Well Log DB

Written Report

Survey / Exploratory Group 4

Zuhair Mehdee
XiaoFeng Kuang
Riley Flynn
Daniel Power

Background

In the world of oil and gas research, the LAS file format is ubiquitous. The format is very flexible, specifying very few required fields [12], resulting in a large amount of variations & inconsistencies in the data within from company to company, and even operator to operator. The particular inconsistency we aim to solve is the inconsistencies of curve parameter mnemonics,

This topic is an instance of a topic known as schema matching. At the core, the inconsistencies within LAS files effectively result in the files having disparate schemas, and the problem of schema matching focuses on identifying matching fields within distinct schemas [13] - exactly the problem we aim to solve. More specifically, within the area of schema matching our problem is an instance of semantic heterogeneity, which occurs when the same underlying, real-world data is referred to in multiple distinct ways [4]. It is most commonly caused by distinct schemas being designed for their own purposes, but in our case it is equally as applicable to the problem of the inconsistent parameters names from one well log file to another.

As was identified by Hong-Hai Do in his thesis [9] and we mentioned in our proposal [11], this is a problem that occurs frequently in the application of database systems. In many real-world applications, data is produced without any consideration for its future use in a database, resulting in the need to perform some form of mapping to enable the integration of the data with a final schema. This leads to schema matching being a broadly applicable topic in any use of databases where existing data or other schemas are being integrated.

In Amit Sheth's report on the history of the topic [21], he identifies that concrete evidence of history on the initial advent of the topic is not easily accessible, but he believes that the topic first began being discussed in 1981 when Fèlix Saltor identified the concept of different "worlds" in databases, first acknowledging the real world schema and the conceptual schema as being separate things needing integration.

In 2001, Erhard Rahm, along with Philip A. Bernstein, performed a survey of the state of schema matching methods as existed at the time [20]. In it, they identify a core problem with the state of schema matching: while a number of papers had been written on the topic in the intervening 20 years, they were each focused on their own specific schema and did not provide methods that were generally applicable. For example, a number of researchers at Stanford built SKAT, which allowed authoring rules in first-order logic for expressing matches [18], a number of researchers at Tel Aviv University built TransScm which uses a graph-based approach [17], and Erhard Rahm and Philip Bernstein, along with Jayant Madhavan, contributed to Cupid at Microsoft, which used auxiliary information such as synonyms to extract information from XML [14]. This disparate knowledge left schema matching as a typically entirely manual process, requiring a person to define mapping rules by hand. Seeing this problem, they enumerated all the distinct tools for mapping used by all prior research, and used it to provide a comprehensive set of methods.

Armed with this knowledge Erhard teamed up with Hong-Hai Do, in addition to several other collaborators, and built COMA [8]. Released in 2002, COMA provided a catalog of the matching methods Erhard's prior work found and enabled the easy development of matching processes

targeted for your specific data using them. Next, they released COMA++ in 2005 [2] which added a GUI and the ability to correct incorrect matches, making COMA an even more powerful tool. Finally, in 2011, COMA 3.0 was released [15]. It improved on COMA++ in a number of ways, but the most notable was adding an API, making COMA truly suited to be used within an application for addressing schema matching needs.

In 2011, Erhard would collaborate with many other researchers in the field and release “Schema Matching and Mapping” [3]. This book aimed to be a comprehensive guide on the topic - bringing together and refining the disparate research further and acting as a thorough core text on the field.

Framework

There is no universal, stable, enumerable set of well log curve parameters. The LAS specification explicitly allows arbitrary values (called “parameters”) to be recorded and as such, the exact schema is left up to companies, researchers, and tools [12].

From the perspective of database theory, this variability has direct implications for relational schema design. Decomposition (the process of breaking a larger schema up into smaller components to reduce data duplication) requires a closed set of attributes and a corresponding set of functional dependencies (rules enabling reaching one attribute from another) that must hold for all future instances of the relation [16]. In the case of well logs, it is impossible to create a schema that is capable of accommodating new LAS well log files without modifying the schema in some capacity.

Each new LAS file may introduce new curve names. For example, starting with a relation $R(\text{well\_id}, \text{log\_id}, \text{depth}, \text{gamma})$. A newly ingested file may introduce a previously unseen curve such as neutron, forcing an expansion to: $R(\text{well\_id}, \text{log\_id}, \text{depth}, \text{gamma}, \text{neutron})$. Because the new attribute is unconstrained by any preexisting functional dependency, this expansion invalidates any prior decomposition. The relational model cannot be guaranteed to maintain dependency preservation when the attribute set itself is not stable [16]. This makes representing all values with a fixed-column relational model unsuitable.

