COMP3311 19T3 Database Systems

Exercises 01
Data Modelling, Entity-Relationship Model, (ODL Model)

1. In the context of database application development (aka "database engineering"), what are the aims of **data modelling**?

   Answer:

   Data modelling works from a description of the *requirements* and aims to build a comprehensive description of the entities involved in the application and the relationships among these entities. The model constructed should ensure that all of the requirements can be met i.e. that all of the relevant data is represented and is structured in such a way that all of the operations mentioned in the requirements can be carried out.

   The data model should give enough detail that a database schema can be derived from it without significant additional information being needed.

2. Describe the similarities and differences between the following similarly-named concepts:

   a. **relationship** in the ODL object-oriented design language

   Answer:

   - describes an association between objects in a pair of ODL classes
   - cannot have attributes associated with the relationship
     e.g. you cannot model the sales (and dates) made by a salesman to his customers like this:
     `relationship Set<Struct {Person customer, Date saleDate}> salesMade;`
   - an ODL relationship instance could be implemented as a collection of OIDs stored in an object

   b. **relationship** in the entity-relationship data model

   Answer:

   - describes an association between *n* specific entities
   - a collection of relationships forms a relationship set
   - a relationship set between two entity sets is similar to an ODL relationship
   - a relationship between two entities is similar to an ODL relationship instance
   - a relationship can have associated attributes

   c. **relation** in the relational data model

   Answer:

   - describes an association between a collection of attributes
   - an instance of a relation is a set of tuples of attribute values
   - a relation can be used to model a class or entity set
   - a relation can also be used to model relationship sets (*n*-ary relationships with associated attributes)
   - every relation has a primary key

3. Why is the notion of a **key** not as important in the OO data model as it is in the ER and relational models?

   Answer:

   In the OO data model, each object (instance) has an associated object identifier (OID) that distinguishes it from every other object in the system.

   In the ER and relational models, entity instances and tuples do not have an implicit "instance identifier". They must rely on attributes having distinct values to distinguish instances. A subset of attributes that *does* distinguish all instances (either entity instances or tuples), is a **key**. Without the notion of a key, entity sets and relations (tuple sets) would be bags rather than sets.

   With an OID separate from the attributes, objects in the OO model can be distinguished with needing to rely on attribute values, and so a key is not necessary.

4. What kind of data, relationships and constraints exist in this scenario?

   - Patients are identified by an SSN, and their names, addresses and ages must be recorded.
   - Doctors are identified by an SSN. For each doctor, the name, specialty and years of experience must be recorded.
   - Each pharmacy has a name, address and phone number. A pharmacy must have a manager.
   - A pharmacist is identified by an SSN, he/she can only work for one pharmacy. For each pharmacist, the name, qualification must be recorded.
   - For each drug, the trade name and formula must be recorded.
   - Every patient has a primary physician. Every doctor has at least one patient.
   - Each pharmacy sells several drugs, and has a price for each. A drug could be sold at several pharmacies, and the price could vary between pharmacies.
   - Doctors prescribe drugs for patients. A doctor could prescribe one or more drugs for several patients, and a patient could obtain prescriptions from several doctors. Each prescription has a date and quantity associated with it.

   Answer:

   Possible data:
   - people: doctors, patients, pharmacists, managers(?)
   - pharmacies, drugs
   - person's SSN, name, address(?)
   - doctor: as for person + specialty
   - pharmacist: as for person + qualification
   - etc.

   Possible relationships:
   - doctors **treat** patients, patients **have primary** physician
   - pharmacists **work in** pharmacies
   - drugs **are sold in** pharmacies
   - doctors **prescribe** drugs **for** patients
   - etc.

   Possible constraints:
   - every person has **exactly one, unique** SSN
   - pharamacist works in ≤ 1 pharmacy
   - patient has **exactly one** primary physician
   - doctor treats ≥ 1 patient
   - etc.

5. What kind of data, relationships and constraints exist in this scenario?

   - for each person, we need to record their tax file number (TFN), their real name, and their address
   - everyone who earns money in Australia has a distinct tax file number
   - authors write books, and may publish books using a ``pen-name'' (a name which appears as the author of the book and is different to their real name)
   - editors ensure that books are written in a manner that is suitable for publication
   - every editor works for just one publisher
   - editors and authors have quite different skills; someone who is an editor cannot be an author, and vice versa
   - a book may have several authors, just one author, or no authors (published anonymously)
   - every book has one editor assigned to it, who liaises with the author(s) in getting the book ready for publication
   - each book has a title, and an edition number (e.g. 1st, 2nd, 3rd)
   - each published book is assigned a unique 13-digit number (its ISBN); different editions of the same book will have different ISBNs
   - publishers are companies that publish (market/distribute) books
   - each publisher is required to have a unique Australian business number (ABN)
   - a publisher also has a name and address that need to be recorded
   - a particular edition of a book is published by exactly one publisher

   Answer:

   Possible data:
   - people: authors, editors; organisations: publishers
   - person's name, TFN, address
   - books: published, unpublished
   - book's title, edition, ISBN, ...
   - publisher: ABN, name, address
   - etc.

   Possible relationships:
   - author **writes** books (using pen-name)

- book **is edited by** editor
- etc.

Possible constraints:

- every person has **exactly one, unique** TFN
- every published work has **exactly one, unique** ISBN
- editors and authors must be distinct sets
- editor works for ≤ 1 publisher
- etc.

6. Consider some typical operations in the myUNSW system ...

   - student enrols in a lab class
   - student enrols in a course
   - system prints a student transcript

   For each of these operations:

   a. identify what data items are required
   b. consider relationships between these data items
   c. consider constraints on the data and relationships

   Answer:

   Operation: student enrols in a class.

   Possible data:

   - class information (day, time, location, quota, ...)
   - room information (name, building, capacity, ...)
   - course information (code, title, ...)
   - etc.

   Possible relationships:

   - class enrolment (for classes in this course)
   - course enrolment (for course containing class)
   - etc.

   Possible constraints:

   - cannot enrol in more than one class for single course
   - must be enrolled course before enrolling in classes
   - etc.

   Operation: student enrols in a course.

   Possible data:

   - course information (code, title, syllabus, pre-reqs, ...)
   - course offering information (quota, lic, ...)
   - etc.

   Possible relationships:

   - student enrolment (in previous/current courses)
   - etc.

   Possible constraints:

   - cannot enrol in a full course
   - cannot enrol if don't have pre-reqs
   - cannot enrol in more than 24UC per semester
   - etc.

   Operation: system prints a student transcript.

   Possible data:

   - student information (name, id, ...)
   - course information (code, title, UOC, ...)
   - session information (year, code, dates, ...)
   - etc.

