

Data Warehouse Building Based on ETL Processes and Applications for Visual Analysis

Yun Lou

Michigan State University, East Lansing, MI, 48824, US

Keywords: Data warehouse; incremental ETL; selective refresh; impact coverage; visual analytics

Abstract: Addressing the issues of coarse-grained refresh scope and difficulty in consistently translating logical pruning into operational benefits in continuously updated data warehouses, this study constructs a star-structured data warehouse for three types of visual analytics workloads: time trends, regional revenue, and payment structure. A selective refresh method driven by a two-layer field-partition dependency is proposed. This method comprehensively considers change operations, changed fields, and pre- and post-change groupings to determine the necessary refresh keys for the three types of aggregations on the partition incremental baseline. This study conducts deterministic change replay based on NYC TLC Yellow Taxi open data and evaluates the refresh method from four aspects: refresh scope, runtime, stage cost, and analytical consistency. Results show that in low-coverage local corrections, fine-grained dependencies can eliminate irrelevant maintenance; however, as the impact scope expands, scan pruning and write granularity weaken wall-clock benefits. Under high-coverage updates, partition-level incremental updates have completed major pruning. This study demonstrates that the granularity of data warehouse maintenance should be selected in conjunction with the change structure and downstream analytical workloads.

1 Introduction

Data warehouses organize heterogeneous business data into a stable and consistent analytical structure through ETL. Walha et al. pointed out that as ETL shifts from traditional integration to big data environments, the data scale, heterogeneous sources, and complexity of the processing chain will increase the data integration cost [1]. Khan et al.'s review also shows that transformation logic, quality processing, and execution environment jointly affect the data warehouse loading efficiency [2]. For a given ETL task, partitioning and parallel execution can shorten the processing time in big data scenarios and improve resource utilization [3]. HTAP research shows that the requirement for data freshness will make traditional batch processing ETL face a more obvious delay-analysis cost trade-off [4].

When the source schema or transformation logic changes, the impact will be propagated to downstream objects along the ETL relationship. Therefore, existing ETL repair studies have linked

change propagation with process repair [5]. F-IVM reduces full recomputation by maintaining intermediate states affected by update relationships [6]. DBSP starts from incremental computation on the stream and emphasizes reusing existing computation results, making new work more dependent on input changes rather than the full data size [7]. These studies show that maintenance costs should not only be considered in terms of the number of change records, but also in which computation states the change ultimately affects.

Data warehouses also include fact partitioning and pre-aggregates built for analytical tasks. When the amount, payment method or region key changes, the downstream objects affected are not the same; if the grouping key changes, the old group and the new group need to be maintained at the same time. Transactional stream processing research shows that continuous update scenarios need to take into account both transaction consistency and streaming state maintenance[8]. Evolutionary research on stream processing systems also points out that out-of-order data and historical state revisits will cause the already processed time range to re-enter the maintenance process[9].

Based on the above issues, this study focuses on a specific problem: In a data warehouse oriented towards visual analytics workloads, how to use field and partition dependencies to determine the necessary refresh range for incremental ETL, and identify the effective workload boundary for selective refresh while maintaining semantic equivalence between analysis results and complete reconstruction?

This study does not consider dependency graph traversal itself as innovative. The main contribution is to map field-level changes, pre- and post-change grouping, and data warehouse aggregation keys into independently maintainable analytical units, and to compare partitioned incremental and selective refresh under the same fact changes, aggregation expressions, and transaction semantics. This study also analyzes logical refresh scope reduction and actual runtime separately to determine when fine-grained maintenance yields real benefits. Visual analytics is used to determine aggregation granularity and verify the analytical semantics after maintenance, but it is not extended to an independent human-computer interaction study.

2 Research Methods

2.1 Data Objects and Data Warehouse Construction

This study used NYC TLC Yellow Taxi Trip Records. Initial database construction was conducted from October to December 2024, the main experiments from January to December 2025, and out-of-time validation from January to May 2026. The original Parquet files remained read-only; the stable record key was determined by the source filename, Parquet row group number, and row offset.

