Entity-Relationship Model

Dr. Odelu Vanga

Indian Institute of Information Technology Sri City

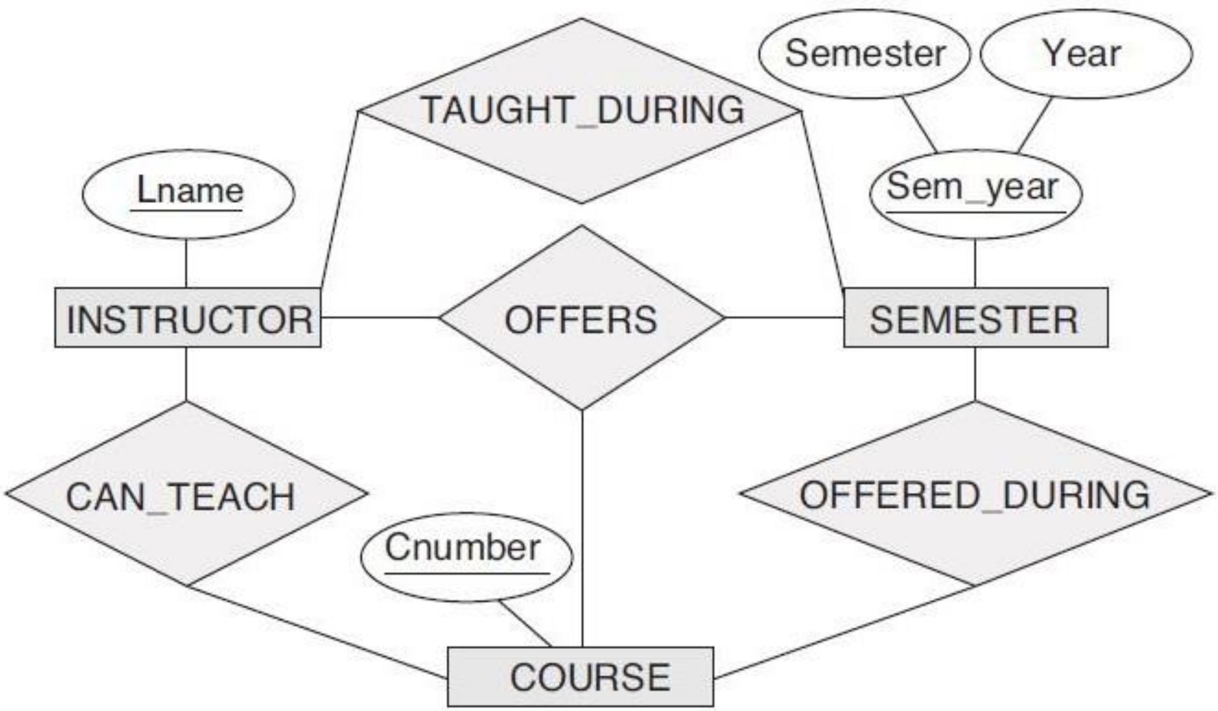
http://www.iiits.ac.in/people/regular-faculty/dr-odelu-vanga/

Outline

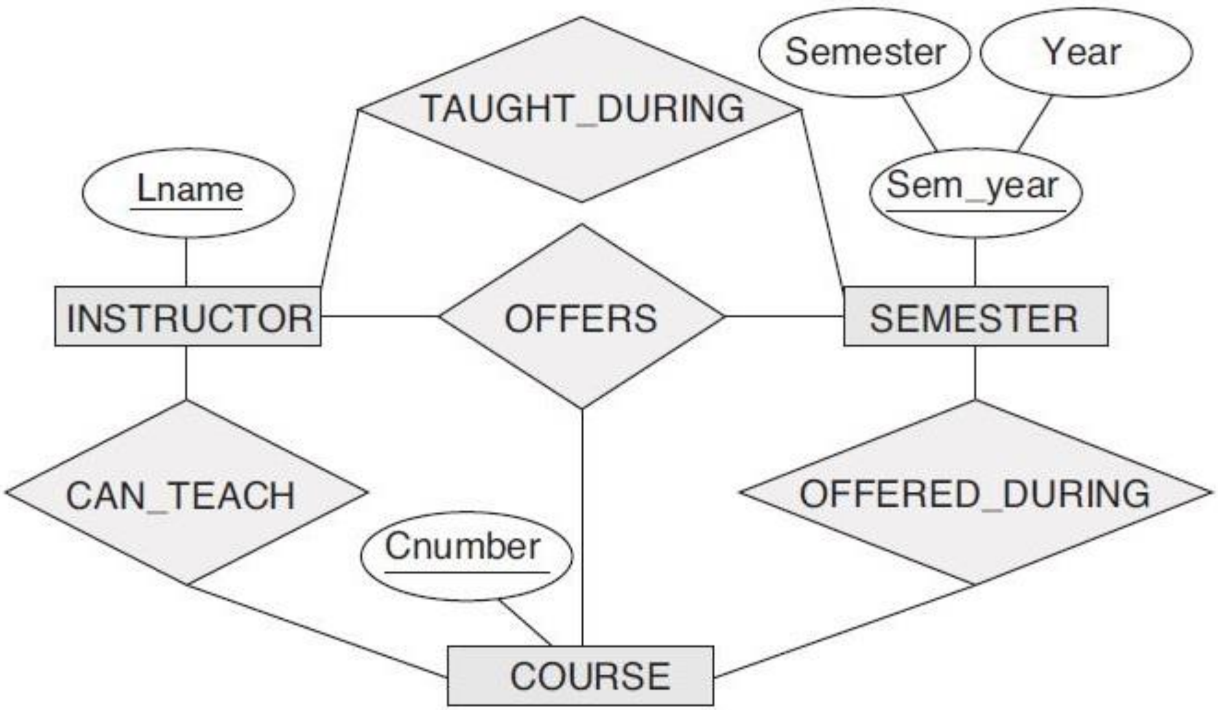
- **Binary Relationship**
- **N-ary Relationship**
- **Examples:**
    1. **University**
    2. **Supply**
- **Week Entity Sets**

