The **Graph** extension for the **JEntigrator**.

Alexander Imas, 2016, Darmstadt.

1.Introduction.

The **Graph** extension serves entities having relations. Let us consider the example from the **movie** database (***Neo4j**'s* demo database). The actor **Tom Hanks** acted in the movie **You have got a mail** as **Joe Fox.** More formally, the entity **Tom Hanks** has the **ACTED_IN** relation to the entity **You have got a mail.** We can see, that this connection has three parameters: the outgoing entity **Tom Hanks**, the ingoing entity **You have got a mail** and the instance of the **ACTED_IN** relation. The instance of the relation has the property **Joe Fox.** Follow this example, we can define base terms needed for the **graph** extension:

- **Node -** an object having relations.
- **Edge -** the relation object.
- **Bond –** an instance of the relation connecting outgoing and ingoing node.
- **Bond detail –** an object linked to the bond as a bond property.

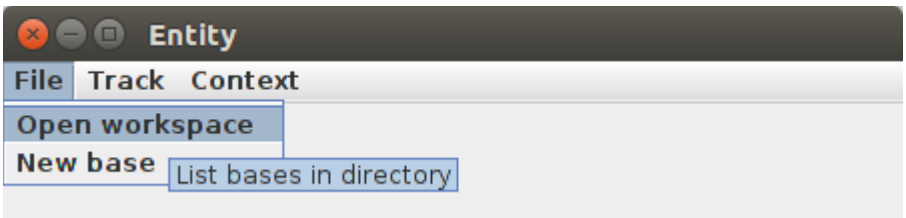
In the **JEntigrator** an **edge** is implemented as an entity having the **edge** type. The **node** feature can be assigned to any entity as the **node** facet. The **bond detail** feature is also an assignable facet. A **bond** is presented as a record within the **edge** and **node** entities. The **movie** database (***Neo4j**'s* demo database) illustrates the usage of the **graph** extension.

2.Open the **Movie** database.

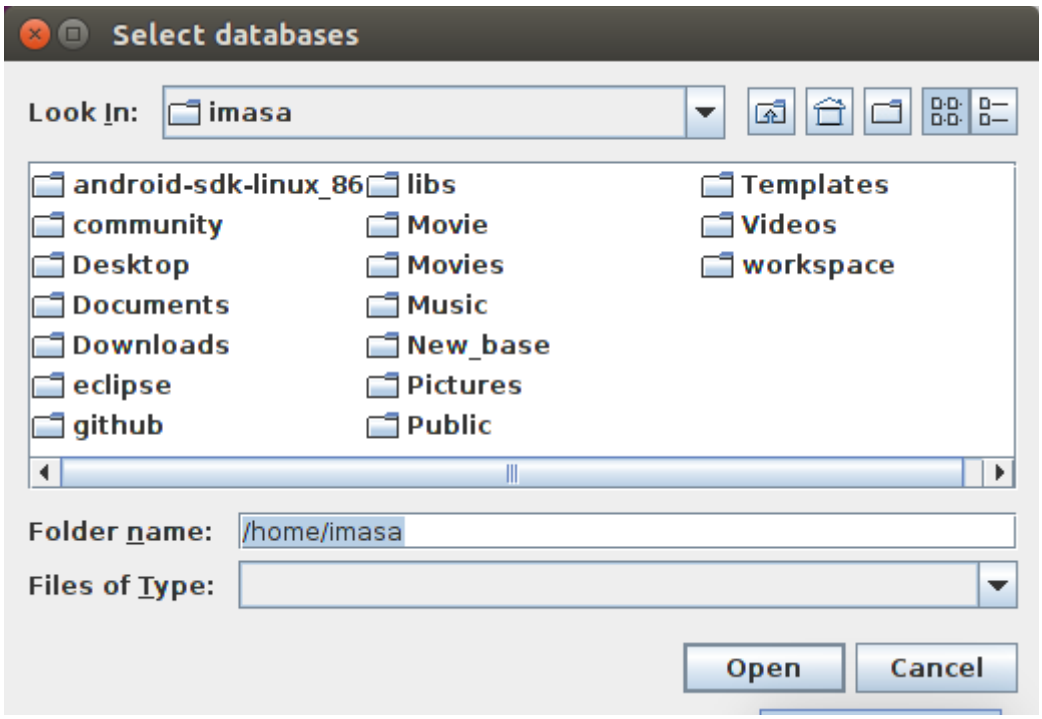
Download the **movie** database archive and extract it locally, for example, in the home directory. Navigate to the directory where the **JEntigrator** is located and run it (replace the jar file name with your version)**:**

$ java -jar JEntigrator-0.0.2-20160826.163236-4-jar-with-dependencies.jar

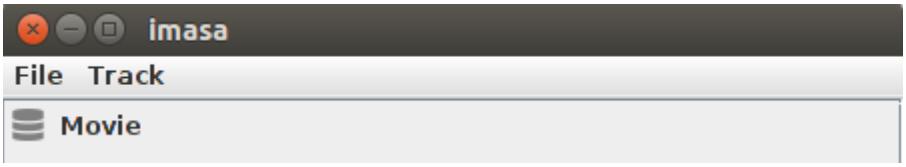
Click the **Open workspace** menu item from the **File** menu.