To address this, we adopt an Entity-Attribute-Value (EAV) pattern for storing these dynamic values. In this pattern, values are stored in a table comprised of three columns - a foreign key to a parent entity, a foreign key to the attribute being described, and the value for the given attribute [7]. This structure naturally supports arbitrary numbers of parameters without schema changes and aligns with approaches used in other domains with highly extensible attribute sets [19]. In our implementation of this design, the parent entity is the well log the value was recorded in, the attribute is the metadata for the value being described, and the value is the value recorded.

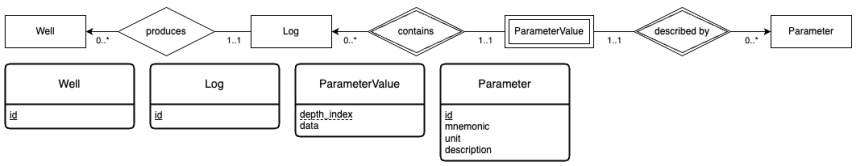

Figure 1: An ER diagram for the core EAV representation. The fields shown for well and log are illustrative rather than exhaustive; the objective is to demonstrate structural approach, not enumerate all possible LAS metadata.

To standardize mnemonics, the identifiers used for parameters [12], we incorporate an auxiliary mapping table that represents known correspondences between variant names and a chosen canonical parameter [22]. This aligns with techniques used by previous research on this topic, such as PyLogFinder, which uses an Excel spreadsheet for this purpose [1]. This also enables us to

leverage a greater variety of techniques, as some depend on knowing past matching results to determine future ones [20].

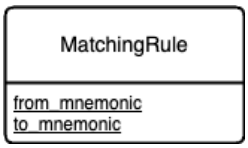

Figure 2: An ER diagram for the schema matching auxiliary table.

During ingestion, if a mnemonic in the LAS file matches an alias in the table, the system will resolve it to the canonical parameter. This ensures that both old and new parameters conform to the same standardized vocabulary and reduces the duplicated effort.

To populate the alias table, we will use each of the automated schema-matching techniques we aim to test. Tools such as COMA can generate candidate matches between observed mnemonics and canonical parameters (e.g., GM → gamma, GR → gamma, DEPTH → depth) [2]. These matches can be human-validated and inserted into the auxiliary mapping table.

In the event a schema-matching technique requires additional auxiliary information, we will use additional purpose-built tables to contain this data. The simplest example is the legacy manual approach for mapping where a user must manually define mapping rules [20], in which case Figure 2 above is a suitable example.

Methodology

We aim to design and evaluate a repeatable process for normalizing disparate LAS curve mnemonics into a canonical set of mnemonics suitable for use in a relational database schema. Rather than defining the entire schema up front, we will iteratively ingest logs, and create mapping rules as we see new mnemonics such that all input mnemonics map to some canonical mnemonic. This iterative approach allows us to build a comprehensive mapping table that grows organically with the dataset while maintaining consistency across the database.

To evaluate different schema mapping methods for this purpose, we aim to perform a small-scale test with a variety of potentially-applicable methods. We will begin by implementing a script using Puppeteer [6] to traverse the C-NLOER well log inventory [5] and download well log LAS files. From there, we will select a subset of these files and parse those files with las-js [10]. We will extract curve mnemonics, units, and descriptions (if available) to produce an initial raw dataset. Duplicates are expected at this point.

Given this dataset, we will remove duplicates and manually build a canonical mnemonic list, and a reference table (`source_mnemonic -> canonical_mnemonic`). This matching table will serve as the gold standard against which our automated schema matching methods will be compared. To aid with reducing the work required for this manual processing, we will evaluate the existing dataset provided by the PyLogFinder project [1], to determine if we can leverage their work.

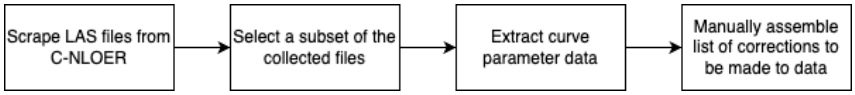

Figure 3: The process for preparing our target dataset.

