ТИНЬКОФФ

Как построить data lineage.

Обзор решений и опыт нашей команды

Дима Зуев

17.10.2022

План доклада

01

Что такое lineage? Каким он бывает? Для чего можно использовать?

02

Подходы к формированию data lineage. Pros&cons.

03

Обзор решений для работы с data lineage.

04

Для чего мы используем lineage в TEDI? Цели и кейсы

05

Наша реализация: что мы пробовали, на чем остановились, на какие грабли наступили, trade offs.

06

Выводы

Пирамида потребностей в данных

Базовые потребности (collection)

Доступность данных, актуальность

Удобство (description)

Data Quality, Data Catalog, Lineage

Высшие (growth)

Сложный анализ, предсказания

Определение lineage

Очень зависит от потребностей, но попробуем

Определение #1

это набор сложных отношений между датсетами и трансформациями в пайплайне.

Определение #2

процесс *понимания*, документирования и визуализации того, как данные проходят через экосистему

Определение #3

информация, которая описывает движение данных от источника их происхождения по точкам обработки и применения

Области применения

Описать все возможные варианты невозможно.

Из нашего опыта:

- отслеживание зависимостей
- системный анализ для доработок etl
- поиск узких мест (оптимизация процессов)
- change management
- анализ первопричин: аномалии, сбои
- информационная безопасность
- комплаенс: персональные данные, чувствительные данные
- etc

Области применения, обобщение

Поиск источников

отслеживание зависимостей, системный анализ, поиск, узких мест (оптимизация процессов), change management

Анализ влияния

change management, анализ первопричин: аномалии, сбои, информационная безопасность, комплаенс

Подходы к построению lineage

Вручную

Суть подхода в явном указании зависимостей.

Кто использует: SSIS, Pentaho DI

Pros:

- Простота в реализации
- Можно связывать black boxes
- Поддержка множеством etl инструментов

Cons:

- Сложно разрабатывать и поддерживать
- Column-column lineage обычно не доступен

По мета-данным

Необходимо заполнять мета-данные трансформаций

Кто использует: Informatica, SAS

Pros:

- Автоматизированный подход
- Возможен column-column lineage
- Граф выполнения исходя из графа зависимостей

Cons:

- Мета-данные могут не соответствовать действительности
- Сложность в описании неявных зависимостей

Автоматически

Мета-данные извлекаются из кода (настроек)

Кто использует: dbt (table-table only), atlan, TEDI

Pros:

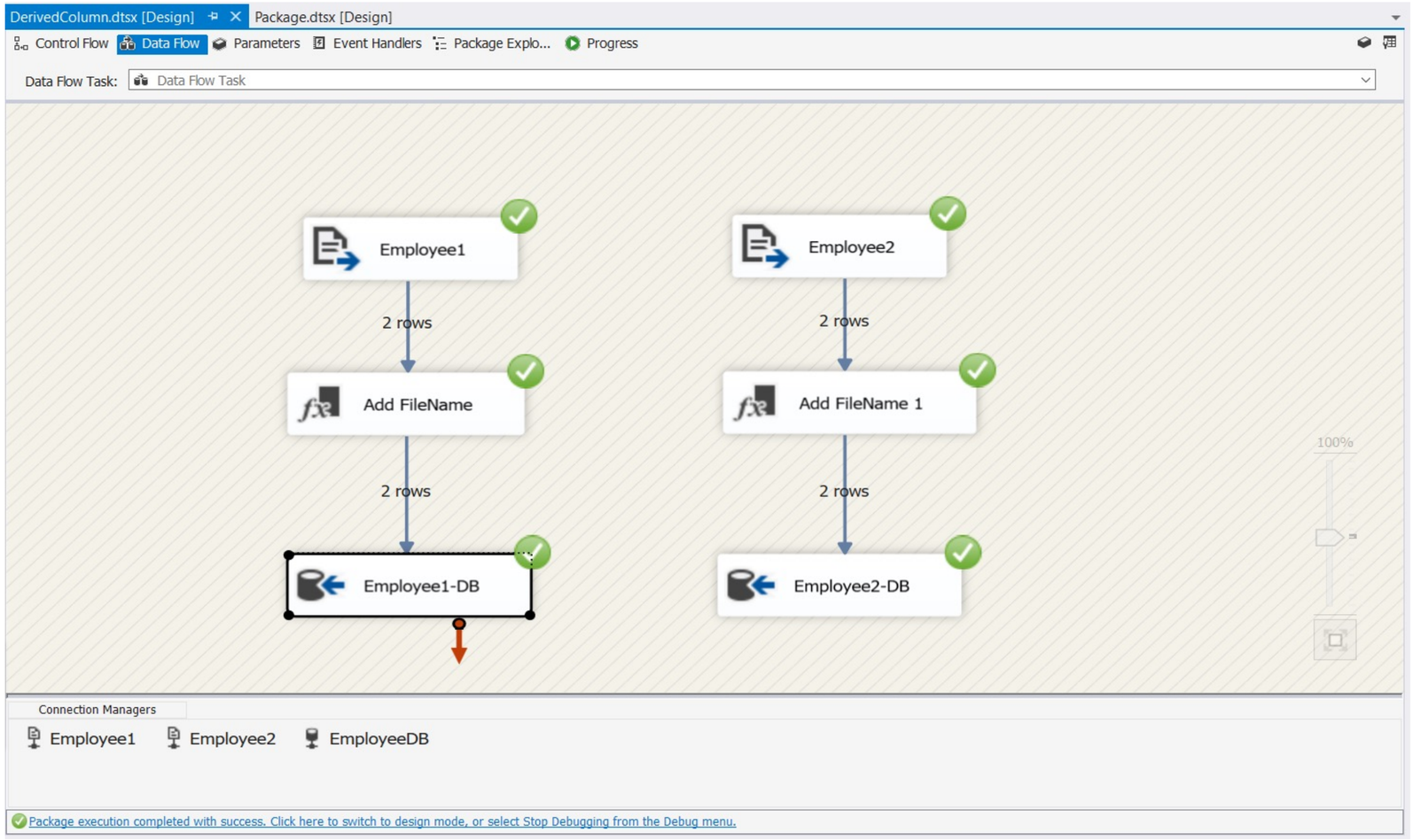
- Не требует дополнительных действий
- Всегда актуальное состояние
- Возможен column-column lineage
- Граф выполнения исходя из графа зависимостей

Cons:

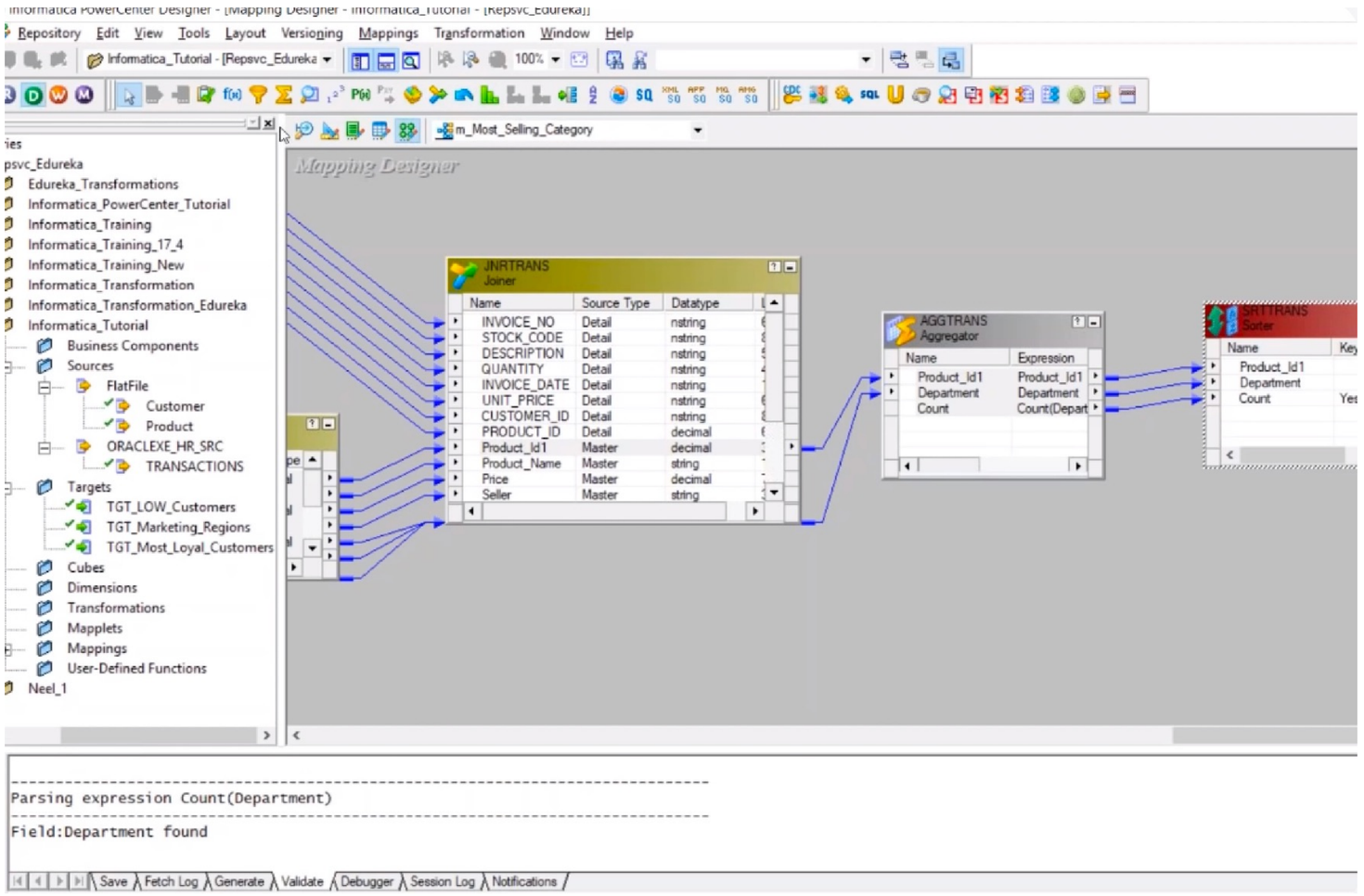
- Ограниченное количество инструментов
- Большие различии у синтаксисов

