Journal of Theoretical and Applied Information Technology
30th November 2024. Vol.102. No. 22

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

AN INTELLIGENT MODEL FOR INTEGRATING ENTERPRISE DATA WAREHOUSE LAYERS TO MANAGE SCHEMA EVOLUTION

WADEA GEORGE[1,2], MOHAMMED MARIE[2], AHMED YAKOUB[2]
[1]Department of Computer Science and Math, Faculty of Science, Helwan University Cairo, Egypt
[2]Department of Information Systems, Faculty of Computers and Artificial Intelligence, Helwan University, Cairo, Egypt

E-mail: [1]wadih.gorg@gmail.com, [2]eng_ahmedyakoup@yahoo.com

ABSTRACT

In contemporary organizations, data warehousing centralizes and manages data from diverse sources, such as relational databases and semi-structured systems like Temenos core banking (T24) XML systems, facilitating data analytics and informed decision-making. However, the dynamic nature of data and evolving source schemas pose challenges in adapting data warehouses efficiently, leading to interruptions in data loading processes. This research proposes an intelligent model to enhance data integration in data warehouses, aiming to automate or semi-automate the development cycle and integrate various layers, including business analysis, data modeling, ETL (Extract, Transform, Load), and data quality enhancement through rejection handling. Leveraging metadata, data dictionaries, data mining techniques, and Data Vault modeling, the model aims to reduce development time and costs. The proposed model offers an efficient solution to adapt to changes, significantly reducing adaptation costs compared to prior approaches. By seamlessly integrating all layers of the data warehouse (DWH), it streamlines development cycles through automation or semi-automation, introducing additional features to expedite the process. This research demonstrates the model's superiority through three implemented experiments, showing significant time savings and cost reductions (75% decrease compared to manual processes). It successfully integrates development layers, semi-automates the development process, filters rejected data, and provides a clear vision of schema storage during new changes deployment.

Keywords: *ETL; DWH; Data modeling; Schema evolution; Information Retrieval; Metadata.*

1. INTRODUCTION

The integration of diverse data sources into Data Warehouses (DWH) poses a complex and evolving challenge within the realm of Data Management. This challenge arises from the frequent evolution of schemas due to structural changes or system upgrades, with External Data Sources (EDSs) often undergoing significant alterations. For instance, telecommunication data sources typically experience schema changes every 7-13 days on average, while banking data sources, although relatively more stable, still encounter schema changes every 2-4 weeks. These changes generally involve modifications such as columns. Length adjustments, data type alterations, and the addition of new columns [1]. In practice, changes to a DWH schema are driven by several factors, including (1) the evolution of external data sources, (2) changes in the real-world entities represented by a DWH, (3) new user requirements, and (4) the creation of simulation environments, which are among the most common causes [2]. Previous research has extensively addressed issues related to metadata and schema management, ETL process optimization, schema design and evolution, performance optimization, and data quality in data warehousing environments. For instance, a metadata-based framework [3] effectively manages schema changes by detecting, understanding, and organizing Metadata into layers, while a

wrapper/mediator architecture [4] addresses schema evolution in web data sources using wrappers for data extraction and mediators for integration. Additionally, schema transformations [5] incorporate temporal elements and key adjustments, and an adaptive transformation framework [6] automates the detection and adaptation to source schema changes. Furthermore, a robust metadata repository [7] supports schema conversion, integration, and transformation into a star schema. To optimize ETL processes, various techniques have been proposed, such as dynamically adjusting workflows based on time constraints and workload variations [8] to ensure ETL scalability, data freshness, and time efficiency. The E-ETL framework [9] employs Case-Based Reasoning for ETL workflow repair and adaptability, while an RDF-based ETL [10] enhances semantic data integration through schema-level metadata automation. Other strategies for ETL optimization include source data optimization, parallel processing, caching, incremental loading, and monitoring [11], with a graph-based model [12] combining policy annotations and automated algorithms for schema change impact prediction.

In terms of enhancing data quality, advanced ETL frameworks [13] manage the complexities of integrating Traditional Chinese Medicine (TCM) data, while rule-based frameworks [14] use graph-based models to handle schema changes in ETL processes. Comprehensive approaches [15-20] discuss designing, evolving, and managing data warehouse schemas with requirement-driven and user-centric design principles. Research on balancing OLTP and OLAP systems [21] explores dimensional data modeling, data loading, staging techniques, and partitioning strategies, while methods for enhancing data warehouse performance [22-27] include automated schema generation and dynamic architecture adaptation. Logical Schema-Based Mapping (LSM) [28] improves data retrieval efficiency through keyword-based searches, and studies on adapting to business needs [29, 32] focus on semi-automatically generating schema versions based on changing requirements. Enhancing data quality [30] utilizes advanced algorithms during the ETL process, while research on evolving ETL and multi-version data warehouses (MVDW) [1] discusses managing structural changes. An ontological approach [31] suggests handling schema evolution at the ontological level to minimize adaptation costs, and other approaches [2] propose solutions for managing temporal and multi-version data warehouses. Despite these efforts, previous approaches to managing schema evolution in data warehouses have often been fragmented, addressing metadata management, schema evolution, and ETL optimization separately. This fragmentation has led to increased maintenance efforts, inconsistencies, and a lack of comprehensive integration. Furthermore, traditional methods have typically involved significant manual intervention, making the process cumbersome and prone to errors. As a result, the absence of a cohesive framework that integrates different aspects of DWH development has hindered the efficiency and reliability of data management. In contrast, the proposed model in this paper offers a more advanced solution by reducing both development time and cost. This model specifically addresses the automation of core development processes across DWH layers, including schema evolution management, business analysis, schema modeling, ETL automation, rejection handling, and schema storage assessment. By streamlining these processes, the model aims to reduce time, cost, and manual effort compared to traditional approaches.

The scope of this research is confined to practical development and experimental validation, demonstrating the model's effectiveness in optimizing DWH integration. Key achievements include significant reductions in development cycle time (up to 75% savings) and cost savings through semi-automated and automated processes.

This work does not extend to data governance and security layers, alternative data modeling methodologies beyond Data Vault modeling, or in-depth theoretical examination of individual development phases. Additionally, detailed industry-specific case studies are beyond this scope, aside from selected examples illustrating the model's impact. These topics are recognized as future work, positioning this study as a targeted exploration of schema evolution and layer integration within enterprise DWH environments. The model integrates data warehouse layers through six phases: Schema Evolution Trigger, Business Analysis, Schema Modeling Automation, ETL Generation, Rejection Handling, and Schema Storage Assessment. Through these integrated

layers, the development cycle can be automated within a few hours, significantly improving the assessment of impacts before deployment and enhancing the handling of garbage data. Each phase addresses specific challenges, aligns business objectives, ensures efficient data management, enhances scalability, and allows resumption from any layer when needed. Together, these phases streamline data warehouse development, enhance functionality, and enable adaptation to evolving enterprise needs.

