Data Analysis 3

SET08104 Database Systems

Mapping ER Models into Relations

Overview

- map 1:1 relationships into relations
- map 1:m relationships into relations
- map m:n relationships into relations
- differences between mapping optional and mandatory relationships.

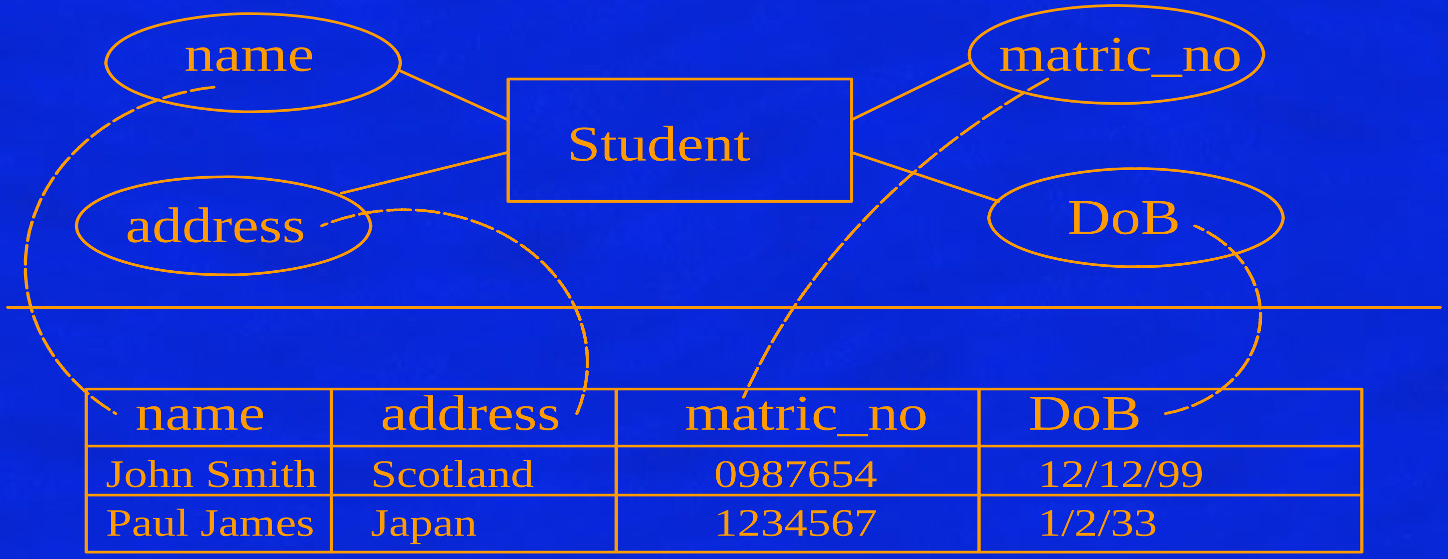
What is a relation?

A relation is a table that holds the data we are interested in. It is two-dimensional and has rows and columns.

Each entity type in the ER model is mapped into a relation.

- The attributes become the columns.
- The individual entities become the rows.

What is a relation cont...



name	address	matric_no	DoB
John Smith	Scotland	0987654	12/12/99
Paul James	Japan	1234567	1/2/33

What is a relation cont...

Relations can be represented textually as:

tablename(<u>primary key</u>, attribute 1, attribute 2, ... , *foreign key*)

If matric_no was the primary key, and there were no foreign keys, then the table above could be represented as:

student(<u>matric no</u>, name, address, date_of_birth)

Foreign keys

A foreign key is an attribute (or group of attributes) that is the primary key to another relation.

- Roughly, each foreign key represents a relationship between two entity types.
- They are added to relations as we go through the mapping process.
- They allow the relations to be linked together.
- A relation can have several foreign keys.
- It will generally have a foreign key from each table that it is related to.
- Foreign keys are usually shown in italics or with a wiggly underline.

Preparing to map the ER model

Before we start the actual mapping process we need to be certain that we have simplified the ER model as much as possible.

This is the ideal time to check the model, as it is really the last chance to make changes to the ER model without causing major complications.

Mapping 1:1 relationships

Before tackling a 1:1 relationship, we need to know its optionality.

There are three possibilities the relationship can be:

1. mandatory at both ends
2. mandatory at one end and optional at the other
3. optional at both ends

Mandatory at both ends

If the relationship is mandatory at both ends it is often possible to subsume one entity type into the other.

- The choice of which entity type subsumes the other depends on which is the most important entity type (more attributes, better key, semantic nature of them).

