Individuell uppgift 1, Sammanfattning

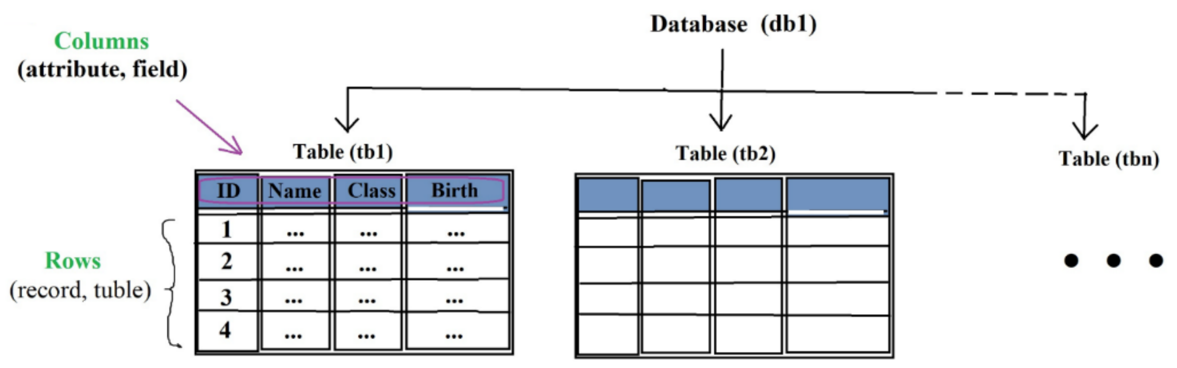
Version 2

Yasir Riyadh Jabbar (TIDAA KTH)

Kurslitteratur: Kapitel 1 (hel), 3 (hel), 4.1 - 4.4, 4.7 - 4.9

CHAPTER 1: Introductory Database Concepts

A database (DB) is an organized data (2D array tables) that is set up for easy access, management and updating. It is controlled by database management system (DBMS) which provides an interface for queries operations and all access to DB passes through it. The one who is responsible for creating, updating and maintaining the operating DB is called database administrator (DBA).



Example of simple University DB:

Here are simple tables that contain information about classes, students, and lecturers which represents university DB.

Student Table

SID	FName	LName	Major	Credits
S101	Noah	Henry	Control	80
S102	Oliver	Jackson	Math	75
S103	James	Daniel	Control	70
S104	William	Michael	Math	85
S105	Lucas	Mason	IT	80

Class Table

ClassNo	FID	Date	Room
MT10A	F102	M9	A20
IT20B	F103	Tu11	A22
CT30A	F101	W9	B24
CT30B	F101	Th12	B25
MT20B	F102	F11	C10

Enroll Table

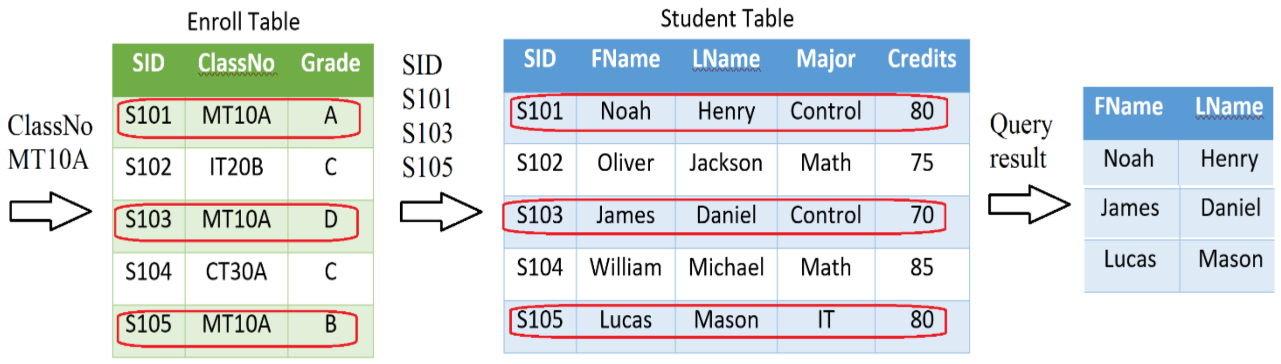
SID	ClassNo	Grade
S101	MT10A	A
S102	IT20B	C
S103	MT10A	D
S104	CT30A	C
S105	MT10A	B

Faculty Table

FID	Name	Dep	Rank
F101	Matthew	Control	Professor
F102	Carter	Math	Assistance
F103	Lincoln	IT	Associate

We notice that every row in Enroll table, represents a relationship between one student and one class.

For example, to find all students who enrolled in class MT10A, we should first search for student's IDs in the Enroll table for this class, then we use these resulting IDs in Student table to extract the related names.



Enroll Table

SID	ClassNo	Grade
S101	MT10A	A
S102	IT20B	C
S103	MT10A	D
S104	CT30A	C
S105	MT10A	B

Student Table

SID	FName	LName	Major	Credits
S101	Noah	Henry	Control	80
S102	Oliver	Jackson	Math	75
S103	James	Daniel	Control	70
S104	William	Michael	Math	85
S105	Lucas	Mason	IT	80

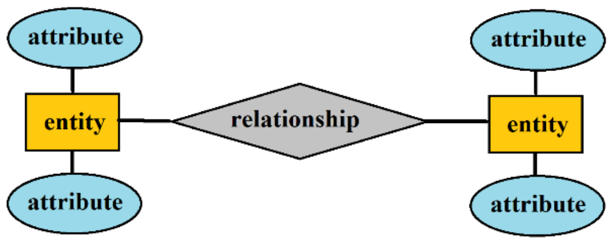
FName	LName
Noah	Henry
James	Daniel
Lucas	Mason

When data is stored for multiple applications, this is called integrated DB and has advantages over separated small DBs: data will be consistent and accurate (data integrity) with high security. Also provides faster data access with easy data recovery and backup.

