AI-Powered Database Schema Optimizer & Query Generator

Indracare Lab 2.0

Prepared for

myOnsite Healthcare, LLC.

AUG 2025

Document Control

Rev. No.	Description of Change	Effective Date
1.0		06/18/2025

Authored By

Name	Role	Signature	Date
myOnsite Healthcare			22nd Aug 2025

Reviewed and approved By

Name	Role	Signature	Date

1. Project Overview Section

Project Brief

This project delivers an intelligent system that ingests natural-language descriptions of data and business rules and generates an optimized, normalized relational database schema (3NF+ where appropriate) along with primary/foreign keys, unique constraints, indexes (including composite and partial), and CRUD/reporting queries. The system validates the design against stated requirements, estimates storage and cost, and can automatically provision the schema into a target database and execute generated queries.

Beyond generation, the system provides explainability: why a table exists, why a relation cardinality was chosen, why a particular index or constraint is proposed, and what trade-offs were considered (normalization vs. performance vs. maintainability). A feedback loop allows users to refine requirements iteratively and to perform safe migrations from prior versions.

System Overview

The system:

- Parses natural-language requirements into a formal intermediate model.
- Produces an optimized, normalized relational schema with constraints.
- Generates DDL/DML, indexes, views, and representative queries.
- Validates correctness with synthetic and user-provided cases.
- Applies changes to a live database with transactional safety and rollback.
- Tracks versions and provides human-readable rationales.

---

2. Data & Requirements Section

Document Corpus / Data Set

- **Inputs:**
    - Natural-language requirement briefs (1–3 pages each), acceptance criteria, sample records (CSV/JSON), relationship hints, and performance goals.
- **Quantity & Types:**
    - 60–90 requirement briefs spanning 8 domains (healthcare, HR, inventory, billing, scheduling, CRM, IoT, analytics).
- **Complexity Indicators:**
    - Many-to-many relationships, weak entities, temporal tables (history/audit), enumerations, composite keys, nullable vs. required attributes, SCD-like patterns, multi-tenant scoping.
- **Special Characteristics:**
    - Conflicting requirements, implicit constraints (“each patient must have exactly one primary provider”), and privacy-sensitive fields.

Evaluation Dataset

- **Total test cases:** 240
    - **Difficulty tiers:**
        - Tier 1 (30%): Simple entities (≤5 tables), 1:N relations, basic keys.
        - Tier 2 (35%): Moderate (6–12 tables), N:M via junctions, composite indexes.
        - Tier 3 (25%): Advanced (13–20 tables), temporal/audit, partial/filtered indexes, unique across partitions.

        - Tier 4 (10%): Expert (≥21 tables), multi-tenant + row-level constraints, migration from v1→v2 with zero data loss.

- **Special requirements:**

    - Referential integrity proof, anti-pattern detection (EAV, over-denormalization), and performance verification on representative query workloads.

---

3. Technical Requirements Section

Primary Technical Domain — Schema Synthesis & Normalization

- **Coverage & Challenging Content**

    - Entity discovery, attribute typing, relationship cardinality inference, junction table synthesis, surrogate vs. natural keys, lookup/reference tables, archive/temporal tables.

    - Edge cases: polymorphic associations, unique-per-scope constraints, sparse attributes, soft-delete vs. hard-delete, optional relationships, inheritance strategies.

- **Interfaces / Contracts**

    - Input: structured NL spec (YAML/JSON) and/or free-form text with prompts; optional sample data.

    - Output: deterministic IR (entities, attributes, constraints), then DDL/DDL-migrations, and rationale metadata.

- **Performance Targets**

    - Synthesis time ≤ **3 s** for Tier 2, ≤ **8 s** for Tier 3; correctness precision ≥ **95%** vs. gold schemas; index recommendation coverage ≥ **90%** of high-benefit cases.

