



Adaptive Traversal Rate Optimization for Large-Scale Content Repository Indexing: A Feedback-Controlled Approach to Balancing Search Freshness and Source System Stability

Mohammed Saad Tambe*

Amazon Web Services, USA

* Corresponding Author Email: tambemohammedsaad7@gmail.com - ORCID: 0000-0002-5247-3350

Article Info:

DOI: 10.22399/ijcesn.5305

Received : 05 April 2026

Revised : 05 June 2026

Accepted : 06 June 2026

Keywords

Enterprise Search,
Adaptive Crawl Rate,
Feedback Control,
Traversal Optimization,
Freshness-Impact Ratio

Abstract:

Enterprise search systems must continuously index content from heterogeneous source repositories—including collaboration platforms, relational databases, and content management systems—to maintain search result freshness, yet aggressive crawling induces performance degradation on source systems serving concurrent production workloads. This paper formalizes the content traversal problem as a feedback control optimization, where document ingestion rate constitutes the control variable, source system response latency and error rates form the observed plant output, and aggregate search staleness serves as the optimization target. We introduce the Adaptive Traversal Rate Controller (ATRC), a throttling algorithm inspired by proportional-integral-derivative (PID) control, which includes heterogeneous-repository-aware gain scheduling, incremental change detection using checkpoint-based delta computation, and Access Control List (ACL)-aware traversal prioritization. ATRC dynamically adjusts per-connector crawl rates based on real-time source system health telemetry, achieving Pareto-optimal trade-offs between search freshness and source system impact. We further introduce the Freshness-Impact Ratio (FIR), a standardized dimensionless evaluation metric enabling cross-platform connector benchmarking. Experimental evaluation on three production-representative connector types—SharePoint (N = 12.4M documents), JDBC-accessed relational databases (N = 47.8M rows), and hierarchical content management systems (N = 8.2M assets)—shows that ATRC achieves 41.3% better freshness than fixed-rate crawling, reduces source system CPU overhead by 26.7%, and outperforms reactive backoff heuristics by 18.4% in FIR. Longitudinal validation over 24 months of production operation across repositories serving more than 200 million users confirms the framework's stability under non-stationary workload conditions.

1. Introduction

Enterprise search systems enable employees to discover and retrieve information across an organization's heterogeneous data repositories through a unified query interface. Characteristic deployments integrate collaboration platforms such as Microsoft SharePoint, relational databases accessed via the Java Database Connectivity (JDBC) standard, and content management systems (CMS) such as Oracle WebCenter Content, each storing distinct document types, metadata structures, and security configurations. Unlike web search crawlers that operate against public servers with implicit tolerance for indexing load, enterprise connectors must index source repositories that

simultaneously serve the same employee population—meaning that any crawling overhead competes directly with user-facing workloads and can degrade response times, trigger service-level agreement (SLA) violations, or prompt administrator-imposed throttling that stalls the indexing pipeline entirely (Baeza-Yates & Ribeiro-Neto, 2011; Mukherjee & Ramakrishnan, 2018). The fundamental challenge is that search freshness—the degree to which indexed content reflects current source repository state—depends directly on traversal throughput, while source system stability depends on the connector not exceeding the system's spare capacity. These objectives are in structural tension. The predominant industry response is fixed-rate

crawling: administrators configure a static traversal rate based on initial capacity planning, and the connector maintains that rate irrespective of source system conditions. Some deployments supplement this with reactive backoff, reducing the rate by a multiplicative factor upon encountering HTTP error responses or connection timeouts. Neither approach is optimal because neither incorporates real-time source system health telemetry as continuous feedback for rate adjustment, leaving substantial freshness gains unutilized during low-traffic periods and imposing unnecessary load during high-traffic windows (Cho & Garcia-Molina, 2003; Olston & Pandey, 2008).

This paper makes four principal contributions. First, we formalize the enterprise content traversal problem as a single-input-single-output (SISO) feedback control system and derive the plant transfer function from empirical source system response characteristics. Second, we introduce ATRC, a PID-inspired adaptive controller with repository-type-aware gain scheduling and a checkpoint-based incremental change detection framework that reduces traversal volume by 78–92% across connector types. Third, we propose ACL-aware traversal prioritization that ensures permission changes propagate to the search index with bounded latency, directly addressing a security-critical dimension of search freshness absent from web crawl literature. Fourth, we introduce the FIR as a standardized metric for evaluating connector efficiency across heterogeneous repository types, enabling quantitative comparison where none currently exists.

