three participants followed this approach. Instead, the other nine participants tried to locate the cell by performing a filter based on the raw value of the cell. This approach worked for the "SecondStreet.org" case as there was only one row with this value in the original table. However, for the "6000" case, due to a large number of table cells containing the same value, after filtering, the table still contains 2683 rows. While in the end, out of the nine participants, eight were able to locate the cell correctly by adding an additional filter, sorting the column, or manually match the information in other columns, the performance of the participants in this task indicated that it was not intuitive to locate a cell based on the unique identifier for most of the users. Being able to locate the cell from the original table automatically is still an important feature of the system.

Additionally, this task also reveals Governor's weakness in filtering the data. Since Governor only features a global filter, the participants cannot filter based on the column they are interested in. If the participant were able to perform the filter based on the "5030" column, they would be able to complete the locating much quicker, as there would only be two rows remaining in the table after the filtering.

T3.2: Provenance + Integration: On average, participants spent 3 min 20 s (σ = 1 min 39 s) for the T3.2. An optimal solution for this task is to first add rows from the "Financial data" table for year 2017 and 2018, then use the system's suggestions to automatically fill the unfilled blocks and complete the Working Table. Out of twelve participants, ten followed this approach. P11 filled the columns manually but ended up with the correct table. P7 got stuck trying to add more columns due to misunderstanding the instructions and required a hint.

T4: Open-ended Data Integration: On average, participants spent 9 min 57 s (σ = 5 min 58 s) for T4. All of the participants were able to integrate at least three tables in the 20-minute time limit. On average, the participants previewed 6.7 tables (σ = 2.9) and integrated 4.7 tables (σ = 2.4) via joining and unioning operations

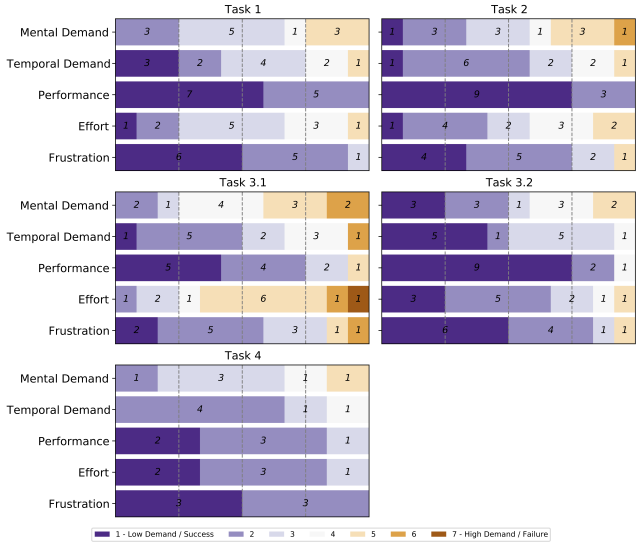

Figure 7: Participants' ratings on the NASA TLX questionnaire for the Tasks (the lower the better)

during the task. Participants were able to create a variety of tables. We give several examples. One participant constructed a table of Ontario government research funding allocated to McMaster University until 2019. Another participant constructed a table of CIHR grants awarded to researchers from the University of Waterloo as main applicant, co-applicant, or partner from year 2001 - 2003. Another participant created a table which summarized the research funds granted to collaborative projects involving both York University and University of Toronto. Three participants, such as the latter two examples, performed both union and join operations, while the other three, such as the first example above, only performed union operations. Participants also used a mix of tuple and description search modes, and all of them previewed some of the original tables before integrating and used hiding/unhiding columns both in previewed tables and in their Working Table. Overall, the variety of tables participants created and how they created these tables indicated that participants were able to understand and use the core features of Governor in several different combinations with relative ease.

Similar to T3.1, the weakness of the filtering feature of Governor impacted the participants' performance on the task as some of them were not able to filter based on the specific pattern of interest. For example, when typing in a keyword like "University of Toronto", the filter will match all the rows that contain "University", "of" and "Toronto", which will cause other institutions in the same city to be matched. Despite this drawback, two participants were able to find a workaround by filtering on the unique identifier of the institutions such as "OrgId".