2. PROBLEM STATEMENT

Despite extensive research addressing metadata management, schema evolution, and ETL process optimization in data warehousing environments, existing approaches have often been fragmented, tackling these issues separately and leading to increased maintenance efforts, inconsistencies, and a lack of comprehensive integration within the data warehouse development cycle; traditional methods typically involve significant manual intervention, making the process cumbersome, error-prone, and time-consuming, especially as frequent schema changes in external data sources exacerbate these challenges by causing interruptions in data loading processes, delays in analytics, and elevated adaptation costs—thus, the absence of a cohesive framework that integrates different aspects of DWH development, such as schema evolution detection, business analysis, schema modeling, ETL automation, rejection handling, and storage assessment, hinders the efficiency and reliability of data management; the primary problem this research addresses is how data warehouse development processes can be integrated and automated to efficiently manage schema evolution in external data sources, thereby reducing development time, costs, and maintenance efforts while improving data quality and adaptability to evolving business needs—a critical issue for organizations relying on data warehouses for strategic decision-making, which necessitates a holistic approach that unifies various layers of the data warehouse development cycle into an intelligent, integrated model capable of automating responses to schema changes, streamlining development processes, enhancing data quality, and being scalable and adaptable to future changes; by focusing on this problem, the research aims to fill the gap left by previous fragmented approaches, providing a comprehensive solution that enhances the efficiency and reliability of data warehouse management in the face of ongoing schema evolution.

3. THE PROPOSED MODEL

The proposed model streamlines data warehouse development by automating and integrating processes across six phases: schema evolution trigger, business analysis, schema modeling automation, ETL automation, rejection handling, and schema storage assessment. This integrated approach reduces maintenance efforts, accelerates development cycles, and enhances data quality and adaptability. Key features include automated schema updates, faster business analysis via inverted indexing, scalable schema modeling with Data Vault, reduced manual ETL processes, advanced data quality mechanisms, and optimized schema storage as shown in Figure 1. Experiments show significant reductions in time, cost, and manual effort, improving overall efficiency and functionality to meet dynamic organizational needs.

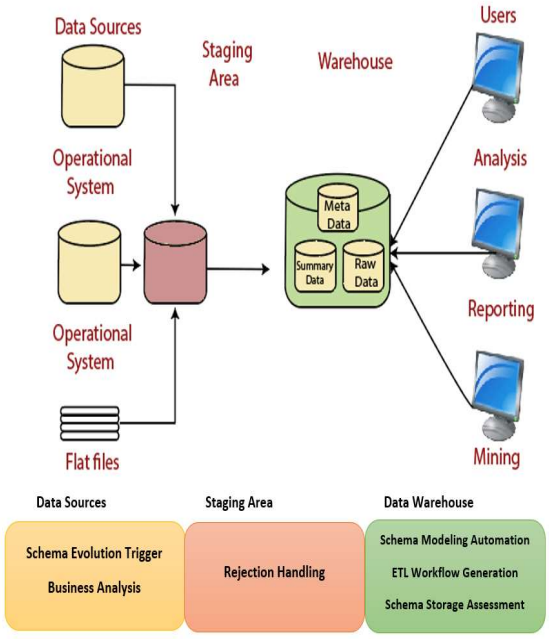

Figure 1. The impact of the proposed model phases on DWH architecture (Source, Staging area, and DWH repository) [35]

3.1. Schema Evolution Trigger

Objective: The Schema Evolution Trigger phase plays a crucial role in maintaining the integrity of the data warehouse by monitoring and managing

schema changes within the database environment. It ensures meticulous tracking and documentation of all alterations over time, providing a reliable audit trail. This consistent oversight allows the data warehouse to remain resilient and adaptable to changing data source structures.

Algorithm 1: Schema Change Detector
Input: Database connection details
Output: Logged schema changes

1. Create table `table_ddl_audit_log` with columns `event_type`, `object_name`, `event_date`.
2. Create trigger `table_ddl_audit_trigger` after creating, altering, dropping, or renaming events.
3. Inside the trigger:
   a. Retrieve event type using `ora_sysevent`.
   b. Retrieve object name using `ora_dict_obj_name`.
   c. Insert event details into `table_ddl_audit_log`.
4. For column changes (add, delete, rename):
   a. Insert 'ADDED_COLUMN' for newly added columns.
   b. Insert 'DELETED_COLUMN' for removed columns.
   c. Insert 'RENAMED_COLUMN' for renamed columns.
5. Commit the transaction.

3.2. Business Analysis

Objective: The Business Analysis phase is designed to identify and highlight data that aligns with business requirements through advanced information retrieval methods. This approach assists data warehouse modelers in efficiently capturing relevant data, ensuring that the data warehouse supports the strategic objectives of the organization. Consequently, this alignment helps the organization respond more effectively to evolving market conditions. as illustrated in Figure 2.

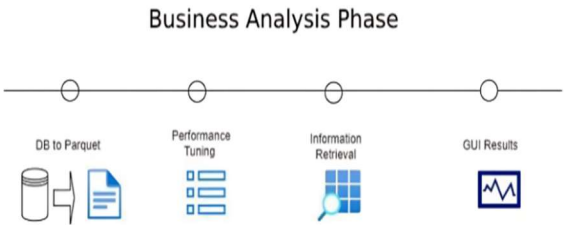

Figure 2. Business Analysis Phase structure

Algorithm 2: Business Keywords Analyst
Input: Oracle database connection parameters, search words, columns to combine, Parquet file path
Output: Plot of word counts across DataFrame columns

1. Connect to the Oracle database and retrieve data.
2. Combine specified columns in the data frame.
3. Write DataFrame to Parquet file.
4. Create an inverted index for the data frame.
5. Perform parallel word counting using ThreadPoolExecutor.
6. Filter and prepare data for plotting.
7. Plot the word counts using Matplotlib.

3.3. Schema Modeling Automation

Data Vault modeling is a methodology for constructing a data warehouse from heterogeneous sources by automating schema modeling. This approach ensures that the data warehouse remains scalable and adaptable to changes in the source systems. By automating the schema modeling process, Data Vault modeling significantly reduces the time and effort required to integrate new data sources. These are the components of data vault modeling:

- **Hubs:** Core entities representing unique business keys (e.g., customers, products).
- **Links:** Elements that connect hubs, modeling many-to-many business relationships.
- **Satellites:** Store time-variant descriptive attributes for hubs and links, maintaining historical data.
- **Satellite Tracking Hubs:** Track historical changes in satellites to manage data changes.
- **Satellite Tracking Links:** Monitor changes across satellites, managing complex attribute relationships over time [33].

Objective: The Schema Modeling Automation phase utilizes Data Vault modeling, a methodology designed to automate the construction of a data warehouse from heterogeneous sources. This phase involves creating a scalable and flexible schema model that can efficiently organize and manage data from various sources. The automated approach significantly reduces development time and enhances the model's ability to adapt to future changes as shown in Figure 3.

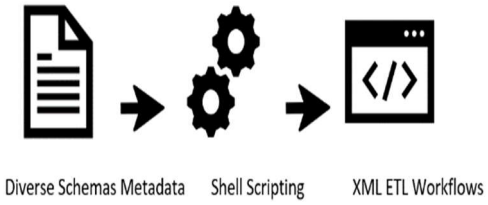

Figure 3. Business Analysis Phase structure

Algorithm 3: Data Warehouse DWH Modeling Generator
Input: User inputs, database connection details
Output: Data Vault model, Data Definition Language (DDL), and Mapping Excel sheet

1. Connect to Oracle and Teradata databases.
2. Extract data and metadata from source systems.
3. Propose a Data Vault schema based on metadata.
4. Generate DDL statements and mapping sheets.
5. Apply business rules and transformations.
6. Create Hubs, Links, and Satellites in the target Data Vault schema.
7. Integrate data into the Data Vault model.
8. Generate SQL scripts for Data Vault structures.
9. Archive data, generate DDL, and export mapping sheets.
10. Finalize and close database connections.

