



Выпускной проект

Курс "Data Engineer"

Фурман Юрий
yuri.furman@rt.ru,
yuri063@yandex.ru

Технологии
ВОЗМОЖНОСТЕЙ

Содержание

- Задачи и требования
- Исходные данные
- Описание структуры хранилища
- Описание ETL-процессов
- Описание витрины
- Контроль качества данных
- Результаты

Задачи и требования

1. Разработать структуру хранилища данных (DWH)
2. Разработать и автоматизировать процессы извлечения, трансформации и загрузки данных (ETL) из источников в DWH и построение витрин данных
3. Обеспечить контроль качества поступающих в хранилище данных
4. Разработать BI-отчеты (дэшборды)

Задачи и требования

1. Требования к хранилищу данных:

- Гетерогенность источников
- Поддержка историчности
- Гибкость модели данных
- Скорость обновления
- Устойчивость к объёму

2. Требования к ETL-процессам:

- Извлечение данных
- Очистка данных
- Трансформация данных
- Загрузка данных

3. Требования к витрине данных

- Агрегирование информации в определенном временном или тематическом разрезе
- Формирование отчетных данных в виде шаблонизированного документа

4. Требования к качеству данных (метрики качества)

- Полнота / Completeness
- Точность / Accuracy
- Согласованность / Consistency
- Валидность / Validity
- Своевременность / Timeliness
- Целостность / Integrity

Исходные данные

Исходные данные представлены **5-тью источниками** (таблицы в операционных базах) некоторой биллинговой системы:

- **Начисления:** исходные данные источника располагаются в бакете GCS

rt-2021-03-25-16-47-29-sfunu-final-project

- **Платежи:** исходные данные источника располагаются в бакете GCS

rt-2021-03-25-16-47-29-sfunu-final-project

- **Обращения в ТП:** исходные данные источника располагаются в бакете GCS

rt-2021-03-25-16-47-29-sfunu-final-project

- **Потребляемый трафик:** исходные данные источника располагаются в бакете

GCS rt-2021-03-25-16-47-29-sfunu-final-project

- **MDM:** исходные данные источника MDM располагаются в схеме mdm в Greenplum

Исходные данные

Источник начисления:

Имя поля	Тип поля	Описание	Пример
user_id	bigint	Идентификатор пользователя	1500023
billing_period	varchar	Период оплаты	2020-12
service	varchar	Услуга	Домашний интернет
tariff	varchar	Тариф	Выгодный 500
sum	float	Сумма начислений	110.0
created_at	datetime	Дата начисления	2021-01-10 14:52:12

Исходные данные

Источник платежа:

Имя поля	Тип поля	Описание	Пример
user_id	bigint	Идентификатор пользователя	1500023
pay_doc_type	varchar	Тип платежного документа	Кредитная карта Visa (на сайте)
pay_doc_num	bigint	Номер платежа (уникален в рамках каждого pay_doc_type)	3485900052
account	varchar	Лицевой счет клиента	ФЛ-18709262
phone	varchar	Телефонный номер клиента	79234567890
billing_period	varchar	Период оплаты	2020-12
pay_date	varchar	Дата оплаты	2021-01-10 14:52:12
sum	float	Сумма платежа в рублях	101.50

Исходные данные

Источник обращения:

Имя поля	Тип поля	Описание	Пример
user_id	number	Идентификатор пользователя	1500023
start_time	datetime	Дата открытия обращения	2021-01-10 14:52:12
end_time	datetime	Дата закрытия обращения	null
title	varchar	Тема обращения	Нет интернета
description	text	Описание	Нет интернета два дня, роутер перезагружал
service	varchar	Услуга	Домашний интернет

Исходные данные

Источник трафик:

Имя поля	Тип поля	Описание	Пример
user_id	bigint	Идентификатор пользователя	1500023
timestamp	bigint	Время регистрации события в миллисекундах	1581177460000
device_id	varchar	Серийный номер пользовательского устройства	AN96763S43
device_ip_addr	varchar	IP-адрес пользовательского устройства	172.16.3.82
bytes_sent	bigint	Объем исходящего трафика	0
bytes_received	bigint	Объем входящего трафика	67330

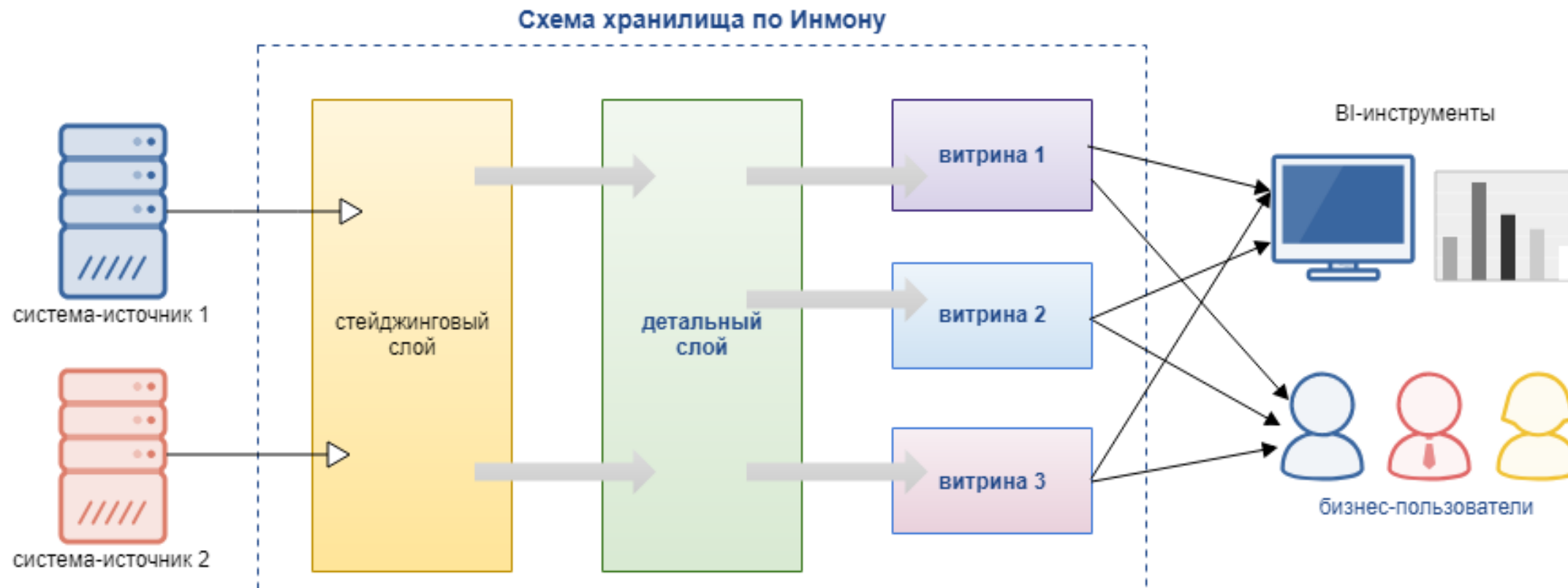
Исходные данные

Источник ***MDM:***

Имя поля	Тип поля	Описание	Пример
id	int	Идентификатор пользователя	10234
legal_type	text	Тип пользователя	Физическое лицо, Юридическое лицо
registered_at	timestamp	Дата регистрации	2012-10-02 12:20:15
billing_mode	text	Режим выставления счета абоненту	Предоплатный, постоплатный
is_vip	bool	Признак уровня обслуживания (VIP)	True/False

Описание структуры хранилища

При проектировании структуры DWH за основу была принята 3-х уровневая модель по Инмону



Описание структуры хранилища

Организация слоев DWH



Описание структуры хранилища

Структура stage слоя (STG)

STG_PAYMENT		
NAME	TYPE	COMMENT
user_id	int	идентификатор пользователя
pay_doc_type	varchar	Тип платежного документа
pay_doc_num	int	Номер платежного документа
account	varchar	Лицевой счет клиента
phone	varchar	Телефонный номер клиента
billing_period	varchar	Период (год, месяц) оплаты
pay_date	varchar	Дата оплаты
sum	double precision	Сумма платежа в рублях

STG_TRAFFIC		
NAME	TYPE	COMMENT
user_id	int	идентификатор пользователя
timestamp	bigint	время регистрации события в микросек.
device_id	varchar	Серийный номер устройства пользователя
device_ip_addr	varchar	IP адрес устройства пользователя
bytes_sent	int	Объем исходящего трафика
bytes_received	int	Объем входящего трафика

STG_BILLING		
NAME	TYPE	COMMENT
user_id	int	идентификатор пользователя
billing_period	varchar	Период (год, месяц) оплаты
service	varchar	услуга
tariff	varchar	тариф
sum	varchar	Сумма начислений в рублях
created_at	datetime	Дата начисления

STG_ISSUE		
NAME	TYPE	COMMENT
user_id	varchar	идентификатор пользователя
start_time	varchar	Дата открытия запроса
end_time	varchar	Дата закрытия запроса
title	varchar	Тема запроса
description	varchar	Описание
service	varchar	услуга

Описание структуры хранилища

Структура операционного слоя (ODS)

ODS_PAYMENT		
NAME	TYPE	COMMENT
user_id	int	идентификатор пользователя
pay_doc_type	varchar	Тип платежного документа
pay_doc_num	bigint	Номер платежного документа
account	varchar	Лицевой счет клиента
phone	varchar	Телефонный номер клиента
billing_period	varchar	Период (год, месяц) оплаты
pay_date	datet	Дата оплаты
sum	decimal(10, 2)	Сумма платежа в рублях

ODS_TRAFFIC		
NAME	TYPE	COMMENT
user_id	int	идентификатор пользователя
time_stamp	timestamp	время регистрации события в милисек.
device_num	varchar	Серийный номер устройства пользователя
device_ip_addr	varchar	IP адрес устройства пользователя
bytes_sent	bigint	Объем исходящего трафика
bytes_received	bigint	Объем входящего трафика

ODS_BILLING		
NAME	TYPE	COMMENT
user_id	int	идентификатор пользователя
billing_period	varchar	Период (год, месяц) оплаты
service	varchar	услуга
tariff	varchar	тариф
sum	decimal(10, 2)	Сумма начислений в рублях
created_at	timestamp	Дата начисления

ODS_ISSUE		
NAME	TYPE	COMMENT
user_id	int	идентификатор пользователя
start_time	timestamp	Дата открытия запроса
end_time	timestamp	Дата закрытия запроса
title	varchar	Тема запроса
description	varchar	Описание
service	varchar	услуга

Описание структуры хранилища

Структура детального слоя (DDS)

*При проектировании DDS была применена методология **Data Vault**.*

Data Vault – набор уникально связанных нормализованных таблиц, содержащих **детальные данные**, отслеживающих историю изменений и предназначенных для поддержки одной или нескольких функциональных областей бизнеса. Автор: Дэн Линстедт (Dan E. Linstedt).

Дизайн Data Vault сосредоточен вокруг функциональных областей бизнеса.

- **Хаб (Hub)** хранит сущности.
- **Связь (Link)** обеспечивает транзакционную интеграцию между Хабами (связи между сущностями).
- **Сателлит (Satellite)** предоставляет контекст первичного ключа Хаба (атрибуты, описания).

Описание структуры хранилища

Структура детального слоя (DDS)

Хаб

Хабы (Hub) являются отдельными таблицами, содержащими как минимум уникальный список бизнес ключей.

Атрибуты Хаба включают:

- › Ключ бизнес-сущности из внешней системы
- › Суррогатный ключ
- › Временная отметка даты загрузки
- › Код источника данных

Линк

Связи (Link) представляет отношения или транзакцию между двумя или более компонентами бизнеса (два или более бизнес ключа).