N-ary Relationship

- A relationship type *R* of degree *n* will have *n* edges in an ER diagram, one connecting *R* to each participating entity type.
- Binary relationship – degree 2
- Ternary relationship – degree 3



Ternary vs Binary Relationship



A relationship instance (*i*, *s*, *c*) whenever INSTRUCTOR *i* offers COURSE *c* during SEMESTER *s*.

- a relationship instance (*i*, *s*, *c*) should not exist in OFFERS *unless* an instance (*i*, *s*) exists in TAUGHT_DURING, an instance (*s*, *c*) exists in OFFERED_DURING, and an instance (*i*, *c*) exists in CAN_TEACH.
- However, the reverse is not always true; we may have instances (*i*, *s*), (*s*, *c*), and (*i*, *c*) in the three binary relationship types with no corresponding instance (*i*, *s*, *c*) in OFFERS.

Ternary vs Binary Relationship

Instructor (i)	Semester (s)	Course (c)
Akhil	Sem-I	C1
Ram	Sem-II	C2
Ravi	Sem-III	C1
Akhil	Sem-III	C3

OFFERS

A relationship instance (*i*, *s*, *c*) – means INSTRUCTOR *i* offers COURSE *c* during SEMESTER *s*

- (Akhil, Sem-I, C1) ϵ OFFERS implies (exists)

  (Akhil, Sem-I) ϵ **TAUGHT_DURING**, ------------(1)

  (Sem-I, C1) ϵ **OFFERED_DURING**, and --------(2)

  (Akhil, C1) ϵ **CAN_TEACH** ---------------------(3)

Whether converse true ?

That is, given (1), (2), (3) can we say the relation (i, s, c) ?

Ternary vs Binary Relationship

Instructor (i)	Semester (s)	Course (c)
Akhil	Sem-I	C1
Ram	Sem-II	C2
Ravi	Sem-III	C1
Akhil	Sem-III	C3

OFFERS

A relationship instance (*i*, *s*, *c*) – means INSTRUCTOR *i* offers COURSE *c* during SEMESTER *s*

Whether converse is true ?

That is, given (1), (2), (3) can we say the relation (i, s, c) ?

Suppose

(Akhil, Sem-III) ϵ **TAUGHT_DURING**, ------------(1)

(Sem-III, C1) ϵ **OFFERED_DURING**, and --------(2)

(Akhil, C1) ϵ **CAN_TEACH** ----------------------(3)

implies (exists)

(Akhil, Sem-III, C1) ϵ **OFFERS** **….?**

Ternary vs Binary Relationship

Instructor (i)	Semester (s)	Course (c)
Akhil	Sem-I	C1
Ram	Sem-II	C2
Ravi	Sem-III	C1
Akhil	Sem-III	C3

