Preliminary Database Design

- **Translating E-R Diagrams into Preliminary Database Designs**

  Data models represented as E-R diagrams can be directly translated into preliminary database designs.

  The database design consists primarily of tables, columns and keys.

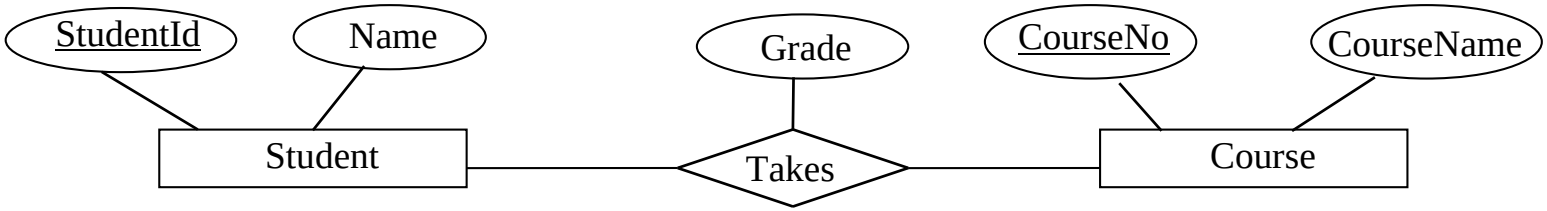
  1. **Each entity set with its attributes is directly translated into a table with the attributes as columns.**

  2. **Each relationship is also translated into a table and its columns will be composed of the attributes of the relationship as well as the key attributes inherited from the related entities.**

  3. **Each dependent entity is also translated into a table and its columns will be composed of the attributes of the entity as well as the key attributes inherited from the corresponding strong entity.**

  The attributes that are primary keys are shown underlined. The inherited keys are not shown in the E-R diagram, but are shown in the translated preliminary database design.

  Example: Data model of a Student-Course information system



Preliminary Database Design (Cont.)

The E-R diagram above can be translated into a corresponding preliminary database design as follows. Note the keys inherited by the relationship Takes are shown in the table defined for Takes below.

Table: Student

StudentID	Name

Table: Takes

StudentID	CourseNo	Grade

Table: Course

CourseNo	CourseName

A second alternative for representing the preliminary database design for the same example is:

Student (StudentID, Name)
Takes (StudentID, CourseNo, Grade)
Course (CourseNo, CourseName)

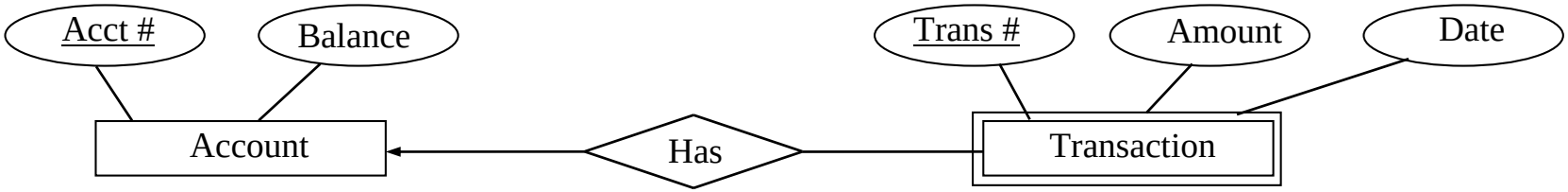
A third alternative for representing the preliminary database design (PK stands for primary key and FK stands for Foreign Key, that is, a key inherited from a related entity) shows data types of attributes:

Student	Data Type	Takes	Data Type	Course	Data Type
StudentID	Integer (PK)	StudentID	Integer (FK)	CourseNo	Text 8 (PK)
Name	Text 30	CourseNo	Text 8 (FK)	CourseName	Text 30
		Grade	Text 2		

NOTE: Only one of the above three representations is necessary to show the preliminary database design. Do not show all three alternatives unnecessarily in the homework, in-class exercise or exam. The third alternative is generally preferred in design documents when the data types are known.

Preliminary Database Design (Cont.)

Example: Data model of an Accounts-Transaction information system



The translated preliminary database design for the above ER diagram is as follows:

Table: Account

AccountNo	**Balance**

Table: Has

AccountNo	**Trans#**

Table: Transaction

AccountNo	**Trans#**	**Amount**	**Date**

Alternative 2 for the same example:

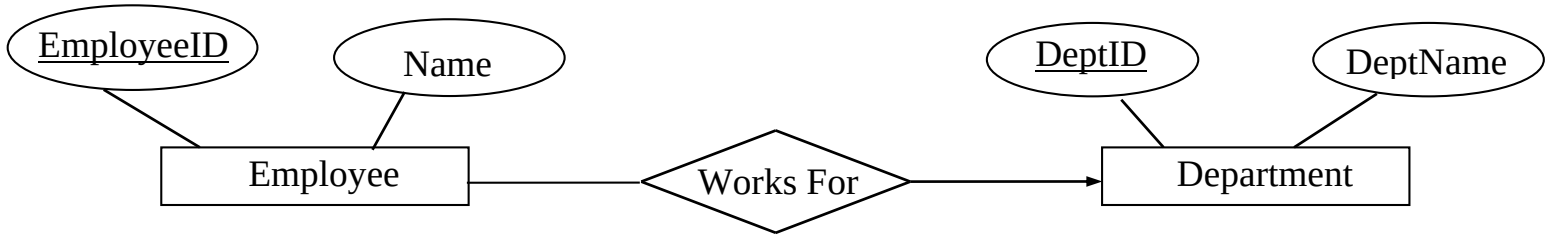
Account (AccountNo, Balance)
Has (AccountNo, Trans#)
Transaction (AccountNo, Trans#, Amount, Date)

Preliminary Database Design (Cont.)

Alternative 3 for the same example:

Account		**Has**		**Transactions**	
AccountNo	**Integer (PK)**	**AccountNo**	**Integer (FK)**	**AccountNo**	**Integer (PK)**
Balance	**Currency**	**Trans#**	**Integer (FK)**	**Trans#**	**Integer (PK)**
				Amount	**Currency**
				Date	**Date**

Example: Data model of an Employee-Department information system



The E-R diagram can be translated into a preliminary database design as follows:

Table: Employee

EmployeeID	**Name**

Table: Works For

EmployeeID	**DeptID**

Table: Department

DeptID	**DeptName**

Preliminary Database Design (Cont.)

A second alternative for representing the same preliminary database design:

Employee (EmployeeID, Name)
Work For (EmployeeID, DeptID)
Department (DeptID, DeptName)

A third alternative for representing the same preliminary database design:

Employee		Works For		Department	
EmployeeID	Integer (PK)	EmployeeID	Integer (FK)	DeptID	Integer (PK)
Name	Text 30	DeptID	Integer (FK)	DeptName	Text 20

- **There is a third situation where foreign keys are necessary, and that involves classes and subclasses of entity sets**

- **One of these is called Generalization in E-R modeling terms.**

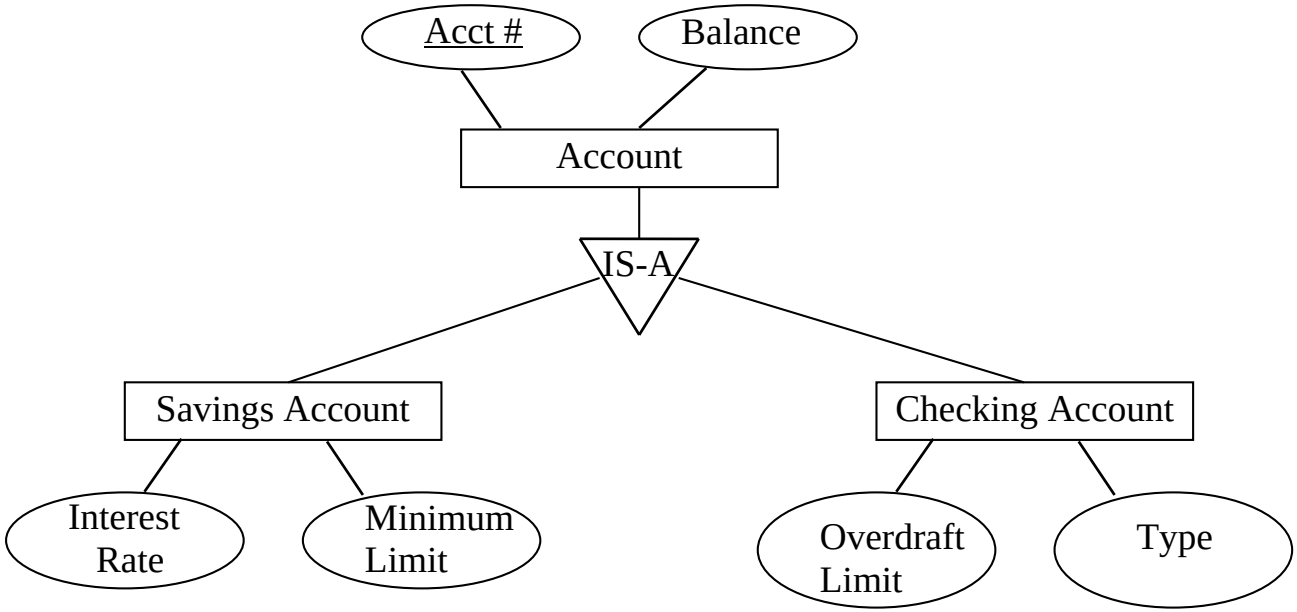
  Each and every entity within a generalization hierarchy, regardless of level, shall inherit the primary key of the root generic parent entity.