Атрибуты линка включают:

- › Суррогатный ключ (Surrogate Key)
- › Ключи Хабов: от 1-го Хаба до N-го Хаба
- › Временная отметка даты загрузки
- › Код источника данных

Саттелит

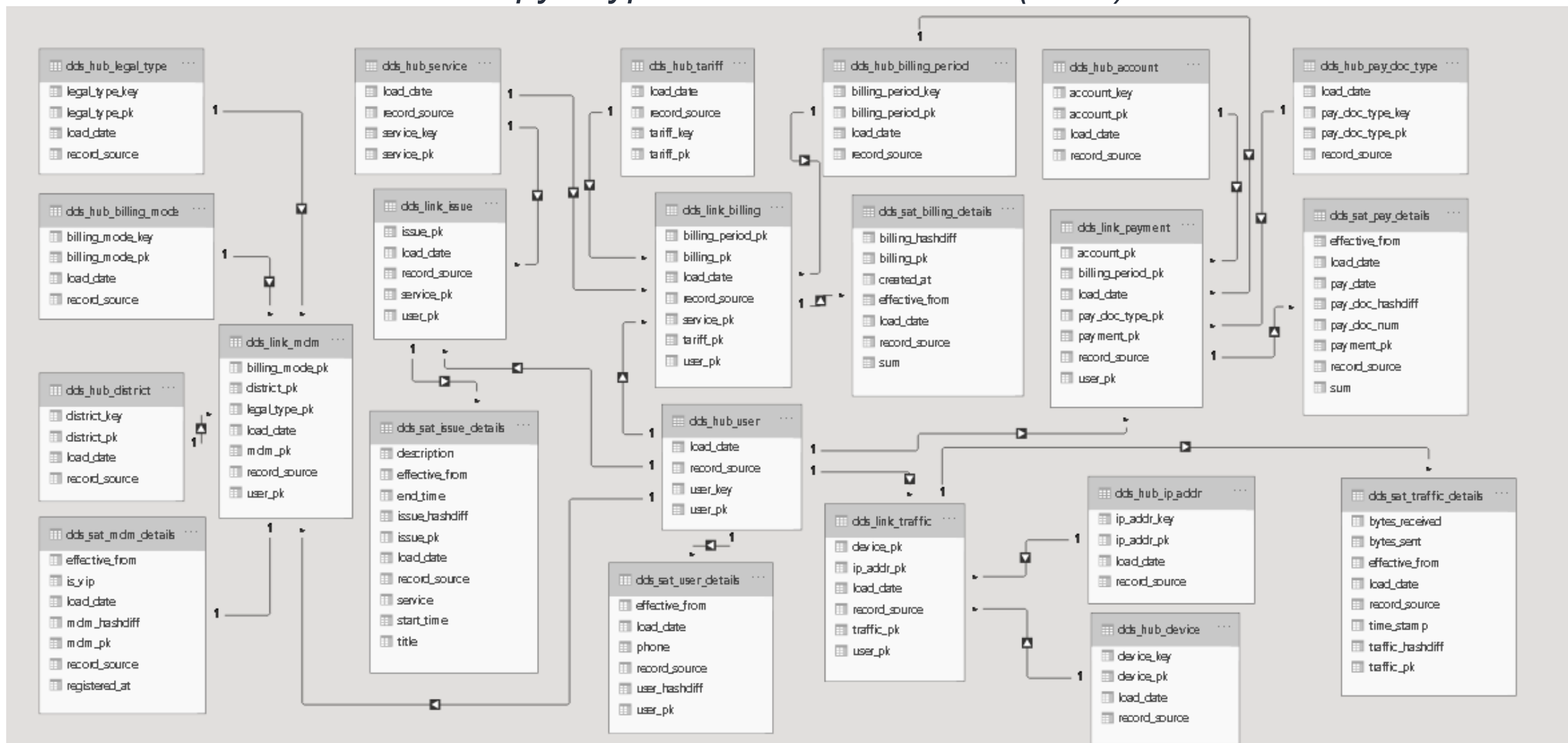
Сателлиты (Satellite) являются контекстной (описательной) информацией ключа Хаба, обычно с историзмом по SCD2.

Атрибуты саттелита включают:

- › Первичный ключ Сателлита: Первичный ключ Хаба или первичный ключ Связи
- › Даты действия записи (SCD2)
- › Временная отметка даты загрузки
- › Код источника данных

Описание структуры хранилища

Структура детального слоя (DDS)



Описание структуры хранилища

Структура слоя витрин данных(DM)

Витрина данных (Data Mart) представляет собой срез DWH в виде массива тематической, узконаправленной информации. Витрина данных, аналогично дэшборд-панели, позволяет аналитику увидеть агрегированную информацию в определенном временном или тематическом разрезе, а также сформировать и распечатать отчетные данные в виде шаблонизированного документа.

Данные для анализа организуются в модель типа **«звезда» (star scheme)**.

Эта модель представляется двумя видами таблиц:

- таблицами **фактов**
- таблицами **измерений**.

Описание структуры хранилища

Структура слоя витрин данных(DM)

Таблица фактов — является основной таблицей DM. Как правило, она содержит сведения об объектах, событиях или процессах, совокупность которых будет в дальнейшем анализироваться.

Характеристики таблиц фактов:

- Таблица фактов содержит числовые параметры (метрики);
- Каждая таблица фактов имеет составной ключ, состоящий из первичных ключей таблиц измерений. Первичный ключ таблицы измерений является внешним ключом в таблице фактов.

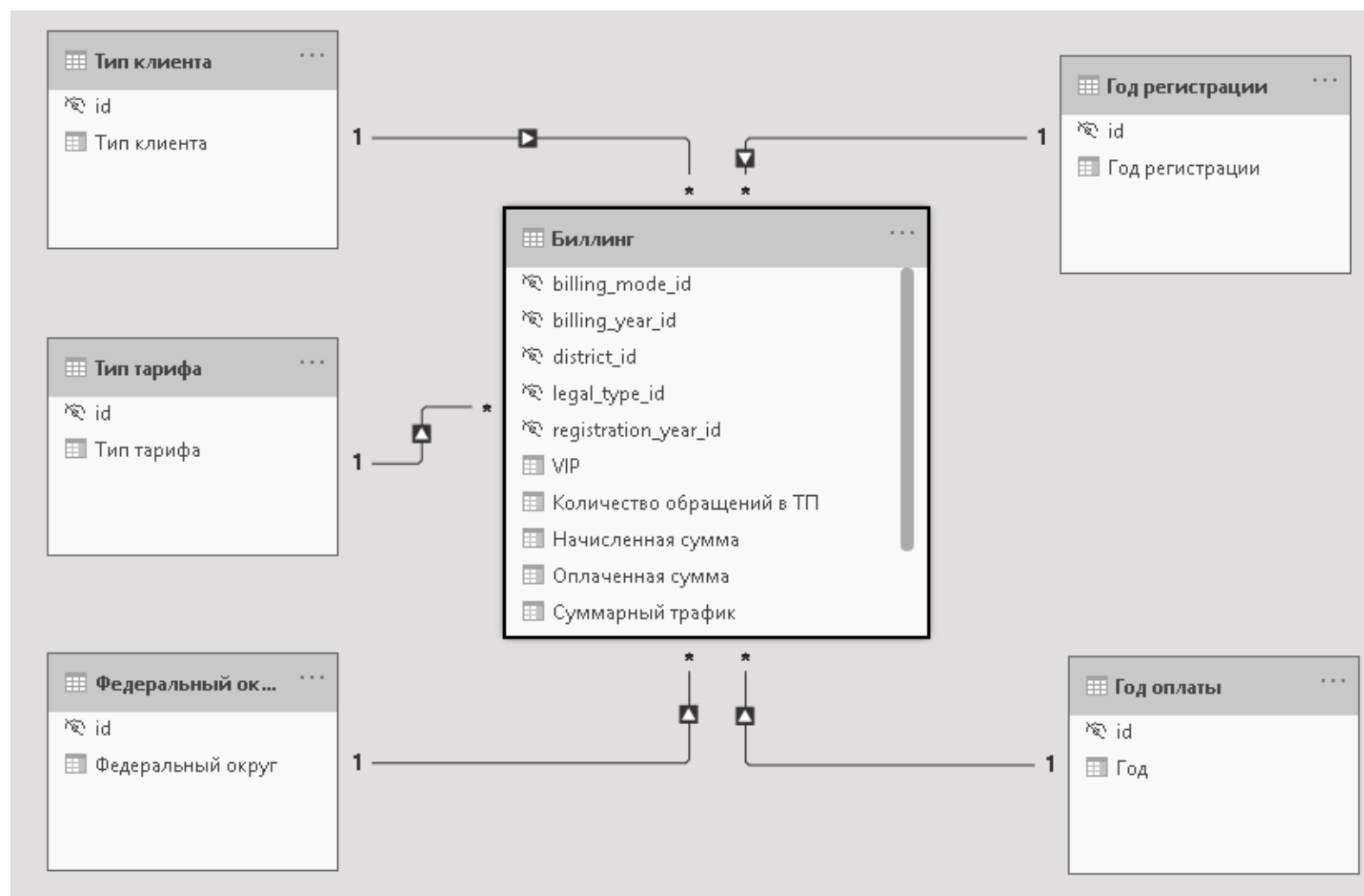
Таблица измерений (англ. dimension table) — содержит атрибуты событий, сохраненных в таблице фактов. Атрибуты представляют собой текстовые или иные описания, логически объединенные в одно целое. Таблица измерения имеет первичный ключ и атрибуты, описывающие факты с точки зрения некоторого направления деятельности организации.

Характеристики измерений:

- Таблицы измерений содержат данные о детализации фактов;
- Таблицы измерений содержат описательную информацию о числовых значениях в таблице фактов, т.е. они содержат атрибуты фактов;
- Атрибуты таблиц измерений обычно используются при визуализации данных в отчетах и запросах;

Описание структуры хранилища

Структура слоя витрин данных(DM)



Описание ETL-процессов

ETL - это совокупность процессов управления хранилищами данных, включая:

- **извлечение данных** из одного или нескольких источников и подготовка их к преобразованию (загрузка в промежуточную область, проверка данных на соответствие спецификациям и возможность последующей загрузки в DWH);
- **трансформация данных** – преобразование форматов и кодировки, агрегация и очистка;
- **загрузка данных** — запись преобразованных данных, включая информацию о структуре их представления (метаданные) в необходимую систему хранения (DWH) или витрину данных.

ETL-процессы по своей архитектуре (способу поступления и обработки данных) делятся на пакетные (*batch processing*) и потоковые (*streaming processing*), либо их сочетание – *lambda архитектура*.

Batch processing - обработка данных, разбитых на непересекающиеся наборы (чаще всего по времени).

Stream processing - обработка данных как непрерывного потока в режиме реального времени.

Описание ETL-процессов

В данной работе мы используем пакетный режим обработки данных.

В пакетном режиме работы ETL-процессы должны обеспечивать *идемпотентность* операций репроцессинга.

Репроцессинг – повторная обработка данных за уже прошедшие периоды.

Идемпотентность — свойство объекта или операции при повторном применении операции к объекту давать тот же результат, что и при первом.

Далее подробно рассмотрим ETL-процессы, примененные для каждого из слоев DWH.

В качестве целевой БД для нашего DWH мы используем MPP-СУБД Greenplum

Описание ETL-процессов

ETL-процессы для STG-слоя

Т.к. наши источники данных уже погружены в **Data Lake** и **партицированы** по времени (по году), то для обеспечения *идемптного репроцессинга* достаточно создать в целевой MPP-базе данных внешнюю таблицу, указывающую на источник. Например, для источника **billing**:

```
create external table yfurman.project_stg_billing (  
    user_id INT,  
    billing_period VARCHAR,  
    service VARCHAR,  
    tariff VARCHAR,  
    sum VARCHAR,  
    created_at VARCHAR)  
location ('pxf://rt-2021-03-25-16-47-29-sfunu-final-project/billing/*/?PROFILE=gs:parquet')  
FORMAT 'CUSTOM' (FORMATTER='pxfwritable_import');
```

На данном этапе мы не производим никаких трансформаций с данными.

Описание ETL-процессов

ETL-процессы для ODS-слоя

Для обеспечения *идемптенного репроцессинга* на данном этапе мы изначально создаем таблицы ODS в Greenplum *партицированными* по времени загрузки данных в системах источниках (интервал – 1 год).

```
create table yfurman.project_ods_billing (  
    user_id INT,  
    billing_period VARCHAR,  
    service VARCHAR,  
    tariff VARCHAR,  
    sum DECIMAL(10,2),  
    created_at TIMESTAMP  
)  
distributed by (user_id)  
partition by range(created_at) (  
    start (TIMESTAMP '1999-01-01') inclusive  
    end   (TIMESTAMP '2040-01-01') exclusive  
    every ('1 year'::interval)  
);
```

На данном этапе мы производим трансформацию типов данных к целевой модели.

Описание ETL-процессов

ETL-процессы для ODS-слоя

При таком подходе, все что нам нужно – это при репроцессинге, перед повторной заливкой, ОЧИСТИТЬ соответствующую партицию от старых данных:

```
alter table if exists yfurman.project_ods_billing truncate partition for('2018-01-01' );
insert into yfurman.project_ods_billing (
    select
        user_id,
        billing_period,
        service,
        tariff,
        sum::DECIMAL(10,2),
        created_at::TIMESTAMP
    from yfurman.project_stg_billing
    where cast(extract('year' from cast(created_at as timestamp)) as int) = 2018
);
```

Описание ETL-процессов

ETL-процессы для DDS-слоя

Перед началом выполнения ETL-процедур для наших сущностей DV-модели (*хабов, линков, сателлитов*), мы должны обогатить наши данные метаданными (указать дату загрузки в DWH и источник данных для каждой записи), рассчитать хеш-значения для ключей. Т.е. мы формируем своеобразный *пред-DDS* слой. Для этого в данной работе были использованы представления (**view**).

Описание ETL-процессов

ETL-процессы для DDS-слоя

Приведем пример выделения метаданных и вычисления хеш-значений для ключей

```
create or replace view yfurman.project_view_billing_one_year as (  
    . . . .  
    'BILLING - DATA LAKE'::varchar as RECORD_SOURCE,  
    cast((md5(nullif(upper(trim(cast(user_id as varchar))), ''))) as TEXT) as USER_PK,  
    cast((md5(nullif(upper(trim(cast(billing_period as varchar))), ''))) as TEXT) as BILLING_PERIOD_PK,  
    cast((md5(nullif(upper(trim(cast(service as varchar))), ''))) as TEXT) as SERVICE_PK,  
    cast((md5(nullif(upper(trim(cast(tariff as varchar))), ''))) as TEXT) as TARIFF_PK,  
    cast(md5(concat_ws('||',  
        coalesce(nullif(upper(trim(cast(user_id as varchar))), ''), '^'),  
        coalesce(nullif(upper(trim(cast(billing_period as varchar))), ''), '^'),  
        coalesce(nullif(upper(trim(cast(service as varchar))), ''), '^'),  
        coalesce(nullif(upper(trim(cast(tariff as varchar))), ''), '^')  
    ), '^'||^'||^'||^') as TEXT) as BILLING_PK,  
    cast(md5(concat_ws('||',  
        coalesce(nullif(upper(trim(cast(created_at as varchar))), ''), '^'),  
        coalesce(nullif(upper(trim(cast(sum as varchar))), ''), '^')  
    )) as TEXT) as BILLING_HASHDIFF,  
    '2018-01-01'::timestamp as LOAD_DATE,  
    created_at as EFFECTIVE_FROM  
);
```

Описание ETL-процессов

ETL-процессы для DDS-слоя. Хабы

Теперь рассмотрим ETL-процесс для каждой из сущностей.