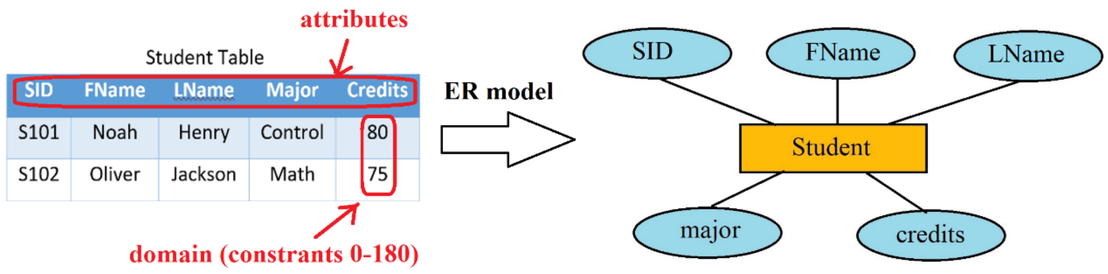
The purpose of data warehouses is to collect data from different DB sources, then for data mining it provides us rich data resources. When wide sources generate huge amounts of data, then this data needs to be capturing, organizing, and analyzing. This type of data is called big data.

CHAPTER 3: The Entity-Relationship Model (ER)

ER model is a diagram that shows the structure of DB. It contains entities, relationships, and attributes.



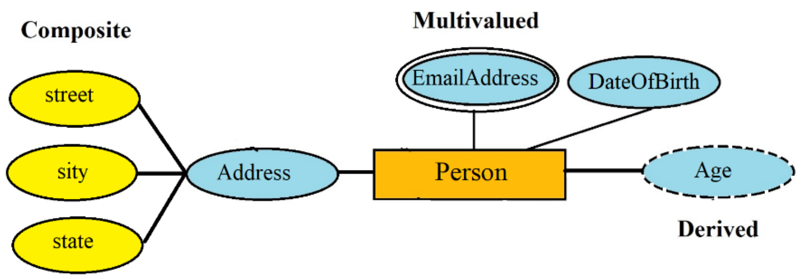
Entity is some unit of data that can be classified or real-world object (e.g., person, place) and has properties which is called attributes. The attribute has values which is called domain. Student table is an example of entity.


Student Table

SID	FName	LName	Major	Credits
S101	Noah	Henry	Control	80
S102	Oliver	Jackson	Math	75

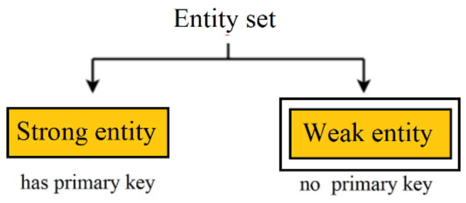
Attribute characteristics:

- if attribute value is unknown at present time, then it has null value.
- attribute may have many values (e.g., one person has many email addresses).
- attribute can be decomposed into sub-attributes (e.g., address includes street, city, state).
- some attributes value can be obtained from other attribute (called derived) (e.g., age attribute can be obtained from date of birth attribute).

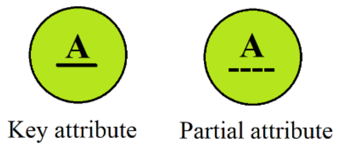


Any attribute can distinctively identify entity is called key. There are different keys:

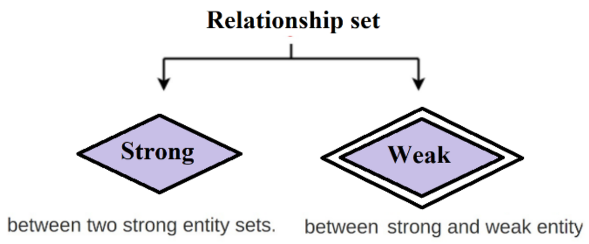
1. **Super Key:** one attribute or more which identify entity. (e.g., SID in Student table). But LName is not because two or more students may share the same last name.
2. **Candidate Key**: combination of attributes can be used to identify entity (e.g., combination of FName, LName, address).
3. **Primary Key**: one of candidate keys can be chosen to uniquely identify the entity. (e.g., SID is primary key for Student table and LName can be used as a secondary key). There is no primary key in Weak entity.



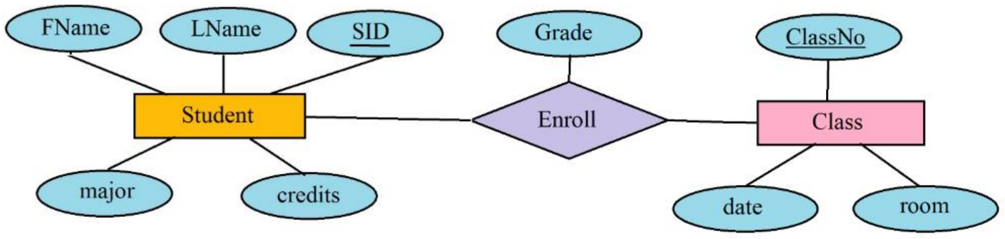
4. **partial key**: It identifies the weak entity but it needs foreign key of the owner entity.



Relationships


Relationship set
Strong
between two strong entity sets.
Weak
between strong and weak entity

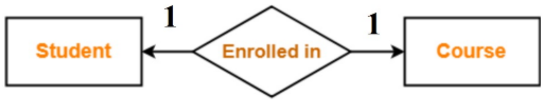
It is the association between tables and could have descriptive attributes. (e.g., Enroll is binary relationship between student and class).

