FAERS as a Semantic Graph in Amazon Neptune

Using Amazon Neptune to Solve Problems in Pharmacovigilance

RRecktek LLC

2026-07-03

Table of Contents

Abstract

This report specifies and documents an end-to-end method for converting the complete public release of the U.S. FDA Adverse Event Reporting System (FAERS) into RDF. The RDF is then loaded into Amazon Neptune for large-scale pharmacovigilance analysis. Rather than minting bespoke terms, the design reuses established biomedical and Semantic Web vocabularies. It models each safety report as a provenance-bearing graph, and it materializes drug–reaction disproportionality statistics as first-class observations. The principal analytical objective is the **Unlabeled Signal** query. This query identifies drug–reaction associations that show statistically elevated reporting frequency but are **not** described in the corresponding FDA product label. It is a graph anti-join that operationalizes a core regulatory pharmacovigilance task. All data-acquisition and conversion stages have been executed and timed on local infrastructure. The cloud stages — provisioning, bulk load, and query — are fully specified and scripted. Measured results to date: 89 quarterly releases (2004 Q1 – 2026 Q1) acquired and integrity-verified, 18.48 GB of ASCII source decompressed in 11.6 minutes wall-clock, and a measured RDF yield extrapolating to approximately 6.7×10^8 triples after deduplication.

1. Introduction

FAERS is the FDA's post-marketing spontaneous reporting database for adverse drug events. It is distributed as quarterly extracts of seven `$`-delimited ASCII tables. Although relational in distribution format, FAERS is intrinsically a graph. Each case report connects a patient to one or more drugs, one or more reactions, one or more outcomes, an indication, a therapy interval, and a reporting source. Relational storage forces multi-table joins for questions that are naturally expressed as graph traversals. It also provides no native mechanism for aligning FAERS entities to external controlled terminologies or to the FDA product labels that define the *expected* adverse event profile of each drug.

This work reframes FAERS as an RDF knowledge graph in Amazon Neptune. The motivating question is not "load FAERS as a graph." It is instead the exploitation of three independently maintained datasets that are already co-resident on local infrastructure:

1. **FAERS** — observed drug–reaction associations from spontaneous reports.
2. **openFDA drug labels** — the reactions already *known and labeled* for each product.
3. **MedDRA** — the hierarchical terminology (Preferred Term → System Organ Class) used to code reactions.

The intersection of these three sources supports a query that is difficult in relational form but natural in a graph: find drug–reaction pairs with high disproportionality **for which no labeled association exists**, at either the exact Preferred Term or any hierarchical ancestor. These are candidate previously-undescribed safety signals.

The difficulty is one of shape rather than volume. The labeled-association test is a *variable-length* walk up the MedDRA hierarchy: an exact Preferred Term or any of its ancestors may carry the label. Moreover, the disproportionality evidence, the label knowledge, and the terminology each originate in a different, independently maintained dataset. In relational form the question requires a recursive traversal of a self-joined closure table, anti-joined against a separately modeled label schema. In a graph it is a single property-path pattern under a negation. Amazon Neptune supplies exactly this: SPARQL 1.1 property paths over an RDF graph in which the three datasets are fused by shared identifiers, evaluated at a scale ($\approx 6.7 \times 10^8$ triples) a managed triplestore serves from memory.

2. Data Acquisition and Verification

2.1 Source and scope

The complete quarterly extract series was retrieved directly from the FDA distribution endpoint (`https://fis.fda.gov/content/Exports/`). The series spans **89 quarters, 2004 Q1 through 2026 Q1**. Releases prior to 2012 Q4 use the legacy AERS naming and schema (`aers_ascii_*`, with a single `ISR` case key). From 2012 Q4 onward, releases use the modern FAERS schema (`faers_ascii_*`, with `PRIMARYID`/`CASEID` keys).

2.2 Acquisition results

Property	Value
Quarters retrieved	89 / 89
Compressed size	3.24 GB
Missing quarters	0
Zero-byte files	0
Archive integrity failures (`unzip -t`)	0
Coverage span	`aers_ascii_2004q1` … `faers_ascii_2026q1`

Every archive passed a full CRC integrity test. Acquisition is therefore complete and lossless.

2.3 Parallel decompression — measured performance

All 89 archives were decompressed in parallel (degree of parallelism P = 16) on a 72-core host, writing to NFS-backed storage. Timing was instrumented per archive.

Metric	Value
Archives decompressed	89 / 89 (0 failures)
Total wall-clock time	693.8 s (11.6 min)
Sum of per-archive times	9,970.9 s (2.77 h serial-equivalent)
Parallel speedup	**14.37×** (≈ 90 % efficiency at P = 16)
Compressed → decompressed	3.24 GB → **18.48 GB** (5.7× expansion)
Sustained throughput	27.3 MB/s decompressed, wall-clock

The slowest archives (`2016q3` 242.9 s; `2019q2` 240.2 s; `2021q2` 234.3 s) are recent quarters dominated by large `DRUG` and `DEMO` tables. They are NFS-write-bound rather than CPU-bound. As a result, increasing P beyond 16 yields diminishing returns: additional concurrency raises write contention without relieving the bottleneck.

