THE PLACE OF USEFUL LEARNING

University of Strathclyde Glasgow

Database Fundamentals – CS990

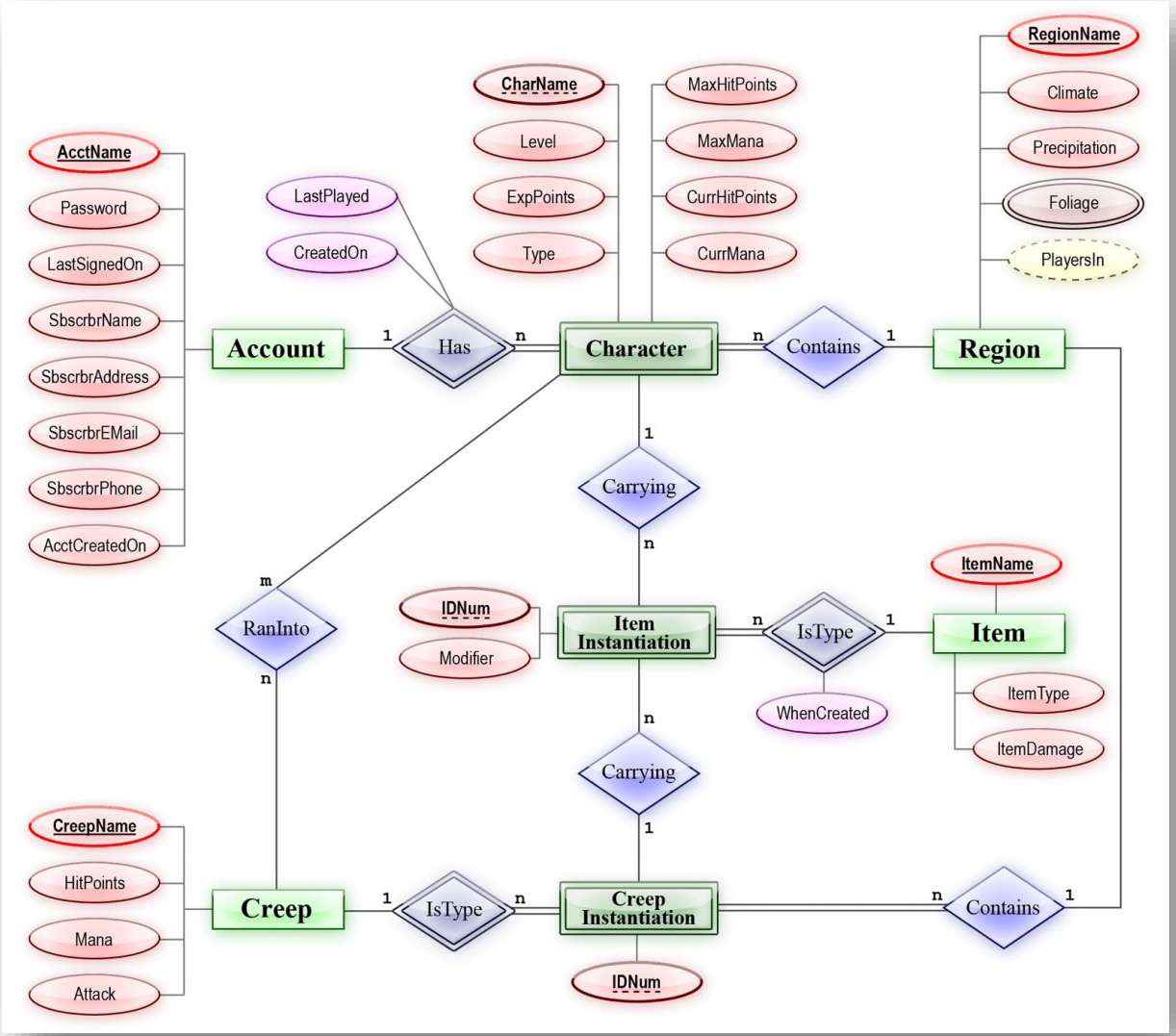
Database and Web Systems Development - CS952

Image source: https://www.strath.ac.uk/branding/

Course Content

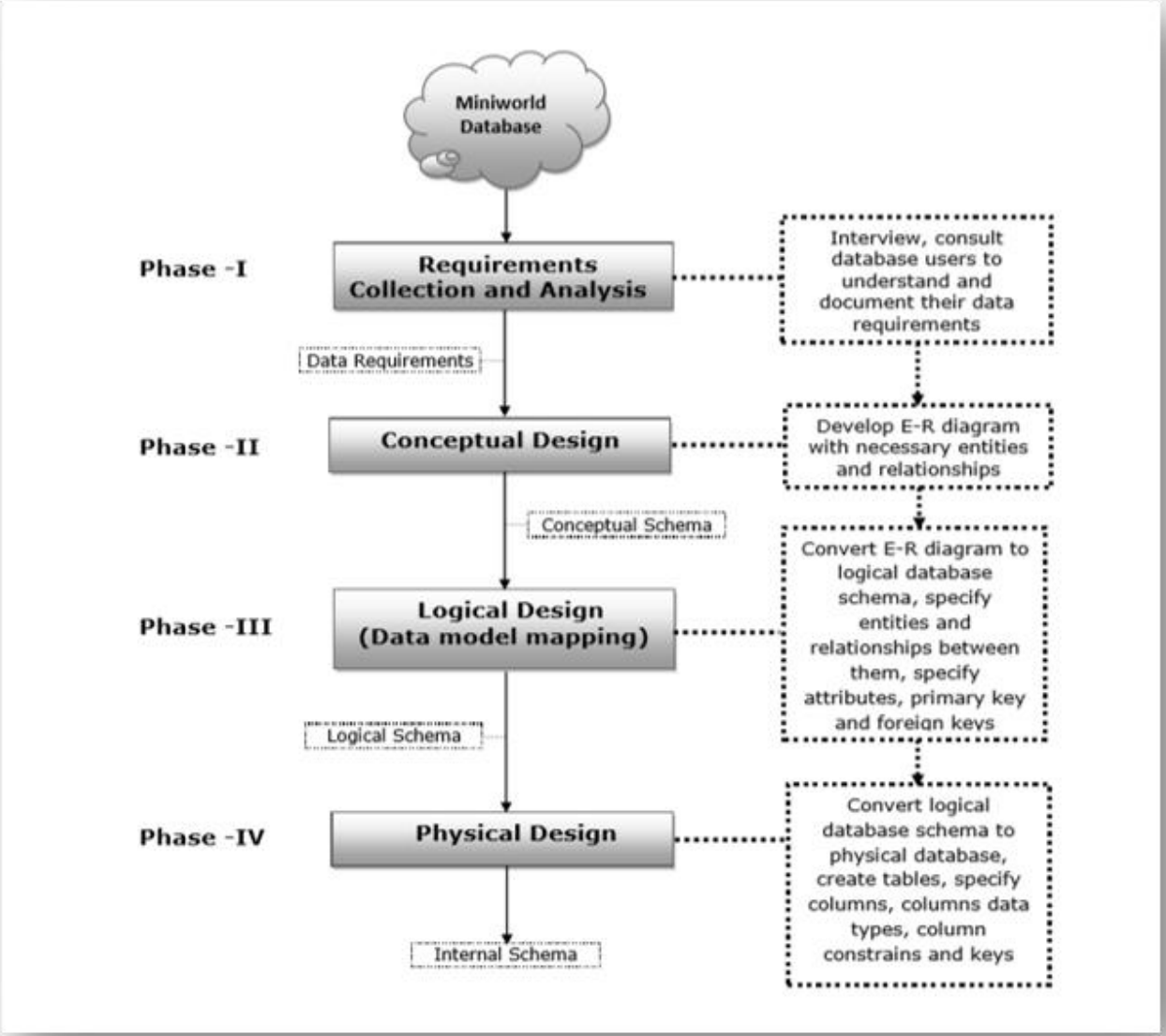
1. Introduction to Relational Databases *(Introduction + Relational Model)*
2. Data Modelling - *(Entity Relationship Modelling + The Enhanced Entity Relationship Model)*
3. Database Design and SQL - *(Logical modelling + Introduction to SQL)*
4. Further SQL - *(Advanced SQL queries + Creating tables with SQL)*
5. Normalisation - *(Normalisation to second normal form + Third normal form)*