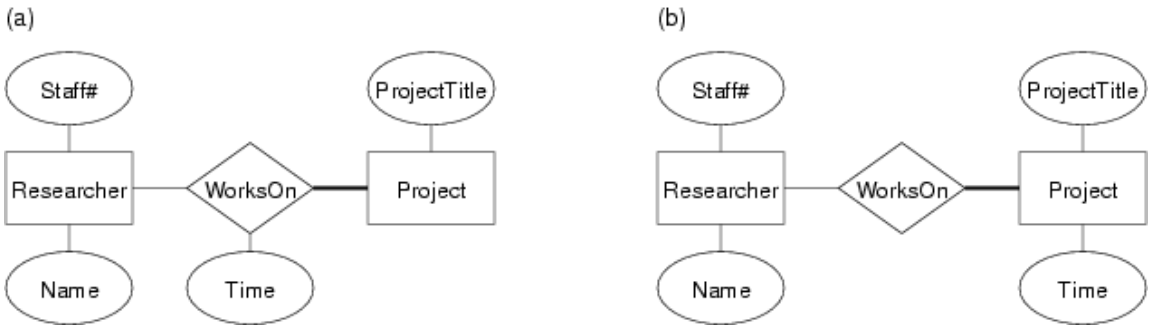
   Possible relationships:

   - student enrolment (previous/current courses, marks/grades)
   - course offering (course, session, ...)
   - etc.

   Possible constraints:

   - none?

7. Researchers work on different research projects, and the connection between them can be modelled by a `WorksOn` relationship. Consider the following two different ER diagrams to represent this situation.



Describe the different semantics suggested by each of these diagrams.

Answer:

ER models for Researchers-WorksOn-Projects relationship.

a. the `Time` attribute is attached to the `WorksOn` relationship, which models the association between a researcher and a project that they work on ... thus, we are modelling the time that each researcher spends on each project that they are involved with

b. the `Time` attribute is attached to the `Project` entity, and so must be a measure of time associated with the project (e.g. the total time allocated to the project)

In both cases, the thick line from `Project` to `WorksOn` indicates that every project must have at least one person working on it. On the other hand, the thin line from `Researcher` to `WorksOn` suggests that not every researcher needs to be working on a project.

8. Consider the following ODL definition to describe researchers working on and managing research projects:

```
interface Researcher {
    attribute string name;
    attribute int staffNumber;
    relationship Project worksOn inverse Project::workers;
    relationship Project manages inverse Project::managedBy;
};

interface Project {
    attribute real number;
    attribute string commander;
    relationship Set<Researcher> workers inverse Researcher::worksOn;
    relationship Researcher managedBy inverse Researcher::manages;
};
```

This definition is based on the assumption that each researcher only works on one project, that a researcher manages at most one project, and that each project has only one manager.

Show how this definition would change if:

a. each researcher's title (e.g. Dr.) is included

b. people's names are broken into separate family and given names

c. each project has a total budget allocation to be recorded

d. a researcher is allowed to be work on more than one project

e. a project may have several managers

Answer:

ODL definition for Researchers-Projects.

```
interface Researcher {
    attribute string name;
    attribute int staffNumber;
```

```
        relationship Project worksOn inverse Project::workers;
        relationship Project manages inverse Project::managedBy;
    };

    interface Project {
        attribute real number;
        attribute string commander;
        relationship Set<Researcher> workers inverse Researcher::worksOn;
        relationship Researcher managedBy inverse Researcher::manages;
    };
```

This definition is based on the assumption that each researcher only works on one project, that a researcher manages at most one project, and that each project has only one manager.

How this definition would change if:

a. each researcher's title (e.g. Dr.) is included

We would add an extra attribute to `Researcher`:

```
        attribute string title;
```

b. people's names are broken into separate family and given names

We would change the `name` attribute to something like:

```
        attribute Struct { string family, string given } name;
```

c. each project has a total budget allocation to be recorded

We would add an extra attribute to `Project`:

```
        attribute float budget;
```

d. a researcher is allowed to be work on more than one project

We would change the definition of `worksOn` in the `Researcher` class (note that nothing else needs to change):

```
        relationship Set<Project> worksOn inverse Project::workers;
```

e. a project may have several managers

We would change the definition of the `managedBy` relationship in the `Project` class:

```
        relationship Set<Researcher> managedBy inverse Researcher::manages;
```

9. Show how you would represent the following OO classes as ER entities with their corresponding attributes:

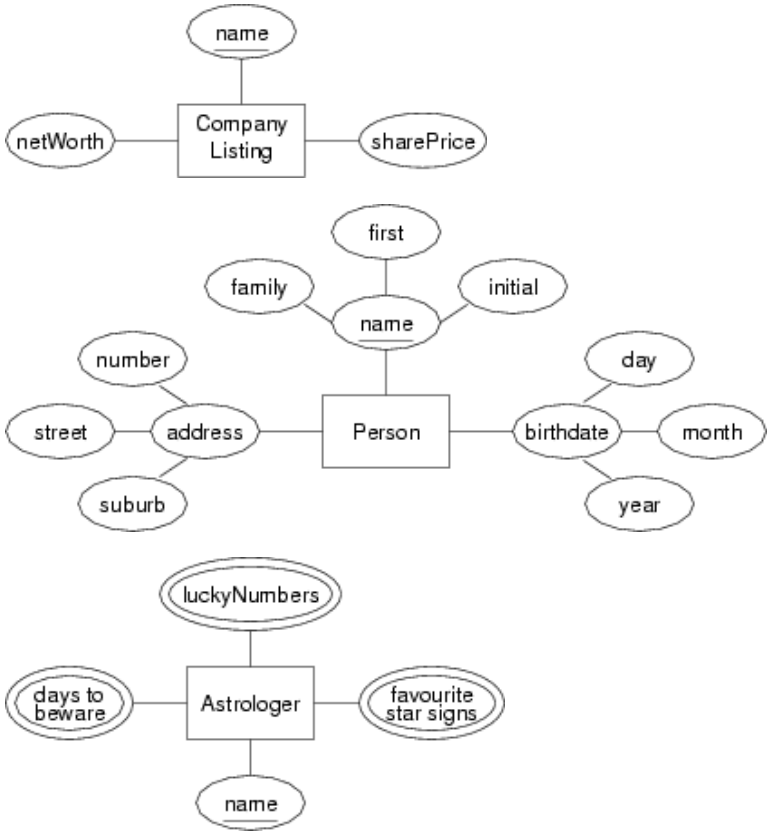
```
    interface CompanyListing {
        attribute string name;
        attribute float sharePrice;
        attribute float netWorth;
    };
    interface Person {
        attribute Struct {integer number, String street, String suburb} address;
        attribute Struct {string family, string first, string initial} name;
        attribute Struct {integer day, integer month, integer year} birthdate;
    };
    interface Astrologer {
        attribute string name;
```

```
        attribute Set<integer> luckyNumbers;
        attribute Set<Enum {capricorn,aquarius,pisces,...}> favouriteStarSigns;
        attribute List<Date> daysToBeware;
    };
```

Assume in the last class that `Date` is an atomic data type.