With a list of candidate data to be corrected identified, we will select a list of candidate schema matching approaches we will evaluate based on a review of the existing literature, such as from "Schema Matching and Mapping" [3] - the primary text on this topic, Erhard Rahm and Philip A. Bernstein's survey of schema matching methods [20], and Hong-Hai Do's thesis [9]. From an initial

review, some likely candidates include schema-based matching using linguistic analysis of mnemonics and descriptions (Levenshtein distance, N-Gram, SoundEx) [9]. Additionally, if time permits, we may test a large-language-model-based approach for comparison.

For each matching method, we will create a fresh database, ingest sampled logs while applying the matching method to map input mnemonics to canonical mnemonic names. This produces a set of databases, one per method, each containing the same underlying log data but with potentially different mnemonic mappings. We will then evaluate each method's performance by comparing its generated mappings against our gold standard reference table, measuring precision and recall [9] to determine which approach most accurately identifies the correct canonical mnemonics. We will also record the time needed for each matching method, as well as any subjective notes on our experience implementing the method. In the case of a tie between methods, these details can help guide us towards which method is better.

For the database we have chosen to use PostgreSQL as our database management system. This choice is based on its powerful geospatial data capabilities via the PostGIS extension, which make it well suited for well log data containing location information. PostgreSQL has previously been used successfully in projects mapping well log data onto a relational database (such as "A cloud-based Well Log Database Prototype" [23]), confirming it as a viable choice for this type of project.

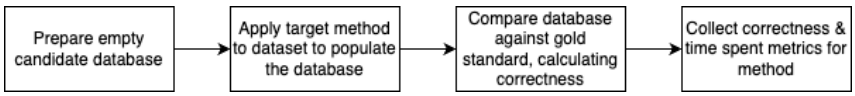

Figure 4: The process for evaluating a given method. This will be performed for each candidate method.

Once we have tested each method and collected metrics, we will compare them and attempt to come to a conclusion on which method or methods are best suited for this task, as well as identifying any caveats associated with implementing them for this kind of work.

Findings

We tested and compared three types of linguistic matching approaches (Levenshtein distance, N-Gram, SoundEx) and their description-based variants. These three approaches each capture different textual characteristics of pattern elements, allowing us to examine how edit-based, structural, and phonological similarities influence matching performance [9]. These methods were chosen as the LAS well log data is uniquely unsuited to other, more sophisticated methods. Methods like Microsoft's Cupid [14] rely on things like analyzing synonyms, which are not applicable due to curve mnemonics being short identifiers and not necessarily words at all, and structure based methods like TransScm [17] or SKAT [18] rely on analyzing the structure of the data and it's schema, but in the well logs the parameters are all in a flat list, so no conclusions can be inferred from them. We tested each method against a gold standard list of 354 canonical mnemonics, which is used as the benchmark for the best performance results that matching methods can achieve.

For mnemonic-based methods, N-Gram is the best performing matching approach, it identified 321 correct results. Levenshtein distance was slightly less accurate than N-Gram, which is 311 correct results. SoundEx produced the lowest correct accuracy, 227 correct results. Therefore, phonetic similarity algorithm is insufficient for mnemonic matching.

Description-based methods have less accuracy than mnemonic-based ones. The correct mnemonics of Description N-Gram decreased to 297, and Description Levenshtein decreased to 265. The Description Soundex method has the worst performance, which produced only 131 correct matches.

This steady decrease across all approaches indicates that descriptions introduce additional variability, ambiguity, and noise compared to mnemonics.

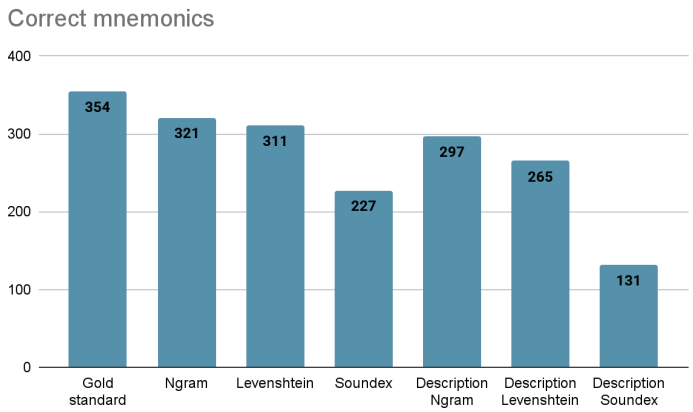

Figure 5: Number of correct canonical mnemonics identified by each matching approach.