- The key of the subsumed entity type becomes a normal attribute.
- If there are any attributes in common, the duplicates are removed.
- The primary key of the new combined entity is usually the same as that of the original more important entity type.

When not to combine

There are a few reason why you might not combine a 1:1 mandatory relationship.

- the two entity types represent different entities in the ‘real world’.
- the entities participate in very different relationships with other entities.
- efficiency considerations when fast responses are required or different patterns of updating occur to the two different entity types.

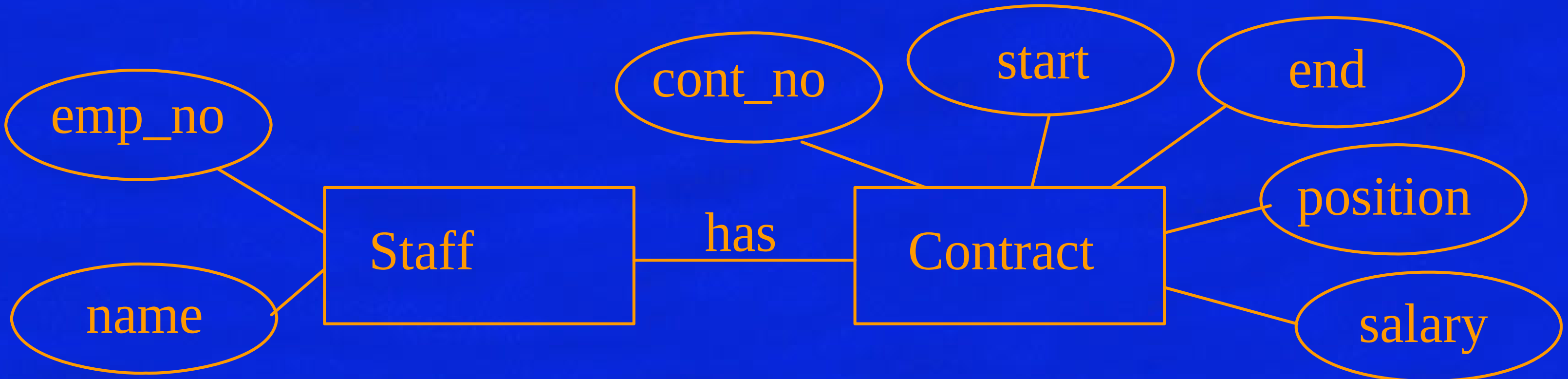
If not combined...

If the two entity types are kept separate then the association between them must be represented by a foreign key.

- The primary key of one entity type becomes the foreign key in the other.
- It does not matter which way around it is done but you should not have a foreign key in each entity type.

Example

- Two entity types; staff and contract.
  - Each member of staff must have one contract and each contract must have one member of staff associated with it.
  - It is therefore a mandatory relations at both ends.



Example cont...

- These two entity types could be amalgamated into one.

  Staff(emp_no, name, cont_no, start, end, position, salary)
- or kept apart and a foreign key used

  Staff(emp_no, name, *contract_no*)

  Contract(cont_no, start, end, position, salary)
- or

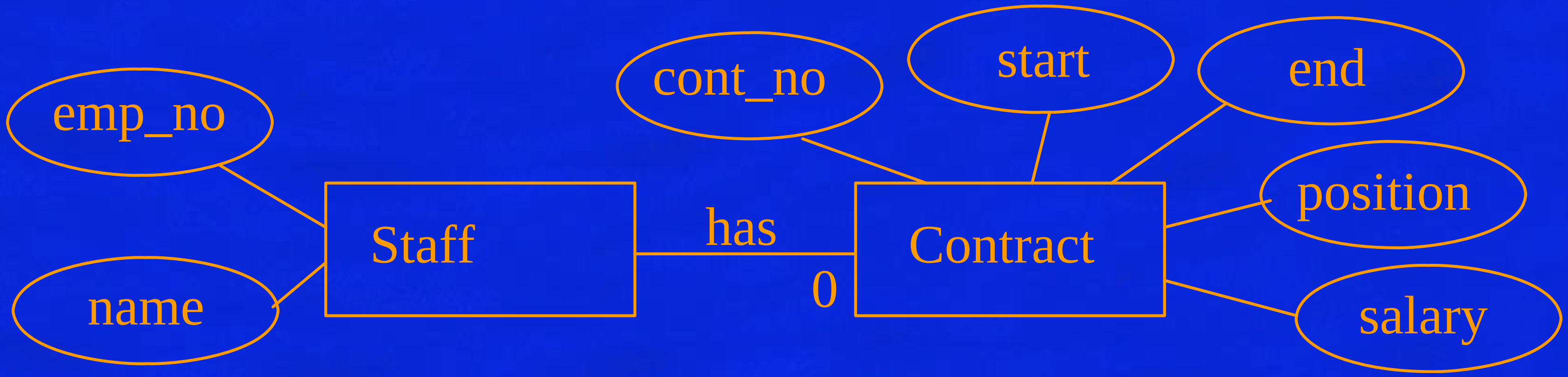
  Staff(emp_no, name)

