## Задания:

1. Напишите, пожалуйста, бизнес-функциональные требования на разработку железного почтового ящика, который висит в каждом доме.
2. В файле «Rule Expression Reference.pdf» – описание языка для построения правил системы автоматизации маркетинга. Пожалуйста определите, что происходит в этих 2-х правилах:

count( filter(filter(filter( get("Account.contact\_history"), "contact\_time", formatDateTime("day"), greaterThan(time(now, 30,"day","past")) ), "channel", "Call Center"), "product", "ТП МР")) > 0&&count( filter( get(Account.active\_services), service\_id, contains (list("11208","11220","18644"))))==0

set("RESPONSE", get("contact")) && set("M", get("Message")) && set("M\_ID", get(M, "ID")) && count(filter(filter(filter( get("Account.contact\_history"), "date\_time", formatDateTime("day"), greaterThan(time(now, 365,"day","past"))), "message\_id", M\_ID ), "RESPONSE", contains(list("Yes")))) == 0

1. Дана Ситуация – ваш клиент просит показать как через trello api удалить карточку (Card) на его доске(Board) с именем ToDoBoard. Необходимо подготовить и предоставить видео ролик с демо для пользователя где виден процесс - сначала карточка ToDoBoard есть потом делается запрос на удаление, потом ее уже нет. Для демо использовать можно любой инструмент для REST запросов - curl, Postman и т.д.
2. Файл 2.groovy - пример скрипта окружения системы. Комментарии и прочие подсказки оттуда убраны. Необходимо определить, что происходит при его выполнении.
3. Таблица TEST имеет несколько миллиардов записей. Баланс у клиента может меняться один раз в день, при изменении в таблицу добавляется новая запись.

CREATE MULTISET TABLE TEST

( SUBS\_ID INTEGER

,BALANCE\_AMT INTEGER

,BALANCE\_DATE DATE)

PRIMARY INDEX (SUBS\_ID)

PARTITION BY RANGE\_N (

BALANCE\_DATE BETWEEN date '2000-01-01' and '2018-12-31' Each interval '1' Month,

NO range OR UNKNOWN );

Напишите, пожалуйста, запрос, который вернёт текущий и предыдущий балансы всех клиентов. Поясните почему вы написали запрос именно таким образом. Как можно решить эту задачу менее затратным способом?

Результат запроса должен выглядеть так:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SUBS\_ID | BALANCE\_DATE\_CURR | BALANCE\_AMT\_CURR | BALANCE\_DATE\_1 | BALANCE\_AMT\_1 |

1. Напишите приложение на Java\Groovy, которое решает следующую задачу:

Имеется массив строк. Необходимо определить строку, которая встречается чаще всего и вывести количество слов в данной строке.

Решение:

**1)**

Цель

Создание инструмента, позволяющего накапливать письма для обеспечения удобной коммуникации между пользователями почтового сервиса

Пользователи системы

* Роль отправитель
* Роль получатель
* Роль почтальон
* Роль руководитель почтового отделения

Проблемы

Отсутствие единого механизма хранения и взаимодействия между пользователями почты

Решение проблемы

Создание почтового ящика и его ввод в эксплуатацию

Задачи

Определение функционала ящика, удовлетворяющего требованиям всех лиц

Определение форм фактора ящика

Согласование с управляющей компанией

Установка в надлежащем для этого месте

Анализ и устранение проблем этапа эксплуатации

Сроки

На разработку предусмотрен месяц с момента начала.

