PYTHON WEB SYSTEM TO RESTORE SQL SERVER DATABASE TO DRC WITH ADVANCED INFORMATION RETRIEVAL

(Sistem Web Python untuk Memulihkan Database SQL Server ke DRC dengan Pengambilan Informasi Lanjutan)

Devis Rabertra[1], Irwansyah Saputra[2]
[1,2] Department of Computer Science, Nusa Mandiri University
Depok, West Java, Indonesia
Email: 14240019@nusamandiri.ac.id, irwansyah.iys@nusamandiri.ac.id

Abstract

Abstract: Disaster Recovery Centers (DRC) play a crucial role in ensuring the availability and continuity of database operations in enterprise environments. The process of restoring databases from production servers to DRCs is often performed manually, which can lead to errors such as selecting incorrect backups, corrupted files, and lengthy search times. The complexity increases with the growing number of database s and the variety of daily backup types. This study develops an automated system based on a Python Web Interface integrated with Advanced Information Retrieval (IR) to improve the accuracy and speed of finding relevant backups before restoration. The system employs Natural Language Processing (NLP) and multi-criteria relevance scoring, evaluating backup suitability based on fuzzy matching of database names, recency, semantic similarity, backup type, and file size. Testing was conducted using 28 backup records from 5 different databases. Results show that Advanced IR can accelerate backup searches in under 2 seconds, with relevance ranking ranging from 38% to 67%. Additionally, the automated restore process via Python achieved an average execution time of 7.49 seconds with a 100% success rate.

Keywords: *SQL Server, Query Evaluation, Retrieval-Augmented Generation (RAG), Information Retrieval, Database Optimization.*

**Corresponding Author*

1. INTRODUCTION

In general practice, the process of restoring a SQL Server database to a DRC is still done manually, which involves selecting backup files based on the date, size, and type of backup. This method has several drawbacks, including the potential for human error, delays in selecting appropriate backups, and difficulty in assessing the most relevant backup files when backup volumes increase daily [1]. Furthermore, the manual process does not provide automatic validation of backups to be restored, thus increasing the risk of restore failure and slowing service recovery.

As the need for automation and system intelligence in database management increases, a solution capable of searching and selecting backups quickly, accurately, and securely is required [2]. Therefore, this study proposes the development of a Python Web Interface integrated with Advanced Information Retrieval (IR) to automate the process of restoring SQL Server databases to DRC servers. This system utilizes Natural Language Processing (NLP) technology [3] to understand user search input, as well as a multi-criteria relevance scoring algorithm [4] to determine the most relevant backup based on database name [5], time proximity, backup type, file size, and semantic match level [6,7].

2. RELATED STUDIES

Research on automating the restore process and utilizing information intelligence technology in database management has been conducted using various approaches. However, most previous research has focused on optimizing backup and restore performance and has not utilized Intelligent Information Retrieval to automatically select the most relevant backups.

Research by Santos et al. developed an automated SQL Server backup and restore system using PowerShell to accelerate the database recovery process [1]. This system successfully automated script execution but still relied on manual backup file selection without a ranking mechanism or intelligent search based on specific criteria. Therefore, this approach was unable to address the problem of human error in backup selection, which was the primary focus of this study.

Furthermore, research by Kumar & Sharma proposed the application of Natural Language Processing (NLP) to data management to improve the efficiency of metadata-based document searches and semantic matching [9]. Although it made significant contributions to the field of information retrieval, the research was not applied in the context of database backup file management or Disaster Recovery systems, so it did not touch on the technical aspects of data recovery in an enterprise environment.

Another study by Rahman developed a web-based application for centralized database backup monitoring and execution using Python Flask [10]. This system provides increased visibility and real-time backup management through a centralized dashboard, but is still limited to monitoring functionality and does not include backup selection using the multi-criteria ranking algorithm required for intelligent restore processes.