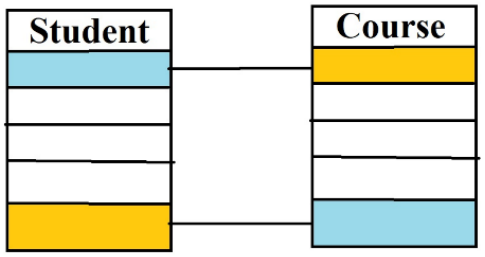


Cardinality of binary relationship:

It is the relation between two different tables by showing how many times one entity occurs comparing to another entity. We have 4 types:

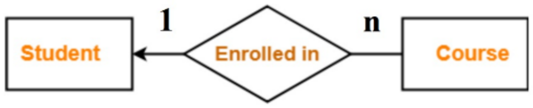
1) one-to- one:

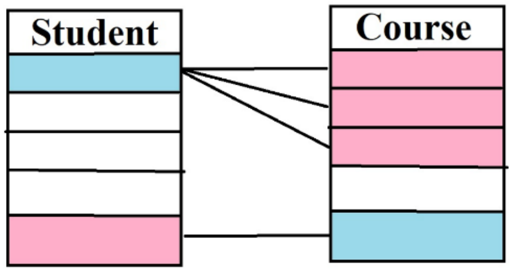




Explanation: 1 student can enroll for maximum 1 course, but 1 course can only be enrolled for maximum 1 student.

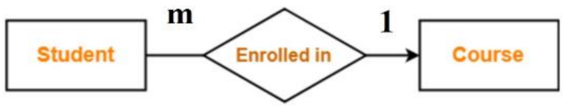
2) one -to-many:

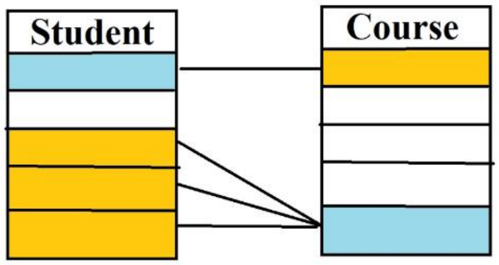




Explanation: 1 student can enroll any number (0 or more) of courses, but 1 course can only be enrolled for maximum 1 student.

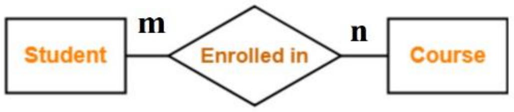
3) Many-to- one:

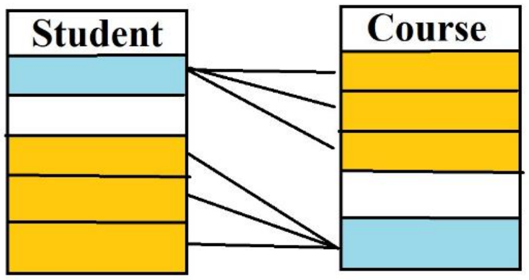




Explanation: 1 student can enroll for maximum 1 course, but 1 course can be enrolled by any number (0 or more) of students.

4) Many-to-many:



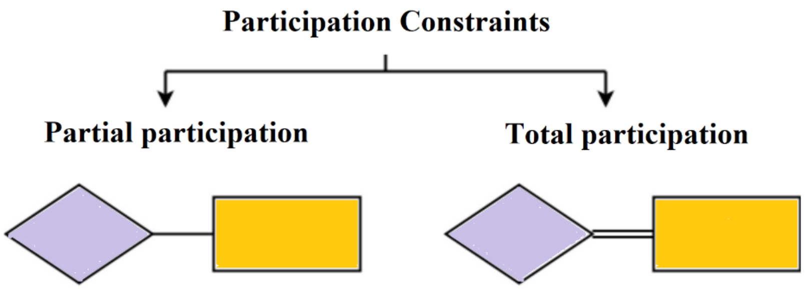


Explanation: 1 student can enroll any number (0 or more) of courses, but 1 course can be enrolled by any number (0 or more) of students.

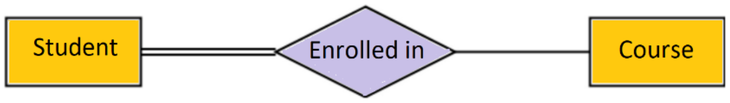
Participation Constraints:

There are 2 types of participation depending on how many members of entity that participate in relationship.

1. Total participation: all members must participate (2 lines)
2. Partial participation: some members may not participate (1 line)



Example:

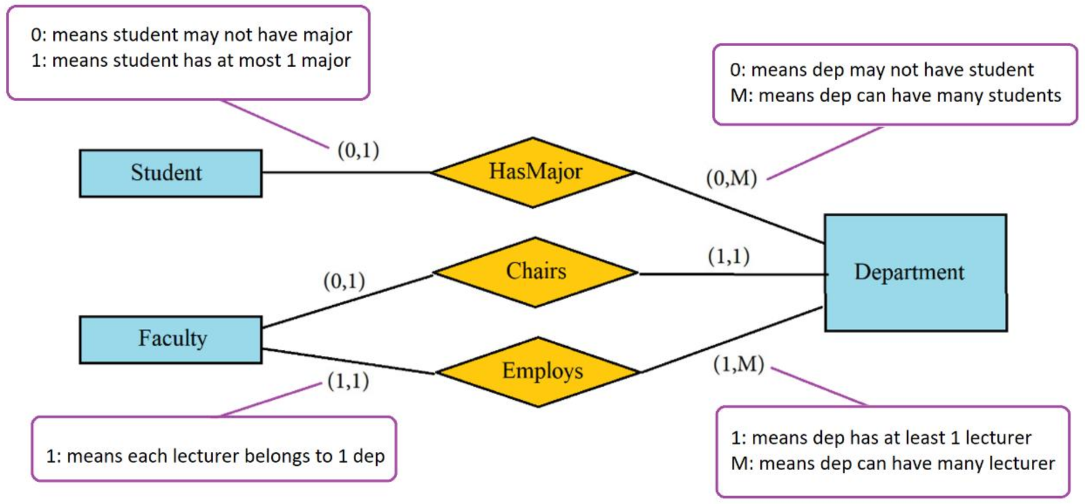


Explanation: each student must attend for minimum 1 course (mandatory), but some courses may be not enrolled.

(min, max) Notation for Cardinality and Participation:

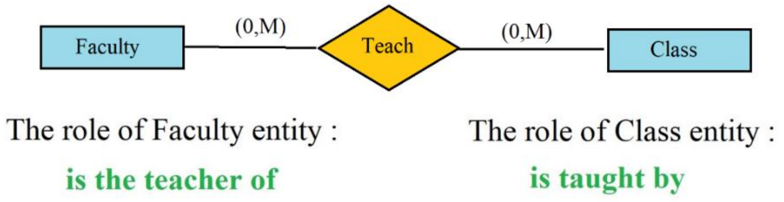
- **min**, specifies minimum number to participate in. 0 for partial, while 1 or more for total participation.
- **max** specifies maximum number (maybe 1, many, or constant integer) to participate in.

Example:


0: means student may not have major
1: means student has at most 1 major
0: means dep may not have student
M: means dep can have many students
(0,1)
Student
HasMajor
(0,M)
Department
(0,1)
Chairs
(1,1)
Faculty
Employs
(1,1)
(1,M)
1: means each lecturer belongs to 1 dep
1: means dep has at least 1 lecturer
M: means dep can have many lecturer

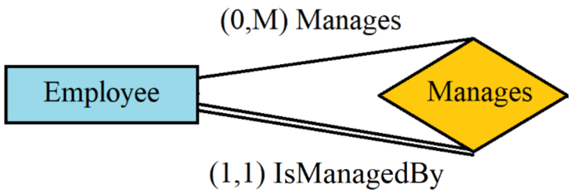
Roles in the relationship:

Each entity that participates in the relationship plays particular role. For example:



When the same entity appears more than once, the relationship is called recursive relationship.

for example:



It is a one -to-many relationship. Every employee must have a manager, but not every manager must have employees.

Sample E-R Diagram (University DB example)

This is simple DB for university that contains information about students, classes, faculty lecturers, and departments that employ the lecturer.

Classifying entities:

Student (SID, LName, FName, major, credits)

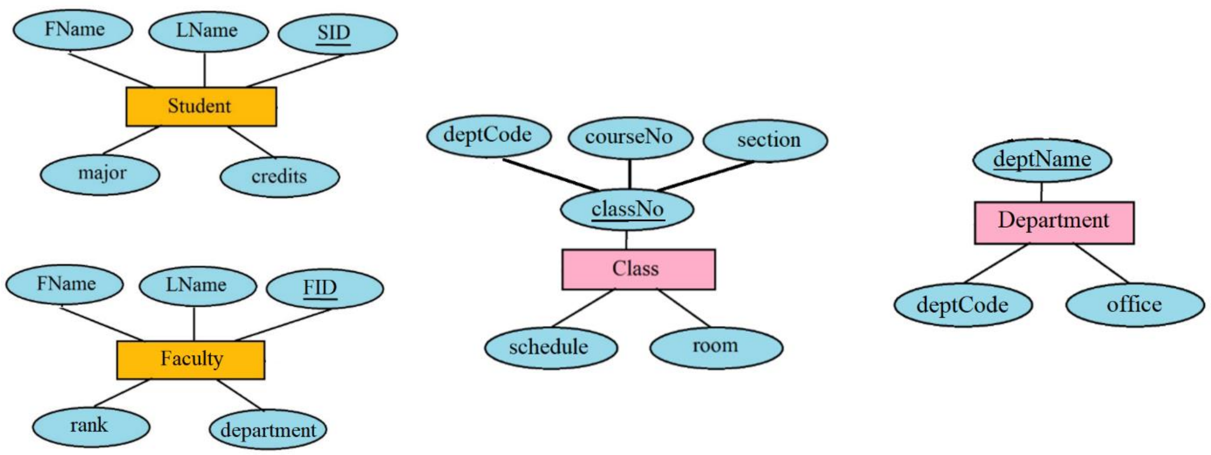
Class (classNo, schedule, room)

Faculty (FID, LName, FName, rank, department)

Department (deptName, deptCode, office)

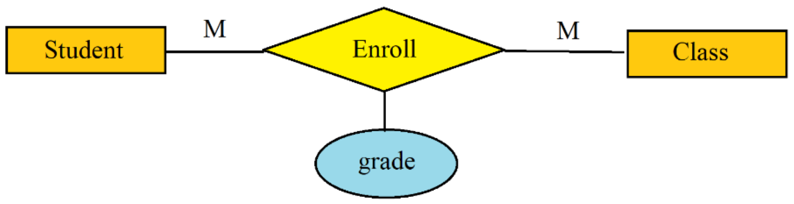
Constraints:

- each student has 1 major.
- classNo consists of dep code, dep number, and section code.
- Each lecturer must belong to 1 department.
- Each lecturer may teach 0 or many classes.
- each department has unique name and must have a chairperson.

