Data Schema Complexity Analysis - GitHub Research Project

Group Members: Rani Maklada, Neil Hamza, Jeries Shomaly, Bashar Yousef

About the Project

Welcome to our intriguing research endeavor - the Data Schema Complexity Analysis. In this study, we explore the intricate relationship between a project's database schema complexity and its GitHub activity. Our working hypothesis suggests that as the complexity of the database schema increases, there's a corresponding augmentation in the number of commits and issues on GitHub. This correlation offers valuable insights into the challenges and maintenance efforts linked to managing such projects.

Our research is centered around an in-depth analysis of a diverse array of projects hosted on GitHub. By harnessing the GitHub API, extracting data with Gitana, visualizing database schemas using DBDiagram.io, and leveraging image analysis with OpenCV, we aim to unveil profound insights into how schema design impacts both project development and management.

Our website:

https://rani-maklada.github.io/speedsters.github.io/

Out code repository in GitHub:

https://github.com/rani-maklada/speedsters

Gitana: Extracting Crucial Data from Source Code Repositories

Tool Explanation:

Gitana is a potent data extraction tool tailored to source code repositories, with a special focus on Java and Python projects hosted on platforms like GitHub. It provides a streamlined approach to gathering essential information from repositories, offering insights into code details, commit histories, issue data, and more. With its versatility and efficiency, Gitana enables researchers to acquire a comprehensive understanding of a project's development lifecycle.

Utilization in Our Solution:

In our research journey, Gitana emerged as a critical instrument in collecting imperative data from source code repositories. We harnessed Gitana to systematically extract commit histories, user data, issue information, and other pertinent details from each identified repository. This extensive data formed the foundation of our analysis, providing insights into the trajectory of project development. Gitana's targeted data extraction capabilities allowed us to efficiently amass the necessary data points required to comprehensively understand the complexities of each project's evolution.

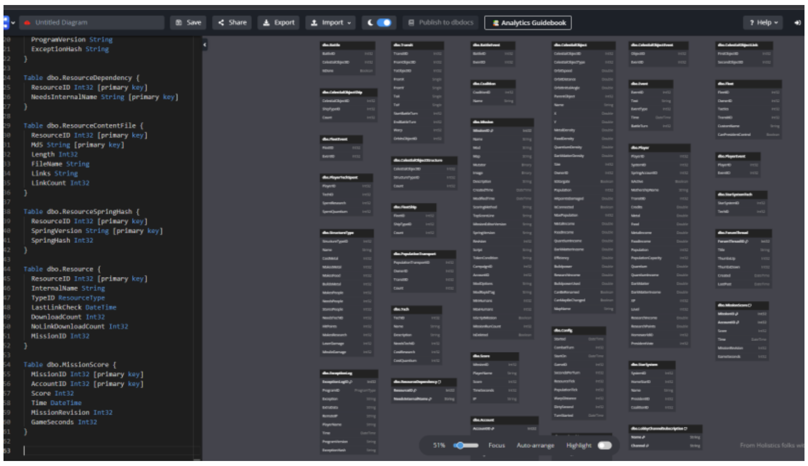
DBDiagram.io: Visualizing and Analyzing Database Schemas

Tool explanation

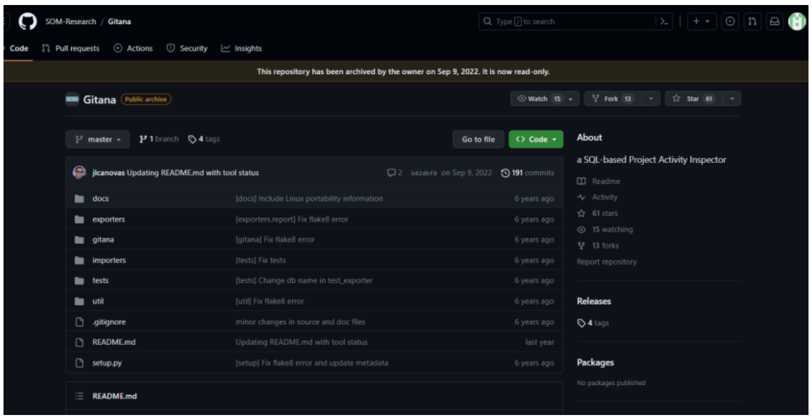
DBDiagram.io is a powerful and intuitive database design tool. It offers the capability to transform textual database schema definitions into clear, visually engaging diagrams that represent the intricate structure and relationships within a database. With its user-friendly interface, DBDiagram.io facilitates the visualization of complex database designs, making it an invaluable tool for researchers and developers alike.