Хабы

Т.к. хабы представляют собой таблицу уникальных бизнес ключей, то для выполнения идемпотентного репроцессинга достаточно обеспечить отсутствие дублирующих записей. При заполнении таблиц хабов данные необходимо брать из всех источников (например, для хаба *billing_period* таких источника два).

Описание ETL-процессов

ETL-процессы для DDS-слоя. Хабы

```
with row_rank_1 as (  
    select * from (select BILLING_PERIOD_PK, BILLING_PERIOD_KEY, LOAD_DATE, RECORD_SOURCE,  
                        row_number() over (partition by BILLING_PERIOD_PK order by LOAD_DATE ASC) as row_num  
                        from yfurman.project_view_payment_one_year  
                    ) as h where row_num = 1  
,  
row_rank_2 as (  
    select * from (select BILLING_PERIOD_PK, BILLING_PERIOD_KEY, LOAD_DATE, RECORD_SOURCE,  
                        row_number() over (partition by BILLING_PERIOD_PK order by LOAD_DATE ASC) as row_num  
                        from yfurman.project_view_billing_one_year  
                    ) as h where row_num = 1  
,  
stage_union as (  
    select BILLING_PERIOD_PK, BILLING_PERIOD_KEY, LOAD_DATE, RECORD_SOURCE from row_rank_1  
    union all  
    select BILLING_PERIOD_PK, BILLING_PERIOD_KEY, LOAD_DATE, RECORD_SOURCE from row_rank_2  
,  
raw_union as (  
    select * from (select BILLING_PERIOD_PK, BILLING_PERIOD_KEY, LOAD_DATE, RECORD_SOURCE,  
                        row_number() over (partition by BILLING_PERIOD_PK order by LOAD_DATE ASC) as row_num  
                        from stage_union where BILLING_PERIOD_PK is not NULL  
                    ) as h where row_num = 1  
,  
records_to_insert as (  
    select a.BILLING_PERIOD_PK, a.BILLING_PERIOD_KEY, a.LOAD_DATE, a.RECORD_SOURCE  
    from raw_union as a  
    left join yfurman.project_dds_hub_billing_period as d  
    on a.BILLING_PERIOD_PK = d.BILLING_PERIOD_PK  
    where d.BILLING_PERIOD_PK is NULL  
)  
insert into yfurman.project_dds_hub_billing_period (BILLING_PERIOD_PK, BILLING_PERIOD_KEY, LOAD_DATE, RECORD_SOURCE)  
(  
    select BILLING_PERIOD_PK, BILLING_PERIOD_KEY, LOAD_DATE, RECORD_SOURCE  
    from records_to_insert  
);
```

Описание ETL-процессов

ETL-процессы для DDS-слоя. Линки

Линки

Линки представляют собой таблицы уникальных наборов хеш-ключей из таблиц хабов. Поэтому, также как и для хабов, в данном случае для выполнения идемпотетного репроцессинга достаточно обеспечить отсутствие дублирующих записей.

Описание ETL-процессов

ETL-процессы для DDS-слоя. Линки

```
with source_data as (  
    select  
        BILLING_PK,  
        USER_PK, BILLING_PERIOD_PK, SERVICE_PK, TARIFF_PK,  
        LOAD_DATE, RECORD_SOURCE  
    from yfurman.project_view_billing_one_year  
)  
,  
records_to_insert as (  
    select distinct  
        stg.BILLING_PK,  
        stg.USER_PK, stg.BILLING_PERIOD_PK, stg.SERVICE_PK, stg.TARIFF_PK,  
        stg.LOAD_DATE, stg.RECORD_SOURCE  
    from source_data as stg  
    left join yfurman.project_dds_link_billing as tgt  
    on stg.BILLING_PK = tgt.BILLING_PK  
    where tgt.BILLING_PK is null  
)  
insert into yfurman.project_dds_link_billing (  
    BILLING_PK,  
    USER_PK, BILLING_PERIOD_PK, SERVICE_PK, TARIFF_PK,  
    LOAD_DATE, RECORD_SOURCE)  
(  
    select  
        BILLING_PK,  
        USER_PK, BILLING_PERIOD_PK, SERVICE_PK, TARIFF_PK,  
        LOAD_DATE, RECORD_SOURCE  
    from records_to_insert  
);
```

Описание ETL-процессов

ETL-процессы для DDS-слоя. Сателлиты

Сателлиты

Таблицы сателлитов, в отличие от хабов и линков, обязаны поддерживать историчность хранимых атрибутов для сущностей (хабов) или транзакций (линков). Прежде чем перейти к описанию ETL-процесса для сателлитов, рассмотрим для них условия обеспечения идемпотентности операций репроцессинга.

Описание ETL-процессов

ETL-процессы для DDS-слоя. Сателлиты

Предположим, что у нас имеются входные данные, состоящие из множества наборов записей, сегментированных по времени (некоторому атрибуту, *effective_from*). Назовем это множество $\{ODS(i)\} = \{ODS(1), ODS(2), \dots, ODS(N-1), ODS(N)\}$. Данное множество должно удовлетворять следующим критериям:

1. каждый элемент $ODS(i)$ – набор записей $\{z(i, k)\}$, где k от 1 до $M(i)$ ($M(i)$ – количество записей в $ODS(i)$)
2. для любых i и j (от 1 до N) выполняется условие: если $i < j$, тогда для любых k (от 1 до $M(i)$) и p (от 1 до $M(j)$) справедливо:

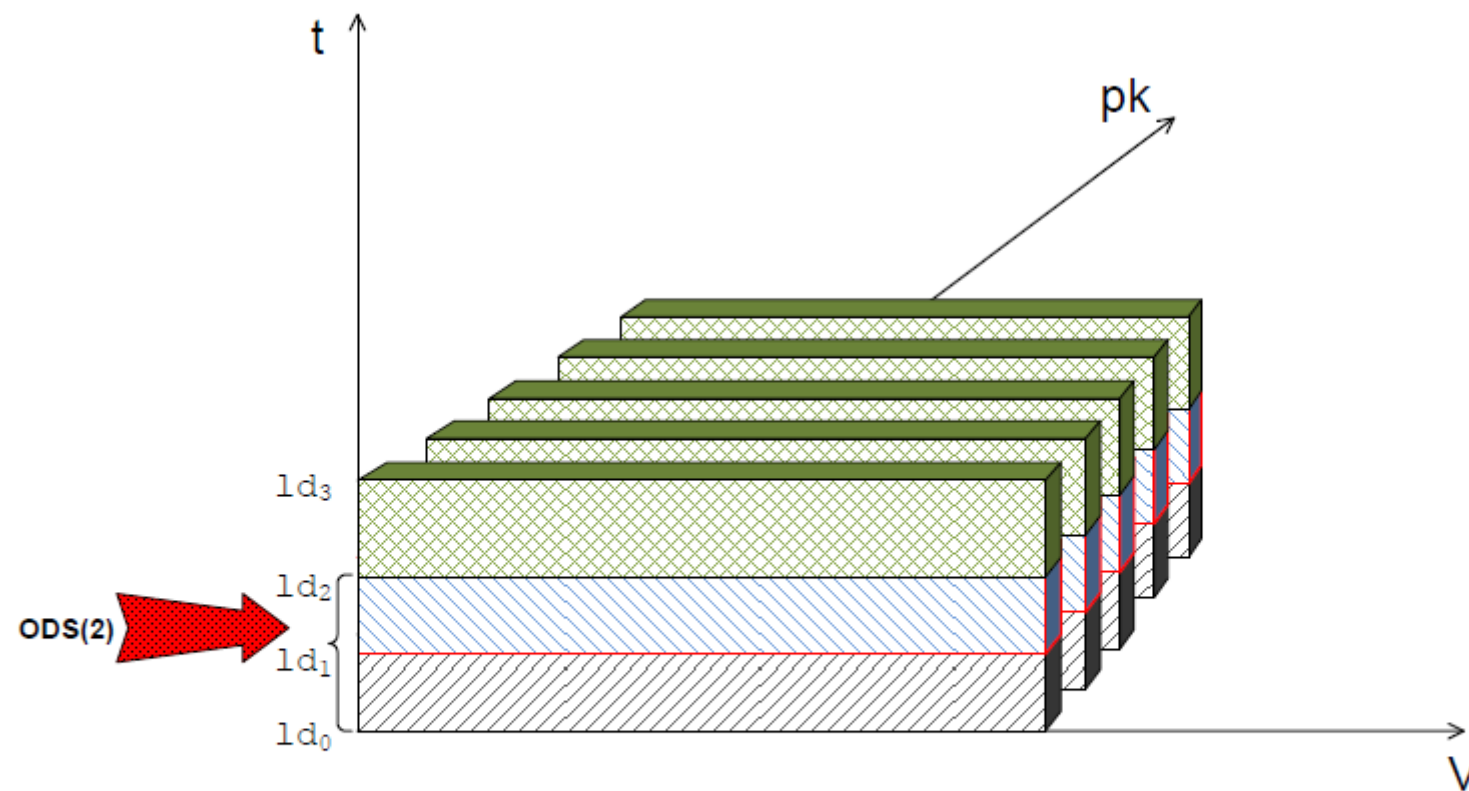
$$z(i, k).effective_from < z(j, p).effective_from$$

Другими словами, наборы записей $ODS(i)$ не пересекаются по временному атрибуту (*effective_from*) для любого i от 1 до N .

Описание ETL-процессов

ETL-процессы для DDS-слоя. Сателлиты

Для фиксированного ключа PK, множество значений атрибутов $\{V\}$ сателлита является функцией времени в n-мерном пространстве (где n число атрибутов сателлита)



$$f(t, pk) = V,$$

$$t \in \{[ld_0, ld_1), [ld_1, ld_2), \dots, [ld_{N-1}, ld_N)\}$$

Описание ETL-процессов

ETL-процессы для DDS-слоя. Сателлиты

Лемма 1. Операция репроцессинга для сателлитов (в терминах *Data Vault*), будет удовлетворять требованию *идемпотентности* тогда и только тогда, когда:

1. Для каждого i от 1 до N набору $ODS(i)$ сопоставлен атрибут $load_date(i)$, такой , что:

- $load_date(i) \geq \max\{z(i, k_1).effective_from\}, k_1 = 1, M(i)$

и для всех $i < N$

$load_date(i) \leq \min\{z(i+1, k_2).effective_from\}, k_2 = 1, M(i+1)$

- для любых i и j (от 1 до N) выполняется условие: если $i < j$, то

$load_date(i) < load_date(j)$

2. Каждая запись из $ODS(i) - z(i, k)$ будет проверяться на совпадение (по первичному ключу и *hashdiff*) только с такими записями $S(j)$ из множества существующих записей в сателлите $\{S\}$, что

$z(i, k).load_date \geq s(j).load_date, k = 1, M(i)$

Описание ETL-процессов

ETL-процессы для DDS-слоя. Сателлиты

3. Если множество записей $\{S(i, k)\}$, сформированное на шаге 2 (*records_to_insert*) не пустое, то повторить шаг 2 для каждой ODS(*m*), где *m* принимает значение от ***i+1*** до ***N***. Иначе переходим на шаг 5.
4. После вставки удалить в сателлите дубли записей по составному ключу (**hub_pk, hash_diff, effective_from**)
5. Операция репроцессинга завершена.

*Смысл шага 3 – означает, что если мы смогли добавить в сателлит новые записи, то нужно перезалить и все данные для всех **load_date(j) > load_date(i)***

Теперь можем описать пункты 1-2 в виде SQL-запроса

Описание ETL-процессов

ETL-процессы для DDS-слоя. Сателлиты

Запишем это в виде sql-запроса:

```
with source_data as (
    select
        HUB_PK, HUB_HASHDIFF,
        attribute_1,..., attribute_N,
        EFFECTIVE_FROM,
        LOAD_DATE, RECORD_SOURCE
    from ods_on_load_date
),
update_records as (
    select
        a.HUB_PK, a.HUB_HASHDIFF,
        a.attribute_1,..., a.attribute_N,
        a.EFFECTIVE_FROM,
        a.LOAD_DATE, a.RECORD_SOURCE
    from dds_sat_hub_details as a
    join source_data as b
    on a.HUB_PK = b.HUB_PK
    where a.LOAD_DATE <= b.LOAD_DATE
),
latest_records as (
    select * from (
        select HUB_PK, HUB_HASHDIFF, LOAD_DATE,
               case when rank() over (partition by HUB_PK
                                     order by LOAD_DATE desc) = 1
                   then 'Y'
                   else 'N'
               end as latest
        from update_records
    ) as s
    where latest = 'Y'
),
records_to_insert as (
    select distinct
        e.HUB_PK, e.HUB_HASHDIFF,
        e.attribute_1,..., e.attribute_N,
        e.EFFECTIVE_FROM,
        e.LOAD_DATE, e.RECORD_SOURCE
    from source_data as e
    left join latest_records
    on latest_records.HUB_HASHDIFF = e.HUB_HASHDIFF and
       latest_records.HUB_PK = e.HUB_PK
    where latest_records.HUB_HASHDIFF is NULL
)
insert into dds_sat_hub_details (
    HUB_PK, HUB_HASHDIFF,
    attribute_1,..., attribute_N,
    EFFECTIVE_FROM,
    LOAD_DATE, RECORD_SOURCE)
(
    select
        HUB_PK, HUB_HASHDIFF,
        attribute_1,..., attribute_N,
        EFFECTIVE_FROM,
        LOAD_DATE, RECORD_SOURCE
    from records_to_insert
);
```

Описание ETL-процессов

ETL-процессы для DDS-слоя. Особенности модели темпоральных данных

Существует два подхода к моделированию темпоральных данных:

Подход к моделированию темпоральных данных, основанный на фиксации событий предметной области, состоит в добавлении временной метки фиксации события (факта) как атрибута экземпляра сущности предметной области и отражении момента времени в таблице БД как истории жизни данных предметной области.