Answer:

Representations of ODL classes as entity sets:

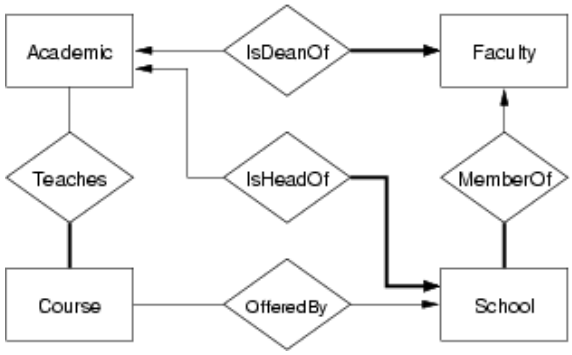


10. Show how you would represent each relationship from the following ODL model in an ER model:

```
interface Faculty {
...
relationship Set<School> schools inverse School::memberOf;
relationship Lecturer dean inverse Lecturer::deanOf;
...
};
interface School {
...
relationship Faculty memberOf inverse::schools;
...
};
interface Lecturer {
...
relationship Faculty deanOf inverse Faculty::dean;
relationship Set<Course> hasTaught inverse Course::taughtBy;
...
};
interface Course {
...
relationship Set<Lecurer> taughtBy inverse Lecturer::hasTaught;
...
};
```

Answer:

Faculty-School-Lecturer-Course relationships as ER diagrams:



Note that the ODL model doesn't say anything about participation (partial/total) constraints on relationships. The participation constraints in the diagram above were based on understanding the semantics of the problem domain (e.g. every faculty has to have a dean, a course must be taught by someone, some academics are pure research and so don't have to teach courses, etc.)

11. Draw an ER diagram for the following application from the manufacturing industry:

    a. Each supplier has a unique name.

    b. More than one supplier can be located in the same city.

    c. Each part has a unique part number.

    d. Each part has a colour.

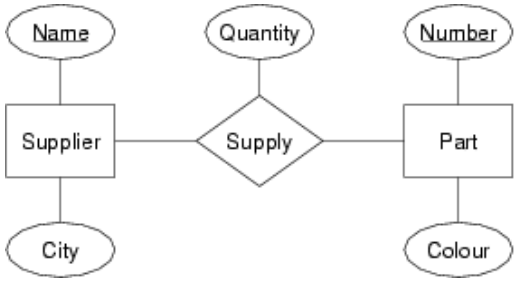
    e. A supplier can supply more than one part.

    f. A part can be supplied by more than one supplier.

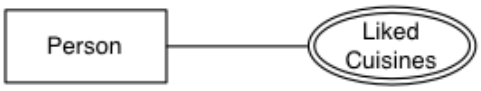
    g. A supplier can supply a fixed quantity of each part.

    Answer:

    ER diagram for Supplier-Part scenario.



12. The following two ER diagrams give alternative design choices for associating a person with their favourite types of food. Explain when you might choose to use the second rather than the first:



    Answer:

    If the only information you needed on each cuisine was its name, either version would be satisfactory. However, if additional information was needed on each cuisine (e.g. country of origin or classic dish or heat-rating), then the second version would be essential, as it is the only one in which this additional information can be associated with the cuisine.

13. [Based on RG 2.2] Consider a relationship `Teaches` between teachers and courses. For each situation described below, give an ER diagram that accurately models that situation:

    a. Teachers may teach the same course in several semesters, and each must be recorded

    b. Teachers may teach the same course in several semesters, but only the current offering needs to be recorded (assume this in the following parts)

    c. Every teacher must teach some course

    d. Every teacher teaches *exactly* one course

    e. Every teacher teaches *exactly* one course, and every course must be taught by some teacher

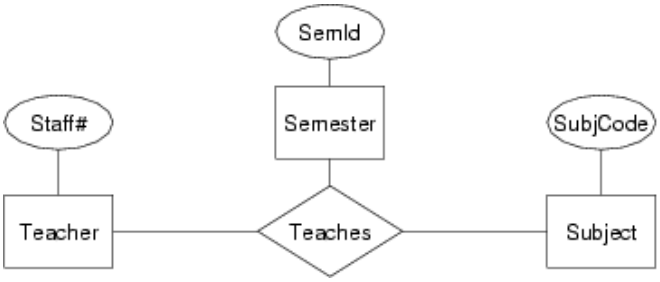
    f. A course may be taught jointly by a team of teachers

    You may assume that the only attribute of interest for teachers is their staff number while for courses it is the course code (e.g. COMP3311). You may introduce any new attributes, entities and relationships that you think are necessary.

    Answer:

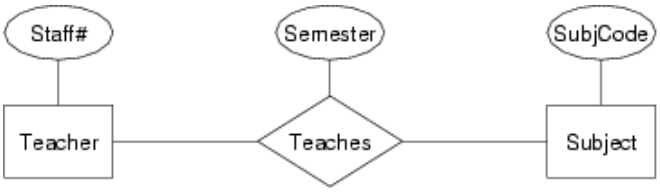
    ER diagrams for Teachers-Teaches-Courses relationship.

    a. Teachers may teach the same course in several semesters, and each must be recorded

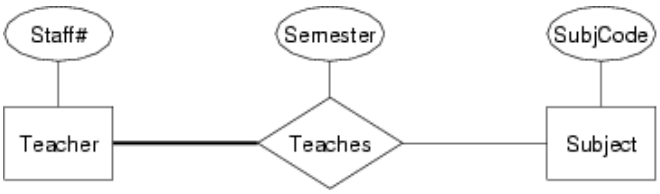


    Note: this solution follows the idea used by some textbooks that relationships must be uniquely identified by the participating entities. The more normal view is that relationships are uniquely identified by the combination of participating entities and attributes, in which case the solution to the next part would be vaild here.