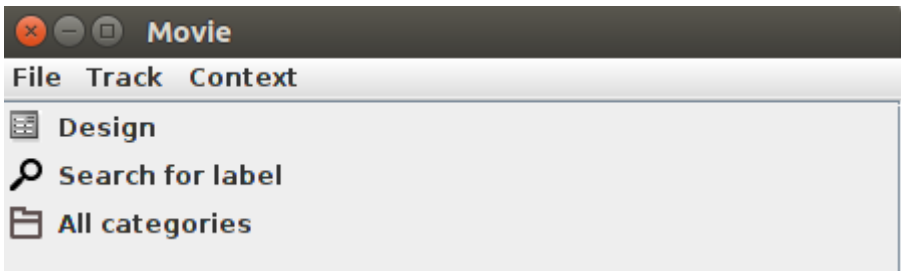
Navigate to the directory where the **Movie** database is located and click the **Open** button.



Click the **Movie** database item.

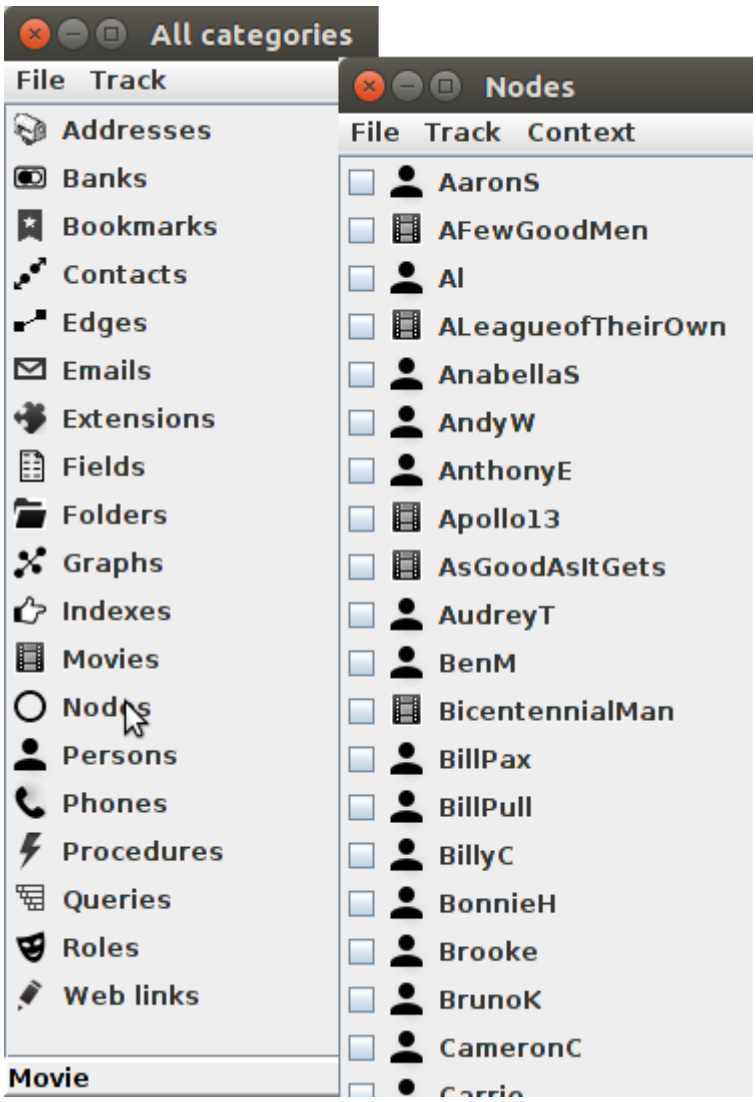


The database navigator appears.

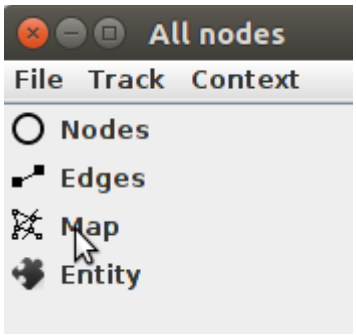


2.Browse the **Movie** database.