The temporary storage layer uses a unified time, category, and value type. Duplicate keys and time anomalies are logged, and unmatched regions or payment categories are uniformly mapped to "Unknown". High-value, long-distance, and other extreme business values are not deleted based on subjective thresholds to avoid altering the original data distribution.

The data warehouse uses a star schema. FactTrip records a single trip, and DimDate, DimZone, and DimPayment provide time, region, and payment semantics, respectively. The analysis layer materializes three types of aggregations: daily trips, regional income, and payment structure. The maintenance granularity is shown in Table 1.

The initial ETL process sequentially completes temporary validation, dimension loading, fact loading, foreign key checking, and aggregation construction. After the addition of `cbd\_congestion\_fee` in 2025, the system only updates the schema version; since this field does not participate in the three types of aggregations, historical aggregations do not need to be rebuilt.

Table 1. Data Warehouse Objects, Granularity, and Analytical Purposes

object	Primary key or combination key	Data granularity	Physical organization	Main metrics	Analytical applications
FactTrip	trip_key	Single trip	Month/Date	distance, fare, tip, total	Basic facts
DimDate	date_key	day	Key sorting	—	Time Filtering
DimZone	zone_key	Taxi Zone	Key sorting	—	Regional Analysis
DimPayment	payment_key	Payment categories	Key sorting	—	Payment Structure
AggDailyTrip	date_key	day	date	trip_count	Time trend
AggZoneRevenue	Date key + region key	Date × Region	date	count, revenue	Regional income
AggPaymentSummary	Date key + payment key	Date × Payment	date	count, revenue	Payment Structure

2.2 Field-Partition Two-Tier Dependency Model

To characterize the propagation range of local changes within the warehouse, this study represents ETL dependencies as a directed graph:

$$H = (N, E)$$

Here, N represents the set of nodes consisting of source fields, fact partitions, aggregation groups, and analysis objects, E represents data dependency edges, and $N_r \subseteq N$ represents computational units that can be refreshed independently.

Changes in a batch are defined as: t

$$\Delta_t = (r, o, C, P_{old}, P_{new})$$

Where r represents the change relationship, $o \in \{I, U, D\}$ represents insertion, update and deletion respectively, C represents the set of changed fields, P_{old} and P_{new} represent the grouping state before and after the change respectively.

a node n is defined as:

$$S_n = (R_n, O_n, C_n, G_n, F_n)$$

Here, R_n represents the set of dependencies, O_n represents the type of operation that can affect the node, C_n represents the field involved in the calculation, G_n represents the grouping key, and F_n represents the filtering predicate.

This study defines the matching relationship between changes and nodes as:

$$\text{Match}(\Delta_t, S_n) = \begin{cases} 1, & r \in R_n, o \in O_n, \Gamma(\Delta_t, S_n) = 1 \\ 0, & \text{Other circumstances} \end{cases}$$

in:

$$\Gamma(\Delta_t, S_n) = \begin{cases} 1, & o = U, C \cap (C_n \cup G_n \cup F_n^{col}) \neq \emptyset, \Phi = 1 \\ 1, & o \in \{I, D\}, \Phi = 1 \\ 0, & \text{Other circumstances} \end{cases}$$

F_n^{col} This indicates the set of fields involved in the filtering predicate. Φ It re-evaluates whether records before and after the change fall into the corresponding group and filtering range; when the group key is updated, both the old and new groups are simultaneously added to the refresh set. Insert and Delete affect aggregations such as COUNT and SUM based on the record's existence status.

Based on the matching results, the initial influencing nodes are defined as follows:

$$I_t = \{n \in N \mid \text{Match}(\Delta_t, S_n) = 1\}$$

propagates from the starting point according to the topological order of the dependency graph: I_t

$$A_t = \text{Reach}_H(I_t)$$

After the system completes the propagation of influence, the final refresh set is defined as:

$$R_t = A_t \cap N_r$$

by the registry record fields, operations, grouping keys, filter fields, and aggregation expressions. This design is consistent with the “processing step-output” dependency emphasized by fine-grained Provenance [10], and this study only uses it to determine the scope of data warehouse maintenance.

