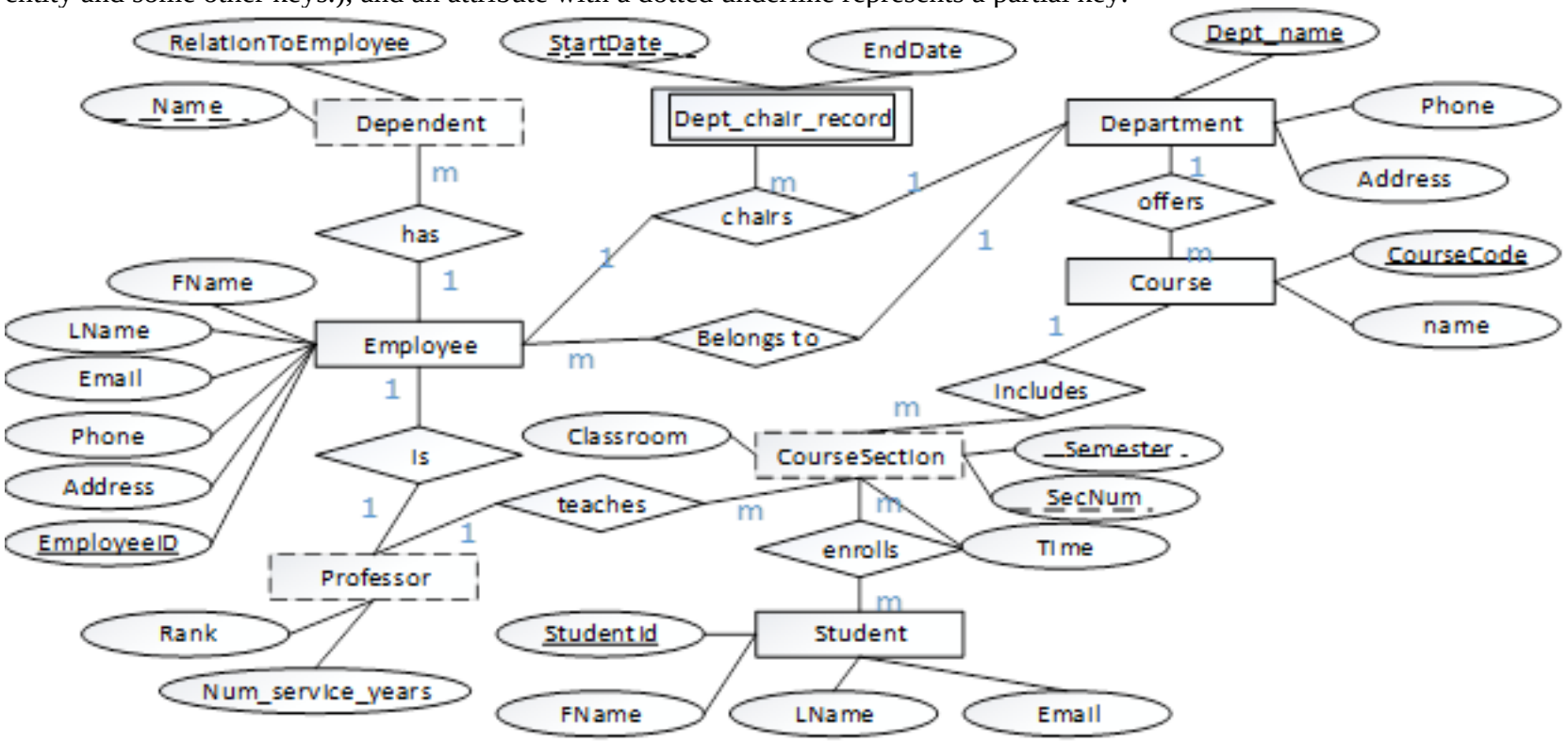
INFS 762
Assignment 2 – Relational Databases
Gavin Gunawardena
Due 10/26/2021 with a one-week grace period

Task 1

Please create a relational model for the following ERD. In the ERD below, a rectangle with dotted-line borders represents a weak entity, a rectangle with double-line borders resents an interaction entity, which can be either weak or strong (In this EDR, Dept-chair-record is a weak interaction entity as it does not have a primary key), an attribute with an underline represents the primary key of an entity (please note, the primary key of an entity and the primary key of the table are not always the same. The primary of an entity is the primary key or a part of the primary key of the table, and the primary key of the table can a combination of the primary key of the entity and some other keys.), and an attribute with a dotted underline represents a partial key.



You don't need to develop a graphical version of the relational databases. Please just list the tables, attributes, primary keys and foreign keys. For instance, you can have
Course (<u>CourseCode</u>, name, Dept_name(FK)) – in this example, you use an attribute with an underline to represent the primary key (Please note the primary key of a table can include multiple attributes) and use (FK) to represent the foreign.

Please note a table can contain multiple foreign keys, and an attribute can be both a primary key and a foreign key. Please specify them clearly.

Please also note, when you develop the relational model, you are not allowed to use surrogate keys – surrogate keys are issues you consider in physical design. They are meaningless and often not useful for queries, but they may be used to enhance query efficiency – anyway, you shouldn't consider surrogate keys during logical design.

