

# Занятие 2

## Нормализация данных, методологии проектирования, планировщики, оркестрация

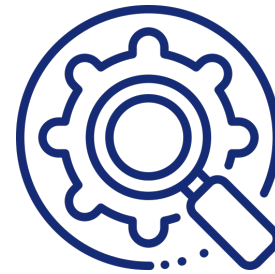
Бояр Владислав

# Занятие состоит из:



## Теория:

- Нормализация данных
- Медленно меняющиеся измерения (SCD)
- Методологии проектирования DWH
- Инструменты проектирования
- Cron, Airflow



## Практика:

- Запуск python скрипта в Cron
- Знакомство с AirFlow.

# Нормализация данных

# Нормализация данных

Нормализация данных – это способ организации данных в БД



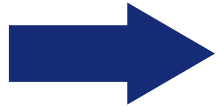
Зачем нормализовать данные в  
БД?



# Зачем нормализовать данные в БД?



Уменьшение объема БД и экономия дискового пространства;




Упрощение поиска необходимых данных (сущности лежат в отдельных таблицах);



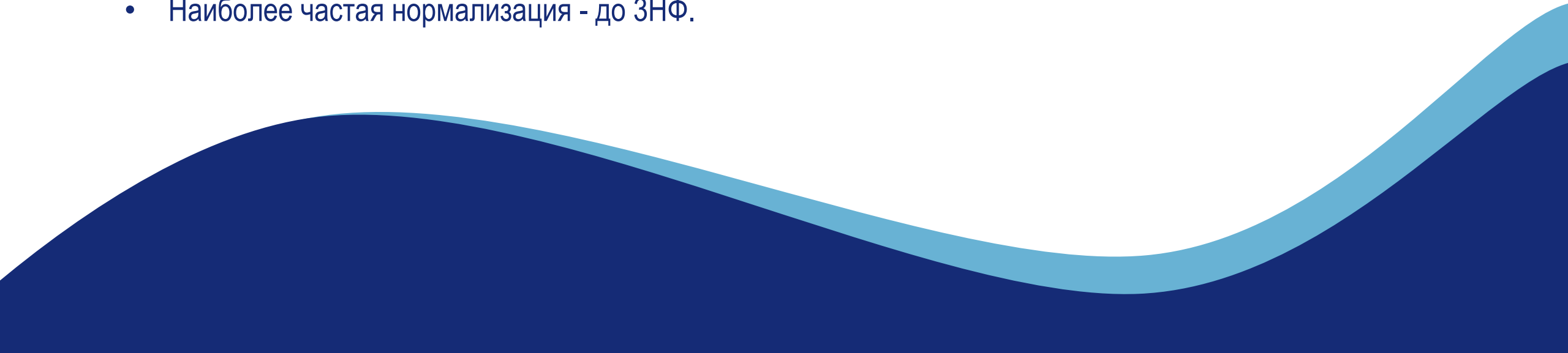
Снижение кол-ва ошибок и аномалий в данных (отсутствие смысловых дублей, обновление связанных данных);

# Нормальные формы (НФ)

**Нормальная форма** - требование, предъявляемое к структуре таблиц в теории реляционных баз данных для устранения избыточных функциональных зависимостей между атрибутами (полями):

- Первая;
  - Вторая;
  - Третья;
  - Нормальная форма Бойса-Кодда (усиленная третья);
  - Четвертая;
  - Пятая;
  - Шестая;
- 

# Особенности нормализации данных

- Как правило, процесс нормализации должен происходить на этапе проектирования БД на основе данных и бизнес-процессов;
  - Проводить нормализацию необходимо последовательно от первой до шестой НФ;
  - В БД с 3НФ (например) данные по умолчанию должны быть нормализованы по всем предыдущим НФ (1 и 2 НФ);
  - Наиболее частая нормализация - до 3НФ.
- 
- The bottom of the slide features a decorative graphic consisting of two overlapping wavy lines. The lower line is a dark blue, and the upper line is a lighter blue, creating a layered, wave-like effect that spans the width of the slide.



Нормализация данных.  
Первая нормальная форма.

# Первая нормальная форма

**В первой нормальной форме:**

- отношение не должно содержать полных дублей;
- значения в ячейках должны быть атомарными:
  - одна ячейка – одно значение
  - для ячеек не должны использоваться типы данных: array, json

# Как можно привести данные к 1НФ?

123 id ▼	ABC name ▼
1	Иванов Илья Александрович
2	Петрова Дарья Олеговна
3	Сидоров Олег Анатольевич

# Как можно привести данные к 1НФ?

123 id ▼	ABC name ▼
1	Иванов Илья Александрович
2	Петрова Дарья Олеговна
3	Сидоров Олег Анатольевич



123 id ▼	ABC first_name ▼	ABC last_name ▼	ABC surname ▼
1	Илья	Иванов	Александрович
2	Дарья	Петрова	Олеговна
3	Олег	Сидоров	Анатольевич

# Как можно привести данные к 1НФ?

123 id ▼	123 id_persons ▼	jobs ▼
1	1	► {Официант, Курьер}
2	2	► {Водитель, Плотник}

# Как можно привести данные к 1НФ?

123 id	123 id_persons	jobs
1	1	► {Официант,Курьер}
2	2	► {Водитель,Плотник}



123 id	123 id_persons	ABC job
1	1	Официант
2	1	Курьер
3	2	Водитель
4	2	Плотник

Нормализация данных.  
Вторая нормальная форма.

# Вторая нормальная форма

## Вторая нормальная форма:

- Только для отношений с составными первичными ключами!
- Отношение находится в 1НФ;
- Каждый неключевой атрибут неприводимо зависит от первичного ключа (неприводимость означает, что в составе первичного ключа отсутствует меньшее подмножество атрибутов, от которых зависит неключевой атрибут);



# Как можно привести данные к 2НФ?

<u>Филиал компании</u>	<u>Должность</u>	Зарплата	Наличие компьютера
Филиал в Томске	Уборщик	20000	Нет
Филиал в Москве	Программист	40000	Есть
Филиал в Томске	Программист	25000	Есть

# Как можно привести данные к 2НФ?

<u>Филиал компании</u>	<u>Должность</u>	Зарплата	Наличие компьютера
Филиал в Томске	Уборщик	20000	Нет
Филиал в Москве	Программист	40000	Есть
Филиал в Томске	Программист	25000	Есть

Декомпозируем

<u>Филиал компании</u>	<u>Должность</u>	Зарплата
Филиал в Томске	Уборщик	20000
Филиал в Томске	Программист	25000
Филиал в Москве	Программист	40000

<u>Должность</u>	Наличие компьютера
Уборщик	Нет
Программист	Есть

# Третья нормальная форма

## Третья нормальная форма:

- Отношение находится во 2НФ;
- Все неключевые столбцы должны зависеть только от первичного ключа (другими словами все атрибуты содержимое которых может относиться к нескольким записям, следует выносить в отдельные таблицы).

