**U2.EXIT TASK**

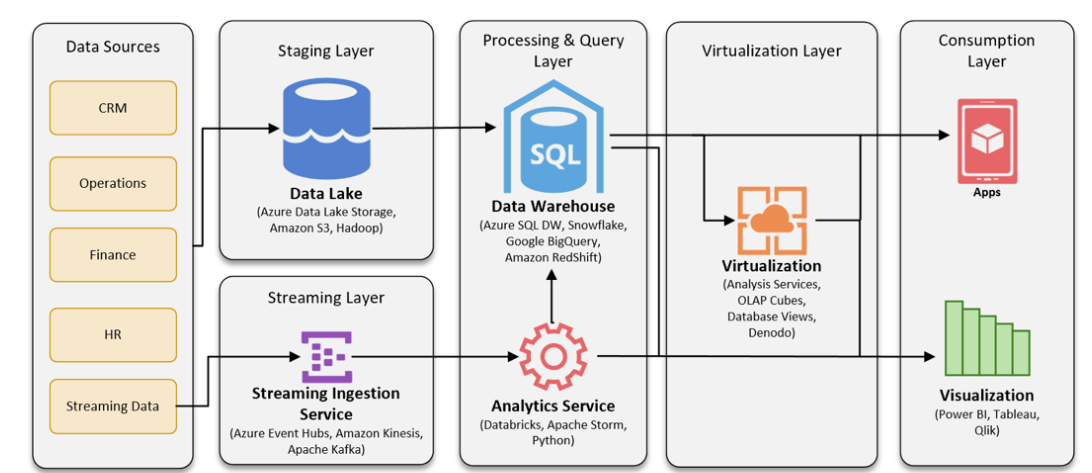
Andrey Neveykov

2022

**Дополнительная информация о базе данных для бизнес-проекта:**

**Architecture**

Для моего бизнеса лучше всего подходит *Lambda* архитектура или смешанная т.к. есть часть Streaming Data (orders и customers), актуальность которых нужно проверять постоянно, остальные обновляются реже, например информация о новых сотрудниках от HR-ов, финансовые операции для квартальных и годовых отчетов, существуют справочники календаря и регионов, которые вообще никогда не обновляются. Так получилось, что идеальная архитектура моего бизнеса полностью совпадает со следующей схемой:

****

**Extraction Description**

Full Extraction

Суть метода: данные полностью извлекаются из исходной системы. Поскольку это извлечение отражает все данные, доступные в настоящее время в исходной системе, нет необходимости отслеживать изменения в источнике данных с момента последнего успешного извлечения.   
 Этот способ в бизнесе может использоваться при получении данных о заказе с сайта: т.к. эти таблицы хранят мало информации, а скорость выполнения заказа очень важна для привлечения и сохранения клиента выгоднее постоянно обновлять информацию о заказе. Быстрее получим информацию -> быстрее начнем выполнять заказ -> быстрее отдадим заказ клиенту -> клиент сильнее захочет купить что-нибудь еще.