7.2 Questionnaire Ratings

Figure 7 shows participants' ratings on the NASA TLX questionnaire for all the tasks, respectively. We can see that for T1, most participants (at least 8 out of 12) rated 4 or below on each question,

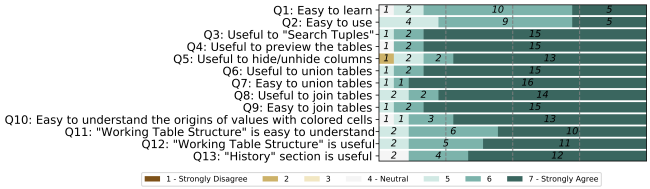

Figure 8: Participants' ratings on the exit-questionnaire (the higher the better)

indicating that they were comfortable using Governor for searching and felt successful in doing their tasks. All of the participants indicated that they successfully accomplished the task, while the mental demand and effort involved depended on the search strategy they use. The results of T2 are similar to that of T1, with a slightly higher rating for frustration. This is consistent with the user's complaints that Governor does not offer the bulk operating feature for performing the union as discussed in Section 7.1. The results of T3.1 shows a high effort level and mental demand as most of the participants were not able to locate the cell of "6000" at the first attempt. While most of them were able to successfully locates the cell in the end, it is not a smooth experience. The results of T3.2 is slightly better than T1 and T2, which might be because participants got familiar with the table structure in T3.1, and also became more proficient with the system after performing the first three tasks. Finally, for T4, most participants (at least 5 out of 6) rated 4 or below for each question. One participant (P18) rated 5 on the "Mental Demand" factor due to the difficulty of finding the columns of interest based on the abbreviated column names. The results of T4 indicates that most of participants felt Governor useful in an open-ended, exploratory data gathering task.

Moreover, the results of the exit-questionnaire as shown in Figure 8. In general, indicate that participants had a good experience of using Governor. A majority of them thought the system was both easy to learn and use (Q1 and Q2). They thought all of the core features of Governor, including searching (Q3), previewing the original tables (Q4), adding rows via unioning (Q6), and adding columns via joining (Q8) are useful. Furthermore, they thought that it was generally easy to perform these operations in Governor (Q7 and Q9). Participants especially appreciate Governor's abilities to summarize the provenance of the integrated table with color-coding and Working Table Structure. They think that it is easy to understand the origins of values with colored cells (Q10), and the information provided in the Working Table Structure is both useful and easy to understand (Q11 and Q12).

7.3 Qualitative Results

Here we report the qualitative feedback from our participants during the semi-structured interviews of our study. We first discuss their general comments and then specific ones grouped into the themes according to our design goals. Finally, we also group the comments where users compared Governor with existing tools they would use for the tasks they performed.

General comments: Overall, participants appreciated the usefulness, functions, and design of the tool. P7 commented "*It's a*

really nice platform and the coloring visualization is really nice. And the displays are really clear. And those functions are easy to understand and really simple." P15 said "*I think the UI is very good, and it's sort of simple and clean and I think it would be really useful. So I think it's great.*" Similarly, P5 mentioned "*I'm very impressed that you have done such a great job as for a research thing. So it looks to be very complex and very, like, complete, I would say.*" This is summarized by P1 in comparison to current OGDPs: "*Even though we're doing open data, it's not really providing a lot of power to people yet, we're just saying, here's the 1000 boxes of paper files, good luck, hope you find the one you're looking for.*" Moreover, the participants wanted to use Governor in their daily jobs, as addressed by P9, "*The app is fantastic and I can't wait to use it. Yes. 100%. I would love to have this tool available to our department tomorrow. We desperately need this tool.*" P3 also thought the tool can be a good education tool for democratizing the concept of open data: "*I sure hope we will be able to use this tool soon, it looks really useful. You could give this to high school students, get them to look at things, even elementary students, so they would start to be able to investigate things, and understand that public data is very valuable.*"

However, several participants complained about the complexity and learnability of the tool. For example, P11 commented that "*I found is that the interface of the system is a little bit too cluttered. [...] This system is hard to learn. But once you learn it, it is very easy to use. So I think that the learning curve for this system is a little bit steep.*" P2 shared a similar opinion: "*Overall, I liked it a lot. I think it was very easy to use. Maybe just learning it was a little bit like, it wasn't that difficult, but it was a lot of features. So it took me a while to get comfortable with it, but I like the ability to search for some entries, and also preview the table, these were I think these are very useful, and also the data integration part.*"

Searching and previewing data (G1): Our participants valued a lot for the data search capabilities provided by Governor. The implemented Search Tuples and Search Descriptions in Governor provided participants with an easier way of identifying useful datasets. P3 addressed this by "*Unless you understand the categories of how government data is organized, it's so hard to find the information you're looking for. So by providing this additional layer into the data it is very powerful.*" Moreover, some participants thought the search available in Governor could potentially benefit novices and non-technical people. "*It just starts to give to the basic person who, you know, isn't maybe that data savvy, a lot more power to do the investigation and understand, like, what's going on.*"-P1.