# Как можно привести данные к 3НФ?

ABC trans_num ▼	ABC last ▼	ABC job ▼
f32d1f4b2a918f4c2f6acdc83033ee35	Cox	Civil engineer, contracting
f5dad8e2d7c39d81502d846a20286659	Martinez	Aeronautical engineer
1d023bc78ab93ab65a35bbb53bcc67bd	Richards	Systems developer

# Как можно привести данные к 3НФ?

ABC trans_num ▼	ABC last ▼	ABC job ▼
f32d1f4b2a918f4c2f6acdc83033ee35	Cox	Civil engineer, contracting
f5dad8e2d7c39d81502d846a20286659	Martinez	Aeronautical engineer
1d023bc78ab93ab65a35bbb53bcc67bd	Richards	Systems developer

Декомпозируем

ABC trans_num ▼	ABC last ▼
f32d1f4b2a918f4c2f6acdc83033ee35	Cox
f5dad8e2d7c39d81502d846a20286659	Martinez
1d023bc78ab93ab65a35bbb53bcc67bd	Richards

ABC last ▼	ABC job ▼
Cox	Civil engineer, contracting
Martinez	Aeronautical engineer
Richards	Systems developer

# Недостатки нормализации



Высокие уровни нормализации (4-6 НФ) предполагают создание огромного количества таблиц и связей, что затрудняет выборку данных из хранилища (требуется большое количество операций объединения) и снижает производительность СУБД;



Для проведения нормализации необходимо много времени и ресурсов;


Медленно меняющиеся  
измерения (SCD)

# Медленно меняющиеся измерения (SCD)

## Медленно меняющиеся измерения (Slowly Changing Dimensions, SCD)

- способ разграничения данных по частоте их изменения во времени:
- применяется при нормализации данных, для группировки атрибутов

### Типы SCD:

- Нулевой тип (SCD0);
  - Первый тип (SCD1);
  - Второй тип (SCD2);
  - Третий тип (SCD3).
- 



# Нулевой тип (SCD0)

Значения атрибутов никогда не меняются (дата создания записи, дата и место рождения, серийный номер устройства)

Идентификатор пользователя	Место рождения
1	г. Краснодар
2	г. Екатеринбург

# Первый тип (SCD1)

- Для изменения значения атрибута применяется перезапись значения;
- Старое значение удаляется из БД;
- В таблице всегда находится только актуальное значение;
- Суррогатный ключ не изменяется.

**До изменения:**

Идентификатор	ФИО	Должность
1	Иванов Иван Иванович	Младший специалист

**После изменения:**

Идентификатор	ФИО	Должность
1	Иванов Иван Иванович	Старший специалист

# Второй тип (SCD2)

- При изменении значения атрибута добавляется новая строка;
- Позволяет сохранить историю изменения данных;
- Суррогатный ключ для новой записи будет также новым;

**До изменения:**

Идентификатор	ФИО	Должность
1	Иванов Иван Иванович	Младший специалист

# Второй тип (SCD2)

До изменения:

Идентификатор	ФИО	Должность
1	Иванов Иван Иванович	Младший специалист

После изменения:

Идентификатор	ФИО	Должность	Дата создания записи
1	Иванов Иван Иванович	Младший специалист	18.06.2022
2	Иванов Иван Иванович	Старший специалист	19.07.2023

# Третий тип (SCD3)

- Новые данные добавляются в качестве значений в другие колонки этой же строки;
- Используется редко;

**До изменения:**

Идентификатор	ФИО	Должность
1	Иванов Иван Иванович	Младший специалист

**После изменения:**

Идентификатор	ФИО	Должность	Предыдущая должность
1	Иванов Иван Иванович	Старший специалист	Младший специалист

# Методологии проектирования DWH


# Методологии проектирования DWH

① Классическая: 3NF + звезда/снежинка;

② Гибкие:

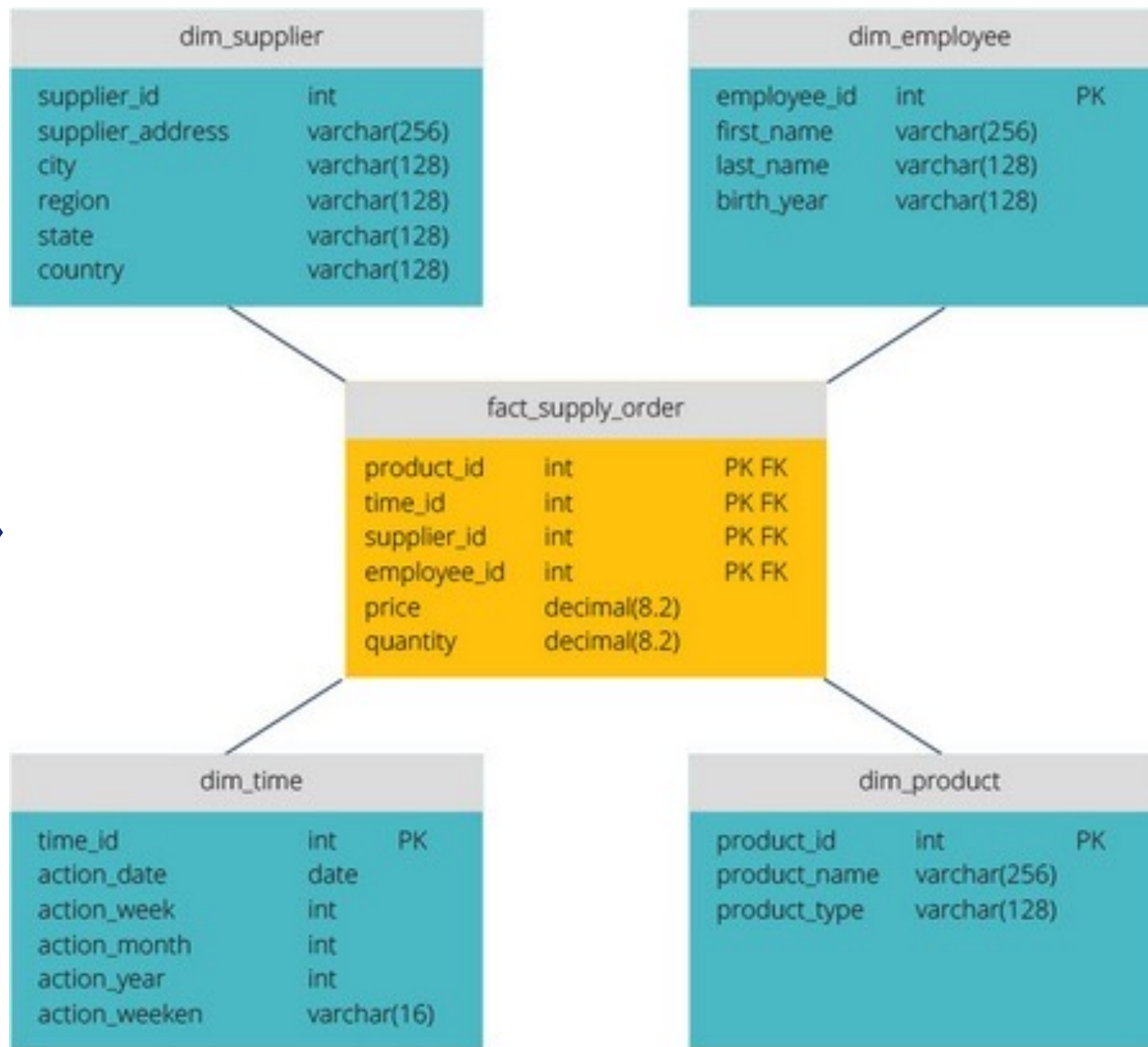
- Data Vault
- Anchor Model

# Схема «Звезда»

- Состоит из таблицы фактов и таблиц измерений;
  - Таблица фактов содержит данные о бизнес-процессах (транзакции покупок);
  - Таблицы измерений описывают данные таблицы фактов;
  - Таблицы измерений имеют один уровень глубины относительно таблицы фактов;
  - Наиболее денормализованная схема из используемых в продуктовых решениях.
- 



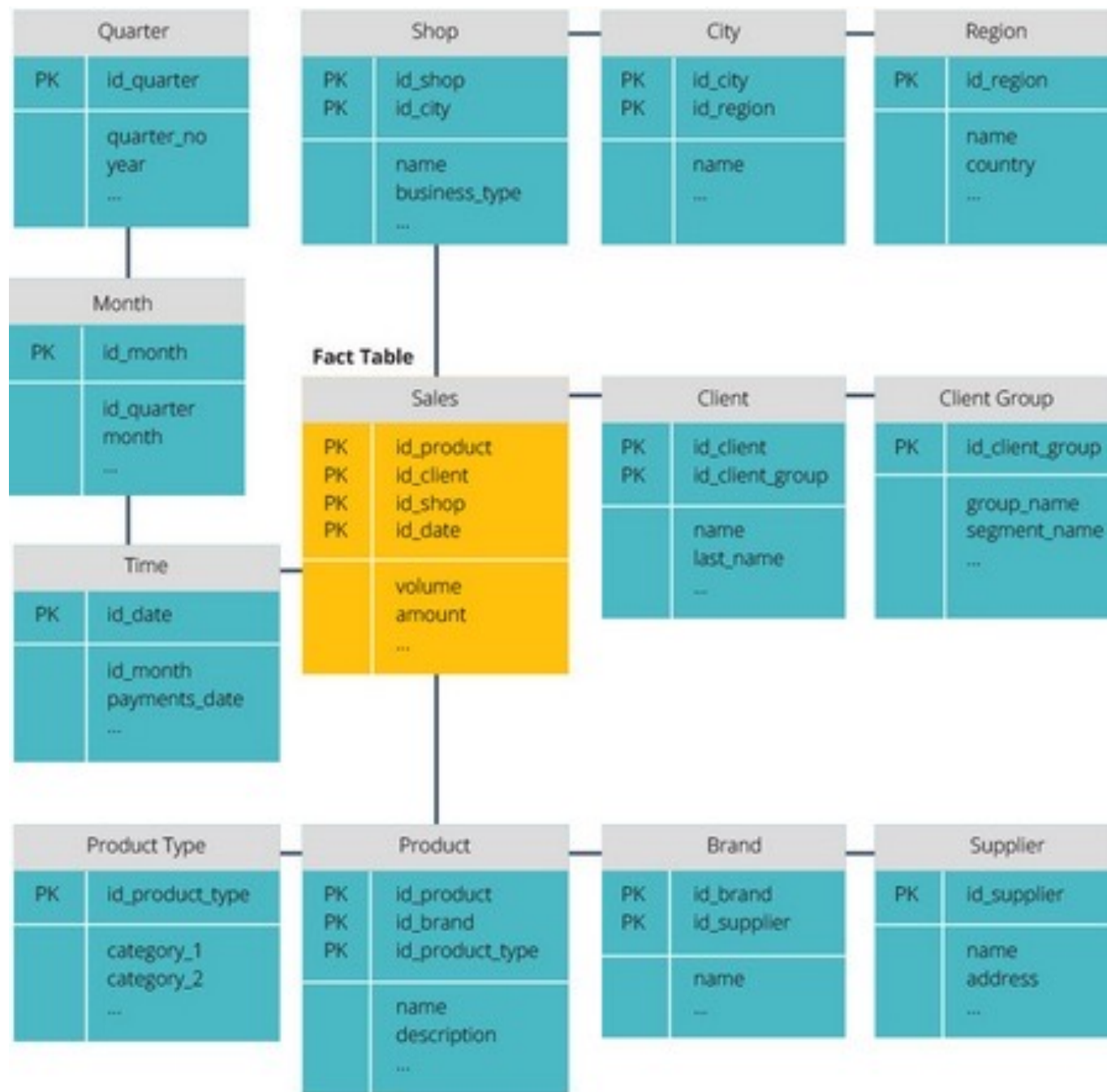
# Схема «Звезда»



# Схема «Снежинка»

- Более нормализованная «звезда»;
- Использует меньше дискового пространства за счет нормализации;
- Более сложные запросы, относительно «звезды»;
- Таблицы измерений могут иметь больше одного уровня глубины относительно таблицы фактов.