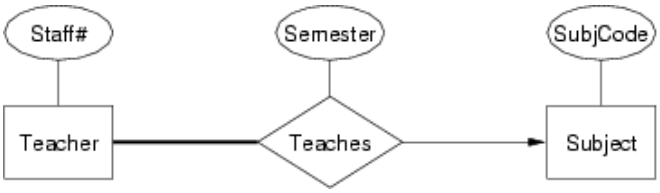
    b. Teachers may teach the same course in several semesters, but only the current offering needs to be recorded (assume this in the following parts)



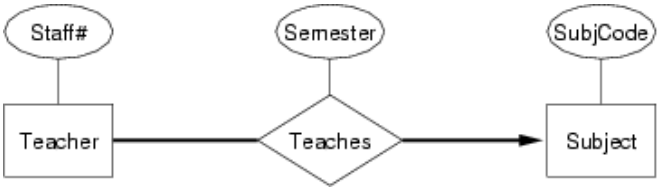
    c. Every teacher must teach some course



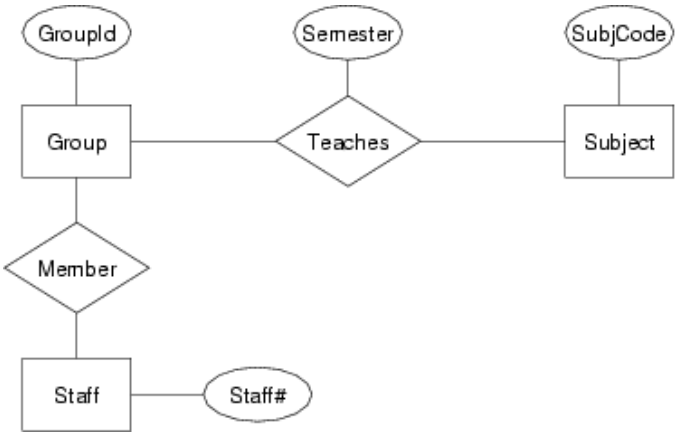
    d. Every teacher teaches *exactly* one course



    e. Every teacher teaches *exactly* one course, and every course must be taught by some teacher



f. A course may be taught jointly by a team of teachers



14. Assume there is a `Person` entity type. Each person has a home address. More than one person can live at the same home address.

    a. Create two, different ER diagrams to depict `Person`s and their addresses, one with `Address` as an attribute, the other with `Address` as an entity.

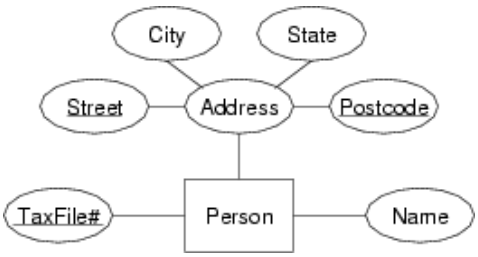
    b. Why would we choose one rather than the other?

    c. Assume that we have a `ElectricCompany` entity type. Only one of these companies supplies power to each home address. Add that information to each ER diagram.

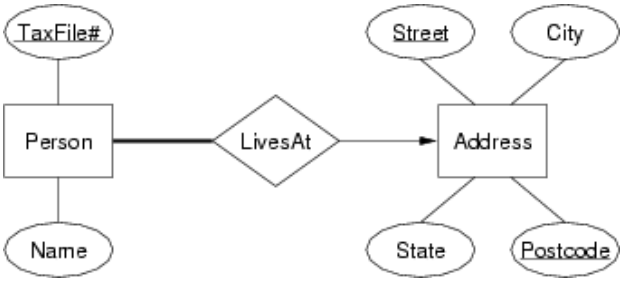
Answer:

ER diagrams for Person-Address-ElectricCompany scenario.

a. Address as attribute



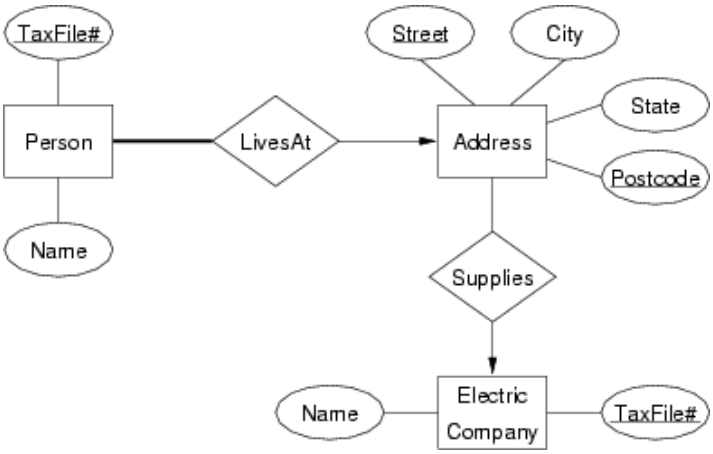
b. Address as entity



In the first example, a person was forced to be associated with exactly one address. In this example, we indicate this constraint via the arrow from `LivesAt` to `Address` and the heavy line from `Person` to `LivesAt`.

c. If the address is an attribute then it is not possible to relate electric companies directly to a home address, instead they can only be related to persons, and through them indirectly related to addresses.

This does not quite capture the information that we wish to model. We wish to relate electric companies directly to home addresses, so we would model the latter as an entity rather than as an attribute. This additional information demands that we favour using an entity for modelling home addresses:



The arrow from `Address` to `ElectricCompany` indicates that each home is supplied by at most one electric company.

15. [Based on GUW 2.1.3] Give an ODL design for a database recording information about teams, players, and their fans, including:

    a. For each team, its name, its players, its captain (one of its players) and the colours of its uniform.

    b. For each player, their name and team.

    c. For each fan, their name, favourite teams, favourite players, and favourite colour.

    Answer:

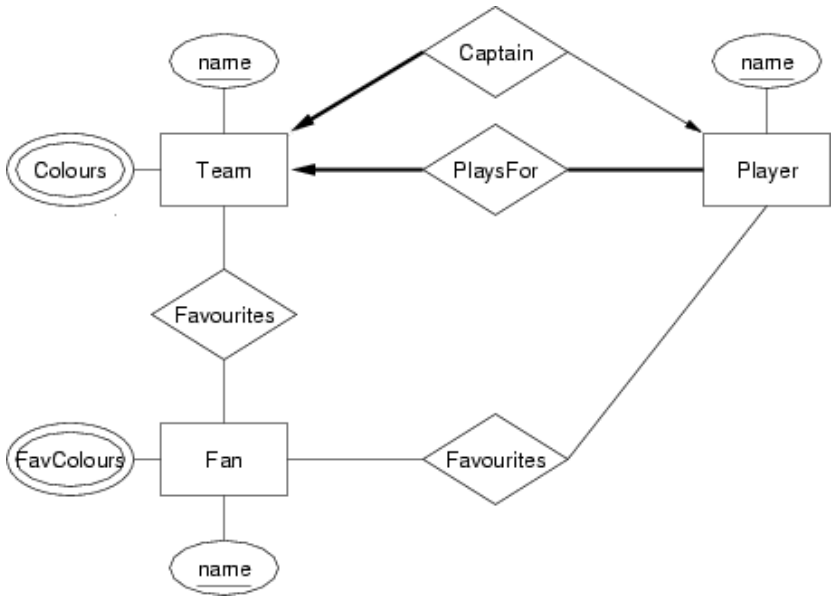
    ODL design for Teams-Players-Fans.

```
interface Team {
    attribute string name;
    attribute Set<Enum {red,green,blue,...}> colours;
    relationship Set<Player> players inverse Player::playsFor;
    relationship Set<Fan> fans inverse Fan::favTeams;
    relationship Player captain inverse Player:captainOf;
};
interface Player {
    attribute string name;
    relationship Team playsFor inverse Team::players;
    relationship Team captainOf inverse Team::captain;
    relationship Set<Fan> fans inverse Fan::favPlayers;
};
interface Fan {
    attribute string name;
    attribute Set<Enum {red,green,blue,...}> favColours;
    relationship Set<Team> favTeams inverse Team::fans;
    relationship Set<Player> favPlayer inverse Player::fans;
};
```