  The subclasses that fall under the same class of entity are related by the IS-A relationship with the parent class entity set. "IS-A" is a special type of relationship that relates classes and subclasses.

Preliminary Database Design (Cont.)

Examples of Generalization:

Savings-Account IS-An Account
Checking-Account IS-An Account



- **From the sense of modeling we can model these as shown above to hide the differences at a lower level.**
- **But when the entities are translated into a database design, lower level entity sets (subclass) should inherit the primary key "Account#" from the parent class entity set Account.**
- **It is important to note that the "IS-A" relationship is not translated into a table.**

Preliminary Database Design (Cont.)

- **There are two ways to translate generalization situations into a preliminary database design.**

  1. **Create three tables as shown below:**

     ACCOUNT(Account #, Balance)

     SAVINGS-ACCOUNT(Account #, Interest Rate, Minimum Limit)

     CHECKING-ACCOUNT(Account #, Overdraft Amount, Type)

  OR

  2. **Create two tables**

     SAVINGS-ACCOUNT(Account #, Balance, Interest Rate, Minimum Limit)

     CHECKING-ACCOUNT(Account #, Balance, Overdraft Amount, Type)

  The selection of an alternative for translating a generalization situation into a preliminary database design will depend on other considerations such as types of queries that may have to be answered, etc.

- **There is another E-R modeling situation called Aggregation**

  Aggregation allows expressing relationships among relationships.

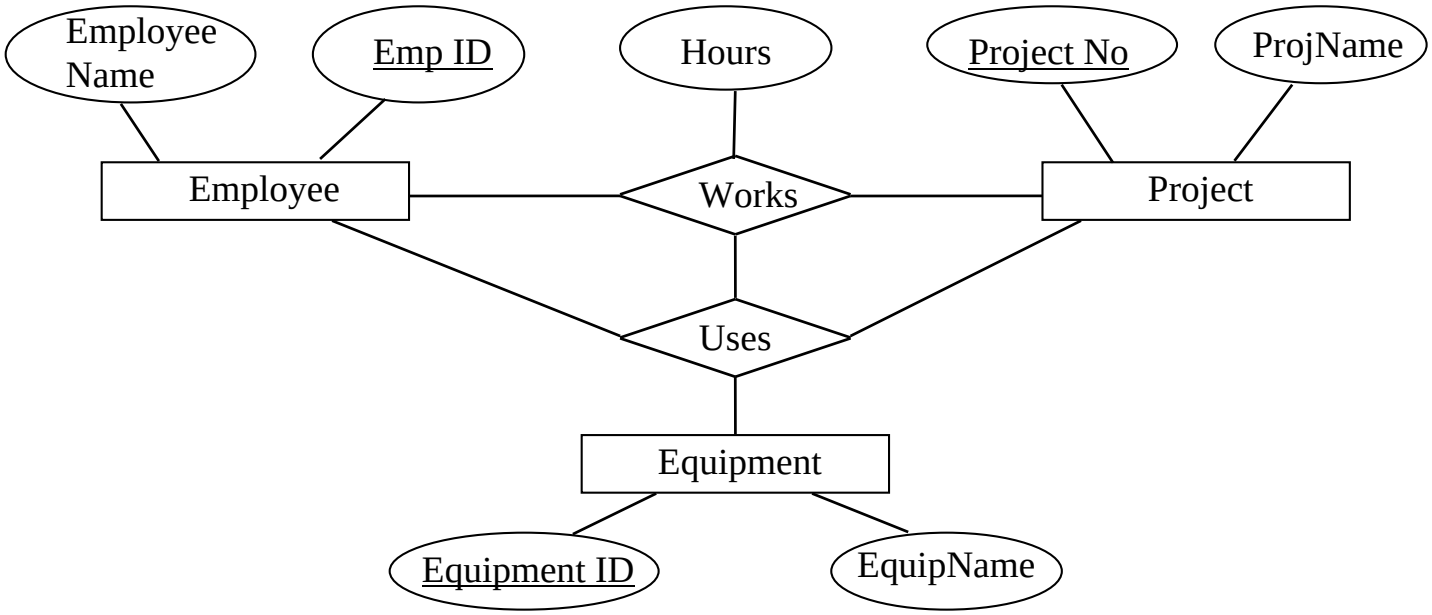
  Aggregation is an abstraction through which relationships are treated as higher level entities.

Preliminary Database Design (Cont.)

Example: In the situation shown below, an employee can work for a project using equipment.