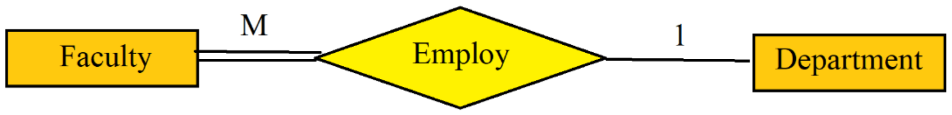


Classifying relationships:

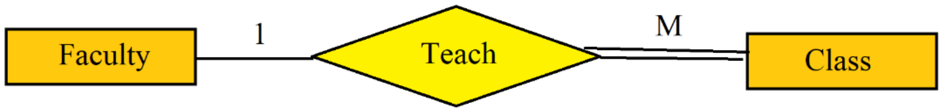
1) **Enroll**: many-to-many relationship (between Student and Class). grade is a descriptive attribute.



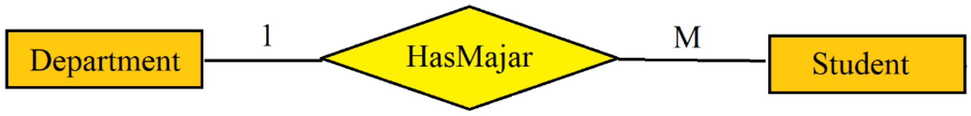
2) **Employ**: one-to-many relationship (between Department and Faculty). Lecturer must belong to one department, but department may or may not have lecturer.



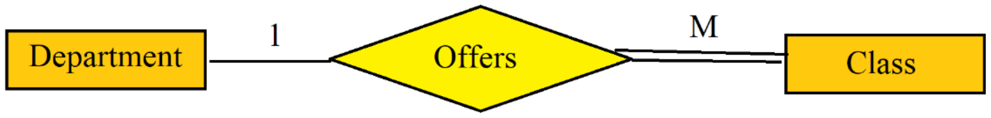
3) **Teach**: one-to-many relationship (between Faculty and Class). Each class has one lecturer.



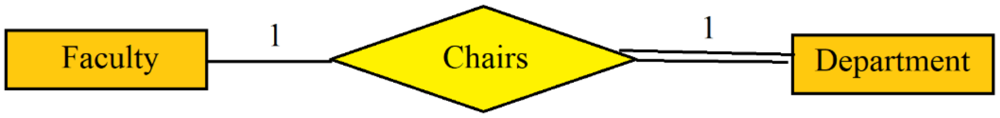
4) **HasMajor**: one -to-many relationship (between Student and Department). Not every department has majors. The values of major are identical to the department name. So, the major attribute became redundant.



5) **Offers**: one-to-many relationship (between Department and Class). Maybe the department has no class to offer, but every class has department that offers it.

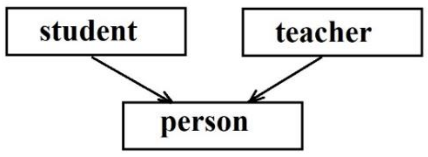


6) **Chairs**: one-to-one relationship (between Department and Faculty). chairperson is one of lecturers, but not every lecturer must be chairperson.

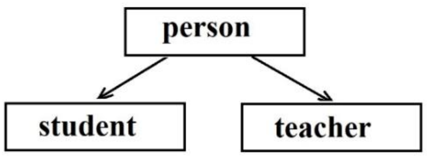


Specialization and generalization:

Specialization means dividing entity into **subsets**. While generalization means representing subsets (that have common features) by **superset**.

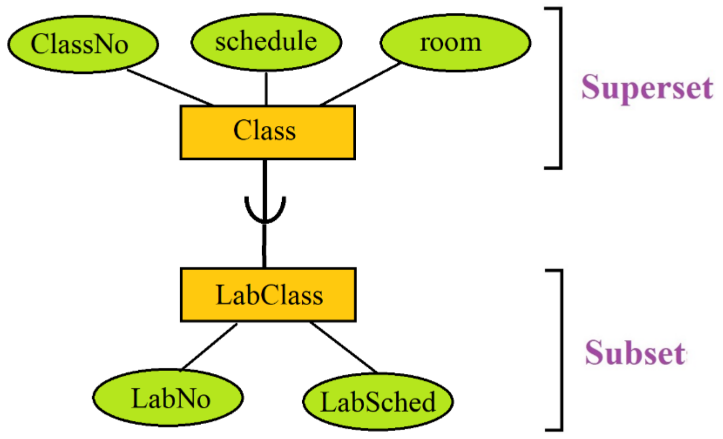


Generalization



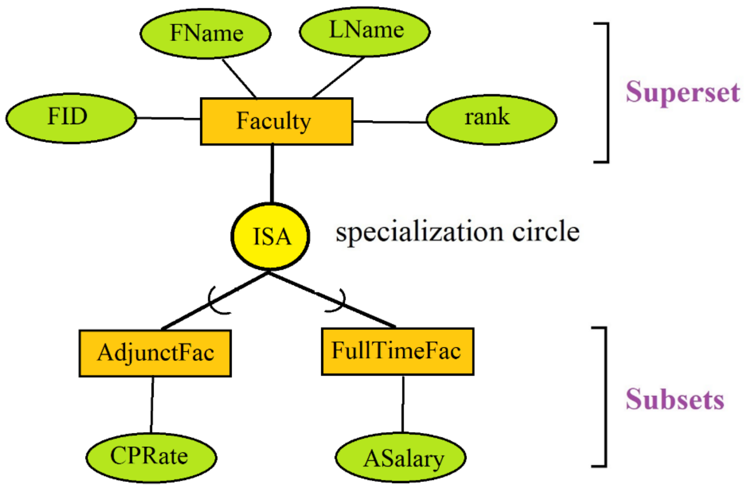
Specialization

Ex.1 Specialization with one subset



Here the LabClass takes all attributes of Class and adds to it new attributes.

