AUTOMATED BIRTHDAY EMAIL ALERT SYSTEM POWERED BY HASHING

G.Wiselin Jiji
Professor & Head, Department of Computer Science & Engineering
Dr. Sivanthi Aditananr College of Engineering, Tiruchendur

M.M.Saara Raihana
UG Student, Department of Computer Science & Engineering
Dr. Sivanthi Aditananr College of Engineering, Tiruchendur

V. Dharshini
UG Student, Department of Computer Science & Engineering
Dr. Sivanthi Aditananr College of Engineering, Tiruchendur

DOI: doi.org/10.34293/shanlax.9789361631474.ch024

Abstract

This work introduced an Automated Birthday Alert System implemented using data structure technique. Managing and remembering birthdays in organizations, colleges, or groups can be challenging, especially when handling a large number of members. This project presents an Automated Birthday Alert System developed using Java and MySQL that addresses this problem efficiently. The system stores user details, including name, email, and date of birth, in a structured database. A generated hash column (dob_hash) is used to transform birthdates into a simplified format, enabling fast and accurate detection of upcoming birthdays. Indexing the hash column further optimizes query performance, ensuring timely identification even for large datasets. Once birthdays are detected, a Java-based application automatically sends advance birthday wishes to the concerned individual and reminder emails to other members. The system's execution is fully automated using MySQL Event Scheduler or Task Scheduler, eliminating the need for manual intervention. Security is ensured through Gmail App Password authentication and SSL/TLS encryption for safe email transmission. A comparative study was conducted between hashing-based flat list searching and tree-based postorder traversal. Results indicate that hashing provides significantly faster execution (55,448 ms) compared to tree traversal (133,934 ms), while maintaining 100% alert accuracy. Hashing is ideal for flat datasets such as offices or clubs, whereas tree traversal suits hierarchical data such as family trees or structured organizations.

Keywords: *Birthday Alert, Hashing, MySQL Indexing, Java Mail API, Event Scheduler*

Introduction

In today's digitally connected world, automation plays a vital role in improving efficiency, accuracy, and user engagement [1] across various applications. One such area where automation has proven beneficial is in the management of personal and organizational reminders. Remembering important dates such as birthdays can be challenging, especially when dealing with large numbers of users or records. Manual tracking methods are often inefficient, error-prone,[2] and time-consuming. To address these challenges,

automated alert systems have gained prominence as reliable and scalable solutions.

Keeping track of birthdays in organizations, colleges, or teams becomes challenging when the number of members is large. Relying on manual methods often leads to forgotten dates and missed birthday greetings [3] . To address this problem, an automated system has been developed that stores birthday details in a well-structured database, applies hashing techniques to efficiently identify birthdays occurring today or the next day, and automatically sends email notifications without requiring human intervention. This system [4] is particularly beneficial for colleges, companies, and group-based environments, where timely birthday reminders help strengthen social interaction and foster a sense of connection among members.

An Automated Birthday Alert System is designed to automatically identify upcoming birthdays from stored user data and notify relevant users through electronic communication channels[5]. Such systems are particularly useful in organizations, institutions, and social platforms where maintaining user relationships and timely communication is essential. By automating birthday reminders and greetings, organizations can enhance user satisfaction, improve engagement, and reduce the administrative burden associated with manual reminder systems.

With the rapid growth of data, efficient data storage and retrieval have become critical concerns in software system design. Databases often contain large volumes of records, and [6] searching through them repeatedly for specific conditions—such as matching birth dates—can negatively impact performance. Traditional query-based search mechanisms may become inefficient as data size increases. To overcome this limitation, advanced data organization [7] and optimization techniques such as hashing and indexing are commonly employed. Hashing allows data to be transformed into a fixed-size value, enabling faster comparisons and lookups, while indexing significantly reduces query execution time.

This work introduces an Automated Birthday Alert System implemented using Java and MySQL, integrating database-level hashing techniques to optimize birthday searches. The system stores user details, including date of birth, in a relational database. To enable efficient searching based on month and day, a generated hash column, referred to as *dob_hash*, is created using hashing logic. This column combines relevant components of the date of birth and is indexed to facilitate rapid retrieval of upcoming birthday records. By using hashing at the database level, the system minimizes computational overhead and improves overall performance.

Java has been chosen as the core programming language for this work due

to its platform independence, robustness, and extensive support for database connectivity and email services. The Java Database Connectivity (JDBC) API enables seamless interaction between the Java application and the MySQL database. Additionally, Java's strong ecosystem allows easy integration of scheduling and communication modules, making it suitable for developing automated alert systems.

A key feature of the proposed system is automated email notification. Once the system identifies users whose birthdays are approaching, it automatically sends advance birthday wishes and reminder emails [8]. This functionality is implemented using the Jakarta Mail API, which provides a reliable framework for composing and sending emails from Java applications. Gmail's Simple Mail Transfer Protocol (SMTP) service is used to ensure secure and dependable email delivery. Automated email scheduling ensures that notifications are sent at predefined times without manual intervention, enhancing reliability and consistency.

