

Healthcare Data Engineering and Interoperability

Study Guide

by Richard J. Young, Ph.D.

richard@deepneuro.ai

Chapter 2 of "Healthcare Analytics and AI: Building Systems That Actually Work"

The essentials: the standards your data actually lives in, and why extraction is most of the work.

One-sentence thesis. In healthcare AI, data engineering is not the warm-up before modeling, it is the majority of the work, and the model's performance ceiling is fixed by the extraction, mapping, and temporal alignment that happen before the first line of training code runs.

PART 1: Data Engineering Is the Job

The Two-Month Lesson

Begin with a schedule that was wrong by a factor of three. A skilled team was asked to build a 30-day readmission model with a clean target, six engineers, and full Epic access, and they estimated three weeks. It took eight weeks just to assemble the data: demographics in Caboodle, diagnoses in Clarity, labs in a separate repository, meds in Chronicles, and three different patient-ID systems to reconcile. The logistic regression took two days. The data took two months.

ANALOGY

Cooking a restaurant meal where the ingredients arrive unlabeled, in four different warehouses, measured in incompatible units. The recipe is trivial once everything is on the counter; getting it onto the counter is the job.

The lesson generalizes. If you only ever train on MIMIC-IV, where joins are documented and identifiers are clean, your first day on production hospital data will cost you months, not days, because every temporal filter and identity-resolution rule you write is a modeling decision, not plumbing.

Watch out. A join that looks right can silently mean the wrong thing. Convince yourself the assembled dataset means what you think it means before you trust an AUROC computed on it.

The Resource Graph

FHIR (Fast Healthcare Interoperability Resources, pronounced “fire”) is a RESTful API standard from HL7, and the mental model you need is a web, not a table. It stores data as **resources**: self-contained JSON objects with unique URLs that reference one another, linked like index cards pinned together rather than rows in one giant spreadsheet. A Patient is an anchor that holds no clinical data; every clinical resource points back to a Patient, and most point to a specific Encounter. That shape dictates how you model: most predictive systems are built at the encounter level, one row per Encounter with features aggregated from the linked resources.

⚠ WATCH OUT

Filtering matters as much as fetching. A Condition of “possible sepsis” later **refuted** is a completely different data point than a **confirmed** sepsis diagnosis. Training on all Conditions without filtering on `clinicalStatus` and `verificationStatus` is one of the most common errors in healthcare ML.

Even the structure needs checking: two nominally R4-compliant servers can expose the same LOINC

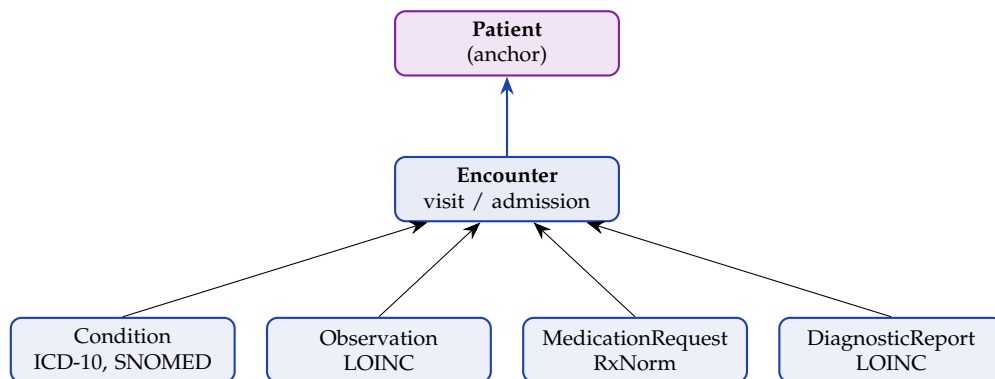
code as a Quantity on one and a CodeableConcept on another, so validate the shape of every resource before you assume its content.

Search, SMART, and CDS Hooks

Three layers turn the resource graph into a working pipeline. FHIR exposes a consistent search API (GET [base]/[resource]?param=value) that returns a **Bundle**, and the `_include` parameter pulls linked resources in one request instead of an N+1 storm of calls. **SMART on FHIR** adds the OAuth 2.0 security layer: a provider-facing launch carries patient context into the EHR, while a patient-facing launch is scoped to one patient. **CDS Hooks** fire your model at a specific workflow moment (patient-view, order-sign, patient-discharge) and give it a few hundred milliseconds to return cards.

