ERDPlus – instructions

Databases

Academic year 2020./2021.

Content

Programski alat ERDPlus

The programming tool ERDPlus acts as a help in modelling a relational database.
ERDPlus is a free online tool and doesn't have to be installed locally on the computer.

Main features of the ERDPlus tool which will be mostly used on the *Databases* course are:

- creation of the ER-diagrams and relational diagrams (relational shemas)
- automatically mapping the ER-diagram to the relational schema
- export of the standard SQL DDL queries based on the relational schema
- export of the diagrams in the PNG format.

This tool will be used in the 3rd homework and during the 2nd lab work.

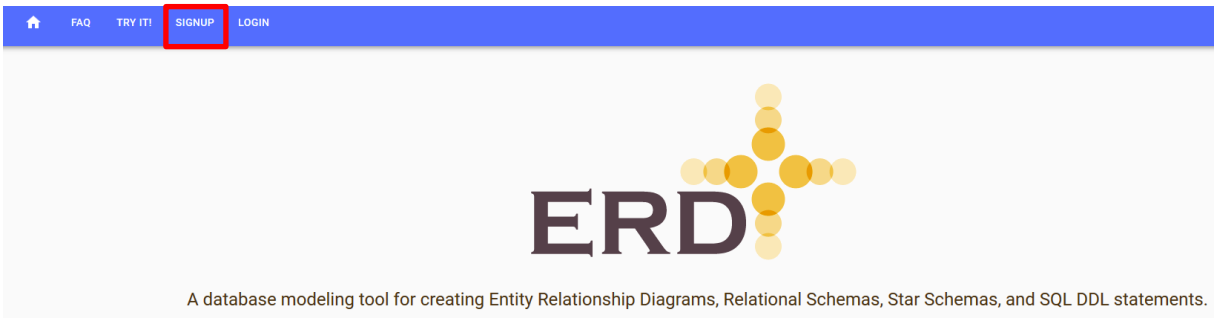
Creating a user account

The tool can be used with the basic (limited) fuctionalities as an anonymous used, although in that case the diagrams can be saved and later used. In case a registered user with a user account uses the tool gives the option of saving the created diagrams to the server. By creating a user account and without logging in, the tool can be used with the limited functionalities.

The web page of ERDplus is https://erdplus.com/.

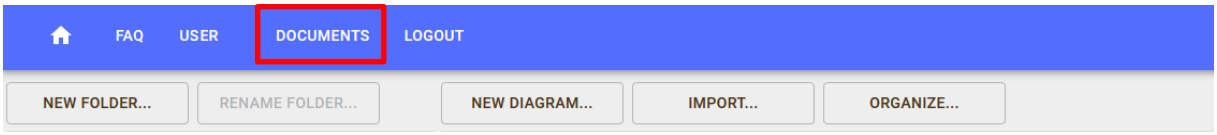
As a broader functionalities are needes for this course, it is necessary to create a user account.

To create a user account, click on the *SIGNUP* option in the menu.

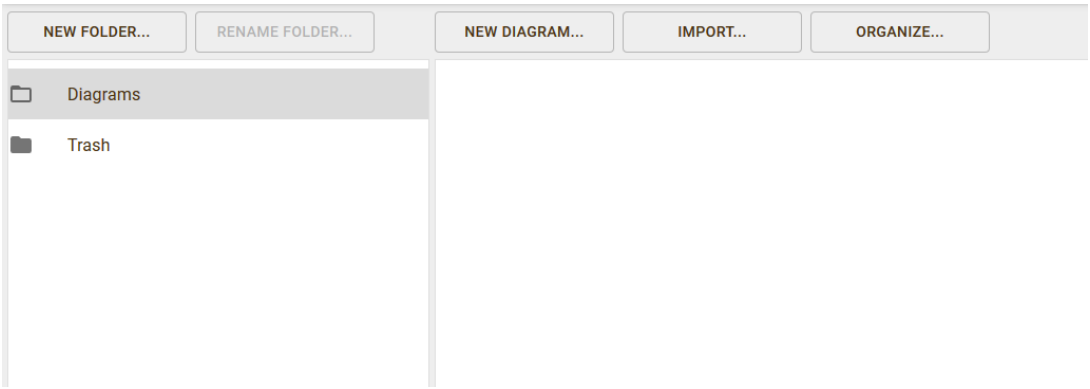


After which the user data have to be entered and finalized by clicking on the *CREATE ACCOUNT*.

Upon the user account creation, the user is logged in the system and redirected to its own storage space (*DOCUMENTS*) for saving the documents on the server.



On the left the folders are shown, while on the right the content of the chosen folder are visible.



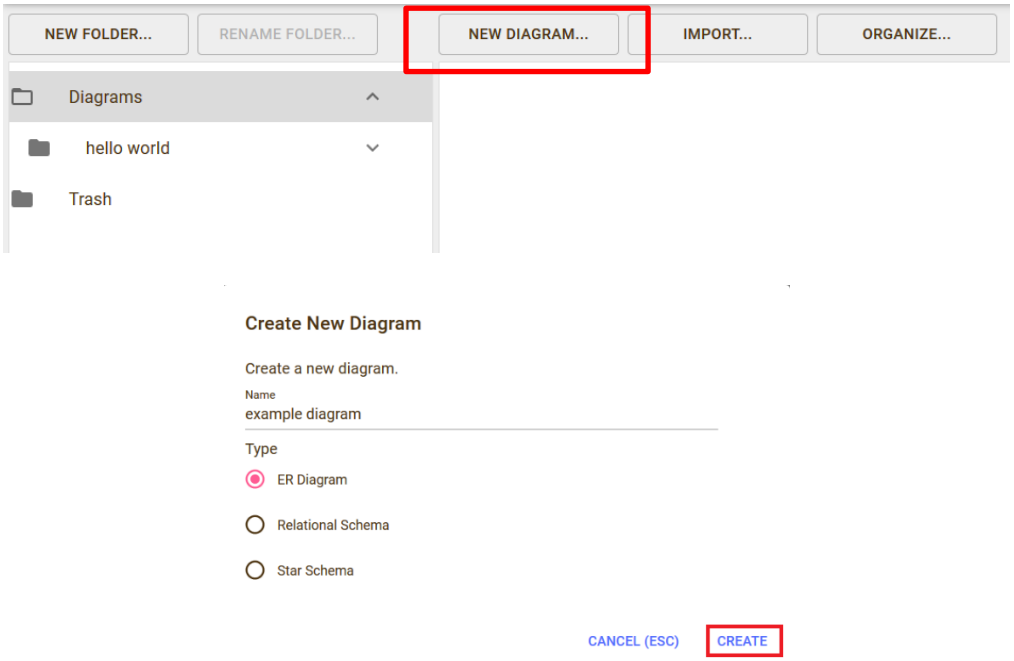
Inside its own storage, the user can create, rename and edit diagrams and folders.

The created diagrams can be imported and exported as files. By clicking on the button IMPORT, the locally stored digrams can be imported to the chosen folder in the tool.

Working with diagrams and files

Creating the diagram

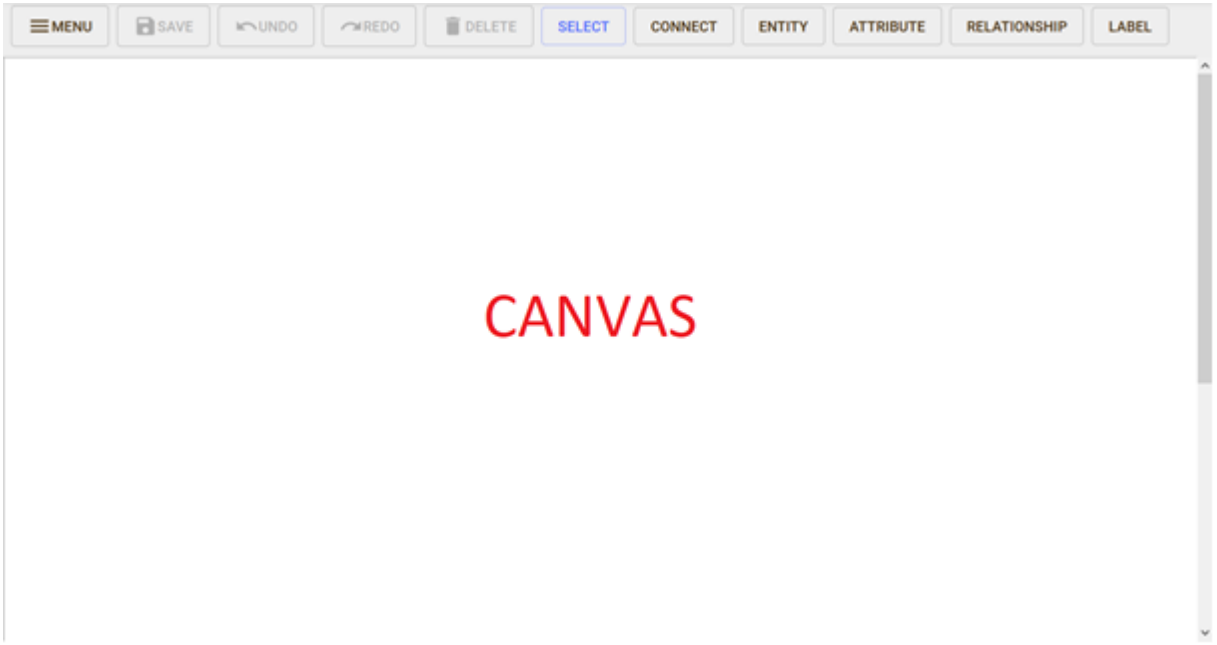
The diagram is created by clicking on the button *NEW DIAGRAM…* after which the diagram name should be entered. As the ER diagram has to be created in the example, choose the option *ER Diagram* and click *CREATE.*



The effect of this action in the user space is similar to the action „New File“ in the filesystem; a new ER –diagram is created in the chosen folder („Diagrams“).

