FEUP Universidade do Porto
Faculdade de Engenharia

Industrial Informatics

[Informática Industrial]

2022/23 edition

Lazarus DB - CRUD

José Faria, Andry Pinto

Contents

0. Introduction

1. CRUD-insert

2. CRUD-delete

3. CRUD-update

4. Putting it all together

0. Introduction

Assignment #4

- What about the assignment, everything ok?
- Did you all managed to submit on time and on quality?

--

- Did you have problems with the VPN?
- Have you tried the local database installed in assignment #1?

Dynamic queries

- How did you create the dynamic queries, using string **concatenation**, or query **parameters**?
- What **solution do you prefer**, concatenation or parameters?
- Probably, you prefer parameters but, as we'll see soon, there are **many situations** where we **can't avoid** the use of string **concatenation** ☹ !

CRUD

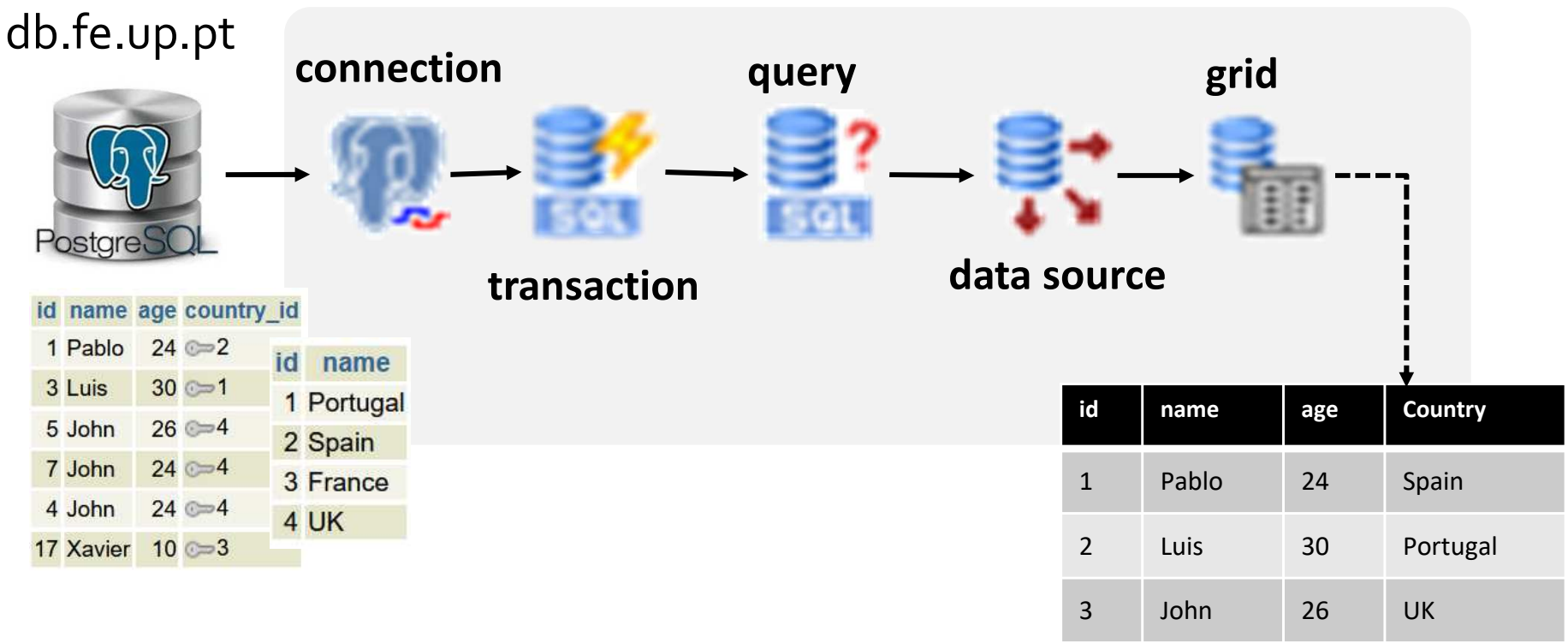
- The acronym CRUD stands for the **4 basic database operations**:

Create	(SQL Insert)
Read	(SQL Select)
Update	(SQL Update)
Delete	(SQL Delete)

- You probably already know these commands from Information System 1.
- If you don't, look at W3Schools to get familiar with them.