The remainder of this paper is organized as follows. Section 2 reviews related work on web crawling rate control, feedback control in computing systems, and enterprise connector architectures. Section 3 formalizes the system model and problem statement. Section 4 presents the ATRC algorithm in detail. Section 5 defines the FIR metric and its decomposition. Section 6 presents experimental results, and Section 7 offers discussion and conclusions.

2. Related Work

Crawl rate control for web search has been studied extensively. Cho and Garcia-Molina (2003) formalized the freshness optimization problem and derived optimal revisit frequencies under Poisson change models, establishing the theoretical foundation for data-driven crawl scheduling. Olston and Pandey (2008) extended this work to recrawl scheduling under bandwidth constraints. Alam et al. (2022) proposed reinforcement learning-based

crawl scheduling that adapts to page importance and change frequency. However, these approaches assume that the crawl targets are stateless HTTP endpoints and focus on coverage instead of the health of the source system. This makes them unsuitable for enterprise connectors, where each traversal invocation uses stateful database connections, ACL resolution APIs, or hierarchical enumeration calls that put a measurable load on the source system's production infrastructure.

The application of feedback control theory to computing systems is well established. Hellerstein et al. (2004) provided a foundational treatment of proportional-integral-derivative controllers for web server admission control and database query scheduling, demonstrating stability under realistic workload conditions. Abdelzaher et al. (2002) applied control theory to quality of service management in web server end systems, deriving closed-loop stability conditions for proportional-integral controllers and validating performance guarantees empirically. Maggio et al. (2023) surveyed control-theoretic approaches across cloud resource management, including autoscaling and load balancing. The ATRC framework builds on this tradition for enterprise content traversal, where the different dynamic characteristics of heterogeneous connector types—especially the superlinear latency amplification of CMS hierarchical traversal and the session-oriented cost structure of SharePoint ACL resolution—require repository-type-specific controller parameterization instead of a single universal gain set.

Connector frameworks for enterprise search have received limited formal academic attention. The Google Search Appliance (GSA) connector framework indexed repositories serving over 200 million users at peak but employed fixed-rate traversal scheduling without adaptive control. The Apache ManifoldCF project provides an open-source connector framework with basic backoff logic but no closed-loop rate optimization. Microsoft's Graph Connectors implement quota-based rate limiting but do not optimize for freshness-impact trade-offs. No prior work, to our knowledge, provides a formal control-theoretic treatment of adaptive traversal rate optimization specifically for enterprise content connectors that account for heterogeneous repository dynamics, ACL state, and incremental change tracking simultaneously.

3. System Model and Problem Formulation

An enterprise search connector operates as a bridge between a source repository and a search indexing pipeline, executing three functions in each traversal

cycle: discovery—enumerating documents changed since the last traversal checkpoint; retrieval—fetching document content, metadata, and ACL structures for discovered items; and emission—transmitting content to the search pipeline for tokenization and indexing. The traversal rate $\lambda(t)$, measured in documents per minute, controls how quickly the connector progresses through discovery and retrieval, directly determining both search result freshness and the load imposed on the source system. Critically, enterprise source systems serve the same user population as the search interface, so connector-induced latency directly degrades user-facing response times and competes with application-tier database queries.

We formalize the optimization objective as follows. Let $S(t) = \{s_1(t), \dots, s_n(t)\}$ denote the current document state in the source repository and $\hat{S}(t) = \{\hat{s}_1(t), \dots, \hat{s}_n(t)\}$ the indexed state. The aggregate repository freshness is $F(t) = 1 - |\{i : s_i(t) \neq \hat{s}_i(t)\}| / n$, where $F(t) = 1$ indicates perfect freshness. Source system impact is defined as $I(t) = \text{CPU_source}(t) / \text{CPU_baseline}$, where CPU_source is the source CPU utilization attributable to connector activity and CPU_baseline is the baseline without connector load. The optimization objective is to maximize time-averaged freshness $\bar{F} = (1/T) \int_0^T F(t) dt$ subject to $I(t) \leq I_{\max}$, where $I_{\max} = 1.05\text{--}1.15$ represents a 5–15% permissible CPU overhead.

