**Where this idea breaks (major flaw categories) — and how to mitigate**

**1) Semantic / terminology mapping (ICD, SNOMED, RxNorm, LOINC)**

**Problem:** Natural language criteria (“diabetic neuropathy”, “severe COPD”, “on metformin”) map to many codes, hierarchies, value sets, and local variants. Text2SQL may produce queries that reference the wrong code system or miss synonyms.  
**Consequence:** False positives/negatives in cohorts; biased research cohorts.  
**Mitigations:**

* Use a terminology service (SNOMED/ICD/RxNorm resolver) to expand and normalize concepts before SQL generation.
* Maintain curated, site-specific value sets and let the system prefer them; fall back to mapped expansions if needed.
* Show the researcher the resolved code lists and let them accept/reject edits (human-in-the-loop).

*(Why this is essential: FHIR is flexible and clinical codes live in different fields; SQL-on-FHIR views expose fields but don’t solve term resolution.)* [FHIR+1](https://build.fhir.org/ig/FHIR/sql-on-fhir-v2/StructureDefinition-ViewDefinition.html?utm_source=chatgpt.com)

**2) Temporal, episodic, and index-date logic**

**Problem:** Cohort criteria frequently include temporal constraints (e.g., “diagnosis in last 12 months before index date”, “two prescriptions within 90 days”, “lab rise of X after event”). Translating ambiguous natural language into correct SQL temporal logic is hard.  
**Consequence:** Substantially different cohorts depending on off-by-one, timezone, event-selection semantics.  
**Mitigations:**

* Normalize temporal language into an explicit intermediate representation (IR) capturing index date, windows, event anchors.
* Require explicit index date semantics for ambiguous inputs or present multiple interpretations.
* Unit tests for typical temporal patterns; compare output vs a gold standard.

**3) Complex cohort logic (nested inclusion/exclusion, prior history, lookback windows)**

**Problem:** Queries may contain nested logic (include patients with A *and* (B or C) *but not* D within 6 months) and cohort builders often have nuanced semantics (patient-level deduplication, encounter vs patient scoping). Text2SQL models can get the boolean and scoping wrong.  
**Mitigations:**

* Convert NL → structured cohort DSL (e.g., inclusion/exclusion lists with explicit grouping) before SQL emission.
* Render a human-readable preview of interpreted logic and the generated SQL (and underlying code lists) for sign-off.

**4) FHIR structural variation & view coverage**

**Problem:** FHIR resources are nested JSON and sites implement different profiles. SQL-on-FHIR uses view definitions (FHIRPath) to flatten into tables, but **view runners vary** in completeness of FHIRPath function support and coverage. The spec exists, but real-world implementations can differ. [GitHub+1](https://github.com/FHIR/sql-on-fhir-v2?utm_source=chatgpt.com)  
**Consequence:** A query that works on one site’s SQL-on-FHIR view runner fails (different column names, missing fields).  
**Mitigations:**

* Deploy standardized, versioned view definitions across sites (or ship site-specific adapters).
* Add capability discovery: confirm which columns/views the runner supports before generating SQL.
* Fall back to Bulk Data + transformation into a local analytic schema if view runner is incomplete.

**5) Real-time vs. batch (performance & scale)**

**Problem:** “Real-time” cohort counts over large populations are expensive. Hitting many FHIR resources or live view runners with complex SQL on production EHRs risks performance issues and throttling. SMART Bulk Data / Flat FHIR is the common pattern for population scale and analytics; it’s often a better path for heavy analytic loads. [SMART Health IT](https://smarthealthit.org/smart-hl7-bulk-data-access-flat-fhir/?utm_source=chatgpt.com)  
**Mitigations:**

* Use a hybrid pattern: for lightweight exploratory queries use SQL-on-FHIR view runners; for heavy / sitewide runs, use Bulk Data exports + materialized analytic tables (or cache views in a data warehouse).
* Implement query cost estimation and warn/restrict heavy queries; throttle and queue long jobs.
* Maintain incremental refresh pipelines so “near-real-time” means a short window (minutes–hours), not milliseconds.

**6) LLM/Text2SQL failure modes (hallucination, SQL injection, brittle generation)**

**Problem:** LLMs can hallucinate schema elements or generate unsafe SQL. They may produce syntactically plausible but semantically wrong queries. Security risk if raw user text influences SQL (injection-like issues).  
**Mitigations:**

* Don’t allow LLM→DB unrestricted. Use template-driven generation + constrained grammar, or produce an IR that’s validated by deterministic rules.
* Sanitize and canonicalize generated SQL; run in read-only, audited query engines with query parameterization.
* Keep a human-in-the-loop approval step for production cohort definitions.

*(Papers show Text2SQL for cohorts works well in constrained setups but requires careful pipeline design and validation.)* [PMC+1](https://pmc.ncbi.nlm.nih.gov/articles/PMC11129920/?utm_source=chatgpt.com)

**7) Data quality, provenance, and representational differences (FHIR vs OMOP etc.)**

**Problem:** FHIR is optimized for clinical exchange; OMOP and other CDMs are optimized for research analytics. Mapping FHIR→OMOP/analytic model has pitfalls; errors in mapping change cohort semantics. [FHIR](https://build.fhir.org/ig/HL7/fhir-omop-ig/F2OGeneralIssues.html?utm_source=chatgpt.com)  
**Mitigations:**

* Document provenance; include mapping rules in outputs and logs.
* Where possible, run parallel queries on both FHIR-derived views and OMOP (if available) to cross-check results.
* Provide confidence metrics and provenance to researchers.

**8) Privacy, governance, and authorization**

**Problem:** Running ad-hoc natural-language cohort queries risks exposing PHI. Auditing, role-based access, de-identification, and IRB governance must be enforced.  
**Mitigations:**

* Enforce SMART/OAuth scopes and fine-grained RBAC; require researcher identity and IRB/project metadata for cohort queries. [FHIR](https://build.fhir.org/ig/HL7/smart-app-launch/scopes-and-launch-context.html?utm_source=chatgpt.com)
* Audit every generated SQL and returned result.
* Use de-identified or aggregate counts until approvals are in place.

**Recommended architecture (practical, hybrid)**

1. **NL input** → **Parser/Intent extractor**
   * Produce an explicit intermediate cohort DSL (inclusion/exclusion blocks, index date, time windows, domain constraints). Use deterministic NLP components (regex, rule engine) + LLM for paraphrase when needed.
2. **Concept expansion & term resolution**
   * Call terminology service to expand terms into code sets (SNOMED/ICD/RxNorm/LOINC), with site overrides.
3. **Capability discovery**
   * Query the target site’s SQL-on-FHIR view runner capabilities (available views/columns, supported FHIRPath functions).
4. **SQL generation**
   * Use a template/grammar-based generator that consumes the DSL + resolved code lists + view metadata. Avoid direct freeform LLM→SQL.
5. **Static validation & cost estimation**
   * Validate for semantic integrity (e.g., no ambiguous temporal windows), estimate cost, and enforce query limits.
6. **Execution path decision**
   * If small/fast: run on SQL-on-FHIR view runner (live).
   * If broad/expensive: run on pre-materialized analytic tables / Bulk Data export transformed into warehouse schema.
7. **Human review & sign-off**
   * Present readable cohort definition, codes, and sample SQL to researcher for approval (or auto-approve for low-risk exploratory counts).
8. **Execution, logging, provenance**
   * Execute read-only queries; log SQL, translations, user, purpose, and results; expose lineage metadata.
9. **Monitoring & feedback loop**
   * Capture false positives/negatives reported by researchers; update term maps and templates.