For computation time, mnemonic-based matching approaches are faster than description-based ones. SoundEx completed in 8.95 seconds, Levenshtein in 9.36 seconds, and N-Gram in 15.25 seconds. However, description-based approaches have more higher computational cost. Description N-Gram (33.41s) and Description Levenshtein (27.39s) take more than twice the time of their mnemonic-based variants. This confirms description-based approaches are not only less accurate but also more computationally expensive to process.

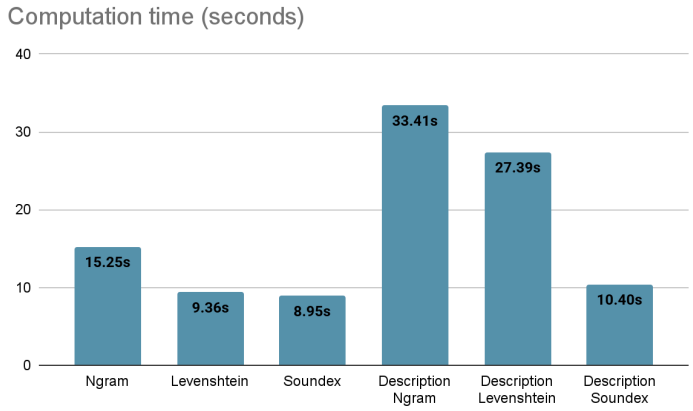

Figure 6: Execution time required for each matching approach.

Above all, mnemonic-based N-gram is the most effective automated method because of its high matching accuracy and reasonable computation time. However, none of the approaches achieved the perfect accuracy, which means a manual review process remains necessary. This leads to an optimal workflow: automated matching first generates candidate results, followed by a human reviewer confirms and corrects any incorrect matches.

Discussion

Through this project, we have confirmed that schema matching is a way to address well log data inconsistencies. Specifically, the mnemonic-based N-gram method can significantly reduce the manual workload required for well log data standardization. However, this project has several limitations and requires further investigation.

One limitation is that the gold standard is artificially generated. Because the artificial generation process introduces the possibility of human error, the gold standard itself may not be entirely authoritative. Meanwhile, considering the amount of work required to manually construct a complete gold standard, our gold standard is generated based on samples from the complete dataset. This means that the observed performance metrics may not generalize to the entire dataset. Different subsets of mnemonics may produce different accuracy rankings.

Another limitation is semantic ambiguity of mnemonic. We assume a one-to-one correspondence mapping between a mnemonic and its canonical form. However, a mnemonic may represent different measurements or parameters depending on operator or company. If the contextual meaning is ignored, a fully automatic matching system may reinforce incorrect mapping relationships.

In conclusion, full automation cannot fully address this problem now. Human intervention is still required to verify results and resolve ambiguous cases. However, automated methods can accelerate data cleaning by generating reasonable matching results, thereby reducing manual workload.

Reflections

Zuhair Mehdee

I was very happy to work on this project, knowing I was working on a full-scale real-world problem and all the messiness that entails, instead of a contrived toy project. That made the work feel grounded and motivating. In this project I worked mainly on the data pipeline, and I also wrote the framework. I cleaned the well metadata from the CNLOER Excel file and removed duplicates according to the project rules. I wrote a scraper to download all the LAS files from the CNLOER website. I also designed an initial database and proposed using an EAV structure, since the parameters in LAS files vary widely between wells. I wrote the code to parse the LAS files, load the data into the database, and extract random parameter samples for testing our schema-matching approaches.

Through this work I learned a lot about how LAS files are structured, how inconsistent real-world geological data can be, and how to design a schema that can handle large and variable datasets. During my research, I also learned about time-series database concepts and how similar ideas can apply to depth-indexed data.

I encountered many issues related to the scale of the data that forced me to be very conscious of writing performant code. I often ran out of memory or faced untenably slow progress when implementing naive versions of the code. By necessity, I implemented more informative logging to monitor progress and track errors, ensuring the code ran as expected and failed where it should.

Overall, the experience working on this project was very educational and rewarding, and I would love to continue working on it after the course is over to meet the remaining requirements and create a truly useful tool.

XiaoFeng Kuang

My contributions to this project were: combined the group's insights, observations, and experimental results to write the Findings and Discussion sections of the final report and manually creating the Gold Standard dataset.