2.3 Execution Logic of M0, M1 and M2

M0 regenerates three analytical aggregates based on all current valid fact records and serves as a deterministic benchmark for subsequent consistency verification.

M1 employs partition-aware incremental refresh. After each change batch is written to FactTrip, the system extracts the event dates from the recorded old and new states, and then uses DISTINCT to obtain the set of affected dates.

$$D_t = \text{Distinct}(D_{old} \cup D_{new})$$

The system deletes the old results that satisfy the aggregation expressions in the three aggregation tables $\text{date_key} \in D_t$, and then regenerates these dates from the changed fact table according to the original aggregation expressions. Identical dates are only deleted and recalculated once, so duplicate old and new dates do not cause redundant maintenance. After M1 is committed, it is compared with M0 for grouping keys, COUNT, and monetary aggregations.

M2 and M1 use the same fact changes, aggregate expressions, DELETE+INSERT semantics, transaction boundaries, and database snapshots. The only difference lies in the refresh key selection: M2 constructs three independent, non-nested collections respectively.

$$K_D = \{\text{date_key}\}$$

$$K_Z = \{(\text{date_key}, \text{zone_key})\}$$

$$K_P = \{(\text{date_key}, \text{payment_key})\}$$

Specifically, K_D it serves only AggDailyTrip, K_Z only AggZoneRevenue, K_P and only AggPaymentSummary. Each collection is deduplicated independently before executing the SQL. If the grouping key changes, both the old and new keys are added to the corresponding collections simultaneously.

Therefore, the comparison between M1 and M2 maintains the same execution conditions, with the main differences coming from the refresh range and the additional dependency determination cost added in M2.

2.4 Replay of Changes and Locality Experiments

The publicly available TLC data does not produce CDC logs. Therefore, this study directly builds Monthly Append onto subsequent month files, while Update, Delete, and Late-arriving Update are all deterministically replayed on the actual fact record.

Amount updates are only extracted from records where the payment was made by credit card and $\text{tip_amount} > 0$. For each record, another real record is selected as the donor within the same month and the same fare_amount decimal group using a fixed hash, and the following is defined:

$$\begin{aligned}\text{tip}' &= \text{tip}_{\text{donor}} \\ \text{total}' &= \text{total} - \text{tip} + \text{tip}'\end{aligned}$$

Payment category correction limits are set to $1 \leftrightarrow 21$ and $2.3 \leftrightarrow 4$. When category 1 changes to 2, the system simultaneously resets `tip_amount` and corrects `total_amount`. Delete is performed according to:

$$\text{SHA256}(\text{stable_key} \parallel \text{seed})$$

After sorting, select the first nrecord from those that never participated in the same batch of updates. Late-arriving updates extract records from 1, 2, and 3 months prior to the current batch in a 1:1:1 ratio, retaining the original business event dates.

The main experiment set the change ratios to 0.1%, 0.5%, 1%, 2%, 5%, 10%, 20%, and 40%.

To observe the locality of change independently, this study fixed the number of changed records to 1% of the fact table for the current month and uniformly used monetary updates to avoid additional impact from operation types. Four workload tiers only changed the candidate date-region space: L1 limited to 1 day $\times$ 5 regions, L2 limited to 7 days $\times$ 20 regions, L3 limited to 30 days $\times$ 80 regions, and L4 had no date or region restrictions. Each tier selected the same number of records in the same hash order, and the impact coverage was calculated by the actual refresh key, rather than pre-specified.