Реализации

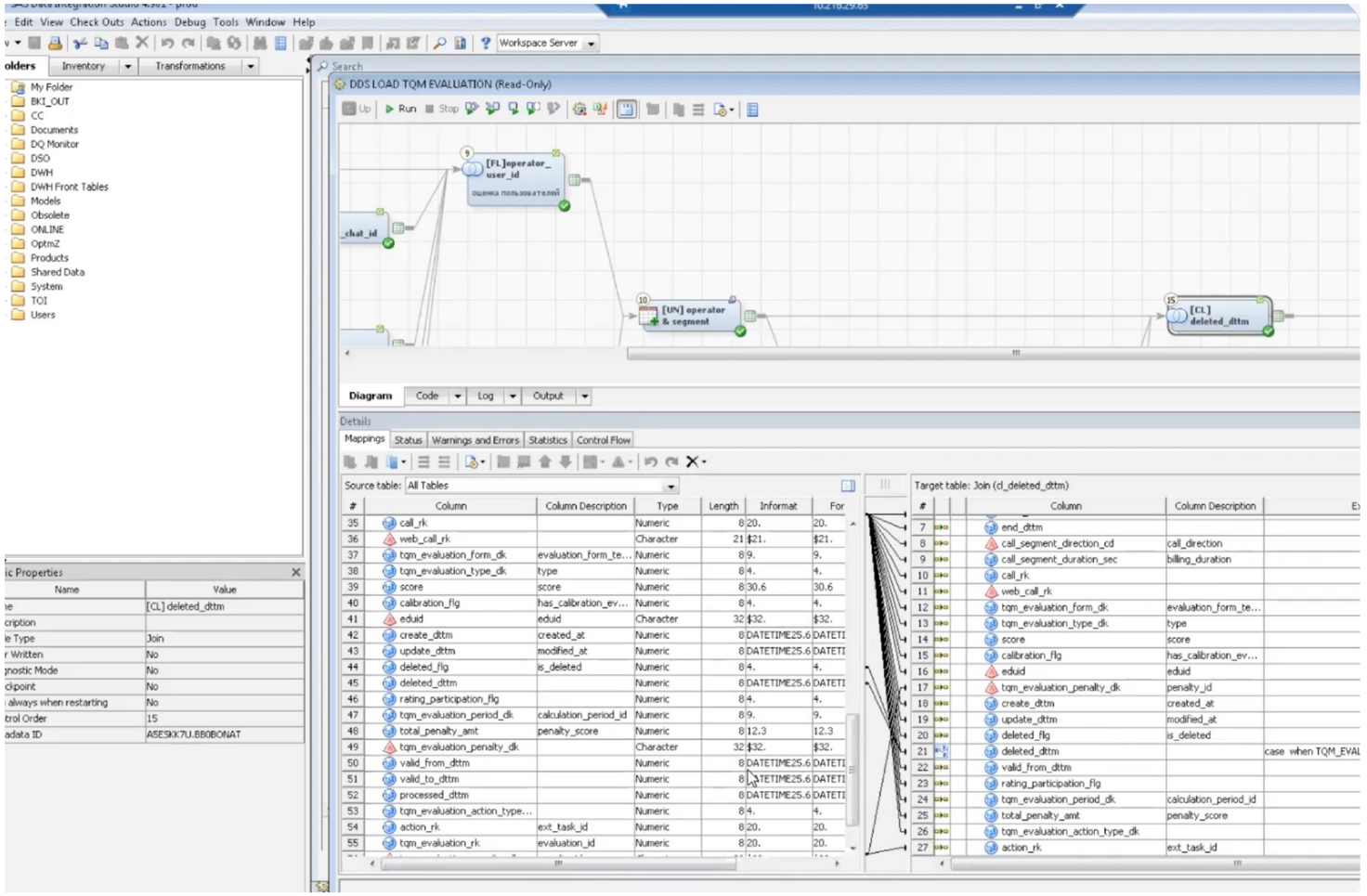
SSIS


DerivedColumn.dtsx [Design]
Package.dtsx [Design]
Control Flow
Data Flow
Parameters
Event Handlers
Package Explo...
Progress
Data Flow Task:
Data Flow Task
Employee1
2 rows
Add FileName
2 rows
Employee1-DB
Employee2
2 rows
Add FileName 1
2 rows
Employee2-DB
100%
Connection Managers
Employee1
Employee2
EmployeeDB
Package execution completed with success. Click here to switch to design mode, or select Stop Debugging from the Debug menu.