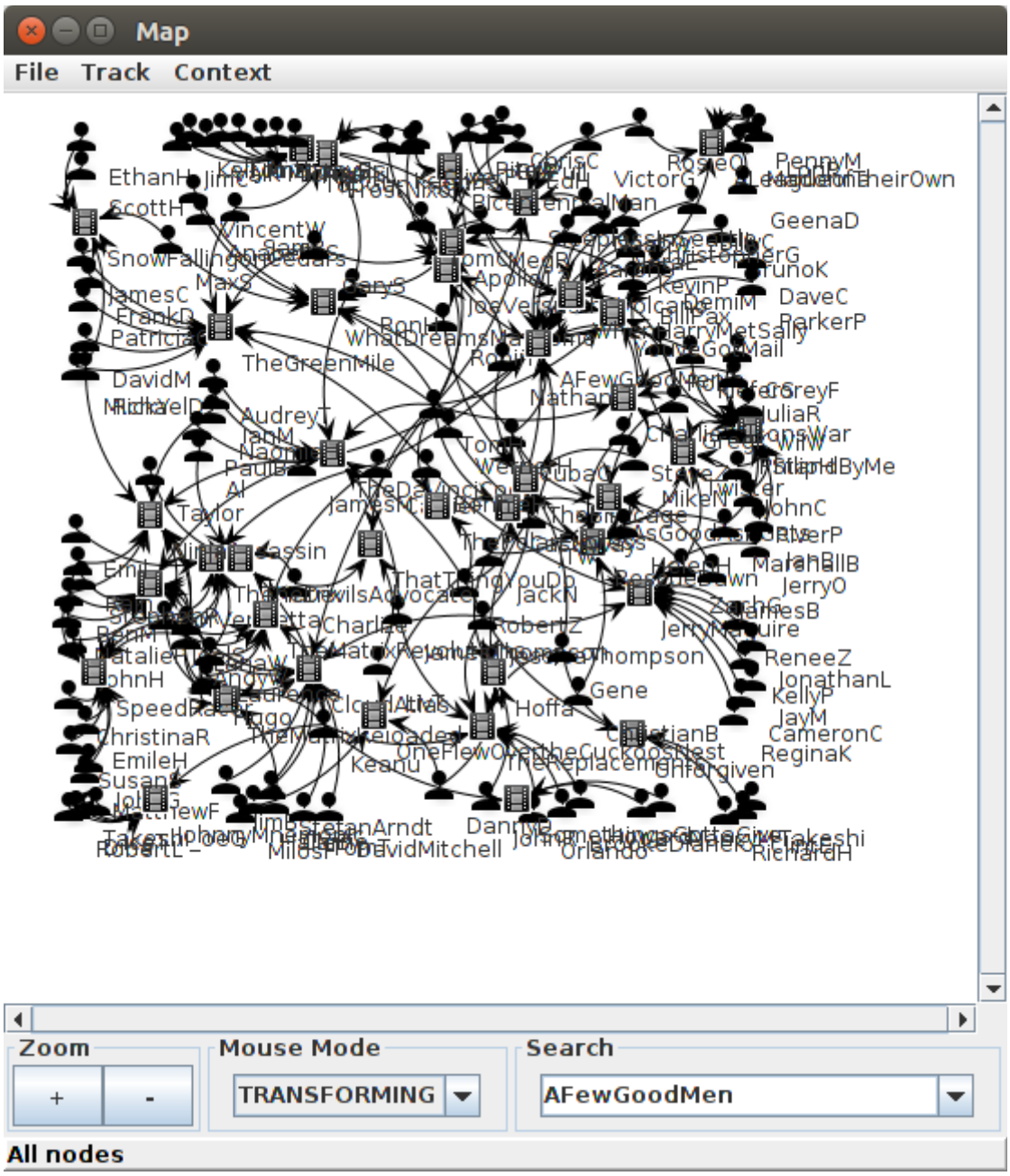
Click on the **All categories** item in the base navigator context displays the list of all entity types in the database. The **Movie** database includes **community, graph** and **movies** extensions , so all types from these extensions will be shown in the list. Click on the category item opens the list of all entities of this type. Example below shows all nodes. You may want to investigate edges , movies or roles.



The database contains the **All nodes** graph entity. You may navigate to the **All nodes** graph facet follow the **All categories** → **Graphs** → **All nodes** → **Graph** path. Click the **Map** item to display the graph.

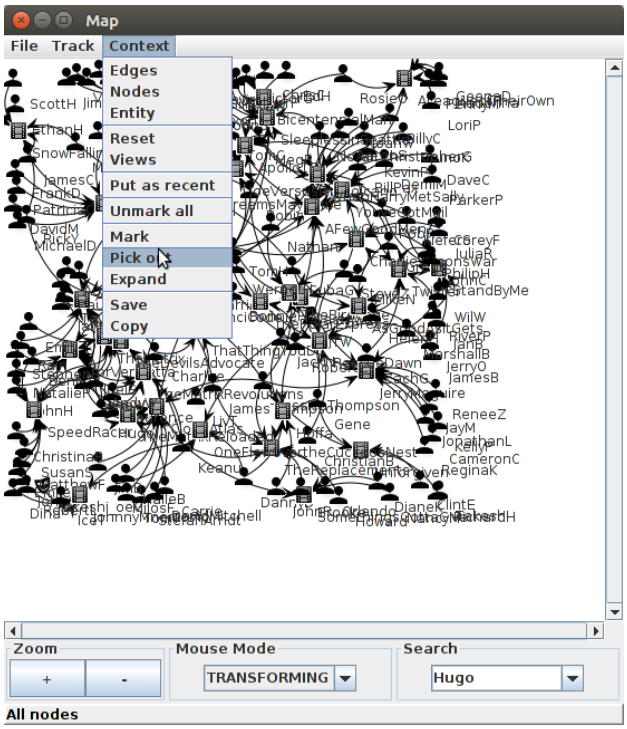


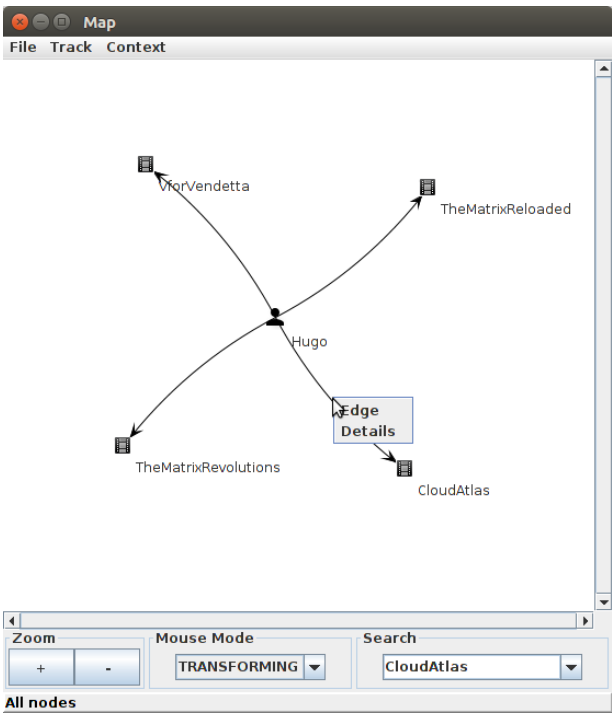
The graph visualization exploits the open source graph library JUNG2. We can see that the original graph demonstrates the known “hairball effect” where too many edges link too many vertexes. Fortunately there are tools to present the necessary information in clear and comprehensive way.