Ex.2 Specialization with two subsets



Here the Faculty has been specialized into two subsets. The circle is called the specialization circle (ISA relationship).

CHAPTER 4: The Relational Model

Relation schema R:

Relation schema R consists of attributes and domains. Relation schemas can be represented by name of relation with attribute names. For example, Student table can be represented by:

Student (SID, LName, FName, major, credits)

SID is underlined to indicate that attribute is the primary key in the relation schema. Each attribute has single valued and the order of attributes is important.

Degree and Cardinality:

Degree of the relation is the number of columns (for example the degree of Student is 5 because it has 5 attributes), while the number of rows in a table is the cardinality of the relation.

Relation Keys:

Relation Key is a specific attribute (or combination of attributes) that makes every row in the table is distinct. There is primary key for every relation. When an attribute is not the primary key of the relation but for different relation, this key is called foreign key (advantage of foreign key is to represent the connections between relations). For example:

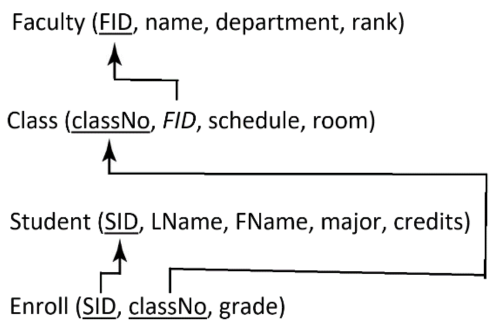
Employee (empId, LName, FName, *departmentName*)

Department (departmentName, office)

Here, departmentName, which is a primary key in the Department table is foreign key in Employee. departmentName has unique values in the Department table, but it is not unique in the Employee table.

Representing Relational Database Schemas

Foreign key is clarified by drawing arrows from them to the primary keys they refer to. For example:


Faculty (FID, name, department, rank)
Class (classNo, FID, schedule, room)
Student (SID, LName, FName, major, credits)
Enroll (SID, classNo, grade)

Explanation: The attribute FID is the primary key of Faculty relation, but in the same time it is a foreign key in Class relation. Also, the attribute classNo is the primary key of Class relation, but in the same time it is a foreign key in Enroll relation.

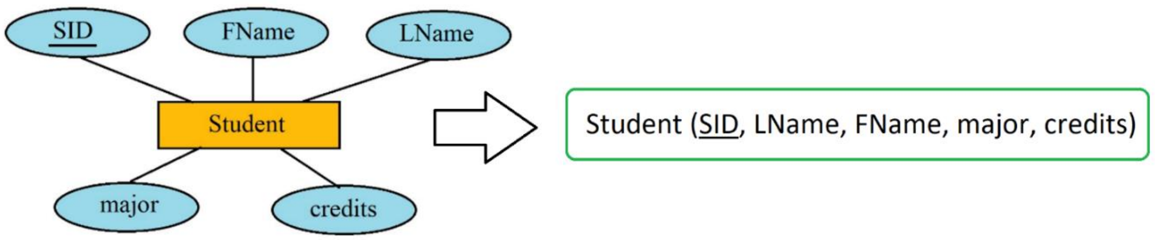
When we want to discriminate between two of SID, we use the name of relation followed by a period and then followed by the attribute name (For example, Student.SID and Enroll.SID).

Mapping an E-R Model to a Relational Schema

Strong entity mapping

Case 1: For simple attribute, strong entity sets can be mapped into tables that have column for each single valued. For example:

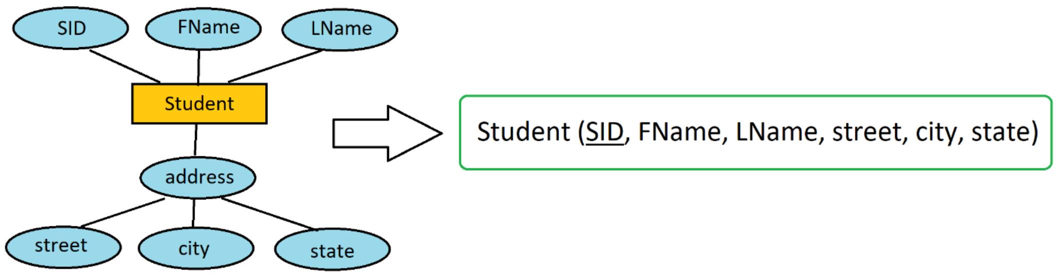
(Strong Entity With Simple Attributes)



Student (SID, LName, FName, major, credits)

Case 2: For composite attributes, new column must be created for each sub-attribute with not representing the composite attribute. For example:

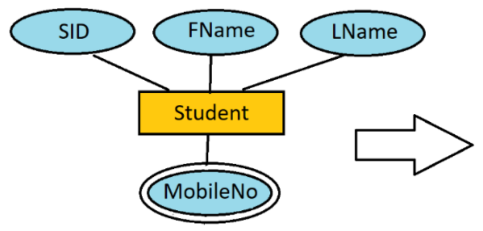
(Strong Entity With Composite Attributes)



Student (SID, FName, LName, street, city, state)

Case 3: For each multi-valued attribute, separate table must be created that having the primary key of the table, along with a column for the multivalued attribute. For example:

(Strong Entity With Multi Valued Attributes)