However, the distinction between Tuple Search and Description Search can be confusing to some participants. For example, P4 commented that "*I still do not get the difference between the tuple and description search.*" Similarly, P10 said that "*The term tuple can be confusing to people from a non-technical background.*" Finally, P3 suggested that we should merge the two search modes: "*What if I want to match both the description and the value? Can it do that?*" P5 also demanded "*searching specific cells within a specific description.*"

Integrating data (G2): Governor supports the core relational data processing operations, including the union and join interactively via the front-end DuckDB to help users integrate datasets (Section 5). In our study, five of the participants felt these terminologies were easy to understand, while five other participants thought that they can be confusing to the end-users and offered us alternative suggestions. For example, P2 suggested us to "*add a description or small example for joining*" to make it more understandable, which was echoed by P3. Additionally, P6 suggested using different wording for "union", such as "append" as "*'union' is a mathematical term.*" Similarly, P7 suggested replacing the wording of "join" with "combine".

Despite the confusion, the participants were excited about data integration capabilities in Governor and found these functions useful. P3 commented that "*I'm maybe not very skilled, but I often, you know, copy-paste data, so I do those. And this is great, because you know, you're not making those copy-paste errors or mismatching the columns or anything. So, I think it is really useful.*" Moreover, some participants appreciated the efficiency of Governor and its potential in handling a large amount of data. For example, P10 said: "*See how fast it is, is pretty amazing!*" Similarly, P18 noticed that when she uses Governor to join 2 tables, "*it just takes a few seconds.*" P9 also mentioned "*With massive datasets, like a big collection of tables, I think it's (Governor) probably very handy.*" Finally, P10 provided a unique insight into integrating tables: "*For adding tables with union and being able to join that. Like, this is all very good. I'm very excited. Just, it's not going to be like a broadly popular thing,*" where he meant data cleaning is an essential step before integrating the tables. While this is out of the scope of this paper, it points out an important future direction to extend Governor with advanced data cleaning features.

However, one concern raised by several participants was the limitation of only integrating tables suggested by the tool. For example, P16 suggested that "*it might be better to provide the option to actually play with SQL stuff*" directly in Governor to give users more flexibility. P7 asked for the feature of joining an arbitrary table: *"Hmm... Do I have to pick up from the list (suggested by the system)? Can I join the table based on my own choice?"* Similar questions were raised by P4 and P9. Participants also suggested several other functions to enhance the data integration experience of Governor. For example, P6, P7, and P12 requested "bulk operation" of selecting tables to union and join, as commented: "*There are lots of clicks to do maybe if you use the union select or like group select.*" Specifically, P10 wanted to have the functions to combine data from different agencies. P4, P10, and P12 demanded some Excel-like features in data aggregation, such as computing the means, sums, etc. Moreover, P8 would like to have "*two separate working tables on two separate panels*" for side-by-side comparisons.

Data provenance (G3): The Working Table Structure (WTS) visualization is one major support offered by Governor to investigate the provenance of the integrated datasets. Participants highly appreciated this feature, as it provided a clear view and mental model of the working table. For example, P12 commented "*Oh, yeah. Like the colors and stuff. That's really cool. WTS is easy to understand. It's like a table structure.*" This was echoed by P9: "*I love this thing. It is amazing. The visual piece really works out.*" Also, P10 said, "*The idea of the squares as different tables was very well done. I like that.*" and also appreciated the history, "*I do like the ability to see all the things that have been applied.*"

Another data provenance support of Governor to distinguish different tables that have been integrated is the color coding. Each integrated table is assigned a color automatically and the color is used consistently across multiple user interface components. This

design is favoured by most of the participants. For example, P2 said: "*I really like how you use the colors,*" which was agreed by P3, P7, P9, and P12. However, P8 was confused by the colors: "*There are like ten colors. I cannot remember which color corresponds to which table.*" P10 also brought up an interesting point regarding the accessibility of the system: "*There are more color-blind people than you know. You should think about them (when designing the system). Maybe you can add some patterns to the blocks.*"