ER Modelling



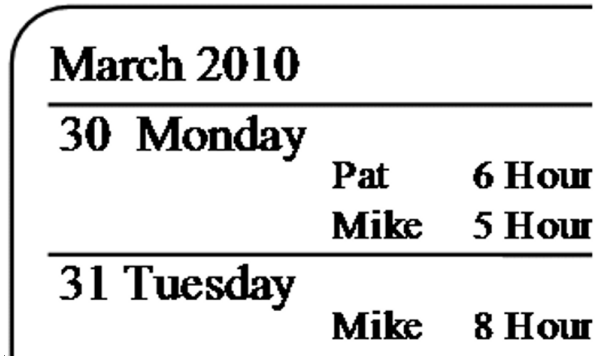
https://en.wikipedia.org/wiki/Entity%E2%80%93relationship_model

Database Design Process Flowchart



https://www.chegg.com/homework-help/questions-and-answers/database-design-process-flowchart-data-models-university-database-system-look-exactly-like-q87822343#question-transcript

Real world



March 2010

30 Monday

Pat	6 Hour
Mike	5 Hour

31 Tuesday

Mike	8 Hour

Conceptual model

Labourer	Day	Hours
Pat	Mon	6
Pat	Tue	8
Pat	Wed	2
Mike	Mon	5
Mike	Tue	6

Logical model hours_worked (ID, Labourer, day, hours)

Physical model labourer (ID, Labourer)
work(IDW, ID, hours)

The Entity-Relationship (ER) Model

- The Entity-Relationship (ER) Model is a high-level *conceptual* data model.
  - It was originally developed in the 1970s to facilitate database design.
- *Recap*: Conceptual data models serve two main purposes:
  - To support a user's perception of data.
  - To conceal the technical aspects of database design.
  - They are also independent of the particular DBMS used to implement the database.
- The basic concepts of the ER model include:
  - *Entities*
  - *Attributes*
  - *Relationships*
- These concepts are represented **pictorially** in an ER diagram.

Some Terminologies

- An **Entity** is an object that exists and is distinguishable from other objects
  - Object . . . Person, Car, City
  - Event . . . . Birth, Goal
  - Activity. . . Oil production from a well
  - Concepts. . Supplies (a relationship!)

- An ENTITY TYPE is distinct from an INSTANCE

- **Attribute** - A property of an Entity.
  - Entity type: Purchase-order
  - Entity instance: A particular purchase order
  - Attributes: Order Number Supplier Date

- An **Identifier** is an attribute or set of attributes that uniquely identify each instance of an entity type.
  - Single attribute identifier Car : Registration Number
  - Multiple attribute identifier Goal : Team1, Team2, Date, Time

Some Terminologies continue...

A **Relationship** is an association between two or more entities

E.g. Relationship **stores** between **product** and **warehouse**.

Relationship type ... **Stores**

Relationship occurrence ... Product 1 is stored in Warehouse 3

Relationship identifier ... A composite of identifiers of entities

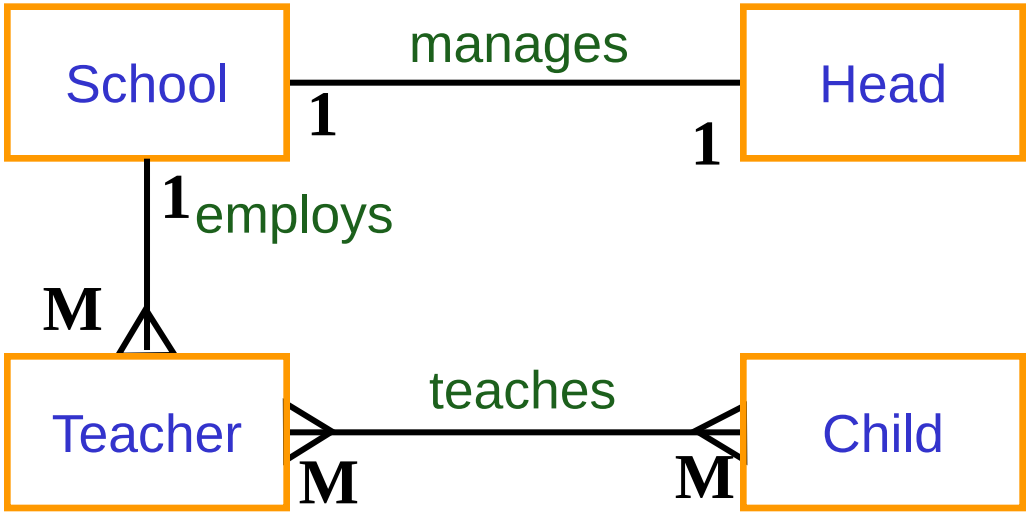
Entities and Relationships - I

- An example of a basic ER diagram is shown below:

- The entities *Student* and *School* are shown as rectangles.
- The relationship *attends* is shown as a labelled connection between the entities.
- We can read this diagram in two ways:
  - One Student "attends" one School (1:1 relationship from point of view of Student).
  - One School "is attended by" one Student (1:1 relationship from point of view of School).