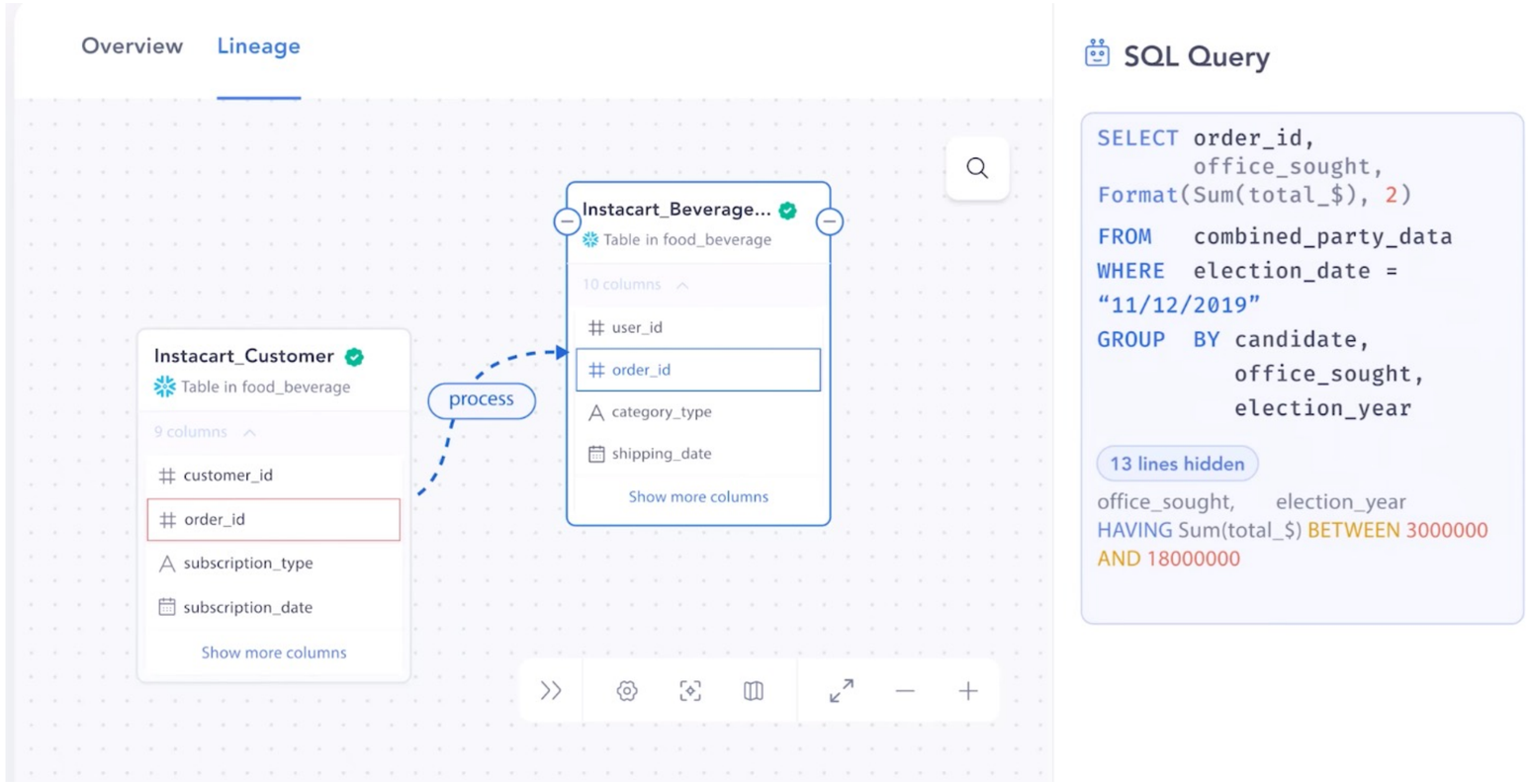
Informatica



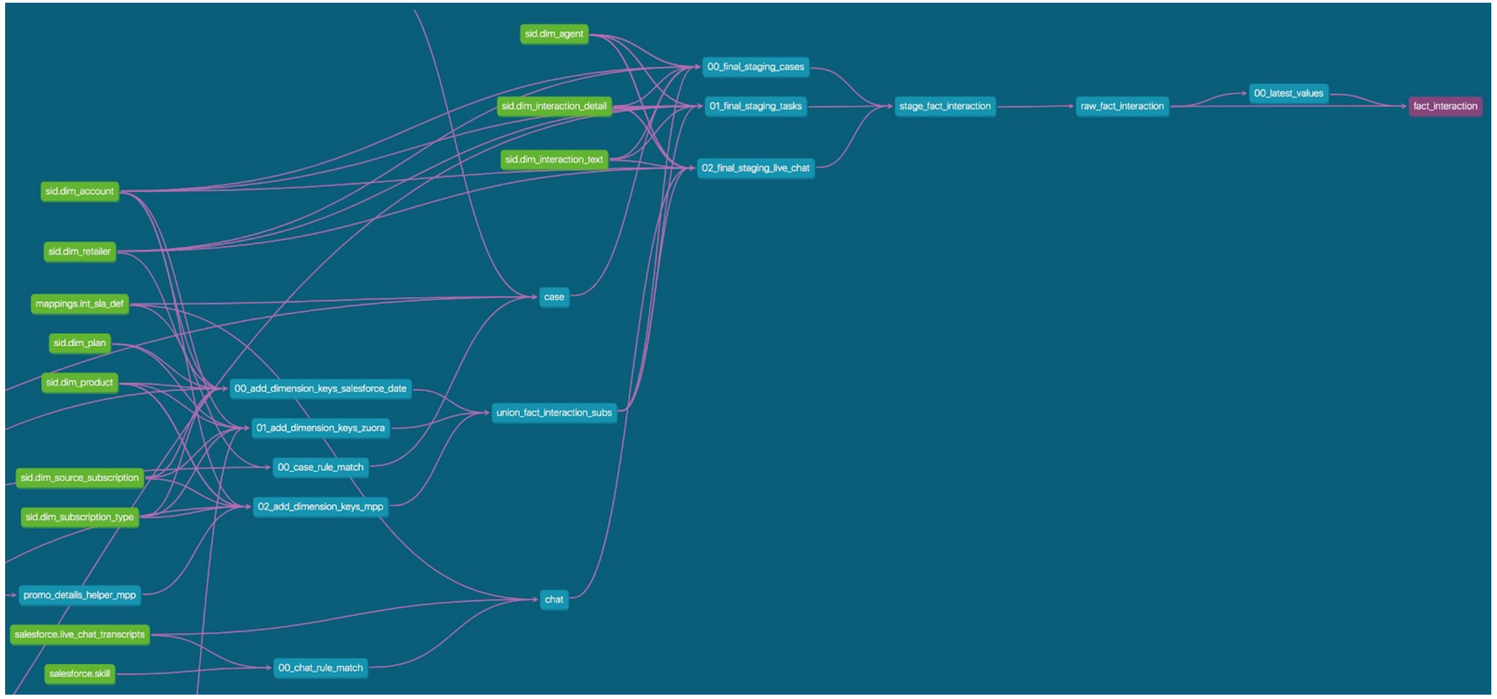
SAS


Edit View Check Outs Actions Debug Tools Window Help
Workspace Server
Inventory
Transformations
My Folder
BKI_OUT
CC
Documents
DQ Monitor
DSO
DWH
DWH Front Tables
Models
Obsolete
ONLINE
OptmZ
Products
Shared Data
System
TOI
Users
Search
DDS LOAD TQM EVALUATION (Read-Only)
Run
Stop
[FL]operator_ user_id
оценка пользователей
[UN] operator & segment
[CL] deleted_dttm
Diagram
Code
Log
Output
Details
Mappings
Status
Warnings and Errors
Statistics
Control Flow
Source table: All Tables
Target table: Join (cl_deleted_dttm)
Name
Value
[CL] deleted_dttm
Join
No
No
No
15
A5E5KK7U.BB0BONAT

atlan


Overview
Lineage
Instacart_Customer
Table in food_beverage
9 columns
customer_id
order_id
subscription_type
subscription_date
Show more columns
process
Instacart_Beverage...
Table in food_beverage
10 columns
user_id
order_id
category_type
shipping_date
Show more columns
SQL Query
SELECT order_id,
office_sought,
Format(Sum(total_$), 2)
FROM combined_party_data
WHERE election_date =
“11/12/2019”
GROUP BY candidate,
office_sought,
election_year
13 lines hidden
office_sought, election_year
HAVING Sum(total_$) BETWEEN 3000000
AND 18000000

dbt


sid.dim_agent
00_final_staging_cases
sid.dim_interaction_detail
01_final_staging_tasks
stage_fact_interaction
raw_fact_interaction
00_latest_values
fact_interaction
sid.dim_interaction_text
02_final_staging_live_chat
sid.dim_account
sid.dim_retailer
mappings.int_sla_def
case
sid.dim_plan
sid.dim_product
00_add_dimension_keys_salesforce_date
union_fact_interaction_subs
01_add_dimension_keys_zuora
00_case_rule_match
sid.dim_source_subscription
02_add_dimension_keys_mpp
sid.dim_subscription_type
promo_details_helper_mpp
chat
salesforce.live_chat_transcripts
00_chat_rule_match
salesforce.skill