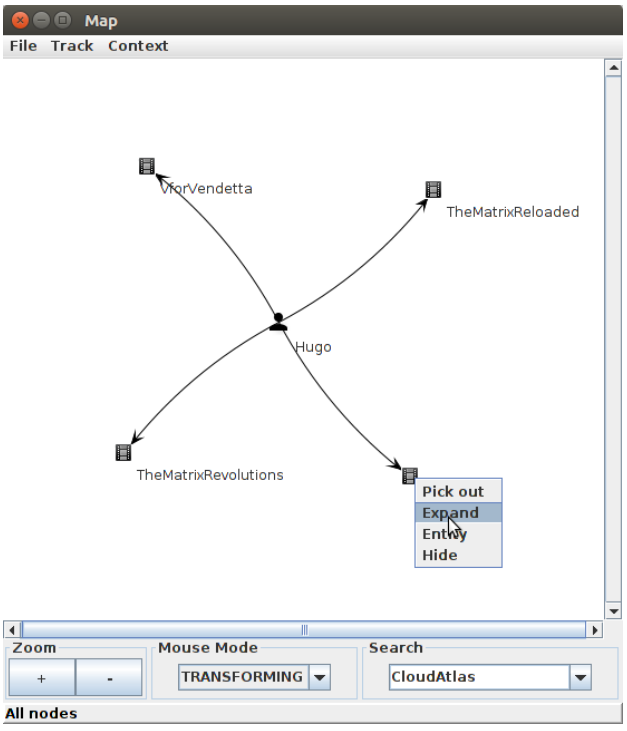
2.1. Browse a node.

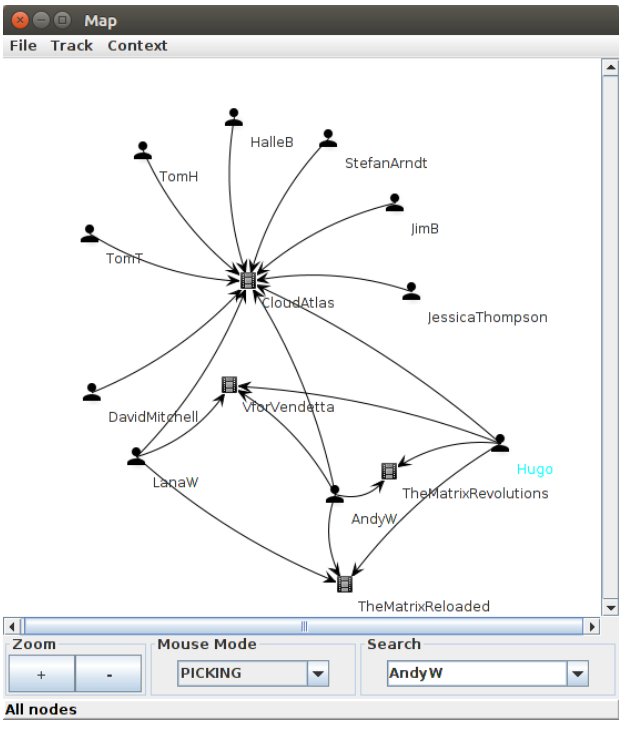
We can investigate a certain node. Select the **Hugo** node in the **Search** selector and click the **Pick out** item in the context menu. The relations of the **Hugo** node will be displayed. The right click on the node or the edge pops up the context menu.



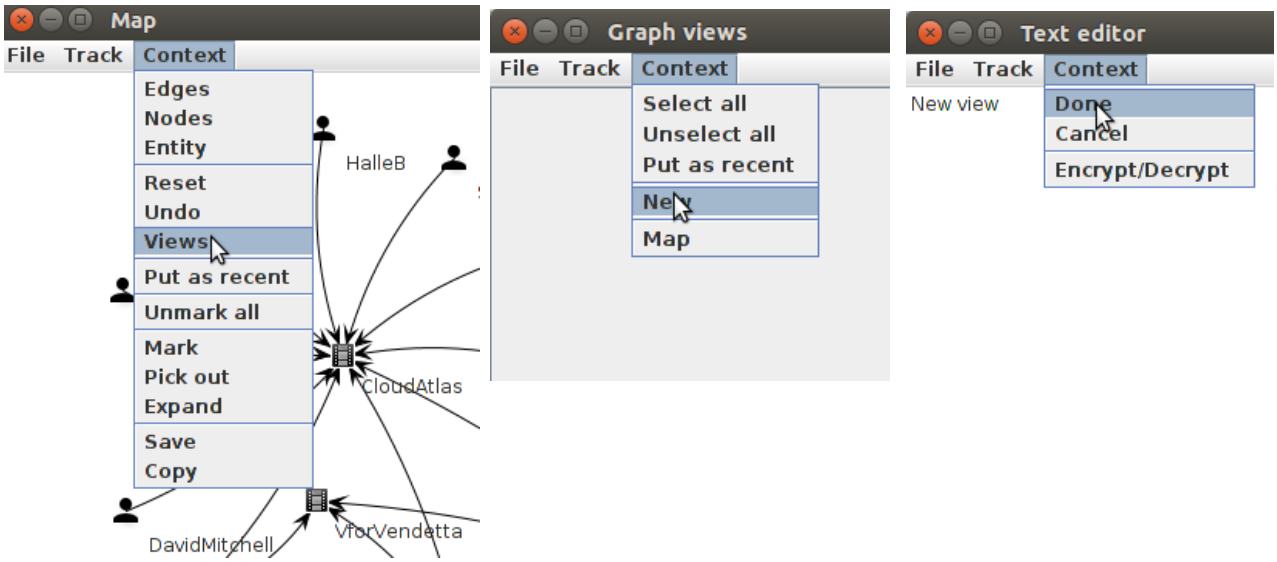


We can expand any node and navigate through the graph.

