|  |
| --- |
| АО Банк ДОМ.РФ |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| Леонтьев Б. П. |
|  |
| **Отчет о выполнении тестовых заданий** |
|  |
| Вакансия «Data Engineer / Автоматизация и аналитика корпоративного бизнеса» |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
|  |
| Москва |
| 2024 |

Оглавление

[Оглавление 2](#_Toc168906489)

[Введение 3](#_Toc168906490)

[1 Тестовое задание №1 4](#_Toc168906491)

[1.1 Постановка задачи 4](#_Toc168906492)

[1.2 Решение 5](#_Toc168906493)

[1.2.1 Таблица GRAPH 5](#_Toc168906494)

[1.2.2 Программы 6](#_Toc168906495)

[2 Тестовое задание №2 9](#_Toc168906496)

[2.1 Постановка задачи 9](#_Toc168906497)

[2.2 Решение 10](#_Toc168906498)

[2.2.1 Таблица customers 11](#_Toc168906499)

[2.2.2 Таблица invoices 13](#_Toc168906500)

[2.2.3 Вид v$\_zad\_2\_1 15](#_Toc168906501)

[2.2.4 Вид v$\_zad\_2\_2 17](#_Toc168906502)

[2.2.5 Процедура pzad2 19](#_Toc168906503)

[2.2.6 Функция fzad2\_3\_najti\_kol\_schetov() 22](#_Toc168906504)

[2.2.7 Результат работы 24](#_Toc168906505)

[3 Тестовое задание №3 25](#_Toc168906506)

[3.1 Постановка задачи 25](#_Toc168906507)

[3.2 Решение 26](#_Toc168906508)

[3.2.1 Таблица test\_db1 27](#_Toc168906509)

[3.2.2 Таблица test\_db2 27](#_Toc168906510)

[3.2.3 Просмотр и сопоставление записей обеих таблиц до обновления 29](#_Toc168906511)

[3.2.4 Программы 31](#_Toc168906512)

[3.2.5 Результат работы 33](#_Toc168906513)

[Заключение 35](#_Toc168906514)

Введение

В документе описывается выполнение трех тестовых заданий на вакансию «Data Engineer / Автоматизация и аналитика корпоративного бизнеса».

В соответствии с заданиями, предполагается, что среда выполнения – MS SQL, PL SQL, pgSQL ( при формулировании заданий приводится код на диалекте MS SQL, при выборе другой среды необходимо, чтобы весь код в ней запускался – и постановка, и решение).

Нами использована СУБД Oracle 19c, IDE Oracle SQL Developer 21.2.0.187.

Тестовое задание №1

Постановка задачи

--Задача № 1

--Есть табличка #Graph с графиками платежей по кредиту, в которой

--есть дата требования платежа(DatPlat) и дата, когда этот платеж был фактически оплачен(DatOplat).

--Написать табличную функцию, которая на вход принимает ключ графика платежей(KeyGr) и дату на которую проводим расчёт,

--на выходе выдаёт табличку из 3 полей (ДатаНачала) и (ДатаКонца)

--в период которых по договору была просрочка и отдельным полем размер интервала в днях

--Если DatPlat < DatOplat то это считается просроченным платежом.

--DatOplat = null считается что платеж не оплачен.

USE Test\_DB

GO

DROP TABLE IF EXISTS Graph

GO

CREATE TABLE Graph(KeyGr INT, DatPlat DATE, DatOplat DATE)

GO

INSERT INTO Graph (KeyGr, DatPlat, DatOplat)

VALUES (1, '20221205', '20221219')

,(1, '20221219', '20230126')

,(1, '20230102', '20230317')

,(1, '20230116', '20230317')

,(1, '20230213', '20230317')

,(1, '20230227', null)

,(1, '20230313', null)

GO

SELECT \* FROM Graph

GO

Решение

Состав проекта:

1. процедура pzad1\_zapusk;
2. функция FZAD1\_1;
3. таблица graph.

Таблица GRAPH

Создание

CREATE TABLE "DWH"."GRAPH"

( "KEYGR" NUMBER(\*,0),

"DATPLAT" DATE,

"DATOPLAT" DATE

)

Добавление записей

INSERT INTO graph (keygr, datplat, datoplat) VALUES (1, TO\_DATE('2022-12-05', 'YYYY-MM-DD'), TO\_DATE('2022-12-19', 'YYYY-MM-DD'));

INSERT INTO graph (keygr, datplat, datoplat) VALUES (1, TO\_DATE('2022-12-19', 'YYYY-MM-DD'), TO\_DATE('2023-01-26', 'YYYY-MM-DD'));

INSERT INTO graph (keygr, datplat, datoplat) VALUES (1, TO\_DATE('2023-01-02', 'YYYY-MM-DD'), TO\_DATE('2023-03-17', 'YYYY-MM-DD'));

INSERT INTO graph (keygr, datplat, datoplat) VALUES (1, TO\_DATE('2023-01-16', 'YYYY-MM-DD'), TO\_DATE('2023-03-17', 'YYYY-MM-DD'));