Overall, the value of being able to trace down the data sources in an integrated table in our tool was applauded by the participants, which is encouraging. For example, P3 addressed: "*I think it also drives a lot more within organizations. So for example, where I work, like proactive disclosure, it's kind of a painful activity, and it should just be almost automated, like we should have an internal process that is very smooth and easy. And everyone who contributes to that should understand how it's accessible, how people will compare our data to all kinds of other data that we have, to have that clean and consistent data provided.*" P12 added "*There's a lot of mistrust about data out there for people who don't work with data all the time. And we often have to defend our datasets. And that's probably another way of saying, look, this is where we merge, right? We brought these things together. It's proof to someone that this works.*"

Comparisons to existing tools: During the interview, participants also compared Governor with existing tools they would use for similar open data exploration tasks, such as Google Dataset Search, Kaggle, and the search engine of open data portals, as well as tools they would use for similar table integration tasks, such as spreadsheet software, database systems, and programming languages.

For open data exploration, participants preferred Governor for its ability to search the contents of the tables over existing tools that only support searching over metadata of datasets. For example, P17 mentioned that "*I would say the search feature is much better than the one the government portal provides for sure, because it allows you to search deeply within documents instead of just the surface level information about the documents*", which was agreed by P13, P14, and P18. Participants also liked the preview feature provided by Governor. P12 commented that "*That's easy for me to compare from searching the open data. The website (an open government portal) which obviously doesn't do much for you and urges you to download.*" Similarly, P14 mentioned "*I don't have to download anything, I can just take a peek at the data and then decide what I want to do.*"

Several participants commented that they would prefer Governor over the existing tools for data integration tasks. P1 mentioned that "*These are the things (integrating tables) we usually do in Excel. And in Excel, it is tedious to combine them by copying, pasting, and all that.*" Similarly, P15 pointed out that spreadsheet tools such as "*Excel and Google sheets have very little when it comes to doing table joining stuff. Also, you have to download all of the sheets, and often there's issues with merging the tables.*" Several participants emphasized Governor's capability of suggesting joins and unions automatically compared favorably with existing tools. For example, P14 pointed out that "*I have to kind of manually figure out what are the various things I want to merge and then look for those datasets independently, and then bring them all, download them all, look at their schema, and then figure out which column do I join it on and then join it, and finally create my dataset. So yeah, it could be a really long process. I think the tool you have here is kind of cutting all that time out. It is really easy to just have the joinable and unionable tables suggested for me.*" This is echoed by P17: "*Obviously this solution makes it much easier, because it suggests the columns based on which I can do the joins or the unions, so I don't need to go through the tables manually*", and also agreed by P13 and P18.

When asked about comparing Governor with using programming languages for table integration tasks, participants liked Governor's features of previewing the intermediate table during the data integration process and keep track of the actions that the user has performed. For example, P15 mentioned that "*Sometimes I also use the pandas library, and get data frames, and then I do a join or a merge using the data frames. But the issue with it is really hard to visualize anything. For instance, maybe you have to print your data frames, but then there's limited options on the UI because it's just a standard print.*" Similarly, P18 also mentioned "*The web app is visualizing everything. It is easier to keep track of the action I performed. With a good user interface, who would want to use pandas? "*

However, participants also pointed out that Governor is not as featureful as the existing spreadsheet software and database systems. For example, P14 pointed out that "*The filter definitely needs some improvement. There should be a way to filter by column instead of just filtering for a value in all the columns.*" This is echoed by P16 and P18. P13 suggested to add aggregation feature as "*it is commonly used*". P18 requested a feature of "*filtering out the null values*".

8 DISCUSSION

While the results of the user study indicate the effectiveness of Governor for providing our desired feature set, the system still has drawbacks. Through the participant interviews we carried out, we identified three broad usability challenges. We first discuss these challenges and then other limitations of Governor and our study.

Searching: Two participants stated that having 2 separate search modes is confusing and suggested to merge both search features into a single hybrid search. We currently separated these because we use a different back end software for each functionality and we did not explore mechanisms to rank two sets of results within our current scope.

Table Integration Process: Governor supports constructing WT in a step-by-step fashion. After each operation, WT is reloaded to reflect the change immediately. For example, for T2, users have to click the "Union" button eight times, and wait for WT to be reloaded each time. Three participants asked for support of bulk integration operations for table assembling or asserted that reloading the working table after each operation is inefficient.

Data Analytics and Other Data Transformation Functions: Aside from basic filtering and sorting of tables, Governor does not support any data transformation or analytics functions, such as data aggregation, range selection, chart plotting, etc, which were mentioned by several participants. For example; three participants requested a table filtering based on a particular column, three emphasized the importance of data aggregation capability that is not offered by Governor, and two asked for the functionality of reordering the columns in the tables.

