Database-manager

Presentation :

Toaster's database unit is a ros node design to offer services in order to easily create and manage a sql database dealing with Toaster's facts.

Installation :

First of all you need to install two libraries at your home/devel

TyniXML →to « install » this librarie follow instructions from this link

http://khayyam.developpez.com/articles/cpp/tinyxml/

SQLite → go to SQLite home page and download **sqlite-autoconf-*.tar.gz**

https://www.sqlite.org/download.html

then follow these steps

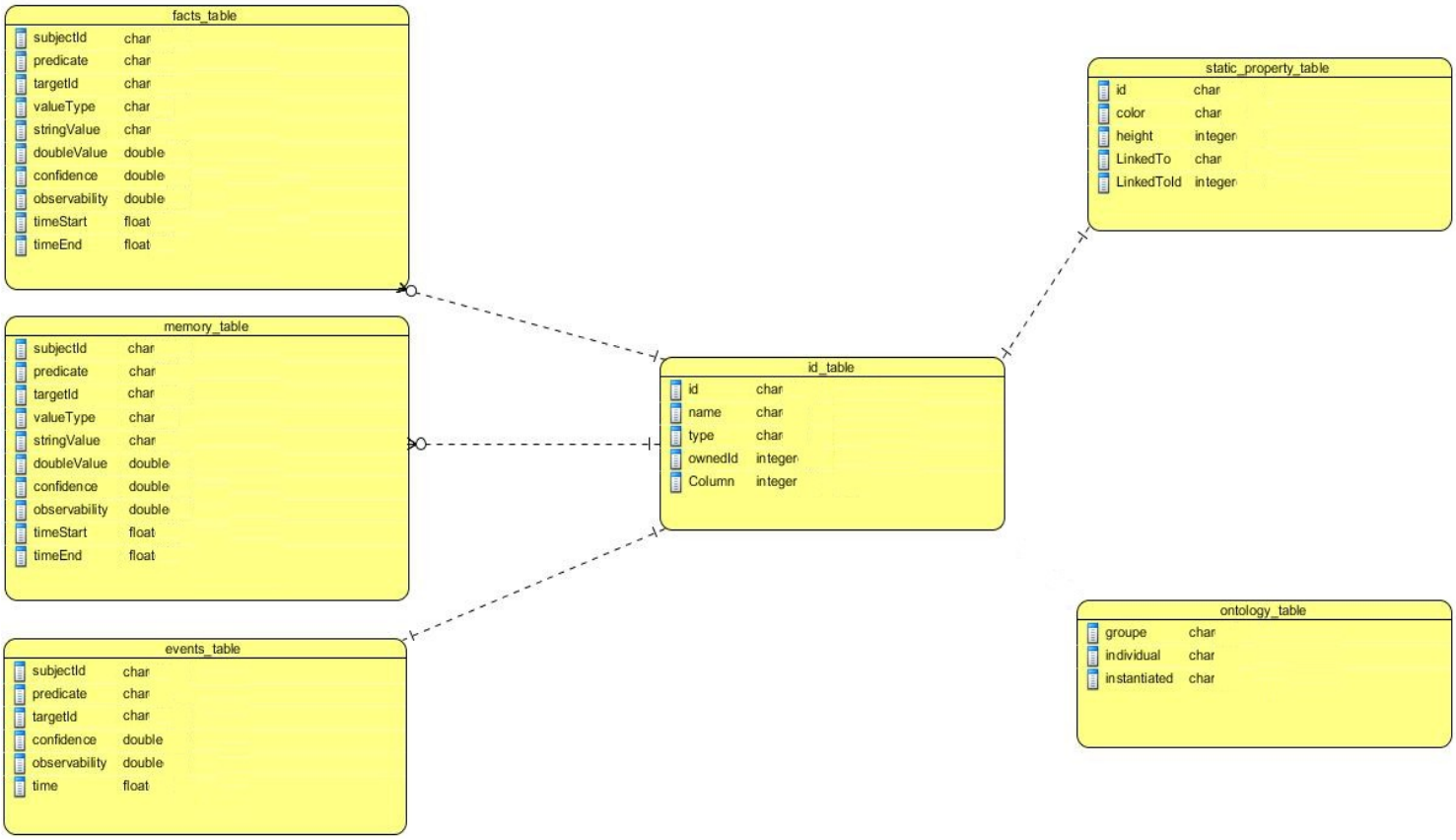
```
$ tar xvfz sqlite-autoconf-3071502.tar.gz

$ cd sqlite-autoconf-3071502

$ ./configure –prefix=/usr/local

$ make

$ make install
```

Launch and architecture :

To start database_manager use the classic rosrun command :

```
$ rosrun   database_manager    run_client
```

then the following database is created



including informations contained in XML files located at :

...toaster/database_manager/database

You may need to modify these files to be consistent with your stage.

id_list.xml ontology.xml static_property.xml

There is one fact_table and one memory_table for each agent.

