CSC 370

Activity Worksheet: Reading Keys from ERD's

Dr. Sean Chester

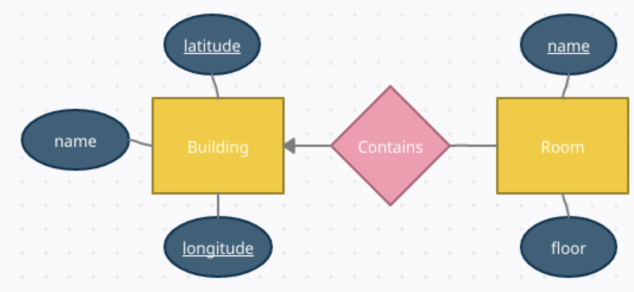
Fall 2022

Notes

This worksheet provides a series of practice questions for converting ERD's into SQL DDL queries. 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.

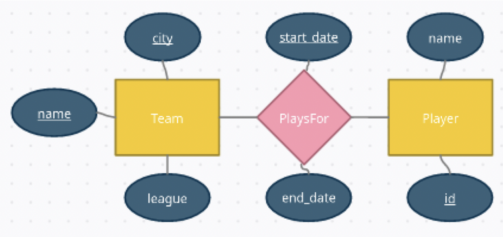


Solution:

Primary Keys
{latitude, longitude} is unique in Building. Two buildings could have the same longitude or latitude, but not both simultaneously.
{name} is unique in Room. Two rooms can have the same floor, but they cannot have the same name.
Contains is a one-many relationship; so, it does not require a table nor a primary key.

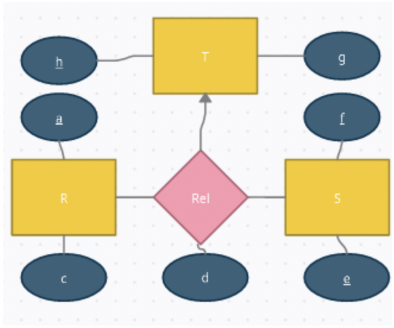
Foreign Keys
Within Room, there would be a foreign key {latitude, longitude} to Building.

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



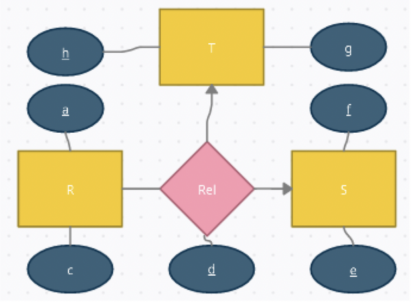
Solution:

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



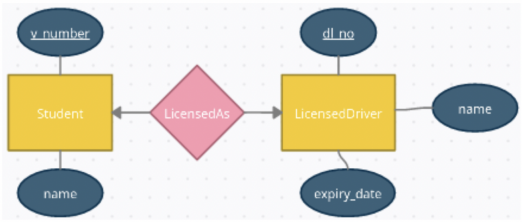
Solution:

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



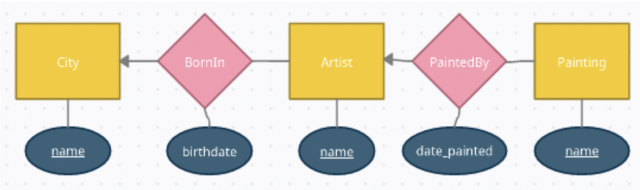
Solution:

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



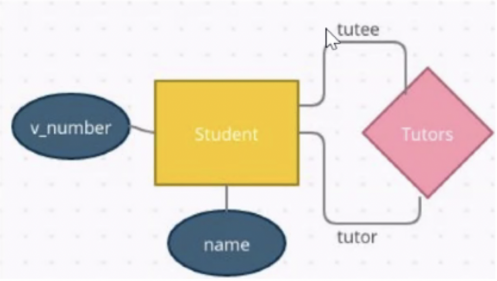
Solution:

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



Solution:

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



Solution:

Solutions

Question 1

Primary Keys
{latitude, longitude} is unique in Building. Two buildings could have the same longitude or latitude, but not both simultaneously.
{name} is unique in Room. Two rooms can have the same floor, but they cannot have the same name. Contains is a one-many relationship; so, it does not require a table nor a primary key.