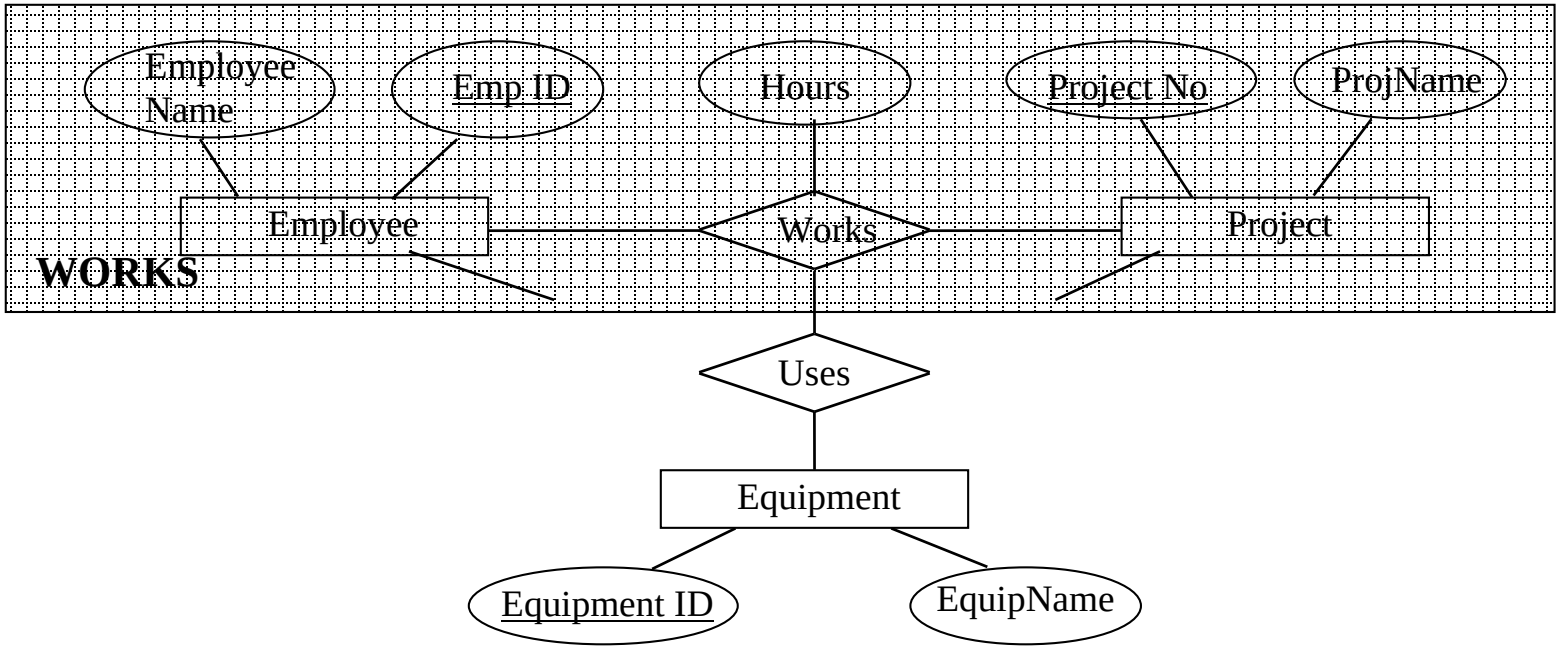
The equipment is used for working on a project, but according to the rules of the ER diagram a relationship between WORKS and USES cannot be created as shown in the ER diagram below.

Therefore, the entity sets EMPLOYEE and PROJECT and their relationship WORKS can be aggregated into a higher-level entity set called WORKS and relate it to the Equipment entity set.



Preliminary Database Design (Cont.)

The modified ER diagram showing the aggregated entity set WORKS is shown below:



This ER diagram can now be translated into a preliminary database design as follows:

Employee(<u>EmpID</u>, EmployeeName)
Project(<u>ProjectNo</u>, ProjName)
Equipment(<u>EquipmentID</u>, EquipName)
Uses(<u>EquipmentId</u>, <u>EmpId</u>, <u>ProjectNo</u>)
Works(<u>EmpID</u>, <u>ProjNo</u>, Hours)

Relational Database Design

- **Translation of ER diagrams so far into preliminary database designs focused mainly on the relational database model.**

- **There are also other database models, such as Hierarchical Model, Network Model, Object Oriented Model, and Semantic Model.**

- **We will continue to focus on the Relational Data Model in this course.**

- **But before finalizing the database design, it is important to understand the relational data model fully.**

