An Energy-Aware SQL Database System

by

Peixiong He

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
August 8, 2026

Keywords: Data Mining, SQL Database, Energy Saving

Approved by

Xiao Qin, Chair, Professor of Computer Science and Software Engineering
Yi Zhou, TSYS School of Computer Science, Columbus State University
Wei-Shinn (Jeff) Ku, Professor of Computer Science and Software Engineering
Samuel Mulder, Associate Professor of Computer Science and Software Engineering
Pan He, Assistant Professor of Computer Science and Software Engineering
Hao Chen, Assistant Professor of Forest Genomics

Abstract

As enterprise data volumes continue to grow, the energy consumption of large-scale database systems has become an important concern alongside performance and reliability. Existing energy-efficient database approaches, including PRE-BUD, Dynamic Power Management, and GreenDB, reduce energy use through power-state management. However, these systems often rely on simple frequency-based heuristics for hot-data placement, which rank tables by individual access frequency. This strategy becomes less effective in multi-tenant and multi-workflow environments, where queries often access coherent groups of related tables rather than isolated high-frequency tables.

This dissertation presents GreenDB_DM, an energy-aware SQL database system that uses workload-aware data mining to improve table placement and reduce unnecessary cold-node wakeups. The first component is GARMT, a grouping-based association rule mining approach for SQL workload analysis. GARMT partitions SQL query logs into fixed-size groups and applies a modified FP-Growth algorithm, GFP-Growth, to discover frequent table co-access patterns. By aggregating consecutive SQL queries before mining, GARMT reduces the sparsity of individual statements and enables more meaningful workload-pattern discovery. Experimental results show that grouping reduces pattern-mining runtime by up to 40% compared with ungrouped baselines while preserving rule quality.

The second component is a complementary study of dynamic data-driven modeling in recommender systems. Although it addresses a different application domain, this study supports the dissertation's broader methodological argument that system efficiency can be improved by modeling temporal and structural behavior rather than relying only on static metrics. Using a network-based Susceptible-Infected model with time-varying transmission rates, the study shows how recommendation strategies produce different propagation patterns over time.

The central contribution is GreenDB_DM, which integrates GFP-Growth-based pattern discovery with a greedy set-cover placement algorithm to maximize full-query hit rate. By identifying frequently co-accessed table groups and placing complete groups on the hot node, GreenDB_DM increases the likelihood that entire queries can be served without activating cold nodes. Evaluation results show that GreenDB_DM improves full-query hit rate by 5–15 percentage points over the strongest placement baseline and reduces total cluster energy by an additional 2.4–4.7% beyond GreenDB.

Together, these contributions show that energy-aware SQL database systems should not only manage hardware power states, but also exploit workload structure. This dissertation establishes workload-aware data mining as a practical foundation for building more adaptive and energy-efficient SQL database systems.

Artificial Intelligence (AI) Use Disclosure Statement

In the preparation of this dissertation, Artificial Intelligence (AI) tools were used to assist with document organization, language editing, LaTeX conversion, figure and table formatting, and dissertation review. The author acknowledges full responsibility for the intellectual content of this work and has ensured that all AI-assisted sections have been reviewed and revised for accuracy and appropriate academic style. All AI-assisted content was reviewed and validated for relevance, appropriateness, and accuracy before incorporation into the final document to maintain scholarly integrity of this research.

Digital Accessibility Disclosure Statement

In the preparation of this dissertation, digital accessibility tools and manual review procedures were used to support compliance with accessibility requirements. These procedures included review of document structure, captions, tables, figures, and PDF output where feasible. The author acknowledges full responsibility for the intellectual content of this work and has made a good-faith effort to comply with digital accessibility requirements in publishing, wherein the nature of the content does not significantly change in order to do so. Furthermore, the document should be reviewed and revised to meet final accessibility requirements prior to final publication.

Acknowledgments

Time flies. My nine-year academic journey at Auburn is quietly coming to an end with the arrival of another summer. Looking back on this precious and meaningful journey, my heart is filled with gratitude. I am thankful for all the bright and kind people I have met along the way, for your generous help and support, and for the ways in which you have shaped me into the person I am today. Above all, I am grateful for God's grace.

First and foremost, I would like to express my deepest gratitude to my advisor, Dr. Xiao Qin. Dr. Qin has provided me with comprehensive guidance and training in both research and scholarship. From the initial development of my dissertation topic, to model design, experimental analysis, and manuscript revision, he has continuously helped me grow into a better researcher. When my experiments did not progress smoothly, Dr. Qin always encouraged me and helped me identify areas for improvement. When my productivity was low, he cared about my physical and mental health and shared his own experiences to help me adjust my pace and improve my efficiency. When I felt uncertain during the job search process, he offered me unconditional help and support, beginning with revising my application package. Dr. Qin's influence on me extends far beyond academics. His professional attitude, integrity, and care for students will continue to inspire me in my future career and life.

I am also tremendously grateful to my committee members, Dr. Wei-Shinn (Jeff) Ku, Dr. Samuel Mulder, Dr. Pan He, and Dr. Yi Zhou, for reviewing my proposal and dissertation and for providing many valuable and constructive suggestions. Their insightful feedback substantially improved the quality of my dissertation. I would also like to express my sincere appreciation to Dr. Hao Chen for serving as my university reader and for his support of my dissertation work.

I feel very fortunate to have shared my lab experience with my labmates, Libo Sun, Zewei Zhang, and Liangliang Xu. Thank you for creating such a harmonious, supportive, and enjoyable lab environment, for helping me in both research and life, and for making

sure there was always an endless supply of Diet Coke in the lab. I am also grateful to my friends, who helped relieve my stress and brought me many wonderful memories throughout this challenging journey. I would like to thank Mingyuan Chen, Nan Yan, and Xiaoniu Du for helping me fall in love with basketball again during some of my most difficult times. I would also like to thank Jiaqi Wang, Boyuan Wang, and Zijie Zhao for inspiring and helping me become a more disciplined person.

Finally, I would like to express my deepest and everlasting gratitude to my family: my father, Kanglin He; my mother, Yun Li; my grandfather, Jianzhong He; my grandmother, Airu Yang; my father-in-law, Min Cui; and my mother-in-law, Mingjie Li. Thank you for raising me, for your selfless love, and for your unconditional support throughout my life. Being part of this family is one of the greatest blessings and treasures of my life. I would also like to thank my younger brother, Peihua He, for accompanying and caring for our parents while I was away from home. Most importantly, I would like to express my most heartfelt gratitude to my wife, Meiying Cui. Thank you for standing by me throughout this journey, for taking such thoughtful care of me, and for giving me unwavering love and support. Because of you, I was able to focus on my studies and complete this degree. Your love and companionship have been the gentlest and strongest source of strength throughout this journey.

I am grateful for everything I have experienced during these nine years at Auburn. This place has witnessed my growth, struggles, and transformation, and it has truly become my second hometown.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In modern data management systems, structured query language (SQL) databases, as a mature and stable technology, have become the standard for processing structured data. These databases ensure data integrity through strongly typed schema definitions and support complex transaction management and efficient query processing capabilities. User interaction with the database primarily occurs through a series of SQL queries collectively stored as query workloads. These workloads encompass various types of SQL statements, including data retrieval and updates, and they support complex transaction management through efficient query processing.

As the volume of SQL queries continues to grow, certain tables experience significantly higher access frequencies, which not only increases the database system's overall energy consumption [82] but also exacerbates data sparsity [77]. This elevated access frequency can further degrade system performance by prolonging response times. Consequently, optimizing queries, judiciously planning table indexes, and adopting efficient data storage and access strategies are essential to reduce energy consumption and improve the operational efficiency of database systems.

Modern data centers consume an estimated 1–2% of global electricity, with database clusters constituting one of the largest contributors due to their continuous availability requirements and high I/O intensity [7, 50]. As enterprise data volumes continue to grow, the energy footprint of large-scale database deployments has become a primary operational concern alongside performance and reliability. This trend has motivated extensive research on energy-efficient database systems, including disk-level techniques such as PRE-BUD [72], node-level Dynamic Power Management (DPM) [9], and more recently

the GreenDB framework [82], which partitions a cluster into one always-active hot node and multiple low-power cold nodes, using prefetching to keep cold nodes in standby for as long as possible. These systems demonstrate that substantial energy savings can be achieved through careful power-state management.

The explosion of online information and digital content today has greatly exacerbated the information overload problem faced by users. This trend highlights the importance of recommender systems in filtering and pushing relevant content, which becomes a key tool for guiding users' attention, increasing interaction frequency, and prolonging platform dwell time [38]. However, despite the remarkable achievements in the industry, we still lack a deep and systematic theoretical understanding of how recommended content diffuses in the user community and how users interact with each other to facilitate clicking or consuming behaviors. Previous studies have focused on improving recommendation accuracy [24], optimizing sorting strategies [67], or introducing contextual information to enhance user satisfaction [45], but these approaches are often based on static evaluation metrics, ignoring the temporal characteristics of recommended content spreading in dynamic user networks and the changing process of user engagement [43].

1.1 Motivations

Building on the above background, this dissertation is motivated by three related challenges: sparse SQL workload representation, limited dynamic modeling of information diffusion, and inefficient table placement in energy-aware database clusters. Among these challenges, SQL workload mining serves as the methodological foundation, while energy-aware database optimization represents the main system-level application.

Data sparsity—where most fields in large table sets remain unused by most queries—leads to inefficiencies in access optimization. In SQL databases, there may be a large number of different tables and fields, but any single query may involve only a small fraction

of them. This results in highly sparse data, i.e., most items do not appear in most transactions. Because SQL statements are generally executed line by line, each statement provides only limited information—particularly regarding table usage and data access patterns. Consequently, capturing comprehensive database interaction patterns from a single SQL statement can be exceptionally difficult.

This limitation motivates the need for a grouping-based representation of SQL workloads. To address this limitation, we propose a statement-grouping strategy that aggregates related multi-line SQL statements into a set. The core idea is to treat multiple SQL operations within a time window or logically interrelated SQL operations as a whole, thereby increasing the density and relevance of data items and mitigating the challenges posed by sparse data. By clustering query workloads and applying frequent-pattern mining, this method uncovers hidden access patterns, enabling database administrators to implement targeted optimizations that improve system performance while reducing energy consumption.

As a secondary line of investigation, this dissertation also considers how dynamic modeling can be applied beyond database workloads. Beyond SQL workload analysis, data-driven modeling techniques can also be used to understand how information and recommendations propagate in user networks. Previous studies have focused on improving recommendation accuracy, optimizing sorting strategies, or introducing contextual information to enhance user satisfaction, but these approaches are often based on static evaluation metrics, ignoring the temporal characteristics of recommended content spreading in dynamic user networks and the changing process of user engagement. In fact, a recommendation algorithm is not only a content matching process, but also a complex information diffusion process with obvious propagation and evolution. Therefore, it is of theoretical and practical significance to understand how recommendation strategies diffuse content across user networks from a macro perspective.

The insights obtained from workload mining and pattern discovery naturally lead to a more practical system-level question: how can discovered access patterns be used to reduce energy consumption in database clusters? At the system level, a common assumption underlying existing energy-efficient database approaches is that determining which tables to place on the hot node is a secondary concern, typically addressed using a simple frequency-based heuristic: ranking tables by access count and placing the top-K tables on the hot node up to its capacity. While effective under workloads dominated by a small subset of tables, frequency-based placement implicitly assumes a highly skewed table-access distribution. This assumption breaks down in two increasingly common scenarios: multi-tenant SaaS deployments, where multiple customers share a database without a dominant tenant; and multi-workflow analytical workloads, where several business processes run concurrently, each touching its own group of related tables.

This observation motivates a shift from individual table frequency to group-aware table placement. Our key observation is that real database queries rarely access tables in isolation; instead, a single query typically touches a group of semantically related tables that together support a specific workflow. If such co-access groups can be discovered automatically and placed jointly on hot nodes, every query within that workflow becomes a full hit, thereby eliminating the cold-node wakeups that frequency-based placement incurs whenever it places only a subset of a group's tables on the hot nodes. This insight reframes hot-node placement from a per-table ranking problem into a group-aware coverage problem.

1.2 Contributions

To address the above challenges, this dissertation develops a sequence of related methods and systems. The contributions progress from foundational SQL workload mining, to a complementary study of dynamic propagation modeling, and finally to an energy-aware database placement framework. Fig. 1.1 summarizes these three primary components and highlights the distinct role of each contribution within the overall dissertation.

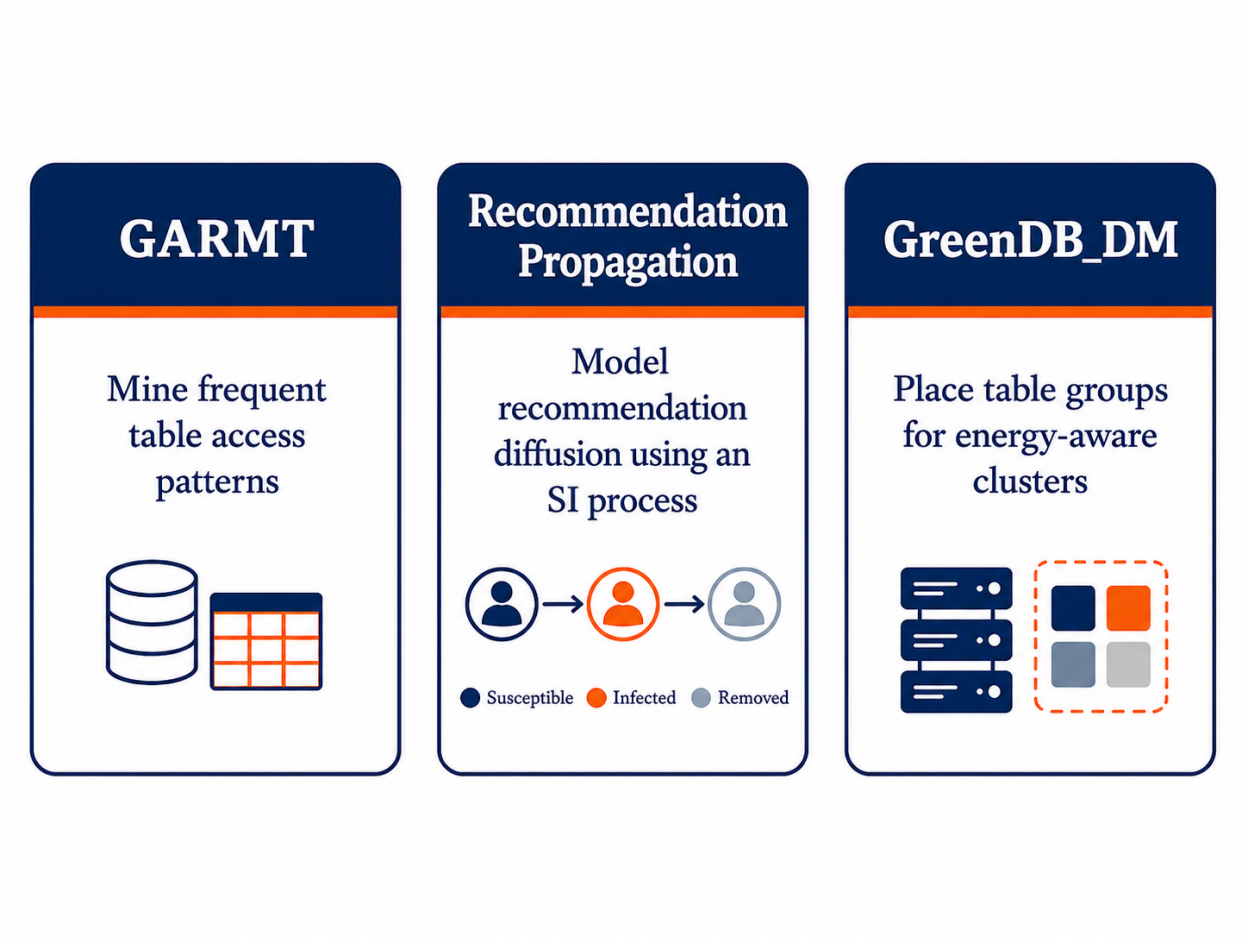

Figure 1.1: Overview of the three primary components developed in this dissertation.

As shown in Fig. 1.1, the first component establishes the workload-mining foundation for the dissertation. Specifically, this dissertation proposes a grouping-based association rule mining approach that rapidly identifies tables and fields with high access frequency.

By clustering query workloads and applying frequent-pattern mining, this method uncovers hidden access patterns, enabling database administrators to implement targeted optimizations that improve system performance while reducing energy consumption. The core idea is to treat multiple SQL operations within a time window or logically interrelated SQL operations as a whole, thereby increasing the density and relevance of data items and mitigating the challenges posed by sparse data. Second, this dissertation identifies and exploits relationships and patterns among tables that may be overlooked in traditional SQL log analysis. This research offers novel strategies and tools for database performance optimization and resource management. It theoretically and empirically demonstrates the computational advantages of grouping, including a detailed complexity analysis and runtime evaluations. Experiments on a real-world dataset show that grouping significantly reduces runtime compared to the ungrouped baseline while preserving rule relevance. These results highlight the practical value of query grouping for efficient pattern discovery in sparse database environments.

In addition to SQL workload mining, this dissertation includes a complementary investigation of dynamic data-driven modeling in recommender systems. Third, as a complementary data-mining application, this dissertation introduces an epidemiological modeling perspective into the study of recommender systems and constructs an analytical framework that can dynamically simulate the process of content diffusion. This work considers the recommendation process as analogous to the spread of an infectious disease in a social network. In a scale-free network environment constructed based on real clickstream data, the system compares the performance of classical recommendation strategies in terms of propagation efficiency. This part of the dissertation provides an additional perspective for understanding dynamic user behavior, while the main technical focus remains on SQL workload mining and energy-aware database optimization.

The final and central system-level contribution applies the discovered workload patterns to energy-aware database clusters. Fourth, this dissertation identifies a previously overlooked limitation of frequency-based placement in node-level energy-efficient database clusters and characterizes the workload regimes under which this limitation becomes significant. It proposes a group-aware hot-node placement framework that combines GFP-Growth pattern mining with a set-cover-based placement algorithm and integrates it with the GreenDB five-case energy model without altering the underlying energy accounting framework.

Finally, this dissertation validates the proposed methods through experimental evaluation and deployment-oriented analysis. Fifth, this dissertation conducts extensive evaluations against placement baselines and energy-management baselines on controlled synthetic workloads spanning a wide range of skewness, group coherence, cluster scale, and data size. It provides practical deployment guidance by characterizing the conditions under which group-aware placement provides the greatest benefit and those under which frequency-based placement remains near-optimal.

1.3 Organization

The rest of this dissertation is organized as follows. Chapter 2 reviews the related literature needed to position this work within existing research. It covers frequent-pattern mining, SQL workload analysis, recommender systems, information diffusion models, and energy-efficient database systems.

After the literature review, the dissertation turns to the first major technical component: SQL workload mining. Chapter 3 presents the grouping-based association rule mining approach for SQL workload analysis. This chapter introduces how SQL query workloads can be grouped and mined to discover frequent table access patterns, which can support query optimization, reduce system load, and provide useful knowledge for energy-aware database management.

The dissertation then briefly extends data-driven modeling to a complementary application domain. Chapter 4 presents a study that integrates classical recommendation algorithms with a network-based SI model incorporating time-varying infection rates. This chapter provides an additional perspective on dynamic behavior modeling, while the main technical focus of the dissertation remains on SQL workload mining and energy-aware database optimization.

The final technical part of the dissertation returns to the main system-level objective of energy-aware SQL database systems. Chapter 5 presents the design and evaluation of GreenDB_DM, including frequent table-group discovery, group-aware hot-node placement, and integration with the GreenDB energy model. This chapter demonstrates how mined workload patterns can be used to improve full-query hit rate and reduce unnecessary cold-node wakeups.

Finally, Chapter 6 summarizes the key contributions of this dissertation and discusses how the proposed methods support the development of an energy-aware SQL database system. Chapter 7 outlines potential future research directions, including AI-driven table prediction, intelligent access forecasting in non-relational databases, and AI-enhanced energy optimization for data systems.

Chapter 2

Related Work

The main goal of frequent-pattern mining is to discover frequently occurring patterns that reveal the intrinsic structure and relationships of data. These patterns can enhance our understanding of data features and support more effective classification, clustering, and prediction.

2.1 Frequent-pattern mining

Frequent-pattern mining focuses on identifying and selecting valuable patterns through evaluation metrics such as support, confidence, and lift. Support measures the frequency of an itemset appearing in the entire dataset, defined as

$$\text{Support}(\text{X}, \text{Y}) = \frac{\text{Frequency of } (\text{X}, \text{Y})}{\text{Total number of samples}},$$

which aids in understanding the prevalence of the pattern in the dataset [28].

Confidence measures the probability of another itemset occurring given that a previous itemset has occurred, expressed as

$$\text{Confidence}(\text{X} \Rightarrow \text{Y}) = \frac{\text{Frequency of } (\text{X}, \text{Y})}{\text{Frequency}(\text{X})},$$

assessing the dependency relationships between itemsets [32].

Lift evaluates the relevance of itemsets, calculated by

$$\text{Lift}(\text{X} \Rightarrow \text{Y}) = \frac{\text{Support}(\text{X}, \text{Y})}{\text{Support}(\text{X}) \times \text{Support}(\text{Y})}$$

and compares the probability of itemset X occurring given itemset Y to the probability of X occurring independently [12]. These metrics not only help researchers identify significant patterns in the data but are also crucial for interpreting the data relationships behind the patterns. Frequent pattern mining has been applied to a range of systems problems beyond market-basket analysis [27], including web access prediction [53], storage-system prefetching [44], and query optimization for OLAP workloads [4]. These studies share with ours the use of mined patterns to inform proactive system decisions, but none targets cluster-level energy efficiency.

2.2 FP-Growth

The FP-Growth (Frequent-Pattern Growth) algorithm is a highly efficient method for discovering frequent itemsets in large datasets without generating a multitude of candidate itemsets [11]. The algorithm relies primarily on two database scans and a compressed data structure called the FP-tree (Frequent-Pattern Tree) to achieve performance gains. During the first scan, the algorithm identifies and counts the frequency of individual items, retaining only those that meet a specified minimum support threshold. These frequent items are then sorted in descending order of frequency to form the item header table. In the second scan, each transaction is read, filtered to include only frequent items, reordered according to the item header table, and inserted into the FP-tree. If the corresponding path in the tree already exists, its count is incremented; if not, a new path is created. Subsequent mining of the FP-tree is performed by recursively constructing conditional pattern bases and conditional FP-trees for each item in the item header table. Each conditional FP-tree represents a frequent pattern under a specific itemset prefix, and this process continues until no further frequent itemsets can be identified. By avoiding the generation of large numbers of candidate itemsets and minimizing the number of database scans,

FP-Growth significantly reduces computational complexity and execution time, thereby offering a faster and more efficient approach for uncovering important association rules in large datasets.

2.3 Recommendation Algorithms

The last twenty years have seen a significant rise in the development of recommender systems that primarily work to deliver personalized content suggestions for improving user experience. In general, recommendation methods can be categorized into three main types: collaborative filtering methods along with content-based recommendation methods and hybrid recommendation methods [38]. Matrix factorization within collaborative filtering methods makes use of historical user–item interactions to predict future preferences based on past similarities between users [52]. Content-based recommendation systems utilize characteristics from items and user profiles to suggest items that resemble those the user previously liked. Hybrid methods combine two techniques [8] in order to enhance precision while solving typical problems like cold-start challenges and incomplete data. The algorithms achieve success but they receive evaluations based solely on static performance measures like precision, recall and click-through rate, which fail to consider the constantly shifting patterns of user behavior and engagement.

2.4 Epidemic Models

Epidemiological models establish a systematic method to simulate disease transmission dynamics across a population throughout time [37]. The primary models used in epidemiology are the SI (Susceptible-Infected) [35], SIR (Susceptible-Infected-Recovered) [69], and SEIR (Susceptible-Exposed-Infected-Recovered) frameworks [68]. The SI model defines a process where susceptible individuals transition into an infected state and stay infected forever. The model excels at describing irreversible cumulative

processes like user engagement in recommender systems because user interactions with content create permanent behavioral effects. The SIR model includes the Recovery (R) state, which enables individuals to leave the infected state, thus modeling decaying or saturating interest. The SI model's straightforward nature and alignment with continuous digital behavior patterns make it especially appropriate for our study despite the more detailed dynamics in SIR and SEIR models. This approach facilitates straightforward representation of recommendation distribution across user networks without presuming disengagement or behavioral decline, which might not fit persistent recommendation scenarios.

2.5 Information Diffusion Models in Social Networks

The study of information dissemination among social network users remains a fundamental area of exploration across computational social science and recommender systems research fields [47]. Researchers have developed multiple information diffusion models to explain how users adopt behaviors or interact after being influenced by their peers. The Independent Cascade model [71] and Linear Threshold model [16] stand out as the two primary examples in the field of information diffusion modeling [43]. According to the Independent Cascade model, information propagation from an active node to its neighboring nodes occurs independently with a constant probability during the initial attempt, while the Linear Threshold model dictates that a node activates when the aggregated influence from its neighbors surpasses its threshold value. These two models establish the theoretical groundwork and simulation structure to examine content dissemination as well as user influence ranking and opinion evolution. As research progressed further numerous modifications were introduced to the foundational models, including node characteristics [56], time delay elements, diverse content types, and mechanisms tracking user interest [13] development, which enhanced model adaptability and expressiveness for complex network environments [64]. Research efforts have integrated Graph Neural Networks from

deep learning with diffusion techniques to learn information propagation paths and forecast user responses [81]. Algorithmic advances and theoretical modeling improvements have been made, yet most diffusion models remain concentrated on forecasting propagation range or pinpointing key nodes rather than examining the sequence of user behaviors activated by recommender system signals. The study extends this approach by integrating diffusion mechanisms with recommendation strategies to examine the influence of various recommendation methods on content reach and user engagement online.

2.6 Energy-Efficient Database Systems

Energy management in storage and database systems has been studied across multiple levels of granularity. At the disk level, early work proposed spinning down idle disks to save power, with schemes such as MAID [18] and PDC [58] migrating frequently accessed data to a small subset of always-on disks. PRE-BUD extended this idea by introducing dedicated buffer disks that absorb read traffic to allow data disks to remain in low-power modes for long time periods [72]. These disk-level techniques achieve modest savings but fail to exploit opportunities to power down entire nodes, leaving CPU, memory, and network components consuming unnecessary energy.

At the node level, Dynamic Power Management (DPM) [9, 34] treats each node independently, transitioning to low-power states after fixed idle-time thresholds. Despite its simple deployment, DPM is oblivious to query semantics and frequently incurs "ping-pong" transitions when queries arrive faster than the power-state break-even time. Subsequent work recognized that database queries often exhibit predictable access patterns and proposed semantic-aware power management. For example, Lang and Patel introduced energy-aware query processing [41], while Poess and Nambiar analyzed the energy efficiency of TPC-H workloads [59]. Cluster-level approaches such as Rabbit [3] and Sierra [65] introduced primary/secondary node partitioning to enable longer idle windows on non-primary nodes.

The GreenDB framework [82] unifies these ideas through a one-hot/N-cold cluster architecture and a five-case energy-saving model that quantifies when prefetching cold-node data to the hot node yields net energy benefits. GreenDB demonstrates substantial savings under high hit rates but treats the hit rate itself as an input parameter to its energy model rather than as an optimizable quantity, leaving open the question of how such high hit rates should be achieved through table placement. This study is motivated by the aforementioned observations and directly bridges this gap.

Frequent pattern mining has been applied to a range of systems problems beyond market-basket analysis [27], including web access prediction [53], storage-system prefetching [44], and query optimization for OLAP workloads [4]. These studies share with ours the use of mined patterns to inform proactive system decisions, but none targets cluster-level energy efficiency.

2.7 Table Placement and Caching

Table placement in distributed databases has traditionally been driven by performance objectives such as load balancing, query latency, and join locality. Horizontal and vertical partitioning techniques [55] distribute table fragments across nodes to parallelize query execution, while replica placement strategies [74, 1] balance read availability against write overhead. Workload-aware placement approaches [20, 60] analyze query logs to co-locate frequently joined tables and reduce cross-node communication. While conceptually similar to our group-aware placement in their use of co-access information, these approaches optimize query latency rather than energy efficiency and distribute tables across many homogeneous nodes rather than concentrating them on a single hot node within an asymmetric cluster.

At the buffer pool and cache level, replacement policies such as LRU, LFU, ARC [51], and LIRS [36] determine page residency in fast storage. These policies operate at a fine granularity on individual pages and adapt online to changing access patterns.They are

complementary to our work: GreenDB_DM makes coarse-grained, table-level placement decisions over longer time scales (offline mining of historical logs), while page-level cache policies handle short-term dynamics within hot nodes.Recent learned cache replacement techniques [46] incorporate machine learning into the cache hierarchy, but they primarily target hit rate rather than cluster energy efficiency.

2.8 Positioning of GreenDB_DM

Existing energy-efficient database systems address power management (i.e., state transition decisions) and energy-saving architectures (e.g., hot/cold partitioning). However, they typically rely on frequency-based heuristics for table placement on the hot node. Prior work on frequent pattern mining, including the GARMT algorithm adopted in this study, focuses on efficient discovery of co-access patterns but does not integrate these patterns into system-level optimization. Similarly, existing placement and caching approaches primarily optimize for performance metrics rather than energy efficiency, and they operate on symmetric clusters or individual pages rather than the asymmetric hot/cold node abstraction central to GreenDB. To the best of our knowledge, GreenDB_DM is the first system to integrate these three threads by leveraging frequent pattern mining to guide group-aware table placement within an energy-efficient cluster architecture.

Chapter 3

Grouping-Based Association Rule Mining to Predict Future Tables in Database Queries

User interaction with the database primarily occurs through a series of SQL queries collectively stored as query workloads. These workloads encompass various types of SQL statements, including data retrieval and updates, and they support complex transaction management through efficient query processing. As the volume of SQL queries continues to grow, certain tables experience significantly higher access frequencies, which not only increases the database system's overall energy consumption [79] but also exacerbates data sparsity (i.e., most table fields remain unused across the majority of queries) [77]. Check Figure 3.1). This elevated access frequency can further degrade system performance by prolonging response times. Consequently, optimizing queries, judiciously planning table indexes, and adopting efficient data storage and access strategies are essential for reducing energy consumption and enhancing the operational efficiency of database systems. To address these challenges, we propose a grouping-based association rule mining approach that rapidly identifies tables and fields with high access frequency. By clustering query workloads and applying frequent-pattern mining [33], this method uncovers hidden access patterns, enabling database administrators to implement targeted optimizations that improve system performance while reducing energy consumption. For example, consider a web application that generates multiple SQL statements during a user session—such as a SELECT statement for browsing products, an INSERT statement for updating the shopping cart, and a DELETE statement for cleaning up the session. While each query may appear to be unrelated to the other, grouping them together reveals consistent access to the User, Product, and Cart tables. This contextual grouping reveals more meaningful patterns of common access than processing queries in isolation [62]. In

addition, strong associations between columns (such as foreign keys or co-occurrences in WHERE clauses) may imply deeper table-level relationships. For example, frequent joint filtering of “orders.user_id” and “users.user_id” may indicate a deeper table-level relationship between “orders” and “users”, implying that they are logically related. Brin et al. [12] have shown that column-level associations can uncover hidden relational dependencies in transactional workloads [25].

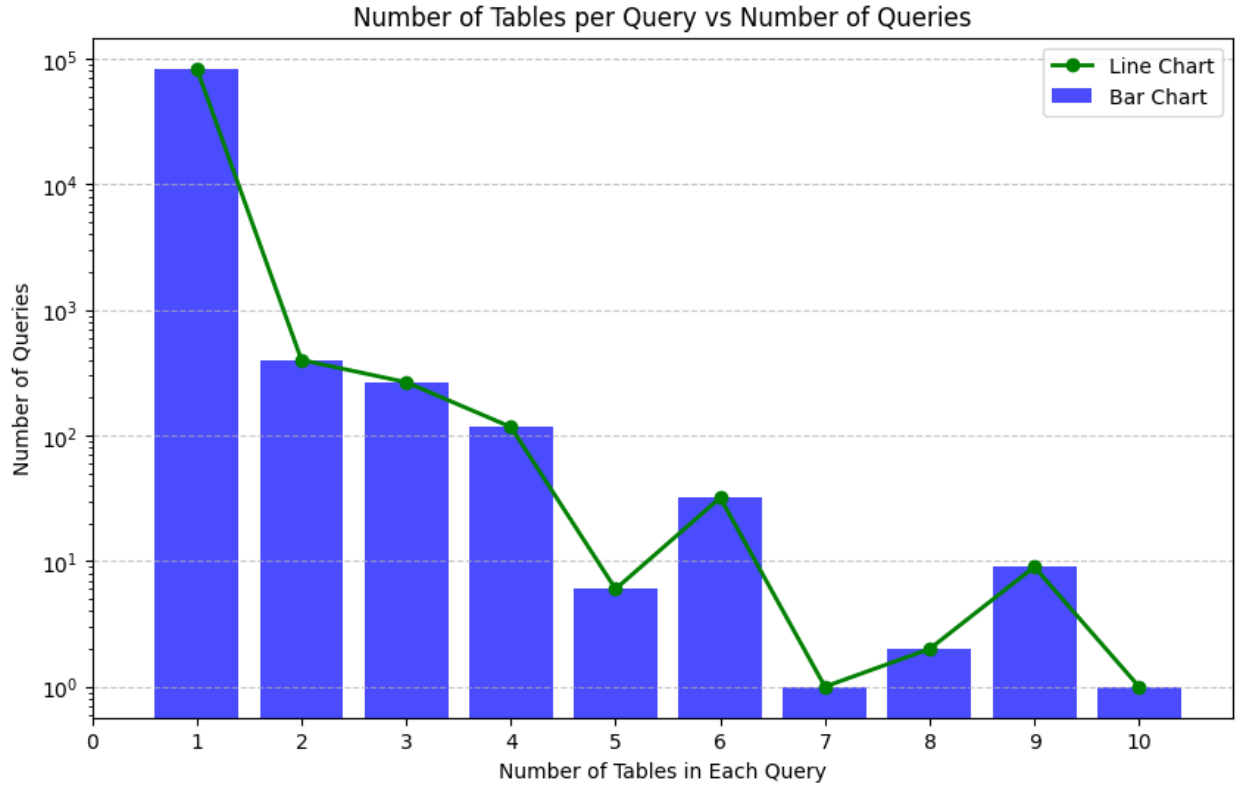

Figure 3.1: In SQL databases, there may be a large number of different tables and fields, but any single query may involve only a small fraction of them. This results in highly sparse data, i.e., most items do not appear in most transactions.

This approach makes three original contributions. First, because SQL statements are generally executed line by line, each statement provides only limited information—particularly regarding table usage and data access patterns. Consequently, capturing comprehensive database interaction patterns from a single SQL statement can be exceptionally difficult. To address this limitation, we propose a statement-grouping strategy that

aggregates related multi-line SQL statements into a set. The core idea is to treat multiple SQL operations within a time window or logically interrelated SQL operations as a whole, thereby increasing the density and relevance of data items and mitigating the challenges posed by sparse data. Second, by identifying and exploiting relationships and patterns among tables that may be overlooked in traditional SQL log analysis, this research offers novel strategies and tools for database performance optimization and resource management. Third, we theoretically and empirically demonstrate the computational advantages of grouping, including a detailed complexity analysis and runtime evaluations. These insights into the interconnections between tables in real-world applications provide crucial support for designing more efficient database systems and improving data processing strategies.