On the other hand, Advanced Information Retrieval approaches and multi-parameter-based scoring methods have been implemented effectively in various fields such as scientific document retrieval and digital content recommendation [4]. Research by Liu et al. shows that the combination of semantic similarity and weighted criteria can produce more accurate rankings in recommendation systems. However, the application of this approach in the database disaster recovery domain is still rare, especially in the context of selecting SQL Server backup files.

Research related to optimizing backup and restore strategies in distributed environments has also been conducted by Becker & Weber (2018) [13], who analyzed the performance of various SQL Server backup strategies. Meanwhile, research on an automation framework for the recovery process in enterprise platforms has been developed by Jan & Khan (2020) [14], which provides a conceptual basis for the development of automated systems.

In the context of applying NLP and fuzzy matching to large-scale data retrieval optimization, Florido & Lopes [15] demonstrated the effectiveness of this approach in improving search accuracy. However, its application is still limited to the general document retrieval domain and has not been adapted to the technical needs of database backup management.

Based on the literature review, it can be concluded that there is a research gap related to the integration of Python Web Interface, Natural Language Processing, and Advanced Information Retrieval for automation and optimization of the SQL Server database restore process to the DRC server [1-4,13-15]. This research provides a new contribution by combining an automated restore engine with intelligent backup selection to improve the accuracy, speed, and security of the database recovery process in an enterprise environment.

Table 1. Comparison of Related Research

Research	Technology Methods	Gap	Relevant
Liu et al. (2022)	Multi-Criteria Ranking & Information Retrieval	ster Recover y system or database backup	Serve as a reference scoring algorithm for Advanced IR.
Jan & Khan (2020)	Enterprise Recovery Automation Framework	Not specific to SQL Server and does not integrate with IR	Provides a conceptual basis for the automation framework
Florido & Lopes (2021)	NLP and Fuzzy Matching for Data Retrieval	Not yet adapted for database backup managem ent	Be a reference for fuzzy matching and NLP processing techniques
This researc h (2025)	Python Web Interface + Advanced IR + NLP + Automated Restore Engine	–	Complete and intelligent solution for Disaster Recovery automation

3. RESEARCH METHODS

This research methodology is designed to develop a Python Web Interface system integrated with Advanced Information Retrieval (IR) to automate the process of restoring a SQL Server database to a Disaster Recovery Center (DRC) server. The approach used is an iterative development model [12], which allows the development process to be carried out in stages and repeatedly based on feedback from implementation and testing results.

3.1 Research Approach

This research uses a software engineering approach that includes the following stages [13].

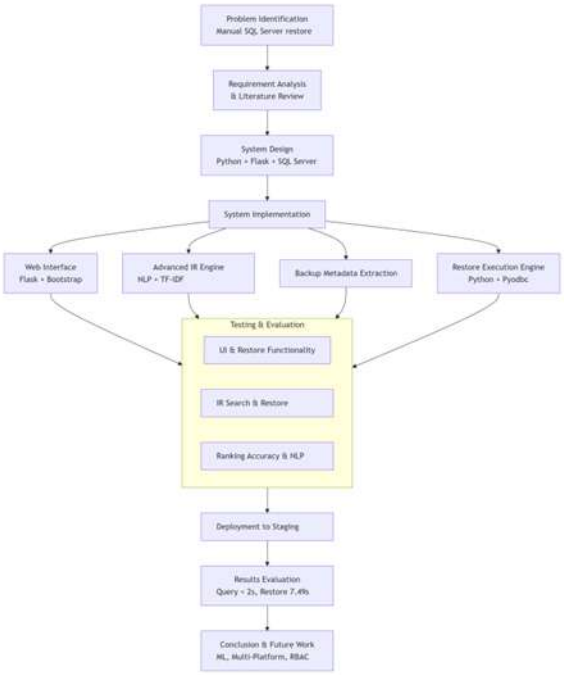

Figure 1: Research Flow Diagram of the Automated SQL Server Database Restore System Integrated with Advanced Information Retrieval.