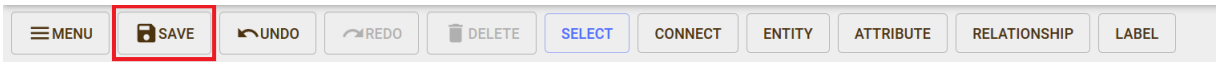
For the newly created diagram, by choosing the three dots on the far right (⋮) an activity menu for the chosen diagram is shown

By choosing the option *Open…* the editor for the chosen diagram is opened that currently contains the empty drawing canvas and the toolbar.



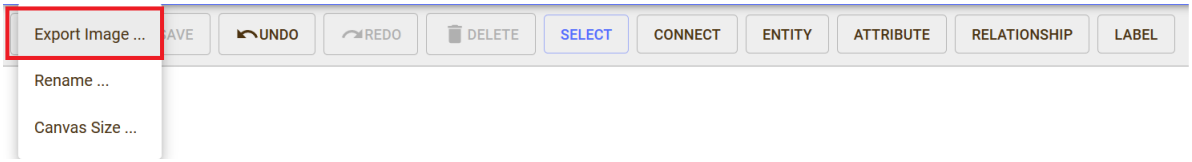
Saving the changes

Although the tool often saves the changes made in the active diagram automatically, the changes can be stored manually by clicking on the button SAVE. In case the button is disabled, either there were no changes since the last diagram import or the tool already saved the changed automatically.



Exporting the diagram as an image

On the toolbar choose the button *MENU* > *Export Image…* > *OK*, after which the exported diagram image in the *.png* format will be available for the download.

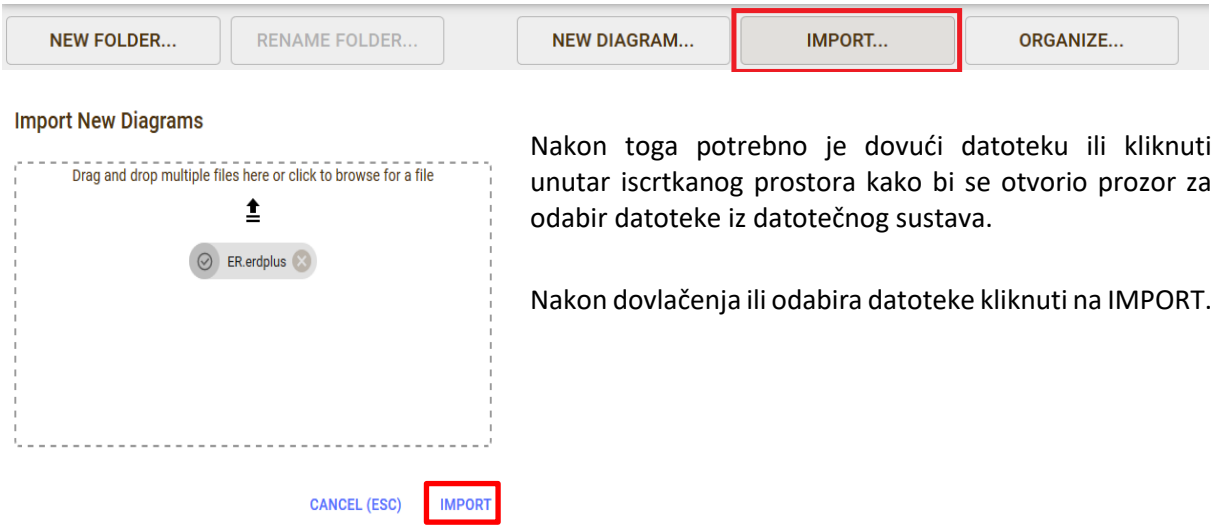


Exporting the diagram as ERDPlus file type

To export the diagram in the ERDplus file type, the user should go to the user space (by clicking on the button *DOCUMENTS*). For the chosen diagram the activity menu has to be activated (clickin on the ⋮ on the right), followed by the options *Export* > *OK*, by which the file of the file type *.erdplus* will be available for the download.

Importing the diagram from the existing ERDPlus file

To import the diagram as ERDPlus datoteke the user has to be in the user space (DOCUMENTS) and choose IMPORT….



Nakon toga potrebno je dovući datoteku ili kliknuti unutar iscrtkanog prostora kako bi se otvorio prozor za odabir datoteke iz datotečnog sustava.

Nakon dovlačenja ili odabira datoteke kliknuti na IMPORT.

Examples – part 1

The examples will guide you through the steps of creating the diagram for the simple database, where each step assumes that the prior steps were mastered. Due to that it is advisable to go through the instructions sequentially, without skipping any of the steps.

Task 1.

Create a simple ER-dijagram for the database which records the data about the students and their activities. Each student record includes JMBAG (student identification number), name and surname, and for activity its code and name. The database records whether a student takes part in an activity;

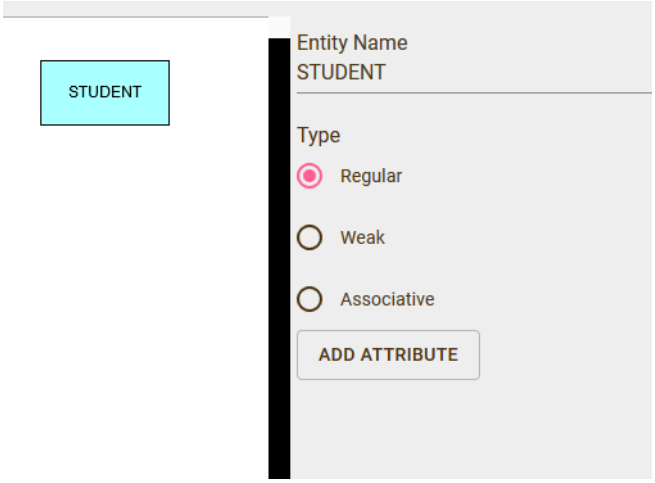
student can take part in zero or more activities, while zero or more students can take part in a single activity.

Entities

To add an entity the button *ENTITY* from the toolbar should be chosen, after which the user should click anywhere on the canvas. An editor for the new entity is opened automatically on the right side, where the entity name should be entered (e. g. STUDENT) and its type chosen. In this case, the chosen entity is regular (not weak).

Generally, from the suggested options you will need:

- **Regular**: Regular entity
- **Weak**: Weak entity



The same steps should be repeated for the entity ACTIVITY.

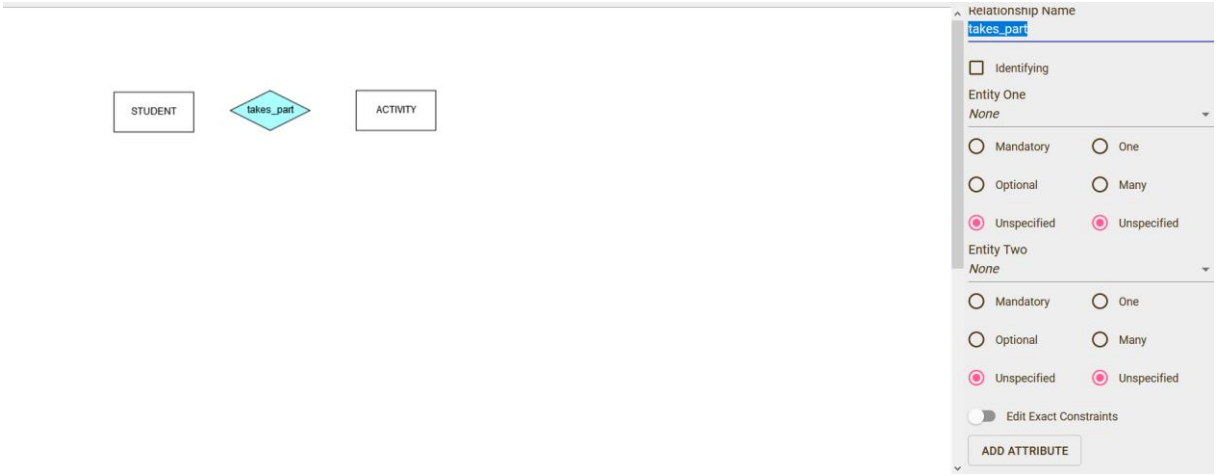
Note:

- Entites can be arranged in the canvas through *drag-and-drop*. Moreover, some or all of the elements in the canvas can be selected (chosen), and the chosen elements can be arranged by dragging over the canvas. To enable the selection, the option ***SELECT*** on the toolbar should be chosed, which will set the tool in the SELECT mode. More elements on the diagram can be chosen (selected) in two ways:
  - Drag the rectangle using the mouse which will „capture“ all the selected elements
  - While holding the CTRL key on the keyboard, clicking the mouse on one element at the time.
- If only one diagram element inside the canvas is chosen (selected), the tool will mark it blue and will open its editor automatically

- For the chosen diagram element:
  - The key *Delete* on the keyboard deletes its name
  - Choosing *DELETE* on the toolbar deletes the selected entity.

