

Banco de Dados – Atividade Hands-on em Laboratório - 07

HSQldb - Driver JDBC - Prof. Dr. Aparecido Freitas

1. Introdução

Neste exercício, iremos repetir o código que faz inserção de dados no banco de dados, com o emprego de **PreparedStatement**.

Porém, iremos considerar agora uma situação no qual o programa (transação) irá inserir 2 registros no banco de dados.

Vamos implementar um método chamado **adiciona**(String, String, PreparedStatement) que irá ser chamado para inserir os produtos.

```
private static void adiciona(String nome, String descricao,
PreparedStatement stmt) throws SQLException {

    stmt.setString(1,nome);

    stmt.setString(2,descricao);

    boolean resultado = stmt.execute();

    System.out.println("resultado = " + resultado);

    ResultSet resultSet = stmt.getGeneratedKeys();

    while (resultSet.next() ) {

        Integer id = resultSet.getInt("id");
        System.out.println("ID = " + id + " gerado ...");
    }

    resultSet.close();
}
```

No nosso método main, o método será chamado da seguinte forma:

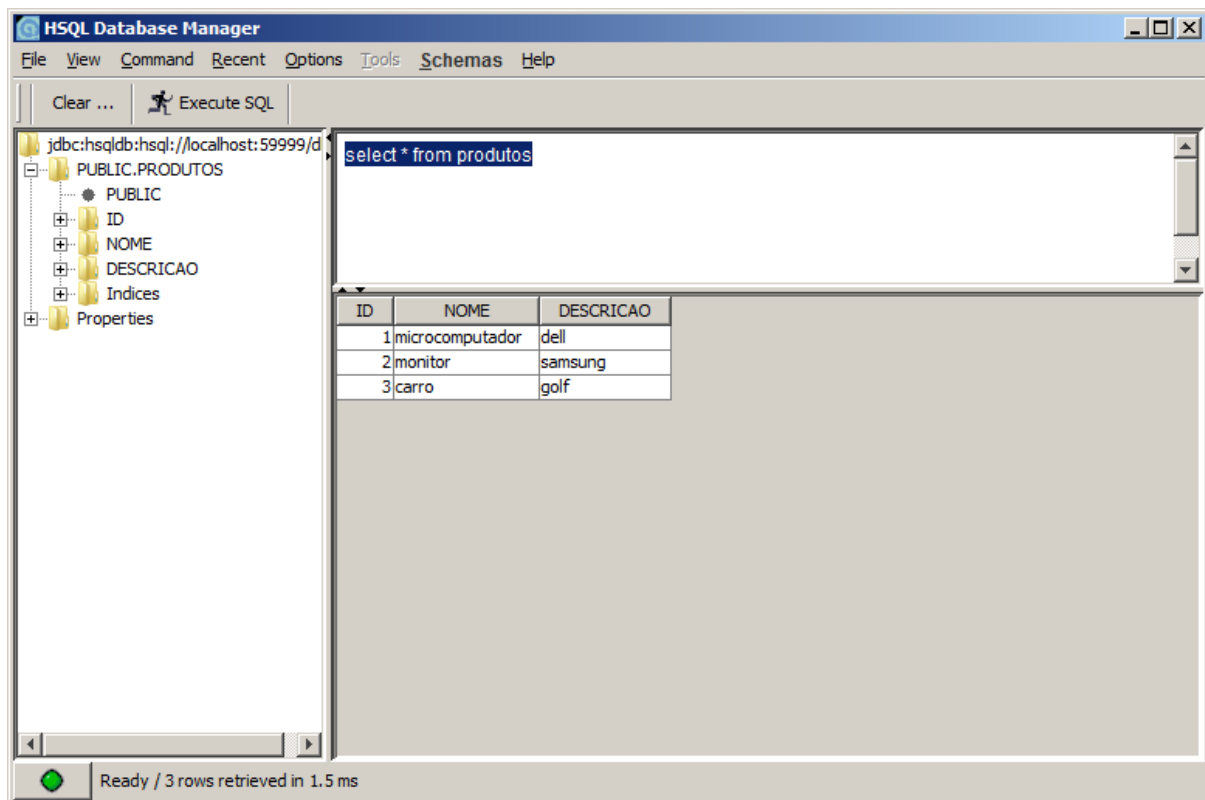
```
adiciona("carro", "VW GOLF", statement);
```

Para adicionar o segundo produto:

```
adiciona("lente", "Nikon 70-200", statement);
```

Vamos executar o código novamente, com o método **adiciona** sendo chamado 2 vezes para a inserção dos produtos acima: VW GOLF e Nikon 70-200.

