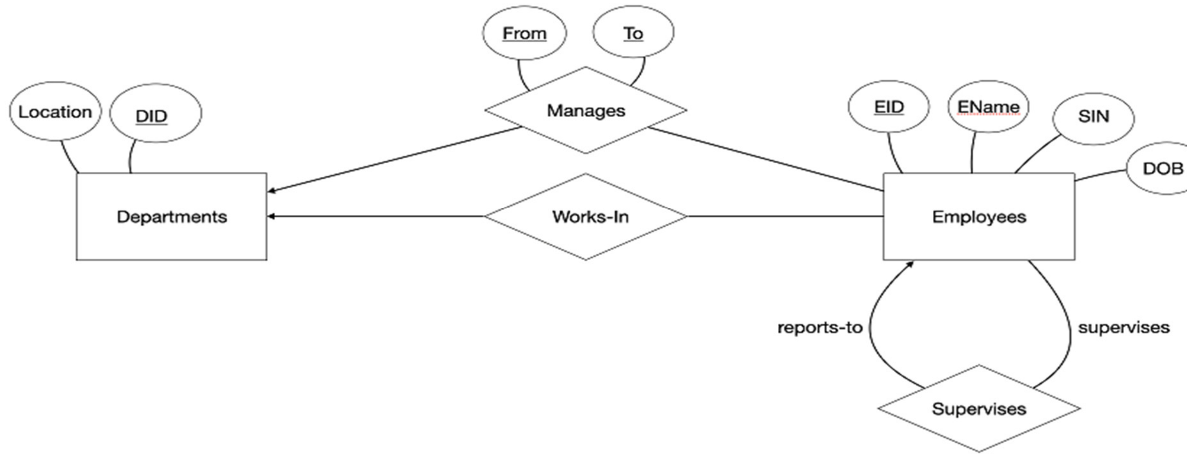
Solution to Assignment 2

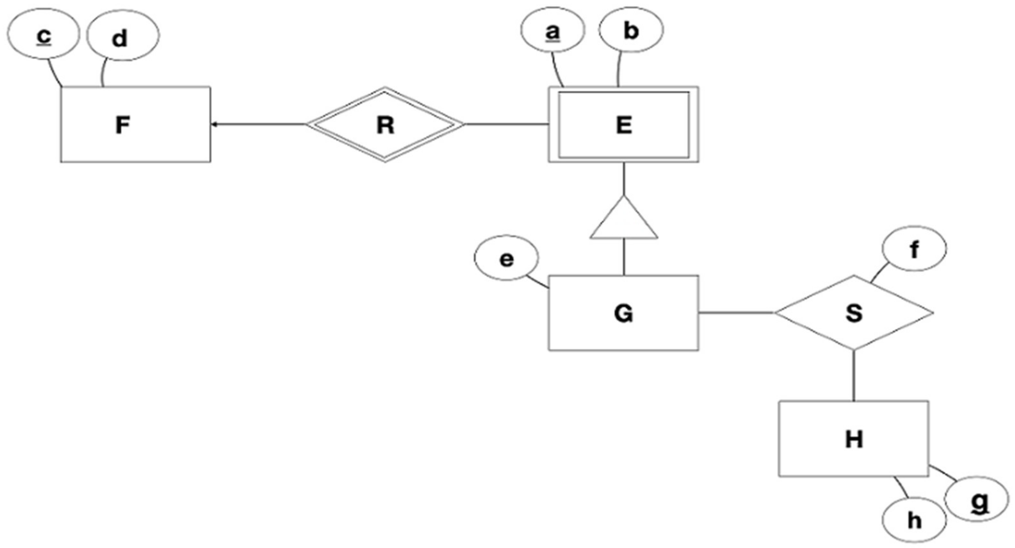
1.(a)



(b) Emps (<u>EID</u>, EName, SIN, DOB)
Depts (<u>DID</u>, Location)
Dept-Managers (<u>DID</u>, <u>EID</u>, <u>From-Date</u>, <u>To-Date</u>)
Work-In (<u>EID</u>, DID)
Supervisors (<u>EMP-EID</u>, SUP-EID)

(c) SNN is in EMP unique. Since the relationship Dept-Managers is m:1 from Emps to Depts, we may merge Depts and Dept-Managers into a single relation. We assumed the employees do not change departments. If this Is not true, then we can add the attributes From-Date and To-Date to record the start and end dates when an employee changes his/her department.

2.



(a) F(c, d), E(a, c, b), G(a, c, e), H(g, h), and S(a, c, g, f)

(b) F(c, d), RE(a, c, b), EG(a, c, b, e), and H(g, h).

Note that while concept E has the same schema in parts (a) and (b), the objects/tuples that belong to relation E in part (a) are divided between the classes E and G in part (b), with the additional information for attribute b for those objects that go to class G (in part b).

Also note the relationships R and S will be represented as part of and within the declaration of the classes created for the entity sets they are connected to. Check out for instance the ODL notation in the textbook used to define classes, where in addition to the usual names and types of the attributes used to define a class, there is also the additional property/feature called **relationship**, to be used to express binary relationships between classes. For example, in ODL, the relationship S is expressed using the relationship property in the class declarations for G and H.