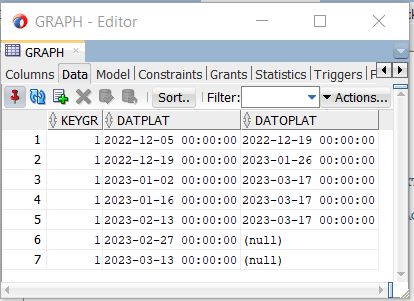
INSERT INTO graph (keygr, datplat, datoplat) VALUES (1, TO\_DATE('2023-02-13', 'YYYY-MM-DD'), TO\_DATE('2023-03-17', 'YYYY-MM-DD'));

INSERT INTO graph (keygr, datplat, datoplat) VALUES (1, TO\_DATE('2023-02-27', 'YYYY-MM-DD'), NULL);

INSERT INTO graph (keygr, datplat, datoplat) VALUES (1, TO\_DATE('2023-03-13', 'YYYY-MM-DD'), NULL);

COMMIT;

Просмотр записей



Программы

Процедура pzad1\_zapusk

CREATE OR REPLACE PROCEDURE pzad1\_zapusk AS

---

i INTEGER ;

---

BEGIN

i := FZAD1\_1( 1, TO\_DATE('2022-12-05', 'YYYY-MM-DD') ) ;

---

END PZAD1\_ZAPUSK ;

Функция FZAD1\_1()

CREATE OR REPLACE FUNCTION FZAD1\_1 ( keygr\_par IN INTEGER, data\_zaprosa\_par IN DATE ) RETURN NUMBER AS

---

CURSOR zapros\_o\_platezhakh( keygr\_par INTEGER, data\_zaprosa\_par DATE ) IS SELECT keygr, datplat AS DATA\_NACHALA, datoplat AS DATA\_KONTSA, TO\_CHAR(datoplat-datplat, '99999999') AS PROSROCHKA FROM graph WHERE keygr = keygr\_par AND datoplat IS NOT NULL ;

rec\_platezh zapros\_o\_platezhakh%ROWTYPE ;

---

BEGIN

---

dbms\_output.put\_line( 'График: ' || keygr\_par || '. Дата запроса о платежах: ' || data\_zaprosa\_par || '.') ;

OPEN zapros\_o\_platezhakh( keygr\_par, data\_zaprosa\_par ) ;

dbms\_output.put\_line( '+-------------------------------------------------------+' ) ;

dbms\_output.put\_line( '| ДАТА НАЧАЛА | ДАТА КОНЦА | ПРОСРОЧКА |' ) ;

dbms\_output.put\_line( '+-------------------------------------------------------+' ) ;

---

LOOP

FETCH zapros\_o\_platezhakh INTO rec\_platezh ;

EXIT WHEN zapros\_o\_platezhakh%NOTFOUND ;

dbms\_output.put\_line( '| ' || rec\_platezh.DATA\_NACHALA || ' | ' || rec\_platezh.DATA\_KONTSA || ' | ' || rec\_platezh.PROSROCHKA || ' |') ;

END LOOP ;

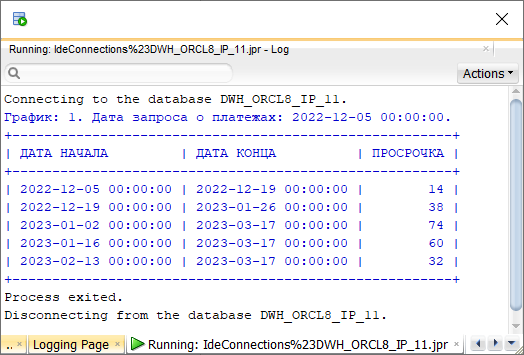
dbms\_output.put\_line( '+-------------------------------------------------------+' ) ;

---

RETURN NULL ;

END FZAD1\_1 ;

Результат работы



# Тестовое задание №2

Постановка задачи

--Задача № 2

--=======================Задание=============================================

-- В таблице #Invoices находятся заказы клиентов (#Customers - таблица клиентов)

-- Требуется написать запрос, который в результате своего выполнения

-- формирует сводку по количеству заказов в разрезе клиентов и месяцев.

-- В строках должны быть месяцы (дата начала месяца), в столбцах - клиенты.

-- Клиенты в столбцах должны быть упорядочены по имени, например Customer:1,

-- Customer:2 и т.д Количество Customer мы не знаем

-- Дата должна иметь формат dd.mm.yyyy, например, 01.03.2023.

-- Пример, как должны выглядеть результаты:

-------------+----------------------+--------------------+----------------+----------------------

--InvoiceMonth | Customer: 1 | Customer: 2 | Customer: 3 | ... (другие клиенты)

-------------+--------------------+--------------------+----------------+----------------------

--01.03.2023 | 3 | 1 | 4 | ...

--01.04.2023 | 7 | 3 | 4 | ...

-------------+--------------------+--------------------+----------------+----------------------

--================================================================================================

--Create 256 Customers=================================

DROP TABLE IF EXISTS #Customers

CREATE TABLE #Customers (ID INT, CustomerName NVARCHAR(400))

;WITH N1(C) AS (SELECT 0 UNION ALL SELECT 0)

,N2(C) AS (SELECT 0 FROM N1 AS T1 CROSS JOIN N1 AS T2)

,N3(C) AS (SELECT 0 FROM N2 AS T1 CROSS JOIN N2 AS T2)

,N4(C) AS (SELECT 0 FROM N3 AS T1 CROSS JOIN N3 AS T2)

,Ides(ID) AS (SELECT ROW\_NUMBER() OVER(ORDER BY (SELECT NULL)) FROM N4)

