A New SQLAlchemyCollector for Emitting Airflow Lineage as DAGs Run

Michael Robinson, Community Manager at Astronomer
Michael Robinson

Email: michael.robinson@astronomer.io
LinkedIn: https://www.linkedin.com/in/michael-robinson/
GitHub: https://github.com/merobi-hub
Project PR: https://github.com/OpenLineage/OpenLineage/pull/2088
1. Why build a SQLAlchemy integration in OpenLineage?
2. SQLAlchemy
   ○ Background & design
3. SQLAlchemy ORM
   ○ Available metadata from database operations
4. OpenLineage-SQLAlchemy Integration Prototype
   ○ Design
   ○ Use cases
   ○ Implementation
5. Integration Output Examples
Why?

Do you know where your SQL queries are?

OpenLineage

Mission
To define an open standard for the collection of lineage metadata from pipelines as they are running.
Before OpenLineage

Analysis Tools

Schedulers

Warehouses

SQL Engines

Amundsen

EGERIA

MARQUEZ

Apache Atlas
With OpenLineage

- Analysis Tools
- Schedulers
- Warehouses
- SQL Engines

OpenLineage

- Amundsen
- EGERIA
- MARQUEZ
- Apache Atlas
SQLAlchemy
SQLAlchemy is the Python SQL toolkit and Object Relational Mapper that gives application developers the full power and flexibility of SQL.

It provides a full suite of well known enterprise-level persistence patterns, designed for efficient and high-performing database access, adapted into a simple and Pythonic domain language.
**Architecture**: Core + separate ORM

**Guiding philosophy**:
- expose the “R” in ORM
- support abstraction but don’t enable mystification
- the ORM is intentionally redundant for most use cases

A relational database provides **rich, set-based functionality that should be fully exposed**. SQLAlchemy’s ORM provides an open-ended set of patterns that allow a developer to construct a **custom mediation layer** between a domain model and a relational schema, turning the so-called "object relational impedance" issue into a distant memory.

With SQLAlchemy, there’s no such thing as "the ORM generated a bad query" - **you retain full control over the structure of queries**, including how joins are organized, how subqueries and correlation is used, what columns are requested. **Everything SQLAlchemy does is ultimately the result of a developer-initiated decision.**

**Don’t use an ORM if the problem doesn’t need one.** SQLAlchemy consists of a Core and separate ORM component. The Core offers a full SQL expression language that allows Pythonic construction of SQL constructs that render directly to SQL strings for a target database, returning result sets that are essentially enhanced DBAPI cursors.
Since SQLAlchemy 1.4, all ORM (Object Relational Mapper) mappings derive from a registry of mapped classes.

A common use case for the ORM is automating database operations.

The ORM also offers event tracking capability through session events:
  - these Object Lifecycle Events track when objects are added, deleted or persisted in sessions.

### Object Lifecycle Events
- after_bulk_update
- after_bulk_delete
- after_commit
- before_commit
- after_insert
- ...

```python
from sqlalchemy import event

@event.listens_for(SomeSessionClassOrObject, 'after_commit')
def receive_after_commit(session):
    "listen for the 'after_commit' event"

    # ... (event handling logic) ... 
```
Some of the ORM metadata available

The ORM’s execute_state offers multiple hooks containing columns:

- `all ORM descriptors.items()`
- `attrs.items()`
- `column_attrs.items()`
- `columns`

queries:

- `statement`

tables:

- `tables`
OpenLineage-SQLAlchemy Prototype
SQLAlchemyCollector and OpenLineageAdapter classes for:

1. listening for SQLAlchemy events using the SQLAlchemy ORM module
   - `orm_execute_state.all_mappers` include:
     - columns object
     - tables object

2. adapting the events to the OpenLineage spec with the OpenLineage Common Integration

---

OpenLineage Classes

- Dataset
- Run
- RunEvent
- RunState
- Job
- SqlJobFacet

SQLAlchemy Hooks

inspect() allows us to get tables and columns from `orm.execute.state.all_mappers`
Apache Airflow testing and debugging
- Metadata about internal database operations on xcom, task_instance, dag, and additional datasets

Web application design and development
- Web developer: new persona for OpenLineage+Marquez
- Debugging: diagnose broken apps, identify compromised datasets, jobs
- Database design? (enabled by static lineage capability, new in OpenLineage 1.0)

Flask
A micro Web framework written in Python. Flask-SQLAlchemy is a commonly used backend.

Bottle
A lightweight, simple WSGI framework in Python. Reportedly plays well with SQLAlchemy.