Offline Extraction

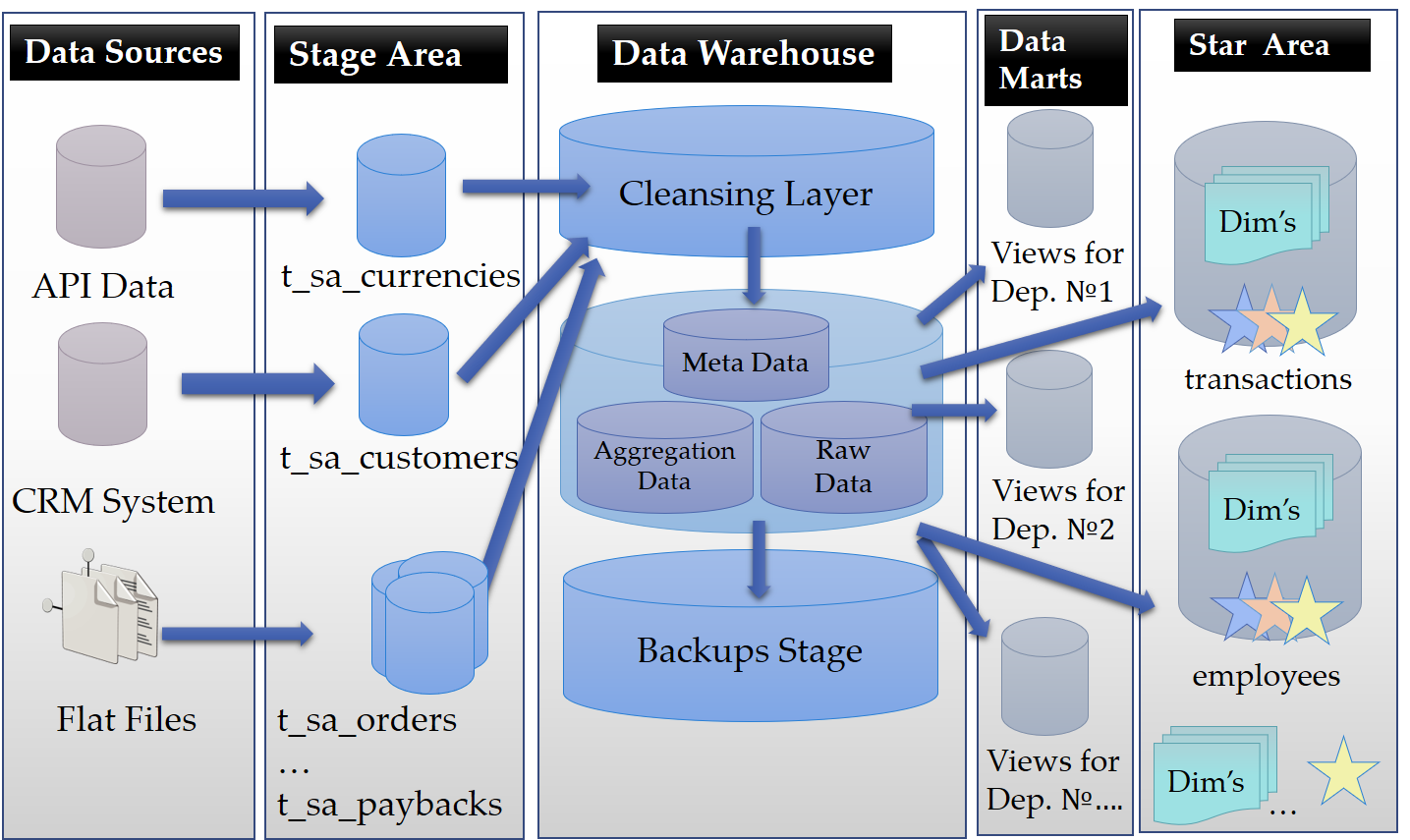
Суть метода: данные не извлекаются непосредственно из исходной системы, а с отдельно выделенных мощностей, чтобы не нагружать основную систему и не блокировать к ней доступ из-за большой загрузки. В бизнесе может использоваться при составлении финансовых отчетов для налоговой, когда необходима точность в подсчете большого объема транзакций за длительный период (например: квартал, финансовый год). Выгоднее проверить не изменились ли исторические данные, не нагружая основную сеть, т.к. подобная проверка может быть выполнена на меньших мощностях, но за большее количество времени.   
 Этот способ выгоднее использовать во всех остальных случаях.

**Преобразование данных в моей бизнес-модели**

В моем бизнесе лучше использовать совместно преобразование данных с помощью SQL для небольших таблиц, таблиц, которые редко обновляются и таблиц-справочников т.к. их простая структура обеспечит максимальную скорость трансформации. Для факт-таблиц, в которые добавляются id из других таблиц лучше использовать преобразование с помощью PL/SQL т.к. надо объединять несколько таблиц.