16. Repeat the previous question, but produce an ER design instead.

    Answer:

    ER diagram for Teams-Players-Fans.

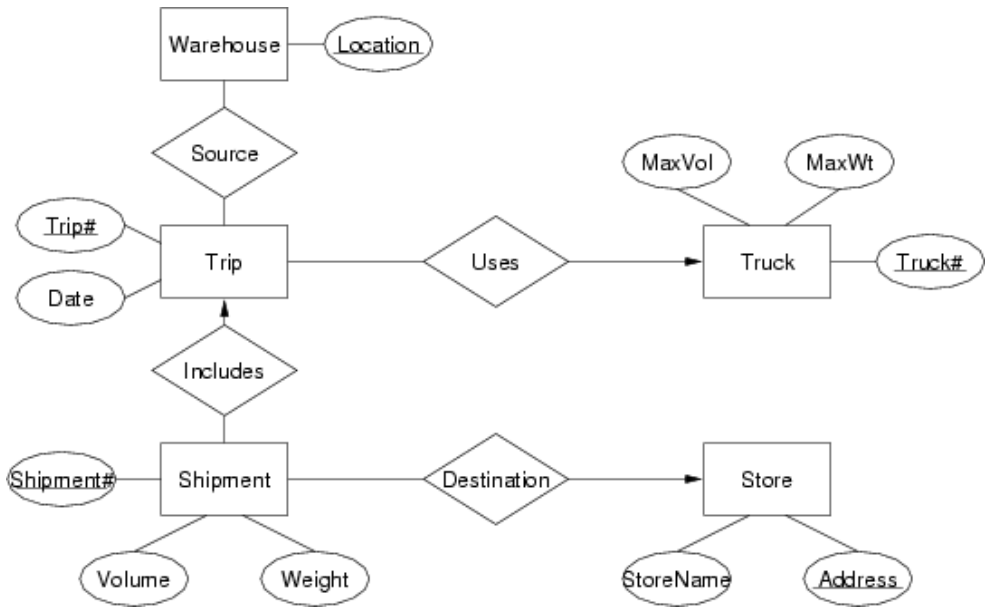


17. A trucking company called "Truckers" is responsible for picking up shipments from the warehouses of a retail chain called "Maze Brothers" and delivering the shipments to individual retail store locations of "Maze Brothers". Currently there are 6 warehouse locations and 45 "Maze Brothers" retail stores. A truck may carry several shipments during a single trip, which is identified by a Trip#, and delivers those shipments to multiple stores. Each shipment is identified by a Shipment# and includes data on shipment volume, weight, destination, etc. Trucks have different capacities for both the volumes they can hold and the weights they can carry. The "Truckers" company currently has 150 trucks, and a truck makes 3 to 4 trips each week. A database - to be used by both "Truckers" and "Maze Brothers" - is being designed to keep track of truck usage and deliveries and to help in scheduling trucks to provide timely deliveries to the stores.

    Design an ER model for the above application. State all assumptions.

    Answer:

    ER design for trucking company.



18. Give an ODL design for a University administration database that records information about faculties, schools, lecturers, students, courses, classes, buildings, rooms, marks. The model needs to include:

    a. for each faculty, its name, its schools and its dean

    b. for each school, its name, the location of its school office, its head and its academic staff

    c. for each lecturer, their names, bithdate, position, staff number, school, office, the courses they have convened, and the classes they have run

d. for each student, their names, birthdate, student number, degree enrolled in, courses studied, and marks for each course

e. for each course, its code, its name, the session it was offered, its lecturer(s), its students, its classes

f. for each class, what kind of class (lecture, tutorial, lab class, ...), its day and time (starting and finishing), who teaches it, which students attend it, where it's held

g. for each building, its name and map reference

h. for each room, its name, its capacity, type of room (office, lecture theatre, tutorial room, laboratory, ...) and the building where it is located

An assumption: staff and student numbers are unique over the union of the sets of staff and student numbers (i.e. each person has a unique identifying number within the University).

Another assumption: the lecturer who "convenes" a course would be called "lecturer-in-charge" at UNSW; lecturers typically teach classes in the courses they convene; they may also teach classes in other courses; a given class is only taught by one lecturer.

State all other assumptions.

Answer:

ODL design for University database

Assumptions:

- each person (staff or student) has an identifying number that uniquely identifies them
- each student undertakes only one degree at the University (at any one time)

```
interface Faculty {
    attribute string name;
    relationship Set<School> schools inverse School::memberOf;
    relationship Lecturer dean inverse Lecturer::deanOf;
};
interface School {
    attribute string name;
    relationship Office office inverse Office::officeFor;
    relationship Lecturer head inverse Lecturer::headOf;
    relationship Faculty memberOf inverse Faculty::schools;
    relationship Set<Lecturer> staff inverse Lecturer::memberOf;
};
interface Person {
    attribute string familyName;
    attribute string givenName
    attribute Date birthdate;
    attribute integer idNumber;
};
interface Lecturer : Person {
    attribute string position;  // could also be Enum
    relationship Office office inverse Office:usedBy;
    relationship School memberOf inverse School:staff;
    relationship School headOf inverse School::head;
    relationship Faculty deanOf inverse Faculty::dean;
    relationship Set<Course> hasConvened inverse Course::convenedBy;
};
interface Student : Person {
    attribute string degree;    // could also be Enum
    attribute List<integer> marks;
    relationship List<Course> studied inverse Course::studiedBy;
    relationship Set<Class> attends inverse Class::attendedBy;
    // Assume the lists are parallel, essentially (Subj,Mark) pairs
    // Unfortunately, they can't be represented directly as pairs
};
interface Course {
    attribute string code;
    attribute string session;
    attribute string name;
    relationship Set<Lecturer> convenedBy inverse Lecturer::hasConvened;
```

```
        relationship Set<Student> studiedBy inverse Student::studied;
    };
    interface Class {
        attribute string classtype;
        attribute Day day;
        attribute integer startTime;
        attribute integer endTime;
        relationship Course classFor inverse Course::classesOf;
        relationship Staff taughtBy inverse Staff::teaches;
        relationship Set<Student> attendedBy inverse Student::attends;
        relationship Room heldIn inverse Room::usedFor;
    };
    interface Building {
        attribute string name;
        attribute Struct{integer x, integer y} mapRef;
        relationship Set<Room> rooms inverse Room::locatedIn;
    };
    interface Room {
        attribute string name;
        relationship Building locatedIn inverse Building::rooms;
    };
    interface Office : Room {
        relationship School officeFor inverse School::office;
        relationship Staff usedBy inverse Staff::office;
    };
    interface ClassRoom : Room {
        attribute integer capacity;
        attribute string roomType;  // could also be Enum
        relationship Set<Class> usedFor inverse Class::heldIn;
    };
```