Security and efficiency are important considerations in automated systems. While the system does not store sensitive data such as passwords, hashing is employed as a performance optimization technique rather than for security purposes. By generating a hash column specifically for birthday searches, the system avoids repetitive date computations and full table scans. Indexing the hash column further improves performance by allowing the database engine to quickly locate matching records. This design demonstrates how database optimization techniques can be effectively applied to real-world automation problems.

At present, the work focuses primarily on database hashing and automated email alert scheduling. The architecture is designed to be modular and extensible, allowing future enhancements without significant restructuring. One planned enhancement is the incorporation of alternative data retrieval methods, such as tree traversal techniques. By comparing hashing-based searches with tree-based approaches, such as binary search trees or B-trees, the system can be evaluated in terms of performance, scalability, and resource utilization. Such comparative analysis can provide valuable insights into the suitability of different data structures for time-based search problems.

Methodology

The Automated Birthday Alert System operates through a sequence of well-defined steps that ensure efficient data management, quick birthday identification, and timely email notifications. The overall design emphasizes automation, performance optimization, and minimal manual intervention, making the system suitable for use in organizations, colleges, and group environments.

Data Storage

The first step in the system involves storing personal details of users in a MySQL database. These details typically include the user's name, email address, and date of birth. MySQL is used as the backend database due to its reliability, structured data handling, and strong support for indexing and generated columns. Storing data in a relational format ensures consistency and allows easy querying and maintenance as the number of records grows.

Hashing Logic

To efficiently identify birthdays based on specific dates, the system applies hashing logic to the date of birth field. Instead of repeatedly performing complex date calculations, the birthdate is transformed into a simplified *MM-DD* format using a generated column called *dob_hash*. This generated hash column extracts only the month and day components from the date of birth, making it easier to compare upcoming birthdays regardless of the year. This approach reduces computational overhead and improves query efficiency.

Fast Search Using Indexing

To further enhance performance, an index is created on the *dob_hash* column. Indexing allows the database engine to locate matching records quickly without scanning the entire table. As the dataset increases, this indexed hash-based approach ensures that the system continues to perform efficiently. The combination of hashing and indexing significantly reduces query execution time, especially when the system runs daily checks for upcoming birthdays.

Birthday Matching

Once the hashing and indexing mechanisms are in place, the system performs birthday matching through an SQL query. The query compares the hash value of the upcoming date—typically tomorrow's date in *MM-DD* format—with the stored *dob_hash* values in the database. If a match is found, the corresponding user records are retrieved. This process allows the system to accurately identify individuals whose birthdays are approaching without unnecessary processing.

Email Automation

After identifying matching birthday records, the system initiates the email notification process. A Java-based application handles email automation using the Jakarta Mail API. Two types of emails are generated automatically: personalized birthday wishes sent directly to the individual celebrating their birthday, and reminder emails sent to all members of the group or organization. This dual-notification approach ensures that the birthday is acknowledged while also encouraging others to participate in the celebration. Gmail's SMTP service is used to ensure secure and reliable email delivery.

Scheduling

To eliminate the need for manual execution, the entire process is automated using the MySQL Event Scheduler. The scheduler is configured to trigger the birthday-checking process once every day at a predefined time. When activated, it runs the necessary queries and initiates the Java email-sending program. This scheduling mechanism ensures that the system operates consistently and independently, providing timely notifications without administrator involvement.

Flowchart Description

The system flow begins with the MySQL database storing user details. The date of birth is converted into a hash format using a generated column. The indexed hash column enables fast searching, after which an SQL query checks for matching birthdays. If a match is found, the Java application sends birthday and reminder emails. Finally, the MySQL Event Scheduler ensures that this entire sequence runs automatically on a daily basis. This logical flow guarantees accuracy, efficiency, and full automation of the birthday alert process as shown.

