SCHOOL OF
COMPUTER AND SECURITY SCIENCE

CSG1207/CSI5135: Systems and Database Design
Workshop 03 - Solutions

Standard Disclaimer

Many questions you encounter in this and other workshops have more than one solution which is valid and correct. There are often numerous ways to model data, numerous ways to achieve the same results in an SQL query, etc.

The solutions provided here are NOT the only correct answers to the questions. If you have arrived at solution to a workshop task that differs substantially from what is provided here and would like feedback on your solution, please contact your tutor.

Answers to review questions are not given in these solutions. If you are not confident in your answer to a review question, it is your responsibility to review the relevant unit content or conduct other research until you can answer the question.

Task 1

Answers to review questions are not given. See the Standard Disclaimer for more information.

Task 2

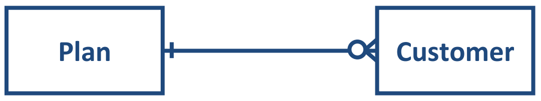
Create physical ER diagrams for the normalised relations (3NF) from Tasks 3 of the second workshop (which were drawn from Tasks 2, 3 and 4 of the first workshop). If you have not completed the first or second workshop, do them now or use the solutions provided.

State any assumptions you make that influence your diagrams.

Workshop 1, Task 2:

Assumptions:

- There may be plans that no customers are signed up to
- All customers are signed up to a plan



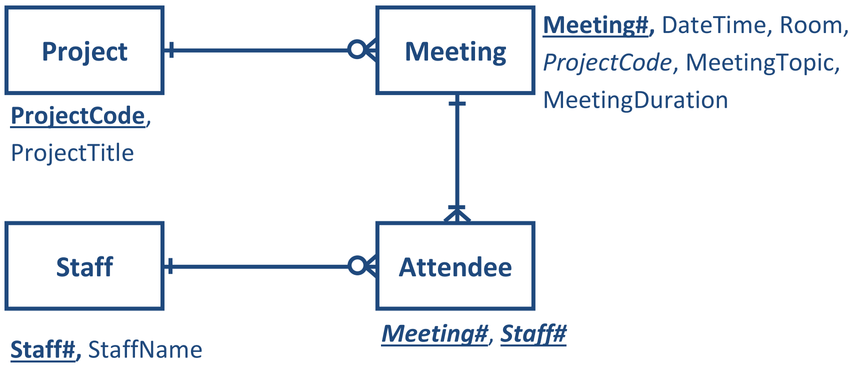
Plan#, PlanName, Quota, Cost

Cust#, *Plan#*, CustName, CustPhone

Workshop 1, Task 3:

Assumptions:

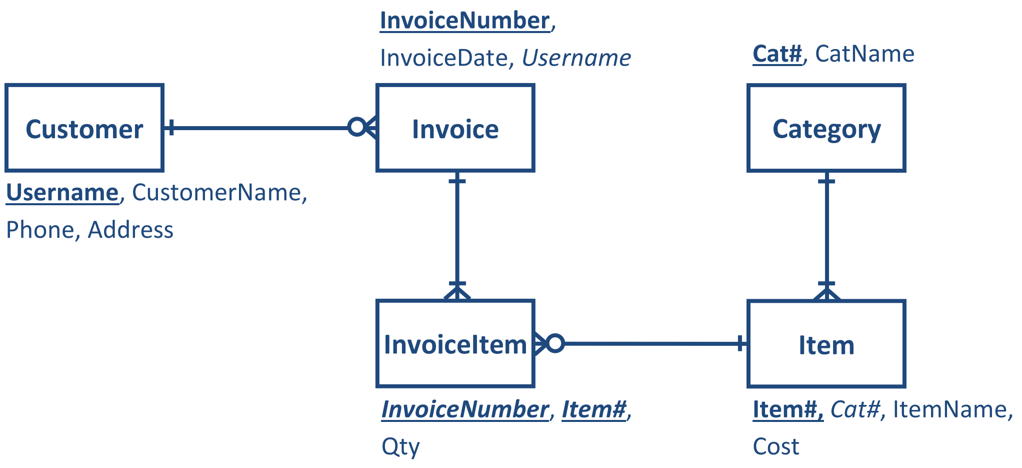
- It is possible for projects to have never had a meeting
- Every meeting must have had at least one attendee
- Staff may never have attended a meeting


