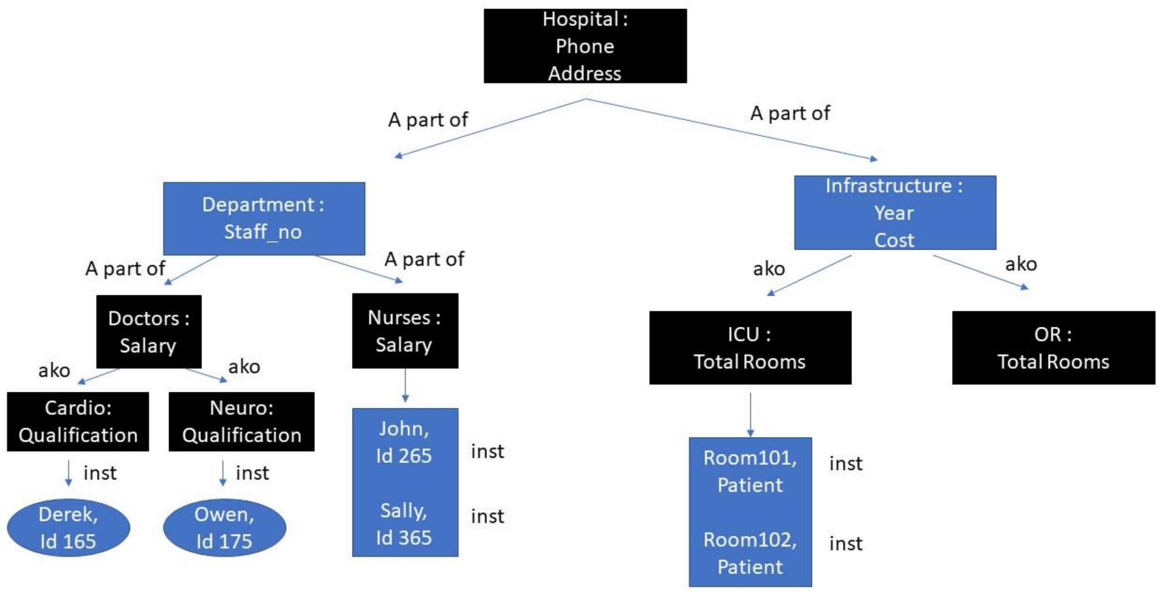
Documentation:



Hospital->
Phone: default-0120 3819100
Address: default-Delhi
Patients: default-500
Doctors: default-50

Department->
Staff_no: range(1,100)

Infrastructure->
Year: range(2010,2020)
Cost: range(50000,50000000)