Подход к моделированию темпоральных данных, основанный на фиксации состояний предметной области, состоит в добавлении временных меток для фиксации начала и завершения определенного состояния как атрибутов экземпляра сущности предметной области экземпляров сущности и отражении моментов времени начала и завершения определенного состояния сущности в таблице БД как истории жизни данных предметной области.

Описание ETL-процессов

ETL-процессы для DDS-слоя. Особенности модели темпоральных данных

Пример модели событий: множество значений атрибутов является функцией событий от времени t:

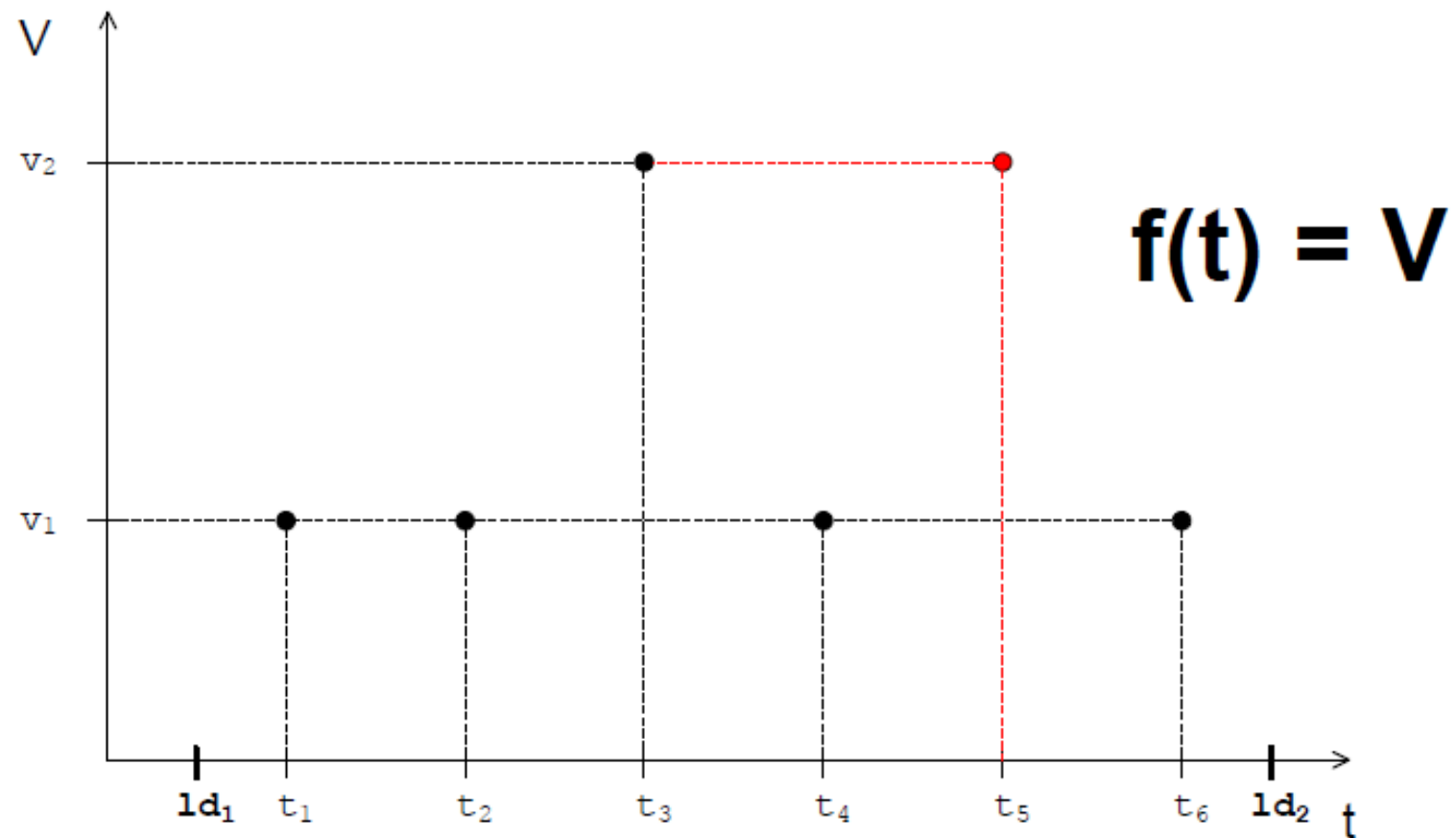
$V = f(t)$

	123 user_id ↑↓	ABC pay_doc_type ↑↓	123 pay_doc_num ↑↓	ABC account ↑↓	ABC billing_period ↑↓	ABC phone ↑↓	🕒 pay_date ↑↓	123 sum ↑↓
7	10 170	MASTER	291 195 617	FL-1787196450	2014-03	+79250341610	2014-04-25	8 039
8	10 170	VISA	264 312 040	FL-1787196450	2014-11	+79250341610	2014-12-10	4 935
9	10 170	MASTER	1 960 386 260	FL-1787196450	2014-03	+79250341610	2014-05-06	7 327
10	10 170	MASTER	706 481 908	FL-1787196450	2014-09	+79250341610	2014-10-28	5 590
11	10 180	MIR	1 680 757 924	FL-2136710735	2014-07	+79324206266	2014-08-27	9 124
12	10 180	MIR	1 512 795 058	FL-2136710735	2014-05	+79324206266	2014-06-24	2 468
13	10 210	MIR	1 828 999 780	FL-616540745	2014-05	+79022472406	2014-06-17	2 002
14	10 230	MIR	491 305 220	FL-645659412	2014-05	+79727768077	2014-06-24	4 362
15	10 230	MASTER	1 150 297 087	FL-645659412	2014-09	+79727768077	2014-10-24	3 812
16	10 250	MIR	1 317 113 798	FL-307197932	2014-05	+79660378230	2014-06-28	1 861
17	10 290	MASTER	1 846 209 445	FL-1933425293	2014-05	+79169958769	2014-06-18	4 723
18	10 300	MASTER	1 840 164 550	FL-1075865221	2014-04	+79138982743	2014-05-17	6 727
19	10 310	MASTER	1 194 530 527	FL-656077164	2014-09	+79820958565	2014-11-09	3 355
20	10 340	VISA	2 103 998 277	FL-1261216314	2014-02	+79814244593	2014-03-13	51
21	10 350	MASTER	983 803 733	FL-1457665697	2014-11	+79157983892	2014-12-27	3 074
22	10 360	VISA	1 035 728 209	FL-1849317627	2014-04	+79466046540	2014-05-31	5 487
23	10 360	MIR	1 147 511 689	FL-1849317627	2014-01	+79466046540	2014-03-07	6 936
24	10 360	MIR	1 889 824 377	FL-1849317627	2014-07	+79466046540	2014-09-01	6 751
25	10 360	MASTER	1 794 657 678	FL-1849317627	2014-07	+79466046540	2014-08-21	3 422
26	10 360	MASTER	1 315 321 880	FL-1849317627	2014-08	+79466046540	2014-09-23	3 020

Описание ETL-процессов

ETL-процессы для DDS-слоя. Особенности модели темпоральных данных

Функция событий



Описание ETL-процессов

ETL-процессы для DDS-слоя. Особенности модели темпоральных данных

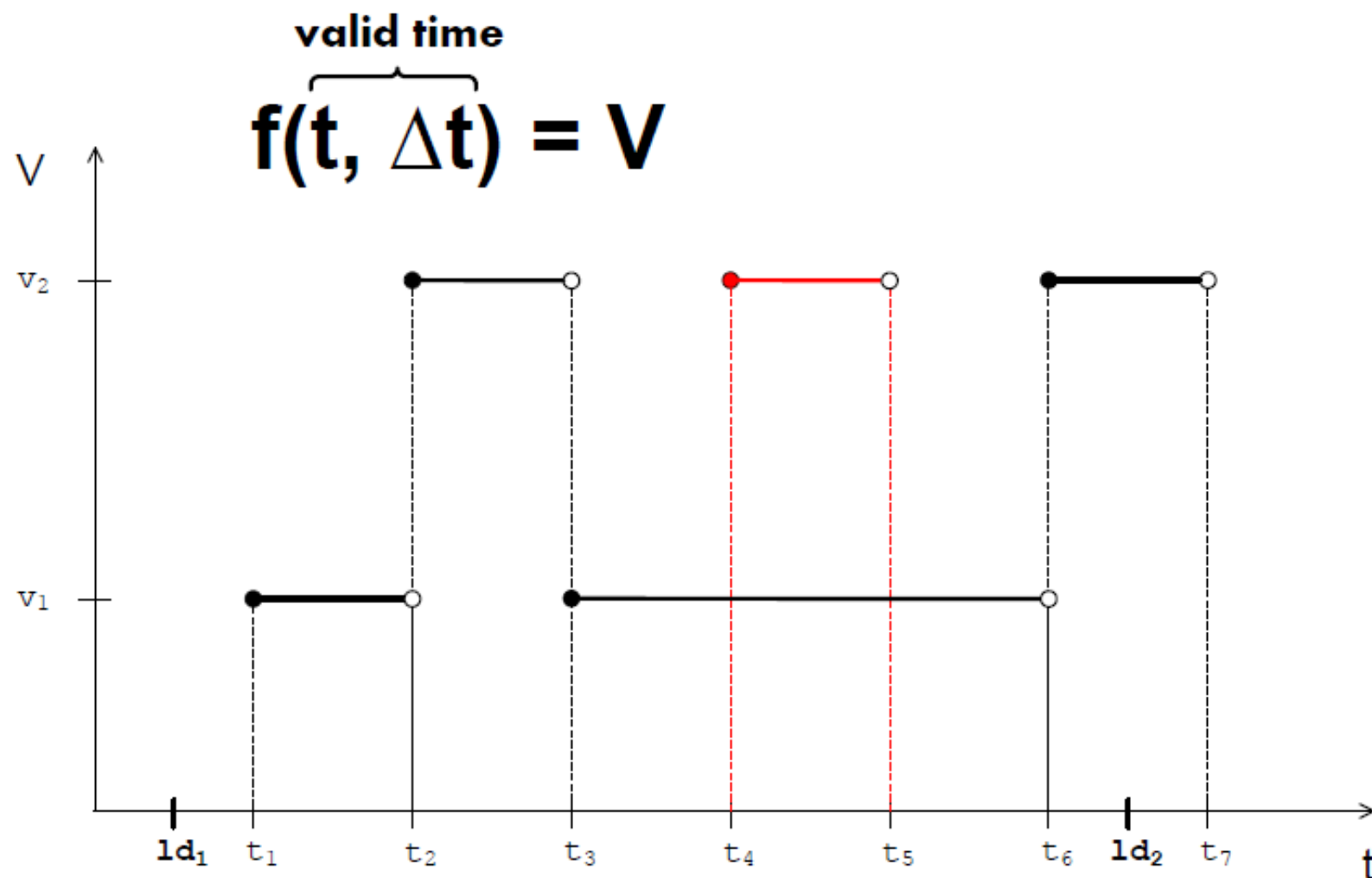
Пример модели состояний: множество значений атрибутов является функцией состояний на интервале от t до $t+\Delta t$: $V = f(t, \Delta t)$