Although we designed Governor as a search and data integration system, these functionalities are frequently used in data gathering and analytics pipelines. For example, if the users would like to figure out the total amount of NSERC funds granted to a specific institution within a year using the current version of Governor, they have to export the data from Governor into a spreadsheet software, in order to perform a sum. In addition, the tables users create/integrate from data lakes can contain results of basic analytics instead of raw tuple values, e.g., results of aggregations instead of the individual raw values. Currently Governor is limited to only integrating raw values.

Other Limitations: Governor currently only indexes the csv files from open.canada.ca. While it is easy to extend Governor to support other CKAN-based data portals, it may require additional works to extend Governor to other open data pulishing systems such as Socrata [11] and magda [8]. Governor is currently designed to support a single portal's data. Users may frequently need to gather data from multiple portals (e.g., both from a federal as well as a state/province government's portal). Furthermore, Governor's table integration features are also limited to the tables suggested automatically by the system as discussed in Section 5. These did not create problems in our user study. However, users may prefer to have a manual mechanism to find and integrate tables.

Study Limitations: Our user study has few unavoidable limitations. First, we did not compare our system with any baseline systems as we discussed in Section 6 because no prior system we are aware of provides the full functionality of Governor and baselines that use mix of systems, e.g., an open data search engine along with a spreadsheet software, would lead to unfair comparisons as they could require a lot of copy pasting and downloading. Furthermore, our study was conducted in a lab setting and we did not have a large number of participants. Future evaluation of Governor within a realistic setting would be necessary to reveal more potential usability issues of the system.

9 CONCLUSIONS

We have presented Governor, a web application that aims to turn OGDPs from raw publishing portals into interactive DBMSs to make them more accessible and useful to non-technical users. Our key approach is to provide a set of DBMS features to end users to search the records in the tables, integrate multiple datasets in OGDPs through a few clicks, and facilitate fact-checking by providing color-guided, visual provenance exploration. We hope further research and development can build an ecosystem of user-friendly tools and systems to make finding, integrating, and analyzing datasets in OGDPs more accessible to general public. We think there is opportunity for academic research to take the lead in this direction, as there is very little available tools in this space and the sizes of the OGDPs are currently relatively small, so it is possible to develop systems that can index and process entire OGDPs with modest resources as we have done in this work.

ACKNOWLEDGMENTS

This research is supported by an NSERC grant and a grant from the Waterloo-Huawei Joint Innovation Laboratory.