Previous class and assignment#4: Read

- Until now, we have only accessed the database to get data through select queries:



Today's class: Create, Delete and Update

- Today, we are going to develop a set of **simple applications** to **insert**, **delete and update records** in table friends.
- Then, **you'll be *invited* to put it all together**, and create a CRUD application for table friends.
- This application is **very similar** to the **mini-ERP's module Client's orders management** that you'll develop later.

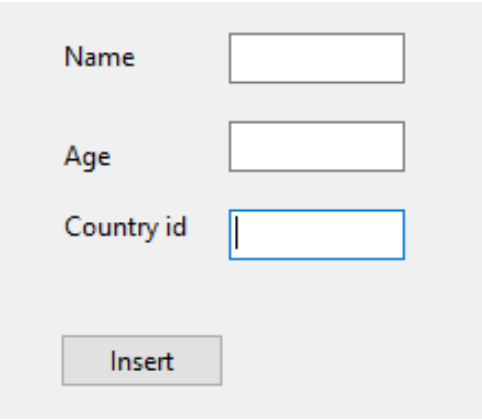
1. CRUD-insert

Application CRUD-insert

- We are going to develop **2 versions** of this application:

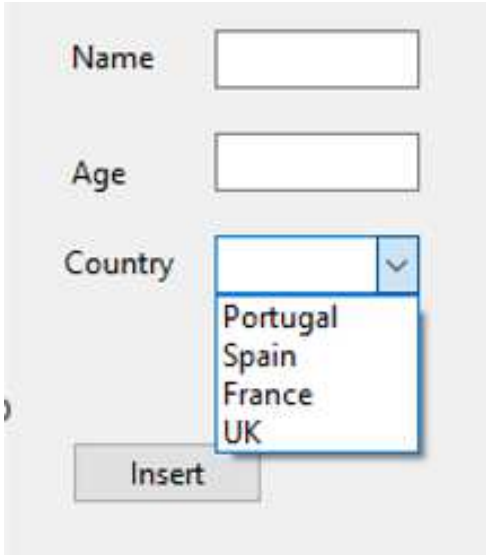
1st version: text boxes

User enters name, age and country_id in text boxes



2nd versions: combo box

User selects the country by its name in a combo box



INSERT statement

Syntax

INSERT INTO *table_name* (*column1*, *column2*, *column3*, ...)
VALUES (*value1*, *value2*, *value3*, ...);

Example

INSERT INTO Customers (CustomerName, City, Country)
VALUES ('Wilman Kala', 'Stavanger', 'Norway');

W3Schools

https://www.w3schools.com/sql/sql_insert.asp

1.1. Insert 1st version: textboxes

Insert 1st version

1. Go to Moodle and download the zip file ***CRUD-insert with textboxes*** and open de **working version**.

2. Analyze the code in the events *FormCreate* and *btInsertClick*

3. Then, follow the instructions in the next slide (the instruction are also embedded in the code).

Insert 1st version → Instructions

1. Edit the dynamic *insertQuery* from the text boxes
2. Run the application and see the dynamic query in *edDebugQuery*
3. Copy/paste it to phppgadmin's SQL window and test it
4. If ok, decomment *the ExecuteDirect* instruction, run the application and confirm the result in phppgadmin

ExecuteDirect

- The **SQLQuery** object **only** allows to execute **SELECT** queries: it sends a command, and stores the result returned by the DB server in a dataset (we'll see it soon).

- **INSERT**, **DELETE** and **UPDATE** queries **do not return data** to the client, so we should employ the method, ***ExecuteDirect*** of the PQConnection object, e.g.:

```
PQConnection1.ExecuteDirect('query');
```

1.2. Insert 2nd version: combo box

Introduction

- Due to the combo box, this 2nd version is ***a bit more complicated*** than the 1st.