2.5 Evaluation Indicators and Statistical Protocols

The recalculated reduction rate is defined as:

$$\text{RRR}_t = 1 - \frac{|\text{R}_t^{\text{M2}}|}{|\text{R}_t^{\text{M1}}|}$$

This study defines all refreshable keys actually existing in the three aggregate tables for the current month as the candidate refresh set:

$$\text{U}_t = \text{U}_D \cup \text{U}_Z \cup \text{U}_P$$

Impact coverage is defined as:

$$\text{AC}_t = \frac{|\text{R}_t^{\text{M2}}|}{|\text{U}_t|}$$

This definition allows different localized workloads to be compared within the same monthly candidate space.

This study uses the mean absolute relative error to measure the difference between the numerical analysis results and M0:

$$\text{MARE}_m = \frac{1}{K} \sum_{k=1}^K \frac{|Q_{k,m} - Q_{k,0}|}{\max(|Q_{k,0}|, \varepsilon)}$$

Where, $m \in \{M1, M2\}$ represents $Q_{k,0}$ the i -th result obtained by M0 k , K is the number of evaluation results, and $\varepsilon = 10^{-12}$.

To explain the difference between logic clipping and wall clock benefits, this study decomposes the running time into:

$$T = T_{\text{change}} + T_{\text{match}} + T_{\text{prop}} + T_{\text{query}} + T_{\text{write}} + T_{\text{commit}}$$

Wherein, T_{change} represents the time for applying fact changes, T_{match} represents the time for change-dependency matching, T_{prop} represents the time for impact propagation, T_{query} represents the time for executing scan and aggregation SQL, T_{write} represents the time for deleting and writing results, and T_{commit} represents the time for committing transactions. M1 does not T_{match} exist T_{prop} .

Each "month $\times$ variation set $\times$ method" model undergoes one warm-up and ten formal runs; the warm-up results are not included in the statistics. This study first calculates the median of the ten formal runs, then calculates the intra-month median for the four variation sets; the phase proportion is calculated using the median of the monthly proportions over 12 months. Method effects are estimated using 12-month paired units, and 95% confidence intervals are estimated using 10,000 monthly bootstrap iterations. This resampling only describes the uncertainty of the existing sample and does not increase the number of independent samples.

2.6 Visual Analytics Application Design

The visual analytics setup includes three fixed tasks. T1 presents daily trip volume using a line chart and observes localized time changes through date selection; T2 uses a Taxi Zone tiered color chart to represent regional revenue, while retaining a Top-10 ranking for precise comparison; T3 uses a 100% stacked bar chart to represent payment method composition. All three views share date and region filtering status. Date selection synchronously updates the region and payment views, and region selection also updates the time trend and payment structure in reverse.

Coscia et al. pointed out that data integration should not be completely separated from subsequent visual analysis, and the data preparation method will affect the organization and execution of the analysis task [11]. Maack et al. explained from the perspective of Visual Analytics workflow that there is a continuous relationship between data processing, visual expression and analysis process [12]. Based on this positioning, this study only evaluates query response, cross-view filtering and result consistency, and does not infer subjective usability or decision quality in the absence of user experiments.

3. Research Results

3.1 Data Warehouse Construction and Quality Verification

The initial data from October to December 2024 generated approximately 11.15 million fact records. ETL retained duplicate keys, time anomalies, unknown mappings, and foreign key logs, and did not remove extreme business values based on amount or distance thresholds. After completing the unknown mappings, the number of orphaned foreign keys in the fact table was 0.

The three analytical aggregations form day, date $\times$ region, and date $\times$ payment granularity, respectively. This study recalculates COUNT, SUM, and AVG using independent SQL on the fact

table and cross-verifies the results with the aggregation table results. After `cbd_congestion_fee` entered the source schema in 2025, the schema version was updated accordingly; since this field did not participate in the three analytical aggregations, dependency registration did not trigger historical aggregation reconstruction.

3.2 Refresh Range and Operating Performance

Table 2 summarizes the main results for the 12-month main window.

