Kramer Johnson

CPSC 5021 02

Homework 2

Chapter 2 Review Questions:

1. The importance of data models lies in their use as a communication tool. They can be used to convey the interaction and relationships of entities within a company to managers, employees, other database designers, or end users. For example, a database that shows a one to many relationships between invoices and vendors conveys the business rule that each vendor can issue many invoices, but each invoice can be for one specific vendor.

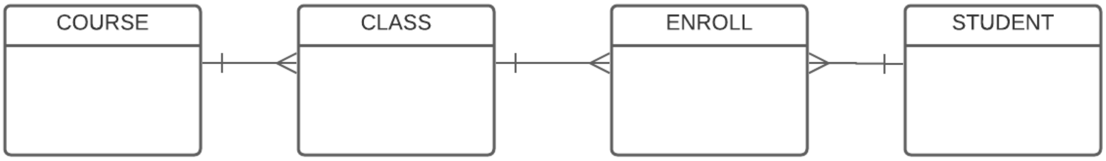
3. Business rules describe the relationships and constraints on the entities in a database. They be used to model these relationships. For example, there may be a business rule that a university's class may enroll a maximum of thirty students. This rule could be used place a cardinality on the many-to-many relationship between students and classes.

5. The entity relationship model helped to produce a more structured relational DB environment by providing designers with a graphical representation of the databases they were creating. Having these models to describe the database's entities, attributes, and relations at a conceptual level helped to provide a clearer picture of how the entire database worked. For example, a relationship between two entities may be implemented by having foreign keys in the tables. To understand this relationship, an end-user or designer would need to look at the two tables, see the foreign keys, and try to interpret what this relationship meant. An ERM can display the entities as boxes, create lines between them with appropriate line ends, and provide some text describing the relationship. This lets the viewer understand the relationship at a single glance.

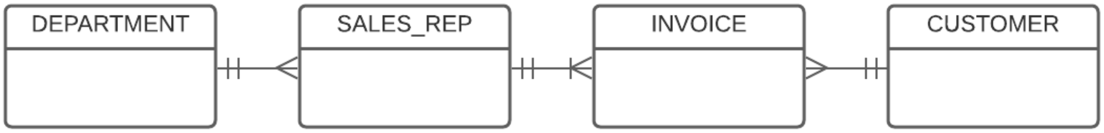
11. A relationship is a description of how two entities may be related to each other in a relational database. The three types of relationships are one-to-many, many-to-many, and one-to-one. An example of a one-to-many relationship is an invoice and a vendor; an invoice will list a single vendor, but each vendor could issue many invoices. A many-to-many relationship is a student to class relationship; each student can take many classes, and each class can have many students. A one-to-one relationship is a front door and a house; each house has one front door, and each front door is for one house.

Problems:

7.



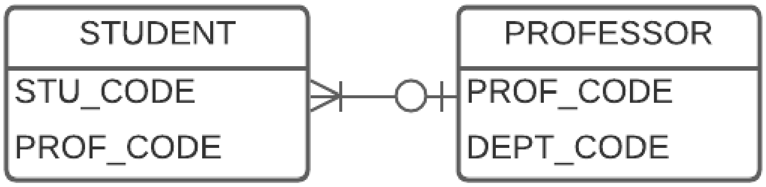
14.



Chapter 3 Review Questions:

2. An entity that displays entity integrity will have a unique, non-null, primary key that will identify each row in the table. An entity that displays referential integrity will have a foreign key that is associated with either a unique primary key or a null value in the table it is associated with. A table that has both will make it easy to look up any value because they are all uniquely keyed, and will also maintain a proper relationship with another table by having each of its rows associated with the correct row in another table.

11.



Problems:

1. EMPLOYEE: Primary key is EMP_CODE

Foreign key is STORE_CODE

STORE: Primary key is STORE_CODE

Foreign keys are REGION_CODE and EMP_CODE

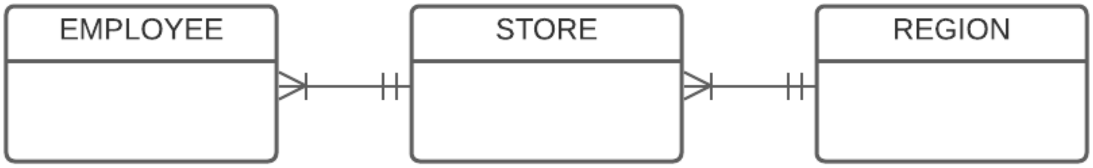
REGION: Primary key is REGION_CODE

Foreign key is None

2. Yes, the tables to exhibit entity integrity. Each table has a non-null, unique, primary key that can identify any specific row in the table.