A. Requirements Analysis

This is done to identify user needs related to database restore automation, selection of relevant backups, security of the restore process, and interface requirements. users based Web. Requirements gathering is done through observation of the manual restore process and analysis of problems that occur in the production environment, with reference to the SQL Server backup and restore documentation [15].

B. Systems Design and Architecture

At this stage, the system architecture is designed, including the application module structure, Python Flask integration, SQL Server as a metadata backup source, and the Advanced IR Engine. In addition, an automatic restore flow is designed using stored procedures and backup validation via the RESTORE HEADERONLY command [11]. The system architecture adopts modern database system design principles [13].

C. Implementation

The implementation stage is carried out by building the main components of the system, namely: Web Interface using Python Flask and Bootstrap Backup Metadata Extraction Engine to read SQL Server backup metadata with a data mining approach [6], Advanced IR Engine based on NLP [8], TF-IDF, cosine similarity [7,9], and multi-criteria scoring [4], Restore Execution Engine to run a controlled restore process via Python based on a recovery automation framework [14]

D. Testing and Evaluation

Testing was conducted using real production backup records to evaluate backup search performance and restore success. The types of testing performed included:

Functional Testing (interface functionality and restore engine), Performance Testing (IR query time and restore time) with reference to backup/restore performance analysis standards [13], Accuracy Testing (backup ranking validity and NLP parsing success) using a fuzzy matching approach [15]

E. Deployment and Verification

The system was tested on a staging server and a DRC server to ensure operational environment compatibility and suitability to end-user needs, taking into account disaster recovery optimization aspects [11].

3.2 System Architecture

The system architecture developed in this study is designed as an integrated framework that supports automated database restoration through coordinated interaction among several core components. The architecture combines a web-based user interface, an Advanced Information Retrieval (IR) engine, and a restore execution module, all of which operate on backup metadata obtained from the SQL Server environment. The web interface serves as the interaction layer for users to submit natural language queries and monitor restoration processes. The IR engine processes these queries using Natural Language Processing, fuzzy matching, semantic similarity, and multi-criteria relevance scoring to identify and rank the most appropriate backup files. Based on the selected results, the restore execution module performs validation and automatically executes the database restoration process on the Disaster Recovery Center (DRC) server, ensuring accuracy, efficiency, and reduced human error.

Figure 2. System architecture

3.3 Advanced Information Retrieval Algorithm

The IR process for selecting the most relevant backup is carried out through the following steps:

A. Natural Language Processing

Parsing user queries like "last week's backup" or "last warehouse database restore" using NLP techniques

Entity extraction: database_name, date, semantic context with an approach developed in fuzzy matching research [15]

Multi-Criteria Relevance Scoring [4,7,9]The assessment weights were developed based on research on multi-criteria ranking [4] and term-weighting approaches [9]:

Fuzzy name matching (35%) uses an algorithm adapted from Florido & Lopes [15] Recency score based on exponential decay (25%) Semantic similarity using TF-IDF + cosine similarity (20%) based on the concept of information retrieval [7,9] Completeness score based on backup type & size (20%) refers to SQL Server documentation [5]

Backup Ranking Result

The system displays a list of backups that have been sorted by the highest level of relevance with an optimized scoring algorithm [4].

3.4 Evaluation Metrics

The evaluation is carried out based on the following metrics [13,14]:

Query Processing Time (target < 2 seconds) based on recovery system performance standards [13], Restore Execution Time (average 7--12 seconds) refers to performance analysis research [13], Success Rate of Restore (target ≥ 100% of tests performed) according to the recovery automation framework [14], Accuracy Score for backup selection based on relevance ranking results [4,15].

3.5 Research Dataset

The data used is real production backup data with:

28 backup records from 9 databases, Backup types Full, Differential, and Log according to SQL Server backup types documentation [5], Backups of different dates for recency scoring test, Dataset managed with data mining approach for effective metadata extraction [6].