INSERT INTO #Customers (ID, CustomerName)

SELECT i.ID

,'Customer: ' + TRY\_CAST(i.ID AS NVARCHAR)

FROM Ides i;

--SELECT \*

--FROM #Customers c

--Create Invoices========================================================

DROP TABLE IF EXISTS #Invoices

CREATE TABLE #Invoices (

InvoiceID int NOT NULL IDENTITY,

CustomerID int NOT NULL,

InvoiceDate date NOT NULL)

INSERT INTO #Invoices (CustomerID, InvoiceDate)

SELECT c.id, '20230301'

FROM #Customers c

INSERT INTO #Invoices (CustomerID, InvoiceDate)

SELECT c.id, '20230310'

FROM #Customers c

INSERT INTO #Invoices (CustomerID, InvoiceDate)

SELECT c.id, '20230402'

FROM #Customers c

INSERT INTO #Invoices (CustomerID, InvoiceDate)

SELECT c.id, '20230412'

FROM #Customers c

INSERT INTO #Invoices (CustomerID, InvoiceDate)

VALUES (1,'20230315'), (2,'20230315')

--SELECT \*

--FROM #Invoices i

--============================================================================

Решение

Состав проекта:

1. таблицы customers, invoices;
2. виды V$\_ZAD\_2\_1, V$\_ZAD\_2\_2;
3. процедура pzad2;
4. функция FZAD2\_3.

Таблица customers

Создание

CREATE TABLE "DWH"."CUSTOMERS"

( "ID" NUMBER(\*,0),

"CUSTOMER\_NAME" NVARCHAR2(400)

) ;

Добавление записей

INSERT INTO customers ( id, customer\_name )

SELECT \* FROM

(

WITH

n1 AS ( SELECT 0 AS c FROM DUAL UNION ALL SELECT 0 AS c FROM DUAL )

, n2 AS ( SELECT 0 AS c FROM n1 CROSS JOIN n1 )

, n3 AS ( SELECT 0 AS c FROM n2 CROSS JOIN n2 )

, n4 AS ( SELECT 0 AS c FROM n3 CROSS JOIN n3 )

, ides AS ( SELECT ROW\_NUMBER() OVER ( ORDER BY c ) AS id FROM n4 )

SELECT i.id, 'Customer: ' || TO\_CHAR ( i.id, '9999' ) FROM ides i

);

Просмотр записей

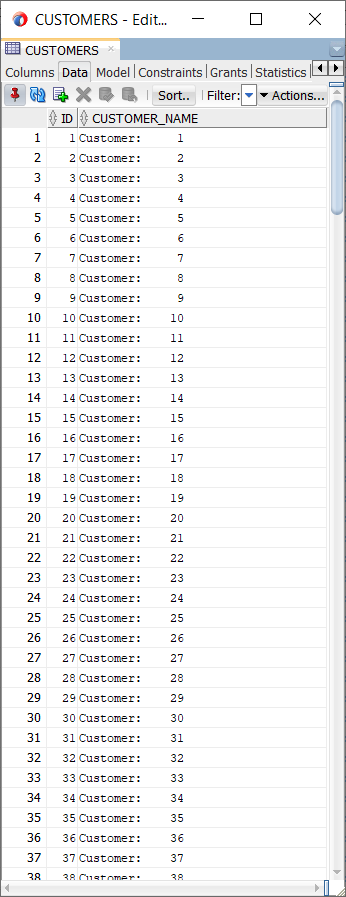


Таблица invoices

Создание

CREATE TABLE "DWH"."INVOICES"

( "INVOICE\_ID" NUMBER(\*,0) GENERATED ALWAYS AS IDENTITY MINVALUE 1 MAXVALUE 9999999999999999999999999999 INCREMENT BY 1 START WITH 1 CACHE 20 NOORDER NOCYCLE NOKEEP NOSCALE NOT NULL ENABLE,

"CUSTOMER\_ID" NUMBER(\*,0) NOT NULL ENABLE,

"INVOICE\_DATE" DATE NOT NULL ENABLE

) ;

Добавление записей

INSERT INTO invoices ( customer\_id, invoice\_date ) SELECT c.id, TO\_DATE('2023-03-01', 'YYYY-MM-DD') FROM customers c;

INSERT INTO invoices ( customer\_id, invoice\_date ) SELECT c.id, TO\_DATE('2023-03-10', 'YYYY-MM-DD') FROM customers c;

INSERT INTO invoices ( customer\_id, invoice\_date ) SELECT c.id, TO\_DATE('2023-04-02', 'YYYY-MM-DD') FROM customers c;

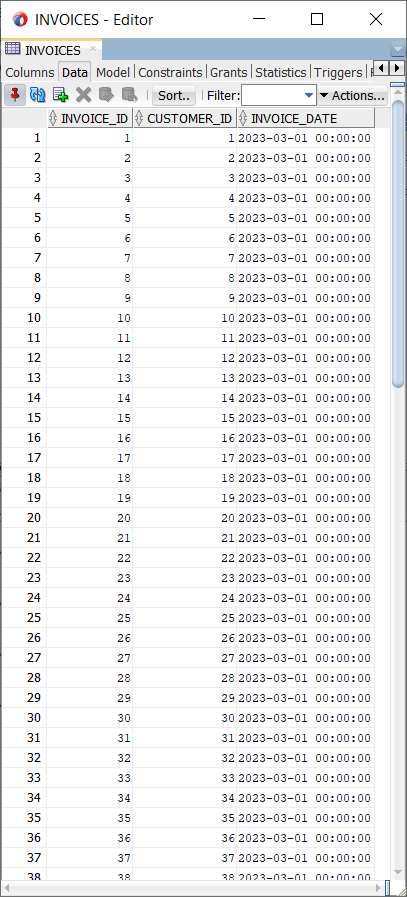
INSERT INTO invoices ( customer\_id, invoice\_date ) SELECT c.id, TO\_DATE('2023-04-12', 'YYYY-MM-DD') FROM customers c;

