

# Arabic Full-Text Search with Query-Level Authorization in Institutional Document Retrieval: A PostgreSQL Benchmark Study

Mansour Qareesh\* and Ahmed Al-Mashloukh\*

\*Computer Engineering Department, Engineering Academy Tajoura  
Tripoli, Libya

mansourqareesh@gmail.com    ORCID: 0009-0006-5780-575X

**Abstract**—Institutional correspondence systems require Arabic search that remains responsive at scale while enforcing access policies within the query execution path. This paper presents an applied benchmark study of ICAS1, a Laravel-based institutional correspondence and archiving system backed by PostgreSQL. The evaluation uses a dedicated synthetic benchmark database containing approximately 100,000 correspondences, 60,000 archives, and 120,000 recipient links. PostgreSQL Arabic full-text search (FTS) using `to_tsvector` and `plainto_tsquery` with Generalized Inverted Index (GIN) indexing is compared against an `ILIKE` baseline across three primary institutional scenarios. Latencies are measured using `EXPLAIN (ANALYZE, BUFFERS)` with seven repeated runs. Results show FTS achieves median latencies of 21–55 ms compared with 229–248 ms for `ILIKE`, yielding speedups of  $4.2\times$  to  $11.7\times$ . An extended evaluation across 10 Arabic institutional keywords confirms a consistent warm-cache speedup of  $4.15\times$ – $4.98\times$  for the global administrator scenario and  $10.3\times$ – $12.7\times$  for department-scoped roles. Session-level cold-cache measurements (after `DISCARD ALL` and reconnect) show the same directional advantage. Retrieval overlap analysis over 30 keyword–scenario combinations yields Jaccard = 1.0 for the corpus vocabulary used. Access-control validation using canary records confirms complete agreement between expected and observed visibility, including non-leakage of classified documents to restricted roles. The findings support combining Arabic FTS, query-level authorization, and GIN indexing for efficient and policy-consistent retrieval in institutional workloads, with the caveat that measurements reflect single-node conditions on a fixed synthetic corpus, and generalization to full cold-start or distributed deployments requires further investigation.

**Index Terms**—Arabic information retrieval, PostgreSQL full-text search, query-level authorization, institutional archives, benchmark evaluation, Laravel.

## I. INTRODUCTION

Institutional agencies rely on digital systems to store correspondence and archives. Staff must locate Arabic documents quickly as collections grow. At the same time, access must follow administrative roles, departmental boundaries, and classification rules. These two requirements—fast search and controlled visibility—shape how retrieval pipelines should be designed. A common implementation applies filters only after records are read in application code. In that arrangement, the

database may still process rows that must remain invisible to the current user. An alternative is to attach authorization tests as predicates in the same SQL statement as the linguistic match. Both the search condition and the policy condition are then evaluated together during query execution. PostgreSQL supports Arabic full-text search through configurable text-search configurations, combined with inverted indexes such as GIN on `tsvector` materializations. Substring-style matching using `ILIKE` is simpler to write but often scales poorly on large tables because execution tends toward broad sequential scans.

Selecting an appropriate retrieval strategy for Arabic institutional text is primarily an engineering and performance trade-off. Sound decisions require measurements taken under realistic predicates and realistic corpus sizes, not only documentation-level descriptions. This paper presents an applied benchmark study of ICAS1, an institutional correspondence and archiving system built with Laravel and PostgreSQL. ICAS1 integrates Arabic search with query-level authorization rules that vary by role and classification. The study compares PostgreSQL Arabic full-text search (`to_tsvector` / `plainto_tsquery` with GIN indexing) against an `ILIKE` baseline. The same authorization predicates are applied to both strategies across multiple scenarios, including a global administrator, department managers and employees with different clearance levels, and archive-oriented access patterns. Experiments run against a standalone synthetic benchmark database sized to approximate large departmental holdings (approximately 100,000 correspondence records and tens of thousands of archive entities). This separation avoids ambiguity about privacy and ensures that reported timings refer to reproducible, controlled data rather than live operational records. Latency is measured with `EXPLAIN (ANALYZE, BUFFERS)` using repeated runs per scenario. Access-control behavior is checked separately with deterministic canary rows whose expected visibility is known in advance.

