# Construção de Sistemas de Software

# SalesSys Gateways — RDGW e TDGW



CSS (2019/2020) Caderno de Exercício

#### Padrões para organização dos dados

- 1. Suponha que se pretende desenhar a camada de acesso aos dados do SalesSys recorrendo ao padrão Row Data Gateway e que essa camada vai ser responsável por realizar a comunicação da aplicação com o SGBD escolhido — Derby — através de um conector JDBC. Adicionalmente deve ter em conta que se pretende usar esta camada por baixo dos transaction scripts concebidos anteriormente (folha 1).
  - (a) Apresente um diagrama que explique a organização em camadas desta solução do SalesSys.
  - (b) Que vantagens temos por ter todo o código responsável pelo acesso à base de dados numa camada separada?
  - (c) Explique em que consiste o padrão Row Data Gateway focando a sua explicação na forma como é feita a abstração da base de dados e nas responsabilidades atribuídas a cada objeto do padrão.
  - (d) Que vantagens advém de ter o código responsável pelas pesquisas numa tabela numa classe separada?
  - (e) Identifique as classes  ${\it gateway}$  de que vai precisar, os seus atributos e métodos.
  - (f) Desenhe uma classe DataSource com uma única instância que abstrai e encapsula a ligação à base de dados (que se considera ser a fonte única de dados).
  - (g) Tendo em conta os atributos definidos para a classe CustomerRowDGW. desenhe o comportamento do método <code>insert()</code>. Identifique as constantes que devem ser definidas nesta classe para confinar ao máximo as dependências do modelo físico de dados.
  - $(h) \ \ Complete \ a \ classe \ {\tt CustomerRowDGW} \ desenhando \ o \ comportamento$ dos métodos de classe
    - getCustomerByVATNumber(int vat):CustomerRowDGW • getCustomerById(int saleId):CustomerRowDGW

  - (i) Defina os atributos da classe  ${\tt SaleRowDGW}$  e desenhe o comportamento do método  ${\tt insert()}$ . Identifique as constantes que devem

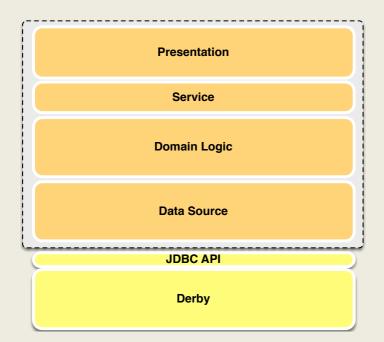
- ser definidas nesta classe para confinar ao máximo as dependências do modelo físico de dados
- (j) Complete a classe SaleRowDGW desenhando o comportamento do método de classe getSaleById(int saleId):SaleRowDGW.
- $\begin{tabular}{ll} (k) Defina os atributos da classe {\tt SaleProductRowDGW} e desenhe \end{tabular}$ comportamento do método getProductSalesBySaleId(int id): Iterable<SaleProductRowDGW>. Identifique as constantes que devem ser definidas nesta classe para confinar ao máximo as dependências do modelo físico de dados
- (l) Repita as tarefas anteriores para as restantes classes da camada de acesso aos dados.
- (m) Suponha que quer combinar a camada de negócio organizada seguindo o *Transaction Script* desenhada no exercício 2 da folha 1 com a camada de acesso aos dados do exercício anterior. Identifique, no desenho da operação addProductToSale, que alterações necessárias e que serviços adicionais é preciso que a classe DataSource forneça.
- 2. Suponha que se pretende desenhar a camada de acesso aos dados do SalesSys recorrendo ao padrão Table Data Gateway e que essa camada vai ser responsável por realizar a comunicação da aplicação com o SGBD escolhido — Derby — através de um conector JDBC. Apesar de existir atualmente uma única fonte de dados, o desenho da camada deve prever a possibilidade de serem usadas várias fontes de dados. Adicionalmente o desenho da camada deve ter em conta que se pretende usar esta camada por baixo dos  $table\ modules$  concebidos anteriormente
  - (a) Explique em que consiste o padrão Table Data Gateway focando a sua explicação na forma como é feita a abstração da base de dados e nas responsabilidades atribuídas a cada objeto do padrão.
  - (b) Identifique as classes gateway de que vai precisar, os seus atributos e métodos.
  - (c) Desenhe uma classe *Persistence* cujos objetos abstraem formas de realizar a persistência e as respetivas fontes dos dados, e que são responsáveis por criar os gateways para as tabelas e suporte a execução de transações. Sugestão: Adapte a classe *DataSource* definida no exercício 1 de forma a poder ter várias instâncias.
  - (d) Defina os atributos da classe CustomerTableDGW e desenhe o comportamento dos métodos