Table 2 Incremental maintenance results under different change ratios

Change ratio	M1 median time / IQR(s)	Median time of M2 / IQR(s)	RRR	95% CI of the time change of M2 relative to M1
0.1%	4.38 / 4.16–4.61	3.99 / 3.74–4.28	27.4%	[−14.7%, −3.0%]
0.5%	4.95 / 4.70–5.21	4.60 / 4.31–4.93	23.1%	[−11.8%, −1.9%]
1%	5.36 / 5.02–5.68	5.02 / 4.69–5.39	21.8%	[−10.8%, −0.8%]
2%	6.14 / 5.78–6.51	5.98 / 5.60–6.39	16.0%	[−6.0%, 1.2%]
5%	7.99 / 7.55–8.43	7.94 / 7.49–8.48	9.3%	[−3.5%, 2.9%]
10%	10.56 / 9.99–11.16	10.73 / 10.11–11.45	5.0%	[−1.7%, 5.2%]
20%	15.42 / 14.68–16.21	15.93 / 15.12–16.88	2.8%	[−0.4%, 7.8%]
40%	23.88 / 22.71–25.09	25.01 / 23.77–26.42	1.6%	[0.8%, 9.4%]

In the 0.1%–1% scenario, the effect range of M2 is consistently below 0, indicating relatively stable operational benefits under localized corrections. Once the change rate reaches 2%, the range begins to cross 0, and the current sample size is insufficient to support stable directional differences. In the 40% scenario, the range is entirely above 0, indicating that when the change range approaches the refresh range of M1, the dependency determination and fine-grained writing of M2 become additional costs.

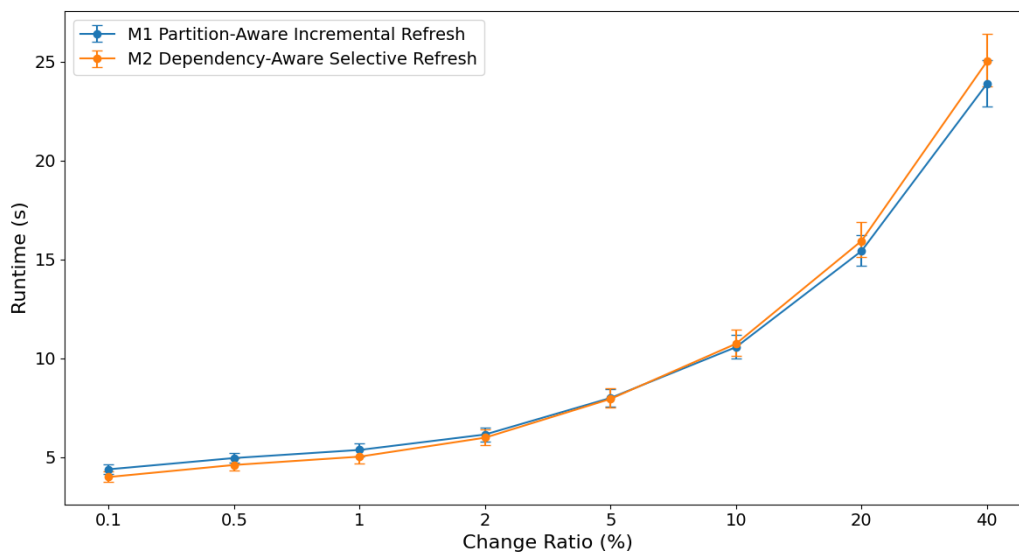


Figure 1. Running time of M1 and M2 under different change ratios

As shown in Figure 1, the running times of the two methods gradually become similar as the

proportion of change increases. The advantage of M2 is mainly concentrated in low-proportion, low-coverage scenarios, and it does not maintain a fixed acceleration as the amount of change increases.

The phase profile for the 1% change scenario provides a more direct cost structure: the median percentages of change application, dependency matching and propagation, SQL scan aggregation, and writes and commits in M2's total runtime are approximately 8.7%, 4.5%, 64.6%, and 22.2%, respectively. These percentages exclude warm-up and are calculated by first taking the median of technical repetition, and then aggregating by change set and month. In actual operation, the newly added dependency calculations in M2 are not the main time-consuming aspect; the benefits depend more on whether the system can truly reduce scanning, aggregation, and result writes.