It's a little difficult to represent the Dept_chair_record, a weak interaction entity, as a table. As we talked about in the lecture (Rule 12), you have two options. 1) you can add a surrogate key (I will allow you to use surrogate key for just this type of situation) as a primary key to the table; 2) you can figure out its primary key of the table without using a surrogate key – As we talked about in the lecture, the primary key of a weak entity with cardinality *m* could be a composite key that includes the partial key and the primary key(s) of the strong entities this weak entity relies on (in this example, this week entity Dept_chair_record relies on two strong entities Department and Employee), but you need to check if there's partial dependency to reach the 2nd Normal Form. You need to verify if the partial key + the PK(s) of one of the two strong entities is sufficient. I hope you can challenge yourself and try the second option.

Tables

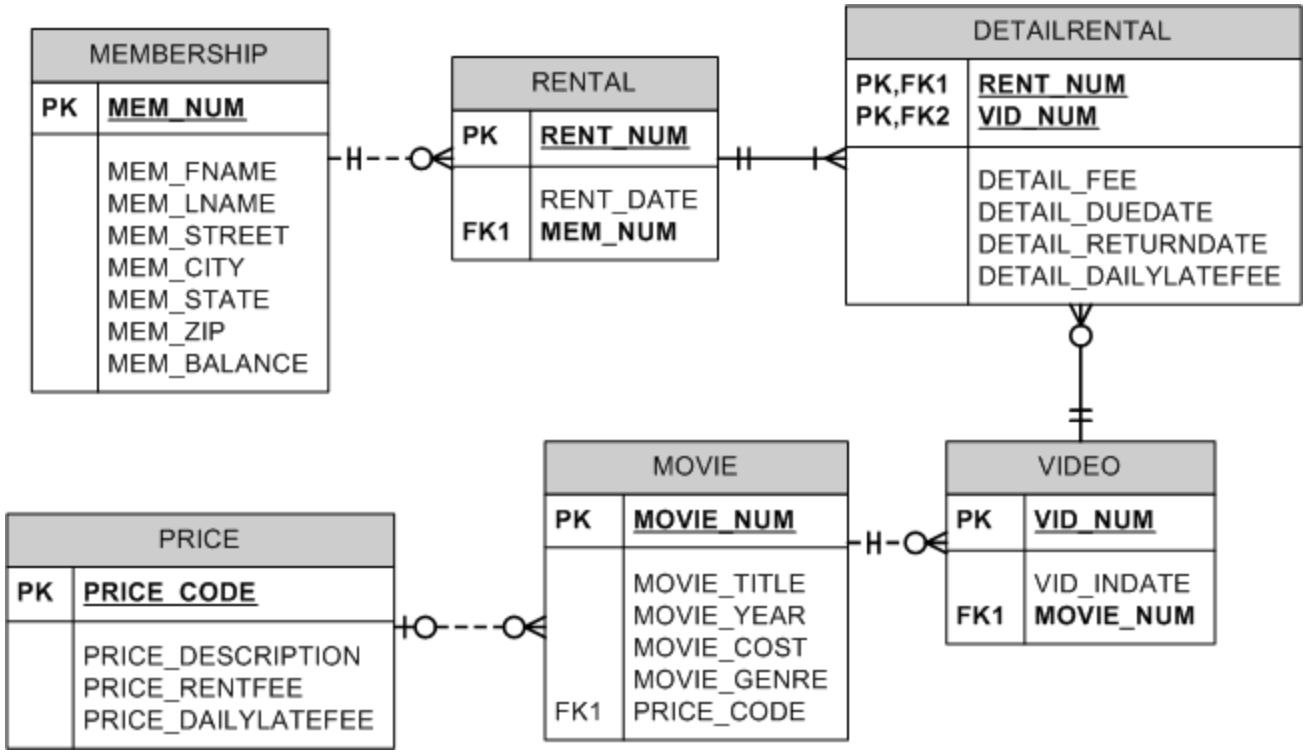
- Employee
  - Fname
  - LName
  - Email
  - Phone
  - Address
  - EmployeeID -PK
- Professor
  - Rank
  - Num_Service_Years
  - EmployeeID -FK -PK
- Dependent
  - Name -PK
  - RelationToEmployee
  - EmployeeID – FK -PK

- Dept_Chair_Record
  - StartDate -PK
  - EndDate
  - EmployeeID -FK -PK
  - Dept_name -FK -PK
- Department
  - Dept_name -PK
  - Phone
  - Address
- Course
  - CourseCode -PK
  - Name
  - Dept_name -FK
- CourseSection
  - Semester -PK
  - SecName -PK
  - Time
  - Classroom
  - CourseCode -FK -PK
  - StudentId -FK -PK
  - EmployeeID -FK
- Student
  - StudentID -PK
  - FName
  - LName
  - Email

Task 2

TinyVideo is a small movie rental company with a single store. TinyVideo needs a database system to track the rental of movies to its members. TinyVideo can own several copies (VIDEO) of each movie (MOVIE). For example, the store may have 10 copies of the movie "Twist in the Wind". "Twist in the Wind" would be one MOVIE and each copy would be a VIDEO. A rental transaction (RENTAL) involves one or more videos being rented to a member (MEMBERSHIP). A video can be rented many times over its lifetime, therefore, there is a M:N relationship between RENTAL and VIDEO. DETAILRENTAL is the bridge table to resolve this relationship. The relational model of this database is show below.