Considerando que o banco de dados possui os seguintes produtos:



Após a execução do programa abaixo:

```
package br.maua;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class CommitRollback {

    public static void main(String[] args) throws SQLException {

        try {

            Connection connection =
DriverManager.getConnection("jdbc:hsqldb:hsq://localhost:59999/db"
, "SA" , null);

            System.out.println("conexao ao HSQLDB feita com
SUCESSO ! ");
```

```

        String sql = "INSERT INTO PRODUTOS (NOME,DESCRICAO)
VALUES (?,?)";

        PreparedStatement stmt =
connection.prepareStatement(sql,Statement.RETURN_GENERATED_KEYS );

        adiciona("carro", "VW GOLF", stmt);
        adiciona("lente", "Nikon 70-200", stmt);

        stmt.close();

    }

    catch (SQLException e) {

        System.out.println("Erro SQLException....");

    }

    catch ( Exception e) {
        System.out.println("Problemas na conexao ao
HSQLDB....");
    }

}

private static void adiciona(String nome, String descricao,
PreparedStatement stmt) throws SQLException {

    stmt.setString(1,nome);

    stmt.setString(2,descricao);

    boolean resultado = stmt.execute();

    System.out.println("resultado = " + resultado);

    ResultSet resultSet = stmt.getGeneratedKeys();

    while (resultSet.next() ) {

        Integer id = resultSet.getInt("id");
        System.out.println("ID = " + id + " gerado ...");

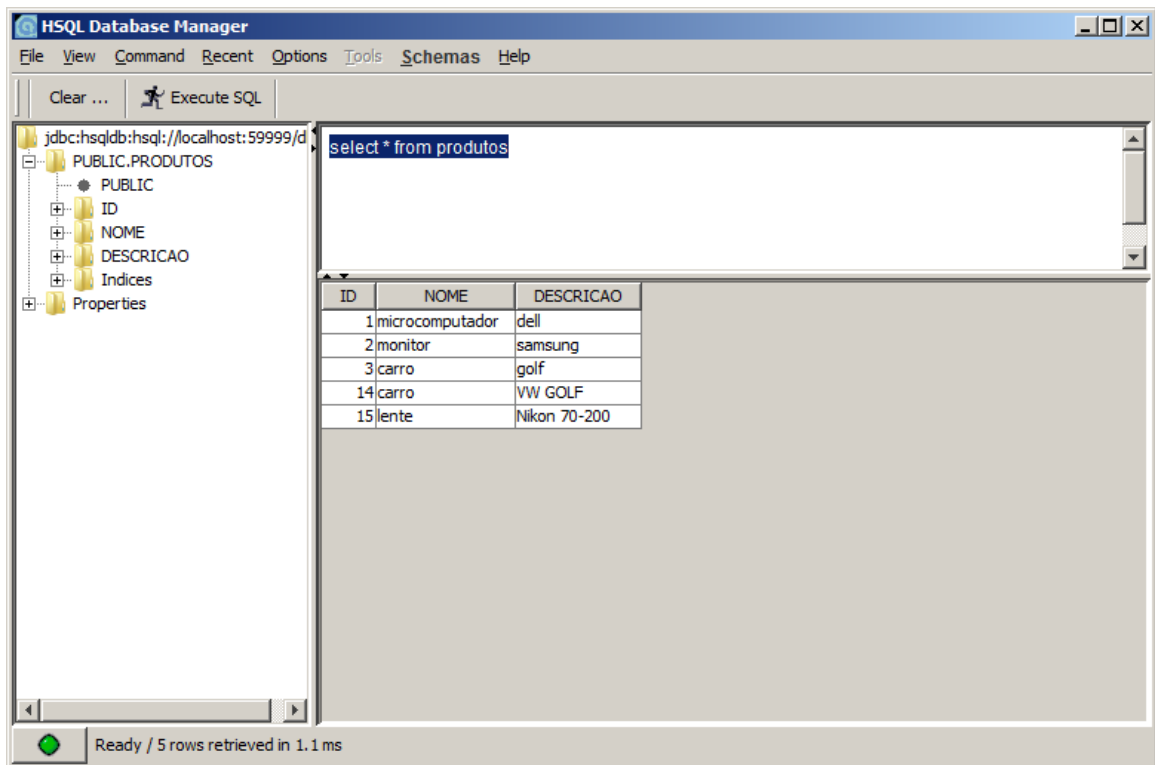
    }

    resultSet.close();

}

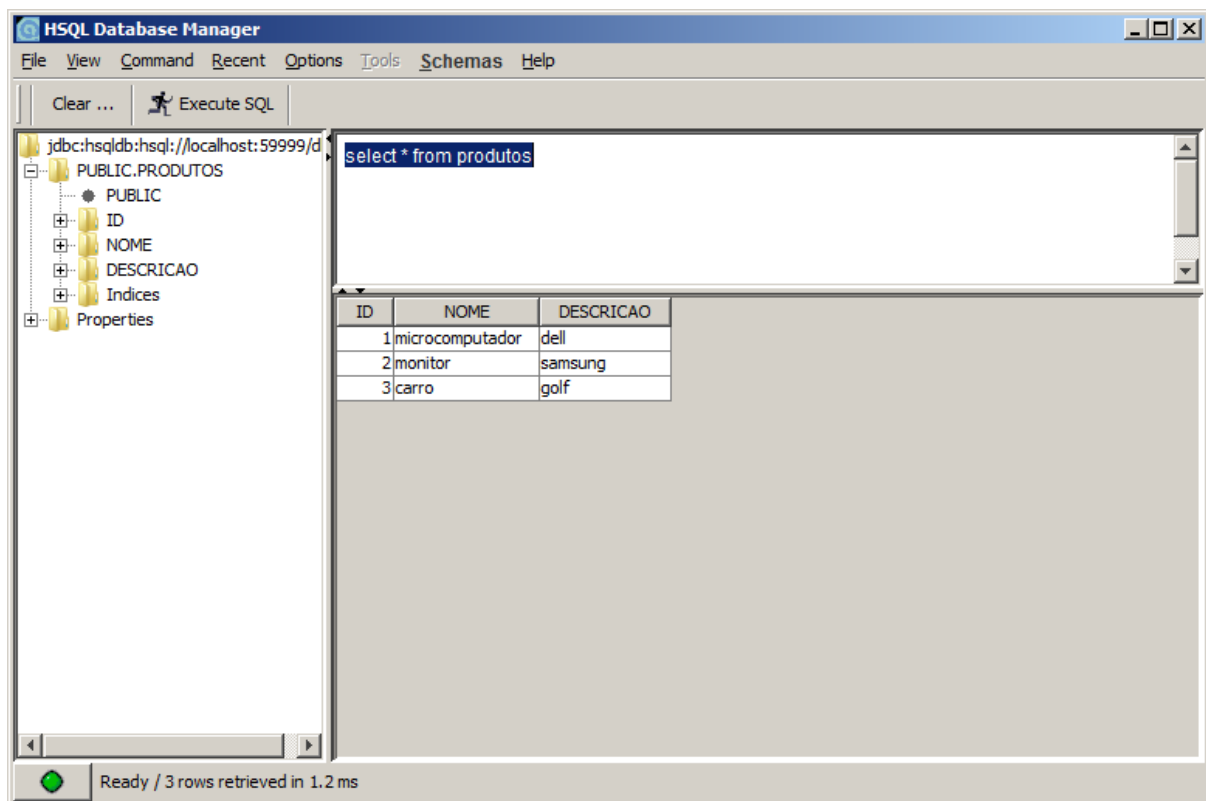
}