Some possible variations:

- degrees could be modelled as a separate class (which might specify such information as title, years of study, ...); in this case, there would be a relationship between a student and a degree, rather than the degree being an attribute of the student class; note that this would also allow us to model the scenario where a student takes more than one degree

The above structures do not capture constraints such as:

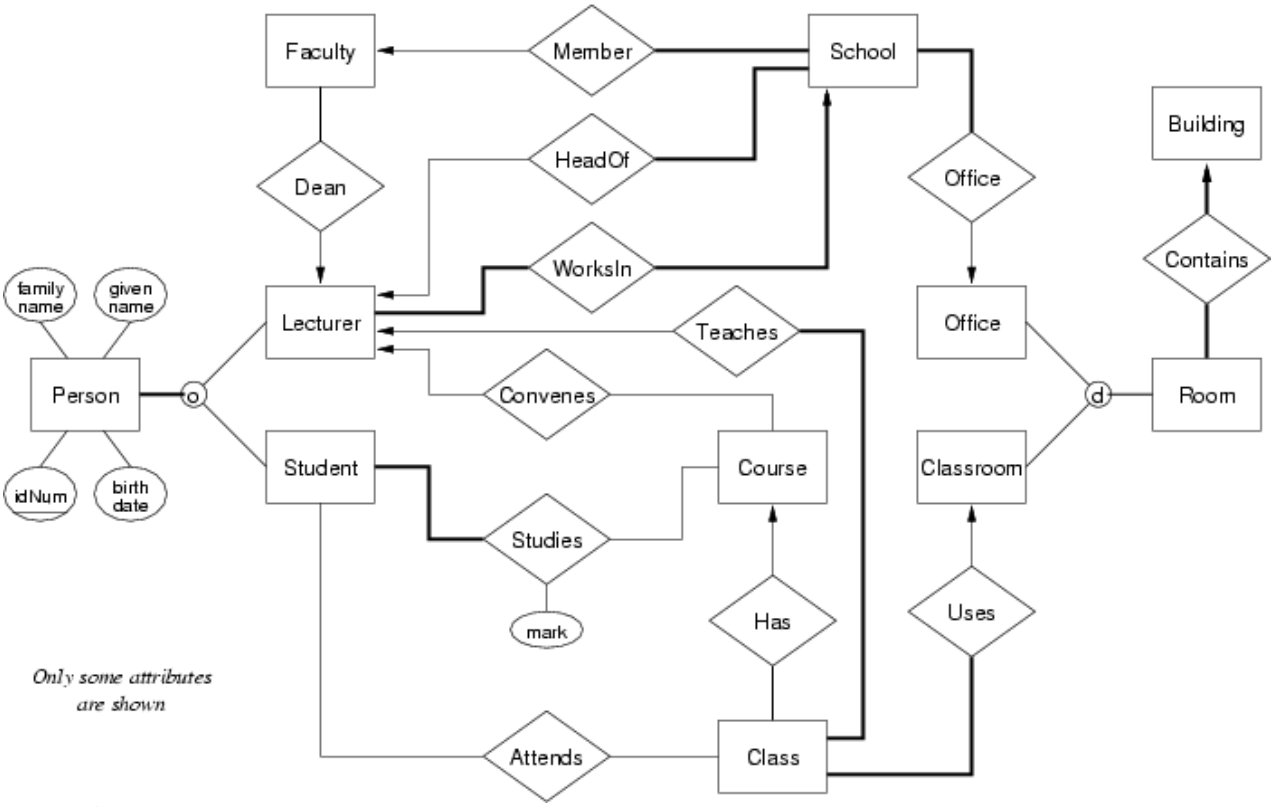
- students only attend classes for courses they are enrolled in
- a lecturer can only be dean of a faculty containing his school

19. Repeat the previous question, but produce an ER design instead.

Answer:

ER design for University database. This is based on the ODL design above, with the additional assumptions:

- every faculty has one dean; every school has one head
- every course has one convener; every class has one teacher
- every student must be studying one or more courses
- not all courses necessarily have classes (e.g. reading courses)
- every class is scheduled in one classroom and is associated with one course
- every class has one teacher



Assumptions:

- each person (staff or student) has an identifying number that uniquely identifies them
- each student undertakes only one degree at the University (at any one time)
- every faculty has one dean; every school has one head
- every course has one convener; every class has one teacher
- every student must be studying/attending one or more courses/classes
- every class is scheduled one classroom and is associated with one course

Some possible variations:

- degrees could be modelled as an entity (which might specify such information as title, years of study, ...); in this case, there would be a relationship between a student and a degree, rather than the degree being an attribute of the student class; note that this would also allow us to model the scenario where a student takes more than one degree

The design does not capture constraints such as:

- students only attend classes for courses they are enrolled in
- a lecturer can only be dean of a faculty containing his school

20. Give an ER design to model the following scenario ...

    - Patients are identified by an SSN, and their names, addresses and ages must be recorded.
    - Doctors are identified by an SSN. For each doctor, the name, specialty and years of experience must be recorded.
    - Each pharmacy has a name, address and phone number. A pharmacy must have a manager.
    - A pharmacist is identified by an SSN, he/she can only work for one pharmacy. For each pharmacist, the name, qualification must be recorded.
    - For each drug, the trade name and formula must be recorded.
    - Every patient has a primary physician. Every doctor has at least one patient.
    - Each pharmacy sells several drugs, and has a price for each. A drug could be sold at several pharmacies, and the price could vary between pharmacies.
    - Doctors prescribe drugs for patients. A doctor could prescribe one or more drugs for several patients, and a patient could obtain prescriptions from several doctors. Each prescription has a date and quantity associated with it.