CDS Hooks are where your model actually meets the clinician. A well-placed hook delivers the right data at the right moment; a hook that fires on every chart open becomes alert-fatigue noise, and that difference is workflow design, not algorithm quality.

Watch out. For population-scale pulls, use **Bulk FHIR** (asynchronous NDJSON export), not the interactive search API, which is rate-limited.



The FHIR reference graph. Every clinical resource points back to a Patient and usually to a specific Encounter; models are built at the encounter level, one row per Encounter.

Resource	Purpose	Code system	Anchors to
Patient	Demographics, identifiers	MRN/SSN	(root)
Encounter	Visit or admission	CPT, SNOMED	Patient
Condition	Diagnosis or problem	ICD-10-CM, SNOMED	Patient, Encounter
Observation	Vital, lab, score	LOINC, SNOMED	Patient, Encounter
MedicationRequest	Prescribing intent	RxNorm	Patient, Encounter
DiagnosticReport	Lab or imaging bundle	LOINC	Patient, Encounter

■ Pipe-Delimited and Everywhere

Before FHIR was the polished API, there was HL7v2, and in 2026 it still carries an estimated 80 percent or more of hospital ADT (admit/discharge/transfer), lab, and order feeds. If FHIR is the finish on the wall, HL7v2 is the copper pipe behind it, and the pipe carries most of the water. Messages are pipe-delimited, segment-oriented text in continuous use since 1989: MSH (header), EVN (event, A01 = admit, A03 = discharge), PID (patient), PV1 (visit). Field meaning is

defined by position, not by label. The real-time signal that a patient moved from ED to ICU arrives as an HL7v2 ADT message, so if you build a live monitoring system you will parse HL7v2 whether you like it or not.

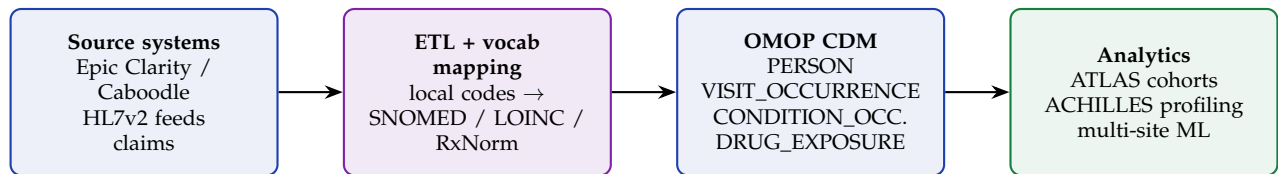
Watch out. HL7v2-to-FHIR mapping is **lossy**. A PV1 segment has dozens of fields; a FHIR Encounter keeps a subset. If you train on FHIR but serve on HL7v2, the feature-availability gap between training and serving is a source of silent error.

Relational, Vocabulary-Normalized

Where FHIR moves one patient's data around, OMOP is built to analyze a million patients at once, and its philosophy is the opposite of FHIR's. OMOP CDM (from the OHDSI collaborative, spanning 2 billion-plus patient records) is the analytics standard: where FHIR is document-oriented and you navigate a reference graph, OMOP is relational and you write JOINS across normalized tables (PERSON, VISIT_OCCURRENCE, CONDITION_OCCURRENCE, DRUG_EXPOSURE, MEASUREMENT, OBSERVATION). FHIR hands you labeled index cards one patient at a time; OMOP hands you a warehouse of aligned tables built for querying a million patients at once.

OMOP's real power is the **standardized vocabulary** layer. "Essential hypertension" coded as ICD-10 I10 at one hospital and ICD-9 401.1 at another both map to the same SNOMED concept, and that normalization is what makes multi-site studies possible. ACHILLES profiles a database (comparing its profile on your training versus deployment population is a direct read on distribution shift), and ATLAS builds portable cohort definitions that run identically at ten sites.

Watch out. Vocabulary mapping is imperfect. Local codes that do not map to a standard concept become clinical data your model silently ignores unless you handle the mapping failure explicitly. FHIR and OMOP are complementary, not rivals: FHIR exchanges data, OMOP analyzes it.



The OMOP pipeline. Extraction and vocabulary mapping normalize messy source systems into relational CDM tables; the standardized vocabulary layer is what lets a cohort definition run identically at ten sites.