Binary N:N relationship

To add a new relationship, the button *RELATIONSHIP* from the toolbar should be selected, followed by clicking anywhere inside the canvas. On the right the new relationship editor is automatically opened.



Followed by entering the name of the relationship (e. g.. „takes_part“).

The next step is connecting the relationship to the entities.
It can be done in two ways:

<u>Method 1:</u> Using the dropdown menus.

Activate the relationship editor. From the dropdown menus manually set the entities *Entity One* i *Entity Two*, after which the connecting lines between the relationship and the chosen entities will be shown.

<u>Method 2:</u> Using the *CONNECT* option.

Choose the *CONNECT* on the toolbar. Click either on the entity STUDENT or on the relationship and while **holding** the left mouse button– drag the line between them. When the relationship, entity and the connecting line go green, release the mouse button.

Note:

After creating the first connection betwen relationship and the entity, the tool isn't in the *CONNECT* mode anymore.
Before creating the second connection the *CONNECT* option should be selected again.

Repeat the procedure for the entity ACTIVITY.

Upon connecting of the relationship with the entities STUDENT and ACTIVITY, in the relationship under *Entity One* the label STUDENT is shown, while in the *Entity Two* the label ACTIVITY is shown. In that way the tool automatically sets the entities which the relationship is connected to.

Any of the two mentioned methods may be used to connect the relationship and entities.

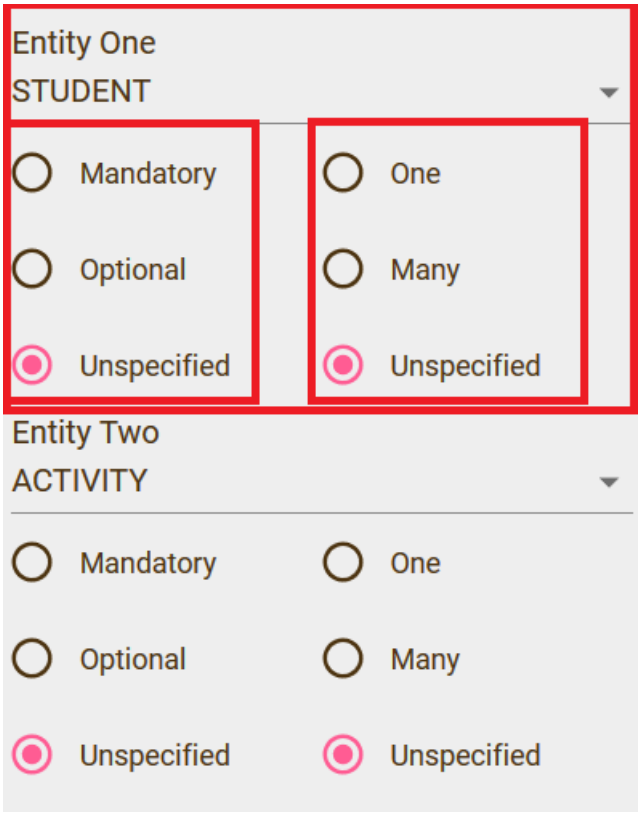
The next step is entering the entity cardinality.

Choose the relationship to activate the editor on the right side.

The relationship in the focus is the binary N:N relationship (student can take part in several activities, and multiple students can take part in a single activity). Therefore, the upper boundary for the both relationship ends will be „many“. But what about the lower boundary? Does a student necessary take part in a activity? Should all activities have to be taken by at least one student? In both cases the answer is „no“, as it „zero or more“ constraints have been specified.

Therefore, set the entity cardinality settings – the upper cardinality setting will be *many*, while the lower cardinality setting is zero for both entities connected by the relationship.

In the relationship editor two columns with options are shown – one for each entity connected to the relationship.



The left option column refers to the lower boundary, and the right to the upper relationship boundary.
In the left column (used for the lower boundary) the options are:

- **Mandatory**: **lower** boundary is 1.
- **Optional**: **lower** boundary is 0.
- ~~**Unspecified**~~: won't be used.

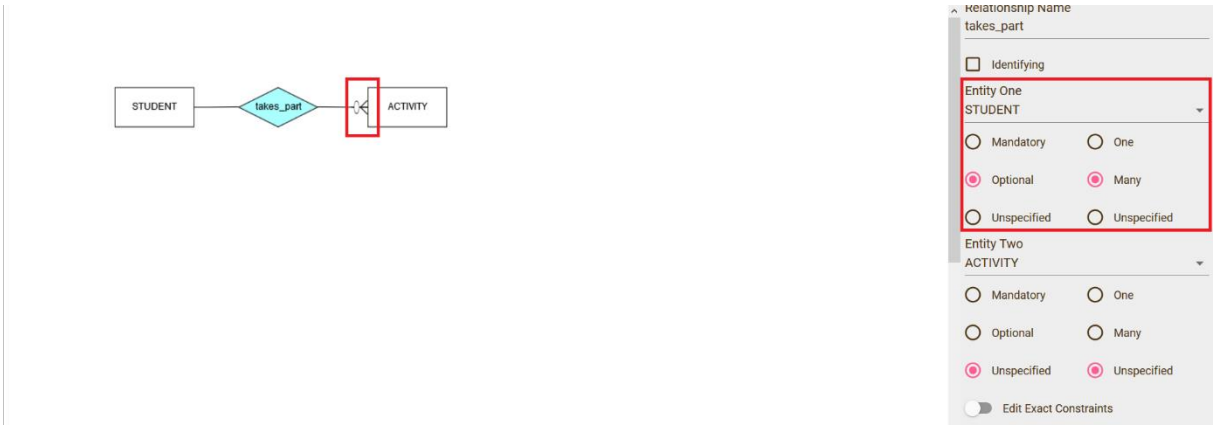
In the right column (used for the upper boundary) the options are:

- **One**: **upper** boundary is 1.
- **Many**: **upper** boundary is N.
- ~~**Unspecified**~~: won't be used.

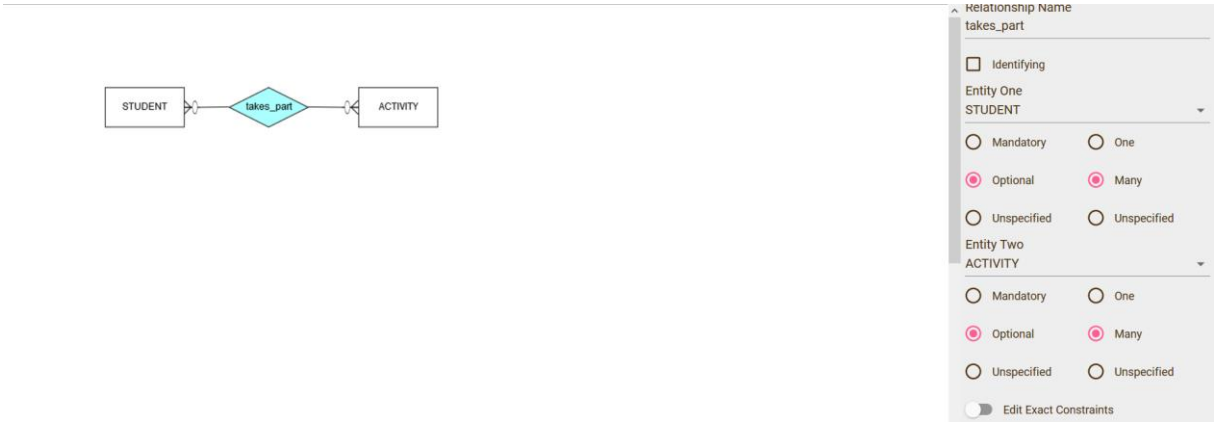
Combining these four options, the entity cardinality can be set to: 1..1, 1..N, 0..1 i 0..N.

Let's set the cardinality for one of the connected entities (e. g. 0..N for the entity STUDENT). In the left column the option *Optional* should be chosen, while in the right the *Many* option should be chosen.

Notice the cange on the diagram – next to the entity ACTIVITY the symbols (*Many*) and (*One*) have appeared.