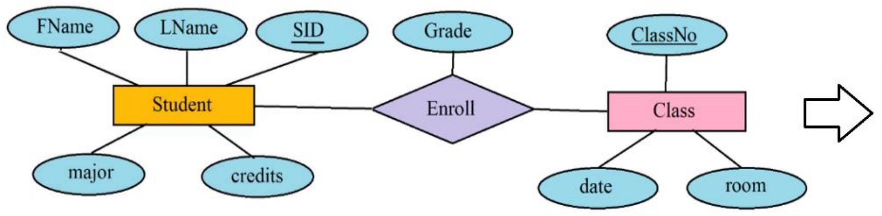


Student (SID, FName, LName)
Mobile (SID, mobileNo)

Relationships mapping

Case 4: For descriptive attributes, separate table must be created that having descriptive attributes with the primary keys of the both entities. For example:

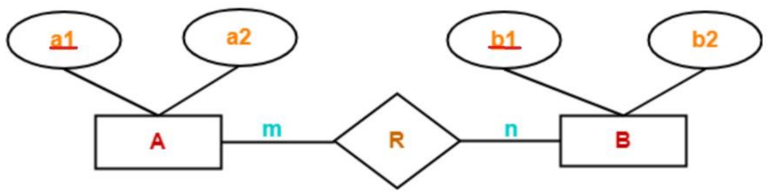
(Relationship with descriptive attributes)



Enroll (SID, ClassNo, Grade)

Case 5: For many-to-many binary relationships, new relationship table must be created that consists of the primary keys of the related entities, along with any descriptive attributes (if any) of the relationship. For example:

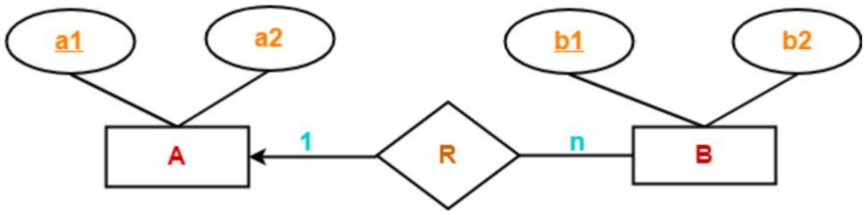
(For Binary Relationship m:n)



A (a1 , a2)   R (a1 , b1)   B (b1 , b2)

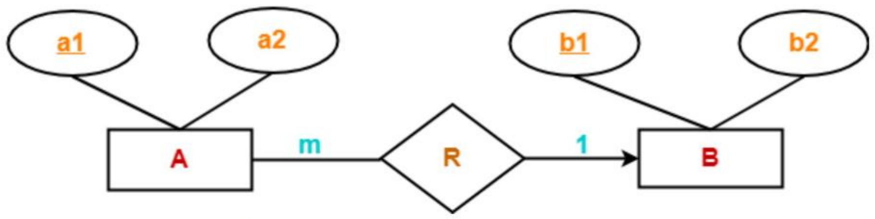
Case 6: For one-to-many (or many-to-one) binary relationships, the primary key of the one side must be placed in the table of the many side and consider it as a foreign key. For example:

(For Binary Relationship 1:n)



A (a1 , a2)   R (a1 , b1 , b2)   **a1: foreign key**

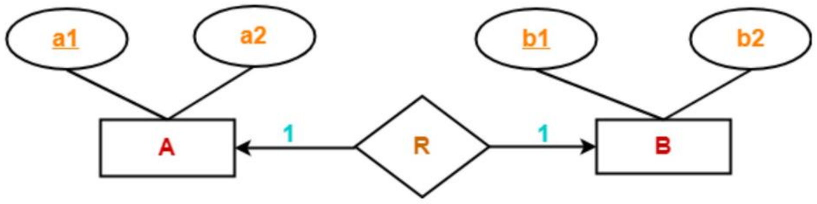
(For Binary Relationship m:1)



b1: foreign key   R (a1 , a2 , b1)   B (b1 , b2)

Case 7: For one-to-one binary relationships, the primary key of the one side must be placed in the table of the other side and consider it as a foreign key. For example:

(For Binary Relationship 1:1)



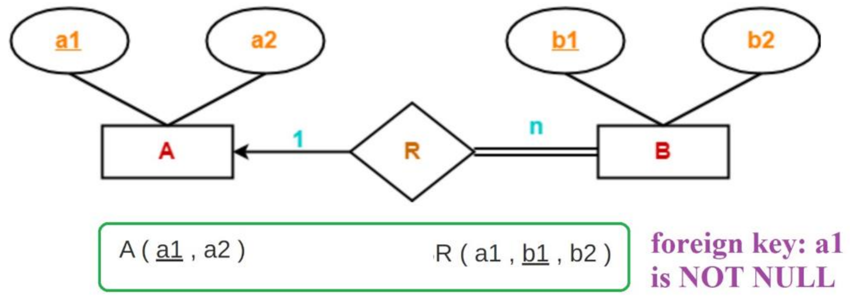
option:1	option:2
R (a1 , a2 , b1)	A (a1 , a2)
B (b1 , b2)	R (a1 , b1 , b2)

b1: foreign key   **a1: foreign key**

Total participation mapping

Case 8: For 1 side total participation, the primary key of the one side must be placed in the table of the many side and consider it as a foreign key. For example:

(for total participation constraint from one side)