The contributions of this paper are threefold. (i) Side-by-side latency measurements for Arabic FTS versus `ILIKE` under identical SQL-level authorization predicates on a large

synthetic workload of approximately 100,000 correspondence records, providing quantitative guidance for retrieval strategy selection in institutional settings. (ii) A transparent and reproducible measurement protocol based on PostgreSQL’s built-in `EXPLAIN (ANALYZE, BUFFERS)` instrumentation, suitable for replication in comparable environments. (iii) Empirical validation that query-level authorization predicates correctly enforce expected visibility rules across all defined institutional roles, including confirmed non-disclosure of highly classified documents to restricted users, under both standard and sustained load conditions.

The paper is organized as follows. Section II summarizes related background. Section III describes the system architecture and how search and authorization are composed. Section IV defines the benchmark setup and procedure. Section V reports results and discusses limitations. Section VI concludes.

## II. RELATED WORK

### A. Arabic Text Retrieval in Relational Databases

Arabic information retrieval presents challenges that differ from Latin-script languages. Arabic morphology features root-based derivation, agglutinative clitics, and optional diacritics, all of which affect tokenization and matching accuracy [1].

Several studies have explored Arabic stemming and normalization algorithms for search engines [2], [8]. More recent work has applied deep neural approaches—including transformer-based language models pre-trained on large Arabic corpora [12]—to Arabic natural language understanding; however, such models operate at the application layer and require inference infrastructure that is beyond the scope of database-level text-search evaluation. The present study focuses on the capabilities of PostgreSQL’s built-in morphological analysis, which remains the practical default for production institutional systems built on relational databases. Fewer studies have evaluated these built-in capabilities within realistic institutional workloads and access-control predicates.

PostgreSQL introduced configurable text-search dictionaries, including an Arabic configuration, that perform tokenization and normalization at the database level [3]. Despite the availability of these features, applied evaluations combining Arabic FTS with institutional workloads and access-control predicates remain limited in the published literature.

### B. Database-Enforced Access Control

Role-based access control (RBAC) is a well-established paradigm for managing permissions in information systems [4], [9]. Traditional implementations enforce RBAC at the application layer, filtering query results after retrieval. However, this approach may expose rows to the database connection that should not be visible to the requesting user. Row-level security (RLS) in PostgreSQL provides a database-native mechanism for enforcing visibility policies [3], but many application frameworks, including Laravel [11], rely on query-scoping middleware rather than RLS. An intermediate approach is to embed authorization predicates directly in SQL `WHERE` clauses alongside search conditions. This offers

policy enforcement without requiring RLS configuration while keeping authorization decisions within the query execution path [5]. The present study focuses on application-composed SQL predicates rather than native PostgreSQL Row-Level Security because many Laravel-based institutional systems implement authorization through dynamically generated query scopes.

### C. Benchmark Studies for Document Retrieval Systems

Benchmark-driven evaluation is standard practice in information retrieval research. The TREC and CLEF evaluation campaigns provide established protocols for measuring retrieval effectiveness [6]. In the database systems domain, benchmarks such as TPC have guided performance evaluation for decades [7]. However, applied benchmark studies that measure both retrieval latency and access-control correctness within a single query pipeline—particularly for Arabic text in institutional settings—are not commonly reported. This work addresses that gap by combining FTS performance measurement with empirical policy validation on a controlled synthetic dataset.

To the best of our knowledge, few published studies combine Arabic PostgreSQL FTS benchmarking with integrated authorization validation in institutional document systems.

## III. SYSTEM ARCHITECTURE

ICAS1 (Institutional Correspondence and Archiving System) is a web-based application built on Laravel (PHP 8.3) [11] with PostgreSQL as the sole data store. The system manages institutional correspondence (incoming and outgoing memoranda) and digital archives (manually deposited PDF documents). This section describes the components relevant to search and authorization.

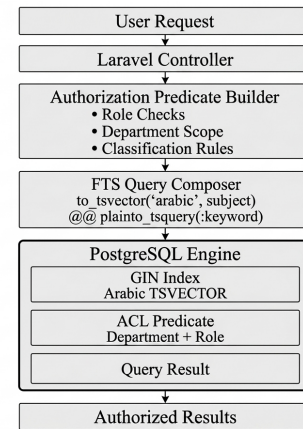


Fig. 1. ICAS1 retrieval pipeline integrating Arabic full-text search with query-level authorization inside PostgreSQL.