Foreign Keys
Within Room, there would be a foreign key {latitude, longitude} to Building.

Question 2

Primary Keys
{city, name} is unique in Team. Two teams could have the same name or be from the same city, but not both simultaneously.
{id} is unique in Player. Two players can have the same names, but they cannot have the same id.
{player_id, team_name, team_city, start_date} is unique in PlaysFor. A player could play for the same team more than once as long as the start date differs. Because PlaysFor is a many-many relationship, it appears as its own table.

Foreign Keys
Within PlaysFor, there would be two foreign keys:
- (team_city, team_name) to (city, name) in Team
- (player_id) to (id) in Player

Question 3

Primary Keys
{a} is unique in R.
{e, f} is unique in S.
{h} is unique in T.
{a, e, f} is unique in Rel. Note that although {a, e, f, h} is also unique, it is a superkey because we can see that h is known for fixed {a, e, f}.

Foreign Keys
Within Rel, there would be three foreign keys:
- (a) to (a) in R
- (e, f) to (e, f) in S
- (h) to (h) in T

Question 4

Primary Keys
{a} is unique in R.
{e, f} is unique in S.
{h} is unique in T.
{a, d} is unique in Rel. We do not need to include the keys from S nor T because that would yield a superkey. However, R.a can appear multiple times, so long as each occurrence is paired with a different value for d. The values of (S.e, S.f) and (T.h) would be determined from the pair (R.a,d). We need to implement Rel as a separate table here, even though S and T are determined by the relationship, because d is part of the key. If we tried to store this relation inside R, we would need to duplicate tuples of R for each value of d with which they are associated.

Foreign Keys
Within Rel, there would be three foreign keys:
- (a) to (a) in R
- (e, f) to (e, f) in S
- (h) to (h) in T

Question 5

Primary Keys
{v_number} is unique in Student.
{dl_no} is unique in LicensedDriver.

It is not necessary to create a table for LicensedAs, because it is a one-one relationship

Foreign Keys
Here, there is some choice, because you could store the relationship in either entity set. I would use the verb in the relationship name to guide the choice, which in this case seems to imply a view as a student pointing to a licensed driver. If you can change the label of the relationship in the ERD, then I would favour storing the relationship in the smaller relation (if there is one) to reduce the number of NULL values. So, I would choose:

Within Student, there would be a foreign key to (LicensedDriver.dl_no)

Note that if you put the foreign key in both tables, e.g., to LicensedDriver.dl_no and Student.v_number, you may need quite complicated insertions if you want to add a new person to the database, as you'll need to add the tuple to both tables at the same time. We will discuss atomicity issues like this when we get to the concept of transactions. It probably isn't a solution that you want, even if it is a truer implementation of the conceptual design.

Question 6

Primary Keys
{name} is unique in Painting.
{name} is unique in Artist.
{name} is unique in City.

Neither of the relationships are implemented as tables, because they are both many- one relationships

Foreign Keys
Within Painting, there would be a foreign key to Artist.name.
Within Artist, there would be a foreign key to City.name

Question 7

Here, we have a many-many relationship involving only one entity set. It is customary to label attributes using the roles.

For the Student(v_number, name) relation, which is a strong entity set, we have the single-attribute key, v_number. There are no foreign keys as it is not involved in any many-one relationships.

The many-many relationship will be implemented as a relation Tutors(tutor, tutee). As a many-many relationship with no attributes, the primary key is clearly the union of the primary keys of the participating entity sets, i.e., tutor, tutee. Moreover, this relationship has two foreign keys, one for each role: tutor is a foreign key to v_number in Student, as is tutee.