3.4. ETL Automation

Objective: The ETL Automation phase leverages metadata to streamline the creation and implementation of ETL (Extract, Transform, Load) processes, optimizing data integration and ensuring efficient data flow. By automating these processes, this phase reduces manual intervention, minimizes errors, and accelerates the data loading process. This automation also ensures consistency in data handling, which is crucial for maintaining data accuracy and reliability.

Algorithm 4: ETL Pipeline Generator
Input: Source files, target directories, configuration files
Output: ETL workflows

1. Initialize variables and paths for data processing tasks.
2. Prepare and upload mapping sheet data to Teradata.
3. Analyze XSD for XML schema definition.
4. Apply business logic transformations and validate data.
5. Generate audit logs and control tables.
6. Create and deploy ETL workflows in Informatica PowerCenter.
7. Execute workflows and manage execution logs.

3.5. Rejection Handling

Objective: The Rejection Handling phase is crucial for maintaining data integrity by systematically filtering and managing invalid records. This phase segregates records into valid and rejected categories, ensuring that only accurate and reliable data is processed within the data warehouse. Effective rejection handling not only preserves data quality but also streamlines the correction process for any invalid data identified as shown in Figure 4

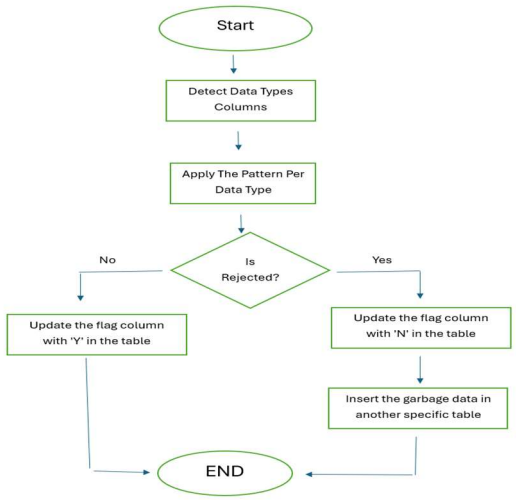

Figure 4. Rejection handling phase flowchart

Algorithm 5: Garbage Data Filter
Input: Data records
Output: Valid and rejected records

1. Initialize lists for valid and rejected records.
2. Define validation checks (Date, Character, Decimal).
3. Iterate over each record for validation.
4. Perform validation based on criteria.
5. Handle validation outcome:
   a. Add rejected records to the rejection list with reasons.
   b. Add valid records to the valid list.
6. Insert rejected records into the Rejection Table.
7. Update the source table to flag processed records.

3.6. Schema Storage Assessment

Objective: The Schema Storage Assessment phase plays a vital role in evaluating and organizing storage utilization within the database schema. It provides visual insights that enable decision-makers to manage storage needs proactively, ensuring that the database remains efficient and scalable. By optimizing storage allocation, this phase supports the long-term sustainability of the data warehouse's infrastructure as shown in Figure 5.

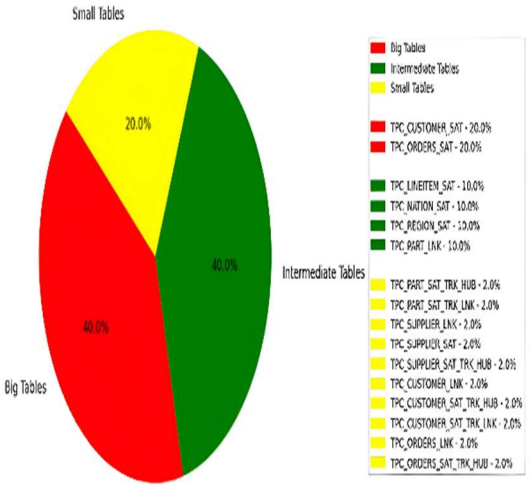

Figure 5. Schema storage assessment phase result

Algorithm 6: Schema Storage Measure
Input: Database credentials
Output: Dash app displaying schema storage utilization

1. Connect to the database and retrieve space utilization data.
2. Fetch individual table sizes.
3. Categorize tables based on size (Big, Intermediate, Small).
4. Group tables by size category and sum up their sizes.
5. Create a Plot pie chart for storage allocation.
6. Generate custom annotations for each table.
7. Set up the Dash web application layout with the pie chart.
8. Run the Dash app server to display the schema storage dashboard.

These algorithms encapsulate the core functionalities and definitions of each phase, providing a comprehensive yet concise view of the proposed model for efficient data warehouse development and maintenance.

4. EXPERIMENTAL RESULTS

This section discusses the implementation details and experimental results of the proposed model. Furthermore, the proposed model is evaluated to demonstrate its superiority over other similar existing techniques. This section discusses the three experiments that were performed to evaluate different aspects of the proposed model for enhancing Data Warehouse (DWH) efficiency. The first experiment assessed the impact of the model's six phases on time and cost reduction in the data warehouse. The second experiment compared the Ontology Approach against the proposed model to see whether the proposed model is effective in reducing time and costs in the data warehouse. The third experiment then compared Schema Modification Operators (SMOs) to the proposed model in terms of their ability to automate Extract, Transform, Load (ETL) processes and reduce maintenance efforts. Each experiment used distinct datasets, such as T24 temenos core banking extensible markup language [37] for Experiment 1 and TPC-H Schema [34] and Sales Schema as shown in Figure 1 on page 2 of [36] for Experiments 2 and 3, respectively. A variety of development tools were employed across these experiments, including Java, Python, extensible markup language, shell scripting, Excel, relational database management system, and Data Vault Modeling manner.

4.1. Experiment 1:

Experiment 1 is an evaluation result for the six phases of the proposed model and how they enhance the data warehouse development cycle using T24 temenos core banking data (Teller extensible markup language dataset [37]) as illustrated in Table 1. The experiment conducted across the six phases of data warehouse (DWH) development provides insight into the performance gains when using the proposed model versus traditional manual approaches. Here's an explanation of the experiment implementation:

Table 1. Performance comparison of the proposed and traditional approaches with respect to timesaving/reduction

Phases	Manual (Before)	Time Reduction	Cost Impact
1- Schema Evolution Trigger	Missing Metadata Schema changes	Immediate detection	Reduced error costs
2- Business Analysis	It takes a long time to get specific business keywords	Getting data in a few minutes	Increased productivity
3- Schema Modeling Automation	1 day per source	75 % reduction (2 hours per source)	Labor cost reduction
4- ETL Automation	1 day per source	75 % reduction (2 hours per source)	Faster deployment
5- Rejection Handling	Phasing rejection through loading	Filtering rejected data in customized tables to avoid data flow failures to save maintenance time	Improved data quality
6- Schema Storage Assessment	Cannot get feedback about the impact of new development on the schema	Full vision of storage schema to make assessments through developing to reduce assessment time while publishing	Better resource allocation

Schema Evolution Trigger:

Setup: Establish connections to T24 temenos core banking extensible markup language (Teller dataset) [37] and configure Informatica to parse extensible markup language data into a relational staging table.
Execution: Implement the Shema Change Detector algorithm via table_ddl_audit_trigger to monitor and log changes. Introduce various schema modifications to test the trigger's responsiveness such as changes of the Teller table by adding some columns and renaming others.
Evaluation: Review the logs to verify that all intended changes on the table and columns such as altering and dropping levels are captured accurately and within the expected time frame, assessing the trigger's efficiency and reliability.