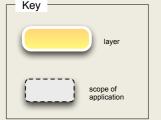
Suponha que se pretende desenhar a camada de acesso aos dados do SalesSys recorrendo ao padrão *Row Data Gateway* e que essa camada vai ser responsável por realizar a comunicação da aplicação com o SGBD escolhido — Derby — através de um conector JDBC. Adicionalmente deve ter em conta que se pretende usar esta camada por baixo dos *transaction scripts* concebidos anteriormente (folha 1).

(a) Apresente um diagrama que explique a organização em camadas desta solução do SalesSys.



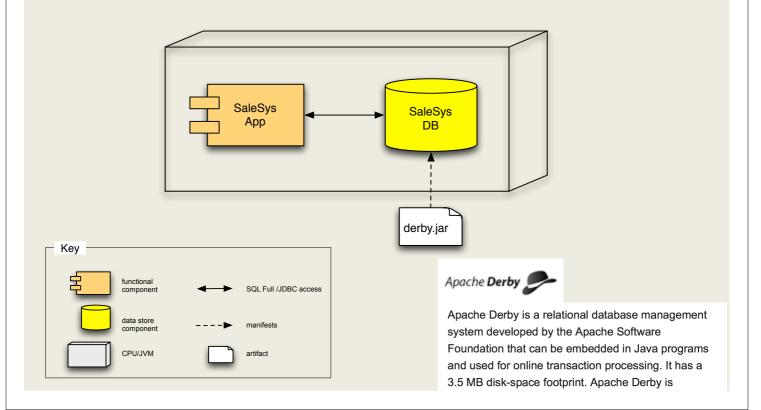
# SalesSysApp: Organização em camadas (do código)







# SalesSys: Arquitetura (BD embebida)

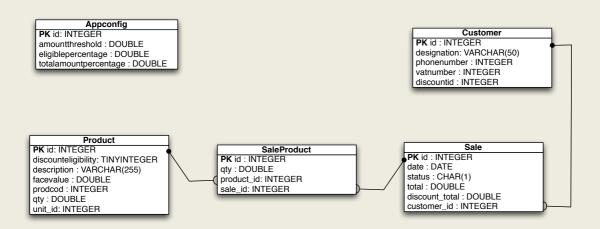


## **Exercícios**

- (b) Que vantagens temos por ter todo o código responsável pelo acesso à base de dados numa camada separada?
- O código que depende do JDBC (e q envolve SQL), do esquema da BD e escolha do SGDB está todo nesta camada, separado portanto do código que trata da lógica de negócio
- Facilita a divisão de trabalho
- Facilita o trabalho dos DBAs (afinar e fazer evoluir a base de dados)
- Facilita mudanças ao nível de como é feita a persistência (mudança de SGBD ou mesmo a forma como é feita a persistência)



- (c) Explique em que consiste o padrão *Row Data Gateway* focando a sua explicação na forma como é feita a abstração da base de dados e nas responsabilidades atribuídas a cada objeto do padrão.
- (d) Identifique as classes *gateway* de que vai precisar, os seus atributos e métodos.



### Camada de Acesso aos Dados com RDGW

#### Customer **CustomerRowDGW** PK id: INTEGER -id: int designation: VARCHAR(50) -vat:int phonenumber: INTEGER -designation:String vatnumber: INTEGER -phoneNumber:int discountid: INTEGER -discountId:int +getCustomerById(int):CustomerRowDGW +getCustomerByVATNumber(int): CustomerRowDGW +CustomerRowDGW(int,String,int,DiscountType) +insert() +getCustomerId():int +getDesignation():String +getDiscountType():DiscountType +getPhoneNumber():int +getVAT():int +setDesignation(String) +setDiscountType(DiscountType) +setPhoneNumber(int)



# Camada de Acesso aos Dados com RDGW

#### SaleProduct

PK id : INTEGER qty: DOUBLE product\_id: INTEGER sale\_id: INTEGER

#### Sale

PK id : INTEGER date : DATE status : CHAR(1) total: DOUBLE discount\_total : DOUBLE customer\_id: INTEGER