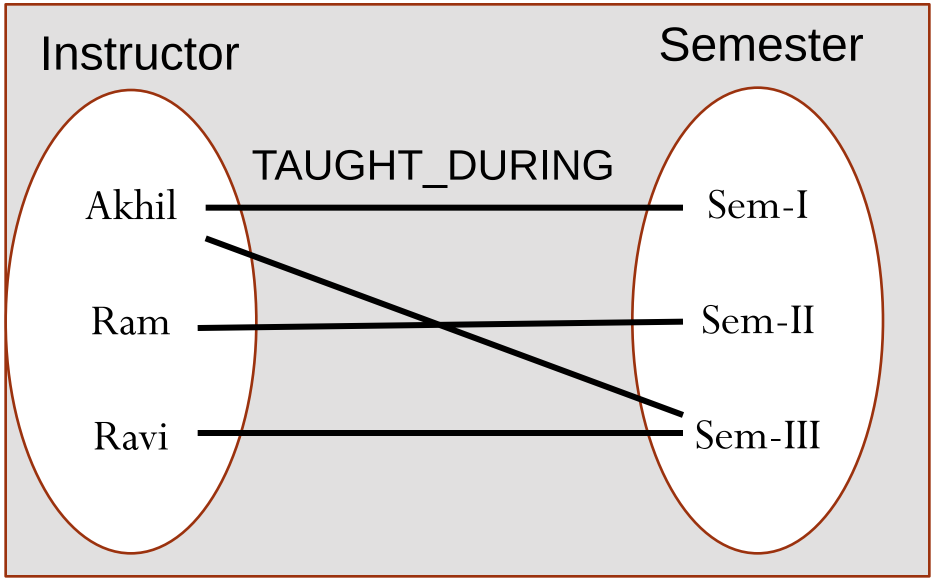
OFFERS

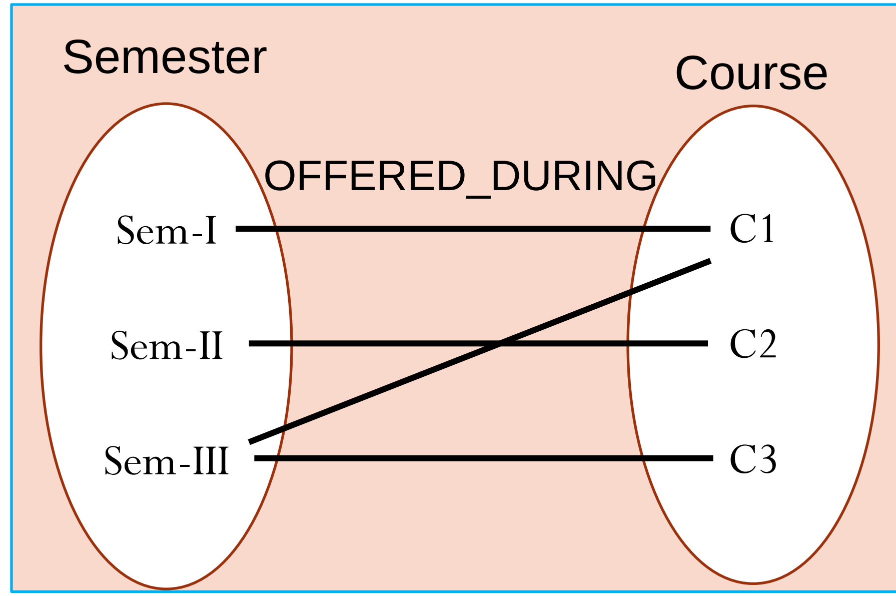
A relationship instance (*i*, *s*, *c*) – means INSTRUCTOR *i* offers COURSE *c* during SEMESTER *s*

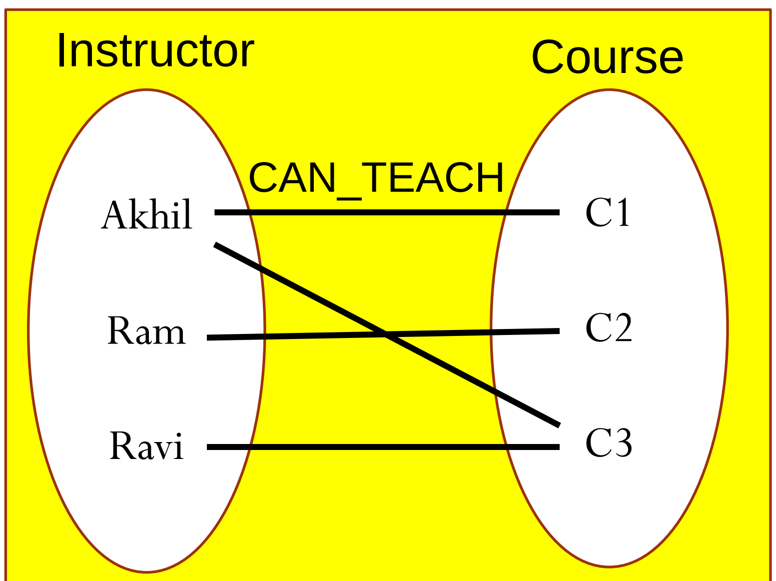
- Based on the meanings of relationships, we can infer the instances of
  - TAUGHT_DURING
  - OFFERED_DURING

  from the instances in OFFERS.
- But, we cannot infer the instances of CAN_TEACH.
- Therefore, TAUGHT_DURING and OFFERED_DURING are redundant, and can be left out.

