

CircePy: Towards a Conformance Standard for Cohort Definitions in the OHDSI Ecosystem

Authors: James P. Gilbert¹, Egill Fridgeirsson², Christopher Knoll¹

¹ **Observational Health Data Analytics, Global Epidemiology, Janssen Research and Development, Titusville, NJ, USA**

² **Department of Medical Informatics Erasmus University Medical Center, Rotterdam, the Netherlands**

Background

Cohort definitions are foundational to observational research using the OMOP Common Data Model (CDM). The CIRCE-BE Java library, maintained by OHDSI, serves as the canonical engine for translating JSON cohort expressions into executable SQL and is the engine underlying ATLAS and WebAPI [1]. The Capr library provides equivalent functionality in R, offering a programmatic API that is fully interoperable with CIRCE-BE output, albeit through a distinct interface. Through this lens, circe can be viewed as a *de facto* standard for reliable, interoperable cohort definitions.

As the OHDSI community grows, multiple independent implementations of cohort SQL generation have emerged across programming languages and frameworks. While this diversity reflects healthy community engagement, it introduces a significant risk: if competing implementations diverge in their SQL output or interpretation of the CDM, the reproducibility and comparability of network studies is undermined. A single cohort definition could yield different patient populations depending on which tool generated the SQL, leading to inconsistent analytics potentially leading to erroneous conclusions.

In line with the OHDSI Technical Advisory Board (TAB) standards around concept sets [2], we argue that circe should be formally recognized as the reference standard for cohort generation, and that alternative implementations should demonstrate strict parity with its output. To test this principle in practice, we developed CircePy, a Python implementation of CIRCE-BE that maintains 1:1 field-level and SQL-output fidelity with the Java original [3]. In addition, we have developed an alternative execution engine within python using the ibis library, producing equivalent cohorts in an independent manner [4].

Methods

CircePy was developed using two complementary strategies:

Strategy 1: Direct T-SQL Translation: A line-by-line port of the Java CIRCE-BE SQL builder logic into Python, replicating the template-based SQL generation approach. This strategy produces T-SQL output identical to the Java implementation and serves as the primary parity benchmark.

Strategy 2: Ibis-Based Backend: An alternative backend leveraging the Ibis dataframe library [5] to construct cohort SQL using a database-agnostic expression API. This approach generates semantically equivalent queries that can target multiple SQL dialects (PostgreSQL, DuckDB, Spark, BigQuery, etc.) natively, without relying on OHDSI SqlRender for dialect translation.

Both strategies consume the same JSON cohort expression schema used by ATLAS and Java CIRCE-BE. Parity was validated through a comprehensive test suite of unit tests, including field-by-field comparison of all OMOP CDM domain builders (ConditionOccurrence, DrugExposure, VisitOccurrence, Measurement, etc.). A detailed Java-to-Python class mapping was maintained to track compatibility across all cohort expression fields, aliases, and types [3]. Figure 1 shows a demonstration of the circe cohort expression model, which is a 1 to 1 typed clone of the Java reference version. Benchmarks comparing execution time, SQL output length, and query plan equivalence across both strategies and the Java reference will be reported. For a comprehensive test all 707 cohort definitions from the OHDSI PhenotypeLibrary were executed against the Merative Commercial Claims and Encounters data set running on the databricks platform.

Results

Strategy 1 (Direct T-SQL Translation) was validated by generating T-SQL for all 707 PhenotypeLibrary cohort definitions and performing character-level comparison against Java CIRCE-BE output. After normalization of non-semantic differences (whitespace and line endings), the generated SQL was identical across all cohort definitions, encompassing the full breadth of cohort expression features including nested inclusion rules, correlated criteria with temporal windows, custom era collapse logic, and multi-domain primary criteria. During development, divergences attributable to language-level formatting conventions (e.g., numeric literal precision, clause ordering within subqueries) were identified and corrected to match the Java reference exactly.

This result establishes that cross-language character-level parity is practically achievable at scale, and provides the community with a second independent reference point against which future implementations can be validated. The resulting SQL remains fully compatible with SqlRender for downstream dialect translation, preserving integration with the existing OHDSI toolchain [3].