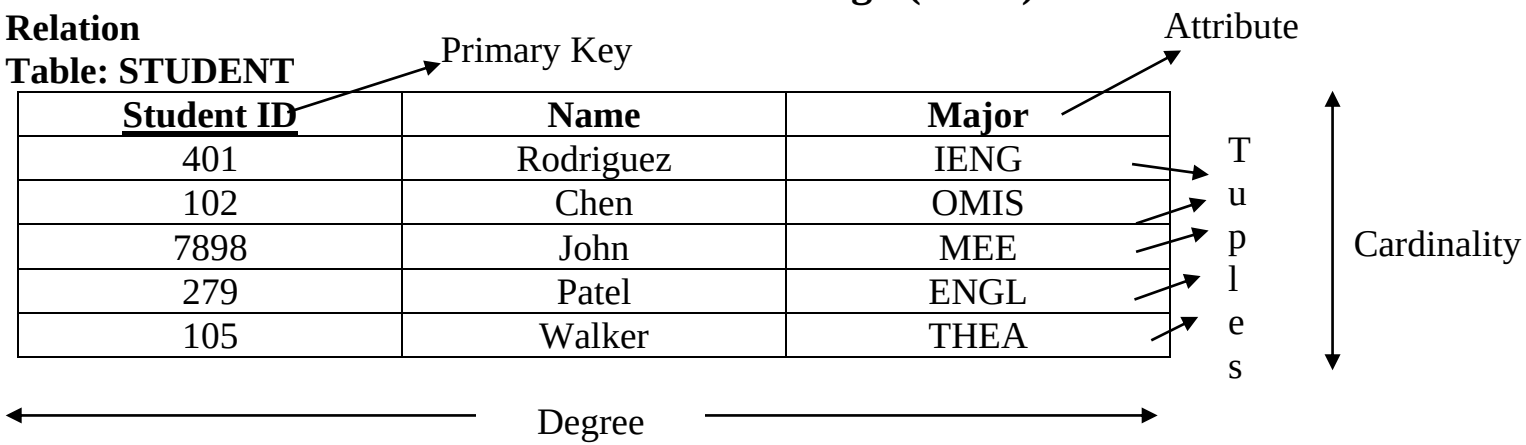
- **The definition of a Relational Data Model:**

  The relational data model was introduced by C. J. Date

Terminology in Relational Data Model	**Informal Equivalent**
Relation	Table
Tuple	Row of a table
Attribute	Column
Cardinality	No. of tuples (e.g. one-to-many)
Degree	No. of columns
Primary Key	Unique identifier
Domain	Pool of acceptable values for an attribute

What is missing in the terminology above compared to the E-R diagram?

Relational Database Design (Cont.)



- **A relational data model (and hence a relational database) should conform to the following properties of relations:**

1. **There are no duplicate tuples. That is, there should not be entire duplicate records in a table.**

2. **Tuples are unordered from top to bottom. That is, there is no order to records in a table.**

   The database management software can sort and order the tuples (records) for you.

3. **Attributes are unordered from left to right. That is, there is no order to attributes in a table.**

   You can choose in which order you want to look at the attributes of a table. The database management software can retrieve records with the attributes in any order you want.

4. **All attributes are atomic. That is, each attribute is single-valued and has no no lists or repeating groups. Single-valued in the sense that an attribute can have only one value and not several values stored in its field. No repeating groups in the sense that an attribute should not be repeated in a table, and this will be explained more later.**

Obligatory/Non-Obligatory Analysis

- **So far we learned how to model a problem situation and develop E-R models of the problem**

  We also translated the E-R diagrams into preliminary database design.

  Do we have to translate all entities and relationships into tables or is it sufficient to translate only some of them? How do we decide which entities or relations to translate and which to leave out?

  How can we go about doing this systematically?

  What is the purpose of combining and breaking apart tables?

- **We can perform something called an obligatory/non-obligatory analysis for this purpose**

  Obligatory – something that is obligated to be or something that is mandatory.

  Non-obligatory – something that is not mandatory or not required to be

  Obligatory – every occurrence of an entity participates in the relationship between entity sets

  Non-obligatory – every occurrence of an entity need not participate in the relationship

- **An entity's occurrence can be obligatory or non-obligatory in a relationship.**

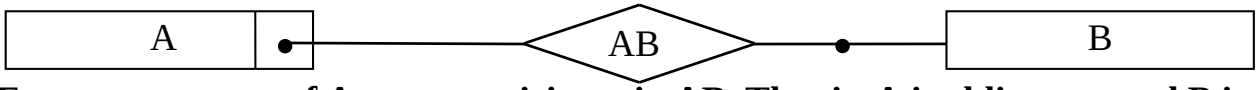
  This may again depend on the problem situation.

Obligatory/Non-Obligatory Analysis (Cont.)

- **Denoting obligatory and non-obligatory entity sets in an E-R diagram**

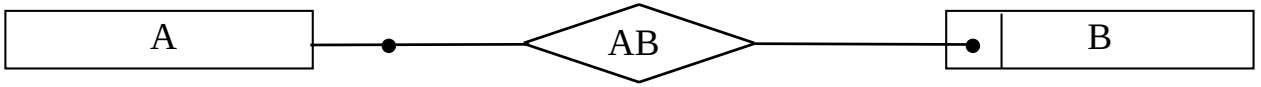
  A blob inside a small rectangle means entity's membership is obligatory. If the blob is outside the entity's rectangle, it means the entity's membership is non-obligatory in the relationship.