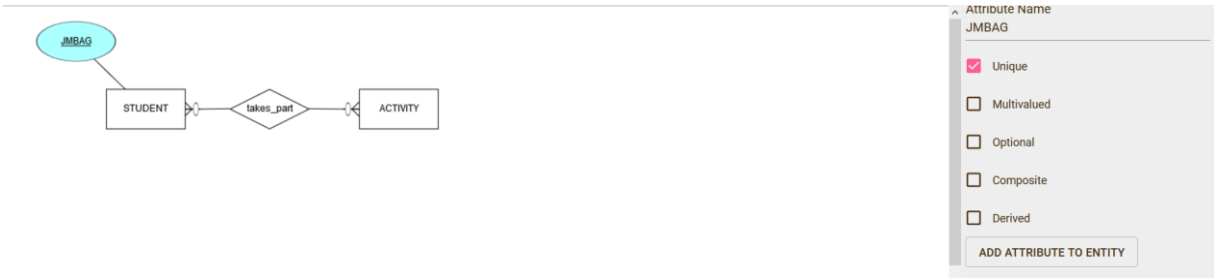
In the same way edit the cardinality for the entity ACTIVITY.



Adding the attribute to the entity

After choosing the entity, in the editor on the right choose the *ADD ATTRIBUTE*, after which the new attribute will be added to the selected entity on the diagram, and the attribute editor will be shown on the right side.

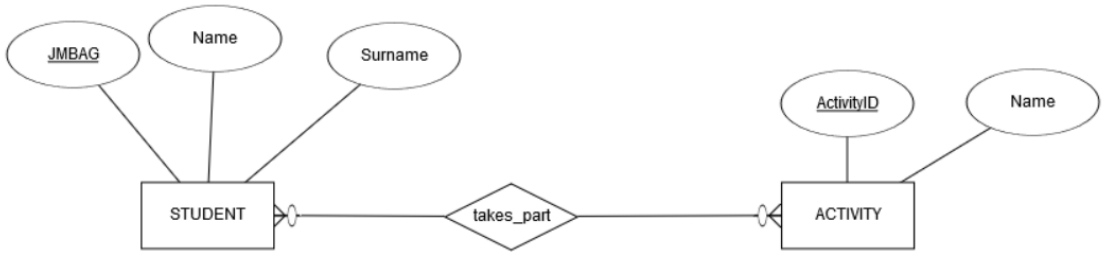
In the editor, add the attribute name (e. g. JMBAG). Since JMBAG has the role of the primary key, the option UNIQUE should be chosen.



From the available options, the following ones are needed:

- **Optional**: The attribute can have NULL value (it is optional). If the option *Optional* isn't explicitly chosen in this tool, **the attribute will have NOT NULL constraint**.
- **Unique**: The attribute should have unique value. Therefore, by setting UNIQUE constraint (without choosing the *Optional* option) with the uniqueness constraint the NOT NULL is alluded. Choosing this option is the equivalent to primary key setting.

Add the rest of the attributes to the entity STUDENT, add the attributes to the entity ACTIVITY and set its primary key.



By this, the ER diagram for the *Task* 1 was created.

To save the changes, choose *SAVE*.

Download the diagram in the *.erdplus* format locally, as you will use it for the following tasks.

Creating the relational schema

Task 2.

Using the ERDPlus tool for the ER-diagram created in the Task 1 create the SQL script for creating the database relations described in the Task 1.

Creating the relational diagram from the ER-diagram

To get the SQL DDL command based on the created ER-diagram, the relational diagram has to be created as an intermediate step.

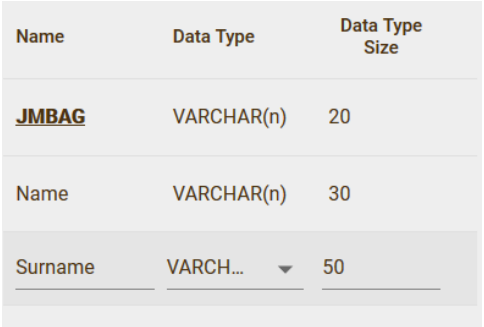
The ERDPlus tool offer the automatic generation of the relational diagram from the ER-diagram. Position yourself to the user space (DOCUMENTS). For the chosen diagram activate the activity menu (⋮ on the right) and choose *Convert to Relational Schema*.

Enter the diagram name (e. g. „example diagram – relational diagram“) and choose *CREATE*. In that way the relational diagram is created.

Open the relational diagram and study it (spread the tables accros the canvas).

Activate the STUDENT table editor by clicking on that table in the diagram. By choosing the *PRIMARY KEY* check whether the primary key is correctly set.

Since all the table attributes are set to the INT type by default, fix the data type for all attributes arbitrarily (as the data types aren't specified in this task).



Repeat the similar procedure for the table ACTIVITY.

Activate the table editor for the ***take_part*** table and study the table properties.

By choosing the primary key *PRIMARY KEY* determine how the primary key is set. Why is it set in that way?

As the table is generated through the connection, the tool automartically sets the relationship key due to its mapping (that we've set to N:N during the ER-diagram modelling). Therefore, the attributes that create the relationship key are automatically added as the relationship attributes and don't have to be manually added.

Note:

As the attributes of this table are „imported“ from the other tables, their type can't be edited here. Moreover, they are also marked as the foreign keys. In case you wish to edit the data type of the attribute, it has to be done in the table which contains the primary key of that foreign key.

Name	Data Type	Data Type Size
JMBAG		(Foreign Key)
ActivityID		(Foreign Key)

Store the changed made over the relational diagram (click on SAVE).

Note:

As for the ER-diagram, by choosing the *MENU > Export Image* the created relational diagram can be exported as an image in the *.png* format and stored locally to the computer.

Moreover, the relational diagram can be exported in the *.erdplus* format and stored locally to the computer. The procedure is identical to the one for the ER-diagram.

Generating the SQL DDL statements from the relational diagrams

The ERDPlus tool enables the automatic generation of the SQL DDL statements from the relational diagrams.

Position to the user space(DOCUMENTS). For the chosen relational diagram activate the activity menu (⋮ on the right) and choose *Generate SQL*, which results with the CREATE TABLE statements.

Observe the generated SQL statements and check whether they match the previously created diagrams.

Select the generated statements and copy them (or choose *COPY*) to store them into a *.sql* (ili *.txt*) file locally on the computer

Generate SQL

```
CREATE TABLE STUDENT
(
  JMBAG VARCHAR(20) NOT NULL,
  Name VARCHAR(30) NOT NULL,
  Surname INT NOT NULL,
  PRIMARY KEY (JMBAG)
);

CREATE TABLE ACTIVITY
(
  ActivityID INT NOT NULL,
  Name VARCHAR(50) NOT NULL,
  PRIMARY KEY (ActivityID)
);

CREATE TABLE takes_part
(
  JMBAG VARCHAR(20) NOT NULL,
  ActivityID INT NOT NULL,
  PRIMARY KEY (JMBAG, ActivityID),
  FOREIGN KEY (JMBAG) REFERENCES STUDENT(JMBAG),
  FOREIGN KEY (ActivityID) REFERENCES ACTIVITY(ActivityID)
);
```

CLOSE (ESC) COPY

Note:
In general, always check the generated SQL DDL statement:

- whether the attribute types are correctly set
- constraints (UNIQUE, NOT NULL)
- are the primary, foreign and altenate keys set correclty
- composite keys
- do the relationhips contain the needed attributes
- are the relationship keys correctly set
- primary and foreign keys of the weak entities

If the tool doesn't manage to create a prt of the diagram or the script automatically, a manual intervention is necessary.

As the scripts generated in the previous example are correct, the Task 2 is finished.

Examples, part 2

Task 3a.
What is changed in the ER-diagram from the Task 1 if OIB (personal identification number) is also stored in the database record for a student? Expand the ER-diagram from the Task 1.

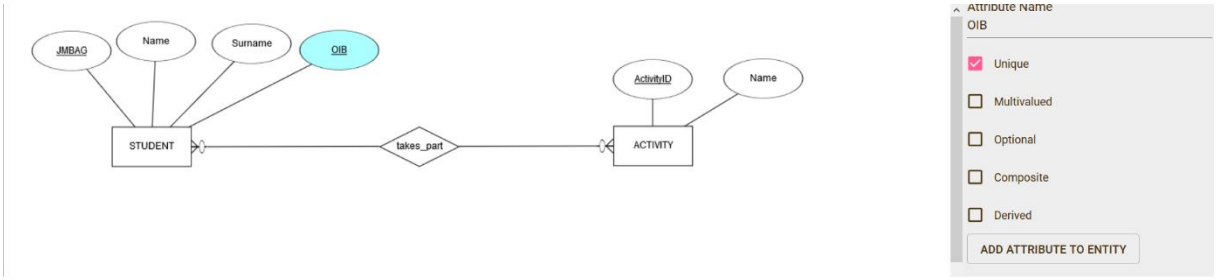
As the ***JMBAG*** and ***OIB*** values are unique for a single student, both attributes can act as a primary key. As the ***JMBAG*** was already set as a primary key, ***OIB*** has to be set as an alternate key.

Alternate entity key

Add the attribute ***OIB*** to the entity STUDENT.

In the attribute editor for ***OIB*** check the option *Unique*. On the diagram both attributes are now underlined.