The reminder of this chapter is organized as follows. In Section 3.1, we present an approach to optimize table access using association rule mining, which not only accelerates query processing and reduces system load but also improves performance for complex queries. Section 3.3 introduces a grouping-based data mining algorithm to predict future attributes in SQL databases. In Section 3.4, we verify the algorithm's effectiveness and feasibility by testing its performance under various grouping scenarios. Based on these results, Section 3.5 examines the positive impact of grouping on the generation of association rules. Finally, Section 3.6 summarizes the key contributions of the article and outlines potential avenues for future research.

3.1 Grouping-based Optimization Process

Optimizing table access patterns in SQL databases is crucial for enhancing system performance [15]. This paper introduces a novel optimization method that leverages frequent-pattern mining and predictive modeling to improve table access. The optimization process is structured into several key steps, as outlined below:

Assume a sequence of n SQL queries, denoted as $Q = \{q_1, q_2, \ldots, q_n\}$. We group these queries into sets of k consecutive queries, with each set denoted as G_j, where

$$G_j = \{q_{k(j-1)+1}, q_{k(j-1)+2}, \ldots, q_{kj}\}, \quad j = 1, 2, \ldots, \left\lceil \frac{n}{k} \right\rceil$$

Each G_j contains k consecutive SQL queries.

For each G_j, extract the set of tables accessed by the queries within the group, denoted as T_j:

$$T_j = \bigcup_{q_i \in G_j} \text{Tables}(q_i)$$

Here, Tables(q_i) represents the set of tables accessed by the SQL query q_i.

Apply frequent-pattern mining algorithms such as FP-Growth to the table sets $T_1, T_2, \ldots, T_{\lceil n/k \rceil}$. The frequent itemsets for each algorithm are denoted as F_k:

$$F_k = \text{FrequentPatterns}(T_1, T_2, \ldots, T_{\lceil \frac{n}{k} \rceil}, k)$$

These patterns reveal tables that are frequently accessed together, aiding in the optimization of access patterns.

Using the frequent-pattern set $F_{\text{FP-Growth}}$ from the FP-Growth algorithm, construct a prediction model to anticipate future table accesses. Let t_c represent the currently accessed table. The model computes the conditional probability of the next table to be accessed, t_n, given t_c:

$$P(t_n|t_c) = \frac{\text{Count}(t_c, t_n)}{\text{Count}(t_c)}$$

where Count(t_c, t_n) is the number of times table t_n was accessed immediately after table t_c. The next table to be accessed, t_{pred}, is then predicted as follows:

$$t_{\text{pred}} = \arg\max_{t \in T} P(t|t_c)$$

Once t_{pred} is identified, it is mapped back to its dataset and the corresponding datasets:

$$t_{\text{pred}} \rightarrow D_{\text{pred}}$$

Here, D_{pred} denotes the dataset of the predicted table. This mapping is crucial for optimizing future query operations.

3.2 Problem Statement and Efficiency Analysis

Our primary objective is to allocate computational resources efficiently by minimizing the overall computational complexity [75], which depends on both the size and distribution of query groups. Let n represent the total number of queries and g the designated number of groups. Each group j has a size s_j, with the constraint

$$\sum_{j=1}^{g} s_j = n.$$

The total computational complexity is defined as the sum of the exponential costs across all groups:

$$\sum_{j=1}^{g} O\big(2^{s_j}\big).$$

The primary objective is to minimize the sum of exponential costs, subject to two potential grouping strategies. In the balanced grouping approach, each group size should approximate $\left\lceil \frac{n}{g} \right\rceil$, ensuring a more equitable distribution of queries among the groups. In contrast, unbalanced grouping simply requires that each group must include at least one query, thus allowing for greater flexibility in how queries are distributed.

In pursuit of enhanced performance in SQL database systems, a detailed examination of SQL statement processing reveals substantial opportunities for optimization. The following section introduces an innovative technique that involves grouping SQL statements within datasets before applying frequent-pattern mining algorithms. This method not only

streamlines query processing but also significantly improves the efficiency of identifying recurring patterns, thus facilitating query optimization. By reorganizing data management, this approach enables more effective analysis and faster response times.

Definition 1. *In balanced grouping, the total number of groups g divides the total number of queries n equally or nearly equally:*

Definition 2. *In unbalanced grouping, groups can have different numbers of queries. Let s_j denote the size of group G_j, where $\sum_{j=1}^{g} s_j = n$.*

Theorem 1. *Grouping the SQL statement dataset improves the efficiency of frequent-pattern mining.*

$$G_j = \{q_{k(j-1)+1}, q_{k(j-1)+2}, \ldots, q_{kj}\}, \quad j = 1, 2, \ldots, g, \tag{3.1}$$

where $k = \lfloor n/g \rfloor$.

Proof. We use mathematical induction on the number of groups g.

Base Case ($g = 1$):

When $g = 1$, all queries are in a single group. The computational complexity is calculated as follows:

$$\text{Complexity} = O(f(n)), \tag{3.2}$$

which is the same as the ungrouped case. Thus, the efficiency is the same.

Inductive Hypothesis:

Assume that for $g = m$, grouping the queries into m balanced groups leads to greater efficiency:

$$m \cdot O(f(k)) < O(f(n)), \tag{3.3}$$

where $n = m \times k$.

Inductive Step ($g = m + 1$):

For $n = (m+1)k$, the complexities are as follows:

$$\text{With Grouping:} \quad (m+1) \times O(f(k)), \tag{3.4}$$

$$\text{Without Grouping:} \quad O(f((m+1)k)). \tag{3.5}$$

Since frequent-pattern mining algorithms have exponential or super-exponential complexity, i.e., $O(f(n)) = O(2^n)$, we have:

$$(m+1) \times 2^k \ll 2^{(m+1)k}. \tag{3.6}$$

Thus, grouping reduces the computational complexity significantly. □

Proof. We analyze the total computational complexity when the queries are grouped versus when they are not.

Without Grouping: The computational complexity is

$$\text{Complexity}_{\text{no grouping}} = O(f(n)) = O(2^n). \tag{3.7}$$

With Unbalanced Grouping:

The total computational complexity is the sum of the complexities for each group:

$$\text{Complexity}_{\text{grouped}} = \sum_{j=1}^{g} O(f(s_j)) = \sum_{j=1}^{g} O(2^{s_j}). \tag{3.8}$$

Comparison:

Since the sum of exponentials is less than the exponential of the sum for positive integers,

$$\sum_{j=1}^{g} 2^{s_j} < 2^{\sum_{j=1}^{g} s_j} = 2^n. \tag{3.9}$$

Thus,

$$\text{Complexity}_{\text{grouped}} < \text{Complexity}_{\text{no grouping}}. \tag{3.10}$$

Edge Cases:

Consider the worst-case scenario where one group contains almost all queries, and the others contain minimal queries.

Let $s_1 = n - (g-1)$ and $s_j = 1$ for $j = 2, 3, \ldots, g$. Then,

$$\begin{aligned} \text{Complexity}_{\text{grouped}} &= O(2^{s_1}) + (g-1) \times O(2^1) && (3.11)\\ &= O(2^{n-(g-1)}) + O(g) && (3.12)\\ &< O(2^n) \quad (\text{since } n-(g-1) < n). && (3.13) \end{aligned}$$

Even in unbalanced grouping, the total computational complexity when grouping is applied is less than the complexity without grouping. This holds true regardless of how the queries are distributed among the groups. □

3.2.1 Leveraging Column Associations to Infer Table Associations

In large-scale SQL databases, associations between columns frequently stem from shared business logic or common dependencies. Building on this observation, we propose the conjecture that "column associations can infer table associations". Specifically, when multiple columns exhibit high-frequency co-occurrence or strong correlations during querying or data processing, the tables housing these columns may share underlying connections. Verifying this conjecture not only uncovers potential dependencies at the table level but also presents novel strategies for database optimization and query pattern mining. The details are as follows.

To illustrate this concept more clearly, consider a simple schema with two tables: Customers (customer_id, name, email) and Orders (order_id, customer_id, order_date, total_amount). In a typical workload, queries frequently join Customers.customer_id with Orders.customer_id and also filter by Orders.order_date. This demonstrates a divergent pattern: a single column from Customers (customer_id) is frequently associated with multiple columns in Orders. Conversely, imagine that both Customers.email and Customers.name are often used together to filter by or join with Orders.customer_id. This reflects a convergent pattern, where multiple columns in one table point to a common destination column in another table. These examples provide an intuitive understanding of how frequent-column co-occurrence can help infer hidden table-level relationships.

Consider two tables, T_1 and T_2, with the respective column sets $\{C_1, C_2, \ldots, C_n\} \in T_1$ and $\{C_{n+1}, C_{n+2}, \ldots, C_m\} \in T_2$. Associations among columns across these two tables can be leveraged to infer relationships between the tables themselves. Such inferences manifest in two principal ways. First, in the case of divergence, when a single column in T_1 is associated with multiple columns in T_2, the association propagates from T_1 to T_2. In contrast, in the case of convergence, when multiple columns in T_1 are associated with the same column in T_2, the association from T_1 to T_2 is consolidated. The overall strength of these table-level associations is further determined by the frequency and support of the underlying column associations.

Case 1: Divergence

In the divergence scenario, a single column from T_1 is associated with multiple columns from T_2. For example: $C_1 \in T_1$ is associated with both $C_2 \in T_2$ and $C_3 \in T_2$, denoted as $C_1 \to C_2$, $C_1 \to C_3$

This implies that table T_1 is frequently queried in conjunction with table T_2. The support for these associations is as follows:

$$P(C_2 \mid C_1) = \frac{\text{Support}(C_1 \cap C_2)}{\text{Support}(C_1)}$$

$$P(C_3 \mid C_1) = \frac{\text{Support}(C_1 \cap C_3)}{\text{Support}(C_1)}$$

Given that a single column from T_1 is linked to multiple columns from T_2, we infer that table T_1 is strongly associated with table T_2. The overall probability of accessing T_2 given that T_1 has been accessed is proportional to the sum of these individual column associations:

$$P(T_2 \mid T_1) \propto P(C_2 \mid C_1) + P(C_3 \mid C_1)$$

This scenario reflects a divergent association where information from a single column in T_1 diverges to multiple columns in T_2, creating a strong overall table association.

Case 2: Convergence

In the convergence scenario, multiple columns from T_1 are associated with a single column from T_2. For example: $C_1 \in T_1$ and $C_2 \in T_1$ are both associated with $C_3 \in T_2$, denoted as $C_1 \to C_3$, $C_2 \to C_3$

This implies that multiple columns from T_1 frequently lead to the same column in T_2, consolidating the association between T_1 and T_2. The support for these associations is as follows:

$$P(C_3 \mid C_1) = \frac{\text{Support}(C_1 \cap C_3)}{\text{Support}(C_1)}$$

$$P(C_3 \mid C_2) = \frac{\text{Support}(C_2 \cap C_3)}{\text{Support}(C_2)}$$

Given that multiple columns from T_1 are converging onto a single column in T_2, the association between T_1 and T_2 is consolidated. The overall probability of accessing T_2 given that T_1 has been accessed is the product of these individual column associations:

$$P(T_2 \mid T_1) \propto P(C_3 \mid C_1) \cdot P(C_3 \mid C_2)$$

This scenario reflects a convergent association where multiple columns in T_1 lead to the same column in T_2, forming a strong, consolidated table association.

Both the divergence and convergence cases demonstrate how column-level associations can infer table-level associations. In the divergence case, a single column from T_1 leads to multiple columns in T_2, indicating a broader association between the two tables. In the convergence case, multiple columns from T_1 point to a single column in T_2, consolidating the relationship between the tables.

The overall strength of a table association depends on the frequency of its column associations, which can be quantified through the support and conditional probability of specific column pairs. This column-to-table association framework can be applied to optimize query planning and enhance database caching strategies.

3.3 Algorithm Design

To explore valuable insights from large-scale SQL query logs, we propose an approach based on frequent-pattern mining, which, at its core, utilizes an improved FP-Growth algorithm. The method reveals hidden structural access patterns in database usage by identifying table name combinations that occur frequently in different SQL queries. Unlike traditional FP-Growth, we introduce a “query grouping” strategy, where SQL queries are divided into subsets according to specific criteria, and frequent-pattern mining is performed on each subset separately. As shown in Algorithm 1, this grouping-enhanced algorithm, called "Group FP-Growth" (GFP-Growth), improves processing efficiency, scalability, and pattern recognition accuracy in large-scale data scenarios.

In this study, we adopt a fixed-size sequential grouping strategy. That is, given a query log $Q = \{q_1, q_2, ..., q_n\}$, we divide it into non-overlapping groups G_j where each group contains k consecutive queries based on their original appearance order. Formally, $G_j = \{q_{k(j-1)+1}, ..., q_{kj}\}$ for $j = 1, 2, ..., \lceil n/k \rceil$. No semantic or user-based information is considered in the grouping process. The entire GFP-Growth process is divided into three

key phases, each corresponding to a different task in the data processing pipeline. The first phase is the preprocessing phase, in which we clean and parse the raw query logs and extract key information related to the table name usage; the second phase is the query grouping phase, in which we cluster queries based on their semantic or structural features to form multiple logically similar subsets; and the third phase is the pattern mining phase, in which the FP-Growth algorithm is generated for each query subset. Through this phased and structured process, GFP-Growth effectively improves the efficiency of the algorithm and provides a data-driven decision basis for database optimization and system maintenance.

Algorithm 1. GFP-Growth.

Require: SQL queries as a pandas Series

Ensure: Frequent patterns from table name sets

function `Analyze_SQL`(sql_series)
    Initialize counters for tables, operations, WHERE clauses
for each query in sql_series **do**
        Extract operations, WHERE clauses, tables using regex
        Update counters for tables, operations, WHERE clauses
end for
    return table counts, operation counts, WHERE clause counts
end function
function `Extract_Table_Names`($query$)
    return regex-matched table names after "FROM" keyword
end function
function `Process_SQL_Groups`(df)
for each group in df **do**
        Extract and count table names for all queries in group
end for

16: **return** table sets for groups

17: **end function**

18: Perform FP-Growth to find frequent patterns

Phase I: SQL Query Analysis:

The first function, Analyze_SQL, accepts a pandas Series of SQL queries as input. It initializes counters for three critical components: table names, SQL operation types (e.g., SELECT, UPDATE, DELETE, INSERT), and WHERE clauses. Each query is processed using regular expressions to detect these elements. SQL operations are identified through their respective keywords, WHERE clauses by their syntactical structure, and table names from tokens following the FROM keyword. The counts of all components are updated for each query, yielding a comprehensive statistical overview of query usage patterns.

Phase II: Group Analysis:

In the data preprocessing phase, we divide the entire SQL query log into subsets (i.e., query groups) according to predefined rules. Such groupings can be divided based on time windows, query structure similarities, user behavior patterns, or other contextual information. Subsequently, the system processes each query group independently by parsing each SQL query statement, extracting the table names involved, and performing frequency statistics within the group. Eventually, each group accumulates the table names used in all queries in the group and their frequency of occurrence, forming group-level table access characteristics. This group-based aggregation process is critical for analyzing local behavioral patterns of database usage. By analyzing smaller, focused groups of queries, we can more easily identify sets of tables that are frequently accessed together. These patterns typically appear in shared business logic, repeated session behavior, or specific time windows. As a result, we can uncover localized access hotspots and usage structures that would be downplayed in a full log analysis. This not only helps to improve the efficiency of frequent-pattern mining, but also provides a more refined reference basis for subsequent index optimization, cache strategy design, and system resource allocation.

Phase III: Frequent-Pattern Mining:

After preparing the data and extracting table names, we apply the FP-Growth algorithm. The FP-Growth algorithm was selected because it can efficiently handle large datasets without the need to generate the large candidate sets required by other algorithms, such as Apriori [10]. By identifying frequent patterns in table name usage, FP-Growth reveals common querying behaviors and highlights opportunities for database optimization. FP-Growth is used to uncover frequent patterns in the set of table names that appear in each query group. These patterns represent combinations of structures that are repeatedly accessed by the database in a particular context, thus reflecting the consistent behavior of the user or the system during operation. For example, the frequent co-occurrence of certain table combinations may suggest the existence of implicit business logic associations or joint query requirements between them. By systematically identifying these high-frequency patterns, FP-Growth not only reveals the structural characteristics of the underlying query behavior, but also provides important clues for database optimization, such as index federation optimization, table join strategy tuning, or data partitioning recommendations.

This complexity analysis investigates the computational requirements of both the preprocessing phase and the GFP-Growth algorithm in the context of SQL query analysis. The preprocessing phase involves parsing each m SQL query and extracting tables, operations, and `WHERE` clauses using regular expressions. This process has linear time complexity, denoted by $O(m \times n)$, where n denotes the average length of the query. While grouping the query imposes only minimal overhead, it requires that each grouping be processed independently, thus maintaining the linear complexity of the overall preprocessing task. However, despite the relatively efficient construction of the FP-tree, its subsequent pattern mining process still faces computational bottlenecks in the worst case. Specifically, the complexity of frequent-pattern mining may reach $O(2^f)$, where f denotes the number of frequent terms. Several parameters critically influence the algorithm's runtime.

Specifically, smaller group sizes and higher minimum support thresholds (`min_sup`) reduce computational complexity by limiting the number of transactions and frequent itemsets. In contrast, larger group sizes and lower support thresholds increase processing demands. Thus, the overall efficiency of the approach hinges on striking an appropriate balance between preprocessing overhead and the potential exponential cost of the GFP-Growth algorithm. This underscores the necessity of selecting suitable grouping strategies and support thresholds to ensure scalable and efficient performance.

3.4 Experiment

The purpose of this experiment is to investigate the impact of dataset size and grouping strategy on the runtime and association rule accuracy of the GFP-Growth algorithm in SQL query analysis. We systematically evaluate the performance, including runtime and rule confidence, under different grouping sizes (5, 10, 20, 30, 50, 100) and support thresholds (0.002, 0.02, 0.2). In the experiments, the ungrouped FP-Growth method (which treats all queries as a transaction set) serves as the baseline, while each grouping configuration represents a different variant of our method, which is used to validate its effectiveness under a wide range of conditions.

3.4.1 Experimental Setup

Dataset Description

The dataset used in this experiment was provided by Lai et al. [40] and contains SQL queries extracted from a CSV file. These queries originate from the Sloan Digital Sky Survey (SDSS), which stores images, spectra, and catalog data for over three million objects [61]. For each SQL record, the original query statement and session category labels were obtained. The SQL queries were then parsed to identify table names and associated operations, which were subsequently used for frequent-itemset mining.

Preprocessing

Preprocessing played a crucial role in the entire GFP-Growth model construction process. To systematically assess the impact of the grouping strategy on the runtime performance of the GFP-Growth algorithm, we divided the original SQL query logs into multiple query subsets, i.e., 'query groups'. This process was not only part of the data preparation, but also a key aspect of the experimental design, aiming to explore the specific impact on the algorithm's efficiency by controlling the number and size of query groups. We adopted two grouping methods: the first was to take the entire query log as a whole, keeping the original structure of the database unchanged without any form of query partitioning; the second was the fixed group size method, which divided the query log into a number of groups by setting the group size artificially, and each group contained 5, 10, 20, 30, 50, or 100 SQL queries. Each grouping approach represented different data granularity and processing complexity, providing us with a multi-dimensional performance evaluation framework. In each query group, all SQL queries were first syntactically analyzed and structurally parsed to extract the table names involved. These table names were then converted into a standardized transaction format so that they could be used as input data for the FP-Growth algorithm. In this way, we mapped the SQL query semantics into structured data that could be used for frequent-itemset mining, thus providing the basis for subsequent GFP-Growth pattern mining. There were two main motivations for introducing these grouping strategies: on the one hand, we want to explore whether smaller group sizes help reduce the runtime of the algorithm, since each group contains fewer transactions, and the FP-tree construction and recursive search process is lighter; on the other hand, we also need to evaluate the potential advantages of larger group sizes, such as more complete frequent-pattern-capturing capabilities, but also the possibility of a higher computational overhead. The ultimate goal is to find an optimal balance between minimizing runtime resource consumption and maximizing the retention of useful information in the

data, thus improving the overall execution efficiency and applicability of the GFP-Growth algorithm.

3.4.2 Experimental Results

In order to perform an in-depth evaluation of the performance of the GFP-Growth algorithm under different parameter settings, we systematically analyzed its runtime performance under a variety of minimum support thresholds (minsup) and query group size conditions. The main goal of the experiments was to clarify how these key parameters interact with each other to affect the efficiency of the algorithm, as shown in Figure 3.2.

For minimum support, we selected three typical threshold levels: 0.2, 0.02, and 0.002. These values represent high-, medium-, and low-frequency pattern mining scenarios, respectively, and can effectively capture the impact of frequent itemsets on runtime efficiency at different granularities. Figure 3.3 shows that higher minimum support (e.g., 0.2) significantly reduces the runtime of the algorithm for all query group size settings. Note: "N/A" values indicate that no association rules satisfied both the minimum support (0.2) and confidence (0.8) thresholds for group sizes of 10 and 5. These small groups lacked sufficient frequent-table co-access patterns under strict thresholds. This is due to the fact that only a small number of frequent itemsets satisfy the support requirement under high-threshold conditions, which reduces the construction depth of the FP-tree as well as the scope of the recursive search, and significantly reduces the computational complexity.

Figure 3.4 shows that lower support thresholds (e.g., 0.002) significantly lengthen the runtime, especially for larger query group sizes (e.g., 50 or 100 queries per group). This is because with lower support, the number of frequent itemsets that the algorithm needs to mine and store increases dramatically, resulting in higher memory consumption and computational burden, especially in the context of large transaction sets.

In addition, we divided the entire dataset into 5, 10, 20, 30, 50, and 100 query subsets and tested the impact of each grouping size on the operational efficiency separately.

The experimental results show that compared to the baseline scenario (i.e., without any grouping processes), the use of small grouping sizes (especially 5 or 10 queries per group) can significantly reduce the runtime, especially when the support threshold is set higher. The main reason for the increased efficiency is the smaller number of transactions in each group, which allows the FP-Growth algorithm to handle smaller FP-trees, greatly reducing the computational overhead.

However, as the size of the query group grows (e.g., to as many as 50 or 100 queries per query group), the runtime starts to increase. This trend is especially noticeable at lower support thresholds. The reason is that, although the grouping strategy was originally intended to reduce computational complexity, the resource consumption during FP-tree construction and recursive mining rises as the size of the set of transactions within each group grows, thus offsetting the original performance advantage.

The runtime efficiency is significantly improved when transitioning from the baseline strategy to a small-sized grouping strategy; however, the runtime bounces back when the group size is too large and the support threshold is too low. This finding highlights the core issue that needs to be considered in the design of grouping policies, that is, how to reduce the runtime while avoiding the growth of computational overhead due to too large a transaction set size. Therefore, choosing the appropriate grouping size and minimum support threshold is key to optimizing the execution efficiency of the GFP-Growth algorithm.

3.5 Discussion

From the above runtime analysis, it is obvious that the grouping strategy of SQL queries and the setting of the minimum support threshold have a significant impact on the overall performance of the FP-Growth algorithm. Specifically, smaller grouping usually leads to a smaller number of transactions contained in each group, which reduces the computational burden of FP-tree construction and frequent-pattern mining. As a result, smaller group sizes tend to achieve faster runtimes and significantly improve processing

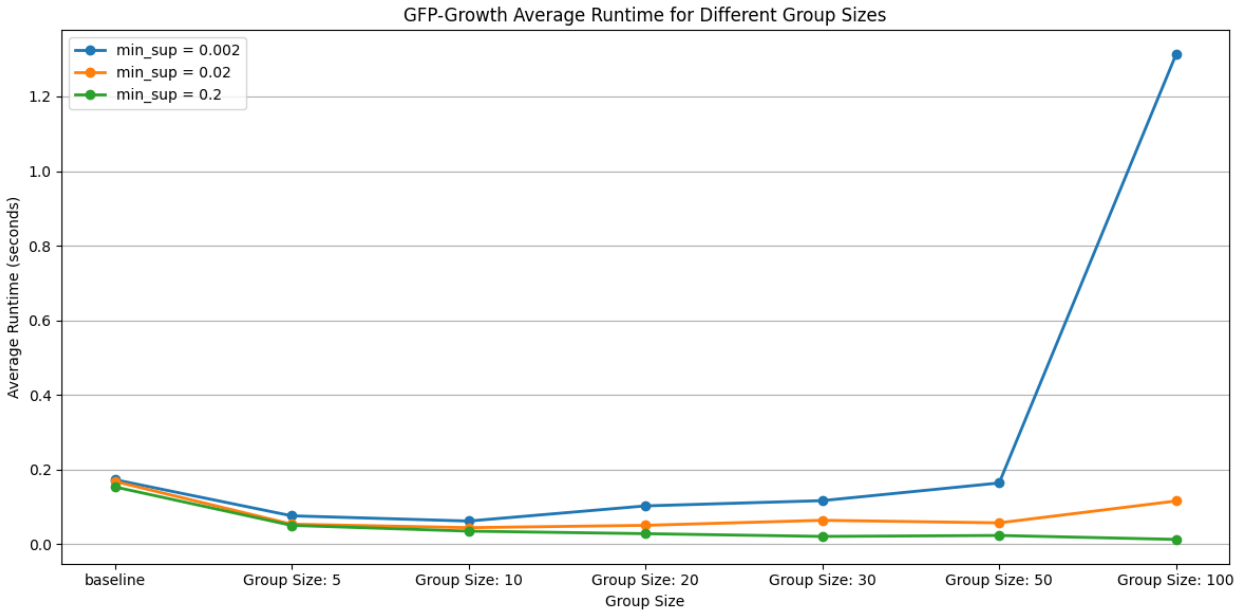

Figure 3.2: GFP-Growth performance under different group sizes.

efficiency at the same level of support. However, this performance improvement often comes at the expense of the quality of mining results. As fewer transactions are included in each subgroup, the set of frequent items that can be identified is correspondingly reduced, resulting in the final generated association rules being more localized and lacking in global representativeness. By comparing the average confidence metrics, it can be found that although small groupings help to speed up the computation, the rules they generate may have a decreasing trend at the confidence level, which makes it difficult to provide sufficient explanations.

To illustrate the practical utility of the discovered patterns, consider the following representative rule identified under a support threshold of 0.02 and a group size of 20: We have an association rule: {A, B} $\Rightarrow$ {C} [support = 0.035, confidence = 0.81]. This rule suggests that when a query accesses tables A and B, it is also highly likely to access table C. Such patterns enable database administrators to optimize performance by preloading these frequently co-accessed tables into memory as a group. By proactively bringing A, B, and C into memory during query processing, the system can significantly

reduce I/O latency, improve cache hit rates, and enhance overall responsiveness in data-intensive environments.

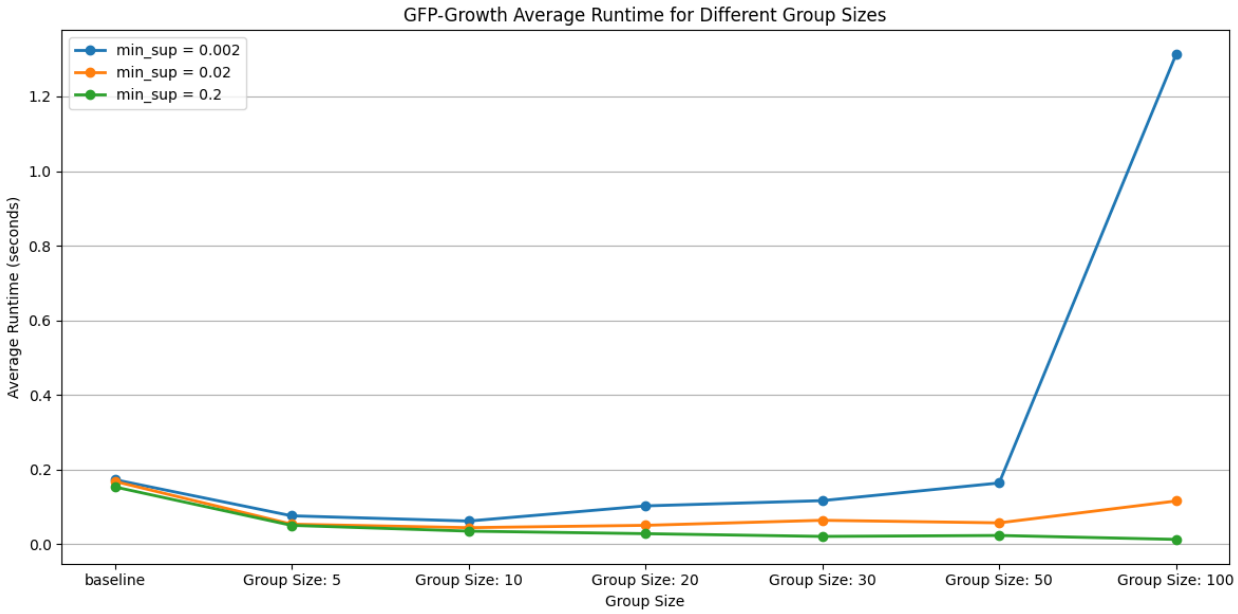

Figure 3.3: Support value = 0.2.

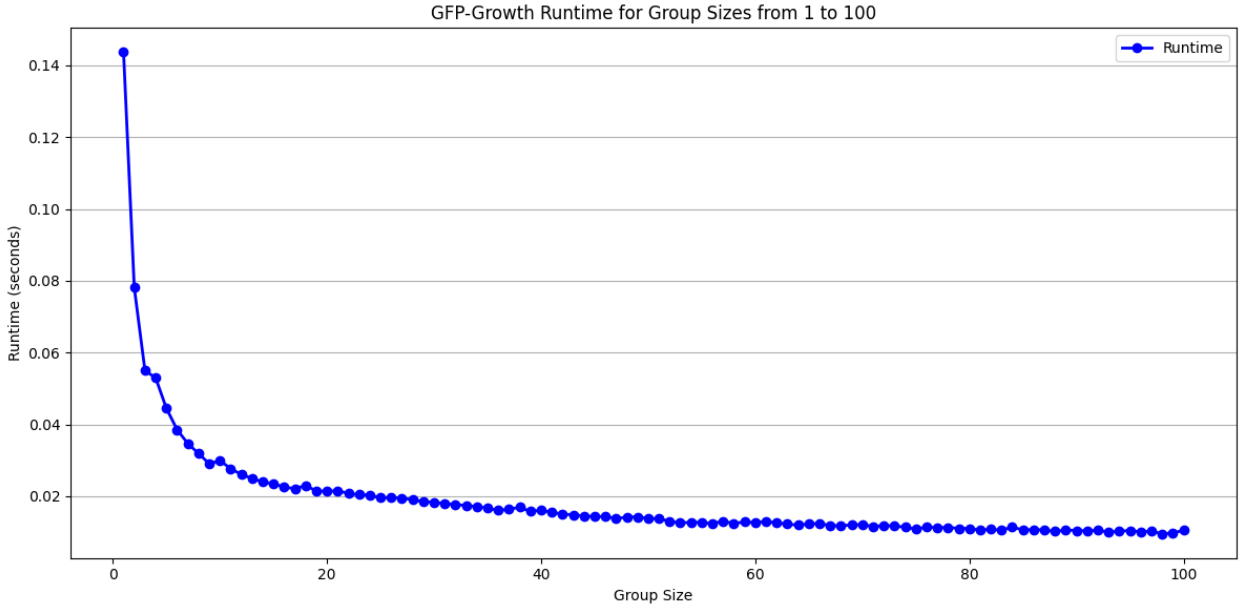

Figure 3.4: Support value = 0.02.

Selecting an appropriate group size is crucial for maintaining the computational efficiency and stability of the GFP-Growth algorithm. Larger groups significantly expand the search space, leading to increased runtime and volatility in the results, whereas smaller groups generally reduce computational overhead but may limit the algorithm's ability to capture all relevant patterns. In addition to group size, the choice of support and confidence thresholds also significantly affects both the quality and the quantity of rules generated. In Tables 3.1 and 3.2, it is indicated that a higher support threshold produces a smaller but more representative set of frequent itemsets, facilitating faster calculations and more reliable patterns. Conversely, Figure 3.5 displays that lower support thresholds can uncover additional frequent itemsets, but risk introducing noise or irrelevant patterns. Combining a high support threshold with a high confidence level yields fewer rules that occur frequently and reliably, reducing the computational burden. However, high support with low confidence may indicate weak correlations between items. Meanwhile, low support with high confidence often requires more extensive computation because the algorithm must process larger itemsets. Finally, low support combined with low confidence produces a voluminous set of rules, many of which may lack practical value. Carefully balancing group size, support, and confidence thresholds is therefore essential to optimize runtime while maintaining the interpretability and practical utility of the discovered patterns.

Table 3.1: The association rules generated by GFP-Growth with support value = 0.02.

Group Size	**Filtered Rules (Support** ≥ 0.02 **and Confidence** ≥ 0.8**)**	**Avg. Support**	**Avg. Confidence**
Size 100	405,885	0.056	0.090
Size 50	116,564	0.084	0.890
Size 30	29,068	0.109	0.885
Size 20	7409	0.169	0.879
Size 10	16	0.048	0.912
Size 5	5	0.059	0.948

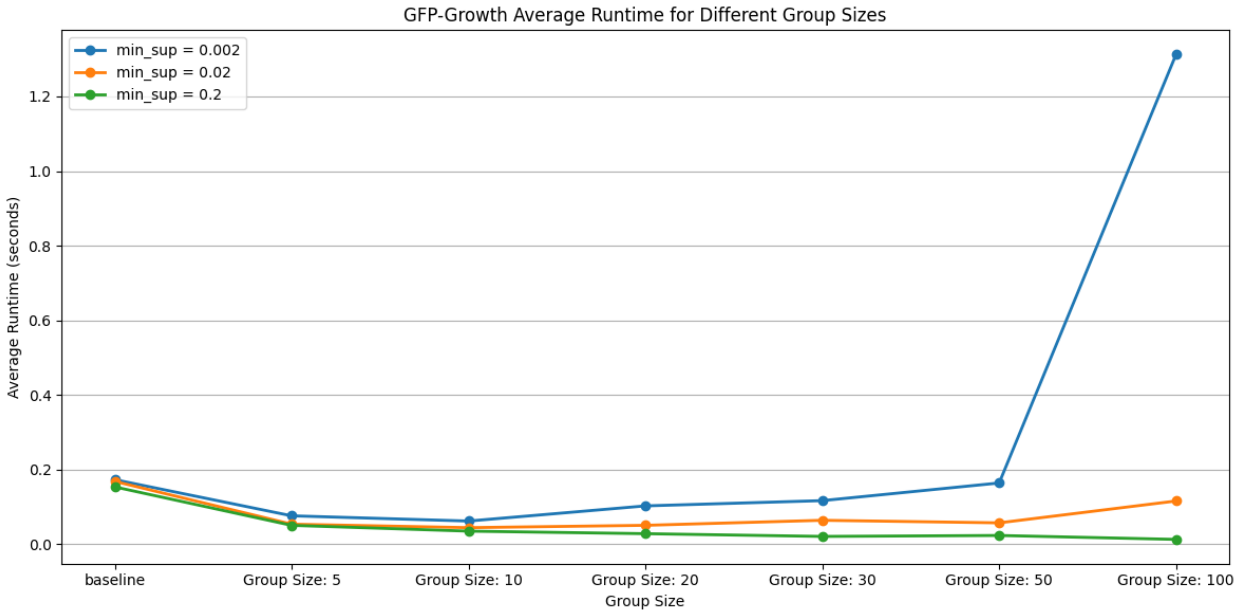

Figure 3.5: Support value = 0.002.