Business Analysis:

Setup: Integrate an advanced search algorithm Business Keywords Analyst through a Python environment with inverted indexing to facilitate rapid keyword retrieval relevant to product types to help the data modeler make the right decision for modeling the needed columns that meet the business requirements.
Execution: Run the algorithm to identify and extract data related to product types such as savings accounts, current accounts, and mortgage loans.
Evaluation: Analyze the completeness and relevancy of the extracted data from the accurate tables and their columns, and measure the time taken against manual methods to determine efficiency gains.

Schema Modeling Automation:

Setup: configure the modeling Java tool and connect to the Teller table, read metadata, and use it to propose a Data Vault schema with the data warehouse modeling generator to automate the new needed columns or tables based on the previous steps.
Execution: Generate the mapping sheet and DDL files for the modeled Teller table, ensuring that the proposed model aligns with Data Vault standards.
Evaluation: Compare the automated Data Vault schema against a manually created model for accuracy, and assess the time saved during this phase.

ETL Automation:

Setup: configure algorithm ETL pipeline generator through shell scripting Linux to prepare the mapping sheet by translating it into a format usable in the relational table for ETL and script the generation of extensible markup language workflows to cover the newly generated model per the previous step.
Execution: Deploy and run the ETL workflows to load data into the data warehouse, utilizing the

generated extensible markup language files of the modeling of the Teller table from source to targets.

Evaluation: Measure the time taken for ETL processes compared to manual development and implement the slowly changing dimension through the Informatica ETL mapping pipeline.

Rejection Handling:

Setup: Configure the Garbage data filter through the procedural language extension to Structured Query Language (SQL) stored procedure which uses data type patterns over the ETL system to identify and redirect invalid records of the Teller table to rejection tables.

Execution: Execute the stored procedure during the ETL process, focusing on how rejected records are managed and isolated from the valid data flow.

Evaluation: Assess the effectiveness of the rejection handling system by reviewing the invalid data captured, reasons for rejection, and the ease of rectifying these records.

Schema Storage Assessment:

Setup: Set up the Schema Storage measure algorithm as a Python tool reading the database schema which is modeled for the Teller table to measure and categorize the storage consumed by different elements of the schema in the development environment, particularly focusing on the new model of the Teller table.

Execution: Perform a storage assessment post-ETL to gauge the impact of the new Teller table model on the schema's storage

Evaluation: Use visual tools to analyze the storage allocation and create reports on how the new model influences storage requirements. Evaluate whether the storage used aligns with projections and performance standards.

In conclusion, this structured approach through setup, execution, and evaluation ensures a thorough analysis of each phase's effectiveness. It benchmarks the proposed automated model against traditional methods, demonstrating its value in enhancing the data warehouse development lifecycle.

4.2 Experiment 2:

Experiment 2 is an evaluation result for comparison between the ontology approach [31] and the proposed model.

The experiment focuses on the adaptability and efficiency of both models in handling schema evolution. The results show that the proposed model outperforms the ontology approach in several key areas, including time savings and cost reduction. Illustration of the approach using TPC-H Schema Benchmark (SSB) [34] and Data Vault Modeling Manner which models the data warehouse. TPC-H is a decision support benchmark that represents the used source. as used previously in research [31]. The evolution of the TPC-H source schema encompassed a series of modifications, including the addition and deletion of attributes and tables, as well as the renaming of existing entities. Overall, there were 849 such evolution operations recorded. The frequency distribution of each type of operation is depicted in Figure 6.

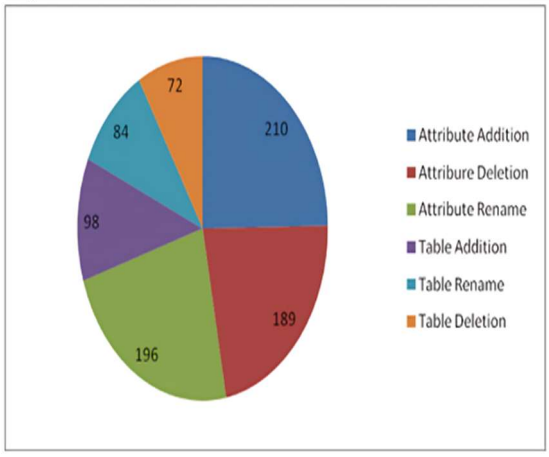

Figure 6. Distribution of occurrence per kind of evolution operations [31]

A summary of the implemented proposed model is presented in Table 2 for finding various types of schema evolution events compared with the total corrected results in the ontology approach of the previous work [31] It was observed that attribute additions and renaming were predominant, significantly influencing the activities within the test cases. Crucially, the proposed framework demonstrated an effective capacity to adapt activities to these frequent changes. Figure 7 and Figure 8 illustrate the efficacy of the proposed model, plotting the evolution operators on the x-axis against the number of impacted entities on the y-axis. These figures illustrate the count of affected entities against those successfully amended through the proposed approach.

Table 2. Affected and corrected operations for the ontology model and the proposed model.

Evolution Event Type	Total Affected	Total Corrected	Total Enhanced

		(Ontology Approach)	(Proposed Model)
Attribute Addition	210	206	209
Attribute Deletion	189	189	189
Attribute Rename	196	194	195
Table Addition	98	95	97
Table Deletion	84	83	84
Table Rename	72	69	71

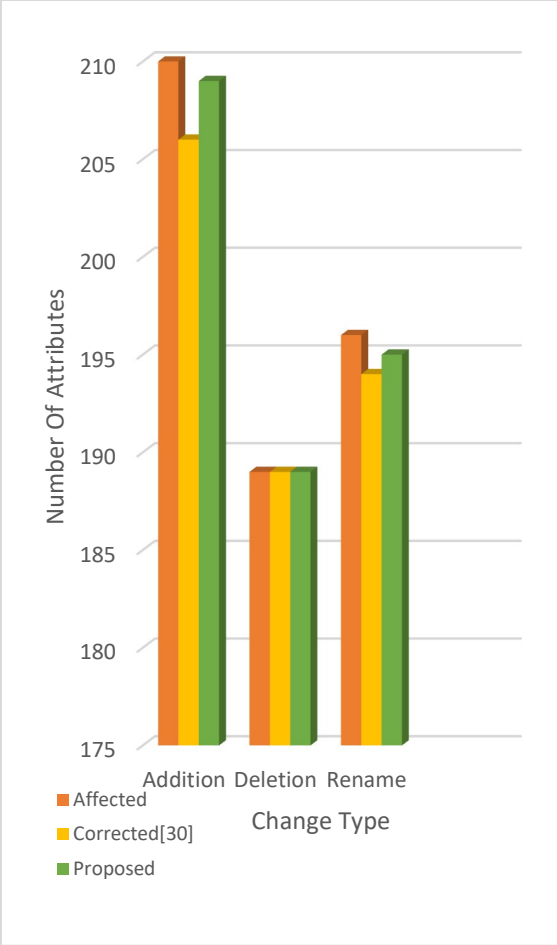

Figure 7 No. Of Attributes Affected, corrected, and enhanced for the Ontology model and the proposed model