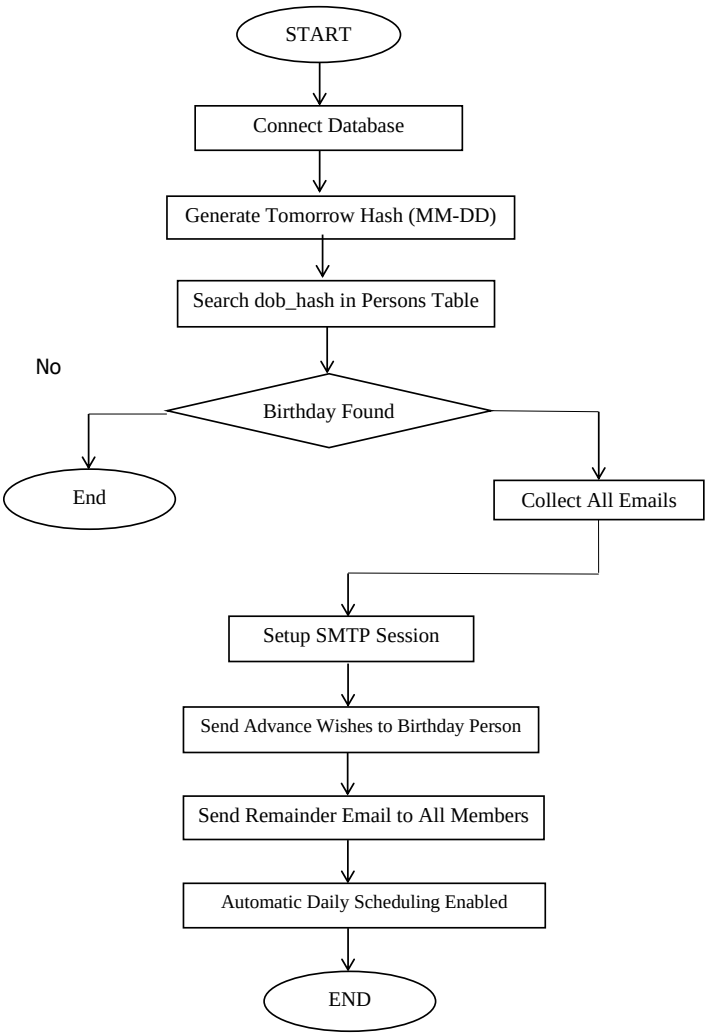

Figure 1. Flowchart for proposed work

Results & Discussion

The system effectively identifies upcoming birthdays using hashing techniques and automatically sends email notifications. This project has been successfully implemented and delivers several key outcomes. A *persons* table is created and maintained in the database to store essential user details, including name, email address, and date of birth. To enable fast and efficient birthday searches, a hashing mechanism in the form of a generated column (*dob_hash*) is utilized. This approach allows the system to quickly process birthday data and improve overall search performance.

The system is capable of automatically detecting members whose birthdays fall on the following day. Once such records are identified, a Java-based application sends advance birthday greetings to the concerned individual along with reminder emails to other registered members. To ensure continuous operation without manual intervention, the system is configured using either the Task Scheduler or the MySQL Event Scheduler, enabling it to execute automatically on a daily basis.

Discussion

From the results listed in Table1 and 2 it is evident that the **hashing-based flat list approach** provides superior execution performance compared to tree-based postorder traversal. Hashing enables **constant-time (O(1)) lookup** for birthdays, making it ideal for scenarios with large flat datasets, such as corporate offices or clubs. On the other hand, **postorder traversal** is more suitable for hierarchical datasets where parent-child relationships need to be maintained, such as family trees or organizations with nested structures, although it incurs higher execution time (O(n) per search).

Table 1. Comparison based on execution Time

Sl. No	Technique used	Execution Time (ms)
1	Hashing (Flat List)	55448
2	Postorder Traversal	133934

Table 2. Comparison analysis

Feature / Metric	Case 1: Hashing (Flat List)	Case 2: Postorder Traversal (Family Tree)
Data Structure	Flat list with dob_hash	Tree with parent-child hierarchy
Birthday Detection Method	Direct hash match	Postorder traversal
Automation	Task Scheduler/ Event Scheduler	Task Scheduler / Event Scheduler
Emails Sent	Birthday wishes to members	Birthday wishes to members
Reminder Emails	Sent to all members	Sent to all members
Duplicate Prevention	UNIQUE constraint (uniq_ alert)	UNIQUE constraint (uniq_ notification)

Execution Time	55448 ms	133934 ms
Scalability	Best for flat lists	Best for hierarchical relationships
Accuracy	100% alerts sent	100% alerts sent
Use Case	Offices, clubs	Families, organizations with hierarchy
Complexity	O(1) lookup per birthday (hash)	O(n) traversal per birthday (tree)

Both methods maintain **full automation**, send birthday and reminder emails, and prevent duplicate notifications using unique constraints. Accuracy is maintained at 100% in both cases, ensuring no birthday alerts are missed.

This comparison highlights a trade-off between **performance and structural complexity**: hashing excels in speed and simplicity for flat datasets, while tree traversal is appropriate for hierarchical contexts where relational dependencies are significant.

Analysis

One of the primary advantages of the Birthday Alert System is fast searching through the use of hashing. The indexed *dob_hash* column significantly minimizes query execution time and enhances database performance. Another key benefit is full automation, as the system eliminates the need for manual birthday tracking. Daily checks and alerts are generated automatically, ensuring no important dates are missed.

