Transform Table to Database Using Large Language Models

Zezhou Huang
zh2408@columbia.edu
Columbia University

Jia Guo
jg4692@columbia.edu
Columbia University

Eugene Wu
ewu@cs.columbia.edu
DSI, Columbia University

ABSTRACT

To unify source tables, various industries including healthcare, marketing, and government have established standardized target databases. Transforming source tables into these databases, while utilizing automated tools for schema matching and column transformation, remains challenging when combining them end-to-end. This paper proposes a novel framework using Large Language Models (LLMs): we decompose the transformation task following an "overview, zoom-in, zoom-out" pattern. Our experiments indicate a significant improvement in accuracy from 14.35% to 54.78%. We conclude by analyzing the errors and propose further research directions.

VLDB Workshop Reference Format:
Zezhou Huang, Jia Guo, and Eugene Wu. Transform Table to Database Using Large Language Models. VLDB 2024 Workshop: Tabular Data Analysis Workshop (TaDA).

VLDB Workshop Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://cocoon-data-transformation.github.io/page/transform.

1 INTRODUCTION

Source tables are difficult to use due to their varied formats [19]. To address this, industries have established standardized target databases. For instance, in healthcare, the OMOP Common Data Model (CDM)[3] standardizes tables like person, encounter, and payment, and organizes relationships using primary and foreign keys (PK/FK). The OMOP CDM has converted ∼12% of Electronic Medical Records globally, covering >928 million patient records in 41 countries[10]. Similarly, industries such as sales, marketing, and supply chain, as well as healthcare, use similar CDMs [1]. Additionally, government agencies use a specific spending data model for managing financial data [2]. All of these necessitate the transformation from source tables to the standardized target database.

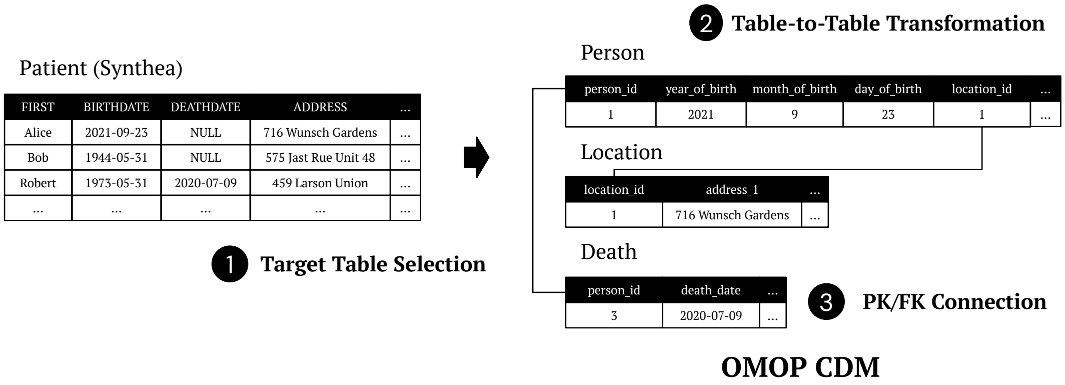
To facilitate such transformation, previous works have automated only parts of the process, which still requires significant manual effort. One class of works is schema matching [5, 8, 20, 22, 26]: these works create mappings between source and target columns based on aspects like column names and cell value similarity. However, matching the columns is just the first step. These columns could have different value representations and span multiple tables that often require extensive coding to transform. Another class of works automates such column transformation, either through an interactive interface [9, 17] or Programming-By-Example (PBE)[11, 16, 27]. While these are effective for columns within single tables, handling columns that span multiple tables poses greater challenges due to the need for joins, selection, aggregation, and groupby. As a result, in domains like healthcare, the de facto integration tool, WhiteRabbit[4], is still entirely manual.

While transforming the source tables to target databases was challenging due to the the variety of complex issues to be addressed, recent advances in Large Language Models (LLMs) make it appealing to revisit these challenges. LLMs exhibit strong few-shot learning capabilities [6]: previous works have demonstrated their SOTA performance in subtasks like schema matching [24] and column transformations [15, 23].

