DataMosaic: An Interactive Demonstration of Constraint-Driven Document-to-Database Construction

Zhengxuan Zhang
HKUST(GZ)
Guangzhou, China
zzhang393@connect.hkust-gz.edu.cn

Zhuowen Liang
HKUST(GZ)
Guangzhou, China
zliang321@connect.hkust-gz.edu.cn

Haixun Wang
EvenUp
Seattle, USA
haixun@gmail.com

Nan Tang*
HKUST(GZ)/HKUST
Guangzhou, China
nantang@hkust-gz.edu.cn

ABSTRACT

Large language models (LLMs) have improved document information extraction, but turning extracted facts into relational databases remains fundamentally difficult. The challenge is that extraction is local to text, whereas database construction must satisfy global semantics defined by schemas, keys, and integrity constraints. Consequently, single-pass LLM extraction often produces inconsistent tuples, missing links, and semantically invalid results. We present **DataMosaic**, an interactive system for constraint-driven document-to-database construction. Given raw documents together with user-specified schemas and constraints, **DataMosaic** incrementally builds entity and relationship tables through a closed loop of extraction, verification, repair, and targeted re-extraction. The system exposes intermediate table snapshots, detected violations, and repair actions, enabling users to audit how database-level feedback improves the final relational output. In the demo, attendees can configure inputs and constraints, inspect real-time analysis logs, and observe how **DataMosaic** repairs inconsistencies and materializes query-ready relational tables. The demonstration highlights how integrating extraction with verification and repair enables reliable document-to-database construction.

PVLDB Reference Format:
Zhengxuan Zhang, Zhuowen Liang, Haixun Wang, and Nan Tang. DataMosaic: An Interactive Demonstration of Constraint-Driven Document-to-Database Construction. PVLDB, 19(12): 4570 - 4573, 2026.
doi:10.14778/3827998.3828068

PVLDB Artifact Availability:
The source code, data, demo video and/or other artifacts have been made available at https://github.com/HKUSTDial/DataMosaic.

1 INTRODUCTION

The LLM era has reopened an old dream in data management: turning messy documents into relational databases [4]. Yet LLMs alone do not solve this problem—they expose why database ideas matter more than ever. Extraction is *local*: a model reads spans, sentences, and chunks [2]. Databases are *global*: they enforce schemas, keys, relationships, and integrity constraints over an entire instance. A one-shot extractor [1] may emit plausible tuples, but still fail to produce a database that is consistent, auditable, and query-ready.

This tension motivates the problem of **document-to-database (Doc2DB)** construction. Given a collection of documents, a target relational schema, and a set of integrity and business constraints, the goal of Doc2DB is to automatically construct a structured relational database instance that conforms to the schema while satisfying the specified constraints. In other words, Doc2DB transforms unstructured or semi-structured document collections into a consistent and queryable relational database.

Crucially, treating Doc2DB as pure extraction collapses global semantics into local decisions—producing tables [5], but not databases. Entity identities and keys are typically absent from text, relationships must be assembled across documents [6], and global constraints must hold over the entire database instance. Relationship extraction is even harder than entity extraction: tuples must be grounded on canonical entities, their foreign keys do not explicitly appear in text, and valid edges may be scattered across documents or only become visible under global constraints.

Example 1. *[Local plausibility is not global validity.] Consider simplified document fragments below:*

Blue denotes entity mentions; purple denotes attribute values; red underlined values indicate sources of errors

Doc A: *"Arena Tech reported cash increasing from 9.867M to 11.2M in fiscal year 2023. Alpha Holdings invested in Arena Tech."*
Doc B: *"Alpha Holdings filed its 2024 report. Cash began at 18,524 (in thousand) and ended at 32.368M."*
Doc C: *"Arena Tech invested in Coyni Corp. Coyni Corp reported cash declining from 4.544M to 4.062M in 2023. Elsewhere, Coyni Inc is mentioned as an affiliate company."*

Suppose the target schema contains an entity table `Company(cid, name, year, cash_begin, cash_end)` *and a self-referencing relationship table* `Hold(investor_id, investee_id)`*, where the surrogate key* `cid` *is system-generated and does not appear in text. Assume the following business rules: (i) no mutual investment, (ii) recursive investment—if* `Hold(A,B)` *and* `Hold(B,C)` *hold, then* `Hold(A,C)` *should also hold, and (iii)* `cash_end / cash_begin < 10`*.*

A one-shot extractor may produce:

Company	*Year*	*$Cash_b$*	*$Cash_e$*
Arena Tech	*2023*	*9.867M*	*11.2M*
Alpha Holdings	*2024*	*18,524*	*32.368M*
Coyni Corp	*2023*	*4.544M*	*4.062M*
Coyni Inc	*2023*	*NULL*	*NULL*

Extracted Company table

Investor	*Investee*
Alpha	*Arena*
Arena	*Alpha*
Arena	*Coyni*

Extracted Hold table

*Nan Tang is the corresponding author.