- **Observability Hooks**

    - Logs: decision trace (entity merge/split, 3NF/BCNF checks), rule applications, risk flags.
    - Traces: spans for parse → model → plan → generate → validate.
    - Dashboards: synthesis latency (P50/P95), validation pass rate, migration success/failure, rollback counts.

Secondary Technical Domain — Query & Index Generation

- **Coverage & Challenging Content**
    - CRUD, reporting queries, aggregations with window functions, parameterized filters, pagination; index planner (single/composite, covering, partial), constraint planner (unique, check).
    - Edge cases: anti-indexing (avoiding redundant indexes), write-heavy vs. read-heavy trade-offs, outlier queries.
- **Contracts**
    - Query pack format (named queries, parameters, expected cardinality), index recommendation set with benefit estimates.
- **Performance Targets**
    - Generated baseline queries achieve **≥1.5×** speedup vs. naive baseline on provided data; P95 query latency ≤ **200 ms** for Tier 2 datasets on commodity hardware.
- **Observability**
    - Query explain plan capture; regression alerting when plan quality degrades > **15%**.

Integration/Architecture Domain — Execution & Safety

- **Coverage & Challenging Content**

    - Connection to target DB, transactional provisioning, idempotent apply, diff & migration (forward/back), environment scoping (dev/test/prod), dry-run and check-only modes.

- **Contracts**

    - Apply API (plan_id, env, dry_run, approvals), artifacts (DDL bundle, migration bundle, rationale).

- **SLOs/SLA**

    - Apply success rate ≥ **99.5%**; automatic rollback in ≤ **30 s** on failure; recovery point objective (RPO) **≤ 5 min**, recovery time objective (RTO) **≤ 10 min** (non-catastrophic).

- **Observability**

    - Events: migration_started, validation_passed/failed, applied, rolled_back; health endpoint; metrics: applies/hour, error rate, long-running DDL warnings.

---

4. Advanced/Challenging Requirements

- **Complex Feature 1 — Normalization Advisor with Trade-off Modes**

    - *What:* Offer “Strict 3NF/BCNF”, “Balanced”, and “Performance-biased” modes with explicit rationale.

    - *Why hard:* Requires reasoning about denormalization candidates, materialized views, and workload hints without violating integrity.

    - *Acceptance:* For a given dataset, Balanced mode reduces read latency by **≥25%** vs. Strict while maintaining **≥99.9%** RI test pass.

- **Complex Feature 2 — Safe Migration from Free-form Spec Changes**

    - *What:* Detect deltas between vN and vN+1 NL specs; generate reversible migrations, including data backfills and constraint changes.

    - *Why hard:* Requires data-safe transforms (split/merge tables, type widen/narrow, nullable changes) with minimal downtime.

    - *Acceptance:* End-to-end migration completes with **zero** integrity violations and **≤30 s** write freeze for Tier 3 cases.

- **Production Considerations**

    - Scale to 200 tables / 2,000 columns per schema; HA for execution component; DR with daily backups and point-in-time restore; IaC for configuration; staged rollouts (blue/green or canary) with automatic rollback triggers.

- **Security/Compliance**

    - AuthN with least-privilege roles; AuthZ via environment scopes; secrets isolation; encryption in transit/at rest; audit log for who/what/when; change approvals; retention policies.

- **Containerization & Dockerization (MANDATORY)**

    - **Multi-stage build** with non-root runtime, minimal attack surface, reproducible builds, build cache, and a **HEALTHCHECK**.

    - **Runtime via compose or equivalent**: services (api, planner, executor, db-sandbox), ports, env vars, secrets, volumes, resource limits, restart policy.

    - **Supply-chain**: image scanning, SBOM creation, provenance attestation.

    - **Operational**: startup readiness/liveness probes; graceful shutdown; stdout structured logs; metrics endpoint.

    - **Example deliverables** (names only): `Dockerfile`, `docker-compose.yml`, `Makefile` targets (`make build`, `make run`, `make test`).