Save the changes.



Note:

Even though both attributes are seemingly marked the same on the diagram, the tool differentiats one as a primary, and another as an alternate key. Soon we will make sure of that by generating the relational schema and SQL DDL statements.

Task 3b. Generate the relational diagram for the ER-dijagram from the Task 3a.

Generate the relational diagram for the ERM diagram created in the Task 3a.

Observe the table STUDENT and analyze:

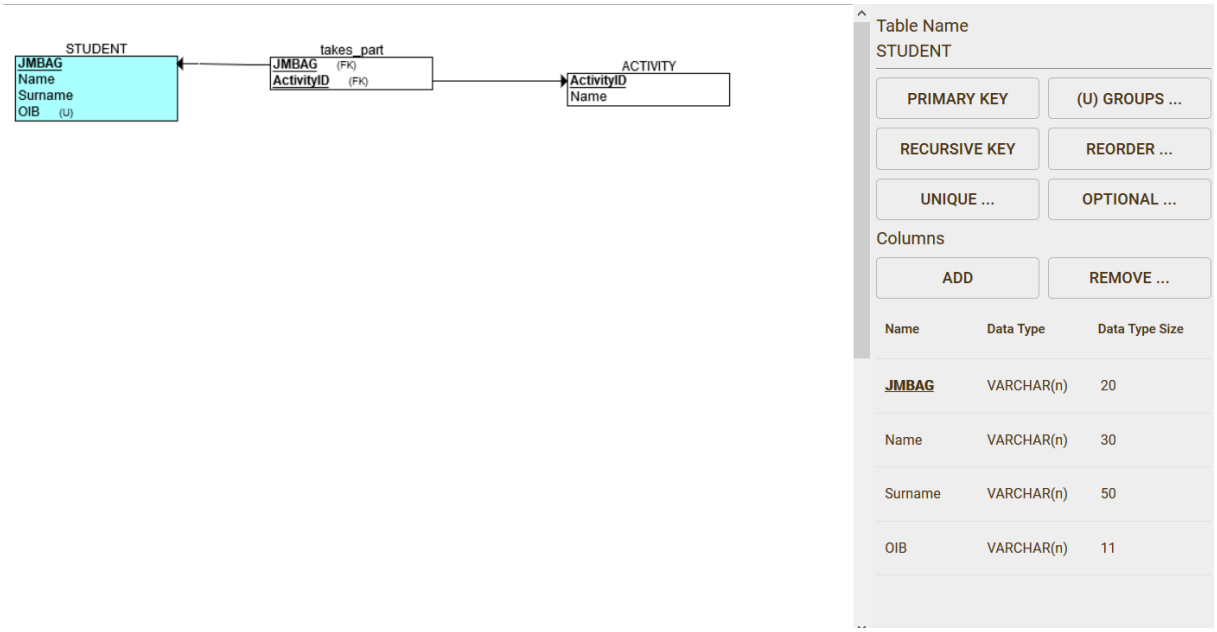
- What is the key in the table STUDENT?
- Next to which attribute in the table STUDENT did the attribute uniqueness label (U) appear?

  The attribute ***JMBAG*** was set first as the primary key, followed by ***OIB***. In the relational diagram it is visible that ***JMBAG*** is the primary key, and the attribute ***OIB*** is declared as *unique* – which is the way of defining the alternate key.

Note:

By each new relational diagram generation, the attribute types are set to INT again and have to be set to the correct types.

Correct the attribute types for all tables and save the changed. The new attribute ***OIB*** should be set to the type VARCHAR(11).



Task 3c. Generate the SQL DDL statements for the relational diagram from the Task 3b

Generirajte i proučite SQL naredbe za kreiranje tablica relacijskog dijagrama iz Zadatka 3b.

Check for the table STUDENT:

- Is the ***JMBAG*** the primary key?
- Is the ***OIB*** unique?

Generate SQL

```
CREATE TABLE STUDENT
(
  JMBAG VARCHAR(20) NOT NULL,
  Name VARCHAR(30) NOT NULL,
  Surname VARCHAR(50) NOT NULL,
  OIB VARCHAR(11) NOT NULL,
  PRIMARY KEY (JMBAG),
  UNIQUE (OIB)
);

CREATE TABLE ACTIVITY
(
  ActivityID INT NOT NULL,
  Name VARCHAR(50) NOT NULL,
  PRIMARY KEY (ActivityID)
);

CREATE TABLE takes_part
(
  JMBAG VARCHAR(20) NOT NULL,
  ActivityID INT NOT NULL,
  PRIMARY KEY (JMBAG, ActivityID),
  FOREIGN KEY (JMBAG) REFERENCES STUDENT(JMBAG),
  FOREIGN KEY (ActivityID) REFERENCES ACTIVITY(ActivityID)
);
```

CLOSE (ESC) COPY

Task 4.

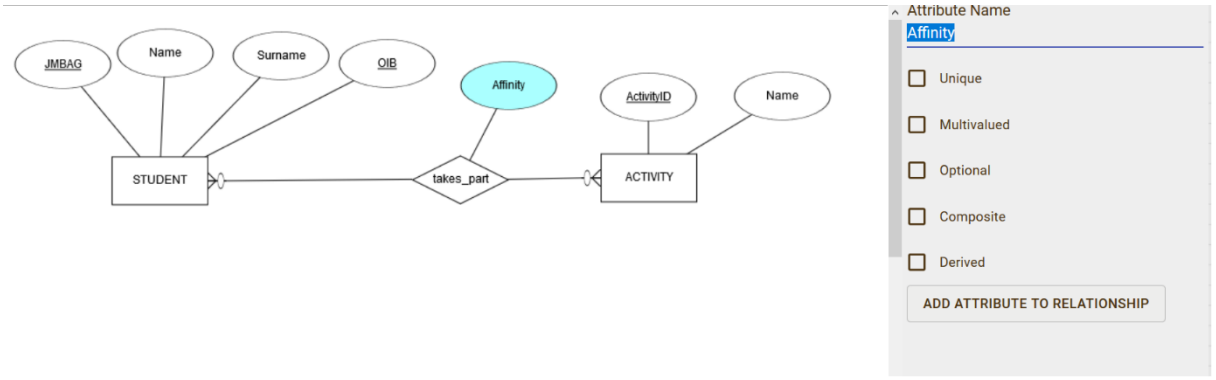
Expand the database model (ER-diagram) created in the Task 3.

For each activity that student takes part in, store the affinity to that activity on the scale 1-5 (1-doesn't like the activity at all, 5-extremely likes to take part in the activity) in the database.

Own relationship attributes

Affinity is the information (attribute) which doesn't describe explicitly neither the entity STUDENT, not the entity ACTIVITY, rather the student's activity, that is – belongs to the relationship ***takes_part***. That is the own relationhsip attribute, which doesn't belong to the key.

Add the attribute ***affinity*** to the relationship ***takes_part***. Select the relationship and choose *ADD ATTRIBUTE*.



Do it yourselfi:

Generate the SQL DDL statements for the database creation. Where is the attribute ***affinity***?

Task 5a.

Extend the ER-diagram from the task 4.

*Let the birth place be stored for **each** student. An integer code and the birth place name are recorded.*

Binary N:1 relationship

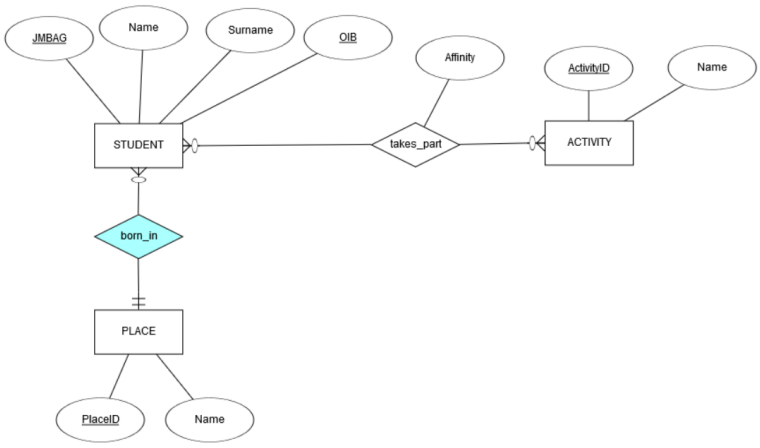
As a student can be born in only one place, entity PLACE should be added and conneced by binary N:1 relationship to the entity STUDENT.

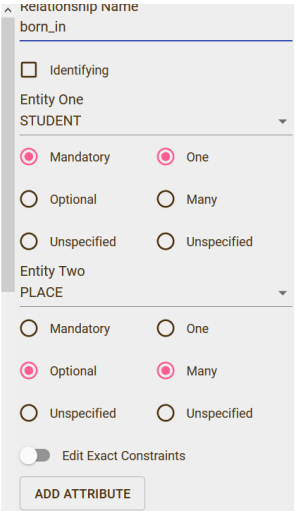
Add the relationhsip ***born_in*** and connect it to the entites PLACE and STUDENT.

The next step is editing the relationship mapping.