Entities and Relationships - II

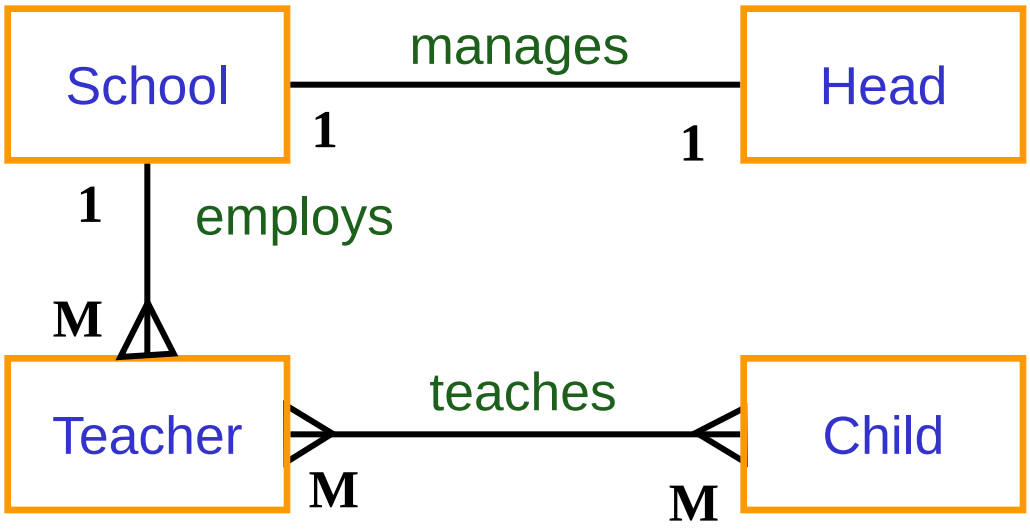
- ER diagrams express relationships between *entity types*.
  - Each entity box in an ER diagram refers to a set of entities, for example a set of students.
  - Each member in the set is potentially involved in the specified relationships.
- Here is a little more complex ER diagram:



Cardinality of Relationships

- In the previous example, some of the connecting lines terminate in a splayed *crowsfoot* arrangement.
  - This is used to denote the *cardinality* of a relationship.
- The cardinality specifies, for one member of the first entity set, the possible number of members it can be related to in a second entity set.
- There are three possibilities:
  - 1:1 One to one e.g. *head* manages *school*
  - 1:M One to many e.g. *school* employs *teacher*
  - M:M Many to many e.g. *teacher* teaches *child*
- The *crowsfoot* device is used to indicate the *many* end of a relationship.

Interpreting ER Diagrams - I



- We can interpret this ER diagram as follows:
  - One *head* manages only one *school* (1:1 from point of view of head) and each *school* has only one *head* (1:1 from point of view of school)
  - One *school* employs many *teachers*, (1:M from point of view of school) but one *teacher* is employed by only one *school* (1:1 from point of view of teacher)
  - One *teacher* teaches many *children* (1:M from point of view of teacher) and each *child* is taught by many *teachers*. (1:M from point of view of child) – hence this relationship is M:M overall (both ways/points of views)

Interpreting ER Diagrams - II

- We always interpret each relationship in an ER diagram from the point of view of one instance of the entity.

  Even in the case of the M:M relationship "*teacher* teaches *child*", we interpret this as:

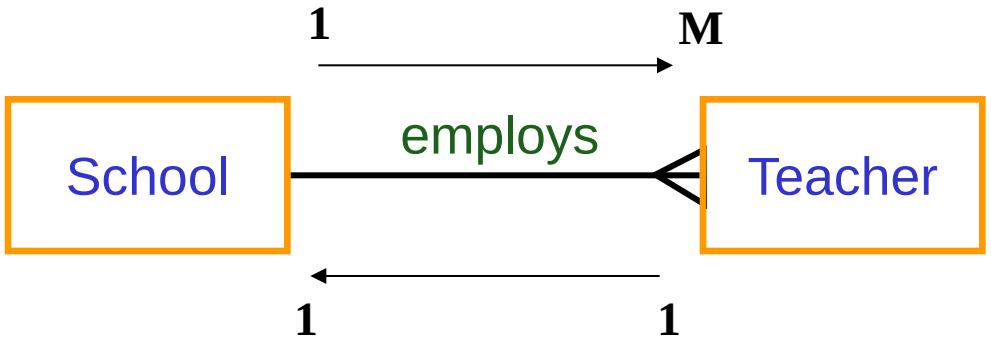
  - one teacher teaches many children (1:M from the point of view of one teacher)
  - one child is taught by many teachers (1:M from the point of view of one child)

- Thus, we always express entity names in the singular to help us remember the above rule.
  - e.g. *teacher* rather than *teachers*.

- For a given relationship we should always be able to make two interpretations.
  - We can interpret the relationship in one direction ("teaches"), and we can interpret it in the reverse direction ("is taught by").

Optionality and Participation - I

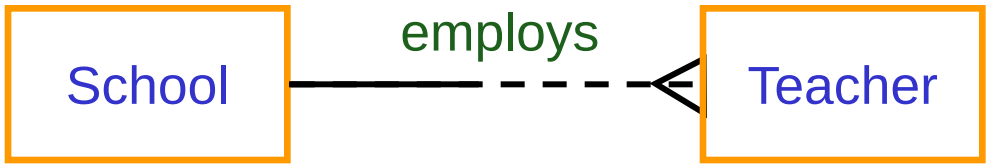
- Consider the 1:M relationship from our earlier example:



- We interpret this as:
  - One school employs one or more teachers. (1:M relationship from point of view of one school)
  - One teacher is employed by only one school. (1:1 relationship from point of view of one teacher)
- However, it may be that some teachers are not employed by any school.
  - We say that the *employed by* relationship is *optional* from the point of view of the teacher.
  - That is, a particular teacher, may or may not by employed by a school.