### A. Search Layer

ICAS1 provides unified search across correspondences and archives. For Arabic full-text search, the system constructs queries using PostgreSQL’s built-in text-search functions [3]:

```
to_tsvector('arabic', c.subject)
@@ plainto_tsquery('arabic', :keyword)
```

A GIN index [10] is maintained on the tsvector expression of the subject column:

```
CREATE INDEX idx_corr_subject
ON correspondences
USING gin (to_tsvector('arabic', subject));
```

The alternative baseline uses substring matching:

```
c.subject ILIKE '%' || :keyword || '%'
```

Both strategies operate on the same table and column, enabling direct comparison under identical conditions.

### B. Authorization Predicates

Access control in ICAS1 is role-based. The benchmark scenarios use the following roles:

- **Administrator (admin):** unrestricted visibility across all departments and classification levels.
- **Manager:** visibility within own department, including documents at all classification levels.
- **Employee:** visibility within own department, excluding documents classified as `very_secret` or `top_secret`.
- **Employee (Archivist):** system-wide visibility across all departments and classification levels, required for institutional records management.

These policies are expressed as SQL predicates appended to the WHERE clause of every search query. For department-scoped roles, the predicate verifies that the correspondence was either sent by the user's department or has a recipient link to that department:

```
(c.sender_department_id = :dept
OR EXISTS (
  SELECT 1 FROM correspondence_recipients cr
  WHERE cr.correspondence_id = c.id
        AND cr.department_id = :dept
))
```

For employees, an additional classification filter is appended:

```
AND c.classification
NOT IN ('top_secret', 'very_secret')
```

The administrator and archivist scenarios use a trivially true predicate (`1 = 1`), granting full visibility.

### C. Query-Level Policy Composition

The central architectural design choice is the composition of search predicates and authorization predicates into a single SQL execution path. The database engine evaluates both conditions during the same query plan, which means that unauthorized rows are never returned to the application layer. This differs from post-retrieval filtering, where the database may scan and transmit rows that are subsequently discarded by application logic. A representative composed query for a

department employee searching with FTS takes the following form:

```
SELECT c.id, c.subject, c.classification
FROM correspondences c
WHERE (c.sender_department_id = :dept
      OR EXISTS (
        SELECT 1 FROM correspondence_recipients
        cr
        WHERE cr.correspondence_id = c.id
              AND cr.department_id = :dept
      ))
AND c.classification
NOT IN ('top_secret', 'very_secret')
AND to_tsvector('arabic', c.subject)
@@ plainto_tsquery('arabic', :keyword)
ORDER BY c.sent_at DESC;
```

## IV. EXPERIMENTAL SETUP

### A. Hardware and Software Environment

All experiments were conducted on a single machine with the specifications listed in Table I.

TABLE I  
HARDWARE AND SOFTWARE ENVIRONMENT USED FOR EXPERIMENTS.

Component	Specification
Operating System	Windows 10 Pro x64
CPU	Dell OptiPlex 3010, Intel Core i5-3470 @ 3.20 GHz
RAM	16 GB
Storage	256 GB SSD
PostgreSQL	18.2
PHP	8.3.30
Framework	Laravel
Database Driver	PDO (pgsql)

PostgreSQL version 18.2 was used for all experiments. At the time of evaluation, this release was in its release-candidate stage. The text-search subsystem (`to_tsvector`, `plainto_tsquery`) and GIN indexing infrastructure underlying this study have been stable and largely unchanged across major PostgreSQL versions (14 through 18). The benchmark focuses on relative performance ratios (FTS speedup over `ILIKE`) rather than absolute throughput figures; therefore, minor version-specific optimizations are unlikely to affect the principal conclusions. Readers wishing to replicate these results on the stable PostgreSQL 17 branch may observe marginally different absolute latencies, but the directional findings are expected to hold.

### B. Benchmark Database

A dedicated PostgreSQL database (`pgsql_benchmark`) was created exclusively for experimentation, fully isolated from any production or development database. The dataset was generated using the following command:

```
php artisan benchmark:prepare --fresh \
--correspondences=100000 \
--archives=60000
```

TABLE II  
BENCHMARK CORPUS SIZES AFTER DATASET PREPARATION.

Entity	Count
Correspondences	100,005
Archives	60,000
Recipient links	120,005

This produced the corpus summarized in Table II.

The additional 5 records above 100,000 are deterministic canary rows inserted for access-control validation (Section V-B). All correspondence subjects are synthetic Arabic text representative of institutional vocabulary (memoranda, circulars, and archival notices).