This paper studies the problem of *table-to-database* transformation using LLMs. Our experiment shows that, while current LLMs are already adept at schema matching and column transformations, directly prompting them to transform the source table to the target database end-to-end is still too overwhelming, achieving only 14.35% accuracy. Instead, we build a framework that breaks down the table-to-database transformation; such a process of task decomposition for LLMs has been shown to be critical for the accuracy and robustness of various data tasks like visualization and transformation [7, 15, 18, 24]. Our main insight behind the break-down is that, such transformation follows a "overview, zoom-in, zoom-out" pattern. To illustrate the challenge, consider the example transformation of patient data from Synthea to the OMOP CDM [25].

Example 1. *The Patient table from Synthea [25], as a data source, contains synthetic patient information such as birth/death dates and addresses. Transforming it to the OMOP CDM involves several steps, as illustrated in Figure 1. (1) Overview for Schema Matching: We first select the target tables in the OMOP CDM that contain columns which can be mapped from the source. Here, three tables are selected: Person, Location, and Death. (2) Zoom-in for Table Transformation: For each target table, we transform columns, including extracting the year, month, and day from the birth date, and selecting individuals who have died (where the clause for the death date is not null) for the Death table. (3) Zoom-out for PK/FK: PK/FK (e.g., person_id, location_id) cannot be directly derived from the source table alone but require coordination across other tables in the database for referential integrity. For example, location_id in the Person table depends on the assignment of the primary key in the Location table. Therefore, we track the lineage between records and assign the FK based on how the PK is assigned.*

Proceedings of the VLDB Endowment. ISSN 2150-8097.



Patient (Synthea)

FIRST	BIRTHDATE	DEATHDATE	ADDRESS	...
Alice	2021-09-23	NULL	716 Wunsch Gardens	...
Bob	1944-05-31	NULL	575 Jast Rue Unit 48	...
Robert	1973-05-31	2020-07-09	459 Larson Union	...
...	...	...	...	...

Person

person_id	year_of_birth	month_of_birth	day_of_birth	location_id	...
1	2021	9	23	1	...

Location

location_id	address_1	...
1	716 Wunsch Gardens	...

Death

person_id	death_date	...
3	2020-07-09	...

Figure 1: Table-to-Database Transformation Example, from Patient (Synthea) to OMOP CDM.

We therefore decompose the problem for LLMs according to the following pattern: we start with mapping to the target database as a whole (overview), then delve into each table (zoom-in), and finally establish PK/FK relationships between the target tables (zoom-out), achieving 54.78% accuracy. We further analyze the errors made during these transformations and suggest directions for future improvements.

2 APPROACH OVERVIEW

We start with the problem, and then present our approach.

2.1 Problem definition

Problem 1. *Table-to-Database Transformation. Given an input table T and a target database schema defined as a set of tables $\{T_1, T_2, \ldots, T_n\}$, the task is to find a transformation $\mathcal{F}$ such that $\mathcal{F}(T)$ is correctly transformed to the target database.*

While $\mathcal{F}$ could be an arbitrary mapping function, in this short paper, we limit $\mathcal{F}$ to a list of SQL statements that create tables based on SPJA (Select-Project-Join-Aggregate) queries for simplicity. SQL-based transformations manipulate the table as a whole; however, sometimes row-based manipulation/mapping is needed. For example, for OMOP CDM, we need to standardize medical concepts (e.g., "Emergency Vehicle" to "Ambulance"). Such standardization requires looking up the terms, essentially an entity matching, for each row, which is challenging for SQL [12]. We leave these as future work.

2.2 System Design

To solve Problem 1, directly prompting the LLMs with the source table, target database schema, and the task instruction in one shot yields poor results, as we show in Section 3. Therefore, we have designed a task decomposition layer on top to enhance performance. Our design is illustrated in Figure 2.