The challenge I faced in this project was creating a standardized, gold-standard database from unstandardized data. In these data, different companies and operators used different mnemonics to represent the same measurement metric, or used the same mnemonic to refer to different measurement metrics. This resulted in the only feasible way to manually create a standardized canonical mnemonic table by comparing each mnemonic description one by one.

Previously, I had no practical experience in building databases for real-world problem. Through this project, I gained a deeper understanding of how to design and implement databases in real-world scenarios. Additionally, through team discussions and collaboration, I also gained a more profound understanding of the entire database construction process.

In the future, I plan to further study database theory and best practices, and use this project experience as a starting point to participate in more data platform construction projects, continuously improving my ability to solve practical data problems.

Riley Flynn

My specific contributions to the project were: authoring the proposal and background deliverables (including addressing feedback for the inclusion of the background in this final written report), creating the diagrams used for the framework and methodology deliverables, providing an outline to use for the framework deliverable, and selecting and implementing the different schema matching methods, in addition to the initial research and proofreading of each other's work we all did.

As someone who has an interest in data science / OSINT (Open-Source Intelligence) and has worked on personal projects in the past surrounding taking existing data and reworking it to be more easily workable (including taking data not intended to be stored in a database and reworking it to fit into a database), this project was of particular interest to me. I hoped that this project would provide me with broadly applicable methods & tools that I could take outside of this class and apply to my own, future projects. I was disappointed when I discovered that due to the structure of the LAS files and the specific inconsistencies they were encountering, that the more interesting schema matching methods would not be applicable and that we'd only be able to apply the more basic, straightforward string-based methods to this topic.

Regardless, I found the assignment interesting and informative to work on. I had never been introduced to the field of schema matching / schema mapping in the fast, and have largely done similar things in the past solely via brute force. It had never occurred to me that such a thing would be a broader topic, with defined methods & research for doing this. Even though I unfortunately never got a chance to play with any of the more interesting methods here, I expect next time I run into a similar problem in the future rather than trying to do everything manually I'll be reaching for Hong-Hai Do's thesis [9] and Erhard Rahm's text [3] and seeing if I can leverage any of their work.

Daniel Power

My direct responsibilities were to write the methodology deliverable, and to produce evaluation results and metrics using the matching methods implemented by Riley against the gold standard mapping table produced by XiaoFeng. This is in addition to the initial research, and proofreading of all deliverables, which were shared responsibilities of all the group members.

I was excited to work on this project because it's rooted in a real-world need for better standardized and freely accessible tooling. From our meeting with Kate, it was clear that the existing tooling is lacking not only in functionality, but also in freedom of use. For example, Kate noted that they can no longer collaborate with researchers located in countries like Iran due to U.S sanctions preventing them from using the software for analyzing well log data. Software freedom is a topic that's important to me, so I was hopeful we could make an impact here.

Through reading Hong-Hai Do's works, I learned that Schema Matching and the broader topic of Data Harmonization are vast areas of study covering many different use-cases. While our project focused on standardizing well log curve mnemonics, principles we explored are applicable far beyond oil and gas data. But I also found that many of the methods discussed on the topic are not

applicable to well log curve mnemonics, as they depend on the schema structure to provide contextual information, while curve mnemonics have little structure.

Bibliography

[1] Adewale Amosu and Hamdi Mahmood. 2018. PyLogFinder: A Python Program for Graphical Geophysical Log Selection. *RIO* (February 2018), 3–12. Retrieved from https://www.proquest.com/scholarly-journals/pylogfinder-python-program-graphical-geophysical/docview/2169976508/se-2

[2] David Aumueller, Hong-Hai Do, Sabine Massmann, and Erhard Rahm. 2005. Schema and ontology matching with COMA++. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data (SIGMOD '05)*, 2005. Association for Computing Machinery, Baltimore, Maryland, 906–908. https://doi.org/10.1145/1066157.1066283

[3] Zohra Bellahsene, Angela Bonifati, and Erhard Rahm (Eds.). 2011. *Schema Matching and Mapping*. Springer Berlin, Heidelberg, Germany.