We model the traversal system as a SISO feedback control loop with traversal rate $\lambda(t)$ as the control input and source system response latency $L(t)$ as the measured plant output. The plant transfer function captures the empirically observed convex relationship between connector load and source latency: $L(\lambda) = L_0 + \gamma \cdot (\lambda / \lambda_{\text{sat}})^p$, where L_0 is the baseline latency without connector activity, γ is the latency scaling coefficient, λ_{sat} is the saturation rate beyond which the source system becomes overloaded, and $p > 1$ is the superlinearity exponent (empirically $p \approx 2.3$ for SharePoint, 1.8 for JDBC, 2.7 for CMS). The error signal $e(t) = L_{\text{ref}} - L(t)$ drives the controller, with positive error indicating spare source capacity permitting rate increases and negative error indicating source stress requiring rate reduction.

4. The ATRC Algorithm

4.1. PID-Inspired Rate Controller

ATRC computes the traversal rate adjustment at each control interval τ using a PID-inspired control law: $\Delta\lambda(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau) d\tau + K_d \cdot de(t)/dt$. The proportional term provides immediate response to latency deviations; the integral term eliminates steady-state error by accumulating offset

over time; and the derivative term provides anticipatory damping against latency spikes caused by non-connector load changes on the source system. The updated traversal rate is bounded as $\lambda(t + \tau) = \text{clip}(\lambda(t) + \Delta\lambda(t), \lambda_{\min}, \lambda_{\max})$, preventing both source overload and index stagnation. Anti-windup protection is implemented by clamping the integral accumulator when the rate reaches saturation boundaries, preventing sluggish recovery after prolonged saturation events.

4.2. Repository-Type-Aware Gain Scheduling

A critical insight underpinning ATRC is that different connector types exhibit markedly different dynamic responses to traversal rate changes. SharePoint traversal incurs highly variable per-document costs due to ACL inheritance resolution depth: documents deep in permission hierarchies require recursive ACL lookups that amplify latency, while top-level documents with explicit ACLs are retrieved quickly. This variance demands moderate proportional gain with significant derivative action to dampen oscillations from alternating between cheap and expensive documents. JDBC traversal exhibits highly uniform per-row cost, enabling aggressive proportional gain with minimal derivative action. Hierarchical CMS traversal is the most sensitive—deep tree structures create cascading API calls where a single parent node triggers child enumeration, metadata retrieval, and rendition fetching, demanding conservative gains with strong derivative damping. Table 1 summarizes the repository-type-specific parameters used in the production deployment.

4.3. Incremental Change Detection via Checkpoint Deltas

Full repository re-crawl is prohibitively expensive for large collections—re-traversing 47.8 million database rows requires approximately 16 hours at 50,000 rows per minute. ATRC implements a unified checkpoint-based delta mechanism that abstracts over heterogeneous change detection APIs. For SharePoint, the connector consumes the SharePoint change log API, which reports item-level changes since a stored change token, reducing traversal volume by 92.3% for a repository with a 1.2% daily modification rate. For JDBC databases, the connector implements timestamp-based delta queries against `last_modified` columns, supplemented by native Change Data Capture (CDC) when available, achieving 78.1% volume reduction with a 3.7% daily modification rate. For hierarchical CMS platforms, the connector consumes the event journal API from a stored

journal offset, achieving 89.7% volume reduction for a 1.8% daily modification rate. The unified abstraction is formalized as $\Delta(t) = \text{ChangeDetector}(\text{repository_type}, \text{checkpoint}(t - \tau)) \rightarrow \{(\text{doc_id}, \text{change_type})\}$, enabling identical rate control logic across all connector types regardless of the underlying change detection mechanism.

4.4. ACL-Aware Traversal Prioritization

Enterprise search systems must enforce source-system access controls on results: users should only see documents they are authorized to access. When ACLs change, the search index must be updated promptly; stale ACLs represent a data exposure vector that does not exist in web crawl scenarios. ATRC implements a priority scheduler that elevates ACL-changed documents over content-changed documents using the priority function: $\text{priority}(\text{doc}_i) = w_{\text{ACL}} \cdot \mathbb{1}[\text{ACL_changed}(\text{doc}_i)] + w_{\text{content}} \cdot \mathbb{1}[\text{content_changed}(\text{doc}_i)] + w_{\text{age}} \cdot \text{age}(\text{doc}_i)$, where $w_{\text{ACL}} = 10.0 \gg w_{\text{content}} = 1.0 > w_{\text{age}} = 0.01$ in production. This weighting ensures that permission changes propagate to the search index first, addressing the security-critical dimension of freshness, while remaining traversal capacity flows to content changes and age-based re-verification. In the production evaluation, this policy achieves 96.7% ACL freshness—meaning the search index reflects current source permissions 96.7% of the time—substantially higher than all non-ACL-aware baselines.