3. The EMPLOYEE and STORE tables both exhibit referential integrity. They both contain foreign keys that have matching entries in the corresponding tables. The REGION table, on the other hand, does not have a foreign key so is not applicable to referential integrity.

8.

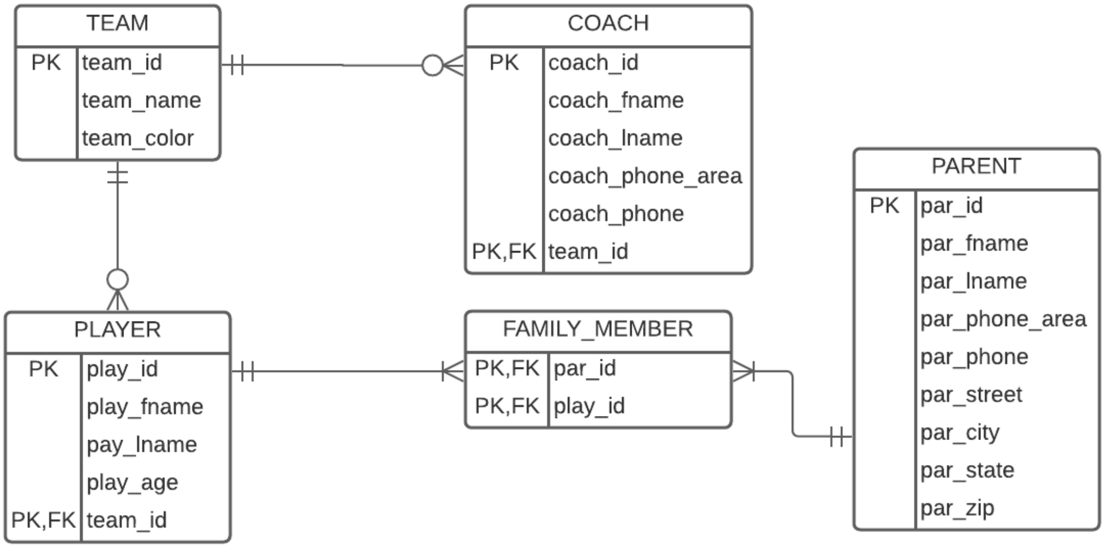


Chapter 4 Review Questions:

1. For an entity to be classified as a weak entity its existence must be dependent on another related entity and its primary key must be partially or fully derived from the parent entity. An example of this is a SONG and an ARTIST. A song cannot be created without an artist having created it so the song is existence-dependent on artist, when implemented SONG would need to inherit a primary key from ARTIST.

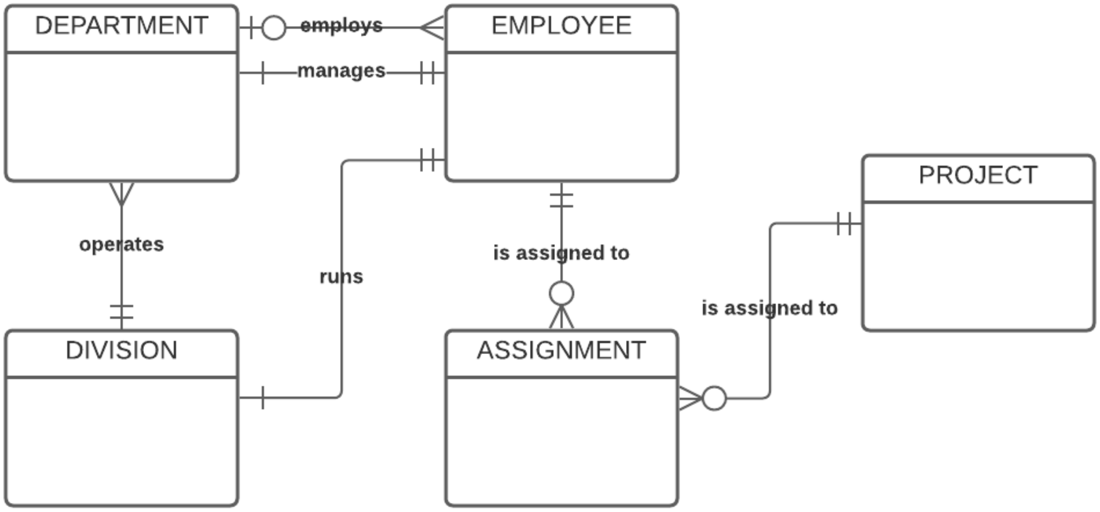
3. A relationship such as “an employee may have many degrees” can cause some interesting issues in a relational model. This is because it describes a multivalued attribute. That is, an attribute that can have multiple values for a single entity instance. One employee may have five degrees while another may have a single degree. The best way to handle a situation such as this, is to create another entity that associates employees with their degrees. The primary keys would be an employee’s name and a specific degree. Each degree would be listed in a new row. This way a single employee can have unlimited degrees that are uniquely identified.