A student obviously can be born in at most one place, so the upper boundary is obviously 1; set it to *One*. Since it was emphasized in the task that for ***each*** student the birth place is recorded, that information will always be stored (won't be NULL). Therefore, the lower boundary is set to *Mandatory*.

For the entity PLACE the upper boundary is set to *Many*. As a place in which neither of the students was born on might exist, the lower boundary is set to *Optional*. Pay attention next to which entity did the label *Many* appear, and next to which the label *One* (the dash).





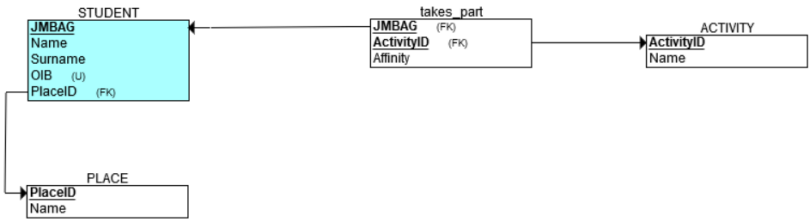
Task 5b. Generate the relational diagram for the ER-diagram from the Task 5a.

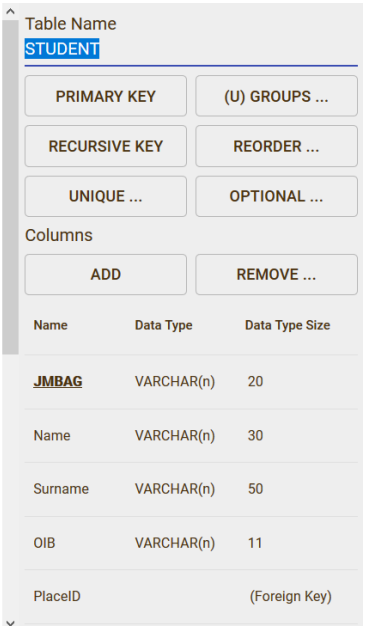
Generate the relational diagram for the ER-dijagram created in the Task 5a.

Observe rthe tables STUDENT and PLACE and analyse:

- Which „new“ attribute did appear in the table STUDENT? Which label is set next to it?
- Note that for the N:1 relationship a new table isn't created from the relationship like in the N:N relationship.

Correct the attribute types for all tables and store the changes.





Task 5c. Generate the SQL DDL statements for the relationship diagram in the Task 5b.

Generate and observe the SQL statements for creating the tables of the relational diagram in the Task 5b.

Check for the table STUDENT:

- Is there an attribute ***codePlace***? Of which type is it?
- Is the attribute ***codePlace*** the foreign key to the relation PLACE?

```
CREATE TABLE ACTIVITY
(
  ActivityID INT NOT NULL,
  Name VARCHAR(50) NOT NULL,
  PRIMARY KEY (ActivityID)
);

CREATE TABLE PLACE
(
  PlaceID INT NOT NULL,
  Name VARCHAR(50) NOT NULL,
  PRIMARY KEY (PlaceID)
);

CREATE TABLE STUDENT
(
  JMBAG VARCHAR(20) NOT NULL,
  Name VARCHAR(30) NOT NULL,
  Surname VARCHAR(50) NOT NULL,
  OIB VARCHAR(11) NOT NULL,
  PlaceID INT NOT NULL,
  PRIMARY KEY (JMBAG),
  FOREIGN KEY (PlaceID) REFERENCES PLACE(PlaceID),
  UNIQUE (OIB)
);
```

```
CREATE TABLE takes_part
(
  Affinity INT NOT NULL,
  JMBAG VARCHAR(20) NOT NULL,
  ActivityID INT NOT NULL,
  PRIMARY KEY (JMBAG, ActivityID),
  FOREIGN KEY (JMBAG) REFERENCES STUDENT(JMBAG),
  FOREIGN KEY (ActivityID) REFERENCES ACTIVITY(ActivityID)
);
```

Task 6a.

Extend additionally the previous ER-diagram.

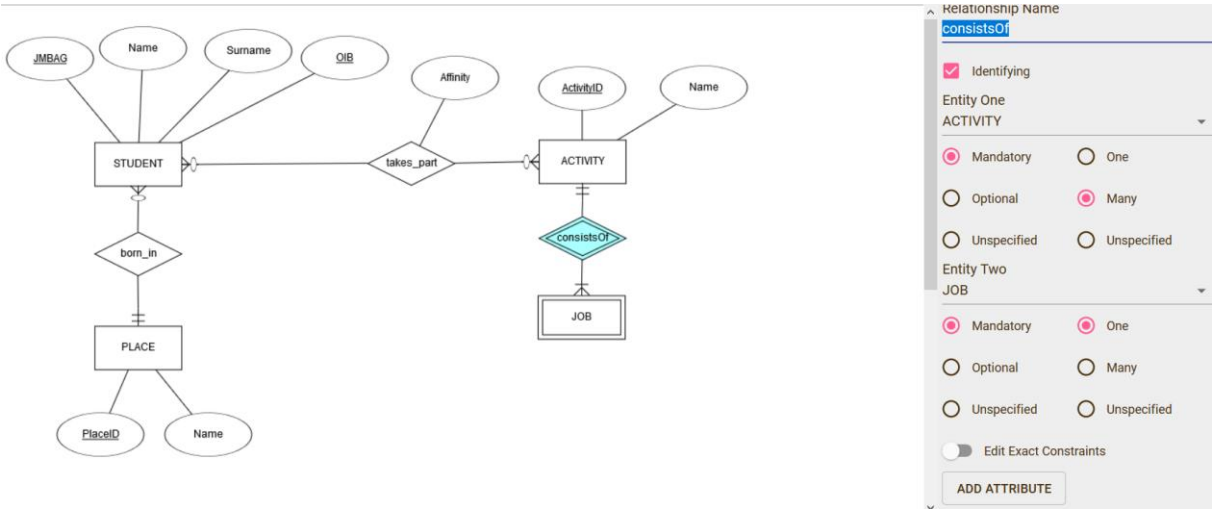
In the database the data about jobs each activity consists of is recorded. Each activity consists of eat least one job. For the job the ordinal number within the activity and its description are recorded. (E.g. for the activity „repetition of materials from Databases“ the first job is „download the lecture from FERweb“, second job is „study the lecture“, third is „download the exercise tasks“ etc.)

Identifying weak entity

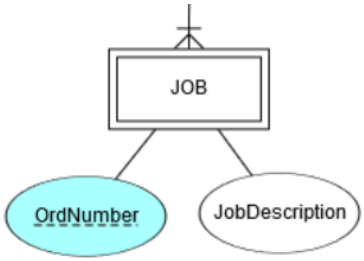
A job which is a part of an activity is obviously modeled as an entity, although the ordinal number isn't enough to idenitfy a single job since all acivities certainly have the job with the ordinal number 1. In order to determine the job unambiguously, it should be known to which activity it „belongs“ to – therefore, it is the identifying weak entity.

Add the new entity called „JOB“ to the ER-diagram. In the entity editor select the entity as weak – notice the change in the entity display on the diagram.

The next step is adding the relationship between the entities ACTIVITY and JOB. Add the relation ***consistsOf*** and connect it with the entities. Edit the relationship mapping: the activity consists of 1 or more jobs, and the job belongs to exactly one activity. As this is the relationship between the „owner“ entity and the weak entity, the relationship has to be marked as identifying – check the box next to *Identifying*. Note that the relationship got another interior rhomb on the diagram. This is a bit different notation than the one shown on the lectures, where the relationship with the weak entity was connected with the arrow instead of the regular line.



Add the attributes ***ordNumber*** and ***jobDescription*** to the entity JOB. Mark the attribute ***ordNumber*** as *Unique* (to specify it is the key); note it is underlined with a dashed instead of a full line– because it is weak idenfiying entity and the attribute is the component of the composite key.



Task 6b.

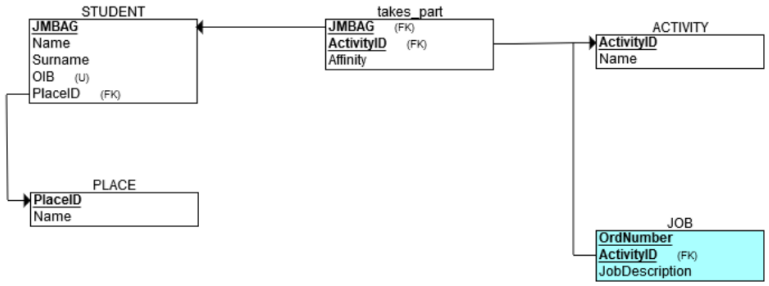
Generate the relational diagram for the ER-diagram for the Task 6a.