REFERENCES

[1] 2022. Apache Arrow https://arrow.apache.org/.
[2] 2022. Apache Parquet https://parquet.apache.org/.
[3] 2022. CKAN https://ckan.org/.
[4] 2022. Data.gov https://data.gov/.
[5] 2022. Data.gov.sg https://data.gov.sg/.
[6] 2022. data.gov.uk https://data.gov/.
[7] 2022. Elasticsearch https://www.elastic.co/.
[8] 2022. Magda https://magda.io//.
[9] 2022. MongoDB https://mongodb.com/.
[10] 2022. Open Data Principles https://open.canada.ca/en/open-data-principles.
[11] 2022. Socrata https://socrata.com/.
[12] 2022. SQLite compiled to JavaScript https://github.com/sql-js/sql.js/.
[13] 2022. Tableau https://www.tableau.com/.
[14] 2022. tidb-wasm https://github.com/tidb-incubator/tidb-wasm.
[15] 2022. Vue.js https://vuejs.org/.
[16] 2022. WebAssembly https://webassembly.org/.
[17] 2023. Governor Web Application Source Code. https://github.com/mewim/GovernorApp.
[18] Omar Benjelloun, Shiyu Chen, and Natasha Noy. 2020. Google dataset search by the numbers. In *International Semantic Web Conference*. Springer, 667–682.
[19] Alex Bogatu, Alvaro A. A. Fernandes, Norman W. Paton, and Nikolaos Konstantinou. 2020. Dataset Discovery in Data Lakes. In *Proceedings of the 2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE. https://doi.org/10.1109/icde48307.2020.00067
[20] Alex Bogatu, Norman W. Paton, Mark Douthwaite, and Andre Freitas. 2022. Voyager: Data Discovery and Integration for Data Science. *Journal of Data and Information Quality* (jul 2022). https://doi.org/10.48786/edbt.2022.47
[21] Dan Brickley, Matthew Burgess, and Natasha Noy. 2019. Google Dataset Search: Building a Search Engine for Datasets in an Open Web Ecosystem. In *The World Wide Web Conference*. https://doi.org/10.1145/3308558.3313685
[22] Steven P. Callahan, Juliana Freire, Emanuele Santos, Carlos E. Scheidegger, Cláudio T. Silva, and Huy T. Vo. 2006. VisTrails: Visualization Meets Data Management. In *Proceedings of the 2006 ACM SIGMOD international conference on Management of data - SIGMOD '06*. https://doi.org/10.1145/1142473.1142574
[23] Sonia Castelo, Rémi Rampin, Aécio Santos, Aline Bessa, Fernando Chirigati, and Juliana Freire. 2021. Auctus: A Dataset Search Engine for Data Discovery and Augmentation. *Proceedings of the VLDB Endowment* 14, 12 (jul 2021), 2791–2794. https://doi.org/10.14778/3476311.3476346
[24] Raul Castro Fernandez, Jisoo Min, Demitri Nava, and Samuel Madden. 2019. Lazo: A Cardinality-Based Method for Coupled Estimation of Jaccard Similarity and Containment. In *Proceedings of the 2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE. https://doi.org/10.1109/icde.2019.00109
[25] Adriane Chapman, Elena Simperl, Laura Koesten, George Konstantinidis, Luis-Daniel Ibáñez, Emilia Kacprzak, and Paul Groth. 2020. Dataset search: a survey. *The VLDB Journal* 29, 1 (2020), 251–272.
[26] Ran Chen, Di Weng, Yanwei Huang, Xinhuan Shu, Jiayi Zhou, Guodao Sun, and Yingcai Wu. 2022. Rigel: Transforming Tabular Data by Declarative Mapping. *IEEE Transactions on Visualization and Computer Graphics* (2022), 1–11. https://doi.org/10.1109/TVCG.2022.3209385
[27] Michael Correll. 2019. Ethical Dimensions of Visualization Research. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/3290605.3300418
[28] Ian Drosos, Titus Barik, Philip J. Guo, Robert DeLine, and Sumit Gulwani. 2020. Wrex: A Unified Programming-by-Example Interaction for Synthesizing Readable Code for Data Scientists. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–12. https://doi.org/10.1145/3313831.3376442
[29] Alon Y. Halevy. 2001. Answering queries using views: A survey. *The VLDB Journal* 10, 4 (dec 2001), 270–294. https://doi.org/10.1007/s007780100054
[30] Jeffrey Heer, Jock Mackinlay, Chris Stolte, and Maneesh Agrawala. 2008. Graphical Histories for Visualization: Supporting Analysis, Communication, and Evaluation. *IEEE Transactions on Visualization and Computer Graphics* (2008), 1189–1196.
[31] Ahmed Helal, Mossad Helali, Khaled Ammar, and Essam Mansour. 2021. A Demonstration of KGLac: A Data Discovery and Enrichment Platform for Data Science. *Proceedings of the VLDB Endowment* 14, 12 (jul 2021), 2675–2678. https://doi.org/10.14778/3476311.3476317
[32] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.
[33] Sean Kandel, Jeffrey Heer, Catherine Plaisant, Jessie Kennedy, Frank van Ham, Nathalie Henry Riche, Chris Weaver, Bongshin Lee, Dominique Brodbeck, and Paolo Buono. 2011. Research Directions in Data Wrangling: Visuatizations and Transformations for Usable and Credible Data. *Information Visualization* 10, 4 (oct 2011), 271–288.
[34] Sean Kandel, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. 2011. Wrangler: Interactive visual specification of data transformation scripts. In *Proceedings of the sigchi conference on human factors in computing systems*. 3363–3372.

[35] Laura M. Koesten, Emilia Kacprzak, Jenifer F. A. Tennison, and Elena Simperl. 2017. The Trials and Tribulations of Working with Structured Data: -A Study on Information Seeking Behaviour. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. 1277–1289. https://doi.org/10.1145/3025453.3025838

[36] André Kohn, Dominik Moritz, Mark Raasveldt, Hannes Mühleisen, and Thomas Neumann. 2022. DuckDB-Wasm: Fast Analytical Processing for the Web. *Proceedings of the VLDB Endowment* 15, 12 (aug 2022), 3574–3577. https://doi.org/10.14778/3554821.3554847