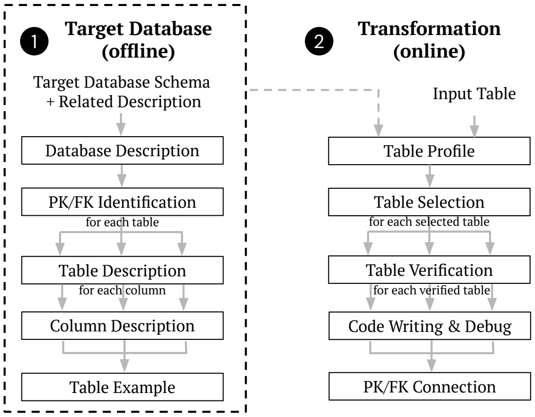


Figure 2: System design for Table-to-Database Transformation. Each box corresponds to a decomposed task.

2.2.1 Target Database (Offline). During the offline phase, we create detailed descriptions for the target database; these are used online during the transformation process for contexts. The input for this phase includes, at a minimum, the schema of the target database tables. This input facilitates the inclusion of contextual descriptions of the target databases for additional context. In this phase, we prompt the LLMs to extract results from the given contextual descriptions if the information is available, or to make the best guess. In practice, this step is intended to be completed once offline and verified by the data provider. Next, we walk through each component:

Database Description. This LLM component provides descriptions of databases and how their tables are related. To ensure the description covers all target tables comprehensively, each table is specifically mentioned and highlighted (enclosed in **). We utilize a Python program to verify that all tables are appropriately enclosed, and we retry if not.

PK/FK Identification. With the database description and the schema of all tables at hand, this LLM component identifies the PK-FK relationships between tables. The output consists of two dictionaries: for each table, dictionary 1 maps to its PK, if it exists, and dictionary 2 maps it to another dictionary, where each key is another table and the value is their FK.

Table Description. Using the database and target schema, this LLM component describes each table. Similar to the database description, to ensure comprehensiveness, each column within the tables is mentioned and highlighted in table description.

Column Description. Given the table description, this component describes each column, and data type (e.g., int, string).

Table Example. Based on the table and its column descriptions, this component provides an example of the target table. The output is a sample table consisting of 5 rows.

2.2.2 Transformation (Online). During the online phase, we transform the input table to the target database using (1) the input table, and (2) target database description prepared offline.

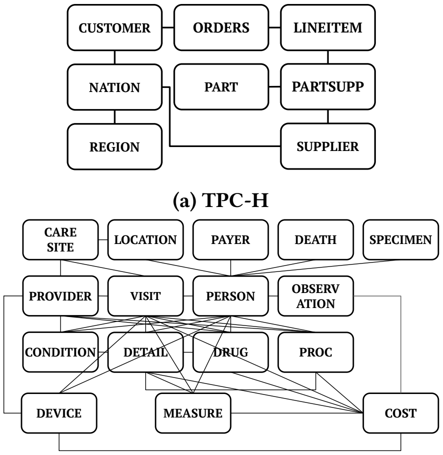
Table Profile. This LLM component profiles the input table using Cocoon [14]. It generates the description of the table, columns, data types and missing values.

Table Selection. Based on the descriptions and samples of the input table and target database, this selects all the relevant target tables. The output is a list of potential target tables.

Table Verification. This component examines each potential target table more closely. Some tables may be relevant but unsuitable for transformation. For example, the patient table in Synthea includes total medical spending, which relates to the cost table in OMOP CDM at a high level but cannot be transformed due to the impossibility of reversing aggregation. For each target table, we provide descriptions and samples for both the input and target tables (excluding PK/FK columns), and ask the LLM to verify whether the transformation is possible (true/false) and provide transformation instructions.

Code Writing and Debug. Following the transformation instructions, this LLM component writes SQL code for the table-to-table transformation. It constructs templated SQL for (a) whether to use distinct (true/false), (b) selection clauses, (c) where clauses, (d) group by clauses, and (e) where clauses. The SQL code is then sent to a debugging component to ensure it runs correctly in the database. It iteratively runs the SQL in the database and debugs based on the error messages, with up to 10 debugging iterations allowed. Note that the code does not execute to create the target table because additional PK and FK columns are not created during this step.