21.

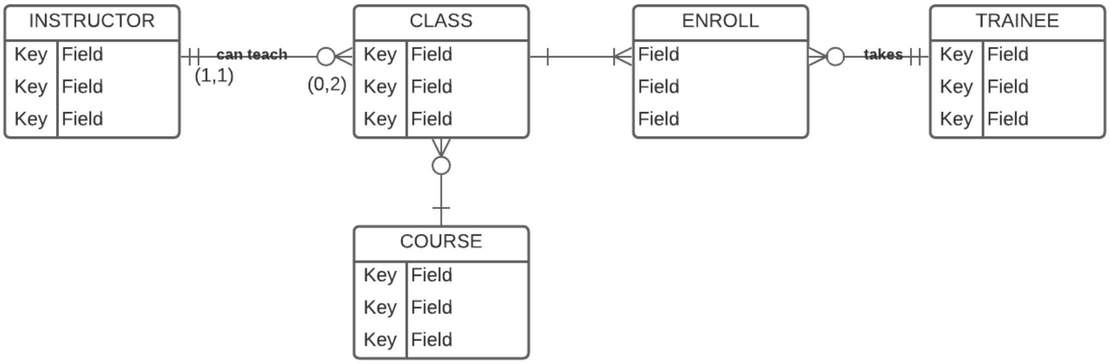


Problems:

1.



5.



a. The entities are INSTRUCTOR, CLASS, TRAINEE, and COURSE. ENROLL is used as an associative entity between CLASS and TRAINEE.

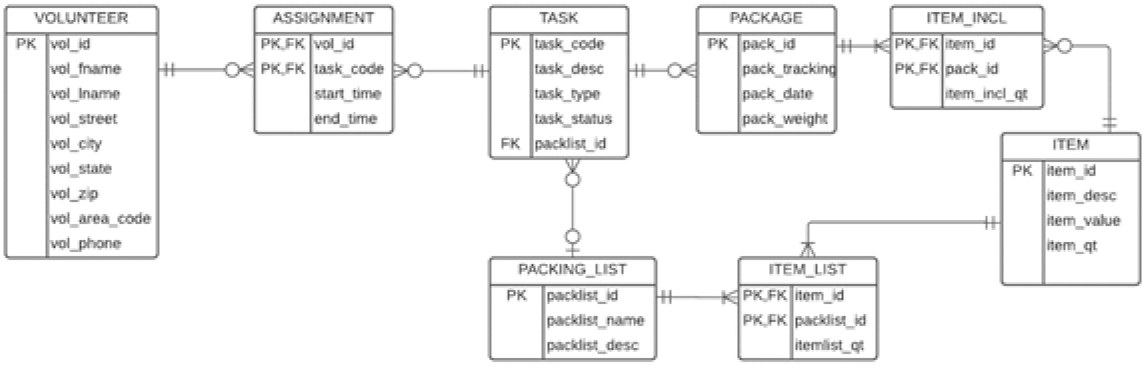
A TRAINEE can take zero to two classes, a CLASS can have ten to thirty students.