  Contract(cont_no, start, end, position, salary, *emp_no*)

Mandatory <-> Optional

For 1:1 optional relationships, take the primary key from the ‘mandatory end’ and add it to the ‘optional end’ as a foreign key.

If it was done the other way around, it would create null entries.



Mandatory <-> Optional cont...

If we add to the specification that each staff member may have at most one contract (thus making the relation optional at one end).

- Mapping the foreign key into Staff – would create null entries for staff without a contract.

  Staff(<u>emp_no</u>, name, *contract_no*)

  Contract(<u>cont_no</u>, start, end, position, salary)

- Map the foreign key into Contract - emp_no is mandatory thus never null.

  Staff(<u>emp_no</u>, name)

  Contract(<u>cont_no</u>, start, end, position, salary, *emp_no*)

Example

Consider this example:

- Staff “Gordon”, empno 10, contract no 11.
- Staff “Andrew”, empno 11, no contract.
- Contract 11, from 1st Jan 2001 to 10th Jan 2001, lecturer, on £2.00 a year.

FK in STAFF

Staff

Empno	Name	Contract No
10	Gordon	11
11	Andrew	**NULL**

Contract

Cont_no	Start	End	Position	Salary
11	1st Jan 2001	10th Jan 2001	Lecturer	£2.00

FK in CONTRACT

Staff

Empno	Name
10	Gordon
11	Andrew

Contract

Cont_no	Start	End	Position	Salary	Staff
11	1st Jan 2001	10th Jan 2001	Lecturer	£2.00	10

Summary...

So for 1:1 optional relationships, take the primary key from the ‘mandatory end’ and add it to the ‘optional end’ as a foreign key.

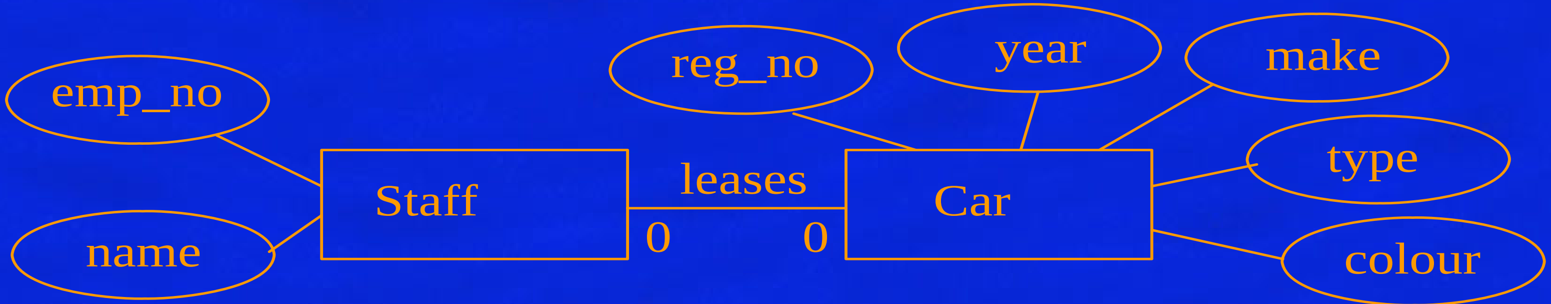
So, given entity types A and B, where A <-> B is a relationship where the A end it optional, the result would be:

A (<u>primary key</u>,attribute,...,*foreign key to B*)

B (<u>primary key</u>,attribute,...)

Optional at both ends...

- Such examples cannot be amalgamated into one relation because you could not select a primary key.
- Instead, one foreign key is used as before.



Optional at both ends cont...

- Each staff member may lease up to one car
- Each car may be leased by at most one member of staff
- If these were combined together...