Apache Airflow
Airflow™ is a platform to programmatically author, schedule and monitor workflows.
OpenLineage-SQLAlchemy Use Cases

- Developing, testing and debugging Apache Airflow internals
  - Start time, completion time, SQL queries, runstate, more
  - Lineage of datasets and jobs (WIP)
  - Metadata emitted about:
    - xcom
    - task_instance
    - rendered_task_instance
    - dag
    - serialized_dag
    - dag_owner_attributes
    - dag_run
    - dag_warning
    - import_error
    - job
    - log_template
    - slot_pool
after_transaction_create(session, transaction)

listener.collect_metadata() → adapter.create_events() → receive_do_orm_execute(orm_execute_state) → listener.assemble_datasets()
Airflow runtime metadata available from the integration include rendered SQL queries:

```
SELECT task_instance_try_number AS task_instance_try_number, task_instance.task_id AS task_instance_task_id, task_instance.dag_id AS task_instance_dag_id, task_instance.run_id AS task_instance_run_id, task_instance.map_index AS task_instance_map_index, task_instance.start_date AS task_instance_start_date, task_instance.end_date AS task_instance_end_date, task_instance.duration AS task_instance_duration, task_instance.state AS task_instance_state, task_instance.max_tries AS task_instance_max_tries, task_instance.hostname AS task_instance_hostname, task_instance.unixname AS task_instance_unixname, task_instance.job_id AS task_instance_job_id, task_instance.pool AS task_instance_pool, task_instance.pool_slots AS task_instance_pool_slots, task_instance_queue AS task_instance_queue, task_instance.priority_weight AS task_instance_priority_weight, task_instance.operator AS task_instance_operator, task_instance.queued_dttm AS task_instance_queued_dttm, task_instance.queued_by_job_id AS task_instance_queued_by_job_id, task_instance.pid AS task_instance_pid, task_instance.executor_config AS task_instance_executor_config, task_instance.updated_at AS task_instance_updated_at, task_instance.external_executor_id AS task_instance_external_executor_id, task_instance.trigger_id AS task_instance_trigger_id, task_instance.trigger_timeout AS task_instance_trigger_timeout, task_instance.next_method AS task_instance_next_method, task_instance.next_kwarg AS task_instance_next_kwarg, dag_run_1.state AS dag_run_1_state, dag_run_1.id AS dag_run_1_id, dag_run_1.dag_id AS dag_run_1_dag_id, dag_run_1.queued_at AS dag_run_1_queued_at, dag_run_1.execution_date AS dag_run_1_execution_date, dag_run_1.start_date AS dag_run_1_start_date, dag_run_1.end_date AS dag_run_1_end_date, dag_run_1.run_id AS dag_run_1_run_id, dag_run_1.creating_job_id AS dag_run_1_creating_job_id, dag_run_1.external_trigger AS dag_run_1_external_trigger, dag_run_1.run_type AS dag_run_1_run_type, dag_run_1.conf AS dag_run_1_conf, dag_run_1.data_interval_start AS dag_run_1_data_interval_start, dag_run_1.data_interval_end AS dag_run_1_data_interval_end, dag_run_1.last_scheduling_decision AS dag_run_1_last_scheduling_decision, dag_run_1.run_hash AS dag_run_1_run_hash, dag_run_1.log_template_id AS dag_run_1_log_template_id, dag_run_1.updated_at AS dag_run_1_updated_at ... FROM task_instance JOIN dag_run AS dag_run_1 ON dag_run_1.dag_id = task_instance.dag_id AND dag_run_1.run_id = task_instance.run_id WHERE task_instance.dag_id = :dag_id_1 AND task_instance.run_id = :run_id_1 AND task_instance.task_id = :task_id_1 AND task_instance.map_index = :map_index_1
```
log_template

SELECT log_template.id AS log_template_id, log_template.filename AS log_template_filename, log_template.elasticsearch_id AS log_template_elasticsearch_id, log_template.created_at AS log_template_created_at FROM log_template WHERE log_template.id = :pk_1