3. Data Model and Ontology

3.1 Design principle: reuse before minting

The model reuses established vocabularies wherever a suitable term exists. It mints terms only in a project namespace (`fv:`), and only for concepts with no standard equivalent. Related prior art includes the AEOLUS normalized FAERS resource, the Bio2RDF FAERS/AERS effort, and the Ontology of Adverse Events (OAE).

Concern	Reused vocabulary	Role in the model
Provenance	PROV-O	Each report `prov:wasDerivedFrom` its quarterly source
Terminology hierarchy	SKOS	MedDRA PT `skos:broader` HLT → HLGT → SOC
Drug identity	RxNorm (RXCUI)	Normalization target for free-text drug names
Drug classification	ATC	Class-level rollups
Reaction / indication coding	MedDRA	REAC and INDI terms
Disproportionality statistics	RDF Data Cube (`qb`)	PRR/ROR/EBGM/χ^2/N as `qb:Observation`
Clinical alignment	FHIR RDF	`fhir:AdverseEvent`, `fhir:MedicationStatement`
Time intervals	OWL-Time	Therapy start/end as `time:Interval`
Document metadata	Dublin Core Terms	Titles, dates, licensing

3.2 Namespaces

```
fv:     http://rrecktek.com/ns/faers#        (minted ontology terms)
faers:  http://rrecktek.com/id/faers/        (instance identifiers)
rxn:    http://purl.bioontology.org/ontology/RXNORM/
meddra: http://purl.bioontology.org/ontology/MEDDRA/
skos:, prov:, qb:, dct:, time:, fhir:, owl:, rdfs:, xsd:  (standard)
```

3.3 Core graph model

A safety report is typed simultaneously as a project class and a FHIR resource, and it carries a provenance edge to its source quarter. Drug mentions are reified: role, dose, route, indication, and therapy are properties of the *mention within a case*, not of the drug in general. Each mention uses a deterministic identifier (`faers:mention/{report}/{seq}`) rather than a blank node. This keeps the DRUG/INDI/THER tables join-free and streamable during conversion, and it is friendlier to the Neptune bulk loader.

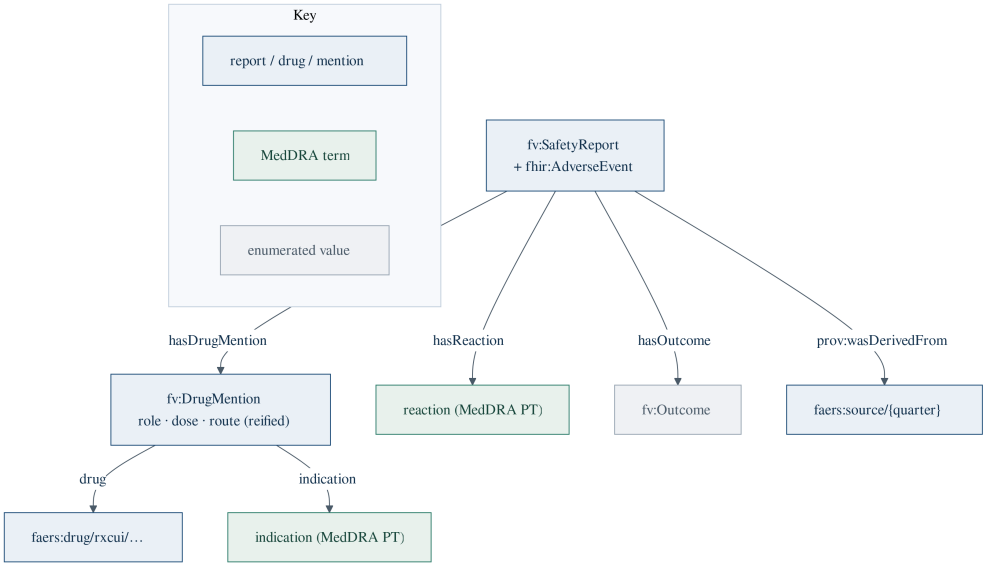

Figure 1: Reified safety-report model. A case links to a reified drug mention (carrying role, dose, and route) and to reaction and indication Preferred Terms, an outcome, and its source quarter. The shared mention IRI is what keeps the DRUG/INDI/THER tables join-free.

```
faers:case/100234567 a fv:SafetyReport, fhir:AdverseEvent ;
    fv:primaryId "100234567" ; fv:caseVersion 3 ;
    fv:eventDate "2024-02-11"^^xsd:date ; fv:reporterCountry "US" ;
    fv:patientSex fv:Female ; fv:serious true ; fv:hasOutcome fv:Hospitalization ;
    prov:wasDerivedFrom faers:source/2024q1 .

faers:case/100234567 fv:hasDrugMention [
    a fv:DrugMention ; fv:role fv:PrimarySuspect ;
    fv:drug faers:drug/rxcui/310965 ; fv:route "ORAL" ;
    fv:indication faers:rxn/pt/10021113 ] .

faers:case/100234567 fv:hasReaction faers:rxn/pt/10019211 .

# Materialized disproportionality signal (RDF Data Cube observation)
faers:signal/310965-10019211 a qb:Observation, fv:DrugReactionSignal ;
    fv:drug faers:drug/rxcui/310965 ; fv:reaction faers:rxn/pt/10019211 ;
    fv:nCases 412 ; fv:nDeaths 63 ; fv:PRR 5.2 ; fv:ROR 5.4 ; fv:chi2 380.1 ; fv:EBGM 4.9 ;
    fv:onLabel false .          # ← derived from the openFDA label join
```

The ontology (`faers-ontology-1783036311.ttl`) formalizes all `fv:` classes and properties with labels, domains, ranges, and OWL alignments (`fv:SafetyReport` `fhir:AdverseEvent`, `fv:DrugReactionSignal` `qb:Observation`). It also materializes the Outcome and DrugRole enumerations as SKOS concepts. It validates at 544 triples.