dbt: реализация

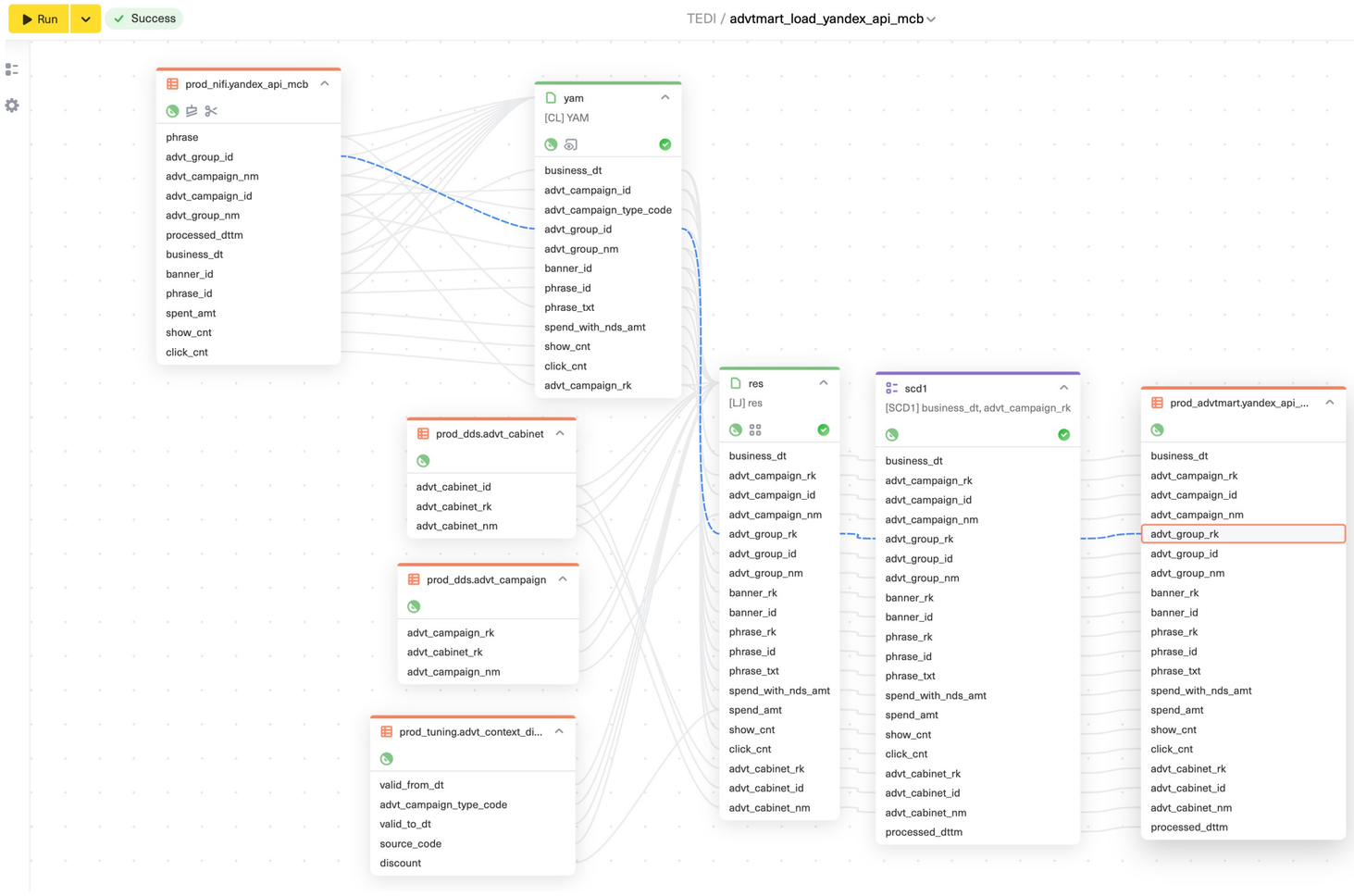
dbt обрабатывает шаблонный код для материализации запросов.

- Описание моделей в yaml
- Шаблон на jinjia2
- *ref* функция для нахождения зависимостей в шаблоне

Можно сломать lineage использовав функцию *ref,* например в комментарии к sql:

```
SELECT * FROM {{ ref('tbl0') }}
-- INNER JOIN {{ ref('tbl1') }} USING (a)
```

TEDI


Run
Success
TEDI / advtmart_load_yandex_api_mcb
prod_nifi.yandex_api_mcb
phrase
advt_group_id
advt_campaign_nm
advt_campaign_id
advt_group_nm
processed_dttm
business_dt
banner_id
phrase_id
spent_amt
show_cnt
click_cnt
yam
[CL] YAM
business_dt
advt_campaign_id
advt_campaign_type_code
advt_group_id
advt_group_nm
banner_id
phrase_id
phrase_txt
spend_with_nds_amt
show_cnt
click_cnt
advt_campaign_rk
prod_dds.advt_cabinet
advt_cabinet_id
advt_cabinet_rk
advt_cabinet_nm
prod_dds.advt_campaign
advt_campaign_rk
advt_cabinet_rk
advt_campaign_nm
prod_tuning.advt_context_di...
valid_from_dt
advt_campaign_type_code
valid_to_dt
source_code
discount
res
[LJ] res
business_dt
advt_campaign_rk
advt_campaign_id
advt_campaign_nm
advt_group_rk
advt_group_id
advt_group_nm
banner_rk
banner_id
phrase_rk
phrase_id
phrase_txt
spend_with_nds_amt
spend_amt
show_cnt
click_cnt
advt_cabinet_rk
advt_cabinet_id
advt_cabinet_nm
scd1
[SCD1] business_dt, advt_campaign_rk
business_dt
advt_campaign_rk
advt_campaign_id
advt_campaign_nm
advt_group_rk
advt_group_id
advt_group_nm
banner_rk
banner_id
phrase_rk
phrase_id
phrase_txt
spend_with_nds_amt
spend_amt
show_cnt
click_cnt
advt_cabinet_rk
advt_cabinet_id
advt_cabinet_nm
processed_dttm
prod_advtmart.yandex_api_...
business_dt
advt_campaign_rk
advt_campaign_id
advt_campaign_nm
advt_group_rk
advt_group_id
advt_group_nm
banner_rk
banner_id
phrase_rk
phrase_id
phrase_txt
spend_with_nds_amt
spend_amt
show_cnt
click_cnt
advt_cabinet_rk
advt_cabinet_id
advt_cabinet_nm
processed_dttm

TEDI: редактор

Основные абстракции редактора:

- Источники и таргеты (таблицы)
- Операторы (настройки):
  - SQL трансформация
  - Загрузчики: SCD1, SCD2
  - Генератор ключей
  - И еще 10 операторов
- «Времянки» (объект в базе, созданный оператором)
- Связи (между объектами БД):
  - Источник-оператор
  - Оператор-оператор («времняка»-«времняка»)
  - Оператор-таргет

TEDI: оркестратор

Основные абстракции оркестратора:

- Операторы (исполняемая логика, Airflow Operator):
  - Создание таблиц и вьюх
  - Запуск библиотечного кода загрузчиков и трансформаций
- Связи: порядок исполнения

