# Minimally-Intrusive Augmentation of Data Science Workflows

Andreas M. Wahl, Peter K. Schwab, and Richard Lenz

Lehrstuhl für Informatik 6 (Datenmanagement), FAU Erlangen-Nürnberg
{andreas.wahl,peter.schwab,richard.lenz}@fau.de

**Abstract.** Data scientists follow individually established workflows and use customized tool chains to deal with complex data analysis scenarios. Novel tools, which aim to support their work, must not disrupt these proven technical environments and must integrate with their existing tool chains in order to further enhance their productivity.
Augmenting data science workflows usually requires to explicitly load data sources into additional tools where they are processed in isolation and exported again for further usage. There is currently no evolvable mechanism to augment data science workflows with additional functionality without mandatory workflow disruption.
To overcome this problem, we introduce a proxy driver for augmenting data science workflows. Our driver provides proxies of Java Database Connectivity (JDBC) objects and is easily evolvable through a plugin system. Driver instances can be synced with a remote plugin repository. Currently, two different driver plugins enable query-driven data ingestion as well as query-driven data profiling.



## 1 Introduction

Data science workflows usually consist of multiple consecutive steps, including data discovery, data exploration, data cleaning, model development, model evaluation and result visualization. Data scientists use specialized tools for each of these steps to overcome different challenges. They assemble customized tool chains, in which data sources are passed along between these tools. Augmenting such workflows currently requires integrating additional tools into these tool chains, which can be time-consuming. However, many data science tools use standardized database mechanisms such as Java Database Connectivity (JDBC) or Open Database Connectivity (ODBC) to connect to the underlying data sources. Hence, we argue that the database driver is a suitable extension point to introduce additional functionality into data science workflows without requiring significant workflow disruption.

***Contribution*** In this paper, we present the fundamentals of a JDBC driver for augmenting data science workflows. We describe the overall driver architecture and discuss evolvability as well as deployment aspects (Sec. 2). We illustrate the benefits of our approach for data scientists by introducing two driver plugins we have developed (Sec. 3). Our driver will soon be publicly available from `http://www.dormantdata.com`.

## 2 Evolvable Database Driver

Our driver provides proxies for objects of the `Connection`, `Statement` and `ResultSet` classes of the JDBC API and wraps the native driver of the respective data management system (Fig. 1). We use reflection techniques to augment objects with additional logic at run time. This logic is encapsulated in plugins, which are initialized when the proxy driver is loaded. Each plugin defines which specific JDBC methods are intercepted and augmented. For each class, the driver maintains a stack of plugins to enable complex augmentation. Plugins are executed consecutively when calls to relevant JDBC methods are detected.

***Evolvability*** Through our plugin system, JDBC objects can be augmented with multiple layers of arbitrary logic. Plugins can define dependencies to other plugins to enable the reuse of augmentation logic. To facilitate system evolvability, local driver instances are synchronized with a remote plugin repository (Fig. 1). Whenever the proxy driver is loaded, the repository is queried for updated versions of previously used plugins to initiate automatic updates. Users are notified about new plugins to decide whether to incorporate them into their workflow.

***Deployment Effort*** Our driver is compatible with existing tools that use JDBC to connect to underlying data management systems. Migrating to our driver is minimally-intrusive: Users deploy the driver binary to their machine and point their analysis tool to this binary (Fig. 1). This is done by adding the prefix `jdbc:dormantdata:` to the connection string of the native driver.

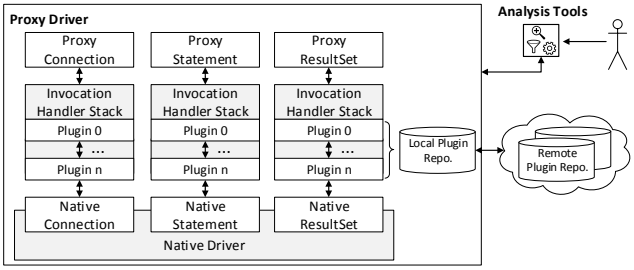


**Fig. 1.** System Architecture

## 3 Driver Plugins

We have developed two plugins that provide useful functionality for data scientists without interfering with established analysis tools and workflows.

### 3.1 Query-Driven Data Ingestion

The first plugin enables the automated ingestion of remote data sources referenced in SQL queries. Data scientists can directly use the URL of data sources, such as CSV files or HTML tables in the `FROM`-clauses of their queries. Our evaluation indicates that they can work more efficiently with our plugin, as they are able to perform data preparation tasks directly in the context of their actual queries without having to use external ETL tools [5].

***Plugin Logic*** The plugin intercepts calls to methods that execute SQL statements (Fig. 2). After logging an intercepted query, it is parsed to extract all references to data sources. Whenever an unknown or outdated data source is detected, it is downloaded from its respective URL, transformed and stored in the data management system. The query is subsequently rewritten to reference the stored copy of the remote data source.