Every occurrence of A and B need not participate in the relationship AB. That is, A and B are non-obligatory.



Every occurrence of A must participate in AB. That is, A is obligatory and B is non-obligatory.

Every occurrence of A and B must participate in AB. That is, A and B are obligatory.



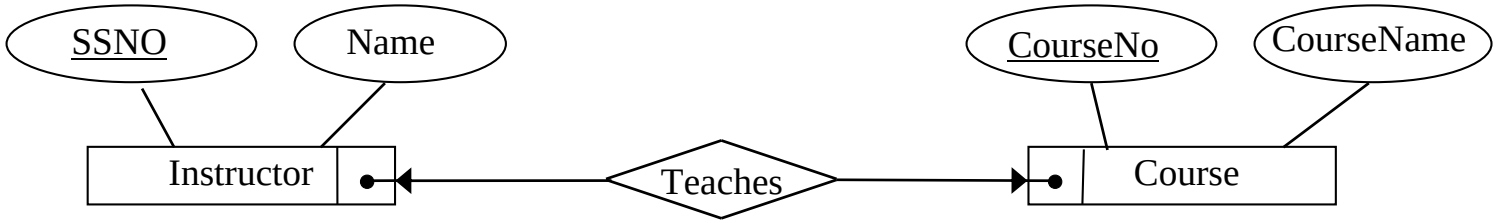
Every occurrence of B must participate in AB. That is, B is obligatory and A is non-obligatory.

Obligatory/Non-Obligatory Analysis (Cont.)

- **How do all these affect table definitions, and do cardinalities have an impact at all on database design?**

- **One to One Relationship: Examples**

  1. **Membership is obligatory for both entity types.**



The assumption here is that each instructor is assigned one course and each course is assigned to only one instructor.

If both entity sets are obligated to participate in the relationship "teaches" and if the cardinality is one-to-one, then what is the minimum number of tables we need?

SSNO	**Name**	**Course No.**	**Course Name**

or

Instructor (SSNO, Name, CourseNo, CourseName)

Each instructor teaches only one course and each course is taught by only one instructor, that is, one cannot exist in the table without the other. Remember that this is a specific example.

Obligatory/Non-Obligatory Analysis (Cont.)

It is assumed that there will be absolutely no other situations in the future such as:

(a) **An instructor may not have a course**

(b) **A course may not have an instructor**

(c) **1: many or many : many relationships**

(d) **Null values for attributes.**

All attributes of both entities can be placed in the same table.

Course attributes can be regarded as attributes of the instructor entity.

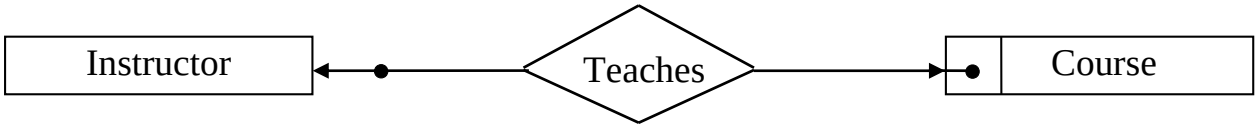
The "teaches" relationship is implicitly represented by the presence of SSNO and CourseNo in the same table and there is no need for a separate table for "teaches". Therefore, one table will do.

Instead of redrawing the E-R diagram to reflect the representation of all the attributes in only one table, we can place an asterisk on the entity and relationship to show they will not be translated into tables.

Note the asterisks in the modified E-R diagram below.

Obligatory/Non-Obligatory Analysis (Cont.)

2. Membership is obligatory for only one entity type.



The assumption in this example is that each course must be taught by one instructor, but not every instructor must teach a course. Note this is only a hypothetical example and is used to illustrate the concepts.

Membership of the entity set "course" in the relationship "teaches" is obligatory and that of the "instructor" is non-obligatory.

So what is the minimum number of tables do we need to represent the entity sets?

Since each course must have an instructor, we can post the primary key of the instructor table in the table for course.

Thus, we can define two tables as follows:

Instructor

SSNO	**Name**

Course

CourseNo	**CourseName**	**SSNO**

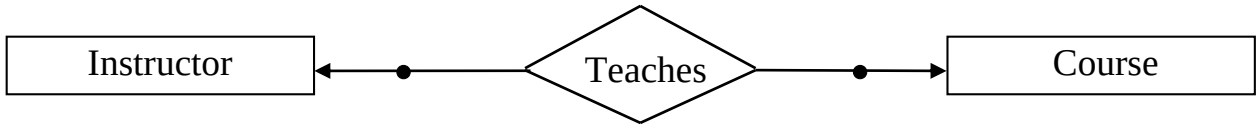
Or Instructor (SSNO, Name) Course (CourseNo, CourseName, SSNO)

Obligatory/Non-Obligatory Analysis (Cont.)

