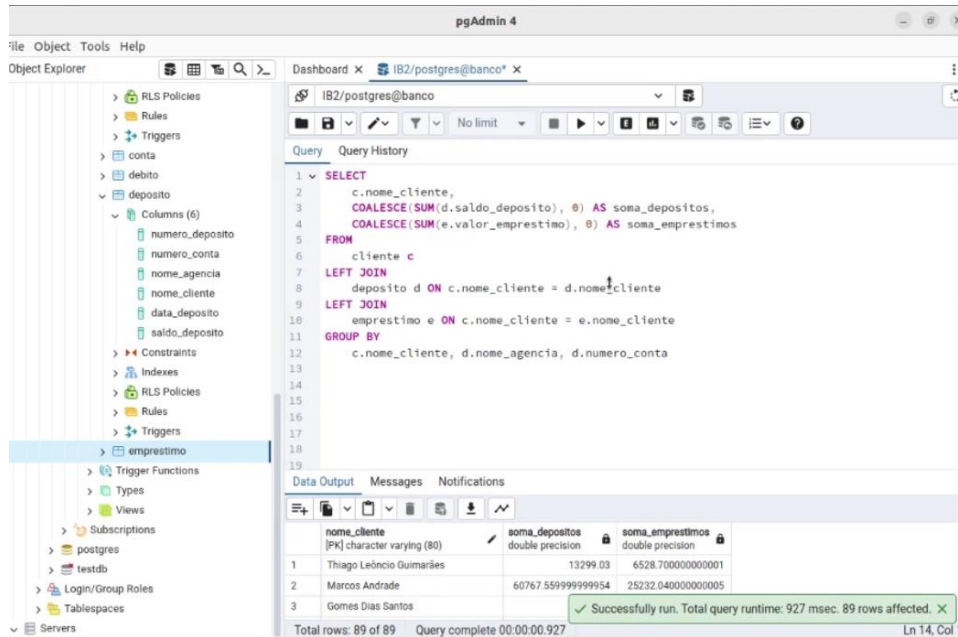


## T07-Recuperando e modificando dados

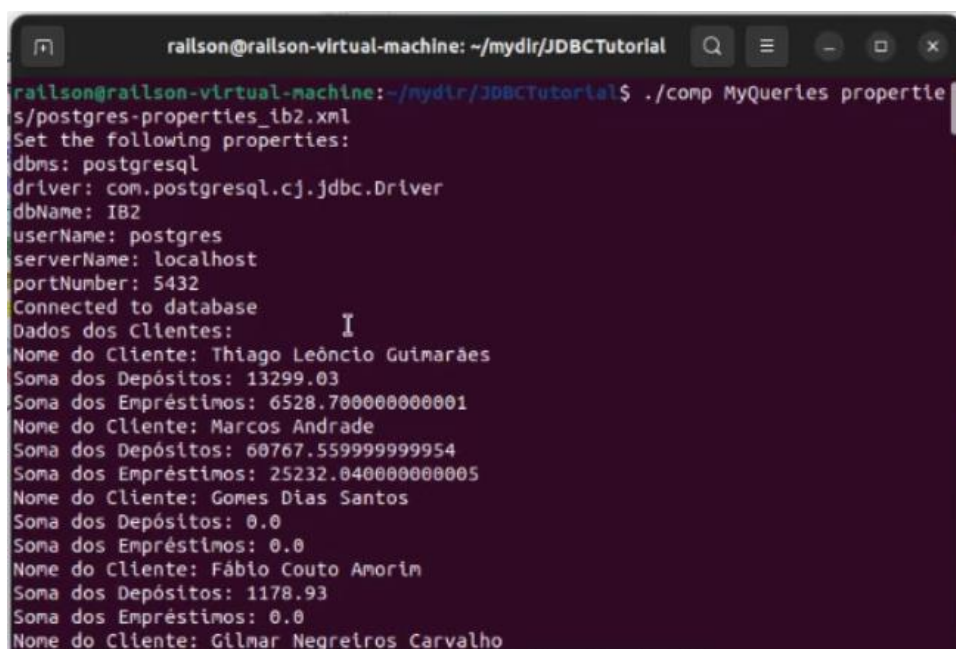
Railson Da Silva Martins - 11811BSI208

Wothon Mateus de Araujo 12111BSI262

Print referente ao passo 1: Executado a função criada sql



Print referente ao passo 2: Modificamos o arquivo MyQueries o print e da execução com o comando comp:



Print referente ao passo 3: Modifique o programa do item 2 para retornar os dados da consulta solicitada no item 1. Código fonte em um arquivo separado print da execução.

```
railson@railson-virtual-machine: ~/mydir/JDBCTutorial
railson@railson-virtual-machine:~/mydir/JDBCTutorial$ ./comp MyQueries properties/postgres-properties_lb2.xml
Set the following properties:
dbms: postgresql
driver: com.postgresql.cj.jdbc.Driver
dbName: IB2
userName: postgres
serverName: localhost
portNumber: 5432
Connected to database
Dados dos Clientes:

Usando índices numéricos

Nome do Cliente: Thiago Leônico Guimarães
Soma dos Depósitos: 13299.03
Soma dos Empréstimos: 6528.700000000001

Usando alias

Nome do Cliente: Thiago Leônico Guimarães
Soma dos Depósitos: 13299.03
Soma dos Empréstimos: 6528.700000000001

Usando nomes dos campos das tabelas alvo

Nome do Cliente: Thiago Leônico Guimarães
Soma dos Depósitos: 13299.03
Soma dos Empréstimos: 6528.700000000001

Usando índices numéricos

Nome do Cliente: Marcos Andrade
```

Print referente ao passo 4: Foi modificado novamente o arquivo MyQueries print da execução do novo MyQueries.

```
railson@railson-virtual-machine: ~/mydir/JDBCTutorial
Soma dos Empréstimos: 0.0

Usando alias

Nome do Cliente: Adilson de Oliveira
Soma dos Depósitos: 4832.31
Soma dos Empréstimos: 0.0

Usando nomes dos campos das tabelas alvo

Nome do Cliente: Adilson de Oliveira
Soma dos Depósitos: 4832.31
Soma dos Empréstimos: 0.0
ResultSet.HOLD_CURSORS_OVER_COMMIT = 1
ResultSet.CLOSE_CURSORS_AT_COMMIT = 2
Default cursor holdability: 1
Supports HOLD_CURSORS_OVER_COMMIT? true
Supports CLOSE_CURSORS_AT_COMMIT? true
Releasing all open resources ...
railson@railson-virtual-machine:~/mydir/JDBCTutorial$
```

Print referente ao passo 5:

```
railson@railson-virtual-machine: ~/mydir/JDBCTutorial

-----

ResultSet.HOLD_CURSORS_OVER_COMMIT = 1
ResultSet.CLOSE_CURSORS_AT_COMMIT = 2
Default cursor holdability: 1
Supports HOLD_CURSORS_OVER_COMMIT? true
Supports CLOSE_CURSORS_AT_COMMIT? true

-----

Suporta ResultSet.TYPE_FORWARD_ONLY com ResultSet.CONCUR_READ_ONLY? true
Suporta ResultSet.TYPE_FORWARD_ONLY com ResultSet.CONCUR_UPDATABLE? true
Suporta ResultSet.TYPE_SCROLL_INSENSITIVE com ResultSet.CONCUR_READ_ONLY? true
Suporta ResultSet.TYPE_SCROLL_INSENSITIVE com ResultSet.CONCUR_UPDATABLE? true
Suporta ResultSet.TYPE_SCROLL_SENSITIVE com ResultSet.CONCUR_READ_ONLY? false
Suporta ResultSet.TYPE_SCROLL_SENSITIVE com ResultSet.CONCUR_UPDATABLE? false

-----

Releasing all open resources ...
railson@railson-virtual-machine:~/mydir/JDBCTutorial$
```

Print referente ao passo 6: Print da análise feita e com o código ficou após a correção

```
1  /* 6 Análise de código
2
3  O erro no código ocorre na configuração do ResultSet.
4
5  stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_READ_ONLY);
6
7  Onde é especificado que ele é sensível a rolagem(ResultSet.TYPE_SCROLL_SENSITIVE) e somente
8  leitura(ResultSet.CONCUR_READ_ONLY).
9
10 No entanto, o método updateRow() é utilizado para atualizar as linhas do ResultSet dentro
11 do loop. Esse método só pode ser usado em ResultSet que é sensível a rolagem e atualizável.
12 Portanto, a correção necessária é definir o ResultSet como sensível a rolagem e atualizável
13 (ResultSet.CONCUR_UPDATABLE). Isso garantirá que o código possa atualizar as linhas do
14 ResultSet corretamente.
15
16 Ficando dessa forma:
17 */
18
19 public static void modifyPrices(Connection con) throws SQLException {
20     Statement stmt = null;
21     try {
22         stmt = con.createStatement(ResultSet.TYPE_SCROLL_SENSITIVE, ResultSet.CONCUR_UPDATABLE);
23         ResultSet uprs = stmt.executeQuery( "SELECT * FROM COFFEES");
24         while (uprs.next()) {
25             float f = uprs.getFloat("PRICE");
26             uprs.updateFloat("PRICE", f * 1.005);
27             uprs.updateRow();
28         }
29     } catch (SQLException e) {
30         JDBCTutorialUtilities.printSQLException(e);
31     } finally {
32         if (stmt != null) { stmt.close(); }
33     }
34 }
35
```