Constraints on Ternary







(Akhil, Sem-III) ϵ **TAUGHT_DURING** ---(1)

(Sem-III, C1) ϵ **OFFERED_DURING** ----(2)

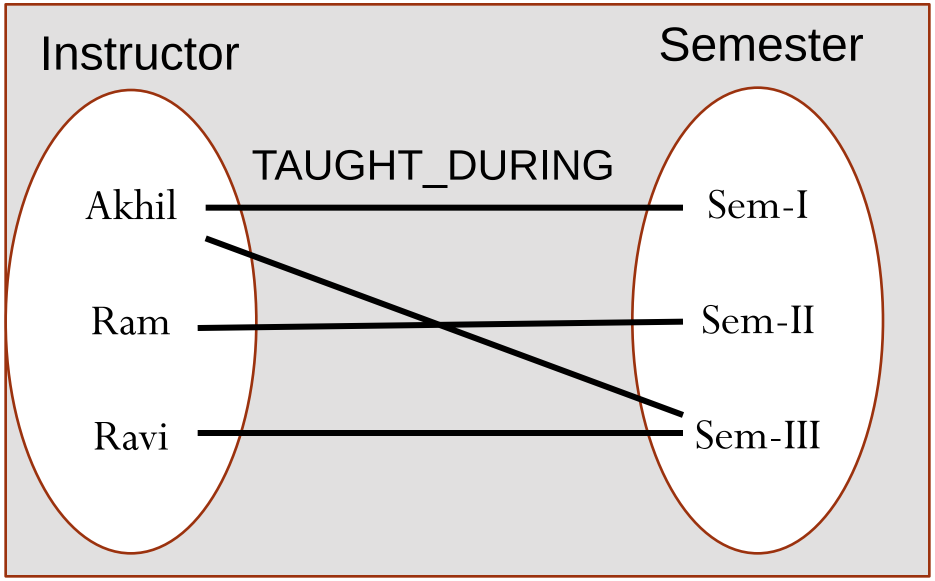
(Akhil, C1) ϵ **CAN_TEACH** --------------(3)

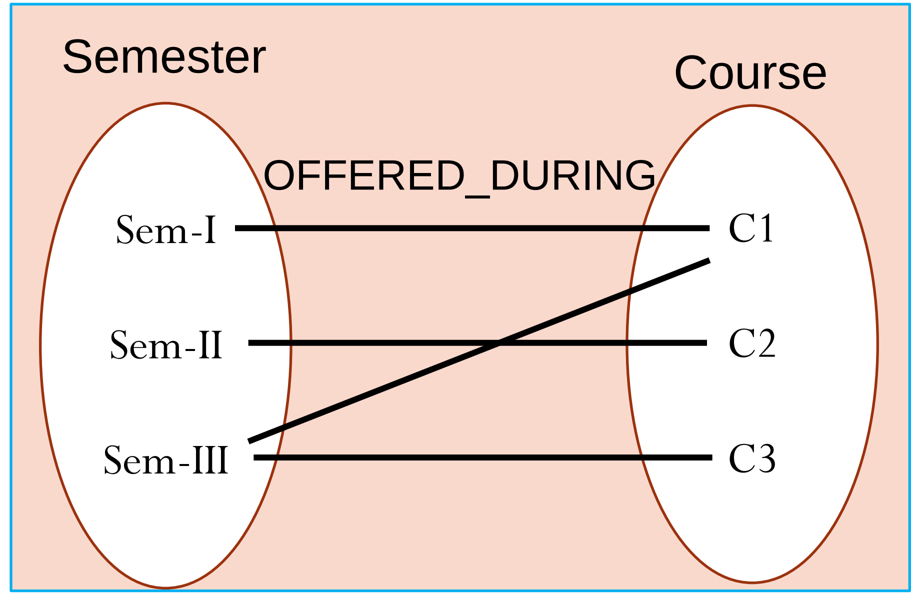
If CAN_TEACH relationship is 1:1

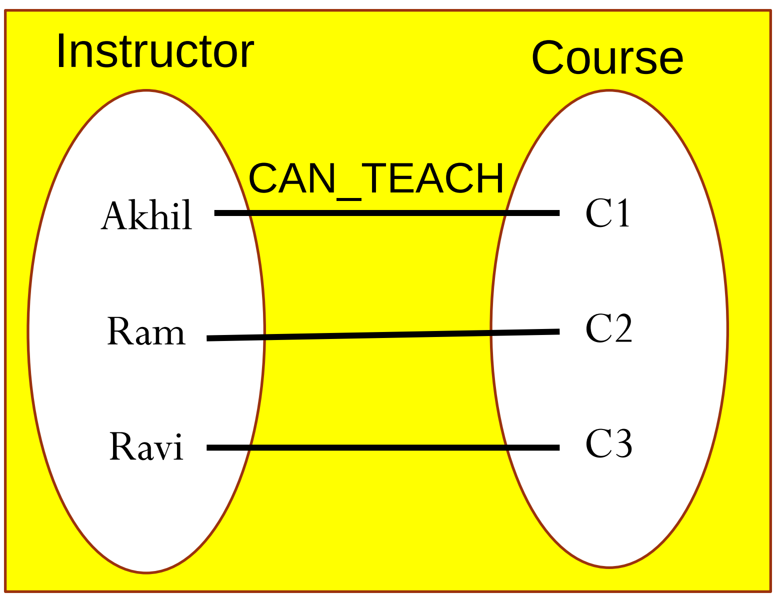
Then ternary can be left out

Because: (i,s), (i, c), (c, s) implies (i, s, c)

