

Modern Storages and Data Warehousing Week 6 - ETL

Попов Илья, i.popov@hse.ru

1 - Домашнее задание

2 - Задача с собеса

Задача с собеседа

- › У нас есть Clickhouse
- › Особенности Clickhouse как СУБД - он очень не любит одиночные insert / update / delete, но при этом очень быстро делает group by
- › У нас есть потребность считать агрегаты с большого объема данных - скажем, с терабайта. Clickhouse такие операции очень любит.
- › Но данные активно меняются / дообогачаются атрибутами, что создает потребность делать много update-операций;
- › Вопрос: можем ли мы построить ETL так, чтобы не использовать update / delete? Если да - то как?

Ресар прошлых занятий

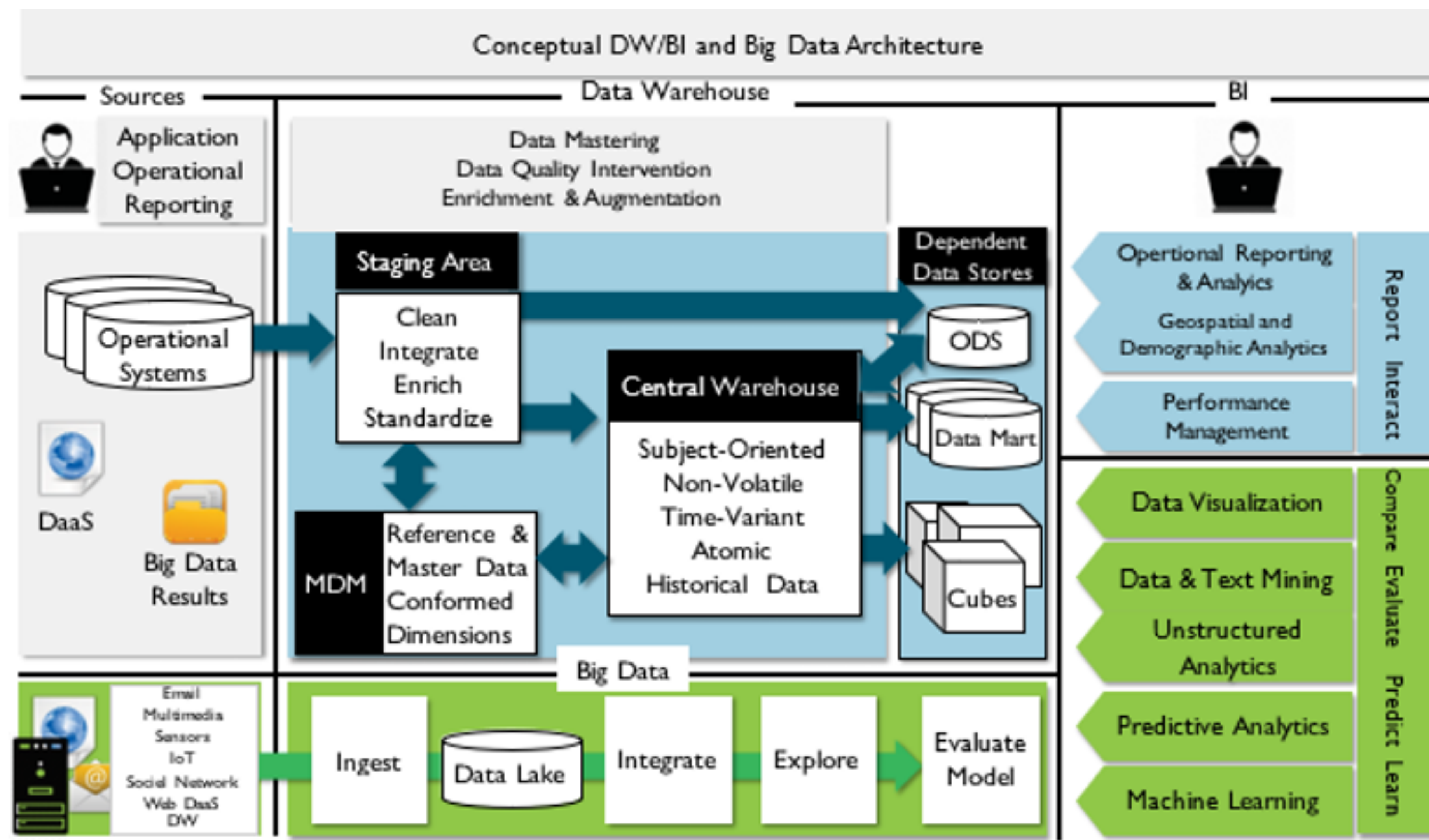


Figure 5: Date Warehouse Concept