Основные этапы:

* 1. Сбор и анализ требований
  2. Разработка прототипа
  3. Тестирование
  4. Установка в назначенном месте
  5. Приемка и сверка результатов

Исполнитель

Подразделение разработки почтовых сервисов, руководитель Багавиев Б. И

Ограничения

* Бюджет (100.000 RUB)
* Временные сроки
* Законодательные требования

Функциональные требования к системе

* Почтовый ящик должен иметь механизм помещения в себя писем и посылок без аутентификации пользователя
* Почтовый ящик должен однозначно идентифицировать своего пользователя;
* Почтовый ящик должен защищать свое содержимое от механических повреждений;
* Почтовый ящик должен быть изготовлен из железа;
* Почтовый ящик должен вмещать в себя достаточное количество писем в рамках разумного срока проверки почты;
* Почтовый ящик должен иметь интуитивно понятный интерфейс своего форм-фактора;
* Почтовый ящик должен аутентифицировать пользователя, чтобы только допущенное лицо могло извлечь содержимое ящика;
* Почтовый ящик должен иметь достаточный габаритный размер отверстия и общего объема для хранения мелких посылок;
* Почтовый ящик должен иметь механизм извлечения своего содержимого после аутентификации пользователя

**2)**

1. Выбираем атрибут contact\_history из сущности Account. Далее смотрим поле contact\_time, передаем условие для фильтрации - данные за последние 30 дней. Далее фильтр по атрибуту channel - выбираем данные типа Call Center, активность по звонкам в КЦ видимо. Далее фильтр по продукту, выбираем какой-то продукт ТП МР.

Считаем, чтобы нашлась хотя бы одна запись. Аналогично следующее условие (надо, чтобы выполнилась как первая часть, так и вторая, оператор &). Находим записи из системной сущности Account - active\_services, которые не содержат service\_id из определенного списка (кол-во нулевое).

По итогу получается посмотрели активность по коллцентру за месяц по конкретному продукту, чтобы такие записи встречались, но при это же, чтобы на этом же аккаунте не было подключено сервисов с такими то номерами.

2. Тут получается создаем переменную RESPONSE на основе поля contact. Создаем переменную M на основе текста системного сообщения. Аналогично поле M\_ID на основе ID сообщения.

Далее берем те записи из истории контактов за последний год, которые содержат ID определенного нами сообщения, и не содержат в поле RESPONSE сообщений с текстом "Yes".

**3)**

Все отработало отлично, см видео семпл «teradata\_trello.mp4». Сначала думал сделать через Java библиотечку, потом закопался, и сделал через предложенный Postman. Впервые попробовал этот сервис, все удобно.

0) Получить api\_key(32) и api\_token(64). Сделать подтверждение на веб-интерфейсе

1) Создать доску, список на доске, карточку на списке.

Узнать их id из интерфейса, открыв их и набрав в конце url'a .json

Копирнуть из json в браузере найденные айдишники.

2) Получить доску:

https://api.trello.com/1/boards/?key=&token=

3) Получить карточку на доске:

https://api.trello.com/1/boards//cards/?key=&token=

4) Добавить карточку на доске:

https://api.trello.com/1/cards?key=&token= &idList=&name=Bang new card from Postman!

5) Удалить карточку:

https://api.trello.com/1/cards/6114104094657748519b7454?key=&token=

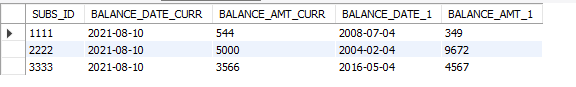
**4)**

Интересно, получается это какая-то fancy версия java, окей. Это очень похоже на java, неплохо читается в целом. Что удалось понять, глядя на исходный код, предположу, это тулза, которая читает зашифрованные файлики PGP из определенной директории, далее расшифровывает их, подключается по jdbc к Teradata, и далее все эти файлики туда грузит в определенную таблицу. Все это происходит вместе логами в консольку и в цикле основного процесса, который ожидает ввода и прерывается в результате выполнения команды. Ну и еще можно отметить наличие где-то конфига и его чтение, для подключения к БД.

**5)**

Скрипт тестировал на MySQL. Решил именно таким образом, т.к. это первое и самое простое, что пришло в голову. Возможно не самый оптимальный вариант, главная оптимизация – таблица партицирована по дате – брать последний срез, чтобы не делать full scan.