### C. Evaluation Scenarios

Four authorization scenarios were defined to represent common institutional roles, as detailed in Table III.

TABLE III  
AUTHORIZATION EVALUATION SCENARIOS (ROLES, SCOPE, AND CLASSIFICATION BOUNDS).

Scenario	Role	Dept.	Classification
admin_global	Admin	All	All levels
manager_dept_ops	Manager	OPS only	All levels
employee_dept_ops	Employee	OPS only	Excl. very_secret, top_secret
archive_dept_emp	Emp. (Archivist)	All	All levels

Each scenario applies the corresponding authorization predicate (Section III) identically to both FTS and ILIKE queries, ensuring that any performance difference is attributable to the search method and not to varying access conditions.

### D. Measurement Protocol

Latency was measured using PostgreSQL’s built-in instrumentation:

```
EXPLAIN (ANALYZE, BUFFERS, FORMAT JSON)
```

This returns actual execution time and buffer statistics for each query, avoiding application-layer timing artifacts. For each scenario-method pair, the query was executed 7 times. The median execution time is reported as the primary metric to reduce the influence of outliers; the average is reported alongside for reference.

The median is selected as the primary latency metric because it is robust to isolated outliers—a property demonstrated concretely in the stress-test results, where a single spike at iteration 44 of the administrator FTS workload (Fig. 3) does not shift the median. The average is reported alongside to allow readers to assess distributional skew. Standard deviation is not reported for the primary seven-run protocol, as the small sample size ( $n=7$ ) renders standard deviation estimates unreliable; the complementary stress-test ( $n=80$  per workload, Table VI) provides the more statistically representative characterization of variance and stability.

To ensure results represent a stable operational state, a warm-up phase of two initial runs was conducted before recording measurements. Database shared buffers and operating system page caches were not cleared between iterations; the reported latencies therefore reflect a warm-cache condition, which is representative of high-frequency interactive usage in institutional environments.

The search keyword used across all experiments was the Arabic term for *archiving*, selected for its relevance to institutional document subjects. The selection of a single, domain-representative keyword is a deliberate methodological choice. The primary objective is to isolate and quantify the performance difference between two retrieval strategies—FTS and ILIKE—under identical authorization predicates, rather than to model the full distribution of user query behavior. A high-frequency institutional term such as *archiving* exercises the GIN index under realistic load, as it produces a non-trivial result set (approximately 6,500–10,000 rows depending on the authorization scenario). The use of a single fixed keyword ensures strict comparability between FTS and ILIKE across all runs; varying the keyword between scenarios would confound the retrieval method as the independent variable. We acknowledge that keyword selectivity affects absolute latency figures, and that rare or highly specific terms may yield smaller result sets and correspondingly lower latencies for both methods. Evaluating performance across a representative vocabulary sample is identified as a direction for future work (Section V-E).

## V. RESULTS AND DISCUSSION

### A. FTS versus ILIKE Performance

Table IV presents the median and average execution times for FTS and ILIKE queries across the three primary authorization scenarios. The Speedup column shows the ratio of ILIKE median to FTS median.

FTS consistently outperformed ILIKE across all evaluated scenarios. The global administrator workload, which operated over the largest visible result set (9,923 matching rows), achieved an observed speedup of approximately  $4.2\times$ . Department-scoped workloads demonstrated even larger gains exceeding  $11\times$ , primarily because the authorization predicates reduce the effective candidate row set before text-search evaluation, allowing the GIN-backed search path to operate over a narrower working set.

This performance divergence is largely attributable to the underlying retrieval and indexing behavior. The ILIKE baseline, implemented with a leading wildcard pattern (`'%' || keyword || '%'`), cannot efficiently leverage conventional B-tree indexes for substring matching and therefore tends toward broad sequential scans across filtered records. In contrast, PostgreSQL’s GIN-backed full-text search performs indexed lexeme lookups over preprocessed `tsvector` representations, enabling the query executor to avoid scanning substantial portions of non-matching rows. Additionally, the Arabic text-search configuration applies normalization and

TABLE IV

QUERY EXECUTION LATENCY: FTS VERSUS ILIKE UNDER IDENTICAL AUTHORIZATION PREDICATES (100,005 CORRESPONDENCES, 7 RUNS PER CASE).