Table 3.2: The association rules generated by GFP-Growth with support value = 0.2.

Group Size	**Filtered Rules (Support** ≥ 0.2 **and Confidence** ≥ 0.8**)**	**Avg. Support**	**Avg. Confidence**
Size 100	17,901	0.439	0.923
Size 50	13,817	0.374	0.903
Size 30	7666	0.307	0.911
Size 20	2958	0.279	0.885
Size 10	N/A	N/A	N/A
Size 5	N/A	N/A	N/A

3.6 Summary

As the volume of SQL queries continues to grow, database systems increasingly face performance and energy consumption challenges arising from high table access frequency and data sparsity. In this paper, we present a grouping-based association rule mining approach that rapidly identifies the tables and fields with high access frequency, enabling targeted optimizations to reduce energy consumption and enhance system efficiency. Central to our method is a grouping-based analysis strategy that aggregates multiple related SQL statements into a cohesive set, thus mitigating the analytical difficulties posed by data sparsity. By uncovering inter-table relationships and access patterns frequently overlooked in conventional SQL log analyses, this research provides innovative methods and tools for optimizing SQL database performance and managing resources more effectively. While we did not directly measure hardware-level energy consumption, we believe that runtime reduction is a reasonable proxy for computational energy efficiency. In modern database systems, CPU utilization and energy consumption are highly correlated with execution time, especially for analytical workloads. By reducing the number of frequent itemsets generated per grouping and localizing the mining process, our grouping strategy significantly reduces execution time, which, in turn, means lower energy consumption.

This chapter shows that SQL workloads become more useful for optimization when individual queries are grouped and mined for recurring access patterns. This finding establishes the first step of the dissertation: isolated events provide limited insight, while structured behavior reveals patterns that can support better system decisions. The next chapter extends this idea to another data-driven system by examining how dynamic behavior modeling improves recommendation diffusion and system efficiency.

Chapter 4

Modeling Recommender Systems using Disease Spread Techniques

Chapter 3 demonstrated that grouping SQL queries can reveal hidden workload structure. Building on this idea, this chapter examines dynamic behavior modeling in recommender systems. Although this chapter does not directly optimize SQL database energy consumption, it supports the dissertation's broader energy-aware perspective: predicting future behavior can reduce redundant operations and avoidable resource consumption.

This chapter serves as a complementary methodological study on dynamic behavior modeling and system efficiency. Although the main focus of this dissertation is energy-aware SQL database optimization, the broader research question is how data-driven systems can reduce unnecessary resource consumption by understanding and predicting future behavior. Recommender systems provide a useful setting for examining this issue because recommendation is not only a matching problem, but also a diffusion process in which content spreads across users over time.

The proposed SI-based diffusion model helps reveal how different recommendation strategies propagate content through user networks. By identifying more efficient propagation patterns, recommender systems may reduce redundant recommendation attempts, repeated ranking operations, unnecessary user-item matching, and avoidable network interactions. In this sense, the model has potential energy-aware implications because it supports more efficient recommendation diffusion with fewer wasted computational and communication resources.

This insight also connects to the SQL database problem addressed later in this dissertation. In recommender systems, inefficient propagation wastes computation and communication resources; in SQL database clusters, inefficient table placement causes unnecessary cold-node wakeups. Both problems require modeling future behavior before resources are consumed. Therefore, this chapter reinforces the dissertation's central argument that energy-aware optimization should combine power-state management with behavior-aware modeling.

The explosion of online information and digital content today has greatly exacerbated the information overload problem faced by users. This trend highlights the importance of recommender systems in filtering and pushing relevant content, which becomes a key tool for guiding users' attention, increasing interaction frequency, and prolonging platform dwell time. Nowadays, recommender systems optimize the user experience through personalized content recommendations [45]. However, despite the remarkable achievements in the industry, we still lack a deep and systematic theoretical understanding of how recommended content diffuses in the user community and how users interact with each other to facilitate clicking or consuming behaviors. Previous studies have focused on improving recommendation accuracy [24], optimizing sorting strategies [67], or introducing contextual information to enhance user satisfaction, but these approaches are often based on static evaluation metrics, ignoring the temporal characteristics of recommended content spreading in dynamic user networks and the changing process of user engagement. In fact, a recommendation algorithm is not only a content matching process [49], but also a complex information diffusion process with obvious propagation and evolution [70]. Therefore, it is of great theoretical and practical significance to understand how recommendation strategies diffuse content across user networks from a macro perspective, as this sheds light on the collective dynamics of user engagement, platform influence, and viral propagation.

The motivation for this study stems from bridging the gap between recommender systems and information diffusion modeling by proposing an interdisciplinary analytical framework. We consider the recommendation process as analogous to the spread of an infectious disease in a social network [63]. This perspective not only captures the dynamic process of user reception and dissemination of content, but also quantifies the performance of different recommendation algorithms in terms of propagation speed, reach, and user coverage. Specifically, we explore the path of integrating three classical recommendation strategies: popularity-based [14], collaborative filtering [26], and content-based recommendation [21] with an epidemic network model [66]. With the help of real user interaction data, we construct a scale-free network reflecting the real social structure [6], and introduce a time-varying propagation rate function to simulate the phenomenon of user engagement decay over time, which makes the model closer to the user behavioral patterns in online platforms. This design can more realistically reflect the evolutionary trend of the recommendation effect under a long running time, and also provides an interpretable theoretical basis for the study of user churn and cold start problems [73]. Based on this methodological framework, we not only systematically compare the differences in content dissemination efficiency between different recommendation strategies, but also explore the interaction mechanism between algorithm design and user network structure. The experimental results reveal the rapid diffusion ability of collaborative filtering in the initial propagation, but also point out its weak propagation sustainability; in contrast, the popularity-based approach maintains high propagation stability over a long period of time despite its limited initial influence. These findings provide empirical references for the dynamic scheduling of recommendation algorithms and the design of hybrid strategies.

The main contributions of this chapter are as follows. First, this chapter introduces the SI model in epidemiology into the study of recommender systems, and constructs an analytical framework that can dynamically simulate the process of content diffusion, so as to be closer to the real-life changes in user behavior. Second, in a scale-free network

environment constructed based on real clickstream data, the system compares the performance of three types of classical recommendation strategies: popularity-based, collaborative filtering, and content-based approaches in terms of propagation efficiency. Third, this chapter introduces time-varying propagation rate to characterize the phenomenon of user engagement decay over time, providing a more nuanced perspective for understanding the evolution of recommendation effectiveness. Finally, this interdisciplinary study not only enriches the theoretical connection between recommender systems and information diffusion mechanisms, but also provides insights with practical value for optimizing recommendation algorithms to maintain user engagement.

The remainder of this chapter is organized as follows. In Section 4.1, we clarify our research motivation and present research questions. Section 4.2 defines the problem of modeling recommendation diffusion and outlines key assumptions. Section 4.3 introduces our proposed approach, which integrates classical recommendation algorithms with a network-based SI model incorporating time-varying infection rates. In Section 4.4, we verify the model's effectiveness and feasibility by testing its performance under various scenarios. Section 4.5 discusses the limitations of the current model and suggests that more sophisticated propagation mechanisms could be introduced in the future to enhance the realistic applicability of the model. Finally, Section 4.6 summarizes the key contributions of the article and outlines potential avenues for future research.

4.1 Research statement

Recommendation algorithms determine content propagation in digital platforms and profoundly affect user behavior. Although most of the existing studies focus on algorithm performance and efficiency, there is still a lack of systematic modeling and empirical analysis on how content dynamically spreads among users and the influence of network structure.

To fill this gap, this study introduces an epidemiological modeling perspective and uses the SI (Susceptible–Infected) model to simulate the propagation of recommendation signals. Figure 4.1 through Figure 4.4 provide a simple demonstration of SI model diffusion. This model provides us with a tool to portray the time-evolving behavior of recommendation content as an “infected” propagation unit, which gradually spreads in the user network. This approach not only breaks through the traditional evaluation of recommendation effects based on static indicators (e.g., accuracy and click-through rate), but also gives us a systemic perspective to re-understand the behavioral characteristics of recommender systems in terms of propagation breadth, speed, and sustainability.

In this work, we have two goals. First, we are interested in analyzing whether the Susceptible-Infected (SI) model can be used to effectively capture the realism in recommendation behavior, for example, if the diffusion can be used to mimic recommendation algorithms' propagation on different levels, for example, propagation speed, final coverage and the time decay of user activities. The second goal is to build a realistic simulator that is based on a real dataset from a music streaming platform. By using the simulator, we can easily sample scale-free networks, attribute different properties to users, such as language and city, and then compare different recommendation strategies on the generated datasets to model the diffusion effects under different settings but in a more realistic data-driven environment. In summary, we believe that the two goals we set can also provide a general basis for interdisciplinary efforts in both advancing the theory of recommender systems and understanding the potential of developing more engagement adaptive recommendation algorithms.

To systematically study the dynamic dissemination of recommended content, we propose a series of research questions as a guide, focusing on evaluating the applicability of epidemiological models in recommender systems and designing close-to-real network structures to simulate the process of user interaction and information dissemination.

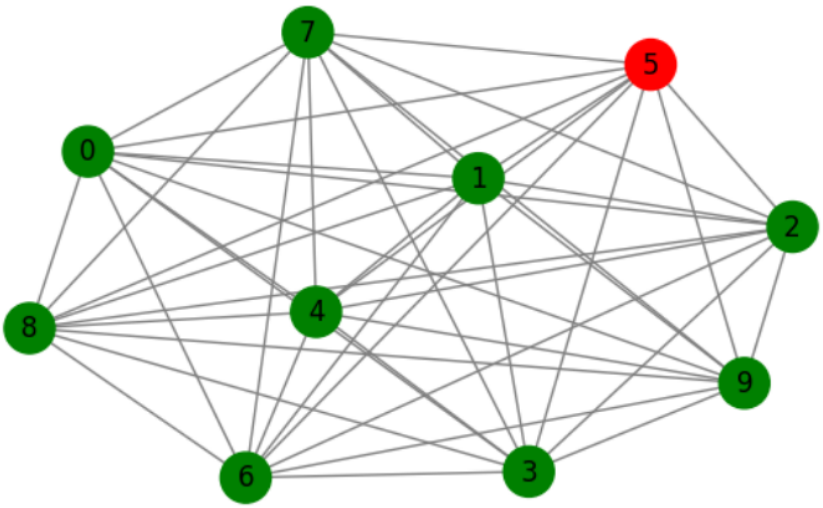

Figure 4.1: An overview of the simulation across the first phases of SI model diffusion. Green nodes represent susceptible individuals, red nodes represent infected individuals.

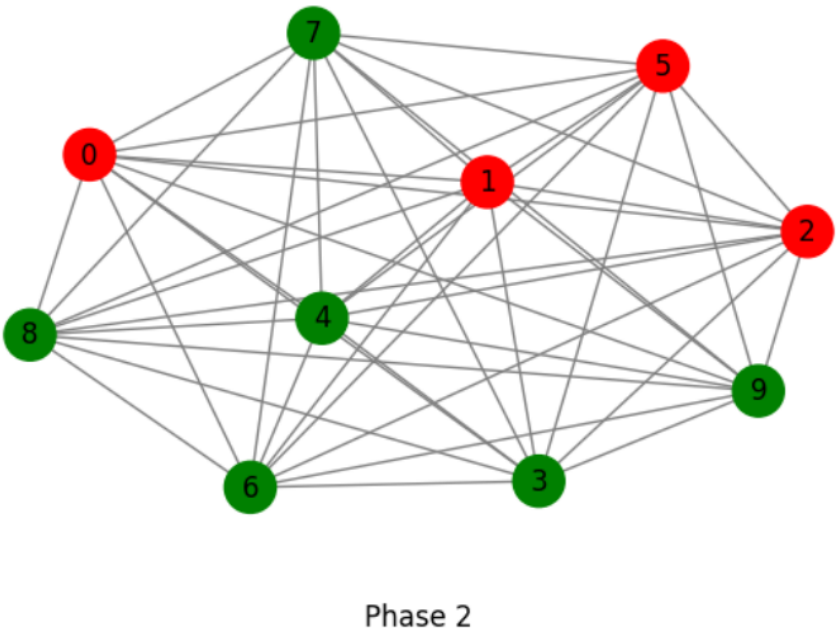

Figure 4.2: An overview of the simulation across the second phases of SI model diffusion. Green nodes represent susceptible individuals, red nodes represent infected individuals.

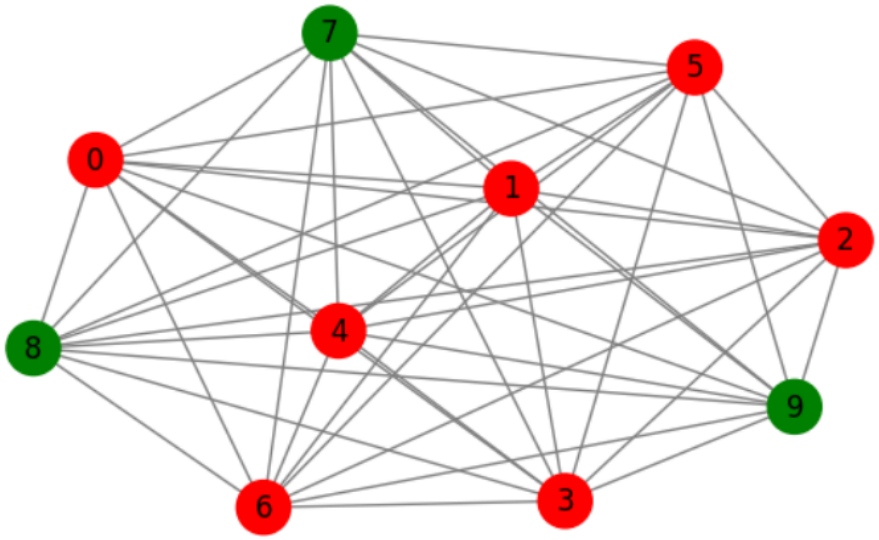

Figure 4.3: An overview of the simulation across the third phases of SI model diffusion. Green nodes represent susceptible individuals, red nodes represent infected individuals.

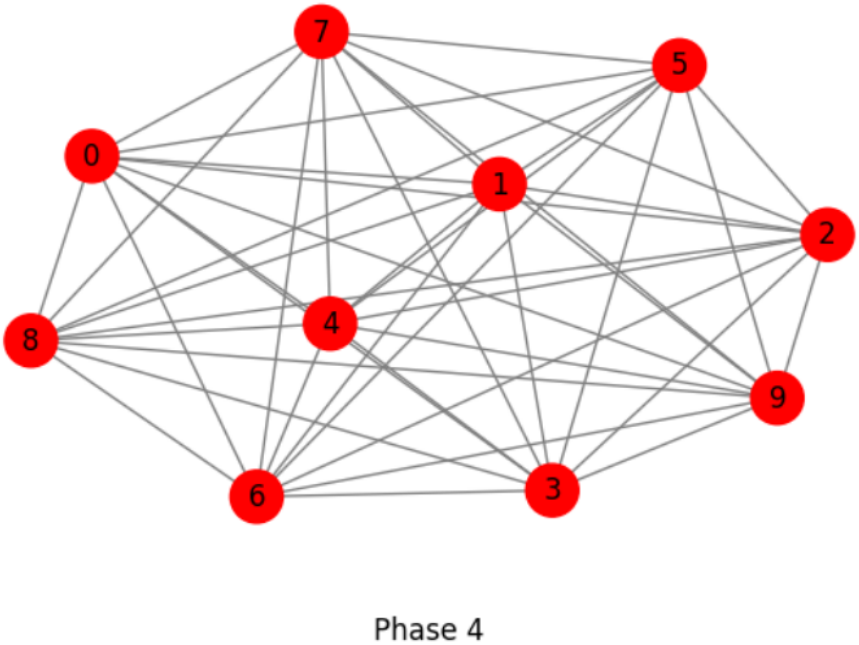

Figure 4.4: An overview of the simulation across the forth phases of SI model diffusion. Green nodes represent susceptible individuals, red nodes represent infected individuals.

Research Questions

- Can the SI epidemiological model effectively represent the dynamics of digital recommendation algorithms?
- How can user interactions be used to construct networks that realistically represent the structure of digital recommendation systems?

To answer the above questions, we first construct a scale-free user network based on real clickstream data to reflect the typical heterogeneous connection characteristics in digital platforms. Then, we design a simulation environment to embed different types of recommendation strategies into the SI model and observe their propagation trajectories evolving over time. Finally, we introduce a dynamic propagation rate mechanism to simulate the real-world scenario of declining user engagement, so as to enhance the ecological fidelity and predictive capability of the model.

4.2 Problem Definition

Recommendation systems face a crucial challenge when it comes to predicting user preferences accurately and spreading those preferences effectively throughout networks. Traditional recommendation systems generally produce fixed recommendations which do not account for dynamism in user preferences. The dynamic nature of user behaviors allows preferences and adoption decisions to spread through networks in the same way infections move in disease models. Recommendation systems that overlook the process of dynamic propagation through networks will miss opportunities to leverage network effects, which restricts their real-world performance.

Formally, we define our research problem as follows. Given n users $U = \{u_1, \ldots, u_n\}$, associated preference labels $L = \{l_1, \ldots, l_n\}$, and group attributes such as language, city, and genre, our goal is to:

1. Develop a robust recommendation algorithm f integrating collaborative filtering with dynamic propagation mechanisms, defined as:

$$\hat{L} = f(U, A, S) \tag{4.1}$$

where $\hat{L}$ denotes predicted preferences, A is the set of group attributes, and S represents the user similarity matrix derived from collaborative filtering.

2. Model the recommendation propagation using an SI (Susceptible-Infected) epidemic framework. In this model, each user $u_i \in U$ transitions from susceptible (S) to infected (I) state based on the probability:

$$P(u_i \text{ infected at } t+1) = 1 - \prod_{u_j \in I_t} (1 - \beta_{ij}(t)) \tag{4.2}$$

where $\beta_{ij}(t)$ is the dynamically decreasing transmission rate between infected user u_j and susceptible user u_i at time t.

3. Investigate the impact of initial recommendation strength (collaborative filtering scores $CF(u_i)$) on the propagation efficiency over time, given by:

$$\beta_{ij}(0) = \alpha \times CF(u_i) \tag{4.3}$$

where α is a scaling factor controlling initial transmission strength.

Specifically, we aim to answer the following key questions:

- What methods can we use to merge collaborative filtering recommendations with an SI epidemic propagation model for precise integration?
- What impact does a transmission rate that decreases over time have on how recommendations are adopted by users during different periods?

- How does the network structure and the attributes of user groups influence the success of recommendation propagation?

The above questions have been addressed from different perspectives, and in this work we seek to further contribute to this end. To this aim, we propose a combined approach of collaborative filtering and an epidemic model, in particular an SI model, to describe, from a macroscopic perspective, the diffusion process of content recommendation in the network of users. This approach allows at the same time to account for the microscopic matching of user–item preferences, while at the same time modeling the macroscopic propagation processes. Modeling the impact of different parameters, the network topology and characteristics of the node groups allows us to gain an understanding of their impact on the diffusion process, thus supporting the development of more flexible and network-aware recommendation mechanisms.

4.3 Approach

Our approach bridges recommendation system algorithms with epidemic diffusion models to simulate and evaluate content propagation within a user network, and Figure 4.5 is the overview of the whole simulation. The methodology consists of three main components: network construction, recommendation strategy implementation, and dynamic SI-based simulation.

4.3.1 Network Construction

We use the Barabási–Albert (BA) model to generate a scale-free user network to mimic social systems at the structural level [2]. The model takes as input a number of users, $N = 27,244$, as each user is represented by a node in the network. As each node is added to the network, it forms links with $m = 3$ other nodes, which are selected based on their degree. In other words, we simulate user-level propagation of posts on a network

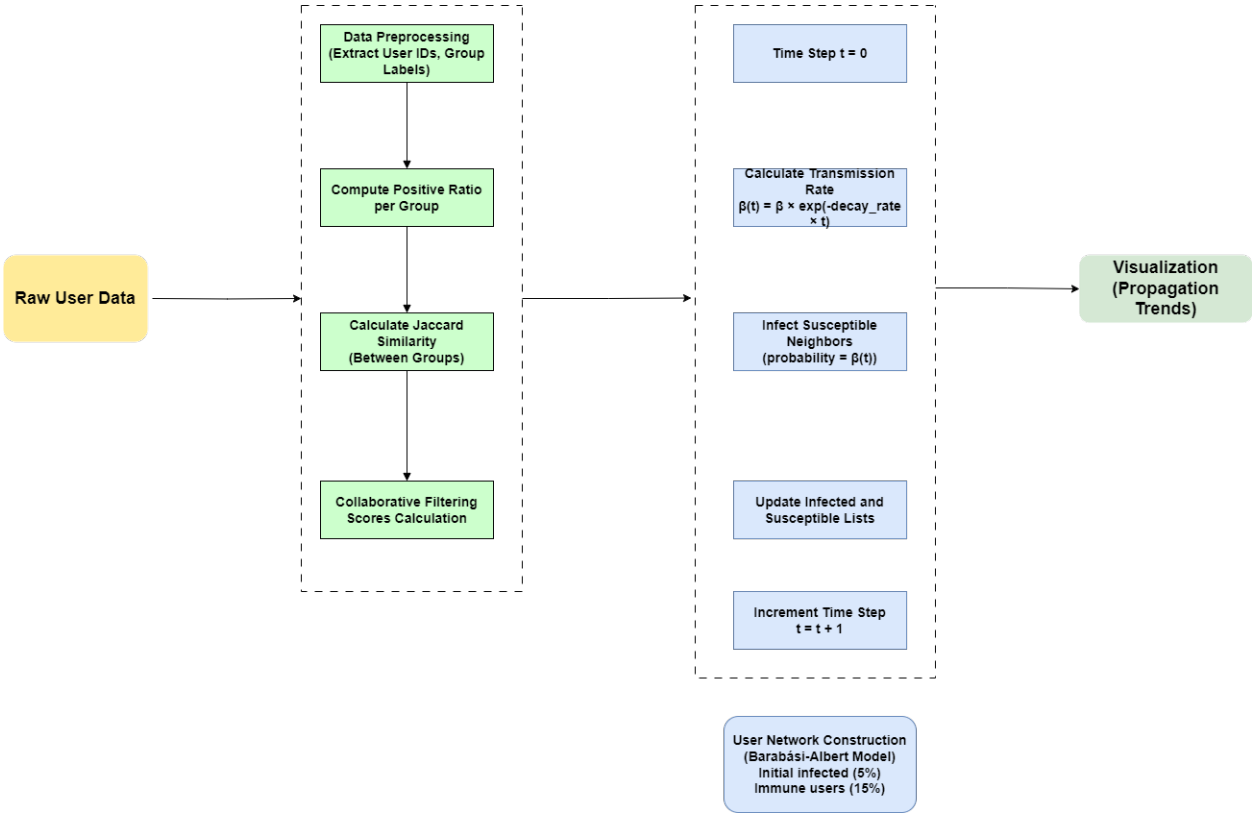

Figure 4.5: An overview of whole simulation.

in which newly added users tend to link to highly connected existing users. As with all the random models, we map real user IDs to the generated network. The generated network has several characteristics of social systems, such as hubs and a power-law degree distribution. We verify this by using different graph metrics. The mean degree of the network is $\approx$6.00, which shows that the network is sparse. The network density is $\approx$0.00022, which shows that the network is not fully connected, and the clustering coefficient is $\approx$0.0024, which shows that friends of friends are unlikely to be friends.

4.3.2 Recommendation Algorithms

This research compares three traditional recommendation algorithms: popularity-based, collaborative-filtering, and content-based recommendation. Under the popularity-based method, users receive the most popular items according to historical click data, representing a general-interest effect. Collaborative filtering methods function through user–item interaction matrices by evaluating item similarities to pinpoint relevant items for users based on their past behaviors and recommend content that other similar users have liked, thereby delivering personalized recommendations [39]. Recommendation systems based on content analysis utilize metadata attributes of items such as genre or creator to generate suggestions. These methods generate personalized recommendation results by matching feature vectors that represent item characteristics with users' past preferences and combining those with users' interests [57]. During the evaluation phase we define each algorithm's recommendation approach at the start of the simulation while identifying initial target users and then simulate how the recommended content spreads through the user network to examine both the diffusion path and coverage among users as well as dynamic performance changes over time.

4.3.3 Dynamic SI Simulation

Our simulation of content propagation relies on an adjusted Susceptible-Infected (SI) model. At the beginning of the simulation, a selected group of users becomes "infected" after encountering recommended content. During each time step, infected users attempt to infect their neighbors with a probability determined by the initial transmission rate and a time-dependent decay factor representing declining engagement. Users who do not interact regardless of exposure to content are not considered in propagation models.

The quantities $S(t)$ and $I(t)$ denote the number of susceptible and infected users at time t in this system. The overall population size remains unchanged as $N = S(t) + I(t)$. The classical SI model explains how infections spread through populations.

$$\frac{dI(t)}{dt} = \beta(t) \cdot \frac{S(t) \cdot I(t)}{N} \tag{4.4}$$

To simulate declining user interest over time, we define the transmission rate $\beta(t)$ as an exponentially decaying function:

$$\beta(t) = \beta_0 \cdot e^{-\lambda t} \tag{4.5}$$

where β_0 is the initial transmission rate, and λ is the decay rate parameter controlling the rate at which recommendation effectiveness decreases. This dynamic SI formulation enables us to account for temporal variations in user receptiveness to recommendations, making the diffusion model more realistic for online environments.

While Equation (4.4) is derived from the classical SI model assuming a fully mixed population, in our implementation, the infection process is constrained by the network structure. Specifically, an infected user can only transmit content to their immediate neighbors, reflecting localized person-to-person interaction. This adapts the global formulation to a network-based setting consistent with realistic user interactions.

4.3.4 Item Representation in Epidemic Framework

Our approach for embedding recommendation items into epidemic modeling involves visualizing each item as a standalone infectious entity that spreads through the user network. The process of suggesting an item to a user functions similarly to exposing that user to a possible infection. A user becomes infected when they engage with (e.g., click) the recommended item.

As shown in Figure 4.6, we define u_i as a user node and v_j as an item within this formal framework. The exposure function $R(u_i, v_j)$ specifies whether v_j gets recommended to user u_i. The interaction probability of u_i with v_j after a recommendation is represented by β_{ij}, which serves as a parallel to the transmission rate in the SI model. The item v_j acts as the propagation medium, while the diffusion of v_j through the network is measured by monitoring the total number of users who become engaged over time.

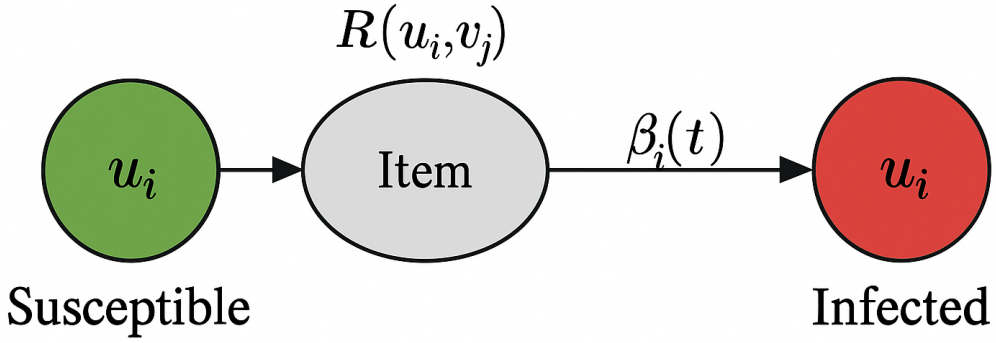

Figure 4.6: Item representation.

In this formulation, multiple items can propagate simultaneously, each governed by their own temporal transmission dynamics. Moreover, to reflect the natural decay in user interest, we define the effective transmission rate of item v_j at time t as:

$$\beta_{ij}(t) = \beta_{ij}(0) \cdot e^{-\lambda_j t} \tag{4.6}$$

where λ_j is the decay constant specific to item v_j, modeling the decline in novelty or visibility over time.

4.4 Experiments

This chapter designs and implements a series of simulation experiments to test the proposed recommendation propagation framework based on epidemic modeling in real-world environments. Real-world user click data serves as the foundation for our experiments while we systematically assess propagation efficiency and dynamic evolution of three classical recommendation algorithms using a scale-free user network that reflects real-world network structure characteristics. The next section details the dataset and network construction method while explaining the recommendation strategy settings and propagation simulation process used in experiments before presenting and analyzing key experimental results.

4.4.1 Dataset

The dataset used in this study is sourced from KKBox, a major Asia-based music streaming platform offering over 30 million tracks. Specifically, we utilize the publicly available KKBox dataset released for the Kaggle Music Recommendation Challenge. The task is to predict whether a user will repeatedly listen to a song within one month after the first recorded listening event. If repeated listening occurs within this window, the label is 1; otherwise, it is 0. The dataset includes user–song pairs with their first observable interaction, along with associated metadata.

4.4.2 Data-Driven Simulation Perspective

In this chapter, based on the real-world click-through behavior data, we design and implement a simulation framework of recommendation propagation from the perspective of epidemiology to empirically measure the diffusion efficiency of various recommendation strategies in the user network. To begin with, we generated a scale-free network based on the Barabási–Albert (BA) model with a total of N nodes. The model incrementally adds new

nodes, each connecting to existing nodes with a probability proportional to their degree, thereby reflecting the heterogeneity in connectivity and the emergence of highly connected "hub" nodes—features commonly observed in real-world social networks. Each node in the network corresponds to a user and is annotated with the anonymized user ID. Each user–content interaction (e.g., a song) is expressed as a binary click to reflect whether the content is effectively received or adopted. To simulate the propagation process of recommended content among users, we then choose three mainstream recommendation algorithms as the benchmark for the experiments, including popularity-based recommendation, collaborative filtering recommendation, and content-based recommendation. Each algorithm produces an initial recommendation list for each user, which corresponds to the "exposure set" in the simulation. Since each recommended item can be seen as an "infectious agent" in the epidemiological analogy, the users who are recommended the item form the initial set of "infected", so the propagation process of the recommender system can naturally be mapped to the modeling logic of infectious disease propagation.

The recommendation propagation process is modeled using a dynamic Susceptible-Infected (SI) framework. In this model, each infected user, i.e., a user who has interacted with the recommended item, has a chance to transmit the item to their neighboring users at each discrete time step. To account for the natural decay of user interest over time, we define a time-dependent transmission probability function as Formula (4.5). The simulation iteratively updates the infection status of users at each time step until the propagation process either converges or reaches a predefined temporal limit.

To provide a detailed comparison on the propagation properties of the various recommendation strategies, we design experiments in several dimensions. We first look at the propagation speed (i.e., the number of infected users growing in unit time) and the terminal coverage (i.e., the overall number of infected users) of items in the network and measure the propagation efficiency of the three recommendation algorithms by the difference. The simulation network is constructed using real user IDs from the KKBox dataset,

where each node corresponds to a user and carries metadata such as language and cities. This metadata is used to initialize node attributes, enabling the simulation to reflect real-world audience segments. In addition, we introduce multiple time dimension settings (e.g., time of day, time of week, time of year) for the multi-scenario testing in order to simulate the various information dissemination paths on real platforms, and explore the time sensitivity of content propagation. We also use the item category information (e.g., language) provided in the dataset to classify items, simulate the propagation trajectories of different categories of items and their dynamic evolutionary process, and test the adaptability of the algorithms in different contexts and preference structures.

4.4.3 Daily Diffusion Simulation

The first experiment simulates the process of content propagation over a 5-day period, setting the propagation decay rate to $\lambda = 0.1$. As shown in Figure 4.7, most content types spread rapidly in the early stage of propagation (the first 5 time units), and then the curves flatten out, suggesting that user engagement experiences an early peak followed by rapid saturation. The percentage of infected users eventually stabilizes at around 0.58, meaning that about 58% of users clicked or interacted after the initial exposure. Among the different content types, the Language 59 and Language 3 categories spread slightly faster in the early stages, but also saturated earlier. This phenomenon suggests that under Language 59, the novelty of recommended content fades quickly in a short period of time, and the spreading range is limited by the rapid decay of user interest. This result is consistent with the shorter life cycle of user attention in real digital media.

4.4.4 Weekly Diffusion Simulation

The second experiment simulates the content propagation process over a 12-week period, again setting the decay rate $\lambda = 0.1$, but with the time granularity adjusted to weekly. As shown in Figure 4.8, compared to the daily simulation, the overall speed of

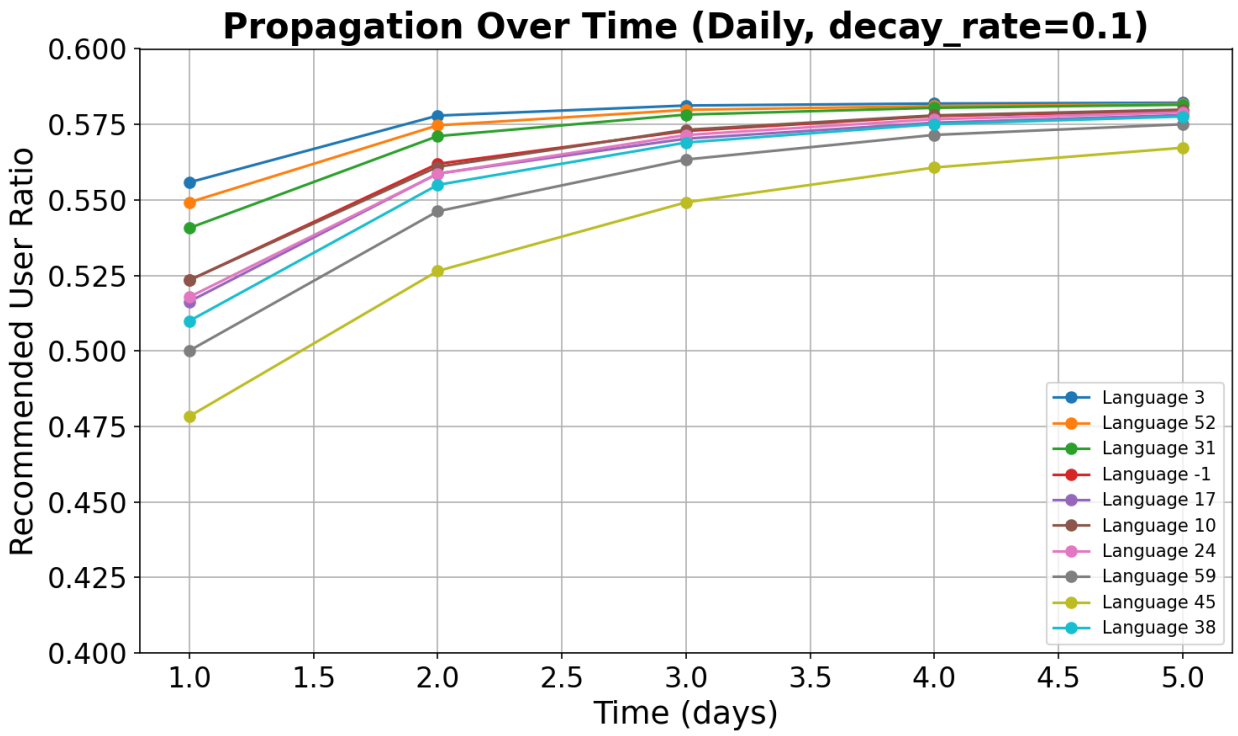

Figure 4.7: Model with dynamically decreasing transmission rate (daily, $\lambda = 0.1$).

this propagation process is slower, the curve grows smoother, and the final percentage of infected users is slightly higher, stabilizing at about 0.585. This suggests that a larger temporal granularity can help to smooth out fluctuations in the early user behaviors, making the propagation process more persistent and more continuous. It is worth noting that the difference in propagation curves between different content types is more pronounced here; in particular, Language 45 and Language 17 show some divergence in growth rates. This phenomenon suggests that time granularity has a significant impact on the assessment of content dissemination efficiency, and a longer time window helps to reveal the subtle differences in dissemination performance across content types.