Email automation is another major advantage. The system sends advance birthday wishes directly to the birthday person while also distributing reminder notifications to all registered members. This dual notification mechanism helps improve participation and social interaction within the group. To prevent repeated or duplicate alerts, a UNIQUE constraint is implemented in the database, ensuring that each birthday notification is sent only once per person.

The project is designed to be scalable and extendable. Future enhancements can include comparative analysis of different searching techniques, such as tree traversal methods like postorder traversal, to evaluate performance and efficiency. Additionally, secure email communication is ensured through the use of Gmail App Password authentication instead of standard passwords, enhancing account security. Email transmission is further protected using SSL/TLS encryption (TLS v1.2), ensuring secure communication between the application and the SMTP server. Trusted SMTP configurations are also employed to prevent unauthorized access and server connections.

The current implementation of the Automated Birthday Alert System demonstrates that hashing is an effective technique for fast and reliable searching within a database environment. By using a generated hash column based on the month and day of birth, along with

proper indexing, the system significantly reduces query execution time. This approach avoids full table scans and minimizes repetitive date computations, making it efficient even as the number of user records increases. As a result, the system is capable of accurately identifying upcoming birthdays and triggering automated email notifications in a timely manner.

The integration of hashing at the database level also simplifies the application logic. Since the database handles the optimized search operations, the Java application can focus primarily on email automation and scheduling. This separation of concerns improves system maintainability and ensures consistent performance. Overall, the hashing-based approach proves to be well-suited for applications that require frequent date-based lookups.

Despite its effectiveness, hashing is not the only method available for optimized searching. Future enhancements to the system may include the implementation of tree-based searching techniques for comparative analysis. Data structures such as binary search trees, AVL trees, or B-trees can be explored to evaluate their performance in handling time-based queries. Tree traversal methods, including inorder, preorder, or postorder traversal, may offer alternative ways to organize and retrieve birthday data.

By comparing hashing and tree-based methods, the system can be evaluated in terms of search speed, scalability, memory usage, and overall efficiency. Such comparative studies would provide valuable insights into the strengths and limitations of each approach and help determine the most suitable technique for different data sizes and use cases. These future enhancements can contribute to further optimization and academic exploration of data retrieval strategies within automated alert systems.

Conclusion

The Automated Birthday Alert System successfully demonstrates the integration of database optimization, hashing techniques, and email automation to provide an efficient solution for managing birthday notifications. By storing user details in a MySQL database and applying a generated hash column (*dob_hash*), the system is able to quickly identify upcoming birthdays with minimal computational overhead. Indexing the hash column further enhances performance, ensuring rapid and reliable detection even for large datasets.

The Java-based application automates the sending of advance birthday wishes and reminder emails, while the use of Task Scheduler or MySQL Event Scheduler ensures the system operates daily without manual intervention. Security measures, including Gmail App Password authentication and SSL/TLS encryption, guarantee safe and reliable email transmission.

Comparative analysis between hashing-based flat list searches and tree-

based postorder traversal demonstrates that hashing offers faster execution times and is well-suited for flat datasets, while tree traversal is appropriate for hierarchical data structures. Both methods maintain 100% alert accuracy and prevent duplicate notifications.

Reference

1. Kakolu, Sridevi, and Muhammad Ashraf Faheem. "Digitization and automation in mobile applications: A catalyst for operational efficiency and user engagement." (2023).
2. Athoillah, Muhammad, and Avicena Fahmi Wibisono. "Automated Goods Tracking System: Enhancing Accuracy, Efficiency, and Real-Time Data Integration." *IT for Society* 10, no. 1 (2025).
3. Martin, Judith. *Miss Manners' Guide to Excruciatingly Correct Behavior (Freshly Updated)*. WW Norton & Company, 2011.
4. Vinokur, Eli, Avinoam Yomtovian, Guy Itzchakov, Marva Shalev Marom, and Liat Baron. "Social-based learning and leadership (SBL): theory development and a qualitative case study." *Sustainability* 15, no. 22 (2023): 15800.
5. Curty, Renata Gonçalves, and Ping Zhang. "Website features that gave rise to social commerce: a historical analysis." *Electronic commerce research and applications* 12, no. 4 (2013): 260-279.
6. Hambrick, Donald C., and Lynn M. Crozier. "Stumblers and stars in the management of rapid growth." *Journal of business venturing* 1, no. 1 (1985): 31-45.
7. de Haes, Helias A. Udo, Reinout Heijungs, Sangwon Suh, and Gjalt Huppes. "Three strategies to overcome the limitations of life-cycle assessment." *Journal of industrial ecology* 8, no. 3 (2004): 19-32.
8. Kannan, Anjuli, Karol Kurach, Sujith Ravi, Tobias Kaufmann, Andrew Tomkins, Balint Miklos, Greg Corrado et al. "Smart reply: Automated response suggestion for email." In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pp. 955-964. 2016