Utilization in Our Solution:

DBDiagram.io played a pivotal role in our research process by providing a visual representation of the database schemas within the repositories. With the help of Selenium automation, we seamlessly uploaded the DBML files, which contain the schema definitions, to DBDiagram.io. This step not only enabled us to visualize the complexities of the database structures but also set the stage for further analysis. The visual diagrams generated by DBDiagram.io formed the basis for our subsequent analysis using OpenCV. Through DBDiagram.io, we gained a visual understanding of the intricate relationships and structures within the databases, enhancing our insights into the projects' complexities.



```
  ProgramVersion String
  ExceptionHash String
}

Table dbo.ResourceDependency {
  ResourceID Int32 [primary key]
  NeedsInternalName String [primary key]
}

Table dbo.ResourceContentFile {
  ResourceID Int32 [primary key]
  Md5 String [primary key]
  Length Int32
  FileName String
  Links String
  LinkCount Int32
}

Table dbo.ResourceSpringHash {
  ResourceID Int32 [primary key]
  SpringVersion String [primary key]
  SpringHash Int32
}

Table dbo.Resource {
  ResourceID Int32 [primary key]
  InternalName String
  TypeID ResourceType
  LastLinkCheck DateTime
  DownloadCount Int32
  NoLinkDownloadCount Int32
  MissionID Int32
}

Table dbo.MissionScore {
  MissionID Int32 [primary key]
  AccountID Int32 [primary key]
  Score Int32
  Time DateTime
  MissionRevision Int32
  GameSeconds Int32
}
```



SOM-Research / Gitana

This repository has been archived by the owner on Sep 9, 2022. It is now read-only.

Gitana Public archive — Watch 15 · Fork 13 · Star 61

master · 1 branch · 4 tags

jlcanovas Updating README.md with tool status — 2 · b82dbf8 on Sep 9, 2022 · 191 commits

docs	[docs] Include Linux portability information	6 years ago
exporters	[exporters.report] Fix flake8 error	6 years ago
gitana	[gitana] Fix flake8 error	6 years ago
importers	[tests] Fix tests	6 years ago
tests	[tests] Change db name in test_exporter	6 years ago
util	[util] Fix flake8 error	6 years ago
.gitignore	minor changes in source and doc files	6 years ago
README.md	Updating README.md with tool status	last year
setup.py	[setup] Fix flake8 error and update metadata	6 years ago

README.md

About

a SQL-based Project Activity Inspector

Readme · Activity · 61 stars · 15 watching · 13 forks · Report repository

Releases: 4 tags

Packages: No packages published

By Using Gitana and DBDiagram.io, we advanced our research to gain multifaceted insights into the interplay between data schema complexities and GitHub repository activities. These tools collectively enabled us to systematically gather, visualize, and analyze data, ultimately contributing to a more comprehensive understanding of the challenges and dynamics of project development and management.

Solution Process Overview

GitHub API and Repository Discovery:

The research commences by utilizing the GitHub API to search repositories for specific file extensions, such as .dbml. This phase serves as the initial data collection step, identifying relevant projects for further analysis.

Data Collection and Storage:

Repositories identified through the GitHub API search are compiled and stored in a structured format, often a TXT file. This compilation acts as a central repository, facilitating easy access and reference for the duration of the research.

Data Extraction with Gitana:

Leveraging tools like Gitana, the research extracts essential information from repositories, such as commit histories, issue data, and user interactions. This extracted data forms the foundation for subsequent analysis and insights.

DBDiagram.io Visualization with Selenium:

DBDiagram.io is employed to visualize database schemas based on the extracted data. Automation tools like Selenium assist in streamlining the process by automating the uploading of data to the visualization platform.

Visual Analysis with OpenCV:

OpenCV is harnessed for visual analysis, evaluating images or diagrams generated by tools like DBDiagram.io. This analysis often involves assessing shapes, patterns, sizes, and other visual elements to derive insights.

Repository Complexity Analysis:

Further extraction of data using tools like Gitana is conducted to gain an in-depth understanding of repository activities. This includes details about commits, issues, and other interactions, contributing to the overall assessment.

Correlation and Graphical Analysis:

The research process synthesizes data from different sources, aiming to establish correlations and patterns. Graphical representations are employed to visualize these correlations, offering a visual summary of insights.

Insightful Documentation and Reporting:

The final phase involves documentation and reporting. The insights derived from the analysis, correlations, and visualizations are documented in comprehensive reports and presentations that communicate the research findings effectively.