# Схема «Снежинка»



# Схема 3NF + «Звезда» / «Снежинка»

- Классический подход к проектированию DWH;

## Преимущества:

- + Простота реализации;
- + Небольшое кол-во джоинов;

## Недостатки:

- Жесткое определение связей между сущностями (сложно изменить тип связи между сущностями);
- При необходимости «версионировать» данные с помощью SCD2, объем таблиц растет очень быстро;
- Сложно масштабировать.

# Причины появления гибких методологий

- ➔ Быстрое изменение бизнес-процессов;
- ➔ Изменение связей между сущностями;
- ➔ ТЗ в начале и конце проекта могут сильно различаться;
- ➔ Внедрение Agile подходов.

# Что делает хранилище гибким?

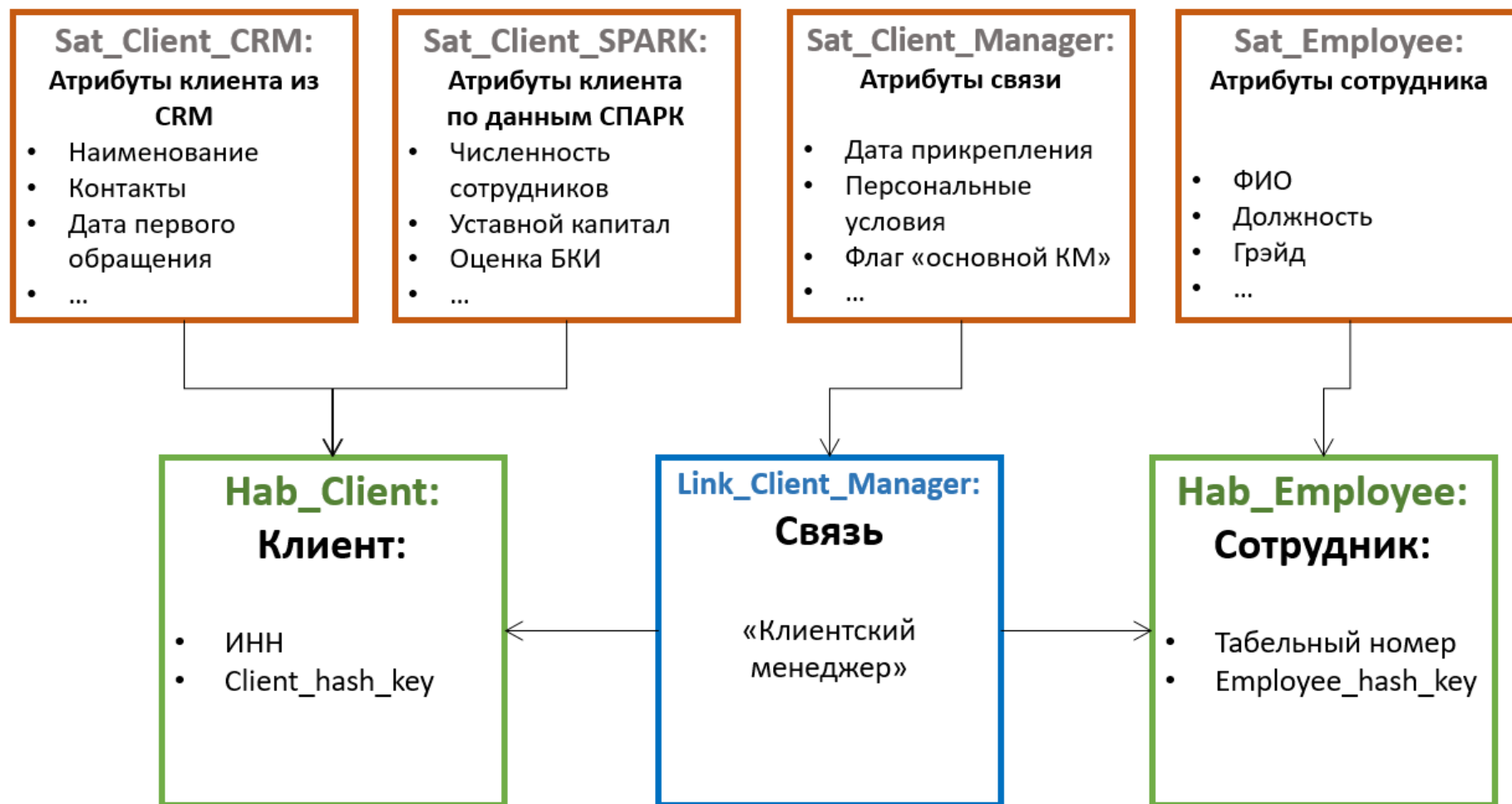
- ① Ранняя постановка и быстрая разработка MVP (Minimal Viable Product - минимально жизнеспособный продукт);
- ② Итеративная доработка (введение нового функционала не должно затрагивать существующую структуру);
- ③ Быстрая адаптация к изменениям бизнес-требований и процессов;

# Data Vault

Объекты модели:

Хаб (Hub) сущность с позиции бизнеса	Связь (Link) таблица связи хабов между собой	Спутник (Satellite) описательные атрибуты хаба или связи
<ul style="list-style-type: none"><li>идентификатор сущности (суррогатный ключ);</li><li>бизнес-ключ;</li><li>дата загрузки;</li><li>указание на источник данных.</li></ul> <p><u>Пример:</u></p> <ul style="list-style-type: none"><li>Продавец</li><li>Покупатель</li></ul>	<ul style="list-style-type: none"><li>тип связи M:M (многие ко многим);</li><li>ссылка может иметь свои собственные атрибуты;</li><li>каждая таблица-связь хранит идентификаторы связываемых таблиц и дату загрузки.</li></ul> <p><u>Пример:</u></p> <ul style="list-style-type: none"><li>Транзакции между продавцом и покупателем</li></ul>	<ul style="list-style-type: none"><li>атрибуты из разных источников хранят в разных спутниках;</li><li>атрибуты сущностей с различными типами изменений (SCD) хранят в разных спутниках;</li><li>как правило, спутники хранят историю изменений по SCD2.</li></ul> <p><u>Пример:</u></p> <ul style="list-style-type: none"><li>Адрес продавца</li><li>Имя покупателя</li></ul>

