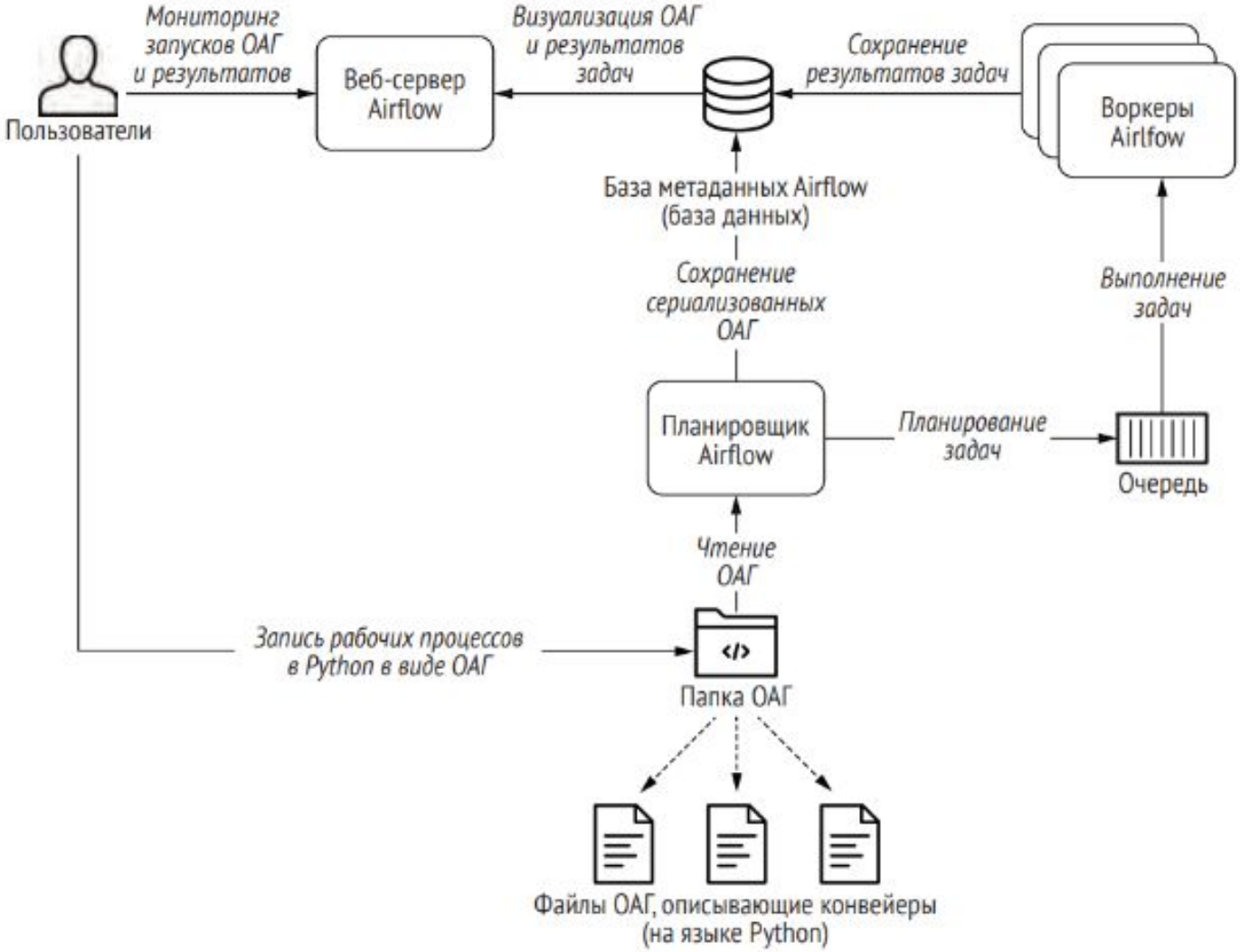
Обработка больших данных с Apache Airflow (ч. 2)