---

5. Output Schema / API Specification

Endpoints / Interfaces

1. **`POST /spec/parse`**
    - Purpose: Ingest NL requirements; return structured IR with ambiguities.
    - Requests: `{ text: string, attachments?: [file], options?: { mode, assumptions_allowed } }`
    - Responses: `{ spec_id, ir, ambiguities[], warnings[] }`
    - Status: 202 accepted for async; 200 sync; 400 invalid input.
    - Errors: `AMBIGUOUS_SPEC`, `INVALID_ATTACHMENT`, `RATE_LIMITED`.
2. **`POST /schema/generate`**
    - Purpose: Generate normalized schema, constraints, indexes, and rationale.
    - Request: `{ spec_id | text, targets?: { normalization, performance_goals } }`
    - Response: `{ plan_id, ddl_bundle, rationale[], quality: { normalization_score, ri_score } }`
    - Pagination: not applicable (bounded output).
    - Errors: `SPEC_NOT_FOUND`, `CONFLICTING_REQUIREMENTS`.
3. **`POST /schema/apply`**
    - Purpose: Provision schema to target DB with safety checks.
    - Request: `{ plan_id, environment, dry_run?: boolean, approvals?: [user], backup?: boolean }`
    - Response: `{ apply_id, status, steps[], rollback_plan? }`
    - Errors: `APPROVALS_REQUIRED`, `APPLY_FAILED`, `ROLLBACK_TRIGGERED`.
4. **`POST /query/generate`**

- Purpose: Produce CRUD/reporting queries with index recommendations.
- Request: `{ plan_id, intents: [list], limits?: { latency_ms, rows } }`
- Response: `{ queries[], indexes[], explains[] }`
- Errors: `PLAN_EXPIRED`, `UNSUPPORTED_INTENT`.

5. **`GET /health`** — readiness/liveness; **`GET /metrics`** — Prometheus-style metrics.

Unified JSON Schema (contract)

```
{
  "primary_id": "string",
  "main_response": {
    "content": {
      "entities": [
        {"name": "string", "attributes":
[{"name":"string","type":"string","nullable":true,"notes":"string"}]}
      ],
      "relationships": [
        {"from":"string","to":"string","cardinality":"1:N|N:M|1:1","via":"string|null"}
      ],
      "ddl_bundle_ref": "string",
      "queries": [{"name":"string","sql_ref":"string","intent":"CRUD|REPORTING"}],
      "indexes": [{"table":"string","columns":["string"],"type":"btree|hash|gist|...","partial":false}]
    },
    "metadata": [
      {"key":"rationale","value":"text"},
      {"key":"normalization_mode","value":"Strict|Balanced|Performance"}
    ],
    "confidence_score": 0.0,
    "uncertainty_factors": ["ambiguous_cardinality","implicit_unique_constraint"]
  },
  "supporting_data": [
    {
      "reference": "spec_id|plan_id",
      "location": ["section:path"],
      "id": "uuid",
      "excerpt": "string",
      "relevance": 0.0,
```

```
      "quality": 0.0,
      "method": "rule_engine|stat_model|heuristic"
    }
  ],
  "performance_metrics": {
    "latency_ms": 0,
    "processing_ms": 0,
    "cost_unit": 0.0
  },
  "system_metadata": {
    "model_info": "string",
    "strategy_info": "string",
    "timestamp": "2025-08-22T00:00:00Z"
  }
}
```


Validation rules & constraints

- All entities must have a primary key; all FKs must reference existing PK/unique keys; junction tables require composite PK or unique constraint; nullable rules enforced; generated indexes must not duplicate PK/unique coverage; partial indexes require predicate validation; queries must be parameterized; all outputs carry rationale entries.

---

6. Evaluation Framework

- **Multi-dimensional scoring**
    - **Schema correctness (40%)**: conformance to gold schemas; RI checks pass rate; normalization score.
    - **Performance (30%)**: P95 query latency and throughput vs. baseline; index benefit realized.
    - **Safety & reliability (20%)**: migration apply success, rollback efficacy, dry-run accuracy.
    - **Explainability (10%)**: rationale completeness and user comprehension rating.