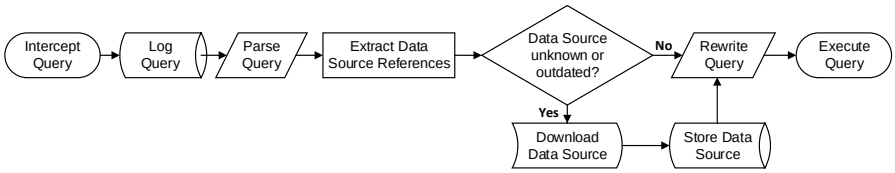


**Fig. 2.** Query-Driven Data Ingestion: Plugin Logic

### 3.2 Query-Driven Data Profiling

Through a second plugin, we provide support for analyzing query results in the context of previous queries of the respective session [12]. We therefore integrate data profiling algorithms into query processing. Among others, these algorithms can derive basic statistics about the query results, find unique column combinations and discover different types of constraints valid for the query results. The plugin provides aggregated visualizations of the profiling results of the respective session as well as measures to rank results according to their usefulness. This allows data scientists to conduct targeted data exploration in the context of their actual queries as well as plausibility assessment of intermediate query results.

***Plugin Logic*** The plugin intercepts calls to methods that execute statements and retrieve result rows from the underlying data management system. Parallel to query processing, result rows are streamed to a remote server to avoid impact of computationally expensive profiling on normal query processing. Result rows are fed into an instance of the Metanome data profiling system [10]. We execute multiple data profiling algorithms on the result rows in parallel to generate a selection of different data profiles. After ranking the data profiles according to their deviation from the profiles of previous queries, the plugin launches a companion dashboard next to the analysis tool from which queries are formulated. The current query is displayed (Fig. 3; a), along with a visualization of the previous query flow (Fig. 3; b). Branches indicate output schema changes. Data scientists can freely choose which previous queries are used as reference points during analysis. A query history (Fig. 3; c) allows inspecting previous profiling results and visualizations. Suitable visualizations for the historic and current results of each profiling algorithm are displayed (Fig. 3; d-i).

## 4 State of the Art

In contrast to our approach, the technique of using a proxy driver has previously only been used to add non-functional aspects to existing data management systems and not for data science workflow augmentation. Among others, these aspects include virtual clustering [4], transparent multi-tenancy [8], caching techniques [7], virtual data integration [6] and security mechanisms [9]. System evolvabilty is not addressed by previous efforts.

There are several publicly available implementations of JDBC proxy drivers. In contrast to our work, they either do not support all relevant parts of the API (e.g. [1]) or target specific purposes such as logging (e.g. [3]) or virtual clustering (e.g. [2]). These implementations do also not address system evolvability.

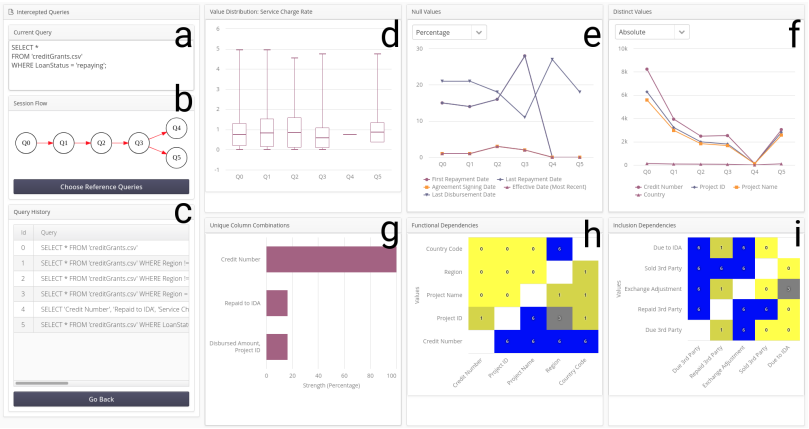


**Fig. 3.** Query-Driven Data Profiling: Companion Dashboard (Screenshot) [12]

## 5 Conclusion and Future Work

Our JDBC proxy driver provides a minimally-intrusive way to augment data science workflows with additional functionality. The currently implemented driver plugins support data scientists with integrating and exploring unfamiliar data sources. Our driver can also be a valuable tool for other researchers who are monitoring existing data science workflows.

We are currently extending our work by developing additional driver plugins. These will provide functionality from existing research projects (cf. Sec. 4) as well as novel context-sensitive auto-completion for SQL queries based on knowledge mined from existing query logs (cf. [11]).

## References

1. DSProxy, https://github.com/dapphub/ds-proxy, accessed: 2018-05-29
2. HA-JDBC, http://ha-jdbc.org/, accessed: 2018-05-29
3. P6Spy, https://github.com/p6spy/p6spy, accessed: 2018-05-29
4. Cecchet et al.: C-JDBC: Flexible Database Clustering Middleware. In: ATEC '04
5. Haller: Tsunami - Anfrage-getriebene Anbindung von Datenquellen an ein Datenmanagementsystem. In: INFORMATIK'17 (2017)
6. Lawrence: Integration and virtualization of relational sql and nosql systems including mysql and mongodb. In: CSCI'14 (2014)
7. Lawrence et al.: Next Generation JDBC Database Drivers for Performance, Transparent Caching, Load Balancing, and Scale-out. In: SAC '17 (2017)
8. Ma et al.: A Transparent Data Middleware in Support of Multi-tenancy. In: NWeSP'11 (2011)
9. Mitropoulos, Spinellis: SDriver: Location-Specific Signatures Prevent SQL Injection Attacks. Computers and Security **28** (2009)
10. Papenbrock et al.: Data Profiling with Metanome. PVLDB **8**(12) (Aug 2015)
11. Wahl et al.: Query-Driven Knowledge-Sharing for Data Integration and Collaborative Data Science. In: ADBIS'17 (2017)
12. Wahl et al.: Query-Driven Data Profiling with OCEANProfile. In: BIRTE'18 (2018)