Doctor->
Salary: default-20000000

Nurses->
Salary: default-20000

ICU->
Rooms:10

OR->
Rooms:5

Cardio->
Phone: default-0120 3819100

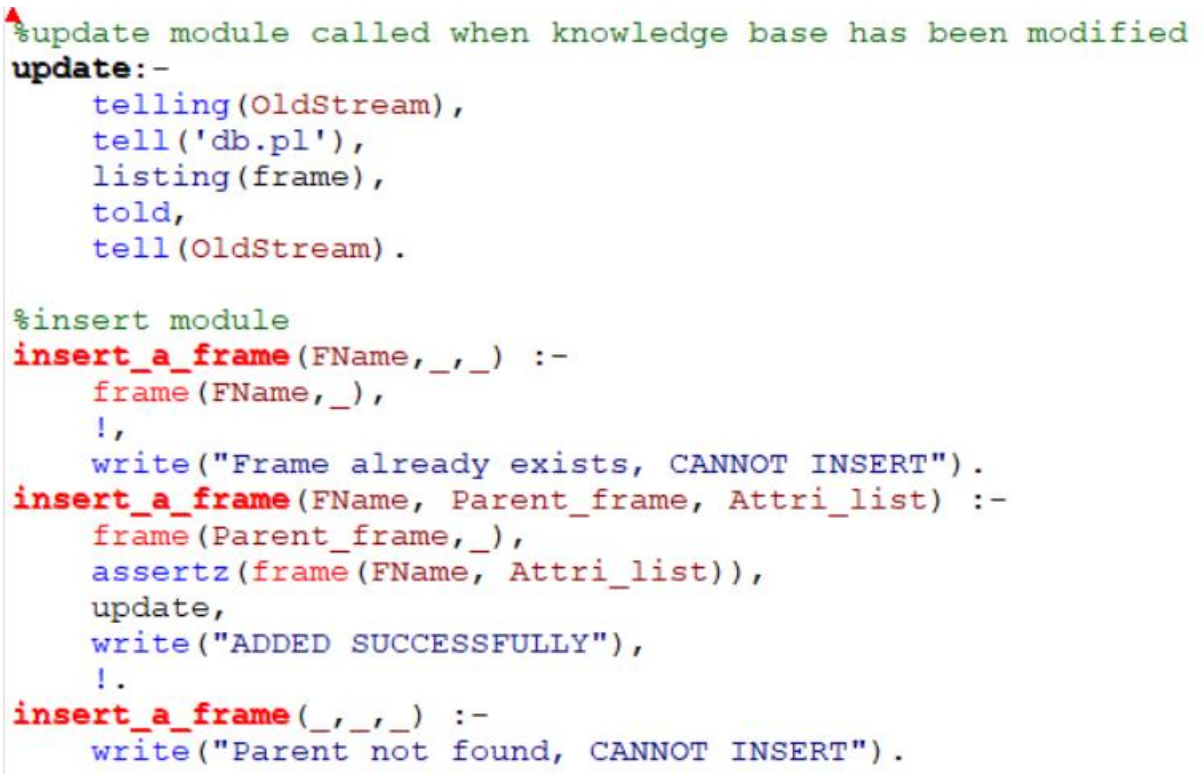
Neuro->
Phone: default-0120 3819100

Instances->
Name: respectively For Doctors and Nurses
Id: respectively

Instances->
Room: respectively For ICU and OR
Patient: respectively

Modules:

Insert



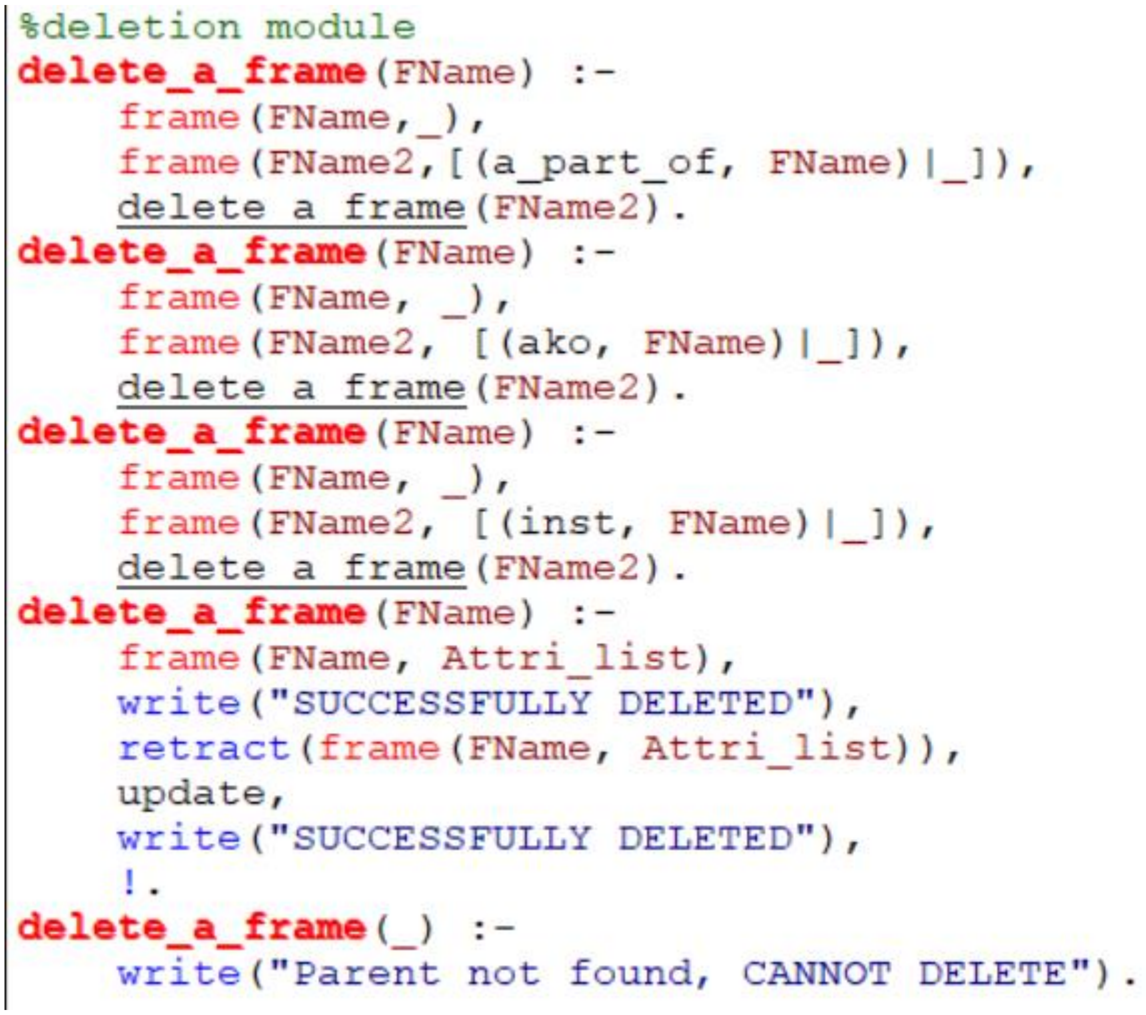
```
%update module called when knowledge base has been modified
update:-
    telling(OldStream),
    tell('db.pl'),
    listing(frame),
    told,
    tell(OldStream).

%insert module
insert_a_frame(FName,_,_) :-
    frame(FName,_),
    !,
    write("Frame already exists, CANNOT INSERT").
insert_a_frame(FName, Parent_frame, Attri_list) :-
    frame(Parent_frame,_),
    assertz(frame(FName, Attri_list)),
    update,
    write("ADDED SUCCESSFULLY"),
    !.
insert_a_frame(_,_,_) :-
    write("Parent not found, CANNOT INSERT").
```