4.4.5 Yearly Diffusion Simulation

A third experiment extended the propagation time to 5 years and increased the propagation decay rate to $\lambda = 0.2$, reflecting longer-term decline in user engagement. As shown

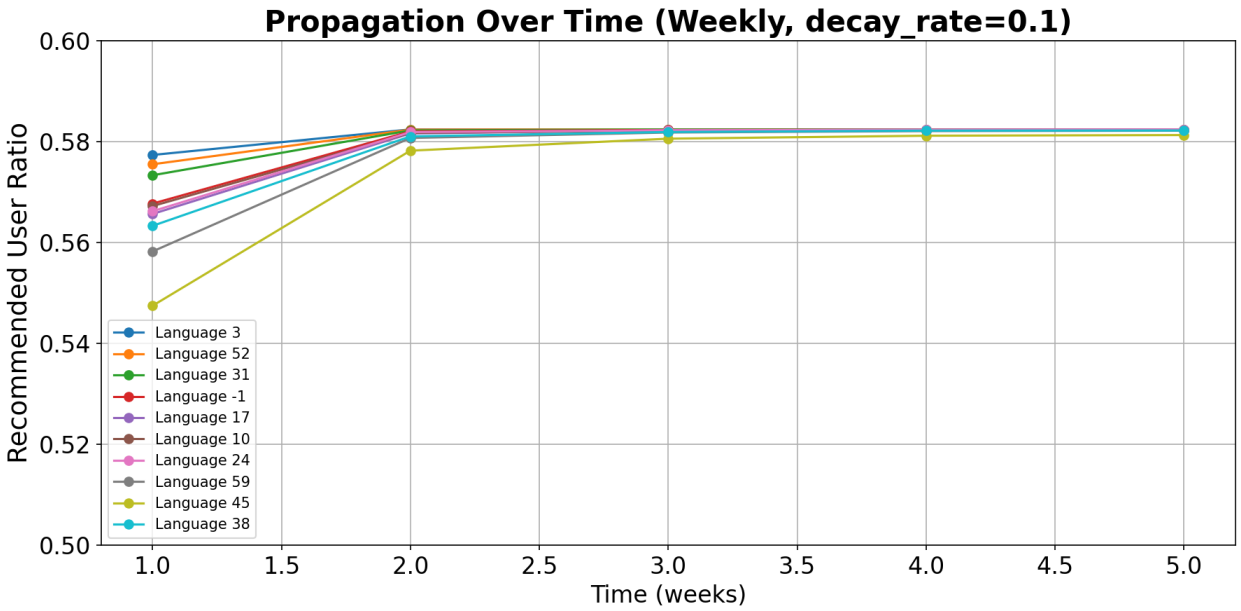

Figure 4.8: Model with dynamically decreasing transmission rate (weekly, $\lambda = 0.1$).

in Figure 4.9, although the proportion of infected users continued to grow slowly throughout the process, the overall spreading range remained limited, eventually converging between 0.56 and 0.58. Compared to the previous two experiments, the rate of spread is significantly slower and the saturation time is longer. Higher decay coefficients lead to a rapid weakening of user influence over time, simulating the phenomenon of content gradually losing its appeal due to decreasing novelty in a long-term recommendation environment. In addition, the differences in the curves between different content types are significantly reduced, suggesting that the variability in spreading ability is further compressed under the effect of strong attenuation. This experiment emphasizes the critical roles of time scale and decay mechanism in recommender system evaluation, suggesting that in the absence of content reactivation mechanism or continuous push, recommendation propagation will be difficult to sustainably expand its influence in the long-term scope.

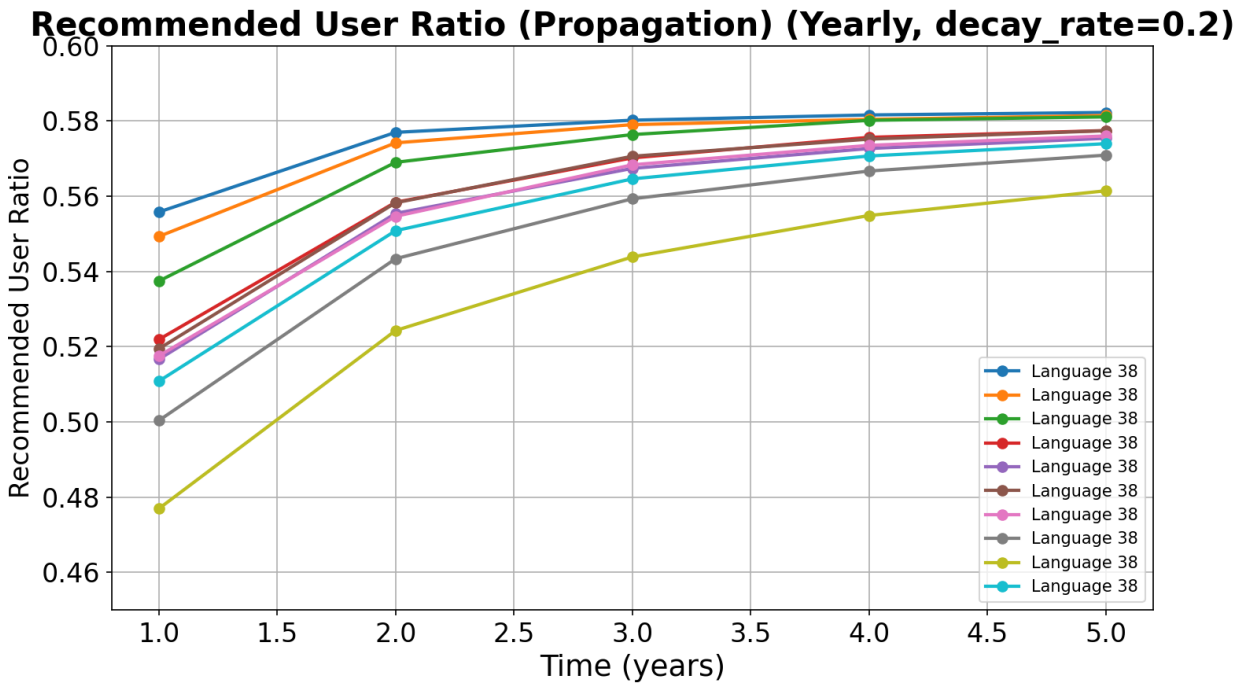

Figure 4.9: Model with dynamically decreasing transmission rate (yearly, $\lambda = 0.2$).

4.4.6 Comparative Analysis of Propagation Performance Across Recommendation Algorithms

In Figures 4.10–4.18 we observe the result of the performance propagation of three recommendation algorithms, namely Content-Based Filtering algorithm, Collaborative Filtering algorithm and a popularity-based algorithm (Popularity-Based) on the three dimensions of user attributes (language, city and type). In each simulation, we run all three algorithms in the same simulation settings, with the nine subfigures according to different algorithms and user attributes.

In all of the subgraphs, we can see the content adoption rate of users increases rapidly in the first two time steps, and then the rate of growth slows down and levels off, showing a trend of obvious decreasing marginal effect. This is the typical information diffusion in the scale-free network: fast propagation at the beginning, but as it approaches saturation, the propagation capacity will be significantly reduced.

In all of the three algorithms, the collaborative filtering algorithm (subfigures d, e and f) has the best performance in propagation speed and coverage. The algorithm fully plays the role of the similarity of interests between users; it could deliver the content to users who are more interested faster by taking advantage of the network proximity structure, thus effectively triggering the network effect.

The overall spreading performance of the content-based recommendation algorithm (subfigures a, b and c) is in the middle. Its spreading rate is relatively slow, the main reason is that the algorithm depends on the feature information of items rather than the behavioral relationship between users, and thus has limited propagation ability in the network structure. We could also find that the algorithm's performance is slightly different in different user attribute dimensions, the spreading effect based on the "content type" dimension is relatively stable.

The popularity-based recommendation model (subgraphs g, h, i) introduces a time-decaying mechanism for the propagation probability, and exhibits more diverse diffusion characteristics. Although the method shows a strong spreading ability in the initial stage, its curve stabilizes earlier and fluctuates more among different user groups. This phenomenon is particularly pronounced in language or genre subgroups with smaller user bases, suggesting that the decay mechanism has a greater impact on long-term diffusion in smaller populations.

It is interesting to note that the spread gap between different user groups (e.g., language, city, type) is most significant in the popularity-based model. This suggests that a single strategy relying on popularity may inadvertently exacerbate the imbalance in content exposure, allowing groups that are already dominant to receive more referral resources.

Combining collaborative filtering algorithms with epidemic propagation modeling methods provides new perspectives for understanding user behavior and network interaction mechanisms. On the one hand, the method highlights the important influence of initial recommendation strength on the overall propagation path; on the other hand, by

introducing the time-decaying propagation rate, it more realistically simulates the behavioral characteristics of users' decreasing interest over time, such as information fatigue or interest transfer.

In addition, this study also emphasizes the influence of network structure and user attribute differences on recommendation propagation efficiency. In a network structure with high connectivity and "hub" users, it is easier for recommendation content to achieve rapid diffusion, while in a network with sparse structure or more edge users, recommendation diffusion may be limited, and more targeted strategies need to be designed to improve the effect.

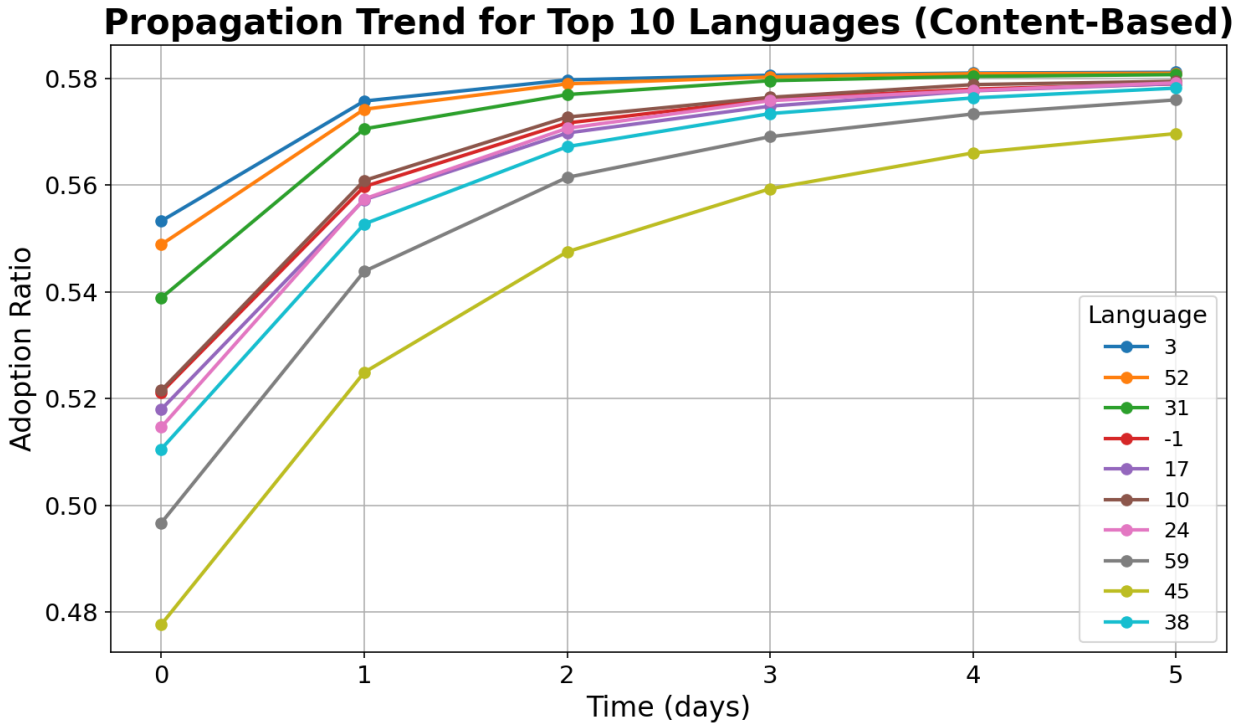

Figure 4.10: Language—Content-Based.

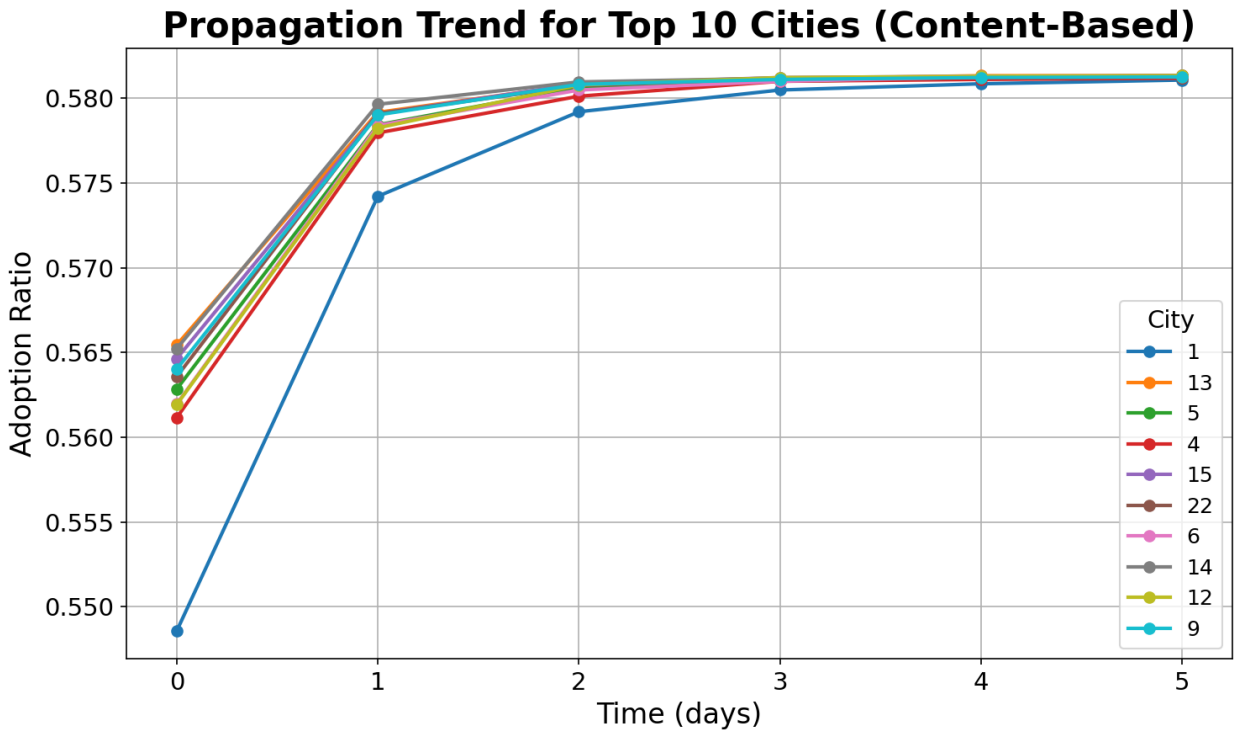

Figure 4.11: City—Content-Based.

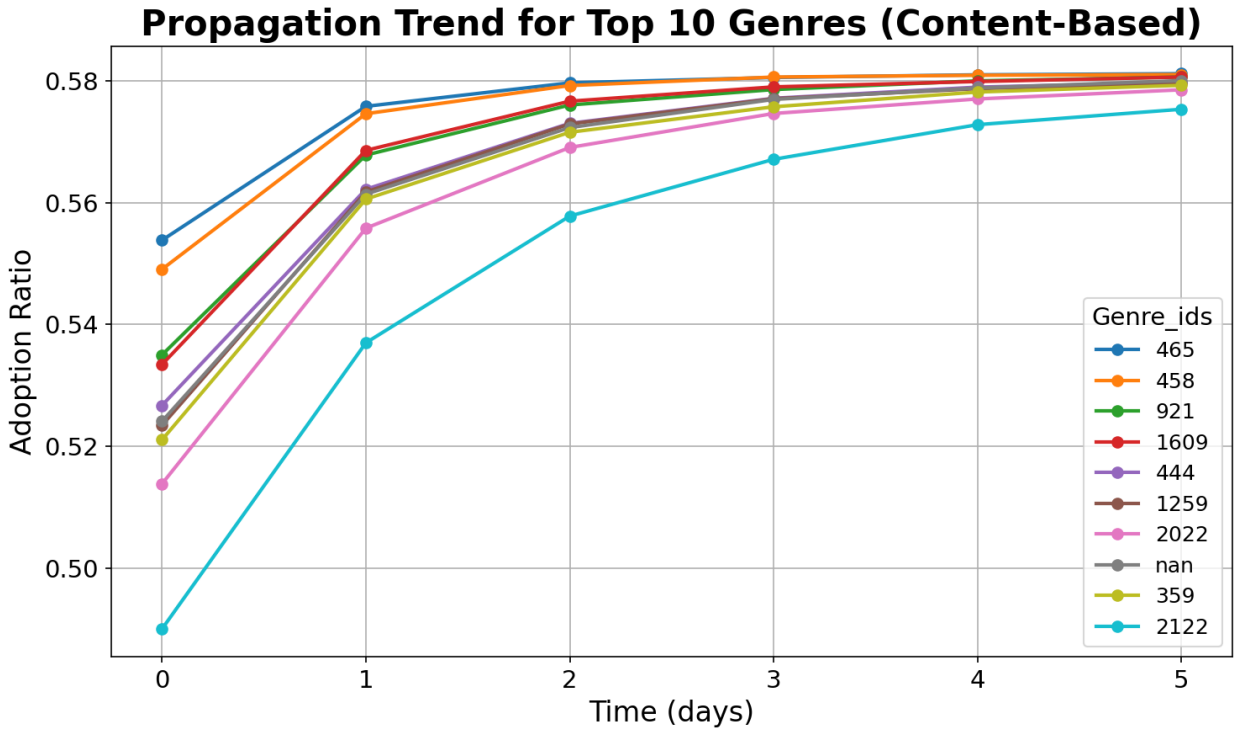

Figure 4.12: Genre—Content-Based.

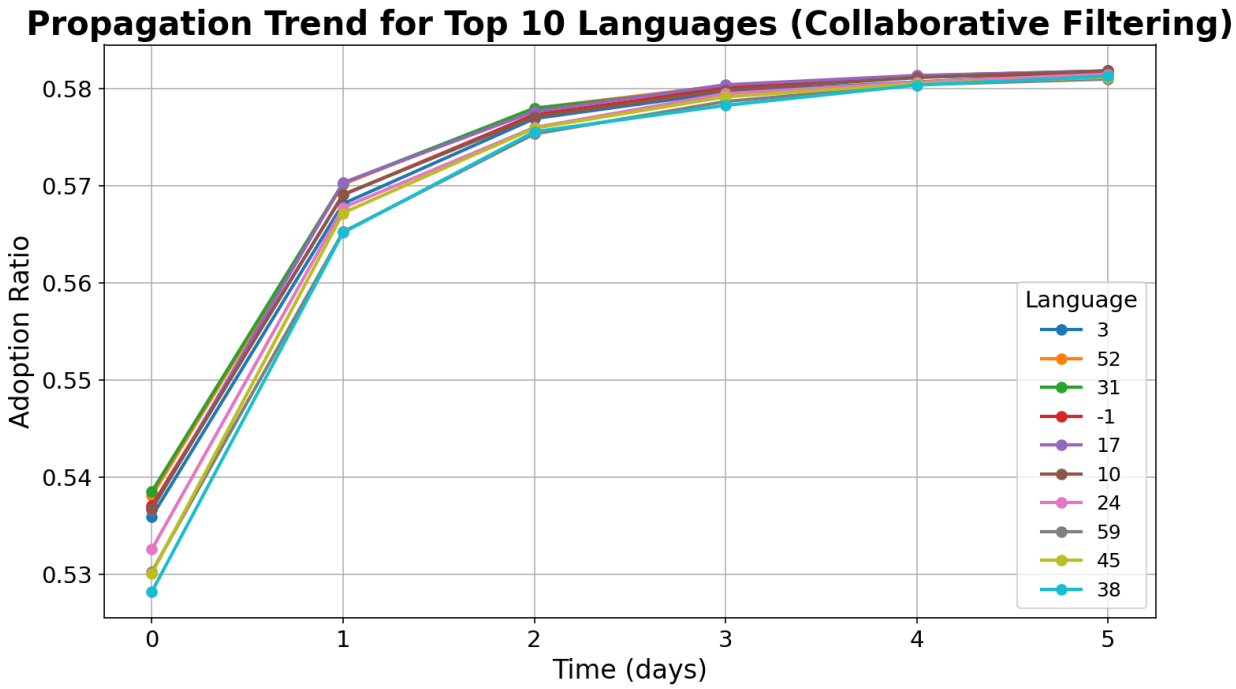

Figure 4.13: Language—Collaborative Filtering.

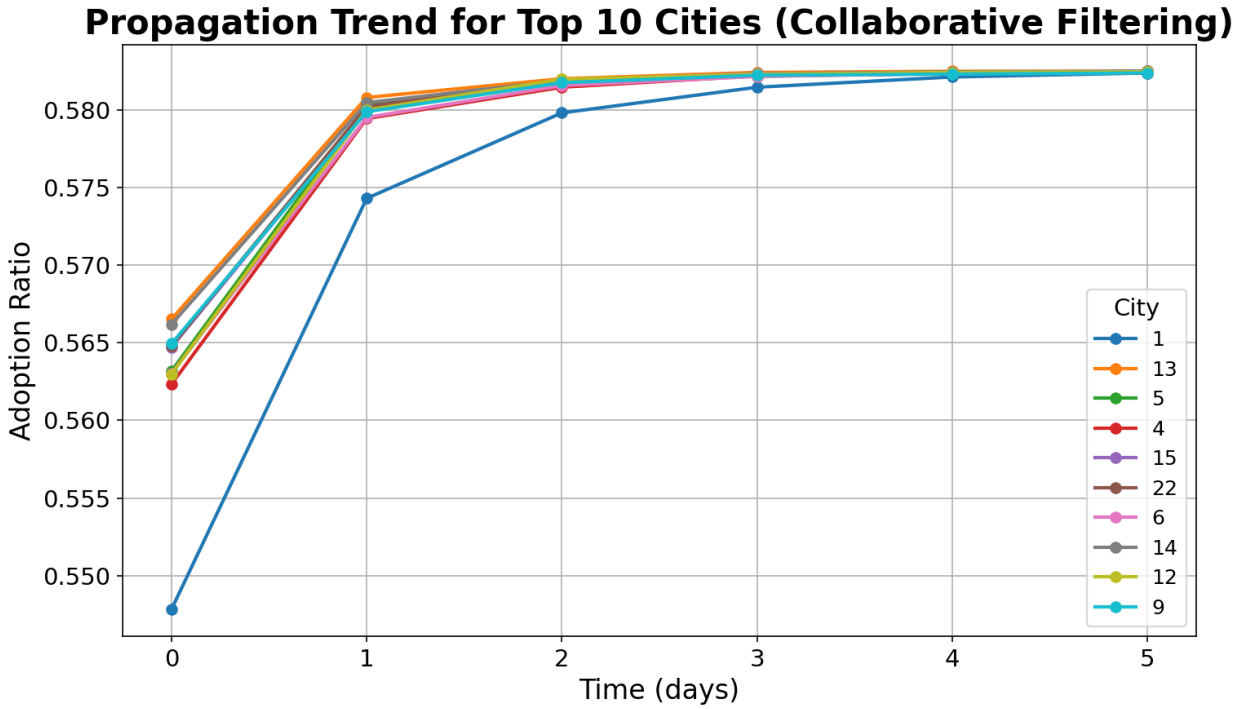

Figure 4.14: City—Collaborative Filtering.

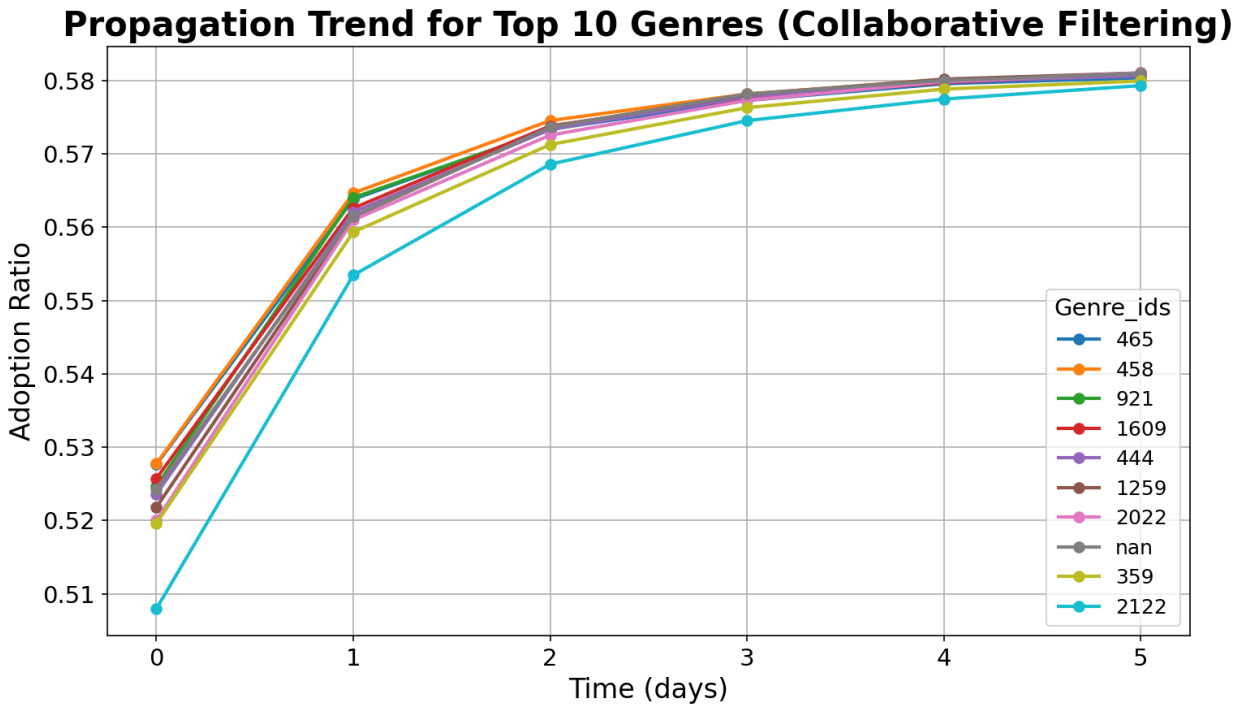

Figure 4.15: Genre—Collaborative Filtering.

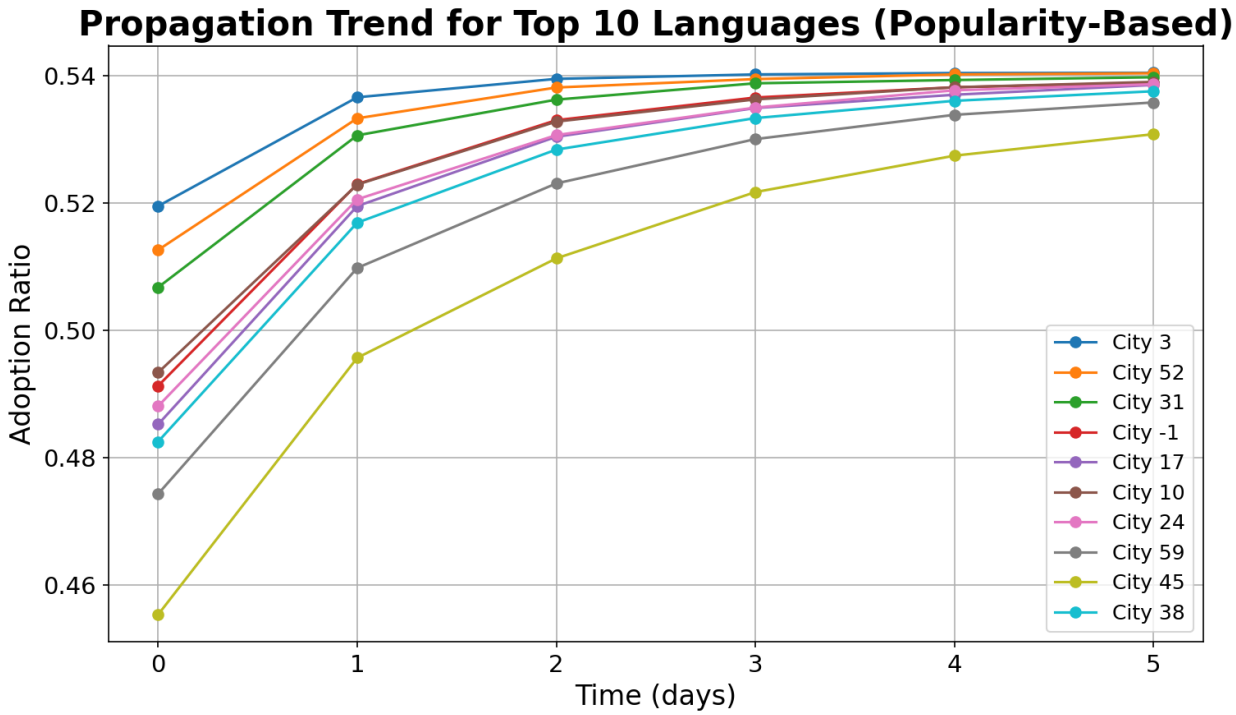

Figure 4.16: Language—Popularity-Based.

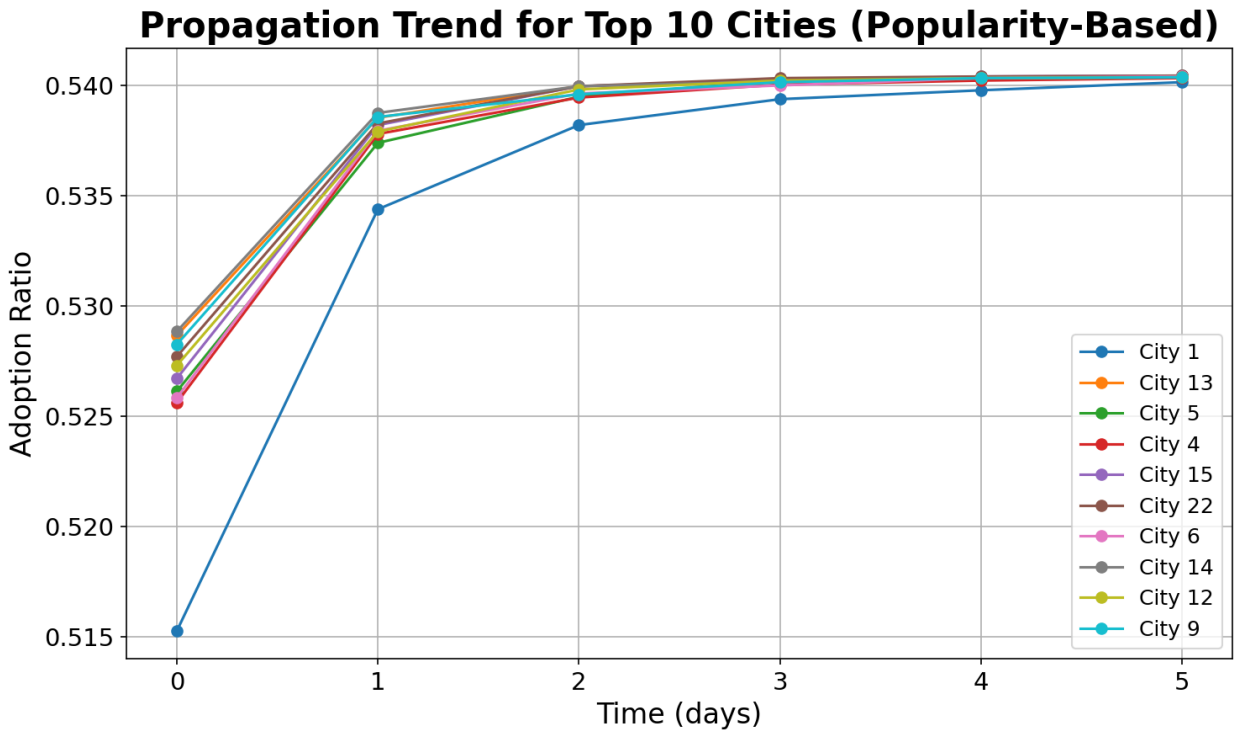

Figure 4.17: City—Popularity-Based.

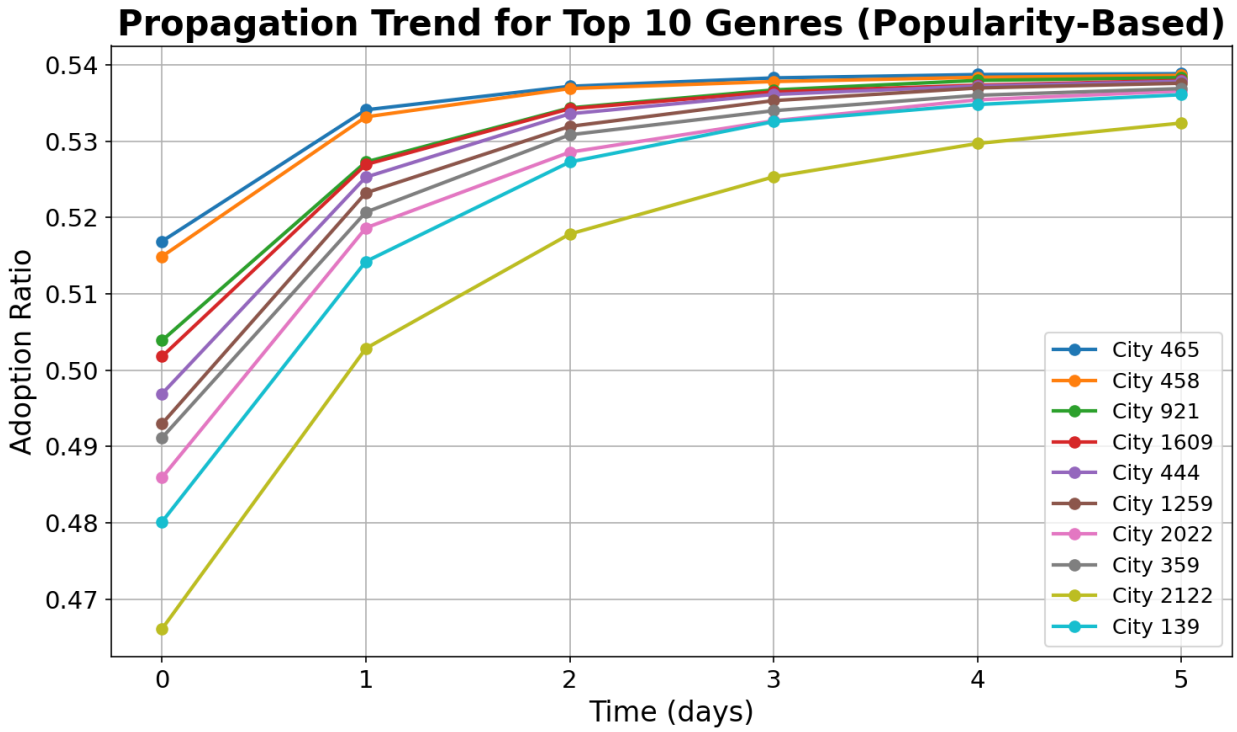

Figure 4.18: Genre—Popularity-Based.