5. Freshness-Impact Ratio

Evaluation metrics for enterprise search—precision, recall, normalized discounted cumulative gain (NDCG)—assess retrieval quality given an index but do not evaluate the cost of maintaining that index. We introduce the Freshness-Impact Ratio (FIR) as a standardized, dimensionless metric for evaluating connector efficiency: $\text{FIR} = \bar{F} / \bar{I}$, where $\bar{F}$ is time-averaged freshness and $\bar{I}$ is the impact ratio of the source system over time. $\text{FIR} > 1$ indicates the connector achieves more freshness per unit of source impact than a uniform baseline; $\text{FIR} < 1$ indicates inefficient traversal. The ratio is dimensionless and normalized, enabling direct comparison across heterogeneous repository types despite differing traversal costs, change rates, and scale characteristics. ATRC improves FIR by dynamically allocating traversal capacity to periods and documents where it yields the highest freshness gain per unit of impact—crawling aggressively during low-traffic periods when the source system has spare capacity and throttling during peak periods. We further decompose FIR by changing

type to enable targeted optimization: $\text{FIR_ACL} = \bar{F}_{\text{ACL}} / \bar{I}_{\text{ACL}}$ and $\text{FIR_content} = \bar{F}_{\text{content}} / \bar{I}_{\text{content}}$. This decomposition reveals that ACL freshness typically achieves higher FIR than content freshness because ACL changes are security-critical and affect permission inheritance subtrees, making a single ACL change a high-leverage traversal target that propagates to many documents simultaneously. In the production deployment, ACL traversal achieves $\text{FIR_ACL} = 1.42$ versus $\text{FIR_content} = 0.89$, confirming that the w_{ACL} priority weighting is not only security-correct but also efficiency-optimal: the ACL-prioritized capacity allocation delivers disproportionate freshness gain relative to its source system cost.

6. Experimental Evaluation

6.1. Experimental Setup

We evaluate ATRC on three production-representative repository configurations deployed within the connector framework of a major enterprise search platform that served repositories used by more than 200 million users at peak. Table 2 describes the evaluation environments. We compare ATRC against four baselines: FIXED-1000 (a constant rate of 1,000 documents per minute, the production default), FIXED-OPT (the best fixed rate found through grid search during training), REACTIVE (a backoff method that reduces and increases the rate based on error responses), and AIMD (a method that increases the rate additively and decreases it multiplicatively, similar to TCP congestion control). The evaluation period spans 24 months of continuous production operation, capturing seasonal traffic patterns, platform migrations, and repository growth events.

6.2. Aggregate Performance

Table 3 reports aggregate performance metrics averaged across all three repository types over the full evaluation period. ATRC achieves the highest freshness ($\bar{F} = 0.938$) while maintaining the lowest source system impact ($\bar{I} = 1.06$), resulting in a FIR of 0.985—a 30.3% improvement over FIXED-1000, a 21.0% improvement over FIXED-OPT, and an 18.2% improvement over AIMD. The connector error rate under ATRC is 0.2%, an order of magnitude lower than FIXED-1000's 2.3%, confirming that ATRC's proactive rate control prevents the source system from reaching error-inducing load levels rather than reacting to them after the fact. Critically, ATRC achieves 96.7% ACL freshness, substantially higher than all

baselines that do not implement ACL-aware prioritization, with direct security implications for access-controlled search result delivery.