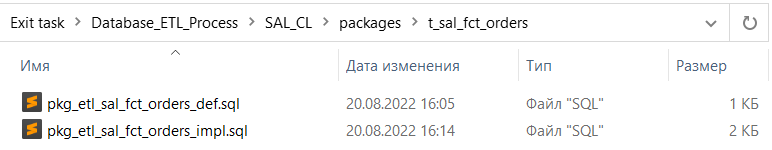
Схема трансформации данных:

Из data sources данные попадают в stage area, далее из нее в cleansing layer, где происходит поиск повторяющихся значений, null­-полей, проверка форматов данных и происходит трансформация под единый стандарт, после cleansing layer-а создаются индексы и данные разделяются между таблицами в Data Warehouse. Когда в DW собрана вся информация, создаются Data Marts для каждого отдела (бухгалтеров, продавцов, директоров и т.д.). На star area объединяются данные из нескольких dimentions для составления финальных отчетов или создания информационных продуктов для клиентов на основе проанализированных данных.



# 

Создание скриптов по следующей структуре:



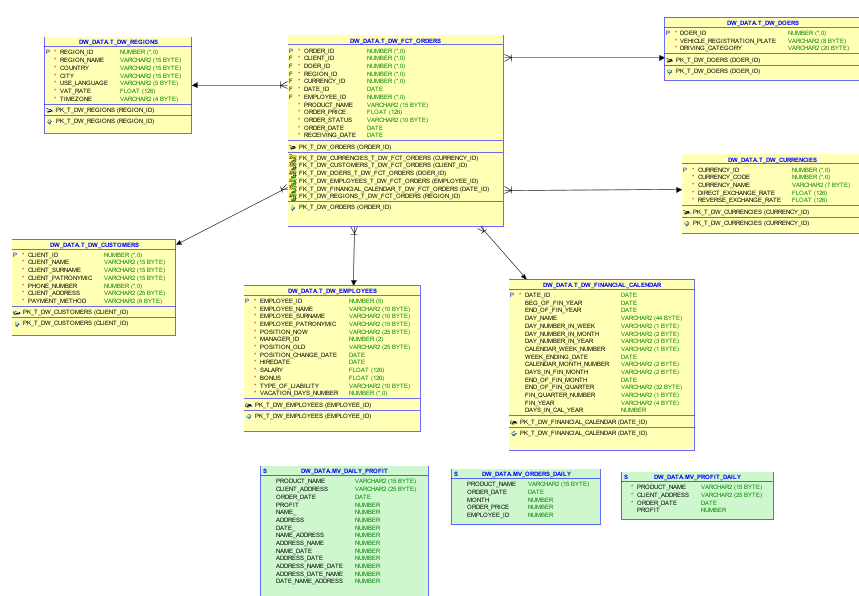
Сначала указывается к какому заданию относится папка «Exit task», дальше назначение папки «Database\_ETL\_Process» (в ней хранится всё, что нужно для развертывания базы данных). Далее один из уровней «SA/CL/DW/SAL\_CL», дальше две папки «tables» и «packages»: в tables хранится всё, что касается создания таблиц, создания ключей, связей, партиций, а в package – всё, что касается заполнения данными.

# 2. Result Task 01: Test Star Data

Create required screenshots:

* SCD2 Dimension tests – Repeatable, Content
* Common Dimension
* Fact Table
* Star Solution Scheme (Physical Objects)
* Statistics for Star Scheme

**Star Solution Scheme (Physical Objects):**



**Statistics for Star Scheme:**

В таблицах Star схемы 5 типов валют, в которых ведется расчет в зависимости от региона и страны (таблица справочник)



В таблице клиентов 200.000 записей



В таблицах сотрудников и исполнителей по 150 записей т.к. каждый сотрудник может выполнять заказ.