```



Vamos agora considerar a seguinte situação. Considere que algo tenha dado errado entre o primeiro e o segundo insert.

Vamos executar novamente o comando SQL: **delete from produtos where id > 3.**



Vamos forçar uma situação de erro após a execução do primeiro insert, por meio da adição de um if dentro do método adiciona para lançar-se uma exceção.

```
package br.maua;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class CommitRollback {

    public static void main(String[] args) throws SQLException {

        try {

            Connection connection =
DriverManager.getConnection("jdbc:hsqldb:hsql://localhost:59999/db"
, "SA" , null);

            System.out.println("conexao ao HSQLDB feita com
SUCESSO ! ");

            String sql = "INSERT INTO PRODUTOS (NOME,DESCRICA0)
VALUES (?,?)";

            PreparedStatement stmt =
connection.prepareStatement(sql,Statement.RETURN_GENERATED_KEYS );

            adiciona("carro", "VW GOLF", stmt);
            adiciona("lente", "Nikon 70-200", stmt);

            stmt.close();

        }

        catch (SQLException e) {

            System.out.println("Erro SQLException....");

        }

        catch (IllegalArgumentException e ) {
```

```

        System.out.println("Problema na insercao do
produto...");
    }

    catch ( Exception e) {
        System.out.println("Problemas na conexao ao
HSQLDB...");
    }

}

private static void adiciona(String nome, String descricao,
PreparedStatement stmt) throws SQLException {

    if (nome.equals("lente")) {

        throw new IllegalArgumentException("Problema com o
segundo insert... ");
    }

    stmt.setString(1,nome);

    stmt.setString(2,descricao);

    boolean resultado = stmt.execute();

    System.out.println("resultado = " + resultado);

    ResultSet resultSet = stmt.getGeneratedKeys();

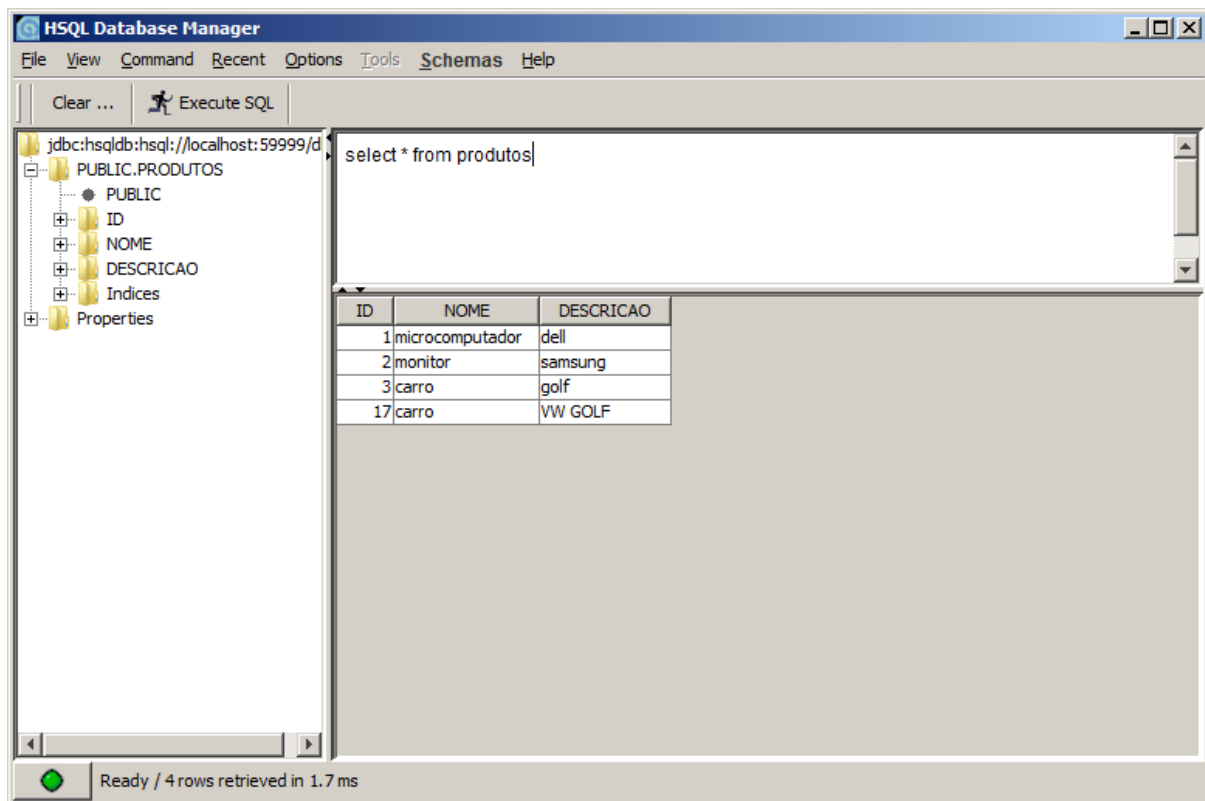
    while (resultSet.next() ) {

        Integer id = resultSet.getInt("id");
        System.out.println("ID = " + id + " gerado ...");
    }

    resultSet.close();
}
}

