dbt for OMOP Phase 1.5: dqdbt

Katy Sadowski[1], Lawrence Adams[2], Thomas Wylie[3]

[1]Boehringer Ingelheim Pharmaceuticals, Inc., Ridgefield, CT, [2]London Secure Data Environment (SDE), UK,[123]OHDSI

Background

High-quality real-world data is the foundation of reliable observational research. It is therefore critical to identify and address data quality issues before publishing a dataset for research use. Data quality issues can stem from multiple sources, including inherent characteristics of the data source (e.g., data entry errors in an electronic health record system) and errors in data processing.

The conversion of a dataset into the OMOP Common Data Model (CDM) may introduce such errors if data quality is not properly assessed throughout the extract, transform, and load (ETL) process. The OHDSI community makes available several tools for monitoring ETL quality, including the **DataQualityDashboard** (DQD).[1]

DQD is an R package offering a comprehensive suite of quality checks based on the Kahn framework, many of which can be used to expose errors in the ETL process.[2] OMOP ETL developers and data users are encouraged to run DQD on their OMOP CDM to ensure the CDM is high-quality and compliant with the OMOP specification.[3] However, due to its design as a standalone R package, DQD is generally only used after a full run of the ETL is complete. This approach risks allowing quality issues to propagate through the ETL pipeline, delaying detection and complicating remediation.

Inspired by established best practices for shifting testing "left" in software and data pipelines, with **dqdbt** we aim to integrate DQD's standard quality checks natively within the OMOP ETL pipeline. To demonstrate this concept, we leverage Data Build Tool (dbt), an open-source data transformation tool with built-in testing features that enable the detection of anomalies at runtime throughout the ETL development process.[4]

This project represents a continuation of "dbt for OMOP Phase I", in which we introduced **dbt-synthea**.[5,6] dbt-synthea is an open-source OMOP ETL project which showcases how dbt can be used to develop an OMOP ETL and promotes a set of ETL development principles we believe will make OMOP ETL easier and more robust.

Methods

Twelve DQD check types were implemented in the dbt-synthea project using **dbt test**, dbt's framework for validating the contents and structure of the data tables it creates. dbt tests are defined in YAML and implemented as parameterized SQL queries.

To do so, we extended a Python script, **generate_dbt_yaml**, which auto-generates the YAML schema files used to define the OMOP tables in dbt-synthea. The script was updated to retrieve field- and table-level specifications from the OMOP Common Data Model and DQD GitHub repositories. These specification files are parsed and merged by table and field name to create a YAML representation of the documentation

Figure 1. dbt tests for condition_occurrence.condition_concept_id

```yaml
- name: condition_concept_id
  description: The CONDITION_CONCEPT_ID field is recommended for primary use
  in analyses, and must
    be used for network studies. This is the standard concept mapped from
    the source value which
    represents a condition
  data_type: integer
  tests:
    - not_null
    - dbt_utils.relationships_where:
        to: ref('concept')
        field: concept_id
        from_condition: condition_concept_id <> 0
        to_condition: domain_id = 'Condition'
    - concept_record_completeness:
        threshold: '5'
    - dbt_expectations.expect_column_to_exist
    - dbt_expectations.expect_column_values_to_be_in_type_list:
        column_type_list: "{{ get_type_variants(['bigint', 'integer']) }}"
    - dbt_utils.relationships_where:
        to: ref('concept')
        field: concept_id
        from_condition: condition_concept_id != 0
        to_condition: standard_concept = 'S' AND invalid_reason IS NULL
```

and quality checks for each table.

The DQD checks were implemented in dbt (**Table 1**) using a combination of built-in dbt tests, macros from the **dbt_expectations** package, and custom Jinja-based SQL macros.[7]

Table 1. DQD Checks Implemented in dbt-synthea

DQD Check Type	Description	dbt test implementation
cdmTable	Checks for presence of each OMOP CDM table in the database.	dbt_expectations.expect_column_to_exist
cdmField	Checks for presence of each OMOP CDM column in the database.	dbt_expectations.expect_column_to_exist
cdmDatatype	Checks that each column has the correct data type per the OMOP CDM specification.	dbt_expectations. expect_column_values_to_be_in_type_list NB: The DQD implementation of this check only verifies if integer columns contain digits; in dbt all columns' data types are checked.
isPrimaryKey	Checks that primary key columns contain only unique values.	unique (built-in test)
isForeignKey	Checks that each value in a foreign key column exists in the column it references.	relationships (built-in test)
isRequired	Checks that there are no NULL values in required columns.	not_null (built-in test)
fkDomain	Checks that each column which is restricted to concepts from a specific domain only contains concepts from that domain.	relationships_where (built-in test)
fkClass	Checks that each column which is restricted to concepts from a specific class only contains concepts from that class.	relationships_where (built-in test)
measurePersonCompleteness	Measures the percent of persons in the database with no rows in a given table. Fails if the percentage exceeds a specific threshold.	custom SQL macro
isStandardValidConcept	Checks that all non-zero concepts in a standard concept column are standard and valid.	relationships_where (built-in test)
standardConceptRecordCompleteness	Measures the percent of records in a table with a value of 0 in a given standard concept column. Fails if the percentage exceeds a specific threshold.	custom SQL macro
sourceConceptRecordCompleteness	Measures the percent of records in a table with a value of 0 in a given source concept column. Fails if the percentage exceeds a specific threshold.	custom SQL macro

These included table- and field-level checks for both conformance (e.g., required fields, key constraints) and completeness (e.g., standard concept coverage). For checks with a non-zero failure threshold, default threshold values were assigned based on the DQD specification. We were able to enhance the cdmDatatype check beyond DQD's capabilities by using dbt macros designed to handle database-specific type variations.