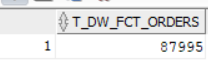




Финансовый календарь сгенерирован на 2,5 года



Количество заказов



Количество финансовых операций

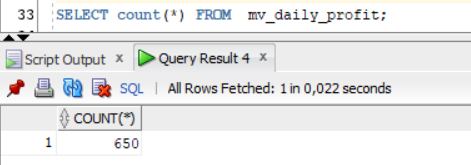


Количество регионов в которых закупается товар (таблица-справочник)

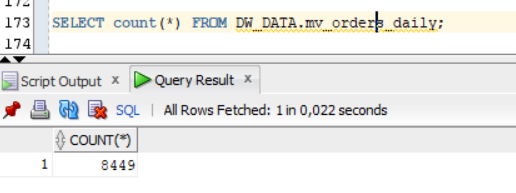


Материализованные представления для расчета бизне-метрик:

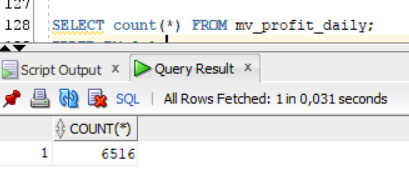
Профит по наименованию товара в зависимости от адреса доставки:



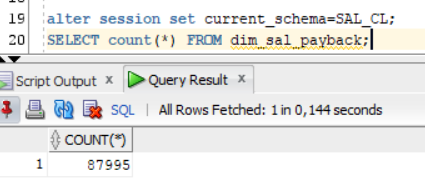
Структура заказов кондиционеров за 3 месяца:



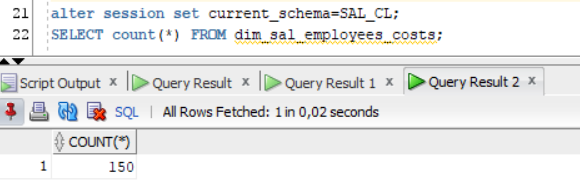
Профит за день со всех заказанных компьютеров:



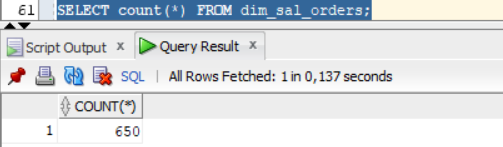
Dim sal\_paybacks, показывающий структуру прибыли/убытков на каждый заказ (покупка, транспортировка, страховка, …):



Dim sal\_employees\_costs показывает суммарные расходы на сотрудников:

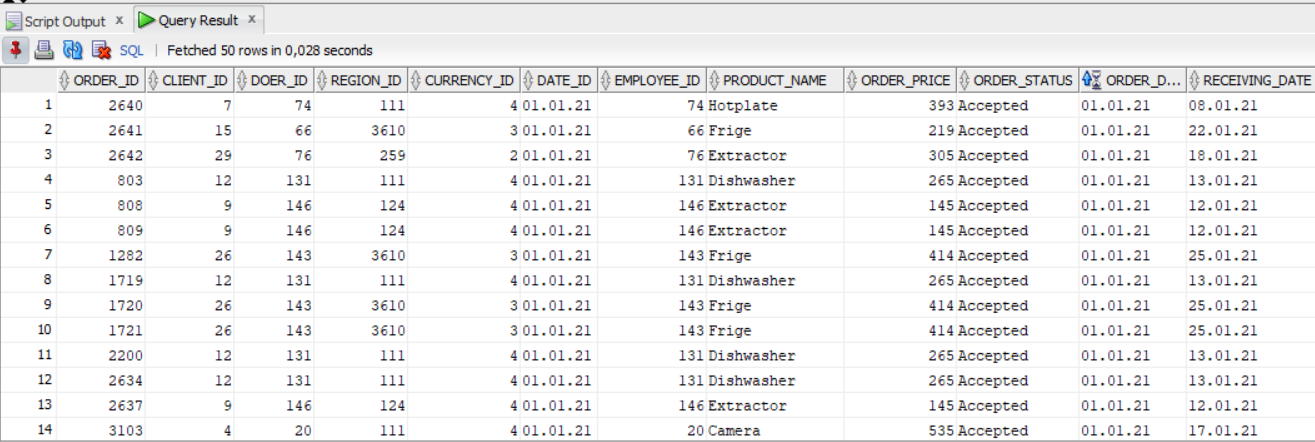


Dim sal\_orders, состоящий только из группировок, показывающий структуру заказов:

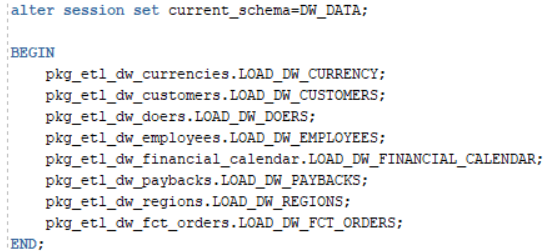


**Fact Table:**

Select из fact table:

****

Чтобы заполнить фактовую таблицу, сначала необходимо прогрузить dimensions в правильном порядке, чтобы не нарушать constraint на связи между таблицами. Скриншот процедуры:



**Common Dimension**

Dimention, собранный из всех dimention и fact table, такие есть на каждом уровне (SA/CL/DW/SAL) для проверки переноса данных.

Common Dimension содержит все возможные поля:

****

А заполняется вот так (через JOIN таблиц):

alter session set current\_schema=DW\_DATA;

INSERT INTO t\_dw\_transactions /\*+ parallel(DW\_CL.t\_cl\_transactions, 4)\*/

(

SELECT ord.order\_id,

product\_name,

order\_price,

order\_status,

order\_date,

receiving\_date,

-----------------------------------------------------------

cust.client\_id,

client\_name,

client\_surname,

client\_patronymic,

phone\_number,

client\_address,

payment\_method,

-----------------------------------------------------------

emp.employee\_id,

employee\_name,

employee\_surname,

employee\_patronymic,

position\_now,

manager\_id,

position\_old,

position\_change\_date,

hiredate,

salary,

bonus,

type\_of\_liability,

vacation\_days\_number,

-----------------------------------------------------------

doer.doer\_id,

vehicle\_registration\_plate,

driving\_category,

-----------------------------------------------------------

customer\_payment,

product\_costs,

fuel\_costs,

vehicle\_depreciation,

insurance\_cost,

unexpected\_expenses,