#### SaleProductRowDGW

- -saleId:int
- -productId: int
- -qty:int
- +getSaleProducts(int):Iterable<SaleProductRowDGW>
- +SaleProductRowDGW(int,int,double)
- +insert()
- +getSaleId():int
- +getQty():int

#### SaleRowDGW

- -date:Date
- -discount:double
- -status:String
- -total:double
- -customerId:int
- +getSaleById(int):SaleRowDGW
- +SaleRowDGW(int,Date,SaleStatus)
- +insert()
- +getClientId():int
- +getDate():Date
- +getDiscount():double
- +getId():int
- +getStatus():SaleStatus
- +getTotal():double
- +setDiscount(double)
- +setStatus(SaleStatus)
- +setTotal(double)



# Camada de Acesso aos Dados com RDGW

#### dataacess

#### CustomerRowDGW

- -id: int
- -vat:int
- -designation:String -phoneNumber:int
- -discountId:int
- +getCustomerById(int):CustomerRowDGW +getCustomerByVATNumber(int):
- CustomerRowDGW
- +CustomerRowDGW(int,String,int,DiscountType)
- +insert() +getCustomerId():int
- +getDesignation():String
- +getDiscountType():DiscountType
- +getPhoneNumber():int
- +aetVAT():int
- +setDesignation(String)
- +setDiscountType(DiscountType)
- +setPhoneNumber(int)

#### ConfigurationRowDGW

- -amountThreshold:double
- eligiblePercetange:double
- -totalAmountPercetange:double +getConfiguration():ConfigurationRowDGW
- -ConfigurationRowDGW(int,int,double)
- +getAmountThreshold():double
- +getEligiblePercentage():double
- +getAmountThresholdPercentage():double

### SaleRowDGW

- -date:Date -discount:double
- -status:String
- -total:double
- -customerld:int
- +getSaleById(int):SaleRowDGW
- +SaleRowDGW(int,Date)
- +insert() +getClientId():int
- +getDate():Date
- +getDiscount():double
- +getId():int
- +getStatus():SaleStatus
- +getTotal():double
- +setDiscount(double)
- +setStatus(SaleStatus)
- +setTotal(double)

### SaleProductRowDGW

#### -saleId:int

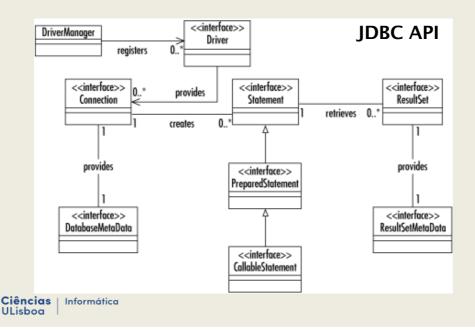
- -productId: int
- qty:int
- +getSaleProducts(int):Iterable<SaleProductRowDGW>
- +SaleProductRowDGW(int,int,double)
- +insert()
- +getSaleId():int
- +getQty():int

#### ProductRowDGW -id: int

- -discountEligibility:String
- -description:String
- -faceValue:double
- -prodCod:int
- -atv:double -unitld:int
- +getProductById(int): ProductRowDGW
- +getProductByCode(int): ProductRowDGW +ProductRowDGW(int,String,double,double,
- boolean,int)
- +insert()
- +getDescription():String +getFaceValue():double
- +getProductCod():int
- +getProductId():int
- +getQty():int
- +getTotal():double +isEligibleForDiscount():boolean
- +setDescription(String)
- +setEligibleForDiscount(boolean)
- +setQty(double) +setUnitId(int)
- +updateStockValue()



(e) Desenhe uma classe DataSource com uma única instância que abstrai e encapsula a ligação à base de dados.

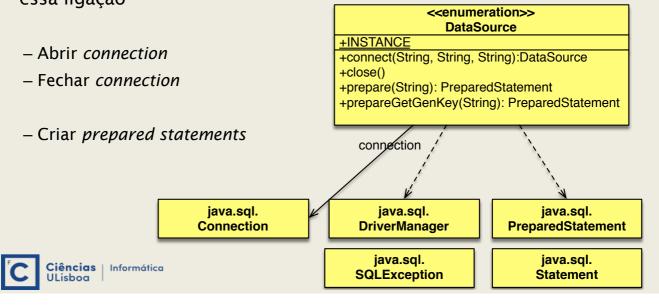


# Fonte de Dados e JDBC

Classe DataSource com uma única instância que

 abstrai a ligação à base de dados (a fonte de dados) através de um conector JDBC

 é responsável por estabelecer a ligação e por todas as ações sobre essa ligação



# <<enumeration>> Fontes de Dados e JDBC **DataSource** +INSTANCE +connect(String, String, String):DataSource Na classe DataSource o método connect usa connection DriverManager.getConnection(url, username, password) o método close usa java.sql. Connection connection.close() Na classe StartUp do SaleSys - no arranque da aplicação DataSource.INSTANCE.connect( "jdbc:derby:data/derby/cssdb;create=false", "SaleSys", • e no fecho da aplicação DataSource.INSTANCE.close()