A COURSE can generate 0 to several classes, a CLASS is associated with one COURSE

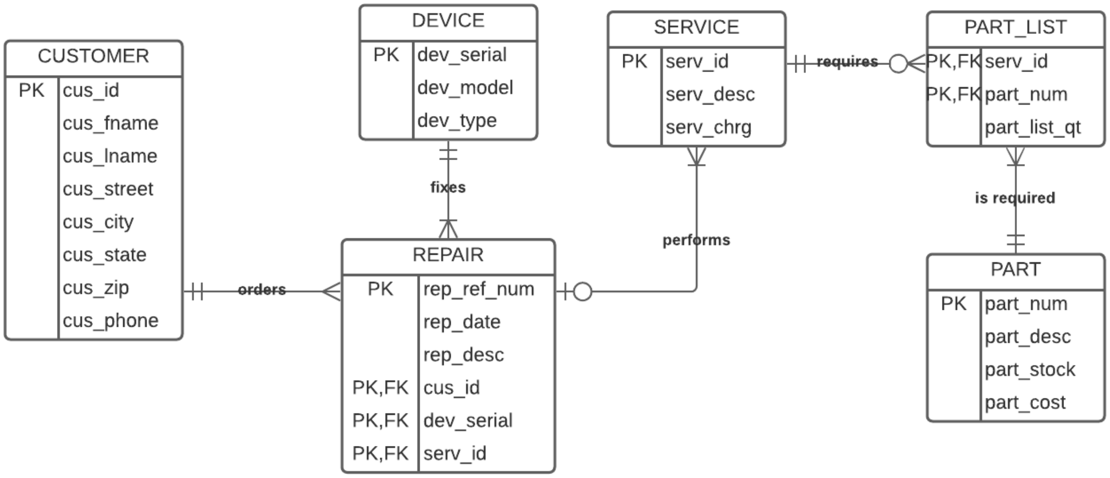
A CLASS is taught by one instructor, and an INSTRUCTOR can teach zero to two classes.

b. INSTRUCTOR and CLASS are in a 1:M relationship, the one side has a cardinality of (1,1) and the many side has a cardinality of (0,2), INSTRUCTOR is existence-independent of CLASS but class is existence-dependent on INSTRUCTOR.

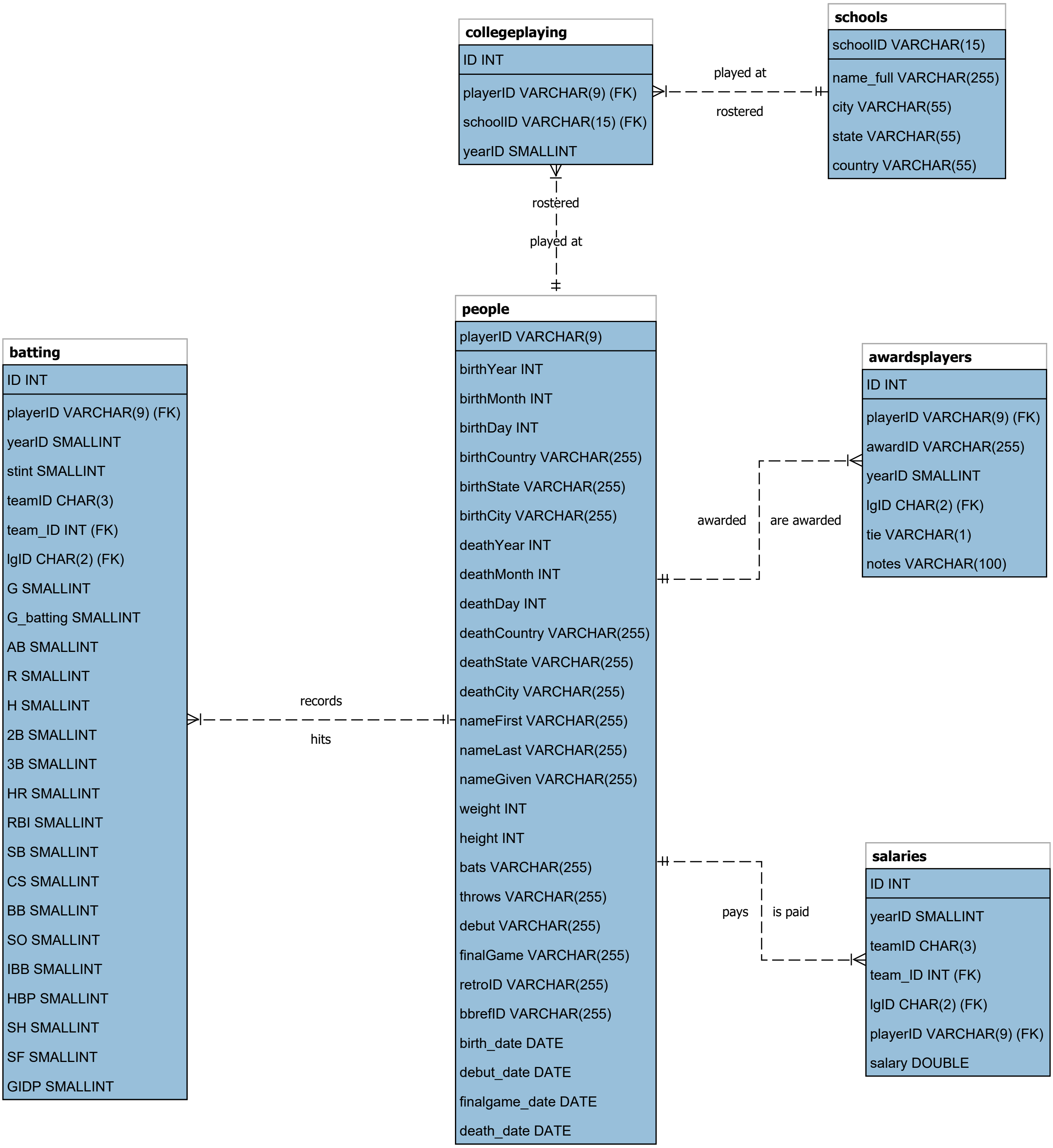
7.