[4] Yaser Bishr. 1998. Overcoming the semantic and other barriers to GIS interoperability. *International Journal of Geographical Information Science* 12, 4 (1998), 299–314. https://doi.org/10.1080/136588198241806

[5] C-NLOER. Well Data and Information. Retrieved September 26, 2025 from https://home-cnlopb.hub.arcgis.com/pages/well-inventory

[6] The Chromium Authors. Puppeteer. Retrieved November 7, 2025 from https://github.com/puppeteer/puppeteer

[7] Valentin Dinu and Prakash Nadkarni. Guidelines for the effective use of entity-attribute-value modeling for biomedical databases. *International Journal of Medical Informatics* 76, 11, 769–779. https://doi.org/https://doi.org/10.1016/j.ijmedinf.2006.09.023

[8] Hong-Hai Do and Erhard Rahm. 2002. COMA - A System for Flexible Combination of Schema Matching Approaches. In *VLDB*, August 2002. 610–621. https://doi.org/10.1016/B978-155860869-6/50060-3

[9] Hong-Hai Do. 2005. Schema Matching and Mapping-based Data Integration. Doctoral dissertation. Retrieved September 26, 2025 from https://core.ac.uk/download/pdf/226125846.pdf

[10] Ikechukwu Eze. las-js. Retrieved November 7, 2025 from https://github.com/laslibs/las-js

[11] Riley Flynn, XiaoFeng Kuang, Zuhair Mehdee, and Daniel Power. 2025. Well Log DB - Proposal.

[12] Dave Greenwood and Case Struyk. 2017. *LAS Version 2.0: A Digital Standard for Logs*. Canadian Well Logging Society. Retrieved September 26, 2025 from https://web.archive.org/web/20240407183515/https://www.cwls.org/wp-content/uploads/2017/02/Las2_Update_Feb2017.pdf

[13] J. Madhavan, P.A. Bernstein, A. Doan, and A. Halevy. 2005. Corpus-based schema matching. In *21st International Conference on Data Engineering (ICDE'05)*, 2005. 57–68. https://doi.org/10.1109/ICDE.2005.39

[14] Jayant Madhavan, Philip Bernstein, and Erhard Rahm. 2001. Generic Schema Matching with Cupid. *Proc 27th VLDB Conference* (2001), .

[15] Sabine Massmann, Salvatore Raunich, David Aumüller, Patrick Arnold, and Erhard Rahm. 2011. Evolution of the COMA match system. In *Proceedings of the 6th International Conference on Ontology Matching - Volume 814 (OM'11)*, 2011. CEUR-WS.org, Bonn, Germany, 49–60.

[16] Steph McIntyre. COMP 4754_6908 - FDs Part 1. Retrieved October 30, 2025 from https://online.mun.ca/d2l/le/content/636466/viewContent/5892952/View

[17] Tova Milo and Sagit Zohar. 1998. Using Schema Matching to Simplify Heterogeneous Data Translation. In *Proceedings of the 24rd International Conference on Very Large Data Bases (VLDB '98)*, 1998. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 122–133.

[18] P. Mitra, G. Wiederhold, and J. Jannink. 1999. Semi-automatic Integration of Knowledge Sources. In *2nd International Conference on Information Fusion (FUSION 1999)*, 1999. Retrieved from http://ilpubs.stanford.edu:8090/384/

[19] Prakash M. Nadkarni, Luis Marenco, Roland Chen, Emmanouil Skoufos, Gordon Shepherd, and Perry Miller. 1999. Organization of Heterogeneous Scientific Data Using the EAV/CR Representation. *Journal of the American Medical Informatics Association* 6, 6 (1999), 478–493. https://doi.org/10.1136/jamia.1999.0060478

[20] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal* 10, 4, 334–350. https://doi.org/10.1007/s007780100057

[21] Amit P. Sheth. 2004. *Early Work in Database Research on Schema Mapping/Merging/ Transformation, Semantic Heterogeneity, and Use of Ontology and Description Logics for Schematic and Semantic Integration.* Retrieved from https://corescholar.libraries.wright.edu/knoesis/25/

[22] Serena Sorrentino, Sonia Bergamaschi, Maciej Gawinecki, and Laura Po. 2010. Schema label normalization for improving schema matching. *Data & Knowledge Engineering* 69, 12 (2010), 1254–1273. https://doi.org/https://doi.org/10.1016/j.datak.2010.10.004

[23] Chitra Viswanathan, Irina Emelyanova, Ben Clennell, and Stacy Maslin. 2018. A cloud-based Well Log Database Prototype. *ASEG Extended Abstracts* 2018, 1 (2018), 1–4. https://doi.org/10.1071/ASEG2018abP065