- To implement it, we need to:

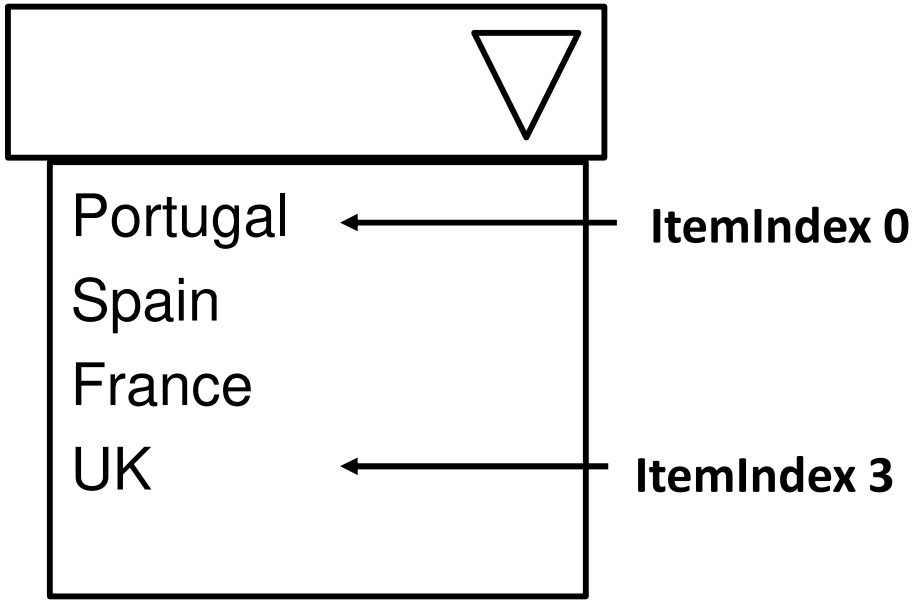
  1. Get the list of countries from the database
  2. Display the names of the countries in the combo
  3. Get the id of the country selected by the user (note that the user sees and selects the country by name but, in the Insert query, we need the id).

Relevant properties

- Before looking at the code of the application, you **need to know** a few **properties** of **combo boxes** and **SQLQuery**'s.

- Look at the **following slides** for an explanation of those properties.

Combo box: ItemIndex property



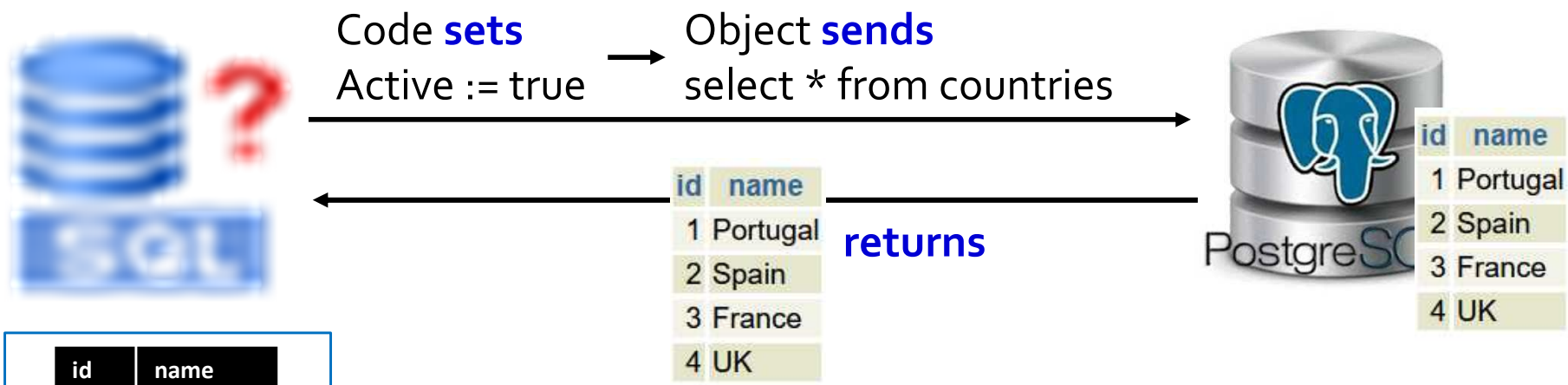
ItemIndex takes the **number of the item** selected (from 0 to nr of items – 1).

If no item is selected, ItemIndex takes the value -1.

- **user selects Spain** → property **cbCountry.ItemIndex** takes? **1**
- **code sets** cbCountry.**ItemIndex := 2**, what **item is selected?** **France**

SQLQuery: internal dataset

SQL property = 'select * from countries'



id	name
1	Portugal
2	Spain
3	France
4	UK

SQLQuery's internal dataset

- The **data returned** by the query is **stored** in an internal dataset (array of objects), whose **cells can be accessed from the code**.