[37] Christos Koutras, George Siachamis, Andra Ionescu, Kyriakos Psarakis, Jerry Brons, Marios Fragkoulis, Christoph Lofi, Angela Bonifati, and Asterios Katsifodimos. 2021. Valentine: Evaluating Matching Techniques for Dataset Discovery. In *Proceedings of the 2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE. https://doi.org/10.1109/icde51399.2021.00047

[38] Fatemeh Nargesian, Erkang Zhu, Ken Q. Pu, and Renée J. Miller. 2018. Table Union Search on Open Data. *Proceedings of the VLDB Endowment* 11, 7 (mar 2018), 813–825. https://doi.org/10.14778/3192965.3192973

[39] Sebastian Neumaier, Jürgen Umbrich, and Axel Polleres. 2016. Automated Quality Assessment of Metadata across Open Data Portals. *J. Data and Information Quality* 8, 1, Article 2 (oct 2016), 29 pages. https://doi.org/10.1145/2964909

[40] Ouellette, Paul and Sciortino, Aidan and Nargesian, Fatemeh and Bashardoost, Bahar Ghadiri and Zhu, Erkang and Pu, Ken Q. and Miller, Renée J. 2021. RONIN: Data Lake Exploration. *Proceedings of the VLDB Endowment* 14, 12 (jul 2021), 2863–2866. https://doi.org/10.14778/3476311.3476364

[41] Eric D. Ragan, Alex Endert, Jibonananda Sanyal, and Jian Chen. 2016. Characterizing Provenance in Visualization and Data Analysis: An Organizational Framework of Provenance Types and Purposes. *IEEE Transactions on Visualization and Computer Graphics* 22, 1 (Jan. 2016), 31–40. https://doi.org/10.1109/TVCG.2015.2467551

[42] El Kindi Rezig, Anshul Bhandari, Anna Fariha, Benjamin Price, Allan Vanterpool, Vijay Gadepally, and Michael Stonebraker. 2021. DICE: Data Discovery by Example. *Proceedings of the VLDB Endowment* 14, 12 (jul 2021), 2819–2822. https://doi.org/10.14778/3476311.3476353

[43] Nischal Shrestha, Titus Barik, and Chris Parnin. 2021. Unravel: A Fluent Code Explorer for Data Wrangling. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. https://doi.org/10.1145/3472749.3474744

[44] Avi Silberschatz, Henry F. Korth, and S. Sudarshan. 2020. Chapter 7: Relational Database Design. In *Database System Concepts, Seventh Edition*. McGraw-Hill Book Company, 303–360.

[45] Monica Swamiraj and Luanne Freund. 2015. Facilitating the discovery of open government datasets through an exploratory data search interface.

[46] Marcin Wylot, Manfred Hauswirth, Philippe Cudré-Mauroux, and Sherif Sakr. 2018. RDF Data Storage and Query Processing Schemes: A Survey. *Comput. Surveys* 51, 4, Article 84 (jul 2018), 36 pages. https://doi.org/10.1145/3177850

[47] Siyuan Xia, Nafisa Anzum, Semih Salihoglu, and Jian Zhao. 2021. KTabulator: Interactive Ad hoc Table Creation using Knowledge Graphs. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 100:1–14. https://doi.org/10.1145/3411764.3445227

[48] Kai Xiong, Siwei Fu, Guoming Ding, Zhongsu Luo, Rong Yu, Wei Chen, Hujun Bao, and Yingcai Wu. 2022. Visualizing the Scripts of Data Wrangling with SOMNUS. *IEEE Transactions on Visualization and Computer Graphics* (2022).

[49] Kai Xu, Alvitta Ottley, Conny Walchshofer, Marc Streit, Remco Chang, and John Wenskovitch. 2020. Survey on the Analysis of User Interactions and Visualization Provenance. *Computer Graphics Forum* 39, 3 (2020), 757–783. https://doi.org/10.1111/cgf.14035

[50] Zhang, Yi and Ives, Zachary G. 2020. Finding Related Tables in Data Lakes for Interactive Data Science. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. https://doi.org/10.1145/3318464.3389726

[51] Erkang Zhu, Dong Deng, Fatemeh Nargesian, and Renée J. Miller. 2019. JOSIE: Overlap Set Similarity Search for Finding Joinable Tables in Data Lakes. In *Proceedings of the 2019 International Conference on Management of Data*. https://doi.org/10.1145/3299869.3300065

[52] Erkang Zhu, Yeye He, and Surajit Chaudhuri. 2017. Auto-Join: Joining Tables by Leveraging Transformations. *Proceedings of the VLDB Endowment* 10, 10 (jun 2017), 1034–1045. https://doi.org/10.14778/3115404.3115409