Constraints on Ternary







(Akhil, Sem-III) ∈ **TAUGHT_DURING** ---(1)

(Sem-III, C1) ∈ **OFFERED_DURING** ----(2)

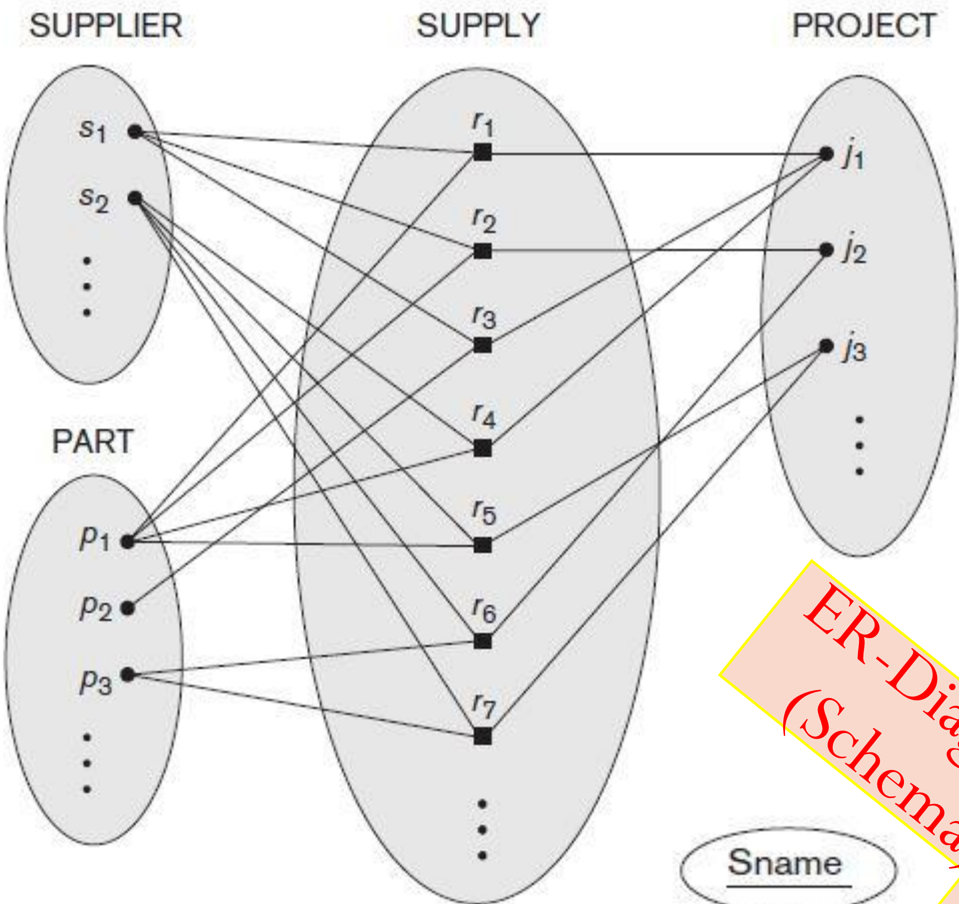
(Akhil, C1) ∈ **CAN_TEACH** --------------(3)