# Пример хранилища Data Vault



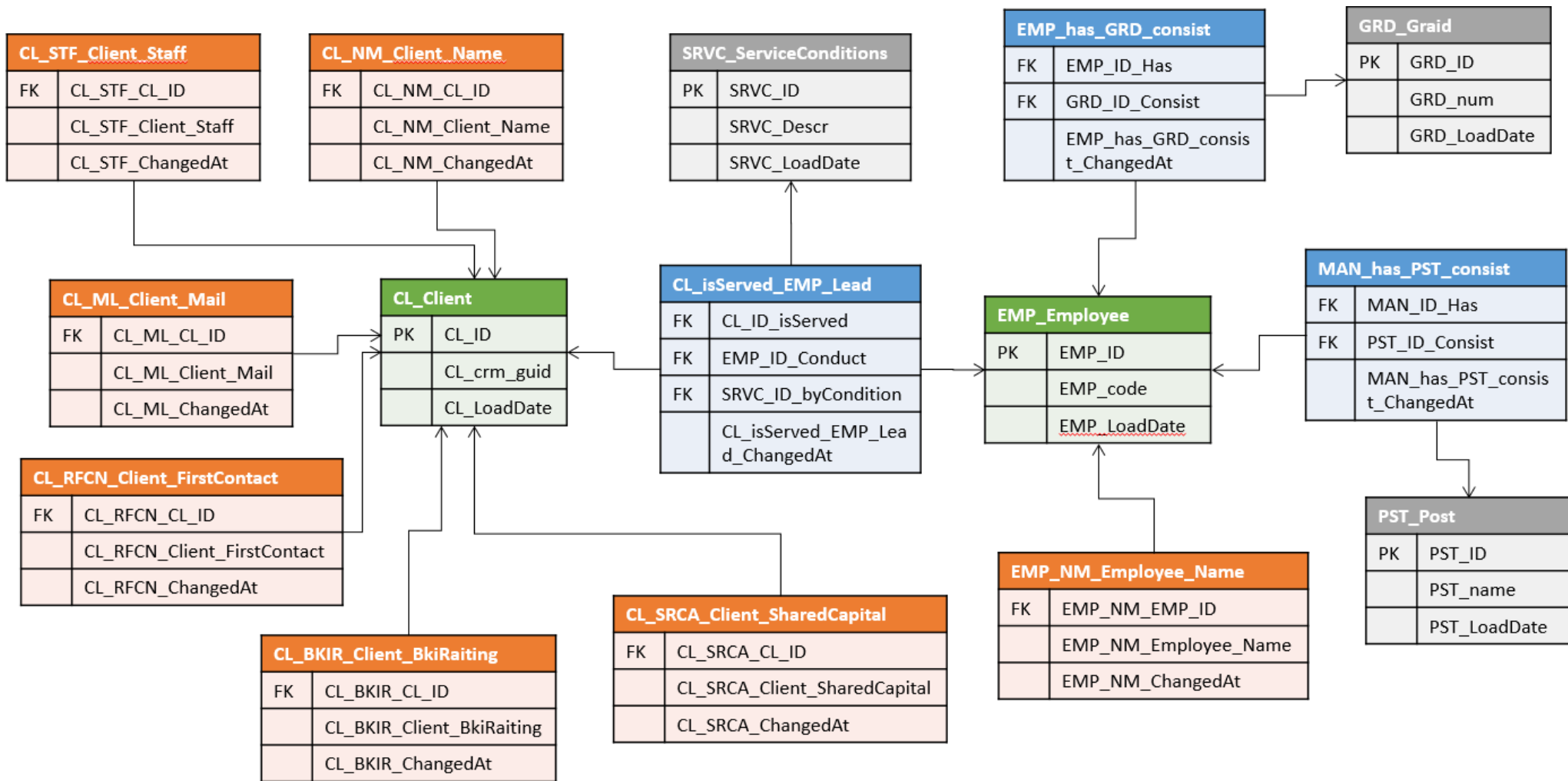


# Anchor Model (Якорная модель)

Объекты модели:

Якорь (Anchor) бизнес-сущность	Связь (Tie) таблица, для хранения связи между объектами	Атрибут (Attribute) таблица хранения атрибутов объектов.
<p><b>Содержит:</b></p> <ul style="list-style-type: none"><li>• суррогатный ключ;</li><li>• время и дата загрузки;</li><li>• ссылка на источник.</li></ul> <p><b>НЕ содержит бизнес-ключ</b></p> <p>С точки зрения якорной модели он является атрибутом</p>	<p>Связи не могут иметь атрибуты</p>	<p>На каждый атрибут – одна таблица</p> <p><b>Содержит:</b></p> <ul style="list-style-type: none"><li>• суррогатный ключ;</li><li>• время и дата загрузки;</li><li>• ссылка на источник;</li><li>• значение атрибута.</li></ul>

# Пример якорной модели



# Data Vault и Anchor Model

**Общее:**

- Гибкая структура, позволяющая быстро и легко добавлять новые атрибуты и сущности;
- Высокий уровень нормализации: много таблиц, много джоинов, тяжелые запросы;
- Связи – отдельные таблицы, а не атрибуты сущностей.