- **Advanced evaluation methods**
    - Adversarial requirement mutations (conflicts, hidden unique constraints).
    - Statistical comparison of execution plans vs. baseline; bootstrap CI on latency deltas.
    - Red-team prompts to induce dangerous DDL; must be rejected with safe messaging.
- **Benchmark targets**
    - **P95 synthesis latency:** ≤ 8 s (Tier 3).
    - **Throughput:** ≥ 20 plans/hour sustained.
    - **Correctness:** ≥ 95% entity/relationship match; **RI tests:** 100% pass.
    - **Index benefit:** ≥ 1.5× median speedup on target workloads.

---

7. System Adaptation / Flexibility Requirements

The system must adapt to:

- Varying NL input forms (bullets, prose), partial specs, and iterative refinements.
- Domain-specific constraints (e.g., healthcare privacy fields), multi-tenant scopes, temporal/audit needs.
- Different performance goals (OLTP vs. reporting-heavy).

Structured flexible message example

```
{
  "input_variant": "prose|bulleted|yaml",
  "assumptions_allowed": true,
  "domain_hints": ["healthcare","inventory"],
```

```
  "tolerances": {"latency_ms": 250, "storage_growth_pct": 20},
  "scoping": {"tenant_scoped": true, "row_level_policy": "by_org_id"},
  "conflict_resolution": "prefer_integrity|prefer_performance",
  "output_controls": {"normalization": "Balanced", "index_budget": 12}
}
```


---

8. Implementation Challenges

- **Technical Challenge 1: Ambiguity Resolution**
    - *Risk:* Misinterpreting cardinalities and optionality.
    - *Mitigation:* Ask-back list of explicit clarifying prompts; heuristic + rules; ambiguity flags.
    - *Test:* Gold cases with known traps; require ≥ 95% correct disambiguation.
- **Technical Challenge 2: Migration Safety**
    - *Risk:* Data loss on column type change or table split/merge.
    - *Mitigation:* Dry-run with shadow tables, backfills, dual-write simulation, checksum validation.
    - *Test:* Inject failures; verify rollback and integrity hold.
- **Production Challenge: Performance Regressions**
    - *Risk:* Generated indexes harming write throughput.
    - *Mitigation:* Cost-benefit scoring, write-penalty caps, canary evaluation.
    - *Test:* Before/after benchmark; alert if write latency ↑ > 20%.
- **Security/Compliance Category**
    - *Risk:* Excessive privileges.
    - *Mitigation:* Least-privilege roles per environment; secrets rotation; audit logs.

- *Test:* Access reviews, role simulation, policy linting.

---

9. Evaluation Complexity

Tiering & Progression

- Tier 1 (30%): ≤5 tables; simple 1:N; basic constraints.
- Tier 2 (35%): 6–12 tables; at least one N:M; 2–3 composite indexes.
- Tier 3 (25%): 13–20 tables; temporal/audit; 1 complex migration; partial indexes.
- Tier 4 (10%): ≥21 tables; multi-tenant; zero-downtime migration with hot-swap validation.

---

10. Advanced Challenge Specifications

Hot-swap / Migration Requirements

Support seamless adaptation of:

- **Schema versions:** plan, preflight, apply, verify, rollback.
- **Index sets:** create concurrently, validate benefit, drop safely.
- **Constraint policies:** introduce unique/check constraints with online validation and backfill.

Zero-downtime / High-availability

- Targets: Read availability **99.95%+**, write freeze windows ≤ **30 s** for expert cases.

- Rollout: Blue/green (or canary) apply; automatic rollback on health degradation; dual-read verification.
- Monitoring: Migration health dashboard; alerts on failed backfills, lock contention, replication lag.