INSERT INTO invoices ( customer\_id, invoice\_date ) VALUES (1, TO\_DATE('2023-03-15', 'YYYY-MM-DD'));

INSERT INTO invoices ( customer\_id, invoice\_date ) VALUES (2, TO\_DATE('2023-03-15', 'YYYY-MM-DD'));

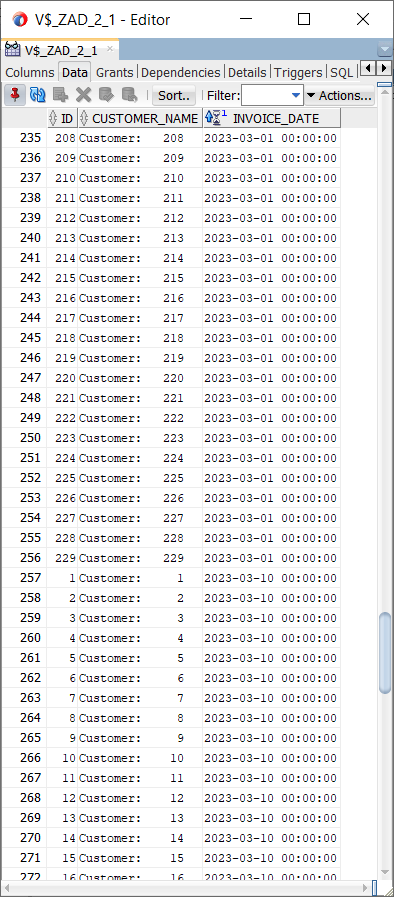
COMMIT;

Просмотр записей



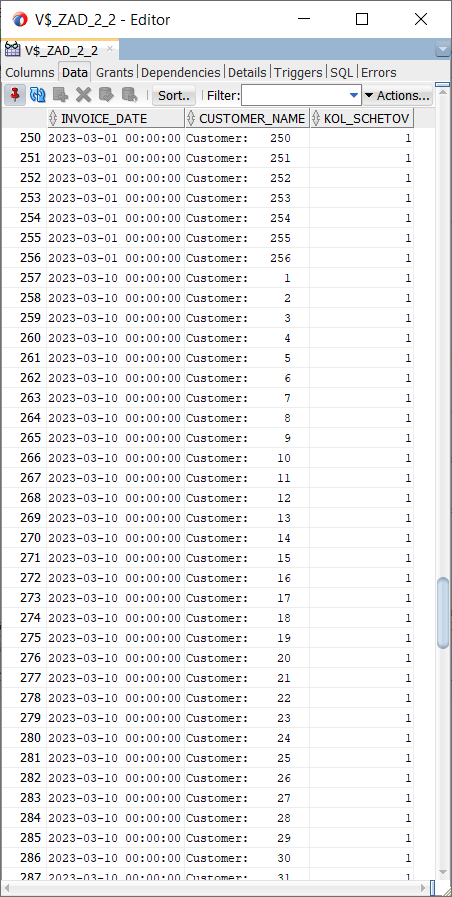
Вид v$\_zad\_2\_1

CREATE OR REPLACE VIEW v$\_zad\_2\_1 AS SELECT id, customer\_name, invoice\_date FROM ( SELECT c.id, c.customer\_name, i.invoice\_date FROM customers c, invoices i WHERE c.id = i.customer\_id ) ;



Вид v$\_zad\_2\_2

CREATE OR REPLACE VIEW v$\_zad\_2\_2 AS SELECT invoice\_date, customer\_name, COUNT(id) AS kol\_schetov FROM v$\_zad\_2\_1 GROUP BY invoice\_date, customer\_name ORDER BY invoice\_date, customer\_name ;



Процедура pzad2

CREATE OR REPLACE PROCEDURE pzad\_2 AS

---

TYPE CUSTOMER\_T IS RECORD (id INTEGER, name NVARCHAR2(400) ) ;

TYPE CUSTOMER\_ARRAY\_T IS TABLE OF CUSTOMER\_T INDEX BY pls\_integer ;

customer\_tek CUSTOMER\_T ;

customer\_array CUSTOMER\_ARRAY\_T := CUSTOMER\_ARRAY\_T() ;

---

TYPE MASSIV\_SHIRIN\_STOLBTSOV\_T IS TABLE OF INTEGER INDEX BY pls\_integer ;

shirina\_stolbtsa\_1 INTEGER ;

shirina\_stolbtsa\_tek INTEGER ;

dobavka\_stroki\_schetov\_tek VARCHAR2(400) ;

massiv\_shirin\_stolbtsov MASSIV\_SHIRIN\_STOLBTSOV\_T := MASSIV\_SHIRIN\_STOLBTSOV\_T() ;