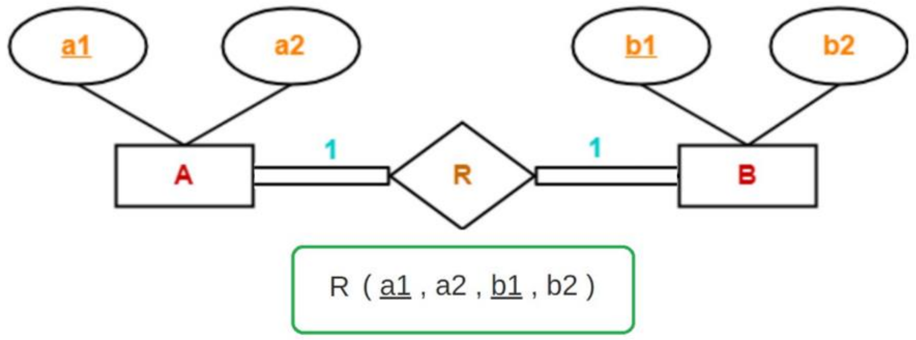


A (a1 , a2)   R (a1 , b1 , b2)

foreign key: a1
is NOT NULL

Case 9: For 2 sides total participation, new table must be created (replacing the old two tables) and includes all attributes. For example:

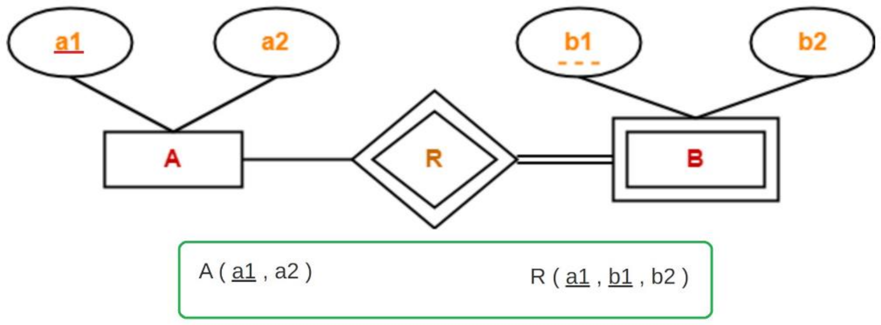
(for total participation constraint from both sides)



R (a1 , a2 , b1 , b2)

Weak entity mapping

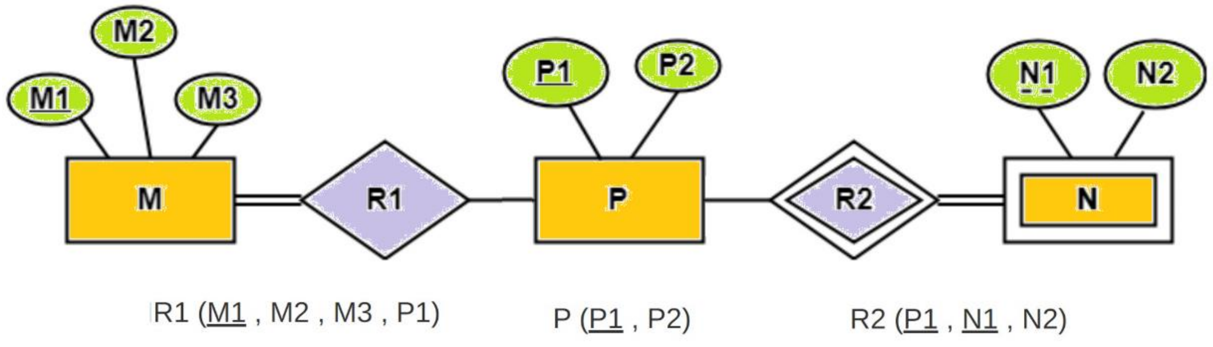
Case 10: Weak entity always appears with total participation constraint. For each weak entity, the table of weak entity must include the primary key of strong entity. For example:



A (a1 , a2)   R (a1 , b1 , b2)

Examples

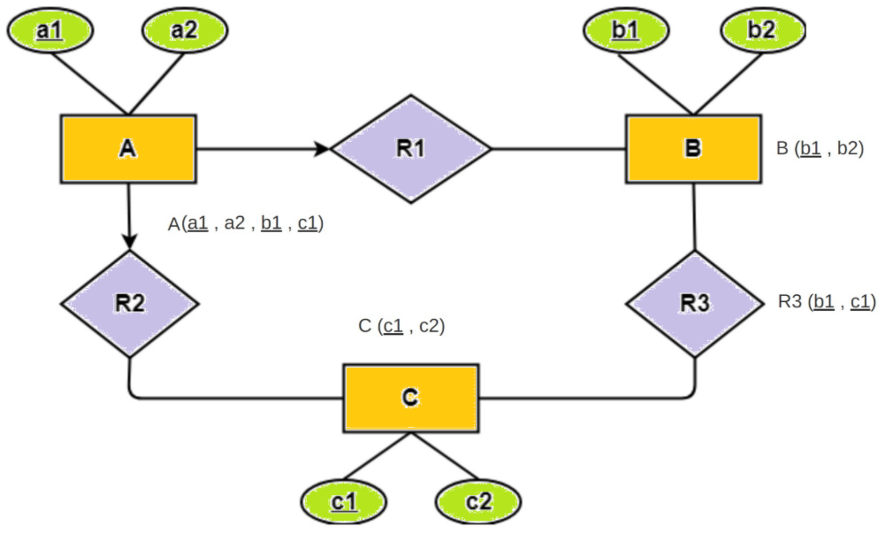
Example 1:



Explanation:

- The primary key P1 is used in relation M as a foreign key (Case 8)
- The primary key P1 is used in relation N as a foreign key (Case 10)

Example 2:



Explanation:

- The primary keys c1 (from relation C) and b1 (from relation B) are used in relation A as foreign keys.
- The primary keys c1 (from relation C) and b1 (from relation B) are also used to create a new relation table R3 (Case 5).