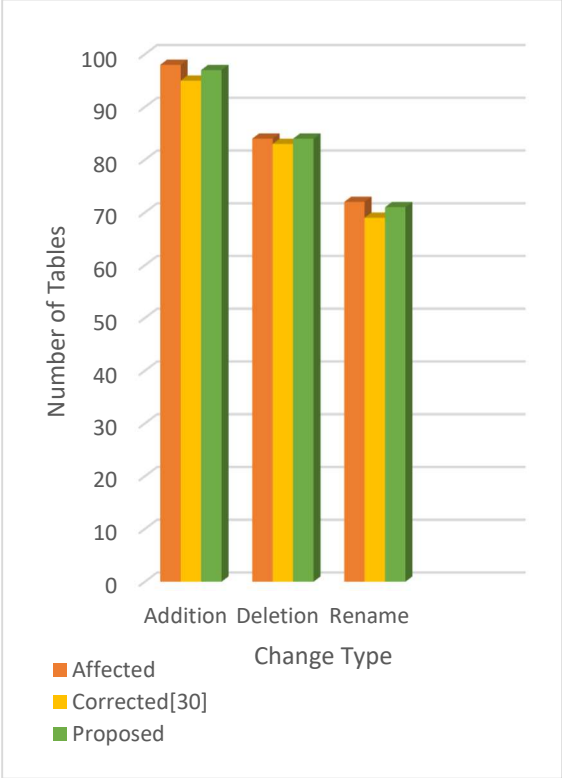

Figure 8 No. of tables affected, corrected and enhanced for the ontology model and the proposed model.

Result Analysis:

To assess the proposed model, these assumptions are applied based on the algorithm that was used in the ontological approach [31]

The manual effort comprises detection, inspection, and where necessary the rewriting of affected activities by an event.

Human effort for manual handling of schema evolution for a change c, over an event e, is expressed as:

$$MC_c^e = AX_c^e + RX_c^e \quad (1)$$

Where,

AX = no. of changes c, affected by event e, that is manually detected

RX = no. of changes c, which must be manually updated to event e

For a set of evolution operators O, in activity A, the overall cost of manual adaption to the change c, for an event e, is given as:

$$CMA = \sum_{c \in O} \sum_{e \in A} MC_c^e \quad (2)$$

Automatic handling of schema evolution using the proposed ontological approach is quantified as

a sum of no. of changes imposed on the DW schema CS and the cost of manually discovering and adjusting activities AMC that escape the automation Ad, the latter cost AMC is expressed as:

$$\mathbf{AMC}= \sum_{c\in O} \sum_{e\in Ad} \mathbf{MC}_{c}^{e} \quad (3)$$

The overall cost of automated adoption is given by,

$$CAA=CS+AMC \quad (4)$$

Applying that approach to Addition detection:

Manual Effort for Addition Detection:

- Detection Effort (AX)=308 changes detected, suppose each requires 1 effort unit, So the total detection effort is 308 ∗1=308 effort units.
- Update Effort (RX): 308 changes requiring updates, each needing 2 effort units, leading to 616 effort units.
- Total Manual Effort (CMA): The sum of detection and update efforts, amounting to 924 effort units.

Applying accordingly to the Ontology Model:

Now, let's consider the automated approach as described:

- Automatically Handled Changes (CS): the automation can handle 301 out of 308 detected changes. Ideally requiring no manual effort.
- Manually Corrected Post-Automation (AMC): 7 changes that the automation cannot handle, each requiring 3 effort units (2 for updates and 1 for detection), totaling 21 effort units.
- Total Cost of Automated Adaptation (CAA): Only includes AMC, amounting to 21 effort units since CS requires no manual effort.
- Thus, the total cost of automated adaptation (CAA) would be 0+21 effort units (since CS changes don't add to the manual effort, only AMC does).

Applying accordingly to the Proposed Model:

Now, let's consider the automated approach as described:

Automatically Handled Changes (CS): The proposed model can automatically handle 306 out of the 308 detected changes, also requiring no manual effort.

Manually Corrected Post-Automation (AMC): 2 changes that the automation cannot handle, each requiring 3 effort units (2 for updates and 1 for detection), totaling 6 effort units.

Total Cost of Automated Adaptation (CAA): Only includes AMC, amounting to 6 effort units since CS requires no manual effort.

Similarly, can calculated over three levels Addition, Deletion, and Renaming.

Based on the results presented in the ontology work [31] and depicted in Figure 9, it has been observed that the cost of automated adaptation (CAA) using the ontology approach is lower than the manual cost of adoption (CMA) associated with the Multi-Version Trajectory Data Warehouse (MVTDW) approach [32].

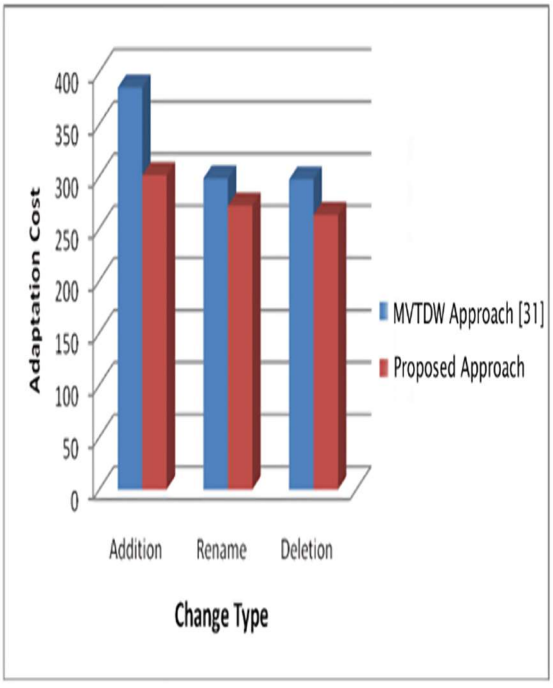

Figure 9 Comparison of Adaptation cost for existing and proposed model [31]

Journal of Theoretical and Applied Information Technology
30th November 2024. Vol.102. No. 22

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 3. Comparison cost between the ontology model and the proposed model

	Addition	Deletion	Rename
Affected	308	273	268
Corrected (Ontology)	301	272	263
Enhanced (Proposed)	306	273	266
CMA(Affected)	924	819	804
AMC(Ontology)	21	3	15
AMC (Proposed Model)	6	0	6

Referring to the results articulated in Table 3 for the proposed model, it can deduce the following metrics:

Ontological approach for addition = 300 Adaptation Cost
So, 300 Cost = 21 effort unit
So, 1 effort unit = 14.2 Cost on the figure
So, the Proposed Model will be 14.2 * 6 ≈ 85

Ontological approach for Deletion = 275 Adaptation Cost
So, 275 Cost = 3 effort unit
So, 1 effort unit = 91.6 Cost on the figure
So, the Proposed Model will be 91.6 * 0 =0

- Ontological approach for Renaming = 260 Adaptation Cost
  So, 260 Cost = 15 effort unit
  So, 1 effort unit = 17.3 Cost on the figure
  So, the Proposed Model will be 17.3 * 6 ≈ 104

This outcome demonstrates the efficiency of the proposed model in reducing manual labor and overall costs compared with MVTDW and Ontology approaches as shown in Figure 10

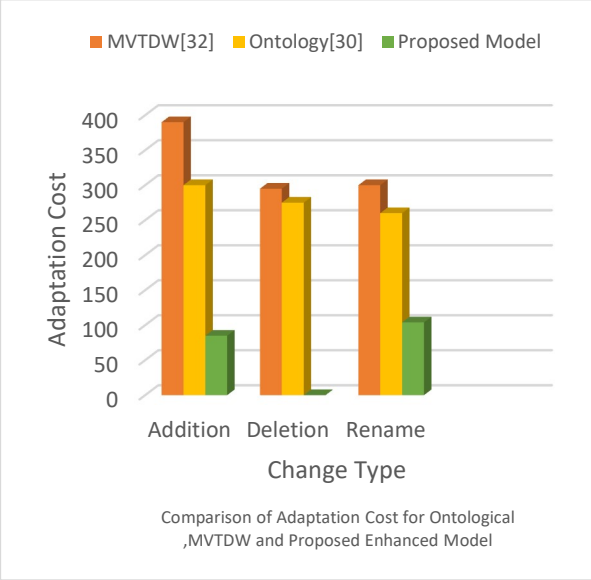

Figure 10 Comparison results of Adaptation cost for MVTDW, ontology, and the proposed model.