4.5 Discussion

In fact, although the modeling method of recommendation propagation dynamics in this chapter based on epidemiology provides some new ideas, there are also many shortcomings. Here we summarize several obvious deficiencies to be improved in this work, hoping to provide some direction for follow-up work.

Firstly, this chapter employs the classical SI (susceptible-infected) model, where a user who has seen and clicked on the recommended content is considered "infected" and will not recover. It is assumed that the infected individuals do not lose their interest in the item and will be eternally infected. However, user behavior in real life is generally more complicated and also time-aware. In the real propagation process, there may be two kinds of effects of time on users. On the one hand, in the short term, users may quickly forget the recommended contents and lose interest in it; on the other hand, due to the intervention of external forces (such as information overload or deletion of platform authority), users may temporarily or permanently interrupt the propagation. But these processes are impossible to reflect in an SI model.

In order to depict the change of user behavior more realistically, it is also possible to introduce a more complex model of user communication in future research. For instance, in the SIR model, users can transition to the "recovered" state after a certain period of "infection" [19], which is suitable for the situation where the content attraction is declining or users are gradually cooling down. In the SEIR model, users have to go through the "exposed" state before becoming infected [31], and it is used to model the propagation behavior of users in the latent period after receiving the recommendation. The extended models mentioned above are expected to further enrich the description of the diffusion of recommended content and the time characteristics of user response.

In addition, the scale-free network used for propagation simulation in this chapter was generated by the Barabasi–Albert model, which can better reflect the "central node" and "long-tailed distribution" structural characteristics of the real social network, but it is still

an idealized modeling. Real networks have many complex attributes, such as community structure, homogeneous connection [48], asymmetric influence, etc. Therefore, future research can construct a more representative network structure based on real social platforms or content dissemination data, so as to improve the external validity and predictive power of the model.

From the perspective of propagation performance of recommendation algorithms, it is observed that collaborative filtering algorithms have more balanced and efficient propagation ability across different user groups, which fully leverages the potential of user preference similarity and network structure. In comparison, the performance of popularity-based recommendation methods decreases more rapidly over time after an impressive initial propagation stage, while their performance also varies across different user groups more. This gap indicates the possibility that a popularity-driven recommendation mechanism might further amplify the imbalance of content exposure, leading to higher exposure in mainstream groups and further marginalization of marginalized groups.

These results provide crucial guidance for developing practical recommender systems. Recently, as more attention has been paid to platform user stickiness, fairness, content distribution, and other issues, how to improve recommendation efficiency and distribute content more scientifically has become an urgent problem to be solved. Our experiments have found that the recommendation algorithm can not only improve the individual recommendation experience, but also affect the overall content diffusion mode at the network level. Therefore, when designing more socially responsible recommendation strategies, platforms need to consider not only the dissemination mechanism itself, but also the characteristics of the user structure. Methodologically, by connecting collaborative filtering algorithms with epidemic propagation models, we have found a new approach for studying user behavior mechanisms and network interaction. On the one hand, the method emphasizes the impact of the initial recommendation strength on the entire propagation path; on the other hand, it introduces the time-varying propagation rate to more

realistically reflect the behavioral characteristics of users' gradually waning interest, such as fatigue or change of interest.

4.6 Summary

By integrating recommender systems with epidemiological modeling, this study provides a new perspective for modeling and analyzing the spread of recommended content in user networks. We construct a scale-free network based on actual clickstream data, model the recommendation behavior as an "infection"-like spreading process, and use a dynamic SI model to evaluate the diffusion efficiency of three mainstream recommendation algorithms in different scenarios. The results show that different recommendation strategies have different spreading speed, scope and persistence, and the spreading process is also significantly affected by the time granularity modeling and the user interest decay.

Specifically, collaborative filtering algorithms show more balanced and efficient spreading ability among most user groups, while popularity-based methods, despite their higher spreading intensity in the early stage, have a relative disadvantage in long-term sustainability and fairness. Moreover, our simulations further reveal that the patterns of recommendation propagation also show significant differences in days, weeks and years, further highlighting the importance of using a time-sensitive evaluation mechanism as a complement to the traditional static performance metrics.

This work not only broadens the modeling of recommender systems, but also supplements the theoretical support for how algorithms shape user behavior in complex social networks. The introduction of the epidemiological communication logic into recommender system evaluation provides a new way for system designers to optimize recommendation strategies and user engagement management. The current model continues to operate under certain restrictive assumptions. For instance, the SI model does not consider the

recovery mechanism of user interest and the impact of external events on recommendation propagation. Therefore, future work can further extend the current framework in the following aspects.

On the one hand, it can be beneficial to incorporate more complicated propagation mechanisms, such as SIR or SEIR, to better reproduce the attenuation and resurgence process of user interest in reality; on the other hand, we should also attach importance to the heterogeneity of users themselves, such as the differences of user behaviors, the fluctuations of user preferences and users' sensitivity to the propagation of the recommended content to improve the personalization of the model and its conformity to reality. Additionally, in the future, we can also look at the mechanism of parallel propagation of multiple recommendation programs in the network [76], and see how they compete or complement each other along the propagation path.

On the other hand, integrating the platform's real time exposure data and contextual information will bring additional gains in interpretability and predictability. Last but not least, generalizing the current single static network structure and migrating to a dynamic network [80] or a multi-layer social network [22] will also help to reveal more details about the effect of the user's cross-platform behavior on the propagation path [78]. In conclusion, this work contributes new theoretical tools and ideas to the modeling of propagation in recommender systems, with practical values to the design of future more complex, dynamic and network-aware recommender systems.

Overall, this chapter shows that dynamic behavior modeling can improve system efficiency by reducing redundant or low-impact propagation. This insight prepares the transition to Chapter 5. In recommender systems, inefficient propagation wastes computation and communication resources; in SQL database clusters, inefficient table placement triggers unnecessary cold-node wakeups. Both problems require anticipating future demand before resources are consumed.

Chapter 5

Data-Mining-Driven Energy-Aware Prefetching for Database Clusters

The previous chapters establish the foundation for GreenDB_DM. Chapter 3 shows that SQL query logs contain recurring table co-access patterns, while Chapter 4 shows that dynamic behavior modeling can improve system efficiency by reducing redundant activity. This chapter applies these ideas to energy-aware SQL database optimization by using mined workload patterns to guide hot-node placement, reduce unnecessary cold-node wakeups, and lower cluster energy consumption.

Modern data centers consume an estimated 1-2% of global electricity, with database clusters constituting one of the largest contributors due to their continuous availability requirements and high I/O intensity [7, 50]. As enterprise data volumes continue to grow, the energy footprint of large-scale database deployments has become a primary operational concern alongside performance and reliability. This trend has motivated extensive research on energy-efficient database systems, including disk-level techniques such as PRE-BUD [72], node-level Dynamic Power Management (DPM) [9], and more recently the GreenDB framework [82], which partitions a cluster into one always-active *hot node* and multiple low-power *cold nodes*, using prefetching to keep cold nodes in standby for as long as possible. These systems demonstrate that substantial energy savings - exceeding 90% compared with synchronous replication baselines - can be achieved through careful power-state management.

A common assumption underlying these approaches is that determining *which* tables to place on the hot node is a secondary concern, typically addressed using a simple frequency-based heuristic: ranking tables by access count and placing the top-K tables

on the hot node up to its capacity. The original GreenDB work, for instance, takes the hot-node hit rate as an input parameter and focuses its analysis on the energy-saving model rather than on how the hit rate itself is achieved [82]. While effective under workloads dominated by a small subset of tables, frequency-based placement implicitly assumes a highly skewed table-access distribution. This assumption breaks down in two increasingly common scenarios: *multi-tenant SaaS deployments*, where multiple customers share a database without a dominant tenant [5]; and *multi-workflow analytical workloads*, where several business processes run concurrently, each touching its own group of related tables. In these settings, accesses are dispersed across multiple table groups, and frequency-based placement fragments hot-node capacity across partial members of many groups rather than concentrating it on complete, jointly-needed table sets.

Our key observation is that real database queries rarely access tables in isolation; instead, a single query typically touches a *group* of semantically related tables that together support a specific workflow - e.g., `users`, `orders`, `order_items`, and `payments` co-occurring in an e-commerce checkout query. If such co-access groups can be discovered automatically and placed *jointly* on hot nodes, every query within that workflow becomes a full hit, thereby eliminating the cold-node wakeups that frequency-based placement incurs whenever it places only a subset of a group's tables on the hot nodes. This insight reframes hot-node placement from a per-table ranking problem into a *group-aware coverage problem*. To realize this idea, we leverage frequent pattern mining [29, 1] - specifically the GFP-Growth algorithm recently proposed for SQL workload analysis [30] - to extract frequently co-accessed table groups from historical query logs. We cast hot-node placement as a set-cover optimization [17] that maximizes full-query hit rate rather than per-table frequency. The resulting hit-rate improvements are then translated into cluster-level energy savings through GreenDB's five-case energy model.

We integrate these ideas into GreenDB_DM, a data-mining-driven extension of the GreenDB framework. The contributions of this paper are summarized as follows.

- We identify a previously overlooked limitation of frequency-based placement in node-level energy-efficient database clusters and characterize the workload regimes under which this limitation becomes significant.
- We propose a *group-aware* hot-node placement framework that combines GFP-Growth pattern mining with a set-cover-based placement algorithm; we integrate it with the GreenDB five-case energy model without altering the underlying energy accounting framework.
- We conduct extensive evaluations against four placement baselines (Random, Frequency, LRU, Markov) and four energy-management baselines (Non-E, DPM, PRE-BUD, GreenDB) on controlled synthetic workloads spanning a wide range of skewness, group coherence, cluster scale, and data size. GreenDB_DM raises full-query hit rate by 5 - 15% across the entire skewness range, and lowers cluster energy by 2.4 - 4.7% of GreenDB.
- We provide practical deployment guidance by characterizing the conditions under which group-aware placement provides the greatest benefit and those under which frequency-based placement remains near-optimal.

The remainder of this paper is organized as follows. Section 5.1 provides background on the GreenDB framework and motivates the group-aware approach with a concrete example. Section 5.2 presents the GreenDB_DM design and algorithms. Section 5.3 reports our experimental evaluation. Section 5.4 discusses applicability and limitations, and finally Section 5.5 concludes this study.

5.1 Background and Motivation

This section provides background on the GreenDB framework on which our work builds, outlines the limitations of frequency-based placement through a concrete example, and characterizes the workflow-coherent structure of real-world database workloads that motivates a group-aware approach.

5.1.1 GreenDB Architecture and Energy Model

Our work builds upon the GreenDB framework proposed by Zhou *et al.* [82]. We briefly summarize its key components below; readers familiar with GreenDB may skip to Section 5.1.2.

GreenDB partitions a database cluster into one always-active *hot node* and N low-power *cold nodes*. The hot node serves queries directly from a working set of frequently accessed tables, while cold nodes hold the remainder of the database and remain in low-power standby until needed. When a query arrives, the system first checks whether the required tables reside on the hot node; if so, the query is served locally (a *full hit*). Otherwise, one or more cold nodes are woken up to retrieve the missing tables. This asymmetric architecture exploits the observation that keeping nodes in standby ($P_S = 2.5\,\text{W}$) consumes substantially less power than active ($P_A = 58\,\text{W}$) or idle ($P_I = 27.7\,\text{W}$) states, but transitions between states incur both time ($T_U = 8.5\,\text{s}$ wake-up, $T_D = 3.3\,\text{s}$ sleep-down) and energy ($E_U = 35\,\text{J}$, $E_D = 32\,\text{J}$) overhead.

The decision of when to wake or sleep a cold node is governed by the *break-even time* $T_{BE} = 13.3\,\text{s}$: the minimum idle period during which standby remains more energy-efficient than staying idle. GreenDB introduces a five-case energy model (Equations 5, 7, 8, 10, and 12 in [82]) that quantifies the energy savings of prefetching a table from a cold node to the hot node, based on the relative magnitudes of the surrounding idle periods and T_{BE}.

The total energy savings of GreenDB relative to a no-energy-management baseline (Non-E) depends critically on the *hot-node hit rate*-the fraction of queries that can be served entirely from tables present on the hot node.

In the original GreenDB evaluation, the hit rate is treated as an *input parameter*, varied between 20% and 90% to characterize the behavior of the energy model. However, the mechanism by which a high hit rate is achieved through table placement is left to the system designer. Our work addresses this gap by introducing a placement strategy that maximizes the hit rate for workloads in which it is most critical.

5.1.2 The Limitation of Frequency-Based Placement

The most natural placement strategy—and the one implicitly assumed by GreenDB and similar systems—is to rank tables by individual access frequency and place the top-K tables on the hot node up to its capacity. We refer to this as *frequency-based placement*. Although this strategy works well when a small set of tables dominates the access distribution, it can fail significantly when access is dispersed across multiple workflow groups.

Consider a database with six independent workflow groups, each containing five tables. Within each group, all five tables are accessed together by the queries belonging to that workflow (e.g., `users`, `orders`, `order_items`, `payments`, `shipping` forming a checkout workflow). Across groups, table accesses are roughly uniform - no single table or group dominates. Suppose the hot node has a capacity of 10 tables. Under frequency-based placement, the top-10 tables ranked by individual access count are distributed across all six groups—approximately two tables per group. The result is striking: although every group has some of its tables placed on the hot node, no group has all of its tables placed on the hot node. Every query in the workload still requires waking up at least one cold node to retrieve the missing tables in its group. As a result, the full-query hit rate is effectively zero, and the energy savings predicted by the GreenDB model collapse.

In contrast, under group-aware placement, the same 10 hot-node slots are allocated to two complete workflow groups. Although fewer groups are represented on the hot node, queries belonging to those two groups—one third of all queries—are served entirely locally. Cold-node wakeups are triggered only for queries from the remaining four groups, and when they occur, they retrieve coherent table sets that fully satisfy the corresponding queries.

The key insight is that hit rate is not determined by individual table access frequency, but by the *sets of tables that jointly satisfy queries*. When the workload exhibits a clear group structure, placing complete groups together is strictly more effective than selecting the top-K individually frequent tables. The challenge is to automatically discover these groups from query logs, without requiring manual schema annotation or workflow specification.

5.1.3 Workflow-Coherent Structure in Real Workloads

The motivating example above relies on the assumption that real database workloads exhibit workflow-coherent structure, in which queries access tables in groups rather than independently. We verify this assumption empirically using the Join Order Benchmark (JOB) [42], a standard benchmark consisting of 113 queries over 21 IMDb tables. Each query in JOB performs a multi-way join over a subset of the tables, with an average of 8.0 tables accessed per query.

An empirical analysis of the table co-access matrix on JOB shows that the conditional probability $P(t_j \mid t_i)$ of accessing table t_j when table t_i is accessed in the same query is far from uniform. Certain pairs of tables, such as `title`, `movie_companies`, and `movie_keyword`, co-occur in nearly every query in which they appear, whereas other pairs rarely co-occur. Across the 21 tables, we observe 150 ordered pairs (i, j) with $P(t_j \mid t_i) > 0.5$, indicating strong co-access relationships that a group-aware placement strategy can exploit.

The structural source of this co-access pattern is revealed by the organization of JOB queries into 33 *query families* (1a, 1b, 1c; 2a, 2b; etc.), where queries within the same family represent variants of the same business question under different selectivity parameters. Queries within the same family share nearly identical table sets, whereas different families exhibit distinct but partially overlapping table sets. This is precisely the workflow-coherent structure exploited by group-aware placement: each family corresponds to a workflow group, and placing its tables together on the hot node converts every query in that family into a full hit.

Such structure is a natural consequence of how applications interact with databases: a single business operation translates into a predictable set or sequence of table accesses, and the same operation is repeated across many sessions. Group-aware placement is an algorithmic embodiment of this structural regularity.

5.2 The GreenDB_DM Approach

This section presents the design of GreenDB_DM. We first give an overview of the system pipeline (Section 5.2.1), then formalize the placement problem and present our two algorithms: GFP-Growth for discovering frequent table groups (Section 5.2.3) and a greedy set-cover algorithm for converting these groups into a placement decision (Section 5.2.4). Finally, we describe how placement decisions are integrated with the GreenDB energy model in Section 5.2.5.

5.2.1 System Overview

Fig. 5.1 illustrates the pipeline of GreenDB_DM.The system operates in two stages: *Offline* and *Online*. In the *Offline* stage, GreenDB_DM ingests a historical SQL query log $\mathcal{Q} = \{q_1, q_2, \ldots, q_n\}$, where each query q_i is associated with a set of accessed tables. It applies the GFP-Growth algorithm (Algorithm 2) to discover frequent table groups $\mathcal{F} =$

$\{F_1, F_2, \ldots, F_m\}$, where each $F_j \subseteq \mathcal{T}$ is a subset of the table universe $\mathcal{T}$ that co-occurs in queries above a specified support threshold. A greedy set-cover algorithm (Algorithm 3) then selects the hot-node table set $\mathcal{H}^* \subseteq \mathcal{T}$ that maximizes the expected full-query hit rate subject to the hot-node capacity constraint $|\mathcal{H}^*| \leq C$.

In the *Online* stage, the GreenDB cluster operates as described in [82]: queries that access only tables in $\mathcal{H}^*$ are served entirely by the hot node (full hits), while other queries trigger cold-node wakeups. The energy cost of each query is computed using the unmodified GreenDB five-case energy model. The mining and placement steps may be re-executed periodically (e.g., daily) to adapt to workload drift; we found in our experiments that mining itself takes only seconds even on workloads of tens of thousands of queries (Section 5.3.4).

5.2.2 Problem Formulation

We formalize the hot-node placement as a coverage optimization problem. Let $\mathcal{T} = \{t_1, t_2, \ldots, t_N\}$ be the set of all tables in the database, and let $\mathcal{Q}_{\text{test}} = \{q_1, \ldots, q_n\}$ be the expected (test) workload, where each $q_i \subseteq \mathcal{T}$ is the set of tables accessed by query q_i. Given a hot-node capacity $C \in \mathbb{N}$, we seek a hot-node set $\mathcal{H} \subseteq \mathcal{T}$ with $|\mathcal{H}| \leq C$ that maximizes the *full-query hit rate*:

$$\mathcal{H}^* = \arg \max_{\mathcal{H} \subseteq \mathcal{T},\, |\mathcal{H}| \leq C} \frac{1}{|\mathcal{Q}_{\text{test}}|} \sum_{q \in \mathcal{Q}_{\text{test}}} \mathbb{1}[q \subseteq \mathcal{H}]. \tag{5.1}$$

A query q contributes to the hit rate if and only if all of its accessed tables reside on the hot node (i.e., $q \subseteq \mathcal{H}$). This is precisely the criterion that triggers the no-wakeup case in the GreenDB energy model.

The objective in (5.1) is a special case of the *maximum coverage problem*, which is known to be NP-hard [23] but admits a $(1 - 1/e)$- approximation via a simple greedy algorithm. Two considerations make a direct application of this approximation non-trivial

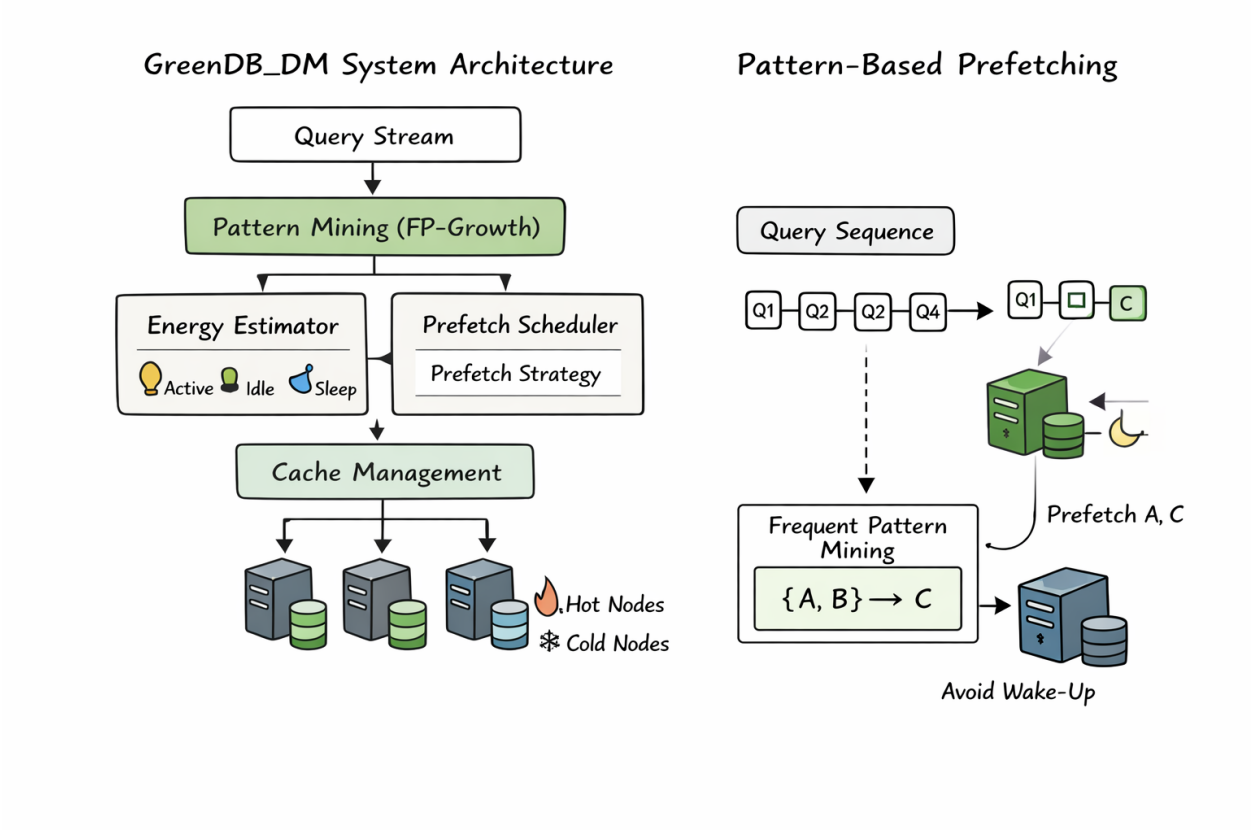

Figure 5.1: GreenDB_DM system overview. The left panel shows the offline pipeline: a query stream is fed to a pattern mining module (FP-Growth) whose output drives an energy estimator and a prefetch scheduler; together these decisions guide the cache management layer that partitions tables across hot and cold nodes. The right panel illustrates how frequent patterns ($\{A, B\} \rightarrow C$) discovered from past queries enable proactive prefetching of table C when A and B are accessed, allowing the cold node holding C to remain in standby longer and avoid an unnecessary wake-up.

in our setting. First, the test workload $\mathcal{Q}_{\text{test}}$ is unknown at placement time; we approximate it using the observed training workload. Second, treating each individual training query as a coverage target leads to a problem of size $O(|\mathcal{Q}_{\text{train}}|)$, which is computationally expensive when query logs are large. GreenDB_DM addresses both issues by first *compressing* the training queries into a much smaller set of frequent table groups via GFP-Growth, and then performing greedy coverage over these groups - with the empirical support count of each group serving as a proxy for the number of queries it covers.

5.2.3 GFP-Growth: Mining Frequent Table Groups

The first stage of GreenDB_DM mines frequent table groups from the training query log using GFP-Growth, a grouped variant of FP-Growth [29] introduced in the GARMT framework [30]. The proposed approach is summarized in Algorithm 2, and its key design choices are described below.

Algorithm 2. GFP-Growth Mining of Frequent Table Groups

Require: Training query log $\mathcal{Q}_{\text{train}}$, group size k, minimum support σ

Ensure: Frequent table groups $\mathcal{F}$ with support counts

1: Partition $\mathcal{Q}_{\text{train}}$ into groups of k consecutive queries each:

$$\mathcal{G}_j = \{q_{k(j-1)+1}, \ldots, q_{kj}\},\ j = 1, \ldots, \lceil |\mathcal{Q}_{\text{train}}|/k \rceil$$

2: For each group $\mathcal{G}_j$, form the merged transaction $T_j \leftarrow \bigcup_{q \in \mathcal{G}_j} q$

3: Construct an FP-tree from $\{T_1, T_2, \ldots, T_{\lceil n/k \rceil}\}$ with minimum support threshold σ

4: $\mathcal{F} \leftarrow \emptyset$

5: Run FP-tree mining with prefix $\emptyset$, threshold σ, output $\mathcal{F}$

6: **return** $\mathcal{F}$

7:

8: **procedure** MINEFPTREE(node, prefix, σ, $\mathcal{F}$)

9: **for** each item α in node's header table **do**

10: $\beta \leftarrow$ prefix $\cup \{\alpha\}$

11: count$(\beta) \leftarrow$ support of β in tree

12: **if** count$(\beta) \geq \sigma$ **then**

13: $\mathcal{F} \leftarrow \mathcal{F} \cup \{(\beta, \text{count}(\beta))\}$

14: Construct conditional FP-tree of β

15: Run MineFPTree(cond. tree, β, σ, $\mathcal{F}$)

16: **end if**

17: **end for**

18: **end procedure**

Motivation for Grouping. Standard FP-Growth treats each query as an independent transaction. In SQL workloads, however, individual queries are often short, involving only a few tables. As a result, an itemset that appears infrequently on a per-query basis may nonetheless represent a meaningful workflow pattern that recurs across many queries [30]. Grouping aggregates k consecutive queries into a single transaction, increasing the density of the transaction database and enabling the mining algorithm to capture workflow patterns that span multiple queries. In our experiments (Section 5.3.4), we vary k from 3 to 80 and find that moderate values ($k = 5$ - 20) yield the best balance between pattern quality and computational cost.

Choice of Support Threshold. The minimum support σ controls the trade-off between the number of patterns discovered and their statistical reliability. A low minimum support threshold produces many spurious patterns and increases mining time, whereas a high threshold may exclude meaningful but infrequent groups. We use $\sigma = 0.02$ (i.e., 2% of the number of grouped transactions) as the default, following the recommendation in [30]. The sensitivity of GreenDB_DM to this parameter is evaluated in Section 5.3.4.

Maximum Pattern Depth. To limit mining overhead on workloads with many tables per query, we cap the recursion depth of FP-tree mining at $d_{\max} = 4$. Deeper patterns

are unlikely to fit within a typical hot node's capacity and contribute disproportionately to mining time.

Proposition 1 (GFP-Growth Complexity)**.**

Let $n = |\mathcal{Q}_{\text{train}}|$ be the number of training queries, k the group size, $\bar{\ell}$ the average number of tables per query, $d_{\max}$ the maximum FP-tree mining depth, and $|\mathcal{F}|$ the number of discovered frequent patterns. Then GFP-Growth runs in time can be expressed as:

$$O\big(n \cdot \bar{\ell} \cdot \log(k\bar{\ell}) + |\mathcal{F}|\big), \tag{5.2}$$

with the worst-case pattern count bounded by $|\mathcal{F}| \leq \binom{|\mathcal{T}|}{d_{\max}}$.

Proof. Constructing the $\lceil n/k \rceil$ grouped transactions requires $O(n)$ time. Each grouped transaction, with an expected length of $k\bar{\ell}$, is then inserted into the FP-tree. This process involves sorting the items by global frequency, which costs $O(k\bar{\ell}\log(k\bar{\ell}))$ per transaction, followed by traversing and updating the tree at a cost of $O(k\bar{\ell})$. Summing over all $\lceil n/k \rceil$ grouped transactions yields the first term in (5.2). Mining the resulting FP-tree produces $|\mathcal{F}|$ frequent patterns, each processed $O(1)$ times under standard FP-Growth bookkeeping. The depth bound $d_{\max}$ further limits the size of the search space, capping the maximum pattern length and bounding the number of candidate patterns by $\binom{|\mathcal{T}|}{d_{\max}}$.

□

Corollary 1 (Practical Tractability)**.** *For typical SQL workloads with bounded $\bar{\ell}$ and fixed k, GFP-Growth runs in time* linear *in the query-log size n. In our experiments on workloads of up to $84{,}000$ queries (Section 5.3), the mining phase completed in less than 200,ms on a single CPU core.*

5.2.4 Group-Aware Placement via Greedy Set Cover

Given the frequent table groups $\mathcal{F}$ produced by GFP-Growth, the second stage of GreenDB_DM selects the hot-node set $\mathcal{H}^*$ that maximizes coverage of training queries (a

proxy for test queries). To accomplish this, we use a greedy set-cover algorithm shown in Algorithm 3.

Algorithm 3. Greedy Set-Cover Placement

Require: Table universe $\mathcal{T}$, training queries $\mathcal{Q}_{\text{train}}$, hot-node capacity C

Ensure: Hot-node table set $\mathcal{H}^*$, $|\mathcal{H}^*| \leq C$

$\mathcal{H} \leftarrow \emptyset$

Let $\mathcal{Q}_{\text{fit}} \leftarrow \{q \in \mathcal{Q}_{\text{train}} : |q| \leq C\}$ {queries that could fit at all}

Initialize COVERED $\leftarrow \emptyset$

while $|\mathcal{H}| < C$ **do**

  $t^* \leftarrow \arg\max_{t \in \mathcal{T} \setminus \mathcal{H}} \; |\{q \in \mathcal{Q}_{\text{fit}} \setminus \text{COVERED} : q \subseteq \mathcal{H} \cup \{t\}\}|$

  if the maximum value above equals 0 **then**

    break {no more queries can be covered}

  end if

  $\mathcal{H} \leftarrow \mathcal{H} \cup \{t^*\}$

  COVERED $\leftarrow$ COVERED $\cup \{q \in \mathcal{Q}_{\text{fit}} : q \subseteq \mathcal{H}\}$

end while

if $|\mathcal{H}| < C$ **then**

  Fill remaining slots with most frequent unselected tables {tie-breaker for low-coverage workloads}

end if

return $\mathcal{H}^* \leftarrow \mathcal{H}$

Design Rationale. The greedy strategy is the standard $(1-1/e)$-approximation for the maximum coverage problem [54, 17]. At each step, the algorithm selects the table whose addition to the hot node converts the largest number of previously uncovered queries into full hits. Because the objective function in (5.1) is monotone and submodular with respect

to $\mathcal{H}$, the greedy algorithm provides a constant-factor approximation guarantee. Empirically, on the workloads evaluated in this study, the greedy solution remains within a few percentage points of the optimal solution obtained through exhaustive search on small instances.

Role of Frequent Groups $\mathcal{F}$. Although Algorithm 3 iterates over individual queries in $\mathcal{Q}_{\text{fit}}$, the mined groups $\mathcal{F}$ from Algorithm 2 drastically reduce the effective size of the input. Specifically, two queries that touch the same table set are coalesced into a single coverage target weighted by their joint frequency, and queries whose table sets do not intersect any group above support σ are pruned. This yields a substantially smaller and more representative coverage instance.

Frequency Tie-breaker. If the workload is highly concentrated and the greedy procedure exhausts useful coverage before filling all C slots (i.e., remaining queries either fit fully in current $\mathcal{H}$ or are larger than C), we fill remaining slots with the most-frequent unselected tables. This design ensures that GreenDB_DM never performs worse than frequency-based placement, regardless of workload characteristics. We empirically validate this property in Section 5.3.2.

Complexity. The outer loop executes at most C times. In each iteration, the algorithm evaluates up to $|\mathcal{T}| - |\mathcal{H}|$ candidate tables, for each candidate scans the set of training queries in $|\mathcal{Q}_{\text{fit}}|$ training queries. Consequently, the total running time is $O(C \cdot |\mathcal{T}| \cdot |\mathcal{Q}_{\text{train}}|)$. With workload sizes typical in our experiments ($|\mathcal{T}| \leq 100$, $|\mathcal{Q}_{\text{train}}| \leq 10^4$, $C \leq 30$), the placement phase completes within milliseconds on a single CPU core.

Approximation Guarantee. The greedy procedure is not merely heuristic in nature; its performance is supported by established results in combinatorial optimization. Specifically, the solution it produces is guaranteed to lie within a constant factor of the optimum. The following theorem formalizes this result.

Theorem 5.1 (Monotone Submodularity of Hit Coverage)**.** *Fix a workload $\mathcal{Q}$ and define the set function $f : 2^{\mathcal{T}} \to \mathbb{R}_{\geq 0}$ by*

$$f(\mathcal{H}) \;=\; \sum_{q \in \mathcal{Q}} \mathbb{1}[\, q \subseteq \mathcal{H} \,]. \tag{5.3}$$

then f is monotone non-decreasing and submodular on the ground set $\mathcal{T}$.

Proof. *Monotonicity:* for any $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{T}$ and any query $q \in \mathcal{Q}$, the predicate $q \subseteq \mathcal{A}$ implies $q \subseteq \mathcal{B}$, so $\mathbb{1}[q \subseteq \mathcal{A}] \leq \mathbb{1}[q \subseteq \mathcal{B}]$. Summing over q yields $f(\mathcal{A}) \leq f(\mathcal{B})$. □

Proof. *Submodularity:* let $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{T}$, $t \in \mathcal{T} \setminus \mathcal{B}$, and the marginal gain be $\Delta(t \mid \mathcal{S}) \triangleq f(\mathcal{S} \cup \{t\}) - f(\mathcal{S})$. It is sufficient to show that $\Delta(t \mid \mathcal{A}) \geq \Delta(t \mid \mathcal{B})$. A query q contributes to $\Delta(t \mid \mathcal{S})$ if and only if $t \in q$ and $q \setminus \{t\} \subseteq \mathcal{S}$. Since $\mathcal{A} \subseteq \mathcal{B}$, the set of queries satisfying this condition with $\mathcal{S} = \mathcal{B}$ is a subset of those satisfying it with $\mathcal{S} = \mathcal{A}$. Hence $\Delta(t \mid \mathcal{A}) \geq \Delta(t \mid \mathcal{B})$, which is the definition of submodularity. □

Corollary 2 (Approximation Guarantee)**.** *Algorithm 3 returns a placement $\mathcal{H}_{\text{greedy}}$ satisfying*

$$f(\mathcal{H}_{\text{greedy}}) \;\geq\; \left(1 - \frac{1}{e}\right) \cdot f(\mathcal{H}^*) \;\approx\; 0.632 \cdot f(\mathcal{H}^*), \tag{5.4}$$

where $\mathcal{H}^*$ is the optimal placement of cardinality at most C. This bound is tight in the worst case [23].