3 - ETL

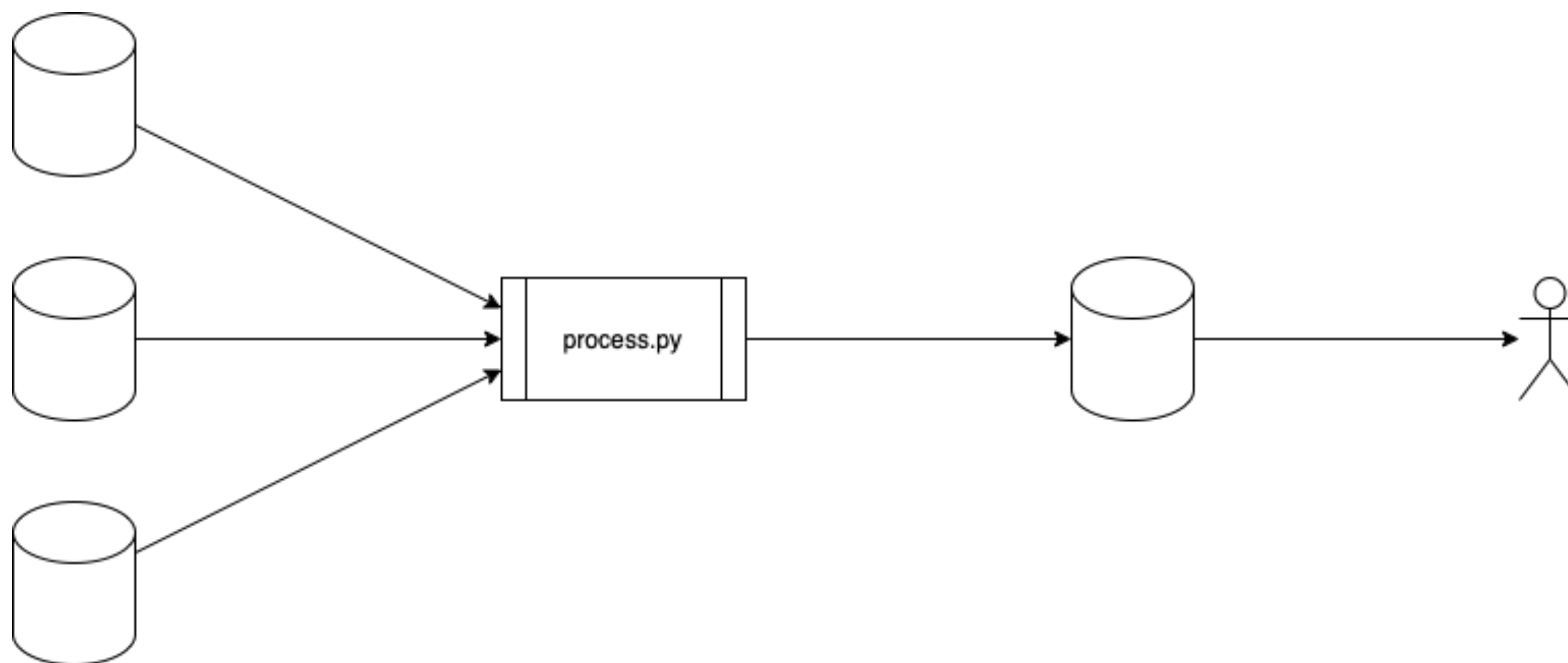
Что такое ETL

ETL (Extract, Transform, Load) - это класс процессов управления данными. Он состоит из:

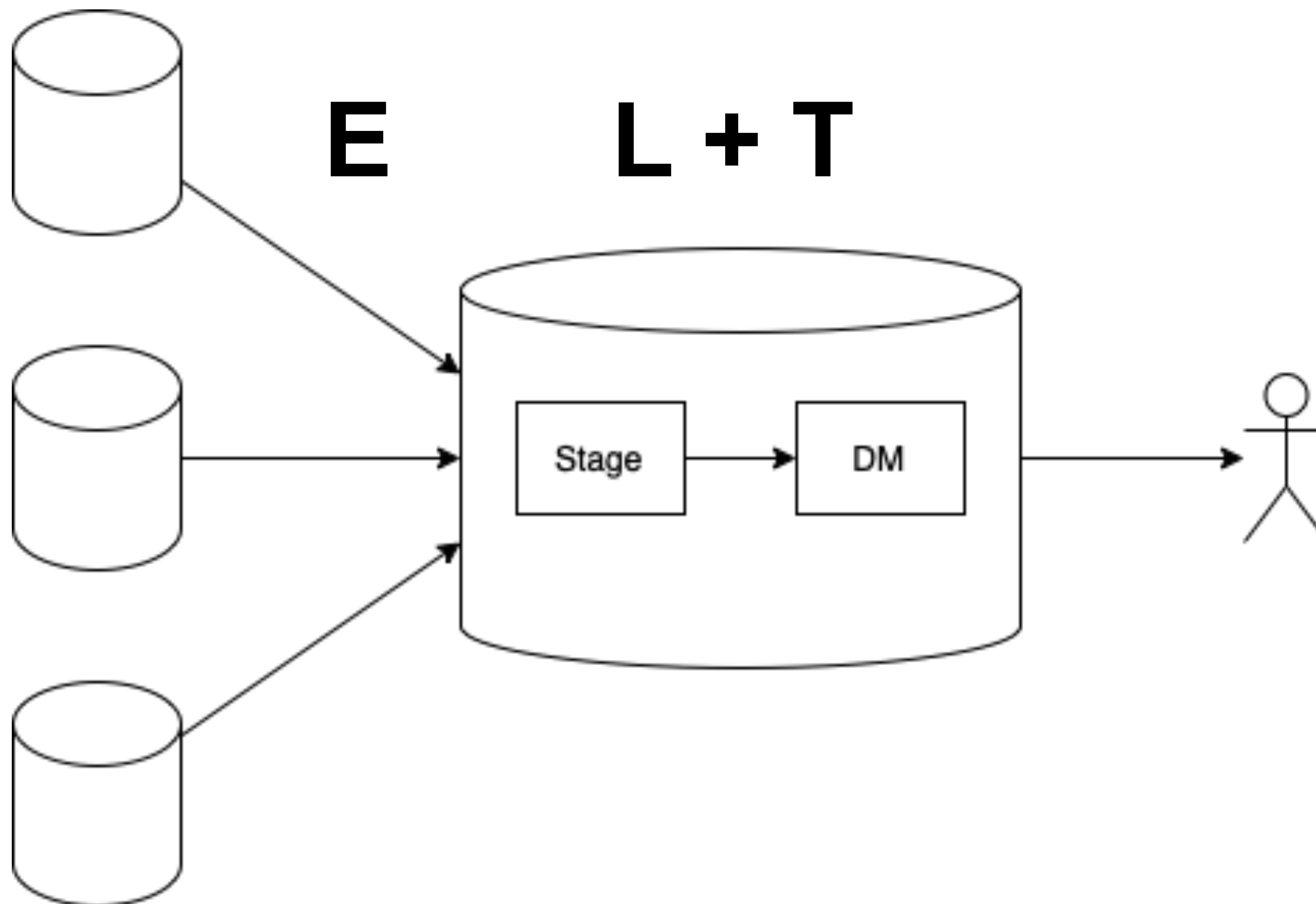
- › **Извлечение данных (Extraction - E)** - из одного или нескольких источников и подготовка их к преобразованию (загрузка в промежуточную область, проверка данных на соответствие спецификациям и возможность последующей загрузки в ХД);
- › **Трансформация данных (Transform - T)** - преобразование форматов и кодировки, агрегация и очистка;
- › **Загрузка данных (Load - L)** - запись преобразованных данных, включая информацию о структуре их представления (метаданные) в необходимую систему хранения или витрину данных.