Proceedings of the VLDB Endowment, Vol. 19, No. 12 ISSN 2150-8097.
doi:10.14778/3827998.3828068

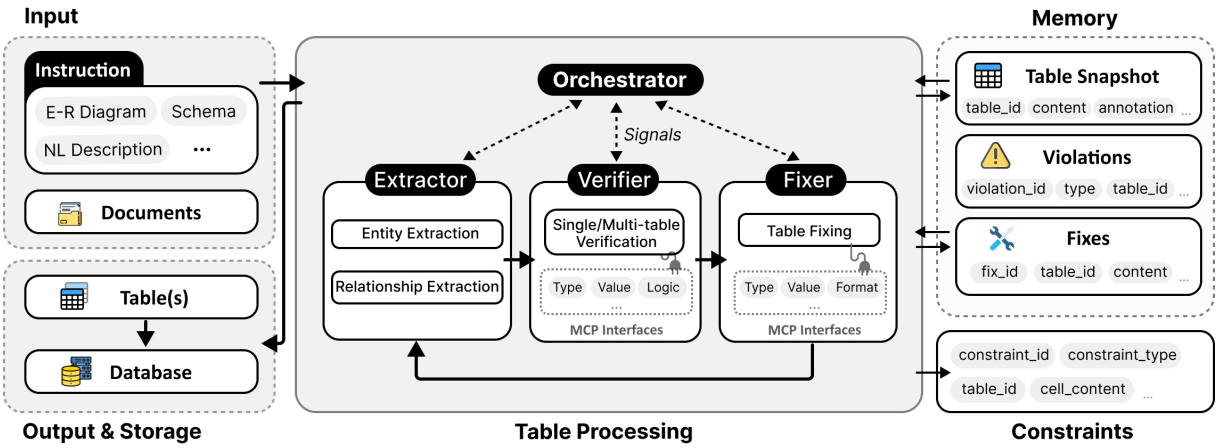

Figure 1: Overview of DataMosaic.

Each tuple looks plausible locally; together, they do not form a database. □

Example 1 exposes key challenges in Doc2DB:

- ***Document–schema mismatch:*** documents are written as narratives, while databases require normalized entities and relationships; evidence for a single tuple may be scattered across multiple passages or documents.
- ***Things, not strings:*** databases operate on canonical entities and keys, but documents provide only surface mentions, requiring alias resolution and entity consolidation before reliable keys can be established.
- ***Relationships are harder:*** relationship tuples must connect canonical entities whose foreign keys do not explicitly appear in text; edges are implicit, often cross-document, and may only become visible under global constraints.
- ***No one-shot guarantees:*** even when individual rows look locally plausible, the full instance may still violate uniqueness, referential integrity, or business rules, requiring iterative verification and refinement.

This is where database semantics matter: schemas and constraints are not just filters—they are **feedback signals**. **DataMosaic** closes the loop with an *extract* → *verify* → *iterate* workflow. Instead of a single pass, it verifies extractions against constraints Σ, repairs identity conflicts, and re-queries the LLM to fix specific violations. The result is a consistent, globally-validated database instance rather than a collection of noisy tables.

Interactive Demonstration. This demonstration is not a passive showcase of extracted tables—it is a live demonstration of how database semantics can actively improve LLM-based document understanding. The demo makes visible the central insight of Doc2DB: in the LLM era, reliable document understanding does not stop at extraction time—it must remain continuously guided by database semantics, including schemas, keys, integrity constraints, and business rules.

What makes the demonstration particularly interesting is that the database evolves on screen. As **DataMosaic** extracts entities and relationships, attendees can observe table snapshots, detected violations, repair actions, and warm-start re-extraction targeted only at problematic cells or links. Rather than re-running the entire pipeline from scratch, the system incrementally refines the database while preserving trustworthy content, making the closed-loop nature of **DataMosaic** transparent and easy to explore.