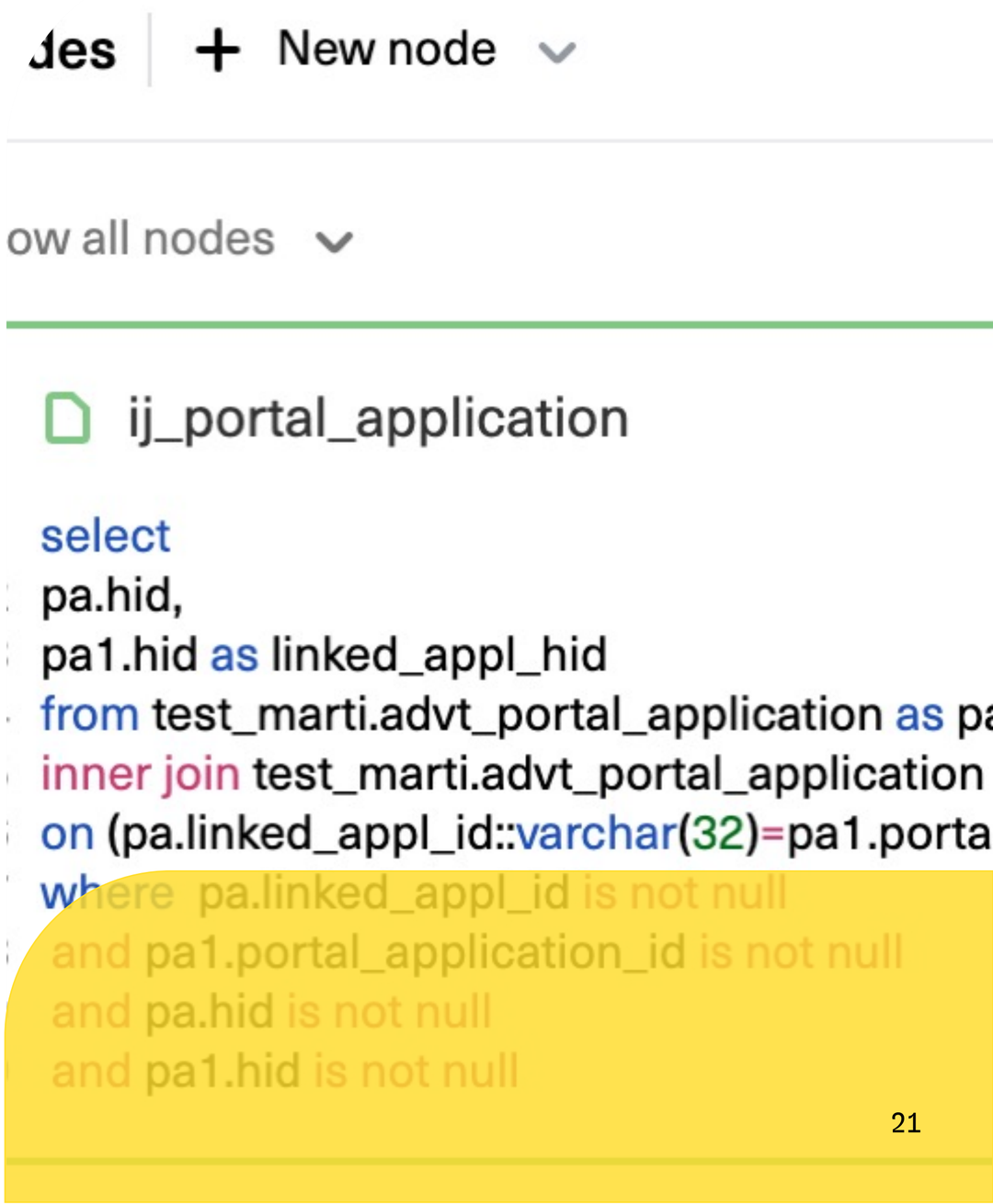
Порядок исполнения строится на связях между объектами

SQL оператор

Код трансформации

Любой* SELECT запрос, в том числе с CTE и рекурсивными CTE.