If CAN_TEACH relationship is 1:1

Then ternary can be left out

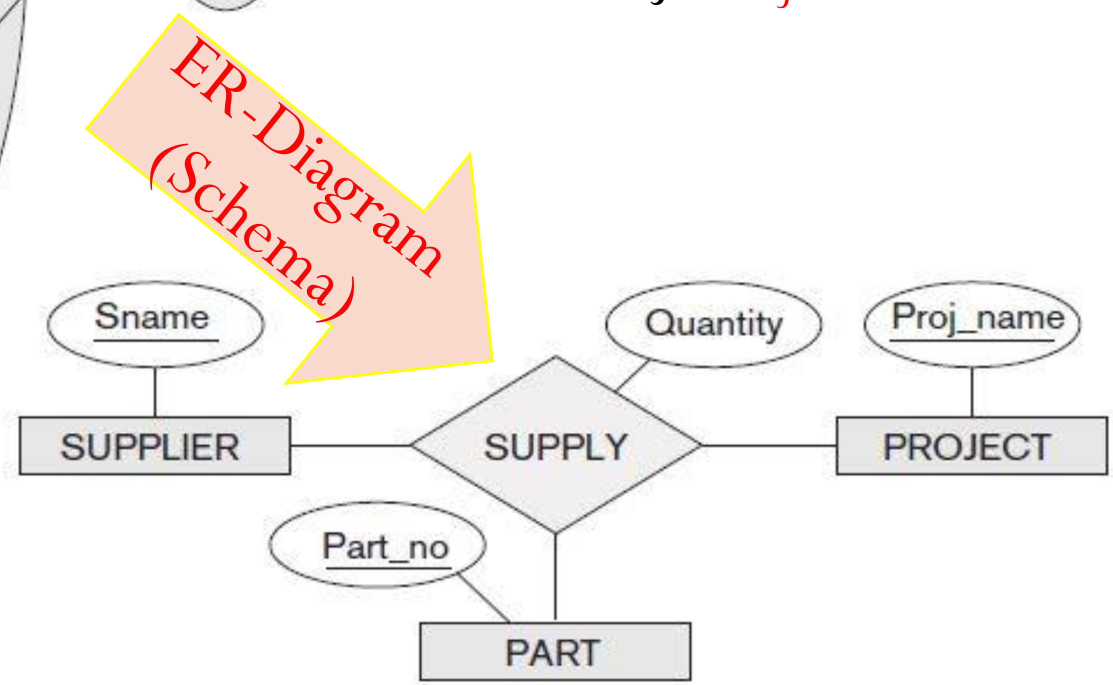
Because: (i,s), (i, c), (c, s) implies (i, s, c)

SUPPLY relation

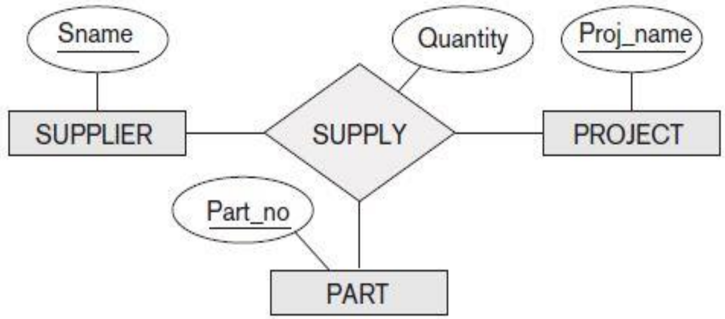


The relationship set of SUPPLY is a set of relationship instances (s, j, p) – that is,

- a SUPPLIER s who is currently
- - supplying a PART p
- - to a PROJECT j.



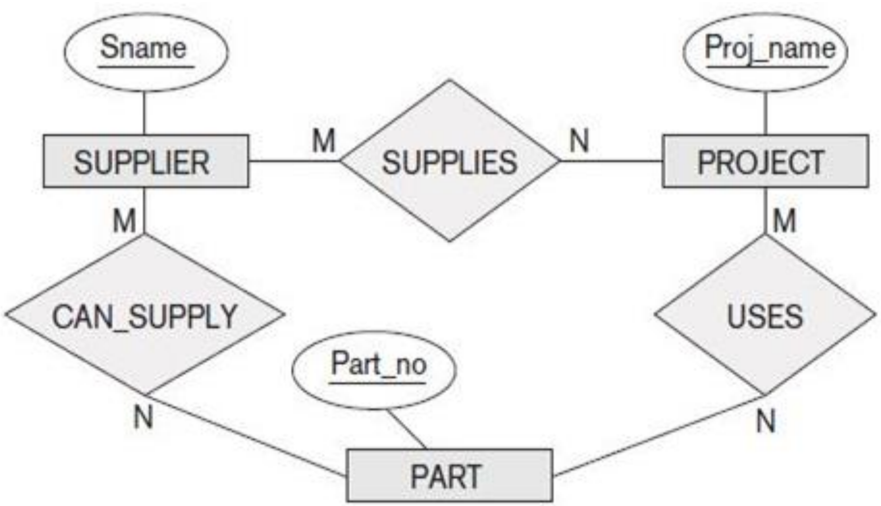
SUPPLY relation



(a) Ternary Relation

The relationship set of SUPPLY is a set of relationship instances (s, j, p) – that is,