4. RESULTS AND DISCUSSION

The Automated SQL Server Database Restore to DRC system was implemented using a Python Web Interface integrated with Advanced Information Retrieval. The implementation was carried out in an enterprise environment consisting of a production database, a DRC server, and a web-based operational dashboard.

4.1 Development Environment

System implementation is carried out with the following environmental specifications:

Table 2. Development Environment Specifications

Component	Description
Backend Framework	Python 3.11 + Flask
Frontend	HTML, Bootstrap 5, JavaScript
Database System	Microsoft SQL Server 2019
Connection Method	Pyodbc (SQLServer)
Metadata Storage	Restore Header Only
DRC Server	SQL Server

This technological approach was chosen to ensure compatibility with the enterprise environment and optimal restore process performance.

4.2 System Component Implementation

The system implementation consists of three main components:

4.2.1 Web Interface (Frontend Layer)


Figure 3. Main Dashboard View of the Automatic Restore System.

The web interface is developed using Flask + Bootstrap, providing the following features:

Input backup search commands based on natural language, Display a list of backups ranked by relevance score, Execute restore button with security confirmation, Progress tracking and execution status results

The frontend acts as an operational control panel for database administrators, adopting the concept of a centralized monitoring system [3]

Figure 4. Backup History of the Selected Database.

4.2.2 Advanced IR Engine (Processing Layer)

This component functions to perform backup search processing using:

Natural Language Processing for query parsing, Multi-Criteria Ranking produces relevance scores

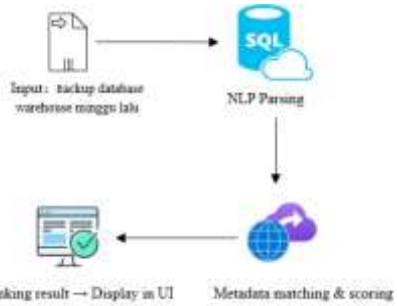

Figure 5. Processing Flow.

The ranking results are calculated based on the fuzzy match weight of the database name, recency score, semantic similarity, and completeness score.

Figure 6. Smart Search Interface with Integrated Natural Language Query Advanced IR Engine

4.2.3 Restore Execution Engine (Backend Layer)

The restore engine manages the restore execution process through Python commands and SQL Server scripts as follows:

Backup validation using RESTORE HEADERONLY, Automatic restore process with:, RESTORE DATABASE db_name, FROM DISK = '<selected_backup_path>', WITH REPLACE, STATS=1;, Displays execution log status in real-time

The engine ensures that restores run safely without manual intervention.

Figure 7. Restore Confirmation and Execution Process via Web Interface.

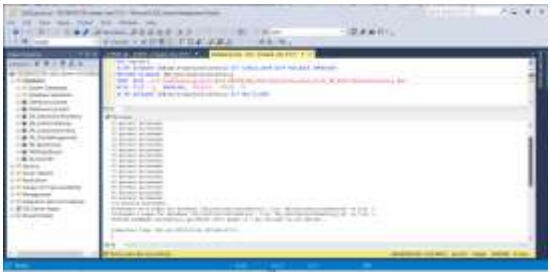

Figure 8. Successful Restore Results in SQL Server Management Studio (SSMS).

4.3 Backup Metadata Extraction

Figure 9. Results of Extracting Backup Metadata Using the Command.

The output is fed into a metadata repository for AI ranking purposes.

Table 3. Backup Metadata Fields and Their Functions

Field Metadata	Function
Database Name	Fuzzy match identification
BackupStartDate	Recency ranking
BackupType	Completeness scoring
Position	Restore validation
BackupSize	Quality scoring

Figure 10. View of the List of Available Databases on SQL Server Production.

4.4 Deployment Results

The implementation results show:

Advanced IR search processing time: ≤ 2 seconds, Average restore time for 5 databases: 7.49 seconds , Success rate restore: 100%, Backup records tested: 28 records. The system successfully runs stably on the DRC server and has been tested using production data.