# Fontes de Dados e JDBC

Ciências | Informática

ULisboa

<<enumeration>>
DataSource

### +INSTANCE

+connect(String, String, String):DataSource

+prepare(String): PreparedStatement

+prepareGetGenKey(String): PreparedStatement

## Na classe DataSource

 Os métodos que permitem obter objetos PreparedStatement a partir de Strings com frases SQL usam

### e constantes

Statement.RETURN\_GENERATED\_KEYS
Statement.NO\_GENERATED\_KEYS



- (f) Defina os atributos da classe CustomerRowDGW e desenhe o comportamento do método insert().
- (g) Complete a classe CustomerRowDGW desenhando o comportamento dos métodos de classe

```
getCustomerByVATNumber(int vat): CustomerRowDGW
getCustomerById(int saleId): CustomerRowDGW.
```

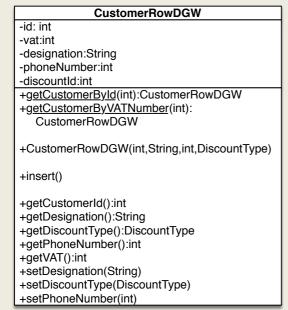


# SalesSys: Esquema da base de dados (Derby)

```
CREATE TABLE CUSTOMER (
        ID INTEGER GENERATED BY DEFAULT AS IDENTITY (START WITH 1)
                 PRIMARY KEY NOT NULL,
        DESIGNATION VARCHAR(50) NOT NULL,
        PHONENUMBER INTEGER,
        VATNUMBER INTEGER NOT NULL UNIQUE,
        DISCOUNT_ID INTEGER NOT NULL
)
CREATE TABLE PRODUCT (
        ID INTEGER GENERATED BY DEFAULT AS IDENTITY (START WITH 1)
                 PRIMARY KEY NOT NULL,
        DESCRIPTION VARCHAR(255),
        DISCOUNTELIGIBILITY SMALLINT DEFAULT 0,
        FACEVALUE DOUBLE,
        PRODCOD INTEGER,
        QTY DOUBLE CHECK (QTY>=0),
        UNIT_ID INTEGER
      Ciências | Informática
```

# Classe CustomerRDGW

Localizar o mais possível as dependências do modelo físico da fonte dos dados, definindo constantes