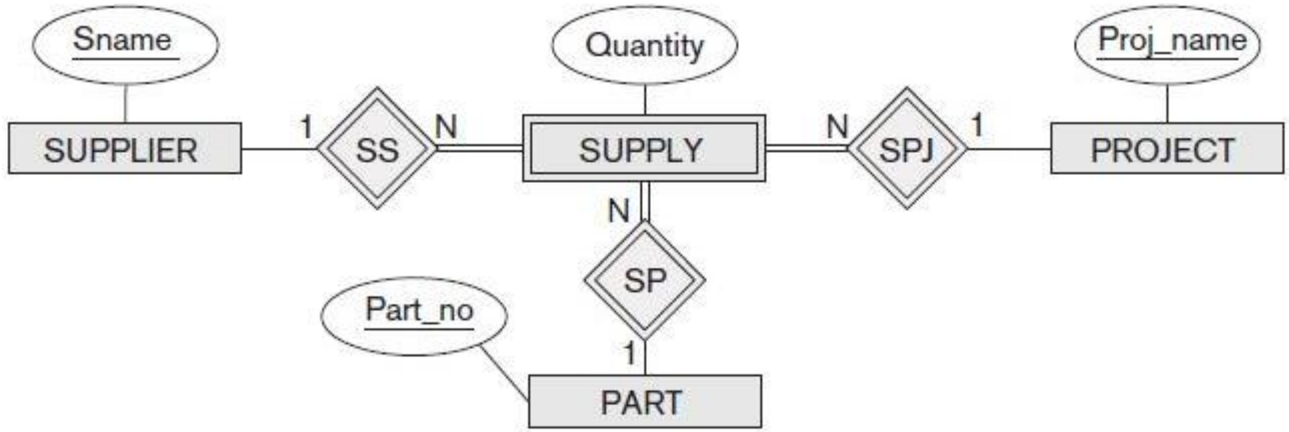
- a SUPPLIER s who is currently
- - supplying a PART p
- - to a PROJECT j.



(b) Binary Relation

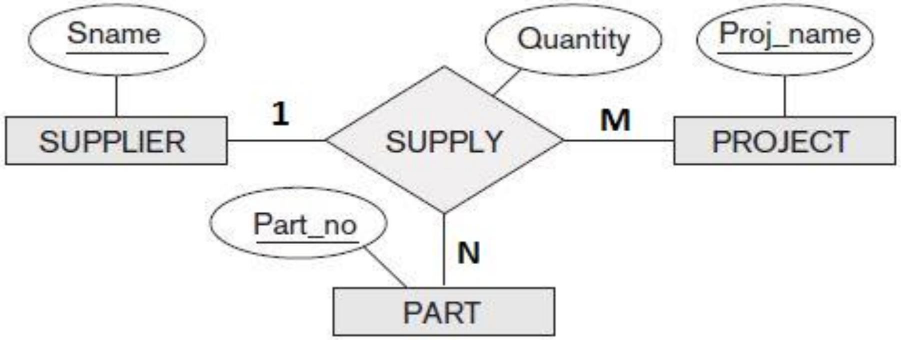
- CAN_SUPPLY, between SUPPLIER and PART, includes an instance (s, p) whenever supplier s *can supply* part p (to any project);
- USES, between PROJECT and PART, includes an instance (j, p) whenever project j uses part p;
- SUPPLIES, between SUPPLIER and PROJECT, includes an instance (s, j) whenever supplier s supplies *some part* to project j.

SUPPLY represented as a weak entity type



- SUPPLY represented as a weak entity type, with no partial key and with three identifying relationships
- SUPPLIER, PART, and PROJECT are together owner entity types
- An entity in weak entity type SUPPLY is identified by the combination of its three owner entities from SUPPLIER, PART, and PROJECT.
- It is also possible to represent the ternary relationship as a regular entity type by introducing an artificial or surrogate key.
  - In this example, a key attribute Supply_id could be used for the supply entity type, converting it into a regular entity type.

Constraints on Higher-Degree Relationships



- **Suppose constraint exists that, for a particular project-part combination, only one supplier will be used.**
- This specifies the constraint that a particular (j, p) combination can appear at most once in the relationship set because each such (PROJECT, PART) combination uniquely determines a single supplier.
- Hence, any relationship instance (s, j, p) is uniquely identified in the relationship set by its (j, p) combination, which makes (j, p) a key for the relationship set.
- In this notation, the participations that have a **1** specified on them are not required to be part of the identifying key for the relationship set.
- If all three cardinalities are M or N, then the key will be the combination of all three participants.