[53] Erkang Zhu, Fatemeh Nargesian, Ken Q. Pu, and Renée J. Miller. 2016. LSH Ensemble: Internet-Scale Domain Search. *Proceedings of the VLDB Endowment* 9, 12 (aug 2016), 1185–1196. https://doi.org/10.14778/2994509.2994534

[54] Erkang Zhu, Ken Q. Pu, Fatemeh Nargesian, and Renée J. Miller. 2017. Interactive Navigation of Open Data Linkages. *Proceedings of the VLDB Endowment* 10, 12 (aug 2017), 1837–1840. https://doi.org/10.14778/3137765.3137788

A USING DUCKDB-WASM IN THE BROWSER

The data displayed in WT (as well as the Original Table View) is stored in DuckDB-Wasm, which is the WebAssembly [16] version of DuckDB [36]. DuckDB is a SQL OLAP database management system that can run inside the web browser. Loading entire tables and processing them at the front end is against today's common wisdom of running the database at the back end and sending paginated results to the front end. However, this architecture is suitable for our system (and possibly for other applications) and has the following benefits.

- WebAssembly: The WebAssembly [16] technology finally enables full-fledged databases [12, 14, 36] to be compiled for the web browser and run at near-native performance.
- High compressibility of open datasets: Open dataset tables have large number of repeated values, which makes them highly compressible. For example, the size of the largest OGDP, data.gov is 2120 GB in uncompressed raw file size and 434 GB when compressed. Similarly open.canada.ca is 346 GB vs 126 GB compressed size. By loading the entire table at once, the system can send the data in a format with a high compression ratio such as Apache Parquet and reduce the total traffic required to load the table.
- Eliminating unnecessary data transfers: Storing entire tables in the browser gives Governor opportunities to avoid data transfers in several cases. When users change the sorting of the WT or filter it, the front end does not fetch any data from the back end. These operations are directly performed by DuckDB's highly optimized parallel sorting and filtering capabilities.
- Simpler state maintenance: Using DuckDB at the front end enables WT to be modeled as a database view, which is created by a single SQL command compiled from all the joining, unioning and filtering operations the user has performed. Upon a new operation, the system first appends a log for the operation. Then, the front end checks if all the required tables are loaded to the local DuckDB and pulls any missing table from the server. After that, the front end compiles all the logs into a single SQL command, drops the current WT view from DuckDB and recreates it with the newly compiled SQL command, all in the browser. This approach also enables the WT to be re-created easily, which facilitates Governor's "shareable link" feature. When a shareable link is created, the front end sends all the logs to the back end, which gets stored in the MongoDB. When the link is later used, the front end simply retrieves the logs and loads all the required data tables from the back end, then recreates the DuckDB view, and finally renders the user interface.

One potential drawback of sending full tables to front end is the slowdown of the initial loading. A system that loads data with server-side pagination can usually output the first page quickly, as only partial results are required by the front end. Despite loading full tables from a remote server, we measured that Governor can still load large tables at a speed comparable to native desktop spreadsheet software such as Excel. Table 2 shows a comparison of the loading time between Governor and Excel (version 16.63.1) for a set of large tables sampled from open.canada.ca. The files loaded by Governor are pre-compressed and cached by Cloudflare CDN [1],

[1] https://www.cloudflare.com/

Table 2: Comparison of the loading times between Governor (using DuckDB-Wasm) and Excel.

File	Original Size	Compressed Size	# Rows	# Cols	Excel Loading Time	Governor Loading Time
1991 Awards	10 MB	2 MB	18458	34	**2.00 sec**	2.04 sec
Historical DriveBC Events	91 MB	8 MB	201802	25	10.62 sec	**7.55 sec**
Complete file: 2009 to today	335 MB	31 MB	442690	46	29.72 sec	**17.44 sec**
Proactive Disclosure - Grants and Contributions	850 MB	68 MB	590021	37	58.59 sec	**39.21 sec**

while the same original csv files are loaded by Excel directly from the local file system. The comparisons are performed on a Mac Pro computer with a 3.5 GHz 8-Core Intel Xeon W CPU and 32 GB of RAM. The Internet connection is throttled down to 50 Mbps, and the browser cache is disabled. As shown in Table 2, Governor can load all of the listed tables at a speed similar to or faster than Excel due to the high compression ratio and the near-native performance of WebAssembly.