Project
Meeting
Meeting#, DateTime, Room, ProjectCode, MeetingTopic, MeetingDuration
ProjectCode, ProjectTitle
Staff
Attendee
Staff#, StaffName
Meeting#, Staff#

Workshop 1, Task 4:

Assumptions:

- It is possible for customers to have an account but have never made an order
- All invoices must have at least one item associated with them
- There may be items which have never been purchased
- All items must be in a category, and all categories must have at least one item in them


InvoiceNumber, InvoiceDate, Username
Cat#, CatName
Customer
Invoice
Category
Username, CustomerName, Phone, Address
InvoiceItem
Item
InvoiceNumber, Item#, Qty
Item#, Cat#, ItemName, Cost

Task 3

Create a physical ER diagram from the following set of normalised relations:

Player (<u>**Username**</u>, Password, DoB, Real Name, Hobbies**)**

Game (<u>**Game ID**</u>**,** Name, Description**)**

Play (<u>**Play#**</u>**,** *Game ID*, Start Time, Duration, *Winner***)**

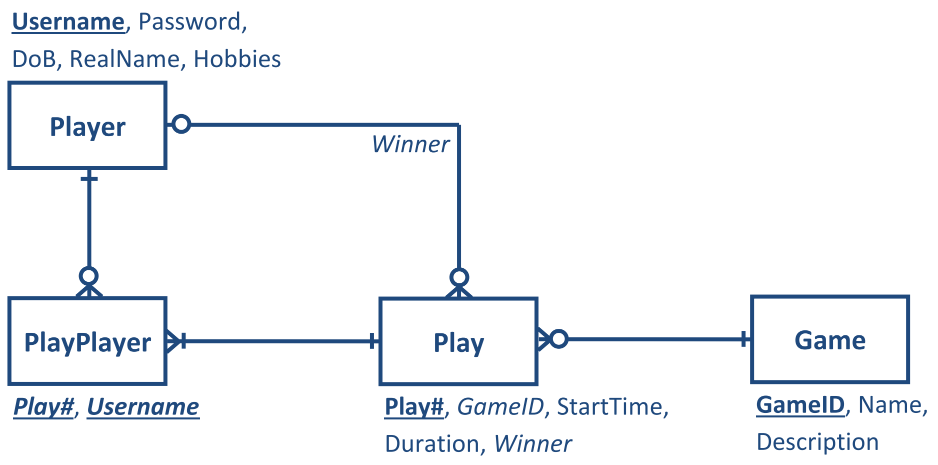
Play Player (<u>***Play#***</u>**,** <u>***Username***</u>**)**

You have the following information about the scenario:

The database is for an online board game website. There is a record in the Play entity for each game that is started, and a record is added to the Play Player entity when a Player joins a game that has been started. The Winner foreign key in the Play entity refers to a Username in the Player entity.

Assumptions:

- There may be games in the database that have never been played
- It is possible for a play of a game to have not had a winner
- It is possible for a player to have never played (or won) a game
- It is not possible for a play of a game to occur without any players joining it (i.e. a record is not added to the Play entity unless the game actually starts, which requires players)



Note: *Naming the foreign key "Winner" in the Play entity makes it less ambiguous. If it had been "Username", the purpose of it would be unclear – could be the player that started/hosted the game, etc.*

Note 2: *You may notice that the "Winner" relationship has a flaw: With this database design, it is possible for the winner of a game to be any of the players in the database – not necessarily one of the players who played in that game. This is something which is likely to be checked by application code as needed, using the content of the PlayPlayer entity to determine which Players were playing in that game.*

Task 4

Create a logical ER diagram for the following problem description, and then convert it to a physical ER diagram. Remember to include cardinality on all relationships. State any assumptions you make that influence your diagrams.

I run a small computer consultancy firm with a number of employees. As well as basic information about the employees (name, DoB, contact details, etc) I need to be able to keep track of what type of role they are able to perform, such as Hardware Technician, Programmer, Software Installer. Many employees can perform multiple roles, and each role has an associated hourly pay.