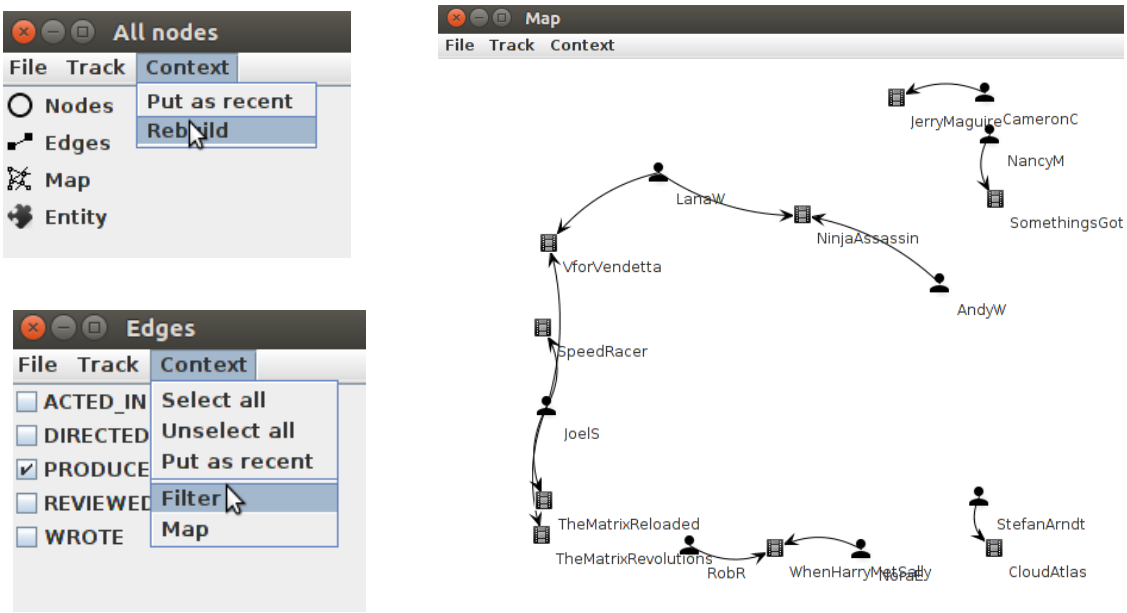




It is possible to save the interim picture as a view within the database using the **Views** menu item. To save the graph as a png picture use the **Save** menu item.



It is possible filter out types of edges. Navigate to the **graph** facet, rebuild the graph and open the edges list. Select desirable edge types and click the **Filter** menu item.

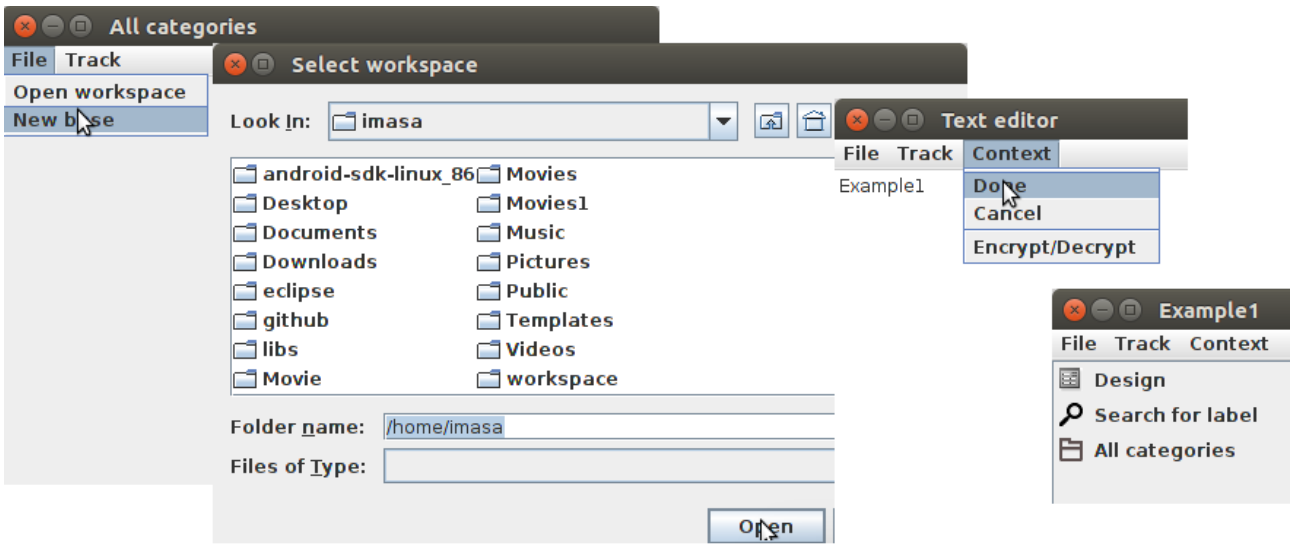


3. Create a graph database from scratch.

The primary task of the database designer is to reflect real data objects into the database. The example below shows how to create a simple database including related objects. The database has three objects **Grandmother**, **Mother** and **Daughter** tied with the relation **CHILD_OF.** First of all we have to create a new empty database.

3.1. Create a new database.

Run the **JEntigrator** as described above and create a new database using the **File** menu.