3.3 Impact on Coverage and Workload Boundaries

Under the condition of a fixed 1% update amount, the actual AC of the four local workloads L1-L4 are approximately 19.6%, 38.4%, 61.7% and 81.2%, respectively, and the runtime change of M2 relative to M1 is approximately -10.7%, -5.1%, -0.9% and +3.6%.

These results demonstrate that the change ratio at the source cannot solely explain incremental maintenance costs. If the same 1% change is concentrated in a small number of dates and regions, M2 can exclude more irrelevant aggregations; as the change spreads across the entire month and more regions, the number of aggregation keys requiring maintenance quickly approaches that of M1. AC describes this difference more directly than the Change Ratio, but it is still not a complete performance predictor because physical sorting, row-group pruning, and the number of local writes also affect actual time consumption.

Monthly Append provides clearer boundaries. New month records involve date, region, payment method, and amount, requiring updates for most analytical aggregations. At this point, M1 has already avoided recalculating historical data through date partitioning, and M2's continued field and grouping decisions have not generated stable additional benefits.

3.4 Visual Analytics Applications and Semantic Consistency

All three analysis tasks directly read the corresponding aggregated objects. As shown in Table 3, T1 observes daily travel volume, T2 compares regional income and the Top-10 regions, and T3 analyzes payment structure; the three views form a linked analysis by sharing date and regional filtering status.

Table 3 Consistency between Visual Analytics Tasks and Analytics

Task	Visual encoding and interaction	Query median	Consistency results
T1 Time Trend	Line chart; Date selection	0.31 s	COUNT consistent; sequence MARE≈0
T2 area income	Hierarchical color scheme + Top-10; Regional linkage	0.38 s	Top-10 overlaps 10/10; Jaccard=1.00; Spearman=1.00
T3 payment structure	100% stacked column chart; date/region filtering	0.29 s	The median MARE is approximately [value missing] 2.9×10^{-11} , and P95 is approximately [value missing]. 4.8×10^{-11}

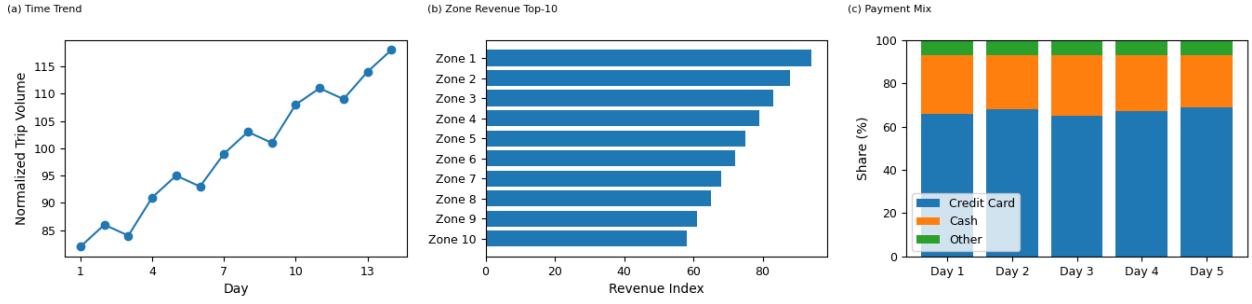


Figure 2. Data warehouse-driven three-view visual analytics structure (illustrated)

Figure 2 shows three interconnected views: time trend, top 10 regional revenue, and payment structure. This figure is only used to illustrate the correspondence between warehouse aggregation and downstream analysis.

The discrete aggregation results for M0, M1, and M2 remain consistent, with differences in floating-point indices only on the order of machine precision. This indicates that selective refresh does not trade runtime for necessary updates by omitting them. In the out-of-time window from January to May 2026, some low-proportion corrections still show faster performance for M2, but inter-month fluctuations increase, and the effect range also crosses zero. Therefore, this study does not interpret this phenomenon as a stable out-of-time performance gain.