```

Considerando que o segundo insert apresentou problema, apenas o primeiro insert deve ter sido processado com sucesso. O banco de dados ficará:



O sistema gerenciador de banco de dados HSQLDB está considerando que cada insert processado deverá estar automaticamente associado a um commit. Isto ocorre, uma vez que por default a propriedade AutoCommit está sendo definida por true;

Isto é, para cada novo statement do tipo insert (update, delete e similares), o comando é comitado automaticamente para o servidor, não sendo permitido portanto, voltar-se para trás.

Este é o comportamento padrão de uma Connection segundo a especificação do JDBC.

Mas, é se quisermos executar os dois inserts ou nada? Isto é, tudo ou nada. Ou se executa os dois inserts com sucesso ou nenhum.

Nesse caso, precisa-se desativar o auto commit. Isso é feito, após a conexão, por meio de:

```
connection.setAutoCommit(false);
```

Vamos executar novamente o programa, com a definição de autoCommit para false:

```

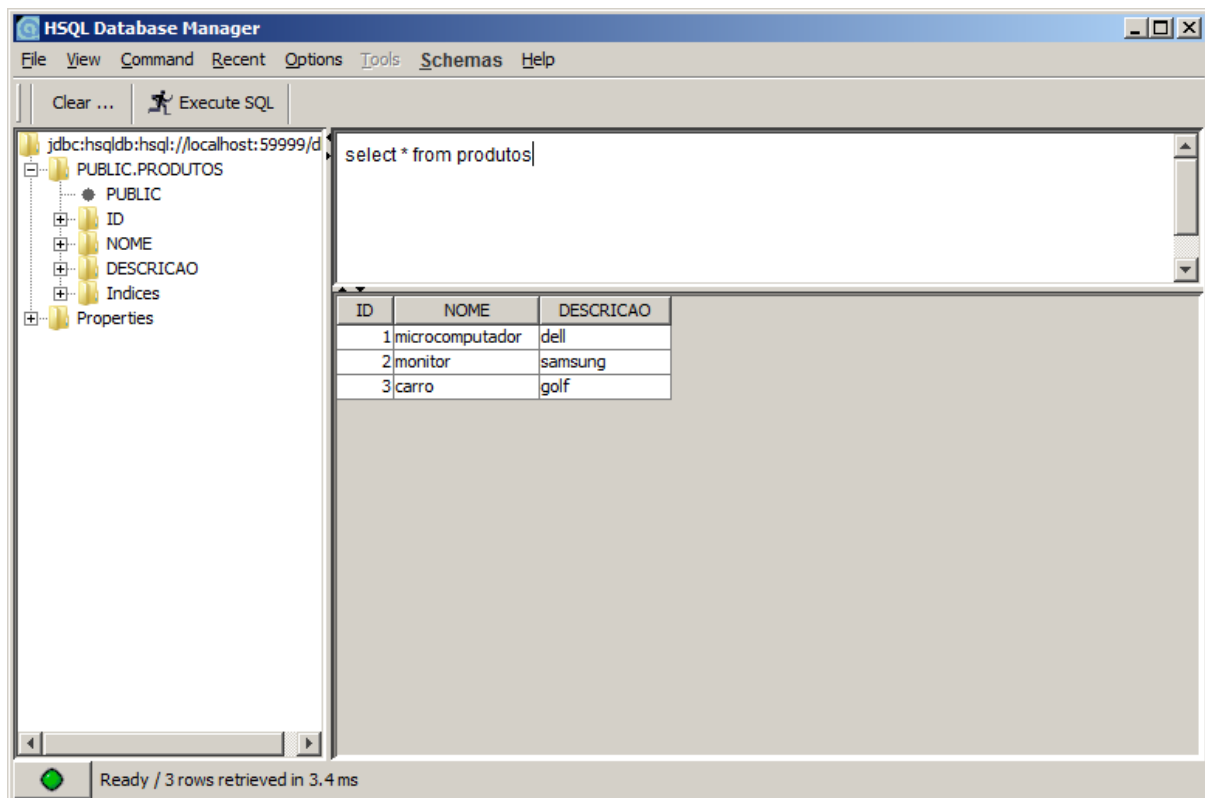
