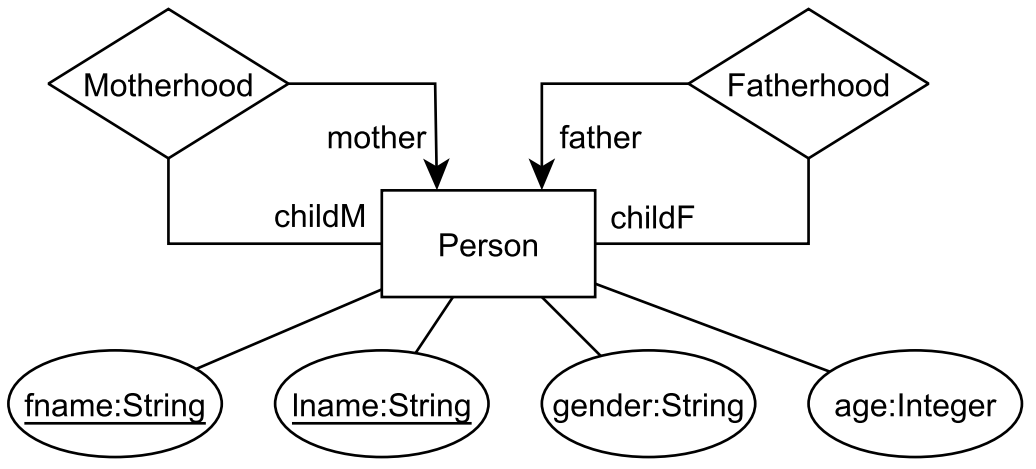
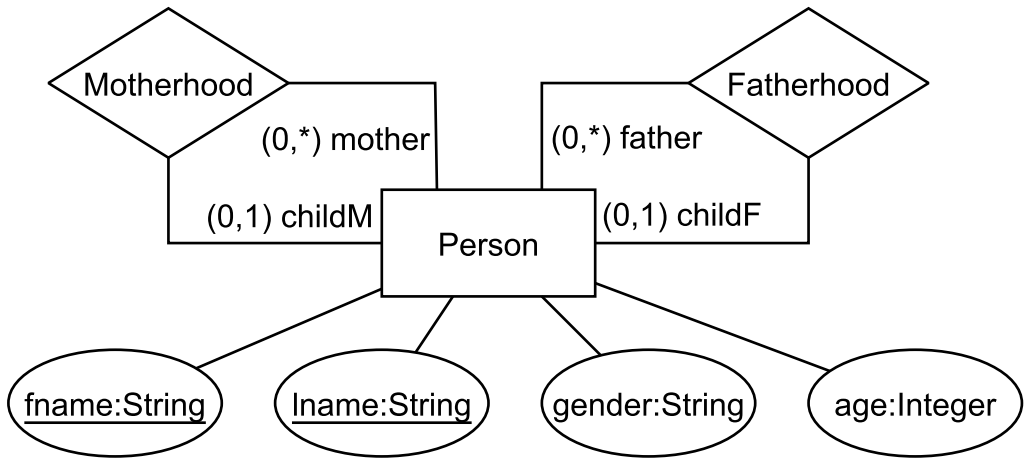
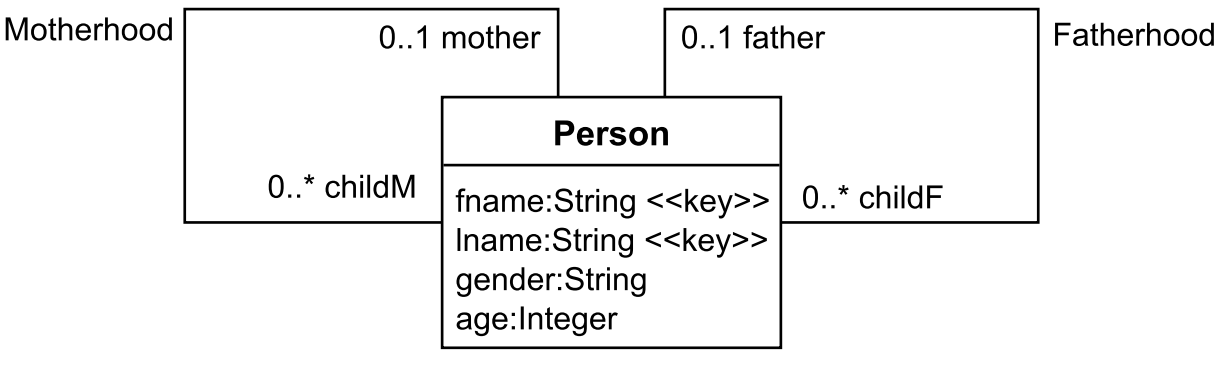
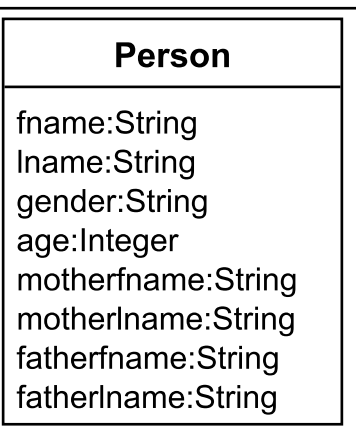







```
Person(fname:String, lname:String, gender:String, age:Integer,
  motherfname:String, motherlname:String,
  fatherfname:String, fatherlname:String)
Person.[motherfname,motherlname] -> Person.[fname,lname]
Person.[fatherfname,fatherlname] -> Person.[fname,lname]
```

```
CREATE TABLE Person (
  fname VARCHAR NOT NULL,
  lname VARCHAR NOT NULL,
  gender VARCHAR NOT NULL,
  age INTEGER NOT NULL,
  motherfname VARCHAR NULL,
  motherlname VARCHAR NULL,
  fatherfname VARCHAR NULL,
  fatherlname VARCHAR NULL,
  CONSTRAINT fname_lname_key
    PRIMARY KEY (fname,lname),
  CONSTRAINT mother_fname_lname_foreign_key
    FOREIGN KEY (motherfname,motherlname) REFERENCES Person(fname,lname),
  CONSTRAINT father_fname_lname_foreign_key
    FOREIGN KEY (fatherfname,fatherlname) REFERENCES Person(fname,lname)
);
```




```
context p1,p2:Person inv fname_lname_key:
  p1<>p2 implies (p1.fname<>p2.fname or p1.lname<>p2.lname)
context p:Person inv mother_fname_lname_foreign_key:
  p.motherfname<>null and p.motherlname<>null implies
    Person.allInstances->exists(m | p.motherfname=m.fname and p.motherlname=m.lname)
context p:Person inv father_fname_lname_foreign_key:
  p.fatherfname<>null and p.fatherlname<>null implies
    Person.allInstances->exists(f | p.fatherfname=f.fname and p.fatherlname=f.lname)
```

```
Person(fname:String, lname:String, gender:String, age:Integer)

Motherhood(childMfname:String, childMlname:String,
  motherfname:String, motherlname:String)
Motherhood.[childMfname,childMlname] -> Person.[fname,lname]
Motherhood.[motherfname,motherlname] -> Person.[fname,lname]

Fatherhood(childFfname:String, childFlname:String,
  fatherfname:String, fatherlname:String)
Fatherhood.[childFfname,childFlname] -> Person.[fname,lname]
Fatherhood.[fatherfname,fatherlname] -> Person.[fname,lname]
```