    State all assumptions used in developing your data model.

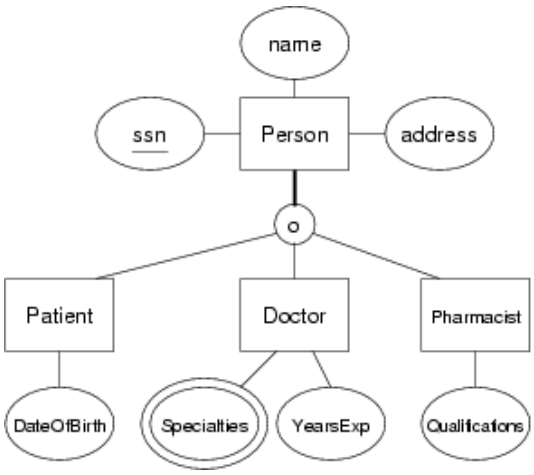
    Answer:

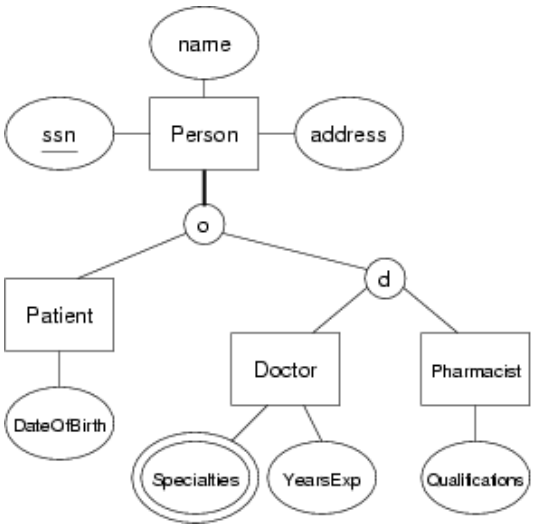
    ER design for medical scenario:

    We first look at the classes of people in the scenario, and then worry about other entities and all of the relationships among them.

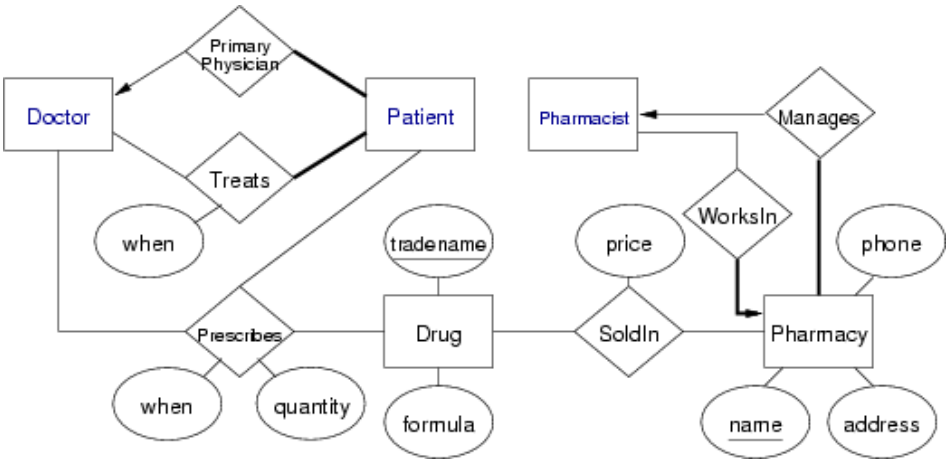
    There are two possible ways to model the various classes of people. Both diagrams inidcate (via a think line) that the only Persons in the system belong to one of the sub-classes. The diagram on the left gives a simple model where any Person in the system can be either a Patient or a Doctor or a Pharmacist. The diagram on

the right is more realistic and indicates that someone who is a Doctor cannot also be a Pharmacist. It is unclear from the formalisms in the texts whether it is necessary to include an intermediate subclass such as HealthProfessional between the (o) and (d) sub-class nodes. I did this in lectures; I've ommitted it here.

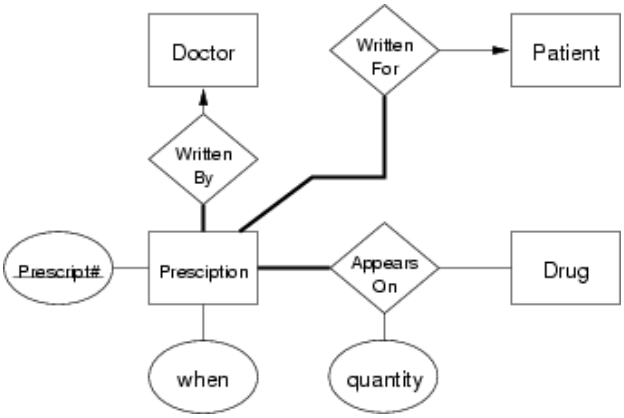




Now we consider how the above entities might fit into the overall data model. Note that the Person entity is not involved directly in any relationships, so we omit it from the diagram. The Doctor, Patient and Pharmacist entities correspond to the ones in the diagram above. Since we have given their attributes on that diagram, we omit them from the following diagram for the sake of clarity.



The above model treats the line-items (i.e. drug+quantity) on a single prescription separately (i.e. it does not model whether two drugs prescribed by the same doctor for the same patient on the same day are part of the same prescription).This may or may not be a problem, but it arises because the model doesn't have an explicit representation for Prescriptions. If we wanted to add that, the revised model would look something like:



Some of the assumptions inherent in the above:

- every prescription has some kind of unique identifying number
- every prescription must be written by a particular doctor
- every prescription must be written for a particular patient
- a prescription must contain some drugs, but may have several

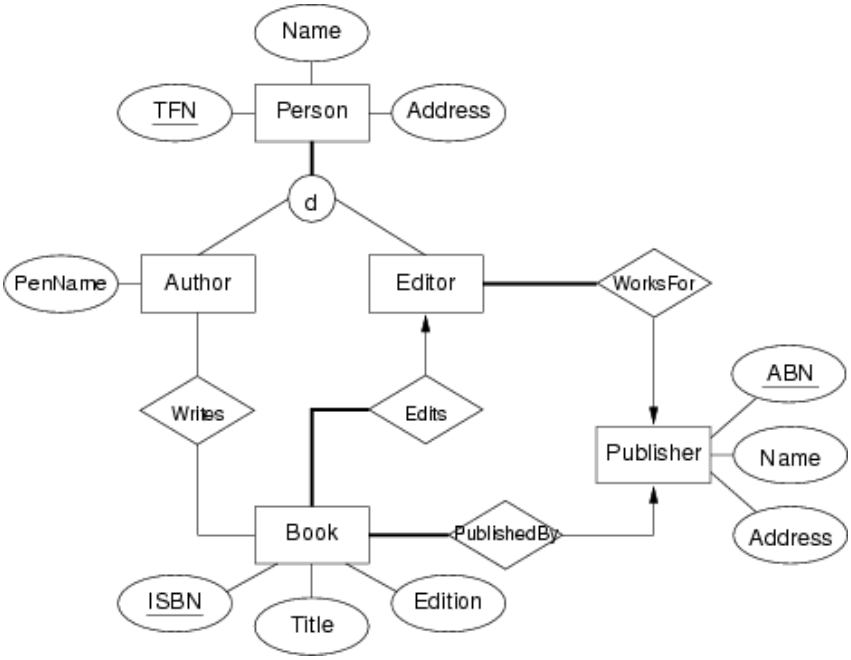
21. Give an ER design to model the following scenario ...