Proof. The cardinality-constrained problem $\max_{|\mathcal{H}| \leq C} f(\mathcal{H})$ is a *monotone submodular maximization* problem under a cardinality constraint. Nemhauser, Wolsey, and Fisher [54] proved that the greedy algorithm achieves a $(1-1/e)$-approximation for this problem class, and Feige [23] showed this bound is tight under standard complexity assumptions. By Theorem 5.1, f satisfies the required hypotheses, and Algorithm 3 implements precisely this greedy procedure (the table t^* selected at each step maximizes the marginal gain $\Delta(t \mid \mathcal{H})$). □

Corollary 2 provides the central theoretical guarantee of GreenDB_DM: regardless of the workload distribution or underlying group structure, Algorithm 3 achieves hit coverage within a constant factor of the unattainable optimum, without requiring assumptions on the data. In practice, the gap is significantly smaller than this worst-case bound suggests, typically remaining within a few percentage points of the optimum on the workloads evaluated in our experiments.

Skewness Threshold Analysis. The worst-case bound in Corollary 2 is distribution-free. We now derive a sharper, distribution-aware lower bound on the hit-rate advantage of GFP-Growth over the Frequency baseline as a function of workload skewness. This analysis offers a first-principles explanation for the V-shaped degradation of the Frequency baseline observed in Fig. 5.2 5.3.

Definition (Group-coherent Workload). A workload $\mathcal{Q}$ is (k, s, ρ)-*coherent* if the table universe $\mathcal{T}$ can be partitioned into k disjoint groups $G_1, \ldots, G_k$ of equal size s (so that $|\mathcal{T}| = ks$), and at least a fraction $\rho \in (0, 1]$ of queries in $\mathcal{Q}$ access tables from exactly one group. Let π_j denote the share of single-group queries directed at group G_j, so that $\sum_{j=1}^{k} \pi_j = \rho$. The group-popularity distribution $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_k)$ follows a Zipf-like skewness parameter α, with $\pi_j \propto j^{-\alpha}$.

Theorem 5.2 (Hit-rate Gap on Coherent Workloads)**.** *Let $\mathcal{Q}$ be a (k, s, ρ)-coherent workload and let C denote the hot-node capacity. Assume $s \leq C < ks$, so that the hot node can accommodate at least one group but not all groups. Let h_{GFP} and h_{Freq} denote the full-query hit rates achieved by Algorithm 3 and the Frequency baseline, respectively. Then*

$$h_{\mathrm{GFP}} - h_{\mathrm{Freq}} \;\geq\; \rho \cdot \left[\sum_{j=1}^{\lfloor C/s \rfloor} \pi_j \;-\; \frac{C}{ks} \right]_+ , \tag{5.5}$$

where $[x]_+ \triangleq \max(x, 0)$.

Proof. GFP-Growth identifies the k groups via frequent pattern mining, where each group's s tables co-occur in at least the queries associated with that group. Algorithm 3 then greedily selects the $\lfloor C/s \rfloor$ most popular groups and places each of them entirely on the hot node. As a result, every query directed at a selected group becomes a full hit, yielding $h_{\mathrm{GFP}} \geq \rho \cdot \sum_{j=1}^{\lfloor C/s \rfloor} \pi_j$.

The Frequency scheme selects the top-C tables ranked by individual access frequency. Under a (k, s, ρ)-coherent workload with uniform per-query table access within each group, every table in group G_j receives the same expected access count, proportional to π_j/s. The top-C tables under this ranking form a prefix of the sorted multiset of per-table frequencies. When$C < ks$ and π_j follows a Zipf-like distribution, these top-C tables are distributed across at least $\lceil C/s \rceil \geq \lfloor C/s \rfloor + 1$ groups, and no group is fully represented unless C is an exact multiple of s. A query directed at group G_j becomes a full hit under Frequency placement only if all s of G_j's tables are included among the top-C selected tables. Under uniform-within-group access, the expected number of fully represented groups is at most $C/(ks) \cdot k = C/s$. This finding implies a per-query full-hit probability of at most $C/(ks)$ for a coherent query under the constant-fraction allocation model. Thus $h_{\mathrm{Freq}} \leq \rho \cdot C/(ks)$. Subtracting the two expressions and clamping to non-negative values yields (5.5). □

Corollary 3 (V-Shape Collapse of Frequency)**.** *Under a (k, s, ρ)-coherent workload with $k > 1$ groups of equal size s, hot-node capacity satisfying $s \leq C < ks$, and a group-popularity distribution characterized by skewness parameter α:*

1. *As $\alpha \to 0$ (*uniform groups*), we have $\pi_j \to 1/k$ for all j. Consequently, the bracketed term in* (5.5) *converges to $\lfloor C/s \rfloor/k - C/(ks)$, which is positive whenever s does not divide C. Therefore, the advantage remains positive, although its magnitude is small.*

2. *As* $\alpha \to \infty$ *(*one dominant group*),* $\pi_1 \to 1$ *and other* $\pi_j \to 0$*. Consequently, the bracket tends to* $1 - C/(ks)$*. Both strategies eventually capture the dominant group, and the gap closes when* $C \geq s$*.*

3. *At intermediate values of* α *(*moderate skew*), the top groups concentrate a substantial fraction of the workload mass, causing* $\sum_{j=1}^{\lfloor C/s \rfloor} \pi_j$ *to approach* ρ*, while* $C/(ks)$ *remains significantly smaller than* ρ*. Consequently, the gap in* (5.5) *reaches its maximum in this regime.*

This non-monotonic behavior in α *predicts the V-shape collapse of Frequency (and the corresponding peak in the GFP-Growth advantage) observed empirically in Fig. 5.2 and Fig. 5.3, especially around* $\alpha \in [0.8, 1.2]$*.*

The theorem identifies the workload regime where GreenDB_DM provides the largest gains: *coherent multi-group workloads with moderate skewness*, precisely the regime characteristic of multi-tenant SaaS and multi-workflow analytical systems. Outside this regime (uniform access or one dominant group), By the frequency tie-breaker in Algorithm 3 (line 13), GreenDB_DM performs no worse than Frequency; therefore, deployment remains safe even under an unknown workload regime.

5.2.5 Integration with the GreenDB Energy Model

A design principle of GreenDB_DM is to keep the energy accounting machinery of GreenDB unchanged. The hot set $\mathcal{H}^*$ output by Algorithm 3 is deployed directly in the GreenDB cluster, and the existing five-case energy model from [82] computes the resulting energy consumption based on the realized hit rate, query inter-arrival pattern, and node power-state transitions.

The following lemma makes the connection between *maximizing hit rate* (the objective of Algorithm 3) and *minimizing cluster energy* (the system-level goal) precise.

Lemma 1 (Monotone Energy Reduction in Hit Rate)**.** *Let $E_{\mathrm{GreenDB}}(h)$ denote the total cluster energy as a function of the hot-node full-hit rate $h \in [0,1]$, holding all other workload and hardware parameters fixed. Under the GreenDB five-case energy model [82] with parameters from Table 5.1, $E_{\mathrm{GreenDB}}(h)$ is continuously differentiable and strictly decreasing on $[0,1]$, with derivative*

$$\frac{\mathrm{d}E_{\mathrm{GreenDB}}}{\mathrm{d}h} \;=\; -N_q \cdot \Big[T_s(P_A - P_S) + (E_U + E_D)\Big], \tag{5.6}$$

where N_q is the number of queries in the evaluation window, T_s is the average per-query service time, and P_A, P_S, E_U, E_D are the active power, standby power, wake-up energy, and sleep-down energy from Table 5.1.

Proof. Each miss, occurring with probability $1-h$ per query, incurs two sources of energy overhead: (i) serving the query on a cold node for duration T_s, during which the node consumes active power P_A instead of standby power P_S; and (ii) a wake-up transition followed by a subsequent sleep-down transition, with energy costs $E_U + E_D$, respectively. Queries that result in a hit incur none of these additional costs.

Letting $m(h) = N_q(1-h)$ denote the expected number of misses, the contribution of cold-node activity to total energy is

$$E_{\mathrm{cold}}(h) = m(h) \cdot \big[T_s(P_A - P_S) + (E_U + E_D)\big] + \text{const},$$

where the constant subsumes the baseline standby energy of cold nodes during idle periods (independent of h). Substituting $m(h) = N_q(1-h)$ and differentiating with respect to h yields (5.6). Plugging the parameter values from Table 5.1 gives $T_s(P_A - P_S) + (E_U + E_D) > 0$, implying that the derivative is strictly negative. □

Corollary 4 (Hit-Rate Improvements Translate to Energy Savings)**.** *For any two placements* $\mathcal{H}_1, \mathcal{H}_2$ *with hit rates* $h_1 < h_2$*,*

$$E_{\mathrm{GreenDB}}(h_1) - E_{\mathrm{GreenDB}}(h_2) = N_q(h_2 - h_1)\big[T_s(P_A - P_S) + (E_U + E_D)\big] > 0.$$

Substituting the parameters from Table 5.1 and a representative service time of $T_s = 12\,\mathrm{s}$ yields an energy savings of approximately $393\,\mathrm{J}$ per query for each percentage-point increase in hit rate. This measure provides a useful quantitative link between the placement objective and deployment-level energy budgets. The resulting energy reductions are validated empirically in Section 5.3.3.

This decoupling idea delivers two concrete benefits in practice. First, GreenDB_DM can serve as a drop-in replacement for the placement component in the existing GreenDB system, with no modifications to the prefetching or power-management mechanisms. Second, it allows a direct comparison with the original GreenDB results: any observed energy savings can be attributed solely to improved table placement rather than changes to the underlying energy model. In Section 5.3.3, we leverage this property to isolate the impact of group-aware placement from that of the underlying architectural factors.

5.3 Experimental Evaluation

We evaluate GreenDB_DM along two complementary dimensions. Section 5.3.1 describes the experimental setup. Section 5.3.2 compares group-aware placement against four placement alternatives under varying workload characteristics, isolating the contribution of the placement decision itself. Section 5.3.3 evaluates the full GreenDB_DM system against four energy-management competitors from the prior work. Section 5.3.4 reports a sensitivity analysis of GreenDB_DM's mining parameters.

Table 5.1: Hardware energy parameters (adopted from [82]).

Parameter	Value
Active power P_A	58.0 W
Idle power P_I	27.7 W
Standby power P_S	2.5 W
Wake-up energy E_U	35 J
Sleep-down energy E_D	32 J
Wake-up time T_U	8.5 s
Sleep-down time T_D	3.3 s
Break-even time T_{BE}	13.3 s
Default cluster: 1 hot + 4 cold nodes	

5.3.1 Experimental Setup

Hardware energy parameters. We adopt the hardware parameters reported by Zhou *et al.* for the original GreenDB cluster [82], summarized in Table 5.1. These values reflect a representative commodity database server (Dell OptiPlex 3020 with Intel i5-4570, 4 GB RAM, 7200 RPM SATA disk, PostgreSQL 9.3.5) and have been measured directly with the physical power meters in the early study. Using the identical parameters makes our results comparable to those findings reported in the literature [82].

Workloads. Following the experimental philosophy of GreenDB [82], our primary evaluation make use of the synthetic workloads with controllable characteristics. This method allows systematic parameter sweeps that would be impossible on any single fixed real-world dataset. We parameterize workloads by: (i) the number of tables $|\mathcal{T}|$; (ii) the number of independent workflow groups G; (iii) the average group size $\bar{s}$; (iv) the Zipf skewness α controlling the popularity distribution across groups; (v) the group coherence $\rho \in [0, 1]$, the probability that a query stays within its primary group; and (vi) the hot-node capacity C. For each parameter setting, we generate 800 queries (500 training, 300 testing) using the procedure described in our companion synthetic workload generator. We also validate on the real-world Join Order Benchmark (JOB) [42], comprising 113 queries over 21 IMDb tables grouped into 33 query families.

Baselines. We compare against eight solutions spanning two categories:

Placement strategies (used in Section 5.3.2):

- **Random**: uniformly randomly selected C tables (a sanity-check lower bound).
- **Frequency**: top-C tables ranked by access count—the implicit placement strategy of GreenDB [82].
- **LRU-based**: tables retained by simulating an LRU cache of size C over the training queries.
- **Markov-cluster**: tables clustered using first-order query-to-query transition probabilities, selecting the largest cluster up to capacity.
- **GFP-Growth (ours)**: GreenDB_DM's group-aware placement (Algorithm 3).

Energy-management schemes (used in Section 5.3.3):

- **Non-E**: no energy management; all nodes always active.
- **DPM** [9]: per-node dynamic power management with fixed idle threshold.
- **PRE-BUD** [72]: disk-level prefetch buffering.
- **GreenDB** [82]: hot/cold node partitioning with frequency-based placement.
- **GreenDB_DM (ours)**: hot/cold node partitioning with group-aware placement.

Metrics. We report four metrics to quantitatively evaluate the analysis: (i) *full hit rate*, the fraction of queries fully served by the hot node; (ii) *partial hit rate*, the fraction with $\geq 70\%$ of accessed tables on the hot node; (iii) *cold-node wakeup rate*, the fraction of queries triggering at least one wakeup; and (iv) *total energy consumption* in kilojoules, computed via the GreenDB five-case energy model with the parameters of Table 5.1.

Implementation. All the tested algorithms were implemented in Python 3.10. The experiments were executed on a single Linux workstation; mining and placement decisions

are complete within one second per workload, so we report deterministic results obtained from fixed random seeds rather than running-time distributions.

5.3.2 Placement Strategy Evaluation

We first evaluate placement strategies in isolation, holding the underlying GreenDB cluster architecture fixed while varying only the algorithm that selects the hot-node table set. This approach isolates the contribution of group-aware placement from architectural factors.

Impact of Workload Skewness

Fig. 5.2 and Fig. 5.3 sweep the Zipf skewness α from 0.2 (dispersed access) to 1.8 (highly concentrated access), holding all other workload parameters fixed (50 tables, 8 workflow groups, hot capacity $C = 10$, group coherence $\rho = 0.9$).

GFP-Growth (purple) dominates the four strategies across the entire skewness range. Its full hit rate remains stable in the 14–20% range, peaking around $\alpha = 1.0$ where moderately concentrated access aligns well with mined group structure. In contrast, the Frequency baseline (blue) and the Markov-cluster baseline (red) both exhibit a characteristic "V-shape" collapse around $\alpha \in [0.8, 1.2]$, where full hit rate drops below 5%.This collapse occurs because, with moderately skewed access distributions, the top-C frequent tables are distributed across many partially touched workflow groups, leaving few queries fully covered.

The advantage of GFP-Growth over Frequency varies with α:

- At $\alpha = 0.2$ (dispersed): GFP-Growth 15.0% vs. Frequency 7.8% ($+7.2$ points).
- At $\alpha = 1.0$ (moderate): GFP-Growth 20.1% vs. Frequency 0.8% ($+19.3$ points).
- At $\alpha = 1.8$ (concentrated): GFP-Growth 14.3% vs. Frequency 5.2% ($+9.1$ points).

The partial-hit-rate panel ($\geq 70\%$ tables in hot) shows the same qualitative ranking, with GFP-Growth consistently among the top two strategies.

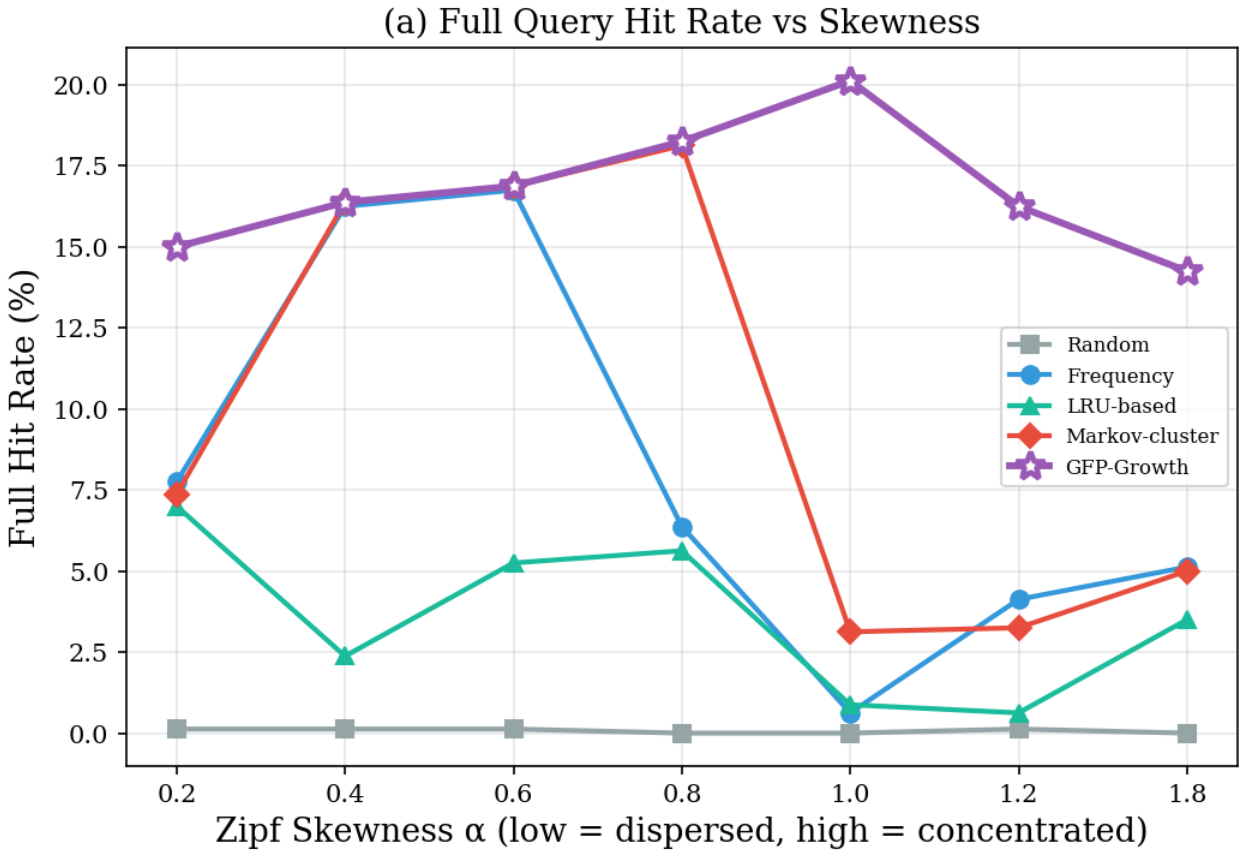

Figure 5.2: Impact of workload skewness on the full-query hit rate. GFP-Growth maintains the highest full-query hit rate across the entire skewness range, while frequency-based and Markov-cluster strategies experience substantial performance degradation around moderate skewness values. Configuration: 50 tables, 8 groups, $C = 10$, and $\rho = 0.9$.

This result is the central empirical finding of our work: group-aware placement provides robust hit-rate improvements across the entire range of workload skewness, with the largest gains observed precisely where simple frequency ranking is the weakest.

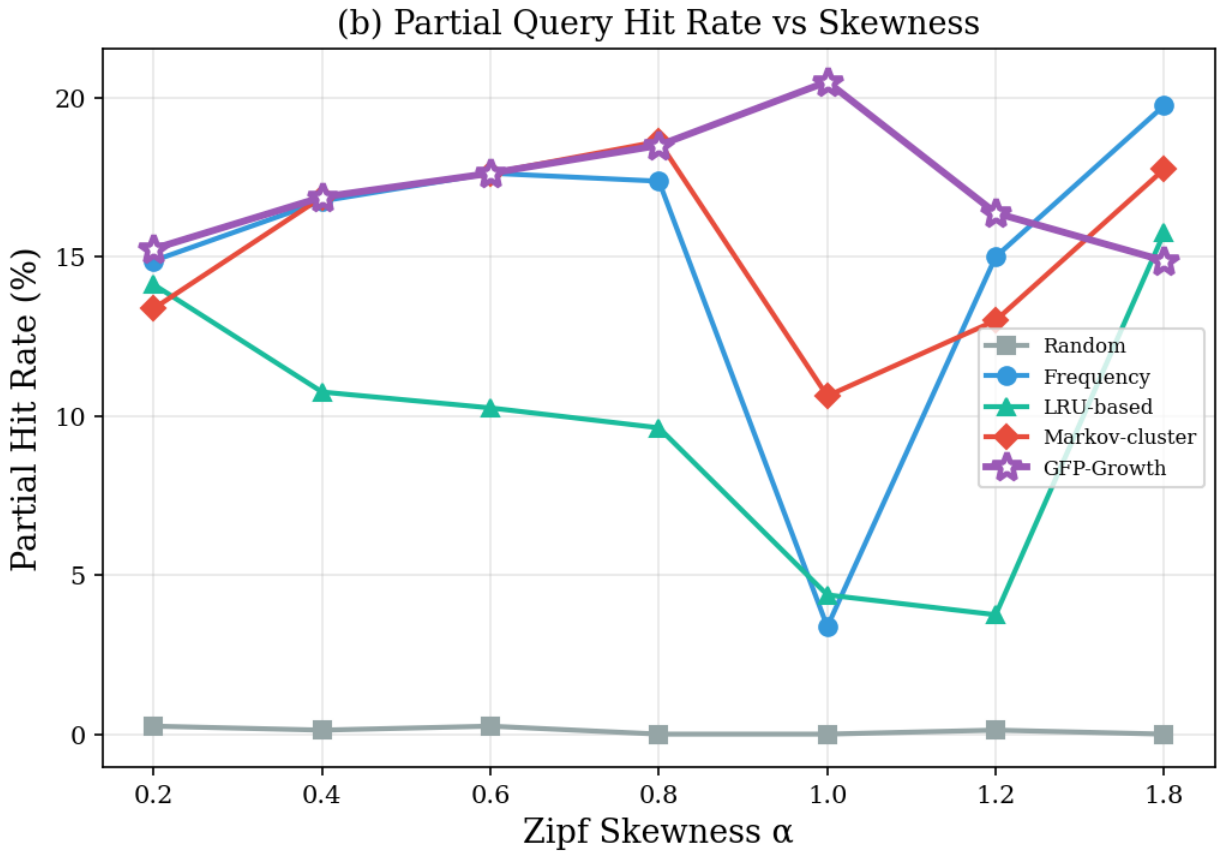

Figure 5.3: Impact of workload skewness on the partial-query hit rate, where a partial hit indicates that at least 70% of the queried tables are placed on the hot node. GFP-Growth achieves consistently high partial-query hit rates, whereas frequency-based, LRU-based, and Markov-cluster strategies exhibit greater sensitivity to changes in workload skewness. Configuration: 50 tables, 8 groups, $C = 10$, and $\rho = 0.9$.

Impact of Hot-Node Capacity

Fig. 5.4 and Fig. 5.5 plot the results where the hot-node capacity C is varied from 3 to 22 tables, holding workload parameters fixed at the moderate setting ($\alpha = 0.8$, $\rho = 0.9$, 50 tables, 8 groups).

GFP-Growth outperforms all the competitors at every capacity setting on full hit rate (panel a). The improvement over Frequency is the largest in the mid-capacity range: at $C = 10$, GFP-Growth achieves 22.0% full hit rate compared to 6.3% for Frequency—a 15.7-point advantage. At $C = 22$, GFP-Growth reaches 55.0% vs. 50.4% for Frequency, still leading by nearly 5 points. The wakeup-rate panel (b) tells the same story in reverse: GFP-Growth achieves the lowest cold-node wakeup rate at every capacity, dropping to 43% at $C = 22$ compared to 51% for Frequency—a 16% relative reduction in energy-costly wakeup events.

The capacity-sweep result is particularly important for deployment because it shows that the benefit of group-aware placement *does not* require precisely tuning the hot node to a specific capacity; GFP-Growth provides robust gains across the practical operating range.

5.3.3 Energy Efficiency Evaluation

Having established the placement-level advantage of GFP-Growth, we now evaluate the end-to-end energy consumption of the integrated GreenDB_DM system. All the experiments presented in this subsection use the GreenDB five-case energy model with the parameters of Table 5.1, and we report the additional energy savings GreenDB_DM achieves over the GreenDB counterpart. We focus on the two most relevant system parameters – data size and cluster scale –and report results across three hot-node capacities (tight, moderate, generous) to capture different deployment regimes.

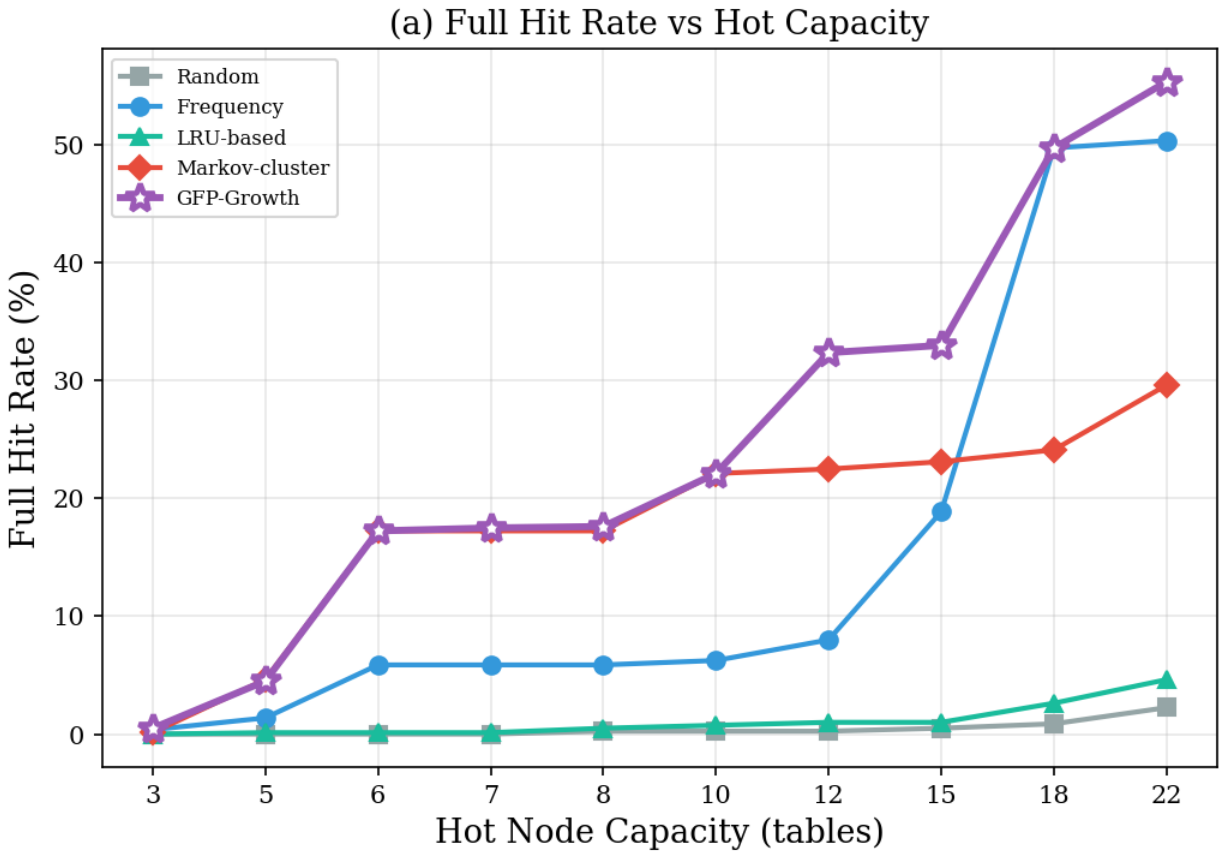

Figure 5.4: Impact of hot-node capacity on the full-query hit rate. GFP-Growth achieves the highest full-query hit rate across all capacity settings and provides the largest advantage over frequency-based placement in the mid-capacity range, where preserving complete table groups is especially important. Configuration: 50 tables, 8 groups, $\alpha = 0.8$, and $\rho = 0.9$.

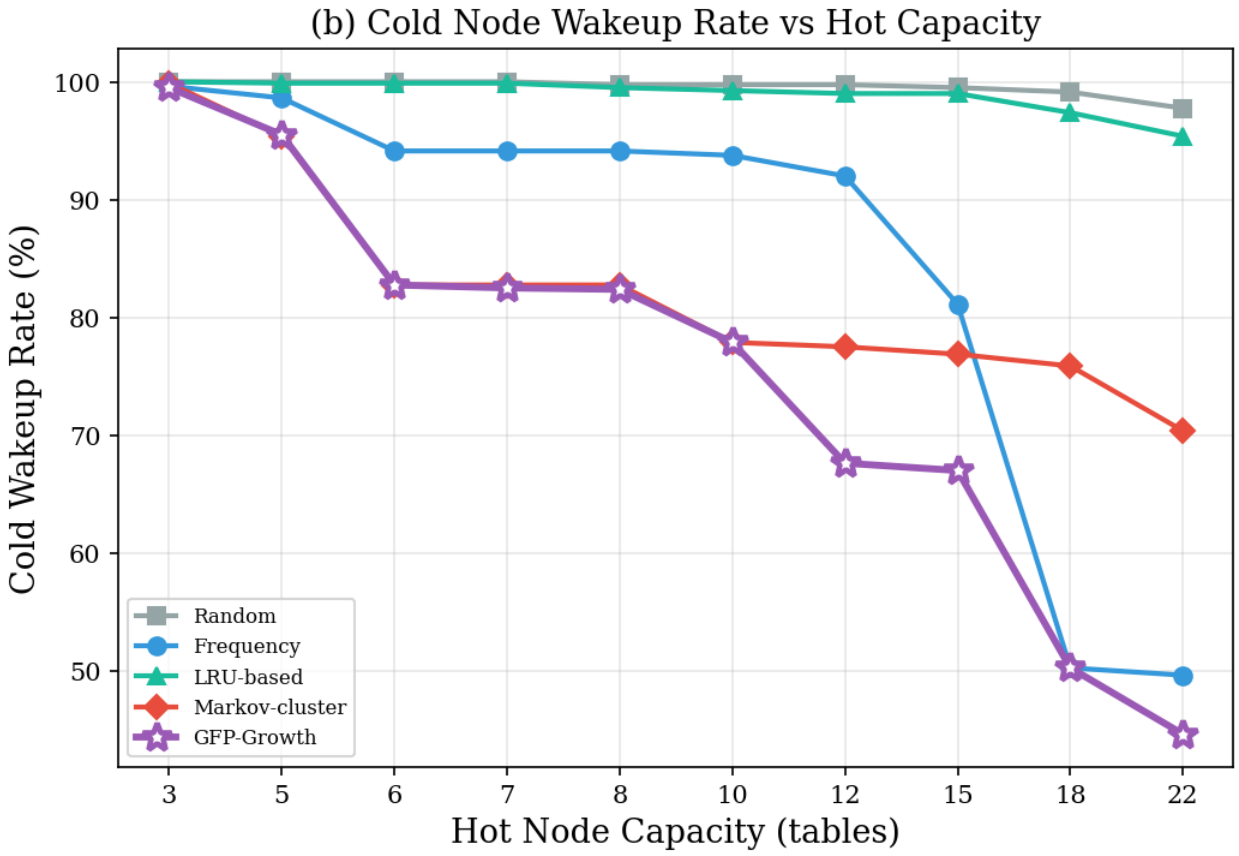

Figure 5.5: Impact of hot-node capacity on the cold-node wakeup rate. GFP-Growth consistently produces the lowest wakeup rate by placing frequently co-accessed table groups together on the hot node. Its advantage is most pronounced in the mid-capacity range, where frequency-based and LRU-based strategies still trigger substantially more cold-node wakeups. Configuration: 50 tables, 8 groups, $\alpha = 0.8$, and $\rho = 0.9$.

Impact of Data Size

Fig. 5.6, Fig. 5.7, and Fig. 5.8 plot the per-table data size from 200 K to 1 M records across three hot-node capacity regimes: *tight* ($C = 5$, panel a), *moderate* ($C = 7$, panel b), and *generous* ($C = 12$, panel c). In every regime and at every data size, GreenDB_DM consumes less total energy than baseline GreenDB, and the relative gap widens as more capacity becomes available for group-aware decisions to exploit:

- Tight ($C = 5$): GreenDB_DM saves 1.7% at 1 M records relative to GreenDB (376 kJ vs. 383 kJ).
- Moderate ($C = 7$): GreenDB_DM saves 4.3% at 1 M records (360 kJ vs. 376 kJ).
- Generous ($C = 12$): GreenDB_DM saves 4.7% at 1 M records (342 kJ vs. 359 kJ).

Total energy scales roughly linearly with data size for both schemes. The GreenDB_DM curve remains consistently below GreenDB, showing that the placement-level hit-rate improvement translates predictably into system-level energy savings.

Impact of Cluster Scale

Fig. 5.9, Fig. 5.10, and Fig. 5.11 reveal total energy as the cluster scales from 3 to 13 total nodes (1 hot + N cold), again across the three capacity regimes. This experiment follows the methodology of the original GreenDB evaluation [82] and tests whether the gains of group-aware placement persist as the cluster grows.

The shape of the results are consistent with those plotted in Fig. 5.6, Fig. 5.7, and Fig. 5.8 : at $C = 5$, GreenDB_DM saves only 1.5% at the largest cluster size; at $C = 7$, this grows to 3.9%; and at $C = 12$, GreenDB_DM saves 4.2% at 13 nodes (349 kJ vs. 364 kJ). Total energy grows linearly with node count for both schemes (as expected, since each cold node adds a near-constant standby cost), but the relative ranking and the absolute gap remain stable across the entire range.

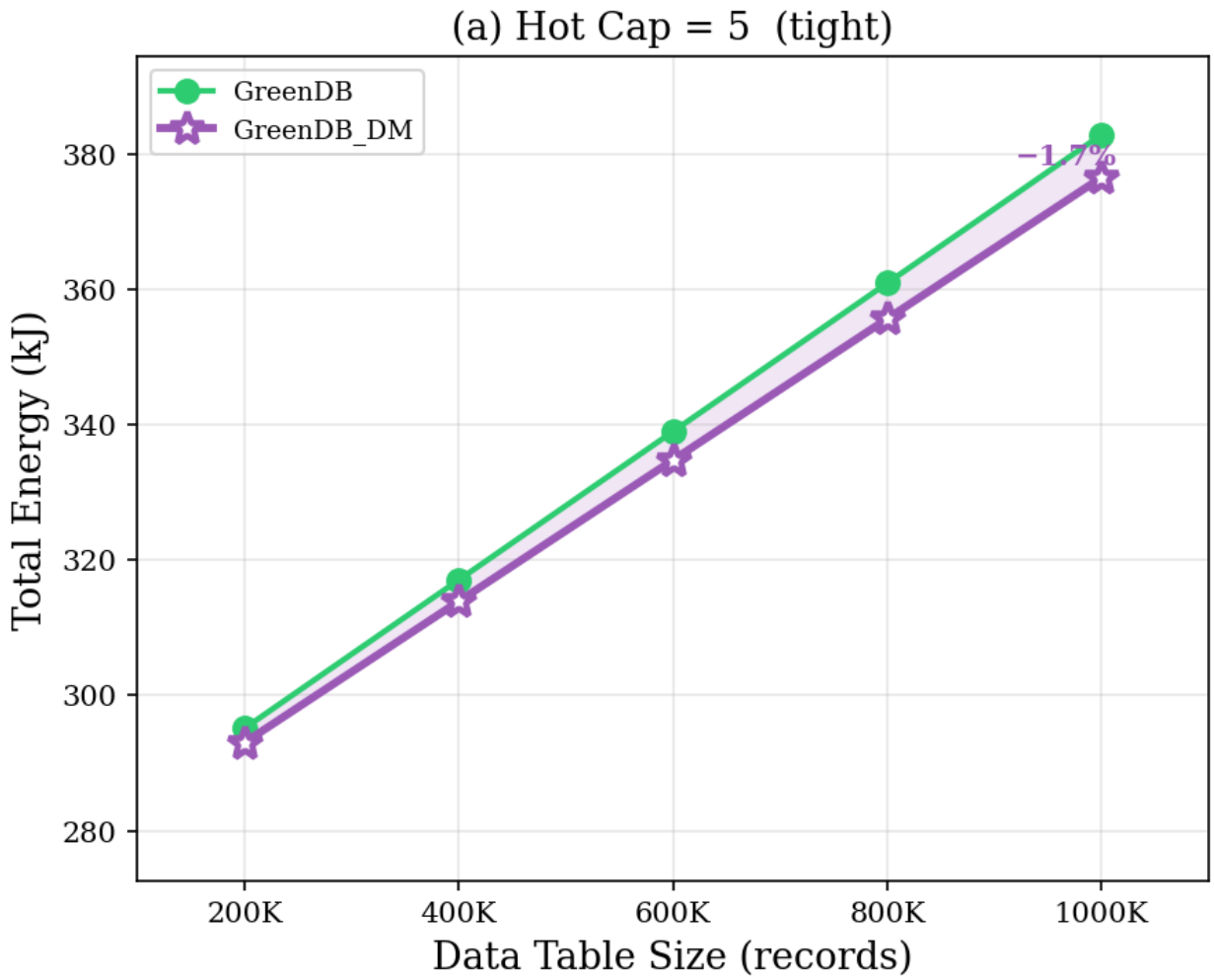

Figure 5.6: Impact of data table size on total cluster energy under a tight hot-node capacity of $C = 5$. Total energy increases approximately linearly with data size for both GreenDB and GreenDB_DM. GreenDB_DM consistently consumes less energy, achieving a 1.7% reduction at one million records.