Multi-Tier Evaluation Suite

- **Tier 1:** Correct entities, PK/FK integrity, basic CRUD generation.
- **Tier 2:** Proper junctions, composite indexes, explain-plan improvements.
- **Tier 3:** Temporal tables, partial indexes, safe migration applied.
- **Tier 4:** Multi-tenant scoping, zero-downtime rollout, data validation at scale.

Challenge-type Testing

- Conflicting uniqueness constraints across scopes.
- Type narrowing (text→enum) with existing dirty data.
- Splitting a wide table into two with backfilled FK.
- Adding row-level security without breaking existing queries.
- Deleting an index that appears redundant but serves a niche filter.

Performance Benchmarking

- **Synthesis P95:** ≤ 8 s.
- **Query P95:** ≤ 200 ms (Tier 2 data scale).
- **Index benefit:** ≥ 1.5× median.
- **Apply success:** ≥ 99.5% first-attempt applies.

---

11. Deliverables Section

System Deliverables

- Natural-language → IR parser and validator.
- Schema synthesis engine (normalization checks, relation inference).
- DDL/migration generator with rationale artifacts.
- Query and index recommender with explain-plan capture.
- Execution service (apply/dry-run/rollback) with health/metrics.

Evaluation Deliverables

- Gold schemas and query workloads per tier.
- Automated test harness (correctness, performance, safety).
- Adversarial and red-team case pack.

Documentation Deliverables

- Architecture overview (C4 style levels: Context → Container → Component → Code).
- Runbook (provisioning, rollback, incident response).
- API reference (endpoints & schemas in Section 6).
- Security checklist (roles, secrets, audit, approvals).
- Decision log (trade-offs, normalization decisions).

Tools & Packaging Deliverables

- Container artifacts: `Dockerfile`, `docker-compose.yml`, `Makefile` targets (`build/run/test`).
- SBOM and image scan report; provenance attestation.

- Environment bootstrap script; sample config; example inputs/outputs.

---

12. Success Criteria

Technical Excellence

- Normalization verification score ≥ **95%**; zero dangling FKs; rationale for all design choices.

System Performance

- Synthesis P95 ≤ **8 s**; query P95 ≤ **200 ms** (Tier 2); index benefit ≥ **1.5×** median.

Production Readiness

- Apply success ≥ **99.5%**; automated rollback ≤ **30 s**; comprehensive metrics/traces/logs; backup/restore validated.

Innovation & Depth

- Explainable trade-off modes; adversarial robustness; hot-swap migrations with zero data loss.

During Development Proof Points

1. Load test: 100 sequential plan generations; P95 ≤ 8 s; zero crashes.
2. Failure injection: kill apply mid-migration → automated rollback and integrity holds.
3. Chaos experiment: introduce lock contention → system defers DDL and succeeds within SLA.

APPENDIX A — Content Guidelines for AI Generation

This assessment employs professional, senior-engineer language, quantified targets, and structured lists, aligning with enterprise documentation norms for production deployment and evaluation across basic→advanced→expert tiers. It requires explicit Dockerization, monitoring, security, and multi-tier evaluation artifacts.

APPENDIX B — Usage Instructions

- Assumptions are explicitly labeled where needed.
- The document stays vendor-neutral; no specific product stacks or sample code beyond schemas and interface contracts.
- Metrics have concrete thresholds; diagrams described for reproducibility.
- Deliverables span System, Evaluation, Documentation, Tools, matching the evaluation framework.

Diagram Description (C4-style you can implement)

- **Context:** “AI DB Designer” interacting with “User” and “Target Database”.
- **Container:**
    - Parser/IR Service, Synthesis/Planner, Query/Index Generator, Execution Service, Observability Stack.
- **Components:**
    - Rule Engine, Constraint Checker, Migration Planner, Index Benefit Estimator, Health/Metric endpoints.
- **Code-level:**
    - Contracts defined in Section 6 JSON; artifacts emitted as references (DDL bundle, query pack).