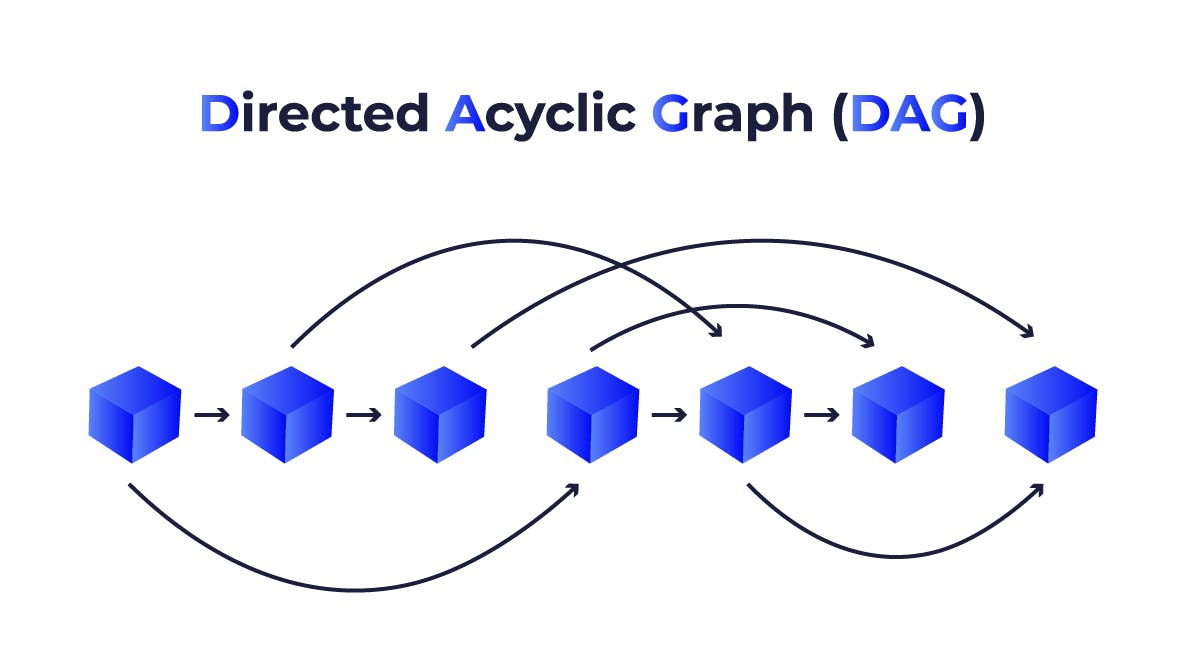
**Описание DAG (Directed Acyclic Graph) в Apache Airflow**



**DAG (Directed Acyclic Graph)** — это фундаментальная концепция в **Apache Airflow**, представляющая собой набор задач, организованных в виде направленного ациклического графа.

**Вот ключевые характеристики:**

1. **Направленный (Directed)**:
   * Задачи выполняются в определенной последовательности
   * Зависимости между задачами строго определены (например, задача B выполняется только после успешного завершения задачи A)
2. **Ациклический (Acyclic)**:
   * Не содержит циклов или петель
   * Задачи не могут ссылаться на самих себя или создавать бесконечные циклы
   * Поток выполнения всегда идет в одном направлении
3. **Граф (Graph)**:
   * Визуальное представление зависимостей между задачами
   * Узлы графа = отдельные задачи (tasks)
   * Ребра графа = зависимости между задачами

**Основные компоненты DAG в нашем ETL-процессе:**

| **Компонент** | **Описание** | **Пример из кода** |
| --- | --- | --- |
| **Задачи (Tasks)** | Отдельные единицы работы | extract\_lms\_data, transform\_data |
| **Операторы (Operators)** | Шаблоны для создания задач (PythonOperator, MySqlOperator и т.д.) | PythonOperator |
| **Зависимости** | Определяют порядок выполнения задач | extract\_task >> transform\_task |
| **Расписание** | Интервал запуска DAG (cron-выражение или предустановки) | schedule\_interval='@daily' |
| **Параметры** | Конфигурация выполнения (повторные попытки, таймауты, уведомления) | retries=2, retry\_delay |

**Особенности нашего DAG "etl\_educational\_data":**

1. **Структура выполнения:**```  
   graph LR

A[extract\_lms\_data] --> C[transform\_data]

B[extract\_electronic\_journal] --> C

C --> D[load\_to\_dwh]  
```

1. **Ключевые параметры:**

* **Расписание: Ежедневный запуск (@daily)**
* **Обработка ошибок: 2 повторные попытки с интервалом 5 минут**
* **Уведомления: Отправка email при сбоях**
* **Теги: Классификация по домену (education, ETL, BMSTU)**

1. **Механизм обмена данными**:

* Использует **XCom** для передачи информации между задачами
* Пример: ti.xcom\_push(key='lms\_status', value='success')
* Позволяет последующим задачам получать результаты предыдущих

1. **Идемпотентность**:

* Повторный запуск при ошибках не вызывает дублирования данных
* Режим REPLACE в SQL-запросе гарантирует обновление существующих записей

**Преимущества использования DAG для ETL:**

1. **Визуальный мониторинг**:
   * Интерфейс Airflow показывает статус выполнения в реальном времени
   * Возможность отслеживания длительных процессов
2. **Повторное использование**:
   * Единая конфигурация для регулярных задач
   * Параметризация для разных окружений
3. **Масштабируемость**:
   * Распределенное выполнение задач на разных воркерах
   * Поддержка горизонтального масштабирования
4. **Отказоустойчивость**:
   * Автоматические повторы при сбоях
   * Сохранение состояния выполнения
   * Возможность ручного перезапуска с места сбоя
5. **Экосистема**:
   * Интеграция с различными системами (базы данных, облачные сервисы, очереди сообщений)
   * Богатая библиотека операторов

**Данный DAG обеспечивает полную автоматизацию ETL-процесса** - от получения сырых данных из образовательных систем до их загрузки в аналитическое хранилище, соответствуя принципам надежности, наблюдаемости и поддерживаемости.

**📅 Дата:** 16.06.2025

**📖 Автор:** Дуплей Максим Игоревич

**📲 TG:** @quadd4rvn17

**❃ Хижина программиста Æ**

💡 [**@hut\_programmer\_07**](https://t.me/hut_programmer_07)