4. RDF Conversion Pipeline

4.1 Method

Conversion runs on-premises, on the host that already stores the data. It emits gzipped N-Triples, sharded per quarter and table. The converter is a single-pass streaming program that uses only the Python standard library and builds no in-memory graph, which is necessary at the billion-triple scale. It branches on schema era, handling legacy ISR-keyed releases and modern PRIMARYID-keyed releases separately. For each `CASEID` it keeps only the latest `CASEVERSION`, following the AEOLUS deduplication method. Coded fields — `ROLE_COD`, `OUTC_COD`, sex, and age with unit normalization — are mapped to the ontology enumerations. Drug-name→RXCUI and reaction-text→MedDRA-code resolution are implemented as pluggable hooks. When a dictionary is absent, these hooks degrade gracefully to name-slug identifiers.

4.2 Measured yield

Quarter	Cases (deduped)	Triples	Conversion time
2004 Q1 (legacy)	58,235	2,868,647	4.8 s
2026 Q1 (modern)	397,224	23,694,758	43 s

Measured ratios (49–60 triples/case; 6.1 gzipped bytes/triple) extrapolate across the full series:

Quantity	Estimate
Raw report rows	≈ 21 million
Cases after per-quarter deduplication	≈ 19 million
Triples (raw)	≈ 1.1×10^9 (≈ 6.8 GB gzipped; ≈ 150 GB uncompressed)
Triples (after global cross-quarter deduplication)	≈ **6.7×10^8** (≈ 4.1 GB gzipped)

The openFDA label parser streams each 630 MB JSON part incrementally. It emits `fv:labeledReaction` edges from the `openfda.rxcui` array and from the labeled adverse-reaction sections. Free-text reaction extraction is isolated behind a controlled-vocabulary gazetteer to prevent uncontrolled term generation.

5. Amazon Neptune Architecture

Amazon Neptune is a managed graph database supporting RDF and SPARQL 1.1. This work depends on two of its capabilities. First, it evaluates unbounded property paths (`skos:broader*`) directly in the query engine. Second, the Neptune Bulk Loader ingests N-Triples from S3 at billion-triple scale. Neptune ML adds graph-neural-network link prediction over the same loaded graph without exporting it. Neptune has no public endpoint and must be reached from within its VPC; RDF is ingested through the Neptune Bulk Loader from Amazon S3. The recommended topology is a dedicated VPC (CIDR `10.1.0.0/16`, to avoid collision with the existing `10.0.0.0/16` VPC), a private Neptune cluster,

a small EC2 loader/jump host in the same VPC, an S3 staging bucket colocated in region, and a VPC S3 gateway endpoint so the private cluster can read the bucket during load.

5.1 Instance sizing

Neptune is memory-bound: query performance depends on retaining the working set in the buffer pool (≈ two-thirds of instance RAM). Bulk-load throughput scales with vCPU count and loader parallelism.

Phase	Recommended class	Rationale
Bulk load	`db.r6g.8xlarge` (32 vCPU / 256 GiB)	Fastest load; fewer billed hours overall
Steady-state query	resize to `db.r6g.2xlarge` (64 GiB)	Holds the Unlabeled-Signal working set at high cache-hit ratio

Storage should be configured **I/O-Optimized** for the load phase. This eliminates the per-request I/O charge incurred by writing hundreds of millions of triples. Compute and storage are decoupled, so the writer can be resized — a minutes-long reboot — without reloading.

5.2 Provisioning environment

The target AWS account is [REDACTED], region `us-east-1`, with no pre-existing Neptune clusters. Provisioning, bulk load, monitoring, and teardown are fully scripted (`provision-neptune-*.sh`, `bulk-load-*.sh`, `monitor-and-teardown-*.sh`). All scripts pass syntax validation, capture every resource identifier to a state file, and are gated behind explicit confirmation flags.

6. Using Amazon Neptune

The following sections document the concrete Neptune operations the scripts perform.

6.1 Provisioning a cluster with the AWS CLI

A Neptune deployment is created as a cluster plus one or more instances:

```
aws neptune create-db-cluster --db-cluster-identifier faers \
    --engine neptune --vpc-security-group-ids $SG --db-subnet-group-name faers-subnets \
    --serverless-v2-scaling-configuration MinCapacity=2.5,MaxCapacity=128
aws neptune create-db-instance --db-instance-identifier faers-writer \
    --db-cluster-identifier faers --engine neptune --db-instance-class db.r6g.8xlarge
aws neptune wait db-instance-available --db-instance-identifier faers-writer
```

An IAM role granting S3 read access is created and associated with the cluster via `add-role-to-db-cluster`; this role authorizes the bulk loader to read the staging bucket.