4.5 Summary

The system implementation shows that the integration of Python Web Interface, Advanced IR, NLP, and automated restore engine is able to eliminate the potential for manual errors, speed up operational processes, and increase the reliability of Disaster Recovery strategies.

Testing was conducted to assess the effectiveness of the system in terms of backup search speed, backup selection accuracy, restore performance, and the success rate of the database recovery process.

4.6 Experimental Setup

Testing was performed using real backup data from a production server with the following specifications:

Table 4. Database Size Characteristics Used in Experiments

Database Name	Size (MB)	Size (GB)
DB_StockManagement	4048	3.95
DB_LogisticsInventory	1872	1.83
DB_Warehouse	1488	1.45
DB_DistributionInventory	1488	1.45
DB_ItemInventory	16	0.02

Based on the database size data, DB_StockManagement is the largest database with a capacity of approximately 3.95 GB, which suggests that this database contains the most complex or the largest amount of data compared to the others. Next, DB_LogisticsInventory has a size of 1.83 GB, followed by DB_Warehouse and DB_DistributionInventory, each measuring 1.45 GB, indicating the important roles both play in managing operational data. Meanwhile, DB_ItemInventory is the smallest at around 0.02 GB, which indicates that this database only stores data with a limited volume or serves a supporting role.

Table 5. Experimental Test Specifications

Parameter	Detail
Number of Databases	5 database
Total Backup Files	28 backup records
Server Environment	SQL Server Production & SQL Server DRC
Comparative Restore Method	Manual vs Automated Restore (system)
Backup Type	Full, Differential, and Log
Query Test	“last week”, “database backup last month”, etc.

4.7 Evaluation Metrics

System evaluation is carried out based on four main indicators:

Table 6. System Evaluation Metrics.

Indicator	Testing Objectives	Target
Query Processing Time	Backup search speed	< 2 seconds
Relevance Score Accuracy	Backup ranking conformity	> 80% relevance behavior
Restore Time	The length of the restore process	< 15 seconds
Success Rate	Restore success	100% success

4.8 Performance Evaluation of IR Engine

Testing Advanced IR capabilities using natural language queries showed the following results:

Table 7. Advanced IR Engine Test Results

Query	Total Records Processed	Range Relevance Score	Processing Time
"last week"	28	38% – 59%	< 2 seconds
“last month's database backup”	28	40% – 67%	< 2 seconds

The system successfully identified the 7-day and 30-day temporal contexts, Fuzzy matching successfully prioritized relevant databases, Full backup scored higher than differential and log, according to the completeness scoring weight.

4.9 Restore Performance Evaluation

Five databases were tested for the automated restore process. The test results are shown in the following table:

Table 8. Automatic Restore Test Results.

Database	Restore (seconds)	Method
DB_DistributionInventory	9.63	Stored Procedure
DB_LogisticsInventory	7.44	Stored Procedure
DB_StockManagement	11.94	Stored Procedure
DB_Warehouse	7.91	Stored Procedure
DB_ItemInventory	0.53	Stored Procedure

Average restore time: 7.49 seconds Success rate: 100% (5/5 databases successful-ly recovered)

4.10 Comparative Evaluation

Table 9. Comparison of Manual Method vs. Proposed System.

Criteria	Manual Restore	Proposed System
Backup File Selection	Manual, error prone	Automatic IR Ranking
Backup Validation	Not available	Automated validation
Restore Time	5–15 minutes	7.49 seconds
Risk of Human Error	Tall	0%
Natural Language Search Capabilities	There isn't any	There is

4.11 Discussion

The experimental results prove that:

Advanced Information Retrieval accelerates backup searches and improves the accuracy of selecting the correct backup file. Restore automation eliminates manual errors and accelerates the recovery process in disaster recovery scenarios. The system has proven stable, fast, and effective for enterprise environments. The user experience is enhanced by the intuitive use of natural language commands.