PK/FK Connection. We find that current LLMs fail at handling PK/FK for referential integrity (Section 3); furthermore, after we identify the selected columns for each target table, PK/FK can be directly constructed without the need for semantic understanding by LLMs [19, 22]. We therefore use a non-LLM component to create PK and FK. This process begins by enriching the input table with all additional columns from the selection clauses, excluding aggregation columns. For each target table requiring a PK, the PK is created based on the need for distinct values. If distinct values are required, the PK is generated using a unique value (e.g., MD5 hash) over the enriched attributes in the selection clauses. If distinct values are not necessary, a unique column for each row is created (currently using rowid). If the selection involves aggregation, this implicitly requires distinct values because the group by key is different. The FK is simply selected from the corresponding PK (potentially also with a rename).



(a) TPC-H

(b) OMOP CDM

Figure 3: Target Database Schema. Each edge is a PK/FK.

3 EXPERIMENT

Table 1: Transformation Accuracy over 10 runs from the different source tables. Decomposed method greatly improves accuracy from 14.35% to 54.78%.

Database	Source Table	Direct	Decomposed
SSB	customer	0%	**80%**
	date	**100%**	0%
	lineorder	30%	**70%**
	part	80%	**100%**
	supplier	0%	**70%**
Synthea	allergies	0%	**100%**
	careplans	0%	**30%**
	claims	0%	0%
	claims_transactions	0%	0%
	conditions	0%	**100%**
	devices	0%	**100%**
	encounters	0%	**20%**
	imaging_studies	0%	0%
	immunizations	0%	**20%**
	medications	0%	**10%**
	observations	0%	0%
	organizations	0%	**100%**
	patients	0%	**100%**
	payer_transitions	**100%**	**100%**
	payers	20%	**40%**
	procedures	0%	**70%**
	providers	0%	**100%**
	supplies	0%	**50%**
	Total	**14.35%**	**54.78%**

Table 2: Errors made by Direct for Patient (Synthea) to OMOP CDM Transformation. Direct struggles with syntax errors in date parsing due to the lack of debugging, omitting 'Distinct' in selections, incorrect PK/FK assignments, and fails to include necessary target tables in transformations.

	Table Selection	Distinct Clause	Selection Clause	Where Clause	FK	PK
Person	0%	0%	100%	0%	100%	0%
Death	20%	0%	0%	0%	0%	0%
Location	0%	100%	0%	0%	100%	100%

Benchmarks. We conduct experiments using the following target databases (schema in Figure 3) and source tables:

- **TPC-H**. For the data source, we use tables from the Star Schema Benchmark [21], which is a simplified TPC-H.
- **OMOP CDM** [3]. Note that OMOP CDM has columns for concept IDs in its standardized vocabularies, which require an entity matching problem that is beyond the scope of SPJA queries; we have removed these columns from the target database schema and documentation. For the data source, we use tables from Synthea [25], which has synthetic elecontronic health care records.

Methods. We provide both the target schema and its related documents for the offline phase to generate descriptions for tables, columns, as well as the PK/FK relationships and table examples. Then, we compare two methods for transformation:

- Direct. This method uses the results from the offline phase and directly provides the input table samples to the LLM, asking it to generate the SQL queries in one prompt.
- Decomposed. This method corresponds to the decomposed design discussed in Section 2.2.2.

We use GPT-4 Turbo as the LLM and DuckDB as the database. We inform the LLM about DuckDB in the prompt when writing SQL to ensure the correct syntax. We run each methods 10 times (temperature=0.1), and record the accuracy.

Evaluation. For correctness evaluation, transformation to TPC-H is straightforward, and we verify equality between the transformed target database and the ground truth database. For columns in the target tables that are not transformable, excluding these columns or setting them to NULL are both considered correct. However, for OMOP CDM, many columns have ambiguous meanings [13]; we manually judge if the transformation codes are acceptable. For example, the "value_as_string" column in the OBSERVATION table is described in the documentation as "the categorical value of the result of the observation," but it lacks clarification on which types of results are included and how they should be represented. In the case of the allergies source table, the system constructs "value_as_string" by concatenating the columns of 'Reaction', 'Description', and 'Severity'. While this is not the only choice, it is a reasonable one, and we consider it as correct.