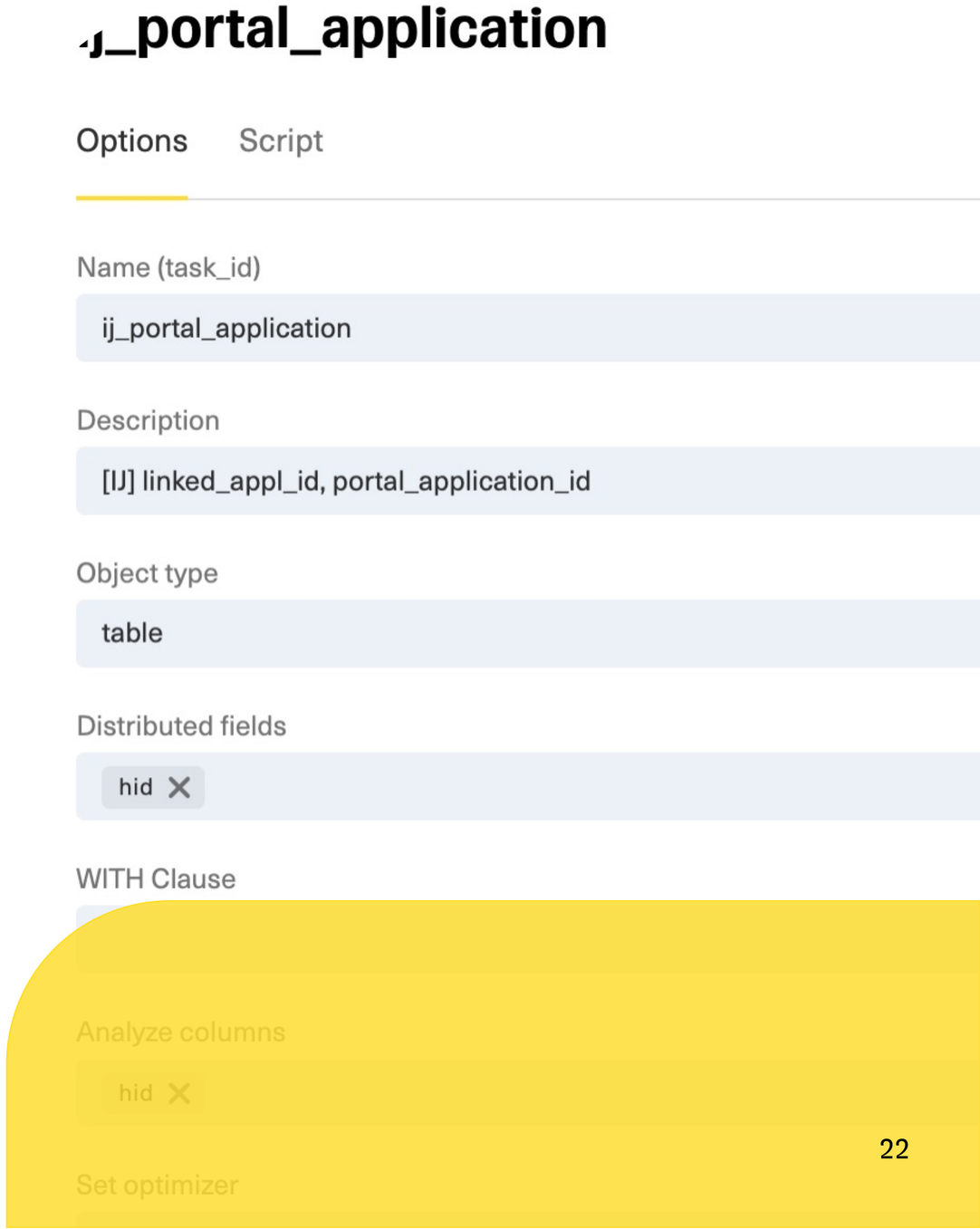
* С некоторыми ограничениями



SQL оператор

Настройки трансформации

Дополнительные параметры для создания объекта: тип объекта, распределение и тд



Lineage из sql кода

Статический анализ SQL

Статический анализ - анализ программного обеспечения, производимый (в отличие от динамического анализа) без реального выполнения исследуемых программ.

Что может sql парсер:

- Проверить синтаксис
- Построить AST

Примеры статических анализаторов SQL: Calcite, python-sqlparse, libpg_query

ANTLR грамматики

ANTLR принимает в качестве входных данных грамматику, определяющую язык, и генерирует код парсера для целевого языка (Java, Python, C++).

Pros:

- Генерирует код в множество языков: Java, Python, C++, Go, Rust
- Доступно много SQL грамматик: postgresql, mysql, hive, trino

Cons:

- Производительность
- Большинство грамматик *неофициальные*

Парсинг запросов GP

Greenplum популярная MPP база, НО:

- Нет официальной ANTLR грамматики
- Внутри GP лексер и парсер реализованы на flex и bison
- Расширенный синтаксис относительно sql92
- Generic решения типа Calcite и python-sqlparse не справляются

Попытка #1

Пробовали ANTLR:

- Взяли грамматику для PG
- Допилили поддержку GP: distributed в ddl, etc
- Ломалось на сложных запросах
- Высокая сложность фиксов, в некоторых случаях казалось, что починить уже не получится
- Плохой перфоманс, жрало много ресурсов

Попытка #2

Пробовали libpg_query (pg_last)**:**

- Нативный парсер грамматики **Postgresql**
- *C* код с биндингами под питон
- Прожевывает селекты любой сложности
- Очень производительно
- Blackbox

Trade offs

В итоге мы остановились на варианте с pg_last. Какие ограничения остались:

- только DML запросы
- Поддержка синтаксиса pg/gp, другие движки не получится использовать

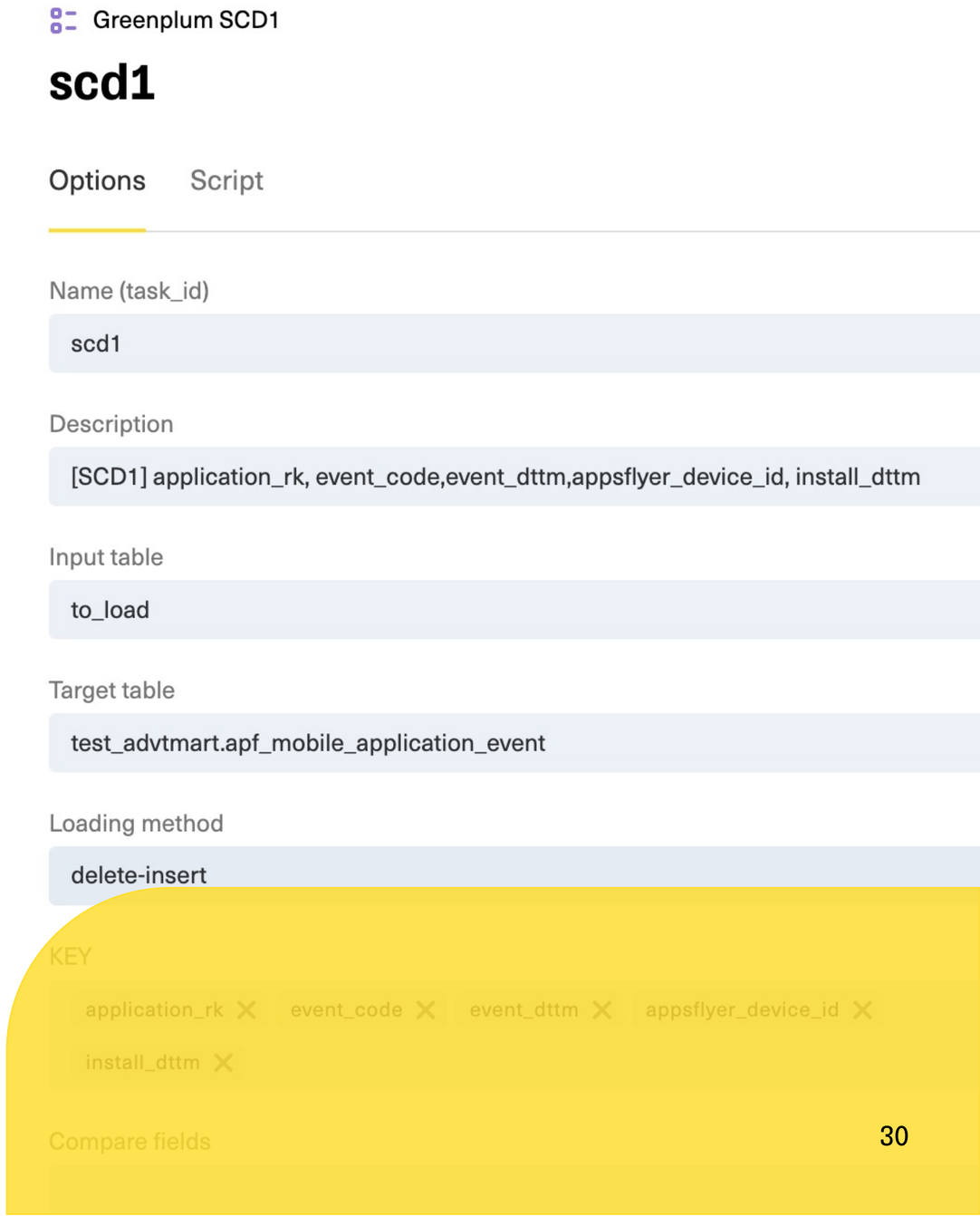
Ограничения связанные с статическим анализом:

- У нас нет state of the world, мы не знаем реальный ddl источника
  - SELECT * запрещен
  - Возможно обращение к несуществующей таблице
  - Возможно обращение к несуществующему полю
- Статически невозможно разрулить ambiguous column name (e.g. SELECT t0.col, t1.col FROM t0, t1)
  - Обращение к полю без указания имени таблицы
  - Обязательное указание алиасов в селекте

Пример оператора (SCD1)

Настройки трансформации

Входные параметры для исполнения оператора: входная/выходная таблица, ключи загрузки, алгоритм и тд



Lineage из параметров оператора

Lineage операторов

TEDI поддерживает не только SQL преобразования.