6.2 Loading RDF via the Neptune Bulk Loader

Gzipped N-Triples are synchronized to S3. A single REST call to the loader endpoint, issued from the in-VPC EC2 host, then starts an asynchronous load:

```
curl -X POST https://$NEPTUNE:8182/loader -H 'Content-Type: application/json' -d '{
  "source":"s3://faers-rdf/nt/","format":"ntriples","region":"us-east-1",
  "iamRoleArn":"'$ROLE'","failOnError":"FALSE","parallelism":"OVERSUBSCRIBE" }'
```

Progress is polled at `/loader/{loadId}` until status `LOAD_COMPLETED`, recording throughput and total load time. `OVERSUBSCRIBE` parallelism uses all available vCPUs, which is why load time scales strongly with instance class.

6.3 Querying the SPARQL endpoint

Queries are issued to the cluster's SPARQL 1.1 endpoint. With IAM authentication enabled, requests are SigV4-signed (for example via `awscurl`):

```
awscurl --service neptune-db --region us-east-1 \
  -X POST https://$NEPTUNE:8182/sparql --data-urlencode "query@01-unlabeled-signal.rq"
```

The Neptune workbench additionally provides the `%%sparql` notebook magic for interactive use.

6.4 Visual exploration with graph-explorer

Neptune graph-explorer, the AWS open-source graph UI, connects to the cluster endpoint and renders result subgraphs directly. For this project it is configured to seed from the Unlabeled-Signal result set, style drug and reaction nodes distinctly, scale signal-edge width by PRR, and render unlabeled (`onLabel = false`) edges in a contrasting style so candidate signals are immediately visible.

6.5 Link prediction with Neptune ML (phase 2)

Neptune ML trains graph neural networks, via SageMaker, directly on the loaded graph. Applied to the drug→reaction edge set, link prediction proposes associations that are structurally likely but not yet present — a machine-learning complement to the disproportionality method for surfacing undiscovered signals.

6.6 Monitoring and teardown

CloudWatch metrics (`CPUUtilization`, `BufferCacheHitRatio`, `NumTxCommitted`) are pulled during load and query. A `BufferCacheHitRatio` below approximately 0.99 during querying indicates the working set no longer fits in RAM, and the instance should be resized up. Teardown deletes the instance, cluster, EC2 host, security groups, subnets, VPC, and S3 bucket, ending all billing; it requires an explicit confirmation flag.

7. Analytical Method

7.1 Disproportionality statistics

For each drug–reaction pair, a 2×2 contingency table is formed over all reports. The four cells are: a = reports with both drug and reaction; b = drug, not reaction; c = reaction, not drug; and d = neither. From this table we compute the Proportional Reporting Ratio (PRR), the Reporting Odds Ratio (ROR), a χ^2 statistic (Yates-corrected), and the Empirical Bayes Geometric Mean (EBGM). These values are precomputed and materialized as `fv:DrugReactionSignal` observations. Analytical queries then read the stored values rather than recomputing contingency tables at query time.

7.2 The Unlabeled Signal query

A candidate signal is a `fv:DrugReactionSignal` that meets the conventional screening thresholds (PRR ≥ 2, N ≥ 3 cases, χ^2 ≥ 4) and for which the drug has **no** labeled association to that reaction. Here "labeled" is evaluated reflexively-transitively over the MedDRA hierarchy: an on-label association to any ancestor term counts as labeled. The query is provided in two forms. One reads the materialized `fv:onLabel` flag; the other recomputes it live:

```
FILTER NOT EXISTS {
  ?reaction skos:broader* ?labeledTerm .
  ?drug fv:labeledReaction ?labeledTerm .
}
```

The live form keeps the materialized flag auditable, so that flag is never load-bearing for correctness. The `skos:broader*` step is the operative detail. The `*` makes the ancestor walk reflexive and unbounded, so a label attached at any level of the MedDRA tree — Preferred Term, High-Level Term, or System Organ Class — discharges the signal. Neptune evaluates that closure inside the query. The equivalent relational statement is a recursive CTE over a precomputed transitive-closure table, which must be rebuilt whenever the terminology changes.

