



Академия
Аналитиков
Авито

AAA SQL+DB

10. Модификация данных+DDL



Повторим пройденное

- ▶ Чем отличается LEAD от LAG?
- ▶ Что делает FIRST_VALUE?
- ▶ Для чего нужны CTE?
- ▶ Когда удобнее использовать рекурсию?

ЧТО БУДЕМ ДЕЛАТЬ СЕГОДНЯ

Создавать и модифицировать таблицы:

- ▶ **CREATE**
- ▶ **INSERT**
- ▶ **UPDATE**
- ▶ **DELETE**
- ▶ **TRUNCATE**



Create



Зачем это нужно?



Обсуждение



5 минут

Зачем это нужно?

Для создания витрин данных

- Для материализации результата запросов.
- Витрины с разными таблицами и источниками для конкретных задач.
- Специальные витрины для дашбордов (привет Redash).



Обсуждение

Вопросы по прериду

- Что такое **схема** данных, зачем она нужна ?
- Из каких основных частей должен состоять запрос на создание таблицы?
- Зачем используется **not null** и **unique** ?
- Как создать таблицу и не наполнять ее данными?



Обсуждение



3 минуты

Задача 1: Создание таблицы клиентов

Условие:

Создайте таблицу с названием <ваш логин>_clients, которая будет хранить информацию по клиентам магазина.

Атрибуты клиента: уникальный идентификатор, имя, фамилия, количество заказов.



Работаем вместе



3 минуты

Задача 2: Создание таблицы заказов

Условие:

Создайте таблицу с названием <ваш логин>_orders , которая будет хранить информацию по заказам магазина.

Атрибуты заказа: идентификатор заказа, идентификатор клиента, название товара, количество и стоимость за единицу.



Работаем вместе



3 минуты



insert, update, add column



Вопросы по прериду

- Можно ли вставить в таблицу результат запроса?
- Как изменить порядок колонок при вставке данных в таблицу?
- Как удвоить значение колонки в уже созданной таблице?



Обсуждение



3 минуты

Задача 3: Заполнение таблицы данными

Условие:

Заполните таблицу с пользователями фиктивными данными с помощью `insert into ... values`. Значение колонки с количеством заказов принять за `null`.

Заполните таблицу с заказами фиктивными данными с помощью `insert into ... select ...`

Заполните таблицу с заказами фиктивными данными с помощью `generate_series()` + `random()`



Практика



9 минут

Задача 4: Обновление таблицы

Условие:

Обновите таблицу с пользователями, используя таблицу с заказами как источник для последней колонки



Практика



7 минут

Задача 5: Добавление колонок

Синтаксис:

ALTER TABLE <schema>.<name> ADD COLUMN <column_name> <column_type>

Условие:

Добавьте колонку с отчеством (patronymic) в таблицу с клиентами



Практика



5 минут



Truncate Delete



Вопросы по прериду

- Чем Truncate отличается от Delete?



Обсуждение



2 минуты

Задача 6: Удаление данных

Условие:

Удалите какого-нибудь пользователя и все его заказы из таблиц



Работаем вместе



4 минуты



Перерыв

10 минут

Задача 7: Пройдем весь путь

- ▶ В таблицу с клиентами добавьте колонку с общей суммой заказов и числом невыполненных заказов.
- ▶ Создайте таблицу, хранящую номер и статус заказа
- ▶ Наполните эту таблицу данными с помощью `generate_series`
- ▶ Напишите запрос, обновляющий таблицу с пользователями исходя из новых требований



Практика



35 минут

Summary

- ▶ Ключевые слова - **Create, Insert, Update, Truncate, Delete**
- ▶ Узнали как создавать, модифицировать и удалять таблицы и данные в них



Теория

Вопросы

Что осталось непонятным?

Фидбек

Что нового вы узнали на этом занятии?

Что показалось самым важным?

Что будете применять и в каких ситуациях?

Что хочется изучить подробнее?

[обратная связь](#)

Что делаем в следующий раз?