	123 user_id	start_time	end_time	abc title	abc description	abc service
5	11 020	2013-03-29 12:24:36	2013-04-01 01:24:36	gZGhWuWZbdbeBi	VXDkkqxDSUrQRR	Цифровое ТВ
6	11 020	2013-01-09 08:47:11	2013-01-09 17:47:11	SYdRsnqOPSVYFv	pETaMRVLozeeP	Домашний интернет
7	11 020	2013-05-28 22:05:56	2013-05-29 11:05:56	AbrNNFQROFeGeM	dYctHZtwMJdQBc	Домашний интернет
8	11 020	2013-04-30 07:34:31	2013-05-02 07:34:31	SiXPwlrhOLkON	LJBBOKJcpnzFOd	Цифровое ТВ
9	11 020	2013-10-28 15:19:21	2013-10-30 02:19:21	PdkzamQgFjCTqx	LUnvCIfelJylki	Домашний интернет
10	10 990	2013-02-18 01:00:36	2013-02-18 04:00:36	RGiOJvKaqushtF	acTvRUEaqZIAZI	Домашний интернет
11	10 990	2013-06-18 11:27:16	2013-06-21 06:27:16	tcgezFVuthbIaU	cflkQwBduRDxQU	Цифровое ТВ
12	10 990	2013-05-18 15:49:43	2013-05-19 06:49:43	mRnPtYkoaGQytj	bYvvfvCfKVNGBV	Домашний интернет
13	10 980	2013-10-01 09:53:24	2013-10-02 20:53:24	trNnxrlEydfAsW	FWeFBagSfiwCmn	Домашний интернет
14	10 980	2013-02-16 21:59:28	2013-02-18 14:59:28	uMAWLoENlpSVUj	OstLlyXuRUojKS	Цифровое ТВ
15	10 970	2013-10-31 12:03:33	2013-11-01 16:03:33	pDKtlnZUyqZGXf	CcFHjxVDJqNilf	Домашний интернет
16	10 970	2013-07-13 16:28:09	2013-07-14 11:28:09	QdZwVZOIQTyqPB	KOcsYFedhoKnPq	Цифровое ТВ
17	10 970	2013-09-21 11:35:26	2013-09-24 02:35:26	QVeRFhzNcUVgXu	LhsJvwwlEsxbwm	Домашний интернет
18	10 970	2013-09-11 10:48:08	2013-09-14 02:48:08	EHbmYgNKicUACe	sPBgkYbhWWILXc	Цифровое ТВ
19	10 970	2013-06-01 11:28:57	2013-06-04 07:28:57	KvweAtqeDdhRsf	QWJUcTisCWmvno	Цифровое ТВ
20	10 970	2013-07-15 03:01:32	2013-07-16 06:01:32	QjqavPoqaIvBPW	XxVsrjZfUrbDaa	Цифровое ТВ
21	10 970	2013-08-25 01:59:42	2013-08-26 17:59:42	qXihxhmkzjKgE	bUiigyfkzFqnJh	Домашний интернет

Описание ETL-процессов

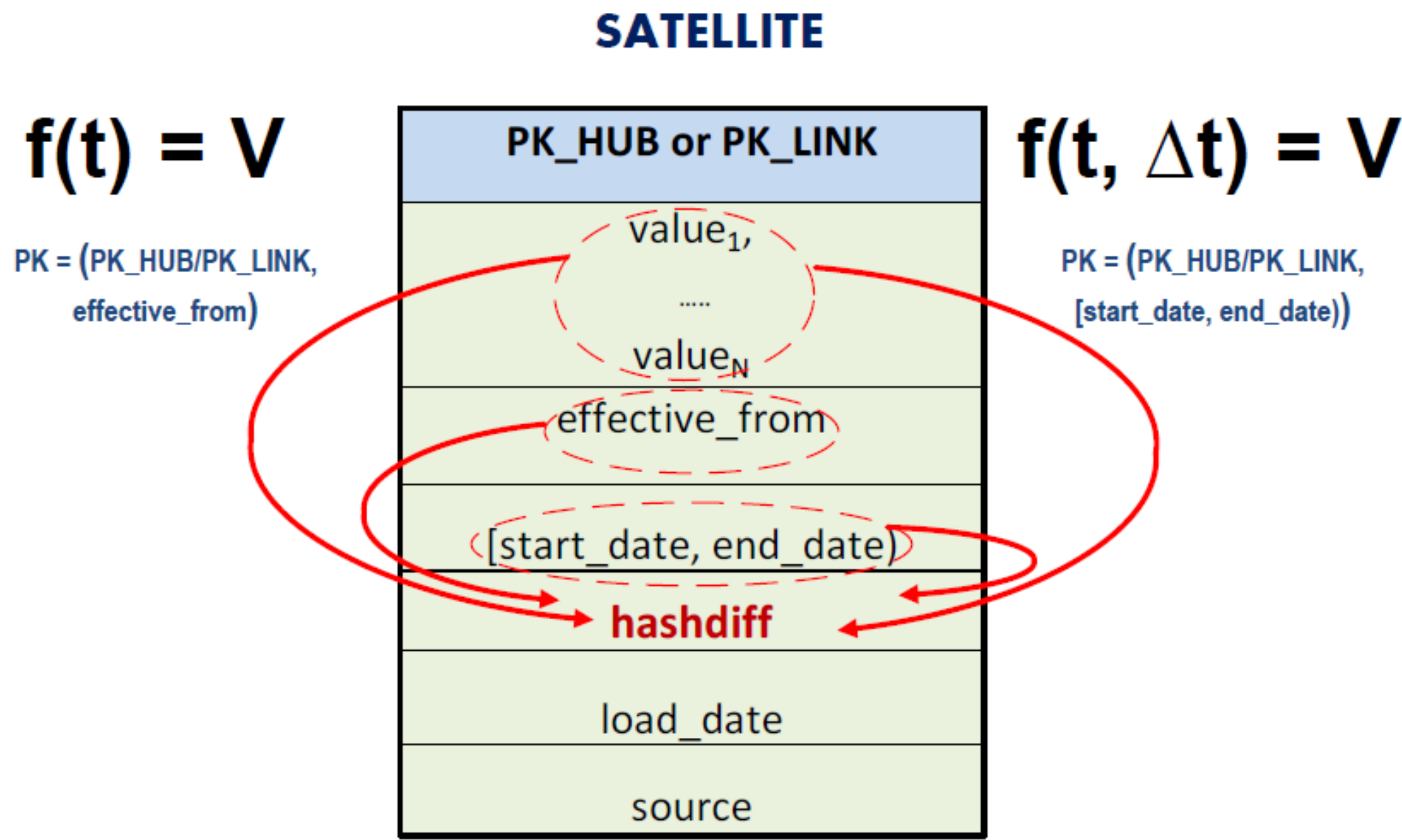
ETL-процессы для DDS-слоя. Особенности модели темпоральных данных

Функция состояний



Описание ETL-процессов

ETL-процессы для DDS-слоя. Особенности модели темпоральных данных



Описание ETL-процессов

ETL-процессы для DDS-слоя. Особенности модели темпоральных данных

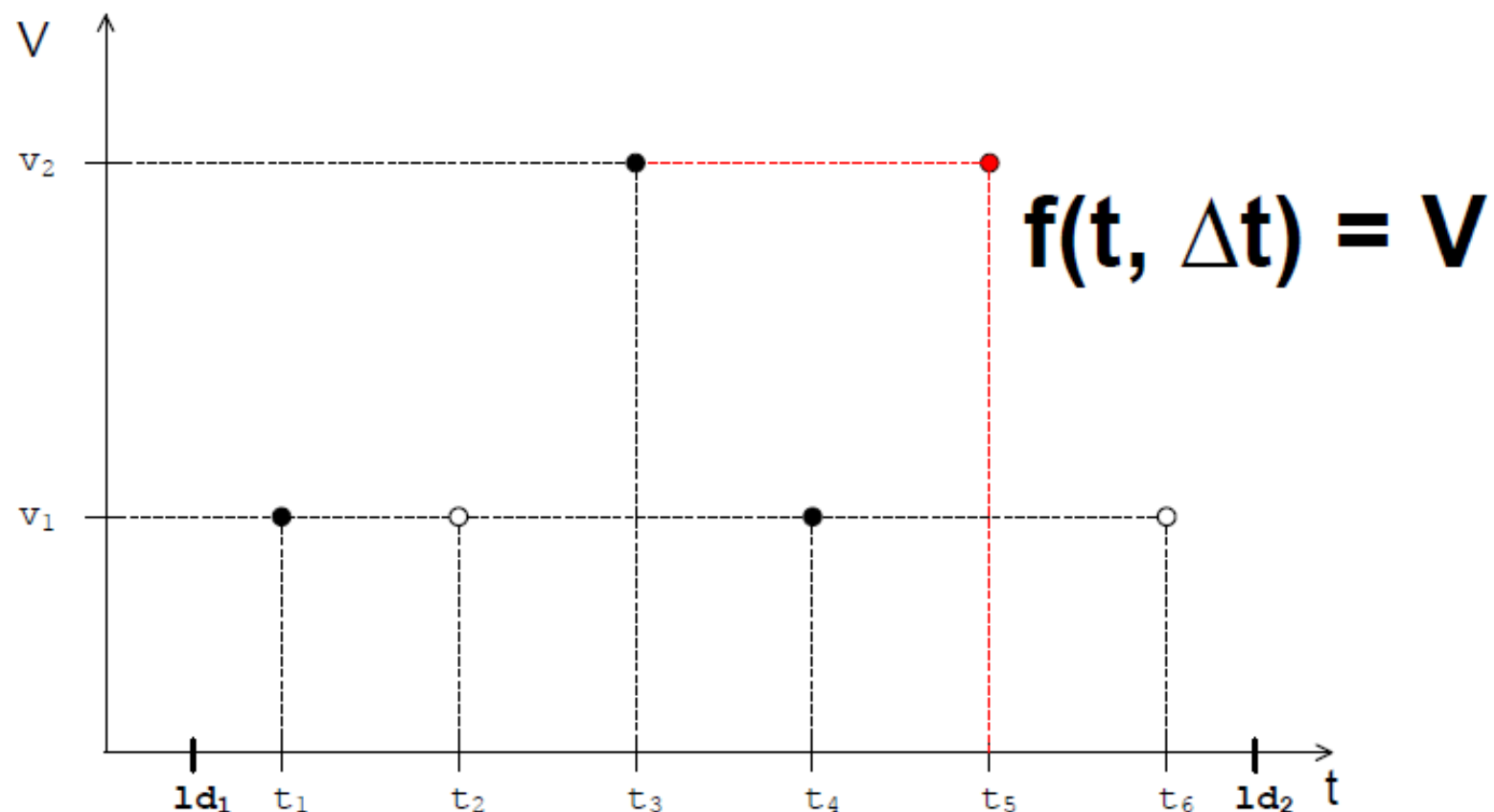
Пример модели состояний при неизвестном Δt : $V = f(t, \Delta t)$

	123 user_id ↑↓	ABC pay_doc_type ↑↓	123 pay_doc_num ↑↓	ABC account ↑↓	ABC billing_period ↑↓	ABC phone ↑↓	🕒 pay_date ↑↓	123 sum ↑↓
7	10 170	MASTER	291 195 617	FL-1787196450	2014-03	+79250341610	2014-04-25	8 039
8	10 170	VISA	264 312 040	FL-1787196450	2014-11	+79250341610	2014-12-10	4 935
9	10 170	MASTER	1 960 386 260	FL-1787196450	2014-03	+79250341610	2014-05-06	7 327
10	10 170	MASTER	706 481 908	FL-1787196450	2014-09	+79250341610	2014-10-28	5 590
11	10 180	MIR	1 680 757 924	FL-2136710735	2014-07	+79324206266	2014-08-27	9 124
12	10 180	MIR	1 512 795 058	FL-2136710735	2014-05	+79324206266	2014-06-24	2 468
13	10 210	MIR	1 828 999 780	FL-616540745	2014-05	+79022472406	2014-06-17	2 002
14	10 230	MIR	491 305 220	FL-645659412	2014-05	+79727768077	2014-06-24	4 362
15	10 230	MASTER	1 150 297 087	FL-645659412	2014-09	+79727768077	2014-10-24	3 812
16	10 250	MIR	1 317 113 798	FL-307197932	2014-05	+79660378230	2014-06-28	1 861
17	10 290	MASTER	1 846 209 445	FL-1933425293	2014-05	+79169958769	2014-06-18	4 723
18	10 300	MASTER	1 840 164 550	FL-1075865221	2014-04	+79138982743	2014-05-17	6 727
19	10 310	MASTER	1 194 530 527	FL-656077164	2014-09	+79820958565	2014-11-09	3 355
20	10 340	VISA	2 103 998 277	FL-1261216314	2014-02	+79814244593	2014-03-13	51
21	10 350	MASTER	983 803 733	FL-1457665697	2014-11	+79157983892	2014-12-27	3 074
22	10 360	VISA	1 035 728 209	FL-1849317627	2014-04	+79466046540	2014-05-31	5 487
23	10 360	MIR	1 147 511 689	FL-1849317627	2014-01	+79466046540	2014-03-07	6 936
24	10 360	MIR	1 889 824 377	FL-1849317627	2014-07	+79466046540	2014-09-01	6 751
25	10 360	MASTER	1 794 657 678	FL-1849317627	2014-07	+79466046540	2014-08-21	3 422
26	10 360	MASTER	1 315 321 880	FL-1849317627	2014-08	+79466046540	2014-09-23	3 020

Описание ETL-процессов

ETL-процессы для DDS-слоя. Особенности модели темпоральных данных

Функция состояний, при неизвестном Δt



Аномалия, связанная с темпоральностью данных: если не принять специальных мер, то количество записей в хранилище будет зависеть от порядка заполнения

Описание ETL-процессов

ETL-процессы для DM-слоя

Т.к. витрина данных служит для предоставления агрегированной информации в определенном временном или тематическом разрезе, то мы должны определить по каким основным **бизнес-ключам** будем строить агрегатные функции. В нашем примере – это *user_id* и *billing_period*

Используя их как каркас, мы будем собирать нашу витрину данных. На первом этапе мы построим вспомогательную таблицу, для которой подготовим данные по каждому тематическому разделу - нас интересуют данные в разрезе: **выставленных и оплаченных счетов, суммарного трафика, количества обращений в ТП на каждого пользователя в календарный период (год).**

Важное замечание: в данной реализации репроцессинг для слоя DM не является **идемпотентным без полной очистки витрины** по той причине, что данные в ней являются агрегированными за все периоды, и, следовательно, нельзя выделить отдельные записи за обрабатываемый период. Например, не возможно отделить платеж за 2018 год , сделанный пользователем в 2017 от 2018 или 2019 года.