Какие бывают ETL

E T L



Какие бывают ETL



Свойства ETL-процесса

Мы с вами будем говорить про 3 свойства разных видов ETL:

- › Идемпотентность
- › Delete Policy
- › Change Tracking

Идемпотентность

Идемпотентность - гарантия того, что при повторном применении трансформации к данным результат будет таким же, что и при первом.

Это свойство может гарантироваться только в случае, когда на вход подаются одинаковые данные.

Возможные значения:

- › True
- › False

Delete Policy

Delete Policy определяет способ обработки факта удаления данных на источнике при трансформации.

Возможные значения:

- › **Never** - данные из целевой таблицы не могут быть удалены в штатном режиме работы даже при их удалении с источника
- › **Delete** - удаление данных на источнике приводит к удалению данных в целевой таблице
- › **Delete on backfill** - удаление данных может произойти только при историческом пересчете

Change Tracking

Change Tracking определяет, выполняется ли построение истории изменения объекта по его идентификатору или нет. Существуют различные способы сохранения этой истории.

Возможные значения:

- › No tracking
- › Overwrite
- › History
- › Accumulate

Snapshot

Полная перезапись таблицы из источников.

Snapshot

Данные в целевой таблице

id=1
id=2
id=3
id=4

Новые данные

id=1
id=3
id=4
id=5

Результат

id=1
id=3
id=4
id=5

Snapshot

Требования к целевой таблице:

- › Никаких

Когда можно:

- › Маленькие таблицы

- › Нужно хранить только текущее состояние

Когда нельзя:

- › Большие таблицы (1M+ rows)

Snapshot

Требования к целевой таблице:

- › Никаких

Когда можно:

- › Маленькие таблицы
- › Нужно хранить только текущее состояние

Когда нельзя:

- › Большие таблицы (1M+ rows)

Вопросы:

- › Идемпотентность?
- › Delete Policy?
- › Change Tracking?

Snapshot

Требования к целевой таблице:

- › Никаких

Когда можно:

- › Маленькие таблицы
- › Нужно хранить только текущее состояние

Когда нельзя:

- › Большие таблицы (1M+ rows)

Вопросы:

- › Идемпотентность
True
- › Delete Policy
Delete
- › Change Tracking
Overwrite

Periodic Snapshot

Загружаем обновления за последний период с некоторой
гранулярностью: час, день, неделя, месяц, квартал, год и т.д.

Periodic Snapshot за 2020-01-02								
Данные в целевой таблице			Новые данные			Результат		
dt=2020-01-01	id=1	v=10	dt=2020-01-02	id=1	v=15	dt=2020-01-01	id=1	v=10
dt=2020-01-01	id=2	v=20	dt=2020-01-02	id=3	v=30	dt=2020-01-01	id=2	v=20
dt=2020-01-01	id=3	v=30	dt=2020-01-02	id=4	v=40	dt=2020-01-01	id=3	v=30
						dt=2020-01-02	id=1	v=15
						dt=2020-01-02	id=3	v=30
						dt=2020-01-02	id=4	v=40

Periodic Snapshot

Требования к целевой таблице:

- › Должно быть поле с датой загрузки
- › Может быть партицирована

Когда можно:

- › Мало данных
- › Надо знать состояние сущности с какой-то гранулярностью

Когда нельзя:

- › Большие таблицы, т.к. избыточность

Periodic Snapshot

Требования к целевой таблице:

- › Должно быть поле с датой загрузки
- › Может быть партицирована

Когда можно:

- › Мало данных
- › Надо знать состояние сущности с какой-то гранулярностью

Когда нельзя:

- › Большие таблицы, т.к. избыточность

Вопросы:

- › Идемпотентность?
- › Delete Policy?
- › Change Tracking?