Please run the script (Assignment2Script.txt) available on D2l->Content-> Assignments ->Assignment2 to create the tables and upload the data first. The script is used for SQLite. You can use other DBMS tools for this assignment, but if you are oracle, sqlserver, or mysql, you need to modify the script a little. If you use SQLite but running the script gives an error, you may want to download and use sqlite browser for the assignment.



For the following queries, you just need to submit your SQL statements, and don't need to submit the outputs.

--1.Write the SQL command to change the movie year for movie number 1245 to 2008.
update movie
set movie_year = 2008
where movie_num = 1245;
--2.Write a single SQL command to increase all price rental fee values by $0.50.
update price
set price_rentfee = price_rentfee + .5;
--3.Write a query to display the movie year, movie title, and movie cost sorted by movie year in descending order
select movie_year, movie_title, movie_cost
from movie
order by movie_year desc;
--4.Write a query to display the movie title, movie year, and movie genre for all movies
select movie_title, movie_year, movie_genre

```
from movie;
--5.Write a query to display the movie title, movie year, and movie genre for all movies sorted by movie genre in ascending order, then sorted by movie year in descending order within genre
Select movie_title, movie_year, movie_genre
from movie
order by movie_genre asc, movie_year desc;
--6.Write a query to display the movie number, movie title, and price code for all movies with a title that starts with the letter “R”
Select distinct movie_num, movie_title, price_code
from movie
where movie_title like 'R%';
--7.Write a query to display the movie number, movie title, and movie cost for all movies with a cost greater than $40
Select distinct movie_num, movie_title, price_code
from movie
where movie_cost > 40;
--8.Write a query to display the movie number, movie title, movie cost, and movie genre for all movies that are either action or comedy movies and have a cost that is less than $50. Sort the results in ascending order by genre. (Reminder: because the default order of operations for logical connectors is to evaluate all of the ANDs, then evaluate all of the ORs, it is necessary to either use the IN operator or use parentheses to have the OR evaluated first).
Select movie_num, movie_title, movie_cost, movie_genre
from movie
where lower(movie_genre) in ('action','comedy') and movie_cost < 50
order by movie_genre asc;
--9.Write a query to display the movie genre and the number of movies in each genre.
select movie_genre, count(*)
from movie
group by movie_genre;
--10.Write a query to display the movie title, movie genre, price description, and price rental fee for all movies with a price code
Select movie.movie_title, movie.movie_genre, price.price_description, price.price_rentfee
from movie, price
where movie.price_code = price.price_code and price.price_code IS NOT NULL;
--11.Write a query to display the movie genre and average cost of movies in each genre
select movie_genre, avg(movie_cost)
from movie
group by movie_genre;
--12.Write a query to display the movie title, movie year, price description, and price rental fee for all movies that are in the genres Family, Comedy, or Drama
```

```
Select movie.movie_title, movie.movie_year, price.price_description, price.price_rentfee
from movie, price
where lower(movie.movie_genre) in ('family','comedy','drama');
--13.Write a query to display the movie genre and average price rental fee for movies in each genre that have a price
Select movie.movie_genre, avg(price.price_rentfee)
from movie, price
where movie.PRICE_CODE IS NOT NULL
group by movie.movie_genre;
--14.Write a query to display the movie title, movie year, and the movie cost divided by the price rental fee for each movie that has a price to determine the number of rentals it will take to break even on the purchase of the movie
Select movie.movie_title, movie.movie_year, movie.movie_cost/price.price_rentfee as 'breakeven'
from movie, price
where movie.price_code IS NOT NULL;
--15.Write a query to display the movie title, movie year, and movie cost for all movies that have a cost between $44.99 and $49.99
Select movie_title, movie_year, movie_cost
from movie
where movie_cost between 44.99 and 49.99;
--16.Write a query to display the membership number, first name, last name, and balance of the memberships that have a rental
Select membership.mem_num, membership.mem_fname, membership.mem_lname, membership.mem_balance
from membership, rental
where membership.mem_num = rental.mem_num;
--17.Write a query to display the minimum balance, maximum balance, and average balance for memberships that have a rental
Select min(membership.mem_balance) as 'minimum', max(membership.mem_balance) as 'maximum', avg(membership.mem_balance) as 'average'
from membership, rental
where membership.mem_num = rental.mem_num;
--18.Write a query to display the membership number, last name, and total rental fees earned from that membership. The total rental fee is the sum of all the detail fees (without the late fees) from all movies that the membership has rented.
Select membership.mem_num, membership.mem_lname, sum(detailrental.detail_fee) as 'rental fees earned'
from membership, rental, detailrental
where membership.mem_num = rental.mem_num AND rental.rent_num = detailrental.rent_num

group by membership.mem_num, membership.mem_lname;
```