---

TYPE VREM\_PERIOD\_ARR\_T IS TABLE OF DATE INDEX BY pls\_integer ;

vrem\_period\_tek DATE ;

vrem\_period\_array VREM\_PERIOD\_ARR\_T := VREM\_PERIOD\_ARR\_T() ;

---

kol\_klientov INTEGER ;

kol\_periodov\_vremeni INTEGER ;

CURSOR klienty\_cur IS SELECT DISTINCT id, customer\_name FROM v$\_zad\_2\_1 ORDER BY customer\_name ;

rec\_klient klienty\_cur%ROWTYPE ;

CURSOR vrem\_periody\_cur IS SELECT DISTINCT invoice\_date FROM invoices ORDER BY invoice\_date ;

rec\_vrem\_period vrem\_periody\_cur%ROWTYPE ;

---

CURSOR period\_klient\_cursor IS SELECT invoice\_date, customer\_name, kol\_schetov FROM v$\_zad\_2\_2 ;

rec\_period\_klient period\_klient\_cursor%ROWTYPE ;

--- id, customer\_name, invoice\_date

period\_tek DATE ;

kol\_schetov\_tek INTEGER ;

imya\_klienta\_tek NVARCHAR2(400) ;

customer\_name\_tek NVARCHAR2(400) ;

---

stroka\_s\_imenami\_kompanij VARCHAR2(32000) := '' ;

prefiks\_shapki\_tablitsy VARCHAR2(32000) := '' ;

postfiks\_shapki\_tablitsy VARCHAR2(32000) := '' ;

metka\_stolbtsa\_1 VARCHAR2(32000) := '' ;

stroka\_s\_kol\_schetov VARCHAR2(32000) := '' ;

stroka\_s\_kol\_schetov\_iz\_odnoj\_daty VARCHAR2(4000) := '' ;

dlina\_vyravnivaniya\_stolbtsa\_tek INTEGER ;

---

i INTEGER ;

j INTEGER ;

---

data\_par DATE ;

data\_pred DATE ;

data\_tek DATE ;

imya\_kompanii\_par NVARCHAR2(400) ;

CURSOR zapros\_s\_filtrom\_po\_date\_i\_imeni\_kompanii\_cursor ( data\_par DATE, imya\_kompanii\_par NVARCHAR2 ) IS SELECT invoice\_date, kol\_schetov FROM v$\_zad\_2\_2 WHERE invoice\_date = data\_par AND customer\_name = imya\_kompanii\_par ;

rec\_zapros\_s\_filtrom\_po\_date\_i\_imeni\_kompanii zapros\_s\_filtrom\_po\_date\_i\_imeni\_kompanii\_cursor%ROWTYPE ;

---

predstavlenie\_scheta VARCHAR2(16) ;

---

BEGIN

---

DBMS\_OUTPUT.ENABLE(300000) ;

---

SELECT COUNT(\*) INTO kol\_klientov FROM ( SELECT DISTINCT id FROM v$\_zad\_2\_1 ) ;

dbms\_output.put\_line( 'Количество клиентов: ' || kol\_klientov ) ;

SELECT COUNT(\*) INTO kol\_periodov\_vremeni FROM ( SELECT DISTINCT invoice\_date FROM v$\_zad\_2\_1 ) ;

dbms\_output.put\_line( 'Количество периодов времени: ' || kol\_periodov\_vremeni ) ;

OPEN klienty\_cur ;

OPEN vrem\_periody\_cur ;

OPEN period\_klient\_cursor ;

/\*

Заполнение массива ширин столбцов.

\*/

metka\_stolbtsa\_1 := 'InvoiceMonth' ;

shirina\_stolbtsa\_1 := GREATEST( LENGTH( '2023-03-01 00:00:00' ), LENGTH( metka\_stolbtsa\_1 ) ) ;

stroka\_s\_imenami\_kompanij := '|' || CHR(32) || RPAD ( metka\_stolbtsa\_1, shirina\_stolbtsa\_1, CHR(32) ) || CHR(32) || '|' ;

prefiks\_shapki\_tablitsy := '+' || '-' || RPAD( '-', shirina\_stolbtsa\_1, '-' ) || '-' || '+' ;

---

i := 1 ;

LOOP

FETCH klienty\_cur INTO rec\_klient ;

EXIT WHEN klienty\_cur%NOTFOUND ;

customer\_array(i).id := rec\_klient.id ;

customer\_array(i).name := rec\_klient.customer\_name ;

massiv\_shirin\_stolbtsov(i) := LENGTH( rec\_klient.customer\_name ) ;

stroka\_s\_imenami\_kompanij := stroka\_s\_imenami\_kompanij || CHR(32) || customer\_array(i).name || CHR(32) || '|' ;

shirina\_stolbtsa\_tek := LENGTH( customer\_array(i).name ) ;

prefiks\_shapki\_tablitsy := prefiks\_shapki\_tablitsy || '-' || RPAD( '-', shirina\_stolbtsa\_tek, '-' ) || '-' || '+' ;

i := i + 1 ;

END LOOP;

postfiks\_shapki\_tablitsy := prefiks\_shapki\_tablitsy ;

dbms\_output.put\_line( prefiks\_shapki\_tablitsy );

dbms\_output.put\_line( stroka\_s\_imenami\_kompanij ) ;

dbms\_output.put\_line( postfiks\_shapki\_tablitsy ) ;