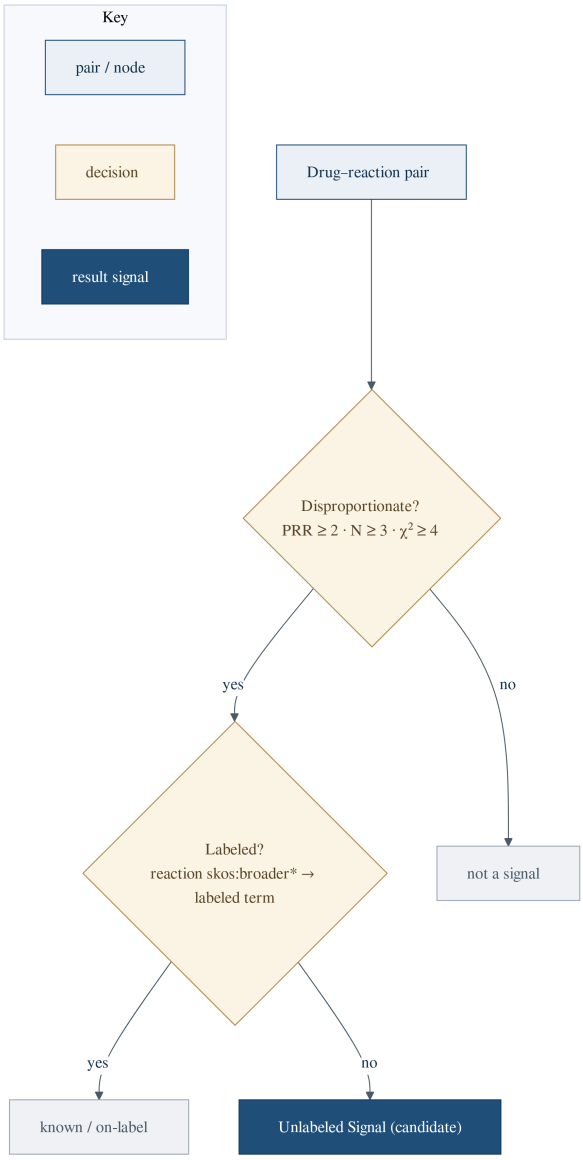

Figure 2: The Unlabeled-Signal test. A drug–reaction pair is a candidate only when it is disproportionate and is not labeled at the reaction Preferred Term or any MedDRA ancestor.

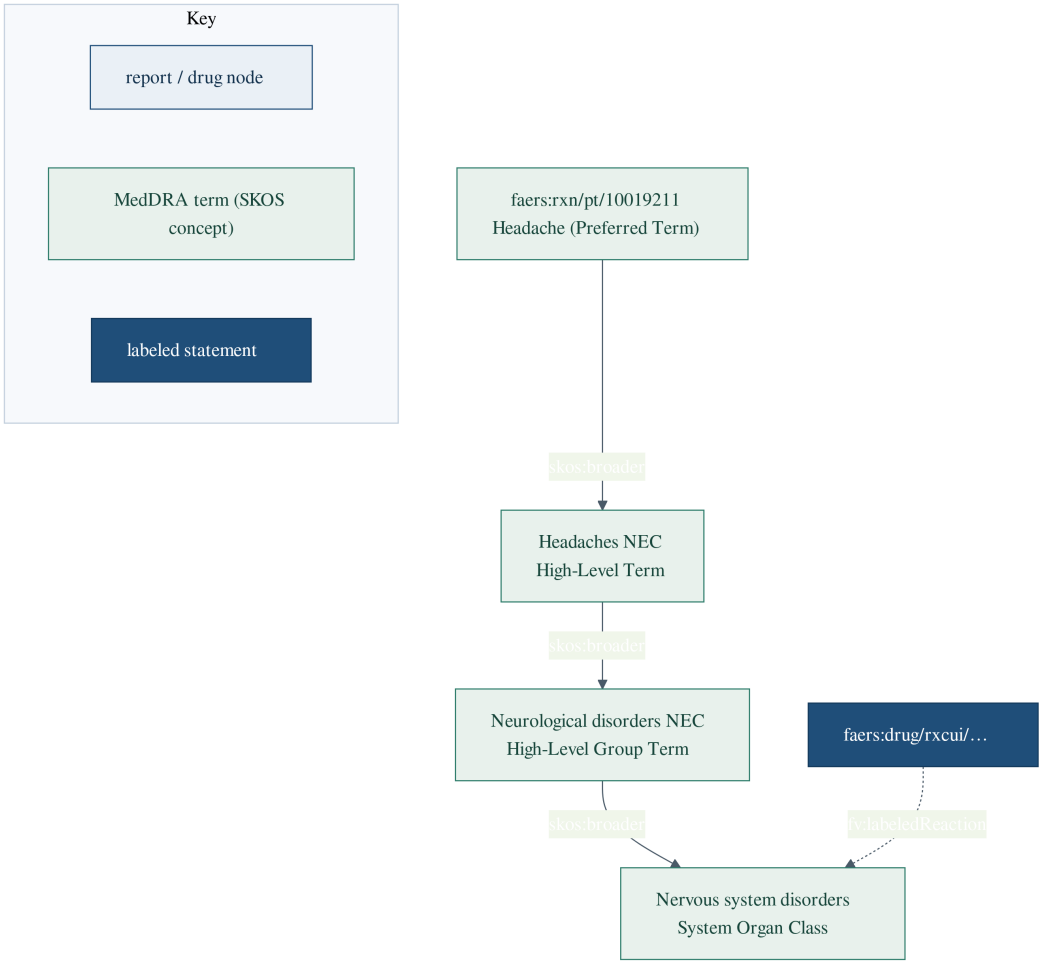

Figure 3: The `skos:broader*` lift. A reaction Preferred Term climbs to its High-Level Term, High-Level Group Term, and System Organ Class; a drug label attached at any of those ancestors discharges the signal.

Four additional queries support the broader analysis: disproportionality top-N, drug–drug interaction cliques (stratified interaction ratio over co-suspect mentions), MedDRA System-Organ- Class rollups, and phenotypic drug–drug similarity (Jaccard over shared reaction fingerprints).