  Staff_car(emp_no, name, reg_no, year, make, type, colour)

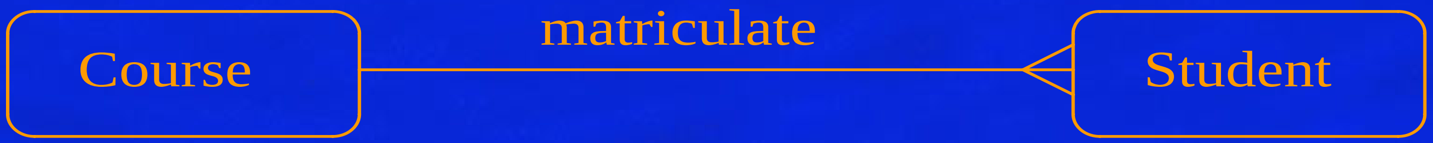
what would be the primary key?

- If emp_no is used then all the cars which are not being leased will not have a key.
- Similarly, if the reg_no is used, all the staff not leasing a car will not have a key.
- A compound key will not work either.

Mapping 1:m relationships

To map 1:m relationships, the primary key on the ‘one side’ of the relationship is added to the ‘many side’ as a foreign key.

For example, the 1:m relationship ‘course-student’:



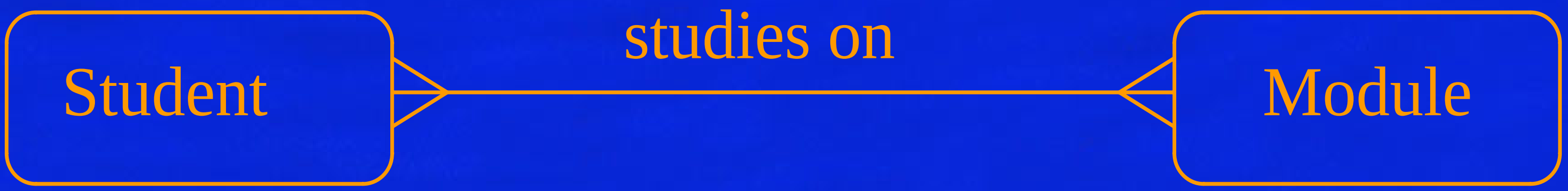
Mapping 1:m relationships cont...

- Assuming that the entity types have the following attributes:
  Course(course_no, c_name)
  Student(matric_no, st_name, dob)
- Then after mapping, the following relations are produced:
  Course(<u>course_no</u>, c_name)
  Student(<u>matric_no</u>, st_name, dob, *course_no*)
- If an entity type participates in several 1:m relationships, then you apply the rule to each relationship, and add foreign keys as appropriate.

Mapping n:m relationships

If you have some m:n relationships in your ER model then these are mapped in the following manner.

- A new relation is produced which contains the primary keys from both sides of the relationship
- These primary keys form a composite primary key.



Mapping m:n relationships cont...

- Thus

  Student(matric_no, st_name, dob)

  Module(module_no, m_name, level, credits)

- becomes

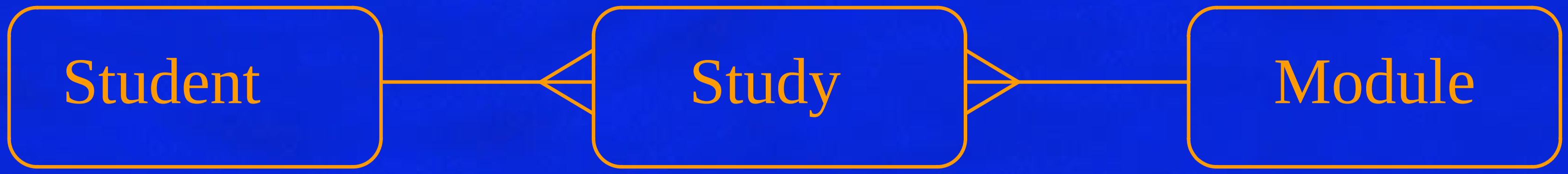
  Student(<u>matric_no</u>, st_name, dob)

  Module(<u>module_no</u>, m_name, level, credits)

  Studies(<u>*matric_no,module_no*</u>)

Mapping m:n relationships cont…

This is equivalent to:



Student(matric_no,st_name,dob)

Module(module_no,m_name,level,credits)

Study(*matric_no,module_no*)

Summary

- 1-1 relationships
  Depending on the optionality of the relationship, the entities are either combined or the primary key of one entity type is placed as a foreign key in the other relation.
- 1-m relationships
  The primary key from the ‘one side’ is placed as a foreign key in the ‘many side’.
- m-n relationships
  A new relation is created with the primary keys from each entity forming a composite key.