dag_run

SELECT dag_run.state AS dag_run_state, dag_run.id AS dag_run_id, dag_run.dag_id AS dag_run_dag_id, dag_run.queued_at AS dag_run_queued_at, dag_run.execution_date AS dag_run_execution_date, dag_run.start_date AS dag_run_start_date, dag_run.end_date AS dag_run_end_date, dag_run.run_id AS dag_run_run_id, dag_run.creating_job_id AS dag_run_creating_job_id, dag_run.external_trigger AS dag_run_external_trigger, dag_run.run_type AS dag_run_run_type, dag_run.conf AS dag_run_conf, dag_run.data_interval_start AS dag_run_data_interval_start, dag_run.data_interval_end AS dag_run_data_interval_end, dag_run.last_scheduling_decision AS dag_run_last_scheduling_decision, dag_run.dag_hash AS dag_run_dag_hash, dag_run.log_template_id AS dag_run_log_template_id, dag_run.updated_at AS dag_run_updated_at FROM dag_run WHERE dag_run.dag_id = :dag_id_1 AND dag_run.execution_date < :execution_date_1 AND dag_run.state = :state_1 ORDER BY dag_run.execution_date DESC LIMIT :param_1
LogTemplate dataset columns

[('id', <ColumnProperty at 0xffffa0766a40; id>), ('filename', <ColumnProperty at 0xffffa05be40; filename>), ('elasticsearch_id', <ColumnProperty at 0xffffa05be740; elasticsearch_id>), ('created_at', <ColumnProperty at 0xffffa05be840; created_at>)]

DagRun dataset columns

[('_state', <ColumnProperty at 0xffffa05be940; _state>), ('task_instances', <RelationshipProperty at 0xffffa068840; task_instances>), ('dag_model', <RelationshipProperty at 0xffffa068850; dag_model>), ('dag_run_note', <RelationshipProperty at 0xffffa06885c0; dag_run_note>), ('state', <SynonymProperty at 0xffffa05c290; state>), ('id', <ColumnProperty at 0xffffa05bea40; id>), ('dag_id', <ColumnProperty at 0xffffa05beb40; dag_id>), ('queued_at', <ColumnProperty at 0xffffa05bec40; queued_at>), ('execution_date', <ColumnProperty at 0xffffa05bed40; execution_date>), ('start_date', <ColumnProperty at 0xffffa05bee40; start_date>), ('end_date', <ColumnProperty at 0xffffa05bef40; end_date>), ('run_id', <ColumnProperty at 0xffffa05bff40; run_id>), ('creating_job_id', <ColumnProperty at 0xffffa069c240; creating_job_id>), ('external_trigger', <ColumnProperty at 0xffffa069c340; external_trigger>), ('run_type', <ColumnProperty at 0xffffa069c440; run_type>), ('conf', <ColumnProperty at 0xffffa069c540; conf>), ('data_interval_start', <ColumnProperty at 0xffffa069c640; data_interval_start>), ('data_interval_end', <ColumnProperty at 0xffffa069c740; data_interval_end>), ('last_scheduling_decision', <ColumnProperty at 0xffffa069c840; last_scheduling_decision>), ('dag_hash', <ColumnProperty at 0xffffa069c940; dag_hash>), ('log_template_id', <ColumnProperty at 0xffffa069ca40; log_template_id>), ('updated_at', <ColumnProperty at 0xffffa069cb40; updated_at>), ('consumed_dataset_events', <RelationshipProperty at 0xffffa069cc40; consumed_dataset_events>), ('creating_job', <RelationshipProperty at 0xffffa069cd40; creating_job>), ('serialized_dag', <RelationshipProperty at 0xffffa069ce40; serialized_dag>)]
## DATASETS