Описание ETL-процессов

ETL-процессы для DM-слоя

Рассмотрим отдельно вопрос обеспечения **идемпотентности репроцессинга** для слоя витрин данных (DM).

Таблицу итоговой витрины (таблицу фактов), как и промежуточные таблицы фактов, для каждой «порции» данных из источников (за любой интервал времени), можно представить как матрицы, имеющие следующий вид:

A_1	A_2	...	A_N	F_1	F_2		F_M
$a_{1,1}$	$a_{2,1}$...	$a_{N,1}$	$f_{1,1}$	$f_{2,1}$		$f_{M,1}$
$a_{1,2}$	$a_{2,2}$...	$a_{N,2}$
...
$a_{1,P(A1)}$...	$a_{N,P(AN)}$

Где A_1, A_2, \dots, A_N - измерения, а F_1, F_2, \dots, F_M - факты.

Описание ETL-процессов

ETL-процессы для DM-слоя

Очевидно, что наборов значений $(a_{1,j}, a_{2,j}, \dots, a_{N,j})$ будет столько, сколько всего существует комбинаций всех значений измерений, т.е.

$$P = P(A_1) \times P(A_2) \times \dots \times P(A_N), \quad \text{где } P(A_i) \text{ – число значений измерения } A_i, \quad i = 1, N$$

Т.к. каждый такой набор является уникальным, то ему можно сопоставить результат хеш-функции – хеш-ключ:

$$h_j = h(a_{1,j}, a_{2,j}, \dots, a_{N,j}), \quad \text{где } j = 1, P$$

Тогда таблицы фактов можно преобразовать к виду:

Н	F ₁	F ₂		F _М
h ₁	f _{1,1}	f _{2,1}	...	f _{М,1}
h ₁
...
h _Р	f _{1,Р}	f _{2,Р}	...	f _{М,Р}

Описание ETL-процессов

ETL-процессы для DM-слоя

A_1	A_2	...	A_i	...	A_N	F_1	F_2	...	F_M
$a_{1,1}$	$a_{2,1}$...	$a_{i,1}$...	$a_{N,1}$	$f_{1,1}$	$f_{2,1}$...	$f_{M,1}$
$a_{1,2}$	$a_{2,2}$...	$a_{i,2}$...	$a_{N,2}$
...
$a_{1,P(A_1)}$...	$a_{i,P(A_i)}$...	$a_{N,P(A_N)}$

Значения измерений, в общем случае могут быть НЕ УНИКАЛЬНЫМИ для каждого из поступивших в хранилище пакетов («порций») данных ODS(i). Следствием этого является, то, что для обеспечения идемпотенности репроцессинга мы должны хранить таблицы фактов для каждого интервала времени.

	123 user_id	ABC pay_doc_type	123 pay_doc_num	ABC account	ABC phone	ABC billing_period	pay_date	123 sum
1	10 810	MASTER	427 820 469	FL-1210809915	+79010168768	2013-12	2014-01-19	8 725
2	11 070	MASTER	1 623 090 989	FL-943794062	+79342178413	2013-12	2014-01-06	1 712
3	10 610	MIR	803 805 944	FL-921488443	+79898918916	2013-12	2014-01-19	2 504
4	11 050	VISA	1 859 579 573	FL-1267332307	+79672841946	2013-12	2014-01-12	4 638
5	10 700	MASTER	1 013 475 541	FL-470957464	+79978855132	2013-12	2014-01-14	8 862
6	10 360	MIR	1 147 511 689	FL-1849317627	+79466046540	2014-01	2014-03-07	6 936
7	10 860	MIR	1 859 110 066	FL-1679842082	+79145464669	2014-01	2014-02-14	3 792
8	10 700	MASTER	68 880 093	FL-470957464	+79978855132	2014-01	2014-02-23	7 391
9	11 150	MIR	2 062 917 526	FL-1656077958	+79293902808	2014-01	2014-02-19	616
10	10 650	VISA	1 649 212 405	FL-1665093401	+79385725317	2014-01	2014-02-08	1 847

В противном случае, если таблица фактов рассчитана только на полном наборе данных из всех интервалов, то в случае необходимости учета изменений на каком-либо из интервалов, возникает необходимость ее полного пересчета.

Описание ETL-процессов

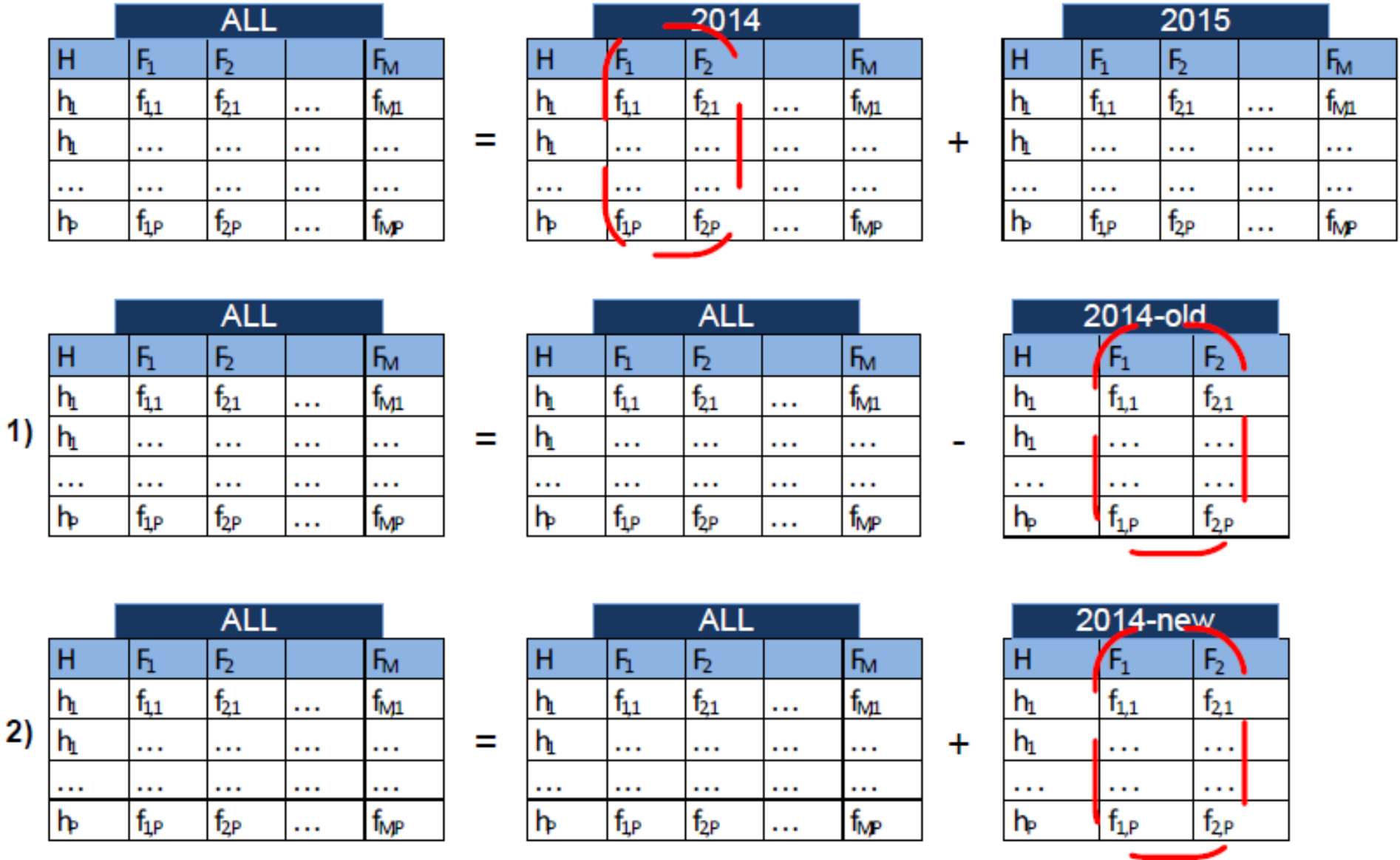
ETL-процессы для DM-слоя

Мы можем обеспечить **идемпотентность репроцессинга** для пакетной обработки данных, поступающих за каждый временной интервал, при соблюдении следующих условий:

1. Все факты F_1, F_2, \dots, F_M - аддитивны (т.е. являются результатом аддитивных агрегатных функций: сложения, вычитания и т.д.)
2. При расчете временной таблицы фактов, в ней будут учитываться данные, полученные из источников, только за тот интервал времени, которому соответствует данный набор. Т.е., применительно к модели DV, значения атрибутов из таблиц *satеллитов* будут выбираться за период, соответствующий определенной LOAD_DATE.
3. Необходимо хранить все промежуточные (временные) витрины за каждый период загрузки данных (один период – одна промежуточная витрина)
4. При выполнении репроцессинга для некоторого интервала времени V , необходимо:
 - a. для каждого (или группы) столбцов фактов из итоговой матрицы (витрины) выполнить обратную аддитивную операцию (для сложения – вычесть, и т.д) с соответствующими столбцами фактов из ранее рассчитанной временной витрины для этого интервала V .
 - b. Пересчитать все или часть столбцов фактов в витрине для интервала V на новых загруженных данных
 - c. Выполнить аддитивную операцию для каждого пересчитанного столбца фактов для интервала V с итоговой матрицей (витриной)

Описание ETL-процессов

ETL-процессы для DM-слоя



Описание ETL-процессов

ETL-процессы для DM-слоя

Подготавливаем данные по тематике:

1) Выставленные пользователям счета:

```
with row_billing as (  
    select * from (  
        select BILLING_PK, sum, EFFECTIVE_FROM, LOAD_DATE, RECORD_SOURCE,  
            row_number() over (  
                partition by BILLING_PK  
                order by EFFECTIVE_FROM DESC  
            ) as row_num  
        from yfurman.project_dds_sat_billing_details  
    ) as h where row_num = 1  
,  
  
billing_sum_data as (  
    select USER_PK, BILLING_PERIOD_PK, sum(sum) as bill_sum  
    from row_billing rb  
    join yfurman.project_dds_link_billing lb on rb.BILLING_PK = lb.BILLING_PK  
    group by USER_PK, BILLING_PERIOD_PK  
)
```

Описание ETL-процессов

ETL-процессы для DM-слоя

Подготавливаем данные по тематике:

2) Оплаты пользователей:

```
payment_sum_data as (  
    select USER_PK, BILLING_PERIOD_PK, sum(sum) as pay_sum  
    from yfurman.project_dds_sat_pay_details spd  
    join yfurman.project_dds_link_payment lp on spd.PAYMENT_PK = lp.PAYMENT_PK  
    group by USER_PK, BILLING_PERIOD_PK  
) ,
```

3) Обращения пользователей в ТП:

```
issue_sum_data as (  
    select  
        USER_PK,  
        to_char(start_time, 'YYYY-MM') as BILLING_PERIOD_KEY,  
        count(*) as issue_count  
    from yfurman.project_dds_sat_issue_details sid  
    join yfurman.project_dds_link_issue li on sid.ISSUE_PK = li.ISSUE_PK  
    group by USER_PK, BILLING_PERIOD_KEY  
) ,
```

Описание ETL-процессов

ETL-процессы для DM-слоя

Подготавливаем данные по тематике:

4) Суммарный трафик пользователей:

```
traffic_sum_data as (  
    select  
        USER_PK,  
        to_char(time_stamp, 'YYYY-MM') as BILLING_PERIOD_KEY,  
        sum(bytes_sent) as traff_out,  
        sum(bytes_received) as traff_in  
    from yfurman.project_dds_sat_traffic_details std  
    join yfurman.project_dds_link_traffic lt on std.TRAFFIC_PK = lt.TRAFFIC_PK  
    group by USER_PK, BILLING_PERIOD_KEY  
) ,
```

5) И, собственно, сам каркас из бизнес-ключей:

```
raw_user_period as (  
    select USER_PK, USER_KEY,  
        BILLING_PERIOD_PK, BILLING_PERIOD_KEY  
    from yfurman.project_dds_hub_user, yfurman.project_dds_hub_billing_period  
) ,
```

Описание ETL-процессов

ETL-процессы для DM-слоя

Теперь можем собрать это все в одну таблицу:

```
raw_data as (  
    select  
        LEGAL_TYPE_KEY as legal_type,  
        DISTRICT_KEY as district,  
        BILLING_MODE_KEY as billing_mode,  
        extract(year from registered_at) as registration_year,  
        is_vip,  
        extract(year from to_date(rup.BILLING_PERIOD_KEY, 'YYYY-MM')) as billing_year,  
        pay_sum,  
        bill_sum,  
        issue_count,  
        traff_out,  
        traff_in  
    from raw_user_period rup  
    left join payment_sum_data psd on rup.USER_PK = psd.USER_PK  
        and rup.BILLING_PERIOD_PK = psd.BILLING_PERIOD_PK  
    left join billing_sum_data bsd on rup.USER_PK = bsd.USER_PK  
        and rup.BILLING_PERIOD_PK = bsd.BILLING_PERIOD_PK  
    left join issue_sum_data isd on rup.USER_PK = isd.USER_PK  
        and rup.BILLING_PERIOD_KEY = isd.BILLING_PERIOD_KEY  
    left join traffic_sum_data tsd on rup.USER_PK = tsd.USER_PK  
        and rup.BILLING_PERIOD_KEY = tsd.BILLING_PERIOD_KEY  
    left join yfurman.project_dds_link_mdm lm on rup.USER_PK = lm.USER_PK  
    left join yfurman.project_dds_hub_legal_type hlt on lm.LEGAL_TYPE_PK = hlt.LEGAL_TYPE_PK  
    left join yfurman.project_dds_hub_district hd on lm.DISTRICT_PK = hd.DISTRICT_PK  
    left join yfurman.project_dds_hub_billing_mode hbm on lm.BILLING_MODE_PK = hbm.BILLING_MODE_PK  
    left join yfurman.project_dds_sat_mdm_details smd on lm.MDM_PK = smd.MDM_PK  
)
```

Описание ETL-процессов

ETL-процессы для DM-слоя

Далее, мы агрегируем данные из полученной объединенной таблицы по остальным интересующим нас бизнес-ключам – *отчетному периоду, типу пользователя, региону, режиму выставления счетов, году регистрации, признаку VIP:*

```
select billing_year, legal_type, district, billing_mode, registration_year, is_vip,  
       sum(pay_sum) as payment_sum,  
       sum(bill_sum) as billing_sum,  
       sum(issue_count) as issue_cnt,  
       sum(traff_out + traff_in) as traffic_amount  
from raw_data  
group by billing_year, legal_type, district, billing_mode, registration_year, is_vip  
order by billing_year, legal_type, district, billing_mode, registration_year, is_vip
```

Эти значения мы вставляем во временную таблицу, на основании которой построим таблицы измерений и фактов.