Periodic Snapshot

Требования к целевой таблице:

- › Должно быть поле с датой загрузки
- › Может быть партицирована

Когда можно:

- › Мало данных
- › Надо знать состояние сущности с какой-то гранулярностью

Когда нельзя:

- › Большие таблицы, т.к. избыточность

Вопросы:

- › Идемпотентность
True
- › Delete Policy
Delete on backfill
- › Change Tracking
Accumulate

Append

Старые данные не меняются. Новые данные дозаписываются.

Append

Данные в целевой таблице

dt=2020-01-01	id=1	v=10
dt=2020-01-01	id=2	v=20
dt=2020-01-01	id=3	v=30

Новые данные

dt=2020-01-02	id=1	v=15
dt=2020-01-02	id=3	v=30
dt=2020-01-02	id=4	v=40

Результат

dt=2020-01-01	id=1	v=10
dt=2020-01-01	id=2	v=20
dt=2020-01-01	id=3	v=30
dt=2020-01-02	id=1	v=15
dt=2020-01-02	id=3	v=30
dt=2020-01-02	id=4	v=40

Append

Требования к целевой таблице:

- › Никаких

Когда можно:

- › Данные не меняются после записи
- › Для каких-то целей нужен лог изменений

Когда нельзя:

- › При большом кол-ве изменений таблица будет быстро расти
- › Может потребоваться TTL

Вопросы:

- › Идемпотентность?
- › Delete Policy?
- › Change Tracking?

Append

Требования к целевой таблице:

- › Никаких

Когда можно:

- › Данные не меняются после записи
- › Для каких-то целей нужен лог изменений

Когда нельзя:

- › При большом кол-ве изменений таблица будет быстро расти
- › Может потребоваться TTL

Вопросы:

- › Идемпотентность
False
- › Delete Policy
Never
- › Change Tracking
Accumulate

Insert by key

Если появляются новые данные - они добавляются в таблицу. Старые данные не меняются. При удалении из источника данные не удаляются.

Insert

Данные в целевой таблице

id=1	value=10
id=2	value=20
id=3	value=30

Новые данные

id=1	value=10
id=2	value=25
id=4	value=40

Результат

id=1	value=10
id=2	value=20
id=3	value=30
id=4	value=40

Insert by key

Требования к целевой таблице:

- › Нужен уникальный ключ

Когда можно:

- › Данные могут меняться за произвольную глубину

- › Нужно хранить только первое состояние

Когда нельзя:

- › Нет уникального ключа

Insert by key

Требования к целевой таблице:

- › Нужен уникальный ключ

Когда можно:

- › Данные могут меняться за произвольную глубину
- › Нужно хранить только первое состояние

Когда нельзя:

- › Нет уникального ключа

Вопросы:

- › Идемпотентность?
- › Delete Policy?
- › Change Tracking?

Insert by key

Требования к целевой таблице:

- › Нужен уникальный ключ

Когда можно:

- › Данные могут меняться за произвольную глубину
- › Нужно хранить только первое состояние

Когда нельзя:

- › Нет уникального ключа или его тяжело брать

Вопросы:

- › Идемпотентность
True
- › Delete Policy
Never
- › Change Tracking
No tracking

Upsert

Если появятся новые данные - они добавляются в таблицу. Старые данные таблицы обновляются из новых по ключу. При удалении из источника данные не удаляются.

Upsert

Данные в целевой таблице

id=1	value=10
id=2	value=20
id=3	value=30

Новые данные

id=1	value=10
id=2	value=25
id=4	value=40

Результат

id=1	value=10
id=2	value=25
id=3	value=30
id=4	value=40

Upsert

Требования к целевой таблице:

- › Нужен уникальный ключ

Когда можно:

- › Данные могут меняться за произвольную глубину

- › Нужно хранить текущее состояние

Когда нельзя:

- › Нет уникального ключа

Upsert

Требования к целевой таблице:

- › Нужен уникальный ключ

Когда можно:

- › Данные могут меняться за произвольную глубину
- › Нужно хранить текущее состояние

Когда нельзя:

- › Нет уникального ключа

Вопросы:

- › Идемпотентность?
- › Delete Policy?
- › Change Tracking?

Upsert

Требования к целевой таблице:

- › Нужен уникальный ключ

Когда можно:

- › Данные могут меняться за произвольную глубину
- › Нужно хранить текущее состояние

Когда нельзя:

- › Нет уникального ключа

Вопросы:

- › Идемпотентность
True
- › Delete Policy
Never
- › Change Tracking
Overwrite

History по SCD2

History SCD2

Данные в целевой таблице

id=1	name=a	v=10	from=2019-01-01	to=2019-12-31
id=1	name=a	v=10	from=2020-01-01	to=9999-12-31

Новые данные за 2020-02-01

id=1	name=a	v=20
id=2	name=b	v=15

Результат

id=1	name=a	v=10	from=2019-01-01	to=2019-12-31
id=1	name=a	v=10	from=2020-01-01	to=2020-01-31
id=1	name=a	v=20	from=2020-02-01	to=9999-12-31
id=2	name=b	v=15	from=2020-02-01	to=9999-12-31

Берется срез изменений за определенную дату.
На основе этого среза изменяются или создаются сроки в исходной таблице.