The attribute "SSNO" posted from instructor onto course will form a logical link between the two entity sets.

The absence of a table for "teaches" can be shown in the E-R diagram with an asterisk as before.

3. Membership is Non-obligatory for both entity sets



The assumption in the example is that some instructors need not teach any courses and some courses need not be taught by any instructor.

Membership of both entity sets in the relationship "teaches" is non-obligatory.

Therefore, eliminating a table for "teaches" and posting attribute values onto either table will result in some null values.

As a minimum, we need a table for each entity set and also the relationship, as shown below:

Instructor (SSNO, Name)
Teaches (SSNO, CourseNO)
Course (CourseNo, CourseName)

Obligatory/Non-Obligatory Analysis (Cont.)

Instructor

SSNO	Name

Teaches

SSNO	CourseNo

Course

CourseNo	CourseName

- **One-to-One Relationship Summary:**

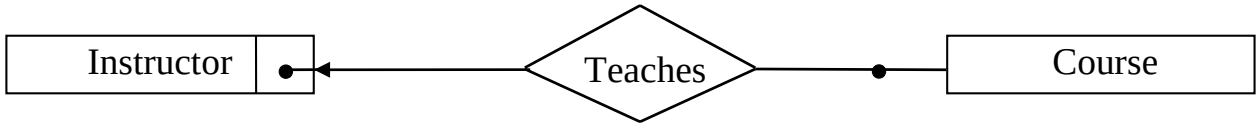
  1. **If membership in the relationship is obligatory for both entity sets, place all attributes of both entity sets into one table. There is no need for a table for the relationship.**

  2. **If membership in the relationship is obligatory for only one entity type, define two tables; one for each entity set. Place the key of the non-obligatory entity set into the obligatory entity's table.**

  3. **If membership is non-obligatory for both entity sets, define three tables; one for each entity set and one for the relationship.**

- **One to Many Relationships: Examples**

  1. **Membership in the relationship is obligatory for the "one" entity set.**



Obligatory/Non-Obligatory Analysis (Cont.)

The assumption in this example is that every instructor must teach many (1 or more) courses, but some courses need not have an instructor.

The "instructor" entity set is obligatory and must participate in the relationship "teaches" and the "course" entity set is non-obligatory and need not participate in the relationship.

What is the minimum number of tables do we need to represent the entity sets?

We need three tables since defining only two tables will result in either null values in the "course" table or repetition of rows in the "instructor" table.

We can define 3 tables as follows:

Instructor (SSNO, Name)
Teaches (SSNO, CourseNO)
Course (CourseNo, CourseName)

or

Instructor

SSNO	Name

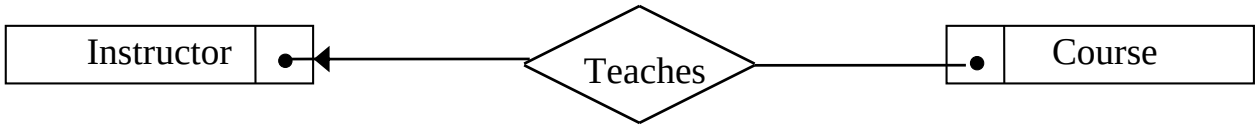
Teaches

SSNO	CourseNo

Course

CourseNo	CourseName

2. **Membership in the relationship is obligatory for both entity sets.**



Obligatory/Non-Obligatory Analysis (Cont.)

We will need 2 tables as shown below in this case since both entity sets are obligatory.

Instructor (SSNO, Name)
Course (CourseNo, CourseName, Section No, SSNO)

or

Instructor

SSNO	Name

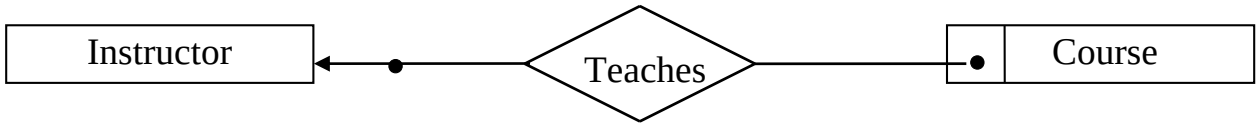
Course

CourseNo	CourseName	SectionNo	SSNO

Since each instructor has to teach a course and each course must be taught by an instructor, we can post the instructor's SSNO on to the course table.

The asterisk must be shown in the relationship in the ER diagram to indicate the relationship need not be translated.

3. Membership in the relationship is obligatory for "many" entity sets



The assumption in this example is that every course section must have an instructor and not every instructor needs to be assigned a course section.

Therefore, the "course" entity set is obligatory and must participate in the relationship teaches.