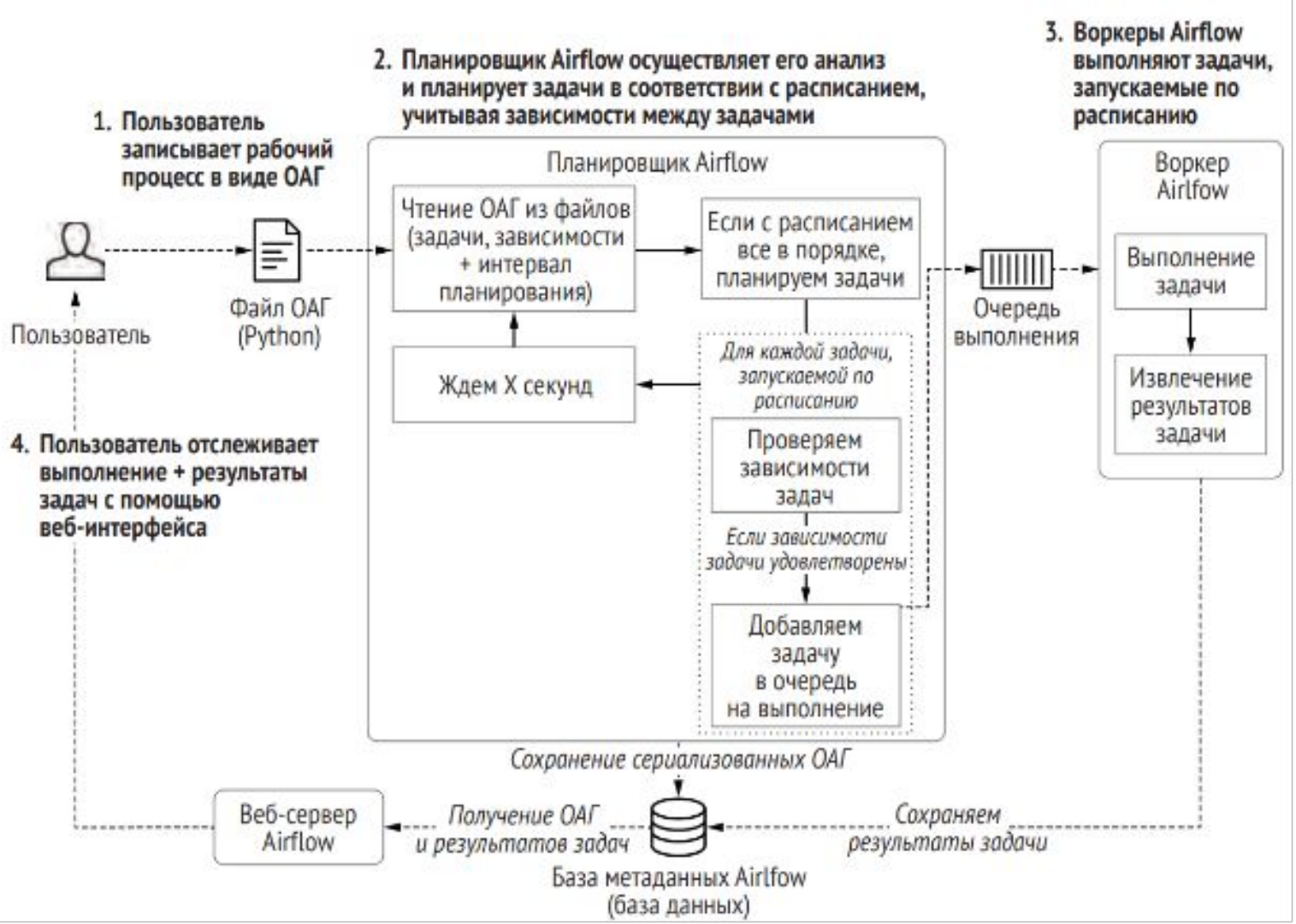
Астахов Сергей ИУ6-22М

Agenda

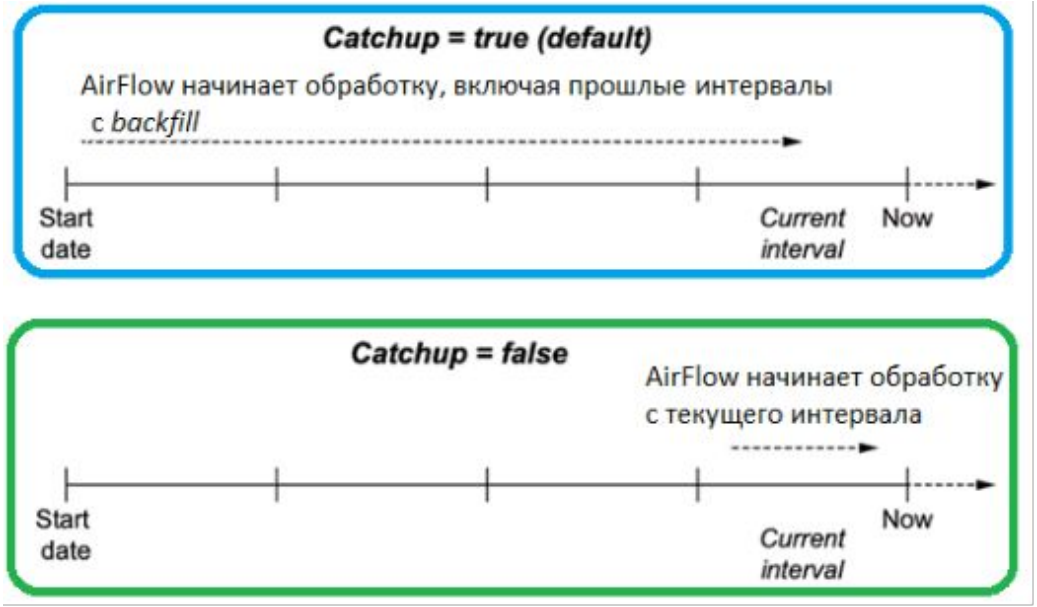
- Планирование и выполнение конвейеров
- Инкрементальная загрузка и обратное заполнение
- Атомарность и идемпотентность задач
- Ветвление