4.3 Experiment 3:

This section is an evaluation result for comparison between the SMOs approach [2] and the proposed model. The experiment examines how each approach handles schema modifications in a multi-version data warehouse environment. The findings indicate that the proposed model offers better scalability and reduced maintenance efforts compared to the SMO approach through:

1. **Multi-version Multidimensional Model (MV MD model):** This part of the model focuses on managing schema changes by allowing the creation of multiple versions of the data warehouse schema, each representing different structural configurations over time.

2. **Schema Modification Operators (SMOs):** SMOs are mechanisms used within the MV MD model to enact schema changes systematically. They enable the addition, deletion, or modification of schema elements. After creating the same used schema metadata Sales Schema as illustrated in Figure 11 focusing on the schema change evolution scope, these results were obtained after comparing with the functional description of the schema modification operations (SMOs) [2] as illustrated in Table 4. The proposed model

outlines six steps for smooth data warehouse (DW) development, aiming to simplify and automate changes across different schemas. It focuses on improving schema scalability within the same version to reduce maintenance efforts and complexity. These changes are integrated into the Extract, Transform, Load (ETL) layer, a process not addressed by the Schema Modification Operators (SMO) approach [2]. However, managing complex systems with multiple schema versions poses challenges. It requires a thorough understanding and careful planning for smooth operation. Yet, this complexity comes with drawbacks. Performance overhead is a concern due to dynamic coercion functions and managing multiple versions, potentially slowing down query execution. Moreover, maintaining historical versions and potential data duplication increases storage needs and leads to inconsistency in retrieving information from different DW versions.

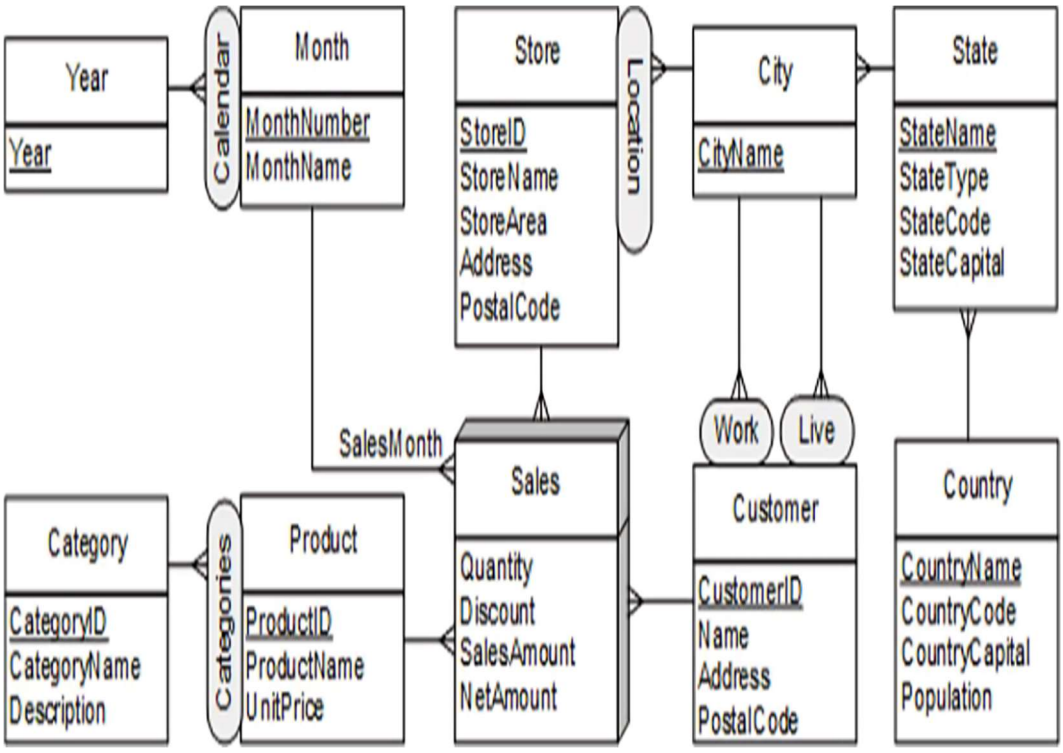

Figure 11. Sales Schema in a fictitious company [36].

Table 4. Comparison of Results and features between the SMO model and the proposed model

Schema Change	Nonconformity	Semantics of SMO [2]	Proposed Model	SMO ETL [2]	Proposed Model ETL
Add level	None	Create metadata for a new MV level/fact.	Archive the old version and add the current version	Not Applied	The changes are applied in the ETL Layer
Delete level	Deleted level remains in existing versions.	Update metadata to reflect no new data loaded for it; physical data deletion is not performed.	Archive the old version and delete from the current version	Not Applied	The changes are applied in the ETL Layer

Add attribute	Existing schema versions won't reference the new attribute/measure.	Update metadata to include a new attribute/measure in the metalevel/meta fact, creating a new schema version.	Archive the old version and add the current version	Not Applied	The changes are applied in the ETL Layer
Delete attribute	New schema versions won't reference the deleted attribute/measure.	Update metadata to exclude the attribute/measure in the new schema version; existing versions remain unchanged.	Archive the old version and delete it from the current	Not Applied	The changes are applied in the ETL Layer
Rename attribute	None	Update metadata to include alias mapping for the renamed attribute/measure, ensuring consistency across versions.	Update metadata to include alias mapping for the renamed attribute/measure, ensuring consistency across versions.	Not Applied	The changes are applied in the ETL Layer
Change attribute/measure domain to a specific.	Potential type mismatch in existing versions.	Update metadata to reflect the type change, ensuring data type conversion where applicable.	Archive the old version and modify the attribute of the current version	Not Applied	The changes are applied in the ETL Layer
Change attribute/measure domain to generic.	Potential type mismatch due to inconvertibility.	Similar to specific-to-generic change, update metadata and ensure conversion of existing data types.	Archive the old version and modify the metadata of the current version	Not Applied	The changes are applied in the ETL Layer
Add relationship	Orphaning of child members in existing schema versions.	Update metadata to define a new relationship, ensuring the linking of orphan child members to a default parent.	Archive the old version and update the relationship in the Link entity of the current version	Not Applied	The changes are applied in the ETL Layer
Delete relationship	Orphaning of child members in future schema versions.	Update metadata to reflect the relationship deletion; no action on the physical data.	Archive the old version and delete the relationship in the Link entity of the current version	Not Applied	The changes are applied in the ETL Layer
Change cardinality	Violation of new cardinality constraints in existing/future schema versions.	Update metadata to create a new relationship version, adjusting constraints as needed.	Archive the old version and delete the relationship in the Link entity of the current version.	Not Applied	The changes are applied in the ETL Layer
Make granularity finer/coarser.	Data semantics variation across schema versions.	Update metadata to reflect the addition of a new dimension" to accommodate granularity changes, applying transformations as needed.	Update metadata to reflect the addition of a new "dimension" to accommodate granularity changes, applying transformations as needed in ETL	Not Applied	The changes are applied in the ETL Layer