Importantly, this demonstration is grounded in substantial algorithmic and systems work. Our underlying extraction framework enables accurate and efficient structured extraction from long documents [3], while the full **DataMosaic** system formalizes Doc2DB as a closed-loop database construction problem with verification and repair [6].

Section 3 turns this idea into a live experience: attendees first build an entity table, then connect tables through relationships, and finally watch the database repair itself under constraints.

2 PROBLEM AND SYSTEM OVERVIEW

The Doc2DB Problem. Given a document collection F, a target schema E, and a set of integrity and business constraints Σ, the goal is to construct a relational database instance D such that:

$$(F, E, \Sigma) \xrightarrow{\text{Doc2DB}} D,$$

where D populates all tables in E and satisfies Σ.

Unlike text-to-table extraction, Doc2DB is a database construction problem: entity identities must be resolved, relationships must be assembled across documents, and global constraints must hold over the entire instance.

The DataMosaic Framework. **DataMosaic** addresses Doc2DB through a closed-loop, constraint-driven workflow, as depicted in Figure 1:

$$\textit{extract} \rightarrow \textit{verify} \rightarrow \textit{fix} \rightarrow \textit{re-extract}.$$

Given (F, E, Σ), the system incrementally constructs the database in three stages:

- **(1) Entity extraction.** Documents are processed to populate entity tables under schema E, consolidating mentions into canonical entities.
- **(2) Relationship construction.** Based on extracted entities, the system identifies inter-entity relationships and materializes relational tables.
- **(3) Constraint-driven refinement.** The current database instance is verified against Σ. Violations trigger either repair actions or targeted re-extraction, iteratively improving consistency.

