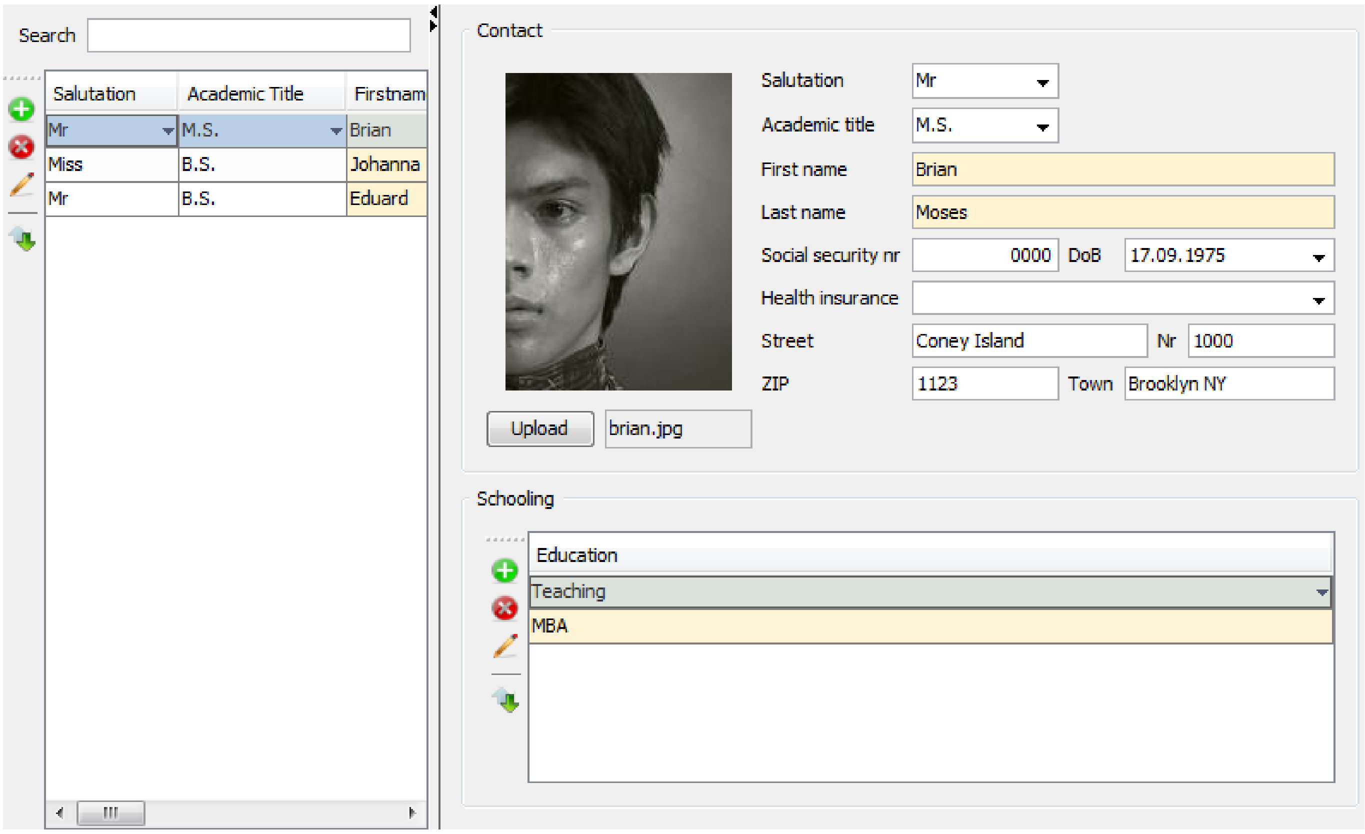
One of the most laborious tasks during the development of applications is the definition of list boxes. List boxes require a lot of source code and the developer wastes time that would be better spent on complex tasks.

Selection lists for database applications can, for the most part, be created automatically if the underlying data model is cleanly designed, i.e. PKs, UKs, FKs are available.

JVx can handle the creation of list boxes based on the relationships between tables.

Example

We will take a simple contact administration as an example:



Search

Salutation	Academic Title	Firstnam
Mr	M.S.	Brian
Miss	B.S.	Johanna
Mr	B.S.	Eduard