5. CONCLUSION AND FUTURE WORK

This paper introduced an intelligent In addressing the critical challenge of managing schema evolution in data warehouses, this paper has introduced an intelligent model that integrates and automates key layers of the development cycle—including schema evolution detection, business analysis, schema modeling automation, ETL processes, rejection handling, and schema storage assessment. By directly tackling the problem of fragmented approaches and manual intervention, the model achieves significant reductions in development time and costs—up to 75% compared to traditional methods—while enhancing data quality and adaptability to evolving business needs. The model's contributions include comprehensive integration that reduces maintenance efforts and inconsistencies, and automation that minimizes errors and manual labor. Through advanced rejection handling mechanisms, it improves data quality, and its use of Data Vault modeling and metadata-driven processes provides scalability and adaptability. This integrated approach empowers organizations to maintain uninterrupted data-loading processes and make timely, informed decisions, thereby enhancing efficiency and reliability in data warehouse management. Future work will extend the model to encompass data governance and security layers, further enhancing its robustness and ensuring compliance with regulatory standards and data integrity throughout the data warehouse development cycle.

REFERENCES

[1] R. Wrembel, "On Handling the Evolution of External Data Sources in a Data Warehouse Architecture," Integrations of Data Warehousing, Data Mining and Database Technologies: Innovative Approaches, Poznan University of Technology (Poland), 2011, pp. 42. doi: 10.4018/978-1-60960-537-7.ch006. [Online]. Available: https://www.igi-global.com/chapter/handling-evolution-external-data-sources/53074

[2] W. Ahmed, E. Zimányi, A. A. Vaisman, and R. Wrembel, "Schema Evolution in Multiversion Data Warehouses," International Journal of Data Warehousing and Mining, vol. 17, no. 4, 2021, pp. 1-21. doi: 10.4018/IJDWM.2021100101. [Online]. Available: https://econpapers.repec.org/article/iggjdwm00/v_3a17_3ay_3a2021_3ai_3a4_3ap_3a1-28.htm

[3] G. Shankaranarayanan, "Managing Changes to Schema of Data Sources in a Data Warehouse," Proceedings of AMCIS 2001, Dec. 2001, pp. 1-10. [Online]. Available: http://aisel.aisnet.org/amcis2001/68

[4] A. Marotta, R. Motz, and R. Ruggia, "Managing Source Schema Evolution in Web Warehouses," Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Montevideo (Uruguay), 2001. [Online]. Available: https://www.researchgate.net/publication/220947302_Managing_Source_Schema_Evolution_in_Web_Warehouses

[5] A. Marotta, "Data warehouse design: a schema-transformation approach," Proceedings of IEEE SCCC 2002, Feb. 2002. doi: 10.1109/SCCC.2002.1173188. [Online]. Available: https://www.researchgate.net/publication/4000971_Data_warehouse_design_a_schema-transformation_approach

[6] D. Solodovnikova, "Data Warehouse Adaptation after the Changes in Source Schemata," presented at the Conference Paper, July 2006. [Online]. Available: https://www.researchgate.net/publication/281776380_Data_Warehouse_Adaptation_after_the_Changes_in_Source_Schemata

[7] N. M. Dar and N. Zahra, "Generic Metadata Repository for a Data Warehouse," Dept. of Computer Science, Mohammad Ali Jinnah University (Pakistan) and Islamic International University, Islamabad (Pakistan), 2010. [Online]. Available: https://www.researchgate.net/publication/224100155_Generic_metadata_repository_for_a_data_warehouse

[8] P. M. de Oliveira Martins, "Elastic ETL+Q for Any Data-Warehouse Using Time Bounds," Ph.D. dissertation, Faculdade de Ciências e Tecnologia, Universidade de Coimbra, July 2015. [Online]. Available: https://estudogeral.uc.pt/bitstream/10316/29090/3/Elastic%20ETL+Q%20for%20any%20data-warehouse%20using%20time%20bounds.pdf

[9] "ETL workflow reparation by means of case-based reasoning," Information Systems Frontiers, vol. 20, 2018, pp. 21-43. doi: 10.1007/s10796-016-9732-0. [Online]. Available: https://www.researchgate.net/publication/3121

50128_ETL_workflow_reparation_by_means_of_case-based_reasoning

[10] R. P. Deb Nath et al., "High-level ETL for Semantic Data Warehouses," Semantic Web, vol. 13, 2022, pp. 85-132. doi: 10.3233/SW-210429. [Online]. Available: https://www.researchgate.net/publication/356611581_High-level_ETL_for_Semantic_Data_Warehouses

[11] D. Seenivasan, "Improving the Performance of the ETL Jobs," International Journal of Computer Trends and Technology, vol. 71, no. 3, Mar. 2023, pp. 27-33. doi: 10.14445/22312803/IJCTT-V71I3P105. [Online]. Available: https://www.researchgate.net/publication/369666785_Improving_the_Performance_of_the_ETL_Jobs

[12] "What-If Analysis for Data Warehouse Evolution," presented at the Conference Paper, Sept. 2007. doi: 10.1007/978-3-540-74553-2_3. [Online]. Available: https://www.researchgate.net/publication/220802661_What-If_Analysis_for_Data_Warehouse_Evolution

[13] X. Pan, X. Zhou, H. Song, R. Zhang, and T. Zhang, "Enhanced Data Extraction Transforming and Loading Processing for Traditional Chinese Medicine Clinical Data Warehouse," Proceedings of the 2012 IEEE 14th International Conference on e-Health Networking Applications and Services (Healthcom), 2012. doi: 10.1109/Healthcom.2012.6380066. [Online]. Available: https://ieeexplore.ieee.org/document/6380066

[14] G. Papastefanatos, P. Vassiliadis, A. Simitsis, T. Sellis, and Y. Vassiliou, "Rule-Based Management of Schema Changes at ETL Sources," presented at the Conference Paper, Sept. 2009. doi: 10.1007/978-3-642-12082-4_8. [Online]. Available: https://www.researchgate.net/publication/221651284_Rule-Based_Management_of_Schema_Changes_at_ETL_Sources

[15] P. Jovanovic, O. Romero, A. Simitsis, and A. Abelló, "A requirement-driven approach to the design and evolution of data warehouses," presented at the Conference Paper. [Online]. Available: https://www.researchgate.net/publication/260045268_A_requirement-driven_approach_to_the_design_and_evolution_of_data_warehouses

[16] D. Solodovnikova and L. Niedrite, "Evolution-Oriented User-Centric Data Warehouse," presented at the Conference Paper, Sept. 2011. doi: 10.1007/978-1-4419-9790-6_58. [Online]. Available: https://www.researchgate.net/publication/252709100_Schema_Evolution_for_Data_Warehouse_A_Survey