History по SCD2

Требования к целевой таблице:

- › Нужен уникальный ключ

Когда можно:

- › Нужно отслеживать историю изменения атрибутов

- › Чаще всего применяется для справочников

Когда нельзя:

- › При большом количестве изменений таблица быстро растёт в размере

History по SCD2

Требования к целевой таблице:

- › Нужен уникальный ключ

Когда можно:

- › Нужно отслеживать историю изменения атрибутов
- › Чаще всего применяется для справочников

Когда нельзя:

- › При большом количестве изменений таблица быстро растёт в размере

Вопросы:

- › Идемпотентность?
- › Delete Policy?
- › Change Tracking?

History по SCD2

Требования к целевой таблице:

- › Нужен уникальный ключ

Когда можно:

- › Нужно отслеживать историю изменения атрибутов
- › Чаще всего применяется для справочников

Когда нельзя:

- › При большом количестве изменений таблица быстро растёт в размере

Вопросы:

- › Идемпотентность
True
- › Delete Policy
Never
- › Change Tracking
History

Особенности времени

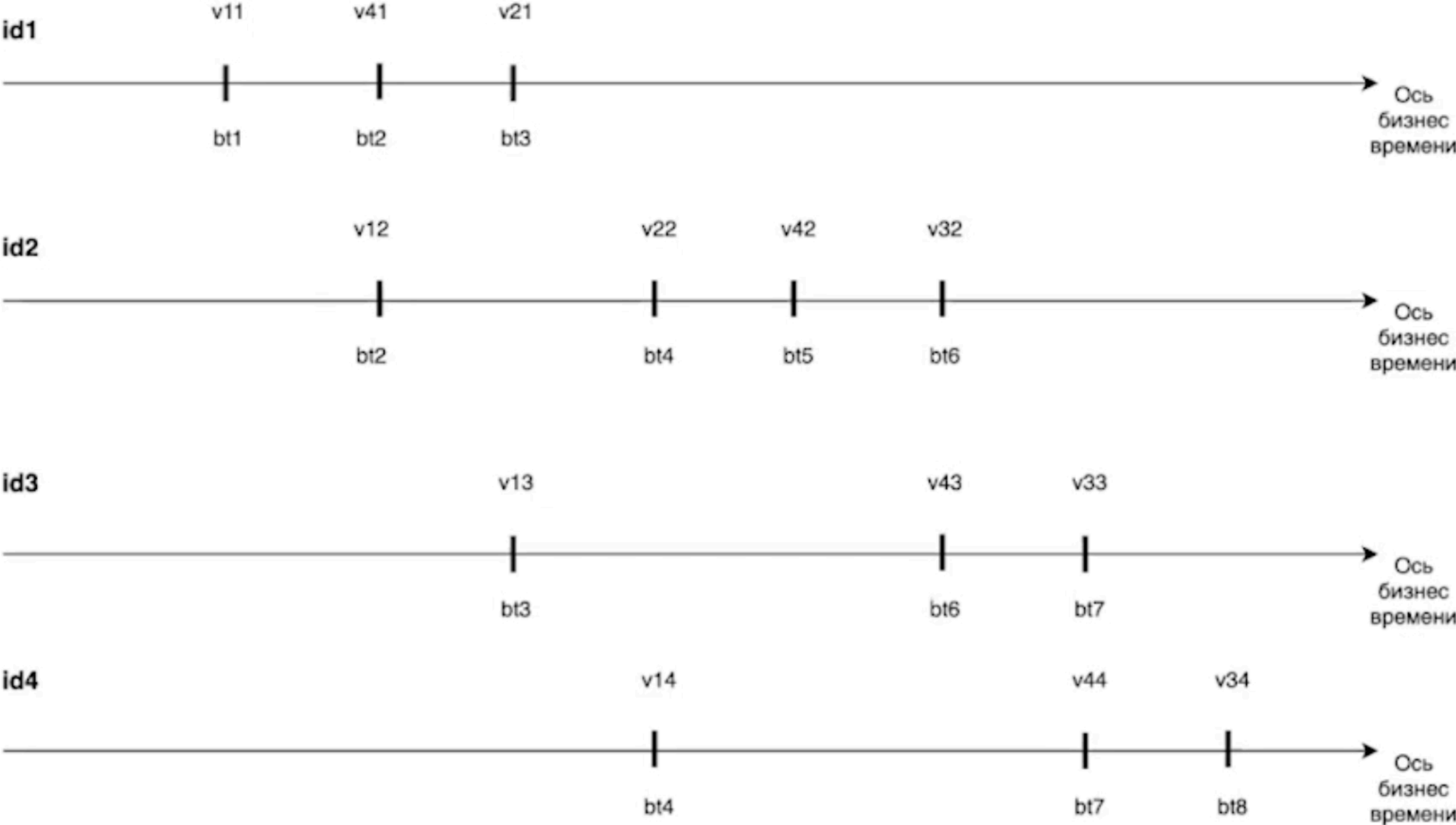
У нас есть 2 вида времени:

- › **Бизнес-время** - время актуальности данных с точки зрения бизнеса; при получении данных с бизнес-временем мы можем говорить, что они актуальны на указанное бизнес-время;
- › **Техническое время** - реальное время на сервере при получении партии данных;