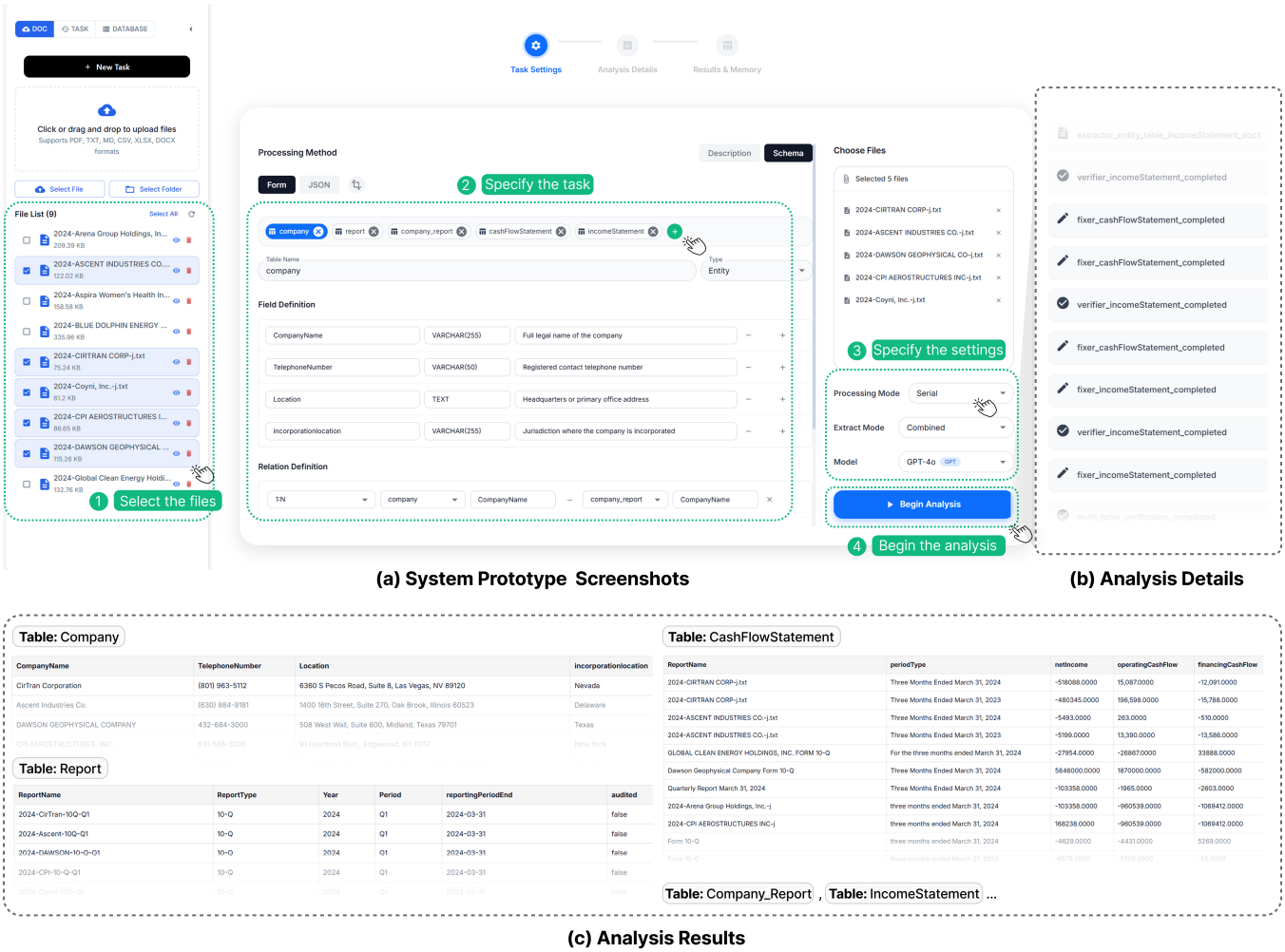

Figure 2: Screenshot of `DataMosaic` system. (a) Users configure the task by selecting files, defining the E-R schema, and setting extraction modes. (b) The Analysis Details panel streams real-time logs from the Verifier and Fixer modules. (c) The constructed, highly-accurate relational tables are materialized for user audit.

In the prototype, these stages are not hidden backend steps: they are surfaced to attendees through live table snapshots, violation reports, and repair logs.

Remark. The framework is modular and extensible, allowing different extraction, verification, and repair strategies to be plugged into the loop.

3 DEMONSTRATION SCENARIOS

We organize the demo as three *hands-on* scenarios over the same real-world financial analytics workload. Rather than watching a canned workflow, attendees actively drive the system through the GUI in Figure 2: they select documents, define schemas and constraints, launch analysis, inspect live logs, and compare intermediate and final tables. The three scenarios progressively enrich the task, moving from local extraction to full database construction.

Scenario 1: Build the First Entity Table

The demo begins with the most immediate interaction: attendees turn a collection of raw corporate filings into a clean entity table. Using the file panel in Figure 2(a), they first select a set of documents (*e.g.,* 10-Q filings) (**Step 1**), then define an entity schema (**Step 2**) such as `Company` with fields like `CompanyName`, `Location`, and `Year`. They can further adjust the extraction settings (**Step 3**), including the processing mode and underlying LLM, before clicking `Begin Analysis` (**Step 4**). This setup makes the task easy to try and vary: attendees can change the schema, add or remove attributes, or switch the model to see how the extracted table responds.

What makes this scenario attractive is the immediacy of the payoff. As soon as the run starts, attendees can watch noisy document content turn into a structured relational table in the results panel (Figure 2(c)). By revising the schema and rerunning the task, attendees can also see how the system adapts to different table definitions, which makes the demo feel less like a fixed showcase and more like a live database construction tool.

This first scenario also communicates an important systems idea in a very accessible way. The result is not merely a flat list of extracted fields; it is the *entity layer* of a database that users can immediately audit and reason about.

Scenario 2: Add Relationships and Watch the Schema Come Alive

In the second scenario, attendees move from isolated entity rows to connected tables. Starting from the entity tables produced in the first scenario, they return to the schema editor (**Step 2**) and add a relation schema, such as `Company_Report`, to link entities. With one more run (**Step 4**), **`DataMosaic`** materializes new relationship tables alongside the entity tables in Figure 2(c), showing how separate rows are woven into a relational structure.

This scenario is engaging because attendees can experiment with how the schema itself changes the shape of the database. By defining different relations and re-running the analysis, they can observe how new semantic links emerge across tables. Instead of seeing only isolated records, the audience begins to see a relational view of the underlying document collection, making them understand how much harder document understanding becomes once the system must recover inter-entity connections.

From a demonstration standpoint, the audience moves from "extracting a table" to "connecting a database", making the prototype feel truly database-centric. This highlights a deeper challenge that many extraction systems hide: recovering the right relationships across documents. By making these links visible and explorable, the demo turns a difficult systems problem into something attendees can directly see, test, and appreciate.