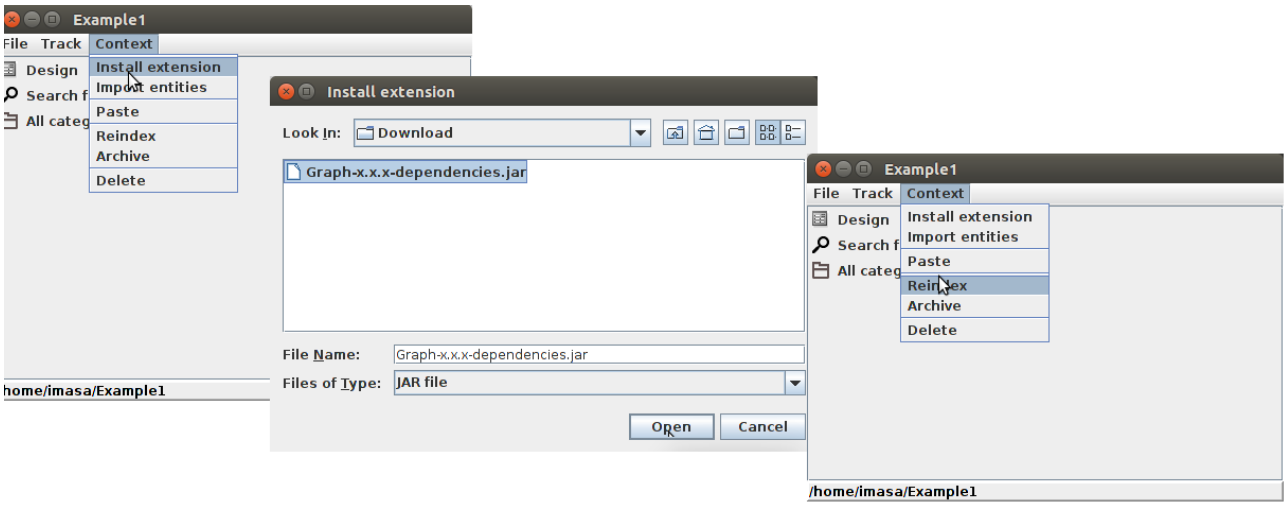


3.2. Install the ***graph*** extension.

Download the the graph extension(replace **x.x.x** with the last version) :

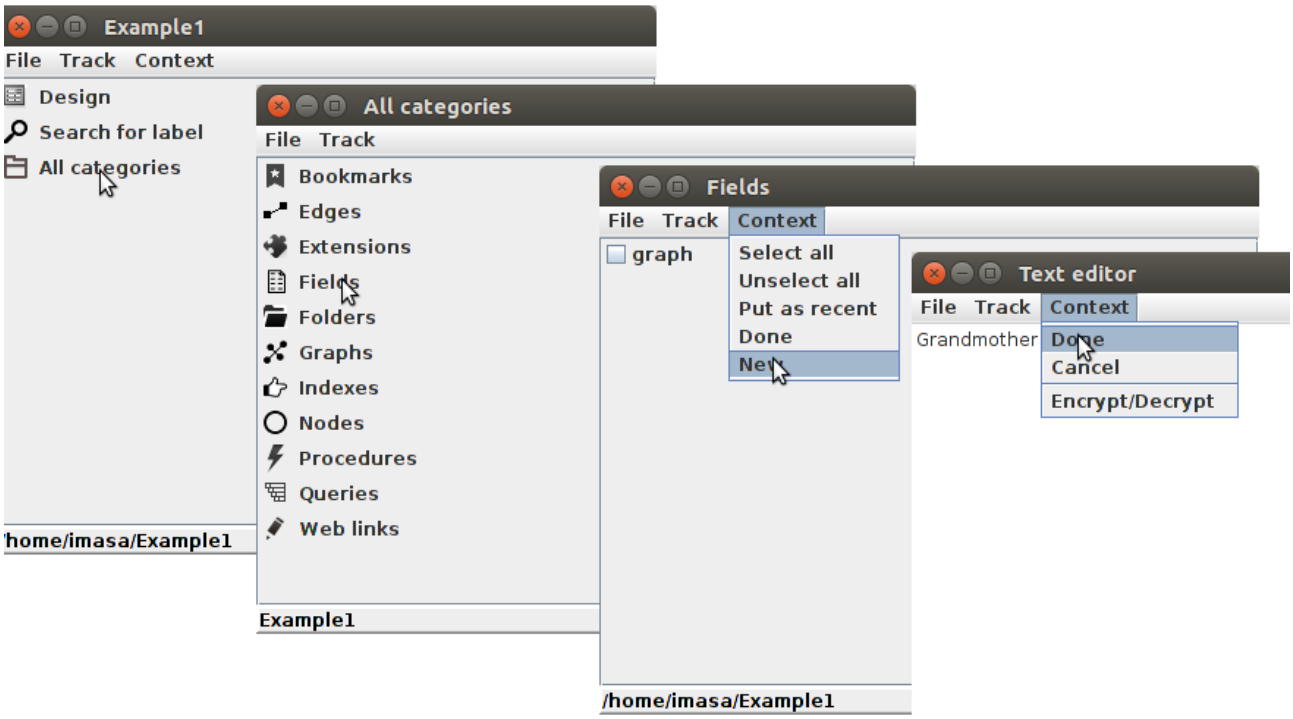
https://oss.sonatype.org/service/local/repositories/releases/content/com/github/imas-alex/Graph/x.x.x/Graph-x.x.x-dependencies.jar