- **Interestingly:** we don't need to declare the columns nor the rows of the dataset, as it **adjusts automatically to the data retrieved by the DB server**

SQLQuery: RecordCount and RecNo

- Property RecordCount holds the nr of records in the dataset
- Property SQLQuery.RecNo takes the nr of the record currently selected (only one record can be selected at a time)

id	name
2	Portugal
4	Spain
5	France
7	UK

→ **RecNo = 1**

RecordCount = 4

→ **RecNo = 4**

- SQLQuery.RecNo := 2 → sets current record to

2	Spain

SQLQuery: Fields[] and FieldByName()

id	name
2	Portugal
4	Spain
5	France
7	UK

We can the **access the values in the cells** of the selected record by:

- by the **index** of the column (index ranges from 0 to FieldCount -1)
- by the **name** of the column

SQLQuery.RecNo := 3;

SQLQuery.Fields[1].AsString returns? **'France'**

SQLQuery.FieldByName('id').AsString returns? **'5'**

SQLQuery.FieldByName('id').AsInteger returns? **5**

- Knowing these properties, you can now **easily understand** the **version** of the application **with the combo** *cbCountry* !

- Follow the instruction on the next slides.

insert 2nd version → Instructions

1. Download application ***CRUD-insert with combo*** from Moodle and open the **working version**.

2. See the ***SQL*** property of the object ***SQLCountries***.

3. Look at the ***FormCreate*** event and see how the application **populates the items of the combo box** from the countries retrieved from the DB.

4. Run the application and see the list of countries being displayed in the combo box.

insert 2nd version → Instructions

5. Now, look at the ***btInsertClick* event** and see how it gets the id of the country from the name selected by the user in the combo (this is the key point of this version).

6. **Edit the insert query**, getting name and age from the text boxes, and country_id from variable countryId.

7. As before, run the application and **validate the query**.

8. If ok, **decomment** the instruction **ExecuteDirect** and check the result.

2. CRUD-delete

DELETE statement

Syntax

DELETE FROM *table_name* WHERE *condition*;

Example

DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';

W3Schools

https://www.w3schools.com/sql/sql_delete.asp

Now, let's go for the Delete version!

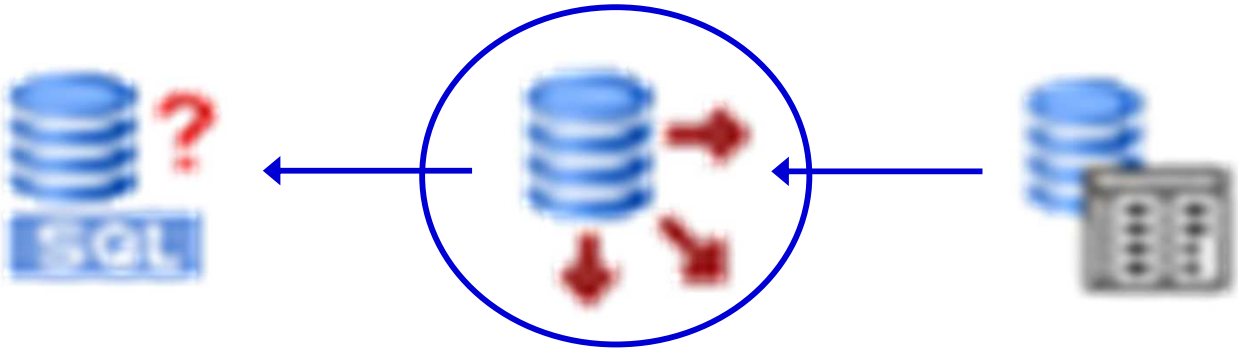
This new version is closer to a *real world* application:

1. The application **starts** by **displaying** the content of the table **friends in a grid**.
2. When the user **clicks in a grid's cell**, the corresponding **record (row) is displayed** in an auxiliar form (one control per each column we want to show).
3. The ***btDeleteClick*** event, **deletes** from the database the **record selected** by the user in the grid, and displayed in the form.