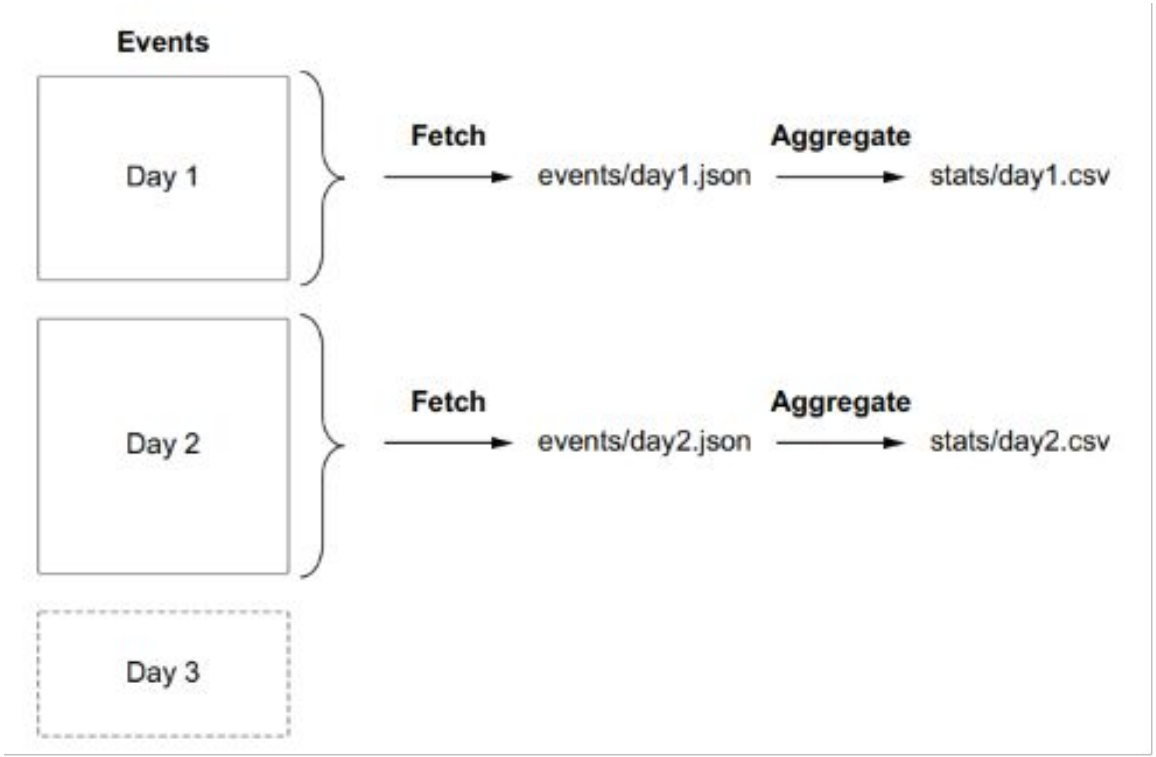
Планирование и выполнение конвейеров


Мониторинг запусков ОАГ и результатов
Пользователи
Веб-сервер Airflow
Визуализация ОАГ и результатов задач
Сохранение результатов задач
Воркеры Airlfow
База метаданных Airflow (база данных)
Сохранение сериализованных ОАГ
Выполнение задач
Планировщик Airflow
Планирование задач
Очередь
Чтение ОАГ
Запись рабочих процессов в Python в виде ОАГ
Папка ОАГ
Файлы ОАГ, описывающие конвейеры (на языке Python)


1. Пользователь записывает рабочий процесс в виде ОАГ
2. Планировщик Airflow осуществляет его анализ и планирует задачи в соответствии с расписанием, учитывая зависимости между задачами
3. Воркеры Airflow выполняют задачи, запускаемые по расписанию
4. Пользователь отслеживает выполнение + результаты задач с помощью веб-интерфейса
Пользователь
Файл ОАГ (Python)
Планировщик Airflow
Чтение ОАГ из файлов (задачи, зависимости + интервал планирования)
Если с расписанием все в порядке, планируем задачи
Ждем X секунд
Для каждой задачи, запускаемой по расписанию
Проверяем зависимости задач
Если зависимости задачи удовлетворены
Добавляем задачу в очередь на выполнение
Очередь выполнения
Воркер Airlfow
Выполнение задачи
Извлечение результатов задачи
Сохранение сериализованных ОАГ
Веб-сервер Airflow
Получение ОАГ и результатов задач
Сохраняем результаты задачи
База метаданных Airlfow (база данных)

Инкрементальная загрузка и обратное заполнение


Catchup = true (default)
AirFlow начинает обработку, включая прошлые интервалы
с backfill
Start date
Current interval
Now
Catchup = false
AirFlow начинает обработку
с текущего интервала
Start date
Current interval
Now


Events
Day 1
Fetch
events/day1.json
Aggregate
stats/day1.csv
Day 2
Fetch
events/day2.json
Aggregate
stats/day2.csv
Day 3