Install the extension from the context menu and restart the **JEntigrator.**

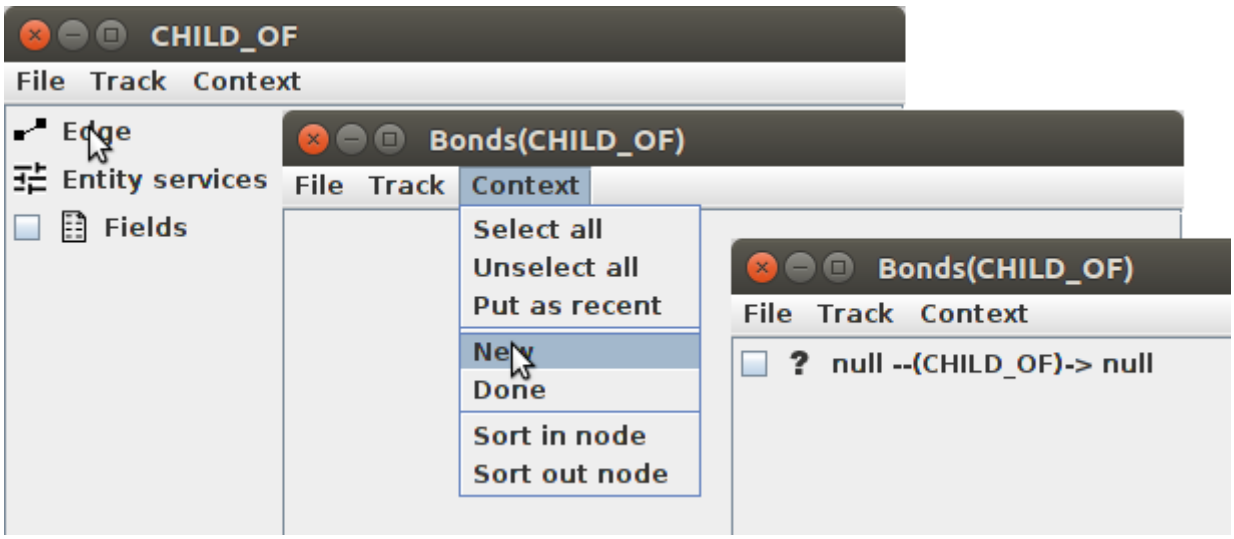


3.3. Create data.

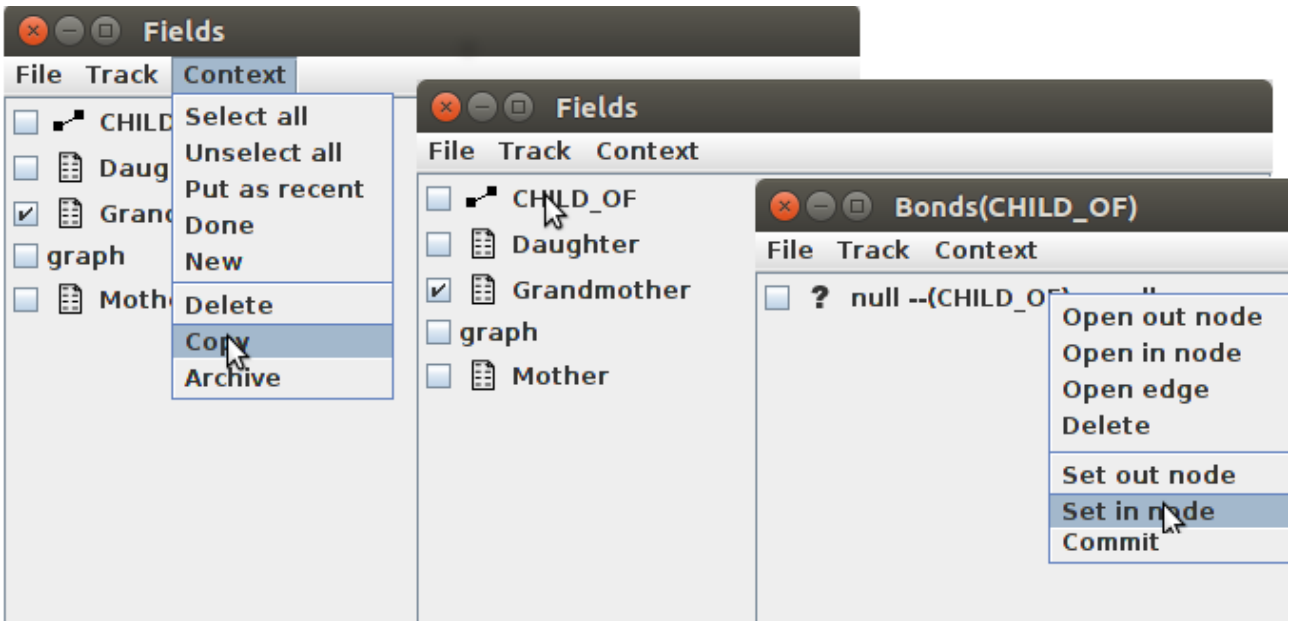
Open the **All categories** list from the **Base navigator** context. Open the **Fields** category. Click the **New** item, type the **Grandmother** in the text editor and click the **Done** item. Use the **Back** item from the **Track** menu to return in **Fields** category. Repeat last operations for **Mother** and **Daughter** .



Use the **Show track** item from the **Track** menu to return in the **All categories** list and open the **Edges** category. Create the **CHILD_OF** edge. Open the **Edge** facility and create a new (empty) bond.