Print referente ao passo 7 e 8: Print com a opção de escolha o juro que deve ser aplicada aos débitos. Código entregue em um arquivo separado

```
railson@railson-virtual-machine: ~/mydir/JDBCTutorial
Suporta ResultSet.TYPE_SCROLL_SENSITIVE com ResultSet.CONCUR_UPDATABLE? false

-----

Entre com a porcentagem de juros que deseja aplicar (5% = entre com 1,05):
1

Realizado com sucesso

-----

Inserts Realizados

-----

Releasing all open resources ...
railson@railson-virtual-machine: ~/mydir/JDBCTutorial$ clear
```

Print referente ao passo 9: No print anterior a saída “Inserts Realizado” e reverente ao código solicitado no passo 9 sendo executado. Abaixo o print do código.

```
1 public static void populateTable(Connection con) throws SQLException, IOException {
2     Statement stmt = null;
3     BufferedReader reader = null;
4
5     try {
6         stmt = con.createStatement();
7         con.setAutoCommit(false); // desativa commit
8
9         stmt.executeUpdate("TRUNCATE TABLE debito");
10
11         reader = new BufferedReader(new FileReader("/home/railson/Downloads/debito-populate-table.txt"));
12         String line;
13         String create = "";
14
15         while ((line = reader.readLine()) != null) {
16
17             String[] values = line.split("\t");
18             int numero_debito = Integer.parseInt(values[0]);
19             double valor_debito = Double.parseDouble(values[1]);
20             int motivo_debito = Integer.parseInt(values[2]);
21             String data_debito = values[3];
22             int numero_conta = Integer.parseInt(values[4]);
23             String nome_agencia = values[5];
24             String nome_cliente = values[6];
25
26             String insertQuery = "INSERT INTO debito (numero_debito, valor_debito, motivo_debito, data_debito, nume"
27                                 + "VALUES (" + numero_debito + ", " + valor_debito + ", " + motivo_debito + ", " + data_debito + "
28                                 + numero_conta + ", " + nome_agencia + ", " + nome_cliente + ")";
29             create += insertQuery + ";\n";
30
31         }
32
33         stmt.executeUpdate(create);
34         con.commit(); // confirma
35     }
36 }
```

Print referente ao passo 10: Executado o código feito no passo 10 “Linhas incluídas”

```
railson@railson-virtual-machine: ~/mydir/JDBCTutorial
Entre com a porcentagem de juros que deseja aplicar (5% = entre com 1,05):
1

Realizado com sucesso

-----

Inserts Realizados

-----

Linhas incluídas

-----

Releasing all open resources ...
railson@railson-virtual-machine: ~/mydir/JDBCTutorial$
```

Print referente ao passo 10.1: Indicado que realmente a linhas foram incluídas com sucesso.

The image shows the pgAdmin 4 interface. On the left, the Object Explorer shows the database structure. The main pane displays a SQL query and its results.

Query:

```
1 select * from debito
2 where numero_debito in (2000, 2001, 2002)
```

Data Output:

	numero_debito [PK] integer	valor_debito double precision	motivo_debito smallint	data_debito date	numero_conta integer	nome_agencia character varying (50)	nome_cliente character varying
1	2000	150	1	2014-01-23	46248	UFU	Carla Soares Sou
2	2001	200	2	2014-01-23	26892	Glória	Carolina Soares t
3	2002	500	3	2014-01-23	70044	Cidade Jardim	Eurides Alves da

Successfully run. Total query runtime: 129 msec. 3 rows affected.

Total rows: 3 of 3 Query complete 00:00:00.129 Ln 2, Col 42