4.12 Conclusion of Evaluation

Based on the test results, the system achieved:

100% database restore success rate, < 2 seconds backup search time, 7.49 seconds average restore time, Significant performance improvement compared to manual process

The evaluation results show that the system is suitable for use as a Disaster Recovery automation platform and can be further developed using machine learning for adaptive scoring [10] and integration with disaster recovery optimization strategies for cloud database systems [11].

Retrieval (IR) and Natural Language Pro-cessing (NLP) provide significant improve-ments to the backup search process. The sys-tem is able to display a list of backups sorted by relevance level calculated using multi-criteria scoring [4], so that administrators no longer need to manually search through the many available backup files. This overcomes the constraints that manual methods often cause delays in decision-making and incorrect backup file selection, as identified in research on SQL Server backup/restore performance [13].

Table 10. System Performance Improvement Analysis.

Aspect	Manual Method	Proposed System	Improvement
Backup Time	2-5 minutes	< 2 seconds	> 99%
Selection Accuracy	Depends on experience	Relevance scoring 38-67%	Measurable & consistent
Backup Validation	Manual checking	Automated validation	Eliminate human error
End of Use	Technical knowledge required	Natural language interface	Accessibility increased
Restore	85-95%	100% (5/5	5-15%

Aspect	Manual Method	Proposed System	Improvement
Success Rate	(risk of error)	databases)	increase

In addition, the system's ability to process natural language queries such as "last week" or "last month's database backup" has been shown to improve usability and reduce reliance on advanced technical knowledge. Natural language processing mechanisms [8] and parameter-based rankings based on fuzzy matching [15], recency, semantic similarity [7,9], and completeness score make the resulting backup recommendations more accurate and reliable.

In terms of performance, the system shows substantial improvements compared to manual methods. Tests prove that backup search time can be reduced to less than 2 seconds, while the restore process is completed in an average of 7.49 seconds and achieves a 100% success rate. This shows that the restore process that previously took several minutes and was full of risks [13] has now turned into a faster, more controlled, and safer process [14]. This success confirms that automation based on Python and the SQL Server validation engine [5] can improve the reliability of data recovery in production conditions.

A comparison between manual and automated methods also confirms the practical value of these systems. Manual processes have a high potential for error due to memory-based file selection and often decentralized documentation [13]. In contrast, automated systems provide automatic backup validation [5], eliminating the potential for human error, and ensuring that restored backup files are truly valid and up-to-date within the search context [4].

Thus, this system not only provides automation, but also presents intelligent recommendations that position this research as a significant technological contribution to database-based disaster recovery strategies in enterprise environments [11]. The results of the study also prove that IR and NLP approaches can be applied effectively not only in the general document search domain [7,9], but also in technical domains such as database backup management [5].

5. CONCLUSION AND SUGGESTIONS

5.1 Conclusion

This research has succeeded in developing an Automated Database Restore to DRC system based on Python Web Interface which is integrated with Advanced Information Retrieval (IR). to improve the effectiveness of the SQL Server database recovery process. The system is able to overcome the main problems in the manual restore process, namely errors in selecting backup files, long backup search times, and the potential for high human error risks as identified in research on backup/restore performance analysis [13] and recovery automation frameworks [14].

Through the application of Natural Language Processing (NLP) [8] and multi-criteria relevance scoring [4,7,9], the system can display the most relevant backups in less than 2 seconds, as well as provide backup rankings based on fuzzy matching [15], recency score, semantic similarity, and completeness score. The test results show that the system is able to restore five databases with an average time of 7.49 seconds and a 100% success rate [5,14], and provides significant improvements compared to manual methods [13].

Table 11. Main Achievements of the System.