DBGridCellClick event

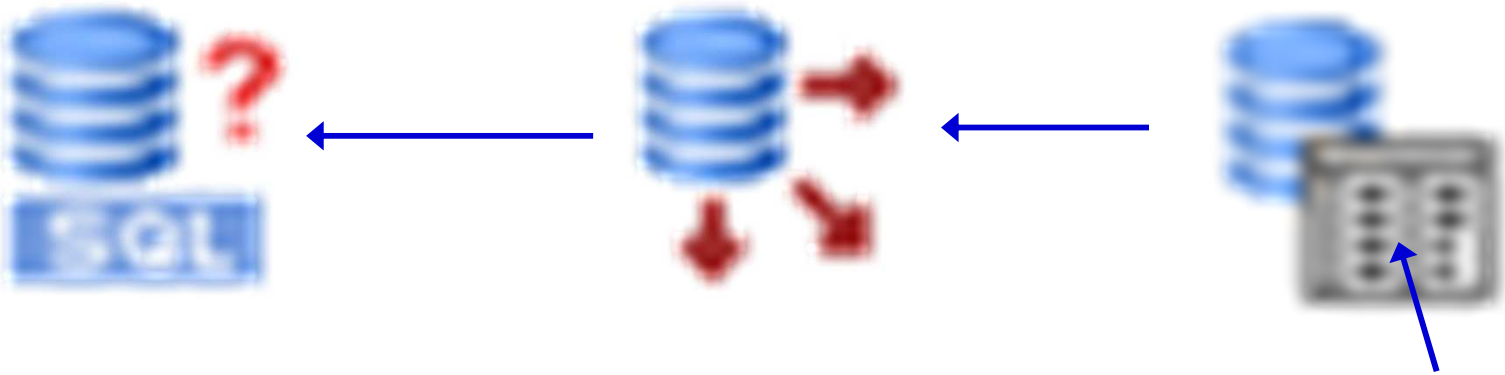
- In order to understand the application, you need to know ***DBGridCellClick***, an event that occurs when the user clicks a cell in the grid.

- When this happens, the data source connecting SQLQuery and DBGrid,



sets RecNo of the **SQLQuery** to the record corresponding to the **cell clicked** in **DBGrid !**

SQLQuery **DBDataSource** **DBGrid**



2. The DataSource **sets RecNo** of the SQLQuery to the corresponding record

1. User **clicks** in the **DBGrid**

friend_id	friend_name	friend_age	country_id	country_name
5	John	26	4	UK
7	John	24	4	UK
4	John	24	4	UK
3	Luis	30	1	Portugal
1	Pablo	24	2	Spain
22	Claire	26	3	France
23	Meggie	22	4	UK

RecNo = 4

RecNo = 6

	friend_id	friend_name	friend_age	country_id	country_name
1	5	John	26	4	UK
2	7	John	24	4	UK
3	4	John	24	4	UK
4	3	Luis	30	1	Portugal
5	1	Pablo	24	2	Spain
6	22	Claire	26	3	France
7	23	Meggie	22	4	UK

delete → Instructions

1. Download application ***CRUD-delete*** from Moodle and open the **working version**.

2. Start by looking attentively to the ***SQL*** property of ***SQLFriends***.

3. Look at the ***DBCellClick*** event to see how the application displays, in the auxiliar form, the record in the grid.

   …

delete → Instructions

4. **Edit the Delete query** , getting the **id of the record to be deleted** in table friends from the **dataset of SQLFriends**.

5. As usual, **validate the query** and decomment the *ExecuteDirect* instruction.

3. CRUD-update

UPDATE statement

Syntax

UPDATE *table_name*

SET *column1* = *value1*, *column2* = *value2*, ...

WHERE *condition*;

Example

```
UPDATE Customers
SET ContactName = 'Alfred Schmidt', City= 'Frankfurt'
WHERE CustomerID = 1;
```

W3Schools

https://www.w3schools.com/sql/sql_update.asp

Finally, the update version

This version does not require new properties or events, but includes a ***tricky stuff***:

- When the user clicks a cell in the grid, the name of the corresponding country is displayed in the combo

- So we need to:
  1. populate the combo with all possible countries (so that the user can select a different one)
  2. display by default the country stored in the database

update → Instructions

1. Download application **CRUD-update** from Moodle and open the **working version**.

2. Look at **DBGridCellClick** to see to chose in the combo the country name corresponding to the record selected.

3. Then create the Update query and proceed to its validation as usual.

4. Putting it all together

- Now, **you are ready** to implement an application implementing the **4 CRUD operations** in a entity (table) related with other entities (foreign keys)

- Download application CRUD-All from Moodle, open it and **edit the code of the events** highlighted.

- You can implement CRUD in table friends, or you can do it in table Customer_orders you created in assignment #4.

good luck

and enjoy it ☺ !

thank you !