---

i := 1 ;

LOOP

FETCH vrem\_periody\_cur INTO rec\_vrem\_period ;

EXIT WHEN vrem\_periody\_cur%NOTFOUND ;

period\_tek := rec\_vrem\_period.invoice\_date ;

vrem\_period\_array(i) := period\_tek ;

i := i+1 ;

END LOOP ;

stroka\_s\_kol\_schetov := '' ;

data\_pred := TO\_DATE('1999-12-31', 'YYYY-MM-DD') ;

data\_tek := TO\_DATE('1999-12-31', 'YYYY-MM-DD') ;

FOR i IN 1..vrem\_period\_array.count LOOP

--- Новая строка с новой датой и количеством счетов.

stroka\_s\_kol\_schetov\_iz\_odnoj\_daty := '|' || CHR(32) || vrem\_period\_array(i) || CHR(32) || '|' ;

stroka\_s\_kol\_schetov := stroka\_s\_kol\_schetov || stroka\_s\_kol\_schetov\_iz\_odnoj\_daty ;

FOR j IN 1..customer\_array.count LOOP

data\_tek := vrem\_period\_array(i) ;

kol\_schetov\_tek := FZAD2\_3\_NAJTI\_KOL\_SCHETOV ( data\_tek, customer\_array(j).name ) ;

IF kol\_schetov\_tek != -1

THEN

predstavlenie\_scheta := TO\_CHAR( kol\_schetov\_tek, '999' ) ;

ELSE

predstavlenie\_scheta := 'н/д' ;

END IF;

IF data\_tek = data\_pred

THEN

shirina\_stolbtsa\_tek := LENGTH( customer\_array(j).name ) ;

dobavka\_stroki\_schetov\_tek := RPAD( predstavlenie\_scheta, shirina\_stolbtsa\_tek, ' ' ) ;

dlina\_vyravnivaniya\_stolbtsa\_tek := 15 ;

dlina\_vyravnivaniya\_stolbtsa\_tek := massiv\_shirin\_stolbtsov(j) ;

stroka\_s\_kol\_schetov := stroka\_s\_kol\_schetov || CHR(32) || LPAD( predstavlenie\_scheta, massiv\_shirin\_stolbtsov(j), ' ' ) || CHR(32) || '|' ;

ELSE

stroka\_s\_kol\_schetov := stroka\_s\_kol\_schetov\_iz\_odnoj\_daty ;

stroka\_s\_kol\_schetov := stroka\_s\_kol\_schetov || CHR(32) || LPAD( predstavlenie\_scheta, 15, ' ' ) || CHR(32) || '|' ;

END IF ;

data\_pred := data\_tek ;

END LOOP; --- по j.

dbms\_output.put\_line( stroka\_s\_kol\_schetov ) ;

END LOOP; --- по i.

---

dbms\_output.put\_line( postfiks\_shapki\_tablitsy ) ;

---

END PZAD\_2 ;

Функция fzad2\_3\_najti\_kol\_schetov()

CREATE OR REPLACE FUNCTION fzad2\_3\_najti\_kol\_schetov ( data\_par DATE, imya\_kompanii\_par NVARCHAR2 ) RETURN INTEGER AS

---

kol\_schetov\_tek INTEGER ;

---

BEGIN

---

SELECT kol\_schetov INTO kol\_schetov\_tek FROM v$\_zad\_2\_2 WHERE invoice\_date = data\_par AND customer\_name = imya\_kompanii\_par ;

IF kol\_schetov\_tek IS NOT NULL

THEN

RETURN kol\_schetov\_tek ;

ELSE

RETURN -1;

END IF ;

---

EXCEPTION

WHEN no\_data\_found

THEN

RETURN -1;

---

END FZAD2\_3\_NAJTI\_KOL\_SCHETOV;