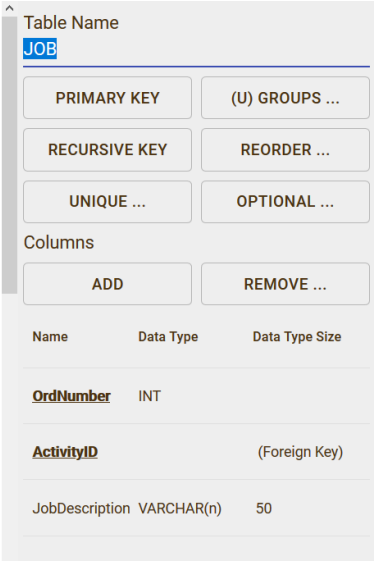
Generate and observer the relational diagram

Analyze:

- What happened with the relationship ***consistsOf*** from the ER-diagram? Why?
- Which foreign key did appear in the relation JOB? Why?
- Click on the button *PRIMARY KEY* on the relation JOB. Which attribute, besides ***ordNumber*** „participates“ in the primary key?

Fix the attribute types (for all tables) and save the changes.





Task 6c.

Generate the SQL DDL statements for the relation diagram from the Task 6b.

Analyze the statement for creating the table POSAO, especially its primary and foreign keys.
Analyze if there are changes in the statements for creating the table ACTIVITY after adding the weak entity to which it belongs. Why?

Generate SQL

```
CREATE TABLE ACTIVITY
(
  ActivityID INT NOT NULL,
  Name VARCHAR(50) NOT NULL,
  PRIMARY KEY (ActivityID)
);

CREATE TABLE PLACE
(
  PlaceID INT NOT NULL,
  Name VARCHAR(50) NOT NULL,
  PRIMARY KEY (PlaceID)
);

CREATE TABLE JOB
(
  OrdNumber INT NOT NULL,
  JobDescription VARCHAR(50) NOT NULL,
  ActivityID INT NOT NULL,
  PRIMARY KEY (OrdNumber, ActivityID),
  FOREIGN KEY (ActivityID) REFERENCES ACTIVITY(ActivityID)
);
```

```
CREATE TABLE STUDENT
(
  JMBAG VARCHAR(20) NOT NULL,
  Name VARCHAR(30) NOT NULL,
  Surname VARCHAR(50) NOT NULL,
  OIB VARCHAR(11) NOT NULL,
  PlaceID INT NOT NULL,
  PRIMARY KEY (JMBAG),
  FOREIGN KEY (PlaceID) REFERENCES PLACE(PlaceID),
  UNIQUE (OIB)
);

CREATE TABLE takes_part
(
  Affinity INT NOT NULL,
  JMBAG VARCHAR(20) NOT NULL,
  ActivityID INT NOT NULL,
  PRIMARY KEY (JMBAG, ActivityID),
  FOREIGN KEY (JMBAG) REFERENCES STUDENT(JMBAG),
  FOREIGN KEY (ActivityID) REFERENCES ACTIVITY(ActivityID)
);
```

Task 7a.

Further expand the previous ER-diagram.

Let the database record the data whether a student follows another student on a social network (doesn't matter which one). One student can follow zero or more other students. A student can be followed by zero or more other students.. In case the student A follows student B, the student B doesn't necessary have to follow student A.

Reflexive N:N relationship

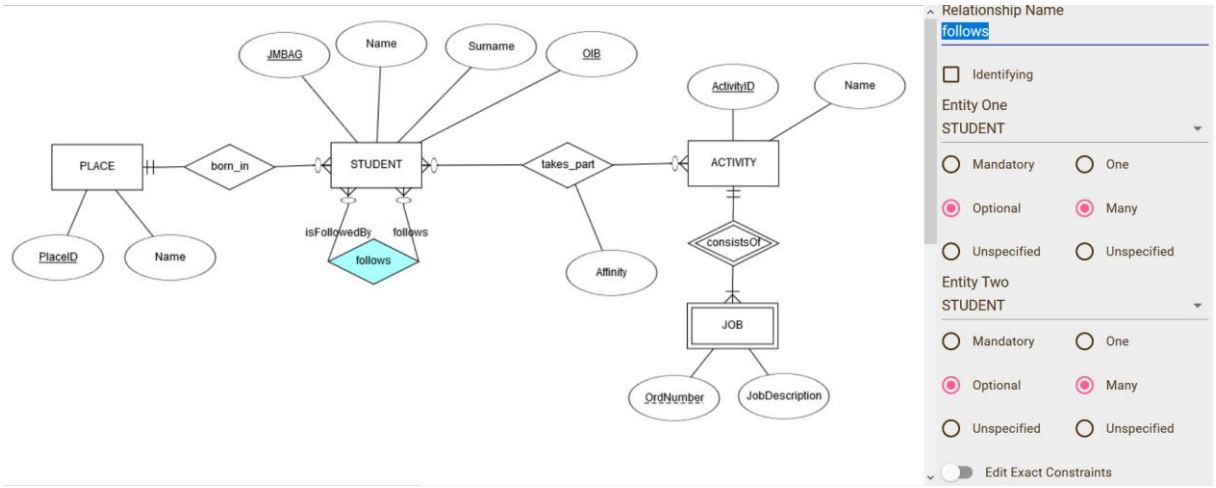
As a student can follow 0 or more students, and can be followed by 0 or more students, it is reflexive N:N relationship over the entity STUDENT. In both cases, the lower boundary is e 0 („***zero*** or more“).

Add the relationship ***follows*** to tje ER-diagram. Set the entity STUDENT as *Entity One* and *Entity Two*, lower boundary to *Optional*, upper boundary to *Many*.

The relationship editor offers the choice *Edit Exact Constraints* on the bottom, by which you edit the exact relationship settings. Here the roles can be named (ROLE) for the entities connected by the relationship, which in this case can be very practical.

Let's try it!

Since the mapping of the relationship ***follows*** is 0..N in both cases, it doesn't matter how the roles will be set; e. g.., add to the first entity the role „follows“, and to the second one „isFollowedBy“. Note the labels that have appeared next to the connecting lines of the relationship ***follows*** to the entity STUDENT.



Save the changes.

Task 7b.

Generate the relational diagram for the ER-diagram from the Task 7a.

Generate and observe the relational diagram.

Analyze:

- What was created from the relationship ***follows*** from the ER-diagram?
- How many attributes does the table ***follows*** have?
- Which is the primary key of the table ***follows***?
- How many foreign keys does the table ***follows*** have? To which tables?
- How are the attributes of the table ***follows*** named? What happened to the name of one of the attributes?

Note:

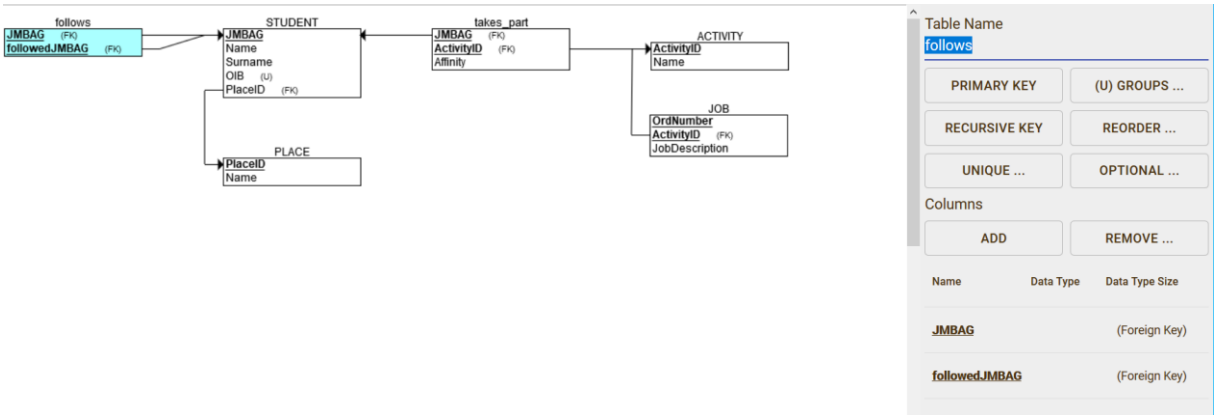
Since the table can't have two attributes of the same name (***JMBAG***), the tool gave the attribute names appendices ***_1*** and ***_2*** respectively.

As it is still difficult to determine from the names ***JMBAG_1*** i ***JMBAG_2*** which JMBAG belongs to the student which „follows“, and which to the one „being followed“ (as it is not the same – the relationship isn't reciprocal!), the tool additionally expanded the name of one of the attributes with the relationship (table) name.

It means that in the case of a „good“ relationship naming (using „best practices“, e. g. by using verb) the renamed attribute *might* get a semantically sufficiently „informing“ name. Though, it is recommended to approach the automatically generated attribute names critically. Be careful when (re)naming the attributes of the reflexive relationship to make sure that it is clear from the attribute name to which role of the same entity it belongs to. If needed, by manual intervention in the relational diagram the attribute names can be edited.

For example. In this case the attributes got the names ***JMBAG_1*** and ***followsJMBAG_2***. Does the attribute ***followsJMBAG_2*** belong to the student that follows, or is it the JMBAG of the followed student? Not clear!

Thus, the schema of the table ***prati*** will be changed to (***JMBAG***, ***followedJMBAG***), where ***JMBAG*** denotes the JMBAG of the student that follows the student with JMBAG ***followedJMBAG***.