    - for each person, we need to record their tax file number (TFN), their real name, and their address
    - everyone who earns money in Australia has a distinct tax file number
    - authors write books, and may publish books using a ``pen-name'' (a name which appears as the author of the book and is different to their real name)
    - editors ensure that books are written in a manner that is suitable for publication
    - every editor works for just one publisher
    - editors and authors have quite different skills; someone who is an editor cannot be an author, and vice versa
    - a book may have several authors, just one author, or no authors (published anonymously)
    - every book has one editor assigned to it, who liaises with the author(s) in getting the book ready for publication
    - each book has a title, and an edition number (e.g. 1st, 2nd, 3rd)
    - each published book is assigned a unique 13-digit number (its ISBN); different editions of the same book will have different ISBNs
    - publishers are companies that publish (market/distribute) books
    - each publisher is required to have a unique Australian business number (ABN)
    - a publisher also has a name and address that need to be recorded
    - a particular edition of a book is published by exactly one publisher

    State all assumptions used in developing your data model.

    Answer:

    ER design for book publishing scenario:

    Some of the assumptions made in developing this model:

    - authors use only one pen-name for their entire career
      (if they used different pen-names for different books,
      the pen-name attribute would move to the Writes relationship)
    - the only Books recorded are those which are edited and published
    - both ISBN and (Title,Edition) are candidate keys for Books
    - a particular edition is published by one Publisher

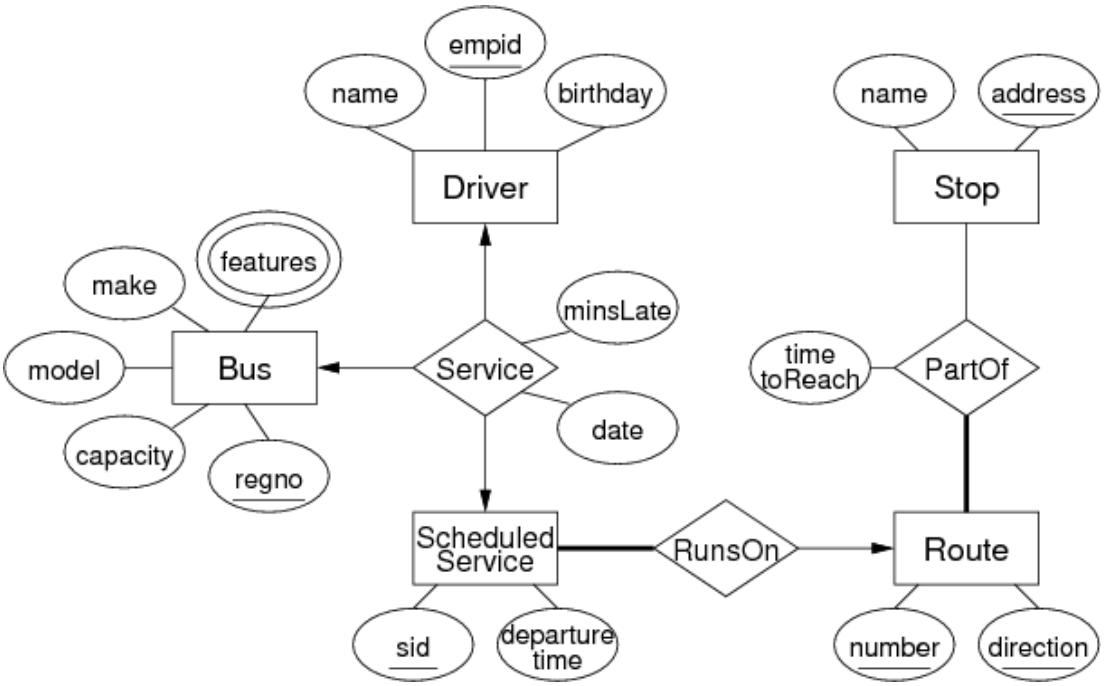
22. Give an ER design to model the following scenario ...

    - a *driver* has an employee id, a name and a birthday
    - a *bus* has a make, model, registration number and capacity
      (e.g. a Volvo 425D bus which can carry 60 passengers, with registration MO-3235)
    - a *bus* may also have features (e.g. air-conditioned, disabled access, video screens, etc.)
    - a *bus-stop* (normally abbreviated to simply *stop*) is a defined place where a bus may stop to pick up or set down passengers
    - each stop has a name, which is displayed on the timetable (e.g. ``Central Station'')
    - each *stop* also has a location (street address) (e.g. ``North side of Eddy Avenue'')
    - a *route* describes a sequence of one or more stops that a bus will follow
    - each *route* has a number (e.g. route 372, from Coogee to Circular Quay)

- each *route* has a direction: ``inbound'' or ``outbound''
  (e.g. 372 Coogee to Circular Quay is ``inbound'', 372 Circular Quay to Coogee is ``outbound'')
- for each stop on a route, we note how long it should take to reach that stop from the first stop
- the time-to-reach the first stop on a route is zero
- stops may be used on several routes; some stops may not (currently) be used on any route
- a *schedule* specifies an instance of a route (e.g. the 372 departing Circular Quay at 10:05am)
- schedules are used to produce the timetables displayed on bus-stops
- a *service* denotes a specific bus running on a specific schedule on a particular day with a particular driver
- services are used internally by the bus company to keep track of bus/driver allocations
- the number of minutes that each bus service arrives late at its final stop needs to be recorded

State all assumptions used in developing your data model.

Answer:



Some of the assumptions made in developing this model:

- every route must have some stops on it; the same stop may be used by several routes
- Info Sys people would probably suggest making **Service** into an entity
  (they don't seem to like *n*-way relationships where *n* > 2)
- we had to introduce a service id to make **ScheduledService** into an entity
  (we could have made it a weak entity tied to **Route** with key (number,direction,departureTime))
- we don't allow drivers to change part-way through a service