Различия	
<b>Update</b> – не приветствуется, но допустим	<b>Insert Only</b>
<b>Большинство</b> доработок не влияют на существующую структуру	<b>Любая</b> доработка не влияет на существующую структуру
<b>Связи</b> имеют свои атрибуты	<b>Связи</b> не имеют своих атрибутов
<b>Атрибуты объединяются</b> по частоте обновления и источникам	Максимальная <b>декомпозиция</b>

# Инструменты проектирования БД

# Инструменты проектирования БД



Drawlo

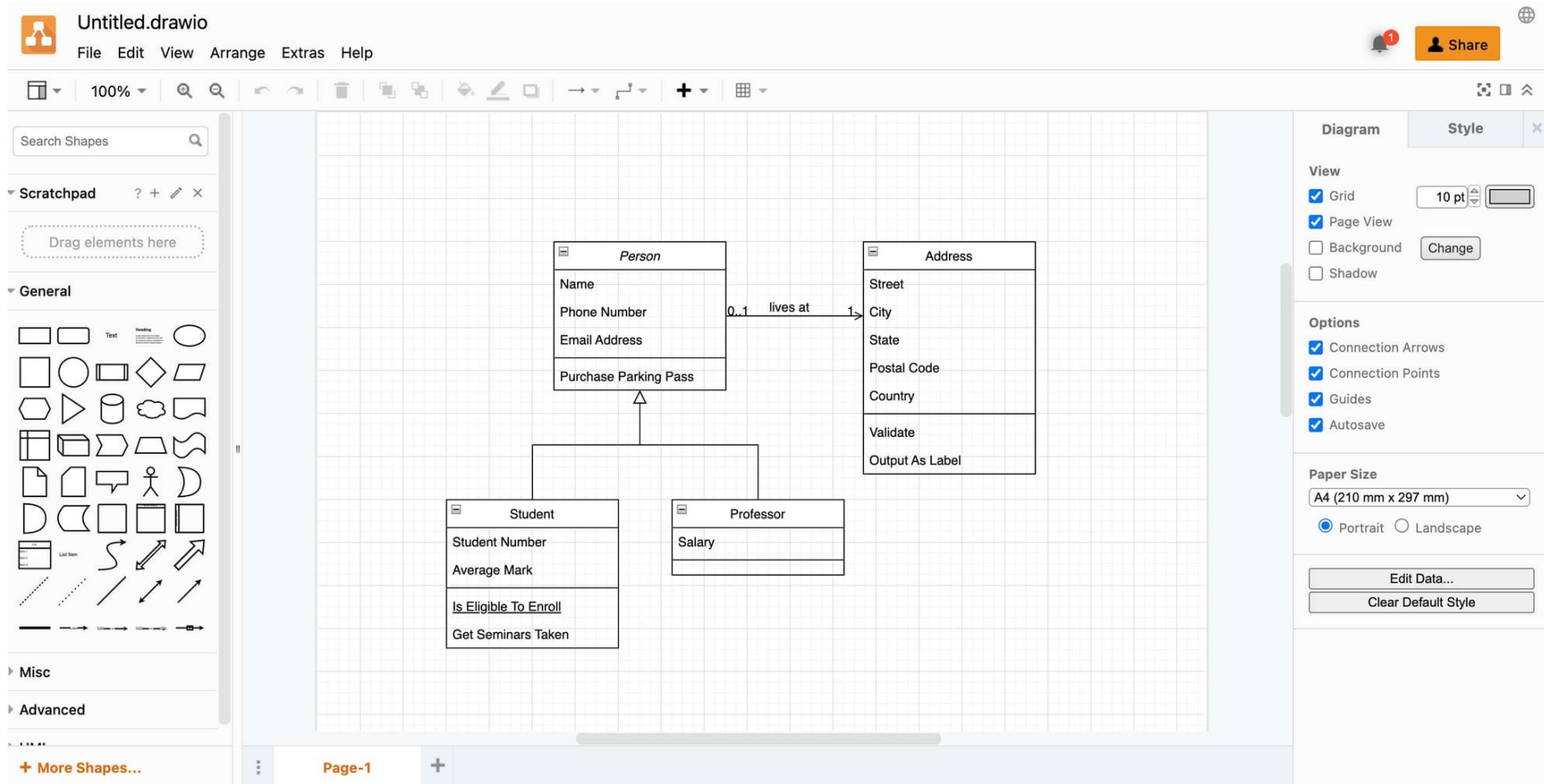


Miro

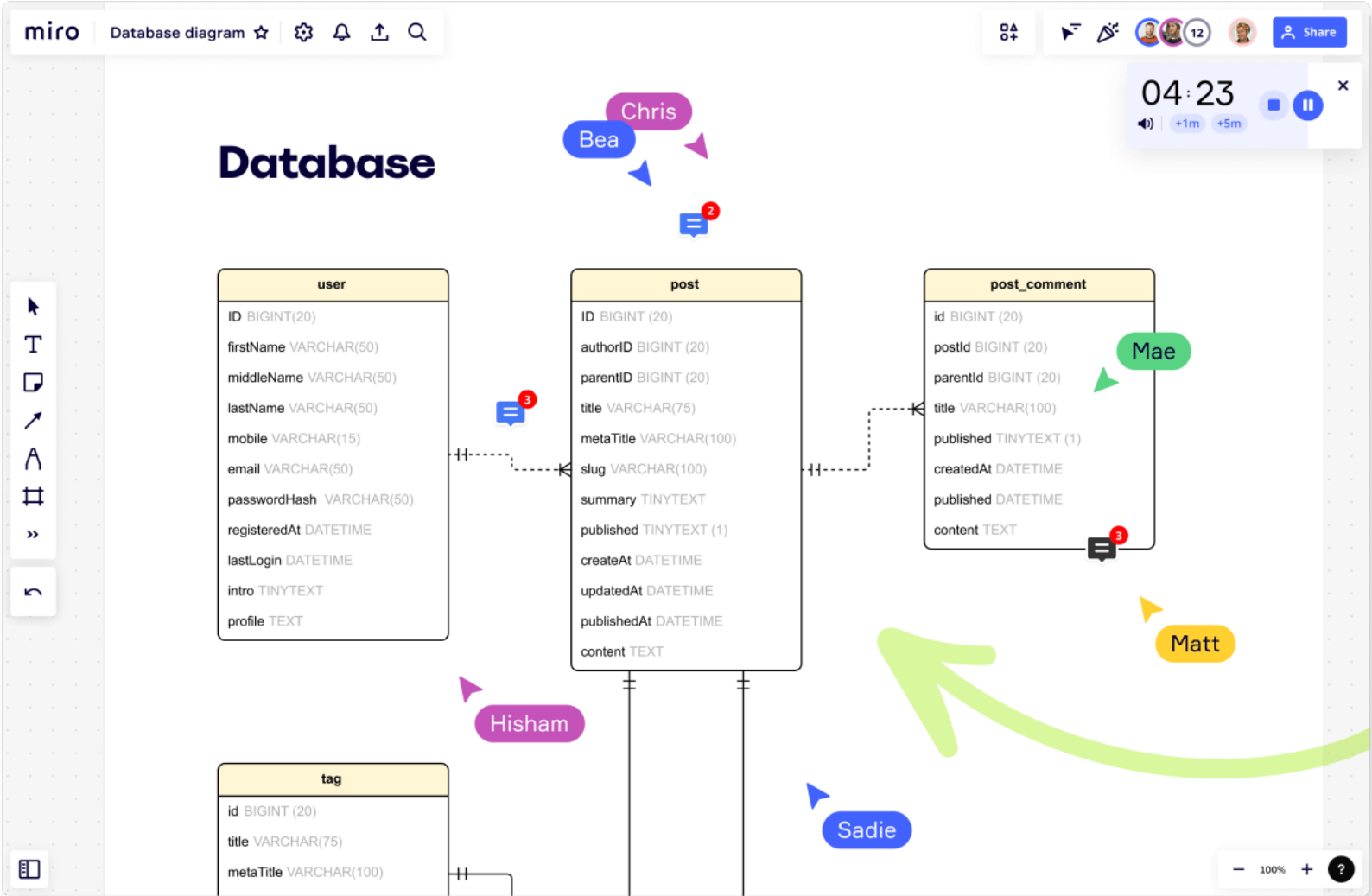


PgModeler

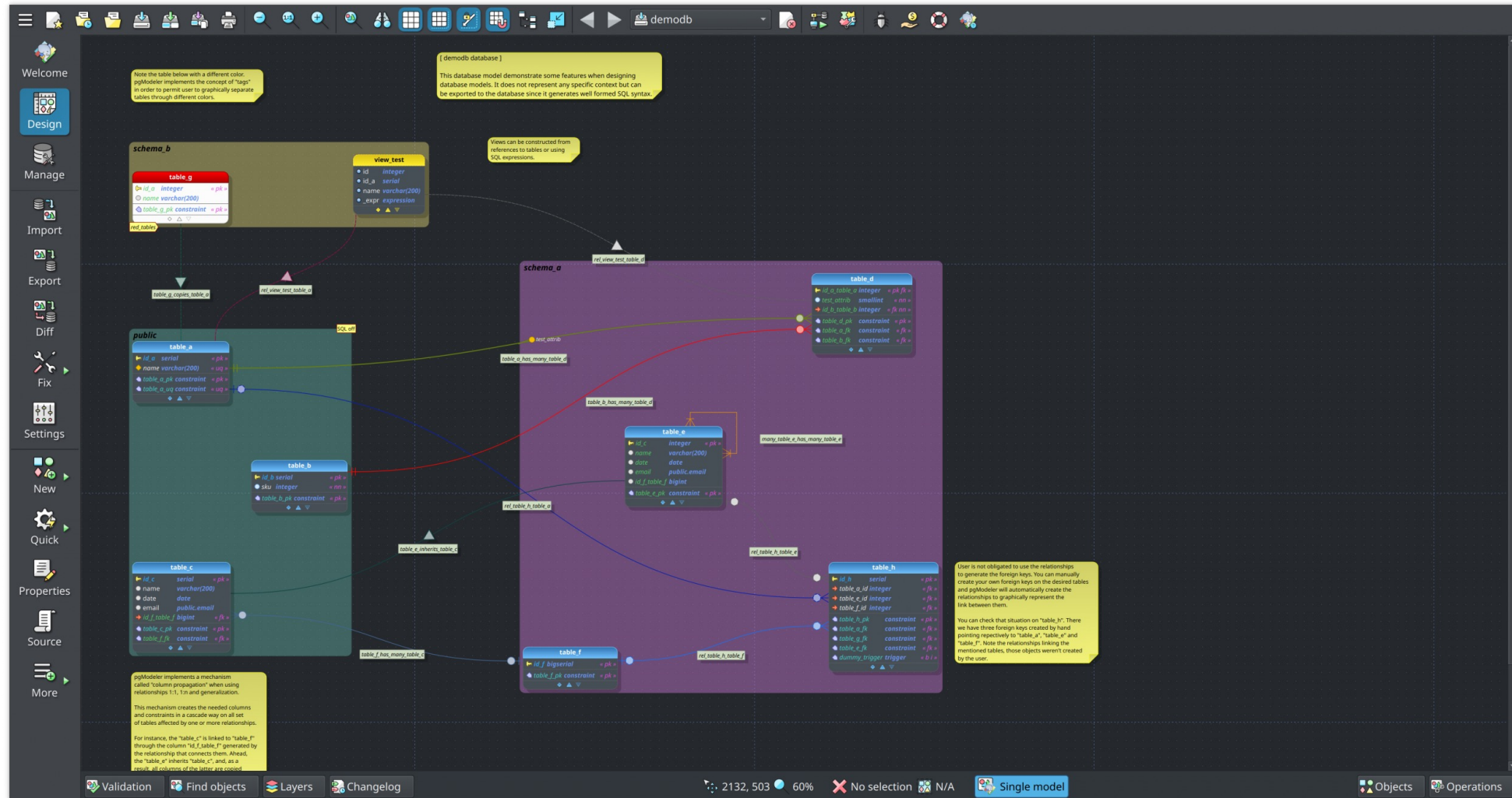
# Drawlo



# Miro



# PgModeler





Cron, Airflow

# Cron

**Cron** - планировщик задач, используемый для выполнения задач или запуска скриптов по расписанию;

**Crontab** - команда, которая используется, для управления планировщиком Cron. Команда `crontab` создает и редактирует файл `crontab`, содержащий команды и инструкции.

**Аналог Cron в Windows:** «Планировщик заданий».



# Основные команды Crontab

<b>crontab -l</b>	Вывод содержимого расписания текущего пользователя
<b>crontab -e</b>	Редактирование или создание файла расписания для текущего пользователя
<b>crontab -r</b>	Удаление файла расписания текущего пользователя
<b>crontab -u username</b>	Работа с расписаниями конкретных пользователей. Доступно только суперпользователю

# Шаблон заданий Crontab

minute(s) hour(s) day(s) month(s) weekday(s) command(s)

Поле	Диапазон значений	Описание
minute	0-59	Минута запуска команды
hour	0-23	Час запуска
day	1-31	Число (день) запуска
month	1-12	Месяц запуска
weekday	0-6	День недели (воскресенье = 0, понедельник = 1 и т.д.)
command		Последовательность команд для выполнения. Это могут быть команды, исполняемые файлы (например, скрипты) или комбинации файлов

# Пример команды Crontab

**Запуск скрипта каждый понедельник в 9:00 и 18:00 часов:**

```
0 9,18 * * 1 /test_crontask.py
```

**Запуск скрипта каждый день в 0:30 и 12:30:**

```
30 */12 * * * /test_crontask.py
```

# Аналог Cron в Python

Модуль **schedule** позволяет настроить периодическое выполнение задач внутри питона. Для запуска различных программ из питона используется модуль `subprocess`.