Results. The experiment results are shown in Table 1. Decomposed significantly improves accuracy over Direct from 14.35% to 54.78%; it outperforms Direct in all but one task. To understand where Direct falls short, we conduct a detailed case study of errors in the transformation from Patient to OMOP CDM. The results are shown in Table 2. Specifically, (1) it makes syntax errors in SELECTION when parsing dates for Person, which could potentially be fixed by employing debugging, (2) it fails to add 'distinct' for Location, (3) it fails in PK/FK assignments, and (4) it fails to include the Death target table 20% of the time. In general, we find that current LLMs are already proficient at schema matching and column transformation, but frequently makes errors in detailed issues. Decomposing the tasks makes it better at attending to details.

For Decomposed, aside from a few obvious column mapping errors (e.g., confusing procedure dates with a person's birthday), it encounters the following challenges:

- **Abstract and Physical Concepts.** In SSB, dates are stored in a dimension table as abstract concepts, while in LINEITEM and ORDERS of TPC-H, dates are part of fact tables representing physical dates. Transformations can only be made from physical to abstract, not the reverse, even when they involve the same concept. Direct tends to take a more holistic approach and correctly returns an empty SQL, whereas Decomposed tries to transform based on concept similarity.
- **IS-A Relationship.** OMOP CDM includes tables that follow an IS-A relationship, where entities like Condition, Procedure, Drug, Specimen, Measurement, or Device are all "IS-A" to Observations. The CDM documentation specifies that records not fitting these specific categories should be stored in the Observations table. Such IS-A relationship are currently not represented by PK/FK and might require representations like ER diagrams. This leads to low accuracy in transforming image_studies and observations. Building such ER diagrams and integrating them into LLMs prompts is an interesting future work.
- **Domain-Specific Knowledge:** Numerous issues require nuanced, domain-specific knowledge. For instance, "encounter" in a healthcare context refers to visits within the healthcare system, but LLMs often confuse it with the occurrence of a medical condition, leading to low accuracy for procedures, claims, and claims_transaction. Additionally, LLMs mistakenly interpret care plans (e.g., "Fracture care") as the condition (e.g., "Fracture") itself, resulting in low accuracy for careplans. Another complex area is cost aggregation, which relies on domain-specific formulas and leads to low accuracy for medications and suppliers. For example, Total Patient Cost is calculated by the formula (Base Cost - Payer Coverage) × Number of Dispenses.

4 CONCLUSION

This paper introduced a framework that decomposes Table-to-Database transformations for LLMs based on the "overview, zoom-in, zoom-out" pattern. Looking ahead, several improvements can be made: (1) Refining methods to handle abstract and physical concepts in databases; (2) Enhancing the representation of IS-A relationships in schemas, potentially through the integration of ER diagrams; (3) Incorporating domain-specific knowledge, potentially through documentation and retrieval.

ACKNOWLEDGMENT

This work is supported by NSF grants 1845638, 1740305, 2008295, 2106197, 2103794, and by Adobe, Amazon, CAIT, and Google. The views expressed are those of the authors and do not necessarily reflect those of the funders.

REFERENCES

[1] [n.d.]. Common Data Model. https://learn.microsoft.com/en-us/common-data-model/.

[2] [n.d.]. Governmentwide Spending Data Model (GSDM). https://fiscal.treasury.gov/data-transparency/GSDM-current.html.

[3] [n.d.]. OMOP Common Data Model. http://ohdsi.github.io/CommonDataModel/index.html.

[4] [n.d.]. White Rabbit. https://github.com/OHDSI/WhiteRabbit.

[5] Paul Brown, Peter Haas, Jussi Myllymaki, Hamid Pirahesh, Berthold Reinwald, and Yannis Sismanis. 2005. Toward automated large-scale information integration and discovery. *Data Management in a Connected World: Essays Dedicated to Hartmut Wedekind on the Occasion of His 70th Birthday* (2005), 161–180.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[7] Victor Dibia. 2023. LIDA: A Tool for Automatic Generation of Grammar-Agnostic Visualizations and Infographics using Large Language Models. *arXiv preprint arXiv:2303.02927* (2023).

[8] Ronald Fagin, Laura M Haas, Mauricio Hernández, Renée J Miller, Lucian Popa, and Yannis Velegrakis. 2009. Clio: Schema mapping creation and data exchange. *Conceptual Modeling: Foundations and Applications: Essays in Honor of John Mylopoulos* (2009), 198–236.