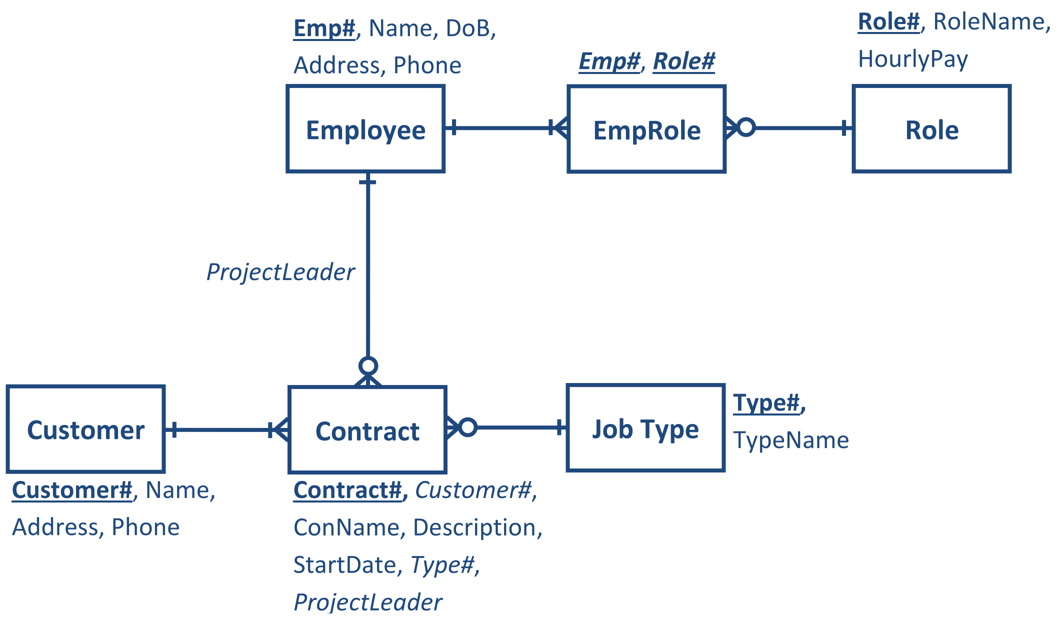
I need to keep name and contact details of any customers that have a contract with us. A customer can have multiple contracts at the same time, but each contract is only associated with one customer. Each contract has a name, a description, a creation date, and a job type – e.g. System Development, Software Upgrade. No details of job types need to be stored other than their name – it simply makes searching easier. Each contract also has a single employee designated as the project leader. One employee may be the project leader of multiple contracts.

As an additional challenge, try to incorporate the following into your ER diagrams. This can be implemented in a number of ways.

I need to be able to record the work done by each employee on each contract. The database must record the employee who did work, the contract it was performed on, the date it was performed on, and the hours of work performed. To determine pay, the role which the employee was performing must also be recorded.

Assumptions:

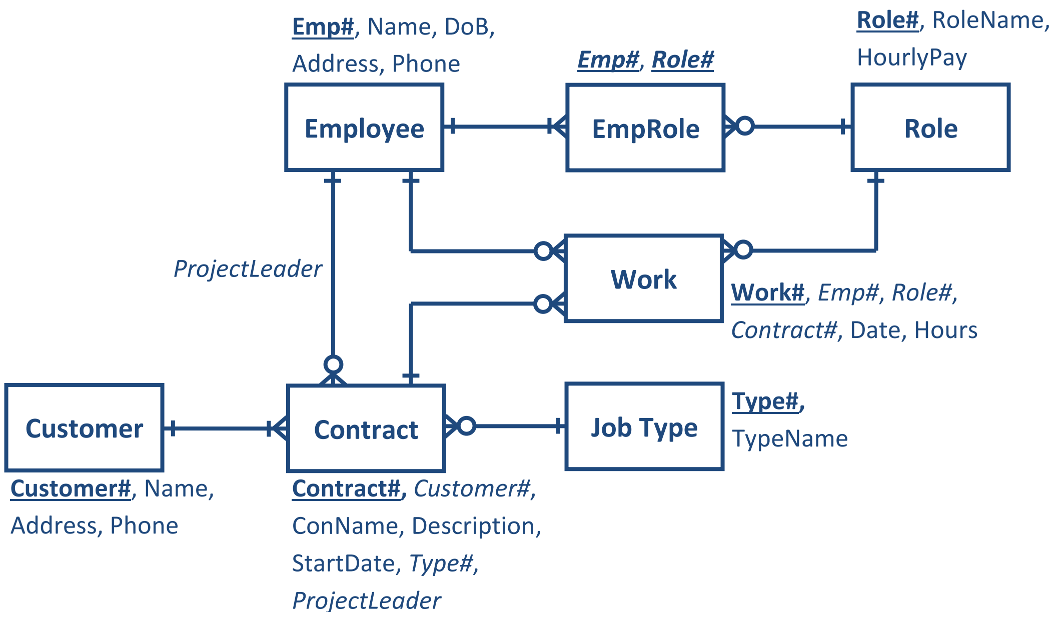
- Every customer must have a contract (otherwise no reason to store details)
- Every employee must be able to perform at least one role
- Some roles and job types may not have anything associated with them