<table>
<thead>
<tr>
<th>NAME</th>
<th>NAMESPACE</th>
<th>SOURCE</th>
<th>UPDATED AT</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>dag</td>
<td>airflow</td>
<td>airflow</td>
<td>Sep 06, 2023 07:33am</td>
<td>N/A</td>
</tr>
<tr>
<td>dag_code</td>
<td>airflow</td>
<td>airflow</td>
<td>Sep 06, 2023 07:33am</td>
<td>N/A</td>
</tr>
<tr>
<td>dag_owner_attributes</td>
<td>airflow</td>
<td>airflow</td>
<td>Sep 06, 2023 07:33am</td>
<td>N/A</td>
</tr>
<tr>
<td>dag_run</td>
<td>airflow</td>
<td>airflow</td>
<td>Sep 06, 2023 07:33am</td>
<td>N/A</td>
</tr>
<tr>
<td>dag_warning</td>
<td>airflow</td>
<td>airflow</td>
<td>Sep 06, 2023 07:33am</td>
<td>N/A</td>
</tr>
<tr>
<td>import_error</td>
<td>airflow</td>
<td>airflow</td>
<td>Sep 06, 2023 07:33am</td>
<td>N/A</td>
</tr>
<tr>
<td>job</td>
<td>airflow</td>
<td>airflow</td>
<td>Sep 06, 2023 07:33am</td>
<td>N/A</td>
</tr>
<tr>
<td>log_template</td>
<td>airflow</td>
<td>airflow</td>
<td>Sep 06, 2023 07:33am</td>
<td>N/A</td>
</tr>
<tr>
<td>rendered_task_instance_fields</td>
<td>airflow</td>
<td>airflow</td>
<td>Sep 06, 2023 07:33am</td>
<td>N/A</td>
</tr>
<tr>
<td>serialized_dag</td>
<td>airflow</td>
<td>airflow</td>
<td>Sep 06, 2023 07:33am</td>
<td>N/A</td>
</tr>
<tr>
<td>slot_pool</td>
<td>airflow</td>
<td>airflow</td>
<td>Sep 06, 2023 07:33am</td>
<td>N/A</td>
</tr>
<tr>
<td>task_instance</td>
<td>airflow</td>
<td>airflow</td>
<td>Sep 06, 2023 07:33am</td>
<td>N/A</td>
</tr>
<tr>
<td>xcom</td>
<td>airflow</td>
<td>airflow</td>
<td>Sep 06, 2023 07:33am</td>
<td>N/A</td>
</tr>
<tr>
<td>NAME</td>
<td>TYPE</td>
<td>DESCRIPTION</td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------------</td>
<td>---------------</td>
<td>---------------------------</td>
<td></td>
<td></td>
</tr>
<tr>
<td>dag_id</td>
<td>VARCHAR(250)</td>
<td>no description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>root_dag_id</td>
<td>VARCHAR(250)</td>
<td>no description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>is_paused</td>
<td>BOOLEAN</td>
<td>no description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>is_subdag</td>
<td>BOOLEAN</td>
<td>no description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>is_active</td>
<td>BOOLEAN</td>
<td>no description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>last_parsed_time</td>
<td>TIMESTAMP</td>
<td>no description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>last_pickled</td>
<td>TIMESTAMP</td>
<td>no description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>last_expired</td>
<td>TIMESTAMP</td>
<td>no description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>scheduler_lock</td>
<td>BOOLEAN</td>
<td>no description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pickle_id</td>
<td>BOOLEAN</td>
<td>no description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>fileloc</td>
<td>VARCHAR(2000)</td>
<td>no description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>processor_subdir</td>
<td>VARCHAR(2000)</td>
<td>no description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>owners</td>
<td>VARCHAR(2000)</td>
<td>no description</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Integration Events Consumed by Marquez

<table>
<thead>
<tr>
<th>ID</th>
<th>State</th>
<th>Name</th>
<th>Namespace</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>950e3521-95ef-4e93-be67-ad10fa720197</td>
<td>COMPLETE</td>
<td>airflow.SELECT dag_warning.dag_id AS dag_warning_dag_id, dag_warning.dag_id AS dag_warning.dag_id, dag_warning.warning_type AS dag_warning_warning_type, dag_warning.message AS dag_warning_message, dag_warning.timestamp AS dag_warning_timestamp FROM dag_warning WHERE dag_warning.dag_id IN (<strong>(POSTCOMPILE_dag_id</strong>))</td>
<td>airflow</td>
<td>Sep 06, 2023 07:37am</td>
</tr>
</tbody>
</table>

```
{  "root": {    "eventType": "COMPLETE",    "eventTime": "2023-09-06T11:37:26.887748"  },  "run": {  "runId": "950e3521-95ef-4e93-be67-ad10fa720197",  "facets": []  },  "job": {    "namespace": "airflow",    "name":      airflow.SELECT dag_warning.dag_id AS dag_warning_dag_id, dag_warning.dag_id AS dag_warning_dag_id, dag_warning.warning_type AS dag_warning_warning_type, dag_warning.message AS dag_warning_message, dag_warning.timestamp AS dag_warning_timestamp FROM dag_warning WHERE dag_warning.dag_id IN (__(POSTCOMPILE_dag_id__))"  },  "inputs": {    "0": {      "namespace": "airflow",      "name": "dag_warning",      "facets": []    }  },  "inputFacets": null,  "outputFacets": null}
```
```
"sourceCodeLocation" : null

"sql" : { 3 items
  "_producer" : https://github.com/OpenLineage/OpenLineage/tree/0.29.2/integration/airflow
  "schemaURL" : https://raw.githubusercontent.com/OpenLineage/OpenLineage/main/spec/OpenLineage.json#/definitions/SqlJobFacet
  "query" : "SELECT dag_warning.dag_id AS dag_warning_dag_id, dag_warning.warning_type AS dag_warning_warning_type,
            dag_warning.message AS dag_warning_message, dag_warning.timestamp AS dag_warning_timestamp FROM dag_warning WHERE
            dag_warning.dag_id IN (__[POSTCompile_dag_id_1])"
}

"inputs" : [ 1 item
    0 : { 5 items
      "namespace" : "airflow"
      "name" : "dag_warning"
      "facets" : {...} 7 items
      "inputFacets" : null
      "outputFacets" : null
    }
```