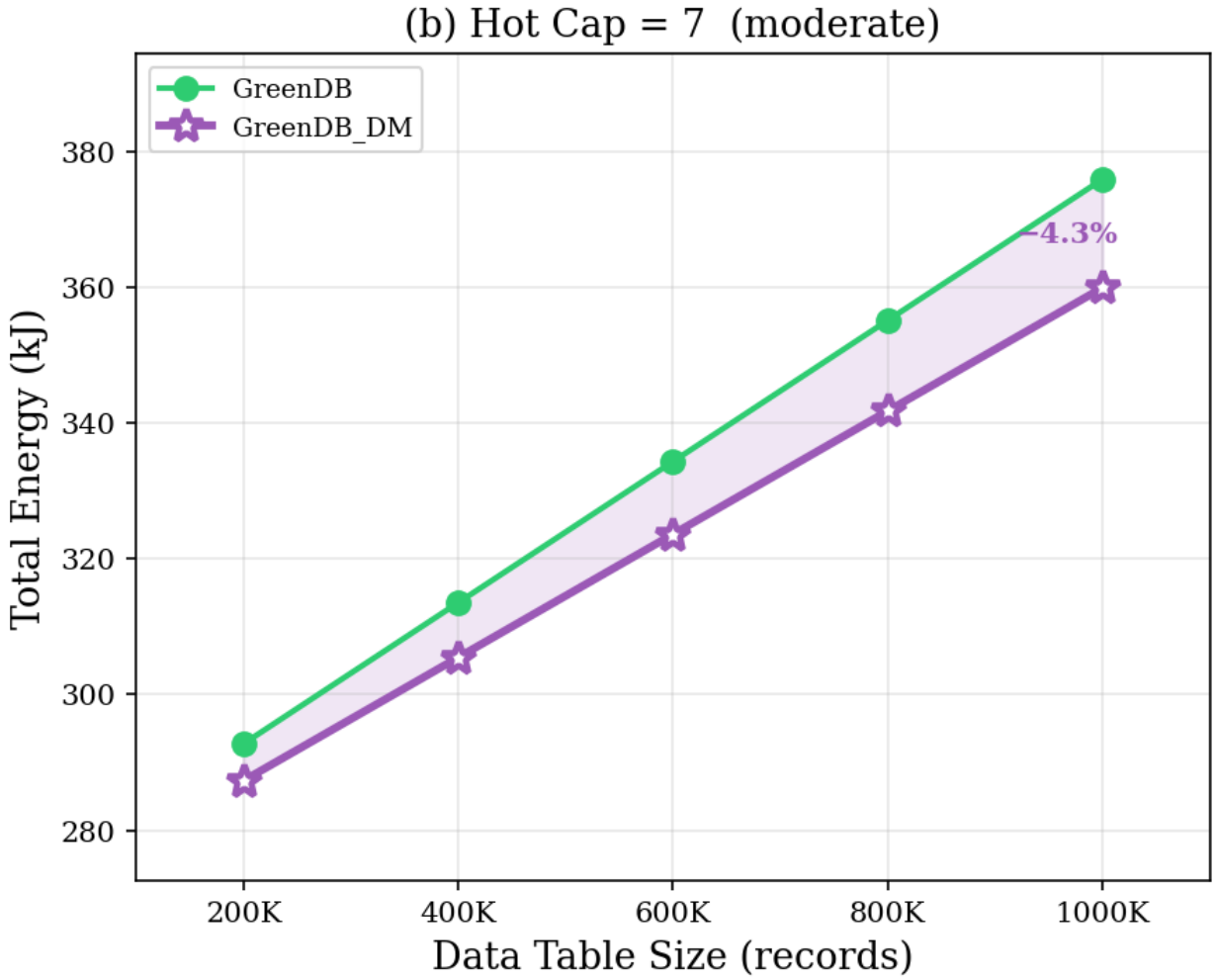

Figure 5.7: Impact of data table size on total cluster energy under a moderate hot-node capacity of $C = 7$. GreenDB_DM remains consistently below GreenDB across all data sizes, and the energy-saving gap becomes more pronounced as the workload grows. At one million records, GreenDB_DM reduces total energy consumption by 4.3%.

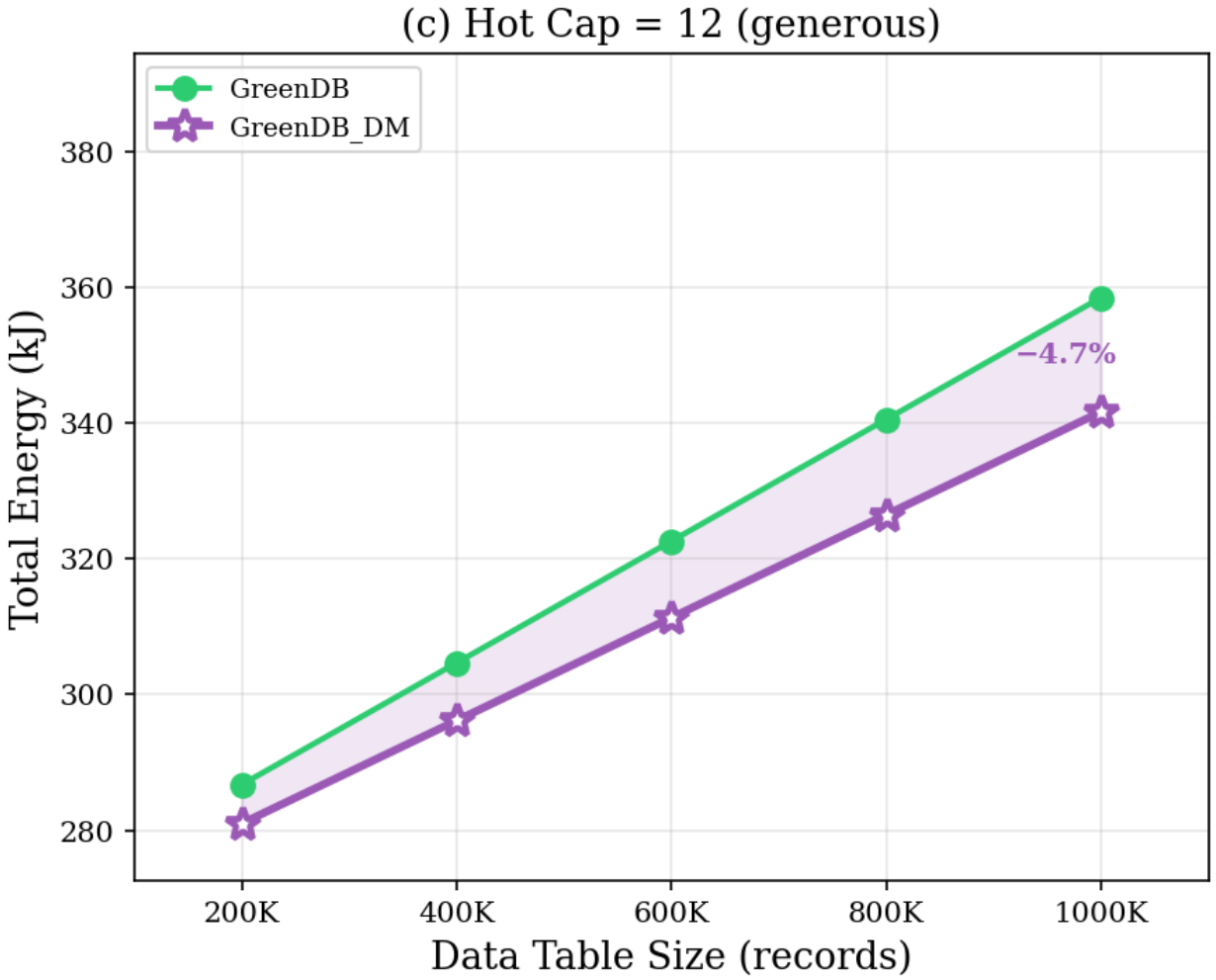

Figure 5.8: Impact of data table size on total cluster energy under a generous hot-node capacity of $C = 12$. The larger hot-node capacity allows GreenDB_DM to preserve more complete table groups, resulting in consistently lower energy consumption than GreenDB. The maximum reduction reaches 4.7% at one million records.

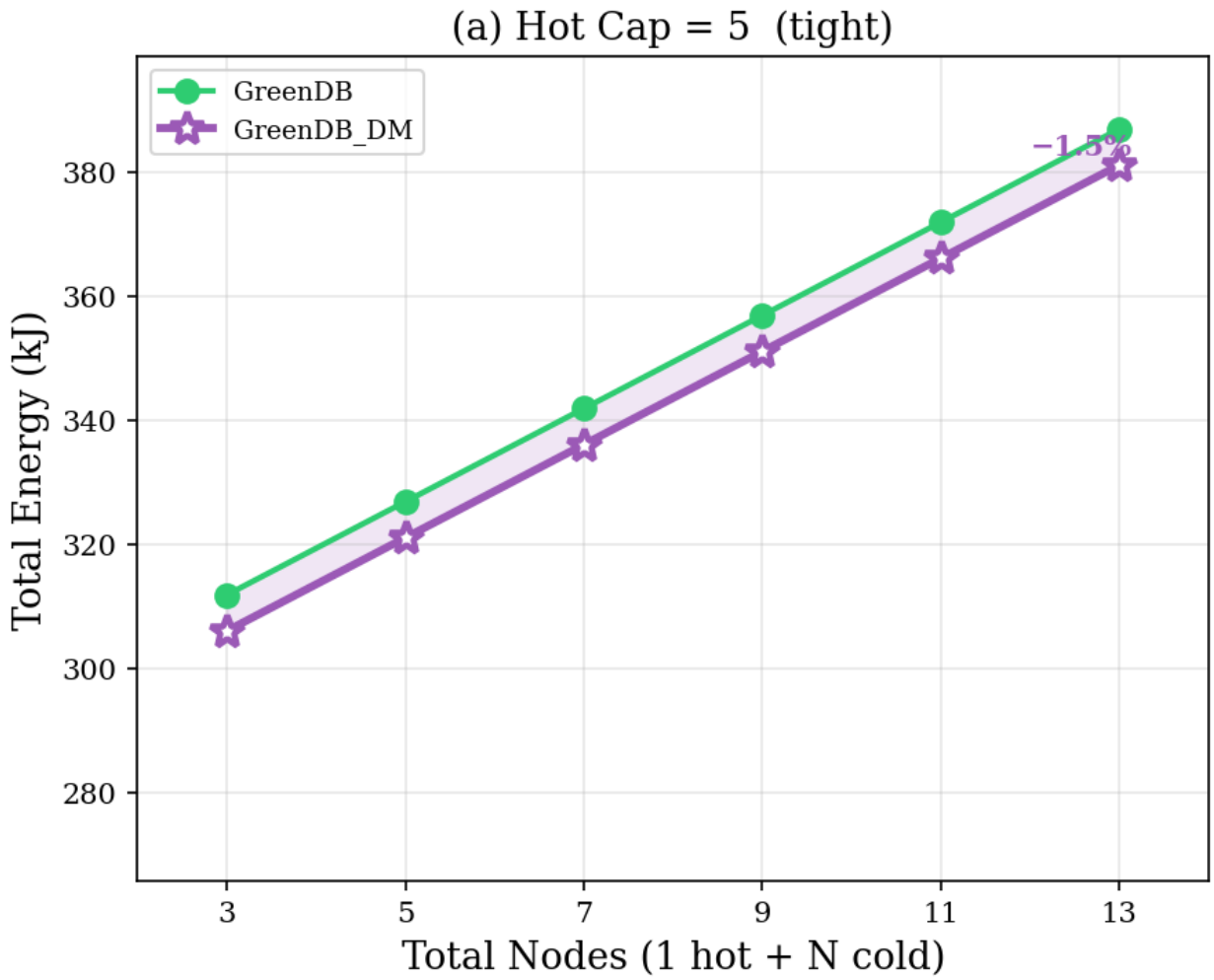

Figure 5.9: Impact of cluster size on total cluster energy under a tight hot-node capacity of $C = 5$. Energy consumption increases approximately linearly as additional cold nodes are introduced. GreenDB_DM consistently consumes less energy than GreenDB, achieving a 1.5% reduction at 13 total nodes.

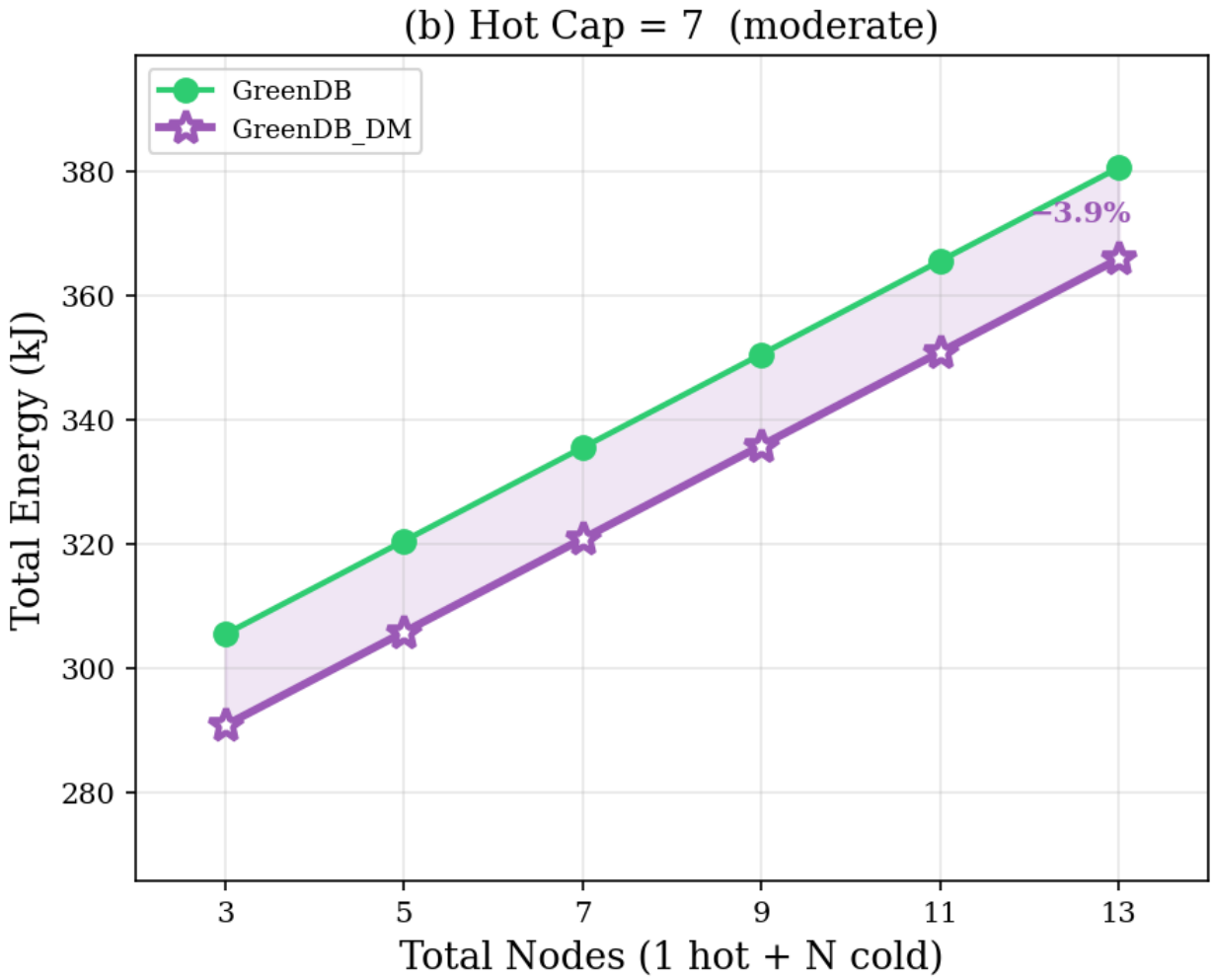

Figure 5.10: Impact of cluster size on total cluster energy under a moderate hot-node capacity of $C = 7$. GreenDB_DM maintains a stable energy advantage over GreenDB as the cluster scales, reducing total energy consumption by 3.9% at 13 total nodes.

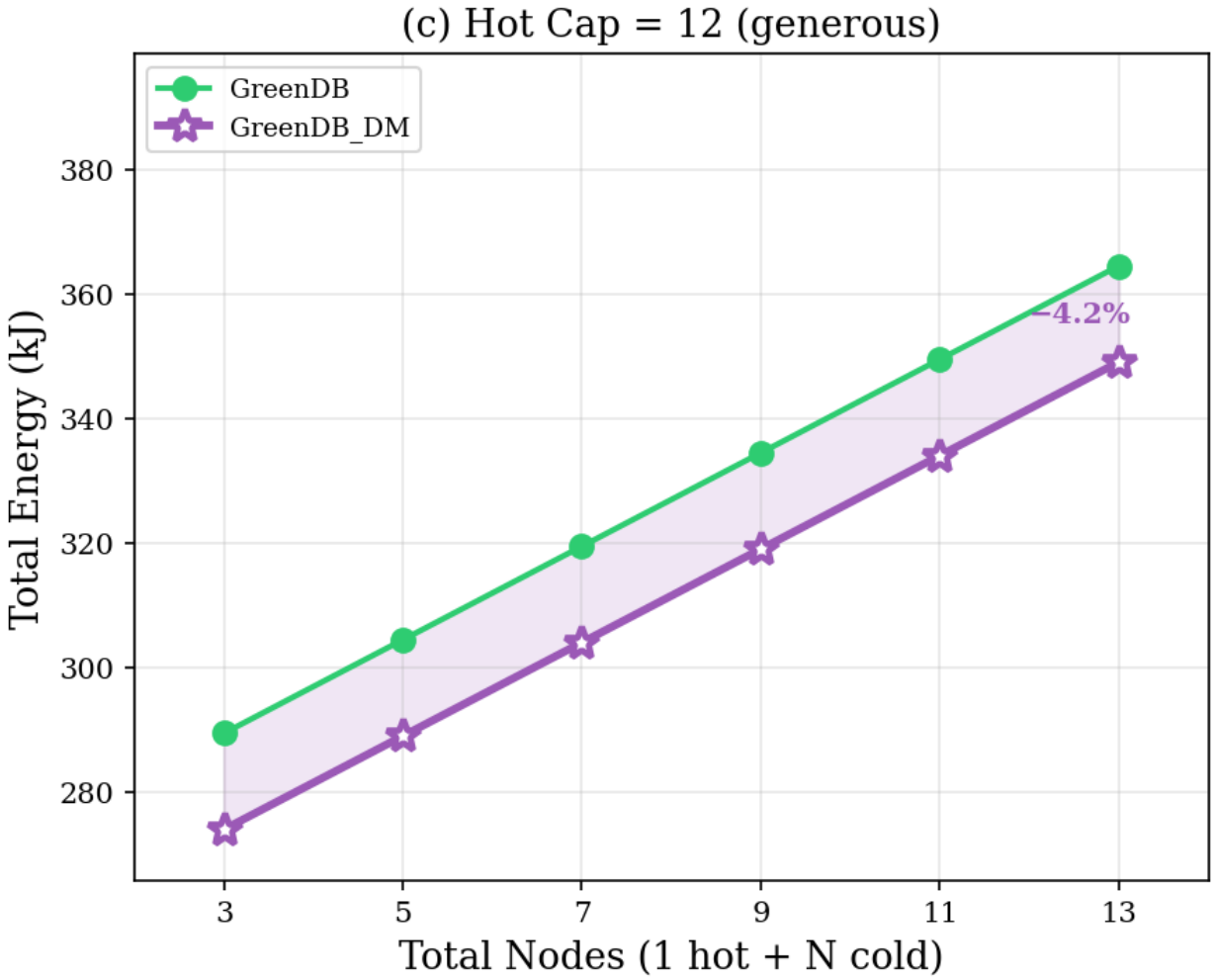

Figure 5.11: Impact of cluster size on total cluster energy under a generous hot-node capacity of $C = 12$. The larger hot-node capacity enables GreenDB_DM to retain more complete table groups and sustain a wider energy gap over GreenDB. At 13 total nodes, GreenDB_DM achieves the largest relative energy reduction of 4.2%.

Validation on JOB

To examine whether the energy benefits of GreenDB_DM generalize beyond the workload used in the preceding experiments, we conduct an additional evaluation using the Join Order Benchmark (JOB). JOB consists of complex, multi-table join queries derived from the IMDb database and exhibits table co-access patterns that differ substantially from those of the primary workload. We use a hot-node capacity of $C = 15$, for which GreenDB achieves a full-query hit rate of 49.6%, whereas GreenDB_DM increases the hit rate to 64.6%, representing an absolute improvement of 15.0 percentage points.

Fig. 5.12 shows the impact of per-table data size on total cluster energy with one hot node and ten cold nodes. As the data size increases from 200 K to 1 M records per table, the total energy consumption of both systems increases approximately linearly. Nevertheless, GreenDB_DM consistently remains below GreenDB across the entire range. At one million records, GreenDB_DM consumes approximately 297 kJ, compared with approximately 317 kJ for GreenDB, corresponding to an energy reduction of 6.4%.

Fig. 5.13 evaluates the effect of increasing the cluster size from 3 to 13 total nodes while fixing the data size at 500 K records per table. Total energy again increases approximately linearly because each additional cold node introduces a near-constant standby energy cost. GreenDB_DM maintains a stable energy advantage throughout the experiment. At 13 total nodes, it reduces total energy consumption by 5.7%, from approximately 327 kJ to 308 kJ.

These results demonstrate that the advantages of group-aware table placement are not limited to the primary experimental workload. The substantial hit-rate improvement and the consistent energy reductions on JOB confirm that GreenDB_DM remains effective for complex, join-intensive SQL workloads with different table-access distributions.

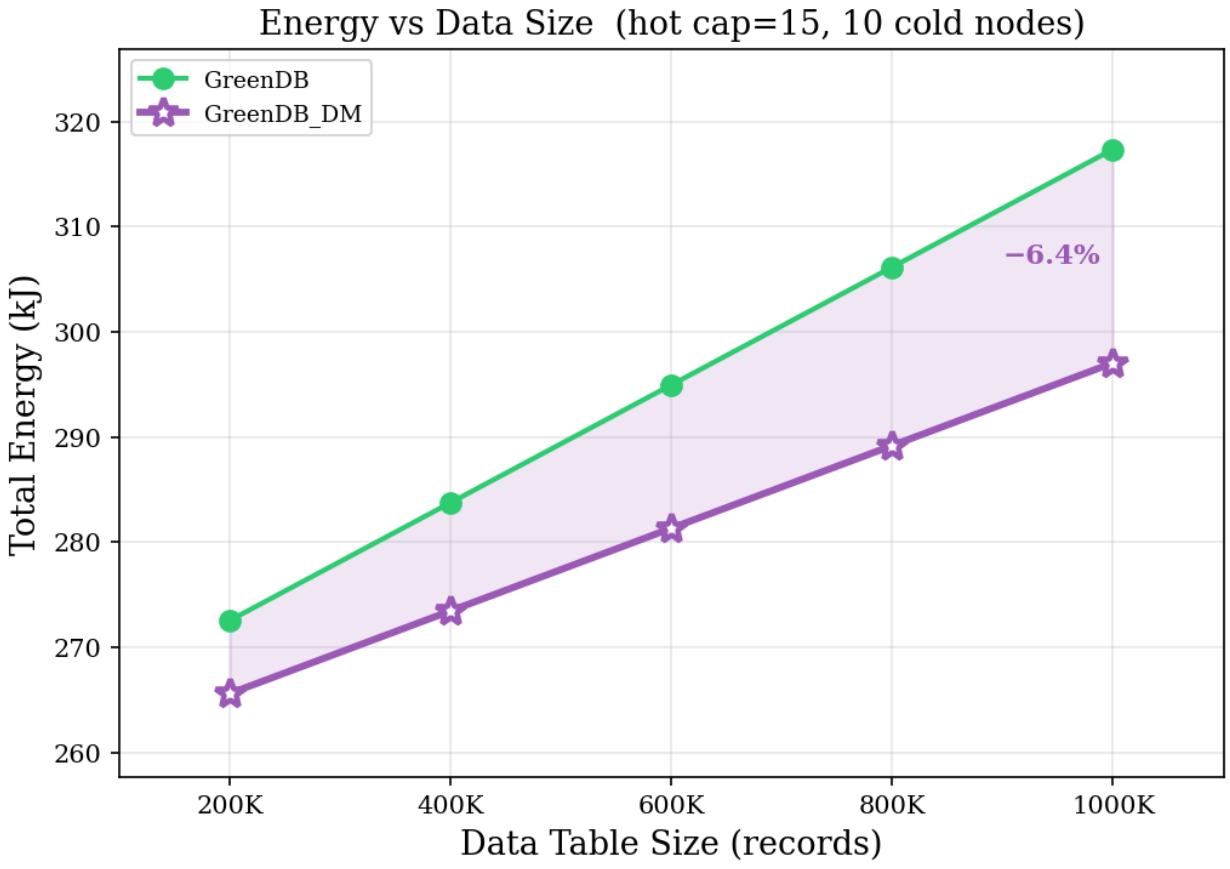

Figure 5.12: Total cluster energy versus data table size for the JOB workload with $C = 15$ and ten cold nodes. GreenDB_DM achieves a 6.4% energy reduction at one million records per table.

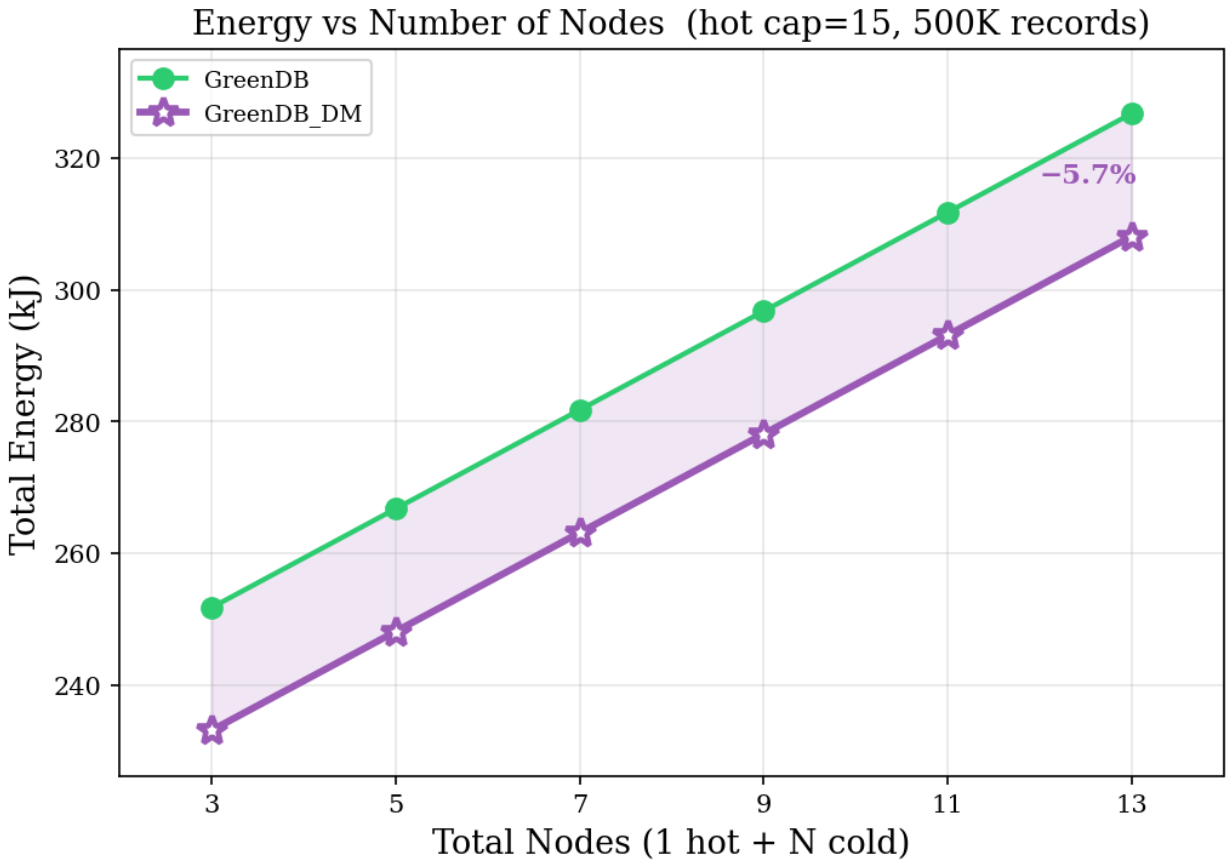

Figure 5.13: Total cluster energy versus cluster size for the JOB workload with $C = 15$ and 500 K records per table. GreenDB_DM achieves a 5.7% energy reduction at thirteen nodes.

Comprehensive Comparison Across Workload Regimes

Fig. 5.14 and Fig. 5.15 provides the headline end-to-end comparison: total energy under three workload regimes for all the five schemes (panel a), and the additional energy reduction of GreenDB_DM over the strongest existing GreenDB system (panel b).

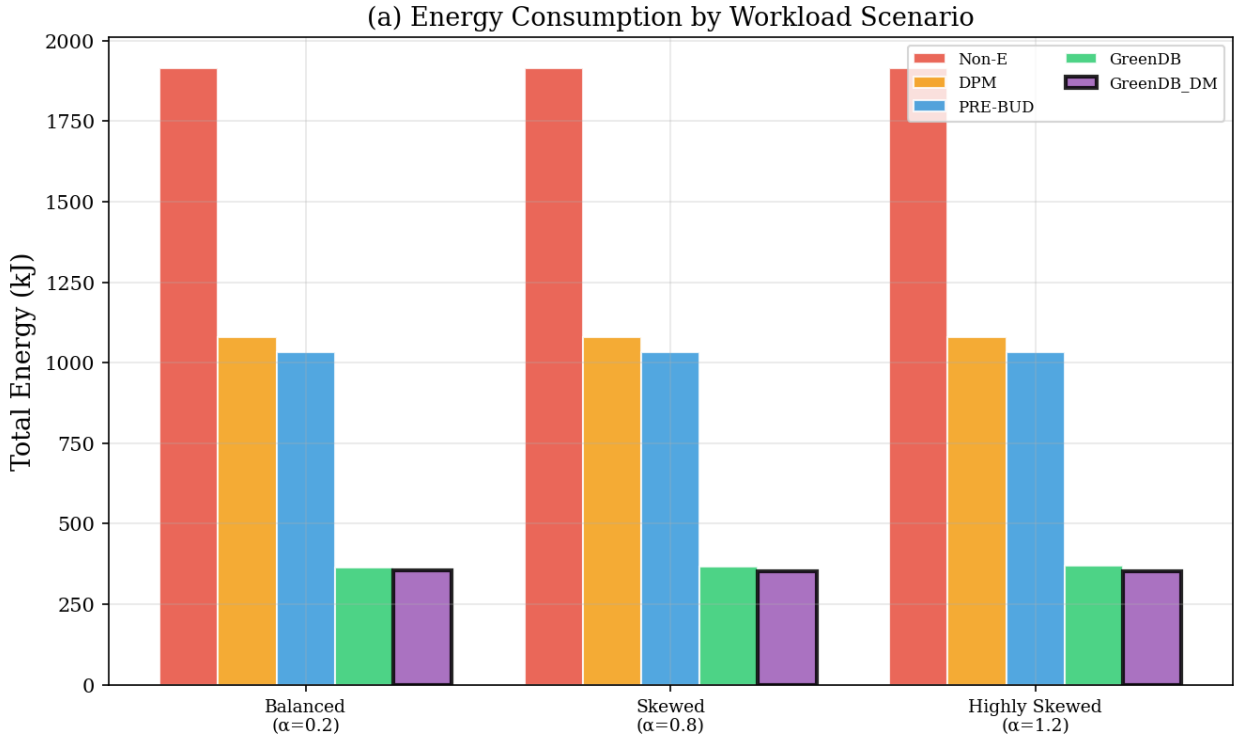

Figure 5.14: Total cluster energy consumption across three workload regimes. GreenDB and GreenDB_DM achieve substantially lower energy consumption than Non-E, DPM, and PRE-BUD under balanced ($\alpha = 0.2$), skewed ($\alpha = 0.8$), and highly skewed ($\alpha = 1.2$) workloads. GreenDB_DM consistently provides the lowest total energy by exploiting group-aware table placement.

Panel (a) shows that all the hot/cold-partitioned schemes (GreenDB, GreenDB_DM) achieve massive energy reductions over the Non-E baseline (which keeps all nodes active), DPM, and PRE-BUD across every workload regime. Within this top tier, GreenDB_DM consistently consumes less energy than GreenDB. Panel (b) quantifies the *additional* reduction of GreenDB_DM over GreenDB:

- **Balanced** ($\alpha = 0.2$): $\mathbf{2.4\%}$ additional reduction.