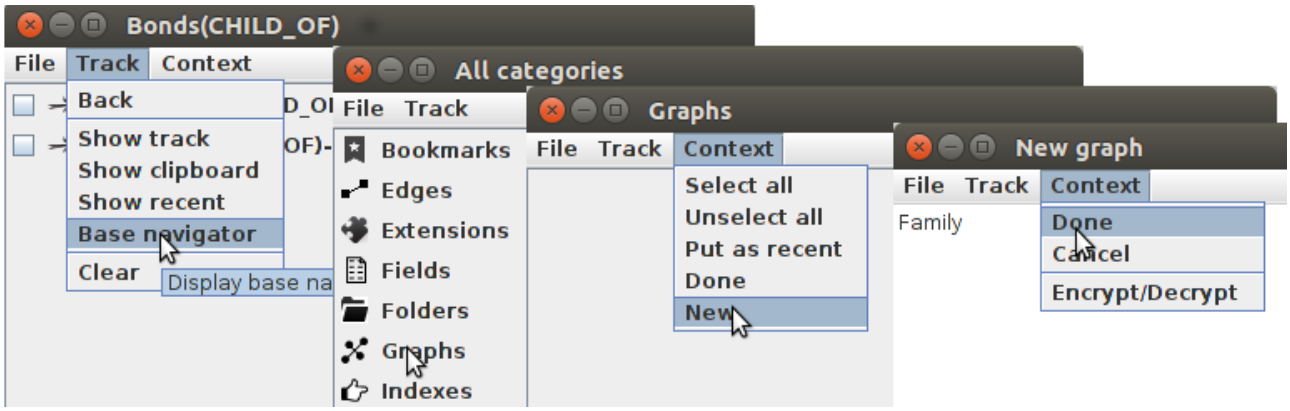
Now we have to add outgoing and ingoing nodes to the bond. Go to the **Fields** category and copy the **Grandmother** node into the clipboard. Then open the **CHILD_OF** edge again and make the right click on the bond item. Click the **Set in node** item on the pop up menu. Then return to **Fields**, copy the **Mother** node and set it as outgoing node. Commit the bond. Create the second bond and set the **Mother** → **Daughter** relation in the same way. After that we have our related data inserted in the database.



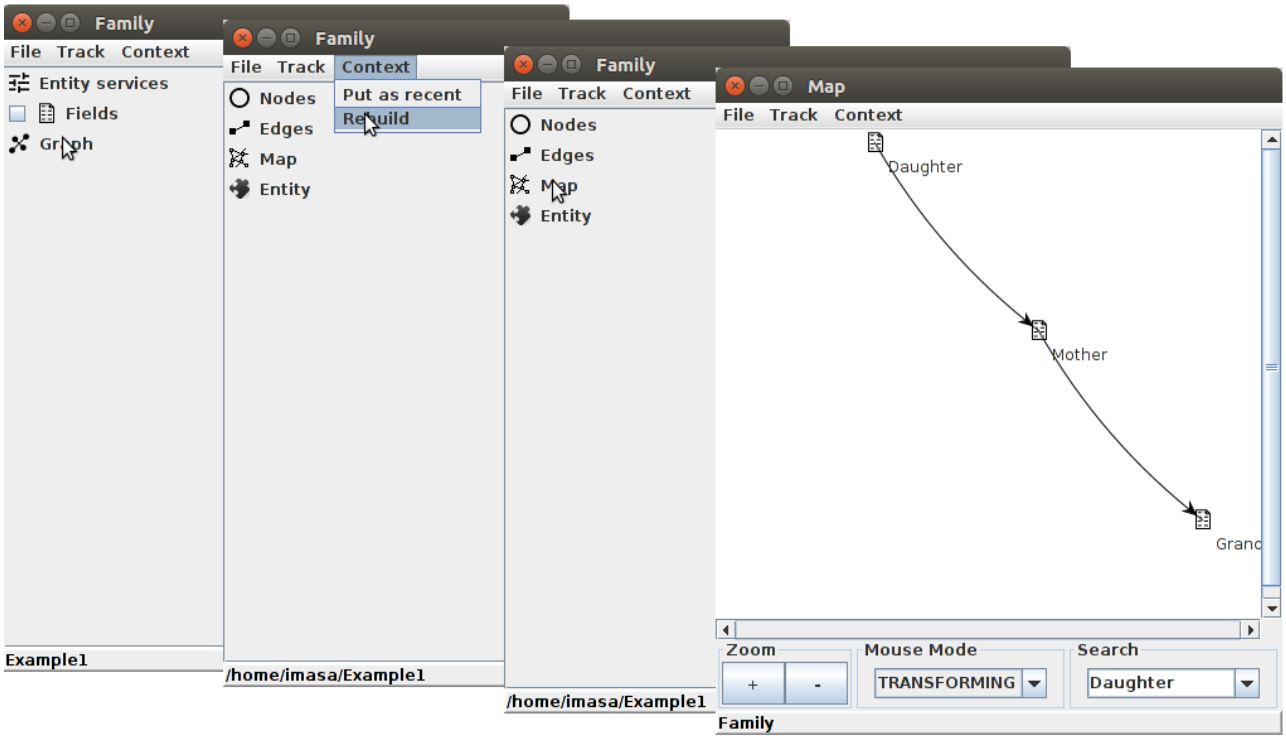
3.4. Create a graph.

An entity of the **graph** type in the **JEntigrator** is NOT the primary data container. All data together with their relations are kept in nodes and edges. A **graph** is rather a view that represents the data in the convenient form. Many different graphs can include the same nodes or a graph can include only subset from the set of tied nodes.

To create the new graph **Family** follow the picture below.



Then navigate to the **Graph** facet and rebuild the graph. Open the graph.



4. Conclusion.

The **Graph** extension introduces facets **Graph** , **Node, Edge** and **Bond detail**. Relations are implemented as bond records enclosed within **node** and **edge** entities. Any entity can be linked to a bond as a **Bond detail.** The **graph** entity is a view representing related entities in the graphical form. A user creates, edits and navigates graphs from the GUI without any coding.