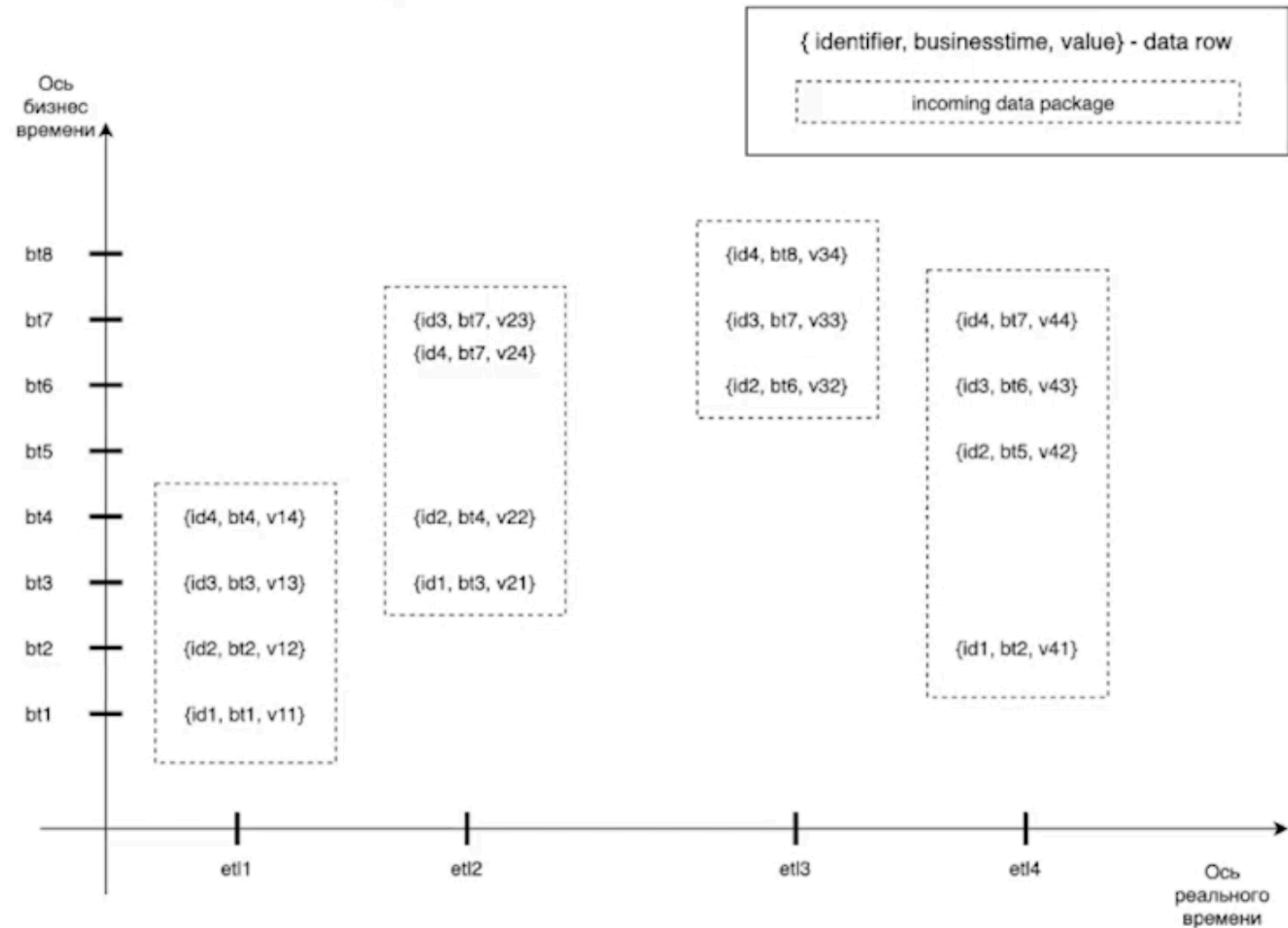
Свойства времен:

- › Техническое время монотонно возрастает - мы не можем получать данные в прошлом
- › Бизнес-время может приходить за любой период - прошлое, настоящее или будущее
- › Для одной бизнес-сущности не может быть двух значений данных в одно бизнес-время