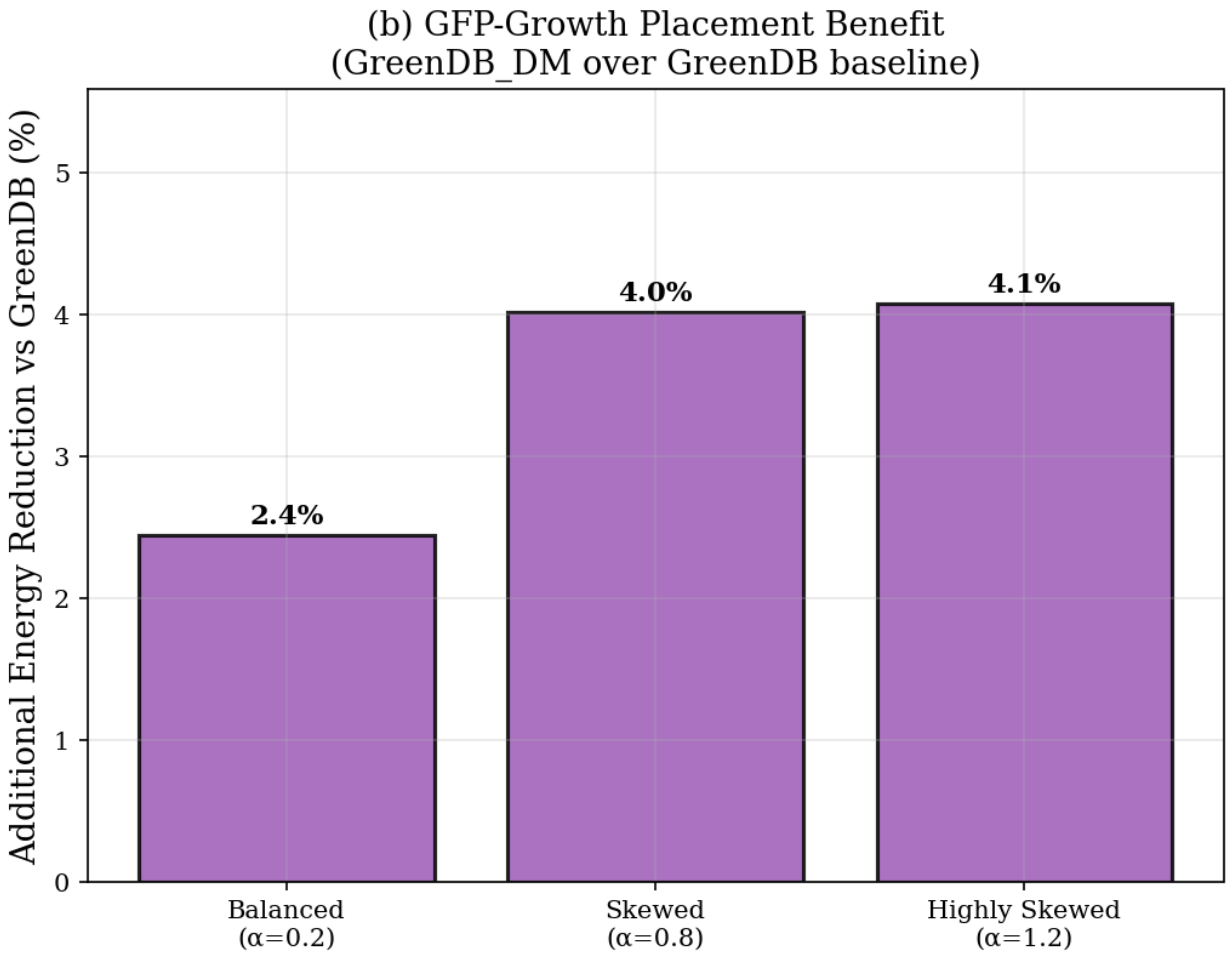

Figure 5.15: Additional energy reduction achieved by GreenDB_DM relative to the GreenDB baseline across three workload regimes. GreenDB_DM reduces energy consumption by 2.4% under the balanced workload, 4.0% under the skewed workload, and 4.1% under the highly skewed workload. The larger improvement under skewed workloads demonstrates that group-aware placement is most effective when table accesses exhibit stronger co-access structure.

- **Skewed** ($\alpha = 0.8$): $\mathbf{4.0}\%$ additional reduction.
- **Highly skewed** ($\alpha = 1.2$): $\mathbf{4.1}\%$ additional reduction.

The largest gains appear precisely under the skewed and highly skewed workloads, where Frequency placement is fragmented across many partially-needed groups, and group-aware placement most effectively recovers full-coverage opportunities. With the balanced workloads, where accesses are nearly uniform, there is less group structure available to exploit, resulting in smaller, though still positive, gains.

5.3.4 Mining Cost and Practical Overhead

The GFP-Growth mining stage runs offline against historical query logs. Across all workloads in our evaluation (up to $10{,}000$ queries, 60 tables), mining completes in under 200 ms on a single CPU core, with typical runtimes in the range of 5–50 ms. The set-cover placement step completes in under 10 ms. Together, these costs are negligible compared to the placement refresh interval, which we envision as daily or weekly in practice. Mining is therefore trivially amortized over the placement period it informs, and re-mining can be triggered as often as workload drift demands.

5.3.5 Comparison Summary

The experimental evaluation supports three central claims below.

1. *Robust placement gains.* GFP-Growth raises full-query hit rate by 5–15 percentage points over the strongest placement baseline across the entire range of workload skewness (Fig. 5.2 and Fig. 5.3), and dominates all the alternative schemes at every hot-node capacity (Fig. 5.4 and Fig. 5.5).
2. *Predictable energy translation.* Hit-rate improvements translate predictably into total energy savings, with GreenDB_DM saving an additional 2.4–4.7% beyond the

energy-effective GreenDB across data sizes, cluster scales, and workload regimes (Fig. 5.6, Fig. 5.7, Fig. 5.8, Fig. 5.9, Fig. 5.10, Fig. 5.11, Fig. 5.14 and Fig. 5.15).

3. *Negligible overhead.* Mining and placement are completed in under 200 ms, even with the largest workload sizes, making GreenDB_DM a seamless substitute for the placement module of any existing GreenDB deployment.

5.4 Discussion

Having presented our experimental evaluation, we now reflect on the applicability and limitations of GreenDB_DM and discuss several practical considerations for deployment.

5.4.1 When Group-Aware Placement Helps

The experimental results in Section 5.3 reveal a clear pattern: GreenDB_DM provides the significant improvements when the workload is *dispersed* – when no small set of tables dominates access counts but the workload nonetheless decomposes into coherent groups. We characterize this regime by two empirically measurable workload properties.

- *Low Zipf skewness* ($\alpha < 0.5$), indicating that no small set of tables dominates the access distribution.
- *High group coherence* ($\rho > 0.7$), indicating that queries access tables in coherent groups rather than uniformly across the schema.

These conditions arise naturally in two practical classes of deployments:

- *Multi-tenant SaaS* systems, where each tenant accesses its own – often disjoint – set of tables and no single tenant dominates global access counts.
- *Multi-workflow* analytical systems (ERP, BI dashboards, and scientific computing pipelines), where several business processes run concurrently and each touches a distinct, internally coherent group of tables.

Conversely, on highly concentrated workloads such as JOB, GreenDB_DM converges with frequency-based placement and provides no incremental benefit. The frequency tie-breaker in Algorithm 3 ensures that GreenDB_DM never performs worse than GreenDB in this regime, so deployment is safe even when the workload distribution is unknown a priori.

5.4.2 Practical Considerations for Deployment

Mining cost. GFP-Growth runs offline against historical query logs and completes in well under one minute even on workloads of tens of thousands of queries (see Section 5.3.4). Mining is therefore trivially amortized over the placement period it informs; in our intended use case, mining is re-executed periodically (e.g., daily or weekly) to refresh placement decisions.The step itself completes in milliseconds and adds no noticeable overhead, even if re-executed more frequently.

Workload drift. Production workloads change over time as new applications are added, schema changes are implemented, or seasonal access patterns shift. GreenDB_DM handles drift through periodic re-mining: by replaying GFP-Growth and Algorithm 3 on a sliding window of recent queries, the placement adapts to the current workload. The migration cost of moving tables between hot and cold nodes is governed by the underlying GreenDB infrastructure and is not exacerbated by our mining-based placement, since the set of tables on the hot node typically changes incrementally between re-mining cycles. A more aggressive online variant of GreenDB_DM that updates placement continuously is left as future work.

Cold-start scenarios. When a database is freshly deployed without any existing query log, GreenDB_DM is unable to mine patterns and must fall back to a frequency-based scheme like GreenDB or to schema-driven heuristics based on declared foreign-key relationships. After a brief warm-up period (e.g., a few hours of typical query traffic),

enough log data accumulates for GFP-Growth to produce informative groups. Bootstrapping from foreign-key graphs as a structural prior, before sufficient query data becomes available, is a promising direction for future work.

Compatibility with existing GreenDB deployments. By design, GreenDB_DM modifies only the placement decision and reuses the unmodified GreenDB energy model and prefetching infrastructure (Section 5.2.5). The existing GreenDB system can readily adopt GreenDB_DM by replacing their frequency-based placement module with our mining-driven one, without disturbing operational components.

5.4.3 Limitations and Threats to Validity

Single-node hot tier. Following the GreenDB architecture, we consider a single always-active hot node. Extending GreenDB_DM to multiple hot nodes (e.g., one per workflow group) is conceptually straightforward – each hot node would receive one or more groups – but introduces new coordination questions, such as inter-hot-node load balancing and cross-group join handling, which will be investigated in our future work.

Static placement granularity. Our placement operates at the granularity of entire tables, matching GreenDB. Finer-grained placements such as partitions or column groups are an interesting extension that may yield additional savings on workloads where individual tables are too large to fit on the hot node, but it requires re-deriving the energy model.

Pattern mining choice. We adopt GFP-Growth from the GARMT framework [30] as our pattern mining backend. Other pattern-discovery techniques— sequence mining, graph clustering on foreign-key graphs, or learned embedding methods—may produce different groups and warrant comparison. We expect the *set-cover-based placement* component (see Algorithm 3) to remain valid regardless of the upstream pattern source, so this question is largely orthogonal to our main contribution.

5.5 Summary

We have presented GreenDB_DM, a data-mining-driven extension of the GreenDB framework that brings group-aware table placement to energy-efficient database clusters. By mining frequent table co-access patterns from historical query logs with GFP-Growth and by casting hot-node placement as a set-cover problem, GreenDB_DM directly maximizes the metric that drives the GreenDB energy model—the full-query hit rate—rather than the proxy metric of individual table frequency. Our extensive evaluation on various workloads spanning a wide range of skewness, group coherence, cluster scale, and data size shows that GreenDB_DM raises full-query hit rate by 5–15 percentage points over the strongest placement baseline, and lowers total cluster energy by an additional 2.4–4.7% beyond the strongest energy-management baseline GreenDB—with the largest gains under moderately to highly skewed workloads where group-aware placement most effectively exploits multi-table co-access structure. Equally important, GreenDB_DM gracefully degrades to the behavior of frequency-based placement on workloads where group structure is absent, making it a safe drop-in replacement for the placement module in the existing GreenDB system.

A handful of directions remain open for future work. First, extending GreenDB_DM to support online, continuously-updated placement would allow the system to adapt to workload drift on shorter time scales than the offline mining cycle considered in this study. Second, generalizing the method to multi-hot-node clusters and finer-grained placement units (partitions or column groups) is expected to unlock additional energy savings on large databases. Third, integrating schema-aware structural priors—such as foreign-key graphs—will enable bootstrapping in cold-start scenarios where query logs have not yet accumulated. These directions, together with broader implementations on production workloads, will further establish data mining as a foundational building block for energy-efficient database management systems.

Chapter 6

Conclusion

6.1 Summary of Contributions

In this dissertation, we proposed GreenDB_DM, an energy-aware SQL database system that integrates SQL workload mining, dynamic data-driven modeling, and group-aware table placement to improve the efficiency of modern database systems. The central objective of this dissertation was to understand how SQL query workloads can be analyzed, how hidden access patterns can be discovered from sparse query logs, and how these patterns can be used to support energy-efficient database management. As database systems continue to process increasingly large and complex workloads, reducing unnecessary computation, improving table access prediction, and minimizing cold-node wake-ups have become important goals for both performance optimization and energy-aware system design.

The first major component of this dissertation focused on grouping-based association rule mining for SQL workload analysis. In large SQL databases, individual queries often access only a small fraction of the available tables and fields, creating a sparse transaction environment. This sparsity makes it difficult to discover meaningful table access patterns when SQL statements are processed in isolation. To address this limitation, we proposed a grouping-based strategy that aggregates related SQL statements before applying frequent-pattern mining. By treating multiple SQL operations within a time window or logical context as a whole, the proposed method increases transaction density and enables more effective discovery of frequent table access patterns.

This grouping-based mining approach provides the foundation for the energy-aware SQL database system developed in this dissertation. By identifying tables that are frequently accessed together, the system can support more informed decisions for query optimization, caching, prefetching, and table placement. The experimental results showed that grouping can reduce mining runtime while preserving rule relevance, demonstrating the practical value of query grouping in sparse SQL database environments. These results indicate that workload-aware data mining can improve SQL workload analysis and provide useful information for downstream system-level optimization.

The second component of this dissertation examined dynamic data-driven modeling through a complementary study of recommender systems using disease spread techniques. Although this component is not the main focus of the dissertation, it supports the broader theme that data-driven systems should be analyzed not only through static metrics, but also through dynamic behavior over time. By modeling recommendation diffusion from an epidemiological perspective, this study showed how user interaction, network structure, and time-varying engagement can influence propagation outcomes. This component provides an additional perspective on dynamic behavior modeling while the main technical direction remains centered on SQL workload mining and energy-aware database optimization.

6.2 Implications for Energy-Aware SQL Database Systems

The final and most important system-level component of this dissertation was the development of GreenDB_DM, a data-mining-driven extension of the GreenDB framework. Existing energy-efficient database systems reduce energy consumption through disk-level or node-level power management, but they often rely on frequency-based hot-data placement. Such placement strategies rank tables by individual access frequency and place the most frequently accessed tables on the hot node. However, this approach overlooks the

fact that real SQL queries rarely access tables in isolation. Instead, queries often access groups of semantically related tables that jointly support specific workflows.

To address this limitation, GreenDB_DM reframes hot-node placement from a per-table frequency-ranking problem into a group-aware coverage problem. The system uses grouped frequent-pattern mining to discover frequently co-accessed table groups from historical SQL query logs and then applies a set-cover-based placement strategy to maximize full-query hit rate. By placing complete table groups on the hot node, GreenDB_DM increases the likelihood that an entire query can be served without waking cold nodes. This design directly connects workload mining with energy-aware placement and allows the system to reduce unnecessary cold-node transitions.

The proposed GreenDB_DM framework integrates the discovered co-access patterns with the GreenDB energy model without changing the underlying energy accounting framework. This makes the approach practical because it improves the placement component while preserving the original hot-node and cold-node architecture. The experimental results showed that group-aware placement can improve full-query hit rate and reduce total cluster energy compared with frequency-based placement and existing energy-management baselines. The largest benefits appear in workloads with meaningful group structure, such as multi-workflow analytical workloads and multi-tenant database environments.

Overall, this dissertation demonstrates that an energy-aware SQL database system should not only manage power states, but also understand workload structure. The grouping-based mining method discovers useful access patterns from sparse SQL logs; the dynamic modeling study reinforces the importance of time-sensitive and behavior-aware analysis; and GreenDB_DM applies mined workload patterns to reduce cold-node wakeups and improve cluster-level energy efficiency. Together, these contributions show that data mining can serve as a bridge between workload analysis and energy-aware system optimization.

6.3 Closing Remarks

In conclusion, this dissertation presents a unified approach for building an energy-aware SQL database system through workload-aware data mining and group-aware placement. By grouping SQL queries, mining frequent table access patterns, and using these patterns to guide hot-node placement, the proposed system improves the understanding of database workload behavior and reduces unnecessary energy consumption.

The dissertation shows that energy efficiency in database systems can be achieved not only through hardware-level power management, but also through intelligent analysis of how data is accessed. This work provides a foundation for designing SQL database systems that are more efficient, adaptive, and aware of the relationship between workload patterns and energy consumption.

Chapter 7

Future Work

In previous chapters, we proposed GreenDB_DM, an energy-aware SQL database system that integrates SQL workload mining, grouped frequent-pattern discovery, and group-aware hot-node placement to reduce unnecessary cold-node wakeups and improve cluster-level energy efficiency. This dissertation demonstrated that SQL query workloads contain meaningful co-access structures, and that these structures can be mined and used to guide more intelligent caching, prefetching, and table placement decisions. Although the proposed system provides an effective step toward workload-aware and energy-efficient database management, there are several avenues for future research and improvement.

In future research, key areas for development include AI-driven table prediction for intelligent databases, intelligent access forecasting in non-relational databases, and AI-enhanced energy optimization for data systems. These directions extend the central idea of this dissertation: database systems should not only react to incoming queries, but also learn from historical workloads, predict future access patterns, and proactively manage resources to reduce energy consumption while maintaining stable performance.

First, future work can develop AI-driven table prediction models for intelligent SQL database systems. In this dissertation, GreenDB_DM uses grouped frequent-pattern mining to discover frequently co-accessed table groups from historical SQL workloads. This approach is effective for identifying stable and recurring table access patterns. However, future SQL workloads may contain more complex temporal, sequential, and structural dependencies that cannot be fully captured by traditional association rule mining alone. To address this challenge, future work can build AI models that learn from SQL workloads and predict future table access patterns.

Sequence learning methods, such as Long Short-Term Memory (LSTM) networks and Transformer-based models, can be used to capture the temporal order of SQL queries, query-to-query dependencies, and long-range access relationships among tables. Compared with static frequent-pattern mining, these models may better understand how table access evolves over time and how earlier queries influence later queries in the same session or workload stream. The predicted table access patterns can then be integrated into prefetching and caching modules for SQL database systems. Instead of waiting for a query to arrive and then determining whether cold nodes need to be activated, the system could proactively predict which tables are likely to be accessed in the near future. Based on these predictions, the system could prefetch selected tables, adjust cache contents, and update hot-node placement decisions in advance. This would allow database systems to perform more intelligent and efficient data management by reducing query latency, avoiding unnecessary cold-node wakeups, and improving energy efficiency.

This future direction naturally extends the current GreenDB_DM framework. While GreenDB_DM discovers co-access groups through grouped frequent-pattern mining, an AI-driven prediction model could further enhance the system by learning dynamic workload behavior. In this way, the system would move from pattern-based placement toward predictive and adaptive placement.

Second, future research can extend predictive workload modeling beyond traditional SQL databases to non-relational database systems. Modern data platforms increasingly rely on NoSQL, key-value, document-based, wide-column, and graph databases. These systems are widely used in large-scale applications because they provide flexible data models, high scalability, and efficient support for heterogeneous data. However, they also introduce new challenges for workload prediction and energy-aware optimization because their access patterns are structurally different from relational SQL workloads.

In future work, the application of predictive modeling can be extended to NoSQL, key-value, and document-based database systems. New AI methods can be developed

to learn data access patterns specific to these systems, including temporal and structural behaviors in scenarios such as nested documents, wide-column formats, and graph traversal. For example, in document-based databases, the model may need to learn which nested fields or document collections are likely to be accessed together. In key-value stores, the model may focus on predicting key access sequences and hot-key transitions. In graph databases, the model may learn traversal paths, node neighborhoods, and frequently visited subgraphs.

These predicted access patterns can then be incorporated into caching, prefetching, and data placement strategies to improve operational efficiency and reduce energy consumption. Similar to the role of table-group prediction in GreenDB_DM, access forecasting in non-relational databases could help the system identify which data objects should remain in fast storage, which nodes should remain active, and which parts of the system can safely transition to low-power states. This direction broadens the impact of the dissertation from SQL database systems to heterogeneous data platforms. The core principle remains the same: data systems should learn from workload behavior and use this knowledge to manage resources more efficiently.

Third, future work can integrate workload forecasting and reinforcement learning to enable AI-enhanced energy optimization for data systems. In the current dissertation, GreenDB_DM improves energy efficiency by using mined table co-access patterns to guide hot-node placement. This approach reduces cold-node wakeups by increasing the probability that complete queries can be served by the hot node. However, future database systems may need to make more complex and continuous decisions under changing workloads, including when to activate nodes, when to put nodes into standby, which data should be cached, and how data should be placed across different storage tiers.

Future research can address these challenges by developing reinforcement-learning-based energy optimization methods. In such a system, the database cluster could continuously observe workload characteristics, cache hit rates, query latency, node power states,

and energy consumption. Based on these observations, the learning agent could make real-time decisions about caching, node activation, prefetching, and data placement. The goal would be to reduce energy consumption without causing unacceptable performance loss.

This direction would transform an energy-aware database system from a rule-based or pattern-based framework into a self-managing system. Instead of relying only on predefined thresholds or static placement strategies, the system could learn optimal policies from workload feedback. For example, if the workload becomes more concentrated on a small group of tables, the system could keep only the necessary hot data active and allow more cold nodes to remain in low-power states. If the workload becomes more diverse or shifts toward new table groups, the system could update its placement and prefetching strategy accordingly.

Overall, these future research directions extend the main contributions of this dissertation toward more intelligent and sustainable data systems. The proposed GreenDB_DM framework demonstrates that SQL workload mining and group-aware table placement can improve energy efficiency by connecting access-pattern discovery with system-level resource management. Future work can build on this foundation by introducing AI-driven table prediction, extending access forecasting to non-relational databases, and applying reinforcement learning to real-time energy optimization.

Over the long term, this research aims to connect data analytics, database systems, and sustainable computing. By integrating data mining, predictive modeling, and energy-aware system design, future data systems can learn from complex workloads, adapt to workload changes, manage resources more effectively, and achieve higher computational and energy efficiency. These directions will further support the development of intelligent database systems that are not only high-performing, but also sustainable and energy aware.

Bibliography

[1] S. Agrawal, S. Chaudhuri, and V. R. Narasayya. Automated selection of materialized views and indexes in sql databases. In *VLDB*, volume 2000, pages 496–505, 2000.

[2] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.

[3] H. Amur, J. Cipar, V. Gupta, G. R. Ganger, M. A. Kozuch, and K. Schwan. Robust and flexible power-proportional storage. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, pages 217–228, 2010.

[4] K. Aouiche, J. Darmont, O. Boussaïd, and F. Bentayeb. Automatic selection of bitmap join indexes in data warehouses. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 64–73. Springer, 2005.

[5] S. Aulbach, T. Grust, D. Jacobs, A. Kemper, and J. Rittinger. Multi-tenant databases for software as a service: schema-mapping techniques. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1195–1206, 2008.

[6] A.-L. Barabási. Scale-free networks: a decade and beyond. *science*, 325(5939):412–413, 2009.

[7] L. A. Barroso, U. Hölzle, and P. Ranganathan. *The datacenter as a computer: Designing warehouse-scale machines*. Morgan & Claypool Publishers, 2018.

[8] J. Basilico and T. Hofmann. Unifying collaborative and content-based filtering. In *Proceedings of the twenty-first international conference on Machine learning*, page 9, 2004.

[9] L. Benini, A. Bogliolo, and G. De Micheli. A survey of design techniques for system-level dynamic power management. *IEEE transactions on very large scale integration (VLSI) systems*, 8(3):299–316, 2000.

[10] F. Bodon. A fast apriori implementation. In *FIMI*, volume 3, page 63. Citeseer, 2003.

[11] C. Borgelt. An implementation of the fp-growth algorithm. In *Proceedings of the 1st international workshop on open source data mining: frequent pattern mining implementations*, pages 1–5, 2005.

[12] S. Brin, R. Motwani, and C. Silverstein. Beyond market baskets: Generalizing association rules to correlations. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 265–276, 1997.

[13] T. Bujlow, V. Carela-Español, J. Sole-Pareta, and P. Barlet-Ros. A survey on web tracking: Mechanisms, implications, and defenses. *Proceedings of the IEEE*, 105(8):1476–1510, 2017.

[14] R. Cañamares and P. Castells. Should i follow the crowd? a probabilistic analysis of the effectiveness of popularity in recommender systems. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 415–424, 2018.

[15] S. Che, J. W. Sheaffer, and K. Skadron. Dymaxion: Optimizing memory access patterns for heterogeneous systems. In *Proceedings of 2011 international conference for high performance computing, networking, storage and analysis*, pages 1–11, 2011.

[16] W. Chen, Y. Yuan, and L. Zhang. Scalable influence maximization in social networks under the linear threshold model. In *2010 IEEE international conference on data mining*, pages 88–97. IEEE, 2010.

[17] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4(3):233–235, 1979.

[18] D. Colarelli and D. Grunwald. Massive arrays of idle disks for storage archives. In *SC'02: Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, pages 47–47. IEEE, 2002.

[19] I. Cooper, A. Mondal, and C. G. Antonopoulos. A sir model assumption for the spread of covid-19 in different communities. *Chaos, Solitons & Fractals*, 139:110057, 2020.

[20] C. Curino, E. P. C. Jones, Y. Zhang, and S. R. Madden. Schism: a workload-driven approach to database replication and partitioning. In *Proceedings of the 36th International Conference on Very Large Data Bases (VLDB)*, pages 48–57, 2010.

[21] T. Di Noia, R. Mirizzi, V. C. Ostuni, D. Romito, and M. Zanker. Linked open data to support content-based recommender systems. In *Proceedings of the 8th international conference on semantic systems*, pages 1–8, 2012.

[22] A. Farseev, I. Samborskii, A. Filchenkov, and T.-S. Chua. Cross-domain recommendation via clustering on multi-layer graphs. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, pages 195–204, 2017.

[23] U. Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4):634–652, 1998.

[24] F. Gedikli and D. Jannach. Improving recommendation accuracy based on item-specific tag preferences. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(1):1–19, 2013.

[25] B. Goethals and J. Van den Bussche. Relational association rules: getting warmer. In *Pattern Detection and Discovery: ESF Exploratory Workshop London, UK, September 16–19, 2002 Proceedings*, pages 125–139. Springer, 2002.

[26] D. Goldberg, D. Nichols, B. M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[27] J. Han, H. Cheng, D. Xin, and X. Yan. Frequent pattern mining: current status and future directions. *Data mining and knowledge discovery*, 15(1):55–86, 2007.

[28] J. Han, M. Kamber, and J. Pei. 6-mining frequent patterns, associations, and correlations: Basic concepts and methods. *Data Mining (Third Edition), The Morgan Kaufmann Series in Data Management Systems*, pages 243–278, 2012.

[29] J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. *ACM sigmod record*, 29(2):1–12, 2000.

[30] P. He, L. Sun, X. Gao, Y. Zhou, and X. Qin. Garmt: Grouping-based association rule mining to predict future tables in database queries. *Computers*, 14(6):220, 2025.

[31] S. He, Y. Peng, and K. Sun. Seir modeling of the covid-19 and its dynamics. *Nonlinear dynamics*, 101:1667–1680, 2020.

[32] J. Hipp, U. Güntzer, and G. Nakhaeizadeh. Algorithms for association rule mining—a general survey and comparison. *ACM sigkdd explorations newsletter*, 2(1):58–64, 2000.

[33] L. Huang, H. Chen, X. Wang, and G. Chen. A fast algorithm for mining association rules. *Journal of Computer Science and Technology*, 15:619–624, 2000.

[34] S. Irani, S. Shukla, and R. Gupta. Online strategies for dynamic power management in systems with multiple power-saving states. *ACM Transactions on Embedded Computing Systems (TECS)*, 2(3):325–346, 2003.

[35] J. A. Jacquez and C. P. Simon. The stochastic si model with recruitment and deaths i. comparison with the closed sis model. *Mathematical biosciences*, 117(1-2):77–125, 1993.

[36] S. Jiang and X. Zhang. Lirs: An efficient low inter-reference recency set replacement policy to improve buffer cache performance. *ACM SIGMETRICS Performance Evaluation Review*, 30(1):31–42, 2002.

[37] M. J. Keeling and K. T. Eames. Networks and epidemic models. *Journal of the royal society interface*, 2(4):295–307, 2005.

[38] H. Ko, S. Lee, Y. Park, and A. Choi. A survey of recommendation systems: recommendation models, techniques, and application fields. *Electronics*, 11(1):141, 2022.

[39] Y. Koren, S. Rendle, and R. Bell. Advances in collaborative filtering. *Recommender systems handbook*, pages 91–142, 2021.

[40] E. Y. Lai, Z. Zolaktaf, M. Milani, O. AlOmeir, J. Cao, and R. Pottinger. Workload-aware query recommendation using deep learning. In *EDBT*, pages 53–65, 2023.

[41] W. Lang and J. Patel. Towards eco-friendly database management systems. *arXiv preprint arXiv:0909.1767*, 2009.

[42] V. Leis, A. Gubichev, A. Mirchev, P. Boncz, A. Kemper, and T. Neumann. How good are query optimizers, really? *Proceedings of the VLDB Endowment*, 9(3):204–215, 2015.

[43] M. Li, X. Wang, K. Gao, and S. Zhang. A survey on information diffusion in online social networks: Models and methods. *Information*, 8(4):118, 2017.

[44] Z. Li, Z. Chen, S. M. Srinivasan, Y. Zhou, et al. C-miner: Mining block correlations in storage systems. In *FAST*, volume 4, pages 173–186, 2004.

[45] T.-P. Liang, H.-J. Lai, and Y.-C. Ku. Personalized content recommendation and user satisfaction: Theoretical synthesis and empirical findings. *Journal of Management Information Systems*, 23(3):45–70, 2006.

[46] E. Liu, M. Hashemi, K. Swersky, P. Ranganathan, and J. Ahn. An imitation learning approach for cache replacement. In *International Conference on Machine Learning*, pages 6237–6247. PMLR, 2020.

[47] P. Luarn, J.-C. Yang, and Y.-P. Chiu. The network effect on information dissemination on social network sites. *Computers in Human Behavior*, 37:1–8, 2014.

[48] D. Macpherson. A survey of homogeneous structures. *Discrete mathematics*, 311(15):1599–1634, 2011.

[49] M. Mao, J. Lu, G. Zhang, and J. Zhang. A fuzzy content matching-based e-commerce recommendation approach. In *2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, pages 1–8. IEEE, 2015.

[50] E. Masanet, A. Shehabi, N. Lei, S. Smith, and J. Koomey. Recalibrating global data center energy-use estimates. *Science*, 367(6481):984–986, 2020.

[51] N. Megiddo and D. S. Modha. {ARC}: A {Self-Tuning}, low overhead replacement cache. In *2nd USENIX Conference on File and Storage Technologies (FAST 03)*, 2003.

[52] A. Mnih and R. R. Salakhutdinov. Probabilistic matrix factorization. *Advances in neural information processing systems*, 20, 2007.

[53] B. Mobasher, H. Dai, T. Luo, and M. Nakagawa. Using sequential and non-sequential patterns in predictive web usage mining tasks. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 669–672. IEEE, 2002.

[54] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions—i. *Mathematical programming*, 14(1):265–294, 1978.

[55] M. T. Özsu, P. Valduriez, et al. *Principles of distributed database systems*, volume 2. Springer, 1999.

[56] J. Park and A.-L. Barabási. Distribution of node characteristics in complex networks. *Proceedings of the National Academy of Sciences*, 104(46):17916–17920, 2007.

[57] M. J. Pazzani and D. Billsus. Content-based recommendation systems. In *The adaptive web: methods and strategies of web personalization*, pages 325–341. Springer, 2007.

[58] E. Pinheiro and R. Bianchini. Energy conservation techniques for disk array-based servers. In *Proceedings of the 18th annual international conference on Supercomputing*, pages 68–78, 2004.

[59] M. Poess and R. O. Nambiar. Energy cost, the key challenge of today's data centers: a power consumption analysis of tpc-c results. *Proceedings of the VLDB Endowment*, 1(2):1229–1240, 2008.

[60] A. Quamar, K. A. Kumar, and A. Deshpande. Sword: scalable workload-aware data placement for transactional workloads. In *Proceedings of the 16th international conference on extending database technology*, pages 430–441, 2013.

[61] M. J. Raddick, A. R. Thakar, A. S. Szalay, and R. D. Santos. Ten years of skyserver i: Tracking web and sql e-science usage. *Computing in Science & Engineering*, 16(4):22–31, 2014.

[62] Y. Remil, A. Bendimerad, R. Mathonat, P. Chaleat, and M. Kaytoue. " what makes my queries slow?": Subgroup discovery for sql workload analysis. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 642–652. IEEE, 2021.

[63] J. Sooknanan and D. Comissiong. Trending on social media: integrating social media into infectious disease dynamics. *Bulletin of Mathematical Biology*, 82(7):86, 2020.

[64] S. H. Strogatz. Exploring complex networks. *nature*, 410(6825):268–276, 2001.

[65] E. Thereska, A. Donnelly, and D. Narayanan. Sierra: practical power-proportionality for data center storage. In *Proceedings of the sixth conference on Computer systems*, pages 169–182, 2011.

[66] P. Van Mieghem. The n-intertwined sis epidemic network model. *Computing*, 93(2):147–169, 2011.

[67] L. D. Vasto-Terrientes, A. Valls, P. Zielniewicz, and J. Borras. A hierarchical multicriteria sorting approach for recommender systems. *Journal of Intelligent Information Systems*, 46:313–346, 2016.

[68] A. Viguerie, G. Lorenzo, F. Auricchio, D. Baroli, T. J. Hughes, A. Patton, A. Reali, T. E. Yankeelov, and A. Veneziani. Simulating the spread of covid-19 via a spatially-resolved susceptible–exposed–infected–recovered–deceased (seird) model with heterogeneous diffusion. *Applied mathematics letters*, 111:106617, 2021.

[69] E. Volz and L. A. Meyers. Susceptible–infected–recovered epidemics in dynamic contact networks. *Proceedings of the Royal Society B: Biological Sciences*, 274(1628):2925–2934, 2007.

[70] S. Wan and Z. Niu. A hybrid e-learning recommendation approach based on learners' influence propagation. *IEEE Transactions on Knowledge and Data Engineering*, 32(5):827–840, 2019.

[71] C. Wang, W. Chen, and Y. Wang. Scalable influence maximization for independent cascade model in large-scale social networks. *Data Mining and Knowledge Discovery*, 25:545–576, 2012.

[72] J. Wang, H. Zhu, and D. Li. eraid: Conserving energy in conventional disk-based raid system. *IEEE Transactions on Computers*, 57(3):359–374, 2008.

[73] Y. Wei, X. Wang, Q. Li, L. Nie, Y. Li, X. Li, and T.-S. Chua. Contrastive learning for cold-start recommendation. In *Proceedings of the 29th ACM international conference on multimedia*, pages 5382–5390, 2021.

[74] O. Wolfson, S. Jajodia, and Y. Huang. An adaptive data replication algorithm. *ACM Transactions on Database Systems (TODS)*, 22(2):255–314, 1997.

[75] L. Yang, R.-S. Chen, Y.-M. Siu, and K.-K. Soo. Papr reduction of an ofdm signal by use of pts with low computational complexity. *IEEE Transactions on broadcasting*, 52(1):83–86, 2006.

[76] Z. Ye, L. Zhang, K. Xiao, W. Zhou, Y. Ge, and Y. Deng. Multi-user mobile sequential recommendation: An efficient parallel computing paradigm. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2624–2633, 2018.

[77] H. Yin, Q. Wang, K. Zheng, Z. Li, and X. Zhou. Overcoming data sparsity in group recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 34(7):3447–3460, 2020.

[78] J. Yin, H. Jia, B. Zhou, T. Tang, L. Ying, S. Ye, T.-Q. Peng, and Y. Wu. Blowing seeds across gardens: Visualizing implicit propagation of cross-platform social media posts. *IEEE Transactions on Visualization and Computer Graphics*, 2024.

[79] X. Zeng, Y. Hui, J. Shen, A. Pavlo, W. McKinney, and H. Zhang. An empirical evaluation of columnar storage formats. *Proceedings of the VLDB Endowment*, 17(2):148–161, 2023.

[80] M. Zhang, S. Wu, X. Yu, Q. Liu, and L. Wang. Dynamic graph neural networks for sequential recommendation. *IEEE Transactions on Knowledge and Data Engineering*, 35(5):4741–4753, 2022.

[81] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020.

[82] Y. Zhou, S. Taneja, C. Zhang, and X. Qin. Greendb: Energy-efficient prefetching and caching in database clusters. *IEEE Transactions on Parallel and Distributed Systems*, 30(5):1091–1104, 2018.