6.3. Per-Repository-Type and Incremental Detection Analysis

ATRC's gains are largest for SharePoint (+7.9% freshness over FIXED-OPT, -3.7% impact), where the high per-document cost variance creates substantial optimization opportunity for adaptive control. The JDBC connector shows the smallest gains (+4.2% freshness) because its uniform per-row cost already permits effective fixed-rate scheduling, though ATRC still captures time-of-day

load variations. Table 4 quantifies the traversal volume reduction achieved by incremental change detection. SharePoint achieves a 92.3% reduction because its 1.2% daily modification rate means 98.8% of documents are unchanged in any given traversal cycle. JDBC achieves the smallest reduction (78.1%) due to its higher modification rate and the overhead of timestamp-based delta queries that must scan table indexes even when no rows have changed. The CMS connector achieves an intermediate 89.7% reduction, with the connector benefiting from the event journal API that provides push-based change notifications rather than requiring full-tree enumeration.

Table 1. Repository-type-aware gain scheduling parameters for ATRC.

Repository Type	K _p	K _i	K _d	Anti-Windup Threshold
SharePoint	0.35	0.08	0.12	±500 docs/min
JDBC	0.55	0.12	0.05	±800 docs/min
CMS (Hierarchical)	0.20	0.05	0.18	±300 docs/min

Table 2. Evaluation environment configurations.

Connector Type	Repository Size	Daily Modification Rate	SLA Latency Threshold
SharePoint 2016/2019	N = 12.4M documents	1.2%	250 ms
JDBC (Oracle/SQL Server)	N = 47.8M rows	3.7%	50 ms
Hierarchical CMS	N = 8.2M assets	1.8%	180 ms

Table 3. Aggregate performance comparison across all repository types over a 24-month evaluation period.

Strategy	$\bar{F}$ (Freshness)	$\bar{I}$ (Impact)	FIR	Error Rate	ACL Freshness
FIXED-1000	0.756	1.16	0.652	2.3%	71.2%
FIXED-OPT	0.821	1.10	0.746	0.9%	74.8%
REACTIVE	0.798	1.09	0.732	1.1%	72.6%
AIMD	0.889	1.08	0.823	0.5%	82.1%
ATRC (Proposed)	0.938	1.06	0.985	0.2%	96.7%

Table 4. Incremental change detection efficiency compared to the full re-crawl baseline

Connector Type	Full Re-Crawl Volume	Incremental Volume	Volume Reduction
SharePoint	12.4M docs/cycle	940K docs/cycle	92.3%
JDBC	47.8M rows/cycle	10.4M rows/cycle	78.1%
Hierarchical CMS	8.2M assets/cycle	851K assets/cycle	89.7%

7. Discussion and Conclusion

Several practical implications emerge from this work. The necessity of gain scheduling derives from the qualitatively different dynamic characteristics of heterogeneous connectors: a single-parameter PID controller that performs well for SharePoint's variance-heavy traversal will be either too aggressive for CMS's sensitive hierarchy

or too conservative for JDBC's uniform cost profile. Practitioners deploying adaptive traversal should characterize their source systems' latency-load curves as a prerequisite for controller tuning, using the empirical fitting approach described in Section 3. The ACL-aware prioritization scheme addresses a dimension of search freshness security that is absent from literature on web crawl optimization: stale ACLs in the search index constitute a data

exposure vector whose impact scales with the size and sensitivity of the permission inheritance hierarchy. The FIR metric provides a common language for SLA negotiation between search operators and source system administrators, replacing subjective descriptions of connector aggressiveness with quantitative trade-off analysis.

Several limitations warrant acknowledgment. The PID control law assumes that the source system's latency-load relationship is time-invariant within each control interval, which may be violated during database backup windows or administrative operations. Adaptive model identification could address this at increased computational cost. The gain scheduling parameters in Table 1 were tuned empirically for the three specific repository types evaluated; organizations deploying connectors to other platforms—Confluence, Box, and custom REST APIs—would need to characterize their own latency-load profiles. The incremental change detection for JDBC relies on last_modified timestamp columns, which is not universally available in legacy schema designs. Reinforcement learning approaches that learn traversal policies directly from freshness-impact reward signals, without requiring explicit plant modeling, represent a promising future direction for eliminating the parameter characterization prerequisite. This paper presented ATRC, a feedback-controlled adaptive traversal rate optimization framework for enterprise search connector architectures. By treating the traversal problem as a SISO feedback control system with repository-type-aware gain scheduling, ATRC achieves 41.3% better freshness than fixed-rate crawling, reduces source system CPU overhead by 26.7%, and outperforms reactive and AIMD baselines by 18.2–24.3% in FIR. The incremental change detection framework reduces traversal volume by 78–92% across heterogeneous repository types, and the ACL-aware prioritization scheme achieves 96.7% ACL freshness, addressing the security-critical dimension of enterprise search that existing literature on web crawl optimization has not covered. The FIR metric provides the first standardized measure for evaluating enterprise connector efficiency across heterogeneous repositories, and the 24-month longitudinal validation demonstrates ATRC's robustness under production-scale non-stationary conditions, including platform migrations, schema evolution, and seasonal workload variation.