9.



CUSTOMER
PK cus_id
cus_fname
cus_lname
cus_street
cus_city
cus_state
cus_zip
cus_phone
DEVICE
PK dev_serial
dev_model
dev_type
SERVICE
PK serv_id
serv_desc
serv_chrg
PART_LIST
PK,FK serv_id
PK,FK part_num
part_list_qt
REPAIR
PK rep_ref_num
rep_date
rep_desc
PK,FK cus_id
PK,FK dev_serial
PK,FK serv_id
PART
PK part_num
part_desc
part_stock
part_cost
orders
fixes
performs
requires
is required



collegeplaying

ID INT

playerID VARCHAR(9) (FK)
schoolID VARCHAR(15) (FK)
yearID SMALLINT

schools

schoolID VARCHAR(15)

name_full VARCHAR(255)
city VARCHAR(55)
state VARCHAR(55)
country VARCHAR(55)

people

playerID VARCHAR(9)

birthYear INT
birthMonth INT
birthDay INT
birthCountry VARCHAR(255)
birthState VARCHAR(255)
birthCity VARCHAR(255)
deathYear INT
deathMonth INT
deathDay INT
deathCountry VARCHAR(255)
deathState VARCHAR(255)
deathCity VARCHAR(255)
nameFirst VARCHAR(255)
nameLast VARCHAR(255)
nameGiven VARCHAR(255)
weight INT
height INT
bats VARCHAR(255)
throws VARCHAR(255)
debut VARCHAR(255)
finalGame VARCHAR(255)
retroID VARCHAR(255)
bbrefID VARCHAR(255)
birth_date DATE
debut_date DATE
finalgame_date DATE
death_date DATE

batting

ID INT

playerID VARCHAR(9) (FK)
yearID SMALLINT
stint SMALLINT
teamID CHAR(3)
team_ID INT (FK)
lgID CHAR(2) (FK)
G SMALLINT
G_batting SMALLINT
AB SMALLINT
R SMALLINT
H SMALLINT
2B SMALLINT
3B SMALLINT
HR SMALLINT
RBI SMALLINT
SB SMALLINT
CS SMALLINT
BB SMALLINT
SO SMALLINT
IBB SMALLINT
HBP SMALLINT
SH SMALLINT
SF SMALLINT
GIDP SMALLINT

awardsplayers

ID INT

playerID VARCHAR(9) (FK)
awardID VARCHAR(255)
yearID SMALLINT
lgID CHAR(2) (FK)
tie VARCHAR(1)
notes VARCHAR(100)

salaries

ID INT

yearID SMALLINT
teamID CHAR(3)
team_ID INT (FK)
lgID CHAR(2) (FK)
playerID VARCHAR(9) (FK)
salary DOUBLE