Research Workflow

To ensure the precision, accuracy, and comprehensiveness of our research, we've established a systematic and thorough workflow. This workflow guarantees that we obtain meaningful insights from the data while maintaining data integrity. The entire process can be divided into the following stages:

1. **Initial Project Setup:**
   Before delving into the research, ensure the proper setup of your environment. Clone the project repository and navigate through the requirements.txt file to install the necessary dependencies. For utilizing Gitana, an additional setup is required: Gitana is developed on Windows 7 and relies on the following:
   - Git 2.9.3
   - MySQL Server 5.6
   - Python 2.7.6

   User Manual and Setup Steps:

   - Each team member should follow the installation instructions for Gitana's requirements.
   - Activation of the virtual machine is crucial for Gitana usage.

   Input Format:

   - Project repository and dependencies.
   - Virtual machine activation steps.

   Output Format:

   - Prepared environment for research

2. **Repository Discovery:** Before data extraction, you need to create your GitHub access token. This token acts as the input to the research process. The GitHub API will utilize this token to search for repositories containing .dbml files. The result of this search is then saved in a structured .txt file.

   Input Format:

   - GitHub access token.

   User Manual and Setup Steps:

   - Detailed instructions on creating a GitHub access token.

   Output Format:

   - Structured .txt file containing repository details.

3. **Data Collection:**
   Building upon the .txt file created in the previous step, the data collection process involves extracting essential information from each identified repository.

   Input Format:

   - Structured .txt file with repository details.

   Output Format:

   - Extracted repository data.

4. **Repository Cloning:**
   Gitana's capabilities require local cloning of repositories. Utilizing the list of repositories, clone each identified repository locally.

   Input Format:

   - List of repositories from .txt file.
   - User Manual and Setup Steps:
   - Step-by-step instructions for repository cloning.

   Output Format:

   - Cloned repositories locally.

5. **Data Extraction:**
   Leveraging Gitana, extract detailed commit histories, issue data, user information, and more from each cloned repository. Save this data for further analysis.

   Input Format:

   - Extracted repository data.

   Output Format:

   - Structured MySQL database with extracted data.

6. **Database Management:**
   Streamline and store the extracted data in a structured MySQL database. This design facilitates rapid data retrieval and complex analyses.

   Input Format:

   - Extracted repository data.

   Output Format:

   - Structured MySQL database.

7. **DBML Analysis:**
Utilize the GitHub API to extract .dbml files from repositories and save them as separate files.

Input Format:

- Extracted repository data.

Output Format:

- .dbml files saved individually.

8. **Automation:**
Utilize Selenium to automate the uploading of .dbml files to DBDiagram.io. This step yields diagram images that are saved in the same folder.

Input Format:

- Individual .dbml files.
- User Manual and Setup Steps:
- Selenium automation instructions.

Output Format:

- Diagram images saved locally.

9. **Image Processing:**
Analyze the diagram images using OpenCV to extract insights such as the number of tables and schema complexity.

Input Format:

- Diagram images.

Output Format:

- Analyzed diagram insights.

10. **Repository Complexity Analysis:**
Continue using Gitana to extract commit histories and issue data for each repository, contributing to an in-depth understanding of their development dynamics.

Input Format:

- Extracted repository data.

Output Format:

- Commit and issue insights.

11. **Final Analysis:**
Combine insights from the diagram analysis and repository complexities to establish correlations. This contributes to the holistic understanding of the interplay between schema intricacies and repository activities.

Input Format:

- Diagram analysis and repository complexities.

Output Format:

- Correlations and insights.

12. **Documentation and Reporting:**
Compile the research findings, correlations, and insights to craft comprehensive reports and presentations.

Input Format:

- Correlations, insights, and analysis results.

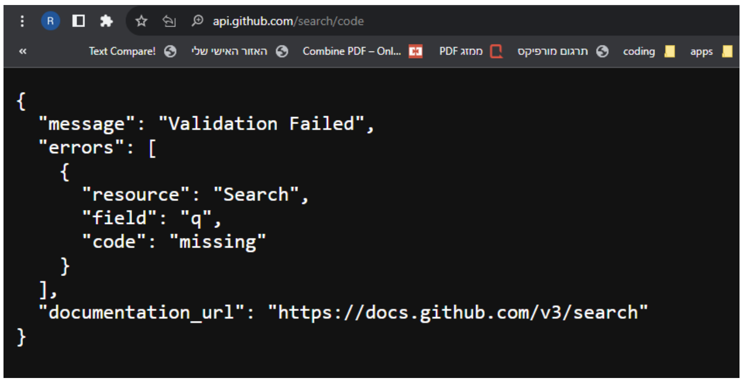
Output Format:

- Research reports and presentations.

By meticulously following these steps and adhering to the provided setup instructions, our research aims to provide a profound view of the relationship between data schema complexity and GitHub repository activities, ultimately shaping best practices in project development and management.