[9] Sumit Gulwani, William R Harris, and Rishabh Singh. 2012. Spreadsheet data manipulation using examples. *Commun. ACM* 55, 8 (2012), 97–105.

[10] Christine Mary Hallinan, Roger Ward, Graeme K Hart, Clair Sullivan, Nicole Pratt, Ashley P Ng, Daniel Capurro, Anton Van Der Vegt, Siaw-Teng Liaw, Oliver Daly, et al. 2024. Seamless EMR data access: Integrated governance, digital health and the OMOP-CDM. *BMJ Health & Care Informatics* 31, 1 (2024).

[11] Yeye He, Xu Chu, Kris Ganjam, Yudian Zheng, Vivek Narasayya, and Surajit Chaudhuri. 2018. Transform-data-by-example (TDE) an extensible search engine for data transformations. *Proceedings of the VLDB Endowment* 11, 10 (2018), 1165–1177.

[12] Zezhou Huang. 2024. Disambiguate Entity Matching using Large Language Models through Relation Discovery. In *Proceedings of the Conference on Governance, Understanding and Integration of Data for Effective and Responsible AI*. 36–39.

[13] Zezhou Huang, Pavan Kalyan Damalapati, and Eugene Wu. 2023. Data Ambiguity Strikes Back: How Documentation Improves GPT's Text-to-SQL. In *NeurIPS 2023 Second Table Representation Learning Workshop*.

[14] Zezhou Huang and Eugene Wu. 2024. Cocoon: Semantic Table Profiling Using Large Language Models. In *Proceedings of the 2024 Workshop on Human-In-the-Loop Data Analytics*. 1–7.

[15] Zezhou Huang and Eugene Wu. 2024. Relationalizing Tables with Large Language Models: The Promise and Challenges. In *2024 IEEE 40th International Conference on Data Engineering Workshops (ICDEW). IEEE.*

[16] Zhongjun Jin, Yeye He, and Surajit Chauduri. 2020. Auto-transform: learning-to-transform by patterns. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2368–2381.

[17] 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.

[18] Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406* (2022).

[19] Maurizio Lenzerini. 2002. Data integration: A theoretical perspective. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*. 233–246.

[20] B Niswonger, LM Haas, and RJ Miller. 2009. Transforming Heterogeneous Data with Database Middleware beyond Integration.

[21] Patrick E O'Neil, Elizabeth J O'Neil, and Xuedong Chen. 2007. The star schema benchmark (SSB). *Pat* 200, 0 (2007), 50.

[22] Erhard Rahm and Philip A Bernstein. 2001. A survey of approaches to automatic schema matching. *the VLDB Journal* 10 (2001), 334–350.

[23] Ankita Sharma, Xuanmao Li, Hong Guan, Guoxin Sun, Liang Zhang, Lanjun Wang, Kesheng Wu, Lei Cao, Erkang Zhu, Alexander Sim, et al. 2023. Automatic data transformation using large language model-an experimental study on building energy data. In *2023 IEEE International Conference on Big Data (BigData)*. IEEE, 1824–1834.

[24] Eitam Sheetrit, Menachem Brief, Moshik Mishaeli, and Oren Elisha. 2024. ReMatch: Retrieval Enhanced Schema Matching with LLMs. *arXiv preprint arXiv:2403.01567* (2024).

[25] Jason Walonoski, Mark Kramer, Joseph Nichols, Andre Quina, Chris Moesel, Dylan Hall, Carlton Duffett, Kudakwashe Dube, Thomas Gallagher, and Scott McLachlan. 2018. Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record. *Journal of the American Medical Informatics Association* 25, 3 (2018), 230–238.

[26] Ling Ling Yan, Renée J Miller, Laura M Haas, and Ronald Fagin. 2001. Data-driven understanding and refinement of schema mappings. In *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*. 485–496.

[27] Junwen Yang, Yeye He, and Surajit Chaudhuri. 2021. Auto-pipeline: synthesizing complex data pipelines by-target using reinforcement learning and search. *arXiv preprint arXiv:2106.13861* (2021).