Obligatory/Non-Obligatory Analysis (Cont.)

3. Membership in the relationship is obligatory for "many" entity sets (Cont.)

Each instructor may or may not teach 1 or more courses, and therefore, the instructor entity set is non-obligatory.

What is the minimum number of tables do we need to represent the entity sets?

The simplest way to represent the "teaches" relationship is to place the primary key of the Instructor entity set onto the Course table.

Each course entity (row) will contain precisely one non-null value of instructor's key since every course must have an instructor.

Instructor (SSNO, Name)
Course (CourseNo, CourseName, Section No, SSNO)

or

Instructor

SSNO	**Name**

Course

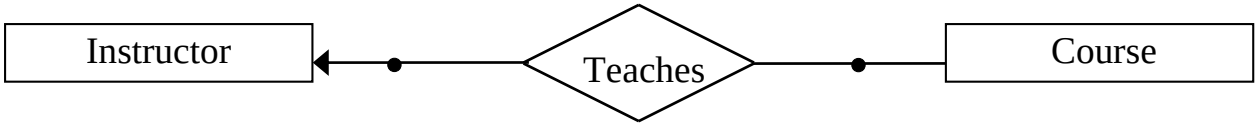
CourseNo	**CourseName**	**SectionNo**	**SSNO**

To indicate there is no need for a table for the relationship "teaches" an asterisk can be placed in the E-R diagram for "teaches".

If we place the "CourseNo" on the instructor table, then entire rows of the instructor table will be repeated, and it is not allowed.

Obligatory/Non-Obligatory Analysis (Cont.)

4. Membership in the relationship is non-obligatory for both entity sets



The assumption this example is that both entity sets need not participate in the relationship "teaches". That is, both entity sets are non-obligatory.

When both entity sets are non-obligatory, we have to define three tables; one for each entity set and one for the relationship.

- **One-to-Many relationship Summary**

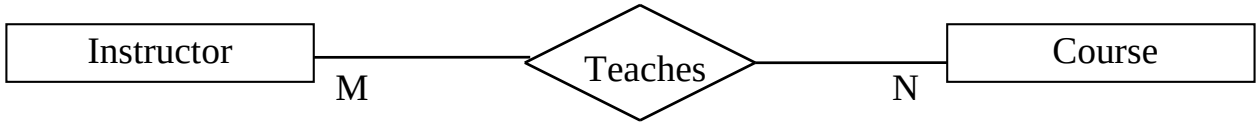
1. **If membership in the relationship is obligatory for the "many" entity set, but obligatory or non-obligatory for the "one" entity set, define 2 tables, one for each entity set and place the key of the "one" entity set into the "many" entity's table.**

2. **If membership in the relationship is obligatory, for the "one" entity set, but not for the "many" entity set, define 3 tables; one for each entity set and one for the relationship.**

3. **If membership is non-obligatory in the relationship for both entity sets, define 3 tables; one for each entity set and one for the relationship.**

Obligatory/Non-Obligatory Analysis (Cont.)

- **Many to Many Relationships: Examples**



The assumption in this case is that each instructor can teach many courses and each course can be taught by many instructors.

Regardless of the membership (obligatory or non-obligatory) of each entity set in the relationship, three tables are needed to represent this case.

Reducing this E-R diagram to just 2 tables will result in either null values or repeating groups. Therefore, 3 tables are needed as follows to represent this design.

Instructor (SSNO, Name)
Teaches (SSNO, CourseNO)
Course (CourseNo, CourseName)

Instructor

SSNO	Name

Teaches

SSNO	CourseNo

Course

CourseNo	CourseName

For a "many to many" relationship between two entity sets, define 3 tables; one for each entity set and one for the relationship, regardless if an entity set is obligatory or non-obligatory.

Obligatory/Non-Obligatory Analysis (Cont.)

- **Summary**

In this module, we looked at translating an ER diagram into a preliminary database design by converting entities and attributes into tables and columns, respectively

We also looked at translating special modeling situations, such as Specialization (class and subclasses related by IS-A relationship) and Aggregation (relationships between relationships)

Note the purpose of special modeling situations is only to capture real-life data modeling situations so that the unique situations are not lost once a data model of the situation has been developed

We learned about the Relational Data Model, its important terminology, and the 4 properties of relations

Next, we looked at the Obligatory/Non-Obligatory analysis, which allows us to evaluate a database design at the entity (table) level and decide on the minimum number of table that may be necessary.

The minimum number of tables necessary depends on the cardinalities of relationships between entities, and if an entity is required to participate in a relationship or not

However, this is only the first step. There is another step called Data Normalization that helps us to validate the database design and ensure that the database design is free of anomalies.

In the Data Normalization step, we will undo what we did in the Obligatory/Non-Obligatory step and analyze the database design at the attribute level and their dependencies, and group the attributes into tables accordingly into much more robust design.