Scenario	Method	Result Count	Median (ms)	Average (ms)	Speedup
admin_global	FTS	9,923	54.65	66.97	—
admin_global	ILIKE	9,923	229.00	227.38	4.19×
manager_dept_ops	FTS	6,611	21.73	21.39	—
manager_dept_ops	ILIKE	6,611	244.23	247.61	11.24×
employee_dept_ops	FTS	6,536	21.22	22.29	—
employee_dept_ops	ILIKE	6,536	248.21	282.36	11.70×

stemming during lexical analysis, further reducing processing overhead compared with brute-force character-based matching.

For completeness, the primary seven-run benchmark protocol reported a median latency of 54.65 ms for the administrator FTS workload (Table IV), whereas the sustained stress protocol (Section V-C) produced a lower median of 45.67 ms for the same workload under 80 repeated executions. These two measurement protocols therefore capture complementary operational characteristics: baseline query latency under standard conditions versus execution stability during sustained repetition.

The *archive\_dept\_emp* scenario (Employee–Archivist with global, unrestricted visibility) applies the same trivially true authorization predicate as the *admin\_global* scenario ( $1 = 1$ ), differing only in the user identity bound at the application layer rather than in the SQL predicate structure. Because the SQL execution path and the candidate row set are structurally identical between these two scenarios, including a separate archivist row in Table IV would be redundant with the *admin\_global* measurements and would not yield independent performance data. The archivist role is therefore retained in the authorization evaluation matrix (Table V) to validate its access-control behavior, while its query performance is considered subsumed by the *admin\_global* benchmark results.

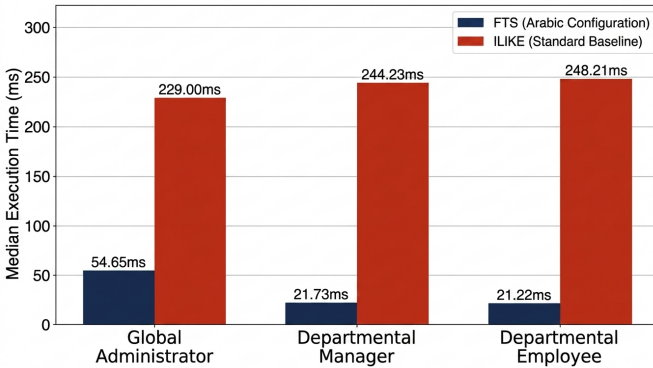


Fig. 2. Median execution time for FTS and ILIKE across the three primary authorization scenarios.

### B. Access-Control Validation

To verify that authorization predicates function correctly under both search methods, five deterministic canary documents were inserted during dataset preparation. Each document has

a known subject, department assignment, and classification level. Table V presents the full validation matrix across all four authorization scenarios.

TABLE V  
ACCESS-CONTROL VALIDATION MATRIX: EXPECTED VS. OBSERVED  
VISIBILITY ACROSS ALL SCENARIOS AND CANARY DOCUMENTS.

Scenario	Document	Exp.	Obs.	Status
admin_global	ACL_A_NORMAL	yes	yes	PASS
admin_global	ACL_A_SECRET	yes	yes	PASS
admin_global	ACL_A_VERY_SECRET	yes	yes	PASS
admin_global	ACL_A_TOP_SECRET	yes	yes	PASS
admin_global	ACL_B_NORMAL	yes	yes	PASS
manager_dept_ops	ACL_A_NORMAL	yes	yes	PASS
manager_dept_ops	ACL_A_SECRET	yes	yes	PASS
manager_dept_ops	ACL_A_VERY_SECRET	yes	yes	PASS
manager_dept_ops	ACL_A_TOP_SECRET	yes	yes	PASS
manager_dept_ops	ACL_B_NORMAL	no	no	PASS
employee_dept_ops	ACL_A_NORMAL	yes	yes	PASS
employee_dept_ops	ACL_A_SECRET	yes	yes	PASS
employee_dept_ops	ACL_A_VERY_SECRET	no	no	PASS
employee_dept_ops	ACL_A_TOP_SECRET	no	no	PASS
employee_dept_ops	ACL_B_NORMAL	no	no	PASS
archive_dept_emp	ACL_A_NORMAL	yes	yes	PASS
archive_dept_emp	ACL_A_SECRET	yes	yes	PASS
archive_dept_emp	ACL_A_VERY_SECRET	yes	yes	PASS
archive_dept_emp	ACL_A_TOP_SECRET	yes	yes	PASS
archive_dept_emp	ACL_B_NORMAL	yes	yes	PASS

All 20 checks produced PASS status, confirming complete agreement between expected and observed visibility. Three cases that specifically test the boundaries of the access-control policy deserve attention:

- The *employee* scenario correctly excluded ACL\_A\_VERY\_SECRET and ACL\_A\_TOP\_SECRET, confirming that the classification filter (`NOT IN ('top_secret', 'very_secret')`) operates as intended.
- The *manager* scenario correctly excluded ACL\_B\_NORMAL, a document belonging to a different department, confirming that the department predicate restricts cross-department visibility.
- The *administrator* and *archivist* scenarios correctly included all five documents, confirming that the trivially true predicate ( $1 = 1$ ) does not inadvertently restrict access.