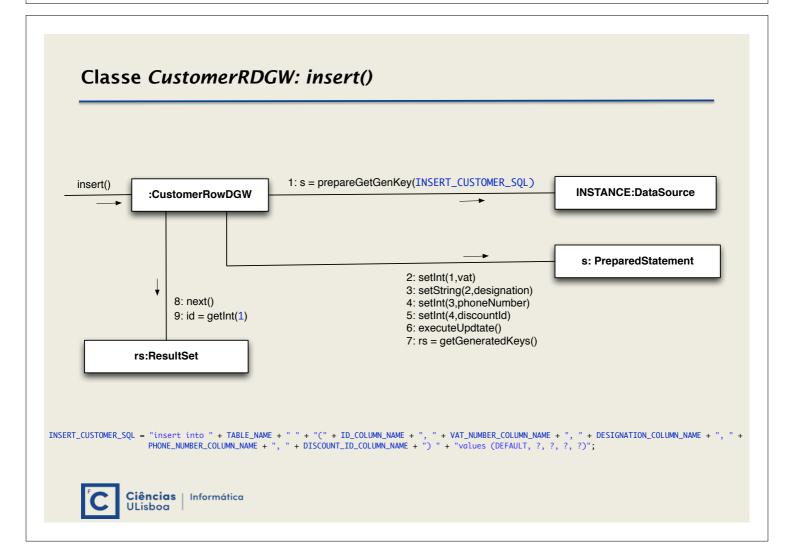


```
CREATE TABLE CUSTOMER (
ID INTEGER GENERATED BY DEFAULT AS IDENTITY (START WITH 1) PRIMARY KEY NOT NULL,
DESIGNATION VARCHAR(50) NOT NULL,
PHONENUMBER INTEGER,
VATNUMBER INTEGER NOT NULL UNIQUE,
DISCOUNT_ID INTEGER NOT NULL
)

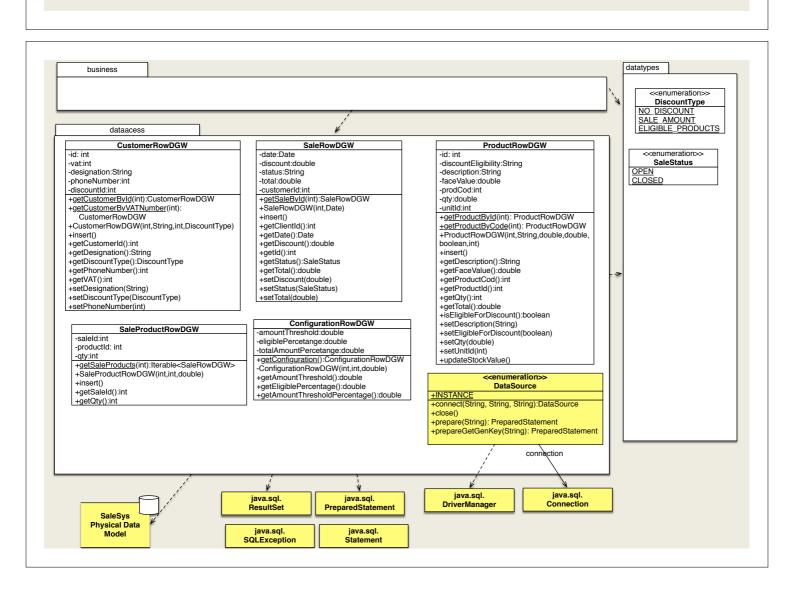
TABLE_NAME = "customer";

ID_COLUMN_NAME = "id";
VAT_NUMBER_COLUMN_NAME = "vatnumber";
DESIGNATION_COLUMN_NAME = "designation";
PHONE_NUMBER_COLUMN_NAME = "phonenumber";
DISCOUNT_ID_COLUMN_NAME = "discount_id";
```

```
Ciências | Informática ULisboa
```



# Classe CustomerRDGW: getCustomerByVATNumber(vat) c = getCustomerByVATNumber(vat)1: s = prepare(GET\_CUSTOMER\_BY\_VAT\_NUMBER\_SQL) <<metaclass>> INSTANCE:DataSource CustomerRowDGW s: PreparedStatement 2: setInt(1,vat) 3: rs = executeQuery() c:CustomerRowDGW 9: create(vat,dsg,ph,dis) 10: setId(i) 5: dsg = getString(DESIGNATION\_COLUMN\_NAME) 6: ph = getInt(PHONE\_NUMBER\_COLUMN\_NAME) 7: dis = getInt(DISCOUNT\_ID\_COLUMN\_NAME) 8: i = getInt(ID\_COLUMN\_NAME) rs:ResultSet



GET\_CUSTOMER\_BY\_VAT\_NUMBER\_SQL = "select " + ID\_COLUMN\_NAME + ", " + VAT\_NUMBER\_COLUMN\_NAME + ", " + DESIGNATION\_COLUMN\_NAME + ", " + PHONE\_NUMBER\_COLUMN\_NAME + ", " + DISCOUNT\_ID\_COLUMN\_NAME + " " + " + "from " + TABLE\_NAME + " " + "where " + VAT\_NUMBER\_COLUMN\_NAME + " = ?";

Suponha que se pretende desenhar a camada de acesso aos dados do SalesSys recorrendo ao padrão *Table Data Gateway* e que essa camada vai ser responsável por realizar a comunicação da aplicação com o SGBD escolhido — Derby — através de um conector JDBC. Apesar de existir atualmente uma única fonte de dados, o desenho da camada deve prever a possibilidade de serem usadas várias fontes de dados. Adicionalmente o desenho da camada deve ter em conta que se pretende usar esta camada por baixo dos *table modules* concebidos anteriormente (folha 2).

- (a) Explique em que consiste o padrão *Table Data Gateway* focando a sua explicação na forma como é feita a abstração da base de dados e nas responsabilidades atribuídas a cada objeto do padrão.
- (b) Identifique as classes *gateway* de que vai precisar, os seus atributos e métodos.