А значит и на lineage они тоже влияют. Каким бы не был сложным оператор, он ссылается на сущности в бд или временную таблицу. Некоторые операторы образуют связи неявно.

В операторах работает валидация, НО только в случае с ссылками на «наблюдаемый» сущности. В случае с ссылками на объекты БД, подсказать или провалидировать мы не можем. У нас нет **state of the world.**

Lineage в TEDI

Цели и кейсы

Нужды ETL разработчиков

Визуализация всех зависимых полей в процессе: явные зависимости и зависимости из фильтров и условий соединений; отображение различных трансформаций

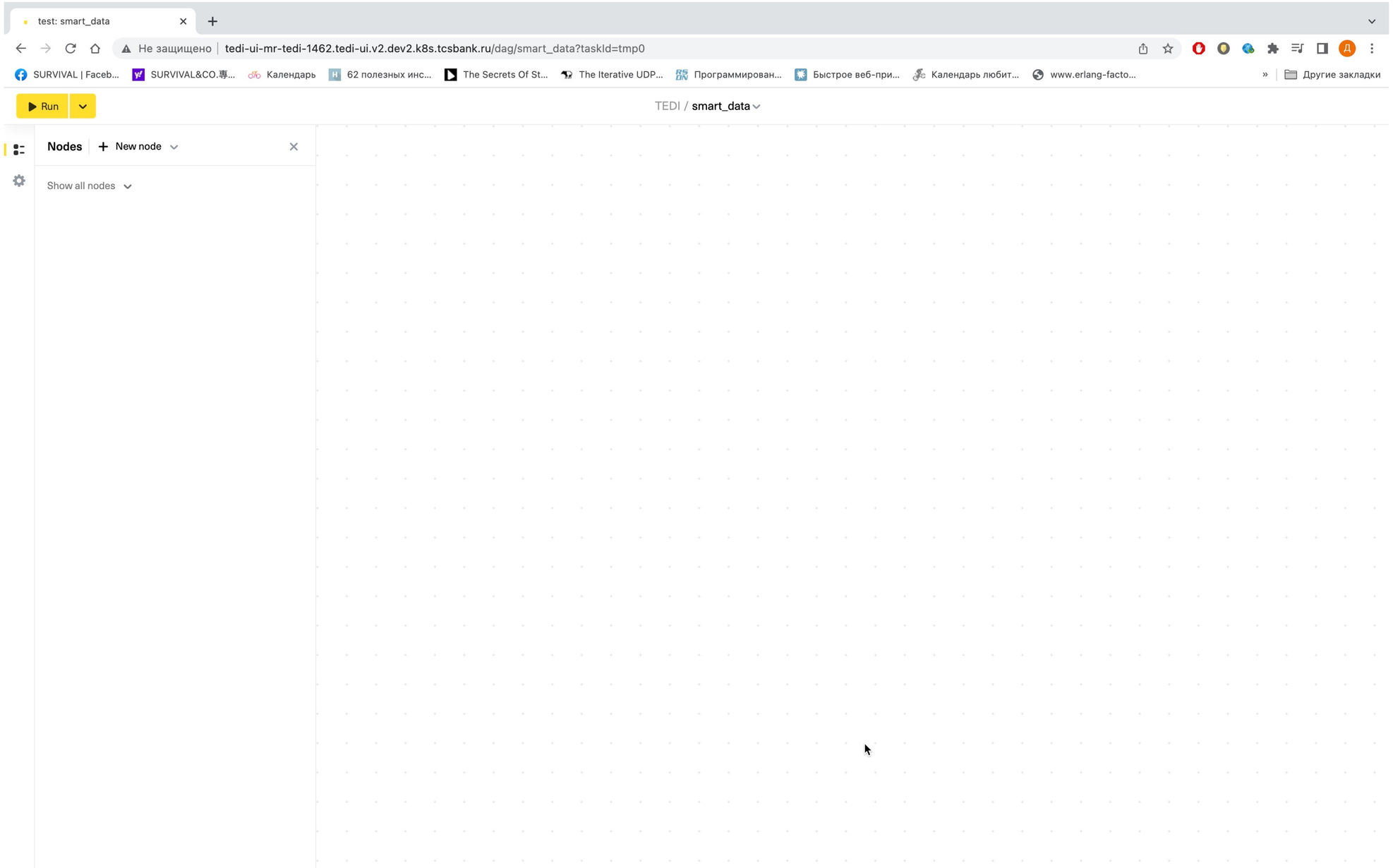
Нужды системных аналитиков

Оценка влияния изменений, анализ аномалий, поиск узких мест

Нужды ETL тестировщиков

Валидация процесса end-to-end: список источников, схема целевой таблицы.

Demo


test: smart_data
Не защищено
tedi-ui-mr-tedi-1462.tedi-ui.v2.dev2.k8s.tcsbank.ru/dag/smart_data?taskId=tmp0
SURVIVAL | Faceb...
SURVIVAL&CO.專...
Календарь
62 полезных инс...
The Secrets Of St...
The Iterative UDP...
Программирован...
Быстрое веб-при...
Календарь любит...
www.erlang-facto...
Другие закладки
Run
TEDI / smart_data
Nodes
New node
Show all nodes

Выводы

Автоматический lineage возможен

За счет статического анализа можно строить полноценный lineage

Generic решение сложно реализовать

Синтаксисы движков очень различаются, только простые запросы

На рынке почти нет готовых решений

Atlan и dbt ближайшие соседи TEDI, но у каждого свой путь

Планируем добавить мету из БД

К статическому анализу добавим данные о типах полей, существовании таблиц и тд

ТИНЬКОФФ

Спасибо!

SmartData 2022

Дима Зуев

17.10.2022