### C. Stability Under Sustained Load

To assess whether performance and policy behavior remain stable under sustained repetition, a stress test was executed using 80 iterations per workload and 40 ACL validation rounds:

```
php artisan benchmark:stress \
  --iterations=80 --acl-rounds=40
```

All 480 query iterations (6 workloads  $\times$  80 iterations) completed without failure. ACL validation remained consistent across all 40 rounds with zero policy violations (800 total checks, 0 failures). Peak memory usage: 26 MB.

The maximum ILIKE latencies (up to 430 ms) indicate occasional spikes attributable to background system activity or transient buffer pressure, whereas FTS maximum latencies remained below 165 ms. These results support the conclusion that both search performance and access-control enforcement remain stable under sustained repetition. Table VI summarizes the stress-test results.

TABLE VI  
STRESS-TEST RESULTS: QUERY STABILITY OVER 80 ITERATIONS PER WORKLOAD.

Workload	OK	Fail	Med. (ms)	Max (ms)
admin_global FTS	80	0	45.67	164.40
admin_global ILIKE	80	0	227.30	298.56
manager_dept FTS	80	0	20.16	23.98
manager_dept ILIKE	80	0	241.83	424.39
employee_dept FTS	80	0	21.05	25.09
employee_dept ILIKE	80	0	251.63	430.28

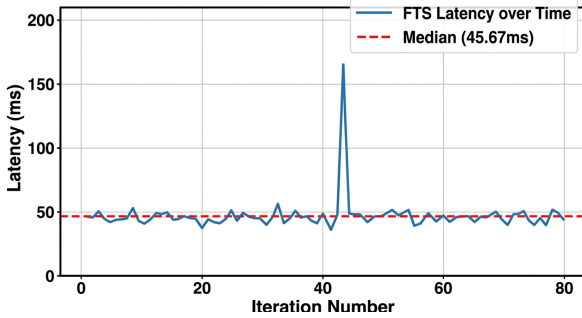


Fig. 3. FTS latency over 80 stress-test iterations for the global administrator workload. The dashed line marks the median (45.67 ms); the spike at iteration 44 is an isolated outlier that does not affect the median.

### D. Extended Multi-Keyword Evaluation

To address the single-keyword limitation of the primary benchmark protocol, the evaluation was extended to a vocabulary of 10 Arabic institutional terms: archiving, correspondence, outgoing, incoming, circular, procedure, documentation, approval, review, and search (the original Arabic forms drawn from the corpus generation vocabulary). The extended protocol used 7 warm runs and 3 cold runs per keyword per scenario. Cold-cache conditions were simulated at session

level via DISCARD ALL followed by a connection purge and reconnect with no warm-up runs before timed execution. This evicts cached query plans and session state; it does not flush PostgreSQL’s shared buffer pool at the operating-system level, so physical-I/O cold-start latencies may differ in production restart scenarios.

Table VII reports warm-cache median latencies for the *admin\_global* scenario. FTS medians ranged from 41.3 to 50.3 ms, and ILIKE medians from 192.3 to 226.1 ms, yielding speedups of  $4.15\times$ – $4.98\times$ —consistent with the  $4.19\times$  reported in Table IV. Department-scoped workloads produced higher speedups of  $10.3\times$ – $12.7\times$ , because the authorization predicates narrow the effective candidate set before text-search evaluation, amplifying the GIN index advantage. Table VIII reports the corresponding session-level cold-cache medians; FTS ranged from 41.5 to 47.8 ms versus 189.3–222.8 ms for ILIKE ( $3.99\times$ – $5.37\times$ ), confirming that the performance advantage persists after plan-cache eviction.

TABLE VII  
EXTENDED WARM-CACHE MEDIAN LATENCY BY KEYWORD, *admin\_global* SCENARIO (100,005 CORRESPONDENCES, MEDIAN OF 7 RUNS PER KEYWORD).