4. Discussion

in this study is not that "M2 is always faster than M1," but rather that field-partition dependencies can reduce logically unnecessary aggregate keys. To translate this logical reduction into practical operational benefits, the following conditions also need to be met:

$$T_{\text{saved,query}} + T_{\text{saved,write}} > T_{\text{match}} + T_{\text{prop}} + T_{\text{extra,write}}$$

In low-coverage local corrections, M2 reduces the scanning and aggregation range, thus logical pruning translates into wall clock benefits. As changes gradually spread, the objects that M1 and M2 need to maintain tend to be consistent, but M2 still bears the costs of change matching, dependency propagation, and finer-grained writes. This mechanism explains the phenomenon in Table 2 where the RRR remains positive, but the runtime advantage gradually disappears.

Compared to existing research on incremental view maintenance, this study does not propose a new general incremental algebra. Instead, it focuses on analytical data warehouses, unifying field changes, date partitioning, old and new grouping, and analytical aggregation keys into refreshable units. Therefore, the contribution of this study lies in the fine-grained maintenance strategy for incremental ETL and its workload boundaries.

this study, visual analytics serves only for application constraints and semantic verification. The three tasks define three materialized granularities: day, date $\times$ region, and date $\times$ payment, respectively, becoming independent refresh units for M2, thus forming a research chain of "analysis granularity—maintenance granularity." This study does not further interpret sub-second response times as an improvement in user satisfaction or decision quality.

Statistical evidence also has clear boundaries. The main experiment only covers 12 independent months, and 10,000 bootstraps can only describe the uncertainty of the results for these months, without increasing the independent sample size. When the confidence interval crosses 0, this study can only conclude that "no stable directional evidence was obtained," and cannot interpret it as the two methods being equivalent. The NYC Yellow Taxi case is a single-fact table, single-node

analysis scenario; therefore, the conclusions of this study cannot be directly extrapolated to multi-fact table, MPP data warehouse, or high-concurrency transaction environments.

5. Conclusion

This study addresses the continuous maintenance of data warehouses for visual analytics workloads by constructing a star schema and three fixed analytics aggregates, and examines whether field-partition dependencies can further reduce the incremental ETL refresh range while maintaining complete refactoring semantics.

The research results indicate that selective refreshing is more suitable for localized corrections with a concentrated impact range that can effectively reduce scanning and aggregation work. As changes spread, the refresh set gradually approaches the date partition incremental baseline, and fine-grained dependency analysis no longer provides stable benefits. High-coverage workloads such as Monthly Append demonstrate that the finest-grained maintenance is not a universally optimal strategy for continuous ETL in data warehouses.

Therefore, in practical systems, it is more suitable to select the maintenance granularity by combining the type of change, impact coverage, scan range, and write scale. The field change-aggregate key mapping established in this study provides an executable basis for this selection, while the visual analysis task clarifies the downstream business semantics that incremental maintenance needs to maintain.