All cohort definitions from the Phenotype Library were executed against the CCAE dataset (~150M patients) on the Databricks platform using both R and Python implementations, targeting the same SQL Warehouse endpoint. Row-level output was verified identical across to verify that output definitions were functionally identical in both implementations.

Execution time comparisons revealed that the R stack completed the full benchmark in 15.3 hours (median 52.0s per cohort), while the Ibis backend completed in 24.6 hours (median 70.0s per cohort), yielding an overall R-to-Python time ratio of 0.62. The median per-cohort ratio was 0.70 (Python approximately 1.4× slower). R was faster for 82% of cohorts (581/707), while Python was faster for 18% (126/707). Some overhead is attributable to differences in connection management and query dispatch rather than SQL generation.

Conclusions

CircePy demonstrates that strict fidelity to a reference implementation is achievable across language boundaries without sacrificing developer flexibility. The dual-strategy architecture shows that innovation in how SQL is generated (e.g., via Ibis) can coexist with conformance to what SQL is generated, provided the reference standard is clearly defined. Though ibis engine performance was

generally slower, it has distinct advantages within context that allow ease of use of data frames within tools that would not be straightforward with the conventional SQL generation. The guarantee that both implementations use the same underlying data model representation allows confidence in population selection, regardless of context.

We advocate for the OHDSI community to formally adopt CIRCE-BE as the reference standard for cohort SQL generation and to establish a conformance testing framework that any alternative implementation must pass. This would preserve the ecosystem's openness to new tools and languages while safeguarding the reproducibility that underpins the distributed and collaborative environment OHDSI is founded on. Without such a standard, the proliferation of cohort generation tools risks creating a fragmented landscape where "the same cohort" means different things depending on the software used and interpretation becomes increasingly context dependent. Similarly, the conformance to a consistent standard encourages a formal framework to improve existing tools whilst maintaining cross platform compatibility.

One limitation of our current approach is that CIRCE-BE originated as an implementation for web application development and therefore should not be treated as a final or definitive specification. In future work, we plan to demonstrate that the cohort description language generalizes beyond this context. For example, to constructing custom features in other tools that remain directly traceable to cohort definitions. Accordingly, we will submit a proposed standard to the Technical Advisory Board [2] and strengthen testing and conformance requirements for package implementers.

This project demonstrates the potential for a standard for logical and temporal operations on the OMOP CDM and aim to publish this in formal manner, adopting versioning and interoperability guidelines. We also wish to call upon the community to improve the adoption, development and compliance towards software standards to reduce fragmentation and improve interoperability between software tools.