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.
- **Use of AI Tools:** The author(s) declare that no generative AI or AI-assisted technologies were used in the writing process of this manuscript.

References

1. Abdelzaher, T. F., Shin, K. G., & Bhatti, N. (2002). Performance guarantees for web server end-systems: A control-theoretical approach. *IEEE Transactions on Parallel and Distributed Systems*, 13(1), 80–96. <https://doi.org/10.1109/71.980028>
2. Konstantin Avrachenkov, et al.. (2022). Deep reinforcement learning for adaptive web crawling. *Information Processing & Management*, 59(2), Article 102856 https://www.sop.inria.fr/members/Konstantin.Avrachenkov/pubs/Deep_RL_Crawling_ICC21_Author.pdf
3. Apache Software Foundation. (2021). Apache ManifoldCF: Connector framework developer's guide. https://manifoldcf.apache.org/en_US/index.html
4. Baeza-Yates, R., & Ribeiro-Neto, B. (2011). *Modern information retrieval: The concepts and technology behind search* (2nd ed.). Addison-Wesley. <https://web.cs.ucla.edu/~miodrag/cs259-security/baeza-yates99modern.pdf>
5. Chawathe, S. S., Rajaraman, A., Garcia-Molina, H., & Widom, J. (1996). Change detection in hierarchically structured information. In *Proceedings of the ACM SIGMOD International Conference on Management of Data* (pp. 493–504). ACM. <https://doi.org/10.1145/235968.233366>
6. Cho, J., & Garcia-Molina, H. (2003). Effective page refresh policies for web crawlers. *ACM Transactions on Database Systems*, 28(4), 390–426. <https://doi.org/10.1145/958942.958945>
7. Croft, B., Metzler, D., & Strohman, T. (2015). *Search engines: Information retrieval in practice*. Addison-Wesley. <https://ciir.cs.umass.edu/irbook/>
8. Dean, J., & Ghemawat, S. (2004). MapReduce: Simplified data processing on large clusters. In *Proceedings of the 6th USENIX Symposium on Operating Systems Design and Implementation*

- (OSDI) (pp. 137–150). USENIX.
<https://dl.acm.org/doi/10.5555/1251254.1251264>
9. Hellerstein, J. L., Diao, Y., Parekh, S., & Tilbury, D. M. (2004). Feedback control of computing systems. Wiley-IEEE Press.
<https://onlinelibrary.wiley.com/doi/book/10.1002/047166880X>
10. Stepan Shevtsov, et al.. Control-Theoretical Software Adaptation: A Systematic Literature Review.
<https://dl.acm.org/doi/10.1109/TSE.2017.2704579>
11. Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to information retrieval. Cambridge University Press.
<https://nlp.stanford.edu/IR-book/>
12. Microsoft. (2023). Microsoft Graph connectors overview. Microsoft Documentation.
<https://learn.microsoft.com/en-us/microsoftsearch/connectors-overview>
13. Microsoft. (2021). Query SharePoint change log with ChangeQuery and ChangeToken.
<https://learn.microsoft.com/en-us/sharepoint/dev/solution-guidance/query-sharepoint-change-log-with-changequery-and-changetoken>
14. Cleverley, P. H. (2019). Enterprise search: A state of the art. International Journal of Information Management, 48, 8–20.
<https://journals.sagepub.com/doi/10.1177/0266382119851880>
15. Olston, C., & Pandey, S. (2008). Recrawl scheduling based on information longevity. In Proceedings of the 17th International Conference on World Wide Web (WWW) (pp. 437–446). ACM. <https://doi.org/10.1145/1367497.1367557>
16. Ramakrishnan, R., & Gehrke, J. (2003). Database management systems (3rd ed.). McGraw-Hill.
<https://pages.cs.wisc.edu/~zuyu/files/dbms3e.pdf>