- ▶ Чтение планов запросов
- ▶ Оптимизация запросов
- ▶ Консультация перед экзаменом
(готовьте вопросы)

Обязательно ознакомьтесь с преридом =)

Домашнее задание

Мягкий дедлайн: 10:00 BC

Жесткий дедлайн: 10:00 BT

ОБСУЖДЕНИЕ ДЗ

01.

На каждую дату изменения выведите остаток по счету, собрав его по истории движений накопительным итогом

```
with src as (  
    select acc_id, dt, amnt  
    from d9_income  
    union all  
    select acc_id, dt, -1 * amnt  
    from d9_withdraw  
)  
, agg as (  
    select acc_id, dt, sum(amnt) amnt  
    from src  
    group by 1, 2  
)  
select dt, acc_id, sum(amnt) over (partition by acc_id order by dt) balance  
from agg
```

ОБСУЖДЕНИЕ ДЗ

02.

Для каждого менеджера первого уровня найдите долю, которую составляет его зарплата и зарплата всех его подчиненных от общих трат на зарплату

```
with recursive tree as (  
    select id, name, salary, id as top_manager_id  
    from d9_employee  
    where manager_id is null  
    union all  
    select d9_employee.id, d9_employee.name, d9_employee.salary, tree.top_manager_id  
    from d9_employee  
        join tree on d9_employee.manager_id = tree.id  
)  
select top_manager_id  
    , round(100.0 * sum(salary) / sum(sum(salary)) over (), 0) as department_salary_pct  
from tree  
group by 1
```

ОБСУЖДЕНИЕ ДЗ

03.

Найдите игроков, возвращавшихся в свой первый город.

```
select distinct player_id
from (
    select *, first_value(city) over w first_city
           , lag(city) over w prev_city
    from d8_player_city
    window w as (
        partition by player_id
        order by actual_date
    )
) _
where city = first_city
       and not city = prev_city
```

ОБСУЖДЕНИЕ ДЗ

04.

Вывести цепочку таблицек которая заполняется дольше всего.

```
with recursive all_paths as (  
    select dm as last_dm, dm as calc_path, calc_time  
    from d9_datamarts  
    where dm not in (select tgt from d9_dag)  
    union all  
    select  
        d9_dag.tgt as last_dm,  
        CONCAT(all_paths.calc_path, ' -> ', d9_dag.tgt),  
        all_paths.calc_time + d9_datamarts.calc_time  
    from all_paths  
    join d9_dag on d9_dag.src = all_paths.last_dm  
    join d9_datamarts on d9_dag.tgt = d9_datamarts.dm  
)  
select calc_path, calc_time  
from all_paths  
order by calc_time desc  
limit 1
```

ОБСУЖДЕНИЕ ДЗ

05.

Для каждой платформы найдите пользователя с наибольшей средней транзакцией в рамках трехдневного окна.

```
with user_windows as (  
    select platform_id  
           , user_id  
           , dt  
           , amnt  
           , round(avg(amnt) over w, 0)      avg_amnt_w_3d  
           , first_value(log_id) over w      first_row_w_3d  
           , last_value(log_id) over w       last_row_w_3d  
           , cast((min(dt) over w) as date) first_date_w_3d  
           , cast((max(dt) over w) as date) last_date_w_3d  
           , log_id  
    from d9_payments  
    window w as (  
        partition by platform_id, user_id  
        order by cast(dt as date)  
        range between interval 2 day preceding and current row  
    )  
) , -- продолжение на следующем слайде
```

ОБСУЖДЕНИЕ ДЗ

05. Для каждой платформы найдите пользователя с наибольшей средней транзакцией в рамках трехдневного окна.

```
...
    user_windows_rn as (
        select *, row_number() over (partition by platform_id order by avg_amnt_w_3d desc, dt) rn
        from user_windows
        order by platform_id, user_id, dt
    )
select d9_platform.nm as platform_name,
       user_id,
       avg_amnt_w_3d,
       first_row_w_3d,
       last_row_w_3d,
       first_date_w_3d,
       last_date_w_3d
from user_windows_rn
     join d9_platform on id = platform_id
where rn = 1
order by platform_id, user_id, dt
```