Описание ETL-процессов

ETL-процессы для DM-слоя

Пример заполнения таблицы для измерений:

```
insert into yfurman.project_report_dim_billing_year(billing_year_key)
select distinct billing_year as billing_year_key
from yfurman.project_report_tmp a
left join yfurman.project_report_dim_billing_year b on b.billing_year_key = a.billing_year
where b.billing_year_key is null;
```

Заполнение таблицы фактов:

```
insert into yfurman.project_report_fct(
    billing_year_id, legal_type_id,
    district_id, registration_year_id,
    billing_mode_id, is_vip,
    payment_sum, billing_sum,
    issue_cnt, traffic_amount
)
select biy.id, lt.id, d.id, ry.id, bm.id, is_vip,
    raw.payment_sum, raw.billing_sum, raw.issue_cnt, raw.traffic_amount
from yfurman.project_report_tmp raw
join yfurman.project_report_dim_billing_year biy on raw.billing_year = biy.billing_year_key
join yfurman.project_report_dim_legal_type lt on raw.legal_type = lt.legal_type_key
join yfurman.project_report_dim_district d on raw.district = d.district_key
join yfurman.project_report_dim_registration_year ry on raw.registration_year = ry.registration_year_key
join yfurman.project_report_dim_billing_mode bm on raw.billing_mode = bm.billing_mode_key;
```

Описание ETL-процессов

Оркестрация

Для управления рабочими ETL процессами необходимо использовать специальные программные средства - *оркестраторы*. В ряде случаев для этого достаточно обычного **cron**. Мы для оркестрации использовали **Apache Airflow**.

Это ПО предоставляет следующий функционал:

- Визуальное представление задач с управлением через GUI
- Единая система для всех задач во всех сервисах на всех серверах
- Обеспечение зависимостей между задачами
- Перезапуск задач со сложной логикой
- Алертинг

Описание ETL-процессов

Оркестрация

Apache Airflow удобен тем, что сценарии запуска задач в нем описываются Python-кодом. Например, в нашем случае логику зависимостей ETL-процессов можно описать простой структурой:

```
FactoryPhase = namedtuple('FactoryPhase', ['name', 'latest_only', 'list_jobs'])
FactoryJob = namedtuple('FactoryJob', ['name', 'source_path', 'mask'])

PHASES = (
    FactoryPhase(
        name='ETL', latest_only = False,
        list_jobs=(
            FactoryJob(name='ODS', source_path='ods', mask='.sql'),
            FactoryJob(name='LOAD_VIEWS', source_path='views', mask='.sql'),
            FactoryJob(name='HUBS', source_path='hubs', mask='.sql'),
            FactoryJob(name='LINKS', source_path='links', mask='.sql'),
            FactoryJob(name='SATELLITES', source_path='satellites', mask='.sql'),
            FactoryJob(name='CLEAR_VIEWS', source_path='clear_views', mask='.sql'),
        )
    ),
    FactoryPhase(
        name='DM', latest_only = True,
        list_jobs=(
            FactoryJob(name='CREATE_TMP_REPORT', source_path='tmp_report', mask='.sql'),
            FactoryJob(name='DIMENSIONS', source_path='dimensions', mask='.sql'),
            FactoryJob(name='FACTS', source_path='facts', mask='.sql'),
            FactoryJob(name='CLEAR_TMP', source_path='clear_tmp', mask='.sql'),
        )
    ),
)
```

Описание ETL-процессов

Оркестрация

Тогда весь ETL процесс можно описать следующим образом:

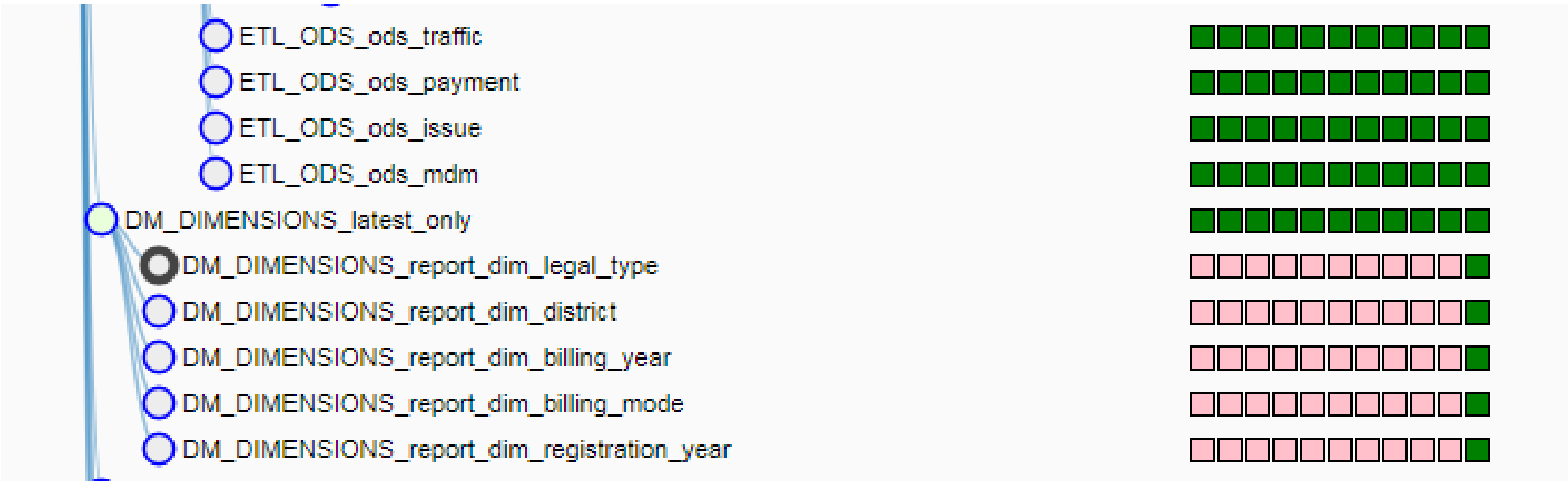
```
def get_job_context(phase_name, job):
    tasks = []
    for task_file_name in [i for i in os.listdir(os.path.join(ROOT_DIR, DATA_DIR, job.source_path)) \
                           if i.endswith(job.mask)]:
        tasks.append(PostgresOperator(
            task_id='{}_{}_{}'.format(phase_name, job.name, os.path.splitext(task_file_name)[0]),
            dag=dag,
            params={'prefix': PREFIX_NAME},
            sql=os.path.join(job.source_path, task_file_name)
        ))
    return tasks

check_point_last = None
for phase in PHASES:
    if phase.latest_only:
        last_only_point = LatestOnlyOperator(task_id="{}_latest_only".format(phase.name), dag=dag)
        if check_point_last:
            check_point_last >> last_only_point
            check_point_last = last_only_point
    for job in phase.list_jobs:
        check_point = DummyOperator(task_id="{}_{}_complete".format(phase.name, job.name), dag=dag)
        job_context = get_job_context(phase.name, job)
        if check_point_last:
            check_point_last >> job_context >> check_point
        else:
            job_context >> check_point
        check_point_last = check_point
```

Описание ETL-процессов

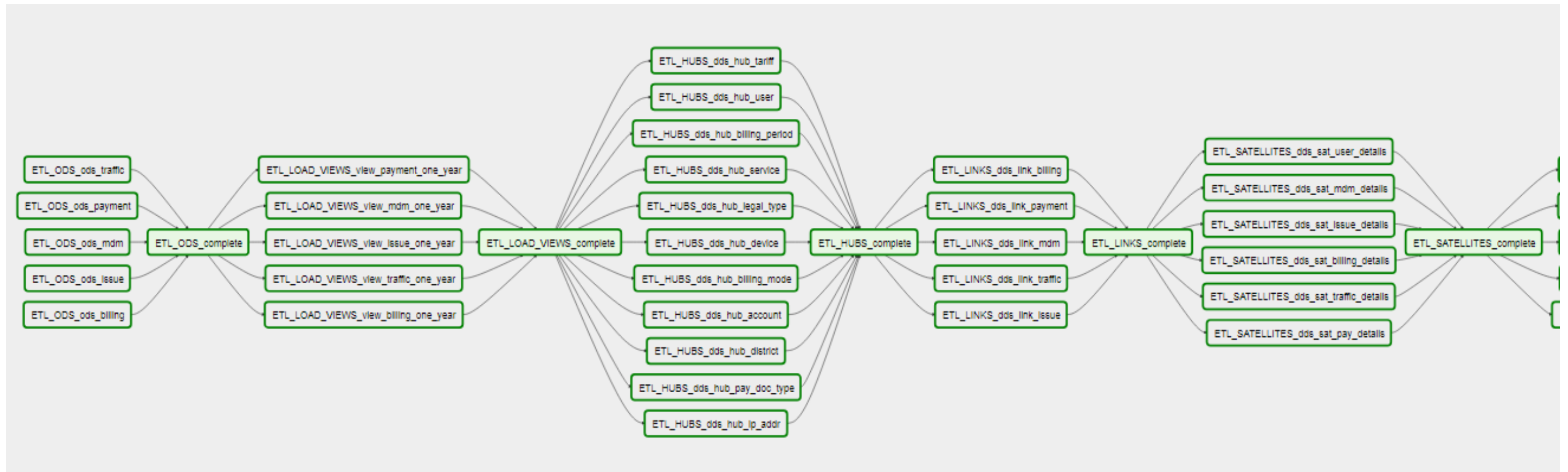
Оркестрация

Обращаем внимание, что задачи для витрин данных выполняются только, если текущий инстанс DAG-airflow является последним возможным в данный момент (т.е. *execution_date* = *end_date*). Это следствие не идемпотентности репроцессинга для DM. Нет смысла каждый раз пересчитывать витрину, если сразу при очередном запуске ее необходимо очистить.



Описание ETL-процессов

Т.к. пайплайны в Apache Airflow представлены в виде направленного ациклического графа (*DAG*), то весь наш ETL процесс будет представлен примерно таким образом (на картинке только *часть графа*)



Описание витрины

Согласно поставленной задаче построенная витрина должна содержать следующие данные:

Название поля	Тип поля	Значение поля
year*	int	Отчетный период (год)
legal_type*	string	Юридическое или физическое лицо
district*	string	Регион обслуживания клиента
billing_mode*	string	Постоплата / предоплата
registration_year*	int	Год подключения пользователя
is_vip*	boolean	Является ли клиент премиальным
payment_sum*	decimal	Сумма платежей за период
billing_sum**	decimal	Сумма начислений за период
issue_cnt**	int	Количество обращений в ТП за период
traffic_amount**	bigint	Объем суммарного (входящего и исходящего) трафика, потребленного за период

Описание витрины

Получим несколько записей из таблицы фактов витрины :

project_report_fct 1										
select * from yfurman.project_report_fct limit 100										
	123 billing_year_id	123 legal_type_id	123 district_id	123 registration_year_id	123 billing_mode_id	is_vip	123 payment_sum	123 billing_sum	123 issue_cnt	123 traffic_amount
17	2	2	1	3	1	[]	123 168	116 642	26	27 543 753 621
18	2	2	2	2	1	[]	23 398	23 398	19	25 818 118 710
19	2	2	3	3	1	[]	30 992	30 992	9	27 782 554 870
20	2	2	3	1	1	[]	98 161	90 170	25	55 139 058 712
21	2	2	3	2	2	[]	51 424	51 424	10	14 147 441 382
22	2	2	5	2	2	[]	79 592	71 992	16	27 588 397 423
23	2	2	7	1	1	[v]	45 985	41 928	12	26 209 226 805
24	2	2	7	3	2	[]	212 532	202 745	60	125 284 353 278
25	2	2	8	2	2	[v]	81 487	76 859	10	21 574 834 008
26	2	1	4	3	2	[]	94 852	87 966	12	22 425 542 380
27	2	1	4	1	2	[]	68 289	61 262	10	32 392 426 832
28	2	1	5	2	2	[]	37 405	23 564	19	24 421 132 746
29	2	1	7	2	1	[]	68 370	68 370	13	29 874 871 493
30	4	2	6	3	1	[]	48 741	43 488	9	24 705 835 354
31	4	2	6	1	2	[v]	199 799	195 524	36	82 179 460 281
32	3	2	6	3	1	[]	77 176	75 608	15	20 005 306 987
33	3	2	2	1	1	[]	113 243	103 684	14	27 172 266 415
34	3	2	3	2	1	[v]	50 687	37 414	19	17 721 471 655
35	3	2	3	1	1	[v]	69 078	63 747	9	23 729 811 149

