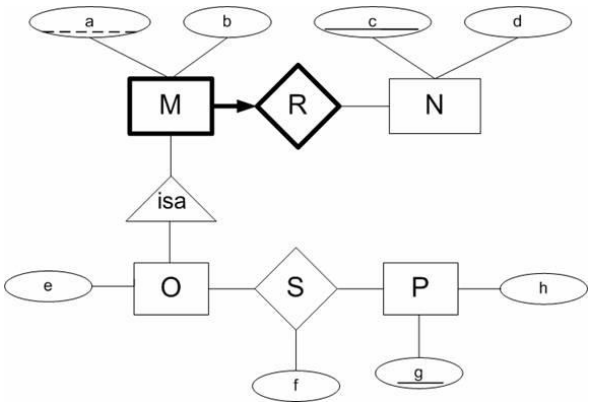
Practice Exercises on ER to SQL

Question 1



a) Translate the above E-R diagram to relations using the first method on page 83 – i.e., create a relation for each class and sub-class in an ISA hierarchy.

b) Translate the answer from part (a) to SQL.

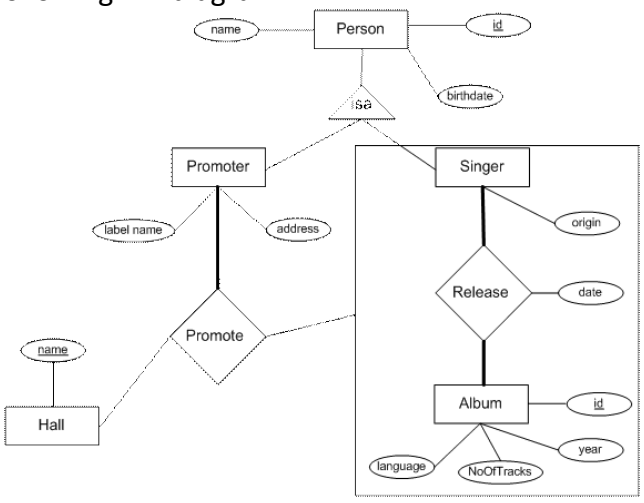
c) Would you consider using the second method on page 83 (i.e., create a relation only for the sub-classes in the ISA hierarchy) to translate the ISA hierarchy in this diagram? Why or why not? Note: the reasoning is what matters, not that you get the "right" answer.

Question 2

Consider the following case in music industry:

- An album is released by one or more singers.
- Singers promote their new albums with a dinner party in a large hall.

It's represented in the following E-R diagram:



a) Translate the diagram above to relations by writing the database schema. To represent the ISA hierarchy, use the second method on page 83, i.e., create one relation for each subclass but not the superclass.

Practice Exercises on ER to SQL

Question 1 Answer

a)

- N is an entity, so we'd create a table for it: N(c,d)
- P is an entity, so we'd create a table for it: P(h,g)
- Since M is a weak entity, we'd create one table for it and R, which contains the key of N as a key: M_R(a,b,c), where c is a foreign key of N. Because R is a weak entity, we must delete a M_R tuple if the corresponding N tuples disappears
- Now we create a relation for O, which must include the key of M. The key of M includes the key of N since it is a weak entity, resulting in: O(e,a,c), where a and c are a foreign key of M_R. Note that technically speaking c is really a foreign key of N, but since the requirements are that you must refer to the entire key of a table, we must have it refer to M_R's key, rather than N's.
- S is a many to many relationship, so we'd create a table for it which includes the attributes of S and the keys of O and P, which together form the primary key of S: S(f,a,c,g), where a and c are foreign key references to O, and g is a foreign key reference to P.

b)

```
CREATE TABLE N  (
        c       integer,
        d       integer,
        PRIMARY KEY (c)
)

CREATE TABLE P   (
        h       integer,
        g       integer,
        PRIMARY KEY (g)
)

CREATE TABLE M_R (
        a       integer,
        b       integer,
        c       integer,
        PRIMARY KEY (a,c),
        FOREIGN KEY (c) REFERENCES N ON DELETE CASCADE
)
```

```
CREATE TABLE O (
        e       integer,
        a       integer,
        c       integer,
        PRIMARY KEY(a,c),
        FOREIGN KEY (a,c) REFERENCES M_R
)

CREATE TABLE S (
        f       integer,
        a       integer,
        c       integer,
        g       integer,
        PRIMARY KEY(a,c,g),
        FOREIGN KEY (a,c) REFERENCES O),
        FOREIGN KEY (g) REFERENCES P
)
```

c) In general, this wouldn't be a good idea since there are relationships attached to the superclass, but since there is only one subclass, in this case, it would work. However, you'd only want to do this if all M's are O's, which does make you wonder why you'd make the distinction in the first place.

Question 2 Answer

- Because we're using the second method of translating an ISA hierarchy, there is no separate table for Person. Thus we require tables for:
    - Promoter(name,id,birthdate,label_name, address)
    - Singer(name,id,birthdate,origin)
- We create a table for Hall: Hall(name)
- We create a separate table for Album: Album(id,language,NoOfTracks,year)
- We create a table for the Release relationship, which includes the keys of the concepts that it refers to (Singer, and Album) as foreign key references. Note that we have to change the attribute names of ID since we can't have two attributes named id: Release(singer_id,date,album_id). Note that we can't express the constraint that every singer must have an album and that every album must be sung by someone with what we know so far.
- Now we create a table for Promote, which has foreign keys to the keys of the concepts it relates: Promoter, Hall, and Release. Renaming the ID of the promoter and the name of the Hall is a really good idea, but not actually required by the constraints since we've already renamed the IDs of Singer and Album. This results in: Promote(promoter_id, hall_name,singer_id,album_id). Again, you must have the foreign keys to Promoter, Hall, and Release. Also, you cannot enforce the total participation constraint on Promoter in the Promotes relationship by adding a "not null" constraint on Promoter. What this would do would require that each Promotes relationship would have to have

a promoter, not that each promoter has to promote something. See the bottom of page 80.

```
CREATE TABLE Promoter (
        name            CHAR(20),
        id              INTEGER,
        birthdate       DATE,
        label_name      CHAR(20),
        address                 CHAR(100),
        PRIMARY KEY (id)
)

 CREATE TABLE Singer  (
        name            CHAR(20),
        id              INTEGER,
        birthdate       DATE,
        origin          CHAR(20),
        PRIMARY KEY (id)
)

CREATE TABLE Hall (
        name            CHAR(20),
        PRIMARY KEY (name)
)

CREATE TABLE Album (
        id              INTEGER,
        Language        CHAR(20),
        NoOfTracks      INTEGER,
         Year           INTEGER,
        PRIMARY KEY (id)
)

CREATE TABLE Release (
        singer_id       INTEGER,
        Album_id        INTEGER,
        Date            DATE,
        PRIMARY KEY(singer_id,album_id),
        FOREIGN KEY (singer_id) REFERENCES Singer,
        FOREIGN KEY (album_id) REFERENCES Album
)
```

Practice Exercises on ER to SQL

```
CREATE TABLE Promote (
        singer_id       INTEGER,
        Album_id        INTEGER,
        Promoter_id     INTEGER,
        Hall_name       Char(20),
        PRIMARY KEY(singer_id,album_id,promoter_id, hall_name),
        FOREIGN KEY (singer_id,album_id) REFERENCES Release,
        FOREIGN KEY (hall_name) REFERENCES Hall,
        FOREIGN KEY (promoter_id) REFERENCES Promoter
)
```