update module saves the knowledge base file when it is modified for both **insertion** and **deletion** actions.

insert_a_frame takes frame name and the parent frame, the new frame is added under the parent frame and from the 3rd param Attribute list which is a list with first element describing the link between parent frame and new frame.
assertz adds the new frame at the end of frame predicate list
example: ?- insert_a_frame(admin,infrastructure,[(ako, infrastructure), (rooms, value, 1)]).

Delete



```
%deletion module
delete_a_frame(FName) :-
    frame(FName,_),
    frame(FName2,[(a_part_of, FName)|_]),
    delete_a_frame(FName2).
delete_a_frame(FName) :-
    frame(FName, _),
    frame(FName2, [(ako, FName)|_]),
    delete_a_frame(FName2).
delete_a_frame(FName) :-
    frame(FName, _),
    frame(FName2, [(inst, FName)|_]),
    delete_a_frame(FName2).
delete_a_frame(FName) :-
    frame(FName, Attri_list),
    write("SUCCESSFULLY DELETED"),
    retract(frame(FName, Attri_list)),
    update,
    write("SUCCESSFULLY DELETED"),
    !.
delete_a_frame(_) :-
    write("Parent not found, CANNOT DELETE").
```

delete_a_frame takes a frame name and deletes the frame and it loops till all the child frames are deleted, cycling through inst, ako and a_part_of links.
retract deletes the frame from the frame predicate list.
example: ?- ?- delete_a_frame(admin).

Query

```
search([(A, B)|_], A, B) :-
    !.
search([_|A], B, C) :-
    search(A, B, C).

%query module
find_a_frame(FName, Param, X) :-
    frame(FName, Y),
    search(Y, Param, X).
find_a_frame(FName, Param, X) :-
    frame(FName, [(ako, FName2)|_]),
    find_a_frame(FName2, Param, X).
find_a_frame(FName, Param, X) :-
    frame(FName, [(a_part_of, FName2)|_]),
    find_a_frame(FName2, Param, X).
find_a_frame(FName, Param, X) :-
    frame(FName, [(inst, FName2)|_]),
    find_a_frame(FName2, Param, X).
```

search loops from the entire attribute list of the frame.

find_a_frame cycling through inst, ako and a_part_of links till the query is satisfied. It takes the frame name, parameter and a variable which returns the value if it exists

example: ?- find_a_frame(john,a_part_of,X).