Fix the attribute types (for all tables ☹) and save the changes.

Task 7c.

Generate the SQL DDL statements for the relational diagram from the Task 7b.

```
CREATE TABLE follows
(
  JMBAG VARCHAR(20) NOT NULL,
  followedJMBAG VARCHAR(20) NOT NULL,
  PRIMARY KEY (JMBAG, followedJMBAG),
  FOREIGN KEY (JMBAG) REFERENCES STUDENT(JMBAG),
  FOREIGN KEY (followedJMBAG) REFERENCES STUDENT(JMBAG)
);
```

Analizirajte naredbe za kreiranje
tablice PRATI i njezine primarne
i strane ključeve.

Task 8a.

Expand the previous ER-diagram.

Assume there exists a programme of a volunteering student mentoring, where the students can apply voluntarily for accepting the help from the student-mentor or for helping (mentoring) other students. Student doesn't have to take part in the programme – but in case of enrolling can be a mentor, have a mentor, or both. A student can have only one mentor. A student can be a mentor to one or more other students.

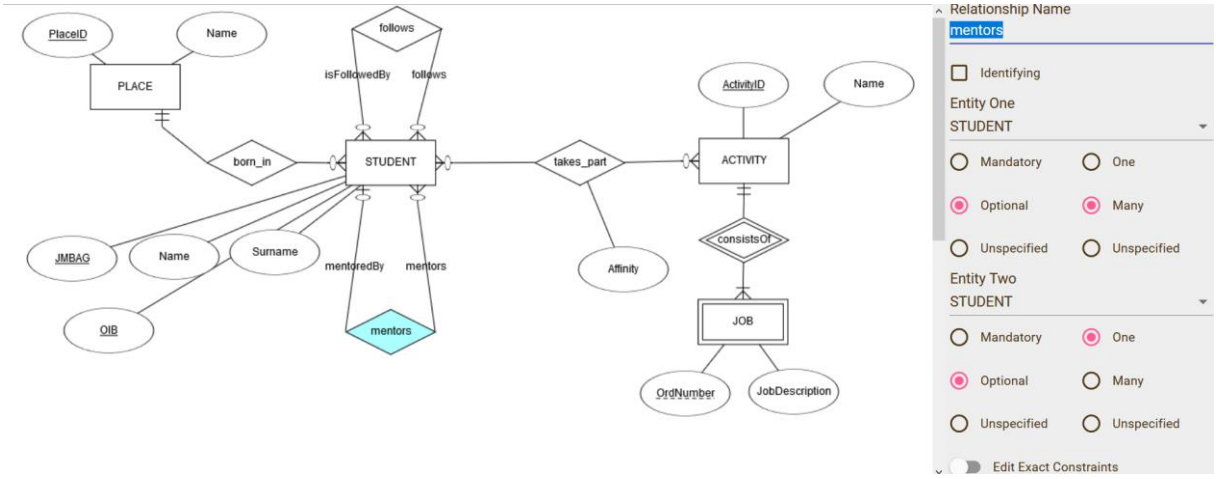
Reflexive N:1 relationship

Since the student can have only one mentor, but can be a mentor to multiple students, it is a reflexive N:0..1 relationship over the entity STUDENT. As both mentoring and having a mentor are optional, in both cases the lower boundary is 0.

Add the relationship ***mentors*** to the ER-diagram. Set the entity STUDENT for *Entity One* and *Entity Two*. Then model the „student mentors 0 or N students“: set the boundary for one end of the relationship to *Optional*, and the upper to *Many*.
The next step is modeling „student can be mentored by 0 or 1 students“: to the other end of the relationship set the lower boundary to a *Optional*, and the upper *One*.

In the exact properties edit the roles:

- for „student mentors 0 or N students“ set the ROLE to „mentoring“
- for „student can be mentored by 0 or 1 students“ set the ROLE to „isMentoredBy“.



Task 8b.

Generate the relational diagram for the ER-diagram from the Task 8a.

Generate and observer the relational diagram.

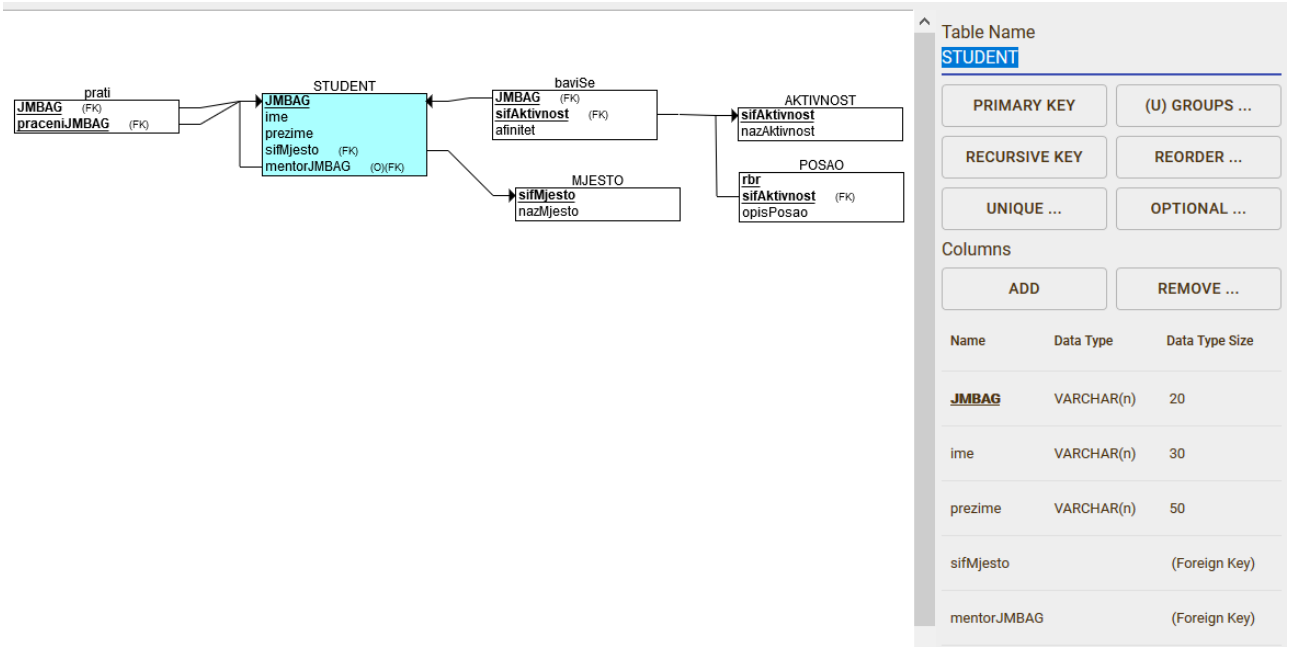
Analyze:

- What was created from the relationship ***isMentoring*** from the ER-diagram?
- Why wasn't a new table created (in regards to the relational diagram from the Task 7b)?
- Which new attribute (and foreign key) did appear in the table STUDENT?
  Which attribute (from which table) referents that foreign key?
- Note the label *(O)* next to the new attribute– its meaning is *Optional*. What does it mean for the possible values of that attribute and the possibility of NULL values? Why?
- What is the name of the new attribute in the table STUDENT? What did happen to its name?

Note:

Similar to the Task 7b, as the table (STUDENT) can't have to attributes with the same name (***JMBAG***), the tool automatically renamed the new attribute by adding the name of the relationship and the underscore (***mentors_***).
THINK – does the new name of the attribute fit semantically? How would be interpretthe attributes ***JMBAG*** and ***mentors_JMBAG*** in the tabe STUDENT? This name could take us to the wrong conclusion that the student with JMBAG value ***JMBAG*** *mentors* the student with the JMBAG value ***mentors_JMBAG***, which is **wrong**! The role of the new attribute in the table STUDENT is the JMBAG of the student's mentor (can have at most one mentor), and not the student which is mentored.

In cases like this a good practice is to **MANUALLY CHANGE** the name of the new attribute into something semantically more correct and logic, e. g.. to ***mentorJMBAG***.



Change the attribute types (for all tables ☹) and save the changes.

Task 8c.

Generate the SQL DDL statements for the relational diagram from the Task 8b.

Analyze the statements for creating the table STUDENT and its foreign keys.

Note:

- Nexto to the attribute ***mentorJMBAG*** there is no NOT NULL constraint. Why?
- The definition of a new foreign key appeared (***mentorJMBAG***) with the role of the reflexive referent of the tble STUDENT to itself.

```
CREATE TABLE STUDENT
(
  JMBAG VARCHAR(20) NOT NULL,
  Name VARCHAR(30) NOT NULL,
  Surname VARCHAR(50) NOT NULL,
  OIB VARCHAR(11) NOT NULL,
  PlaceID INT NOT NULL,
  mentorJMBAG VARCHAR(20),
  PRIMARY KEY (JMBAG),
  FOREIGN KEY (PlaceID) REFERENCES PLACE(PlaceID),
  FOREIGN KEY (mentorJMBAG) REFERENCES STUDENT(JMBAG),
  UNIQUE (OIB)
);
```