Emp#, Name, DoB, Address, Phone
Emp#, *Role#*
Role#, RoleName, HourlyPay
Employee
EmpRole
Role
ProjectLeader
Customer
Contract
Job Type
Type#, TypeName
Customer#, Name, Address, Phone
Contract#, *Customer#*, ConName, Description, StartDate, *Type#*, *ProjectLeader*

Challenge Version:

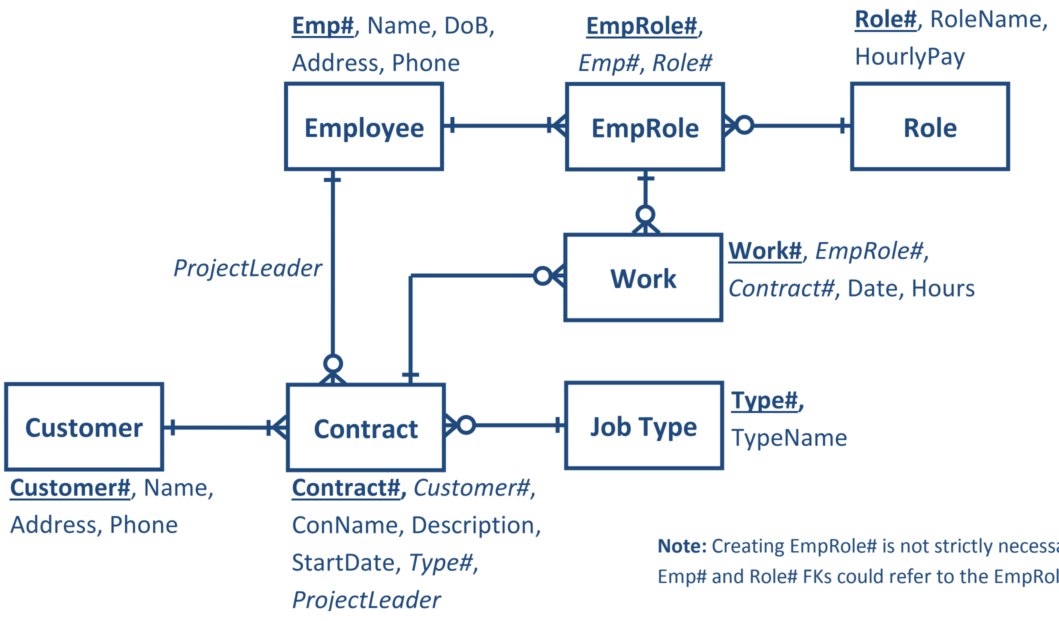
This version of the ER diagram incorporates the challenge condition – the Work entity contains the necessary keys and attributes to record each instance of an employee working on a contract.