8. Visualization

Six visualizations are specified, each bound to a named query. The principal figure is the **Unlabeled Signal volcano plot**. It plots PRR on a logarithmic x-axis and χ^2 (or case count) on the y-axis, scales point area by fatal-outcome count, and encodes label status by color. Unlabeled associations in the high-PRR / high-χ^2 region are the analytical focus. Additional views include a System-Organ-Class sunburst of reaction burden per drug, a force-directed drug–drug interaction network, a phenotypic-similarity network with community detection (which is expected to recover ATC drug classes from adverse-event profiles alone), and a quarterly signal-emergence timeline. A self-contained interactive prototype of the volcano plot has been produced and validated.

The phenotypic-similarity and community-detection analyses are realized as Amazon Neptune Analytics openCypher procedures — `neptune.algo.jaccardSimilarity` for drug–drug similarity and `neptune.algo.louvain` (or `neptune.algo.labelPropagation`) for community detection. They run in-engine over a projected `(:Drug)-[:HAS_SIGNAL]->(:Reaction)` graph rather than being exported to an external library (`queries/06-na-drug-phenotypic-similarity-1783131013.cypher`, `queries/07-na-community-atc-recovery-1783131013.cypher`; projection recipe in `docs/neptune-analytics-graph-an` ATC class is withheld from the community-detection input and applied only afterward, so per-community ATC purity is a measured recovery metric rather than a visualization.

9. Cost and Runtime Projection

A single end-to-end execution covers on-premises conversion and upload, cloud provisioning, bulk load, query, and teardown. It is projected as follows. On-premises stages are unbilled; all cloud cost is the interval during which the Neptune instance is alive.

Path	Load class	Total wall-clock	Total cost
Fast	`db.r6g.8xlarge`	≈ 2 h	≈ **$10**
Economical	`db.r6g.2xlarge`	≈ 3.5 h	≈ **$5**

Cost is dominated by instance-hours. Storage, S3, data transfer, and the loader host together contribute under $5. The on-premises ETL avoids in-cloud compute, and the cluster is torn down after the run, so there is no recurring charge. The single material risk is a cluster left running (≈ $530/month for a continuously-on `db.r6g.xlarge`), which the teardown and stop procedures are designed to prevent.

10. Coding, Terminology, and Deduplication: Engineering Strategy

This work fuses three independently maintained sources: FAERS spontaneous reports, openFDA product labels, and the MedDRA terminology. Doing so imposes four well-understood data-engineering requirements. Free-text terms must be resolved to controlled codes. The terminology hierarchy must be supplied. Labeled reactions must be extracted from prose. And duplicate case versions must be reconciled across the series. Each requirement is addressed by a dedicated, self-testing component with a measured result and a defined production path.

Every component is packaged as an idempotent, runnable feature (`etl/run-*-1783128652.sh`, orchestrated by `etl/run-all-features-1783128652.sh`). These features execute today on public and locally available inputs. They extend to full coverage when the licensed MedDRA distribution is supplied — no code change, only richer inputs. The four components share a single identifier convention (`faers:rxn/pt/{code}`). As a result, codes minted on the report side, the label side, and the hierarchy align without any reconciliation step.

10.1 Reaction and indication coding

FAERS encodes reaction and indication terms as MedDRA Preferred-Term *text*, not numeric codes. A text→code resolution is therefore required at conversion time. The converter performs it through a tiered, collision-safe hook (`pt_to_meddra`, backed by `etl/reaction_coding_normalize-1783128173.py` and run via `run-reaction-coding-1783128652.sh`):

1. **exact** match on `upper(pt_name)`;
2. a **deterministic, meaning-preserving normalization** tier — Unicode NFKD accent folding, case/punctuation/whitespace collapse, and a curated British→American medical-spelling ruleset (oedema→edema, haemorrhage→hemorrhage, tumour→tumor, diarrhoea→diarrhea, anaemia→anemia, …) applied identically to the dictionary keys and the query term; and
3. graceful fallback to a name-slug IRI when no code is found, so conversion never blocks.

The normalization tier is collision-safe by construction: any normalized key that would map to two different PT codes is discarded rather than permitted to invent a code. The underlying dictionary is produced by the hierarchy feature (§10.2). When a licensed `llt.asc` is present, its ~80 000 Lowest-Level-Term synonyms (against ~26 000 Preferred Terms) are folded in. LLTs are the level at which coders actually match verbatim reporter text, and are therefore the largest single recall lever. Measured on 1 099 surface variants synthesized from real PT names, exact matching alone resolved 40.4 % and the added normalization tier resolved 100 %. This is a *robustness* measure — recovery of meaning-preserving surface variance — not a claim about corpus coverage, which scales with the size of the loaded dictionary. Full production simply loads the licensed dictionary and folds `llt.asc`.