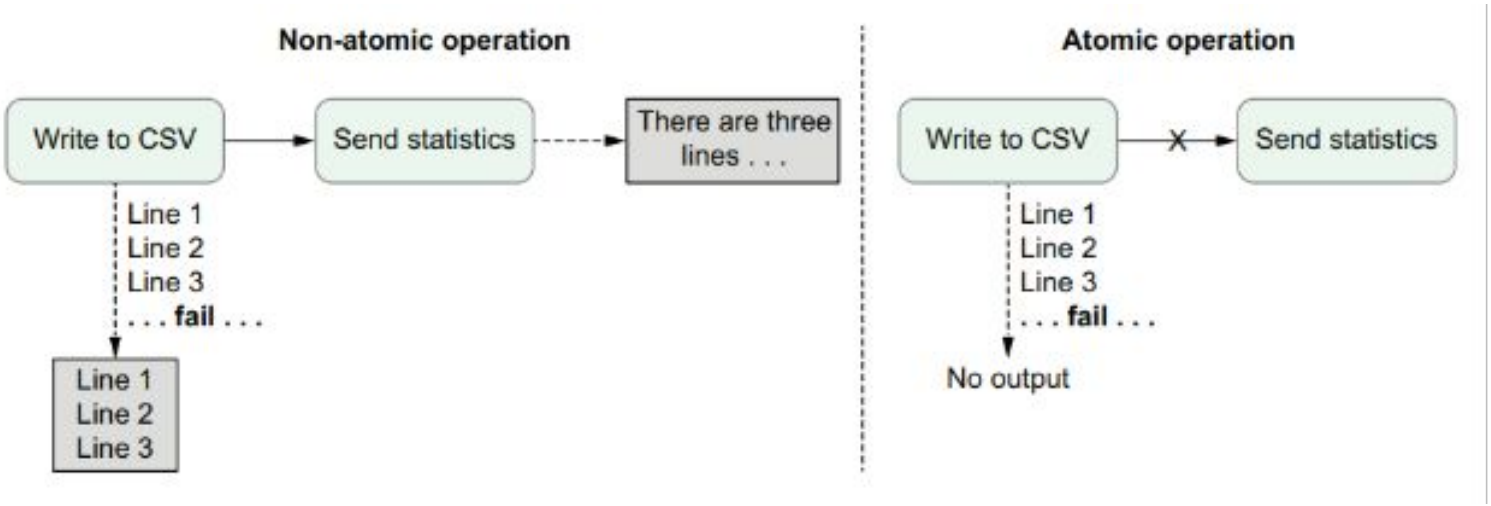
Атомарность и идемпотентность

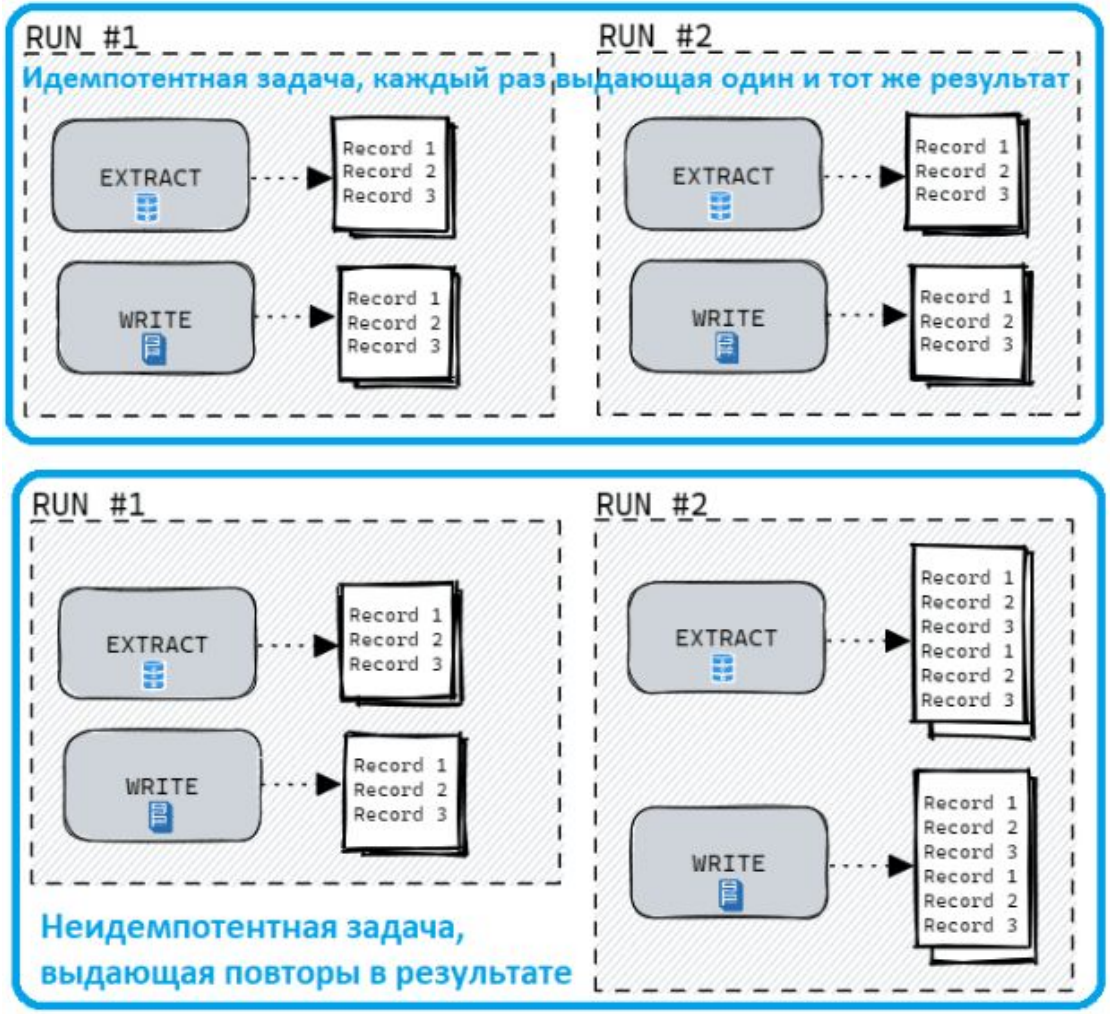
Не атомарно

```
def _calculate_stats(**context):
    """Calculates event statistics."""
    input_path = context["templates_dict"]["input_path"]
    output_path = context["templates_dict"]["output_path"]

    events = pd.read_json(input_path)
    stats = events.groupby(["date",
"user"]).size().reset_index()
    stats.to_csv(output_path, index=False)

    send_stats(stats, email="user@example.com")
```