Scenario 3: Watch the Database Repair Itself

The third scenario is the centerpiece of the demo. Here attendees upgrade the task from extraction to full database construction by adding integrity and business constraints to the schema (**Step 2**). Once they launch the run (**Step 4**), **`DataMosaic`** enters its closed-loop *extract–verify–iterate* workflow. The Analysis Details panel in Figure 2(b) then becomes a live window into the system: attendees can watch the `Verifier` flag concrete violations, see the `Fixer` apply targeted repairs, and observe warm-start re-extraction triggered only for problematic cells or links.

What makes this scenario especially compelling is that the database visibly evolves on screen. Initial outputs may look locally plausible, yet still violate keys, referential integrity, or value constraints. Instead of hiding these failures, **`DataMosaic`** surfaces them and incrementally improves the database while preserving already trustworthy content. Attendees can inspect intermediate table states, follow before/after changes, and see how global database semantics feeds back into local document extraction. By the end of the run, the refined entity and relationship tables (Figure 2(c)) can be explored as a standard relational database.

Attendees can vary constraints—for example, a "no mutual investment" rule, a value sanity rule, or a referential-integrity check—and immediately observe the effect. When a unit mismatch or missing implied relationship is detected, **`DataMosaic`** repairs or re-extracts only the affected cells or links.

Summary. Taken together, these three scenarios make the demo both intuitive and interesting: attendees first build a table, then connect tables, and finally watch the database repair itself. This progression highlights the central message of **`DataMosaic`**: the most interesting part of document-to-database construction is not merely extracting tuples, but letting database semantics actively shape and improve document understanding.

4 IMPLEMENTATION

`DataMosaic` is implemented as a lightweight, real-time system that supports interactive document-to-database construction. The design prioritizes responsiveness, modularity, and auditability to enable a smooth demonstration experience.

Frontend & Interaction. The user interface is built using React, TypeScript and Material-UI. It provides an interactive workspace where users can upload documents, define schemas and constraints, launch analysis, and inspect results. Full-duplex WebSocket communication enables real-time streaming of system outputs, including intermediate table snapshots, detected violations, and repair actions, making the closed-loop process visible to the user.

Backend & Execution Engine. The backend is implemented as a Flask-based service that coordinates document processing and model interaction. At its core, an orchestrator manages the extract–verify–iterate workflow in an asynchronous manner, enabling non-blocking execution of extraction, verification, and repair tasks. This design allows the system to incrementally update results and stream them to the frontend without waiting for full pipeline completion.

State & Provenance. All intermediate results are maintained in a centralized in-memory store, which records table snapshots, detected violations, and applied fixes. This design supports fine-grained provenance tracking, allowing users to trace each value back to its source documents and transformation steps. It also enables comparison of intermediate states during the demo.

Extensibility. **`DataMosaic`** is designed to be modular and extensible. Extraction, verification, and repair components are exposed through pluggable interfaces, allowing domain-specific rules and tools to be integrated without modifying the core system. The infrastructure layer provides unified wrappers for document parsing and LLM APIs, making it easy to adapt the system to new domains and data sources.

REFERENCES

[1] Google. 2025. Langextract. https://langextract.io.

[2] Yupan Huang, Tengchao Lv, Lei Cui, Yutong Lu, and Furu Wei. 2022. Layoutlmv3: Pre-training for document ai with unified text and image masking. In *Proceedings of the 30th ACM international conference on multimedia*. 4083–4091.

[3] Zhuowen Liang, Xiaotian Lin, Zhengxuan Zhang, Yuyu Luo, Haixun Wang, and Nan Tang. 2026. Long-Document QA with Chain-of-Structured-Thought and Fine-Tuned SLMs. In *ICLR*.

[4] Yiming Lin, Mawil Hasan, Rohan Kosalge, Alvin Cheung, and Aditya G Parameswaran. 2025. Twix: Automatically reconstructing structured data from templatized documents. *arXiv preprint arXiv:2501.06659* (2025).

[5] Nancy Xin Ru Wang, Douglas Burdick, and Yunyao Li. 2021. TableLab: an interactive table extraction system with adaptive deep learning. In *Companion Proceedings of the 26th International Conference on Intelligent User Interfaces*. 87–89.

[6] Zhengxuan Zhang, Zhuowen Liang, Jiazhuo Chen, Haixun Wang, and Nan Tang. 2026. Document-to-Database: Extraction Meets Relational Semantics. *Proceedings of the VLDB Endowment* 19, 9, 2522–2535.