CSC 370

Activity Worksheet: Weak Entities & Sub-classes

Sean Chester

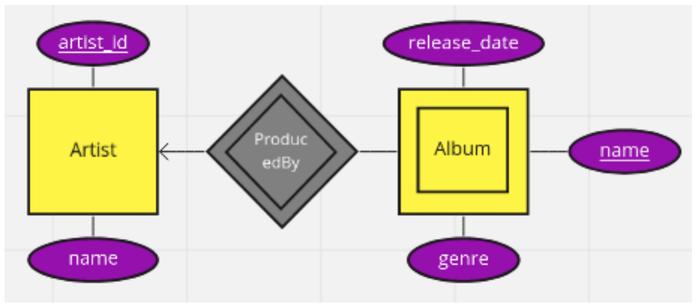
Fall 2022

Notes

This worksheet provides a series of practice questions for modelling complex entity sets. In each question, you are given an ERD and asked to list all primary and foreign keys. If you can do that successfully, then you could certainly write the CREATE TABLE statements to build the database. The first question has been answered already as a model solution.

Questions

1. You are given the entity-relationship diagram (ERD) below. List all primary and foreign keys that arise from the diagram.



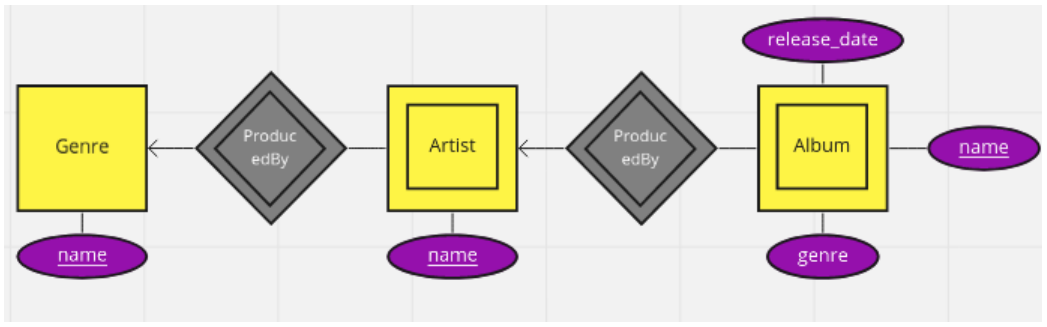
id artist_id

unique {artist_id, name}

Album is released by a single artist, but artist can have multiple albums.

Album has foreign Key artist_id.

2. You are given the (admittedly strange but technically valid) entity-relationship diagram (ERD) below, which supposes that Artist names are unique when partially defined by the genre. List all primary and foreign keys that arise from the diagram.



Name

Name

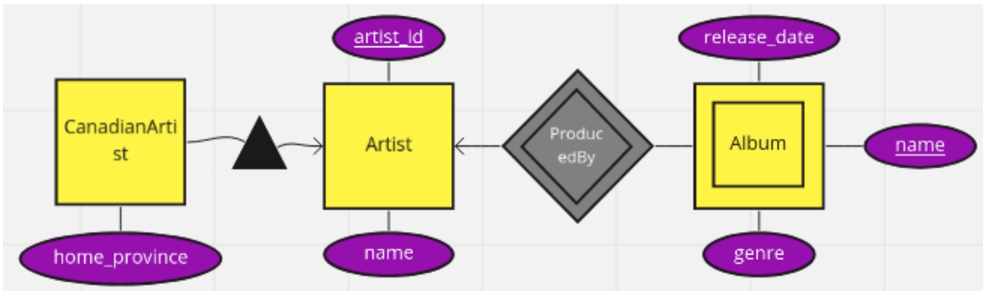
Genre_Name

name

Genre_Name

Artist_Name

3. You are given the entity-relationship diagram (ERD) below that differentiates Canadian artists from non-Canadian artists. List all primary and foreign keys that arise from the diagram.



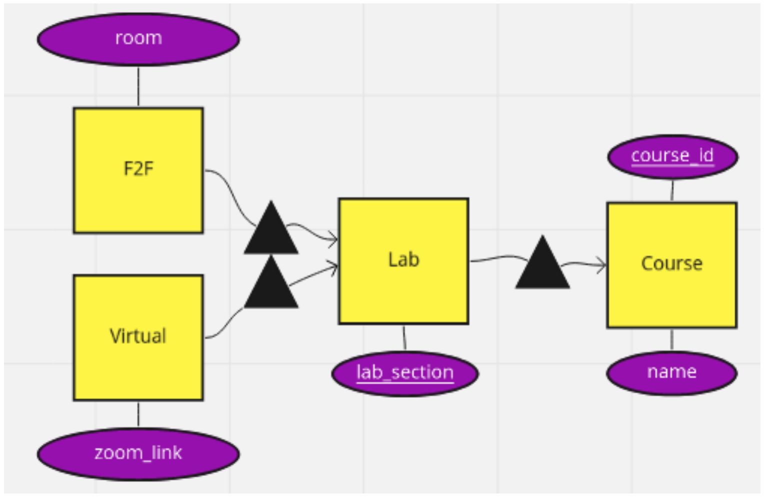
artist_id

artist_id

name

artist_id

4. You are given the entity-relationship diagram (ERD) below. List all primary and foreign keys that arise from the diagram.