TEFCA, QHINs, and the CMS Rule

Two regulatory shifts changed what data is actually reachable, not just what exists. **TEFCA** (operational 2025) designates Qualified Health Information Networks (QHINs: Epic Nexus, eHealth Exchange, CommonWell, KONZA) that move data across vendor lines, and the **CMS Interoperability and Patient Access rule** requires payers to expose claims and clinical data via FHIR APIs. For the first time, a query can approximate “show me everything in the system about this patient,” so a system designed in 2026 can assume FHIR access where a system designed in 2016 could not.

Watch out. “Actually accessible” still lags “legally accessible.” Payer FHIR maturity varies widely, and the claims exposed through a patient-access API are typically limited to what the payer has adjudicated, which may exclude denied claims, pending claims, and claims where the payer is secondary.

Where the Data Actually Lives

Real extraction means specific environments, each with its own shape. Epic **Clarity** is 15,000-plus relational tables mirroring Chronicles (the raw ingredients), while Epic **Caboodle** is a star-schema

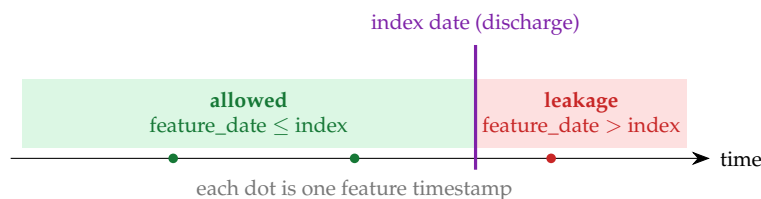
warehouse and the right place for cohort building (the prepped ingredients in labeled containers). Cerner **Millennium** makes everything a row in `CLINICAL_EVENT`, and `i2b2` (`OBSERVATION_FACT`) is great for cohort discovery but weak for building feature matrices.

Temporal Anchoring, or Leakage in SQL

The most common extraction bug is not a bad join, it is a time-travel bug. If a readmission-at-discharge model accidentally includes a lab drawn on day 2 of the readmission, you have built label leakage in SQL: the model peeks at the future it is supposed to predict. The defense is one clause on every feature query.

`feature_date ≤ index_date`

Watch out. Data infrastructure is a governance structure, not just a stack. EHR vendors gatekeep the APIs, health systems own but fragment the data, payers are newly required to share, and patients hold rights they cannot easily exercise. You cannot change this; you can build within it and document what the model was never shown.



Leakage is a temporal bug in SQL. Any feature dated after the index date leaks the future into training; every feature query needs a `WHERE feature_date ≤ index_date` clause.

Should You Build This?

Build it when a standard access pathway already exists for the data you need: FHIR APIs for patient-level reads, Bulk FHIR or a warehouse extract for cohorts, OMOP for multi-site research. The opening story is the business case, the model took two days and the data took two months, so pipeline investment pays back before any model does. **Think twice when** the plan depends on access you have not verified: vendor endpoints with unsupported search

parameters or rate limits, payer FHIR implementations of uneven maturity, identifier systems nobody has reconciled, or a pipeline with no temporal anchoring, because a leaky extraction flatters every model trained on it. **Who bears the risk.** Mis-joined records and silently lossy mappings surface downstream as wrong outputs attached to real patients, and clinicians act on them, so the builder’s obligation is to validate what arrives and document what the model was never shown.

1. A team has six engineers and full Epic access but still loses two months before the first model run. What consumed the time, and what does that imply about where model performance is actually determined?
2. Why is filtering FHIR Conditions on `clinicalStatus` and `verificationStatus` not optional?
3. You train a model on FHIR data but the deployment site feeds you HL7v2. Name the specific failure mode this creates and why.
4. In one sentence each, what does FHIR do that OMOP does not, and vice versa?
5. Write the WHERE clause that prevents label leakage in a discharge-time feature query, and explain what breaks without it.

If You Remember One Thing

The model is the fast part. FHIR, HL7v2, and OMOP are three different data layers with different shapes (reference graph, pipe-delimited feed, JOIN-driven warehouse), and the extraction, vocabulary mapping, and temporal anchoring you do across them set the ceiling on everything the model can achieve. A clean AUROC computed on a dataset you do not fully understand is not a result, it is a liability.

Richard J. Young, Ph.D. · richard@deepneuro.ai · Senior AI Research Scientist, UnitedHealth Group · UNLV Lee Business School · ORCID 0000-0002-1109-7552