**Документация schedule:**

<https://schedule.readthedocs.io/en/stable/>

**Больше информации о subprocess:**

<https://schedule.readthedocs.io/en/stable/>

```
1  import schedule
2  import subprocess as sp
3
4
5  # Create function which do necessary job
6  def job():
7      """
8      Run ``echo "Test text"`` in a terminal
9      """
10     command = ['echo', 'Test text']
11     sp.run(command)
12
13
14     # Specify to run job every minute
15     schedule.every().minute.do(job)
16
17
18     # Run job
19     while True:
20         schedule.run_pending()
```

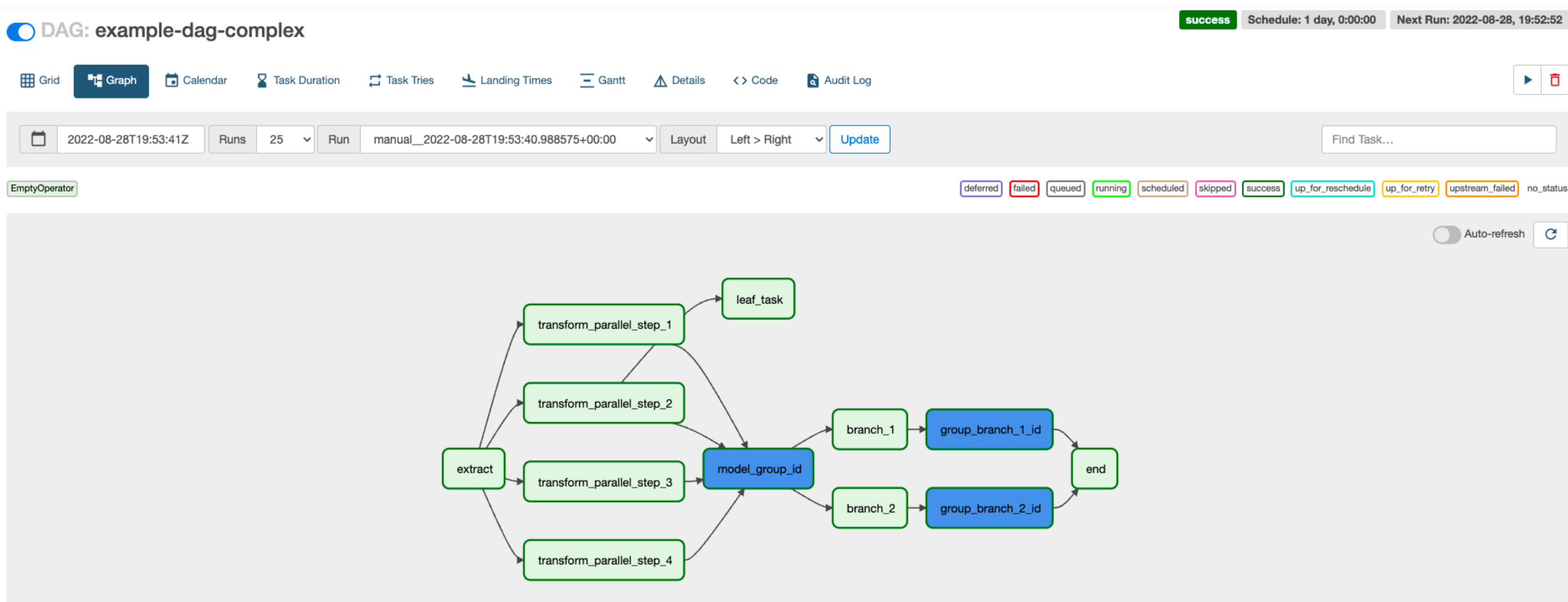


**Airflow** – инструмент, позволяющий создавать различные последовательности задач и ставить их на расписание.

Основные сущности Airflow:

- DAG, Directed Acyclic Graph (Направленный ациклический граф) – совокупность задач, которые выполняются по определённому расписанию с заданной последовательностью;
- Task (задача) – выполняет указанное действие (запуск python/bash/sql скрипта).

# Airflow DAG пример





# Airflow DAG пример

DAGs

Datasets

Security

Browse

Admin

Docs

11:49 UTC

AU

DAG: example-dag-complex

Schedule: 0:05:00

Next Run: 2022-08-31, 11:43:45

Grid

Graph

Calendar

Task Duration

Task Tries

Landing Times

Gantt

Details

Code

Audit Log

31.08.2022, 11:48:26

25

All Run Types

All Run States

Clear Filters

deferred

failed

queued

running

scheduled

skipped

success

up\_for\_reschedule

up\_for\_retry

upstream\_failed

no\_status

Auto-refresh

→

⌵

Duration

00:00:22

00:00:11

00:00:00

Aug 31, 11:44

extract

transform\_parallel\_step\_1

transform\_parallel\_step\_2

transform\_parallel\_step\_3

transform\_parallel\_step\_4

model\_group\_id

transform\_sequential\_step\_1

transform\_sequential\_step\_2

transform\_sequential\_step\_3

parallel\_step\_1

parallel\_step\_2

branch\_1

branch\_2

group\_branch\_1\_id

group\_branch\_2\_id

end

leaf\_task

DAG

example-dag-complex

Run

2022-08-31, 11:43:25 UTC

DAG Run Details

Graph

Mark Failed

Mark Success

Re-run:

Clear existing tasks

Queue up new tasks

Status

success

Run ID

manual\_\_2022-08-31T11:48:25.641513+00:00

Run type

manual

Run duration

00:00:10

Last scheduling decision

2022-08-31, 11:48:36 UTC

Started

2022-08-31, 11:48:25 UTC

Ended

2022-08-31, 11:48:36 UTC

Data interval start

2022-08-31, 11:43:25 UTC

Data interval end

2022-08-31, 11:48:25 UTC

04

Практика