Optionality and Participation - II

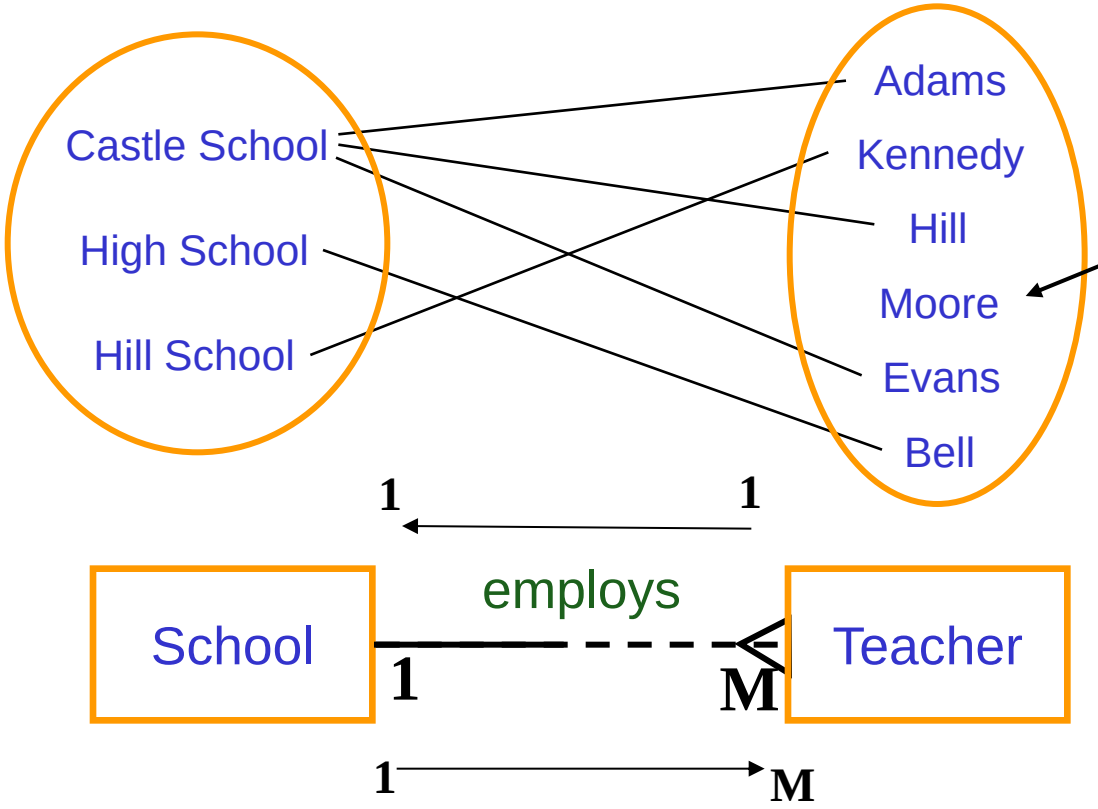
- To express the idea of *optionality* in an ER diagram, we need to draw or relationship connectors slightly differently.
- For example:



- In this diagram, the DASHED portion of the (half) line (connecting to the Teacher entity in the ER diagram above) signifies that a teacher may optionally be employed by a school
- However, from the point of view of a particular school, it is vital that the school employs at least one teacher.
  - Thus, schools must always employ teachers.
  - This is indicated by the SOLID HALF of the connector attached to *School* above

Visualising Relationships

- Relationships represent a mapping from one entity set to another.
  - For example a 1:M relationship map an entity in one set to one or more entities in another set.
- We can visualise a relationship using a mapping diagram:



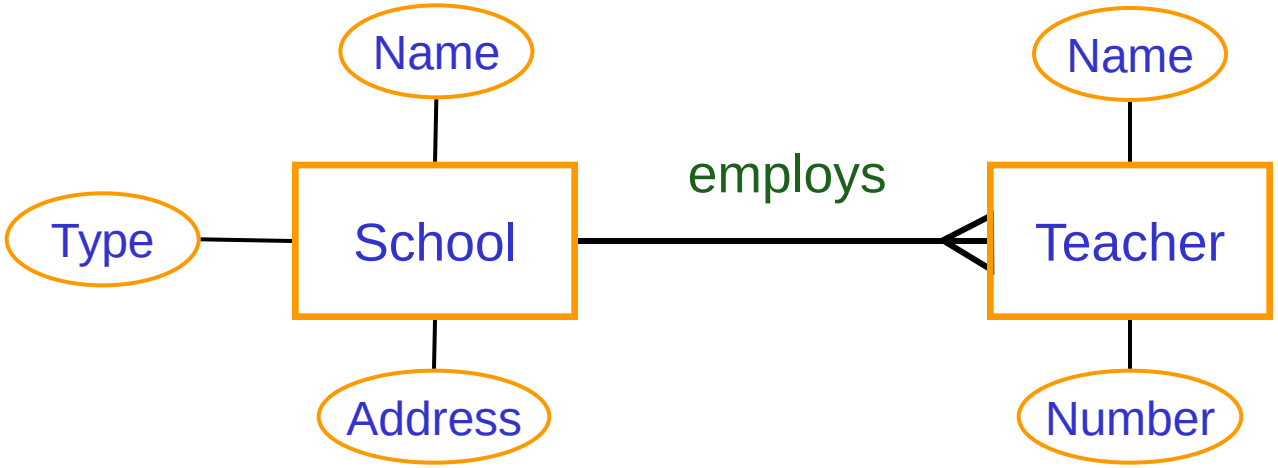
Not employed by any school (i.e. teacher's participation in the relationship is optional rather than mandatory – which is indicated by dashed half line connecting to the teacher entity in the ER diagram)

Displaying Attributes in ER Models-I

- The ER diagrams that we have seen up until now have simply showed structure.
- We now expand slightly by thinking about the properties or *attributes* of each entity.
- For example, in the school example possible attributes are:

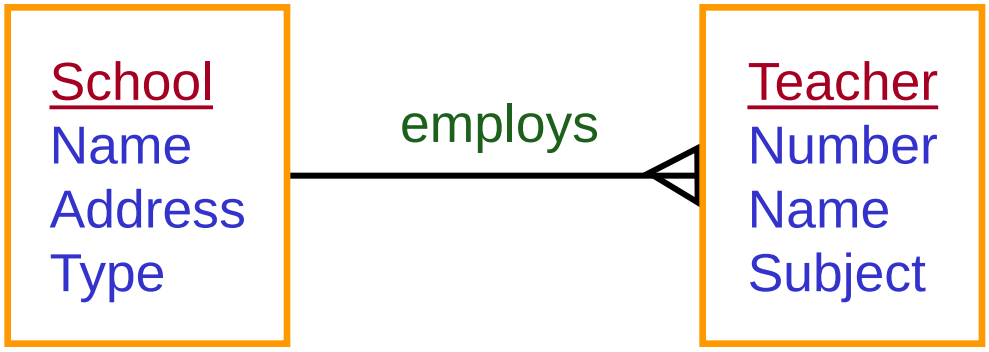
  Name, Address, Type of School, Name and Number of Teacher.