public class CommitRollback {
    public static void main(String[] args) throws SQLException {
        try {
            Connection connection =
DriverManager.getConnection("jdbc:hsqldb:hsqldb://localhost:59999/db"
, "SA" , null);
            connection.setAutoCommit(false);
            System.out.println("conexao ao HSQLDB feita com
SUCESSO ! ");
            String sql = "INSERT INTO PRODUTOS (NOME,DESCRICAO)
VALUES (?,?)";
            PreparedStatement stmt =
connection.prepareStatement(sql,Statement.RETURN_GENERATED_KEYS );

            adiciona("carro", "VW GOLF", stmt);
            adiciona("lente", "Nikon 70-200", stmt);

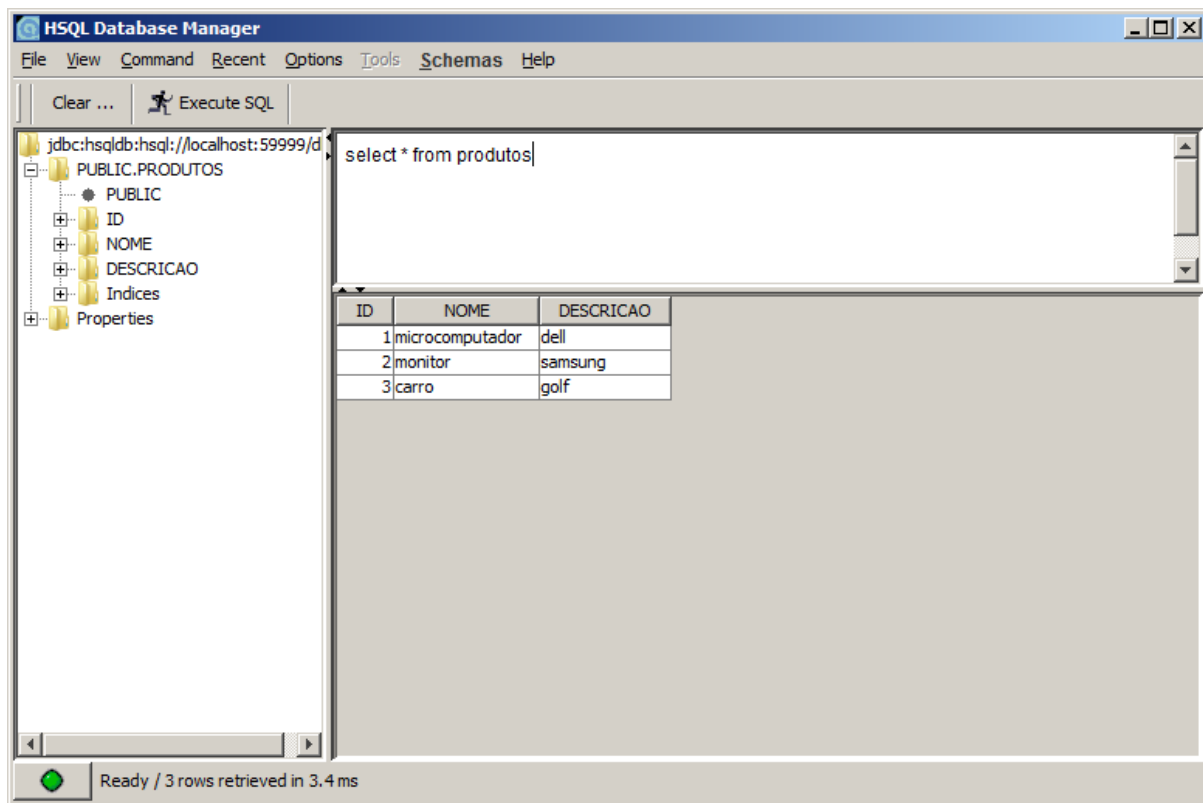
            stmt.close();
        }
    }
}

```

Vamos retornar o banco de dados para armazenar apenas 3 produtos, por meio do comando: **delete from produtos where id > 3**



Reexecutando-se o programa, verifica-se que em função do erro no segundo insert, **nada** foi inserido no banco de dados. Nem o carro e nem a lente.