-----------------------------------------------------------

reg.region\_id,

region\_name,

country,

city,

use\_language,

VAT\_rate,

timezone,

-----------------------------------------------------------

cur.currency\_id,

currency\_code,

currency\_name,

direct\_exchange\_rate,

reverse\_exchange\_rate,

-----------------------------------------------------------

fin\_c.date\_id,

beg\_of\_fin\_year,

end\_of\_fin\_year,

day\_name,

day\_number\_in\_week,

day\_number\_in\_month ,

day\_number\_in\_year,

calendar\_week\_number,

week\_ending\_date,

calendar\_month\_number,

days\_in\_fin\_month,

end\_of\_fin\_month,

end\_of\_fin\_quarter,

fin\_quarter\_number,

fin\_year,

days\_in\_cal\_year

FROM

DW\_DATA.t\_dw\_fct\_orders ord

INNER JOIN DW\_DATA.t\_dw\_customers cust ON cust.client\_id = ord.client\_id

INNER JOIN DW\_DATA.t\_dw\_employees emp ON emp.employee\_id = ord.employee\_id

INNER JOIN DW\_DATA.t\_dw\_doers doer ON doer.doer\_id = ord.employee\_id

INNER JOIN DW\_DATA.t\_dw\_paybacks payb ON payb.order\_id = ord.order\_id

INNER JOIN DW\_DATA.t\_dw\_regions reg ON reg.region\_id = ord.region\_id

INNER JOIN DW\_DATA.t\_dw\_currencies cur ON cur.currency\_id = ord.currency\_id

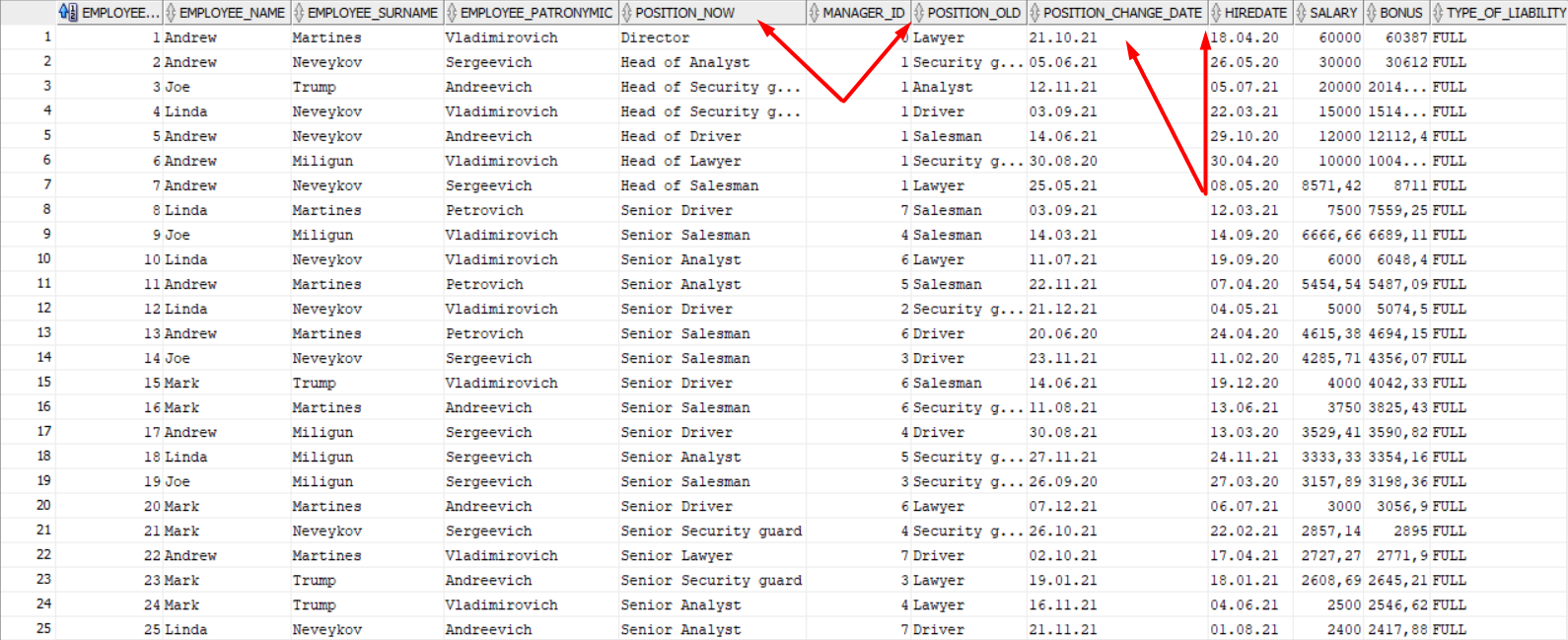
INNER JOIN DW\_DATA.t\_dw\_financial\_calendar fin\_c ON fin\_c.date\_id = ord.date\_id

);

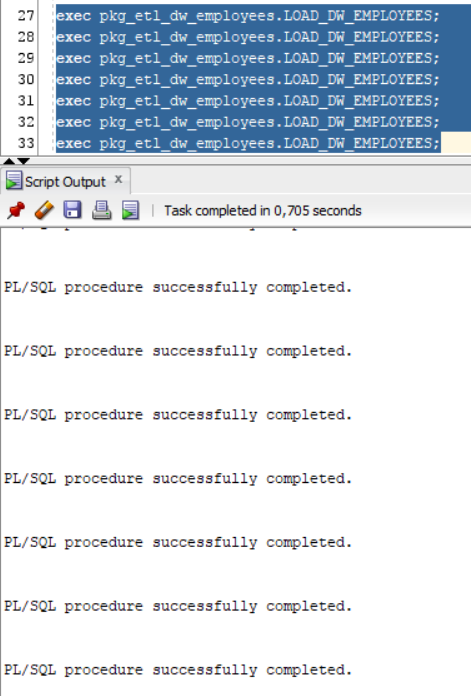
**SCD3 Dimension tests – Repeatable, Content**

В результате выполнения работы был создан dim *SCD3*.