[17] A. Gosain, "Schema Evolution for Data Warehouse: A Survey," International Journal of Computer Applications, vol. 22, no. 9, May 2011, pp. 23-30. doi: 10.5120/2590-3588. [Online]. Available: https://www.researchgate.net/publication/252709100_Schema_Evolution_for_Data_Warehouse_A_Survey

[18] J. W. Kang, F. Basrizal, Q. Yu, and E. P. Holden, "Web-Based Implementation of Data Warehouse Schema Evolution," Rochester Institute of Technology, Rochester (NY, USA), 2015. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-16211-9_32

[19] P. Manousis, P. Vassiliadis, A. Zarras, and G. Papastefanatos, "Schema Evolution for Databases and Data Warehouses," Proceedings of the Department of Computer Science, University of Ioannina, Ioannina (Greece), 2016. [Online]. Available: https://www.researchgate.net/publication/302591145_Schema_Evolution_for_Databases_and_Data_Warehouses

[20] M. Thenmozhi and K. Vivekanandan, "An Ontological Approach to Handle Multidimensional Schema Evolution for Data Warehouse," International Journal of Database Management Systems (IJDMS), vol. 6, no. 3, June 2014. [Online]. Available: https://www.researchgate.net/publication/274174299_An_Ontological_Approach_to_Handle_Multidimensional_Schema_Evolution_for_Data_Warehouse

[21] R. Dumoulin, "Architecting Data Warehouses for Flexibility, Maintainability, and Performance," Computer Science, 2000. [Online]. Available: https://www.olap.it/Articoli/architectingdwh.pdf

[22] R. A. Ahmed and T. M. Ahmed, "Generating Data Warehouse Schema," International Journal of Foundations of Computer Science & Technology (IJFCST), vol. 4, no. 1, Jan. 2014. doi: 10.5121/ijfcst.2014.4101. [Online]. Available: https://www.researchgate.net/publication/3643

24920_GENERATING_DATA_WAREHOUSE_SCHEMA

[23] C. Gröger, H. Schwarz, and B. Mitschang, "The Deep Data Warehouse: Link-based Integration and Enrichment of Warehouse Data and Unstructured Content," Proceedings of the IEEE 18th International Enterprise Distributed Object Computing Conference, 2014. [Online]. Available: https://www.researchgate.net/publication/270574862_The_Deep_Data_Warehouse_Link-Based_Integration_and_Enrichment_of_Warehouse_Data_and_Unstructured_Content

[24] M. M. Jaber, M. K. Abd Ghani, N. S. Mohammed, A. M. Mohammed, and T. Abbas, "Flexible Data Warehouse Parameters: Toward Building an Integrated Architecture," Proceedings of the IEEE 7th International Conference on Inventive Communication and Computational Technologies (ICICCT), 2015. doi: 10.7763/IJCTE.2015.V7.984. [Online]. Available: https://www.researchgate.net/publication/266554118_Flexible_Data_Warehouse_Parameters_Toward_Building_an_Integrated_Architecture

[25] A. Mendes, "Datawarehouser: A data warehouse artist who has the ability to understand data warehouse schema pictures," presented at the Conference Paper, Nov. 2016. doi: 10.1109/TENCON.2016.7848419. [Online]. Available: https://www.researchgate.net/publication/313584797_Datawarehouser_A_data_warehouse_artist_who_have_ability_to_understand_data_warehouse_schema_pictures

[26] A. H. Al-Rammahi, "Designing a Variety of Data Warehouse Schemas Suitable for Meta-Search Engines," Faculty of Sciences and Fine Arts, June 2013. Book published May 2016. [Online]. Available: https://www.researchgate.net/publication/303683685_DESIGNING_A_VARIETY_OF_DATA_WAREHOUSE_SCHEMAS_SUITABLE_FOR_META-SEARCH_ENGINES

[27] P. Tiwari, A. C. Mishra, S. Kumar, V. Kumar, and B. Terfa, "Improved Performance of Data Warehouse," Proceedings of the International Conference on Inventive Communication and Computational Technologies (ICICCT 2017), 2017. doi: 10.1109/ICICCT.2017.7975220. [Online]. Available: https://www.academia.edu/34829918/Improved_Performance_of_Data_Warehouse

[28] F. Majeed and M. Shoaib, "Reduce Search Space in the Data Warehouse for Keyword-Based Search," International Arab Journal of Information Technology, vol. 14, no. 1, Jan. 2017, pp. 70. [Online]. Available: https://www.semanticscholar.org/paper/Logical-schema-based-mapping-technique-to-reduce-in-Majeed-Shoaib/612a015a939cd38b395761ac154382f817b64e81

[29] D. Solodovnikova, L. Niedrite, and N. Kozmina, "Handling Evolving Data Warehouse Requirements," Proceedings of the East European Conference on Advances in Databases and Information Systems, Sept. 2015. doi: 10.1007/978-3-319-23201-0_35. [Online]. Available: https://www.researchgate.net/publication/285245720_Handling_Evolving_Data_Warehouse_Requirements

[30] N. Gupta and S. Jolly, "Enhancing Data Quality at ETL Stage of Data Warehousing," International Journal of Data Warehousing and Mining, vol. 17, no. 1, Jan.-Mar. 2021, pp. 74. [Online]. Available: https://www.researchgate.net/publication/348951742_Enhancing_Data_Quality_at_ETL_Stage_of_Data_Warehousing

[31] M. Thenmozhi and K. Vivekanandan, "An Ontological Approach to Handle Multidimensional Schema Evolution for Data Warehouse," International Journal of Database Management Systems (IJDMS), vol. 6, no. 3, June 2014. [Online]. Available: https://www.researchgate.net/publication/258652420_An_Ontology_based_Hybrid_Approach_to_Derive_Multidimensional_Schema_for_Data_Warehouse

[32] W. Oueslati and J. Akaichi, "A Multiversion Trajectory Data Warehouse to Handle Structure Changes," International Journal of Database Theory and Application, vol. 4, no. 2, 2011. [Online]. Available: https://www.researchgate.net/publication/275619765_Handling_Instance_Changes_in_a_Multiversion_Trajectory_Data_Warehouse

[33] D. Linstedt and M. Olschimke, Building a Scalable Data Warehouse with Data Vault 2.0, Morgan Kaufmann, Elsevier, Waltham, MA, USA, 2015. [Online]. Available: https://www.amazon.sa/-/en/Building-Scalable-Data-Warehouse-Vault/dp/0128025107

[34] T. P. P. Council, "TPC-H benchmark specification," [Online]. Available:

www.tpc.org/tpch/, 2008. [Accessed: May 20, 2014].

[35] "Data Warehouse Architecture," Javatpoint. [Online]. Available: https://www.javatpoint.com/data-warehouse-architecture. [Accessed: May 24, 2024].

[36] W. Ahmed, E. Zimányi, A. A. Vaisman, and R. Wrembel, "Schema Evolution in Multiversion Data Warehouses," International Journal of Data Warehousing and Mining, vol. 17, no. 4, 2021, pp. 1-21. doi: 10.4018/IJDWM.2021100101.

[37] "T24 temenos core Banking Teller extensible markup language Dataset," Google Drive. [Online]. Available:https://drive.google.com/file/d/1awa8xmwt8x9Mgyew0tbTMWdrSAj7S2MC/view?usp=drive_link. [Accessed: Jun. 1, 2024].