Результат работы

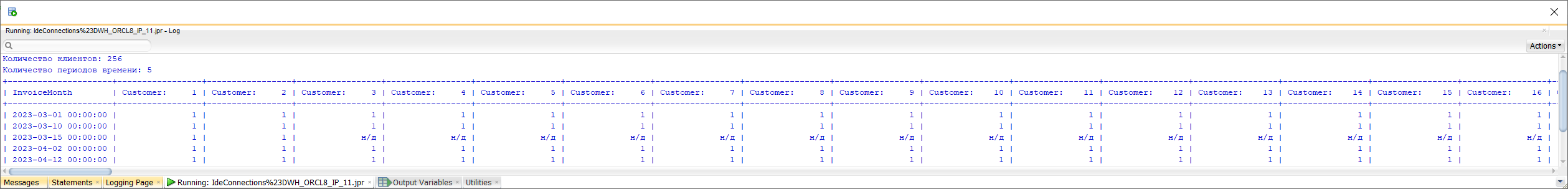


Рисунок 1. Левая начальная часть целевой таблицы

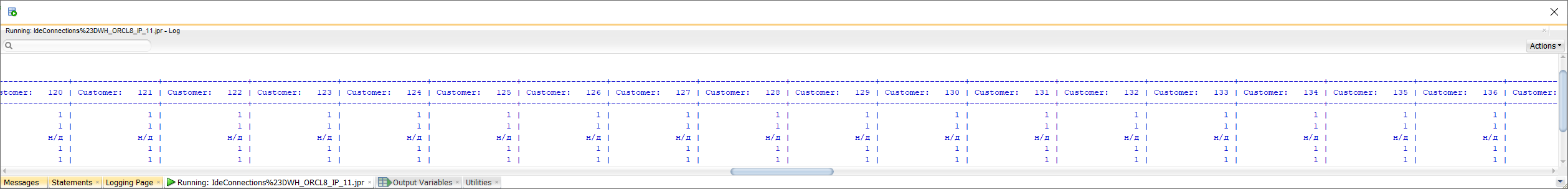


Рисунок 2. Срединная часть целевой таблицы

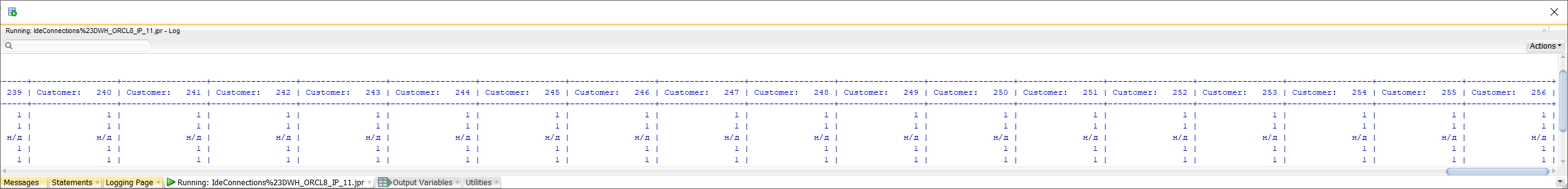


Рисунок 3. Правая конечная часть целевой таблицы

Тестовое задание №3

Постановка задачи

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

-- Задача № 3

-- Существуют 2 таблицы, одинаковые по структуре: test\_tb1 и test\_tb2

--

-- Необходимо разработать хранимую процедуру, которая принимает 3 параметра:

-- имя 1-й таблицы, имя 2-й таблицы, наименование уникального поля идентификатора (в данном случае id)

--

-- Данная процедура должна обновить все поля таблицы test\_tb1 новыми значениями аналогичных полей test\_tb2

-- соблюдая следующее условие:

-- значения NULL при обновлении игнорируются, в этом случае в test\_tb1 остается старое значение.

--

-- Пример вызова процедуры: p\_update\_t1\_from\_t2 'test\_tb1','test\_tb2','id'

--

-- Процедура должна обрабатывать любые аналогичные по структуре «пары» таблиц,

-- по переданным параметрам, независимо от количества полей.

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

drop table if exists #test\_tb1

drop table if exists #test\_tb2

create table #test\_tb1 (

id int

,a varchar(50)

,b varchar(50)

,c varchar(50)

,d varchar(50)

,e varchar(50)

)

create table #test\_tb2 (

id int

,a varchar(50)

,b varchar(50)

,c varchar(50)

,d varchar(50)

,e varchar(50)

)

;with t as (

select 1 id

,'tbl1\_a' a

,'tbl1\_b' b

,'tbl1\_c' c

,'tbl1\_d' d

,'tbl1\_e' e

union all

select id + 1,a, b, c, d, e from t

where id<50

)

insert into #test\_tb1

select id

,concat(a,id) a

,concat(b,id) b

,concat(c,id) c

,concat(d,id) d

,concat(e,id) e

from t

insert into #test\_tb2

select id

,iif(id<15 or id in (22,35,44),replace(a,'tbl1\_','tbl2\_'),null) a

,iif(id<10,null,replace(b,'tbl1\_','tbl2\_')) b

,iif(id<36,iif(right(c,1)=5,null,c),replace(c,'tbl1\_','tbl2\_')) c

,null d

,replace(e,'tbl1\_','tbl2\_') e

from #test\_tb1

select \* from #test\_tb1

select \* from #test\_tb2

Решение

Состав проекта:

1. таблицы test\_tb1, test\_db2;
2. процедура pzad3\_0\_zapusk;
3. функция fzad3\_1\_obnovit\_pole.

Таблица test\_db1

Создание

CREATE TABLE test\_tb1 ( id INTEGER, a VARCHAR(50), b VARCHAR(50), c VARCHAR(50), d VARCHAR(50), e VARCHAR(50) ) ;

Добавление записей

INSERT INTO test\_tb1( id, a, b, c, d, e )

SELECT id, CONCAT( a, id) a, CONCAT( b, id) b, CONCAT( c, id) c, CONCAT( d, id) d, CONCAT( e, id) e

FROM

(

WITH t( id, a, b, c, d, e ) AS ( SELECT 1 id, 'tbl1\_a' a, 'tbl1\_b' b, 'tbl1\_c' c, 'tbl1\_d' d, 'tbl1\_e' e FROM DUAL

UNION ALL

SELECT id + 1, a, b, c, d, e FROM t WHERE id < 50 )

SELECT id, a, b, c, d, e FROM t

)

;

Таблица test\_db2

Создание

CREATE TABLE test\_tb2 ( id INTEGER, a VARCHAR(50), b VARCHAR(50), c VARCHAR(50), d VARCHAR(50), e VARCHAR(50) ) ;

Добавление записей

INSERT INTO test\_tb2( id, a, b, c, d, e )

SELECT id, CONCAT( a, id) a, CONCAT( b, id) b, CONCAT( c, id) c, CONCAT( d, id) d, CONCAT( e, id) e