Example :
if you add a new agent paul
then fact_table_paul and memory_table_paul are created

The first entity placed in id_list.xml will have the main table in the database.

You should notice that the ontology_table include an adjacency list model :

Example :

groupe	individual	intantiated
color	red	true
agent	human	true
tape	grey_tape	true
viewable	tape	false
tape	red_tape	true

There is a self join between first and second colomns.

This also can be seen as a tree :

```
|--color
|  |--red
|--agent
   |--human
|--viewable
|  |--tape
|  |  |--grey_tape
|  |  |--red_tape
```

so you need to take care of this features in your queries.

What to do with database manager ?

As soon as database_manager is started you have acces to all theses services :

Basic services

- add_entity: Add a new entity in id_table, create fact and memory tables if the entity is a robot or an human.

- add_fact: Add a fact in main fact table and create a new event, update fact if allready in table.
If the fact implements new entities, those are added in id_table without informations about their type.
Warning : Take care to do not add a fact dealing with a non declared agent.

- add_fact_to_agent: Add a fact in a specific fact table and create a new event, update fact if allready in table.
If the fact implements new entities, those are added in id_table without informations about their type.
Warning : take care to do not add a fact dealing with a non declared agent.

- remove_fact: Delete a fact from the main fact table and add it into the main memory table.

- remove_fact_to_agent: Delete a fact from a specific fact table and add it into the relativ memory table.

Get informations from main table

- get_facts: Send back all facts from main tables (current and passed).

- get_fact_value: Send back values of a specific fact from main tables.

- get_current_facts: Send back all facts from main fact table.

- get_passed_facts: Send back all facts from main memory table.

Get informations from other agents tables

- get_facts_from_agent: Send back all facts relativ to a specific agent (current and passed).

- get_fact_value_from_agent: Send back values of a specific fact from a targeted agent.

- get_current_facts_from_agent : Send back all facts from a specific fact table.

- get_passed_facts_from_agent : Send back all facts from a specific memory table.

Other usefull services

- get_properties: Send back all properties from property table.

- get_property_value: Send back values of a specific property from property table.

- get_agents : Send back all agents from id_table (humans and robots).

- get_all_id : Send back all elments from id_table.

- get_id_value : Send back all values of a specific element from id_table.

- add_event: Add a new event in event table.

- get_events: Send back all events from event table.

- get_event_value: Send back values of a specific event.

- get_ontologies: Send back all informations from ontology table.

- get_ontology_values: Send back values of a specific ontologic group.

- get_ontology_leaves: Send back leaves of an ontologic group.

- execute_SQL : Execute an SQL query, all results of this request are returned in a vector.

How to use toaster manager ?

To use services provided by database_manager with console :

Take a look at all actives ros services with the commande

```
$ rosservice list
```

then if you need more details about one of them type

```
$ rossrv show servicename
```

finaly to use a service do

```
$ rosservice call servicename
```

and complete request fields if needed.

To use services provided by database_manager in a ros c++ programm :

First declare a new ros node
ros::NodeHandle node;

then a new client
ros::ServiceClient client =
node.serviceClient<toaster_msgs::AddFact>("database/add_fact");

create your request
toaster_msgs::AddFact myRequest;
myRequest.request.fact.subjectId = "cube";
myRequest.request.fact.property = "isOn";
myRequest.request.fact.targetId = "table";

call the database serveur
client.call(myRequest);

get your results in
myRequest.reponse

How to perform other queries:

If you did not succeed to do what you want with previous services there is also a specific one allowing you to perform any query of your choice :

```
$ rosservice call /database/SQl_order
```

any SQL request is available except those including : right outer join, left outer join or full outer join.

Here is some usefull paterns you could use for your queries :

to select an element in table :

```
$  SELECT id , name FROM table WHERE type = ' 'object' ' AND  color = ' 'red' '
```

to get the union of table1 and table2 :

```
$ SELECT id FROM table1
$ UNION
$ SELECT id FROM table2
```

to get the intersection of table1 and table2 :

```
$SELECT id FROM table1
$INTERSECT
$SELECT id FROM table2
```

or

```
$ SELECT name FROM table1
$INNER JOIN table2
$ ON table1.id = table2.id
```

Moreover there is many tutorials dealing with SQLite on internet that can help you to perform more complex queries.