

1. 이 업무를 왜 하는가? — 팀원이 기억할 한 문장

숫자를 반복하는 것이 아니라 어떤 시스템이 어떤 유형에서 왜 개선/실패했는지 항공법적 의미·한계·시사점으로 해석합니다.

2. 시작 전에 확인할 자료 — 무엇을, 어디서, 누구에게 받나?

선행/입력: TT38 Research_Master의 최종지표·통계·오류분석

- 07_Research_Management/Research_Master.xlsx의 Final_Metrics/Statistics/Errors
- 08_Manuscript/Thesis_Manuscript.docx의 현재 원고
- 07_Research_Management/Research_Master.xlsx의 Protocol/Decision_Log

받을 때 확인: 파일명보다 “승인/Freeze 상태, updated_at, 담당자, hash 또는 version”을 먼저 확인합니다. 최신 작업본과 공식 기준본을 혼동하지 않습니다.

3. 실제 해야 할 일 — 아래 순서대로 실행

1) 연구결과 해석: RQ별로 어떤 시스템이 어떤 질문유형에서 왜 개선/실패했는지 외적타당성·항공법적 의미·한계로 해석한다. 마지막 공통 확인: 작업한 내용의 근거(원문 ID/질문 ID/config key/DB row)를 기록하고, 다음 담당자가 추가 해석 없이 사용할 수 있는 상태인지 스스로 확인합니다.

4. 어떤 툴을 쓰고, 어떤 파일을 만들거나 업데이트하나?

권장 툴: Word, Excel(Research_Master), Zotero/EndNote ※ 실제 설치 버전은

04_Config/RAG_Config_Master.yaml 및 07_Research_Management/Research_Master.xlsx의 REPRODUCIBILITY에 기록

[CREATE/UPDATE] 08_Manuscript/Thesis_Manuscript.docx → 4장 결과해석 + 5장 Discussion/Conclusion 초안부터 같은 파일에서 업데이트

파일 최소화 원칙: 팀이 의식적으로 관리하는 핵심 파일은 6개만 유지합니다 — ①

07_Research_Management/Research_Master.xlsx ② 02_Database/Aviation_Law_Data.duckdb ③

04_Config/RAG_Config_Master.yaml ④ 03_Code/research_pipeline.py ⑤

06_Experiment/Experiment_Run_Master.parquet ⑥ 08_Manuscript/Thesis_Manuscript.docx.

평가표·문헌표·결과표·재현성 manifest는 Research_Master.xlsx의 기존 시트에 통합하고, 원본 법령은

01_Raw_Corpus_V1/에 원형 그대로 보존합니다. Milvus는 05_Vector_Graph_DB/Milvus/, Neo4j는

05_Vector_Graph_DB/Neo4j/를 사용하며 중간 xls/csv/parquet를 별도로 만들지 않습니다.

▶ 실제 작업 프로그램·파일 — 현재 프로젝트 기준

실행 프로그램: Microsoft Word(논문 작성), Microsoft Excel(최종 결과 확인), PyCharm/Matplotlib(필요한 표·그림 산출), DBaver(법령 데이터 수치 확인)

실제 입력 파일/데이터: Research_Master.xlsx FINAL_METRICS-STATISTICS-ERROR_ANALYSIS;

Experiment_Run_Master.parquet; DuckDB; YAML

실제 저장·업데이트 위치: 08_Manuscript/Thesis_Manuscript.docx 4장 결과·5장 시사점/Discussion

작업 시 주의: 논문 수치는 Source of Truth 파일에서 재확인하고 새로운 임의 계산값을 직접 입력하지 않음.

5. 완료 후 누구에게 무엇을 넘기며, 받은 사람은 무엇을 하나?

전달: TT40 최동욱(연구총괄)에게 같은 Thesis_Manuscript.docx의 Discussion/Conclusion 업데이트본을 넘긴다.

다음 담당자의 행동: 최종 원고에서 Method/Results와 논리적 일관성을 확인하고 재현성 manifest와 함께 제출본을 확정한다.

완료 처리 조건: 결과 저장 → QA/검토 → 실제 인계 → 다음 담당자가 열어볼 수 있음을 확인한 뒤 Team Workplan 상태를

“완료”로 변경합니다.

6. 선행연구 근거 — 논문이 직접 보고한 내용과 본 연구 적용을 구분**■ de Martim (2025) — SAT-Graph RAG**

Hudson de Martim (2025), “An Ontology-Driven Graph RAG for Legal Norms: A Hierarchical, Temporal, and Deterministic Approach,” Legal Knowledge and Information Systems, DOI 10.3233/FAIA251598. 표준 RAG가 법의 hierarchy, cross-reference, amendment/repeal에 따른 diachronic 구조를 보지 못해 anachronistic answer를 만들 수 있다고 지적하고, Norm-Component-Temporal Version(CTV)-Language Version 및 Action을 명시적으로 모델링하는 SAT-Graph RAG를 제안했다. 평가는 대규모 정량 benchmark가 아니라 qualitative trace-based proof-of-concept이므로, 본 연구에서는 이 구조를 항공법령에 구현한 뒤 별도의 정량 비교실험으로 효과를 검증한다. DOI: <https://doi.org/10.3233/FAIA251598>

■ Huwiler et al. (2025) — VersionRAG

Daniel Huwiler, Kurt Stockinger, Jonathan Fürst (2025), “VersionRAG: Version-Aware Retrieval-Augmented Generation for Evolving Documents,” arXiv:2510.08109 (preprint). 34개 versioned technical documents에서 수작업으로 만든 VersionQA 100문항을 평가하여 VersionRAG 90%, naive RAG 58%, GraphRAG 64%의 정확도를 보고했고, implicit change detection은 60%로 baseline 0–10%보다 높았다. 구현은 Neo4j+Milvus Lite, 512-token/50-token overlap, DeepDiff 기반 연속버전 비교를 사용했으며 indexing token을 GraphRAG보다 97% 적게 사용했다고 보고했다. 본 연구 적용: 동일한 “버전 인식의 효과”를 대한민국 항공법령에서 검증하되, 날짜필터 baseline과 SAT-Graph를 분리하여 구조효과와 시간효과를 따로 측정한다. URL: <https://arxiv.org/abs/2510.08109>

주의: 위 “선행연구 결과”는 해당 논문이 보고한 범위이고, “본 연구 적용”은 항공법령 도메인에 맞춘 연구설계 판단입니다. 선행연구가 대한민국 항공법령에서 동일 성능을 보장한다고 해석하지 않습니다.

7. 실무 예시와 흔한 실수 — 이 부분을 먼저 보면 일이 더 쉬워집니다

예시: 예: “D가 전체 평균에서 높다”로 끝내지 않고 multi-hop에서 D>C가 난 이유가 hierarchy/delegation edge 때문인지 error traces로 확인해 설명합니다.

흔한 실수/금지: 금지: 결과에 없는 인과관계를 단정하거나 SAT-Graph 선행논문의 주장과 본 실험 결과를 혼동하는 것.

8. 완료 체크 — 네 가지가 모두 YES여야 완료

□ 필요한 선행자료가 공식/Freeze 버전이었다. □ Master 파일/DB/config/code에 결과와 근거가 저장되었다. □ 오류·예외·미해결사항을 숨기지 않고 기록했다. □ 후속 담당자에게 실제 인계했고, 다음 작업에 사용할 위치/방법을 설명했다.