FROM

(

SELECT id

, CASE WHEN ( id < 15 OR id IN (22,35,44) ) THEN REPLACE( a, 'tbl1\_', 'tbl2\_' ) ELSE null END AS a

, CASE WHEN ( id < 10 ) THEN NULL ELSE REPLACE( b, 'tbl1\_', 'tbl2\_' ) END AS b

, CASE WHEN id < 36 THEN CASE WHEN id < 18 AND SUBSTR( c, -1 ) = 'c' THEN c ELSE replace( c,'tbl1\_c', 'tbl1\_x' ) END ELSE replace( c,'tbl1\_','tbl2\_') END AS c

, NULL d

, REPLACE( e, 'tbl1\_', 'tbl2\_') e

FROM test\_tb1

)

;

Просмотр и сопоставление записей обеих таблиц до обновления

|  |  |
| --- | --- |
| Рисунок 4. Записи таблицы test\_tb1 до обновления | Рисунок 5. Записи таблицы test\_tb2 |

Программы

Процедура pzad3\_0\_zapusk

CREATE OR REPLACE PROCEDURE PZAD3\_0\_ZAPUSK AS

---

rezultat INTEGER ;

---

BEGIN

rezultat := FZAD3\_1\_OBNOVIT\_POLE( 'test\_tb1', 'test\_tb2', 'id' );

---

END PZAD3\_0\_ZAPUSK;

Функция FZAD3\_1\_OBNOVIT\_POLE()

CREATE OR REPLACE FUNCTION FZAD3\_1\_OBNOVIT\_POLE ( imya\_tablitsy\_tseli\_par IN VARCHAR2, imya\_tablitsy\_iistochnika\_par IN VARCHAR2, imya\_unik\_polya\_par IN VARCHAR2 ) RETURN INTEGER AS

---

CURSOR polya\_tablits\_cursor( imya\_tablitsy\_iistochnika\_par IN VARCHAR2 ) IS SELECT column\_name FROM USER\_TAB\_COLUMNS WHERE UPPER( table\_name ) LIKE UPPER( imya\_tablitsy\_iistochnika\_par ) ;

rec\_pole\_tabltsy polya\_tablits\_cursor%ROWTYPE;

---

sql\_operator\_dlya\_obnovleniya VARCHAR2( 400 ) ;

sql\_operator\_dlya\_polucheniya\_nabora\_polej VARCHAR2( 400 ) ;

TYPE MASSIV\_STROK\_T IS TABLE OF VARCHAR2( 400 ) INDEX BY pls\_integer ;

massiv\_imen\_polej MASSIV\_STROK\_T := MASSIV\_STROK\_T() ;

---

i INTEGER ;

---

BEGIN

dbms\_output.put\_line( imya\_tablitsy\_tseli\_par || CHR(10) || imya\_tablitsy\_iistochnika\_par || CHR(10) || imya\_unik\_polya\_par );

/\*

Получаем список полей таблиц.

\*/

OPEN polya\_tablits\_cursor( imya\_tablitsy\_tseli\_par ) ;

i := 1 ;

LOOP

FETCH polya\_tablits\_cursor INTO rec\_pole\_tabltsy ;

EXIT WHEN polya\_tablits\_cursor%NOTFOUND;

massiv\_imen\_polej(i) := rec\_pole\_tabltsy.column\_name ;

--- dbms\_output.put\_line( i || CHR(9) || massiv\_imen\_polej(i) );

i := i + 1 ;

END LOOP;

CLOSE polya\_tablits\_cursor ;

FOR i IN 1..massiv\_imen\_polej.count LOOP

IF UPPER( massiv\_imen\_polej(i) ) != UPPER( imya\_unik\_polya\_par )

THEN

--- dbms\_output.put\_line( i || CHR(9) || massiv\_imen\_polej(i) || CHR(9) || imya\_unik\_polya\_par ) ;

sql\_operator\_dlya\_obnovleniya := 'UPDATE ' || imya\_tablitsy\_tseli\_par || ' x SET x.' || massiv\_imen\_polej(i) || ' = ( SELECT y.' || massiv\_imen\_polej(i) || ' FROM ' || imya\_tablitsy\_iistochnika\_par || ' y WHERE x.' || imya\_unik\_polya\_par || ' = y.' || imya\_unik\_polya\_par || ' AND y.' || imya\_unik\_polya\_par || ' IS NOT NULL)' ;

dbms\_output.put\_line( sql\_operator\_dlya\_obnovleniya );

EXECUTE IMMEDIATE sql\_operator\_dlya\_obnovleniya ;

EXECUTE IMMEDIATE 'COMMIT' ;

END IF ;

END LOOP; --- по i.

---

RETURN NULL;

END FZAD3\_1\_OBNOVIT\_POLE;

Результат работы

Видим, что все поля таблицы test\_tb1 были обновлены значениями полей таблицы test\_tb2 для соответствующих значений поля id.

|  |  |
| --- | --- |
| Рисунок 6. Записи таблицы test\_tb1 после обновления полями таблицы test\_tb2 | Рисунок 7. Записи таблицы test\_tb2, использованные для обновления полей таблицы test\_tb1 |

Заключение

В документе представлен отчет о выполнении трех тестовых заданий на вакансию «Data Engineer / Автоматизация и аналитика корпоративного бизнеса».

Все пункты заданий были выполнены.