Тут беру срез за последний месяц, далее оконной функцией нумерую для каждого абонента его строку с датой. Т.е. каждая дата будет иметь теперь порядковый номер. Затем оставляю только последнюю и предпосленюю записи для каждого абонента. Всего это в CTE выражении, чтобы удобно заиспользовать результат далее. После чего привожу к нужной структуре результат – Join предагрегата на самого себя, соединяю в одну строку текущий баланс и даты вместе предыдущими, для каждого абонента. Не исключаю, что могут быть баги, но на тестовом примере все отработало ожидаемо.



WITH Result AS (

SELECT \* FROM (

SELECT \*, (max\_rn - row\_num) st FROM

(

SELECT \*, max(row\_num) OVER (partitiON by SUBS\_ID) max\_rn FROM (

SELECT \*, ROW\_NUMBER() OVER (PARTITION BY SUBS\_ID ORDER BY BALANCE\_DATE) row\_num

FROM test

WHERE BALANCE\_DATE >= current\_date() - INTERVAL 30 DAY

) a

) b

WHERE (max\_rn - row\_num) <= 1

) c

)

SELECT DISTINCT

t1.SUBS\_ID

, t1.BALANCE\_DATE as BALANCE\_DATE\_CURR

, t1.BALANCE\_AMT as BALANCE\_AMT\_CURR

, t2.BALANCE\_DATE as BALANCE\_DATE\_1

, t2.BALANCE\_AMT as BALANCE\_AMT\_1

FROM Result t1

INNER JOIN (SELECT \* FROM Result WHERE st = 1) t2 ON t1.SUBS\_ID = t2.SUBS\_ID

WHERE t1.st = 0

**6)**

// Код рабочий, запускается

Входной захардкоженный массив строк. Сделал через вспомогательный список и объект. Последний нужен для хранения состояния i-ой строки – исходной позиции и количеству повторов. Идея подобная основана на удобной сортировке объектов в списке через компаратор.

Дубликаты искал перебором исходного списка для i-го элемента проверяем все элементы в списке, если повтор инкрементируем счетчик, иначе смотрим дальше. Полуается n^2 сравнений. Память возможно O(n) потребуется, за счет вспомогательного списка. Так же можно было сделать количество повторов не вручную, а через метод Collections.frequency(), но захотел показать, что руками тоже могу (когда-то делал подобные задачи по поиску дубликатов, тогда запомнил основную идею). Проверки case sensitive, можно было приводить к нижнему регистру все например, но явного требования не было.

import java.util.\*;

public class Solution {

private static String strings[];

private static List<Item> list;

static {

list = new ArrayList<>();

strings = new String[7];

strings[0] = "This was a good day";

strings[1] = "This was a bad day 312 win ggwp";

strings[2] = "Once upon a time...";

strings[3] = "Rolling the deep, i'm an astronaut";

strings[4] = "Once upon a time...";

strings[5] = "This was a good day";

strings[6] = "This was a good day";

}

static class Item {

private int position;

private int counter;

public int getPosition() {

return position;

}

public int getCounter() {

return counter;

}

public Item(int position, int counter) {

this.position = position;

this.counter = counter;

}

}

public static String findMaxStr() { // O(n^2)

for (int i = 0; i < strings.length; i++) {

int cnt = 0;

for (int j = 0; j < strings.length; j++) { // еще можно было сделать через утильный Collections.frequencies()

if (strings[i].equals(strings[j]))

cnt++;

}

list.add(new Item(i, cnt));

}

Collections.sort(list, Comparator.comparing(Item::getCounter));

return strings[list.get(list.size() - 1).getPosition()]; // Отсортировали список по возрастанию. Взяли последний элемент.

}

public static int findWordCnt(String str) {

String[] words = str.split(" ");

return words.length;

}

public static void main(String[] args) {

String result = findMaxStr();

System.out.println(findWordCnt(result));

// 5

// "This was a good day" 3 раза встречается

}

}