SELECT "user".id AS user_id, "user".email AS user_email, "user".password AS user_password
FROM "user"
WHERE "user".id = :pk_1

Sep 03, 2023 07:54am
# Integration Events Consumed by Marquez

<table>
<thead>
<tr>
<th>NAME</th>
<th>NAMESPACE</th>
<th>SOURCE</th>
<th>UPDATED AT</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>book</td>
<td>library</td>
<td>library</td>
<td>Sep 03, 2023 07:57am</td>
<td>N/A</td>
</tr>
<tr>
<td>book_history</td>
<td>library</td>
<td>library</td>
<td>Sep 03, 2023 07:57am</td>
<td>N/A</td>
</tr>
<tr>
<td>post</td>
<td>library</td>
<td>library</td>
<td>Sep 03, 2023 07:58am</td>
<td>N/A</td>
</tr>
<tr>
<td>user</td>
<td>library</td>
<td>library</td>
<td>Sep 03, 2023 07:57am</td>
<td>N/A</td>
</tr>
</tbody>
</table>
### Integration Events Consumed by Marquez

#### Book Table

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>VARCHAR</td>
<td>no description</td>
</tr>
<tr>
<td>author</td>
<td>VARCHAR(150)</td>
<td>no description</td>
</tr>
<tr>
<td>title</td>
<td>VARCHAR(300)</td>
<td>no description</td>
</tr>
<tr>
<td>publisher</td>
<td>VARCHAR(150)</td>
<td>no description</td>
</tr>
<tr>
<td>description</td>
<td>VARCHAR(500)</td>
<td>no description</td>
</tr>
<tr>
<td>genre</td>
<td>VARCHAR(50)</td>
<td>no description</td>
</tr>
<tr>
<td>image</td>
<td>VARCHAR(200)</td>
<td>no description</td>
</tr>
<tr>
<td>pub_date</td>
<td>VARCHAR(10)</td>
<td>no description</td>
</tr>
<tr>
<td>more_info</td>
<td>VARCHAR(200)</td>
<td>no description</td>
</tr>
<tr>
<td>user_id</td>
<td>VARCHAR</td>
<td>no description</td>
</tr>
<tr>
<td>NAME</td>
<td>TYPE</td>
<td>DESCRIPTION</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>id</td>
<td>VARCHAR</td>
<td>no description</td>
</tr>
<tr>
<td>title</td>
<td>VARCHAR(100)</td>
<td>no description</td>
</tr>
<tr>
<td>content</td>
<td>VARCHAR(300)</td>
<td>no description</td>
</tr>
<tr>
<td>date_created</td>
<td>DATETIME</td>
<td>no description</td>
</tr>
<tr>
<td>email</td>
<td>VARCHAR</td>
<td>no description</td>
</tr>
</tbody>
</table>

**FACETS**

- root: 0 keys
  - schema: 0 keys
  - dataSource: 0 keys
### Latest Schema

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>VARCHAR</td>
<td>no description</td>
</tr>
<tr>
<td>email</td>
<td>VARCHAR(150)</td>
<td>no description</td>
</tr>
<tr>
<td>password</td>
<td>VARCHAR</td>
<td>no description</td>
</tr>
</tbody>
</table>

### Facets

- **root**: 2 keys
  - **schema**: 3 keys
  - **dataSource**: 4 keys
Michael Robinson
- Community Manager, OpenLineage+Marquez Communities, Astronomer
- OpenLineage+Marquez committer
- Airflow contributor
- GitHub: https://github.com/merobi-hub

For more information about the integration, see the PR in OpenLineage:
- Project PR: https://github.com/OpenLineage/OpenLineage/pull/2088

Thank you
(After) Party Under the Stars

Wednesday, September 20th
6:30pm-10:00pm

The Sheraton Centre
123 Queen St W
(7 min walk)

RSVP Now
Let’s flow together

Workshop
Get Airflow Certified

Thursday, September 21st
12:00 pm in Trinity 4

Marc Lamberti
Head of Customer Education
at Astronomer