Keyword (EN)	FTS (ms)	ILIKE (ms)	Speedup
Archiving	50.29	219.02	$4.36\times$
Correspondence	45.41	226.07	$4.98\times$
Outgoing	48.08	199.51	$4.15\times$
Incoming	46.97	196.77	$4.19\times$
Circular	43.93	209.93	$4.78\times$
Procedure	46.66	217.14	$4.65\times$
Documentation	44.26	218.99	$4.95\times$
Approval	43.81	218.12	$4.98\times$
Review	44.67	218.07	$4.88\times$
Search	41.34	192.35	$4.65\times$
Range	41.3–50.3	192.3–226.1	$4.15\times$ – $4.98\times$

TABLE VIII  
EXTENDED SESSION-LEVEL COLD-CACHE MEDIAN LATENCY BY KEYWORD, *admin\_global* SCENARIO (MEDIAN OF 3 RUNS AFTER DISCARD ALL AND RECONNECT; NO SHARED-BUFFER FLUSH).

Keyword (EN)	FTS (ms)	ILIKE (ms)	Speedup
Archiving	48.19	213.30	$4.43\times$
Correspondence	47.35	217.06	$4.58\times$
Outgoing	41.79	204.20	$4.89\times$
Incoming	42.09	202.61	$4.81\times$
Circular	47.62	220.91	$4.64\times$
Procedure	45.31	221.25	$4.88\times$
Documentation	42.92	214.23	$4.99\times$
Approval	41.48	222.78	$5.37\times$
Review	47.84	217.72	$4.55\times$
Search	47.45	189.29	$3.99\times$
Range	41.5–47.8	189.3–222.8	$3.99\times$ – $5.37\times$

Across all 30 keyword–scenario combinations (10 keywords  $\times$  3 scenarios), FTS and ILIKE returned identical result counts in every case. The mean Jaccard similarity index was 1.0000,



with precision and recall of 1.0 for all pairs (treating ILIKE as the reference set). For the institutional vocabulary used in this corpus, Arabic FTS and ILIKE exhibit equivalent retrieval coverage. This equivalence is specific to the corpus vocabulary; it may not generalize to morphologically complex Arabic root forms outside the terms used in synthetic data generation.

#### E. Limitations

The following limitations apply to this study:

- 1) **Synthetic dataset.** The benchmark used synthetic institutional records rather than operational governmental datasets. While the synthetic data was designed to reflect Arabic institutional vocabulary and realistic role structures, differences in vocabulary distribution, query patterns, and record complexity may exist in production environments.
- 2) **Cache conditions.** The primary benchmark (Table IV) reflects warm-cache, steady-state conditions. The extended evaluation (Section V-D) adds session-level cold-cache measurements using `DISCARD ALL` and reconnect, confirming the same directional finding. However, neither protocol includes a full flush of the PostgreSQL shared buffer pool or the operating-system page cache. First-query latencies following server restart or OS-level buffer eviction may therefore be higher than reported here and require separate measurement.
- 3) **Hardware specificity.** All timings are specific to the hardware and software configuration described in Section IV. Results on different hardware, operating systems, or PostgreSQL versions may vary.
- 4) **Single-node deployment.** The evaluation was conducted on a single-node PostgreSQL deployment. The findings do not address distributed or replicated database configurations.
- 5) **Index maintenance at scale.** Although FTS with GIN indexing is theoretically more scalable for text retrieval than sequential scans, the impact of multi-million-record index maintenance and hardware I/O saturation on Arabic-specific configurations remains a subject for future investigation.
- 6) **Retrieval effectiveness.** This study measures retrieval latency as the primary metric. The extended evaluation (Section V-D) reports Jaccard similarity, precision, and recall computed by treating ILIKE as the reference set; these are engineering overlap metrics, not human relevance judgments. For the institutional vocabulary in this corpus, FTS and ILIKE returned identical result sets (Jaccard = 1.0 across all 30 cases). However, this equivalence is specific to the corpus vocabulary and may not hold for morphologically complex Arabic root forms or arbitrary query terms. A comprehensive evaluation against human relevance-judged queries remains deferred to future work.
- 7) **Vocabulary scope.** The extended evaluation covers 10 institutional Arabic terms drawn from the same vocab-

ulary used to generate the synthetic corpus. Generalization to rare or highly specific terms with different morphological profiles or selectivity characteristics requires evaluation on a broader, independently assembled query set.