Other Tools We Used – Our research used several key tools:

- **GitHub API:** This platform enables programmatic interaction with GitHub features and data, facilitating data extraction, analysis, and manipulation directly from GitHub repositories. It proves invaluable in identifying repositories containing .dbml files, ensuring comprehensive data collection.

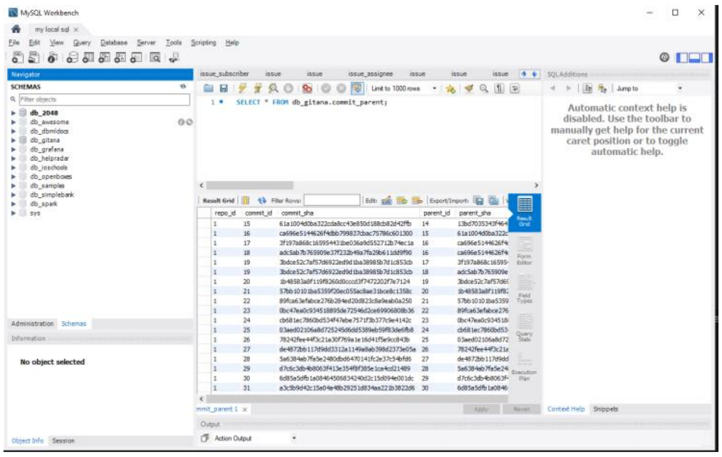


```
{
  "message": "Validation Failed",
  "errors": [
    {
      "resource": "Search",
      "field": "q",
      "code": "missing"
    }
  ],
  "documentation_url": "https://docs.github.com/v3/search"
}
```

- **Selenium:** A robust browser automation tool, Selenium automates manual tasks within a browser environment. In our research, Selenium simplifies the process of uploading DBML files to DBDiagram.io, enhancing efficiency and accuracy.

- **OpenCV:** OpenCV is an open-source computer vision and machine learning library, offering versatile tools for image analysis and processing. It plays a crucial role in our study by enabling us to analyze visual data schema representations and assess their complexity based on factors like shapes, sizes, and line counts.

- **MySQL:** A widely-used relational database management system, MySQL is known for its speed, reliability, and ease of use. In our research, we store and analyze data extracted from various sources, including GitHub and DBDiagram.io, in a structured MySQL database. This enables us to perform complex queries, analyze relationships, and extract insights from aggregated data.



```
SELECT * FROM db_gitana.commit_parent;
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

repo_id	commit_id	commit_sha	parent_id	parent_sha
1	15	61a1004d0ba322cda8cc43e850d188cb82d42ffb	14	13bd7035343f464
1	16	ca696e5144626f4dbb799837dbac75786c601300	15	61a1004d0ba322c
1	17	3f197a868c16595443 1be036a9d552712b74ec1a	16	ca696e5144626f4
1	18	adc5ab7b765909e37f232b49a7fa29b611dd9f90	16	ca696e5144626f4
1	19	3bdce52c7af57d6922ed9d1ba38985b7d1c853db	17	3f197a868c16595
1	19	3bdce52c7af57d6922ed9d1ba38985b7d1c853db	18	adc5ab7b765909e
1	20	1b48583a8f119f8260d0cccd3f7472202f7e7124	19	3bdce52c7af57d6
1	21	57bb10101ba5359f20ec055ac8ae31bce8c1358c	20	1b48583a8f119f8
1	22	89fca63efabce276b284ed20d823c8a9eab0a250	21	57bb10101ba5359
1	23	0bc47ea0c934518895de72546d2ce69906808b36	22	89fca63efabce276
1	24	cb681ec7860bd534f47ebe7571f3b377c9e4142c	23	0bc47ea0c934518
1	25	03aed02106a8d725245d6dd5389eb59f83de6fb8	24	cb681ec7860bd53
1	26	78242fee44f3c21a30f769a1e16d41f5e9cc843b	25	03aed02106a8d72
1	27	de4872bb117d9dd3312a1149a8ab398d2373e05a	26	78242fee44f3c21a
1	28	5a6384ab7fa5e2480dbd6470141fc2e37c54bfd6	27	de4872bb117d9dd
1	29	d7c6c3db4b8063f413e354f8f385e1ca4cd21489	28	5a6384ab7fa5e24
1	30	6d85a5dfb1a08464506834240d2c15d094e001dc	29	d7c6c3db4b8063f
1	31	a3c3b9d42c15a04a48b29251d834aa221b3822d6	30	6d85a5dfb1a0846

By effectively harnessing these tools, our research aims to unravel the intricate relationship between data schema complexity and GitHub repository activities, providing valuable insights into the realm of project development and management.