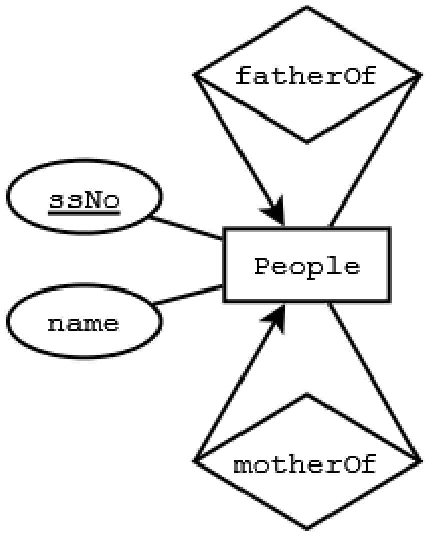
Note: This solution for part b gives CLASSES we need if we want to implement the DB application using an OO DBMS, for which ODL is an example language.
On the other hand, if we want to create relations BUT follow an OO approach to covert the entity sets involved in the hierarchy into RELATIONS, then we create a table for any other entity or relationship set we have outside the hierarchy. In this case, the resulting RELATIONAL DB schema we get in part (b) following an OO approach is:

(b) F(c, d), RE(a, c, b), EG(a, c, b, e), and H(g, h), S(a, c, f, g).

(c) F(c, d), E(a, c, b), GH(a, c, b, e), S(a, c, g, f), H(g, h)

Note: relation GH is obtained by merging the relations E and G involved the isa hierarchy.

3.(a)



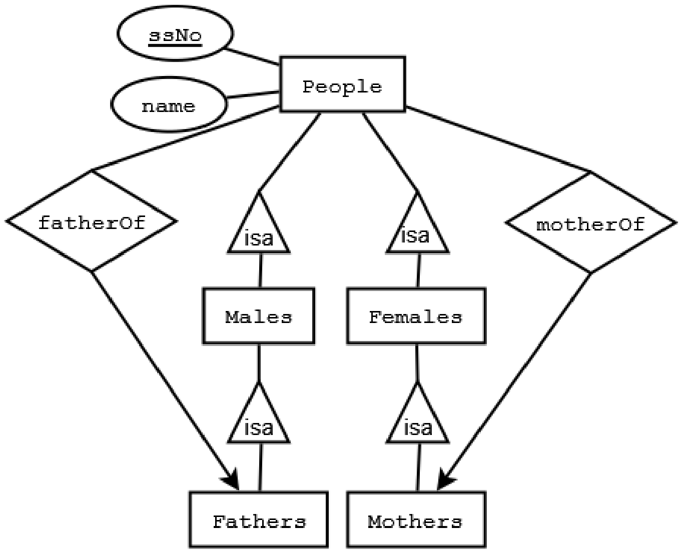
(Figure: by the textbook authors)

(b) If we convert this E/R into a relational database, using the E/R-style, we obtain the following relation schemas -- the key attributes are underlined:

People (<u>ssNo</u>, Name)
MotherOf (<u>Child-ssNo</u>, Moher-ssNo)
FatherOf (<u>Child-ssNO</u>, Father-ssNo)

Just as an aside note (not needed for the full mark): Since MotherOf and FatherOf relationships are both m:1, we may merge them to get: ParentsOf (<u>Child-ssNo</u>, Mother-ssNo, Father-ssNo). In this case, the resulting relational database schema will include ParentsOf and People.

4.



(Figure: by the authors)

5.

(a) |R| ≤ e2*e3 (that is, R could at most have e2*e3 tuples)

(b) There could be two different answers depending on the meanings/semantics of the symbols used. The first answer is |R| ≤ min{e1, e2}* e3 (that is, for every entity set with a sharp arrow, the number of tuples in R can't be more than different combinations of tuples in all the other entity sets involved in R. This means, |R| can't be more thn e1*e3, and can't be more that e2*e3, and hence the number of tuples in R is not more than the minimum of the values in {e1*e3, e2*e3}.

Another possible answer would be |R| ≤ e3, which is a stricter interpretation. This makes sense, for example, when E3 represents the notion of children, E1 represents the notion mother, and E2 represents the father. Then, a triplet in R represents for every child, his/her mother and father).

(c) In this scenario, there are two possible answers. The first answer follows the same view as in the first answer in case b. As such |R| ≤ min{e1*e2, e2*e3, e1*e3}. The second possible answer would be |R| ≤ min{e1, e2, e3}.

(d). In this case, following the first interpretation, |R| = e1*e3, or following the second view, we have |R|= e3. An example application which justifies the second view is when E3 represent guests in a hotel, E2 represents the rooms (or suits) in the hotel, and E1 represents the parking lots for guests in the hotel. In this case, R represents, for every guest, the room-parking allocated, where suit is a must but parking could be null. In this case, the maximum number of tuples in R is e3.

Note that in any n-ary relationship R, when there is at most one sharp arrow and the rest are all simple connections, different interpretations / views of application scenarios will all yield the same, unique answer.