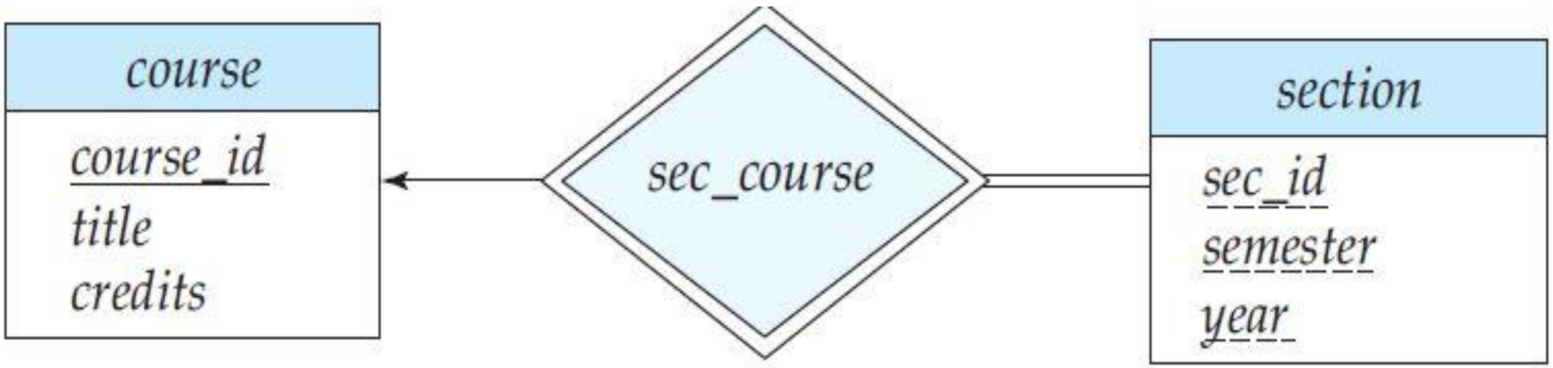
Weak and Strong Entity Sets

- An entity set that does not have sufficient attributes to form a primary key is termed a **weak entity set**.
- An entity set that has a primary key is termed a **strong (regular) entity set**.

  course: with attributes (*course_id*, *title*, *credits*)

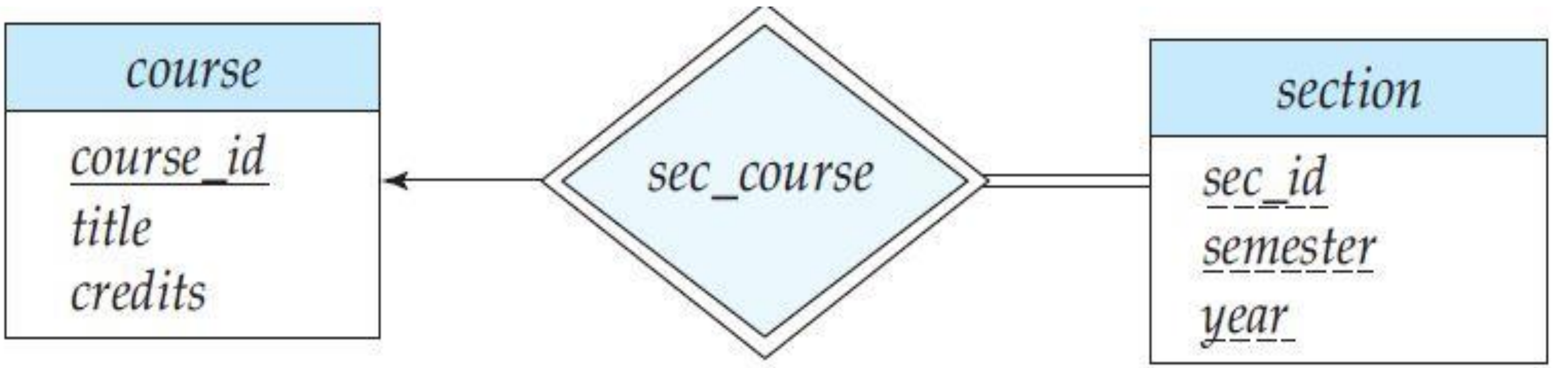
  section: with attributes (*course_id*, *sec_id*, *semester*, *year*)
- Suppose create a relationship-set *sec_course* between entity sets *section* and *course*.
- For a weak entity set to be meaningful, it must be associated with another entity set, called the **identifying** or **owner entity set**.
- Every weak entity must be associated with an identifying entity; that is, weak entity set is said to be **existence dependent** on the identifying entity set.
- The identifying entity set is said to **own** the weak entity set that it identifies.
- The relationship associating the weak entity set with the identifying entity set is called the **identifying relationship**.

Weak Entity Set



- **Identifying entity set** for *section* is *course*
- Relationship *sec_course* : associates *section* entities with their corresponding *course* entities, is the **identifying relationship**
- A weak entity type normally has a **partial key (discriminator)**, which is the attribute that can uniquely identify weak entities that are *related to the same owner entity*.
- The primary key of a weak entity set is formed by the primary key of the identifying entity set, plus the weak entity set's discriminator.

Weak Entity Set



- A weak entity type always has a ***total participation constraint*** (existence dependency) with respect to its identifying relationship because a weak entity cannot be identified without an owner entity.

Whether every existence dependency results in a weak entity type ?

- **DRIVER_LICENSE entity cannot exist unless it is related to a PERSON entity, even though it has its own key (License_number) and hence is not a weak entity.**

THANKS