  – We can show this information in an ER diagram as follows:



Displaying Attributes in ER Models-II

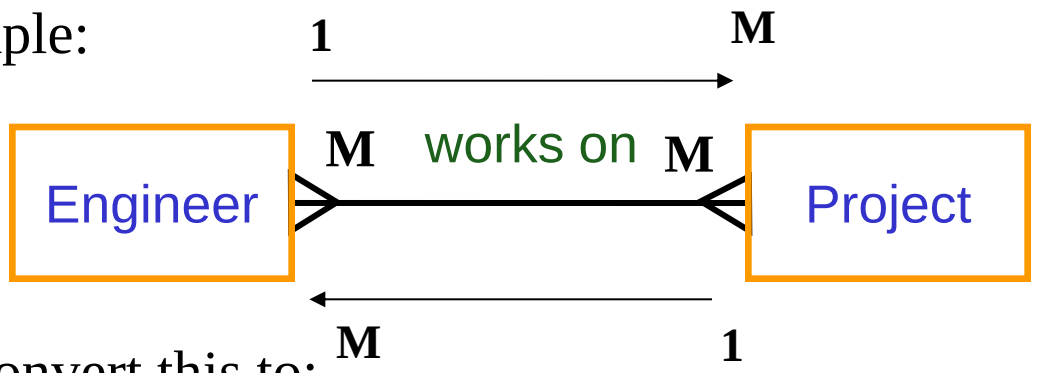
- In practical ER diagrams, displaying attributes in the manner shown on the previous slide tends to clutter the diagram.
- An alternative notation for displaying attributes is shown below:



- In practice, we often have to create ER diagrams from a description, either verbal or written, of what is required.
  - The textbook contains numerous examples of these.

Many-to-Many Relationships - I

- On first analysis of a given system, some relationships may appear to be M:M.
- Sometimes it is possible to convert a M:M relationship into two 1:M relationships.
- For example:



- We can convert this to:

Many-to-Many Relationships - II

- The previous example effectively turns the original M:M relationship into an entity.
  - As it happens, this *Contract* entity is something that is reasonably natural to represent.
  - It may well have its own attributes that we feel are important.
- It is a good idea to examine all M:M relationships to see if they can be converted as shown.
  - Often this reveals a hidden entity that has been overlooked.
  - *It simplifies the process of translating an ER model of a system into a set of relations for use with a relational database.*

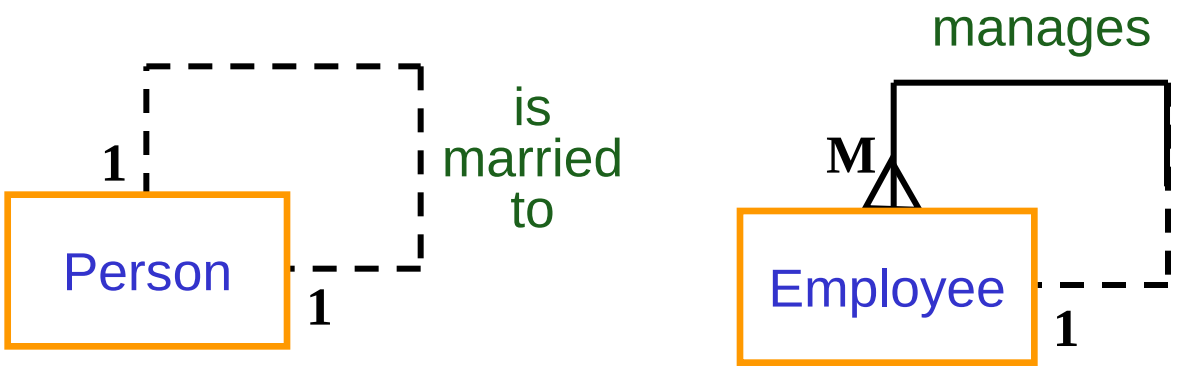
Weak Entities

- A *weak entity* is one which cannot exist without the existence of some other entity.
- For example, in a video hire shop, we may have the following ER diagram:

- This represents the fact that a given movie must be recorded on one or more DVDs, and a given DVD must contain a particular film.
  - So, we cannot have DVDs for which there is no corresponding movie.
  - If we delete a movie from our database, we would also like to delete all of the information about the DVDs containing that movie.
  - Thus, *DVD* is a weak entity since it depends on a corresponding movie.

Unary Relationships

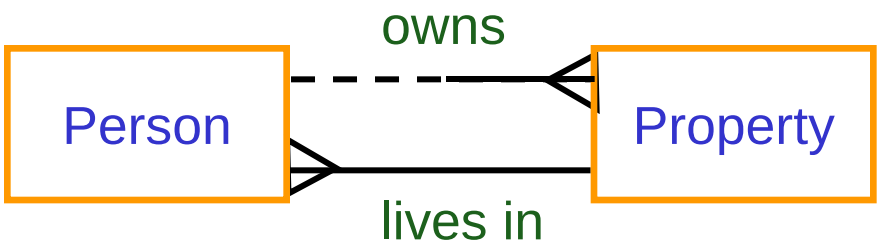
- We said earlier that a relationship expresses a mapping between an entity from one set and one or more entities from another set.
- We sometimes need to show a relationship between entities from the same set. Some typical examples are:



- Left: A person is optionally married to one other person.
- Right: An employee manages zero or more other employees. Each employee is managed by one other employee.

Multiple Relationships Between Entities

- So far we have had only one relationship between any two entities.
- However, there is nothing to stop us having more than one if it makes sense to do so.
  - For example, if two entities can be related to each for different reasons simultaneously.
- Here is a typical example:

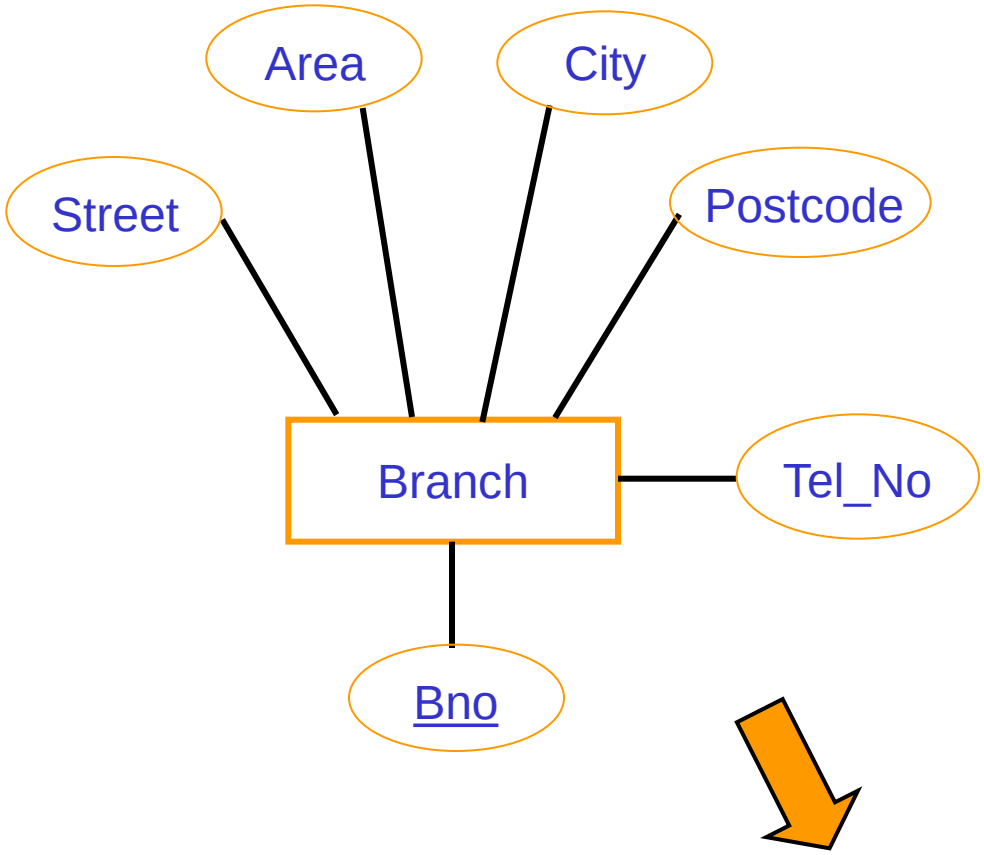


- Here we consider that owning a property is a separate concept from living in it. Thus, we represent this using separate relationships.

Converting an ER model to Relations

- How can we translate an ER model into a set of relations?
- We start by defining a number of relations to represent each of the entities present in our ER model.
- Columns are then added to each relation to represent the attributes of the corresponding entity.
- Relationships are represented using *foreign keys*, or possibly a separate relation.
- Identify primary and foreign keys (if any) in each relation.
  - This last step normally happens in parallel with the rest of the steps.
- We will work through some examples to see how the conversion process works.

Converting Entities Into Relations



Branch (Bno, Street, Area, City, Postcode, Tel_No)

Representing Relationships - I

- Relationships are represented differently depending on their cardinality (1:1, 1:M, or M:M).
- Consider a 1:M relationship such as:

- To represent this relationship we start off with the two relations:
  - *Dept(Dno, Name, NumRooms)*
  - *Staff(Sno, Name)*
- These relations simply represent the two entities and some sample attributes.
- To create the relationship we must link each staff member to his/her corresponding department by embedding a *foreign key*.

Representing Relationships - II

- Thus we extend the *Staff* relation (which is the Many side) to include an extra field, the foreign key *DeptNo*.
- This produces the following result:

DEPARTMENT		
DeptNo	**Name**	**NumRooms**
31	Computing Science	18
49	Management	15
55	Basket-Weaving	3

STAFF		
Sno	**Name**	**DeptNo**
SG86	David Hulse	31
SP52	Paul Kingston	49
SJ12	Michael Smith	55
SQ63	Alan Dearle	31

- By embedding *DeptNo* as a foreign key in the *Staff* relation, we create a 1:M relationship from departments to staff members.
- This is illustrated in the diagram since each staff member can be associated with only one department, but each department may have many associated staff members.
- Always be careful about which relation the foreign key is added to (it is always the relation on the Many side of the relationship!).

Representing Many-to-Many Relationships

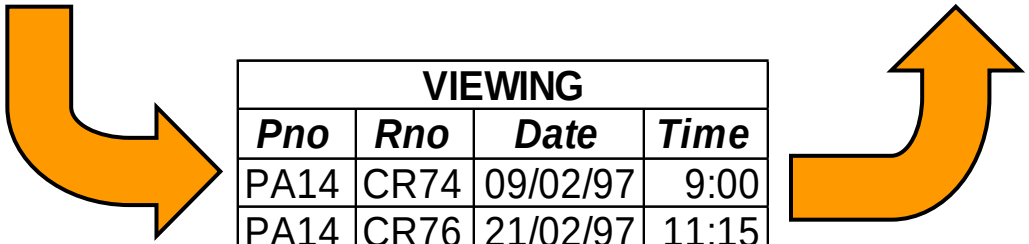
- To handle a many-to-many relationship, such as the *Viewing* relationship between *Properties* and *Renters*, we introduce a separate relation to represent the M:M relationship:

PROPERTY

Pno	*Street*	*Area*	*City*	*Rent*
PA14	16 Holhead	Dee	Aberdeen	650
PL94	6 Argyll St	Kilburn	London	400
PG21	18 Dale Rd	Hyndland	Glasgow	600

RENTER

Rno	*Name*	*Address*
CR76	John Kay	56 High St
CR74	Mike Ritchie	18 Tain St
CR62	Mary Tregear	5 Tarbot Rd



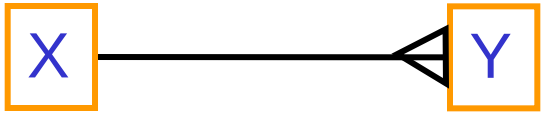
VIEWING

Pno	*Rno*	*Date*	*Time*
PA14	CR74	09/02/97	9:00
PA14	CR76	21/02/97	11:15
PG21	CR74	15/06/97	3:45
PL94	CR62	18/08/97	9:00

- The *Viewing* relation holds foreign keys from each of the entities in the M:M relationship. We also add some extra fields as this happens to be convenient and natural.

Summary: Rules for Relationships - I

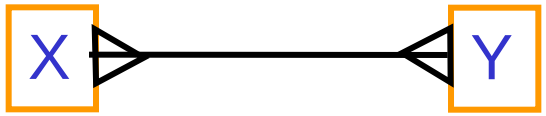
- Consider two relations *X* and *Y* with primary keys *keyx* and *keyy*.
- To create a 1:M relationship from X to Y:



we embed *keyx* (the primary key of the *one* side) as a foreign key in *Y* (the *many* side):

X (*keyx*, …) *Y* (*keyy*, …, *keyx*)

- To create a M:M relationship between *X* and *Y*:



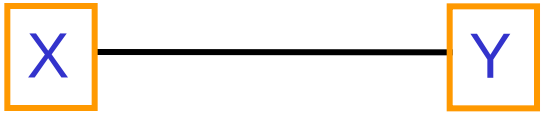
we create a third relation, say *Z*, containing the primary keys from both sides of the relationship:

X (*keyx*, …) *Z* (*keyx*, *keyy*) *Y* (*keyy*, …)

Note that this means creating a whole new relation

Summary: Rules for Relationships - II

- To create a 1:1 relationship from X to Y:



  Where *X* and *Y* are two relations with primary keys *keyx* and *keyy*. There are two ways to create a 1:1 relationship from X to Y.