Атомарно

```
def _send_stats(email, **context):
        stats =
        pd.read_csv(context["templates_dict"]["stats_path
        "])
        email_stats(stats, email=email)


send_stats = PythonOperator(
        task_id="send_stats",
        python_callable=_send_stats,
        op_kwargs={"email": "user@example.com"},
        templates_dict={"stats_path":
        "data/stats/{{ds}}.csv"},
        provide_context=True,
        dag=dag,
)

calculate_stats >> send_stats
```


Non-atomic operation
Write to CSV
Send statistics
There are three lines . . .
Line 1
Line 2
Line 3
. . . fail . . .
Line 1
Line 2
Line 3
Atomic operation
Write to CSV
Send statistics
Line 1
Line 2
Line 3
. . . fail . . .
No output

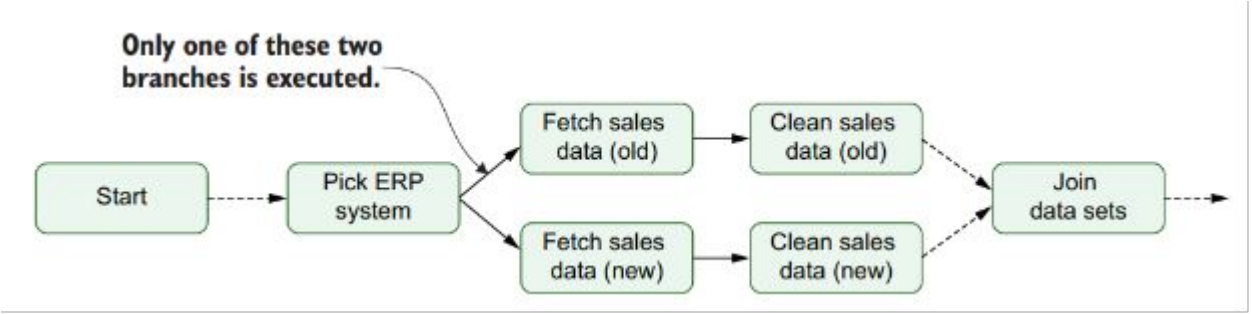

RUN #1
Идемпотентная задача, каждый раз выдающая один и тот же результат
EXTRACT
Record 1
Record 2
Record 3
WRITE
Record 1
Record 2
Record 3
RUN #2
EXTRACT
Record 1
Record 2
Record 3
WRITE
Record 1
Record 2
Record 3
RUN #1
EXTRACT
Record 1
Record 2
Record 3
WRITE
Record 1
Record 2
Record 3
Неидемпотентная задача,
выдающая повторы в результате
RUN #2
EXTRACT
Record 1
Record 2
Record 3
Record 1
Record 2
Record 3
WRITE
Record 1
Record 2
Record 3
Record 1
Record 2
Record 3

IDEMPOTENCE

WHEN PERFORMING AN OPERATION AGAIN GIVES THE SAME RESULT

HTTP METHOD	IDEMPOTENCE	SAFETY
GET	YES	YES
HEAD	YES	YES
PUT	YES	NO
DELETE	YES	NO
POST	NO	NO
PATCH	NO	NO

Ветвление




```
def _pick_erp_system(**context):
        if context["execution_date"] < ERP_SWITCH_DATE:
        return "fetch_sales_old"
        else:
        return "fetch_sales_new"

sales_branch = BranchPythonOperator(
        task_id='sales_branch',
        provide_context=True,
        python_callable=_pick_erp_system,
)

sales_branch >> [fetch_sales_old, fetch_sales_new]
```

- all_success
- all_failed
- all_done
- all_skipped
- one_failed
- one_success
- one_done
- none_failed
- none_failed_min_one_success
- none_skipped
- always

- BranchPythonOperator
- ShortCircuitOperator
- BranchSQLOperator
- BranchDayOfWeekOperator
- BranchDateTimeOperator

Спасибо
за внимание

Спасибо
за внимание