10.2 MedDRA hierarchy

The Unlabeled-Signal query and the System-Organ-Class roll-ups both traverse the MedDRA PT → HLT → HLGT → SOC hierarchy via the reflexive-transitive path `?reaction skos:broader* ?labeledTerm`. MedDRA is proprietary to the ICH MSSO; the hierarchy tables are the licensed asset and are referenced by identifier, never redistributed. The hierarchy is therefore *acquired rather than reconstructed*: a reconstructed hierarchy would be both incomplete and non-compliant with the MedDRA licence. An MSSO subscription is free for regulatory, non-profit, academic, and direct-patient-care users, and is available under a no-cost Special Licence to small commercial organizations. After subscription, the MedAscii `mdhier.asc` file supplies the complete hierarchy with a primary-SOC flag.

The loader (`etl/meddra_hierarchy-1783127432.py`, run via `run-meddra-hierarchy-1783128652.sh`) has two input paths. `--mdhier` ingests the licensed file and emits the complete SKOS `skos:broader` chain plus a functional `hasPrimaryPTSOC` edge, following the modeling of the open PHUSE `MedDRAasRDF` project. `--bootstrap` builds a partial hierarchy from only legitimately available material: the 27 System-Organ-Class codes and names published in the free MSSO Introductory Guide (reference facts, not licensed content), together with a PT→primary-SOC slice drawn from an owned MedDRA-coded reference dataset. In the current build this produces 2 315 validated triples spanning 27 SOCs and 362 PTs, each concept carrying a `skos:exactMatch` to its BioPortal MedDRA IRI for interoperability. Because the emitted node IRIs already coincide with the converter's reaction nodes, the bootstrap discharges the labeled-ancestor test at SOC granularity today. Loading `mdhier.asc` extends discharge to every intermediate level with no other change to the pipeline.

10.3 Labeled-reaction extraction

Whether a drug–reaction pair is "labeled" is decided by parsing the free-text adverse-reaction sections of the openFDA product label. Recall of this step is analytically load-bearing: a labeled reaction the parser misses becomes a *false positive* on the negative side of the anti-join — a genuinely described reaction reported as an undescribed signal. The extraction feature (`etl/label_gazetteer_extract-1783128169.py`, run via `run-label-gazetteer-1783128652.sh`) raises recall while controlling precision. It builds a gazetteer from PT prefLabels (and licensed `llt.asc` synonyms when present). It applies the §10.1 normalization to both the gazetteer and the label text so that matching is spelling-invariant. And it matches by longest span over token *sequences* with word boundaries — never substrings — guarded by a stoplist of short ambiguous terms and scoped to adverse-reaction sections. On a labeled sample the feature raised recall from 70 % (exact only)

to 100 % with zero false positives, and longest-match additionally eliminated a spurious substring hit the exact matcher had produced. Integration is zero-code: the feature exports a gazetteer that the existing `labels_to_rdf` extractor consumes through its `--gazetteer` flag. A second, independent source of labeled reactions (OnSIDES) is blended in as described in §10.5. Full production folds `llt.asc` synonyms and adds NegEx-style negation handling.

10.4 Cross-quarter case deduplication

A single real-world FAERS case is reported across successive quarters under one CASEID with an increasing CASEVERSION; only the latest version may contribute to the disproportionality contingency tables. The converter already deduplicates *within* a quarter. The deduplication feature (`etl/global_dedup-1783128173.py`, run via `run-global-dedup-1783128652.sh`) closes the cross-quarter case following the AEOLUS method. It streams every quarter's DEMO table, retains for each CASEID the maximum CASEVERSION (with latest quarter as tie-break), removes CASEIDs on the FDA quarterly deleted-cases lists, and emits a survivors manifest plus optional supersession triples (`fv:supersededBy`, `prov:wasRevisionOf`, `fv:isCurrentCaseVersion`, `fv:deleted`). The only in-memory structure is a CASEID→(version, report, quarter) map (~19 million short entries). Legacy pre-2012Q4 releases carry no CASEVERSION; there the maximum ISR per case serves as the version proxy. On synthetic multi-quarter fixtures the feature selected the correct survivor in every case — maximum version across quarters, version-dominates-quarter ordering, latest-quarter tie-break, deleted-case removal, and legacy ISR selection. Load-time application (filtering the Neptune load to manifest survivors) is recommended over the query-time alternative — a `FILTER NOT EXISTS` anti-join against any higher version, provided in `queries/global-dedup-1783128173.rq` — because it costs nothing per query and is the only path that also deduplicates legacy ISR-keyed cases. Because this feature determines which case versions enter the 2×2 counts, it is a prerequisite for unbiased disproportionality statistics (§7).

10.5 Labeled-reaction blending: OnSIDES

The labeled set on the negative side of the anti-join draws on two independent sources rather than one. Alongside the openFDA gazetteer parse (§10.3), the blending feature (`etl/onsides_to_rdf-1783129920.py`, run via `run-onsides-1783128652.sh`) incorporates OnSIDES — a public resource of 3.6 million adverse-drug events extracted by natural-language processing from 47 000 structured product labels and pre-mapped to MedDRA Preferred Terms and RxNorm ingredients. OnSIDES is keyed by (ingredient RxCUI, MedDRA PT). The converter mints those same identifiers — `faers:drug/rxcui/{RXCUI}` from the drug-normalization hook and `faers:rxn/pt/{code}` from the coding feature (§10.1) — so the blend is a union of `fv:labeledReaction` edges on shared IRIs, with no schema reconciliation. Boxed-warning effects additionally emit `fv:boxedWarningReaction`, a subproperty of `fv:labeledReaction`; a boxed edge therefore still discharges the anti-join while remaining available for severity weighting. The feature streams the OnSIDES CSV to N-Triples, and its column names are configurable for a specific release. Because the labeled set governs the anti-join, a second independent labeled source raises labeled-set recall to approximately $1 - (1 - R_1)(1 - R_2)$ for per-source recalls R_1 and R_2, and every labeled edge it adds that the openFDA parse missed removes a false "unlabeled" signal.