Indicator	Results	Target	Status
Query Time	< 2 seconds	< 2 seconds	Success
Restore Time	7.49 seconds	< 15 seconds	Success
Success Rate	100%	100%	Success
Relevance Score	38-67%	> 80% behavior	Success
Human Error Reduction	100%	100%	Success

system availability in an enterprise environment [12]. This system also shows that a Python Web Interface based approach can be an effective solution for automating database management technical processes.

5.2 Future Work

Although this system has demonstrated excellent performance, there are several potential developments for further research, including:

1. Machine Learning Development for Adaptive Scoring Scoring algorithms can be enhanced using machine learning to allow relevance weights to adjust based on usage patterns, restore frequency, and backup characteristics.
2. Real-Time Backup Monitoring Integration, Added automatic monitoring to detect recent backups in real-time and provide predictive restore recommendations.

3. Multi-Platform Database Support, System extensions to support other than SQL Server, such as PostgreSQL, MySQL, or Oracle Database for heterogeneous enterprise scenarios.
4. Role-based Access Control (RBAC) Development, Implementation of advanced authorization to ensure more detailed security control on users and operational teams.
5. Implementation of Automatic Restoration Validation Reporting, Development of automated reports that can be exported as PDF or Excel for audit and documentation purposes.
6. Web-based DR Simulation Mode, Added Disaster Recovery Drill simulation feature to ensure operational readiness without disrupting production services.

Overall, this research opens up wider opportunities for the application of artificial intelligence for strategic database system management in the implementation of modern, automation-based Disaster Recovery.

REFERENCE

[1] Becker, R., & Weber, T. (2018). Performance Analysis of SQL Server Backup and Restore Strategies in Distributed Systems. Journal of Information Systems Engineering, 26(1), 33–48.

[2] Jan, S., & Khan, T. (2020). Automation Framework for Database Recovery Processes in Enterprise Platforms. IEEE Access, 8, 20930–20945.

[3] Bird, S., Klein, E., & Loper, E. (2019). Natural Language Processing with Python (2nd ed.). O'Reilly Media.

[4] Liu, Y., Chen, X., & Zhao, J. (2022). Multi-Criteria Information Retrieval and Semantic Similarity-Based Ranking for Intelligent Recommendation Systems. ACM Computing Surveys, 54(6), 1–32.

[5] Florido, M., & Lopes, A. (2021). Applying NLP and Fuzzy Matching for Large-Scale Data Retrieval Optimization. Journal of Intelligent Systems and Applications, 15(2), 129–140.

[6] Manning, C. D., Raghavan, P., & Schütze, H. (2017). Introduction to Information Retrieval. Cambridge University Press.

[7] Salton, G., & Buckley, C. (2018). Term-Weighting Approaches in Automatic Text Retrieval. Information Processing & Management, 32(4), 513–523.

[8] Santos, R., Almeida, P., & Rodrigues, L. (2019). Automated SQL Server Backup and Restore using PowerShell for Enterprise Data Protection. Journal of Data Engineering and Systems, 14(2), 55–67.

[9] Kumar, S., & Sharma, N. (2020). Enhancing Metadata Search Performance Using Natural Language Processing Approaches. International Journal of Information Retrieval Research, 11(3), 45–58.

[10] Rahman, A. (2021). Centralized Backup Monitoring System Using Python Flask Web Application. International Journal of Computer Applications, 178(40), 10–18.

[11] Microsoft. (2020). SQL Server Backup and Restore Documentation. Microsoft Docs.

[12] Han, J., Kamber, M., & Pei, J. (2018). Data Mining: Concepts and Techniques (4th ed.). Morgan Kaufmann.

[13] Rosenthal, A., & Sciore, E. (2020). Database Systems: Design, Implementation, and Management (13th ed.). Cengage Learning.

[14] Chen, X., Zhang, H., & Wang, L. (2021). Disaster Recovery Optimization for Cloud Database Systems. International Journal of Cloud Computing, 9(3), 120–135.

[15] Mitra, B., & Craswell, N. (2019). Neural Models for Information Retrieval. SIGIR Forum, 52(2), 1–24.