### Classe CustomerTableDGW

Localizar o mais possível as dependências do modelo físico da fonte dos dado, definindo constantes

#### CustomerTableDGW

+insert(int,String,int,DiscountType) +find(int):TableData

+findByVATNumber(int): TableData

+readId(Row):int

+readVatNumber(Row):int

+readDesignation(Row):String

+readPhoneNumber(Row):int

+readDiscountType(Row):DiscountType

#### CREATE TABLE CUSTOMER (

ID INTEGER GENERATED BY DEFAULT AS IDENTITY (START WITH 1) PRIMARY KEY NOT NULL,

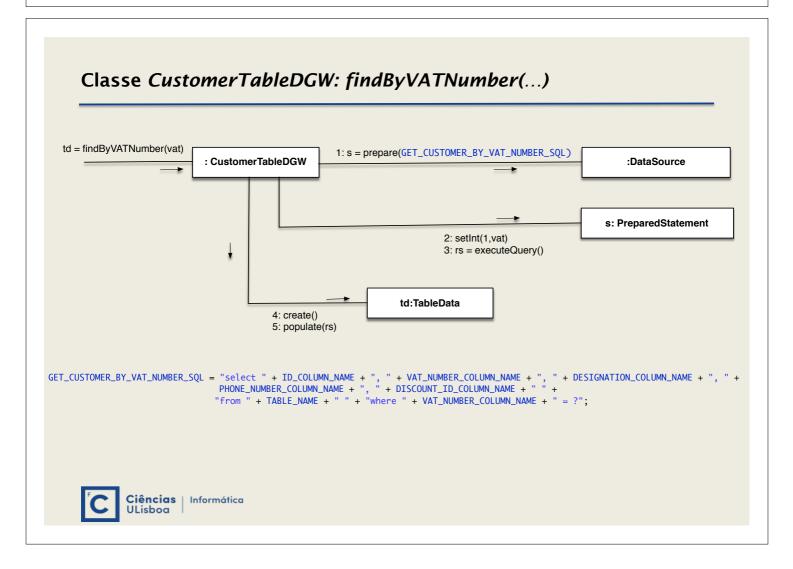
DESIGNATION VARCHAR(50) NOT NULL,
PHONENUMBER INTEGER,
VATNUMBER INTEGER NOT NULL UNIQUE,
DISCOUNT\_ID INTEGER NOT NULL

```
TABLE_NAME = "customer";

ID_COLUMN_NAME = "id";
VAT_NUMBER_COLUMN_NAME = "vatnumber";
DESIGNATION_COLUMN_NAME = "designation";
PHONE_NUMBER_COLUMN_NAME = "phonenumber";
DISCOUNT_ID_COLUMN_NAME = "discount_id";
```



# Classe CustomerTableDGW: insert(...) insert(vat, designation, phoneNumber,discountType) 1: s = prepare(INSERT\_CUSTOMER\_SQL) :DataSource :CustomerTableDGW s: PreparedStatement 2: setInt(1,vat) 3: setString(2,designation) 4: setInt(3,phoneNumber) 5: discountId = getId() 6: setInt(4,discountId) 7: executeUpdate() discountType:DiscountType INSERT\_CUSTOMER\_SQL = "insert into " + TABLE\_NAME + " " + "(" + ID\_COLUMN\_NAME + ", " + VAT\_NUMBER\_COLUMN\_NAME + ", " + DESIGNATION\_COLUMN\_NAME + ", " + PHONE\_NUMBER\_COLUMN\_NAME + ", " + DISCOUNT\_ID\_COLUMN\_NAME + ") " + "values (DEFAULT, ?, ?, ?, ?)"; Ciências | Informática ULisboa



# Table Data

 TableData é uma classe cujos objetos representam dados tabulares (record sets)

dataaccess

data: List<Row>
data: List<Row>
Source
Row()
update(String, Object): void
getString(String): String
getInt(String): int
getDate(String): Date
getDouble(String): double
populate(ResultSet): TableData
isEmpty(): boolean
iterator(): Iterator<Row>

• É um tipo semelhante a java.sql.ResultSet mas mais abstrato

public interface ResultSet
extends Wrapper, AutoCloseable



A table of data representing a database result set, which is usually generated by executing a statement that queries the database.

# SalesSys: Solução com Table Module + Table Data Gateway