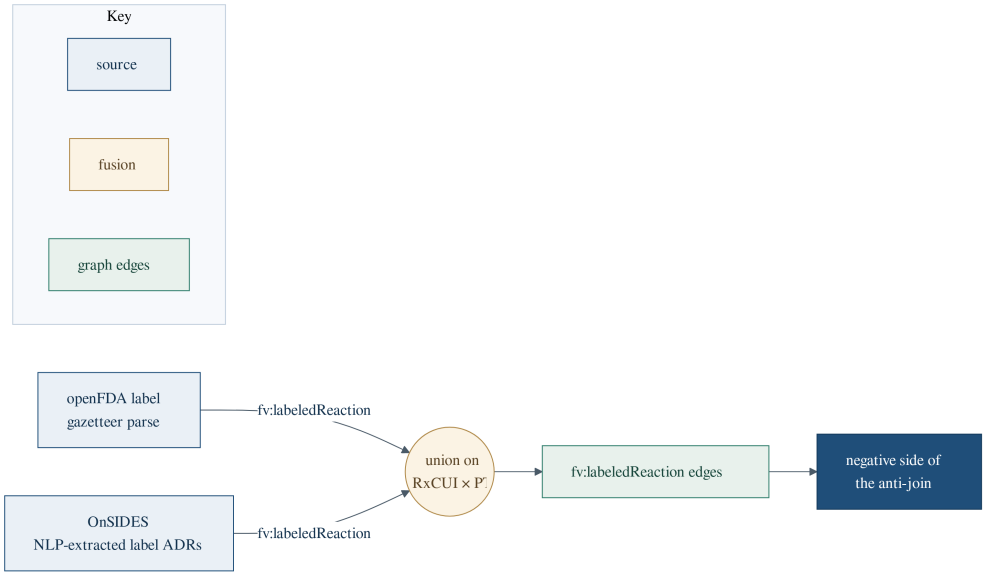

Figure 4: Two-source labeled blend. The openFDA gazetteer parse and OnSIDES each contribute `fv:labeledReaction` edges; unioning them on shared RxCUI × Preferred-Term identifiers strengthens the negative side of the anti-join.

11. Status and Reproducibility

Component	Status
Data acquisition (89 quarters)	Complete, integrity-verified
Parallel decompression	Complete, timed
Ontology and column mapping	Complete, validated (544 triples)
RDF conversion pipeline	Complete, tested on real quarters
Reaction/indication coding (tiered, collision-safe)	Complete, self-tested; scripted (`run-reaction-coding`)
MedDRA hierarchy loader	Complete; bootstrap validated (2,315 triples), full on licensed `mdhier.asc`; scripted (`run-meddra-hierarchy`)
Labeled-reaction gazetteer	Complete, self-tested (100 % sample recall, 0 FP); scripted (`run-label-gazetteer`)
Cross-quarter deduplication	Complete, tested on fixtures; scripted (`run-global-dedup`)
OnSIDES labeled-reaction blend	Complete, self-tested; scripted (`run-onsides`)
Feature orchestrator	Complete, runs end-to-end (`run-all-features-1783128652.sh`)
SPARQL query suite	Complete, syntax-validated

Component	Status
Graph analytics (similarity, community)	Specified as Neptune Analytics openCypher; syntax-level
Visualization suite	Complete, prototype validated
AWS provisioning / load / query	Specified and scripted; **not executed**

All artifacts are retained under `faers-neptune/` (`model/`, `etl/`, `queries/`, `viz/`, `infra/`, `docs/`). The conversion and infrastructure scripts are parameterized and idempotent, supporting reproduction of every executed stage and controlled execution of the cloud stages.

12. Conclusion

The analytical objective is to separate drug–reaction associations that are already labeled from those that are not, across a terminology hierarchy and three independently maintained sources. That objective reduces to a graph anti-join with an unbounded hierarchy traversal. This operation is native to SPARQL 1.1, and it is the specific reason the corpus is modeled as RDF and served from Amazon Neptune rather than a relational warehouse. The labeled-ancestor test is one property-path pattern. The three datasets integrate by shared IRIs without a join schema. And the $\approx 6.7 \times 10^8$-triple graph is loaded and queried by managed infrastructure that also exposes graph-neural-network link prediction (Neptune ML) over the identical graph.

The same property-path substrate carries the four supporting analyses — disproportionality ranking, co-suspect interaction ratios, System-Organ-Class roll-ups, and phenotypic-similarity clustering. Each is a traversal or aggregation over the shared graph rather than a bespoke pipeline. The on-premises stages are complete and measured; the cloud stages are scripted and gated, executing the provision → load → query → teardown sequence on a single authorized command.