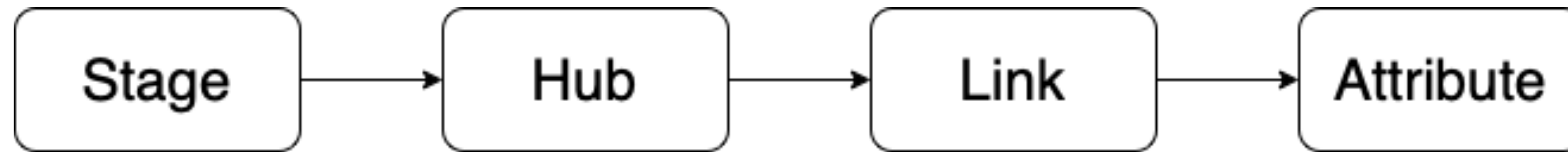
Бизнес-время



Техническое время

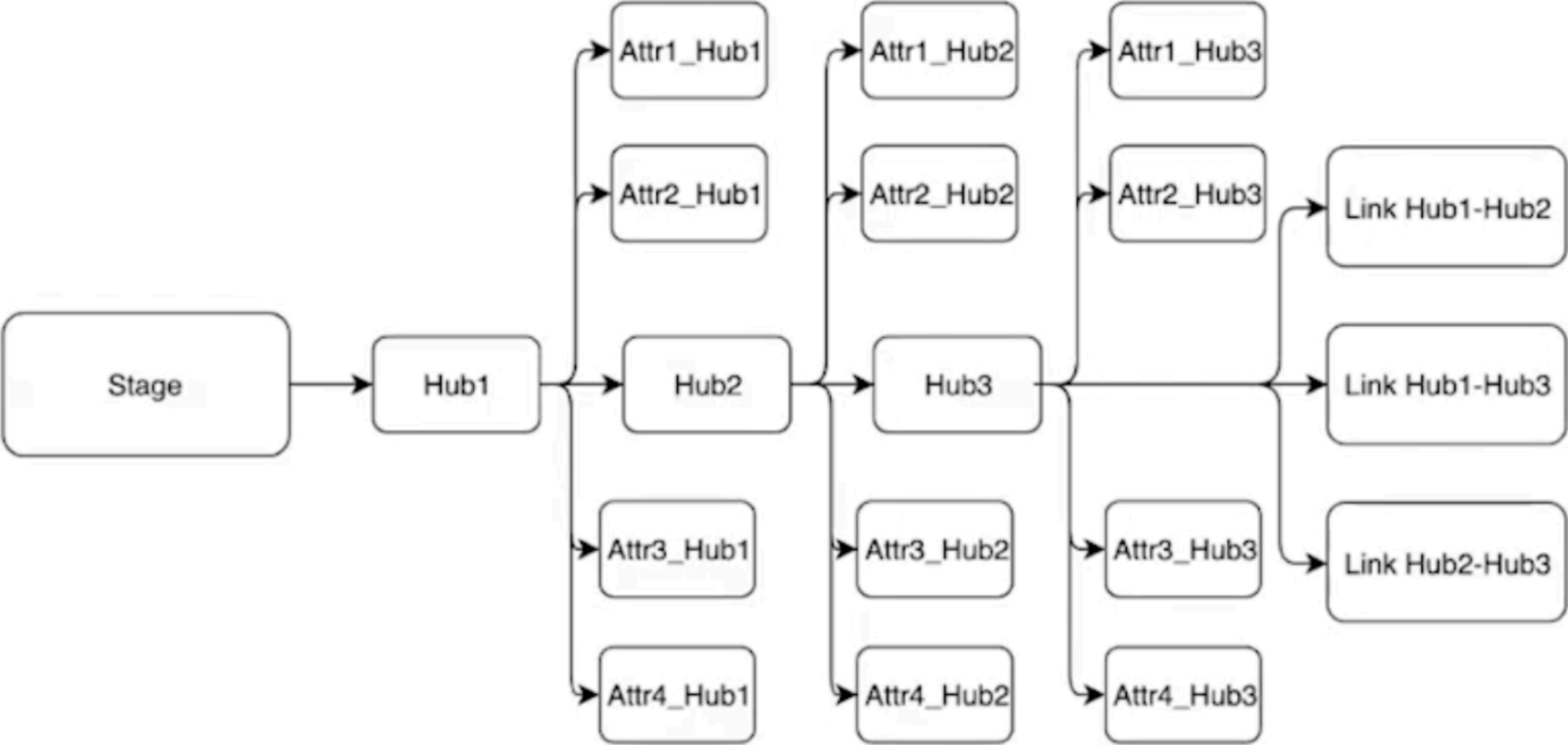


Как работает загрузка в DWH



- › Грузим данные в STG
- › Генерируем суррогатные ключи для Hub
- › Грузим Hub
- › Грузим Link
- › Грузим Attribute

Как работает загрузка в DWH



4 - Оркестрация

Мотивация

- › Мы написали процесс, который везет изменения из источников к нам в целевую витрину
- › Хотим сделать так, чтобы мы его не запускали руками, а он включался автоматически по какому-нибудь условию или расписанию

Crontab

- › Системный daemon Linux
- › Работает из коробки
- › Может выполнять произвольную bash-команду по заданному расписанию

Crontab

“At 22:00 on every day-of-week from Monday through Friday.”

next at 2023-10-20 22:00:00 random

0

22

*

*

1-5

minute

hour

day
(month)

month

day
(week)

*	any value
'	value list separator
-	range of values
/	step values
@yearly	(non-standard)
@annually	(non-standard)
@monthly	(non-standard)
@weekly	(non-standard)
@daily	(non-standard)
@hourly	(non-standard)
@reboot	(non-standard)

Crontab

Как с ним работать:

- › Команда `crontab` -е открывает файл с командами для cron (default `/etc/crontab`)
- › Каждая команда на новой строке, команда имеет вид:
`* * * * * /path/to/exec`

Crontab

Плюсы:

› Простой и безотказный, как автомат Калашникова

Минусы:

› Как и у автомата Калашникова, у него только одно простое применение, и больше он ничего не умеет

Что мы можем еще сделать?