- Option 1: Use separate tables (Recommended Option for you to follow!)

  A foreign key is used to link the tables. For example, we may insert *keyx* into Table Y, **or** insert *keyy* into Table X

  that is, use: X (*keyx*, …, *keyy*) OR *Y* (*keyy*, …, *keyx*)

- Option 2: Use a single table for both entities
  - we can do this if the relationship is mandatory for at least one of X and Y
  - what is the primary key of the combined table?

ENTITY RELATIONSHIP MODELLING

HOW TO DO IT

1. Start with a system description

2. Deduce entities by searching for nouns or noun-phrases
   - be careful ..... attributes will also appear as noun/noun-phrases.
   - exclude nouns that are qualities of entities.
   - Subjective.
   - Iterative.
   - leave out global entities.

3. Deduce the relationships by searching for phrases such as:
   - HAS
   - IS COMPOSED OF
   - REQUIRE A NUMBER OF
   - ETC...

The Bus Company: Important noun phrases

A bus company owns a **number of buses**. **Each bus** is allocated to **a particular route** although **some routes** may have **several buses**. **Each bus** has a **unique bus number**. It is important to store information about **the seating capacity** and the **make/type** of **all buses**. **Each route** , distinguished by **a route number**, passes through **a number of towns**. Information is available on **the average number of passengers carried per day for each route**.

One or more **drivers** are allocated to **each stage** of a **route**, which corresponds to **a journey** through some or all of the **towns** on **a route**. **Drivers** have an **employee number**, **a name, address** and **sometimes a telephone number**.

Entities

BUS: The buses owned by the company

ROUTE: The routes served by the company

STAGE: The sections of the routes

DRIVER: The drivers employed by the company

TOWN: The towns through which the stages pass

The Bus Company: Potential relationship pointers.

A bus company owns a number of buses. Each bus **is allocated to** a particular route although some routes may **have** several buses. Each bus **has** a unique bus number. It is important to store information about the seating capacity and the make/type of all buses. Each route , distinguished by a route number, **passes through a number of** towns. Information **is available on** the average number of passengers carried per day for each route.

One or more drivers **are allocated to** each stage of a route, which corresponds to a journey through **some or all of** the towns on a route. Drivers **have an** employee number, a name, address and sometimes a telephone number.

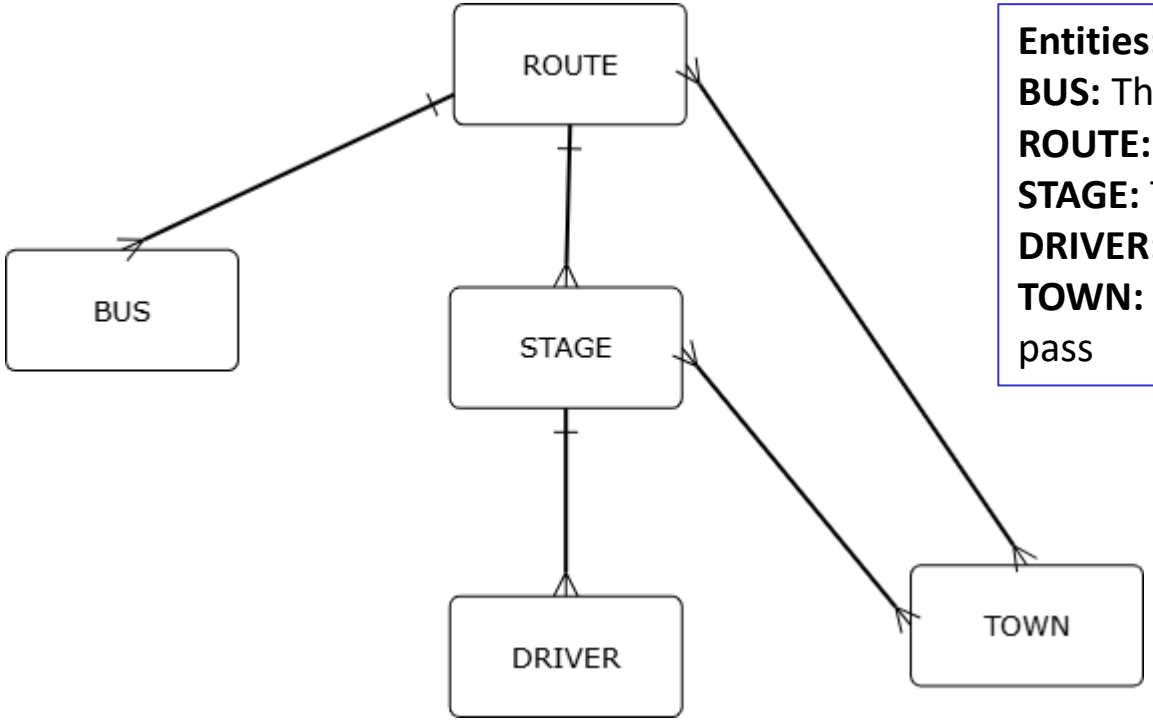
Relationships

Relationship name	Entities	Degree	Optionality
ALLOCATED TO	Route, Bus	1:N	oblig (?)

Relationships

Relationship Name	Entities	Degree	Optionality
ALLOCATED TO	Route, Bus	1:N	oblig (?)
PASSES THROUGH 1	Route, Town	M:N	oblig (?)
PASSES THROUGH 2	Stage, Town	M:N	oblig(?)
DRIVER ALLOCATED	TO Stage, Driver	1:N	oblig (?)
ROUTE HAS STAGE	Route, Stage	1:N	oblig(?)