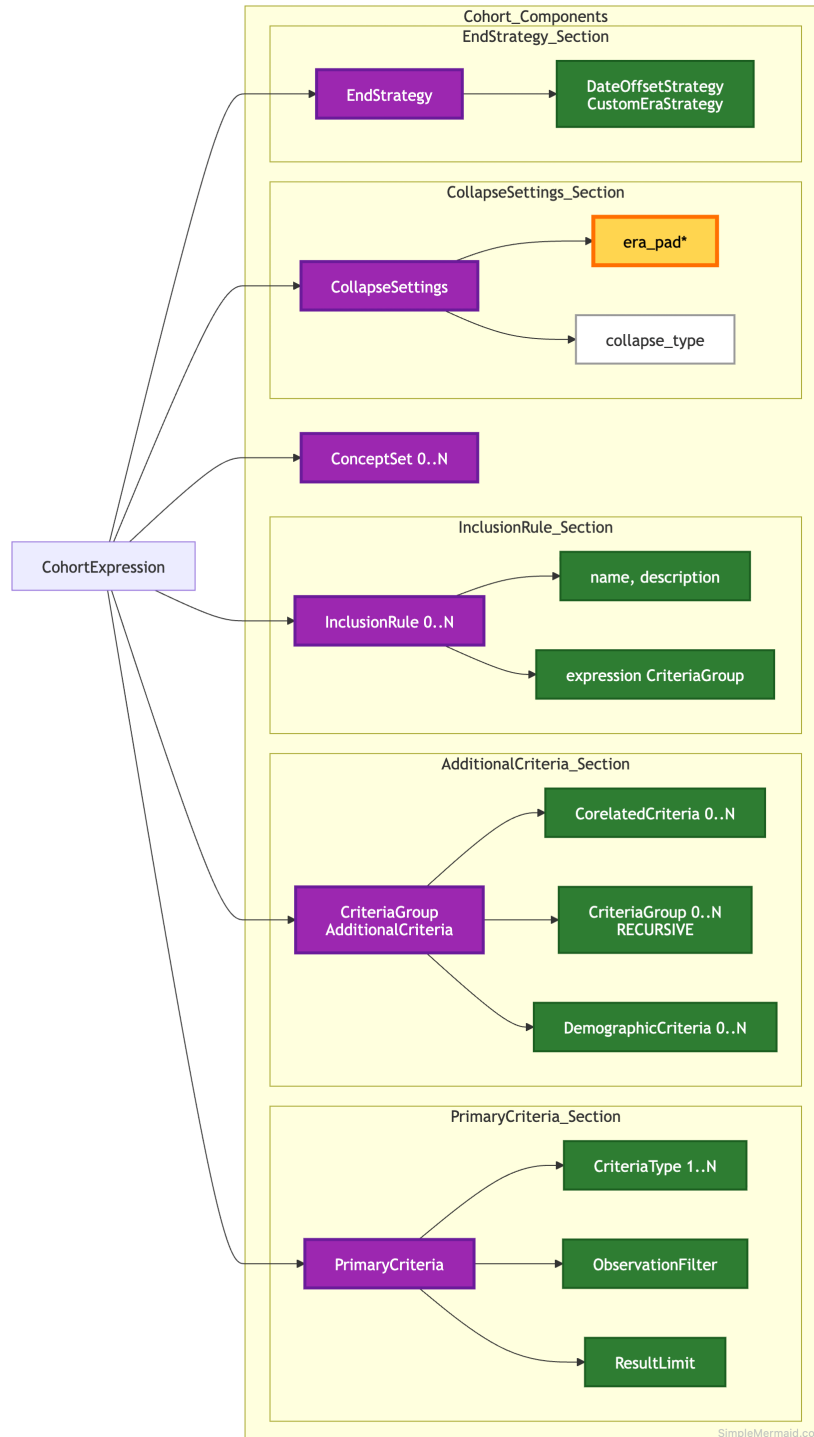


Figure 2: A class diagram of the circe expression schema. CircePy implements this schema as a typed Python class hierarchy as a one to one mirror of the Java reference version. The model supports 16 OMOP CDM domain criteria (excluded for brevity), each inheriting from the abstract `Criteria` base class. Temporal relationships between events are expressed through `CorelatedCriteria` with `Window` bounds, enabling arbitrary time-relative logic.

References

- [1] OHDSI. ATLAS (software). Observational Health Data Sciences and Informatics. GitHub repository. Available: <https://github.com/OHDSI/Atlas>
- [2] OHDSI Technical Advisory Board (TAB) Concept Set standard (Currently in DRAFT) Website Available: <https://ohdsi.github.io/TAB/Concept-Set-Specification.html>
- [3] OHDSI. CircePy: A one-to-one Python implementation of the circe-be cohort definition classes. GitHub repository. Available: <https://github.com/OHDSI/Circepy>. PyPI: <https://pypi.org/project/ohdsi-circepy/>
- [4] Fridgeirsson EA, Rijnbeek PR. Mitos: A Python implementation of OHDSI Circe cohort expressions with schema fidelity and cross-engine validation. Poster/abstract presented at: OHDSI Europe Symposium 2026; 2026.
- [5] Ibis Project. Ibis: the portable Python dataframe library. Available: <https://ibis-project.org/>. GitHub: <https://github.com/ibis-project/ibis>
- [6] OHDSI. circe-be: Cohort Inclusion and Restriction Criteria Engine (backend). GitHub repository. Available: <https://github.com/OHDSI/circe-be>
- [7] OHDSI. Capr: Cohort definition Application Programming in R. GitHub repository. Available: <https://github.com/OHDSI/Capr>. Documentation: <https://ohdsi.github.io/Capr/>