Emp#, Name, DoB, Address, Phone
Emp#, ***Role#***
Role#, RoleName, HourlyPay
Employee
EmpRole
Role
ProjectLeader
Work
Work#, *Emp#*, *Role#*, *Contract#*, Date, Hours
Customer
Contract
Job Type
Type#, TypeName
Customer#, Name, Address, Phone
Contract#, *Customer#*, ConName, Description, StartDate, *Type#*, *ProjectLeader*

This version of the diagram is not perfect – the EmpRole entity is already there to contain what role each employee can perform, but the Work table does not take this into account. It would be possible for an employee to work on a contract in a role that they cannot actually perform, since the EmpRole and Work entities are independent and both contain the foreign keys to the Employee and Role entities.

To make the diagram better, we can use the EmpRole to simplify the Work entity and make sure that work can only be performed in the roles an employee can actually perform. First, we add an EmpRole# PK to simplify the compound key of EmpRole, then we use that as a FK in Work. Now, the EmpRole# FK in Work refers to the EmpRole entity, which is ensuring that work can only be done in appropriate roles.



Emp#, Name, DoB, Address, Phone
EmpRole#, *Emp#*, *Role#*
Role#, RoleName, HourlyPay
Employee
EmpRole
Role
ProjectLeader
Work
Work#, *EmpRole#*, *Contract#*, Date, Hours
Customer
Contract
Job Type
Type#, TypeName
Customer#, Name, Address, Phone
Contract#, *Customer#*, ConName, Description, StartDate, *Type#*, *ProjectLeader*

Note: Creating EmpRole# is not strictly necessary – The Emp# and Role# FKs could refer to the EmpRole entity

Task 5

In this week's lecture, we created an ER diagram for the following problem statement:

> *In a given housing unit at any one time, there may be a given number of tenants. Each of these tenants owns specific appliances, however it is possible for certain appliances to be jointly owned by two or more tenants.*
>
> *The strata company requires a database that will allow tenants to track which appliances they have ownership of, and their value, to allow the fair distribution of property when a unit is vacated. At the moment, each tenant fills out a form with his or her driver's licence details along with the serial number, brand, description and cost of any appliance they own.*

The solution we arrived at was based on these three assumptions:

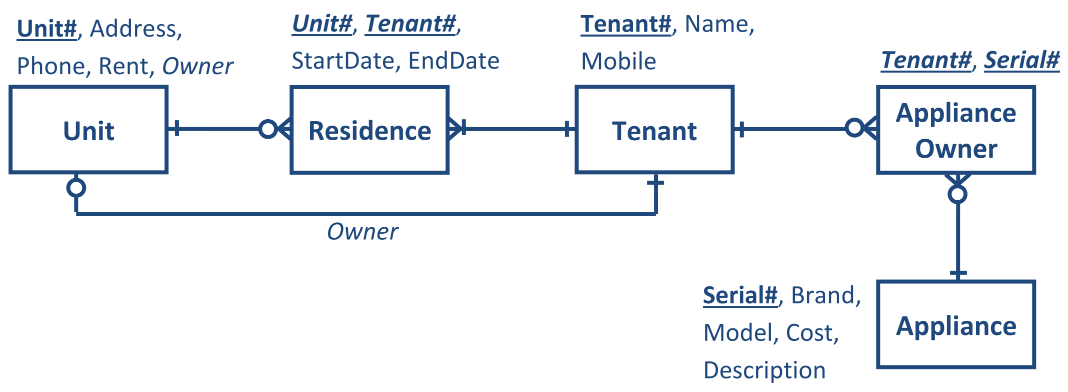
- A tenant will only ever live in one unit
- Ownership of the unit is NOT important/relevant
- Every tenant has a driver's licence (as an identifying field)

Attempt to create a physical ER diagram which does *not* make these assumptions. That is, the database must account for a tenant living in more than one unit, each unit has an owner (who is a tenant), and not all tenants have a driver's licence.

State any assumptions or decisions you have made in order to incorporate these elements.

Assumptions:

- Units may be empty
- A tenant must be living in a unit
- A tenant can only be the owner of one unit



Unit#, Address, Phone, Rent, *Owner*
Unit#, ***Tenant#***, StartDate, EndDate
Tenant#, Name, Mobile
Tenant#, ***Serial#***
Unit
Residence
Tenant
Appliance Owner
Owner
Serial#, Brand, Model, Cost, Description
Appliance