References

- [1] Walha A, Ghazzi F, Gargouri F. *Data integration from traditional to big data: main features and comparisons of ETL approaches*[J]. *The Journal of Supercomputing*, 2024, 80(19): 26687-26725. DOI: 10.1007/s11227-024-06413-1.
- [2] Khan B, Jan S, Khan W, Chughtai M I. *An overview of ETL techniques, tools, processes and evaluations in data warehousing*[J]. *Journal on Big Data*, 2024, 6(1): 1-20. DOI: 10.32604/jbd.2023.046223.
- [3] Setiawan D, Isa S M. *Improving Data Warehouse ETL Efficiency with Partitioning and Parallelism in Financial Services*[J]. *Jurnal Sains dan Teknologi*, 2025, 14(3): 590-599. DOI: 10.23887/jst-undiksha.v14i3.103391.
- [4] Zhang, Z. (2026). *Research on Service-oriented Packaging of AI Systems and Data-driven Decision Support for Real-world Business Processes*. *Engineering Advances*, 6(3).
- [5] Zhang, Y. (2025, January). *Research on the Application and Optimization of Multi Dimensional Data Model Based on Kylin in Enterprise Technology Management*. In *International Conference on Innovative Computing* (pp. 234-241). Singapore: Springer Nature Singapore.
- [6] Fei, W., Xue, W., & Wang, B. (2025). *Research on the innovative path of rural education teaching methods from the perspective of PBE and PBL integration*. *Int. Acad. J. Humanit. Soc. Sci.*
- [7] Zhang, Z. (2026). *User Behavior Anomaly Identification Integrating Telemetry Data and Its Application in Consumer Finance Risk Control*.
- [8] Zhang, Y. (2026). *Exploration of Enterprise Big Data Microservice Architecture Based on Domain-Driven Design (DDD)*. *Procedia Computer Science*, 282, 1994-2003.
- [9] Cui, N. (2024, December). *Research on Adaptive Urban Traffic Green Wave Control Algorithm and Safety Based on Reinforcement Learning DQN-COP*. In *International Conference on Big Data Analytics for Cyber-Physical System in Smart City* (pp. 379-387). Singapore: Springer Nature Singapore.

- [10] Liu, S. (2026). Automatic Generation Method and Implementation of Web Front-End Components Based on Generative AI.
- [11] Wendan Xue, Wen Fei. An Analytical Study on the Differences and Determinants of Basic Education Resource Allocation in Rural China and the United States. *International Journal of Educational Research and Development*, 1(3), 13-17.
- [12] Jingzhi Yin. (2026). Constructing a CMBS Default Risk Sentiment Index Using Financial Text Embeddings and Evaluating Its Predictive Effectiveness. *Engineering Advances*, 6(1), 50-54.
- [13] Wang, Z. (2026). Research on Operation Data Analysis and Efficiency Improvement of the New Energy Industry Based on Machine Learning. *Procedia Computer Science*, 285, 605-612.
- [14] Zhang, Z. (2026). User Behavior Device Reliability Joint Modeling and Intelligent Risk Control Practice In Consumer Technology.
- [15] Wang, B. (2025, November). Research on Performance Improvement of Personalized Recommendation System for E-commerce Platform Based on Big Data and Collaborative Filtering Algorithm. In *International Conference on Computational Technologies for Research in Data Analytics* (pp. 36-46). Cham: Springer Nature Switzerland.
- [16] Liu, S. (2026). Design and Implementation of Retrieval Enhancement Generation Method for Enterprise Knowledge Base.
- [17] Fei, W., & Qian, J. (2026). Application and Exploration of Reverse Instructional Design in the Teaching of the Primary School Physical Education" Walking and Playing" Unit. *Journal of Sociology and Education*, 2(2).
- [18] Boli Wang, Wen Fei. Innovations and practices in the classroom for rural aesthetic education in the perspective of school integration. *International Journal of Educational Research and Development*, 3(1), 36-41.
- [19] Hou, Y. (2026, March). Adaptive Upgrading of Computing Power Platforms for Large Model Workloads. In *International Conference on Computing and Communication Systems for Industrial Applications* (pp. 429-437). Cham: Springer Nature Switzerland.
- [20] Xin, H. (2026). Research on Distributed Data Pipelines and Event-Driven Architecture for Freight Visibility.
- [21] Jin, C. (2026). Application of Data Mining Technology in Predicting User Churn and Retention Strategies in E-Commerce.
- [22] Wen Fei, Jianrong Qian. Research on Value Guidance and Practical Pathways for Constructing the Professional Culture of Teachers. *International Journal of Educational Research and Development*, 2(4), 27-30.
- [23] Qian, X. (2026, July). Application of the Combination Model of Prophet and LSTM in Multi-Level Inventory Demand Prediction for Retail Supply Chain of CPG-Consumer Packaged Foods. In *2026 5th International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)* (pp. 1-7). IEEE.
- [24] Jin, C. (2026). Construction and Empirical Study of Credit Risk Assessment Index System for Small and Micro Enterprises in Commercial Banks.