lab_section

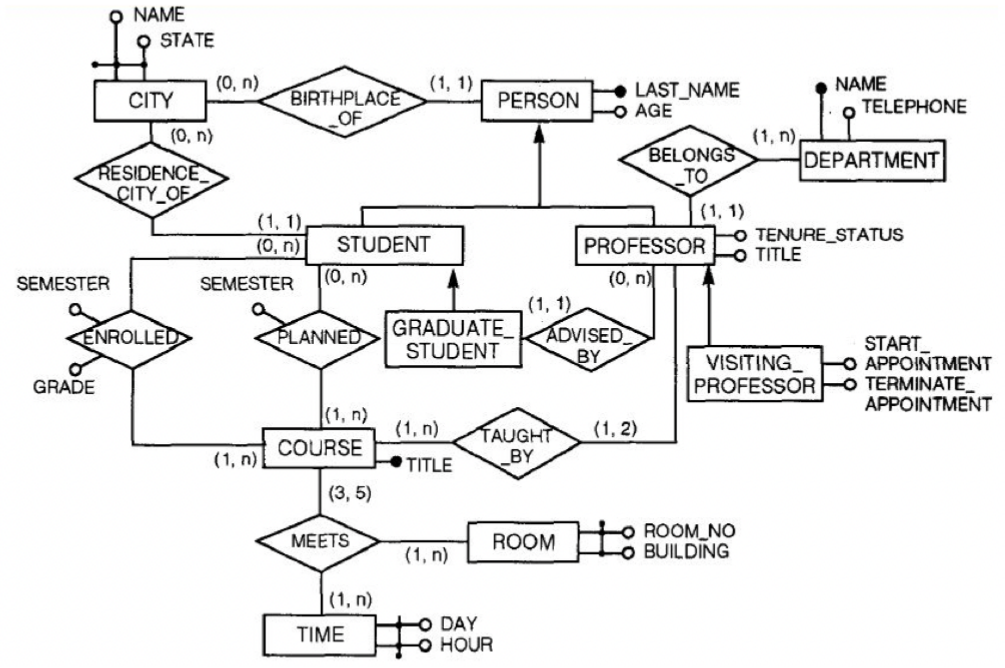
lab_section

course_id

course_id

+ course_id

5. How do we interpret the schema given by the ERD? Where do we even start?



Solutions

Question 1

Primary Keys {artist_id} is unique in Artist. Two artists could have the same name, but not the same id {name, artist_id} is unique in Album. Because Album is a weak entity set, it inherits part of the key from its supporting relation ProducedBy is a one-many relationship; so, it does not require a table nor a primary key.

Foreign Keys Within Album, there would be a foreign key {artist_id} to Artist to indicate the tuple that supports its definition.

Question 2

This is unnaturally complex, but helps to develop a systematic approach. If we can identify dependencies in the definitions, then we can build this out in the reverse order. Here we see that the definition of Album depends on Artist which depends on Genre. So let us define Genre first and go in reverse from there.

Primary Keys {name} is unique in Genre. There is a straight-forward strong entity set. {artist_name, genre_name} is unique in Artist. Because it is a weak entity set, it borrows part of its primary key through its supporting relationship {album_name, artist_name, genre_name} is unique in Album. Because it is a weak entity set, it borrows part of its primary key through its supporting relationship, and the primary key of the supporting relation is {artist_name, genre_name}.

For the foreign keys, it is again probably easiest to reason through in the same direction as the primary keys.

Foreign Keys Genre does not have any many-one relationships, one one-many relationships. Thus, it does not have any foreign keys. Artist has a many-one relationship to Genre, so it has a foreign key on {genre_name} to the primary key of Genre. Album has a many-one relationship to Artist, so it has a foreign key on {artist_name, genre_name} to the primary key of Album.

Question 3

As in the previous question, let's start with the strong entity set(s).

For Artist, we have a primary key on {artist_id}.

Considering next the weak entity sets (including subclasses), we have:

{artist_id} is a primary key for CanadianArtist as well, since it does not contribute any new unique attributes {artist_id, name} is a primary key for Album (as in the previous question), since it contributes one unique attribute itself.

Regarding the foreign keys, the relationships are not many-many and have no unique attributes; so, they won't materialise as tables. The Artist table is a strong entity set with no many-one relationships; so, it also will not have any foreign keys.

Both CanadianArtist and Album with have a foreign key from {artist_id} to the Artist table.

Question 4

Again, we shall start from the strong entity sets and work backwards through the dependencies. We can see that all relationships are IsA relationships, so will not generate tables.

For Course, the primary key is {course_id}.

For Lab, it inherits the primary key of its parent class but also contributes a unique attribute of its own. Thus, the primary key is {course_id, lab_section}.

For both F2F and Virtual, there are no additional unique attributes, so they will inherit the primary key of their parent class, {course_id, lab_section}.

Regarding foreign keys:

For Course, there are no many-one relationships; so, it does not have any foreign keys.

For Lab, there is a foreign key through the IsA relationship, namely on {course_id} to its parent class, Course.

Finally, for both F2F and Virtual, they have inherited their primary key so those are also foreign keys, i.e., they have foreign keys on {course_id, lab_section} to Lab.