## VI. CONCLUSION

This study evaluated Arabic full-text search performance and query-level authorization enforcement in ICAS1, an institutional correspondence and archiving system built on Laravel and PostgreSQL. Using a synthetic benchmark database of approximately 100,000 correspondence records, the evaluation compared PostgreSQL Arabic FTS (with GIN indexing) against an ILIKE baseline under identical authorization predicates across four institutional scenarios.

The results demonstrate four principal findings. First, FTS consistently achieved lower execution latencies than ILIKE, with speedups ranging from  $4.2\times$  in the global administrator scenario to  $11.7\times$  in department-scoped scenarios. Second, an extended evaluation across 10 Arabic institutional keywords confirmed the speedup range of  $4.15\times$ – $4.98\times$  (admin, warm cache) and  $10.3\times$ – $12.7\times$  (department scope), with session-level cold-cache measurements preserving the same directional advantage ( $3.99\times$ – $5.37\times$ ). Third, retrieval overlap analysis across 30 keyword–scenario combinations yielded Jaccard = 1.0, confirming equivalent result-set coverage between FTS and ILIKE for the corpus vocabulary. Fourth, access-control validation using deterministic canary records and stability testing over 480 query iterations and 800 ACL checks produced zero failures, supporting the reliability of the integrated approach under sustained workloads.

It should be noted that the evaluation was conducted under single-node conditions on a fixed synthetic corpus. The session-level cold-cache results reduce but do not fully eliminate the warm-cache caveat, as full operating-system and shared-buffer-pool cold starts remain unmeasured. Generalization to production operational datasets, distributed deployments, or morphologically complex query vocabularies outside the corpus vocabulary should be validated through further study.

These findings suggest that combining PostgreSQL Arabic full-text search with query-level authorization predicates and appropriate GIN indexing offers a practical and measurable approach to document retrieval in institutional environments. Future work may extend this evaluation to operational datasets, full cold-start measurement protocols, and multi-node database deployments.

## REFERENCES

- [1] A. Farghaly and K. Shaalan, “Arabic natural language processing: Challenges and solutions,” *ACM Trans. Asian Lang. Inf. Process.*, vol. 8, no. 4, pp. 1–22, 2009.
- [2] L. S. Larkey, L. Ballesteros, and M. E. Connell, “Light stemming for Arabic information retrieval,” in *Arabic Computational Morphology*, A. van den Bosch and K. Soudi, Eds. Dordrecht: Springer, 2007, pp. 221–243.
- [3] PostgreSQL Global Development Group, “PostgreSQL 18 Documentation: Full Text Search,” 2025. [Online]. Available: <https://www.postgresql.org/docs/current/textsearch.html> [Accessed: May 2026].

- [4] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *IEEE Computer*, vol. 29, no. 2, pp. 38–47, 1996.
- [5] S. Rizvi, A. Mendelzon, S. Sudarshan, and P. Roy, "Extending query rewriting techniques for fine-grained access control," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, 2004, pp. 551–562.
- [6] E. M. Voorhees and D. K. Harman, *TREC: Experiment and Evaluation in Information Retrieval*. Cambridge, MA, USA: MIT Press, 2005.
- [7] Transaction Processing Performance Council, "TPC Benchmarks," 2024. [Online]. Available: <https://www.tpc.org/information/benchmarks5.asp> [Accessed: May 2026].
- [8] K. Darwish and D. W. Oard, "Term selection for searching printed Arabic," in *Proc. 25th Annu. Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, 2002, pp. 261–268.
- [9] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed NIST standard for role-based access control," *ACM Trans. Inf. Syst. Secur.*, vol. 4, no. 3, pp. 224–274, 2001.
- [10] PostgreSQL Global Development Group, "PostgreSQL 18 Documentation: GIN Indexes," 2025. [Online]. Available: <https://www.postgresql.org/docs/current/gin.html> [Accessed: May 2026].
- [11] Laravel, "Laravel Documentation: Database Query Builder," 2025. [Online]. Available: <https://laravel.com/docs/13.x/queries> [Accessed: May 2026].
- [12] W. Antoun, F. Baly, and H. Hajj, "AraBERT: Transformer-based model for Arabic language understanding," in *Proc. LREC Workshop on Language Resources and Evaluation for Arabic NLP*, 2020, pp. 9–15.