Contact

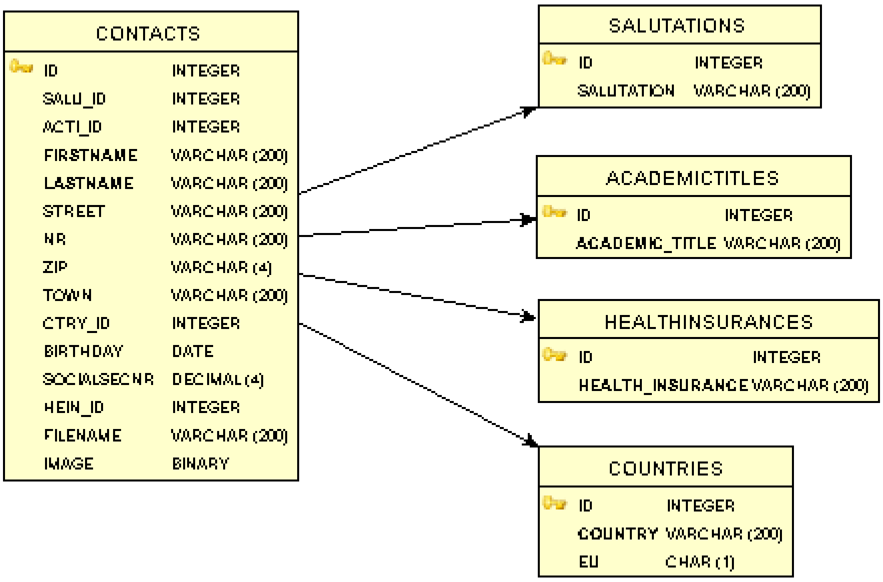
Salutation	Mr		
Academic title	M.S.		
First name	Brian		
Last name	Moses		
Social security nr	0000	DoB	17.09.1975
Health insurance			
Street	Coney Island	Nr	1000
ZIP	1123	Town	Brooklyn NY

brian.jpg

Schooling

Education
Teaching
MBA

For each contact the following data is recorded: salutation, academic title, insurance, country. The data model looks as follows:



We want to show salutation, academic title, insurance and country in a list box.

We would usually need a DBStorage for each table in the database. However, we only need one! In our lifecycle object at the server we therefore only define the following:

```
/**
 * Returns the contacts storage.
 *
 * @return the Contacts storage
 * @throws Exception if the initialization throws an error
 */
public DBStorage getContacts() throws Exception
{
   if (dbsContacts == null)
   {
      dbsContacts = new DBStorage();
      dbsContacts.setDBAccess(getDBAccess());
      dbsContacts.setWritebackTable("CONTACTS");
      dbsContacts.setAutoLinkReference(true);
      dbsContacts.open();
   }

   return dbsContacts;
}
```

The command

```
setAutoLinkReference(true);
```

activates the automatic recognition of list boxes. It should be noted that automatic recognition can be activated/deactivated using the following command:

```
DBStorage.setDefaultAutoLinkReference(false);
```

This option is activated by default!

At the client, there is also only a single RemoteDataBook required:

```
RemoteDataBook rdbContacts = new RemoteDataBook();

rdbContacts.setDataSource(dataSource);
rdbContacts.setName("contacts");
rdbContacts.open();
```

The list boxes are created and assigned automatically. The developer only needs to edit if the columns offered in the list boxes need to be changed.

Configuring List Boxes

The columns shown in the automatically created list boxes are subject to the following rules:

PK and UK columns are displayed; if there are no UKs, all of the table´s columns are displayed.

In the original DBStorage a new column is added automatically. This column is selected from the list table and corresponds to the first text column.

To limit the list boxes' columns, it is generally sufficient to define a UK in the list table. If that is not enough, it is possible to manually define the list box:

```
/**
 * Returns the contacts storage.
 *
 * @return the Contacts storage
 * @throws Exception if the initialization throws an error
 */
public DBStorage getContacts() throws Exception
{
   if (dbsContacts == null)
   {
      dbsContacts = new DBStorage();
      dbsContacts.setDBAccess(getDBAccess());
      dbsContacts.setFromClause("V_CONTACTS");
      dbsContacts.setWritebackTable("CONTACTS");
      dbsContacts.setAutoLinkReference(true);
      dbsContacts.open();

dbsContacts.getMetaData().getColumnMetaData("COUNTRY").setLinkReference(
         new StorageReferenceDefinition(new String[] {"CTRY_ID", "COUNTRY"},
                                        "countries",
                                        new String[] {"ID", "COUNTRY"}));

      //another way to set a link reference, for more than one column:
CTRY_ID, COUNTRY
      dbsContacts.createAutomaticLinkReference(new String[] {"CTRY_ID",
"COUNTRY"},
                                               getCountries(),
                                               new String[] {"ID",
"COUNTRY"});
   }

   return dbsContacts;
}

/**
 * Returns the countries storage.
 *
 * @return the Countries storage
 * @throws Exception if the initialization throws an error
 */
public DBStorage getCountries() throws Exception
{
...
}
```

When manually defining list boxes we must note that the reference columns also have to be defined manually. For this reason we use a database view for the selection of the data. This view includes the field COUNTRY. The field is not included in our original table, but should be offered at the client as a list box!

Note

Automatic list boxes are only recommended for the direct use of tables, since the recognition of relationships using database views is not (yet) possible. If manual definition of list boxes is used anyways, at least the definition of the selection lists at the client is not required, which saves some time.