This automated and standards-driven approach ensures that our dbt schema files remain synchronized with the latest OMOP and DQD specifications and allows quality checks to be embedded continuously within the ETL lifecycle. No manual updates or separate configuration are required, making the framework portable and reproducible across any dbt-based OMOP ETL implementation.

Results

All conformance checks and the majority of completeness checks from the DataQualityDashboard were successfully implemented as tests in dbt-synthea, resulting in a total of 1,601 individual tests across 12 check types.

Several quality issues in dbt-synthea were identified and remedied as a result of the addition of these tests (**Table 2**). Running dbt test is a standard part of the dbt development workflow; developers making changes in dbt-synthea now have instant visibility into the impact of their changes on the quality of the OMOP CDM. dbt test will also be included in the automated GitHub Actions workflow for pull requests in dbt-synthea, such that any test failures will block the merge of new code into the main branch of the repository.

Table 2. DQD Check Failures Identified and Fixed in dbt-synthea

DQD Check Type	Number of Failures	Solutions
cdmField	1	Fixed column name typo in OMOP model SQL.
cdmDatatype	17	Enforced consistent type casting of source data by implementing standardized, parameterized scripts for loading Synthea data. Added type casts and truncated varchars in OMOP models where necessary.
isStandardValidConcept	1	Replaced hard-coded non-standard type concept with appropriate standard type concept.
standardConceptRecordCompleteness	2	Replaced hard-coded values of 0 in two non-required concept ID columns.

To validate that the DQD checks were correctly specified in dbt, the DataQualityDashboard R package was run on the OMOP CDM output by dbt-synthea's built-in duckdb ETL. After the above failures were addressed, no failures were raised in DQD for any of the 12 check types added as dbt tests.

Conclusion

Integrating DQD logic into dbt's build process transforms data quality assessment from a post-ETL task into a proactive, continuous practice for ETL developers. This integration enables early detection of quality issues, accelerates feedback loops, and enhances trust in the ETL. By automating validation at build time and maintaining parity with the comprehensive DQD suite, we propose a scalable, robust pattern for ETLs across

the OHDSI community.

Our YAML generation script is publicly available in the dbt-synthea repository and can be used to embed core DQD checks into any dbt-based OMOP ETL. Updates are planned in the near future to include the full set of DQD checks.

Future work will expand on this concept by incorporating comprehensive quality checks at the source data level within the ETL. dbt tests can be applied to any model in the pipeline, including staging models representing the raw source data undergoing transformation.[8] Visibility into source data quality is essential for a complete understanding of the quality of an OMOP CDM. Since the DataQualityDashboard validates only the final product of an ETL, relying solely on its output may allow some quality issues to remain undetected or lead to lengthy root cause analyses when checks fail.

dbt's testing capabilities facilitate proactive data quality monitoring and improvement. OMOP ETL developers, whether or not they currently use dbt, are encouraged to adopt this principle of continuous validation in their work. Raising data integrity standards early in the pipeline will enhance trust in OMOP data and reduce the amount of time spent on quality assessment.

References

1. DataQualityDashboard [Internet]. OHDSI; Available from: https://github.com/OHDSI/DataQualityDashboard
2. Kahn MG, Callahan TJ, Barnard J, Bauck AE, Brown J, Davidson BN, et al. A harmonized data quality assessment terminology and framework for the secondary use of electronic health record data. EGEMS (Wash DC). 2016 Sep 11;4(1):1244.
3. Observational Medical Outcomes Partnership Common Data Model [Internet]. OHDSI; Available from: https://github.com/OHDSI/CommonDataModel
4. dbt-core [Internet]. dbt Labs; Available from: https://github.com/dbt-labs/dbt-core
5. Sadowski K, Chandrabalan V, Adams L, Bouras A, Burrows E, Carlson R. dbt for OMOP Phase I: dbt-synthea. In: OHDSI Global Symposium Collaborator Showcase [Internet]. 2024. Available from: https://www.ohdsi.org/wp-content/uploads/2024/10/124-Sadowski-dbt-synthea-Abstract-Julien-Nakache.pdf
6. dbt-synthea [Internet]. OHDSI; Available from: https://github.com/OHDSI/dbt-synthea
7. dbt_expectations [Internet]. Metaplane; Available from: https://hub.getdbt.com/metaplane/dbt_expectations/latest/
8. How we structure our dbt projects [Internet]. dbt Labs. [cited 2025 Jun 23]. Available from: https://docs.getdbt.com/best-practices/how-we-structure