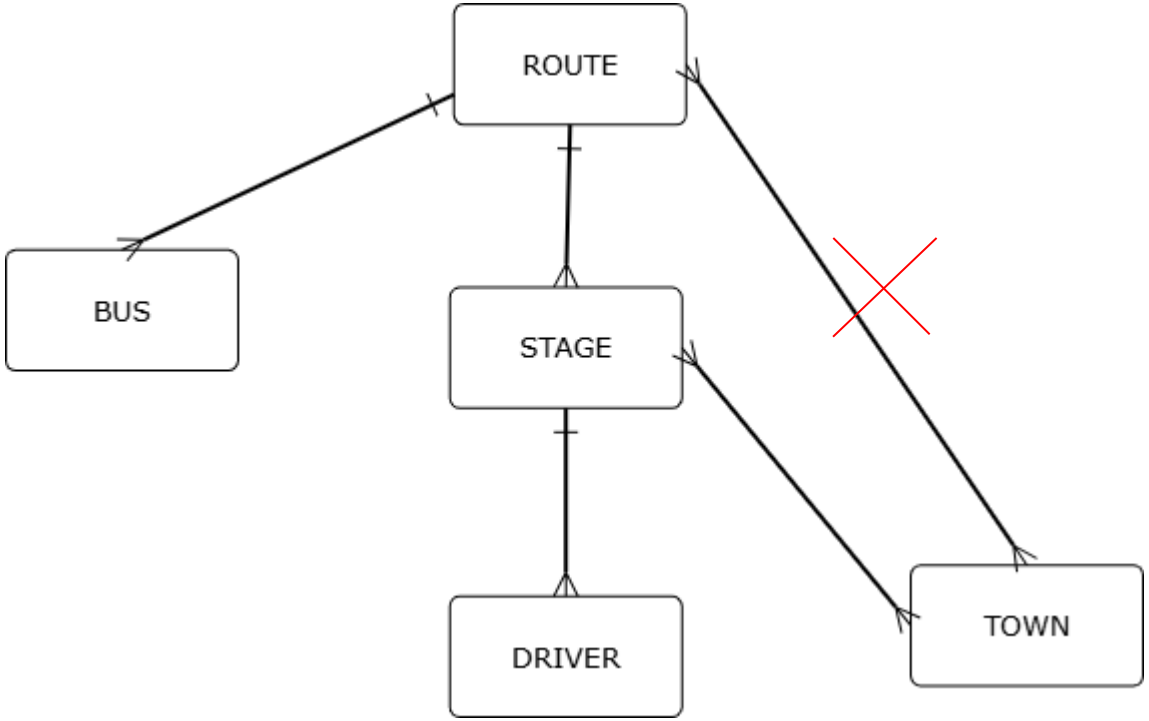
Constructing an entity-relationship diagram to show corresponding entities and their relationships.



Entities:
BUS: The busses owned by the company
ROUTE: The routes served by the company
STAGE: The sections of the routes
DRIVER: The drivers employed by the company
TOWN: The towns through which the stages pass

Relationship Name	Entities	Degree	Optionality
ALLOCATED TO	Route, Bus	1:N	oblig (?)
PASSES THROUGH 1	Route, Town	M:N	oblig (?)
PASSES THROUGH 2	Stage, Town	M:N	oblig(?)
DRIVER ALLOCATED	TO Stage, Driver	1:N	oblig (?)
ROUTE HAS STAGE	Route, Stage	1:N	oblig(?)

Continued...



Revise:

- check each pair of entities for a possible relationship (!)
- look for non-obligatory relationships
- look for redundant relationships

Continued...

Deduce the attributes for the entities provided and add surrogates. (A surrogate is an artificial identifier).

BUS Bus Number, Capacity, Make, Type

ROUTE Route Number, Ave passengers

STAGE StageID, Stagename

DRIVER EmpID, Name, Address, Telephone No.

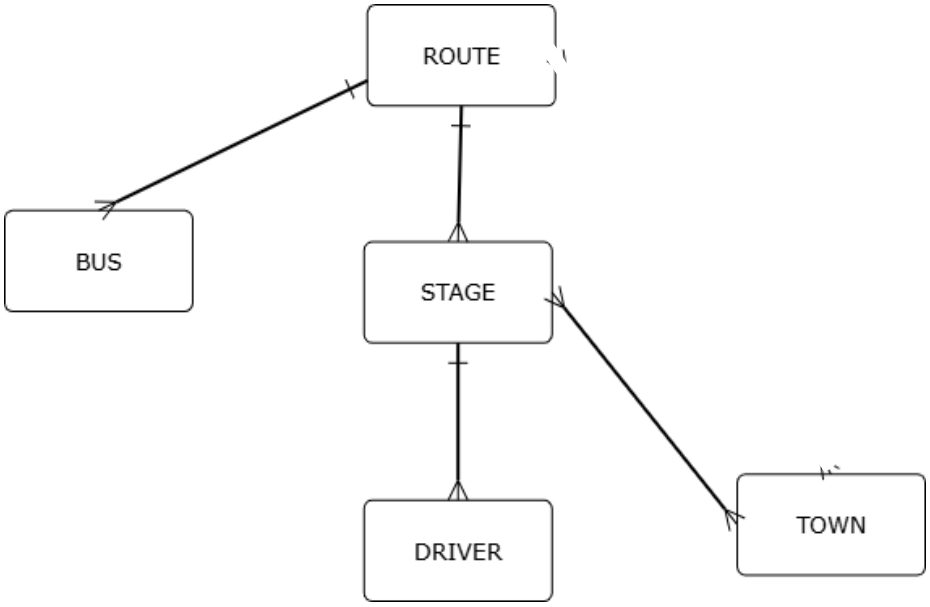
TOWN TownID, Name

NO foreign keys at this stage…..

Final ERD

Entities:
BUS: The busses owned by the company
ROUTE: The routes served by the company
STAGE: The sections of the routes
DRIVER: The drivers employed by the company
TOWN: The towns through which the stages pass

Relationship Name	Entities	Degree	Optionality
ALLOCATED TO	Route, Bus	1:N	oblig (?)
~~PASSES THROUGH 1~~	~~Route, Town~~	~~M:N~~	~~oblig (?)~~
PASSES THROUGH 2	Stage, Town	M:N*	oblig(?)
DRIVER ALLOCATED	TO Stage, Driver	1:N	oblig (?)
ROUTE HAS STAGE	Route, Stage	1:N	oblig(?)



BUS	Bus Number, Capacity, Make, Type
ROUTE	Route Number, Ave passengers
STAGE	StageID, Stagename
DRIVER	EmpID, Name, Address, Telephone No.
TOWN	TownID, Name

Assumptions: All relationships are obligatory
* This needs to be settled in the next lectures.

Tour

smartdraw®

https://www.smartdraw.com/entity-relationship-diagram/

Course Content

1. Introduction to Relational Databases *(Introduction + Relational Model)*
2. Data Modelling - *(Entity Relationship Modelling + The Enhanced Entity Relationship Model)*
3. Database Design and SQL - *(Logical modelling + Introduction to SQL)*
4. Further SQL - *(Advanced SQL queries + Creating tables with SQL)*
5. Normalisation - *(Normalisation to second normal form + Third normal form)*

Questions

THE PLACE OF USEFUL LEARNING

University of Strathclyde Glasgow

Database Fundamentals – CS990

Database and Web Systems Development - CS952

Image source: https://www.strath.ac.uk/branding/