Описание витрины

и несколько записей из таблиц измерений:

project_report_dim_billing_year1

select * from yfurman.project_report_dim_billing_yea

Таблица	id	billing_year_key
1	1	2012
2	2	2015
3	3	2013
4	4	2016
5	5	2014
6	6	2018
7	7	2017
8	8	2019

project_report_dim_district1

select * from yfurman.project_report_dim_district lim

Таблица	id	district_key
1	1	СЗФО
2	5	УрФО
3	2	СКФО
4	3	СФО
5	4	ДФО
6	6	ПФО
7	7	ЦФО
8	8	ЮФО

project_report_dim_legal_type1

select * from yfurman.project_report_dim_legal_type

Таблица	id	legal_type_key
1	1	Юрлицо
2	2	Физлицо

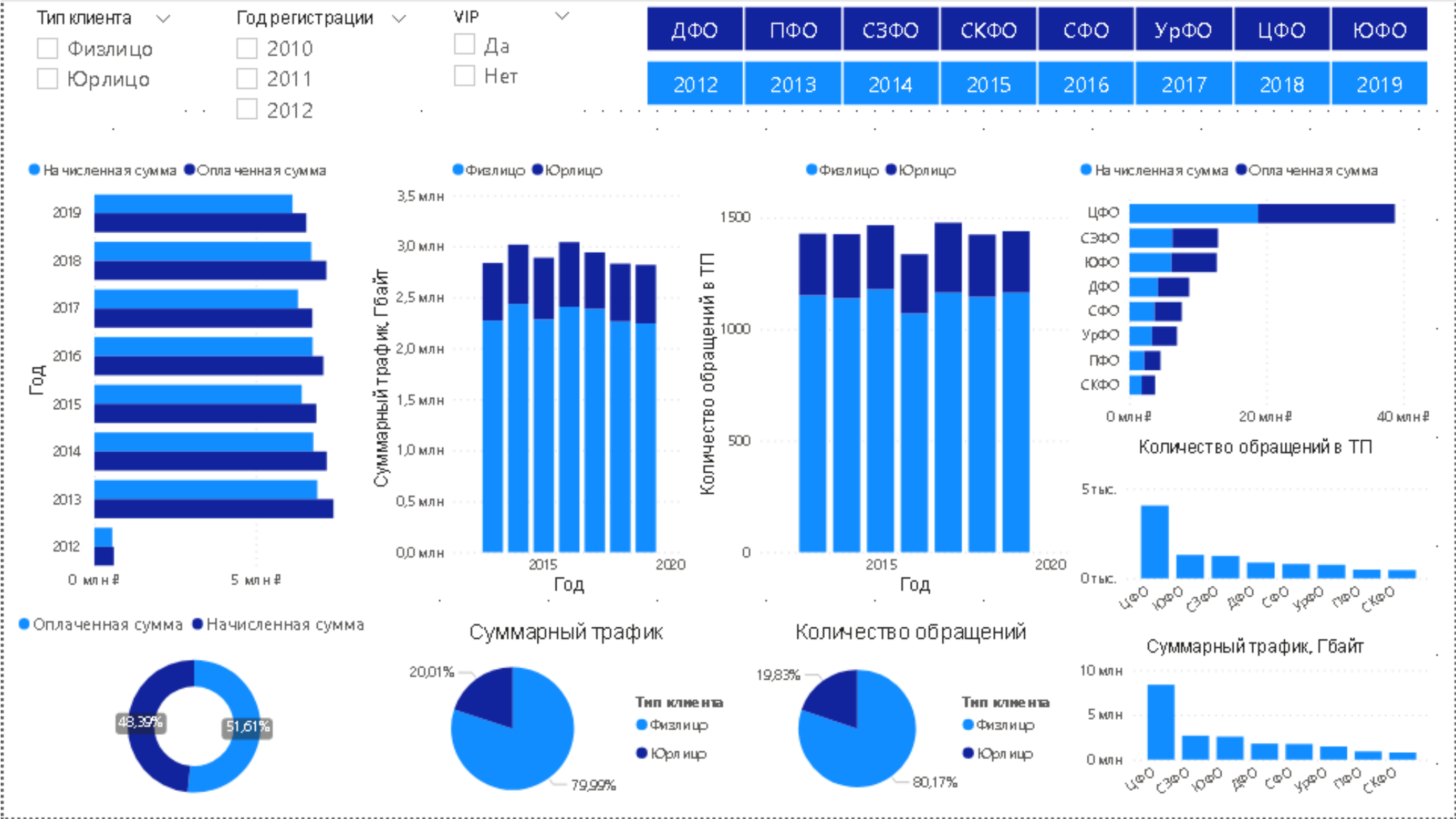
project_report_dim_billing_mode1

select * from yfurman.project_report_dim_billing_mo

Таблица	id	billing_mode_key
1	1	Постоплатный
2	2	Предоплатный

Описание витрины

С использованием инструментария **Microsoft PowerBI** был построен отчет для витрины:



Контроль качества данных

Под качеством данных, применительно к DWH, понимают – характеристику, показывающую степень пригодности данных к дальнейшему использованию.

Из основных метрик качества в данной работе реализован контроль :

Полноты данных:

- Данные присутствуют за весь период, без пропусков
- Все объекты из исходной системы представлены в целевой

Валидности данных:

- Даты соответствуют заданному формату: 2018-03-05
- Номера телефонов имеют вид: +79271235678
- IP-адреса удовлетворяют шаблону: 195.217.200.125
- Суммы платежей находятся в диапазоне: 0.1 до 1000 000 руб.

Целостности данных:

- Правильные типы данных

Контроль качества данных


Для контроля качества данных в данной работе была использована Python-based open-source library **Great Expectations** (<https://greatexpectations.io/>). Проверки данных проводились для четырех источников данных (**billing, issue, payment, traffic**) в слое ODS.

В результате проверок было обнаружено, что данные в источнике **payment** *не прошли проверку* по метрике **валидность** — имеются записи, содержащие платежи со значением сумм **0.00**, что противоречит бизнес-логике.

Данные в остальных источниках прошли успешные проверки по всем приведенным выше метрикам.

Контроль качества данных

Приведем несколько примеров проведенных проверок качества данных:

 great_expectations

Data Docs autogenerated using [Great Expectations](#).

Data Docs | local_site

Validation Results

Expectation Suites


Search

Clear Filters

Status	Run Time	Run Name	Asset Name	Batch ID	Expectation S
✓	2021-05-17 20:25:30 +04	20210517T162530.312915Z	yfurman.project_ods_issue	e114f6a9f193b2511643b8420f70fe91	yfurman.project_ods_issue.warning
✓	2021-05-17 20:03:22 +04	20210517T160322.469357Z	yfurman.project_ods_traffic	c3716ed43188ae88341ce2c4da46e612	yfurman.project_ods_traffic.warning
✓	2021-05-17 18:58:30 +04	20210517T145830.663119Z	yfurman.project_ods_billing	5494e0af290b6ceef7cb9d766704d076	yfurman.project_ods_billing.warning
✗	2021-05-17 15:59:28 +04	20210517T115928.039717Z	yfurman.project_ods_payment	76bc6c6826ca09eb4f233a8fd8097592	yfurman.project_ods_payment.warning

Контроль качества данных

Пример для источника **billing**:

 great_expectations

Home / Validations / yfurman.project_ods_billing.warning / yfurman.project_ods_billing / 2021-05-17T14:58:30.663119+00:00


expectations.

Actions

Validation Filter:

Show All

Failed Only

 How to Edit This Suite

Show Walkthrough

Table of Contents

Overview

Table-Level Expect

Statistics

Evaluated Expectations	29
Successful Expectations	29
Unsuccessful Expectations	0
Success Percent	100%

Show more info...


Table-Level Expectations

Search

Status	Expectation	Observed Value
✓	Must have these columns in this order: <code>user_id</code> , <code>billing_period</code> , <code>service</code> , <code>tariff</code> , <code>sum</code> , <code>created_at</code>	['user_id', 'billing_period', 'service', 'tariff', 'sum', 'created_at']

Контроль качества данных

Пример для источника **billing**:

 great_expectations

Home / Validations / yfurman.project_ods_billing.warning / yfurman.project_ods_billing / 2021-05-17T14:58:30.663119+00:00

Actions

Validation Filter:

Show All

Failed Only

✎ How to Edit This Suite

Show Walkthrough

Table of Contents

Overview

Table-Level Expectations


billing_period

Search

Status	Expectation	Observed Value
✓	values must never be null.	100% not null
✓	value types must belong to this set: <div>CHAR VARCHARNVARCHARTEXTSTRINGStringTypestringstr</div> .	VARCHAR
✓	values must match this regular expression: <div>(?:19 20)[0-9]{2}-(0[1-9] 1[012])</div> .	0% unexpected
✓	values must belong to this set: <div>2011-012011-022011-032011-042011-052011-062011-072011-082011-092011-102011-112011-122012-012012-022012-032012-042012-052012-062012-072012-082012-092012-102012-112012-122013-012013-022013-032013-042013-052013-062013-072013-082013-092013-102013-112013-122014-012014-022014-032014-042014-052014-062014-072014-082014-092014-102014-112014-122015-012015-022015-032015-042015-052015-062015-072015-082015-092015-102015-112015-122016-012016-022016-032016-042016-052016-062016-072016-082016-09</div>	0% unexpected

Контроль качества данных

Пример для источника **payment**:

 great_expectations

Home / Validations / yfurman.project_ods_payment.warning / yfurman.project_ods_payment / 2021-05-17T11:59:28.039717+00:00

Validation Result


Evaluates whether a batch of data matches expectations.

Actions

Validation Filter:

Show All

Failed Only

 How to Edit This Suite

Show Walkthrough

Table of Contents

Overview

Expectation Suite: yfurman.project_ods_payment.warning

Data asset: yfurman.project_ods_payment

Status: ✖ Failed

Statistics

Evaluated Expectations	35
Successful Expectations	34
Unsuccessful Expectations	1
Success Percent	≈97.14%

Show more info...

sum

Search

Status	Expectation	Observed Value
✖	minimum value must be greater than or equal to 0.01 .	--

Результаты

Выводы:

В данной работе реализовано:

1. Разработана структура DWH. Выделены слои: **STG** (источники), **ODS** (операционный слой), **DDS** (детальный слой), **DM** (витрины). Ядро DWH (детальный слой), построено по методике **Data Vault**.
2. Разработаны/реализованы и автоматизированы **ETL**-процессы для **4 источников** данных из DataLake (*Google Cloud Storage*) и помещение данных из них в нормализованную структуру (**модель DV**) в OLAP базе данных (*Greenplum Database*). Все реализованные ETL-процессы обеспечивают **идемпотентный репроцессинг**
3. Построена витрина данных (**модель «звезда»**). Для витрины данных построен **отчет** в *Microsoft PowerBI*
4. Обеспечен контроль качества поступающих в хранилище данных (с использованием библиотек *Great Expectations*)

Результаты

Выводы:

Что осталось «за кадром»:

1. Не реализована оркестрация перерасчета витрины в случае репроцессинга (повторного ETL-процесса) данных из источников за «прошлые» периоды. Т.к. мы принудительно запрещаем расчет витрины для всех инстансов, кроме последнего, это приводит к тому что, если витрина уже была рассчитана, то любой повторный ETL-процесс за период, отличный от последнего, заданного в планировщике, не приведет к ее перерасчету.
2. Не реализована оркестрация контроля качества данных источников. *Great Expectation* позволяет создать *checkpoint* для каждого набора проверок. Для каждой загружаемой партии источника можно создать в DAG задачу проверки качества (*great_expectations checkpoint run название_checkpoint*). В случае возврата кода ошибки – прерывать ETL-процесс для данного источника.
3. Функционал репроцессинга для слоя DM реализован только с условием полного пересчета таблицы фактов,

Результаты

При выполнении данной работы были использованы:

Материалы лекций и практических занятий курса РТК Data Engineer по следующим темам:

1. Архитектура аналитических решений.
2. Управление процессами обработки данных
3. Построение Data Lake
4. Построение Data Warehouse
5. Data Governance
6. Инструменты BI

Сервисы, программные средства и библиотеки:

- | | |
|------------------------------------|-----------------------|
| 1. Google Cloud Platform | 8. Great Expectations |
| 2. Apache Airflow | 9. Jupiter Notebook |
| 3. Python (v. 3.8) | 10. Oracle WorkBench |
| 4. Greenplum Database (Postgresql) | 11. GitHub.com |
| 5. DBeaver (v. 21.0.4) | |
| 6. Microsoft Azure Platform | |
| 7. Microsoft PowerBI | |



Ростелеком

Спасибо за внимание!

Технологии
возможностей