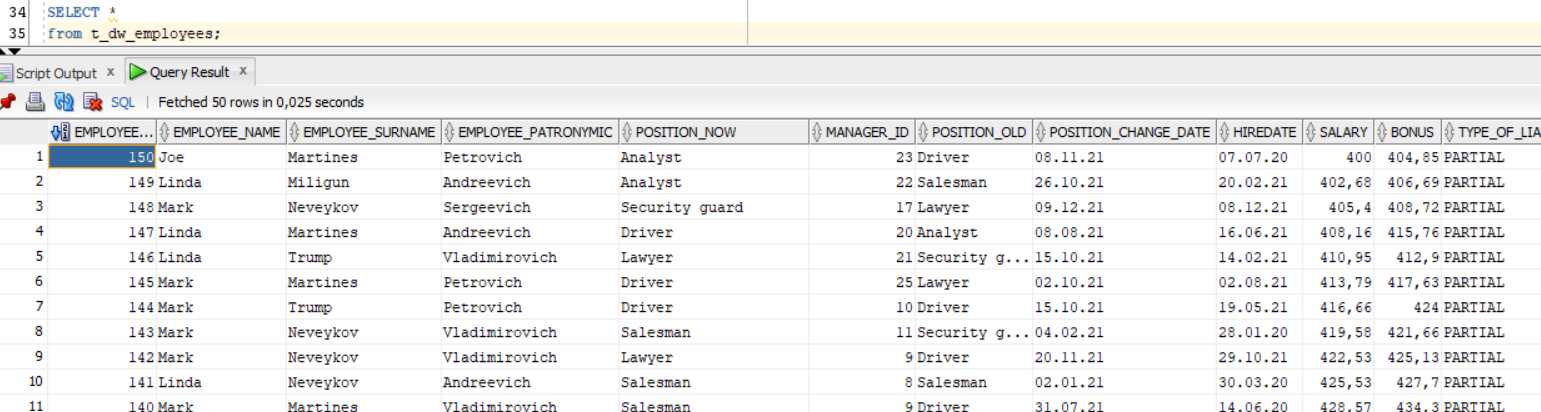
Content: dimension содержит текущую должность, предыдущую и дату, когда наняли на работу, когда поменял должность.



Repeatable: выполнил процедуру по загрузке dim 7 раз подряд.



Результат такой же, как если бы выполнил один раз:



# 3. Result Task 02: Performance comparison

**The Main Task** is to create summarize table with comparison Performance of next Report Layout:

* Advancing Grouping (GROUP BY GROUPING SETs – LabWork 02)
* Model Clause (LabWork 05)
* Star Schema (LabWork 11)

**Task Results:**

Create required objects:

Prepare Document with Summarize table

|  |  |  |  |
| --- | --- | --- | --- |
| № | Source Type | Explain Plan - Statistics | Time, Sec. |
| 1 | Advancing Grouping |  | 0.259 |
| 2 | Model Clause |  | 0.211 |
| 3 | Star Schema |  | 0.00  (моментально) |







**Вывод:** Самый эффективный через Star Schema, т.к. Ad Hoc использует более оптимальные алгоритмы. И результат выполнения моментальный.

Самая низкая производительность оказалась у Advancing Grouping. Т.к. результат вывода у всех трех методов одинаковый, судя по плану выполнения, Advancing Grouping – это решение задачи в лоб (простая последовательность действий), значит не самым оптимальным образом.

У Model Clause и Star Schema совпадает план выполнения, но использование Ad Hoc в Star Schema делает его быстрее.