- › Запустить что-то в CI (GitHub CI, GitLab CI, Jenkins)
- › Использовать проприетарный GUI-based ETL
 - › MS SQL SERVER INTEGRATION SYSTEM
 - › ORACLE DATA INTEGRATOR
 - › INFORMatica POWER CENTER
 - › SAS DATA INTEGRATION STUDIO
- › Написать что-то свое с нуля (обычно - обертка над Crontab, но есть Яндекс с его Nirvana / Reactor)

Вселенная opensource code-based ETL фреймворков



Apache Airflow

- › Написали в 2014 внутри Airbnb
- › 2016 - выпустили в OpenSource в рамках Apache Incubator
- › 2019 - top-level проект Apache
- › 2020 - Apache Airflow 2.0



Почему именно он?

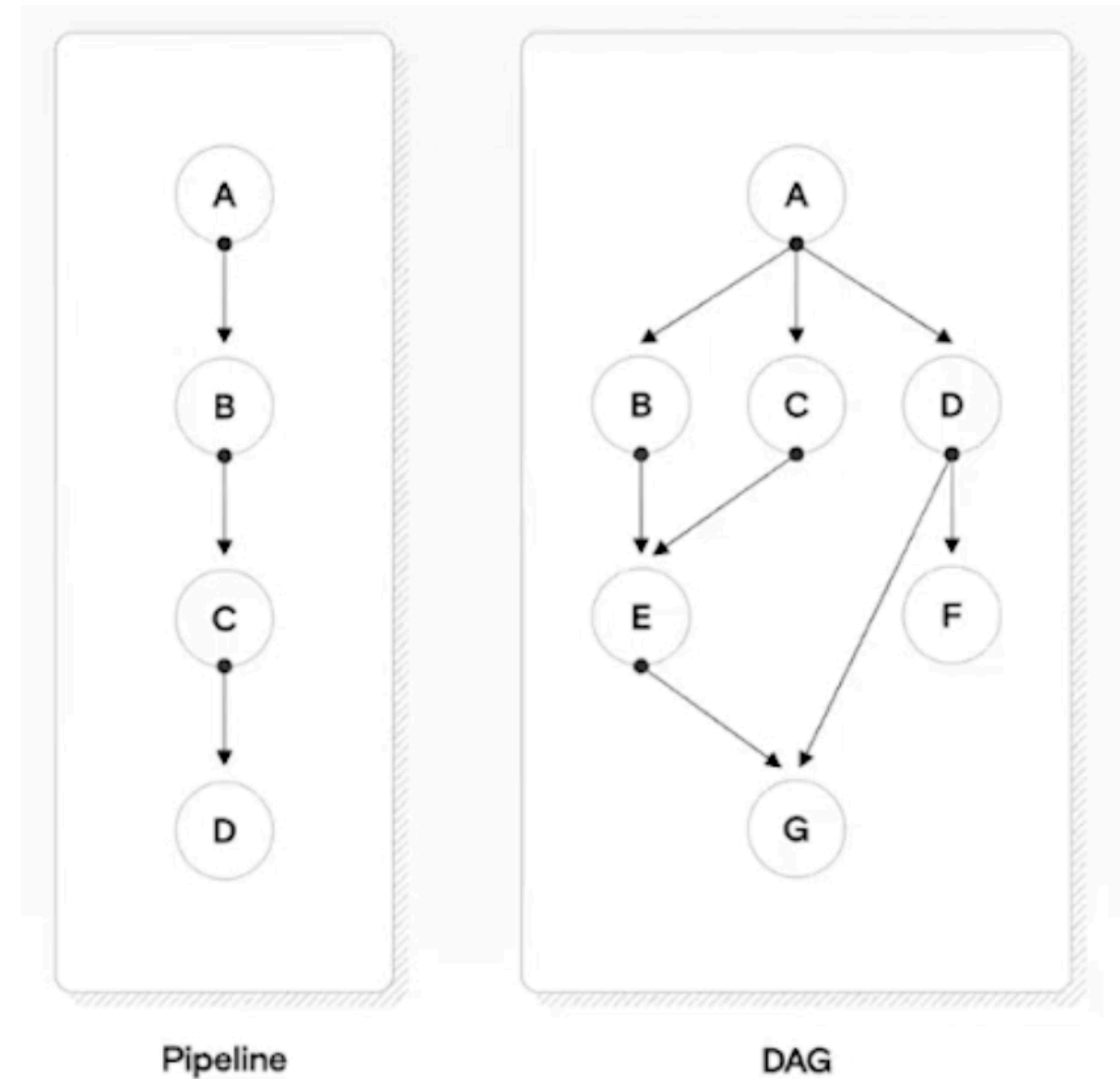
- › Open source
- › Великолепная документация (!)
- › Простой код на Python
- › Отличный Web UI
- › Встроенные алерты и мониторинги
- › Легко масштабируется
- › Легко кастомизируется и дорабатывается
- › Огромное комьюнити



DAG

DAG (Directed Acyclic Graph) -
направленный ациклический
граф.

DAG объединяет отдельные
задачи в единый Pipeline, в
котором явно прослеживаются
зависимости узлов друг от друга.



Виды операторов Airflow

Сенсор - внутри имеет произвольную функцию. Она выполняется в течение указанного времени, при падении/логическом “нет” запускается повторно через какое-то время. Как только эта функция возвращает логическое “да” - продолжает выполнение графа.

Пример: `SqlSensor`, `PythonSensor` и т.д.

Оператор - выполняет произвольное действие. Пример: `SqlOperator`, `PythonOperator` и т.д.