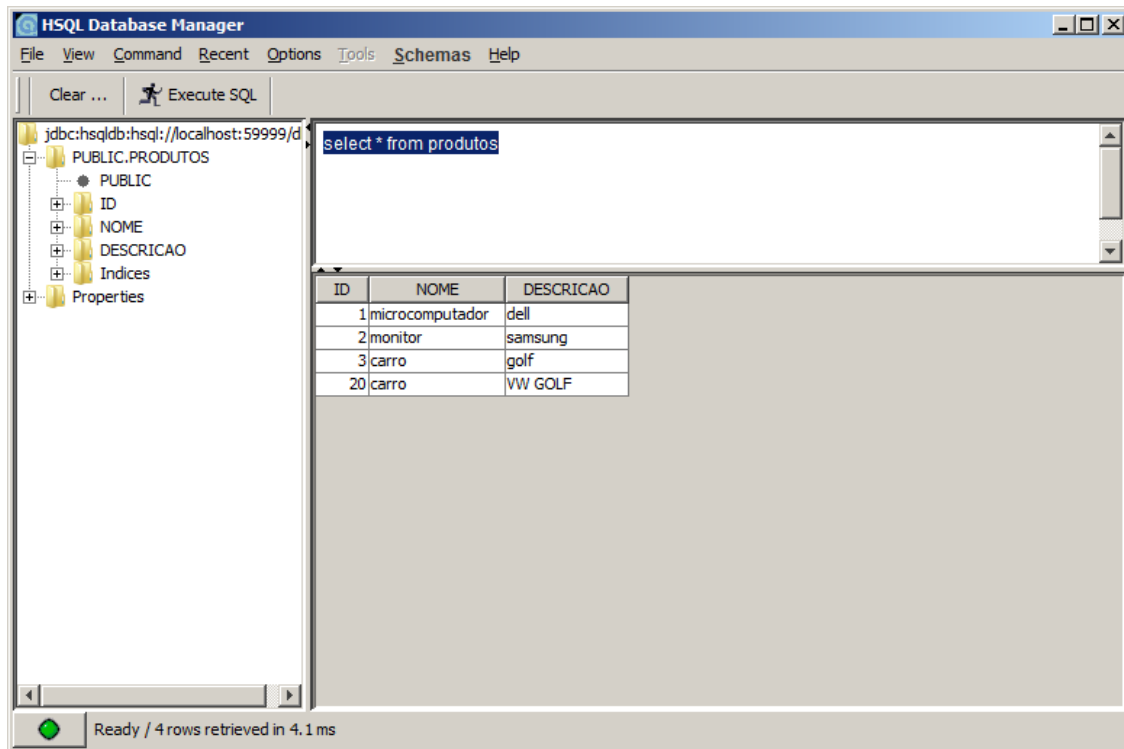
Quando o auto commit está configurado como false, precisamos indicar o momento de ser executado o commit.

Vamos alterar o código, incluindo o commit após cada chamada de insert:

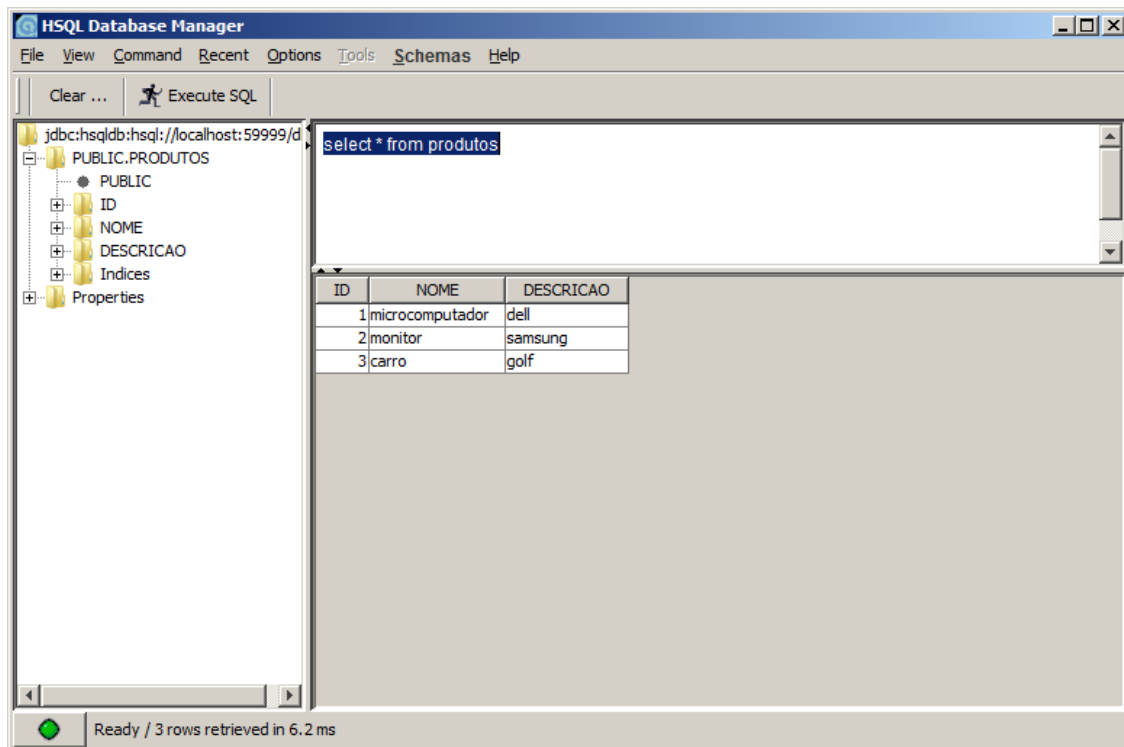
```
adiciona("carro", "VW GOLF", stmt);  
connection.commit();  
  
adiciona("lente", "Nikon 70-200", stmt);  
connection.commit();
```

Vamos reexecutar o código com as definições de commit, feitas conforme acima.

Verifica-se agora que o primeiro insert foi gravado no banco de dados, mas o segundo não.



Vamos agora, deletar novamente os produtos com id>3.



Vamos agora deixar explícito a maneira pela qual se decida voltar atrás em nossa decisão, ou seja fazer um rollback em caso de algum erro em nossa transação.

```
package br.maua;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class CommitRollback {

    public static void main(String[] args) throws SQLException {

        try {

            Connection connection =
DriverManager.getConnection("jdbc:hsqldb:hsql://localhost:59999/db"
, "SA" , null);

            connection.setAutoCommit(false);

            System.out.println("conexao ao HSQLDB feita com
SUCESSO ! ");

            String sql = "INSERT INTO PRODUTOS (NOME,DESCRICA0)
VALUES (?,?)";

            try {
                PreparedStatement stmt =
connection.prepareStatement(sql,Statement.RETURN_GENERATED_KEYS );

                adiciona("carro", "VW GOLF", stmt);

                adiciona("lente", "Nikon 70-200", stmt);
                connection.commit();

            }

            catch (SQLException e) {

                System.out.println("Erro SQLException....");
                connection.rollback();

            }

        }

    }

}
```

```

        catch (IllegalArgumentException e ) {
            System.out.println("Problema na insercao do
produto...");
            connection.rollback();
        }
    }

    catch ( Exception e) {
        System.out.println("Problemas na conexao ao
HSQLDB....");
    }
}

private static void adiciona(String nome, String descricao,
PreparedStatement stmt) throws SQLException {

    if (nome.equals("lente")) {

        throw new IllegalArgumentException("Problema com o
segundo insert.... ");
    }

    stmt.setString(1,nome);

    stmt.setString(2,descricao);

    boolean resultado = stmt.execute();

    System.out.println("resultado = " + resultado);

    ResultSet resultSet = stmt.getGeneratedKeys();

    while (resultSet.next() ) {

        Integer id = resultSet.getInt("id");
        System.out.println("ID = " + id + " gerado ...");
    }

    resultSet.close();
}
}

```

Agora, embora o primeiro insert tenha sido executado com sucesso, considerando que o segundo insert resultou em erro, o estado do banco de dados foi retornado à condição inicial uma vez que o rollback foi executado.

Ou seja, nada foi gravado no banco de dados.

